IMPROVING TCP PERFORMANCE OVER WIRELESS WANS USING TCP/IP-FRIENDLY LINK LAYER

Markku Kojo, Davide Astuti, Laila Daniel, Aki Nyrhinen and Kimmo Raatikainen

Department of Computer Science, University of Helsinki
P.O. Box 26, FIN-00014 University of Helsinki, Finland

Keywords: TCP/IP, Satellite Networks, DVB-S/RCS, FEC, ARQ, Link-Layer protocols, TCP performance

Abstract: In this paper we propose the use of a TCP/IP-friendly link level error recovery mechanism with novel design in conjunction with state-of-the-art Transmission Control Protocol (TCP) enhancements to improve TCP performance on network paths involving Wireless Wide-Area Network (W-WAN) links. We show that by combining a selected set of TCP enhancements TCP performance is significantly improved over W-WAN links. In addition, we employ a TCP/IP-friendly link layer protocol which minimizes the additional delay due to the Automatic Repeat reQuests (ARQ) by limiting the number of retransmission attempts and by adding redundancy in the retransmitted frames in a novel way. We perform experiments in an emulated satellite environment with a real implementation of the TCP/IP-friendly link layer and TCP enhancements in Linux. The results show that both TCP enhancements and link-level ARQ significantly improve TCP performance over W-WAN links, and combining the approaches yields the best performance.

1 INTRODUCTION

Transmission Control Protocol (TCP) is the main transport protocol in the Internet and it carries the bulk of the traffic in the Internet. With the rapid rise in wireless communication in recent years, it has become important to adapt TCP to heterogeneous environments that include both wireline networks and Wireless Wide-Area Networks (W-WANs), such as satellite and terrestrial wireless networks. TCP performance over W-WANs is often poor due to the characteristics of wireless links such as high latency, link losses and often limited bandwidth. The link losses often occur in bursts resulting in the loss of several segments in a single TCP window. In terms of resulting transport performance the TCP loss recovery tends to be inadequate with such loss patterns. Typically wireless links also possess other characteristics, such as bandwidth-on-demand allocation, bandwidth asymmetry, that may affect TCP performance.

The several schemes proposed to improve the performance of TCP over wireless links can be broadly classified as split-connection, link-layer and end-to-end approaches (Balakrishnan et al., 1997). The split-connection approach replaces an end-to-end TCP connection with two or more separate connections.

One of the connections is across the problematic wireless link allowing TCP modifications for more efficient loss recovery or even replacing TCP with an alternative transport protocol. This, however, has serious implications as it breaks the end-to-end semantics of the connection. In particular, it cannot coexist with the end-to-end use of Internet Protocol security (IPsec)(Border et al., 2001).

In the link-layer approach, link-level error recovery is used locally on an error-prone link to improve the reliability of the link. Local knowledge of the link can be used to optimize the recovery mechanism. Many link-layer recovery mechanisms are based on Automatic Repeat reQuests (ARQ) used in a highly persistent mode of recovering lost frames. This may cause unwanted interaction with TCP retransmissions (Balakrishnan et al., 1997)(Fairhurst and Wood, 2002). In particular, highly persistent link ARQ easily leads to delay spikes that can result in suboptimal TCP performance by causing spurious TCP timeouts, unnecessary retransmissions and a multiplicative decrease in the congestion window size.

The end-to-end approach preserves the end-to-end semantics of TCP. The proposals in this category include TCP enhancements that follow the Internet congestion control principles (Floyd, 2000)(Allman et al.,

1999) suggested by Internet Engineering Task Force (IETF) such as large initial window (Allman et al., 2002), TCP Selective Acknowledgment Option (TCP SACK) (Mathis et al., 1996), window scaling (Jacobson et al., 1992), and TCP Control Block Interdependence (TCP-CBI) (Touch, 1997). Other proposals include research work such as TCP Peach (Akyildiz et al., 2001) and TCP Westwood (Mascolo et al., 2001).

In this paper, we propose the use of a selected set of state-of-the-art TCP enhancements in conjunction with a TCP/IP-friendly link-level error recovery mechanism to achieve acceptable TCP performance over W-WANs. There has been little earlier work in evaluating TCP performance when a proper set of TCP enhancements are combined. Studying the combined effect of the TCP enhancements allows better understanding how well a state-of-the-art TCP can perform in such a challenging environment. The TCP/IP-friendly link protocol is employed over the error-prone wireless link to reduce the residual packet-error rate and thereby allow more efficient TCP operation. It minimizes the additional delay due to the ARQ by limiting the retransmission attempts to a low number, but still keeping the residual packet-error rate at low level. This is possible by adding FEC-encoded redundancy in the retransmitted frames in a novel way, increasing the probability that a retransmitted frame is successfully delivered without additional retransmissions.

In addition, we incorporate other important and useful features into the TCP/IP-friendly link protocol implementation. One such feature is the use of flow control between the IP layer and the link layer together with limiting the amount of link buffering. With wireless systems, such as Digital Video Broadcasting-/Return Channel via Satellite (DVB-RCS) satellite systems (ETSI, 2003), IP packets often flow directly to the Medium Access Control (MAC) buffer without flow control between the layers and the excess packets are dropped if the MAC buffer becomes full. The IP queue will always be empty; thus, proper IP router queue sizes to control total amount of buffering cannot be used and the use of the IP active queue management mechanisms is not effective. In the TCP/IP-friendly approach, arriving packets are forwarded to the MAC buffer only if space is available in the MAC buffer; otherwise, packets are kept in the IP buffer. In addition, any unnecessary link-level buffering is avoided to minimize the overall delay.

We perform an extensive set of performance experiments in an emulated satellite DVB-S/DVB-RCS environment with real TCP/IP stacks in the end hosts and employing our implementation of the TCP/IP-friendly link-layer protocol, called Satellite-Link Aware Communication Protocol (SLACP) (Kojo et al., 2004), over the satellite segment. A satellite platform is used to emulate several levels of error rate and a Demand Assignment Multiple Access (DAMA) bandwidth-on-demand allocation scheme on the satellite return link.

The results show that employing the selected set of TCP enhancements reduces the median transfer time up to 68 % depending on the link error rate. Employing the SLACP protocol on a lossy link is extremely beneficial to TCP, reducing the median transfer time more than 90 % for high error rates. Although both TCP enhancements and SLACP protocol can significantly improve TCP performance, combining the use of the TCP enhancements with the SLACP protocol yields additional performance gain.

2 TCP/IP-FRIENDLY LINK LAYER FOR W-WAN LINKS

Typically link-layer design follows a strict layering paradigm. This implies that most design choices are done almost in full isolation from the other protocol layers. However, when designing a link layer for a W-WAN link with an intention to best support Internet protocols, several crucial design choices that heavily affect the performance of the TCP/IP traffic carried over the link must be made.

As W-WAN links typically experience specific link characteristics, in particular high latencies and frequent frame losses due to bit-corruption, designing link-level error recovery requires specific care. TCP suffers from uncorrected link errors as it interprets all losses as congestion signals and reduces the transmission rate drastically. With high error rates TCP spends excessive time in slow start or congestion avoidance procedures triggered by packet losses due to transmission errors, resulting in severe performance penalty. In addition, W-WAN links not only incur high loss rates but often errors occur in bursts, resulting in multiple segment losses within a TCP window. Regular TCP with NewReno-based fast recovery (Floyd and Henderson, 1999) can recover at most one segment per Round Trip Time (RTT), resulting in additional performance penalty in presence of bursty losses. The TCP SACK option (Mathis et al., 1996) with an appropriate loss recovery algorithm (Blanton et al., 2003) is more effective than NewReno when multiple TCP segments are lost within a window as it may recover multiple segments without requiring one or more round-trips per lost segment (Dawkins et al., 2001b). However, even the SACK-based loss recovery is often inefficient if a large number of segments are lost in a single TCP window or if the window is small, because the number of duplicate ACKs may remain too low to provide enough SACK-information for recovering the lost segments.

The high link latencies typical for W-WAN links translate to long end-to-end RTT for TCP. Long RTT, in particular when combined with high error rate, adversely affects TCP throughput (Padhye et al., 1998) (Dawkins et al., 2001b). This is because after each loss event it may take multiple round trips for TCP to repair the lost segments and, more importantly, after recovering the lost segments TCP enters congestion avoidance with a reduced congestion window, requiring several round trips to increase the congestion window back to the level prior the loss event. Therefore, it is much slower for TCP to restore the earlier transmission rate if the RTT is long.

When traversing over a lossy W-WAN link, it would be highly beneficial for TCP to employ link-layer ARQ as it lowers the residual packet loss rate significantly. However, link-layer ARQ techniques tend to introduce delay spikes that easily lead to spurious TCP Retransmission Timeouts (RTOs), if highly persistent ARQ is employed (Fairhurst and Wood, 2002). After a spurious RTO, the late TCP acknowledgements of original TCP segments arriving at the sender usually trigger unnecessary retransmissions of whole window of segments during the RTO recovery (Sarolahti et al., 2003). This is inefficient and results in wasting the scarce link capacity. Another way to reduce the frame loss rate is to use Forward Error Correction (FEC), but it decreases the available link bandwidth by introducing significant amount of overhead in the form of redundancy.

Perfect reliability is not a requirement for IP networks, nor is it a requirement for links (Karn, 2004). In order to best serve TCP traffic, the link-level ARQ mechanism should reduce the number of retransmission attempts down to minimum with the intention to minimize the additional delay and possible interaction with TCP timers. This can be achieved in our TCP/IP-friendly approach with a novel use of FEC with retransmitted frames. The original transmission of frames is not protected with FEC to save bandwidth when link conditions are good.

The frames that require retransmission are immediately retransmitted when a repeat request arrives. The FEC-encoded redundancy is not added separately to each frame as usual. Instead, the retransmitted frames are organized as FEC blocks. Each FEC block consists of actual frames and redundancy frames (see Figure 1). The FEC-encoded redundancy is added to the redundancy frames by computing the Reed-Solomon codeword vertically so that the ith octet of each frame in a FEC block comprises a codeword. As soon as a predetermined amount of retransmitted frames, or other frames deserving FEC protection, has been sent or a threshold timer expires, a proper amount of FEC-encoded redundancy frames are computed to complete the FEC block and transmitted. A link receiver can use the redundancy frames in a FEC block for recovering any of the retransmitted but lost actual frames. In addition, the position of the errors (missing octet in a codeword) is known, resulting in an erasure channel with much better recovering capability.
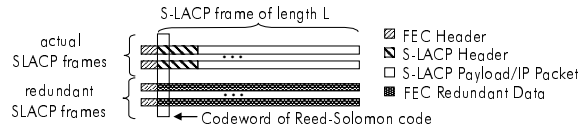

Figure 1: Organization of a FEC block

To minimize the additional delay due to the retransmissions we suggest reducing the number of retransmission attempts down to one attempt only. The delay is further reduced by assigning higher priority for retransmitted frames by scheduling them first for transmission and by implementing selective repeat requests without relying on link sender retransmission timers. Instead, an ACK timer is used at the link receiver to repeat the latest acknowledgement if new frames are not arriving.

With this approach low delay is achieved without sacrificing the reliability or adding excessive amount of redundancy to all frames. Even though the approach improves reliability with minimal amount of extra delay, it is not desirable for all types of flows to increase the reliability at the cost of any extra delay. The additional delay and variations in delay that ARQ introduces may be highly undesirable for delay-sensitive flows such as interactive audio, for example. Therefore, the link layer should be able to treat separately IP flows with different classes of service and turn off the link ARQ mechanism for flows not benefiting from it. This requires implementing several logical link channels over a single physical link.

Implementing an efficient ARQ mechanism requires storing the sent but not yet acknowledged frames in a relatively large send buffer for possible retransmission. Many link-layer ARQ implementations accept a full buffer of packets from the upper layer to be queued at the link head before transmission over the channel. However, buffering unsent packets at the link layer adds to the queuing delay and thereby to the end-to-end RTT which is undesirable for TCP. A link sender implementing an ARQ mechanism does not need to buffer a full window of unsent data, but it is quite enough to accept only one or a few unsent packets. This requires flow control between the IP layer and link layer. The additional benefit is that the majority of packets can be kept in IP queues subject to proper IP-level queue management.

We have implemented the SLACP protocol on Linux (Kojo et al., 2004). It is a logical link layer protocol that runs on top of satellite link service and follows closely the TCP/IP-friendly link protocol principles discussed in this section.

3 TCP IMPROVEMENTS

In this section we briefly introduce several techniques that we have selected for enhancing the performance of TCP in W-WAN networks. We have selected only such enhancements that follow the congestion control principles (Floyd, 2000)(Allman et al., 1999) and therefore can be safely used over the global Internet.

Increasing TCP's Initial Window from 1 up to 4 segments (Allman et al., 2002) increases the number of segments during the first RTT, allowing more rapid opening of the congestion window. This is extremely useful for high latency environments.

With the **Delayed ACKs After Slow Start (DAASS)** (Allman et al., 2000) technique the receiver is sending an acknowledgment for every segment during slow start instead of using delayed acknowledgements from the very beginning of the connection. This will also inflate the congestion window faster.

TCP Window Scaling (Jacobson et al., 1992) allows a TCP connection to use a window larger than the standard maximum TCP window size (65,535 bytes) in order to fully utilize the available bandwidth on a network path with large bandwidth-delay product. This is useful in many high latency environments where the bandwidth-delay product easily exceeds 65,535 bytes.

Limited transmit (Allman et al., 2001) allows the sender to transmit a segment for every of the first two duplicate acknowledgments. This is useful for TCP connections with small congestion windows or when a large number of segments are lost in a single transmission window as otherwise it may happen that not enough duplicate acknowledgments arrive at the sender to trigger the fast retransmit algorithm and the TCP sender must wait for a costly retransmission timeout.

TCP SACK option (Mathis et al., 1996) enables the use of a loss recovery algorithm (Blanton et al., 2003) that allows TCP to recover more efficiently from multiple segment losses in a window of data as discussed in Section 2.

Forward RTO Recovery (F-RTO) (Sarolahti et al., 2003) (Sarolahti and Kojo, 2004) algorithm effectively helps detecting spurious TCP RTOs and avoiding unnecessary retransmissions and thereby improves TCP performance in the presence of delay spikes. This is very useful as delay spikes may still occur even though a TCP/IP-friendly link ARQ mechanism is used, for example, due to dynamic link bandwidth allocation. An alternative to F-RTO is the TCP Eifel algorithm (Ludwig and Katz, 2000), which can detect spurious TCP timeouts and avoid unnecessary retransmissions as well.

TCP Control Block Interdependence (TCP-CBI) (Touch, 1997) mechanism aims to share a part of the TCP Control Block (TCB) to improve transient TCP performance. TCB is a data structure associated with each TCP connection containing information about the connection state such as the RTT estimate, congestion window size, and slow-start threshold (*ssthresh*). In particular, sharing the *ssthresh* value from a previous TCP connection is useful. If no packet losses occur during the initial slow-start, the slow-start overshoot (Dawkins et al., 2001a) occurs in the end of the slow-start, resulting in multiple packet losses in a single TCP window. Sharing the *ssthresh* value from the previous connection allows a TCP sender to use this value as a better estimate of the available bandwidth when initializing the *ssthresh* for a new connection. Thus, the new connection is able to complete the initial slow-start earlier and avoid the slow-start overshoot problem. However, reusing the *ssthresh* value can also create problems of using too low initial *ssthresh* value when the *ssthresh* of a previous connection is reduced by the occurrence of a packet loss due to a link error.

4 PERFORMANCE EXPERIMENTS

We ran experiments to evaluate the performance of the TCP enhancements combined with our implementation of the TCP/IP-friendly link layer, the SLACP. We compare the TCP performance of regular and enhanced TCP variants over an emulated satellite link. We also compare the TCP performance with and without SLACP.

Figure 2 shows an overview of the network topology used in the experiments. The satellite end-host (H1) accesses the emulated satellite network through a Satellite Terminal (ST). The other end host (H2) is connected to the satellite network through a Broadband Access Server (BAS). The ST and BAS act as routers. The direction from the ST to the BAS is referred to as Return Link; the other direction is referred to as Forward Link. The SLACP protocol is employed over the satellite link between the ST and the BAS. The SLACP protocol is configured to retransmit a missing frame at most once. The retransmitted frames are protected with FEC-encoded redundancy frames. The number of redundancy frames in a FEC block equals to the number of actual frames in the block.
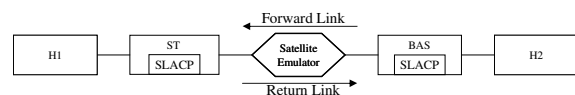

Figure 2: Network Topology for Performance Experiments

For emulating the satellite link characteristics we

used a satellite emulator platform provided by Alcatel Space. The Network Emulation Platform (NEP) is representative of a DVB-S/DVB-RCS satellite system. In this study, we used the following three DAMA classes of traffic assignment defined in the DVB-RCS standard.

In the **Constant Rate Assignment (CRA)** class the allocated bandwidth is guaranteed and requires no dynamic signaling. The traffic is not subjected to any scheduling delay.

In the **Rate Based Dynamic Capacity (RBDC)** class the bandwidth allocation is based on instantaneous rate requests sent by the ST. The ST can request bandwidth up to a pre-fixed ceiling value that the ST is guaranteed to get. The request remains effective until it is updated by ST or until it is timed out. During the time that a bandwidth request remains effective, the ST traffic is not subjected to any additional bandwidth allocation delay. In contrast to CRA, RBDC strategy allows for statistical multiplexing among many terminals, resulting in a more efficient use of the satellite bandwidth.

In the **Volume Based Dynamic Capacity (VBDC)** class the bandwidth is assigned in response to a request by the ST. The bandwidth request is based on the amount of data waiting in the MAC buffer. The scheduler assigns capacity from a queue of requests among all STs it serves, within the constraint of the remaining capacity after CRA and RBDC assignments. As bandwidth is shared by a number of terminals there is no guarantee on capacity availability for a specific bandwidth request. This strategy provides an efficient use of the network resources but traffic may experience a significant delay.

The DAMA bandwidth allocation scheme is applied on the Return Link and accurately emulated by the NEP emulator. The emulated bandwidth on the satellite Forward link is 512 kbps and on the Return link 128 kbps. The Maximum Transmission Unit (MTU) size is 1450 bytes and the one-way propagation delay is set to 250ms. The connection between H1 and ST as well as H2 and BAS is 100 Mbps fixed LAN.

The satellite MAC buffer size has been tuned experimentally to allow the maximum utilization of the bandwidth in the case of VBDC. Test results showed that a MAC buffer size of 14400 bytes is enough. As discussed earlier, a typical IP-to-satellite interface does not provide flow control between the IP and MAC layer and therefore all buffering occurs at the MAC layer. However, a certain amount of buffer space is needed to allow the TCP congestion window to grow up and correctly estimate the available link bandwidth. Therefore, when SLACP is not used the MAC buffer size is set to 28800 bytes, which is roughly the Return Link bandwidth-delay product with an RTT of 2 seconds (typical RTT values

with VBDC are between 1.3 and 2 sec). When using SLACP with flow control between the layers, we were able to reduce the MAC buffer to 14400 bytes. The SLACP accepts only one unsent packet to its send buffer. The IP queue size is 40 kbytes.

When modeling the satellite errors, we wanted to experiment with several error levels having the emphasis on bursty error behavior. This is modeled with a two-state Markov model with an error-free good state and a bad state where an error burst corrupts all packets. In order to achieve better control over the state lengths, we use uniform distributions for both burst inter-arrival time (good state duration) and error burst length. The burst inter-arrival time is selected such that several error bursts occur during a single transfer. The parameter values of the six error models used in this study are shown in Table 1. The error-free link model (None) is used as a baseline. In the "Low" error model, error burst length is static 60 ms representing error behavior in which only one or two full-sized packets are lost during the error burst. In the "Medium" (Med) and "High" error models, the error burst occurs more frequently, but the burst length is not very long; from 2 to 5 packets are lost. In the "Very High" (VHigh) and "Huge" models adjacent or close-to-each-other error bursts are possible, resulting in an error behavior that is challenging for the SLACP protocol as a latter error burst is likely to affect the retransmitted SLACP frames. We had to limit the error burst length with the "Huge" model to static 100 ms as otherwise TCP started to experience serious problems in completing the transfer in too many test cases.

Table 1: Satellite error models

Error model	None	Low	Medium	High	VHigh	Huge
Burst interval (s)	-	10-30	5-15	2-10	0-10	0-7
Burst length (ms)	-	60	100-300	100-300	100-300	100

We modified the TCP implementation in Linux to implement two TCP variants: Reference TCP and Enhanced TCP. The Reference TCP modifies the default Linux TCP behavior to use the initial window size of one segment, disable Limited Transmit and rate-halving, and set delayed ack threshold to 200ms. TCP SACK and Window Scaling options are disabled. We consider the Reference TCP to implement a behavior that is typical for a regular TCP today.

The Enhanced TCP enables the enhancements discussed in Section 3, that is, the initial window of 4 segments, Window Scaling, TCP SACK, Limited Transmit, DAASS, and F-RTO. Using the initial window of 4 segments with the Maximum Segment Size (MSS) of 1410 bytes is not exactly allowed (Allman et al., 2002). Therefore, we repeated a large number of the tests also with the initial window of 3 segments but there were no notable differences in the results.

Table 2: TCP performance results (32 replications)

DAMA	Error Model	TCP	Transfer time for 1 flow 1x1MB (sec)						Transfer time for 4 flows 4x250KB (sec)					
			NOSLACP			SLACP			NOSLACP			SLACP		
			25perc	Median	75perc	25perc	Median	75perc	25perc	Median	75perc	25perc	Median	75perc
CRA	None	Ref	90,20	90,21	90,22	86,69	86,71	86,76	80,02	80,04	80,05	84,81	85,17	85,97
	None	Enh	85,07	85,09	85,10	86,92	86,97	87,04	83,44	83,47	83,48	86,05	86,19	86,29
	Low	Ref	94,61	98,35	104,62	91,02	91,84	92,36	81,59	83,32	84,98	86,91	87,54	87,98
	Low	Enh	86,52	86,95	88,13	88,51	88,88	90,74	84,24	84,60	85,08	86,28	87,70	88,60
	Med	Ref	154,80	169,68	183,68	96,26	98,25	100,36	87,09	90,60	93,67	90,50	91,59	92,96
	Med	Enh	105,07	113,96	123,85	92,20	93,24	95,33	86,47	89,42	95,01	90,62	92,44	94,08
	High	Ref	228,59	239,93	256,75	102,04	103,74	108,62	98,70	102,59	110,63	95,60	97,33	98,48
	High	Enh	145,62	153,12	174,69	96,45	98,21	101,12	89,83	95,27	99,56	93,24	96,12	98,15
	VHigh	Ref		231,66		105,40	107,41	110,40	103,77	112,61	137,83	97,61	98,84	101,83
	VHigh	Enh		163,30		101,3	102,22	105,21	93,20	100,49	102,34	96,89	98,59	100,73
	Huge	Ref		490,37		117,25	122,71	130,80	176,85	211,47	302,74	112,32	115,47	119,27
	Huge	Enh		279,14		112,49	120,17	146,96	125,05	165,10	223,66	109,56	112,00	115,94
RBDC	None	Ref	103,98	104,04	104,06	89,97	90,20	90,67	83,54	83,61	84,70	87,88	88,29	89,02
	None	Enh	93,67	93,69	93,70	89,32	90,17	91,41	82,68	89,36	89,46	87,35	89,45	90,12
	Low	Ref	113,84	123,25	137,85	96,58	97,19	98,80	85,06	86,78	89,64	89,44	90,48	91,65
	Low	Enh	90,94	97,37	110,05	91,33	93,16	95,86	85,63	91,13	93,83	87,37	89,20	92,43
	Med	Ref	368,25	422,91	463,37	103,73	106,47	108,25	95,33	101,19	107,00	94,05	95,04	96,65
	Med	Enh	151,92	172,40	201,63	95,72	99,40	106,15	92,60	95,30	106,36	91,11	93,19	96,72
	High	Ref	1019,10	1125,12	1454,49	109,84	113,79	117,05	119,00	135,85	188,21	99,30	101,53	105,91
	High	Enh	309,15	359,58	456,51	102,35	106,34	110,98	100,92	114,94	129,95	96,33	97,36	102,14
	VHigh	Ref		1420,18		115,89	118,50	124,96	140,97	162,23	245,02	101,59	103,58	107,82
	VHigh	Enh		453,21		102,66	106,40	112,48	102,47	136,64	228,25	99,19	101,83	107,27
	Huge	Ref		>3hours		138,11	152,00	169,80	>3hours			119,21	123,27	131,07
	Huge	Enh		>3hours		118,54	131,73	146,86	>3hours			112,94	118,20	122,66
VBDC	None	Ref	126,65	126,68	126,69	95,39	96,31	97,34	110,53	110,54	110,55	97,36	98,78	100,05
	None	Enh	100,14	100,15	100,17	105,55	108,62	114,82	87,62	88,29	90,48	92,97	96,83	103,98
	Low	Ref	200,18	229,06	253,26	108,07	110,37	114,52	103,69	108,49	115,01	101,13	103,56	105,41
	Low	Enh	137,92	147,77	156,14	109,92	112,50	118,67	91,83	96,13	99,55	95,06	99,31	103,62
	Med	Ref	351,65	400,74	428,87	120,86	124,98	138,97	118,82	135,19	147,22	106,45	109,85	116,19
	Med	Enh	234,82	256,29	291,03	109,04	114,78	127,31	101,51	109,74	120,35	99,84	103,45	111,82
	High	Ref	451,97	493,56	551,65	122,68	134,23	149,69	149,80	162,77	195,18	113,13	118,94	124,66
	High	Enh	340,80	369,50	402,14	113,75	121,87	132,38	109,56	125,76	140,68	105,17	108,52	117,27
	VHigh	Ref		557,26		136,04	139,76	166,18	158,32	178,98	201,33	119,74	125,71	131,12
	VHigh	Enh		419,87		118,01	124,43	132,76	120,63	130,57	143,41	108,49	112,79	119,72
	Huge	Ref		1559,18		158,78	178,67	226,87	245,52	270,51	347,66	137,71	144,94	163,53
	Huge	Enh		675,34		144,07	162,19	177,96	153,38	172,85	213,58	121,78	130,54	135,76

The DAASS mechanism is implemented by enabling the Quick ACKs in Linux. With Quick ACKs, Linux TCP receiver acknowledges every segment in the beginning of the connection until the threshold equaling a half of the receiver advertised window is reached. In addition, a subset of tests is repeated using the TCP-CBI with the *ssthresh* reuse.

A single TCP connection is used for a bulk data transfer of 1 MB from H1 to H2 (Return Link). We choose the Return Link direction as it is more challenging to TCP due to the DAMA mechanism. We repeated the same tests transferring the same amount of data with four parallel TCP connections (4x250 kbytes) to verify the results in presence of competing TCP flows. Each test case is replicated 32 times.

5 RESULTS

Table 2 presents the TCP performance results for one TCP flow and for four competing TCP flows. We report the median elapsed time as well as lower and upper quartiles (25th and 75th percentiles) over 32 replications. The test cases with the "Very High" and "Huge" error models and without SLACP have been executed only with a reduced amount of replications as the resulting performance turned out to be very

poor and the time needed to complete the tests became very long. Only the median values over 5 replications are reported. With RBCD we were not even able to complete the transfers within 3 hours when the "Huge" error model is in use without SLACP.

When a single TCP flow is used without SLACP, the Enhanced TCP performs better than the Reference TCP for all error models and for all DAMA schemes. When the error rate becomes higher, the performance gain with the Enhanced TCP clearly increases. When the DAMA allocation delay increases, the elapsed time increases for both TCP variants, indicating that the TCP performance is strongly affected by the increasing delay; the long RTT slows down TCP loss recovery and the congestion window increase. However, the Enhanced TCP suffers less from the increased delay as it is able to fill up the available bandwidth faster and is able to recover from multiple losses within a few round trips.

Even on an error-free link the Enhanced TCP is slightly more efficient than the Reference TCP as it is able to inflate the congestion window faster due to the larger initial window and the DAASS mechanism. This difference is more significant when the transfer size is small, like in most Web transfers. Table 3 shows the median elapsed times for transferring the first 50 kbytes with a single TCP connection and

employing VBDC. The elapsed time of the Reference TCP is roughly 50 % longer in all cases.

Table 3: Median elapsed times for short transfers using a single TCP flow (with VBDC, 32 replications)

	Transfer Size	TCP	Error model			
			None	Low	Med	High
NOSLACP	50KB	Ref	16,58	16,63	18,75	26,03
		Enh	10,51	10,73	11,66	14,66
SLACP	50KB	Ref	15,83	16,57	18,89	19,00
		Enh	10,48	11,40	11,42	12,78

When a single TCP flow is used, employing the SLACP protocol improves TCP performance considerably, especially at high error rates. This is true for both Reference and Enhanced TCP. When the error rate increases to high values, it becomes obvious that Enhanced TCP alone is not able to provide adequate performance; the Reference TCP with SLACP is able to provide significantly better performance compared to Enhanced TCP without SLACP. Even on an error-free link and at low error rate (error models "None" and "Low"), employing SLACP with the Reference TCP is beneficial. This is because of flow control between the IP and the MAC layer. Without SLACP, IP packets flow directly to the MAC buffer. This prevents from utilizing the IP level buffering and the slow-start overshoot occurs earlier. In addition, packets are frequently dropped also in later phases of the connection due to MAC buffer congestion. With SLACP, the new arriving packets are kept in the IP buffer if no space is available in the MAC buffer. This does not avoid packet drops in the case of congestion (IP buffer may also overflow) but allows TCP to keep up larger congestion window with the help of the additional IP-level buffering capacity.

Table 2 shows that without the SLACP and with the RBDC allocation scheme the elapsed time of a single TCP flow grows steeply when the error rate increases, resulting in very poor performance. The resulting performance is even lower than with VBDC. This is because the TCP congestion window remains small due to the high error rate. With RBDC, the amount of bandwidth allocated is based on the instantaneous rate requests. As the TCP sender is sending only an occasional burst with a small number of packets that arrive at the MAC buffer one by one, the resulting RBDC allocation request is made for the data in the MAC buffer comprising of a few packets only. The amount of data in the MAC buffer is less than the ceiling value and a single RBDC allocation cycle is not enough to flush the MAC buffer once all packets in the burst have arrived. Furthermore, since the allocated bandwidth is low the packets arrive at the TCP receiver at very low rate, triggering TCP ACKs at low rate too. The problematic behavior continues as the low rate of TCP ACKs is not able to trigger new data packets fast enough so that more packets could accumulate in

the MAC buffer and result in an RBDC allocation request with more bandwidth. The Enhanced TCP is able to keep up larger congestion window and recover from the error losses more efficiently than the Reference TCP. Hence, it suffers little of this phenomenon. When SLACP is used, TCP performance with RBDC is not adversely affected since the SLACP drastically reduces the residual packet error rate as seen by TCP.

In the case of multiple competing flows the TCP performance is better than in case of single TCP flow since the four parallel TCP connections act more aggressively and the TCP loss recovery performs significantly better than with a single TCP flow. The TCP loss recovery works efficiently because usually not all simultaneous TCP flows suffer from packet losses at the same time and therefore some of the TCP flows may continue utilizing the link while the other flows are recovering from losses and proceed slowly. Consequently, the performance gain with the SLACP protocol is not as significant as for one TCP flow. The usefulness of SLACP becomes evident when error rate is high. With CRA, the Reference TCP with SLACP starts yielding shorter elapsed times than the Enhanced TCP without SLACP when the error rate becomes very high (the error models "Very High" and "Huge"). With lower error rates the Enhanced TCP alone performs better than either of the TCP variants with SLACP bacause the SLACP overhead slightly degrades performance and the SLACP recovery does not provide additional benefit over the TCP recovery. On the other hand, with RBDC and VBDC that introduce longer RTTs, the TCP loss recovery is less efficient and using SLACP with multiple simultaneous connections becomes useful already with the "High" error model.

Combining Enhanced TCP with SLACP yields additional performance gain for a single TCP flow compared to the other cases, except in the case of error-free link. In the presence of errors, the combined approach (Enhanced TCP with SLACP) performs drastically better than the baseline approach (Reference TCP without SLACP). A single TCP flow on an error-free link and at low error rates with SLACP starts to suffer from the slow-start overshoot problem (Dawkins et al., 2001a). For example, the Reference TCP performs slightly better than Enhanced TCP when SLACP is used as the latter experiences a larger number of congestion-related losses during the slow-start overshoot. This problem is more serious with the Enhanced TCP as it uses a larger initial window and the DAASS mechanism being more aggressive during the initial slow-start phase and therefore having larger number of outstanding packets at the time when the router buffer at the ST overflows. The recovery from the large number of lost packets in a single TCP window lasts very long as there are not enough duplicate ACKs with SACK information

arriving at the TCP sender.

The slow-start overshoot occurs also with the four simultaneous TCP flows. Now it affects the Enhanced TCP also when used without SLACP. The Enhanced TCP yields longer elapsed time than the Reference TCP when using an error-free link or the "Low" error model with CRA and RBDC. On the other hand, with VBDC the Enhanced TCP performs better than the Reference TCP also at low error conditions. This happens because the Enhanced TCP does not experience the VBDC bandwidth allocation delay as it is able to fill up the available satellite link bandwidth already during the initial RTT with the four connections injecting 16 segments to the network. The total number of the outstanding packets remains high enough to avoid the VBDC allocation delay for entire duration of the transfer. The Reference TCP starts with four segments only and it takes several RTTs with the VBDC allocation delay on each RTT before it is able utilize the available bandwidth. In addition, the Reference TCP suffers heavily from the VBDC allocation delay towards the end of the transfer. Typically one of the four TCP flows is proceeding very slowly with a very small congestion window. When the other flows complete, the last flow continues alone with the small window and is forced to experience the VBDC allocation delay on each of its remaining RTTs.

In the case of four TCP flows with SLACP the slow-start overshoot is the other dominating factor in addition to the SLACP overhead that keeps the elapsed times fairly long with both TCP variants when the error rates are not high. At higher error rates the first packet losses occur already during the initial slow start, terminating the slow start and preventing the slow start overshoot.

Figure 3 shows an example trace for the slow-start overshoot with the Enhanced TCP over an error-free link using the VBDC DAMA scheme. The graph indicates the packet trace of a single TCP flow from the sender point of view. The sequence numbers of the sent packets, acknowledgements and retransmissions are shown. The overshoot starts roughly at t = 10 sec. The first loss is detected with the three duplicate ACKs arriving roughly at t = 15 sec, when the sender has transmitted approximately 250 Kbytes. A large number of packets are lost between t = 10 sec and t = 15 sec and the recovery phase lasts over 30 seconds.

It turned out that the slow-start overshoot remained the only major problem with the combined approach of using SLACP and Enhanced TCP. In order to attack the slow-start overshoot problem, we used two modified versions of Enhanced TCP. The first version uses the TCP Control Block Interdependence (CBI), which allows the TCP sender to reuse the *ssthresh* of the previous connection. The use of CBI can be better justified when there are little or no uncorrected link
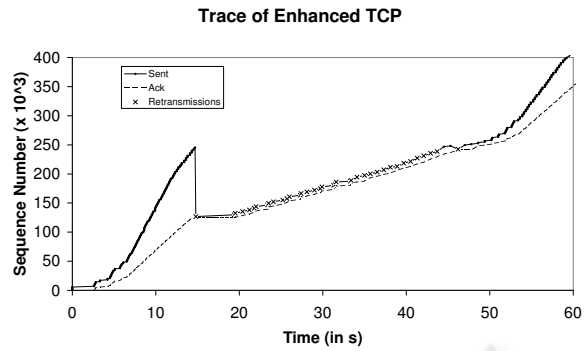

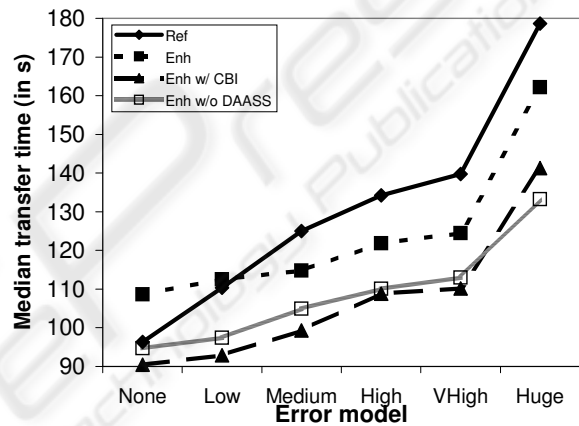Figure 3: Example of slow-start overshoot

Figure 4: Comparison of Enhanced TCP variants

errors (which is the case when SLACP is used). The second version disables the DAASS mechanism, so that it is less aggressive during the initial slow-start phase. Figure 4 compares the median elapsed times of a single TCP flow with these two Enhanced TCP versions included and when SLACP is employed over the satellite link with VBDC DAMA scheme. Table 4 summarizes the medians as well the lower and upper quartiles for the elapsed times.

In general, the two new Enhanced TCP versions outperform the original Enhanced TCP as well as the Reference TCP with all error models. The two new Enhanced TCP versions perform the best because disabling DAASS in the Enhanced TCP mitigates the slow-start overshoot problem whereas the Enhanced TCP with CBI prevents the overshoot problem from occurring even though DAASS mechanism is enabled with this TCP version. On the other hand, Enhanced TCP with CBI misbehaves in the presence of error related losses as the initial *ssthresh* value derived from the previous connection tends to have too low value, forcing the TCP sender to enter congestion avoidance

Table 4: Comparison of Enhanced TCP variants (with SLACP and VBDC, 32 replications)

Error Model	TCP	Transfer Time for 1MB		
		25perc	Median	75perc
None	Ref	95,39	**96,31**	97,34
	Enh	105,55	**108,62**	114,82
	Enh w/ CBI	89,93	**90,43**	92,01
	Enh w/o DAASS	93,37	**94,69**	95,74
Low	Ref	108,07	**110,37**	114,52
	Enh	109,92	**112,50**	118,67
	Enh w/ CBI	92,17	**92,87**	94,31
	Enh w/o DAASS	95,41	**97,45**	98,81
Med	Ref	120,86	**124,98**	138,97
	Enh	109,04	**114,78**	127,31
	Enh w/ CBI	96,71	**99,36**	102,89
	Enh w/o DAASS	101,96	**104,94**	107,49
High	Ref	128,68	**134,23**	149,69
	Enh	113,75	**121,87**	132,38
	Enh w/ CBI	104,63	**108,82**	112,80
	Enh w/o DAASS	107,65	**110,11**	117,78
VHigh	Ref	136,04	**139,76**	166,18
	Enh	118,01	**124,43**	132,76
	Enh w/ CBI	107,8	**110,17**	111,73
	Enh w/o DAASS	109,89	**113,01**	120,05
Huge	Ref	158,78	**178,67**	226,87
	Enh	144,07	**162,19**	177,96
	Enh w/ CBI	131,68	**141,25**	166,65
	Enh w/o DAASS	130,18	**133,22**	146,23

too early. Therefore, the Enhanced TCP with CBI is performing worse than the Enhanced TCP without DAASS in the "Huge" error model case, where the SLACP protocol is not able to recover from all frame losses. One additional reason for the suboptimal performance of the Reference TCP with SLACP is spurious TCP RTOs. The Reference TCP is suffering from the spurious TCP RTOs in all error models except with the "None" and "Low" error models. The additional delay due to the link-level error recovery introduces an occasional delay spike, resulting in unnecessary retransmission of the last TCP window. The F-RTO algorithm employed with the Enhanced TCP variant is able to detect the most of the spurious RTOs and thereby avoid the unnecessary retransmissions.

The results suggest that either of the Enhanced TCP variants should be selected for use in order to receive the best TCP performance. The fact that it is very hard in general for a TCP sender to gauge in advance whether a new TCP connection would experience error-related losses on the end-to-end path and therefore possibly incur suboptimal behavior with the TCP CBI is likely to elevate the Enhanced TCP without DAASS to the recommended Enhanced TCP variant.

6 CONCLUDING REMARKS AND FUTURE WORK

This paper proposes the use of a TCP/IP-friendly link-level error recovery mechanism in conjunction with a number of the state-of-the-art TCP enhancements to improve TCP performance over W-WAN links. While the proposed link-level error recovery mechanism for W-WAN links allows efficient TCP operation over error-prone wireless links it minimizes the additional delay due to the ARQ by adding FEC-encoded redundancy in the retransmitted frames in a novel way. A TCP/IP-friendly link layer implementation also incorporates flow control between IP and MAC layer which limits the amount of the link buffering and allows the use of queue management at IP layer.

We ran an extensive set of experiments to evaluate the performance of a selected set of TCP enhancements together with our implementation of the TCP/IP-friendly link layer, called SLACP. We compared the performance of a regular version and enhanced version of TCP over a satellite link, with and without SLACP. In the experiments we used real TCP implementations on Linux end hosts and a satellite emulator platform representative of a DVB-RCS satellite system.

The results show that both SLACP and TCP enhancements are beneficial on their own. However, the TCP enhancements alone cannot cope with high error rates efficiently, especially in case of one TCP flow. On the other hand, employing SLACP over the error prone satellite link yields very significant performance gain. The combined approach of the TCP enhancements and SLACP yields the best performance. We also found that Enhanced TCP performance suffers from the slow-start overshoot phenomenon when the residual link-error rate is zero. Introducing TCP CBI mechanism for sharing the slow start threshold from previous TCP connection is helpful to attack the problem in most cases. Alternatively, disabling the DAASS mechanism from the set of the selected TCP enhancements turned out to be equally worthwhile solution.

In the future we intend to study the performance of the SLACP approach with different levels of FEC as well as with different ARQ persistency levels. We also intend to seek for alternative approaches to the TCP slow-start overshoot problem. In addition, incorporating an explicit loss notification to the SLACP seems a promising approach as with a novel design in the SLACP we avoid the common problem of not being able to reliably infer the TCP sender to notify when a frame is corrupted on the link. This is not a problem with the SLACP as the link sender keeps the original copy of each TCP segment and the link receiver informs it whether or not the corresponding frame was successfully delivered.

ACKNOWLEDGEMENTS

REFERENCES

Akyildiz, I. F., Morabito, G., and Palazzo, S. (2001). TCP-Peach: a new congestion control scheme for satellite IP networks. *IEEE/ACM Transactions on Networking*, 9(3):307–321.

Allman, M., Balakrishnan, H., and Floyd, S. (2001). Enhancing TCP's Loss Recovery Using Limited Transmit. RFC 3042, Internet Society.

Allman, M., Ed., Dawkins, S., Glover, D., Griner, J., Tran, D., Henderson, T., Heidemann, J., Touch, J., Kruse, H., Ostermann, S., Scott, K., and Semke, J. (2000). Ongoing TCP Research Related to Satellites. RFC 2760, Internet Society.

Allman, M., Floyd, S., and Partridge, C. (2002). Increasing TCP's Initial Window. RFC 3390, Internet Society.

Allman, M., Paxson, V., and Stevens, W. (1999). TCP Congestion Control. RFC 2581, Internet Society.

Balakrishnan, H., Padmanabhan, V. N., Seshan, S., and Katz, R. H. (1997). A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769.

Blanton, E., Allman, M., Fall, K., and Wang, L. (2003). A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP. RFC 3517, Internet Society.

Border, J., Kojo, M., Griner, J., Montenegro, G., and Shelby, Z. (2001). Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. RFC 3135, Internet Society.

Dawkins, S., Montenegro, G., Kojo, M., and Magret, V. (2001a). End-to-end Performance Implications of Slow Links. RFC 3150, Internet Society.

Dawkins, S., Montenegro, G., Kojo, M., Magret, V., and Vaidya, N. (2001b). End-to-end Performance Implications of Links with Errors. RFC 3155, Internet Society.

ETSI (2003). Digital Video Broadcasting (DVB): Interaction Channel for Satellite Distribution System. Technical Specification EN-301-790, ETSI.

Fairhurst, G. and Wood, L. (2002). Advice to link designers on link Automatic Repeat reQuest (ARQ). RFC 3366, Internet Society.

Floyd, S. (2000). Congestion Control Principles. RFC 2914, Internet Society.

Floyd, S. and Henderson, T. (1999). The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582, Internet Society.

Jacobson, V., Braden, R., and Borman, D. (1992). TCP Extensions for High Performance. RFC 1323, Internet Society.

Karn, P. (2004). Advice for Internet Subnetwork Designers. RFC 3819, Internet Society.

Kojo, M., Astuti, D., Daniel, L., Nyrhinen, A., and Raatikainen, K. (2004). Enhancing TCP Performance Over Satellite Networks - A TCP/IP-Friendly Link-Layer Approach. Technical Report C-2004-50, University of Helsinki, Department of Computer Science.

Ludwig, R. and Katz, R. (2000). The Eifel algorithm: Making TCP robust against spurious retransmissions. *ACM SIGCOMM Computer Communication Review*, 30(1):30–36.

Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M. Y., and Wang, R. (2001). TCP Westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 287–297. ACM Press.

Mathis, M., Mahdavi, J., Floyd, S., and Romanow, A. (1996). TCP Selective Acknowledgement Options. RFC 2018, Internet Society.

Padhye, J., Firoiu, V., Towsley, D., and Kurose, J. (1998). Modeling TCP throughput: a simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 303–314. ACM Press.

Sarolahti, P. and Kojo, M. (2004). F-RTO: An Algorithm for Detecting Spurious Retransmission Timeouts with TCP and SCTP. Internet Draft. Work in progress.

Sarolahti, P., Kojo, M., and Raatikainen, K. (2003). F-RTO: An enhanced recovery algorithm for TCP retransmission timeouts. *ACM SIGCOMM Computer Communication Review*, 33(2):51–63.

Touch, J. (1997). TCP Control Block Interdependence. RFC 2140, Internet Society.