To appear in ACM/Baltzer Wireless Networks Journal (Special issue: Selected papers from ACM/IEEE 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

Pure end-to-end error recovery fails as a general solution to optimize throughput when wireless links form parts of the end-to-end path. It can lead to decreased end-to-end throughput, an unfair load on best-effort networks, and a waste of valuable radio resources. Link layer error recovery over wireless links is essential for reliable flows to avoid these problems. We demonstrate this through an analysis of a large set of block erasure traces measured in different real-world radio environments, with both stationary and mobile hosts. Our analysis is based on a case study of the circuit-switched data service implemented in GSM. We show that the throughput on this wireless 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 18 percent even under poor radio conditions. Our results suggest that adaptive frame length control could further increase the channel throughput. Finally, we discuss link and transport layer error control mechanisms and their interactions with end-to-end congestion control schemes. For reliable flows, we argue in favor of highly persistent error recovery and lossless handover schemes implemented at the link layer.

1. Introduction

The Internet is evolving to become *the* communication medium of the future. It will not be long before virtually all people-to-people, people-to-machine, and machine-to-machine communication are carried end-to-end in IP (Internet Protocol) [39] packets. The recent tremendous 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) [40] 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 TCP sender to underestimate its share of bandwidth along the path. However, related work has mostly focused on the problem that wireless links cause for the congestion control scheme used in most implementations of TCP [1]. Employing a link layer error recovery scheme over the wireless link removes this problem. Furthermore, for TCP our previous work shows that problems resulting from competition between end-to-end and link layer error recovery do not exist. In [31] we show this for the wireless network examined in this paper. For other wireless networks, related work [5], [14], [35] comes to the same conclusion. In [32] we provide a general analysis of this subject quantifying the high degree of conservativeness implemented in TCP's retransmission timer [49].

Motivated by this result, we study the impact of link layer frame sizes on application layer throughput and the consumption of radio resources. We then quantify the benefit of link layer error recovery by comparing it against the performance of pure end-to-end error recovery.

The key premise for our analysis is that we assume a model of a network-limited bulk data transfer based on a reliable flow (e.g., based on TCP). In addition, we assume a *flow-adaptive* link layer implementation [30], [32]. A flow-adaptive link layer error recovery scheme distinguishes among unreliable and reliable flows to control its retransmission persistency. This ensures that link layer error recovery does not interfere with delay-sensitive (usually unreliable) flows (e.g., based on UDP (User Datagram Protocol) [38]). The advantages of this solution over approaches that require access to transport layer headers in the network (e.g., [2], [3], [4], [9], [13], [21], [28]), are

- its independence from transport (or higher) layer protocol semantics making it a "non-TCP-specific" solution,
- the possibility of co-existence with any form of network layer encryption as proposed in [27], and
- the fact that no per-flow state needs to be maintained in the network making this solution more scalable.

The concept of flow-adaptive link layer implementations was first introduced in [30]. There we proposed to use the protocol identifier field in the IP header for the purpose of distinguishing among flow types. This solution is limited, because not every UDP-based flow is delay-sensitive as some application layer protocols build end-to-end reliability on top of UDP [20], [45], [48]. Therefore, we propose in [32] to encode the "reliability" QoS (Quality of Service) requirement in the IP header's differentiated service field [6].

The analysis presented in this paper is based on a case study of the circuit-switched data service implemented in GSM (Global System for Mobile communications). Our measurement-based approach

gave us the unique opportunity to capture the aggregate of real-world effects such as noise, interference, fading, and shadowing. This is a key advantage as unrealistic assumptions about the error characteristics of a wireless channel can completely change the results of a performance analysis and lead to non-optimal design decisions. 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 carried out.

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 the goals, assumptions, and the methodology of our trace-based analysis; Section 4 presents and discusses our measurement results; Section 5 discusses error control performance from an end-to-end systems view; 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 service (CSD) provides an optional reliable link layer protocol called Radio Link Protocol. Using Figure 1, we briefly describe the latter three control schemes as implemented for the GSM-CSD. More details can be found in [36].

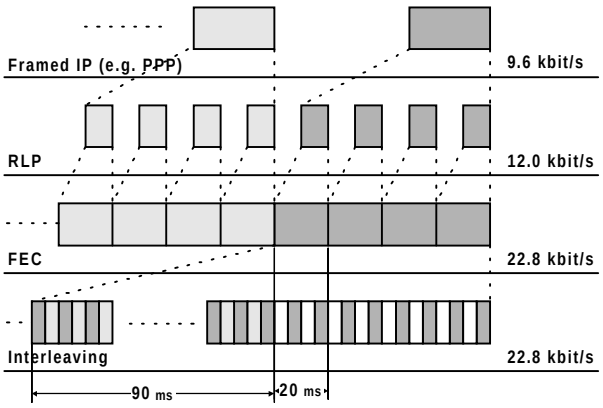

Figure 1: Error control in GSM Circuit-Switched Data.

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. This allows 114 bits to be transmitted in each slot, yielding 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) [17].

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 [19]. 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 deep interleaving is that it introduces a significant one-way latency of approximately 90 ms[1]. This high latency can have a significant adverse effect on interactive protocols [30].

The Radio Link Protocol (RLP) [15], [16] is a full duplex 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 IP packets). RLP loses data when the link is reset, e.g., after a maximum number of retransmissions of a single frame has been reached. Although, this can have a severe impact on higher layer protocol performance, it rarely happens under "normal" radio conditions [31].

3. Analysing Block Erasure Traces

In this section, we describe the measurement platform we developed to collect *block erasure traces*. We have used this measurement platform in [31] to study the interactions between TCP and RLP. We then explain the goals, assumptions, and the methodology of our trace-based analysis.

3.1 What is a Block Erasure Trace?

The error characteristics of a 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. The average Bit Error Rate (BER) is the first-order metric commonly used to describe the trace. The same approach can be applied on block level instead of on bit level. Hence, a block erasure trace contains information about whether a particular data block was transmitted correctly and the BLock Erasure Rate (BLER) denotes the average rate at which block erasures occur in the trace.

The error characteristics we have measured are only valid for the particular FEC and interleaving scheme implemented in GSM-CSD (see Section 2). Nevertheless, we believe that the results presented in Section 4 provide new insights into how this widely deployed system can be optimized and suggests techniques that can be used to design future wireless links. For example, the GSM packet-switched data service implements a similar FEC scheme [19].

3.2 Measurement Platform

Our measurement platform is depicted in Figure 2. A single-hop network running the Point-to-Point Protocol (PPP) [44] 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 [46]). To collect block erasure traces, we have ported the RLP protocol implementation of a commercially available GSM data PC-Card 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. This is possible because every RLP frame corresponds to an FEC encoded data block (see Section 2). Thus, a received block suffered an erasure whenever the corresponding RLP frame has a frame checksum

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.

error. We then generated bulk data traffic and used RLPDUMP to capture the corresponding block erasure trace.

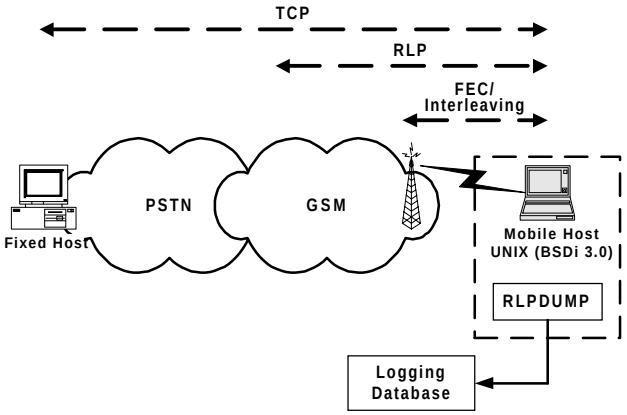

Figure 2: The measurement platform.

We have only performed measurements in commercially deployed GSM networks, where the network-side of RLP was not accessible. Thus, 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 for our analysis. Additional uplink block erasure traces would not have changed our conclusions, assuming similar channel error characteristics on both up- and downlink.

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 flow-adaptive 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 are made about the existence of dedicated support from the link layer (e.g., error recovery) or the network layer (e.g., cell handover indications, as proposed in [10]). Nevertheless, throughout this paper, when we use the term "pure end-to-end error recovery" we imply that the wireless link is *not* protected by link layer error recovery.

We perform the evaluation of the two alternatives through a case study of the GSM-CSD wireless link. First, we investigate the impact of changing the (fixed) RLP frame size on application layer throughput and the consumption of radio resources (e.g., spectrum and transmission power). We then quantify the benefits of link layer error recovery by comparing it against the performance of pure end-to-end error recovery, as proposed in [43].

The performance difference between the two protocol design alternatives depends on the wireless channel's time varying error characteristics versus the channel's transmission delay for an IP packet. This is sketched in Figure 3, where "burst error" denotes time intervals during which data in transit is corrupted to the extent that it cannot be recovered at the destination. With respect to GSM-CSD, a burst error corresponds to a series of back-to-back block erasures where the channel is error-free before and after that series. A wireless channel's error characteristic can be described by the length of burst errors and their correlation expressing the degree of clustering. 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 3). In this case, pure end-to-end error recovery often yields higher throughput results by saving link layer protocol overhead. Another case is sketched with "Channel 2" in Figure 3 where the length of burst errors is small compared to the packet transmission delay and the burst errors often occur isolated. In this case, the link layer overhead is likely to be amortized when the "right" frame size is chosen. Studying this trade-off requires a realistic characterisation of the wireless channel and motivates our measurement-based analysis approach further outlined in Section 3.4.

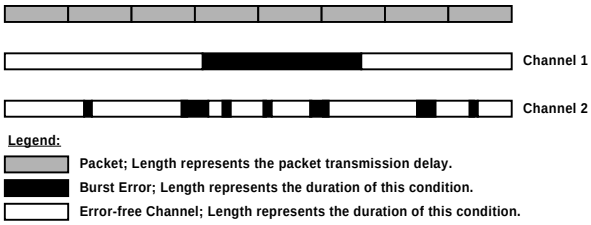

Figure 3: Two different channel error characteristics.

The key premise for our analysis is a model of a network-limited bulk data transfer based on a reliable flow (e.g., TCP-based). A *flow* is a single instance of an application-to-application flow of packets which is identified by source address, source port, destination address, destination port and protocol identifier [6]. The end-to-end throughput provided by a *network-limited* flow is limited by the path's bottleneck link. Consequently, to compare throughput among the two alternatives we assume that the GSM-CSD wireless link is the path's bottleneck. The requirements of applications using *reliable flows* are simple: transfer the application layer data object as fast as possible *but* reliably, i.e., the transfer fails if the data object is corrupted when received by the destination. This translates into similarly simple QoS requirements for reliable flows: maximize throughput while the per packet delay is (almost) irrelevant[3].

We perform a best-case analysis that assumes that the bulk data transfer always fully utilizes the wireless bottleneck link. We use the term *utilization* as defined in [31]: a link is fully utilized if it never runs idle and never transmits a packet/frame which had already been *successfully* transmitted. The latter can happen in TCP, which exhibits go-back-N behavior after spurious timeouts [33]. Concerning link layer error recovery, this implies (1) the use of a selective reject based protocol, like RLP; and (2) fully-persistent retransmissions (i.e., a large value for "maximum number of retransmissions" only reached when the link effectively becomes disconnected). It also requires the use of large enough windows to allow the link layer sender to always fully utilize the link. This avoids the *stalled window* condition [32]. Concerning a network-limited, TCP-based flow and because of the congestion control policies required by [1], this implies that the maximum number of packets that the flow may queue at the bottleneck link before a packet is dropped exceeds the pipe capacity. The *pipe capacity* is

3. In theory, it would not matter in a file transfer if the first packet reached the destination last. What usually 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 and whether a delayed packet triggers a congestion signal place a limit on the maximum per packet delay that is tolerable without affecting performance. This limit is nevertheless low.

the minimum amount of data a flow needs to have in transit to fully utilize its share of bandwidth at the bottleneck link. The best-case assumption ignores interactions with end-to-end congestion control schemes that may lead to an under-utilization of the bottleneck link. For TCP over RLP this is valid [31]. For pure end-to-end error recovery, however, this is unrealistic, as discussed in Section 4.3 and Section 5. Nevertheless, a best-case study indicates the theoretical maximum application layer throughput that pure end-to-end error recovery could provide. Moreover, the best-case application layer throughput, as defined here, directly translates into radio resource consumption. For example, if transport layer sender A only achieves half the throughput that sender B achieves, then sender A is using twice as much radio resources. This may happen if sender A has to rely on pure end-to-end error recovery, and has to retransmit packets of which only a small fraction of the corresponding original transmission was corrupted on the unreliable wireless link[4].

3.4 Analysis Methodology

Following the approach suggested in [37], we are not interested in identifying those physical link factors that caused measured block erasures. Rather, we are interested in the aggregate result captured by block erasure traces. We have collected block erasure traces for over 500 minutes of "air-time" and distinguish between measurements where the host was stationary versus mobile when driving in a car. All stationary measurements were taken in the exact same location. We were not able to log internal receiver signal strength measurements from the mobile phone to correlate them with the block erasure traces. Instead, we read the mobile phone's visual signal level indicator ranging from 1-5. The following three 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.

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 the new version of RLP [16] in the next generation of GSM-CSD, which also uses weaker FEC [17]. Larger frames introduce less relative overhead per frame, but an entire frame has to be retransmitted even if only a single data block incurs an erasure. Applying our technique of *retrace analysis*, we study this trade-off using the collected block erasure traces. Based on a given block erasure trace and a given bulk data transfer size, retrace analysis reverse-engineers the value of target metrics (e.g., channel throughput or number of retransmissions). It emulates RLP while assuming a particular fixed frame size and fixed per frame overhead. We then iterate the retrace analysis over a range of RLP frame sizes, defined as multiples of the data block size. We can thereby find the frame size that maximizes the bulk data throughput for a particular block erasure trace.

We use different block erasure traces for our analysis. *Trace_A* is a concatenation of all block erasure traces we collected in environment *A*. Likewise, *trace_B* and *trace_C* are concatenations of all block erasure traces we collected in environment *B* and *C*, respectively. We then choose 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 reaches the end of a trace, it wraps around to its beginning. In addition, we investigate the impact of error burstiness, i.e., the extent to which the distribution of block erasures within a trace influences 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*. These have the same BLER as the corresponding real traces, but with an even block erasure distribution, i.e., those traces have single and isolated block erasures with a constant distance from each other.

4. Measurement Results

In this section, 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 increasing channel throughput. Furthermore, we argue why pure end-to-end error recovery fails to optimize end-to-end performance on this wireless link.

4.1 Block Erasure Rates and Burstiness

Deriving the overall BLERs for *trace_A*, *trace_B* and *trace_C* would have delivered little useful information. Instead, we also capture how "fast" the BLER changes over time in a given radio environment. We therefore divide each trace into one minute *sub-traces* and treat each of those independently.

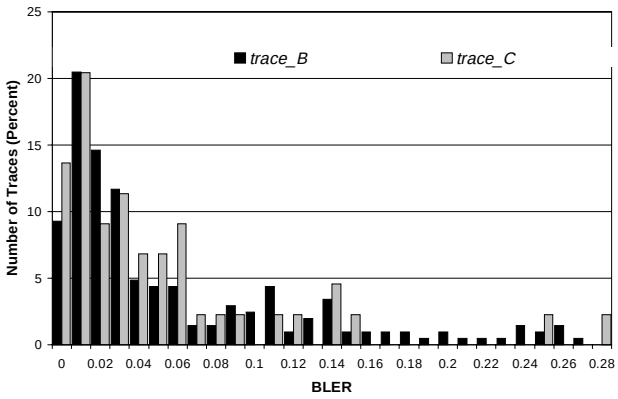

Figure 4: Measured BLERs.

Figure 4 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 do not have a single block erasure and the remaining ones have 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 far from ideal, as it only provided a receiver signal strength of 3-4. Many radio environments provide a maximum receiver signal strength of 5. This suggests that a weaker FEC scheme and/or a larger RLP frame size would increase channel throughput. The results for *trace_B* and *trace_C* are similar but different from the results for *trace_A*. In these, over 30 percent of all sub-traces have no single block erasure

4. If only a single byte of a 1500 bytes packet gets corrupted during transmission over an unreliable link, then still the entire packet has to be retransmitted.

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, which corresponds to 3000 RLP frames. This is "slow" enough to make adaptive error control schemes applicable even within the same radio environment. This is important 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 only adapt with a certain latency, which depends on the delay required to feedback channel state information. In future work, we will study the potential of adaptive frame length control (e.g., proposed in [14] and [29]) to increase channel throughput. This decision is partly driven by our measurement-based analysis approach and the fact that we are currently not able to implement schemes like adaptive FEC (e.g., as standardized in [18], [19]) in our measurement platform.

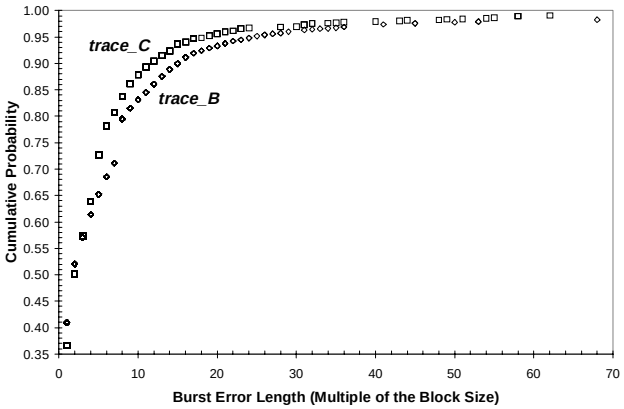

Figure 5: Burst error length distribution.

Figure 5 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 this for *trace_A*, as it was basically error-free. Over 50 percent of burst errors are only 1 or 2 blocks long. 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. This comparison is valid as the BLERs of both traces are of the same order. As discussed in Section 3.3, the distributions shown in Figure 5 do not sufficiently describe the wireless channel's error characteristic. They do not show whether the burst errors occur in clusters or are isolated, i.e., the correlation between error bursts is not captured. In the following section, we show how the (fixed) frame size that maximizes channel throughput can be used to quantify this correlation.

4.2 Error Burstiness Allows for Larger Frames

In this section, we determine the fixed RLP frame size that maximizes channel throughput in the radio environments *A*, *B*, and *C*. This also indicates maximum throughput improvement that adaptive frame length control could yield. The implementation complexity of such techniques must be justified with substantial performance improvements. Thus, if the margin for improvements was too small, it would not be worth studying algorithms for adaptive frame length control in GSM.

We perform the retrace analysis described in Section 3.3 leading to the results shown in Figure 6. An optimal frame size of 1410 bytes yields a throughput of 1423 bytes/s for *trace_A* and a frame size of 210 bytes maximizes throughput to 1295 bytes/s for *trace_C*. The results for *trace_B* are close to those of *trace_C*. 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 still yields a throughput of 1392 bytes/s. This indicates that for an adaptive frame length control algorithm, it would be sufficient to adapt the frame size in a range of about 30-210 bytes.

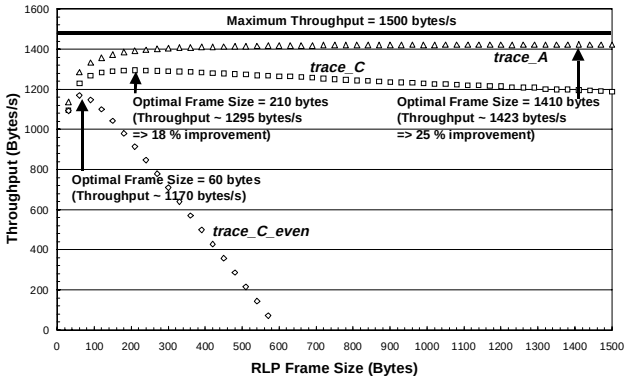

Figure 6: Throughput versus frame size.

A key result of our analysis is that the (fixed) frame size chosen for RLP was overly conservative. Increasing it to 210 bytes improves the channel throughput by at least 18 and up to 23 percent depending on the radio environment[5]. 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 led to the decision to standardize an RLP frame size of 30 bytes [15]. However, our results show that they must have been based on an unrealistic error model of the GSM-CSD radio channel. This highlights the importance of measurement-based analysis of protocol performance over wireless links.

Another result is that the error burstiness on the GSM-CSD channel allows for larger frame sizes than if block erasures are not bursty. This effect can seen by comparing the graphs *trace_C* and *trace_C_even* in Figure 6. The retrace analysis for *trace_C_even* yields an optimal frame size of only 60 bytes (*trace_B* and *trace_B_even* are similar). One could view the quotient of the optimal frame size 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, 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 perform the best-case analysis described in Section 3.3 using TCP [40] as an example of a pure end-to-end error recovery protocol. For that purpose we repeat the retrace analysis assuming a per *MTU (Maximum Transmission Unit)* [46] overhead of 47 bytes (20 bytes TCP header, 20 bytes IP header, and 7 bytes

5. 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_C*, these frame sizes yield a throughput of 1295 bytes/s and 1096 bytes/s, respectively.

of PPP overhead). The retrace analysis shows that the end-to-end throughput is maximized with an MTU size of 690 bytes. The reason for the difference with 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, providing a channel throughput of 1295 bytes/s (see Figure 6).

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

Table 1: Application layer throughput in bytes/s.

Pure end-to-end error recovery achieves a 2.4 and 5.2 percent increase in best-case application layer throughput for MTU sizes of 576 and 296 bytes, respectively. This shows that pure end-to-end error recovery consumes less radio resource for these MTU sizes as discussed in Section 3.3. However, even when TCP-SACK [34] is used, it is unlikely that the advantage in end-to-end throughput would be achieved in practice, due to interference with the end-to-end congestion control scheme commonly implemented in TCP [1]. 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 IP header compression is used over the wireless link[6].

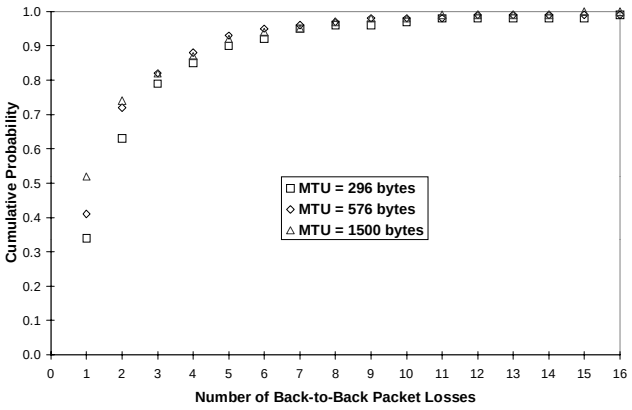

Figure 7: Distribution of back-to-back packet losses.

For pure end-to-end error recovery, IP header compression, as defined in [12] and [23], are not an option[7]. The reason is that the mechanism described in [23] causes the loss of an entire window worth of packets for each packet lost *after* the compression point. While the *Twice* algorithm proposed in [12] is more robust, it causes the same problem when two 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 7. The cumulative distribution of the number of back-to-back packet losses shows that 66, 59, and 48 percent of all such losses have a length of 2 or larger for an MTU of size 296, 576, and 1500 bytes, respectively. Alternatively, [12] also defines a "header request" mechanism but as our results show, link layer error recovery would be more appropriate on this wireless link.

One could argue in favor 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* [46] to adapt the path's MTU accordingly. However, that does not work when the link's end points (e.g., the PPP peers) are not "aware" that the link includes a wireless segment as in GSM-CSD (see Figure 2). Also, the path's MTU cannot be re-negotiated during a connection in current transport protocols[8] when the wireless link's error characteristics change. Link layer error recovery does not have these problems. It is independent of MTU sizes and also interworks well with IP header compression. Future systems favor link layer error recovery even more. Weaker FEC schemes are being deployed[9], which further decrease the throughput optimal frame size on those wireless links. Also, the next generation of the IP protocol [11] requires a minimum MTU of 1280 bytes and recommends an MTU of 1500 bytes or more on links such as GSM-CSD.

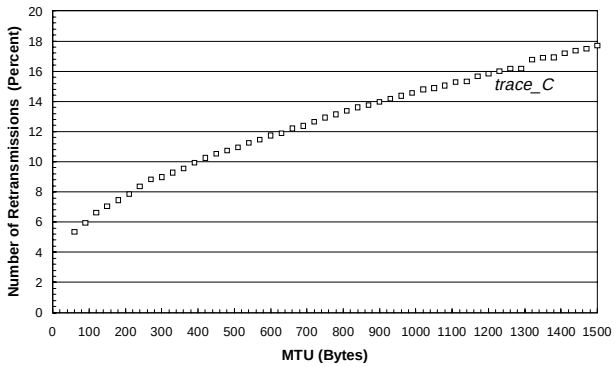

Figure 8: Number of end-to-end retransmissions.

Another shortcoming of pure end-to-end error recovery is that each retransmission has to traverse the entire path. This is depicted in Figure 8 for *trace_C*, showing the number of retransmissions (as a 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. Such flows impose an unfair load on a best-effort network, such as the Internet, and also on shared wireless access links. Apart from fairness, 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 it 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 wire-

6. In this case, we assume that the TCP/IP header is compressed to 6 bytes. Although compressed TCP/IP headers are typically 4 bytes long, a network-limited TCP connection drops one packet - in the ideal case - 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 a GSM-CSD link this leads to an average of about 6 bytes.
7. Unfortunately, this is not an option in any case when network layer encryption spans across the link.
8. 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.
9. Weaker FEC schemes are used in the new GSM-CSD service [17] and the GSM packet-switched data service [19].

less link. End-to-end error recovery complemented with link layer error recovery running over the wireless link "typically" does not require a single end-to-end retransmission [32].

5. Discussion: Error Control Performance

When addressing the problem of "reliable flows over wireless links" it is not sufficient to study a particular link as an isolated entity. Moreover, the entire end-to-end system including link and transport layer error control mechanisms and their interactions with end-to-end congestion control schemes has to be taken into account. We have deliberately ignored this for the best-case analysis presented in Section 3 and Section 4. In this section we discuss this subject in detail.

In today's Internet, senders have to interpret "packet loss" as a signal of congestion [1], [8], [22], [41]. We refer to such flows as being *loss-responsive*. However, "packet loss" is not unambiguous. A packet can get lost because of packet drop due to congestion at a router or a host, or because of packet corruption due to a transmission error causing a *non-congestion related packet loss*. A sender cannot discriminate among these, because packet corruption usually leads to a frame checksum error and subsequent discard of the packet at the link layer. Transmission errors beyond a certain rate inevitably lead to an underestimation of available bandwidth by the sender of a loss-responsive flow, reflected in reduced application layer throughput. This explains why wireless links are problematic: whereas transmission error rates on today's wireline links can be neglected, this is not true for wireless links. In addition, when hosts are mobile, cell handovers may cause data loss and some wireless networks may in certain situations only provide intermittent connectivity. We view the latter two cases as "long" transmission errors that do not have to be treated differently from "normal" transmission errors. This conflict between end-to-end congestion control and wireless links is not TCP-specific but exists for any loss-responsive flow.

Error control performance is the strongest argument in favor of flow-adaptive link layer implementations [30], [32]. In Section 4 we showed 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[10]. Implementing an optimal solution only from the end points of a path seems impossible; even if knowledge about the time varying error characteristics of each wireless link in the path was available. Flow-adaptive link layer implementations adapt their error control schemes to the individual QoS requirements of each flow sharing the link. For example, channel throughput can be aggressively maximized for a reliable flow as the per packet delay is less important. The flows' QoS requirements are derived (only) from the IP headers, e.g., the proposed differentiated service field [6], and are made available to the link layer on a per packet basis. This is implemented by an appropriate network-layer/link-layer interface or the link layer itself inspecting each IP header. The latter violates the principle of "protocol layering", but has the advantage that existing interface implementations need not be changed. Flow-adaptive implementations of link layer error control schemes are what the end-to-end argument [42] calls "an incomplete version of the function provided by the communication system [that] may be useful as a performance enhancement". We believe that carrying the application's QoS requirements as part of the flow's headers and accordingly adapting lower layer functions, such as error control, advances the discussion provided in Section 2.3 of [42].

There have been debates [14], [26], [31], [47] about how persistent link layer error recovery should be implemented. For non-flow-adaptive link layer implementations, retransmission persistency must be low to avoid 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. 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 packet, and resumes transmission with the next packet. This introduces a *retransmission delay* on the order of a few 100 milliseconds [26] or in more persistent implementations on the order of a few seconds, as in RLP (see Section 2). A *fully-reliable sender*, on the other hand, does not lose any packets, even over long link outages up to some conservative termination condition[11]. An upper limit for such a condition is the *MSL (Maximum Segment Lifetime)* [40] of 2 minutes[12], which also serves as an upper bound for the reassembly timeout after IP fragmentation [7]. We are not aware of the existence of such a reliable link layer protocol implementation.

The end-to-end argument [42] 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 interference with end-to-end congestion control schemes. Implementing semi-reliable link layer error recovery is always a compromise that avoids this conflict by emphasizing end-to-end error recovery [26]. However, this approach has 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 and/or to reduce the rate of non-congestion packet losses below a certain target rate. This is not feasible as it requires knowledge of the path's round trip time, which cannot be known at the link layer (unless it was explicitly signalled from the transport to the link layer). Therefore, 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. Together with the non-data-preserving property of semi-reliable link layer error recovery, this cannot yield optimal end-to-end throughput. Another fundamental problem occurs in case of temporary link outages, e.g., when a user temporarily roams into an area without wireless connectivity. In this case, all of the flow's unacknowledged packets are eventually discarded by a semi-reliable link layer sender. This causes an idle wait for a possibly backed-off transport layer retransmission timer to expire before the next packet is sent (up to 64 seconds [46]), while the link may already have become available again. If, on the other hand, packets are still queued at the wireless link, the end-to-end flow of data is re-started immediately after the link has become available.

Fully-reliable link layer error recovery for reliable flows has none of these problems and guarantees that any loss at the link is due to congestion[13]. This is the right signal to give to the senders of loss-responsive flows. In case of temporary link outages, this most likely

10. We believe that a similar line of argument applies to unreliable, but delay-sensitive loss-responsive flows.

11. 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 [8] are never handed to the link layer.
12. 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)* [46] field in the IP header.

causes a spurious timeout which in turn forces a go-back-N behavior in TCP [33], but (1) that is still better than the idle wait and (2) can be avoided with the Eifel algorithm described below. For the same reasons that favor fully-reliable link layer error recovery for reliable flows, should wireless networks implement mechanisms to support lossless intra- (and if possible also inter-) system cell handovers for data belonging to reliable flows.

On transport layer, the *Eifel algorithm* [33] increases end-to-end error recovery performance. When applied to TCP (*TCP-Eifel*) it uses the timestamp option [24] to eliminate the *retransmission ambiguity problem* [25]. It thereby avoids unnecessary duplicate packet transmissions caused by TCP's go-back-N behavior after spurious timeouts. In addition, it avoids unnecessary reductions of the flow's send rate by restoring the TCP sender's *congestion window* [22] after spurious timeouts and after spurious fast retransmits. The latter happens in the case of packet re-orderings beyond the *DUPACK threshold* [46]. TCP-Eifel can be incrementally deployed as it is backwards compatible and does not change TCP's congestion control behavior [1]. In addition, we argue that an adaptive transport layer retransmission timer should not be tuned to prevent all spurious timeouts. Some wireless networks often only provide intermittent connectivity. Despite a highly conservative retransmission timer, e.g., implemented in TCP [32], spurious timeouts cannot be avoided in such an environment unless dedicated transport layer support is implemented in the network, as proposed in [9]. 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 when the receiver's receive buffer cannot absorb any additional out-of-order packets. As a result, the sender is blocked from sending further packets. Instead, 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 [34] have nothing to add apart from improving end-to-end performance in the case of burst packet losses that may be caused by congestion. Nevertheless, any transport layer protocol should be robust against such cases. Also, some legacy wireless networks do not (e.g., some WLAN systems) or cannot (e.g., some satellite links are only uni-directional) provide reliable wireless links. Thus, for both reasons transport layer selective acknowledgements should be implemented.

6. Conclusion and Future Work

In this paper, we presented the results of a performance evaluation of link layer error recovery over wireless links for reliable flows. The analysis is based on a case study of the circuit-switched data channel implemented in GSM. We showed that the throughput on this channel can be increased by using a larger (fixed) link layer frame size. This yields an improvement of up to 25 percent when the channel quality is good and 18 percent even 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. These results highlight the importance of measurement-based analysis in wireless networks where protocol performance is highly dependent on the error characteristics of the wireless channel. Our results also suggest that techniques such as adaptive frame length control are worth further exploration for increasing channel throughput. This is a topic for our future research, as we plan to implement a measurement-based adaptive frame length control scheme in our testbed. This will also require a study of interactions with adaptive FEC schemes as implemented in upcoming wireless systems such as the General Packet Radio Service (GPRS) and the Universal Mobile Telecommunications System (UMTS).

We showed that pure end-to-end error recovery fails as a general solution for optimizing throughput when wireless links form parts of the end-to-end path. The fundamental problems are 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 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. In fact, we show that link layer error recovery over wireless links is essential for reliable flows to avoid these problems.

For reliable loss-responsive flows, we argued that fully-reliable link layer error recovery and lossless handover mechanisms should be implemented. This minimizes non-congestion related packet losses as far as possible. Furthermore, a flow-adaptive implementation ensures that link layer error recovery does not interfere with delay-sensitive flows. The attractiveness of this solution is that it does not require access to transport layer headers in the network. Its major advantages are its independence from transport (or higher) layer protocol semantics and the possibility of co-existence with any form of network layer encryption. As an optional, but complementary mechanism, we argued that the Eifel algorithm should be implemented in reliable transport protocols such as TCP. It increases end-to-end performance by avoiding unnecessary duplicate packet transmissions and unnecessary reductions of the flow's send rate. We believe that the combination of these mechanisms solves the problem of "reliable flows over wireless links" in the best possible way. Beyond that, we are currently experimenting with an implementation of TCP where the sender does not retransmit only 12 times [46] and then closes the connection, but instead retransmits until the *application* decides to close the connection. This is necessary in wireless networks where long disconnections may occur but are not critical for the application's operation.

In future work, we intend to focus on flow-adaptive link layers that optimize error control schemes for the QoS requirements of unreliable, but delay-sensitive flows as used by audio and video applications. These applications can often adapt to a range of bandwidths and loss rates provided by the end-to-end system. However, the QoS requirements of such flows are not as simple as those of reliable flows. Hence, more explicit information about these QoS requirements need to be available at the link layer.

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 Tom Henderson, Michael Meyer, Bela Rathonyi, and the anonymous reviewers for comments on earlier versions of this paper.

13. Apart from the more unlikely events of link layer error detection failures.

References

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

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

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

[4] H. Balakrishnan, R. H. Katz, *Explicit Loss Notification and Wireless Web Performance*, In Proceedings of IEEE GLOBECOM 98.

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

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

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

[8] B. Braden, et al., *Recommendations on Queue Management and Congestion Avoidance in the Internet*, RFC 2309, April 1998.

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

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

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

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

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] ETSI, *Digital cellular communications system (GSM Radio Access Phase 3); Channel coding*, GSM Specification 05.03, Version 6.0.0, January 1998.

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

[21] T. R. Henderson, R. H. Katz, *Transport Protocols for Internet-Compatible Satellite Networks*, IEEE JSAC, Vol. 17, No. 2, pp. 326-344, February1999.

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

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

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

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

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

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

[28] M. Kojo, K. Raatikainen, M. Liljeberg, J. Kiiskinen, T. Alanko, *An Efficient Transport Service for Slow Wireless Telephone Links*, IEEE JSAC, Vol. 15, No. 7, pp. 1337-1348, September 1997.

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

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

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

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

[33] R. Ludwig, R. H. Katz, *The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions*, ACM Computer Communication Review, 30(1), January 2000.

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

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

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

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

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

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

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

[41] K. K. Ramakrishnan, S. Floyd, *A Proposal to add Explicit Congestion Notification (ECN) to IP*, RFC 2481, January 1999.

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

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

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

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

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

[47] The Mailing List of the PILC Working Group (Performance Implications of Link Characteristics) of the Internet Engineering Task Force, http://pilc.grc.nasa.gov/pilc/list/archive/

[48] Sun Microsystems Inc., *NFS: Network File System Protocol Specification*, RFC 1094, March 1989.

[49] G. R. Wright, W. R. Stevens, *TCP/IP Illustrated, Volume 2 (The Implementation)*, Addison Wesley, January 1995.