The Robustness of Stability under Link and Node Failures *

Carme Álvarez Maria Blesa Maria Serna

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

November 30, 2009

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 incorporate 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 many different ways depending on whether they can receive or serve packets, or they rather can not. In most of the cases, protocols and networks containing very simple topologies, which were known to be universally stable in the AQT model, turn out to be unstable under some of the newly proposed adversarial models. Thus showing that universal stability of networks 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 very 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 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 malicious) 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 as number of nodes or edges or the diameter, 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 of networks has been shown a robust property as it is preserved under many variations of the AQQT model [3, 12, 2, 10]. And, except for the LIS protocol, the universally stable protocols remain stable under those variations of the AQQT model [2, 10]. We are interested in studying the influence of queue management during failures on the robustness of stability in faulty scenarios.

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 (AQQT) model proposed by Borodin et al. [11, 4] 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 AQQT 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. 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 AQQT 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, 14, 18, 3, 9, 17, 19, 20, 21], just to mention a few).

**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 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 LIPO (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, LIPO, 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 in
Related work

Initially, all the studies about the stability of networks and protocols focused on static systems. Later, different particularities related to dynamic networks started to be introduced. Although failures were not initially considered as such, some adversarial models were proposed that deal with variations on the capabilities of the edge for transmitting a packet.

In [12], links in packet routing networks can have capacities or speed/slowdowns (not both) that can change dynamically. However, since only finite slowdowns and positive capacities are considered in [12], the effect of failures is not really simulated, but roughly approximated. With such model, the set of networks that are universally stable when considering dynamic capacities or slowdowns coincides with the networks that are universally stable in the AQT model. From the point of view of protocols, LIS is shown not to be universally stable, in contrast to its behavior under AQT. The capacity of the links in the network are also dynamic in [15, 16, 10], however all those models also limit significantly the change in the capacities and thus they do not study real faulty scenarios neither. Moreover, the results obtained in them show that the set of universally stable networks remains the same as in the AQT model under those models. The configuration and evolution of the systems considered in all those works are also very different to those suffered by the systems we present here: in [12, 15, 16, 10] the packets are always flowing, since the effect of a real failure (i.e., null link capacity or infinite slowdown) is never considered; thus packets are only buffered for congestion reasons, but never due to failures. Instead we consider that, when a failure occurs, the packet using the failed link/node cannot flow and must be retransmitted later. Then, one need to study how to store and manage those affected packets while the failure last. In our opinion, this is a more realistic way of tackling the problem.

In [8, 7], other models were considered for studying distributed balancing algorithms for dynamically changing input streams and flow routing in dynamically changing networks. In those models, the injected packets are defined by specifying only source and destination (no path is pre-specified), and only the single receiver case is considered. The adversary is thus restricted to guarantee that a static multi-commodity flow problem has a solution. The proposed load balancing algorithms are shown to keep the system stable when the adversary injection rate is one. 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, 7] have

\^\textsuperscript{1}Observe that, when retransmissions are considered, the packets affected by a failure compete again with other packets when the failed link/node is restored.
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 \( S \)). However, the focus of [8, 7] is on the balancing proposals and they do not deal with failures neither.

Another interesting approach in the literature considers input/output blocking in the nodes [5, 6]. Although this model does not consider failures on the network structure, the addition of input-output constraints that this model does on the arrival of packets seem to have some behavioral similarities with some of the models considered here. This might be the case for the models with failures at the links that block the reception of packets (i.e., namely the edge-nR and edge-nR-B models). From the point of view of networks, the results in [5, 6] show that networks containing very simple topologies (so as the one we reach in this work) can become unstable. However, from a protocol point of view, the results obtained in both works are completely opposed, for example sis is unstable while lis is universally stable.

The first work where failures were considered as such was [2], where two AQT-based models allowing faults (namely the failure model and the reliable model) were proposed to consider dynamic networks in which links can stop transmission. 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 \( \alpha \) such that \( r < \alpha \leq 1 \).

In both models, the adversary is not able to fail an edge permanently. Concerning the universal stability of networks, both models are equivalent to the AQT model. Concerning the universal stability of protocols, RTG, NFS, and SIS are universally stable but FIFO, LIFO, NTG and FFS are not as in the AQT model. However, the LIS protocol is universally stable in the AQT model, but it is not neither in the failure nor in the reliable model. This first work settled the basement for the work we present here.

It is worth mentioning that none of these existing models for dealing with dynamic networks consider the possibility of changing the capabilities of the nodes of the topology, but only of the links. Also the failure and reliable models in [2] only considered link failures. Here we also consider node failures.

**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 overloaded worst-case scenarios.

First, we study 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 rise 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 (extension -R) packets or not (extension -nR) and, in case it is not, on whether the affected packets are kept in the normal queues or in a special buffer (extension -B) that the node keeps to that aim.

Then, we propose a adversarial model and study stability of systems in which the faulty elements are the nodes. The model has four variations which we denote in the forthcoming as node-RnT, node-nRnT-B, node-nRnT and node-nRT models. The main difference among them relies on whether the queues of a failed node are able to receive (extension -R) packets or not (extension -nR) and, on whether a failed node that can not receive new packets while being failed, is able to transmit (extension -T) those packets that are already queued, or it is not (extension -nT).
In some of these cases, there might be also an additional buffer (extension B) where the affected packets are kept. To the best of our knowledge, this is the first time that the stability of dynamic adversarial systems with failing nodes is studied.

As a natural conclusion to our study of systems with (exclusively) link or node failures, we consider also systems in which both the nodes and the links may fail. The adversarial models we propose in this case are combinations of the models considered separated previously. Obviously, the non-stability properties are inherited from the link/node faulty systems to the combined one, and thus the interest in on stability results. We show that in the case that we combine link models and node models with a common positive stability property, then this property also holds 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 becomes 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 becomes unstable. The main contribution of this work is to show that the stability of a system depends 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.

On the methodological side we introduce the notion of system simulation and we use it to compare the global behaviour of models. This technique, which was first used in [2], allows us to perform a systematic and incremental study of faulty adversarial models. For the first time in this topic of research, we provide in this work with a formal definition for such technique and for several other concepts which were often used in the literature but never formalized. Our definitions are general enough to adapt naturally when different models are considered. That required a clear definition of what a model is, an effort that was never done until now and that resulted in a slight mess in the existing literature.

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. Also in Section 2, most of the key concepts that are used in this work (specially those concerning the adversaries and stability) are formalized.

In Section 3, 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 4 four adversarial models are presented whose focus is on studying the effect of node failures. 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 network 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 only 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 5. Finally, Section 6 posts some conclusions and review some open problems for future research.

2 Faulty model 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, \((N, A, P)\), where \(N\) is the network topology, \(A\) is an adversary defining the traffic and failure pattern on \(N\), and \(P\) 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 and the failure pattern. 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. 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. 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 3 and 4, when presenting the different adversarial models under study.

Short-lived failures. 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 no packet kept in its queue can be served. But we also have to consider what to do with the incoming packets. Since they cannot 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. 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 (and possibly also different nodes) deal with failures in different ways, remains open.

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 M)** A faulty adversarial model $\mathcal{M}$ specifies the following three 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 time step \( t \).

Before introducing the general restrictions that an adversary must obey we introduce some notation that will help on their description. Let \( A \) 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

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

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

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

\[
D_e(t) = F_e(t) + H_e(t) - F_e(t)H_e(t).
\]

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 maximum duration of any of those failures.

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

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

\[
N_e(I) \leq \lceil r(|I| - D_e(I)) \rceil + b \quad \text{and} \quad (|I| = D_e(I)) \Rightarrow |I| \leq \omega.
\]  \hspace{1cm} (2)

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

Confronting such \((r, b, \omega)\)-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 M)** We say that \( S = (\mathcal{N}, \mathcal{A}, \mathcal{P}) \) is a valid system in model \( M \) if \( \mathcal{A} \) is an \((r, b, \omega)\)-adversary in model \( M \), for some \( r, 0 < r < 1 \), and some \( b, \omega \geq 0 \), that controls the type of failures allowed in model \( M \).
The evolution over time of a system under a model is fully determined by the model, i.e., by the queue management, accessibility of the queues, the existence of additional buffers, and the restrictions that the model puts on the allowed failure types, link, edges, or both. Thus, apart from the components of a system, the model in which the system is considered 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 be 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 \( S = (N, A, P) \) be a valid system in model \( M \). \( S \) is stable in model \( M \) 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 \( c(r, b, \omega, N) \) as the system dynamically evolves in time according to \( M \).

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 \( P \))** A network \( N \) is \( r \)-stable under protocol \( P \) in model \( M \), if for any \( b, \omega \geq 0 \) and any \( (r, b, \omega) \) adversary \( A \) for \( N \) in model \( M \), the system \((N, A, P)\) is stable in model \( M \).

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

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

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

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

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 \( S = (N, A, P) \) be a system in model \( M \), and let \( S' = (N', A', P') \) be a system in model \( M' \). The system \( S \) is simulated by the system \( S' \) if all the following conditions hold:

- when \( A \) injects at time \( t \) a packet \( p \) with route \( \rho \) in the system \( S \), the adversary \( A' \) injects a corresponding packet \( p' \) at the same time \( t \) with the same route \( \rho \) in the system \( S' \),
- every time \( t \) that, according to \( P \), the packet \( p \) crosses the edge \( e \) of \( N \) in the system \( S \), the packet \( p' \) crosses in \( S' \), according to \( P' \), the same edge \( e \) of \( N' \) 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 \( S = (N, A, P) \) be a system in model \( M \) that can be simulated by a system \( S' = (N', A', P') \) in model \( M' \). If the system \( S' \) is stable in model \( M' \), then system \( S \) is stable in model \( M \).

Since any \((r, b)\)-adversary in the AQI 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 AQI model. Thus, for any of the faulty adversarial models \( M \) introduced in this paper, we can show trivially that

Fact 1 Any system that is not stable in the AQI model, remains not stable in model \( M \).

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

3 Failures on links

In this section, we focus on the study of stability conditions in dynamic networks in which only the edges may fail. The constraints of an adversary in such faulty environments correspond to the restriction given in (2), in this cases only links can be failed. We study the stability of networks and protocols 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 consider three situations:

---

2 When the context is clear enough and we want to focus on some property of the topology \( G \) of a network \( N' \), we will do an abuse of terminology and refer to \( G \) as the network.
(type 1) 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.

(type 2) 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.

(type 3) 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 packet leaves the queue of a link during its failure.

3.1 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\(^3\) 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.\(^4\) 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 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

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

\(^3\)Note that, when an edge e is failed during an interval I, then \(N_e(I) \leq b\) packets can still be injected.

\(^4\)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).
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:

**Theorem 2** Given a network $\mathcal{N}$, any $(r, b)$-adversary $A$ 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 $\omega$.

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

**Proof.** Recall that $\omega$ 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 $\frac{\omega}{\omega + 1}|I|$. Then, we can rewrite the constraint

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

for an $(r, b, \omega)$-adversary $A$ in the edge-R model as

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

therefore,

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

$$= r|I| + (1-r)F_e(I) + b$$

$$\leq r|I| + ((1-r)\omega/(\omega + 1))|I| + b$$

$$= ((r + \omega)/(\omega + 1))|I| + b.$$  

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

\[\]
Since the behavior of the queues and the systems in the failur e model in [2] and in the edge-R model is the same, we have (from Theorems 2 and 3) that any valid system in the edge-R model 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 the edge-R model is the same as 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:

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

**Corollary 5** A network $N$ is universally stable in the edge-R model if, and only if, $N$ is universally stable in the $\mathcal{AQT}$ 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 $\mathcal{AQT}$ 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 $\mathcal{AQT}$ model [13] and not known to be universally stable in the failure model. We can show that LIQ is not universally stable in the failure model.

**Lemma 7** LIQ is not universally stable in the failure model.

The proof of this lemma is given in the appendix, and it is the case that the adversary given in that proof makes also the system unstable under LIS. Therefore, taking into account Theorem 2, we can conclude the following result.

**Corollary 8** LIQ and LIS are not universally stable in the edge-R model.

In order to enhance the readability of the paper, we placed most of the instability proofs to be included in this paper into an appendix. That is the case also for the proofs of Lemma 7 and Lemma 8.

### 3.2 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 4.

We propose the edge-nR model, in which the network has non-receiver links as the ones just described. In this model, we show that there are very simple topologies, like line graphs, that turn out not to be universally stable. Observe that in this model when a packet cannot traverse a link it remains in the queue of the previous link and so it competes with a different set of packets in subsequent steps.

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

**Theorem 9** 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.
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 10** 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.

From the previous Theorems 9 and 10, it follows Corollary 11:

**Corollary 11** 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, the graph given in Figure 11.

**Lemma 12** LIS and LIQ are not universally stable in the edge-$nR$ model.

### 3.3 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 6.
Let us consider the behavior of edge $e_1$. We show by induction on $t$ that when no failures occur, the protocol $P$–if $p$–Q$(t)$–with $p$–Q$(t)$–is the same selected by $P$–if $p$–Q$(t)$–. W e propose the edge-nR-B model. In this model, the greedy protocol $e$-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 13** Let $N$ be a network in which all the nodes have output degree at most one. Any system $S = (N, A, P)$ in the edge-nR-B model can be simulated by a system $S' = (N, A, P')$ in the edge-R model.

**Proof.** When no failures occur, the protocol $P'$ is a greedy protocol $P$ makes the predicate to hold trivially. When failures occur in the system $S$, the packets requiring to traverse any failed edge $e = (u, v) \in E(N)$ are kept in the extra buffer at node $u$. We simulate this system $S = (N, A, P)$ by $S' = (N, A, P')$ in the edge-R model, the greedy protocol $P'$ 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 $u \in V(N)$ and the queue corresponding to edge $e \in E(N)$, 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 $P'$ 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 $P$. 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:

- $Q'_e(t) = Q_u(t) \cup Q_e(t)$, and
- if $p$ crosses in $S$ the link $e$ of $N$ 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 two 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 $P$ and selects a packet $p^*_e$ in order to be scheduled. Hence, $Q_e(t+1) = (Q_e(t) \cup L_e \cup I_e \cup P_u) \setminus \{p^*_e\}$ 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 $P'$ can select the same packet $p^*_e$ for being scheduled. Then we have that $Q'_e(t+1) = (Q'_e(t) \cup L_e \cup I_e) \setminus \{p^*_e\} = (Q_e(t) \cup Q_u(t) \cup L_e \cup I_e) \setminus \{p^*_e\} = Q_e(t+1) \cup Q_u(t+1)$.

![Figure 6: The edge-nR-B model: Behavior when no failure occurs (left) and when it does (right).](image-url)
2. \( e \) is failed. In this case the queue of edge \( e \) in system \( S \) can not receive any packet. If a packet \( p_e \in L_e \) 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 \( t + 1 \) they are also stored in the extra buffer at \( u \). Then we have that \( e(t + 1) = Q_e(t) \) and, as \( u \) has output degree 1, \( Q_u(t + 1) = Q_u(t) \cup I_e \).

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 \( p_e \) injected by the adversary and packet \( p_e \in L_e \) scheduled at time step \( t \). Then we have that \( Q_e'(t + 1) = Q_e(t) \cup L_e \cup I_e = Q_e(t + 1) \) and \( Q_u(t + 1) = Q_u(t + 1) \).

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

Proof. When no failures occur, the protocol \( P' = P \) make the predicate to hold trivially. When failures occur in the system \( S \), the packets requiring to traverse any failed edge \( e \in E(N) \) are kept in the queue of edge \( e \). We simulate this system \( S = (N, A, P) \) in the edge-R model by the system \( S' = (N, A, P') \) in the edge-nR-B model. Protocol \( P' \) is defined 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 \( P' \) in such a way that at any time step the packet selected to traverse \( e \) is the same selected by \( P \). Taking this into account, the rest of the proof is similar to the one of Theorem 13.

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

**Corollary 15** 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 16** 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.

In contrast to the edge-nR model, even if 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.

By considering the graph and adversary used for the proof of Lemma 7, we can also show that the protocols LIS and LIQ are not universally stable in the edge-nR-B model. Therefore, we have that

**Lemma 17** LIS and LIQ are not universally stable in the edge-nR-B model.
4 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 only nodes can fail. 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 failure management:

(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 queueing 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.

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.

4.1 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 7.\(^5\) We consider now the node-RnT model, in which the network has receiver non-transmitter nodes.

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.

\(^5\)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.
Theorem 18 Any system $S = (N, A, P)$ in the node-RnT model can be simulated by a system $S' = (N, A', P)$ in the edge-R model.

Proof. The new adversary $A'$ is obtained from $A$ by keeping the same sequence of packets injections and replacing any failure of a node $u \in V(N)$ 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 $A$ in system $S$ coincides with the value $F_e(I)$ incurred by adversary $A'$ in $S'$. Therefore, from the definition of the models it is straightforward to show that when $A$ is an $(r, b, \omega)$-adversary in model node-RnT then $A'$ is also an $(r, b, \omega)$-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 19** 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 AQ7 model. Therefore, taking into account Fact 1, we obtain

**Corollary 20** A network $N$ is universally stable in the node-RnT model if, and only if, $N$ is universally stable in the AQ7 model.

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

**Corollary 21** 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 AQ7 model extends to the node-RnT model and, taking into account Theorem 18, any universal stability result in the edge-R model extends to the node-RnT model. We then get

**Corollary 22** 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 AQ7 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 23** lis and liq are not universally stable in the node-RnT model.
4.2 The models with non-receiving nodes

In the failure management forms we consider here, during the failure of a node the queues do not receive (i.e., are not accessible) we have to analyze whether they can transmit or not. When a link is unable to transmit the packets wanting to cross an edge \((u, v)\) must wait at their output queue in \(u\) until \(v\) recovers. The possible behaviors are depicted in Figure 8 and Figure 9.

Following previous cases, we propose two models, the node-nRT model, in which the network has non-receiver but transmitter nodes, and the node-nRnT model, in which the network has non-receiver non-transmitter nodes. We will consider also a variation of the later the in which \(u\) has in addition a buffer for the received packets that cannot be put on the blocked queues, the node-nRnT-B model. This additional buffer is 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 no access for all the queues associated to 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 10.

Although it is easy to show that on networks whose vertices have all output degree at most one the failure of a node in the node-nRnT or the

Theorem 24 Any network with at least one link can be made unstable in the node-nRnT and the node-nRT models.

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 25 There are no universally stable greedy protocols neither in the node-nRnT model nor in the node-nRT model.

In the node-nRnT-B model, we focus 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 26 Let $P$ be a greedy protocol and $N$ a network in which all the nodes have output degree at most one. Any system $S = (N, A, P)$ in the node-nRnT-B model can be simulated by a system $S' = (N, A', P')$ in the edge-R model.

Proof. The new adversary $A'$ is obtained from $A$ 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 $A$ in system $S$ coincides with the value $F_e(I)$ incurred by adversary $A'$ in $S'$. Therefore, from the definition of the models it is straightforward to show that when $A$ is an $(r, b, \omega)$-adversary in model node-nRnT-B then $A'$ is also an $(r, b, \omega)$-adversary in model edge-R.

We consider the protocol $P'$ defined in such a way that, from any node $u$, the packet that is transmitted through the unique outgoing edge $e = (u, v) \in E(N)$ is selected by $P$ 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 $P'$ guarantees that at any time step the set of packets stored in the queue $Q'_e$ associated to $e' \in E(N)$ 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 27 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 28 For every injection rate $\frac{1}{2} < r < 1$, 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.

Corollary 29 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 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. The restrictions on an $(r, b, w)$-adversary in an adversarial model $M$ are described by the Equation 2.

We consider combined faulty adversarial models in which the queue management for a failed link implements one of the models presented in Section 3 (namely the edge-R, edge-nR or edge-nR-B model), and the queue management for a failed nodes follows one of the models presented in Section 4 (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, 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 3. Table 2 summarizes the results obtained concerning the universal stability of the same networks and protocols for the models presented in Section 4.

Table 1: Universal stability property in the combination of adversarial models in which the nodes do not fail, i.e., they implement the AQTA 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 AQTA universal stable protocols mentioned in this paper, i.e. SIS, NFS, and FTG.

<table>
<thead>
<tr>
<th>MODELS</th>
<th>edge-R</th>
<th>edge-nR-B</th>
<th>edge-nR</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-faulty node</td>
<td>ring</td>
<td>ring</td>
<td>ring</td>
</tr>
<tr>
<td></td>
<td>dags</td>
<td>dags</td>
<td>dags</td>
</tr>
<tr>
<td></td>
<td>LIS, LIQ</td>
<td>LIS, LIQ</td>
<td>LIS, LIQ</td>
</tr>
<tr>
<td></td>
<td>other-USP</td>
<td>other-USP</td>
<td>other-USP</td>
</tr>
</tbody>
</table>

Table 2: Maintenance 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.

<table>
<thead>
<tr>
<th>MODELS</th>
<th>node-RnT</th>
<th>node-nRnT-B</th>
<th>node-nRnT</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-faulty edge</td>
<td>ring</td>
<td>ring</td>
<td>ring</td>
</tr>
<tr>
<td></td>
<td>dags</td>
<td>dags</td>
<td>dags</td>
</tr>
<tr>
<td></td>
<td>LIS, LIQ</td>
<td>LIS, LIQ</td>
<td>LIS, LIQ</td>
</tr>
<tr>
<td></td>
<td>other-USP</td>
<td>other-USP</td>
<td>other-USP</td>
</tr>
</tbody>
</table>

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 3 and 4 (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 30** Any system $S = (\mathcal{N}, \mathcal{A}, P)$ in the (edge-R,node-RnT) model can be simulated by a system $S' = (\mathcal{N}', \mathcal{A}', P)$ in the edge-R model.
Proof. Let $A$ be an $(r, b, w)$-adversary for $N$ in the $(\text{edge-R}, \text{node-RnT})$ model. Assume that $A$ at time $t$ has associated triple $< P_t, E_t, V_t >$ following the same idea used in Theorem 18, we replace node failures by outgoing links failure. Thus, $A'$ at time $t$ has associated tuple $< P_t, E'_t, V_t >$ where

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

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

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

Corollary 31 Directed acyclic graphs and rings are universally stable in the $(\text{edge-R}, \text{node-RnT})$ model.

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

Corollary 32 A network $N$ is universally stable in the $(\text{edge-R}, \text{node-RnT})$ model if, and only if, $N$ is universally stable in the $\text{AQT}$ model.

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

Corollary 33 The universal stability of a given network in the $(\text{edge-R}, \text{node-RnT})$ model can be decided in polynomial time.

From Theorem 26 we know that any system $S = (\mathcal{N}, A, P)$ in the $\text{node-RnT}$ or $\text{node-nRnT-B}$ model, in which all the nodes of $\mathcal{N}$ have output degree at most one, can be simulated by a system $S' = (\mathcal{N}', A', P')$ in the $\text{edge-R}$ model. Therefore, for the particular case of lines and rings, we have the following result when taking into account Corollary 4

Theorem 34 Lines and rings are universally stable in the $(\text{edge-R}, \text{node-RnT})$ model and in the $(\text{edge-R}, \text{node-nRnT-B})$ model.

From Theorem 13 we know that any system $S = (\mathcal{N}, A, P)$ in the $\text{edge-nR-B}$ model, in which all the nodes of $\mathcal{N}$ have output degree at most one, can be simulated by a system $S' = (\mathcal{N}', A', P')$ in the $\text{edge-R}$ 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 35 Lines and rings are universally stable in the $(\text{edge-nR-B}, \text{node-RnT})$ model and in the $(\text{edge-nR-B}, \text{node-nRnT-B})$ model.

As a consequence of the instability results obtained in Sections 3 and 4, 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 $(\text{edge-R}, \text{node-RnT})$ combined adversarial model remains to be studied. As we have just shown in Theorem 30, systems in the $(\text{edge-R}, \text{node-RnT})$ model can be simulated by systems in the $\text{edge-R}$ model, and this allows us to transfer any positive network or protocol stability result from the $\text{edge-R}$ model to the combined $(\text{edge-R}, \text{node-RnT})$ model. Thus, from Corollary 6 we obtain the following result:

Corollary 36 SIS, FTG, and NFS are universally stable in the $(\text{edge-R}, \text{node-RnT})$ model, while FIFO, LIFO, LIS, LIQ, NTG, and FFS are not universally stable in the $(\text{edge-R}, \text{node-RnT})$ model.
Table 3: Maintenance of the universal stability property in faulty adversarial models. Symbol √ means that the network (respectively, the protocol) is universally stable, while symbol × means it is not. Symbol other-USP represent the other AQ$T$ 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 AQ$T$ model.

6 Conclusions

The main contribution of this paper is twofold: first, the rigorous and formal definition of concepts and new techniques that was never done before; second, the exhaustive study and analysis of the effect of edge or/and node failures on the stability of networks and protocols in relation to the way those failures are administrated. To the best of our knowledge, node failures were never studied before when studying stability in the context of adversarial systems.

Using as base model the AQ$T$ for static networks, we have proposed extensions for dynamic packet-switched networks in which short-lived link and/or edge failures might occur. Different models can be considered, depending on how the management of link failures or node failures affects the accessibility to their queues and depending also on where the affected packets are stored during the failure. Table 3 summarizes the most relevant results of our work concerning the universal stability of networks and protocols. The rows in Table 3 consider the node-faulty models, while the columns consider the edge-faulty models, both from less to more restrictive. Thus, the results at the leftmost upper cell represent the results for AQ$T$, and the rest of the results at the first row and column of the table state the results for the homogeneous cases, i.e., for the (exclusively) edge-faulty or node-faulty models. The rest of cells show the results for the combined models, where both link and nodes might fail.

Concerning networks, it is worth mentioning the fact that only in one of the model combinations (edge-R with node-RnT) the property of universal stability is robust with relation to AQ$T$. In the rest of combinations, even very simple directed acyclic network topologies (and thus of course any network containing those topologies) can be made unstable when failures occur. Ring topologies remain universally stable under more models (and combinations of them) than simple directed acyclic topologies do. Interestingly enough, rings become not universally stable as soon as the failures at the edges and/or at the nodes block their respective receiving capabilities and no extra buffers are available.

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 network stability) when related to the AQ$T$ 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, however it is quite unrealistic as assumption. 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.

Concerning protocols, it is interesting to observe how LIS and LIQ, which are known to be universally stable under AQT, loose easily this property as soon as failures are considered, even when they occur only at nodes or only links and even when those failures are managed in the simplest way. The protocols SIS, NFS, and FTG (which are also universally stable under AQT) remain so under the receiving models edge-R and node-RnT and their combination, but they also loose the property as soon as failures impede the reception of packets.

Open problems

Focusing on 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. This characterization will allow to assess the equivalence of models from the point of view of network universal stability, as we have done between the AQT model and the edge-R and node-RnT models and the model that combines both. Although we believe that the edge-nR-B model is equivalent, in this sense, to the node-nRnT-B model, to prove this fact an exact characterization of the universally stable networks in both models is needed. In relation to this, another related interesting question is whether the universally stability of networks under faulty models can be decided in polynomial time.

The models presented in this work consider that all the links, respectively nodes, of the network share same failure management strategy. It would be of interest to study the stability of heterogeneous systems, i.e., 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 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 guaranties or, from the opposite point of view, knowledge about which system configurations are not convenient because of their potentiality to provoke instability.

Observe that we have got very strong instability results as soon as the queues 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.

Other models for dealing with adversarial traffic and failure under other forms of failure management are of interest. For example, adversarial models that drop those packets that suffer long delays due to failures along their traversal, and require a posterior packet retransmission. One might also require additional properties on the trajectories of the packets, in order to minimize (or to overcome) the effect of a failure, for example 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.

The methodology used in this work to obtain most of our proofs of stability is based on system simulations. We believe that such technique will be helpful in comparing the global behaviour of different adversarial models that incorporates other features of interests.
References


Appendix

Most of 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.

In order to enhance the readability of the paper, we placed all the instability proofs of the paper which based on induction, into this appendix.

A Some proofs of instability

Lemma 7 \textit{LIQ} is not universally stable in the failure model.

\textbf{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 11. 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 under \textit{LIQ}.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{network.png}
\caption{Network $\mathcal{N}$, with $V(\mathcal{N}) = \{u, v, w\}$ and $E(\mathcal{N}) = \{(u, w), (w, v), (u, v), (v, u)\}$}
\end{figure}

\textit{Round 1}: For the first $s$ steps, we inject a set $\alpha$ 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.

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

\textit{Round 3}: For the next $r^2s$ steps we inject a set $\delta$ of $r^3s$ packets that follow the path $\{uwv\}$ together with a set $\eta$ 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 $\{uw\}$ and $r^3s$ packets waiting to traverse the path $\{uwv\}$.

\textit{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 $\{uwu\}$ and $r^4s$ packets waiting to traverse the path $\{uwv\}$.

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 by repeating the set of rounds the number of packets in the system grows unboundedly, thus making the system not stable under \textit{LIQ}.

Observe that the proposed adversary obeys the restrictions of the failure model.
Theorem 9. 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-$n\mathcal{R}$ model under protocols SIS, LIFO, NTG, NFS, and FFS.

Proof. Let $k = \lceil 2/r \rceil$, $\ell = \lceil k/r \rceil$ and $n_0 = 5 + \ell$. Observe that $rk > 1$. Consider a $n_0$-line graph with edges labelled as in Figure 5(a). 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 $\ell$ steps, we inject a set $\alpha$ 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 $n_0$ steps, this guarantees that all the $\alpha$ packets will be waiting at $e_c$ at the end of the round. So, at the end of this round there are $k$ packets queued at $e_c$.

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

Round 4: Edge $e_d$ and $e_c$ fail for $n_0$ steps. Again this provides enough time to guarantee that all the $\beta$ 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 $\gamma$ 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 $\beta$ packets queued at $e_b$ leave and, at the end of this round, there will be the $k$ packets of the $\alpha$ 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 $n_0$ steps. Thus all the set $\gamma$ will be waiting at $e_c$ after those steps. 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 6, 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 $\omega = n_0$.

The above adversary will make unstable any network containing a path of length $n_0$, in particular any $n$-line graph with $n \geq n_0$.

Theorem 10. 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-$n\mathcal{R}$ model under FTG.

Proof. Let $k = \lceil 2/r \rceil$, $\ell = \lceil k/r \rceil$ and $n_0 = 4 + \ell$. Observe that $rk > 1$. Consider a $\ell$-fork graph with edges labelled as in Figure 5(b). 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 $\phi$ there are $k + \phi rk$ packets queued at edge $e_c$ with destination $e_d^\phi$. 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 $\phi$ that the number of packets in the system grow up unboundedly.

Round 1: For $\ell$ steps, $k$ packets with destination $e_c$ 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 $e_c$ fail for $n_0$ steps. At the end of the round, packets with destination $e_d$ or $e_c$ are waiting 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 $e_c$.

\footnote{To start from an empty initial configuration just consider two initial preceding rounds: (1st round) For $k/r$ steps, inject $k$ packets at $e_a$ with destination $e_d$ and, (2nd round) edge $e_d$ fails enough to guarantee that all the packets get blocked at the queue of $e_c$.}
Round 3: For the next \( k \) steps, the adversary injects a set \( \beta \) of \( rk \) packets at edge \( e_a \) with destination \( e_v \). Since neither edge \( e_d \) nor edge \( e_c \) fail a tie arises. FTG solves the tie in such a way that the \( k \) packets at the queue of \( e_v \) with destination \( e_c \) will flow to \( e_c \).

Round 4: Edge \( e_d \) fails for the next \( n_0 \) steps. Time enough to guarantee that the \( \beta \) 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 \( \omega = n_0 \). Again the adversary will make unstable any network containing a \( n_0 \)-fork graph.

\[ \square \]

Lemma 12 \textsc{lis} and \textsc{liq} are not universally stable in the edge-\( nR \) model.

Proof. Consider the network \( \mathcal{N} = (V,E) \), with vertex set \( V(\mathcal{N}) = \{u,v,w\} \) and edge set \( E(\mathcal{N}) = \{(u,w),(w,v),(v,u)\} \), depicted in Figure 11. Assume an initial configuration consisting of \( s/2 \) packets located at the node \( u \) that want to cross the path \( \{uvwu\} \) and a set of \( s/2 \) packets located at the node \( u \) that wants to cross the path \( \{uv\} \).

The adversary follows the same four rounds of the adversary used for the proof of Lemma 7. Since no failures occur during the first three rounds, at the end of the third round, both under \textsc{liq} and \textsc{lis}, the packet distribution on the network is as follows:

- there are \( r^3s \) packets waiting to traverse only the edge \( (u,v) \), and
- \( r^3s \) packets waiting at node \( u \) to traverse the path \( \{uvwu\} \).

In the fourth round, the adversary makes the link \( (w,v) \) fail and, since we are dealing with the edge-\( nR \) model, they packets remain blocked at their initial position at node \( u \) (i.e., with path \( \{uvwu\} \) still to traverse).

Thus, at the end of the fourth round, the system has a configuration analogous to the initial one but with \( 2r^3s \) packets in the system instead of \( s \). For any injection ratio that makes \( 2r^3s > s \) (i.e., for \( r > 0.841 \)), the infinite repetition of the described rounds makes the system not stable, both under \textsc{liq} and \textsc{lis}.

\[ \square \]

Theorem 16 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: \textsc{lifo}, \textsc{sis}, \textsc{ntg}, \textsc{ftg}, \textsc{nfs}, and \textsc{ffs}.

Proof. Given a valid injection rate \( r \), let \( k = \left\lceil 2/r \right\rceil \), \( \ell = \left\lceil k/r \right\rceil \) and \( n_0 = 4 + \ell \). Observe that \( rk > 1 \). Consider a \( 4 + \ell \)-fork like as the one depicted in Figure 5(b), whose length depends on \( r \). Again we suppose that the initial configuration is empty. The following adversary six can make the network not stable for any of the above-mentioned greedy protocols.

Round 1: For the first \( \ell \) steps, the adversary injects a set \( \alpha \) of \( k \) packets of the form \( \{e_a, \ldots, e_d\} \).

Round 2: The edge \( e_d \) fails for the next \( n_0 \) steps, time enough to make all the set \( \alpha \) accumulate in the queue of node \( v_3 \).

Round 3: For \( \ell \) steps, the edge \( e_d \) continues failing and a set \( \beta \) of \( k \) packets are injected, which wish to traverse the path \( \{e_a, \ldots, e_c\} \).

Round 4: The edges \( e_d \) and \( e_c \) fail for the next \( n_0 \) steps. This is enough to accumulate the packets from the \( \alpha \) and the \( \beta \) 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 \( e_c \) fail, \( k \) packets will leave the queue at edge \( v_3 \). The adversary injects a
set \( \gamma \) 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 \( \beta \) set. An additional round has to be considered.

**Round 6:** The edge \( e_d \) fails for \( n_0 \) steps, blocking the packets from \( \alpha \) and the packets from \( \gamma \) 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 3 to 6, after rounds 1 and 2, 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 \( \omega = n_0 \).

**Lemma 17** LIS and LIQ are not universally stable in the edge-nR-B model.

**Proof.** Consider the graph and adversary given for the proof of Lemma 7. Observe there, in Figure 11 of the proof of Lemma 7, that 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.

**Lemma 23** LIS and LIQ are not universally stable in the node-RnT model.

**Proof.** This result can be easily shown by applying the adversary given in the proof of Lemma 7 over the graph given in Figure 11, where every failure of edge \((w, u)\) is replaced by the failure of the node \( w \).

**Theorem 24** Any network with at least one link can be made unstable in the node-nRnT and the node-nRT models.

**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 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, \( \omega = q \) and that this result is independent of the protocol used in the system and also independent of the length of the line.

**Theorem 28** For every injection rate \( \frac{1}{2} < r < 1 \), 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 12, where we denote as \( \lambda_1 \) and \( \lambda_2 \) 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 $\lambda_1$ with destination $v_1$, and also $rn$ packets at the first node of $\lambda_2$ 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.

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

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