The Robustness of Stability under Link and Node Failures *

Carme Àlvarez Maria Blesa Maria Serna

ALBCOM Research Group
Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Jordi Girona 1-3, Ω Building, E-08034 Barcelona, Spain
{alvarez,mjblesa,mjserna}@lsi.upc.edu

July 20, 2007

Abstract

In the area of communication systems, *stability* refers to the property of keeping the amount of traffic in the system always bounded over time. Different communication system models have been proposed in order to capture the unpredictable behavior of some users and applications. Among those proposed models the *adversarial queueing theory* (AQT) model turned out to be the most adequate to analyze an unpredictable network. Until now, most of the research done in this field did not consider the possibility of the adversary to produce failures on the network structure. The adversarial models proposed in this work are inspired by the AQT model, but incorporating the possibility of dealing with node and link failures provoked by the adversary. Such failures produce temporal disruptions of the connectivity of the system and increase the collisions of packets in the intermediate hosts of the network, and thus the average traffic load. Under such a scenario, the network is required to be equipped with some mechanism for dealing with those collisions.

In addition to proposing adversarial models for faulty systems we study the relation between the robustness of the stability of the system and the management of the queues affected by the failures. When the adversary produces link or node failures the queues associated to the corresponding links can be affected in may different ways depending on whether they can receive or serve packets or they can not. We show that, in most of the cases, very simple network topologies and protocols, which were known to be universally stable, i.e. stable under any allowed configuration of the system, in the AQT model, turn out to be unstable when considering some of the newly proposed adversarial models. Thus showing that in general stability is not a robust property in the presence of failures.

1 Introduction

As the world becomes more dependent on communications and computer networks, there is a growing concern about the effects of network failures. Failures can occur as the result of natural disasters, as the result of human action or by unintentional errors in software or control systems. However, failures do not only happen in relation to catastrophic or accidental situations. In a more natural way, disruptions also appear in wireless mobile networks, where some connections between nodes may fail or change quickly and unpredictably due to mobility, network components' configuration or topology variations. In fact, our work is inspired by the growing importance of

*Work partially supported by the FET pro-actives Integrated Project 15964 (AEOLUS), and by the Spanish CICYT projects TIN2005-09198-C02-02 (ASCE) and TIN2005-25859-E. A preliminary version of part of this work appeared at the proceedings of the IEEE 10th International Conference on Parallel and Distributed Systems, 153–160, 2004.

this type of networks and the necessity of theoretical frameworks for modelling their traffic flow and disruptions.

We must be realistic and accept that our communication and computer networks nowadays are so widely extended, complex and heterogeneous, and they do carry so much traffic, that suffering from overloading and failures is unavoidable. To address these and other issues, we need first to understand the types of failures that a system can suffer and the immediate consequences of those failures. Thus, appropriate models to study real network systems which suffer from failures are needed. Those models could help on detecting, understanding, and overcoming the conditions leading to these mentioned negative effects, as well as helping on their further prevention. All these goals are strongly concerned with network survivability, which is the ability of the network to maintain or restore an acceptable level of performance during network attacks or failures, and to mitigate or prevent service outages.

In this work we consider systems in which nodes and links may fail, and we study the impact of the organization and accessibility of the intermediate packet-storage devices at the network (i.e., the intermediate queues at the hosts) on the stability of the system. We use adversarial models in which the traffic and the failures are assumed to be produced in an unpredictable (and rather malevolous) way. The theoretical study of stability via adversarial models has been a hot research topic in the last decade. *Stability* refers to the fact that the number of packets in the system remains bounded as the system dynamically evolves in time. This bound, which can be a function of the system parameters, is not dependent on time. In this context, stability and universal stability (a stronger notion of this term) are key features to be preserved. Universally stability either of networks or protocols has been shown a robust property as except for few protocols it is preserved under many variations of the AQT model []. We are interested in knowing whether this robustness is maintained in the presence of faults.

Stability in the Adversarial Queueing Theory model

Stability is studied in relation to the three main components $(\mathcal{N}, \mathcal{A}, \mathcal{P})$ forming a communication system: the network $\mathcal{N}$, the traffic pattern defined by the adversary $\mathcal{A}$, and the scheduling protocol $\mathcal{P}$. *Networks* are modeled by directed graphs in which nodes represent the hosts and edges represent the links between those hosts. The *protocol* (or queueing policy) determines the order in which the packets requiring to cross a link are scheduled to be forwarded. The *adversary* controls the traffic pattern.

The Adversarial Queueing Theory (AQT) model proposed by Borodin et al. [11] has become in recent times an important model to study stability issues, since it can describe the behavior of both connectionless and short-term connection networks, as well as connection-oriented networks. The AQT model considers the time evolution of a packet-routing network as a game between the adversary and the queue policy of the system. The system is considered to be *synchronous*.

At each time step the adversary may inject a set of packets to some of the nodes. Since the AQT is a non-faulty model, the adversary is only responsible for the traffic injection but it is not allowed to produce any failure in the topology or to control any other factor of the dynamics of the system. For each injected packet, the adversary specifies the route that it must traverse (*static routing*), after which the packet will disappear from the system. If more than one packet wishes to cross an edge e at the same time step, then the queueing policy chooses one of those packets to send across e. The remaining packets must wait in the queue. This game then advances to the next time step. The goal of the adversary is to try to prevent the protocol from guaranteeing load and delay bounds. On the contrary, the main goal of the model is to study conditions for stability of the network under different protocols.

In order not to trivially overload the system and in order to be able to guarantee delay bounds, it is necessary to restrict the traffic arriving to the network. The constraints on the traffic pattern must ensure that, over long periods of time, the maximum traffic injected in a link is roughly the amount of traffic that the link can forward. Two parameters (r, b) constraint an adversary in the AQT model, where $b \geq 0$ is *the burstiness* and $0 < r < 1$ is the *injection rate*. An adversary like this is henceforth referred to as a (r, b)-adversary. Let $N_e(I)$ be the number of packets injected by the adversary in a time interval I that have paths requiring a particular edge e. Then, a (r, b)-adversary in the AQT model must obey the following (leaky-bucket) constraint that restricts its injection power,

$$N_e(I) \leq \lceil r|I| \rceil + b. \tag{1}$$

An important term related to stability is that of *universal stability*, which is an extension of the stability property to systems with any configuration. It can be studied from the point of view of the network or from the point of view of the protocol. In broad strokes, a network is said to be universally stable if the number of packets in the system is bounded whatever adversary is considered and whatever protocol is used for scheduling packets at the edges. Analogously, a protocol is said to be universally stable if the number of packets in the system is bounded whatever adversary is considered and whatever network topology is implemented underneath.

A considerable amount of results concerning stability in the AQT model is available nowadays (see, e.g., [11, 4, 16, 20, 3, 9, 19, 21, 22, 23], just to mention a few). Also inspired by the AQT model, other more general adversarial models that try to capture additional network features have been proposed lately (see, e.g., [8, 7, 17, 14, 18, 12, 2, 10]).

Greedy protocols

We consider a situation in which there is a queue associated to each link of the network. The role of such queues is to store those packets that require to cross the link but can not be immediately forwarded because the link is busy or temporally unavailable. As it is usual in the literature, we only consider *greedy protocols* which apply, in a work-conserving manner, their queueing policies according to some local or global criteria. In general, a protocol is greedy if whenever there is at least one packet waiting in the queue, the queue serves a packet. The main queueing protocols considered in the study of stability are: FIFO, LIFO, SIS, LIS, NTG, FTG, NFS and FFS. In this work, we will also consider protocol LIQ.

The protocol LIFO (*last-in-first-out*) gives priority to the packet which joined the queue the latest; in FIFO (*first-in-first-out*), highest priority is given to the packet that has arrived first in the queue. The protocol which gives priority to the packet last introduced into the system is SIS (*shortest-in-system*), while in LIS (*longest-in-system*) every queue gives priority to the packet that has been in the system the longest time. The protocol NTG (*nearest-to-go*) assigns highest priority to the packet that is closest to its destination and FTG (*farthest-to-go*) assigns it to the packet that is farthest away from its destination. NFS (*nearest-from-source*) and FFS (*farthest-from-source*) consider the same policies but taking the distance to the source node of the packets as reference point. LIQ (*longest-in-queue*) is a variation of LIS that gives priority to the packet that in total has been waiting in queues for the longest time. As usual we assume that any arising ties are broken by the adversary.

The universal stability of most of these protocols in the AQT model was already studied in the work of Andrews et al. [4], where FTG, NFS, SIS and LIS were shown to be universally stable, while FIFO, LIFO, NTG and FFS were shown not to be universally stable. Recently, LIQ has been shown to be universally stable in the NSCAQT model [13] (and thus also in the AQT model), which is a generalization of the AQT model in which the condition of system synchronism is relaxed.

Networks Rings and directed acyclic graphs (which includes lines, forks and crossings) are known to be universally stable in the AQT model [11, 4]. In fact, the property of universal stability of networks in the AQT model was fully characterized in [3]. That characterization was described in terms of the existence of certain forbidden subtopologies, i.e., forbidden subdigraphs, in the network. More precisely (see Theorem 8 in [3]), a digraph is known to be universally stable in the AQT model if, and only if, it does not contain as subgraphs any of the digraphs in $\mathcal{E}(U_1) \cup \mathcal{E}(U_2)$, where U_1 and U_2 are the digraphs depicted in Figure 1, and $\mathcal{E}(G)$ denotes the family of digraphs formed by G and all the digraphs obtained from G by successive arc or 2-cycle subdivisions. Also in [3], it was shown that the universal stability of a given digraph in the AQT model can be decided

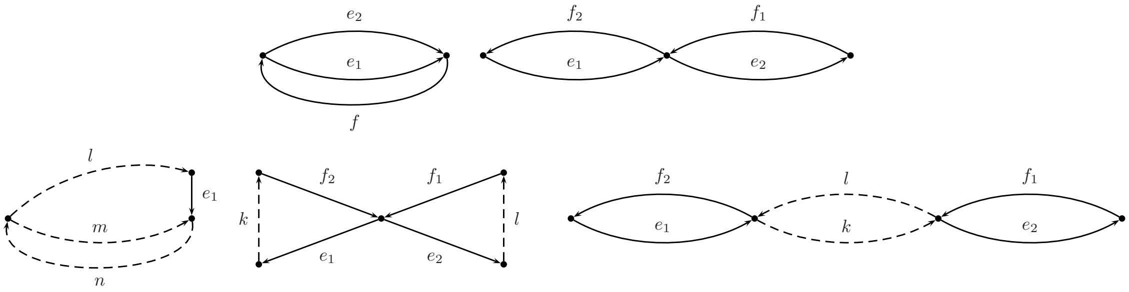


Figure 1: Forbidden subdigraphs for detecting universally stability; from left to right: U_1, U_2, shape of a digraph in $\mathcal{E}(U_1)$, and shape of two digraph in $\mathcal{E}(U_2)$, the second one correspond to a two cycle extension combined with an arc extension. The dashed arcs represent paths of the length specified by their labels, where $k, l \geq 0$ and $m, n \geq 1$.

in polynomial time.[1] The same property characterizes the set of networks stable under a protocol, for protocols, NTG [3] and FFS [1]. A different characterization for FIFO can be found in [23].

Other existing work

Other models can be found in the literature which could be interpreted as dealing with other variations on the capabilities of the edge for transmitting a packet, however none of them consider the possibility of no transmission. In [17, 12], packet routing networks can have capacities and speed/slowdowns. The model has been recently extended to take into account other features in [10]. Another interesting approach in the literature considers input/output blocking in the nodes [5, 6]. This latter model can produce unstable systems close to the ones that our edge-nR and edge-nR-B models can produce.

Two other proposals for faulty networks have been proposed in [8] and [7]. In both cases the injected packets are defined by specifying only source and destination, and thus are not forced to follow a pre-specified path. The adversary is restricted to guarantee that a static multi-commodity flow problem has a solution. Stability results are obtained using a load balancing algorithm, for the case that the adversary injection rate is one, and the packets have a unique common destination. The main difference in both models is that in [8] the adversary has to provide a solution to the associated multi-commodity flow problem, while in [7] the injection pattern must obey a condition that guarantees the existence of the solution. Our models and the models proposed in [8] and in [7] have the common characteristic that, for every interval I, the adversary can not inject to any edge e (or to any set S of nodes for the model in [7]), more packets than the number of packets that e can absorb (or the edges with only one extreme in S).

In [2], two additional AQT-based models were proposed to consider dynamic networks, namely the *failure model* and the *reliable model*. Both models consider only the possibility of link failures. For any edge e and any interval I, the adversary must obey the constraint

$$N_e(I) + \alpha F_e(I) \leq r|I| + b,$$

where $F_e(I)$ is the number of steps during a time interval I in which the edge e is down. The failure model considers $\alpha = 1$, while the reliable model considers any α such that $r < \alpha \leq 1$.[2] In both models, the adversary is not able to fail an edge permanently. As shown in [2], the failure and the reliable model turn out to be equivalent. Concerning the universal stability of networks, both models are also equivalent to the AQT model; thus rings and directed acyclic graphs (which includes lines, forks and crossings) are universally stable. Concerning the universal stability of

[1] In fact, the algorithm's running time is linear in the number of links (i.e., edges) in the network (i.e., graph).

[2] The case $\alpha = r$ is not considered since it implies that a link can be failed for an unbounded amount of time.

protocols, FTG, NFS, SIS, FIFO, LIFO, NTG and FFS are classified as in the AQT model, i.e., FTG, NFS, and SIS are universally stable but FIFO, LIFO, NTG and FFS are not. However, the LIS protocol makes a difference, since it is universally stable in the AQT model, but it is not neither in the failure nor in the reliable model. As we will see, from the point of view of universal stability of networks and protocols, the failure model is equivalent to the edge-R model that we present in this work.

It is worth to mention that none of these existing models for dealing with dynamic networks consider the possibility of having failures in the nodes of the topology.

Our contributions

In this work we deal with communication networks in which links, nodes, or both may fail. We propose adversarial models regulating the network behavior and restricting the traffic patterns occurring in this type of faulty systems. We study the dependence between the conditions for the stability of the system, and how the packets affected by failures are managed, under non-trivial underloaded worse-case scenarios. The adversary is responsible, not only for the traffic joining the network, but also for the failures of the links and nodes. Those failures have the effect of blocking temporally both those packets that arrive to a link/node while it is failed, and those waiting already to be transmitted through at the moment when the failure occurs.

In addition to proposing adversarial models for such faulty networking systems, we study the dependence between the conditions for the stability of the system, and how the packets affected by failures are managed. As we will see, depending on how the system is organized and prepared to deal with failures, the dynamics of the system change and thus its conditions for stability.

First, we study those systems in which the faulty elements are the links. We propose an adversarial model for them and study the conditions for stability under three different ways of managing the packets involved in the failures. This treatment gives raise to three versions of the proposed adversarial model for link-faulty networks, one for each management type, that we denote in the forthcoming as edge-R, edge-nR and edge-nR-B models. The difference relies on whether the queue of a failed link is able to receive (R) packets or not (nR) and, in case it is, on whether the affected packets are kept in the normal queues or in a special buffer (B) that the node keeps to that aim.

Then, we study those systems in which the faulty elements are the nodes. We propose an adversarial model for them and study the conditions for stability under four different ways of managing the packets involved in the failures. This treatment gives raise to four versions of the proposed adversarial model for node-faulty networks, that we denote in the forthcoming as node-RnT, node-nRnT-B, node-nRnT and node-nRT models. The main difference on these models relies on whether the queues of a failed node are able to receive (R) packets or not (nR) and, on whether a failed node that can not receive new packets while being failed, is able to transmit (T) those packets that are already queued, or it is not (nT). In some of these cases, there might be also an additional buffer (B) where the affected packets are kept.

Finally, we study systems in which both the nodes and the links may fail. The adversarial models we propose in this case are combinations of the previous models for links and nodes that inherit non-stability properties. We show that in the case that we combine edge and node models with some common positive stability property, then this stability property holds also in the combined model.

We show that for the failure models edge-R, node-RnT, and their combination the set of universally stable networks coincides with the set of universally stable networks on the AQT model, while some universally stable protocol in the AQT becames unstable. For the remaining models we show that very simple directed acyclic and topologies are unstable and that more universally stable protocols in the AQT becames unstable. The main drawback of this work is to show that there are differences on the requirements for stability, and that those differences depend directly on the way the packets affected by failures are managed. This provides us with interesting information on what is the more convenient way of dealing with failures depending on what is the topology of the network, and the protocols of the system.

Organization of the article

Section 2 describes the characteristics of the network scenario that this work assumes. The properties presented there are common to all the models that will be presented along this paper.

Section 3 formalizes some of the key concepts that are used in this work, specially those concerning the adversaries and stability. This section also describes how simulations can be done between systems in order to transfer knowledge about their stability.

In Section 4, we concentrate on studying the effect of link failures and propose three adversarial models for describing systems that suffer from that type of failures. Complementary, in Section 5 the focus is on studying the effect of node failures and propose four adversarial models. All the adversarial models proposed in both sections distinguish each other on how the system is organized for managing those packets that suffer from failures. Under all those models, we study the stability of simple network topologies as well as the stability of some well-known protocols.

This separation on the exposure between link and node failures, allows us to study their direct effect on the dynamics of the systems and on the conditions for stability in a quite simple and readable way. Moreover, it allows us to extract clear conclusions about what happens in systems that can suffer from link or node failures and highlight those combinations of link and node failure management that require further analysis. The reader will find those final results in Section 6. Finally, Section 7 posts some conclusions an review some open problems for future research.

2 Common network scenario

In this section, we detail the characteristics of the faulty network scenario that is assumed in this work. Hence, the following properties are common to all the models that will be introduced later:

Systems composed by network, adversary and protocol. The systems we deal with are composed by three main elements, $(\mathcal{N}, \mathcal{A}, \mathcal{P})$, where $\mathcal{N}$ is the network topology, $\mathcal{A}$ is an adversary defining the traffic and failure pattern on $\mathcal{N}$, and $\mathcal{P}$ is a scheduling protocol. *Networks* are modeled by directed graphs in which nodes represent the hosts and edges represent the links between those hosts. We assume that the graphs might have multiple arcs but no loops. The *adversary* controls the traffic pattern, where do packets join the system and which trajectory do they follow, and the failure pattern, which links and nodes fail and for how long. The *protocol* determines the order in which the packets requiring to cross a link are scheduled to be forwarded.

As we will see, systems are always considered in the context of an adversarial model, which will fix how the adversary is restricted in its power for injecting and making failures, and how the infrastructure of the network is organized to deal with the packets affected by those failures.

Adversarial traffic flow and adversarial connection disruption. As commented before, in the non-faulty AQT model the adversary was only responsible for the traffic injection and it was not allowed to control any other factor of the dynamics of the system. In this work we deal with faulty adversarial models in which the links and/or the nodes may fail temporarily. In such a dynamic scenario, the adversary is allowed to control, not only the traffic injection, but also those link and node failures. However, that control must be constrained in order not to trivially overload the network and not to trivially disconnect it because of an excess of failures. How this is constrained is one of the things to be determined by the model.

The different models we present in this work apply different restrictions on the adversary. In broad strokes, in any interval of time I, the adversary will be allowed to inject an amount of packets requiring a certain link e proportional to the steps in I in which the e is really available. The precise shape of the restrictions on the adversaries will be introduced later, when introducing each of the faulty models under consideration.

Queues at the head nodes of the links. We consider a situation in which there is a queue associated to each link of the network. The role of such queues is to store those packets that require to cross the link but can not immediately, either because the link is busy due to the transmission of a different packet, or because some transmission disruption in the context of that link happened. Those queues are physically kept in the node which is the head of the link

We consider that every packet has unit size and has to be received integrally before it is forwarded to the next outgoing link, i.e., networks do not have *cut-through* capabilities.

In some cases, with the aim of non-overloading the outgoing queues, we consider the existence of an additional buffer in the nodes. This consideration gives raise to the buffered models. In such models, that buffer is in charge of storing the blocked packets that suffered a failure. When the faulty element (link or node) is recovered from the failure, those packets affected by the failure are set to non-blocked. After that, the protocol chooses one packet among the non-blocked packets according to the scheduling policy of the system and puts the chosen packet, instantaneously, in the corresponding output queue (i.e., the queue corresponding to the next link in its path). Thus, only one packet can leave the buffer in one time step.

Synchronization. As it is usual in the studies of stability, we assume that the evolution of the packets in the network is synchronized. To maintain synchronization, each time step is divided into three basic phases: (1) *receive*, in which the packets (if any) arrive to the node and are placed in their corresponding outgoing queue; (2) *request connectivity* and *schedule*, in which the state of the node and every corresponding outgoing link is checked, and in which a packet to be forwarded is selected among all the available packets all the outgoing queues (if any) according to the protocol of the system; and (3) *send*, in which, for each outgoing queue with packets, the packet chosen by the protocol leave the queue.

This behavior varies slightly when some node failure occurs or when dealing with buffered nodes. How this behavior is changed in every case will be detailed further in the paper, in Sections 4 and 5, when presenting the different adversarial models under study.

Short-lived failures. When a link or node failure occurs, there is inevitably a period of disruption of the delivery of part of the traffic in the network. Although failures occurring in real networks are usually short-lived [15] (specially, the link failures), they can result in significant gaps and delays in received information. We focus on short-lived failures, and consider that no link or node can be failed for an arbitrary large amount of time, i.e., there exists a constant w, independent of time, bounding the number of consecutive steps in which any edge or node can be down. That means that, if any edge or node fails at any time step t then it will be recovered after at most $w + 1$ steps. This parameter will be part of the limits imposed to the sequence of failures that can be provoked by an adversary.

Evolution under failures. When a link failure is produced we assume, as it is natural, that the packets can not go through this link. Hence no packet kept in its queue can be served. But we also have to consider what to do with the incoming packets. Since they can not be lost, either they are received by its queue, or they are not. In the latter case, either the packets wait in the previous link of their path or they are kept in an additional buffer at the link's head node which is common for all the output links of such node.

When a node failure is produced we consider different managements depending on whether the queues of the output links of the failed nodes can transmit packets or not and in this latter situation we take into account whether there is an additional buffer where to keep all the packets that need to go through the node.

When failures can occur, greedy protocols refer to those ones advancing a packet whenever the link is able to transmit and there is at least one available packet waiting in the queue. In all the situations we assume an ideal world in which no packet already in the system is lost.

Homogeneity. We assume that all the links (respectively, all the nodes) will manage the packets and the queues affected by failures in the same way. The different options considered for such management will be presented further in the paper. The case of non-homogeneous management, i.e., when different links (respectively, different nodes) deal with failures in different ways, remains open.

3 Definitions

Having in mind the common networking scenario presented in previous Section 2, we need to specify clearly what is a faulty adversarial model, what is an adversary, and the definition of the different concepts of stability we will use. Let us start with the definition of a faulty adversarial model.

Definition 1 (Model $\mathcal{M}$) *A faulty adversarial model $\mathcal{M}$ specifies the following two characteristics of a system:*

- *whether the queues are or are not accessible during a failure,*
- *whether there are or there are not additional buffers, and*
- *what are the restrictions that apply on the adversary.*

Since all the models we introduce in this work are aimed for adversarial systems in which failures may occur, we skip often the adjective *faulty* and refer just to *model* $\mathcal{M}$.

In our systems, the adversary represents the unpredictable source of traffic of the system and additionally, in the case of faulty scenarios, it is also the source of failures for the network elements. Although unpredictable, the behavior of an adversary is expected to be as harmful as possible, inside some limits that avoid the trivial overflooding of the network. We need to provide a more formal definition for an adversary and its representation, which will be assumed for the rest of the paper.

Definition 2 (Adversary) *An adversary $\mathcal{A}$ for network $\mathcal{N}$ is an infinite sequence of triples. The t-th triple $< P_t, E_t, V_t >$ corresponds to the changes in traffic and failures at time t; it is composed of the following elements:*

- *the set of packets P_t injected in the system at time t,*
- *the set of links E_t that fail at time t, and*
- *the set of nodes V_t that fail at time t.*

In general we will omit the reference to the network in those cases where $\mathcal{N}$ is clear from the context.

In faulty models the adversary decides not only the packet injections but also the failures (in links, nodes or both), and thus this information must be part of its definition. Of course, since the AQT model is a non-faulty model, an adversary in AQT is described just by an infinite sequence of singletons; each singleton is formed by the set P_t of packets injected in the system at at time step t.

Before introducing the general restrictions that an adversary must obey we introduce some notation that will help on their description. Let $\mathcal{A}$ be an adversary for network $\mathcal{N}$ which at time step t has associated triple $< P_t, E_t, V_t >$; then, for any edge $e = (u, v)$ of $\mathcal{N}$, we denote as

$$N_e(t),\ \text{the number of packets } p \in P_t \text{ whose trajectory uses edge } e$$

$$F_e(t) = \begin{cases} 1 & e \in E_t \\ 0 & \text{otherwise} \end{cases}$$

$$H_e(t) = \begin{cases} 1 & u \in V_t \\ 0 & \text{otherwise} \end{cases}$$

$$D_e(t) = F_e(t) + H_e(t) - F_e(t)H_e(t).$$

Observe that $F_e(t)$ indicates whether the link e fails at time t, and $H_e(t)$ indicates whether the link e is unavailable because its head node fails at time t; thus, $D_e(t)$ gives evidence of whether the link e is available or not at time t, independently of which is the reason for it (the link failure or the head-node failure). We keep the notation $N_e(t)$ and $F_e(t)$ to be compatible with the usual notation in the AQT model and the *failure model* in [2]. The above functions can be extended as usual, by addition, to an interval of time $I = [t_0, t_1]$ composed of $|I| = t_1 - t_0 + 1$ time steps. For example $D_e(I) = \sum_{t \in I} D_e(t)$.

As we said, in order to avoid trivial overloading situations, the behavior of the adversaries must be restricted. Different restrictions on the power and shape of an adversary will give raise to different adversarial models for faulty networks. In a faulty adversary, these restrictions are based on three elements:

- on the amount of traffic that the adversary has permission to inject,
- on the amount and type of failures that the adversary is allowed to produce, and
- on the maximum duration of any of those failures.

To capture these issues on the definition of a faulty adversary, we introduce the concept of (r, b, w)-adversary, which is defined as follows:

Definition 3 ((r, b, w)-adversary in model $\mathcal{M}$) *An adversary $\mathcal{A}$ is an (r, b, w)-adversary in model $\mathcal{M}$ if the sequence of injections and failures of $\mathcal{A}$ satisfy the restrictions imposed by the model $\mathcal{M}$. Those restrictions are defined by the injection rate r, with $0 < r < 1$, the burstiness $0 \leq b$, and the maximum failure life-time $w \geq 0$. For any edge e and any time interval I, the adversary must respect the equations*

$$N_e(I) \leq \lceil r(|I| - D_e(I)) \rceil + b \quad \text{and} \quad (|I| = D_e(I)) \Rightarrow |I| \leq w. \tag{2}$$

Observe that, with the previous definition, an (r, b)-adversary in the AQT model is always an $(r, b, 0)$-adversary in any faulty model $\mathcal{M}$, just by skipping the possibility of producing failures.

Confronting such (r, b, w)-adversaries, we will study the stability of different systems with different characteristics. Stability refers to the fact that the number of packets in the system remains bounded as the system dynamically evolves in time. This bound, which can be a function of the system parameters, is not dependent on time. This desirable property is studied in relation to the three main components forming a system $(\mathcal{N}, \mathcal{A}, \mathcal{P})$: the network $\mathcal{N}$, the traffic pattern defined by the adversary $\mathcal{A}$ on $\mathcal{N}$, and the scheduling protocol $\mathcal{P}$.

Definition 4 (System in model $\mathcal{M}$) *We say that $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ is a* valid *system in model $\mathcal{M}$ if $\mathcal{A}$ is an (r, b, w)-adversary in model $\mathcal{M}$ for some r, $0 < r < 1$, and some $b, w \geq 0$.*

The evolution over time of a system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ valid in model $\mathcal{M}$ is fully determined by $\mathcal{M}$, i.e., by the accessibility of the queues, the existence of additional buffers, and the restrictions that $\mathcal{M}$ puts on $\mathcal{A}$. Thus, apart from the components of a system, the model in which the system is considered strongly determines the contents of the queues (and the buffers, if any) at any time step.

Now, let us state the main definitions of stability. These stability concepts will we used in the forthcoming of the paper. We start describing the most general notion of stability, i.e., the stability of a system in a certain model.

Definition 5 (System stability) *Let $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ be a valid system in model $\mathcal{M}$. S is stable in model $\mathcal{M}$ if the number of packets in the system (i.e., the number of packets in all the queues of the network) remains upper bounded by a time-independent constant $c(r, b, w, \mathcal{N})$ as the system dynamically evolves in time according to $\mathcal{M}$.*

We formally define now the property of network stability in a certain model. In this notion of stability, the network and a rate r (with $0 < r < 1$), are fixed.

Definition 6 (Network r-stability under a protocol $\mathcal{P}$) *A network $\mathcal{N}$ is* r-stable *under protocol $\mathcal{P}$ in model $\mathcal{M}$, if for any $b, w \geq 0$ and any (r, b, w)-adversary $\mathcal{A}$ for $\mathcal{N}$ in model $\mathcal{M}$, the system $(\mathcal{N}, \mathcal{A}, \mathcal{P})$ is stable in model $\mathcal{M}$.*

A stronger concept is that of *universal stability.* In broad strokes, a network $\mathcal{N}$ is said to be universally stable if the number of packets in the system is bounded whatever adversary $\mathcal{A}$ is considered and whatever protocol $\mathcal{P}$ is used for scheduling packets at the edges.

Definition 7 (Network universal stability) *A network $\mathcal{N}$ is universally stable in model $\mathcal{M}$ if it is r-stable in model $\mathcal{M}$ under any greedy protocol and for any $0 < r < 1$.*

As for networks, the concept of universal stability can be applied to protocols. In broad strokes, a protocol $\mathcal{P}$ is said to be universally stable if the system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ is stable for every $\mathcal{N}$ and $\mathcal{A}$. The following is a more formal definition of this concept:

Definition 8 (Protocol universal stability) *A protocol $\mathcal{P}$ is universally stable in model $\mathcal{M}$ if, for any network $\mathcal{N}$ and any $0 < r < 1$, $\mathcal{N}$ is r-stable under protocol $\mathcal{P}$ in model $\mathcal{M}$.*

As much as possible, it is always useful and interesting to infer new knowledge about the stability of some system, network or protocol from the already existing knowledge about some other systems, networks or protocols. In this section, we describe how systems can be simulated by other systems, which, as we will see, will be a useful tool for inferring some of the stability results that appear later in this work.

Definition 9 (System simulation) *Let $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ be a system in model $\mathcal{M}$, and let $S' = (\mathcal{N}, \mathcal{A}', \mathcal{P}')$ be a system in model $\mathcal{M}'$. The system S is* simulated *by the system S' if all the following conditions hold:*

- *when $\mathcal{A}$ injects at time t a packet p with route ρ in the system S, the adversary $\mathcal{A}'$ injects a corresponding packet p' at the same time t with the same route ρ in the system S',*
- *every time t that, according to $\mathcal{P}$, the packet p crosses the edge e of $\mathcal{N}$ in the system S, the packet p' crosses in S', according to $\mathcal{P}$', the same edge e of $\mathcal{N}$ at the same time t.*

It is possible to define a more generic definition of system simulation, however the previous one is enough for the system simulations used in the paper. Observe that when a system S can be simulated by another system S', the two systems have the same network and furthermore the number of packets present in the system S at time t is upperbounded by the number of packets present at time t in system S'. Therefore, we get

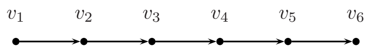
Lemma 1 *Let $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ be a system in model $\mathcal{M}$ that can be simulated by a system $S' = (\mathcal{N}, \mathcal{A}', \mathcal{P}')$ in model $\mathcal{M}'$. If the system S' is stable in model $\mathcal{M}'$, then system S is stable in model $\mathcal{M}$.*

Since any (r, b)-adversary in the AQT model is a $(r, b, 0)$-adversary in faulty adversarial model, the system behavior in a faulty adversarial model when no failure occurs will be the same as in the AQT model. Thus, for any of the faulty adversarial models $\mathcal{M}$ introduced in this paper, we can show trivially that

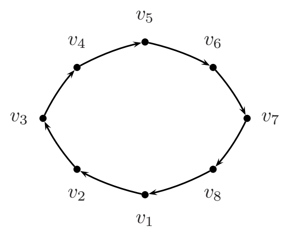
Fact 1 *Any system that is not stable in the* AQT *model, remains not stable in model $\mathcal{M}$.*

Finally we need some notation and definitions for directed graphs. Given a network $\mathcal{N}$ whose topology is described by the graph G,[3] we denote as $V(G)$ and $E(G)$ the set of nodes and edges

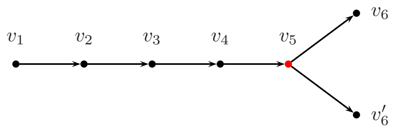
[3]When the context is clear enough and we want to focus on some property of the topology G of a network $\mathcal{N}$, we will do an abuse of terminology and refer to G as the network.



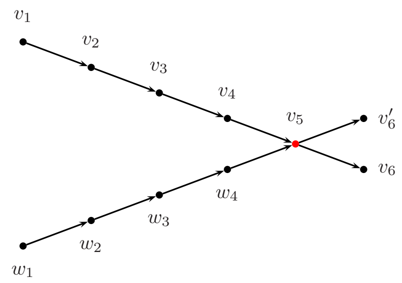
(a) A 6-line graph



(b) A 8-cycle or 8-ring graph



(c) A 5-fork graph; v_5 is its forking node



(d) A 5-crossing graph; v_5 is its crossing vertex

Figure 2: Examples of line, fork, crossing, and ring graphs.

of G, respectively. As usual, an edge $e \in E(G)$ in a directed graph G is represented by the two endpoint vertices that define it, i.e., $e = (u, v)$, where $u \in V(G)$, is said to be the *head* of the edge, and $v \in V(G)$ is said to be its *tail*. This representation implicitly defines also the orientation of the edge; thus, the edge $e = (u, v)$ is oriented from u to v, i.e., from its head node to its tail node. The output degree of a vertex $v \in V(G)$ is the number of edges whose head node is v; on the contrary, its input degree is the number of edges whose tail node is v.

With this widely-used graph notation in mind, we introduce in the following some of the particular network topologies that will be mentioned during the rest of this work. Basically, the only topologies we need to distinguish are cycles and some specific types of directed acyclic graphs.

Definition 10 (Simple dags) *Let G be a connected directed graph, and let $n > 1$.*

(i) G is a n-cycle (or n-ring) graph if it has n vertices and n edges organized in the following way: $G = (\{\{v_1, \ldots, v_n\}, \{(v_i, v_{i+1}) \mid 1 \leq i < n\} \cup \{(v_n, v_1)\}\})$.

(ii) G is a n-line graph if it is obtained from an n-cycle graph, by removing one of its edges, i.e., $G = (\{v_1, \ldots, v_n\}, \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\})$.

(iii) G is a n-fork graph if it is obtained from an n-line with two additional end vertices, in the following way $G = (\{v_1, \ldots, v_n, v_{n+1}, v'_{n+1}\}, \{(v_1, v_2), \ldots, (v_{n-1}, v_n), (v_n, v_{n+1}), (v_n, v'_{n+1})\})$.

(iv) G is a n-crossing graph if it is obtained joining a n-fork with a $(n-1)$-line in the following way: $G = (\{v_1, \ldots, v_n, v_{n+1}, v'_{n+1}, w_1, \ldots, w_{n-1}\}, \{(v_i, v_{i+1}) \mid 1 \leq i < n\} \cup \{(w_i, w_{i+1}) \mid 1 \leq i < n-2\} \cup \{(v_n, v_{n+1}), (v_n, v'_{n+1}), (w_{n-1}, v_n)\})$.

Figure 2 depicts some examples. In the forthcoming, we sometimes refer to line graphs, fork graphs, crossing graphs and rings to denote the whole family of n-line graphs, n-fork graphs, n-crossing graphs and n-rings, respectively, for every $n > 1$. Lines and cycles have the particularity that the input and output degree of their vertices is at most one. As we will see, this fact will provoke that the dynamics of systems with such topologies behave alike. Fork graphs and crossing graphs have only one vertex with out degree two; we will refer to this vertex as the *forking*, respectively *crossing vertex*.

4 Failures on links

In this section, we focus on the study of stability conditions in dynamic networks in which the edges may fail. We understand link failures in a natural way as the disruption on the transmission of packets. Therefore during a link failure packets are not sent. The constraints of an adversary in such faulty environments should take into consideration the number of steps during interval I that every edge is down. Recall that $F_e(I)$ is the number of steps in which the edge e is down during time interval I. For every edge e, all the models presented in this section consider that the number of packets and failures produced by an (r, b, w)-adversary during any time interval I which are concerned with link e must satisfy that

$$N_e(I) \leq \lceil r(|I| - D_e(I)) \rceil + b \quad \text{and} \quad (|I| = D_e(I)) \Rightarrow |I| \leq w. \tag{3}$$

Under these constraints on the flow injection and the failure production, we study the stability of networks when the management of those packets that suffer a failure is performed in different ways, depending on the accessibility of the queues. More precisely, we base our studies on which queue in the system keeps a packet that wants to traverse a failed link. We consider three situations:

(i) Although the link being failed, the queue at the head of the link can still receive packets, in such a case packets are kept at the queue associated to the failed link.
(ii) During a failure of a link the queue at its head is not accessible, in such a case packets have to wait in their previous location.
(iii) During a failure of a link the queue at its head is not accessible, but there is a buffer at each node that can store those packets that want to traverse a failed outgoing link.

We call the different models arising under such assumptions the edge-R model (i), the edge-nR model (ii), and the edge-nR-B model (iii), respectively. Of course, in any of these models, we assume that a failed edge can not transmit any packet, and thus no packets leaves the queue of a link during its failure.

4.1 The edge-R model

We consider the first model in which during a failure of a link e the packets wait to traverse e in its associated queue. Packets from incoming links and/or from the excess of the burstiness[4] might arrive to e since its queue is accessible. However, the arriving packets will wait in the queue until the edge recovers. When the link recovers, its queue always advances a packet if there is one. This behavior is depicted in Figure 3.[5] Observe that this coincides with the failure management under the failure model introduced in [2] and, in the case of adversaries that do not create any failure, it coincides also with the behavior in the AQT model.

We propose the edge-R model, in which the adversary is restricted by the expression in Equation 3, and the network has receiver links as the ones just described. We will show that the edge-R model can be simulated by the *failure model* introduced in [2], in which the injection rate strongly constraints both the maximum number of failures and the maximum number of packet injections per edge. The management of a failure is the same in both models, i.e., they both manage failures with links that can receive packets although being failed. However, the number of injections and failures that an (r, b)-adversary in the failure model can produce is restricted to

$$N_e(I) + F_e(I) \leq r|I| + b.$$

Observe that this expression already restricts the duration of failures by itself, without the necessity of fixing an explicit upper bound for it. Observe also that, given a concrete injection rate r, our edge-R model allows more powerful adversaries than the failure model, both in terms of failure production and in terms of packet injection. We can establish the following relation:

[4]Note that, when an edge e is failed during an interval I, then $N_e(I) \leq b$ packets can still be injected.

[5]With the aim of making the figure more clean and comprehensive, we depict the queue associated to each link right over the link (instead of at the head of the link, where it really is).

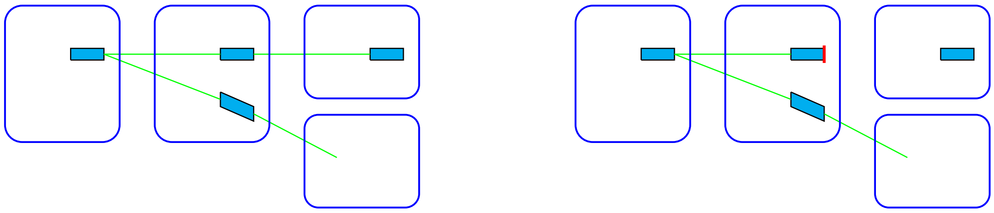


Figure 3: The edge-R model: Behavior when no failure occurs *(left)* and when a failure happens *(right)*.

Theorem 2 *Given a network $\mathcal{N}$, any (r, b)-adversary $\mathcal{A}$ for $\mathcal{N}$ in the failure model is an $(r, b, \lceil b/(1-r)\rceil)$-adversary in the* edge-R *model.*

Proof. The restriction for the edge-R model can be easily obtained from the failure model restriction, taking into account that $0 < r < 1$:

$$N_e(I) + rF_e(I) < N_e(I) + F_e(I) \leq r|I| + b.$$

The restriction $N_e(I) + F_e(I) \leq r|I| + b$ implies also that $F_e(I) \leq r|I| + b$. Therefore, in a time interval of length t in which an edge is failed, it must hold that $t \leq rt + b$, which implies that $t \leq b/(1 - r)$. ∎

Moreover, any adversary in the edge-R model is also an adversary in the failure model with an increased injection rate. Such an increase depends on w.

Theorem 3 *Given a network $\mathcal{N}$, any (r, b, w)-adversary $\mathcal{A}$ in the* edge-R *model is a $(\frac{r+w}{w+1}, b)$-adversary in the failure model.*

Proof. Recall that w bounds the number of consecutive steps in which any edge can be down. Since the duration of a failure in any edge e is bounded, the maximum number of failures that can occur to e in the edge-R model at any interval of time I is $\frac{w}{w+1}|I|$. Then, we can rewrite the constraint

$$N_e(I) \leq r(|I| - F_e(I)) + b$$

for an (r, b, w)-adversary $\mathcal{A}$ in the edge-R model as

$$N_e(I) + F_e(I) \leq r|I| - rF_e(I) + F_e(I) + b,$$

therefore,

$$\begin{aligned} N_e(I) + F_e(I) &\leq r|I| - rF_e(I) + F_e(I) + b \\ &= r|I| + (1 - r)F_e(I) + b \\ &\leq r|I| + ((1 - r)w/(w + 1))\,|I| + b \\ &= ((r + w)/(w + 1))\,|I| + b\,. \end{aligned}$$

This expression corresponds to the restriction that applies to a (r', b)-adversary in the failure model, in which $r' = (r + w)/(w + 1)$ and for which it holds that $0 < r' < 1$.

∎

Since the behavior of the queues and the systems in the failure model in [2] and in the edge-R model is the same, we have (from Theorems 2 and 3) that any valid system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the edge-R system is also a valid system in the failure model, and vice versa. Furthermore, at any time step, the number and position of any packet in S in the edge-R model is the same as in S in the failure model. Thus, the failure model and our edge-R model are equivalent from the point of view of universal stability of networks and protocols. Therefore, according to the results in [2], we can state the following corollaries:

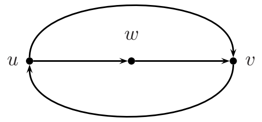


Figure 4: Network $\mathcal{N}$, with $V(\mathcal{N}) = \{u, v, w\}$ and $E(\mathcal{N}) = \{(u, w), (w, v), (u, v), (v, u)\}$

Corollary 4 *Directed acyclic graphs and rings are universally stable in the* edge-R *model.*

Corollary 5 *A network $\mathcal{N}$ is universally stable in the* edge-R *model if, and only if, $\mathcal{N}$ is universally stable in the* AQT *model.*

Corollary 6 SIS, FTG, *and* NFS *are universally stable in the* edge-R *model. However,* FIFO, LIFO, NTG, FFS, *and* LIS *are not universally stable in the* edge-R *model.*

In this last corollary, the protocol LIS is put in a different category with respect to the AQT adversarial model for non-faulty systems, where LIS was universally stable. This is due to the instability of LIS under the failure model, which was shown in [2]. The LIQ protocol remains unclassified, as this protocol is universally stable in the AQT model [13] and not known to be universally stable in the failure model. However, as the following lemma proves, we can show that LIQ is neither universally stable in the failure, nor in the node-RnT model.

Lemma 7 LIQ *is neither universally stable in the failure, nor in the* edge-R *model.*

Proof. Consider the network $\mathcal{N} = (V, E)$, with vertex set $V(\mathcal{N}) = \{u, v, w\}$ and edge set $E(\mathcal{N}) = \{(u, w), (w, v), (u, v), (v, u)\}$, depicted in Figure 4. Assume an initial configuration consisting of $s/2$ packets located at the node w that want to cross the path $\{wvu\}$ and a set of $s/2$ packets located at the node u that wants to cross the path $\{uvu\}$. The following adversary makes the network described above unstable[6] under LIQ.

Round 1: For the first s steps, we inject a set α of rs packets that want to traverse the path $\{vuv\}$. Due to the initial configuration and the protocol those packets will be stacked at node v at the end of the round.

Round 2: For the next rs steps we inject a set β of r^2s packets that follow the path $\{vuwv\}$ together with a set γ of r^2s packets that follow edge (u, v). Those packets compete with the set α. At the end of the round there will be r^2s packets waiting to traverse edge $\{vuwv\}$ and r^2s packets waiting to traverse $\{uv\}$.

Round 3: For the next r^2s steps we inject a set δ of r^3s packets that follow the path $\{uv\}$ together with a set η of r^3s packets that follow the path $\{uwvu\}$. Those packets compete with the surviving packets and at the end of the round there will be r^3s packets waiting to traverse $\{uv\}$ and r^3s packets waiting to traverse the path $\{uwvu\}$.

Round 4: For the next r^3s steps we make link (w, v) fail for r^4s steps and inject a set of r^4s packets that follow the path $\{uvu\}$. Those packets compete with the surviving packets of the previous round and the failures keeps some of the packets waiting to traverse link (w, v). So, at the end of the round there will be r^4s packets waiting to traverse the path $\{wvu\}$ and r^4s packets waiting to traverse the path $\{uvu\}$.

At the end of the fourth round, we are in the initial situation but with a set of $2r^4s$ packets in the system instead of s. For $r > 0.841$, we have that $2r^4s > s$ and thus but repeating the set

[6] All the proofs of instability in this work are based on induction. A set of rounds compose a step of the induction reasoning. The goal is to demonstrate that the number of packets in the system can increase from step to step (and, by applying the inductive hypothesis, they can increase infinitely). The configuration of the system at the end of every step must be the same as at the beginning (in terms of the type and the location of the packets). For the sake of simplicity, we only reproduce the inductive step and sometimes we omit some additive constants in our analysis, however, those omissions will not change the final result.

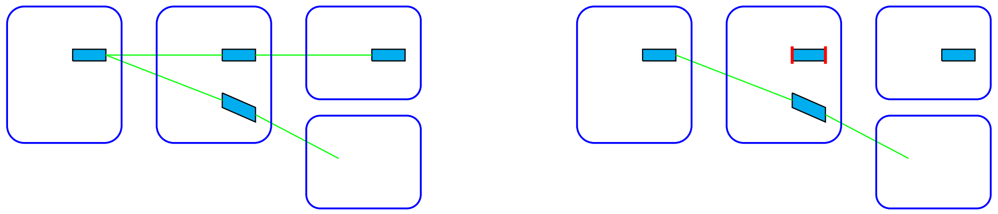


Figure 5: The edge-nR model: Behavior when no failure occurs *(left)* and when a failure happens *(right)*.

of rounds the number of packets in the system grows unboundedly, thus making the system not stable under LIQ.

Observe that the proposed adversary obeys the restrictions of the failure model, and thus also those of the edge-R model for an adequate w. ■

Observe also that the adversary constructed in the previous lemma also makes the system unstable under LIS.

4.2 The **edge-nR** model

Let us take now into consideration the case in which during a link failure the associated queue is not accessible. This fact forces that no packet can be injected requiring a failed edge. Moreover packets directed to an edge $e = (v, w)$ which come from an incoming link $e' = (u, v)$ will not be able to join the queue of e since we consider it is not accessible. They will stay in the queue of e'. In order to avoid loosing packets, the packets in an active queue are classified as blocked or not. A packet is *blocked* if the link connecting to its next destination is down. The queuing policy, applied to such a queue type, will select one packet among those that are not blocked. Blocked packets remain in the queue. This behavior is depicted in Figure 5.

We propose the edge-nR model, in which the adversary is restricted by the expression in Equation 3, and the network has non-receiver links as the ones just described. In this model, we address the stability of protocols for different topologies of networks. As we will see, in contrast to what occurs both in the AQT model and in the edge-R model, there are very simple topologies that turn out not to be universally stable in the edge-nR model. Note that this result shows a significant difference between the edge-nR model and the other models, specially in contrast to the adversarial model AQT.

In the following we show that line graphs are not stable in the edge-nR model under SIS, LIFO, NTG, NFS, and FFS.

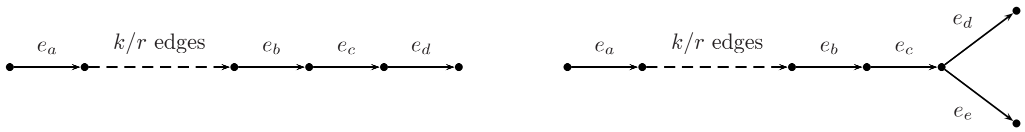
Theorem 8 *For every injection rate $0 < r < 1$, there is a $n_0 > 1$ such that any n-line graph with $n \geq n_0$ is not r-stable in the edge-nR model under protocols* SIS, LIFO, NTG, NFS, *and* FFS.

Proof. Consider a $(5 + k/r)$-line graph with edges labelled as in Figure 6(a), where k is selected to guarantee that $rk > 1$. We assume that initially the network is empty and that the protocol is SIS. The adversary operates in the following rounds.

Round 1: For the first k/r steps, we inject a set α of k packets that want to traverse the line starting at e_a and ending at e_d. Note that due to the length of the line all the injected packets will stay in the system at the end of the round, distributed along the line.

Round 2: Edge e_d fails for as many steps as to guarantee that all the α packets are waiting at e_c. At the end of this round there are k packets queued at e_c.

Round 3: For the next k/r steps, e_d fails and we inject a set β of k packets that want to traverse the line starting at e_a and ending at e_c.



(a) $(5 + k/r)$-line graph

(b) $(4 + k/r)$-fork graph

Figure 6: Line graphs and fork graphs used in Theorems 8 and 9

Round 4: Edge e_d and e_c fail for as many steps as to guarantee that all the β packets are waiting at e_b. At the end of this round there are k packets at the queue of e_c and k packets at the queue of e_b, all of them requiring e_c.

Round 5: For the next k steps, we inject a set γ of rk packets that want to traverse the line graph starting at e_a and ending at e_d. Since the scheduling policy is SIS the β packets queued at e_b leave and, at the end of this round, there will be the k packets of the α set still at e_c . Furthermore, there are rk packets traversing the line with final destination e_d.

Round 6: Edge e_d fails for as many steps as to guarantee that all the set γ is waiting at e_c. At the end of this round there are $k + rk$ packets queued at e_c, all with destination e_d.

The adversary continues repeating rounds 3 to 7, thus creating at every completed phase an additional increase at edge e_c of $rk > 1$ packets, with respect to the quantity of packets queued at this edge at the beginning of every phase. The above construction shows instability of a line graph for SIS, but can be also applied for LIFO and NTG, NFS and FFS (with an adequate tie breaking). Furthermore, observe that in the above description we have that $w \leq n$.

■

Furthermore, for the FTG protocol there are very simple directed acyclic graphs which are not stable. In the following we show that networks with topologies describing some forking can be made unstable.

Theorem 9 *For every injection rate $0 < r < 1$ there is a $n_0 > 1$ such that any n-fork graph with $n \geq n_0$ is not r-stable in the edge-nR model under* FTG*.*

Proof. Consider a $(4 + k/r)$-fork graph with edges labelled as in Figure 6(b), where k is selected to guarantee that $rk > 1$. The adversary operates in phases. During each phase $\phi \geq 0$ (rounds $1 - 4$) we will accumulate rk packets at the queue of e_c. Let us suppose that at the beginning of phase ϕ there are $k + \phi rk$ packets queued at edge e_c with destination e_d[7]. The four rounds composing every phase $\phi \geq 0$ are described in the following. It is easy to see by induction on the number of phases ϕ that the number of packets in the system grow up unboundedly.

Round 1: For k/r steps, k packets with destination e_e are injected in edge e_a. During all these steps edge e_d fails, thus accumulating all the packets still queued at edge e_c.

Round 2: Edges e_d and e_e fail enough time to keep the packets with destination to them at the queue of e_c. At the queue there are $k + \phi rk$ packets with destination e_d and k more packets with destination e_e.

Round 3: For the next k steps, the adversary injects a set β of rk packets at edge e_a with destination e_d. Since neither edge e_d nor edge e_e fail a tie arises. FTG solves the tie in such a way that the k packets at the queue of e_c with destination e_e will flow to e_e.

[7]To start from an empty initial configuration just consider two initial preceding rounds: (1^{st} round) For k/r steps, inject k packets at e_a with destination e_d and, (2^{nd} round) Edge e_d fails enough to guarantee that all the packets get blocked at the queue of e_c.

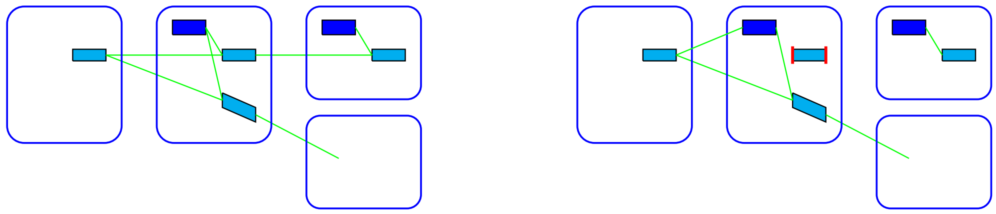


Figure 7: The edge-nR-B model: Behavior when no failure occurs *(left)* and when it does *(right)*.

Round 4: Edge e_d fails enough time to guarantee that the β packets get queued at edge e_c. At the end of this round there are $k + (\phi + 1)rk$ packets at the queue of e_c with destination e_d.

The adversary continues repeating the same sequence of rounds, thus creating at every completed phase an additional increase of rk packets. Furthermore, observe that in the above description we have $w \leq n$.

■

From the previous Theorems 8 and 9, it follows Corollary 10:

Corollary 10 *Directed acyclic graphs and rings are* not *universally stable in the edge-nR model. More precisely, there are directed acyclic graphs which are not stable under protocols* SIS*,* LIFO*,* NTG*,* NFS*,* FFS *and* FTG*.*

All these protocols can bring the system to instability because they allow to reorder the accumulated packets in such a way that old queued packets are kept on as blocked, while newer ones have priority over them. On the contrary, the intrinsic ordered nature of the protocols FIFO and LIS makes it difficult to accumulate packets. Although we suspect that acyclic topologies and even rings are stable under FIFO, LIS, and LIQ, these questions are left as open in this work. Nevertheless, we can show that LIS and LIQ are not universally stable protocols by using a simple cyclic topology. Using the graph given in Figure 4, and the adversary given in the proof of Lemma 7, we can conclude that

Lemma 11 LIS *and* LIQ *are not universally stable in the edge-nR model.*

4.3 The edge-nR-B model

Finally, we consider a mixed situation in which the queue of a failed link is not accessible, but in which the packets that can not join it are kept in an additional buffer placed at the head node of the link. This buffer is shared by all the links that have the same head node, thus acting as an intermediate buffer for packets that have to continue to the queue of a link that was failed when they arrived. We assume that the extra buffer at every node is always accessible and always active. Moreover, the state of the packets in those extra buffers is also kept (blocked and non-blocked packets). Thus if a packet p traverses the link $e' = (u, v)$, and next p has to traverse $e = (v, w)$ but e goes down, then p is kept in the extra buffer at v. It will be considered as a blocked packet while e is failed. The node queue policy will select one packet among the non-blocked ones, i.e., among those addressed to currently non-failed edges. This behavior is depicted in Figure 7.

We propose the edge-nR-B model, in which the adversary is restricted by the expression in Equation 3, and the network has non-receiver links with extra buffers as the ones just described.

In this model, we study a specific subset of networks: those whose vertices have all output degree at most one. This includes line graphs and ring graphs. In this case, the buffer in a node v of the graph will (only) store the packets that can not be served towards the queue of the (unique) outgoing edge (v, w). It seems that, in this case, the edge-nR-B model has some similarities with the edge-R model. The main idea behind this is that each queue in the edge-R model can store the packets of the queue of the corresponding edges as well as the packets in the extra buffer of its source node.

Theorem 12 *Let $\mathcal{N}$ be a network in which all the nodes have output degree at most one. Any system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the* edge-nR-B *model can be simulated by a system $S' = (\mathcal{N}, \mathcal{A}, \mathcal{P}')$ in the* edge-R *model.*

Proof. When no failures occur, the protocol $\mathcal{P}' = \mathcal{P}$ makes the predicate to hold trivially. When failures occur in the system S, the packets requiring to traverse any failed edge $e = (u, v) \in E(\mathcal{N})$ are kept in the extra buffer at node u. We simulate this system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ by $S' = (\mathcal{N}, \mathcal{A}, \mathcal{P}')$ in the edge-R model, the greedy protocol $\mathcal{P}'$ will be defined during the proof.

For system S, let us denote as $Q_u(t)$ and $Q_e(t)$ the contents at time t of the extra buffer at node $u \in V(\mathcal{N})$ and the queue corresponding to edge $e \in E(\mathcal{N})$, respectively. By $Q'_e(t)$ we refer to the queue at the same time t and the same edge e, but in the system S'. Protocol $\mathcal{P}'$ is defined in such a way that whenever link e is not failed and $Q'_e(t) = Q_u(t) \cup Q_e(t)$, the packet that traverses e, is the same selected by $\mathcal{P}$. We describe for each time step t the behavior of S' simulating S, and we show by induction on t that each edge $e = (u, v)$ satisfies that:

– $Q'_e(t) = Q_e(t) \cup Q_u(t)$, and
– if p crosses in S the link e of $\mathcal{N}$ at time t, the corresponding packet crosses in S' the same link e at the same time t.

Let us consider the behavior of edge e at time step $t + 1$ in both systems in the following three cases:

1. e **is alive.** The queue associated to e may receive a set of packets L_e scheduled by the queues of the incoming edges at time step t, a set of packets I_e injected by the adversary at this time step, and a set of packets P_u coming from the buffer, notice that P_u is either empty, when $Q_u(t) = \emptyset$, or contains only one packet, otherwise. After receiving these packets, it applies the protocol $\mathcal{P}$ and selects a packet p_e^t in order to be scheduled. Hence, $Q_e(t+1) = (Q_e(t) \cup L_e \cup I_e \cup P_u) \setminus \{p_e^t\}$ and $Q_u(t + 1) = Q_u(t) \setminus P_u$.
In the system S', the queue of edge e receives the same packets as in S: L_e by induction hypothesis and I_e because the adversaries are identical. Now, since by induction hypothesis $Q'_e(t) = Q_e(t) \cup Q_u(t)$ and e receives the same new packets, protocol $\mathcal{P}'$ can select the same packet p_e^t for being scheduled. Then we have that $Q'_e(t + 1) = (Q'_e(t) \cup L_e \cup I_e) \setminus \{p_e^t\} = (Q_e(t) \cup Q_u(t) \cup L_e \cup I_e) \setminus \{p_e^t\} = Q_e(t + 1) \cup Q_u(t + 1)$.

3. e **is failed.** In this case the queue of edge e in system S can not receive any packet. If a packet $p_e^t \in L_e$ is scheduled to e at time step t, then it is stored in the extra buffer at node u at time step $t+1$. If the adversary injects a set of packets I_e at time step $t+1$ they are also stored in the extra buffer at u. Then we have that $Q_e(t + 1) = Q_e(t)$ and, as u has output degree 1, $Q_u(t + 1) = Q_u(t) \cup L_e \cup I_e$.
In the system S', the queue of e can not schedule any packet but it receives the same packets than the queue of u in S, packet p_i injected by the adversary and packet $p_e^t \in L_e$ scheduled at time step t. Then we have that $Q'_e(t + 1) = Q'_e(t) \cup L_e \cup I_e = Q_e(t) \cup Q_u(t) \cup L_e \cup I_e = Q_e(t + 1) \cup Q_u(t + 1)$.

Observe that, the contents of the queue corresponding to edge $e = (u, v)$ in the system S' is the same as the sum of contents of the extra buffer at the node u plus the queue for the edge e in the system S'. The quantity of packets does not vary, but their location. This applies for every edge e and time.

■

As the following theorem proves, this simulation is in fact two sided.

Theorem 13 *Let $\mathcal{N}$ be a network in which all the nodes have output degree at most one. Any system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the* edge-R *model can be simulated by a system $S' = (\mathcal{N}, \mathcal{A}, \mathcal{P}')$ in the* edge-nR-B *model.*

Proof. When no failures occur, the protocol $\mathcal{P}' = \mathcal{P}$ make the predicate to hold trivially. When failures occur in the system S, the packets requiring to traverse any failed edge $e = (u, v) \in E(\mathcal{N})$ are kept in the queue of edge e. We simulate this system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ by $S' = (\mathcal{N}, \mathcal{A}, \mathcal{P}')$ in the edge-R model, protocol $\mathcal{P}'$ is defines in such a way that the same packets requiring to traverse e would be kept either in the queue of e or in the extra buffer at u. As the output degree of u is one, all the packets on those queues must traverse e, we define $\mathcal{P}'$ in such a way that at any time step the packet selected to traverse e is the same selected by $\mathcal{P}$. Taking this into account, the rest of the proof is similar to the one of Theorem 12. ∎

Combining these theorems with Corollary 4, we can state the following result.

Corollary 14 *Any graph in which all the nodes have output degree at most one (this includes line graphs and rings) is universally stable in the* edge-nR-B *model.*

However, a significant difference appears when considering nodes with output degree greater than one, e.g., when considering forks. In the following we show that for every injection rate, we can find a fork which is not stable under several of the usual greedy protocols, namely LIFO, SIS, NTG, FTG, NFS, and FFS.

Theorem 15 *For every injection rate $0 < r < 1$, there exists an $n_0 > 0$ such that the n-fork graph, for $n \geq n_0$, is not r-stable in the* edge-nR-B *model under any of the following protocols:* LIFO, SIS, NTG, FTG, NFS, *and* FFS.

Proof. Given a valid injection rate r, let us consider a $(4 + k/r)$-fork like the one depicted in Figure 6(b), whose length depends on r. Let us suppose, without loss of generality, that the initial configuration is empty. An adversary playing the following six rounds indefinitely can make the network not stable for any of the above-mentioned greedy protocols.

Round 1: For the first k/r steps, the adversary injects a set α of k packets of the form $(e_a, \ldots, e_d)$.

Round 2: The edge e_d fails enough to make all the set α accumulate in the queue of node v_3.

Round 3: For k/r steps, the edge e_d continues failing and a set β of k packets are injected, which wish to traverse the path $(e_a, \ldots, e_e)$.

Round 4: The edges e_d and e_e fail enough to accumulate the packets from the α and the β sets in the queue of node v_3. At the end of this round there are $2k$ packets in the queue of this node.

Round 5: We consider now an interval of time of k steps. This represents that, except in the case that both edges e_d and e_e fail, k packets will leave the queue at edge v_3. The adversary injects a set γ of rk packets composed by rk packets of the form $(e_a, \ldots, e_d)$. The packets which abandon the queue at v_3 are the packets from the β set. An additional round has to be considered.

Round 6: The edge e_d fails enough to keep blocked the packets from α and to block the packets from γ, at the queue of the vertex v_3.

At the end of the sixth round, there are $k + rk$ packets in the queue of v_3 with destination e_d. By indefinitely repeating the rounds from 3 to 6, the system accumulates rk additional packets at every iteration. The number of packets in the system grows unboundedly, thus making the system unstable under the mentioned protocols. For the protocols NTG, FTG, NFS and FFS, the forms of the packets are valid to bring the system to instability, solving ties appropriately. Furthermore, observe that in the above description we have $w \leq n$.

∎

In contrast to the edge-nR model, even that the queues of the failed links are not accessible, the packets requiring those links are kept in a different queue, i.e., the extra buffer. However, this fact does not imply significant changes on the networks which are stable. Already in networks whose nodes have output degree one, situations similar to those in the edge-nR model are obtained.

We can also show that the protocols LIS and LIQ are not universally stable in the edge-nR-B model. For doing so, consider the graph given in Figure 4 and the adversary given in the proof of Lemma 7.[8] Therefore, we have that

Lemma 16 LIS *and* LIQ *are not universally stable in the edge-nR-B model.*

5 Failures on nodes

In this section, we consider dynamic networks in which the nodes may fail. All the models presented in this section impose the restriction that, for every edge e and any time interval I , the number of packets injected by the adversary that use e, together with the number of node failures produced by the adversary at the head of e, must satisfy that

$$N_e(I) \leq \lceil r(|I| - H_e(I)) \rceil + b \quad \text{and} \quad (|I| = D_e(I)) \Rightarrow |I| \leq w. \tag{4}$$

Recall that $H_e(I)$ is the number of steps in which the head of edge e fails during time interval I. Observe that this restriction is obtained from the general restriction for faulty adversarial models described in Equation 2, when no failures at the links are taken into account.

Under these constraints on the flow injection and the node failure production, we study the stability of networks when the management of those nodes that suffer a failure is performed in different ways. More precisely, we take into consideration the possibility that the nature of the failure affects transmission and/or reception of packets. This feature leads us to consider three types of node failures:

(type 1) during a node failure nodes are unable to transmit and to receive packets.

(type 2) during a node failure nodes are unable to transmit but able to receive packets.

(type 3) during a node failure nodes are able to transmit but unable to receive packets.

The type 1 of failure is the strongest one, under this type of failure packets do not arrive and leave, therefore it is natural to consider a queueing system in which all the queues of the outgoing links of a node are not accessible during a failure. Observe that in such a case packets whose next link to traverse has a failed head node must wait

(i) in the queue of the previous link.

In the type 2 of failure packets are not leaving from the node, but can arrive. Thus, in the associated queuing systems the packets arriving to the node can be kept

(ii) in the queue associated to the edge, or
(iii) in an extra buffer at the node. Here we assume that even though the node can receive its output link queues are not accessible.

The type 3 of failures leave us with a failed node which is unable to receive but it can still transmit. Then those packets that have to arrive to the failed node can not follow their next link and are kept

(iv) in the queue of the previous link, while packets at a failed node can be transmitted provided that the next node is not failed.

[8] Observe that now, during the failure of edge (w, v), packets have to wait in the queue of link (u, w) but, even in that case, the system accumulates packets.

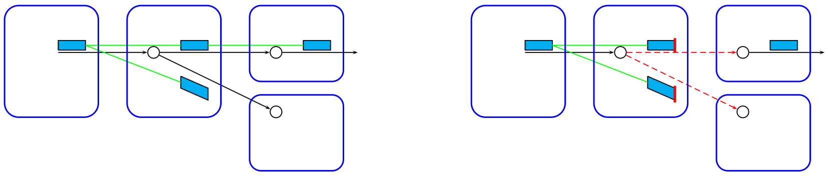


Figure 8: The node-RnT model: Behavior with no failures (left), and when a failure occurs (right)

Each of these options will be considered in the adversarial models we present in this section. Option (ii) will characterize the node-RnT model, while option (iii) will define the node-nRnT-B model. Option (i) is used in the node-nRnT model and, finally, option (iv) will be considered in the node-nRT model.

5.1 The **node-RnT** model

We consider the first type of node failure management in which a failure of a node v means that the packets wait to traverse v in their corresponding output queue. Packets from incoming links and/or from the excess of the burstiness might arrive to v since its queue is accessible. However, the arriving packets will wait in the corresponding output queue until the node recovers. When the node recovers, every outgoing queue advances a packet if there is one. This behavior is depicted in Figure 8.[9]

We consider now the node-RnT model, in which the adversary is restricted by the expression in Equation 4, and the network has receiver non-transmitter nodes as the ones just described.

Using the fact that a node failure represents somehow the failure of all its outgoing edges, we will show that the node-RnT model can be simulated by the edge-R model. For doing so we have to take into account that the management of a failure is the same in both models, i.e., they both manage failures with receiver links. However, the adversaries in the edge-R model are more powerful as they can make fail one outgoing edge independently of the other outgoing edges.

Theorem 17 *Any system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the node-RnT model can be simulated by a system $S' = (\mathcal{N}, \mathcal{A}', \mathcal{P})$ in the edge-R model.*

Proof. The new adversary $\mathcal{A}'$ is obtained from $\mathcal{A}$ by keeping the same sequence of packets injections and replacing any failure of a node $u \in V(\mathcal{N})$ by the failure of all the outgoing edges of u. From the definition of the adversaries it follows that, for every link e and any time interval I, the amount $H_e(I)$ incurred by adversary $\mathcal{A}$ in system S coincides with the value $F_e(I)$ incurred by adversary $\mathcal{A}'$ in S'. Therefore, from the definition of the models it is straightforward to show that when $\mathcal{A}$ is an (r, b, w)-adversary in model node-RnT then $\mathcal{A}'$ is also an (r, b, w)-adversary in model edge-R, and that the two systems act alike. ■

With this result, any (positive) network or protocol stability result from the edge-R model can be transferred to the node-RnT model. Thus, from Corollary 4 we obtain the following result

Corollary 18 *Directed acyclic graphs and rings are universally stable in the node-RnT model.*

Corollary 5 establishes that the set of networks that are universally stable in the edge-R model coincide with the set of networks that are universally stable in the AQT model. Therefore, taking into account Fact 1, we obtain

[9]With the aim of making the figure more clean and comprehensive, nodes are depicted now as bigger blocks, with a dispatcher switch that puts every incoming packet in the corresponding output queue. The queue for each outgoing link is depicted now inside the node, as it really is.

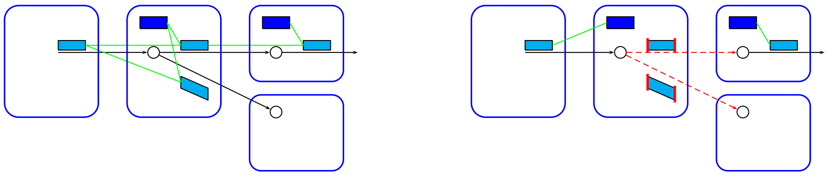


Figure 9: The node-nRnT-B model: Behavior with no failures (left), and when a failure occurs (right)

Corollary 19 *A network $\mathcal{N}$ is universally stable in the node-RnT model if, and only if, $\mathcal{N}$ is universally stable in the* AQT *model.*

Knowing this, with the results from [3] we can state that

Corollary 20 *The universal stability of a given network in the node-RnT model can be decided in polynomial time.*

For protocols, we have a similar scenario. Taking into account Fact 1, any instability result for protocols in the AQT model extends to the node-RnT model and, taking into account Theorem 17, any universal stability result in the edge-R model extends to the node-RnT model. We then get

Corollary 21 SIS, FTG, *and* NFS *are universally stable in the node-RnT model, while* FIFO, LIFO, NTG, *and* FFS *are not universally stable in the node-RnT model.*

Only LIS and LIQ remain unclassified as those protocols are universally stable in the AQT model but not universally stable in the edge-R model. The following lemma shows that they are not universally stable in the node-RnT model.

Lemma 22 LIS *and* LIQ *are not universally stable in the node-RnT model.*

The proof uses the graph given in Figure 4 and the adversary given in the proof of Lemma 7, but replaces every failure of edge (w, u) just by the failure of the node w.

5.2 The node-nRnT-B model

We assume now that, in addition to the queues associated to its outgoing edges, every node has an extra buffer. In the failure management form we present here, the failure of a node v means that the queues of the outgoing edges are not accessible during a failure and that packets arriving at v wait to traverse v in the extra buffer. This buffer is then shared by all the packets that at some time step could not be forwarded from v. As before, we assume that the extra buffers at the nodes are always accessible and always active. The only difference with respect to the status of packets in such a queue is that now either all the packets are blocked or non-blocked, because the failure of a node implies somehow the failure of all the outgoing edges. In those periods in which the node is not failed, the node queue policy will select one packet among the packets queued in it to be forwarded. This behavior is depicted in Figure 9.

We propose the node-RnT model, in which the adversary is restricted by the expression in Equation 4, and the network has non-receiver non-transmitter nodes as the ones just described.

In this model, we focus again on a specific subset of networks: those whose vertices have all output degree at most one. This includes line graphs and ring graphs. In this case, the failure of a node, both in terms of injections and buffer occupancy, have the same impact as a failure of the outgoing edge in the edge-nR-B model. As a consequence of this fact, we have the following result:

Theorem 23 *Let $\mathcal{P}$ be a greedy protocol and $\mathcal{N}$ a network in which all the nodes have output degree at most one. Any system $\mathcal{S} = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the node-nRnT-B model can be simulated by a system $\mathcal{S}' = (\mathcal{N}, \mathcal{A}', \mathcal{P}')$ in the edge-R model.*

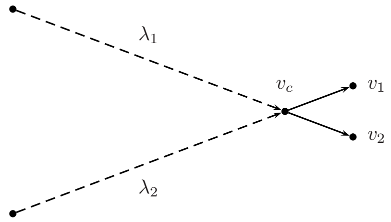


Figure 10: n-crossing graph used in Theorem 25

Proof. The new adversary $\mathcal{A}'$ is obtained from $\mathcal{A}$ by keeping the same sequence of packets injections and replacing any failure of a node u by the failure of the unique outgoing edge of u. From the definition of the adversaries, it follows that for every link e and any time interval I, the amount $H_e(I)$ incurred by adversary $\mathcal{A}$ in system S coincides with the value $F_e(I)$ incurred by adversary $\mathcal{A}'$ in S'. Therefore, from the definition of the models it is straightforward to show that when $\mathcal{A}$ is an (r, b, w)-adversary in model node-nRnT-B then $\mathcal{A}'$ is also an (r, b, w)-adversary in model edge-R.

We consider the protocol $\mathcal{P}'$ defined in such a way that, from any node u, the packet that is transmitted through the unique outgoing edge $e = (u, v) \in E(\mathcal{N})$ is selected by $\mathcal{P}$ when considering the packets present in the queue Q_e associated to e and the extra buffer Q_u. The sequence of injections and the definition of $\mathcal{P}'$ guarantees that at any time step the set of packets stored in the queue Q'_e associated to $e' \in E(\mathcal{N})$ coincides with $Q_e \cup Q_u$, and thus the simulation follows. ∎

As a consequence of the previous theorem we have that, those networks which are universally stable in the edge-R model and whose vertices have output degree at most one, remain universally stable in the node-nRnT-B model.

Corollary 24 *Line graphs and rings are universally stable in the node-nRnT-B model.*

However, we can show that in the node-nRnT-B model there are directed acyclic graphs other than line graphs that are not stable under any greedy protocol.

Theorem 25 *For every injection rate $0.5 < r < 1$, there is a $n_0 > 1$ such that any n-crossing graph with $n \geq n_0$ is not r-stable in the node-nRnT-B model under any greedy protocol.*

Proof. Let $n > 2$ and consider the n-crossing graph depicted in Figure 10, where we denote as λ_1 and λ_2 the two n-line graphs with endpoints in the crossing vertex v_c, and as v_1 and v_2 the tail vertices of the two edges outgoing from the crossing vertex.

Let us assume that initially, there are $2n$ packets stored in the extra buffer of the crossing vertex v_c. An adversary playing the following rounds indefinitely will make the network unstable, independently of the greedy protocol used:

Round 1: For the first n steps, the adversary injects rn packets at the first node of λ_1 with destination v_1, and also rn packets at the first node of λ_2 with destination v_2. Those injections are done simultaneously at regular intervals of length $1/r$ starting at time 1. Since $|\lambda_1| = |\lambda_2| = n$, none of the injected packets have arrived to the crossing vertex v_c at the end of this round.

Round 2: The crossing vertex v_c fails for n steps, thus at the end of the round all the packets injected in the previous round will be stored in its extra buffer.

Round 3: For the next $1/r$ steps no packet is injected and no node is failed.

At the end of the third round, there will be $2rn + n - 1/r$ packets stored in the extra buffer of the crossing vertex v_c. Observe that the last round provides time to guarantee that in the interval between the last failure and the first injection the adversary restriction is fulfilled.

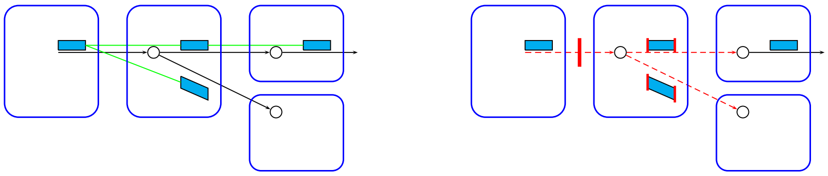


Figure 11: The node-nRnT model: Behavior with no failures (left), and when a failure occurs (right)

By infinite repetitions of the above pattern of injections and failures, this adversary would make the system unstable for any r such that $2rn + n - 1/r \geq 2n + 1$, in particular this holds for any r such that

$$r \geq \frac{1}{4} \cdot \frac{n + 1 + \sqrt{n^2 + 10n + 1}}{n}.$$

Observe that this result is independent of the protocol used in the system. Observe also that 0.5 is the lower bound for the instability of the n-crossing graphs, since

$$\lim_{n \to \infty} \left(\frac{1}{4} \cdot \frac{n + 1 + \sqrt{n^2 + 10n + 1}}{n} \right) = \frac{1}{2}.$$

■

Corollary 26 *There are directed acyclic graphs that are not universally stable in the node-nRnT-B model. In fact, there are directed acyclic graphs that are not stable in the node-nRnT-B model under any greedy protocol.*

5.3 The node-nRnT model

In the failure management form we present here, the failure of a node v means two things: that the queues in it do not receive (i.e., are not accessible) and that they can not transmit neither. Thus, the packets wanting to cross an incoming edge (u, v) must wait at their output queue in u until v recovers. This behavior is depicted in Figure 11.

We propose the node-nRnT model, in which the adversary is restricted by the expression in Equation 4, and the network has non-receiver non-transmitter nodes as the ones just described.

Although it is easy to show that on networks whose vertices have all output degree at most one, this includes line and ring graphs, the failure of a node has the same impact as the failure of the outgoing edge in the edge-nR model and derive non-stability results from this fact we can provide strongest non stability results.

Theorem 27 *Any network with at least one link can be made unstable in the node-nRnT model.*

Proof. We show that for any r, $0 < r \leq 1$, the 2-line graph is not r-stable under any greedy protocol. Consider a 2-line graph, and let us denote as (v_1, v_2) its unique edge. Let us assume that initially, there are q packets stored in the queue at v_1 whose destination is the next adjacent vertex v_2. An adversary playing the following round indefinitely can make the network unstable for any of the usual greedy protocols:

Round 1: For q time steps, the adversary injects rq packets at v_1, whose destination is v_2. During that time, the vertex v_2 is failed. This failure makes the packets stored at v_1 to remain there.

Round 2: The vertex v_2 recovers for one time step.

At the end of the second round, there will be $rq + q - 1$ packets stored in queue at v_1 with destination v_2. By infinite repetitions of this pattern of injections and failures, this adversary

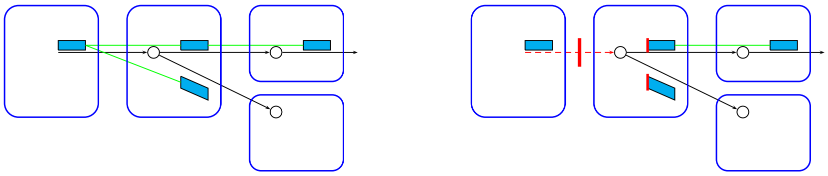


Figure 12: The node-nRT model: Behavior with no failures (left), and when a failure occurs (right)

would make the system unstable for any r such that $rq > 1$, as $r > 0$, we have that this happens for any $q > 1/r$. Thus for any required r, a sufficiently big enough initial load q can be found, such that this inequality holds. Observe that, implicitly, $w = q$ and that this result is independent of the protocol used in the system and also independent of the length of the line.

■

This bad behavior is achieved because the restrictions imposed by the model on the adversaries do not take into account the failures at the tail of the link. As a consequence of the previous theorem we have also that

Corollary 28 *There are no universally stable greedy protocols in the node-nRnT model.*

5.4 The node-nRT model

In the failure management form we present here, the failure of a node v means that the queues of the outgoing edges are not accessible during a failure and that packets wanting to cross (u, v) must wait at u until v recovers. However, even when v is failed, packets waiting to traverse any link $e = (v, w)$ outgoing from v can proceed provided that the forwarding protocol selects them and that their destination node w is not failed. This behavior is depicted in Figure 12.

We propose the node-nRT model, in which the adversary is restricted by the expression in Equation 4, and the network has non-receiver transmitter nodes as the ones just described.

In this model, we are also able to show that any network with at least one link is not stable under any greedy protocol (the proof is the same as for Theorem 27).

Theorem 29 *Any network with at least one link can be made unstable in the node-nRT model.*

As a consequence of the previous theorem we have also that

Corollary 30 *There are no universally stable greedy protocols in the node-nRT model.*

6 Failures on both, links and nodes

Until now, we have considered separately the case in which the communication network can only suffer link failures from the case in which the network can only suffer node failures. In this section we tackle the general case in which both, link and node failures, might occur in the system.

In this general case, the restrictions on an (r, b, w)-adversary in an adversarial model $\mathcal{M}$ are described by the Equation 2. Recall that Equation 2 constrained the injection power and the failing power of an (r, b, w)-adversary in the following way:

$$N_e(I) \leq \lceil r(|I| - D_e(I)) \rceil + b \quad \text{and} \quad \left(|I| = \sum_{t \in I} D_e(t)\right) \Rightarrow |I| \leq w.$$

where $D_e(I)$ counted the number of time steps in I that the edge e is not available, either because e is failed or because the head of e is down (but without counting double those time steps in which both the edge e and the head of e are down).

We consider combined faulty adversarial models in which the links implement one of the models presented in Section 4 (namely the edge-R, edge-nR or edge-nR-B model), and the nodes follow one of the models presented in Section 5 (namely the node-RnT, node-nRnT-B, node-nRnT or node-nRT model). Since both the model implemented at the links and the model implemented at the nodes apply over the same set of queues, we must carefully specify how their access to them is organized. When a node failure occurs (independently of whether some of the outgoing edges also fails), the system uses the adversarial model implemented by the node. If, on the contrary, a node does not fail but some of its outgoing edges does, then the system uses the adversarial model implemented by the links to deal with that failure.

When, in a combination of models, only the model for links (respectively, for nodes) is buffered, then the extra buffer at the nodes is used only to store the packets suffering the failures of the links (respectively, of the nodes). In the combined (edge-nR-B,node-nRnT-B) model, in which both the model for links and the model for nodes are buffered, we consider that the extra buffer at each node is also unique also, and thus the buffer is shared by both models.

Table 1 summarizes the results obtained concerning the universal stability of rings and directed acyclic graphs, and the universal stability of the protocols SIS, NFS, FTG, LIS and LIQ for the models presented in Section 4. Table 2 summarizes the results obtained concerning the universal stability of the same networks and protocols for the models presented in Section 5.

Table 1: Universal stability property in the combination of adversarial models in which the nodes do not fail, i.e., they implement the AQT model, but the links do. Symbol √ means that the network (respectively, the protocol) is universally stable, while symbol × means it is not. Symbol other-USP represent the other AQT universal stable protocols mentioned in this paper, i.e. SIS, NFS, and FTG.

MODELS	edge-R		edge-nR-B		edge-nR	
non-faulty node	ring	√	ring	√	ring	×
	dags	√	dags	×	dags	×
	LIS, LIQ	×	LIS, LIQ	×	LIS, LIQ	×
	other-USP	√	other-USP	×	other-USP	×

Table 2: Maintainance of the universal stability property in faulty adversarial models. Symbols denote the same as in Table 1. The results for the combination of non-faulty nodes and non-faulty edges are omitted, since they are the same as in Table 1.

MODELS	node-RnT		node-nRnT-B		node-nRnT node-nRT	
non-faulty edge	ring	√	ring	√	ring	×
	dags	√	dags	×	dags	×
	LIS, LIQ	×	LIS, LIQ	×	LIS, LIQ	×
	other-USP	√	other-USP	×	other-USP	×

It is clear that a system with a combination of models, in which at least one of them can produce some type of instability, can automatically suffer from the same instability. Thus, those results on instability that we obtained in Sections 4 and 5 (see Tables 1 and 2) apply directly to the combined adversarial models incorporating any of the models in which the negative result arises. In the following we study the universal stability of networks and protocols in combined adversarial models, considering only those combinations of models that guarantee universal stability when considered separately.

When considering the universal stability of rings, the combined models still to be studied are those combining either edge-R or edge-nR in their links, and node-RnT or node-nRnT-B in their nodes. When considering the universal stability of directed acyclic graphs, only the combined model (edge-R,node-RnT) needs still to be studied. We study all these combined models in the following. First, let us point out the fact that systems in the (edge-R,node-RnT) model can be simulated by systems in the edge-R model. As we will see, this will have further implications.

Theorem 31 *Any system $S = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the (edge-R,node-RnT) model can be simulated by a system $S' = (\mathcal{N}, \mathcal{A}', \mathcal{P})$ in the edge-R model.*

Proof. Let $\mathcal{A}$ be an (r, b, w)-adversary for $\mathcal{N}$ in the (edge-R,node-RnT) model. Assume that $\mathcal{A}$ at time t has associated triple $< P_t, E_t, V_t >$ following the same idea used in Theorem 17, we replace node failures by outgoing links failure. Thus, $\mathcal{A}'$ at time t has associated tuple $< P_t, E'_t >$ where

$$E'_t = E_t \cup (\cup_{u \in V_t} \{e \in E(\mathcal{N}) \mid e = (u, v)\}) .$$

As the behaviour of the queues in the (edge-R,node-RnT) model is the same as in the edge-R model, the result follows.

■

With this result, any (positive) network or protocol stability result from the edge-R model can be transferred to the (edge-R,node-RnT) model. Thus, from Corollary 4 we obtain the following result:

Corollary 32 *Directed acyclic graphs and rings are universally stable in the (edge-R,node-RnT) model.*

Moreover, recall that Corollary 5 stated that universal stability of networks in the edge-R model is equivalent to universal stability of networks in the AQT model. Taking also into account that the set of networks that are not universally stable in the AQT model are neither stable in the edge-R, we can conclude the equivalence also between the (edge-R,node-RnT) model and the AQT model.

Corollary 33 *A network $\mathcal{N}$ is universally stable in the (edge-R,node-RnT) model if, and only if, $\mathcal{N}$ is universally stable in the AQT model.*

Knowing this, with the results from [3] we can state that

Corollary 34 *The universal stability of a given network in the (edge-R,node-RnT) model can be decided in polynomial time.*

From Theorem 23 we know that any system $\mathcal{S} = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the node-RnT or node-nRnT-B model, in which all the nodes of $\mathcal{N}$ have output degree at most one, can be simulated by a system $\mathcal{S}' = (\mathcal{N}, \mathcal{A}', \mathcal{P}')$ in the edge-R model. Therefore, for the particular case of lines and rings, we have the following result when taking into account Corollary 4

Theorem 35 *Lines and rings are universally stable in the (edge-R, node-RnT) model and in the (edge-R, node-nRnT-B) model.*

From Theorem 12 we know that any system $\mathcal{S} = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ in the edge-nR-B model, in which all the nodes of $\mathcal{N}$ have output degree at most one, can be simulated by a system $\mathcal{S}' = (\mathcal{N}, \mathcal{A}', \mathcal{P}')$ in the edge-R model. This simulation can be extended to combined models in which the adversarial model of the nodes do not change. Therefore, taking into account the previous theorem, we have the following result for the particular case of lines and rings:

Theorem 36 *Lines and rings are universally stable in the (edge-nR-B, node-RnT) model and in the (edge-nR-B, node-nRnT-B) model.*

As a consequence of the instability results obtained in Sections 4 and 5, the instability of some of the protocols considered in this work can be directly stated in some combined models. Thus, only the universal stability of SIS, NFS and FTG in the (edge-R,node-RnT) combined adversarial model remains to be studied. As we have just shown in Theorem 31, systems in the (edge-R,node-RnT) model can be simulated by systems in the edge-R model, and this allows us to transfer any positive network or protocol stability result from the edge-R model to the combined (edge-R,node-RnT) model. Thus, from Corollary 6 we obtain the following result:

Corollary 37 SIS, FTG, *and* NFS *are universally stable in the (edge-R,node-RnT) model, while* FIFO, LIFO, LIS, LIQ, NTG, *and* FFS *are not universally stable in the (edge-R,node-RnT) model.*

7 Conclusions and open problems

In this paper, we have studied the behavior of adversarial systems under node and link failures. We have focused on analyzing the stability of packet-switched networks and greedy protocols. Stability refers to the fact that the quantity of packets in the network stays always bounded independently of time, which has direct implications on the delay that the packets might experience. The model adopted to study the network system is the Adversarial Queueing Theory (AQT) model, which has been shown to be appropriate for exploring the behavior of both connectionless networks and networks with short-term connections.

We have proposed extensions of the adversarial model for packet-switched networks in which (non-perpetual) link and/or edge failures might occur. We have proposed three models depending on how the link-failure affects the accessibility to link queues and another set of four models depending on the type of node-failure and the behavior of queues during a failure. Table 3 summarizes the most relevant results of our work and points out the differences among all the models introduced in this paper together with the AQT model (which is reported in the area corresponding to the combination of non-faulty nodes and non-faulty edges). It is worth to mention the fact that only in one of the combinations stability is robust, in the remaining cases even very simple directed acyclic network topologies, e.g., lines, forks, or crossings, can be made unstable when failures occur.

It is specially important (and the main contribution of this paper) to realize that the effect of those failures on the stability of the system depends strongly on the type of failures and where the interrupted packets are stored. It is clear from our results that the combination of the edge-R model with the node-RnT model, in which the queues are always accessible, would be preferable since it is robust (in terms of stability) when related to the AQT model: This management form assures the same stability conditions for faulty communication networks as does the AQT model for non-faulty ones. This is a specially interesting and desirable property, since it is transferring the conditions for stability from a non-faulty environment to a faulty one. When the access to the queues corresponding to failed edges can not be assured, then a management in the way considered in the buffered models is preferable since it assures at least universal stability of simple topologies as lines and rings.

Many open problems arise from our work since, to the best of our knowledge, this is the first time that node failures are explicitly studied in the context of stability in faulty networks. This work represents also the first attempt to study the effect of the management of failures under different strategies.

Observe that we have got very strong instability results as soon as the queue kept at a node are not accessible during a failure. This is due to the fact that we can make the tail of a node fail and use this fact to overflow the network while preserving the adversary restrictions. A better understanding of the behavior of such system under other adversary restrictions is an interesting open problem.

Concerning the proposed models, it would be specially interesting to provide a characterization for universal stability of networks in the models in which some of the AQT universally stable networks are not universally stable. In particular to see if the property can also be expressed in terms of forbidden subdigraphs as it was shown in [3] for the AQT adversarial model for non-faulty networks. Even if this is not the case, can the universally stability of networks be decided in polynomial time?

Taking into consideration that the models proposed in this work are homogeneous, it would be of interest to study the stability of heterogeneous systems, e.g., systems whose nodes and links do not share the same failure management forms. For an heterogeneous faulty environment, it seems natural to consider that the adversary could also be restricted by the constraint proposed in Equation 2 (but, of course, other constraints might also be appropriate). An important matter

Table 3: Maintainance of the universal stability property in faulty adversarial models. Symbol $\surd$ means that the network (respectively, the protocol) is universally stable, while symbol $\times$ means it is not. Symbol other-USP represent the other AQT universal stable protocols mentioned in this paper, i.e. SIS, NFS, and FTG. Observe that the area corresponding to the combination of non-faulty nodes and non-faulty edges correspond to the AQT model.

MODELS		non-faulty edge	edge-R	edge-nR-B	edge-nR
non-faulty node	ring	$\surd$	$\surd$	$\surd$	$\times$
	dags	$\surd$	$\surd$	$\times$	$\times$
	LIS, LIQ	$\surd$	$\times$	$\times$	$\times$
	other-USP	$\surd$	$\surd$	$\times$	$\times$
node-RnT	ring	$\surd$	$\surd$	$\surd$	$\times$
	dags	$\surd$	$\surd$	$\times$	$\times$
	LIS, LIQ	$\times$	$\times$	$\times$	$\times$
	other-USP	$\surd$	$\surd$	$\times$	$\times$
node-nRnT-B	ring	$\surd$	$\surd$	$\surd$	$\times$
	dags	$\times$	$\times$	$\times$	$\times$
	LIS, LIQ	$\times$	$\times$	$\times$	$\times$
	other-USP	$\times$	$\times$	$\times$	$\times$
node-nRnT node-nRT	ring	$\times$	$\times$	$\times$	$\times$
	dags	$\times$	$\times$	$\times$	$\times$
	LIS, LIQ	$\times$	$\times$	$\times$	$\times$
	other-USP	$\times$	$\times$	$\times$	$\times$

in the study of faulty heterogeneous systems would be the study of the influence of certain types of failure management and the importance of their incidence and location in the system. This would give us some knowledge about system configurations with certain stability guarantees or, from the opposite point of view, knowledge about which system configurations are not convenient because of their potentiality to provoke instability.

Of course the door remains open for the proposal of other models for systems with adversarial traffic and adversarial failure production with different forms of failure management. It seems reasonable, for example, to think about adversarial models that drop those packets that suffer long delays due to failures along their traversal, and require a posterior packet retransmission. Thinking differently, one might also think about requiring additional properties on the trajectories of the packets, in order to minimize (or to overcome) the effect of a failure (e.g., to require that, during a given period of time, the packets in the system are guaranteed to describe edge/vertex disjoint trajectories). Also related to this latter option of requiring extra properties on the paths, one could consider an alternative approach, which is to compute backup or repair routes that allow the failure to be repaired locally by re-routing the affected packets to their backup routes.

References

[1] C. Àlvarez, M. Blesa, J. Díaz, A. Fernández, and M. Serna. The complexity of deciding stability under FFS in the adversarial model. *Information Processing Letters*, 90(5):261–266, 2004.

[2] C. Àlvarez, M. Blesa, J. Díaz, A. Fernández, and M. Serna. Adversarial models for priority-based networks. *Networks*, 45(1):23–35, 2005.

[3] C. Àlvarez, M. Blesa, and M. Serna. A characterization of universal stability in the adversarial queueing model. *SIAM Journal on Computing*, 34(1):41–66, 2004.

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

[5] M. Andrews and L. Zhang. Stability results for networks with input and output blocking. In *30th. Annual ACM Symposium on Theory of Computing* (STOC'98), pages 369–377. ACM Press New York, 1998.

[6] M. Andrews and L. Zhang. Achieving stability in networks of input-queued switches. *IEEE/ACM Transactions on Networking*, 11(5):848–857, 2003.

[7] E. Anshelevich, D. Kempe, and J. Kleinberg. Stability of load balancing algorithms in dynamic adversarial systems. In *34th. Annual ACM Symposium on Theory of Computing* (STOC'02), pages 399–406. ACM Press New York, 2002.

[8] B. Awerbuch, P. Berenbrink, A. Brinkmann, and C. Scheideler. Simple routing strategies for adversarial systems. In *42th. IEEE Symposium on Foundations of Computer Science* (FOCS'01), pages 158–167, Las Vegas, USA, 2001. IEEE Computer Society Press.

[9] R. Bhattacharjee, A. Goel, and Z. Lotker. Instability of FIFO at arbitrarily low rates in the adversarial queueing model. *SIAM Journal on Computing*, 34(2):318–332, 2004.

[10] M. Blesa, D. Calzada, A. Fernández, L. López, A. Martínez, M. Santos, A. Serna, and C. Thraves. Adversarial queueing model for continuous network dynamics. *Theory of Computing Systems*, 2007. To appear.

[11] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson. Adversarial queueing theory. *Journal of the ACM*, 48(1):13–38, 2001.

[12] A. Borodin, R. Ostrovsky, and Y. Rabani. Stability preserving transformations: Packet routing networks with edge capacities and speeds. *Journal of Interconnection Networks*, 5(1):1–12, 2004.

[13] J. Céspedes, A. Fernández, J. López-Presa, M. 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. Technical Report RoSaC-2006-6, Grupo de Sistemas y Comunicaciones, Universidad Rey Juan Carlos, 2006.

[14] J. Echagüe, V. Cholvi, and A. Fernández. Universal stability results for low rate adversaries in packet switched networks. *IEEE Communication Letters*, 7(12):578–580, 2003.

[15] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *ACM SIGCOMM Internet Measurement Workshop* (IMW'02), pages 237–242. ACM Press New York, 2002.

[16] D. Koukopoulos, M. Mavronicolas, S. Nikoletseas, and P. Spirakis. On the stability of compositions of universally stable, greedy contention-resolution protocols. In *16th International Symposium on Distributed Computing* (DISC'02), volume 2508 of *Lecture Notes in Computer Science*, pages 88–102. Springer-Verlag GmbH, 2002.

[17] D. Koukopoulos, M. Mavronicolas, and P. Spirakis. Instability of networks with quasi-static link capacities. In *10th Internaltional Colloquium on Structural Information Complexity* (SIROCCO'03), volume 17 of *Proceedings in Informatics*, pages 179–194. Carleton Scientific, 2003.

[18] D. Koukopoulos, M. Mavronicolas, and P. Spirakis. Performance and stability bounds for dynamic networks. In *7th International Conference on Parallel Architectures, Algorithms and Networks* (I-SPAN'04), pages 239–246. IEEE Computer Society Press, 2004.

[19] Z. Lotker, B. Patt-Shamir, and A. Rosén. New stability results for adversarial queuing. *SIAM Journal on Computing*, 33(2):286–303, 2004.

[20] A. Rosén. A note on models for non-probabilistic analysis of packet switching networks. *Information Processing Letters*, 84(5):237–240, 2002.

[21] A. Rosén and M. Tsirkin. On delivery times in packet networks under adversarial traffic. In *16th ACM Symposium on Parallel Algorithms and Architectures* (SPAA'04), pages 1–10. ACM Press New York, 2004.

[22] M. Weinard. The necessity of timekeeping in adversarial queueing. In *4th International Workshop on Efficient and Experimental Algorithms* (WEA'05), volume 3503 of *Lecture Notes in Computer Science*, pages 440–451. Springer-Verlag GmbH, 2005.

[23] M. Weinard. Deciding the FIFO stability of networks in polynomial time. In *6th Italian Conference on Algorithms and Complexity* (CIAC'06), volume 3998 of *Lecture Notes in Computer Science*, pages 81–92. Springer-Verlag GmbH, 2006.