Ensuring Fair Coexistence of Multimedia Applications in a Wireless Home

Claudio E. Palazzi, Nicola Stievano
Dipartimento di Matematica Pura ed Applicata
Università degli Studi di Padova
Via Trieste 63, 35121 Padova, Italy
Email: cpalazzi@math.unipd.it,
nstievan@studenti.math.unipd.it

Marco Roccetti, Gustavo Marfia
Dipartimento di Scienze dell'Informazione
Università di Bologna
Mura Anteo Zamboni 7, 40127 Bologna, Italy
Email: roccetti@cs.unibo.it,
marfia@cs.unibo.it

***Abstract*—Nowadays, wireless connectivity is increasingly available in homes to support heterogeneous multimedia applications. However, with current off-the-shelf systems, real-time applications (e.g., video streaming, online games) suffer from delays caused by the interference with elastic (e.g., TCP-based downloading sessions) ones. Furthermore, simultaneous elastic application flows are not able to fairly share the same bottleneck, severely damaging those featured with long round-trip times (RTTs). In this article, we discuss how a solution based on a smart access point can actually solve these problems, ensuring low per-packet delays to real-time applications and both high and RTT-fair throughput to elastic ones. We discuss the experimental outcome of our solution showing the benefits achieved with respect to using the legacy protocols and even to adopting a TCP Vegas-based solution, known for providing the same goals we are aiming at.**

***Index Terms*—Computer-Centered Home Entertainment, Smart Access Point, Wireless Multimedia, Wireless Home.**

I. Introduction

Nowadays, more and more home devices such as personal computers, consoles, TV sets, and other appliances, may make be connected with each other or to the Internet through wireless connectivity. Wireless DSL connectivity may be provided to the various home devices by a gateway and an access point (AP). File downloading or sharing, Internet browsing, video or audio streaming, VoIP, and online gaming are just a few, even if very representative, examples of connectivity-based multimedia applications employed by users in a wireless home [1]–[7] Yet, the current TCP/IP suite of Internet protocols has been developed tens of years ago and presents several issues when plunged in today's wireless, heterogeneous scenarios (e.g., efficiency, fairness, ability to support interactive applications).

Indeed, focusing on a wireless home scenario new problems emerge, requiring the employment of specific solutions. For instance, in this work, we consider two of these problems having a significant impact of performance as perceived by final users and discuss a solution we designed to solve them.

The first problem is related to the tendency of TCP's probing mechanism in utilizing more and more bandwidth. This behavior, especially when coupled with the presence of large buffers at the connection bottleneck, can negatively affect real-time applications as it creates queues, increasing the per-packet delivery latency [8]. This result, even if predictable, is quite surprising as TCP is generally considered more friendly than UDP since only the former has a congestion control functionality. Instead, if one user is engaged with an interactive online game while another user, sharing the same bottleneck, is downloading a big file, the former will frequently notice a lack of responsiveness in receiving game updates and in executing game actions; the game will be less fun and the user will probably not be able to compete in a fair way.

The second problem we consider is the well known RTT-unfairness suffered by TCP. In essence, competing TCP flows sharing the same bottleneck but featured with different round trip times (RTTs) surprisingly achieve different throughputs, with an inverse proportion to the respective RTTs. As an example, think to two devices inside a house, engaged in TCP sessions to download some big multimedia files through the same bottleneck. One would expect that they achieved a fair sharing of the available bandwidth if they share the same bottleneck; instead, the small RTT session unfairly captures most of the bandwidth with respect to the longer RTT download [9], [10].

Summarizing, if we want to provide an efficient home network, able to support simultaneous and heterogeneous applications, we need to achieve three goals: i) low per-packet delays, ii) high downloading throughput, and iii) fair utilization of the shared bottleneck. In this paper, we discuss a solution we developed to achieve these three goals. Our solution is named *Smart Access Point with Low Advertised Window (SAP-LAW)* and makes use of a smart AP, standard protocol features, and available information on the on-going traffic. As SAP-LAW is significantly based on a TCP's feature, we compare its experimental outcome with results achieved when using the legacy TCP SACK protocol and when adopting a state-of-the-art solution based on TCP Vegas, which is actually known for providing the same goals we are aiming at [11], [12].

The rest of the paper is organized as follows. Section II discusses our SAP-LAW solution. The experimental testbed is presented in Section III, whereas obtained results are shown in Section IV. Finally, Section V concludes this paper.

978-1-4244-5661-1/09/$26.00 ©2009 IEEE

II. A SMART SOLUTION: SAP-LAW

One of the main features of TCP is represented by its congestion control functionality by which the sending rate of a transmission session continuously grows, queuing packets on the buffer associated with the bottleneck of a connection, until some packet gets lost; at that point the sending rate is halved before starting again its growing cycle. If one considers that the same wireless connection might be shared by several devices and applications, it becomes evident how this aggressive behavior of TCP-based connections (i.e., elastic downloading applications) causes packet buffering that delays all packets passing through that bottleneck, even those belonging to real-time applications [8].

This problem can be solved through the use of a smart AP that monitors all the on-going traffic and forces TCP flows to not exceed a certain amount of bandwidth consumption. This can be done through a regular feature of standard TCP protocol: the *advertised window*. In essence, an appropriate advertised window is computed for the TCP flows so as to maximize the throughput of elastic applications but, at the same time, to not generate queues at the bottleneck that would jeopardize the requirements for low per-packet delays of real-time applications. This computed value for the advertised window is then included in all the TCP's acknowledgments (ACKs) traveling through the AP by the AP itself. The formula utilized to compute the appropriate value for the advertised window of TCP-based elastic flows can simply be as follows:

$$maxTCPrate(t) = \frac{(C - UDPtraffic(t))}{\#TCPflows(t)} \qquad (1)$$

where $UDPtraffic$ is the amount of bandwidth occupied by the UDP-based traffic at time t, $\#TCPflows$ is the concurrent number of TCP flows at time t, and C is the capacity of the bottleneck link.

However, as it is evident, (1) does not solve completely the well known RTT bias that affects TCP-based flows. In essence, since the dependence on the RTT in determining the ratio of the returning ACKs and hence the increase of the sending rate, TCP-based flows with small RTT unfairly capture the available bandwidth with respect to other flows [9]. When leveling the sending windows through (1), the throughput of each TCP-based flow becomes equal to the ratio between the constant sending window (at steady state equal to the computed advertised window) and the flow's RTT, thus maintaining some RTT unfairness among the various flows. The outcome is better than with regular TCP where, in addition, the sending window of small RTT flows always becomes bigger than the sending window of bigger RTT flows; yet, the RTT unfairness problem is not completely solved.

As an example, Fig. 1 shows the average throughput of each of three simultaneous downloads of a multimedia file from three different servers. The three servers are located at 30, 50, and 100 ms, respectively, of RTT from three downloading clients inside the house. This simple but revealing example considers a regular AP implementing IEEE 802.11g MAC

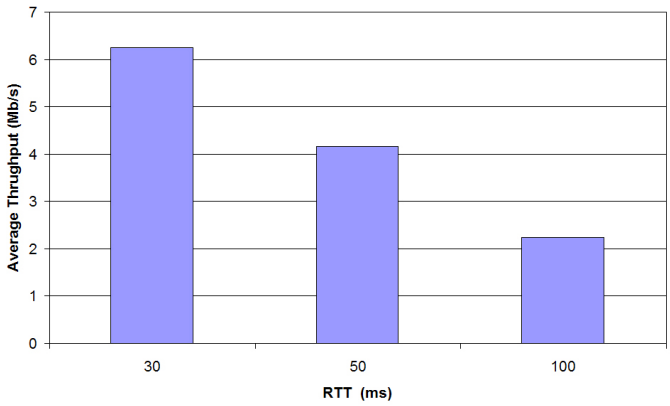

Figure 1. Average thrughput for three simultaneous flows with different RTTs (30, 50 and 100 ms, respectively); the inverse proportion between the RTT and the throughput is evident.

layer and the last mile wireless link between the AP and the clients within the house represent the shared bottleneck of the connections (*circa* 19 Mb/s of bandwidth). The chart clearly shows the unfair divergence among the data rates.

From these considerations arises the need to modify (1) in order to have different maximum transmission rates, through different advertised windows, for flows with different RTTs. The formula for a flow i can be hence written as follows:

$$maxTCPrate_i(t) = \frac{(C - UDPtraffic(t)) * RTT_i}{\sum \frac{RTT_i}{avg\_RTT_{min}}} \qquad (2)$$

where RTT_i are the smallest RTTs for the TCP-based flows sharing the AP, and avg_RTT_{min} is the average among these smallest RTTs.

The rationale behind (2) is the fact the throughput at steady state can be computed as the ratio between the maximum sending rate and the RTT for that connection. Therefore, if two connections have different RTTs but we want equal throughput, then their maximum sending rates should be simply chosen with an inverse proportion.

III. SIMULATION ASSESSMENT

Our intention is to compare the efficacy of our SAP-LAW with respect to other technical solutions. In particular, as our solution is significantly based on a TCP's feature, i.e., the advertised window, we compare its experimental outcome with those achieved when using a legacy TCP protocol, i.e., TCP SACK, and when adopting a TCP-based solution known for providing the same goals we are aiming at, i.e., TCP Vegas.

The main metrics used for our comparison are the three goals listed in Section I:

- low per-packet delays;
- high downloading throughput;
- fair utilization of the shared bottleneck.

To this aim, we have built an experimental scenario utilizing the well-known NS-2 simulator with modifications to implement our SAP-LAW solution [13].

We intend to analyze a general house environment with four devices connected to the Internet through the wireless connectivity provided by an AP. The main settings and parameters of the experimental scenario are as follows.

The distance between each device and the AP is 10 m and the MAC layer parameters have been set accordingly to the IEEE 802.11g standard allowing us to reach a maximum wireless bandwidth of *circa* 19 Mb/s; this represents a reasonable maximum value for TCP-based flows over the declared 54 Mb/s, even in the real world [14].

For the wireless medium in our simulations we have utilized the *Shadowing Model* present in the NS-2 simulator which realistically simulates signal fading. We followed the directions provided by the official NS-2 manual to represent a home environment partitioned into several rooms. Specifically, the *path loss exponent* and the *shadowing deviation* parameters have been set equal to the worst possible case suggested for an indoor environment: 4 and 9, respectively.

As for the simulated protocols, a new module was added to simulate SAP-LAW implementing (2), whereas standard modules already present in NS-2 were utilized with default parameter values to simulate TCP SACK and TCP Vegas.

In our experiments we consider three TCP flows generated by three FTP applications running for the whole 180 s of duration of the simulations, whereas a UDP flow generated by an online game application is added from second 45. The scenario for this set of simulations is depicted in Fig. 2, whereas the RTT values, the start time, and the protocols employed are summarized in Table I.

Therefore, we assume one user in the house engaged in one of the very popular first person shooter games, e.g., Quake or Counter Strike, with other 25 remote players connected through the Internet. To model the packet size and the inter-arrival time of the traffic generated by the online game, we exploit the model suggested in [15], which is based on real game platform measurements. Game events are hence generated at client side every 60 ms, whereas the server transmits back game state updates every 50 ms toward the client. Moreover, game packet sizes generated by the client inside the house and the server are set to 42 bytes and 200 bytes, respectively.

In this context, we have studied the performance achieved in terms of per-packet delays and total throughput achieved when changing the size in Mb/s of the link between the AP and the Internet (the link between AP and W0 in Fig. 2): 4, 10, and 20 Mb/s. When the bandwidth available on this link is less than or equal to 19 Mb/s then it also represents the bottleneck of the network. Otherwise, the bottleneck is located at the wireless link between the AP and the wireless nodes; as said, this oscillates around 19 Mb/s. Since space limitation, for the sake of conciseness we report here only result for the most interesting case: the bottleneck located at the wireless link (*circa* 19 Mb/s) as the factual capacity of the link oscillates, thus challenging our solution. However, the other two cases' outcomes were very similar to the presented one; we are hence not losing relevant information or results by not presenting their results.

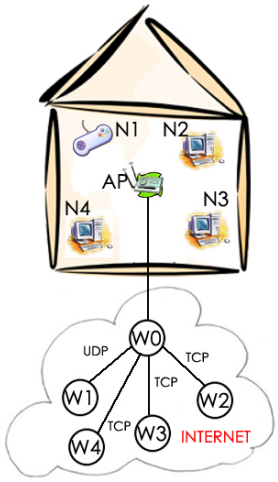

Figure 2. Simulalted scenario.

Table I
SIMULATIVE CONFIGURATION

Flow Type	Protocol	From	To	Start	End	RTT
FTP 1	TCP	N2	W2	1 s	180 s	30 ms
FTP 2	TCP	N3	W3	1 s	180 s	50 ms
FTP 3	TCP	N4	W4	1 s	180 s	100 ms
Online Game	UDP	N1	W1	45 s	180 s	20 ms

IV. EXPERIMENTAL RESULTS

In this section we present results collected through the aforementioned simulative scenario. In particular, we compare the three aforementioned solutions (our SAP-LAW, legacy TCP SACK, and TCP Vegas) in their ability in supporting both real-time flows and elastic ones (even with different RTTs). To this aim, we show and discuss the inter-arrival time of online game packets, the shape of the TCP sending window, and the throughput achieved.

To start with, we compare the three considered solutions based on how much TCP-based flows harm concurrent interactive, real-time flows such as UDP-based game packets. To this aim, Fig. 3, Fig. 4, and Fig. 5 show the inter-arrival time of packets associated with the gaming flow going from the server to the client, when coexisting with three simultaneous TCP-based flows (three regular TCP SACK flows in Fig. 3, three TCP Vegas flows in Fig. 4, and SAP-LAW in Fig. 5). As it is evident, when TCP SACK is employed, the inter-arrival time between consecutive game packets is very variable with continuous peaks surpassing 60 ms or going under 35 ms; this is due to the continuous queuing of packets at the bottleneck. This queuing delay jeopardize the interactivity perceived by online game users [15].

Instead, both TCP Vegas and SAP-LAW demonstrate to be able to limit the oscillations of the game packet inter-arrival time keeping it not very far from the 50 ms of inter-departing time. This is a guarantee of interactivity preservation for the online game application.

As a second evaluation metrics we analyze the RTT-fairness

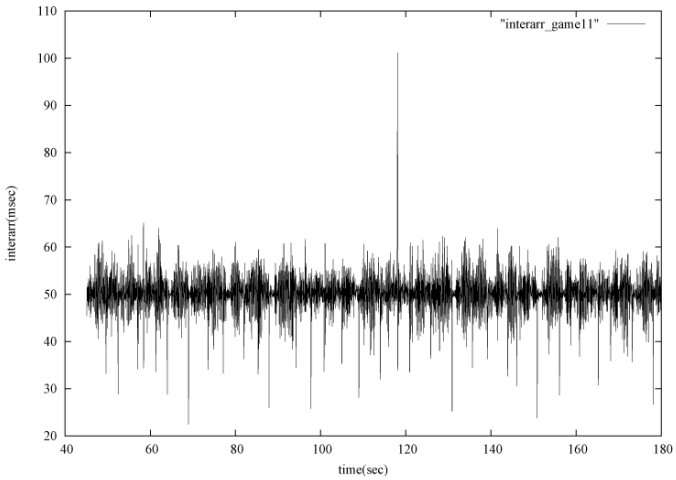

Figure 3. Inter-arrival time of online game packets with three concurrent TCP flows; TCP SACK employed.

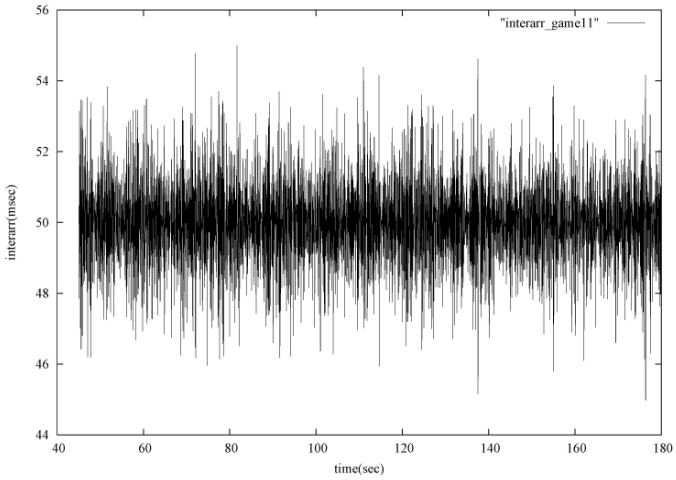

Figure 4. Inter-arrival time of online game packets with three concurrent TCP flows; TCP Vegas employed.

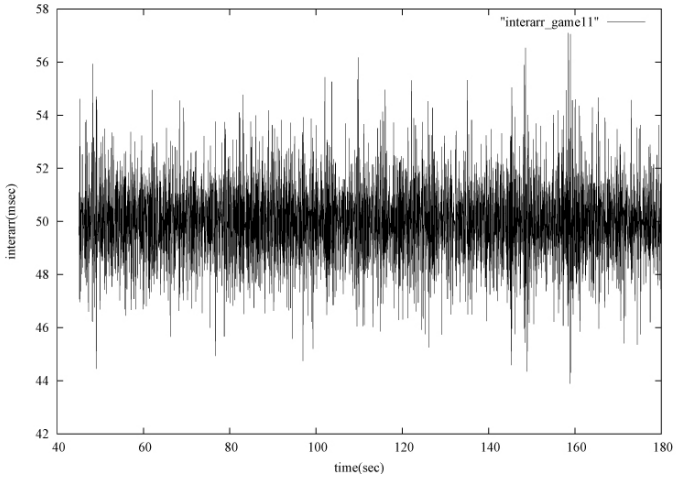

Figure 5. Inter-arrival time of online game packets with three concurrent TCP flows; SAP-LAW employed.

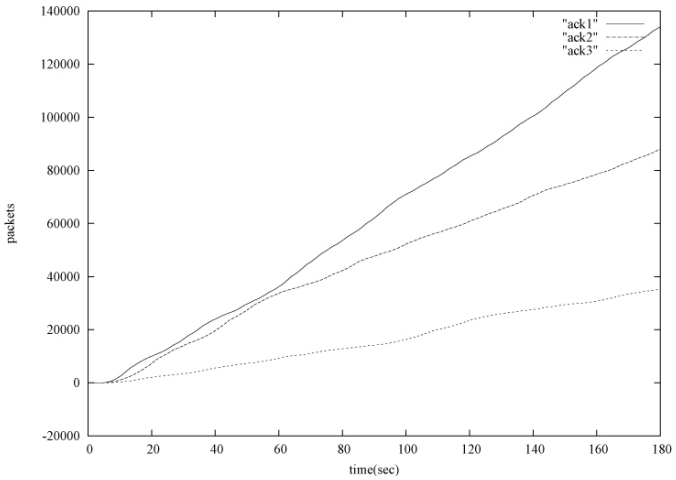

Figure 6. ACKs received back by the senders of three TCP flows with different RTTs; TCP SACK employed.

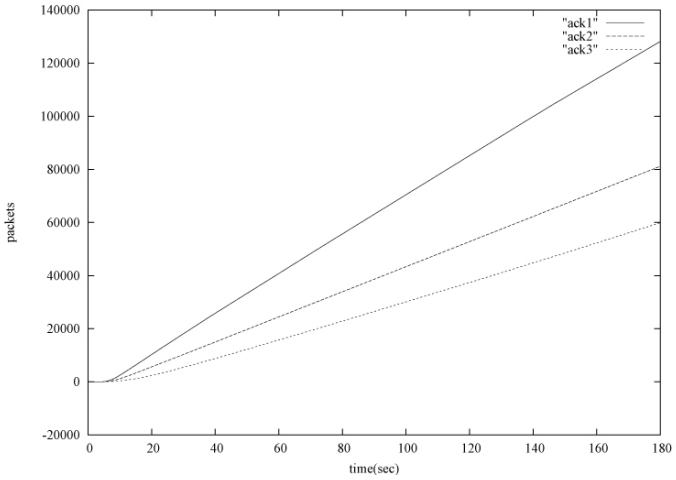

Figure 7. ACKs received back by the senders of three TCP flows with different RTTs; TCP Vegas employed.

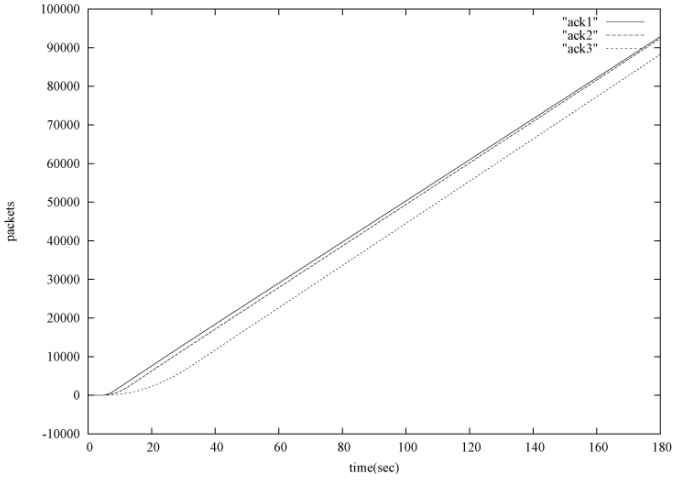

Figure 8. ACKs received back by the senders of three TCP flows with different RTTs; SAP-LAW employed.

ensured by the three compared solutions and, for each of them, we show the returning ACK rates of the three TCP-based flows. Since the tree flows share the same bottleneck we would expect that they achieved the same throughput and hence the same returning ACK ratio. Instead, as anticipated, when the RTTs are different, traditional systems suffer by the RTT-unfairness problem. In fact, Fig. 6 shows the returning ACK rates for the three TCP-based flows when legacy TCP SACK is employed. In the chart, *ack1*, *ack2*, and *ack3*, corresponds to the returning ACKs for the TCP flows with 30, 50, and 100 ms of RTT, respectively. As it is evident, the actual data transmission rates vary inversely with the corresponding RTT duration.

This RTT-unfairness is only partially solved by TCP Vegas, as Fig. 7 shows. In particular, *ack1* and *ack2* have slightly decreased their slopes and *ack3* has increased its slope; yet, a substantial difference remains. Instead, Fig. 8 demonstrates that the problem is completely solved by the use of SAP-LAW: the three ack rates are almost identical thus achieving the aimed RTT-fairness.

So far, we have demonstrated that SAP-LAW is able to avoid TCP-based flows damaging the interactivity performance of real-time UDP-based flows. However, we have also to demonstrate that this is not done at the expenses of loss of efficiency in terms of TCP throughput. To this aim, Fig. 9 shows the average throughput achieved by the three TCP flows together when employing different protocols. As it is evident, even if TCP Vegas and SAP-LAW avoided queuing delays for online game packets, they also achieved to provide TCP flows with a higher average throughput than legacy TCP SACK. This result is due to the fact that each time a TCP flow experiences a loss it halves its transmission rates. Instead, the limitation imposed by TCP Vegas and SAP-LAW to TCP flows was just enough to avoid losses without wasting bandwidth. Even better, SAP-LAW achieves the higher average throughput with a final bandwidth utilization that is very close to the total available of 19 Mb/s (also considering that about 32 Kb/s are occupied by the simultaneous online game flow).

V. Conclusion

In this article we have investigated two important problems affecting the performance of typical multimedia applications run in a wireless home. The first one is how to facilitate the coexistence among heterogeneous flows in a shared wireless communication channel, whereas the second one regards RTT-fairness of TCP-based flows.

We have compared our SAP-LAW solution with the use of regular TCP SACK or the use of TCP Vegas for the TCP-based flows. Experimental results showed that SAP-LAW outperforms the other two solutions as it is the only solutions among the tested ones to achieve all the three aforementioned goals.

Finally, another very important advantage of SAP-LAW is that it can be employed through a very simple formula that can be easily introduced on real APs.

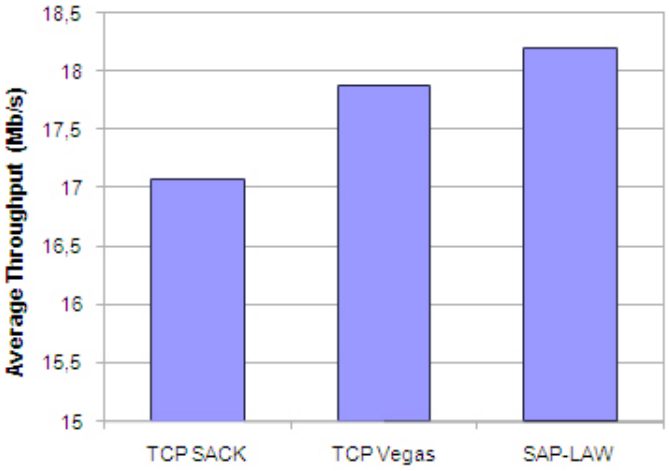

Figure 9. Average thrughput for three simultaneous flows when employing different protocols.

References

[1] M. Furini, "Mobile games: What to expect in the near future," in *Proc. of GAMEON 2007*, Bologna, Italy, Nov 2007.

[2] A. Ploss, S. Wichmann, F. Glinka, and S. Gorlatch, "From a single- to multi-server online game: A quake 3 case study using rtf," in *Proc. of ACM ACE 2008*, Yokohama, Japan, Dec 2008.

[3] A. Balk, D. Maggiorini, M. Gerla, and M. Sanadidi, "Adaptive mpeg-4 video streaming with bandwidth estimation," in *Proc. of 2nd QOS-IP 2003*, Milan, Italy, Feb 2003.

[4] A. Balk and Dario Maggiorini and M. Gerla and M. Sanadidi, "Adaptive mpeg-4 video streaming with bandwidth estimation: Journal version," vol. 44, no. 4, pp. 415–439, Mar 2004.

[5] S. Ferretti, "A synchronization protocol for supporting peer-to-peer multiplayer online games in overlay networks," in *of the 2nd ACM DEBS'08*, Rome, Italy, Jul 2008.

[6] S. Ferretti, "Cheating detection through game time modeling: A better way to avoid time cheats in p2p mogs?" *Multimedia Tools and Applications, Springer*, vol. 37, no. 3, pp. 339–363, May 2008.

[7] V. Ghini, G. Pau, and P. Salomoni, "Always best served music distribution for nomadic users over heterogeneous networks," *IEEE Communication Magazine Entertainment Everywhere: System and Networking Issues in Emerging Network-Centric Entertainment Systems*, vol. 43, no. 5, pp. 69–74, May 2005.

[8] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, and M. Gerla, "What's in that magic box? the home entertainment center's special protocol potion, reealed," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1280–1288, Nov 2006.

[9] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Y. Sanadidi, and M. Roccetti, "Tcp libra: Exploring rtt-fairness for tcp," in *Proc. of the IFIP/TC6 NETWORKING 2007*, Atlanta, GA, USA, May 2007.

[10] C. Caini and R. Firrincieli, "Tcp hybla: A tcp enhancement for heterogeneous networks," *International Journal of Satellite Communications and Networking*, vol. 22, no. 5, p. 547566, 2004.

[11] L. S. Brakmo, S. W. OMalley, and L. L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *Proc. of the ACM SIGCOMM 1994*, 1994.

[12] S. H. Low, L. L. Peterson, and L. Wang, "Understanding tcp vegas: A duality model," in *Proc. of the ACM SIGCOMM 2001*, 2001.

[13] "The network simulator, ns-2," http://www.isi.edu/nsnam/ns/.

[14] A. L. Wijesinha, Y. Song, M. Krishnan, V. Mathur, J. Ahn, and V. Shyamasundar, "Throughput measurement for udp traffic in an ieee 802.11g wlan," in *Proc. of SNPD/SAWN'05*, Towson, MD, USA.

[15] J. Farber, "Traffic modelling for fast action network games," *Multimedia Tools and Applications*, vol. 23, pp. 31–46, 2004.