Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 51 (2007) 4460–4474

Computer Networks

www.elsevier.com/locate/comnet

Stability of FIFO networks under adversarial models: State of the art [☆]

Vicent Cholvi [*], Juan Echagüe

Universitat Jaume I, Campus del Riu sec, 12071 Castellón, Spain

Received 28 October 2006; received in revised form 23 April 2007; accepted 2 July 2007
Available online 12 July 2007

Responsible Editor: E. Altman

Abstract

Network stability is an important issue that has attracted the attention of many researchers in recent years. Such interest comes from the need to ensure that, as the system runs for an arbitrarily length of time, no server will suffer an unbounded queue buildup.

Over the last few years, much research has been carried out to gain an understanding of the factors that affect the stability of packet-switched networks. In this paper, we attempt to review the most noteworthy results in this area. We will focus on networks where the scheduling policy is of the FIFO type, which is, by far, the most widely adopted policy. We gather these results and present them in an organized manner. Furthermore, we also identify some directions open to future research.

Keywords: Network stability; FIFO scheduling; Adversarial queuing theory

1. Introduction

A growing number of networking applications today have constraints in terms of their maximum allowable end-to-end delay, packet loss rate, bandwidth, availability, and so forth. Therefore, it is becoming increasingly important to find out the conditions under which a given communication network guarantees performance bounds when dealing with emerging real-time-oriented networking scenarios. In spite of the significant advances in the complexity of communication networks, much work still needs to be done in that direction.

In order to characterize the performance of a network, one crucial issue is that of *stability*, which has become a major topic of study in the last decade. Roughly speaking, a communication network system is said to be *stable* if the number of packets waiting to be delivered (backlog) is finitely bounded at any one time. The importance of such an issue is obvious, since if one cannot guarantee stability, then one cannot hope to be able to ensure deterministic

[☆] This work was partially supported by the Spanish Ministry of Science and Technology under Grant No. TSI2004-02940.

[*] Corresponding author. Tel.: +34 964728332; fax: +34 964728435.

E-mail address: vcholvi@uji.es (V. Cholvi).

guarantees for most of the network performance metrics.

In a paper dating back to 1975 [1], Kelly proved that in stochastic networks where packet sizes and packet inter-arrival times are exponentially distributed, if the service time at servers follows the same distribution then the well-known scheduling policy FIFO (which is by far the most widely adopted policy) is stable. After that and for many years, the common belief was that only overloaded queues[1] could generate instability, while underloaded ones could only induce delays that are longer than desired, but always remain stable. This general wisdom goes back to the models of packet-switching networks originally developed by Kleinrock [2], and based on Jackson queuing networks [3]. Stability results for more general classes of queueing networks [4,5] also confirmed that only overload generates instability. This belief was shown to be wrong when it was observed that, in some networks, the backlogs in specific queues could grow indefinitely even when such queues were not overloaded [6,7]. It was later observed that instability could also occur, even when the ratio between arrival rate and service rate is arbitrarily small, under the FIFO scheduling policy, both by using probabilistic assumptions [8,9] and by considering deterministic ones [10]. However, the above mentioned counterexamples required that the time needed to process a packet be different from one to another. Clearly, this is not, in general, a valid assumption in packet-switched networks, where servers generally have the same service rates for all packets (such networks are usually said to be of the *Kelly type* [5]). Nevertheless, shortly afterwards it was shown that instability could also arise in some types of Kelly networks, including networks using the FIFO scheduling policy [11,12]. These later results aroused an interest in understanding the stability properties of packet-switched networks.

This paper provides a review and synthesis of the most important results concerning stability of networks using the FIFO scheduling policy, which have usually appeared in a continuous but dispersed form. Here, we bring these results together and present them in an organized manner. Furthermore, throughout this survey we also identify a number of directions open to future research.

[1] A queue is considered to be overloaded when the total arrival rate at any server is greater than the service rate.

The paper has two clearly differentiated parts. In the first, which comprises Sections 2–4, we talk about network stability and discuss the different dimensions from which stability can be investigated. We also characterize the model of input traffic we will use and formally define what we mean by stability in that model. In the second part, which comprises Sections 5–7, we review the state of the art concerning stability in networks with a FIFO scheduling policy, and present the results obtained when considering several different points of view; whereas in Section 5 we present some instability results, in Section 6 we present the most relevant directions used to approach stability and in Section 7 we tackle the problem of deciding which networks are stable. We finish with some concluding remarks in Section 8.

In order to keep a historical perspective of the results, in each reference we include the first version of the paper (mostly conference versions) as well as the final one (mostly journal papers).

2. Network scenario

Network stability can be approached from three dimensions (G, A, P): the *network topology* G, the *input traffic pattern* A and the *packet scheduling protocol* P.

2.1. Network topology

The network topology constitutes the underlying infrastructure by means of which packets travel from their source to their destination. It is composed of network switches (also referred to as routers, servers or nodes), which are interconnected by means of unidirectional or bidirectional links. Each node contains a server for each outgoing link. Servers may have different service rates (link bandwidths), measured in packets per unit of time. However, in general and for the sake of simplicity, it is assumed that each link can transmit a single packet in each time step (i.e., they have a normalized service rate of 1). Furthermore, there is a propagation delay associated to each link. Each server schedules the packets that must cross the link using a nonpreemptive scheduling policy (which may be different at each server). Packets are forwarded in a store-and-forward manner.

We will represent networks by means of directed graphs $\mathcal{G} = (V, E)$, where vertices V represent the nodes and edges E are the links between servers, the orientation of which represents the direction in which the traffic flows through the network link. In the case where traffic can flow in both directions of a network link, such a link will be represented by two edges, one in each direction.

2.2. Input traffic

When a request arrives for the transmission of a certain amount of data, a connection is established along one or more routes. Clearly, network stability is affected by the requests for transmission, their variabilities and the routes followed to reach their destinations. The input traffic that is allowed is usually characterized by specifying a *constraining function* that bounds the maximum number of packets that can be injected at each time interval, the nodes where packets ingress into the network and the route followed by each of the packets until they reach their destination.

Although the average load on each link must be within its bandwidth capacity in order to guarantee stability, this is not a sufficient condition [6–11]. This raises a fundamental question about how to bound the input traffic to make the network stable. In traditional queuing theory, the input traffic pattern is generally assumed to be characterized by a stochastic distribution (e.g., a Poisson distribution). However, while such an assumption is convenient for theoretical analysis, questions have been raised about its realism [13]. In real data networks, connection arrivals may be seriously correlated and bursty, rather than stationary in nature. To model the bursty phenomena in data arrivals, instead of assuming stochastic stationary arrival processes, *bursty models* [14,12] have been introduced for data communication networks, thus allowing us to study how burstiness affects network stability. These latter models will be formally defined in Section 3.

2.3. Scheduling protocol

The scheduling protocol is responsible for deciding the order in which packets trying to cross a link simultaneously will be served. Since only one packet can cross the link in a single step, the rest of the packets will have to wait in the queue associated with the congested link.

Although scheduling algorithms have been studied for decades, almost all routers currently implement the first-in first-out protocol (FIFO); that is, all arriving packets are treated equally by placing them into a single queue, and then serving them in a *greedy* fashion[2] in order of arrival. The reasons for the widespread adoption of FIFO as a scheduling algorithm are clear. On the one hand, FIFO is easily implementable, which makes it very attractive for system designers. On the other hand, FIFO is also very fast, since the time required to make a scheduling decision is insignificant. In addition, because it only works with local information (as opposed to other policies, like Farthest-to-Go, that rely on the packets subsequent path, or the Longest-in System, that relies on the packet injection time), it prevents packets from being faked to a higher priority.

Nevertheless, and contrary to what one could expect, it has not usually been easy to analyze its properties. This occurs because the ordering imposed by the FIFO policy is so "loose" that it is difficult to ascertain the individual behavior of packets and, consequently, extract consequences about the behavior of the whole system.

3. Adversarial models of input traffic

As has been pointed out in the previous section, in real data networks, the assumption that input traffic is characterized by a stationary process is not realistic. Therefore, to model the bursty phenomenon in data arrivals, a new type of model has been proposed. Such models consider the time evolution of a packet-routing network as a game between a malicious *adversary* that has the power to perform a number of actions (such as injecting packets at particular nodes, choosing their destination, routing them, etc.) and the underlying system. Such an adversary, based on the knowledge of behavior of the system, can devise the scenario that maximizes the "stress" on the system. Consequently, it provides us with a valuable tool with which to analyze the network in a worst-case scenario. On the one hand, positive results (i.e., stability results) are more robust in that they do not depend on particular stationary assumptions about the input sequences. On the other hand, since an adversary could encompass a wider range of actions than stationary inputs, it may produce unstable scenarios that are not allowed using a stochastic model. Thus, since the stability results derived using

[2] A scheduling protocol is called greedy (also known as work-conserving) if it cannot be idle as long as there is at least one packet queued.

adversarial models[3] are, in the above mentioned sense, more general than those obtained using stationary models, we will focus on findings that take into account only adversarial models of input traffic.

In an adversarial model each packet is injected (by the adversary) into a node and follows a specific unique path, after which it is absorbed. Paths, however, cannot contain the same link more than once. If more than one packet wishes to cross an edge e in the current time step, then the protocol chooses one of these packets to send across e; and remaining packets wait in a queue at the tail of e.

To describe the dynamics of the system being considered, we introduce some notation adopted from [15]. Let P be a set of paths that cannot contain the same edge more than once in a network $\mathcal{G} = (V, E)$. Packets will follow paths in P, which might be the set of all paths or just a subset of it. For each path $p \in P$, let $\{e_0^p, e_1^p, e_2^p, \ldots, e_{k(p)}^p\}$ be the set of consecutive edges in p. Let $A_p(t_1, t_2)$ be the total number of packets that are injected during time interval $[t_1, t_2)$ and use path p. Let $D_{e,p}(t_1, t_2)$ be the total number of packets following path p and traversing edge e within the time interval $[t_1, t_2)$. Finally, let $Q_{e,p}(t)$ be the total number of packets following path p that are waiting to traverse edge e at time t.

The dynamics of the network are described as follows. For each $t = 0, 1, 2, \ldots$ and each path $p \in P$

$$Q_{e_0^p,p}(t) = Q_{e_0^p,p}(0) + A_p(0,t) - D_{e_0^p,p}(0,t) \tag{1}$$

and for all $i = 1, 2, \ldots, k(p)$

$$Q_{e_i^p,p}(t) = Q_{e_i^p,p}(0) + D_{e_{i-1}^p,p}(0,t) - D_{e_i^p,p}(0,t). \tag{2}$$

For each edge e and each time interval $[t_1, t_2)$ the following constraint must hold:

$$\sum_{p:e\in p} D_{e,p}(t_1,t_2) \leqslant t_2 - t_1. \tag{3}$$

The first concrete model that implicitly contained the concept of an adversary was proposed by Cruz in [14] and it is called as *Permanent Session Model* (*PSM*) (also known as the *Session Oriented Model* or the (σ, ρ)-*Regulated Session Model*). In this model all packets are forced to belong to some session, and packets from the same session follow the same path. Each session p is associated to a *rate* ρ_p and a burst allowance σ_p so that the number of packets injected by a session p during time interval $[t_1, t_2)$ is bounded by

$$A_p(t_1,t_2) \leqslant \sigma_p + \rho_p(t_2 - t_1). \tag{4}$$

It is clear that, in order to avoid trivially overloading the system, the maximum traffic injected in every link over long periods of time should not exceed the amount of traffic that the link can serve. Therefore, the following *load condition* must be fulfilled for all e

$$\sum_{p:e\in p} \rho_p \leqslant 1. \tag{5}$$

Let $Q(t)$ denote the vectors of queue lengths at time t, and $A(t)$ and $D(t)$ respectively denote the vector of arrivals and departures up to time t. Any feasible solution $(Q(t), A(t), D(t))$ to (1)–(4) will be called a *realization* in the $(\mathcal{G}, PSM, \mathcal{P})$ system, where $\mathcal{P}$ is the scheduling protocol by which packets are chosen to cross edges.

Informally, it can be said that a *PSM* adversary can control individual input streams. However, in some scenarios it seems convenient to provide a better model for networks having heterogeneous and frequently changing rates of traffic. That is, it seems convenient to have an adversary that globally controls the entire input process. In [12], Borodin et al. provided a new perspective to the analysis of stability in packet-switched networks by introducing a new framework, known as *Adversarial Queuing Theory*[4] (*AQT*), which has given rise to the appearance of a large number of results. In *AQT*, in each time step, the adversary injects a set of packets at some of the nodes. For each packet, it also specifies the path it must follow (i.e., the sessions in which packets are being injected can change over time). As in the case of *PSM*, in order to avoid trivially overloading the system, the maximum traffic injected in every link over long periods of time should not exceed the amount of traffic that the link can serve. Formally, the number of packets injected by the adversary in any consecutive time steps w ($w \geqslant 1$ represents a window size) requiring any particular link e is bounded by

$$\sum_{p:e\in p} A_p(t_1,t_2) \leqslant \lceil wr \rceil, \tag{6}$$

where $w = t_2 - t_1$ and $0 < r \leqslant 1$ represents the normalized injection rate.

[3] They are so called to reflect the fact that the emphasis is on stability with respect to an adversarial model of input traffic (i.e., a model where packets are injected and routed by an adversary), rather than on an oblivious randomized process.

[4] Also known as *Windowed Adversarial Queuing*.

In [11], Andrews et al. introduced a new adversarial model known as the *Leaky-Bucket Model* (*LB*) that differs from *AQT* in that the number of packets that the adversary is allowed to inject during any time interval $[t_1, t_2)$ and that require the link e is now bounded by

$$\sum_{p:e\in p} A_p(t_1, t_2) \leqslant r(t_2 - t_1) + b, \tag{7}$$

where $b \geqslant 0$ represents a *burst* allowance and $0 < r \leqslant 1$ represents the sustainable normalized injection rate.

Fig. 1 compares *AQT* and *LB* in terms of the number of packets the adversary could inject in a period of time that have to cross a given edge e. As can be seen, an *LB* adversary can inject at least as many packets as an *AQT* adversary. In fact, Rosén [16, Fact 1] compared both models (starting with an empty configuration), showing that they have the same power provided $r < 1$. Two adversaries $\mathscr{A}_1$ and $\mathscr{A}_2$ have the same power if the set of actions (such as injecting packets at particular nodes, choosing their destination, routing them, etc.) performed by $\mathscr{A}_1$ can also be performed by an $\mathscr{A}_2$, and vice-versa. To establish the equivalence, it was necessary to use adversaries with different injection rates, but not different sequences of packet trajectories. Therefore, we consider them as being the same, jointly called *Adversarial Queuing Model* (*AQM*). Similar to the case of *PSM*, any feasible solution $(Q(t), A(t), D(t))$ to (1)–(3) (6)/(7) will be called a *realization* in the $(\mathscr{G}, AQM, \mathscr{P})$ system, where $\mathscr{P}$ is the scheduling protocol by which packets are chosen to cross edges.

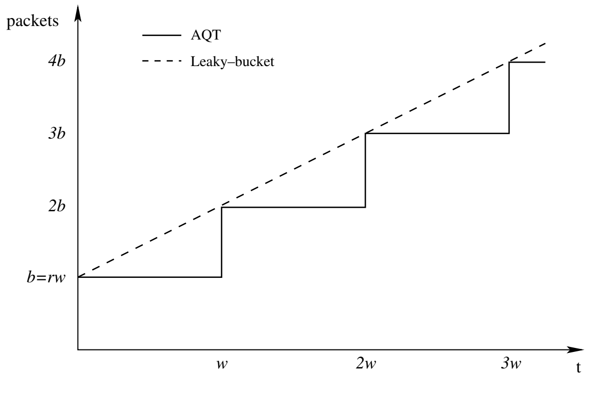

Fig. 1. Number of packets injected over time that require a given edge for *AQT* and *LB* adversaries. The same value for the injection rate r has been taken, and it is assumed that $b = rw$.

Clearly, *PSM* is more restrictive than *AQM*, since the adversarial injection strategies in *AQM* are more general than in *PSM*. Hence, any stability result in *AQM* implies an analogous result in *PSM*; the converse, however, does not necessarily hold. In turn, any instability result in *PSM* implies an analogous result in *AQM*; again, the converse does not necessarily hold.

Adversarial models other than *AQM* and *PSM* have been proposed. For instance, Aiello et al. [17] proposed a variation of *AQM* where the adversary specifies both the origin and destination of each packet and they are dynamically routed according to certain network parameters (i.e., the adversary does not specify the trajectories packets follow). A different variation was proposed by Andrews et al. in [18], where the entire trajectory of a packet is known at the source, instead of being dynamically routed. Alvarez et al. [19] proposed yet another variation of *AQM* that allows the adversary to prioritize packets, either in a fixed fashion (at packet injection time) or in a variable manner (the priority of each packet is assigned at each time step). However, these models can be seen as being either a reinforcing or a weakening of the adversarial power. So, in the following sections when we present the results, we only consider *AQM* and *PSM* and refer, when necessary, to the factors that strengthen or restrain their power.

4. Network stability in adversarial models

In the previous sections, it has been outlined what we mean by network stability. Now, we define stability in a more formal way when considering adversarial models of input traffic.

Given a network $\mathscr{G}$, an adversary $\mathscr{A}$ and a scheduling protocol $\mathscr{P}$, a realization $(Q(t), A(t), D(t))$ in $(\mathscr{G}, \mathscr{A}, \mathscr{P})$ is *stable* if the number of packets in the system is bounded at all times by a fixed value. That is, if

$$\sup_{t\in Z^+} \sum_{e\in p,\ p\in P} Q_{e,p}(t) < \infty.$$

We say that $(\mathscr{G}, \mathscr{A}, \mathscr{P})$ is *stable* if every realization is stable. Furthermore, stability can also be addressed from the point of view of the scheduling protocol or the network. Therefore, if $(\mathscr{G}, \mathscr{A}, \mathscr{P})$ is stable, we say that the scheduling protocol $\mathscr{P}$ is stable against adversary $\mathscr{A}$ with network $\mathscr{G}$. Alternatively, we also say that network $\mathscr{G}$ is stable against adversary $\mathscr{A}$ with scheduling protocol $\mathscr{P}$.

In the most concrete case where a scheduling protocol $\mathscr{P}$ is stable against *every* adversary $\mathscr{A}$ covered by a given adversarial model with *every* network, we say that it is stable against such an adversarial model. Similarly, when a network $\mathscr{G}$ is stable against *every* adversary $\mathscr{A}$ covered by a given adversarial model with scheduling protocol $\mathscr{P}$, we say that it is $\mathscr{P}$-stable against such an adversarial model.

We note that in the previous definitions nothing has been said about the initial configuration. It has been argued in [11, Lemma 2.9] that systems with empty initial configurations (i.e., assuming that, at time zero, there are no packets in the system) and systems with nonempty initial configurations are equivalent, since any adversary in the latter can be transformed into an adversary in the former that behaves similarly. That allows us to work, without loss of generality, with models with empty or nonempty initial configurations. But it must be taken into account that the construction used to establish such a result needs to change the network topology and creates a set of packets that have a specific age; therefore, if the scheduling policy bases its queuing decision on its history (e.g., Longest-in-System, Farthest-from-Source, etc.) it is not clear if the above mentioned result remains valid. By using a very simple transformation, Blesa [20, Fact 2] has shown the equivalence of adversaries with and without initial configuration only by changing the parameters of the adversary. This is a stronger result than the analogous one given in [11, Lemma 2.9] since here the graph does not need to be changed. Nevertheless, we remark that the FIFO scheduling policy does not take into account the packets history.

We conclude this section by noting that one important direction for investigating stability of a queuing network is the analysis of the associated fluid limits, which are the different "limits" one obtains by shrinking the weight of individual packets and time proportionally. The fluid limits will satisfy fluid model equations, which correspond to the deterministic analog of the queuing network under consideration. To prove stability, typically one attempts to show that solutions of the fluid model equations are stable (i.e., their queue lengths are 0 by a fixed time). The stability of the (non-adversarial) queuing network then follows from the stability of these solutions, as proved by Dai [21]. Subsequently and parallelizing the result obtained by Gamarnik [15] has shown that the stability of an adversarial fluid model implies the stability of an underlying adversarial queuing network (AQM). However, the connection between a queueing network (either adversarial or not) and the associated fluid model is a fairly complex issue, neither trivial nor subtle, and it is presently not known when the stability of a fluid model follows from that of the corresponding queueing network. Only some partial results were proven by Dai [22] and Meyn [23], stating that when the fluid limits all have a uniformly positive drift, then the queuing network itself is unstable. Bramson [24] and Dai et al. [25] have also shown that there are families of queuing networks that are stable, but whose fluid models are unstable.

5. Instability results

Perhaps the most natural question regarding the stability of FIFO is to unveil whether or not it is a stable policy with every network. Unfortunately, it has been found that FIFO can be unstable in some circumstances, contrary to what happens when considering some scheduling disciplines like Farthest-to-Go, Longest-in-System, Nearest-to-Source, etc.

To show that a $(\mathscr{G},\mathscr{A},\mathscr{P})$ system is not stable, one has to find an adversarial injection strategy for $\mathscr{A}$ such that, as time goes by, the number of packets in the system grows unboundedly. But, there is an interesting result by Hajek in [26, Propositions 1 and 2] that shows that if a $(\mathscr{G},\mathscr{A},\mathscr{P})$ system is stable then any other $(\mathscr{G},\mathscr{A}',\mathscr{P})$ system will be stable provided $\mathscr{A}$ and $\mathscr{A}'$ have the same injection rate. This shows that large bursts are not, in themselves, enough to cause instability and rules out burstiness as a factor that could produce instability, and it also implies that one must center on the injection rate to find the adversarial injection strategy that causes the system to be unstable. Therefore, we can turn out our attention to the injection rate as the only factor that affects network instability.

The first result regarding instability of FIFO in adversarial models was given by Andrews et al. In [11, Theorem 2.10] it was proven that there is a network $\mathscr{G}_B$ (see Fig. 2) and an AQM adversary $\mathscr{A}$ of rate $r \geqslant 0.85$ such that $(\mathscr{G}_B,\mathscr{A},FIFO)$ is unstable. The proof breaks the construction of $\mathscr{A}$ down into phases. Briefly, it assumes that, at the very beginning, there are s packets in the queue of v_0. During the first s steps, a set X of rs packets that want to traverse edges $e_0 f_0' e_1$ are injected in v_0. Then, for the next rs steps, a set Y of $r^2 s$ packets that want to traverse edges $e_0 f_0 e_1$ are injected in v_0. The core of the proof consists in delaying the packets in X

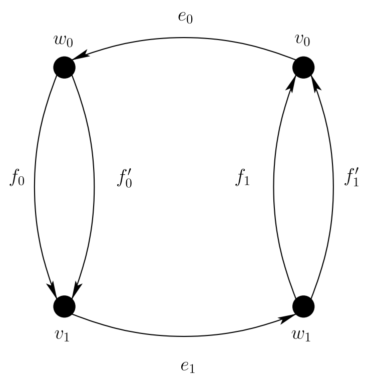

Fig. 2. Baseball network $\mathcal{G}_B$ used in [11, Theorem 2.10] to show instability of FIFO.

using single-edge injections until packets in Y are ready to traverse edge f_0'. Therefore, the packets that cross f_0' and f_0 will merge in v_1. By injecting new packets in v_1 that want to traverse e_1, they show that the queue of e_1 will contain $r^3s + r^2s/(r+1)$ packets, which is greater than s if $r \geqslant 0.85$. Since the graph is symmetric, one can repeat the same process to increase the queue size in v_0, and then also repeat the whole process with a value $s' > s$.

The result obtained by Andrews et al. triggered an effort to reduce the injection rates for which FIFO is unstable. Díaz et al. [27, Theorem 3] decreased the instability bound to 0.8357, Koukopoulos et al. [28, Theorem 3] lowered it to 0.749, and Lotker et al. [29, Theorem 3.17] brought it down to 0.5. The definitive result that determines the minimum injection rate for which FIFO is unstable was given by Bhattacharjee et al. [30, Theorem 5.4]. Specificaly, they proved that FIFO can be unstable at arbitrary low-load injection rates. The main idea for the proof was the construction of a gadget which, assuming certain initial conditions, allows only a small fraction of packets to pass through it for a long period of time. In particular, the fraction of packets which escape is bounded by $k/(1+r)^k$, where k is a parameter of the gadget and can be increased arbitrarily. The network is constructed using this gadget and the adversary works in phases. At the beginning of a phase, it is assumed that there are some packets waiting to pass through a column of gadgets. Using each gadget in the column more packets are generated which want to ultimately traverse through a second column. Additional copies of the gadget are used to delay and synchronize these new packets so that, at the end of the phase, there are more packets waiting to traverse the second column than there were waiting at the first column at the beginning of the phase. Applying this inductively leads to instability. One feature of the network constructed in the above mentioned instability proof is that its size is polynomial in $1/r$, which is quite strong. However, this is unavoidable, since it has been shown (see Section 6.2) that, regardless of the network topology, FIFO is stable if $r < 1/(d-1)$, where d is the network diameter (i.e., the largest number of links crossed by any packet). Clearly, this implies that, to obtain a FIFO-unstable network for a small injection rate, one must increase the network diameter.

The previous instability results also raise the question as to what happens if the path each packet must follow is chosen by a routing algorithm instead of being dictated by the adversary. Let us, for instance, consider the baseball network in Fig. 2 and not route any packet throughout the links f_0' and f_1'. Clearly, this network will behave exactly like a ring network. But since any ring is FIFO-stable against AQM (see Section 6.1), then the baseball network in Fig. 2 will also be FIFO-stable, contrary to what happens if the adversary dictates the routes. This evidences the fact that the ability of the adversary to select the routes that packets must follow could make a difference. However, that fact does not imply that FIFO is stable against AQM when the routes are not chosen by the adversary. Indeed, it was shown by Andrews et al. [18, Theorem 4.1] that there is a network $\mathcal{G}_E$ (see Fig. 3) and an AQM adversary $\mathscr{A}$ of rate $r \geqslant 0.9$ such that $(\mathcal{G}_E, \mathscr{A}, FIFO)$ is unstable, regardless of how the routes for the packets are chosen. The proof is similar to the proof in [11, Theorem 2.10] to show instability of FIFO. It breaks the packet injections into

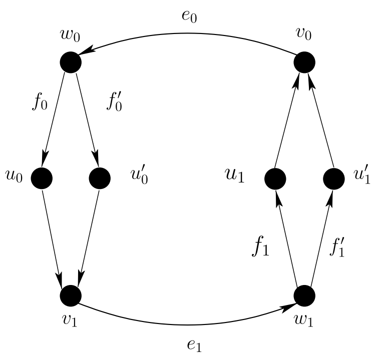

Fig. 3. Extended baseball network $\mathcal{G}_E$ used in [18, Theorem 4.1] to show instability of FIFO, regardless of how the paths for packets are chosen.

phases. Inductively, it is assumed that at the beginning of phase j a set S of s packets with destination u_0 is in the queue of e_0. For the first s steps, a set X of rs packets are injected at node v_0 with destination u_1. These packets are held up at e_0 by the packets in S. Furthermore, rs packets are injected at w_0 with destination u_0. These newly injected packets get mixed with those of S into the set S'. At the end of the first s steps, rs packets from S' are at f_0. For the next rs steps, a set Y of r^2s packets are injected at node v_0 with destination u_1. These packets are held up at e_0 by the packets in X. At the same time, packets are injected at w_0 with destination u'_0 at rate r. These packets delay the packets from X that are routed through f'_0. Hence, at most $rs/(r+1)$ packets of X cross f'_0 and, at the end of these rs steps, a set $X' \subseteq X$ of at least $r^2s/(r+1)$ packets are still at w_0. Finally, for the next $|X'|+|Y|$ steps the packets in X' and Y move forward, and merge at v_1. Meanwhile, packets are injected at v_1 with destination u_1 at rate r. In the end, the number of packets at v_1 with destination u_1 is at least $r^3s + r^3s/(r+1)$. For $r \geqslant 0.9$, $r^3 + r^3/(r+1) > 1$. Since the graph is symmetric, one can repeat the same process to increase the queue size in v_1, and then also repeat the whole process with a value $s' > s$. Therefore, this proves that one cannot hope to achieve FIFO-stability with general networks, even if we have the freedom to choose the routes.

Regarding stability against *PSM*, it must be recalled that the set of injection strategies in *AQM* are more general than in *PSM*. This means that instability against *AQM* does not directly imply a similar result when considering *PSM*. The first result regarding stability when considering *PSM* was presented by Andrews in [31, Theorem 1], where he proposed a network topology that exhibits instability for a maximum injection rate of $1-3\times10^{-4}$, the analogous instability result being parallelized for *AQM*. Such a network consisted of a cycle of seven gadgets, each one made up of 15 servers, partitioned into three columns. The proof was performed by using a fluid network, and it was argued that it is trivial to modify it for a non-fluid network with discrete packets. The creation of instability involves a number of phases, where the aim was to build up fluid in the left and right columns of the gadget. By injecting fluid into each session at a rate of 0 or 1/3 (i.e., the fluid is either *on* or *off*) at given time intervals and using quite involved calculations (derived using computer calculation), he shows that it is possible to inject more fluid into the system than the amount that is escaping. Then, repeating the same process indefinitely, it is possible to build up an arbitrary amount of fluid in the network. In a subsequent work, Andrews extended his instability result to arbitrary low injection rates [32, Lemma 6], thus answering what was, for several years, perhaps the most relevant open question about FIFO stability. Obviously, since the stability result for *AQM* also holds for *PSM*, the size of the networks used to create instability was forced to grow with the inverse of the injection rate (i.e., for all $p \in P : \rho_p < 1/(d-1)$, where d is the network diameter).

- *Open Issue #1*: Whereas the above mentioned instability results at arbitrary low injection rates have, in some sense, closed an important question regarding the stability of FIFO in general networks, it must be noted that no restriction was set on the way the adversaries injected packets into the network (other than satisfying either the *AQM* or *PSM* specifications). Finding under which conditions constraining the input traffic makes FIFO stable seems to be an interesting issue.
- *Open Issue #2*: In *PSM*, determining whether FIFO is stable or not when the routes are not dictated by the adversaries it is still open matter.
- *Open Issue #3*: Similarly, determining the minimum injection rate for which FIFO is unstable against *AQM* when the routes are not dictated by the adversaries is still open.

6. Stability results

The fact that FIFO is not a stable policy under all circumstances does not imply that such a policy could not be stable under some conditions. In this section, we present the most relevant directions that have been used to approach the stability of FIFO under adversarial models.

6.1. Stable topologies

DAGs: The first results regarding the stability of FIFO in particular network topologies were achieved by Cruz in [14, Theorem 4.1]. He proved that any *directed acyclic graph* or DAG is WC-stable against *PSM*, where WC denotes any work-conserving packet-scheduling protocol (remember that a scheduling protocol is said to be greedy or work-conserving if it cannot be idle as long as there is at least one packet queued). This result was extended to *AQM* by Borodin et al. in [12, Theorem 1]. In the case of *PSM*, the known upper bounds on queue sizes and delays of packets are exponential in the length of the longest path in the network and, in the case of *AQM*, in the number of edges.

Trees: The above mentioned bounds for DAGs can be improved when considering rooted tree networks (which are a special case of DAG). In this case, the queue sizes and the packet delays are linear in the length of the longest path in the network [12, Theorem 1], when considering both *PSM* and *AQM*.

Rings: In [33, Theorem 1], Tassiulas and Georgiadis proved that unidirectional n-node ring topologies are WC-stable against *PSM*, evidencing that cyclicity is not, by itself, a problem to achieve stability. Later, Andrews et al. [11, Theorem 3.7] extended that result to *AQM* and showed that the queue size is also linear on the number of nodes in the ring. More precisely, there are never more than $(b+1)n/(1-r)$ packets in the system that require any given edge, and the maximum number of steps a packet spends in the system is $O(bn/(1-r)^2)$ (where (b,r) are the parameters that characterize the *AQM* adversary).

In Table 1, we summarize the upper bounds on the queue sizes of servers and on the maximum end-to-end delay of packets found for the above mentioned network topologies.

6.2. Network structure

Network stability has been also studied from the point of view of the network structure.

By taking into account information on the largest number of links crossed by any session in the network, denoted d, Charny and Le Boudec [34, Theorem 1] proved that if the *load condition* is lower than $\frac{1}{d-1}$ (i.e., for all $p \in P$: $\rho_p < 1/(d-1)$), then FIFO is stable against *PSM*. Furthermore, they also showed that $\frac{1}{d-1}$ is a tight bound, in the sense that if the load condition overpasses $\frac{1}{d-1}$, then for any value of the delay δ, there exists a network with a maximum diameter d where the delay of some packet exceeds δ. The same bound was obtained by Zhang et al. [35, Theorem 1] by employing a technique based on bounding the maximum delay experienced by any packet at each server.

An analogous result for *AQM* was achieved by Lotker et al. [29, Theorem 4.3]. Namely, they proved that any network is FIFO-stable against *AQM* if the adversarial injection rate r is lower than $\frac{1}{d}$ (in this case, d means the network diameter). Such a bound was reduced to $\frac{1}{d-1}$ by Echagüe et al. [36, Theorem 3.1], also showing that the worst-case end-to-end packet delay is bounded above by $d(b/(1-r(d-1)))$, where (b,r) are the parameters that characterize the *AQM* adversary. The proof of this last stability result is based on finding the conditions that bound the maximum time interval a packet takes to cross a server. If we denote by a_s^i the time instant that packet i arrives at its sth server and denote by Q_s^i the time interval packet i takes to cross its sth server, then we have that $Q_s^i \leqslant r(a_s^i - a_1) + b - (a_s^i - t_B)$, where a_1 is the injection time in the network of the oldest packet present in the sth server at time instant a_s^i and t_B is the last time no later than a_s^i that no packet was scheduled by the sth server. Making some algebra, the previous inequality becomes $Q_i^s \leqslant rdQ + b$ (where $Q = \max_{i,s} Q_i^s$), which implies that if $r < 1/(d-1)$ then $Q < \infty$.

As can be readily observed, the stability bound for *AQM* coincides with the tight bound found for *PSM*. Thus, since *PSM* is more restrictive than *AQM*, this implies that $\frac{1}{d-1}$ is also a tight bound for *AQM*. To show an example of the guarantees provided by this result, observe that the diameter of the baseball network in Fig. 2 is 5 (e.g., $f_0e_1f_1e_0f_0'$). Then, if the adversarial injection rate is below 0.25, the baseball network is FIFO-stable against *AQM*. Similarly, if the maximum sum of the rates of sessions that cross any node is lower

Table 1
Upper bounds on the queue size and the end-to-end packet delay for FIFO under DAG, TREE and RING topologies

Model	DAG		TREE		RING	
	Queue size	Packet delay	Queue size	Packet delay	Queue size	Packet delay
PSM	$O(c^d)$	$O(c^d)$	$O(d)$	$O(d)$	$O(n)$	$O(n)$
AQM	$O(2^{m-1})$	$O(2^{m-1})$	$O(d)$	$O(d)$	$\frac{(b+1)n}{(1-r)}$	$O\left(\frac{bn}{(1-r)^2}\right)$

The parameter n is the number of nodes in the ring, d is the length of the longest path in the network, m is the number of edges in the graph and $c \geqslant 2$.

than 0.25, the baseball network is stable against *PSM*.

By considering a scenario where each link capacity may take on integer values from $[1, C]$ with $C > 1$, which may or may not vary over time (such a scenario was introduced by Borodin et al. in [37] to study stability of some protocols when the link capacity is changing dynamically), Koukopoulos et al. [38, Theorem 15] proved that any network is FIFO-stable against *AQM* if the adversarial injection rate r is lower than $\frac{1}{Cd}$. That is, the performance bound in the dynamic setting has as expense a multiplicative factor of C.

Koukopoulos et al. also studied the problem of the stability of FIFO by considering other parameters that characterize the network structure, in addition to the network diameter. In [39, Theorem 4.1], they showed that FIFO is stable against *AQM* if the adversarial injection rate is lower than or equal to r_G, where r_G is a real number in $(0, 1)$ satisfying the equation $r_G^2 \sum_{i=0}^{d-1} (\alpha + r_G)^i = 1/p$, d being the network diameter, p the minimum number of edge-disjoint paths that cover the network and α the maximum number of ingoing edges in a vertex in the network.

6.3. Pathwise constant injection rates

It is known that, even for arbitrary low injection rates, FIFO can be unstable against both *PSM* and *AQM* (see Section 5). However, one may ask if there are situations, regarding the injection rates, that make FIFO to be stable, either against *PSM* or *AQM*.

Consider the case where the long-term injection rate for each session is constant. More formally, when the number of packets injected by a session p during time interval $[t_1, t_2)$ is bounded not only by Eq. (4) but also by

$$A_p(t_1, t_2) \geqslant \rho_p(t_2 - t_1) - \sigma'_p, \tag{8}$$

where $\sigma'_p \geqslant 0$.

With this type of injection rates (usually called *pathwise constant injection rates*), Gamarnik [15, Theorem 5] showed that FIFO is stable against *PSM*. For the proof, he proved that, given a system $(\mathcal{G}, PSM/AQM, FIFO)$, if its associated adversarial fluid model (as defined in [15]) is stable then the system itself is also stable [15, Theorem 4]. Combining this with Bramson's result [40] that shows that FIFO solutions are stable in fluid networks with pathwise constant arrival rates, Gamarnik states that FIFO is stable against *PSM*. A consequence of this previous result is that, combined with Andrews' instability result in [31, Theorem 1], it means that FIFO exhibits non-monotonicity properties in *PSM*: for pathwise constant injection rates we have stability, but if the adversary can occasionally reduce some session rates then we can also have instability.

6.4. Interfering sessions

Another different way to approach stability in *PSM* considers how the paths followed by the traffic sessions interfere with each other. The *route interference number* of a traffic session p, denoted RIN_p, is defined as the number of other traffic sessions whose paths interfere with the path followed by p, counted with multiplicity if some sessions share several distinct sub-paths along the same path. Chlamtac et al. [41, Corollary 1] proved that if, for each session p, the interpacket time between any two consecutive packets in the same session is at least RIN_p (i.e., for all p, $\rho_p \leqslant 1/(RIN_p + 1)$), then

1. Any network is FIFO-stable.
2. The end-to-end queuing delay for a given traffic session p is bounded by its RIN_p.
3. The backlog at any queue is bounded by $\sum_{i \in I} N_i - \min_i N_i$, where I is the number of input links in the queue and N_i is the number of sessions entering the node via input link i.

The essence of the proofs consisted in showing that, at any given time, there is at most one packet per flow present in each queue. Then, the above mentioned results follow almost directly.

In a subsequent work, Boudec and Hebuterne [42, Theorem 2.3] reduced the backlog queue bound to $\sum_{i \in I} N_i - \max_i N_i$, instead of $\sum_{i \in I} N_i - \min_i N_i$. Essentially, the same result was found independently by Zhang [43, Theorem 2], who also analyzed the tightness for a multipoint-to-point tree topology [43, Theorem 3]. Fig. 4 illustrates with an example the concept of route interference.

- *Open Issue #4*: Clearly, the above mentioned interpacket condition is not tight, in the sense that, in some circumstances, it is possible to increase the injection rate and still preserve the network stability. For instance, in Fig. 4, the three sessions could inject up to a rate of 1/3,

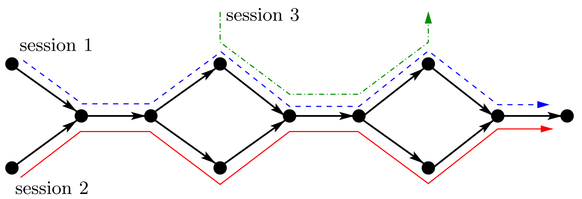

Fig. 4. In this network, $RIN_1 = 4$, $RIN_2 = 4$ and $RIN_3 = 2$. Therefore, if session 1 and session 2 inject up to a rate of 1/5, and session 3 injects up to a rate of 1/3, this network is FIFO-stable.

which is more than what is permitted by the current interpacket condition. A challenging issue is to find how to relax it and still preserve network stability.

6.5. Transforming networks into FIFO-stable

The previous results presented features that, in some sense, are intrinsic to the networks under study. However, it is possible to implement, on top of any given network, a virtual one emulating a FIFO-stable topology and use it for the communication between servers. Bearing this in mind, a simple way to transform any network into a stable topology is to construct a virtual ring, and use only links belonging to the ring. However, this does not appear to be a good solution, since it will greatly affect the system's performance. A more scalable solution consists in implementing a virtual DAG. But this does not guarantee bidirectional graph connectivity among the nodes.

One technique that has been proven to be very convenient for transforming any network into FIFO-stable consists in prohibiting only certain carefully selected *turns* within the original network. The concept of turn was introduced by Schroeder et al. in [44] and it is like a triplet of nodes connected by two links; a prohibited turn (a, b, c) would forbid the forwarding of a packet from link (a, b) to link (b, c) (and vice-versa). Of course, it is necessary to take care to choose the turns that guarantee that the resulting topology will become FIFO-stable, while also preserving network connectivity. The routing of packets can be performed by using a routing protocol, such as *Turnnet* [45], which has been specially designed for such a type of network.

Taking the turn-prohibition approach, in [46] Starobinski et al. devised an algorithm that, while preserving network connectivity, breaks all the existing cycles. Therefore, the resulting topology will behave like a DAG, which is FIFO-stable against *AQM* (see Section 6.1). Fig. 5 illustrates how to transform a network topology into a FIFO-stable network by using turns, while also preserving network connectivity. In addition, they also showed that the maximum number of forbidden turns is bounded by 1/3 of the total number of turns. In a subsequent paper [47], Starobinsky and Karpovsky presented another algorithm (also based on forbidden turns) that allows some of the nodes in a given spanning tree to be replaced by turn-based nodes without creating any cycles. However, in this case the maximum number of forbidden turns increases up to 1/2 the total number of turns.

One drawback of the previous algorithms is that they are centralized. Therefore, to implement them, it is necessary to know the whole network topology. In [48], Echagüe et al. presented a fully distributed turn-based algorithm that prevents the occurrence of cycles and prohibits at most 1/2 the total number of turns. This algorithm was later extended to allow multiple nodes to initiate it in an independent manner, and it can be used to cope with new nodes entering the system as well as with node crashes.

Finally, we remark that the previous algorithms are designed to break *cycles of nodes*. Therefore, paths cannot pass through the same node more than once. Note that such a type of paths usually called *simple paths*, are a slightly restrictive version of *AQM* paths.

- *Open Issue #5*: It would be useful to have a distributed turn-based algorithm that works in a collaborative manner (i.e., where several parts of the network are transformed in an independent fashion and then merged to obtain the full cycle-free network).

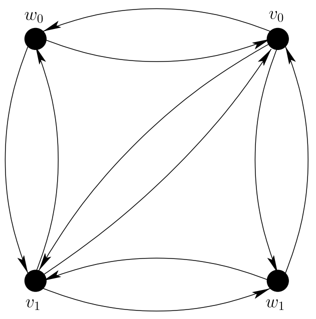

Fig. 5. This network contains cycles and, therefore, may be unstable. However, if we prohibit turns (v_0, w_0, v_1) and (v_1, v_0, w_1) and turns of the (x, y, x) form, then no cycles can occur and consequently the resulting network will become FIFO-stable.

- *Open Issue #6*: An approach that has not yet been considered is how to obtain FIFO-stable topologies that are not necessarily cycle-free. For instance, it is known that the ring is FIFO-stable, even though it has cycles. Perhaps, in some circumstances, these topologies could perform better than cycle-free ones.

7. Deciding FIFO-stability

In this section, we consider the problem of deciding which networks are FIFO-stable and if it is possible to detect them from the knowledge of the network's topological structure.

Since to decide the stability of a network against an adversarial model it is necessary to implicitly quantify over all adversaries and all protocols, it is not clear, a priori, whether stability is a decidable property. Surprisingly, Andrews et al. [11, Section 3.2] showed that, contrary to what one could expect, WC-stability against AQM is a decidable property. The approach followed to obtain the above mentioned result was to prove that if a network G is WC-stable against AQM, so is every *minor* of G. A graph is a minor of G if it can be obtained from G by a sequence of operations involving edge deletions, vertex deletions, and edge contractions (i.e., by merging endpoints together). Then, they used the results obtained by Robertson and Seymour [49–51] to prove the existence of an algorithm to decide WC-stability that runs in time $O(n^2)$ (n being the number of nodes in the network). However, they did not provide an explicit characterization of the networks that are WC-stable.

The first explicit algorithm to decide WC-stability was proposed by Goel in [52, Theorem 2.8]. Namely, he constructed two simple directed graphs that were unstable, and proved that any graph $\mathscr{G}$ is WC-stable against AQM if and only if none of these unstable graphs is a minor of $\mathscr{G}$. In spite of the obvious relevance of this result, a compelling aspect of it is that it is only valid on a slightly restrictive version of AQM. Namely, the paths chosen by the adversaries must be simple paths (i.e., paths where all the nodes, and necessarily all the edges, are different).

In [53, Theorem 3.1], Gamarnik focused on the stability of undirected graphs, proving that any connected undirected graph is WC-stable if and only if it has, at most, two edges. Although now adversaries do not need to use simple paths but all paths allowed by AQM (i.e., edges must be different but not nodes), he also used a slightly restrictive version of AQM in which each undirected link can be crossed in one direction in one time step, contrary to what happens in the original specification of AQM where each link can be traversed, at the same time, in each direction.

Following the same approach as Goel, in [54] Álvarez et al. attained an explicit polynomial time algorithm for deciding WC-stability against the original specification of AQM. Specifically, they constructed three simple directed graphs, and proved that any graph $\mathscr{G}$ is WC-stable against AQM if and only if no extension of these unstable graphs is a minor of $\mathscr{G}$.

At this point, we note that the previous results apply to all work-conserving policies and not only to FIFO, which is also a work-conserving policy. Regarding the stability of the particular FIFO scheduling policy Weinard, based on previous work of Blesa [55], proposed an algorithm to decide FIFO-stability in polynomial time [56, Theorem 2]. More particularly, the algorithm decides that any given graph $\mathscr{G}$ is FIFO-stable against AQM if and only if none of the three simple forbidden graphs ($\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$) in Fig. 6 is a minor of $\mathscr{G}$. He also proposed an algorithm to decide simple path FIFO-stability (i.e., FIFO-stability when the paths are simple paths). For instance, the baseball network in Fig. 2 is FIFO-unstable since the forbidden subdigraph $\mathscr{A}$ is a minor of $\mathscr{G}_B$. In turn, both DAGs and trees are FIFO-stable, since they do not contain any of the three forbidden subdigraphs.

As a consequence of the previous result, it has been evidenced that there are FIFO-stable networks that are not WC-stable [56, Corollary 1] (until recently, the only FIFO-stable networks that had been observed were also WC-stable). Fig. 7 shows a very simple network that is FIFO-stable but not WC-stable.

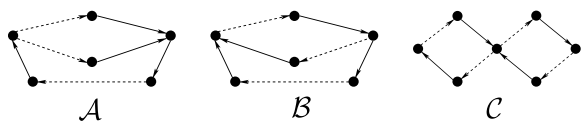

Fig. 6. Set of forbidden subdigraphs for deciding FIFO-stability. A dotted line indicates a simple path of an arbitrary number (including 0) of edges. If a path has 0 edges, the vertices at its ends coincide.

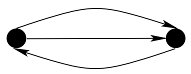

Fig. 7. Example that illustrates a FIFO-stable network, usually called $\mathscr{U}_1$, which is not WC-stable. Specifically, there is an *AQM* adversary $\mathscr{A}$ of rate $r > 0.841$ and a scheduling policy *NTG-LIS* (in *NTG-LIS*, the highest priority is assigned to the packet that still has to cross the smallest number of edges, solving ties by giving priority to the packet that has been in the system the longest time) such that $(\mathscr{U}_1, \mathscr{A}, NTG-LIS)$ is unstable [54, Theorem 3].

- *Open Issue #7*: An interesting open problem is to determine whether or not there is an efficient algorithm that can decide FIFO-stability at reduced injection rates.
- *Open Issue #8*: Contrary to what happens in the case of *AQM*, deciding FIFO-stability against *PSM* is an issue that has not been tackled at all. Thus, parallelizing Weinard's result and finding an efficient algorithm to decide FIFO-stability against *PSM* seems a natural direction to follow.

8. Concluding remarks

Undoubtedly, network stability is an important issue that, in recent years, has attracted the attention of many researchers. Such interest comes from the need to ensure that, as the system runs for an arbitrary length of time, no server will suffer an unbounded queue buildup.

In this survey, we have presented, in a structured manner, the recent advances concerning the stability of FIFO in packet-switched networks. Furthermore, we have also identified some directions open to future research. Although the results obtained are mainly theoretical, they have contributed to form a core of fundamental results, and it is a matter of time before such results will be taken into account for more practical issues, such as the design of network protocols, the analysis of multiprocessor systems, and so forth.

The current state of the art regarding the stability of FIFO constitutes a first step toward the development of a more mature *theory of stability*. We think that in the next few years many more new results will appear, especially in the form of identifying situations where FIFO is stable (e.g., new forms of injection rates, networks structures, etc.), together with work carried out on devising techniques for guaranteeing stability.

Acknowledgement

The authors wish to thank the anonymous referees for several constructive suggestions and corrections.

References

[1] F. Kelly, Networks of queues with customers of different types, Journal of Applied Probability 12 (1975) 542–554.
[2] L. Kleinrock, Queueing Systems vol. I: Theory, vol. 1, John-Wiley & Sons, 1975.
[3] J. Jackson, Jobshop-like queueing systems, Management Science 10 (1) (1963) 131–142.
[4] F. Baskett, K.M. Chandy, R.R. Muntz, F.G. Palacios, Open, closed, and mixed networks of queues with different classes of customers, Journal of the ACM 22 (2) (1975) 248–260.
[5] F. Kelly, Reversibility and Stochastic Networks, Wiley, 1979.
[6] S. Lu, P. Kumar, Distributed scheduling based on due dates and buffer priorities, IEEE Transactions on Automatic Control 12 (36) (1991) 1406–1416.
[7] A. Rybko, A. Stolyar, Ergodicity of stochastic processes describing the operation of open queuing networks, Problems of Information Transmission 28 (1992) 199–220.
[8] M. Bramson, Instability of FIFO queuing networks, Annals of Applied Probability 4 (41) (1994) 414–431.
[9] M. Bramson, Instability of FIFO queuing networks with quick service times, Annals of Applied Probability 4 (3) (1994) 693–718.
[10] T. Seidman, First come first served can be unstable! IEEE Transactions on Automatic Control 4 (39) (1994) 2166–2171.
[11] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu, J. Kleinberg, Universal-stability results and performance bounds for greedy contention-resolution protocols, Journal of the ACM 48 (1) (2001) 39–69 (earlier version appeared in Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, 1996).
[12] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, D. Williamson, Adversarial queueing theory, Journal of the ACM 48 (1) (2001) 13–38 (earlier version appeared in Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 1996).
[13] V. Paxson, S. Floyd, Wide-area traffic: the failure of Poisson modeling, IEEE/ACM Transactions on Networking 3 (3) (1995) 224–226.
[14] R. Cruz, A calculus for network delay. Parts I and II, IEEE Transaction on Information Theory 37 (1) (1991) 114–131, and 131–141.
[15] D. Gamarnik, Using fluid models to prove stability of adversarial queueing networks, IEEE Transactions on Automatic Control 4 (2000) 741–747 (earlier version appeared in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, 1998).
[16] A. Rosén, A note on models for non-probabilistic analysis of packet-switching networks, Information Processing Letters 84 (5) (2002) 237–240.
[17] W. Aiello, E. Kushilevitz, R. Ostrovsky, A. Rosén, Adaptive packet routing for bursty adversarial traffic, Journal of Computer and System Sciences 60 (3) (2000) 482–509 (earlier

version appeared in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998).

[18] M. Andrews, A. Fernández, A. Goel, L. Zhang, Source routing and scheduling in packet networks, Journal of the ACM 52 (4) (2005) 582–601 (earlier version appeared in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001).

[19] C. Álvarez, M. Blesa, J. Díaz, A. Fernández, M. Serna, Adversarial models for priority-based networks, Networks 45 (1) (2005) 23–35 (earlier version appeared in Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 2747, 2003).

[20] M. Blesa, Stability in communication networks under adversarial models, Ph.D. thesis, Universitat Politcnica de Catalunya, 2005.

[21] J.G. Dai, On the positive harris recurrence for multiclass queueing networks: a unified approach via fluid models, Annals of Applied Probability 5 (1995) 49–77.

[22] J.G. Dai, A fluid–limit model criterion for instability of multiclass queueing networks, Annals of Applied Probability 6 (1996) 751–757.

[23] S. Meyn, Transience of multiclass queueing networks via fluid limit models, Annals of Applied Probability 5 (1995) 946–957.

[24] M. Bramson, A stable queueing network with unstable fluid model, Annals of Applied Probability 9 (3) (1999) 818–853.

[25] J.G. Dai, J.J. Hasenbein, J.H. Vande Vate, Stability and instability of a two-station queueing network, Annals of Applied Probability 14 (2004) 326–377.

[26] B. Hajek, Large bursts do not cause instability, IEEE Transactions on Automatic Control 45 (1) (2000) 116–118.

[27] J. Díaz, D. Koukopoulos, S. Nikoletseas, M. Serna, P. Spirakis, D. Thilikos, Stability and non-stability of the FIFO protocol, in: 13th ACM Symposium on Parallel Algorithms and Architectures, 2001.

[28] D. Koukopoulos, M. Mavronicolas, S. Nikoletseas, P. Spirakis, On the stability of compositions of universally stable, greedy contention-resolution protocols, in: 16th International Symposium on Distributed Computing, Lecture Notes in Computer Science 2508, 2002, pp. 88–102.

[29] Z. Lotker, B. Patt-Shamir, A. Rosén, New stability results for adversarial queuing, SIAM Journal on Computing 33 (2) (2004) 286–303 (earlier version appeared in Proceedings of the 14th ACM Symposium on Parallel Algorithms and Architectures, 2002).

[30] R. Bhattacharjee, A. Goel, Z. Lotker, Instability of FIFO at arbitrarily low rates in the adversarial queueing model, SIAM Journal on Computing 34 (2) (2004) 318–332 (earlier version appeared in Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, 2003).

[31] M. Andrews, Instability of FIFO in session-oriented networks, Journal of Algorithms 50 (2) (2004) 232–245 (earlier version appeared in Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, 2000).

[32] M. Andrews, Instability of FIFO in the permanent sessions model at arbitrarily small network loads, in: ACM-SIAM Symposium on Discrete Algorithms, 2007.

[33] L. Tassiulas, L. Georgiadis, Any work conserving policy stabilizes the ring with spatial reuse, IEEE/ACM Transactions on Networking 4 (2) (1996) 205–208.

[34] A. Charny, J.L. Boudec, Delay bounds in a network with aggregate scheduling, in: First International Workshop on Quality of Future Internet Services, Berlin, Germany, 2000.

[35] Z. Zhang, Z. Duan, Y. Hou, Fundamental trade-off in aggregate packet scheduling, IEEE Transactions on Parallel and Distributed Systems 16 (12) (2005) 1166–1177 (earlier version appeared in Proceedings of the 9th International Conference on Network Protocols, 2001).

[36] J. Echagüe, V. Cholvi, A. Fernández, Universal stability results for low rate adversaries in packet switched networks, IEEE Communication Letters 7 (12) (2003).

[37] A. Borodin, R. Ostrovsky, Y. Rabani, Stability preserving transformations: packet routing networks with edge capacities and speeds, Journal of Interconnection Networks 5 (1) (2004) 1–12 (earlier version appeared in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, 2001).

[38] D. Koukopoulos, M. Mavronicolas, P. Spirakis, Performance and stability bounds for dynamic networks, Journal of Parallel and Distributed Computing 67 (4) (2007) 386–399 (earlier version appeared in Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks, 2004).

[39] D. Koukopoulos, M. Mavronicolas, S.E. Nikoletseas, P. Spirakis, The impact of network structure on the stability of greedy protocols, Theory of Computing Systems 38 (4) (2005) 425–460 (earlier version appeared in Proceedings of the 5th Italian Conference on Algorithms and Complexity, Lecture Notes in Computer Science 2653, 2003).

[40] M. Bramson, Convergence to equilibria for fluid models of FIFO queuing networks, Queueing Systems 22 (1–2) (1996) 5–45.

[41] I. Chlamtac, A. Faragó, H. Zhang, A. Fumagalli, A deterministic approach to the end-to-end analysis of packet flows in connection oriented networks, IEEE/ACM Transactions on Networking 6 (4) (1998) 422–431.

[42] J.L. Boudec, G. Hebuterne, Comments on “A deterministic approach to the end-to-end analysis of packet flows in connection oriented network”, IEEE/ACM Transactions on Networking 8 (1) (2000) 121–124.

[43] H. Zhang, A note on deterministic end-to-end delay analysis in connection oriented networks, in: IEEE International Conference on Communications, 1999, pp. 1223–1227.

[44] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, Autonet: a high-speed, self-configuring local area network using point-to-point links, IEEE Journal on Selected Areas in Communications 9 (8) (1991) 1318–1335.

[45] M. Fidler, G. Einhoff, Routing in turn–prohibition based feed–forward networks, in: Networking, Lecture Notes in Computer Science, vol. 3042, 2004, pp. 1168–1179.

[46] D. Starobinski, M. Karpovsky, L. Zakrevski, Application of network calculus to general topologies using turn-prohibition, IEEE/ACM Transactions on Networking 11 (3) (2003) 411–421 (earlier version appeared in Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, 2002).

[47] D. Starobinski, M. Karpovsky, Turn prohibition–based packet forwarding in gigabit ethernets, in: Gigabit Networking Workshop, 2001.

[48] J. Echagüe, M. Prieto, J. Villadangos, V. Cholvi, A distributed algorithm to provide QoS by avoiding cycles in routes, in: First International Workshop on QoS Routing, Lecture Notes in Computer Science, vol. 3326, 2004.
[49] N. Robertson, P.D. Seymour, Graph minors. IV. Tree-width and well-quasi-ordering, Journal of Combinatorial Theory, Series B 48 (1990) 227–254.
[50] N. Robertson, P.D. Seymour, Graph minors. V. Excluding a planar graph, Journal of Combinatorial Theory, Series B 41 (1986) 92–114.
[51] N. Robertson, P.D. Seymour, Graph minors. XIII. The disjoint paths problem, Journal of Combinatorial Theory, Series B 63 (1995) 65–110.
[52] A. Goel, Stability of networks and protocols in the adversarial queueing model for packet routing, Networks 37 (4) (2001) 219–224 (earlier version appeared in Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, 1999).
[53] D. Gamarnik, Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks, SIAM Journal on Computing 2 (32) (2003) 371–385 (earlier version appeared in Proceedings of the 31th Annual ACM Symposium on Theory of Computing, 1999).
[54] C. Álvarez, M. Blesa, M. Serna, A characterization of universal stability in the adversarial queueing model, SIAM Journal on Computing 1 (34) (2005) 41–66 (earlier version appeared in Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures, 2002).
[55] M. Blesa, Deciding stability in packet-switched FIFO networks under the adversarial queuing model in polynomial time, in: 19th International Symposium on Distributed Computing, Lecture Notes in Computer Science, vol. 3724, 2005, pp. 429–441.
[56] M. Weinard, Deciding the FIFO stability of networks in polynomial time, in: 6th International Conference on Algorithms and Complexity, Lecture Notes in Computer Science, vol. 3998, 2006, pp. 81–92.

Vicent Cholvi graduated in Physics from the Universitat de Valéncia (Spain) and received his doctorate in Computer Science in 1994 from the Polytechnic University of Valencia (Spain). In 1995, he joined the Jaume I University in Castellò (Spain) where he is currently an associate professor. His interests are in distributed and communication systems.

Juan Echagüe graduated in Computer Sciences from the Polytechnic University of Valencia and received his doctorated in 2005 from the Jaume I University in Castellón (Spain), where he is currently an associate professor. His current interests are related with providing with quality of service to broadband networks and routing in ad-hoc networks.