Analysis of ECN/RED and SAP-LAW with simultaneous TCP and UDP traffic

Armir Bujari[a], Andrea Marin[b], Claudio E. Palazzi[a], Sabina Rossi[b]

[a] *Università degli Studi di Padova, Italy*
[b] *Università Ca' Foscari Venezia, Italy*

Abstract

Internetworking often requires a large amount of users to share a common gateway to obtain connectivity to the Internet. Congestion avoidance mechanisms are used to prevent the saturation of the gateway which represents a bottleneck of the system. The most popular congestion avoidance mechanisms are the Explicit Congestion Notification (ECN) and the Random Early Detection (RED). Recently, a new method for the congestion avoidance has been proposed: the Smart Access Point with Limited Advertised Window (SAP-LAW). The main idea is to hijack the acknowledge packets in the TCP connections in order to artificially reduce the advertised destination window according to some bandwidth allocation policy. Therefore, the flux control mechanism is artificially exploited to control the congestion at the bottleneck. The advantage of this approach is that it does not require any modification in the TCP implementations at the clients. In this paper, we propose stochastic models for the ECN/RED and SAP-LAW mechanisms in order

Email addresses: abujari@math.unipd.it (Armir Bujari), marin@dais.unive.it (Andrea Marin), cpalazzi@math.unipd.it (Claudio E. Palazzi), srossi@dais.unive.it (Sabina Rossi)

to compare their performances under different scenarios. The models are studied in mean field regime, i.e., under a great number of TCP connections and UDP based transmissions. Augmenting previous work on ECN/RED, we consider the presence of UDP traffic with bursts, and short lived TCP connections. The models for SAP-LAW are totally new. The comparison is performed in terms of different performance indices including average queue length, system throughput and expected queuing time.
Keywords: ECN/RED; SAP-LAW; queueing theory; stochastic processes; congestion control; transport protocols

---

1. Introduction

The explosive growth of computer networks has made the analysis of protocols designed to share one gateway among many devices more and more important. Differently from the early usage of the Internet, now the requirements in terms of bandwidth and responsiveness of many applications is very high and it is often the case that if these requirements are not met then the application usability is not acceptable. A popular and remarkable example is given by online games whose requirements in terms of response time are very strict.

When several devices share the same gateway, we have to address the problem of allocating the resources, i.e., the bandwidth. The mechanisms that are implemented to this end and that we consider in this paper are based on the transport protocols TCP and UDP. As it is well known [1], TCP is used mainly for reliable data transfer (including web browsing), and UDP in multimedia traffic or for applications with stringent requirements

on responsiveness such as online gaming (obviously there are exceptions). The reason why real time applications usually prefer UDP, is that it is a connectionless protocol, and it does not manage the retransmission of lost packets, guaranteeing minimum delays in data arrivals. On the contrary, TCP is a connection-oriented transport protocol: it provides reliable delivery and high throughput. For this reason it is particularly suited for applications whose predominant activity is the downloading or the uploading of content, as their main requirement is the complete and correct arrival of data.

TCP and UDP do not have an intrinsic mechanism to regulate the bandwidth when a gateway is shared, but the TCP (in its various versions) admits congestion and flux controls, while the UDP does not implement any of them. Every TCP connection keeps track of two window sizes: its own sending window and the destination window. Congestion control is obtained by allowing the sender window size to increase when packets are properly sent and to decrease when the resources are not enough to satisfy the bandwidth request. Flux control is obtained by choosing a sending rate which is given by the minimum between the sender window size and the destination window size.

Popular approaches that use the built-in congestion mechanism to regulate the TCP transmission rates across a gateway are the Random Early Detection (RED) [2, 3] and the Explicit Congestion Notification (ECN) [4, 5] while the Smart Access Point with Limited Advertised Window (SAP-LAW) [6, 7] is based on a flux control mechanism.

The SAP-LAW has been introduced in the context of integrated home server with shared access point. Indeed, due to the rapid evolution of commercial mobile devices, as well as the development of more and more ad-

vanced home entertainment systems, ensuring good performances to the applications sharing the same bottleneck has become a hard challenge [8].

In this paper, we study what happens when TCP and UDP are sharing the same link whose congestion is regulated by one of the mechanisms mentioned above: the ECN/RED and the SAP-LAW. On one side, the UDP-based flows seem to be aggressive towards the TCP-based ones since UDP does not provide any flux or congestion control mechanism. However, it has been shown that the TCP's congestion control mechanism causes serious problems to the critical deterioration of UDP-based applications [6, 7].

Even if a larger and larger bandwidth is offered today to the user, the coexistence of TCP and UDP continues to present problems: due to its congestion control functionality, TCP continuously probes for higher data rates, queuing packets on the buffer associated with the bottleneck of the connection and, consequently, causing delays for real-time applications. In literature we can find different solutions proposed to overcome the problem, and, in particular, different TCP variants have been proposed, since the UDP is simpler and there are not many variable aspects [9, 10, 11, 12, 13]. A large number of papers have studied the performance of TCP and ECN, see [14, 15, 16, 17, 18, 19] as a not exhaustive list, but the main focus is on the behaviour of a single TCP connection under various scenarios. More similar to the analysis carried out here is that proposed in [20] where the authors consider the overall system performance in the mean field approximation for ECN/RED. The theoretical results are then extended in [21]. Starting from these two papers we extend the considered model in several directions, as discussed later on, and we propose new ones for SAP-LAW. The mean field

theory [21, 22, 23] is an important approach to study large stochastic models with a deterministic approximation thus overcoming the well-known problem of the state space explosion.

From the modelling point of view, the contributions of this paper are:

- The extension of the mean field model presented in [20] in order to encompass the presence of UDP traffic.

- The introduction of a totally new model to analyse the SAP-LAW congestion mechanism. This model is completely different from those previously known in the literature such as [20, 21] since, according to SAP-LAW, each TCP connections does not necessary use its full congestion window size for each time slot. Therefore, a more detailed state representation is required that poses some computational challenges that we address by resorting to a state space aggregation.

- Another novelty is the relaxation of the common assumption [6, 20, 21] that TCP connections are greedy, i.e., they always have packets to send so that the sender window is always completely used at each time slot. We will study models that allow TCP connections to send a geometrically distributed random number of packets and then finish their life-cycle. We show that this extension is substantial for comparing the ECN/RED and SAP-LAW algorithms since the latter may allocate some bandwidth to TCP connections that are still in the slow start phase hence cause a waste of bandwidth. We show that our model is able to estimate the throughput reduction given a certain level of greediness of the TCP connections for the SAP-LAW.

Finally, we use these models to compare the performances of ECN/RED and SAP-LAW in the presence of UDP traffic. We consider both the cases of bursty and smooth UDP traffic, and the cases of greedy or non-greedy TCP connections.

The paper is structured as follows. Section 2 explains the ECN/RED and the SAP-LAW congestion mechanisms and Section 3 gives some background about the mean field theory. In Section 4 we introduce our models and give the iterative schemes for the computation of the performance indices. Section 5 briefly discusses the validation of the models and in Section 6 we compare the performance indices of the two congestion mechanisms under different scenarios. Finally, Section 7 concludes the paper.

2. ENC/RED and SAP-LAW congestion control mechanisms

In this section we review the two congestion control mechanisms that we study in this paper: the ECN/RED and the SAP-LAW.

2.1. ECN/RED

ECN [4] is an extension of TCP that avoids packet drops due to channel congestion. This scheme is used in conjunction with RED (Random Early Detection) gateways, marking the packets instead of dropping them, when the queue size is inside a given interval. In particular, RED fixes a minimum and a maximum queue threshold: when the number of packets in the queue is less then the minimum, the packets are not marked, when it is between the minimum and the maximum threshold, it marks the packets with a probability that depends on the queue length. Finally, if the number of the packets is larger than the maximum, all arriving packets are marked or dropped. In

this paper we approximate this behaviour by using an exponential law to decide the packet marking probability [24]. The weak point of this strategy is that it does not take into consideration the different requirements of real time applications, for whom packet arrival delays may have catastrophic effects on the usability (e.g., for video streaming or online gaming).

2.2. SAP-LAW

SAP-LAW is a solution proposed in [6] with the specific aim of reducing the problems caused by the simultaneous presence of both TCP-based and UDP-based applications. The basic idea is to find a trade-off between throughput and time delays by working on a dynamic modification of the TCP flow sending rate, in order to avoid the over utilization of the buffer while at the same time guaranteeing a full utilization of the bandwidth. The key factor is to determine the upper bound for the TCP flow sending rate as a function of the amount of UDP traffic. In this way, we are always sure to reserve enough bandwidth to the real time applications activity, without limiting too much the TCP flows. The general formula proposed in [6] is:

$$maxTCP_{\text{traffic}}(t) = \frac{(C^* - UDP_{\text{traffic}}(t))}{\#TCP_{\text{flows}}(t)},$$

where $UDP_{\text{traffic}}(t)$ is the amount of bandwidth occupied by the real time applications at time t, C^* is the capacity of the bottleneck link, and $\#TCP_{\text{flows}}(t)$ is the number of TCP connections at time t. It is noteworthy to point out, that the formula above computes the maximum amount of bandwidth share that can be allocated to a TCP connection while the resulting amount of data is computed by multiplying the computed bandwidth share with the

flows current round trip time (RTT) measurement [25, 26, 27]. Work [25] proposes an extension to SAP-LAW, addressing the RTT fairness problem amongst TCP flows. We do not contemplate for this feature in the modelling for SAP-LAW and leave it as a potential future work.

3. Theoretical background

The results presented in this paper are based on models for interacting objects, with individual states and global memory, which are defined in the spirit of the one proposed by Le Boudec et al. in [21].

According to [21], we consider a heterogeneous population of N objects whose states have the form (c, i), where $c \in \Gamma$ is the class of the object, and $i \in \epsilon_c = \{1, ..., S_c\}$ is its internal state. At each time epoch the objects interact with the environment and they can undergo a state transition (in this paper we do not consider the possibility that an object changes its class). $X_n^N(t)$ indicates the state of the n-th object at time slot t, while $K^N_{(c,i),(c,j)}(\vec{r})$ describes the probability that an object of class c goes from state i to state j depending on a certain quantity $\vec{r}$ that we describe hereafter. The object state transitions depend on the current state of the objects and on the memory $\vec{R}^N(t) \in \mathbb{R}^P$ where P is the dimension of the vector. $\vec{M}^N(t)$ is the *occupancy measure*, i.e., the row vector where each component $M^N_{c,i}(t)$ is the proportion of objects that are in state (c, i) at time t, i.e.,

$$M^N_{c,i}(t) = \frac{1}{N} \sum\nolimits_{n=1}^{N} 1_{\{X_n^N(t)=(c,i)\}} ,$$

where $1_{x=(c,i)} = 1$ when $x = (c, i)$ and 0 otherwise. If we want to consider the single classes of objects, the proportion of class c objects in state i at time

t is $\frac{1}{p_c} M^N_{c,i}(t)$, where $p_c = \sum_{i=1}^{S_c} M_{c,i}(t)$ at some time slot t. Function g is a deterministic continuous function $g : \mathbb{R}^P \times \mathbb{P}(\varepsilon) \mapsto \mathbb{R}$ such that, for $t \geq 0$:

$$\vec{R}^N(t+1) = g(\vec{R}^N(t), \vec{M}^N(t+1)), \tag{1}$$

where $\mathbb{P}(\varepsilon)$ is the set of vectors whose size is the number of possible states for one object $S = \sum_{c \in \Gamma} S_c$ with non negative components summing to 1. The role of function g is that of governing the evolution of the model's memory. Hence, $\forall n$ we have:

$$\mathbb{P}\{X^N_n(t+1) = (c,j) \mid \vec{R}^N(t) = \vec{r}, X^N_n(t) = (c,i)\} = K^N_{(c,i),(c,j)}(\vec{r}), \tag{2}$$

For each class c, the $S_c \times S_c$ matrix $K^N_c(\vec{r})$ is the transition matrix for an individual object of class c, and may depend on the population N and on the value $\vec{r}$ of the global memory. $K^N_{(c,i),(c,j)}(\vec{r})$ is the transition probability from state i to state j, therefore we assume $K^N_{(c,i),(c,j)}(\vec{r}) \geq 0$ and $\sum_{j=1}^{S_c} K^N_{(c,i),(c,j)}(\vec{r}) = 1$ for all $1 \leq i \leq S_c$. We assume that:

Assumption 3.1. $\forall c \in \Gamma, \forall i,j \in \varepsilon_c$, for $N \to \infty$, $K^N_{(c,i),(c,j)}(\vec{r})$ converges uniformly in $\vec{r}$ to some $K_{(c,i),(c,j)}(\vec{r})$, which is a continuous function of $\vec{r}$.

A sufficient condition for this assumption to be satisfied is that the transition matrix $K^N(\vec{r})$ does not depend on the number of objects N and that it is continuous on $\vec{r}$, which is always the case in our models.

Theorem 3.1 has been introduced and proved in [21]. Informally, it states that, as $N \to \infty$, $\vec{M}^N(t)$ and $\vec{R}^N(t)$ can be replaced by deterministic approximations if the initial state is known. The proof of the correctness of the

models that we propose in this paper widely depends on this theorem.

Theorem 3.1. Assume that the initial occupancy measure $\vec{M}^N(0)$ and memory $\vec{R}^N(0)$ converge almost surely to deterministic limits $\vec{\mu}(0)$ and $\vec{\rho}(0)$ and for $t \geq 0$:

$$\vec{\mu}(t+1) = \vec{\mu}(t)K(\vec{\rho}(t)) \quad (3)$$

$$\vec{\rho}(t+1) = g(\vec{\rho}(t), \vec{\mu}(t+1)) \quad (4)$$

Then for any fixed time t, almost surely:

$$\lim_{N\to\infty} \vec{M}^N(t) = \vec{\mu}(t) \text{ and } \lim_{N\to\infty} \overline{R}^N(t) = \vec{\rho}(t)\,.$$

4. Models for ECN/RED and SAP-LAW congestion mechanisms

In this section we propose different models for the ECN/RED and SAP-LAW congestion control mechanisms. Differently from other approaches previously addressed by the literature, we consider:

- the presence of UDP traffic as well as TCP. Specifically, we will be able to model both the case in which UDP traffic is smooth and the case in which it is bursty;
- the case in which the TCP connections are not greedy, i.e., in case of an infinite capacity gateway, their windows would not tend to be at their maximum size. In practice this is a frequent scenario which approximates the situation in which TCP connections are closed and new ones are newly opened maintaining the total number of active connections approximatively constant.

4.1. Modelling ECN/RED with UDP traffic and greedy TCP

In this section we propose a mean field model for TCP and UDP connections sharing the same gateway. We assume that TCP connections are greedy, i.e., the process associated with the connection fills the entire window with data to be sent at each time slot. We assume the time slot to be a round trip time as in [20, 21]. As a consequence, the unit of measure that we consider is normalised for one RTT, e.g., the window sizes are measured in packets for RTT and the bandwidth in number of packets which can be sent in one RTT.

According to the methodology described in Section 3, we annotate each object with a class $c \in \{p, u\}$ where p denotes the TCP connections and u the UDP agents. Analogously to [21] each TCP connection has a state $i \in \mathcal{S}_p$, where $\mathcal{S}_p = \{1, \ldots, I_p\}$ and I_p is the maximum number of states for a TCP connection. At state i, the sending rate is $s_p(i) \in \mathbb{N}$ and, without loss of generality we assume $i' > i'' \implies s_p(i') > s_p(i'')$. Although UDP transmissions are stateless, we model them by using a probabilistic automaton whose transition probabilities are independent of the rest of the model (in particular of the queue length). This choice allows us to use the automata to model different behaviours of the transmission of a UDP agent, e.g., with smooth or bursty traffic. In practice, the transition probabilities may be decided with a model-fitting approach such as those proposed in [28, 29]. The set of states associated with a process using a UDP transmission is denoted by $\mathcal{S}_u = \{1, \ldots, I_u\}$, where I_u is the maximum number of states modelling a UDP agent, and the sending rate at state $i \in \mathcal{S}_u$ is $s_u(i)$. Notice that the flexibility of this modelling choice is very high: it allows us to represent

synchronised or partially synchronised UDP transmissions (such as those occurring during online gaming), the autocorrelation and the periodicity in the transmission bursts, as well as a simple smooth transmission of packets. Thus the state of an object is denoted by a pair (c, i) where $c \in \{p, u\}$ and $i \in \{1, \ldots, I_p\}$ if $c = p$ and $i \in \{1, \ldots, I_u\}$ otherwise. To avoid trivialities we assume $I_p > 1$ and $I_u \geq 1$.

Let $N \in \mathbb{N}$ be the number of objects and $C \in \mathbb{R}$ the bottleneck capacity for every connection. Since we do not admit class switching during the system activity (TCP connections can not turn into UDP connections, and vice-versa) we define N_p as the number of TCP connections and N_u the number of UDP transmissions, i.e., for all $t \geq 0$:

$$N_p = \sum_{n=1}^{N} 1_{X_n^N(t)=(p,i)} \quad \text{for some } i \in \mathcal{S}_p$$
$$N_u = \sum_{n=1}^{N} 1_{X_n^N(t)=(u,i)} \quad \text{for some } i \in \mathcal{S}_u$$

where $\vec{X}^N(t)$ is the stochastic process underlying the model and $X_n^N(t)$ is the state of object n at time t. Clearly, $N = N_p + N_u$. Let $p_{\mathtt{p}}$ ($p_{\mathtt{u}}$ respectively) be the proportion of TCP (respectively UDP) connections, then we have:

$$p_p = \frac{N_p}{N} = \sum_{i=1}^{I_p} M_{p,i}^N(t) \quad \forall t \geq 0$$
$$p_u = \frac{N_u}{N} = \sum_{i=1}^{I_u} M_{u,i}^N(t) \quad \forall t \geq 0\,,$$

where $M_{c,i}^N(t)$ is the occupancy vector defined in Section 3. The proportion

of TCP connections in state i at time t is then $p_p^{-1} M_{p,i}^N(t)$ and for UDP we have $p_u^{-1} M_{u,i}^N(t)$. The average normalised sending rate of TCP connections at time t is hence $S_p^N(t) = \sum_i^{I_p} s_p(i) M_{p,i}^N(t)$, while for UDP connection it is $S_u^N(t) = \sum_i^{I_u} s_u(i) M_{u,i}^N(t)$.

We now define the memory of the system, $\vec{R}^N(t) = (R_c^N(t), R_e^N(t))$, that consists of a pair of real numbers that represents the normalised population of packets enqueued at the bottleneck at the end of time slot t and $t-1$, respectively. The update equation is:

$$\begin{aligned} R_c^N(t+1) &= \max\left(R_c^N(t) + S_p^N(t+1) + S_u^N(t+1) - C, 0\right) \\ R_e^N(t+1) &= R_c^N(t)\,. \end{aligned} \tag{5}$$

We approximate the behaviour of the ECN/RED mechanisms as proposed in [20]: as soon as a packet arrives at the bottleneck it is marked for being discarded or sent. The marking of each packet is independent of the others arriving during the same time slot, and cannot be changed. The marking of packets at time slot t depends only on the queue length seen by the packets immediately before their arrival, i.e., on the memory at time t via function $q(\vec{R}^N(t))$. According to [2, 4, 24] we define function q as follows:

$$q(R_e^N(t)) = 1 - \exp(-\gamma R_e^N(t))\,, \tag{6}$$

i.e., the probability of discarding a packet grows exponentially with the total queue length. Observe that the marking policy is coherent with that proposed in [20] where the packets are marked according to the queue length that they find at their arrival epoch, while in [21] the queue length after their departure

is used. Parameter γ is a positive real number whose value will be specified in the following sections. Finally, the transition matrix for each object is:

$$\begin{aligned} K_{(p,i),(p,i+1)}(\vec{r}) &= (1-q(r_p))^{s_p(i)} 1_{i<I_p} \\ K_{(p,I_p),(p,I_p)}(\vec{r}) &= (1-q(r_p))^{s_p(I_p)} \\ K_{(p,i),(p,d(i))}(\vec{r}) &= 1-(1-q(r_p))^{s_p(i)} \\ K_{(u,i),(u,j)}(\vec{r}) &= \kappa_{(u,i),(u,j)}, \qquad 1 \le i,j \le I_u \end{aligned} \tag{7}$$

where $\vec{r} = (r_c, r_p)$ is a value of $\vec{R}^N(t)$, $\kappa_{(u,i),(u,j)} \in [0,1]$ and for all $i \in [1, I_u]$:

$$\sum_{j=1}^{I_u} \kappa_{(u,i),(u,j)} = 1\,.$$

Function $d(i)$ is used to model the destination state in case of a marked packet. We may have $d(i) = 1$ for all i or as in [20] $d(i) = \lfloor i/2 \rfloor$ (and the number of packets sent $s_p(i)$ is proportional to i). Generally speaking, we require that $d(i) = i$ only for $i = 1$, while for $i > 1$ we have that $d(i) < i$.

Remark 4.1. The model we propose for ECN/RED is novel with respect to the one presented in [20] since it takes into account different classes of traffic, not only TCP. In [21] the authors propose a solution for different types of TCP connection. However, in their model the probability of a TCP connection to reduce its window size depends only on the queue length, i.e., is the same for connections with a large and already small windows. Differently, our approach takes into account that a TCP connection with a large window sends more packets than one with a small window and hence has higher probability to be selected for being slowed down.

$$
\begin{aligned}
\mu_{p,i}(t+1) &= (1-q(\rho_e(t)))^{s_p(i-1)}\mu_{p,i-1}(t)1_{\{i>1\}} \\
&+ \sum_{j:d(j)=i}\left(1-(1-q(\rho_e(t)))^{s_p(j)}\right)\mu_{p,j}(t) \quad 1 \le i < I_p \\
\mu_{p,I_p}(t+1) &= (1-q(\rho_p(t)))^{s_p(I_p-1)}\mu_{p,I_p-1}(t) + (1-q(\rho_p(t)))^{s_p(I_p)}\mu_{p,I_p}(t) \\
\mu_{u,i}(t+1) &= \sum_{j=1}^{I_u}\kappa_{(u,j),(u,i)}\mu_{u,j}(t) \\
\sigma_p(t+1) &= \sum_{i=1}^{I_p}\mu_{p,i}(t+1)s_p(i) \\
\sigma_u(t+1) &= \sum_{i=1}^{I_u}\mu_{u,i}(t+1)s_u(i) \\
\rho_c(t+1) &= \max\left(\rho_e(t)+\sigma_p(t+1)+\sigma_u(t+1)-C,0\right) \\
\rho_e(t+1) &= \rho_c(t)
\end{aligned}
$$

Table 1: Iteration system for ECN/RED model with greedy TCP connections.

Proposition 4.1. If $\vec{M}^N(0)$ and $\vec{R}^N(0)$ converge almost surely to $\vec{\mu}(0)$ and $\vec{\rho}(0)$, respectively, as $N \to \infty$ then for any finite horizon t we have that:

$$\lim_{N\to\infty}\vec{M}^N(t) = \vec{\mu}(t) \text{ and } \lim_{N\to\infty}\vec{R}^N(t) = \vec{\rho}(t)$$

almost surely, where $\vec{\mu}(t)$ and $\vec{\rho}(t)$ are defined by the iterative scheme depicted in Table 1.

Proof. The proof follows from the fact that Assumption 3.1 is satisfied and from Theorem 3.1. □

Table 1 should be read interpreting $\mu_{p,i}(t)$ and $\mu_{u,j}(t)$ as the proportion of TCP connections and UDP agents in state i and j, respectively. $\sigma_p(t)$ and

$\sigma_u(t)$ represent the normalised traffic generated by TCP and UDP, respectively. Finally, $\rho_c(t)$ and $\rho_e(t)$ are the functions describing the limits of $R_c^N(t)$ and $R_r^N(t)$. The right-hand side of the equations defining $\overline{\mu}$ describe the *fluxes* arriving into a certain state. Let us consider the first equation defining the proportion of TCP connections in state i at time $t+1$ for $1 \leq i < I_p$. The term $(1-q(\rho_e(t)))^{s_p(i-1)}\mu_{p,i-1}(t)1_{\{i>1\}}$ denotes the flux incoming from state $i-1$ which is given by the proportion of TCP connections in state $i-1$, i.e., $\mu_{p,i-1}(t)$ multiplied by the probability of not having any packet marked. Since $q(\rho_e(t))$ is the probability of marking a packet and from state $i-1$ the connection has sent $s_p(i-1)$ packets, then the probability of not having any marked packet is $(1-q(\rho_e(t)))^{s_p(i-1)}$. The second term of the right hand side of the first equation models the flow into state i due to a packet dropping. The sum is over all the TCP object states that sends the window state to i in case of packet marking and the flow is computed in a similar way of what we have just described for the first term. The second equation is similar to the first and model the behaviour of the last state of the TCP connections. It cannot be reached due to a packet dropping and in case of not having any marked packet it stays in I_p. The third equation models the state transitions for UDP agents and finally the remaining ones follow the rules for the memory update previously described in this section.

4.2. Modelling SAP-LAW with UDP traffic and greedy TCP

The Smart Access Point with Limited Advertised Window (SAP-LAW) is a solution to the problem of congestion at the bottleneck proposed in [6] with the specific aim of reducing the problems caused by the simultaneous presence of both TCP-based and UDP-based applications. The key aspect of this

scheme is the manipulation of the TCP acknowledgement packet forwarded by the gateway. The idea is to artificially reduce the value of the receiving window size of the destination in the acknowledgement packet. Basically, if the ECN/RED uses the TCP congestion detection and resolution algorithm to handle the bottleneck, the SAP-LAW uses the flux control algorithm that imposes the sender to use a window which is the minimum between its sending window and the destination's receiving window (see, e.g., [30]). According to [6] the gateway counts the arrived UDP packets in a time interval and hence estimates the total instantaneous UDP arrival rate at time t, $\xi(t)$. Let NC be the total capacity of the gateway, and $N_p(t)$ the number of TCP connections at time t, then the maximum window size which is sent back to the TCP transmitters is the same for all the TCP connections and equal to:

$$maxTCP(t) = \frac{NC - \xi(t)}{N_p(t)}\,. \tag{8}$$

In our setting, we assume $N_p(t) = N_p$, $N = N_p + N_u$ and the time interval that we use to estimate the instantaneous arrival rate for the UDP packets to be equal to a multiple Y of the round trip time, i.e., $\xi(t) = y(S_u^N(t-Y), \ldots S_u^N(t-1))$, where y is a function which estimates the bandwidth need of UDP traffic based on the latest $Y+1$ time slots. Function y is arbitrary although we expect to weight the needs of recent time slots more than the older ones and compute a mean. The notation, unless differently specified, is that of Section 4.1. The packets sent by each TCP connection at each time slot depend on its state and on Equation (8) and the granularity of the model. For the sake of simplicity, we assume that $s_p(i) = \alpha i$ for some $\alpha \in \mathbb{N}$, i.e., the maximum number of packets sent in a time slot

by a TCP connection grows linearly with the state numbering. The memory $\vec{R}^N(t) = (R_c^N(t), R_e^N(t), \vec{R}_u^N(t))$ is a pair of real numbers (r_c, r_e) followed by a vector of reals $\vec{r}_u$ where r_c denotes the normalised queue length at the bottleneck (counting both TCP and UDP packets) at a given time slot, r_e is the normalised queue length at the previous time slot and $\vec{r}_u = (r_{u0}, \ldots, r_{uY})$ denotes the normalised counting of the arrived UDP packets at the latest $(Y+1) > 1$ time slots.

The transitions for UDP objects are the same of those shown in Section 4.1. Each TCP connection has a state represented by a pair of numbers $(i, j) \in \mathbb{N}^2$, where i denotes the state corresponding to the sender window size $s_p(i)$, while j takes into account the number of received acknowledges. Indeed, the SAP-LAW mechanism does not allow the sender to use its full window size and consequently the growth of the sender window size is in general slower than what would be observed with a gateway with infinite capacity. In fact, this is due to the computation of the minimum between the congestion window size and the advertised window size. Formally, the state of each object in our model is $(\ell, \vec{x})$ where $\ell \in \{p, u\}$ (where p stands for TCP and u for UDP) and:

$$\vec{x} = \begin{cases} (i,j) : 1 \leq i \leq I_p \wedge 0 \leq j < i & \text{if } l = p \\ i : 1 \leq i \leq I_u & \text{if } l = u\,. \end{cases}$$

The dynamic of the memory is specified by the following equation:

$$\begin{aligned} R_c^N(t+1) &= \max\left(R_c^N(t) + S_p^N(t+1) + S_u^N(t+1) - C, 0\right), \\ R_e^N(t+1) &= R_c^N(t)\,, \\ R_{ui}^N(t+1) &= R_{u(i-1)}^N(t) \qquad\qquad 1 \le i \le Y \\ R_{u0}^N(t+1) &= S_u^N(t+1)\,. \end{aligned}$$

Function $S_u^N(t)$ is specified as shown in Section 4.1, while S_p^N is defined as:

$$S_p^N(t) = \sum_{i=1}^{I_p} \sum_{j=0}^{i-1} M_{p,i,j}(t) \cdot \min\left(s_p(i), s_p(h(\vec{R}_u^N(t)))\right)\,,$$

where

$$h(\vec{r}_u) \;=\; \arg\max_j \left(s_p(j), s_p(j) \;\le\; (C \;-\; \tilde{y}(\vec{r}_u))\frac{1}{p_p} \;\vee\; j \;=\; 1\right), \quad (9)$$

and $\tilde{y}(\vec{r}_u) = y(r_{u1}, \ldots, r_{uY})$. Given a state (i, j) of a TCP connection, and the state z corresponding to the maximum sending window obtained by Equation (9), the following state of the connection is $f(i, j, z)$ defined as follows:

$$f(i,j,z) \;=\; \begin{cases} (i, j + \min(z, i)) & \text{if } j + \min(z, i) < i \\ (i+1, j + \min(z, i) - i) & \text{if } j + \min(z, i) \ge i \wedge i < I_p \\ (I_p, I_p - 1) & \text{otherwise}\,. \end{cases} \quad (10)$$

We can now give the mean field approximation for $N \to \infty$.

Proposition 4.2. If $\vec{M}^N(0)$ and $R^N(0)$ converge almost surely to $\vec{\mu}(0)$ and

$$
\begin{aligned}
\mu_{p,i,j}(t+1) &= \sum_{i'=1}^{I_p}\sum_{j'=0}^{i-1} \mu_{p,i',j'}(t) 1_{f(i',j',h(\vec{\rho}_u(t)))=(i,j)} \\
\mu_{u,i}(t+1) &= \sum_{j=1}^{I_u} \kappa_{(u,j),(u,i)} \mu_{u,j}(t) \\
\sigma_p(t+1) &= \sum_{i=1}^{I_p}\sum_{j=1}^{i-1} \mu_{p,i,j}(t+1) \min(s_p(i), s_p(h(\vec{\rho}_u(t)))) \\
\sigma_u(t+1) &= \sum_{i=1}^{I_u} \mu_{u,i}(t+1) s_u(i) \\
\rho_c(t+1) &= \max\Big(\rho_c(t) + \sigma_p(t+1) + \sigma_u(t+1) - C, 0\Big) \\
\rho_p(t+1) &= \rho_c(t) \\
\rho_{ui}(t+1) &= \rho_{u(i-1)}(t), 1 \le i \le Y \\
\rho_{u0}(t+1) &= \sigma_u(t+1)
\end{aligned}
$$

Table 2: Iteration system for SAP-LAW model with greedy TCP connections.

$\vec{\rho}(0)$, respectively, as $N \to \infty$ then for any finite horizon t we have that:

$$\lim_{N\to\infty} \vec{M}^N(t) = \vec{\mu}(t) \text{ and } \lim_{N\to\infty} \vec{R}^N(t) = \vec{\rho}(t)$$

almost surely, where $\vec{\mu}(t)$ and $\vec{\rho}(t)$ are defined by the iterative scheme depicted in Table 2.

PROOF. The proof follows straightforwardly from Theorem 3.1 after verifying Assumption 3.1. □

4.3. Modelling ECN/RED with UDP and temporary TCP connections

When comparing the SAP-LAW and the ECN/RED congestion avoidance mechanisms we should take into account that the former tends to work

better when the TCPs sending windows are very large. Intuitively, if all the TCP connections are willing to send an infinite queue of packets then the SAP-LAW allows them to constantly increase their sending window (possibly slower than what happens with ECN/RED). In the long run, all the windows will be at their maximum size and the SAP-LAW decides which actual rate they can send according to the estimated UDP traffic. However, the SAP-LAW is fair with respect to the distribution of the bandwidth to each TCP connection as shown by Equation (8), therefore it may happen that part of the bandwidth is allocated to a TCP connection whose sending window is still small causing a waste of the resources.

In this and in the following section, we modify the previous models in order to take into account that a TCP connection has a random amount of packets to send. Once they are sent, we assume that the connection is closed and a new one restarts from the minimum window size.

We assume that once the connection is established it has to send an amount of packets which is geometrically distributed with parameter $w \in (0,1)$, i.e, the probability of sending exactly ξ packets is $w(1-w)^{\xi-1}$, with an expected value of $1/w$. We modify the model presented in Section 4.1 in order to take into account this aspect as follows (when not differently specified, the notation is inherited by the previous sections). Let $L_f(t)$ for $1 \leq f \leq N_p$ and $t \geq 0$ be a set of independent and identically distributed geometric random variables with parameter $w \in (0,1)$ that represent the number of packets that TCP connection i has still to send at time t. The event of closure of TCP connection f occurs at time t_0 if the number of packets that it sends at t_0 is equal to $L_f(t_0)$. Notice that due to the memoryless property of the

geometric random variable, the distribution of the number of packets sent by a TCP connection from its opening to its closing epoch is still geometric.

The state (i, s) of each TCP object consists of an integer $i \in [1, I_p]$ that denotes the connection state and an integer $s \in [1, s_p(I_p)+1]$ that denotes the number of packets that have still to be sent. So the transition probabilities are given by the equations:

$$\begin{aligned}
&\tau(s') = w(1-w)^{s'-1} 1_{s' \le s_p(I_p)} + (1-w)^{s_p(I_p)} 1_{s'=S_p(I_p)+1} \\
&K_{(p,i,s),(p,i+1,s')}(\vec{r}) = (1-q(r_p))^{s_p(i)} 1_{i<I_p, s>s_p(i)} \tau(s') \\
&K_{(p,i,s),(p,1,s')}(\vec{r}) = 1_{s \le s_p(i)} \tau(s') \qquad (11) \\
&K_{(p,I_p,s),(p,I_p,s')} = (1-q(r_p))^{s_p(I_p)} 1_{s>s_p(i)} \tau(s') \\
&K_{(p,i,s),(p,d(i),s')} = (1-(1-q(r_p))^{s_p(i)})\ 1_{s>s_p(i)} \tau(s')\,.
\end{aligned}$$

Note that Equation (11) is the key point to decide when a TCP connection finishes its transmission and a new one is created. Function τ is the density function of a truncated geometric random variable, where the probability mass of outcomes greater that $s_p(I_p)$ is concentrated in the last state, $s_p(I_p)+1$.

Proposition 4.3. The number of packets which are sent from two consecutive observations of the transition given by Equation (11) is geometrically distributed with mean $1/w$.

Proof. The proof follows from the memoryless property of the geometric random variable and by the observation that the transition given by Equation (11) occurs with probability 1 from each state (i, s) with $s \le s_p(i)$. □

The memory of the system is still a pair of reals $\vec{R}^N(t) = (R_c(t), R_e(t))$

that denotes the normalised queue length at the gateway at epoch t and $t-1$, respectively. The update rule for the memory is:

$$\begin{aligned} R_c^N(t+1) &= \max\left(R_c^N(t) + S_p^N(t+1) + S_u^N(t+1) - C, 0\right) \\ R_e^N(t+1) &= R_c^N(t)\,, \end{aligned}$$

where:

$$S_p^N(t+1) = \sum_{i=1}^{I_p} \sum_{s=1}^{s_p(I_p)+1} M_{p,i,s}(t+1)\min(s_p(i), s)\,.$$

Proposition 4.4 gives the mean field approximation for this model. Notice that the state space of each TCP object is rather large but we are interested only in the normalised population of a connection state, i.e., $\tilde{\mu}_{p,i}(t) = \sum_{s=1}^{s_p(I_p)+1} \mu_{p,j,s}(t)$.

Proposition 4.4. If $\vec{M}^N(0)$ and $\vec{R}^N(0)$ converge almost surely to $\vec{\mu}(0)$ and $\vec{\rho}(0)$, respectively, and $\forall i = 1, \ldots, I_p$ it holds that $M_{p,i,s}^N(0)/\sum_{s'=1}^{s_p(I_p)+1} M_{p,i,s'}$ converge almost surely to $\tau(s)$, as $N \to \infty$, then for any finite horizon t we have that:

$$\begin{aligned} &\lim_{N\to\infty} \sum_{s=1}^{s_p(I_p)+1} M_{p,i,s}^N(t) = \tilde{\mu}_{p,i}(t) \\ &\lim_{N\to\infty} M_{u,i}^N(t) = \mu_{u,i}(t) \\ &\lim_{N\to\infty} \vec{R}^N(t) = \vec{\rho}(t) \end{aligned}$$

almost surely, where $\tilde{\mu}_{p,i}(t)$, $\mu_{u,i}(t)$ and $\rho(t)$ are defined by the iterative scheme depicted in Table 3.

Notice that Theorem 3.1 is applicable to obtain the mean field approximation

$$
\begin{aligned}
\tilde{\mu}_{p,1}(t+1) &= \sum_{j:d(j)=1} \left(1-(1-q(\rho_p(t)))^{s_p(j)}\right) \tilde{\mu}_j(t)(1-w)^{s_p(j)} && (12)\\
&+ \sum_{j=1}^{I_p} \tilde{\mu}_{p,j}(t)(1-(1-w)^{s_p(j)}) && (13)\\
\tilde{\mu}_{p,i}(t+1) &= \sum_{j:d(j)=i} \left(1-(1-q(\rho_p(t)))^{s_p(j)}\right) \tilde{\mu}_j(t)(1-w)^{s_p(j)}\\
&+ (1-q(\rho_p(t)))^{s_p(i-1)}\tilde{\mu}_{p,i-1}(t)(1-w)^{s_p(i-1)} \quad 1<i<I_p\\
\tilde{\mu}_{p,I_p}(t+1) &= (1-q(\rho_p(t)))^{s_p(I_p-1)}\tilde{\mu}_{p,I_p-1}(t)(1-w)^{s_p(I_p-1)} && (14)\\
&+ (1-q(\rho_p(t)))^{s_p(I_p)}\tilde{\mu}_{p,I_p}(t)(1-w)^{s_p(I_p)}\\
\mu_{u,i}(t+1) &= \sum_{j=1}^{I_u} \kappa_{(u,j),(u,i)}\mu_{u,j}(t)\\
\sigma_p(t+1) &= \sum_{i=1}^{I_p} \tilde{\mu}_{p,i}(t+1)\frac{1-(1-w)^{s_p(i)}}{w} && (15)\\
\sigma_u(t+1) &= \sum_{i=1}^{I_u} \mu_{u,i}(t+1)s_u(i)\\
\rho_c(t+1) &= \max\left(\rho_c(t)+\sigma_p(t+1)+\sigma_u(t+1)-C,0\right)\\
\rho_e(t+1) &= \rho_c(t) && (16)\\
& && (17)
\end{aligned}
$$

Table 3: Iteration system for ECN/RED model with temporary TCP connections.

for $\vec{\mu}$, however the number of states of each component may grow very quickly and make the computation of the iterative model inefficient. In order to derive the interesting performance indices we can use the iterative scheme of Table 3 which aggregates the TCP object states that share the same window size. We now prove its correctness.

PROOF. Let $X_i^p(t)$ be the random variables describing the state of TCP object i at time slot t. The support of $X_i^p(t)$ is $[1, I_p] \times [1, S_p(I_p) + 1]$. Let $X_{i1}^p(t) = j$ if and only if $X_i^p(t) = (j, s)$ for $s \in [1, S_p(I_p) + 1]$ and $X_{i2}^p(t) = s$ if and only if $X_i^p(t) = (j, s)$ for some $j \in [1, I_p]$. By definition we have:

$$\mu_{p,i,s}(t) = \lim_{N \to \infty} \frac{1}{N} \sum_{j=1}^{N_p} 1_{X_j^p(t)=(i,s)} \,.$$

Notice that the random variables $X_j^p(t)$ are asymptotically independent (the proof is analogue to that of [20, Theorem 1]) and are identically distributed, therefore for the law of large numbers we have that for $N \to \infty$, $\mu_{p,i,s}$ almost surely converges to $p_p E[1_{X_j^p(t)=(i,s)}] = p_p \mathbb{P}\{X_j^p(t) = (i, s)\}$ for some j. Noting that by model definition $X_{j1}^p(t)$ and $X_{j2}^p(t)$ are independent, we have:

$$\mu_{p,i,s}(t) = p_p \mathbb{P}\{X_{j1}^p(t) = i\} \mathbb{P}\{X_{j2}^p(t) = s\} = \tilde{\mu}_{p,i}(t) \mathbb{P}\{X_{j2}^p(t) = s\}\,, \text{ a.s.} \quad (18)$$

We now prove the correctness of Equation (13), the other proofs follow the

same line. By definition of $\tilde{\mu}_{p,i}(t)$ and by Theorem 3.1 we have:

$$\begin{aligned}\tilde{\mu}_{p,1}(t) &= \sum_{s=1}^{s_p(I_p)+1} \mu_{p,1,s}(t) \\ &= \sum_{s=1}^{s_p(I_p)+1} \Big(\sum_{j:d(j)=1} \sum_{s'=1}^{s_p(I_p)+1} (1-(1-q(\rho_p(t)))^{s_p(j)}) \\ &\quad \cdot \tau(s)\mu_{p,j,s'}(t)1_{s'>s_p(j)} + \sum_{j=1}^{I_p} \sum_{s'=1}^{s_p(I_p)+1} \tau(s)\mu_{p,j,s'}(t)1_{s'\leq s_p(j)} \Big),\end{aligned}$$

that by Equation (18) becomes:

$$\begin{aligned}&\sum_{s=1}^{s_p(I_p)+1} \tau(s) \sum_{j:d(j)=1} (1-(1-q(\rho_p(t)))^{s_p(j)})\tilde{\mu}_{p,j}(t) \\ &\qquad \cdot \sum_{s'=s_p(j)+1}^{s_p(I_p)+1} \tau(s') + \sum_{s=1}^{s_p(I_p)+1} \tau(s)\tilde{\mu}_{p,j}(t) \sum_{s'=1}^{s_p(j)} \tau(s') \,.\end{aligned}$$

Now, Equation (13) follows after some algebraic simplifications. □

4.4. Modelling SAP-LAW with UDP and temporary TCP connections

The last model we propose considers the SAP-LAW mechanism with temporary TCP connections. We start from the basic model of Section 4.2, and maintain the definition of the memory $\vec{R}^N(t)$ and the transition matrix $K_{(u,i),(u,j)}(\vec{r})$ for UDP objects (see the last equation of (7)). The state of an object associated with a TCP connection has the form $(p, \vec{x}, s)$ where p marks the TCP object, $\vec{x}$ has the same meaning given in Section (4.2) and s denotes the number of packets that has to be sent. Similarly to what we proposed in Section 4.3, we assume that the number of packets sent by a TCP connec-

tion before being closed and restarted is geometrically distributed with mean $1/w$ and hence, by exploiting the memoryless property of geometric random variables (see Proposition 4.3), we obtain the following transition matrix for TCP objects:

$$\begin{aligned} K_{(p,i,j,s),(p,i',j',s')(\vec{r})} &= 1_{f(i,j,h(\vec{r}_u))=(i',j')} \cdot 1_{s>s_p(h(\vec{r}_u))}\tau(s') \\ K_{(p,i,j,s),(p,1,1,s')(\vec{r})} &= 1_{s\leq s_p(h(\vec{r}_u))}\tau(s') \end{aligned}$$

where $\vec{r} = (r_p, r_c, r_{u0}, \vec{r}_u) = (r_p, r_c, r_{u0}, r_{u1}, \ldots, r_{uY})$ is a value of the memory at a certain time slot, $\tau(s')$ is defined in (11) and $f(i,j,z)$ in (10). We can now state the mean field result that, similarly to that proposed by Proposition (4.4) gives an iterative scheme whose computation is more efficient than that straightforwardly obtained by the application of Theorem 3.1 since it considers an aggregation of states for the TCP objects.

Proposition 4.5. If $\vec{M}^N(0)$ and $\vec{R}^N(0)$ converge almost surely to $\vec{\mu}(0)$ and $\vec{\rho}(0)$, respectively, and $\forall i = 1, \ldots, I_p$ it holds that $M^N_{p,i,j,s}(0)/\sum_{s'=1}^{s_p(I_p)+1} M_{p,i,j,s'}$ converge almost surely to $\tau(s)$, as $N \to \infty$, then for any finite horizon t we have that:

$$\begin{aligned} &\lim_{N\to\infty} \sum_{s=1}^{s_p(I_p)+1} M^N_{p,i,j,s}(t) = \tilde{\mu}_{p,i,j}(t) \\ &\lim_{N\to\infty} M^N_{u,i}(t) = \mu_{u,i}(t) \\ &\lim_{N\to\infty} \vec{R}^N(t) = \vec{\rho}(t) \end{aligned}$$

almost surely, where $\tilde{\mu}_{p,i,j}(t)$, $\mu_{u,i}(t)$ and $\vec{\rho}(t)$ are defined by the iterative scheme of Table 4 and function h is defined by Equation (9).

The proof follows the lines of that of Proposition 4.4.

$$
\begin{aligned}
\tilde{\mu}_{p,1,1}(t+1) &= \sum_{i'=1}^{I_p}\sum_{j'=1}^{i}\tilde{\mu}_{p,i',j'}(t)(1-(1-w)^{\min(s_p(i'),s_p(h(\vec{\rho}_u(t))))}) \\
\tilde{\mu}_{p,i,j}(t+1) &= \sum_{i'=1}^{I_p}\sum_{j'=1}^{i}\tilde{\mu}_{p,i',j'}(t)1_{f(i',j',h(\vec{\rho}_u(t)))=(i,j)}(1-w)^{\min(s_p(i'),s_p(h(\vec{\rho}_u(t))))} \\
\sigma_p(t+1) &= \sum_{i=1}^{I_p}\tilde{\mu}_{p,i,j}(t+1)\frac{1-(1-w)^{\min(s_p(i),s_p(h(\vec{\rho}_u(t))))}}{w} \\
\sigma_u(t+1) &= \sum_{i=1}^{I_u}\mu_{u,i}(t+1)s_u(i) \\
\rho_c(t+1) &= \max\Big(\rho_c(t)+\sigma_p(t+1)+\sigma_u(t+1)-C,0\Big) \\
\rho_e(t+1) &= \rho_c(t) \\
\rho_{ui}(t+1) &= \rho_{u(i-1)}(t), \quad 1\le i\le Y \\
\rho_{u0}(t+1) &= \sigma_u(t+1)
\end{aligned}
$$

Table 4: Iteration system for SAP-LAW model with temporary TCP connections.

Remark 4.2. Notice that there are some equivalence relations among the models specified in this section. Specifically the ECN/RED, with greedy or temporary TCP connections is equivalent to the corresponding SAP-LAW model when $C \geq \max\{s_p(i), 1 \leq i \leq I_p\} \cup \{s_u(j), 1 \leq j \leq I_u\}$. In fact, in this case none of the two congestion mechanisms is applied and hence the gateway is not a bottleneck for the system. Another interesting relation is that the model presented in Section 4.1 (Section 4.2) can be derived as the limit for $w \to 0^+$ of the model presented in Section 4.3 (Section 4.4). In fact when w is very small, the amount of data that each TCP connection sends is so high that its behaviour resembles the one of a greedy connection.

Nevertheless, we think that having presented and described the four models should have helped the clarity of the exposition.

5. Validation of the model

We validate the proposed modelling methodology in two ways:

- We compare the trace of the queue length at the AP obtained by the Network Simulator (NS2);
- We show the convergence rate to the mean field model by comparing the stochastic model with a finite number of objects to the deterministic limiting model.

The network simulator scenario is the following: we considered 10 TCP connections (Reno) with maximum rate of $10Mb/s$ and transmitting through a bottleneck with capacity $10Mbps$. For the sake of brevity, we show the comparison obtained with the ECN/RED model with thresholds set at 30 and 70 packets, with each packet size of $1KB$. The dropping probability function is chosen as:

$$max_p \frac{ql - min_{th}}{max_{th} - min_{th}},$$

with $max_p = 0.1$ [31]. The round trip time of the topology is $100ms$. The comparison between the mean field model and the simulation trace of the queue length is shown in Figure 1.

The models for ECN/RED have been validated with that proposed in [20] by setting $p_p = 1$ and $p_u = 0$. In Figure 2 we show the convergence to the mean field of the ECN/RED model with increasing number of objects. The parameters for the TCP connections and the gateway used for the simulation

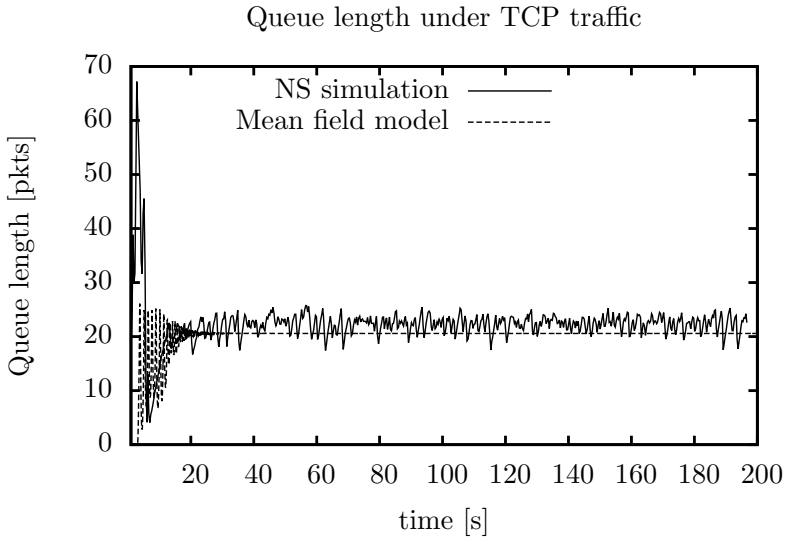

Figure 1: Model validation against NS simulation: queue length.

are: $I_p = 100$, $s_p(i) = 10 \cdot i$, $p_p = 0.7$, $C = 600$, $\gamma = 5 \cdot 10^{-8}$, function d halves the size of the windows.

We assume a bursty behaviour of UDP traffic. We model the state of each UDP object as a periodic Descrete Time Marchov Chain (DTMC) consisting of $I_u = 15$ such that $\kappa_{(ui),(u,i+1)} = 1$ for $1 \leq i < I_u$, and $\kappa_{(u,I_u),(u,1)} = 1$. Vector $s_u(i) = (0, \ldots, 700, 800, 700)$. The initial state of the TCP connections is chosen with uniform probability between 1 and I_p, while all the UDP objects start from state 1. Figure 2 shows a comparison between the mean field approximation and a simulation trace obtained with different numbers of objects in terms of normalised queue length. Notice the effects of the periodic bursts on the queue length and the reaction of the ECN/RED mechanism to contain the queue length. Analogously the SAP-LAW mean field model has been validated against simulation.

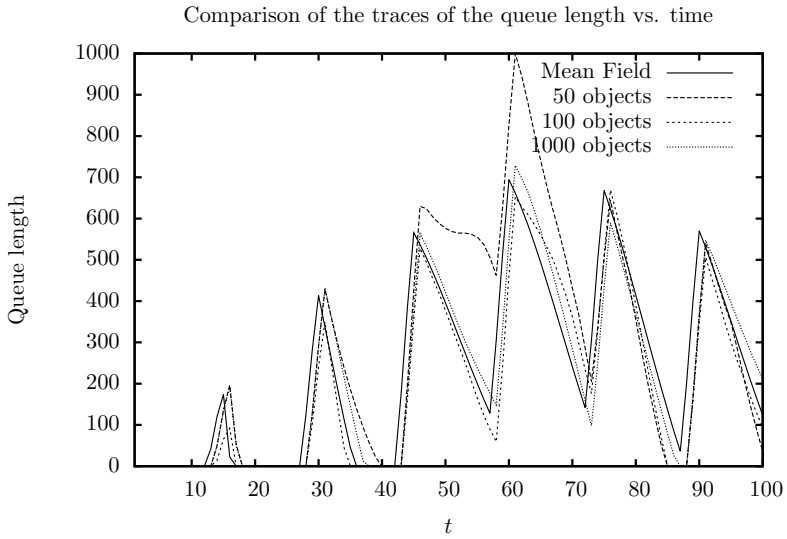

Figure 2: Convergence to mean field of greedy ECN/RED: queue length trace plot.

6. Performance evaluation

In this section we use the mean field models presented in Section 4 to compare the ECN/RED, SAP-LAW mechanisms in terms of expected queue length Q (expressed in normalised number of packets) and system's throughput T (expressed in normalised sent packets per time slot). Assuming that no packet is discarded, i.e., the control mechanism for ECN/RED is performed by sending notification packets rather than by discarding some random packets, we also indirectly derive the expected queuing time using Little's law $W = Q/T$ (expressed in time slots) and assuming a simulation time horizon long enough to stabilise these values. Notice that W denotes the expected number of time slots that a packet has to wait in the queue *before* being sent. In all the models of our experiments we use for SAP-LAW $Y = 1$ and function $\tilde{y}$ is the identity, i.e., we use a time window of one round trip time

	SAP-LAW	ECN $\gamma = 5E-6$	ECN $\gamma = 5E-7$
T	591.27	585.69	594.23
Q	0	5.73	53.72
W	0	0.098	0.0901

Table 5: Comparison of ECN/RED with SAP-LAW under greedy TCP and smooth UDP. Parameters: $I_p = 100$, $I_u = 10$, $s_p(i) = 10 * i$, $C = 600$ $s_u = (0, \ldots, 1000, 3000, 1000)$.

to estimate the UDP traffic. Function $d(i)$ for ECN/RED simply halves the size of the current window. The percentage of TCP objects is 70% of the total. The finite horizon is fixed at 800 time slots.

6.1. Greedy TCP connections and smooth UDP traffic

In this section we study the models for ECN/RED and SAP-LAW under the scenario of greedy TCP connections, i.e., they tend to use all the size of their sending windows, and smooth UDP traffic. The behaviour of each UDP connection is that described in Section 5 but in order to smooth the UDP packet sending we set the initial state of a connection with uniform probability on the object's state. The results of the experiment and its parameters are shown in Table 5. Figure 3 shows the queue length trace for the two ECN/RED models obtained with different values of γ. This scenario is ideal especially for the SAP-LAW mechanism. In fact, the smoothness of the UDP traffic (in the limit it tends to become constant) allows the gateway to accurately predict the bandwidth to reserve for UDP. On the other hand, the greedy behaviour of TCP connections prevents the SAP-LAW mechanism to over-estimate the bandwidth needed by each connection since, after some time all the sending windows will be opened at their maximum sizes. For this reason, in this ideal scenario, SAP-LAW gives the best performances by

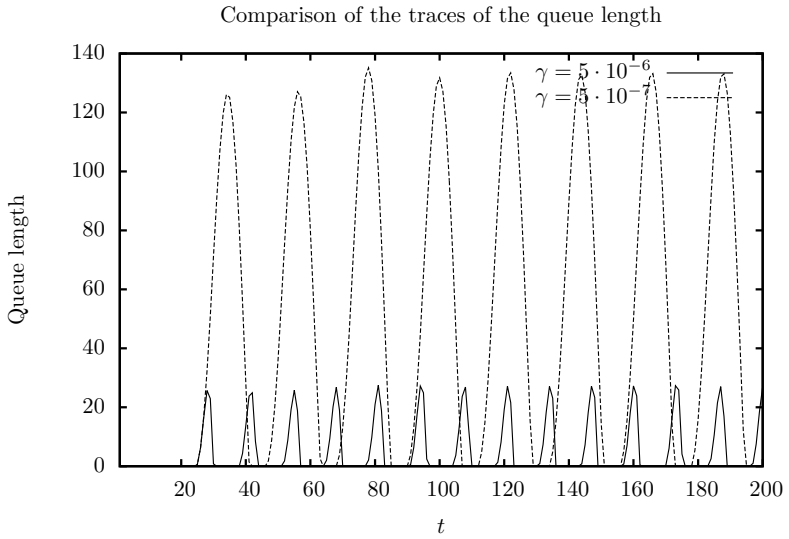

Figure 3: Traces of the queue length for ECN/RED models with greedy TCP and smooth UDP connections.

obtaining a high throughput with no waiting time.

6.2. Greedy TCP connections and bursty UDP traffic

In this part we compare the ECN/RED and the SAP-LAW congestion control mechanisms under bursty UDP. We still use the models presented in Sections 4.1 and 4.2. In this case the SAP-LAW model shows an instable behaviour that proves that this mechanism alone is not sufficient to safely prevent the congestion in any scenario. This issue is presented when the UDP traffic is very bursty and the monitor window is small as in our case. The SAP-LAW policy does not take into account the actual queue length but only estimates the instantaneous UDP traffic. Therefore, this parameter must be chosen carefully when implementing this congestion control mechanism. Figure 4 shows the traces of the queue lengths in the ECN/RED

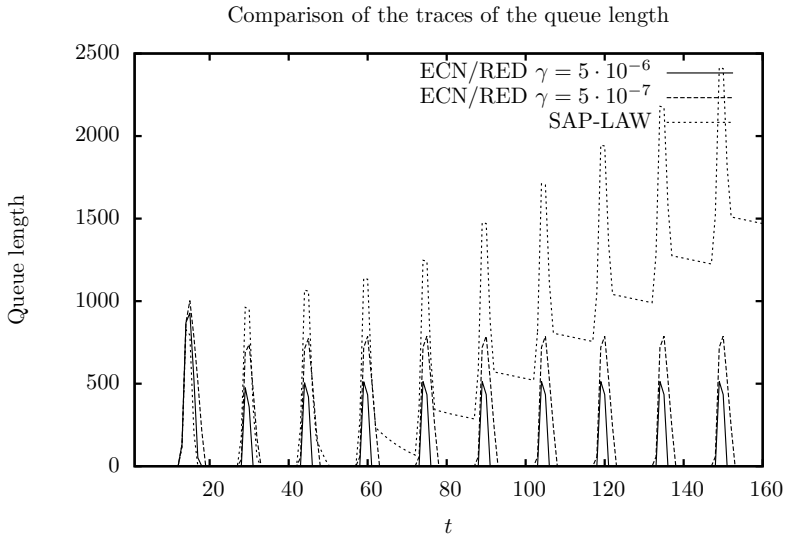

Figure 4: Instability of SAP-LAW model under bursty UDP traffic.

and SAP-LAW. The graphs have been obtained with the same parameters of the experiment run in Section 6.1 but with all identical UDP objects' initial states. Table 6 shows the performance measures for ECN. It is worth of notice that the burstiness of the UDP activity strongly deteriorates the performance measures of TCP traffic. In Table 7 we show the behaviour of the two congestion mechanisms under bursty UDP traffic but in stability. We can see that although the throughput of SAP-LAW is much higher than that of ECN, also the queue length tends to be higher, so reducing the benefits

	SAP-LAW	ECN $\gamma = 5E-6$	ECN $5E-7$
T	n/a	276.00	428.85
Q	n/a	64.52	155.34
W	n/a	0.234	0.3622

Table 6: Comparison of ECN/RED with SAP-LAW under greedy TCP and bursty UDP. Parameters: $I_p = 100$, $I_u = 10$, $s_p(i) = 10 * i$, $C = 600$ $s_u = (0, \ldots, 1000, 3000, 1000)$.

	SAP-LAW	ECN $\gamma = 5 \cdot^1 0{-}6$	ECN $\gamma = 5 \cdot 10^{-7}$
T	713.27	333.29	529.4077
Q	153.95	35.20	83.79
W	0.2000	0.1056	0.1583

Table 7: Comparison of ECN/RED with SAP-LAW under greedy TCP and bursty UDP. Parameters: $I_p = 100$, $I_u = 10$, $s_p(i) = 10 * i$, $C = 800$ $s_u = (0, \ldots, 1000, 3000, 1000)$.

that we observed in the scenario with smooth UDP.

6.3. Temporary TCP connections

In this section we assume that each TCP connection has to send a number of packets modelled by an independent geometric random variable with mean w^{-1}. After sending this amount of packets, the connection restarts from state 1. The models that are studied have been presented in Sections 4.3 and 4.4. The results of the analysis are shown in Table 8. The table shows that SAP-LAW pays a reduction of the throughput because the mechanism is fair in dividing the residual bandwidth among the TCP connections not considering that some of them may be in their initial phase. This does not happen for ECN/RED since it tends to reduce the window size of fast connections and allows the new ones to grow.

This is even more evident when we introduce UDP burstiness. Figure 5 and 6 compare the queue length and the bottleneck's throughput for different sizes of w. We assume $w(\omega) = 10^3 2^\omega$, and $C = 300$, $I_p = 100$, $s_p(i) = 10i$, $I_u = 15$, $s_u = (0, 0, \ldots, 500, 600, 500)$. Notice that for low values of w ECN/RED shows its benefits in the queue length control policy as discussed above, whereas when w grows SAP-LAW takes advantage of being more aggressive in the bandwidth usage and this compensates the assignment of

	SAP-LAW	ECN $\gamma = 5E-6$	ECN $\gamma = 5E-7$
T	219.17	300.06	295.50
Q	0.2160	361.1346	39.335
W	$0.85E-4$	1.2035	0.133

Table 8: Comparison of ECN/RED with SAP-LAW under greedy TCP and smooth UDP. Parameters: $I_p = 100$, $I_u = 10$, $s_p(i) = 10 * i$, $C = 800$ $s_u = (0, \ldots, 1000, 3000, 1000)$, $w = 10^4$.

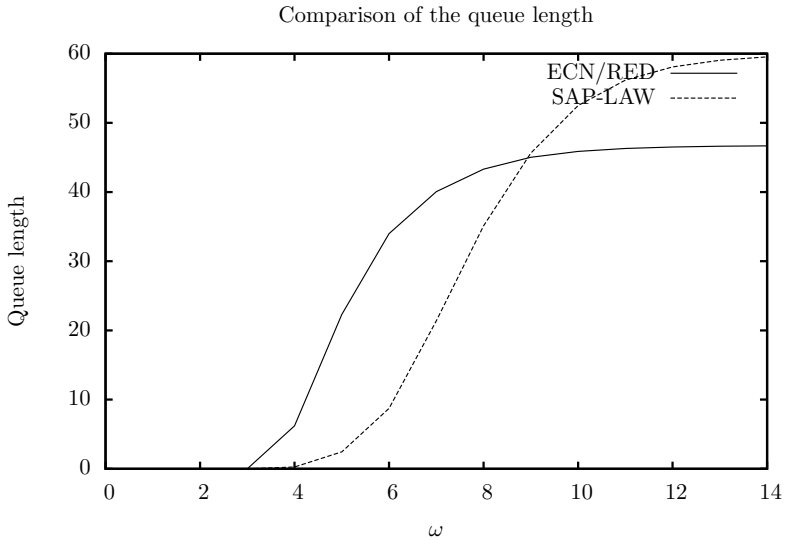

Figure 5: Comparison of the queue lengths with UDP burstiness and temporary TCP conncetions.

bandwidth to slow connections.

7. Conclusion

In this paper we have compared the performance of two mechanisms for the congestion control at a shared gateway in a TCP/IP based network: the ECN/RED and the SAP-LAW. The comparison is performed assuming a large number of TCP and UDP transmissions and considering two different

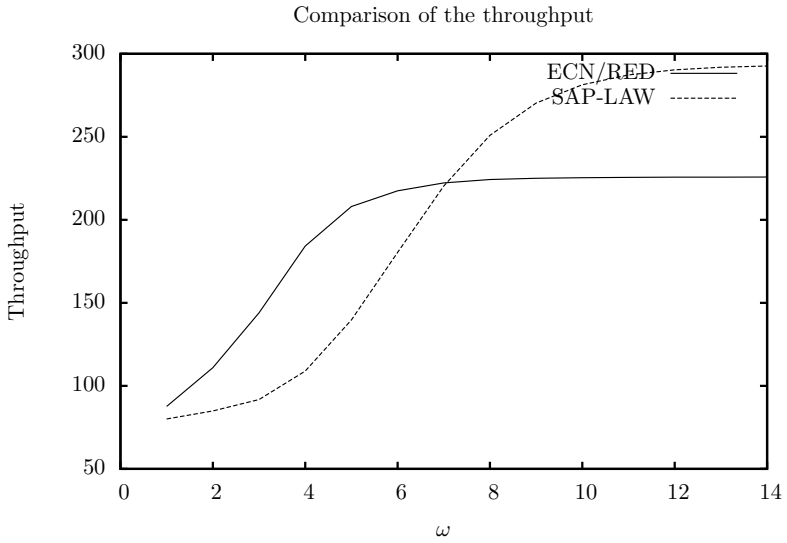

Figure 6: Comparison of the gateway throughput with UDP burstiness and temporary TCP conncetions.

behaviours for the TCP connections (greedy or temporary) and for the UDP traffic (smooth or bursty). We showed, according to the mean field models, that the SAP-LAW gives better performance (in terms of expected queue length and throughput) than ECN/RED in case of greedy TCP connections and smooth UDP traffic. If the TCP connections are not greedy, SAP-LAW shows worse performance because it allocates part of the available bandwidth to TCP connections that are in their initial phases. The impact of burstiness on SAP-LAW can be dangerous: we showed that a wrong choice of the time interval used to estimate the intensity of the UDP traffic may lead to instability. As for the ECN/RED mechanism, the performance of its congestion control mechanism is not affected by temporary TCP connections since when a potential congestion is detected, the fastest TCP connections are those with highest probability of being slowed down. On the other hand, the impact

of burstiness on the UDP traffic can strongly reduce the throughput. This happens because during the peak of UDP traffic, the ECN/RED marks the packets with high probability thus reducing the associated connections' window size, not taking into account that the burst is a short-time phenomenon (if there where only TCP connections, it would be correct to expect that a high queue length at time t is followed by a worse situation at time $t+1$ if countermeasures are not adopted). It worth of mentioning that the mean field analysis supports the *fast simulation* that allows to study the detailed performance indices associated with a single object in the mean field regime in a very efficient way.

Table of notation

In this section we summarise the essential notation that we have used along the paper. Some general rules are that letter p is associated with TCP variables while letter u is associated with UDP variables. When passing to the mean field approximation the meaning of the Greek letters is inherited by the Latin ones according to the following schema: M is associated with μ, S with σ and R with ρ.

Symbol	Meaning
$\mathcal{S}_p$, $\mathcal{S}_u$	Set of states of a TCP connection
I_p	Number of possible window sizes of a TCP connection
I_u	Number of states of a UDP agent
$s_p(i)$	Number of packets sent by a TCP conn. in state i
$s_u(i)$	Number of packets sent by a UDP agent in state i
N_p, N_u, N	Number of TCP conn., UDP agents, and their
p_p, p_u	Proportion of TCP and UDP objects
$M^N_{p,\nu}(t)$	Proportion of TCP conn. in state ν for N objects at time t
$M^N_{u,\nu}(t)$	Proportion of UDP agents in state ν for N objects at time t
$S^N_p(t)$	Expected number of packets sent by each TCP conn. at time t with N objects
$S^N_u(t)$	Expected number of packets sent by each UDP agent at time t with N objects
$q(\cdot)$	Packet marking function for the ECN/RED
$K_{(c,\nu),(c,\nu')}(\vec{r})$	Transition probability for an object of class c from state ν to ν' depending on the bottleneck state $\vec{r}$
$1/w$	Expected number of packets sent by a TCP connection before terminating the transmissions
$y(\cdot)$	Function used to compute the estimation of the UDP needs for the SAP-LAW model
$d(j)$	Congestion window size in case of packet dropping at size j
T	Throughput
Q	Expected queue length
W	Expected waiting time

References

[1] J. F. Kurose, K. W. Ross, Computer Networking: A Top-Down Approach, 6th Edition, Pearson, USA, 2013.

[2] S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Trans. on Networking 1 (4) (1995) 397–413.

[3] S. H. Low, F. Pananini, J. Wang, J. C. Doyle, Linear Stability of TCP/RED and a Scalable Control, Computer Networks 43 (5) (2003) 633–647.

[4] S. Floyd, TCP and Explicit Congestion Notification, SIGCOMM Comput. Commun. Rev. 24 (5) (1994) 8–23.

[5] J. H. Salim, U. Ahmed, Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks, RFC Editor.

[6] 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 Trans. on Consumer Electronics 52 (4) (2006) 1280–1288.

[7] C. E. Palazzi, S. Ferretti, M. Roccetti, Smart Access Points on the Road for Online Gaming in Vehicular Networks, Entertainment Computing 1 (1) (2009) 17–26.

[8] G. Marfia, M. Roccetti, TCP at Last: Reconsidering TCP's Role for Wireless Entertainment Centers at Home, IEEE Trans. on Consumer Electronics 56 (4) (2010) 2233–2240.

[9] C. Barakat, E. Altman, W. Dabbous, On TCP Performance in a Heterogeneous Network: A Survey, IEEE Communications Magazine 38 (1) (2000) 40 – 46.

[10] J. Widmer, R. Denda, M. Mauve, A Survey on TCP-Friendly Congestion Control, IEEE Network 15 (3) (2001) 28 – 37.

[11] S. Liu, T. Başar, R. Srikant, TCP-Illinois: A Loss- and Delay-based Congestion Control Algorithm for High-Speed Networks, Performance Evaluation 65 (6–7) (2008) 417 – 440.

[12] S. Ha, I. Rhee, L. Xu, CUBIC: A New TCP-Friendly High-Speed TCP variant, ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel 42 (5) (2008) 64 – 74.

[13] L. A. Grieo, S. Mascolo, Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control, Computer Communication Review 34 (2) (2004) 25 – 38.

[14] E. Altman, K. Avrachenkov, C. Barakat, TCP in Presence of Bursty Losses, in: Proc. of SIGMETRICS, 2000, pp. 124–133.

[15] G. Chatranon, M. A. Labrador, S. Banerjee, A Survey of TCP-Friendly Router-Based AQM Schemes, Computer Communications 27 (15) (2004) 1424 – 1440.

[16] W. Kang, F. Kelly, N. H. Lee, R. R.J. Williams, Fluid and Brownian Approximations for an Internet Congestion Control Model, in: Proc. of CDC 2004, Vol. 4, 2004, pp. 3938–3943.

[17] A. Chydzinski, A. Brachman, Performance of AQM Routers in the Presence of New TCP Variants, in: Proc. of 2nd Intl. Conf. on Advances in Future Internet (AFIN 2010), 2010, pp. 88–93.

[18] T. Bonald, M. May, J.-C. Bolot, Analytic Evaluation of RED Performance, in: Proc. of INFOCOM 2000, Vol. 3, 2000, pp. 1415–1424.

[19] H. Rahman, K. Giridhar, G. Raina, Performance Analysis of Compound TCP with AQM, in: Proc. of 11th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 2013), 2013, pp. 492–499.

[20] P. Tinnakornsrisuphap, A. Makowski, Limit Behavior of ECN/RED Gateways Under a Large Number of TCP flows, in: Proc. of INFOCOM 2003, Vol. 2, 2003, pp. 873–883.

[21] J.-Y. Le Boudec, D. McDonald, J. Mundinger, A Generic Mean Field Convergence Result for Systems of Interacting Objects, in: Proc. of QEST 2007, IEEE, 2007, pp. 3–18.

[22] A. Bobbio, M. Gribaudo, M. Telek, Analysis of Large Scale Interacting Systems by Mean Field Method, in: Proc. of QEST 2008, 2008, pp. 215–224.

[23] F. Baccelli, M. Lelarge, D. McDonald, Mestable Regimes for Multiplexed TCP Flows, in: 42nd Annual Allerton Conference on Communication Control, and Computing, University of Illinois at Urbana-Champaign, Allerton House, Monticello, Illinois, USA, 2004, pp. 1005–1011.

[24] R. Srikant, The Mathematics of Internet Congestion Control, Springer, 2004.

[25] C. E. Palazzi, N. Stievano, M. Roccetti, A Smart Access Point Solution for Heterogeneous Flows, in: International Conference on Ultra Modern Telecommunications Workshops, 2009, pp. 1–7.

[26] M. Gerla, R. Locigno, S. Mascolo, W. Weng, Generalized Window Advertising for TCP Congestion Control, European Transactions on Telecommunications (6) (2002) 549–562.

[27] M. Barbera, A. Lombardo, C. Panarello, G. Schembra, Active Window Management: An Efficient Gateway Mechanism for TCP Traffic Control, in: IEEE International Conference on Communications, 2007, pp. 6141–6148.

[28] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.

[29] G. Casale, E. Zhang, E. Smirni, KPC-Toolbox: Simple yet effective trace fitting using Markovian Arrival Processes, in: Proc. of QEST 2008, 2008, pp. 83–92.

[30] A. S. Tanenbaum, Computer Networks, Prentice-Hall, 2003.

[31] C. V. Hollot, V. Misra, D. Towsley, W. bo Gong, A control theoretic analysis of RED, in: Proc. of INFOCOM 2001, IEEE, 2001, pp. 1510–1519.