Performance of scheduling policies in adversarial networks with non synchronized clocks*

Juan Céspedes† Antonio Fernández† José Luis López-Presa‡ M. Araceli Lorenzo§
Pilar Manzano§ Juan Martínez-Romo† Alberto Mozo§ Anna Puig-Centelles¶
Agustín Santos† Christopher Thraves†‖

Abstract

In this paper we generalize the Continuous Adversarial Queuing Theory (CAQT) model [5] by considering the possibility that the router clocks in the network are not synchronized. Clearly, this new extension to the model only affects those scheduling policies that use some form of timing. First, if all clocks run at the same speed, maintaining constant differences, we show that all universally stable policies in CAQT that use the injection time and the remaining path to schedule packets remain universally stable. These policies include, for instance, Shortest in System (SIS) and Longest in System (LIS). Then, if clock differences can vary over time, but difference is bounded, we show the universal stability of SIS and a family of policies related to LIS. The bounds we obtain in this case depend on the maximum difference between clocks. We then present a new policy that we call Longest in Queues (LIQ), which gives priority to the packet that has been waiting the longest in edge queues. This policy is universally stable and, if clocks maintain constant differences, the bounds do not depend on them. To finish, we provide with simulation results that compare the behavior of some of these protocols in a network with stochastic injection of packets.

*Partially supported by the Spanish MEC under grants TIN2005-09198-C02-01 and TSI2006-07799, and the Comunidad de Madrid under grant S-0505/TIC/0285.

†GSyC, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid, Spain.
‡EUITT, Universidad Politécnica de Madrid, 28031 Madrid, Spain.
§EUI, Universidad Politécnica de Madrid, 28031 Madrid, Spain.
¶Universitat Jaume I, 12071 Castellón, Spain.
‖Corresponding author. Ph: +34-914887054 Fax: +34-914887049 E-mail: cbthraves@gmail.com. Partially supported by Universidad de Chile, Mecesup fellowship and Anillo en Redes.

1. Introduction

Stability is a requirement in a packet switched network in order to be able to provide some quality of service. Stability means that the amount of traffic that is being routed in the network is always bounded. It is well known that an appropriate scheduling of packets is fundamental in order to guarantee stability [2]. A fundamental question is to identify scheduling policies that are able to guarantee stability, and among these, policies that guarantee good quality of service (e.g., latency).

The study of the capability of scheduling policies to guarantee stability under worst-case situations has been done with adversarial models [6, 9, 10, 2, 7]. In these models, the arrival of packets to the network is controlled by an adversary which defines, for each packet, the instant and node in the network where it is injected, and very often, its path in the network. To avoid the overload of links in the network, the number of packets that the adversary can inject is bounded. As we said, the main objective of these models is to explore the ability of scheduling policies at the routers to maintain the system stable or provide good quality of service even in the worst conditions.

Adversarial models. In this general framework two main adversarial models have been defined. In the Permanent Session Model [9, 10, 7], also known as the (σ, ρ)-regulated injection model, all the traffic in the network is grouped in sessions, whose route and maximum packet injection rate are defined by the adversary. The adversary is restricted on the rates σ_i that it can assign to sessions in the sense that the total rate of the sessions that cross a link does not saturate the link. Additionally, the adversary is in control of the arrival of packets. The adversary always tries to create instability or to increase the maximum latency experienced by packets. There have been many works exploring this model, and proposing stable policies with different guarantees of quality of service (e.g., [9, 10, 11, 12, 7]). It is also

interesting to observe that, in this model, FIFO (or FCFS), which is the most popular scheduling policy by far, can be made unstable at any constant network load [1].

The Temporary Session Model, commonly known as the Adversarial Queuing Theory (AQT) model [6, 2], relaxes the restriction that packets are assigned to sessions. This relaxation is similar to allowing the adversary to dynamically change the sessions over time. In this model, each packet is injected with its own path and the only restriction on the adversary is that it cannot overload any link, in an amortized sense. The AQT model assumes that the network evolves in steps, and in each step at most one packet crosses each link. The Continuous AQT (CAQT) model, recently presented in [5], is an extension of the AQT model in which packets can have arbitrary sizes, and links have different bandwidths and propagation delays. This model is closer to reality than AQT, and due to the fact that AQT is a particular case of CAQT, the instability results obtained for AQT remain valid for CAQT. Additionally, it was shown in [5] that many positive results under the AQT model also hold under the CAQT model. The system model we consider in this paper is strongly based on the CAQT model.

Time-based scheduling policies. From all the policies that have been shown to be stable in all networks (which we call *universally stable*) under the AQT and CAQT models, those that seem to provide the lowest end-to-end packet delays are based on timing. In fact, it has been shown by Weinard [15] that, for any policy in the family of Without-Time-Stamping strategies, there are n-node under-loaded networks in which the delays and queue sizes are $2^{\Omega(\sqrt{n})}$. A policy of this family is assumed to know the network topology, and it assigns the priority of a packet as a function of its path and the number of edges it already crossed. This implies that, in general, it is convenient to use some timing information for scheduling. Unfortunately, the simplest time-based policies, can also suffer of large delays. For instance, for Shortest in System (SIS), the policy that gives the highest priority to the newest packet in the network, there are networks in which the delays and queue sizes are $2^{\Omega(\sqrt{n})}$ [2]. Similarly, for Longest in System (LIS), the policy that gives the highest priority to the oldest packet in the network, there are networks with diameter d in which the delays and queue sizes are $2^{\Omega(d)}$ [4].

The good news are that time-based policies can in fact provide low delay guarantees. For instance, in [2] it is presented a randomized scheduling algorithm that guarantees delays polynomial on the network parameters. This algorithm basically uses a longest-in-system strategy with random permutations. The deterministic scheduling algorithm with polynomial delays presented in [3] uses a similar approach. Additionally, there are simulation studies [13] which show that LIS may in fact behave much better in practice than one may expect from the lower bounds mentioned above.

The Non Synchronized CAQT model. The above mentioned results for time-based scheduling policies are obtained in network models in which it is implicitly assumed that each node in the network has a local clock to provide the time, and that all these clocks are synchronized and provide the same time. However, this latter assumption is not realistic in practice, since the oscillation frequency of each timer is different, what produces different *clock drifts*. The consequence of these drifts is that it is not unusual that different clocks provide different times. The differences between the clock times is what we call *clock skews*. In practice, in order to limit the effect of clock drifts, and to bound the clock skews, there are mechanisms, like the Network Time Protocol (NTP), that allow the resynchronization of clocks.

In this paper we propose a model of adversary in which clocks do not need to be synchronized. We call the new model *Non Synchronized CAQT* (NSCAQT), since it is basically the CAQT model with this additional generalization. Under this model we will study the behavior of different queue scheduling policies which depend on time and can be affected both by clock skews and clock drifts. To study these policies, we need to make assumptions on how they use the local clocks. For instance, for LIS and SIS we will assume that packets are assigned the local clock value of their injection node at their injection time, value that they carry with them and is used for scheduling.

In this paper we will study two main variations of the NSCAQT model. In the first one we assume that the system has clock skews but no clock has drifts. Hence, in this model clocks do not have the same time, but their time differences remain constant. We call this the *NSCAQT model with constant skews*. The second model we will study is a model in which skews can vary over time, but there is a bound on the maximum difference between the time of the clocks. We call this model the *NSCAQT model with bounded skews*, and is very suited to model a network in which a protocol like NTP is used to periodically resynchronize all the clocks. A third natural model which we do not explore here is one in which clock drifts are present and no resynchronization mechanism guarantees that the skews are bounded. This model is left for future study.

Contributions. The main contribution of this work is the detailed definition of a model in which clocks need not be synchronized. We have not found a model that considers this possibility in any previous adversarial model. Then, the first result we provide is on the NSCAQT model with constant skews. We study under this model scheduling policies that assign priorities to packets based on their injection

times and their remaining paths. For these policies we will show how the NSCAQT system can be transformed into a CAQT system by changing the topology of the network and the adversary. As a consequence, we conclude that any such policy that is universally stable under CAQT is also universally stable under NSCAQT with constant skews. [1]

We then explore universal stability under the NSCAQT model with bounded skews. In this model, we prove the universal stability of the SIS policy. We also define a family of policies, that include LIS as a particular case, and show that all the policies in the family are universally stable in this model as well. We call this family of policies *Longest in System considering Path* (LISP). Policies from the LISP family assign packet priorities depending on both the injection time and the number of edges already traversed by the packet.

Unfortunately, for the universally stable policies that we identified with the previously mentioned results, all the upper bounds on delays and queue sizes that we could prove depend on the clock skews. In fact, in several cases it can be easily shown that these parameters become larger as the skews grow. Then, the question is whether there are policies whose performance does not depend on the clock skews. We introduce a new policy, *Longest in Queues* (LIQ), which gives priority to the packet that has been waiting in queues the longest. We show that this policy is universally stable in the NSCAQT model with bounded skews. More interestingly, we show that in the NSCAQT model with constant skews this policy has an upper bound on the end-to-end delay that does not depend on the skews and is close to that of LIS in CAQT.

Finally, we present some simulations which try to shed some light on the behavior of LIS, SIS, and LIQ in a network with stochastic arrival patterns, instead of adversarial, in the NSCAQT model with constant skews. The results show that, as expected by analysis, LIQ is not affected by the clock skews, and presents the best performance from among the three policies.

Structure. The structure of the rest of the paper is the following. In Section 2 we define the NSCAQT model in detail and introduce some notation to be used on the paper. In Section 3 we study stability under the NSCAQT model with constant skews. In Section 4 we study SIS and the policies in LISP under the NSCAQT model with bounded skews. In Section 5 we explore the performance of LIQ under both NSCAQT models. In Section 6 we present the simulations that have been done. Finally, in Section 7 we present some conclusions.

[1] Note that due to space limitations several proofs have been omitted and can be found in [8]

2. System model

Like most previous adversarial network models, the NSCAQT system model has three major elements: an underlying network $\mathcal{G}$, a scheduling policy used $\mathcal{P}$, and an adversary $\mathcal{A}$. With these elements, the evolution of the system can be seen as a game between the adversary, which injects packets in the network trying to create instability, and the scheduling policy, that decides which packets move along their paths in the network, trying to prevent instability. The model of system considered in this paper is a direct extension of that presented in [5].

The network. In this model a network is modeled by a directed graph $\mathcal{G}$, formed by a set of nodes $V(\mathcal{G})$, representing the hosts and routers, and a set of edges $E(\mathcal{G})$, representing links between the nodes. Each link e of the network has associated a positive finite bandwidth B_e, which determines the transmission speed of the link, and a finite propagation delay $P_e \geq 0$. We use a specific notation for the largest propagation delay and smallest bandwidth as follows: $P_{\max} = \max_{e \in E(\mathcal{G})}\{P_e\}$ and $B_{\min} = \min_{e \in E(\mathcal{G})}\{B_e\}$. (Since $\mathcal{G}$ is finite, these values are well defined.)

The bounded adversary. In a system with network $\mathcal{G}$, the adversary $\mathcal{A}$ defines the traffic pattern, continuously deciding which packets are injected. Additionally, for each packet p, the adversary chooses the moment of injection $T_0(p)$, the source node $v_0(p)$, the destination node $v_{d_p}(p)$, and the path the packet has to traverse $\Pi(p) = (e_0(p), e_1(p), ..., e_{d_p-1}(p))$. (When clear from the context, we may omit the packet p from the notation.) Notice that d_p represents the length of the path packet p has to traverse. We assume that a packet path is edge-simple, i.e. it does not contain the same edge more than once, although it can visit the same node several times[2]. We denote by $d_{\max}$ the length of the longest path of a packet, which is clearly open bounded by the length of the longest edge-simple path in the network.

Although the adversary controls the traffic arrival, it is restricted on the load that it can inject to the system. We assume that the injection of a packet is instantaneous. Then, if $N_e(I)$ represents the number of bits of the packets which want to cross edge e injected by $\mathcal{A}$ during an interval I, it must satisfy that

$$N_e(I) \leq r|I|B_e + b = (1-\varepsilon)|I|B_e + b \qquad (1)$$

[2] This assumption does not decrease the generality of the model in terms of universal stability of policies, since it is known that a system in which packets may traverse the same edge several times can be simulated by another system with only edge-simple paths [1].

for every edge e and for every time interval I. We denote by r, $0 < r \leq 1$, the long term rate (load) the adversary can impose on the system. For convenience we sometimes use the notation $r = 1 - \varepsilon$, for $\varepsilon \geq 0$. The parameter b, $b \geq 1$, is the burstiness allowed to the adversary injections, which is the excess of bits allowed to arrive at any time during the complete game. An adversary that satisfies this condition is called an (r, b)-adversary.

Packets, queues, and buffers. Packets are sequences of bits of possibly different sizes. We denote by L_p the size in bits of a packet p and by $L_{\max}$ the maximum size of a packet. Because of the above restriction (1) on the adversary and the assumption of instantaneous injection of packets, it can be easily observed that $L_{\max} \leq b$. Note that b is also an upper bound on the number of packets that the adversary can inject instantaneously (which is achieved in the improbable case that all packets have size 1).

Packets in the system follow their path traversing one edge after the other toward their destination. As explained in [5], in every node in the network there is a reception buffer for each edge entering the node and an output queue for each edge leaving the node. The output queue of an edge has unbounded capacity and holds the packets that are ready to cross this edge. The scheduling policy of the edge's output queue chooses the next packet to cross the edge from those in this output queue. The reception buffer is used to store the received portion of a packet until it has been completely received. Then, the packet is placed instantaneously by a dispatcher in the corresponding output queue or it disappears from the system if it already reached its destination.

Note that once a packet p starts to cross edge e, it will spend $\frac{L_p}{B_e} + P_e$ units of time to completely cross it. As parameter of the network we have the greatest amount of time that a packet can spend crossing an edge, denoted $D_{\max}$, and defined as $D_{\max} = \max_{e \in E(\mathcal{G})} \left\{ \frac{L_{\max}}{B_e} + P_e \right\} \leq \frac{b}{B_{\min}} + P_{\max}$.

Clocks. As we said, the main difference between the NSCAQT model we propose and previous models [6, 7, 2, 5] is that we consider here the impact of clocks not being synchronized on the performance. In order to make the model as general as possible, we assume that the output queue of each edge has its own internal clock, called *edge clock* (this is clearly more general than assuming one clock per node). Additionally, we assume there is an external reference clock which is always on time. We refer to this clock as the *real clock* and we say that it provides the *real time*. We assume that the adversary has access to both the edge clocks and the real clock, while the scheduling policy at a given edge has only access to the clock of that edge.

The difference between the real clock and the edge clock of e at real time t is what we call the *clock skew* of e's edge clock at time t, and is denoted by $\phi_e(t)$. Then, if t_e denotes the value of the edge clock of e at real time t, we have $t_e = t - \phi_e(t)$. If this value changes over time, we say that the edge clock has a *drift*. If an edge clock has no drift we omit the time and denote its skew by ϕ_e. Note that, at any given time, the skew of an edge clock can be positive or negative. However, for convenience we assume that these skews are all non-negative if all edge-clock skews are lower bounded. We can do this freely since the real clock is not available to the scheduling policies and does not interfere in the relation between edge clocks.

We denote by $T_i(p)$, $0 \leq i < d_p$, the real time at which a packet p arrives to the output queue of the edge $e_i(p)$. Due to clock skews, according to edge $e_i(p)$'s clock, the instant when packet p arrives to the output queue is $T_i(p) - \phi_{e_i(p)}(T_i(p))$. Additionally, we denote by $T_{d_p}(p)$ the time at which p is completely received at its destination and leaves the system.

Scheduling policies. As we said above, the scheduling policy is in charge of deciding, whenever a link e is available, which packet from those in the output queue of e must be sent next across e. In this paper we only consider distributed work-conserving time-based scheduling policies. We say that policies are distributed if they do not use the state (and in particular the clock) of other edges to make scheduling decisions. Policies are work-conserving (also called greedy) if they always send a packet across the link as long as the edge's output queue is not empty. Finally, we only consider time-based policies, which are policies that use the edge clocks for scheduling. Note that policies that are not time-based are not affected by clock skews and drifts.

We will only consider in this work systems in which all the queues use the same scheduling policy. The study of systems under the NSCAQT model in which different queues may use different scheduling policies is left for future work.

Two of the most studied distributed work-conserving time-based scheduling policies are Longest in System (LIS) and Shortest in System (SIS). The LIS policy gives the highest priority to the packet that has been in the system for the longest time, while the SIS policy gives the highest priority to the packet that has been in the system for the shortest time. These definitions do not clearly show the use these policies make of the edge clocks. For that, we need to look at the natural implementation of these policies: upon arrival of a packet p into the system, it is assigned a timestamp, $TS(p)$, which p carries with it. Then, the LIS and SIS policies only compare the timestamps of the packets to decide which to schedule next. The edge scheduler in LIS gives the highest priority to the packet with the smallest times-

tamp, while in SIS it gives the highest priority to the packet with the largest timestamp. Note that when clocks are not synchronized, these timestamps are not accurate, since the timestamp for a packet p is $TS(p) = T_0(p) - \phi_{e_0(p)}(T_0(p))$. These two policies have been proved to be universally stable in [5] for the CAQT model, where $\phi_e = 0$ for all $e \in E(\mathcal{G})$.

In addition to these two well-known policies, we will study a family of policies derived from LIS, that we call Longest in System considering Path (LISP). In the policies of this family, packets carry their timestamp and the length of the traversed path, so that at its edge $e_i(p)$, packet p is assigned a priority label of the form $PL(p,i) = TS(p) + f(i)$, where $f(i)$ is a function which assigns a real number to each $i \in \{0, 1, \ldots, d_{\max} - 1\}$, being i the number of crossed edges. A policy $\mathcal{P}_f$ is in LISP if at each queue it gives the highest priority to the packet p with the smallest value $PL(p,i)$. Notice that when $f(i) = 0$ for all i, $\mathcal{P}_f$ is equivalent to LIS. Since the function f is defined over the finite set of number of edges crossed by a packet, it has a maximum and a minimum values, that we denote by $f_{\max}$ and $f_{\min}$, respectively.

Finally, we will consider a new policy named Longest in Queues (LIQ). In this policy the highest priority is assigned to the packet that has been waiting the longest in all the output queues it has visited. In our model NSCAQT, in which clocks are not synchronized, we assume that the time in queues is measured locally at each output queue. The time a packet p waits at the edge e's queue is the difference between the value of the edge clock when the packet arrives and the value when it starts being transmitted, or the current value of the edge clock if it is still waiting. The time used to schedule p is the sum of these waiting times in all the visited queues.

System stability. To study stability and performance in packet switching networks, we introduce the concept of a $(\mathcal{G}, \mathcal{P}, \mathcal{A})$ system to represent the game played between an adversary $\mathcal{A}$ and the packet scheduling policy $\mathcal{P}$ over the network $\mathcal{G}$. In the NSCAQT model, a system $(\mathcal{G}, \mathcal{P}, \mathcal{A})$ is stable if the maximum number of packets (or bits) present in the system is bounded at any time by a constant that may depend on system parameters: the network, the adversary or the policy. A policy $\mathcal{P}$ is *universally stable* if the system $(\mathcal{G}, \mathcal{P}, \mathcal{A})$ is stable on every network $\mathcal{G}$ and against every (r,b)-adversary $\mathcal{A}$ with $r < 1$ and $b \geq 1$.

3. Stability of policies for constant clock skews

In this section we study the case in which all the edge clocks have zero drift, so that ϕ_e is constant. This framework allows to assure the stability in a non synchronized system if there is stability in a synchronized system for many policies, in particular for those policies that only depend on the injection time and the remaining path of the packets.

We present a proof by transformation of this case. We start from a $(\mathcal{G}, \mathcal{P}, \mathcal{A})$ system with non synchronized clocks, where $\mathcal{A}$ is an (r,b)-adversary, $r \leq 1$. Then, we vary the network $\mathcal{G}$ and the adversary $\mathcal{A}$ to obtain a synchronized system $(\mathcal{G}', \mathcal{P}, \mathcal{A}')$, where $\mathcal{A}'$ is an (r,b')-adversary, so that if $(\mathcal{G}', \mathcal{P}, \mathcal{A}')$ is stable, then $(\mathcal{G}, \mathcal{P}, \mathcal{A})$ is also stable. The details of the proof can be found in [8].

Theorem 1 *[8] Let $(\mathcal{G}', \mathcal{P}, \mathcal{A}')$ be the synchronized system in the CAQT model obtained from the non synchronized system $(\mathcal{G}, \mathcal{P}, \mathcal{A})$ through the above process. Let $\mathcal{P}$ be a scheduling policy which considers only the time of injection of the packets and the paths that the packets still have to traverse. Then, $(\mathcal{G}, \mathcal{P}, \mathcal{A})$ is stable if and only if $(\mathcal{G}', \mathcal{P}, \mathcal{A}')$ is stable.*

Corollary 1 *The scheduling policies that are universally stable in CAQT and only consider the times of injection and the paths that the packets still have to traverse are universally stable in the NSCAQT model with constant clock skews.*

4. Stability of policies for bounded clock skews

In this section we will study the case in which clocks may experience drifts. Hence, we assume here that the clock skews are not necessarily constant. However, the maximum difference between real time and any edge clock is bounded. As we said, this model fits naturally with a system in which edge clocks are periodically resynchronized, for instance via NTP.

In this section we will again adapt the real time reference clock, in order to simplify the analysis and the presentation. Like in the previous section, we will assume that all clock skews are non-negative, i.e., for any edge e and any time t, $\phi_e(t) \geq 0$. Additionally, since we assume that skews are bounded, we can safely define $\phi_{\max} = \max_{e,t}\{\phi_e(t)\}$.

Under these assumptions, we show first that the SIS policy is universally stable (universal stability for CAQT was proved in [5]). Then we consider the family of policies LISP, to which LIS belongs, and we also show its universal stability for the NSCAQT model, and consequently, for CAQT (which was previously unknown in general).

Universal stability of SIS. As we said above, the SIS scheduling policy gives the highest priority to the packet which has been in the system for the shortest time. Additionally, we assume that SIS is implemented by making the first edge in the path of a packet p to attach the arrival time to the packet. Since this arrival time is obtained from the local edge clock, SIS can be affected by clock skews.

Theorem 2 *[8] Let $\mathcal{G}$ be a network and $d_{\max}$ the length of its longest edge-simple directed path, let $\mathcal{A}$ be an (r,b)-adversary with $r = 1-\varepsilon < 1$ and $b \geq 1$. Then the system $(\mathcal{G}, SIS, \mathcal{A})$ is stable under the NSCAQT model with bounded clock skews, no queue ever contains $k_{d_{\max}-1} + L_{\max}$ bits, and no packet spends more than*

$$\frac{d_{\max} b + \sum_{i=0}^{d_{\max}-1} k_i}{\varepsilon B_{\min}} + d_{\max} D_{\max}$$

units of time in the system.

Corollary 2 *SIS is universally stable in the NSCAQT model bounded clock skews.*

Universal stability of LISP. In this subsection we explore the stability of the new family of policies LISP defined in Section 2, which is based on the injection time and the number of edges already crossed by a packet. As described there, a policy $\mathcal{P}_f$ in LISP assigns to each packet p at the queue of its edge $e_i(p)$ a priority label $PL(p,i) = TS(p) + f(i)$, and gives the highest priority to the packet with the smallest label.

Theorem 3 *[8] Let $\mathcal{G}$ be a network and $d_{\max}$ the length of its longest edge-simple directed path, let $\mathcal{A}$ be an (r,b)-adversary, with $r = 1-\varepsilon < 1$ and $b \geq 1$, and let $\mathcal{P}_f$ be a policy in LISP. Then the system $(\mathcal{G}, \mathcal{P}_f, \mathcal{A})$ is stable under the NSCAQT model with bounded clock skew, and no packet spends more than*

$$\frac{1-\varepsilon^{d_{\max}}}{\varepsilon^{d_{\max}}} \left(f_{\max} - f_{\min} + \phi_{\max} + \frac{D_{\max}}{1-\varepsilon} \right)$$

units of time in the system.

Corollary 3 *Any protocol in LISP, and in particular LIS, is universally stable under the NSCAQT model with bounded clock skew, and hence under the CAQT model.*

5. Universal stability of LIQ

In previous sections we have shown how several policies are universally stable in the NSCAQT models with constant and bounded clock skews, respectively. Unfortunately, the bounds on end-to-end packet latencies we derived were dependent on the maximum clock skew that can occur in the system. This means that in a system with high maximum skew, the latencies can be very high. It is easy to construct examples for policies like SIS and LIS in which this can be observed.

In this section we study a new policy named LIQ, which gives the highest priority to the packet that has been waiting in output queues for the longest time. We prove that LIQ is universally stable in the NSCAQT model with but bounded clock skews. The bad news is that in this case the end-to-end latency bound we obtain depends also on the maximum skew. The good news is that for the NSCAQT model with constant clock skews LIQ is universally stable, and the bound does not depend on the maximum skew, and it is similar to that obtained with LIS in a synchronized system.

As in previous sections, we assume that $\phi_e(t) \geq 0$ for all e and t. Then, we define $\phi_{\min}(e) = \min_t\{\phi_e(t)\}$ and $\phi_{\max}(e) = \max_t\{\phi_e(t)\}$. Finally, let $\Delta\phi = \max_e\{\phi_{\max}(e) - \phi_{\min}(e)\}$. Observe that in the model of constant skews, $\Delta\phi = 0$.

Theorem 4 *[8] Let $\mathcal{G}$ be a network with d_{max} the length of its longest edge-simple directed path, let $\mathcal{A}$ be an (r,b)-adversary with $r = 1-\varepsilon < 1$. Then (1) the system $(\mathcal{G}, LIQ, \mathcal{A})$ is stable under the NSCAQT model with bounded clock skews, and no packet spends more than*

$$\frac{1-\varepsilon^{d_{\max}}}{\varepsilon^{d_{\max}}} \left(d_{\max}(2\Delta\phi + D_{\max}) + \frac{D_{\max}}{1-\varepsilon} \right)$$

units of time in the system, and (2) the system $(\mathcal{G}, LIQ, \mathcal{A})$ is stable under the NSCAQT model with constant clock skews, and no packet spends more than

$$\frac{1-\varepsilon^{d_{\max}}}{\varepsilon^{d_{\max}}} \left(d_{\max} D_{\max} + \frac{D_{\max}}{1-\varepsilon} \right)$$

units of time in the system.

Corollary 4 *LIQ is universally stable under the NSCAQT model with bounded clock skews, and hence under the CAQT model.*

6. Simulations

In order to partially evaluate the theoretical results we have developed several simulation experiments. All the experiments in this article have been carried out using the J-Sim discrete event simulator [14]. J-Sim has been designed to simulate network behaviors in a realistic way, including propagation delays, packet processing times, etc. The J-Sim package has been modified in several ways, mainly to adapt it to our model. First, the traffic generator has been modified in order to ensure that destinations are uniformly distributed over all nodes in the network. Then, the sink monitor has also been changed in order to log several parameters that are not stored by J-Sim by default (e.g., the mean and the variance of the packet delay and the queue size, and samples of these values chosen at random). Also, the routing algorithm has been replaced and some scheduling policies discussed in this paper have been implemented.

The network topology used in all the experiments is an 11×11 torus, in which every node is, at the same time, router, source, and sink of packets. Each node periodically generates new packets, whose destination is chosen randomly and uniformly among the nodes of the network. The routing is deterministic, so that the traffic is balanced among all the links. We have adjusted the average load of the network to 99% in the simulations because we are interested on the response of the network with high load levels. In our experiments all the queues are of unbounded size. Links have no propagation delays because we consider it negligible. The link bandwidth is set to 100 Kbps and the packet size is 125 bytes (105 bytes plus 20 IP header bytes). The reason why we set this bandwidth is for not overloading the simulator, and the packet size is chosen in order to obtain an entire number of packets per second (100000 bits/s is equal to 12500 bytes/s). The simulation experiments have been run for 6000 seconds, ignoring the first 1000 seconds in the analysis of the results.

We assign to local node clocks (all output edge clocks in a node are the same) different skews following a normal distribution with a mean value of 0 seconds and a standard deviation of up to 10^5 milliseconds. Before starting the experiment, each node randomly chooses a constant skew for its clock from the above distribution.

Figure 1 shows the mean and maximum latencies experienced by packets that cross 10 links when, as said before, the distribution of clock skews have standard deviations ranging from 0 to 10^5 milliseconds. As expected, LIQ is not affected by clock skews, since it does not consider injection time (which could be affected by clock skews), but waiting time, which is always correctly computed since all clocks run at the same speed (there are no drifts). It is also noticeable that the mean and the maximum latencies of LIQ are very low, which is not the case for the other policies, especially when clock skews grow. At first sight, it seems a bit paradoxical the fact that the mean latency with SIS decreases when skews grow. However, this behavior may be attributed to the fact that increasing skews randomizes the behavior of the policy. Note that the maximum latency with SIS does not seem to be significantly affected by the skew variation. LIS suffers from increasing clock skews, since its effectiveness relies on the accuracy of clocks. When skews grow, LIS clearly degrades its performance. Finally, we want to emphasize the great distance between the mean and the maximum in the case of SIS, and in the case of LIS for large skews.

Figure 2 shows how the number of hops a packet needs to reach its destination affects the latency. Here we see that LIS when all clocks are synchronized (no skews) and LIQ give analogous results, and behave quite uniformly on the number of hops. It is again noticeable that the mean and the maximum are much closer in the cases of LIS with no

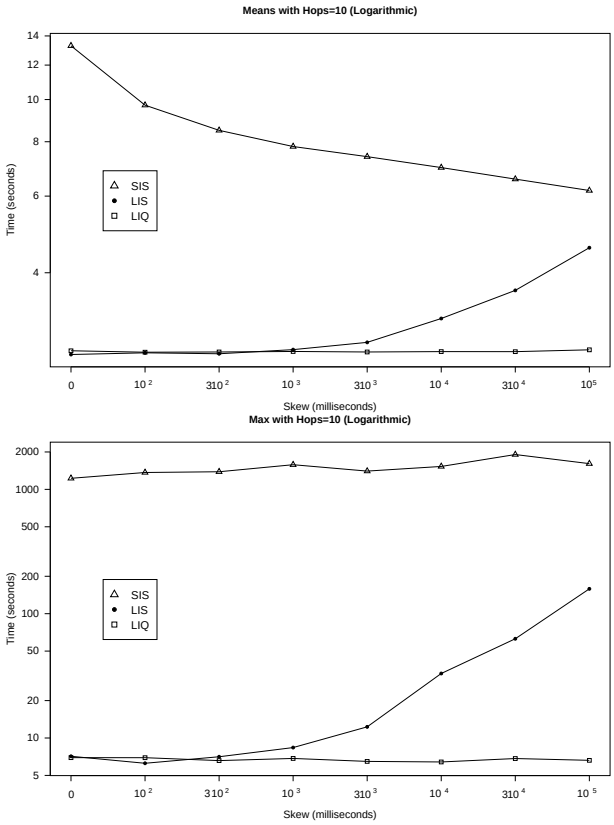

Figure 1. Average latency experienced by packets that cross 10 links with policies LIS, SIS, and LIQ under distribution of skews with different standard deviations.

skews and LIQ, than in the other cases (between one and two orders of magnitude). Observe that, while with LIS the slope of the curve increases with the skew, with SIS the slope decreases.

7. Conclusions

We introduce a model to analyze communication networks in which router clocks are not necessarily synchronized. This new model is an extension of the Continuous Adversarial Queuing Theory (CAQT) model [5]. We show that, if all clocks run at the same speed, all universally stable policies in CAQT that use the injection time and the remaining path to schedule packets remain universally stable. Furthermore, we show the universal stability of SIS and a family of policies related to LIS, when time differences can vary over time (but this difference is bounded). We then present a new policy which gives priority to the packet that has been waiting the longest in edge queues (LIQ). We show

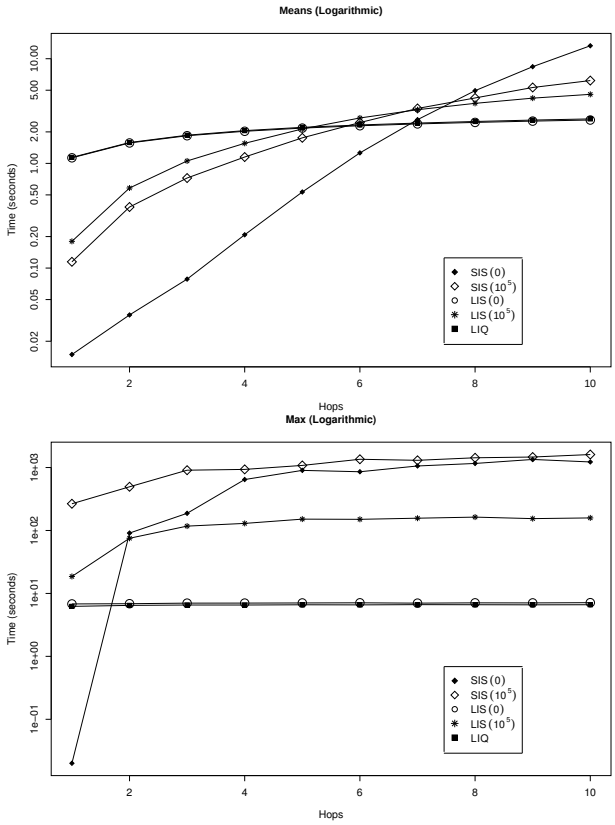

Figure 2. Average latency experienced by packets with policies LIS, SIS, and LIQ under normal distribution of skews with standard deviations of 0 and 100000.

that LIQ is universally stable and, if clocks maintain constant differences, the bounds do not depend on them. To finish, we provide simulations that compare the behavior of some of these protocols in a network with stochastic injection of packets.

References

[1] M. Andrews. Instability of fifo in the permanent sessions model at arbitrarily small network loads. In *SODA*. ACM Press, 2007.

[2] M. Andrews, B. Awerbuch, A. Fernández, T. Leighton, Z. Liu, and M. Kleinberg. Universal-stability results and performance bounds for greedy contention-resolution protocols. *J. ACM*, 48(1):39–69, 2001.

[3] M. Andrews, A. Fernández, A. Goel, and L. Zhang. Source routing and scheduling in packet networks. *J. ACM*, 52(4):582–601, 2005.

[4] M. Andrews and L. Zhang. The effects of temporary sessions on network performance. *SIAM J. Comput.*, 33(3):659–673, 2004.

[5] M. J. Blesa, D. Calzada, A. Fernández, L. López, A. L. Martínez, A. Santos, M. J. Serna, and C. Thraves. Adversarial queueing model for continuous network dynamics. *Theory of Computing Systems*, in press.

[6] A. Borodin, M. Kleinberg, P. Raghavan, M. Sudan, and P. Williamson. Adversarial queuing theory. *J. ACM*, 48(1):13–38, 2001.

[7] J.-Y. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[8] J. Céspedes, A. Fernández, J. L. López-Presa, M. A. Lorenzo, P. Manzano, J. Martínez-Romo, A. Mozo, A. Puig, A. Santos, and C. Thraves. Performance of scheduling policies in adversarial networks with non synchronized clocks. *Reports on Systems and Communications*, 6(6), 2006. http://gsyc.es/tr

[9] R. L. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114 – 131, 1991.

[10] R. L. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132 – 141, 1991.

[11] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344 – 357, 1993.

[12] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Transactions on Networking*, 2(2):137 – 150, 1994.

[13] A. Santos, A. Fernández, and L. López. Evaluation of packet scheduling policies with application to real-time traffic. In *Actas de las V Jornadas de Ingeniería Telemática, JITEL 2005*, Vigo, Spain, 2005.

[14] J-Sim simulator. http://www.j-sim.org/.

[15] M. Weinard. The necessity of timekeeping in adversarial queueing. In Sotiris E. Nikoletseas, editor, *WEA*, volume 3503 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2005.