The Impact of End-to-end vs. Link-layer Mechanisms on Real-Time Performance over Wireless Links

Panagiotis Papadimitriou, Vassilis Tsaoussidis and Ageliki Tsioliaridou
Demokritos University, Electrical & Computer Engineering Department, Xanthi, Greece
E-mail: {ppapadim, vtsaousi, atsiolia}@ee.duth.gr

Abstract

We evaluate selected research proposals towards the efficient real-time QoS management over wireless links. We mainly focus on real-time performance of link- and transport-level mechanisms that bind operationally wired and wireless links. Employing a new metric for the evaluation of real-time performance we demonstrate that there are occasions where increased goodput does not correspond to real-time performance gains. In the sequel, we exploit further the potential of transport layer approaches.

1. Introduction

Towards a next-generation Internet, mobile computing gains popularity and falls under extensive research activity. Wireless links exhibit distinct characteristics, such as limited bandwidth, increased delays, diverse error-rates and potential handoff operations. Consequently, *Quality of Service* (*QoS*) requirements in wireless networking are stringent and complicated, taking additionally into account the influencing mobile device characteristics and limitations.

Transmission Control Protocol (*TCP*) is basically designed to provide a reliable service for wired Internet. The *Additive Increase Multiplicative Decrease* (*AIMD*) algorithm [6], incorporated in standard TCP versions, achieves stability and converges to fairness when the demand of competing flows exceeds the channel bandwidth. TCP is further enhanced with a series of mechanisms for congestion management, including *Congestion Avoidance* [10], *Slow Start*, *Fast Retransmit* and *Fast Recovery* [14].

Despite these features, TCP demonstrates inadequate performance in heterogeneous wired/wireless environments Authors in [15] outline three major shortfalls of TCP: (i) ineffective bandwidth utilization, (ii) unnecessary congestion-oriented responses to wireless link errors (e.g. fading channels) and operations (e.g. handoffs), and (iii) wasteful window adjustments over asymmetric, low-bandwidth reverse paths. More precisely, a suitable TCP for wired/wireless networks should be able to detect the nature of the errors that result in packet losses in order to determine the appropriate error-recovery strategy. Based on such an approach, the sender would not be obliged to reduce its transmission rate in the event of a wireless error or handoff. A next level of enhancement for TCP would enable a more sophisticated error-recovery strategy adjusted to the error characteristics of the underlying network, device constraints and performance tradeoffs.

The difficulty of the task that TCP has to perform is further enhanced, when the protocol provides services for real-time applications. Such applications are comparatively intolerant to delay and variations of throughput and delay. They are also affected by reliability factors, such as packet drops due to congestion or link errors. Hence, time-sensitive applications yield satisfactory performance only under certain QoS provisions, which may vary depending on the application task and the type of media involved. Evaluating the performance of real-time traffic based on traditional performance metrics (e.g. throughput) may produce misleading results, since such metrics do not account for variable delays that usually degrade real-time applications. Consequently, the inefficiency of TCP in conjunction with the lack of accurate performance metrics, render the applicability of TCP for real-time applications limited.

User Datagram Protocol (*UDP*) has been widely used instead of TCP in real-time applications. UDP lacks all basic mechanisms for error recovery and flow/congestion control. Thus, it allows for transmission attempts at application speed. That said, UDP can not guarantee reliability, and certainly is not able to deal with network delays either. In [12] we have shown that UDP may perform worse than TCP in several occasions. Along these lines, we do not include UDP in this study.

Although numerous research proposals have emerged towards improving transport services over wireless links, the converged domain of real-time traffic over wireless networks has not attracted the required attention from the research community so far. Several approaches operate on transport layer, most of them pronounced as enhanced TCP versions. In addition, a series of independent mechanisms have been proposed, which normally interact with TCP and provide reliable transmission over wireless links. Most of them operate on link-layer. Along these lines, the following question is raised: *"Where is the right place to add the integrative functionality of the wired and wireless Internet?"* Rendering this question as our primary motivation, we investigate a solution-framework based on the most prominent research proposals from the perspective of real-time application performance.

This work was funded by the European Commission and the project PENED 2003 of GSRT.

We organize the rest of the paper, as follows. In Section 2 we refer to selected approaches, which enhance TCP performance over wireless links. Furthermore, we provide an overview of research proposals towards the improvement of real-time performance over TCP. Section 3 includes our evaluation methodology, while in Section 4 we analyze the results of the experiments we performed. Finally, in the last section we highlight our conclusions.

2. Related Work

2.1. Improving TCP performance over wireless links

In the sequel, we summarize the most remarkable proposals which target at improving the performance of TCP over wireless links. Authors in [2] provide a comparative review of such approaches. Furthermore, open issues of TCP in mobile environments are extensively discussed in [15]. Remarkable mechanisms operating on link-layer include *Forward Error Correction* (*FEC*) and *Automatic Repeat Request* (*ARQ*) [7]. FEC introduces added overhead to data bits in order to cope with data corruption. However, the redundant information is not exploited in the absence of link errors resulting in a waste of bandwidth. On the other hand, ARQ mechanisms are invoked when packets containing bit errors can not be corrected. In this case, the erroneous packets are discarded and a retransmission is directly triggered. Unlike FEC, ARQ allocates additional network resources only when a packet is retransmitted. Furthermore, ARQ may interfere with TCP [3].

Snoop protocol [3, 2] provides a reliable solution by maintaining TCP end-to-end semantics while recovering the wireless errors locally. Snoop uses link level buffers at the base station to cache packets traversing the wireless link. It retransmits unacknowledged packets and consequently, unnecessary timeouts are avoided. Furthermore, Snoop suppresses duplicate acknowledgments for locally retransmitted packets in order to prevent TCP from performing fast retransmissions and backward window adjustments.

Additional proposals include split connection protocols. A split connection protocol virtually splits a TCP connection into two separate connections. The first one connects the sender with the base station, while the other connection is maintained between the base station and the receiver. A well-known representative of this family of protocols is *Indirect-TCP* (*I-TCP*) [1]. However, these protocols do not handle handoff operations efficiently [5], since such procedures tend to be slow and complicated. Furthermore, due to the split scheme, end-to-end semantics of TCP is violated [2].

2.2. Improving real-time performance over TCP

Congestion episodes often damage the timely delivery of packets and consequently, degrade real-time application performance. Hence, congestion avoidance mechanisms usually provide improved real-time performance. Congestion avoidance may be achieved through packet dropping (i.e. *RED*) or otherwise through bandwidth and delay estimation, which trigger transport-level adjustments prior to congestion. Alternatively, *ECN* is proposed in [13], where packets are marked rather than dropped when congestion is about to happen. A well-designed, congestion avoidance mechanism is *TCP Vegas* [4, 9]. Every *Round Trip Time* (*RTT*) the sender calculates the throughput rate which subsequently is compared to an expected rate. Depending on the outcome of this comparison the transmission rate of the sender is adjusted accordingly. Based on [9] admissions, Vegas achieves higher transmission rates than TCP Reno and TCP Tahoe.

Authors in [8, 17] proposed a family of TCP compatible protocols, called *TCP-Friendly*. TCP-Friendly protocols achieve smooth window adjustments, while they manage to compete fairly with TCP flows. *TCP-Friendly Rate Control* (*TFRC*) [8] is a representative TCP-Friendly protocol, where its transmission rate is adjusted in response to the level of congestion as it is indicated by the loss rate. TFRC eventually achieves the smoothing of the transmission gaps and therefore, is suitable for applications requiring a smooth sending rate, such as streaming media. However, this smoothness has a negative impact, as the protocol becomes less responsive to bandwidth availability [16]. *TCP Westwood* [11] is a TCP-Friendly protocol that emerged as a sender-side-only modification of TCP Reno congestion control. TCP Westwood exploits end-to-end bandwidth estimation to properly set the values of slow-start threshold and congestion window after a congestion episode. TCP-Real [18] is high-throughput transport protocol that incorporates congestion avoidance mechanism in order to minimize transmission-rate gaps. The protocol employs a receiver-oriented and measurement based congestion control mechanism that significantly improves TCP performance over heterogeneous networks and asymmetric paths.

3. Evaluation Methodology

3.1. Scenarios and parameters

The evaluation plan was implemented on the *NS-2* network simulator. In our experiments we used a wired-cum-wireless topology (Fig. 1), where two LANs are connected by a high bandwidth wireless link (5 Mbps). We simulated local retransmissions based on snooping at the wireless base station in order to study the interactions between TCP and the Snoop protocol. Error models were configured on both (forward and reverse) directions of the wireless bottleneck link with configurable packet error rates (*PER*). PER is adjusted at 0.01, unless otherwise explicitly stated. The number of source and sink nodes are always equal. In all experiments, we used droptail routers with buffer size adjusted in accordance with the *delay-bandwidth* product.

In order to simulate real-time traffic, we developed an *MPEG-4 Traffic Generator*. The traffic generated closely matches the statistical characteristics of an original video trace. We used three separate *Transform Expand Sample*

(*TES*) models for modeling I, P and B frames respectively. The resulting MPEG-4 stream is generated by interleaving data obtained by the three models. The MPEG traffic generator was integrated into NS-2 and provides the adjustment of the data rate of the MPEG stream, as well as useful statistical data (e.g. average bit-rate, bit-rate variance).

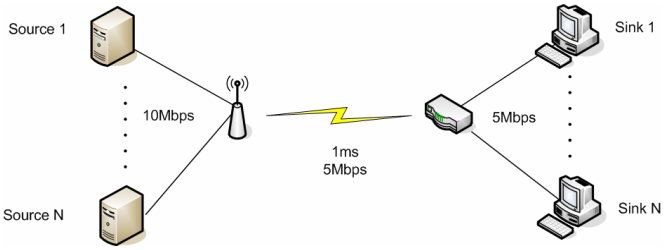

Figure 1. Simulation topology

Although we performed a series of experiments over various TCP protocols, we only comment on results from TCP variants Reno and Vegas, as well as TCP-Friendly protocols Westwood and TCP-Real, due to space limitations. We set the packet size to 1000 bytes and the maximum congestion window for all TCP connections to 64 KB. Each simulation lasts for 60 seconds, and diverse randomization seeds were used in order to reduce simulation dynamics.

3.2. Measuring performance

System goodput is used to measure the overall system efficiency in bandwidth utilization. Fairness is measured by the *Fairness Index*, derived from the formula given in [6], and defined as $(\sum_{i=1}^{n} \text{Throughput}_i)^2 / (n \sum_{i=1}^{n} \text{Throughput}_i^2)$, where $Throughput_i$ is the throughput of the i_{th} flow and n is the total number of flows.

In [12] we proposed a new metric for the performance evaluation of time-sensitive traffic, called *Real-Time Performance*. The metric monitors packet inter-arrival times and distinguishes the packets that can be effectively used by the client application from delayed packets (according to a configurable inter-arrival threshold). The proportion of the delayed packets is reflected in *Delayed Packets Rate*. Hence, Real-Time Performance index is defined as the ratio of the number of *"timely received packets"* over the total number of packets sent by the application:

$$\text{Real} - \text{Time Performance Index} = \frac{\#\text{timely received packets}}{\#\text{sent packets}} \leq 1$$

In our experiments, the inter-arrival threshold is adjusted at 100ms. Since real-time traffic is sensitive to packet losses, we additionally define *Packet Loss Rate*, as the ratio of the number of lost packets over the number of packets sent by the application. Most of our experiments were performed on several flows, so we present the average of the real-time performance of each MPEG flow.

4. Results and Discussion

In the sequel, we demonstrate and analyze the most prominent results from the experiments we performed based on three distinct scenarios. The basic parameters of each simulation scenario are as described in the previous section.

4.1. Interactions of Snoop with TCP

In the first scenario, we evaluate real-time application performance by investigating the interactions between TCP and the Snoop protocol. We hereby present some conclusive results over TCP Reno (Figs. 2-5) and TCP Vegas (Figs. 6-9). We performed our experiments for each TCP protocol with and without Snoop.

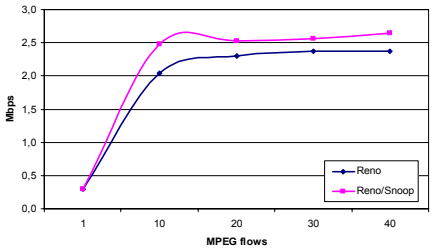

Figure 2. System goodput

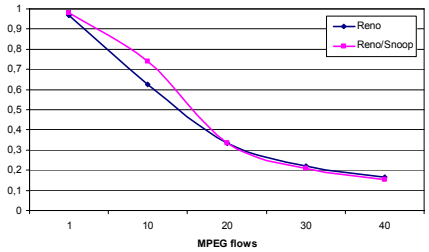

Figure 3. Average Real-Time Performance

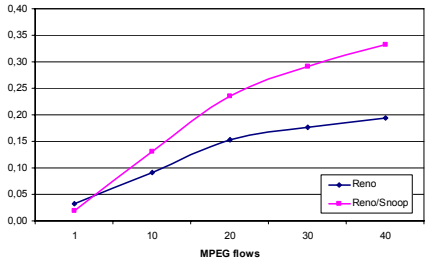

Figure 4. Delayed Packets Rate

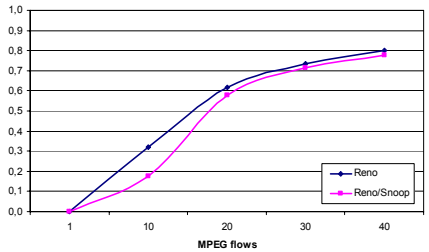

Figure 5. Packet Loss Rate

Fig. 2 illustrates that Snoop enables TCP Reno to achieve a more efficient performance. Snoop, running at the link layer, responds to packet losses faster than Reno. Each packet drop due to a wireless error is locally retransmitted (within TCP's timeout) and consequently, an end-to-end retransmission along with a wasteful backward window adjustment is prevented. This observation is depicted in Packet Loss Rate results (Fig. 5), where the combination of Reno and Snoop exhibits fewer packet drops than Reno alone.

However, from the perspective of real-time delivery, Snoop's supportive role is not profound, since minor performance gains are occasionally achieved (Fig. 3). On the contrary, real-application performance may be slightly degraded, primarily due to the increased number of delayed packets (Fig. 4). Consequently Snoop induces relatively long and variable delays which impact the timely delivery of packets at the receiver.

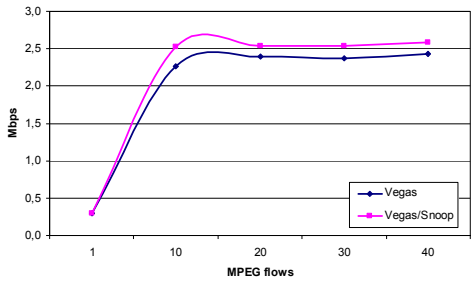

Figure 6. System goodput

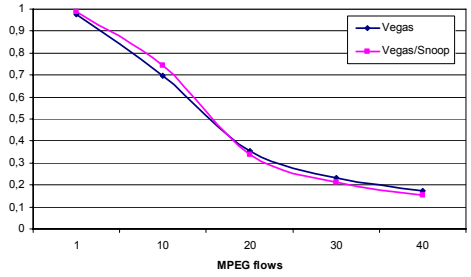

Figure 7. Average Real-Time Performance

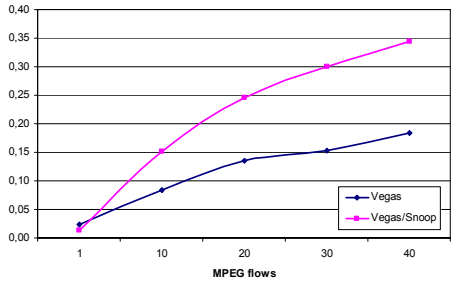

Figure 8. Delayed Packets Rate

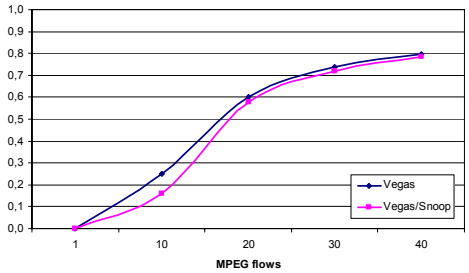

Figure 9. Packet Loss Rate

A comparative view between the interactions of Snoop with Reno (Figs. 2-5) and Vegas (Figs. 6-9) respectively, leads to the overall conclusion that the combination of Reno and Snoop is more effective. In the situation of Vegas, the gains in goodput are slighter (Fig. 6). Fig. 9 depicts that the advantage of Snoop (i.e. reducing end-to-end retransmissions) is diminished, as contention increases. Therefore, Snoop does not occasionally react to packet drops fast enough in order to prevent retransmissions from TCP Vegas. In terms of real-time delivery, Snoop degrades the performance of TCP Vegas (Fig. 7), since the proportion of delayed packets (Fig. 8) is remarkably increased.

Apart from Reno and Vegas, we investigated the interactions of Snoop with other TCP protocols and we evaluated the impact on real-time application performance. The overall conclusion of these efforts is that Reno interacts with Snoop more efficiently among all the TCP versions tested. Hence, the experimental study of Snoop is limited only under TCP Reno in the following scenarios.

4.2. Link-layer vs. end-to-end mechanisms

Departing from the analysis of Snoop's supportive role, we investigate the efficiency of selected transport layer mechanisms. In the sequel, we present conclusive results from TCP Reno interacting with Snoop, TCP Vegas, TCP Westwood and TCP-Real (Figs. 10-14).

Fig. 10 illustrates that the combination of TCP Reno and Snoop achieve the highest link utilization. The performance gains are also notable in the situation of increased contention (i.e. 20 and 40 flows), where packet drops are caused both by wireless errors and congestion. Therefore, Reno and Snoop exhibit most efficient responses to wireless errors, preventing TCP from unnecessary fast retransmissions and congestion control invocations. Both TCP Vegas and Real achieve a remarkable performance, although Vegas is not designed for wireless environments, since it is not able to distinguish the nature of error. Furthermore, Fig. 10 depicts the deficiency of TCP-Westwood, especially at increased contention. The protocol does not support error classification invoking congestion-oriented responses to wireless errors. In addition, the increased packet loss ratio of TCP-Westwood (Fig. 14) indicates that the protocol is unable to recover from heavy congestion, due to its smooth window adjustments.

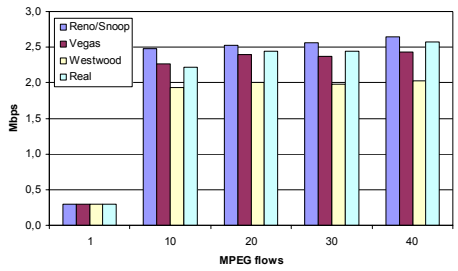

Figure 10. System goodput

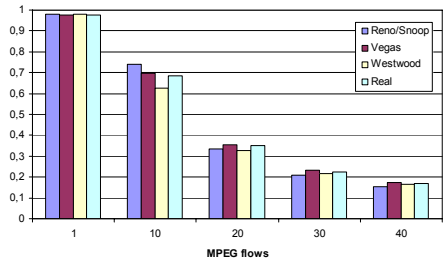

Figure 11. Average Real-Time Performance

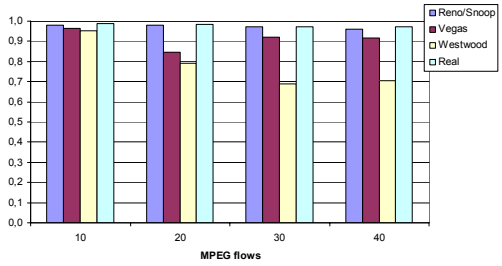

Figure 12. Fairness Index

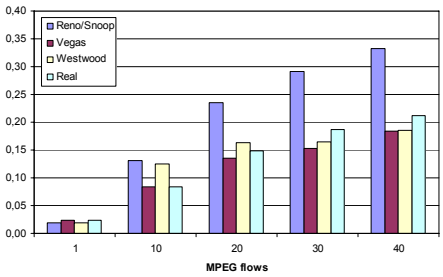

Figure 13. Delayed Packets Rate

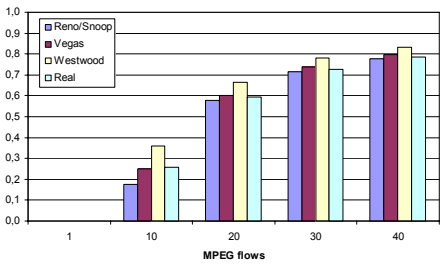

Figure 14. Packet Loss Rate

A comparative view in the results of Figs. 10 and 11 reveals that high goodput rates do not necessitate improved real-time application performance. Hence, the superior performance of Reno and Snoop in terms of goodput is not pronounced in the real-time performance results. Although Snoop deals effectively with link errors, it is responsible for excessive delays (Fig. 13), which degrade real-time performance. TCP Vegas is the only end-to-end solution that combines efficient link utilization with an acceptable amount of delayed packets, due to its sophisticated congestion avoidance mechanism. However, Vegas trades fairness for a remarkable performance; its congestion avoidance mechanism can not handle bandwidths sharing efficiently (Fig. 12). The inadequate link utilization of TCP-Westwood inevitably confines real-time performance, despite the timely delivery of packets. Finally, TCP-Real lies slightly behind TCP-Vegas in terms of real-time performance, due to the increased number of delayed packets.

4.3. Real-time performance vs. packet errors

In the last scenario, we performed the experiments using various packet error rates (PER: 0.01 - 0.05). We also carried out the same experiment without link errors and used it as a reference. Our objective is to demonstrate the impact of diverse packet error rates on goodput (Figs. 15, 17) and primarily on real-time delivery (Figs. 16, 18).

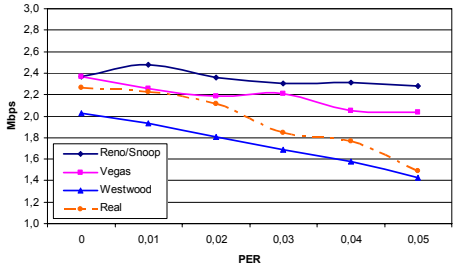

Figure 15. System goodput (10 Flows)

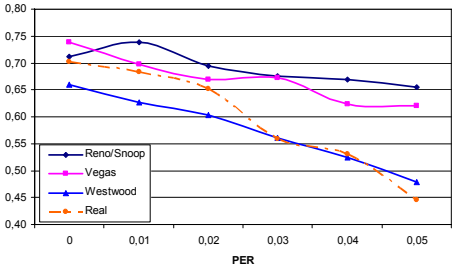

Figure 16. Average Real-Time Performance (10 Flows)

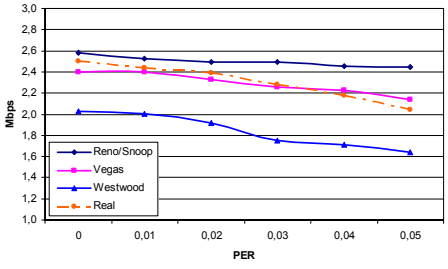

Figure 17. System goodput (20 Flows)

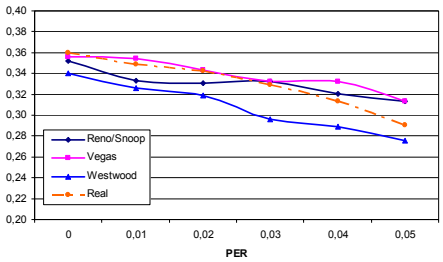

Figure 18. Average Real-Time Performance (20 Flows)

According to our expectations, the associated results illustrate TCP's performance degradation in the situation of increasing link errors. However, Reno in conjunction with Snoop is less responsive to the diverse packet error rates, due to the extended reliability provided by Snoop. The supportive role of Snoop is more effective, as link errors increase across the wireless channel. Similar to Reno/Snoop, Vegas exhibits minor implications in the event of increasing wireless errors. On the contrary TCP-Real and especially TCP-Westwood, demonstrate limited efficiency at relatively high packet error rates (PER: 0.04, 0.05). However, the performance of TCP-Real is notably improved when contention is increased (i.e. 20 flows).

5. Conclusions

We investigated selected approaches towards the efficient real-time QoS over heterogeneous networks, which integrate new functionality either on link-layer or end-to-end. Among the solutions that operate on link-layer, we evaluated the most prominent one: Snoop protocol. Based on our experimental results, we reached the outcome that Snoop interacts more effectively with TCP-Reno. Reno is based on a *"blind"* increase/decrease window mechanism that dynamically exploits bandwidth availability without relying on precise measurements of current conditions. However, Snoop's supportive role renders the specific TCP protocol a more efficient solution in the context of real-time performance over links prone to wireless errors.

The comparison of Reno/Snoop with selected end-to-end proposals reveals the disability of Snoop to achieve equivalent performance gains. The bandwidth estimation algorithm of TCP Vegas does not always obtain accurate estimates; yet it is more effective than Snoop interacting with TCP. However, the efficiency of Vegas comes at a cost: the protocol does not achieve a fair behavior. TCP-Real also delivers a remarkable performance, especially when contention is increased. Consequently, we reach the outcome that end-to-end solutions are of higher prospect. However, none of these proposed solutions is able to guarantee real-time QoS. A combined effort that guarantees both real-time performance and efficiency over wireless links has yet to be presented.

We also showed that the efficiency of most protocols and the associated real-time performance is drastically affected by awkward network conditions, such as increased link errors. Finally, we highlighted the importance of defining new metrics which explicitly evaluate real-time performance.

6. References

[1] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", In Proc. of *15th Int/nal Conference on Distributed Computing Systems (IDCS)*, May 1995

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *ACM/IEEE Transactions on Networking*, 5(6), pp. 756-769, 1997

[3] H. Balakrishnan, S. Seshan, and R. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks", *ACM Wireless Networks,* 1(4), December 1995

[4] L. Brakmo and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", *IEEE Journal on Selected Areas of Communications*, 13(8), October 1995

[5] R. Cáceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments", *IEEE Journal on Selected Areas in Communications*, 13(5), June 1995

[6] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks", *Journal of Computer Networks*, 17(1), June 1989

[7] A. Chockalingam, M. Zorzi, and V. Tralli, "Wireless TCP performance with link layer FEC/ARQ", In Proc. of *ICC '99*, Vancouver, Canada, June 1999

[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", In Proc. of *ACM SIGCOMM* , Stockholm, Sweden, Aug. 2000

[9] U. Hengartner, J. Bolliger, and T. Cross, "TCP Vegas Revisited", In Proc. of *IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000

[10] V. Jacobson, "Congestion avoidance and control", In Proc. of *ACM SIGCOMM '88*, Stanford, USA, August 1988

[11] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", In Proc. of *MobiCom '01*, Rome, Italy, July 2001

[12] P. Papadimitriou, V. Tsaoussidis and S. Tsekeridou, "The Impact of Network and Protocol Heterogeneity on Real-Time Application QoS", In Proc. of *10th IEEE ISCC*, Cartagena, Spain, June 2005

[13] K. Ramakrishnan, S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP", *RFC 2481*, Jan. 1999

[14] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", *RFC 2001*, Janary 1997

[15] V. Tsaoussidis and I. Matta, "Open issues on TCP for Mobile Computing", *Journal of Wireless Communications and Mobile Computing*, 2(2), pp. 3-20, February 2002

[16] V. Tsaoussidis and C. Zhang, "The dynamics of responsiveness and smoothness in heterogeneous networks", *IEEE Journal on Selected Areas in Communications (JSAC)*, 23(6), pp. 1178-1189, June 2005

[17] Y. R. Yang and S.S. Lam, "General AIMD Congestion Control", In Proc. of *8th Int/nal Conference on Network Protocols (ICNP)*, Osaka, Japan, November 2000

[18] C. Zhang and V. Tsaoussidis, "TCP Real: Improving Real-time Capabilities of TCP over Heterogeneous Networks", In Proc. of *11th ACM NOSSDAV*, New York, USA June 2001