Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

CrossMark

Congestion control of TCP flows in Internet routers by means of index policy ☆

K. Avrachenkov [a], U. Ayesta [b,c,d,g], J. Doncel [c,d,f,*], P. Jacko [e,f]

[a] INRIA Sophia Antipolis, Valbonne, France
[b] IKERBASQUE, Basque Foundation for Science, 48011 Bilbao, Spain
[c] Universite de Toulouse, LAAS, F-31400 Toulouse, France
[d] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[e] Department of Management Science & LANCS Initiative, Lancaster University, Lancaster, UK
[f] BCAM-Basque Center for Applied Mathematics, Bilbao, Spain
[g] UPV/EHU, University of the Basque Country, 20018 Donostia, Spain

ARTICLE INFO

ABSTRACT

In this paper we address the problem of fast and fair transmission of flows in a router, which is a fundamental issue in networks like the Internet. We model the interaction between a source using the Transmission Control Protocol (TCP) and a bottleneck router with the objective of designing optimal packet admission controls in the router queue. We focus on the relaxed version of the problem obtained by relaxing the fixed buffer capacity constraint that must be satisfied at all time epoch. The relaxation allows us to reduce the multi-flow problem into a family of single-flow problems, for which we can analyze both theoretically and numerically the existence of optimal control policies of special structure. In particular, we show that for a variety of parameters, TCP flows can be optimally controlled in routers by so-called index policies, but not always by threshold policies. We have also implemented the index policy in Network Simulator-3 and tested in a simple topology their applicability in real networks. The simulation results show that the index policy achieves a wide range of desirable properties with respect to fairness between different TCP versions, across users with different round-trip-time and minimum buffer required to achieve full utility of the queue.

1. Introduction

This paper deals with congestion control and buffer management, two of the most classical research problems in networking. The objective of congestion control is to control the traffic injected in the network in order to avoid congestive collapse. Most traffic in the Internet is governed by TCP/IP (Transmission Control Protocol and Internet Protocol) [1,2]. TCP protocol tries to adjust the sending rate of a source to match the available bandwidth along the path. In the absence of congestion signals from the network TCP increases congestion window gradually in time, and upon the reception of a congestion signal TCP reduces the congestion window, typically by a multiplicative factor. Buffer management determines how congestion signals are generated. Congestion signals can be either packet losses or ECN (Explicit Congestion Notifications) [3]. At the present state of the Internet, nearly all congestion signals are generated by packet losses. Packets can be dropped either when the router buffer is full or when an AQM (Active Queue Management) scheme is employed [4].

In this paper we develop a rigorous mathematical framework to model the interaction between a TCP source and a bottleneck queue with the objective of designing optimal packet admission controls in the bottleneck queue. The TCP sources follow the general family of *Additive Increase Multiplicative Decrease* pattern that TCP versions like New Reno or SACK follow [5]. A TCP source is thus characterized by the decrease factor, which determines the multiplicative decrease of the congestion window in the event of a congestion notification (in TCP New Reno the decrease factor takes the value 1/2).

The objective is to design a packet admission control strategy to use the resources efficiently and provide satisfactory user experience. Mathematically we formulate the problem using the Markov Decision Process (MDP) [6] as a dynamic resource allocation problem, which extends the restless bandit model introduced in [7]. The router aims at maximizing the total aggregated utility. As utility function we adopt the parameterized family of generalized α-fair utility functions, which depending on the value of α permit to recover a wide variety of utilities such as max–min, maximum throughput and proportional fair [8,9]. The fixed bandwidth capacity constraint that must be satisfied at all time epochs makes that the problem can be solved analytically only in simplistic scenarios. However, the problem becomes tractable if the fixed capacity constraint is relaxed so that the bandwidth allocation must be satisfied only on average, known as the Whittle relaxation. This relaxation allows to see congestion control at the router as a family of per-flow admission control problems, thus reducing the complexity of a multi-flow problem into a family of single-flow problems. In our main contribution, for the single-flow problem we analyze both theoretically and numerically the existence of optimal control policies of special structure. In particular, we show that for a variety of parameters, TCP flows can be optimally controlled in routers by so-called index policy, but not always by threshold policy.

This solution approach based on the Whittle relaxation has gained notorious success in recent years and has been for instance used in scheduling in wireless networks, dynamic/stochastic knapsack problems, online marketing, or military applications. We refer to [10] for a recent account on the methodology and applications. The interest in the approach is justified by mathematical results that show that, under some additional assumptions, the heuristic is asymptotically optimal [11]. In practice it has been reported on various occasions that the heuristic provides a close-to-optimal solution to the problem studied [10, Chapter 6].

The index policy requires that the TCP sources are assigned a dynamic index that depends on their current congestion window and the TCP variant implemented at the source. Such a *transmission index* measures the efficiency of transmission, and therefore can be interpreted as a priority for transmission. When a packet arrives and the router wants to send a congestion notification (i.e., the buffer is full or the implemented AQM requires to drop or mark a packet), the router will choose the packet with *smallest index*. In the event that more than one packet share the same

smallest index, the packet that has been the longest in the queue is chosen.

Building on the cooperative nature of the TCP protocol, we assume that, apart from adjusting the window according to the congestion signals received from the network, the source writes the current index into the packet header, which are read by the routers. Alternatively, index policy can be implemented only in the routers, which however requires to identify or infer the flows, their TCP variants, and the current congestion window of each flow.

Note that our scheme requires as much or less information exchanges than the related congestion control protocols XCP [12], ACP [13] and RCP [14]. Moreover, these congestion control protocols aim at achieving max–min fairness, whereas our scheme allows to achieve several notions of fairness by using the generalized α-fairness framework. Max–min is a particular case of α-fairness when the parameter α goes to infinity. The most important TCP variants are FAST TCP [15], TCP CUBIC [16] and TCP Compound [17], which are implemented in Linux and Windows, respectively. The main originality of our approach with respect to previously existing variants is that our index protocol aims at achieving efficiency and fairness on shorter-term time scales, by maximizing the mean fairness of the actual rate instead of the fairness of the mean rates.

We have implemented our solution in NS-3 [18] and performed extensive simulations in a benchmark topology to explore and validate the properties of the algorithm, and to assess the improvement with respect to *DropTail router* (that drops packets when the buffer is full) and *RED router* (that has implemented the Random Early Detection (RED) AQM mechanism). In the simulations we focus on the case $\alpha = 1$, since it was shown that the current Internet (with DropTail routers) maximizes the aggregate sum of logarithmic utilities of the time-average transmission rates [19].

The simulation results show that the algorithm has several desirable properties with respect to fairness and efficiency:

- We achieve inter protocol fairness between different versions of AIMD TCP.
- We achieve fairness with respect to the round-trip-time on the scale of congestion periods.
- We improve fairness across users with different round-trip-time.
- We improve fairness across users compared to DropTail and RED.
- In a router implementing index policies, a smaller buffer size is needed to get the same throughput.

We believe that our approach opens a new avenue of research to design in a combined manner congestion control and queue management policies. This paper represents a first attempt, and briefly discuss in Section 8 how to assess several aspects that we did not consider in this paper, as, for instance, the performance of the algorithm in the presence of short-lived TCP connections, or when implemented in many routers in the network. We also believe that we provide fundamental framework and ideas useful or

directly applicable also in next-generation Internet architectures such as the Information-centric networking (ICN) and wireless networks.

The rest of the paper is organized as follows. In Section 2 we put our work in the context of the existing literature. In Section 3 we describe the model and in Section 4 we state the problem. Section 5 shows the benefits of the relaxation of the described problem. In Section 6 we analyze the index policies for single-flow problems. We also establish several properties of index policies to be taken into account. With this results in the hand, we present in Section 7 the conclusion of the simulation done with NS-3. In Section 8 we discuss possible modeling extensions and practical implementation aspects. Finally, some conclusions are drawn and possible extensions are discussed in Section 9.

A three-pages extended abstract of this paper appears in [20].

2. Related work on TCP and buffer management

The seminal work by [21] established the basis of TCP, and [2] added several key features and brought the TCP protocol very close to its current form. TCP is a completely distributed algorithm run by the end hosts, and it aims to share the resources of the network among all flows in an efficient and fair way. In other words, assuming that the network is responsible for the end-to-end transmission of packets, TCP tries to determine the fair share of the connections. The latter is characterized by the *congestion window*, denoted by *cwnd*, which captures the number of packets a connection can have simultaneously in the network at any given time. The basic principles according to which TCP works are extremely simple. The key idea is the dynamic window sizing scheme proposed by [2]. Consider that any packet has an identifier that allows the receiver to identify it uniquely. The sender sends packets to the destination, and the destination sends back to the sender small packets (known as acknowledgements, or simply ACKs) acknowledging the correct reception of each packet. This simple mechanism allows the sender to infer whether its packets are reaching the destination, as well as implicitly to infer the congestion level of the network. While packets reach the destination, the sender is allowed to increase the amount of packets per unit of time it injects into the network. More precisely, TCP increases by one the value of *cwnd* in the absence of network congestion notification.

On the other hand, when a packet of a sequence of packets is lost or dropped, the receiver will notice it by receiving an out-of-order packet, and it will send a special acknowledgment (known as duplicate ACK) back to the source. Upon the reception of three duplicate ACKs or an ECN acknowledgement (in case of marked packet), the source will infer that the network is congested and it will take two actions: first, the lost packet (and the subsequent ones already sent) will be retransmitted and second, the packet sending rate will be reduced by a multiplicative factor. In the last two decades there has been a huge effort in order to improve and develop new congestion control

algorithms, and we briefly mentioned some of them in the introduction. An extensive overview and comparison of different TCP version is given in [22].

Buffer management algorithms determine how packets are dropped/marked, or more generally how congestion signals are generated. Packets can be dropped either when the router buffer is full, which is referred to as DropTail, or when AQM (Active Queue Management) scheme, e.g., RED, is employed [4]. At the present state of the Internet, nearly all congestion signals are generated by packet drops. An alternative to packet drops is ECN (Explicit Congestion Notifications) [3], which marks packets but does not drop them. Despite the tremendous research effort it seems that given the ambiguity in the choice of parameters, in reality AQM schemes are rarely used in practice. On the other hand, in the basic Drop Tail routers, the buffer size is the only one parameter to tune apart from the router capacity. We refer the interested reader to [23] and references therein for more information on the problem of optimal choice of buffer size. The main drawbacks of the basic DropTail mechanism is the synchronization of TCP flows and RTT unfairness. TCP connections with shorter RTT achieve faster high transmission rates than TCP connections with larger RTT.

Our approach to AQM, based on sound theoretical background of Markov Decision Processes, does not require tuning of any additional parameters and allows to realize the following interesting features: higher network utilization and more fair data transmission of users with different TCP variants, different RTTs.

3. Problem description

We describe in this section the congestion control problem of multiple flows at a bottleneck router. Suppose there are K flows trying to deliver packets to their destinations via a bottleneck router with the following parameters:

- C the bandwidth, i.e., the deterministic *link capacity* (in packets per second).
- B the buffer size (in packets).

Suppose further that flow $k \in \mathcal{K} := \{1, 2, \ldots, K\}$ has implemented an additive-increase multiplicative-decrease (AIMD) mechanism as in the Transmission Control Protocol (TCP). The congestion window *cwnd* is adapted according to received acknowledgements: for each received non-duplicate acknowledgment (*positive acknowledgement*), *cwnd* is increased by the reciprocal of the current value of *cwnd* (which approximately corresponds to an increase by one packet during a round-trip-time RTT without lost packets) unless it has reached the maximum advertised congestion window allowed N_k

$$cwnd := min\left\{ \left\lfloor cwnd + \frac{1}{cwnd} \right\rfloor, N_k \right\}; \tag{1}$$

for each triple duplicate acknowledgment (*negative acknowledgment*), *cwnd* is decreased multiplicatively using the formula

$$cwnd := \max\left\{ \lfloor \gamma_k \cdot \text{congestion window} \rfloor, 1 \right\}, \tag{2}$$

where $0 \leqslant \gamma_k < 1$ is the multiplicative decrease factor. The function $\lfloor \cdot \rfloor$ denotes the floor function. We consider that independently of the number of dropped/marked packets, the congestion window is only decreased once per RTT. The flow starts in the (deterministic) initial congestion window value n_k and it always has packets willing to deliver. For model transparency we ignore timeouts (and the corresponding slow-start phase of TCP), since they are known to be significantly less frequent than congestion events and we expect they would not affect our solution significantly. Nevertheless, it would be possible to deal with it with our approach.

The objective for the router is to use the resources efficiently and provide satisfactory user experience so that

- the overall number of delivered packets per long time intervals (time-average throughput) is as large as possible and
- the flows are treated fairly by having their congestion windows (number of packets in the network) as equal as possible and
- the utilization of the bottleneck queue is as high as possible.

4. Markov decision processes model

In this section we present a general mathematical formulation of the congestion control problem using a Markov Decision Process (MDP) framework of resource allocation [24], which extends the restless bandit model [7]. To the best of our knowledge restless bandit model formulation has not been used previously in congestion control, an area in which an important body of literature is devoted to deterministic fluid models [25].

In order to make the model tractable and to be able to design in a simple, implementable solution, we will consider several simplifications of the features of the problem. Nevertheless, the performance of the solution is then evaluated in Section 7 in the original setting of the problem without simplifications described above.

Let us consider the time slotted into discrete time epochs $t \in \mathcal{T} := \{0, 1, 2, \ldots\}$, which correspond to time periods of one round-trip time (RTT), which is assumed equal for all the flows. However, in Section 7 this constrain disappears and we obtain nice results too. So, we consider users with equal RTT in the theoretical part of this paper, although we can conclude from the simulations this constrain can be omitted.

We assume that all packets are of the same size, which we further define to be one bandwidth capacity unit. The router takes decisions about admitting or rejecting the flows at every time epoch t.

4.1. AIMD flows

Every flow can be allocated either the capacity required by its current congestion window (being *admitted*) and transmitted, or zero capacity (being *rejected*). We denote by $\mathcal{A} := \{0, 1\}$ the *action space*, where 0 corresponds to blocking and 1 corresponds to admitting. This action space is the same for every flow k.

Each flow k is defined independently of other flows as the tuple

$$\left(\mathcal{N}_k, \left(\boldsymbol{W}_k^a \right)_{a \in \mathcal{A}}, \left(\boldsymbol{R}_k^a \right)_{a \in \mathcal{A}}, \left(\boldsymbol{P}_k^a \right)_{a \in \mathcal{A}} \right),$$

where

- $\mathcal{N}_k := \{1, 2, \ldots, N_k\}$ is the *state space*, i.e., a set of possible congestion windows flow k can set;
- $\boldsymbol{W}_k^a := \left(\boldsymbol{W}_{k,n}^a \right)_{n \in \mathcal{N}_k}$, where $\boldsymbol{W}_{k,n}^a$ is the expected one-period capacity consumption (in number of packets), or *work* required by flow k at state n if action a is decided at the beginning of a period; in particular, $\boldsymbol{W}_{k,n}^0 := 0$ and $\boldsymbol{W}_{k,n}^1 := n$;
- $\boldsymbol{R}_k^a := \left(\boldsymbol{R}_{k,n}^a \right)_{n \in \mathcal{N}_k}$, where $\boldsymbol{R}_{k,n}^a$ is the expected one-period generalized α-fairness or *reward* earned by flow k at state n if action a is decided at the beginning of a period; in particular $\boldsymbol{R}_{k,n}^0 := 0$ and

$$\boldsymbol{R}_{k,n}^1 := \begin{cases} \frac{(1+n)^{1-\alpha} - 1}{1 - \alpha}, & \text{if } \alpha \neq 1, \\ \log(1 + n), & \text{if } \alpha = 1; \end{cases}$$

- $\boldsymbol{P}_k^a := (p_{k,n,m}^a)_{n,m \in \mathcal{N}_k}$ is the flow-k stationary one-period *state-transition probability matrix* if action a is decided at the beginning of a period, i.e., $p_{k,n,m}^a$ is the probability of moving to state m from state n under action a; in particular, $p_{k,n,m}^1 = 1$ iff $m = \min \{n + 1, N_k\}$(representing additive increase) and $p_{k,n,m}^0 = 1$ iff $m = \max \{\lfloor \gamma_k \cdot n \rfloor, 1\}$ (representing multiplicative decrease); the remaining probabilities are zero.

The dynamics of flow k is thus captured by the *state process* $X_k(\cdot)$ and the *action process* $a_k(\cdot)$, which correspond to state $X_k(t) \in \mathcal{N}_k$ and action $a_k(t) \in \mathcal{A}$, respectively, at all time epochs $t \in \mathcal{T}$. The states $n \in \mathcal{N}_k$ denote possible levels of the sending rate. In particular $\boldsymbol{W}_n^{\text{sent}} := n$ can therefore be interpreted as the bandwidth capacity the flow requires for complete transmission at the current period.

The schematic behavior of the AIMD flow as a Markov chain is shown in Fig. 2, where "ACK" represents a congestion-free delivery of the flow packets to the receiver (positive acknowledgments) and "NACK" represents a congestion-experienced transmission (negative acknowledgments). Notice that the evolution is completely deterministic; this can be extended to stochastic evolution like for instance in [26].

4.2. Multi-flow optimization problem

The flows dynamics is as follows (see Fig. 1). At epoch t, the sender of each flow $k \in \mathcal{K}$ sets its state $X_k(t)$ (that depends on whether the previous-epoch workload was transmitted, given by the positive or negative acknowledgement of the receiver of flow k sent back to the sender in the previous period) and sends the workload of $\boldsymbol{W}_{k,X_k(t)}^{\text{sent}}$ packets to the bottleneck router. However, only $0 \leqslant \boldsymbol{W}_{k,X_k(t)}^{a_k(t)} \leqslant \boldsymbol{W}_{k,X_k(t)}^{\text{sent}}$ packets are allowed to queue in the buffer (admitted) for being transmitted. The transmitted packets arrive to the receiver of flow k, who obtains the fairness (reward) $\boldsymbol{R}_{k,X_k(t)}^{a_k(t)}$. If the router admitted the flow,
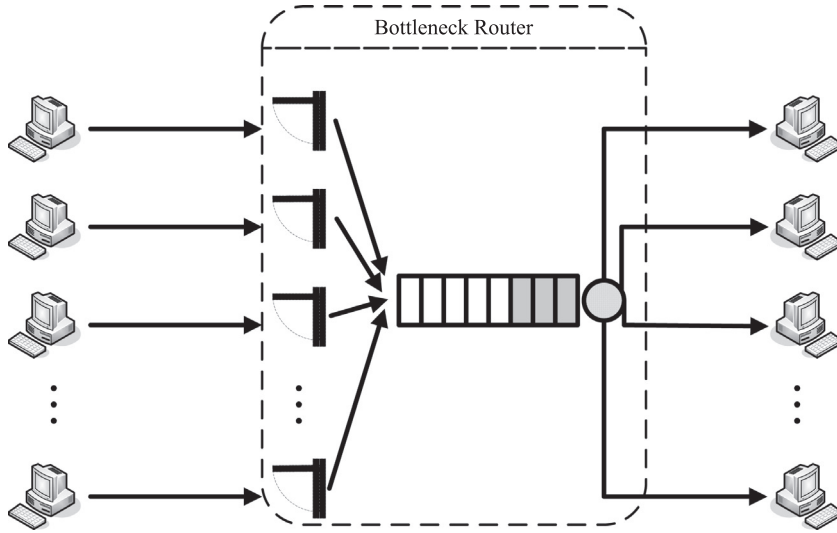
Fig. 1. A scheme of K flows sharing a bottleneck router.

then the receiver sends positive acknowledgement back to the sender; otherwise negative acknowledgment is sent. According to this, the sender sets its next-epoch state $X_k(t+1)$ and repeats the process.

The congestion avoidance decision at the router is taken in the following way. At epoch t the router observes the states (congestion windows) $X_k(t)$ of all flows $k \in \mathcal{K}$. Based on that it decides the flow actions $a_k(t)$ (which may be viewed to be taken in virtual gates, as illustrated in Fig. 1), instantaneously appends (in FIFO order) $\mathcal{W}_{k,X_k(t)}^{a_k(t)}$ packets of each flow k to the buffer, and transmits (in FIFO order) W packets (or all the packets if there are less than W packets in the buffer) during the period.

To summarize, the senders make no decisions and therefore the flows dynamics can be modeled as a Markov chain.

At this moment we present a generic formulation of the congestion control optimization problem. Let Π be the set of all history-dependent randomized policies. Denote by the symbol $\mathbb{E}_{\boldsymbol{n}}^\pi$ the conditional expectation given that the initial conditions are $\boldsymbol{n} := (n_k)_{k\in\mathcal{K}}$, and the policy applied is $\pi \in \Pi$. The router controller's problem to solve is under the discounted criterion (if $\beta < 1$)

$$\max_{\pi\in\Pi} \mathbb{E}_{\boldsymbol{n}}^\pi \left[\sum_{t=0}^{\infty}\sum_{k\in\mathcal{K}} \beta^t \boldsymbol{R}_{k,X_k(t)}^{a_k(t)} \right] \tag{3}$$

$$\text{s.t. } \mathbb{E}_{\boldsymbol{n}}^\pi \left[\sum_{t=0}^{\infty}\sum_{k\in\mathcal{K}} \beta^t \boldsymbol{W}_{k,X_k(t)}^{a_k(t)} \right] \leqslant \frac{\overline{W}}{1-\beta} \tag{4}$$

$$\sum_{k\in\mathcal{K}} \boldsymbol{W}_{k,X_k(t)}^{a_k(t)} \leqslant B, \quad \text{for all } t \in \mathcal{T}. \tag{5}$$

The problem can also be formulated under the time-average criterion. It is known by [6, Lemma 7.1.8] that the optimal policy for the time-average criterion can be obtained by the limit $\beta \to 1$ of the optimal policy for the discounted problem.

The virtual capacity seen as the target time-average router throughput $\overline{W}$ is equal to the bandwidth-delay-product $C \times RTT$, that is, the maximum number of packets that can be served in one slot. We note that an analogous constraint (4) formulation was used in [27]. To avoid the trivial problem of underloaded router, we assume that for policy π that always admits we have

$$\mathbb{E}_{\boldsymbol{n}}^\pi \left[\sum_{t=0}^{\infty}\sum_{k\in\mathcal{K}} \beta^t \boldsymbol{W}_{k,X_k(t)}^{\text{sent}} \right] > \frac{\overline{W}}{1-\beta}.$$

5. Decomposition of the multiple-flows problem

The problem (3)–(5) is difficult to solve due to the sample path constraint (5). One possibility for relaxing the problem is to assume that the buffer space B is infinite, so that the constraint (5) is trivially fulfilled. Another possibility is to relax that constraint as did [7], by requiring it only in discounted expectation, i.e.,

$$\mathbb{E}_{\boldsymbol{n}}^\pi \left[\sum_{t=0}^{\infty}\sum_{k\in\mathcal{K}} \beta^t \boldsymbol{W}_{k,X_k(t)}^{a_k(t)} \right] \leqslant \frac{B}{1-\beta}.$$

However, such a constraint is weaker than (4), because $B \geqslant \overline{W}$, since in our model B is the total number of packets
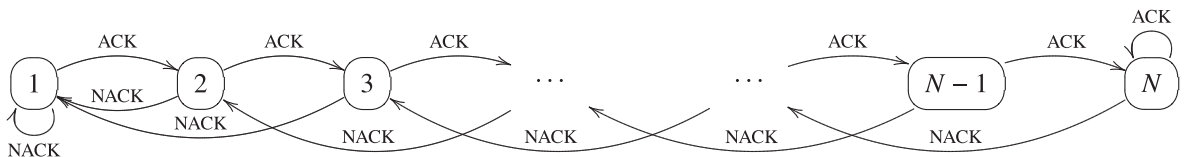

Fig. 2. A model of an AIMD flow as a Markov chain. The arrows represent one-period transitions among the states $1, 2, \ldots, N$ after a congestion-free (ACK) and a congestion-experienced (NACK) transmission.

that the buffer can handle in one RTT, whereas $\overline{W}$ corresponds to the number of packets that can be served in one RTT.

Either of these two relaxation possibilities results in omitting the constraint (5). Thus, we end up with a problem formulation (3) and (4), which is analogous to the Whittle relaxation of the multi-armed restless bandit problem [7].

The standard solution of such a formulation is by solving for each v the Lagrangian relaxation of (3) and (4), which is

$$\max_{\pi \in \Pi} \mathbb{E}_{\boldsymbol{n}}^{\pi} \left[\sum_{t=0}^{\infty} \sum_{k \in \mathcal{K}} \beta^t \left(\boldsymbol{R}_{k,X_k(t)}^{a_k(t)} - v\boldsymbol{W}_{k,X_k(t)}^{a_k(t)} \right) \right] + v\frac{\overline{W}}{1-\beta} \qquad (6)$$

where v is the Lagrangian parameter that can be interpreted as a per-packet *transmission cost*. The Lagrangian theory assures that there exists v^*, for which the Lagrangian relaxation (6) achieves optimum of (3) and (4). Since for any fixed v the flows are independent and the second term of (6) is constant, we can decompose (6) into K individual-flow problems.

Proposition 1. *Let Π_k be the set of all history-dependent randomized policies for flow k, and individual-flow policies $\pi_k^* \in \Pi_k$ such that they form the joint policy $\pi^* \in \Pi$. If for a given parameter v, each policy π_k^* for $k \in \mathcal{K}$ optimizes the individual-flow problem*

$$\max_{\pi_k \in \Pi_k} \mathbb{E}_{n_k}^{\pi_k} \left[\sum_{t=0}^{\infty} \beta^t \left(\boldsymbol{R}_{k,X_k(t)}^{a_k(t)} - v\boldsymbol{W}_{k,X_k(t)}^{a_k(t)} \right) \right], \qquad (7)$$

then π^* optimizes the multi-flow problem (6).

In Section 6 we will find an optimal solution to such a v-parameter problem in terms of flow- and state-dependent *Whittle indices* $v_{k,n}$, which in our setting can be interpreted as *transmission indices*. If the optimal transmission cost v^* is known, then these indices define the following optimal policy for problem (3) and (4): "At each time slot admit all the flows of actual-state transmission index greater than the transmission cost v^* and reject the remaining flows".

Since in practice v^* is typically unknown, the buffer space is finite, and it is desirable to have work-conserving transmission in order to increase bandwidth utilization, we use the transmission indices to define practically feasible policy for problem (3)–(5):

Heuristic Policy: In every slot order the flows in decreasing order with respect to their current indices, and admit the flows until reaching the constraint (5).

Following the results in the literature, we expect that our heuristic policy will be asymptotically optimal as the number of flows and buffer capacity grow to infinity [10].

In Section 7 we will develop a heuristic admission control policy for a TCP/IP network at packet level and study its performance with NS-3 simulations. In this case the index of each packet represents the admission priority into the buffer. If the buffer is full, the packet in the queue with smallest transmission index will be dropped.

6. Index policies for single-flow subproblems

The single flow admission control subproblem can be optimally solved by means of index policies under some index-existence conditions [28]. In a particular case, the optimal policy is of threshold type (if it is optimal to reject the flow under certain congestion window, then it is optimal to do the same also under any higher congestion window). However, depending on the parameters, threshold policies may not be optimal, as will be illustrated in Section 6.2. Therefore, in the rest of the paper we will obtain index policies only numerically, and the investigation whether index policies can be characterized by index values in closed form is left out of this paper. (This is indeed possible in some cases, see, e.g., [26,29]).

Since no decisions are taken by the sender, congestion control is implemented in the router. The router decides whether the incoming flow in state n_k should be admitted (and transmitted) (achieved by action $a_k(t) = 1$ of transmitting $\boldsymbol{W}_{n_k}^1 := \boldsymbol{W}_{n_k}^{\text{sent}}$ packets), or rejected (action $a_k(t) = 0$ of transmitting 0 packets). The difference of receiver rewards and transmission costs (i.e., $\boldsymbol{R}_{n_k}^0 - v\boldsymbol{W}_{n_k}^0$ and $\boldsymbol{R}_{n_k}^1 - v\boldsymbol{W}_{n_k}^1$) will be henceforth called the *net reward* under transmission cost v.

To summarize, the (unconstrained) MDP problem of the single AIMD flow addressed in this section under both the discounted and time-average criteria is defined as follows:

- *State space:* is $\mathcal{N}_k$.
- *Actions: admitting rejecting* are available in each state.
- *Dynamics if admitting:* If the flow is in state n_k and the flow is admitted at a given period, then during that period it generates net reward $\boldsymbol{R}_{n_k}^1 - v\boldsymbol{W}_{n_k}^1$ and the flow moves to state $n_k + 1$ (or remains in N_k, if $n_k = N_k$) for the next period.
- *Dynamics if rejecting:* If the flow is in state n_k and the flow is rejected at a given period, then during that period it generates net reward $\boldsymbol{R}_{n_k}^0 - v\boldsymbol{W}_{n_k}^0$ and the flow moves to state $n_k - 1$ (or remains in n_k if $n_k = 1$) for the next period.

To evaluate a policy π under the β-discounted criterion, we consider the following two measures. Let $\mathbb{W}_i^{\pi} := \mathbb{E}_i^{\pi} \left[\sum_{t=0}^{\infty} \beta^t \boldsymbol{W}_{X(t)}^{a(t)} \right]$ be the *expected total β-discounted bandwidth utilization* if starting from state i under policy π. For convenience, we will also call $\mathbb{W}_i^{\pi}$ the expected total β-discounted work, since the bandwidth utilization can be seen as the work performed by the router in order to transmit the flow. Analogously we denote by $\mathbb{R}_i^{\pi} := \mathbb{E}_i^{\pi} \left[\sum_{t=0}^{\infty} \beta^t \boldsymbol{R}_{X(t)}^{a(t)} \right]$ the *expected total β-discounted reward* if starting from state i under policy π.

The objective (7) is for each transmission cost v,

$$\max_{\pi \in \Pi} \mathbb{R}_i^{\pi} - v\mathbb{W}_i^{\pi}. \qquad (8)$$

We will address this problem in the following subsections, noting that an optimal solution for the time-average variant is obtained in the limit $\beta \to 1$.

6.1. Threshold policies and indexability

Since for finite-state MDPs there exists an optimal stationary policy independent of the initial state [6], we narrow our focus only to those policies and represent them via *admission sets* $\mathcal{S} \subseteq \mathcal{N}$. In other words, a policy $\mathcal{S}$ prescribes to admit the flow in states in $\mathcal{S}$ and to reject the flow in states in $\mathcal{S}^\complement := \mathcal{N}/\mathcal{S}$.

We will therefore write $\mathbb{R}_i^{\mathcal{S}}$ and $\mathbb{W}_i^{\mathcal{S}}$ for the expected total β-discounted reward and work, respectively, under policy $\mathcal{S}$ starting from initial state i. Then, formulated for initial state i, the optimization problem is the following combinatorial problem

$$\max_{\mathcal{S} \subseteq \mathcal{N}} \mathbb{R}_i^{\mathcal{S}} - v\mathbb{W}_i^{\mathcal{S}}. \tag{9}$$

In order to solve this problem, we will be interested in two related structural properties, which are *optimality of threshold policies* and *indexability*, formally defined below.

Definition 1 (*Optimality of Threshold Policies*). We say that problem (9) is *optimally solvable by threshold policies*, if for every real-valued v there exists threshold state $n \in \mathcal{N}_k \cup \{0\}$ such that threshold policy admitting the flow in states $\mathcal{S}_{N_k}^{(n)} := \{m \in \mathcal{N}_k : m \leqslant n\}$ and rejecting otherwise is optimal for problem (9).

Of our interest will be the index proposed in [7], which often furnishes a nearly-optimal solution, and typically recovers the optimal index rule if such exists. We adopt the definition of indexability from [30].

Definition 2 (*Indexability*). We say that v-parameter problem (9) is *indexable*, if there exist unique values $-\infty \leqslant v_{k,n} \leqslant \infty$ for all $n \in \mathcal{N}_k$ such that the following holds for every state $n \in \mathcal{N}_k$:

1. if $v_{k,n} \geqslant v$, then it is optimal to admit flow k in state n,
2. if $v_{k,n} \leqslant v$, then it is optimal to reject flow k in state n.

The function $n \mapsto v_{k,n}$ is called the *(Whittle) index*, and $v_{k,n}$'s are called the *(Whittle) index values*.

An immediate consequence of the two definitions is formulated in the following previously known result.

Proposition 2. *If problem* (9) *is indexable and the index is nonincreasing, i.e.,* $v_{k,1} \geqslant v_{k,2} \geqslant \cdots \geqslant v_{k,N_k}$, *then problem* (9) *is optimally solvable by threshold policies. Moreover, for a given v the optimal threshold policy is* $\mathcal{S}_{N_k}^{n^*}$ *with* $n^* \in \mathcal{N}_k \cup \{0\}$ *such that* $v_{k,n^*} \geqslant v \geqslant v_{k,n^*+1}$ *(defining* $v_{k,0} := -\infty, v_{k,N_k+1} := \infty$*).*

For transparency, we limit ourselves in this paper to numerical testing of indexability and computation of the index values, which then allow to conclude about optimality of threshold policies. We have employed an algorithm based on the restless bandit framework [28], which both numerically checks the conditions of existence and calculates the index values, if they exist. It is a one-run algorithm (analogous to parametric simplex method) and in each step it calculates one of the index values. Thus, it performs N_k steps, and the overall computational complexity of the algorithm is $O(N_k^4)$.

We have performed testing of indexability over a large number of flows with different parameters. The algorithm always confirmed that the flow was indexable. These tests give us evidence to conjecture that the flows as defined in this paper are always indexable. However, the complexity of the problem impeded us to find a structure that could be exploited for establishing indexability analytically. We will illustrate the difficulty in the next subsections.

6.2. One-, two-, and three-state flows

In this subsection we provide analytical results showing that optimality of threshold policies depends on values of parameters α, β, N_k, and γ_k.

It is obvious (see, e.g., [26]) that a one-state flow (i.e., $N_k = 1$) is indexable and solvable by threshold policies. The index value for the unique state is $v_{k,1} = \boldsymbol{R}_{k,1}/\boldsymbol{W}_{k,1}$. Similarly [26], proved that a two-state flow (i.e., $N_k = 2$) is indexable and solvable by threshold policies. The index values for the two states are

$$v_{k,1} = \frac{\boldsymbol{R}_{k,1}}{\boldsymbol{W}_{k,1}}, \quad v_{k,2} = \frac{\boldsymbol{R}_{k,2} + \beta(\boldsymbol{R}_{k,2} - \boldsymbol{R}_{k,1})}{\boldsymbol{W}_{k,2} + \beta(\boldsymbol{W}_{k,2} - \boldsymbol{W}_{k,1})}.$$

Finally [26], proved that a three-state flow (i.e., $N_k = 3$) is indexable and solvable by threshold policies, if

$$\boldsymbol{P}_k^0 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

which holds if and only if $\gamma_k < 2/3$. Under such a condition, the index values for the three states are

$$v_{k,1} = \frac{\boldsymbol{R}_{k,1}}{\boldsymbol{W}_{k,1}},$$

$$v_{k,2} = \frac{\boldsymbol{R}_{k,2} + \beta(\boldsymbol{R}_{k,2} - \boldsymbol{R}_{k,1})}{\boldsymbol{W}_{k,2} + \beta(\boldsymbol{W}_{k,2} - \boldsymbol{W}_{k,1})},$$

$$v_{k,3} = \frac{\boldsymbol{R}_{k,3} + \beta(\boldsymbol{R}_{k,3} - \boldsymbol{R}_{k,1}) + \beta^2(\boldsymbol{R}_{k,3} - \boldsymbol{R}_{k,2})}{\boldsymbol{W}_{k,3} + \beta(\boldsymbol{W}_{k,3} - \boldsymbol{W}_{k,1}) + \beta^2(\boldsymbol{W}_{k,3} - \boldsymbol{W}_{k,2})}.$$

In all the above cases, the threshold policies are optimal, because the index values are nonincreasing (due to concavity of the generalized α-fairness function).

It is tedious but straightforward to show (by a detailed inspection of the algorithm that computes the indices) that $\gamma_k \geqslant 2/3$ if and only if the index values for the three states are as follows. If $\alpha < 1$, then

$$v_{k,1} = \frac{\boldsymbol{R}_{k,1}}{\boldsymbol{W}_{k,1}},$$

$$v_{k,2} = \frac{\boldsymbol{R}_{k,2} - \beta\boldsymbol{R}_{k,1}}{\boldsymbol{W}_{k,2} - \beta\boldsymbol{W}_{k,1}},$$

$$v_{k,3} = \frac{\boldsymbol{R}_{k,3} + \beta(\boldsymbol{R}_{k,3} - \boldsymbol{R}_{k,2})}{\boldsymbol{W}_{k,3} + \beta(\boldsymbol{W}_{k,3} - \boldsymbol{W}_{k,2})}.$$

In this case, the threshold policies are optimal, because the index values are nonincreasing. On the other hand, if $\alpha \geqslant 1$, then for some values of β the index values are as above, but for other values of β the index values are

$$v_{k,1} = \frac{\boldsymbol{R}_{k,1}}{\boldsymbol{W}_{k,1}},$$

$$v_{k,2} = \frac{\boldsymbol{R}_{k,2} + \beta(\boldsymbol{R}_{k,3} - \boldsymbol{R}_{k,1}) + \beta^2(\boldsymbol{R}_{k,3} - \boldsymbol{R}_{k,2})}{\boldsymbol{W}_{k,2} + \beta(\boldsymbol{W}_{k,3} - \boldsymbol{W}_{k,1}) + \beta^2(\boldsymbol{W}_{k,3} - \boldsymbol{W}_{k,2})},$$

$$v_{k,3} = \frac{\boldsymbol{R}_{k,3} - \beta^2 \boldsymbol{R}_{k,1}}{\boldsymbol{W}_{k,3} - \beta^2 \boldsymbol{W}_{k,1}}.$$

In this case, we have $v_{k,1} > v_{k,3} > v_{k,2}$, therefore threshold policies are not optimal in general. In particular, threshold policy $\mathcal{S}_{N_k}^{(2)}$ is never optimal.

6.3. General flows

In this subsection we illustrate using numerical results that the structure of optimal policies and the index values depend heavily on values of parameters α, β, N_k, and γ_k. This feature brings complexity to the mathematical analysis of the problem, but on the other hand, it shows that indices nicely capture the nature of different TCP variants under different optimization criteria. Nevertheless, the figures provide insights for better understanding of what an optimal admission control policy is.

As for the basic instance, we set $\alpha := 1$, $\beta := 0.9999$, $\gamma_k := 1/2$ and $N_k := 70$. In Fig. 3 we present index values as a function of congestion window for (a) different values of α, (b) different values of β, and (c)–(d) different values of γ_k. We note that we have observed that different values of N_k may also influence the index values, especially if both β and γ_k are large, but the differences are often not noticeable in the figures, so they are omitted.

First of all, it can be seen in Fig. 3(a) and (b) that as α or β diminish, threshold policies become optimal, due to having the index values nonincreasing, as opposed to the non-monotone (zig–zag) index when both α or β are large. A similar behavior can be observed also for other values of γ_k, which are not reported in this paper. Fig. 3(a) further indicates that index values are decreasing in α and their slopes for a given α are "more convex", so higher α ensures a stronger discouragement of dropping packets belonging to flows with smaller actual congestion window. Note that case $\alpha = 0$ gives all the index values equal to 1, so no flow is prioritized over other ones, which is analogous to the DropTail policy. Fig. 3(b) indicates that index values are decreasing in β, so longer flows get lower priority for transmission over shorter flows with the same actual congestion window.

From Fig. 3(c) and (d) we can learn interesting insights as well. The index functions for $\gamma_k = 0, 0.99$ are smooth and nonincreasing, as well as, rather surprisingly, $\gamma_k = 1/3$. The remaining cases result in "tooth-shaped" index functions, with remarkably different tooth widths (akin to periods) of two ($\gamma_k = 1/2$), three ($\gamma_k = 2/3$), and six ($\gamma_k = 1/6, 5/6$ look similar, with their period point shifted), which may happen because γ_k's are multiples of six in this figure. In spite of that, index values of different γ_k's coincide or come very close to each other at congestion windows that are multiples of these periods. At multiples of six (e.g., 6, 12, ...), the index values are increasing in γ_k (at least over the range 1/3, 1/2, 2/3, 5/6), while at the subsequent points (e.g., 7, 13, ...) the index values are decreasing in γ_k, so the priority ordering of flows with such congestion windows is completely reversed.

We underline that Fig. 3(c) and (d) suggests that the index function for $\gamma_k = 0$ could be a reasonable smooth approximation for the index functions for the remaining values, especially for $\gamma_k = 1/3$. This is of special interest from practical point of view, because the index function for $\gamma_k = 0$ is known in closed-form due to [26]. On the other hand, the same figure also suggests that the index function for $\gamma_k = 0.99$ (which is essentially a birth–death dynamics) is a lower bound for the index functions for the remaining values of γ_k, which is also known in closed-form due to [29]. However, $\gamma_k = 0$ does not necessarily provide an upper bound. Such an upper bound could be obtained by using the myopic index by $\beta = 0$. Note also that as β gets closer to 0, numerical differences between index values for different γ_k's become smaller.

7. Simulations results

In this section we present experimental results from implementing index policy in simulations. We assume that TCP sources include in the header of each packet the index value corresponding to the actual *cwnd*. Based on the mathematical results we define the following heuristic index policy to be implemented in the Internet routers:

Heuristic index policy at packet level: Upon a packet arrival, if the buffer is not full, then admit the packet. Otherwise, drop the packet (either the new one or from the queue) with smallest index value. In case of ties, drop the packet that has been the longest in the queue.

We have employed and modified TCP New Reno in the NS-3 simulator [18] to obtain the results in several scenarios. The main objective of this section is to show implementability of the proposed index policy, to give fundamental insights about its effect, and to evaluate possible gains of these results with the DropTail and RED policies. As the measure of fairness we employ the Jain's fairness index, whose value ranges from 1 (perfect fairness) to $1/K$ in a K-user system [31].

We focus on case $\alpha = 1$, since it was shown that the loss networks (like the current Internet with DropTail) maximizes the aggregate sum of logarithmic utilities of the time-average transmission rates [19]. Note that our approach (with $\alpha = 1$) maximizes the aggregate sum of the time-average logarithmic utilities of the immediate transmission rates.

The access links of each user is 5 Mb/s and the delay is 10 ms. The delay of the bottleneck link in this scenario is 10 ms and the bandwidth capacity of the bottleneck link is 1500 kb/s. The packet size is 576 Bytes. We set $\beta = 0.9999$ which approximates the time-average criterion. We set the maximum value of the congestion window of each of the users to $N_k = 70$, required to compute the index values.

For each of the five scenarios below, we plot the time evolution under the DropTail, RED and index policy of the following elements: the size of the queue in the router buffer, the congestion window of each of the users, and (only for index policy) the index value of each of the users.
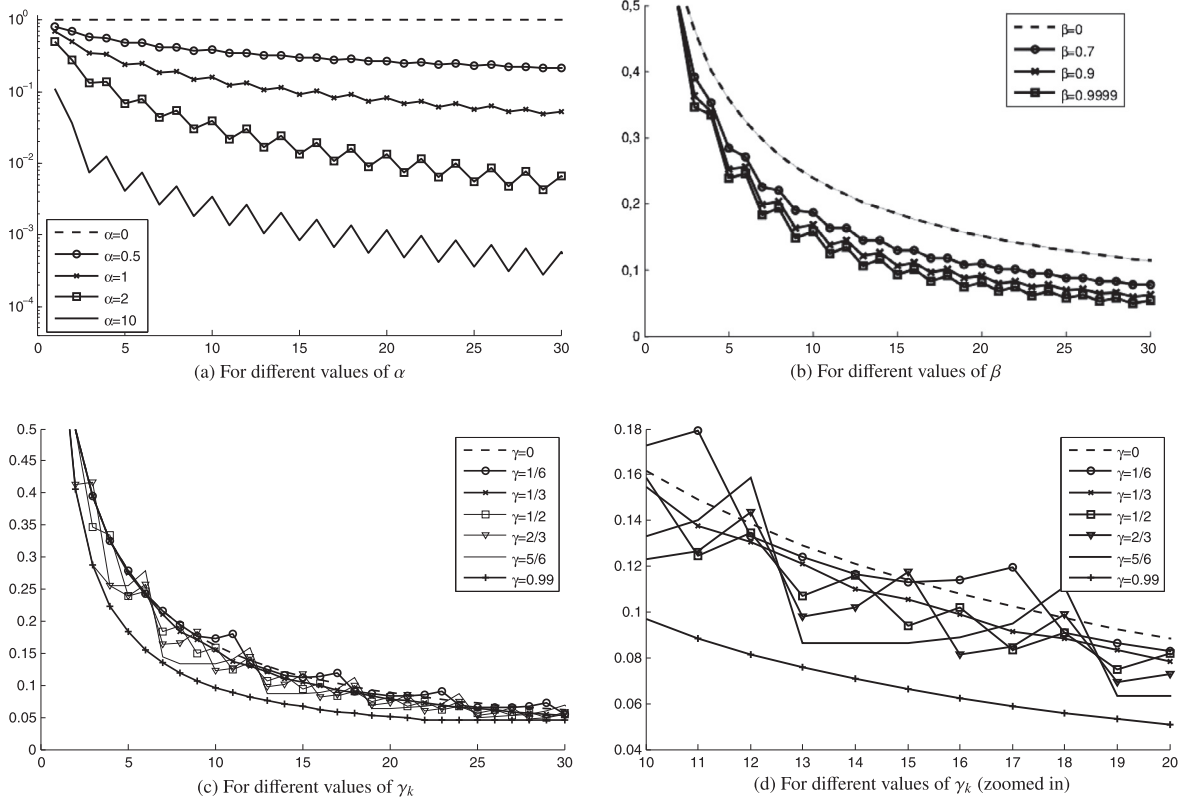
Fig. 3. Index values as a function of congestion window for different values of parameters.

In all figures of the simulation section, user 1 is depicted with a black and thin line, and user 2 with a black and thick solid line.

The results are summarized in Table. 1.

7.1. Scenario 0 (Baseline): Two symmetric users

As a baseline scenario, we consider two symmetric (equal) users that are sending data to a server through a bottleneck router. Each user $k = 1,2$ is halving her congestion window, i.e., $\gamma_k = 1/2$.

The buffer size is set to the Bandwidth-Delay product of a single user,

$$\frac{1500 \cdot 10^3 \text{ b/s} \cdot 2 \cdot (10^{-2} \text{ s} + 10^{-2} \text{ s})}{576\text{B} \cdot 8\text{b/B}} \approx 13.$$

We present the evolution of the congestion window and the size of the queue of the router in time for the DropTail case in Fig. 4a. As expected due to the well-known phenomenon, the two users are completely synchronized.

We depict the evolution of the congestion window and the size of the queue of RED policy in Fig. 4b. We observe that the users get unsynchronized and in this instance a periodic behavior of the congestion window is not achieved because the packets are dropped randomly.

We show the evolution of the indices, the congestion window and the size of the queue of the router in time for the index policy in Fig. 4c. Interestingly, we can observe that users become ideally unsynchronized and as a

consequence the utilization of the buffer increases. As it can be seen from Table 1, the throughput increases by 1.1% and 0.6% with respect to Droptail and RED policies. However, the user fairness remains essentially the same as in DropTail policy, but it improves user fairness comparing with RED.

7.2. Scenario 1: Reducing buffer size

In this scenario we analyze the influence of setting a smaller buffer size in the bottleneck queue for index policies, DropTail and RED. To investigate this effect, we set the buffer size of the router to 6.

As we can see in Fig.5a, with DropTail users are synchronized and the buffer is empty more time than in the previous scenario.

In Fig. 5b, we observe again that the congestion window with RED does not change periodicly.

In the case of the index policy (see Fig. 5c) the users are desynchronized, and as a result the number of delivered packets is larger than DropTail policy and also higher than RED. We note that the fairness among users does not change much in this case. We can also observe that the utilization increases by 7.3% and 6.7% with respect to DropTail and RED, but the RTT increases only 2.3% and 2%, respectively.

The main conclusion of this scenario is that a buffer size smaller than the bandwidth-delay product, the throughput of index policies is larger than DropTail and RED.

Table 1

Utilization of the bottleneck queue, the Jain's fairness index between users (over the 20 s interval), the Mean Queue Size (in number of packets) of the bottleneck buffer and the RTT of a connection of one user (in ms) in the simulations for different scenarios. In parentheses, the improvement of index policy with respect to DropTail and RED.

Scenario	Policy	Utilization	User fairness	Mean queue size	RTT
0	DropTail	97.14%	0.999999	6.98	61.4
0	RED	97.63%	0.982572	6.11	58.77
0	Index	98.22% (+1.1%,+0.6%)	0.999798 (−0.1%,+1.8%)	8.07	64.7 (+5.2%,+10.2%)
1	DropTail	81.16%	1.000000	2.41	47.4
1	RED	81.71%	0.998233	2.54	47.8
1	Index	88.45% (+7.3%,+6.7%)	0.999837 (−0.1%,+0.1%)	2.89	48.8 (+2.3%,+2%)
2	DropTail	90.84%	0.786535	7.1	61.8
2	RED	92.74%	0.911862	6.04	58.5
2	Index	95.37% (+4.5%,+2.6%)	0.962784 (+22.3%,+5.5%)	7.34	62.5 (+1.1%,6.8%)
3	DropTail	86.36%	0.739875	6.63	60.3
3	RED	94.08%	0.821966	6.31	59.4
3	Index	96.86% (+10.5%,+2.8%)	0.917895 (+24.5%,+11.6%)	6.84	62.5 (+3.6%,+5.2%)
4	DropTail	93.84%	0.765318	6.11	See Section 7.4
4	RED	91.6%	0.89428	5.81	
4	Index	94.97% (+1.1%,+3.3%)	0.929756 (+21.4%,+3.9%)	6.37	

7.3. Changing multiplicative decrease factor

We illustrate the inter-protocol properties of the index policy. In Scenario 2 we change the multiplicative decrease factor of user 1 to $\gamma_1 = 0$, and in Scenario 3 to $\gamma_1 = 0.9$.

7.3.1. Scenario 2:

In this setting user 1 is conservative comparing to user 2, and reinitializes the congestion window to 1 every time a packet is lost, i.e., $\gamma_1 = 0$.

We observe in Fig. 6a that with DropTail the congestion window of user 2 is consistently bigger. This implies that the number of delivered packets by user 2 is much larger, as can be seen in Table 1.

In Fig. 6b we see that once again RED causes an unsynchronized behavior of the users. Besides, RED policy improves fairness among users with respect to DropTail, as we show in Table 1.

With the index policy the congestion window of user 1 is reduced less often than user 2 (see Fig. 6c). In this case, users are completely desynchronized with the property of higher Jain's fairness value, which improves with respect to DropTail and RED by 22.3% and 5.5%, respectively. At the same time, the utilization of index policy is larger, so that the total number of delivered packets is increased by 4.5% and 2.6% comparing with DropTail and RED, while the RTT of one user increases by 1.1% and 6.8%, respectively.

From this scenario, we conclude that index policy improves user fairness and throughput for users with different TCP models.

7.3.2. Scenario 3:

User 1 is now much more aggressive than user 2, since it barely reduces its congestion window in response to a congestion event, i.e., $\gamma_1 = 0.9$.

With DropTail we observe that user 1 has a significantly bigger cwnd in every moment (see Fig. 7a). This illustrates that DropTail is not fair in this setting.

In Fig. 7b we see that RED policy is not fair either, because the congestion window of the user 1 is always higher than the congestion window of user 2.

On the other hand, we see in Fig. 7c that with the index policy the difference in the congestion window between both users is not big, which results in a significantly better Jain's fairness index value, which is increased by 24.5% and 11.6% comparing with DropTail and RED, respectively. At the same time, the utilization is increased by 10.5% and 2.8% with respect to DropTail and RED and the RTT of one user increases by 3.6% and 5.2%.

The main contribution of this scenario consists on showing that for index policy the user fairness is larger than DropTail and RED when a user is more aggressive than the other.

7.4. Scenario 4: Changing propagation delay

In this scenario we modify the propagation delay of the access link of user 1 and we set it to 50 ms.

In this setting, the congestion window of both users under DropTail, RED and the index policy becomes very similar. The main difference is that with the index policy, the users gets again "ideally" unsynchronized, while the behavior of the congestion window in DropTail and RED is more random. As we can see in the Jain's index in Table 1, the user fairness for index policy is 24.5% and 11.6%, larger than DropTail and RED policies, respectively.

We explain the differences of the RTT of each user in DropTail, RED and index policy. The values of the RTT of the user 1 for DropTail, RED and index policy are 138.8 ms, 137.8 ms and 141 ms, respectively. For this user, the increasing of the RTT in index policy with respect to DropTail and RED is 1.5% and 2.3%. On the oher hand, the values of the round-trip-time of the user 2 for Droptail, RED and index policy are 58.8 ms, 57.8 ms and 61 ms respectively, so the increase of the RTT of this user with index policy with respect to DropTail and RED is 3.7% and 5.5%.
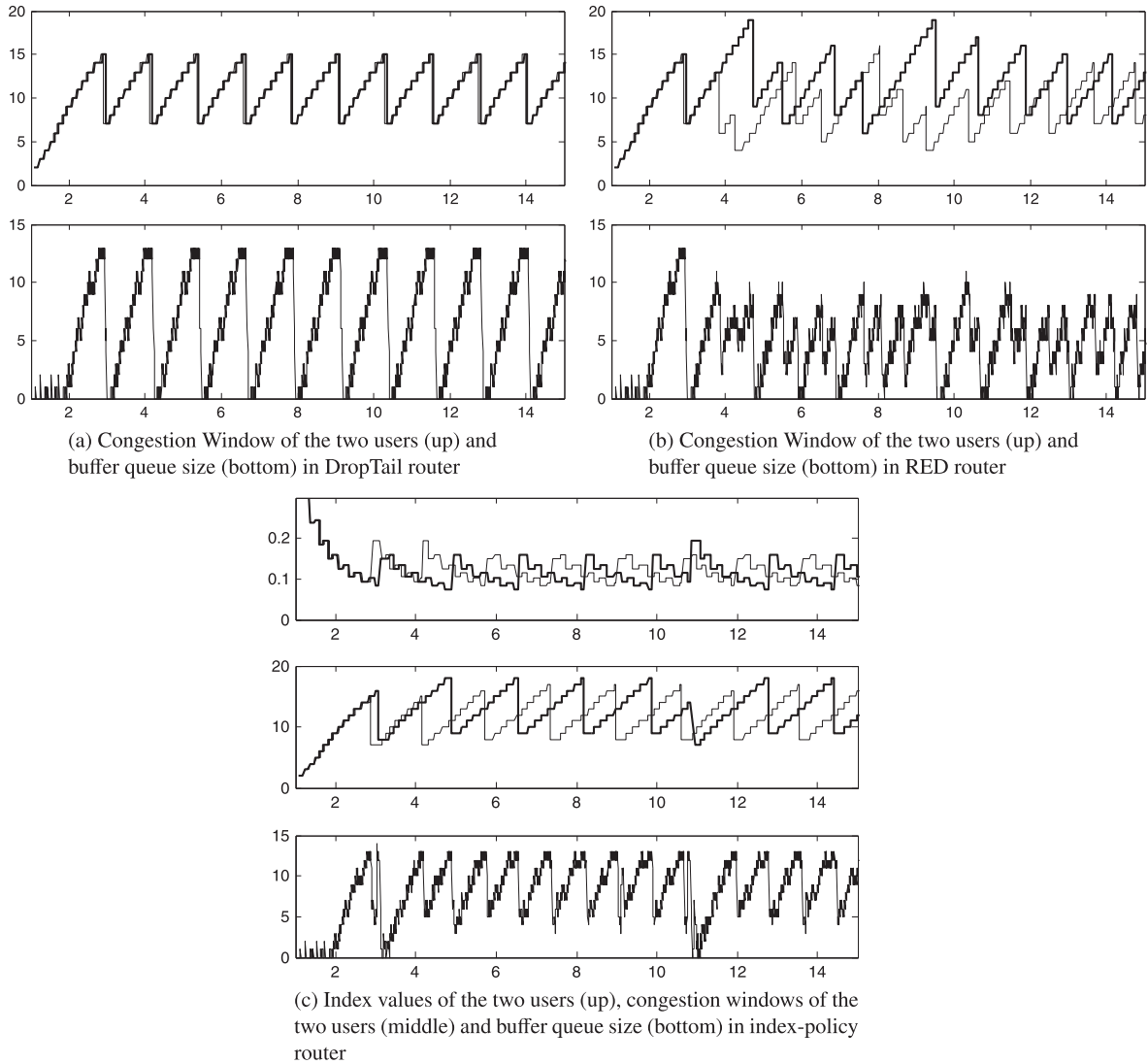
(a) Congestion Window of the two users (up) and buffer queue size (bottom) in DropTail router

(b) Congestion Window of the two users (up) and buffer queue size (bottom) in RED router

(c) Index values of the two users (up), congestion windows of the two users (middle) and buffer queue size (bottom) in index-policy router

Fig. 4. Scenario 0 (Baseline): Simulation of a bottleneck with two equal users with standard TCP ($\gamma_1, \gamma_2 = 0.5$).

According to the results obtained in this scenario, we conclude that index policy has the property of RTT fairness and higher throughput than DropTail and RED policies (see Fig. 8).

8. Discussion

8.1. Cooperative sources

The ideal implementation of the proposed index policy is that the end-users (sources) are cooperative, as the TCP sources are today, and they write the index in the packet header of every packet sent. In this way, there is no computational overhead, because the index values can be computed offline for the TCP variant the source has implemented, and the indices can be stored in a lookup table, where an index is assigned to every possible value of congestion window. This table needs to be accessed only when (the integer part of) the congestion window changes, which happens approximately once per RTT, which is enough to access the data. While (the integer part of) the congestion window does not change, all outgoing packets have the same index, so the lookup table does not need to be accessed.

From the point of view of the router, an efficient implementation seems to be to maintain a dynamic table of indices of all the buffered packets, with the pointer to the packet of lowest index (in FIFO order). That is, upon arrival of every packet, its index is memorized, and upon departure of every packet, it is removed from the list. This dynamic table of indices can thus be maintained separately from the regular buffer operation. When there is a requirement to drop/mark a packet, the pointer indicates which packet it should be applied to. The index corresponding to the pointed packet can be interpreted as the *congestion*
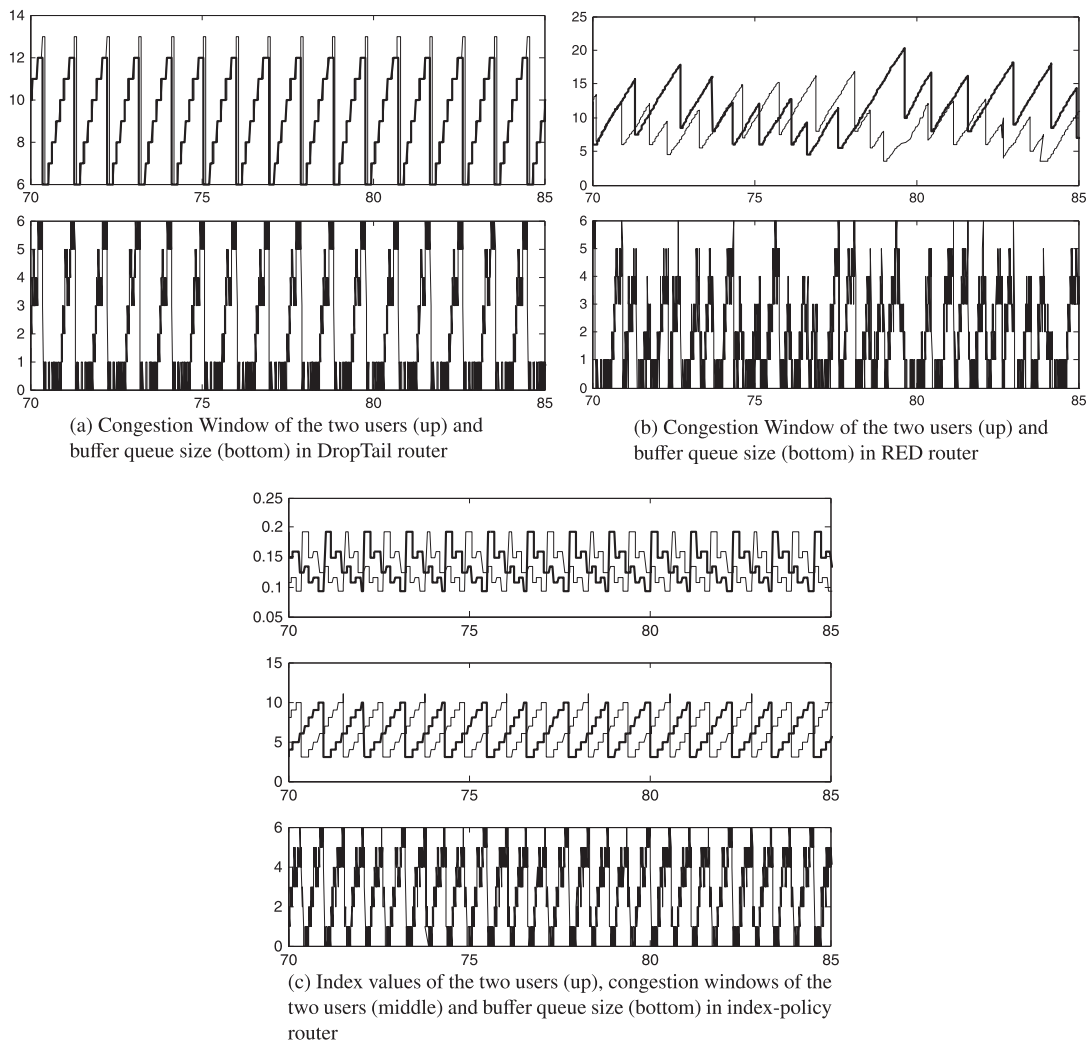
(a) Congestion Window of the two users (up) and buffer queue size (bottom) in DropTail router

(b) Congestion Window of the two users (up) and buffer queue size (bottom) in RED router

(c) Index values of the two users (up), congestion windows of the two users (middle) and buffer queue size (bottom) in index-policy router

Fig. 5. Scenario 1: Simulation of a bottleneck link with two equal users with standard TCP and a buffer size smaller than the bandwith-delay product.

price (the higher the congestion price, the more congested the router is), which changes dynamically as the buffer utilization evolves. Note further that such an implementation implies that the router *does not need to identify flows*.

In the following paragraphs we discuss several important features not explicitly covered by our model.

Short flows: Our model assumes that the source has always data to transmit, so that the connection is permanent. This is a common assumption in the TCP literature, however, in today practice the flows are often short, intertwined by long periods without transmission. It is straightforward to extend our model to such flows: if the exact flow size is known by the source, then the MDP model will have a two-dimensional state (including the remaining number of packets alongside the congestion window); if (more likely) the exact flow size is unknown, e.g., because of the user impatience or incomplete information, it could be modeled for instance by incorporating discounting. In either case, the indices are computed offline and the implementation would not be altered.

Timeouts and slow-start: Our model can be extended in order to include the slow-start mode of the TCP algorithm. A second Markov chain representing the slow-start mode must be considered so that we can interchange from one chain to the other, when the dynamics of the flow change from slow-start mode to congestion avoidance mode and viceversa. Although we are able to generalize the model to include slow-start, we leave it out of the paper, as we believe it would not affect the index values significantly.

Many-flows scalability: The index policy requires packet header inspection, where the index value is read apart from the data (such as destination) currently read. Implemented naively by reading before buffering, this may cause delays and decrease the input capacity of the router. However, it may be perfectly acceptable to read the index once the packet has been buffered, thus avoiding any delays in processing.

Another possibility is to read indices of only some of the packets (*the power of few*), which is likely to identify a packet with sufficiently low index (though maybe not the
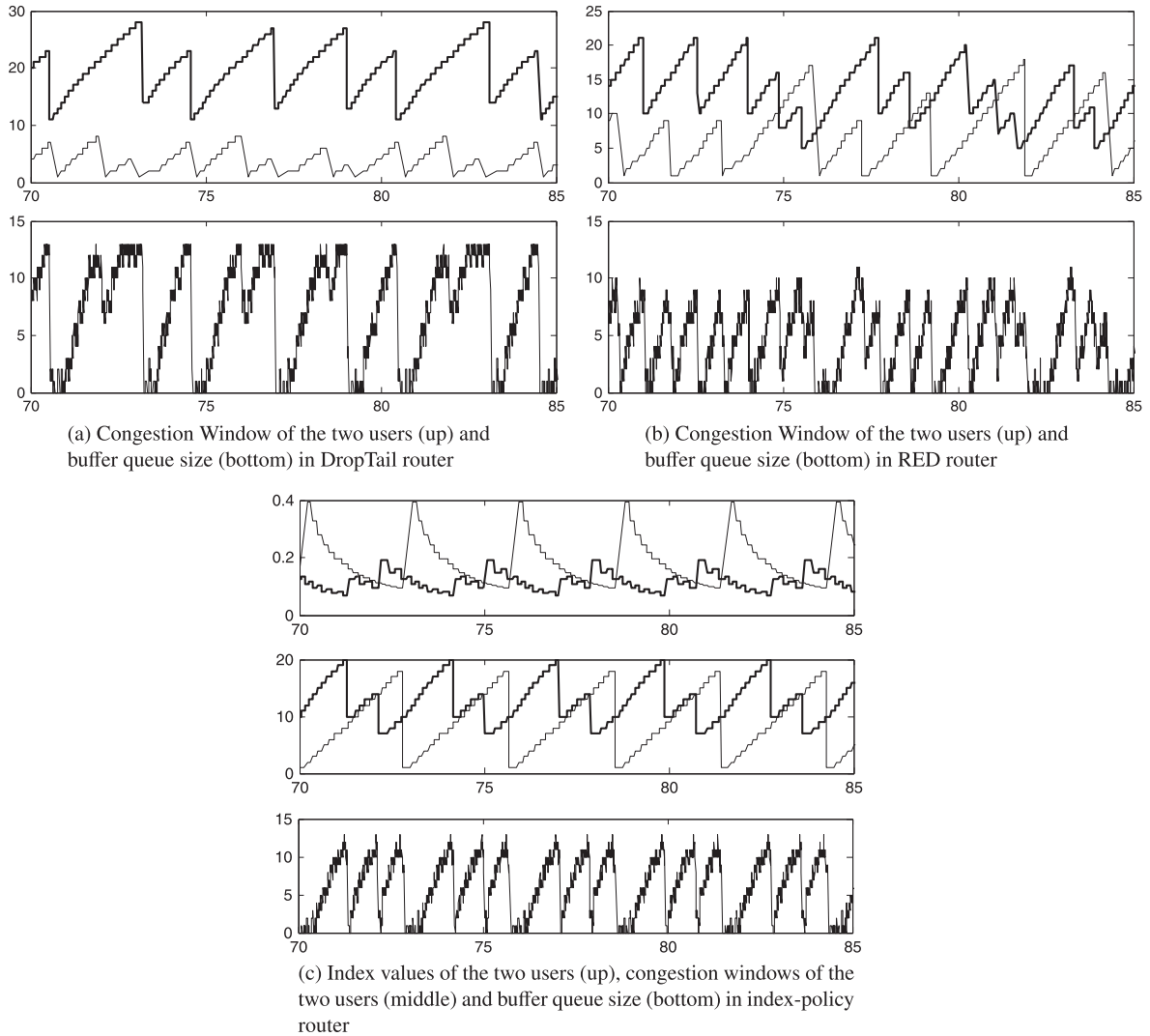
(a) Congestion Window of the two users (up) and buffer queue size (bottom) in DropTail router

(b) Congestion Window of the two users (up) and buffer queue size (bottom) in RED router

(c) Index values of the two users (up), congestion windows of the two users (middle) and buffer queue size (bottom) in index-policy router

Fig. 6. Scenario 2: Simulation of a bottleneck link with two users, where User1 has restarting TCP ($\gamma_1 = 0$) and User2 has standard TCP ($\gamma_2 = 0.5$).

lowest and maybe not following the FIFO tie-breaking, which may not be essential).

Multiple links: In this paper we consider a single-bottleneck network, i.e., other routers on the path are implicitly assumed congestion-free. We believe that this assumption is reasonable because the congestion of a flow when it traverses multiple links typically occurs only in one router, often at the edge or at the server (see [8] for details). Moreover, from the point of view of the source that computes the index, it is indistinguishable where exactly the congestion occurs or whether there are multiple bottlenecks on the route.

From the point of view of multiple-bottleneck routers, the situation is slightly different. If a given router believes that further downstream there is a more congested router, it may be worth to drop/mark a packet earlier than usual (e.g., set the buffer size smaller or increase the dropping/marking probability in AQM). Another possibility is to car-

ry this information further downstream by rewriting the index of the packet: decreasing the index by the current congestion price at every router on the path. Thus, the packet is likely to traverse the whole congested route without congestion notification only if its index is high (i.e., it is efficient to transmit it) relatively to the sum of the congestion prices in all the routers on the path, and relatively to other packets. Thus, the dropped/marked packet would be of the flow that most significantly decreases the congestion in the network by reacting to a congestion notification.

Interaction with IPSEC: The index policy requires including the index value in the IP header of the packets. In the case that IPSEC encapsulation is utilized, a portion of the IP packet is, at least, encrypted. In case that only the payload of the IP packet is encrypted, then the routing will not be altered, and our method can be applied directly. However, if all the IP packet is encrypted, then the packet is encapsulated in a new IP header that will be read by the
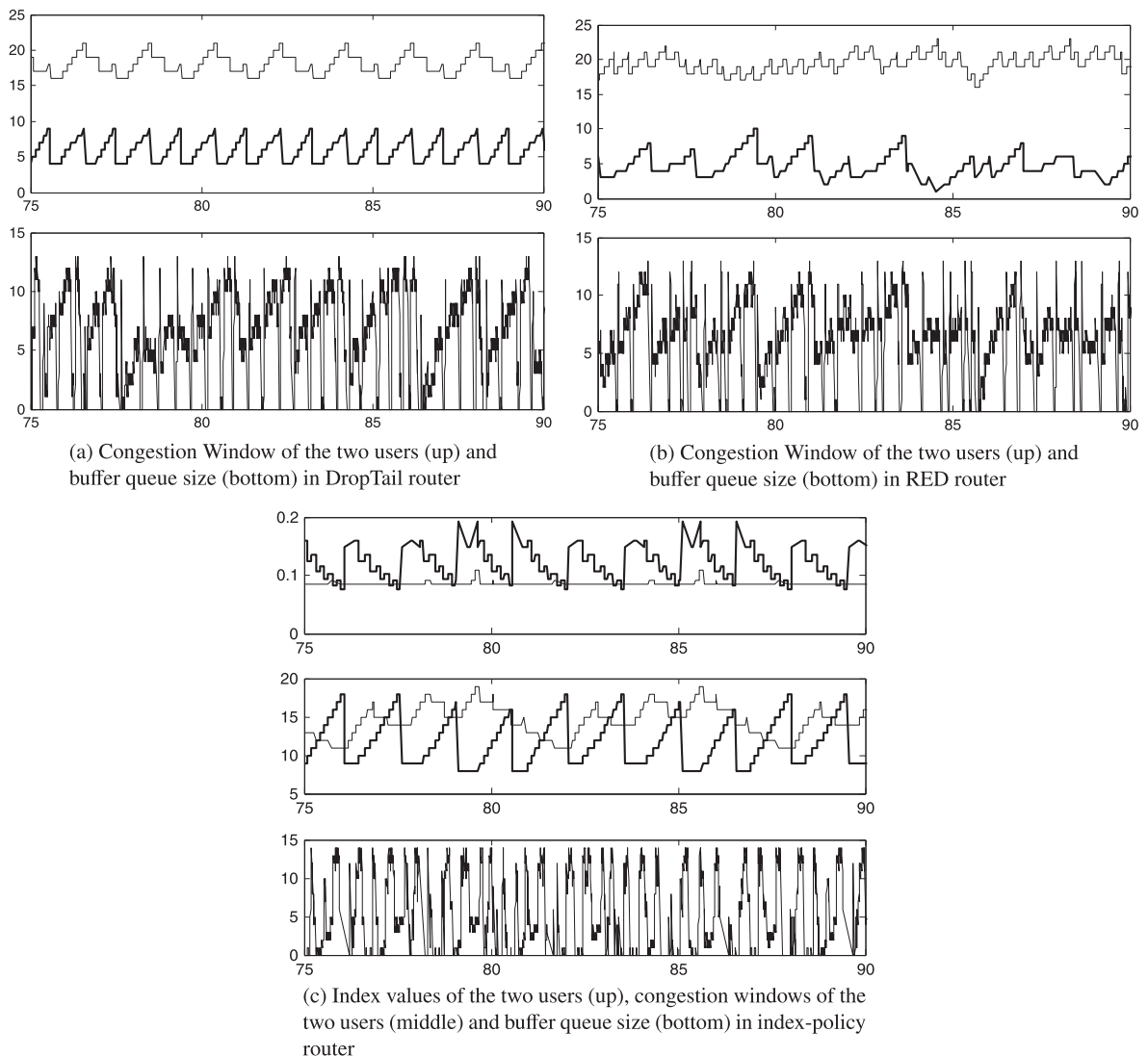
(a) Congestion Window of the two users (up) and buffer queue size (bottom) in DropTail router

(b) Congestion Window of the two users (up) and buffer queue size (bottom) in RED router

(c) Index values of the two users (up), congestion windows of the two users (middle) and buffer queue size (bottom) in index-policy router

Fig. 7. Scenario 3: Simulation of a bottleneck link with two users, where User1 has a TCP with $\gamma_1 = 0.9$ and User2 has standard TCP ($\gamma_2 = 0.5$).

routers. In this scenario, the index could be written in the new IP header, so that our packet method in the routers is also applicable.

Gradual implementation: There is no need to implement the index mechanism in all the routers at the same time, and it is in general more beneficial for the bottleneck routers (e.g., edge routers and servers), since non-congested routers will have their congestion price close to zero. Also it is not necessary that all the end-users start to write indices in the packet headers at the same time – those without index could be treated in the routers as today or defining a fixed index, which would incentivize the end-users to adopt the index mechanism.

8.2. Router-only implementation

The packets may not come with their index in the header, either because the TCP sources are not cooperative, or because modification of TCP headers is not implementable, or because adding the index cell to the packet header of the current TCP variants is not acceptable. Then, our mechanism could be implemented at the routers only. The router must be able to recognize which packet belongs to which flow, the current window and the TCP variant (future behavior) of each flow, and compute (an estimation of) the index for each flow.

In this case, a possibility would be that routers focus only on long TCP flows. Traffic measurements on the Internet have consistently showed that even though most TCP flows are small in size, the majority of data is carried by long TCP flows. One could then implement a priority scheme, see for example [32], such that short TCP flows receive high priority. As a consequence, losses will only be suffered by long TCP flows. At any given time, the number of concurrent long TCP flows is rather small (see [8] for details), which makes feasible the implementation of an
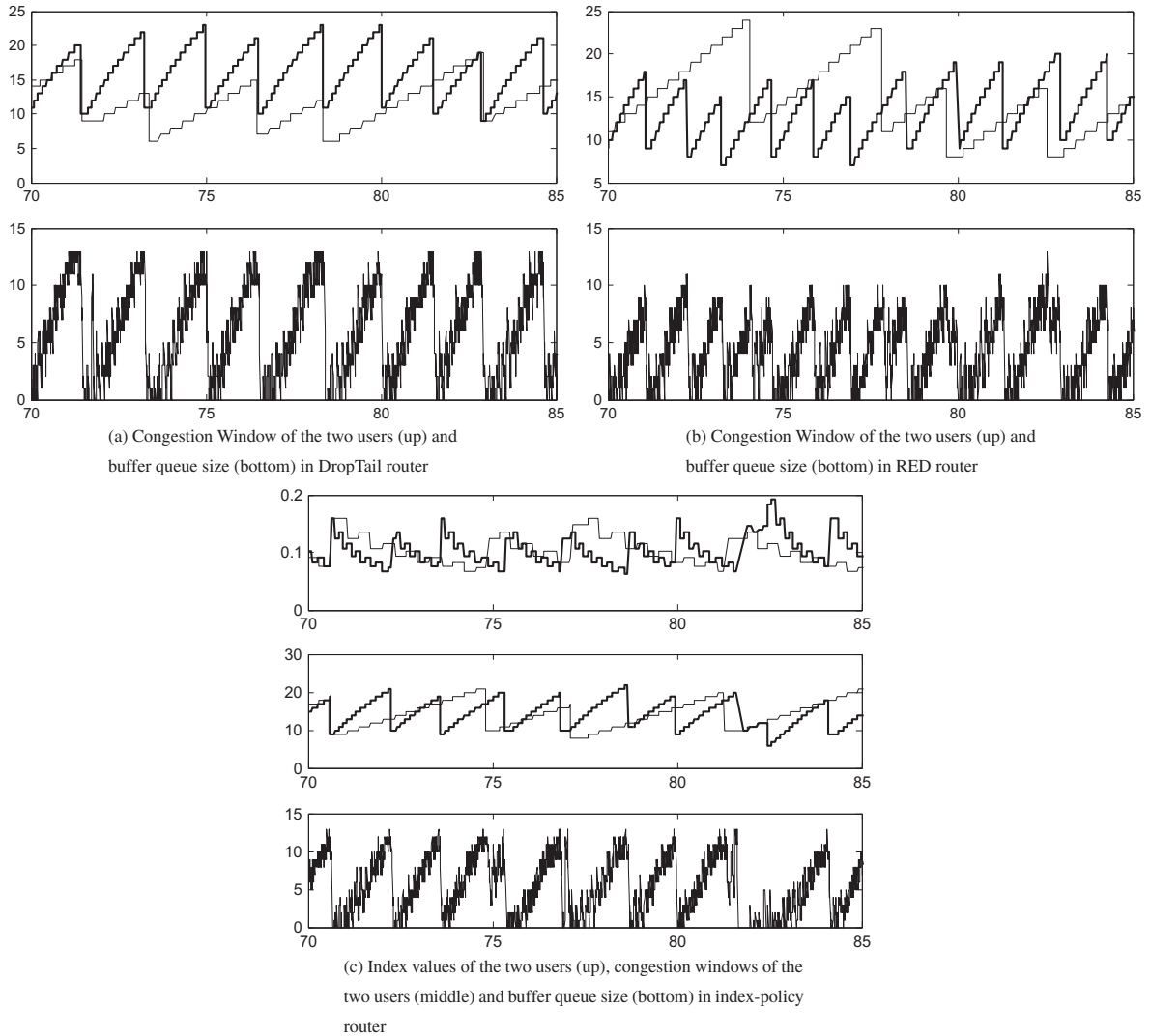
(a) Congestion Window of the two users (up) and
buffer queue size (bottom) in DropTail router

(b) Congestion Window of the two users (up) and
buffer queue size (bottom) in RED router

(c) Index values of the two users (up), congestion windows of the
two users (middle) and buffer queue size (bottom) in index-policy
router

Fig. 8. Scenario 4: Simulation of a bottleneck link with two users, where the propagation delay of User1 is 50 ms.

index-based algorithm to classify the packets. We refer to [32] for a reference justifying how flow-based routers can be implemented on the Internet.

9. Conclusion

In this paper we have introduced a rigorous framework of Markov Decision Processes for the problem of optimal queue management in a bottleneck router in order to achieve fast and fair transmission. We have focused on showing tractability of the model and designed a heuristic by means of transmission indices that can be implemented for packet-level admission to the buffer.

The main goal has been to study in a benchmark topology the fundamental features and performance of the proposed heuristic for flows that behave according to TCP/IP protocol. A noticeable feature is that the proposed heuristic manages to desynchronize (and even counter-synchronize)

the flows, so that network resources can be used more efficiently. We have shown in NS-3 simulations that the proposed heuristic significantly improves over the DropTail and RED policies in several aspects, including fairness across users with different TCP variants and fairness with respect to different Round Trip Times. In scenarios where DropTail and RED performed well in these types of fairness, the proposed heuristic maintained the same levels of fairness, and improved the throughput at the same time. We also believe that we provide fundamental framework and ideas useful or directly applicable also in next-generation Internet architectures such as the Information-centric networking (ICN) and wireless networks.

We have made several simplistic assumptions in the modeling of TCP. However, we believe that our approach opens an interesting research avenue to design admission control policies in Internet routers. In order to fully validate our approach, future work must address some of the limitations of our model, namely, the efficient computation

of index values, impact of misbehaving TCP sources, simulations in more realistic scenarios.

References

[1] M. Allman, V. Paxon, W. Stevens, TCP Congestion Control, RFC2581, April 2002.

[2] V. Jacobson, Congestion avoidance and control, in: Proceedings of ACM SIGCOMM, 1988, pp. 314–329.

[3] K. Ramakrishnan, S. Floyd, D. Black, The Addition of Explicit Congestion Notification (ECN) to IP, RFC3168, September 2001.

[4] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.

[5] [5] K. Fall, S. Floyd, Simulation-based comparisons of Tahoe, Reno and SACK TCP, ACM Computer Communication Review 26 (3) (1996).

[6] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, Inc.,, Hoboken, New Jersey, 2005.

[7] P. Whittle, Restless bandits: activity allocation in a changing world a celebration of applied probability, in: J. Gani (Ed.), Journal of Applied Probability 25A, 1988, pp. 287–298.

[8] J. Mo, J. Walrand, Fair end-to-end window-based congestion control, IEEE/ACM Transactions on Networking 8 (5) (2000) 556–567.

[9] E. Altman, K. Avrachenkov, A. Garnaev, Generalized α-fair resource allocation in wireless networks, in: Proceedings of the 47th IEEE Conference on Decision and Control, Cancun, Mexico, 2008, pp. 2414–2419.

[10] J. Gittins, K. Glazebrook, R. Weber, Multi-Armed Bandit Allocation Indices, Wiley, 2011.

[11] R. Weber, G. Weiss, On an index policy for restless bandits, Journal of Applied Probability (27) (1990) 637–648.

[12] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, in: Proceedings of SIGCOMM '02, 2002.

[13] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, N. McKeown, Processor sharing flows in the Internet, in: Proceedings of IWQoS '05, Lecture Notes in Computer Science, vol. 3552, 2005, pp. 271-285.

[14] M. Lestas, A. Pitsillides, P. Ioannou, Congestion control in computer networks, in: Modeling and Control of Complex Systems, CRC Press, 2008.

[15] D. Wei, C. Jin, S. Low, S. Hegde, FAST TCP: motivation, architecture, algorithms, performance experimental evaluation of TCP protocols for high-speed networks, IEEE/ACM Transaction on Networking 14 (6) (2006) 1246–1259.

[16] S. Ha, I. Rhee, L. Xu, CUBIC: a new TCP-friendly high-speed TCP variant, Operating Systems Review 42 (5) (2008) 64–74.

[17] K. Tan, J. Song, Q. Zhang, M. Sridharan, A compound TCP approach for high-speed and long distance networks, in: Proceedings of IEEE INFOCOM, 2006.

[18] Network Simulator, ver.3, (ns-3), 2008. <http://www.nsnam.org/>.

[19] F. Kelly, Charging and rate control for elastic traffic, European Transactions on Telecommunications 8 (1997) 33–37.

[20] K. Avrachenkov, U. Ayesta, J. Doncel, P. Jacko, Optimal congestion control of TCP flows for internet routers, Performance Evaluation Review 40 (3) (2012) 62–64.

[21] V. Cerf, R. Kahn, A protocol for packet network intercommunication, IEEE Transactions on Communications 22 (5) (1974) 637–648.

[22] Y. Li, D. Leith, R. Shorten, Experimental evaluation of TCP protocols for high-speed networks, IEEE/ACM Transaction on Networking 15 (5) (2007) 1109–1122.

[23] G. Vu-Brugier, R. Stanojevic, J. Leith, R. Shorten, A critique of recently proposed buffer-sizing strategies, ACM SIGCOMM Computer Communication Review 37 (1) (2007) 43–48.

[24] P. Jacko, Adaptive greedy rules for dynamic and stochastic resource capacity allocation problems, Medium for Econometric Applications 17 (4) (2009) 10–16.

[25] S. Shakkottai, R. Srikant, Network Optimization and Control, NoW Publishers, 2007.

[26] P. Jacko, B. Sansò, Optimal anticipative congestion control of flows with time-varying input stream, Performance Evaluation 69 (2) (2012) 86–101.

[27] K. Ma, R. Mazumdar, J. Luo, On the performance of primal/dual schemes for congestion control in networks with dynamic flows, in: Proceedings of The 27th Conference on Computer Communications (IEEE INFOCOM), 2008, pp. 960–967.

[28] J. Niño-Mora, Dynamic priority allocation via restless bandit marginal productivity indices, TOP 15 (2) (2007) 161–198.

[29] K. Avrachenkov, P. Jacko, Index policies for congestion control of TCP flows, 2013 (in preparation).

[30] P. Jacko, Restless bandits approach to the job scheduling problem and its extensions, in: A.B. Piunovskiy (Ed.), Modern Trends in Controlled Stochastic Processes: Theory and Applications, Luniver Press, United Kingdom, 2010, pp. 248–267.

[31] R. Jain, D. Chiu, W. Hawe, A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems, DEC Research Report TR-301, 1994.

[32] J.W.R. Abdesselem Kortebi, Sara Oueslati, Cross-Protect: Implicit Service Differentiation and Admission Control, IEEE HPSR.

Konstantin Avrachenkov received the Master degree in Control Theory from St. Petersburg State Polytechnic University (1996), Ph.D. degree in Mathematics from University of South Australia (2000) and Habilitation (Doctor of Science) from University of Nice Sophia Antipolis (2010). Currently he is a Director of Research at Inria Sophia Antipolis, France. He is an associate editor of International Journal of Performance Evaluation. His main research interests are Markov processes, singular perturbation theory, queueing theory, mathematical programming, game theory and performance evaluation of communication networks.

Urtzi Ayesta received a PhD degree from Université de Nice-Sophia Antipolis (France), a MS degree in Electrical Engineering from Columbia University (US) and a BS/MS degree in Telecomunication Engineering from Nafarroako Unibertsitate Publikoa-Universidad Publica de Navarra (Spain). His PhD research work was carried out at the research laboratories of INRIA Sophia-Antipolis and France Telecom R&D. He is currently a CNRS researcher working at LAAS, Toulouse, and an IKERBASQUE Research Professor in the Computer Science Faculty at the University of the Basque Country.

Josu Doncel obtained his B. Sc. in Mathematics from the University of the Basque Country (UPV/EHU) in 2010 and his Industrial Engineering, in Industrial Electronics, degree at UPV/EHU in 2007. He did the Master in Mathematical Modeling, Statistics and Computation between 2010 and 2011, at the UPV/EHU. In 2011, he was an intern at BCAM, co-supervised by U. Ayesta (LAAS-CNRS), K. Avrachenkov (INRIA) and P. Jacko (BCAM). Josu is a Ph.D. student at LAAS-CNRS under the supervision of O. BRUN (LAAS-CNRS) and B. PRABHU (LAASCNRS).

Peter Jacko is Lecturer in the Department of Management Science at Lancaster University Management School, UK. He is also affiliated as Researcher with the BCAM – Basque Center for Applied Mathematics, Spain, where he was formerly co-leader of the Networks research group. He obtained his Ph.D. in Business Administration and Quantitative Methods (2009) and D.E.A. in Statistics and Operations Research (2006) from the Universidad Carlos III de Madrid, Spain. He obtained his Mgr. (2003) and Bc. (2002) degrees in Mathematics from the Univerzita P.J. Šafárika v Košiciach, Slovakia.