Smart Access Points on the Road for Online Gaming in Vehicular Networks

Claudio E. Palazzi[a], Stefano Ferretti[b], Marco Roccetti[b]

[a]*Dipartimento di Matematica Pura e Applicata, Università degli Studi di Padova, Italy*
[b]*Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy*

Abstract

Online games represent one of the most important revenue sources for entertainment based companies and a challenging field in multimedia application research. With vehicular networks poised to become the new wireless frontier of the Internet, car passengers embody the next consumers that will be targeted by online game providers. Yet, the high mobility and heterogeneity of vehicular networks pose serious challenges; previous work on online games demonstrated the importance of the network's performance in determining the quality level perceived by consumers. A main problem is related to competing heterogeneous network traffic generated by real-time multimedia applications and concurrent bulk data traffic sharing the same access points along the road. Such problem causes low responsiveness in the gaming application and is further exacerbated by the continuous variations in the number and type of flows, due to the cars' mobility. To this aim, we show how smart access points can be deployed in infrastructure-based vehicular networks to ensure efficient coexistence among heterogeneous types of flow even in presence of frequent network traffic variations due to the vehicles' high mobility. As a result, delivery delays are kept small, satisfying the main requirement for the deployment of responsive online games.

Key words: Online Games, Smart Access Point, Entertainment, Vehicular Networks

1. Introduction

Wireless vehicular networks are soon going to be a reality thanks to the factual interest shown by many Governments and to the market responsiveness toward new technology, both for work and entertainment, offered today on cars. Indeed, it is no secret that the US and EU governments have reserved the 5.9 GHz frequency spectrum to implement vehicular networks, both in infrastructure and ad-hoc mode, through the DSRC/IEEE 802.11p standard [1]. This suggests that in a few years we will have city streets and highways mostly covered by wireless connectivity.

Applications run by vehicular users will be both classic ones already utilized every day on home/office PCs (e.g., email, web-surfing, chatting, online gaming, video streaming) and new ones specifically designed for the vehicular context (e.g., traffic safety, driving directions, location based data collection, parking lot payment).

Clearly, such a scenario presents many novel issues that deserve scientific investigation. In this context, our aim is that of supporting online gaming even for passengers traveling in cars. We are interested in this class of real-time multimedia applications both for its strict performance requirements and for its huge success among consumers.

More in detail, we investigate the coexistence between elastic (TCP-based) and real-time (UDP-based) applications in infrastructured vehicular networks and propose a solution able to improve it. In fact, whereas every computer science student is taught that the lack of congestion control in UDP-based flows is a potential harm toward TCP-based flows, we show that in the UDP vs TCP quarrel even the latter represents a source of problems, since persistent TCP-based flows are responsible for performance deterioration of concurrent UDP-based applications [2]. We already investigated such problem in home scenarios, where several users in the same house share the same wireless link for running their online applications [3]. In vehicular networks, this problem is obviously exacerbated due to the definitely higher number of users sharing the same wireless link. Moreover, the high mobility of nodes (i.e., cars) traveling from the coverage area of an access point (AP) to another one generates continuous variations in the number and type of flows served by APs along the road.

To address this problem, we propose the use of "smart APs" along the road, able to regulate heterogeneous transmission flows and make them coexist efficiently. With our solutions, the AP continuously snoops transiting packets of the various flows and computes the maximum data rate at which each elastic application will be able to transfer their files without incurring in congestion losses. This data rate is used to compute an appropriate advertised window that is on-the-fly included in transiting ACKs of TCP flows [4].

We name this solution, Smart Access Point with Low Advertised Window

(SAP-LAW). As a result of its employment, elastic applications produce a smooth traffic that efficiently utilizes the available channel without incurring in congestion losses that would degrade their performances and, at the same time, does not create queues at the AP that would increase per-packet delays of real-time applications. A point in favor of this scheme is also the fact that it entirely operates at the AP. Therefore, changes in the number and type of flows served by the considered AP are immediately detected and addressed. Moreover, SAP-LAW does not require changes in the whole Internet, but just the deployment along the road of APs with SAP-LAW features, which can happen also gradually.

We demonstrate how SAP-LAW represents a valid solution to find the best tradeoff solution between the throughput achieved by elastic applications and the per-packet delay of real-time ones, even in a challenging scenario such as vehicular networks. In particular, in Section 2 we provide background information to understand the considered problem. Section 3 outlines our proposed solution. The experimental assessment and collected results are shown in Section 4 and Section 5, respectively. Finally, conclusions are drawn in Section 6.

2. Background

In this section we provide a framework model that highlights the main delay sources in the considered scenario and explain the tradeoff relationship existing among heterogeneous applications that can be simultaneously present in this context.

2.1. System Model

Online gaming represents a prominent example of real-time application as interactivity, i.e., the fast delivery of game events, embodies its main requirement. At the same time, requirements for available bandwidth and delivery reliability can be relaxed as games utilize very small packets (even just few tens of bytes) and can tolerate some packet losses [5].

Every class of game is featured by a peculiar Game Interactivity Threshold (GIT) that represents the maximum delay endurable before visualizing a game event on players' screens if one wishes to preserve interactivity. The typical GIT for fast-paced interactive games (i.e. vehicle racing, first person shooter) is around 100 ms, even if this value can be increased up to seconds in case of slow paced games (i.e., strategic, role play game) [6, 7].

If we call $t^{g(e)}$ the generation time of event e and $t_i^{v(e)}$ the visualization time of the same event at player i, then interactivity is preserved at i during the delivery of e when the following condition is satisfied:

$$t_i^{v(e)} - t^{g(e)} \leq GIT. \tag{1}$$

Beside interactivity, other two important features of online games are represented by consistency and fairness. The former is related to the uniformity of the game state evolution seen by all nodes in the system, whereas the latter refers to providing all players with the same chances to win regardless, for instance, of networking conditions.

Consistency and fairness share similar goals as, practically, they would both benefit from having a simultaneous evolution of the game state on all the nodes of the system. Therefore, the same class of techniques is generally used to achieve each of them (or both); a prominent example of these common solutions is represented by local lag techniques, which are based on the local introduction of different artificial delays in order to contemporary visualize game events on the screens of all remote players [8, 9].

In essence, since the generation time of each game event is unique and naming P, the set of players, we can say that we have event-related fairness [10] for event e if the following condition is satisfied; simply stated, if there is a unique $t^{v(e)}$ value for all the players:

$$t_i^{v(e)} = t^{v(e)} \ \forall i \in P. \tag{2}$$

Since a single game event experiences different overall delivery delays (DDs) in its path from the source to all the diverse destinations, different amounts of artificial delay δ should be added in order to contemporary visualize the same event e on all the players' screens. We can hence rewrite (2) into the following condition:

$$t^{g(e)} + DD_i(e) + \delta_i(e) = t^{v(e)}. \tag{3}$$

A possible value typically chosen for the unique $t^{v(e)}$ is represented by the highest DD in transmitting events amongst nodes. When the highest DD is greater than GIT, however, fairness is preserved at the cost of jeopardizing interactivity for all the players. To maximize the possibility to obtain both interactivity and fairness, $t^{v(e)}$ should be set as

$$t^{v(e)} = t^{g(e)} + GIT. \tag{4}$$

Yet, with (4) we may not be able to guarantee fairness for every player in the system, but only for a subset of players whose interconnections are featured with DD values smaller than the GIT. In point of this, the best way to increase the size of this subset of players who experience a fair (and interactive) gaming experience, passes through providing the online game system with means for improving the general interactivity degree of the system [11]. In essence, by reducing unnecessary computational and networking overhead the gaming system delivery delays can be generally lowered thus allowing more players, even located at farther distances, to be accommodated in the same game session while still maintaining the highest DD in the system below the GIT.

Since the DD plays such a fundamental role in ensuring interactivity, fairness, and consistency, we further analyze it. The $DD_i(e)$ experienced by an event e when it finally reaches player i is composed by several delay components; respectively: physical latency $ld_i(e)$, queuing time $qd_i(e)$ on nodes along the path, and processing time $pd_i(e)$. Therefore, $DD_i(e)$ can be written as

$$DD_i(e) = ld_i(e) + qd_i(e) + pd_i(e). \tag{5}$$

A very important component in (5) is represented by $qd_i(e)$; indeed, as we show in Section 5, queuing embodies a significant waste of time that can cause high DD values and jeopardize the perceived performance of the online gaming application. Even worse, the DD could be not only high but also highly variable (i.e., having high jitter), thus impeding to players to adapt to network delays (for instance, constantly anticipating the steering at each curve when playing car racing games) [7]. This is especially true in vehicular networks, since APs along the road experience continuous variations in the number and type of flows that each of them has to serve. Considering its importance, we devote the following subsection to further elaborate on queuing delay built up at the APs.

2.2. Access Point's Queuing Delay

As just discussed, the queuing delay qd represents a crucial component in the DD. Elaborating more on the wastage of time caused by packet queuing, we can say that queues are built up along the path from the sender to the receiver when the arriving rate of events at some node is higher than the serving rate of that node.

However, as recently demonstrated by measurements on a real OC48 link, the capacity of the Internet is generally larger than the aggregate bandwidth utilized by transiting flows [12]. Moreover, providers are offering today guaranteed high speed connectivity to home customers. This implies that the bottleneck of the connection is generally located at the edge of the path connecting sender and receiver.

Focusing on online games, we can further support this assumption. As revenues for online game providers come from the subscription payments of many satisfied customers, every commercial online game system is generally supported by adequate resources in terms of connectivity speed, number, and computational capability of their servers [13]. Moreover, efficient game server architecture and synchronization can be put to good use to improve the performance on the wired part of the online game platform [11].

Yet, even when the game network platform is able to bring game events to the road's curb with delivery times within the GIT, problems may still arise at the last hop, which represents the bottleneck in terms of the available capacity for the connection. In fact, it may happen that the AP received packets at a higher rate than it can forward to destination. This is especially true in the case of vehicular networks; indeed, it can happen for several reasons related to this scenario such as, for instance, the fact that the wireless medium allows only one node at a time to transmit.

Moreover, interference, errors, fading, and mobility may cause packet losses which are handled by the MAC protocol through local retransmissions. These local retransmissions hide error losses to the TCP and are useful to increment the reliability of the connection. Without them, the TCP would misinterpret error losses as congestion evidences, thus reducing its sending rate and decreasing its performance. On the other hand, retransmissions follow the well known back off mechanism by which an increasing amount of time is utilized to determine whether a packet has been lost, and hence retransmit it. The 802.11 MAC protocol performs up to seven retransmissions of short packets (i.e., RTS/CTS, ACKs) and four retransmissions of long packets (i.e., data packets) [14]. This means that subsequent packets have to wait in queue until the preceding one, or one of its retransmissions, finally reaches the receiver and the corresponding ACK is successfully sent back.

Finally, especially in an open environment such as vehicular networks, it is very likely to have the same wireless connection shared by several devices and applications that increase the congestion level and cause queuing. As it is well known, TCP connections continuously probe the channel for more and more bandwidth until buffers along the path are fully utilized and overflowed. In presence of persistent TCP connections to support some elastic application, buffers at the bottleneck will probably be steadily fully utilized, thus queuing packets, slowing down their delivery time, and deteriorating the performance of real-time applications such as online games.

3. Smart Access Point With Low Advertised Window

In this section we describe our proposed solution to provide both high throughput to (TCP-based) elastic applications and to (UDP-based) real-time ones, even in a highly variable environment such as an infrastructure-based vehicular network.

3.1. TCP's sliding window algorithm

Since our proposed solution makes use of regular TCP functionalities, before presenting it, it is important to briefly summarize how the TCP algorithm for congestion and flow control works [4].

In essence, the rate at which TCP sends out its packet (i.e., the *sending rate*) is determined as the minimum between the *advertised window* and the current *congestion window*. The former is determined by the receiver node, it corresponds to an upper bound to the number of packets that could be handled by the receiver, and its value is communicated back to the sender through a specific field in ACK packets. Instead, the latter is continuously computed by the sender node. Upon any successful transmission demonstrated by returning ACKs, the congestion window is increased. Thereby, until a transmitted packet gets lost, the congestion window is steadily augmented, whereas when one or more packets are lost the congestion window is halved.

As a result, during TCP operations, packets are transmitted at a sending rate (or *sending window*) that steadily grows until the congestion window surpasses the advertised window, or until a packet is lost. Then, in the former case, the sending rate remains constant and corresponds to the advertised window; whereas, in the latter case, the sending window is halved before restarting its growth.

3.2. Limiting TCP's Advertised Window

To find the best solution to the tradeoff relationship existing between the TCP throughput and the real-time application delays, it is important to notice that when the transmission rate of a TCP sender surpasses the available bandwidth on the channel, the throughput cannot increase further; rather, the higher sending speed just generates packet queuing at the bottleneck that will eventually cause congestion losses and consequent halving of the TCP's sending rate.

Therefore, the TCP's sending rate should be kept high enough to efficiently utilize the available bandwidth but, at the same time, limited in its growth so as to not utilize buffers. This way, the throughput is maximized by the absence of packet losses that would halve the congestion window, while the delay is minimized by the absence of queues.

To determine an appropriate upper bound for the TCP's sending rate we should also consider other flows sharing the same bottleneck. Specifically, the aggregate bandwidth utilized by TCP flows on a given bottleneck at a given time t should not exceed the total capacity C of the bottleneck link diminished by the portion of the channel occupied by the concurrent real-time traffic, *UDPTraffic(t)*, i.e.,

$$AggregateTCP(t) \leq C - UDPTraffic(t). \tag{6}$$

Such aggregate bandwidth should be fairly shared by the TCP flows. Hence, the maximum sending rate for each TCP flow at time t, namely *TCPFlowRate(t)*, is represented by

$$TCPFlowRate(t) = \frac{AggregateTCP(t)}{\#TCPFlows(t)}. \tag{7}$$

where $\#TCPFlows(t)$ is the number of simultaneously present TCP flows at time t.

Having determined the appropriate upper bound for a TCP flow's sending rate, we need then to find a practical way to enforce it. This is a delicate issue as proposing a solution that required to modify Internet's core or the TCP code on all senders or receivers is clearly not a feasible option. For this reason we propose to limit the scope of intervention by exploiting existing features already present on regular TCP. In particular, the advertised window included in TCP's ACKs represents the perfect candidate to limit TCP's sending rate without having to modify the functioning of the whole Internet.

Since the actual sending window is determined as the minimum between the congestion window and the advertised window, the advertised window embodies a natural upper bound to the congestion window and is already implemented in all TCP versions. By appropriately modifying its value, we can achieve both efficiency and low delays.

With regular TCP, the advertised window is determined by the receiver; however, the receiver is not in the most suitable place for the modification that we need to perform. Indeed, to determine the appropriate value for the advertised window, a comprehensive knowledge about all flows that are transiting through the bottleneck (i.e., the last hop wireless link) is needed. Instead, the AP represents the node able to implement our scheme since all flows have to pass through it. By snooping the channel, the AP can also infer the number of active TCP connections and the aggregate amount of current UDP traffic; it can hence easily compute the value of $TCPFlowRate(t)$ as shown by (7). Therefore, even if the value of the advertised window in TCP's ACKs is already established by the receiver, these ACKs have to transit through the AP, which can hence modify on-the-fly the advertised window field with $TCPFlowRate(t)$.

In summary, the proposed solution is aimed at enabling an efficient coexistence among heterogeneous (i.e., TCP-based elastic and UDP-based real-time) flows, even in a highly variable environment such as vehicular networks, by limiting the advertised window of TCP flows through the deployment of smart APs. For this reason, we name it *Smart Access Point with Low Advertised Window* (SAP-LAW).

Finally, a great advantage of SAP-LAW is that it does not require to have all APs in the network endowed with its mechanism. In fact, APs implementing SAP-LAW can coexist side by side with regular ones; the former bringing advantages to their served flows, without affecting the latter. Therefore, not only requires SAP-LAW very little and feasible modifications to existing architecture/protocols, but it can also be gradually deployed, enormously facilitating its adoption.

4. Performance Evaluation

We test SAP-LAW through the NS-2 simulator in a urban-like vehicular scenario, collecting results for a vehicle transiting in the transmission range of different APs. The scenario is made more realistic by having other heterogeneous traffic sharing each considered AP. Results are collected both for the mobile node and for the other nodes that generate traffic in the network.

More in detail, we consider a portion of road comprising four APs (AP_0, AP_1, AP_2, AP_3), seven wired nodes ($W_0, W_1, \ldots, W_7$), and seven wireless ones ($N_0, N_1, \ldots, N_7$). Among the wireless nodes, we monitor N_0 while it is driving through the transmission ranges of the various APs. For the sake of a clearer description of the simulative configurations, Fig. 1 shows the network topology for the considered scenario: a block, or a group of blocks, with 1000 m between any two consecutive corners, around which a car (i.e., N_0) is traveling.

In the simulations, wired links have a 100 Mbps capacity, whereas wireless ones have a variable capacity (depending on channel interferences) of circa 19 Mbps. All wired links directly connecting two APs have less than 1 ms of propagation delay (they are only 1000 m far from each other), whereas all other wired links have a propagation delay of 20 ms. Buffers in the wired connections are set equal to 70 packets, which corresponds to the pipe size, i.e. the bandwidth-RTT product. The TCP's advertised window is initially set to a very high value, 550 packets, and remains constant when regular TCP New Reno and APs are employed, whereas SAP-LAW dynamically and continuously sets it by employing (7).

Since we are considering a vehicular scenario, we have modified the IEEE 802.11 module available for NS-2 to behave following IEEE 802.11p's specifications; coherently, this results in having about 750 m of transmission range [1]. MAC layer buffers on the APs are set equal to 100 packets as this is one of the most common values in off-the-shelf APs.

In the simulated scenario, wireless nodes continuously transmit and/or receive data through certain APs; the distance between these wireless nodes and their engaged APs is 100 m. The scenario also includes wireless nodes connected to the various APs so as to have a predetermined background traffic continuously utilizing these APs and allowing a clearer understanding of the outcomes. Specifically, the employed application flows are described in Table 1.

A mobile node, N_0, is traveling along the road passing by the coverage area of each of the APs with a speed of 14 m/s (about 50 Km/h, or 32 Mph); movement details of N_0 are reported in Table 2. When N_0 moves into the coverage area of a new AP, it connects with the new antenna and continues its operations through Mobile IP's packet redirection [15]. The mobile node N_0 runs different applications in different sets of simulations, specifically: a TCP-based FTP or a UDP-based online game. In the former case the

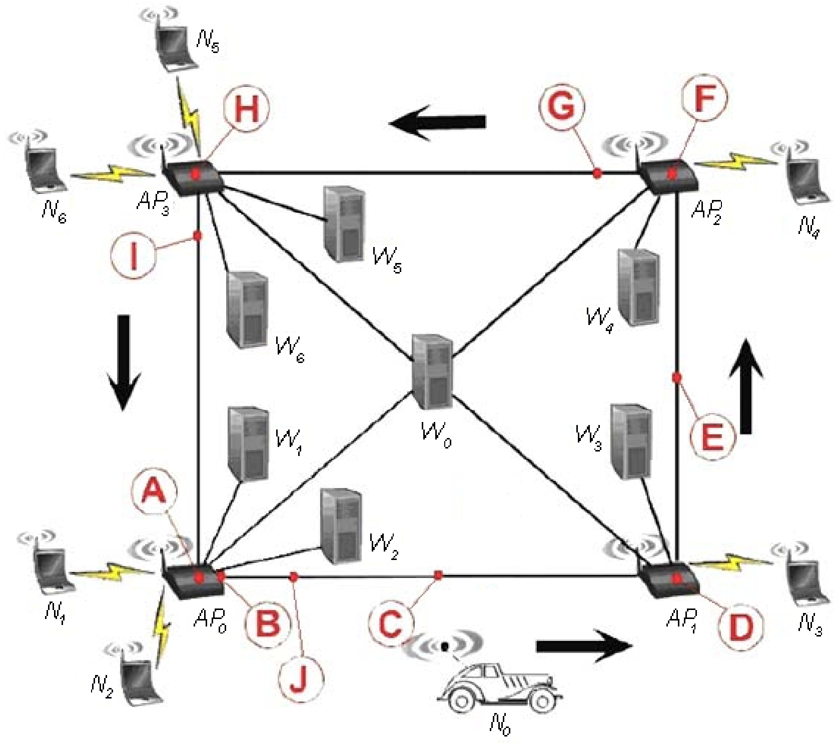

Figure 1: Simulated urban vehicular scenario.

Table 1: Simulated Flows

From	**To**	**Home Agent (AP)**	**Flow Type**	**Transport Protocol**
W_1	N_1	AP_0	FTP	TCP New Reno
N_2	W_2	AP_0	Online gaming	UDP
W_2	N_2	AP_0	Online gaming	UDP
N_3	W_3	AP_1	Online gaming	UDP
W_3	N_3	AP_1	Online gaming	UDP
W_4	N_4	AP_2	Video streaming	UDP
W_5	N_5	AP_3	FTP	TCP New Reno
W_6	N_6	AP_3	Video streaming	UDP

Table 2: Movement Details Of The Traveling Node

Time	**Location**	**Distance**	**AP**	**Description**
0.0 s	A	0 m	AP_0	Simulation start
3.0 s	B	42 m	AP_0	Data transmission start
36.8 s	C	515 m	AP_0	Start handoff $AP_0 - AP_1$
71.4 s	D	1000 m	AP_1	Min distance from AP_1
87.5 s	E	1505 m	AP_1	Start handoff $AP_1 - AP_2$
142.8 s	F	2000 m	AP_2	Min distance from AP_2
153.7 s	G	2152 m	AP_2	Start handoff $AP_2 - AP_3$
214.3 s	H	3000 m	AP_3	Min distance from AP_3
224.6 s	I	3144 m	AP_3	Start handoff $AP_3 - AP_0$
285.7 s	A	4000 m	AP_0	Min distance from AP_0
300.0 s	J	4200 m	AP_0	Simulation end

data flow is mostly unidirectional, from a server in the Internet to N_0 (plus ACKs on the returning path), whereas in the latter case the data flow is bidirectional; game events go from N_0 to a game server in the Internet and game updates go from the server to N_0.

The considered multimedia applications are simulated in a very realistic way. In fact, the video streaming corresponds to the real trace file of the movie *Star Wars IV* in high quality MPEG4 format [16]; frames of different sizes are hence sent with a 25 fps frequency. Online gaming traffic is inspired by real traces of the popular Counter Strike action game, and has i) a server-to-client flow characterized by an inter-departing time of game updates of 200 bytes every 50 ms and ii) a client-to-server flow characterized by an inter-departing time of game events of 42 bytes every 60 ms [5].

As for parameter C in (7), three different values are tested: 18, 19, and 20 (Mbps); however, if not differently stated, C is set equal to 19. Clearly, the C value has an impact only on SAP-LAW's performances, whereas it is not employed when regular protocols and APs are utilized. For the sake of clarity, in the next sections we identify the configuration employing regular protocols and APs with the name *TCP regular*.

5. Results

In this section we assess the performance improvement achieved by SAP-LAW in a vehicular scenario with infrastructure (see Fig. 1); the considered metrics are the final goodput achieved by elastic flows (FTP/TCP) and parameters discussed in Section 2.1 (jitter, DD, and qd) for real-time ones (gaming/UDP).

5.1. Elastic Flow Evaluation

First, we analyze the case with the mobile node N_0 driving around in the urban scenario downloading a file through FTP/TCP. In particular, Fig. 2 presents the sending window and the channel pipe size (computed as the RTT-bandwidth product) for a TCP regular (the chart above in the figure) and for SAP-LAW (the chart below in the figure). The well known saw-tooth shape is evident for TCP regular; those peaks generally corresponds to a packet loss due to congestion. Yet, before losing a packet, others were queued at the bottleneck buffer generating queuing delays that affected simultaneous real-time applications (as we demonstrate in Section 5.2). Instead, as evident in the lower chart of Fig. 2, when employing SAP LAW, the sending window is limited in its growth by the advertised window computed through (7), thus showing an almost-flat shape and never exceeding the pipe size. As a result, SAP-LAW avoids congestion, packet losses, and queuing at buffers.

Clearly, regardless of utilizing TCP regular or SAP-LAW, when N_0 moves from the coverage area of a given AP to a new one, a disconnection period occurs during which the sending window is just 1 packet. In particular, from left to right of Fig. 2, it is easy to notice when N_0 is connected to AP_0, AP_1, AP_2, AP_3, and finally AP_0 again. Furthermore, as evident by the lower chart of Fig. 2, when N_0 is in the transmission range of an AP serving no other FTP/TCP traffic (i.e., AP_1 or AP_2) the height of SAP-LAW's sending window is roughly twice with respect to the case when the engaged AP serves also another FTP/TCP flow (i.e., AP_0 or AP_3).

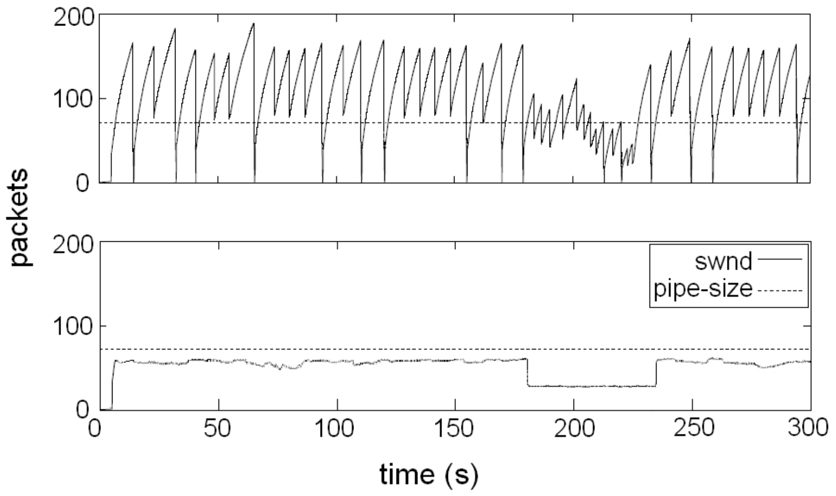

Figure 2: TCP's sending window and link's pipe size for i) TCP Regular (chart above) and ii) SAP-LAW with $C = 19$ (chart below). N_0 is moving passing by the various APs while downloading a file from W_0; when engaged with AP_0 and AP_3, N_0 shares the channel with another TCP flow.

Similarly, if we focus our attention on N_5, which is a static wireless node engaged in a FTP/TCP flow through AP_3, we should be able to detect the passage of N_0 through the coverage area of AP_3 by noticing a sudden decrease of N_5's sending window, followed by a sudden increase when N_0 leaves. This is evident in Fig. 3, which shows the sending window of N_5, when considering the TCP regular-case (upper chart in the figure) or when utilizing SAP-LAW (lower chart in the figure). In particular, the sending window computed through (7) is halved during the period of time (roughly, 180 s - 220 s) when the number of simultaneous FTP/TCP flows doubles, passing from one to two.

Lastly, Fig. 4 reports the average goodput achieved by N_0 and N_3, considering TCP regular or SAP-LAW with different values of C. This chart demonstrates that the goodput decrease due to the employment of SAP-LAW is negligible, especially if setting C equal to 19 or 20.

5.2. Real-Time Flow Evaluation

Analyzing the performance achieved by real-time applications, we focus now on parameters relative to the DD described by (5). To this aim, we compare TCP regular and SAP-LAW by considering two cases: i) N_0 moving around while downloading a file through a FTP/TCP session and ii) N_0 moving around while engaged in an online game session.

In the first case, we evaluate how N_2's online game session is affected by the continuous presence of a competing FTP/TCP flow (i.e., N_1's) under the coverage of the same AP (i.e., AP_0) and by the sudden arrival of another FTP/TCP flow (i.e., N_0's). To this aim, Fig. 5 shows the jitter of the DD as experienced by the game flow directed from server W_2 to the client N_2. As stated in Table 1, this game session has to share the same AP_0 with a FTP/TCP flow; this allows us to appreciate the different performances achieved by employing TCP regular (leftmost chart in the figure) or SAP-LAW (rightmost chart in the figure). The charts demonstrate that, when the game session competes with a FTP flow based on TCP regular, jitter values result consistently higher, also achieving peaks of $\sim$60 ms. Instead, with SAP-LAW, the jitter continuously stays under 12 ms. Only in one case this does not happen: when the mobile node N_0 enters into the coverage area of the same AP engaging N_2. Yet, this single high peak reaches $\sim$20 ms whereas with TCP regular we can witness the jitter systematically surpassing this value. Note that the entrance of N_0 in AP_0's coverage area is not so evident with TCP regular. This happens because the buffer size at any AP is constant and having it used by one or two FTP/TCP flows does not change the maximum queuing delay that it can generate.

In the second case, we consider the mobile node N_0 running an online game application engaged with the game server W_0, while passing by the various APs. To this aim, Fig. 6 shows the jitter experienced by the aforementioned game session when employing, from the leftmost chart to the rightmost one: a) TCP regular, b) SAP-LAW with $C = 18$, c) SAP-LAW with $C = 19$, and d) SAP-LAW with $C = 20$. As it is evident, SAP-LAW outperforms TCP regular with all C values. The lowest jitter is experienced when $C = 18$ is employed; whereas progressively increasing C causes a (little) raise of the jitter. This is due to the fact that the bandwidth oscillated on the wireless channel; yet, it was more often closer to 18 Mbps than to higher values (i.e., 19 Mbps or 20 Mbps).

As explained in Section 2.1, the qd is a fundamental component in packets' DD. Indeed, high variations in packets' DD generally corresponds to high qd values. This is confirmed also by Fig. 7 where we consider again the same simulative scenario of Fig. 6 and report the maximum qd value registered every 5 s by the game events. As it is evident, with TCP regular qd values are much higher than when employing SAP-LAW and its peaks correspond in simulation time with peaks visible in Fig. 6. In particular, with TCP regular qd reaches an overall maximum value of 79.21 ms, which represents a huge amount of time when trying to deliver game events in less than 150 ms from their generation. This demonstrates one more time how TCP regular would be affected by continuous gaming interactivity loss, whereas SAP-LAW is effective in maintaining a smooth gaming flow.

To conclude, we show in Fig. 8, the cumulative function of the DD jitter for the compared schemes. The difference among the various schemes is evident, yet we provide also a quantitative evaluation in Fig. 9. In the chart the height of the columns corresponds to the maximum DD jitter value associated to the 95 % and 99 % of the cumulative function. It is particularly interesting to observe that the 99 % of game messages delivered when SAP-LAW is employed experience very low delay jitter; whereas the same cannot be said for TCP regular.

6. Conclusions

Vehicular networks represent the next frontier in wireless communications. Through APs along the road, car passengers will be able to access the Internet and all their favorite online applications. We can hence expect to have several heterogeneous applications competing for the same wireless resources, suddenly appearing and disappearing as cars move.

Whereas it is well known that a mobile environment is deleterious for TCP-based elastic applications, with this paper we went beyond, showing also how the vehicular networking conditions are particularly harmful toward the emerging application of interactive online games and, in general, of real-time applications, especially when sharing the channel with elastic ones.

In this context, we have evaluated a solution, named SAP-LAW, based on the deployment of smart APs able to exploit regular features of existing transport protocols in order to improve the performance of both elastic and real-time applications. As a result of employing SAP-LAW, heterogeneous flows can efficiently coexist even in presence of frequent network traffic variations due to the vehicular scenario, ensuring at the same time the best possible throughput and per-packet delay. In particular, the latter embodies the main requirement for the successful deployment of interactive online games.

Finally, we plan to extend this work in several directions. We would like to test SAP-LAW in even more complex vehicular scenarios, also considering the benefits achievable by other popular applications that requires small per-packet delivery delay such as, for instance, video/audio streaming, interactive storytelling, and alert propagation [17, 18, 19].

7. Acknowledgments

We wish to express our sincere gratitude to Salvatore Frizzoli for his technical help in setting the experimental testbed.

References

[1] Dedicated Short Range Communications (DSRC) Home. [Online]. Available: http://www.leearmstrong.com/dsrc/dsrchomeset.htm

[2] *anonymous reference due to double blind review.*

[3] *anonymous reference due to double blind review.*

[4] V. Jacobson, "Congestion Avoidance and Control," in *Proc. of ACM SIGCOMM'88*, Stanford, CA, USA, pp. 314-329, Aug 1988.

[5] J. Färber, Traffic Modelling for Fast Action Network Games, Multimedia Tools and Applications, vol. 23, no. 1, 2004, 31-46.

[6] G. Armitage, "An Experimental Estimation of Latency Sensitivity in Multiplayer Quake 3," in *Proc. of ICON*, Sydney, Australia, pp. 137-141, Sep 2003.

[7] L. Pantel, L. C. Wolf, "On the Impact of Delay on Real-Time Multiplayer Games," in *Proc. of the 12th ACM NOSSDAV 2002*, Miami, FL, USA, pp. 23-29, May 2002.

[8] S. Zander, I. Leeder, G. Armitage, "Achieving Fairness in Multiplayer Network Games through Automated Latency Balancing," in *Proc. of ACM SIGCHI ACE2005*, Valencia, Spain, pp. 117-124, 2005.

[9] M. Mauve, J. Vogel, V. Hilt, W. Effelsberg, Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications, IEEE Transactions on Multimedia, vol. 6, no. 1, 2004, 47-57.

[10] *anonymous reference due to double blind review.*

[11] *anonymous reference due to double blind review.*

[12] H. Jiang, C. Dovrolis, “Why Is the Internet Traffic Bursty in Short (Sub-RTT) Time Scales?” in *Proc. of ACM SIGMETRICS 2005*, Banff, AL, Canada, 2005.

[13] Butterfly Grid Solution for Online Games [Online]. Available: http://www.butterfly.net

[14] IEEE, “Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (Phy) Specifications,” Specifications, ISO/IEC 8802-11:1999(E), 1999.

[15] *anonymous reference due to double blind review.*

[16] Movie Trace Files [Online]. Available: http://www-tkn.ee.tu-berlin.de/research/trace/ltvt.html

[17] L. Egidi, M. Furini, Bringing Multimedia Contents into MP3 Files, IEEE Communications Magazine, vol. 43, no. 5, May 2005, 90-97.

[18] P. Salomoni, S. Mirri, L. A. Muratori, “YEAST: The Design of a Cooperative Interactive Story Telling and Gamebooks Environment,” in *Proc. of the GAMEON’2007 International Conference, Eurosis*, Bologna, Italy, pp. 83-87, Nov 2007.

[19] E. Fasolo, A. Zanella, M. Zorzi “An Effective Broadcast Scheme for Alert Message Propagation in Vehicular Ad Hoc Networks,” in *Proc. of IEEE ICC 2006*, Instanbul, Turkey, pp. 3960-3965, Jun 2006.

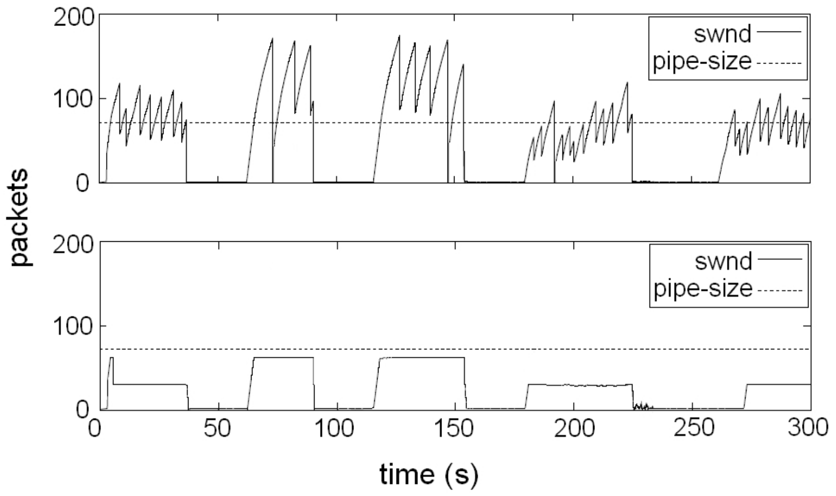

Figure 3: TCP's sending window and link's pipe size for i) TCP Regular (chart above) and ii) SAP-LAW with $C = 19$ (chart below). N_5 is downloading a file from W_5 through AP_3 and shares the channel with another TCP flow (between the mobile node N_0 and W_0) from 178.7 s to 234.6 s.

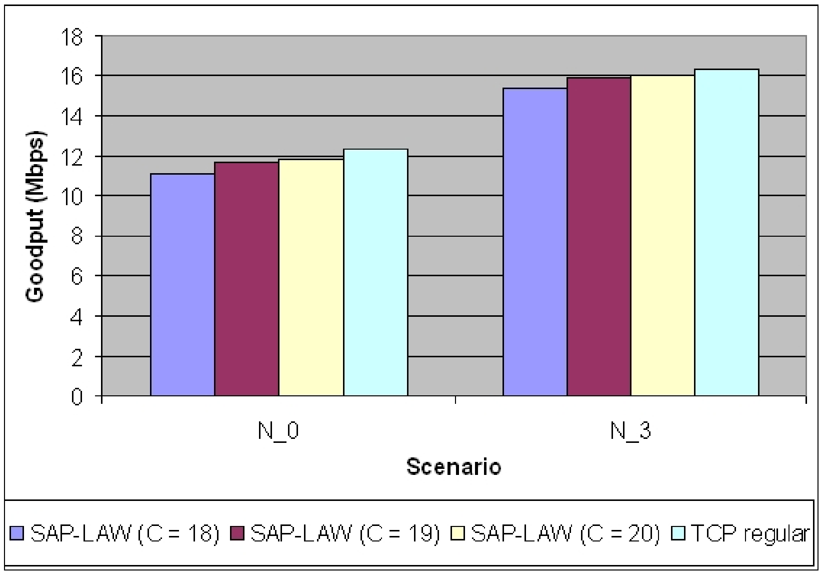

Figure 4: Average goodput achieved by TCP flows of N0 and N3, respectively, when employing alternatively TCP regular, SAP-LAW with $C = 18$, SAP-LAW with $C = 19$, and SAP-LAW with $C = 20$.

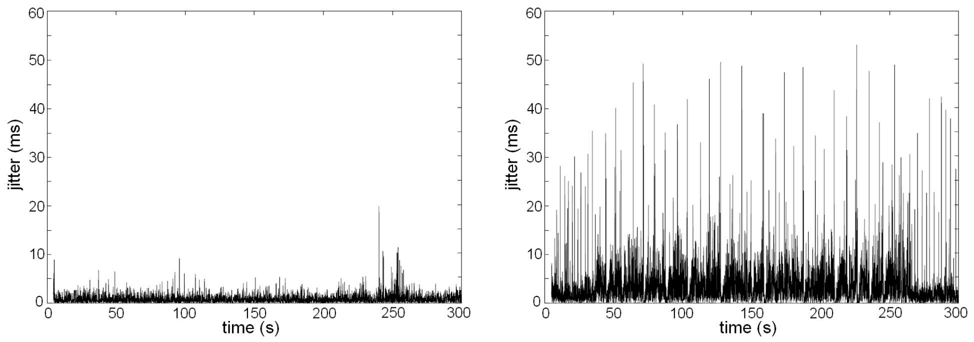

Figure 5: Delivery delay (DD) jitter experienced by the game flow going from server W_2 to client N_2 when employing TCP regular (leftmost chart in the figure) or SAP-LAW (rightmost chart in the figure); the game session shares AP_0 with another FTP/TCP flow downloading a file from W_1 to N_1.

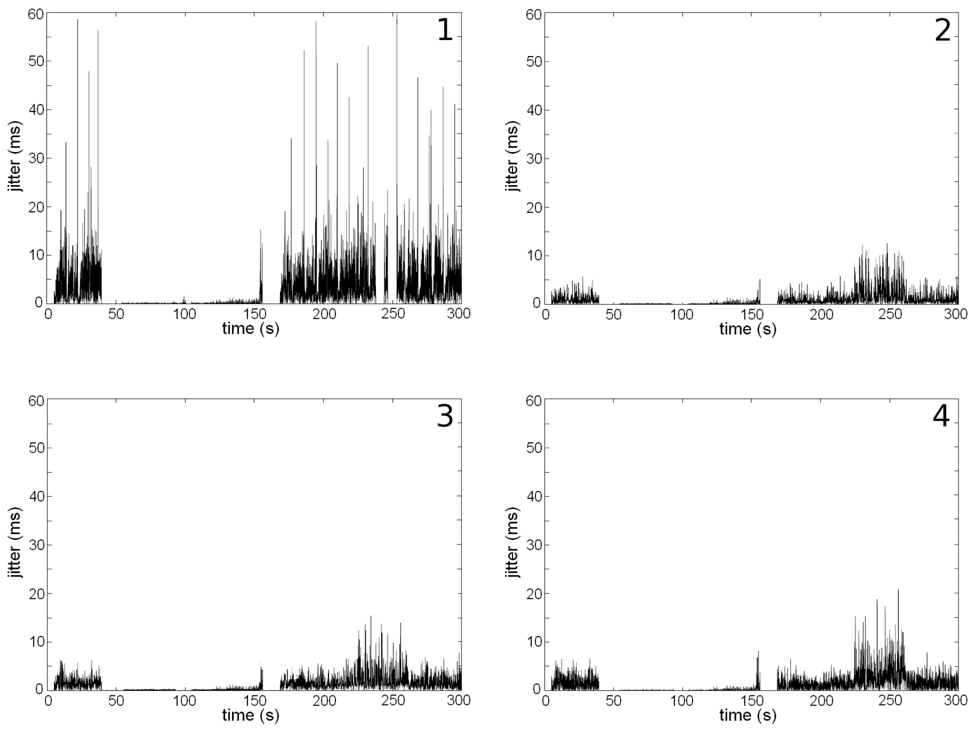

Figure 6: Delivery delay (DD) jitter experienced by the game flow going from server W_0 to client N_0; concurrent TCP flows (when N_0 is engaged with AP_0 and AP_3) employ alternatively TCP regular (chart 1), SAP-LAW with $C = 18$ (chart 2), SAP-LAW with $C = 19$ (chart 3), and SAP-LAW with $C = 20$ (chart 4).

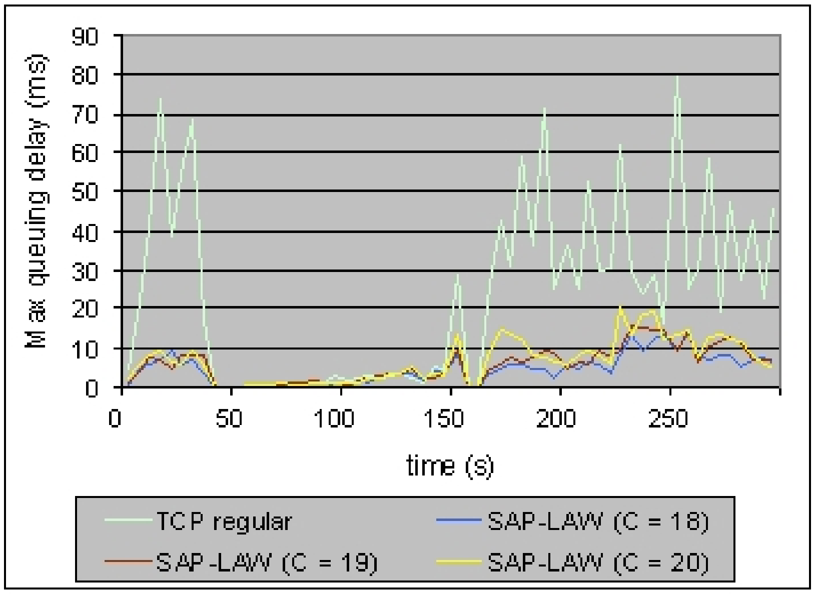

Figure 7: Maximum queuing delay (qd) values registered every 5 s as experienced by game events going from server W_0 to client N_0; concurrent TCP flows (when N_0 is engaged with AP_0 and AP_3) employ alternatively TCP regular, SAP-LAW with $C = 18$, SAP-LAW with $C = 19$, and SAP-LAW with $C = 20$.

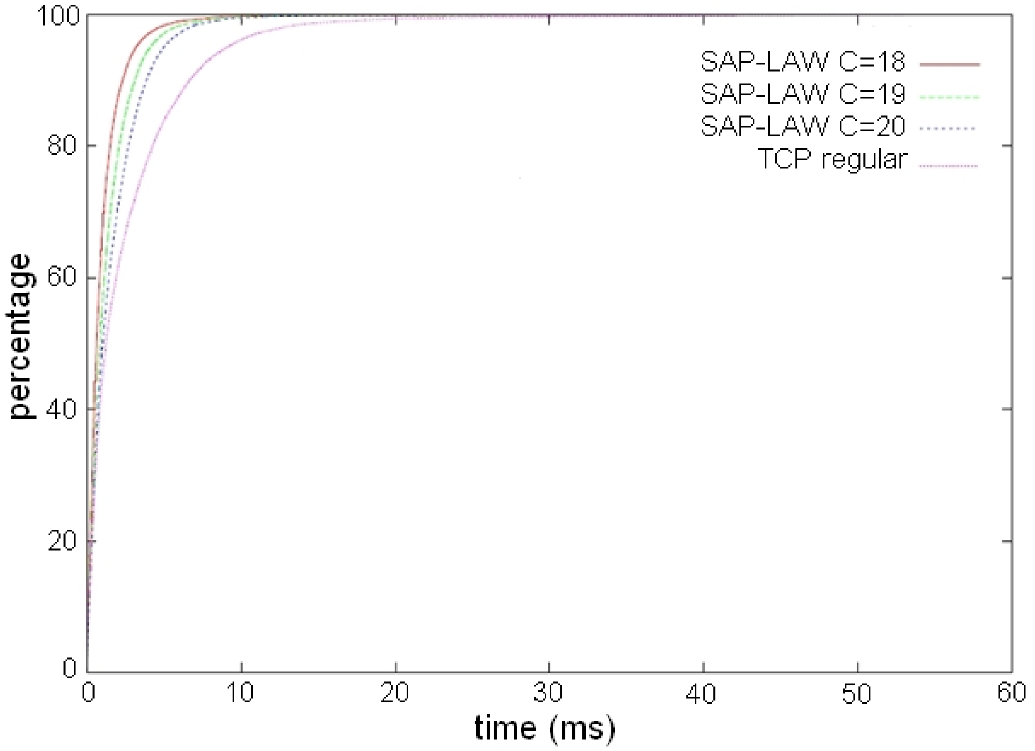

Figure 8: Cumulative function of the delivery delay (DD) jitter experienced by the game flow going from server W_0 to client N_0; concurrent TCP flows (when N_0 is engaged with AP_0 and AP_3) employ alternatively TCP regular, SAP-LAW with $C = 18$, SAP-LAW with $C = 19$, and SAP-LAW with $C = 20$.

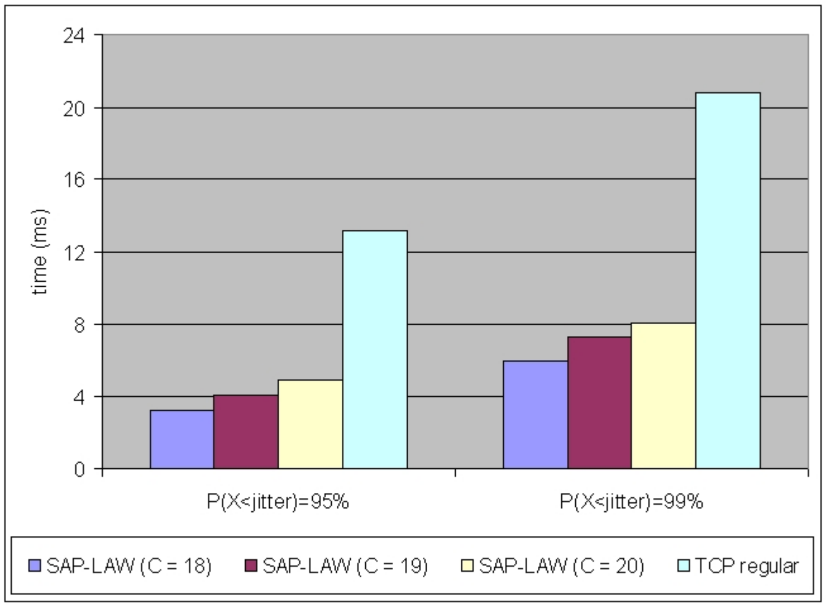

Figure 9: Delivery delay (DD) jitter value associated with the 95% and 99% of the cumulative function; game flow going from server W_0 to client N_0; concurrent TCP flows (when N_0 is engaged with AP_0 and AP_3) employ alternatively TCP regular, SAP-LAW with $C = 18$, SAP-LAW with $C = 19$, and SAP-LAW with $C = 20$.