http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 6 (3): 363-369, 2007
ISSN 1812-5638
© 2007 Asian Network for Scientific Information

Wireless Network: Performance Analysis of TCP

[1]K.H. Walse and [2]D.R. Dhotre
[1]Department of Information Technology, Anuradha Engineering College,
Chikhli, SGB Amravati University (M.S.) India
[2]Department of Computer Science and Engineering, Anuradha Engineering College,
Chikhli, SGB Amravati University (M.S.) India

Abstract: In traditional networks (wired) TCP are tuned to perform well where packet losses occur due to congestion. In wireless networks losses can results from bit errors, fading and handoffs. There have been many schemes proposed to improve the performance of the TCP over network that have high BER wireless link. In this research, we compare two techniques designed to improve the performance of TCP in such networks, that have been proposed and simulated by various authors and we give analytical reviews on these proposals. These schemes are link-layer protocol that provides local reliability and split-connection protocol, that break the end-to-end connection into two parts at the base station. We discuss that some TCP sender adaptation work well for certain environments only, while others are generally useful. TCP responds to all losses by invoking congestion control and avoidance algorithms, resulting in degraded end-to-end performance in wireless and lossy systems. It is shown that a reliable link-layer protocol that is TCP-aware provides very good performance by shielding the TCP sender from duplicate acknowledgment caused by wireless losses improves throughput by 10-30%. Furthermore, it is possible to achieve good performance without splitting the end-to-end connection at the base station.

Key words: TCP (Transmission Control Protocol), wireless link, wired link, congestion, link-layer, split-connection

INTRODUCTION

Due to the low loss nature of wired link, TCP assumes that packets losses mostly occur due to congestion. TCP reacts to packet losses by decreasing its transmission (congestion) window size thus reducing network utilization. In wireless network however losses occur due to the high bit error (BER) or transmission medium or due to fading and mobility. TCP still reacts to losses according to its congestion control scheme. Several studies (Xu and Saadwi, 2001; Balakrishnana *et al*., 1997; Kim *et al*., 2001; Chandran *et al*., 2001) have shown that TCP cannot handle mobility well. There are two different approaches given by (Balakrishnan *et al*., 1997) to improving TCP performance in such lossy systems. The first approach hides any non-congestion-related losses from the TCP sender and therefore requires no changes to existing sender implementations. The intuition behind this approach is that since the problem is local, it should be solved locally and that the transport layer need not be aware of the characteristics of the individual links. Protocols that adopt this approach attempt to make the lossy link appear as a higher quality link with a reduced effective bandwidth. As a result, most of the losses seen by the TCP sender are caused by congestion. The second class of techniques attempts to make the sender aware of the existence of wireless hops and realize that some packet losses are not due to congestion. The sender can then avoid invoking congestion control algorithms when non-congestion-related losses occur. Finally, it is possible for a wireless-aware transport protocol to coexist with link-layer schemes to achieve good performance. The many schemes are classify into three basic groups as per (Balakrishnana *et al*., 1997) based on their fundamental philosophy: Split-connection proposals and link-layer proposals. Split-connection approaches completely hide the wireless link from the sender by terminating the TCP connection at the base station. The second connection can use techniques such as negative or selective acknowledgments, rather than just standard TCP to perform well over the wireless link. The link-layer protocols attempt to hide link related losses from the TCP sender by using local retransmissions and perhaps forward error correction over the wireless link. The local

Corresponding Author: D.R. Dhotre, Department of Computer Science and Engineering, Anuradha Engineering College, Chikhli, SGB Amravati University (M.S.) India

retransmissions use techniques that are tuned to the characteristics of the wireless link to provide a significant increase in performance. Since the end-to-end TCP connection passes through the lossy link, the TCP sender may not be fully shielded from wireless losses. This can happen either because of timer interactions between the two layers or more likely because of TCP's duplicate acknowledgments causing sender fast retransmissions even for segments that are locally retransmitted. As a result, some proposals to improve TCP performance use mechanisms based on the knowledge of TCP messaging to shield the TCP sender more effectively and avoid competing and redundant retransmissions.

TCP RESPONSE TO PACKET LOSS

TCP assumes that packet loss caused by damage is very small generally much less than 1%. Therefore for TCP, a packet loss generally signifies the presence of congestion between the sender and the receiver a typical loss situation is shown in Fig. 1. To avoid these packet losses and improve the efficiency of transmission, TCP respond to packet losses by reducing the transmission rate. This is implemented by reducing the congestion window (*cwnd*). The number of unacknowledged packets that a sender can send into network is the minimum of the current *cwnd* and the receivers advertised window. If *cwnd* is small then fewer packets can send during a time period. TCP sender can detect a packet loss through two ways (1) a time out and (2) duplicate acknowledgments (ACKs). A time out occurs when the sender does not receive any ACK from the receiver within a prescribed time, when a time-out occurs TCP interprets this as severe congestion in the network. So it sets the slow start threshold (*ssthresh*) to half of the minimum of the current *cwnd* and the receivers advertised window. Then it decreases the *cwnd* to 1 segment and performs a slow start. During slow start, the TCP sender increases *cwnd* by 1 segment every time a good ACK is received. This tends to an exponential increase in *cwnd* during slow start. When the *cwnd* reaches the *ssthresh*, it leaves slow start and switches to congestion avoidance when every new ACK increments cwnd by 1 segment.

When duplicate ACK are received, TCP interprets this as less severe congestion. When the third duplicate ACK is received, TCP goes into the fast retransmit and fast recovery algorithm. The *ssthresh* is set to half of the minimum *cwnd* and receivers advertised window. The TCP sender retransmits the lost packet immediately without waiting for the retransmission time-out. During this stage it considers every duplicate ACK as an acknowledgment of an out-of-order segment so to increase the *cwnd* correspondingly and continues to send packets. When the ACK of the retransmitted packet is finally received, it goes to fast recovery.

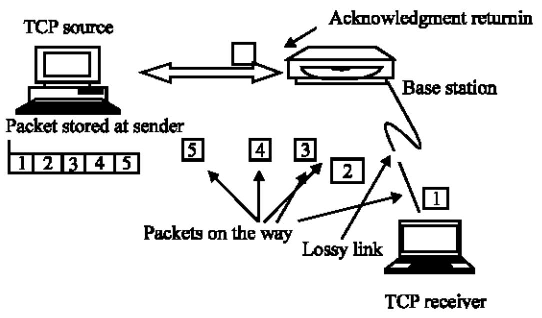

Fig. 1: A typical loss situation

EXPERIMENTAL METHODOLOGY

Several experiments (Baklakrishnan *et al*., 1997) are performed to determine the performance and efficiency of each of the protocols. The protocols were implemented as a set of modifications to the BSD/OS TCP/IP (Reno) network stack. To ensure a fair basis for comparison, none of the protocols implementations introduce any additional data copying at intermediate points from sender to receiver. Experimental test bed consists of IBM ThinkPad laptops and Pentium-based personal computers running BSD/OS 2.1 from BSDI. The machines are interconnected using a 10 Mbps Ethernet and 915 MHz AT andT WaveLANs, a shared-medium wireless LAN with a raw signaling bandwidth of 2 Mbps. The network topology for experiment as shown below in Fig. 2.

The peak throughput for TCP bulk transfers is 1.5 Mbps in the local area testbed and 1.35Mbps in the wide area testbed in the absence of congestion or wireless losses. These testbed topologies represent typical scenarios of wireless links and mobile hosts, such as cellular wireless networks. In addition, experiments focus on data transfer to the mobile host, which is the common case for mobile applications.

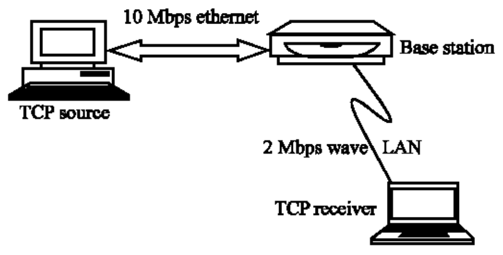

Fig. 2: Experimental methodology

LINK-LAYER PROTOCOL TECHNIQUE

Several proposals are there for reliable link-layer protocols. The two main classes of techniques employed by these protocols are: Error correction, using techniques such as Forward Error Correction (FEC) and retransmission of lost packets in response to Automatic Repeat Request (ARQ) messages. The link-layer protocols for the digital cellular systems in the U.S- both CDMA and TDMA primarily use ARQ techniques. While the TDMA protocol guarantees reliable, in-order delivery of link-layer frames, the CDMA protocol only makes a limited attempt and leaves eventual error recovery to the (reliable) transport layer. Other protocols like the AIRMAIL protocol employ a combination of FEC and ARQ techniques for loss recovery.

The main advantage of employing a link-layer protocol for loss recovery is that it fits naturally into the layered structure of network protocols. The link-layer protocol operates independently of higher-layer protocols and does not maintain any per-connection state. The main concern about link-layer protocols is the possibility of adverse effect on certain transport-layer protocols such as TCP.

Unlike TCP for the transport layer, there is no de facto standard for link-layer protocols. Existing link-layer protocols choose from techniques such as Stop-and-Wait, Go-Back-N, Selective Repeat and Forward Error Correction to provide reliability. The base link-layer algorithm given by Balakrishnan *et al.* (1997) called LL, uses cumulative acknowledgments to determine lost packets that are retransmitted locally from the base station to the mobile host. To minimize overhead, implementation of LL leverages off TCP acknowledgments instead of generating its own. Timeout-based retransmissions are done by maintaining a smoothed round-trip time estimate, with a minimum timeout granularity of 200 ms to limit the overhead of processing timer events. This still allows the LL scheme to retransmit packets several times before a typical TCP Reno transmitter would time out. LL is equivalent to the snoop agent that does not suppress any duplicate acknowledgments and does not attempt in-order delivery of packets across the link.

While the use of TCP acknowledgments by LL protocol renders it atypical of traditional ARQ protocols, it still preserves the key feature of such protocols: the ability to retransmit packets locally, independently of and on a much faster time scale than TCP. More sophisticated link-layer protocol investigated (Balakrishnan *et al.*, 1997) is (LL-SMART) that uses selective retransmissions to improve performance. The LL-SMART protocol performs this by applying a SMART-based acknowledgment scheme at the link layer. Like the LL protocol, LL-SMART uses TCP acknowledgments instead of generating its own and limits its minimum timeout to 200 ms. LL-SMART is equivalent to the snoop agent performing retransmissions based on selective acknowledgments but not suppressing duplicate acknowledgments at the base station.

They have added TCP awareness to both the LL and LL-SMART protocols, resulting in the LL-TCP-AWARE and LLSMART-TCP-AWARE schemes. The LL-TCP-AWARE protocol is identical to the snoop protocol, while the LLSMART-TCP-AWARE protocol uses SMART-based techniques for further optimization using selective repeat. LLSMART-TCP-AWARE link-layer protocol results best in experiments - it performs local retransmissions based on selective acknowledgments and shields the sender from duplicate acknowledgments caused by wireless losses.

Experimental results: Traditional link-layer protocols operate independently of the higher-layer protocol and consequently, do not necessarily shield the sender from the lossy link. In spite of local retransmissions, TCP performance could be poor for two Reasons: (i) competing retransmissions caused by an incompatible setting of timers at the two layers and (ii) unnecessary invocations of the TCP fast retransmission mechanism due to out-of-order delivery of data. In experiment done by Dessimone *et al.* (1993) the effects of the first situation are simulated and analyzed for a TCP-like transport protocol (that *closely* tracks the round-trip time to set its retransmission timeout) and a reliable link-layer protocol. The conclusion was that unless the packet loss rate is high (more than about 10%), competing retransmissions by the link and transport layers often lead to significant performance degradation. However, this is not the dominating effect when link layer schemes, such as LL, are used with TCP Reno and its variants. These TCP implementations have coarse retransmission timeout granularities that are typically multiples of 500 ms, while link-layer protocols typically have much finer timeout granularities. The real problem is that when packets are lost, link-layer protocols that do not attempt in-order delivery across the link (e.g., LL) cause packets to reach the TCP receiver out-of-order. This leads to the generation of duplicate acknowledgments by the TCP receiver, which causes the sender to invoke fast retransmission and recovery. This can potentially cause degraded throughput and goodput, especially when the delay-bandwidth product is large.

Result of authors (H. Balakrishnan *et al.*, 1997) substantiates this claim, as can be seen by comparing the LL and LL-TCP-AWARE results (Fig. 3 and Table 1). For a packet size of 1400 bytes, a bit error rate of 1.9×10^{-6}

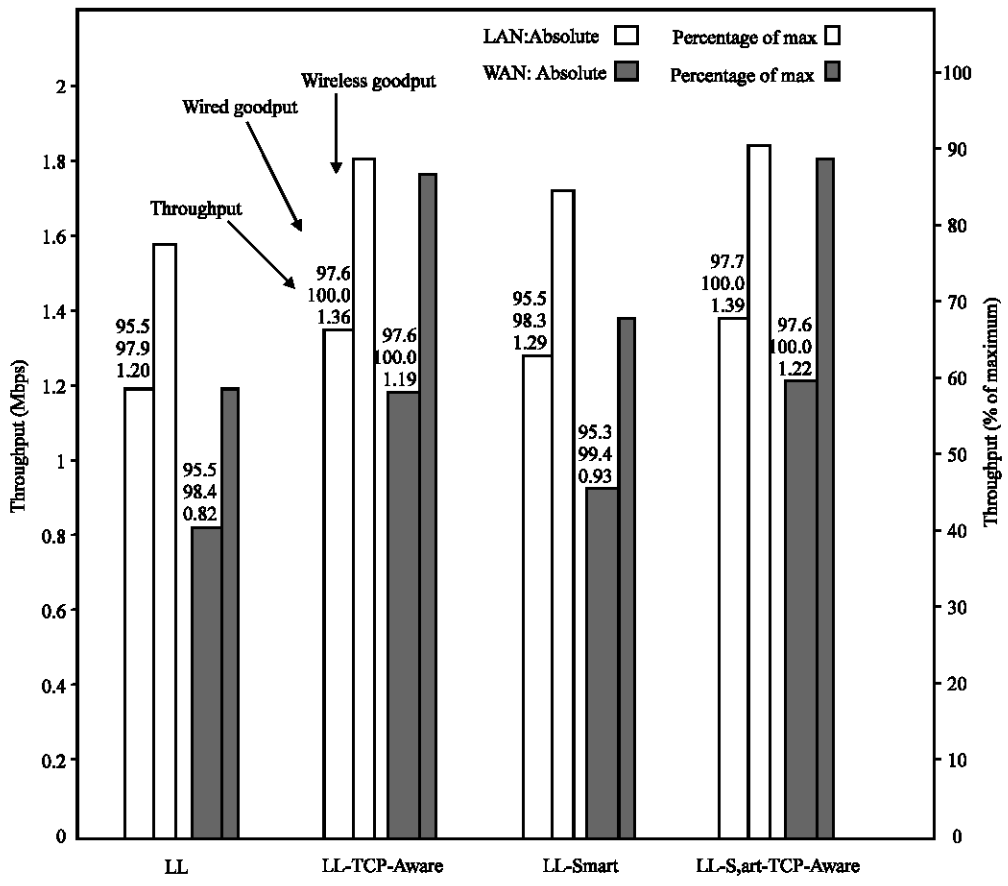

Fig. 3: Performance of link-layer protocols: bit-error rate = 1.9×10^{-6} (1 error/65536 bytes), socket buffer size = 32 KB. For each case there are two bars: the thick one corresponds to the scale on the left and denotes the throughput in Mbps; the thin one corresponds to the scale on the right and shows the throughput as a percentage of the maximum, i.e., in the absence of wireless errors (1.5 Mbps in the LAN environment and 1.35 Mbps in the WAN environment)

Table 1: Results for the link-layer schemes for an average error rate of one every 65536 bytes of data. Each entry is of the form: Throughput wireless goodput, wired goodput). Throughput is measured in Mbps. Goodput is expressed as a percentage

	LL	LL-TCP-aware	LL-smart	LL-smart-TCP-aware
LAN (8 KB)	1.20 (95.6, 97.9%)	1.29 (97.6, 100%)	1.29 (96.1, 98.9%)	1.37 (97.6, 100%)
LAN (32 KB)	1.20 (95.5, 97.9%)	1.36 (97.6, 100%)	1.29 (95.5, 98.3%)	1.39 (97.7, 100%)
WAN (32 KB)	0.82 (95.5, 98.4%)	1.19 (97.6, 100%)	0.93 (95.3, 99.4%)	1.22 (97.6, 100%)

(1/65536 bytes) translates to a packet error rate of about 2.2 to 2.3%. Therefore, an optimal link-layer protocol that recovers from errors locally and does not compete with TCP retransmissions should have a wireless goodput of 97.7% and a wired goodput of 100% in the absence of congestion.

In the LAN experiments, the throughput difference between LL and LL-TCP-AWARE is about 10%. However, the LL wireless goodput is only 95.5%, significantly less than LL-TCP-AWARE's wireless goodput of 97.6%, which is close to the maximum achievable goodput. When a loss occurs, the LL protocol performs a local retransmission relatively quickly. However, enough packets are typically in transit to create more than 3 duplicate acknowledgments. These duplicates eventually propagate to the sender and trigger a fast retransmission and the associated congestion control mechanisms. These fast retransmissions result in reduced goodput; about 90% of the lost packets are retransmitted by both the source and the base station. The effects of this interaction are much more pronounced in the wide-area experiments-the throughput difference is about 30% in this case. The cause for the more pronounced deterioration in performance is the higher bandwidth-delay product of the wide-area connection. The LL scheme causes the sender to invoke congestion

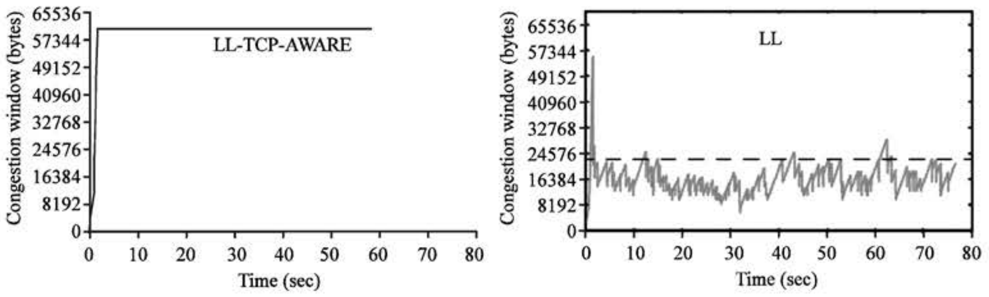

Fig. 4: Congestion window size for link-layer protocols in wide area tests. The horizontal dashed line in the LL graph shows the 23000 byte WAN bandwidth-delay product

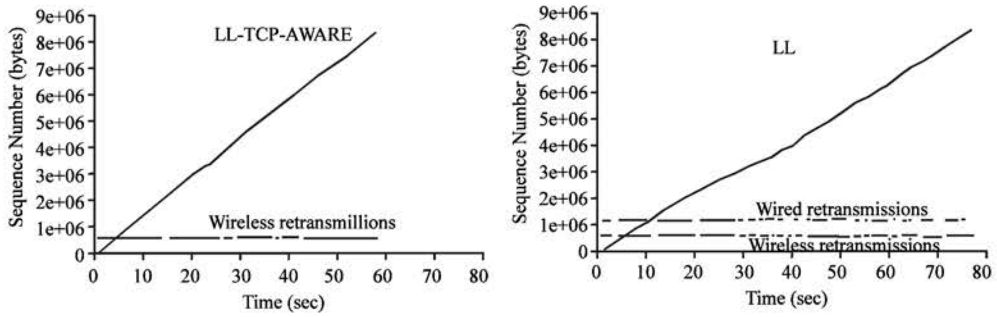

Fig. 5: Packet sequence traces for LL-TCP-AWARE and LL. No. Coarse rimeouts occur in either case. For LL-TCP-AWARE, the horizontal row of dots shows the times of wireless link retransmissions. For LL, the top row shows sender fast retransmission times and the bottom row shows both local wireless and sender retransmissions

control procedures often due to duplicate acknowledgments and causes the average window size of the transmitter to be lower than for LL-TCPAWARE. This is shown in Fig. 4, which compares the congestion window size of LL and LL-TCP-AWARE as a function of time. Note that the number of outstanding data bytes in the network is the minimum of the congestion window and the receiver advertised window. This is bounded by the receiver's socket buffer size. In the congestion window graphs for each protocol, the receiver socket buffer is 32 KB. In the wide area, the bandwidth-delay product is about 23000 bytes (1.35 Mbps * 135 ms) and the congestion window drops below this value several times during each TCP transfer. On the other hand, the LAN experiments do not suffer from such a large throughput degradation because LL's lower congestion-window size is usually still larger than the connection's delay-bandwidth product of about 1900 bytes (1.5 Mbps * 10 ms). Therefore, the LL scheme can maintain a nearly full "data pipe" between the sender and receiver in the local connection but not in the wide area one. The 10% LAN degradation is almost entirely due to the excessive retransmissions over the wireless link and to the smaller average congestion window size compared to LLTCP-AWARE. Another important point to note is that LL successfully prevents coarse timeouts from happening at the source. Figure 5 shows the sequence traces of TCP transfers for LL-TCP-AWARE and LL.

In summary, the results indicate that a simple link-layer retransmission scheme does not entirely avoid the adverse effects of TCP fast retransmissions and the consequent performance degradation. An enhanced link-layer scheme that uses knowledge of TCP semantics to prevent duplicate acknowledgments caused by wireless losses from reaching the sender and locally retransmits packets achieves significantly better performance.

SPLIT CONNECTION TECHNIQUE

Split connection protocols split each TCP connection between a sender and receiver into two separate connections at the base station-one TCP connection between the sender and the base station and the other between the base station and the receiver. Over the wireless hop, a specialized protocol tuned to the wireless environment may be used. Yavatkar and N Bhagwat (1994) propose two protocols-one in which the wireless hop uses TCP and another in which the wireless hop uses a Selective Repeat Protocol (SRP) on top of UDP. They study the impact of handoffs on performance and conclude that they obtain no significant advantage by using SRP instead of TCP over the wireless connection in their experiments. However, experiment of authors (Balakrishnan *et al.*, 1997) demonstrate benefits in using a simple selective acknowledgment scheme with TCP over the wireless connection. Indirect-TCP is a split-connection solution that uses standard TCP for its connection over the wireless link. Like other split-connection proposals, it attempts to separate loss recovery over the wireless link from that across the wireline network, thereby shielding the original TCP sender from the wireless link. However, as experiments (Balakrishnan *et al.*, 1997) indicate, the choice of TCP over the wireless link results in several performance problems. Since TCP is not well-tuned for the lossy link, the TCP sender of the wireless connection often times out, causing the original sender to stall. In addition, every packet incurs the overhead of going through TCP protocol processing twice at the base station (as compared to zero times for a non-split-connection approach), although extra copies are avoided by an efficient kernel implementation. Another disadvantage of split connections is that the end-to-end semantics of TCP acknowledgments is violated, since acknowledgments to packets can now reach the source even before the packets actually reach the mobile host. Also, since split-connection protocols maintain a significant amount of state at the base station per TCP connection, handoff procedures tend to be complicated and slow.

Experimental results: Like I-TCP, SPLIT scheme uses an intermediate host to divide a TCP connection into two separate TCP connections. The implementation avoids data copying in the intermediate host by passing the pointers to the same buffer between the two TCP connections. A variant of the SPLIT approach is, SPLIT-SMART, uses a SMARTbased selective acknowledgment scheme on the wireless connection to perform selective retransmissions. There is little chance of reordering of packets over the wireless connection since the intermediate host is only one hop away from the final destination. The main advantage of the split-connection approaches is that they isolate the TCP source from wireless losses. The TCP sender of the second, wireless connection performs all the retransmissions in response to wireless losses. Figure 6 and Table 2 show the throughput and goodput for the split connection approach in the LAN and WAN environments. Report for the results for two cases: When the wireless connection uses TCP Reno (labeled SPLIT) and when it uses the SMART-based selective acknowledgment scheme described earlier (labeled SPLIT-SMART). Figure 7 which show the progress of the data transfer and the size of the congestion window for the wired and wireless connections. We see that the wired connection neither has any retransmissions nor any timeouts, resulting in a wired goodput of 100%. However, it (eventually) stalls whenever the sender of the wireless connection experiences a timeout, since the amount of buffer space at the base station (64 KB in our experiments) is bounded3.

As expected, the throughput for the SPLIT-SMART scheme is much higher. It is about 1.3 Mbps in the LAN case and about 1.1 Mbps in the WAN case. The SMART-based selective acknowledgment scheme operating over the wireless link performs very well, especially since no reordering of packets occurs over this hop. However, there are a few times when both the original transmission and the first retransmission of a packet get lost, which sometimes results in a coarse timeout (as described earlier. This explains the difference in throughput between the SPLIT-SMART scheme and the LL-SMART-TCP-AWARE scheme (Fig. 3).

In summary, while the split-connection approach results in good throughput if the wireless connection uses special mechanisms, the performance is worse than that of a well tuned, TCP-aware link-layer protocol (LL-TCP-AWARE or LL-SMART-TCP-AWARE). Moreover, the link-layer protocol preserves the end-to-end semantics of TCP acknowledgments. This demonstrates that the end-to-end connection need not be split at the base station in order to achieve good performance.

Table 2: Results for the split-connection schemes at an average error rate of 1 every 64 KB

	Spllit	Split-smart
LAN (8 KB)	0.54 (97.4, 100%)	1.30 (97.6, 100%)
LAN (32 KB)	0.60 (97.3, 100%)	1.30 (97.2, 100%)
WAN (32 KB)	0.58 (97.2, 100%)	1.10 (97.6, 100%)

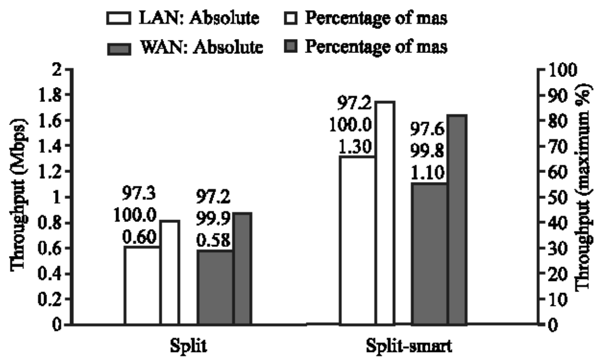

Fig. 6: Performance of split-connection protocol: Bit error rate = $1.9*10^{-6}$ (1 error/65536) procedures tend to be completed and slow

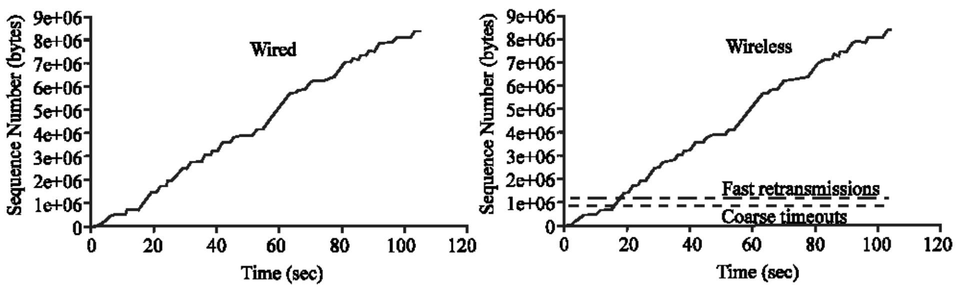

Fig. 7: Packet sequence trace for the wired and wireless parts of the SPLIT protocol. The wireless part has two rows of horizontal dots: the top one shows the times of fast retransmission and the bottom one the times of the timeout-based ones

CONCLUSIONS

- A reliable link-layer protocol that uses knowledge of TCP (LL-TCP-AWARE) to shield the sender from duplicate acknowledgments arising from wireless losses gives a 10-30% higher throughput than one (LL) that operates independently of TCP and does not attempt in-order delivery of packets. Also, the former avoids redundant retransmissions by both the sender and the base station, resulting in a higher goodput.

The TCP-aware link-layer protocol with selective acknowledgments performs the best.

- The split-connection approach, with standard TCP used for the wireless hop, shields the sender from wireless losses. However, the sender often stalls due to timeouts on the wireless connection, resulting in poor end-to-end throughput. Using a SMART-based selective acknowledgment mechanism for the wireless hop yields good throughput. However, the throughput is still slightly less than that for a well-tuned link-layer scheme that does not split the connection. This demonstrates that splitting the end-to-end connection is not a requirement for good performance.

ACKNOWLEDGMENTS

We are grateful to Principal Prof. M.U. Kharat and Prof. S.B. Wankhade for several comments and suggestion that helped to improve quality of this research.

REFERENCES

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

Chandran, K. *et al*., 2001. A feedback based scheme for improving of TCP performance in ad hoc wireless networks. IEEE Personal Commun. Magazine, 8: 34-39.

Dessimone, M., C. Chuah and O.C. Yue, 1993. Throughput performance of transport-layer protocols over wireless LANs. In: Proc. Globecom, 1: 542-549.

Kim, D., C-K. Toh and Y. Choi, 2001. TCP-BuS: Improving TCP Performance in wireless ad hoc networks. J. C. Networks, 3: 59-71.

Xu, S. and T. Saadawi, 2001. Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks. IEEE Communications, 39: 130-137.

Yavatkar, R. and N. Bhagwat, 1994. Improving end-to-end performance of tcp over mobile internetworks. In Mobile 94 Workshop on Mobile Computing Systems and Applications, pp: 146-152.