An earlier version of this paper appears in Proceedings of ACM/IEEE MobiCom 99. This paper is a revised and extended version and appears in the Wireless Networks Journal (ACM and Baltzer Science Publishers), Special issue: Selected papers from MobiCom '99.

Optimizing the End-to-End Performance of Reliable Flows over Wireless Links

Reiner Ludwig
Ericsson Research
Herzogenrath, Germany

Almudena Konrad, Anthony D. Joseph, Randy H. Katz
Computer Science Division
University of California at Berkeley

Abstract

We present the results of a performance evaluation of link layer error recovery over wireless links. Our analysis is based upon a case study of the circuit-switched data service implemented in GSM digital cellular networks. We collected a large set of block erasure traces in different radio environments, with both stationary and mobile end hosts. Our measurement-based approach gave us the unique opportunity to derive real-world models of the wireless link. We show that the throughput of the GSM circuit-switched data channel can be increased by using a larger (fixed) frame size for the reliable link layer protocol. This yields an improvement of up to 25 percent when the channel quality is good and still 18 percent under poor radio conditions. Our results also suggest that adaptive frame length control could further increase the channel throughput. In general, our case study shows that design alternatives that rely on pure end-to-end error recovery fail as a universal solution to optimize throughput when wireless links form parts of the end-to-end path. In many cases, it leads to decreased end-to-end throughput, an unfair load on a best-effort network, such as the Internet, and a waste of valuable radio resources (e.g., spectrum and transmission power). In fact, we show that link layer error recovery over wireless links is essential for reliable flows to avoid these problems.

1. Introduction

The Internet is evolving to become *the* communication medium of the future. It will not be long before the last circuit switch is taken out of service and virtually all people-to-people, people-to-machine, and machine-to-machine communication are carried in IP [44] packets end-to-end. The tremendous recent growth of the Internet in terms of connected hosts is only matched by the similar tremendous growth of cellular telephone subscribers. While most hosts on today's Internet are still wired, the next *big* wave of hosts has yet to hit the Internet. We believe that the predominant Internet access of the future will be wireless. Not only every cellular phone, but every *thing* that communicates will have: (1) an IP protocol stack and (2) a wireless network interface.

It is well known that the performance of reliable transport protocols such as TCP (Transmission Control Protocol) [1], [45] may degrade when the end-to-end path includes wireless links. This is due to non-congestion related packet losses on the wireless link causing a network-limited TCP sender to underestimate its share of the bottleneck link's bandwidth. However, related work has mostly focused on the problem that wireless links cause for the congestion control scheme used in most implementations of TCP. Employing a link layer error recovery scheme over the wireless link removes this problem. Furthermore, our previous work [32], [33] shows that, for the case of TCP and at least in some wireless networks - including the one we study in this paper - the potential problems that may result from competition between end-to-end and link layer error recovery do not exist. Motivated by this result, we study link layer enhancements that further increase application layer throughput while minimizing the consumption of precious radio resources like transmission power. The latter is of particular importance for battery-powered mobile devices. We then quantify the benefit of link layer error recovery by evaluating the performance of pure end-to-end error recovery for comparison purpose.

The key premise for our analysis is that we assume the model of a network-limited bulk data transfer based on a reliable end-to-end flow (e.g., TCP). This is a valid assumption given the concept of *flow-adaptive* wireless links introduced in [31]. A flow-adaptive implementation of a link layer error recovery scheme can perform the flow type differentiation required to identify reliable flows. This ensures that link layer error recovery does not interfere with delay-sensitive (usually unreliable) end-to-end flows (e.g., UDP (User Datagram Protocol) [43]). The attractiveness of link layer solutions over approaches that require access to the transport layer headers in the network (e.g., [3], [4], [5], [10], [11], [15], [22], [29], [34]), are their independence from transport (or higher) layer protocol semantics and the possibility of co-existence with any form of network layer encryption as proposed in [28].

The analysis presented in this paper is based upon a case study of the circuit-switched data service implemented in GSM (Global System for Mobile communications) digital cellular networks. Our measurement-based approach gave us the unique opportunity to derive models of the wireless link that capture the aggregate of real-world effects like noise, interference, fading, and shadowing. This is a key advantage as an unrealistic error model of the wireless channel can completely change the results of a performance analysis leading to non-optimal design decision. For wireless systems it is therefore particularly important that prototypes are developed early in the design process so that measurement-based performance studies can be performed. Our analysis approach also provided us with new insights into how the current system can be optimized, and suggested techniques that can be used to design future wireless links. The fact that GSM has been deployed globally and is in widespread use, highlights the relevance of our results.

The rest of this paper is organized as follows: Section 2 provides background on the circuit-switched data service implemented in GSM; Section 3 describes the measurement platform we developed to collect *block erasure traces*, and explains our analysis goals, assumptions, and the methodology we used for our trace-based analysis; Section 4 presents and discusses our measurement results; Section 5 discusses related work and relevant design considerations; and Section 6 closes with our conclusions and plans for future research.

2. Circuit-Switched Data in GSM

GSM implements several error control techniques, including adaptive power control, frequency hopping, Forward Error Correction (FEC), and interleaving. In addition, the Circuit-Switched Data (CSD) service provides an optional fully reliable link layer protocol called Radio Link Protocol. We briefly describe the latter three control schemes as implemented for GSM-CSD using Figure 1. More details can be found in [38].

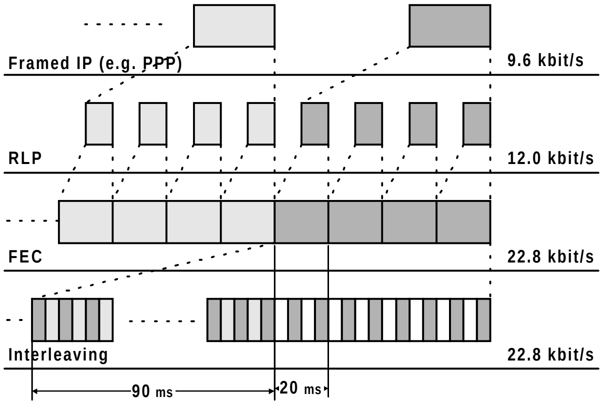

Figure 1: Error control in GSM Circuit-Switched Data.

GSM is a TDMA-based (Time Division Multiple Access) circuit-switched network. At call-setup time a mobile terminal is assigned a user data channel, defined as the tuple (carrier frequency number, slot number). The slot cycle time is 5 milliseconds on average, allowing 114 bits to be transmitted in each slot which yields a gross data rate of 22.8 kbit/s. The fundamental transmission unit in GSM is a *data block* (or simply *block*). The size of an FEC encoded data block is 456 bits (the payload of 4 slots). In GSM-CSD the size of an unencoded data block is 240 bits resulting in a data rate of 12 kbit/s (240 bits every 20 ms).

Interleaving is a technique that is used in combination with FEC to combat burst errors. Instead of transmitting a data block in four consecutive slots, it is divided into smaller fragments. Fragments from different data blocks are then interleaved before transmission. The interleaving scheme chosen for GSM-CSD, interleaves a single data block over 22 TDMA slots. The benefit is that a few of these smaller fragments can be completely corrupted, while the corresponding data block can still be reconstructed by the FEC decoder. The disadvantage of this large interleaving depth is that it introduces a significant one-way latency of approximately 90 ms. This high latency can have a significant negative effect on interactive protocols, as discussed in [31].[1]

The Radio Link Protocol (RLP) [17] is a full duplex HDLC-derived logical link layer protocol. RLP uses selective reject and checkpointing for error recovery. The RLP frame size is fixed at 240 bits aligned to the above mentioned FEC coder. RLP introduces an overhead of 48 bits per RLP frame yielding a user data rate of 9.6 kbit/s in the ideal case (no retransmissions)[2]. RLP transports user data as a transparent byte stream (i.e., RLP does not “know” about PPP frames or IP packets). It is important to note that, although RLP usually provides a fully reliable link, data loss can occur if the link is reset. This can have a severe impact on higher layer protocol performance as shown in [32].

3. Analysing Block Erasure Traces

In this section, we describe the measurement platform we developed to collect *block erasure traces*. We then explain our analysis goals, assumptions, and the methodology we used for our trace-based analysis. The measurement platform is basically the same as the one used in [32] to study the interactions between TCP and RLP.

3.1 What is a Block Erasure Trace?

In wireless networks that do not employ FEC, the error characteristics of the wireless channel over a certain period of time can be captured by a bit error trace. A bit error trace contains information about whether a particular bit was transmitted correctly or not. The average Bit Error Rate (BER) is commonly used to describe a bit error trace. The same approach can be applied to networks that *do* employ FEC, as in GSM, but on block level instead of on bit level. Hence, a block erasure trace contains information about whether a particular data block was transmitted correctly or not. Likewise, the average BLock Erasure Rate (BLER) is commonly used to describe a block erasure trace.

It is important to emphasize that the error characteristics we have measured are only valid for the particular FEC and interleaving scheme implemented in GSM-CSD (see Section 2). Nevertheless, this data service has been deployed globally and is in widespread use. As such, we believe that our results (see Section 4) provide useful insights into how the current system can be optimized, and also suggest techniques that can be used to design future wireless links.

3.2 Measurement Platform

The architecture of the measurement platform we have developed to collect the block erasure traces is depicted in Figure 2. A single-hop network running the Point-to-Point Protocol (PPP) [49] connects the mobile to a fixed host which terminates the circuit-switched GSM connection. Various tools can be used to generate traffic on the link (e.g., `ping` as described in [51]). In order to collect block erasure traces, we have ported the RLP protocol implementation of a commercially available GSM data PC-Card (Ericsson DC23) to BSDi3.0 UNIX. In addition, we developed a protocol monitor for RLP which we call RLPDUMP. It logs (among other RLP events) whether a received block could be correctly recovered by the FEC decoder or not. This is possible because every RLP frame corresponds to an FEC encoded data block (see Section 2). Thus, a received block had suffered an erasure whenever the corresponding RLP frame was received with a frame checksum error. We then generated bulk data traffic for a cer-

[1] Note, that voice is treated differently in GSM. Unencoded voice data blocks have a size of 260 bits and the interleaving depth is 8 slots.

[2] Note, that the transparent (not running RLP) GSM-CSD service introduces a wasteful overhead of modem control information that also reduces the user data rate to 9.6 kbit/s.

tain time and used RLPDUMP to captured the corresponding block erasure trace.

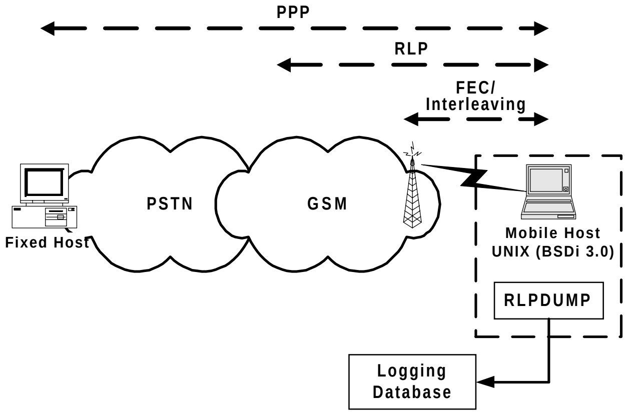

Figure 2: Current measurement platform.

At this stage of our work we have only performed measurements in commercially deployed GSM networks where the network-side of RLP was not accessible. This means that we could only collect downlink block erasure traces. Nevertheless, this allowed us to understand the GSM-CSD channel error characteristics to a degree that was sufficient enough for our analysis. We do not believe that additional uplink block erasure traces would have changed our conclusions.

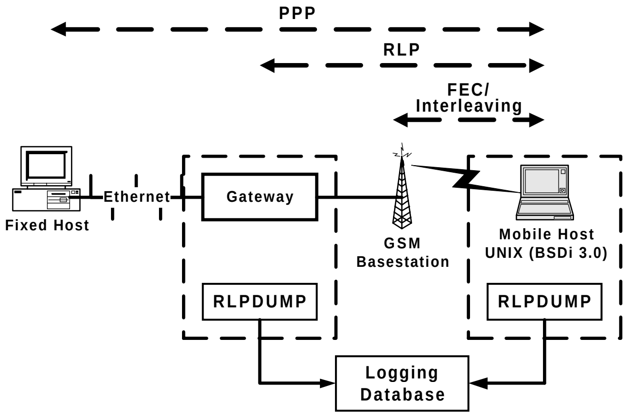

Figure 3: Future measurement platform.

Ultimately, we will use a stand-alone GSM basestation with a dedicated gateway that is being developed as a part of the ICEBERG project [52]. The gateway "translates" between circuit-switched (voice and data) and IP-based packet-switched traffic. As shown in Figure 3 we are currently enhancing this gateway to also terminate RLP and run RLPDUMP. With this platform we will then be able to also collect detailed uplink information and, e.g., trigger cell handovers in a controlled fashion.

3.3 Analysis Goals and Assumptions

Our goal is to evaluate the performance of the following two protocol design alternatives for reliable data transfer over a path that includes a wireless link:

- End-to-end error recovery complemented with link layer error recovery running over the wireless link.
- Pure end-to-end error recovery.

In general, "pure end-to-end" implies that no transport layer state is maintained in the network and that no assumptions about the existence or non-existence of dedicated link or network layer support are made. Nevertheless, throughout this paper, when we use the term "pure end-to-end error recovery", we implicitly refer to the case where the wireless link is *not* protected by link layer error recovery. We perform the evaluation of the two protocol design alternatives through a case study of the GSM-CSD wireless link. For TCP, our previous work [32], [33] shows that at least in some wireless networks - including GSM-CSD - the potential problems that may result from competition between end-to-end and link layer error recovery do not exist. Motivated by this result, we study enhancements to RLP that further increase application layer throughput while minimizing the consumption of precious radio resources like transmission power. The latter is of particular importance for battery-powered mobile devices. We then quantify the benefit of link layer error recovery by evaluating the performance of pure end-to-end error recovery (e.g., "standard" TCP [45] or "non-standard" TCP extensions as, e.g., studied in [48]) for comparison purpose.

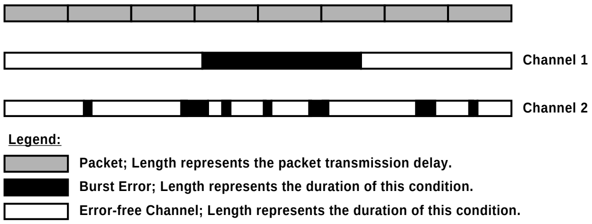
Channel 1
Channel 2
Legend:
Packet; Length represents the packet transmission delay.
Burst Error; Length represents the duration of this condition.
Error-free Channel; Length represents the duration of this condition.

Figure 4: Two different channel error characteristics.

The performance difference between the two protocol design alternatives mainly depends on the wireless channel's error characteristics versus the channel's packet[3] transmission delay. This is sketched in Figure 4 where "burst error" denotes time intervals during which data in transit is *entirely* corrupted. With respect to GSM-CSD a burst error corresponds to a series of back-to-back block erasures (see Section 3.1) where the channel is error-free before and after that series. A wireless channel's error characteristic is determined by the length of burst errors and the correlation between them, i.e., whether burst errors occur in clusters or more isolated. Link layer error recovery is less effective on wireless links where the length of burst errors is large compared to the packet transmission delay (see "Channel 1" in Figure 4). In this case pure end-to-end error recovery often yields better throughput results by saving link layer protocol overhead. Another case is sketched with "Channel 2" in Figure 4 where the length of burst errors is small compared to the packet transmission delay and burst errors often occur isolated (e.g., one burst error per packet transmission delay). In this case the link layer overhead is often more than amortized when the "right" frame size is chosen (see Section 4.2). Studying this tradeoff requires a realistic model of the wireless channel and motivates our measurement-based analysis approach further outlined in Section 3.4.

The key premise for our analysis is a model of a network-limited bulk data transfer based on a *reliable* end-to-end flow (e.g., TCP-based). In this case the path's bottleneck link limits the end-to-end throughput. Consequently, to make a throughput comparison

[3] Comprising the transport layer segment, the transport and network layer headers, and the packet framing overhead (e.g., PPP [49]) required on this channel. Note, that this does *not* include RLP overhead.

between the two above mentioned protocol design alternatives, we must assume that the GSM-CSD wireless link is the path's bottleneck link. A flow-adaptive implementation of RLP can perform the flow-type differentiation required to identify reliable flows [31]. This ensures that link layer error recovery does not interfere with delay-sensitive (usually unreliable) end-to-end flows (e.g., UDP-based). The requirements of applications that use reliable flows are simple: the application layer data object should be transferred as fast as possible *but* reliable, i.e. the transfer fails if the data object is corrupted when received by the destination application. This translates into similarly simple quality of service requirements for reliable flows: maximize throughput while the per packet delay is (almost) irrelevant[4].

We perform a best-case analysis which assumes that the bulk data transfer always fully utilizes the wireless bottleneck link. We use the term *utilization* as defined in [32] which states that a link is fully utilized if it never runs idle and never transmits a packet/frame which had already been successfully transmitted before. The latter can, e.g., happen in TCP which exhibits go-back-N behaviour after spurious timeouts [33]. Concerning link layer error recovery this implies (1) the use of a selective reject based protocol, like RLP (see Section 2); and (2) fully-persistent retransmissions (i.e., a large value for "maximum number of retransmissions" which is only reached when the link is considered to be disconnected) as opposed to a semi-persistent mode as proposed in [27]. It also requires the use of large enough windows to allow the transport/link layer sender to always fully utilize the link, i.e. to avoid so-called "stalled window" conditions [33]. The best-case assumption basically says that we ignore interactions with end-to-end congestion control schemes. For the case of TCP over RLP we have shown that this is valid [32]. For pure end-to-end error recovery, however, this is clearly an unrealistic assumption. Certain "patterns" of packet transmission errors causing false congestion signals will inevitably lead to a reduction of the transport layer send rate below the speed of the bottleneck link. Nevertheless, a best-case study indicates the theoretical maximum application layer throughput that pure end-to-end error recovery could possibly provide. Moreover, the best-case application layer throughput as defined here directly translates into radio resource consumption (e.g., spectrum and transmission power). For example, if transport layer sender A only provides half the throughput that sender B provides, it is using twice as much radio resources. Sender A would also have higher service costs in case the wireless access network implements volume-based charging.

Given these analysis goals, we were not interested in identifying those factors (e.g., noise, fading, interference, or shadowing) that caused measured block erasures. Rather, we were interested in the aggregate result (similar to the approach suggested in [40]). That is, we were interested in the above mentioned characteristics of block erasure traces required for the outlined performance evaluation. In particular we wanted to find answers to the following questions:

- Considering the non-adaptive FEC scheme implemented for GSM-CSD: Is the fixed frame size chosen for RLP optimal or would a larger frame size yield higher channel throughput?
- Considering adaptive frame length control schemes: How "fast" do channel error characteristics change in GSM-CSD? What is the margin of potential throughput improvement that adaptive frame length control could possibly yield?
- How does pure end-to-end error recovery perform as compared to complementing end-to-end with link layer error recovery (running an optimal frame size) given the same radio channel conditions?

3.4 Analysis Methodology

Altogether we have collected block erasure traces for over 500 minutes of "air-time". We distinguish between measurements where the mobile host was stationary versus mobile when driving in a car. All stationary measurements were taken in the exact same location. The following three categories of radio environments were chosen:

A. Stationary in an area with good receiver signal strength (3-4): 258 minutes.

B. Stationary in an area with poor receiver signal strength (1-2): 215 minutes.

C. Mobile in an area with mediocre receiver signal strength (2-4): 44 minutes.

The method we used to determine the receiver signal strength is rather primitive. We simply read the mobile phone's visual signal level indicator which has a range from 1-5. In the future, we will continuously log internal signal strength measurements from the mobile phone. That way we will then be able to correlate changing receiver signal strength with the block erasure traces.

Clearly, the size of an RLP frame does not need to match the size of an unencoded data block. Any multiple of the size of an unencoded data block would have been a valid design choice. In fact a multiple of 2 has been chosen for new RLP [18] in the next generation of the GSM-CSD service which uses a weaker FEC scheme [19]. The trade-off here is that larger frames introduce less overhead per frame, but an entire frame has to be retransmitted even if only a single data block incurs an erasure. Applying a technique, we call *retrace analysis*, we studied this trade-off using the large amount of block erasure traces we collected. Based on a given block erasure trace and a given bulk data transfer size, retrace analysis is a way to reverse-engineer the value of target metrics (e.g., channel throughput or number of retransmissions). Retrace analysis emulates RLP while assuming a particular fixed frame size and fixed per frame overhead. We then iterated the retrace analysis over a range of RLP frame sizes defined in terms multiples of the data block size. That way we could for example find the frame size that would have maximized the bulk data throughput for a particular block erasure trace. For the analysis presented in Section 4, we assume a per RLP frame overhead of 6 bytes for the regular header (see Section 2) plus 1 byte for each block in a frame larger than one block. The extra byte per block is needed for frame synchronisation for our planned implementation of a modified RLP using larger frames[5].

We used different block erasure traces for our analysis. One which we call *trace_A* is a concatenation of all block erasure traces we collected in environment *A* (see above). Likewise, *trace_B* and *trace_C* are the concatenations of all block erasure traces we col-

[4] In theory, it would not matter in a file transfer if the first packet reached the destination last. What matters is that the file transfer is completed in the shortest amount of time. In practice, e.g., transport layer receiver buffers required for packet re-sequencing place a limit on the maximum per packet delay that is tolerable without affecting performance. This limit is nevertheless low.

[5] One bit per block would have been sufficient to distinguish the beginning block in a frame but that would have made the implementation more complex.

lected in environment *B* and *C*, respectively. We then chose an appropriate bulk data size to cover the entire trace (e.g., for *trace_B* a size corresponding to a transmission time of 215 minutes was chosen). Once the retrace analysis had reached the end of a trace it wrapped around to its beginning. In addition, we wanted to understand the impact of error burstiness, i.e., the extent to which the distribution of block erasures within a trace influenced our results. For that purpose, we artificially generated three more "non-bursty" block erasure traces, *trace_A_even*, *trace_B_even* and *trace_C_even*, which had the same BLER as the corresponding real traces, but with an even block erasure distribution.

4. Measurement Results

In this section we provide the answers to the questions we put forward at the end of Section 3.3. We show that the throughput of the GSM-CSD channel can be improved by up to 25 percent by increasing the (fixed) RLP frame size. Our results also suggest that techniques like adaptive frame length control and adaptive FEC are worth further exploration for additional increases in channel throughput. Furthermore, we argue why in systems like GSM-CSD, pure end-to-end error recovery fails to optimize end-to-end performance.

4.1 Block Erasure Rates and Burstiness

Deriving the overall BLERs for *trace_A*, *trace_B* and *trace_C* (see Section 3.3) would have delivered little useful information. Instead, we also captured how "fast" the BLER changes over time in a given radio environment. We therefore divided each trace into one minute long *sub-traces* and treated each of those independently.

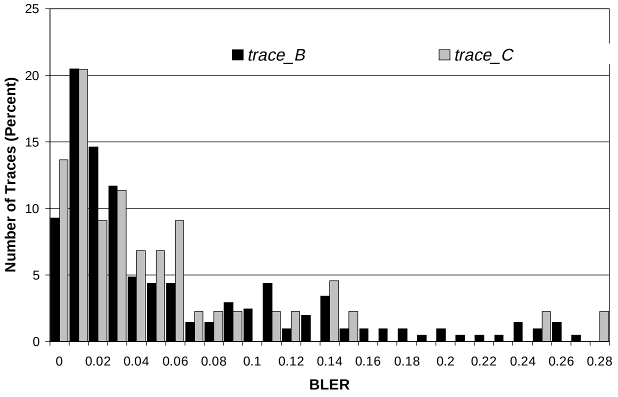

Figure 5: Measured BLERs.

Figure 5 summarizes the BLERs that we have determined in this manner. The BLERs for the sub-traces of *trace_A* are not shown because we found *trace_A* to be almost free of block erasures: over 96 percent of all sub-traces did not have a single block erasure (!) and the remaining ones had a BLER of less then 0.0025. This result shows how strongly the GSM-CSD channel is protected by FEC and interleaving, leaving little error recovery work for RLP. This is especially striking because radio environment *A* was not even ideal as it only provided a receiver signal strength of 3-4. Many radio environments often provide a maximum receiver signal strength of 5. This indicates that a weaker FEC scheme and/or a larger RLP frame size would increase the channel throughput in such radio environments. The results for *trace_B* and *trace_C* are similar but very different from the results for *trace_A*. In both of these environments, over 30 percent of all sub-traces had no single block erasure or a BLER of less then 0.01. But overall the BLERs vary considerably and can be as high as 0.28 (!). These large variations take place over time scales of one minute (the length of one sub-trace). This seems "slow" enough to make adaptive error control schemes applicable even within the same radio environment (e.g., environment *B*). This is an important result because otherwise such schemes would only be effective if the mobile user changed location to a different radio environment. The reason is that adaptive error control schemes can only adapt with a certain latency, which depends on the delay required to feedback channel state information. In our future work, we will study the potential of adaptive frame length control (e.g., proposed in [16] and [30]) as a technique to increase channel throughput. This decision is partly driven by our measurement-based analysis approach and the fact that we are not able to implement schemes like adaptive FEC in our testbed (see Section 3.2).

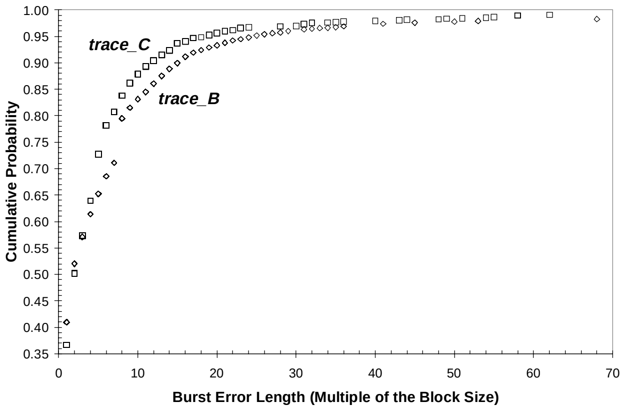

Figure 6: Burst error length distribution (block level).

Figure 6 shows the cumulative distribution function for the burst error lengths, i.e., the number of consecutive blocks that suffered an erasure, for *trace_B* and *trace_C*. There was no point in showing the distribution for *trace_A* as it was basically error-free. As seen in both graphs, over 50 percent of all burst errors are only 1 or 2 blocks long. It can also be seen that longer error bursts are more common when the mobile host is stationary, e.g., in *trace_B* less than 5 percent of all error bursts are larger than 26 blocks whereas in *trace_C* this number drops to 18. As discussed in Section 3.3, the problem with the model shown in Figure 6, is that it does not sufficiently describe the wireless channel's error characteristic. The model does not show whether the burst errors occurred in clusters or were isolated, i.e., the correlation between error bursts is not captured by this simple model. In the following section we show how the (fixed) frame size, which maximizes channel throughput, can be used as a metric to quantify this correlation.

4.2 Error Burstiness Allows Larger Frames

The results from the preceding section already suggest that in many GSM radio environments, a higher channel throughput could be achieved by increasing the RLP frame size. Those results also indicate that - given a non-adaptive FEC scheme - an optimal "on size fits all" RLP frame size cannot exist. Nevertheless, we wanted to determine the fixed RLP frame size that maximized channel throughput in the three radio environments *A*, *B*, and *C*. This is relevant because it indicates the margin of potential throughput improvement that adaptive frame length control could possibly yield. The implementation complexity of such techniques must be justified with substantial performance improvements. Thus, if the margin was too small, it would not be worthwhile to continue studying algorithms for adaptive frame length control in the current GSM-CSD. For that purpose, we performed the retrace analysis described in Section 3.3. The results are shown in Figure 7. As can be seen an optimal frame size of 1410 bytes would have yielded a throughput of 1423 bytes/s for *trace_A* and a frame size of 210 bytes would have maximized throughput to 1295 bytes/s for *trace_C*. The results for *trace_B* are so close to those of *trace_C* that we do not show them here. However, the gradual performance improvements in the case of *trace_A* rapidly decrease above a frame size of 210 bytes. A frame size of 210 bytes would still have yielded a throughput of 1392 bytes/s. This is important for our future work as it indicates that for an adaptive frame length control algorithm it would probably be sufficient to adapt the frame size in a range of about 30-210 bytes.

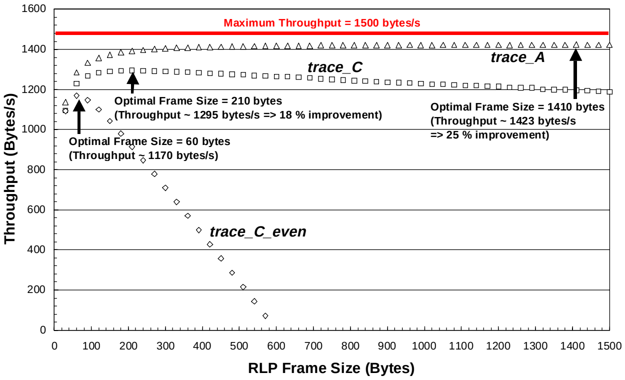

Figure 7: Throughput versus Frame Size.

Thus, a key result is that the (fixed) frame size chosen for RLP in the current GSM-CSD was an overly conservative design decision. Increasing it to 210 bytes would increase the channel throughput by at least 18 and up to 23 percent (!) depending on the radio environment[6]. This still leaves a (theoretical) margin of potential throughput improvement of 8-16 percent for adaptive frame length control, depending on the radio environment. We were not able to verify which studies have led to the decision to standardize an RLP frame size of 30 bytes [17]. However, if such studies had been carried out at all, our results clearly show that they must have been based on an unrealistic error model of the GSM-CSD radio channel. In any case our results highlight the importance of measurement-based analysis of protocol performance over wireless links.

Another less obvious, but nonetheless plausible result is that the error burstiness on the GSM-CSD channel allows for larger frame sizes than if block erasures were evenly distributed, i.e., not bursty. This effect can seen by comparing the graphs *trace_C* and *trace_C_even* (see definition in Section 3.3) in Figure 7. The retrace analysis for *trace_C_even* yields an optimal frame size of only 60 bytes (comparing *trace_B* and *trace_B_even* gives the same result). In fact one could view the quotient of the optimal frame sizes for an error trace (bit error trace or block erasure trace) and the corresponding "*_even*" trace as the *burst error factor*. The closer a trace's burst error factor is to 1 the less the corresponding channel exhibited error burstiness. Note, though, that the burst error factor also depends on the per frame overhead chosen for the retrace analysis. To eliminate this dependency one could base the definition of the burst error factor on a retrace analysis that assumes a per frame overhead of zero.

4.3 Problems of Pure End-to-End Error Recovery

Based on *trace_C*, we performed the best-case analysis described in Section 3.3 using TCP [45] as an example of a pure end-to-end error recovery protocol. For that purpose we repeated the retrace analysis assuming a per *MTU (Maximum Transmission Unit)* [51] overhead of 47 bytes[7]. The retrace analysis yields that the end-to-end throughput is maximized with an MTU size of 690 bytes. The reason for the difference compared to the retrace analysis of RLP is the larger overhead per transmission unit. The first row of Table 1 shows the result for commonly used MTU sizes. The second row shows the end-to-end throughput that is achieved when running RLP with a frame size of 210 bytes which provides a channel throughput of 1295 bytes/s (see Figure 7).

	MTU 296 bytes	MTU 576 bytes	MTU 1500 bytes
Pure End-to-End (No Header Compr.)	1151	1219	1196 4.9%
End-to-End with RLP (No Header Compr.)	5.2% 1094	2.4% 1191	1255
End-to-End with RLP (With Header Compr.)	7.6% 1239	3.8% 1265	7.4% 1284

Table 1: Application Layer Throughput in bytes/s.

As can be seen, pure end-to-end error recovery achieves a 2.4 and 5.2 percent higher best-case application layer throughput for MTU sizes of 576 and 296 bytes, respectively. This shows that pure end-to-end error recovery requires less radio resource consumption for these MTU sizes as discussed in Section 3.3. However, even when TCP-SACK [35] is used, it is unlikely that the advantage in end-to-end throughput would be achieved in practice. This is due to interference with the end-to-end congestion control scheme commonly implemented in TCP [1] as discussed in Section 3.3. The benefit of link layer error control becomes evident with larger MTU sizes (e.g., the commonly used 1500 bytes - see Table 1) and when TCP/IP header compression is used over the wireless link[8]. For pure end-to-end error recovery, TCP/IP header compression as defined in [14] and [24], are not an option. The reason is that [24] requires a reliable link and causes the loss of an entire window worth of packets for each packet lost *after* the compression point. While, the *twice* algorithm proposed in [14] is more robust, it will cause the same problem when 2 or more packets with compressed headers are lost back-to-back. However, this is a likely event for the GSM-CSD wireless link (if not protected by RLP) as shown in Figure 8. The cumulative distribution of the length of back-to-back packet losses

[6] For example, for *trace_A* the retrace analysis yields a throughput of 1392 bytes/s for a frame size of 210 bytes and a throughput of 1138 bytes/s for a frame size of 30 bytes/s. For *trace_B* and *trace_C* these frame sizes yield a throughput of 1295 bytes/s and 1096 bytes/s, respectively.

[7] 40 bytes for the TCP and IP headers and 7 bytes of PPP overhead.

[8] In this case we assume that the TCP/IP header is compressed to an average of 6 bytes. Although, compressed TCP/IP headers are typically 4 bytes long, a network-limited TCP connection will drop one packet per congestion avoidance cycle. This causes one packet to be sent with a full header (40 bytes), and 2 packets - after the packet loss and after the retransmission - to be sent with a compressed header of 7 bytes. Given the bandwidth-delay product of GSM-CSD link this leads to an average of about 6 bytes.

shows that **???** percent of all such losses have a length of 2 or larger. Alternatively, [14] also defines a "header request" mechanism which in fact is a link layer error recovery mechanism.

Figure 8: Burst error length distribution (packet level).

One could argue in favour of pure end-to-end error recovery by requiring the wireless link's MTU to be set to small values. Transport protocols like TCP could then use the *MSS option* (Maximum Segment Size) or *path MTU discovery* [51] to adapt the path's MTU accordingly. However, that will not work when the link's end points (e.g., the PPP peers) are not "aware" of the fact that the link includes a wireless segment as, e.g., in GSM-CSD (see Figure 2). Also, the path's MTU cannot be re-negotiated during a connection in current transport protocols[9]. Thus, potential advantages of adaptive packet/frame length control schemes could not be realized. Link layer error recovery, on the other hand, does not have these problems. It is independent of MTU sizes and also interworks well with IP header compression schemes. Future systems favour link layer error recovery even more. For once weaker FEC schemes are being deployed[10] which will most likely further decrease the throughput optimal frame size on wireless links such as GSM-CSD. Also, the next generation of the IP protocol [13] requires a minimum MTU of 1280 bytes and recommends an MTU of 1500 bytes or more on links such as GSM-CSD.

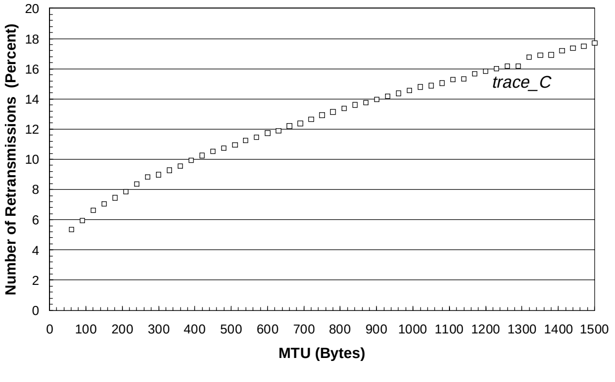

Figure 9: Number of end-to-end retransmissions.

Another shortcoming of pure end-to-end error recovery is that each retransmission will have to traverse the entire path. This is depicted in Figure 9 for *trace_C* showing the number of retransmissions (as fraction of the overall number of transmissions) that are required for a range of different MTU sizes. The commonly used MTU size of 1500 and 576 bytes would cause 18 and 12 percent retransmissions, respectively. Thus, such flows impose an unfair load upon a best-effort network, such as the Internet, and also upon shared wireless access links (e.g., 802.11 WLANs). Apart from fairness concerns, a higher fraction of retransmissions also decreases the end-to-end throughput if the corresponding packets had already traversed the bottleneck link regardless of where that is located in the path. This is a common situation when, e.g., data is downloaded from the Internet and the last-hop is an unreliable wireless link. End-to-end error recovery complemented with link layer error recovery running over the wireless link, on the other hand, does not require a single end-to-end retransmission in the case of GSM-CSD [32].

5. Discussion

The problem of "TCP over wireless links" has been a hot research topic for many years. The fact that it still is can, e.g., be seen from the fact that the IETF (Internet Engineering Task Force) has recently created the PILC (Performance Implications of Link Characteristics) working group in response to the emerging need for a "standard" solution. In this section, we provide a comprehensive state-of-the-art survey and discuss the pros and cons of existing approaches.

5.1 Restating the Problem: Congestion or Corruption?

Applications sharing a connection-less best-effort network need to respond to congestion to ensure network stability. Traditionally, congestion control has been implemented at the transport layer. [23] first described the fundamental algorithms that are most used in the Internet today [1], [54]. One of the key elements for any congestion control algorithm is the *congestion signal* that informs senders that congestion has or is about to occur. In this section we assume a sender-side implementation of transport layer congestion control, and if applicable also error control. The same discussion also applies to receiver-based implementations. One distinguishes between *explicit* congestion signals issued by the network and *implicit* congestion signals inferred from certain network behavior. Nevertheless, routers in today's Internet do not issue explicit congestion signals[11] although this might be implemented in the future [46]. Two approaches have been discussed for senders relying on an implicit congestion signal: delay-based and loss-based. Unfortunately, it is often not possible to draw sound conclusions from network delay measurements (see e.g. [7]). In particular it is difficult to find characteristic measures such as the path's minimum round trip time due to persistent congestion at the bottleneck link or because of route changes [41]. Consequently, "packet loss" is the only signal that senders can confidently use as an indication of congestion. It is implemented either as a direct [1] or an indirect trigger based on a perceived packet loss rate [54] to throttle the flow's send rate. We refer to such flows as being *loss responsive*. In this sense a TCP-based flow is a reliable loss responsive flow, whereas a "TCP-friendly" UDP-based flow is an unreliable loss responsive flow.

However, "packet loss" is not unambiguous. Packets can get lost because of packet drops due to a buffer overflow at the bottleneck link or because of packet corruption due to a transmission error. The

[9] Implementing such a mechanism would also be a poor design choice as optimizing a link's frame length is not an end-to-end issue.

[10] Weaker FEC schemes will be used in the new GSM-CSD service [19] and the upcoming GSM packet-switched data service [20].

[11] At least after the *source quench* [51] has been banned.

former indicates congestion, the latter does not. A sender is not able to discriminate among these events, because packet corruption usually leads to a frame checksum error and subsequent discard of the packet at the link layer. Hence, transmission errors beyond a certain rate inevitably lead to an underestimation of available bandwidth for loss responsive flows. As a consequence, applications can only fully utilize their share of bandwidth along the path if transmission errors are rare events. This explains why wireless links are often problematic: whereas transmission errors on today's wireline links can be safely neglected, this is not true for wireless links, especially when the end host is mobile.

The rate at which packet transmission errors occur for a given flow is called the *damage loss rate*. We can approximate an upper limit for the damage loss rate up to which the flow's send rate is insensitive. A network limited sender cyclically probes the path for more bandwidth. With the additive increase policy of one packet per round trip time [23] this leads to a single - in the ideal case - dropped packet at the end of each cycle. Thus, the reciprocal of the number of packets that are sent per cycle determines the *probing loss rate*[12]. This rate is different for every path, depending on its bandwidth/delay product (see e.g. [32]) and MTU[13]. Hence, a sender is insensitive to transmission errors as long as the damage loss rate stays below the probing loss rate. It is worth pointing out that [51] misinterprets [23] by stating that TCP's congestion avoidance algorithm assumes that packet loss caused by damage is much less than 1 percent. This is not correct: if the bandwidth/delay product is already exhausted with a few packets, the damage loss rate may be much higher than 1 percent without considerably affecting performance.

5.2 Existing Approaches

While we are not aware of any work that studies the problem of loss responsive flows over wireless links in general, the particular problem of TCP over wireless links has been investigated in several studies discussed in this section. We have categorized the proposed solutions as shown in Figure 10. Note that the dark shaded areas indicate whether a transport protocol or its implementation must be changed, or whether transport protocol dependent state has to be maintained in the network. The lightly shaded areas indicate changes required at the link layer. Conceptual design considerations that favour one or another solution are further discussed in Section 5.3. We do not discuss solutions that suggest protocols resulting in flows which are not loss-responsive, e.g. [12].

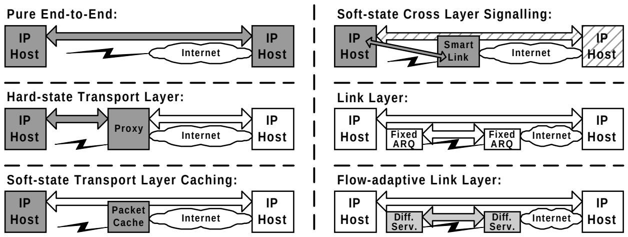

Figure 10: Approaches to solve "TCP over Wireless".

Pure end-to-end approaches do not maintain transport layer state in the network and make no assumptions about the existence or non-existence of dedicated link layer (e.g., error recovery)[14] or network layer (e.g., cell handover indications) support. This category includes (1) existing end-to-end protocols (e.g. TCP itself [1], [45]); (2) "non-standard" extensions of existing end-to-end protocols and/or their implementation (e.g., [25], [33], [35], [48]); and (3) new or not widely deployed end-to-end protocols (e.g., [12]). Adding the notion of selective acknowledgements (SACK) to TCP [35] is a way to deal with damage loss over unreliable wireless links [5], [48]. The advantage is that a sender can quickly recover from multiple lost packets in a single round trip time and that such an event is treated as *one* congestion signal instead of one signal for each lost packet. In case a particular packet must be retransmitted more than once, [48] proposes a further enhancement to the TCP sender assuming a SACK receiver. In [32] we recommend implementing the timestamp option [25] as a way for the TCP sender to more closely track the round-trip time. This yields a more accurate prediction used as a basis for the retransmission timer and can thereby avoid spurious timeouts. *TCP-Eifel* proposed in [33] uses the timestamp option to eliminate the *retransmission ambiguity problem* [26]. It thereby avoids duplicate retransmissions caused by TCP's go-back-N behavior after spurious timeouts. In addition, it avoids an unnecessary reduction of the flow's send rate by restoring the TCP sender's congestion window after spurious timeouts and after spurious fast retransmits. The latter happens in case of packet reorderings beyond the *DUPACK threshold* (see [51]).

Hard-state transport layer approaches encompass all forms of "splitting" by running a separate transport layer connection over the wireless link. The concept was initially proposed in [3], and has been used in other work including transit satellite links [22]. Any protocol can be chosen for the wireless link, e.g., [10] and [29] suggest combining splitting with a link layer approach. Some split solutions [3], [22], [29] violate the end-to-end semantics of TCP's error control scheme. They do this by allowing the network-based proxy to acknowledge data before it has reached the TCP receiver. The solution proposed in [10] implements splitting while maintaining these end-to-end semantics. The major benefit of hard-state transport layer solutions is that the end-to-end flow is shielded from damage loss on the wireless link, and the flow can fully utilize its share of bandwidth over the entire path. The concept of splitting lends itself well to non-TCP flows.

The Snoop protocol developed in [4] implements "TCP-aware" local error recovery. As discussed further below, we avoid calling this *link layer* error recovery as we associate a different meaning with that term. Variations of the Snoop protocol were studied in [5]. Its advantage over split solutions is that the network state is soft, i.e., it is not crucial for the end-to-end connection. One problem is that it can only be applied to the those edges of a path that are free of congestion. The reason is the suppression of duplicate acknowledgements (DUPACKs), which filters out a congestion signal, and a proposed negative acknowledgement (NACK) scheme. When sending *to* the mobile host, packets dropped at a bottleneck link between the wireless link and the mobile host are mistaken for damage loss by the TCP-aware cache. The congestion signal (the three DUPACKs) is not propagated back to the sender. For packets sent *from* the mobile host, the NACK scheme causes a problem. If the wireless link itself (or any other link between the mobile host and the wireless link) becomes the bottleneck, packets lost due to congestion[15] cannot be discriminated from those lost due to damage. A

[12] In [23] the number of packets sent per cycle is called the window equilibration length and is approximated as $W^2/3$ where W is the window size at the end of a cycle. More detail can e.g. be found in [36].

[13] As a consequence, it is impossible to "know" the probing loss rate at any link layer along the path.

[14] Despite this definition, in the preceding sections we have used the term "pure end-to-end error recovery" implicitly referring to the case where the wireless link is *not* protected by link layer error recovery.

NACK is sent in either case, and the sender again relies on external means to get the congestion signal (e.g., the *source quench* [51]). Consequently, the suppression of DUPACKs and the NACK scheme violate the end-to-end semantics of the congestion control scheme found in most implementations of TCP.

Soft-state cross layer approaches make the flow's sender aware of the wireless link. This is achieved by having the link layer (or network layer in the case of Mobile-IP [42]) inform the transport layer sender about specific events so that it can adapt accordingly. The solution proposed in [15] uses ICMP (Internet Control Message Protocol) [51] to signal all active receivers that the link is in a bad state. The receiver reflects the signal to the sender using a dedicated TCP option field. In the network that was studied in [15], the reverse path did not traverse the "problem link". [5] proposes an explicit loss notification, which the link layer piggy-backs onto a TCP acknowledgement (ACK) as a TCP option to inform the sender that a particular packet was lost due to damage[16]. This solution, however, has the same problem as the above mentioned NACK scheme. [10] focuses on the problem of frequent and long disconnections. In case of disconnections a transport layer proxy issues TCP ACKs which shrink the advertised window to zero. This forces the TCP sender into *persist mode* [51]. In this mode the TCP sender will not suffer from timeouts nor from exponential back-off of the retransmission timer value. [11] and [34] focus on the problem of data loss or delay caused by cell handovers. Both solutions are based on the deployment of [42] and suggest informing the transport layer sender about a cell handover to trigger, e.g., the fast retransmit algorithm [11].

Link layer error control schemes aim at hiding the artifacts of the wireless link to higher layer flows. The techniques include adaptive forward error correction, interleaving, adaptive power control, and link layer error recovery protocols [2], [17], [18], [20], [27], [39]. Some wireless networks use none of those (e.g., early commercially available 802.11 WLANs), while others implement combinations, e.g., GSM digital cellular networks (see Section 2). Note that none of the variations of the Snoop protocol discussed in [5] as "link layer solutions" are considered here. The basic difference being that link layer solutions as used in this context are not tied into the semantics of any higher layer protocol. Link layer error recovery can yield excellent TCP bulk data throughput without interfering with end-to-end error recovery [6][16][32][37]. The key advantage is that local knowledge about the continuously changing channel error characteristics can be exploited to optimize error control efficiency as shown in Section 4. The second advantage is that it does not require any changes to the IP-based protocol stacks. The drawback is that in state-of-the-art networks link layer error control schemes are applied irrespective of the QoS (Quality of Service) requirements of individual flows sharing the link. A flow that is best served with persistent link layer error recovery cannot share the link with a delay-sensitive flow intolerable of delays introduced by link layer retransmissions. On the other hand, an application might be able to tolerate higher loss rates in return for higher available bit rates than the link's forward error correction scheme provides.

Flow-adaptive link layer solutions remove the drawback of "pure" link layer schemes mentioned in the preceding paragraph. The key idea is to adapt the link layer error control schemes to the individual QoS requirements of each flow sharing the link. The flows' QoS requirements are derived (only) from the IP headers - mainly the proposed differentiated service field [8] but also any other field - at the link layer. This may be viewed as a violation of the design principle of "protocol layering", but has the advantage that implementations of network-layer/link-layer interfaces do not have to be changed. However, changing the latter would allow for a "protocol layering" conformant implementation. This concept was introduced in [31], which developed a coarse grained differentiation between reliable (TCP, ICMP) and unreliable (UDP) flows. In that study the protocol identifier field in the IP header is used to choose whether or not to run link layer error recovery; avoiding it for UDP-based flows which is assumed to carry delay-sensitive data. However, this solution is limited as not every UDP-based flow is delay-sensitive as some application layer protocols build end-to-end reliability on top of UDP (see e.g., [21], [50], [55]). Therefore, [33] proposes to encode the "reliability" QoS requirement in the IP header's differentiated service field.

5.3 Design Considerations

In this section we discuss design considerations we believe are relevant when solving the problem of loss-responsive flows over wireless links. We use these guidelines to assess the approaches presented in Section 5.2.

Error Control Performance

Error control performance is the strongest argument in favour of link layer error control schemes. In Section 4 we show that link layer error recovery over wireless links is essential for reliable flows to optimize end-to-end performance (throughput and fairness) while minimizing radio resource consumption. We believe that a similar line of argumentation applies to unreliable but delay-sensitive flows. One challenge here is to find, e.g., the optimal amount of channel coding required to achieve a target range of user data bandwidths versus residual loss rates. Implementing an optimal solution from the end points of a path is impossible; even if knowledge about the ever changing error characteristics of each wireless link in the path was available. Therefore, **we promote flow-adaptive implementations of link layer error control schemes**. Flow-adaptive wireless links [33] are what the end-to-end argument calls "an incomplete version of the function provided by the communication system [that] may be useful as a performance enhancement". We believe that carrying communication-related QoS requirements as part of the flow's headers, as proposed in [8], and adapting lower layer functions such as error control accordingly, advances the discussion provided in section 2.3 of [47]. Flow-adaptive implementations of link layer error control schemes allow optimization of those schemes with respect to the flows' QoS requirements. For example, as discussed in Section 3.3, channel throughput can be aggressively maximized for a *reliable* flow as the per packet delay is less important.

Transport layer error recovery algorithms, such as the *Eifel* algorithm [33] (see also Section 5.2), further increase end-to-end error recovery efficiency. Despite the deployment concerns described below, **we promote the deployment of the Eifel algorithm** for TCP in particular, and in general for any reliable transport protocol that implements congestion control similar to [1]. It yields the most efficient error control by avoiding duplicate retransmissions. In addition, it provides higher end-to-end throughput (compared to "standard" TCP) over paths that exhibit frequent packet re-orderings beyond the DUPACK threshold and/or large delay variations[17]

[15] In [32] these effects were measured where packets got dropped locally at the mobile host because of congestion at the first-hop wireless link.

[16] This requires that the IP *and* the TCP checksum be re-computed.

(high enough to frequently cause spurious timeouts). In addition, we argue that an adaptive transport layer retransmission timer should not be tuned to prevent all spurious timeouts. In some wireless networks spurious timeouts will be unavoidable, anyway. A transport layer retransmission timer which is too conservative has a negative impact on end-to-end performance whenever the sender has to resort to a (long) timeout to recover a lost packet. This affects interactive applications but also bulk data transfers as soon the receiver's receive buffer is exhausted to absorb any further out-of-order packets. As a result the sender is blocked from sending any further packets. Instead, we believe that an adaptive transport layer retransmission timer should be "reasonably" conservative while a sender should be able to detect spurious timeouts and react appropriately by using the Eifel algorithm.

Given a reliable wireless link, transport layer selective acknowledgements [35] have nothing to add apart from improving error recovery efficiency in case of burst packet loss caused by congestion. Nevertheless, many legacy networks do *not* provide reliable wireless links. Thus, for "back-wards compatibility" selective acknowledgements are an attractive alternative.

Semi-Reliable versus Fully-Reliable Link layer Error Recovery

There has been debate [16], [27], [32], [53] about how persistent link layer error recovery should be implemented. For link layer implementations that do not provide differentiated error control but treat all flows the same, i.e., that are not flow-adaptive, the answer is straight forward. In this case retransmission persistency must be low to not cause interference with delay-sensitive flows. However, for senders (in this context we omit the prefix "link layer") that are capable of discriminating reliable from delay-sensitive flows the question about how to treat *reliable* flows remains. In the following we assume a reliable flow (defined in Section 3.3). The options are to implement either semi-reliable or fully-reliable link layer error recovery. A *semi-reliable sender* gives up after a few retransmission attempts, discards the corresponding frame, and resumes transmission with the next frame. This introduces *retransmission delay* on the order of a few 100 milliseconds [27] or in more persistent implementations on the order of a few seconds [17]. A *fully-reliable sender*, on the other hand, does not lose any of a flow's packets even over long link outages, up to some conservative termination condition[18]. An upper limit for such a condition is the *MSL (Maximum Segment Lifetime)* [45] of 2 minutes[19] which also serves as an upper bound for the reassembly timeout after IP fragmentation [9]. We are not aware of the existence of such a reliable link layer protocol implementation, although [17] can be configured to attempt 128 retransmissions which corresponds to about 25-50 seconds depending on the GSM-CSD implementation.

The end-to-end argument [47] tells us that it is not worth the effort to implement "perfect" reliability at the link layer. Yet, our design should eliminate non-congestion related packet loss to avoid the problems outlined in Section 5.1. Implementing semi-reliable link layer error recovery is always a compromise that avoids this conflict by emphasizing end-to-end error recovery. However, this approach has some fundamental problems. First, the sender has no way to decide *when* to "give up" and discard the packet to, e.g., stay within the bounds of TCP's retransmission timer. Optimizing this solution, however, requires knowledge of the path's round trip time, which cannot be known at the link layer (unless it was carried in the IP header). Secondly, a semi-reliable sender requires a channel with strong FEC in order to keep the rate of false congestion signals due to non-congestion related packet discards low. Hence, the channel throughput cannot be maximized as discussed above in the context of error control efficiency. Together with the non-data-preserving property of semi-reliable link layer error recovery, this cannot yield optimal end-to-end performance as discussed in Section 3.3. Another fundamental problem occurs in case of temporary link outages, e.g., when a user roams into (and back out of) an area without wireless connectivity. In this case all of the flow's unacknowledged packets will eventually be discarded by the semi-reliable sender. This will often cause an idle wait for a possibly backed-off transport layer retransmission timer to expire. If, on the other hand, packets were still queued at the wireless link, the end-to-end flow of data could be re-started immediately after the link has become available again. Therefore, **we promote the implementation of fully-reliable link layer error recovery for reliable flows** as it has none of these problems and guarantees that any loss[20] at the link is due to congestion. This is exactly the right signal to give to the sender of a loss-responsive reliable flow. In case of temporary link outages, this will most likely cause a spurious timeout which in turn forces a go-back-N behavior in TCP but (a) that's still better than the idle wait and (b) can be avoided with the Eifel algorithm.

General Purpose vs. Dedicated Solutions

We believe that it is a wrong design decision to make the network, transport or any higher layer protocol, aware of mobility (cell handovers) [11], [34] or aware of wireless links [5], [15]. A wireless network must hide the error characteristics of wireless links, while supporting seamless mobility[21]. Developers of existing and future networking protocols should be able to abstract from these purely local issues. We believe that it is also a wrong design decision to make link layer protocols aware of higher layer protocol semantics [4] or to install protocol-dependent gateways [3], [4], [5], [10], [15], [22], [29]. This would require upgrading for every new or changed higher layer protocol, adding to the deployment problems mentioned before. Also, the link layer solutions proposed in [2], [17], [18], [20], [27], [39] have the problem of not being general purpose solutions as mentioned in Section 5.2 with respect to the undifferentiated use of error recovery. Flow-adaptive wireless links, on the other hand, are truly general purpose as they are independent from transport (or higher) layer protocol semantics while offering differentiated error control.

[17] For example, over paths that include a *fully-reliable* wireless link which provides highly intermittent connectivity. Despite the conservative retransmission timer implemented in TCP [33], spurious timeouts cannot be avoided in such an environment unless dedicated transport layer support is implemented in the network, as proposed in [10].

[18] Note, that this has nothing to do with queue management techniques. Packets that are dropped by the network layer according to simple drop-tail or a more advanced active queue management scheme (e.g., [9]) will never be handed to the link layer.

[19] In theory, additional fully-reliable links could exist "further down" the path. Thus, a more conservative upper limit is to divide the MSL by the value of the *TTL (Time To Live)* [51] field in the IP header.

[20] Apart from the more unlikely events of link layer error detection failures.

[21] A wireless access network which provides seamless mobility must "look like" *one* single-hop link.

Deployment

This concerns the required effort to deploy a particular solution, the incentives for the involved players to do so, but also the interworking with other network elements and protocols. Solutions that require changes to transport layer protocols, or implementations thereof, rely on a large scale effort to be incorporated into operating system software of wireless hosts and/or wireless network gateways (see dark shaded boxes in Figure 10). Pure end-to-end solutions have the additional drawback that they require upgrading the large base of existing web servers to become effective. Solutions that require access to the transport layer headers in the network (e.g., [3], [4], [5], [10], [11], [15], [22], [29], [34]) fail when network layer encryption [28] spans the gateway[22], and hard-state solutions further complicate cell handover. Deployment is also a concern for flow-adaptive link layer solutions. The problem is that applications today do not explicitly include their QoS requirements in their flows' headers. Thus, making flow-adaptive link layer implementations viable requires further standardization, adoption, and deployment of the differentiated services framework [8].

6. Conclusion and Future Work

In this paper, we presented the results of a performance evaluation of link layer error recovery over wireless links. Our analysis is based upon a case study of RLP as an example of a reliable link layer protocol implemented in GSM digital cellular networks. The study leverages of the large set of block erasure traces we collected in different radio environments, with both stationary and mobile end hosts. Our measurement-based approach gave us the unique opportunity to derive models of the wireless link that capture the aggregate of real-world effects like noise, interference, fading, and shadowing. The key premise for our analysis is a model of network-limited bulk data transfer based on a reliable end-to-end flow (e.g., a TCP-based flow).

For the case of GSM, we show that the throughput of the circuit-switched data channel can be increased by using a larger (fixed) RLP frame size. This yields an improvement of up to 25 percent when the channel quality is good and still 18 percent under poor radio conditions. Larger frame sizes are made possible due to the channel's error burstiness, a quantity we define as the *burst error factor*. Our results also suggest that techniques such as adaptive frame length control and/or adaptive FEC are worth further exploration as a basis for further increasing channel throughput in GSM. This is a topic for our future research, as we plan to implement a measurement-based adaptive frame length control scheme in our testbed.

In general, our case study shows that design alternatives that rely on pure end-to-end error recovery fail as a universal solution to optimize throughput when wireless links form parts of the end-to-end path. The fundamental problem is that the path's MTU is often too large to yield efficient error recovery, and that the path's end points are not capable of dynamically adapting their MTU to changing local error characteristics on (possibly multiple) wireless links. In many cases, this will lead to decreased end-to-end throughput, an unfair load on a best-effort network, such as the Internet, and a waste of valuable radio resources (e.g., spectrum and transmission power). In fact, we show that link layer error recovery over wireless links is essential for reliable flows to avoid these problems. Also, our results highlight the importance of measurement-based analysis in wireless networks where protocol performance is highly dependent on the radio error characteristics.

We conclude the paper by providing a comprehensive state-of-the-art survey and discuss the pros and cons of existing approaches. We argue that it is possible to design link layer error control protocols that can flexibly satisfy varying QoS requirements of flows sharing the link. Towards this end we promote the implementation of flow-adaptive link layers. Concerning reliable flows, such as TCP-based flows, we argue in favour of fully-reliable link layer error recovery. As an optional but complementing transport layer mechanism for reliable loss-responsive flows we promote the use of the Eifel algorithm. We believe that the combination of these three mechanisms solve the problem of "reliable flows over wireless links" in the best possible way. The major advantages being the independence from transport (or higher) layer protocol semantics and the possibility of co-existence with any form of network layer encryption. Beyond that we currently experiment with implementations of TCP where the sender retransmits not only 12 times [51] and then closes the connection but until the *application* decides to close the connection.

In our future work we will focus on flow-adaptive link layers that can optimize the available error control schemes for the QoS requirements of unreliable but delay-sensitive flows. This will most likely require much more explicit information to be provided in the IP header than it is the case today.

Acknowledgments

We would like to thank Keith Sklower for helping us port the RLP code to UNIX and for lots of fruitful discussions. We thank Mikael Degermark for deepening our insights into IP header compression. We thank Bela Rathonyi, Phil Karn, and the anonymous reviewers for comments on earlier versions of this paper.

[22] Unfortunately, this is also true for transport layer header compression schemes.

References

[1] Allman, M., Paxson, V., Stevens, W., *TCP Congestion Control*, RFC 2581, April 1999.

[2] Ayanoglu, E., Paul, S., LaPorta, T. F., Sabnani, K. K., Gitlin, R. D., A*IRMAIL: A link-layer protocol for wireless networks*, Wireless Networks, Vol. 1, No. 1, pp. 47-60, February 1995.

[3] Bakre, A., Badrinath, B. R., *I-TCP: Indirect TCP for Mobile Hosts*, In Proceedings of ICDCS 95, May 1995.

[4] Balakrishnan, H., Seshan, S., Katz, R. H., Improving reliable transport and handoff performance in cellular wireless networks, Wireless Networks, Vol. 1, No. 4, pp. 469-481, December 1995.

[5] Balakrishnan, H., Padmanabhan, V., Seshan, S., Katz, R. H., *A Comparison of Mechanisms for Improving TCP Performance over Wireless Links*, In Proceedings of ACM SIGCOMM 96.

[6] Bhagwat, P., Bhattacharya, P., Krishna, A., Tripathi, S. K., *Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs*, Wireless Networks, Vol. 3, No. 1, pp. 91-102, 1997.

[7] Biaz, S., Vaidya, N. H., *Is the Round-trip Time Correlated with the Number of Packets in Flight?*, Computer Science Dept., Texas A&M University, Technical Report 99-006, March 1999.

[8] Blake, S. et al., *An Architecture for Differentiated Services*, RFC 2475, December 1998.

[9] Braden, R., Requirements for Internet Hosts - Communication Layers, RFC 1122, October 1989.

[10] Brown, K., Singh, S., *M-TCP: TCP for Mobile Cellular Networks*, Computer Communications Review, 27(5), October 1997.

[11] Cáceres, R., Iftode, L., *Imroving the Performance of Reliable Transport Protocols in Mobile Computing Environments*, IEEE JSAC, Vol. 13, No. 5, pp. 850-857, June 1995.

[12] Clark, D. D., Lambert, M. L., Zhang, L., *NETBLT: A bulk data transfer protocol*, In Proceedings of ACM SIGCOMM 87.

[13] Deering, S., Hinden, R., *Internet Protocol, Version 6 (IPv6) Specification*, RFC 2460, December 1998.

[14] Degermark, M., Nordgren, B., Pink, S., *IP Header Compression*, RFC 2507, February 1999.

[15] Durst, R. C., Miller, G. J., Travis, E. J., *TCP Extensions for Space Communications*, In Proceedings of ACM MOBICOM 96.

[16] Eckhardt, D. A., Steenkiste, P., *Improving Wireless LAN Performance via Adaptive Local Error Control*, In Proceedings of IEEE ICNP 98.

[17] ETSI, *Radio Link Protocol for data and telematic services on the Mobile Station - Base Station System (MS-BSS) interface and the Base Station System - Mobile Switching Center (BSS-MSC) interface*, GSM Specification 04.22, Version 5.0.0, December 1995.

[18] ETSI, *Digital cellular communications system (Phase 2+); Radio Link Protocol for data and telematic services on the Mobile Station - Base Station System (MS-BSS) interface and the Base Station System - Mobile Switching Center (BSS-MSC) interface*, GSM Specification 04.22, Version 6.1.0, November 1998.

[19] ETSI, *Digital cellular communications system (Phase 2+); Rate adaption on the Mobile Station - Base Station System (MS - BSS) Interface*, GSM Specification 04.21, Version 7.0.0, October 1998.

[20] ETSI, *Digital cellular communications system (Phase 2+); General Packet Radio Service (GPRS); Mobile Station (MS) Base Station System (BSS) interface; Radio Link Control / Medium Access Control (RLC/MAC) protocol*, GSM Specification 04.60, Version 6.1.0, August 1998.

[21] Floyd, S., Jacobson, V., Liu, C., McCanne, S., and Zhang, L., *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing*, IEEE/ACM Transactions on Networking, December 1997, Volume 5, Number 6, pp. 784-803.

[22] Henderson, T. R., Katz, R. H., *Transport Protocols for Internet-Compatible Satellite Networks*, To appear in IEEE JSAC in 1999.

[23] Jacobson, V., *Congestion Avoidance and Control*, In Proceedings of ACM SIGCOMM 88.

[24] Jacobson, V., *Compressing TCP/IP Headers for Low-Speed Serial Links*, RFC 1144, February 1990.

[25] Jacobson, V., Braden, R., Borman, D., *TCP Extensions for High Performance*, RFC 1323, May 1992.

[26] Karn, P., Partridge, C., *Improving Round-Trip Time Estimates in Reliable Transport Protocols*, In Proceedings of ACM SIGCOMM 87.

[27] Karn, P., *The Qualcomm CDMA Digital Cellular System*, In Proceedings of the USENIX Mobile and Location-Independent Computing Symposium, USENIX Association, August 1993.

[28] Kent, S., Atkinson, R., *Security Architecture for the Internet Protocol*, RFC 2401, November 1998.

[29] Kojo, M., Raatikainen, K., Liljeberg, M., Kiiskinen, J., Alanko, T., *An Efficient Transport Service for Slow Wireless Telephone Links*, IEEE JSAC, Vol. 15, No. 7, pp. 1337-1348, September1997.

[30] Lettieri, P., Srivastava, M. B., *Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency*, In Proceedings of IEEE INFOCOM 98.

[31] Ludwig, R., Rathonyi, B., *Link Layer Enhancements for TCP/IP over GSM*, In Proceedings of IEEE INFOCOM 99.

[32] Ludwig, R., Rathonyi, B., Konrad, A., Oden, K., Joseph, A., *Multi-Layer Tracing of TCP over a Reliable Wireless Link*, In Proceedings of ACM SIGMETRICS 99.

[33] Ludwig, R., *A Case for Flow-Adaptive Wireless Links*, Technical Report UCB//CSD-99-1053, University of California at Berkeley, May 1999.

[34] Manzoni, P., Ghosal, D., Serazzi, G., *Impact of Mobility on TCP/IP: An Integrated Performance Study*, IEEE JSAC, Vol. 13, No. 5, pp. 858-867, June 1995.

[35] Mathis, M., Mahdavi, J., Floyd, S., Romanow A., *TCP Selective Acknowledgement Options*, RFC 2018, October 1996.

[36] Mathis, M., Semke, J., Mahdavi, J., Ott, T., *The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm*, Computer Communications Review, 27(3), July 1997.

[37] Meyer, M., *TCP Performance over GPRS*, In Proceedings of IEEE WCNC 99.

[38] Mouly, M., Pautet, M.-B., *The GSM System for Mobile Communications*, Cell & Sys, France 1992.

[39] Nanda, S., Ejzak, R., Doshi, B. T., *A Retransmission Scheme for Circuit-Mode Data on Wireless Links*, IEEE JSAC, Vol. 12, No. 8, October 1994.

[40] Noble, B. D., Satyanarayanan, M., Nguyen, G. T., Katz, R. H., *Trace-Based Mobile Network Emulation*, In Proceedings of ACM SIGCOMM 97.

[41] Paxson, V., *End-to-End Routing Behavior in the Internet*, IEEE/ACM Transactions on Networking, Vol.5, No.5, pp. 601-615, October 1997.

[42] Perkins, C., *IP Mobility Support*, RFC 2002, October 1996.

[43] Postel, J., User Datagram Protocol, RFC 768, August 1980.

[44] Postel, J., Internet Protocol, RFC 791, September 1981.

[45] Postel, J., Transmission Control Protocol, RFC793, September 1981.

[46] Ramakrishnan, K. K., Floyd, S., *A Proposal to add Explicit Congestion Notification (ECN) to IP*, Internet Draft, Work in progress, January 1999.

[47] Saltzer, J. H., Reed, D. P., Clark, D. D., *End-To-End Arguments in System Design*, ACM Transactions on Computer Systems, Vol. 2, No. 4, November 1984.

[48] Samaraweera, N. K. G., Fairhurst, G., *Reinforcement of TCP Error Recovery for Wireless Communication*, Computer Communications Review, 28(2), April 1998.

[49] Simpson, W., *The Point-to-Point Protocol*, RFC 1661, July 1994.

[50] Srinivasan, R., *RPC: Remote Procedure Call Protocol Specification Version 2*, RFC 1831, August 1995.

[51] Stevens, W. R., *TCP/IP Illustrated, Volume 1 (The Protocols)*, Addison Wesley, November 1994.

[52] The ICEBERG Project, CS Division, EECS Department, University of California at Berkeley,
http://iceberg.cs.berkeley.edu/.

[53] The Mailing List of the PILC Working Group within the IETF,
http://pilc.grc.nasa.gov/pilc/list/archive/

[54] The TCP-Friendly Website, http://www.psc.edu/networking/tcp_friendly.html.

[55] Walsh, D., et al., *The overview of the sun network file system*, In Proceedings of Winter USENIX, January 1985.