This is a draft manuscript of an article accepted for publication in IEEE Transactions on Industrial Informatics

Memory-Efficient Segment-Based Packet-Combining Schemes in Face of Deadlines

Andreas Willig

Abstract

In this paper we consider segment-based hard-decision packet-combining schemes. The schemes presented here are memory-efficient and easy to implement, and some of them appproach the performance of majority-voting schemes without having the same memory requirements. One particularly interesting scheme combines segment-based transmission with Luby-type erasure codes.

Index Terms

hard-decision packet combining, memory-efficiency, deadline constraints, stochastic/Markov modeling

I. Introduction

The networking technologies and protocols used in distributed industrial control and monitoring applications often have to fulfill stringent real-time and reliability requirements, which is reflected in the design of their protocols. When wireless transmission technologies are involved, it is well-known that channel errors are a significant obstacle in fulfilling the real-time and reliability requirements [1] and their occurence should be carefully considered. One important part of wireless real-time protocols is therefore the error-control strategy, which often is based on the usage of acknowledgements and retransmissions (such protocols are called ARQ protocols – Automatic Repeat reQuest [2]). The goal is thus to make the best possible use of the extra time required to perform the retransmissions so that a packet can be successfully transmitted within its deadline.

Packet combining schemes are an attractive ingredient for ARQ-based error-control schemes (which are then called type-II or type-III hybrid-ARQ schemes [2], [3], [4]), especially over error-prone channels like wireless channels. In packet combining schemes a receiver does not throw away erroneous copies of a frame but keeps them and tries to use them for decoding upon arrival of future frames. In this paper we consider hard-decision packet-combining schemes, i.e. schemes operating on demodulated data bits as they are delivered by a wireless transceiver to the higher layers. The term hard-decision refers to the lack of any reliability information associated with the received bits or any other channel side-information. This assumption is practically relevant, since transceivers compliant with standardized wireless technologies like IEEE 802.15.4 or the family of IEEE 802.11 standards behave according to this assumption: they just deliver decided bits to the higher layers, no additional per-bit reliability information.

One technique to perform packet combining under these constraints is majority-voting. The receiver buffers all erroneous copies of a frame and for each bit position (including the trailing CRC) applies a majority-voting rule. The resulting packet is checked for correctness. A major drawback of majority voting, however, are the significant buffer requirements, since at least three copies of a frame are required for majority voting to be effective. This can be a serious drawback in memory-limited stations like for example the nodes of a wireless sensor network.

In this paper we consider a class of alternative schemes with much more modest buffer requirements at the receiver. We call these schemes *segment-based schemes*. The receiver needs to provide only the buffer space for one copy of the frame, plus some fixed overhead. In classical framing- and retransmission schemes a packet is equipped with one trailing CRC value for error detection – if the CRC is wrong, the receiver discards the whole packet. The transmitter then (re-)transmits the frame until it receives an acknowledgement or the frame's deadline is exhausted. In contrast, in the segment-based schemes investigated in this paper the data part is partitioned into *segments* and each segment has its own CRC. The receiver does not throw away erroneous frames, but buffers all correct segments in order to combine them with segments arriving in later re-transmissions. We present three different segment-based schemes. One of them is based on the observation that the segment-based schemes can be combined with Luby-type erasure codes [5]. These codes allow to encode k user-data symbols into $n > k$ symbols such that for the receiver it suffices to receive any k out of these n symbols to decode the k user-data symbols. It is possible to regard one of our segments as one symbol in these codes. Since they also allow

linear-time coding and decoding, they are suitable candidate schemes in the context of wireless sensor networks. Segment-based schemes can be implemented on top of commercial transceivers for technologies like IEEE 802.15.4 and IEEE 802.11, they only require the ability for quick in-memory computations of CRC values. If no dedicated hardware is available for this, CRC computations can be done in software with computational overhead that is linear in the amount of data (e.g. [6]).

We compare the performance achievable with the different segment-based schemes against three "baseline" schemes (the classical, single-CRC scheme, a majority voting scheme and a fragmentation scheme similar to IEEE 802.11) under a deadline constraint. In many applications (like for example in healthcare or process control) it is desirable to achieve both reliable and timely transmission of data in an efficient way. This is reflected by the major performance parameters investigated in this paper: the first major performance criterion is the probability of successful message delivery within the prescribed deadline (called *success probability*), and the second one is the number of transmitted bits used without any deadline (called the *unbounded costs*) and within this deadline (called the *bounded costs*). These performance measures are mostly obtained from stochastic models of the different schemes and under idealized and simplified conditions. Firstly, as a channel model the binary symmetric channel (BSC) is chosen, in which all bit errors occur independently of each other and with the same probability. This allows to keep the models tractable. Secondly, it is assumed that the transmitter has precise knowledge of the current channel bit error rate, so that for each scheme optimal parameters can be chosen. Thirdly, it is assumed that the feedback channel is perfect, i.e. is error-free and has no delay. These assumptions allow to compare the optimum performance achievable with each of the schemes and furthermore the obtained results are useful as upper bounds for the actual performance obtained under more realistic conditions. Our results show that all segment-based schemes improve significantly upon the classical scheme and some actually come close to the performance of the majority-voting scheme without having its buffer requirements. Under perfect feedback, one of the segment-based schemes even outperforms majority voting for a large range of bit error rates.

This paper extends previous work (see [7]), an extended version is available as technical report [8]. It is structured as follows: in Section II we describe the system model and the different schemes considered in this paper. For all of the described schemes we derive an analytical

model, and for most models we derive expressions for the desired performance metrics in terms of these models. In Section III we present numerical results for different performance measures under the assumptions that the channel is a static binary symmetric channel and the bit error rate is perfectly known to all schemes. In the following Section IV we present results for a time-varying channel in which the transmitter has no access to channel state information. Related work is discussed in Section V and the conclusions are given in Section VI.

II. System model and considered schemes

After discussing the system model, we describe the segment-based packet-combining schemes and their associated analytical models. In addition, we describe (and model) three baseline schemes (classical scheme, fragment-scheme, majority-voting) against which we compare the segment-based schemes.

We give a brief summary of our notations. The sets $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ and $\mathbb{N} = \{1, 2, 3, \ldots\}$ denote the sets of non-negative integers with and without zero. For a set B the notation $|B|$ refers to the cardinality of the set B. For some real value x the notation $\lfloor x \rfloor$ denotes the largest integer that is smaller than or equal to x. For some matrix $\mathbf{P}$ the notation $[[\mathbf{P}]]_{i,j}$ refers to the i, j-th component of $\mathbf{P}$, with i and j from the state space on which the matrix is defined. Furthermore, the function $\mathbf{1}_A(x)$ is the indicator function of the set A, i.e. $\mathbf{1}_A(x) = 1$ if $x \in A$ and $\mathbf{1}_A(x) = 0$ otherwise. For reference, we remind here three different discrete probability distributions:

- A binomial random variable with parameters $n \in \mathbb{N}$ and $p \in [0, 1]$ models the number k of successes seen in n iid Bernoulli experiments with success probability p. It has probability mass function

$$b(k; n, p) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \qquad (k \in \{0, 1, \ldots, n\})$$

  and expectation np.
- A geometric random variable with range $\mathbb{N}$ models the number of iid Bernoulli experiments with individual success probability p that are required so that the first success shows up. Its probability mass function is:

$$p(k) = p(1-p)^{k-1} \qquad (k \in \mathbb{N})$$

  and its expectation is $1/p$.

- A negative binomial random variable counts in a series of iid Bernoulli experiments with individual success probability p the number k of failed trials that are required to achieve r successes. The probability mass function of a negative binomial random variable is

$$f(k; r, p) = \binom{k+r-1}{k} p^r (1-p)^k \qquad (k \in \mathbb{N}_0)$$

and the expected number of failed trials until the r-th success is $r\frac{1-p}{p}$.

A. System model

We consider a system comprising of one transmitter, one receiver and one channel in between. The channel is a BSC with bit error rate $p \in (0, 1)$. We assume that the feedback channel is perfect, i.e. acknowledgements are received error-free and without delay. During discussion of the schemes we point out which schemes depend critically on the feedback channel. It is noteworthy to point out that schemes which do not critically rely on feedback can have advantages over asymmetric channels. We furthermore ignore for simplicity the time needed for switching the transceiver from receive to transmit mode or vice versa.

We do not consider interactions with the medium access control layer. For simplicity, we assume that the transmitter has uninterrupted channel access from the start of the packet transmission process until the packets deadline. The link-layer entity of the transmitter receives from its upper layers a user message of size s bits, to which a packet deadline d is associated (expressed in a technology-independent fashion as number of bits that could be transmitted within the deadline). Within this packet deadline, the transmitter performs a number of frame transmissions (initial transmission and retransmissions) according to the stop-and-wait protocol. We refer to these transmissions also as *trials* and the set of all trials carried out for the same packet is also referred to as a *transaction*. The transmitter performs further trials either until it receives a positive acknowledgement from the receiver or the deadline is exhausted. From the BSC assumption, all trials are stochastically independent.

We make the following assumptions regarding the framing. A frame has different kinds of headers: a physical-layer (PHY) header and a MAC header. The PHY header contains a preamble of length $o_{P,Pr}$ bits and a start-frame delimiter (SFD) of length $o_{P,SFD}$ bits. The preamble allows the receiver to acquire symbol synchronization and the SFD denotes the start of the useful part of the frame. It may well happen that the receiver does not acquire the preamble properly or

receives an erroneous SFD field. In both cases the whole packet is lost and nothing of it is visible to the receiver. To model packet losses induced by preamble/SFD misdetection, we use as a simple approximation the rule that a preamble misdetection happens if more than $o_{P,0}$ bits of the preamble are erroneous. The probability of properly receiving the PHY header is then given by

$$P_{\text{PH}} = \left(\sum_{k=0}^{o_{P,0}} b(k; o_{P,Pr}, p) \right) \cdot (1-p)^{o_{P,SFD}} \tag{1}$$

The MAC header typically contains addressing information and control information like sequence numbers, which we assume to be the same for all schemes. However, depending on the scheme, the MAC header contains further fields. Specifically, in all schemes performing packet combining the MAC header also contains a separate header checksum.[1] The MAC header consists of $o_{M,A}$ bits of common addressing and control information, and the packet combining schemes additionally use h bits for a header checksum and $o_{M,P}$ additional control bits for packet combining purposes. If the header checksum is wrong, the packet is fully discarded and no action whatsoever is taken. The probability of correctly receiving the MAC header when $o_{M,P}$ bits of additional bits for the packet combining scheme are used is given by:

$$P_{\text{MH}}(o_{M,P}) = (1-p)^{o_{M,A}+h+o_{M,P}} \tag{2}$$

A final assumption concerns the CRC checksums used in this paper. We allocate $h = 16$ bits for a CRC checksum, and as a simplifying assumption these checksums are perfect, i.e. there are no undetected errors.

We consider two major performance parameters:

- The *success probability* is defined as the probability that a packet can be successfully transmitted within its deadline d. We measure the success at the receiver: it is required that the receiver has received (or combined) a correct copy of the packet. The transmitter is not necessarily aware of this when acknowledgements can get lost.

[1]Rationale: the receiver must maintain a cache of erroneous copies of a packet for combining purposes. When a new packet transaction starts, the cache must be cleared, otherwise the erroneous frames belonging to the last transaction would "poison" the frames for the new transaction. To detect a new transaction, the receiver checks the addressing fields and the sequence number field. Without a separate header checksum these fields can be erroneous and the packet cache might be cleared in the mid of an ongoing transaction. These "cache breakdowns" would be harmful to the performance of packet combining schemes and furthermore they would make the stochastic models developed below significantly more complex.

- The *unbounded costs* measure the average number of bits that the transmitter requires until success when there is no deadline constraint. Again, we measure success at the receiver. With an imperfect feedback channel, the transmitter might not be aware of the success and performs further trials. The costs for these further trials are not considered.

In the technical report [8] a third measure is investigated: the *bounded costs* measure the average number of bits that the transmitter sends within the deadline d. However, to save space we omit the model and the results for this performance measure.

B. Classical and fragment-based schemes

In the ***classical scheme*** no packet combining is performed at the receiver. The s-bit user message is prepended with a physical layer header and a MAC layer header containing no combining-related fields (i.e. $o_{M,P} = 0$) and no separate header checksum. A CRC checksum of size h bits is appended to the packet. The total packet length is $L_{\text{class}}(s) = o_{P,Pr} + o_{P,SFD} + o_{M,A} + s + h$ bits. Not counting the time required for acknowledgement transmission, the transmitter can transmit the frame

$$t_1(d,s) = \left\lfloor \frac{d}{L_{\text{class}}(s)} \right\rfloor \tag{3}$$

times within the given deadline d. No packet combining is applied at the receiver. From the BSC assumption, the success probability of the classical scheme is given by:

$$\partial_{\text{class}}(s,d,p) = 1 - \left(1 - P_{\text{PH}} \cdot (1-p)^{o_{M,A}+s+h}\right)^{t_1(d,s)} \tag{4}$$

To determine the unbounded costs, we observe that each trial can be considered as a Bernoulli experiment with success probability $P_{\text{PH}} \cdot (1-p)^{o_{M,A}+s+h}$ and the number of trials is a geometric random variable X. Therefore, the average unbounded costs are:

$$K_{U,\text{class}}(s,p) = \frac{L_{\text{class}}(s)}{P_{\text{PH}} \cdot (1-p)^{o_{M,A}+s+h}} \tag{5}$$

The ***fragment-based scheme*** is a modified (and simplified) version of the fragmentation- and reassembly scheme used in the IEEE 802.11 standard. The transmitter splits the s user data bits into fragments of size f (plus one smaller slack fragment when f does not evenly divide s). Each fragment is transmitted according to the classical scheme (i.e. with own header and trailing checksum) and with no bound on the number of trials spent for this fragment, but

within the deadline d all of the fragments must be transmitted successfully.[2] When f is the size of a full fragment, then the size of the slack fragment is $f_s = s - \left\lfloor \frac{s}{f} \right\rfloor$. Furthermore, let $L_F := L_{\text{frag,full}}(f) = o_{P,Pr} + o_{P,SFD} + o_{M,A} + f + h$ denote the frame size (in bits) of a full fragment, and $L_S := L_{\text{frag,slack}}(s, f) = o_{P,Pr} + o_{P,SFD} + o_{M,A} + f_s + h$ denote the frame size of the slack fragment (if any, otherwise we assume $L_S = 0$). Within deadline d and for fixed fragment size f a number of

$$t_2(d, f) = \left\lfloor \frac{d - L_S}{L_F} \right\rfloor \tag{6}$$

full fragments can be transmitted (at the end of the time we always need some spare time to transmit the slack segment, if any). When there is no slack fragment (i.e. $f_s = 0$), we can express the success probability as the probability that in $t_2(d, f)$ independent trials at least $r = \left\lfloor \frac{s}{f} \right\rfloor$ are successful (r is just the required number of successful transmissions), which can be written as:

$$\partial_{\text{frag}}(s, d, p, f) = \sum_{i=r}^{t_2(d,f)} b(i; t_2(d, f), P_F) \tag{7}$$

where $P_F = P_{\text{PH}} \cdot (1 - p)^{o_{M,A}+f+h}$ is the probability of successful transmission of one full fragment. When there is a slack fragment (i.e. $f_s > 0$) then the success probability can be expressed as:

$$\partial_{\text{frag}}(s, d, p, f) = \tag{8}$$
$$\sum_{i=r}^{t_2(d,f)} f(i - r; r, P_F) \cdot \left(1 - b\left(0; \left\lfloor \frac{d - i \cdot L_F}{L_S} \right\rfloor, P_S\right)\right)$$

where $P_S = P_{\text{PH}} \cdot (1 - p)^{o_{M,A}+f_s+h}$ is the probability of successful transmission of the slack fragment. The sum extends over the numbers $r \leq i \leq t_2(d, f)$ of trials required to transmit all r full fragments with $i - r$ failed trials in between. For each i the term $f(\cdot)$ gives (from the probability mass function of the negative binomial distribution) the probability that exactly $i - r$ failed trials are required, and the second term $b(\cdot)$ gives the probability that in the remaining the

[2]In realistic implementations, including the scheme of IEEE 802.11, the number of trials spent for one fragment is bounded. Furthermore, an efficient implementation would stop any effort to transmit further trials once it becomes clear that the remaining fragments cannot be transmitted fully within the remaining time until deadline expiration. These optimizations, however, do not improve the success probability of the fragment-scheme, they just tend to decrease the number of transmitted bits. In this paper for simplicity we do not consider them.

remaining $d - i \cdot L_F$ bits the slack fragment can be transmitted successfully at least once – the expression $\left\lfloor \frac{d-i\cdot L_F}{L_S} \right\rfloor$ gives the number of trials available for the slack fragment.

To compute the unbounded costs, we observe the following: One trial to transmit a full fragment corresponds to one Bernoulli experiment with success probability P_F and the number of Bernoulli trials required to achieve r successes is a negative binomial random variable. The average number of trials required until all full fragments are successfully received is then

$$r + r \cdot \frac{1 - P_F}{P_F}$$

and the unbounded costs when choosing a fragment size f are therefore

$$K_{U,\text{frag}}(s,p,f) = \\ L_F \cdot r \cdot \left(1 + \frac{1-P_F}{P_F}\right) + \mathbf{1}_{(0,\infty)}(f_s) \cdot \frac{L_S}{P_S} \tag{9}$$

When there is a slack segment, then we additionally have the costs required to transmit one slack segment successfully. This can be modeled as a geometric random variable and the additional number of bits is L_S/P_S.

Please note that the classical scheme is not critically dependent on the presence of feedback, but the fragment-based scheme is. When the feedback is lost, the classical scheme would simply repeat the frame as often as possible within the deadline and the receiver is able to decode it with a certain probability. In the fragment-based scheme with $f < s$ and without feedback the receiver would never get the second segment, thus having no chance to fully decode the frame.

C. Majority voting scheme

The ***classical-mv scheme*** is similar to the classical scheme, but the receiver performs bitwise majority voting on those (erroneous) frames it receives, followed by a CRC check on the resulting frame. In contrast to the classical scheme the MAC header requires a separate header checksum of h bits (see Footnote 1) but no further combining-related data, i.e. $o_{M,P} = 0$. The total packet length is therefore $L_{\text{mv}}(s) = o_{P,Pr} + o_{P,SFD} + o_{M,A} + h + s + h$. Correspondingly, the transmitter has

$$t_3(d,s) = \left\lfloor \frac{d}{L_{\text{mv}}(s)} \right\rfloor \tag{10}$$

trials at its disposal.

To compute the performance measures, we model the reception process as a time-homogeneous discrete-time Markov chain $(X_n)_{n\geq 0}$ [9]. The state X_n of the Markov chain reflects the number of received frames with proper PHY and MAC headers but erroneous data part that the receiver has in its combining cache. The state space is given by $\mathcal{S} = \{0, 1, \ldots, t_3(d,s) - 1\} \cup \{succ, fail\}$ where $succ$ refers to the success state, in which the receiver either has received a fully correct packet (proper PHY and MAC header, correct trailing checksum), or, after receiving a packet with proper headers and erroneous data part, was able to combine the correct packet from the majority-voting algorithm applied to the new packet and all previously stored copies. The state $fail$ corresponds to the fail state, in which the deadline has been exhausted without success. The start state is $X_0 = 0$. One time step of the Markov chain corresponds to one transmission trial of the transmitter. To represent the state transition probabilities, it is convenient to introduce the following abbreviations:

- $\beta = P_{\text{PH}} \cdot P_{\text{MH}}(0) \cdot (1-p)^{s+h}$ represents the probability that the receiver receives a trial without any errors, and
- $v(k,s,h) = \left(\sum_{i=\lfloor \frac{k}{2} \rfloor + 1}^{k} b(i;k,1-p)\right)^{s+h}$ represents the probability that with k independent erroneous copies of a frame the majority-voting procedure gives a correct frame – to make this happen, for each bit position in the data part and the trailing checksum more than half of the received packets must have the correct value.

With these abbreviations one can write the state transition probabilities as follows. The states $succ$ and $fail$ are absorbing states, i.e. we have $p_{succ,succ} = 1$ and $p_{fail,fail} = 1$. For all states $i \in \{0, \ldots, t_3(d,s) - 1\}$ the probability to remain in this state is given by the probability that one of the PHY or MAC headers is corrupt, i.e.:

$$p_{i,i} = 1 - P_{\text{PH}} \cdot P_{\text{MH}}(0)$$

For $i \in \{0, \ldots, t_3(d,s) - 1\}$ the probability to reach the success state is given by:

$$\begin{aligned}
p_{0,succ} &= \beta \\
p_{1,succ} &= \beta \\
p_{i,succ} &= \beta + P_{\text{PH}} \cdot P_{\text{MH}}(0) \\
&\quad \cdot (1 - (1-p)^{s+h} s) \cdot v(i+1, s, h) \qquad , (i \geq 2)
\end{aligned}$$

In state 0 only a proper frame reception can lead to a success. In state 1, a success is also only possible when a proper frame is received, since only two packets (one freshly received, one cached) are not sufficient for majority voting. In state $i \geq 2$ a success is possible if either the received frame is proper, or if the PHY and MAC headers are proper, the data part and checksum are not proper but the receiver is able to combine the packet together using the $i+1$ copies it now has (one new, i cached). Finally, for $i \in \{0, \ldots, t_3(d,s) - 2\}$ the probability to reach the successor state $i+1$ is given by:

$$p_{i,i+1} = 1 - p_{i,succ} - p_{i,i}$$

and from the last state $i = t_3(d,s) - 1$ the only alternative to success is failure, i.e.

$$p_{i,fail} = 1 - p_{i,succ}$$

The success probability can be computed from the state transition matrix $\mathbf{P}$ as follows:

$$\partial_{\text{class-mv}}(s,d,p) = [[\mathbf{P}^{t_3(d,s)}]]_{0,succ} \tag{11}$$

Please note that this majority-voting procedure requires buffer space for up to $t_3(d,s)$ frames at the receiver. Three buffers are the minimum for majority voting to be effective.

In order to obtain the unbounded costs for both the majority voting scheme, we use for this and most of the other Markov models the framework of potential theory for Markov chains [9, Sec. 4.2]. This is a generalization of the theory of hitting times and hitting probabilities, the relevant definitions and a relevant theorem are paraphrased in Appendix VII. In order to utilize this theorem we have to assign cost vectors, we have to specify the set of inner states and final states, and we have to ensure that the final states are reached with probability one within finite time. We can use the following setup:

- The inner states are $D = \{0, 1, \ldots, t_3(d,s) - 1\}$, the final states are $\partial D = \{succ, fail\}$.
- The costs (bits per trial) are the same in all inner states $i \in D$ and are given by $c_i = L_{\text{mv}}(s)$. In the final state $\partial D = \{0\}$ no further transmissions are made, so $\mathbf{f} = \mathbf{0}$.
- The final states are absorbing, all other states are transient, so indeed the final states will be reached in finite time with probability one.

The unbounded costs $K_{U,\text{class-mv}}$ are then obtained from solving the linear equation system given in Appendix VII for the variable ϕ_0, i.e. the average costs for the start state $X_0 = 0$.

However, for the majority-voting model one twist must be applied: since with an unbounded number of trials there can be no failure, the failure state should theoretically be dropped and the Markov chain turns into one with infinite state space. In this case, however, the solution of the linear equation system for ϕ_0 is hard to obtain in closed form. Therefore, the unbounded costs are approximated with the bounded costs using a very large deadline d.

One cannot expect the majority-voting model to be entirely accurate. When the receiver has already $i > 2$ copies in its cache and receives another erroneous copy with correct header, it tries to combine $i+1$ copies. From the Markov assumption, this combination trial is independent of the previous trial to combine i copies. In reality, the combining trials are *not* stochastically independent, since i of the $i+1$ copies are identical to the previous combining trial. For example: If a receiver with three copies has for one bit position two wrong votes, a fourth packet can at most equalize the votes for this position but it does not enable correct decoding. Under the independence assumption all four observations for this bit would be drawn freshly and not taken from the cache. It is shown in the report [8] that indeed the analytical model gives more optimistic results for the success probability than simulation results obtained using a real packet cache.

The majority-voting scheme does not critically rely on feedback.

D. Segment-based schemes

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 splitting the data part into smaller segments such that each segment has its own checksum, the errors are confined to smaller parts and the information in correct segments can be kept. A key advantage of the segment-based schemes as compared to other packet combining schemes are the modest buffer requirements. At the receiver side only one buffer is needed in which the full message can be assembled from the correct segments.

In the ***segment-normal scheme*** the s user data bits are partitioned into L *segments*, each having a size of $c = \lfloor s/L \rfloor$ bits (slack segments are stuffed up). To each segment a separate checksum of h bits is appended, which is computed only over the data bits of the segment. The initial frame is formed by appending all the segments to a PHY header and a MAC header. The overall header size is $o_{P,Pr} + o_{P,SFD} + o_{M,A} + o_{M,P} + h$, as we require an additional header checksum and a number $o_{M,P}$ of bits to encode the segment size. To keep the overhead $o_{M,P}$

small, the most straightforward approach is to fix a common codebook for a set of up to $2^{o_{M.P}}$ pre-defined segment sizes and to encode the actual choice of segment size in the header. The total packet size is then $L_{\text{seg-norm}}(s,c) = o_{P,Pr} + o_{P,SFD} + o_{M,A} + o_{M,P} + h + L \cdot (c+h)$. The transmitter transmits the initial frame. If the PHY and MAC header are proper, the receiver checks each segment separately and buffers those correct segments that it has not stored yet. If the receiver possesses all segments of a message, it delivers the frame to its upper layers and sends a *final acknowledgement*. Otherwise, the receiver transmits an empty *incomplete acknowledgement* frame and in response the transmitter re-transmits the whole initial frame again. The transmitter performs re-transmissions until the deadline d is exhausted. The number of available trials is given by

$$t_4(d,s) = \left\lfloor \frac{d}{L_{\text{seg-norm}}(s,c)} \right\rfloor \tag{12}$$

Please note that this scheme does not critically depend on successful transmission of acknowledgements.

The segment-normal scheme, however, does not make optimal use the available time budget, since it re-transmits segments that the receiver already has. We therefore consider an alternative scheme. In the ***segment-reduced scheme*** the receiver includes the identifications of the missing segments 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 segments, resulting in smaller packets and in more available retransmission trials within the deadline d. However, the available number of trials is now random. Again, it suffices for the transmitter to encode the segment size c into the MAC header. When the feedback channel does not work, this scheme degenerates into the segment-normal scheme. The segment-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.

Both the segment-normal scheme and the segment-reduced scheme can be modeled with the same time-homogeneous discrete-time Markov chain model $(X_n)_{n\geq 0}$. As a state variable X_n we choose the number of missing segments at the receiver after the n-th trial. When the packet consists of L segments, the state space is $0,\ldots,L$ and the start state is $X_0 = L$. A single segment is correctly received with probability $P_S = (1-p)^{c+h}$. The probability that the PHY and MAC headers are proper and that k out of M segments in a frame are erroneous is given

by (binomial distribution):

$$r(k, M) = P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P}) \cdot b(k; M, 1 - P_S)$$

With this, the state transition probabilities are as follows. The state 0 corresponds to a success, i.e. there are no further segments outstanding. This state is absorbing and therefore $p_{0,0} = 1$. To go from state $i \in \{1, \ldots, L\}$ to a lower state $0 \leq j < i$ it is required that the next received packet has proper PHY and MAC headers and that $i - j$ out of i missing segments are received, i.e. $p_{i,j} = r(j, i)$ for $j < i$. Finally, the diagonal elements for states $i \in \{1, \ldots, L\}$ are simply given by $p_{i,i} = 1 - \sum_{j=0}^{i-1} p_{i,j}$.

After collecting the state transition probabilities into a state transition matrix $\mathbf{P}$, the success probability for the segment-normal scheme can be expressed as:

$$\partial_{\text{seg-norm}}(s, d, p) = [[\mathbf{P}^{t_4(d,s)}]]_{0,L} \tag{13}$$

It is possible to derive an expression[3] that does not use matrix powers, but this is numerically much more unstable (especially for small values of p) than the equivalent computation based on matrix powers. For the segment-reduced scheme, however, things are more difficult since the number of trials that can be made within the deadline d is itself random. It would theoretically be possible to replace the simple Markov model presented here by an extended one which also tracks the accumulated transmission costs (the final states would then be all states with accumulated costs $\geq d$), but the state space of such a model becomes very large. Therefore, we rely on a simulation approach to assess the success probability $\partial_{\text{seg-rdcd}}(s, d, p)$.

In order to obtain the unbounded costs for both the segment-normal and the segment-reduced scheme, we again use the framework of potential theory with the following setup:

- The inner states are $D = \{1, 2, \ldots, L\}$, the final state is $\partial D = \{0\}$.
- In the segment-normal scheme, the costs (bits per packet transmission) are the same for all inner states $i \in D$ and are given by $c_i = L_{\text{seg-norm}}(s, c)$. In the segment-reduced scheme,

[3]It is shown in [10] that the success probability of the segment-normal and the segment-reduced scheme after n trials can be expressed as:

$$1 + \sum_{i=1}^{L} (-1)^i \cdot \binom{L}{i} \cdot \left(1 - P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P}) \cdot (1 - (1 - P_S)^i)\right)^n$$

the costs in state $i \in D$ are given as $c_i = o_{P,Pr} + o_{P,SFD} + o_{M,A} + o_{M,P} + h + i \cdot (c + h)$. In the final state $\partial D = \{0\}$ no further transmissions are made, so $\mathbf{f} = \mathbf{0}$.

- The final state is absorbing, all other states are transient, so indeed the final state will be reached in finite time with probability one.

The unbounded costs $K_{U,\text{seg-norm}}$ and $K_{U,\text{seg-rdcd}}$ are then obtained from solving the linear equation system given in Appendix VII for the variable ϕ_L, i.e. the average costs for the start state $X_0 = L$.

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

E. Segment-based schemes with Luby-type erasure codes

In [5], Luby et al have introduced a class of systematic erasure codes has been introduced in which k symbols are encoded into $n > k$ symbols so that reception of *any* k-element subset of the n symbols suffices to successfully decode the k symbols with high probability. Erasure-codes are not designed for the correction of erroneous symbols but of missing symbols. One symbol, once received, is assumed to be correct. Two key feature of Luby-type codes are that their coding and decoding time is linear in the number of symbols, and furthermore that they make only very loose assumptions on the nature of symbols. In particular, it is possible to use an entire segment including its checksum as a symbol and the computation of redundant symbols out of the given ones is easy to achieve. When the checksum appended to a segment has good error-detection capabilities, bit errors are translated into segment erasures and can then be corrected by the code. Therefore, Luby-type codes are a very interesting extension for the segment-based schemes described so far. To support this, the transmitter and the receiver must be able to run the coding and decoding algorithm, respectively.

We use these codes in the ***segment-erasure scheme***. The transmitter first splits the s-bit message into k segments of size $c = \lfloor s/k \rfloor$ bits (a slack segment is stuffed up). To these k segments then the coding algorithm is applied, resulting in $n > k$ segments. The code rate $R = k/n$ is assumed to be fixed. For each trial the transmitter selects m (with $1 \leq m \leq n$) of these n segments and puts them in the frame. The parameter m is fixed and known to both transmitter and receiver, its purpose is to give additional control (besides the code rate) on the size

of packets. The transmitter indicates in the MAC header which m segments have been included. For facilitate stochastic modeling, we assume in this paper that the m segments are chosen randomly and independently, in which case the transmitter would need to include a bitmap of n bits into the MAC header, instructing the receiver which segments are included.[4] The receiver, when the PHY and MAC header are proper, checks all segments and keeps all segments which are proper and which it does not have already stored until it has k different segments. Then it can successfully decode the packet. Therefore, the receiver must be able to buffer k distinct segments plus the additional space required for decoding the message and storing the result. The number of available trials is given by

$$t_5(d,s) = \left\lfloor \frac{d}{o_{P,Pr} + o_{P,SFD} + o_{M,A} + o_{M,P} + h + m \cdot (c+h)} \right\rfloor \tag{14}$$

Please note that this scheme does not critically depend on the feedback channel.

Again we model the transmission process using a time-homogeneous discrete-time Markov chain $(X_n)_{n\geq 0}$. We define as the state the number of *different* segments that the receiver possesses. One time slot corresponds to one packet transmission by the transmitter. The initial state of the receiver is $X_0 = 0$ and the common range of all X_n is $\{0, 1, \ldots, k\}$, since after having k different segments we can decode.

In Appendix VII the transition probability matrix is derived. To express this, we first consider the probability

$$T_{i,\delta} = \Pr\left[X_1 = i + \delta \middle| X_0 = i\right]$$

(for $i \geq 0, m \geq \delta \geq 0, i + \delta \leq n$) that a receiver having already i different segments receives with the next trial (with PHY and MAC header being proper) exactly δ different segments that differ from all i segments it already has. It is derived in the appendix that for this probability we have:

$$T_{i,\delta} = \sum_{r=\delta}^{i+\delta} \binom{i}{r-\delta} \cdot \binom{n-i}{\delta} \cdot P_S^r \cdot Q_S^{m-r} \cdot \frac{(n-r)! \cdot m!}{(m-r)! \cdot n!} \tag{15}$$

[4] For practical implementations the additional header overhead of n might be too much. As an alternative, one could pre-define a fixed number M of different m-element subsets of all n segments and in each trial transmit one of the M corresponding packets in a round-robin fashion. In the packet header then only the index in the M-set would be given.

With this, the state transition probabilities can be obtained as follows. When the receiver is in state $k-1$, it is required to receive the PHY and MAC headers successfully and to receive at least one additional segment it does not have so far. Therefore:

$$p_{k-1,k} = P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P}) \cdot \left(T_{k-1,1} + T_{k-1,2} + \ldots + T_{k-1,\min\{m,n-k\}}\right)$$

which holds since the events that the receiver receives 1, 2, ..., $\min\{m, n-k\}$ segments that it does not yet have are mutually disjoint. For the diagonal element of state $k-1$ we simply have $p_{k-1,k-1} = 1 - p_{k-1,k}$. For all states $i \in \{0, \ldots, k-2\}$ we can distinguish three cases:

- For target states $j \in \{i+1, \ldots, k-1\}$ we must receive the PHY and MAC headers properly and exactly $j-i$ new segments, which happens with probability

$$p_{i,j} = P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P}) \cdot T_{i,j-i}$$

- For the target state $j = k$ we must receive the headers successfully and receive $k-i$, or $k-i+1$, or ..., or $\min m, n-k$ new segments, therefore:

$$p_{i,k} = P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P}) \cdot \left(T_{i,k-i} + T_{i,k-i+1} + \ldots + T_{i,\min\{m,n-k\}}\right)$$

- The diagonal element $p_{i,i}$ is one minus the sum of all other transition probabilities.

For the other case $m < k$ the state transition probabilities can be obtained in a similar fashion. After collecting the state transition probabilities in a transition matrix $\mathbf{P}$, we can express the success probability as:

$$\partial_{\text{seg-erase}}(s, d, p) = [[\mathbf{P}^{t_5(d,s)}]]_{0,k} \tag{16}$$

To compute the unbounded costs, we can apply the framework of potential theory (Appendix VII) in a fashion similar to the segment-normal scheme:

- The inner states are $D = \{0, 1, \ldots, k-1\}$, the final state is $\partial D = \{k\}$.
- The costs (in bits per packet) are for all inner states $i \in D$ given by $c_i = o_{P,Pr} + o_{P,SFD} + o_{M,A} + o_{M,P} + h + m \cdot (c+h)$, in the final state no costs are incurred.
- All inner states are transient, the final state is absorbing.

The unbounded costs $K_{U,\text{seg-erase}}$ are then obtained from solving the linear equation system given in Appendix VII for the variable ϕ_0, i.e. the average costs for the start state $X_0 = 0$.

F. Implementation considerations

An important comment on the segment-based schemes concerns their implementation. In general, segment-based schemes can be useful as part of the link-layer error-control strategy and should be implementable so as to be transparent to the higher layers.

On the transmitter side the following tasks need to be handled for the segment-based schemes:

- Selection of the segment size set.
- For each new transaction an actual segment size must be chosen.
- Partitioning of user data into segments and checksum computation.
- On-the-fly assembly of actually transmitted frames.

The segment size set must be determined at network configuration time and the mapping of actual segment sizes to codewords must be pre-configured in both transmitter and receiver. At runtime, in order to select a good segment size for the current channel conditions, the transmitter needs an estimate of the bit error rate. To simplify implementation, the selected segment size is maintained for the whole packet transaction, i.e. the initial transmissions and all retransmissions use the same segment size. Hence, the partitioning of user data into segments and the computation of segment checksums needs to be done only once, when the packet is prepared for the transaction. It is of course preferrable to perform checksum computation in hardware, but there are also efficient software implementations available.[5] For example, an algorithm developed in [6] for CRC computation requires only four shift operations and four exclusive-OR's to process one byte of user data in software.

For the segment-reduced and the segment-erasure scheme potentially no two trials for the same packet are identical – in the segment-reduced scheme the number of contained segments can shrink over time, in the segment-erasure scheme segments are randomly chosen. Therefore, for each trial a new frame must be assembled. The runtime costs of this assembly depend on the transceiver interface. Transceivers like the IEEE 802.15.4-compliant ChipCon CC2420 have a packet-based interface, i.e. a buffer with the full packet must be ready before transmission starts, and during transmission this buffer must not be modified. For the next trial the frame must

[5]Many commercially available transceivers for wireless technologies like IEEE 802.11 or IEEE 802.15.4 have the ability to compute a CRC checksum, but they typically do not more than appending it to an outgoing packet and do not provide mechanisms to compute multiple-checksums at user-defined points in a frame.

be assembled anew (at least in parts, the header does not change). This re-assembly involves copying of the prepared segments into the packet buffer. Other transceivers offer an interface in which the processor feeds the bits sequentially into the transceiver. In this case copying segments is not necessary, as the processor can determine the memory locations from which to read the next data bit on the fly.

In contrast, the implementation complexity of the segment-normal scheme is lower: since in all trials the same packet is transmitted, the packet needs to be prepared only once and placed in a buffer.

III. Results for the case of perfect channel knowledge

In this section we show performance results for the different schemes. These results have been obtained either by evaluating the analytical models described in the previous section or by stochastic simulation. For the segment-reduced scheme only simulation results are available for the success probability, whereas the unbounded costs can be computed analytically. In the simulations, for each bit error rate p a number of 50,000 transactions has been simulated.[6] Further simulation results have been produced to validate the analytical models for the classical, fragment, segment-normal and segment-erasure schemes. For all these schemes the simulation results and the analytical results are almost identical, so we do not show the simulation results for validation purposes.

The major parameters have been chosen as follows: the user message size is $s = 1024$ bits, the preamble size is $o_{P,Pr} = 64$ bits, the size of the SFD field is $o_{P,SFD} = 8$ bits, the number of allowable bit errors in the preamble is $o_{P,0} = 2$ bits, the size of the common parts of the MAC header is $o_{M,A} = 24$ bits, and the size of the checksums is $h = 16$ bits. The size $o_{M,P}$ for the MAC extension header (without header checksum) for the majority voting scheme is $o_{M,P} = 0$, and for all the segment-based schemes it is $o_{M,P} = 4$ bits encoding the segment size.[7]

[6]This leads to very tight confidence intervals. More specifically, the 99% confidence interval for the success probability is at most 0.0058 (compare [p. 417][11]). The confidence intervals are not shown in the figures.

[7]This includes the segment-erasure scheme as well, see the discussion in footnote 4 on how to keep the extension header size small by exchanging the random selection of segments by preparing a few pre-defined selections. We follow the hypothesis that the random selection underlying the Markov model provides a good approximation to the results that would be obtained with the pre-defined-selection approach. With two bits for encoding the segment size (see Section III-A) there are two bits available for specifying a pre-defined selection.

The bit-error rate p has been varied between 10^{-4} and 10^{-1}. In order to compare all schemes under ideal conditions, we assume that p is known precisely to the transmitter and that the fragment-scheme and all segment-based schemes can choose optimal fragment / segment sizes accordingly. The code rate for the segment-erasure scheme has been fixed to 2/3, and the number m of segments included in a transmission is always set to $(k+n)/2$ where again k is the number of segments required for the user data (which depends on the segment size) and $n = (3/2) \cdot k$ is the number of coded segments. The deadline d has been chosen as $d = 11400$ bits, which is sufficient for the classical scheme to perform ten trials. The segment-based schemes with fixed-size packets (segment-normal, segment-erasure) have in general less than ten trials at their disposal, depending on their configuration.

A. Selection of segment and fragment size

It is intuitively clear that a single fixed segment size c will not give the optimal results for all bit error rates p. At the same time, when only a few bits in the header are used as a codebook for a set of pre-defined segment sizes, it becomes important to choose an optimal (w.r.t. success probability) or at least a good segment size set. Similar considerations apply for the fragment scheme. In the technical report [8] this issue is discussed in some detail, here we only summarize the results. It turns out (not surprisingly) that the optimal segment / fragment size indeed depends on the bit error rate p, but it suffices to consider only very few alternative values for each scheme. For the remaining paper we have used the mappings shown in Table I for the segment-based schemes and in Table II for the fragment scheme. It should be noted that for the segment-based schemes the optimization has been carried out over the restricted set $\{8, 16, 32, 64, 128, 256, 512\}$, whereas for the fragment-scheme all integers between 8 and 512 have been considered as fragment sizes.

B. Major results

We first compare the success probability of all schemes under idealized conditions. This means that the bit error rate p is known, the fragment scheme is allowed to choose the optimal fragment size and the segment-based schemes choose their optimal segment size from the set $\mathcal{C} = \{8, 16, 32, 64, 128, 256, 512\}$ according to the results obtained in the previous section. The results are shown in Figure 1. The following points are noteworthy:

Segment-normal	**BER range**	**Segment size**
	$p \leq 0.01024$	64
	$0.01025 \leq p \leq 0.014$	32
	$0.015 \leq p$	16
Segment-erasure	**BER range**	**Segment size**
	$p \leq 0.00315$	128
	$0.00316 \leq p \leq 0.00763$	64
	$0.00764 \leq p \leq 0.0137$	32
	$0.0138 \leq p$	16
Segment-reduced	**BER range**	**Segment size**
	$p \leq 0.0153$	32
	$0.0154 \leq p$	16

TABLE I: Segment sizes versus bit error rate p for the segment-based schemes

BER range	**Fragment size**
$p \leq 0.0147$	61
$0.0147 \leq p$	57

TABLE II: Approximately optimal fragment sizes versus bit error rate p and the fragment scheme

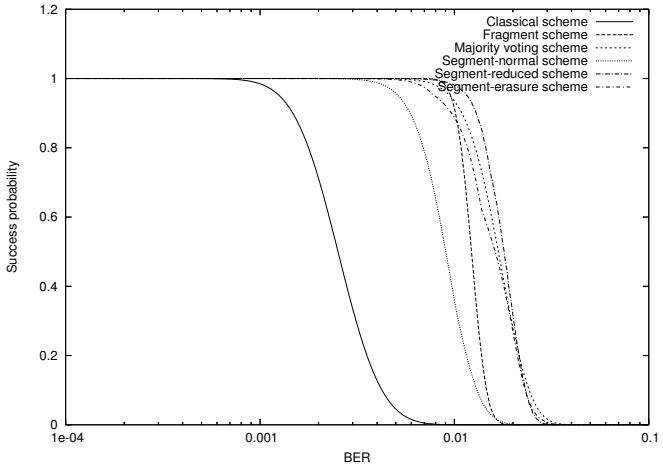

Fig. 1: Success probabilities of all schemes under idealized conditions for varying BER p

- The first, not very surprising, result is that all schemes are better than the classical scheme.
- There are three schemes which do not critically rely on feedback: the segment-normal scheme, the segment-erasure scheme and the majority-voting scheme (without feedback the segment-reduced scheme would degenerate into the segment-normal scheme). Between these three there is a clear ordering: segment-normal is clearly inferior to segment-erasure, which in turn is inferior to majority-voting, but comes close to it without having its buffer requirements.
- The segment-reduced scheme is the best one over a wide range of bit error rates, only for high bit error rates the segment-erasure and the majority voting schemes are better. Furthermore, the segment-reduced scheme is always better than the fragment scheme (whichs performance declines very quickly). Stated differently: the segment-reduced scheme makes much better use of the ideal feedback channel than the fragment scheme and at the same time does not depend *critically* on feedback. Therefore, it can even be used on asymmetric wireless links where the feedback channel is much worse than the forward channel.
- The performance difference between segment-normal and segment-reduced gives an idea about the value of feedback.

It should also be kept in mind that we have made no attempt to fully optimize the segment sizes for the segment-based schemes, we have restricted our attention to optimization over a small set $\mathcal{C}$ of selections.

We next compare the unbounded costs, which for all schemes are shown in Figure 2. The range of bit error rates shown in the graph has been restricted so that the differences for low bit error rates are still visible. For very low bit error rates the classical scheme and the majority voting schemes are the best (since they have no or very little frame overhead), for all the other schemes their respective overhead becomes visible – at low bit error rates the additional error handling capabilities of the other schemes are not needed and the overhead is wasted. The fragment-based scheme has the largest overhead, followed by the segment-reduced scheme. The larger overhead of the segment-reduced scheme over the segment-normal and segment-erasure schemes is caused by the fact that the segment-reduced scheme chooses smaller segment sizes than the other schemes for the low bit error rates. The additional overhead of segment-reduced pays out as the bit error rate increases – this is also true for the other segment-based schemes and the fragment scheme. The observable "nonlinearities" in the curve for the segment-erasure

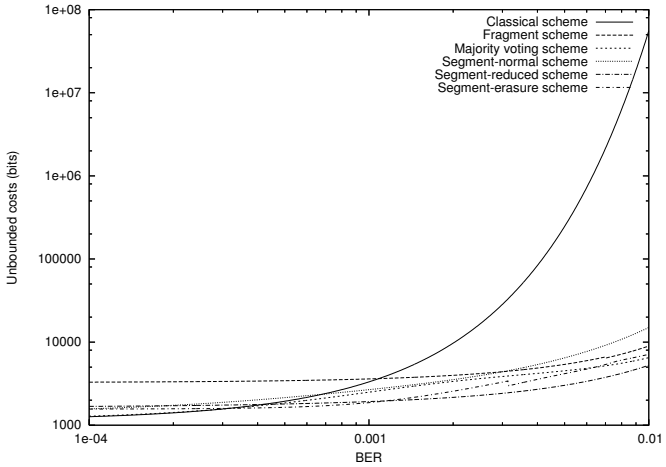

Fig. 2: Unbounded costs of all schemes under idealized conditions for varying BER p

scheme and the fragment-based schemes are due to the change of segment/fragment sizes as p varies.

IV. Results for the case of unknown channels

The results in the previous section have been obtained under the assumption that the channel bit error rate was constant all the time and precisely known to the transmitter. In this section we present simulation results comparing the performance of the different schemes for a time-varying channel in which additionally the bit error rate is not known to the transmitter. The same simulation code has been used as in the previous section.

We consider the following setup. The channel varies its bit error rate (BER) according to a random process, in which the BER is not varied from bit to bit, but remains constant for a longer time (the *holding time*) which is chosen randomly from a sequence of iid exponential random variables. The mean of these exponential holding times is varied between 20000, 50000, 100000 and 200000 bit times (which for a 250 kBit/sec channel would correspond to average holding times of 80 ms, 200 ms, 400 ms and 800 ms). Whenever a holding time ends and a new one starts, a new value for the bit error rate is chosen. We consider two different cases: in the first case the bit error rate exponent is chosen according to a uniform random distribution from the interval $[-4, -1]$ and in the second case from $[-5, -2]$. The actual bit error rate is then ten to the power of the chosen exponent.

The further major parameters are chosen as follows. The deadline d (or its transmission budget) for each packet has again been chosen as $d = 11400$ bits, and the source wants to transmit packets of $s = 1024$ bits size during this time. The overall duration of a simulation is $\Delta = 250,000 \cdot d$ bits, i.e. within the available time at least 250,000 transactions can be performed. The transactions are performed back to back, i.e. when one ends the next one starts and the total number of transactions that can be performed within Δ is typically higher. The two performance measures considered in the simulation are:

- The percentage of successful transactions (which is called *average success probability*).
- The *channel goodput*, which is defined as the total number of user data bits transmitted successfully through the channel during time Δ (which increases by s bits for each successful transaction), divided by Δ.

The following schemes have been considered in the comparison: the classical scheme, the majority voting scheme, three variants of the segment-normal scheme, which always use a fixed segment size of 64, 32 or 16 bits, respectively (compare Table I), two variants of the segment-reduced scheme with fixed segment sizes of 32 and 16 bits, respectively, four variants of the segment-erasure scheme with fixed segment sizes of 128, 64, 32 and 16 bits, respectively and three versions of the fragment-based scheme with fixed fragment sizes of 128, 64 and 32 bits, respectively. Furthermore, for all three segment-based schemes and the fragment-based scheme we have devised a simple adaptive version, whichs operation we explain for the case of the segment-erasure scheme. The adaptation scheme uses a pre-defined list of segment sizes (here: $\{128, 64, 32, 16\}$) and one of these is always the *current segment size* (CSS). The CSS is varied after each transaction. If the previous transaction has failed, the CSS is set to the next smaller value, if there is one available. If the previous transaction was successful and its two predecessors have been successful as well, the CSS is set to the next larger value, if one is available. Stated differently: upon a failed transaction it reduces segment size immediately, whereas to increase the segment size three successful trials in a row must have occured. The adaptive version of the segment-erasure scheme uses the segment size set $\{128, 64, 32, 16\}$, for the segment-normal scheme the set $\{64, 32, 16\}$ is used and for the segment-reduced scheme the set is $\{32, 16\}$. For the fragment-based scheme we have used the same adaptation algorithm, which varies the fragment size from the set $\{512, 256, 128, 64, 32, 16\}$.

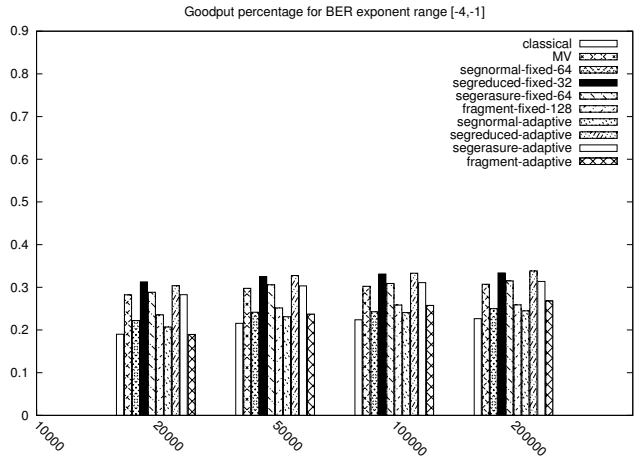

Fig. 3: Goodput for a BER exponent range of $[-4, -1]$

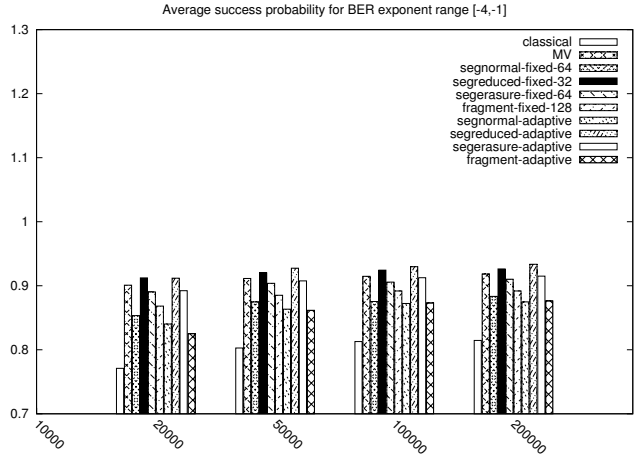

Fig. 4: Average success probability for a BER exponent range of $[-4, -1]$

The goodput and the average success probability for the BER exponent range $[-4, -1]$ are showns in Figures 3 and 4, respectively. Not all schemes are shown. For example, the fixed segnormal scheme with 64 bits segment size is consistently better than the other fixed segnormal schemes, so the latter are omitted. Similarly, the fixed segment-reduced scheme with a segment size of 32 is consistently better than the scheme with a segment size of 16, the fragment scheme with a fixed fragment size of 128 bits is consistently better than the other fragment schemes with

fixed fragment sizes. From the fixed segment-erasure scheme only the one with a fixed segment size of 64 bits is selected, which is consistently very close to the optimum. The following points are noteworthy:

- In terms of success probability all schemes are consistently and significantly better than the classical scheme. The fixed and adaptive segment-reduced scheme are consistently the best ones (with the adaptive scheme having slight advantages over the fixed scheme, the advantage increases with the average state holding time), followed by the majority voting scheme and then by the segment-erasure schemes (again, the adaptive versions of the segment-erasure scheme having slight advantages over the non-adaptive one). The segment-erasure schemes are consistently better than the fragment-based schemes, which in turn are consistently better than the segment-normal schemes. These findings are consistent with the findings reported in Section III-B reported for the success probability under perfect channel knowledge. Furthermore, it can be observed that each adaptive scheme improves its performance when the state holding time becomes larger. For the fragment-based scheme and the segment-normal scheme the adaptive version is inferior to the non-adaptive version.
- For the goodput it is not longer true that all schemes have a distinct advantage over the classical scheme. The adaptive segment-normal and fragment-based scheme gain nothing over the classical scheme for the smallest average holding time, for the larger holding times they achieve slight improvements over the classical scheme, but are outperformed by their fixed counterparts. On the other hand, again the segment-reduced schemes are consistently the best ones, with the adaptive version achieving approximately the same performance as the fixed one for increasing average holding times. Again, the segment-erasure schemes and the majority-voting scheme show the second-best level in performance, with only small differences among each other. They are significantly better than the fragment schemes. None of the adaptive schemes represents an improvement over their fixed-size versions.

The second set of results concerns the the BER exponent range $[-5, -2]$, which on average leads to better channels than the previous case. Consequently, all schemes achieve a success probability close to one, only the classical scheme achieves lower success probabilities in the range between 94.7% (for the smallest average holding time) and 96.6%. All other schemes are significantly better, with the adaptive fragment-based scheme being the poorest one (having

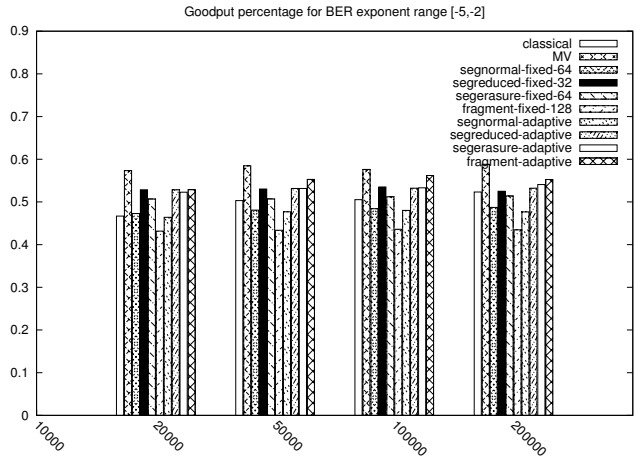

Fig. 5: Goodput for a BER exponent range of $[-5, -2]$

98.1% as the smallest average success probability). In terms of goodput (compare Figure 5) the majority-voting (having no extra overhead for error correction) scheme is consistently the best one, followed by the adaptive version of the fragment-based scheme, which significantly outperforms its fixed-size version.

V. Related work

In general, the segment-based schemes considered here are packet combining schemes [12], [13], [4]. Many packet combining schemes have been proposed and investigated, several of them, e.g. [14], [15], are designed for particular classes of coding schemes (here: soft-decoding Viterbi decoders). The segment-based schemes have been introduced under the name intermediate checksum schemes in [16]. To the best of the authors knowledge, this was the first paper discussing this class of schemes, except from [17], where the approach is briefly sketched but not followed anymore. In [18] a very similar approach 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 focus of the protocol is on efficiently streaming the blocks such that one frame can at the same time contain retransmissions of earlier failed blocks and new blocks from the same or the next message. 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 [19]. In [20] a scheme similar to the segment-reduced scheme (they actually speak of "sub-packet schemes") is considered, in which not only error-detection checksums are appended to each segment, but where also the whole resulting packet is encoded using a convolutional code. The authors of [20] then determine the segment-size that maximizes throughput. However, none of these works consider deadlines, and furthermore this paper is to the best of the authors knowledge the first paper combining segment-based schemes with (Luby-type) erasure codes.

VI. Conclusions

The segment-based schemes presented in this paper have been demonstrated for both static BSC channels and time-variable channels to achieve significant gains in terms of success probability and number of transmitted bits for more error-prone channels, while having only moderate buffer requirements at the receiver. Therefore, these schemes are a very attractive approach for memory-constrained stations like sensor network nodes. The segmentation-based approach is useful for not too small message sizes s which occur for example in re-programming operations in sensor networks.

Among the investigated segment-based schemes the segment-normal scheme has the worst performance, but does not critically rely on the feedback channel and furthermore requires only one frame construction at the transmitter side. On the other hand, the segment-reduced scheme shows the best performance over a wide range of bit error rates but its performance depends on the quality of the feedback channel and furthermore the transmitter must assemble a new frame for each trial. The segment-erasure scheme, which achieves better performance than the segment-normal scheme, does not depend on feedback and it can be configured (by choosing $m = n$) so that one frame assembly suffices.

Therefore, one of the natural next steps would be a synthesis of the segment-erasure and the segment-reduced scheme, in which the selection of segments at the transmitter is controlled by feedback obtained from the receiver. There are many further opportunities for future research, for example a more comprehensive exploitation of the design space of the segment-erasure codes (choices of code rate, m, etc.), incorporation of schemes for bit error rate estimation, adaptation of chosen segment sizes based on the re-transmission history of previous transactions,

an experimental evaluation, and many others.

VII. Appendix 1: Potentials of Markov chains

Be $(X_n)_{n\geq 0}$ a time-homogeneous Markov chain with discrete (i.e. finite or countably infinite) state space $\mathcal{S}$ and state-transition matrix $\mathbf{P}$. The state space is partitioned into *inner states* D and *boundary states* or *final states* ∂D so that $\mathcal{S} = D \cup \partial D$. Suppose that $\mathbf{c} = (c_i)_{i\in D}$ and $\mathbf{f} = (f_i)_{i\in\partial D}$ are non-negative vectors representing the costs c_i when the chain is in the inner state $i \in D$ and the costs f_i when the chain is in the boundary state $i \in \partial D$. Let the random variable T be the hitting time for the boundary: $T = \inf\{n \geq 0 : X_n \in \partial D\}$. Set

$$\phi_i = \mathrm{E}_i\left[\sum_{n<T} c(X_n) + f(X_T)\mathbf{1}_{T<\infty}\right]$$

Then ϕ_i is the expected total costs when the chain starts in state $X_0 = i$ and operates in the inner states D, each time incurring a cost c_i, until it reaches a final state in ∂D, incurring a final cost corresponding to the final state. The final costs are incurred only when the hitting time T is finite. Then the following holds [9, Theorem 4.2.3]:

- The potential $\phi = (\phi_i)_{i\in\mathcal{S}}$ satisfies:

$$\begin{cases} \phi = \mathbf{P}\cdot\phi + \mathbf{c} & : \text{ in } D \\ \phi = \mathbf{f} & : \text{ in } \partial D \end{cases} \tag{17}$$

- If $\Pr_i[T < \infty] = 1$ (i.e. the probability to hit the final states when the starting state is $X_0 = i$) for all i then Equation 17 has at most one bounded solution.

In other words, we are looking for a solution of the system of linear equations given in 17.

Appendix 2: Derivation of transition probabilities for the segment-erasure scheme

We start by computing the probability

$$T_{i,\delta} = \Pr[X_1 = i+\delta | X_0 = i]$$

(for $i \geq 0, m \geq \delta \geq 0, i+\delta \leq n$) that a receiver having already i different segments receives with the next trial (with PHY and MAC header being proper) exactly δ different segments that differ from all i segments it already has. Suppose that the receiver currently has the segments

$B_1 = \{b_{1,1}, b_{1,2}, \ldots, b_{1,i}\}$ and receives the segment set B_2. For simplicity, we denote the segments simply by the integers $1, 2, \ldots, n$, so that $B_1 \subset \{1, \ldots, n\}$ and $B_2 \subset \{1, \ldots, n\}$. Then, from the law of total probability:

$$T_{i,\delta} = \sum_{\substack{B_2 \subset \{1,\ldots,n\} \\ \delta \leq |B_2| \leq i+\delta}} \Pr\left[B_2 \text{ received}\right] \cdot \Pr\left[|B_2 \setminus B_1| = \delta | \, B_2 \text{ received}\right] \tag{18}$$

where the sum extends over all $B_2 \subset \{1, \ldots, n\}$ which have at least δ segments and at most $i + \delta$ segments.

We first consider $\Pr\left[B_2 \text{ received}\right]$. It is possible to receive $B_2 = \{b_{2,1}, b_{2,2}, \ldots, b_{2,r}\}$ (with $\delta \leq r \leq \delta + i$) when the transmitter has picked an m-element set B_2^* for transmission for which the following conditions hold: (i) $B_2 \subset B_2^*$ (ii) Exactly the segments in B_2 are received successfully and the segments in $B_2^* \setminus B_2$ are received in error. Therefore:

$$\Pr\left[B_2 \text{ received}\right] = \sum_{\substack{B_2^* \subset \{1,\ldots,n\} \\ |B_2^*| = m, B_2 \subset B_2^*}} \Pr\left[B_2^* \text{ transmitted}\right] \cdot \Pr\left[B_2 \text{ received} | \, B_2^* \text{ transmitted}\right] \tag{19}$$

From our assumptions, each of the $\binom{n}{m}$ m-element subsets of $\{1, \ldots, n\}$ is equiprobable, which implies that $\Pr\left[B_2^* \text{ transmitted}\right] = \binom{n}{m}^{-1}$. Suppose now that B_2^* with $B_2 \subset B_2^*$ has been sent and received with proper headers. When $P_S = (1-p)^{c+h}$ is the probability of correctly receiving a segment over a BSC channel and $Q_S = 1 - P_S$ is the probability of a reception failure (an erroneous segment is detected from the segment checksum and then thrown away, which translates into an erasure) then the probability of receiving $B_2 = \{b_{2,1}, b_{2,2}, \ldots, b_{2,r}\}$ when B_2^* is transmitted is given by $\Pr\left[B_2 \text{ received} | \, B_2^* \text{ transmitted}\right] = P_S^r \cdot Q_S^{m-r}$. Furthermore, the number of m-element subsets $B_2^* \subset \{1, 2, \ldots, n\}$ that contain $B_2 = \{b_{2,1}, b_{2,2}, \ldots, b_{2,r}\}$ as a subset can be obtained as follows: r out of the m places are fixed, for the remaining $m-r$ places a number of $n - r$ segments can be chosen, and the number of $m - r$-sized subsets of $n - r$ elements is

given by $\binom{n-r}{m-r}$. Putting everything together and simplifying we obtain:

$$\begin{aligned}&\Pr\left[B_2 = \{b_{2,1}, b_{2,2}, \ldots, b_{2,r}\} \text{ received}\right]\\ &= \quad P_S^r \cdot Q_S^{m-r} \cdot \frac{(n-r)! \cdot m!}{(m-r)! \cdot n!}\end{aligned} \tag{20}$$

To compute $T_{i,\delta}$ from Equation 18 one can observe that the conditional probability $\Pr\left[|B_2 \setminus B_1| = \delta | \, B_2 \text{ receiv}\right.$ is either zero or one and the computation boils down to determining the number of such sets B_2. Assume that $B_2 = \{b_{2,1}, b_{2,2}, \ldots, b_{2,r}\}$ is received with $\delta \leq r \leq \delta + i$. The number of r-element sets with elements from $\{1, \ldots, n\}$ which contain exactly δ elements that are not in $B_1 = \{b_{1,1}, b_{1,2}, \ldots, b_{1,i}\}$ is given by:

- the number of subsets of $\{b_{1,1}, b_{1,2}, \ldots, b_{1,i}\}$ with $r - \delta$ elements, which is given by $\binom{i}{r-\delta}$, times
- the number of subsets of $\{1, \ldots, n\} \setminus \{b_{1,1}, b_{1,2}, \ldots, b_{1,i}\}$ having δ elements, which is $\binom{n-i}{\delta}$.

Therefore we can express $T_{i,\delta}$ as:

$$T_{i,\delta} = \sum_{r=\delta}^{i+\delta} \binom{i}{r-\delta} \cdot \binom{n-i}{\delta} \cdot P_S^r \cdot Q_S^{m-r} \cdot \frac{(n-r)! \cdot m!}{(m-r)! \cdot n!} \tag{21}$$

We next turn our attention to the computation of the state transition probabilities $p_{i,j} = \Pr\left[X_1 = j | \, X_0 = i\right]$ of the Markov chain. It is clear that $p_{i,j} = 0$ for $j < i$, since the receivers number of different segments cannot decrease. Furthermore, it is also clear that $p_{k,k} = 1$, since state k is the (absorbing) success state in which the receiver has k different segments and can decode the message. Let us first suppose that $m \geq k$, i.e. the source node always transmits at least as many randomly chosen segments as are required for decoding. Then the state transition probabilities can be obtained as follows. When the receiver is in state $k-1$, it is required to receive the PHY and MAC headers successfully and to receive at least one additional segment it does not have so far. Therefore:

$$\begin{aligned}p_{k-1,k} &= P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P})\\ &\quad \cdot \left(T_{k-1,1} + T_{k-1,2} + \ldots + T_{k-1,\min\{m,n-k\}}\right)\end{aligned}$$

which holds since the events that the receiver receives 1, 2, …, $\min\{m, n-k\}$ segments that it does not yet have are mutually disjoint. For the diagonal element of state $k-1$ we simply have $p_{k-1,k-1} = 1 - p_{k-1,k}$. For all states $i \in \{0, \ldots, k-2\}$ we can distinguish three cases:

- For target states $j \in \{i+1, \ldots, k-1\}$ we must receive the PHY and MAC headers properly and exactly $j-i$ new segments, which happens with probability

$$p_{i,j} = P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P}) \cdot T_{i,j-i}$$

- For the target state $j = k$ we must receive the headers successfully and receive $k-i$, or $k-i+1$, or …, or $\min m, n-k$ new segments, therefore:

$$\begin{aligned} p_{i,k} = {} & P_{\text{PH}} \cdot P_{\text{MH}}(o_{M,P}) \\ & \cdot \left(T_{i,k-i} + T_{i,k-i+1} + \ldots + T_{i,\min\{m,n-k\}}\right) \end{aligned}$$

- The diagonal element $p_{i,i}$ is one minus the sum of all other transition probabilities.

For the other case $m < k$ the state transition probabilities can be obtained in a similar fashion.

REFERENCES

[1] A. Willig, K. Matheus, and A. Wolisz, "Wireless Technology in Industrial Networks," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1130–1151, June 2005.

[2] 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.

[3] S. Kallel, "Analysis of a type-II hybrid ARQ scheme with code combining," *IEEE Trans. on Communications*, vol. 38, no. 8, pp. 1133–1137, Aug. 1990.

[4] ——, "Complementary punctured convolutional (cpc) codes and their applications," *IEEE Trans. on Communications*, vol. 43, no. 6, pp. 2005–2009, June 1995.

[5] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp. 569–584, Feb. 2001.

[6] D. V. Sarwate, "Computation of Cyclic Redundandy Checks via Table Look-Up," *Communications of the ACM*, vol. 31, no. 8, pp. 1008–1013, Aug. 1988.

[7] A. Willig, "Memory-Efficient Segment-Based Packet-Combining Schemes in Face of Deadlines," in *Proc. 5th International Wireless Communications and Mobile Computing Conference (IWCMC'09)*, Leipzig, Germany, 2009, accepted for publication.

[8] ——, "Memory-efficient segment-based packet-combining schemes in face of deadlines (extended version)," Telecommunication Networks Group, Technical University Berlin, TKN Technical Report Series TKN-09-001, Jan. 2009.

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

[10] A. Willig, "Intermediate checksum schemes in the presence of deadlines," Telecommunication Networks Group, Technical University Berlin, TKN Technical Report Series TKN-08-002, Jan. 2008.

[11] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2000.

[12] P. S. Sindhu, "Retransmission Error Control with Memory," *IEEE Trans. on Communications*, vol. 25, no. 5, pp. 473–479, May 1977.

[13] A.-G. A. Dairaseh and C. W. Baum, "Methods for packet combining in HARQ systems over bursty channels," *MONET - Mobile Networks and Applications*, vol. 2, pp. 213–224, Oct. 1997.

[14] B. A. Harvey and S. B. Wicker, "Packet combining systems based on the Viterbi decoder," *IEEE Trans. on Communications*, vol. 42, no. 2, pp. 1544–1557, Feb. 1994.

[15] S. B. Wicker, "Adaptive rate error control through the use of diversity combining and majority-logic decoding in a hybrid-arq protocol," *IEEE Trans. on Communications*, vol. 39, no. 3, pp. 380–385, Mar. 1991.

[16] A. Willig, "Intermediate Checksums for Improving Goodput over Error-Prone Links," in *Proc. IEEE Vehicular Technology Conference (VTC), Fall 04*, Los Angeles, CA, Sept. 2004.

[17] 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.

[18] R. K. Ganti, P. Jayachandran, H. Luo, and T. F. Abdelzaher, "Datalink streaming in wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems (ACM SenSys)*, 2006, pp. 209–222.

[19] H. Dubois-Ferriere, D. Estrin, and M. Vetterli, "Packet combining in sensor networks," in *Proc. 3rd Intl. Conf. on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, Nov. 2005.

[20] Y. Zhou and J. Wang, "Optimum Subpacket Transmission for Hybrid ARQ Systems," *IEEE Trans. on Communications*, vol. 54, no. 5, pp. 934–942, 2006.

PLACE PHOTO HERE

Andreas Willig (M) is a senior researcher with the Telecommunication networks group (TKN) at the Technical University of Berlin, Germany since April 2005. From 2002 to 2005 he was an assistant professor with the Hasso-Plattner-Institue at University of Potsdam (Germany). He obtained the Dr.-Ing. degree in electrical engineering from Technical University Berlin (Germany) in 2002, and the diploma degree in computer science from University of Bremen (Germany) in 1994. His research interests include wireless networks, fieldbus and real-time systems, ad-hoc and sensor networks, all with specific focus on protocol design and performance aspects.