Technical University Berlin

Telecommunication Networks Group

Intermediate Checksum Schemes in the Presence of Deadlines

Andreas Willig

awillig@tkn.tu-berlin.de

Berlin, January 2008

TKN Technical Report TKN-08-002

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

Almost all Automatic Repeat reQuest (ARQ) protocols rely on checksums to let the receiver decide about the presence of transmission errors. If the checksum is wrong a retransmission takes place. In case of transmission errors often only a few bits are erroneous, but in a frame retransmission *all* bits are transmitted again, including the correct ones. This paper introduces the so-called *intermediate checksum framing scheme*, which attempts to rescue most of the correct bits and to restrict retransmissions only to those parts of a frame where bit errors actually occured. We specifically consider the case where a deadline is associated to packets and the number of retransmissions is bounded.

Contents

1 Introduction **2**

2 The Intermediate Checksum Approach **3**

3 Markov Model **5**

4 The case of an unbounded number of retransmissions **8**
4.1 Expected number of frames 8
4.2 Expected number of bits 12

5 The case with a deadline **14**
5.1 The complete scheme 14
5.2 The reduced and the redundant schemes 17

6 Selection of Chunk Sizes **21**

7 Related work **29**

8 Conclusions **30**

Chapter 1

Introduction

Almost all Automatic Repeat reQuest (ARQ) protocols [17, 10] rely on *checksums* to let the receiver decide about the presence of transmission errors. If the checksum is wrong, the receiver provides the transmitter with appropriate feedback, which triggers a frame retransmission. When the channel bit error rate is not too high, often only a few bits are erroneous, but in a frame retransmission *all* bits are transmitted again, including the correct ones. In [26] the author has introduced the so-called *intermediate checksum framing scheme* (ICF), which attempts to rescue most of the correct bits and to restrict retransmissions only to those parts of a frame where bit errors actually occured; a similar idea is briefly sketched in [16]. A distinguishing feature of the intermediate checksum approach is that it does not rely on coding, but requires only the ability to compute checksums. Since this can easily be done in software, the intermediate checksum scheme is readily implementable on top of commercial transceivers, like for example the ChipCon CC2420 transceiver that is compliant to the IEEE 802.15.4 standard [1].

This paper differs from [26] in the following ways:

- The comparison of transmission with the classical header-data-checksum scheme and the intermediate checksum scheme for the case of an unbounded number of retransmissions is refined, and in addition the average number of bits required until success is evaluated.
- The design of intermediate checksum schemes under a deadline constraint is investigated and different design options are compared and evaluated.

In this paper we frequently assume binary symmetric channels (BSC), i.e. a channel model in which bit errors occur independently and with the common bit error rate $p \in (0, 1)$. Furthermore, we make throughout the assumption that p is known. In [26] a simple scheme for estimating p from the success or failure of packet and chunk transmissions is described. Therefore, in this paper this issue is neglected.

Chapter 2

The Intermediate Checksum Approach

The traditional packet format used in many protocol is shown in the upper part of Figure 2.1.

This traditional format consists of a packet header of size o bits, the user data of size s bits and a packet trailer of size h bits, which is usually the checksum. The overall packet size is thus $l = o+s+h$. The checksum covers both the packet header and the data part. When the receiver detects a checksum error, it drops the whole packet and sends a negative acknowledgement. Dropping a whole packet, however, is a waste of information, since typically only a few bits in a packet are wrong.

Many schemes have been devised to make effective use of the information contained in the erroneous packet copy. Such schemes are called type-II or type-III hybrid-ARQ schemes [17, 12, 13]. A simple example of a type-II ARQ scheme is bit-by-bit majority voting: Once the receiver has received at least three erroneous versions of the same packet, it can guess what the received packet should be by applying a majority voting procedure to all bits. Other schemes are discussed for example in [25, 12].

The scheme we consider here is constructed in another way. The problem with using a single checksum for the whole packet (as in the classical scheme) is that we cannot infer any information about positions of bit errors, so each bit is in suspect. By segmenting the packet into smaller chunks such that each chunk is equipped with an own checksum, the error information can be localized and the information in correct chunks does not need to be thrown away. A key advantage of this scheme is that it does not rely on coding but on (much easier) checksum computations. This allows implementation on top of commercial transceivers like the ChipCon CC2420 transceiver [1], and furthermore checksum computations on the receiver side are much more energy-efficient than decoding algorithms.

To put this approach into a protocol, some additional rules are needed: in the intermediate checksum scheme the s user data bits are partitioned into L *chunks*, each having a raw size of $c = s/L$ bits,

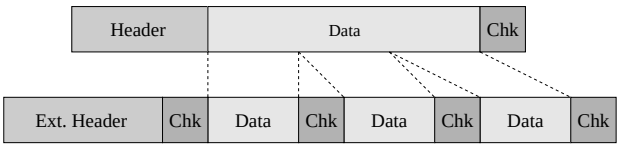


Figure 2.1: Traditional packet format and intermediate checksum packet format

to which a checksum of h' bits is appended (for simplicity we do not consider slack chunks). A frame is formed by appending all the chunks to a frame header of size $o' \geq o$ bits, and the overall frame has size $o' + h + L \cdot (c + h) > o + s + h$ bits. The increased header size $o' > o$ results from the fact that the header needs an extra field for the intermediate checksum scheme. Specifically, the chunk size c needs to be stored. Furthermore, the MAC header and the extra field need to be protected by a separate header checksum of size h bits. For simplicity in this paper we assume that $h' = h$ holds.

The transmitter transmits this initial frame. The receiver behaves as follows: if it detects an error in the frame header, the whole frame is discarded and the transmitter has to retransmit the full frame. If the header is correct, the receiver checks each chunk separately and buffers the correct chunks. If the receiver finds all chunks in its buffer, it delivers the frame to its upper layers and sends a *final acknowledgement*. In the other case, different schemes are investigated in this paper:

- ***Complete scheme***: The receiver transmits an empty *incomplete acknowledgement* and in response the transmitter re-transmits the whole packet again. In this case the extended header needs only to specify the chunk size c, the receiver can infer the number of chunks and the size of a slack chunk easily from knowledge of c and the total received packet size.

- ***Reduced scheme***: The receiver includes the identifications of the missing chunks into its incomplete-acknowledgement packet. One method to accomplish this is to use a bitmap. In its re-transmission the transmitter includes only the missing chunks. Again, it suffices for the transmitter to include the chunk size c into the extended header, since the receiver can at any time infer the number and identity of the chunks present in a packet from the total packet size and his own knowledge of missing chunks. The reduced scheme has the beneficial effect that the retransmission frame is much smaller, consumes less energy, produces less interference, is less likely hit by errors and reaches the receiver with smaller delay.

- ***Redundant scheme***: A possible drawback of the reduced scheme is that for increasing number of trials it (hopefully) produces shorter and shorter packets, as more and more of the chunks are successfully received. However, this also means that relatively more and more packet headers are transmitted as time progresses – the time budget is hence not used optimally. When the original packet consists of L chunks, the re-transmission packets again consist of L chunks, but these L chunks are used to repeat the yet unacknowledged chunks a number of times (the receiver again indicates the identification of the missing chunks in his incomplete acknowledgement). For example, when the initial packet consists of $L = 10$ different chunks, out of which the receiver receives all but chunks 4, 5 and 8, the retransmitted packet consists of four times a copy of chunk 4, and of three copies of chunks 5 and 8, respectively. In this scheme the transmitter must specify the chunk size c as well as the true number of different chunks in the packet. In the initial packet the number $L = 10$ would be transmitted, in the first retransmission the number 3 would be transmitted. This information, together with the additional rule that all outstanding chunks are replicated fairly until all chunks are exhausted, suffices to let the receiver figure out all necessary information.

Either way, the retransmissions continue until the receiver has all chunks in its buffer or until the deadline expires.

It will be shown that the intermediate checksum scheme provides significant benefit for channels with higher error rates. However, if the channel is extremely good, the larger header and all the extra checksums are likely wasted.

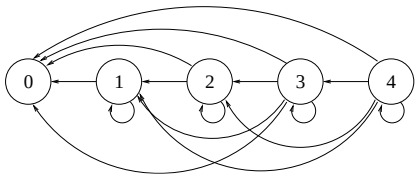
Chapter 3

Markov Model

In this section we develop a discrete-time Markov chain model [19, 23] that covers the complete scheme and the reduced scheme. The purpose is to compare send-and-wait protocols using the classical packet formats and the intermediate-checksum scheme with respect to the number of frames needed to transmit s user data bits, and with respect to the total number of bits needed to transmit the s user data bits (goodput ratio) in the reduced scheme. To ease analysis, we assume the BSC channel model with bit error probability p. We assume that the send-and-wait protocol is used. All used checksums are wide enough to provide a reasonable approximation to perfect checksums. Furthermore, for simplicity we assume a perfect back channel, i.e. acknowledgements get never lost.[1]

A discrete-time Markov chain is adequate, when we associate one trial to transmit a frame with a time slot of the Markov chain. The BSC assumption ensures that the memoryless property of Markov chains indeed applies and that the Markov chain is time-homogeneous.

As a state variable X_n, we take the number of yet unacknowledged chunks at the transmitter side after the n-th trial. When we foresee a maximum number of L chunks per packet, the state space is $0, \ldots, L$. It is helpful to create a sketch of the possible state transitions (in the following figure with $L = 4$):



We now determine the state transition probabilities. Because of the BSC assumption, a single chunk is erroneous with probability

$$Q_C = 1 - (1 - p)^{c+h}$$

and correct with probability

$$P_C = 1 - Q_C = (1 - p)^{c+h}$$

[1] Acknowledgements can easily be accommodated by increasing the header size o' according to the size of acknowledgement packets.

The header (including its checksum) is erroneous with probability

$$Q_H = 1 - (1-p)^{o'+h}$$

and correct with probability

$$P_H = 1 - Q_H = (1-p)^{o'+h}$$

The probability that the header is correct and that k out of M chunks in a frame are erroneous is given by (binomial distribution):

$$r(k,M) = P_H \cdot b(k;M,Q_C) = (1-p)^{o'+h} \cdot \binom{M}{k} \cdot \left(1-(1-p)^{c+h}\right)^k \cdot \left((1-p)^{c+h}\right)^{M-k}$$

(where $b(k;n,p) = \binom{n}{k} \cdot p^k \cdot q^{n-k}$ for $k \in \{0,\ldots,n\}$ is the probability mass function of the binomial distribution with parameters n and p).

For our example with $L = 4$ the transition matrix becomes

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ r(0,1) & 1- & 0 & 0 & 0 \\ r(0,2) & r(1,2) & 1- & 0 & 0 \\ r(0,3) & r(1,3) & r(2,3) & 1- & 0 \\ r(0,4) & r(1,4) & r(2,4) & r(3,4) & 1- \end{pmatrix} \tag{3.1}$$

where the first row corresponds to state 0, the second row to state 1, and the last row to state 4. Furthermore, the entries $1-$ are chosen such that the respective row sums up to one. For the i-th row ($i \in \{1,\ldots,L+1\}$ since matrix elements are counted starting from one) of the matrix $\mathbf{P}$ ($i \in \mathbb{N}$) the diagonal elements are given by:

$$\begin{aligned} 1 - \sum_{j=0}^{i-2} r(j,i-1) &= 1 - P_H \sum_{j=0}^{i-2} b(j;i-1,Q_C) = 1 - P_H \cdot (1 - b(i-1;i-1,Q_C)) \\ &= 1 - P_H \cdot \left(1 - Q_C^{i-1}\right) \end{aligned}$$

Please note that since $\mathbf{P}$ is a lower triangular matrix, the diagonal elements of $\mathbf{P}$ correspond to its Eigenvalues. Since all Eigenvalues are distinct, $\mathbf{P}$ is diagonalizable.

The protocol starts at initial state $\lambda = \mathbf{e}_L$ and terminates once the target state $\mathcal{A} = \{0\}$ has been reached.

For $p \in (0,1)$ the state 0 is an absorbing state, all other states are transient states. It can be easily checked that state 0 is reached with probability one from an arbitrary start state.[2] And indeed, the stochastic row vector $\pi = (1,0,0,\ldots,0)$ is the unique stationary distribution of $\mathbf{P}$, i.e. it is the only stochastic row vector satisfying

$$\pi = \pi \cdot \mathbf{P}$$

We finally compute an expression for the success probability when only n trials are allowed. This success probability is then given by:

$$[[P^n]]_{L+1,1}$$

[2] This follows from a straightforward application of a theorem on hitting probabilities [19, Theorem 1.3.2] for the state set $\mathcal{A} = \{0\}$.

Since $\mathbf{P}$ is diagonalizable, it is clear that the matrix power $\mathbf{P}^n$ can be written as follows:

$$\mathbf{P}^n = \mathbf{U} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & \lambda_1^n & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \lambda_2^n & 0 & \dots & 0 & 0 \\ \dots & & & & \dots & & \\ 0 & 0 & 0 & 0 & \dots & \lambda_{L-1}^n & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \lambda_L^n \end{pmatrix} \cdot \mathbf{U}^{-1}$$

for some invertible matrix $\mathbf{U}$ which has the Eivenvectors as its columns, and by setting $\lambda_i = 1 - P_H(1 - Q_C^i)$. As a result, we can represent the success probability as follows:

$$[[P^n]]_{L+1,1} = a_0 + a_1 \cdot \lambda_1^n + a_2 \cdot \lambda_2^n + \ldots + a_L \cdot \lambda_L^n$$

for some uniquely determined coefficients $a_0, \ldots, a_L$ that are still unknown. Since for each $i \in \{1, \ldots, L\}$ we have $0 < \lambda_i < 1$ and since furthermore for $n \to \infty$ the success probability becomes one, we have that:

$$1 = \lim_{n \to \infty} [[P^n]]_{L+1,1} = a_0$$

Since $\mathbf{P}^0 = \mathbf{I}$, we have that:

$$0 = [[P^0]]_{L+1,1} = a_0 + a_1 + \ldots + a_L$$

and considering that $a_0 = 1$ we readily have $a_1 + a_2 + \ldots + a_L = -1$. Now we consider the case of $n = 1$, for which $\mathbf{P}^n = \mathbf{P}$ and correspondingly $[[P]]_{L+1,1} = r(0, L) = P_H \cdot (1 - Q_C)^L$. From this we have:

$$\begin{aligned} P_H \cdot (1 - Q_C)^L &= a_0 + a_1 \cdot \lambda_1 + \ldots + a_L \cdot \lambda_L \\ &= 1 + a_1 \cdot (1 - P_H \cdot (1 - Q_C)) \\ &\quad + a_2 \cdot (1 - P_H \cdot (1 - Q_C^2)) \\ &\quad + \ldots \\ &\quad + a_L \cdot (1 - P_H \cdot (1 - Q_C^L)) \\ &= P_H \cdot \left(1 + \sum_{i=1}^{L} a_i \cdot Q_C^i\right) \end{aligned} \tag{3.2}$$

Expanding the left-hand-side of this equation and performing a comparison of coefficients of Q_C gives the solutions:

$$a_i = (-1)^i \cdot \binom{L}{i} \qquad i \in \{0, \ldots, L\}$$

Since the coefficients a_i are uniquely determined and since the proposed choice of the a_i is indeed a solution to Equation 3.2, we have found the right solution. Summarizing, the success probability for n allowed trials is given by:

$$\sigma(c, L, p, n) = \Pr[\text{Success}] = [[P^n]]_{L+1,1} = 1 + \sum_{i=1}^{L} (-1)^i \cdot \binom{L}{i} \cdot \left(1 - P_H(1 - Q_C^i)\right)^n \tag{3.3}$$

It should be noted, however, that this explicit expression is numerically much more unstable (especially for small values of p) than the equivalent computation $[[P^n]]_{L+1,1}$ based on matrix powers / iterations on the initial state distribution vector.

Chapter 4

The case of an unbounded number of retransmissions

4.1 Expected number of frames

We now attack the question of how many frames the intermediate checksum scheme (more specifically: the complete and the reduced scheme) needs on average to be successful, i.e. to reach the final state 0 (assuming an unlimited number of retransmissions). Referring to [19, Theorem 1.3.5][1] and using the abbreviation $k_i = k'_i$ to denote the average number of steps that is needed for the Markov chain to reach the final state 0 when the chain starts in state i, we can set up the following linear equations:

$$\begin{aligned}
k_0 &= 0 \\
k_1 &= 1 + p_{1,1} \cdot k_1 \\
&= 1 + (1 - r(0,1)) \cdot k_1 \\
k_2 &= 1 + p_{2,1} \cdot k_1 + p_{2,2} \cdot k_2 \\
&= 1 + r(1,2) \cdot k_1 + (1 - r(0,2) - r(1,2)) \cdot k_2 \\
k_n &= 1 + \sum_{i=1}^{n} p_{n,i} k_i \\
&= 1 + \sum_{i=1}^{n-1} r(i,n) \cdot k_i + \left(1 - \sum_{i=0}^{n-1} r(i,n)\right) \cdot k_n
\end{aligned}$$

[1] Be $\mathcal{I}$ the (finite or countably infinite) state set of a Markov chain $(X_n)_{n\in\mathbb{N}_0}$ and be $\mathcal{A} \subset \mathcal{I}$ with $\mathcal{A} \neq \emptyset$. The hitting time $H^{\mathcal{A}}$ is a random variable associated to $(X_n)_{n\in\mathbb{N}_0}$ that has range $\mathbb{N}_0 \cup \{\infty\}$ and which is defined as $H^{\mathcal{A}} = \inf\{n \geq 0 : X_n \in \mathcal{A}\}$. For starting state $X_0 = i$ the average hitting time is defined as $k_i^{\mathcal{A}} = E\left[H^{\mathcal{A}} | X_0 = i\right]$. Then [19, Theorem 1.3.5] asserts that the $k_i^{\mathcal{A}}$ are the minimal non-negative solution of the following system of equations:

$$k_i = \begin{cases} 0 & : \ i \in \mathcal{A} \\ 1 + \sum_{j \notin \mathcal{A}} p_{i,j} k_j & : \ i \notin \mathcal{A} \end{cases}$$

where the $p_{i,j}$ are the transition probabilities from state i to state j contained in the state transition matrix $\mathbf{P}$.

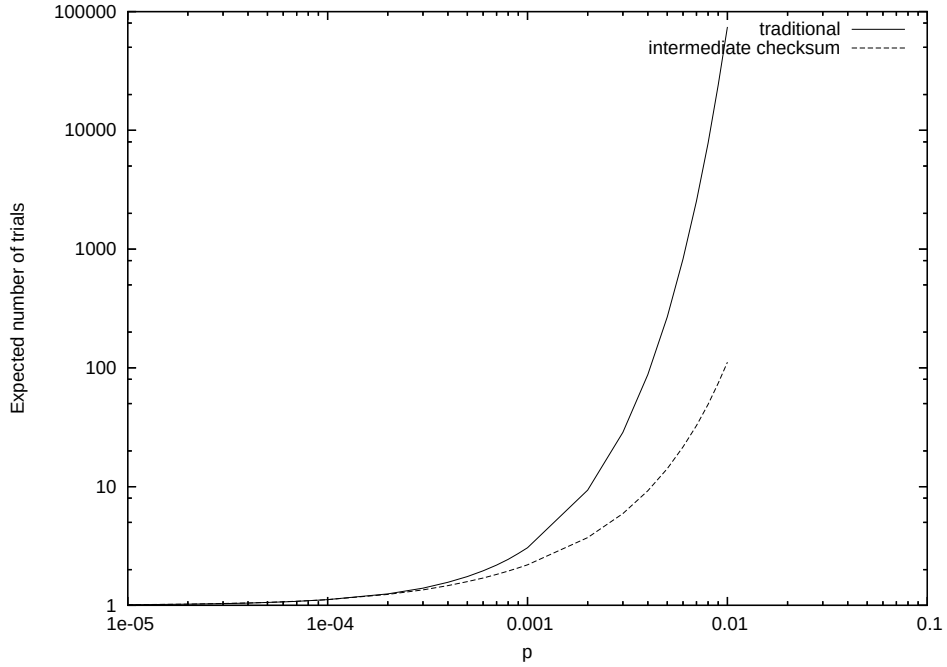


Figure 4.1: Average number of frames for the traditional scheme and the intermediate checksum scheme for varying bit error rate p

$$= \quad 1 + \sum_{i=1}^{n-1} r(i,n) \cdot k_i + (1 - P_H \cdot (1 - Q_C^n)) \cdot k_n$$

The complexity of direct expressions for k_n would quickly become infeasible. We therefore provide an iterative solution. We readily have $k_0 = 0$ and $k_1 = \frac{1}{r(0,1)}$. For general n we can solve for k_n as follows:

$$k_n = \frac{1 + \sum_{i=1}^{n-1} r(i,n) \cdot k_i}{\sum_{i=0}^{n-1} r(i,n)} \tag{4.1}$$

This last equation allows to compute the k_n from the previously computed values $k_{n-1}, \ldots, k_1$.

We discuss a numerical example. We compare the scheme with traditional packet format and the intermediate checksum scheme (complete and reduced scheme) for the average number of frames required until success. We fix $o = 100$ bits as the overhead of the traditional packet format, whereas $o' = 116$ bits is the overhead of the intermediate checksum scheme. Checksums are $h = 16$ bits long in both cases. We want to transmit $s = 1000$ bits of user data, which in the intermediate checksum case is split over $L = 4$ chunks of $c = 250$ bits size each. The expected number of trials for the intermediate checksum scheme is just k_4, whereas for the Send-And-Wait protocol the number of trials for the traditional scheme is a geometric random variable with expectation:

$$\frac{1}{(1-p)^{o+s+h}}$$

p	traditional scheme	intermediate checksum
1.0E-5	1.0112392	1.0119644
1.0E-4	1.1180886	1.1184505
5.0E-4	1.747466	1.5832658
0.001	3.054252	2.1931293
0.003	28.590128	5.9287066
0.005	268.81122	14.1644535
0.006	825.63873	21.569471
0.007	2538.9705	32.673874
0.01	74321.445	111.66118

Table 4.1: Average number of frames for the traditional scheme and the intermediate checksum scheme for varying bit error rate p

These two schemes are compared under the given parameters and for varying bit error rate p in Figure 4.1 (please note the logarithmic scale on both axes), and some numerical results are displayed in Table 4.1. Looking at these numbers shows that using intermediate checksums has no advantage for bit error rates below 10^{-4}, since the expected number of trials are almost identical, but intermediate checksum requires more overhead. However, for higher bit error rates the advantage becomes obvious.

However, the comparison given here is not entirely fair: protocols like IEEE 802.11 give users the opportunity to fragment the user data into a number of smaller packets [14], each one having its own header. However, it is shown in [26] that even when IEEE 802.11 is allowed to pick the optimal fragment size given knowledge of p, the intermediate checksum scheme is still significantly better, i.e. requires much fewer frames. The optimal number of user bits in such a fragment is for known p given by:

$$s_{opt}(p) = \frac{-(o+h)}{2} - \frac{1}{2\log(1-p)} \cdot \sqrt{(o+h)\log(1-p)\left((o+h)\log(1-p)-4\right)} \quad (4.2)$$

where $o \leq o'$ is the frame overhead (which without intermediate checksums can be smaller). This solution results from an optimization of the goodput of the classical framing scheme, which is given by

$$G(s) = \frac{s}{E\left[T_{HDTF}\right] \cdot (o+s+h)} = \frac{s(1-p)^{(o+s+h)}}{o+s+h} \quad (4.3)$$

where T_{HDTF} is a geometric random variable describing the number of trials that are needed to successfully transmit the frame. We could similarly for the intermediate checksum scheme use the chunk size that optimizes the per-chunk-goodput. This optimal chunk size is given by:

$$c_{opt}(p) = -\frac{h}{2} + \sqrt{\frac{h(h\log(1-p)-4)}{4\log(1-p)}} \quad (4.4)$$

For the sake of completeness we repeat the comparison already presented in [26] for the average number of frames required by the intermediate checksum scheme with optimum chunk sizes and the classical scheme with fragmentation and reassembly using the optimal fragment sizes. The average number of frames required until success versus the bit error rate is shown in Figure 4.2.

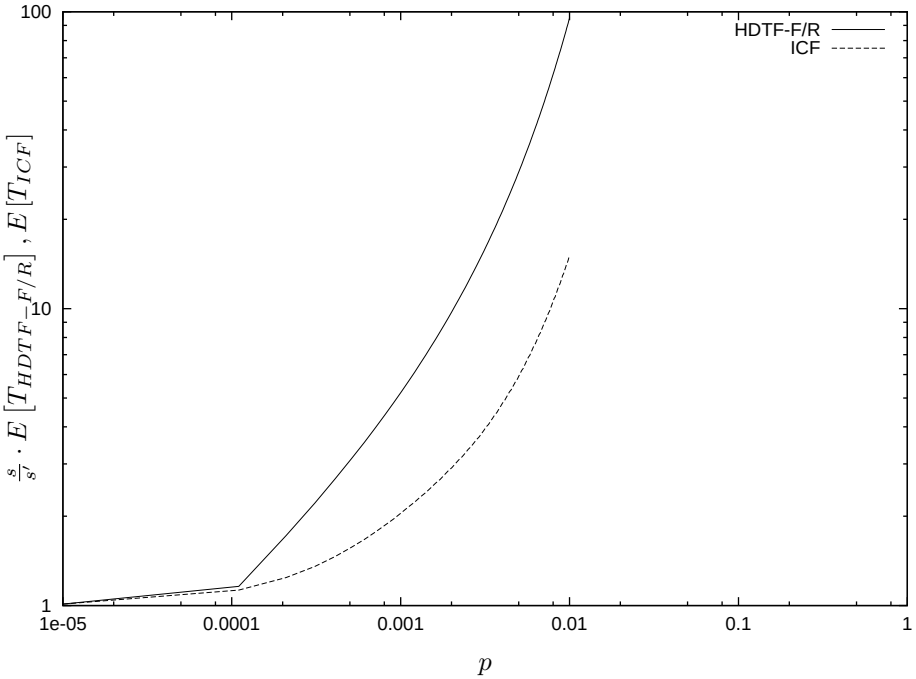


Figure 4.2: Expected number of frames needed to transmit 1000 bits of payload vs. the bit error rate p for the ICF and the classical scheme with fragmentation and reassembly, using optimum frame / chunk sizes

4.2 Expected number of bits

Next we turn our attention to another question: how many *bits* does the intermediate checksum scheme (specifically: the reduced scheme) require on average to successfully transmit s user data bits? Stated differently: what is the overhead of the intermediate checksum scheme?

We approach this question by resting on first-step analysis [24, Sec. 3.4]. In a nutshell, one conditions (by the law of total probability) on the state of the chain after one step, and from now on, by the Markov property, we can look at the chain $X_1, X_2, X_3, \ldots$ as being a new Markov chain with known start state. In the present case the approach goes as follows. Be $b_i = b_i^{\mathcal{A}}$ the average number of bits needed to reach state $\mathcal{A} = \{0\}$ when starting in state i. Suppose we start in state i. For the first transmission we need a number of $(o' + h) + i(c + h)$ bits. After this transmission we go with probability $p_{i,i}$ back into the same state i, and from here on we need (by the Markov property) another b_i bits on average. With probability $p_{i,i-1}$ we go into state $i - 1$ and from there on we need another b_{i-1} bits on average. By continuing this, we arrive at the following set of equations:

$$\begin{aligned}
b_0 &= 0 \\
b_1 &= (o' + h) + (c + h) + p_{1,1} \cdot b_1 \\
&= (o' + h) + (c + h) + (1 - r(0,1)) \cdot b_1 \\
b_2 &= (o' + h) + 2(c + h) + p_{2,1} \cdot b_1 + p_{2,2} \cdot b_2 \\
&= (o' + h) + 2(c + h) + r(1,2) \cdot b_1 + (1 - r(0,2) - r(1,2)) \cdot b_2 \\
&\ldots \\
b_n &= (o' + h) + n(c + h) + \sum_{i=1}^{n-1} r(i,n) \cdot b_i + \left(1 - \sum_{i=0}^{n-1} r(i,n)\right) \cdot b_n
\end{aligned}$$

From this set of equations, we can similar to the expected hitting times represent b_n in terms of previous values $b_0, \ldots, b_{n-1}$ and thus obtain a recursive scheme:

$$b_n = \frac{(o' + h) + n(c + h) + \sum_{i=1}^{n-1} r(i,n) \cdot b_i}{\sum_{i=0}^{n-1} r(i,n)} \tag{4.5}$$

For $L = 4$ and $c = 250$ the number b_4 is shown for different bit error rates p in Table 4.2. With respect to the average packet size it can be seen that significant savings in the number of transmitted bits can be achieved as compared to the complete scheme.

Finally, we fix $s = 1000$ and compare the average number of required bits for different values of L such that $L \cdot c = 1000$. The resulting curves for $L \in \{1, 2, 4, 8, 10, 20\}$ are shown in Figure 4.3. As could be expected, smaller chunk sizes are less efficient for smaller bit error rates but more efficient for higher bit error rates.

p	avg. number of packets	avg. number of bits	avg. packet size
1.0E-5	1.0119644	1201.8287	1187.6195
1.0E-4	1.1184505	1254.8448	1121.9493
5.0E-4	1.5832658	1507.342	952.04614
0.001	2.1931293	1873.9254	854.45276
0.003	5.9287066	4300.4053	725.353
0.005	14.1644535	9692.421	684.2778
0.006	21.569471	14519.536	673.15216
0.007	32.673874	21736.934	665.26953
0.01	111.66118	72833.24	652.27

Table 4.2: Average number of frames and average number of total transmitted bits for the intermediate checksum scheme versus bit error rate p

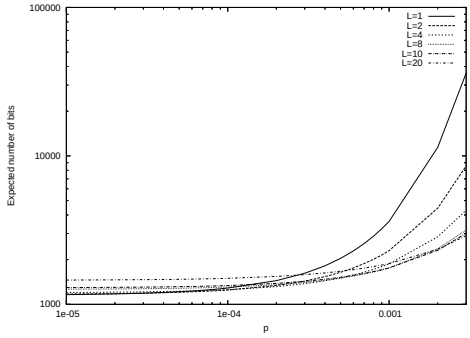


Figure 4.3: Average number of required bits for $s = 1000$ user data bits and different numbers L of chunks for varying bit error rate

Chapter 5

The case with a deadline

In real-time systems packets are usually equipped with a deadline and if this deadline expires without success, the packet is dropped. Suppose that the user data has a size of $s > 0$ bits and the time budget amounts to b bits, where b is expressed as $b = n_0 \cdot (o' + h + s + h)$ with $n_0 \in \mathbb{N}$ and the term in brackets being the size of the packet when only one single chunk (of size s) would be used. For simplicity, the time required for the acknowledgement packets is not explicitly tracked, it could for example be included into the header size o'.

The prime performance measure adopted is the *success probability*, i.e. the probability that the transmitter receives a final acknowledgement before the packet deadline expires.

5.1 The complete scheme

Given knowledge of the bit error rate p, one important goal is to find good values for L and c that maximize the probability that a packet can be successfully transmitted within its deadline. One possible approach is to re-formulate this goal, where for the time being integer constraints are disregarded. Specifically, when chunk size c is chosen:

- The total number of frames that in the complete scheme can be sent before the deadline is given by:

$$F(c) = n_0 \cdot \frac{o' + h + s + h}{o' + h + s + \frac{s}{c} \cdot h} =: n_0 \cdot \frac{m_1}{m_2 + \frac{s}{c} \cdot h} \tag{5.1}$$

  where the numerator corresponds to the size of the packet when the chunk size is chosen as s and the denominator corresponds to the result packet size for a chunk size of c.

- For one single packet, the average number of bits that are successfully received by the receiver is given by the product of the probability P_H of correctly receiving the header, times the average number of correctly received chunks times the size of a chunk:

$$\begin{aligned} P_H \cdot c \cdot \frac{s}{c} \cdot P_C &= (1-p)^{o'+h} \cdot c \cdot \frac{s}{c} \cdot (1-p)^{c+h} \\ &= (1-p)^{o'+h} \cdot s \cdot (1-p)^{c+h} \end{aligned}$$

Putting everything together, the worst-case average number of successful bits that can be transmitted within the given deadline is given by the product of these terms, i.e. by:

$$\eta(c,p) = \frac{n_0 \cdot m_1 \cdot s}{m_2 + \frac{s}{c} \cdot h} \cdot (1-p)^{o'+h} \cdot (1-p)^{c+h} \tag{5.2}$$

The unique value $c^* > 0$ which maximizes Equation 5.3 for known p is given by:

$$c^*(p) = \frac{-s \cdot h + \sqrt{s^2 \cdot h^2 - 4 \cdot m_2 \cdot \frac{s \cdot h}{\log(1-p)}}}{2 \cdot m_2} \tag{5.3}$$

Please note that Equation 5.3 gives different values for the optimal chunk size than Equation 4.4. Furthermore, this optimal value is independent of n_0 but not independent of s. For later evaluations (compare Chapter 6) we will also use the following modification of the function $\eta(\cdot,\cdot)$:

$$\eta^*(c,p) = \left\lfloor \frac{n_0 \cdot m_1 \cdot s}{m_2 + \frac{s}{c} \cdot h} \right\rfloor \cdot (1-p)^{o'+h} \cdot (1-p)^{c+h} \tag{5.4}$$

which considers the integer constraint on the number of available packets.

For the remainder of this paper we choose a more practical approach. We assume that the transmitter can choose one among a discrete number of chunk sizes. Specifically, we assume that $c \in \mathcal{C} = \{32, 64, 128, 256, 512, 1024\}$. For given p the transmitter chooses $c^*(p)$ as

$$c^*(p) = \arg\max_{c \in \mathcal{C}} \eta(c,p) \tag{5.5}$$

and the maximum number of allowed trials is then given by $n = \lfloor F(c^*(p)) \rfloor$. After computing the state transition matrix $\mathbf{P}$ for the chosen $c^*(p)$ and p (compare Equation 3.1), the success probability is given by the probability that after n steps the Markov chain has reached the final state 0, i.e. it is given by:

$$\Pr[\text{Success}|\,\text{Complete}] = [(0\ 0\ \ldots\ 0\ 1) \cdot \mathbf{P}^n]_1$$

In Figure 5.1 we present numerical results for $s = 1024$ bits and $n_0 = 10$ allowed trials. Specifically, we compare the success probability for the optimal $c^*(p)$ (Equation 5.5) with the success probabilities that can be obtained for each individual $c \in \mathcal{C}$. The chunk size of 1024 bits amounts to a raw send-and-wait version without using intermediate checksums. As can be expected, the complete scheme with $c^*(p)$ indeed provides the best delivery probabilities (together with $c = 32$), but there is a general trend that delivery probabilities improve with decreasing the chunk size. As opposed to the success probability, however, there is a difference between the complete scheme using $c^*(p)$ and the fixed one using $c = 32$. This difference is in the average number of bits that would be required until success (in case of an unbounded number of retransmissions, computed as k_L [compare Equation 4.1] times the resulting packet size for chunk size c, where $L = s/c$ is the number of chunks), shown for this example in Figure 5.2.

A limiting factor for the performance of all schemes is clearly the header size o', since this is common for all schemes and a header must be successfully received before the intermediate checksum scheme becomes effective.

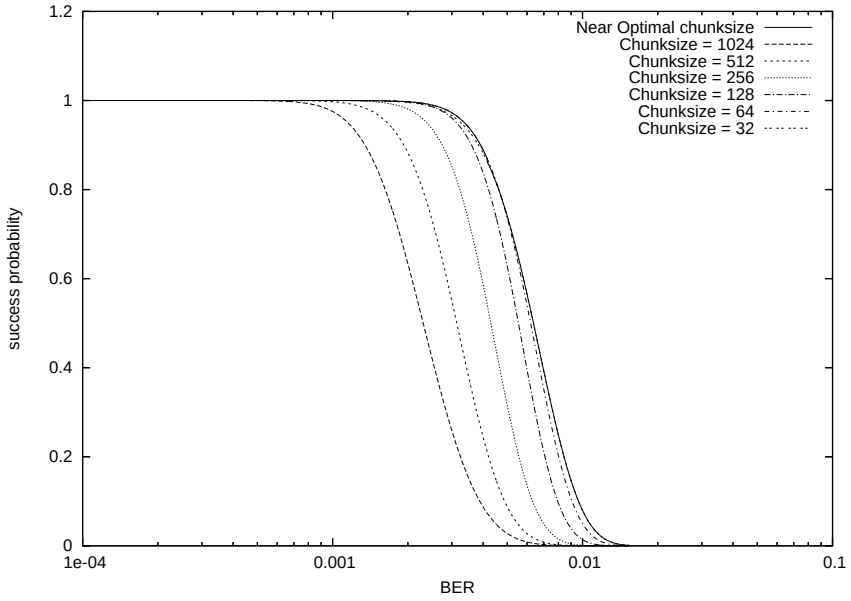


Figure 5.1: Success probabilities of different intermediate checksum schemes (four with fixed chunk sizes, plus the one with nearly optimal chunk size) versus bit error rate for $s = 1000$ user data bits and $n_0 = 10$ allowed trials

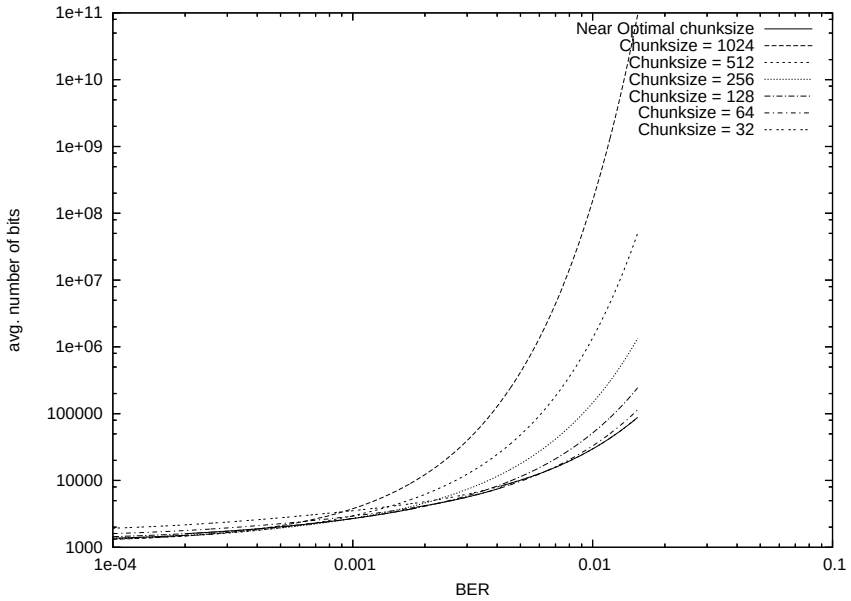


Figure 5.2: Success probabilities of different intermediate checksum schemes (four with fixed chunk sizes, plus the one with nearly optimal chunk size) versus bit error rate for $s = 1000$ user data bits and $n_0 = 10$ allowed trials

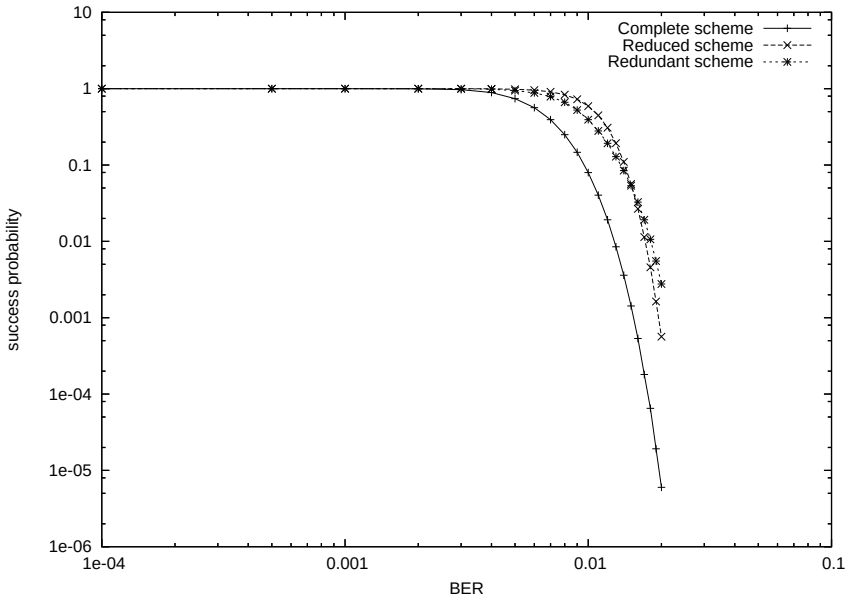


Figure 5.3: Success probabilities of different intermediate checksum schemes (complete, reduced and redundant) versus bit error rate for $s = 1024$ user data bits and $n_0 = 10$ allowed trials

5.2 The reduced and the redundant schemes

We now want to compare the reduced and the redundant scheme against the complete scheme. In all these three schemes we assume that for given bit error rate p always the optimal chunk size $c^*(p)$ according to Equation 5.5 is chosen.

The reduced scheme is not easily analyzed numerically, since the number of trials that can be made before the deadline expires is essentially random. The redundant scheme uses a fixed number of packets, but the probability that an outstanding chunk is correctly received also depends in a random fashion on time, since the redundancy grows randomly with subsequent numbers of trials. These schemes are therefore evaluated by simulation. More specifically, a custom simulator has been developed in the Common Lisp language [9, 7]. This simulator contains all three schemes (the complete scheme, the reduced and redundant scheme) such that much functionality is shared among these different schemes. The author has compared the simulation results for the complete scheme and the numerical results obtained in the previous Section 5.1 and they match very well. In the simulation results reported in the following, for each protocol scheme and each investigated bit error rate p a number of five million packet transactions has been simulated, leading to very tight confidence intervals that are not shown in the Figures.

In Figure 5.3 we compare the simulation results for the complete, the reduced and the redundant scheme for varying bit error rate p. The results show that both the reduced and the redundant scheme improve upon the complete scheme, but there is no clear winner among them: for values of $p > 0.015$ the redundant scheme is the best one, for smaller values the reduced scheme performs best. A possible explanation for this is that for very high bit error rates p the number of additional frames that the

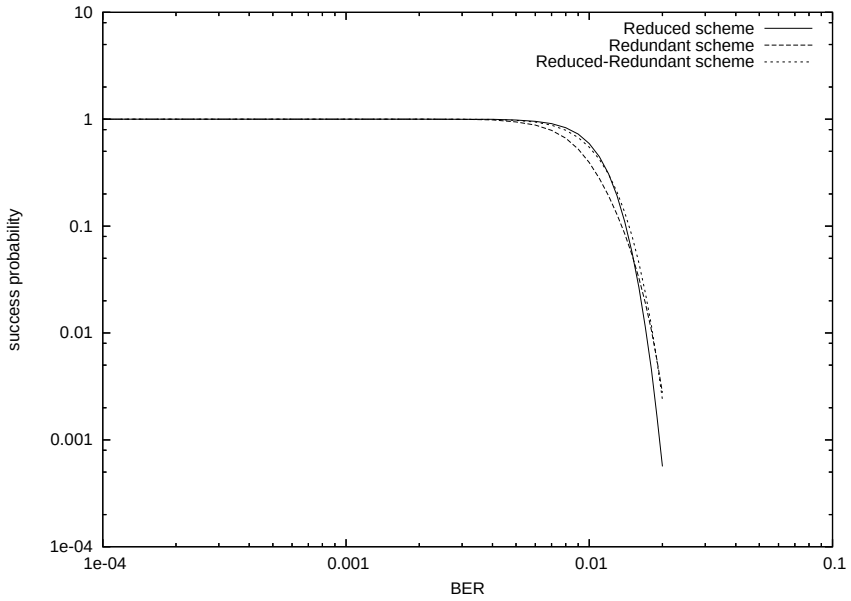


Figure 5.4: Success probabilities of different intermediate checksum schemes (reduced, redundant and reduced-redundant) versus bit error rate for $s = 1024$ user data bits and $n_0 = 10$ allowed trials

reduced scheme can transmit shrinks and becomes similar to the number of frames available to the redundant scheme. But in this case the redundant scheme has more potential to be successful since individual outstanding chunks have a higher probability to be successfully received.

Motivated by these findings, we consider a combined scheme, called ***reduced-redundant*** in which the number of replications of outstanding chunks is limited to two. In Figure 5.4 we compare the reduced scheme, the redundant scheme and the reduced-redundant scheme for their success probability. The results show that the reduced-redundant scheme is indeed a good mixture of the reduced and the redundant scheme, always very closely resembling the actual optimum of these two.

It is also instructive to compare the average bits that each of the four schemes (complete, reduced, redundant, reduced-redundant) spents per packet transaction, whether successful or not. The fewer bits spent on average, the earlier a packet's fate is known, and more bandwidth and time is available for other packets. In Figure 5.5 the corresponding simulation results are shown. Over a wide range of bit error rates the reduced scheme has the lowest average number of bits, while the complete scheme has always the highest average number of bits. The redundant scheme, while always using maximum sized frames, improves upon the complete scheme due to its higher success probability (and therefore its smaller average number of required frames). The reduced-redundant scheme is very close to the reduced scheme, it requires significantly fewer bits than the redundant scheme.

In summary, the reduced-redundant scheme behaves always very close-to-optimal both in terms of success probability and in terms of the average number of required bits.

Before closing, we want discuss a negative result concerning the reduced scheme. Specifically, there is a straightforward numerical approximation whichs actual quality we want to explore. The approximation works as follows:

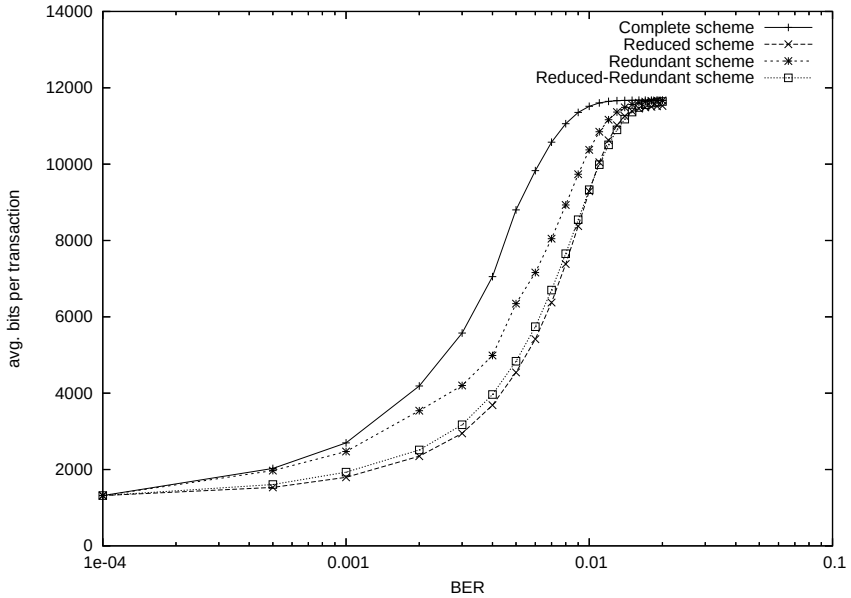


Figure 5.5: Average number of bits per transaction for different intermediate checksum schemes (complete, reduced, redundant and reduced-redundant) versus bit error rate for $s = 1024$ user data bits and $n_0 = 10$ allowed trials

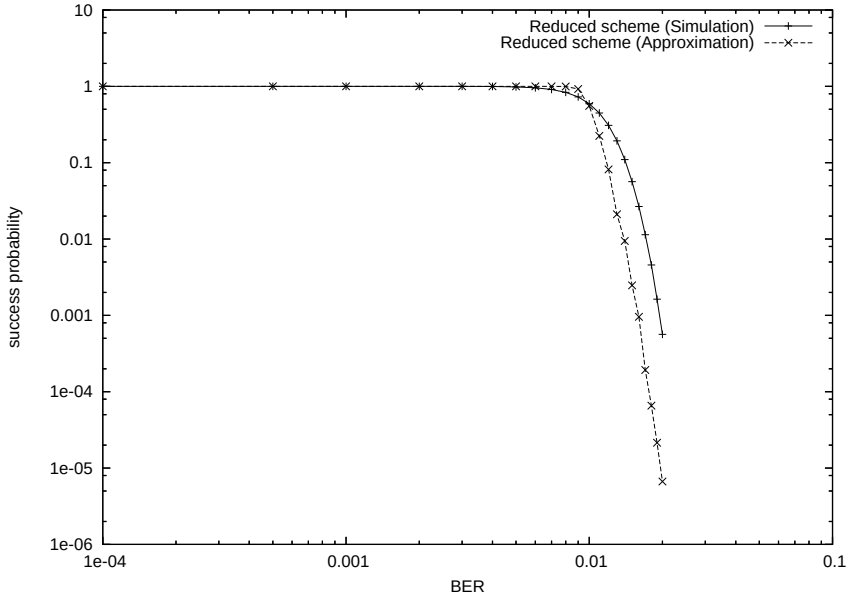


Figure 5.6: Reduced scheme: comparison of simulation and approximation versus bit error rate for $s = 1024$ user data bits and $n_0 = 10$ allowed trials

- Be $\lambda^{(0)} = (0\ 0\ \ldots\ 0\ 1)$ be the initial state probability vector of the underlying Markov chain, and let $\lambda^{(n)} = \lambda^{(0)} \cdot \mathbf{P}^n$ the state probability vector after n steps, and be

$$\widehat{c_n} = \lambda^{(n)} \cdot \begin{pmatrix} 0 \\ 1 \\ 2 \\ \ldots \\ L-1 \\ L \end{pmatrix}$$

  the average number of outstanding chunks after n steps, where $L = \lceil \frac{s}{c^*(p)} \rceil$ is the initial total number of chunks. Let furthermore be

$$\widehat{l_n} = o' + h + \lceil \widehat{c_n} \rceil \cdot (c^*(p) + h)$$

  be the average packet size after the n-th trial.

- Compute:

$$n^* = \max_{n \in \mathbb{N}_0} \left\{ \sum_{i=0}^{n} \widehat{l_n} \leq b \right\}$$

- To obtain the success probability, then the expression

$$\Pr\left[\text{Success} | \text{ Complete}\right] = \left[(0\ 0\ \ldots\ 0\ 1) \cdot \mathbf{P}^{n^*} \right]_1$$

  is computed.

In Figure 5.6 we compare the quality of this approximation for varying bit error rate. It can be concluded that the approximation is too coarse to be useful.

Chapter 6

Selection of Chunk Sizes

In this part we look at the problem of determining a finite set of chunk sizes from which the protocol can choose. In the previous Chapter 5 we have used the set $\mathcal{C} = \{32, 64, 128, 256, 512, 1024\}$ as our *chunk-size set* (CSS) without considering the question whether this is a good such set, i.e. the set giving the highest success probabilities or the lowest average numbers of required bits.

To address this question, we make a number of (simplifying) assumptions:

- The allowed chunk sizes are larger or equal to a certain minimum chunk size c_{min} and smaller than or equal to a certain maximum chunk size c_{max}.
- The bit error rate that the channel assumes is a random variable B with range $(0, 1)$ and distribution function $F_B(\cdot)$ with $F_B(0) = 0$ and $F_B(1) = 1$.
- The transmitter must encode the actual chunk size in this frame. If we allow this specification to consist of m bits, the CSS consists of 2^m distinct chunk sizes. Both transmitter and receiver possess a common codebook that maps the 2^m distinct values to actual chunk size values.
- We consider only the complete scheme since for this scheme we have a method to perform numerical calculations. For the time being we assume that a CSS that is good or even optimal for the complete scheme is also good or even optimal for the reduced, redundant or reduced-redundant schemes. It is a possible subject of future work to confirm or disprove this assumption.
- Instead of maximizing the average success probability (which in general depends on the bit error rate p in very complicated ways) we want to maximize the worst case average number of successful bits within the given deadline, i.e. we consider $\eta(c, p)$ instead of $\Pr[\text{Success}]$. It is reasonable to assume that maximizing $\eta(c, p)$ is also beneficial for the success probability.**[FIX!]**[1]

We can cast the identification of a good CSS in the framework of quantization and rate-distortion theory [8], [2, Chap. 10]. Roughly, the problem is to represent realizations of a continuous scalar random variable by one of a finite number of values. This finite set of values (sometimes called *reproduction points*) should be chosen such that an average distortion measure is minimized. Such

[1] **[AW]:** In fact, i believe that the success probability is a monotonic function of $\eta(c, p)$.

a distortion measure accounts for the difference between a realization of the random variable (a real number) and its associated representation. We want to minimize the average distortion measure for a fixed number of reproduction points (i.e. a fixed rate representation). By casting the problem in this framework it is possible to utilize known algorithms for finding reproduction points like for example Lloyd's Algorithm. Unfortunately, all these algorithms require that the probability distribution of the underlying random variables (bit error rate or chunk size) is known.

When B is the random variable for the current bit error rate, then the random variable giving the current optimal chunk size is given by a random variable $C = c^*(B)$ (compare Equation 5.3). From Equation 5.3, for the distribution function $F_C(c) = \Pr[C \leq c]$ we have:

$$F_C(c) = 1 - F_B\left(1 - \exp\left(\frac{-s \cdot h}{c \cdot s \cdot h + m_2 \cdot c^2}\right)\right)$$

In practical realizations the chunk sizes cannot be arbitrary, but they have lower and upper bounds c_{min} and smaller than or equal to a certain maximum chunk size c_{max}. Because of this we use the following modified distribution function:

$$G_C(c) = \begin{cases} 0 & : \quad c < c_{\text{min}} \\ F_C(c) & : \quad c_{\text{min}} \leq c \leq c_{\text{max}} \\ 1 & : \quad c > c_{\text{max}} \end{cases}$$

Now define by $\hat{c}(p)$ the function that associates one of the 2^m chunk sizes to a given bit error rate p. We consider two different distortion criteria:

- Squared-error in the chunk size: the distortion is measured by:

  $$d_2\left(c^*(p), \hat{c}(p)\right) = \left(c^*(p) - \hat{c}(p)\right)^2$$

  This distortion measure, however, has the problem that it is not sensitive to the achieved success probability: for very small bit error rate p a larger deviation in the actual chunk size $\hat{c}(p)$ from the optimal chunk size $c^*(p)$ is less critical than for larger values of p. This motivates the second distortion measure.

- A distortion measure that includes the success probability (compare Equation 3.3 can be defined as:

  $$d_s\left(c^*(p), \hat{c}(p)\right) = \left[\sigma\left(c^*(p), \left\lceil \frac{s}{c^*(p)} \right\rceil, p, \lfloor F(c^*(p)) \rfloor\right) - \sigma\left(\hat{c}(p), \left\lfloor \frac{s}{\hat{c}(p)} \right\rfloor, p, \lfloor F(\hat{c}(p)) \rfloor\right)\right]^2$$

In the remainder of this chapter we perform a numerical study to assess the difference between these two distortion measures. Specifically, we assume that the bit error rate is chosen as $B = 10^U$ where the exponent U is a random variable drawn uniformly from $[-6, -1]$. This means that:

$$F_B(p) = \Pr[B \leq p] = \Pr\left[10^U \leq p\right] = \Pr\left[U \leq \frac{\log p}{\log 10}\right] = F_U\left(\frac{\log p}{\log 10}\right)$$

For the case of the $d_2(\cdot, \cdot)$ distortion measure Lloyd's algorithm [20]**[FIX!]**[2] can be applied to find the CSS that minimizes the average distortion. Lloyd's algorithm is an iterative descent algorithm.

[2] **[AW]:**Give a precise citation

Listing 6.1: Lloyd's Algorithm for CSS determination based on $d_2(\cdot,\cdot)$

parameters: initial CSS of 2^m reproduction points from $[c_{\mathtt{min}}, c_{\mathtt{max}}]$

// initializations
set $\mathbf{Q}_0$ **to** initial CSS
set c_l **to** $c_{\mathtt{min}}$

// main loop
repeat

for i = 1 **to** $2^m - 1$
// determine upper end of current Voronoi region
set c_u **to** $G_C^{-1}\left(\frac{G_C([\mathbf{Q}_0]_i)+G_C([\mathbf{Q}_0]_{i+1})}{2}\right)$
// centroid of this region is new reproduction point
set $[\mathbf{Q}_1]_i$ **to** $G_C^{-1}\left(\frac{G_C(c_l)+G_C(c_u)}{2}\right)$
set c_l **to** c_u
endfor

// consider final Voronoi region
set $[\mathbf{Q}_1]_{2^m}$ **to** $G_C^{-1}\left(\frac{G_C(c_l)+G_C(c_{\mathtt{max}})}{2}\right)$

until $\|\mathbf{Q}_0 - \mathbf{Q}_1\|_1 \leq 1$

return $\mathbf{Q}_1$

Point	**Mean / StdDev** ($m = 2$)	**Mean / StdDev** ($m = 3$)	**Mean / StdDev** ($m = 4$)
1	14.8 / 0.01	10.0 / 0.01	8.2 / 0.02
2	57.3 / 0.09	21.6 / 0.03	12.6 / 0.07
3	188.5 / 0.29	42.44 / 0.07	18.8 / 0.16
4	585.4 / 0.36	79.24 / 0.12	27.2 / 0.29
5		143.38 / 0.17	38.4 / 0.48
6		254.45 / 0.20	53.5 / 0.74
7		446.17 / 0.18	73.3 / 1.07
8		776.72 / 0.09	99.4 / 1.48
9			133.5 / 1.96
10			178.0 / 2.48
11			235.6 / 3.00
12			310.1 / 3.43
13			406.4 / 3.68
14			530.6 / 3.55
15			690.8 / 2.84
16			897.9 / 1.25

Table 6.1: Averages (rounded) and standard deviations of results generated by 500 rounds of the Lloyd algorithm for 4, 8, and 16 reproduction points and the $d_2(\cdot, \cdot)$ distortion measure

Adapted to our setting the algorithm can be described as in Listing 6.1. Since this algorithm is only guaranteed to converge to a local optimum, we have run it five hundred times with randomly chosen initial CSS sets (specifically: all 2^m initial points are drawn uniformly from $[c_{\text{min}}, c_{\text{max}}]$). In Table 6.1 we show the results. It can be seen that the difference between neighbored reproduction points increases, the most reproduction points are small.

The $d_s(\cdot, \cdot)$ distortion measure is more complicated to handle, since the mapping $p \mapsto \sigma(\cdot)$ is not monotone and the distribution function of the success probabilities is hard to compute explicitly. Instead of applying Lloyd's algorithm we have applied a genetic algorithm to determine good CSS. Genetic algorithms are a well-known approach to find local extrema in large search spaces [6]. Our approach for identifying good CSS works as follows:

- The algorithm works on a *population* of individual CSS (each of length $n \in \{4, 8, 16\}$), having a fixed *population size*. The initial population includes the solution given in Table 6.1, a number of mutations of this solution, while the remaining members are random vectors chosen from $[c_{\text{min}}, c_{\text{max}}]^n$.
- Each CSS in the population is judged for its quality, measured as the average success probability over the given range of bit error rates $[10^{-6}, 10^{-1}]$ (see below). Judging all the members of a population is referred to as a *judging round*. The algorithm performs a limited number of judging rounds.
- At the end of a judging round a new population is created using the results available for the current population. The $\alpha \cdot 100\%$ of the best members of the current population are carried over into the new population. These are called *survivors*. The next $\beta \cdot 100\%$ of the members of

Algorithm	$m = 2$ (4 points)	$m = 3$ (8 points)	$m = 4$ (16 points)
plain ARQ	0.6721	0.6721	0.6721
IC with Lloyd	0.7584	0.7553	0.7567
IC with GA	0.7598	0.7594	0.7594

Table 6.2: Comparison of average success probabilities for 4, 8, and 16 reproduction points of a plain ARQ scheme without intermediate checksums, and of the intermediate checksum scheme configured with the results of Lloyd's algorithm ($d_2(\cdot,\cdot)$ measure) and of the genetic algorithm ($d_s(\cdot,\cdot)$ measure). Average is taken over 5000 equidistant BER exponents from [-6,-1].

Algorithm	$m = 2$ (4 points)	$m = 3$ (8 points)
IC with Lloyd	$\{15, 57, 188, 585\}$	$\{10, 22, 42, 79, 143, 254, 446, 777\}$
IC with GA	$\{32, 54, 883, 892\}$	$\{20, 32, 57, 70, 839, 908, 936, 948\}$

Table 6.3: Optimal CSS for 4 and 8 reproduction points of the intermediate checksum scheme configured with the results of Lloyd's algorithm ($d_2(\cdot,\cdot)$ measure) and of the genetic algorithm ($d_s(\cdot,\cdot)$ measure).

the new population are created from mutations of randomly chosen members of the $\alpha \cdot 100\%$ survivors. Specifically, to create a mutated member one survivor CSS $(c_1, \ldots, c_n)$ is picked randomly and each of its component is set with a certain probability to a new value drawn uniformly from $[c_{\text{min}}, c_{\text{max}}]$. The next $\gamma \cdot 100\%$ of the members of the new population are created from crossovers of the survivors. Specifically, two survivor chains $(c_1, \ldots, c_n)$ and $(c'_1, \ldots, c'_n)$ are picked randomly and a new chain is created as $(c_1, \ldots, c_{n/2}, c'_{n/2+1}, \ldots, c'_n)$. The remaining $(1-\alpha-\beta-\gamma)\cdot 100\%$ of the members are randomly chosen from $[c_{\text{min}}, c_{\text{max}}]^n$.

By including into the initial population the results from Table 6.1 (with each chunk size given there rounded to the next integer value), which are only based on c^* and not on the success probability, we ensure that the results found by the genetic algorithm are not worse in terms of average success probability. The average success probability has been obtained by sampling the range $[-6, -1]$ of BER exponents with 5000 equidistant points. For each resulting BER p the judging function determines for the given CSS $(c_1, c_2, \ldots, c_n)$ the actual chunk size c' as (compare Equation 5.4)

$$c'(p) = \underset{c \in \{c_1, c_2, \ldots, c_n\}}{\arg\max} \eta^*(c, p)$$

The population size has been chosen as 40 members, the number of rounds has been chosen as 30, and the further parameters have been chosen as $\alpha = 0.3$, $\beta = 0.2$, and $\gamma = 0.2$.

The results for the average success probabilities for the CSS generated by Lloyds algorithm for the $d_2(\cdot,\cdot)$ distortion measure and by the genetic algorithm for the $d_s(\cdot,\cdot)$ are compared in Table 6.2, and the resulting optimal CSS sets for CSS sizes of 4, 8 and 16 reproduction points are given in Tables 6.3 and 6.4. It can be noted that in terms of the average success probabilities the improvements of the CSS generated by the GA algorithm is only minor: from 75.84% average success probability to 75.98 for the case of four reproduction points, from 75.53% to 75.94% for eight reproduction points and from 75.67% to 75.94% for sixteen reproduction points. It is also interesting to note that for the

Algorithm	$m = 4$ (16 points)
IC with Lloyd	$\{8, 13, 19, 27, 38, 53, 73, 99, 134, 178, 236, 310, 406, 531, 691, 898\}$
IC with GA	$\{33, 37, 54, 304, 392, 432, 590, 601, 405, 432, 531, 539, 590, 680, 691, 984\}$

Table 6.4: Optimal CSS for 16 reproduction points of the intermediate checksum scheme configured with the results of Lloyd's algorithm ($d_2(\cdot,\cdot)$ measure) and of the genetic algorithm ($d_s(\cdot,\cdot)$ measure).

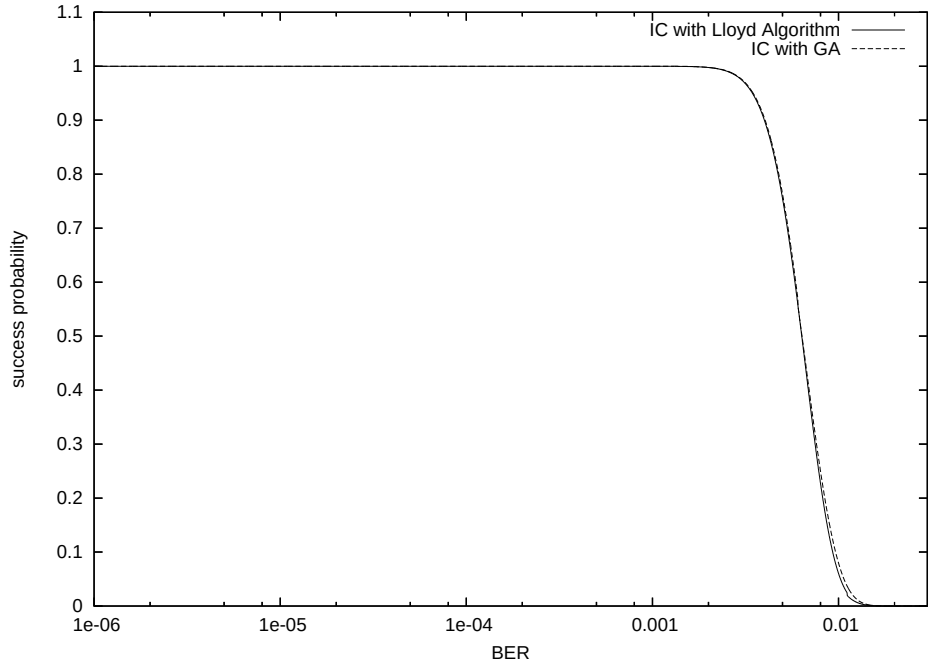


Figure 6.1: Success probability versus bit error rate for the CSS generated by Lloyds algorithm and by the genetic algorithm for a CSS size of four reproducation points.

GA the differences in performance between four, eight and sixteen reproduction points are negligible, while for the Lloyd algorithm they are (slightly) larger. However, in both cases the achieved success probability is significantly larger than in the case without using an intermediate checksum scheme. Looking at the chosen chunk sizes one can note the following:

- The minimum chunk sizes generated by the GA are at least twice the one generated by Lloyds algorithm, the smallest chunk sizes are not adopted.
- For four and eight reproduction points the chunk sizes generated by the GA appear to cluster at the lower end and at the higher end, the chunk sizes from Lloyd's algorithm are clustered only at the lower end.

Finally, we provide also a visual comparison of the achieved success probability over the range of bit error rates from 10^{-6} to 10^{-1}. In Figures 6.1, 6.2 and 6.3 it is confirmed that indeed the difference

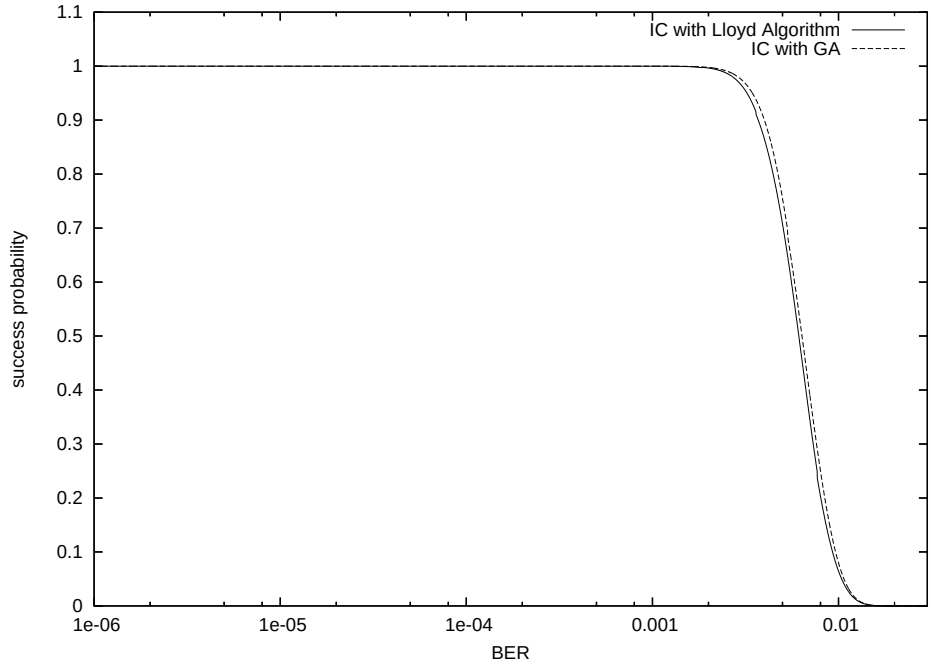


Figure 6.2: Success probability versus bit error rate for the CSS generated by Lloyds algorithm and by the genetic algorithm for a CSS size of eight reproduction points.

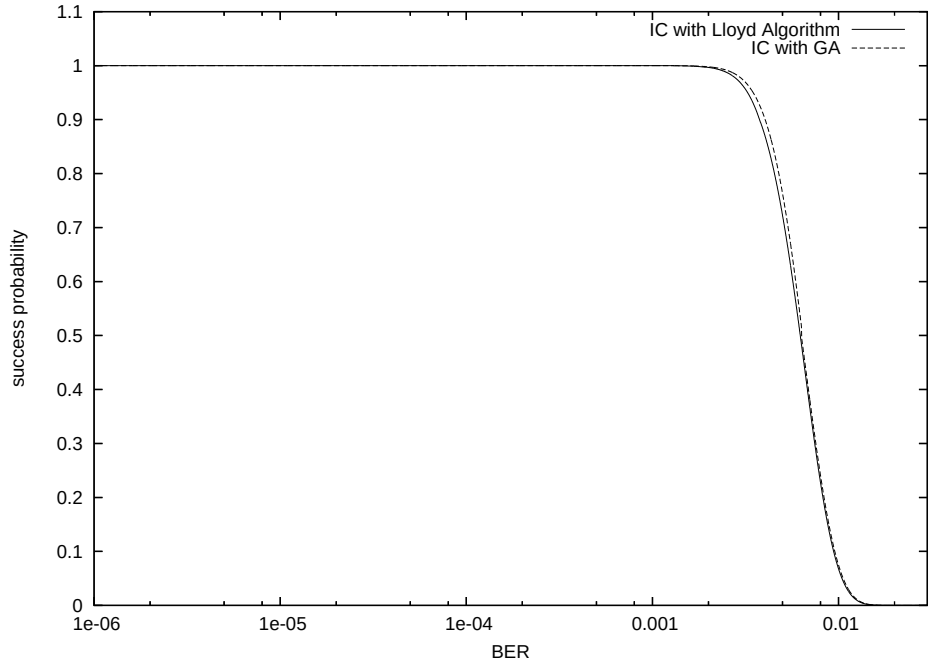


Figure 6.3: Success probability versus bit error rate for the CSS generated by Lloyds algorithm and by the genetic algorithm for a CSS size of sixteen reproduction points.

in success probability between the CSS generated by the Lloyd algorithm and the genetic algorithm are minor.

Chapter 7

Related work

In general, the intermediate checksum scheme considered here belongs is a packet combining scheme [22, 3] or type-III hybrid ARQ schemes [13].

At the time of writing the first publication [26] the intermediate checksum scheme was, to the best of the authors knowledge, not yet discussed in the literature, except from [16], where the approach is briefly sketched but not followed anymore. In [5] an approach very similar to our intermediate checksums has been designed, implemented and evaluated in the context of wireless sensor networks. They consider that the allowed frame size is in general smaller than the message size. The message is fragmented into small blocks, several of which can fit into a frame. The block size can be adapted according to channel feedback. The focus of the protocol is on efficiently streaming the blocks of a message such that one frame can at the same time contain retransmissions of earlier failed blocks and new blocks. They also suggest a cooperative version of the intermediate checksum scheme, something, which has also in other contexts been considered a very rewarding addition to link layer schemes [4, 18]. However, the work presented in [5] does not consider deadlines and protocol schemes (like the reduced and redundant schemes) that are actually designed with deadlines in mind, and this paper is to the best of the authors knowledge the first paper doing that.

The related issue of chosing optimal packet sizes has been considered a number of times in the literature, For example [21] and [11] consider the problem of determining good frame sizes on fading channels. The combination of adaptive frame length control and FEC is explored in [15].

Chapter 8

Conclusions

In this paper we adapted the intermediate checksum scheme to the case where packet deadlines are present. It is shown that the intermediate checksum scheme gives significant improvements in terms of the success probability as well as a significant reduction in the number of transmitted bits (in the reduced scheme) as compared to the traditional scheme using only a single checksum. We therefore consider the intermediate checksum scheme as being a very attractive design approach for applications in which larger packets must be transmitted subject to a deadline.

There is significant potential for future work. A first one considers the intermediate checksums themselves. In this paper we have assumed that they are perfect, but to achieve this they should be of a certain minimum size, for example 16 bit. It is attractive to use smaller checksums for the chunks in addition to a perfect checksum over the whole user data. In this case, however, the intermediate checksum scheme might accept frames that the perfect checksum detects as erroneous and as a result *all* chunks must be transmitted again because it is not clear which of them is wrong.

Secondly, one could consider a variation of the redundant scheme in which the outstanding scheme are not simply repeated multiple times but broken down into (redundant) smaller chunks.

Thirdly, both intuition as well as the formula given in Equation 3.3 point to the fact that the header error probability P_H should be made as small as possible, which can be done by reduction of the header size (possibly adopting some header compression technique) or by protecting the header with an error-correcting code.

Finally, throughout this paper we have assumed that the bit error rate p is known. In practice, it has to be estimated somehow, and without requiring additional functionalities from the physical layer the only data available for this is the previous history of the intermediate checksum scheme. It has been discussed in [26] how the BER can be estimated, but the influence of the estimation error needs to be assessed in future work. A fundamental problem here is that the intermediate checksum scheme tries to minimize the number of packets / chunks required, and this tends to reduce the amount of data that can be used in a BER estimation, and this in turn makes the estimation more error-prone.

Bibliography

[1] Chipcon. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*. Chipcon Products from Texas Instruments, 2004.

[2] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, second edition, 2006.

[3] Abdel-Ghant A. Dairaseh and Carl W. Baum. Methods for packet combining in HARQ systems over bursty channels. *MONET - Mobile Networks and Applications*, 2:213–224, October 1997.

[4] Henri Dubois-Ferriere, Deborah Estrin, and Martin Vetterli. Packet combining in sensor networks. In *Proc. 3rd Intl. Conf. on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, November 2005.

[5] Raghu K. Ganti, Praveen Jayachandran, Haiyun Luo, and Tarek F. Abdelzaher. Datalink streaming in wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems (ACM SenSys)*, pages 209–222, 2006.

[6] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Bonn, Germany, 1989.

[7] Paul Graham. *ANSI Common Lisp*. Prentice Hall, 1995.

[8] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Trans. on Information Theory*, 44(6):2325–2383, October 1998.

[9] Jr. Guy L. Steele. *Common LISP – The Language*. Digital Press, 2nd edition, 1990.

[10] David Haccoun and Samuel Pierre. Automatic repeat request. In Jerry D. Gibson, editor, *The Communications Handbook*, pages 181–198. CRC Press / IEEE Press, Boca Raton, Florida, 1996.

[11] Shinsuke Hara, Akira Ogino, Makoto Araki, Minoru Okada, and Norihiko Morinaga. Throughput Performance of SAW-ARQ Protocol with Adaptive Packet Length in Mobile Packet Data Transmission. *IEEE Trans. on Vehicular Technology*, 45(3):561–569, August 1996.

[12] Samir Kallel. Analysis of a type-II hybrid ARQ scheme with code combining. *IEEE Trans. on Communications*, 38(8):1133–1137, August 1990.

[13] Samir Kallel. Complementary punctured convolutional (cpc) codes and their applications. *IEEE Trans. on Communications*, 43(6):2005–2009, June 1995.

[14] LAN/MAN Standards Committee of the IEEE Computer Society. *Information technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.

[15] Paul Lettieri, Curt Schurgers, and Mani B. Srivastava. Adaptive link layer strategies for energy-efficient wireless networking. *Wireless Networks*, 5(5):339–355, November 1999.

[16] Paul Lettieri and Mani Srivastava. Adaptive frame length control for improving wireless link throughput, range and energy efficiency. In *Proc. INFOCOM 1998*, pages 564–571, San Francisco, CA, 1998. IEEE.

[17] 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.

[18] Allen Miu, Hari Balakrishnan, and Can Emre Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proc. 11th annual international conference on Mobile computing and networking 2005 (MobiCom)*, Cologne, Germany, August 2005.

[19] J. R. Norris. *Markov Chains*. Cambridge University Press, Cambridge, UK, 1997.

[20] John G. Proakis. *Digital Communications*. McGraw-Hill, Boston, MA, fourth edition, 2001. International edition.

[21] Chee Kheong Siew and David J. Goodman. Packet data transmission over mobile radio channels. *IEEE Trans. on Vehicular Technology*, 38(2):95–101, May 1989.

[22] Pradeep S. Sindhu. Retransmission Error Control with Memory. *IEEE Trans. on Communications*, 25(5):473–479, May 1977.

[23] Daniel W. Stroock. *An Introduction to Markov Processes*. Springer, Berlin, 2005.

[24] Howard M. Taylor and Samuel Karlin. *An Introduction to Stochastic Modeling*. Academic Press, San Diego, California, third edition, 1998.

[25] Xin Wang and Michael T. Orchard. On reducing the rate of retransmission in time-varying channels. *IEEE Trans. on Communications*, 51(6):900–910, June 2003.

[26] Andreas Willig. Intermediate Checksums for Improving Goodput over Error-Prone Links. In *Proc. IEEE Vehicular Technology Conference (VTC), Fall 04*, Los Angeles, CA, September 2004.