Technical University Berlin

Telecommunication Networks Group

Analysis and Tuning of the PROFIBUS Token Passing Protocol for Use over Error Prone Links

Andreas Willig

willig@ft.ee.tu-berlin.de

Berlin, March 1999

TKN Technical Report TKN-99-001

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

In this paper we investigate the properties of the PROFIBUS MAC protocol when operated over error prone links, with error behaviour similar to e.g. wireless links. First we show that the frame error detection method of PROFIBUS has inferior performance as compared to standard techniques like CRC's. Then we show with a simulation approach, that the MAC protocol is very sensitive to loss or errors in the special control frames incorporated in the protocol. Under bad link conditions the logical ring can completely break down. We propose some slight changes to the protocol and to the frame formats. Two of them are investigated by simulation and an significant increase in delay performance and ring stability can be observed. However, our results indicate that the PROFIBUS protocol is not really a good choice for use over error prone links.

Contents

Chapter 1

Introduction

The PROFIBUS is a widely used and well standardized field bus [2], [3]. It is mainly used in industrial environments for applications like interconnection of industrial controllers (PLC, CNC, RC), coupling of sensors and actors to a controller and so forth (distributed control applications). It is designed to meet some hard real time requirements for industrial communication purposes. As transmission medium shielded twisted pair or fibre optic cables can be used. However, during the last years there was rapidly growing interest in wireless technology. Making the different benefits of wireless technology available for PROFIBUS installations has some advantages:

- Stations can be attached and (re-)moved easily without changing a cabling system.
- Stations can be mobile, switching from PROFIBUS LAN to PROFIBUS LAN or moving within a single PROFIBUS LAN.
- When using a wireless link there is no cable which can be damaged or destroyed, thus there are less opportunities for breakdowns of a production plant.

Our current research effort aims at the definition of a wireless extension to wired PROFIBUS with the final goal of joint operation of wired and wireless parts within a single LAN. One important question is, whether it is possible or desirable to use the PROFIBUS MAC protocol (which uses token passing similar IEEE 802.4) on top of a wireless medium or better to use something else. So it is natural to ask, how the token passing protocol behaves on links, which show error behaviour similar to wireless ones. Unfortunately, the characteristics of wireless technology are different from most of the cable types used in wired LANs. First, they tend to exhibit a nonstationary and bursty error behaviour, with very high bit error rates during an error burst. Furthermore, due to the path loss it may happen that within a single LAN

not all stations hear each other, i.e. we have only partial reachability. For these reasons one cannot expect the same behaviour of the PROFIBUS MAC protocol on a wireless link as on a wired link.

When considering transmission over an error prone and lossy medium, performance degradations mainly stem from two sources: one source is the loss of data frames, making retransmissions necessary, the other source is the vulnerability of the additional protocol mechanisms and frame formats used. The main question is, how the loss of special control frames affects the performance as compared to the case where there is no loss of special frames or where the protocol does not use special frames at all.

In this paper we analyze the deficiencies of the PROFIBUS MAC protocol when operated over error prone links[1], however, in a topology with full reachability. We show that the special control packets used for token passing and ring maintenance make the protocol vulnerable for serious performance degradations if the link exhibits a high error rate. In addition, we show that the protocol behaves differently under different error models. This is established by careful analysis of the standards document and by a simulation study. As performance measures we use the mean delay for user data, the mean station outage time (i.e. the mean duration needed to re-include a station in the ring after it gets lost) and the cumulated station outage times. After that we propose some small modifications of the protocol and frame formats, which make the protocol more robust against errors.

The paper is structured as follows: in section 2 we give an overview on the relevant properties of the PROFIBUS, then doing in sec. 3 the same for the modeling of the error behaviour of wireless channels. In section 4 we analyze the frame formats and the error detection capabilities of the PROFIBUS and show that they are significantly inferior to standard techniques like CRC checksums. In section 5 we present the results from a simulation study on different aspects of the PROFIBUS protocol performance when operated in an error prone environment. We will see that the original protocol completely breaks down when operated in a harsh environment. In section 6 we propose some changes to the PROFIBUS protocol, which makes the protocol much more stable under bad link conditions. Two of these enhancements are evaluated by simulation. Finally, in section 7 our conclusions are given.

[1] An analysis of the IEEE 802.4 Token Bus with error prone links can be found in [8].

Chapter 2

Overview on the PROFIBUS Token Passing Protocol

The PROFIBUS is standardized in [2], [3] and in the additional documents [13], [12]. It comes in different "flavors". One of them (PROFIBUS-FMS) is intended for use on the cell level in a factory, e.g. interconnecting different control devices on the same hierarchy level, having multiple active stations (see below). In this paper we focus solely on PROFIBUS-FMS.

The PROFIBUS uses two different protocol approaches on the MAC layer: a master/slave protocol for exchange of data frames[1] and a token passing protocol for managing the case of multiple masters. The token passing protocol uses a broadcast medium, e.g. a bus or a tree topology in case cables are used. A logical ring is formed by ascending station addresses. The address space is very small, a stations address is in the range of 0 to 126. Every station (denoted as TS: This Station) knows the address of its logical successor (NS: Next Station) and its logical predecessor (PS: Previous Station). This knowledge is obtained from the ring maintenance mechanisms described below. If TS receives a token frame, it checks whether it was sent by its PS. If so, the token is accepted, otherwise the token 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. In any case after accepting the token TS determines its token holding time and sends its own data[2]. After finishing TS tries

[1] In the PROFIBUS standard frames are also called "telegrams".

[2] The algorithm for computing the token holding time (THT) is very similar to the one used in FDDI: every station measures the time between two successive token arrivals and subtracts this from a constant global time, the *target token rotation time* (TTRT). If this difference (the *token holding time* (THT) is positive the station may send some data (for a duration of up to THT). If it is zero or negative the station is allowed to send one high priority frame and then passes the token.

to pass the token to NS. On this behalf a token frame is sent to NS. After that TS listens on the medium for some activity. This can be 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 a short time (called *slot time*) which is typically chosen very sharp, e.g. in the range of 100 μsec to 400 μsec. If this time passes without any bus activity the token frame is repeated. If there is again no activity 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 information gathered during ring maintenance, see below. If TS finds no other station, it sends a token frame to itself. A special case in passing the token is the following: TS is required to listen simultaneously while sending the token telegram ("hearback"). If TS encounters a difference the first time, it behaves as after a correct token telegram, i.e. it waits for some response. If there is no response it repeats the token telegram. If TS again encounters a difference, it discards the token and removes itself from the ring, behaving as newly switched on.

If TS is newly switched on it is required to first listen passively on the medium, until it has received two successive identical token cycles. During this time it is not allowed to send or answer to data frames. Every station address found in a token frame belonging to this two cycles is included into a locally maintained *list of active stations* (LAS). After that TS can enter the ring if it is included by its predecessor (see below). In addition TS is required to maintain its LAS by inspecting every received token frame. A special rule in this maintenance process is the following: if TS is already included in the logical ring and founds itself "skipped" by a token frame (i.e. the address of TS lies 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.

The algorithm described so far makes it easy for a station TS to leave the ring: it just stops accepting token frames, no special control frames are needed for this case. Its predecessor station PS will be able to detect the loss of the station and to find out who is the successor station of TS. Including a new station in a ring is more complicated. Every station maintains a gap list (GAPL), containing all station addresses between TS and NS. TS is required to scan periodically all stations in GAPL by explicitly sending a special request frame (Request-FDL-Status) to a single address and waiting for an answer, indicating the type of the station and its 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 TS

pings at most one station address in GAPL, depending on the presence of high priority data traffic. If a station in GAPL responds as "ready" TS will change its NS, shorten its GAPL and sending 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 TTRT. Adjusting this timer is a critical parameter for the delay necessary to (re-) include a station. If the period is short bandwidth is wasted, if it is long then ring inclusion delays get larger.

A special mechanism is used for very first ring initialization or token loss due to system crash of the current token owner: every station listens permanently on the bus. If there is no bus activity for some time (the corresponding timer is called *timeout timer*), the station "claims the token", i.e. it starts transmitting either data or a token frame. The timeout value is linearly dependent on the stations address[3].

In the PROFIBUS a distinction is made between *active stations* and *passive stations*. Active stations are capable of participating in the token passing process, thus they get the token from time to time and perform some data transmission. A passive station cannot handle the token. In all cases it acts only as a slave, i.e. it responds only to request telegrams.

[3]This timeout timer is one of the reasons for introduction of the hearback feature: it is necessary in order to resolve collisions, which may occur e.g. when two stations are newly switched on and their timeout timer expires at the same time.

Chapter 3

Channel Models for Wireless LANs

When studying protocol behaviour over wireless channels, some kind of channel model is needed. In this work we consider radio transmission, e.g. using the license free 2.4 GHz ISM band (Industrial, Scientific and Medical band). It is widely accepted, that the radio channel is a bad channel with non-stationary error characteristics. Bit error rates of $\approx 10^{-2\ldots-3}$ are reported (for measurements see [1], [5], [4]). The error process is constituted by phenomena like slow fading, fast fading, noise, delay spread, interference and a path loss which is quadratic or even worse in the distance between two stations, [11]. It is known that the error process often exhibits a bursty behaviour.

For modeling the error characteristics of a wireless channel on a high level (as compared to ray-tracing channel simulations) we use two different models throughout this paper. The first is the simple "independent" model, where bit errors occur independent from each other according to a predefined fixed bit error rate (BER). This model is very simple but it is not capable of capturing the bursty error characteristics of wireless LANs. The second model is the widely used "Gilbert/Elliot model" (simply denoted as Gilbert model) [16], [7], [6]: the channel state is modulated according to a two state continuous markov chain (general: n-state) with the states named *Good* and *Bad*, see figure 3.1. Every state is assigned a specific

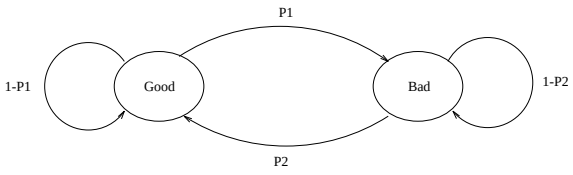

Figure 3.1: Gilbert/Elliot Channel Model

constant bit error rate (BER), E_g in the good state, E_b in the bad state ($E_g \ll E_b$). Within one state bit errors are assumed to be independent. The bit error rates in general depend on the frequency and coding scheme used and on environmental conditions (e.g. number of paths between source and destination). When the error model is aimed to be accurate then between every pair of stations a separate channel is needed. These channels are in general not synchronous, but some correlations may be present e.g. due to interference. However, in order to keep computational complexity low we use only a single channel for all stations.

We use a special incarnation of such a markov model, based on the methodology given in [16]. This methodology assumes BPSK coding over a rayleigh fading channel. The rationale for choosing these assumptions were the following:

- BPSK (Binary Phase Shift Keying) is a widely used coding scheme for wireless transmission ((D)BPSK is used e.g. in the 802.11 Wireless LAN standard).
- Rayleigh fading is appropriate for mobile stations or for scenarios with many paths between sender and receiver, which exhibits all approximately the same signal strength. We believe that the latter scenario is not uncommon in industrial environments due to many reflections on metal.
- The model can capture bursty error characteristics.
- It is computationally tractable.

This model allows one to estimate the parameters of a homogeneous discrete time markov chain model, where every state has its own, independent BER, and state transition occur after every channel symbol. These parameters are the bit error rates E_g, E_b and the state transitions probabilities p_{GG} (for the probability that the next state of the markov chain is the good state, given that the markov chain is currently in the good state) and p_{BB}. In this case the state sojourn time is geometrically distributed. However, we have not used a discrete time markov chain due to its high computational overhead during simulation. Instead we have defined a continuous time markov chain with the same two states good and bad. The state sojourn times are exponentially distributed and their mean state sojourn time is selected to be the mean of the corresponding geometric distributions in the discrete case multiplied with the duration of a single channel symbol. Let λ denote the parameter for the exponential distribution of the state sojourn time in the good state, μ the same for the bad state. If one looks at the system at a random point in time, it is straightforward to show, that the

	Mean BER = 0.001	Mean BER = 0.0001
Mean Bad Duration (sec)	0.0244	0.0144
Mean Good Duration (sec)	0.0617	0.0944
Bad BER	0.0036	0.00064
Good BER	0.000082	0.00002

Table 3.1: Parameters for Gilbert/Elliot-Model for channel errors

probability p_g that the system is found in good state (resp. bad state) is given by

$$\begin{aligned} p_g &= \frac{\lambda}{\lambda+\mu} \\ p_b &= \frac{\mu}{\lambda+\mu} \end{aligned}$$

The mean bit error rate E_m is then given by

$$E_m = p_g E_g + p_b E_b$$

The numerical parameters generated by the model of [16] are given in table 3.1. These are used throughout this paper.

Chapter 4

Analysis of the PROFIBUS Error Detection Method and Frame Formats

In this section we analyze the performance of the error detection method and some deficiencies of the frame formats chosen for PROFIBUS. The frame formats are shown in figure 4.1. For the analyis we make the following assumptions:

- bit errors occur independent from each other with a constant probability p.
- the data bits are random and independent of each other, with the values 0 and 1 equiprobable[1].
- bit errors change the value of a bit from 0 to 1 and vice versa, the bit error probability is independent from the bit value.

The error detection method of PROFIBUS works as follows: in the physical layer every byte is transmitted with 11 bits: one startbit, eight data bits, one parity bit and a stopbit. Errors in the start- and stopbit are always detected[2]. In addition, all frame types except the three bytes short token frame are equipped with a checksum byte (FCS), which covers source and destination address, function code and the data unit. However, for simplicity we assume that all bytes are covered by the checksum. The checksum is the sum modulo 256

[1] For "real data" this assumption is not true, as is shown in [15].

[2] Please note that start and stopbits are intended for bit synchronization, not for error detection. However, these bits need to be correct anyway.

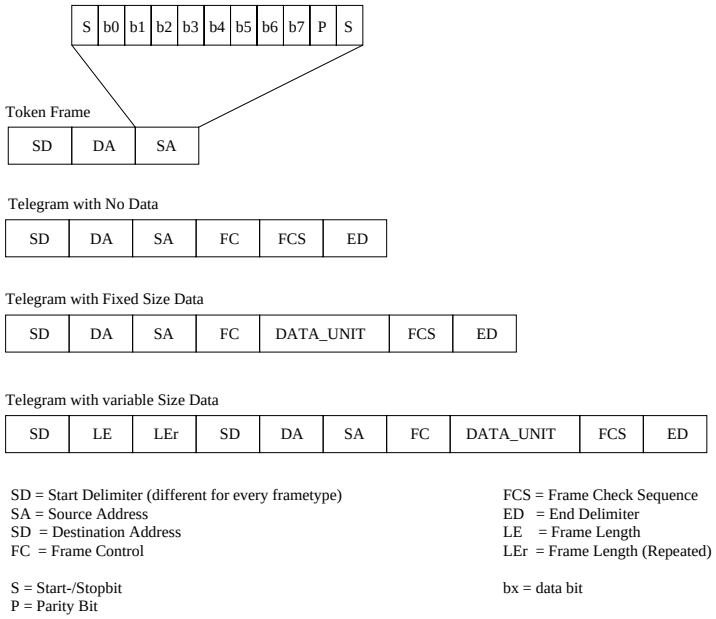

Figure 4.1: Frame Formats

of all the bytes (taken as unsigned). In the following we consider only the frame types with a checksum. Denote $v_o(i)$ the bit value of the original data on bit position i, accordingly $v_e(i)$ for the erroneous data (with respect to the 11 bit representation). Our question is the following: if we transmit a frame of length l bytes with $n := 11 \cdot l$ bits and there occur k ($k \leq n$) errors within the frame at different positions[3], whats the probability that the error is not detected by the checksum computation algorithm? We start with some observations.

Remark. *If $k > \frac{9}{11}n$ an error is always detected*

Proof. At least one start- or stopbit is hit □

Remark. *If $k \leq \frac{9}{11}n$ then the probability, that no start- or stopbit is hit is given by*

$$\Pr[\{e_1, \ldots, e_k\} \cap \{s_1, \ldots, s_{2l}\} = \emptyset] = \frac{\binom{9l}{k}}{\frac{(11l)!}{(11l-k)!}}$$

where $e_1, \ldots, e_k$ are the pairwise different bit error positions and $s_1, \ldots, s_{2l}$ are the positions of the start- and stopbits.

Proof. We use an ordinary generating function approach [10]. We start from the problem of placing k indistinguishable balls within n distinguishable urns, such that n_1 urns carry no ball

[3] We exclude the case where two bit errors on a single bit compensate each other.

(corresponding to the start- and stopbits) and n_2 urns carry at most one ball ($n = n_1 + n_2$). Then the corresponding power series for the first n_1 urns is simply given by

$$1$$

and the power series for the remaining urns is given by

$$1 + x$$

Multiplying all power series for all urns yields the generating function $G(x)$:

$$G(x) = 1^{n_1} \cdot (1+x)^{n_2} = (1+x)^{n_2}$$

The number of possibilities for placing k balls within n urns is then given by the coefficient of $G(x)$ for the x^k term, in this case $\binom{n2}{k}$. The overall number of possibilities for placing k balls in n different urns is given by $\frac{n!}{(n-k)!}$. Thus the overall probability that the k balls are only in the n_2 urns is given by the ratio

$$\frac{\binom{n_2}{k}}{\frac{n!}{(n-k)!}}$$

The theorem follows when choosing $n = 11l$, $n_1 = 2l$, $n_2 = 9l$ □

Theorem. *All odd numbers of bit errors are detected.*

Proof. If the number of bit errors is odd there is at least one byte within the frame with an odd number of bit errors within it. Thus the parity check will detect this or the start-/stopbit will be wrong. □

Theorem. *If exactly two bits are wrong within a frame this is always detected*

Proof. If one start- or stopbit is hit, then the error is detected. Otherwise, if the wrong bits are located in different bytes the respective parity bits will be wrong. If the are located in the same byte the checksum will be wrong. □

Theorem. *If exactly four bits are wrong then there exists a data pattern and an error pattern such that no error is detected, given that $l \geq 2$. The probability, that exactly four errors occur and that these errors are not detected is given by:*

$$\Pr[k = 4, \textit{The errors are not detected}] \quad = \quad b(4; 11l, p) \cdot \frac{1}{16} \cdot \frac{72l(l-1)}{(11l)(11l-1)(11l-2)(11l-3)}$$

where

$$b(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$$

is the distribution function of the binomial distribution.

Proof. First we show the existence: consider the case $l = 2$, the original sequence being

$$s11110000ps \quad s00001111ps$$

(start- and stopbits denoted by s and the parity bit denoted by p) and the disturbed sequence being

$$s11101000ps \quad s00010111ps$$

For the probability statement first recall that the number k of errors within n bits is a binomial random variable with the above given distribution function. It is easy to see that four bit errors remain undetected, if and only if the following conditions hold simultaneously:

- The start- and stopbits must not be hit.
- Within a single byte no error or exactly two errors must occur (two errors in the user data or one bit in the user data and a wrong parity bit), since all odd number of errors (1, 3) are detected by the parity bit and the case of four errors within a single byte is detected by the checksum.
- Thus we need two erroneous bytes. Furthermore it is necessary that e_1 and e_3 share the same relative bit position within their respective bytes, the same must hold for e_2 and e_4 (eventually after renumbering).
- Last it is required that the values $v_o(e_1)$ and $v_o(e_3)$ are each others complement, the same must hold for e_2 and e_4. By our assumptions on the data distribution this probability is $(1/4)^2$

Next we determine the number of "acceptable" positions (i.e. such that they are not detected) for the four errors. We can place e_1 freely on all positions except start- and stopbits, leaving $9l$ possible positions. For e_2 we have only eight possible positions (eight other bits within the same byte including parity bit). For e_3 we require it to be in another byte on the same relative position as e_1, thus we have $l - 1$ possibilities. The position of e_4 is then uniquely determined. As a result the total number of possible positions is $9l \cdot 8 \cdot (l - 1)$. The total number of possible positions of 4 errors in n bits is given by $\frac{n!}{(n-4)!}$. The ratio of both numbers and thus the probability that four errors occur in correct positions is given by

$$\Pr[\text{Errors have right positions in } l \text{ bytes} | k = 4] = \frac{72l(l-1)}{(11l)(11l-1)(11l-2)(11l-3)}$$

Now we have the probability, that exactly four errors occur and these are not detected:

$$\begin{aligned}\Pr[k=4, \text{The errors are not detected}] &= \Pr[k=4] \\ &\quad \cdot \Pr[\text{Errors have right positions in } l \text{ bytes} | k=4] \\ &\quad \cdot \Pr[\text{The errors have the right values}] \\ &= b(4; 11l, p) \cdot \frac{1}{16} \cdot \frac{72l(l-1)}{(11l)(11l-1)(11l-2)(11l-3)}\end{aligned}$$

□

Theorem. *If exactly six bits are wrong then there exists a data pattern and an error pattern such that no error is detected, given that $l \geq 3$.*

Proof. Consider the case $l = 3$, the original sequence being

$$s11110000ps \quad s00001111ps \quad s00001111ps$$

and the disturbed sequence being

$$s11111010ps \quad s00001010ps \quad s00001010ps$$

In this case the probability Pr[6 errors occur in n bits] is some orders of magnitude lower than the probability Pr[4 errors occur in n bits], thus it is reasonable to neglect this case. □

Remark. *For all even k such that $k \geq 8$ it is possible give a lower bound for the probability of undetected errors, since for every $k \geq 8$ we can find integers $m_1, m_2 \in \mathbb{N}$ such that $k = 4m_1 + 6m_2$. Thus for every case of k errors we can compose m_1 patterns of four-bit sequences as above and m_2 patterns of six-bit sequences such that these sequences have pairwise disjunct sets of error positions and the whole block of n bits is accepted as correct. However, this is only a lower bound, since not necessarily all undetectable error patterns of k bits (with k even and $k \geq 8$) can be decomposed in pairwise disjunct four-bit and six-bit sequences.*

In order to compare the PROFIBUS method with a standard 32 bit CRC checksum we have plotted the probabilities of Pr[$k = 4, 4$ errors not detected in l bytes, PROFIBUS method] and Pr[$k = 4, 4$ errors not detected in l bytes, 32 bit CRC method] for a BER of $p = 0.001$ and for varying l, see figure 4.2 (the upper curve belongs to the PROFIBUS method). In a 32 bit CRC checksum (for which the generator polynomial satisfies some constraints) 4-bit-errors are always detected, if the first and last erroneous bit have a distance of less than 32 and are accepted as correct with with probability $1/2^{32}$, if the distance is 32 or more bits ([14]).

Furthermore, all odd number of errors and two bit errors are always detected. We assume that no parity is transmitted, however start- and stopbit are needed for bit synchronization. Thus every byte is transmitted with 10 bits. For a 4 bit error pattern to be not detected under a 32 bit CRC checksum the following conditions necessarily have to be met:

- There must occur exactly four errors
- These four errors must not hit start- and stopbits
- They must span a range of 32 bit or more.

For the latter probability we look at the complementary event that the 4 bits are located within the same 31 bits. We can choose e_1 freely without including start- and stopbits, thus we have $8l$ possible positions. For e_2 there are 22 positions remaining (30 positions subtracting start- and stopbits), for e_3 we have 21 and for e_4 we have 20 positions, thus $L = 8l \cdot 22 \cdot 21 \cdot 20$. The total number of possibilities to place 4 errors in $10l$ bits is given by $K = \frac{(10l)!}{(10l-4)!}$. As a result the total number of possibilities for placing 4 errors such that they span a range of at least 33 bit is given by $\frac{K-L}{K}$. Then we have

$$\begin{aligned}\Pr[k = 4, 4 \text{ errors not detected in } l \text{ bytes, CRC}] &= \Pr[k = 4] \\ &\quad \cdot \Pr[\text{Errors have right positions in } l \text{ bytes} | k = 4] \\ &\quad \cdot \Pr[\text{The error pattern is not detected by CRC}] \\ &= b(4, 10l, p) \cdot \frac{K - L}{K} \cdot \frac{1}{2^{32}}\end{aligned}$$

The graph in figure 3 shows clearly, that the PROFIBUS error detection method is considerably weaker than a 32 bit CRC (cyclic redundancy check) for small packet sizes up to $l = 255$ (the maximum frame size in PROFIBUS) and a BER of $p = 0.001$. For $l = 15$ the difference is three orders of magnitude, for $l = 5$ it is five orders. In addition to that, if the frame has 24 bytes or more, the 32 bit CRC checksum uses fewer bits.

While the error detection method used in PROFIBUS is comparably weak, the most important control frame, the token frame, has no checksum at all. The rationale for this was probably that token frames are small and a PROFIBUS will not be operated in such a harsh environment, so the overhead for providing a checksum is not necessary. However, if the bit error rate is high and the token frame is hit by two-bit or four-bit errors such that they are not detected the following scenarios may occur for an active station A:

- The first corrupted token telegram of A can be corrupted such that the destination address is changed and the successor B of A feels itself skipped and removes from the

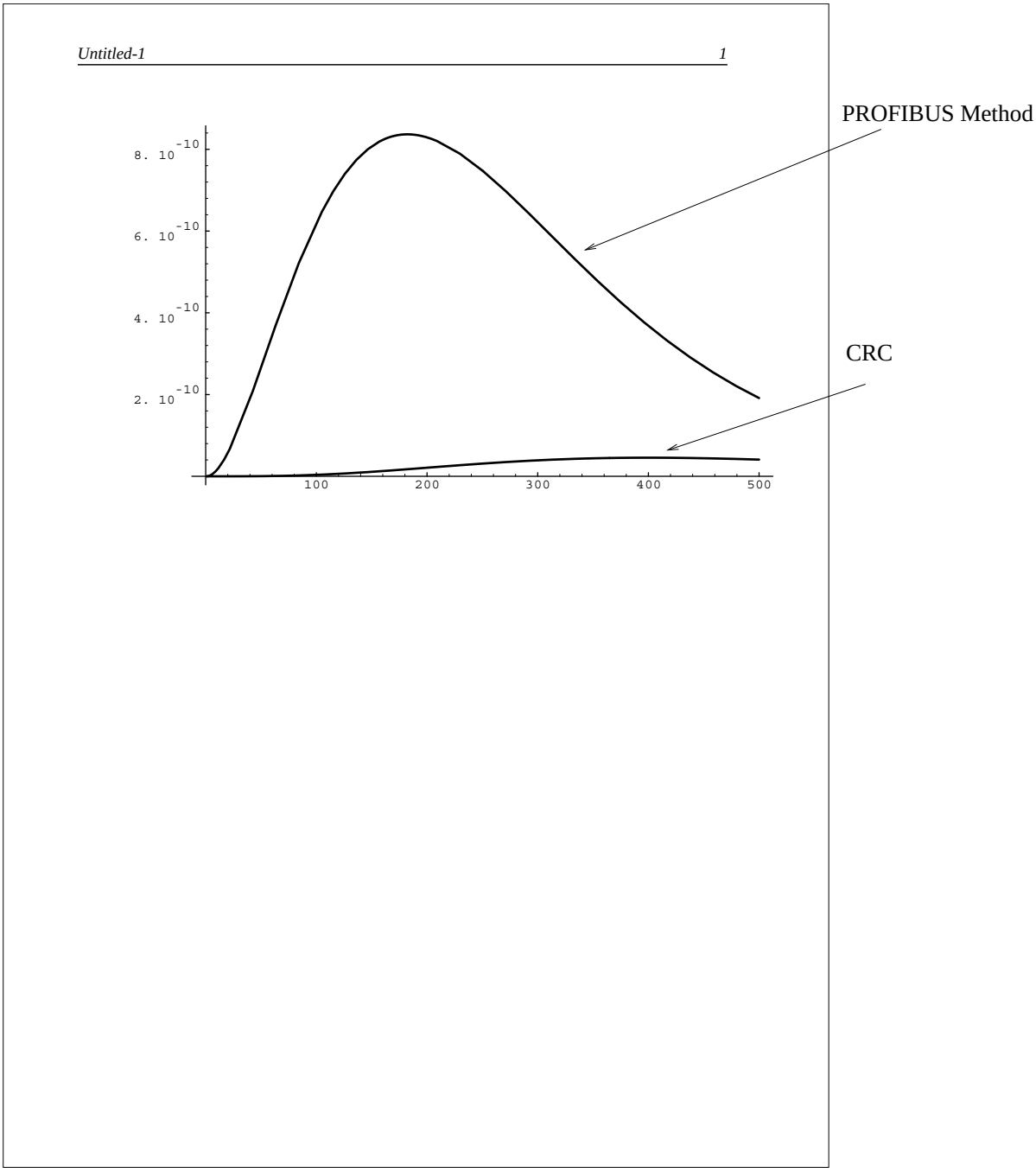

Figure 4.2: Probability of not detecting 4 bit errors with PROFIBUS method and CRC method

ring. Then the second token telegram will not be answered and B is lost from the ring. This scenario was observed a few number of times during our simulations.

- A is the only station in the ring, B is ready to enter. A sends token telegrams to itself (and continues transmitting data after that), but unfortunately the destination address is changed so that B is hit, which then enters the ring, accepts the token and starts transmitting. As a result collisions occur. This scenario also was observed a few times.

Chapter 5

Simulation Results

In this section we define our performance measures of interest, describe our simulation scenarios and present our results. We are only interested in the case that there is more than one active station. Our results do not apply for a PROFIBUS LAN with only a single active station. In short, we look at the following performance measures:

- Mean time between station losses (from the ring)
- Mean station outage time (i.e. the time necessary for re-including a lost station into the ring)
- Fraction of time that a station is in the ring
- Mean Delay

We have built a detailed simulation model using the CSIM simulation library [9]. This model includes the PROFIBUS link layer, the PROFIBUS MAC protocol and a shared medium with the property that all attached stations see the same signals and bits on the medium[1], this corresponds to a fully meshed topology. While all time intervals which belong to the behaviour of the medium (e.g. bit times, required idle times) are considered within the model, we assume that protocol processing time within the stations is negligible, with the exception of the station delay between a request telegram and the corresponding immediate

[1] The assumption of a common channel simplifies the model; it does not necessarily hold on wireless links. The assumption, that the sender also hears the (maybe errored) signals is critical to the behaviour, see below. With wireless technology it is often not possible for a sender to send and receive simultaneously on the same channel. However, we have decided to make this assumption in order to analyze the protocol behaviour under just the physical layer properties for which it is originally designed. Performance results for the case without hearback will be published in the future.

Parameter	Value
Bitrate	500 kBit/sec
Slot-Time	400 μsec
Max-Retry	3
Number of Active Stations	4
Number of Passive Stations	1

Table 5.1: Fixed Parameters for all simulations

Parameter	Value
TTRT	20 msec
Gap Factor	6

Table 5.2: Fixed Parameters for MTBSL

ack. The validation of the simulator was done by inspection. The simulations are carried out with 95 % confidence interval of width of 2 % of the absolute value, where appropriate. The confidence intervals are not shown in the figures. In our simulator an arbitrary number of active and passive stations can be defined, also the workload can be chosen free. The set of parameters which are fixed throughout all simulations is given in table 5.1. The active stations have the fixed station addresses 22, 39, 65 and 69 (taken once from uniform distribution).

5.1 Time Between Station Losses

The first performance measure we have investigated is the *mean time between station losses* (MTBSL), defined as follows. Every station alternates between two states: it is in the ring or not (more precisely: it feels itself being member of the ring or not). If it is not in the ring the station experiences an *outage time*. The time between two successive changes from the state of being a member to the other state is denoted as *time between station loss* (TBSL). This times are defined separately for every station. There are basically two different szenarios where station losses can actually occur:

- By the protocol A is required to perform a "hearback" while transmitting token telegrams (see sec. 2). After two subsequent hearback errors A removes itself immediately from the ring without any further transmissions (thus A is lost).

- The first corrupted token telegram of A can be corrupted in a way, that the destination address is changed (be aware that the token telegram has no checksums, only parity bits are used; see sec. 4) in such a way that the successor of A feels itself skipped and removes itself from the ring. Then the second token telegram will not be answered.

We have investigated the TBSL for different independent bit error rates and under the gilbert model with parameters given in sec. 3. All other parameters are fixed, they are summarized in table 5.2. There was no load in the system, thus there are only token frames and frames for pinging stations in the gap list, which occur on the medium. This was done in order to capture only the effects of bit errors to the involved protocol mechanisms and frames with no bias from data transmissions.

For the case of independent errors we show the mean TBSL in figure 5.1, the maximum observed TBSL are shown in figure 5.2. All times are given in seconds. For the special case of station 22 and a BER of 0.001 we show in figure 5.3 the distribution of the TBSL for that specific simulation run. We can see the following:

- The mean TBSL is for all stations nearly identical, except station 22 (lowest station address). The reason for this will be explained in the next section, the maximum observed TBSL shows the same trends.
- For higher bit error rates station losses occur very often.
- The distribution shown looks very similar to an exponential distribution. This is supported by the fact that for every simulated BER and for every station address the coefficient of variation of the TBSL is very close to 1, which is is also the coefficient of variation for the exponential distribution. The exponential distribution is the natural choice for the TBSL distribution, since station losses (coming from token errors) are independent from each other and exhibit the memoryless property.

Since by the time of this writing the set of simulations for the case of gilbert errors is incomplete, we show in table 5.3 for each station and for a Mean BER of 0.001 the MTBSL for the independent model and the gilbert model. One can see, that the increased burstiness of the errors under the gilbert model leads to an increased station loss frequency, even for station 22.

Station	MTBSL (sec) - Gilbert	MTBSL (sec) - Independent
22	2.31	3.31
39	1.13	1.60
65	1.18	1.62
69	1.10	1.52

Table 5.3: Comparison of MTBSL for MBER = 0.001 for Gilbert and Independent Errors

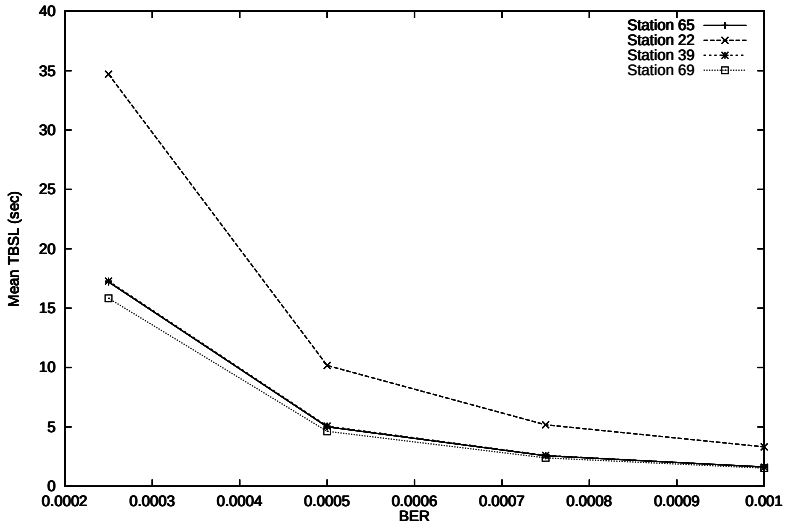

Figure 5.1: Mean TBSL vs. BER with independent bit errors

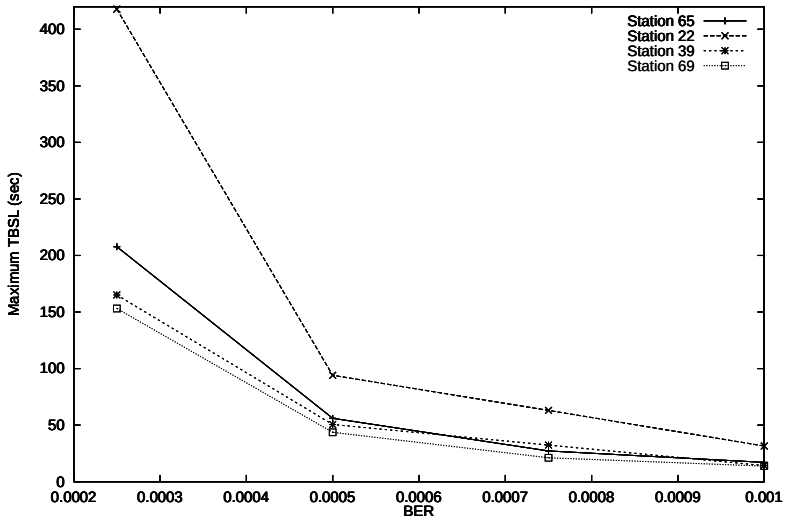

Figure 5.2: Maximum observed TBSL for independent bit errors

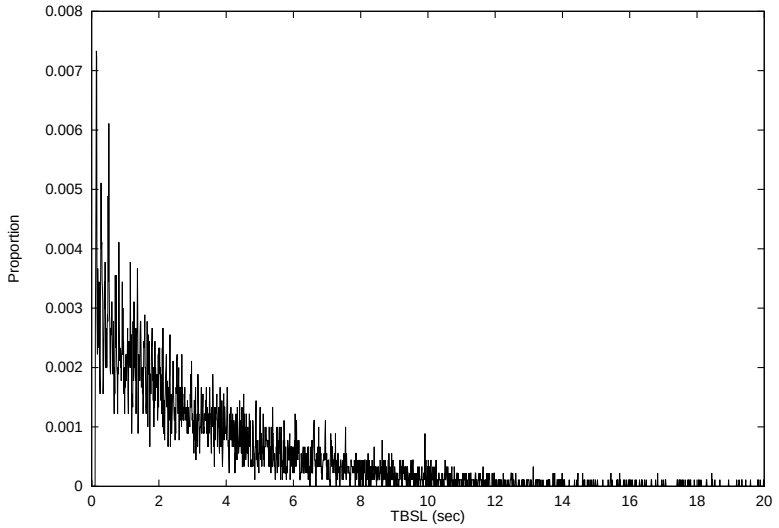

Figure 5.3: Distribution of TBSL for BER 0.001 (Independent Errors)

5.2 Station Outage Times

The next set of experiments investigates the *station outage times*, as defined in the preceding subsection. An outage time can occur due to the following scenarios:

- if station A wants to pass the token to its NS and there occur two successive hear-back errors, and no other station accepts the (erroneous) token, A discards the token immediately and removes itself from the ring.
- If A detects a token frame from its PS (say, Z) where the destination address is B with $B \neq A$ and $Z < A < B$ with respect to the ring ordering, then A interprets this as beink "skipped" by its predecessor and removes itself from the ring.

As described in section 2, after leaving the ring a station behaves as newly switched on and constructs a new LAS, which takes at least two successive token cycles[2].

We investigated the station outage times when varying three different parameters: the workload, the gap factor and the TTRT, all other parameters are fixed. Each parameter was separately investigated under the independent error model and under the gilbert error model, each with a (mean) BER of 0.001[3]. The load scenario is as follows: with every active station there is a single traffic source associated, which generates requests of fixed size with fixed interarrival time. The sources are synchronous. All the requests are addressed to the single passive station in the ring. The interarrival time was chosen to be 10 msec.

When varying the gap factor, the TTRT was chosen to be 20 msec and the request size was fixed at 14 bytes, thus yielding a load of approximately 20 % when all stations are in the ring (when the 9 bytes overhead of variable length telegrams are taken into account). For the case of independent errors we show the *mean station outage time* (MSOT) for every active station in figure 5.4, the *cumulated outage times* (defined as the fraction of time that a station is not in the ring) is shown in figure 5.5 and the observed maximum values for station outage time are shown in figure 5.6 (no simulation run was less than 8000 seconds). For the case of gilbert errors the MSOT is shown in figure 5.7, the cumulated outage times are shown in figure 5.8 and the maximum observed values are shown in figure 5.9. The following points are interesting:

[2]It is not necessary that the whole station behaves as newly switched on, however, for the network subsystem this is necessary.

[3]We have investigated only this high bit error rate since for an MBER of 0.0001 the influence of the ring maintenance mechanisms on the performance is much less, as is indicated by our results on the mean delay.

- 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 as follows: when station 22 experiences two successive hearback errors, it immediately removes itself from the ring after the second token telegram. It sends no further data, especially no token. As described in section 2, it then behaves as newly switched on, i.e. it has only an empty LAS. In this situation the token is lost and thus there is no activity on the medium. As a result, the timeout timer expires. But unfortunately, since 22 has the lowest station address, the timeout timer expires first for station 22, which then claims the token and thinks it is the only station in the ring, since there was no other transmission during the time between the removal of 22 and its timeout. Station 22 will send the next token to itself, all other stations feel themselves skipped and remove from the ring. It will take then some time to re-include the stations. By the definition of the ring inclusion algorithm it is then clear that the mean time needed for re-including the station with higher addresses increases almost linearly with the gap factor.

- The results on the cumulated station outage time are dramatic: for gap factors of around 30 all active stations except station 22 are only 50 % of the time member of the ring. 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 maximum observed values show a behaviour similar to the cumulated outage times. Even for the smallest gap factor some stations are be excluded for more than one second in our simulations.

- Under the gilbert error model both the MSOT and the cumulated SOT are slightly higher for all stations except station 22 than under the independent error model.

- Under the independent error model we have not observed any case where station 22 gets lost due to being skipped by erroneous token telegrams of other stations. However, under the gilbert model this has happened a few number of times.

A first conclusion is that the gap factor should be pretty low in order to achieve at least a bad result for the fraction of time of being a ring member (as compared to the unacceptable results for higher gap factors). However, this has the drawback that more bandwidth is wasted for

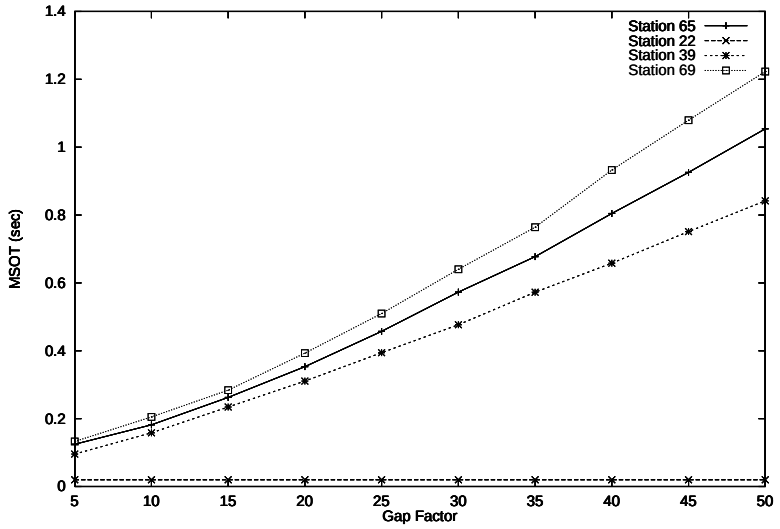

Figure 5.4: MSOT vs. gap factor (Independent Errors)

pinging stations. In practice it would also be a good idea to use consecutive station addresses in order to decrease the number of ping packets to unused station addresses.

When varying the TTRT the situation is almost equivalent to varying the gap factor, since the gap timer is the product of gap factor with TTRT. The gap factor was chosen to be 6 in all simulation runs, and also the 20 % load was used. The equivalence to the gap factor simulations is reflected in the simulation results, which look very similar to those for the gap factor, thus we can draw the same conclusions. The MSOT for independent errors are shown in figure 5.10, the respective cumulated SOTs and maximum observed SOTs are shown in figures 5.11 and 5.12, for the gilbert case please refer to figures 5.13, 5.14, 5.15.

As last experiment we have varied the load, while keeping TTRT (20 msec) and gap factor (6) fixed. For the load we have chosen to keep the request size fixed (10 bytes) and to vary the interarrival time from 5 msec to 10 msec. For the case of independent errors the MSOTs are shown in figure 5.16, the cumulative SOTs are shown in figure 5.17 and the maximum observed values are shown in figure 5.18. As compared to the gap factor and the TTRT the MSOT and cumulated SOT are not very sensitive against varying load. Even more, for increasing load (smaller IAT values) the cumulated SOT decreases. This can be explained by the fact that with higher load there occur less token telegrams per fixed unit of time and thus less occasions to get lost from the ring. However, even a cumulated outage time of approximately 4 % is not really acceptable for realtime applications.

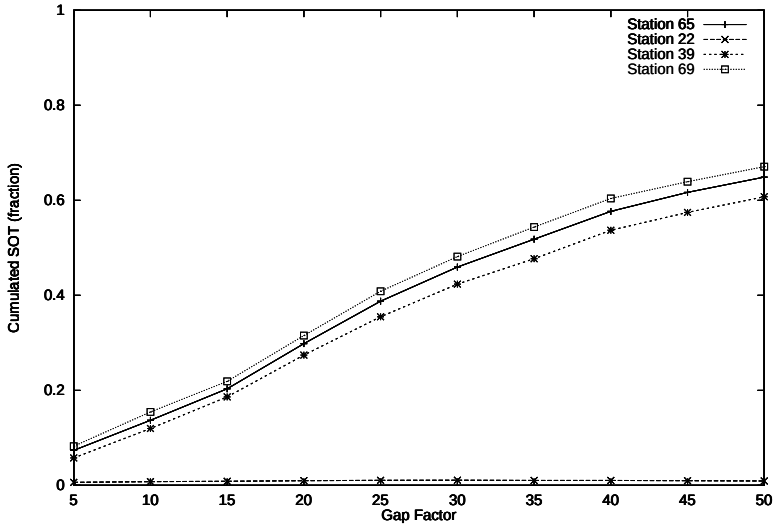

Figure 5.5: Cumulated SOT vs. gap factor (Independent Errors)

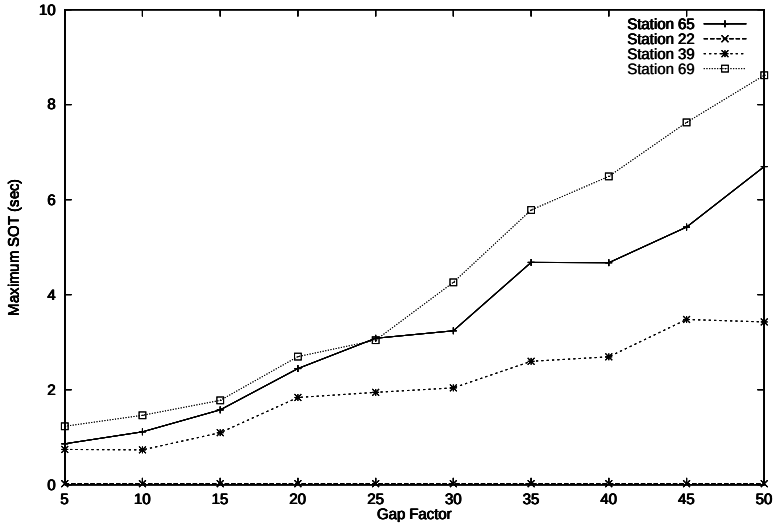

Figure 5.6: Maximum observed SOT vs. gap factor (Independent Errors)

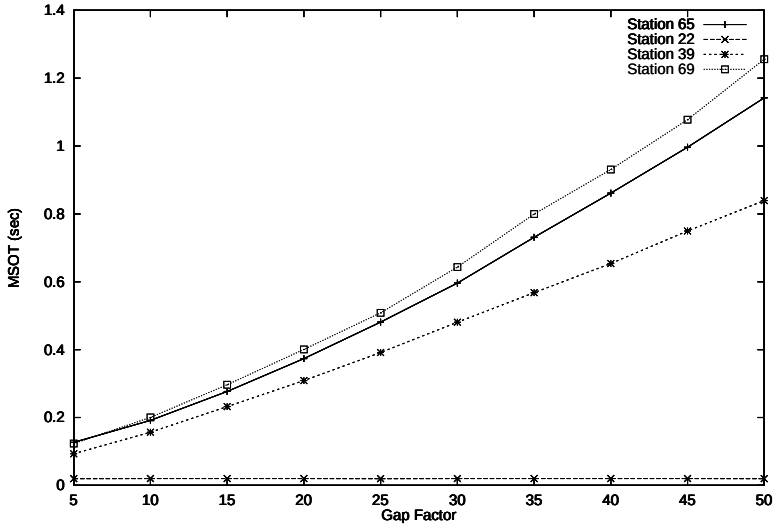

Figure 5.7: MSOT vs. gap factor (Gilbert Errors)

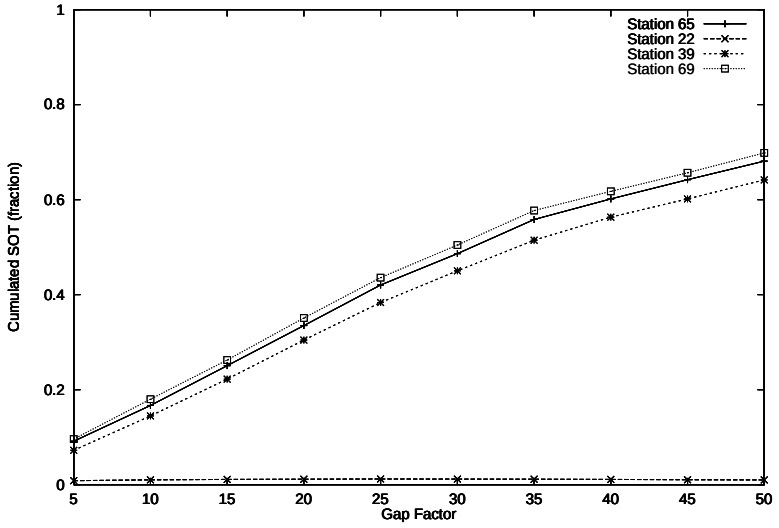

Figure 5.8: Cumulated SOT vs. gap factor (Gilbert Errors)

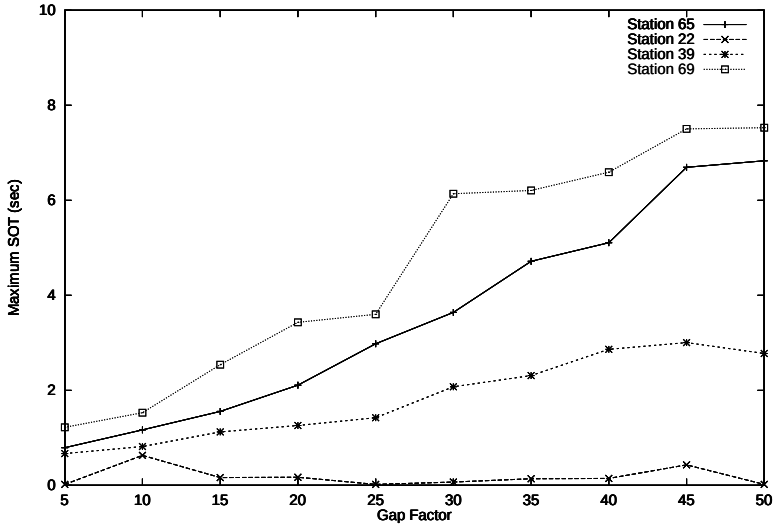

Figure 5.9: Maximum observed SOT vs. gap factor (Gilbert Errors)

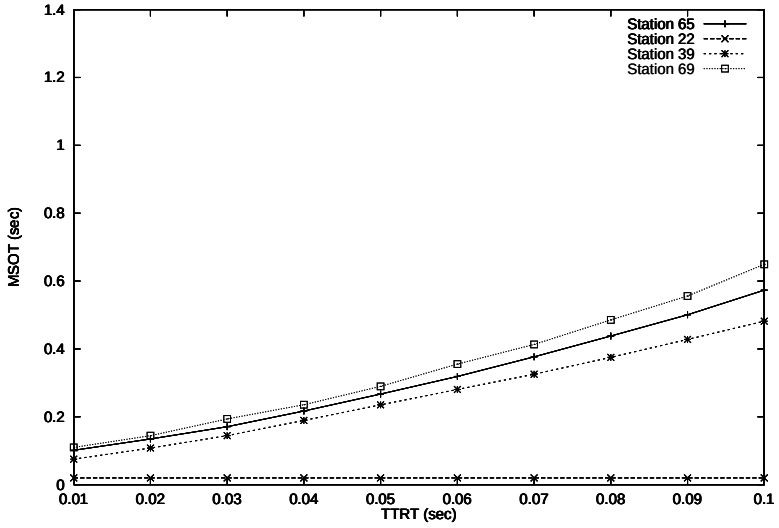

Figure 5.10: MSOT vs. TTRT (Independent Errors)

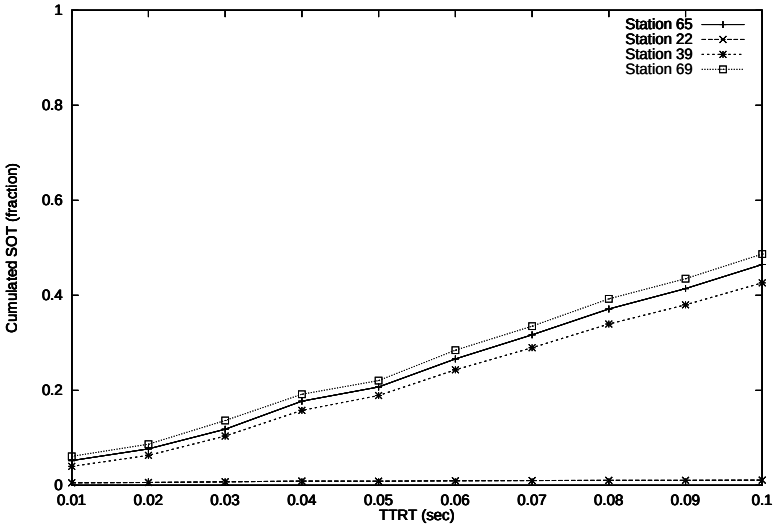

Figure 5.11: Cumulated SOT vs. TTRT (Independent Errors)

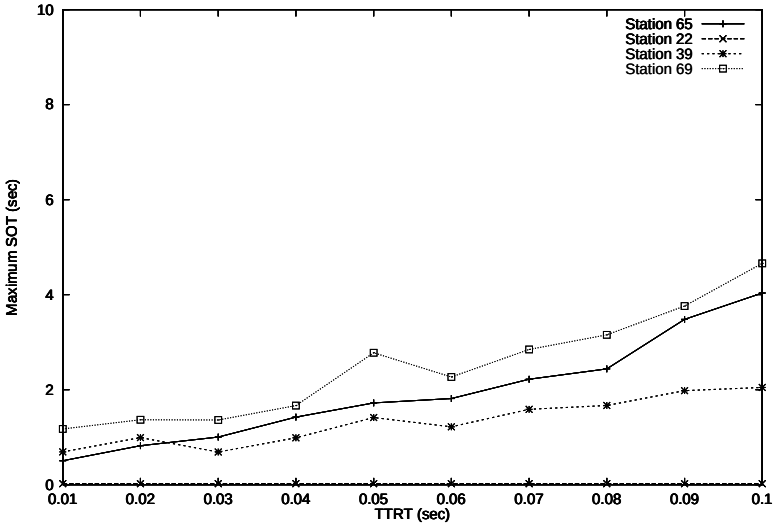

Figure 5.12: Maximum observed SOT vs. TTRT (Independent Errors)

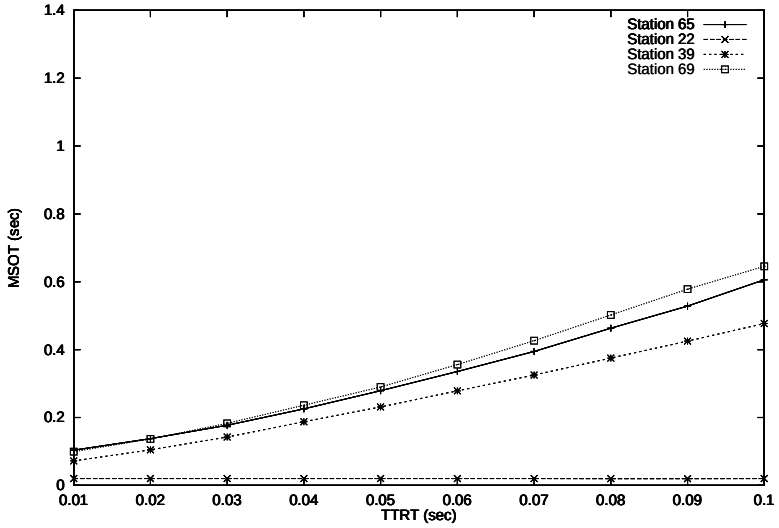

Figure 5.13: MSOT vs. TTRT (Gilbert Errors)

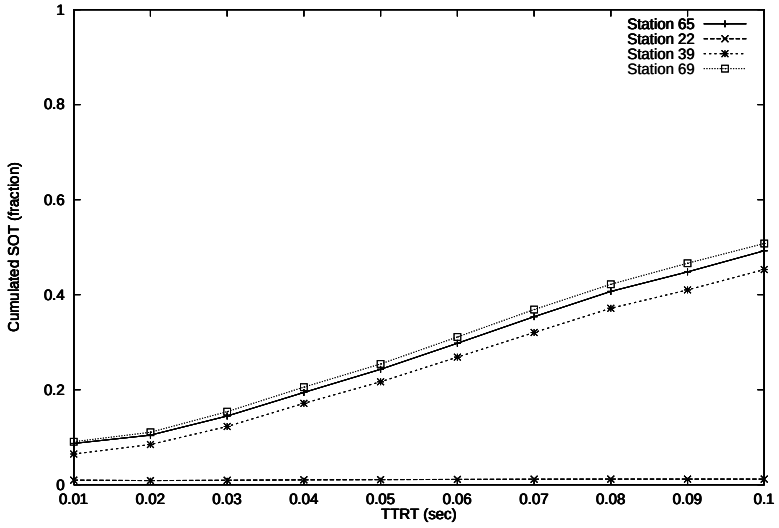

Figure 5.14: Cumulated SOT vs. TTRT (Gilbert Errors)

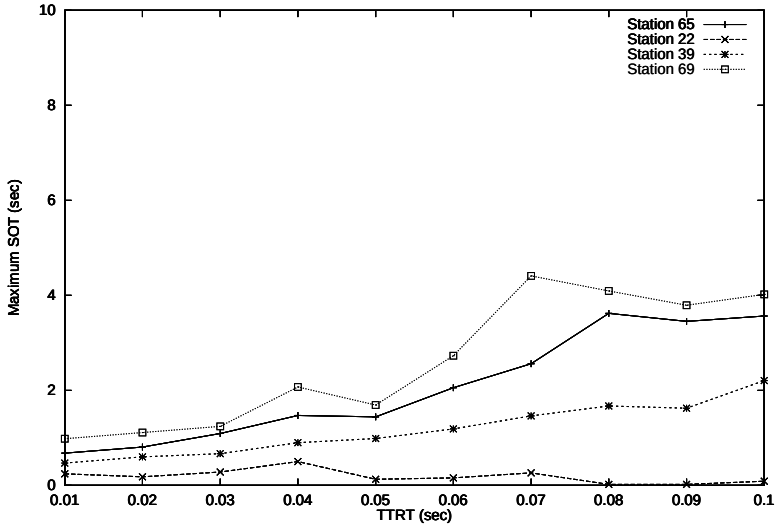

Figure 5.15: Maximum observed SOT vs. TTRT (Gilbert Errors)

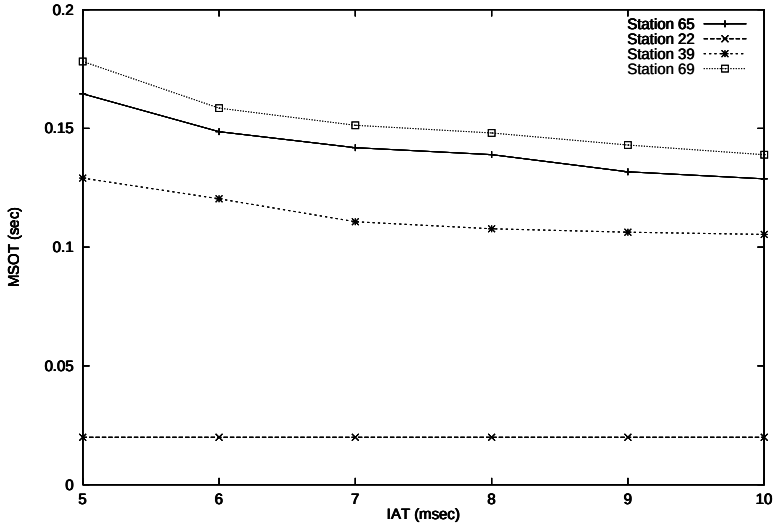

Figure 5.16: MSOT vs. IAT (Independent Errors)

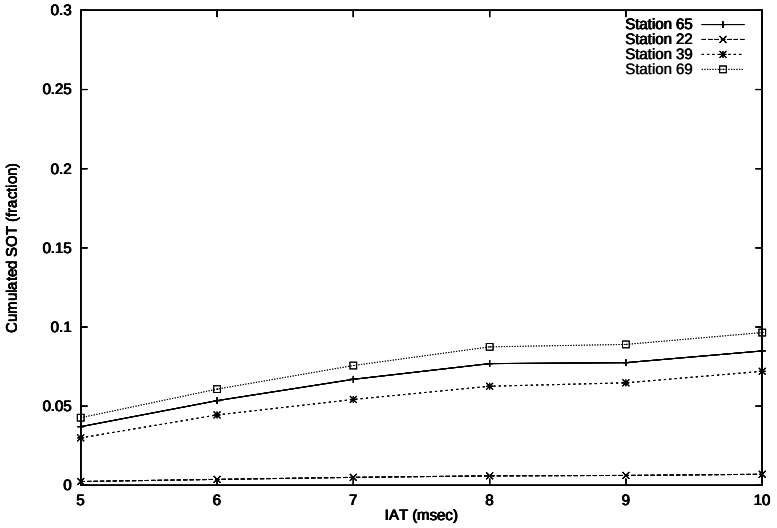

Figure 5.17: Cumulated SOT vs. IAT (Independent Errors)

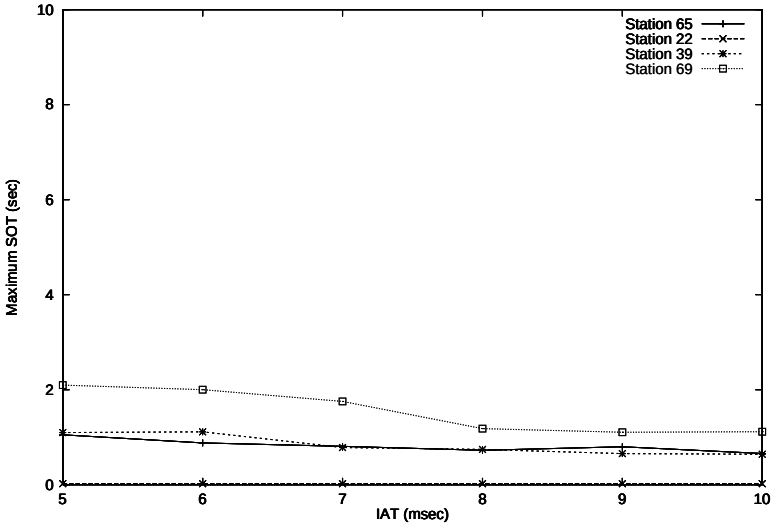

Figure 5.18: Maximum observed SOT vs. IAT (Independent Errors)

5.3 Mean Delay

In order to show the performance loss due to the PROFIBUS token passing and ring maintenance mechanism we have investigated the mean delay for different error rates and error models, while varying the load, all other parameters are fixed, see table 5.4. For the load we consider two cases, using the same number and distribution of sources as described above: in the first case we have varied the request size while the interarrival time (IAT) is fixed, the second case vice versa. We have investigated the mean delay under the normal PROFIBUS protocol and under an idealized protocol, where the token frames are transmitted correctly, but all other frames can be corrupted. The delay is measured at the link layer interface: it is defined as the time between issueing the request for transfer of data and the corresponding indication at the remote station (thus it includes any retransmission before the first correct reception). However, the delay is measured only for telegrams which are received correctly. For a proper interpretation of the curves we should note the fact that in our simulation every source does only generate requests when the corresponding station is currently member of the ring. This was done in order to keep the memory consumption of our simulator within the available memory. It is reasonable to expect significantly higher mean delays, when this restriction is removed.

In figure 5.19 we show the results for a mean bit error rate of 0.001, both under the independent error model and under the gilbert model, when the interarrival time was varied. Additionally, in figure 5.20 we show the maximum observed delay values and in figure 5.21 we show the observed coefficient of variation, which is a good index on the jitter introduced by channel errors. The results are worth some explanations:

- Under both error models the delay under the normal protocol is significantly higher than under the ideal protocol, as is the delay variation and the maximum observed delay. This is mainly due to frames which are already in the stations queue, when the station is lost from the ring[4].

- It can be seen from figure 5.19 that for the independent error model for both the idealized protocol and the normal protocol the mean delay increases with increasing load (decreasing interarrival time), however, for increasing load the graph of the normal protocol converges to the graph of the ideal protocol. This can be explained by the

[4] In the standards document it is left open what happens to the remaining requests, when the station is lost. In our simulations we have assumed that the requests remain in the queue, since the link layer interface has no concept of a deadline and there is no reason to discard the frames.

observation that for higher loads there are less token frames per fixed unit of time and thus less opportunities for a station to get lost from the ring. The same observation holds for both protocols under the gilbert error model.

- For higher loads (interarrival time smaller than 7 msec) the ideal protocol under gilbert errors behaves worse than the normal protocol under independent errors. This can be explained by the bursty nature of channel errors: while the channel is in bad state, it is likely that a single frame experiences several consecutive retransmissions and thus stays longer in the queue. Due to the shorter interarrival times it is likely that new requests arrive meanwhile, which then queue up behind the first one and will be delayed longer.

- For lower loads (IAT greater than 7 msec) the ideal protocol under gilbert errors behaves better than the normal protocol under independent errors and approaches the ideal protocol under independent errors for the lightest loads. We think this is due to two circumstances: under the ideal protocol no station gets lost (no frames queue up during outages) and for lighter loads one can expect that the queues are almost empty or contain a single frame even under gilbert errors.

- The normal protocol under gilbert errors has always approximately twice the mean delay as the ideal protocol under independent errors.

- The maximum observed delays and the delay variance for the normal protocol under both error models is significantly higher than for the idealized protocol. Thus the loss of control frames has a significant impact on delay variation.

- The coefficient of variation gets smaller for higher loads, the delay variances "smoothes out".

The results indicate that the protocol is sensitive to the loss of control frames and that there is a serious performance degradation with respect to delays. Especially for higher loads the protocol behaves more bad under the gilbert model than under the independent model.

In figure 5.22 we show the mean delays for a mean bit error rate of 0.0001, also under the independent error model and under the gilbert model with varying interarrival times. The curves are almost identical. This is an indication that for this MBER the loss of control frames has no significant impact on the mean delay The main reason is that it is very unlikely to loose two or three consecutive token frames as compared to the MBER 0.001.

For the next experiment we have chosen to keep the interarrival time fixed (10 msec) and to vary the request size in the range from 10 to 30 bytes user data in steps of five bytes. The

Parameter	Value
TTRT	20 msec
Gap factor	6

Table 5.4: Fixed Parameters for Mean Delay

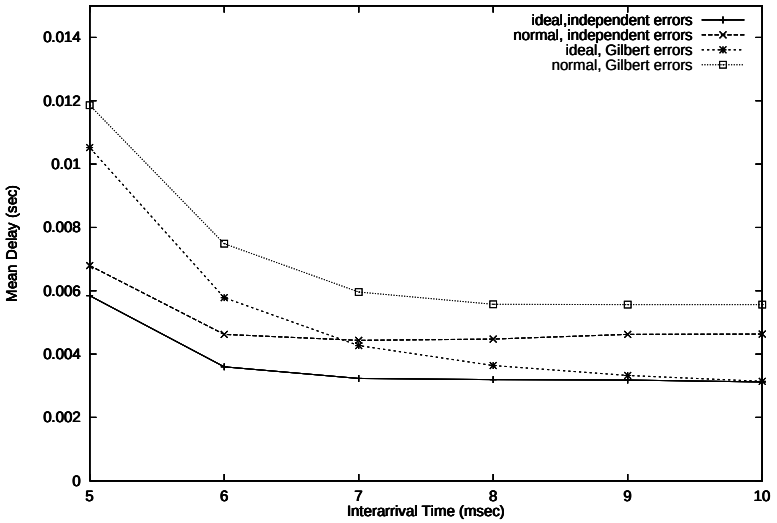

Figure 5.19: Comparison of Mean Delay with normal and idealized protocol under both error models with MBER = 0.001

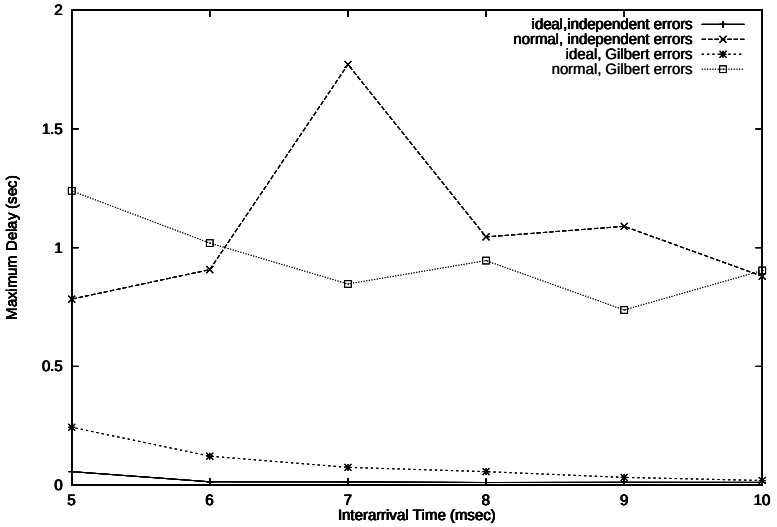

Figure 5.20: Maximum observed delay for MBER = 0.001

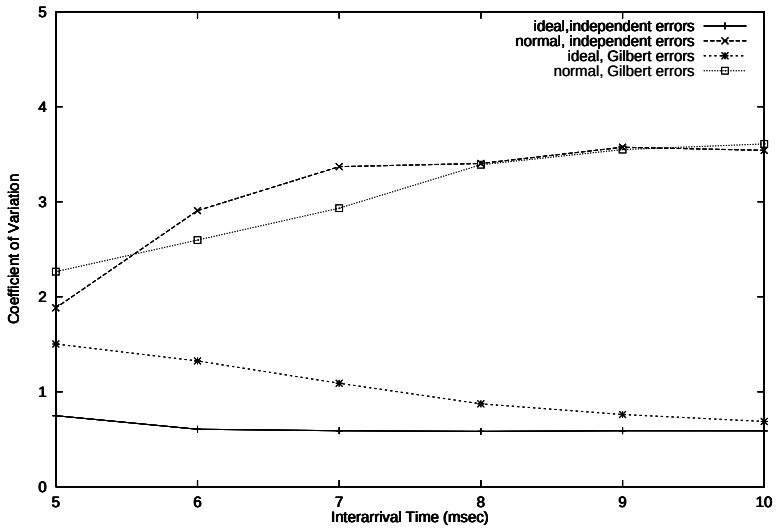

Figure 5.21: Coefficient of Variation for MBER = 0.001

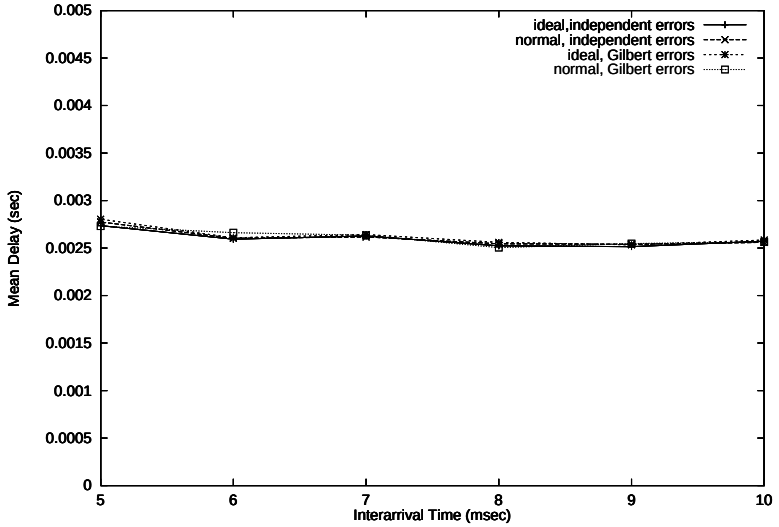

Figure 5.22: Comparison of Mean Delay with normal and idealized protocol under both error models with MBER = 0.0001

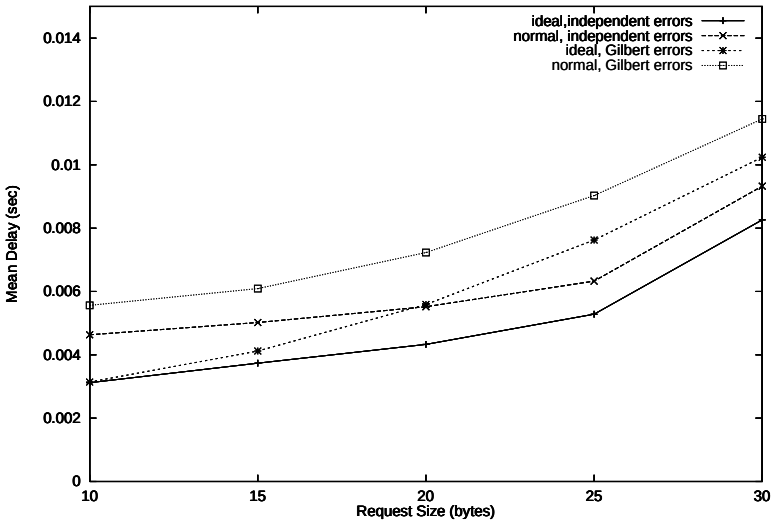

Figure 5.23: Comparison of Mean Delay with normal and idealized protocol under both error models with MBER = 0.001

mean delays for the case of MBER equal to 0.001 are shown in figure 5.23, the corresponding maximum values and coefficients of variations are shown in figs. 5.24 and 5.25 respectively. We can observe almost the same trends as in the case of varying the interarrival time. In addition, for the case of MBER equal to 0.0001 we again can see that the mean delay curves are very close to each other, thus that the influence of control frame loss is not visible for this bitrate (see figure 5.26).

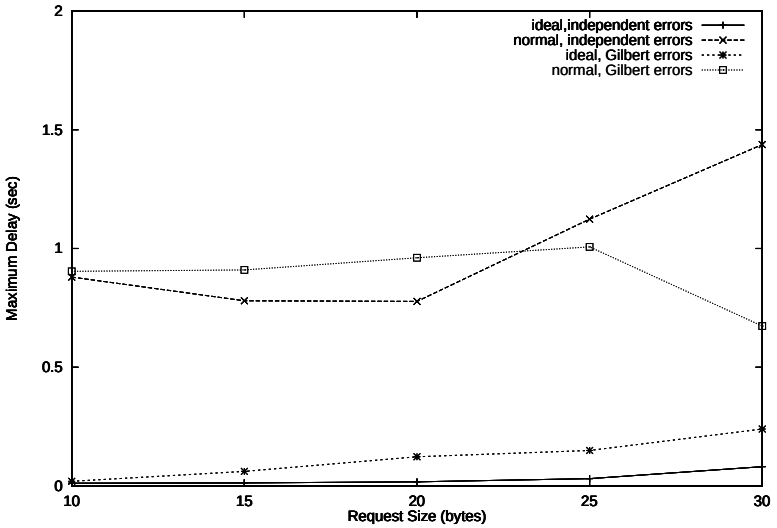

Figure 5.24: Maximum observed delay for MBER = 0.001

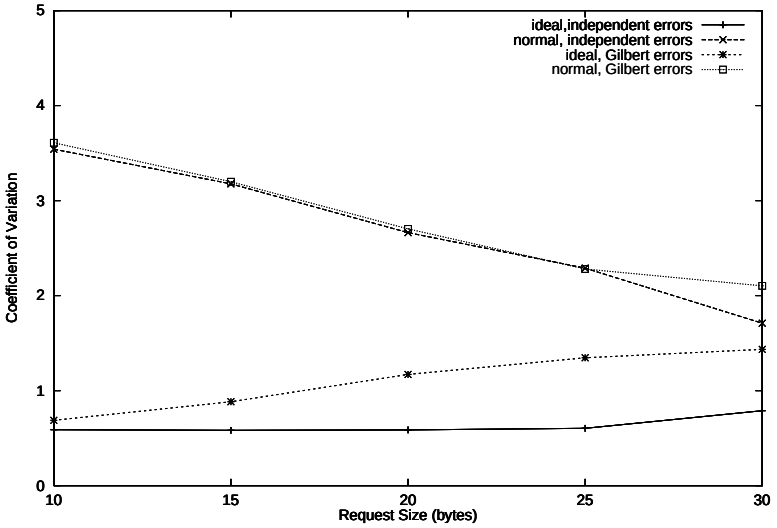

Figure 5.25: Coefficient of Variation for MBER = 0.001

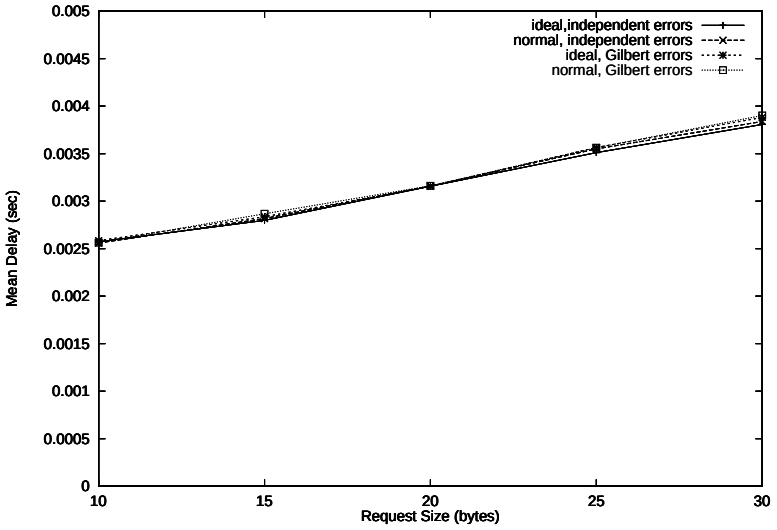

Figure 5.26: Comparison of Mean Delay with normal and idealized protocol under both error models with MBER = 0.0001

Chapter 6

Improvements

In this section we describe some modifications of the protocol and the frame formats. The basic protocol is leaved untouched. Two modifications are investigated using simulations. The first one is a new timeout calculation method and the second one is fast re-inclusion of lost stations. Both proposals have the advantage that changes to the protocol implementation can be done local, i.e. it is possible to incorporate both schemes in only a subset of stations. Both schemes provide a significant improvement in different performance parameters.

6.1 Timeout Calculation

From our simulations and from analysis we have observed that the following situation often occurs: if A is the station with the lowest address and A passes the token to a successor and experiences two successive errors in its token frame (hearback errors), it immediately throws away the token and behaves as newly switched on (it goes into the state "listen token", see the standards document). As a result there is no traffic on the bus until the timeout timer expires at an active station. Each station maintains this timer. It is resetted anytime when the station receives some data on the bus. If n is the station address, then the timeout value is set to

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

where T_{SL} is the slot time. However, in our case it is the timeout timer of A which expires first. Since A is newly switched on and there was no traffic on the bus (especially no token frames) A cannot construct a valid LAS and thinks, that it is the only station in the ring. After timer expiration A claims the token and sends a token frame to itself, thus announcing

that there is a logical ring and A is the only member. As a result, all other stations feel skipped and remove themselves from the ring.

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 LAS. If the timer of a station in the ring (not in "listen token") expires, the ring keeps alive. Thus we propose to make the timeout calculation state dependent, the following way:

$$T_{TO} = \begin{cases} 6 \cdot T_{SL} + 2 \cdot n \cdot T_{SL} & : \text{ Station is not in "listen token"} \\ (254 + 6) \cdot T_{SL} + 2 \cdot n \cdot T_{SL} & : \text{ Station is in "listen token"} \end{cases}$$

in order to make sure that the timeout timer expires first for stations in the ring and as a result to avoid this scenario. In figure 6.1 we show the mean delays for the ideal protocol, the normal protocol and the modified protocol for varying request size and independent errors (the remaining scenario is the same as in section 5.3), and in figure 6.2 we show the same for the gilbert model. It can be seen that there is a significant improvement in the mean delays especially for lower loads (with higher station loss rate). But what is much more interesting is the improvement (reduction) in the cumulated station outage time. In order to show this we have performed the same simulations as described in subsection 5.2 for varying the gap factor. However, simulation duration was chosen to be fixed (10000 seconds for every run, the range of simulation durations reported on in subsection 5.2 varies between 8000 seconds and 30000 seconds). We show here only the results for the case of independent errors. The cumulated outage times for all stations are shown in figure 6.3. It can be seen that the modified timeout computation has two effects: it significantly reduces the cumulated outage times for all stations (except for station 22) and it makes the behaviour more symmetric, i.e. all stations behave roughly equal. However, the nearly linear dependency of the choice of the gap factor choice remains visible.

In addition it should be mentioned that the modified timeout calculation is a purely "local" algorithm, i.e. it is possible to use it only in a subset of stations without the need to modify the remaining stations implementation.

6.2 Frame Formats

Perhaps the most obvious change in the frame formats is to equip the token frame with a checksum. We believe that an 8 bit CRC checksum suffices. Furthermore, the analysis in section 4 has shown that for most frames of the use of a 32 bit CRC is more effective, both in terms of error detecting capabilities and, for more than 24 bytes user information, in the

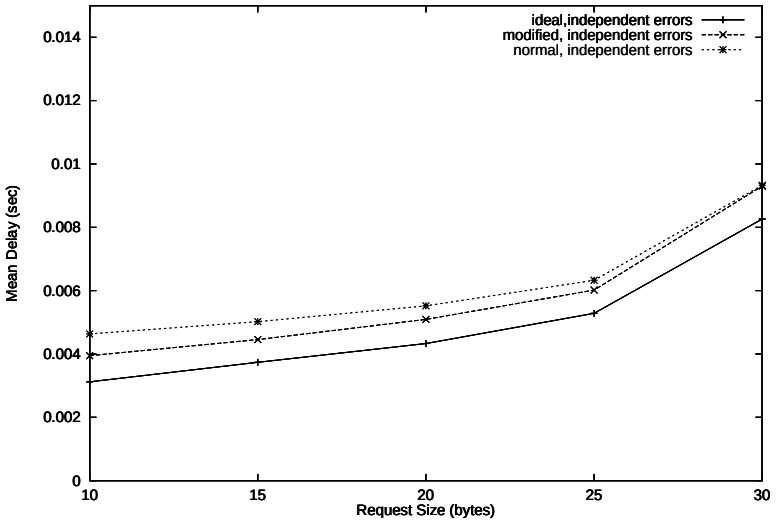

Figure 6.1: Comparison of Mean Delay with normal, modified and idealized protocol under independent error model with MBER = 0.001

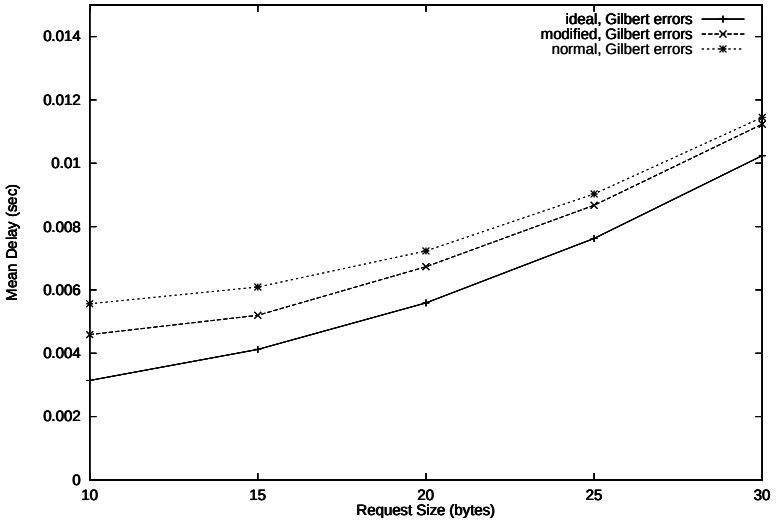

Figure 6.2: Comparison of Mean Delay with normal, modified and idealized protocol under gilbert error model with MBER = 0.001

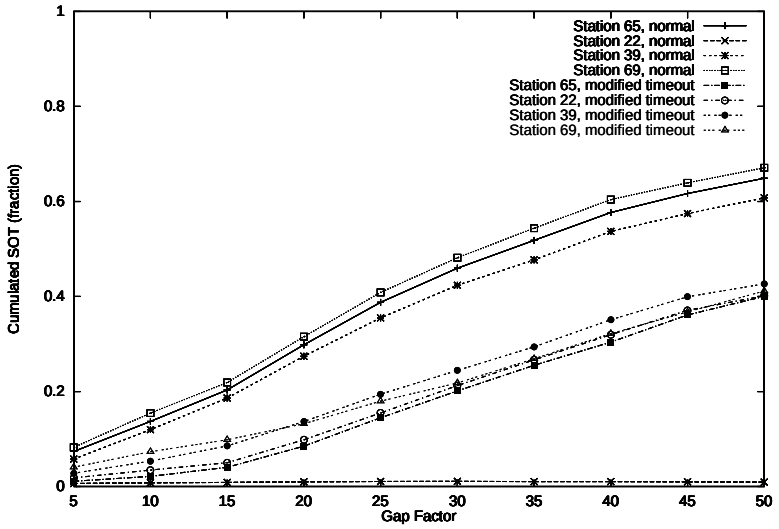

Figure 6.3: Comparison of cumulated station outage times for normal protocol and modified protocol (Independent Errors)

number of bits, which need to be transmitted. Thus it makes sense to equip the variable sized frame with a 32 bit CRC. For the smaller formats (no data frame and data frame with fixed and small size) we propose to use a 16 bit CRC.

6.3 Protocol Enhancements

In this subsection we propose some further enhancements which modify the protocol slightly. We have not implemented these enhancements in our simulator, with the exception of the fast re-inclusion scheme described below.

6.3.1 Increasing Number of allowed Hearback Errors

In our simulations we have observed that by far the most station losses occur due to repeated hearback errors. The obvious countermeasure is to increase the number of allowed hearback errors to at least three.

In the original standard the hearback feature is required for two reasons:

- to perform a self test of the transceiver.
- to resolve collisions e.g. after switching on all stations (see section 2) or after erroneous token frames, where no error is detected and the token "hits" another station in the

ring which accepts the token.

However, this feature assumes that it is possible to send and receive simultaneously, which is often not possible with wireless radio modems. So for wireless technology another mechanism is needed. Especially for the collision detection / resolution feature another solution needs to be found. A very simple solution is to omit the hearback error, hoping that a collision can be resolved when station A sends its token while station B is transmitting something else, so that A thinks its successor has accepted the token and stops transmitting data until receiving the next token addressed to A. However, this scheme will not work when station A and B transmit in perfect synchronisation.

6.3.2 Fast Re-Inclusion of lost stations

When a station gets lost from the ring, it may take a while before it is re-included, as is shown in subsection 5.2. First, the station is required to observe twice the same sequence of token frames, second it will not be reincluded before it is pinged by its successor using the Request-FDL-Status frame (see section 2). We propose to add the following extra feature to the protocol: after station A has lost its successor B (i.e. there is no reaction 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 re-included. However, B should only be included, when B lies in the range between A and A's new successor C, otherwise C will be kicked out. This procedure should not affect the normal operation of the station inclusion algorithm. This protocol extension can also be implemented in a "local" way, since it is not necessary to introduce new frame formats or to change all stations protocol implementation.

However, using this protocol extension makes only sense when the timeout calculation method described in subsection 6.1 is also used, because otherwise the ring behaviour is dominated by the effect generated by double hearback errors of the lowest station and fast re-inclusion gets no chance to work.

We have investigated the effects of joint operation of fast reinclusion and new timeout calculation on the cumulated station outage times for varying gap factors and for the case of independent errors. The simulations were run for 10000 seconds, the results are shown in figure 6.4. We can make two observations:

- for all stations but 69 there is again a drastic decrease in cumulated outage times, as compared to the case where only the new timeout calculation is used. The dependence

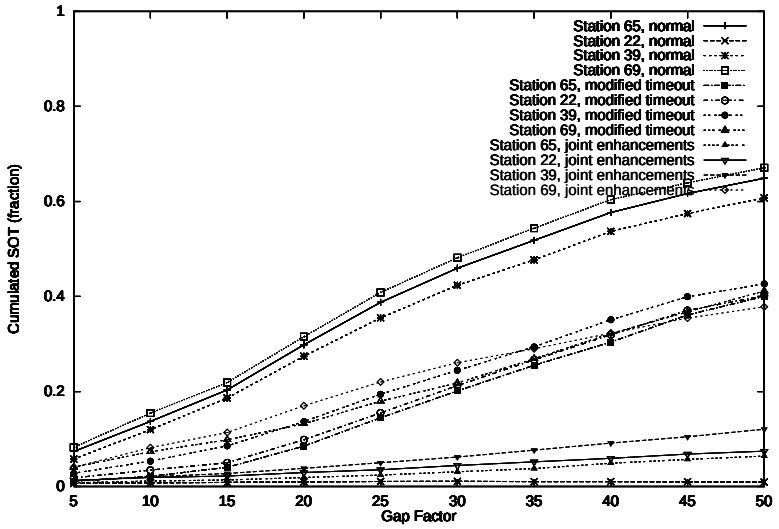

Figure 6.4: Comparison of cumulated station outage times for normal protocol, the protocol with new timeout calculation and the protocol with both new timeout calculation and fast reinclusion (Independent Errors)

on cumulated outage times of the gap factors is nearly linear. However, since values of 15 % are reached, the results are still not good.

- A significant asymmetry is introduced, since station 69 behaves significantly worse than all other stations, however, the achievements of the timeout calculations remain visible.

6.3.3 Ring-Inclusion using Random Access Slots

An alternative approach for including new stations in the ring is as follows: instead of requiring A to ping every address in its gap list from time to time A may send a special control frame, carrying the first and the last address of its gap list. Every station within this range that wants to enter the ring, answers immediately with a short packet. If there are no stations answering, A performs no further action until the next time sending the new control frame. If A receives a correct answer from a station willing to enter the ring, it shortens its gap list and passes the token to the new station. If A receives an erroneous frame, this can have two reasons: channel errors or collisions in case that there is more than one new station. In both cases A starts now to ping every address within its gap list until the list is empty or A finds a new station. This approach has two advantages:

- It is deterministic when the control frames are sent in regular intervals.
- Bandwidth is saved when no stations want to enter the ring.

However, the main disadvantage of this protocol is the necessity to introduce a new frame format and thus the need to change the implementation of all stations within the ring.

Chapter 7

Discussion and Conclusions

In this paper we have gained some insight in the dynamics and the behaviour of the PROFIBUS token passing protocol over error prone links. We see the following main results:

- The protocol is very sensitive to loss or corruption of control frames, especially token frames. If some parameters are chosen bad (e.g. gap factors), the ring breaks completely down, at least under the relatively high mean bit error rate of 10^{-3}. While significantly increased mean delays are a serious performance problem, the high percentage of cumulated station outage times is a catastrophe.
- The results are asymmetric in the sense that the station with the lowest address receives much better performance than all other stations. And even within the remaining stations higher station addresses are a penalty. One can say that the lowest station determines the fate of the ring.
- If the MBER is a magnitude smaller (10^{-4}) then things look better and station losses are rare.
- For delays and outage times the protocol has shown to be more sensitive against bursty error behaviour than for "smooth" independent errors.
- Frame formats and the error detection capabilities are designed for low error rates. Under higher error rates they are outperformed by CRC checksums.

We have proposed some modifications to the protocol and its parameters. A significant effect can be observed for making the timeout value T_{TO} state dependent, which additionally is only a very small change to a protocol implementation. A second modification yielding significant improvements is fast re-inclusion. Both methods have the advantage that it suffices

to modify only a subset of stations, however, the results are still not good for realtime applications. We can conclude that the PROFIBUS protocol is not designed for error prone links, even if some modifications are done.

However, we must note that all results rely critically on the assumption, that a station can hear its own transmissions ("hearback" feature). In our future research we will investigate the PROFIBUS performance and ring stability in the case without the hearback feature.

In addition, another problem of using the PROFIBUS protocol over wireless links was not covered within this study: the requirement for full reachability. This imposes a serious restriction on the geographical coverage of a single wireless PROFIBUS LAN.

Bibliography

[1] 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.

[2] DIN - Deutsches Institut für Normung, Beuth Verlag Berlin. *DIN 19245 Teil 1 - PROFIBUS: Übertragungstechnik, Buszugriffs- und Übertragungsprotokoll, Dienstschnittstelle zur Anwendungsschicht, Management*, April 1991.

[3] 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.

[4] D. Duchamp and N.F.Reynolds. Measured performance of wireless lan. In *Proc. of 17th Conf. on Local Computer Networks, Minneapolis*, 1992.

[5] 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.

[6] E. O. Elliot. Estimates of error rates for codes on burst-noise channels. *Bell Syst. Tech. J.*, 42:1977–1997, September 1963.

[7] E. N. Gilbert. Capacity of a burst-noise channel. *Bell Syst. Tech. J.*, 39:1253–1265, September 1960.

[8] 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.

[9] Mesquite Software, Inc., T. Braker Lane, Austin, Texas. *CSIM18 Simulation Engine – Users Guide*, 1997.

[10] Randolph Nelson. *Probability, Stochastic Processes, and Queueing Theory – The Mathematics of Computer Performance Modeling.* Springer Verlag, New York, 1995.

[11] K. Pahlavan and A.H. Levesque. *Wireless Information Networks.* J. Wiley and Sons, 1995.

[12] PROFIBUS Nutzerorganisation e.V., PROFIBUS Nutzerorganisation e.V., Haid-und-Neu-Str. 7. *PROFIBUS – Entwurf Technische Richtlinie – Implementierungshinweise zur DIN 19245 Teil 2*, December 1993.

[13] PROFIBUS Nutzerorganisation e.V., PROFIBUS Nutzerorganisation e.V., Haid-und-Neu-Str. 7. *PROFIBUS – Entwurf Technische Richtlinie – Implementierungshinweise zur DIN 19245 Teil 1*, August 1994.

[14] John D. Spragins, Joseph L. Hammond, and Krzystof Pawlikowski. *Telecommunications – Protocols and Design.* Addison-Wesley, Reading, Massachsetts, 1991.

[15] Jonathan Stone, Michael Greenwald, Craig Partridge, and James Hughes. Performance of checksums and crc's over real data. *IEEE/ACM Transactions on Networking*, 6(5):529–543, 1998.

[16] 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.