Intermediate Checksums for Improving Goodput over Error-Prone Links

Andreas Willig

Communication Systems Group
Hasso-Plattner-Institute, University of Potsdam
Prof.Dr-Helmert-Strasse 2-3
14482 Potsdam, Germany
Fon: +49 331 5509216, Fax: +49 331 5509229
email: willig@hpi.uni-potsdam.de, awillig@ieee.org

***Abstract*— When a frame transmitted over an error-prone link is hit by bit errors, it is retransmitted entirely, despite the fact that often only a few bits are erroneous. In this paper we propose to not use a single checksum covering all data bits, but multiple checksums covering only parts of a frames data block (a *chunk*). In case of a retransmission only the chunks with a wrong checksum are retransmitted. We show analytically that for high bit error rates and a time-invariant binary symmetric channel this approach can give significant advantages in terms of goodput over the traditional scheme using only a single checksum per frame. In addition, we propose a simple scheme for adapting the chunk size to the current channel conditions and present first results for this, indicating that for bad channel conditions close-to-optimal goodput is achieved.**

I. Introduction

Almost all Automatic Repeat reQuest (ARQ) protocols [1], [2] 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. This kind of protocols is also used in wireless LANs (e.g. in the IEEE 802.11 WLAN standard [3]), which sometimes have to cope with high bit error rates.

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* (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 [4]. The rough idea is as follows: if there are s bits of user data, protocols with conventional header/data/trailer framing (HDTF) schemes put a header of o bits in front of the user data, and a trailer of h bits behind the user data. The header might carry source and destination address, frame length information and further control information, while the trailer consists of the frames checksum. Thus, the overall frame has size $o + s + h$. In case of a retransmission all of these $o + s + h$ bits have to be transmitted again. In contrast, in the ICF scheme the s user data bits are partitioned into L *chunks*, each having a raw size of c bits, to which a checksum of h bits is appended (we do not consider slack chunks for the moment). 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 $o' + L \cdot (c + h) > o + s + h$ bits. The receiver behaves as follows: if it detects an error in the frame header (which has a separate checksum), the 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 all chunks are correct, the receiver delivers the frame to its upper layers and sends a *final acknowledgement*. If not all chunks are correct, the faulty chunks are indicated to the transmitter with an *incomplete acknowledgement*. The transmitter retransmits only the faulty chunks. For example: if the first frame has $L = 8$ chunks and the receiver receives five out of eight, it requests the missing three chunks. This 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. However, the intermediate checksums impose a higher overhead, which may eat up any gains in goodput for small bit error rates.

In this paper we describe the ICF scheme in some more detail and compare it to the HDTF schemes applied in the IEEE 802.11 wireless LAN (full frames, fragmented frames) with respect to the maximum achievable goodput under a simple time-invariant binary symmetric channel. It shows up that despite its increased overhead ICF can reach a significantly higher goodput for high bit error rates (which occur frequently in wireless LAN environments) than what could be achieved with any of the IEEE 802.11 HDTF schemes. On the other hand, under good / ideal channel conditions the overhead of the ICF scheme leads to a smaller goodput than that achievable with HDTF. Therefore, we introduce a simple scheme for adapting the chunksize c to the current channel conditions and show its ability to adapt to different channel conditions. We note that the notion of goodput refers to the framing overhead needed to transmit one bit of user data. Other error source like collisions are not taken into account. Additionally, we note that maximizing goodput is equivalent to minimizing the energy expenditure.

The paper is structured as follows: in the following Section

II we describe the system model used in this paper and provide a more detailed description of the operation of both the HDTF and ICF schemes. In Section III we compare the goodput of both schemes analytically for a channel with independent bit errors and constant bit error rate. Following this, we present in Section IV a simple scheme for adapting the chunk size to the current channel conditions, and show its performance for simple examples. After briefly reviewing related work in Section V, the paper is concluded in Section VI.

II. System Model

For the purposes of this paper we consider a simple system, consisting of two stations called *transmitter* and *receiver*, and a communications channel in between them. The transmitter consists of a *data source*, generating blocks of user data having s bits size, and a protocol engine, which implements the send-and-wait protocol [2] along with either the ICF or HDTF scheme. The receiver generates acknowledgement frames and delivers the received data in-sequence to its upper layers.[1]

The precise operation of the HDTF scheme used in this paper is as follows: a block of s bits of user data is transformed into a frame by prepending a header of o bits and appending a trailer of h bits. For example, in generic frames of the IEEE 802.11 wireless LAN [3], [5] the checksum field has a width of $h = 32$ bits and a MAC header of $o = 240$ bits (without considering the header of the physical layer). We consider two different modes for HDTF, similar to what is supported in IEEE 802.11: HDTF with and without fragmentation. In the mode without fragmentation the s bits of user data are encapsulated into a single frame and the transmitter performs retransmissions until the receiver sends an immediate acknowledgement (which is a positive acknowledgement / final acknowledgement) or the maximum number of trials has been exhausted. However, for the purposes of this paper we assume the maximum number of trials to be unlimited. In the mode with fragmentation (henceforth referred to as HDTF-F/R) large frames are fragmented into a number of smaller frames (according to a predefined *fragmentation threshold*), to improve the probability of successful delivery of a fragment. Each fragment has a full header of o bits and is transmitted until it is successfully received or the maximum number of retransmissions is exhausted. After the receiver has successfully received all fragments, these are reassembled and the resulting data block is delivered to the upper layers.

Before presenting the ICF scheme in some more detail, we discuss the frame headers for the HDTF and ICF schemes. We distinguish between the *common header* and the *extended header*. The common header contains all those fields which both the HDTF scheme and the ICF scheme have in common, e.g. source and destination address, control fields etc. Specifically, we assume that the common header contains a *sequence number* or `seqno` field which helps in distinguishing new frames from retransmitted frames. Furthermore, the common header contains a *frame length* or `fl` field, which indicates the overall frame length in bits.[2] The extended header contains two different fields: the *chunk size* or `cs` field indicates the size of a full chunk in bits (or larger units like bytes, words, etc.), excluding the chunk checksum, while the *header checksum* or `hcs` field contains a checksum covering both the common and extended headers, but none of the chunks in the frames data part. The receiver can infer the number of chunks contained in a frame and the size of slack chunks from both the `fl` and `cs` fields.

The ICF scheme works as follows: after accepting a request of size s bits from the upper layers, the transmitter fragments this into chunks of size c (stored in the `cs` field) and an additional *slack chunk*. For each chunk a checksum is computed and appended to the chunk. The initial frame is constructed from all chunks. Let us first assume that the receiver receives a frame with a correct header checksum `hcs` and a new sequence number. The receiver can infer the total number of chunks needed for the user request from the initial frame and allocates a number of *chunk-buffers* accordingly. The receiver checks all chunks from the initial frame separately and copies all those with a correct checksum into the respective chunk-buffer. If some chunks are missing, the receiver sends an *incomplete ack* frame. This contains basically a bitmap where each bit indicates the state of the respective chunk-buffer: a zero bit indicates that the corresponding chunk is still missing, while a one bit indicates that the receiver received the chunk correctly. If all chunks are received correctly, the receiver sends a *final ack* frame. If the receiver receives a retransmitted frame (i.e. it receives a frame with an already seen sequence number), it copies the correct chunks to their respective chunk-buffer and generates the appropriate acknowledgement. If the transmitter receives a final ack, it may start to process the next user request (with a new sequence number). In case of an incomplete ack, the transmitter prepares a retransmission frame containing only the missing chunks. If the receiver fails to acquire bit synchronization or if the header checksum `hcs` is wrong, the receiver keeps quiet. Therefore, the transmitter needs mechanisms to detect the absence of an acknowledgement frame, e.g. based on timers. In this case the last transmitted frame (either initial or retransmitted frame) is repeated completely.

For the checksums we assume that both the HDTF scheme and the ICF scheme use the same kind of checksums.[3] Typical checksum algorithms are parity bits, Cyclic Redundancy Checks (CRC)'s [6], [7] or the Fletcher checksum used in TCP. In the IEEE 802.11 WLAN standard CRC's are used. For the purposes of this paper we do not require any specific kind of checksum, however, for simplicity we assume the checksum to be perfect, i.e. the residual error probability is negligible.

Although not necessary for the protocol, we assume hence-

[1]We note that nothing in the proposed ICF scheme prevents using more than two stations, and furthermore ICF is not tied to send-and-wait. Instead, it can also be integrated with the Goback-N or Selective-Repeat ARQ protocols.

[2]This information can also be part of the physical layer header, like in the IEEE 802.11 WLAN with DSSS PHY, or can be inferred from gaps surrounding a frame.

[3]An interesting aspect would be to shorten the checksum in the ICF scheme if the chunk size is small.

forth that the channel has zero propagation delay, which is a reasonable assumption for wireless LAN environments. Furthermore, in the following analytical and simulation-based evaluations for simplicity we assume that only bit errors happen, packet losses due to failure to acquire bit synchronization is not considered.[4]

III. Analytical Evaluation for a Time-Homogeneous Binary Symmetric Channel

In this section we compare the ICF and the HDTF schemes with respect to the achievable goodput and the required number of packets, other performance measures are not considered. For analytical tractability, we restrict to the simple case of a time-homogeneous binary symmetric channel (BSC) with a fixed bit error probability p (i.e. all bit errors occur independently and with the same bit error rate). For the HDTF scheme with s bits of user data, o bits of header and h bits of trailer / checksum, the frame error probability P is given by:

$$P = 1 - (1-p)^{o+s+h}$$

The number T_{HDTF} of necessary frame transmissions until successful reception is a geometric random variable with parameter P, and:

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

The goodput for the HDTF scheme can be characterized as the ratio of the size of the user data and the expected number of bits needed to successfully deliver them:

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

It is straightforward to find the optimal size s of user data which maximizes the goodput $G(s)$ for a given bit error rate p:

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

It is a little bit more complex to obtain the goodput of the ICF scheme. The basic tool for this is the development of a time-homogeneous discrete time Markov chain (TH-DTMC) describing the evolution of the transmission of s bits of user data. We assume an overhead of $o' > o$ bits, $L \geq 1$ chunks, furthermore each chunk consists of c bits user data bits plus h bits for the chunk checksum. The overall frame size is thus $o' + L \cdot (c+h) \geq o + s + h$ bits (if $s = L \cdot c$). For a single user request, we introduce the state variable X_n as the number of unacknowledged chunks after the n-th frame transmission, clearly we have $X_0 = L$. The receiver discards a frame completely, if the header checksum is incorrect, which happens with probability $1 - (1-p)^{o'}$. The probability that a single chunk is erroneous is given by $1-(1-p)^{c+h}$. The probability, that the receiver receives the header correctly while k out of M chunks are received in error is given by:

$$\begin{aligned} &r(k, M; p, c, h, o') \\ &\quad = (1-p)^{o'} \cdot \binom{M}{k} \cdot \left(1 - (1-p)^{c+h}\right)^k \left((1-p)^{c+h}\right)^{M-k} \end{aligned}$$

which is due to the independence assumption for the bit errors and the binomial distribution. We give the state transition matrix governing the process $(X_n)_{n\in\mathbb{N}_0}$ for the special case of $L = 4$, the extension to a general L is quite straightforward:

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

where the first row corresponds to state 0, the second row to state 1, and the last row to state 4, and the constant parameters have been suppressed from $r(k, M; p, c, h, o')$. Furthermore, the entries $1-$ are chosen such that the respective row sums up to one. We will denote the entry in the i-th row and j-th column of $\mathbf{P}$ as $[[\mathbf{P}]]_{i,j} = p_{i,j}$, and the entry in the i-th row and j-th column of the matrix power $\mathbf{P}^n$ is denoted as $[[\mathbf{P}^n]]_{i,j} = p_{i,j}^{(n)}$. In general, $\mathbf{P}$ is an $(L+1)\times(L+1)$ stochastic matrix.

The first interesting performance measure is the expected number of frames $E\left[T_{ICF}\right]$ needed by the ICF scheme to successfully deliver $s = c \cdot L$ bits of payload. It is convenient to use the survivor representation of a discrete random variable N with range $\mathbb{N}_0$ [9, p.181]:

$$E[N] = \sum_{i=1}^{\infty} \Pr[N \geq i]$$

In our case we can write:

$$\begin{aligned} E\left[T_{ICF}\right] &= \\ &\quad \Pr\left[\text{one or more steps are needed}\right] \\ &\quad + \Pr\left[\text{two or more steps are needed}\right] \\ &\quad + \Pr\left[\text{three or more steps are needed}\right] \\ &\quad + \Pr\left[\text{four or more steps are needed}\right] \\ &\quad + \ldots \\ &= 1 + \sum_{i=2}^{\infty}\left(1 - \Pr\left[X_i = 0 \,|X_0 = L\right]\right) \\ &= 1 + \sum_{i=2}^{\infty}\left(1 - [[P^i]]_{L+1,1}\right) \end{aligned}$$

As a first showcase, we compare $E\left[T_{ICF}\right]$ and $E\left[T_{HDTF}\right]$ for transmission of $s = 1000$ bits of user data. We have used $h = 16$ bits for each checksum, the common header is of $o = 100$ bits length, while the common and extended header sum up to $o' = 116$ bits. For the ICF scheme we have used fixed values of $c = 250$ and $L = 4$. For the HDTF scheme we have assumed the variant without segmentation and reassembly. The results for varying bit error rate p are

[4] In [8] it is shown that frame losses due to lack of bit synchronization are a major source of frame errors.

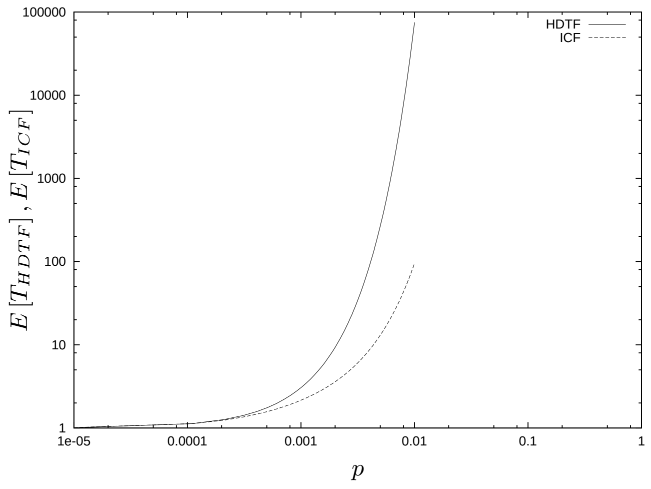

Fig. 1. Expected number of frames needed to transmit 1000 bits of payload vs. the bit error rate p, the HDTF scheme uses a single frame with $s = 1000$, the ICF scheme uses $c = 250$, $L = 4$

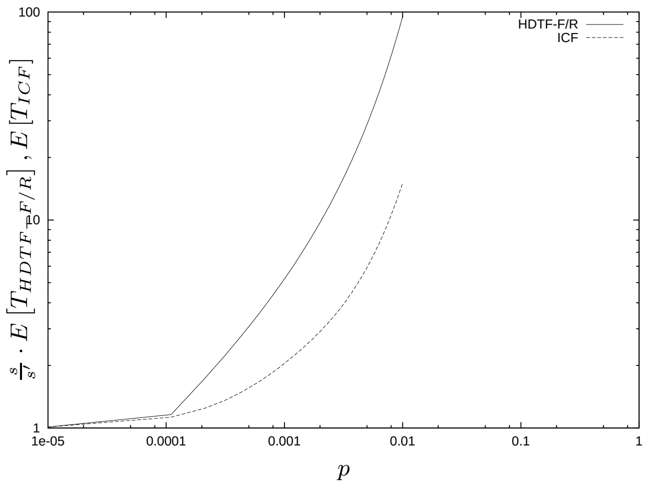

Fig. 2. Expected number of frames needed to transmit 1000 bits of payload vs. the bit error rate p for the ICF and the HDTF-F/R schemes, using optimum frame / chunk sizes

shown in Figure 1 as a log-log-plot. It can be seen that for increasing p we have $E[T_{HDTF}] \gg E[T_{ICF}]$ and the order of magnitude of the difference increases with increasing p. As an example, for $p = 0.004$ we have $E[T_{HDTF}] \approx 88.6$ and $E[T_{ICF}] \approx 8.7$. Therefore, if the transmitter has no chance to select a good frame size for data transmission and the channel has high error rates, the intermediate checksum approach can give dramatic improvements in transmission efficiency.

In the second step, we compare the ICF scheme with the HDTF scheme with fragmentation and reassembly (see Section II), termed HDTF-F/R. The best use of HDTF-F/R is made if the size of a frames data part is chosen according to Equation 2, since this choice maximizes the goodput by minimizing the frame retransmission probability. Again, both schemes are faced to the task of transmitting $s = 1000$ bits of user data over a channel with bit error rate p. The HDTF scheme is allowed to pick the optimum frame size $s' = \min\{s_{opt}(p), 1000\}$ and for this frame size $E[T_{HDTF}]$ is computed and multiplied with the needed number of frames $\frac{s}{s'}$ (ignoring rounding to the next integral number). As for the ICF scheme, we have to decide about the optimal chunk size, which can be shown to be given by:

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

This chunk size optimizes the goodput of a single chunk:

$$H(c) = \frac{c \cdot (1-p)^{c+h}}{c+h} \tag{5}$$

With this relationship, L is determined as:

$$L = \left\lceil \frac{1000}{\min\{c_{opt}(p), 1000\}} \right\rceil$$

and c as $\frac{1000}{L}$, again ignoring the need to round to the next integer. The result is shown in Figure 2. It can be seen that even if the HDTF-F/R scheme is allowed to pick the optimum frame size, the ICF scheme needs significantly fewer frames for error rates $p \geq 10^{-4}$.

As a third comparison, we present results for the optimum achievable goodput for the ICF and HDTF schemes for varying bit error rate p. For the HDTF scheme, the optimal goodput is directly determined from Equation 1 picking the optimal frame size determined from Equation 2. The other parameters are $o = 100$ bits and $h = 16$ bits. For the ICF scheme the optimal goodput is (at least in principle) obtained by counting for each possible execution the number of bits spent for it, and summing over all possible executions, weighing each execution with its probability. An efficient computation procedure was developed, however, the details are omitted due to lack of space. The computations were carried out using $o' = 116$ and $h = 16$ bits, and the chunk size is determined from Equation 4. In Figure 3 we compare the achievable goodput for both schemes for bit error rates between $p = 10^{-5}$ and $p = 10^{-2}$ and for different values of L, thus making the initial frames in the ICF scheme larger. The following points are remarkable:

- The ICF scheme gives significantly better optimal goodput than the HDTF scheme for error rates $\geq 10^{-4}$
- The goodput of the intermediate checksum scheme tends to increase with L, but the advantage becomes smaller as L becomes larger.
- The ICF scheme has its biggest advantages for $p \approx 2 \cdot 10^{-3} \ldots 3 \cdot 10^{-3}$ (for $p = 0.002$ we have: goodput of HDTF ≈ 0.335, goodput of ICF ≈ 0.5) while for even larger p the curves move closer together. Accordingly, we can observe in Figure 2 that for larger p the curve for the ICF scheme has a larger slope than the curve for the HDTF scheme. An explanation for this is that we have used $o = 100$ and $o' = 116$, which was done to account for the extension header needed by the ICF scheme. Therefore, as p becomes larger, the additional retransmissions caused by the larger header tend to eat

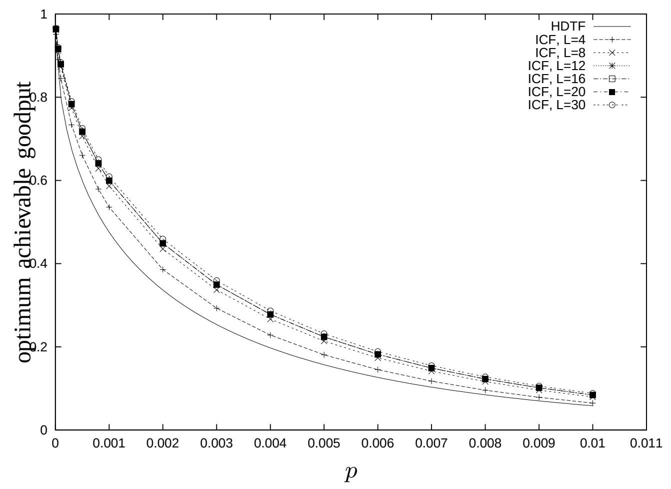

Fig. 3. Comparison of the optimum achievable goodput for the HDTF scheme (with $s = s_{opt}(p)$) and the ICF scheme for different L and $c = c_{opt}(p)$

up the gains of the ICF scheme.

IV. Making the Scheme Adaptive

In the previous section we made the unrealistic assumptions that: a) the channel bit error rate is constant; and b) it is known to the transmitter, which can choose the optimum chunk or frame size. In practice, however, neither of these conditions is true. We could always choose a small chunk size. This may give close-to-optimal goodput for high bit error rates, but for small rates significant bandwidth is wasted. In this section we propose a simple adaptation scheme, which gives almost the optimum goodput for high error rates, reasonable goodput for low error rates and which uses only information readily available; no additional information from the hardware (like a received signal strength indicator) is used.

Our starting point are Equations 5 and 4, which allow to find the optimum chunk size for the instantaneous bit error rate p. Therefore, the transmitter has to estimate p. To do this, it keeps for a given chunksize c' a *history* of *outcomes* (t_i, L', r_i, w_i), where an outcome belongs to a single user request to transmit s bits of user data. The history is required to have a certain depth for performing the adjustment algorithm. This depth is either specified by a fixed number n of entries, or by restricting the entries to belong to a time window. For simplicity, we choose the first approach. For outcome i, t_i denotes the time instant where the final ack is received, L' denotes the nominal number of chunks ($L' = s \div c'$), $r_i \geq L'$ denotes the number of actually transmitted chunks, and w_i denotes the number of actually transmitted slack chunks, which occur if $L' \mod c' \neq 0$. The proposed scheme assumes that channel errors occur independently with a constant rate during the time spanned by the history entries. Therefore, the number of trials needed to transmit a single chunk can be modeled as a geometric random variable T_C with parameter p_C. We have $E[T_C] = \frac{1}{p_C}$. If we have n entries in the history and if $\widehat{X}(n) = \frac{1}{n}\sum_{i=1}^{n} r_i$ is the mean number of actual chunks needed to transmit L chunks of size c', then we estimate

$$E[T_C] \approx \frac{\widehat{X}(n)}{L'}$$

and accordingly $\frac{L'}{\widehat{X}(n)}$ gives the estimated success probability p_C to transmit a chunk correctly. The chunk error probability is then estimated as $1-p_C$. For a BSC channel the relationship between the bit error rate p and the chunk error rate $1-p_C$ is given by:

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

which can be solved for p as:

$$p = 1 - p_C^{\frac{1}{c+h}}$$

However, practical experimentation with known channel conditions showed that a more accurate estimate is given by:

$$p = 1 - p_C^{\frac{1}{c+h+o'}}$$

which can be explained by the fact that all frames for a request have a header of size o' bits which for the first and subsequent retransmission is shared between only a small number of chunks. In addition, errors in the header lead to loss of all the contained chunks. The actual adjustment algorithm requires a fixed $n \geq 1$ and adjusts the chunksize c' only within some bounds c_{min} and c_{max}, the initial chunk size is $c_{initial}$. Furthermore, if $\widehat{X}(n) = L'$ then we choose $c' = c_{max}$. After adjusting the chunk size the history is cleared.[5]

We discuss some experiments with this adaptation scheme. The results are obtained using simulations. The simulator uses the OMNet++ discrete-event simulation package [10] and is validated by comparison with the analytical results from Section III. We have chosen $h = 16$, $o' = 116$, $c_{min} = 2 \cdot h = 32$, $c_{max} = 100 \cdot h = 1600$ and $c_{initial} = 300$. We have set s to 3600 bits and new requests are generated every 10 milliseconds. The channel bandwidth b is 1 MBit/s, the history depth is $n = 10$ and we considered two constant bit error rates of $p = 0.001$ and $p = 0.00001$. For $p = 0.001$ the optimal chunk size is $\approx$ 118 bits and for $p = 0.00001$ it is $\approx$ 1256 bits. The evolution of the chunksize over the first 100 seconds is shown for $p = 0.001$ in Figure 4 (the corresponding figure for $p = 0.00001$ is omitted due to lack of space). The following points are remarkable:

- For $p = 0.001$ the actual chunksizes have only a comparably small variance (coefficient of variation of $\approx$ 0.2) and a mean value of $\approx$ 126 bits, which is close to the optimal chunksize of 118 bits. Therefore, it is not surprising that the achieved goodput of 0.643 is close to the optimal goodput of 0.646.

[5]We made the implicit assumption that all history entries have the same weight. However, one can argue that old entries are useless and that entries should be weighted according to their age. However, in the evaluations discussed in this paper we have assumed a periodic traffic source with 100 Hz, and the setting $n = 10$ means that the history entries are at most 100 milliseconds old when they are considered for adjusting c'. But clearly, for a practical implementation a more sophisticated scheme is needed.

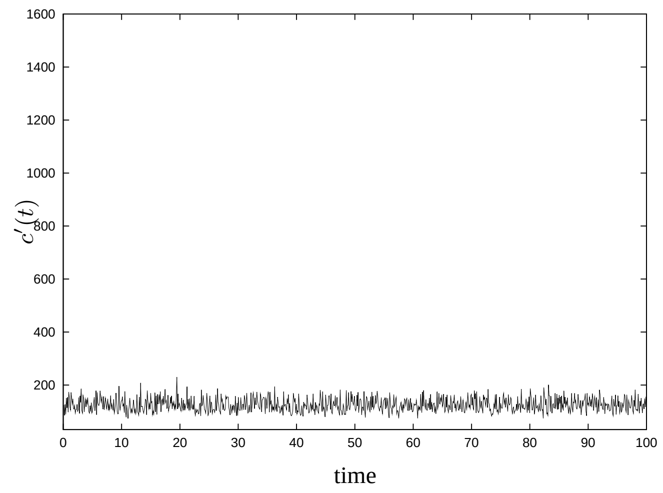

Fig. 4. Evolution of estimated chunk sizes for a BSC with $p = 0.001$ for the first 100 seconds

- For $p = 0.00001$ the mean value of the actual chunksizes is 1343 bits, as compared to the optimal value of 1256 bits. The coefficient of variation of the chunk sizes is ≈ 0.3, therefore, for smaller p the chunk sizes are more variable than for larger p. The actually achieved goodput is 0.94, which three percent less than the theoretical optimum of 0.97.

The main advantage of this simple estimator is that it uses readily available data and that its computation costs are quite moderate, the computation time of the mean value $\widehat{X}(n)$ is basically $O(n)$.

V. Related Work

The idea to use intermediate checksums seems, to the best of our knowledge, not be discussed in the literature so far, except from [4], where the approach is briefly sketched but not followed anymore. On the other hand, the idea to adapt frame sizes to the current channel conditions is not new, in [4] it is shown that adaptation can give significant gains in goodput. The adaptation is done purely below the IP layer, whichs frames are fragmented into smaller ones and reassembled at the receiver. The bit error rate is estimated at the receiver, e.g. by observing frame/fragment losses or from an FEC decoder. After a certain time the receiver requests a new frame size from the transmitter, piggybacked on an acknowledgement. Another paper dealing with adapting the frame sizes for the send-and-wait protocol over fading channels is [11]. Again, the adaptation strategy is based on counting positive and negative acknowledgements. It is shown by simulation that the adaptive protocol gives significantly better goodput than with static packet sizes, even for difficult fading channels. The combination of adaptive frame length control and FEC is explored in [12].

VI. Conclusions

In this paper we have introduced the idea to use intermediate checksums in large frames to avoid loss of many correct bits when only a few bits are erroneous. We have shown that in case of a static binary symmetric channel and comparably high error rates as they might occur in wireless channels, the ICF scheme delivers significantly better user data goodput than the traditional HDTF scheme. In addition, the ICF scheme can be easily incorporated into other ARQ protocols than send-and-wait.

However, since wireless channels are time-varying and typically unknown, the right chunk size cannot be statically configured. Therefore, we have developed a simple scheme for adapting the chunk size to the current channel conditions. Despite its simplicity (only readily available information is taken into account) the adaptation scheme achieves close-to-optimum goodput for high bit error rates, for small bit error rates the results are still satisfactory.

There is a lot of room for further research. One important topic is the behavior of the ICF scheme and the adaptation scheme for different types of wireless channels, e.g. when errors occur in bursts. Another important topic is to investigate other adaptation schemes and to incorporate the effects of FEC.

References

[1] H. Liu, H. Ma, M. E. Zarki, and S. Gupta, “Error control schemes for networks: An overview,” *MONET – Mobile Networks and Applications*, vol. 2, no. 2, pp. 167–182, 1997.

[2] D. Haccoun and S. Pierre, “Automatic repeat request,” in *The Communications Handbook*, J. D. Gibson, Ed. Boca Raton, Florida: CRC Press / IEEE Press, 1996, pp. 181–198.

[3] The Editors of IEEE 802.11, *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, Nov. 1997.

[4] P. Lettieri and M. Srivastava, “Adaptive frame length control for improving wireless link throughput, range and energy efficiency,” in *Proc. INFOCOM 1998*. San Francisco, CA: IEEE, 1998, pp. 564–571.

[5] M. S. Gast, *802.11 Wireless Networks – The Definitive Guide*. Sebastopol, CA: O’Reilly, 2002.

[6] A. M. Michelson and A. H. Levesque, *Error-Control Techniques for Digital Communication*. New York: John Wiley and Sons, 1985.

[7] S. Lin and D. J. Costello, *Error Control Coding – Fundamentals and Applications*. Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

[8] A. Willig, M. Kubisch, C. Hoene, and A. Wolisz, “Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer,” *IEEE Transactions on Industrial Electronics*, vol. 49, no. 6, pp. 1265–1282, 2002.

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

[10] *OMNet++ V. 2.3 Simulation Package*, 2001.

[11] S. Hara, A. Ogino, M. Araki, M. Okada, and N. Morinaga, “Throughput Performance of SAW-ARQ Protocol with Adaptive Packet Length in Mobile Packet Data Transmission,” *IEEE Transactions on Vehicular Technology*, vol. 45, no. 3, pp. 561–569, Aug. 1996.

[12] P. Lettieri, C. Schurgers, and M. B. Srivastava, “Adaptive link layer strategies for energy-efficient wireless networking,” *Wireless Networks*, vol. 5, no. 5, pp. 339–355, Nov. 1999.