Investigations on MAC and Link Layer for a wireless PROFIBUS over IEEE 802.11

von Diplom-Informatiker
Andreas Willig
aus Berlin

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Tag der wissenschaftlichen Aussprache: 17. Mai 2002

Berlin 2002
D 83

Contents

1 Introduction **20**

1.1 Structure of the Thesis . . . 24

2 Industrial Communication Systems **26**

2.1 Requirements for Industrial Communication Systems / Field Buses . . . 27

2.2 Architectural Characteristics . . . 28

2.3 Popular Systems . . . 28

2.3.1 Systems using Token-Passing-MAC's . . . 29

2.3.2 Other Systems . . . 29

2.3.3 MAP/MMS . . . 30

3 Wireless LANs / IEEE 802.11 **31**

3.1 Wireless LANs . . . 31

3.1.1 Basics . . . 31

3.1.2 WLAN Properties Important for MAC Design . . . 33

3.1.3 WLAN Standards / Systems . . . 35

3.2 IEEE 802.11 . . . 36

3.2.1 Architecture . . . 37

3.2.2 PHY Layer . . . 37

3.2.3 MAC Layer . . . 40

4 PROFIBUS and Wireless PROFIBUS **45**

4.1 The PROFIBUS Fieldbus System . . . 46

4.1.1 Architecture . . . 46

4.1.2 Physical Layer . . . 47

4.1.3 Link Layer Services . . . 48

4.1.4 MAC- and Data Link Protocol . . . 52
4.1.5 Important Properties of the PROFIBUS . . . 56
4.2 Wireless Industrial Communication Systems . . . 58
4.2.1 Integrated Scenario: General Considerations . . . 59
4.3 Integrated Scenario for Wireless PROFIBUS . . . 60
4.3.1 System under Study and Realtime Performance Measures . . . 60
4.4 Wireless PROFIBUS over IEEE 802.11 MAC . . . 64
4.4.1 DCF-based approaches . . . 64
4.4.2 PCF-based approaches . . . 65
4.4.3 SRD service handling . . . 65
4.4.4 Final Remarks . . . 66
4.5 Related Work . . . 67
4.5.1 Overview of other Work in Wireless Fieldbus Systems . . . 67
4.5.2 Real-Time Data Transmission with IEEE 802.11 . . . 69

5 Behavior of the PROFIBUS Protocol under Link Errors 72
5.1 PROFIBUS over Error Prone Links . . . 72
5.1.1 Major Causes for Ring Instability . . . 73
5.1.2 Ring Stability Metrics . . . 74
5.1.3 Analytical PROFIBUS Ring Membership Model . . . 75
5.1.4 Simulation Results . . . 82
5.1.5 Improvements . . . 91
5.2 PROFIBUS over Wireless Links . . . 99
5.3 Related Work . . . 101
5.4 Conclusions . . . 101

6 Error Behavior of Wireless Channels and its Modeling 104
6.1 Sources of Errors . . . 106
6.1.1 Path Loss . . . 107
6.1.2 Multipath Fading . . . 107
6.2 Measurement Setup . . . 110
6.2.1 IEEE 802.11 / PRISM I PHY . . . 110
6.2.2 Measurement Setup . . . 110

6.2.3 Format of the Generated Packet-Stream . . . 112
6.3 Measurement Evaluation Methodology . . . 113
6.3.1 Indicator Sequences . . . 113
6.3.2 Trace Evaluation . . . 114
6.4 Measurement Parameters and Environment . . . 115
6.4.1 Environment . . . 115
6.4.2 Parameters . . . 116
6.5 Measurement Results . . . 119
6.5.1 Packet-Related Phenomena . . . 119
6.5.2 Packet Losses . . . 120
6.5.3 Positions of Bit Errors . . . 125
6.5.4 Mean Bit Error Rates . . . 126
6.5.5 Burst Length Statistics . . . 128
6.5.6 Error Densities and Error Clustering . . . 131
6.6 Review of other Measurement Studies . . . 134
6.7 Stochastic Modeling of Bit- and Packet-Errors . . . 137
6.7.1 Overview of Common Stochastic Models . . . 137
6.7.2 Bipartite Model for Generating Indicator Sequences . . . 140
6.7.3 Comparison of Different Models . . . 142
6.8 Conclusions . . . 146
6.8.1 Summary of Measurement Results . . . 146
6.8.2 Stochastic Modeling . . . 147
6.8.3 Overall Channel Model . . . 148
6.8.4 Consequences for Design of MAC and Link-Layer Protocols . . . 149

7 Polling Protocols for Wireless PROFIBUS 151
7.1 System under Study . . . 153
7.1.1 Description of the System . . . 154
7.2 Polling-based Protocols . . . 158
7.2.1 k-limited Round-Robin . . . 159
7.2.2 Functional Repolling Framework . . . 160
7.2.3 Relaying . . . 162
7.2.4 Adaptive Functional Repolling . . . 165

7.3 Realtime-Performance Results . . . 166
7.3.1 Method of Investigation . . . 166
7.3.2 Comparison of Round-Robin with PROFIBUS . . . 168
7.3.3 Comparison of the Polling-protocol Modifications with Round-Robin . . . 179
7.4 Polling-based MACs for wireless PROFIBUS . . . 202
7.4.1 Integration Issues . . . 203
7.4.2 Semantics . . . 204
7.5 Related Work . . . 204
7.6 Discussion and Future Work . . . 205

8 Conclusions and Outlook **209**

A Main Characteristics of the Bipartite Model **212**
A.1 Asymptotic Behavior . . . 212
A.2 Distribution of Generated Burst Lengths . . . 214
A.3 Correlation Properties . . . 215

B Applicability of Simple FEC Schemes to the Measurement Traces **217**

List of Figures

2.1 Hierarchy of information flows in manufacturing applications (from [68, p. 13]) 26

3.1 Hidden terminal scenario 33

3.2 Exposed terminal scenario 34

3.3 DSSS PHY PPDU Format 39

3.4 DSSS PHY Schematic 40

4.1 PROFIBUS protocol stack 47

4.2 Interactions of SDA service 50

4.3 Interactions of SDN service 51

4.4 Interactions of SRD service 51

4.5 PROFIBUS frame formats 57

4.6 Integrated PROFIBUS LAN 59

5.1 A single stations life cycle 75

5.2 Logical structure of PROFIBUS simulation model 82

5.3 $\bar{N}(3600)$ and $\bar{N}$ vs. BER (independent errors) 84

5.4 $\bar{M}(3600)$ and $\bar{M}$ vs. BER (independent errors) 84

5.5 $\bar{M}(3600)$ vs. BER (independent errors) 86

5.6 $\bar{N}(3600)$ vs. BER (independent errors) 87

5.7 Distribution function C(s) for different BER's 88

5.8 $\bar{N}(3600)$ vs. BI for m = 0.001 (Gilbert errors) 88

5.9 $N(t)$ vs. time (Gilbert errors) 89

5.10 MSOT vs. gap factor (independent errors) 89

5.11 CSOT vs. gap factor (independent errors) 90

5.12 $\bar{M}(3600)$ vs. BER (independent errors) 94

5.13 $\bar{N}(3600)$ vs. BER (independent Errors) 94

5.14 Sample coefficients of variation for N vs. BER (independent errors) 95

5.15 $\bar{N}(3600)$ vs. BI for m = 0.001 (Gilbert errors) 95

5.16 $\bar{N}(10000)$ vs. BER (independent errors) with 10 masters and $\approx 36\%$ load 96

5.17 $\bar{N}(10000)$ vs. BI for m = 0.001 (Gilbert errors) with 10 masters and $\approx 36\%$ load . . . 96

5.18 $\bar{M}(10000)$ vs. BER (independent errors) with 4 and 10 masters and $\approx 36\%$ load . . . 97

5.19 $\bar{M}(10000)$ vs. BI $m = 0.001$ (Gilbert errors) with 4 and 10 masters and $\approx 36\%$ load . 97

5.20 $N(t)$ vs. time (Gilbert errors, both protocol improvements) 98

5.21 $\bar{N}(3600)$ vs. PLR (independent packet losses) and no bit errors 100

5.22 $\bar{M}(3600)$ vs. PLR (independent packet losses) and no bit errors 101

5.23 $\bar{N}(3600)$ vs. PLR (independent packet losses) and BER of 10^{-3} 102

5.24 $\bar{M}(3600)$ vs. PLR (independent packet losses) and BER of 10^{-3} 103

6.1 Multipath fading 108

6.2 Measurement setup 111

6.3 Position of our setup within the building 116

6.4 Setup of PTZ measurement 117

6.5 Packet loss rate vs. trace number for **longterm1** measurement 120

6.6 Position of portal crane (0=close proximity, 1=short distance, 2=longer distance) for **longterm1**-measurement 121

6.7 Cumulative distribution functions of packet loss burst lengths and packet loss-free burst lengths (for COMP-PLIS) 122

6.8 Conditional probability that packet $n+k$ is lost given that packet n is lost (for COMP-PLIS) 123

6.9 Autocovariance function of packet loss burst lengths for the compound loss sequence and $k_0 = 1$ for **longterm1** measurement (for COMP-PLIS) 124

6.10 Autocovariance function of packet loss-free burst lengths for the compound loss sequence and $k_0 = 1$ for **longterm1** measurement (for COMP-PLIS) 124

6.11 Positions of bit errors, **factorial** trace 83 (QPSK modulation, with scrambling, 6012 bytes packet size) 125

6.12 Positions of bit errors, **factorial** trace 37 (5.5 MBit/s CCK modulation, no scrambling, 2016 bytes packet size) 126

6.13 Mean bit error rate vs. trace number for remaining traces (logarithmic scale) 127

6.14 Mean bit error rate vs. trace number for **longterm1** measurement (logarithmic scale) 127

6.15 Mean error burst length vs. k_0 for selected BPSK traces BEIS (**factorial** measurement) 129

6.16 Mean error burst length vs. k_0 for selected QPSK traces BEIS (**factorial** measurement) 129

6.17 Mean error burst length vs. trace number for selected k_0 (for BEIS of **longterm1** measurement) 130

6.18 Coefficients of variation (CoV) of error burst lengths and error-free burst lengths vs. k_0 for selected BPSK traces BEIS (**factorial** measurement) 130

6.19 Coefficients of variation (CoV) of error burst lengths and error-free burst lengths vs. k_0 for selected QPSK traces BEIS (**factorial** measurement) 131

6.20 Cumulative distribution function for the error densities of all BPSK traces without scrambling 132

6.21 Cumulative distribution function for the error densities of all QPSK traces without scrambling 133

6.22 A sample bipartite Markov chain 141

6.23 Conditional probability that packet $n+k$ is erroneous given that packet n is erroneous 144

6.24 Conditional probability that packet $n+k$ is erroneous given that packet n is erroneous 145

7.1 Logical structure of polling simulation model 167

7.2 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round-robin bounds k 170

7.3 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round-robin bounds k 171

7.4 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, independent error model and different round-robin bounds k 171

7.5 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, complex error model and different round-robin bounds k 172

7.6 Fraction of time, that all stations are member of the logical PROFIBUS ring over all different gap factors and T_{TTRT} values vs. station number N for different error models (10% low priority load) 172

7.7 Remaining bandwidth for low priority data for the rr-1 protocol and the PROFIBUS protocol (best value over all parameters) B_L vs. number of wireless terminals N for 50% low priority load and independent and complex error models 173

7.8 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round-robin bounds k 175

7.9 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round-robin bounds k 176

7.10 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, independent error model and different round-robin bounds k 176

7.11 Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, complex error model and different round-robin bounds k 177

7.12 Fraction of time, that all stations are member of the logical PROFIBUS ring over all different gap factors and T_{TTRT} values vs. station number N for different error models (50% low priority load) 177

7.13 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k 182

7.14 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k 182

7.15 Ratio of the remaining bandwidth for low priority data for the rrk+spr protocol and the rrk protocol $\frac{B_L(rrk+spr)}{B_L(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k 183

7.16 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k 183

7.17 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k 184

7.18 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k 184

7.19 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k 185

7.20 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k 185

7.21 Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k 186

7.22 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k 188

7.23 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k 188

7.24 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k 189

7.25 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k 189

7.26 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k 190

7.27 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k 190

7.28 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k 191

7.29 Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k 191

7.30 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k 193

7.31 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k 193

7.32 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k 194

7.33 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k 194

7.34 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k 195

7.35 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k 195

7.36 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k 196

7.37 Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k 196

7.38 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k 198

7.39 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k 198

7.40 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k 199

7.41 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k 199

7.42 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k 200

7.43 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k 200

7.44 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k 201

7.45 Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k 201

List of Tables

3.1 Maximum allowed transmit powers in different regions 39

4.1 Relation between bitrate and distance for the RS-485 PHY and cable type A 48

4.2 PROFIBUS FDL-Services 49

5.1 Fixed parameters common for the analytical model and the simulation model 83

5.2 Fixed parameters for ring stability simulations 83

5.3 Fixed parameters for ring re-inclusion simulations 86

5.4 Fixed parameters for ring stability simulations over wireless channel 99

6.1 Path loss exponents for different environments 108

6.2 Adjustable radio parameters 112

6.3 Adjustable packet stream parameters 112

6.4 Fixed parameters for **longterm1** and **longterm2** measurements 118

6.5 Fixed parameters for **factorial** measurement 118

6.6 Variable parameters for **factorial** measurements 118

6.7 First order statistics of compound packet fate sequence of **longterm1** measurement (BL: burst length) 120

6.8 First order statistics of burst lengths (BL) of the **longterm1** measurement (COMPPLIS, $k_0 = 1$) 121

6.9 Number of lost packets 122

6.10 Mean Bit Error Rates for different modulation types (**factorial** measurement) 126

6.11 Burst lengths of error burst with density one for QPSK and BPSK traces 131

6.12 Summary statisitis of trace 24 (EBL=error burst length, EFBL=errorfree burst length, MBER=mean bit error rate) 143

6.13 Transmission times for 1 GB data over channels with different error models (based on **factorial** trace 24) 144

7.1 Summary statistics of selected traces constituting the complex error model (EBL=error burst length, EFBL=errorfree burst length, MBER=mean bit error rate, PLR=packet loss rate, PLB=packet loss burst) . . . 158

7.2 Different frametypes of polling-based protocols, with minimum parameters, x is either SDR or SPR . . . 159

7.3 Simulation parameters for comparison between PROFIBUS and k-limited round robin 169

7.4 Maximum number of negative confirmations over all N's for the different error models (PROFIBUS protocol, $T_{TTRT} = 0.005$s, gap factor = 1) . . . 169

7.5 Common simulation parameters for performance comparison of the different protocol modifications rrk+x vs. k-limited round robin . . . 180

7.6 Repoll function set for the frk protocols, for all functions additionally $f_x(i) = -1$ if $i > max\_retry$ holds . . . 180

7.7 `execute-explicit-poll` frame statistics for rrk+spr, independent error model, $N = 20$, $k = 1$ . . . 181

B.1 Mean packet error rate (PER) and max. PER for QPSK and BPSK modulation and different scrambling modes . . . 217

List of Acronyms

CAN Controller Area Network Small scale fieldbus for automotive applications

PROFIBUS PROcess FIeld BUS

PROFIBUS-DP PROFIBUS Decentralized Periphery

PHY Physical layer

FDL Fieldbus Data Link

MAC medium access control

FMS Fieldbus Message Specification PROFIBUS application layer services

MMS Manufacturing Message Specification

MAP Manufacturing Automation Protocol

LAN local area network

WLAN wireless local area network

NRZ non-return to zero

STP shielded twisted pair

FIFO first in first out

SAP service access point

SDA send data with acknowledge

SDN send data with no acknowledge

SRD send and receive data

CSRD cyclic send and receive data

DSAP destination service access point

SSAP source service access point

LAS list of active stations

GAPL gap list

FCB frame count bit

FCV frame count valid bit

SC short acknowledgement

DA destination address

SA source address

FC frame control

FCS frame check sequence

FIP factory instrumentation protocol

TDMA time division multiple access

DSSS direct sequence spread spectrum

FHSS frequency hopping spread spectrum

DCF distributed coordination function

PCF point coordination function

QoS quality of service

DTMC discrete time markov chain

BER bit error rate

MBER mean bit error rate

PER packet error rate

PLR packet loss rate

BI burstiness index

MSOT mean station outage time

CSOT cumulated station outage time

FEC forward error correction

LOS line of sight

NLOS non line of sight

BPSK binary phase shift keying

QPSK quaternary phase shift keying

DBPSK differential binary phase shift keying

DQPSK differential quaternary phase shift keying

BMBOK binary m-ary bi-orthogonal keying

QMBOK quaternary m-ary bi-orthogonal keying

CCK complementary code keying

ISM industrial, scientific and medical

CoV coefficient of variation

EDM electrical discharge machine

PER packet error rate

NIC network interface card

BEIS bit error indicator sequence

PLIS packet loss indicator sequence

COMP-PLIS compound packet loss indicator sequence

PEIS packet error indicator sequence

OSI Open Systems Interconnection

ICS Industrial Communication System

MMPP Markov Modulated Poisson Process

OFDM Orthogonal Frequency Division Multiplexing

MSDU MAC service data unit

PDU protocol data unit

AP access point

ARQ automatic repeat request

Acknowledgements

This work was carried out during my stay as a research and teaching assistant at TU Berlin. It would not have been possible without the help, understanding and friendship of several persons.

First of all I would like to thank Prof. Adam Wolisz, who served not only as a reviewer for my thesis, but was a very fair, helpful and humorable “boss” during the last six-and-a-half years. He provided me not only with insightful discussions, but also with the necessary degree of freedom. I learned a lot from him.

I am very grateful to Prof. Klaus David, who agreed to review my thesis on a very short-term basis.

The Deutsche Forschungsgemeinschaft (DFG) supported parts of my work during the last two years.

There are three persons who carried out work helpful for my thesis. Christian Hoene and Martin Kubisch put much effort into the measurement hard- and software, while Andreas Köpke helped me a lot with performing simulations.

For the tedious effort of proofreading and critiquing parts of my thesis I have to thank the following persons: Holger Karl, Morten Schläger, Helen J. Wang and Adam Wolisz. Specifically Holger Karl has put much effort into my thesis and into earlier papers and reports.

I had the pleasure to share much of my work and life during the last years with my dear colleagues Morten Schläger, Jean-Pierre Ebert, Berthold Rathke, Holger Karl, and Petra Hutt, who were a constant source of helpful discussions, practical help in certain situations, spontaneous chats, and several kinds of entertainment. We spent much time together and I consider them as friends.

This work would not have been possible without the support of my family and friends, who helped me over certain difficult situations by just being there. I would like to name here my mom Angela Hauschildt and my sister Christiane Reiners, and my closest friends Klaus Müller and Christoph Volkmann. I am also being indebted to my former wife Bärbel Peters. I still have many reasons to be grateful to her. Helen Wang provided me with lots of motivation during the last stage of my thesis.

Berlin, in October 2001

Andreas Willig

Summary

Fieldbus systems are a specific class of LAN technology for industrial and factory applications, targeted for fulfilling hard (safety-critical) realtime requirements in harsh environments. Many of these applications involve mobile subsystems and could benefit from recent wireless LAN technologies replacing the current cable-based systems.

The IEEE 802.11 wireless LAN standard is currently the leading WLAN technology, many components are commercially available. Hence, an immediate question is if and how this technology could be used for wireless fieldbus systems. An important aspect of this question is how to provide good "realtime performance" over the error-prone and time-variable wireless link. The notion of realtime performance captures the hard timing- and reliability requirements of industrial applications.

This thesis answers this question for the PROFIBUS, a fieldbus system popular in Europe. As a final goal, it follows the vision of integrating wired and wireless stations into a single PROFIBUS LAN. The scope of the investigations is focused to the MAC- and link-layer, where the PROFIBUS employs a token-passing protocol on top of a broadcast medium, where the stations form a logical ring. The results reported in this thesis suggest to use a specifically tailored MAC protocol on top of the 802.11 physical layer, and to drop the 802.11 MAC protocol as well as the PROFIBUS MAC protocol. One obvious candidate, namely the 802.11 MAC protocol, could not be matched well with the PROFIBUS services, and the other obvious candidate, the PROFIBUS protocol, has problems with the stability of the logical ring over the error-prone wireless medium, even after introducing some modifications.

The choice of dropping the PROFIBUS MAC protocol complicates integration of wired and wireless stations in a single LAN (where clearly the wish is to leave the wired station's protocol stack unchanged), but offers the opportunity to look for MAC protocols with better realtime performance.

In this thesis the class of polling-based protocols is proposed as a candidate. It is shown, that even a very simple k-limited round-robin variant often achieves much better realtime performance than the PROFIBUS protocol. Furthermore, the performance of round-robin can be improved. Three modifications of round-robin are proposed, which are designed to cope with the error characteristics of a wireless link as they are found in measurements in an industrial environment. However, the much better realtime performance of these protocols comes at the price of a slightly modified semantics of the PROFIBUS link layer services.

The overall contribution of this thesis is to provide the first steps towards a wireless PROFIBUS integrating wired and wireless stations.

Zusammenfassung

Bei sogenannten Feldbussystemen handelt es sich um eine spezielle Klasse lokaler Netzwerke (LANs), die insbesondere industrielle Anwendungen im Blick hat. Diese Anwendungen zeichnen sich durch harte Echtzeit-Bedingungen aus: es müssen sicherheitskritische Nachrichten, z.B. Alarme, innerhalb einer maximalen Zeit sicher übertragen werden können. Hinzu kommt, daß Feldbussysteme vielfach in rauhen Umgebungen eingesetzt werden. Viele industrielle Anwendungen haben mobile Subsysteme, und können somit von aktuellen drahtlosen Netzwerktechnologien (wireless LANs, WLANs) profitieren, die Mobilität in natürlicher Weise unterstützen, im Gegensatz zu den bisherigen kabelgebundenen Technologien.

Der IEEE 802.11 wireless LAN Standard ist derzeit die führende WLAN-Technologie. Der Standard ist ausgereift, und fertige Systeme bzw. Komponenten sind kommerziell erhältlich. Es ist daher naheliegend zu fragen, ob und wie diese Technologie für drahtlose Feldbussysteme genutzt werden kann. Eine zentrale Frage dabei ist, wie über das drahtlose Medium trotz hoher Fehlerraten und zeitvariablen Fehlerverhaltens eine möglichst gute "Echtzeit-Leistung" ("Realtime-Performance") erzielt werden kann. Der Begriff der Echtzeit-Leistung faßt gleichzeitig Zeit- und Zuverlässigkeitsaspekte der Übertragung sicherheitskritischer Daten ins Auge.

Die vorliegende Dissertation bearbeitet diese Frage für den PROFIBUS, einem in Deutschland und Europa weit verbreiteten Feldbussystem. Das Fernziel, auf das hin diese Arbeit die ersten Schritte macht, ist die Integration drahtloser und verdrahteter Stationen in einem einzigen PROFIBUS LAN. Der Focus liegt hierbei auf der Mediumzugriffs- und Sicherungsschicht des OSI-Referenzmodells, wo der PROFIBUS ein Token-Passing-Protokoll auf einem Broadcast-Übertragungsmedium einsetzt, und die angeschlossenen Stationen einen logischen Ring bilden. Die vorgestellten Ergebnisse legen nahe, ein speziell auf die (Fehler-) Eigenschaften des drahtlosen Mediums zugeschnittenes Zugriffs- und Sicherungsprotokoll zu verwenden, und die existierenden Protokolle, namentlich das PROFIBUS Token-Passing Protokoll und das IEEE 802.11 Zugriffsprotokoll, nicht in Betracht zu ziehen. Das 802.11 Zugriffsprotokoll bietet keine ausreichende Unterstützung zur Implementierung der Dienste der PROFIBUS-Sicherungsschicht. Das PROFIBUS-Protokoll hingegen hat über einem fehlerbehafteten bzw. drahtlosen Medium erhebliche Probleme mit der Stabilität des logischen Rings, selbst nach Einführung geeigneter Protokollmodifikationen.

Die Entwurfsentscheidung, das PROFIBUS-Zugriffsprotokoll durch ein anderes zu ersetzen, erschwert die Integration drahtloser und verdrahteter Stationen in einem PROFIBUS LAN, insbesondere unter der Randbedingung, daß der Protokollstack der verdrahteten Stationen nicht verändert werden soll. Umgekehrt ermöglicht sie aber auch die Suche nach Protokollen mit besserer Echtzeit-Leistung, als sie das PROFIBUS-Protokoll bietet.

Die vorliegende Arbeit identifiziert die Klasse polling-basierter Protokolle als aussichtsreichen Kandidaten. Es wird gezeigt, daß bereits ein einfaches k-beschränktes Round-Robin Protokoll oft eine erheblich bessere Echtzeit-Leistung erzielt als das PROFIBUS-Protokoll, der Unterschied beträgt bis zu einer Größenordnung. Weiterhin werden drei Modifikationen des Round-Robin-Protokolls vorgestellt, mit denen die Echtzeit-Leistung weiter erhöht werden kann. Beim Entwurf dieser Modifikationen wurden die spezifischen (Fehler-) Eigenschaften drahtloser Medien einbezogen, wie sie sich in Messungen in einer industriellen Umgebung zeigten. Die verbesserte Echtzeit-Leistung der modifizierten Protokolle wird allerdings auf Kosten einer leicht veränderten Semantik der Dienste der PROFIBUS-Sicherungsschicht erreicht.

Insgesamt gesehen liefert die vorliegende Arbeit die ersten Schritte auf dem Weg zu einem PROFIBUS-System, in welchem drahtlose und verdrahtete Stationen integriert sind.

Chapter 1

Introduction

In the past 20 years we have seen rapid advances in local area network (LAN) technologies, going hand in hand with an ever increasing number of application areas. One specific field of application are distributed control systems in industrial environments, e.g, in machine plants. As compared to office environments, these applications have different requirements. First, often the environmental conditions are harsh. For example, explosible environments or environments with aggressive chemicals, strong motors, drives, robots, controllers, and much more devices capable of creating electromagnetic noise and strong magnetic fields, are frequently found. Second, in distributed control systems a multitude of intelligent controllers exist. These are jointly responsible for controlling and monitoring an industrial process. Hence, these controllers have to communicate with each other and with lower level devices like sensors and actuators. These devices constitute the controllers interface to the underlying physical process. For communication between controllers and low-level devices often *hard realtime requirements* are posed: the correctness of a message transmission depends not only on its integrity, but also on timely and reliable/acknowledged delivery before a certain deadline. In hard realtime systems a task or a message missing a deadline can lead to damage of persons or material (accordingly, we speak of *safety-critical messages*), or at least to an interruption of the industrial process. As an example, consider in a milling machine the pressure on the cooling water tube drops rapidly due to a program malfunction damaging the tube. The machine should be stopped within a certain time for avoiding severe damage. In contrast to this, with *soft realtime requirements* a message missing a deadline may disturb a user, but does not lead to real damage.

For interconnecting controllers and low-level devices often LAN technology is used. However, instead of using popular technologies as Ethernet or Token-Ring, people resort to a special class of LANs designed for meeting hard realtime requirements and to survive in harsh environments. These systems are called *fieldbuses*, and for even more rigid requirements on communications (isochronous data and response times in microsecond range) they are called *sensor/actuator buses*. [128], [30], [100], [127]. With respect to the Open Systems Interconnection (OSI) reference model [166, chap. 1.4], most fieldbus systems have the distinguishing property that only layer 1 (physical), 2 (medium access control (MAC) and data link) and 7 (application) are covered, all other layers are empty. Fieldbus applications are usually not considered to need internetworking capabilities.

Another class of LANs, which attracted much attention in the last ten years, is the wireless local area network (WLAN) technology. Users appreciate the obvious advantages WLAN technology can offer:

mobility and reduced need for cabling. Nowadays, the IEEE 802.11 WLAN standard is the leading wireless technology. It is standardized, offers comparably high bit rates of up to 11 MBit/s, makes use of license-free bands and it is possible to buy off-the-shelf components.

Much of the research effort spent on 802.11 and on WLANs in general centers around the question, how to overcome the sometimes bad, unpredictable, and time-variable error behavior of wireless transmission. As opposed to todays wired LANs, transmission errors cannot considered to be a rare exception. Instead, they are a serious problem, and total transmission outages lasting for several seconds are frequently observed (see Chapter 6).

It is obvious that the advantages of WLAN technology would be also desirable for industrial applications. Some examples are: [52, chap. 2]:

- Applications with mobile subsystems, e.g., autonomous transport vehicles, robots, turntables.
- Implementation of distributed control systems in explosible areas (where sparks created by cables or adaptors have to be avoided) or in the presence of aggressive chemicals, capable of damaging cables.
- Rapid Prototyping of industrial plants without putting much effort into cabling.
- Mobile machine plant diagnosis systems, and wireless stations for programming and configuring.

Hence, a basic question is if and how the leading 802.11 WLAN technology can be used for a *wireless fieldbus*. A major challenge is clearly to satisfy the hard realtime requirements even for the case of bad and unpredictable transmission errors.

In this thesis we demonstrate that using the popular IEEE 802.11 direct sequence spread spectrum (DSSS) Physical layer (PHY) there is no hope for implementing hard, i.e., deterministic guarantees of acknowledged delivery of safety-critical messages within a prescribed time-bound. A wireless link showing total transmission outages in the timescale of several seconds cannot be expected to allow for reliable message transmission within a desired timescale of milliseconds, as often required by fieldbus applications. Furthermore, due to the physical mechanisms leading to transmission errors, we expect that the sometimes high error rates and the variability of the error behavior will not disappear when using comparable wireless transmission technologies. Hence, many of the problems which occur with 802.11 will also occur with other wireless technologies.

Therefore, instead of seeking absolute guarantees for wireless fieldbuses[1], this thesis follows an approach, which could somewhat sloppy be called *stochastic hard realtime*: the percentage of safety-critical messages which can be transmitted reliably (i.e., acknowledged) within a prespecified time-bound should be as high as possible ($> 99.x\%$), even at the cost of other performance measures like throughput or mean delay. We define a set of performance measures, called the *realtime performance measures*, which capture the timeliness and reliability requirements. Of course, this approach limits the application areas of a wireless fieldbus: when deterministic guarantees in the range of 10 to 100 msec are essential (e.g., cooling water control in a nuclear power plant), wireless fieldbuses are ruled out. However, in applications, where occasionally emergency stop conditions due to message losses are tolerable, wireless fieldbuses can offer their potential.

[1]From now on we will use the term "wireless fieldbus" as an abbreviation for "a wireless fieldbus based on IEEE 802.11 DSSS-compliant technology".

This thesis focuses on the PROcess FIeld BUS (PROFIBUS), a standardized and well-known fieldbus. This system is quite popular in Europe, there are many products and applications.[2] Due to its widespread usage it is attractive to have a *wireless extension* of PROFIBUS, i.e., a system where wired stations and wireless station can run jointly in a single PROFIBUS LAN.

Now we can state the central question this thesis focuses on:

> *How and to what extent can the IEEE 802.11 technology be used to create a wireless PROFIBUS extension with the best possible realtime performance? What are good MAC- and link-layer protocols for this?*

The MAC and link-layer protocol is in general a key issue for performing hard real-time communications, since, if these layers are not able to give tight time- and reliability guarantees (medium access time), this can be hardly corrected by other protocol layers.

The approach taken in this thesis is to keep the PHY layer fixed (to the IEEE 802.11 DSSS PHY), and also the PROFIBUS link-layer interface. The first decision takes the advantage that IEEE 802.11 technology is available, stable, and cheap. The latter decision allows easy porting of applications and application-layer instances (as is typical for fieldbus systems, the PROFIBUS specification covers only layer 1, 2, and 7 of the OSI reference model). Within this range several alternatives are explored:

- The PROFIBUS MAC and link-layer protocol (a semi-reliable protocol with an underlying token-passing scheme on top of a broadcast medium) runs directly on the 802.11 DSSS PHY. As one contribution of this thesis we show that the PROFIBUS protocol even with some modifications has serious problems regarding its realtime-behavior, when transported over error-prone, wireless type links (Chapter 5). This finding applies also to the case where all PROFIBUS stations have a wired transceiver and a piece of cable is replaced by a wireless link. Here also the token-passing protocol has to be transported over a wireless link, which should be avoided.
- Find a mapping between the PROFIBUS link-layer services and the IEEE 802.11 MAC protocol, which in some parts is designed with realtime services in mind (Chapter 3). However, MAC-services and link-layer services do not match well.
- Look for a specifically tailored MAC and link-layer protocol for the wireless side, which, however, could be integrated with the existing PROFIBUS protocol. This integration is important in order to run wired and wireless stations in a single LAN without changing wired station's protocol stacks.

The design space for the alternative using specifically tailored protocols has to the author's best knowledge not been explored so far. It includes architecture and protocol stack, physical layer, MAC and link-layer, mobility, power saving, security and authentication, management and configuration issues, applications. Out of this design space, this thesis makes the following contributions, focused on the design of MAC and link-layer protocols:

- We identified some problems in the PROFIBUS MAC and link-layer protocol when faced to a lossy link. We propose incremental improvements to the protocol, but still the delay results obtained for safety-critical messages are unsatisfying. And this motivated our investigation on entirely different approaches.

[2]The WWW site of the PROFIBUS user organization (www.profibus.com) gives a number of 200.000 installations.

- We advocate the class of polling-based protocols as an attractive candidate for a wireless PROFIBUS MAC and link-layer protocol, when the goal is to maximize probability of successful delivery of safety-critical messages within time.
- It provides characterization and stochastic modeling of a wireless link using real-world traces taken in an industrial environment. The results are used as a design input for the polling-based MAC and link-layer protocols. Furthermore, they are used for parameterizing existing stochastic error models and as a motivation for developing a new class of stochastic error models. This class offers better modeling accuracy as compared to well-known stochastic models at moderate increase in model complexity.
- Some approaches for polling-based protocols are presented, taking the findings from the measurements and the corresponding stochastic error models into account. Starting from a baseline protocol (k-limited round robin), three additional protocol mechanisms are discussed and compared with respect to their realtime performance. It is shown that already the round-robin protocol significantly outperforms the PROFIBUS protocol under many circumstances. Furthermore, it is shown that the realtime performance already delivered by k-limited round robin can be improved significantly with the proposed modifications. All this together justifies the recommendation given in this thesis to drop the PROFIBUS protocol on the wireless link and replace it by another one.

Polling is desirable for the following reasons:

- In the PROFIBUS protocol LAN membership depends critically on permanent and reliable transmission of specific control frames. Loss of these frames causes undesirable loss of ring members. This is critical, since only ring members are allowed to transmit data. In contrast, in the class of polling-based protocols considered here, after successful registration at a central controller LAN membership is not an issue.
- A polling approach frees the stations from competing for the right to transmit data frames or to make reservations with a central scheduler, while on the other hand being more efficient than strict TDMA. In many wireless MAC protocols all stations share a single (or a few) logical channels for transmitting data or reservations, introducing danger of collisions. In polling-based systems, one logical channel is used per station, eliminating collisions a priori. This is appealing for systems, where the focus is not on optimizing throughput, but on optimizing the probability of successful delivery of alarm messages.
- In general, bandwidth assignment by a central scheduler can be more efficient than decentralized approaches, since potentially more knowledge about pending requests is available at the scheduler.

Clearly, polling schemes have also some disadvantages. A single point of failure is introduced, which needs to be addressed by introducing redundancy. Furthermore, scalability concerns immediately come to mind. However, these pose no problem, since distributed control systems tend to have only small to medium numbers of stations.

1.1 Structure of the Thesis

Chapter 2 gives a general introduction to the topic of field buses and industrial communication systems: their requirements, architecture, traffic characteristics, popular systems.

Chapter 3 focuses on wireless LAN technology and 802.11. After a brief introduction to the general topic of WLANs (including some important properties of wireless media, as relevant for MAC design), we portray some existing systems. Following this, a more detailed overview on the IEEE 802.11 WLAN standard is given, including its architecture, the different physical layers and both the DCF and PCF MAC layers. The properties of the IEEE 802.11 DSSS PHY are of immediate importance, since they serve as the basis for the work on wireless link error characterization and the design of the polling protocols.

In Chapter 4 we describe the PROFIBUS system in detail including all the important protocol aspects, a description of the link layer services including the *link layer interface*, and a summary of the most important properties of the PROFIBUS protocol w.r.t. real-time behavior. After this, some general issues for wireless industrial communication systems are discussed, which are then specialized to the case of a wireless PROFIBUS. An important issue is a general description of the system under study. Of special importance is the definition of the *realtime performance measures*, capturing the realtime- and reliability requirements. An overview on related work on wireless field bus systems and wireless PROFIBUS system follows. This chapter also discusses the approach to find a mapping between the PROFIBUS link-layer services and the IEEE 802.11 MAC services (and protocol). It turns out that both do not match well.

In Chapter 5 the behavior of the classical PROFIBUS MAC and link layer protocol when operated over different types of error prone links, is investigated. Two different case are distinguished. In the first case the protocol is operated in a "cable-like" (yet error-prone) environment, i.e. the kind of environment for which the PROFIBUS was designed. The results indicate that the protocol was not designed with too high error rates in mind. The second case is that of a "wireless-like" medium, where no immediate feedback information from the channel is available and where beneath simple bit errors the loss of whole packets is an issue. It shows up that in both cases the PROFIBUS protocol has serious problems with keeping all stations in a state of being LAN members. These problems can be attributed to the need of explicit LAN membership / ring membership maintenance using special control frames. Two modifications of the protocol and its parameters are proposed, which give significant improvements, however, the ring membership problem remains serious. This is taken as a motivation to look for alternative protocol approaches, which do not require permanent exchange of control frames for LAN membership maintenance.

Chapter 6 is devoted to a more precise characterization of the error behavior of a wireless link. After a brief review of the physical mechanisms leading to distorted waveforms, and hence transmission errors, the most important results of bit- and packet error measurements taken in an industrial environment are reported. These results can serve as an input for the design of wireless MAC protocols for industrial applications. A second usage of these results is the "real-world data" parameterization of stochastic packet level error models, which are an important part of simulation models for assessing the performance of wireless MAC protocols. Several models are presented, and the class of *bipartite models* is introduced, giving good accuracy in predicting relevant performance measures at a moderate computational complexity.

In Chapter 7 we take the results of the measurements as input for the design of some concrete polling algorithms. After a description of the system under study and the measures of interest (the “realtime performance measures”) the realtime performance of the algorithms is evaluated and compared with a baseline algorithm, an k-limited round robin algorithm. Furthermore, the algorithms are compared with the classical PROFIBUS protocol, both operated with the same load and channel models. This comparison allows to check the basic claim that polling algorithms can give better realtime performance.

Finally, in Chapter 8 the conclusions and an outlook on possible research directions is given.

Chapter 2

Industrial Communication Systems

In this chapter we give an introduction to the notion and some basic architectural characteristics of industrial communication systems and field buses, including a brief overview on existing systems. We discuss the set of requirements for field buses and industrial communication systems, which make them distinct from other types of LANs.

Industrial communication systems and fieldbuses are only a single, yet important cornerstone in the process of making production plants more flexible and yielding a higher degree of integration of machines and tools from different vendors. In the eighties and in the beginning of the nineties the Computer Integrated Manufacturing (CIM) concept was a major stream of this process [146].

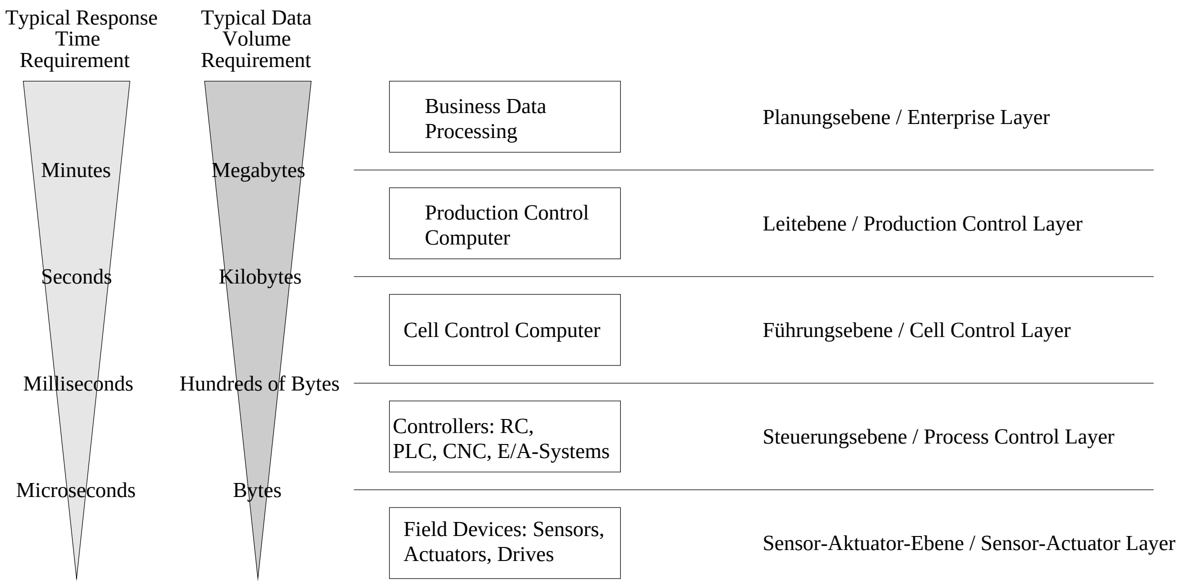
Typical Response Time Requirement
Typical Data Volume Requirement
Minutes
Seconds
Milliseconds
Microseconds
Megabytes
Kilobytes
Hundreds of Bytes
Bytes
Business Data Processing
Production Control Computer
Cell Control Computer
Controllers: RC, PLC, CNC, E/A-Systems
Field Devices: Sensors, Actuators, Drives
Planungsebene / Enterprise Layer
Leitebene / Production Control Layer
Führungsebene / Cell Control Layer
Steuerungsebene / Process Control Layer
Sensor-Aktuator-Ebene / Sensor-Actuator Layer

Figure 2.1: Hierarchy of information flows in manufacturing applications (from [68, p. 13])

The trend towards distributed control systems posed, amongst others, the problem of proper communications between devices of different vendors, using different protocols, media, and running different applications. An established way of grouping the different types of manufacturing applications and their needs for communications is to place them in a hierarchy of information flows, such that similar functions and similar data are grouped into the same layer. An example, taken from [68, p. 13] is shown in Figure 2.1. Although this distinction is not really sharp, it can be said that the boundary between the cell control layer and the process control layer is targeted by *field buses*, while the lower boundary between the process control layer and the sensor/actuator layer is targeted by *sensor/actuator buses.* Some general references for field buses and realtime MAC protocols are [128, 30, 13, 32, 127, 94].

2.1 Requirements for Industrial Communication Systems / Field Buses

The requirements for industrial communication systems / field buses are much different from those for LANs in office environments. Clearly, they depend on the targeted application area, but some requirements are typical.

For this thesis the most important requirements are those regarding the timing- and reliability behavior. Fieldbuses are targeted for *hard realtime* requirements [100, chap. 2], [153]. In hard realtime communication systems there exists a class of messages with timing constraints. If a message belonging to this class is not transmitted successfully within a certain time bound, this is a system malfunction, possibly leading to a catastrophe. Consider as an (extreme) example an alarm message generated by a pressure sensor in the cooling water circulation system of a nuclear power plant. In contrast to this, for *soft realtime* systems missing a timing constraint may lower the systems usability, but does not compromise the systems integrity. As an example, consider an online transaction system, where users expect their answers within one second, but do not get too annoyed, if it occasionally takes two seconds. As another example, for packet-based speech conversations like in Voice over IP systems, there are stringent delay requirements (the delay is desired to be ≤ 250 ms), but a certain packet loss rate (typically $\approx 1\%$) seems tolerable, depending on the codec and the influence of error concealment techniques [65, chap. 7].

The requirements for hard realtime communications can be summarized as follows (compare [127]):

- Safety-critical messages must be reliably (i.e., acknowledged) transmitted within a bounded time. The time bound is application-dependent. For cell control and process control applications they are often in the range of 1-100 msec (compare Figure 2.1). If the time bound cannot be met, this should be signaled to upper layers.
- There should be support for priorities, to allow distinction between urgent (safety-critical) and non-urgent messages.
- Packets can be equipped with deadlines. On deadline expiration appropriate actions have to be taken (dropping the packet, notifying upper layers).
- For some data it must be known, how old they are ("freshness"). For example, the current position of a moving drive is valid only for a short time.
- Messages with stringent timing constraints are typically only of some few bytes length (compare Figure 2.1).

- Both periodic and aperiodic (asynchronous) traffic types should be supported. Some applications (e.g., drive control) require even strictly isochronous traffic.

A frequently observed phenomenon are *alarm storms*: a machine plant runs for some time in a "normal" operation mode. Then something critical happens, and all the subsystems involved in an industrial plant detect the error condition and try to send alarm messages to other stations. This often leads to generation of further alarm messages, congesting the communications system more and more.

Beneath the traffic- and realtime requirements fieldbuses have often to fulfill environmental requirements:

- Fieldbuses are expected to operate in environments with much electromagnetical noise, e.g., due to the presence of strong motors, drives, electrical discharges, remote controls.
- In some applications the environment can be explosible or aggressive chemicals may be present.
- Operation in free air or wet environments, with large temperature differences, vibrations.

In chemical engineering installations a single fieldbus installation can extend to up to 10 kilometres, however, in manufacturing applications the geographical area covered is much smaller, typical in the range of 20-50 meters. In some installations small devices like sensors and actuators get their power supply via the fieldbus.

2.2 Architectural Characteristics

Many fieldbuses and sensor/actuator networks share the property that their specification covers only the layers 1 (physical layer), 2 (MAC- and data link-layer), and 7 (application layer) of the OSI reference model, compare Section 2.3. Distributed control applications consist typically only of a small to medium number of stations, hence, internetworking capabilities are not needed. The other layers are empty or their functionality is put into one of the remaining layers:

- The representation layer is unnecessary, since all usable data types and their memory representation are defined in a fixed manner.
- Session layer functionality is considered unnecessary.
- The network layer is empty, since internetworking is not needed.
- With an empty network layer the transport layer can also be left empty.

2.3 Popular Systems

In this section a brief overview on some popular fieldbus and sensor/actuator buses is given, with emphasis on their MAC protocols.

Many fieldbus and sensor/actuator buses (including the PROFIBUS) use an explicit token-passing scheme on top of a broadcast medium. Broadcast mediums are preferred over ring-type networks, since in the latter case a cable-break destroys the whole network. The token-passing systems are discussed first.

2.3.1 Systems using Token-Passing-MAC's

The PROFIBUS is a fieldbus, hence its targeted application area is the interconnection of cell controllers and process controllers. It uses a token-passing protocol. This fieldbus is the focus of this thesis, it is discussed in much more detail in Section 4.1.

The IEEE 802.4 Token-bus [71] was designed for factory automation applications with the goal of guaranteeing an upper bound on medium access time [165, chap. 3]. It is part of the Manufacturing Automation Protocol (MAP) protocol stack (see below), but has not gained widespread usage. The system uses a token passing protocol on top of a broadcast medium, and the stations form a *logical ring*. In this respect it is quite similar to the PROFIBUS. There are differences in the ring maintenance / ring-inclusion mechanisms: in the IEEE 802.4 Token-bus a contention-based approach is used, while in PROFIBUS the ring members have to poll a certain address range in order to include new stations.

The BITBUS specification came from Intel in 1983; in 1991 an extended version was accepted as an IEEE standard [70], [68]. It is designed as a field bus, its main application area is at the cell layer. The standard covers only the layers 1, 2, and 7. The BITBUS concept includes a communication system and a microcontroller card. The communication system is based on a master/slave access scheme (employing SDLC, a predecessor of HDLC [151] created by IBM) and round-robin polling by the master. In an extended version the protocol allows for changing masters by means of explicit token passing. Hence, the protocol has similarities to the PROFIBUS protocol. The microcontroller card contains a small realtime operating system and allows the user to load arbitrary tasks into the card. The hardware is not specified, but the operating systems interface is. It seems that nowadays the BITBUS is not widely used in Europe.

2.3.2 Other Systems

The FIP fieldbus (fr.: flux information processuale, engl.: factory instrumentation protocol) is a european fieldbus standard, which emerged from a french standard [178], [30], [89]. The standard covers only the layers 1, 2 and 7 of the OSI reference model. The distinguishing feature of FIP is its *real-time database* concept. The basic idea is that most applications are interested merely in the value of some process variable (as denoted by a variable identifier), and not in which station actually produces this variable. Therefore, FIP introduces a producer / consumer concept and utilises a central station (called *bus arbiter*, BA). The BA keeps a preconfigured polling table, holding the variable identifiers of interest. When traversing the poll table, the BA broadcasts the variable identifier. The producer of the variable responds to this by broadcasting the variables value, and each station interested in this value (consumer) takes a copy and places it into an internal cache. FIP is designed for fixed configurations, since adapting to shifts in communication needs requires changing the polling table.

The P-NET protocol was originally developed in Denmark, and later on adopted as a european standard [176]. Two types of stations are used: master stations and slave stations. The right to transmit is passed between multiple master stations according to a virtual token passing scheme, between master and slave a simple request/answer scheme is used. In the virtual token passing scheme each station maintains an *access counter*, which is incremented upon every medium idle time longer than 40 bit times. If this access counter happens to be the same as one master's station address, this master is said to have the token.

The InterBus-S system was first defined by a german vendor, in 1994 the specification became a german standard [36], [37], [38]. The standard covers the layers 1, 2, and 7 of the OSI reference model. This system is mainly designed as a master / slave communication system with strictly isochronous services. The latter are required to enable drive control applications or the isochronous operation of a PLC (Programmable Logic Controller). Instead of using a bus, the architecture is built upon a ring topology. The master station sends out data frames in fixed intervals. Each slave is assigned a specific slot in this frame. When a frame reaches a slave, it reads the contents of its slot (considering this as input data) and writes some other data to the same position (output data). After this the frame is sent to the next station. While this approach is fine for exchanging cyclic measurement data, the transport of asynchronous data is more cumbersome, since only a small number of additional slots within a frame is provided. Hence, a message can span several cycles.

The Controller Area Network (CAN) is a widely used fieldbus for (geographically) small installations, e.g., for interconnecting several devices and controllers in cars. First invented by Bosch and Intel, it has become an ISO standard [50]. Controller Area Network (CAN) is a decentralized multi-master network using a CSMA/CD with priority arbitration. The protocol allows for smooth implementation of rate monotonic scheduling approaches [98]. A station that wants to transmit, waits until the bus goes idle, then starts with a startbit. After the startbit the 12 (or 30) bit long *arbitration field* is transmitted. More precisely: every station starts transmitting the first bit of the arbitration field, and in parallel reads back the signal from the bus. If the read back bit differs from the transmitted bit, the station has lost contention and waits for the next cycle. If a station is not eliminated during the arbitration phase, it continues to transmit the next bit of the arbitration field. A prerequisite for implementing this is that overlapping signals on the medium still give a valid signal. A similar protocol is used on the ISDN S_0 bus [83].

2.3.3 MAP/MMS

A much broader approach was taken by the MAP consortium [80], [112]. Within MAP a full protocol stack covering all seven layers of the OSI reference model was specified. In fact, MAP version 3.0 was the first true OSI protocol stack on the market. In the application layer it uses Manufacturing Message Specification (MMS), a well defined set of services for automation purposes [72]. MMS offers services like variable access, event management, semaphore management, file transfers, program execution etc. The application layer services implement the *virtual field device* abstraction, which in turn is used by the application processes. A virtual field device represents a real field device as a collection of objects (variables, execution environments, events, files) and operations on them, accessible from other stations using the MMS services. On the MAC layer the IEEE 802.4 token bus is used [71]. However, MAP has not gained widespread acceptance. One reason for this were the serious performance problems, which are due to the fact that a single message has to pass fourteen protocol layers [101].

Chapter 3

Wireless LANs / IEEE 802.11

This chapter serves the purpose of providing background information about wireless local area network (WLAN) systems, and specifically about the IEEE 802.11 WLAN standard. The latter is of particular relevance for this work, since it is nowadays the leading technology. The commercial availability of IEEE 802.11 equipment makes it attractive and allows for experimentation and measurements. The experience we can gain from measurements gives practical hints for design of MAC and link layer protocols.

In Section 3.1 we describe some general characteristics of WLANs and discuss some specific problems for design of MAC and link layer protocols. Furthermore, a brief overview of current WLAN standards is provided. In Section 3.2 the IEEE 802.11 WLAN standard is presented, including the main characteristics of a specific 801.11 compliant chipset.

Some general references covering wireless transmission and wireless networking are [4, 28, 78, 124, 182, 183, 166], and a lot of separate topics in wireless are treated in [56]. References [107, 150, 125, 9], deal with the general topic of WLANs, and for MAC aspects please refer to [6, 60, 93].

3.1 Wireless LANs

3.1.1 Basics

The notion of wireless local area network (WLAN) systems subsumes systems, where data is transmitted wireless over short distances and in packet-switched mode. The focus on packet-switching makes WLANs different from existing cellular networks (e.g., GSM), which are primarily designed to support telephony or pager applications. WLANs typically offer higher bitrates. For example, the current IEEE 802.11 WLAN systems support bit rates up to 11 MBit/s, as compared to GSM (9.6 kBit/s) [144], the GSM enhancement GPRS (116 kBit/s) or the future UMTS (2 MBit/s indoor) [182].

Current WLAN systems use either infrared (e.g., IrDA [187, 166]) or radio frequencies below 6 GHz. Using radio frequencies allows to cover distances of 50-300 m (depending on transmission power). Furthermore, radio waves below 6 GHz can propagate through walls (depending on both frequency and material) and can be reflected on several types of surfaces, enabling non line of sight (NLOS) communications. In contrast, systems based on infrared only allow for line of sight (LOS) communications.

Most radio-based WLANs transmit in license-free frequency bands, e.g., the industrial, scientific and medical (ISM) bands, which are granted by the FCC[1] and the CEPT[2]. In these bands 26 MHz of spectrum between 902 MHz and 928 MHz, 83.5 MHz of spectrum between 2.4 GHz and 2.4835 GHz and 125 MHz of spectrum between 5.725 GHz and 5.85 GHz are allocated. The transmit power is legally restricted to 1 Watt. Since radio waves in the ISM bands can penetrate walls, can be reflected / diffracted and are subject to multipath fading, the wireless channel is comparably bad and time-varying (see Chapter 6).

Spread Spectrum Technologies

Many types of wireless LANs, including the IEEE 802.11 WLAN, rely on *spread spectrum* techniques [58], where a narrowband information signal is spreaded to a wideband signal at the transmitter and despreaded to a narrowband signal at the receiver. The two most important spread spectrum techniques are *direct sequence spread spectrum (DSSS)* and *frequency hopping spread spectrum (FHSS)*. By using a wideband signal the effects of narrowband noise or narrowband interference are reduced [58, chap. E].

In DSSS systems a data bit is multiplied with a fixed sequence of bits, called *chip sequence*, i.e., every chip is `XOR`ed with the data bit. The resulting chip sequence is transmitted. The receiver knows the chip sequence of the transmitter and tries to decode the original data bit from the received chip sequence, e.g., using correlation techniques. In many DSSS systems with multiplexing in the time domain all stations use the same chip sequence (e.g., the IEEE 802.11 with DSSS PHY, see Section 3.2), while in CDMA systems every station has its own.

In FHSS the available frequency band is divided into a number of subbands. The transmitter station hops through the subbands according to a predetermined schedule. The receiver must know this schedule and change frequencies synchronously with the transmitter. It is distinguished between *slow FHSS* and *fast FHSS*. In slow FHSS the transmitter transmits several information bits on the same frequency before hopping to another frequency. In fast FHSS the transmitter changes the subband several times during a single information bit. Fast FHSS systems are more costly to realize, because of the need of fast and accurate synchronization.

A third technique called time hopping spread spectrum is not used in current WLAN technologies.

In recent time there has been considerable interest in Orthogonal Frequency Division Multiplexing (OFDM) techniques [179]. OFDM is a multi-carrier technique, where blocks of symbols are transmitted in parallel over a number of subcarriers. A symbol transmitted on each subcarrier has an increased symbol duration τ as compared to full-rate transmission. The symbol duration τ is usually much larger than the delay spread of the channel, this way combatting intersymbol-interference. The recently submitted IEEE 802.11a standard [122] uses an OFDM PHY, as well as the recently submitted HIPERLAN/II standard [44, 45].

[1]Federal Communications Commission, a US government organization for Telecommuncation regulation issues.
[2]Conference of European Posts and Telecommunications Admistration.

Architecture

Many wireless LANs have a (micro- or pico-) cellular structure, where two types of stations are distinguished: *wireless terminals* (WT) and *base stations* (BS), the latter constituting a cell by its maximum transmission radius (typically between 5 and 100 m). The base stations are often interconnected by a *backbone system* or *distribution system.* Usually, the WTs communicate only with a single BS and with other WTs located in the same cell. For inter-cell communications the BS act as a forwarder, using the backbone to direct packets to the cell containing the target WT. When a WT moves from one cell to another, it has to associate itself with the new BS (*handover*), as future packets destined to the WT have to be directed to the new BS. In some systems the BS does not only act as a forwarder, but also plays a major role in wireless MAC protocols, e.g., as a central scheduler granting transmission rights to its associated WTs. The class of WLANs with BS and a backbone is often denoted as *infrastructure WLANs.* In contrast to infrastructure WLANs are *ad-hoc WLANs*, where WT in close proximitiy communicate with each other on a peer-to-peer basis or in a multi-hop fashion, without having a central station.

3.1.2 WLAN Properties Important for MAC Design

The wireless medium has some special properties affecting the design of MAC- and link layer protocols.

A major challenge is the error-prone and time-varying channel (see Chapter 6). Common ways to address this are ARQ protocols (based on immediate acknowledgements), forward error correction (FEC) codes or (maybe adaptive) combinations of both approaches (hybrid error control) [99].

It is not possible to transmit and receive simultaneously on the same frequency band, due to overcharge of receive filters . Hence, the transmitter cannot detect collisions by itself, as is required by, e.g., the collision detection part of a CSMA/CD MAC protocol. A possible solution would be to use feedback given by the receiver on another frequency band (busy-tone solutions [167]; these require a second antenna).

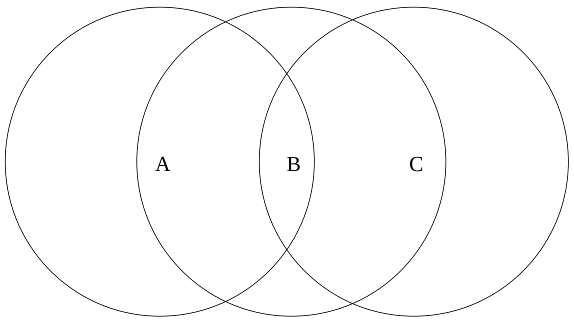

Figure 3.1: Hidden terminal scenario

Several problems arise due to *path loss* (discussed in Chapter 6) in conjunction with a *threshold property*: wireless receivers require the signal to have a minimum strength, before it is recognized. Therefore, if the distance between two stations exceeds some threshold, they cannot hear each other. For protocols incorporating carrier sensing (e.g. CSMA) this gives rise to the *hidden terminal scenario* [167], depicted in Figure 3.1. Consider three stations A, B, and C with transmission radii as indicated by the circles. Stations A and C are in range of B, but A is not in the range of C and vice versa. If C starts to transmit to B, A does not hear this and considers the medium to be free. Hence, A also starts packet transmission and a collision occurs at B.

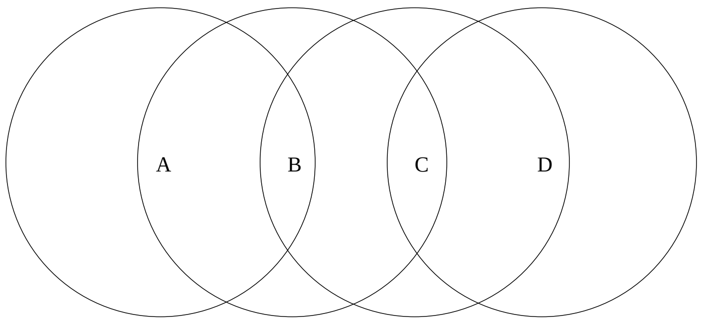

Figure 3.2: Exposed terminal scenario

There exists a second scenario, where carrier sensing leads to false predictions about the channel state at the receiver: the so-called *exposed terminal scenario*, depicted in Figure 3.2. The four stations A, B, C and D are placed such that the pairs A/B, B/C, and C/D can hear each other, all remaining combinations can't. Consider the situation where B transmits to A, and one short moment later C starts to transmit to D. Station C performs carrier sensing and senses the medium busy, due to B's transmission. As a result, C postpones its transmission. However, C could safely transmit its packet to D, without disturbing B's transmission to A. This leads to a loss of efficiency.

Radio waves can reach the receiver on several paths of different length, leading to *multipath fading*. As a result, the *channel impulse response* shows several maxima (*delay spread*) and different information symbols can overlap in time (*intersymbol interference*) [4, chap. 4.5]. Often, equalization techniques are used to compensate delay spread [132]. However, these techniques frequently need *training sequences* or *preambles*, as well as processing power and energy. Another reason for the need of preambles is the need to let the receiver learn about the clock frequency of the transmitter (bit synchronization). In such systems, in front of every packet a fixed sequence needs to be transmitted. These training sequences can occur at several places. For example, in the GSM standard a training sequence is placed in the middle of a packet [182, chap. 3.3], while IEEE 802.11 with DSSS (see 3.2) use a preamble at the beginning of a packet.

To summarize, MAC protocol designs based on an IEEE 802.11 PHY are faced to the following problems:

- Error-prone and time-varying channel.
- It is not possible to transmit and receive simultaneously on the same channel. The send/receive turnover costs some time.
- Not all stations see the same signals, carrier sensing gives accurate information only for the sensing point.
- Need for preambles.

The susceptibility of CSMA-based protocols to the hidden-terminal and exposed-terminal problem and the complexity of appropriate countermeasures (e.g., the RTS/CTS protocol described in Section 3.2.3) makes their usage in MAC protocols for industrial WLANs questionable.

3.1.3 WLAN Standards / Systems

Beneath the IEEE 802.11 WLAN standard, discussed in Section 3.2, some other systems have emerged.

The european HIPERLAN standard [43], [183, chap. 9], standardized by ETSI in 1996, works in an exclusively assigned frequency band of 150 MHz width (5.15 GHz - 5.3 GHz), subdivided into 5 channels a 23 MHz. The standard covers the physical layer, the MAC layer and the data link layer (including some routing functionality for multihop-forwarding). It uses two different bit rates: 1.5 MBit/s (low bitrate) for a packets preamble and header, and 23 MBit/s (high bitrate) for a packets data part. HIPERLAN employs the EY-NPMA MAC protocol, a CSMA-based stochastic MAC protocol with priorities (based on packet deadlines), collision avoidance algorithm and immediate acknowledgements. Basically, HIPERLAN is an ad-hoc network, with support for overlapping cells: stations situated in more than one cell can act as a forwarder. HIPERLAN has not gained any commercial impact, there have never been any products available.

The successor of HIPERLAN, the HIPERLAN Type 2 or HIPERLAN/2 standard was finalized in 2000 [44], [143]. HIPERLAN/2 uses OFDM transmission in the 5GHz frequency band with 52 subcarriers. The user bitrate is up to 54 MBit/s. The standard prescribes several modulation schemes (BPSK, QPSK, 16QAM, 64QAM, see [155]) and code rates, and stations can adapt to current transmission conditions by selecting proper modulation schemes / code rates. HIPERLAN/2 is an infrastructure network, i.e. it consists of picocells, each one organized around an access point (AP). Every station must be able to operate as AP. In peer-to-peer situations without any fixed (cabled) network infrastructure, a network can be built up by explicitly electing an AP from the available stations. The MAC layer uses small packets with only two possible sizes: 9 bytes (for control packets) and 54 bytes (for data and control packets). The AP runs a central scheduler, which grants transmission rights on demand (demand-assignment protocol). By selecting proper scheduling schemes, quality of service issues can be addressed. As basic transmission mode, a TDD/TDMA (Time Division Duplexing / Time Division Multiple Access) MAC scheme with fixed length superframes is used [60]. The data link control (DLC) protocol uses a selective repeat protocol. The radio link control (RLC) protocol includes connection management, mobility management, frequency management and power management. Several convergence layers placed on top of the DLC/RLC protocols support interconnection between HIPERLAN/2 and different types of core networks (e.g., Ethernet, FireWire). No products are available at time of writing.

There are two emerging standards specifically targeted for home environments and for wireless coupling of small devices: the HomeRF and Bluetooth systems. The Bluetooth system [61] is defined by an industry-driven working group, and aims at interconnecting low-cost devices, using cheap wireless transceivers. It is defined to be a scatter ad-hoc network, i.e., many ad hoc Bluetooth networks can share the same area and frequency band. It is based on a frequency-hopping CDMA scheme in the license free 2.4 GHz ISM band, working on 79 subchannels of 1 MHz width. The frequency is changed every 625 μs (dwell time), corresponding to a hopping frequency of 1600 Hz. A channel is defined by a particular hopping sequence. On each channel a raw bitrate of 1 MBit/s is available. A *piconet* is formed by a particular hopping sequence and consists of maximum eight stations: one *master station* and maximum seven *slave stations*. A large number of (nonorthogonal) hopping sequences is predefined. Within a piconet the master is responsible for arbitrating access to the channel, furthermore, only master-slave communications is used. The channel is organized in slots of one dwell-time length. One slot can carry a single packet. It is possible to transmit long packets of 3 or 5 slots length. Within such packets frequency hopping is suppressed. Full duplex communications is achieved by using a TDD scheme. Both isochronous connection oriented service and asynchronous connectionless services are available

The HomeRF system [114] is designed for home scenarios, where users want to access the Internet or the plain switched telephone system (PSTN) with handheld devices from everywhere within their homes. The HomeRF working group was formed in 1997 and delivered a first specification in january 1999. A HomeRF network consists of a *control point* and several clients. The control point is attached both to the PSTN and to a main home PC with internet access. Hence, voice and data are transmitted over the same wireless network, the HomeRF SWAP protocol has support for asynchronous data (TCP/IP communications) and for isochronous data (telephony). For asynchronous data transfer a peer-to-peer mode is available. In contrast, voice data transmission requires a control point. The physical layer is largely adapted from the IEEE 802.11 frequency hopping PHY (see Section 3.2). It uses a hop frequency of 50 Hz, operates in the 2.4 GHz ISM band and offers a raw bitrate of 1.6 MBit/s. The MAC layer is based on a superframe structure, with a superframe being 20 ms long (each superframe is transmitted on a single frequency). A superframe is subdivided into two contention free periods (CFP) and a contention period (CP), located between the CFPs. Speech data is transmitted on a TDMA basis in the second CFP, the corresponding retransmissions are placed in the first CFP of the following superframe (thus using another frequency). The CP is used for asynchronous data transmission and employs a CSMA/CA scheme derived from IEEE 802.11.

Finally, wireless asynchronous transfer mode (ATM) WLANs are targeted to support ATM services on a wireless medium. Some prototypes have been built (e.g., the WATM system [113], or the Magic WAND demonstrator [105]), however, none of these systems converged into commercial products.

3.2 IEEE 802.11

The IEEE 802.11 WLAN standard [120] was finalized in 1997, a revised version appeared in 1999. It belongs to the IEEE 802.x family of LAN and MAN standards. It offers the same abstract MAC interface as IEEE 802.3 Ethernet, hence, it can be used with an IEEE 802.2 logical link control (LLC) sublayer [119]. In 1999, two extensions were defined, providing additional physical layers: [122] describes an OFDM PHY in the 5 GHz band, while [121] describes an 11 MBit/s extension of the 2.4 GHz DSSS PHY. A detailed description of IEEE 802.11 can be found in [123].

Basically, the standard describes an architecture, services and protocols for an ethernet-like wireless LAN, using a CSMA/CA-based multiple access method with enhancements for time-bounded services. The protocols are designed to run on top of several physical layers. In this section an overview of the IEEE 802.11 architecture and its PHY and MAC layers is given, with emphasis on the mostly deployed DSSS PHY. Because of its widespread usage we have chosen the DSSS PHY as the basis for this thesis.

Certain aspects of IEEE 802.11, e.g., security, authentication or power saving are not relevant for this thesis and therefore not discussed.

For the discussion about services the terminology of the OSI reference model is used, which is summarized in Section 4.1.3.

3.2.1 Architecture

The main elements of the IEEE 802.11 architecture are *stations*, *access points* (AP), *portals*, the wireless medium, the *basic service set* (BSS), the *distribution system* (DS) and *extended service sets* (ESS).

A *station* is usually some computer or portable device with a *network interface card* (NIC), the latter carrying IEEE 802.11 PHY and MAC entities. On a station typically a user runs applications which initiate or respond to communications with other devices or users. Every station has to support *station services*: authentication, deauthentication, privacy and data delivery. While data delivery is self-explaining, the other services are designed to cope with the fact that a WLAN is “open” by its broadcast nature, as compared to wired LANs, where every member has to be explicitly attached to a wire:

- With authentication / deauthentication services different stations can assure each other that they are administratively allowed to communicate. Without authentication a station is not allowed to transmit data frames.
- The privacy service provides encryption of data frames from being eavesdropped.

A *basic service set* (BSS) is a set of stations which can communicate directly using the wireless medium. It is not required that every pair of stations belonging to a BSS can hear each other. It suffices that, if we take the stations as the edges of a graph and draw vertices between them if radio connectivity is given, the graph is connected.

There are two types of BSS: a BSS with an AP is denoted as *infrastructure BSS*, while a BSS without an AP is denoted as *independent BSS* (IBSS). The case of an IBSS corresponds to ad-hoc networks, which do not use authentication/deauthentication services. In an IBSS all communications is done on a peer-to-peer basis, however, without multihop forwarding.

The case of an infrastructure BSS is different. All communications is relayed through the AP. Several APs can be coupled via the distribution system (DS) to form an extended service set (ESS). In the infrastructure case every station has to *associate* itself with an AP in order to be allowed to communicate with other stations. When moving through the network the station may loose connectivity to its current AP and move into reach of another AP. In this case the station has to *reassociate* with the new AP. When station A wants to transmit a data frame to station B, it addresses the frame towards its AP X. The AP X checks whether B is in its BSS. If so, X sends the frame to B. If not, it transmits the frame via the DS to the AP Y in whichs BSS B resides. Finally, Y forwards the frame to B. The DS itself is not part of the standard. If A’s frame is not destined to a station in the ESS, the AP forwards it to a *portal* (e.g., an Ethernet bridge).

3.2.2 PHY Layer

The PHY layer of IEEE 802.11 is subdivided into two sublayers. The upper *physical layer convergence procedure* (PLCP) sublayer offers the interface to the MAC layer, the lower *physical medium dependent*

(PMD) sublayer actually transmits and receives data frames.

In the original specification [120] three physical layers are defined: an infrared PHY, a DSSS PHY in the 2.4 GHz ISM band and a FHSS PHY in the 2.4 GHz ISM band. The additional PHY layers are an OFDM PHY in the 5 GHz band [122] and an 11 MBit/s extension of the DSSS PHY [121]. We discuss only the DSSS PHY in some detail, the others are briefly summarized.

PHY Interface

The standard defines an abstract interface the PHY has to offer to an IEEE 802.11 MAC instance. The interface offers primitives to start and stop transmission of frames, to pass data, to obtain carrier sense information, and to indicate frame reception.

The `PHY-DATA`.request and `PHY-DATA`.indication service primitives are used for transferring single data bytes from the MAC to the PHY and vice versa. The `PHY-DATA`.request can only be used when the PHY is in transmit mode. The PHY answers this request with a `PHY-DATA`.confirm primitive. Received data bytes are transferred with the `PHY-DATA`.indication from the PHY to the MAC.

With the `PHY-TXSTART`.request primitive the MAC asks the PHY to start transmission of a frame. The result of this operation is signalled with the `PHY-TXSTART`.confirm primitive. To complete the frame transmission after the last data byte, the MAC entity issues a `PHY-TXEND`.request primitive, which is acknowledged by the PHY with a `PHY-TXEND`.confirm primitive.

When receiving packets, the PHY indicates this to the MAC with two primitives: the `PHY-RXSTART`.indication is issued after the PHY has acquired bit synchronisation and received a valid start frame delimiter. The `PHY-RXSTOP`.indication is issued after finishing frame reception.

The MAC layer needs carrier sense information for performing its CSMA/CA protocol (see Section 3.2.3). The `PHY-CCARESET`.request and `PHY-CCARESET`.confirm primitives allow the MAC to control the carrier sense (or *clear channel assessment* (CCA)) logic of the PHY. The `PHY-CCA`.indication is isssued by the PHY every time the channel changes its state from idle to busy or vice versa. This primitive indicates the channel state to the MAC.

DSSS PHY

The DSSS PHY offers two bitrates: 1 MBit/s with differential binary phase shift keying (DBPSK) modulation or 2 MBit/s with differential quaternary phase shift keying (DQPSK) [155]. A channel bandwidth of 22 MHz is used, the center frequencies of the DSSS channel are placed in 5 MHz steps in the 83.5 MHz wide 2.4 GHz ISM band (for Northern America 11 center frequencies are defined, for Europe 13). Hence, within the same geographical area three overlapping 802.11 WLANs with DSSS can be operated without mutual interference. The maximum allowed transmit powers in different geographical regions are shown in Table 3.1.

The 11 MBit/s DSSS extension [121] is an addendum to the DSSS PHY. Additional modulation schemes (complementary code keying (CCK)) are specified for 5.5 MBit/s and 11 MBit/s data rates. Furthermore, an additional mechanism (rate shift mechanism) is defined, which allows to set the default rate to binary phase shift keying (BPSK) or quaternary phase shift keying (QPSK), in order to cooperate with legacy WLANs or to adapt to channel conditions.

North America	1000 mW
Europe	100 mW
Japan	10 mW/MHz

Table 3.1: Maximum allowed transmit powers in different regions

The DSSS PHY uses an 11 bit Barker code for direct sequence spreading. Hence, one data bit is mapped to 11 chips. Each chip is transmitted either as a BPSK or as a QPSK waveform [155]. Each chip of the barker sequence takes the values 1 or -1. It has the interesting property that the inner product S of the Barker code with a shifted version of it (with lag k) takes the value $S = 11$ for $k = 0$ or $|S| \leq 1$ for $k \neq 0$. This simplifies the design of correlation receivers and maintenance of bit synchronisation.

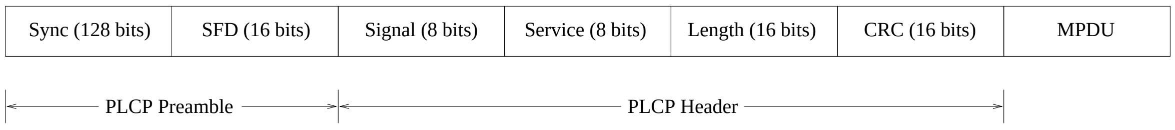

Figure 3.3: DSSS PHY PPDU Format

The PLCP part of the DSSS PHY forms its own protocol data units (denoted as PPDU) by adding some fields to a MAC frame. The PPDU format is shown in Figure 3.3, it is subdivided into the *PLCP preamble*, the *PLCP header* and the *data part*. The PLCP preamble consists of 128 one bits (*sync sequence*), followed by a constant value (start frame delimiter, SFD). The sync sequence and SFD allow the receiver to synchronize on the sender's clock (bit synchronization) and to determine the start of the frame. The signal field indicates the modulation type used in the data portion of the PPDU, while the length field indicates the length of the data portion in microseconds. The service field is not used yet. The CRC field contains a 16 bit cyclic redundancy check checksum which is computed from the three previous values. If the checksum is wrong or the signal field carries an unknown value, the MAC instance is signalled and can decide on aborting PPDU reception. It is important to note that while the data part can use different modulation types, the PLCP preamble and PLCP header fields are always transmitted with BPSK modulation. When the data part uses another modulation type, both transmitter and receiver must switch the modulation type within a PHY packet (more precisely, after the CRC field). With the PLCP preamble and the PLCP header a PPDU has a minimum duration of $128 + 64 = 192\mu$s.

The logical structure of the PMD part of the DSSS PHY is shown in Figure 3.4 (as taken from [123, chap. 6]). At the transmitter, first a *scrambling* step is applied to a PPDU. This is done to randomize the data, specifically to eliminate long runs of zeros or ones (like in the PLCP preamble). The *modulo-2 adder* actually performs the DSSS algorithm. The *transmit mask filter* restricts the spectrum of the spreader output to 22 MHz, while the QPSK/DPSK modulator produces actual waveforms. On the receiver side basically the transmitter operations are inverted, with the additional duty of acquiring the transmitters clock from the PLCP preamble. The receive process is based on a correlator for the Barker sequence.

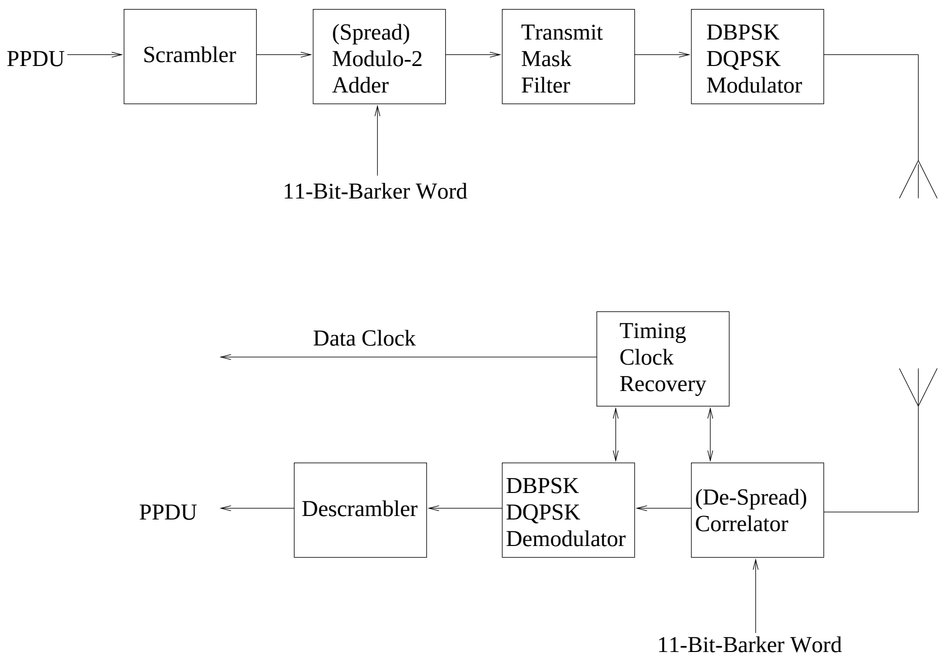

Figure 3.4: DSSS PHY Schematic

Other PHYs

The Infrared (IR) PHY uses infrared light as transmission medium. This restricts it to line-of-sight applications. In addition, infrared waves cannot propagate through walls. Data rates of 1 MBit/s and 2 MBit/s are supported, using pulse position modulation.

The FHSS PHY uses the 2.4 GHz ISM band. The user can specify more data rates than with the IR or DSSS PHY: from 1 MBit/s to 4.5 MBit/s in increments of 0.5 MBit/s, the standard prescribes the support of 1 MBit/s and optionally 2 MBit/s. The ISM band is subdivided into 79 different subbands of 1 MHz width (except in France, Spain, and Japan, where fewer subbands are used). The hopping process is slow (2.5 Hz), three hopping sequences are defined. The FHSS PHY has not gained as much market acceptance as the DSSS PHY.

The OFDM PHY uses three different bands in the 5 GHz range (U-NII bands): from 5.15 GHz to 5.25 GHz, from 5.25 GHz to 5.35 GHz and from 5.725 GHz to 5.825 GHz. Using different modulation types (BPSK, QPSK, 16QAM, 64QAM) and different code rates it offers bitrates of 6, 9, 12, 18 or 24 MBit/s, optionally 36, 48 or 54 MBit/s. The overall bandwidth is subdivided into 52 subbands, 48 of which are for data transmission, and 4 for carrier pilots. Additionally the PHY uses bit interleaving and convolutional error correcting codes to compensate narrowband noises on one or few subbands.

3.2.3 MAC Layer

This section provides a brief description of the IEEE 802.11 MAC protocol. We restrict the discussion to the simple data transmission procedures, leaving out management functionality, power saving issues, multirate support, privacy/encryption or details like the fragmentation/reassembly scheme.

MAC Services

Basically, the IEEE 802.11 MAC provides a connectionless best-effort service to its user (typically an LLC instance). However, for increasing transmission reliability, a bounded number of retransmissions is performed for all unicast frames.

When the MAC user wants the MAC to transmit a MAC MAC service data unit (MSDU), it passes a `MA-UNITDATA`.request primitive to the MAC layer. This primitive carries several parameters: the source and destination MAC address, the data block (up to 2304 bytes), and a priority value, indicating whether the MSDU should be transmitted in the contention or contention free period of the point coordination function (PCF), see below. Transmission in the contention free periods reduces MSDU losses by eliminating collisions. The last parameter describes, whether the MSDU should be transmitted strictly in order, i.e., the MSDU may not be postponed or accelerated relative to other MSDUs.[3] The reception of this primitive causes the MAC instance to generate a well-formed frame and to try to deliver this to the destination address.

For every `MA-UNITDATA`.request service primitive the MAC entity generates a `MA-UNITDATA-STATUS`.indication primitive, which tells about the success of the corresponding request primitive. As parameters the source and destination address, the priority value, the transmission status, and the provided priority and service class are passed. The transmission status can take several values, e.g, indicating successful transmission, unsuccessful transmission due to exceeding retry limits or MSDU lifetimes, notifications on illegal parameters in the request primitive and so forth. The priority value indicates the priority (contention or contention free) actually used for the frame. This, in general, need not be the same as the value wished by the user in the request primitive. Another example for this is the service class parameter.

Finally, the `MA-UNITDATA`.indication service primitive is passed from the MAC instance to the LLC upon successful frame reception. As parameters, it carries the source and destination address, the priority and service class, and the data block. There is no notification upon erroneously received frames.

The MAC services do not allow the user to express quality of service (QoS) requirements. It is not possible to assign a separate lifetime or a separate number of retransmissions for every packet, not even for the two priority classes. Instead, the corresponding protocol parameters are global for all packets. It is also not possible to specify a packets transmit power or its bit rate via the MAC interface.[4] Even the priority field of the `MA-UNITDATA`.request primitive is only a “proposal”, and the MAC is not strictly required to stick to it.

Basic Frame Transmission Procedures

Consider the case that station A has a unicast frame for station B. After A has obtained channel access (see next section), it transmits the frame. Station B is, upon successful reception, required

[3] The case of MSDU reordering applies to multicast and broadcast MSDUs when power saving mechanisms are used. It is not further discussed in this thesis.

[4] Clearly, a MAC implementation can try to adapt these to environmental conditions, but typically the user has no mechanism to influence this.

to send an immediate acknowledgement frame to A after a very short time (no more than SIFS, see below). The basic access mechanism ensures that between a frame and its corresponding ack no other transmission can take place. Therefore, the frame/ack sequence is considered to be *atomic*. If A does not receive an ack, it performs a retransmission, however, the maximum number of retransmissions is bounded by a configurable parameter.[5] The retransmissions have to contend for the channel by just the same way as initial frames. Furthermore, for every frame the MAC instance maintains a lifetime variable. If a frame is not successfully delivered within this lifetime, it is discarded.

IEEE 802.11 defines the RTS/CTS mechanism for attacking the hidden terminal problem. Consider again the case that A has a unicast frame for B. After A has obtained channel access, it sends a short RTS frame to B, indicating the time duration needed for the following CTS frame, the data frame and the acknowledgement. If B receives the RTS frame without errors and is ready to answer, it does so after a very short time (no more than SIFS) with a CTS frame, which contains the time duration needed for the remaining transmission cycle (data frame plus ack). If A successfully receives the CTS frame, it starts transmission of the data frame after a SIFS-bounded time. Finally, if B receives the frame correctly, it sends an ack after a SIFS-bounded time. By using only very short gap times in the whole frame exchange sequence, it is, by the basic access mechanism, atomic. Another station C hearing the CTS from A waits, whether A starts to transmit a data frame. If so, C defers any transmission, until A has finished the frame exchange (including ack). If not, C can transmit frames. Any station D hearing the CTS from B waits with its transmission, until the frame exchange finished. To implement this, C and D write the length information in the RTS and CTS frames into their NAV fields. This field is part of the virtual carrier sense mechanism, described below.

The RTS/CTS mechanism is inefficient for small sized frames. Therefore, for small frames (w.r.t. a configurable frame size) the RTS/CTS exchange can be switched off. The RTS/CTS mechanism is not used for broadcast / multicast frames.

Basic Access Mechanism (DCF)

The basic access mechanism is called *distributed coordination function (DCF)*. On top of it the centrally controlled *point coordination function (PCF)* is defined, described in the next section. The distributed coordination function (DCF) is a CSMA/CA mechanism, implemented by every station.

As a prerequisite, the DCF uses a *combined carrier sense* mechanism. This mechanism combines carrier sense information delivered by the PHY with a *virtual carrier sense* mechanism: Every station is required to maintain a NAV variable (network allocation vector), which conceptually is a timer. Most frame types (e.g., the RTS and CTS frames, the beacon frames of the point coordination function (PCF), or data frames) carry in their MAC header a duration field, indicating the time needed for the actual frames and the remaining frames of the frame exchange sequence. A station successfully receiving such a frame copies the duration field into their NAV value, if it is greater than the current NAV value. As long as the NAV variable contains a nonzero value, it is periodically decremented. The medium is considered busy, if either the PHY indicates a busy condition or if the NAV variable has a nonzero value.

The DCF employs different time durations and *inter frame spaces* for prioritization of different frame types:

[5] In fact, there is an additional distinction between short and long frames (w.r.t. a configurable bound), for each type different retry counters are maintained.

- The *slot time* is the length of one contention slot (20 μs for the DSSS PHY)
- The *short inter frame space* (SIFS) is the maximum time stations are allowed to wait before sending acks, CTS frames or during the contention free period in the PCF.[6] In the DSSS the SIFS value is 10 μs.
- The *priority inter frame space* (PIFS, used in the PCF) and the *distributed inter frame space* (DIFS) are defined as follows:

$$\begin{aligned} \text{PIFS} &= \text{SIFS} + \text{slot time} \\ \text{DIFS} &= \text{SIFS} + 2 \cdot \text{slot time} \end{aligned}$$

- Finally, the *extended inter frame space* (EIFS) is much larger.

The access procedure works as follows: when a station A wants to initiate transmission of a new or retransmitted frame, it senses the medium using the combined carrier sense mechanism. If the medium is sensed idle and remains idle for DIFS time[7], A starts to transmit its frame. If the medium is busy, A defers until the medium becomes idle for at least DIFS (or EIFS, if the last frame was erroneous[8]). After DIFS, the contention period starts. Station A generates a random backoff value for an additional deferral and puts this into the backoff timer. However, this is done only, if the backoff timer has a zero value. The random value is taken uniformly distributed from $[0, CW]$, where CW is the current contention window value. The CW variable is maintained by a modified truncated binary exponential backoff algorithm, taking the number of retransmissions of this frame into account. The minimum contention window is given by $CW_{min} = 32$, its maximum by $CW_{max} = 1024$. The CW variable is reset to CW_{min} after successful frame transmission, or if a frame is discarded after the maximum number of retries or exceeded lifetime.

In the contention period, station A performs carrier sensing after every slot. If the medium is sensed idle, the backoff timer is decremented by a slot time. If the medium is busy, the timer is not decremented and the backoff procedure is suspended. Station A has to wait again, until the medium is idle for at least DIFS, and then resumes the backoff procedure, however, using the old timer and not computing a new backoff value. Transmission is started, when the backoff timer reaches zero.

Contention Free Service / Point Coordination Function (PCF)

The *PCF* provides a contention-free frame transfer to stations, which can help to implement nearly isochronous services. The PCF adds a special station (point coordinator, PC) and some frame types to the basic protocol. While the presence of the PC, and thus the contention free service, is optional, every station must understand the frame types. The PC is always co-located with an AP. Hence, the PCF is available only in infrastructure mode.

The PCF defines a *superframe structure*, with variable- but maximum-length superframes. A superframe consists of a *superframe header*, followed by a *contention free period* (CFP) of variable length,

[6] In fact, it is also used for the second and subsequent data frames of a fragment burst (segmentation / reassembly protocol).

[7] As SIFS ¡ DIFS, by this behavior acknowledgement and CTS frames have priority.

[8] This way, other stations can ack a frame, which is considered erroneous by A. The view on the channel quality may be vastly different within a set of stations.

followed by a *contention period* (CP) of variable length. During the CP all stations, including the AP/PC operate in DCF mode. The superframe header is formed by a *beacon packet.* The beacon is a small packet sent by the PC to all stations. A part of the beacon packet is the maximum time duration of the CFP. Every station receiving the beacon packet puts this value into its NAV variable. The PC ends the CFP with a special frame (CFP-End), which allows all stations to reset their NAV and work in the DCF mode. The CP must be long enough for at least one frame exchange, e.g., for transmitting management frames with association requests, allowing stations to be put on the poll list.

The PC has a poll list of station addresses. Members of this list are polled during the CFP and can transmit their data contention free. Vice versa, the PC/AP can transmit frames during the CFP to its stations (downward frames). For efficiency reasons, it is possible to piggyback acknowledgements for frames sent by stations and poll-commands to the downward frames. When polled by the PC, each station A can transmit only one frame (if A's queue is empty, it answers with an ack frame). It does so after maximum SIFS time, ignoring combined carrier sense information. This frame is not necessarily targeted to the AP/PC, as is normally required in infrastructure mode. The station's frame can piggyback an ack for a previous downward frame addressed to A. A station is only allowed to send a data frame, if the frame transmission including ack can be finished before the maximum CPF length. If there is insufficient time, a station may answer with only an ack to the PC, however, indicating that it has a non-empty queue.

The polling scheme is not fully specified. Every station which wants to be polled, has to signal this to the PC/AP. When a station associates with its AP, it sets the *CF-Polling-Request* bit in the association request. The PC then includes this station into its poll list. The poll list membership ends upon disassociation or when the station reassociates itself without setting the CF-Polling-Request bit. The standard prescribes only the following rules for poll list handling: a) during every CFP at least one station must be polled; b) the poll list is traversed in the same order as the stations have associated themselves; c) if there is spare time during a CFP the PC may use it arbitrarily; d) the PC may decide to poll stations not in the poll list.

The PC maintains a local periodic timer. When this timer expires, the PC senses the medium. If it is idle for at least PIFS, it starts the CFP by sending the beacon frame. If the medium is sensed busy, the PC defers until the medium is idle for at least PIFS, and then sends the beacon frame. Hence, by using the PIFS inter frame space (which is smaller than DIFS), the PCF has priority over the DCF mode. Since the medium need not be idle when the timer expires, the CFP start times show some jitter, preventing strictly isochronous services. This jitter is denoted as *superframe stretching.* The PC transmits beacon packets regularly during the CFP. Each beacon indicates the maximum time to the end of the CFP.

During the CFP the RTS/CTS mechanism is not used. If a frame transmitted during CFP is not acked, retransmissions may be performed after the next poll or during the following CP. The PC may retransmit an unacked downward frame after a PIFS period. In general, the PC uses only SIFS inter frame spaces during the CFP, except on getting no ack on a downward frame or no response to a poll.

Chapter 4

PROFIBUS and Wireless PROFIBUS

In this chapter we introduce the relevant aspects of the PROFIBUS fieldbus systems (Section 4.1). We give a brief description of its architecture, the different physical layers, the link-layer services, the MAC- and link-layer protocol, and the most important properties with respect to realtime behavior. The link-layer services (and their interface) are the common denominator of the existing wired PROFIBUS and the wireless PROFIBUS targeted in this thesis, since one of the most important design goals of wireless PROFIBUS is to implement this interface to allow easy porting of application layer instances. The discussion is restricted to the transmission of user data and leaves out any management functionality.

In the next step we depict a general framework for wireless industrial communication systems (Section 4.2). It turns out that there are several means for integrating wired and wireless stations into a single fieldbus LAN, and for a certain class of fieldbuses (including PROFIBUS) a coupling on the MAC- and link-layer is advocated. The approach is to use the existing MAC- and link-layer protocol only for the wired stations and to use a specifically tailored protocol on the wireless side.

This framework is then specialized to the case of a wireless PROFIBUS (Section 4.3). This includes a description of the system under study. Of special importance, however, is the definition of the *realtime performance measures*, capturing the realtime- and reliability requirements, which protocols for wireless PROFIBUS should fulfill as good as possible.

This chapter also includes a discussion about the possibility to find a mapping between the PROFIBUS link-layer services and the IEEE 802.11 MAC services (and protocol). It turns out that both do not match well.

Finally, in Section 4.5 we give an overview on the related work on wireless fieldbus systems, wireless PROFIBUS and realtime transmission over IEEE 802.11.

4.1 The PROFIBUS Fieldbus System

The PROFIBUS is a german national standard since 1991 ([33], [34], some corrections in [135] and [133], the english versions being [55], [134]), which was also adopted as a european standard in 1996 [177]. A short tecnical description can be found in [136]. The PROFIBUS comes in two variants. The variant this thesis focuses on is a fieldbus system (compare Chapter 2), targeted to the coupling of intelligent field devices. A simplified variant belongs to the class of sensor/actuator buses. The PROFIBUS is designed to deliver real-time services in harsh, industrial environments. It has gained widespread usage, the "PROFIBUS International" user organization (*www.profibus.org*) states that there are more than 2 million PROFIBUS devices in more than 200000 installations in factory and process automation, making PROFIBUS the most popular fieldbus in Europe.

In this section the most relevant characteristics of the PROFIBUS are presented. Other descriptions of the PROFIBUS can be found in [11, 116, 30]. A performance assessment can be found in [89]. The real-time properties of the PROFIBUS are investigated in [169], [168, chap. 3, chap. 5].

It should be noted that the standards document and the corrections are written in plain text, not using any formal description techniques like, e.g, SDL or LOTOS [49], [24], [175].

4.1.1 Architecture

The PROFIBUS targets a wide range of applications in manufacturing and control. In order to avoid "one size fits all" solutions it offers different *profiles*, i.e. different sets of protocols and application layer services. The *communication profiles* denote different sets of protocols, while the *application profiles* distinguish between different sets of application layer services, and the *physical profiles* distinguish between different transmission technologies.

The PROFIBUS Decentralized Periphery (PROFIBUS-DP) communications profile (defined in 1993 [35]) is designed for coupling several simple sensors or actuators (or *slaves*) to a single controller, using a cyclic polling scheme with only a single *master* station. The master station is the only station controlling the medium. In this profile only the layers one (physical layer) and two (MAC and data link layer) of the OSI reference model are covered. Application layer entities use the link layer interface to obtain services from the link layer.

The PROFIBUS-FMS (Fieldbus Message Specification) communications profile is targeted at distributed control applications, with many intelligent nodes. The role of being a master station is changed over time between a set of stations. The PROFIBUS-FMS covers the layers one, two and seven of the OSI reference model (an overview of the communication-related part of the PROFIBUS-FMS protocol stack is shown in Figure 4.1, see [11, chap. 1.7.3]). The PHY covers physical transmission of single bits. The Fieldbus Data Link (FDL) layer provides MAC functionality and semi-reliable transmission of data (link layer functionality). The application layer is subdivided into the *application layer services* or Fieldbus Message Specification (FMS), and the *lower layer interface*, the latter providing e.g., connection management. The application layer services are similar to the MMS services (Section 2.3). The part of the PROFIBUS-FMS protocol stack dealing with data exchange is shown in Figure 4.1 (see [11, chap. 1.7.3]).

For both communication profiles the PROFIBUS is a LAN technology, where a number of stations share a single physical medium. The lack of routing functionality and proper addressing schemes

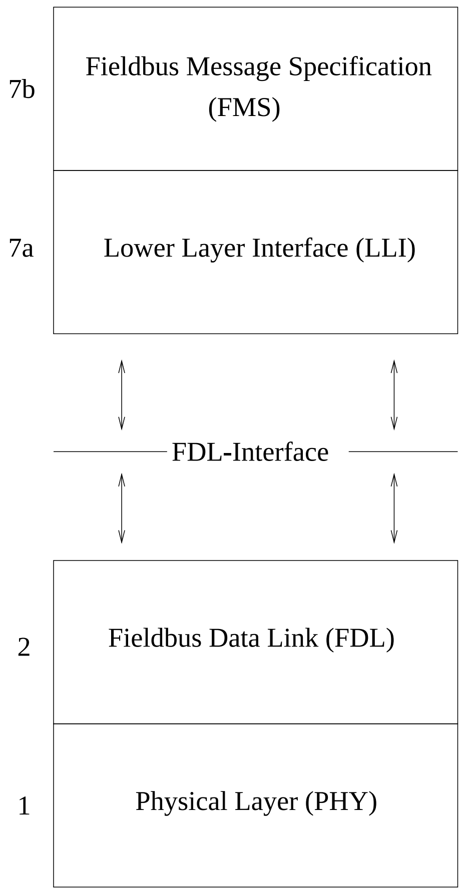

Figure 4.1: PROFIBUS protocol stack

prevent the coupling of different PROFIBUS LANs to form larger networks. A coupling of different PROFIBUS LANs has to be provided at the application layer (gateways).

Because of its generality the focus is exclusively on the PROFIBUS-FMS (or PROFIBUS for short), thus explicitly covering the case of multiple master stations.

4.1.2 Physical Layer

The PROFIBUS is defined for different physical layers: an RS-485 version, a fiber-optic version and a special version (IEC 1185-2) for use in explosible environments [136].

The RS-485 version uses serial transmission over a shielded twisted pair (STP) cable. Two cable types are standardized, however, one of them is not used anymore. The data is encoded with non-return to zero (NRZ) coding [62]. The transmission is byte-oriented, to support proper bit-synchronization the MAC layer adds start- and stopbits to a data byte, enforcing a minimum rate of signal level changes. The basic unit of the physical topology is called *segment* and has a bus structure, i.e. all stations attached to a segment see the same signals. The maximum number of stations on a single segment

Bitrate (kBit/s)	9.6	19.2	93.75	187.5	500	1500	12000
max. Distance (m)	1200	1200	1200	1000	400	200	100

Table 4.1: Relation between bitrate and distance for the RS-485 PHY and cable type A

is 32. Segments can be coupled using *repeaters*, a maximum of three repeaters is allowed between any pair of stations. The PROFIBUS addressing scheme restricts the number of stations in a single PROFIBUS LAN with multiple segments to 127. The set of available bit rates vs. the maximum distance between any pair of stations is shown in Table 4.1.

The IEC 1185-2 version uses a fixed bitrate of 31.25 kBit/s and Manchester coding [62] over STP cables. This version allows small devices to get their power supply from the cables. Furthermore, its electrical properties are such that hardware failures do not create sparks. Hence, the cable and field devices can be used in explosible environments. The cabling allows for bus and tree topologies, again the basic unit is a single segment. Segments can be coupled with repeaters. Special *segment coupler* devices allow the coupling of IEC 1185-2 segments to RS 485 segments, while *link devices* subsume a single IEC 1185-2 segment into a single RS 485 station.

The fiber optic version is targeted for use in harsh environments, i.e., where strong electromagnetical interferers are present. Different types of fiber can be used (e.g., monomode, multimode), the physical topology is a ring or a passive star coupler is used. Fiber optic segments and RS 485 segments can be coupled with special devices.

4.1.3 Link Layer Services

The link layer offers four service types to the upper layers: three asynchronous service types and a cyclic one (or polling service). All services except the cyclic service allow the user to distinguish between two priorities: high priority (i.e., important) data and low priority (less important) data. The high priority class is devoted to the exchange of time sensitive and safety critical sporadic data (e.g., alarms), the low priority class is used for everything else. The cyclic services belong to the low priority class, however, the bandwidth assignment rules are different than for ordinary low priority data.

While not explicitly specified in the standard, for the asynchronous service types it is reasonable to assume that within each priority class the service requests are processed in first in first out (FIFO) order, regardless of the service type. In other words, conceptually there are two request queues, one for low priority requests and the other for high priority requests. Every request primitive, regardless of its service type, is sorted into one of these queues according to its priority. For the cyclic service type a kind of polling-table is maintained.

For the discussion about services the terminology of the OSI reference model is used (see [165, chap. 1]): services are accessed via a service access point (SAP), and typically four *service primitives* are involved in the communication between *service provider* and *service user*: request, indication, response and confirmation. As a convention, for a service primitive belonging to a service A we will write `A`.primitive, e.g., `FDL_DATA_ACK`.request. For all services there is a distinction between the roles of the *initiator station* and the *responder station*. Service handling is initiated when an application layer entity (called *FDL* user) at the initiator station issues a request primitive. Furthermore, a protocol

Service	SDA	SDN	SRD	CSRD
Immediate Ack	yes	no	yes	yes
Connection Oriented	no	no	no	local connection setup
Data in Ack	no	-	yes	yes
Occurence	sporadic	sporadic	sporadic	cyclic

Table 4.2: PROFIBUS FDL-Services

data unit (PDU) is the means of communications between peer entities.

A key attribute of several service primitives are address informations. The address information consists of an *address byte* and an optional service access point (SAP) [33, chap. 4.8.2]. PROFIBUS supports three types of addresses: a *unicast address* has an address byte with a value between 0 and 126 (therefore, the number of stations in a PROFIBUS LAN is restricted), and optionally a SAP can be used. For *multicast addresses* and *broadcast addresses* the address byte has a value of 127 and the group address / multicast address is selected via the SAP value. In fact, the address scheme can be more sophisticated (address extensions, segment addresses [33, chap. 4.8.2]), but this is not relevant to this work.

For every request primitive exactly one confirmation primitive is created, which contains information about the success of the request. By the FIFO assumption discussed above, within one priority class the confirm primitives occur in the same order as the corresponding request primitives. The standard prescribes that once packet transmission for a request has started, it cannot be interrupted by other requests. Stated differently: the PROFIBUS does not allow for preemption in handling requests. This property is called *atomicity property.*

The PROFIBUS utilizes no respond primitives. All services are designed in a way that a responder's station's FDL instance can generate answer frames immediately, without interacting with higher layer instances.

The PROFIBUS services are explained in the following sections, a short summary of the services can be found in table 4.2.

SDA: Send Data with Acknowledge

The send data with acknowledge (SDA) service is basically a semi-reliable acknowledged datagram service. A sketch of the interactions is shown in Figure 4.2.

The FDL user at the initiator station starts this service. He prepares a data block of up to 246 bytes length, chooses a priority, provides destination address byte and destination service access point (DSAP) for the target station and its own SAP as source service access point (SSAP). All this information is passed with the `FDL_DATA_ACK`.request service primitive to the FDL instance via the local SAP. The destination address is required to be a unicast address.

Some time later the local FDL instance attempts to send this data block within a single frame[1] to the responder station. The responder has to acknowledge receipt of the frame using an immediate MAC-layer acknowledgement. The acknowledgement carries no data. If the initiator receives no ack

[1]called *telegram* in the PROFIBUS standard, we use both terms interchangeably.

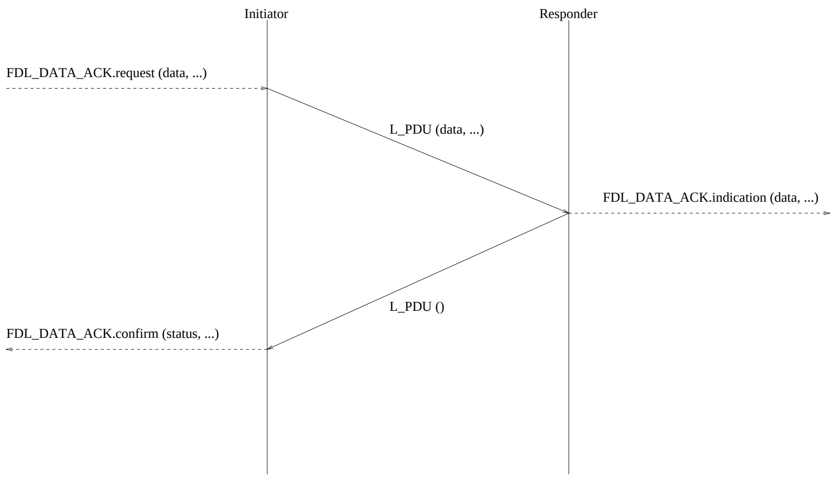
Initiator
Responder
FDL_DATA_ACK.request (data, ...)
L_PDU (data, ...)
FDL_DATA_ACK.indication (data, ...)
L_PDU ()
FDL_DATA_ACK.confirm (status, ...)

Figure 4.2: Interactions of SDA service

within a pre-specified time (called *slot time*, T_{SL}), it repeats the frame. The overall number of trials is upper-bounded. When the result of the transmission trial is known, i.e. an acknowledgement is received or the maximum number of trials is exhausted, the local FDL user is informed about this result with the FDL_DATA_ACK.confirm primitive. Additionally, this primitive contains the address information and prioritiy level passed with the corresponding FDL_DATA_ACK.request primitive.

When the FDL instance at the initiator has started processing a FDL_DATA_ACK.request primitive X, it is required to get the result as fast as possible, i.e., it is not allowed to process other requests Y or to pass the token (see Section 4.1.4) while processing X.

At the responder station, at the first reception of a data frame a FDL_DATA_ACK.indication primitive is generated, the data is passed as a parameter. If the responder identifies a frame as retransmitted (by the alternating bit protocol, see Section 4.1.4), it is acknowledged and silently discarded. The FDL layer ensures that for every FDL_DATA_ACK.request primitive at most one FDL_DATA_ACK.indication primitive and exactly one FDL_DATA_ACK.confirmation primitive is generated.

SDN: Send Data with No Acknowledge

The send data with no acknowledge (SDN) service is basically an unacknowledged datagram service, applicable to unicast- and multicast-/broadcast-addresses. A sketch of the interactions is shown in Figure 4.3.

The FDL user at the initiator passes a FDL_DATA.request primitive to the FDL instance. This primitive carries the same parameters as the FDL_DATA_ACK.request primitive described in the previous section. The destination address may be a unicast, a multicast or broadcast address. At a later time the MAC instance transmits one data frame with the data and then produces a FDL_DATA.confirm primitive for the local FDL user. Every other station which is addressed and which successfully receives the frame, creates an FDL_DATA.indication primitive and delivers the data to its FDL user, hence, not confirming any proper reception. No station is allowed to send an acknowledgement.

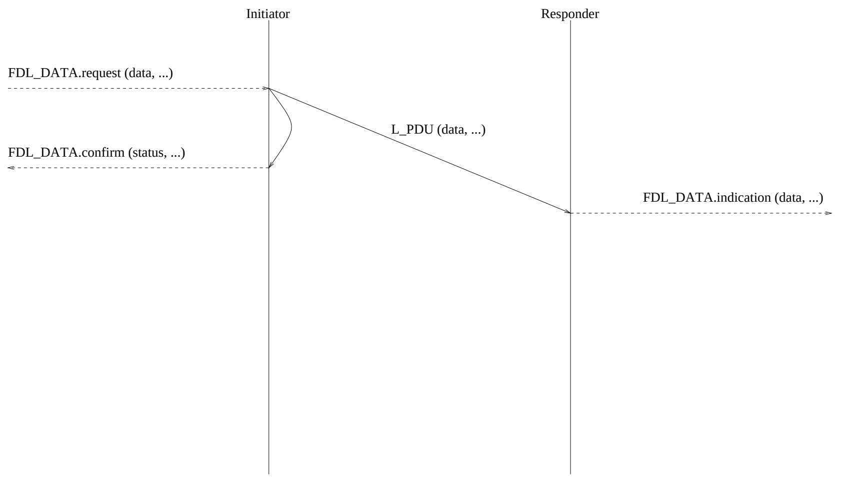
Initiator
Responder
FDL_DATA.request (data, ...)
L_PDU (data, ...)
FDL_DATA.confirm (status, ...)
FDL_DATA.indication (data, ...)

Figure 4.3: Interactions of SDN service

SRD: Send and Reply With Data

The send and receive data (SRD) service is basically the same as the SDA service, however, the acknowledgement sent by the responder can carry data. A sketch of the interactions is shown in Figure 4.4.

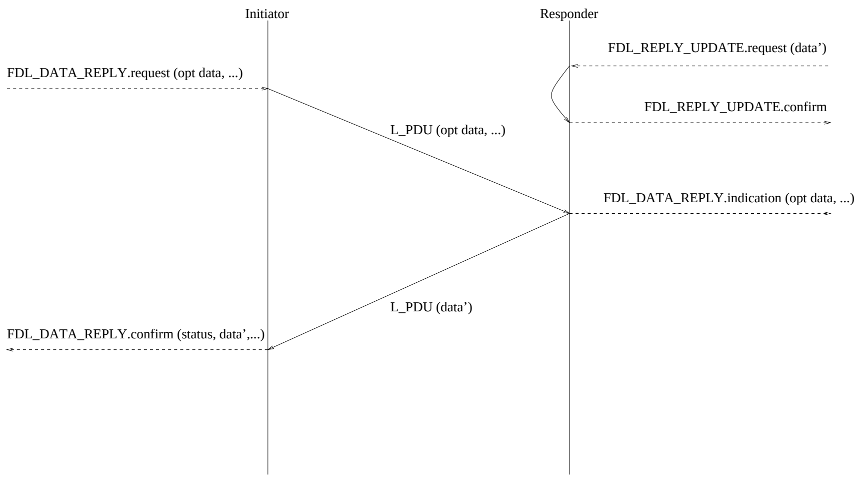
Initiator
Responder
FDL_REPLY_UPDATE.request (data')
FDL_DATA_REPLY.request (opt data, ...)
FDL_REPLY_UPDATE.confirm
L_PDU (opt data, ...)
FDL_DATA_REPLY.indication (opt data, ...)
L_PDU (data')
FDL_DATA_REPLY.confirm (status, data',...)

Figure 4.4: Interactions of SRD service

The FDL user at the responder station can place a piece of data (up to 246 bytes) along with an associated SAP and a priority into an internal buffer of the FDL instance. This is done using the `FDL_REPLY_UPDATE`.request primitive. After filling the buffer the FDL instance generates the corresponding `FDL_REPLY_UPDATE`.confirm primitive, indicating the status of the operation. An additional parameter (*reuse information*) of the request primitive indicates, whether the piece of data is used for answering a single request or multiple requests. The FDL user at the responder station can write data into this buffer at arbitrary times.

The remaining service proceeds in much the same way as the SDA service (using the `FDL_DATA_REPLY`.request, `FDL_DATA_REPLY`.confirm, and `FDL_DATA_REPLY`.indication primitives). However, if the responder generates its acknowledgement, it checks whether there exists a buffer for the requested SAP, and, if so, writes the contents of this buffer into the immediate ack frame. This data is passed to the FDL user at the initiator station with the `FDL_DATA_REPLY`.confirm primitive. If the buffer contents should be transmitted only for a single request, the buffer is deallocated. Necessary frame repetitions, e.g., due to a corrupted ack frame, are handled by buffering the ack frame as long as necessary (Section 4.1.4).

If the acknowledgement frame is erroneous, the initiator station retransmits its request frame. The responder then retransmits the last frame without reconstructing it. If the FDL user at the responder station uses the `FDL_REPLY_UPDATE`.request primitive between the first and second ack frame, it has no effect on the retransmission.

CSRD: Cyclic Send and Reply With Data

The cyclic send and receive data (CSRD) service is the most complicated service of PROFIBUS, and not discussed in full detail. The realtime performance measures defined in Section 4.3.1 focus entirely on high priority messages, while the CSRD service is a priori of low priority.

At the beginning of this service the local FDL user gives a list with unicast station addresses to the local FDL instance (*poll list*). The local FDL instance maintains for every station in this list a *marker* and an optional buffer. The local FDL user can put a block of data (up to 246 bytes) into this buffer, along with SAP, priority and reuse information, i.e., the buffer contents can be sent only once or multiple times. On the responder station the FDL instance can also allocate buffers, which are handled just as for the SRD service.

The stations in the poll list are polled in a round robin fashion. To every station in the poll list a request frame is sent, which may carry some data, if the associated buffer exists and is non-empty. If the buffer contents should be transmitted only once, the buffer is deallocated. The responder station is required to send an immediate acknowledgement, which also can carry the contents of a corresponding buffer. For every request the initiator station performs a bounded number of retransmissions. If all attempts fail, the responder station is marked as "dead" by modifying the marker variable accordingly. At some later time, the dead station is pinged again by the local FDL instance, however, no retransmissions are performed. If the dead station responds, it is marked as "alive" and the local FDL instance proceeds with its normal operation.

4.1.4 MAC- and Data Link Protocol

On the MAC layer the PROFIBUS combines two principles: master/slave communication for data exchange and token passing for managing the right to initiate packet transmissions. Furthermore, two types of stations are distinguished: *active stations* can participate in the token passing process, while *passive stations* cannot. An active station which currently owns the token, is called a *master station*, all other stations have the role of *slave stations*. Only the master station is allowed to initiate a data transfer, a slave station may only transmit data, if it has to send an immediate acknowledgement to a frame directed to itself. Hence, the `FDL_DATA_ACK`.request, `FDL_DATA`.request and `FDL_DATA_REPLY`.request service primitives can only be issued on active stations.

Token Passing and Ring Maintenance

The PROFIBUS token passing protocol is similar to the IEEE 802.4 Token Bus protocol [71] and uses a broadcast medium. A logical ring is formed by ascending station addresses. The address space is small, a station address is in the range of 0 to 126. Every station (denoted as TS: This Station) knows by the ring maintenance mechanism explained below the address of its logical successor (NS: Next Station) and its logical predecessor (PS: Previous Station). If TS receives a valid token frame with TS as destination address, it checks whether it was sent by its PS. If so, the token is accepted, otherwise the frame is discarded. In the latter case, if the same token frame is received again as the very next frame, the token is accepted and the token sender is registered as new PS and the *list of active stations (LAS)* is updated, see below. In any case, after accepting the token TS determines its *token holding time* THT (according to a simplified variant of the timed token protocol with target token rotation time T_{TTRT}) and is allowed to send some data during the THT. If there is no data anymore or THT expires, TS is required to pass the token to NS by sending a token frame. This must be done even if TS is the only ring member (NS = TS = PS), and TS must accept the token in the same way as if PS $\neq$ TS. After sending a token frame, TS listens on the medium for some activity. This can be the reception of a valid frame header (indicating that NS has accepted the token) or reception of some erroneous transmission. However, TS listens on the medium only for the slot time T_{SL} which is typically chosen very tight, e.g., in the range of 100 μsec to 400 μsec.[2] If this time passes without any medium activity the token frame is repeated (clearly, active stations are required to react fast enough on token frames, otherwise collisions occur). If there is again no activity, and a third trial is also unsuccessful, NS is assumed to be dead and TS determines the next station in the ring (i.e. the successor of NS), makes this the new NS and tries to pass the token to it, following the same rules. The new station can be determined from the LAS, which is updated by the ring maintenance mechanism, as explained below. If TS finds no other station, it sends a token frame to itself.

A special protocol rule is the following: TS must read back from the medium bit by bit all token frames it transmits (*hearback*), in order to detect a defective transceiver and to resolve collisions (see below). If TS encounters a difference the first time, it waits for some response (which indeed may occur due to undetected errors in the token frame, see below). If there is no activity on the medium it repeats the token frame. If TS again encounters a difference, it discards the token immediately and removes itself from the ring, behaving as newly switched on and "forgetting" all knowledge previously obtained. The rationale for this is the assumption that the transceiver is is faulty and its results are not trustworthy.

The ring maintenance mechanism works by two different means. First, if a station is newly switched on, it is required to listen passively on the medium, until it has received two successive identical token cycles and thus has a valid view on the whole logical ring (referred to as *listen token* state). During this time it is not allowed to send or answer to data frames or to accept the token. Every station address found in a token frame belonging to this two cycles is included into the LAS. After building a valid view the station can enter the ring if another station passes the token to it. The second rule requires every station to inspect every correctly received token frame and to include the source and destination address into the LAS. An important rule here is the following: if TS feels itself as already included in the logical ring and reads a token frame, where TS is "skipped" (i.e. the address of TS lies truly within the address range spanned by sender and receiver of the token frame) it removes itself from the ring and behaves as newly switched on.

[2]These numbers are taken from a database with configuration data found at *www.profibus.org.*

In order that another station can pass the token to a station newly switched on, every station a maintains a *gap list (GAPL)*, containing all possible station addresses between a and its NS b. A station a is required to periodically poll all addresses in its GAPL by sending a Request-FDL-Status frame to a single address c and waiting one slot time T_{SL} for an answer, which indicates c's current status (ready / not ready for the ring). A station which tries to detect two identical token cycles will respond with a "not ready" status. Within every token cycle a polls at most one station address in its GAPL. If a station in the GAPL responds as "ready", a will change its NS, shorten its GAPL, update its LAS, and then send a token frame to the new station. The period for scanning the GAPL is created by a special timer (*gap timer*), which is set as an integral multiple (*gap factor*, the standard requires values between 1 and 100) of the target token rotation time T_{TTRT}.

For leaving the ring it suffices to just stop all transmissions. In this case PS will detect the station loss when unsuccessfully trying to pass the token to TS.

A special mechanism is used for the very first ring initialization or to handle token loss due to system crash of the current token owner: every station listens permanently on the medium. Every time the medium goes idle, TS starts a special timer, the *timeout timer*, with the *timeout value* T_{TO}. The timer is resetted each time the medium goes busy. If the timer expires (no transmission on the medium for some time), TS "claims the token", i.e. it starts with behaving as the current token owner and performs some frame transmission: it sends data frames or passes the token to its current NS. If TS was not in the listen token state when the timeout timer expires, there is no change in its internal state, specifically in its LAS, NS and PS. In the other case, since the station has not yet a valid view on the ring, it assumes the ring to be empty and itself being the only member of LAS.

The timeout value depends linearly on the station's address n:

$$T_{TO}(n) = (6 + 2 \cdot n) \cdot T_{SL}$$

This can lead to collisions, and the hearback feature is necessary to resolve them. One situation where collisions can occur is the following: consider that in an empty ring two stations are newly switched on at different times, such that their timeout timers expire simultaneously. When both stations start transmitting token frames, the resulting collision induces hearback errors. Both stations retire from the ring and stop transmissions, while simultaneously starting their timeout timers. Because of the different station addresses the timers expire at different times, and now a valid ring can be built up without further collisions.

Bandwidth Allocation

After an active station receives the token, it computes its *token holding time* (T_{HT}) by subtracting the *real token rotation time* T_{RR} (measured time between two token arrivals) from the configured *target token rotation time* T_{TTRT}:

$$T_{TH} = T_{TTRT} - T_{RR}$$

The real token rotation time is measured continuously as the time between successive token arrivals. A timer (*token holding timer*) is started with the computed T_{TH}.

After receiving the token a master station is allowed to handle one high priority frame including necessary retransmissions, regardless of the value of T_{HT} (the handling of a frame including retransmissions is denoted as *cycle*). Hence, this feature can only be exploited in the SDA, SDN and SRD services, since the CSRD service is performed with low priority.

If further high priority messages are available and the THT is not expired, the station continues with high priority cycles. Except from the first cycle, it is required that at the start of a new cycle of low or high priority, the token holding timer must not be expired. If the timer expires meanwhile, the station is allowed to finish the cycle, including all retransmissions. Then the station must pass the token to its NS.

The station must first perform all available high priority cycles, until the timer expires or the high priority queue empties.[3] In the latter case the station the station may proceed with low priority or cyclic frames, if the THT is not expired. After having started, the low priority service is nonpreemptive, i.e. newly arriving high priority requests are handled upon the next token arrival. It runs until there are no frames to transmit or until the token holding timer expires.

The low priority service starts with traversing the poll list of the CSRD service. If all stations in this list are polled and the token holding timer is not expired, the station proceeds with handling asynchronous low priority cycles (SDA, SDN and SRD services) until timer expiration. If the poll list can be traversed within one token cycle but there is no remaining time for low priority data, the low priority queue is handled upon the next token arrival where after processing high priority requests there is still time for low priority cycles. After this the poll list is traversed again.

If traversing the poll list takes more than the THT would permit, the remaining list is handled in the subsequent token cycles, without interruption by asynchronous low priority frames. The latter are handled if the poll list is fully traversed.

Some special frames for ring maintenance (Request-FDL-Status) are treated as asynchronous low priority frames.

The protocol described so far is a variant of the well-known *timed token protocol* which is also used in the FDDI and IEEE 802.4 standards. It is known to be capable of transmitting multimedia data [117], [3].

Data Link Protocol

The PROFIBUS data link layer provides a semi-reliable service for the SDA and SRD services, with a bounded number of retransmissions, given by the global *max_retry* parameter. The necessary feedback is provided by immediate MAC layer acknowledgements, which may for the SRD service also carry some data.[4] By the protocol all trials are performed subsequently, no other frame transmission or token passing is allowed in the meantime. If the result of a frame transmission is known (successful reception of an ack or *max_retry* trials without receiving an ack) the upper layers are notified with a confirmation primitive. In order to allow the receiver to distinguish between new and retransmitted frames, a variant of the well-known alternating bit protocol is used [10].

The FDL instance of the initiator keeps for every possible target station a state variable (distinguishing between "dead" and "alive", [33, chap. 4.1]) and a *frame count bit (FCB)* [33, chap. 4.8.3]. The state variable is changed according to the following rule: if the target station is "alive" and does not answer on any trial belonging to a cycle (i.e, it does not ack the request frame and all of its *max_retry* transmissions), it is marked as dead. If a cycle is targeted to a dead station, no retransmissions are

[3]The standards document [33] does not state explicitly whether the service is exhaustive (all requests are processed) or gated. In this work exhaustive service is assumed.

[4]The acknowledgement frame in the CSRD service may also carry data, but uses a different retransmission scheme.

performed within the CSRD service. If a station marked as dead answers again, it is marked as alive.

The FDL instance of the responder keeps for every possible station address a FCB. Furthermore, it maintains a global buffer for the last acknowledgement frame (*acknowledgement buffer*).

The FCB allows the responder station to distinguish between new frames and retransmitted frames. It is transmitted as part of the frame header (the frame formats are described in the next section), along with a *frame count valid bit (FCV)*, which indicates, whether the FCB is valid. When the initiator sends a frame to a responder the first time, or if the responder wakes up after it was dead, both stations have to synchronize on a FCB value. For doing this the initiator sends frames with FCV = 0 and FCB = 1 to the responder, however, such frames are not retransmitted. After the responder has answered, the initiator sends all further frames with FCV = 1 and inverts the FCB after every finished cycle.

If the responder receives a frame with FCV = 1 and FCB = x, it checks, whether this FCB was different from that stored in its table. If so, the cycle is considered as successfully finished, the FCB is stored in the table, a new acknowledgement frame is generated, stored in the global buffer and transmitted to the initiator, and an indication primitive is generated for the responder's FDL user. If the responder receives a frame from a station with address b after receiving a frame with FCV = 1 from station a, it considers the last cycle of a as finished. If the responder receives two successive frames from the same station, both with FCV = 1 and the same FCB, it concludes that the last frame is a retransmission. In this case only the contents of the acknowledgement buffer is retransmitted, no further action is taken.

Frame Formats

The PROFIBUS supports five different types of frames, shown in figure 4.5. With the exception of the one byte long *short acknowledgement (SC)* frame, all other frame types start with a *start delimiter* (each frame type with a different one), and carry at least a *destination address (DA)* byte and a *source address (SA)* byte. The *frame control (FC)* byte has different meanings when transmitted from initiator to responder and vice versa. When sent by the initiator, the FC byte carries control information: FCB and FCV, a service type tag (SDA, SDN, SRD, management frame), or the answer code in case of acknowledgement frames (positive or negative acknowledgement, error codes for indicating malformed request frames, lack of memory etc.) [33, chap. 4.8.3]. The *frame check sequence (FCS)* byte is a simple checksum, computed for the frame with no data, the frame with fixed length data and the frame with variable length data. The FCS byte is the sum modulo 256 of all bytes located between the DA-field (inclusive) and FCS-field (exclusive) of the frame. The short acknowledgement and the token frame have no checksum at all.

The RS 485 version of PROFIBUS uses serial transmission with NRZ coding (see Section 4.1.2). Every byte of data is transmitted with eleven bits: one startbit, one stopbit, eight data bits and a parity bit (see Figure 4.5).

4.1.5 Important Properties of the PROFIBUS

We shortly summarize the important features of the PROFIBUS with respect to realtime and reliability behavior. In the absence of transmission errors the protocol can give the following guarantees:

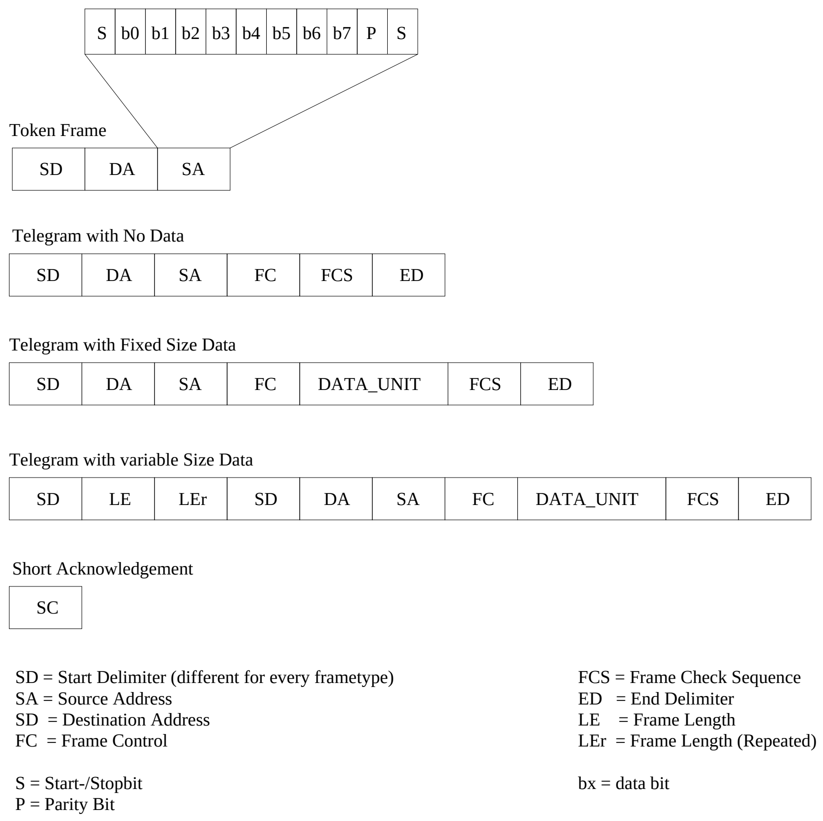

Figure 4.5: PROFIBUS frame formats

- For high priority messages for every station the handling of at least one request within a bounded time can be guaranteed. An upper bound to this time is given by the target token rotation time T_{TR} plus N times the maximum duration of a high priority message exchange (including the bounded number of retransmissions), where N is the number of active stations in the ring [33, p.26]. This is due to the rule that every station may handle at least one high priority frame exchange per token arrival. Stated differently: for every high priority request primitive the time until the occurence of the next high priority confirmation primitive is upper bounded, regardless whether the primitives belong together.

- The time needed for ring initialization after newly switching on the first stations is upper bounded. The reason is that for the expiration of the timeout timer two cases occur: it expires only in a single station, or it expires in two or more stations simultaneously. In the former case the single station starts to transmit, and all other stations reset their timeout timers. The first station builds a valid ring with only itself being member. Upon transmitting the first token frame with source and destination address being its own address, all other stations get the same view on the ring and wait patiently for being included. In the latter case the collision can be resolved as described in Section 4.1.4.

- The timed-token protocol gives long-term fair distribution of bandwidth between stations [76, chap. 23]. The bandwidth splitting between high priority and low priority data is performed in a purely local manner in every station.

- The FDL offers a semireliable service, for every request exactly one confirmation primitive is generated, indicating the success of the request. Furthermore for every request at most one indication is generated. Within one priority class the confirms have the same order as the requests, the indications are a subset with preserved order.

It should be noted that most of the bounds, e.g., the T_{TR} value and the ring initialization time, depend on the number of stations in the ring. However, since this number is fixed upper bounded by the limited address range, an absolute upper bound can be given.

4.2 Wireless Industrial Communication Systems

The creation of a wireless Industrial Communication System (ICS) and specifically of wireless fieldbus systems is nowadays "up in the air", due to the ever increasing acceptance of WLAN technology and their attractive features, like mobility and reduced cabling need. Of specific interest is the idea of a *wireless fieldbus extension*, i.e., to create a possibility to integrate wireless stations in already existing, wired fieldbus systems to form a single LAN.

For wireless fieldbus extensions we propose a classification, which reflects the architectural characteristics of fieldbus systems just covering layers 1, 2 and 7 of the OSI reference model (see Sections 2.2 and 2.3):

- *wireless repeater scenario*: all stations are attached to a cable and have wired transceiver, and just a piece of cable is replaced by a wireless link. In this scenario no station has to be aware of the wireless link. From now on we use the term *wireless station* to denote a station with a wireless transceiver.
- *Wireless bridging scenario*: Integration solely on the physical layer. Wired stations are attached to a cable, wireless stations have a radio frontend, and a bridge-like device translates the different framing rules used on wired and wireless media into each other. The MAC- and link-layer protocol and all application layer protocols are the same for both wired and wireless stations.
- *Integrated scenario*: Integration at the MAC- and data link-layer, with two different MAC- and link-layer protocol stacks on both sides, but with fixed link-layer interface.
- *Wireless gateway scenario*: Integration at the application layer.
- Some mixture of these scenarios.

In this thesis we focus on the integrated scenario, since, for a wireless PROFIBUS it appears to be an attractive choice:

- Both the wireless repeater and the wireless bridging scenario require the transport of PROFIBUS MAC frames over a wireless medium. In Chapter 5 it is shown that, however, the error behavior of a wireless link can seriously harm the realtime capabilities of PROFIBUS.
- In the wireless gateway scenario a tight coupling of the timing behavior between wired and wireless stations is hard to achieve.

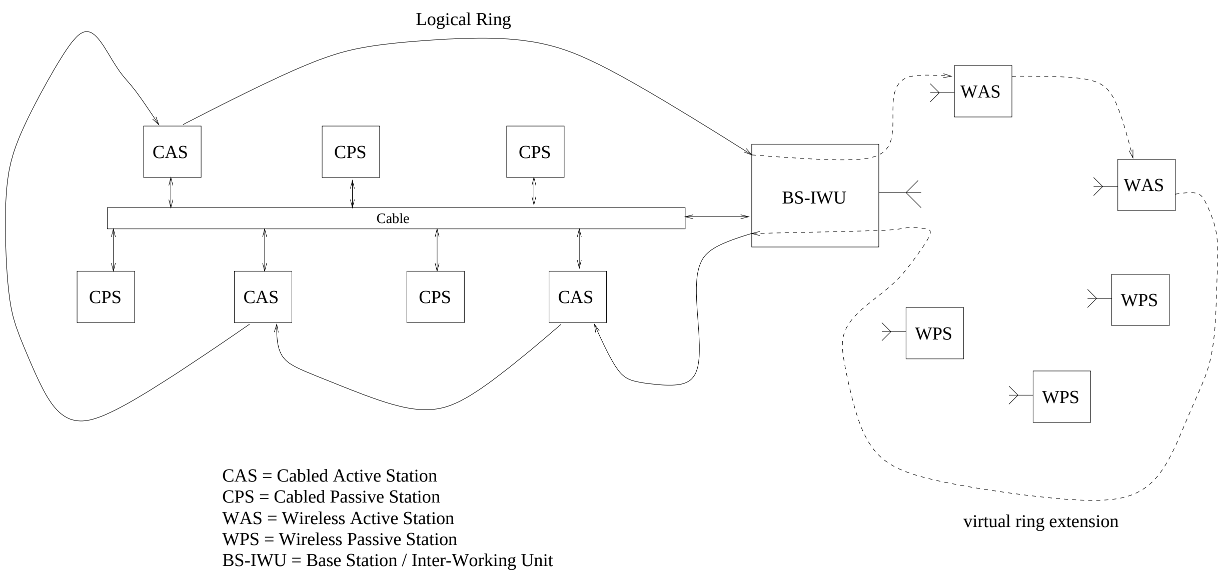
Logical Ring
CAS
CPS
CPS
Cable
BS-IWU
WAS
WAS
CPS
CAS
CPS
CAS
WPS
WPS
WPS
CAS = Cabled Active Station
CPS = Cabled Passive Station
WAS = Wireless Active Station
WPS = Wireless Passive Station
BS-IWU = Base Station / Inter-Working Unit
virtual ring extension

Figure 4.6: Integrated PROFIBUS LAN

4.2.1 Integrated Scenario: General Considerations

There is a certain class of fieldbus systems employing a decentralized MAC protocol with explicit token passing, used to form a *logical ring*. Besides the PROFIBUS also the IEEE 802.4 Token Bus and the multi-master mode of BITBUS belong to this class (see Section 2.3). The basic situation for this class of protocols is sketched in Figure 4.6 for the specific case of PROFIBUS. The distinction between active and passive stations refers to the ability to take part in the token passing process: active stations can participate, passive stations not.

In the integrated scenario it is assumed that on the wired part the given protocol is used, while on the wireless part a specifically tailored MAC- and link-layer protocol is used. The need for protocols specifically tailored to the wireless medium can be easily justified by the properties of the wireless medium, as discussed in Section 3.1 and Chapter 6. While the MAC- and link-layer protocol on the wireless side will be different from that of the wired side, the link-layer interface should be kept fixed. This requirement allows easy porting of application layer software.

The approach of using the unchanged protocol on the wired side and using another protocol on the wireless side has important consequences, specifically for the role that the coupling element (denoted as *base station / interworking-unit (BS-IWU)*) has to play. In general, it has at least the following tasks:

- Give the wired stations a consistent view on the logical ring.
- Give the wireless stations a consistent view on the logical ring.
- Synchronize both MAC protocols.
- Translate possibly different link-layer protocols.

- Forward frames from the wired to the wireless side and vice versa.
- Keep track about the wireless stations belonging to "its" LAN.
- Optionally performing proxy operations for certain types of wireless terminals.[5]

4.3 Integrated Scenario for Wireless PROFIBUS

This thesis proposes to use a polling-based protocol on the wireless side, for the sake of realtime-performance, as defined in Section 4.3.1. This class of protocols assumes a central controller, granting access to the wireless medium. Since this centralized protocol has to be synchronized with the distributed PROFIBUS token passing MAC protocol, and since furthermore the BS-IWU is the only instance capable of knowing the state of both sides (compare Figure 4.6), it is natural to put the central controller functionality into the BS-IWU.

With this approach, the BS-IWU is a fairly complex device performing several functions:

- Frame forwarding between the wireless and wired side. In PROFIBUS data frames require an immediate acknowledgement, for which the time bounds are typically sharp (see Section 4.1.4). This suggests using cut-through forwarding instead of store-and-forward.
- MAC-Integration (*mimikry functions*), and optionally link-layer integration (translation of the PROFIBUS alternating bit protocol to the wireless link-layer protocol).
- Central scheduler for the wireless stations.
- Signalling functions: mobility support, authentication, station registration, address allocation.

The mimikry functions are due to the concept of keeping explicit token passing and logical ring maintenance away from the wireless part. Therefore, the BS-IWU is required to act on the wired segment on behalf of the wireless stations. This includes: generating token frames on behalf of wireless stations, sending and answering the frames necessary for including new stations into the logical ring, reaction on certain frames asking for static information about wireless stations (e.g., vendor name, product id), and to prevent the wired medium becoming idle when the token is logically in the wireless part (timeout timer).

A more detailed discussion of a possible architecture for a wireles PROFIBUS based on the integrated scenario can be found in [191].

4.3.1 System under Study and Realtime Performance Measures

In this section and Section 4.2.1 a framework for wireless fieldbuses in general, and more specific, for a certain class of fieldbuses employing token passing protocols was depicted. Within this framework now the system of interest and the performance measures of interest (*realtime performance measures*) are described, along with their associated load models. This description, although originating from the properties of the PROFIBUS, is more abstract and fits both the PROFIBUS protocol described in Section 4.1 and the polling-based protocols described in Chapter 7.

[5]Consider for example a small, energy-constrained wireless sensor observing a slowly varying physical process. The BS-IWU can poll this sensor from time to time and answer other stations requests on behalf of it.

System Description

Consider a scenario with N wireless terminals (WT). All these terminals are assumed to be active stations in the sense that they are willing to acquire transmission rights. In the case of PROFIBUS this means that they want to participate in the token passing process. All data transmissions are between the WT's. For this work a WT consists mainly of the MAC- and link-layer, which is attached to the physical medium on the one hand and to the upper layers on the other hand. The interface offered to the upper layers is denoted as *link-layer interface*. At each WT a number of *traffic sources* generating *requests* and consuming confirmations can be attached to its link-layer interface.

Although the PROFIBUS offers more link-layer services, the service of interest is the acknowledged datagram service, specifically the PROFIBUS SDA service (see Section 4.1.3). This choice is convenient, since:

- For the unacknowledged datagram (SDN) service there are no guarantees anyway.
- In the (C)SRD services the acknowledgement may carry data. However, this adds no new quality to the realtime behavior of acknowledged services, and is left out for simplicity.

A traffic source generates a *request* and hands this over to the MAC- and link-layer protocol via the link-layer interface. The MAC- and link-layer protocol tries to successfully transmit this request, however, the number of trials is bounded (*max_retry* parameter). In any case, when the fate of the request is known, the MAC- and link-layer instance generates a *confirm* primitive and passes this to the traffic source via the link-layer interface. There is no segmentation and reassembly scheme applied to the requests.

There are two types of traffic: *low priority* and *high priority* traffic: the high priority traffic is meant for transmission of safety critical data (like alarms), all other data types (e.g., periodic process data, file transfers) belong to the class of low priority traffic. In this thesis only the behavior of the high priority traffic is of interest, the low priority traffic serves only as background load.

A fully meshed topology is assumed, i.e. all stations can hear each other. Furthermore, the distance between the wireless terminals is assumed to be small (max. 30-50 m, according to machine plant applications), hence the propagation delay is small and can be neglected.

Load Models

Definition. *For a single priority a load value of x percent has the meaning that in the case of no errors and without packets of other priority present in the system the load offered via the link-layer interface is such that the time spent for transmitting* `data` *frames of the given priority and including overhead is x percent of the theoretical link bandwidth.*

Two different load models are defined, which are closely related to the *realtime performance measures*.

In both models and for both priorities, the data requests generated by the traffic sources have a size of 40 bytes user data. The load is varied by varying the interarrival times of the sources accordingly. There can be several sources attached to a single station.

Smooth Model The first model is denoted as the **smooth** model. It assumes some low priority background load of $x \in \{10, 50\}$ percent, which is splitted half into periodic arrivals and asynchronous arrivals (Poisson sources).

For the high priority load Poisson arrivals with an overall load value of 10% are assumed. However, the sources generate high priority requests for a station only when there is no pending high priority request. Stated differently, for the high priority requests a single station is modeled as an M/G/1/1 queueing system (with its service times involving vacations due to the polling algorithm or the token passing) [90].

This model allows investigating the *confirmation delay* for high priority requests defined below without taking queueing delays at the originating stations into account.

Batch Model The **batch** model is an approximation to the *alarm storm* phenomenon discussed in Section 2.1.

This phenomenon is approximated by a batch arrival, happening simultaneously at all stations. This definition is related to the associated measure of interest, namely the *consecutive confirm delay*, which measures the distribution of the times between successive confirmation primitives at a given station. This model is unrealistic by not using different times for batch arrivals at the stations. However, in the "more realistic" case the load conditions, as seen from a single station, vary over time as the batches at other stations appear. Hence, the obtained consecutive confirm delay values cannot be assumed to be from the same distribution.

So the model is as follows: given a constant low priority background load of 50% and no high priority load before time $t_0 = 10$ s, at time $t_0 = 10$ s at each station a batch of high priority requests and infinite size arrives.

As a combined scenario, for the high priority arrivals a two state Markov Modulated Poisson Process (MMPP) can be used, which is aimed to approximate best the "real world" behavior. However, with an instationary arrival process the delay measures can not be obtained in a meaningful fashion.

Realtime Performance Measures

The *realtime performance measures* are a set of measures targeted to capture the reliability regarding time and reliability. Due to the sometimes harsh and variable error conditions on certain types of wireless links, it is hopeless to give tight and deterministic guarantees on successful delivery of certain messages within a bounded time (see the results reported in Chapter 6, where it is shown that there are periods where for several seconds no packet is successfully received). Therefore, it is appropriate to express the requirements stochastically.

The main measures are delay-oriented: the *confirmation delay* $D_C(i)$ and the *consecutive confirmation delay* $D_{CC}(i)$, taken for a fixed wireless station i. Both are defined with respect to certain traffic scenarios: the $D_C(i)$ measure is evaluated within the **smooth** scenario, the $D_{CC}(i)$ measure within the **batch** scenario. Furthermore, both measures are taken only for high priority requests.

For a fixed station i the $D_C(i)$ measure denotes the time between the arrival of a high priority request at the link-layer interface of station i and the time instant when the corresponding confirmation primitive is generated, i.e., the transmission outcome is known. As from the **smooth** scenario, high priority requests arrive always to a system with an empty high priority queue, hence, this measure doesn't take any additional queueing delays into account.

The $D_{CC}(i)$ values for a fixed station i denotes the time between the generation of high priority confirmation primitives in the **batch** scenario.

Clearly, $D_C(i)$ and $D_{CC}(i)$ are taken as random variables, and it is assumed that (due to the traffic scenarios chosen) all the values belong to the same distribution.

Now the main optimization target for this work can be stated conveniently. Denote for given station i by $F_{D_C(i)}(x) = \Pr[D_C(i) \leq x]$ the distribution function of the confirmation delay values. The 99% percentile $x_{99}(i)$ of this distribution is given by:

$$x_{99}(i) = \inf\{x \in \mathbb{R} : F_{D_C(i)}(x) \geq 0.99\}$$

We want to minimize the quantity $\widetilde{D_C}$ (which we denote as *overall confirmation delay*):

$$\widetilde{D_C} := \max\{x_{99}(i) : i \in \{1, \ldots, N\}\}$$

i.e. we want to minimize the maximum of the 99% $D_C(i)$ percentiles over all stations. The $\widetilde{D_C}$ measure can be defined the same way for other percentiles, e.g. a 99.9% percentile.[6]

A similar quantity is of interest for the $D_{CC}(i)$ measures. This aggregate measure is denoted as $\widetilde{D_{CC}}$.

These measures now can be used to ask for a *realtime capacity.* Let us assume that from the application some bound on maximum transmission times of high priority messages is given, say D_{max}. As an example, consider a pressure sensor in a tank, which is configured to have a threshold value. If the pressure exceeds this threshold, an alarm is generated and sent to a controller station. The latter is then requested to do something against it as fast as possible. Then, for this given bound, the realtime capacity is defined as the maximum number N of wireless stations, such that $\widetilde{D_C} \leq D_{max}$ holds. From the definition of $\widetilde{D_C}$ it is clear that this value depends on N, although this is not explicitly shown in the notation.

In this definition the realtime capacity is computed for the $D_C(i)$ measures. A similar definition can be given with respect to the $D_{CC}(i)$ values, but this is not considered furthermore within this work.

As a side measure, it is also worthwhile to investigate the remaining bandwidth for low priority traffic B_L in the **smooth** scenario with 50% low priority load. Clearly, if two polling schemes show the same performance with respect to $\widetilde{D_C}$, the scheme which offers more bandwidth to low priority traffic is preferrable. The B_L measure is always given as a fraction of the overall available bandwidth, and incorporates data and overhead of low priority frames.

Some additional remarks are in order:

- It is assumed that the maximum number of retransmissions (*max_retry* parameter) is set to a high value ≥ 20, in order to increase reliability. Hence, the *negative confirmation rate*, where the MAC-/data link-layer has to report about finally unacknowledged requests, should be zero. If the *max_retry* parameter has a lower value (say, 3 to 5), then the negative confirmation rate would be of much interest.
- It is exactly the movement from a deterministic delay measure to 99% percentiles which accounts for the variability and error behavior of the wireless link, while simultaneously expressing hard delay requirements.

[6] The question, whether the results change for different percentile values is not covered in this thesis.

4.4 Wireless PROFIBUS over IEEE 802.11 MAC

For designing a wireless PROFIBUS considerable development effort could be saved, if existing technologies are used. This applies not only to wireless PHY's, but also reusing existing MAC protocols seems attractive.

More specifically, the approach discussed in this section is to take an existing MAC protocol providing a MAC interface, and to construct a mapping between the PROFIBUS link layer interface and the MAC interface. This mapping should implement the link-layer interface's syntax and semantics as closely as possible.

A natural candidate is the IEEE 802.11 WLAN standard with either DCF or PCF. Especially the PCF seems interesting with its time bounded services. However, this approach makes sense only if no changes or only minor changes to the existing protocols are required, in order to benefit from existing implementations. The feasibility of this approach is discussed.

4.4.1 DCF-based approaches

We consider first the DCF-based case. If no additional media access rules are implemented on top of the DCF (e.g., token passing on top of IEEE 802.11 MAC, as suggested in [130]), the following problems arise:

- By the decentralized protocol, it cannot be guaranteed that in the integrated scenario (see Section 4.3) the wireless medium is idle, when the token is logically in the wired segment and the current token-owner sends a request frame or the token to a wireless station.
- According to the SDL specification given in the IEEE 802.11 standard [120] the MAC entity maintains internal queues for its MAC-PDUs, and the MAC interface offers no way for upper layers to inspect or modify these queues. Only the current queue length can be inferred from the number of outstanding `MA-UNITDATA-STATUS`.indication primitives. Consider the case of an arriving high-priority request when the queues already contain a number of low-priority requests. There is no way to put the high-priority request in front of the queue, instead it is blocked by the low-priority requests.
- The CSMA-based protocol is subjected to the hidden terminal and exposed terminal scenarios.
- In general, a stochastic MAC protocol cannot give any guarantees on delays and losses.

Hence, additional distributed media access rules should be implemented. The focus here is on distributed rules, since otherwise we need a central controller, which leads to PCF-based scenarios, discussed below. An obvious candidate rule is explicit token passing [130], which, in the integrated scenario, could be integrated with the PROFIBUS token passing protocol (otherwise two different token passing protocols need to be maintained in parallel). However, in Chapter 5 it is shown that explicit token-passing over error-prone wireless links has serious problems and should not be considered for a wireless PROFIBUS.

Other additional media access rules (e.g., virtual token passing as in P-Net, see Section 2.3) must be made compatible with the properties of the PROFIBUS token passing.

Even if one of these approaches could be implemented successfully, it is questionable whether the result will be more efficient or less complex than a specifically tailored wireless PROFIBUS MAC protocol. The 802.11 DCF is of considerable complexity, and additional rules increase the complexity of the resulting MAC protocol.

There are other DCF-based approaches, which, however, extend the DCF such that existing equipment could not be used. For example, the authors of [156] propose a DCF extension for adding station priorities. Their approach uses a special jam signal which is not part of the standard and thus requires special capabilities of the wireless transceiver, which makes this approach unattractive.

4.4.2 PCF-based approaches

In this section we make the assumption that the 802.11-PCF AP serves also as the boundary between the wired and wireless parts, where the PROFIBUS frames are translated between the different media.

A critical situation for PCF-based approaches is when the token is logically in the wired segment (*wired token situation*). To avoid collisions when a wired master sends a frame to a wireless slave, the wireless medium should be kept idle.

Consider first the case, that the AP works in conformance to the IEEE 802.11 standard, i.e., it implements the concept of superframes subdivided into CFP and minimum length CPs. If the wired token situation happens within the CFP, the AP can keep the wireless medium idle by continuosly transmitting short dummy frames to a non-existing address.[7] These frames, however, need a PLCP preamble and header of at least 192 μs duration, not counting the MAC-PDU data. When a request frame arrives during the PLCP preamble, it may be possible to replace the dummy frame by the request frame, if the request frame arrives early enough, say, up to the τ-th bit of the preamble ($\tau < 128$). The value of τ is determined by the MAC processing latency. If the request frame arrives just after the τ-th bit, the remaining bits of the dummy frame induce additional forwarding delay.

If the wired token situation happens within the CP, it is hardly possible to keep the wireless medium idle without additional medium access rules or without the AP occupying the wireless medium. Furthermore, due to the variable PROFIBUS token cycle times it is hard to guarantee that always the wired token situation and CFP occur jointly.

The problems with the CP can be attacked by getting rid of the superframe structure and by letting the AP controlling the medium all the time. However, this is close to the polling-based approach adopted in this thesis, which does not need all the complexity introduced with the 802.11 MAC protocol, but runs directly on top of an 802.11 DSSS PHY.

4.4.3 SRD service handling

It is difficult to implement the (C)SRD service with the DCF or PCF of IEEE 802.11. The basic problem is that the immediate acknowledgement frames cannot carry any data, as is requested by the SRD service.

First we consider the DCF case.[8] If the token is logically in the wired segment and a wired master

[7] Another approach is to equip the AP with modified wireless transceivers, allowing to send jam signals of arbitrary length. However, this cannot be done with off-the-shelf components and is not considered furthermore.

[8] We assume the RTS/CTS protocol being disabled, since PROFIBUS frames are typically small.

sends a SRD frame to a wireless station, the following approaches are possible:

- The SRD frame could be directed to a unicast 802.11 MAC address corresponding to the destination PROFIBUS MAC address. The immediate ack sent by the wireless station can be suppressed by the AP, i.e., it does not appear on the wired part. Immediately after receiving the request frame the wireless slave prepares a response frame and hands it over to its MAC entity (which should have no other frames in its internal queues). After some delay (at least DIFS), the answer frame is transmitted and forwarded by the AP to the wired master. This approach has the problem of substantial delay before the first bit of the answer frame appears on the wired medium: SIFS plus ack frame duration plus the time needed for preparing the answer plus DIFS plus PLCP preamble and header. If the wireless station's internal MAC queues are not empty, this approach does not work.
- The SRD request frame could be directed to the 802.11 broadcast address, as well as the answer data. This way, two immediate acks are eliminated. This approach imposes additional burdens to all wireless stations, since, by increasing the number of broadcast frames, it increases the processing load. Again, it works only if there are no other frames in the wireless station's internal MAC queues.

When using the PCF, two cases can be distinguished: if the SRD frame exchange is performed during the CP, the same considerations as for the DCF case apply. When performed during the CFP, the following cases occur:

- When the SRD request frame is sent by a wired station, the AP can piggyback a poll-request to the forwarded frame. If the wireless slave has given the answer data already to the MAC instance (with the `MA-UNITDATA`.request service primitive), it can answer with the stored answer data. Unfortunately a piece of data given to the MAC cannot be replaced by some updated data, instead both pieces of data are transmitted. Hence, with this mechanism the PROFIBUS SRD service cannot be fully implemented, since this service has the concept of an FDL buffer, which can be modified several times and only the last version is put into an ack frame. If the answer data is not readily available, the wireless slave must be polled some time later, introducing delay.
- When the SRD request is sent by a wireless station to another, unicast frames have to be used and costly immediate acks cannot be suppressed. Furthermore, the AP has to poll the destination station immediately after the ack, in order to allow for transmission of answer data.

In summary, the IEEE 802.11 MAC is not well suited to implement the SRD services.

4.4.4 Final Remarks

Using a full 802.11 protocol stack with its MAC and management functionalities, brings significant complexity to wireless devices. Especially for small and cheap sensors, which only deliver some measurement values from time to time, 802.11 seems to be oversized in terms of cost.

The approach to implement a mapping of the link layer interface to the 802.11 MAC interface introduces several problems and unneeded complexity. The IEEE 802.11 MAC protocol is not designed for the services needed and constraints given in a wireless PROFIBUS LAN. Many functions are not needed for wireless PROFIBUS (e.g., always sending immediate acks), others are missing. Additional functionality must be added to achieve reasonable behavior. In this case, however, approach looses its appeal of using off-the-shelf components.

A specifically tailored MAC protocol is the solution adopted for this thesis.

4.5 Related Work

4.5.1 Overview of other Work in Wireless Fieldbus Systems

There exists much literature on "wireless realtime MAC" related topics, however, in almost all cases the focus of interest is on time-sensitive and somewhat loss-tolerant data types like voice and video. For IEEE 802.11 the related literature is reviewed in Section 4.5.2. Much literature exists in the context of wireless ATM systems (some references are [6, 93, 25]) or for integrating voice and data. In contrast, the topic of hard real-time communications or fieldbus systems over wireless media is only sparsely covered.

Wireless PROFIBUS

The Funbus project [52], [81] was an industry-driven project with the goal of finding a cheap and reliable technology for wireless and transparent coupling of field devices. Three different fieldbus technologies (PROFIBUS-DP, INTERBUS-S [36], [37], [38], and CAN [50]) and several wireless technologies (e.g., GSM, DECT, 802.11, TETRA) were investigated, but finally the participants have chosen 802.11 DSSS related technologies working in the 2.4 GHz ISM band.

The project focused on the wireless repeater scenario. Some alternative approaches for this have been investigated: a) the forwarding could be done over the 802.11 DCF MAC with either unicast addressing (thus every wireless station or bridge has to send an ack) or broadcast addressing; or b) forwarding is done by encapsulating the PROFIBUS frames in 802.11 PHY frames without using the MAC and sending them in broadcast mode. For performance reasons, the latter approach was adopted. The forwarding is done by bridge-like devices in a cut-through manner. Some bridge prototypes were built, based on the Silver Data Stream radio modem [157]. This setup was evaluated with laboratory measurements and field trials. The minimum forwarding delay introduced by the wireless link and bridges was measured to be 200μs. A simple laboratory setup was evaluated, consisting of one PROFIBUS DP master linked to two PROFIBUS DP slaves. The connection between master and slaves was either fully wired or contained a wireless link. The other parameters were: bus speed on the wired part of 500 kBit/s, line of sight (LOS) connection for the wireless link, no antenna diversity and 1 MBit/s BPSK modulation on the wireless link. For the wireless case a throughput reduction of 40% was observed in terms of handled requests/s [52, p.72].[9] In the field trial the same setup was placed in a gypsum warehouse: the DP master has a fixed position at the warehouses walls, while the DP slaves were put on a moving gypsum conveyor system (LOS connection). On the cabled segment a transmission speed of 500 kBit/s was used, on the wireless segment 1 MBit/s. The conveyor system was actively moving during the 9h measurement. The data throughput was stable most of the

[9]The numbers given in the report (13800 requests/s in the wired case, 8500 requests/s in the wireless case) are too high for a 500 kBit/s wired bit rate, but we presume that the relation is correct, i.e., in the wireless case the efficiency is reduced to $\approx$ 60% of the wired case.

time, but frequently outliers occur. For an hour no communication was possible, the authors propose multipath effects as a possible explanation.

Meanwhile, some companies offer products for the wireless repeater scenario, using infrared communications. These products allow to couple wireless stations into a wired PROFIBUS or to link different wired PROFIBUS segments (two exemplarily references are [84], [66], both employing infrared waves).

Other Wireless Fieldbus Systems

Within the Funbus project [52] also the INTERBUS-S and CAN were investigated. For the INTERBUS-S a repeater/bridge solution based on the 802.11 DSSS PHY was used. This was feasible, since the INTERBUS MAC protocol operates on point-to-point links with no further access mechanism. For the CAN protocol an application layer gateway was constructed, since the approach of emulating the CAN MAC protocol on the wireless link was found overly difficult. For performance reasons a specific MAC protocol was implemented on top of an IEEE 802.11 PHY.

The R-FIELDBUS project (*www.rfieldbus.de*) within the European Union Information Society Technologies (IST) program evaluates the use of different radio technologies (UMTS, IEEE 802.11, HIPERLAN, DECT, Bluetooth) for the fieldbus systems specified in the EN50170 european standard (including PROFIBUS, P-NET and WorldFIP) with focus on multimedia support [141]. At the time of writing no technical details were available.

A group at EPFL Lausanne has worked on transparent integration of wireless stations into FIP [109]. The factory instrumentation protocol (FIP) fieldbus is a european fieldbus standard, which emerged from a french standard [178]. It uses a polling table to implement a realtime database (a more detailed description can be found in reference [30]). A main element of the approach in [109] is a wireless-to-wired gateway, which serves as central base station for the wireless part. The MAC protocol is based on a time division multiple access (TDMA) scheme. The base station is responsible for caching all process variables produced by mobile stations and to transmit these on the wired part, if requested. Furthermore the gateway caches all process variables produced by wired stations and consumed by wireless stations, and broadcasts these on the wireless link. The case of asynchronous directed message transmission is not discussed. In [110] it is investigated, how the MAP/MMS application layer protocol [80] can be enhanced with mobility. In the proposed system the IEEE 802.11 MAC protocol with the (stochastic) DCF is used, time critical transmissions are not considered. In [108] the same question was investigated with DECT as underlying technology. Again, time critical transmissions were not considered.

The european community project OLCHFA (June 92 until September 94) was targeted at enhancing FIP with wireless stations at the 2.4 GHz ISM band using a DSSS physical layer [74]. However, the available publications put emphasis on the management of configuration data and on distributed algorithms for clock synchronization. Within the project a communications controller was developed, which can switch between using a wired and a wireless medium. The MAC and data link protocol of FIP was not modified.

For the IEC FieldBus [69] (which uses a centralized, polling-based access protocol for supporting periodic data and a token passing protocol for asynchronous data) in [22] an architecture was proposed, which allows coupling of several fieldbus segments using a wireless backbone based on IEEE 802.11 with PCF.

A group at the university of Sussex has worked on the topic of wireless CAN. Since the CAN MAC protocol is not implementable on a wireless medium, two different approaches were developed: the WMAC approach uses backoff times directly proportional to the message priority before starting to transmit [96], in the second approach the priority value is mapped onto the channel using an on-off-keying scheme: a station transmits a short burst, if the current priority bit is a logical one, or it switches to receive mode if it is a zero. If the station receives something in the receive mode, it resumes from contention. The priority bits are handled from the most significant bit to the least significant bit, all stations have to be synchronized on bit boundaries [95]. The papers cited do not take channel errors and retransmissions into account. Furthermore, this approach requires fast switching between transmit and receive mode for the radio modem.

4.5.2 Real-Time Data Transmission with IEEE 802.11

In this section existing work on real-time transmission with IEEE 802.11 is summarized. In almost all publications the notion of "real-time transmission" is not to be understood in the industrial "hard real-time" sense, where packet losses should not happen. Instead, it is used in the context of multimedia transmission, e.g., for speech, audio and video data. While these data types also have stringent timing requirements, they can tolerate some losses. For voice transmissions the delay is desired to be ≤ 250 msec, but a certain packet loss rate (typically $\approx 1\%$) seems tolerable, depending on the codec and the influence of error concealment techniques [65, chap. 7].

The literature in this field can be broadly subdivided into approaches for using and enhancing the DCF and those using the PCF, see the next two sections. A comparison of several schemes for transmitting speech data over an IEEE 802.11 WLAN is presented in [131].

Several studies, however, deal with the performance of 802.11 without explicitly referring to real-time data, e.g., [185]. Especially the backoff algorithm has been the focus of several studies, e.g., [15] and [186]. In reference [20] capacity limits for IEEE 802.11 DCF were derived using analytical estimates for the protocol overhead. They showed that for certain scenarios the theoretical capacity is not reached and proposed a modified backoff algorithm. This algorithm takes the current load into account.

In most of these papers transmission errors were not taken into account.

DCF-based Approaches

In [31] an enhancement of the 802.11 DCF for prioritizing frames is proposed, targeted to enhance the real-time capabilities for voice and video. This approach distinguishes two types of data: time-bounded data and asynchronous data. Asynchronous data may be delayed arbitrarily, but should definitely reach the destination. On the other hand, time-bounded data looses its meaning after deadline expiration. With the help of new inter frame spaces and shorter backoff intervals a prioritization of stations is introduced. When prioritizing video frames over data frames using this scheme, an improved transmission delay and reduced loss probabilities for video frames can be observed.

Recently, QoS is becoming an issue also in the IP world. One of the most prominent approaches is the *Differentiated Services* approach [86]. In this approach packets are classified into different flows and each flow is treated by different rules in the routers and edge nodes. One prerequisite is the ability of the underlying networks to provide different service levels. Especially in those technologies, where bandwidth is a scarce resource, this is often achieved with introducing packet priorities and proper packet scheduling approaches. In [12] two approaches are combined for introducing two service classes: time-bounded and asynchronous data. The first approach uses two different CW_{min} values for time-bounded data (lower value, see Section 3.2.3) and asynchronous data (higher value). It is shown with simulations that time-bounded data gets faster channel access (in the mean) and that throughput of time-bounded data is largely insensitive to the load of asynchronous data. By additionally selecting a lower CW_{max} value for time-bounded data packets, these are discarded earlier in high congestion conditions. The second approach employs measurement based admission control. The current network load is estimated by measuring the delays, a virtual MAC instance would experience under current network conditions.

A second approach for modifying the backoff procedure to achieve service differentiation is presented in [1]. However, instead of modifying the CW_{min} and CW_{max} bounds, the growth exponent of the CW variable is set to different values than two. Each flow is assigned its own exponent. It is shown that this approach can provide throughput differentiation for different CBR flows, if the growth factor ratios are not too large. However, for TCP flows no significant service differentiation can be observed.

The authors of [156] propose a DCF extension for adding station priorities. Their approach uses a special jam signal which is not part of the standard and thus requires special capabilities of the wireless transceiver.

PCF-based Approaches

In [181] a scenario is investigated, where the PCF is used for voice transmission, while data is transmitted with the DCF. Hence, superframe stretching occurs. For the voice sources a simple ON/OFF model [19] is used (modeling a voice coder with silence detection). The model is based on a two state markov chain, i.e., the state holding times are exponentially distributed (mean time in talk state: 1 s, mean time in silent state: 1.35 s). During the talk state speech frames are generated with 8 kBit/s. Additionally, unsent voice frames are discarded, if their lifetime exceeds a threshold of 25 ms. Data frames have exponentially distributed interarrival times and frame sizes. The load in the CP part of the superframe, as given by subtracting the nominal length of the CFP from the superframe period, is set to 98%. There are 15 stations, all stations send data frames, while a variable number of stations generates additional voice frames. The voice stations are polled during the CFP in round robin fashion. Their main result is that there is a large overhead necessary for voice transmission, and that superframe stretching further limits the voice capacity. For example, with a superframe period of 20 ms and the CFP length set to support 8 voice connections, only 50% of the bandwidth is available for DCF transmission. Exploring the potential statistical multiplexing gain for ON/OFF sources increases the number of conversations up to 15 with 3.5% drop rate (life time exceeds threshold). In this case the maximum throughput for data frames is limited to 200 kBit/s (channel rate: 1 MBit/s). Channel errors were not taken into account.

The work described in [92] investigates a similar scenario, i.e., a mixture of voice and data sources. It provides a comparison of DCF and PCF mode for speech transmission. In the DCF case a local scheduler is used for providing priority to speech frames, in the PCF case the CFP is used for speech frames. Similar to reference [181] a substantial overhead (up to 50%) is found, but nonetheless the PCF allows to support more voice streams than the DCF variant. A major source of overhead are empty polls during the CFP, i.e., stations have no speech packets available when polled by the BS. When the PC has perfect knowledge about all stations queue status, a significant increase (600 kBit/s) of DCF capacity as compared to the case with imperfect knowledge can be achieved (11 MBit/s, 64 kBit/s speech rate). However, errors are not taken into account.

The authors of [180] also consider voice transmission using the PCF and propose suitable settings for different protocol parameters. The maximum number of admissible CBR voice calls depends on the superframe length, e.g., for a 90 ms period with 11 MBit/s DSSS up to 26 voice calls can be carried, with 2 MBit/s FHSS up to 11 calls can be carried, both in case of no transmission errors. They propose to use these numbers to perform admission control for voice calls.

Some further studies about voice and video transmission with the IEEE 802.11 PCF can be found in [27] and [159]. A protocol extension, which addresses a group of stations with a single poll-frame, thereby reducing overhead, is described in [53].

Chapter 5

Behavior of the PROFIBUS Protocol under Link Errors

This chapter provides an answer to the question, why it is not a good solution to run the PROFIBUS MAC and link layer protocol directly on top of a wireless PHY. The main problem is the need for explicit token passing and the consequences that token losses can have for the stability of the logical ring and the active stations opportunities to transmit their data.

The investigation proceeds in two steps: in the first step we show that the PROFIBUS protocol is not designed for coping with higher error rates or packet losses (Section 5.1). To do this, the protocol is investigated in its "natural environment" with an underlying RS 485 PHY and certain error assumptions.

In the second step the PROFIBUS protocol is investigated with the characteristics of an 802.11 PHY (Section 5.2). The main differences are the error assumptions and the fact that the hearback feature is not available.

Finally, in Section 5.3 the related literature is reviewed, and in Section 5.4 the results of this chapter are summarized.

The material presented in Section 5.1 is also published in [188], [190], [189], and [198].

5.1 PROFIBUS over Error Prone Links

The PROFIBUS is designed to deliver real-time services in harsh, industrial environments, as is the IEEE 802.4 token bus protocol. In both protocols a logical ring is built on top of a broadcast medium, using special control (token) frames for ring maintenance, however, the maintenance mechanisms differ: IEEE 802.4 uses a contention-based mechanism for including active stations (stations for short) into the ring, while PROFIBUS uses explicit polling. In both protocols only members of the logical ring are allowed to transmit data. Thus, one important goal of the PROFIBUS protocol is that all stations, who wants to be, are member of the ring and remain so. The degree to which this is achieved is referred to as *ring stability*, and can be captured with different metrics. Since the ring membership is maintained by exchanging special control frames, the ring stability can be affected by loss of these frames. Since data transmission is restricted to ring members it is clear that ring stability strongly impacts the achievable QoS and system reliability.

In this section we study the ring stability of the PROFIBUS protocol operated over the RS 485 PHY in the presence of transmission errors and under different error models. It is shown that the protocol has serious stability problems under higher error rates and that ring stability is sensitive to the "burstiness" of errors. We propose two improvements of the protocol and its parameters, which require no modifications in frame formats and are interoperable with the unchanged protocol rules. These improvements yield a significant increase in ring stability, but give still not satisfactorily results, as is shown in Section 5.2 for the case of a wireless link and also in Chapter 7 when compared with polling-based protocols.

We introduce two definitions: a *station loss event* (or simply *station loss*) denotes the single point in time where an active station detects its loss from the ring and discards all of its knowledge previously obtained by its ring maintenance mechanism, especially the LAS. After a station loss a station behaves as newly switched on.

A *station outage time* denotes the time duration needed for a lost station to become a ring member again (by expiration of its timeout timer or by being reincluded by another station).

5.1.1 Major Causes for Ring Instability

By analysis of the protocol specification and of simulator traces (see Section 5.1.4), we have identified three different ways how a station can get lost.

The first way is due to the fact that the token frame has no checksum. It is only protected with a parity bit, startbit and stopbit for every single byte (every byte is transmitted serially with 11 bits, see Section 4.1.4). Thus there is some probability that a token frame can be corrupted such that no station except the sender (by the hearback feature) will recognize an error.[1] Consider now the case of two stations with addresses a and b respectively, where $a < b$ holds to ease presentation. If a sends a token frame to b where the destination address is corrupted and equal to c with $a < b < c$, b considers itself being skipped and immediately removes itself from the ring, behaving as newly switched on. If a retransmits the token, b has not yet built a valid LAS and does not accept the token. After another token frame a considers b as lost from the ring, since again b is not allowed to answer. We refer to this as *error skipping*.

The other scenarios are due to the presence of the hearback feature: when station a experiences hearback errors in two successive trials to send a token frame it gets lost from the ring (i.e. forgets its LAS). When the token frames are detected as faulty by all other stations, then the medium is idle until the timeout timer of the station with the lowest address expires. Within this scenario two cases can be distinguished: a has the lowest station address of all current ring members or not (assume that a has negligible initialization delay). If a has the lowest address, it is the timeout timer of a that expires. Since there was no transmission during the idle time and a has forgotten its LAS, a now

[1] This probability can be lower bounded by the probability P_R, that exactly two bit errors occur within the same byte, which cannot be detected by the parity scheme. The token frame is $3 \cdot 11 = 33$ bits long. Assuming that bit errors are independent (at least over the length of a token frame) and occur with fixed probability p, P_R is then given by $P_R = \frac{168}{1056} \cdot b(2; 33, p)$ where $b(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$ is the distribution function of the binomial distribution. We have used the fact that from 1056 ways to distribute two errors over 33 bits only 168 of these lead to undetectable errors, all others are detected. With $p = 0.001$ we have $P_R \approx 0.00008$.

thinks it is alone in the ring and sends a token frame to itself. Then all other stations remove from the ring, feeling themselves skipped. We refer to this scenario as *ring jacking*. If a has not the lowest address, the remaining ring keeps alive and a is reincluded later. This is denoted as *hearback removal*.

To summarize, the mechanisms for loosing stations are as follows:

- Station a gets lost due to error skipping.
- Station a experiences a hearback removal.
- Station a gets lost because another station b with the lowest address performs ring jacking.

5.1.2 Ring Stability Metrics

The metrics for ring stability can be roughly divided into two classes: the *global stability metrics* are focused on the whole logical ring, while the *local stability metrics* look at a single station.

Let K be the number of stations and $\{N(t)\}_{t \in \mathbb{R}}$ a set of integer-valued random variables, denoting the number of stations that are members of the ring at time t (more precisely: which consider themselves being member). We have $0 \leq N(t) \leq K (t \in \mathbb{R})$, and $N(t)$ changes only at discrete points in time, by the operation of the protocol. All stations want to be member of the ring all the time. We introduce the following global metrics for ring stability:

- Consider at time t_0 we have $N(t_0) = K$ and $\lim_{\epsilon \to 0, \epsilon > 0} N(t_0 - \epsilon) < K$, i.e. the ring has just been completed at t_0. Furthermore let $t_1 = \inf\{t > t_0 : N(t) < K\}$ and $C = t_1 - t_0$. The random variable C denotes the time duration that the ring is complete, before the next time it looses a station. We are interested in its mean value $\bar{C}$ and distribution function $C(s) = \Pr[C \leq s]$. The "dual" of C, i.e. the time needed to re-enter the state of a full ring after the full ring breaks, is not covered here.
- Mean number of stations in the ring during interval $[0, t]$:

$$\bar{N}(t) = \frac{1}{t} \int_0^t N(s) ds,$$

  additionally we are interested in the limiting mean value $\bar{N} = \lim_{t \to \infty} \bar{N}(t)$, which is assumed to exist and approximated by evaluating $\bar{N}(t)$ for some large t.
- Fraction of time where not all stations are member of the ring during time interval $[0, t]$:

$$\bar{M}(t) = \frac{1}{t} \int_0^t \mathbf{1}_{[0, K-1]}(N(s)) ds$$

  where $\mathbf{1}_A(x)$ is the indicator function for the set A, i.e. $\mathbf{1}_A(x) = 1$ if $x \in A$, and $\mathbf{1}_A(x) = 0$ otherwise. Additionally we are interested in the limiting fraction $\bar{M} = \lim_{t \to \infty} \bar{M}(t)$.

Some important local metrics for a single station i are the following: the distribution of times between station loss events, the duration of station outages and the overall fraction of time that i is not member of the ring. Some simulation results for these metrics can be found in [189].

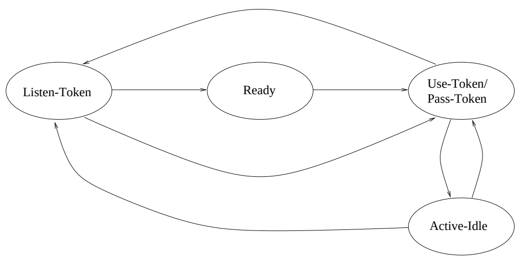

Figure 5.1: A single stations life cycle

5.1.3 Analytical PROFIBUS Ring Membership Model

In the following we describe an analytical model for the behavior of a set of PROFIBUS stations w.r.t. ring membership. This model allows to determine $\bar{M}$ and $\bar{N}$. This model depends on some model parameters, for which in turn approximations are derived using only the protocol description and some static system parameters.

The purpose of this model is twofold: first it is interesting in itself. Second, by successfully comparing the qualitative and quantitative behavior of the analytical models outputs with those of the simulation model described in Section 5.1.4, the confidence in proper working of the simulation model can be increased.

The Model

The approach is to derive a discrete time markov chain (DTMC) from a simplified station life cycle, employing four states (the PROFIBUS protocol state machine [33, chap. 4.1] has eleven states). The initial DTMC description has a four-dimensional state description. However, in order to solve for a steady-state probability vector, the four-dimensional state space is flattened into a one-dimensional state space.

The stations life cycle is shown in figure 5.1 as a set of states, which the station visits in some order within its life. The state *Listen-Token* corresponds to the state, where a station listens to the medium, waiting for two successive identical token cycles and not being able to enter the ring (with the exception of timeout timer expiration). In state *Ready* a station has successfully received two successive identical token cycles and thus has a valid LAS, however, it waits for being included in the ring. In state *Use-Token/Pass-Token* the station is member of the ring and it currently owns the token. It either performs some data transmission, pings a station in its gap list or tries to pass the token to its successor. Finally, when a station is in state *Active-Idle*, it is a ring member, but does not currently own the token. All possible transitions can be explained by normal protocol operation, only the transition from Active-Idle to Listen-Token needs the “hearback removal” condition explained in Section 5.1.1. In order to obtain a Markovian model, the following assumptions hold:

- Time is slotted, slots have fixed length T_{Slot} (different from the protocols slot time T_{SL}). This assumption is reasonable, if there is no load in the system and only token frames and request frames for ring inclusion are exchanged. One slot corresponds to one token frame.
- The system is assumed to be time-homogeneous, and already running for a long time, hence the

system is in the steady-state.

- All feasible state transitions occur for every single station with fixed probability p_{XY}, where X and Y denote shortly the source and target state (e.g., p_{LU} being the probability for transition from Listen-Token to Use-Token), exceptions see below. These fixed probabilities depend on the current number of ring members, and on environmental conditions, e.g., the error rate. The stations are independent of each other.
- The protocol allows for two different transitions from state Ready to state Use-Token: in the first there is no token owner and a station in state Ready experiences a timeout, in the second a station in Ready state is explicitly included by the current token owner. For the first type of transitions we assume that every station performs an independent Bernoulli experiment in every slot with fixed probability p_{RU}. Upon success one station enters the Use-Token state. For the second transition type only a single station can be included, this happens with the state-dependent probability $p_I(i,j)$, where i denotes the number of current ring members (given by $A+U$) and j denotes the number of stations in state Ready.
- The probability p_{LR} for transition from the state Listen-Token to the state Ready depends on i, where i is the number of current ring members.
- Bit errors occur independently with fixed rate p.
- Station addresses are uniformly distributed in $[0, 126]$
- If token frames are transmitted correctly, either a transition from Listen-Token to Ready or a transition from Ready to Use-Token is possible (only one of them at a time); if the token is erroneous, there can be a hearback error, causing a station to leave the ring, and additionally it can happen that other stations are skipped (these events do not exclude each other).
- For experiencing a hearback error, we assume that a single erroneous token suffices, however this occurs with a probability corresponding to two successive erroneous token frames. Furthermore it is assumed that the event of an active station being "skipped" by token frames with unfortunate error patterns is tied to the event of experiencing a hearback error. This assumption is reasonable, since this event is relatively rare.

A DTMC with a four-dimensional state space $\mathcal{S}(K)$ is constructed:

$$\mathcal{S}(K) = \{(L,R,U,A) \in \mathbb{N}_0^4 | (L+R+U+A=K) \wedge (U \leq 1)\}$$

where for $s = (L,R,U,A) \in \mathcal{S}(K)$ we define

- $L = L(n) = \#$ Stns in Listen-Token at time slot n
- $R = R(n) = \#$ Stns in Ready at time slot n
- $U = U(n) = \#$ Stns in Use-Token at time slot n
- $A = A(n) = \#$ Stns in Active-Idle at time slot n

for every slot number $n \in \mathbb{N}$. However, the slot number n is dropped in the notation. It is easy to see that $|\mathcal{S}(K)| = K^2 + 2K + 1$ holds.

A state $x_n = (L, R, U, A) \in \mathcal{S}(K)$ denotes the state of the system at the n-th time slot. In the following we enumerate for every state the possible transitions and give their respective probabilities. When not explicitly mentioned, the probability for staying within the same state is implicitly defined by $(1 - \sum(\text{all other trans. prob.}))$. A specific system usually starts with $x_0 = (K, 0, 0, 0)$.

- $L = K, R = 0, U = 0, A = 0$:
  - Exactly one station experiences a timeout:

    $$\Pr[x_{n+1} = (K-1, 0, 1, 0) | x_n = (K, 0, 0, 0)] = b(1; K, p_{LU})$$

    where $b(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$ is the distribution function of the binomial distribution.
- $L = K-1, R = 0, U = 1, A = 0$:
  - The token owner experiences a hearback error:

    $$\Pr[x_{n+1} = (K, 0, 0, 0) | x_n = (K-1, 0, 1, 0)] = p_{UL}$$

  - A number of stations enters the Ready state:

    $$\Pr[x_{n+1} = (K-1-k, k, 1, 0) | x_n = (K-1, 0, 1, 0)] = (1 - p_{UL}) \cdot b(k; K-1, p_{LR}(1))$$

    where $k \in \{0, \ldots, K-1\}$ holds.
- $L = i, R = j, U = 1, A = 0$ with $i + j + 1 = K$ and $j \geq 1$:
  - The token owner experiences a hearback error:

    $$\Pr[x_{n+1} = (i+1, j, 0, 0) | x_n = (i, j, 1, 0)] = p_{UL}$$

  - A number of stations enters the ready state ($i \geq 1$):

    $$\Pr[x_{n+1} = (i-k, j+k, 1, 0) | x_n = (i, j, 1, 0)] = (1 - p_{UL}) \cdot (1 - p_I(1, j)) \cdot b(k; i, p_{LR}(1))$$

    where $k \in \{0, \ldots, i\}$ holds.
  - A single ready station enters the ring and becomes token owner:

    $$\Pr[x_{n+1} = (i, j-1, 1, 1) | x_n = (i, j, 1, 0)] = (1 - p_{UL}) \cdot p_I(1, j)$$

- $L = i, R = j, U = 0, A = 0$ with $i + j = K$ and $j \geq 1$:
  - A single station in ready state experiences a timeout (no transitions from Listen-Token to Ready can happen):

    $$\Pr[x_{n+1} = (i, j-1, 1, 0) | x_n = (i, j, 0, 0)] = b(1; j, p_{RU})$$

  - A single station in Listen-Token experiences a timeout ($i \geq 1$):

    $$\Pr[x_{n+1} = (i-1, j, 1, 0) | x_n = (i, j, 0, 0)] = b(1; i, p_{LU})$$

- $L = i, R = j, U = 1, A = k$ with $i + j + k + 1 = K$ and $k \geq 1$:

  - The token owner experiences a hearback error and by that way a number of stations in state active idle feel themselves skipped:

$$\Pr[x_{n+1} = (i+1+\nu, j, 0, k-\nu) | x_n = (i, j, 1, k)] = p_{UL} \cdot b(\nu; k, p_{AL})$$

    where $\nu \in \{0, \dots, k\}$ holds.

  - A number of stations enters the ready state ($i \geq 1$):

$$\begin{aligned} &\Pr[x_{n+1} = (i-\nu, j+\nu, 1, k) | x_n = (i, j, 1, k)] \\ &\quad = \begin{cases} (1-p_{UL}) \cdot (1-p_I(k+1, j)) \cdot b(\nu; i, p_{LR}(k+1)) & : \quad j > 0 \\ (1-p_{UL}) \cdot b(\nu; i, p_{LR}(k+1)) & : \quad j = 0 \end{cases} \end{aligned}$$

    where $\nu \in \{0, \dots, i\}$ holds.

  - A single ready station enters the ring ($j \geq 1$):

$$\Pr[x_{n+1} = (i, j-1, 1, k+1) | x_n = (i, j, 1, k)] = (1-p_{UL}) \cdot p_I(k+1, j)$$

- $L = i, R = j, U = 0, A = k$ with $i + j + k = K$ and $k \geq 1$:

  - A station in Listen-Token experiences a timeout and by that way kicks all stations in Active-Idle out of the ring ($i \geq 1$):

$$\Pr[x_{n+1} = (i-1+k, j, 1, 0) | x_n = (i, j, 0, k)] = b(1; i, p_{LU})$$

  - A station in Ready experiences a timeout ($j \geq 1$):

$$\Pr[x_{n+1} = (i, j-1, 1, k) | x_n = (i, j, 0, k)] = b(1; j, p_{RU})$$

  - A station in Active-Idle experiences a timeout:

$$\Pr[x_{n+1} = (i, j, 1, k-1) | x_n = (i, j, 0, k)] = b(1; k, p_{AU})$$

This model has a four-dimensional state space. However, common steady-state solution techniques require Markov chains to be one-dimensional. Hence, a bijective mapping h from the state space $\mathcal{S}(K)$ to a one-dimensional state space of exactly the size $|\mathcal{S}(K)| = K^2 + 2K + 1$ is needed. This mapping can be explicitly constructed by simply enumerating the whole space $\mathcal{S}(K)$ and assign every state a distinct natural number. If $\mathbf{P}$ denotes the time-homogeneous state transition matrix for the one-dimensional markov chain, then we can determine the steady-state probability vector $\pi = (\pi_1, \dots, \pi_{|\mathcal{S}(K)|})^T$ as usual by solving the following system of linear equations:

$$\begin{aligned} \pi^T &= \pi^T \cdot \mathbf{P} \\ \sum_{i=1}^{|\mathcal{S}(K)|} \pi_i &= 1 \end{aligned}$$

where $\mathbf{A}^T$ denotes the transposed matrix of matrix $\mathbf{A}$. After determining π we can translate this back to a four-dimensional steady-state vector $\pi_{\mathcal{S}(K)} = (\pi_{(K,0,0,0)}, \dots, \pi_{(0,0,1,K-1)})$ using the inverse mapping h^{-1}. The steady-state vector exists for all investigated parameter values.

Evaluation of Stability Measures

After obtaining the steady-state vector $\pi_{\mathcal{S}(K)}$ the $\bar{M}$ and $\bar{N}$ measures can be evaluated as follows:

$$\begin{aligned}\bar{M} &= 1 - \sum_{(L,R,U,A)\in\mathcal{S}(K),U+A=K} \pi_{(L,R,U,A)} \\ \bar{N} &= \sum_{(L,R,U,A)\in\mathcal{S}(K)} \pi_{(L,R,U,A)} \cdot (U+A)\end{aligned}$$

Estimating Model Parameters

The next issue to resolve is the estimation of the model parameters $p_I(i,j)$, p_{LU}, p_{RU}, p_{AL}, p_{AU}, p_{UL} and $p_{LR}(i)$ from some selected system parameters, namely:

- the (constant) bit error rate $p \in (0,1)$,
- transmission rate b (bits/sec)
- gap factor g,
- target token rotation time T_{TTRT}
- number of stations K
- protocol slot time T_{SL}

under the assumption of no load in the system.

The probability p_{UL} corresponds to the case, where a single station experiences two successive hearback errors in its token frames. Since a token frame consists of three bytes, each one transmitted with 11 bits, the probability for a single token frame of being correct is given by

$$p_{TokenCorrect} = (1-p)^{33},$$

hence, the probability of two successive token frames being wrong under independent errors is given by

$$p_{UL} = (1 - p_{TokenCorrect})^2 = (1-(1-p)^{33})^2$$

Correspondingly, for the Request-FDL-Status frame (six bytes) and its corresponding answer frame (six bytes) the probability that both frames are successfully transmitted is given by

$$p_{Req-FDL-Succ} = ((1-p)^{66})^2.$$

The model slot time T_{Slot} is for simplicity fixed to 50 bit times (approximately the center value between the 33 bits long token frame and the 66 bits long Request-FDL-Status frame):

$$T_{Slot} = \frac{50}{b}$$

Within a slot a token frame or a Request-FDL-Status frame can be transmitted. The protocol slot time T_{SL} is assumed to be an integral multiple of the model slot time T_{Slot}:

$$T_{SL} = m \cdot T_{Slot}$$

The probability $p_I(i,j)$ is defined to be the probability that the current token owner includes a new station into the ring by the gap update mechanism (where i denotes the number of current ring members and j the number of stations in state Ready). Instead of modeling the gap timer, we have chosen to let every current token owner perform an independent Bernoulli trial. On success, a random station address is selected and a second random experiment is performed to check whether there is a station on the selected address and the Request-FDL-Status frame exchange is performed without error. The probability p_{Gap} that the current token owner performs a gap update trial is defined as:

$$p_{Gap} = \frac{q_1}{q_2} := \frac{\text{mean gap list length}}{\text{number of slots for } g \cdot T_{TTRT}}$$

where $q_1 = \frac{2\cdot(127-i)}{i}$ and $q_2 = \frac{g\cdot T_{TTRT}}{T_{Slot}}$. The factor 2 in q_1 accounts for the need of two time slots for exchange of the Request-FDL-Status frame and its answer frame. Hence

$$p_{Gap} = \frac{2\cdot(127-i)\cdot T_{Slot}}{i\cdot g\cdot T_{TTRT}}$$

The probability p_{AnsFDL} that a valid answer frame is received given that the current token owner has decided to perform a gap update trial to station k can be expressed as:

$$\begin{aligned} p_{AnsFDL} &= \text{Pr[frames correct and } k \text{ hits stn in gap list} \mid \text{Request-FDL-Status frame to } k] \\ &= \text{Pr[frames correct} \mid \text{Request-FDL-Status frame to } k] \\ &\quad \cdot \text{Pr[}k \text{ hits stn in gap list} \mid \text{Request-FDL-Status frame to } k] \\ &= p_{Req-FDL-Succ} \cdot p_{RequestHits} \end{aligned}$$

For i stations in the ring and j stations in state Ready, there are in the mean

$$p_{RequestHits} = \frac{j}{\frac{127-i}{i}}$$

ready stations in the token owners gap list. Hence,

$$p_{AnsFDL} = \frac{i\cdot j}{127-i}\cdot p_{Req-FDL-Succ}$$

Finally, the probability $p_I(i,j)$ that the current token owner includes a station is given by

$$\begin{aligned} p_I(i,j) &= p_{Gap}\cdot p_{AnsFDL} \\ &= \frac{2\cdot T_{Slot}\cdot j}{g\cdot T_{TTRT}}\cdot p_{Req-FDL-Succ} \end{aligned}$$

Next we determine an estimation jointly for p_{LU}, p_{AU} and p_{RU}, since they all occur in a scenario, where a station in a given state experiences a timeout and claims the token. A coarse estimation can be developed as follows: if we assume station addresses to be uniformly distributed over $[0,126]$ the mean station address is given by $e \approx 63$. The timeout time for a station with address n is given by

$$T_{TO} = T_{SL}\cdot(6+2\cdot n) = m\cdot T_{Slot}\cdot(6+2\cdot n)$$

so the mean timeout time is then $132\cdot T_{SL}$ (with $n=e$). We assume that each station in every slot performs an independent Bernoulli experiment with a fixed probability $p_{RU} = p_{LU} = p_{AU}$. Hence, the number of slots necessary for a single station experiencing a timeout is a geometric random variable with success probability p_{LU} (or p_{AU}, p_{RU} respectively) and mean value $\frac{1-p_{LU}}{p_{LU}}$. For this mean value the relation

$$T_{Slot} \cdot \frac{1-p_{LU}}{p_{LU}} = 132 \cdot m \cdot T_{Slot}$$

holds, and we conclude

$$p_{LU} = p_{AU} = p_{RU} = \frac{1}{1+132 \cdot m}$$

For determining the probability p_{AL} we note that error skipping can happen only if bit errors in the token frame occur somewhere in the address bytes and are not detected by the parity bit mechanism or by failures in the start- and stopbits (the token frame is not equipped with a checksum). If an error occurs in a start- or stopbit or in the start delimiter, this will be always detected. We restrict ourselves to the case where only two bit errors occur within a token frame, higher numbers of bit errors are neglected. The probability for exactly two bit errors is given by $b(2; 33, p)$. The number of placements of two errors within the address fields of a token frame, such that they are not detected is given by $16 \cdot 7$, since for the first erroneous bits there are 16 possibilities, while for the second we are restricted to the remaining seven bits of the same byte. Since the overall number of placing two bit errors over 33 bits is given by $\frac{33!}{(33-2)!}$ we finally have:

$$p_{AL} = b(2; 33, p) \cdot \frac{16 \cdot 7}{\frac{33!}{(33-2)!}} = \frac{7}{66} \cdot b(2; 33, p)$$

For the last unknown probability, $p_{LR}(i)$ (where i is the number of stations currently in the ring) we remember that two successive identical token cycles without transmission errors have to be detected. We can think of each token frame as a single Bernouilli experiment with success probability $\tilde{p} := p_{TokenCorrect} = (1-p)^{33}$. A station enters the state Ready, if it encounters a run of $2 \cdot i$ subsequent successes. For the probability f_n that at the n-th Bernouilli experiment we observe r subsequent successes (with $r = 2 \cdot i$) for the first time, it is known that its moment generating function is given by [47, Sec. XIII,7]

$$F(s) = \sum_{\nu=0}^{\infty} f_\nu s^\nu = \frac{\tilde{p}^r \cdot s^r}{1 - \tilde{q}s(1 + \tilde{p}s + \ldots + (\tilde{p}s)^r)}$$

where $\tilde{q} = 1 - \tilde{p}$ and the mean value given by

$$\mu(\tilde{p}, r) = F'(1) = \frac{1-\tilde{p}^r}{\tilde{q}\tilde{p}^r}$$

However, we do not use the random variable given by the f_n's as a model for the number of slots necessary for moving from Listen-Token to Ready, but instead we use a geometric random variable and set its mean value such that it equals $\mu(\tilde{p}, 2 \cdot i)$. Hence, the relation

$$\frac{1}{p_{LR}(i)} = \mu(\tilde{p}, 2 \cdot i)$$

should hold. From this we conclude

$$p_{LR}(i) = \frac{1}{\mu(\tilde{p}, 2 \cdot i)}$$

5.1.4 Simulation Results

We present simulation results for the global stability metrics defined in Section 5.1.2.

The simulations were performed with a detailed simulation model written in C++ and using the CSIM simulation library [103]. The model includes parts of the PROFIBUS link layer (SDA-, SDN, and SRD services), the PROFIBUS MAC protocol and a shared medium with the characteristics of the RS 485 PHY. In the shared medium all attached stations including the transmitter see the same signals and bits (a quite unrealistic assumption for wireless channels), hence the transmitter can perform proper hearback. All timing properties pertaining to the behavior of the medium (e.g., bit times, required idle times), and additionally a station's delay in processing received frames and generating answers are considered in the model.

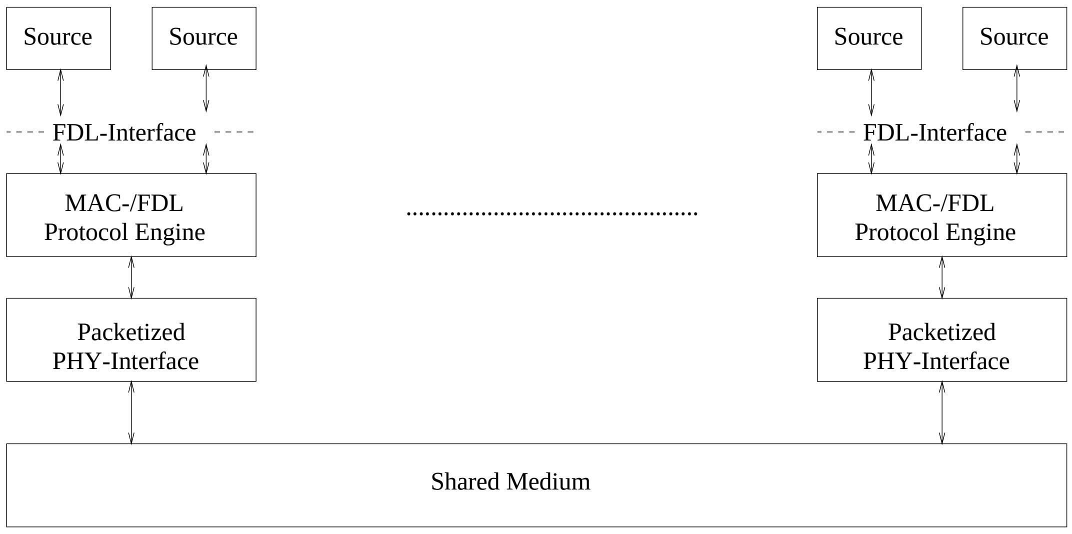

Figure 5.2: Logical structure of PROFIBUS simulation model

The structure of the simulation model is shown in Figure 5.2. A set of stations is attached to the shared medium. Each station consists of a variable number of traffic sources, attached via the FDL / link layer interface to the FDL-/MAC-protocol engine. The packetized PHY interface is an abstraction of the shared medium, delivering and accepting full packets instead of single bits. Furthermore some channel-related low level protocol functions are implemented here (e.g., timeout timer handling).

Simulator Validation

The simulator is validated by code inspection, successful comparison of generated frame sequences with expected frame sequences, and by comparison of the generated stability measures $\bar{N}$ and $\bar{M}$ with those generated by the analytical model described in Section 5.1.3. Both models were developed independently.

The fixed parameters common for both models are shown in Table 5.1. Both models assume independent channel errors with bit error rate (BER) p, varying from 10^{-4} to 10^{-3}. Furthermore, there is no load in the system, therefore only token frames and Request-FDL-Status frames occur on the

Parameter	**Value**
# of stations	$K = 10$
gap factor	$g = 6$
target token rotation time	$T_{TTRT} = 20$ msec
bit rate	$b = 500$ kBits/s
protocol slot time	$T_{SL} = 200\ \mu$s

Table 5.1: Fixed parameters common for the analytical model and the simulation model

Parameter	**Value**
# of stations	$K = 10$
gap factor	$g = 6$
target token rotation time	$T_{TTRT} = 20$ msec
bit rate	$b = 500$ kBits/s
protocol slot time	$T_{SL} = 400\ \mu$s
station delay	$100\ \mu$s

Table 5.2: Fixed parameters for ring stability simulations

medium.

The simulations used for validation are run for 3600 seconds. $N(t)$ is sampled every 100 μsec and the mean values $\bar{N}(3600)$ and $\bar{M}(3600)$ are computed from these values. The confidence intervals for $\bar{N}(3600)$ and $\bar{M}(3600)$ are very tight and thus not shown (see footnote 2).

In Figure 5.3 the values $\bar{N}(3600)$ from the simulation and $\bar{N}$ as from the analytical model are compared. Both curves have the same shape and their values differ by no more than 5%, however, towards higher BERs the simulation model gives better results (higher $\bar{N}$ value) than the analytical model. Given that the analytical model uses many simplifications (e.g., in the protocol state machine), the result is quite satisfactorily.

In figure 5.4 the values $\bar{M}(3600)$ from the simulations and $\bar{M}$ from the analytical model are displayed. While both curves have the same shape, the analytical model predicts that the ring is much more often incomplete than the simulation model does. From the view of the simulation model, for high BERs a $\approx 50\%$ increase in the $\bar{M}$ value is obtained with the analytical model. This is likely due to the coarse handling of time in the analytical model.

In summary, the good match of the $\bar{N}$ curves and the fact that the $\bar{M}$ curves have the same shape, increases confidence in the results delivered by the simulator.

Results

In the first set of simulations there are $K = 10$ stations without any external load, thus only token frames and Request-FDL-Status frames occur. We have chosen this setting for clearly highlighting the ring stability problems, simulations with load are discussed in Section 5.1.5.

Every station always wants to be a member of the ring and there are no failures except transmission

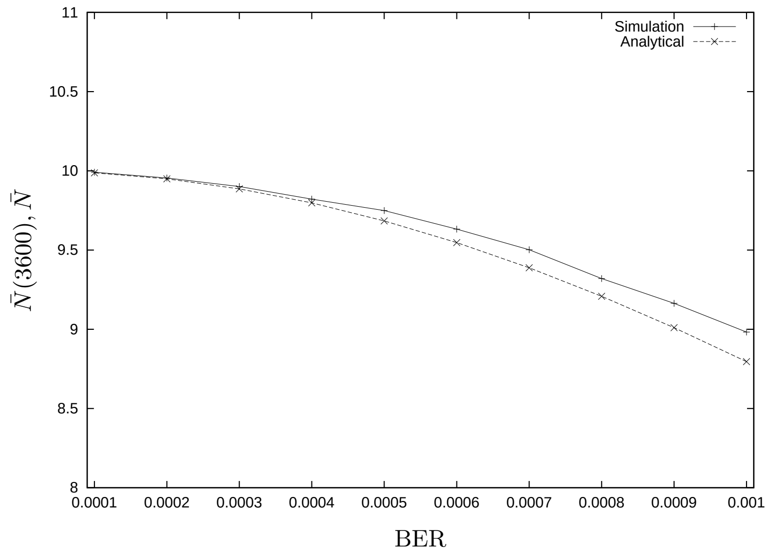

Figure 5.3: $\bar{N}(3600)$ and $\bar{N}$ vs. BER (independent errors)

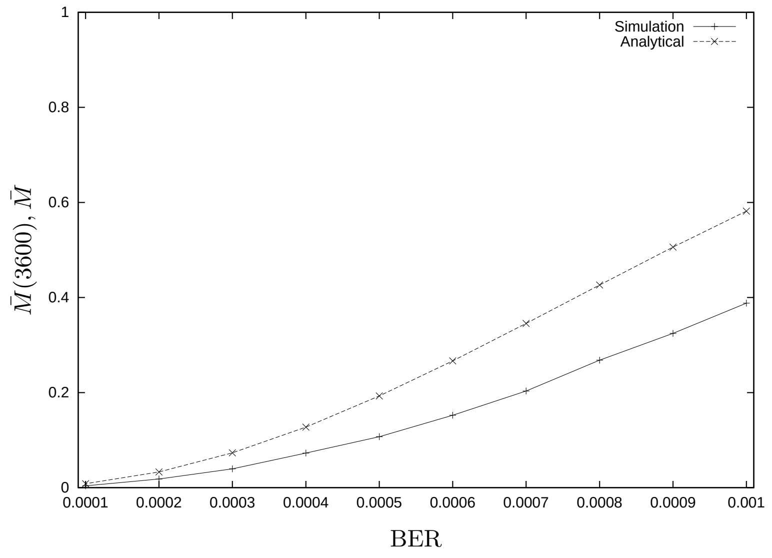

Figure 5.4: $\bar{M}(3600)$ and $\bar{M}$ vs. BER (independent errors)

errors. All simulations were run for 3600 simulated seconds, the fixed parameters are shown in Table 5.2. Two different error models were used: independent errors with fixed BER and the two-state Gilbert/Elliot model [184] (wireless channel models are discussed in Chapter 6). In the Gilbert/Elliot model (or Gilbert model for short) the channel is always in one of two states: *Good* or *Bad*. Within each state, bit errors are assumed to be independent with a fixed rate. The channel state is modulated according to a two-state continuous time markov chain. For parametrization of the Gilbert model four values suffice: BER in good state e_{g}, BER in bad state e_{b} ($e_{\mathrm{g}} \ll e_{\mathrm{b}}$), mean duration of good state λ in seconds, and mean duration of bad state μ in seconds. With $p_{\mathrm{g}} = \frac{\lambda}{\lambda+\mu}$ and $p_{\mathrm{b}} = \frac{\mu}{\lambda+\mu}$ being the steady-state probabilities for being in state good or bad, respectively, the mean BER m is given by

$$m = p_{\mathrm{g}} \cdot e_{\mathrm{g}} + p_{\mathrm{b}} \cdot e_{\mathrm{b}}. \tag{5.1}$$

The Gilbert model is very popular for modeling wireless channels due to its simplicity and its ability to capture bursty error behavior with short term correlation. The (mean) BERs for both error models are in the range $10^{-4} \dots 10^{-3}$ (these values are justified from measurements discussed in Chapter 6).

In Figures 5.5 and 5.6 $\bar{M}(3600)$ and $\bar{N}(3600)$ are displayed, respectively. In both figures we have used the independent error model with varying BER. Furthermore, in Figure 5.7 the distribution functions $C(s)$ (the random variable C indicates, how long a full ring is stable, see Section 5.1.2) for different BERs is shown. The nearly vertical line on the left side comes from the time resolution used (5 ms) and the fact that all distributions have a share between 5% and 21% of their mass within the first 5 ms. The confidence intervals for $\bar{N}(3600)$ are very tight and thus not shown[2]. In Figure 5.5, a nearly linear relationship between the BER and the fraction of time where the ring is incomplete can be observed. For the highest BER this fraction is approximately 1/3. Even more frustrating is the result that for the lowest investigated BER of 10^{-4} a full ring is stable for less than 15 seconds in more than 40% of all cases, although if $\bar{N}(3600)$ and $\bar{M}(3600)$ look good (compare Figure 5.7). This is a serious problem for real-time applications over error prone links, since for reincluding of a lost station some time is needed.

In order to show that the protocol is not only sensible to the overall BER but also to the characteristics of the error process (specifically: its "burstiness"), we have performed simulations with the Gilbert model. We have chosen to keep $m = 0.001$, $e_{\mathrm{g}} = 0.0000820$ and $\lambda = 0.061736$ fixed and to vary μ using values of 5, 10, 20, 30, 40, 50 and 60 ms[3], then determining e_{b} from equation 5.1. The burstiness index (BI) is defined to be λ/μ. The question, whether the ring stability metrics are invariant of the scale of λ and μ is not further investigated. In Figure 5.8 we present $\bar{N}(3600)$ vs BI. Apparently for more bursty errors (larger BI) this metric decreases. This is to be expected, since for constant m the value e_{b} increases when BI increases, it is more likely that a station experiences a hearback error.

As a visual impression that frequently the number of ring members reduces from five or more to one within a very short time, the evolution of $N(t)$ for the first 100 seconds is displayed in Figure 5.9 (Gilbert errors, $m = 0.001$, $\mu = 20$ ms). A careful analysis of the corresponding simulator traces shows that often multiple stations are lost simultaneously, and that these breakdowns are indeed caused by

[2] The maximum relative error of the $\bar{N}(\cdot)$ value for all simulations is with 98 percent confidence not larger than one percent of the absolute value. Most relative errors are smaller than 0.1 percent. For actually calculating these values within the simulation, $N(t)$ was approximated by a sampled version $N_k = N(k \cdot T)$ with $T = 100\mu$s fixed and $k \in \mathbb{N}$. Accordingly, we calculate $\bar{N}(t)$ with $k_t = \max\{k \in \mathbb{N} : k \cdot T < t\}$ as the sample mean: $\bar{N}(t) = \frac{1}{k_t} \sum_{i=0}^{k_t} N_i$ and the variance $\bar{N}^2(t)$ as the sample variance. Furthermore, in the simulator transient removal techniques were used for achieving steady-state results.

[3] The values for λ and e_{g} are calculated directly from [184], while the values chosen for μ have the same order of magnitude as those from [184].

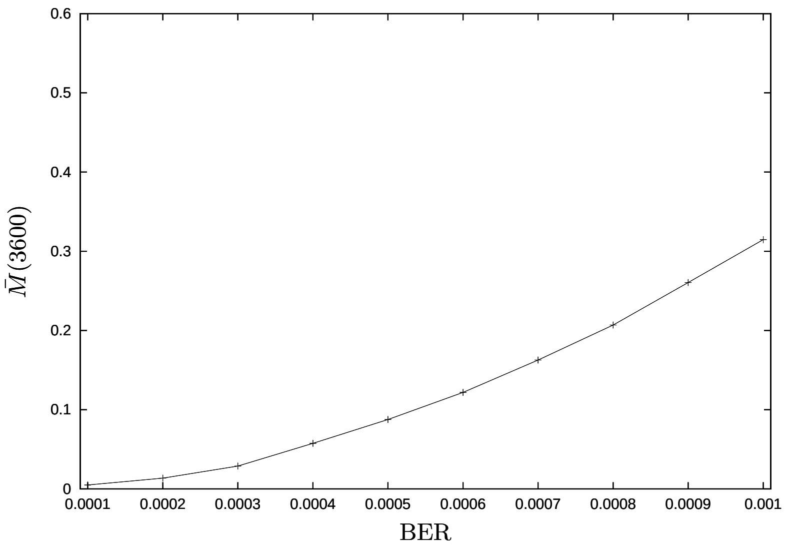

Figure 5.5: $\bar{M}(3600)$ vs. BER (independent errors)

Parameter	**Value**
# of active stations	4
# of passive stations	1
target token rotation time	$T_{TTRT} = 20$ msec
bit rate	$b = 500$ kBits/s
protocol slot time	$T_{SL} = 400\ \mu$s
station delay	100 μs

Table 5.3: Fixed parameters for ring re-inclusion simulations

the ring jacking scenario. Furthermore, the frequent transitions from ten members to nine members are caused by hearback removals. The error skipping scenario is rare: for the worst error parameter setting (Gilbert errors, $m = 0.001$, $\mu = 5$ ms, $e_b \approx 0.012$) a token frame with undetectable errors is observed once every minute in the mean. Therefore this scenario is not considered furthermore.

A station lost from the ring must be re-included by another station, using the ring maintenance mechanisms. This may take some time. For an assessment of this time another set of simulations was performed [189], using the parameters shown in Table 5.3. The scenario consists of four active and one passive station. The active stations addresses are 22, 39, 65 and 69, taken from uniform distribution. The active stations offer a load of $\approx 20\%$ if all stations are in the ring. The channel generates independent errors with a fixed BER of 10^{-3}. We have investigated *station outage times*. From the point of view of a single station a station outage time is the time duration between the instant the station gets lost from the ring and its later reinclusion. In Figure 5.10 the mean station outage time (MSOT) is shown vs. the gap factor. The gap factor is expected to have significant influence on the ring (re-)inclusion times, and indeed this is confirmed by the figure. In Figure 5.11 the cumulated station outage time (CSOT) is shown, i.e., the fraction of time that a station is not in the ring. The following points are interesting:

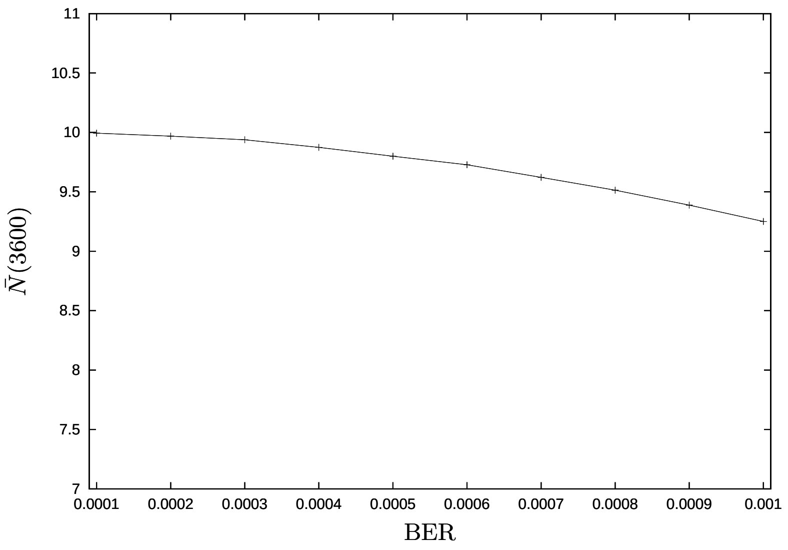

Figure 5.6: $\bar{N}(3600)$ vs. BER (independent errors)

- For all stations except station 22 (lowest station address) the MSOT increases almost linearly with the gap factor. Even more, the slope is greater for higher station adresses. This can be explained by the ring jacking scenario described in Section 5.1.1. After a station is lost, it will take some time to get re-included.
- The results on the cumulated station outage times are dramatic: for gap factors of around 30 all active stations except station 22 are ring members for only 50% of the time. This gets worse for higher gap factors. Even for small gap factors these stations are for approximately 10 % of the time not member of the ring. This shows clearly that the used deterministic algorithm for station inclusion breaks down under a high bit error rate.

The MSOT and CSOT values show the same behavior for a Gilbert channel. Furthermore, varying the T_{TTRT} value gives equivalent results. The MSOT values increase for increasing load, since there is less time available for Request-FDL-Status frames. However, the CSOT value decreases for increased load, since with more data frames the number of vulnerable token frames per fixed unit of time decreases, and thus fewer stations get lost [189].

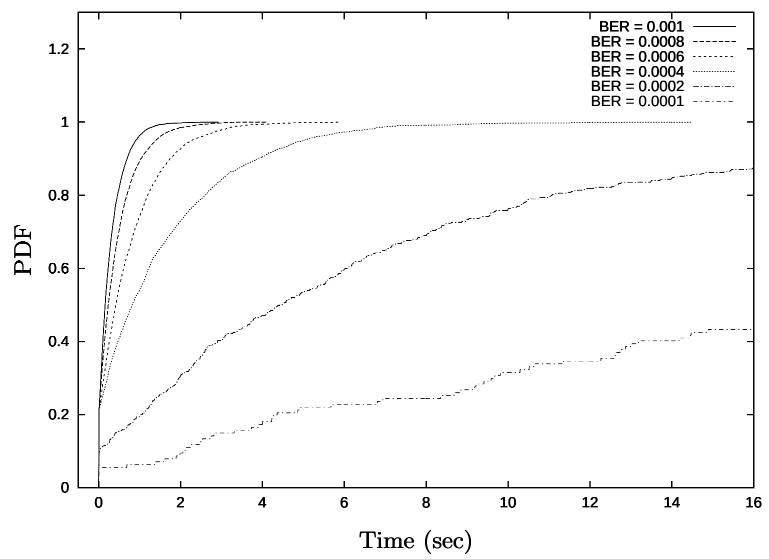

Figure 5.7: Distribution function C(s) for different BER's

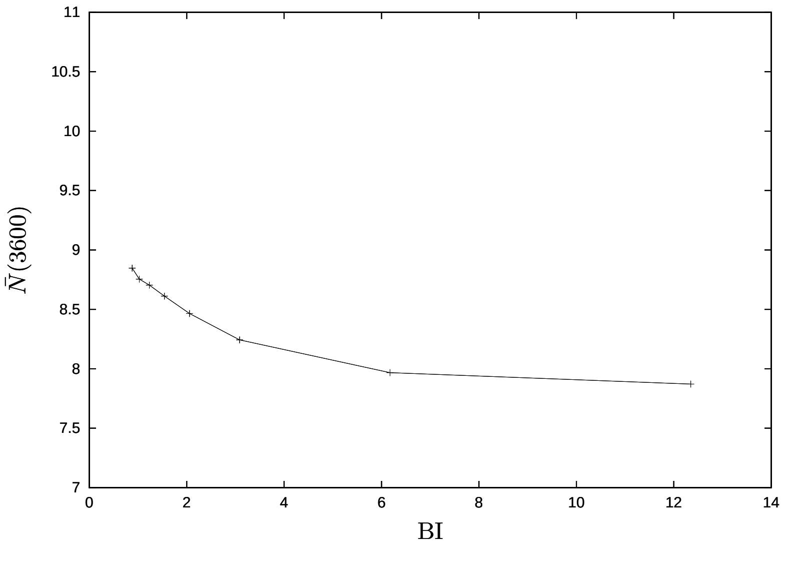

Figure 5.8: $\bar{N}(3600)$ vs. BI for m = 0.001 (Gilbert errors)

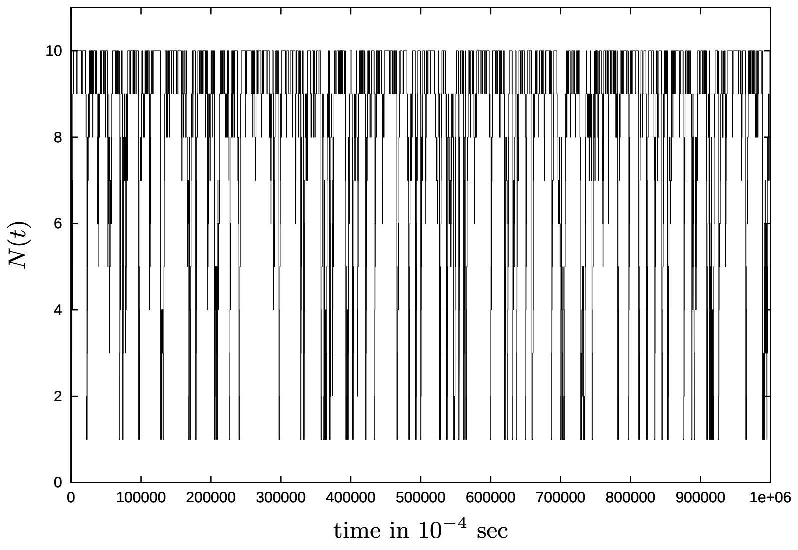

Figure 5.9: $N(t)$ vs. time (Gilbert errors)

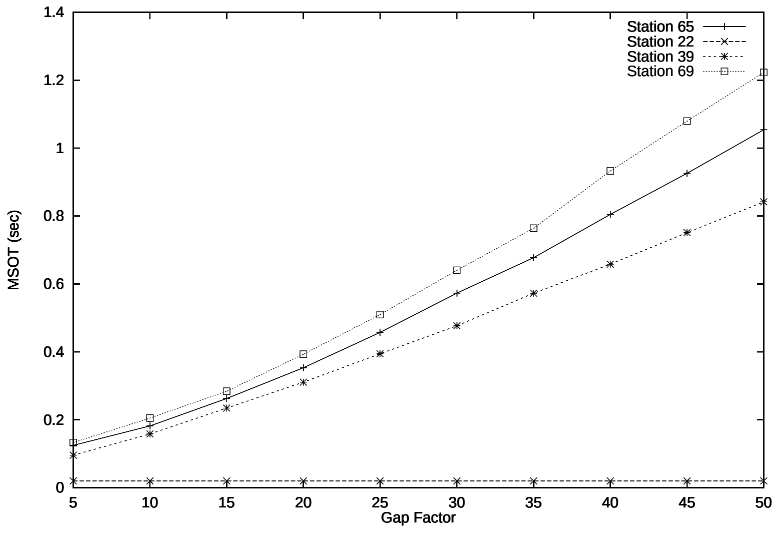

Figure 5.10: MSOT vs. gap factor (independent errors)

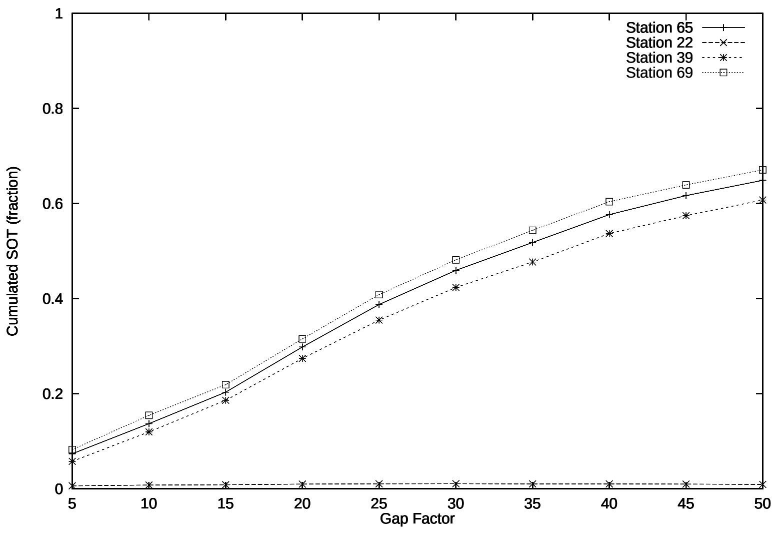

Figure 5.11: CSOT vs. gap factor (independent errors)

5.1.5 Improvements

In this Section we propose a new method for setting timeout timers and an additional protocol feature. The new timer setting tries to prevent the breakdowns of the ring by letting expire the timeout timer for current ring members before that of stations in state Listen-Token. The additional protocol feature aims at reincluding lost stations as fast as possible. Since both of them require no modification of frame formats or protocol operation, they are interoperable with the unchanged protocol. Thus, stations with the modified and the unchanged protocol stack can potentially be operated in the same PROFIBUS LAN. However, the ability to dynamically influence the timeout timer setting is needed, which may require an upgrade of today's ASIC-based protocol implementations. Both methods are targeted to combatting the ring jacking and hearback removal scenarios, avoiding the error skipping scenario requires a better protection of the token frame and thus a change in frame formats.

The effect of the proposed methods is investigated with simulations, using the same scenarios and stability metrics as in Section 5.1.4, and with additional simulations taking the effects of system load and different numbers of active stations into account.

Timeout Calculation

From our simulations and from analysis we have observed that the ring jacking scenario (described in Section 5.1.1), where the station with the lowest address can destroy the whole ring, occurs frequently. The calculation of the timeout value is for station n as follows (see Section 4.1.4):

$$T_{TO}(n) = (6 + 2 \cdot n) \cdot T_{SL}$$

where T_{SL} is the protocol slot time. The basic problem of this scenario is that the timeout timer may expire for a station which is in the listen token state and has no valid LAS. If the timer of a station in the ring (not in the listen token state) expires, the ring keeps alive. Thus we propose to make the timeout calculation state-dependent:

$$T_{TO}(n) = \begin{cases} (6 + 2 \cdot n) \cdot T_{SL} & : \text{ state} \neq \text{listen token} \\ (254 + 6 + 2 \cdot n) \cdot T_{SL} & : \text{ state} = \text{listen token} \end{cases}$$

in order to make sure that the timeout timer expires first for stations in the ring and as a result to avoid ring jacking. The effects of this improvement are shown later in this section.

Fast Reinclusion of Lost Stations

When a station is lost from the ring, it takes some time before it is reincluded. First, the station is required to observe the same sequence of token frames twice, second, it will not be reincluded before it is pinged by its predecessor using the Request-FDL-Status frame. We propose to add the following feature to the protocol: after station a has lost its successor b (i.e. there is no reaction of b to three consecutive token frames), a waits for two token cycles and then pings b with the Request-FDL-Status frame as soon as there is token holding time available. This is the earliest moment where b can be reincluded, due to b's need for observing two identical token cycles. This procedure should be carried out independently of the normal ring-inclusion algorithm. Thus it can happen that a includes another station c during the two token cycles it waits for reincluding b. In this case b should only be reincluded if its address lies in the range between a and c, otherwise c will remove itself from the ring, being skipped by the first token frame a sends to b. However, when the ring jacking scenario occurs more frequently, this protocol extension should be used in conjunction with the new timeout calculation method, since otherwise fast reinclusion will not happen.

Performance Evaluation

Three different versions of the protocol are compared: the normal protocol without any improvements, the protocol with the new timeout calculation method and the protocol with both improvements. The simulation setup is the same as in Section 5.1.4. The results for $\bar{M}(3600)$ are shown in Figure 5.12, the results for $\bar{N}(3600)$ are shown in Figure 5.13, both for independent errors and varying BER. These figures show that the new timeout computation significantly improves stability, the protocol with both improvements performs best. In Figure 5.14 the sample coefficient of variation for N is shown. The improvements reduce the variability of N. In Figure 5.15 the three protocol versions for the case of Gilbert errors and varying BI for fixed mean BER $m = 0.001$ are compared. The stability gain of the improvements as compared to the normal protocol is larger for more bursty errors than for the "smooth" independent errors. As a visual impression in Figure 5.20 the evolution of $N(t)$ for the same system as for Figure 5.9 (ten masters, no load, Gilbert errors with $m = 0.001$ and $\mu = 20$ ms) is shown, however, with both protocol improvements enabled. It can be seen that most of the breakdowns visible in Figure 5.9 are removed.

Additionally, the ring jacking scenario also influences the local stability metrics mentioned in Section 5.1.2. One example is the fraction of time that station i is not in the ring. For the station with the lowest address this fraction is small and nearly independent of the gap factor or the T_{TTRT} value, while for all other stations this metric depends almost linearly on the gap factor, and furthermore increases with increasing station address (see ref. [189] for examples).

In order to show that ring stability problems occur also when there is load in the system (and thus a smaller number of vulnerable token frames per fixed unit of time), two more scenarios were investigated. In the first scenario there are four active stations, two passive stations, and four traffic sources, each attached to a different active station. The traffic sources generate requests, the attached station puts them in a queue of infinite size. Two traffic sources generate requests with a fixed interarrival time of 10 ms. The corresponding requests lead to frames of 25 bytes size (carrying 16 bytes of user data), which are acknowledged by the passive station with frames of the same size. The other sources generate sporadic requests with exponentially distributed interarrival times (10 ms mean value), destined for the second passive station and with data sizes uniformly distributed between 8 and 30 bytes (leading to frame sizes between 17 and 39 bytes), however, the acknowledgement carries no data. Thus, there is a mixture of synchronous and asynchronous traffic.

In the second scenario there are ten active stations and ten traffic sources. The first five sources are periodic (with 25 ms period), the other sources are sporadic (with 25 ms mean value). Thus in both scenarios a minimum bandwidth of $\approx 35\%$ of the medium bandwidth is devoted to exchange of data frames including the acknowledgements, but not including retransmissions. The need for retransmissions at error rates of $\approx 10^{-3}$ saturates the system, higher loads lead to growing request queues. This is true especially for independent errors, for Gilbert errors the queues can be emptied during good channel periods. The simulations run for 10000 simulated seconds, the other parameters (gap factor, T_{TTRT}, bit rate, slot time T_{SL}) are fixed. The $\bar{N}(10000)$ results for the scenario with ten stations are shown in Figure 5.16 (independent errors) and Figure 5.17 (Gilbert errors). It can be seen that for all three protocol versions and both error models this value is better than in the corresponding simulations without any load. However, for high bit error rates the stability problems and their dependence on the type of channel errors are still visible. The proposed improvements again yield a significant gain.

The $\bar{M}(10000)$ values for both station numbers are shown in Figures 5.18 (independent errors) and 5.19 (Gilbert errors) for the normal protocol and the protocol with both improvements. Again, in the presence of load this metric is better (lower) than for the corresponding simulations without load (not shown here for Gilbert errors), and the improved protocol version yields the best results. Interestingly, in both figures the numbers are smaller for fewer stations. While for four stations and ten stations the times for breaking a full ring are comparable (four stations: mean value $\bar{C} \approx 1.12$ sec, stddev ≈ 1.38; ten stations: mean $\bar{C} \approx 1.17$ sec, stddev ≈ 1.27) with ten stations it takes much longer to complete the ring. Likely the difference stems from the time needed to complete the ring after multiple stations have been lost at once, as in the ring jacking scenario. If only a single station gets lost, it is reasonable to expect that reinclusion is slightly faster in the ten station case, since the gap lists are typically shorter than with fewer stations. Furthermore, for a newly reincluded station there might be some delay between its reinclusion and the time it starts to poll its gap list, since in the simulation the gap timer is independent from the stations state of ring membership. As a result, if more stations need to be reincluded, a higher delay for ring completion can be expected.

All these findings together confirm the belief that ring instability is an issue for higher bit error rates, and furthermore that two important sources for instability are the ring jacking and hearback removal scenario, while the error skipping scenario seems to play a much smaller role. The ring jacking and hearback removal scenarios can be combatted with the two proposed improvements. Since for lower bit error rates station losses occur rarely and the improvements are not invoked, they impose no additional cost in terms of bandwidth or delay.

However, even with these improvements the PROFIBUS protocol is not a good choice. As we show in Chapter 7, the achievable *realtime performance* for the PROFIBUS is for many scenarios much worse than for the investigated polling-based protocols. And in Section 5.2 we show the reason for this: when faced to a wireless-type link with *packet losses* (see Chapter 6), the stability problems are still significant.

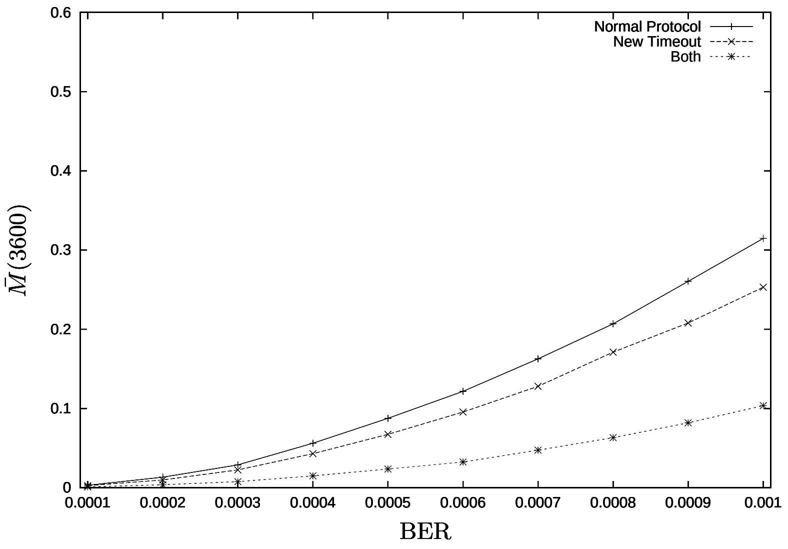

Figure 5.12: $\bar{M}(3600)$ vs. BER (independent errors)

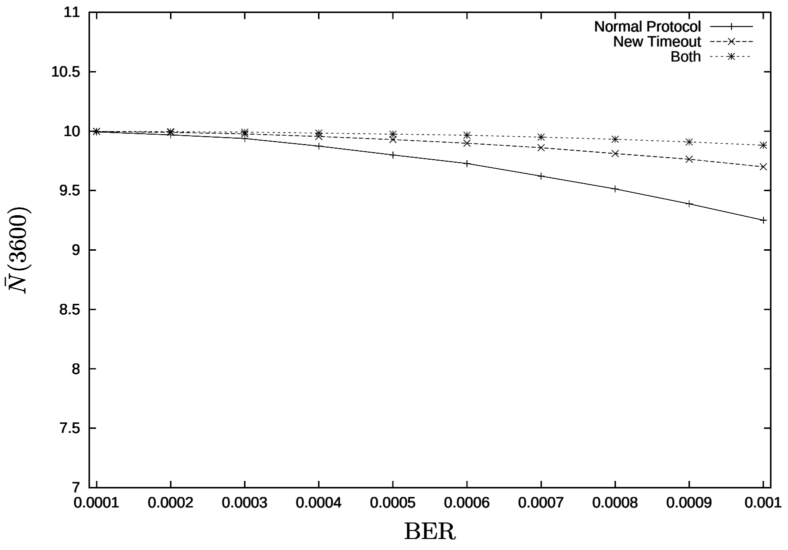

Figure 5.13: $\bar{N}(3600)$ vs. BER (independent Errors)

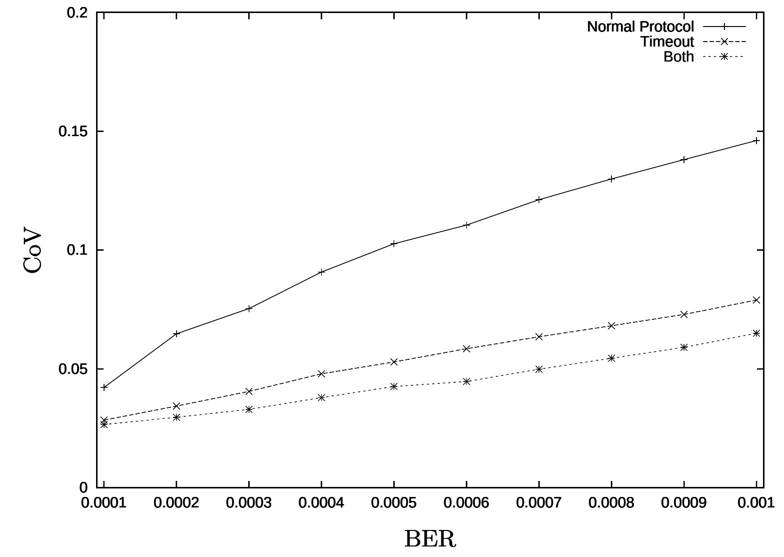

Figure 5.14: Sample coefficients of variation for N vs. BER (independent errors)

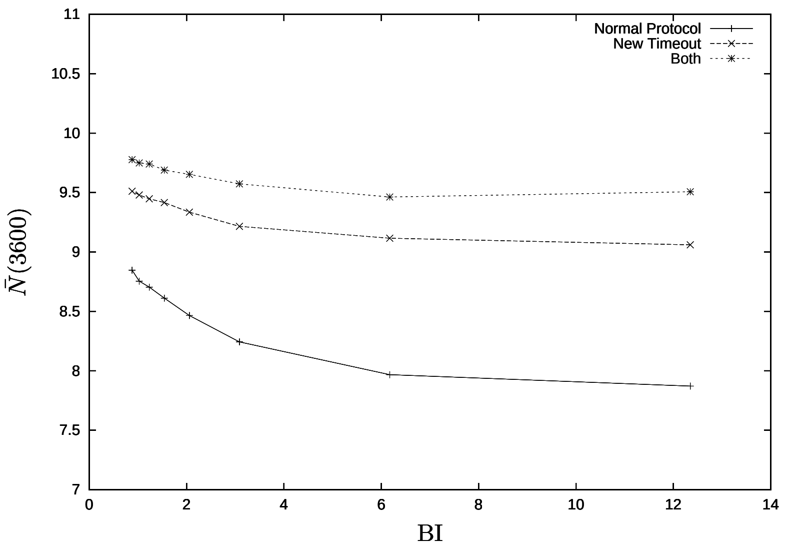

Figure 5.15: $\bar{N}(3600)$ vs. BI for m = 0.001 (Gilbert errors)

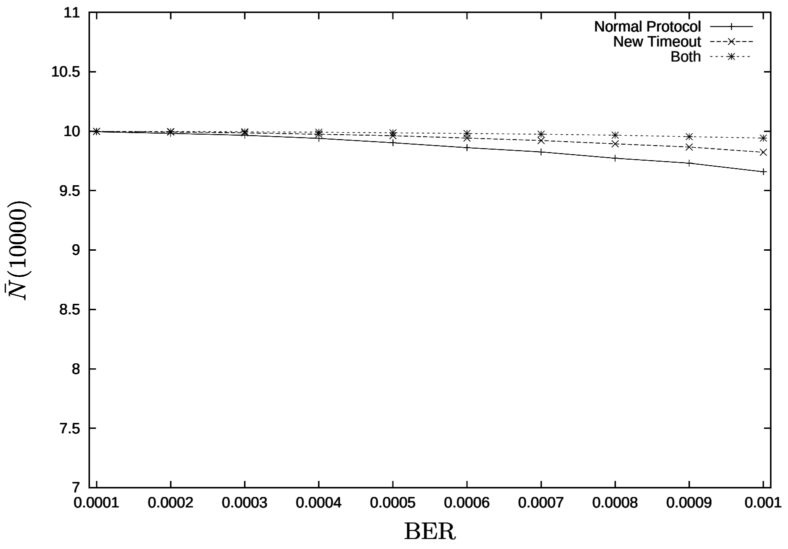

Figure 5.16: $\bar{N}(10000)$ vs. BER (independent errors) with 10 masters and $\approx 36\%$ load

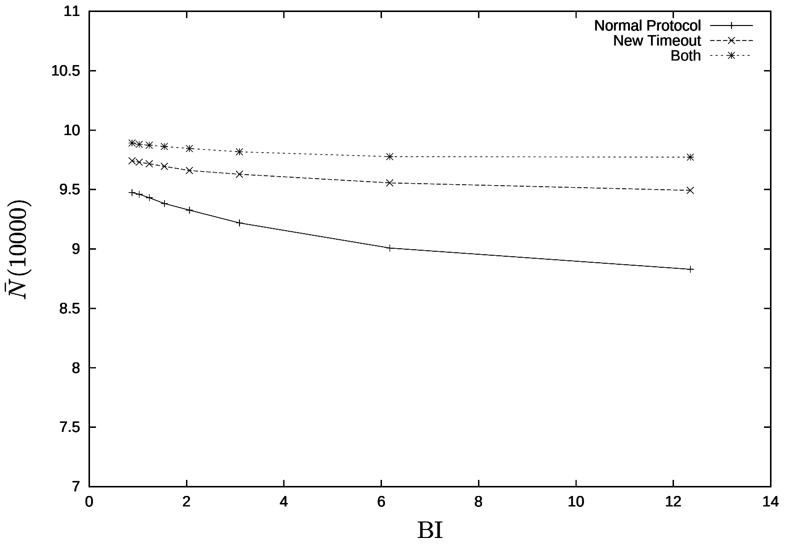

Figure 5.17: $\bar{N}(10000)$ vs. BI for m = 0.001 (Gilbert errors) with 10 masters and $\approx 36\%$ load

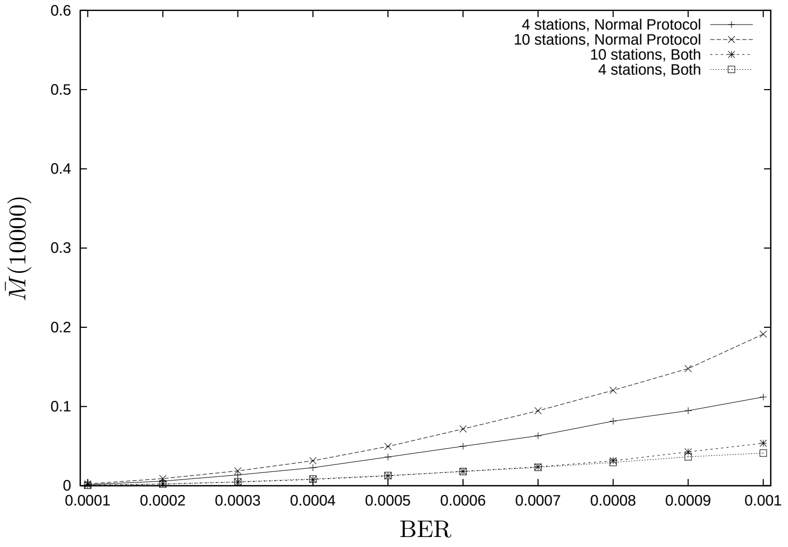

Figure 5.18: $\bar{M}(10000)$ vs. BER (independent errors) with 4 and 10 masters and $\approx 36\%$ load

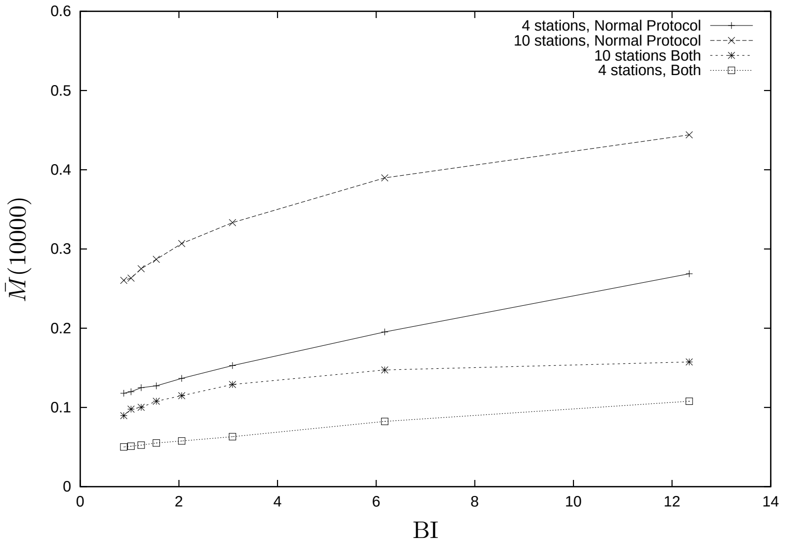

Figure 5.19: $\bar{M}(10000)$ vs. BI $m = 0.001$ (Gilbert errors) with 4 and 10 masters and $\approx 36\%$ load

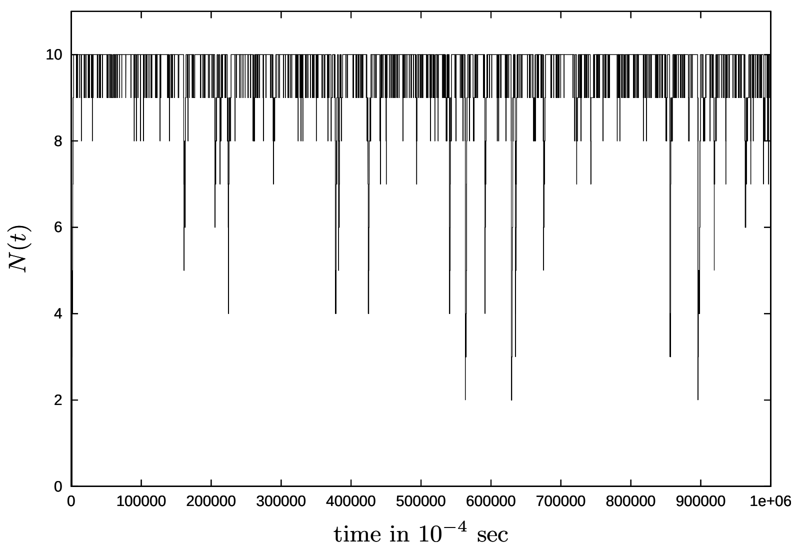

Figure 5.20: $N(t)$ vs. time (Gilbert errors, both protocol improvements)

Parameter	**Value**
# of stations	$K = 10$
gap factor	$g = 6$
target token rotation time	$T_{TTRT} = 20$ msec
bit rate	$b = 1$ MBit/s
protocol slot time	$T_{SL} = 400\ \mu$s
station delay	$100\ \mu$s

Table 5.4: Fixed parameters for ring stability simulations over wireless channel

5.2 PROFIBUS over Wireless Links

When the PROFIBUS protocol runs on top of an IEEE 802.11 DSSS PHY, some properties of data transmission change. First, since it is impossible to send and receive simultaneously on the same channel, the hearback feature is not available. Hence, it is reasonable to expect that the ring jacking scenario occurs less often. Second, every frame is preceded by the 192 μs long PLCP preamble and PLCP header, which affects at least some protocol parameters like the slot time T_{SL}. And third, based on the results reported in Chapter 6, it is reasonable to extend the channel error model with packet losses.

To investigate the PROFIBUS ring stability under wireless conditions, some changes in the protocol, the channel model and simulation parameters were necessary. With respect to the protocol and framing rules the following assumptions were made:

- PROFIBUS frames are embedded as they are into the data part of an 802.11 DSSS PHY PPDUs (see Section 3.2.2). Hence, no 802.11 MAC fields were present and implicit broadcasting is used.
- Every byte is transmitted with eight bits instead of eleven.
- The PLCP preamble and header are assumed to have a length of 192 μs. In addition, to every frame a 16 bit CRC checksum is appended, which for simplicity is assumed to detect all bit errors.
- The hearback feature is not available, hence, collisions could not be detected.
- The protocol slot time T_{SL} is at least 400 μs.
- The timeout timer can rely on a “true” carrier sensing facility, which indicates a carrier if the received signal strength exceeds some threshold and does not require having achieved bit synchronization.

The channel model was changed to incorporate not only bit errors but also losses of whole packets, which can occur due to failure of preamble acquisition (Chapter 6). However, all stations see the same signals on the medium.

Two simple sets of simulations were performed, differing in their respective BER: the first set uses independent errors with a BER of 10^{-3}, in the second set there occur no bit errors, but only packet losses. In both sets the packet loss rate (PLR) is varied from 0.0 to 0.1 in steps of 0.01, assuming independent packet losses. These PLRs are well in the range observed by measurements (see Section 6.5.2).

The scenario consists of $K = 10$ stations with no data load, as in Section 5.1.4 for the PROFIBUS over RS-485 case. The fixed simulation parameters are summarized in Table 5.4. The simulations are run for 3600 simulated seconds.

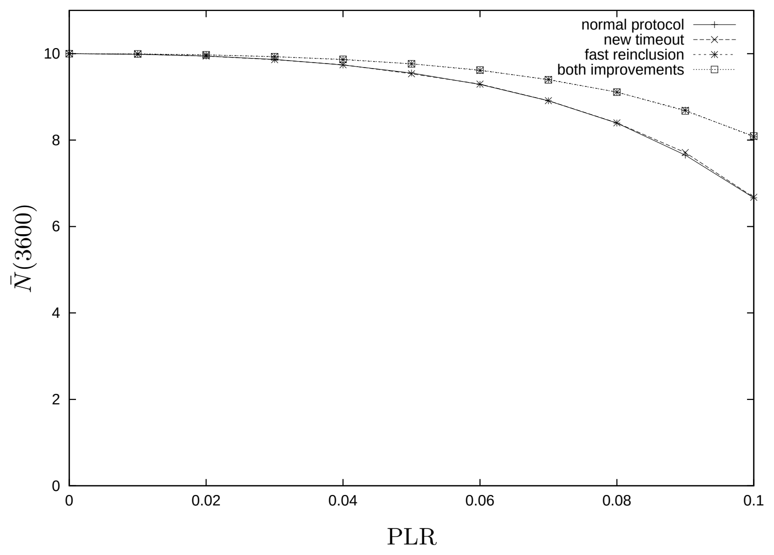

Figure 5.21: $\bar{N}(3600)$ vs. PLR (independent packet losses) and no bit errors

For the case without bit errors in Figure 5.21 the $\bar{N}(3600)$ values are presented for varying PLR, while in Figure 5.22 the $\bar{M}(3600)$ values are displayed. Respectively, for the case with bit errors the corresponding Figures are Figure 5.23 for the $\bar{N}(3600)$ values and 5.24 for the $\bar{M}(3600)$ values. The following points are important:

- The ring stability is sensitive to packet losses. Even for 6% packet losses without bit errors $\approx$ 50% of the time the ring is not full, while enabling both improvements reduce this fraction to $\approx$ 30%. However, this is unacceptable for time critical communications. If in addition bit errors occur, with both improvements the ring is not complete for $\approx$ 56% of the time, while the unchanged protocol completely increases this rate to $\approx$ 78%.
- In all figures the curves for the normal protocol and the protocol with the new timeout computation method are very close, the same holds for the curves for both improvements and the fast reinclusion feature. Hence, only the fast reinclusion feature gives some gain in ring stability, while the new timeout method gains nothing. For the RS-485 simulations with hearback the opposite behavior could be observed. This allows to draw the conclusion that the ring jacking scenario occurs only rarely.

To summarize, the original PROFIBUS protocol behaves inacceptably for moderate packet loss rates of 5% to 10%, as observed in measurements. However, with a wireless link the ring jacking scenario is not the dominant source of ring instability. Instead, as seen by inspection of simulation logfiles, in most cases stations get lost because the token frames do not reach them due to packet losses.

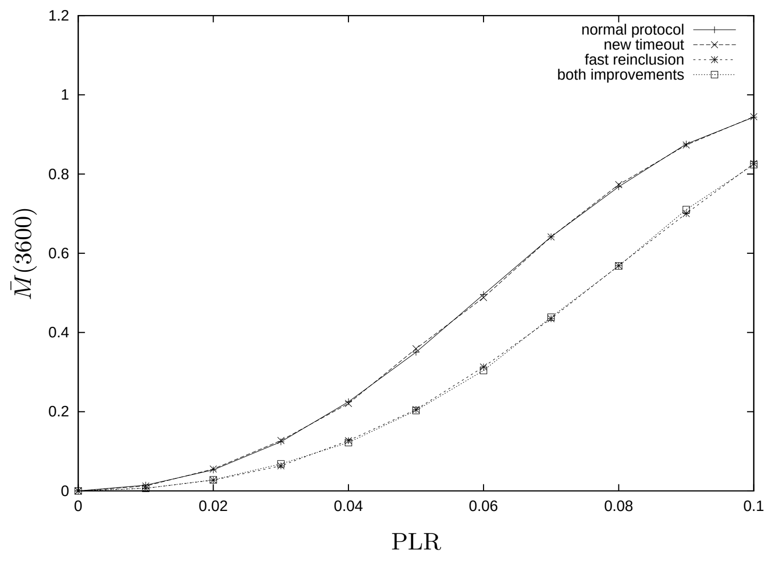

Figure 5.22: $\bar{M}(3600)$ vs. PLR (independent packet losses) and no bit errors

A worthwhile topic for further research would be to investigate other packet loss patterns, e.g., bursty patterns or to use directly some traces from measurements. It is also interesting to see what happens when the "single-channel" assumption is dropped and between every pair of stations a separate channel is used. Some results for realtime performance in this case are presented in Section 7.3.2.

5.3 Related Work

The behavior of PROFIBUS in the presence of transmission errors or its ring membership behavior / ring stability is to our best knowledge not covered in the literature. Most analyses of the PROFIBUS real-time capabilities ([169], [168, chap. 3, chap. 5]) allow for sporadic transmission errors by taking retransmissions into account, however, the influence of transient times where a station is involuntarily not a ring member is not considered.

For the IEEE 802.4 Token Bus it is investigated in [82] using analytical techniques and measurements, how bursty errors affect the token passing process, and how this in turn affects the mean token passing time and, more important, the mean token rotation time. For the PROFIBUS some results on local stability metrics are available in [189].

5.4 Conclusions

The PROFIBUS protocol was not designed with error prone links in mind. This is manifest in some design decisions (e.g., to run member (re-)inclusion at low priority, to use only a weak checksum algorithm, and to not protect the token frame at all) and in the poor ring stability delivered by the protocol both over RS-485 and wireless-type channels. The need for explicit token passing and permanent ring maintenance makes the protocol vulnerable to frame losses or (undetected) bit errors.

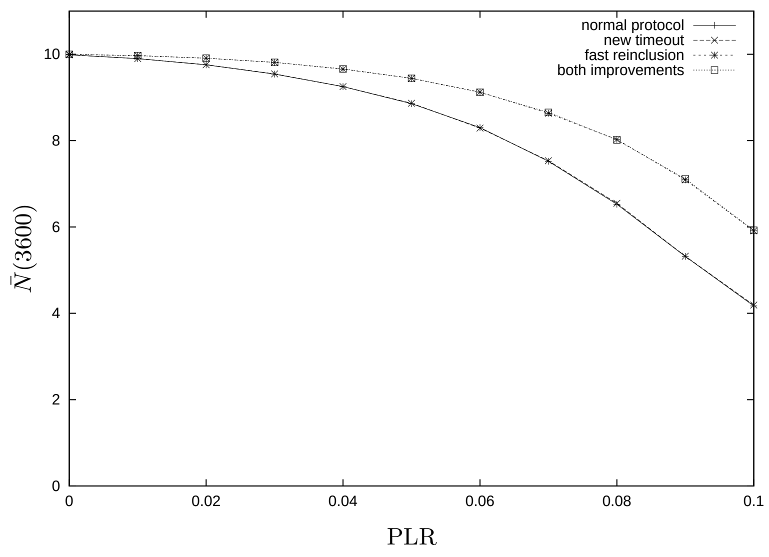

Figure 5.23: $\bar{N}(3600)$ vs. PLR (independent packet losses) and BER of 10^{-3}

In the case of the RS-485 link with hearback, it is especially the ring jacking scenario, which influences ring stability, while in the wireless case it is the loss of token frames. In general, maintaining a distributed state for ring membership is vulnerable in the presence of link errors.

The bad thing about ring instability is that it may take some time to re-include lost members. During these outage times the stations are not allowed to transmit data, no matter how critical they are.

Another disadvantage of explicit token passing not discussed so far is the fact that it explicitly requires a fully meshed topology, i.e., every station must be able to hear all other stations. Modifying the token passing process such that partially meshed topologies are possible and can be integrated with a wired PROFIBUS segment is at least challenging.

The proposed improvements can help a lot on wired-type links without packet losses, but as is shown in Section 5.2, the protocol is vulnerable against packet losses on wireless-type links. The improvements do not avoid them, but they just help in re-including a lost station faster. As we will show in Chapter 7, indeed for wireless-type media the stability problems are serious, hence, the realtime performance of the PROFIBUS protocol is inferior in most cases as compared to the polling-based protocols discussed there.

To summarize, the existing PROFIBUS protocol is not a good candidate for being used on top of a wireless PHY.

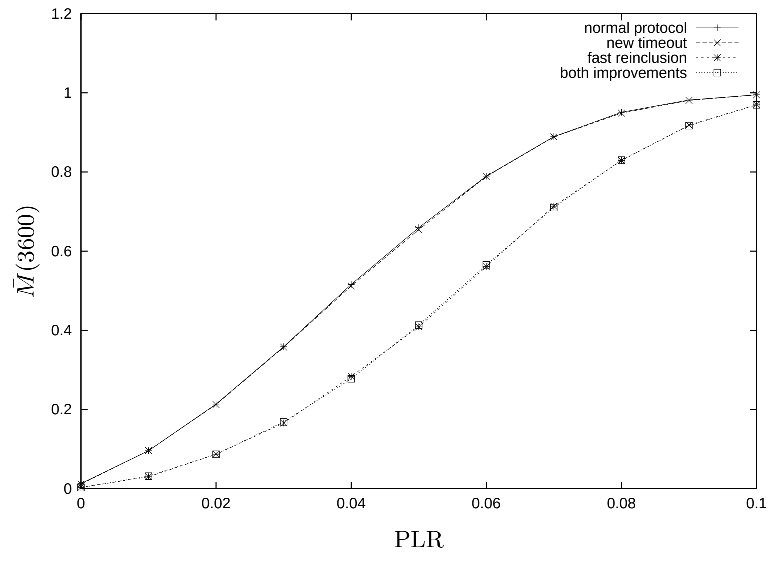

Figure 5.24: $\bar{M}(3600)$ vs. PLR (independent packet losses) and BER of 10^{-3}

Chapter 6

Error Behavior of Wireless Channels and its Modeling

In order to design MAC and link-layer protocols with good real-time performance, it is vital to have some understanding of the error patterns exhibited by the PHY or *wireless link* (this notion is used as an abstraction of the ensemble of transmitter, receiver, spread spectrum modem, the channel, the scrambler, high and intermediate frequency circuitry, and more). This knowledge is important for several reasons:

- The same protocol can show different behavior and performance for different error characteristics. For example, the results presented in references [201] and [200] show that bursty (Markovian) bit errors are beneficial for the performance of TCP, as compared to the case of independent errors with the same mean bit error rate. For the PROFIBUS independent errors result in better delay performance and stability of the logical token passing ring than bursty errors [189], as is demonstrated in Chapter 5.
- Advance knowledge of the error characteristics can help the protocol designer to select appropriate protocol mechanisms, e.g., to choose suitable forward error correction (FEC) schemes or to find good rules on when to perform retransmissions. As a simple example, time-variable characteristics call for adaptive mechanisms.
- The application layer software and applications are affected, since, in contrast to cable-based communications, they cannot assume channel outage conditions to be a rare exception. Instead, safety critical applications over wireless links need to be designed taking longer channel outages explicitly into account. As a prerequisite, the MAC protocol has to detect channel outages and to signal these conditions to the applications.

It is often convenient for protocol designers to evaluate protocols with simulations before developing a prototype implementation and performing complex measurements. A key part of such simulations are link error models, which, in principle, determine for a transmitted packet which of the receiver stations see bit errors; sometimes the exact position of errors within a packet is of interest. Often, simulation-based performance evaluations are done with *stochastic* link error models, as a convenient alternative to handling large and clumsy measurement traces. In these models a simple stochastic process, which often can be described in terms of a few parameters, is used to generate bit error and packet loss patterns. Most of the stochastic models are not designed to reflect or model physical phenomena, but simply to reproduce the statistics of given or conjectured error patterns with some accuracy. Hence, their computational complexity is typically low as compared to, e.g., realistic channel models based on ray tracing. This simplicity is beneficial for packet level simulations, since often the model has to be applied to hundreds of thousands of packets and simulation time is becoming an issue. However, there is a tradeoff between a model's complexity (and its number of parameters) on the one hand, and its qualities in matching some given statistics or in giving good performance predictions on the other hand.

This chapter provides the necessary input for the design of polling schemes for wireless PROFIBUS, and for the stochastic channel models needed for simulation. The foundation is laid by a measurement study of wireless link error characteristics in an industrial environment. The focus of this study is not in "explaining" the results in terms of "physical" phenomena (such as noise sources, propagation characteristics), but on the statistics of the packet loss and bit error patterns delivered by the wireless link (via its interface provided by the *baseband processor*, see Section 6.2.1) to the MAC and link-layer protocol. The results of these measurements are then used in different ways:

- The results allow to draw some basic conclusions regarding the design of MAC and link-layer protocols for a wireless PROFIBUS aiming at achieving good realtime performance.
- It turns out that some of the popular wireless error models, e.g., the independent model and the Gilbert/Elliot model, are not adequate. The statistics of errors generated by the models differ significantly from those of the traces. Furthermore, for an example communication system, the predictions of selected performance parameters as generated by the simple stochastic models deviate significantly from those where a trace is used (see Section 6.7.3). This is taken as an incentive to design an alternative type of error models, the "bipartite" model.
- The measurement data provide "real-world" parameters for several stochastic models.
- A structure for an overall channel model can be derived, clearly separating the issues of packet losses and bit errors.

The measurements were done in an industrial environment to be relevant specifically for design and simulation of MAC protocols for wireless fieldbus systems. In order to achieve long-term results and to assess the influence of different parameters, we have chosen to restrict to a single scenario (short distance non line-of-sight scenario in a factory building).

For the measurements an IEEE 802.11-compliant radio modem with DSSS modulation was used. It was possible to obtain a chipset without any upper layer (MAC) functionality (Harris/Intersil PRISM I chipset as MACless version [2] [73]). When this study started, this chipset was very popular and used in commercial wireless local area network (WLAN) products.

The measurement setup is constructed such that there is no bias introduced by upper layer protocols or operating systems. There is no MAC protocol nor any higher layer protocol, just a small engine for generating well-known packets. Hence, it is possible to have fine grained control over timing and content of the generated packets. But more important, by using a MAC entity would have introduced undesired interaction with MAC mechanisms, e.g., packet discarding in case of wrong checksums or illegal MAC header fields.

Clearly, the study has its limitations. It cannot be extrapolated simply to scenarios other than the chosen one, nor to other radio modems. One fundamental reason for this difficulty are the unique properties of wireless links in the 2.4 GHz range, as described in Section 6.1. An example is the phenomenon of multipath fading. In general, the wave propagation environment (number of propagation paths, their respective loss) and its time-varying nature (moving people, moving machines) play a dominant role in constituting channel characteristics. It is far from being obvious or straightforward how wave propagation characteristics or presence of noise / interference translate into error behavior. However, we assume that with the chosen scenario some common characteristics of industrial environments are captured: presence of strong electrical motors, many (sometimes moving) metal surfaces, moving people, and machines switching on and off. Furthermore we assume that, although the quantitative results like mean bit error rates are likely not valid for other environments, the qualitative results (time-varying behavior; presence, burstiness behavior and order of magnitude of packet losses; high variability of error burst lengths) will carry over to similar environments and are important for designing MAC protocols. This assumption is confirmed by the fact that certain qualititative results (regarding packet losses and time variability) were also obtained in a similar study in an industrial environment [52], using a radio modem of a different manufacturer (see Section 6.6).

This chapter is structured as follows. First, in Section 6.1 a brief overview of the physical phenomena occuring in wireless transmission and leading to transmission errors is given. The next four sections are devoted to the link error measurements: Section 6.2 explains the measurement setup, Section 6.3 describes the approach for evaluating the measurements. After this, in Section 6.4 we describe the industrial facility and the environment where the measurements were taken. Finally, in Section 6.5 an overview of the most important measurement results is given.

After reviewing the literature on other bit- and packet-level wireless measurement studies in Section 6.6, stochastic modeling is discussed in Section 6.7. Following a brief overview of some popular stochastic models (Section 6.7.1), an alternative ("bipartite") model is introduced (Section 6.7.2), which is intended to overcome some deficiencies of the popular models. The "performance" of different stochastic models in matching the measurements statistics and in giving predictions of selected performance parameters of an example system is investigated in Section 6.7.3.

The final Section 6.8 summarizes several conclusions drawn from the measurements. Beneath discussing the measurement results themselves (Section 6.8.1), the issue of stochastic modeling is summarized in Sections 6.8.2 and 6.8.3. In the following Section 6.8.4 some implications of the measurement results for the design of MAC and link-layer protocols are reflected.

Some parts of the work presented here can be found in reference [192].

6.1 Sources of Errors

In this section some basic physical phenomena of wireless transmission and how these lead to distorted information reception are briefly reviewed. More thorough presentations can be found in [137], [28], [124], [182], [77], [23], [145]. The main sources of errors are:

- Path loss and attenuation on obstacles, leading to *slow fading* or *shadow fading.*
- Reflexion, diffraction, refraction and scattering, causing transmission on multiple paths, resulting in *fast fading* (or *multipath fading*) and *intersymbol interference.*

- Adjacent channel or cochannel interference.
- Thermal or man-made noise.
- Imperfections of transmitter and receiver.

While thermal noise is present in almost every communication channel, fast fading and slow fading are specific for wireless transmission. Man-made noise in industrial environments can have several sources, e.g., remote controls, motors, or microwave ovens.

It must be noted that many of the physical aspects and the resulting bit error behavior depend on the frequency, the modulation scheme used, and the current environment (e.g., distance, interferers, number of different paths and their respective loss).

6.1.1 Path Loss

For isotropic antennas the path loss can be modeled approximately as (see Equation 2.8 in [182])

$$P_R = P_T \cdot g_T \cdot g_R \cdot \left(\frac{\lambda}{4\pi}\right)^2 \cdot \frac{1}{d^\gamma}$$

where g_R and g_T are the antenna gains of receiver and transmitter, P_R and P_T are the power levels at receiver and transmitter, λ is the wavelength, d is the distance between transmitter and receiver, and γ varies between 2 (free space wave propagation) and 5 (strong attenuation, e.g., due to obstacles). Some typical values for γ are quoted from [145] in Table 6.1. While the details of this equation vary with the propagation environment and the antenna technology, the qualitative behavior remains the same: the path loss is at best quadratic in the distance between transmitter and receiver.[1] The path loss can be shown to be a source of bit corruption even over short distances like in wireless LANs (see Section 6.6). Furthermore, since wireless receivers typically require the signal strength to be above some threshold value, cell bounds can be established this way [39]. Unfortunately, most of the path loss models available (e.g., the Okumura/Hata model [63]) are targeted for larger distances of 1 - 10 kilometres.

The notion of *slow fading* refers to significant changes in the mean value of the received signal strength, as they occur due to significant changes in distance between transmitter and receiver, or by moving through tunnels or beyond large obstacles. Slow fading phenomena usually occur on longer timescales, they often coincide with human activity (e.g., mobility). For short durations in the range of a few seconds the channel can be assumed to have constant path loss. According to [126] in certain situations the value of the receiver power level P_R fluctuates according to a lognormal distribution about its mean value.

6.1.2 Multipath Fading

Signals transmitted in the 2.4 GHz frequency band are subject to reflection, diffraction, refraction, and scattering. An immediate result is that a signal may travel on multiple different paths from transmitter to receiver (see Figure 6.1). Since these paths usually have different lengths, multiple

[1] For indoor scenarios sometimes path loss exponents $\gamma < 2$ were observed, see Table 6.1. Likely these are due to constructive interference generated by the presence of multiple paths with only small delay difference.

Environment	γ
Free-Space	2
Urban area cellular radio	2.7 - 4
Shadowed urban cellular radio	5-6
In-building Line of Sight	1.6-1.8
Obstructed In-building	4-6
Obstructed in factories	2-3

Table 6.1: Path loss exponents for different environments

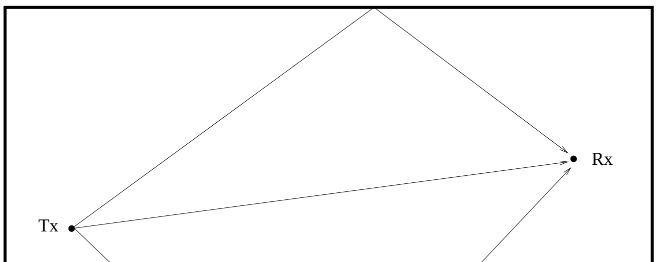

Figure 6.1: Multipath fading

copies of the same signal with different phase angles overlap at the receiver (delay spread). This has two consequences:

- The resulting signal can be amplified or attenuated (constructive or destructive interference), depending on the relative phase shift (signal strength variation). This *fast fading* may lead to loss of received power of up to 40 dB.
- The delay spread leads to intersymbol interference, since signals belonging to different information symbols may arrive at the same time.

If the stations move relative to each other, the number of paths and their phase shifts vary in time, thus giving a fast fluctuating signal strength at the receiver, however, with nearly constant mean value on short timescales. The mean value may vary on longer timescales due to changes in distance or moving beyond obstacles, both leading to slow fading.

If the delay spread is small relative to the duration of a channel symbol, the channel is called *non-frequency selective* or *flat*, otherwise it is called *frequency-selective*.

In general it is a hard task to predict the number of paths and their relative strength, since accurate information about the environment would be needed, including all objects, their material, trajectories of moving people, and so forth. So usually one resorts to stochastic models, where the number of paths, their phase shifts and relative strengths at the receiver are modeled as random variables and the resulting signal strength/signal phase pair at the receiver as a complex-valued random process $\{r(t)\}_{t\in\mathbb{R}}$. This random process is likely to show some correlation, since for continuous waveforms the sum signal at the receiver is continuous, at least for time intervals where the number of paths does not change. For the design of coding and modulation schemes, the following properties of $\{r(t)\}_{t\in\mathbb{R}}$ are of interest: the distributions of the phase $\Psi(t)$ and amplitude $A(t)$ corresponding to $r(t)$, its *level crossing rate*, and the *fade duration*. These are important since most wireless receivers require $A(t)$ to be above some threshold value A_{min} in order to be able to successfully detect and decode a signal. For example, the fade duration can be helpful in designing interleavers. The level crossing rate is defined as the rate with which the stochastic process $\{|r(t)|\}_{t\in\mathbb{R}}$ takes some fixed value r_0 with negative slope (hence, entering a deep fade). The fade duration is defined as the duration that the process is below some fixed value r_0. A more thorough discussion about the computation of these values from the signal strength process can be found in [147], [148] and [78]. According to [138], there are no general expressions for the probability distribution of the fade durations available, not even for the popular cases of *Rayleigh fading* and *Rice fading*, which are summarized below:

- Rayleigh fading: in this case a large number of paths of nearly equal signal strength between transmitter and receiver is assumed. Using the central limit theorem [47, chap. 10], the resulting signal strength at the receiver can be shown to be a Rayleigh process and the resulting amplitude is Rayleigh distributed. This model is often assumed for macrocellular environments without a LOS path [145].
- Rice fading: as opposed to the Rayleigh fading case, a dominant signal path is present, e.g., caused by a LOS path. Here the resulting signal strength is a Rice process and the amplitude is Rice-distributed. The probability of having a LOS component increases with smaller cell diameter [28, p. 34]. For this reason Rice fading is the more appropriate model for LAN environments.

Some stochastic bit error models are built from the behavior of the fading process, e.g., the Wang-/Moayeri model [184], which uses a Rayleigh fading assumption (see below).

6.2 Measurement Setup

In this section we give a brief overview on the measurement equipment. A more detailed discussion of the setup can be found in [196] and [197].

6.2.1 IEEE 802.11 / PRISM I PHY

The IEEE 802.11 DSSS PHY is described in Section 3.2 in some detail. For building the measurement setup we have used a MACless radio modem (based on the Harris/Intersil PRISM I chipset [2]), which is compliant with the IEEE 802.11 DSSS PHY. It offers the following modulation types/bitrates: 1 MBit/s with differential binary phase shift keying (DBPSK), 2 MBit/s with differential quaternary phase shift keying (DQPSK), 5.5 MBit/s with binary m-ary bi-orthogonal keying (BMBOK), 5.5 MBit/s with complementary code keying (CCK), 11 MBit/s with quaternary m-ary bi-orthogonal keying (QMBOK) and 11 MBit/s with CCK. The CCK modes are compliant to IEEE 802.11, the BMBOK and QMBOK modes are only present for compatibility reasons and were not used. It is possible to attach two antennas to the modem and to use receiver diversity (i.e. the receiver selects the antenna with the maximum signal level). The transmitter power was fixed at 18 dBm, corresponding to 63 mWatt. The radio modem basically consists of high frequency circuitry and a baseband processor. The latter accepts and delivers a serial bit stream from upper layers, optionally performs scrambling (employing a shift register with feedback), performs DSSS processing, and generates / receives PHY packets [73]. The characteristics of the serial bit stream is the focus of interest.

The PHY packet format of the PRISM chipset is exactly the same as prescribed by the 802.11 standard for the DSSS PHY (Figure 3.3 and Section 3.2). If the PLCP header checksum is wrong or the signal field carries an unknown value, the whole packet is discarded by the baseband processor.

6.2.2 Measurement Setup

We used two dedicated stations, a *transmitter station* and a *receiver station*, which do not change their roles during a measurement. The setup is sketched in Figure 6.2. The basic idea is that the transmitter station sends a well-known *packet stream* over the wireless link, which is captured and stored by the receiver station into a logfile. For generation and reception of the packets we have used a microcontroller board carrying the radio modem and a separate processor (Motorola PowerQUICC [111] with Tundra PCI Interface [171] and a 50 MHz PowerPC 603e processor). The coupling to the (Windows NT-based) host is achieved with a segment of 64 kByte shared memory, denoted as *host interface*. We call this board a *wireless network interface card (NIC)*. The wireless NIC contains a specific measurement application and neither MAC functionality nor any higher layer protocols. This way we have fine grained control over the packet generation and reception process and no bias is introduced by upper layer protocols. Specifically, using a MAC entity would have introduced undesired interaction with MAC mechanisms, e.g., packet discarding in case of wrong checksums or illegal MAC header fields, and might also have an unwanted influence on the packet's sending time. Especially

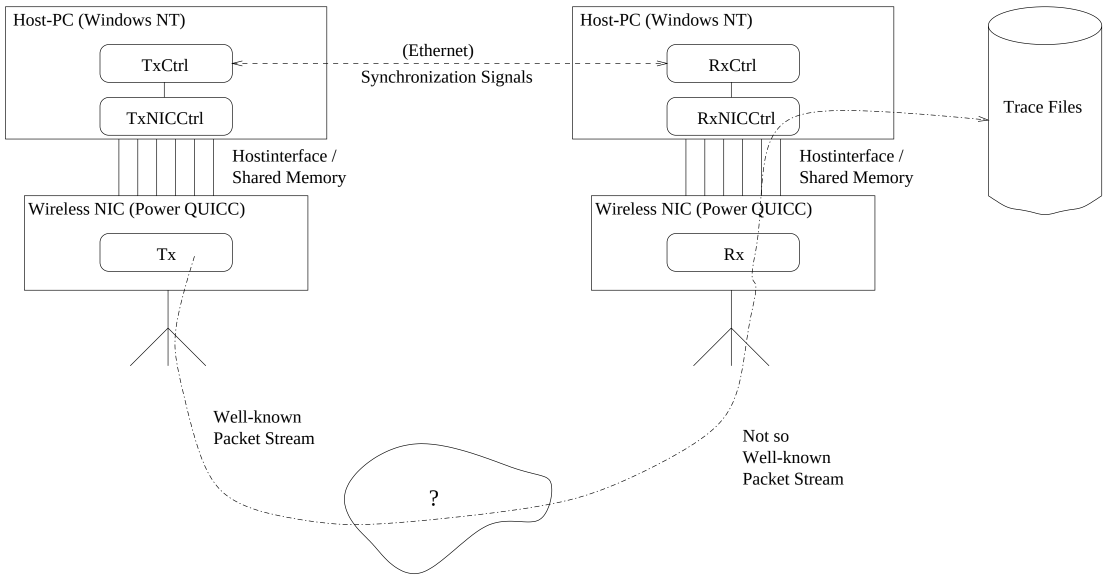

Figure 6.2: Measurement setup

the 802.11 MAC with its carrier sensing mechanism tries to avoid interferences, while for measuring purposes it can be sometimes of interest to see their influence.

We briefly discuss the different software modules of our measurement setup, see Figure 6.2.

- The *Tx module* is located on the wireless NIC of the transmitter station. It accepts configuration commands from the TxNICCtrl module discussed below (allowing to set the variable parameters), and generates a well-known *packet stream*, see Section 6.2.3.
- The *Rx module* is also located on the wireless NIC. Its main task is to capture packets from the wireless link, to add metainformation (e.g., timestamps, packet size, reception status, and signal strength) and to pass them to the host via the host interface (which puts them in a logfile). The resulting stream of received packets is called a *trace*.
- The *TxNICCtrl module* and *RxNICCtrl module* are wrappers which offer a command line interface to the Tx module and Rx module.
- The *TxCtrl module* is a script which synchronizes itself with the RxCtrl software for controlling the measurements (using a TCP connection over the Ethernet).
- The *RxCtrl module* is actually controlling a whole measurement. It loops over all desired values of variable parameters; for each combination of parameters a packet stream is started (by triggering the TxCtrl module) and the trace is logged onto the harddisk.

The evaluation of the traces is done off-line, employing several Perl scripts.

Our setup enables variation of several parameters, which are related both to the properties of the radio modem and packet-stream generation. The important modem-related parameters are shown in Table 6.2 and the set of packet-stream-related parameters is shown in Table 6.3.

Parameter	**Description**
ScramblingEnabled	Determines whether scrambling is used
DiversityEnabled	Determines whether receiver antenna diversity is used
PreambleLength	Number of bits for PHY preamble
ModulationCode	Distinguishes modulation used for data portion: 1 MBit/s BPSK, 2 MBit/s QPSK, 5.5 MBit/s CCK, 5.5 MBit/s BMBOK, 11 MBit/s CCK, 11 MBit/s QMBOK

Table 6.2: Adjustable radio parameters

Parameter	**Description**
NumPackets	Number of Packets
GapTime	Time gap between two packets
NumChunks	Number of chunks per packet, Packet length = *NumChunks* times 288 bits

Table 6.3: Adjustable packet stream parameters

Our setup was tested in laboratory measurements and in other measurement campaigns [59] in controlled environments and works fine. In several traces, either bit errors, packet losses, or other packet-related phenomena occured at all, or it was possible to relate the observed phenomena to environmental conditions.

6.2.3 Format of the Generated Packet-Stream

The transmitter station generates a *packet stream*. What the receiver captures after its activation is called a *trace*. If no errors occur, the trace is the same as the packet stream. The format of the packet stream was chosen such that

- the number of 0's and 1's are equal
- long runs of 0's or 1's are avoided
- it suffices to have a fraction of the packet (denoted as a *chunk*) correctly received in order to determine which packet it originally was.

Especially the last property enables bit-by-bit comparison of a received packet with the transmitted packet.

The generated packet stream consists of a prescribed number of *packets* (according to the *NumPackets* parameter), which are transmitted at equidistant start times, all packets having the same values for all parameters, including packet size. The data part of a packet consists of an integral number of *chunks*. For generating a chunk, every bit of a 32 bit sequence number is mapped to eight bits (with $0 \mapsto 11000011$ and $1 \mapsto 00111100$), making up 256 bits. Additionally, header (`0xffff`) and trailer (`0x0000`) are generated, thus a chunk has an overall size of 288 bits. The sequence numbers are incremented from chunk to chunk. For example, with *NumChunks* = 3 chunks per packet, the first packet of a packet stream carries sequence numbers 0, 1, and 2, the second packet 3, 4, and 5 and so forth.

It is immediately clear that this format has the same number of zero's and one's and that long runs of either value are avoided. A received sequence of 288 bits length is considered as a correctly received chunk, if header and trailer match their specified values and if all bytes in between are either `0xC3` = 11000011 or `0x3C` = 00111100.

6.3 Measurement Evaluation Methodology

Much of the evaluation of the measurements uses the notion of *indicator sequences* or the more specific *binary indicator sequences*. First the according definitions are given, then the use of such sequences in measurement evaluation is described.

6.3.1 Indicator Sequences

In general, an indicator sequence is a finite sequence of natural numbers, the numbers in a binary indicator sequence are restricted to the values zero and one. As a convention, in binary indicator sequences we associate with a 1 an error event (e.g., an erroneous bit or a lost packet) and with a 0 the correct event. A binary indicator sequence can be viewed as finite subset of a sample path of a random process $\{B_n\}_{n\in\mathbb{N}}$, where each B_i is a Bernoulli random variable.

Binary indicator sequences are subdivided into *error bursts* and *error-free bursts* according to a *burst order* k_0. We define an error-free burst of order k_0 to be a maximum-length contiguous all-zero subsequence with a length of at least $k_0 + 1$. In contrast, an error burst of order k_0 is a subsequence of at least one bit length and with ones at its fringes. Furthermore, within an error burst at most $k_0 - 1$ consecutive zeros are allowed.

By this definition a binary indicator sequence $i_1 i_2 \dots i_m$ of m values length is segmented into p alternating error bursts and error-free bursts (these definitions are similar to those used in [91]). The length of the j-th error-free burst is denoted as X_j, the length of the j-th error burst is denoted as Y_j, and Z_j is the actual number of ones occuring in the j-th error burst. We can form the *burst length sequence*[2]:

$$X_1, Y_1, Z_1 \quad X_2, Y_2, Z_2 \quad \dots \quad X_p, Y_p, Z_p$$

Let us denote the sequence $X_1 X_2 \dots X_p$ as the *error-free burst length sequence*, $Y_1 Y_2 \dots Y_p$ as the *error burst length sequence* and $\frac{Z_1}{Y_1}\frac{Z_2}{Y_2}\dots\frac{Z_p}{Y_p}$ as the *error density sequence.* As an example, take the binary indicator sequence 0010010100011000110000. With burst orders of $k_0 = 1$ and $k_0 = 2$ we get the burst length sequences

$$\begin{array}{llllllll} k_0 = 1: & 2,1,1 & 2,1,1 & 1,1,1 & 3,2,2 & 3,2,2 & 5,0,0; \\ k_0 = 2: & 2,1,1 & 2,3,2 & 3,2,2 & 3,2,2 & 5,0,0. \end{array}$$

It is important to note that with only recording the number Z_j of errors within error burst Y_j we loose information about the exact error positions.

[2]We will write $X_j = 0$ or $Y_j = 0$ to denote the absence of a burst at the fringes of a binary indicator sequence. Furthermore, the notation does not explicitly indicate indicate the dependence on k_0.

Using the notion of burst length sequences, some simple statistics can be computed, e.g., the mean error rate $\bar{e}$ or the mean error burst length $\bar{Y}$

$$\bar{e} = \frac{\sum_{j=1}^{p} Z_j}{\sum_{j=1}^{p}(X_j + Y_j)}, \qquad \bar{Y} = \frac{1}{p}\sum_{j=1}^{p} Y_j.$$

Accordingly, some other simple first order statistics [variance, coefficient of variation (CoV)] can also be computed for the burst length sequence.

Taking a binary indicator sequence $i_1 i_2 \dots i_m$ as a sequence of identically distributed Bernoulli random variables, the conditional probability $\Pr[i_{n+k} = 1 | i_n = 1]$ for $1 \leq k \leq m - n$ is of some interest. It is approximated as follows (frequency-based approach):

$$\begin{aligned} \Pr[i_{n+k} = 1 | i_n = 1] &\approx \frac{\#\text{cases with } i_n = 1 \text{ and } i_{n+k} = 1}{\#\text{cases with } i_n = 1} \\ &= \frac{\sum_{j=1}^{m-k} i_j \cdot i_{j+k}}{\sum_{j=1}^{p} Z_j}. \end{aligned}$$

This conditional probability is related to the correlation function of the binary indicator sequence, since, with the assumption of equally distributed i_k (with mean $\bar{e}$ and variance $\sigma^2 = \bar{e}(1-\bar{e})$) we have:

$$\begin{aligned} \operatorname{Corr}[i_n, i_{n+k}] &= \frac{\operatorname{Cov}[i_n, i_{n+k}]}{\sqrt{\sigma^2 \sigma^2}} \\ &= \frac{\operatorname{E}[i_n i_{n+k}] - \operatorname{E}[i_n]\operatorname{E}[i_{n+k}]}{\bar{e}(1-\bar{e})} \\ &= \frac{\operatorname{E}[i_n i_{n+k}] - \bar{e}^2}{\bar{e}(1-\bar{e})} \\ &= \frac{\Pr[i_{n+k} = 1 | i_n = 1]\bar{e} - \bar{e}^2}{\bar{e}(1-\bar{e})} \\ &\approx \Pr[i_{n+k} = 1 | i_n = 1] \end{aligned}$$

where the approximation holds for small $\bar{e}$ values. Here we have used that

$$\begin{aligned} \operatorname{E}[i_n i_{n+k}] &= \sum_{x,y \in \{0,1\}} xy \Pr[i_n = x, i_{n+k} = y] \\ &= \Pr[i_n = 1, i_{n+k} = 1] \\ &= \Pr[i_{n+k} = 1 | i_n = 1] \cdot \Pr[i_n = 1] \\ &= \Pr[i_{n+k} = 1 | i_n = 1] \cdot \bar{e} \end{aligned}$$

A more in-depth treatment of binary indicator sequences can be found in [85].

6.3.2 Trace Evaluation

Given a single trace, the focus of interest is on the packet losses and on the bit error behaviour of the packets actually received. Both are expressed as binary indicator sequences. In a preprocessing step other packet impairments (e.g., ghost packets, truncated packets, bit-shifted packets) are identified and the corresponding packets are marked as lost packets (more details and a justification are given in Section 6.5.1).

The *packet loss indicator sequence (PLIS)* of a single trace is constructed by marking lost packets with a 1 and received packets with a 0. This sequence only displays lost packets while ignoring bit errors: received packets of correct length but with bit errors are marked with a 0.

The *bit error indicator sequence (BEIS)* of a single trace is constructed by XORing every received (i.e., possibly erroneous) packet r_i with its corresponding transmitted (error-free) packet t_i ($i \in \{1, \ldots, NumPackets\}$):

$$\nu_i = r_i \quad \texttt{XOR} \quad t_i$$

The results $\nu_1 \ldots \nu_{NumPackets}$ then are simply concatenated in the order of increasing packet numbers. In the BEIS any information about packet boundaries, lost packets, or packet gap times is completely ignored.

The BEIS can be seen as the available input of a MAC protocol or a coding scheme.

While the PLIS is typically analyzed with $k_0 = 1$, for the BEIS several values of k_0 were used to get more insight into the burst structure.

6.4 Measurement Parameters and Environment

To avoid confusion, we will use the following definitions: a *measurement campaign* consists of one or more *measurements*.

The set of tunable parameters is given in Tables 6.2 and 6.3. For a single measurement, a subset of these parameters is kept fixed, while the remaining parameters are variable. Furthermore, a suitable range of values for the variable parameters must be chosen. Two measurements are distinguished by their choice of variable parameters and parameter ranges. Within a measurement, for each parameter setting a packet stream is generated. Hence, within a packet stream all packets have the same parameters and are transmitted at equidistant times.

We have conducted two measurement campaigns in an industrial environment, namely at the Produktionstechnisches Zentrum (PTZ) in Berlin, Germany. The PTZ is a research facility for machinery engineering, supported by industry and academia. The first campaign was performed on June 26, 2000 and its main purpose was to evaluate our measurement setup and to find out which phenomena are important [196]. The second campaign took place from Aug. 28 to Aug. 30, 2000 [197]. The focus here is on the second campaign.

6.4.1 Environment

The PTZ owns a large factory building which contains several machines of different types and with people walking around all the time. The ground plan of the building has the shape of a circle. At the fringe of the circle is a path which can be used by small vehicles, while the inner circle contains the machinery. During both campaigns we have chosen the same positions for placing our setup within the building. The choice came from asking the PTZ people where they would place both stations (for a discussion of whether the chosen position was a "best case" or "worst case" position see Section 6.8.1). In Figure 6.3, we show the relative position of our measurement equipment in the factory building, while in Figure 6.4 we show the close neighbourhood, especially the machines that are in close proximity. We have investigated a NLOS scenario, with a closet in between the transmitter and

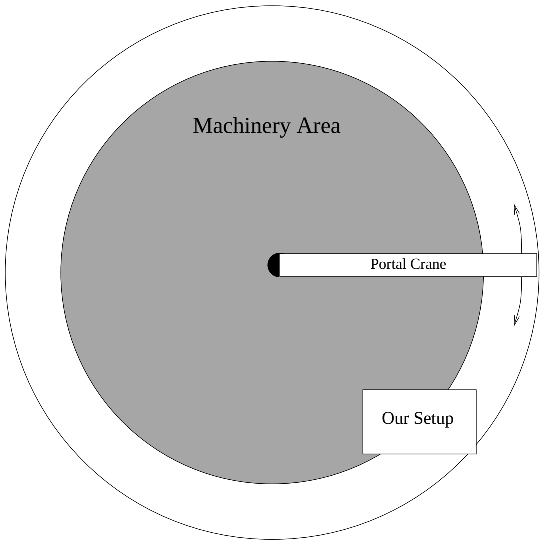

Figure 6.3: Position of our setup within the building

receiver station, and the die sinking electrical discharge machine (EDM) working area very close to the direct line. Both stations are ≈ 7-8 meters apart and stationary during the measurement campaigns. The receiver station was in close proximity (≈ 1 m) to a cabinet containing the power supply for a huge 5 axis milling machine, which, however, was not operating during the second campaign. The die sinking EDM was active most of the time, except when changing the workpiece. A second EDM machine was located behind the first one (see Figure 6.4). It was used by PTZ staff almost all the time. At the ceiling, in a height of ≈ 8 meters, was a portal crane, capable of moving around 20 tons. Its motors are placed at the fringe's end of the portal crane. The crane was used during the first two days of the second campaign.

Instead of investigating different scenarios with a restricted set of measurements, we have chosen to focus on the single scenario described above. This concentration allowed us to get a more in-depth insight into different aspects of wireless transmission (e.g., long-term behavior). The reason for choosing a NLOS scenario is that the measurement results should help in the design of MAC protocols and coding schemes for industrial WLANs, where the response to bad channel conditions is of particular interest for assessing protocol's reliability.

6.4.2 Parameters

The second measurement campaign was designed to assess: a) the packet loss and packet impairment behavior on short and long timescales, b) the long-term bit error rate behavior, c) the dependency of the bit error behavior on packet sizes and modulation types, and d) the dependency of packet losses and impairments and bit error behavior on the scrambling mode. Furthermore, no interferers in the same frequency band (e.g., IEEE 802.11 LANs) were present.

We have performed three different measurements within the second campaign: the **longterm1** measurement is a long-term measurement performed with a single modulation type and packet size, only

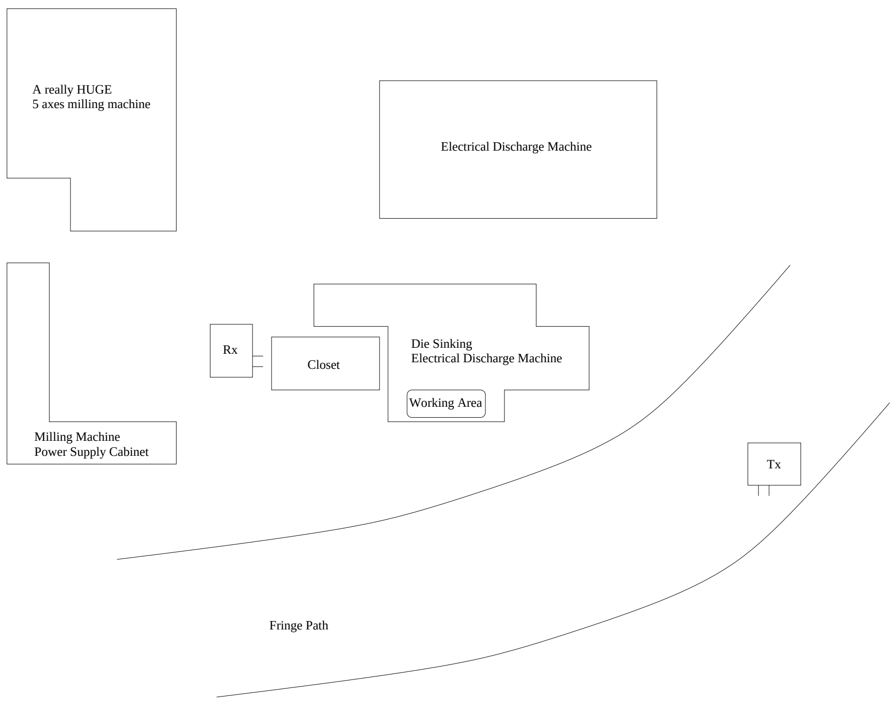

Figure 6.4: Setup of PTZ measurement

varying the scrambling mode (addressing a), b), and d)). The **longterm2** measurement is the same as the **longterm1** measurement, however, another pair of PRISM radio modems was used. In the **factorial** measurement we have varied the scrambling mode, modulation type and packet sizes (thus addressing c)), and for each combination of parameters the short-term bit error behavior was investigated. The main purpose of the **longterm2** measurement was to confirm that the observed phenomena are not due to the particular pair of radio modems used. Indeed, our results confirm this belief and allow us to restrict the discussion to the results obtained with the first pair of radio modems (the **longterm1** measurement and **factorial** measurement) [194].

We have chosen for the **longterm1** and **longterm2** measurements to keep all parameters fixed, except the scrambling mode (on, off) and the pair of radio modems used (see Table 6.4). In both measurements we have taken 90 traces for every scrambling mode. With 90 traces, 2 hours and 10 minutes are covered. Within a measurement the traces are numbered consecutively, thus increasing trace numbers corresponds to increasing time. Within the **longterm1** measurement the first 90 traces are taken without scrambling, the other 90 traces with scrambling. For the **factorial** measurement we have chosen a full factorial design [75, chap. 16], varying the modulation type, packet size, and the scrambling mode as summarized in Table 6.6, while keeping the other parameters fixed (Table 6.5). For every combination of parameters two traces were taken. Depending on modulation type and packet size the trace duration is between ≈ 30 seconds and 1000 seconds. Traces 1 to 56 are taken without scrambling, the traces 57 to 112 with scrambling. Within each of the two groups we have varied the modulation scheme from low bitrates to high bitrates and for each modulation scheme we have varied the packet sizes from small packets to large packets.

Parameter	**Value**
PreambleLength	128 bits
DiversityEnabled	True
Frequency/Channel	12
NumPackets	20000
NumChunks	14 (504 bytes)
GapTime	1000 μs
ModulationCode	2 MBit/s QPSK

Table 6.4: Fixed parameters for **longterm1** and **longterm2** measurements

Parameter	**Value**
PreambleLength	128 bits
DiversityEnabled	True
Frequency	12
NumPackets	20000
GapTime	1000 μs

Table 6.5: Fixed parameters for **factorial** measurement

Parameter	**Value**
ScramblingEnabled	True, False
ModulationCode	1 MBit/s BPSK, 2 MBit/s QPSK, 5.5 MBit/s CCK, 11 MBit/s CCK
NumChunks	3, 9, 14, 28, 56, 112, 167 (corresponding to 108, 324, 504, 1008, 2016, 4032, 6012 bytes)

Table 6.6: Variable parameters for **factorial** measurements

6.5 Measurement Results

In this section we discuss the most important measurement results. A more in-depth presentation can be found in [192], and [197], which in turn relies on data presented in [194], [195], and [193].

6.5.1 Packet-Related Phenomena

There are three major types of transmission errors (compare the PHY PPDU format described in Section 3.2): Failure to acquire bit synchronization or to properly detect the start frame delimiter, an error in the header fields (e.g., wrong value in signal field or CRC error), and bit errors in the packet's data part.

Failing to acquire bit synchronization leads to *packet losses*, discussed in Section 6.5.2. An error in the header fields leads to other packet-related phenomena (*ghost packets*, *missized packets*), which are discussed in more detail in [197, chap. 2 and 5].

It is possible to distinguish between getting no bit synchronization and the other packet-related phenomena, since the baseband processor generates an interrupt when it has acquired bit synchronization and detected the SFD field. For lost packets this interrupt is missing. Therefore, we can conclude that packet losses are due to not acquiring bit synchronization.

An interesting phenomenon are packets with *bit shifts*. In this case, the baseband processor delivers the correct packet length and an appropriate number of bits. However, somewhere in the packet's data (typically at the beginning of a packets data part, but this is not a general rule) some random bits are inserted into or deleted from the bit stream, and a corresponding number of bits is truncated or added to the packets end. As a result, the following bit sequence is a left- or right-shifted version of the original sequence. We have no validated explanation for this phenomenon, but after several personal discussions with independent experts in communication systems the most educated guess is that it is due to the bit synchronization algorithm used in the receiver. The "bit shift patterns" change when the scrambling mode or the environment is varied (see [197], [59]), hence, bit-shifts are influenced by the environment. Furthermore, in other measurements no bit-shifted packets were observed over long time-spans [59]. Both circumstances do not fit together with the assumption of a (permanently) faulty setup, hence, we assume our setup being ok.

For every trace of the **longterm1** measurement the fate of every of its packets was determined, and the "fate sequences" of all traces were concatenated in order of increasing trace numbers. The possible fates are: packet ok (this does not imply the absence of bit errors), packet lost, bit-shifted, missized, ghost packet. Then we have investigated bursts of subsequent packets having the same fate. In Table 6.7 some simple statistics of these bursts are summarized. There are several important points. First, the rates of bit-shifted, missized (truncated and oversized) or ghost packets are negligible. Second, the phenomena of truncated, oversized, and ghost packets tend to occur in single packet bursts, as indicated by their small variations and means. The occurence of bit-shifted packets is slightly more bursty. The overall packet loss rate (PLR) is $\approx 6.3\%$, which is non-negligible and needs to be considered in accurate channel models.

Ghost packets, bit-shifted packets and missized packets are treated as lost packets. This is reasonable, since their rates are low and they have the tendency to occur paired with lost packets [197, chap. 5.1].

	% packets	BL: mean	BL: CoV	BL: max
packet ok	93.5144	51.1254	20.0712	101158
packet loss	6.2789	3.6173	17.2987	14936
bit shifted	0.0324	1.5116	0.8980	20
truncated	0.0383	1.0430	0.1946	2
oversized	0.0197	1.0456	0.2064	3
ghost packet	0.1160	1.0278	0.1880	4

Table 6.7: First order statistics of compound packet fate sequence of **longterm1** measurement (BL: burst length)

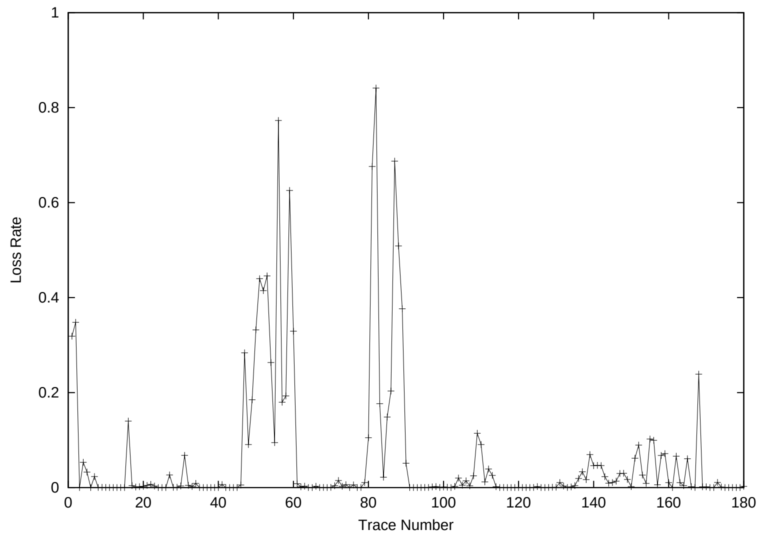

Figure 6.5: Packet loss rate vs. trace number for **longterm1** measurement

6.5.2 Packet Losses

In this section we discuss the packet loss behavior found in the **longterm1** measurement, as manifested in the corresponding packet loss indicator sequence (PLIS), see Section 6.3.2. Instead of "error bursts" for the PLIS we will use the term "packet loss bursts".

For the receiver station a lost packet is indistinguishable from the case that no packet was sent at all. Therefore, for detecting lost packets the timestamps of the packets in the logfile were used (these are generated by the receiver station upon finishing packet reception), together with the property of equidistant packet start times. The difference of subsequent timestamps is compared with the *InterpacketTime* (given as the sum of the *GapTime*, the time needed for the fixed length header (see Figure 3.3), and the known duration of the data part). From this comparison the number of lost packets is easily computed.

The packet loss rates are time-varying. To show this, in Figure 6.5 the PLR of individual traces vs. the trace number for the **longterm1** measurement is presented; it should be noted that this figure spans more than four hours. The PLRs are sometimes very high (more than 80%) and strongly

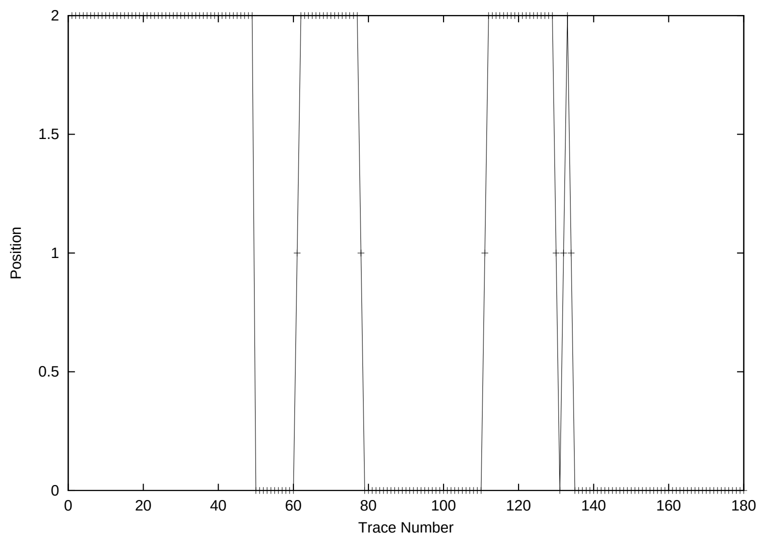

Figure 6.6: Position of portal crane (0=close proximity, 1=short distance, 2=longer distance) for **longterm1**-measurement

varying. A possible explanation offers Figure 6.6, where the “portal crane function” for the **longterm1** measurement is shown. This function displays the distance of the portal crane to our setup (0 = directly above the setup, 1 = no more than five meters away, 2 = more than five meters away). It can be seen that, except for a peak at traces one and two, the PLRs have the highest rates and the highest degree of fluctuation when the portal crane is close to the setup. During the **factorial** measurement the portal crane was not active and the PLR was always below 10%.

Next we consider the “burstiness” of packet losses. To get summary information, we have formed the *compound packet loss indicator sequence (COMP-PLIS)* by concatenating the PLIS of all **longterm1** traces in order of increasing trace number. The COMP-PLIS is a binary indicator sequence. It is analyzed with burst order $k_0 = 1$ (only consecutive packet losses belong to the same packet loss burst).

Fraction of Received Packets	93.5144%
Fraction of Lost Packets	6.4855%
Received Packets: Mean BL	51.1254
Lost Packets: Mean BL	3.5457
Received Packets: CoV BL	20.0712
Lost Packets: CoV BL	17.1956
Received Packets: Max. BL	101158
Lost Packets: Max. BL	14936
Pr[Packet $n+1$ lost|Packet n lost]	0.7179
Pr[Packet $n+1$ received|Packet n received]	0.9804

Table 6.8: First order statistics of burst lengths (BL) of the **longterm1** measurement (COMP-PLIS, $k_0 = 1$)

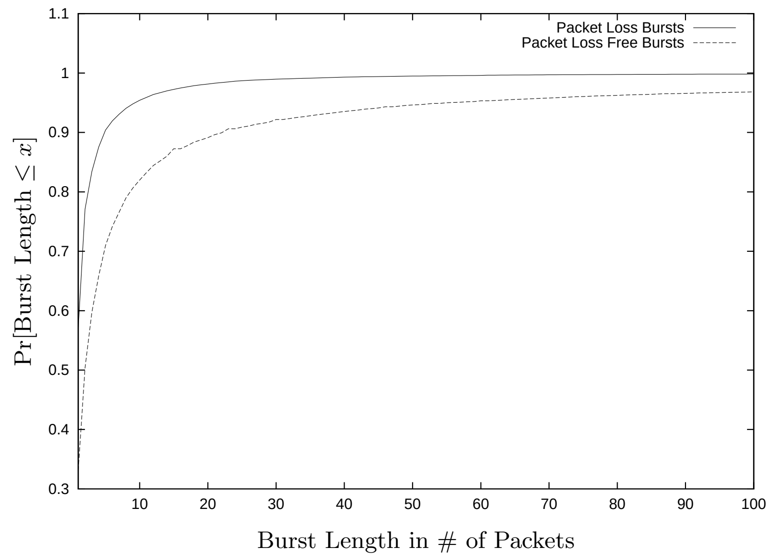

Figure 6.7: Cumulative distribution functions of packet loss burst lengths and packet loss-free burst lengths (for COMP-PLIS)

	factorial	**longterm1**	**longterm2**
w scrambling	9885	34755	26392
w/o scrambling	20411	191456	45159

Table 6.9: Number of lost packets

The main statistics of the packet-loss-burst lengths and packet-loss-free burst lengths are summarized in Table 6.8, and their respective distribution functions are shown in Figure 6.7. The overall packet loss rate (PLR) is $\approx 6.4\%$ and thus non-negligible. The packet-loss bursts are typically short ($\approx 95\%$ of all bursts last ten packets or less), but their lengths are highly variable, and very long bursts can be observed (long tailed distribution). The packet-loss-free burst length distribution is even more variable, has a higher mean value and a longer tail, which fortunately leads to long periods of no packet losses. Another view of the burstiness of packet losses is given by the conditional probability that packet $n+k$ is lost given that packet n is lost, shown in Figure 6.8: it decays monotonically from ≈ 0.71 for $k = 1$ to ≈ 0.44 for $k = 2000$. Hence, by the approximation given in Section 6.3.1, packet losses are strongly correlated over several hundreds of packets. In the uncorrelated case the conditional probability would be equal to the packet loss rate of $\approx 6.3\%$ for all k.

Let $X_1 \ldots X_p$ and $Y_1 \ldots Y_p$ be the corresponding packet-loss-free burst lengths and packet-loss burst lengths. In order to find out whether the burst length sequences $X_1X_2 \ldots X_p$ and $Y_1Y_2 \ldots Y_p$ show correlation, their autocovariance functions $R_X(k)$ and $R_Y(k)$ were computed, using a standard approximation formula [18]:

$$R_X(k) \approx r_k = \frac{c_k}{c_0} \quad (k \in \{1, \ldots, p-1\})$$

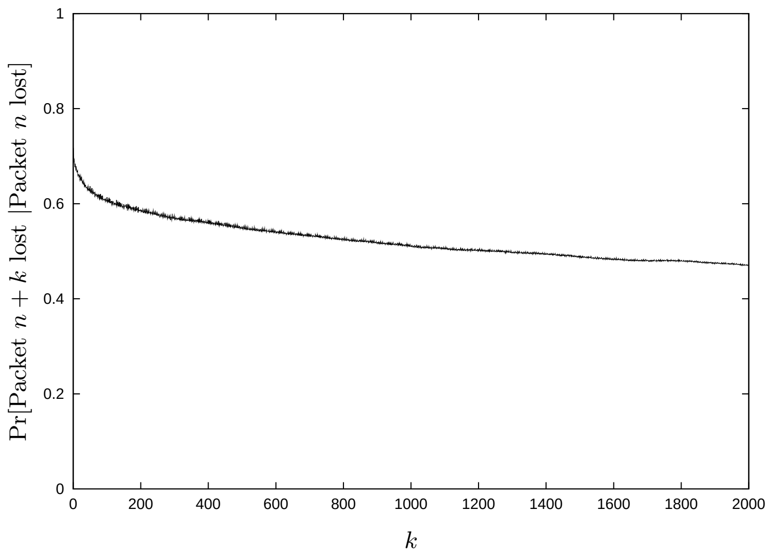

Figure 6.8: Conditional probability that packet $n + k$ is lost given that packet n is lost (for COMPPLIS)

where

$$c_k = \frac{1}{p}\sum_{t=1}^{p-k}(X_t - \bar{X})(X_{t+k} - \bar{X})$$

The rule of thumb is that if for some $k \geq 1$ the absolute value of r_k exceeds 0.2 then there is more than weak correlation and thus the X_i cannot be independent. In Figure 6.10, the autocovariance function for the packet loss-free burst length sequence $X_1 \dots X_p$ is shown, in Figure 6.9, the same is displayed for $Y_1 \dots Y_p$. The conclusion is that the packet loss burst lengths are uncorrelated (and thus can be modeled as independent), while for the packet loss-free burst lengths there is more than weak correlation on the first five lags, then correlation gets weak.

A surprising observation is documented in Table 6.9, which shows the overall number of lost packets for the different measurements with and without scrambling. Packet losses occur significantly more often if scrambling is disabled. Furthermore, not shown here, with scrambling the packet loss bursts are typically shorter than without scrambling. We have not found any clear dependency of packet losses on the modulation scheme used.

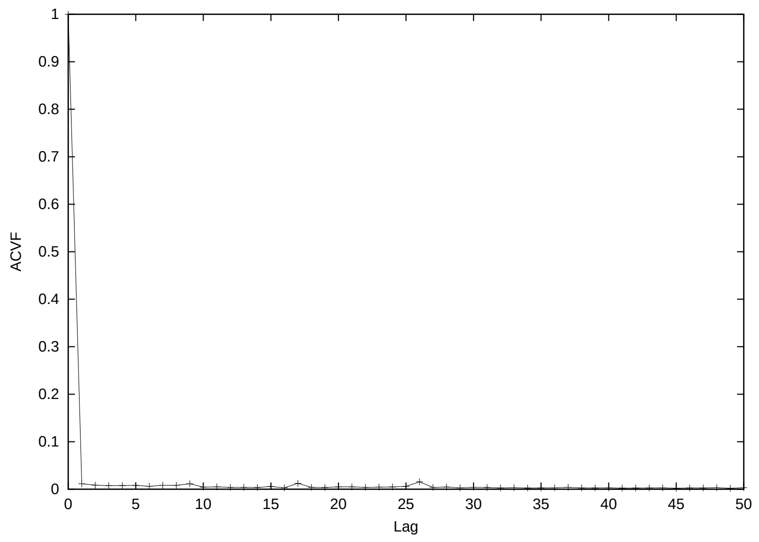

Figure 6.9: Autocovariance function of packet loss burst lengths for the compound loss sequence and $k_0 = 1$ for **longterm1** measurement (for COMP-PLIS)

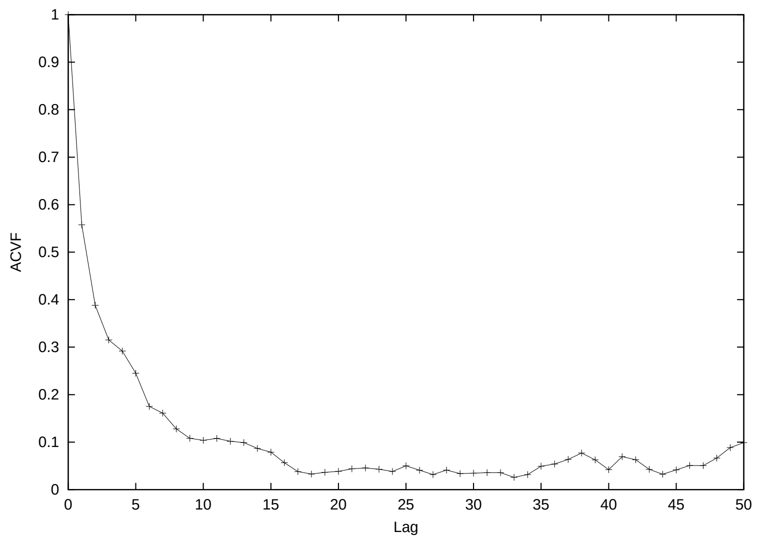

Figure 6.10: Autocovariance function of packet loss-free burst lengths for the compound loss sequence and $k_0 = 1$ for **longterm1** measurement (for COMP-PLIS)

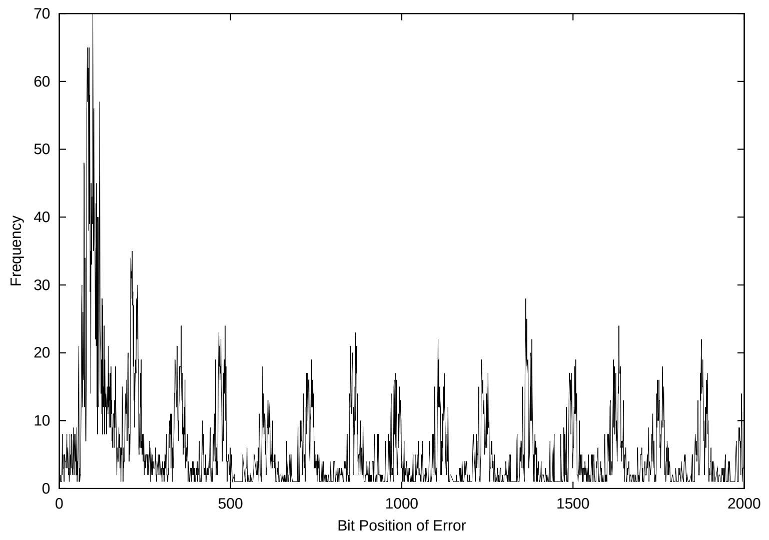

Figure 6.11: Positions of bit errors, **factorial** trace 83 (QPSK modulation, with scrambling, 6012 bytes packet size)

6.5.3 Positions of Bit Errors

Bit errors do not occur in all positions of a packet's data part with equal probability. This is exemplarily shown in Figure 6.11 for a QPSK trace (**factorial** measurement) and in Figure 6.12 for a 5.5 MBit/s CCK trace, where for the first 2000 bit positions within a packet the number of bit errors occuring at this position during a trace is displayed. These figures are representative of the patterns occuring for the respective modulation types (provided that one looks at those traces where the number of errors is sufficiently high; clearly, for traces with only a few errors the figures appear more irregular).

There is a peak at the beginning of a packet's data part. For QPSK and BPSK traces without scrambling it is frequently found between bit ≈ 200 and 250, for traces with scrambling often a peak at positions ≈ 80 to 120 is present.

The figures for the BPSK and QPSK traces show some periodicity. From inspection, for BPSK traces the basic period is 64 bits, for QPSK it is 128 bits. This periodicity is visible with and without scrambling, however, as a visual impression, with scrambling the effect seems to be more pronounced. In [197] some figures are presented which, for selected QPSK traces, show the conditional probability that bit $n + k$ is wrong given that bit n is wrong (calculated over the respective bit error indicator sequence (BEIS), see Section 6.3). These figures indicate that indeed often bit errors have a distance of 128 bits.

As for the bit-shifted packets, we have no validated explanation for both phenomena, but again we find it highly probable that they are due to artifacts of the wireless receiver's bit synchronization algorithm.

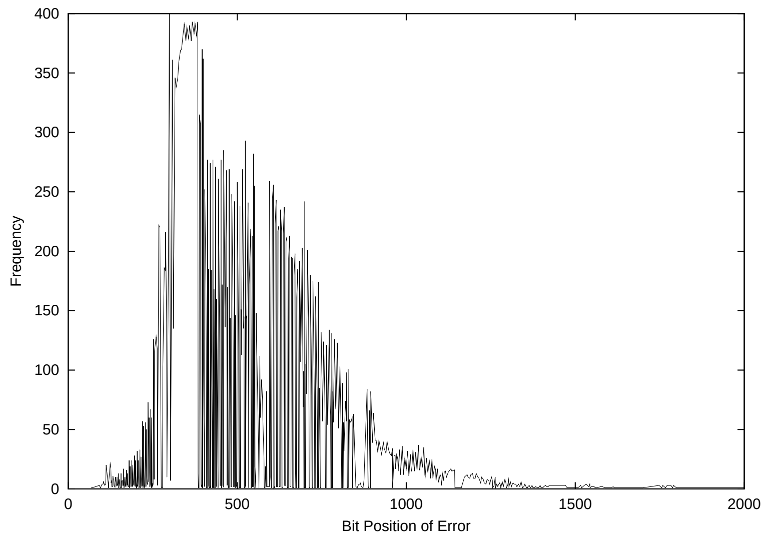

Figure 6.12: Positions of bit errors, **factorial** trace 37 (5.5 MBit/s CCK modulation, no scrambling, 2016 bytes packet size)

Modulation	MBER w/o scrambl.	MBER w/ scrambl.
BPSK	2.5571e-05	0.0003
QPSK	7.4428e-05	0.0001
CCK (5.5 MBit/s)	0.0018	0.0399
CCK (11 MBit/s)	0.0544	0.0589

Table 6.10: Mean Bit Error Rates for different modulation types (**factorial** measurement)

6.5.4 Mean Bit Error Rates

The 11 MBit/s CCK and 5.5 MBit/s CCK traces are excluded from further discussion, since they are extremely error prone. For example, in trace 52 (11 MBit/s CCK, 2016 bytes packet size, without scrambling) 19729 out of 20000 packets do not contain a single well-formed chunk.[3] Many of the 5.5 MBit/s CCK traces are also extremely error prone (specifically with scrambling) and thus are excluded.

The mean bit error rate (MBER) per trace is time-varying over several orders of magnitude, even for the same modulation type and packet size. To illustrate this, we show in Figure 6.14 the mean bit error rate (MBER)s for the **longterm1** measurement; it should be noted that this figure spans more than four hours (see Section 6.4.2). The MBERs reach higher values more often for the traces with scrambling[4] (traces 91 to 180); our data supports this. For the **factorial** measurement the MBER

[3]For many packets it was not possible to compute the error rate, since it was not possible to determine the corresponding expected packet. For those packets where the expected packet could be guessed, bit error rates of 25% to 30% are easily reached.

[4]This is true for both the **factorial** and **longterm1** measurements, taken with the same modem set. For the **longterm2** measurement both scrambling modes show approximately the same mean bit error rate. A possible explanation is as follows: the scrambler `XOR`s the received bit stream with the (already internally `XOR`ed) contents of a shift

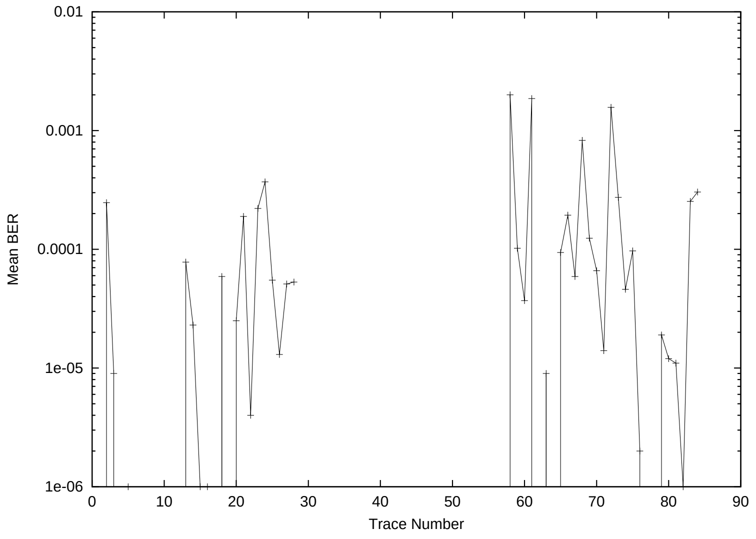

Figure 6.13: Mean bit error rate vs. trace number for remaining traces (logarithmic scale)

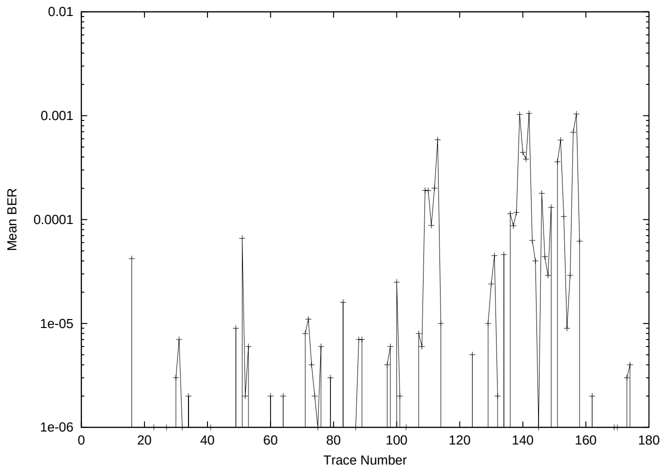

Figure 6.14: Mean bit error rate vs. trace number for **longterm1** measurement (logarithmic scale)

vs. trace# graph (shown in Figure 6.13 for the BPSK and QPSK traces) has the same characteristic of spanning several orders of magnitude.

From looking at the MBERs, only a few clear patterns emerge: MBERs seem to be higher with scrambling enabled, and furthermore, the MBERs increase with transmission speed, as is shown in Table 6.10. The BPSK modulation shows the best error rates, followed by the QPSK scheme. Other patterns, e.g., dependency on packet sizes, were not clearly visible; they are likely overshadowed by the inherently time-varying nature of the link.

6.5.5 Burst Length Statistics

As described in Section 6.3.2, for every trace its bit error indicator sequence (BEIS) $i_1 i_2 \dots i_m$ was formed. Hence, for some given burst order k_0 there is a number of p error-free bursts and error bursts, their respective lengths are given by $X_1 X_2 \dots X_p$ and $Y_1 Y_2 \dots Y_p$.

In Figure 6.15 the mean error burst length of a trace vs. k_0 is shown for some BPSK traces, while in Figure 6.16 the same is shown for selected QPSK traces. The respective curves are typical for BPSK and QPSK traces. In general, clearly the mean error burst length increases when increasing k_0 from k_0' to $k_0'' > k_0'$. The same is true for the mean error-free burst lengths, since the error-free bursts of length l with $l > k_0''$ survive as they are, while the error-free bursts of length l with $k_0' < l < k_0''$ disappear and are not considered in mean burst length calculation. From Figure 6.16, for QPSK traces one can observe a trend to "step functions" with the steps having a distance of ≈ 128. After inspection of several traces it shows that this behavior is due to the periodicity of bit errors described in Section 6.5.3. A similar behavior, however, with a period of 64, can be observed for BPSK traces (for other traces not shown in Figure 6.15 the "step function" character is more pronounced).

It is interesting to look along the time axis: for selected values of k_0 in Figure 6.17 the mean error burst length vs. the trace number for the **longterm1** measurement is displayed. It can be seen that, even for all parameters fixed, the mean error burst lengths vary substantially over time (the mean error-free burst length for fixed k_0 fluctuates over several orders of magnitude, not shown here). Furthermore, as already seen for the mean bit error rates in Figure 6.14, the mean error burst lengths are higher for scrambling enabled. Thus, the bit error characteristics vary not only on short timescales (from burst to burst, range of milliseconds), but also on larger timescales (trace order, range of minutes).

In Figures 6.18 and 6.19 we show for selected BPSK and QPSK traces of the **factorial** measurement and several values of k_0 the coefficients of variation (CoV) of the error burst length and error-free burst length distribution of the respective BEIS. The variability of the error-free burst length distributions is much higher than that of the error burst length distributions (the latter are typically between 1 and 3). This is due to very long periods of no errors within the respective BEIS. For the error-free burst length distributions, there is a tendency of increased variability for increased packet sizes. This is likely due to the tendency of bit errors to cluster at the beginning of packets (see Section 6.5.3): for large packets the packet beginnings have a larger distance in the BEIS, hence, likely longer error-free bursts occur, which increases the variance. Furthermore, the CoV of the error-free burst lengths is larger for BPSK as compared to QPSK. A likely reason is the typically lower MBER of BPSK traces, which leads to longer error-free bursts, the latter increasing the variance. All observations are also true for the other BPSK and QPSK traces.

register of eight bits depth. The shift register changes its content with every bit depending on the incoming bit stream. Hence, an error in the bit stream propagates into the shift register and may influence the following eight bits.

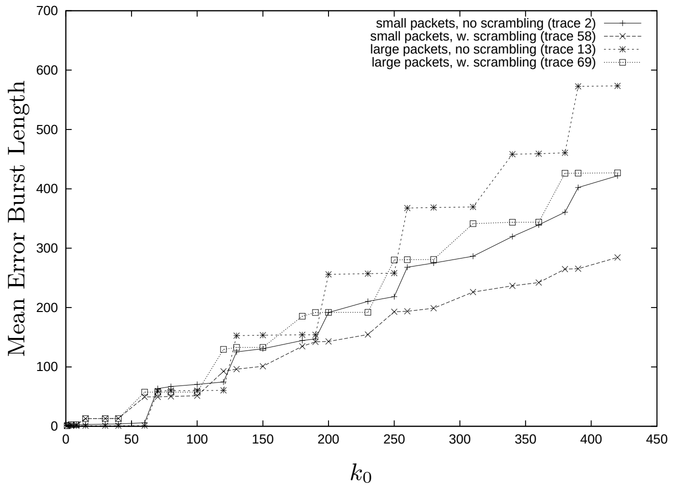

Figure 6.15: Mean error burst length vs. k_0 for selected BPSK traces BEIS (**factorial** measurement)

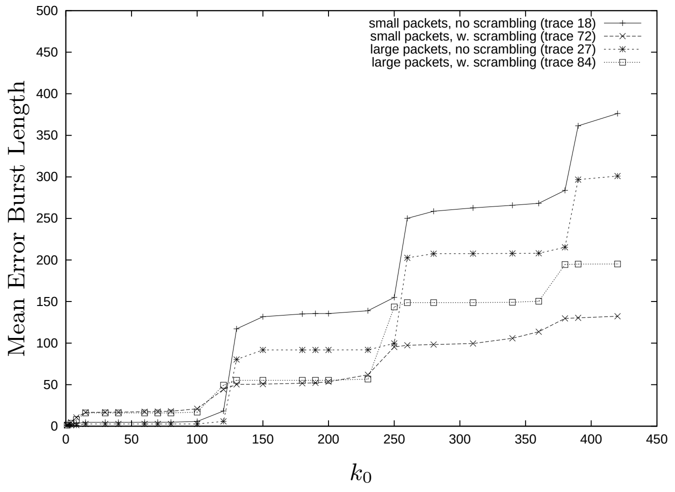

Figure 6.16: Mean error burst length vs. k_0 for selected QPSK traces BEIS (**factorial** measurement)

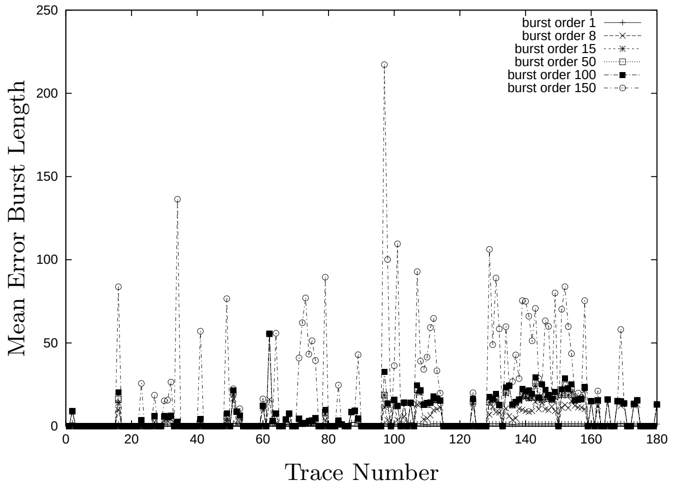

Figure 6.17: Mean error burst length vs. trace number for selected k_0 (for BEIS of **longterm1** measurement)

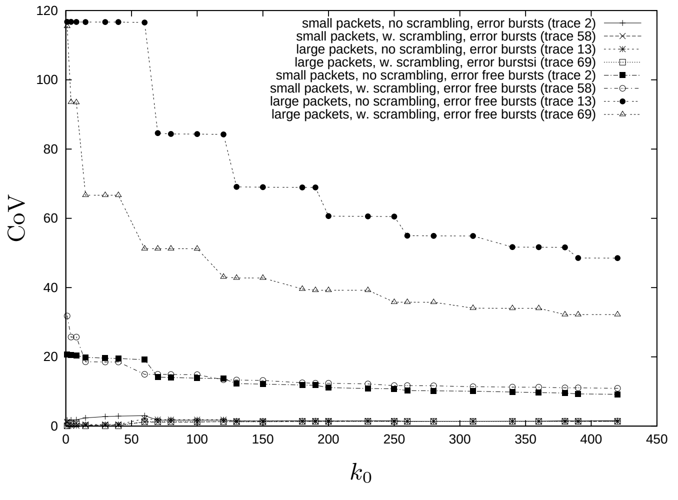

Figure 6.18: Coefficients of variation (CoV) of error burst lengths and error-free burst lengths vs. k_0 for selected BPSK traces BEIS (**factorial** measurement)

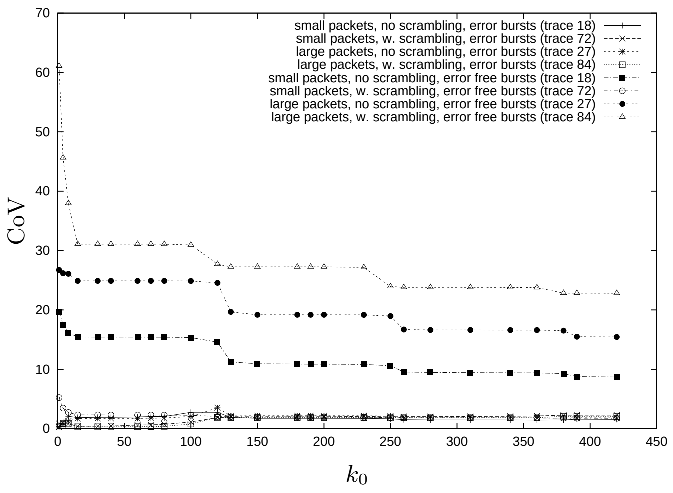

Figure 6.19: Coefficients of variation (CoV) of error burst lengths and error-free burst lengths vs. k_0 for selected QPSK traces BEIS (**factorial** measurement)

Modulation	len=1	len=2	len >2
BPSK/wo	95782	657	250
BPSK/w	987863	615	379
QPSK/wo	191740	65611	7979
QPSK/w	264891	123225	13833

Table 6.11: Burst lengths of error burst with density one for QPSK and BPSK traces

6.5.6 Error Densities and Error Clustering

In this section we investigate briefly the error densities. As a result, we achieve statements about how many errors occur within a small number of bits. This information is interesting for the design of FEC codes, since it provides the design goals for FEC codes.

First we consider the BPSK traces. For every trace and several burst orders k_0 we have calculated its *error density sequence* $\frac{Z_1}{Y_1} \dots \frac{Z_p}{Y_p}$ from the trace's BEIS. Following this, we have merged the error density sequences of all the BPSK traces without scrambling into a single set of values which, after proper renaming, are denoted as $\frac{Z_1}{Y_1} \dots \frac{Z_{p'}}{Y_{p'}}$. In Figure 6.20 the cumulative distribution function $D(x) = \Pr[\frac{Z_i}{Y_i} < x]$ of this single set is shown for different burst orders. The curve for burst order $k_0 = 60$ is representative for all burst orders $k_0 < 64$.[5] Accordingly, the curve for burst order $k_0 = 70$ matches very closely the curves for $64 < k_0 < 128$, and the curve for $k_0 = 130$ is representative for the curves with $k_0 > 128$. It can be seen that for the short burst order $k_0 = 60$ most of the mass is on density one, in almost all cases corresponding to single bit errors (see Table 6.11), while the remaining densities have comparably low frequencies. This means that in almost all cases a single bit error is surrounded by two error-free bursts of at least 61 bits length. The results shown for the larger burst orders k_0 do not contradict this finding, since the mass shifted from density 1 to densities of below

[5]For the BPSK traces $k_0 = 64$ is a critical value because of the 64 bit periodicity described in Section 6.5.3. Correspondingly, for QPSK $k_0 = 128$ is a critical value.

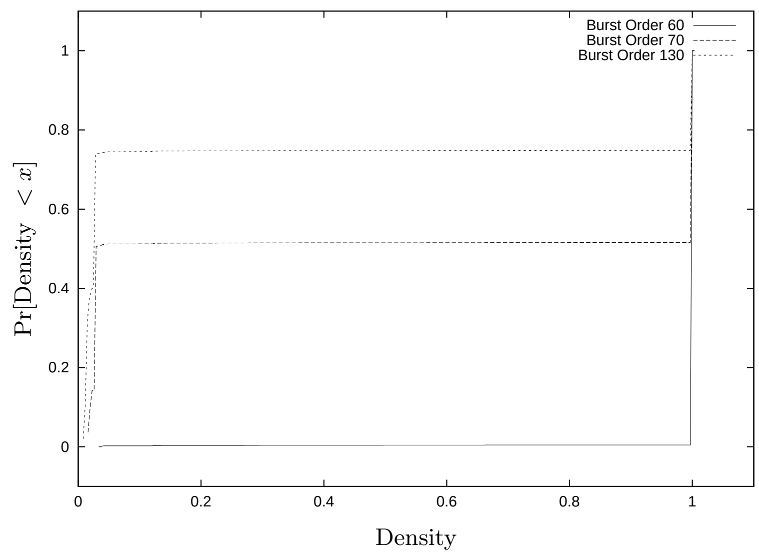

Figure 6.20: Cumulative distribution function for the error densities of all BPSK traces without scrambling

5% corresponds to those bursts where by the 64 bit periodicity phenomenon a single burst consists of two (or very few) erroneous bits with distance of 64 bit.

In Figure 6.20 the cumulative distribution function $D(x)$ for the QPSK traces without scrambling and burst orders $k_0 = 120$ (below the critical value $k_0 = 128$), $k_0 = 130$ and $k_0 = 260$ is shown. Again, the curves are typical for the curves in their respective classes. Of special interest are those bursts with densities between 10% and 80%, which make up $\approx$ 20% of all bursts.[6] An amount of 90% of these bursts have a length of 16 bits or fewer, with the dominant burst lengths being 4, 14 and 16 (the other lengths are insignificant). From the bursts of 14 or 16 bits length, 94% have only two bit errors (at both ends of the burst).

[6]The mass shift from density one to densities below 5% when switching from $k_0 = 120$ to $k_0 = 130$ can be explained by the 128 bit periodicity typical for QPSK.

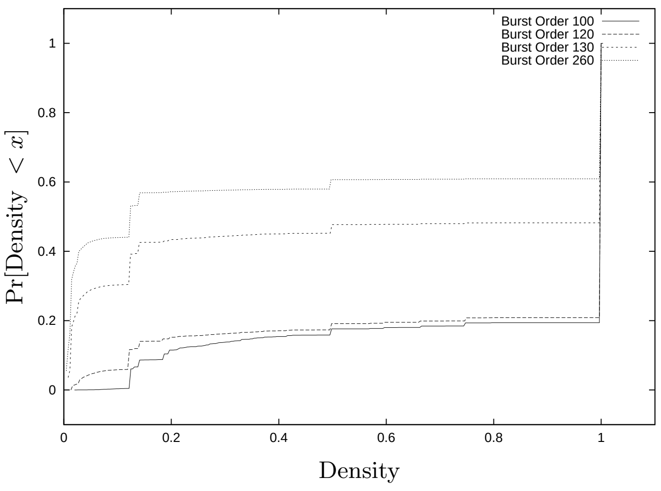

Figure 6.21: Cumulative distribution function for the error densities of all QPSK traces without scrambling

6.6 Review of other Measurement Studies

In this section for the sake of reference and completeness some other measurement studies are summarized, with focus on packet level or bit level measurements. Lower level (wave propagation) measurements of indoor scenarios are discussed in references [17] (measurements on channel impulse response), and [7] (overview of propagation measurements and models). In the following, we restrict the discussion to indoor measurements.

Within the FUNBUS project some measurements with an IEEE 802.11-compliant DSSS PHY were carried out [52, chap. 9 and 10]. Namely, the Silver Data Stream radio modem [157] was used. Their measurement setup (developed at the ifak, Institute for Automation and Communication, Magdeburg/Germany) has similarities to ours: MACless radio modem, dedicated transmitter and receiver stations, packet stream with equidistant start times and well-known packet contents (3, 64 and 252 bytes long packets). All measurements were performed without diversity, BPSK modulation, and scrambling enabled; the transmitter power was not given. Their setup was able to distinguish between lost packets, truncated packets (data part too short), erroneous packets (of correct length but with bit errors) and correct packets. In an outdoor line of sight (LOS) scenario their setup showed no transmission errors for distances up to 800 meters, hence it can be assumed to work properly. Four scenarios were investigated: a) an undistorted free-space scenario (*outdoor LOS*); b) a *residential area* with a building, a sports field, parking cars and trees; c) two indoor scenarios: within a *flat* and in an *industrial environment.*

- In the outdoor LOS scenario without interferers, as long as the weather was fine, no errors were found for distances up to 800 meters, when there is fine weather. However, on a foggy day for distances higher than 530 meters significant fractions of packet losses (more than 90% at 730 meters distance) and packet truncations (up to 20%) can be observed.
- In the residential area scenario up to 100% packet losses were observed in a non line of sight (NLOS) setting with 30 meters distance. For a LOS setting with 100 meters distance, the transmission was error-free. However, when a metallic fence was put between receiver and transmitter, 97% of the packets were lost.
- In the flat scenario, when both stations were moved within the flat, virtually no errors occured, regardless whether there was a LOS or NLOS connection. However, when moving one station to another floor, very high packet loss rates (up to 100%) and packet truncation rates (up to 30%) were observed, and the results were varying.
- The industrial environment scenario was located in a fabric hall of the University of Magdeburg. There was no activity at the time of the measurements. The results indicate that in a LOS scenario for varying distance between 5% and 100% of all frames were error-free, however, there was no relationship to the distance. The missing packets are mainly due to packet losses (sometimes up to 60%) and truncations; the respective rates are varying. The transmission quality in NLOS scenarios was rated as “unusable”.

Some of their results confirm ours, e.g., the occurence and order of magnitude of packet losses and their time-varying nature. These are of prime importance for the design of wireless MAC protocols for fieldbuses.

In a recent paper of Eckhardt and Steenkiste [41] adaptive error correction techniques were applied to WLAN traces, recorded in measurements using a DSSS WaveLAN (902-928 MHz frequency band, 2 MBit/s QPSK modulation, receiver antenna diversity). They generated a specific UDP/IP packet stream, the underlying WaveLAN uses a CSMA/CA variant without retransmission on the MAC level. This stream is captured and stored by a special receiver station, even if the frame checksum generated by the WaveLAN MAC was wrong. Their main focus was on investigating the effect of active interference sources and attenuation on the occurence of bit errors / bit corruption, packet truncations and packet losses. The authors attribute packet truncations to loss of receiver synchronization, and lost packets to receiving a too heavily distorted physical layer preamble. The main findings are:

- Bit errors were insensitive to the bit value.
- At short distances with no interferers the PLR was zero and the packet error rate (PER) (rate of packets with at least one bit error) of $3.4 \cdot 10^{-4}$ was negligible.
- With co-channel interferers (cordless telefone in the same frequency band) the PLR went up to 31%, and truncation rates of up to 23% were observed, depending on the distance and mobility of the interferers. Interestingly the bit corruption rates varied strongly with the scenarios, but seem not to be coupled to the packet loss and packet truncation rates.
- Almost all packets with corrupted bits had fewer than 5% of their bits corrupted. Bit errors did not have a trend to cluster in specific bit positions within a packet. Errors tend to occur in bursts, which were most often restricted to one or two bytes length (burst order $k_0 = 7$).
- The packet truncation rate depends on the packet length: shorter packets were truncated more rarely, also the relative percentage of lost bits was higher for longer packets.
- The PLR and BER were insensitive to the packet size.

The same authors had published another set of results on WaveLAN measurements before [40], using the same measurement setup. They had investigated signal quality parameters in an in-room LOS scenario, a scenario with passive obstacles, and a third scenario with active interferers. In the in-room LOS scenario:

- At fixed distance with good signal level there occured virtually no bit errors but a PLR of 0.04%. Furthermore there was only a single packet truncation for more than a million packets transmitted.
- For varying distance they found that the signal level decreases with increasing distance. The given plot suggested at least a quadratic loss of signal level.
- When the signal level goes below some threshold the PLR drastically increases.

The investigations with passive obstacles showed that a single wall only affects the signal level, but not the signal quality. As a general trend they observed that decreasing signal level tends to increase the number of packets with corrupted bits, while decreasing signal quality tends to increase the number of truncated packets. Moving human bodies between sender and receiver increased all types of errors. The sensitivity against active radiation sources depended on the nature of the source. If the frequency used has enough offset to the 902-928 MHz frequency band, the receiver filters work well and the interferer causes no harm. This was verified with an amateur radio transmitter in the 144 MHz band and with a microwave oven working in the 2 GHz band. Regarding interference in the same frequency band, one can distinguish between narrowband interference (produced with a 900 MHz cordless phone) and broadband interference (produced with a 900 MHz spread spectrum cordless phone). While the WaveLAN was robust against narrowband interference, the broadband interference led to $\approx 50\%$ PLR and truncation of every arrived packet. The susceptibility to broadband interferers was confirmed, when the cordless phone was replaced by a competing WaveLAN station.

The work described in reference [118] is focused on tracing and modeling of wireless channel errors on a packet level, incorporating a full UDP/IP protocol stack over WaveLAN (902-928 MHz frequency band, DSSS, QPSK, 2 MBit/s). For each trace a transmitter sends a sequence of fixed-size UDP packets with a fixed generation rate to a receiver. All other interference and packet sources are suppressed. When only the packet arrival rate is varied, the PER rate does not change. When varying the packet size, the PER doubles with every 300 byte increase of packet size (starting with 100 bytes), reaching $\approx 10^{-3}$ for 1400 bytes. When only varying the distance, the PER doubles every 17 feet, up to ≈ 0.08 at 130 feet. They defined a binary indicator sequence by assigning a one for an erroneous packet and a zero for an error-free packet. The mean error burst length was in most cases between two and three, while the mean error-free length seems to decay almost linearly with increasing distance. The authors calculated suitable parameters for a simple two-state semi-Markov model for generating binary indicator sequences from their measurements. They found that a two-state Markov model (Gilbert/Elliot model, see Section 6.7.1) with its need for geometric burst-length distributions would not fit their data, since they observed a much larger variability than possible for a geometric distribution.

One of the earliest WLAN packet-level studies is [39]. Again, a 902-928 MHz WaveLAN with 2 MBit/s QPSK, DSSS, and receiver antenna diversity was used. They have performed their measurements in a 56 metres long hallway, with a dedicated sender and receiver station. The receiver is placed near a wall.[7] The authors have focused on varying the distance. For increasing distance the PER increases, at 50 m it is below 2%. However, when increasing the distance to 56 m, the PER increases to $\approx 50\%$. This behavior is observed for several packet sizes. Furthermore, for the same distance with increasing packet sizes the PER increases. In their evaluation, if two erroneous bits occured in neighbouring bytes, they belong to the same error burst. When evaluating their bit error indicator sequence (BEIS), they found that errors tend to be non-consecutive, typically only the minimum number of bits for constituting an error burst is erroneous (only one erroneous bit per byte), and most packets show only a single burst. Furthermore, some error burst lengths are strongly preferred at all distances and packet sizes, e.g., 13 or 14 bits long. This is similar to our results with 14 or 16 bits long error bursts for QPSK. When looking at the burst length distribution functions, for longer runs they observed a (decaying) sawtooth pattern with maxima at multiples of 8. Hence, the authors also found some position dependency in the bit error behavior. The mean bit error rates are found to be "roughly constant" over all packet sizes and distances. An explanation for this could be that multipath fading instead of noise is the dominant source of errors. The effects of multipath fading do not correspond in a simple way to the distance.

[7] They reference a preliminary study in the same environment, measuring channel impulse response. One interesting result is that 75% of delay spreads were below 50 ns, which is small compared to 1 μs channel symbol duration. Thus the channel shows no intersymbol interference and the paths other than LOS appear as random noise.

6.7 Stochastic Modeling of Bit- and Packet-Errors

Often, simulation-based performance evaluations are done with *stochastic* link error models. An alternative approach would be to use traces, but these are only rarely available and their handling is often perceived as cumbersome. In stochastic models, a simple stochastic process, which often can be described in terms of a few parameters, is employed to generate bit error patterns.[8] Some models are frequently used in performance studies of MAC or link-layer protocols, e.g., the *Gilbert/Elliot model.* A stochastic model is most useful if:

- it needs only a small number of parameters (say, not more than a few dozens), which can be communicated conveniently to other people and be used in their simulations.
- it can be parameterized from "real data", obtained from measurements.
- it produces error patterns that approximate reality in a to be defined sense.

From the measurements, one can distinguish between packet losses and bit errors. This distinction makes sense for the designer of MAC protocols and coding schemes, since packet losses cannot be combatted by any MAC scheme by influencing the packet's data part, as they are caused by not getting bit synchronization, which would happen in the packets PHY header. After separating both issues, it is possible to describe both packet losses and bit errors by means of (binary) indicator sequences. Hence, for stochastic channel modeling we are interested in methods for generating binary indicator sequences, matching some statistics of a given binary indicator sequence.

There are several requirements for such a process: a) it should have low computational and memory complexity; b) it can be easily implemented on a computer; c) its parameters can be computed from the measurement results; d) the models output should match certain channel statistics with sufficient accuracy; and e) preferrably, it is conceptually simple.

In the following Section 6.7.1 a brief overview of the most popular stochastic models for generating bit errors (or binary indicator sequences) is given. Where appropriate, we explain how these models can be parameterized from the measurements. Then, in Section 6.7.2 an alternative model type for generating binary indicator sequences is introduced, called "bipartite" models. It is targeted to remedie some of the other model's limitations. It gives much better results in matching the first and second order statistics of a trace. In Section 6.8.3, we propose an overall structure of a stochastic channel model which uses explicit submodels for the phenomena of packet losses and bit errors and combines these submodels in a unified framework. Finally, in Section 6.7.3 a simple example system is used to demonstrate the benefits and shortcomings of the different stochastic models, as compared to directly using a trace. The results confirm that the bipartite model is much more accurate than other models, at the expense of only a small increase in model complexity.

6.7.1 Overview of Common Stochastic Models

We briefly present some of the most commonly used stochastic processes used for generating binary indicator sequences (in most cases interpreted as bit errors). The majority of these models use time-

[8] Other types of models, e.g., those directly targeting wave propagation aspects [137], [79], are beyond the scope of this thesis.

homogeneous Markov chains (discrete or continuous). Many of these models are also discussed in [16].

Let us assume that a binary indicator sequence $i_1 i_2 \ldots i_m$ is given, and the associated burst length sequence is $X_1, Y_1, Z_1 \ldots X_p, Y_p Z_p$. Let the mean error rate be $\bar{e}$, the mean error-free burst length be $\bar{X}$ and the mean error burst length be $\bar{Y}$ (see Section 6.3.1).

The first and most simple model is the *independent model*, where one fixed bit error probability $p_b \in (0, 1)$ is given, and, conceptually, for every bit a Bernoulli experiment is carried out, such that every experiment has the same parameter p_b and is independent of all other experiments. In order to match at least the mean value between model and given indicator sequence, clearly $p_b = \bar{e}$ must be chosen.

This model is quite simple to implement, however, does not capture the "bursty" nature of channel errors observed in low level measurements or predicted from propagation models. A quite common approach is to introduce some additional "channel state".

Two-State Models

A very popular model is the two-state *Gilbert model* [57] or *Gilbert/Elliot model* [42]. It assumes a "good" channel state and a "bad" channel state. Within every state, bit errors occur according to the independent model with rates e_{g} and e_{b}, respectively ($e_{\text{g}} \ll e_{\text{b}}$). Conceptually, the next channel state is determined after every bit according to a discrete two-state time-homogeneous Markov chain with transition matrix

$$\mathbf{P} = \begin{pmatrix} p_{\text{gg}} & 1 - p_{\text{gg}} \\ 1 - p_{\text{bb}} & p_{\text{bb}} \end{pmatrix}$$

(with p_{xy} being the probability that the next state is y given that the current state is x). From the Markov or memoryless property, the state holding times have a geometric distribution and are independent of each other. The steady-state probability of being in the good state is given by

$$p_{\text{g}} = \frac{1 - p_{\text{bb}}}{2 - (p_{\text{gg}} + p_{\text{bb}})},$$

and the steady-state probability of being in the bad state is given by

$$p_{\text{g}} = \frac{1 - p_{\text{gg}}}{2 - (p_{\text{gg}} + p_{\text{bb}})}.$$

The mean bit error rate can then be calculated as $\bar{e} = p_{\text{g}} e_{\text{g}} + p_{\text{b}} e_{\text{b}}$.

It is easy to see that the Gilbert/Elliot model has short-term correlation properties for bit errors (i.e., $\Pr[i_{n+k} = 1 | i_n = 1] \neq Pr[i_n = 1]$ for proper model parameters and small values of k), however, the burst length sequences are uncorrelated. The Gilbert model uses $e_{\text{g}} = 0$ and $e_{\text{b}} = 0.5$ while in the Gilbert/Elliot model these values can be chosen arbitrarily. To determine the matrix $\mathbf{P}$ it is sufficient to know the mean state holding times $\frac{1}{1 - p_{\text{gg}}}$ for the good state and $\frac{1}{1 - p_{\text{bb}}}$ for the bad state. In our setting it is natural to associate the "good" state with the error-free bursts and the "bad" state with the error burst. Thus, we choose $e_{\text{g}} = 0$, $e_{\text{b}} = \frac{\sum_{i=1}^{p} Z_i}{\sum_{i=1}^{p} Y_i}$, $p_{\text{gg}} = 1 - \frac{1}{\bar{X}}$ and $p_{\text{bb}} = 1 - \frac{1}{\bar{Y}}$). It is then easy to see that the mean bit error rate $\bar{e}$ and the mean bit error rate generated by the model are the same. Clearly, the same holds for the mean error burst length and mean error-free burst length.

If either the error-free burst length or error burst length distribution is not geometric, it is appropriate to drop the Markov assumption and to use other distributions, which better match the first and second moments of the error-free burst length and error burst length distributions. Candidate distributions are, e.g., the binomial or Poisson distributions, or quantized versions of continuous distributions, e.g., the lognormal distribution [196]. This class of models is denoted as *semi-Markovian models*. However, it is important to note that this type of models also has short-term correlation properties for the bit errors, but, since all burst lengths are independent, the model allows no correlation for the burst length sequences.

N-State Models

A popular model using an N-state Markov chain is described in [184]. In this model, the Markov chain is derived from modeling the instantaneous signal-to-noise ratio at the receiver (R-SNR) under Rayleigh fading assumptions. The range of possible R-SNR values is partitioned into intervals, each interval is associated with a state of the Markov chain and a bit error rate value. Since the R-SNR can be assumed to be time-continuous, transitions are only allowed between neighboring states, and thus the transition matrix has tridiagonal structure. The possibility to generate this model from a few simple physical parameters (e.g., mean R-SNR value and doppler frequency) makes it attractive. However, since these values are not available from our measurements, this model cannot be parameterized.

In [51], a Markov model with N states is described, which are subdivided into two state classes, namely error-free states (class A) and error states (class B). This class of models is called *Fritchman models*. If the system's current state is in class B, the transmitted symbols are erroneous with probability 1. In general, the possible state transitions are not restricted. An application of Fritchman models to measurements of a 142 MHz land mobile channel can be found in [160]. A similar model is described in [102], however, this uses two transition matrices $\mathbf{P}$ and $\mathbf{Q}$, where $\mathbf{P}$ is used every time the preceding channel symbol was in error, while $\mathbf{Q}$ is used otherwise. Here it is allowed to have bit error rates different from 0 and 1.

Kim and Li [87, 88] propose to use a *Markov modulated process* (MMP) for approximating the first and second order statistics of packet error rate measurements. They employ an N-state time-continuous Markov chain, for which the generator matrix $\mathbf{Q}$ is of circulant type (i.e., each row is the previous row, shifted by one element), and within each state i the channel has a packet service rate of γ_i (channels currently subject to errors have a lower service rate) [139]. As input data they use the service rate process $\{R_c(t)\}_{t\in\mathbb{R}}$, obtained from measurements [140]. They use the fact that the power spectral density function $R(t)$ of the MMP generated by $\mathbf{Q}$ and $(\gamma_0, \ldots, \gamma_{N-1})$ can be explictly represented by the eigenvalues of $\mathbf{Q}$. These values are chosen such that $R(t)$ matches the measured power spectral density function of $\{R_c(t)\}_{t\in\mathbb{R}}$ as closely as possible. Then $\mathbf{Q}$ can be constructed from the chosen eigenvalues. Although this approach has attractive features, it is not easily adapted to our methodology and notions.

Another class of models are the *Hidden Markov Models* (e.g., [54, 172, 174], an in-depth treatment can be found in [173]). The methodology proposed in [54], however, uses only one state for the error-free bursts, and thus the error-free burst lengths are a priori independent. Furthermore, the Hidden Markov Models lack a direct intuition between the channel behavior and the underlying Markov chain. We will not discuss these models further.

6.7.2 Bipartite Model for Generating Indicator Sequences

The models described so far have some problems:

- The Gilbert/Elliot model allows only geometrically distributed state holding times. When applying this model to our traces' bit error indicator sequence (BEIS) such that the mean burst lengths match, the resulting geometric distributions for both error burst lengths and error-free burst lengths will have coefficients of variation close to 1. However, especially for the error-free burst lengths coefficients of variations of 20 up to 100 are typical (see Figures 6.18 and 6.19). Thus, the "true" distributions are much more variable than the geometric distribution.
- The Gilbert/Elliot model and other two-state models can only express (very) short-term correlation for the bit error/packet error process and no correlation in the burst length processes, since the burst lengths are a priori independent. However, in several traces, correlation is present [197].
- Most other models are rarely used and their parameterization is quite complex.
- Some models (e.g., [184]) need physical parameters (e.g., receiver-SNR) not accessible to the measurement setup.

It would be nice to have stochastic models of moderate complexity but capable of expressing variability of the burst lengths and longer-term correlation.

We introduce a special class of Markov models, called "bipartite models". This name stems from the fact that the corresponding Markov chain forms a bipartite graph. This model offers advantages over the models discussed so far: a) the underlying distribution functions for the error burst lengths and the error-free burst lengths can be approximated with the desired degree of accuracy (at the cost of increasing the memory needed for the model), and b) depending on the number of states, the model can express longer-term correlation than the two-state models discussed so far. Furthermore, the model is conceptually related to the notion of binary indicator sequences and burst length sequences, its parameterization from the traces is straightforward, and it is intuitively appealing.

The bipartite model is similar to Fritchman models [51], with some distinguishing features:

- the shape of the transition matrix is explicitly restricted to periodic ones;
- bit errors do not necessarily occur with probability 1 in bad states; and
- the burst length distributions can be arbitrarily chosen.

Model Description

The approach is to employ a number n_1 of "bad" states and n_2 of "good" states and to allow state transitions only from good states to bad states and vice versa (forming a bipartite graph). An example model with two good states and two bad states is shown in Figure 6.22. When states are numbered $s_1, \ldots, s_{n_1}, s_{n_1+1}, \ldots s_{n_1+n_2}$, the transition matrix has the form:

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{Q}_1 \\ \mathbf{Q}_2 & \mathbf{0} \end{pmatrix}$$

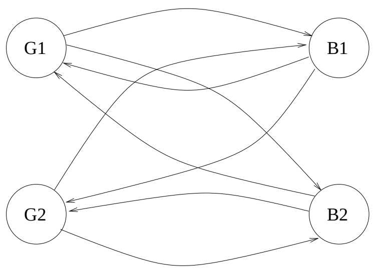

Figure 6.22: A sample bipartite Markov chain

where $\mathbf{Q}_1$ is an $n_1 \times n_2$ stochastic matrix[9] describing the state transitions from the bad states to the good states, while $\mathbf{Q}_2$ is an $n_2 \times n_1$ stochastic matrix for the transitions from the good states to the bad states. A state s_i corresponds to a set I_i of possible (error or error-free) burst lengths that can be generated in this state. Typically, the set I_i is an interval of natural numbers.

The operation of this model is as follows: every state s_i is assigned a discrete random variable p_i with probability distribution $p_i(k) = \Pr[p_i = k]$ (with $k \in \mathbb{N}$ and $p_i(k) = 0$ for $k \notin I_i$) and associated distribution function $F_i(x) = \Pr[p_i \leq x]$. This random variable takes values on a finite interval of the natural numbers. When the system enters a specific good state s_ν, a random number is drawn from the distribution p_ν. This random number is then interpreted as the number of bits for which no errors occur. When the system enters a specific bad state s_μ, again a random number is generated according to p_μ, determining the error burst length in bits. For an error burst we make the assumption that at least at both ends an error occurs, in the remaining burst the bit errors occur independently with a fixed rate r_i.

In order to build a model from the traces one needs to choose the numbers of states n_1 and n_2, the matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$, the probability distributions p_i and the bit error rates in the bad states. Assume that for the error-free burst lengths $X_1 \dots X_p$ the distribution function is $F_X(\cdot)$ and for the error burst lengths $Y_1 \dots Y_p$ the distribution function is $F_Y(\cdot)$. A simple approach can be summarized as follows:

- Select a number of good states and bad states.
- Partition the range of possible error-free burst lengths into subintervals $[a_i, b_i)$ such that for every subinterval we have $F_X(b_i) - F_X(a_i) \approx \frac{1}{n_2}$. Do the same for the error burst lengths. To every error-free subinterval we assign one error-free state, and to every error subinterval one error state. Hence, every state of the markov chain is associated with an interval.
- Construct the transition matrix $\mathbf{P}$ by simply counting in a traces burst length sequence for every state i the number of times it is left towards every possible target state j and divide this by the total number of times the system has left state i.
- Assign to every state i a random variable p_i generating the burst lengths of the corresponding interval. The choice of p_i is somewhat arbitrary, but the best results are achieved if it matches at least the mean value of the burst lengths lying within the interval.

[9]In a *stochastic matrix* all elements are nonnegative and the rows sum up to 1. An analogous definition holds for a *stochastic vector*.

- For the error states we assume that errors occur independently with a fixed rate. For every error state i, let $\Gamma_i \subset \{1, \ldots, p\}$ denote the subset of all error bursts which belong to state i and use

$$r_i = \frac{\sum_{k \in \Gamma_i} Z_k}{\sum_{k \in \Gamma_i} Y_k}.$$

The procedure for constructing the transition matrix $\mathbf{P}$ is described in some more detail in [196]. The model allows to choose arbitrary distributions for the subintervals. The accuracy of the model depends to a large extent on how good the distributions $F_X(\cdot)$ and $F_Y(\cdot)$ are approximated. Especially the error-free bursts with their long but sparsely covered tail must be handled with care.

Main Model Characteristics

From the description given in the previous section it is easy to see that $\mathbf{P}$ generates a periodic Markov chain with period 2, and thus is not ergodic. This means that there is no steady-state probability vector π satisfying

$$\begin{aligned} \pi^T &= \pi^T \cdot \mathbf{P} \\ \sum_{i=1}^{n} \pi_i &= 1, \end{aligned}$$

(where $n = n_1 + n_2$ is the total number of states). However, this class of models satisfies a weaker form of steady state condition. It can be shown, that in the long run under certain assumptions for each state s_i the fraction of the number of visits in state s_i w.r.t. the total number k of state transitions so far converges with probability one to a fixed value a_i. Furthermore, for states 1 to n these values sum up to one. We give a proof for this in Appendix A.1.

Furthermore, it is shown in Appendix A.2 that the bipartite models allows for arbitrary precision in approximating the distributions of error burst lengths and error-free burst lengths.

The bipartite model has basically the same autocorrelation properties as other Markovian models. Correlation vanishes asymptotically, while present in the short run. However, due to the higher number of states, it can express longer-term correlation than the simple two-state models (see Appendix A.3).

6.7.3 Comparison of Different Models

In this section we show that the different stochastic models achieve different accuracy in predicting selected performance parameters of an example system. The simple two-state models give quite good results for an aggregate metric, but they are not able to predict a certain form of correlation over longer timescales. The bipartite model gives much better results for this, at the cost of a moderate increase in model complexity (in terms of number of states).

We have chosen to build an example system of two stations communicating with each other. A part of this system is the wireless link, which is modeled using both a real trace as well as several stochastic models, which in turn are parameterized from this very trace. The accuracy of the models is determined by comparing the results obtained with the models with those of using the real trace.

	trace 24 (k_0 : 100)	trace 24 (k_0 : 150)
MBER	0.000370	0.000370
mean EBL	6.873	114.807
CoV EBL	2.457	2.341
max. EBL	229	6529
mean EFBL	6353.049	11514.112
CoV EFBL	77.441	57.775

Table 6.12: Summary statisitis of trace 24 (EBL=error burst length, EFBL=errorfree burst length, MBER=mean bit error rate)

Description of Example System

One transmitter and one receiver station are connected via a wireless link. The transmitter wishes to transfer a file of 1 GB (gigabytes) to the receiver. The file is split into packets of 1000 bits, the protocol overhead (headers) is neglected. There are no further stations present and no MAC protocol or propagation delay is considered. The transmitter sends a data packet and waits for an acknowledgement. If the ack does not arrive within two bit times after transmission finished, the packet is repeated, otherwise the next packet is transmitted (Send and Wait Protocol). Data packets can be subject to errors, acknowledgements are always error-free and of negligible size. The receiver only acks a packet if it contains no errors. The number of retransmissions per packet is unbounded. The transmission rate is 2 MBit/s QPSK.

Channel Models

For modeling the wireless link we have chosen to use trace 24 of the **factorial** measurement as the basis (no scrambling, 2016 bytes packet size), its corresponding bit error indicator sequence (BEIS) was generated with burst order $k_0 = 150$ (the basic statistics are shown in Table 6.12). The following error models are used:

- in the null model no errors occur at all
- the independent error model with BER $p = 0.000370$
- the Gilbert/Elliot model, parameterized such that the mean burst lengths match those given in Table 6.12 (however, the Gilbert/Elliot model largely underestimates the CoV of the traces, since it generates a CoV close to 1)
- a semi-Markov model, using a quantized lognormal distribution, parameterized such that mean value and variance of the generated burst lengths match the trace.
- several bipartite models, differing in their respective number of good and bad states. The parameterization of these models is done as outlined in Section 6.7.2. As distributions for the burst lengths the Beta distribution [106] are used, such that mean value and variance of the measured burst lengths are matched.
- The trace itself.

Model	**Mean Time**	**Variance**	**Prediction Error**
trace	5915.63 s	0	-
independent	8028.61 s	0.47	35.7%
Gilbert/Elliot	6100.30 s	0.65	3.1%
semi-Markov	5803.03 s	137.41	1.9%
null model	5540.00 s	0	-
bipartite (7,7)	5928.29 s	520.84	0.2%
bipartite (15,15)	5914 s	944.63	0.02%
bipartite (20,20)	5917.76 s	608.74	0.03%

Table 6.13: Transmission times for 1 GB data over channels with different error models (based on **factorial** trace 24)

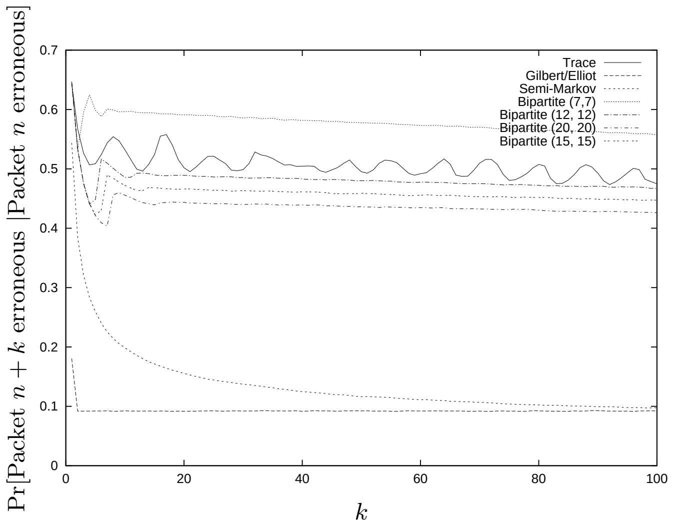

Figure 6.23: Conditional probability that packet $n + k$ is erroneous given that packet n is erroneous

In the simulations only bit errors are considered, the packet loss behavior of trace 24 is ignored.

Results

The performance measure of interest is the time necessary to transmit the whole file, i.e. from sending the first packet until receiving the last acknowledgement. With the exception of the null model and the trace, every simulation was performed 40 times with different seeds of the pseudo random number generator.

The mean values reported in Table 6.13 show that the independent model gives an inacceptably bad prediction of transmission time (35.7% prediction error). The Gilbert/Elliot model predicts the correct result with only 3.1% (Gilbert/Elliot) error, the semi-Markov model improves this result to 1.9%. In both types of models the variation in the results is small. Hence, the error burstiness induced by these models plays a significant role, and it does not suffice to fit only the mean bit error rate. The bipartite models increase prediction accuracy by an order of magnitude, however, the comparably small gain here would not justify the increased model complexity (14, 30 or 40 states as compared to 2).

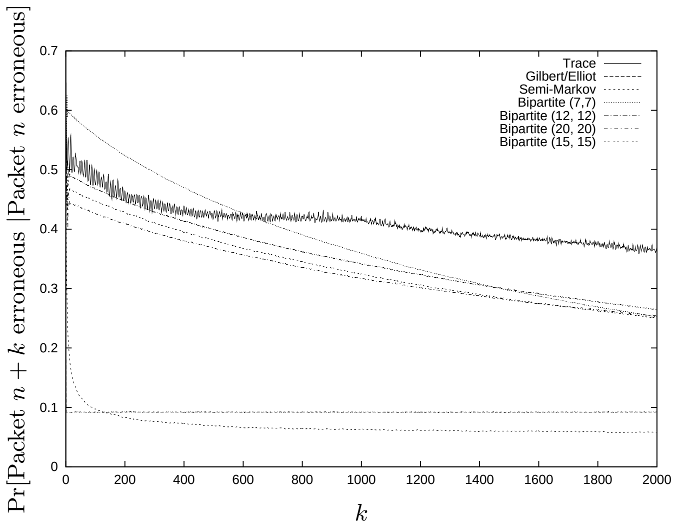

Figure 6.24: Conditional probability that packet $n + k$ is erroneous given that packet n is erroneous

The next example shows that for other purposes the simple two-state models are not accurate enough.

From the receiver's logfile we have generated a special binary indicator sequence, the *packet error indicator sequence (PEIS)*, which is formed by assigning a zero to a correctly transmitted packet and a one to an erroneous packet. From the PEIS we have computed the conditional probability that packet $n + k$ is erroneous, given that packet n is erroneous, using the equation given in Section 6.3.1. The results are displayed in Figure 6.23, while in Figure 6.24 the same is shown for a longer timescale. The following points are remarkable:

- The original trace has a packet error rate (PER) of $\approx 6.3\%$, hence, it is low enough to take the curve for the conditional probabilities as the autocorrelation function of the PEIS (see Section 6.3.1). From Figure 6.24 one can see that the correlation lasts for a large number of packets.
- The simple two-state models (Gilbert/Elliot, semi-Markov) fail to match the long-lasting correlation. Specifically the Gilbert/Elliot model drops off rapidly to its mean PER of $\approx 10\%$. The semi-Markov model with its more variable burst length distributions drops off slower and estimates the mean PER to $\approx 4.5\%$.
- The bipartite models give a rather good approximation on the shorter timescale (up to 100 packets) and drop off much more slowly than the simple models. On very short timescales (<10 packets) the shape of the bipartite (12,12), (15,15) and (20,20) curves follow those of the original trace. On the other hand, the bipartite models predict even for a timescale of 2000 packets significant correlation, where the other models have lost their memory.
- Interestingly, for the bipartite models one cannot say that the approximation is better for higher number of states: the bipartite (12,12) model matches the trace best.

The ability of the bipartite models to match longer-term correlation much better than the other models is confirmed with other traces.

The packet error correlation example is not only of theoretical interest, it has also a practical meaning. Consider the case of a FEC code adding 25% overhead to a packet. The trace's original PER of 6.3%

indicates that applying FEC to every packet would be wasteful. Instead, by the results indicated in Figure 6.23 it makes sense to switch on FEC only in case of retransmissions and to keep it enabled for a large number of packets. The effectiveness of this algorithm will change when switching from the more accurate bipartite model to the two-state models, leading to wrong predictions of the algorithms performance in the latter case.

It is remarkable that the best results are obtained with the bipartite (12,12) model, having 24 states. This finding is somewhat unintuitive, since one would expect a better approximation with increased number of states. However, there is a tradeoff between the number of states and the length of the underlying trace. Increasing the number of states for a given trace decreases the accuracy of the transition probabilites as calculated in Section 6.7.2, since with a high number of states every state is visited fewer times as compared to a lower number of states.

To summarize, the bipartite model gives much better approximations of both the overall transmission time and the packet error correlation than the other models, while having only moderate complexity.

6.8 Conclusions

6.8.1 Summary of Measurement Results

The most obvious, yet far-reaching result is the variability of the wireless link over several timescales, even when looking over hours. This concerns, for example, packet loss rates and mean bit error rates.

This variability can be attributed to frequently changing environmental conditons: moving people, portal crane activity, moving parts of machines, and so forth. Many industrial environments share this property of a frequently changing environment, hence, the measurement study appears representative in this respect. Stated differently: it cannot be said to represent “worst case” or “best case” conditions, but “typical” conditions. This gives us confidence that, although one cannot directly transfer numerical results from this environment to others, the qualitative results in fact can be transferred:

- time-varying behavior over several timescales
- occurence, burstiness properties, and orders of magnitude of packet losses
- great variability of error-free burst length distributions for both packet losses and bit errors, leading to long periods of good conditions. In fact, these long periods of good conditions make a wireless link appear nonstationary.
- sometimes very long lasting correlation in the packet error process of selected traces (see Section 6.7.3) which cannot be captured by models capable of expressing only short-term correlation.
- the tendency of packet losses and bit errors (QPSK) to occur in bursts.

The time variability and the phenomenon of packet losses is also confirmed in another study [52].

We think that the restriction to a single scenario is not really a restriction: when changing to a seemingly “better” position, likely this will not fix the variability and protect from “bad” periods of time. Furthermore, one often has not the freedom to move to “better positions”.

Beyond several statistical results, of some importance is the finding that the 5.5 MBit/s and 11 MBit/s modulation schemes have serious performance problems. For the design of MAC and link-layer protocols the phenomenon of (sometimes long-lasting) packet losses is of utmost importance. More general, MAC and link-layer protocols and coding schemes should incorporate some adaptivity, since the channel is time-varying, both in terms of mean bit error rates and packet loss rates.

Clearly, there is much room for further research. Some interesting questions are the following:

- Can the packet loss rates be influenced by increasing preamble length or by varying the transmission power?
- Error behavior in other scenarios, e.g., in the presence of interferers, with one mobile station, in line of sight scenarios.
- Performance of several (adaptive) FEC schemes for traces with different mean bit error rates, correlation behavior, packet loss rates.

6.8.2 Stochastic Modeling

The popular stochastic models used in the literature (independent model, Gilbert/Elliot model) fail to match the statistics of the measurement data in several respects. The Gilbert/Elliot model:

- does not match the high variability of error burst length and error-free burst length distributions, due to its restriction to geometric distributions;
- does not capture correlation in the burst length sequences; and
- fails to predict long-term correlation in the packet error process of a simulated system (see Section 6.7.3).

Nonetheless, the Gilbert/Elliot model and the semi-Markov model give surprisingly good predictions for selected performance parameters of an example system. Hence, with some care, both models can be useful for simulating errors on a wireless link.

In the literature several alternative error models were proposed, however, most of them have only gained limited popularity. In fact, most performance analyses use the Gilbert/Elliot model. The alternative models often are hard to parameterize, or need higher numbers of parameters. We believe that with the bipartite model a good compromise between simplicity, quality in generating good predictions of performance parameters, and parsimonious parameterization is found. In fact, for the investigated traces frequently good results can be achieved with a moderate overall number of states (between 20 and 40).

There are several interesting topics for further research:

- Comparison of the bipartite model with other N-state Markovian models or Hidden Markov Models of comparable complexity.
- For the bipartite model the influence of the distribution type for the single states on the generated packet error process should be investigated.

- For the bipartite model, from the simulations done so far, selecting the overall number of states between 20 and 40 seems to be a good engineering rule. However, it is interesting to investigate the influence of the number of states in some more detail.

Finally, as a general remark, it does not suffice to take only bit errors into account, but the phenomenon of packet losses should be modeled, too. Hence, an integrated error model as the simple one proposed in Section 6.8.3 should be used.

6.8.3 Overall Channel Model

Single-Channel Case

As basic cornerstone for a stochastic model we use the observations that on the one hand, there are the packet-related phenomena, specifically packet losses, and on the other hand there are bit errors within the packets. For simplicity the following additional assumptions hold:

- There are only packet losses; all other packet impairments are treated as packet losses.
- Packet losses and bit errors are statistically independent of each other.
- The bit error models depend on the modulation type (QPSK, BPSK) but not on the packet size. This contradicts the findings in Section 6.5.3, but leads to simpler models.

We propose to compose the channel model out of three different submodels: the *packet loss submodel* generates a binary indicator sequence which determines for every packet handed to the channel model whether it is lost or not. If the packet is not lost, then one of the two remaining *bit error submodels* is applied to the packet's data part (which can include a MAC protocols header and trailer). If the packet should be transmitted with BPSK, then the *BPSK submodel* is applied, otherwise the *QPSK submodel* is applied. In principle, for each submodel one of the models described in Section 6.7.2 or cited in Section 6.7.1 can be instantiated. In order to keep things simple, we assume that the BPSK and QPSK submodels are independent of each other. If this assumption is not made, and both submodels keep some channel state variable, the problem of how to decide which channel state should be entered when switching between the two submodels arises.

Multiple-Channel Case

We consider the case of multiple stations.

The basic assumption is that between every pair of stations a separate and independent channel w.r.t. its error behaviour exists. The independence assumption can be justified by the following heuristic argument: when multipath fading is the dominant source of errors and by the “nonlinear dependency” of the error behaviour from the propagation environment, it is reasonable to assume that for every pair of stations the propagation environment is “different enough” from those of every other pair. Furthermore, for the case of MAC protocols implementing a strict-TDMA scheme there is no interference from other stations, as opposed to CDMA-based protocols. For a fixed pair of stations the channel in both directions is assumed to be symmetric.

It is clear that the independence assumption has limitations. For example, consider the case where two stations are placed in close vicinity to each other and to an interferer, say, a microwave oven. Clearly, the activity of the oven disturbs both receivers in parallel and makes all the channels from other stations to these two correlated. However, this is not considered furthermore in this thesis.

The channels between different pairs of stations can follow different error models.

6.8.4 Consequences for Design of MAC and Link-Layer Protocols

The measurement results allow to draw some simple conclusions and to make some suggestions for the design of MAC and link-layer protocols aiming at reliability. A general observation is that packet loss rates and mean bit error rates are time-variable, even for the same modulation type bit error rates vary over several orders of magnitude. This calls for inclusion of adaptivity into the protocol implementations.

The occurence of (sometimes long-lasting) packet losses is a major challenge. Packet losses are due to failure of acquiring bit synchronization. This happens already in the header, thus no MAC protocol can protect itself against packet losses by influencing the contents of the data part of a packet. Instead it is necessary to incorporate other mechanisms, e.g., variation of transmit power level, using retransmission schemes, enabling scrambling, or better shielding the radio equipment. Furthermore, for invoking these mechanisms a feedback from the receiver is needed, i.e., the MAC protocol has to incorporate an immediate acknowledgement mechanism. Instead of using whole packets with full-length preambles for immediate acks, it may suffice to rely on the presence or absence of short noise bursts.

The burstiness of bit errors and packet losses suggest to use postponing schemes. Consider a scenario where a single base station (BS) serves a number of spatially distributed wireless terminals (WT), and the traffic is mainly from BS to WT and vice versa. Furthermore, we assume that channel access is time-multiplexed between stations, not code- or frequency-multiplexed. In this case, and with multipath fading being a significant source of errors, the BS has for every WT a different propagation environment (number of paths and their respective losses). Hence, the BS has a separate channel to every WT, which can be viewed as independent from the others. If the errors on all channels have a bursty nature, this calls for introducing link-state dependent scheduling approaches, as proposed in [14]. In this type of schemes the BS may decide to postpone retransmissions (triggered by a packet loss or packet error) and to serve another WT in the meantime. When assuming a strong FEC code and considering only packet losses, our results indicate that, if the retransmission is postponed for 5 to 10 packet times, with $\approx 95\%$ probability it will be successful (compare Figure 6.7).

The occurence of longer outage conditions due to packet losses should be recognized by a MAC protocol and signalled to upper layers to allow them to react properly (e.g., enabling emergency stop procedures). Unlike to wireline or fibre optic communications, applications cannot assume the underlying network to be reliable, but should take changing network conditions explicitly into account. Therefore, some service primitives for signalling network conditions to upper layers should be added to the interface of the MAC or link-layer protocol.

The presence of long-term correlation in the bit error process and packet error process (see Section 6.7.3 for an example) can be considered in different ways. If it is likely that many of the packets following an erroneous packet will also be erroneous, it can be worthwhile to protect them, e.g., by switching back to BPSK (increasing energy per bit), by using FEC, or by increasing transmit power for a while. Another alternative are postponing schemes as sketched above for packet losses in a one BS / many WT scenario.

Chapter 7

Polling Protocols for Wireless PROFIBUS

For MAC-protocols targeted to hard real-time requirements token-based approaches like the PROFIBUS protocol are popular, but they are not appropriate for error-prone wireless links (see Chapter 5). Therefore, it is worthwhile to look whether other approaches do better. In this chapter polling-based protocols are proposed and their realtime performance as defined in Section 4.3.1 is evaluated and compared to that of the PROFIBUS protocol.

As a very general description [163, 161], a polling system consists of a central station (called *base station* (BS)) and a number of stations (called *wireless terminals* (WT)), with each station conceptually having a queue for requests or packets. Conceptually, the BS carries out two different tasks: first it queries the queue's states from the WT's, and second, it assigns bandwidth to the WT's according to the query results and some polling policy. Typically, for a single WT it is assumed that a query is less costly than to serve a packet, otherwise the query overhead could not be justified as compared to a pure TDMA system. The conceptual decoupling between querying a WT's state and serving the WT is useful, since this allows both functions to be carried out at distinct times. In some extreme cases, however, one of these phases can be skipped. Consider the round-robin scheme as an example: the BS does not really care about the WT's queue states, but sends just a poll request, allowing a WT to send a certain amount of data.

For hard-realtime systems we demand that every WT has a collision-free possibility to be queried by the BS, and that always for a nonempty queue (of high priority requests) the time a WT has to wait for the next bandwidth assignment is upper-bounded. These additional requirements make the polling-protocols considered here different from most other polling-based protocols, and also different from demand-assignment MAC protocols [60].

Polling schemes are attractive for the following reasons:

- In PROFIBUS a station gets lost from the ring if (amongst other possibilities) it misses three consecutive token frames, and it needs to be re-included. In polling-based systems a station is not lost from the LAN, if it misses a poll frame, and no re-inclusion is necessary. It suffices to wait for the next poll. In theory, after a one-time registration a WT can be LAN member for indefinitely long time. In practice, it makes sense to implement some maybe very loose time-bounds, within which some life-signs of a WT should be received. If this time expires, some resources allocated to the WT within the BS can be freed.

- There is a contention-free possibility for data transmission and querying a WT's state.
- Bandwidth assignment by a central station (BS) can be more efficient than decentralized approaches, since potentially more knowledge about pending requests is available at the BS.
- If no transmission errors occur, they allow for deterministic system behaviour and thus for guaranteeing time bounds.
- The BS always knows which WT is transmitting. This knowledge can be explored with adaptive antenna array technologies [152].

Polling schemes are chosen for their potential to bound the maximum medium access times under any load conditions, not for the sake of mean delay or throughput, where other schemes may be more appropriate. Clearly, polling schemes have disadvantages:

- They introduce a single point of failure, namely the BS. Hence, appropriate redundancy schemes are needed. This kind of problems is addressed in detail in [154, 129].
- Polling-schemes do not scale easily to a large number of stations. However, for the small number of stations as in PROFIBUS LANs this concern is not relevant.
- The overhead needed for querying or polling the WT's.
- Polling WT's with no packets increases delay for nonempty WT's.
- If membership changes often due to mobility, then registration and deregistration overhead can become significant. In addition, a registration delay is introduced.

In designing polling-based MAC-protocols there are several degrees of freedom:

- polling-sequence and decisions on when to perform retransmissions;
- stations involved in data transmission and the polling process;
- variation of transmit power within the legal constraints;
- framing and coding schemes (FEC, interleaving, packet duplication);
- variation of modulation scheme;
- and as a meta-method: adaptivity, based on feedback about the channel behavior [26].

This thesis focuses on the first two "control knobs" and their potential to combat channel error conditions. They are tightly coupled to the concept of polling. The other knobs have already shown their capabilities elsewhere, and it is an issue for further research to investigate their behavior when combined with polling protocols.

In the design of polling schemes the error behavior of wireless links, as presented and modeled in Chapter 6 should be taken into account. This is true specifically for the phenomenon of (bursty) packet losses, which could a priori not be eliminated by adding redundancy to a packet's MAC-header or data part, as they happen before in the PHY header. But clearly, also bit errors and in general the time-varying nature of the wireless link should be taken into account.

In this chapter we present a specific k-limited round-robin protocol (Section 7.2.1). We use a simulation approach, to compare the realtime performance (see Section 4.3.1) of round-robin and the PROFIBUS protocol for different load models and error models (Section 7.3.2). It is shown that the round-robin protocols have much better realtime performance for error models exhibiting some burstiness. One reason for this is that polling avoids a permanently active ring maintenance mechanism. Instead a mechanism is used, where terminals have to register once at the base station, and the registration can be maintained with a soft-state approach with timeouts in the range of minutes. Stated differently: in the PROFIBUS protocol a station can get lost from the ring in every token cycle. With a registration scheme the "LAN membership loss opportunities" can be much reduced, or even eliminated in the case of a fixed topology and one-time registration. Hence, a major source of problems is removed.

In addition, three different modifications for further improvement of realtime performance are proposed (Sections 7.2.2, 7.2.4 and 7.2.3):

- The functional repolling approach, influencing the polling sequence and decisions on when to perform retransmissions. With this scheme the burstiness of bit errors on the wireless channel is addressed by trying to find a good time for doing retransmissions.
- Two relaying approaches, *simple poll relaying* and *simple data relaying*, modifying the set of stations involved in a data transmission and the polling process. These approaches are targeted to combat longer-lasting packet losses on a specific channel between two WT's by letting other stations help in packet transmission. Hence, spatially different wireless channels come into play. The relaying approaches are nearly "orthogonal" to the basic round-robin and functional repolling protocols.

All the three approaches are to the author's best knowledge not treated in the literatur so far.

Each of these three approaches is compared with the basic round-robin protocol for its realtime performance, and it is shown that often significant gains can be achieved (Section 7.3.3).

The round-robin protocol and the modifications are combined with a proposal for a modified PROFIBUS link layer protocol, better adapted to the behavior of wireless links (Section 7.1). Some issues of the changes in semantics and the interoperability with the PROFIBUS protocol are discussed in Section 7.4.

The findings of this chapter can serve as a starting point for the design and specification of a full MAC- and link layer protocol stack for an integrated wireless PROFIBUS system.

7.1 System under Study

In this section we give a brief description of the system under study, including its basic topology, MAC-protocol framework, automatic repeat request (ARQ) protocols, and the channel models used. The system under study is designed to highlight the issues related to the realtime performance of different polling schemes under error conditions.

For the polling protocols only the framework is described here, the actual polling schemes are presented in Section 7.2. Some of the proposed protocols change the semantics of the PROFIBUS services. This is discussed in Section 7.4.

7.1.1 Description of the System

Topology, PHY, Cell membership

We consider a wireless-only scenario with one base station (BS) and a number N of wireless terminals (WT), together constituting a *cell*. This topology is fixed, the stations are not mobile. This scenario is simpler than the integrated scenario targeted for a wireless PROFIBUS (see Section 4.3), but it allows to concentrate on realtime performance issues over wireless links.

A fully meshed topology is assumed, i.e., all stations can hear each other. The maximum distance between any two stations is small (30-50 m, according to the typical size of a production cell). Therefore, the propagation delay is small and can be neglected. In larger cells we would have to take guard times into account. The question whether this assumption can be relaxed is discussed in Section 7.6. Furthermore, given the small cell size, it is reasonable to assume that there are no near-far effects. Hence, parallel transmissions have the result that no station receives a valid and decodeable signal.

All stations are equipped with an IEEE 802.11 DSSS compliant PHY. They are capable of performing *fast carrier sensing*: for detecting the channels state it suffices to receive a signal of sufficient strength for a short time (10-20 μs), and it is not necessary to have acquired bit synchronization before. Querying carrier sense information from the radio modem takes negligible time. Hence, if any station within a cell transmits, all other stations know this almost immediately.[1] To avoid wrong interpretations of signals from neighboring cells, careful frequency planning is needed in a fabric hall.

All WT's are considered to be active stations in the sense that they require the right to initiate transmissions from time to time. Data transmissions are only amongst the WT's. In the simplified scenario adopted here, the BS is looked at only with respect to its role in medium access control, in the integrated scenario it serves also as a source or sink of data. The WT's have distinct station addresses, each of one byte length. Conceptually, a WT consists of a PHY, a MAC-entity, and a link-layer entity. The link-layer entity offers a link-layer interface to upper layers (the same as PROFIBUS). Furthermore, to each WT a number of traffic sources generating *requests* can be attached. These sources are on top of the link-layer interface.

The BS maintains a list of the WT's which are member of its cell and which have to be polled. To become a member of this list, a WT has to register itself once at the BS. This is done using special registration frames on a separate (physical or logical) channel.[2] There are two ways for a WT to get removed from the list: it removes itself by unregistering, or the BS deletes it after it has not received any signal from this mobile for a long time. However, since the topology is assumed to be fixed, it is reasonable to assume that one-time registration suffices and that this has been already done at system startup.

As a simplification, the necessary overhead for the registration mechanism is not considered. In fact, it can be argued that this overhead can be made low. To back up this claim, we propose a scheme: let

[1] In fact, the Harris/Intersil PRISM chipset [73] used throughout Chapter 6 allows to couple the CCA (clear channel assessment) signal directly to the received signal strength. Hence, it is reasonable to assume this feature.

[2] A possible approach are interspersed random access slots, which the WT's access with a slotted ALOHA protocol.

the BS issue every T_{RASlot} seconds a random access slot solely for registration purposes. If the address space is small (as in the PROFIBUS), the BS can resolve collisions in this slot by invoking a binary search algorithm over the address space, using special random access slots with a restricted address range. If the overall arrival rate is low (say, $\leq$ one arrival per minute), the collision rate will also be low (depending on T_{RASlot}), and the binary search algorithm is invoked rarely. If T_{RASlot} takes values of 50-100 ms, then the short random access slots will cost only a small fraction of bandwidth. With these settings a low degree of mobility can be accommodated.

Frame Structure and Framing

There are two types of data traffic: low priority and high priority, having the same meaning as for the PROFIBUS: the high priority traffic is used for asynchronous transmission of safety critical data (like alarms), all other data types (e.g., periodic process data, file transfers) belong to the class of low priority traffic. However, even if the high priority traffic is asynchronous, it can occur in batches (or *alarm storms*), which may arrive at several stations in parallel. We are only interested in the timing behaviour of the high prioritiy traffic, since this is typically used for safety-critical data. The low priority traffic serves as background traffic.

There are two classes of frames: (several) control frames and a single `data` frame type. All frame types are transmitted using 802.11 DSSS PHY PPDU's (see Section 3.2.2). A PPDU consists of a *PHY header* and a *PHY data part*. The PHY header consists of a PLCP preamble of length 128 bits and a PLCP header length of 64 bits, all transmitted in BPSK mode. Hence, the PHY header is of 192 bits or 192 μs length. The PHY data part is transmitted with QPSK modulation. It consists of a MAC header and optionally a MAC data part, carrying the user data. The MAC header has fixed structure. It is assumed to be of eight bytes length, which is sufficient to accommodate several, frame-type dependent fields (e.g., source and destination address, a tag field describing the frametype, several flags [indicating priority, alternating bit protocol information or the method of data encoding], queue lengths of low and high priority request queues[3], a length field [indicating the number of data bytes in a `data` frame]). Furthermore, we assume that this MAC header contains a separate header checksum. The checksum is assumed to be perfect, i.e., bit errors in the MAC header are always detected. In fact, with proper choice of the checksum algorithm the probability that bit errors remain undetected can be made very low. Hence, it is reasonable to assume that the influence of this case is negligible.

Control frames carry only the MAC header, the `data` frame can carry up to 255 bytes of data. Hence, it allows to embed a PROFIBUS variable length telegram of full length. No segmentation and reassembly scheme is assumed.

We assume that all stations detect the end of a frame within very short time $< 10\mu$s after its last bit. This corresponds to the SIFS of 802.11 with DSSS (see Section 3.2.3), where within this time the transmission of an 802.11 immediate ack must have been started. By this requirement also the modem turnover time between receiving and transmitting modes is upper-bounded by 10 μs.

Every `data` frame is equipped with a checksum, covering only the data part and not including the MAC header. After proper reception the receiver has to send an immediate acknowledgement, using the `ack` frame (belonging to the class of control frames). For proper comparison with the PROFIBUS SDA service (see Section 4.1.3) the `ack` frame carries no data. As for PROFIBUS, the `ack` frame

[3]These can be used by the BS for scheduling decisions.

can indicate both positive and negative acknowledgements (e.g. on the occasion of buffer shortage). However, for simplicity for both the PROFIBUS protocol and the polling protocols we assume that only positive acks are sent.

ARQ-Protocol and WT-Behavior

The PROFIBUS uses a variant of the alternating bit protocol (ABP) with a bounded number of retransmissions (protocol parameter *max_retry*) as ARQ protocol (see Section 4.1.4). The ABP provides protection against losses and duplicates at the receiver.

We introduce two modifications to this protocol: the PROFIBUS atomicity restriction (see Section 4.1.3) is removed and for every priority class a separate instance of the alternating bit protocol is maintained. Hence, given a number of N WT's, every WT maintains $2(N-1)$ instances of the protocol, namely, for every possible target and every priority. The rationale for this is to allow high priority requests to preempt low priority requests currently in work. In the nonpreemptive PROFIBUS scheme the latter may block high priority requests in case of a longer period of transmission errors / retransmissions. The possible preemption violates the PROFIBUS atomicity property, for a discussion of changes in semantics see Section 7.4.2. Since for every possible target address and every priority a separate ABP instance is maintained, the preemption causes no confusion of the alternating bits used. How this protocol can be integrated with the PROFIBUS ABP protocol is briefly discussed in Section 7.4.

Furthermore, two different *max_retry* parameters for low and high priority data are distinguished. For the high priority requests we assume large values of the *max_retry* parameters, e.g., $max\_retry > 20$. This is due to the high reliability requirement for high priority requests.

Two important rules for the behavior of a WT are:

- For both priority classes separate queues are maintained. Upon every poll, a *local scheduler* selects the queue to serve. Within this work we consider only a *simple local priority scheduler*, where always the high priority queue is served, if it is nonempty. However, other types of schedulers can be used, too.
- The WT observes the perceived poll intervals. If it is not polled for a certain time, it generates appropriate indications or management signals for its upper layers, indicating the outage condition.

Channel Models

For investigating the realtime performance of the polling-based protocols and the PROFIBUS protocol, a set of channel models with different degrees of statistical sophistication is used.

The channel models take the phenomena found in the measurements reported in Chapter 6 into account, namely bit errors and packet losses. All other phenomena (ghost packets, bit shifted packets and so forth) are treated as packet losses, see Section 6.5.1. Bit errors and packet losses occur independently from each other, see Section 6.8.3.

The basic assumption is that between every pair of station there is a separate and independent channel w.r.t. its error behaviour, see Section 6.8.3.

We assume that the channels between different station pairs are stochastically independent. If they are in addition stochastically identical (i.e., follow the same stochastic model), this is called the *homogeneous* case, otherwise we have the *inhomogeneous* case. In the inhomogeneous case every pair of stations can have its own channel model: given a set of k different channel models (each including packet losses and bit errors) and N stations with $\frac{N(N+1)}{4}$ (due to the assumed symmetry) channels between them, each of the channels is randomly selected from the k channel models (all channel models are equiprobable).

For a single channel four different channel models are used: the *independent* model, the *Gilbert-Elliot model*, a *Semi-Markov* model (using lognormal distributions for the state holding times) and a *complex* model, described below. Except from the complex model, all models are parameterized from trace 24 of the **factorial** measurement, assuming a burst order $k_0 = 150$ for the bit error submodels and a burst order $k_0 = 1$ for the packet loss submodels.

The independent, Gilbert-Elliot and Semi-Markov model are homogeneous channel models, the complex model is an inhomogeneous model.

For the independent model, the bit errors are assumed to be independent with a fixed bit error rate (BER) of 0.00037, while packet losses occur independently with a fixed packet loss rate (PLR) of 0.1099.

In the Gilbert-Elliot model both the bit errors and the packet losses are assumed to follow a two state markov chain model, respectively, as discussed in Section 6.7.1. With respect to bit errors, the mean error burst length is 114.8, the mean error free burst length is 11514.1, and the bit error rate during an error burst is 0.0375 (independent bit errors). For the packet loss model the following values are used: mean error burst length: 2.68, mean error free burst length: 21.74, and the packet loss rate during a packet loss burst is 1.

In the Semi-Markov model both the bit error and packet loss model are assumed to follow a two-state model, however, with the state holding times drawn from a (quantized) lognormal distribution. For the bit error model the mean error burst length is 114.8, the variance of the error burst length is 72237 (giving a coefficient of variation (CoV) of 2.341), the bit error rate within an error burst is 0.0375. The mean error free burst length is 11514.1, and its variance is 442520151449 (giving a CoV of 57.775). For the packet loss model the mean packet loss burst length is 2.68, its variance is 24.53 (giving a CoV of 1.848) and the packet loss rate during a burst is 1. The packet loss free bursts have a mean burst length of 21.74, a variance of 119492.06 (giving a CoV of 15.9).

For the complex model we use different traces: trace 24 of the **factorial** measurement, trace 21 of the **factorial** measurement, trace 1 of the **longterm1** measurement, and trace 17 of the **longterm1** measurement (QPSK, without scrambling). Some first order parameters of these traces are summarized in table 7.1. Each trace was taken as a single channel model, composed of one packet loss and one bit error submodel. These four traces together form the inhomogeneous complex model. For all traces both the packet loss and bit error submodels were expressed as bipartite models. The bit error submodels use 25 good and bad states, respectively, the packet loss submodels use 5 good and 5 bad states. For the bit-errors the number of 25 states has in Section 6.7.3 shown to deliver good predictions for selected performance measures. The packet loss submodel was chosen with fewer number of states due to the much more limited range of burst lengths.

It should be noted that the complex model is not comparable to the others, since it produces a different mean BER, due to inclusion of traces with almost no bit errors. The overall PLR of ≈ 0.13 is also

	trace 24 (k_0 : 150)	trace 21 (k_0 : 150)	trace 1 (k_0 : 150)	trace 17 (k_0 : 150)
MBER	0.000370	0.000189	0	0
mean EBL	114.8	157.4	-	-
CoV EBL	2.341	1.967	-	-
max. EBL	6529	4482	-	-
mean EFBL	11514.1	17335.4	-	-
CoV EFBL	57.775	19.439	-	-
PLR	0.1099	0.1	0.3193	0.004
mean PLBL	2.68	1.88	4.0	1.053
var PLBL	24.52	4.02	24.67	0.05
CoV PLBL	1.845	1.066	1.239	0.212
max PLBL	55	30	73	2
mean PLFBL	21.73	16.92	8.55	265.6
var PLFBL	119492.06	10240.69	11795.49	3381778.32
CoV PLFBL	15.9	5.98	12.71	6.92

Table 7.1: Summary statistics of selected traces constituting the complex error model (EBL=error burst length, EFBL=errorfree burst length, MBER=mean bit error rate, PLR=packet loss rate, PLB=packet loss burst)

different.

7.2 Polling-based Protocols

In this section we present the polling-protocols used for further study. The first one is a k-limited round-robin protocol with a simple local priority scheduler, described below. This protocol serves as a baseline protocol, in the sense that both the original PROFIBUS protocol and also the proposed improvements are compared to this protocol. The choice of k-limited round-robin is reasonable, since it has the property of a strictly bounded time between two trials for polling a fixed WT.

Among the different means to combat bit errors on wireless links two popular ones are FEC [97] and postponing schemes like the one in [14, 104]. In the postponing schemes a retransmission of an unacknowledged `data` frame is not performed immediately, instead it is delayed until a "better" time. The idea is that for bursty errors an immediate retransmission will fail with high probability, and instead the bandwidth can be used to serve other stations. This kind of scheme can be implemented by influencing the sequence in which the BS polls the WT's. For investigating this approach we propose the *functional repolling* framework. However, in the course of preliminary investigations we have observed that this approach should be applied only to high priority data frames, since otherwise repolling of low priority requests may block high priority requests.

The round-robin and functional repolling protocols are "orthogonally" combined with two approaches called *simple poll relaying* and *simple data relaying*, which aim to overcome (bursty) packet losses on a wireless channel.

The following protocol description makes use of certain frametypes. For reference, these are summarized in Table 7.2. Some further preparing remarks are:

Frame type	min. parameters	protocol	meaning
`poll`	k, a	all	WT a may handle up to k data slots
`data`	a, b, data	all	WT a sends data to WT b
`ack`	a, b	all	WT b ack's data frame of a
`explicit-poll`	k, a	rrk+SPR, frk+SPR	like `poll`, but a must answer
`null`		rrk+SPR, frk+SPR	generated by empty WT on `explicit-poll`
`execute-explicit-poll`	k, a, b	rrk+SPR, frk+SPR	BS asks WT b to send `explicit-poll` frame to a with parameter k
`executed-ack`	status	rrk+SPR, frk+SPR	poll-relayer gives feedback to BS
`invited-poll`	a	frk, frk+x	BS grants a one data slot (special handling)

Table 7.2: Different frametypes of polling-based protocols, with minimum parameters, x is either SDR or SPR

- Time is maintained by the BS not as an absolute value. Instead the BS keeps a variable t_c counting the number of *poll transactions* since system startup. Hence, hence the time is discretized. A poll transaction starts with a `poll` frame, `invited-poll` frame, `explicit-poll` frame or `execute-explicit-poll` frame (all explained below) and ends with the frame before the next of these poll frames.
- In the following, we will use the term *time unit* for a time duration of 64 μs, according to 64 bits transmitted with BPSK. Hence, the PHY header of an 802.11 DSSS PPDU consists of three time units.
- The convenient term *slot* denotes the exchange of a `data` frame including the following `ack` frame.

7.2.1 k-limited Round-Robin

The k-limited round-robin protocol belongs to the class of cyclic polling protocols (as opposed to Markovian or table based protocols, see [64, chap. 9]). The WT's are served in round-robin fashion, and the BS schedules a maximum of $k \in \mathbb{N}$ contiguous slots for a single WT before proceeding to the next one. The k value for the k-limited round-robin protocol and the functional repolling framework (next section) is called *round-robin bound*.

In detail, the protocol works as follows: the BS sends a `poll` frame to WT a, indicating the maximum number k of contiguous slots allocated to a. Hence, a is allowed to send k `data` frames including `ack` frames. If WT a has data to send, it starts immediately to do so by sending a `data` frame to WT b. Furthermore, WT a maintains a *slot counter*, by initially setting it to k and decrementing it after every transmitted data frame. When b properly receives the `data` frame, it sends immediately (within half a time unit) an `ack` frame. When a receives the `ack` frame, it checks, whether there is more data to send and the slot counter is greater than zero. If so, a sends the next `data` frame. If a has no further frames to send, it keeps quiet. After the `data` frame a waits one time unit and then senses the medium. If b has received the `data` frame properly, it transmits a `ack` frame within half a time unit. Hence, a will sense a busy medium. WT a now awaits the transmission outcome and proceeds either proceeds with the next frame (new frame or retransmission) or stops sending, if its slot counter reaches zero. When b does not transmit an `ack` frame, a determines this by sensing an idle medium. In this case WT a waits for a second time unit and performs another carrier sense operation. If WT a senses a carrier, it concludes that the BS has taken control over the channel. WT a then goes back in its initial state, waiting for the next `poll` frame from the BS. If there is no carrier and the slot counter is greater than zero, WT a continues with sending data frames or gives up, if its slot counter reaches zero.

WT a runs a *simple local priority scheduler*: when deciding which frame to send, a always selects the high priority queue for service if it is nonempty. This has the consequence, that the atomicity property is violated for low priority frames. Furthermore, since k has not necessarily any relation to the *max_retry* parameter, the atomicity property can also be violated for high priority frames: if $k <$ *max_retry* a single request can span multiple poll cycles.

The behaviour of the BS is as follows: after sending the `poll` frame it waits one time unit and then senses the medium. If the WT has not started to make any transmissions, the next WT is served (possible errors in fast carrier sensing are not modeled). Otherwise, the BS simply waits for the medium being for at least three time units, which means that the WT has stopped any activity. Then the next WT is served.

It is important to point out that with the protocol described so far the WT's need no prior (preconfigured) knowledge about k, since the BS transmits this value with every `poll` frame. Hence, it is no problem to implement other bandwidth assignment policies within the BS than to just send the same k value to every station.

The decision to keep a quiet if it has no data to send introduces a problem: the BS cannot distinguish between cases where a has no data and where a has not received the `poll` frame correctly. The reason to choose this option is efficiency, since requiring a to send an empty or `null` frame in case of empty queues costs at least one PHY overhead and a MAC overhead, which takes $192+x$ μs in a setup using the IEEE 802.11 DSSS PHY. An alternative would be to let the WT send a short burst or jamming signal, which is significantly shorter than any frame (say, 20-40 μs), and thus can be detected by the BS upon the burst length. This is discussed in Section 7.6.

The k-limited round-robin protocol was chosen as a baseline protocol for the following reasons:

- It is simple and deterministic.
- It provides fair bandwidth distribution, and prevents stations from starvation.
- It has an upper bound on cycle times.

These properties follow immediately from the protocol description and are valid in case of no errors.

7.2.2 Functional Repolling Framework

In this section we present the *functional repolling framework*. It provides a means to express when to perform retransmissions of high priority requests.

Definition. *A repoll function is a function*

$$f \quad : \quad \mathbb{N} \longrightarrow \mathbb{N}_0 \cup \{-1\}$$

with $f(i) = -1$ *for* $i > \text{max\_retry}$*,* $f(i) \geq i-1$ *and* f *monotonically increasing in the interval* $[1, \ldots, \text{max\_retry}]$*. The repoll function indicates the number of poll transactions* $f(i)$*, after which the* i*-th retransmission takes place. This number is expressed with respect to the time* t_0 *of the time where the first trial of transmitting a request fails.*

The very basic idea of functional repolling is to let the BS keep track of necessary retransmissions and to schedule them appropriately. The word "appropriately" only makes sense with respect to the error conditions on a wireless link: if it is known to be bursty and a transmission error occurs on a channel, it is reasonable to postpone the retransmissions and to serve other packets/WT's meanwhile. For expressing "appropriate" time instants for retransmissions, so-called "repoll functions" are used.

Basically, the BS maintains a k-limited round-robin protocol. However, the round-robin protocol is applied only to a subset of all stations, namely, the members of the *poll list*. These are served by sending regular poll frames as in the k-limited round-robin case. The remaining WT's are member of another list, the *repoll list*, discussed below.

The BS tries to capture all frames generated by the WTs. Specifically, consider the case that the BS has sent a `poll` frame to WT a and a has sent a `data` frame to b. Let us assume that b perceives an error. In this case b generates no `ack` frame, which both a and the BS can determine by sensing the medium one time unit after a's `data` frame. The BS performs a *repoll eligibility test*. If this test is passed successfully, the BS removes a from its poll list and enters it into its *repoll list*, while simultaneously resetting the *retry counter* $rc(a)$ associated to a. If this test is not successful, a remains in the poll list.

The repoll eligibility test distinguishes between two cases: if the BS has successfully received a's `data` frame, the test is passed, if the frame is of high priority, for a low priority frame the test fails. If the BS has not received the frame (or the frame's MAC header) successfully, it is a matter of policy, whether the frame is assumed to be of low priority (this is likely the typical case) or of high priority (which is the conservative assumption). It is assumed that it is of high priority.

The BS uses the repoll function f as follows: in the round-robin mode, when b does not acknowledge a `data` frame of a and the repoll eligibility test was passed, the BS sets $rc(a) = 0$ and evaluates $f(rc(a)) = f(0)$. For every trial, if $f(rc(a))$ gives a negative value, the retry counter $rc(a)$ is reset to zero, and a is deleted from the repoll list and re-inserted into the normal poll list. If it returns a nonnegative value, the station is inserted into the repoll list. The repoll list is a list of stations to repoll. Each list element consists of a station's address and the time instant (expressed in poll transactions) when the repoll should take place (denoted as *repoll time*). The list is sorted by this wait values, and new elements are inserted accordingly.

The BS now operates as follows:

- When the repoll-list is nonempty, but the repoll time of the first element is larger than the current time, a station of the poll list is served. As a special case, if the poll list is empty and the repoll time is greater than the current time, the BS sends an `invited-poll` frame to an arbitrary WT, allowing it to perform data transfers. However, it makes sense to not select an arbitrary WT, but one whose queues are nonempty. For determining this the BS uses the queue length information it captures from all WT's MAC headers. The targeted WT has the freedom to perform another trial on the blocking high priority request, to choose another high priority request targeted to a different station or to choose a low priority request. Selecting another request from the high priority queue (to overcome the head of line blocking) is only possible when the requests destination address is different from that of the blocking request (because of the alternating bit) and if a possible change in the sequence of confirmation primitives as compared to the sequence of request primitives can be tolerated. If the WT chooses to serve a high priority frame, this has no effect on the repoll list. If the request was successful, it is removed from the queue.

- When the repoll list is nonempty and the repoll time of the first element is smaller or equal than the current time, the first element is removed, and the BS sends a `poll` frame to the corresponding station a. This frame announces a number of k slots to a. If the WT does not start to transmit any frame, it is inserted into the poll list. In the other case, the BS performs a repoll eligibility test on the outcome. If this test is passed, the retry counter $rc(a)$ is incremented and station a is re-inserted into the repoll list with a value according to $f(rc(a))$. If the test fails, $rc(a)$ is set to zero and a is inserted into the poll list.
- When the repoll list is empty, the round-robin protocol is used.

By these rules, a station a is at every time only member of a single list, and in every list a occurs only once.

The need to keep the repoll list sorted, introduces an $\mathcal{O}(N)$ operation, hence, the protocol does not scale well with increasing number of stations. However, in the specific case of PROFIBUS this is no problem, since the number of stations is restricted to 127.

Some sample repoll functions are:

- immediate repolling: $f(i) = i - 1$
- exponential repolling: $f(i) = 2^i - 1$ (the first repoll is immediately)
- delayed exponential repolling: $f(i) = 2^i - 1 + n_0$.

(here for simplicity the case $f(i) = -1 (i > \mathit{max\_retry})$ is not written out).

7.2.3 Relaying

The idea of relaying is targeted towards combatting packet losses. As we learned from our measurements, there can be very long periods of sustained packet losses on the channel $C(a, b)$ between WT a and b. One possibility to overcome this problem is to let another station c relay the communication between a and b, hoping that the channels $C(a, c)$ and $C(c, b)$ are currently in a good state.

We propose two different schemes, both with the BS as the main actor.

Simple Data-Relaying

Consider the case that WT a sends a `data` frame to WT b over the channel $C(a, b)$, and WT b does not decode this frame correctly, hence, it does not transmit an `ack` frame. The BS and WT a can determine this by performing carrier sensing one time unit after WT a has finished its `data` frame.

However, if the channel $C(a, z)$ from a to the BS z is not distorted, BS z has successfully captured the `data` frame. In this case (and if it was of high priority), the BS may decide to send the `data` frame on the channel $C(z, b)$ to b immediately after it recognized the missing ack frame. If b sends an ack frame, the BS tries to capture it and sends it again. Hence, a may receive two times the same ack frame, once on the channel $C(b, a)$ and once on the channel $C(z, a)$. This requires the ability of a to detect and ignore duplicate ack frames. Furthermore, a does not count z's data frame as a separate retransmission trial.

By the rules for the k-limited round-robin protocol, WT a senses the medium two times after finishing its data frame: after one time unit it can determine whether WT b has answered, after two time units it can determine, whether the BS has (re-)transmitted the data frame. If so, WT a retires from the medium and awaits the next poll from the BS. The BS has different policies to choose from:

- The BS continues with a's successor.
- The BS polls a again with the remaining number of cycles available to a.

We have chosen the second alternative.

This approach is denoted as *simple data relaying (SDR)*. It can be easily implemented as an orthogonal feature to both the k-limited round-robin scheme and the functional repolling framework. Furthermore, it does not need any history information maintained by the BS. Likely this scheme delivers the best results, if the involved channels are not positively correlated. The term "simple" comes from the restriction to let only the BS play the role of a data relayer. If this restriction is removed, however, some synchronization between possible relayers seems necessary. A possible approach can use local timers at every WT, with the timeout set according to a WT's address. If a WT successfully captures a data packet, it starts the timer. If the timer expires and there is no activity on the medium, the WT transmits the captured frame. A similar scheme with local timers is used for triggering retransmissions in the scalable reliable multicast approach reported in [48]. However, such schemes have the disadvantage that they put additional complexity into the WT's.

It is a matter of policy, whether SDR is applied to every every frame, or, in the case of the functional repolling framework, only after a certain number of repolls. We assume that it is applied to every frame.

Simple Poll-Relaying

Definition. *A memory-loss function* $m(\cdot)$ *is a function*

$$m : \mathbb{R}_0^+ \longrightarrow \mathbb{R}_0^+$$

which is monotonically decaying and for which $\lim_{t\to\infty} m(t) = 0$ *holds.*

A critical issue not resolved by the SDR scheme is the case of a heavily distorted channel between the BS z and a fixed WT a. This may lead to the situation that a does not receive a poll frame for long time. Since WT a signals empty queues by keeping quiet, the BS cannot tell whether a's queues are empty or the channel to a is distorted. Hence, when polling a the BS should insist on getting a short answer from time to time, in order to be sure that a's silence is not due to a corrupt channel $C(z, a)$. This is done using an explicit-poll frame, which requests a to send a data frame or a short null frame to the BS.[4]

[4]Special case: a sends a null frame, but the BS does not capture it, so it cannot tell whether it was a data frame. Then the BS has to wait for a sufficiently long medium idle time.

In general, the BS tries to capture all frames, and for every station a keeps the timestamps of the last n successfully received frames / MAC headers (transmitted over the channel $C(a, z)$, which is assumed to be the same as $C(z, a)$). If for a certain station even after sending `explicit-poll` frames this timestamp is older than a certain threshold T_{Quiet}, the BS assumes that its channel to a is currently bad. The question is: how can a be polled despite the bad channel $C(z, a)$?

The approach is very simple: the BS selects a station u (the *poll relayer*), which acts temporarily on behalf of the BS z. The BS z sends an `execute-explicit-poll` frame to u, which carries a's address as a parameter. If u properly receives this frame, it sends an `explicit-poll` frame to a.[5] WT a may then start to do a data transfer to an arbitrary station (including u), or it sends a `null` frame. In both cases u waits for a finishing its transfer and then sends an `executed-ack` frame back to BS z, indicating the success that u has perceived for a's data transfer (u is required to check for `ack` frames on a's `data` frames). This feedback information allows the BS in the functional repolling framework to execute the repoll scheme for station a. The poll relayer u is not required to perform the SDR algorithm on behalf of the BS (however, this can be considered as an interesting extension). If the BS gets no `executed-ack` frame from u (as can be determined by sensing no signal for a certain time), it can choose to select another station v as poll relayer, however, within this work it is assumed that the BS does not do so.

A central question is how the BS selects the poll relayer u. To make a good selection, it is useful that the BS collects *channel history information* for each channel $C(i, j)$. Specifically we propose the following scheme. Be $n \in \mathbb{N}$ the *history depth*, which indicates the amount of history information stored for each channel. For every single channel $C(i, j)$ (with $i, j \in \{1, \ldots, N\}$) the BS stores n *transmission outcomes* $x^1_{i,j} \ldots x^n_{i,j}$ (with $x^\nu_{i,j} \in \{-1, 1\}$ for all $\nu \in \{1, \ldots, n\}$) along with the corresponding *timestamps* $t^1_{i,j} \ldots t^n_{i,j}$. If x is a transmission outcome, the value $x = -1$ denotes an *unsuccessful transmission*, and the value $x = 1$ denotes a *successful transmission*. The *channel quality* $Q(i, j, t)$ of channel $C(i, j)$ at time t is then estimated as follows:

$$Q(i, j, t) = \sum_{\nu=1}^{n} x^\nu_{i,j} \cdot g(t - t^\nu_{i,j})$$

where $g(\cdot)$ is a memory-loss function. With $g(\cdot)$ one can express that older entries are not so important than fresh ones.

The BS chooses the poll relayer u^* such that

$$\min\{Q(z, u^*, t), Q(u^*, a, t)\} \geq \min\{Q(z, u, t), Q(u, a, t)\}$$

for all $u \in \{1, \ldots, N\}$ holds, i.e. it looks for the "route" with the best minimum quality.

The history is updated by the following rules:

- When the BS z successfully captures a frame from a to b at time t, it performs a *history update procedure* on $C(a, z)$. During the history update procedure, the BS removes the oldest entry from $C(a, z)$ and, after renumbering, adds a new entry $x^n_{a,z} = 1$ and $t^n_{a,z} = t$ to the channel history.[6] The same is done for the channel $C(z, a)$ by the channel symmetry assumption. When

[5]This immediately starting activity allows the BS to check whether u has received the `execute-explicit-poll` frame. If u does not react, the BS seeks for another station v. However, the number of trials for selecting a poll relayer is bounded.

[6]This can be easily implemented as an $\mathcal{O}(1)$ algorithm using a ring buffer.

the BS captures only the MAC header of a `data` frame without errors, the transmission outcome $x^n_{a,z}$ is set to -1.

- If the BS polls station a, and senses a signal from a but does not capture a's frame, it performs a history update procedure for channels $C(a, z)$ and $C(z, a)$ with transmission outcome -1.
- If WT a sends a `data` frame to WT b, which the BS can capture, and if this frame is followed by some signals in the time where the BS expects the `ack` frame, the BS performs a history update procedure for the channels $C(a, b)$ and $C(b, a)$, both with a successful transmission. If WT b sends no `ack` frame, the BS performs a history update procedure for the channels $C(a, b)$ and $C(b, a)$, both with unsuccessful transmission.

It is clear from this description that the observed channel state can be inaccurate, e.g., in the last case it may happen that a does not receive b's `ack` frame successfully. However, this scheme has the advantage that all burdens are put onto the BS while the WT's have to do nothing. Specifically, there is no need to transmit history information from the WT's to the BS. Hence, the WT's implementation is comparably simple.

7.2.4 Adaptive Functional Repolling

The functional repolling framework allows to implement an arbitrary repoll policy by selecting the repoll function f. For example, if the channel characteristics are known to correspond to a Gilbert/Elliot model, it makes sense to postpone the retransmission for a short while (say, mean burst error burst length plus two times standard deviation), which can be easily expressed. However, wireless channels have an inherently time-varying nature, both with respect to bit error rates and packet loss rates. Hence, a fixed f may not be a suitable choice for all situations.

In [46] "Meta-MAC" protocols are introduced. The basic idea is simple and elegant: a station contains not only one MAC instance, but several of them, running in parallel. These can be entirely different protocols or the same protocol, but with different parameters. However, only one protocol is really active at a given time in the sense that its decisions are executed, the other instances decisions are only recorded. The active protocol is chosen based on suitable history information about transmission outcomes. For each given protocol it is evaluated, how "successful" the protocol would have been given the outcomes in the history. Based on this ranking a new protocol is chosen.

A similar idea can be adapted for choosing a proper f from a set of given f's. The "success" of each function can be evaluated for the known channel history information. The hope is that the observed, historical channel conditions are a good prediction at least for the "near" future, and thus the selected f is doing good.

Specifically, the proposed scheme can be described as follows: be $\{f_1, \dots, f_m\}$ a set of m repoll functions as defined in Section 7.2.2. To every channel $C(i, j)$ with $i, j \in \{1, \dots, N\}$ the BS maintains a channel history as described in Section 7.2.3. Furthermore, to every channel a *current repoll function* $f^*_{i,j} \in \{f_1, \dots, f_m\}$ is associated.

The main building block is to evaluate the "success" a given repoll function f would have reached on a channel $C(i, j)$ given its channel history $x^1_{i,j}, \dots, x^n_{i.j}$ and $t^1_{i,j}, \dots, t^n_{i,j}$ (for simplifying notations the indices i and j are dropped). First of all, we motivate and specify the notion of "channel quality": consider the example $n = 10$ and $x_1 = 1, x_2 = 1, \dots, x_4 = 1, x_5 = -1, x_6 = 1, \dots, x_{10} = 1$ and

$t_k = k(k \in \{1, \dots, 10\})$. Here, within ten trials only one failure occurs, which should not be interpreted as a "bad channel period", but as a "temporary slip". Hence, it makes sense to "smooth" the channel history using low-pass filtering. A simple means to do this is to replace the channel history with a windowed moving-average version. Specifically, let W be the *window size* and $c := f(max\_retry)$ be the maximum number of poll transactions over which f has any influence. Then we use the following moving average function considering the last c poll transactions as taken from time t:

$$d(k,t) = \frac{1}{W} \sum_{i=1}^{n} x_i \cdot \mathbf{1}_{[t-c-W+k,t-c+k]}(x_i)$$

where $\mathbf{1}_A(x)$ is the indicator function for the set A, i.e. $\mathbf{1}_A(x) = 1$ if $x \in A$, and $\mathbf{1}_A(x) = 0$ otherwise. The "quality" of f is then interpreted as f's ability to schedule the repolls during the "good" channel states:

$$Q(f(\cdot), t) = \sum_{i=1}^{c} d(i,t) \cdot h(c-i) \mathbf{1}_{\mathbb{N}}(f^{-1}(i))$$

where $h(\cdot)$ is an arbitrary memory-loss function as defined in Section 7.2.3. If we do not want to look back c poll transactions, but only $c' < c$, and see how f would have behaved "from $t - c'$ on", we use

$$Q(f(\cdot), t) = \sum_{i=c'}^{c} d(i,t) \cdot h(c-i) \mathbf{1}_{\mathbb{N}}(f^{-1}(i - c'))$$

A basic parameter for this approach is the update frequency T_{AR}, i.e., how often a new repoll function is determined for a channel.

7.3 Realtime-Performance Results

In this section first a comparison of the baseline k-limited round-robin protocol with the classical PROFIBUS protocol is presented (Section 7.3.2). The results indicate that indeed the approach to replace the PROFIBUS protocol by something else is promising. By these results the k-limited round-robin protocol is established as a good starting point for investigating further improvements, which is done in Section 7.3.3.

7.3.1 Method of Investigation

Realtime Performance of PROFIBUS

For evaluating the realtime performance of the PROFIBUS protocol the same simulator is used as in Chapter 5, however, with some adjustments:

- The stations use the wireless-type framing formats as described in Section 5.2, where PROFIBUS frames are embedded into 802.11 DSSS PHY PPDUs (see Section 3.2.2), hence, including the PLCP preamble and PLCP header.
- The channel error model was changed to incorporate the four models as described in Section 7.1.1, and the medium characteristics. In fact, both the PROFIBUS simulator and the Polling-simulator described below use the same code and parameters for the channel models.

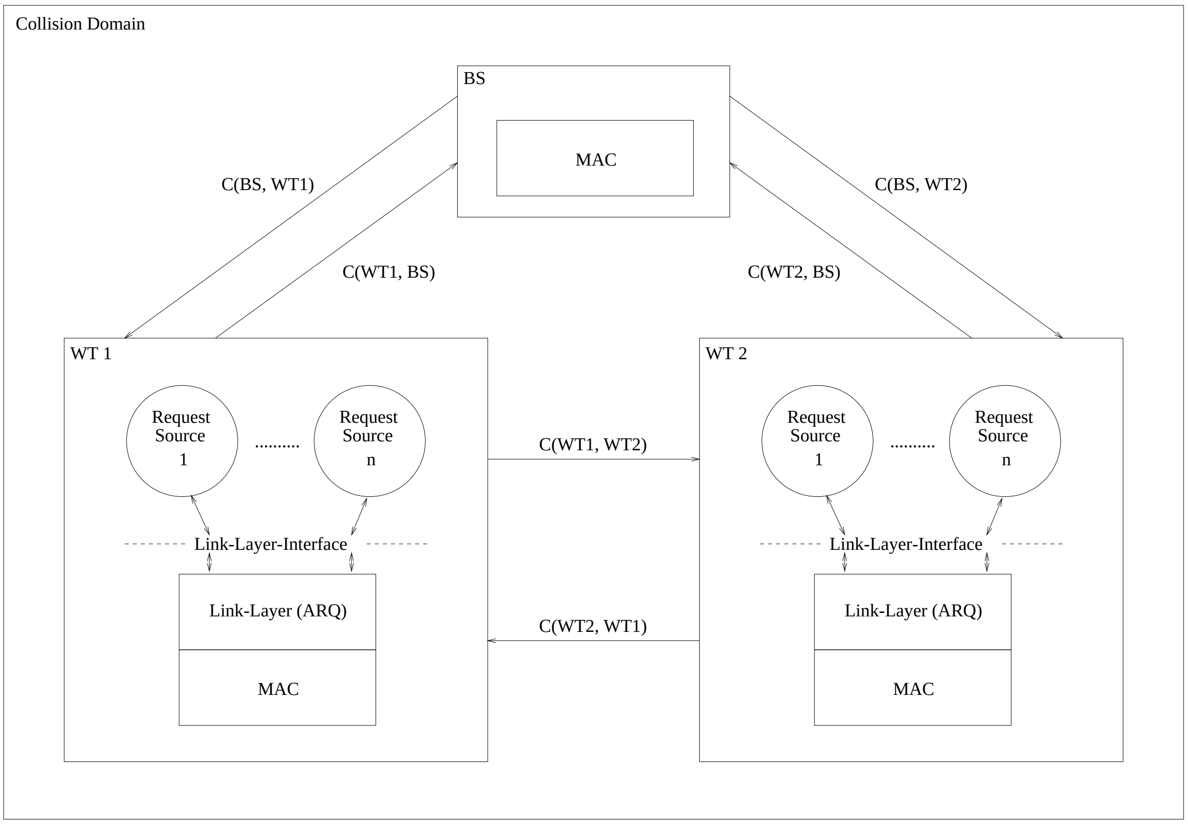

Figure 7.1: Logical structure of polling simulation model

The simulator uses the CSIM simulation library and provides a full implementation of the token-passing and ring-maintenance mechanism, the link-layer protocol and the link-layer interface (see Section 5.1.4). Aspects like management are not considered in the simulator.

Every PROFIBUS station has separate request sources, which issue SDA requests at the link-layer interface and accept the corresponding confirmation primitives. The request sources can generate low and high priority requests.

Realtime Performance of Polling Protocols

For investigating the polling-based protocols a simulation model was developed, written in C++ and using the CSIM simulation library [103]. This model is distinct from the PROFIBUS simulator. The simulation approach was preferred over probabilistic analysis, since even for the simplest cases the stochastic analysis of polling systems gets very involved.

The logical structure of the simulator is shown in Figure 7.1 for a setup of one BS and two WT's. The simulator covers the polling-based protocols as described in Section 7.2. A protocol for registration and deregistration is not considered, nor are management functionalities included.

Every WT can have several request sources for generating requests. A single source generates requests with a configurable priority, request size distribution, interarrival time distribution, and target address distribution. The link-layer instances in the WT's take the requests, enqueue them according to their priority, handle the ABP protocol and finally run the local scheduler every time the MAC entity signals the availability of a slot. The MAC entity within the WT's depends on the polling-protocol, but typically accepts one of the four poll frame types from the BS and acts accordingly. The MAC entity of the BS is responsible for polling the WT's, keeping the history information, executing the adaptive functional repoll protocol, etc.

All stations form a *collision domain.* This means that all stations can hear each other and that two signals transmitted in parallel will destroy the received signals for all stations. However, with respect to the error behavior, between every pair of stations two separate, yet symmetric channels are maintained, according to the models described in Section 7.1.1.

Comparability of Simulation Models

In both simulation models the framing is the same, since both embed their MAC frames into 802.11 DSSS PHY packets. Hence, both models consider a PHY overhead of 192 μs.

The sources generate in both models requests with a data length of 40 bytes, belonging to the acknowledged SDA service. The MAC header length in the polling simulator is eight bytes, the length of header and trailer in the PROFIBUS simulator is nine bytes. The interarrival times are computed such that in every model the load definition of the realtime performance measures (see Section 4.3.1) is satisfied, i.e., both simulators generate the same (error-free) link utilization. The sources of the polling models always generate requests, those of the PROFIBUS model generate requests only, if the attached station is member of the ring. This restriction was necessary to keep the simulators memory consumption stable, otherwise for many parameters the queues grow infinitely. Please note that this behavior is in favor of the polling simulator.

In both simulators it is assumed that there is no protocol processing time, and there is only a very small delay ($<$ 1 bit) between reception of a data frame and the corresponding acknowledgement. Actually, this is a good approximation, when taking todays fast microcontrollers into account.

Both simulators use the same channel error models, namely, those described in 7.1.1.

In summary, although the protocols in both simulators are different, they operate in the same load environment, error environment and PHY medium.

7.3.2 Comparison of Round-Robin with PROFIBUS

Both simulators were given the same load, namely the **smooth** loads with 10% and 50% background load. The set of varied parameters is shown in Table 7.3. The propagation delay is zero, both simulators use 2 MBit/s QPSK modulation. The PROFIBUS slot time T_{SL} was set to 400 μs, the station delay was chosen to be 100 μs (as in Chapter 5.2). The simulation time was always chosen to be 3600 simulated seconds, in order to be comparable with the results in Chapter 5.

For the PROFIBUS the setting of the *max_retry* parameter is interesting. For high values one can expect a low negative confirmation rate (see Section 4.3.1), possibly at the cost of blocking effects due to the non-preemptability of requests (including low priority requests). We have chosen to set this parameter for the PROFIBUS to *max_retry* = 20, in order to have the same level of reliability

Parameter	**Values**
Number of stations	2, 4, 8, 12, 16, 20
Target token rotation time (T_{TTRT})	0.005, 0.01, 0.015, 0.02, 0.04 s
gap factor	1, 2, 4, 6, 8
max_retry parameter rrk	20
max_retry parameter PROFIBUS	20
PROFIBUS improvements	on
channel models	independent, Gilbert-Elliot, Semi-Markov, complex
load models	smooth (10% low priority load), smooth (50% low priority)
round-robin bound k	1, 2, 4, 6
Bitrate	2 MBit/s QPSK

Table 7.3: Simulation parameters for comparison between PROFIBUS and k-limited round robin

Error model	**Max. number of negative confirms**
independent	0
Gilbert-Elliot	8
Semi-Markov	620

Table 7.4: Maximum number of negative confirmations over all N's for the different error models (PROFIBUS protocol, $T_{TTRT} = 0.005$s, gap factor = 1)

as for the polling protocols. In fact, we state without showing this in this thesis, that the realtime performance results show almost no difference for the two parameter values. This can be explained as follows. In Table 7.4 we show for the PROFIBUS protocol and the parameter setting with the highest degree of ring membership ($T_{TTRT} = 0.005$s and gap factor equals 1) the maximum number of negative confirmations, for each error model taken over all different values of N. For the independent and the Gilbert-Elliot model the rates are close to zero, for the Semi-Markov model the negative confirmations made up less than one percent of all confirmations. Hence, it can be reasonably expected that the case of having more than three retransmissions occurs rarely and does not influence the results much.

In Figures 7.2 and 7.8 we show for the Gilbert-Elliot error model and the low priority loads of 10% and 50% respectively, the $\widetilde{D_C}$ values for the different round-robin bounds k and the $\widetilde{D_C}$ value for the PROFIBUS. The latter is determined as follows: for a fixed station number N take the minimum $\widetilde{D_C}$ value for all possible gap factors and T_{TTRT} values, i.e., the best achievable value for the given parameter ranges (which are actually chosen towards fastest re-inclusion of stations, i.e., there is much bandwidth consumption for re-including stations). For the Semi-Markov model the corresponding curves are shown in Figures 7.3 (10% low priority load) and 7.9 (50% low priority load), for the complex model the curves are shown in Figures 7.5 (10% low priority load) and 7.11 (50% low priority load), while for the independent model the curves are shown in Figures 7.4 and 7.10.

Furthermore, in Figure 7.6 we show for the 10% low priority load case the fraction of time, that all stations are member of the logical PROFIBUS ring. For a fixed N, fixed error model and fixed channel model, this mean value is taken over all possible different gap factors and T_{TTRT} values. The same is shown for the 50% low priority load case in Figure 7.12.

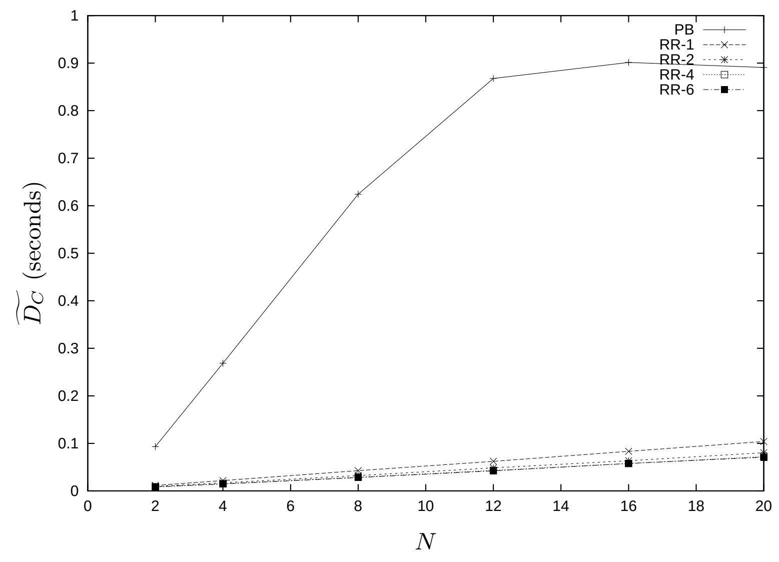

Figure 7.2: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round-robin bounds k

The 10% low priority load case

For the Gilbert-Elliot error model the advantage of the round-robin protocols over the PROFIBUS protocol is impressive, see Figure 7.2. A factor of eight in $\widetilde{D_C}$ performance is reached. The ring stability data shown in Figure 7.6 provides an explanation: since the logical PROFIBUS ring is rather unstable, high priority requests have occasionally rather long waiting times in PROFIBUS stations currently not member of the ring. For the highest number of stations the PROFIBUS performance becomes better. This is likely due to typically shorter gap lists, leading to faster re-inclusion.

For the Semi-Markov model also a clear advantage of the round-robin protocols over the PROFIBUS protocols is visible, although there are some differences:

- The PROFIBUS $\widetilde{D_C}$ performance is better than for the Gilbert-Elliot case (max. $\approx$ 650 ms for 20 stations as compared to 760 ms for the Gilbert-Elliot case). This corresponds with a better ring-stability as indicated in Figure 7.6. This can be attributed to the larger variance of the underlying distributions of the bit-error-free and packet-loss-free periods, leading occasionally to long times of no errors. During these times the PROFIBUS protocol can achieve a full ring and empty its queues.
- The round-robin protocols show worse performance for the Semi-Markov model than for the Gilbert-Elliot model. This can be explained by the fact that longer error bursts or packet loss bursts occur more often in the Semi-Markov model, due to its higher level of variability in the error burst / packet loss burst lengths distributions.
- Although the differences are small, for both the Gilbert-Elliot and the Semi-Markov model the 4-limited and 6-limited round-robin protocol achieves better results than the 1-limited or the 2-limited round-robin. This relationship is inverse to that found in the 50% low priority load

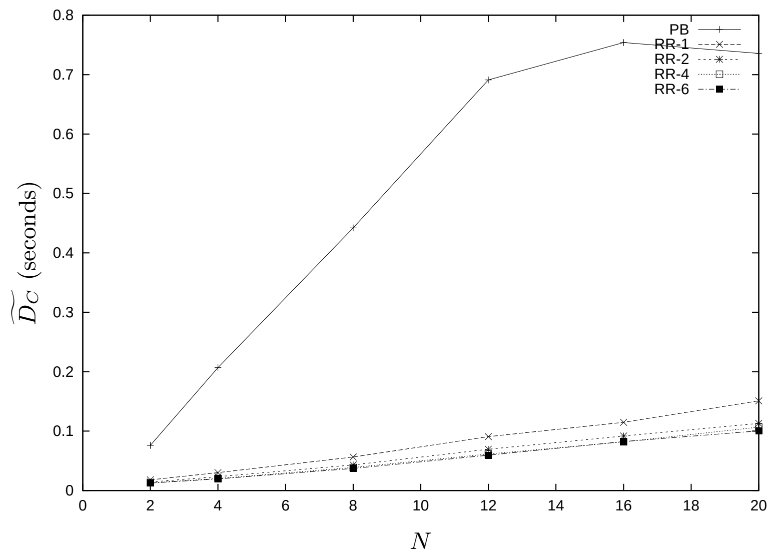

Figure 7.3: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round-robin bounds k

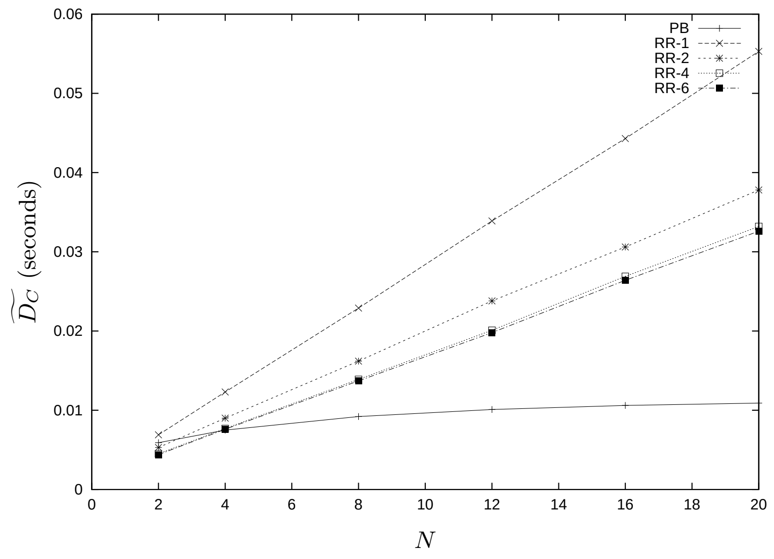

Figure 7.4: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, independent error model and different round-robin bounds k

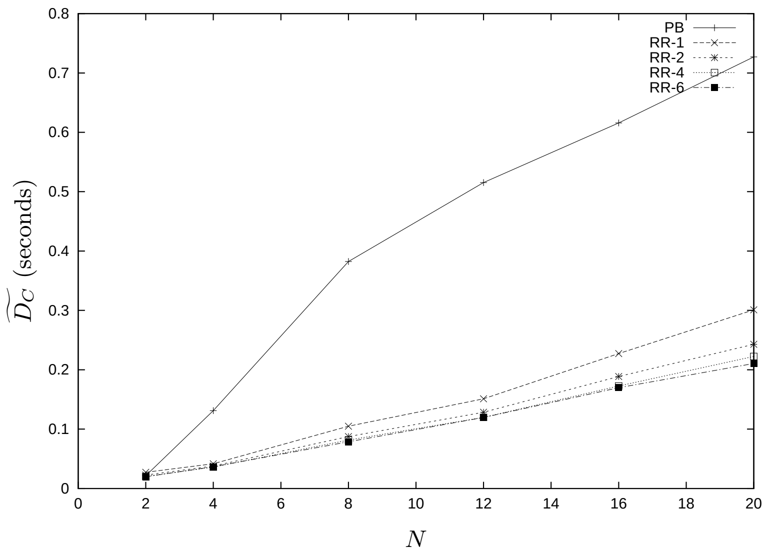

Figure 7.5: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 10% low priority load, complex error model and different round-robin bounds k

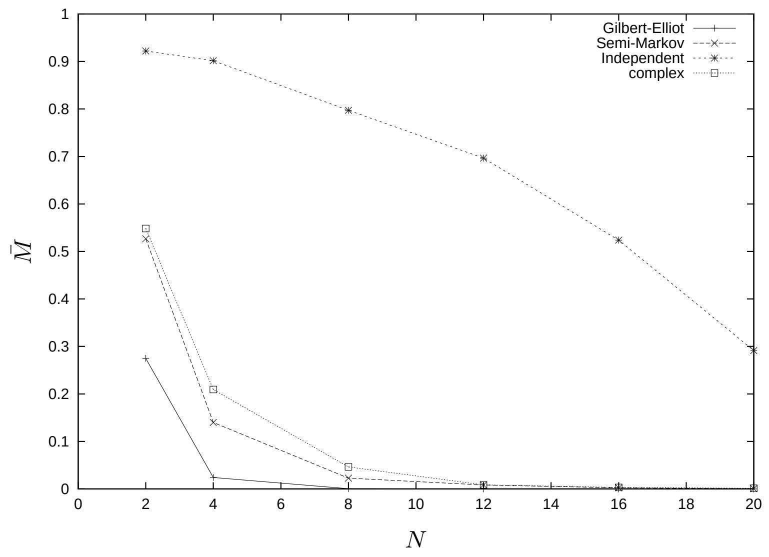

Figure 7.6: Fraction of time, that all stations are member of the logical PROFIBUS ring over all different gap factors and T_{TTRT} values vs. station number N for different error models (10% low priority load)

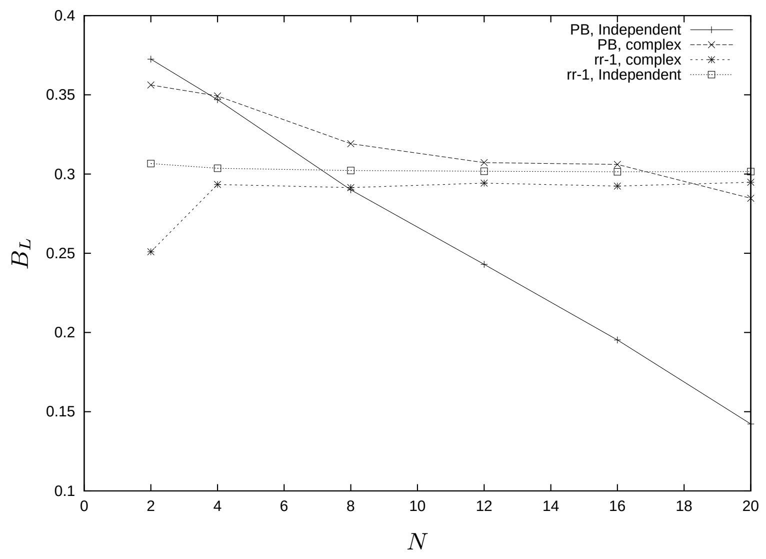

Figure 7.7: Remaining bandwidth for low priority data for the rr-1 protocol and the PROFIBUS protocol (best value over all parameters) B_L vs. number of wireless terminals N for 50% low priority load and independent and complex error models

case (compare Figures 7.8 and 7.9). For the high load case the advantage of small k values over the larger ones can be attributed to having WT ν performing low priority requests, while the high priority requests at its successor WT $\nu + 1$ have to wait. For the low priority load case the advantage of high k values over the smaller ones can be explained as follows. We observe that the curves for $k = 2$ for both the Gilbert-Elliot and the Semi-Markov model are more close to the curves for $k = 4$ and $k = 6$ than to $k = 1$. Hence, it is reasonable to assume that a WT can make benefit of $k > 1$ due to having the ability to perform immediate retransmissions of high priority frames, instead of waiting for the next poll cycle. This effect can only show up in the low priority case, since otherwise high priority requests at WT $\nu + 1$ can be blocked by ν's low priority requests.

For the complex model again the round-robin protocol has a clear advantage of the PROFIBUS protocol (Figure 7.5). The round-robin protocols show a worse performance than for the Gilbert-Elliot model. This is likely due to the stations, which are attached to the bad link corresponding to trace 1 of the **longterm1** measurement. Interestingly, the PROFIBUS does not show worse results than for the Gilbert-Elliot case.

The results for the independent model look different:

- The overall $\widetilde{D_C}$ performance for both the round-robin protocols and the PROFIBUS protocol is much better than for the other error models. For the PROFIBUS this is due to two reasons: the first one is that ring stability is much better than for the other two error models (compare Figure 7.6), hence, fewer requests are blocked in the queues of non-ringmember stations. The second reason is that the PROFIBUS policy of performing immediate retransmissions is the most appropriate for independent errors. In this case, postponing a retransmission gives not a better probability for success (as can be expected for bursty channels after waiting long enough), but only increases the delay. The latter argument is also true for the rr-2, rr-4 and rr-6 protocols. But even the rr-1 protocol achieves a gain for the independent error model.

- The PROFIBUS protocol achieves better results than the rrk protocols (the same is true for the 50% low priority load case). Specifically, a careful inspection of the simulation results shows that the best PROFIBUS version is that with the smallest T_{TTRT} value of 5 msec. This points towards an explanation of this phenomenon: by the PROFIBUS bandwidth assignment rules (see Section 4.1.4) for a high number of stations and extremely tight T_{TTRT} value it happens often that a station computes a negative token holding time upon token arrival. In this case it is allowed to perform at most one high priority request, but no low priority requests. If no high priority request is available, the station is required to pass the token to its successor, instead of handling a low priority request. In contrast to this, in the round-robin protocols a station always handles at least one request, if one of its queues is nonempty. This introduces additional delay for the following stations. To validate this explanation, we show in Figure 7.7 the remaining bandwidth for low priority requests B_L for the independent error model, the PROFIBUS and the rr-1 protocol and the 50% low priority load case. It can be seen that rr-1 constantly uses 30% of the bandwidth for transmission of low priority data, while the PROFIBUS achieves its good $\widetilde{D_C}$ performance at the cost of decreasing B_L performance.
- Again the rr-4 and rr-6 protocols achieve better performance than the rr-1 and rr-2 protocols.

Within the k-limited round-robin framework a similar bandwidth distribution scheme as for the PROFIBUS can be approximated by a small modification of the `poll` frame: a special flag could indicate to a WT that it should only transmit data, if its high priority queue is nonempty. The BS can use this flag according to different policies. One possibility is to switch on this flag, when the cycle time of the last polling cycle exceeded some prescribed threshold.

The 50% low priority load case

For the Gilbert-Elliot model we still have a clear advantage of all round-robin protocols over the PROFIBUS protocol (see Figure 7.8), the latter having at least two times the $\widetilde{D_C}$ value than the worst round-robin protocol (rr-6). For $k = 4$ the ratio is approximately three, for $k = 1$ and $k = 2$ approximately four. It can be seen that the rr-1 and rr-2 protocols have better $\widetilde{D_C}$ performance than rr-4, the rr-6 protocol is even worse. As explained in the preceding section, this can be attributed to emptying low priority queues at WT ν, which increases the waiting times for high priority requests at its successor WT $\nu + 1$ (we call this mechanism *low priority successor blocking*).

The same observations apply to the Semi-Markov model (see Figure 7.9), however, while the PROFIBUS $\widetilde{D_C}$ performance is nearly constant, the $\widetilde{D_C}$ performance for the round-robin protocols decreases slightly. This behavior was already observed for the 10% low priority load case.

For the complex model (shown in Figure 7.11) the rr-1 and rr-2 protocols are always better, rr-4 is beaten by the PROFIBUS protocol only for $N = 20$, while rr-6 looses against PROFIBUS already for $N > 12$. A likely explanation is that the polling-protocols suffer from blocking effects.

Again, when looking at the independent model (see Figure 7.10) all the $\widetilde{D_C}$ values have an absolutely lower level than for the other error models, and the PROFIBUS protocol shows the best performance. The rr-6 protocols shows the worst performance of the round-robin protocols, followed by rr-4. Interestingly, rr-2 seems to be slightly better than rr-1. Hence, the increase from $k = 1$ to $k = 2$ is

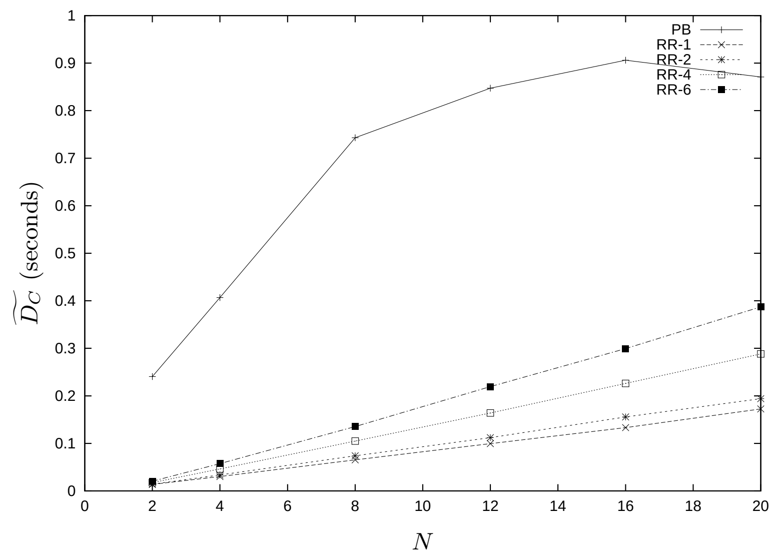

Figure 7.8: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round-robin bounds k

beneficial. Probably it is due to a good balance between the possibility for immediate retransmissions for high priority frames and low priority successor blocking. For $k > 2$ the low priority successor blocking mechanism overshadows this small gain.

The PROFIBUS $\widetilde{D_C}$ performance for the independent error model is the same as for the 10% low priority load case. This indicates that the PROFIBUS protocols realtime performance does not depend much on the low priority load. Again, the price to be paid here is the starvation of low priority traffic for higher station numbers.

Overall Observations

An interesting observation is true for both load scenarios: the round-robin protocols show for all error models a linear dependence of $\widetilde{D_C}$ performance from the number of stations N, only the slope varies between the different error models and k values. Once the slope for a certain protocol is known, this property allows an easy assessment of the realtime capacity (as defined in Section 4.3.1) of the round-robin protocols.

This observation is not surprising, since it reflects a simple property of the round-robin protocols (which shows up nicely under homogeneous load conditions): each station gets the same share of bandwidth and can make purely local decisions on its usage, without considering the behavior of other stations. This can lead to blocking effects.

In contrast to this, for the PROFIBUS other stations have an short-term influence on the bandwidth assignment within a single PROFIBUS station by the modified timed-token protocol. This fits together with the nonlinearities observed in the figures.

From Figures 7.6 and 7.12 it could be observed that for a higher number of stations the ring-stability

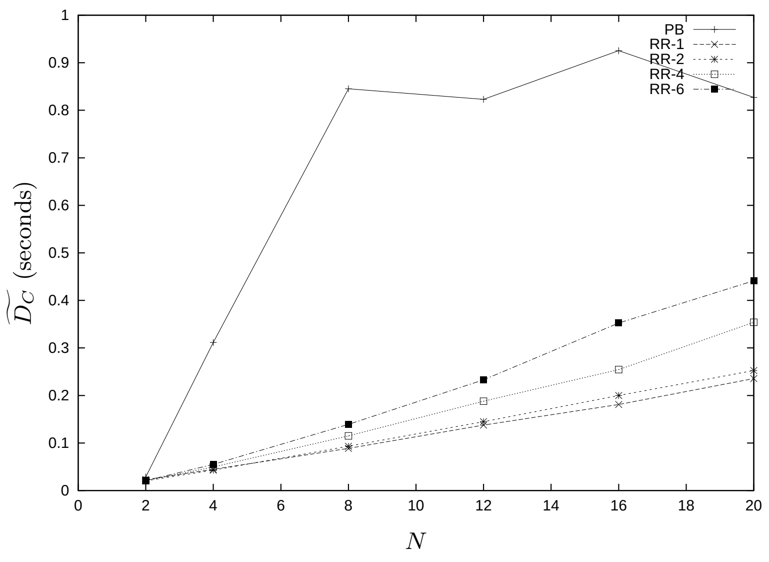

Figure 7.9: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round-robin bounds k

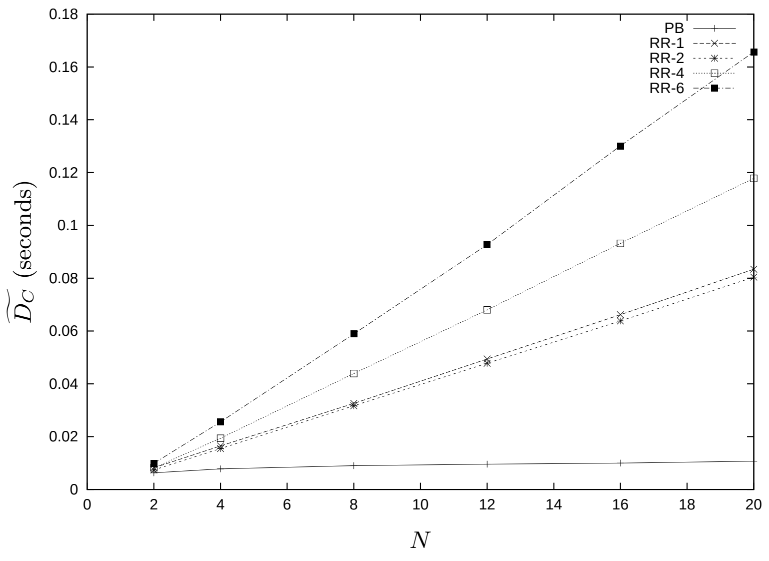

Figure 7.10: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, independent error model and different round-robin bounds k

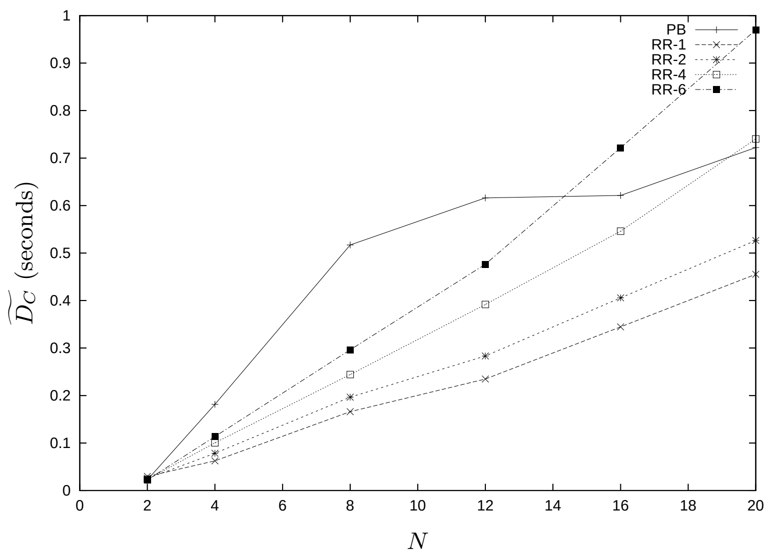

Figure 7.11: Overall confirmation delays $\widetilde{D_C}$ for the rrk-protocols and the original PROFIBUS protocol vs. number of wireless terminals N for 50% low priority load, complex error model and different round-robin bounds k

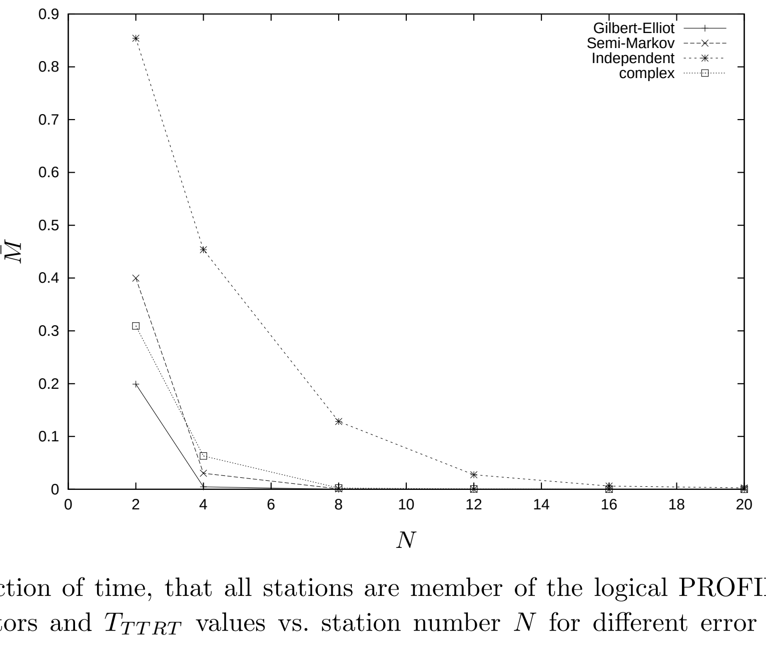

Figure 7.12: Fraction of time, that all stations are member of the logical PROFIBUS ring over all different gap factors and T_{TTRT} values vs. station number N for different error models (50% low priority load)

is much worse for the 50% low priority load case than for the 10% low priority load case. This can be explained that under higher system load it takes longer to re-include a WT into the logical ring, since the ring-maintenance frames are only used when there is spare bandwidth (see Section 4.1.4), which happens not so often as in the low load case.

To summarize, the polling-based protocols have under all bursty error models (Gilbert-Elliot, Semi-Markov, complex) almost always an impressive advantage over the PROFIBUS protocol. Only in the case of independent errors (which is unlikely to occur in real environments) the PROFIBUS shows advantages. However, the polling-based protocols can easily mimic the behavior of the PROFIBUS to suppress low priority traffic in congested situations.

7.3.3 Comparison of the Polling-protocol Modifications with Round-Robin

In the previous section we have shown that the round-robin protocols deliver superior realtime performance over the PROFIBUS protocol for the bursty error models. And even for the independent error model the round-robin protocols deliver good realtime performance (as compared to the other error models), while the better PROFIBUS performance comes at the price of suppressing low priority traffic at higher number of stations.

These results establish round-robin protocols as a good starting point for the design of protocols with improved realtime performance. In this section we investigate the effects of the mechanisms proposed in Section 7.2, using all the error models (independent, Gilbert-Elliot, Semi-Markov, complex) defined in Section 7.1.1.

The results reported in this section are all obtained with the polling-simulator described in Section 7.3.1. We compare the round-robin protocol first with each single mechanism, and finally with a protocol with all three proposals (SDR, SPR, adaptive functional repoll) enabled.

In the following, the simple k-limited round robin protocols is denoted as rrk, rrk augmented with the SPR protocol is denoted as rrk+spr, accordingly rrk+sdr for the SDR protocol. The adaptive functional repoll protocol with a specific round-robin bound k is denoted as fr-k, the class of protocols is denoted as frk.

The common parameters for all protocols are summarized in Table 7.5. The SPR protocol incorporates a memory-loss function $g(\cdot)$, indicating the influence of older history information. This function was chosen to be

$$g(x) = \mathbf{1}_{[0,1000]}(x) \exp\left(\frac{-x}{100}\right)$$

The history depth is set to 20. The timeout T_{Quiet} after which the BS starts an `explicit-poll` cycle, is set to three times $T_{MaxCycle}$, where $T_{MaxCycle}$ denotes the maximum time a single cycle for the k-limited round robin case can take for N stations:

$$T_{MaxCycle} = N \cdot (\text{controlpacketsize} + k \cdot (\text{datapacketsize} + \text{ackpacketsize}))$$

For the frk protocols several functions need to be provided: the memory-loss function $h(\cdot)$ and the set of repoll functions. Furthermore, the update frequency T_{AR} has to be chosen. For this work an update is performed after every 100 poll frames (i.e., one of `poll`, `explicit-poll`, `invited-poll`, `execute-explicit-poll`). The function $h(\cdot)$ is set to $h(x) = e^{-\frac{x}{20}}$, where the number 20 is related to the *max_retry* parameter. The window size for the moving average channel quality estimation is set to 40. As the set of possible repoll functions to choose from in the frk protocols we have considered the functions summarized in Table 7.6.

Comparison of Round-Robin with Round-Robin+SPR

The method of comparison of rrk+spr with rrk is simple: for every investigated number of stations the rrk+spr's $\widetilde{D_C}(rrk + spr)$ value is divided by the corresponding value for the the rrk protocol $\widetilde{D_C}(rrk)$, giving the ratio $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$.

First we look at the B_L performance measure, indicating for the 50% low priority load case the fraction of bandwidth remaining for low priority traffic (see Section 4.3.1). For the Gilbert-Elliot model we first

Parameter	**Values**
Number of stations	2, 4, 8, 12, 16, 20
round-robin bound k	1, 2, 4, 6
Modifications	SDR, SPR, fr-k, SDR+SPR+fr-k
channel models	independent, Gilbert-Elliot, Semi-Markov, complex
load models	smooth (10% low priority load), smooth (50% low priority)
max_retry parameter (low, high)	20
Bit rate	2 MBit/s (QPSK modulation)

Table 7.5: Common simulation parameters for performance comparison of the different protocol modifications rrk+x vs. k-limited round robin

Name	**Expression**
immediate repolling	$f_{ir}(i) = i - 1$
bounded immediate repolling	$f_{bir}(i) = \mathbf{1}_{[1,3]}(i) \cdot (i-1) + (-1)\mathbf{1}_{(3,\infty)}(i)$
fast linear repoll	$f_{cr1}(i) = 2 \cdot i - 1$
slow linear repoll	$f_{cr2}(i) = 4 \cdot i - 1$
quadratic repoll	$f_{qr}(i) = \mathbf{1}_{[1,5]}(i) \cdot i^2 + (-1)\mathbf{1}_{(5,\infty)}(i)$

Table 7.6: Repoll function set for the frk protocols, for all functions additionally $f_x(i) = -1$ if $i > max\_retry$ holds

observe from Figure 7.15 that the additional cost in terms of B_L performance of the SPR mechanism in the rrk+spr protocols as compared to the basic rrk protocols is below 3.5%, which we assume to be acceptable.[7] The bandwidth loss of rrk+spr is larger for smaller k values. This is clearly due to the better ratio of `data` frames to `poll` frames achievable with larger k values. The same observation is true for the Semi-Markov model, with the bandwidth loss approaching 4%. For the complex error model the bandwidth loss is below 1%.

In general, for all the protocols rrk+spr, rrk+sdr, frk, frk+spr+sdr and all error models the bandwidth overhead of the respective additional mechanism is below 4% as compared to rrk. Hence, these mechanisms incur only small penalties in terms of B_L performance. Interestingly, for the rrk+sdr protocol and the frk protocol actually slightly more bandwidth was available for the low priority traffic, giving a small gain in B_L performance.

For the 50% low priority load case the ratio $\frac{\widetilde{D_C(rrk+spr)}}{\widetilde{D_C(rrk)}}$ vs. number of stations N is shown in Figures 7.14, 7.17, 7.19, and 7.21 for the Gilbert-Elliot, Semi-Markov, independent, and complex error models, respectively. The following observations are interesting:

- For $k \geq 2$ the rrk+spr protocol achieves a real gain for all error models. For the Gilbert-Elliot and the Semi-Markov model this gain is significant (rrk+spr shows for $k = 6$ and $N \geq 4$ less than half the $\widetilde{D_C}$ value of rrk, for $k = 2$ still savings of 30% to 40% can be achieved, and even $k = 1$ saves between 10% and 20%), for independent errors up to 20% savings for higher number of stations could be reached, and for the complex error model for $N \geq 8$ between 15% and 30% can be saved.

[7]This fraction likely depends on the T_{Quiet} parameter. This is a subject of further research.

	50%low priority load	**10% low priority load**
# of `execute-explicit-poll` frames	1902	29916
# of successful `execute-explicit-poll` frames	1453	23077
# of null answers to `explicit-poll` frames	1388	22771

Table 7.7: `execute-explicit-poll` frame statistics for rrk+spr, independent error model, $N = 20$, $k = 1$

- For all error models the rrk+spr protocols achieve a higher gain for higher values of k, with $k = 4$ and $k = 6$ giving better gains than $k = 2$, which in turn beats $k = 1$. In all cases the performance gain tends to increase in the number of stations.

For the 10%low priority load case (see Figures 7.13, 7.16, 7.18, and 7.20) the picture changes:

- For $k \geq 2$, the Gilbert-Elliot and Semi-Markov model, the rrk+spr protocols make up to 30% gains over the rrk protocols in $\widetilde{D_C}$ performance. For $k = 2$ we achieve the best gains, for $k = 6$ the smallest. For the independent and the complex error model rrk+spr has an up to 10% worse $\widetilde{D_C}$ performance than rrk.
- For $k = 1$ sometimes the rr-1+spr protocol produces $\widetilde{D_C}$ losses of 10% to 25% (Gilbert-Elliot, independent, complex) as compared to rr-1, while for the Semi-Markov model gains of up to 12% could be observed.

The different behavior of the 10% low priority load and the 50% low priority load case can be explained by the operation of the round-robin protocols: as pointed out in Section 7.2.1, when a WT is polled and has no packet to transmit, it keeps quiet. However, for the BS this is indistinguishable from the case that the WT has not properly received the poll frame. If the overall load is low, a WT will experience a higher probability for no newly arriving requests within a certain time, hence, the probability of the SPR mechanism querying a WT with empty queues is higher. In these cases the SPR mechanism is pure overhead without any gain. To illustrate this, we show in Table 7.7 for $N = 20$, $k = 1$, the independent error model and the rrk+spr protocol some statistics about the occurence of `execute-explicit-poll` frames. It can be seen, that for the 10% low priority load case we have ≈ 15 times more of these frames than for the 50% low priority load case. In this table, a `execute-explicit-poll` frame is counted as successful, if the target station actually responds to the `explicit-poll` frame sent by the relayer station.

This points to a general tradeoff: if a single WT's load is high, the SPR protocol is invoked only rarely for this WT, hence, only small extra bandwidth is spent. If a single WT's load is comparably low, the SPR protocol is going to be invoked more often (depending on the T_{Quiet} parameter). If there is one subset of WT's with high loads and another subset with low loads, the T_{Quiet} parameter has a direct influence on the remaining bandwidth available for the high-load WT's and on the $\widetilde{D_C}$ performance of the low-load WT's.

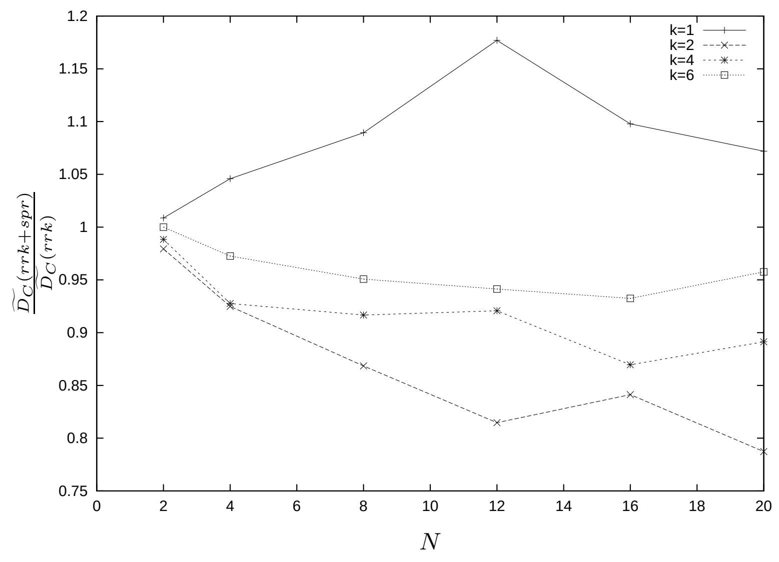

Figure 7.13: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k

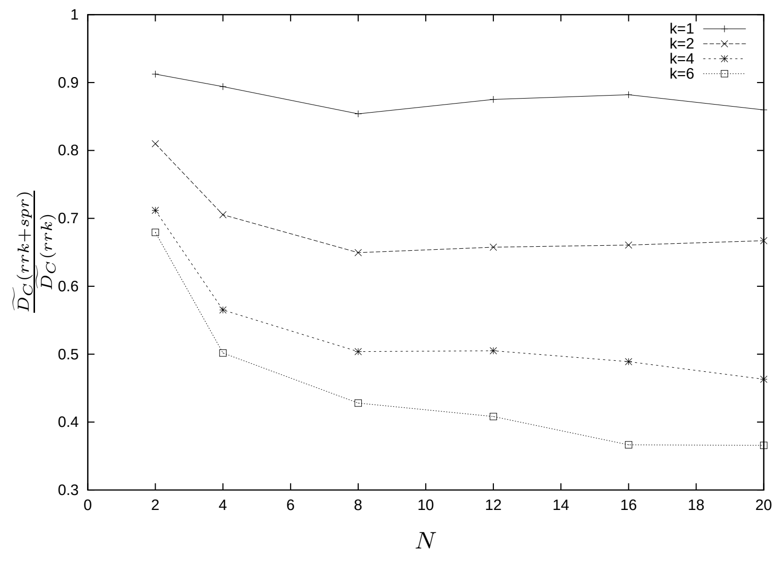

Figure 7.14: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k

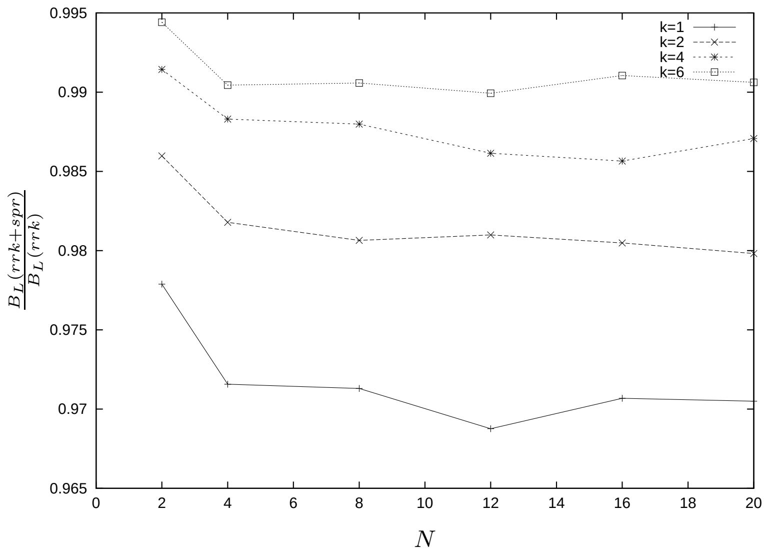

Figure 7.15: Ratio of the remaining bandwidth for low priority data for the rrk+spr protocol and the rrk protocol $\frac{B_L(rrk+spr)}{B_L(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k

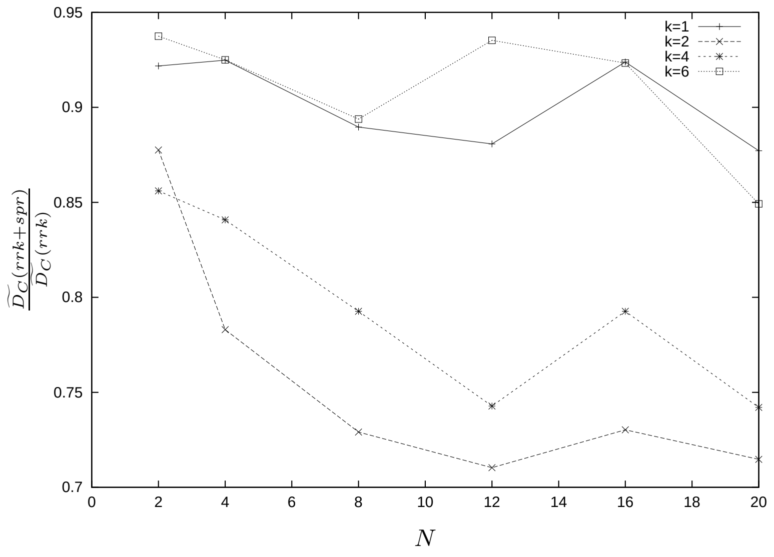

Figure 7.16: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C(rrk+spr)}}{\widetilde{D_C(rrk)}}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k

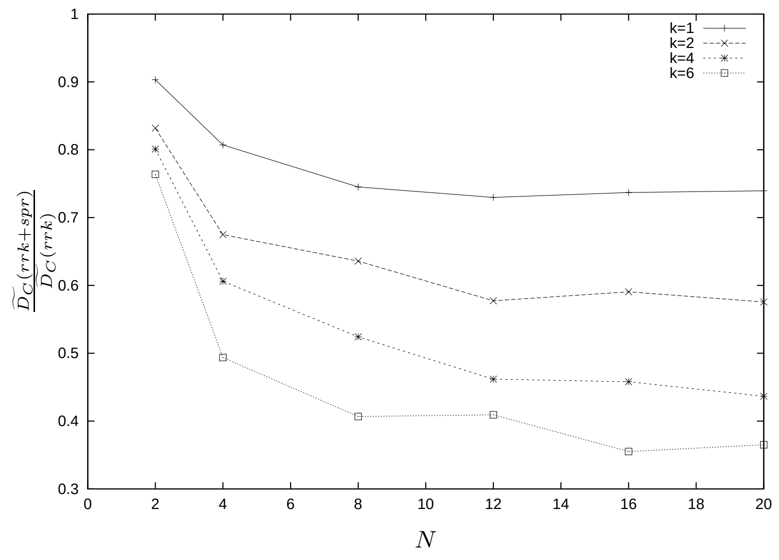

Figure 7.17: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k

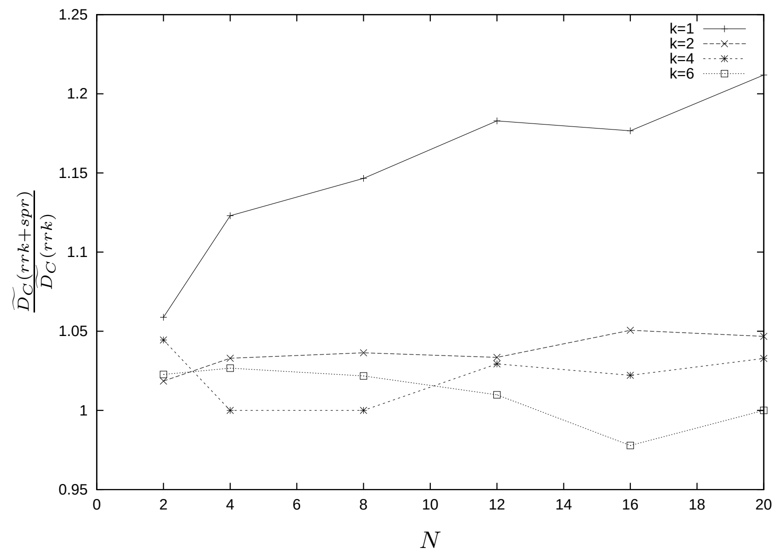

Figure 7.18: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k

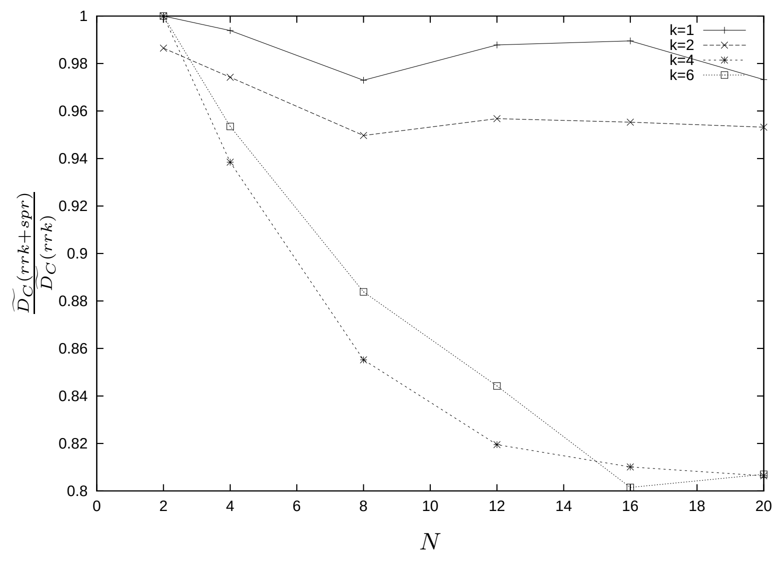

Figure 7.19: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k

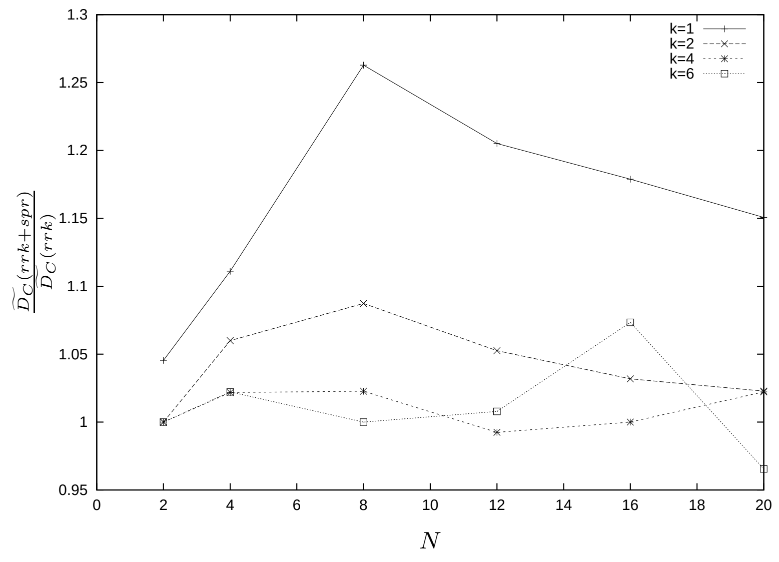

Figure 7.20: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k

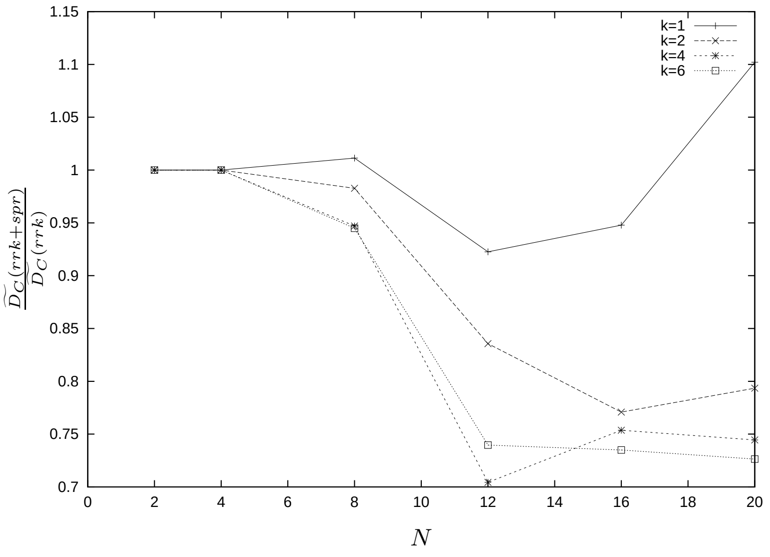

Figure 7.21: Ratio of the Overall confirmation delay for the rrk+spr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+spr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k

Comparison of Round-Robin with Round-Robin+SDR

The approach for comparison of rrk and rrk+sdr is the same as in the previous section, namely, we look at the ratios $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$.

For the 10% low priority load case the ratio $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of stations N is shown in Figures 7.22, 7.24, 7.26, and 7.28 for the Gilbert-Elliot, Semi-Markov, independent, and complex error model, respectively. The same is shown for the 50% low priority load case in Figures 7.23, 7.25, 7.27, and 7.29, respectively.

For all loads and all error models we observe that the rrk+sdr protocol makes the biggest gains for $k = 1$. Furthermore, for $k = 1$ there are always real gains, sometimes up to 26% for the "artificial models", and up to 55% for the complex model (see Figure 7.28).[8] This can be explained by the fact that the SDR approach performs a kind of immediate retransmission, whereas in the rr-1 protocol a retransmission has to wait one poll cycle. For $k = 2$ it can also happen frequently that a retransmission has to wait one poll cycle, and indeed for $k = 2$ rrk+sdr achieves almost all the time the second-best results.

For $k = 4$ and $k = 6$ for both loads and all error models we observe comparably small gains (up to 10%) or even slight losses (up to 6%). Furthermore, the curves show no clear dependence on the number of stations. Even for the complex error model the gain for $k = 4$ reaches only 10%. For higher k values a WT can most often handle a high priority request fully within one token cycle, in contrast to the smaller k values. Hence, it occurs less often that the SDR scheme can save a full poll cycle.

The bias of the SDR scheme of having the largest gains for $k = 1$ and $k = 2$ makes its usage beneficial for high load situations. To justify this, we again observe from the comparison with the PROFIBUS protocol (see Section 7.3.2, Figures 7.8, 7.9, and 7.10) that the absolute values of the $\widetilde{D_C}$ measure for the round-robin protocols are better for small k values. And specifically in these cases rrk+sdr gives gains.

As an overall impression, the rrk+sdr scheme seems to be not very sensitive on the load, but more on the error model. For the same error model the curves for the different low priority loads tend to be similar in shape and order of magnitude. The curves for different error models and fixed load look much more different. Hence, the error model affects rrk+sdr performance more than the overall load does.

[8]Here is an interesting problem for further research: one can guess that the gains of SDR are larger for scenarios with heterogeneous channel error models than with homogeneous ones.

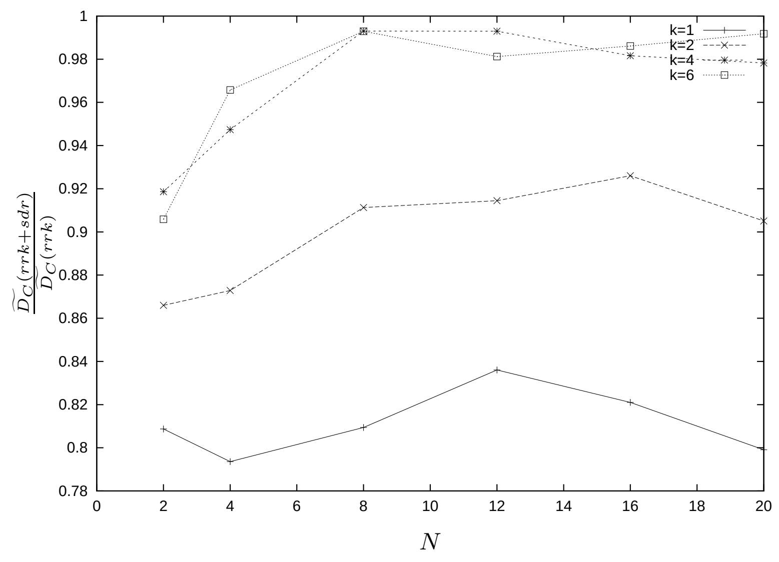

Figure 7.22: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k

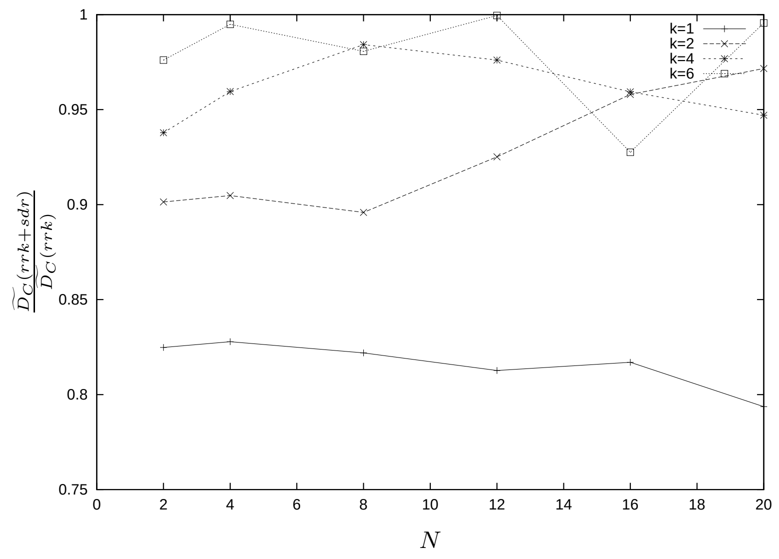

Figure 7.23: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k

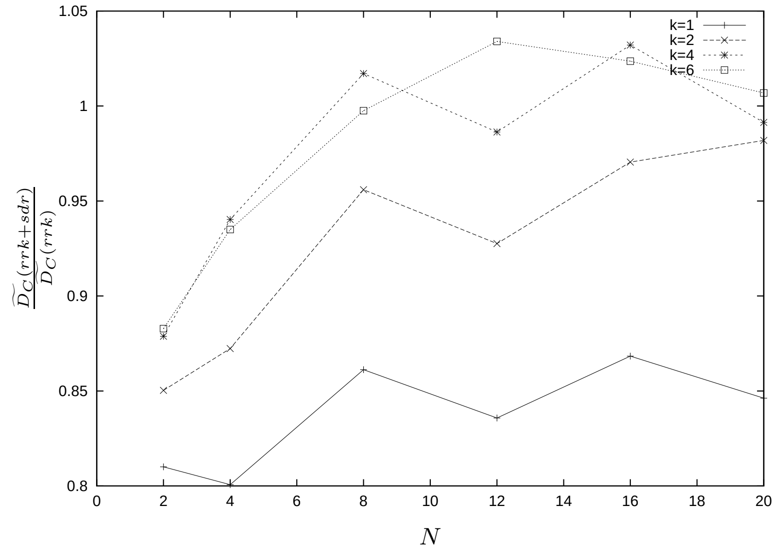

Figure 7.24: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k

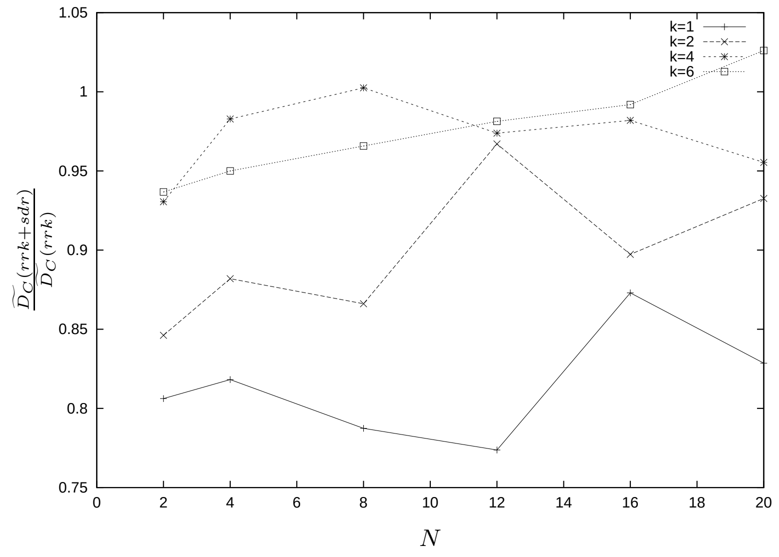

Figure 7.25: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k

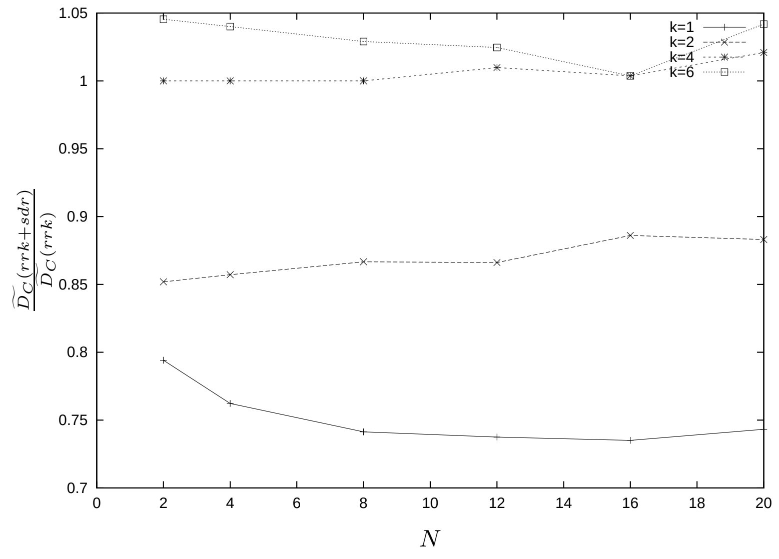

Figure 7.26: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k

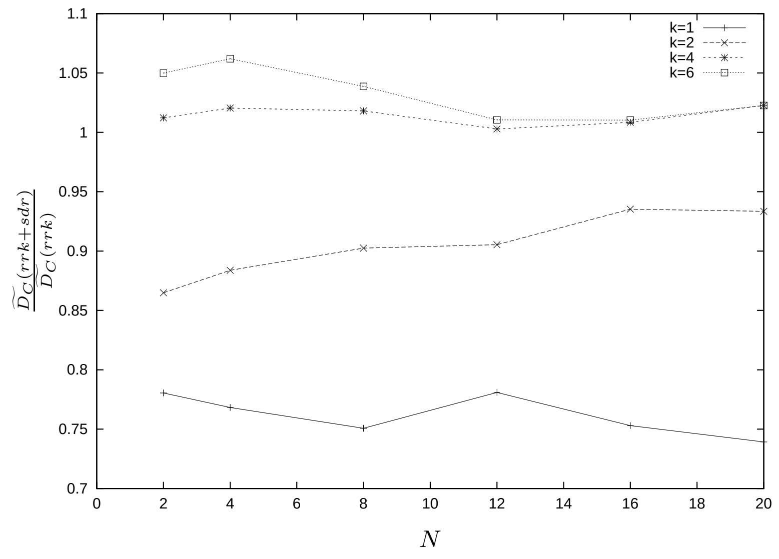

Figure 7.27: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k

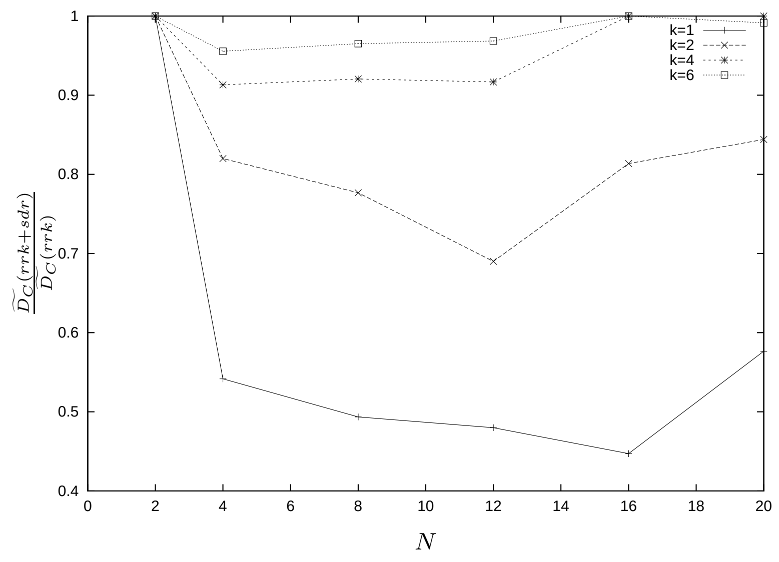

Figure 7.28: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k

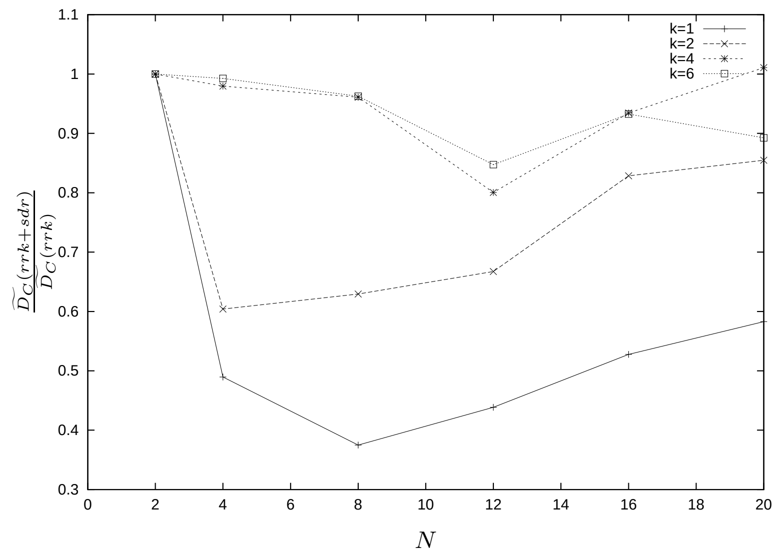

Figure 7.29: Ratio of the Overall confirmation delay for the rrk+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(rrk+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k

Comparison of Round-Robin with Functional Repolling

While the SPR and SDR relaying schemes modify the set of WT's involved in data transmission and the polling process, the functional repolling scheme (frk, see Sections 7.2.2 and 7.2.4) varies the polling sequence and decisions on when to perform retransmissions.

For the 10% low priority load case the ratio $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. the number of stations N is shown in Figures 7.30, 7.32, 7.34, and 7.36 for the Gilbert-Elliot, Semi-Markov, independent, and complex error models, respectively. The same is shown for the 50% low priority load case in Figures 7.31, 7.33, 7.35, and 7.37, respectively.

From these figures it can be seen that the frk scheme produces worse $\widetilde{D_C}$ performance for $k = 4$ and $k = 6$ for all numbers of stations, all error models and all load cases. In contrast, for $k = 1$ we always make gains, and for $k = 2$ only in the complex error model we observe some gains (up to 20% for certain values of N).

To explain the gains for $k = 1$ and $k = 2$, we observe from the set of repolling functions listed in Table 7.6 that all functions have the tendency to make the first retransmissions comparably fast ($f(1) \leq 3$ for all the listed functions). For $k = 1$ and $k = 2$ this is, for N large enough, typically faster than the next time a WT is polled in the normal round-robin protocols.

For higher k values the $\widetilde{D_C}$ performance suffers from two design decisions:

- to announce the full number of k slots to a WT a currently repolled.
- to not interrupt a station for repoll cycles: if the BS sends a `poll` frame with round-robin bound k to WT a at time t_0, and the next repoll cycle is due at $t_0 + 1$, then the repoll has to wait for a maximum of k data transmissions of a before it takes place. Stated differently: repolls are subject to blocking, a `poll` cycle to a WT a is not preempted.

To reduce these blocking times is a topic for further research. One possible approach is that the BS interrupts the poll cycle of a WT a, if it determines that a is going to perform a retransmission of a low priority frame. To interrupt the cycle, the BS sends a `poll` frame to a's successor. By the simple local priority scheduler's operation it is clear, that a has no high priority requests in its queues at this time. The BS can leave a in the normal poll list, while the functional repolling protocol is still executed for the high priority requests.

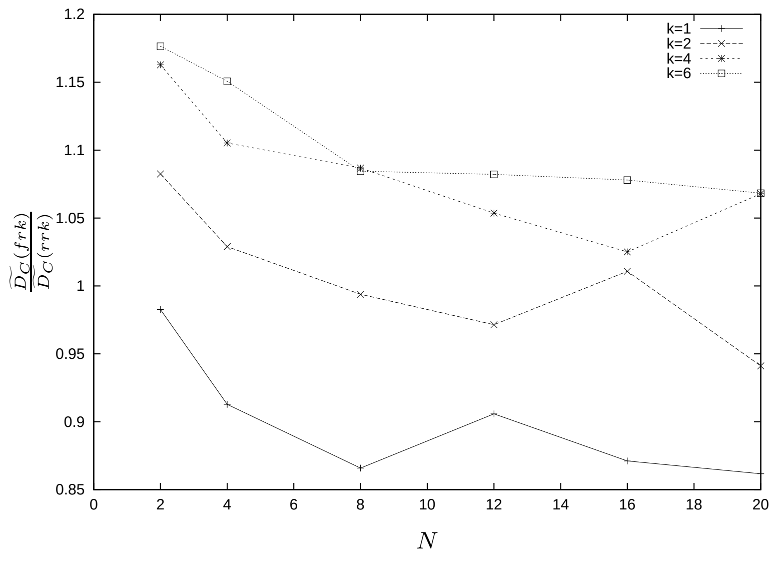

Figure 7.30: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k

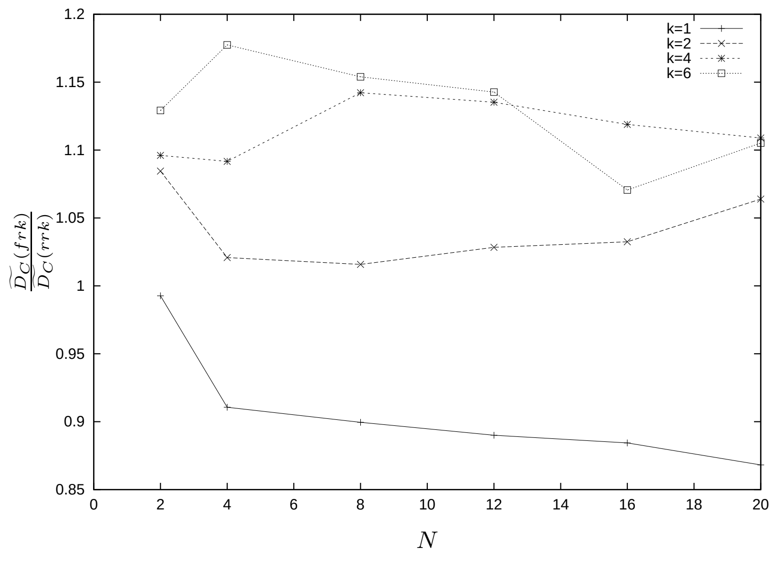

Figure 7.31: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k

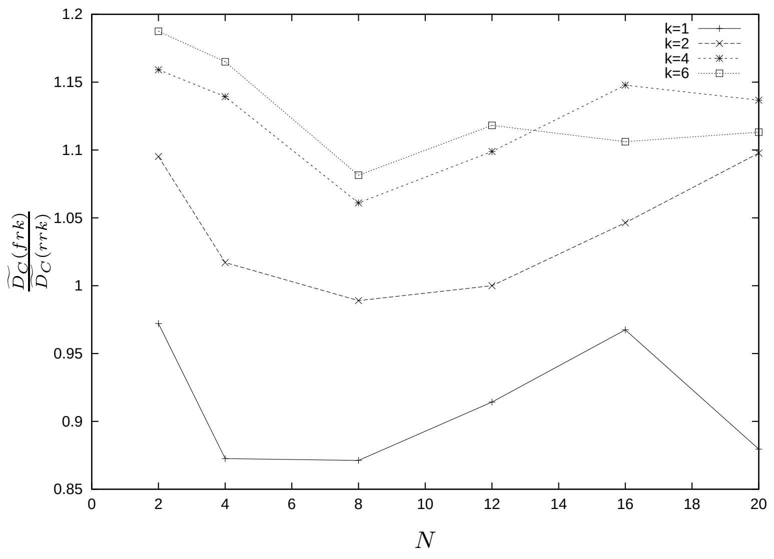

Figure 7.32: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k

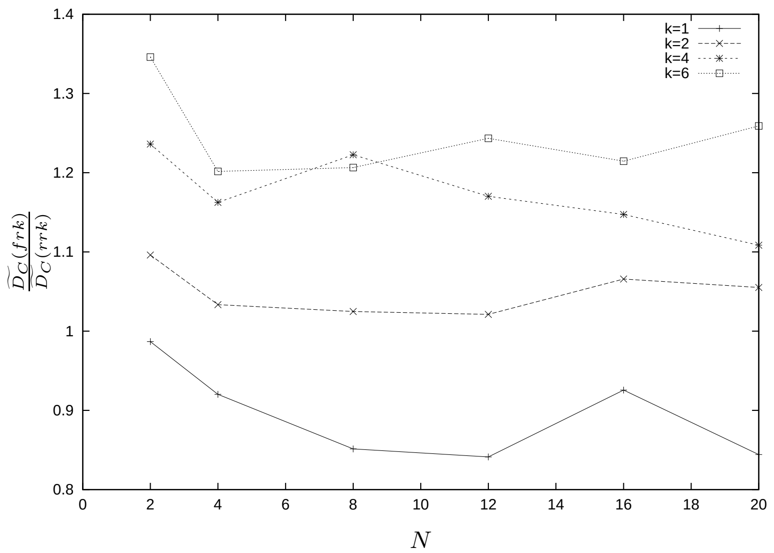

Figure 7.33: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k

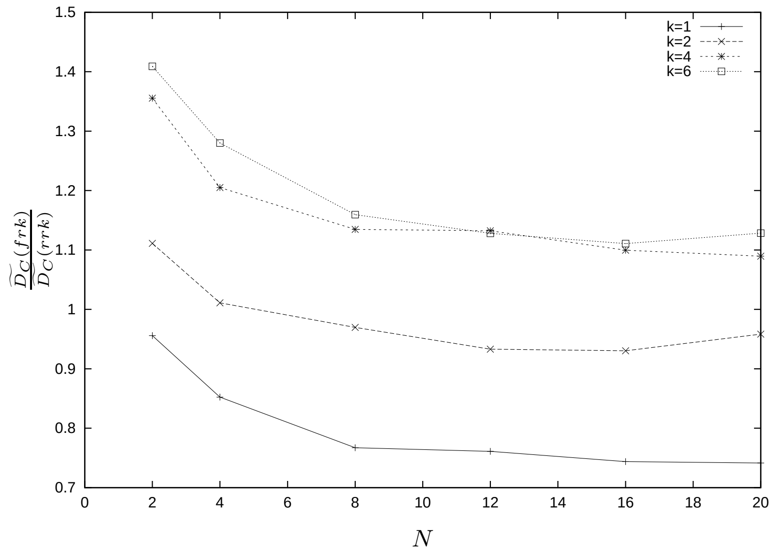

Figure 7.34: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k

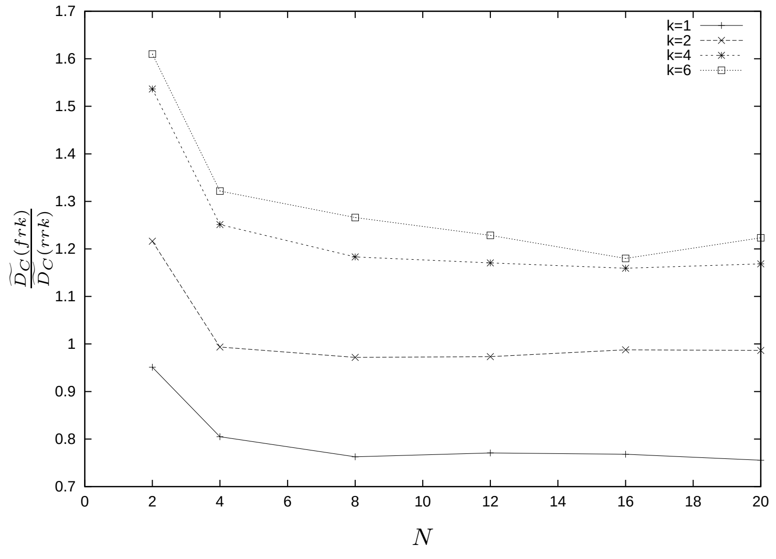

Figure 7.35: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k

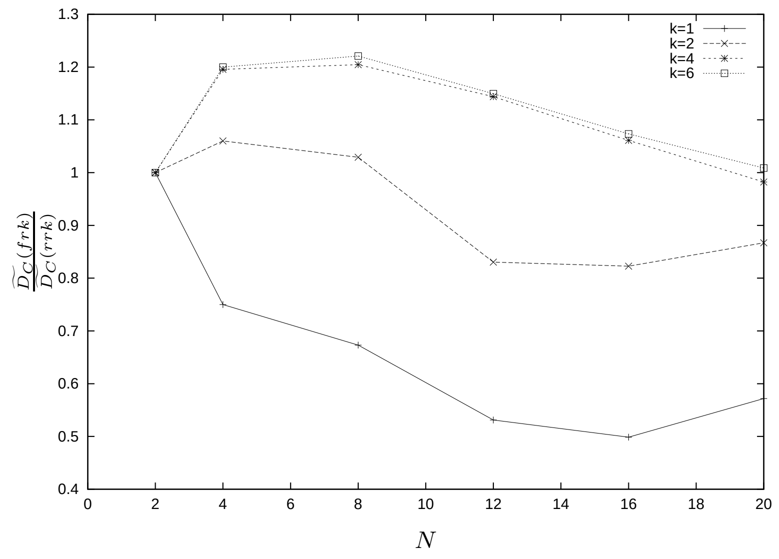

Figure 7.36: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k

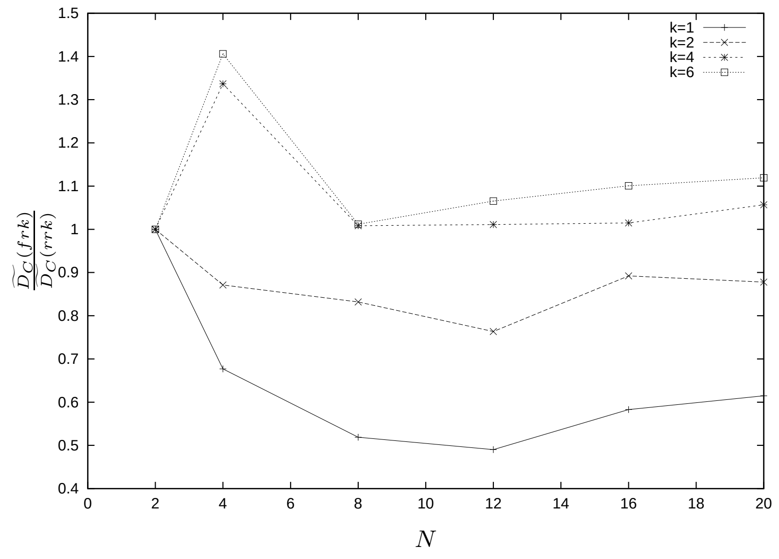

Figure 7.37: Ratio of the Overall confirmation delay for the frk protocol and the rrk protocol $\frac{\widetilde{D_C}(frk)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k

Comparison of Round-Robin with Functional Repolling+SDR+SPR

In the previous sections we have evaluated the three mechanisms functional repolling, SPR and SDR separately. It was shown, that they often achieve gains in $\widetilde{D_C}$ performance in certain situations, but occasionally also worse performance.

It is also interesting to assess different combinations of the mechanisms and their mutual influence. Out of the four possible combinations: rrk vs. SPR+SDR, frk+SPR, frk+SDR, and frk+SPR+SDR, we have chosen to focus on the "full" frk+SPR+SDR approach. The main purpose is to check, how these approaches interfere with each other.

For the 10% low priority load case the ratio $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of stations number N is shown in Figures 7.38, 7.40, 7.42, and 7.44 for the Gilbert-Elliot, Semi-Markov, independent, and complex error models, respectively. The same is shown for the 50% low priority load case in Figures 7.39, 7.41, 7.43, and 7.45, respectively.

The overall result is that the combined approach frk+spr+sdr delivers superior performance over the round-robin protocol for the Gilbert-Elliot and the Semi-Markov error model. For the complex and the independent model we achieve almost always gains for $k = 1$ and $k = 2$, while $k = 4$ and $k = 6$ show significantly worse performance for the independent error model (probably due to the frk protocols, which show their worst performance for the independent error model).

A highlight is the saving for the 50% low priority load case and the Gilbert-Elliot and Semi-Markov models specifically for $k = 1$ and $k = 2$ as compared to the rrk+spr protocol (see Figures 7.39 and 7.14 for the Gilbert-Elliot model, and Figures 7.41 and 7.17 for the Semi-Markov model).

The most impressive result would have been that the frk+spr+sdr approach is always better than rrk+x for x being SPR, SDR or frk. This result could not be achieved due to the problems with the frk approach for $k = 4$ and $k = 6$. When only looking at $k = 1$ and $k = 2$, we state without showing the curves in this thesis that:

- frk+spr+sdr is superior over rrk+spr for $k = 1$ and $k = 2$ for all error models, load cases and all N. Typically the savings are better for $k = 1$.
- frk+spr+sdr is superior over rrk+sdr for $k = 1$ and $k = 2$ for all error models and load cases and all N, except for the complex and independent models at 10% low priority load and the independent model at 50% load. However, the advantage of rrk+sdr for the independent model is only given for $N = 2$, for all other N the frk+spr+sdr protocol performs better.
- frk+spr+sdr is superior over frk for $k = 1$ and $k = 2$ for all error models, load cases and N, except for the complex error model at 10% low priority load (here only for $N = 2$ frk is superior) and the independent error model at 10%.

The savings of the frk+spr+sdr protocol over the other protocols can reach up to 60%. The worse performance of the "full" protocol occurs mostly for $N = 2$, for higher number of WT's N the combined protocol is superior. Hence, for $k = 1$ and $k = 2$ the protocols influence each other in a constructive way.

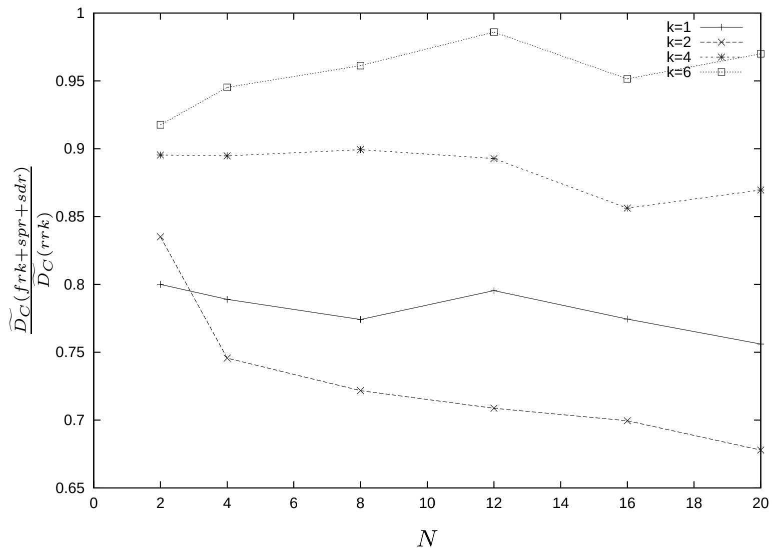

Figure 7.38: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Gilbert-/Elliot error model and different round robin bounds k

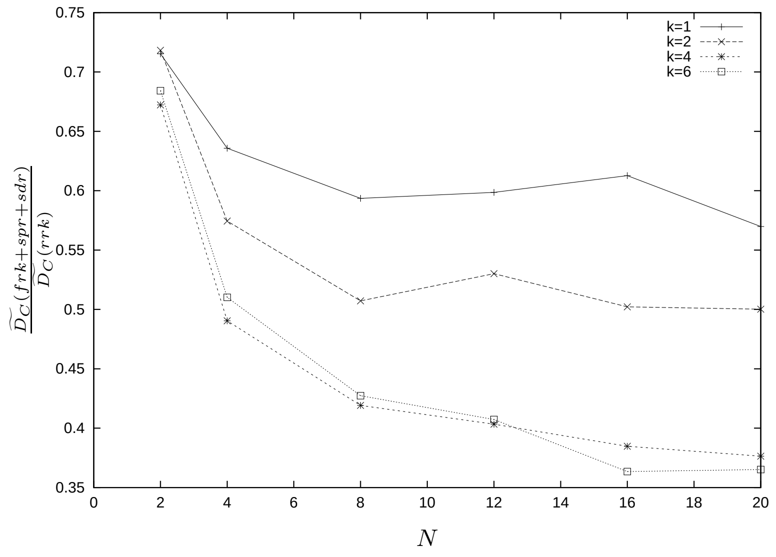

Figure 7.39: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Gilbert-/Elliot error model and different round robin bounds k

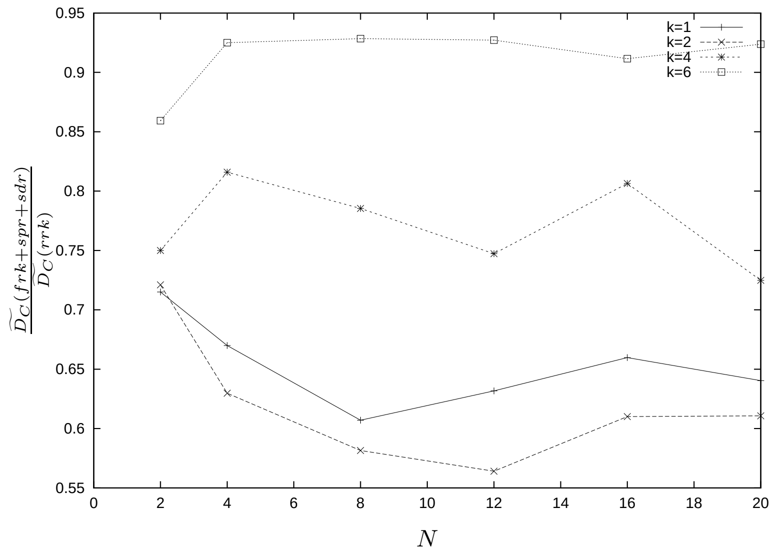

Figure 7.40: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, Semi-Markov error model and different round robin bounds k

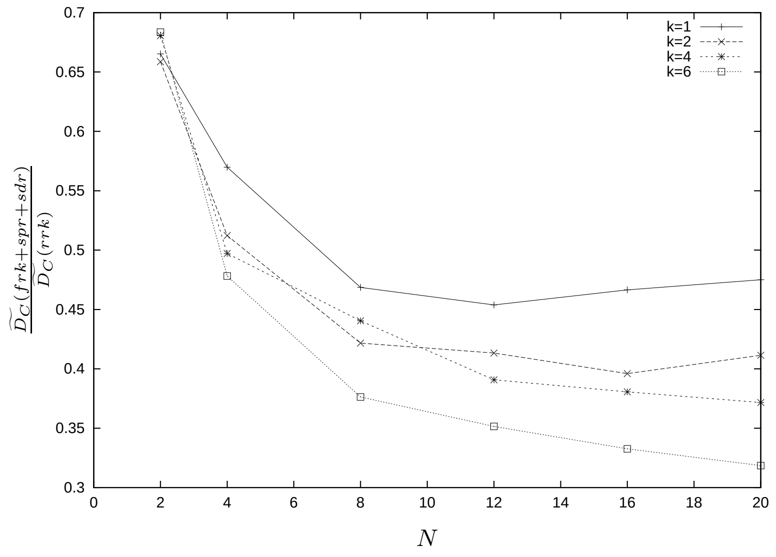

Figure 7.41: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, Semi-Markov error model and different round robin bounds k

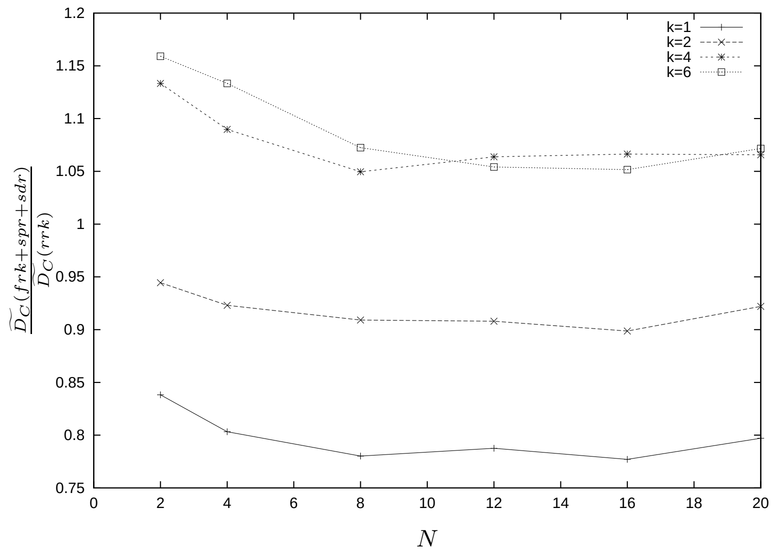

Figure 7.42: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, independent error model and different round robin bounds k

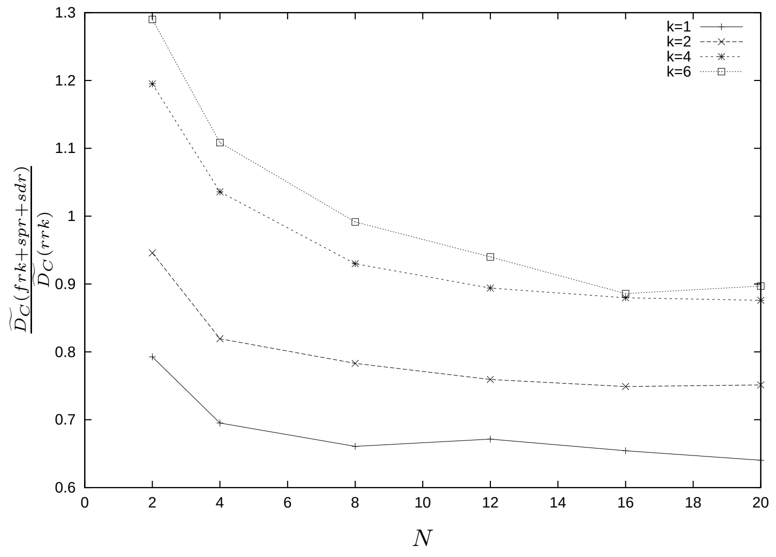

Figure 7.43: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, independent error model and different round robin bounds k

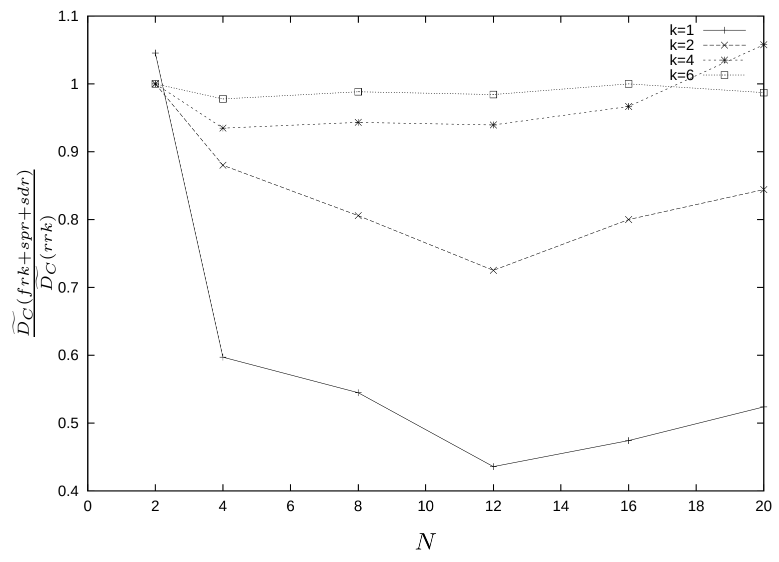

Figure 7.44: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 10% low priority load, complex error model and different round robin bounds k

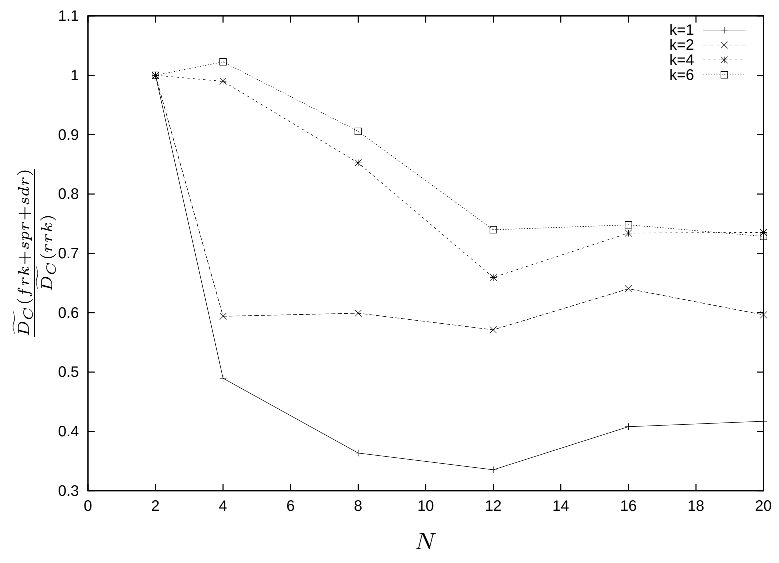

Figure 7.45: Ratio of the Overall confirmation delay for the frk+spr+sdr protocol and the rrk protocol $\frac{\widetilde{D_C}(frk+spr+sdr)}{\widetilde{D_C}(rrk)}$ vs. number of wireless terminals N for 50% low priority load, complex error model and different round robin bounds k

Summary of Results

The difference in remaining bandwidth for low priority data B_L of all three modifications as compared to rrk is always below 4%, and the SDR mechanism sometimes even gives more remaining bandwidth to the low priority data. In comparison to the significant gains (10%-60%) in $\widetilde{D_C}$ performance the slight losses in B_L performance are well tolerable.

The single modifications often achieve gains in $\widetilde{D_C}$ performance as compared to rrk. Specifically:

- The rrk+spr protocol achieves for 50% low priority loads the best gains (more than 50%) for $k \geq 2$, for $k = 1$ the gains are smaller and sometimes performance is worse. For the 10% low priority load case the gains are less impressive ($< 30\%$), and worse performance is observed more often. The rrk+spr protocol seems to work better for high loads.
- The rrk+sdr protocol is not load sensitive and achieves better gains for smaller k (up to 55% for $k = 1$ and the complex model), and worse gains or sometimes even performance losses for $k = 4$ and $k = 6$. For $k = 2$ typically 5% up to 15% gains are observed.
- The frk algorithm pays out only for $k = 1$ and $k = 2$, for higher k values it produces worse $\widetilde{D_C}$ performance due to blocking effects. For $k = 1$ the gains are due to the tendency to have the first retransmission more early than the next polling instant, for $k = 2$ the best gains occur in the complex error model, hence, in a situation where transmission problems occur not "symmetrically" on all links, but only a subset of bad links needs to be corrected.
- The combined frk+spr+sdr makes always gains over rrk for $k = 1$ and $k = 2$. For $k = 4$ and $k = 6$ we often observe losses, due to the problems with frk. When comparing frk+spr+sdr with each of rrk+spr, rrk+sdr, and frk, we find that we make almost always gains for $k = 1$ and $k = 2$, hence, combining the protocols enhances $\widetilde{D_C}$ performance as compared to every single modification.

These results justify to give the heuristic recommendation to use $k = 2$ as basic protocol parameter. This value seems to be a good compromise, since $k = 4$ and $k = 6$ suffer from blocking problems (specifically in frk) and for $k = 1$ a single request can often span over multiple token cycles, therefore increasing delays. Furthermore $k = 2$ has less protocol overhead than $k = 1$ and shows often (close to) the second-best $\widetilde{D_C}$ performance. The value $k = 2$ allows for one immediate retransmission. This seems to be a good heuristic for bursty channels: if one trial and one retransmission do not suffice, then it is good to proceed with other stations.

7.4 Polling-based MACs for wireless PROFIBUS

So far we have investigated the polling-based protocols with only loose reference to the PROFIBUS. For example, we have proposed a modified variant of the alternating bit protocol for the wireless stations, and we have not discussed the resulting integration and semantical issues so far. However, these need to be considered when looking at the integrated scenario, where a wireless segment with a polling-based protocol needs to be integrated on the MAC- and link-layer with the original PROFIBUS protocol.

7.4.1 Integration Issues

As described in Section 7.1.1 the polling protocols employ a modified version of the PROFIBUS alternating bit protocol. This modification lets every WT maintain two instances of the ABP per target station, instead of one. Specifically, a WT maintains for every priority class a separate protocol instance.

In the integrated scenario (see Section 4.3) a polling-based protocol (on the wireless side) and the PROFIBUS protocol (on the wired side) are required to run in a single PROFIBUS LAN. Hence, some kind of translation is needed, which is best performed in the BS-IWU. We assume that the base station functionality of the polling-based protocols is a part of / colocated with the BS-IWU.

Let us first consider the case of a wired master polling a wireless slave. The wired master sets the alternating bit in the frame control (FC) byte of a PROFIBUS data frame or acknowledgement frame (see Section 4.1.4). This byte includes not only the ABP information, but indicates also the priority of the frame. These informations are sufficient for the BS-IWU to construct a corresponding `data` frame on the wireless side. This can be carried out during the frame forwarding process.

The case of a wireless master polling a wired slave (e.g., a wireless plant diagnose station) is more complicated. A problem occurs, when a WT a starts with a low priority transmission to a wired slave x, and gets interrupted by an arriving high priority request for the same target x (all other targets pose no problem). By the rules of the polling protocol the WT interrupts the low priority transmission, continues with the high priority transmission and resumes the low priority transmission with its old alternating bit after finishing the high priority request. The BS-IWU is not only required to perform the forwarding of frames between the wired and wireless segments, but it should also monitor the frame exchange and perform bookkeeping of the alternating bits sent by a to x. If the BS receives at time t_0 the first high priority frame from a to x after receiving low priority frames from a to x, it marks the corresponding PROFIBUS telegram as a new one by toggling the alternating bit. The low priority frame immediately sent before t_0 is called L_1. When WT a finishes the high priority transfer (or a sequence of high priority transfers) and the BS receives a low priority frame from a to x at time t_1 (the frame is called L_2), there are two possibilities: if L_2 is distinct from L_1, the BS-IWU just marks the corresponding PROFIBUS telegram for L_2 as a new frame by toggling the alternating bit. If L_2 is a retransmission of L_1, there are three different cases:

- The BS-IWU received an ack frame from x for L_1. In this case we can assume that x's ack frame got lost on the wireless link. The BS-IWU suppresses forwarding of L_2 and repeats the ack to a instead.
- The BS-IWU receives not even a signal from x for any repetition (original frame or retransmissions) of L_1 sent out before t_0. In this case the BS-IWU can be sure that x has never properly received the frame. Hence, it can safely mark this frame as a new one by toggling the alternating bit.
- Finally, in the case where the BS-IWU has received a signal from x, but no valid ack frame to L_1 or its predecessors, it cannot know, whether x has received the low priority frame properly or not. It is a matter of policy what to do:
  - The BS-IWU can suppress L_2 and its retransmissions, in order to avoid duplicates at x. This approach makes sense, if some losses of low priority frames can be tolerated.

  – The BS-IWU can mark L_2 as a new frame, introducing the danger of duplicates at x.

Some further issues which have to be resolved in the integrated scenario are the following:

- Choice of polling activity: the BS polls WT's when the PROFIBUS token is logically in the wireless segment. It would be beneficial to use also those time slots, where only wired stations are involved in a frame exchange. The BS-IWU can determine this by inspecting all frames on the wired segment. If a data frame of sufficient length is sent from one wired station to another, the BS-IWU can initiate some parallel activity on the wireless medium. However, it must be ensured that no WT sends any frame to a wired station during this time. As an example, these time slots can be used for registration frames sent by newly arriving WT's to the BS.
- During registration the BS collects some static information from the WT's, e.g., their PROFIBUS station type (active or passive, see Section 4.1.4), vendor, product id, and so forth. This enables the BS to answer corresponding inquiry frames from wired stations on behalf of the WT's, and also to keep ring maintenance frames away from the wireless segment.
- The BS-IWU has to perform mimikry functions, as described in Section 4.3.

7.4.2 Semantics

The polling-based protocols discussed in this chapter break with the PROFIBUS semantics in some respects:

- The atomicity property is violated in order to allow high priority requests to preempt low priority requests, and due to the possibility to have $k < max\_retry$, meaning that it can take multiple token cycles to serve one request. Without preemption a low priority request can block high priority requests for long time, if the round-robin bound k is small ($k = 1$ or $k = 2$) and the *max_retry* parameter for low priority requests has moderate to large values (5 to 20). This makes *max_retry* setting in PROFIBUS a rather delicate issue, since for error-prone channels one has to find a balance between blocking times (low values) and reliability requirements for high priority requests (high values).
- Different *max_retry* parameters for low and high priority requests are introduced. This can affect PROFIBUS management.
- The `invited-poll` scheme as proposed in section 7.2.2 has the property, that it can change the sequence of confirmations: the confirmations c_1 and c_2 to the high priority requests r_1 and r_2 (with r_1 occuring before r_2) can be exchanged such that c_2 occurs before c_1.

7.5 Related Work

There exists much literature on polling systems in general, mostly concerned with their queueing analysis. A lot of references can be found in [163], the topic is treated in detail in [161], a shorter treatment can be found in [90] and [64]. Some selected references on more general polling systems are [8], where the sequence of visiting stations is determined by an arbitrary table, and [5], where the general problem of determining the state (e.g. availability of data packets) of a subset of stations over a shared medium is investigated and different methods (polling, TDMA and group testing protocols) are compared. In [29] polling systems imposing an upper bound on the time spent for a single queue are investigated using an embedded Markov chain approach. In [149] an analysis of message delays in polling systems can be found. Tree polling protocols are investigated in [170]. However, typically no transmission errors are taken into account.

The topic of packet polling systems with transmission errors is treated rarely in the literature. In [163] some results for infinite buffer polling systems with Bernoulli feedback are presented. Under Bernoulli feedback for every queue the central station serves the head of the queue. A Bernoulli experiment is performed and the customer leaves the system with some fixed probability or stays at the head of the queue. For this scenario results for the mean message response time can be found in [162]. Another reference for polling systems with Bernoulli feedback is [164]. In [199] a polling system with only downlink traffic (requested by the mobiles) is investigated under the Gilbert-Elliot error model. The focus is on the efficiency of the protocol under high loads. Specifically, the downstream queue size in the central station and the cycle times are investigated. Message delays are not considered. In [67] a table-based polling protocol is investigated under the Gilbert-Elliot error model. The protocol performs retransmissions, hence uses feedback. The polling-table accommodates synchronous and asynchronous traffic. The mean response time and bandwidth utilization vs. load are investigated using a simulation approach.

The use of polling protocols in wireless LANs is not a new idea, however, most wireless MAC protocols use contention-based approaches, even those for supporting delay sensitive data, like e.g. wireless ATM protocols. One of the most prominent protocols employing a polling scheme is the IEEE 802.11 standard with the point coordination function PCF, see Section 3.2. In reference [152] the authors propose a scheme, where the capturing phenomenon is explicitly used: data packets are transmitted in round-robin fashion with a high transmission power. But only those stations which are known to have data are included in the round-robin cycle. In parallel, the BS polls all stations at low power, and separates the parallel signals of poll answers and data frames. This scheme requires special hardware. This scheme is shown to decrease the delay.

7.6 Discussion and Future Work

In this chapter we made the very first steps towards the definition of a wireless PROFIBUS system. We have proposed and investigated several polling-based protocols for use in such a system. These protocols can be implemented using off-the-shelf components for IEEE 802.11 DSSS PHY. We have shown that even the simplest protocol, the k-limited round-robin protocol, has much better realtime/$\widetilde{D_C}$ performance than the PROFIBUS protocol, when the latter has problems with ring stability. And even when PROFIBUS ring stability is not so critical (as for the independent error model), we have seen that the advantages of the PROFIBUS protocols in $\widetilde{D_C}$ performance come at the cost of suppressing low priority transmissions.

The results suggest that the polling-based protocols are a good choice for the wireless part of an integrated PROFIBUS, even if some adaptation between wired and wireless protocols is necessary. For the bursty error models the k-limited round-robin protocol outperforms the best PROFIBUS version up to an order of magnitude.

Although the k-limited round-robin protocol is a much better choice than the PROFIBUS protocol with respect to realtime performance, it has also become clear that there is room for improvements. And indeed, the proposed protocol modifications (SDR, SPR, functional repolling) show better $\widetilde{D_C}$ performance than round-robin, at small costs in B_L performance (smaller than 4%). In some cases savings of more than 50% in $\widetilde{D_C}$ times as compared to round-robin are observed. Specifically, when the improvements are applied to the case $k = 2$ the modifications show convincing results for most of the load and error situations. When we look at the protocol with all modifications combined (frk+spr+sdr), the gains can reach up to 70%.

The modifications adapt much better to the wireless environment than the PROFIBUS protocol, by explicitly taking some of its characteristics into account. The relaying schemes make explicit use of possibly different channel error states between different pairs of stations to circumvent bursty packet losses and longer periods of bit errors. The results for the functional repolling approach (taking advantage of bursty bit errors) are not in all cases convincing, but can give gains for $k = 1$ and $k = 2$.

However, the PROFIBUS has one desirable property, that round-robin and the modified protocols don't have: its $\widetilde{D_C}$ performance is nearly insensitive to the low priority load and also, in the range investigated, to the number of stations N. In contrast, the round-robin protocols have shown to be sensitive to the low priority load and also to the number of stations. This opens potential for further research. As a starting point, we propose a small modification of the round-robin protocol approximating this behavior, see below.

A tradeoff in the k-limited round-robin protocols is the choice of k: due to the WT's simple local priority scheduler large k values lead to increased delays (and hence increased absolute $\widetilde{D_C}$ performance), but to better B_L performance. For small k values we have the opposite behavior. When the priority is on $\widetilde{D_C}$ performance, a topic of further research could be schemes for adaptively varying k: small k's for high load situations, and large k's for low load situations. When looking at the round-robin modifications, the value $k = 2$ seems to be a good heuristic value.

A basic design decision for all these protocols was to put almost all the complexity and computational burdens (channel history maintenance, packet capturing, scheduling, execution of the polling protocols, and furthermore all the other integration-related functions as e.g. the mimikry functions) into the BS/BS-IWU. This has the advantage that the WT's are kept free from these functions and can have simple and cheap implementations. A disadvantage is the introduction of a single point of failure, which requires proper redundancy schemes. A second disadvantage is the inherent inaccuracy, when the channel history information is only collected by the BS.

There are many interesting and open issues for the proposed protocols, and also for the design of future protocols:

- Round-Robin:
  - Modification of `poll` frame by introducing a new bit called *priority restriction bit*. If this bit is set, the polled WT is only allowed to perform data transmissions, if it has high priority requests available, otherwise it has to keep quiet. In addition, a proper control scheme has to be implemented within the BS. As an example, this bit can be set when the previous token cycle took more than 60% time of the maximum token cycle duration.
  - Schemes for adapting k based on load conditions.
- Relaying schemes:

  - Find heuristics for proper choice of memory-loss functions.
  - SDR: find schemes for letting other stations than the BS help in retransmitting a data frame, e.g., using a local, address-dependent timeout timer setting in each WT and the BS, as described in Section 7.2.3.
  - SPR: evaluation of alternative schemes for poll relayer selection. For example: if the best minimum-quality route is not much better than the direct way between the BS and the target WT, then skip the SPR protocol.
  - SPR: find schemes to let WT's collect channel history information and transmit it to the BS in order to have more accurate channel estimates.

- Functional repolling:
  - Development of a systematic approach for finding good repoll function sets.
  - Other types of quality functions.
  - Additional feature: in the scheme described so far a WT is interrupted by the BS upon not getting a high priority frame acknowledged, but not upon bad luck with a low priority frame. In an alternative scheme, the BS could interrupt a WT in either case. In case of a low priority request the WT remains in the poll list, in other cases it is moved into the repoll list. We anticipate reduced blocking times for successor stations.

- Investigation of other schemes, e.g., involving group testing protocols [5, 21] to reduce polling overhead.

- What happens if the assumption about not having near-far effects is not true, i.e. if it can happen that two stations send in parallel and the BS receives a decodeable signal?

- How can the polling-schemes be used with only partially meshed topologies?

It is also worthwhile to have a closer look at the assumptions we made for the protocols and the system model:

- We have assumed that an IEEE 802.11 DSSS PHY is not capable of generating short noise bursts. Here the term "short" refers to burst durations much less than a PLCP preamble of 128 μs length. Indeed, for the Harris/Intersil PRISM I chipset this is true. Let us assume that future radio modems support the following: a) generation of short bursts of, and b) capabilities for detecting these bursts without need to acquire bit synchronization. Let us furthermore assume that we can design the MAC protocols having only positive acknowledgements, and that all conditions leading to negative acknowledgements (e.g., lack of buffer space at the receiver) are signalled by some other means to the transmitter. In this case we can make `ack` frames much shorter by replacing them with a short noise bursts. However, with doing this we loose an occasion for the `ack` sender to piggyback its request queue lengths onto a frame. Another modification is to require a polled WT to generate an answer in any case. If the WT's queues are empty, it generates a short noise burst. In the proposed protocols we avoided to always generate `null` frames for performance reasons, and have introduced the `explicit-poll` frame to resolve the ambiguity between a WT a having no data and the BS having a bad channel to WT a. When the WT is required to send short noise bursts upon empty queues, this ambiguity is resolved, and the `explicit-poll` cycle could be dropped. If the BS does not receive the noise burst for some subsequent polls, it can invoke the SPR protocol directly.

- We have assumed the channels between different pairs of stations to be independent, and for a fixed pair of stations to be symmetric (see Section 6.8.3). Although this assumptions seems to be reasonable, it should be investigated by detailed measurements.
- The assumption of having no propagation delay is not critical, since for small cell sizes (30-50 m) it is a good approximation. In practice, the propagation delay has to be considered by introducing small guard times. These guard times do not affect the protocols proposed here, as long as they are significantly lower than the time unit of 64 μs.

Clearly, besides polling-related control knobs there are many other means of achieving better transmission reliability and realtime performance. Among these methods are the variation of transmit power, modulation schemes, better preamble acquisition methods to avoid packet losses, and more. An interesting area for further research are different framing and coding methods (FEC, interleaving schemes). Some sample schemes are:

- Usage of FEC: always FEC'ing the MAC header, and only FEC'ing a `data` frame's data part on occasion of retransmissions. However, this does not help for packet losses.
- A `data` frame's data part can be split into several chunks, where each chunk is equipped with a separate checksum. Correctly received chunks are stored in a buffer, and from retransmitted frames only the missing chunks need to be successful. This way, the retransmission frame may have bit errors, but when the missing chunks are correct, these do not hurt. By this method the number of needed retransmissions can be reduced at the cost of a slightly increased overhead.

The proposed protocols form only a first step towards the definition of a wireless PROFIBUS capable of supporting the integrated scenario. Some of the next steps are:

- Formal specification and validation of the polling-based protocols, proof of the interoperability between the PROFIBUS protocol and the polling-based protocols.
- Investigation of schemes and protocols for mobility support and the associated registration overhead and registration delay, redundancy / fault tolerance, management, security / authentication, power saving, and hardware implementations.
- Design of a prototypical BS-IWU implementation and field tests.

Although there is much work to do on the way to a wireless PROFIBUS, the proposed approach of using a specifically tailored, polling-based MAC protocol on the wireless side and to integrate this on the MAC and link-layer level with the existing protocol, shows up to be feasible and promising.

Chapter 8

Conclusions and Outlook

A wireless PROFIBUS would be an attractive choice for many applications. However, we are faced to the problem of having hard realtime requirements on the one hand, and having an error-prone and time-variable transmission medium on the other hand. These characteristics of wireless links call for different approaches for design of protocols and transmission schemes than for other types of links. This is especially true for wireless fieldbus systems in general, and more specifically for a wireless PROFIBUS system.

One of the first steps towards a wireless PROFIBUS is the choice of a wireless transmission technology. In this thesis we have taken the leading IEEE 802.11 WLAN technology with DSSS PHY as the basis. It shows up, however, that for use in a wireless PROFIBUS we should only adopt the PHY of 802.11 and drop the MAC layer. The next question is which MAC and link-layer protocol to run on top of the 802.11 DSSS PHY. Taking the error-prone nature of a wireless link in general and specifically the measurement results for an 802.11 PHY into consideration, this thesis provides an answer to this question: the PROFIBUS protocol, a quite obvious candidate, is not well suited for this environment (even not with some modifications) and should be replaced by a specifically tailored protocol. To show this, we have used the notion of realtime performance, which is a set of measures reflecting timing constraints and reliability requirements for the case of error-prone wireless links. We have shown that the PROFIBUS MAC protocols has serious problems with the stability of the logical ring. Since only ring members are allowed to transmit data, these problems lead to poor realtime performance.

In this thesis we suggested polling-based protocols as a starting point for the design of specifically tailored MAC and link-layer protocols. The simplest protocol, a k-limited round-robin protocol, shows under bursty errors a much better realtime performance than the PROFIBUS protocol, the difference reaches up to an order of magnitude.

Furthermore, we have proposed three modifications of round-robin, which can improve the realtime performance significantly (more than 50%) under many circumstances. For the design of these modifications the properties of the wireless link were taken into account, specifically its burstiness and spatial diversity. The search for further improvements and other types of protocols with good realtime performance constitutes a field for further research.

When one wants to integrate wired and wireless stations into a single PROFIBUS LAN, the approach of using a specifically tailored MAC protocol on the wireless side while maintaining the existing protocol for the wired stations is a far-reaching design decision. It requires some MAC-/link-layer translation between both sides. The design space is large and worth future exploration.

A way to avoid this design decision would be to find further improvements of the PROFIBUS protocol, which help to better overcome the deficiencies of this protocol on wireless-type links. However, we believe that the potential of this approach is limited, specifically when keeping the need for interoperability with wired stations (and the unchanged protocol) in mind.

Although the results achieved in this thesis are promising, they form only the first steps towards a wireless PROFIBUS integrating wired and wireless stations in a single LAN. Many steps have to follow. Some of the next steps are the investigation of other modulation schemes and PHY technologies (OFDM is an interesting candidate) and the impact of mobility on the wireless PROFIBUS protocol stack and also on the achievable realtime performance.

There is another aspect, which is not only important for wireless PROFIBUS, but also for another interesting class of networks with realtime requirements, namely, the class of *sensor networks* (e.g., the PicoRadio system developed at the Berkeley Wireless Research Center [142]). The issue of power saving and minimizing energy consumption is a central design goal for these systems. It adds another dimension to the tradeoff between realtime requirements on the one hand and the error behavior of a wireless link on the other hand.

In summary, this thesis has made the following contributions:

- We have shown that the PROFIBUS MAC and link layer protocol has serious problems with the stability of the logical ring over lossy / wireless links. We have proposed two protocol improvements, which can significantly relax the problems on wired-type, error-prone links, but on wireless-type links the ring stability is still unsatisfactorily, which in turn leads to degraded realtime performance.
- We suggested to use specifically tailored MAC- and link layer protocols on top of an IEEE 802.11 DSSS PHY to replace the PROFIBUS protocol. We advocated the class of polling-based protocols as a candidate.
- We have argued that constructing a mapping between PROFIBUS link layer services and the IEEE 802.11 DCF/PCF MAC protocol is not a viable solution.
- We have given a characterization and approaches to stochastic modeling of a wireless link using real-word traces taken in an industrial environment. The results were used:
  - as design input for polling-based MAC protocols.
  - for parameterization of popular stochastic link error models.

  A new class of stochastic models with increased modeling precision at moderate complexity costs was proposed.
- We have defined the notion of *realtime performance*, which takes timeliness and reliability over wireless-type links into account. This notion provides an important optimization goal for wireless MAC protocols targeted at hard realtime / industrial applications.
- We have presented approaches for polling-based protocols and shown that:

  - A simple k-limited round-robin protocol shows substantially better realtime performance than the PROFIBUS protocol under several error assumptions. The difference can reach up to an order of magnitude.
  - The realtime performance of k-limited round-robin can be further improved. We have proposed three modifications of round-robin. We have evaluated these modifications under several error and load assumptions, and shown that they achieve under many circumstances a significantly ($> 50\%$) better realtime performance than round-robin.

- These results justify the suggestion to replace the PROFIBUS protocol by another protocol.

Appendix A

Main Characteristics of the Bipartite Model

It is convenient to remind the necessary definitions for the bipartite model. Let n_1 be the number of "bad" states, n_2 the number of good states, and $n = n_1 + n_2$. For simplicity the states are numbered from 1 to n. The transition matrix $\mathbf{P}$ has the form

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{Q}_1 \\ \mathbf{Q}_2 & \mathbf{0} \end{pmatrix}$$

with $\mathbf{Q}_1$ being a $n_1 \times n_2$ stochastic matrix, and $\mathbf{Q}_2$ being a $n_2 \times n_1$ stochastic matrix.

To every state i there is associated an interval I_i of the natural numbers, and a probability distribution $p_i(k) = \Pr[p_i = k]$ (with $k \in \mathbb{N}$ and $p_i(k) = 0$ for $k \notin I_i$) generating values in this interval. This probability distribution has the distribution function $F_i(x) = \Pr[p_i \leq x]$.

A.1 Asymptotic Behavior

From the description in Section 6.7.2 it is easy to see that $\mathbf{P}$ generates a periodic Markov chain with period 2, and thus has no steady-state. But the bipartite model satisfies a weaker form of steady-state condition. In this section we show that in the long run under certain assumptions for each state i the fraction of the number of visits in state i w.r.t. the total number k of state transitions so far converges with probability one to a fixed value a_i.

For the following we need the observation that for $k \in \mathbb{N}_0$ we have:

$$\mathbf{P}^{2k} = \begin{pmatrix} (\mathbf{Q}_1 \cdot \mathbf{Q}_2)^k & \mathbf{0} \\ \mathbf{0} & (\mathbf{Q}_2 \cdot \mathbf{Q}_1)^k \end{pmatrix}$$

and

$$\mathbf{P}^{2k+1} = \begin{pmatrix} \mathbf{0} & \mathbf{Q}_1 \cdot (\mathbf{Q}_2 \cdot \mathbf{Q}_1)^k \\ \mathbf{Q}_2 \cdot (\mathbf{Q}_1 \cdot \mathbf{Q}_2)^k & \mathbf{0} \end{pmatrix}$$

which can easily be proved by induction. We assume that the Markov chains generated by $\mathbf{Q}_1 \cdot \mathbf{Q}_2$ and $\mathbf{Q}_2 \cdot \mathbf{Q}_1$ are ergodic (i.e., aperiodic and positive recurrent) and thus the limits $\mathbf{A} := \lim_{k\to\infty}(\mathbf{Q}_1 \cdot \mathbf{Q}_2)^k$

and $\mathbf{B} := \lim_{k\to\infty}(\mathbf{Q}_2 \cdot \mathbf{Q}_1)^k$ exist (clearly, $\mathbf{A}$ and $\mathbf{B}$ are also stochastic matrices). The matrices $\mathbf{A}$ and $\mathbf{B}$ have the specific feature that in every row all elements have the same value [115, chap. 8]. Using this, and the fact that $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are stochastic matrices, it is easy to see that $\mathbf{A}' := \mathbf{Q}_2 \cdot \mathbf{A}$ has the same number of rows as $\mathbf{A}$, and again all elements of a single row have the same value, namely the value that the corresponding row of $\mathbf{A}$ has. An analogous property holds for $\mathbf{B}' := \mathbf{Q}_1 \cdot \mathbf{B}$.

Let $n := n_1 + n_2$, ${\pi_0}^T \in \mathbb{R}^n$ with $\pi_0 = (\pi_1^0, \dots, \pi_n^0)$ a stochastic vector describing the initial state distribution, let $z_0 z_1 z_2 z_3 \dots$ be a sample path of the Markov chain (i.e. $z_j \in \{s_1, \dots, s_n\}$), and $\mathbf{e}_1, \dots, \mathbf{e}_n$ denote the unit vectors of $\mathbb{R}^n$. For simplicity we assume that the states s_1 to s_n are given by the natural numbers from 1 to n. Furthermore, denote $\mathbf{1}_S(x)$ the indicator function of the set S, i.e. $\mathbf{1}_S(x) = 1$ if $x \in S$ and 0 otherwise. Define the vector $\mathbf{a}_k$ by

$$\mathbf{a}_k = \mathbf{e}_{z_0} + \sum_{i=1}^{k} \sum_{j=1}^{n} \mathbf{1}_{\{s_j\}}(z_i)\mathbf{e}_j = \mathbf{e}_{z_0} + \sum_{j=1}^{n} \mathbf{e}_j \sum_{i=1}^{k} \mathbf{1}_{\{s_j\}}(z_i)$$

i.e. it counts in coordinate j of $\mathbf{a}_k$ how often the system was in state j during the first k state transitions of the given sample path. We define $\mathbf{a}'_k := \frac{1}{k}\mathbf{a}_k$ and interpret the j-th coordinate of $\mathbf{a}'_k$ as the fraction of the number of visits in state j w.r.t. the total number k of state transitions so far. We write the j-th coordinate of a vector $\mathbf{a}$ as $[\mathbf{a}]_j$, and for a matrix $\mathbf{O}$ the matrix element on the i-th row and j-th column is written as $[[\mathbf{O}]]_{i,j}$. The long-term fraction of visits in state j (exemplarily we choose $j = 1$) is then defined as:

$$a_1 := \lim_{k\to\infty} [\mathbf{a}'_k]_1 = \lim_{k\to\infty} \frac{1}{k} \left([\mathbf{e}_{z_0}]_1 + \sum_{i=1}^{k} \mathbf{1}_{\{s_1\}}(z_i) \right)$$

provided the limit exists. It is convenient to rewrite this, in order to drop the influence of the start of the sample path. Let $\nu \in \mathbb{N}$ fixed, for $1 < 2\nu < k$ we can write:

$$[\mathbf{a}'_k]_1 = \frac{1}{k} \left([\mathbf{e}_{z_0}]_1 + \sum_{i=1}^{2\nu-1} \mathbf{1}_{\{s_1\}}(z_i) \right) + \frac{1}{k} \left(\sum_{i=2\nu}^{k+2\nu} \mathbf{1}_{\{s_1\}}(z_i) \right) - \frac{1}{k} \left(\sum_{i=k+1}^{k+2\nu} \mathbf{1}_{\{s_1\}}(z_i) \right) \tag{A.1}$$

For k large enough we can drop the first and last term of Equation A.1, since in both cases the sum in the braces is $\le 2\nu$ and $\frac{2\nu}{k} \longrightarrow 0$ for $k \to \infty$. Without loss of generality we assume k to be even, $k = 2l$ and $l > 2\nu$. Then we can split the sum in the middle term of Equation A.1 into even and odd terms:

$$[\mathbf{a}'_k]_1 = \frac{1}{2l} \left(\sum_{i=0}^{l} \mathbf{1}_{\{s_1\}}(z_{2\nu+2i}) + \sum_{i=0}^{l} \mathbf{1}_{\{s_1\}}(z_{2\nu+2i+1}) \right) \tag{A.2}$$

The terms in the first sum are independent random variables, the same holds for the second sum. We consider the first sum, denoted as $S_l = \sum_{i=0}^{l} \mathbf{1}_{\{s_1\}}(z_{2\nu+2i})$. For every random variable $\mathbf{1}_{\{s_1\}}(z_{2\nu+2i})$ we have that

$$\Pr[\mathbf{1}_{\{s_1\}}(z_{2\nu+2i}) = 1] = \left[\pi_0 \cdot \mathbf{P}^{2\nu+2i}\right]_1 = \left[\pi_0 \cdot \begin{pmatrix} (\mathbf{Q}_1 \cdot \mathbf{Q}_2)^{\nu+i} & \mathbf{0} \\ \mathbf{0} & (\mathbf{Q}_2 \cdot \mathbf{Q}_1)^{\nu+i} \end{pmatrix}\right]_1$$

An analogous equation holds for the second sum. Since we have assumed the existence of $\mathbf{A}$ and $\mathbf{B}$, for ν large enough we can replace $(\mathbf{Q}_1 \cdot \mathbf{Q}_2)^{\nu+i}$ by $\mathbf{A}$ and $(\mathbf{Q}_2 \cdot \mathbf{Q}_1)^{\nu+i}$ by $\mathbf{B}$. Now each random variable $\mathbf{1}_{\{s_1\}}(z_{2\nu+2i})$ is an independent Bernoulli random variable with fixed probability

$$c := \Pr[\mathbf{1}_{\{s_1\}}(z_{2\nu+2i}) = 1] = \left[\pi_0 \cdot \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}\right]_1$$

Thus the first sum of equation A.2 is a sum of independent Bernoulli random variables. The strong law of large numbers [47, chap. 8] asserts that for $l \to \infty$ the random variable $\frac{S_l}{l}$ converges to $\mathrm{E}\left[\mathbf{1}_{\{s_1\}}(z_{2\nu+2i})\right] = c$ with probability one, i.e., for almost all sample paths. The same calculations can be done for the second sum of equation A.2. Putting this together, it is shown that

$$a_1 = \lim_{k\to\infty} [\mathbf{a}'_k]_1 \xrightarrow{a.s.} \frac{1}{2}\left(\left[\pi_0 \cdot \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}\right]_1 + \left[\pi_0 \cdot \begin{pmatrix} \mathbf{0} & \mathbf{B}' \\ \mathbf{A}' & \mathbf{0} \end{pmatrix}\right]_1\right)$$

where $\xrightarrow{a.s.}$ denotes *almost sure* convergence, i.e., convergence with probability one.

It remains to show that the values a_1 to a_n are nonnegative and sum up to one. The nonnegativity follows immediately from the fact that all coefficients of $\mathbf{A}$, $\mathbf{B}$, $\mathbf{A}'$, $\mathbf{B}'$, and π_0 are nonnegative by definition. The sum of a_1 to a_n is given as

$$\begin{aligned}
\sum_{i=1}^{n} a_i &= \frac{1}{2}\sum_{i=1}^{n}\left(\left[\pi_0 \cdot \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}\right]_i + \left[\pi_0 \cdot \begin{pmatrix} \mathbf{0} & \mathbf{B}' \\ \mathbf{A}' & \mathbf{0} \end{pmatrix}\right]_i\right) = \frac{1}{2}\left(\sum_{i=1}^{n}\left[\pi_0 \cdot \begin{pmatrix} \mathbf{A} & \mathbf{B}' \\ \mathbf{A}' & \mathbf{B} \end{pmatrix}\right]_i\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{n_1}\left[\pi_0 \cdot \begin{pmatrix} \mathbf{A} & \mathbf{B}' \\ \mathbf{A}' & \mathbf{B} \end{pmatrix}\right]_i + \sum_{i=n_1+1}^{n_2}\left[\pi_0 \cdot \begin{pmatrix} \mathbf{A} & \mathbf{B}' \\ \mathbf{A}' & \mathbf{B} \end{pmatrix}\right]_i\right) \\
&= \frac{1}{2}\left(\sum_{i=1}^{n_1}[[\mathbf{A}]]_{1,i} + \sum_{i=n_1+1}^{n_2}[[\mathbf{B}]]_{1,i-n_1}\right) = \frac{1}{2}(1+1) = 1
\end{aligned}$$

In the last step we have used that for each row of $\mathbf{A}$ and $\mathbf{B}$ all elements have the same value, and that these values are the same as for $\mathbf{A}'$ and $\mathbf{B}'$.

If we have the values a_1 up to a_n it is straightforward to compute e.g., the mean bit error rate, the mean error burst length or the mean error-free burst lengths from the process generated by the bipartite model. If $P_i := \mathrm{E}[p_i]$ is the mean state holding time of state i, then the long-term fraction of time the system is in state i is given by a_iP_i, and the mean bit error rate is given by

$$\bar{m} = \frac{\sum_{i=1}^{n} r_i a_i P_i}{\sum_{i=1}^{n} a_i P_i}$$

However, a drawback of the bipartite model is that these mean values cannot be represented in a "simple" and "intuitive" manner from the values of $\mathbf{P}$, as is the case for the Gilbert/Elliot model.

A.2 Distribution of Generated Burst Lengths

The following calculation shows that the distribution functions $F_X(\cdot)$ and $F_Y(\cdot)$ of the error-free burst length and error burst length distributions can be approximated with arbitrary precision by choosing proper number of states and distribution functions for the single intervals. We show this only for the error burst lengths, the calculations for the error-free burst lengths are identical.

The distribution function of the generated error burst lengths Y' in step k can be calculated with the law of total probability:

$$\begin{aligned}
\Pr[Y' \le y | z_k \in \{1,\ldots,n_1\}] &= \frac{\Pr[Y' \le y \cap (\{z_k = 1\} \cup \ldots \cup \{z_k = n_1\})]}{\Pr[\{z_k = 1\} \cup \ldots \cup \{z_k = n_1\}]} \\
&= \frac{\Pr[Y' \le y \cap \{z_k = 1\}] + \ldots + \Pr[Y' \le y \cap \{z_k = n_1\}]}{\Pr[z_k = 1] + \ldots + \Pr[z_k = n_1]} \\
&= \frac{\Pr[Y' \le y | z_k = 1] \Pr[z_k = 1] + \ldots + \Pr[Y' \le y | z_k = n_1] \Pr[z_k = n_1]}{\Pr[z_k = 1] + \ldots + \Pr[z_k = n_1]} \\
&= \frac{F_1(y) \Pr[z_k = 1] + \ldots + F_{n_1}(y) \Pr[z_k = n_1]}{\Pr[z_k = 1] + \ldots + \Pr[z_k = n_1]}
\end{aligned}$$

Hence, the distribution of the generated error burst lengths Y' can be represented in a simple manner as a linear combination of the approximating distributions $F_i(\cdot)$. By properly selecting the $F_i(\cdot)$ the distribution $F_Y(\cdot)$ can be well approximated.

A.3 Correlation Properties

In order to show that the state process generated by $\mathbf{P}$ has short-term or fast decaying correlation properties, we assume that $\mathbf{P}$ is diagonalizable, i.e. there exists two $n \times n$ matrices $\mathbf{R}$ and $\mathbf{D}$ such that $\mathbf{R}^{-1}$ exists and $\mathbf{D}$ is a diagonal matrix with the eigenvalues λ_i of $\mathbf{P}$ on the diagonale and $\mathbf{P} = \mathbf{R}^{-1} \cdot \mathbf{D} \cdot \mathbf{R}$ holds.[1] Since $\mathbf{P}$ is a stochastic matrix, for all eigenvalues $|\lambda_i| \le 1$ holds [158, chapter 1.6]. Then the autocorrelation function of the generated process can be represented as

$$\begin{aligned}
R(k) &= \mathrm{E}[z_0 z_k] = \sum_{i=1}^{n} \sum_{j=1}^{n} ij \Pr[z_0 = i, z_k = j] = \sum_{i=1}^{n} \sum_{j=1}^{n} ij \Pr[z_0 = i] \Pr[z_k = j | z_0 = i] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} ij \, [\pi_0]_i \left[e_i \cdot \mathbf{P}^k\right]_j = \sum_{i=1}^{n} \sum_{j=1}^{n} ij \, [\pi_0]_i \left[e_i \cdot \mathbf{R}^{-1} \cdot \mathbf{D}^k \cdot \mathbf{R}\right]_j
\end{aligned}$$

The influence of the eigenvalues λ_i with $|\lambda_i| < 1$ play a role for small k and vanish for $k \to \infty$. The eigenvalue $\lambda_1 = 1$ contributes the "mean" value of the generated state process.

Now we show that in the asymptotic case the generated process is becoming independent from its start state z_0, hence it looses correlation. To do this we have to show that for k large enough $\mathrm{E}[z_0 z_k] = \mathrm{E}[z_0]\,\mathrm{E}[z_k]$ holds.

For calculating the long-term correlation we first observe that for the case of k even and large enough we have

$$\begin{aligned}
\mathrm{E}[z_0 z_k] &= \sum_{i=1}^{n} \sum_{j=1}^{n} ij \, [\pi_0]_i \left[e_i \cdot \mathbf{P}^k\right]_j = \sum_{i=1}^{n} \sum_{j=1}^{n} ij \, [\pi_0]_i \left[e_i \cdot \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}\right]_j \\
&= \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} ij \, [\pi_0]_i \, [[\mathbf{A}]]_{1,j} + \sum_{i=n_1+1}^{n} \sum_{j=n_1+1}^{n} ij \, [\pi_0]_i \, [[\mathbf{B}]]_{1,j-n_1}.
\end{aligned}$$

[1] The assumption that $\mathbf{P}$ is diagonalizable is just for simplicity. Without this, the calculations are more involved. An approach would be to switch into the complex domain and use the Jordan Normal form of $\mathbf{P}$ instead.

In the last equation we have used that for each row of $\mathbf{A}$ and $\mathbf{B}$ all elements have the same value. On the other hand we have

$$\begin{aligned}
\mathrm{E}[z_0]\,\mathrm{E}[z_k] &= \left(\sum_{i=1}^{n} i\,[\pi_0]_i\right)\cdot\left(\sum_{j=1}^{n} j\left[\pi_0\cdot\begin{pmatrix}\mathbf{A} & \mathbf{0}\\ \mathbf{0} & \mathbf{B}\end{pmatrix}\right]_j\right)\\
&= \left(\sum_{i=1}^{n} i\,[\pi_0]_i\right)\cdot\left(\sum_{j=1}^{n_1} j[[\mathbf{A}]]_{1,j}\sum_{l=1}^{n_1}[\pi_0]_l + \sum_{j=n_1+1}^{n} j[[\mathbf{B}]]_{1,j-n_1}\sum_{l=n_1+1}^{n}[\pi_0]_l\right)
\end{aligned}$$

Using both equations, it is straightforward to verify that for all possible system start states $\pi_0 = \mathbf{e}_\nu$ with $\nu \in \{1,\ldots,n\}$ the relation $\mathrm{E}[z_0 z_k] = \mathrm{E}[z_0]\,\mathrm{E}[z_k]$ holds and thus there is no correlation. Furthermore, this calculation holds also true for k odd. Hence, there is no long term correlation.

Appendix B

Applicability of Simple FEC Schemes to the Measurement Traces

This appendix is an addendum to the evaluations of the measurement traces presented in Chapter 6. We investigate here the feasibility of simple block FEC schemes, when applied to the measurement results.

In block FEC schemes [97] a block of k user bits is mapped onto n code bits, with $n > k$. For general block codes the *Hamming Bound* [97, chap. 3] applies, which states that up to t errors can be corrected in a codeword of n bits length and k user bits, only if the following relation holds:

$$2^{n-k} \geq \sum_{i=0}^{t} \binom{n}{i}$$

The fact that a triple $(n, k, t) \in \mathbb{N}^3$ satisfies this relation, does not imply that a code with this properties really exists. The ratio $\frac{k}{n}$ is denoted as the *code rate*. In the following, we restrict to the case of $n \in \{8, \dots, 32\}$. This restriction is somewhat arbitrary but can be justified by the observation that in industrial communications frequently very small packets are used (e.g., short PROFIBUS frames), i.e. k is often small.

With BPSK modulation in most cases a single bit error is surrounded by many correct bits (see Section

	Mean PER	**Max. PER**
BPSK w/o scrambling	0.9%	6%
BPSK w/ scrambling	6.4%	25.6%
QPSK w/o scrambling	7.7%	20.7%
QPSK w/ scrambling	3.2%	14.7%

Table B.1: Mean packet error rate (PER) and max. PER for QPSK and BPSK modulation and different scrambling modes

6.5.6). Hence, it suffices to look at the case $t = 1$. By the Hamming bound, the best achievable code rate $\frac{k}{n}$ for $t = 1$ is $\approx 84\%$ ($(n, k, t) = (31, 26, 1)$). Stated differently, if every packet is transmitted with FEC, the overhead is at least 16% for $t = 1$. In Table B.1 we show the mean and maximum packet error rate (PER) (all packets with at least one bit error) for BPSK and QPSK with and without scrambling.[1] For the investigated BPSK traces the PER is below 5.5%. Hence, applying FEC to all packets is wasteful and should be restricted to retransmissions.

For QPSK there often occur 14 or 16 bits long bursts with two bit errors (see Section 6.5.6), hence, we consider the case $t = 2$. The best code rate achievable for $t = 2$ and $n \in \{8, \ldots, 32\}$ is $\approx 71\%$ ($(n, k, t) = (31, 22, 2)$). Again, comparing with the PERs reported in Table B.1 it is easy to see that applying FEC to all packets is wasteful.

To summarize, the findings about error densities and the fact that error-free burst lengths are sometimes very long, suggest that FEC should be enabled only for retransmissions. Furthermore, since errors show longer term correlation, FEC should stay enabled for a while (suitable history information should be used).

[1] One would expect increasing PERs for increasing packet sizes. This is only true for QPSK without scrambling.

Bibliography

[1] Imad Aad and Claude Castellucia. Introducing Service Differentiation into IEEE 802.11. In *Proc. Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, Antibes, France, July 2000.

[2] Richard L. Abrahams. *2.4GHz 11Mbps MACless DSSS Radio HWB1151 Users Guide - AN9835.1.* Intersil, 1999.

[3] G. Agrawal, B. Chen, W. Zhao, and S. Davari. Guaranteeing synchronous message deadlines with the timed token medium access control protocol. *IEEE Transactions on Computers*, 43(2):327 – 339, March 1994.

[4] Lars Ahlin and Jens Zander. *Principles of Wireless Communications.* Studentlitteratur, Lund, Sweden, 1998.

[5] Mostafa H. Ammar and George N. Rouskas. On the performance of protocols for collecting responses over a multiple-access channel. *IEEE Transactions on Communications*, 43(2):412–420, February 1995.

[6] Guiseppe Anastasi, Luciano Lenzini, Enzo Mingozzi, Andreas Hettich, and Andreas Krämling. Mac protocols for wideband wireless local access: Evolution towards wireless atm. *IEEE Personal Communications*, 5(5):53–64, October 1998.

[7] Jorgen Bach Andersen, Theodore S. Rappaport, and Susumu Yoshida. Propagation Measurements and Models for Wireless Communications Channels. *IEEE Communications Magazine*, 33(1):42–49, January 1995.

[8] Joseph E. Baker and Izhak Rubin. Polling with a general-service order table. *IEEE Transactions on Communications*, 35(3):283–288, March 1987.

[9] David F. Bantz and Frederic J. Bauchot. Wireless lan design alternatives. *IEEE Network Magazine*, 8(3):43ff, 1994.

[10] K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson. A note on reliable full-duplex transmission over half duplex lines. *Communications of the ACM*, 12(5):260ff, 1969.

[11] Klaus Bender. *PROFIBUS - Der Feldbus für die Automation*, volume 2. Carl Hanser Verlag, München, second edition, 1992.

[12] Michael Berry, Andrew T. Campbell, and Andras Veres. Distributed control algorithms for service differentiation in wireless packet networks. In *Proc. INFOCOM 2001*, Anchorage, Alaska, April 2001. IEEE.

[13] H.-P. Beuerle and G. Bach-Bezenar. *Kommunikation in der Automatisierungstechnik.* Siemens Aktienges., Berlin; München, 1991.

[14] Pravin Bhagwat, Partha Bhattacharya, Arvind Krishna, and Satish K. Tripathi. Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs. *Wireless Networks*, 3(1):91–102, March 1997.

[15] Guiseppe Bianchi. Throughput Evaluation of the IEEE 802.11 Distributed Coordination Function. In *Proc. Fifth International Workshop on Mobile Multimedia Communication (MoMuC'98)*, pages 307–318, Berlin, Germany, 1998.

[16] Janos Bito. *Digitale Mobilfunk-Kanalmodelle unter besonderer Berücksichtigung von adaptiven digitalen Modellen.* Dissertation, Technische Universität Berlin, Department of Electrical Engineering, December 1996.

[17] Kenneth L. Blackard, Theodore S. Rappaport, and Charles W. Bostian. Measurements and models of radio frequency impulsive noise for indoor wireless communications. *IEEE Journal on Selected Areas in Communications*, 11(7):991–1001, September 1993.

[18] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis – forecasting and control.* Holden-Day, San Francisco, 3 edition, 1994.

[19] Paul T. Brady. A model for generating on-off speech patterns in two-way conversation. *Bell Systems Technical Journal*, 48, September 1969.

[20] F. Cali, M. Conti, and E. Gregori. Ieee 802.11 wireless lan: Capacity analysis and protocol enhancement. In *Proc. INFOCOM 1998*, San Francisco, April 1998. IEEE.

[21] J. I. Capetanakis. Tree Algorithm for Packet Broadcast Channels. *IEEE Transactions on Information Theory*, 25(5):505–515, September 1979.

[22] S. Cavalieri and D. Panno. On the integration of fieldbus traffic within ieee 802.11 wireless lan. In *Proc. 1997 IEEE International Workshop on Factory Communication Systems (WFCS'97), Barcelona (Spain)*, 1993.

[23] James K. Cavers. *Mobile Channel Characteristics.* Kluwer Academic Publishers, Boston, Dordrecht, 2000.

[24] CCITT. *Recommendation Z.100: Specification and Description Language SDL.* ITU General Secretariat, 1988.

[25] Cheng-Shang Chang, Kwang-Cheng Chen, Ming-Young You, and Jin-Fu Chang. Guaranteed quality-of-service wireless access to atm networks. *IEEE Journal on Selected Areas in Communications*, 15(1):106–118, January 1997.

[26] Charles Chien, Mani B. Srivastava, Rajeev Jain, Paul Lettieri, Vipin Aggarwal, and Robert Sternowski. Adaptive Radio for Multimedia Wireless Links. *IEEE Journal on Selected Areas in Communications*, 17(5):793–813, May 1999.

[27] Brian P. Crow, Indra Widjaja, Jeong Geun Kim, and Prescott T. Sakai. IEEE 802.11 wireless local area networks. *IEEE Communications Magazine*, 35(9):116–126, September 1997.

[28] Klaus David and Thorsten Benkner. *Digitale Mobilfunksysteme.* Informationstechnik. B.G. Teubner, Stuttgart, 1996.

[29] Edmundo de Souza e Silva, H. Richard Gail, and Richard R. Muntz. Polling systems with server timeouts and their application to token passing networks. *IEEE/ACM Transactions on Networking*, 3, October 1995.

[30] Jean-Dominique Decotignie and Patrick Pleineveaux. A survey on industrial communication networks. *Ann. Telecomm.*, 48(9):435ff, 1993.

[31] Dr-Jiunn Deng and Ruay-Shiung Chang. A priority scheme for ieee 802.11 dcf access method. *IEICE Transactions on Communications*, E82-B(1):96–102, January 1999.

[32] H. Dietsch. Feldbus. *Informatik Spektrum*, 13:217ff, 1990.

[33] DIN - Deutsches Institut für Normung, Beuth Verlag Berlin. *DIN 19245 Teil 1 - PROFIBUS: Übertragungstechnik, Buszugriffs- und Übertragungsprotokoll, Dienstschnittstelle zur Anwendungsschicht, Management*, April 1991.

[34] DIN - Deutsches Institut für Normung, Beuth Verlag Berlin. *DIN 19245 Teil 2 - PROFIBUS: Kommunikationsmodell, Dienste für die Anwendung, Protokoll, Syntax, Codierung, Schnittstelle zur Schicht 2, Management*, April 1991.

[35] DIN - Deutsches Institut für Normung, Beuth Verlag Berlin. *PROFIBUS-DP - Process Field Bus Decentralised Periphery (DP) - Part 3, Draft Standard DIN 19245*, April 1993.

[36] DIN - Deutsches Institut für Normung, Beuth Verlag Berlin. *DIN 19258 Teil 1 - INTERBUS-S, Sensor-/Aktornetzwerk für industrielle Steuerungssysteme – System-Architektur*, May 1994. Entwurf.

[37] DIN - Deutsches Institut für Normung, Beuth Verlag Berlin. *DIN 19258 Teil 2 - INTERBUS-S, Sensor-/Aktornetzwerk für industrielle Steuerungssysteme – Physical Layer (Bitübertragungsschicht)*, May 1994. Entwurf.

[38] DIN - Deutsches Institut für Normung, Beuth Verlag Berlin. *DIN 19258 Teil 3 - INTERBUS-S, Sensor-/Aktornetzwerk für industrielle Steuerungssysteme – Data Link Layer (Sicherungsschicht)*, May 1994. Entwurf.

[39] D. Duchamp and N.F.Reynolds. Measured performance of wireless lan. In *Proc. of 17th Conf. on Local Computer Networks, Minneapolis*, 1992.

[40] David Eckhard and Peter Steenkiste. Measurement and analysis of the error characteristics of an in-building wireless network. In *Proc. of ACM SIGCOMM'96 Conference,*, pages 243–254, Stanford University, California, August 1996.

[41] David A. Eckhardt and Peter Steenkiste. A trace-based evaluation of adaptive error correction for a wireless local area network. *MONET - Mobile Networks and Applications*, 4:273–287, 1999.

[42] E. O. Elliot. Estimates of error rates for codes on burst-noise channels. *Bell Systems Technical Journal*, 42:1977–1997, September 1963.

[43] ETSI. *High Performance Radio Local Area Network (HIPERLAN) - Draft Standard.* ETSI, March 1996.

[44] ETSI. *TR 101 683, HIPERLAN Type 2: System Overview.* ETSI, February 2000.

[45] ETSI. *TS 101 475, BRAN, HIPERLAN Type 2: Physical (PHY) Layer.* ETSI, March 2000.

[46] Andras Farago, Andrew D. Myers, Violet R. Syrotiuk, and Gergely V. Zaruba. Meta-MAC Protocols: Automatic Combination of MAC Protocols to Optimize Performance for Unknown Conditions. *IEEE Journal on Selected Areas in Communications*, 18(9):1670–1681, September 2000.

[47] William Feller. *An Introduction to Probability Theory and Its Applications - Volume I.* John Wiley, New York, third edition, 1968.

[48] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, 1997.

[49] International Organization for Standardization. *ISO Standard 8807 - Information processing systems - Open Systems Interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour.* ISO - Internation Organization for Standardization, February 1989.

[50] International Organization for Standardization. *ISO Standard 11898 - Road Vehicle - Interchange of Digital Information - Controller Area Network (CAN) for High-Speed Communication.* ISO - Internation Organization for Standardization, 1993.

[51] B. D. Fritchman. A binary channel characterisation using partitioned markov chains. *IEEE Transactions on Information Theory*, 13(2):221–227, April 1967.

[52] Projektkonsortium Funbus. Das verbundprojekt drahtlose feldbusse im produktionsumfeld (funbus) – abschlußbericht. INTERBUS Club Deutschland e.V., Postf. 1108, 32817 Blomberg, Bestell-Nr: TNR 5121324, October 2000. http://www.softing.de/d/NEWS/Funbusbericht.pdf.

[53] Aura Ganz, Anan Phonphoem, and Zvi Ganz. Robust Superpoll Protocol for IEEE 802.11 Wireless LANs. In *Proc. IEEE Military Communications Conference*, Boston, Massachusetts, October 1998.

[54] J. Garcia-Frias and P. M. Crespo. Hidden Markov Models for burst error characterization in indoor radio channels. *IEEE Transactions on Vehicular Technology*, 46:1006–1020, November 1997.

[55] German Institute of Standardization (DIN). *PROFIBUS Standard Part 1 and 2*, 1991.

[56] Jerry D. Gibson, editor. *The Communications Handbook.* CRC Press / IEEE Press, Boca Raton, Florida, 1996.

[57] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Systems Technical Journal*, 39:1253–1265, September 1960.

[58] Alois M. J. Goiser. *Handbuch der Spread-Spectrum Technik.* Springer Verlag, Wien, New York, 1998.

[59] James Gross, Michael Jaeger, and Andreas Willig. Measurements of a Wireless Link in different RF-isolated Environments. TKN Technical Report Series TKN-01-005, Telecommunication Networks Group, Technical University Berlin, June 2001. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[60] Ajay Chandra V. Gummalla and John O. Limb. Wireless medium access control protocols. *IEEE Communications Surveys and Tutorials*, 3(2), 2000. http://www.comsoc.org/pubs/surveys.

[61] Jaap C. Haartsen. The Bluetooth Radio System. *IEEE Personal Communications*, 7(1):28–36, February 2000.

[62] Fred Halsall. *Data Communications, Computer Networks and Open Systems*. Addison-Wesley, Reading, Massachusetts, 1996.

[63] M. Hata. Empirical formula for propagation loss in land mobile radio services. *IEEE Transactions on Vehicular Technology*, 29(3):317–325, August 1980.

[64] Boudewijn R. Haverkort. *Performance of Computer Communication Systems – A Model Based Approach*. John Wiley and Sons, Chichester / New York, 1998.

[65] Olivier Hersent, David Gurle, and Jean-Pierre Petit. *IP Telephony – Packet-based multimedia communications systems*. Addison-Wesley, Harlow / England, London, 2000.

[66] Hirschmann Rheinmetall Elektronik, Neckartenzlingen. *IZD Profi 01 – Description and Operating Instructions Infrared Transmission System*, March 1999.

[67] A. Hoffmann, R. J. Haines, and A. H. Aghvami. Performance analysis of a token based mac protocol with asymmetric polling strategy ('topo') for indoor radio local area networks under channel outage conditions. In *Proc. International Conference on Communications (ICC)*, pages 1306–1311, New Orleans, Louisiana, 1994. IEEE.

[68] Frank J. Furrer (Hrsg.). *BITBUS - Grundlagen und Praxis*. Hüthig Buch Verlag, Heidelberg, 1994.

[69] IEC - International Electrotechnical Commission. *IEC-1158-1, FieldBus Specification, Part 1, FieldBus Standard for Use in Industrial Control: Functional Requirements*.

[70] IEEE - Institute of Electrical and Electronics Engineers, IEEE Standards Department, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA. *IEEE Standard 1118 - IEEE Standard Microcontroller System Serial Control Bus*, August 1991.

[71] IEEE/ISO. *Information processing systems - Local Area Networks - part 4: Token-passing bus access method and physical layer specifications*. International Organization for Standardization, August 1990.

[72] International Standards Organization. *ISO 9506 – Industrial Automation Systems Integration and Communications - Manufacturing Message Specification, Part 1: Service Definition, Part 2: Protocol Specification*.

[73] Intersil. *HFA3860B Data Sheet, File Number 4594.1*, 1999.

[74] Ivan Izikowitz and Michael Solvie. Industrial needs for time-critical wireless communication & wireless data transmission and application layer support for time critical communication. In *Proc. Euro-Arch'93*, München, 1993. Springer Verlag, Berlin.

[75] Raj Jain. *The Art of Computer Systems Performance Analysis – Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Wiley Professional Computing. John Wiley and Sons, New York, Chichester, 1991.

[76] Raj Jain. *FDDI Handbook: High-Speed Networking Using Fiber and Other Media.* Addison-Wesley, Reading, Massachusetts, 1994.

[77] W. C. Jakes. *Microwave Mobile Communications.* Wiley, New York, 1974.

[78] W. C. Jakes, editor. *Microwave Mobile Communications.* IEEE Press, New Jersey, 1993.

[79] Michel C. Jeruchim, Philip Balaban, and K. Sam Shanmugan. *Simulation of Communication Systems – Modeling, Methodology and Techniques.* Information Technology: Transmission, Processing and Storage. Kluwer Academic/Plenum Publishers, New York, Boston, second edition, 2000.

[80] Vincent C. Jones. *MAP / TOP Networking – Achieving Computer Integrated Manufacturing.* McGraw-Hill, New York, 1988.

[81] L. Rauchhaupt Jörg Hähniche. Opportunities and problems of wireless fieldbus extensions. In *Proc. FeT'99: Feldbustechnik – Fieldbus Technology*, Magdeburg, 1999. Springer Verlag, Wien / New York.

[82] Hong ju Moon, Hong Seong Park, Sang Chul Ahn, and Wook Hyun Kwon. Performance Degradation of the IEEE 802.4 Token Bus Network in a Noisy Environment. *Computer Communications*, 21:547–557, 1998.

[83] Andreas Kanbach and Andreas Körber. *ISDN. Die Technik. Schnittstellen, Protokolle, Dienste, Endsysteme.* Hüthig Buch Verlag, Heidelberg, third edition, 1999.

[84] Michael Kasper. Profibus goes wireless – drahtlose Übertragung mit infrarotstrahlen. *Industrie Service*, page 30, July-August 1999.

[85] B. Kedem. *Binary Time Series.* Springer, New York, Basel, 1980.

[86] Kalevi Kilkki. *Differentiated Services for the Internet.* Macmillan Technical Publishing, Indianapolis, 1999.

[87] Young Yong Kim and San qi Li. Modeling multipath fading channel dynamics for packet data performance analysis. In *Proc. IEEE INFOCOM 98.* IEEE, 1998.

[88] Young Yong Kim and San qi Li. Capturing Important Statistics of a Fading/Shadowing Channel for Network Performance Analysis. *IEEE Journal on Selected Areas in Communications*, 17(5):888–901, May 1999.

[89] Ulrich Klehmet, Markus Ettl, and Peter Götz. Leistungsbewertung der feldbus-protokolle profibus und fip. *atp - automatisierungstechnische praxis*, 35(6):355ff, 1993.

[90] Leonard Kleinrock. *Queueing Systems – Volume 2: Computer Applications*, volume 2. John Wiley and Sons, New York, 1976.

[91] Almudena Konrad, Ben Y. Zhao, Anthony D. Joseph, and Reiner Ludwig. A Markov-Based Channel Model Algorithm for Wireless Networks. In *Proc. of Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2001)*, Rome, July 2001.

[92] Andreas Köpsel. A comparison between point and distributed coordination function of an ieee802.11 wlan. Diploma thesis, Telecommunication Networks Group (TKN), Technical University Berlin, Berlin, July 2000. in German.

[93] Osama Kubbar and Hussein T. Mouftah. Multiple access control protocols for wireless atm: Problems definition and design objectives. *IEEE Communications Magazine*, 35(11):93–99, November 1997.

[94] J. F. Kurose, M. Schwartz, and Y. Yemini. Multiple-access protocols and time-constrained communication. *ACM Computing Surveys*, 16:43–70, March 1984.

[95] A. Kutlu, H. Ekiz, M. D. Baba, and E. T. Powner. Implementation of "comb" based wireless access method for control area network. In *Proc. 11th Intl. Symp. on Computer and Information Science*, pages 565–573, Antalaya, Turkey, November 1996.

[96] A. Kutlu, H. Ekiz, and E. T. Powner. Performance analysis of MAC protocols for wireless control area network. In *Proc. Intl. Symp. on Parallel Architectures, Algorithms and Networks*, pages 494–499, Beijing, China, June 1996.

[97] Shu Lin and Daniel J. Costello. *Error Control Coding – Fundamentals and Applications.* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[98] C.L. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[99] Hang Liu, Hairuo Ma, Magda El Zarki, and Sanjay Gupta. Error control schemes for networks: An overview. *MONET – Mobile Networks and Applications*, 2(2):167–182, 1997.

[100] Jane W. S. Liu. *Real-Time Systems.* Prentice-Hall, Upper Saddle River, NJ, 2000.

[101] Madhav V. Marathe and Robin A. Smith. Performance of a map network adapter. *IEEE Network*, 2(3):82ff, 1988.

[102] R. H. McCullough. The binary regenerative channel. *Bell Systems Technical Journal*, 47:1713–1735, October 1968.

[103] Mesquite Software, Inc., T. Braker Lane, Austin, Texas. *CSIM18 Simulation Engine – Users Guide*, 1997.

[104] John J. Metzner. Message scheduling for efficient data communication under varying channel conditions. *IEEE Transactions on Communications*, 32(1):48–55, January 1984.

[105] Jouni Mikkonen, James Aldis, Geert Awater, Andrew Lunn, and David Hutchison. The magic wand – functional overview. *IEEE Journal on Selected Areas in Communications*, 16(6):953–972, August 1998.

[106] P. Heinz Müller, editor. *Wahrscheinlichkeitsrechnung und Mathematische Statistik.* Akademie Verlag, Berlin, third edition, 1980.

[107] M. Molinari and M. Zekar. *Drahtlose lokale Netze.* DATACOM-Verlag, Bergheim, 1994.

[108] Philip Morel. Mobility in map networks using the dect wireless protocols. In *Proc. 1995 IEEE Workshop on Factory Communication Systems, WFCS'95*, Leysin, Switzerland, 1995.

[109] Philip Morel and Alain Croisier. A wireless gateway for fieldbus. In *Proc. Sixth International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 95)*, 1995.

[110] Philip Morel and Jean-Dominique Decotignie. Integration of wireless mobile nodes in map/mms. In *Proc. 13th IFAC Workshop on Distributed Computer Control Systems DCCS95*, 1995.

[111] Motorola Inc. *MPC 860 PowerQUICC Users's Manual*, 1998.

[112] General Motors. *Manufacturing Automation Protocol V3.0.* General Motors, 1988.

[113] P. Narasimhan, S. K. Biswas, C. A. Johnston, R. J. Syracusa, and H. Kim. Design and performance of radio access protocols in WATMNet, a prototype wireless atm network. In *Proc. ICUPC 97, 6th WINLAB Workshop*, 1997.

[114] Kevin J. Negus, Adrian P. Stephens, and Jim Lansford. HomeRF: Wireless Networking for the Connected Home. *IEEE Personal Communications*, 7(1):20–27, February 2000.

[115] Randolph Nelson. *Probability, Stochastic Processes, and Queueing Theory – The Mathematics of Computer Performance Modeling.* Springer Verlag, New York, 1995.

[116] P. Neumann, C. Diedrich, and J. Hähniche. Der nationale feldbusstandard profibus - profile, implementationen und tests. *ZwF - Zeitschrift für wirtschaftliche Fertigung*, 87(7):365ff, July 1992.

[117] Joseph Kee-Yin Ng. MPEG transmission schemes for a timed token medium access control network. *ACM Computer Communication Review*, 29(1):66 – 80, January 1999.

[118] Giao T. Nguyen, , Randy H. Katz, Brian Noble, , and Mahadev Satyanarayanan. A trace-based approach for modeling wireless channel behavior. In *Proceedings of the Winter Simulation Conference*, Coronado, CA, December 1996.

[119] The Editors of IEEE 802. *IEEE 802.2, ISO/IEC 8802-2: Local Area Networks: Logical Link Control*, 1989.

[120] The Editors of IEEE 802.11. *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, November 1997.

[121] The Editors of IEEE 802.11. *IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher speed Physical Layer (PHY) extension in the 2.4 Ghz band*, 1999.

[122] The Editors of IEEE 802.11. *IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band*, 1999.

[123] Bob O'Hara and Al Petrick. *IEEE 802.11 Handbook – A Designer's Companion.* IEEE Press, New York, 1999.

[124] K. Pahlavan and A.H. Levesque. *Wireless Information Networks.* J. Wiley and Sons, 1995.

[125] Kaveh Pahlavan, Ali Zahedi, and Prashant Krishnamurthy. Wideband local access: Wireless lan and wireless atm. *IEEE Communications Magazine*, 35(11):34–40, November 1997.

[126] J. D. Parsons. *The Mobile Radio Propagation Channel.* Pentech Press, London, 1992.

[127] Juan R. Pimentel. *Communication Networks for Manufacturing.* Prentice-Hall International, 1990.

[128] Patrick Pleineveaux and Jean-Dominique Decotignie. Time critical communication networks: Field buses. *IEEE Network*, 2(3):55ff, 1988.

[129] D. K. Pradhan, editor. *Fault-tolerant Computer System Design.* Prentice Hall, Upper Saddle River, NJ, 1996.

[130] Prashant Pradhan and Tzi-Cker Chiueh. Real-Time Performance Guarantees over Wired/Wireless LANs. In *Proc. IEEE Real-Time Application and Technology Symposium.* IEEE, June 1998.

[131] Anand R. Prasad. Performance comparison of voice over ieee 802.11 schemes. In *Proc. IEEE Vehicular Technology Conference (VTC) '99.* IEEE, 1998.

[132] John G. Proakis. Channel equalization. In Jerry D. Gibson, editor, *The Communications Handbook*, pages 339–363. CRC Press / IEEE Press, Boca Raton, Florida, 1996.

[133] PROFIBUS Nutzerorganisation e.V., PROFIBUS Nutzerorganisation e.V., Haid-und-Neu-Str. 7, Karlsruhe, Germany. *PROFIBUS – Entwurf Technische Richtlinie – Implementierungshinweise zur DIN 19245 Teil 2*, December 1993.

[134] PROFIBUS Nutzerorganisation e.V., PROFIBUS Nutzerorganisation e.V., Haid-und-Neu-Str. 7, Karlsruhe, Germany. *Implementation Guide to DIN 19245 Part 1*, August 1994.

[135] PROFIBUS Nutzerorganisation e.V., PROFIBUS Nutzerorganisation e.V., Haid-und-Neu-Str. 7, Karlsruhe, Germany. *PROFIBUS – Entwurf Technische Richtlinie – Implementierungshinweise zur DIN 19245 Teil 1*, August 1994.

[136] PROFIBUS Nutzerorganisation e.V., PROFIBUS Nutzerorganisation e.V., Haid-und-Neu-Str. 7, Karlsruhe, Germany. *PROFIBUS – Technical Description*, September 1999.

[137] Matthias Pätzold. *Mobilfunkkanäle – Modellierung, Analyse und Simulation.* Vieweg, Braunschweig, 1999.

[138] Matthias Pätzold and Frank Laue. Level crossing rate and average duration of fades of deterministic simulation models for rice fading channels. *IEEE Transactions on Vehicular Technology*, 48(4):1121–1129, July 1999.

[139] San qi Li and Chia-Lin Hwang. Queue response to input correlation functions: Continuous spectral analysis. *IEEE/ACM Transactions on Networking*, 1:678–692, December 1993.

[140] San qi Li and Chia-Lin Hwang. On the convergence of traffic measurement and queueing analysis: A statistical-match queueing (smaq) tool. *IEEE/ACM Transactions on Networking*, 5:95–110, February 1997.

[141] R-Fieldbus Consortium. *R-FIELDBUS – High Performance Wireless Fieldbus in Industrial Related Multi-Media Environment*, 2000. Presentation Slides from www.rfieldbus.de.

[142] Jan M. Rabaey, M. Josie Ammer, Julio L. da Silva, Danny Patel, and Shad Roundy. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. *IEEE Computer*, 33(7), July 2000.

[143] Markus Radimirsch and Jamshid Khun-Jush. Application of hiperlan type 2 systems in private environments. In *Proc. International Conference on Telecommunications*, Acapulco, Mexico, 2000.

[144] M. Rahnema. Overview of the gsm system and protocol architecture. *IEEE Communications Magazine*, 31(4):92–100, April 1993.

[145] Theodore S. Rappaport, Rias Muhamed, and Varun Kapoor. Propagation models. In Jerry D. Gibson, editor, *The Communications Handbook*, pages 1182–1196. CRC Press / IEEE Press, Boca Raton, Florida, 1996.

[146] U. Rembold, B. O. Nnaji, and A. Storr. *Computer Integrated Manufacturing and Engineering.* Addison-Wesley, 1993.

[147] S. O. Rice. Mathematical analysis of random noise. *Bell Systems Technical Journal*, 23:282–332, July 1944.

[148] S. O. Rice. Mathematical analysis of random noise. *Bell Systems Technical Journal*, 24:46–156, January 1945.

[149] Izhak Rubin and L. F. M. de Moraes. Message Delay Analysis for Polling and Token Multiple-Access Schemes for Local Communication Networks. *IEEE Journal on Selected Areas in Communications*, 1(5):935–947, 1983.

[150] Asuncion Santamaria and Francisco J. Lopez-Hernandez, editors. *Wireless LAN – Standards and Applications.* Mobile Communication Series. Artech House, Boston, London, 2001.

[151] Mischa Schwartz. *Telecommunication Networks - Protocols, Modeling and Analysis.* Addison-Wesley, Reading, Massachusetts, 1988.

[152] Oran Sharon and Eitan Altman. An efficient polling mac for wireless lans. *IEEE/ACM Transactions on Networking*, 9(4):439–451, August 2001.

[153] Kang G. Shin and Parameswaran Ramanathan. Real-Time Computing: A New Discipline of Computer Science and Engineering. *Proceedings of the IEEE*, 82(1):6–24, January 1994.

[154] D. P. Siewiorek and R. S. Swarz. *Reliable Computer Systems Design and Evaluation.* Digital Press, Burlington, MA, 2nd edition, 1992.

[155] Bernard Sklar. *Digital Communications – Fundamentals and Applications.* Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[156] Joao L. Sobrinho and A. S. Krishnakumar. Real-time traffic over the ieee 802.11 medium access control layer. *Bell Labs Technical Journal*, 1(2):172–187, 1996.

[157] SRB Innovative Industrie Elektronik Gmbh. *Silver Data Stream Transfer-Modul – User Manual for SDSTM-F24-V1.0.1*, 1999.

[158] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, New Jersey, 1994.

[159] Takahiro Suzuki and Shuji Tasaka. Performance evaluation of video transmission with the pcf of the ieee 802.11 standard mac protocol. *IEICE Transactions on Communications*, E83-B(9):2068–2076, September 2000.

[160] F. Swarts and H.C. Ferreira. Markov characterization of digital fading mobile vhf channels. *IEEE Transactions on Vehicular Technology*, 43(4):977–985, November 1994.

[161] Hideaki Takagi. *Analysis of Polling Systems.* MIT Press, Cambridge, Massachusetts, 1986.

[162] Hideaki Takagi. Analysis and applications of a multi-queue cyclic service system with feedback. *IEEE Transactions on Communications*, 35(2):248–250, February 1987.

[163] Hideaki Takagi. Queueing analysis of polling models: an update. In Hideaki Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*, pages 267–318. Elsevier, Amsterdam, 1990.

[164] T. Takine, H. Takagi, and T. Hasegawa. Sojourn times in vacation and polling systems with bernouilli feedback. *Journal of Applied Probability*, 28:422–432, June 1991.

[165] Andrew S. Tanenbaum. *Computer-Netzwerke.* Wolframs Fachverlag, Attenkirchen, second edition, 1992.

[166] Andrew S. Tanenbaum. *Computernetzwerke.* Prentice-Hall, Muenchen, third edition, 1997.

[167] Fouad A. Tobagi and Leonard Kleinrock. Packet switching in radio channels: Part ii the hidden terminal problem in csma and busy-tone solutions. *IEEE Transactions on Communications*, 23(12):1417–1433, 1975.

[168] Eduardo Tovar. *Supporting Real-Time Communications with Standard Factory-Floor Networks.* PhD dissertation, Dept. of Electrical Engineering, Univ. of Porto, Portugal, 1999.

[169] Eduardo Tovar and Francisco Vasques. Real-Time Fieldbus Communications Using Profibus Networks. *IEEE Transactions on Industrial Electronics*, 46(6):1241–1251, December 1999.

[170] Don Towsley and J. K. Wolf. On adaptive tree polling algorithms. *IEEE Transactions on Communications*, 32(12):1294–1298, 1984.

[171] Tundra Corporation. *Reference Manual Tundra PCI Interconnect*, 1998.

[172] W. Turin and R. van Nobelen. Hidden markov modeling of fading channels. In *Proc. IEEE Vehicular Technology Conference*, pages 1234–1238, May 1998.

[173] William Turin. *Digital Transmission Systems – Performance Analysis and Modeling.* McGraw-Hill Telecommunications. McGraw-Hill, New York, 1998.

[174] William Turin and Robert van Nobelen. Hidden Markov Modeling of Flat Fading Channels. *IEEE Journal on Selected Areas in Communications*, 16(9):1809–1817, December 1998.

[175] Kenneth J. Turner, editor. *Using Formal Description Techniques - An Introduction to Estelle, LOTOS and SDL.* Prentice Hall, Chichester, New York, 1993.

[176] Union Technique de l'Electricité. *General Purpose Field Communication System, EN 50170, Volume 1: P-NET*, 1996.

[177] Union Technique de l'Electricité. *General Purpose Field Communication System, EN 50170, Volume 2: PROFIBUS*, 1996.

[178] Union Technique de l'Electricité. *General Purpose Field Communication System, EN 50170, Volume 3: WorldFIP*, 1996.

[179] Richard van Nee and Ramjee Prasad. *OFDM for Wireless Multimedia Communications.* Artech House Publisher, 2000.

[180] Malathi Veeraraghavan, Nabeel Cocker, and Tim Moors. Support of voice services in ieee 802.11 wireless lans. In *Proc. INFOCOM 2001*, Anchorage, Alaska, April 2001. IEEE.

[181] Matthijs A. Visser and Magda El Zarki. Voice and Data Transmission over an 802.11 Wireless network. In *Proc. IEEE Personal, Indoor and Mobile Radio Conference (PIMRC) 95*, pages 648–652, Toronto, Canada, September 1995.

[182] Bernhard Walke. *Mobilfunknetze und ihre Protokolle, Band 1.* Informationstechnik. B.G. Teubner, Stuttgart, 1998.

[183] Bernhard Walke. *Mobilfunknetze und ihre Protokolle, Band 2.* Informationstechnik. B.G. Teubner, Stuttgart, 1998.

[184] H.S. Wang and N. Moayeri. Finite State Markov Channel - A Useful Model for Radio Communication Channels. *IEEE Transactions on Vehicular Technology*, 44(1):163–171, February 1995.

[185] Jost Weinmiller, Morten Schl"ager, Andreas Festag, and Adam Wolisz. Performance study of access control in wireless LANs – IEEE 802.11 DFWMAC and ETSI RES 10 Hiperlan. *MONET - Mobile Networks and Applications*, 2(1):55–67, 1997.

[186] Jost Weinmiller, Hagen Woesner, Jean-Pierre Ebert, and Adam Wolisz. Analyzing and improving the 802.11-mac protocol for wireless lans. In *Proc. MASCOT 95*, San Jose, California, February 1995.

[187] Stuart Williams. IrDA: Past, Present and Future. *IEEE Personal Communications*, 7(1), February 2000.

[188] Andreas Willig. Analysis and Tuning of the PROFIBUS Token Passing Protocol for Use Over Error-Prone Links. TKN Technical Report Series TKN-99-001, Telecommunication Networks Group, Technical University Berlin, March 1999.

[189] Andreas Willig. Analysis of the PROFIBUS Token Passing Protocol over Error Prone Links. In *Proc. 25th Annual Conference of the IEEE Industrial Electronics Society (IECON'99)*, pages 1246 – 1252. IEEE, November 1999.

[190] Andreas Willig. Markov Modeling of PROFIBUS Ring Membership over Error Prone Links. TKN Technical Report Series TKN-99-004, Telecommunication Networks Group, Technical University Berlin, May 1999. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[191] Andreas Willig. Architectural Considerations for a wireless integrated PROFIBUS. TKN Technical Report Series TKN-01-006, Telecommunication Networks Group, Technical University Berlin, October 2001. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[192] Andreas Willig, Martin Kubisch, Christian Hoene, and Adam Wolisz. Measurements of a Wireless Link in an Industrial Environment using an IEEE 802.11-Compliant Physical Layer. *IEEE Transactions on Industrial Electronics*, 2001. accepted for publication.

[193] Andreas Willig, Martin Kubisch, and Adam Wolisz. Bit Error Rate Measurements – Second Campaign, factorial measurement. TKN Technical Report Series TKN-00-011, Telecommunication Networks Group, Technical University Berlin, November 2000. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[194] Andreas Willig, Martin Kubisch, and Adam Wolisz. Bit Error Rate Measurements – Second Campaign, longterm1 measurement. TKN Technical Report Series TKN-00-009, Telecommunication Networks Group, Technical University Berlin, November 2000. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[195] Andreas Willig, Martin Kubisch, and Adam Wolisz. Bit Error Rate Measurements – Second Campaign, longterm2 measurement. TKN Technical Report Series TKN-00-010, Telecommunication Networks Group, Technical University Berlin, November 2000. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[196] Andreas Willig, Martin Kubisch, and Adam Wolisz. Results of Bit Error Rate Measurements with an IEEE 802.11 compliant PHY. TKN Technical Report Series TKN-00-008, Telecommunication Networks Group, Technical University Berlin, November 2000. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[197] Andreas Willig, Martin Kubisch, and Adam Wolisz. Measurements and Stochastic Modeling of a Wireless Link in an Industrial Environment. TKN Technical Report Series TKN-01-001, Telecommunication Networks Group, Technical University Berlin, March 2001. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[198] Andreas Willig and Adam Wolisz. Ring Stability of the PROFIBUS Token Passing Protocol over Error Prone Links. *IEEE Transactions on Industrial Electronics*, 48(5):1025–1033, October 2001.

[199] Zhensheng Zhang and Anthony S. Acampora. Performance of a modified polling strategy for broadband wireless lans in a harsh fading environment. *Telecommunication Systems*, 1:279–294, 1993.

[200] Michele Zorzi and Ramesh R. Rao. Performance of arq go-back-n protocol in markov channels with unreliable feedback. *Wireless Networks*, 2:183–193, 1997.

[201] Michele Zorzi and Ramesh R. Rao. Perspectives on the impact of error statistics on protocols for wireless networks. *IEEE Personal Communications*, 6(5), October 1999.