A characterization of universal stability in the adversarial queueing model *†

Carme Àlvarez Maria Blesa Maria Serna

Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica Catalunya
Campus Nord, E-08034 Barcelona, Spain
{alvarez,mjblesa,mjserna}@lsi.upc.es

Abstract

We study universal stability of directed and undirected graphs in the adversarial queueing model for static packet routing. In this setting, packets are injected in some edge and have to traverse a predefined path before leaving the system. Restrictions on the allowed packet trajectory provide a way to analyze stability under different packet trajectories. We consider five packet trajectories, two for directed graphs and three for undirected graphs, and provide polynomial time algorithms for testing universal stability when considering each of them. In each case we obtain a different characterization of the universal stability property in terms of a set of forbidden subgraphs. Thus we show that variations of the allowed packet trajectory lead to non-equivalent characterizations.

Using those characterizations we are able to provide also polynomial time algorithms for testing stability under the NTG-LIS protocol.

1 Introduction

One of the main goals in the study of the behavior of packet-switched communication networks is to determine the conditions of *stability*, i.e., the fact that the number of packets in the system remains bounded as the system dynamically evolves in time. Stability is studied considering that a communication system is formed by three main components: the network, the scheduling protocol and the traffic pattern. *Networks* are modeled with (directed or undirected) graphs in which nodes represent the hosts and edges represent the links between these hosts. The *protocol* is used to schedule a packet when more than one packet want to cross the same edge at the same time step. Packets waiting to traverse an edge are kept in a queue at the tail of the edge. The protocol determines the order in which the waiting packets cross the edge. The *traffic pattern* controls where and how packets join the system and defines their trajectory.

*Work partially supported by the FET Programme of the EU under contract numbers IST-1999-14186 (ALCOM-FT) and IST-2001-33116 (FLAGS), and by the Spanish CICYT projects TIC-2001-4917-E and TIC-2002-04498-C05-03 (TRACER). M. Blesa is also supported by the Catalan Research Council of the Generalitat de Catalunya (grant 2001FI-00659).

†Part of this work was presented in SPAA 2002.

Adversarial Models. The problem of deciding stability has been investigated under various models of packet routing [8, 9, 7, 13, 3]. Some models make probabilistic assumptions on the traffic pattern, while others replace more traditional stochastic arrival assumptions by worst-case inputs in the traffic pattern to perform a worst-case analysis. These latter are closer to the traditional analysis of algorithms and to real network configurations. The *Adversarial Queueing Theory* proposed by Borodin *et al.* [6], which is a robust model of queueing theory in network traffic, can be considered as the pioneering work in studying stability via worst-case analysis.

Adversarial models consider the time evolution of a packet-routing network as a game between an adversary and a protocol. The adversary may inject a set of packets at some nodes at each time step. In this work, we consider static packet routing; in this setting, the adversary also specifies for each packet the complete path that the packet must traverse.

Adversarial models have been shown to be good theoretical frameworks for traffic pattern in modern communication networks. These models can reflect the behavior of connection-oriented networks with transient connections (such as ATM networks) as well as connectionless networks (such as the Internet). Different factors are used in those adversarial models to describe the adversary and quantify its power. Three of these factors refer to the *injection rate* (i.e., the frequency at which the adversary introduces packets into the network), the *burstiness* (i.e., the maximum number of packets that can be injected in an edge in one step), and the *initial configuration* (i.e., the initial quantity of packets distributed over the network at time zero). Restrictions on the quantity of packets injected are introduced in order to avoid adversaries that could trivially collapse the system. In general, during each interval of time (that can be defined in different ways), the number of packets injected during that time interval requiring any edge in their trajectory cannot exceed a certain bound proportional to the length of the time interval. Two main adversarial models have been considered in the last years, both of them assuming an empty initial configuration:

The *windowed* adversarial (queueing) model by Borodin *et al.* [6]. An adversary in this model is restricted by two parameters (w, r), where $w \geq 1$ is the window size and $0 < r \leq 1$ is the injection rate. The adversary is allowed to inject sequences of packets under the restriction that, at any w consecutive time steps, the total amount of packets requiring any concrete edge e is, at most, $\lceil rw \rceil$.

The *leaky-bucket* adversarial model by Andrews *et al.* [4]. An adversary in this model is also restricted by two parameters (b, r), where $b \geq 0$ is the burstiness and $0 < r \leq 1$ is the injection rate. The adversary is allowed to inject sequences of packets under the load condition that, of the packets that the adversary injects in any interval T, at most $\lceil r|T| \rceil + b$ may have paths that contain the same particular edge.

In a recent work Rosén compares the relative power of the *windowed* and the *leaky-bucket* adversarial models [20]. For injection rates $r < 1$, adversaries in one model can be simulated by adversaries in the other model injecting the same sequence of packets. Thus, the results on one model also hold on the other model. However, when $r \leq 1$ a leaky-bucket adversary is more powerful than a windowed adversary, since there are some sequences of packets that can be produced by the former when $r = 1$, but cannot be produced by the latter.

In this work we will study universal stability of directed and undirected graphs when $r < 1$ under the leaky-bucket adversarial model, as it is done in [4]. In general, stability results are shown for any network with an empty initial configuration, while instability results are shown starting form a network with a given non-empty initial configuration (see [6] or [4]). The results in [4] show that any instability result for a network with a given initial configuration can be translated into an instability result with empty initial configuration for a different network. This results can be used to show the non-universal stability of a given protocol but not to characterize universal stability of networks. However, we will show that network stability is independent of whether the initial configuration is empty or not.

Greedy protocols are those that forward a packet across an edge e whenever there is at least one packet stored in the queue of e waiting to traverse the edge. Some natural greedy protocols are: *first-in-first-out* (FIFO), in which highest priority is given to the packet that has arrived first at the queue, *longest-in-system* (LIS), in which every queue gives priority to the packet that has been in the system the longest and, *nearest-to-go* (NTG), in which highest priority is assigned to the packet that still has to cross the smallest number of edges. Other protocols are *last-in-first-out* (LIFO), *shortest-in-system* (SIS), *farthest-to-go* (FTG), *nearest-from-source* (NFS), and *farthest-from-source* (FFS).

In this paper, as it is usual in the literature about stability, we only consider greedy protocols. All the instability results in this paper will be shown under the NTG-LIS protocol, which works as the NTG protocol, and solves ties using LIS.

The packet trajectory refers to the form of the paths that packets are allowed to follow over the network. For directed graphs, two different types of paths have been considered in the literature. Borodin *et al.* [6] and Andrews *et al.* [4] assume that when a packet is injected, its assigned path does not contain any edge more than once (see page 45 of [4]); however, a different definition of packet trajectory was considered by Goel in [15]. Although a detailed definition is not given in [15], only paths that do not contain any vertex more than once are allowed (see for example the comment just before Lemma 2.4 on page 221 of [15]). Hence, the paths considered by Goel (refereed to in the following as '*simple paths*') are a restrictive version of the paths considered by Andrews *et al.* (refereed to in the following as '*paths*').

In the case of undirected graphs, Andrews *et al.* assume that packets can only use paths that do not cross the same edge twice. This means that a packet can not traverse the same edge in both directions (see page 57 of [4]). We refer to these paths as '*undirected paths*'. Together with this type of paths, we will also consider '*paths*' and '*simple paths*' defined over the directed version of the undirected graph.

Gamarnik [14] considers undirected graphs and uses a different model for the allowed packet traffic: each edge is undirected and can only carry a single packet in one step. This contrasts with the Andrews *et al.* model in which the edges can be seen as bi-directed, and each edge can carry a packet in each direction at each step.

Known results. A network is said to be universally stable when the systems composed by that concrete network are stable regardless of the selected protocol and traffic pattern. A protocol is said to be universally stable when all the systems that use it are stable. The existence of networks and scheduling protocols that are, respectively are not, universally stable was initially shown in [6] and [4].

Until the work of Rosén [20], it was not clear if both the windowed and the leaky-bucket models were equivalent or not. Since it was shown that they are for injection rates $r < 1$, results on one model also hold on the other model when the injection ratio accomplishes this condition. Keeping this equivalence in mind, let us summarize anyway some of the most important results obtained in the respective models.

In the *windowed adversarial model*, universal stability of networks with tree, mesh and directed acyclic topologies was shown to hold in [6]. On ring topologies, protocols LIS and FIFO were shown to be non-stable with injection rate $r = 1$, whereas FTG was shown to be stable. Concerning only protocols, it was also shown in [6] that, for every $r > 0$, there exists a queueing network for which NTG is non-stable at rate r. Every greedy protocol is shown in [11] to be stable if the injection rate is not more than $1/d$, where d denotes the diameter of the network.[1] Much effort has been dedicated to the FIFO protocol, for which the best known lower bound for instability was improved in [18] down to 0.5. Recently, FIFO has been shown to be non-stable even at arbitrary low injection rates [16].

In the *leaky-bucket adversarial model* with $r < 1$, stability of networks describing a ring topology was shown to hold under any greedy protocol [4]. In the same work, some commonly used simple greedy protocols (namely FTG, NFS, SIS and LIS) were shown to be universally stable, while some others (namely FIFO, LIFO, NTG and FFS) were shown not to be universally stable. Considering that the system might have initial configuration, stability and instability properties of FIFO have been recently studied in this model. A network-dependent constant is provided in [10] such that FIFO is stable against any adversary with smaller injection rate. A lower bound of 0.749 for the instability of FIFO was given in [17], where also the stability of networks with heterogeneous protocols was addressed. Moreover, in [21] it was shown that FIFO is stable when the injection rate is smaller than $1/(d-1)$. In this model FIFO can also be non-stable at arbitrary low injection rates [5].

With regard to the universal stability of networks, one of the first questions that arose was whether it would be possible to detect stability from the knowledge of the topological structure of the network and the scheduling protocols. For undirected graphs, Andrews *et al.* show in [4] that, for a particular type of packet trajectory (the one we call *undirected path*), cycles and trees are universally stable. Furthermore, they also show that the family of *undirected-path universally stable* graphs is minor-closed and that there exists a finite set of basic undirected graphs such that a graph is stable if and only if it does not contain as a minor any of the graphs in that set. These results guarantee decidability in polynomial time, however a constructive proof is not presented.

Our results. In this paper we consider the computational complexity of deciding universal stability of directed and undirected graphs. We consider different restrictions on the type of paths that the packets can follow, i.e., on the packet trajectory. We would like to highlight the importance of specifying the type of paths since, for each category, the characterization of universal stability is different. For each considered case we obtain a characterization in terms of forbidden subgraphs and provide an explicit polynomial time algorithm for deciding the property of stability. Concerning directed graphs, and under the assumption that packets follow simple paths, we obtain a characterization of universal stability that disproves the characterization in [15] under the same assumption (which was presented in terms of the forbidden minors given in Figure 1).

An interesting question concerning stability is that of deciding the stability of a concrete network under a fixed protocol. Using the fact that all the instability results obtained in

[1] The diameter of a graph is the length of the longest path in it that does not pass twice over the same edge.

this paper apply to networks under the NTG-LIS protocol, we can show that the problem of deciding whether a network is stable under the NTG-LIS protocol can also be solved in polynomial time.

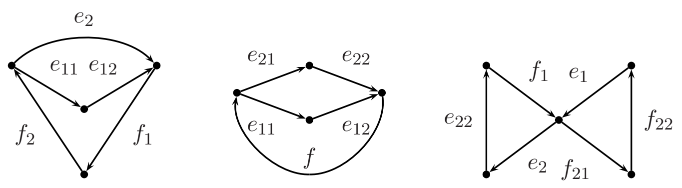


Figure 1: Forbidden minors proposed in [15] for simple-path universal stability of digraphs.

Organization. Section 2 sets out the definitions of all models considered in this work and some preliminary results. In Section 3, universal stability of some particular directed and undirected graphs is shown. Section 4 contains the basic results on instability of directed and undirected graphs. Section 5 details the characterizations for universal stability in terms of forbidden subgraphs, and Section 6 provides alternative characterization in terms of graph properties together with the algorithms that check them. Finally section 7, shows the polynomial time decidability of checking for stability under the NTG-LIS protocol. For the sake of readability several technical proofs are provided in the appendix.

2 Preliminaries

All the graphs in this paper may have multiple edges but no loops. Multiple edges share the same pair of different endpoints, while the endpoints of a loop are the same vertex. We will use the term *digraph* to refer to a directed graph and simply *graph* for an undirected graph. For a digraph we will use the term *arc* instead of edge. Given a graph G, G^d denotes the digraph formed from the same vertex set as G but replacing every edge $\{u, v\}$ in G with the two arcs (u, v) and (v, u).

Two edges are *incident* if they share at least one vertex. A *walk* is an alternating sequence of vertices and edges (respectively, arcs) in the form $v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n$, where for each i, $1 \leq i \leq n$, $e_i = \{v_{i-1}, v_i\}$ (respectively, $e_i = (v_{i-1}, v_i)$) and each v_j is a vertex. A *path* is a walk in which all the edges (respectively, arcs) are different. A *simple path* is a walk in which all the vertices, and thus necessarily all the edges (respectively, arcs) are different. A walk is *closed* if $v_0 = v_n$. A closed walk is a *n-cycle* provided its n vertices are distinct.

In this work we adopt, without lost of generality, the leaky-bucket definition of adversary which is defined by two parameters (r, b), where $b \geq 0$ and $0 < r \leq 1$. Rosén [20] showed that adversaries in the windowed and in the leaky-bucket models have the same power provided that $r < 1$ and under the assumption that the system has an empty starting configuration. The equivalence requires only a change on the parameters of the adversary, not on the sequence of packet trajectories, therefore it provides a valid equivalence for all the subclasses of adversaries proposed in this paper.

When in addition to a (r, b)-adversary $\mathcal{A}$, we are given an initial configuration C, we can define a new (r, b')-adversary $\mathcal{A}$' as follows:

> *Let $b' = b + |C|$; then $\mathcal{A}'$ injects all the packets in C at time step 1, and at any other time step $t > 1$, the same set of packets that $\mathcal{A}$ would inject at time $t - 1$.*

Both the system with empty initial configuration and the system with non-empty initial configuration behave alike. Hence, the equivalence shown by Rosén between windowed and leaky-bucket models with empty initial configuration also applies to the cases in which initial configuration is not empty. Since only the parameters of the adversary have to be changed, this remark is also valid for any system with different restrictions on the packet trajectory. Notice that this is a result stronger than the analogous result given in [4] since here the graph does not need to be changed (if the graph topology is changed then results on stability of networks cannot be translated from one model to the other).

All through the paper we will use the leaky-bucket adversarial model with empty initial configuration and $r < 1$. However, for the sake of simplicity, we will use non-empty initial configurations when describing adversaries provoking network instability. Following we establish the definitions of stability in the adversarial queueing model used in this work.

Definition 1. *Given a network G, a protocol P, and an adversary $\mathcal{A}$, we say that the system $(G, P, \mathcal{A})$ is* stable *if the number of packets in the system is always bounded.*

When the network is represented by a digraph G, we consider two classes of packet trajectories, thus arising two adversary classes: an *adversary* can use as packet trajectory any path in G, while a *simple-path adversary* can use as packet trajectory only simple paths in G. Accordingly we set the two definitions of stability under a protocol.

Definition 2. *Given a digraph G and a protocol P, we say that*

- *G is* stable *under protocol P (the pair (G, P) is stable) if for any adversary $\mathcal{A}$, the system $(G, P, \mathcal{A})$ is stable.*
- *G is* simple-path stable *under protocol P (the pair (G, P) is simple-path stable) if for any simple-path adversary $\mathcal{A}$, the system $(G, P, \mathcal{A})$ is stable.*

Similarly, we define universal stability of digraphs in the following form:

Definition 3. *A digraph G is* universally stable *if, for any protocol P, the pair (G, P) is* stable*. A digraph G is* simple-path universally stable *if, for any protocol P, the pair (G, P) is* simple-path stable*.*

Observe that any universally stable digraph is also simple-path universally stable, but the opposite, as we will see, is not true. The property of simple-path universal stability was shown in [15] to be maintained when acyclically joining simple-path universally stable digraphs. A closer inspection of the proof reveals its validity also for the other models for directed graphs considered in this work.

Lemma 1. *If digraphs G_1 and G_2 are (simple-path) universally stable, then so is any graph G formed by joining them with edges that go only from G_1 to G_2.*

As a consequence of the previous result we have,

Theorem 1. *A digraph G is (simple-path) universally stable if and only if all its strongly connected components are (simple-path) universally stable.*

When representing networks by undirected graphs, we consider three different packet trajectory restrictions: (i) Given a graph G, an *adversary* can use as packet trajectory any path in G^d; (ii) an *undirected-path adversary* can use as packet trajectory any path in G; and (iii) a *simple-path adversary* can use as packet trajectory only simple paths in G^d.

Observe that, in the first case, an edge can be used twice by the same packet provided it is traversed in opposite directions.[2] In the second model the same edge can be traversed only once. This corresponds with the model used by Andrews *et al.* in [4]. The third model does not allow a packet to pass twice through the same vertex. Notice that, in this latter model, a multiedge can only be traversed once and only in one direction. Observe that the condition *simple-path in* G is equivalent to *simple-path in* G^d, and so the fourth possible model is also already considered. As before, we set the definitions of stability under a protocol.

Definition 4. *Given a graph G and a protocol P, we say that*

- *G is* stable *under protocol P (the pair (G, P) is stable) if for any adversary $\mathcal{A}$, the system $(G, P, \mathcal{A})$ is stable.*
- *G is* undirected-path stable *under protocol P (the pair (G, P) is undirected-path stable) if for any undirected-path adversary $\mathcal{A}$, the system $(G, P, \mathcal{A})$ is stable.*
- *G is* simple-path stable *under protocol P (the pair (G, P) is simple-path stable) if for any simple-path adversary $\mathcal{A}$, the system $(G, P, \mathcal{A})$ is stable.*

It is straightforward to show that, if a pair (G, P) is stable then (G, P) is undirected-path stable, and if (G, P) is undirected-path stable then (G, P) is simple-path stable. As we will see later these inclusions are strict.
Now we can write the corresponding definitions of universal stability.

Definition 5. *A graph G is* universally stable *if for any protocol P the pair (G, P) is* stable*. A graph G is* undirected-path universally stable *if for any protocol P the pair (G, P) is* undirected-path stable*. A graph G is* simple-path universally stable *if for any protocol P the pair (G, P) is* simple-path stable*.*

The universal stability of graphs under the undirected-path model (as defined above) was addressed in [4] where it is proved to be closed under minors and therefore decidable in polynomial time, however no constructive algorithm is known. For the other models this question was unresolved. We show a constructive way of deciding the universal stability property also for the other models.

We can also state an equivalent result to that in Theorem 1 for the three forms of universal stability for undirected graphs presented here. Note that in the case of undirected graphs, each pair of connected components are independent, i.e. there is no edge connecting them.

Theorem 2. *A graph G is (simple-path | undirected-path) universally stable if and only if all its connected components are (simple-path | undirected-path) universally stable.*

Finally, let us remark that in any of the five packet trajectory models considered in this paper (i.e., two for digraphs and three for graphs), the instability of a sub(di)graph implies the instability of the whole (di)graph.

[2] But both directions have different queues and thus this model is different from Gamarnik proposal [14].

3 Some universally stable graphs and digraphs

In the following, we use standard graph terminology to denote the following graphs and digraphs: directed and undirected *trees*, the cycle on k vertices (i.e., k-*cycle*, with $k \geq 2$), the *directed cycle* on k vertices ($k \geq 2$), and *directed acyclic* graphs. For graphs and digraphs with multiple edges we define:

- an *unicyclic graph* as those graphs that only contain one cycle, see Figure 3(b.3)),
- a *multi-tree* as an undirected tree with multiple edges (see Figure 3(c.1)),
- a *decorated cycle* is obtained from a k-cycle with $k \geq 3$, and some multi-trees (see Figure 3(c.2)) after identifying one vertex from each tree with a vertex from the cycle,
- an *oriented multi-tree* as the directed version of an undirected tree in which each edge is substituted by two arcs (one in each direction), and it can also contain multiple arcs (see Figure 3(c.3)). Note that all the simple directed cycles have two vertices,
- a *decorated directed cycle* as the directed version of a k-cycle with $k \geq 3$, and some oriented multi-trees, after identifying one vertex from each 'tree' with a vertex from the cycle (see Figure 3(c.4)). Note that the obtained graph is strongly connected.

In this section, we show the universal stability of some of these graphs and digraphs according to the different packet trajectories considered. Concerning digraphs, previous results from [6] and [4] are re-written as Lemmas 2 and 3 according to the terminology introduced in this work.

Lemma 2. *All acyclic digraphs and the directed cycle on any number of vertices are universally stable.*

As we have commented before, universal stability of digraphs implies simple-path universal stability, therefore, acyclic digraphs and directed cycles are also simple-path universally stable. Concerning graphs, we know that

Lemma 3. *All trees and cycles on any number of vertices are undirected-path universally stable.*

Now we give another family of undirected-path universally stable graphs, thus extending the previous result.

Lemma 4. *All unicyclic graphs are undirected-path universally stable.*

Proof. Observe that, if G is an unicyclic graph, by removing the edges in the cycle it would result in a forest. We root each tree in this forest at the vertex that is common with the cycle, this gives an orientation *upwards* (i.e., towards the cycle) or *downwards* (i.e., away from the cycle) to every arc in G^d.

We classify the packets into three flow types. Flow type α is formed by those packets injected in an upward arc. Flow type β is formed by packets injected in a cycle arc, in fact we will consider this flow splitted into two sub-flows, depending on whether the initial arc follows the clockwise or the anti-clockwise cycle orientation. Finally, flow type γ is formed by the packets injected in a downward arc.

Observe that packets starting in a downward arc can only follow downward arcs, otherwise they will use twice an edge in G. Similarly, packets starting in the cycle cannot change their orientation and, either they die in the cycle, or leave the cycle using a downward arc. This provides a directed acyclic interaction of the three flow types. Upward edges can only carry packets from the α flow, as directed acyclic graphs are undirected-path stable, the corresponding queues will have bounded maximum size. The edges in the cycle can be considered as two directed cycles (one in each direction), in which no flow can be passed from one to the other, this edges can get packets from the α flow and from the β flow. In this situation we have a stable network (upward edges) entering in another stable network (one of the directed cycles) and, by Lemma 1, the corresponding queues will have bounded maximum size. For the downwards arcs we have the same situation, a flow coming from a stable network entering in another stable network, and thus the result follows. See Figure 2 for a graphical idea of the proof.

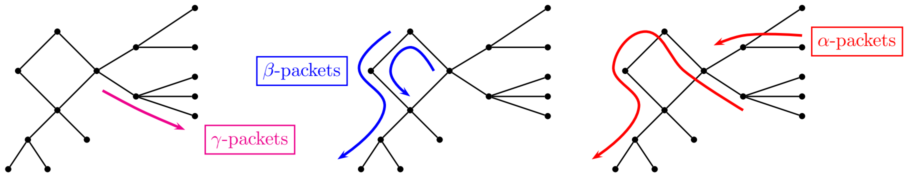


Figure 2: Flows uniclyclic graphs when considering undirected-path trajectories.

□

All the families of graphs which are undirected-path universally stable are also simple-path universally stable. However, we can enlarge the set of simple-path universally stable graphs by allowing multiple edges.

Lemma 5. *A multi-tree is simple-path universally stable.*

Proof. Observe that a packet can only use once, and only in one direction, one of the multiple edges connecting the same pair of vertices. The argument is similar to the one in Lemma 4, we root the tree in order to assign an upward/downward direction to the edges in G^d. The simple-path requirement forces that packets starting in a downward arc can not use an upward arc (even in the case that the arcs in G^d come from different edges in G), because in this case, a vertex would be visited twice. Thus, we can conclude the simple-path universally stability of a multi-tree. □

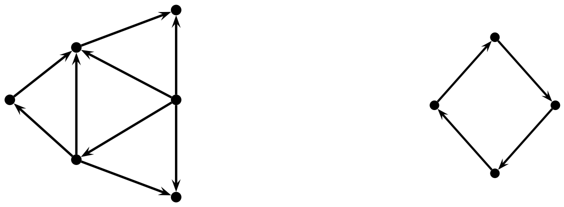
Lemma 6. *A decorated cycle is simple-path universally stable.*

Proof. The result follows from Lemmas 4 and 5 using analogous arguments as in Lemma 5. □

Lemma 7. *An oriented multitree and a decorated directed cycle are simple-path universally stable.*

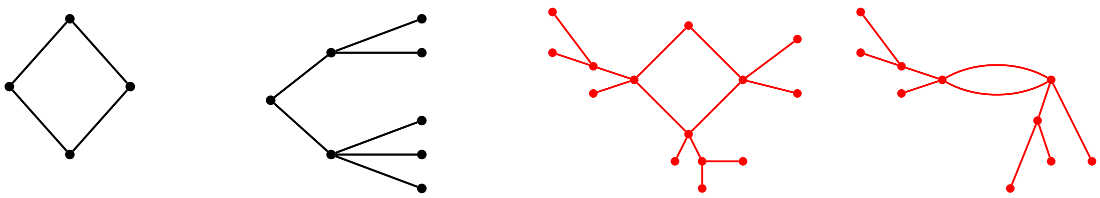
Proof. The result follows from Lemmas 5 and 7 using the arguments analogous to those in Lemmas 4 and 5. □

Figure 3 summarizes and classifies the families of graphs and digraphs that are universally stable according to the different packet trajectories considered.



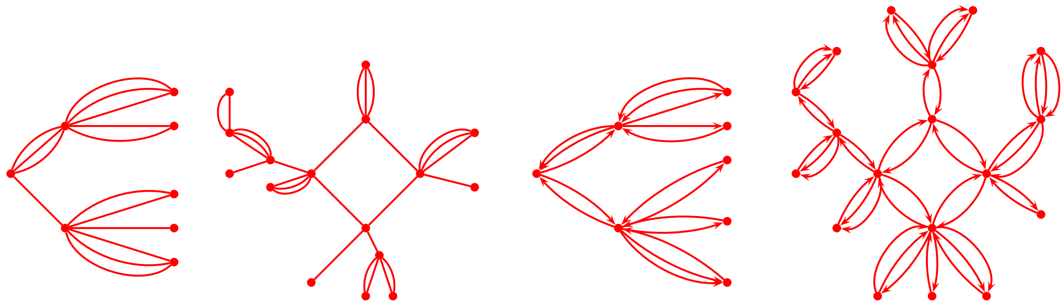
(a.1) A directed acyclic graph [6] (a.2) A directed cycle [6]

(a) Universally stable digraphs



(b.1) A cycle [4] (b.2) A tree [4] (b.3) Two unicyclic graphs

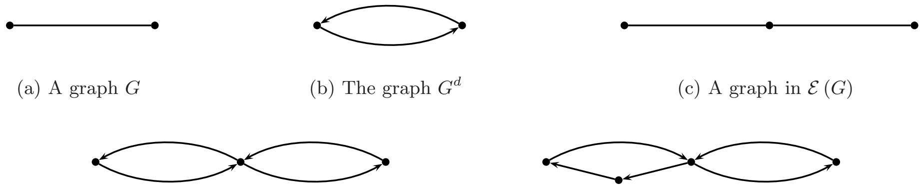
(b) Undirected-path universally stable graphs



(c.1) A multi-tree (c.2) A decorated cycle (c.3) An oriented multi-tree (c.4) A decorated directed cycle

(c) Simple-path universally stable graphs and digraphs

Figure 3: Examples of representatives of the families of universally stable graphs and digraphs in the different cases considered in this work.



(a) A graph G (b) The graph G^d (c) A graph in $\mathcal{E}(G)$

(d) A graph in $\mathcal{E}(G)^d$ and $\mathcal{E}(G^d)$ (e) A graph in $\mathcal{E}(G^d)$ but not in $\mathcal{E}(G)^d$

Figure 4: An example illustrating the differences between $\mathcal{E}(G)^d$ and $\mathcal{E}\left(G^d\right)$.

4 Some graphs and digraphs that are not universally stable

Once the families of universally stable graphs and digraphs are identified for each of the packet trajectories considered, we focus on detecting which are the simplest graphs and digraphs that are not stable. By iteratively applying subdivision operations to those simplest graphs and digraphs we will "extend" them and we will define families of graphs. Each family will characterize one of the cases of universal stability introduced in this work. We consider the following subdivision operations on graphs and digraphs:

- The *subdivision of an edge* $\{u, v\}$ in a graph G consists on the addition of a new vertex w and the replacement of $\{u, v\}$ by the two edges $\{u, w\}$ and $\{w, v\}$.
- The *subdivision of an arc* (u, v) in a digraph G consists on the addition of a new vertex w and the replacement of (u, v) by the two arcs (u, w) and (w, v).
- The *subdivision of a 2-cycle* (u, v), (v, u) in a digraph G consists on the addition of a new vertex w and the replacement of (u, v), (v, u) by the arcs (u, w), (w, u), (v, w) and (w, v).

Given a graph G, $\mathcal{E}(G)$ denotes the family of graphs formed by G and all the graphs obtained from G by successive edge subdivisions. Given a digraph G, $\mathcal{E}(G)$ denotes the family of digraphs formed by G and all the digraphs obtained from G by successive arc or 2-cycle subdivisions. Observe that, for a graph G, $\mathcal{E}(G)^d \subseteq \mathcal{E}\left(G^d\right)$, but it might be the case that $\mathcal{E}(G)^d \neq \mathcal{E}\left(G^d\right)$, see Figure 4 for an example.

In this section, we will prove instability results of networks under the NTG-LIS protocol. To keep lighter the notation, a path is specified by the sequence of its edges. Moreover, the names used to denote the graphs and their edges correspond to the ones depicted in Figures 5 and 6. First, we show that some simple graphs are not stable and second, we apply subdivision operations to them and show also their instability. Observe that, as we are using the NTG protocol, if the length of the path that a packet has to traverse is increased, e.g., when applying an extension, then the priority given to a packet at a given edge can change. Therefore, the instability of a (di)graph G might not be kept when extending the paths defined over G.

Theorem 3. *All the digraphs in $\mathcal{E}(U_1) \cup \mathcal{E}(U_2)$ are not stable under the* NTG-LIS *protocol.*

Proof. We sketch here the main lines and refer the reader to the Appendix A, where some of the most technical auxiliary results are proved. We start by showing that the pair $(U_1, \text{NTG-LIS})$ is not stable, to do so we provide an adversary and an initial configuration. The adversary works in rounds, at the end of the presented rounds the network has a configuration with the same type of packets as in the initial configuration but with an increased number of them. By playing repeatedly the set of rounds the system shows instability.

Initial configuration: At the beginning there are s packets that want to traverse edge f, half of them are of the form $(e_1\ f)$ and half of them are of the form $(e_2\ f)$. The adversary $\mathcal{A}$ will play injections in four rounds:

Round 1: for s steps, the adversary injects rs packets of the form $(f\ e_2)$. These injections get mixed with the initial packets at edge f, and are blocked there because the queueing protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of r^2s packets of the form $(f\ e_1)$ and r^2s packets of the form (e_2). Injections $(f\ e_1)$ are blocked by the packets $(f\ e_2)$ from the first round because they are at the same distance to their destination and the secondary protocol that we apply is LIS. The r^2s injections of the form (e_2) are also blocked.

Round 3: for the next r^2s steps, the adversary injects r^3s packets of the form (e_2) and r^3s of the form $(e_1\ f)$. The simple injections on e_2 will be blocked by the packets at e_2 from the previous round. The $(e_1\ f)$ injections get mixed with the packets of the form $(f\ e_1)$ from the previous round at edge f where the injections get blocked because their distance to destination is longer.

Round 4: for the next r^3s steps, the adversary injects r^4s packets of the form $(e_2\ f)$ and r^4s packets of the form (e_1). The simple injections at e_1 block the packets $(e_1\ f)$ from the previous round because their distance to destination is shorter. The $(e_2\ f)$ injections are blocked by the (e_2) packets from the previous round.

At the end there are $2r^4s$ packets queued at e_1 and e_2 that want to traverse edge f, r^4s are of the form $(e_1\ f)$ and r^4s are of the form $(e_2\ f)$. If cycles are allowed, the adversary $\mathcal{A}$ described above makes the network U_1 non-stable when $2r^4s > s$, i.e. at injection rate $r > 0.84089$.

The second step is to prove instability for the pair $(U_2, \text{NTG-LIS})$, as it is done in Lemma 9 at Appendix A. Once the two base digraphs are shown not to be stable, we have to show instability for any extension of them. Observe that only the extensions obtained by arc subdivision have to be considered since the extensions obtained by 2-cycle subdivision already contain U_2 as subgraph. Therefore, we have only to consider the two extensions depicted in Figures 7(a) and 7(b) whose corresponding instability results are given in Lemmas 10 and 11 (see Appendix A). □

We remark that, for digraphs, simple-path stability and stability are not equivalent. The pair $(U_1,\text{NTG-LIS})$ is not stable although U_1 is simple-path universally stable. The fact that U_1 is simple-path universally stable is easy to prove since the set of simple-paths in U_1 is the set $\{(e_1), (e_2), (f)\}$ and any adversary using this set of disjoint packet trajectories is equivalent to an adversary playing on a digraph with three unconnected arcs, which is acyclic and therefore universally stable. Similar reasoning can be applied to U_2. These considerations lead to digraphs S_1, S_2, S_3 and S_4 (see Figure 5(b)) as the smallest digraphs which are not simple-path universally stable.

Theorem 4. *All the digraphs in $\mathcal{E}(S_1) \cup \mathcal{E}(S_2) \cup \mathcal{E}(S_3) \cup \mathcal{E}(S_4)$ are not simple-path stable under the* NTG-LIS *protocol.*

Proof. The proof follows the same idea as for Theorem 3. We first prove that S_1, S_2, S_3, and S_4 are not simple-path stable (Lemmas 12, 13, 14, and 15). The restriction on considering simple-path trajectories justifies the need of having cycles of length at least 3 in a simple-path non-stable digraph. Extensions by 2-cycle subdivision are only applicable to S_4. Therefore, to finish the proof we show that the digraphs corresponding to the extensions depicted in Figures 8(a), 8(b), 8(c), and 8(d) are not simple-path stable (Lemmas 16, 17, 18, 19, and 11). See Appendix B for details. □

Theorem 5. *All the graphs in* $\mathcal{E}(H_1) \cup \mathcal{E}(H_2)$ *are not stable under the* NTG-LIS *protocol.*

Proof. Observe that H_1^d contains U_1 as a subgraph and that H_2^d is U_2. Furthermore, every graph in $\mathcal{E}(H_1) \cup \mathcal{E}(H_2)$ can be oriented to contain a graph in $\mathcal{E}(U_1) \cup \mathcal{E}(U_2)$ as a subgraph. Therefore, its instability under NTG-LIS follows from Theorem 3. □

Observe that, although the graph H_1 is not stable, it is undirected-path universally stable because the set of possible undirected-path packet trajectories is formed by non-overlapping paths with length one. Therefore, undirected-path stability and stability are non equivalent properties. A similar reasoning can be applied to H_2.

To characterize undirected-path stability we need to consider some larger graphs. Since the packet trajectories in undirected-path stability are defined by paths in the (undirected) graph, we fix an orientation to all the edges (with the exception of those that are graph separators, as those edges may have to be crossed in both directions, by different packets).

Theorem 6. *All the graphs in* $\mathcal{E}(F_1) \cup \mathcal{E}(F_2) \cup \mathcal{E}(F_3)$ *are not undirected-path stable under the* NTG-LIS *protocol.*

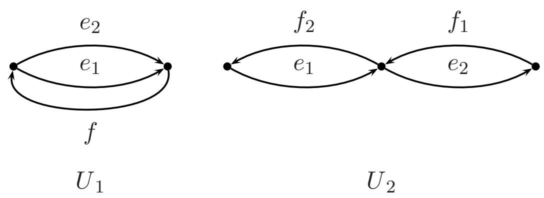
Proof. Let us prove first that the graphs F_1 and F_2 are not NTG-LIS undirected-path stable. Giving an adequate orientation to the edges in F_1 and F_2 we get the digraphs U_1 and U_2 and, as each arc correspond to a different edge, the instability follows. Maintaining these orientations, any graph in $G \in \mathcal{E}(F_1) \cup \mathcal{E}(F_2)$ can be identified with a digraph in $\mathcal{E}(U_1) \cup \mathcal{E}(U_2)$ and therefore G is not undirected-path stable under NTG-LIS.

For the graph F_3 we can fix an orientation for the edges forming the 2-cycles, but we have to keep both directions, as possible, for the middle edge. Although no packet can use both, different packets may traverse the edge in different directions. Starting with the proof of NTG-LIS instability for the graph U_2 and working in a similar way as in Lemmas 15 and 19 it follows that F_3 and any graph in $\mathcal{E}(F_3)$ is not undirected-path stable under NTG-LIS. □

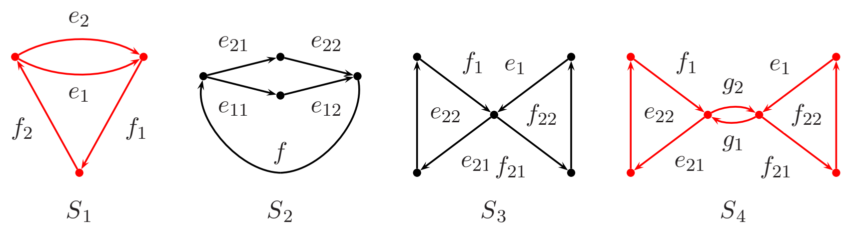
The graph F_1 is simple-path universally stable, but not undirected-path stable, therefore simple-path stability differs from undirected-path stability. Cycles of length 3 or longer are needed to obtain simple-path non-stable graphs, thus obtaining K_1, K_2 and K_3 as smallest graphs which are not simple-path stable.

Theorem 7. *All the graphs in* $\mathcal{E}(K_1) \cup \mathcal{E}(K_2) \cup \mathcal{E}(K_3)$ *are not simple-path stable under* NTG-LIS*.*

Proof. Observe that we can give an orientation to the edges in K_1 and K_2 to obtain the digraphs S_1 and S_3, which are not simple-path stable under NTG-LIS. For the graph K_3 we can orient the two cycles and maintain the two opposite arcs for the middle edge, obtaining the graph S_4. Theorem 4 gives the simple-path instability under NTG-LIS for any graph in $\mathcal{E}(K_1) \cup \mathcal{E}(K_2) \cup \mathcal{E}(K_3)$. □

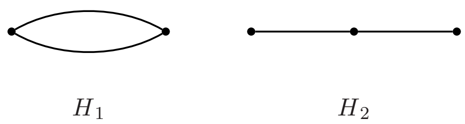


(a) no universally stable digraphs

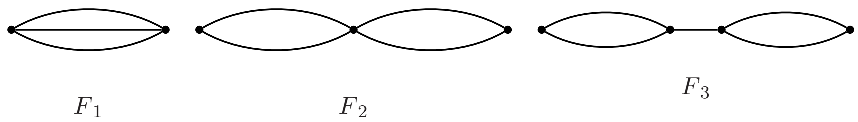


(b) no simple-path universally stable digraphs

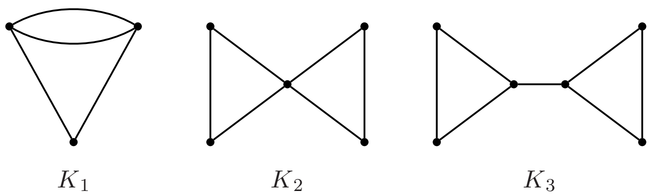
Figure 5: Some no universally stable digraphs



(a) no universally stable graphs



(b) no undirected-path universally stable graphs



(c) no simple-path universally stable graphs

Figure 6: Some no universally stable graphs

5 Characterizing universal stability

In this section, we provide characterizations for the universal stability property of digraphs and graphs for each of the five proposed types of packet trajectories. As before, the graph nomenclature corresponds to Figures 5 and 6.

5.1 Universal stability

Theorem 8. *A digraph is universally stable if and only if it does not contain as subgraphs any of the digraphs in* $\mathcal{E}(U_1) \cup \mathcal{E}(U_2)$.

Proof. The only if part follows from Theorem 3 and the fact that the instability of a subgraph implies the instability of the whole graph. If G does not contain as subgraph a digraph in $\mathcal{E}(U_1) \cup \mathcal{E}(U_2)$ then all its strongly connected component must consist of a simple cycle. Therefore, according with Lemma 2 and Theorem 1 we have that G is universally stable. □

Using the arguments in the previous proof, Theorem 8 gives the following property.

Corollary 1. *A strongly connected digraph G with n vertices is universally stable iff G is the directed cycle on n vertices.*

For the case of universal stability of graphs, the basic set of forbidden subgraphs is given in Figure 6(a).

Theorem 9. *A graph is universally stable if and only if it does not contain as subgraphs any of the graphs in* $\mathcal{E}(H_1) \cup \mathcal{E}(H_2)$.

Proof. The only if part follows from Theorem 5. If a graph G does not contain as subgraph any of the graphs in $\mathcal{E}(H_1) \cup \mathcal{E}(H_2)$, then G has no incident edges, so it is universally stable. □

The corresponding graph property is the following,

Corollary 2. *A graph G is universally stable iff G has no incident edges.*

5.2 Undirected-path universal stability

For the case of undirected-path universal stability of graphs, the basic set of forbidden subgraphs is given in Figure 6(b).

Theorem 10. *A graph G is undirected-path universally stable iff G does not contain as a subgraph any of the graphs in* $\mathcal{E}(F_1) \cup \mathcal{E}(F_2) \cup \mathcal{E}(F_3)$

Proof. The only if part follows from Theorem 6. Assume now that G does not contain as a subgraph any of the graphs in $\mathcal{E}(F_1) \cup \mathcal{E}(F_2) \cup \mathcal{E}(F_3)$. In this case G does not contain any edge with multiplicity 3 and all the connected components of G contain at most one cycle. Therefore, all the connected components of G are undirected-path universally stable by Lemma 3. □

The corresponding property for undirected-path universal stability of graphs is the following,

Corollary 3. *A connected graph G is undirected-path universally stable if and only G is a subgraph of a unicyclic graph.*

5.3 Simple-path universal stability

For the case of simple-path universal stability of graphs, the basic set of forbidden subgraphs is given in Figure 6(c).

Theorem 11. *A graph G is simple-path universally stable iff G does not contain as a subgraph any of the graphs in $\mathcal{E}(K_1) \cup \mathcal{E}(K_2) \cup \mathcal{E}(K_3)$.*

Proof. The only if part follows from Theorem 7. When G does not contain as subgraphs any of the graphs in $\mathcal{E}(K_1) \cup \mathcal{E}(K_2) \cup \mathcal{E}(K_3)$, all the connected components of G, can have at most one cycle with more than two vertices, so they must be subgraphs of a decorated cycle graph. Therefore, using Lemma 6, G is simple-path universally stable. □

Corollary 4. *A connected graph G is simple-path universally stable if and only if G is a subgraph of a decorated cycle graph.*

The case of simple-path universal stability for digraphs is missing to complete the characterization. In this case the basic set of forbidden subgraphs is given in Figure 5(b).

Theorem 12. *A digraph G is simple-path universally stable iff G does not contain as a subgraph any of the graphs in $\mathcal{E}(S_1) \cup \mathcal{E}(S_2) \cup \mathcal{E}(S_3) \cup \mathcal{E}(S_4)$.*

Proof. As before, the only if part follows from Theorem 4. When G excludes the family of forbidden subgraphs, each strongly connected component of G must contain at most one directed cycle with more than two vertices. Therefore all the strongly connected components are subgraphs of a decorated directed cycle. Therefore, by Lemma 7, all the strongly connected components are simple-path universally stable, and by Theorem 1, G is simple-path universally stable. □

Corollary 5. *A strongly connected digraph G is simple-path universally stable iff G is a subgraph of a decorated directed cycle graph.*

Goel proposed in [15] a set of forbidden digraphs to characterize what we have distinguished here as simple-path universal stability (see Figure 1). The only digraph transformation proposed there consisted in replacing arcs by disjoint directed paths (see Corollary 2.5 and 2.7 in [15]). We have shown that the graph S_1 in Figure 5(b) is not simple-path stable under the NTG-LIS protocol. Hence, our graph S_1 is a minor of the first minor proposed in [15]. Moreover, we have also shown that the digraph S_4 is not simple-path stable, even that S_4 does not contain as minor any of the forbidden minors in the characterization in [15]. Therefore, according to Goel result, it must be universally stable; however, we have shown that this is not the case. These considerations disprove Goel characterization and establish the set $\{S_1, S_2, S_3, S_4\}$ (see Figure 5(b)) as the right set of forbidden subgraphs characterizing the property of simple-path stability for digraphs.

6 Deciding universal stability

In this section we show that the five cases of universal stability presented in this work can be decided in polynomial time. For undirected graphs we could use the polynomial-time algorithm for checking subgraph homeomorphism of the corresponding forbidden subgraphs given in [19], however our algorithms are much simpler. Notice that, for directed graphs, checking subdigraph homeomorphism to a fixed digraph is NP-complete (in particular this is the case for the digraphs S_1, S_2, and S_3, see [12]). However the combination of several digraphs and the properties outlined in Corollaries 1, 2, 3, 4 and 5 are, as we will see, easier to test.

Theorem 13. *The universal stability of a given graph or digraph can be decided in polynomial time.*

Proof. According to Corollary 2, to decide universal stability for graphs we only have to check if the graph has two incident edges. To check universal stability for digraphs, following Corollary 1, we have to compute the strongly connected components of the graph and then check whether all of them are just one directed cycle. Both tests can be performed in polynomial time. □

For the remaining adversarial models we need first to characterize the graphs that are subgraphs of an unicyclic graph.

Lemma 8. *A connected graph G with m edges and without multiple edges is a subgraph of an unicyclic graph if and only if any spanning tree of G has m or $m-1$ edges.*

According with the previous lemma, Algorithm 1 checks in polynomial time whether a connected graph G, without multiple edges, is a subgraph of an unicyclic graph. The algorithms to decide universal stability combine this checking with some additional testing for the multi-edges.

Theorem 14. *The undirected-path universal stability of a given graph can be decided in polynomial time.*

Proof. Algorithm 2 checks the undirected-path universal stability of a given connected graph G according to Corollary 3. The total execution time is polynomial. Thus the result follows combining this algorithm with the computation the connected components of the given graph. □

Theorem 15. *The simple-path universal stability of a given graph can be decided in polynomial time.*

Proof. Algorithm 3 checks the simple-path universal stability of a given connected graph G according to Corollary 4. The total execution time is polynomial. Thus the result follows combining this algorithm with the computation the connected components of the given graph. □

Theorem 16. *The simple-path universal stability of a given digraph can be decided in polynomial time.*

Proof. Algorithm 4 checks the simple-path universal stability of a given strongly connected digraph G according to Corollary 5. The total execution time is polynomial. Thus the result follows combining this algorithm with the computation the strongly connected components of the given digraph. □

All the algorithms presented in this section run in polynomial time. Note that the most expensive operations are the computation of the strongly connected components of a digraph, the connected components of a graph, and the computation of a spanning tree of a connected graph.

Algorithm 1 : Subgraph of an unicyclic graph

```
INPUT: A connected graph G
Compute a spanning tree of G
if there are more than one edge left then
   return no
else
   return yes
end if
```

Algorithm 2 : Graph undirected-path stability

```
INPUT: A connected graph G
if some edge has multiplicity 3 then
   return no
else
   compute the connected components of G
   if a connected component has two edges with multiplicity 2 then
      return no
   end if
   || Now all the connected components have at most one edge with multiplicity 2
   if there is a connected component of G that is not a subgraph of an unicyclic graph
   then
      return no
   else
      return yes
   end if
end if
```

Algorithm 3 : Graph simple-path stability

```
INPUT: A connected graph G
Let G′ be the graph obtained from G by setting all edge multiplicities to one
Compute the connected components of G′
if there is a connected component H of G′ s.t. H is not a subgraph of an unicyclic graph
or H is an unicyclic graph having k ≥ 3 vertices in the cycle and with some cycle edge
that had in G multiplicity greater than 1 then
   return no
else
   return yes
end if
```

Algorithm 4 : Digraph simple-path stability

INPUT: A strongly connected digraph G
if G does not have a directed k-cycle with $k \geq 3$ **then**
  return yes
else
  compute a directed cycle $C = (v_k, e_1, v_1, \ldots, v_{k-1} e_k, v_k)$ in G ($k \geq 3$)
  if any of the cycle arcs has multiplicity greater than one **then**
    return no
  end if
  Let $e'_i = (v_i, v_{i-1})$ be the arc opposite to $e_i = (v_{i-1}, v_i)$, $1 \leq i \leq k$
  if all the arcs e'_i are present in G and some of them have multiplicity greater than one **then**
    return no
  end if
  Let G' be the digraph obtained by setting the multiplicity of all arcs in G to one and by removing the arcs in C and all the opposite arcs e'_i (if any).
  if there are two vertices of the cycle C connected by a directed path in G' **then**
    return no
  else
    Compute the strongly connected components of G'
    if a strongly connected component of G' contains a directed k-cycle with $k \geq 3$ **then**
      return no
    else
      return yes
    end if
  end if
end if

7 Stability under NTG-LIS

An interesting question concerning stability is that of deciding the stability of a concrete network under a fixed protocol. Since all the instability results in this work hold for the NTG-LIS protocol, we can conclude that testing universal stability is equivalent to testing stability under NTG-LIS. Moreover, deciding whether a network is stable under the NTG-LIS protocol can also be solved in polynomial time. The equivalence holds for all the cases of universal stability considered in this work, as we state in the following theorem:

Theorem 17. *The following equivalences hold:*

- *A digraph G is stable under* NTG-LIS *iff G is universally stable.*
- *A digraph G is simple-path stable under* NTG-LIS *iff G is simple-path universally stable.*
- *A graph G is stable under* NTG-LIS *iff G is universally stable.*
- *A graph G is undirected-path stable under* NTG-LIS *iff the graph G is undirected-path universally stable.*
- *A graph G is simple-path stable under* NTG-LIS *iff G is simple-path universally stable.*

A similar result was obtained in [2] for the universal stability of digraphs under the FFS protocol, for which the same characterization (i.e., in terms of the family of graphs generated from U_1 and U_2, see Figure 5(a)) is obtained. To the best of our knowledge, the complexity of deciding stability under other non universally stable protocols is still open, in particular for FIFO and LIFO. Even that much effort has been devoted to the study of the FIFO protocol, it is still not known whether deciding stability under FIFO is polynomial time decidable. The drawback, with regard to the characterization of universal stability, is whether the pair (U_1,FIFO) is stable, as the pair (U_2, FIFO) was shown not to be stable in [1].

References

[1] C. Àlvarez, M. Blesa, J. Diaz, A. Fernández, and M. Serna. Adversarial models for priority-based networks. In B. Rovan and P. Vojtas, editors, *28th International Symposium on Mathematical Foundations of Computer Science* (MFCS'03), volume 2747 of *Lecture Notes in Computer Science*, pages 142–151, Bratislava, Slovakia, August 25-29 2003. Springer-Verlag.

[2] C. Àlvarez, M. Blesa, J. Diaz, A. Fernández, and M. Serna. The complexity of deciding stability under FFS in the adversarial model. Technical Report LSI-03-16-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2003. Submitted.

[3] M. Andrews. Inestability of FIFO in session–oriented networks. In *11th. Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'00), pages 440–447, San Francisco, USA, January 2000.

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

[5] R. Bhattacharjee and A. Goel. Instability of FIFO at arbitrarily low rates in the adversarial queueing model. In *44th Annual IEEE Symposium on Foundations of Computer Science* (FOCS'03), pages 160–167, Cambridge, USA, October 11-14 2003.

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

[7] M. Bramson. Inestability of FIFO queueing networks. *Annals of Applied Probability*, 4(2):414–431, 1994.

[8] R. Cruz. A calculus for network delay. Part I: Network elements in isolation. In *IEEE Transactions on Information Theory*, volume 37, pages 114–131, 1991.

[9] R. Cruz. A calculus for network delay. Part II: Network analysis. In *IEEE Transactions on Information Theory*, volume 37, pages 132–141, 1991.

[10] J. Diaz, D. Koukopoulos, S. Nikoletseas, M. Serna, P. Spirakis, and D. Thilikós. Stability and non-Stability of the FIFO Protocol. In *13th annual ACM Symposium on Parallel Algorithms and Architectures* (SPAA'01), pages 48–52, Crete Island, Greece, July 2001.

[11] J. Echagüe, V. Cholvi, and A. Fernández. Universal stability results for low rate adversaries in packet switched networks. *IEEE Communication Letters*, 2003. In press.

[12] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[13] D. Gamarnik. Stability of adversarial queues via fluid models. In *39th. Annual IEEE Symposium on Foundations of Computer Science* (FOCS'98), pages 60–70, Palo Alto, USA, November 1998.

[14] D. Gamarnik. Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks. *SIAM Journal on Computing*, 32(2):371–385, 2003.

[15] A. Goel. Stability of networks and protocols in the adversarial queueing model for packet routing. *Networks*, 37(4):219–224, 2001.

[16] D. Koukopoulos, M. Mavronicolas, and P. Spirakis. FIFO is unstable at arbitrarily low rates. *Electronic Colloquium on Computational Complexity*, 10(16), 2003.

[17] D. Koukopoulos, S. Nikoletseas, and P. Spirakis. The range of stability for heterogeneous and FIFO queueing networks. *Electronic Colloquium on Computational Complexity*, (TR01-099), 2001.

[18] Z. Lotker, B. Patt-Shamir, and A. Rosén. New stability results for adversarial queuing. In *14th ACM Symposium on Parallel Algorithms and Architectures*, pages 175–182, Winnipeg, Canada, August 2002. ACM Press New York, USA.

[19] N. Robertson and P. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory. Series B*, 63(1):65–110, 1995.

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

[21] Z.-L. Zhang, Z. Duan, and Y. Hou. Fundamental trade-offs in aggregate packet scheduling. In *IEEE 9th International Conference on Network Protocols* (ICNP'01), pages 129–137, Riverside, USA, November 2001.

Appendix

All the instability proofs 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 one step to the next (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 the configuration at the beginning of such inductive step (in terms of the type of packets and their location), but with an increased number of packets. In these appendixes we only reproduce the inductive step.

A Proof of Theorem 3

Lemma 9. *The pair* $(U_2, \text{NTG-LIS})$ *is not stable.*

Proof. At the beginning there are s packets requiring to traverse edge f_2. Half of them are of the form $(e_1\ f_2)$ and half of the form $(e_2\ f_1\ f_2)$. Then the adversary will play injections in four rounds.

Round 1: for s steps, the adversary injects rs packets of the form $(f_2\ e_1)$. These injections get mixed with the initial packets at edge f_2, and are blocked there because the queueing protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of r^2s packets of the form $(f_2\ e_1\ e_2)$ that are blocked by the remaining packets at f_2 from the previous round because the protocol is NTG.

Round 3: for the next r^2s steps, the adversary injects r^3s packets of the form (e_2) and r^3s of the form $(e_1\ f_2)$. A total number of r^3s packets of the from (e_2) will remain at e_2. Injections of the form $(e_1\ f_2)$ will meet packets from the previous round at edge e_1 and, because the secondary protocol is LIS, they will be all blocked at this edge.

Round 4: for the next r^3s steps, the adversary injects r^4s packets of the form $(e_2\ f_1\ f_2)$ and r^4s packets of the form (e_1). The simple injections at e_1 block the packets $(e_1\ f_2)$ from the previous round because their distance to destination is shorter. The $(e_2\ f_1\ f_2)$ injections are blocked by the (e_2) packets from the previous round.

At the end of the forth round there are r^4s packets of the form $(e_2\ f_1\ f_2)$ and r^4s packets of the form $(e_1\ f_2)$. If cycles are allowed, the adversary described above makes the network U_2 non-stable when $2r^4s > s$, i.e. at injection rate $r > 0.84089$. □

Lemma 10. *Any graph in* $\mathcal{E}(U_1)$ *is not stable for* NTG-LIS.

Proof. Let G be the graph in $\mathcal{E}(U_1)$ described at Figure 7(a). This graph is formed from $\mathcal{S}_1$ by extending the edges by paths. Let us denote by p_n, p_m and p_d the path replacing the edges. Let us assume that $n \geq m - 1$. The adversary operates in four rounds. Initially, there are s packets wanting to traverse p_d and they are distributed in the following way: x packets of the form $(p_n\ e_1\ p_d)$ and y packets of the form $(p_m\ p_d)$.

Round 1: for s steps, the adversary injects rs packets of the form $(p_d\ p_m)$. These injections are blocked by the initial packets because the queueing protocol is NTG, but rm of them are lost. We consider a big enough s to guarantee that a continuous flow arrives to the first edge of p_d after the arrival of the first packet.

Round 2: for the next $rs - rm$ steps, the adversary injects a set of $r^2s - r^2m$ packets of the form $(p_d\ p_n\ e_1)$ and $r^2s - r^2m$ packets of the form (p_m). These injections are blocked by the remaining packets from the previous round because the queueing protocol is NTG (or because of the LIS protocol if $n = m - 1$). Note that rd injections of the form (p_m) will be lost.

Round 3: for the next $r^2s - r^2m$ steps, the adversary injects $r^3s - r^3m$ packets of the form (p_m) that will be all blocked by packets from the previous round. The adversary also injects $r^3s - r^3m$ packets of the form $(p_n\ e_1\ p_d)$ that will collapse with packets from the previous round. These injections will be blocked there because their distance to destination is longer at that point, but rd of them will be lost. At the end of this round there are $r^3s - r^3m - r^2d$ packets of the form (p_m) and $r^3s - r^3m - rd$ packets of the form $(p_n\ e_1\ p_d)$.

Round 4: for the next $r^3s - r^3m - r^2d$ steps, the adversary injects $r^4s - r^4m - r^3d$ packets of the form $(p_n\ e_1)$ and $r^4s - r^4m - r^3d$ packets of the form $(p_m\ p_d)$. The injections on p_n will maintain blocked some packets of the previous round and, the injections on p_m will be blocked by the remaining packets from the previous round.

At the end of the forth round, there are $r^4s - r^4m + r^3d - r^2d$ packets of the form $(p_n\ e_1\ p_d)$ and $r^4s - r^4m - r^3d$ packets of the form $(p_m\ p_d)$. The adversary described above uses packets describing cycles and makes the graph G described above non-stable when $2r^4s - 2r^4m - r^2d > s$. Note that $C = 2r^4m + r^2d > 2m + d$ and for an big enough s, an injection rate r can be found such that $2r^4s - C > s$ holds. □

Lemma 11. *Any graph in $\mathcal{E}(U_2)$ is not stable for* NTG-LIS.

Proof. Let G be a graph in $\mathcal{E}(U_2)$ described in Figure 7(b). This graph is formed from $\mathcal{S}_2$ by extending its cycles with paths. We call p_k and p_l the additional edges in the two sides, note that k, l or both may be zero. The adversary operates in four rounds. Initially, there are $\frac{s}{2}$ packets of the form $(e_1\ f_2)$ and $\frac{s}{2}$ packets of the form $(e_2\ p_l\ f_1\ f_2)$.

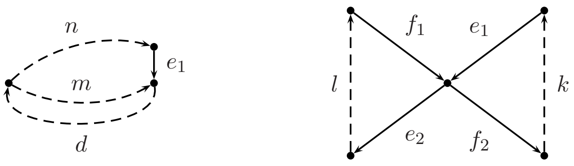
Round 1: for s steps, the adversary injects rs packets of the form $(f_2\ p_k\ e_1)$. These injections are blocked by the initial packets because the queueing protocol is NTG. Note that rl of such new packets are lost.

Round 2: for the next $rs - rl$ steps, the adversary injects a set of $r^2s - r^2l$ packets of the form $(f_2\ p_k\ e_1\ e_2)$. These injections are blocked by the remaining packets from the previous round because the queueing protocol is NTG.

Round 3: for the next $r^2s - r^2l$ steps, the adversary injects $r^3s - r^3l$ packets of the form (e_2) and $r^3s - r^3l$ of the form $(e_1\ f_2)$. These injections will be blocked by packets from the previous round when collapsing at edges e_2 and e_1 respectively, but rk of them will be lost.

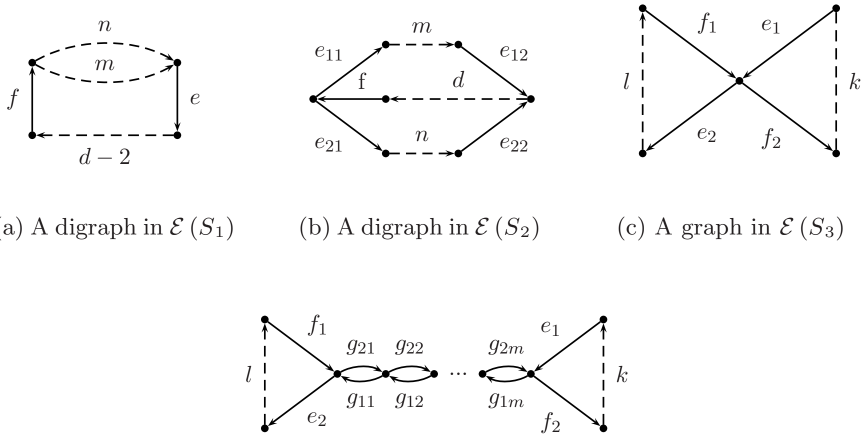
Round 4: for the next $r^3s - r^3l - rk$ steps, the adversary injects $r^4s - r^4l - r^2k$ packets of the form (e_1) that will block packets from the previous round at e_1 because they are closer to their destination. The adversary also injects $r^4s - r^4l - r^2k$ packets of the form $(e_2\ p_l\ f_1\ f_2)$ that will be blocked by the (e_2) remaining packets from the previous round.

At the end of the forth round, there are $r^4s-r^4l-r^2k$ packets of the form $(e_2\ p_l\ f_1\ f_2)$ and $r^4s-r^4l - r^2k$ packets of the form $(e_1\ f_2)$. The adversary described above uses packets describing cycles and makes the graph G described above non-stable when $2(r^4s - r^4l - r^2k) > s$. Note that $C = 2(l + k) > 2(r^4l + r^2k)$, and for a big enough s, an injection rate r can be found such that $2r^4s - C > s$ holds. □



(a) A digraph in $\mathcal{E}(U_1)$ (b) A digraph in $\mathcal{E}(U_2)$

Figure 7: Family of digraphs formed by extensions of U_1 and U_2.



(a) A digraph in $\mathcal{E}(S_1)$ (b) A digraph in $\mathcal{E}(S_2)$ (c) A graph in $\mathcal{E}(S_3)$

(d) A digraph in $\mathcal{E}(S_4)$

Figure 8: Family of digraphs formed by extensions of S_1, S_2, S_3 and S_4.

B Proof of Theorem 4

Lemma 12. *The pair* $(S_1, \text{NTG-LIS})$ *is not simple-path stable.*

Proof. The adversary operates in five rounds. Initially, there are $\frac{s}{2}$ packets of the form $(e_1\ f_1)$ and $\frac{s}{2}$ of the form $(e_2\ f_1)$.

Round 1: for s steps, the adversary injects rs packets of the form $(f_1\ f_2)$. These injections get mixed with the initial packets at f_1 and get blocked there because the protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of r^2s packets of the form $(f_2\ e_2)$ that are blocked at f_2 by the packets from the first round because, at that point, they are at longer distance to their destination.

Round 3: for the next r^2s steps, the adversary injects r^3s packets of the form (e_2) and r^3s of the form $(f_2\ e_1)$. All these injections are blocked by the remaining packets from the previous round because the secondary protocol is LIS.

Round 4: for the next r^3s steps, the adversary injects r^4s packets of the form (e_1) and r^4s packets of the form $(e_2\ f_1)$. A total number of r^4s simple packets will remain at e_1. The injections of the form (e_2 f_1) are blocked at e_2 by packets from the previous round because distance to their destination is longer.

Round 5: for the next $r^4 s$ steps, the adversary injects $r^5 s$ packets of the form (e_1 f_1) and $r^5 s$ packets of the form (e_2). The simple packets queued at e_1 from the previous round will block the injections introduced at this edge. The simple injections at e_2 will block the packets of the form (e_2 f_1) from the previous round.

At the end of the fifth round, there are $r^5 s$ packets of the form (e_2 f_1) and $r^5 s$ packets of the form (e_1 f_1). The adversary described above only uses packets describing simple paths and makes network U_1 non-stable when $2r^5 s > s$, i.e. at injection rate $r > 0.87055$. □

Lemma 13. *The pair* (S_2, NTG-LIS) *is not simple-path stable.*

Proof. The adversary operates in four rounds in this case. Initially, there are $\frac{s}{2}$ packets of the form (e_{12} f) and $\frac{s}{2}$ of the form (e_{22} f).

Round 1: for s steps, the adversary injects rs packets of the form (f e_{21}). These injections get mixed with the initial packets, and are blocked in edge f because the protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of $r^2 s$ packets of the form (f e_{11}) and $r^2 s$ packets of the form (e_{21} e_{22}). The injections made on f are blocked by old packets because in this situation the secondary protocol LIS is applied. When the (f e_{21}) packets from the previous round reach edge e_{21}, they block the injections made on e_{21}. So at this round all the injections introduced are blocked.

Round 3: for the next $r^2 s$ steps, the adversary injects $r^3 s$ packets of the form (e_{22} f) and $r^3 s$ of the form (e_{11} e_{12}). The injections on e_{22} are blocked by packets of the form (e_{21} e_{22}) from the previous round. The injections introduced on e_{11} are blocked by packets from the previous round at e_{11} because their destination is farther.

Round 4: for the next $r^3 s$ steps, the adversary injects $r^4 s$ packets of the form (e_{12} f) and $r^4 s$ packets of the form (e_{22}). Injections on e_{12} are blocked by old packets, and injections on e_{22} block old packet of the form (e_{22} f).

At the end of the forth round, there are $2r^4 s$ packets queued at e_{12} and e_{22} that want to traverse edge f; $r^4 s$ are of the form (e_{12} f) and $r^4 s$ are of the form (e_{22} f). The adversary described above only uses simple paths and makes network S_2 non-stable when $2r^4 s > s$, i.e., at injection rate $r > 0.84089$. □

Lemma 14. *The pair* (S_3, NTG-LIS) *is not simple-path stable.*

Proof. The adversary operates in four rounds. Initially, there are $\frac{s}{2}$ packets of the form (e_1 f_{21}) and $\frac{s}{2}$ of the form (e_{22} f_1 f_{21}).

Round 1: for s steps, the adversary injects rs packets of the form (f_{21} f_{22}). These injections get mixed with the initial packets, and get blocked in edge f_{21} because the protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of $r^2 s$ packets of the form (f_{22} e_1 e_{21}) that are blocked at f_{22} by the packets from the first round because, at that point, they are at longer distance to their destination.

Round 3: for the next $r^2 s$ steps, the adversary injects $r^3 s$ packets of the form (e_{21} e_{22}) and $r^3 s$ of the form (e_1 f_{21}). The injections introduced at e_1 reach the packets from the previous round at this edge, but they are blocked there because the secondary protocol is LIS. Old packets block the injections introduced at their last edge e_{21}.

Round 4: for the next $r^3 s$ steps, the adversary injects $r^4 s$ packets of the form (e_{22} f_1 f_{21}) and $r^4 s$ packets of the form (e_1). The simple injections on e_1 will block the packets queued at this edge from the previous round. Injections of the form (e_{22} f_1 f_{21}) are blocked at e_{22} by packets from the previous round because distance to their destination is longer.

At the end of the forth round, there are $r^4 s$ packets of the form (e_{22} f_1 f_{21}) queued at e_{22}, and $r^4 s$ of the form (e_1 f_{21}) queued at e_1. The adversary described above only uses simple paths and makes network S_3 non-stable when $2r^4 s > s$, i.e., at rate $r > 0.84089$. □

Lemma 15. *The pair* (S_4, NTG-LIS) *is not simple-path stable.*

Proof. The adversary operates in four rounds. Initially, there are $\frac{s}{2}$ packets of the form (e_1 f_{21}) and $\frac{s}{2}$ of the form (e_{22} f_1 g_2 f_{21}).

Round 1: for s steps, the adversary injects rs packets of the form (f_{21} f_{22}). These injections get mixed with the initial packets, and are blocked in edge f_{21} because the protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of $r^2 s$ packets of the form (f_{22} e_1 g_1) that are blocked at f_{22} by the packets from the first round because, at that point, they are at longer distance to their destination.

Round 3: for the next $r^2 s$ steps, the adversary injects $r^3 s$ packets of the form (g_1 e_{21} e_{22}) and $r^3 s$ of the form (e_1 f_{21}). The injections introduced at e_1 reach the packets from the previous round at this edge, but they are blocked there because the secondary protocol is LIS. When the old packets arrive at their last edge g_1, they block the injections introduced at that edge. So, at this round, all the injections are blocked.

Round 4: for the next $r^3 s$ steps, the adversary injects $r^4 s$ packets of the form (e_{22} f_1 g_2 f_{21}) and $r^4 s$ packets of the form (e_1). The simple injections on e_1 will block the packets queued at this edge from the previous round. Injections of the form (e_{22} f_1 g_2 f_{21}) are blocked at e_{22} by packets from the previous round because distance to their destination is longer.

At the end of the forth round, there are $r^4 s$ packets of the form (e_{22} f_1 g_2 f_{21}) queued at e_{22}, and $r^4 s$ packets of the form (e_1 f_{21}) queued at e_1. The adversary described above only uses packets defining simple paths and makes network non-stable when $2r^4 s > s$, i.e., at injection rate $r > 0.84089$. □

Lemma 16. *Any graph in* $\mathcal{E}(S_1)$ *is not stable for* NTG-LIS.

Proof. Let G be the graph in $\mathcal{E}(S_1)$ described at Figure 8(a). There are two vertices a and b with two paths p_n and p_m from a to b and a path p_d from b to a, with $d \geq 2$. We assume that $n \geq m$. We call e and f the first and last edges in p_d, respectively. By including a whole path (e.g. p_d) in the description of the form of a packet, we mean that the path to be followed by the packet must include all this path. The adversary operates in five rounds. Initially, there are $\frac{s}{2}$ packets of the form (p_n e) and $\frac{s}{2}$ of the form (p_m e).

Round 1: for s steps, the adversary injects rs packets of the form (p_d). These injections get mixed with the initial packets at edge e. Except the first rm injections, the rest of them are blocked there because the queueing protocol is NTG (m is the time in which a first packet arrives to e). We consider s big enough to guarantee that a continuous flow arrives to edge e after the arrival of the first packet.

Round 2: for the next $rs - rm$ steps, the adversary injects a set of $r^2 s - r^2 m$ packets of the form $(f\ p_m)$. These packets will be blocked at f by the packets from the first round because they will be at longer distance to their destination, but notice that rd of them will no be blocked (d is the time needed for the first packet in e to arrive to f).

Round 3: for the next $r^2 s - r^2 m - rd$ steps, the adversary injects $r^3 s - r^3 m - r^2 d$ packets of the form (p_m) and $r^3 s - r^3 m - r^2 d$ packets of the form $(f\ p_n)$. The first injections are blocked because secondary protocol is LIS. The later injections are blocked because $n \geq m$ (or also because of the LIS protocol if $n = m$).

Round 4: for the next $r^3 s - r^3 m - r^2 d$ steps, the adversary injects $r^4 s - r^4 m - r^3 d$ packets of the form (p_n), which are blocked due to the LIS effect. Also $r^4 s - r^4 m - r^3 d$ packets of the form $(p_m\ e)$ are injected, which are blocked at the first edge of p_m by packets from the previous round because distance to their destination is longer.

Round 5: for the next $r^4 s - r^4 m - r^3 d$ steps, the adversary injects $r^5 s - r^5 m - r^4 d$ packets of the form $(p_n\ e)$, that will be all blocked by the packets of the form (p_n) from the previous round. The adversary also introduces $r^5 s - r^5 m - r^4 d$ injections of the form (p_m), that will block the packets of the form $(p_m\ e)$ queued since the previous round.
At the end of the fifth round, there are $r^5 s - r^5 m - r^4 d$ packets of the form $(p_n\ e)$ and $r^5 s - r^5 m - r^4 d$ packets of the form $(p_m\ e)$. The adversary described above only uses packets describing simple paths and makes the graph G described above non-stable when $2(r^5 s - r^5 m - r^4 d) > s$. Note that $C = 2(m + d) > 2(r^5 m + r^4 d)$ is a constant quantity and then, for a big enough s, an injection rate r can be found such that $2r^5 s - C > s$ holds. □

Lemma 17. *Any graph in $\mathcal{E}(S_2)$ is not simple-path path stable for* NTG-LIS.

Proof. Let G be a digraph in $\mathcal{E}(S_2)$ described at Figure 8(b). This graph is formed from S_2 by extending its cycles with paths of length m, n and d, $m, n, d \geq 0$. We denote by p_m, p_n and p_d the additional path in the respective side. The adversary operates in four rounds. Initially, there are α packets of the form $(e_{12}\ p_d\ f)$ and β packets of the form $(e_{22}\ p_d\ f)$, where $\alpha + \beta = s$

Round 1: for s steps, the adversary injects rs packets of the form $(p_d\ f\ e_{21})$. These injections are blocked at the first edge of $p_d\ f$ because the queueing protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of $r^2 s$ packets of the form $(f\ e_{11}$ $p_m)$ and also $r^2 s$ packets of the form $(e_{21}\ p_n\ e_{22})$. Both type of injections will collapse with the remaining packets from the previous round but their destination is always farther, so they are blocked. Note, however, that rd of each type of packets are lost[3].

Round 3: for the next $r^2 s - rd$ steps, the adversary injects $r^3 s - r^2 d$ packets of the form $(e_{11}$ $p_m\ e_{12})$, which are blocked by the remaining packets from the previous round because the protocol is NTG. The adversary also injects $r^3 s - r^2 d$ packets of the form $(p_n\ e_{22}\ p_d\ f)$, which block $r^3 s - r^2 d$ of the remaining packets from the previous round because of the same reason.

Round 4: for the next $r^3 s - r^2 d$ steps, the adversary injects $r^4 s - r^3 d$ packets of the form $(e_{12}\ p_d\ f)$, and also $r^4 s - r^3 d$ packets of the form (e_{22}). The injections on e_{12} are blocked by

[3]More exactly, $r(d + 1)$ of the type $(e_{21}\ p_n\ e_{22})$ are lost, but we round to rd to clarify the proof.

packets from the previous round, but rm of them are lost. The simple injections on e_{22} block packets queued at the previous round, but rn of them are lost.

At the end of the forth round, there are $r^4s - r^3d - rm$ packets of the form $(e_{12}\ p_d\ f)$ queued at e_{12} and r^4s-r^3d-rn packets of the form $(e_{22}\ p_d\ f)$ queued at e_{22}. The adversary described above only uses packets defining simple paths and makes the digraph described in Figure 8(b) non-stable when $2(r^4s - r^3d) - rm - rn > s$. Note that $C = 2d + m + n > 2r^3d + rm + rn$, and for a big enough s, an injection rate r can be found such that $2r^4 - C > s$ holds. □

Lemma 18. *Any digraph G in $\mathcal{E}(S_3)$ is not simple-path stable for* NTG-LIS.

Proof. Let G be a digraph in $\mathcal{E}(S_3)$ described at Figure 8(c). This graph is formed from S_3 by extending its cycles with paths of length l and k, $l, k \geq 1$. We denote by p_k and p_l the additional path in the respective side. The adversary operates in four rounds. Initially, there are $\frac{s}{2}$ packets of the form $(e_1\ f_2)$ and $\frac{s}{2}$ packets of the form $(p_l\ f_1\ f_2)$.

Round 1: for s steps, the adversary injects rs packets of the form $(f_2\ p_k)$. These injections are blocked by the initial packets because the queueing protocol is NTG. Note that rl of such new packets are lost.

Round 2: for the next $rs - rl$ steps, the adversary injects a set of $r^2s - r^2l$ packets of the form $(p_k\ e_1\ e_2)$. These injections are blocked by the remaining packets from the previous round because the queueing protocol is NTG.

Round 3: for the next $r^2s - r^2l$ steps, the adversary injects $r^3s - r^3l$ packets of the form $(e_2\ p_l)$ and $r^3s - r^3l$ of the form $(e_1\ f_2)$. These injections will be blocked by packets from the previous round when collapsing at edges e_2 and e_1 respectively, but rk of them will be lost.

Round 4: for the next $r^3s - r^3l - rk$ steps, the adversary injects $r^4s - r^4l - r^2k$ packets of the form (e_1) that will block packets from the previous round at e_1 because they are closer to their destination. The adversary also injects $r^4s - r^4l - r^2k$ packets of the form $(p_l\ f_1\ f_2)$ that will be blocked by the remaining $(e_2\ p_l)$ packets from the previous round.

At the end of the forth round, there are $r^4s - r^4l - r^2k$ packets of the form $(p_l\ f_1\ f_2)$ and $r^4s-r^4l-r^2k$ packets of the form $(e_1\ f_2)$. The adversary described above uses packets following simple paths and and makes the graph $G \in \mathcal{E}(S_3)$ non-stable when $2(r^4s - r^4l - r^2k) > s$. Note that $C = 2(l + k) > 2(r^4l + r^2k)$, and for a big enough s, an injection rate r can be found such that $2r^4s - C > s$ holds. □

Lemma 19. *Any digraph G in $\mathcal{E}(S_4)$ is not simple-path path stable for* NTG-LIS.

Proof. Let G be a digraph in $\mathcal{E}(S_4)$. Notice that, if one of the arcs in the 2-cycle is subdivided, G contains as a subgraph a graph in $\mathcal{E}(S_3)$ that, by lemma 18, is not NTG-LIS stable. Therefore we will consider the case described at Figure 8(d). This graph is formed from S_4 by extending its 3-cycles with paths p_l and p_k of length l and k $(l, k \geq 1)$ and by applying m 2-cycle successive subdivisions $(m \geq 1)$. p_m^1 will denote the path $(g_{11}\ g_{12}\ \dots g_{1m})$ and p_m^2 the path $(g_{21}\ g_{22}\ \dots g_{2m})$. The adversary operates in four rounds. Initially, there are $\frac{s}{2}$ packets of the form $(e_1\ f_2)$ and $\frac{s}{2}$ of the form $(p_l\ f_1\ p_m^2\ f_2)$.

Round 1: for s steps, the adversary injects rs packets of the form $(f_2\ p_k)$. These injections get mixed with the initial packets at edge f_2, and are blocked there because the queueing protocol is NTG. Note that $r(m + l)$ of such new packets are lost.

Round 2: for the next $rs - r(m+l)$ steps, the adversary injects a set of $r^2 s - r^2(m+l)$ packets of the form $(p_k\ e_1\ g_{1m})$ that are blocked at the first edge of p_k by the packets from the first round because, at that point, they are at longer distance to their destination.

Round 3: for the next $r^2 s - r^2(m+l)$ steps, the adversary injects $r^3 s - r^3(m+l)$ packets of the form $(p_m^1\ e_2\ p_l)$ and $r^3 s - r^3(m+l)$ of the form $(e_1\ f_2)$. The injections introduced at e_1 reach the packets from the previous round at this edge, but they are blocked there because the secondary protocol is LIS. When the old packets arrive at their last edge g_{1k}, they block the injections introduced.

Round 4: for the next $r^3 s - r^3(m+l)$ steps, the adversary injects $r^4 s - r^4(m+l)$ packets of the form $(p_l\ f_1\ p_m^2\ f_2)$ and $r^4 s - r^4(m+k)$ packets of the form (e_1). The simple injections on e_1 will block the packets queued at this edge from the previous round. The other injections are blocked at the first edge of p_l by packets from the previous round because distance to their destination is longer, but note that rm of them are lost.

At the end of the forth round, there are $r^4 s - r^4(m+l)$ packets of the form $(e_1\ f_2)$ queued at e_1 and $r^4 s - r^4(m+l) - rm$ packets of the form $(p_l\ f_1\ p_m^2\ f_2)$ queued at the first edge of p_l. The adversary described above only uses packets following simple paths and makes the digraph G described in Figure 8(d) non-stable when $2(r^4 s - r^4(m+l)) - rm > s$. Note that $C = 2(m+l) > 2r^4(m+l) - rm$, and for a big enough s, an injection rate r can be found such that $2r^4 - C > s$ holds. $\square$