The Impact of Network and Protocol Heterogeneity on Real-Time Application QoS

Panagiotis Papadimitriou, Vassilis Tsaoussidis and Sofia Tsekeridou
Demokritos University of Thrace, Electrical & Computer Engineering Department
E-mail: {ppapadim, vtsaousi, tsekerid}@ee.duth.gr

Abstract

We evaluate the impact of network, and protocol heterogeneity on real-time application performance. We focus on TCP and UDP supportive role, also in the context of network stability and fairness. We reach several conclusions on the specific impact of wireless links, MPEG traffic friendliness, and TCP version efficiency. Beyond that, we also reach an unexpected result: UDP traffic is occasionally worse than TCP traffic when the right performance metric is used.

1. Introduction

Quality of Service (QoS) is increasingly important for applications over the Internet. A main factor that complicates and even obstructs the efforts for efficient end-to-end QoS management is the heterogeneity of the Internet. However, identifying the presence of network heterogeneity is not enough. It is necessary to clarify and analyze all the heterogeneity parameters.

The application domain is generally classified into non-real time (e.g. HTTP, FTP) and real-time traffic (e.g. multimedia streaming). Real-time applications are comparatively intolerant to delay and to variations of throughput and delay [4]. They are also affected by reliability parameters, such as packet loss and bit errors. Therefore, a real-time application delivers satisfactory performance only under certain QoS provisions, which may vary depending on the application task and the type of media involved. Real-time applications compete with other network traffic, as they often share the same channel with corporate FTP and HTTP traffic. However, how real-time traffic affects or might be affected by other network traffic is still an open issue.

A real-time application has the option to run over *Transmission Control Protocol* (*TCP*) or *User Datagram Protocol* (*UDP*). TCP is the dominant protocol for data transmission over the Internet. Although TCP is based on the unreliable datagram service offered by IP, it manages to provide a reliable data delivery service to Internet applications. TCP uses a variety of techniques to achieve reliability. Generally, the protocol combines retransmission in conjunction with the sliding window mechanism. In standard TCP, sliding window adjustments are implemented according to the *Additive Increase Multiplicative Decrease* (*AIMD*) algorithm proposed by Chiu and Jain [3]. TCP is designed to allocate the resources of a network channel equally to each application using this channel. However, the demand of competing flows often exceeds the channel bandwidth leading to congestion. Therefore, efficient congestion control is of high importance in order to avoid undesirable implications for the network, such as congestive collapse. A series of mechanisms have been proposed for congestion control, including Congestion Avoidance [9], Slow Start, Fast Retransmit and Fast Recovery. Congestion control is usually triggered after a single packet loss. However, in heterogeneous wired/wireless environments, apart from congestion, hand-offs and fading channels may result in packet loss [14]. Generally, TCP is unable to successfully detect the nature of the errors in such a network environment.

Although the reliable service of TCP and its congestion control are suitable for traditional network traffic, real-time applications often struggle to operate efficiently. The sliding window adjustments of TCP do not provide the regular flow required by real-time applications when transmitting data. In wireless environments, the congestion-oriented responses to wireless link errors lead to wasteful window adjustments. The effect of these awkward conditions is long and varying delays, which damage the timely delivery of real-time data. Several TCP protocol extensions have emerged to overcome the standard TCP limitations providing more efficient bandwidth utilization and sophisticated mechanisms for congestion control [1, 11, 18].

Alternatively, most real-time implementations run over UDP. UDP is a fast, lightweight protocol without any transmission or retransmission control. UDP does not have functionality to override application characteristics, such as its transmission rates. It simply transmits at application rate and pattern. Consequently, UDP appears to be more suitable for real-time applications which tolerate some packet losses. However, the lack of a congestion control mechanism is a significant shortfall for UDP, especially as the Internetworking functionality evolves towards punishing free-transmitting protocols. Furthermore, the design principles of UDP do not anticipate fairness. Thus, any applications running over UDP are not fair. Along these lines, our research work is motivated by the following questions:

- How crucial is congestion control regarding real-time traffic? When the network load increases, does it contribute to efficiency as well?
- What is the efficiency of TCP and UDP with real-time traffic? When packet loss increases and UDP maintains its transmission rate, does it really maintain application efficiency as well?
- Are traditional metrics comparative enough to evaluate real-time application performance?

We organize the rest of the paper as follows: in the sequel, we provide an overview of recent research proposals which manage QoS focusing on network and protocol design. In Section 3 we present our evaluation methodology and we define a new performance metric for real-time applications. In Section 4 we analyze the results of our experiments and in the last section we highlight our conclusions.

2. Related Work

The impact of network heterogeneity on real-time application QoS has not been studied in depth. Relevant work includes [2], where the authors discuss the impact of mobility in QoS, and [4], where the QoS of real-time traffic along with its characteristics are analyzed. Authors in [5] discuss how streaming traffic competes with other TCP traffic over low bandwidth WAN links. Furthermore, there are remarkable research efforts towards the efficient QoS management of real-time applications focusing on protocol design. We discuss them in the rest of this section.

2.1 TCP-friendly Protocols

The disqualification of standard TCP to meet the requirements of real-time applications was the motive for a new family of protocols. Authors in [6, 7, 16, 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. In order to achieve smoothness, this family of protocols use a gentle backward adjustment upon congestion. However, this modification has a negative impact on the protocol responsiveness.

TCP-Friendly Rate Control (*TFRC*) is a representative TCP-friendly, rate-based congestion control protocol. According to *TFRC*, the transmission rate is adjusted in response to the level of congestion as it is indicated by the loss rate [10]. Multiple packet losses in the same *RTT* are considered as a single loss event by *TFRC* and hence, the protocol follows a more gentle congestion control strategy. *TFRC* eventually achieves the smoothing of the transmission gaps and therefore, is suitable for applications requiring a smooth sending rate. However, this smoothness has a negative impact, as the protocol becomes less responsive to bandwidth availability [19].

TCP-Real is a high-throughput transport protocol that incorporates congestion avoidance mechanism in order to minimize transmission-rate gaps. Therefore, this protocol is suited for real-time applications, as it enables better performance and reasonable playback timers. *TCP-Real* [18, 15] employs a receiver-oriented and measurement based congestion control mechanism that significantly improves TCP performance over heterogeneous networks and over asymmetric paths. In *TCP-Real*, the receiver decides with better accuracy about the appropriate size of the congestion window. Slow Start and timeout adjustments are present, but they are only used whenever congestion avoidance fails.

TCP Westwood is a sender-side-only modification of *TCP Reno* congestion control, which exploits end-to-end bandwidth estimation to properly set the values of slow-start threshold and congestion window after a congestion episode. *TCP Westwood* significantly improves fair sharing of high-speed networks capacity. The protocol performs an end-to-end estimate of the bandwidth available along a TCP connection to adaptively set the control windows after congestion [11]. Although *TCP Westwood* does not incorporate any mechanisms to support error classification and the corresponding recovery tactics for wired/wireless networks, the proposed mechanism appears to be effective over asymmetric paths due to its efficient congestion control.

2.2 Congestion Avoidance

A congestion episode usually has a negative impact on the performance of a real-time application, regardless of the effectiveness of the TCP congestion control mechanisms. Based on this observation, an approach dealing with congestion from another perspective has been proposed. The goal of this approach, called congestion avoidance, is to estimate the level of congestion before it takes place, and hence avoid it. 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.

A well-designed, congestion avoidance mechanism is *TCP Vegas*. Every *RTT (Round Trip Time)* the sender calculates the throughput rate which subsequently is compared to an expected rate [14]. Depending on the outcome of this comparison the transmission rate of the sender is adjusted accordingly. Based on [1] admissions, *Vegas* achieves better transmission rates than *TCP Reno* and *TCP Tahoe*. Although the protocol is compliant to the rules of fairness (*AIMD* algorithm), according to [8], *Vegas* can not guarantee fairness.

Reviewing all the proposed solutions, TCP-friendly protocols and the various approaches dealing with congestion extend further the heterogeneity of the network.

3. Experimental Methodology

3.1 Experimental Settings

The evaluation plan was implemented on the NS-2 network simulator. In our experiments we used a high-speed single-bottleneck topology, known as *dumbbell* (Fig. 1). The capacity of the bottleneck is configured to 100Mbps, so that each flow has enough fair-share to expand its window. The buffer size of the router that forwards the packets to the receivers was adjusted in accordance with the *delay x bandwidth* product. The number of source and sink nodes are equal in all experiments.

Apart from a wired network, we also simulated an heterogeneous wired/wireless environment by inserting *NS-2* error models into the access links to the sink nodes. The error models were configured on both (forward and reverse) directions of the link traffic. The Bernoulli model was used in order to simulate link-level errors with packet error rate (*PER*) adjusted at 0.01. Furthermore, we included mobility in our wireless scenario in order to monitor the behavior of the network and its impact on the application in a situation of frequent handoffs. Thus, the protocols experience a situation where 5 handoff events occur and each one lasts 1 sec.

In our experiments, we used the MPEG-4 traffic generator proposed in [12] in order to simulate real-time traffic. The traffic generated closely matches the statistical characteristics of an original video trace. The model developed is based on Transform Expand Sample (TES) [13]. We used three separate TES models for modeling I, P and B frames respectively. The resulting MPEG-4 stream was 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 (i.e. average bit-rate, bit-rate variance).

For each scenario we used different number of flows in order to have more productive results. For all the experiments, the simulation time was fixed at 60 seconds, an appropriate time-period for all the protocols to demonstrate their potential.

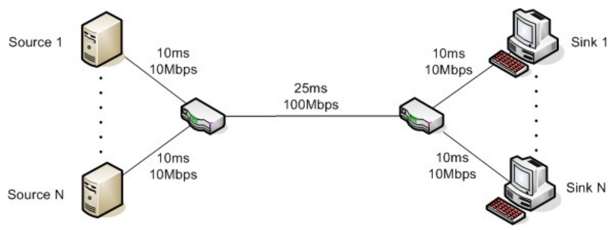

Figure 1. Simulation topology

In order to evaluate TCP performance, we used two representative TCP protocols as reference: Reno and Vegas. However, in the last scenario, we included the TCP-friendly protocols TCP-Westwood and TCP-Westwood+ in order to evaluate their efficiency in comparison with the rest of TCP versions.

3.2 Performance Metrics

System goodput was used to measure the overall system efficiency in bandwidth utilization. In order to evaluate the efficiency of selected flows, such as an MPEG flow competing with other FTP flows, we additionally measured the goodput of each flow separately. Fairness is measured by the *Fairness Index*, derived from the formula given in [3], and defined as:

$$\text{Fairness Index} = \frac{(\sum_{i=0}^{n} \text{Throughput}_i)^2}{n(\sum_{i=0}^{n} \text{Throughput}_i^2)}$$

where *Throughput*$_i$ is the throughput of the ith flow and n is the total number of flows.

In [15] Tsaoussidis and Zhang proposed a new metric for the performance evaluation of real-time traffic, called "*x% Application Success Percentage*". Virtually, the metric captures the number of discrete time slots when the flow achieves at least x% of the *Targeted Receiving Rate*. The calculation of *Application Success Percentage* is based on the *AllottedGoodput(i, j)* which is the goodput of the ith flow within the jth second. Although this metric is more appropriate for the performance evaluation of real-time applications than goodput, it has some limitations. Its major shortfall is that it does not take into account packet interarrival time, which is a critical parameter for time-sensitive applications. Therefore, the performance of such applications can not be effectively evaluated.

Targeting to a more efficient criterion for the justification of real-time traffic performance, we introduce a new metric, called *Real-Time Performance*. The basic observation that goodput can not be directly used for the performance evaluation of a real-time application called for the definition of such a performance metric. The key to evaluate the performance of a real-time application is to measure the packets that arrive at the receiver(s) on time, depending on the application requirements. Thus, *Real-Time Performance* index is the ratio of the number of timely received packets over the total number of packets sent by the application:

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

In order to measure the number of received packets that can be effectively used by the real-time application, an extension has been made to the functionality of the receiver. Each recipient, receiving packets from the MPEG streaming application, calculates the number of *timely received packets* based on the two versions of the algorithm. The first version of the algorithm (*Algorithm 1*) captures the number of *timely received packets* of the MPEG application running over TCP, while the alternative version (*Algorithm 2*) is used for UDP. The two versions of the algorithm have the same functionality, excluding the fact that the TCP version takes retransmission into account, whenever it takes place.

Algorithm 1. Timely Received Packets (TCP)

```
 1: if threshold > 0 then
 2:    set packetTime = currentTime
 3:    increase packets_received
 4:    if seqNo - last_seqNo = 1 then
 5:       if packetTime - lastPacketTime > threshold then
 6:          increase uselessPackets
 7:       end if
 8:    else
 9:       increase uselessPackets
10:    end if
11:    if currentPacket not retransmitted
12:       set last_seqNo = seqNo
13:       set last_packetTime = packetTime
14:    end if
15:    set usablePackets = packetsReceived - uselessPackets
16: end if
```

Algorithm 2. Timely Received Packets (UDP)

```
 1: if threshold > 0 then
 2:    set packetTime = currentTime
 3:    increase packets_received
 4:    if seqNo - last_seqNo = 1 then
 5:       if packetTime - lastPacketTime > threshold then
 6:          increase uselessPackets
 7:       end if
 8:    else
 9:       increase uselessPackets
10:    end if
11:    set last_seqNo = seqNo
12:    set last_packetTime = packetTime
13:    set usablePackets = packetsReceived - uselessPackets
14: end if
```

Several notations used in the pseudocode algorithms are as follows:

1. *threshold* : inter-arrival threshold
2. *packetTime* : arrival time of the current packet
3. *seqNo* : sequence number of current packet
4. *last_seqNo* : sequence number of last packet
5. *lastPacketTime* : arrival time of the last packet
6. *uselessPackets* : number of useless packets
7. *usablePackets* : number of packets that can be effectively used by the application

Both versions of the algorithm basically use two criteria to identify whether each received packet can be used by the client application. The first criterion is packet inter-arrival time. We set here an inter-arrival time threshold at 100ms. Certainly, this can be adjusted depending on the application, user requirements and network delay. Each packet with interarrival time greater than this threshold is considered as *useless*. Practically such packets are either discarded and considered lost, or at the worst they obstruct the proper reconstruction of oncoming packets. Secondly, the algorithm takes into account the sequence number of packets. An arriving packet which is out of transmission order is also considered as *useless*. In the context of TCP, such a packet may be retransmitted. Since a retransmitted packet is *useless*, it is not used as reference (i.e. its sequence number and arrival time) for the next packet (Algorithm 1: lines 11-14).

4. Results and Discussion

In order to evaluate the impact of the protocols and the network characteristics on the QoS of a real-time application, we conducted several experiments based on two separate scenarios. The basic parameters of each scenario are as described in the previous section. We carried out our experiments simulating both a wired and wireless topology.

4.1 The Impact of MPEG on Corporate Traffic

In the first scenario, we simulated (i) 1 FTP flow (over TCP), (ii) 1 MPEG flow over TCP, and (iii) 1 MPEG flow over UDP, each one of them competing with a number of other FTP flows (1, 10, 20, 40 and 80 flows). We measured the aggregated goodput of all flows in the system except the first FTP/MPEG TCP/MPEG UDP flow, correspondingly. The data rate of the MPEG flow was adjusted, so that it can compete fairly with the FTP flow.

Our purpose here is to demonstrate the impact of the MPEG application on the other applications when they all share the same channel. We conducted the experiments simulating both a wired (Figs. 2, 4) and wireless topology (Figs. 3, 5). We used the TCP protocols Reno (Figs. 2, 3) and Vegas (Figs. 4, 5) for the FTP flows and the MPEG TCP flow.

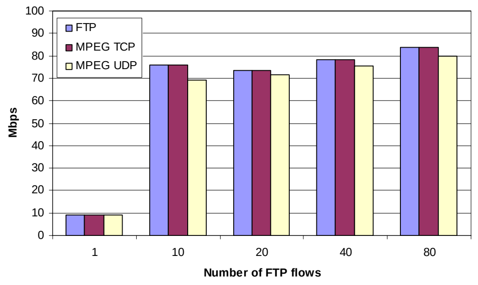

Figure 2. Goodput of remaining FTP flows (Reno/Wired)

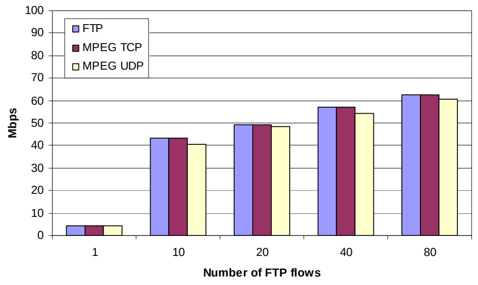

Figure 3. Goodput of remaining FTP flows (Reno/Wireless)

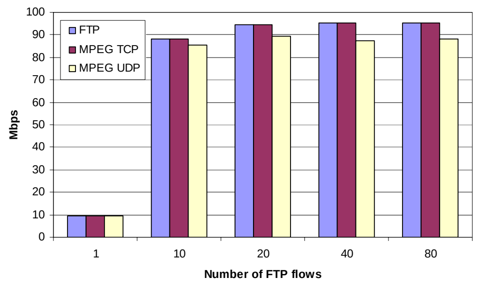

Figure 4. Goodput of remaining FTP flows (Vegas/Wired)

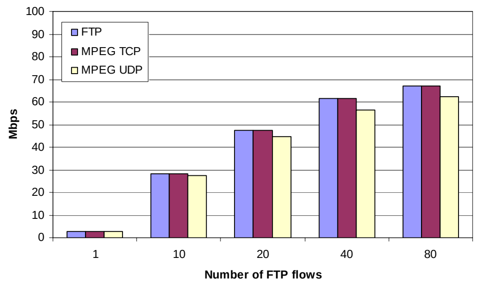

Figure 5. Goodput of remaining FTP flows (Vegas/Wireless)

The results clearly demonstrate the impact of the MPEG application on TCP network traffic. Using the results of the 1st FTP flow as reference, we reach the conclusion that the MPEG flow over TCP does not affect the efficiency of the remaining FTP flows. However, we observe that the MPEG flow over UDP has a negative impact on the other flows in each case, as their aggregated goodput is decreased. This observation is in accordance with our expectations, considering the lack of any back-up policies of the UDP protocol. Indeed, UDP is not fair to the other applications. Therefore, any application sharing the same channel with UDP flows is in an unfavorable situation. Furthermore, the limited efficiency of the FTP flows is obvious in the context of wireless networking. Finally, TCP Vegas, due to its sophisticated congestion avoidance mechanism, is more efficient, since it achieves higher transmission rates than TCP Reno.

4.2 TCP vs. UDP

The final scenario includes the simulation of (i) N MPEG flows over TCP, and (ii) N MPEG flows over UDP, where N is set to 10, 20, 40 and 80 successively. In these experiments, apart from the two referenced TCP versions (i.e. Reno, and Vegas), we additionally used TCP Westwood and TCP Westwood+. Our purpose is to evaluate the impact of each protocol on the real-time application QoS, when there are only MPEG TCP or MPEG UDP flows in the system. Our metrics include system goodput, the average of the Real-Time Performance of each MPEG flow and Fairness Index. In this scenario many heterogeneity parameters are under evaluation, adding the wired and wireless topology.

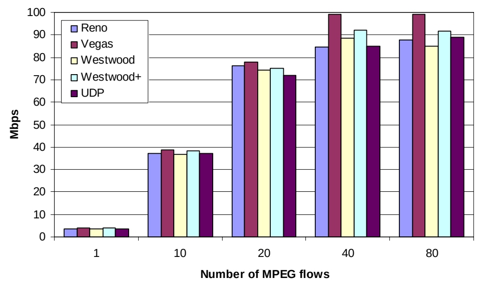

Figure 6. System Goodput (Wired)

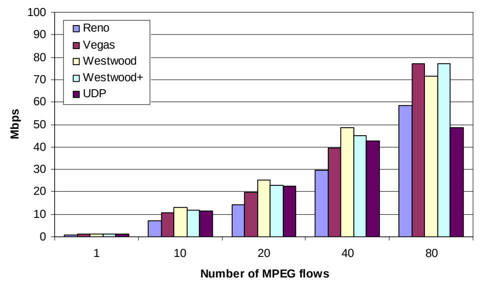

Figure 7. System Goodput (Wireless)

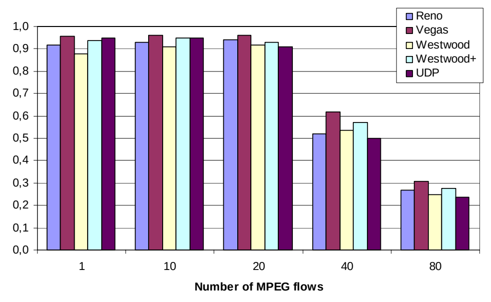

Figure 8. Average Real-Time Performance (Wired)

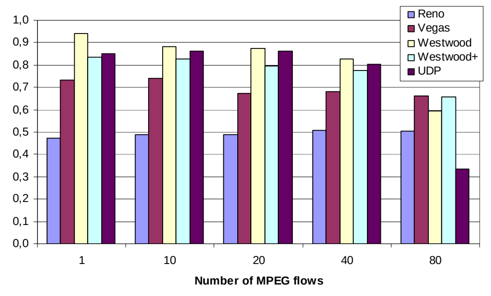

Figure 9. Average Real-Time Performance (Wireless)

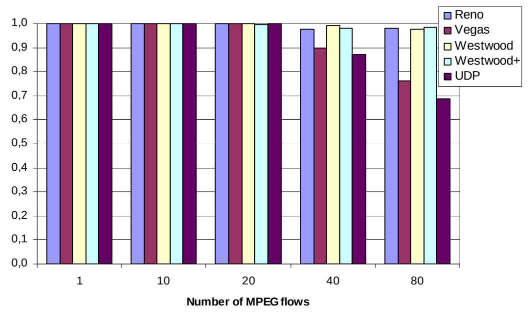

Figure 10. Fairness Index (Wired)

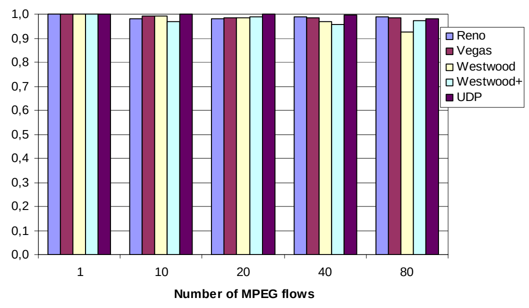

Figure 11. Fairness Index (Wireless)

Taking into account the goodput results (Figs. 6, 7), we observe that all TCP protocols achieve higher goodput than UDP. This difference increases alongside with the number of flows. Either over TCP or UDP congestion episodes occur. Since TCP protocols incorporate congestion control or avoidance mechanisms, they deal with congestion more efficiently. UDP, with the absence of congestion control, is unable to control the congestion. The MPEG streaming application transmits packets over UDP ignoring the network condition, resulting in a large number of lost packets (especially at 40 and 80 flows). This phenomenon is reflected clearly in the Real-Time Performance results (Figs. 8, 9), where UDP performs inadequately. Furthermore, UDP does not compete fairly with the rest of UDP flows (Fig. 10), since UDP is not designed to anticipate fairness.

A very interesting outcome of these results is the performance of TCP-friendly protocols Westwood and Westwood+. This family of protocols use smooth window adjustments which appear to be in accordance with real-time application requirements. However, during excessive congestion (i.e. 40 and 80 flows), the conservative policy of TCP-friendly protocols results in increased packet losses. Therefore, TCP-friendly protocols fail to accomplish their design goals, since they do not achieve any performance gains (i.e. Real-Time Performance: 40 and 80 flows).

5. Conclusion

Reviewing our discussion and based on the above experimental results we reach the following conclusions: (i) Congestion control is mandatory and protocols which do not incorporate such mechanisms (i.e. UDP) have limited efficiency and potential. (ii) TCP has inadequate efficiency in wireless environments, due to its congestion-oriented responses to wireless errors and operations (e.g. handoffs). (iii) UDP is not fair to other flows sharing the same channel, no matter if these flows are over TCP or UDP. (iv) TCP-friendly protocols do not provide the improved real-time performance their designers promised, due to their smooth window adjustments at periods of congestion. (v) TCP Vegas achieves higher transmission rates than standard TCP, especially when congestion occurs. (vi) Goodput is not an accurate criterion for the evaluation of real-time application performance. Since current protocol support for real-time applications does not meet their stringent QoS requirements, future work will be focused on the design of a protocol which will provide improved support for time-sensitive applications. More precisely, departing from current TCP-Real, we intend to design and propose a partially reliable protocol.

References

[1] L. Brakmo and L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet”, *IEEE Journal on Selected Areas of Communications*, October 1995

[2] D. Chalmers, M. Sloman, “A Survey of Quality of Service in Mobile Computing Environments”, *IEEE Communication Surveys*, 1999

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

[4] D. D. Clark, S. Shenker, L. Zhang, “Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism”, *In Proc. of SIGCOMM '92*, pp. 14-26, August 1992

[5] R. Doshi and P. Cao, “Streaming Traffic Fairness over Low Bandwidth WAN Links”, *In Proc. of 3rd IEEE Int/nal Workshop on Internet Applications*, San Jose, June 2003

[6] S. Floyd, M. Handley and J. Padhye, “A Comparison of Equation-based and AIMD Congestion Control”, May 2000

[7] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-Based Congestion Control for Unicast Applications”, *In Proc. of ACM SIGCOMM 2000*, August 2000

[8] U. Hengartner, J. Bolliger, and T. Cross, “TCP Vegas Revisited”, *In Proc. of IEEE INFOCOM 2000*, March 2000

[9] V. Jacobson, “Congestion avoidance and control”, *In Proc. of ACM SIGCOMM ‘88*, August 1988

[10] L. Mamatas and V. Tsaoussidis, “Protocol Behavior: More Effort, More Gains?”, *In Proc. of 15th IEEE PIMRC*, Barcelona, September 2004

[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*, July 2001

[12] A. Matrawy, I. Lambadaris and C. Huang, “MPEG4 Traffic Modeling using the Transform Expand Sample Methodology”, *In Proc. of 4th IEEE Int/nal Workshop on Network Applicances*, Gaithersburg, 2002

[13] B. Melamed, “An Overview of TES Processed and Modeling Methodology”, *Performance Evaluation of Computer and Communication Systems*, 1993

[14] V. Tsaoussidis and I. Matta, “Open issues on TCP for Mobile Computing”, *Journal of Wireless Communications and Mobile Computing*, 2(2), Feb. 2002

[15] V. Tsaoussidis and C. Zhang, “TCP Real: Receiver-oriented congestion control”, *Computer Networks*, 40(4), Nov. 2002

[16] Y.R. Yang, M.S. Kim and S.S. Lam, “Transient Behaviors of TCP-friendly Congestion Control Protocols”, *In Proc. of IEEE INFOCOM 2001*, April 2001

[17] Y.R. Yang and S.S. Lam, “General AIMD Congestion Control”, *In Proc. of 8th 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 IEEE/ACM NOSSDAV*, June 2001

[19] C. Zhang and V. Tsaoussidis, “The interrelation of TCP Responsiveness and Smoothness”, *In Proc. of 7th IEEE ISCC*, July 2002