TCP/IP over IEEE 802.11b WLAN: the Challenge of Harnessing Known-Corrupt Data

Michael Welzl[⊗], Mattia Rossi[⊗], Andrea Fumagalli[⊘], Marco Tacca[⊘]
[⊗]Institute of Computer Science, University of Innsbruck, A-6020 Innsbruck, Austria
[⊘]OpNeAR Lab, The University of Texas at Dallas, Richardson, TX, 75083, USA

***Abstract*—The two transport protocols DCCP and UDP-Lite can make use of data that are known to be erroneous, provided that the link layer hands over such data. A similar functionality has been suggested for TCP. In order to investigate the potential of these mechanisms in WiFi networks, we carried out a measurement study where we examined how often information about corrupt data reaches the transport layer when the corruption control at the link layer is disabled. Our results suggest that this may be a rare occurrence in certain scenarios.**

I. Introduction

Unless otherwise specified, TCP interprets the occurrence of a packet loss as an indicator of network congestion, which is resolved by promptly decreasing the congestion window, i.e., the TCP sender's transmission rate. When internetworking with wireless or satellite links, however, the loss of packets may have other causes, e.g., the transmitted signal is corrupted by interference and the received packet (frame at the link layer) contains bit errors. Even more drastically, the transmitted frame may not be received at all. Either way, the packet is lost due to signal corruption. Packet corruption losses (over the wireless link) may cause the undesired and unnecessary reduction of the TCP sender's rate, as they may be mistaken for congestion losses.

One possible approach to containing the TCP performance loss due to packet corruption losses is based on performance enhancing proxies (PEP) [1]. Other solutions attempt to distinguish congestion losses from corruption losses at the transport layer, e.g., explicit loss notification (ELN) [2]. The vast majority of the published results on the latter subject are based on simulation studies.

This paper presents a measurement study about the feasibility and practicality of ELN based methods in TCP/IP over IEEE 802.11b, whereby corrupt packets received over the wireless link are sent forward to reach the TCP receiver endpoint. The study measures how often information about corrupt data reaches the transport layer when the corruption control at the link layer is disabled, and what performance gain may be achieved by TCP consequently. The study compares the performance of today's most widely used versions of TCP, i.e. NewReno and Sack, against their modified versions, which include both a corruption detection option (CDO) and a corruption notification option (CNO). Since this is intended as a starting point for testing the feasibility of deploying CDO/CNO in a WiFi network, our goal was to find an an upper bound for the number of corrupt packets that can be delivered. Thus, in order to eliminate side effects from MAC, only one sender and receiver were used, directly communicating with each other in ad hoc mode. Conclusions drawn from the conducted experiments (see Section IV) suggest that CDO alone may not always be a sufficient mechanism to leverage effective CNO and overcome the undesired TCP sender's conservative rate decrease, which may result when packets are lost over the wireless link. While the focus of our study is on TCP, the presented results may also apply to the IETF protocols UDP-Lite [3] and DCCP [4], both of which can benefit from the delivery of known-corrupt data to transport endpoints.

II. System Description

A brief description of both CDO and CNO for TCP [5] is provided in this section, along with the modification required in the congestion control to leverage these options.

A. Corruption Detection and Notification

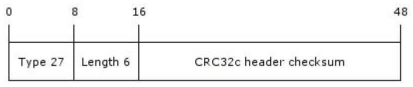

Fig. 1. CDO as specified in [5].

Corruption detection is performed using a TCP header option in the data segment. CDO is a six byte option (Fig. 1). Four of the six bytes contain the CRC32c, which is the same as the one used in the SCTP protocol [6], [7]. CRC32c covers a subset of both the TCP header and the pseudo header, thus providing an additional detection mechanism to the regular 16-bit TCP checksum. The covered fields are the source and destination address of the IP header, the source and destination port, the sequence and acknowledgment number, the CWR, ECE, ACK, RST, SYN and FIN control bits, and the ECN field [8].

CDO can be used to check the integrity of fields that are needed to deliver a partially corrupt packet to the TCP receiver. In turn, the TCP receiver informs the sender about the reception of a partially corrupt packet.

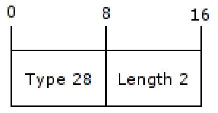

Fig. 2. CNO as specified in [5].

Corruption notification is performed using a TCP header option in the ACK segment. CNO is a 2 byte option (Fig. 2).

Upon the reception of a partially corrupted data segment, the TCP receiver can send an ACK segment with CNO. The CNO is used to request the retransmission of the data segment, while indicating that the segment loss was not caused by congestion. Upon the reception of an ACK segment with CNO, the TCP sender

- updates the values of the round trip time (RTT) and the retransmission timeout (RTO),
- retransmits the corrupt data segment,
- updates the congestion window size (cwnd).

The first two actions are specified in [5], whereas the third one is discussed next.

B. On the Congestion Control Reaction to Corruption

Standard TCP reacts to packet loss by halving cwnd, and therefore the sending rate. As already mentioned, this reduction of cwnd may be too conservative when losses are due to corruption. More precisely, the standard approach to reducing cwnd has a negative impact on a connection's performance with little (if any) overall benefit to the network. While it has frequently been assumed that the right reaction to corruption would be not to reduce the rate at all (e.g., "TCP HACK" [9]) there is currently no agreement in the IETF that this would be the ideal behavior. It was noted that, on shared wireless links, some form of congestion can manifest itself as corruption. Additionally, it seems undesirable to have a source continuously transmit at a high rate when most of its packets are received with errors in the payload.

For these reasons, the specification of CDO/CNO [5] does not define how to update cwnd upon reception of an ACK with CNO. One possible implementation of a sender's congestion control behavior is to react similarly to both corruption and congestion. Even with such a conservative choice, several benefits remain, e.g., earlier retransmission, retained control information for connection setup or teardown, correctly updated RTO, possibility to react to congestion earlier when congestion is explicitly indicated via the "ECN-Echo" flag [8].

Notably, the congestion control reaction to corruption is also left undefined in the DCCP [4] specification. As discussed in Section IV, it appears that some more work is needed at the link layer in the case of IEEE 802.11b before even debating about the ideal transport endpoint reaction.

III. Experimental Setup

This section describes the equipment used for the measurements, along with the modifications to the OS and drivers, which are required to prevent the IEEE 802.11b card from discarding corrupt frames received over the wireless link.

The setup is based on the Linux OS using the 2.6.17 kernel sources and the prism2/2.5/3 chipset WLAN card with the associated hostap driver. The modifications mainly involve the hostap driver and the implementation of the both CDO and CNO in the Linux OS.

The hostap driver modifications include:

- enable reception of corrupt frames,
- use of CRC32c to verify that the corruption does not affect any relevant part of the headers, as defined in [5]. If relevant parts of the headers are corrupt, the packet is dropped, otherwise it is passed to IP.

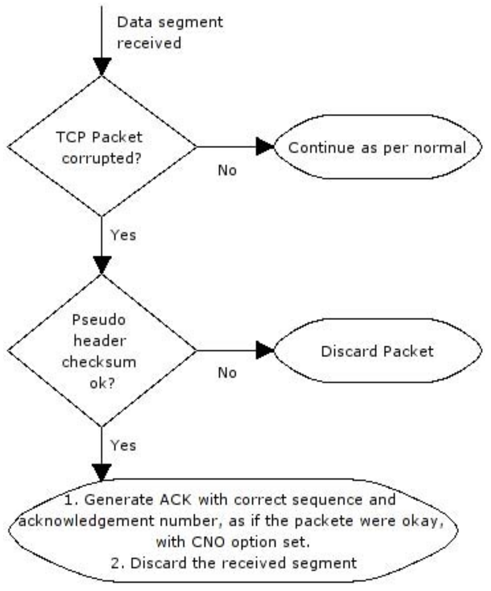
Data segment received
TCP Packet corrupted?
No
Continue as per normal
Yes
Pseudo header checksum ok?
No
Discard Packet
Yes
1. Generate ACK with correct sequence and acknowledgement number, as if the packete were okay, with CNO option set.
2. Discard the received segment

Fig. 3. Reception of a packet carrying CDO and sending of ACK (possibly carrying CNO).

CDO is implemented by:

- adopting CRC32c, borrowed from the SCTP implementation [6],
- verifying that CDO is not added to SYN, FIN, and ACK segments,
- adding CDO to TCP header[1].

The flow chart for processing a data segment with CDO is shown in Fig. 3.

CNO is implemented by:

- identifying corrupted packets, i.e., upon failure of the 16-bit TCP checksum[2] running a check on CRC32c. If CRC32c passes, an ACK segment with CNO is sent,
- generating the following ACK segment with CNO: the value for RN is set to the packet SN plus the packet length[3].

The flow chart for processing an ACK segment with CNO is shown in Fig. 4.

To investigate the ultimate performance gain for a single flow in the experimental setting, cwnd is updated as if there were no congestion when an ACK segment with CNO is received. While this approach may be regarded as too aggressive and imprudent by the IETF, it is a frequently suggested method in the literature, e.g., the simulation study of TCP HACK [9].

IV. Measurements

A. Hardware

The laptop used as a sender uses an integrated miniPCI wireless card. based on a IEEE 802.11g broadcom chipset.

The laptop used as a receiver does not have any integrated WLAN card. As already mentioned, a WLAN card with a

[1]Notice that this requires the MSS to be reduced by 8 bytes.

[2]If the TCP checksum is verified, then the packet contains no errors and the normal TCP behavior is followed.

[3]This choice avoids the generation of duplicate ACKs at the source.

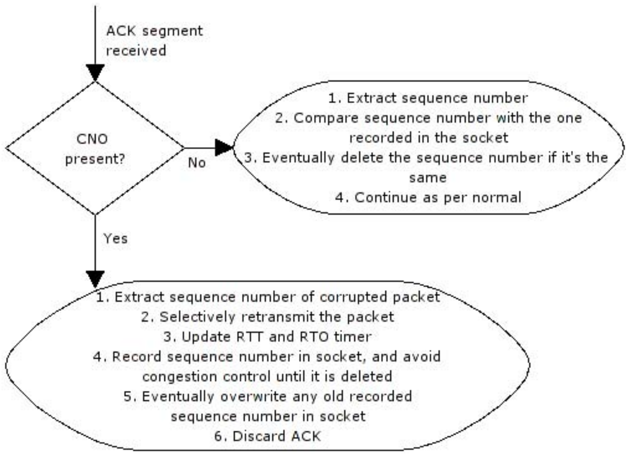
ACK segment received
CNO present?
No
Yes
1. Extract sequence number
2. Compare sequence number with the one recorded in the socket
3. Eventually delete the sequence number if it's the same
4. Continue as per normal
1. Extract sequence number of corrupted packet
2. Selectively retransmit the packet
3. Update RTT and RTO timer
4. Record sequence number in socket, and avoid congestion control until it is deleted
5. Eventually overwrite any old recorded sequence number in socket
6. Discard ACK

Fig. 4. Reception of an ACK carrying CNO.

prism2.5 chipset is added, i.e., the Netgear MA411 802.11b card.

B. Results

This section discusses the performance measurements of the implementation described in Section III over lossy links. For brevity, only one indoor and one indoor to outdoor experiment setups are reported. The performance tests are done using the iperf [4] tool, in usual surroundings with all the typical radio noise from cellphones and similar gadgets. Every measurement lasts 20 minutes.

The figures in this section show the transfer rate measured by iperf at every second in each experiment. The oscillation amplitude in the plots is used to determine how stable the transfer rate is. A smaller amplitude means less oscillations of the transfer rate, thus a stable throughput. The frequency of rate changes gives an idea of how well a connection works.

The lower line in the plots shows the signal strength of the connection in percent. Overall, the figures provide a good overview of how the transfer is affected by the signal strength.

Every setup consists of 4 measurements, i.e., TCP NewReno is used with and without Sack, and with and without CNO. As explained in Section III the Linux kernel NewReno congestion control is used.

1) Indoor measurements: With the indoor setup shown in Fig. 5, both notebooks are on the same floor.

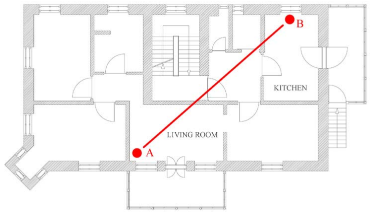

Fig. 5. Indoor setup.

This scenario is characterized by a quite weak wireless signal strength at the receivers. It is reasonable to assume that the maximum distance between the two notebooks is almost reached. The transfer rate hits 0 kbps several times, revealing that the packet loss rate is high. It is interesting to see that, as expected, in all figures the transfer rate follows the signal strength. The signal strength in Figures 6, 7, and 8 is low but remains mostly constant, while in Fig. 9 it gets worse towards the end of the measurement.

[4]http://sourceforge.net/projects/iperf

	CNO		no CNO	
	NewReno	Sack	NewReno	Sack
Tx packets	448318	497769	336058	254395
Rx packets	430254	478987	228324	171933
Lost packets	18064	18782	107734	82462
Corrupt packets	632	178	1435	722
Corrupt payload only	16	14	0	0

TABLE I
PACKET STATISTICS FOR INDOOR SETUP.

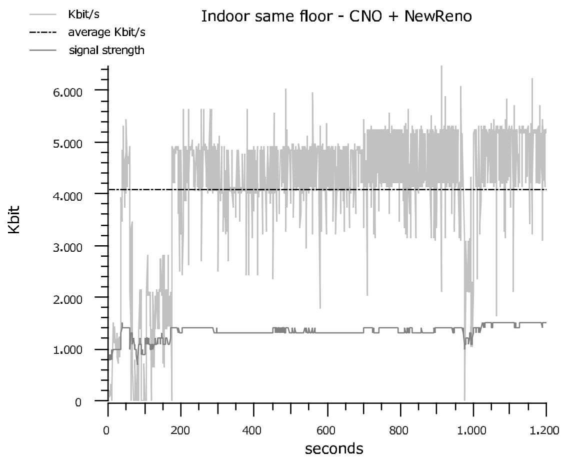

Fig. 6. Indoor setup: NewReno with CNO.

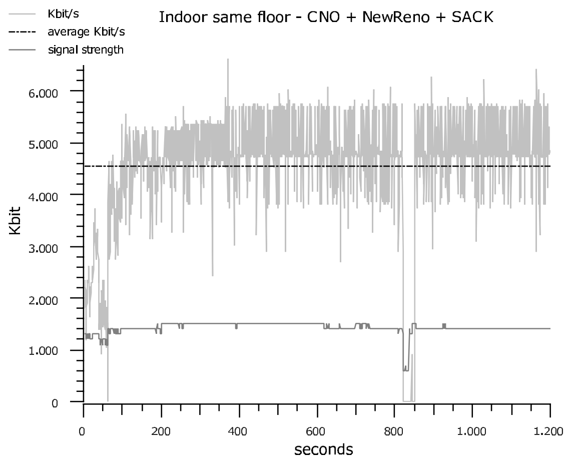

Fig. 7. Indoor setup: NewReno with Sack and CNO.

The results show that the average transfer rate is higher with CNO than without. However, Table I shows that it is unlikely that the higher transfer rate derives entirely from the use of CNO. There are only 16 packets with a corrupt payload, and

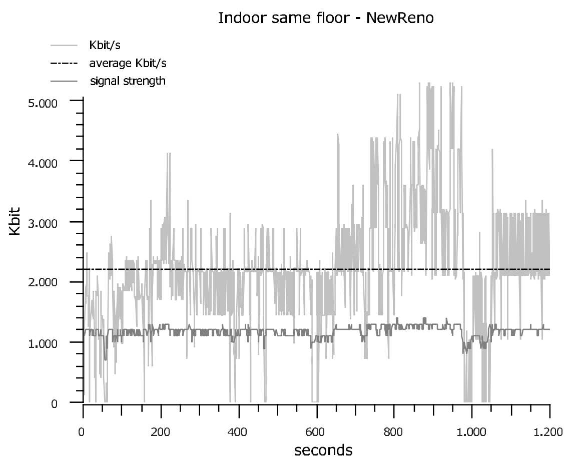

Fig. 8. Indoor setup: NewReno.

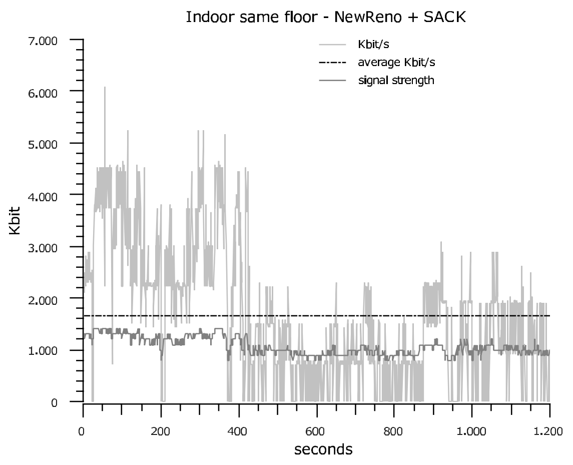

Fig. 9. Indoor setup: NewReno with Sack.

a better signal strength might have emphasized the effect of CNO.

2) Indoor to Outdoor measurements: In this setup, one notebook is inside the building and the other is outdoors. During the measurements the weather was cloudy with irregular wind, which could have affected the results. Fig. 10 shows the setup.

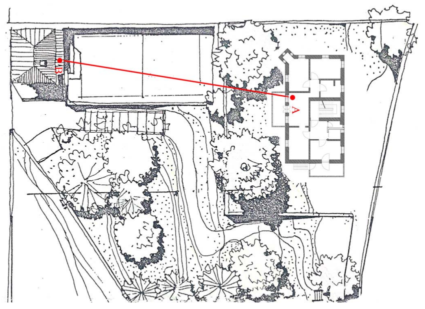

Fig. 10. The setup for the indoor to outdoor test.

Fig. 13 shows some difficulties to recover back to the previous transfer rate after a signal gap, which is a typical behavior expected from NewReno. Both Fig. 11 and Fig. 14

	CNO		no CNO	
	NewReno	Sack	NewReno	Sack
Tx packets	258324	246158	263442	253713
Rx packets	174651	135941	175170	169781
Lost packets	83673	110217	88252	83932
Corrupt packets	1475	2081	1277	1479
Corrupt payload only	109	93	0	0

TABLE II
PACKET STATISTICS FOR INDOOR TO OUTDOOR SETUP.

show some starting difficulties even if the signal strength is already at the same level as for the rest of the measurement. This behavior is reflected in the transfer rate.

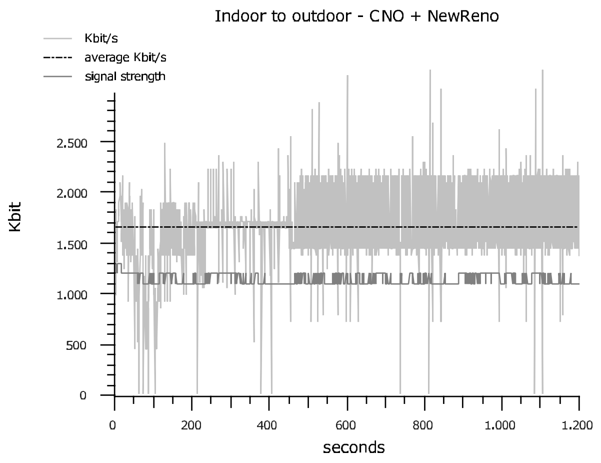

Fig. 11. Indoor to outdoor setup: NewReno with CNO.

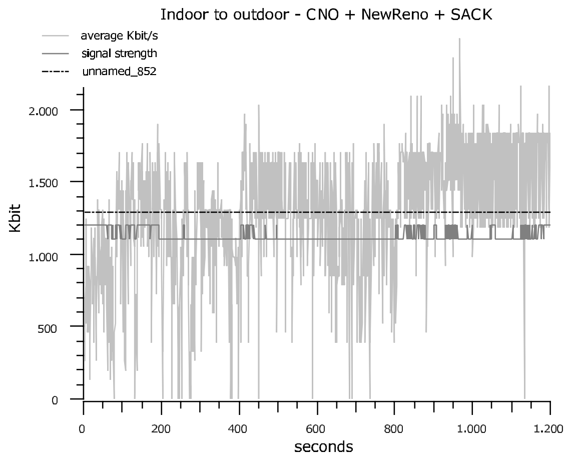

Fig. 12. Indoor to outdoor setup: NewReno with Sack and CNO.

Table II shows the largest amount of corrupt packets with correct header. However, this is combined with a high number of lost packets. As a result, the ratio between usable corrupt packets and lost packets is still low (0.13%), which is too small of a value to leverage CNO.

V. Related Work

The idea of exploiting knowledge about corruption in the network, or even directly using corrupt data at the transport

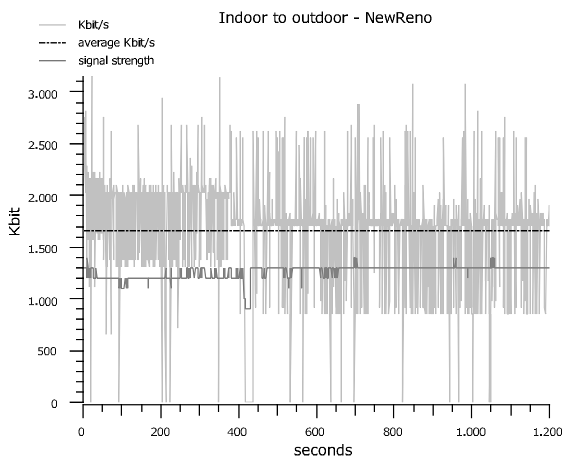

Fig. 13. Indoor to outdoor setup: NewReno.

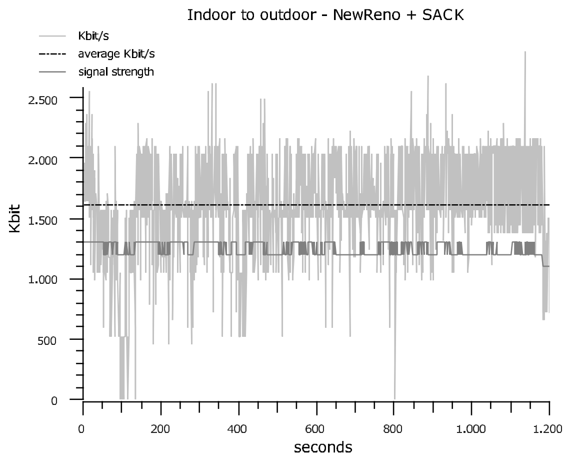

Fig. 14. Indoor to outdoor setup: NewReno with Sack.

layer, is not a new one. In addition to the aforementioned ELN, (TCP HACK [9], IETF protocols UDP-Lite [3] and DCCP [4]) there is a number of other related efforts, many of which are described in [2].

Notably, the authors of [10] obtained more encouraging results with GPRS than those reported in Section IV. This clearly shows that the outcome of any such study can vary greatly depending on the specific lower layer in use. The trade-offs of using link layer ARQ with various levels of persistence are discussed in [11]. A general discussion on the idea of passing corrupt data across layers can be found in [12]. An interesting scheme called "LossTolerant TCP" (LT-TCP) is proposed in [13]: here, FEC is added at the transport layer to cope with corruption based packet losses.

VI. Conclusions and Future Direction

Experimental measurements collected for a TCP/IP connection over a IEEE 802.11b link show that the CDO mechanism alone may not suffice to enable CNO to overcome the undesired and unnecessary transport sender's rate decrease caused by error prone wireless links. The reason is that the majority of the corrupt packets over the wireless link is not delivered to the link receiver at all, thus preventing the CDO mechanism from detecting most of the lost packets due to signal corruption.

Since there was only one wireless data sender (hence experiencing limited MAC interaction), and both ARQ and checksum were turned off at the link layer, the impact of this layer on our results may have been minimal. It is reasonable to believe that failures to deliver packets to the link receiver are due to error bursts (possibly related to the way bits are encoded at the physical layer), which adversely affect the header just as well as the payload.

It should also be noted that only prism2/2.5/3 chipset WLAN cards were used in the experiment at the receiver. It is quite possible that a different chipset could lead to different results. Therefore, it is incorrect to conclude from the presented measurements that ELN mechanisms such as CDO are altogether worthless — again, note that the authors of [10] obtained contradictory results with GPRS. It can be concluded that such mechanisms will not work well with one common piece of equipment, and that they should possibly be extended with complementary mechanisms at lower layers in such scenarios. These mechanisms must be designed to account for the possibility that a packet may not be delivered to the link receiver at all, and yet the TCP receiver must be notified of the packet loss incurred over the wireless link.

References

[1] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135 (Informational), Jun. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3135.txt

[2] R. Krishnan, J. Sterbenz, W. Eddy, C. Partridge, and M. Allman, "Explicit transport error notification (eten) for error-prone wireless and satellite networks," *Computer Networks*, vol. 46, no. 3, October 2004.

[3] L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst, "The Lightweight User Datagram Protocol (UDP-Lite)," RFC 3828 (Proposed Standard), Jul. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3828.txt

[4] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340 (Proposed Standard), Mar. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4340.txt

[5] M. Welzl, "Tcp corruption notification options, internet-draft draft-welzl-tcp-corruption-00.txt, work in progress," June 2004.

[6] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, "Stream Control Transmission Protocol," RFC 2960 (Proposed Standard), Oct. 2000, updated by RFC 3309. [Online]. Available: http://www.ietf.org/rfc/rfc2960.txt

[7] J. Stone, R. Stewart, and D. Otis, "Stream Control Transmission Protocol (SCTP) Checksum Change," RFC 3309 (Proposed Standard), Sep. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3309.txt

[8] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168 (Proposed Standard), Sep. 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3168.txt

[9] R. K. Balan, B. P. Lee, K. R. R. Kumar, J. Jacob, W. K. G. Seah, and A. L. Ananda, "TCP HACK: TCP header checksum option to improve performance over lossy links," in *20th IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2001.

[10] J.Chesterfield, R.Chakravorty, S.Banerjee, P.Rodriguez, I.Pratt, and J.Crowcroft, "Transport level optimisations for sreaming media over wide-area wireless networks," in *WIOPT'04*, March 2004.

[11] G. Fairhurst and L. Wood, "Advice to Link Designers on Link Automatic Repeat ReQuest (ARQ)," RFC 3366 (Best Current Practice), Aug. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3366.txt

[12] M. Welzl, "Passing corrupt data across network layers: An overview of recent developments and issues," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 2, pp. 242–247, February 2005.

[13] O. Tickoo, V. Subramanian, S. Kalyanaraman, and K. Ramakrishnan, "Lt-tcp: End-to-end framework to improve tcp performance over networks with lossy channels," in *IWQoS 2005*, June 2005.