Exploiting TCP Vegas' Algorithm to Improve Real-Time Multimedia Applications

Michele Massaro, Claudio E. Palazzi, Armir Bujari
Department of Mathematics
University of Padua
Padua, Italy
Email: michele.massaro@me.com, cpalazzi@math.unipd.it, abujari@math.unipd.it

***Abstract*—Computer-centered services and broadband wireless connectivity have boosted the demand for delivery of multimedia-based entertainment from the Internet to in-house wireless appliances. In this context, rich-media applications can be supported by a variety of protocols, each sharing the same bottleneck, affecting one another's performance. Indeed, in current systems, real-time applications (e.g., video streaming, online games) suffer from delays caused by the interference of elastic, best-effort (e.g., TCP-based download sessions) traffic. In this paper we propose a solution to this problem, addressed by deploying a Vegas-like congestion control algorithm at the home gateway, requiring no client-side modification.**

Access Point, Home Network, Online Game, TCP Vegas, Wireless

I. Introduction

In no time, the Internet has grown from an interesting distraction into an essential part of our everyday life. Broadband access technology has grown considerably both for entertainment and for communication, leading to a demand for an improved networking support able to satisfy the requirements of heterogeneous traffic flows. A representative example in this category are TCP-based elastic flows (e.g., download) and emerging UDP-based real-time ones (e.g., video stream, online games, etc.).

In this context, the coexistence of elastic and real-time network traffic presents severe issues particularly affecting the latter. The main problem derives from the nature of classic TCP protocols, which are loss-based; continually probing the shared channel in order to increase the transfer speed until some loss occurs. This growth phase is succeeded by a slow down phase, where the flow starts to grow its sending speed again, and these phases are interleaved in a cyclic fashion.

This *modus operandi* cause the queues to fill along the channel, leading to an increase of the per-packet delivery time [1], [2]. Even if this does not represent a major issue for elastic applications, it causes a significant decrease of the quality of service for real-time applications [3]–[5]. Addressing this issue, delay-based TCP congestion protocols employ packet RTT rather than losses to prevent congestion [6]. In this context, TCP Vegas is the main candidate [8]; continually monitoring the packet queueing times and shown to lead to a better quality for UDP services.

However, TCP Vegas cannot be used in a channel shared by loss-based protocols due to the aggressiveness of the latter in getting more and more bandwidth until a loss event happens, thus capturing almost all the available bandwidth from delay-based flows. The only way to employ TCP Vegas in practice would be to completely dismiss all loss-based TCP versions, changing the protocol of every client and server connected to the Internet. Needless to say, this is not a feasible option.

In this paper, we show how to practically exploit the benefits of delay-based protocols, intervening only in the user's Access Point (AP). As a result of this, elastic applications achieve high data throughput, while real-time application maintain a low latency. The AP is enhanced with an algorithm that automatically limits TCP flows when the channel is near to the saturation (like TCP Vegas), avoiding long queues and packet loss while keeping the TCP-based flows at a data rate level which corresponds to a full utilization of the available bandwidth.

In Section II we discuss the state of the art relevant to this domain, while in Section III we present our solution in detail. Section IV provides an overview of the simulation environment and comparison settings, while in Section V and Section VI we discuss the experimentation outcome. Finally, Section VII concludes this paper.

II. Related Works

The problem we are addressing is not new and solutions have been proposed to avoid the latency increase incurred by TCP New Reno's congestion control. As already stated above, the problem resides in the protocol itself, this to say that one solution would be to switch from the current protocol to a delay-based one, Vegas-like. Indeed TCP Vegas is able to detect the congestion in advance using the RTT fluctuation of the packets. Simply said, it increases the transfer speed when the delay is under an α threshold and decreases it when the delay is over β ($\alpha < \beta$). In this way, it is not necessary to lose a packet in order to detect congestion, because it can be detected before it happens. Clearly this also avoids the creation of bottleneck queues and corresponding queuing delays that would harm any real-time application [7].

Unfortunately, TCP Vegas flows cannot coexist with loss-based ones, e.g., generated by TCP New Reno and other classic real life TCP versions; loss-based flows cause congestion by their nature, and delay-based ones detect it as an imminent congestion, slowing down their transfer rate. The result is that loss-based flows will occupy almost all the channel [9], [10]. This incompatibility with legacy protocols made Vegas hardly applicable.

Instead, a solution that would not require a modification of Internet protocols was proposed in [10], [11] and named Smart Access Point with Low Advertised Window (SAP-LAW). Basically, this solution leverages on the ability of the Access Point to monitor all traffic passing through; then, the advertised window of TCP-based flows are appropriately modified to bound them to their fair bandwidth share.

The algorithm detects the ongoing UDP traffic, and computes the available bandwidth, as well as the corresponding appropriate advertised window, for each flow through (1) (three flows in the example).

$$maxTCPrate_i(t) = \frac{(C - UDPtraffic(t)) * RTT_i}{a_1 + a_2 + a_3}$$
$$a_1 = \frac{RTT_1}{avg.RTT_{min}}, a_2 = \frac{RTT_2}{avg.RTT_{min}}, a_3 = \frac{RTT_3}{avg.RTT_{min}} \quad (1)$$

This avoids the classic TCP's congestion window fluctuations, stabilizing it to a computed value. Moreover, the absence of congestion alleviates delay problems for UDP packets, allowing the user to enjoy interactive, real-time services while concurrent elastic applications still achieve high throughputs.

However, this proposal is limited as it requires an *a priori* knowledge of the bottleneck capacity and the RTT of each flow; these parameters may not be always known or correctly computed or even stable enough to be used in a timely fashion.

Similar in spirit, even it has been proposed in the context of multi-hop wireless networks, is Gateway Adaptive Pacing (GAP) [12], [13]. GAP adds an artificial delay between each packet, based on continuous measurements of the network. In this way the interarrival time decrease, but so does even the throughput of TCP flows, thus finding scarce applicability in the regular one-hop case.

III. VoAP

Our goal is to create a solution that could reach SAP-LAW's performance without requiring the *a priori* information needed by SAP-LAW. At the same time, we want to avoid any need for changing the network protocols of the devices connected to the network (with the only exception of the Access Point). The algorithm is designed mainly for the wireless home scenario, where the main bottleneck is represented by the wireless component; however it was kept generic, and it should work even in a more general or wired environment.

Like SAP-LAW, our algorithm is designed to be deployed on an Access Point, so as to be able to monitor all the network traffic in real-time. To not use the additional information on bottleneck capacity and RTTs as done by SAP-LAW, we have taken inspiration from TCP Vegas' congestion control algorithm. From here the name of our algorithm: Vegas over Access Point (VoAP).

The main idea at the basis of VoAP is to monitor all the TCP flows that pass through the AP and measure how long packets remain in its queue before being transmitted over the wireless channel. When this time is below an α threshold it means that the channel is quite free, thus allowing the TCP-based flows to increase their transfer speed without the risk of causing congestion/queuing delays; instead, when the time is over the β threshold, the channel is saturating, and the TCP flows' transmissions should be slowed down as they are not transmitting at higher rates, they are just building up queues.

The increase of the TCP's rate is regulated by its own congestion avoidance protocol, whereas to decrease it we use the advertised window already present in the header of regular ACKs. Therefore, any TCP-based flow transmitted in the considered wireless home, regardless of its congestion window flavor, will be limited in its regular growth if and only if it is detected to cause congestion in the AP. This is an appropriate behavior as to higher transmission rates correspond higher queuing delay, rather than higher receiving rates. Furthermore, as our solution can only limit the TCP's data rate, we do not have a negative impact on other flows present in the Internet core. Finally, as VoAP is applied to all and only the TCP-based flows passing through the considered Access Point, we do not incur the risk to have an excessive limitation of the congestion window caused by the aggressive behavior of other congestion control variants as original TCP Vegas does.

To allow a fast initial growth (slow start) the algorithm is not applied below a packet threshold of the congestion window (10 packets) and the flow is left free until it exceeds for the first time the β threshold; at that point, the algorithm intervenes enforcing the limitation through appropriate modifications of the advertised window, thus decreasing the flow speed until the delay drops under β.

When the delay stays between α and β, the advertised window remains stable, preventing the flow growth, since the channel is almost saturated. If the channel becomes free, the delay drops under the α threshold, so the speed is increased again.

The data are collected for a period of time *T*, where the number of flows and their delays are checked and acted upon accordingly. Every increment or decrement action is per unit, so as to avoid excessive fluctuations, thus there can be at most *1/T* changes of the advertised window per second. After this, the timers are reset and the delay sampling restarts. Furthermore, the time range T is needed to observe whether a flow is active or not, and split the bandwidth in a fair way; it is therefore necessary for T to be small enough to not waste bandwidth on a non-active flow, but also big enough to allow all active flows to transmit at least one packet.

The detailed flow chart of the algorithm can be seen in the Figure 1.

IV. Simulation

To implement the algorithm we chose the network simulator NS2 [14]. First, it was necessary to understand in which layer the algorithm should be added, and we decided to implement it inside the link layer, since it is directly connected to the FIFO queue of the outgoing packets. The considered scenario is described in Figure 2, where the *W* nodes represents remote servers in the Internet, and *node_* represent all the wireless devices connected to the AP (where VoAP is implemented). All the wired connections have a 100 Mbps capacity and the wireless channel is based on IEEE 802.11g.

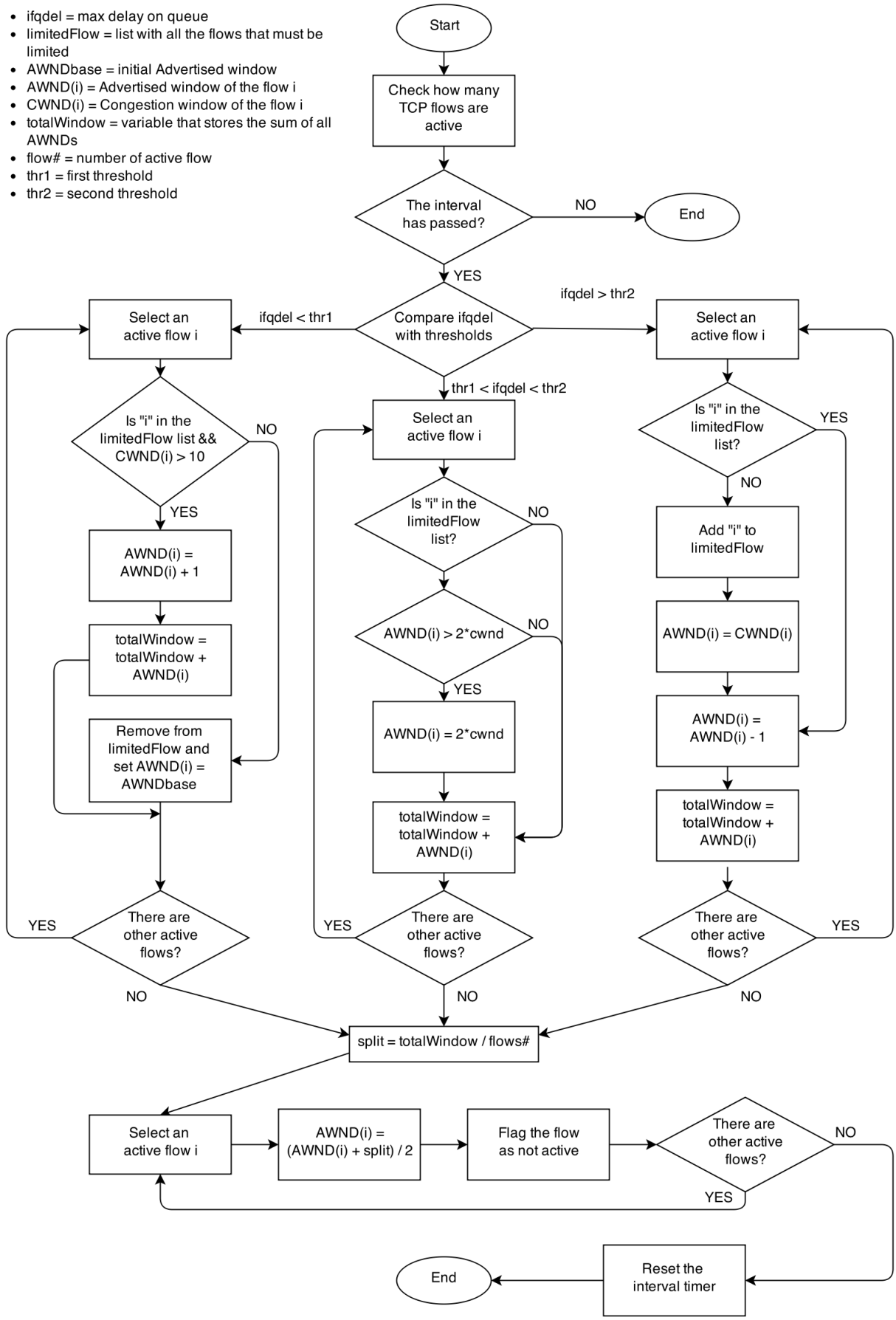

Fig. 1. Flow chart of the VoAP algorithm.

TABLE I. NETWORK FLOWS

Name	From	To	Type
TCP(1)	W(3)	node_(3)	Download (FTP)
UDP(0)	BS(0)	node_(0)	Movie Streaming
UDP(11)	W(1)	node_(1)	Game Server to Client
UDP(12)	node_(1)	W(1)	Game Client to Server
UDP(21)	W(2)	node_(2)	Voice chat Server to Client
UDP(21)	node_(2)	W(2)	Voice chat Client to Server

TABLE II. START/STOP TIMES

Name	Start (s)	End (s)
TCP(1)	135	250
UDP(0)	0	250
UDP(11)	45	250
UDP(12)	46	250
UDP(21)	90	250
UDP(21)	91	250

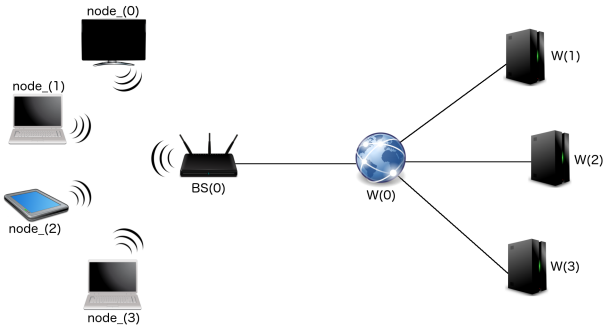

Fig. 2. VoAP considered scenario.

To generate trustworthy simulations, we have considered the flows described in Table I, with the start and end time described in Table II.

This scenario has been used in all the simulations, except for those with multiple TCP flows, in which we included other nodes and additional flows. The network topology is depicted in Figure 2, where all the wireless nodes are at the same distance from the AP (10 m).

V. RESULTS

In this section we start by validating our proposal in presence of single and multiple flows exhibiting different characteristics. Several aspects of the algorithm are discussed and contrasted with current state of the art, showing its performance trend under different performance metrics.

A. One flow

First, we tried to understand the algorithm's behaviour compared to the standard TCP New Reno, and to do so we make use of a single TCP flow in the previously discussed scenario. The α and β thresholds have been set equal to 5 ms and 15 ms respectively, and the time range T has been set equal to 200 ms.

With this configuration, Figure 3 shows how the TCP flow behaviour goes from the standard sawtooth shape to a smoother one, with slightly lower peaks but with a higher number of transferred packets. The lower peaks avoid queuing delays at the AP, allowing to reach lower overall delays for UDP packets.

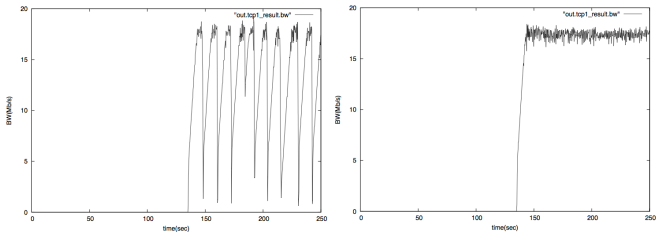

Fig. 3. Bandwidth comparison between New Reno (left) and VoAP (right).

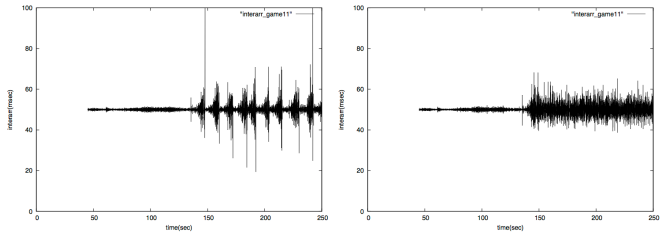

Fig. 4. Game packet inter-arrival comparison between New Reno (left) and VoAP (right).

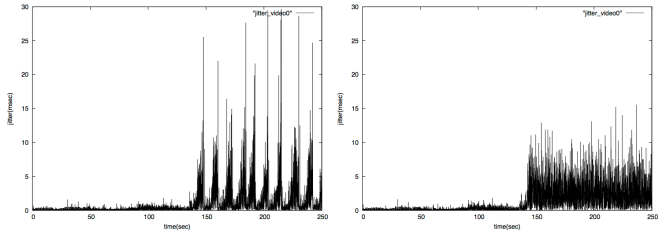

Fig. 5. Video jitter comparison between New Reno (left) and VoAP (right).

This is confirmed by Figure 4, where the peaks relative to the TCP congestion are almost absent. This increased stability can be further noticed by monitoring the jitter, e.g., of the video stream one, depicted in Figure 5, where the peaks are reduced by a factor of 2. In Figure 3 we can also notice that the slow start phase is not affected by the algorithm, indicating that the discussed delayed activation of VoAP leads to the desired result.

All the UDP flows in the considered scenario experienced a lower delay and jitter as expected, and the benefits for the TCP flow does not only concern the stability, but also the amount of transferred data. To better understand the achieved improvement, we counted the TCP flow's ACKs: with TCP New Reno we reached 181 MB of transferred data, whereas with VoAP we reached 222 MB, leading to an increment of about 22%.

B. Multiple TCP flows

The algorithm behaviour in a multi-TCP-flow scenario is fundamental to understand whether the bandwidth is equally distributed among users sharing the same channel. To this end, when active TCP flows reach the stable interval between the α and β thresholds, the sum of the advertised windows indicate how many packets can be in transit without exceeding the channels capacity. This information is used to assign a fair share of bandwidth to every TCP flow.

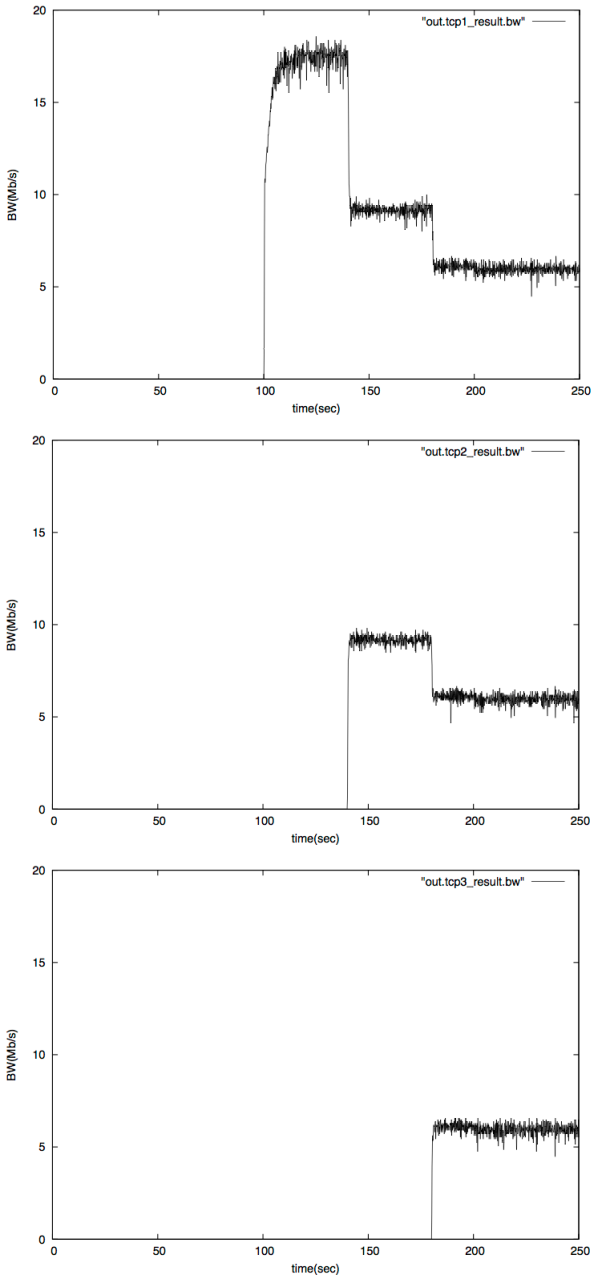

Fig. 6. Bandwidth of 3 TCP flows using VoAP.

In order to avoid excessive fluctuation, the bandwidth share is computed in a smoothed way, i.e., the average between the old and the new value at every T. The result can be seen in Figure 6, where three TCP flows were started in three different moments in time, this to show how the bandwidth is equally shared between flows that are already exploiting the wireless channel and new starting ones.

Obviously, even if the flows started at the same time the result would have been the same; we also made additional tests employing five TCP flows, and each one succeed to obtain an equal share of the bandwidth.

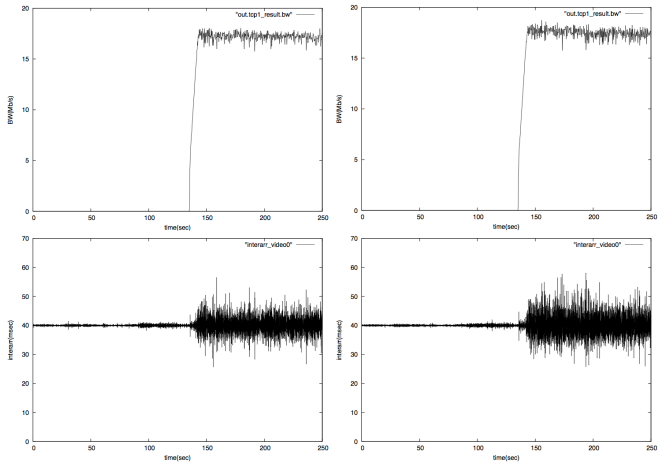

Fig. 7. Comparison between β=0.010s (left) and β=0.020s (right).

C. Thresholds

In the current version of our algorithm, α and β thresholds are static values with the higher one corresponding to the maximum amount of queuing time we are ready to suffer. Their choice entails a different behaviour of the algorithm, because α and β denote how short the queue must be, decision that has an impact on both the UDP packet latency and the TCP transfer rate. Due to the importance of these variables, we have tested different values, where the β threshold varied between 10 ms and 20 ms.

Adopting a low threshold, the maximum queue time is reduced, so the UDP packet latency become lower; however, in this way the TCP flow is penalized, because its advertised window becomes smaller, leading to a lower throughput. Employing a high threshold instead, allows the TCP flow to increment the number of outgoing packet, leading to a higher bandwidth, penalizing UDP packets that will experience a higher latency. The comparison between the two instances can be seen in Figure 7, where the two borderline cases are measured with the TCP bandwidth and the UDP game interarrival time.

It is noteworthy to point out that in both the cases, the latency and the bandwidth are noticeably better than what is obtained with New Reno. The instance 5-15 ms is a compromise between the two behaviours, and for this reason it has been chosen as the default range.

VI. Comparison with SAP-LAW

Since both SAP-LAW and VoAP use the AP as control center, we run some trials employing the same network deployment mentioned before but contrasting the two the approaches. To this aim, Figure 8 shows how the TCP flows' bandwidth and the UDP game's interarrival time change in the schemes, respectively. The bandwidth is improved by VoAP, both in terms of quantitative value and stability. The interarrival time results slightly lower using SAP-LAW; yet its value with VoAP is low enough to not jeopardize the performance of the application and, conversely from SAP-LAW, VoAP achieves this result without the need of prior information about bottleneck capacity and the flows' RTT.

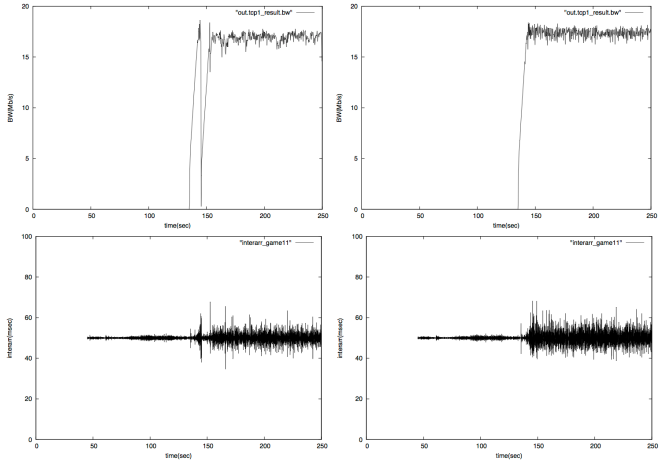

Fig. 8. Bandwidth (above) and interarrival (below) comparison between SAP-LAW (left) and VoAP (right).

VII. Conclusion

We have presented and discussed a solution, named VoAP, addressing the coexistence problem between TCP and UDP flows. Our proposal provides to both kinds of flow the possibility to reach high throughput and low per-packet delay. We used NS2 to demonstrate the effectiveness of the algorithm in significantly improving the network performance of heterogeneous flows in a realistic wireless home configuration. The goal has been achieved through the implementation of a TCP Vegas-inspired algorithm implemented in the Access Point. Therefore, our solution works in a transparent way for the user and without the need to modify any client, server or Internet router. This means that a user that wants to exploit our solution to improve the network support for his applications simply has to buy and install our AP or update his AP with our VoAP software.

Modules and other software useful to implement VoAP in NS2 are available online [15].

As this work has many interesting application in realistic scenarios, we plan to continue our research in several directions. For instance, future extension of this work could include vehicular networks so as to test the adaptability of our solution when considering highly dynamic nodes frequently changing the AP they are connected to [16]–[18]. Furthermore, we would like to devise a distributed version of our solution able to operate even with mobile nodes connected through mesh networks [19], [20], or in ad-hoc mode [21], [22], or even through opportunistic and DTN connectivity [23], [24].

References

[1] G. Marfia, M. Roccetti, "TCP At Last: Reconsidering TCP's Role for Wireless Entertainment Centers at Home", IEEE Transactions on Consumer Electronics 56(4), 2010.

[2] C. E. Palazzi, G. Pau, M. Roccetti, M. Gerla, "In-Home Online Entertainment: Analyzing the Impact of the Wireless Mac-Transport Protocols Interference", in Proc. of IEEE International Conference on Wireless Networks, Communications and Mobile Computing (WIRELESSCOM), Maui, HI, USA, 2005.

[3] A. Kaiser, D. Maggiorini, N. Achir, K. Boussetta, "On the Objective Evaluation of Real-Time Networked Games", in Proc. of GLOBECOM 2009, Honolulu, HI, USA, 2009.

[4] J. Saldana, M. Suznjevic, L. Sequeira, J. Fernandez-Navajas, M. Matijasevic, J. Ruiz-Mas, "The Effect of TCP Variants on the Coexistence of MMORPG and Best-Effort Traffic", in Proc. of ICCCN 2012, Munich, Germany, 2012.

[5] S. Gorlatch, A. Ploss, "Towards a Scalable Real-Time Cyberinfrastructure for Online Computer Games", in Proc. of ICPADS 2009, Shenzhen, China, 2009.

[6] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Roccetti, "TCP Libra: Derivation, analysis, and comparison with other RTT-fair TCPs", Elsevier Computer Networks 54(14), 2010.

[7] S. H. Low, L. L. Peterson, L. Wang, "Understanding TCP Vegas: a duality model", Journal of the ACM (JACM) 49(2), 2002.

[8] H. Choe, S. Low, "Stabilized vegas", Advances in Communication Control Networks, 2005.

[9] L. S. Brakmo, L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communications 13(8), 2002.

[10] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "What's in That Magic Box? The Home Entertainment Center's Special Protocol Potion, Revealed", IEEE Transactions on Consumer Electronics 52(4), 2006.

[11] C. E. Palazzi, N. Stievano, M. Roccetti, "A Smart Access Point Solution for Heterogeneous Flows", in Proc. of the International Conference on Ultra Modern Telecommunications and Workshops (ICUMT), St. Petersburg, Russia, 2009.

[12] S. M. El Rakabawy, A. Klemm, C. Lindemann, "Gateway Adaptive Pacing for TCP Across Multihop Wireless Networks and the Internet", in Proc. of ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM'06), Torremolinos, Spain, 2006.

[13] K. Kim, D. S. Niculescu, S. Hong, "Coexistence of VoIP and TCP in Wireless Multihop Networks", IEEE Communications Magazine 47(6), 2009.

[14] The Network Simulator 2, "NS-2" [ONLINE]. Available: http://www.isi.edu/nsnam/ns

[15] VoAP [ONLINE].
Available: http://www.math.unipd.it/ cpalazzi/VoAP.html

[16] C. E. Palazzi, S. Ferretti, M. Roccetti, G. Pau, M. Gerla, "How do you quickly choreograph inter-vehicular communications? A fast vehicle-to-vehicle multi-hop broadcast algorithm, explained", in Proc. of the 4th Annual IEEE Consumer Communications and Networking Conference, CCNC 2007, Las Vegas, NV, USA, Jan 2007.

[17] C. E. Palazzi, M. Roccetti, S. Ferretti, "An intervehicular communication architecture for safety and entertainment", IEEE Transactions on Intelligent Transportation Systems 11(1), 2010.

[18] G. Marfia, G. Pau, E. De Sena, E. Giordano, M. Gerla, "Evaluating vehicle network strategies for downtown Portland: Opportunistic infrastructure and the importance of realistic mobility models" in Proc. of the First International MobiSys Workshop on Mobile Opportunistic Networking (MobiOpp'07), San Juan, Puerto Rico, 2007.

[19] M. Di Felice, K. R. Chowdhury, A. Kassler, L. Bononi, "Adaptive sensing scheduling and spectrum selection in cognitive wireless mesh networks", in Proc. of International Conference on Computer Communications and Networks, ICCCN, Maui, HI, USA, 2011.

[20] K. R. Chowdhury, M. Di Felice, L. Bononi, K.R. Chowdhury, M. Di Felice, L. Bononi, "XCHARM: A Routing Protocol for Multi-channel Wireless Mesh Networks", in Elsevier Computer Communications 36(14), 2013.

[21] L. De Giovanni, C. E. Palazzi, "Optimal client-server configuration of mobile ad-hoc networks", Electronic Notes in Discrete Mathematics 41, 2013.

[22] M. Gerla, D. Maggiorini, C. E. Palazzi, A. Bujari, "A survey on interactive games over mobile networks", Wireless Communications and Mobile Computing 13(3), 2013.

[23] D. Maggiorini, C. Quadri, L. A. Ripamonti, "Opportunistic mobile games using public transportation systems: a deployability study", Multimedia Systems, 2014.

[24] S. Gaito, D. Maggiorini, G. P. Rossi, A. Sala, "Bus switched networks: An ad hoc mobile platform enabling urban-wide communications", Elsevier Ad Hoc Networks 10(6), 2012.