Alcom-FT Technical Report Series
ALCOMFT-TR-02-103

Universal stability of undirected graphs in the adversarial queueing model *

Carme Àlvarez   Maria Blesa   Maria Serna

Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica Catalunya
Jordi Girona Salgado 1–3
08034-Barcelona
{alvarez,mjblesa,mjserna}@lsi.upc.es

Abstract

In this paper we study the universal stability of undirected graphs in the adversarial queueing model for packet routing. In this setting packets must be injected in some edge and have to traverse a path before leaving the system. Restrictions on the allowed types of path that packets must traverse provide different packet models. We consider three natural models, and provide polynomial time algorithms for testing universal stability. In the three cases we obtain a different characterization, thus showing that slight variations gives raise to non equivalent models.

We finally extend the results to show that the universal stability of digraphs, in the case in which packets follow directed paths without repeated vertices, can be decided in polynomial time.

*Work partially supported by the IST Programme of the EU under contracts number IST-1999-14186 (ALCOM-FT) and IST-2001-33116 (FLAGS) and by the Spanish CICYT projects TIC-1999-0754-C03 (Mallba) and TIC-2000-1970-CE. The second author was also supported by pre-doctoral grant 2001FI-00659

1 Introduction

Adversarial Queueing Theory The model of Adversarial Queueing Theory proposed by Borodin et al. [3] considers the time evolution of a packet-routing network as a game between an adversary and a protocol. At each time step the adversary may inject a set of packets to some of the nodes. For each packet the adversary specifies a sequence of edges that it must trasverse, after which the packet will be absorbed. If more than one packet wishes to cross an edge e at the same time step, then the protocol chooses one of these packets to send across e. The remaining packets wait in the e queue. This game then advances to the next time step.

A crucial parameter of the adversary is its *injection rate*. The rate of an adversary in this model, is specified by a pair (r, b) where $b \geq 1$ is a natural number and $0 < r < 1$. The adversary must obey the following rule:

> Of the packets that the adversary injects in any interval I, at most $\lceil r|I| \rceil + b$ may have paths that contain any particular edge.

The reason for studying the behavior of packet communication networks, is to determine the conditions of *stability*, the fact that the number of packets in the system remains bounded, as the system dynamically evolves in time. Andrews et al. [2] solved several open questions posed in [3]. They also showed the existence of graphs and protocols that are not universally stable.

Greedy protocols In this paper, as in Borodin et al. [3], we only consider greedy protocols, those that advance a packet across an edge e whenever there is at least one packet waiting to traverse e. Some natural greedy protocols are: FIFO (First In First Out), every queue gives priority to the packet that arrives first in the queue, LIS (Longest In System), every queue gives priority to the packet that has been in the system the longest and NTG (Nearest To Go), every queue gives priority to the packet that has to cross the smallest number of edges. Note that there are graphs which are unstable under FIFO and NTG but all graphs are stable under LIS [2]. All the instability results in this paper will be for the NTG-LIS protocol that works as NTG, but solving ties using the LIS protocol.

Packet trajectory For directed graphs, two different types of paths have been considered in the literature. Borodin et al. [3] and Andrews et al. [2] assume that when a packet is injected, its assigned path is simple, i.e. it does not contain any edge more than once (see pag. 45 of [2]).

Goel considers a different definition of *simple path* in [6]. Even though he does not give any detailed definition, he appears to use the term 'simple path' to refer to paths that do not contain any vertex more than once (see for example the comment just before Lemma 2.4 on page 221 of [6]). Hence, Goel's paths, refered to here as simple paths, are a restrictive version of Andrews et al.'s paths to which we assing the term 'general path' or just path.

For 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 pag. 57 of [2]). We refer to these paths as undirected paths.

Universal stability Different definitions of universal stability can been considered depending on the type of paths that packets may follow. Alvarez et al.[1] give a characterization of

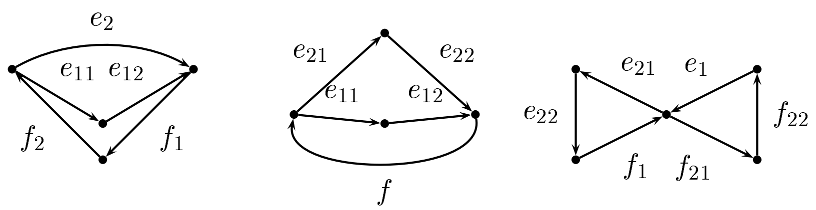


Figure 1: Goel's proposed forbidden minors for digraph simple-path universal stability

universal stability for digraphs in the general case and disprove the Goel's characterization, for the case of simple paths presented in terms of the forbidden minors given in Figure 1. For undirected graphs, Andrews at al. show in [2] the undirected-path universal stability of cycles and trees. In the same paper Andrews et al. show that the family of undirected-path universally stable graphs is minor-closed. This guaranties decidability in polynomial time in this model. However a constructive proof is not presented.

Gamarnik provided in [5] a characterization of universal stability for undirected simple graphs assuming that only one packet can cross an edge at a time from either end. Note that this represents a restriction on the queue policy. Each edge is undirected and can 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.

New results In this paper we consider universal stability for different types of paths for undirected graphs. For each case we obtain a characterization in terms of forbidden subgraphs and provide an explicit polynomial time algorithm. Using the same techniques we also obtain a characterization of universal stability in the simple-path model for directed graphs, thus tying up the loose ends left in [1]. We would like to highlight the importance of specifying the type of paths since for each category the characterizations of universal stability are different.

As all the instability results in this paper have been obtained using the NTG-LIS protocol, we conclude that universal stability coincides with NTG-LIS stability. It would be interesting to know whether this property can be extended to other natural greedy protocols.

Organization Section 2 sets out the definitions of all models considered here. In Section 3 universal stability of some particular graphs and digraphs is shown, while Section 4 contains the basic results on instability for the NTG-LIS protocol. Finally, Section 5 details the characterizations for universal stability.

2 Definitions

All the graphs in this paper may have multiple edges but no loops. 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. Two edges are *incident* if they share at least one vertex. A *walk* is an alternating sequence of vertices and edges (arcs) $v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n$ such that v_0, v_n are vertices and that for each i, $1 \leq i \leq n$, $e_i = \{v_{i-1}, v_i\}$ ($e_i = (v_{i-1}, v_i)$). A *path* is a walk in which all the edges (arcs) are different. A *simple path* is a walk in which all the vertices, and thus necessarily all the edges (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.

We will consider the following subdivision operations:

- The *subdivision of an edge* $\{u, v\}$ in a graph G consists in 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 in 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 in 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 graphs obtained from G by succesive edge subdivisions. Given a digraph G, $\mathcal{E}(G)$ denotes the family of digraphs formed by G and all digraphs obtained from G by succesive arc or 2-cycle subdivisions.

Given a graph G, G^d denotes the digraph formed from the same vertex set as G but replacing every edge $\{u, v\}$ with the two arcs (u, v) and (v, u). Observe that, in general, for a graph G, $\mathcal{E}(G)^d \neq \mathcal{E}\left(G^d\right)$.

For digraphs we will consider two kind of packet paths that will raise two definitions of stability.

Definition 1. *Given a digraph G and a protocol P, we say that*

- *G is* stable *for protocol P (the pair (G, P) is stable) if for any adversary the maximum number of packets in the system is bounded, when packets follow a path in G.*
- *G is* simple-path stable *for protocol P (the pair (G, P) is simple-path stable) if for any adversary the maximum number of packets in the system is bounded, when packets follow a simple path in G.*

Definition 2. *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.*

It is straightforward to prove that, in the two models, the instability of a subgraph implies the instability of the whole graph. Observe that any universally stable digraph is also simple-path universally stable, but the opposite is not true.

In the case of undirected graphs we consider three different notions of stability:

Definition 3. *Given a graph G and a protocol P, we say that*

- *G is* stable *for protocol P (the pair (G, P) is stable) if for any adversary the maximum number of packets in the system is bounded, in the case that packets follow a path in G^d.*
- *G is* undirected-path stable *for protocol P (the pair (G, P) is undirected-path stable) if for any adversary the maximum number of packets in the system is bounded, in the case that packets follow a path in G.*
- *G is* simple-path stable *for protocol P (the pair (G, P) is simple-path stable) if for any adversary the maximum number of packets in the system is bounded, in the case that packets follow a simple-path in G^d.*

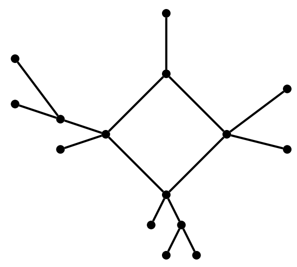


Figure 2: a decorated-cycle

Observe that in the first case an edge can be used twice by the same packet provided it its traversed in opposite directions. In the second model the same edge can be traversed only once, that is Andrews et al's model. The third model does not allow a packet to pass twice through the same vertex.

It is straightforward to show that in the three models the instability of a subgraph implies the instability of the whole graph. Furthermore, 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.

Now we have the corresponding definitions of universal stability.

Definition 4. *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 under the undirected-path model was addressed in [2] where it is proved that the property is closed under minors and therefore decidible in polynomial time, however no constructive algorithm is known. For the other models this question was unresolved.

3 Some universally stable graphs and digraphs

In this section, we use standard terminology for the following graphs and digraphs that will have no multiple edges: *trees*, the *cycle* on k vertices ($k \geq 3$), *unicyclic* graphs, those graphs that only contain one cycle, the *directed cycle* on k vertices ($k \geq 2$) and *directed acyclic* graphs.

A *multi-tree* will be a tree with multiple edges. A *decorated cycle* is obtained from a cycle on k vertices ($k \geq 3$) and some multi-trees, after the identification of one vertex from each tree with a cycle vertex (see Figure 2 for an example).

In this section we prove the universal stability of some graphs and digraphs, the first three results are already known.

Lemma 1 ([3]). *All acyclic digraphs are universally stable.*

Lemma 2 ([2]). *The directed cycle on any number of vertices is universally stable. The cycle on any number of vertices is undirected-path universally stable. All trees are undirected-path universally stable.*

The following result was proved in [6] for simple-path universal stability, a close inspection of the proof reveals,

Lemma 3. *If digraphs G_1 and G_2 are universally stable, then so is any graph G formed by joining them with edges that go only from G_1 to G_2.*

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

Proof. Observe that if G is an unicyclic graph, removing the edges in the cycle results in a forest. We root each tree in this forest at the vertex that is common with the cycle, this gives an orientation *upwards* or *downwards* to every arc in G^d.

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

Observe that packets that start in a downward arc can only follow downward arcs, otherwise they will use twice an edge in G. Similarly, packets that start in the cycle cannot change the initial cycle 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 flows. 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, in which no flow can be passed from one to the other, this edges can get packets from the α flow and 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 3, the corresponding queues will have bounded maximum size. For the downwards arcs we have the same situation, a flow comming from a stable network entering in another stable network, and thus the result follows. □

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

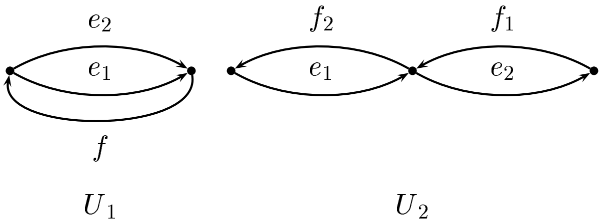
Proof. Observe that a packet can only use once, and only in one direction, one of the multiple edges connecting the same pair of vertices. The argument is similar to the one in Lemma 4, we root the tree in order to assign an upward/downward direction to the edges in G^d. The simple-path requirement forces 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 then 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.*

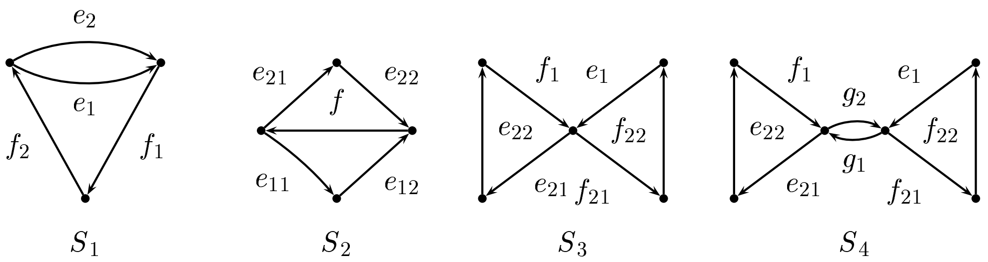
Proof. The result follows from Lemmas 5 and 2 using the same arguments of Lemma 4 □

4 Instability results for the NTG-LIS protocol

In this section we will prove instability results for the NTG-LIS protocol. To keep ligther the notation we will specify a path by only giving its edges. The names of the graph correspond to the graphs depicted in Figure 3. For the first three graphs instability results are already known.



(a) unstable digraphs



(b) simple-path unstable digraphs

Figure 3: Some unstable digraphs for the NTG-LIS protocol

Lemma 7 ([1]). *The pairs* $(U_1, \text{NTG-LIS})$, $(U_2, \text{NTG-LIS})$ *are not stable, and for any digraph* $G \in \mathcal{E}(U_1) \cup \mathcal{E}(U_2)$, *the pair* $(G, \text{NTG-LIS})$ *is not stable.*

Lemma 8 ([1]). *The pair* $(S_1, \text{NTG-LIS})$ *is not simple-path stable and, for any digraph* $G \in \mathcal{E}(S_1)$, *the pair* $(G, \text{NTG-LIS})$ *is not stable.*

Let us prove that the remaining digraphs are unstable for NTG-LIS.

Lemma 9. *The pair* $(S_2, \text{NTG-LIS})$ *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 queueing protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of r^2s packets of the form $(f\ e_{11})$ and r^2s 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^2s steps, the adversary injects r^3s packets of the form $(e_{22}\ f)$ and r^3s 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 farer.

Round 4: for the next r^3s steps, the adversary injects r^4s packets of the form $(e_{12}\ f)$ and r^4s 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 $2r^4s$ packets queued at e_{12} and e_{22} that want to traverse edge f; r^4s are of the form $(e_{12}\ f)$ and r^4s are of the form $(e_{22}\ f)$. The adversary described above only uses simple paths and makes network S_2 unstable when $2r^4s > s$, i.e. at injection rate $r > 0.84089$. □

Lemma 10. *The pair* $(S_3, \text{NTG-LIS})$ *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 are blocked in edge f_{21} 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 $(f_{22}\ e_1\ e_{21})$ 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^2s steps, the adversary injects r^3s packets of the form $(e_{21}\ e_{22})$ and r^3s 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^3s steps, the adversary injects r^4s packets of the form $(e_{22}\ f_1\ f_{21})$ and r^4s 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^4s packets of the form $(e_{22}\ f_1\ f_{21})$ queued at e_{22}, and r^4s 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 S_3 unstable when $2r^4s > s$, i.e. at injection rate $r > 0.84089$. □

Lemma 11. *The pair* $(S_4, \text{NTG-LIS})$ *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 queueing protocol is NTG.

Round 2: for the next rs steps, the adversary injects a set of r^2s 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^2s steps, the adversary injects r^3s packets of the form $(g_1\ e_{21}\ e_{22})$ and r^3s 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^3s steps, the adversary injects r^4s packets of the form $(e_{22}\ f_1\ g_2\ f_{21})$ and r^4s 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 unstable when $2r^4 s > s$, i.e. at injection rate $r > 0.84089$. □

All the graphs in $\mathcal{E}(S_1) \cup \mathcal{E}(S_2) \cup \mathcal{E}(S_3) \cup \mathcal{E}(S_4)$ are unstable for the NTG-LIS protocol. A detailed proof can be found in Appendix a. Observe that in the NTG protocol if the length of the path that a packet has to traverse is increased, its priority at a given edge can be changed. Therefore, edges in a path packet can not be replaced indistinctly with paths. However, for the particular graphs and adversaries we are dealing with, the adversary can be adapted to provide an instability proof.

5 Deciding universal stability

In this section we provide characterizations for the universal stability property for all the models, and give polynomial time algorithms for testing such property. The graph nomenclature corresponds to Figures 4 and 3.

We start given some known results for directed graphs that will be used later or complete the set of characterizations.

Lemma 12 ([6]). *A digraph is universally stable if and only if all of its strongly connected components are universally stable.*

Theorem 1 ([1]). *A digraph is universally stable if and only if it does not contain as subgraphs any of the digraphs in $\mathcal{E}(U_1) \cup \mathcal{E}(U_2)$.*

For the case of universal stability, the basic set of forbidden subgraphs is given in Figure 4(b).

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

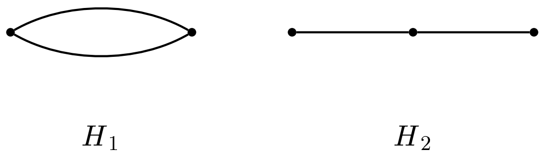
Proof. If G has two incident edges we have two cases, corresponding to the two the graphs given in Figure 4(a). If the two edges form a simple path, the digraph G^d contains U_2 as a subgraph and thus the pair $(G, \text{NTG-LIS})$ is unstable by lemma 7. If the two edges form a 2-cycle, the digraph G^d contains U_1 as a subgraph and thus the pair $(G, \text{NTG-LIS})$ is unstable by lemma 7.

If G has no incident edges all its connected components have only one edge with multiplicity one. In such a case all the connected strongly components of the digraph G^d are either isolated vertices or a directed cycle on two vertices. In both cases the graph G^d is universally stable. □

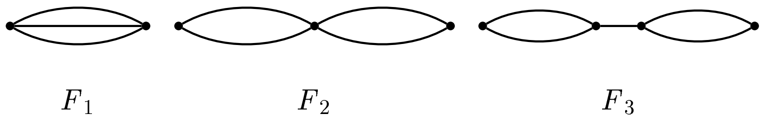
For the case of undirected-path universal stability, the basic set of forbidden subgraphs is given in Figure 4(a).

Theorem 3. *A graph G is undirected-path universally stable iff G does not contain as a subgraph any of the graphs in $\mathcal{E}(F_1) \cup \mathcal{E}(F_2) \cup \mathcal{E}(F_3)$*

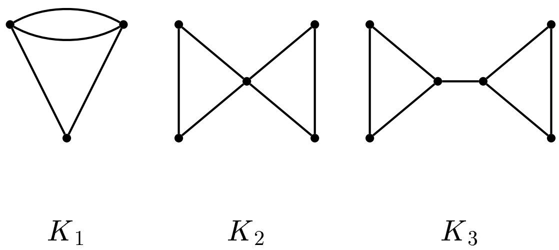
Proof. Let us prove first that the graphs F_1, F_2 and F_3 are NTG-LIS undirected-path unstable. Giving an adequate orientation to the edges in F_1 and F_2 we get the digraphs U_1 and U_2, as each arc correspond to a different edge the instability follows. Maintainig these orientations,



(a) no universally stable graphs



(b) no universally undirected-path stable graphs



(c) no universally simple-path stable graphs

Figure 4: Some no universally stable graphs

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 NTG-LIS undirected-path unstable.

For the graph F_3 we can fix an orientation for the edges forming the 2-cycles, but we have to keep both directions for the middle edge. Starting with the proof of NTG-LIS instability for the graph U_2 and working in a similar way as in Lemmas 11 and 15 it follows that F_3 and any graph in $\mathcal{E}(F_3)$ is NTG-LIS undirected-path unstable.

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 all the connected components of G contain at most one cycle, so they are subgraphs of an unicyclic graph. Therefore, as by Lemma 4, all the connected components of G are stable, the graph G is undirected-path universally stable. □

For the case of simple-path universal stability, the basic set of forbidden subgraphs is given in Figure 3(b)

Theorem 4. *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. 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 that are NTG-LIS simple-path unstable. For the graph K_3 we can orient the two cycles and maintain the two oposite arcs for the middle edge, obtaining the graph S_4. Lemmas 8, 14 and 15 give the NTG-LIS simple-path instability for any graph in $\mathcal{E}(K_1) \cup \mathcal{E}(K_2) \cup \mathcal{E}(K_3)$.

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 are subgraphs of a decorated cycle graph therefore using Lemma 6, G is simple-path universally stable. □

For the case of simple-path universal stability of digraphs, the basic set of forbidden subgraphs is given in Figure 3.

Theorem 5. *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. The instability part follows from Lemmas 8, 13, 14 and 15. When G excludes the family of forbidden subgraps, each strongly connected component of G must be a subgraph of the directed version of a decorated cycle graph. Therefore, by Lemma 6, all the strongly connected components are simple-path universally stable, and by Lemma 12, G is simple-path universally stable. □

All the instability results hold for the NTG-LIS protocol, and we have the following equivalences.

Theorem 6.
A graph G is universally stable iff G is stable for NTG-LIS.
A graph G is undirected-path universally stable iff G is undirected-path stable for NTG-LIS.
A graph G is simple-path universally stable iff G is simple-path stable for NTG-LIS.
A digraph G is simple-path universally stable iff G is simple-path stable for NTG-LIS.

The property giving the characterization for universal stability, in the four cases, can be decided in polynomial time. For undirected graphs we could use the polynomial time algorithm for checking subgraph homeomorphism for the corresponding forbidden subgraphs given in [7], however our algorithm is much simpler. Notice that for directed graphs checking subdigraph homeomorphism can be NP-complete, in particular this is the case for the digraphs S_1, S_2, and S_3 see [4]. However the combination of the four conditions is easier to test. A detailed description of the algorithms is provided in Appendix b.

References

[1] C. Àlvarez, M. Blesa, and M. Serna. On Universal Stability of Directed Graphs in the Adversarial Queueing Model. Technical report LSI-02-4-R, Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 2002. Submitted.

[2] 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.

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

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

[5] D. Gamarnik. Stability of adaptative and non-adaptative packet routing policies in adversarial queueing networks. In *31st. Annual ACM Symposium on Theory of Computing*, pages 206–214, May 1999.

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

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

A Instability of extended graphs

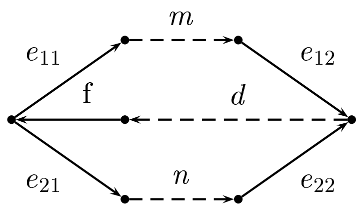


Figure 5: A digraph $\mathcal{E}(S_2)$

Lemma 13. *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 5. 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^2s packets of the form $(f\ e_{11}\ p_m)$ and also r^2s 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 farer, so they are blocked. Note, however, that rd of each type of packets are lost[1].

Round 3: for the next $r^2s - rd$ steps, the adversary injects $r^3s - r^2d$ 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 $r^3s - r^2d$ packets of the form $(p_n\ e_{22}\ p_d\ f)$, which block $r^3s - r^2d$ of the remaining packets from the previous round because of the same reason.

Round 4: for the next $r^3s - r^2d$ steps, the adversary injects $r^4s - r^3d$ packets of the form $(e_{12}\ p_d\ f)$, and also $r^4s - r^3d$ packets of the form (e_{22}). The injections on e_{12} are blocked by 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 5 unstable 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. $\square$

Lemma 14. *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 6. 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.

[1] 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.

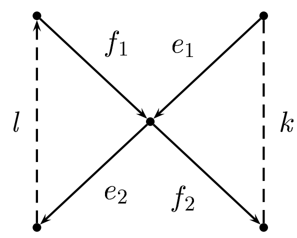


Figure 6: A graph in $\mathcal{E}\ (S_3)$

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 $r^3s - r^3l$ packets 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 $G \in \mathcal{E}\ (S_3)$ unstable 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. □

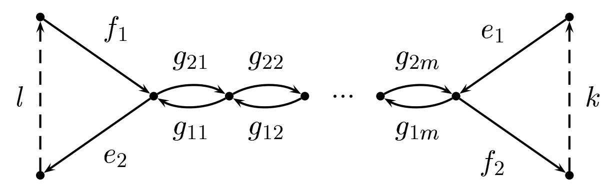


Figure 7: A graph in $\mathcal{E}\ (S_4)$

Lemma 15. *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 14 is NTG-LIS unstable. Therefore we will consider the case described at Figure 7. This graph is formed from S_4 by extending its 3-cycles with paths of length l and k, $l, k \geq 1$, denoted by p_l and p_k. And by m, $m \geq 1$, 2-cycle succesive subdivisions. p_m^1 will denote the path $(g_{11}\ g_{12}\ \ldots\ g_{1m})$ and p_m^2 the path $(g_{21}\ g_{22}\ \ldots\ g_{2m})$.

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 7 unstable 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. □

B Algorithms for checking universal stability

The following algorithm checks whether a connected graph G, without multiple edges, is a subgraph of an unicyclic graph, in polynomial time.

Subgraph of an unicyclic graph
If G is a tree, then return yes.
Otherwise,
  compute a cycle $C = (v_k, e_1, v_1, \ldots, v_{k-1} e_k, v_k)$ in G.
  Let G' be the graph obtained after removing the edges in C
  Compute the connected components of G'.
  If a connected component of G' contains two vertices in C,
    then return no
  If a connected component of G' is not a tree,
    then return no
  Otherwise, return yes.

The universal stability algorithms combine this checking with some additional testing for the multi-edges. The following algorithm checks the undirected-path universal stability of a given graph G in polynomial time.

Graph undirected-path stability
If some edge has multiplicity 3,
  then G is not undirected-path universally unstable.
Otherwise,
  compute the connected components of G.
  If a connected component has two edges with multiplicity 2,
    then G is unstable.
  Now all the connected components have
    at most one edge with multiplicity 2.
  Let G' be the graph obtained from G by
    setting all edges multiplicities to one.
  If there is
    a connected component of G' that is not a subgraph of an unicyclic graph,
      then G is not undirected-path universally stable.
  Otherwise, all the connected components of G' have at most one cycle.
      If in some component the multiple edge belongs to a cycle
        then G is not undirected-path universally stable.
      Otherwise, G is undirected-path universally stable.

The following algorithm checks the simple-path universal stability of a given graph G, in polynomial time.

Graph simple-path stability
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' such that
  H is not a subgraph of an unicyclic graph, or
  H is an unicyclic graph with some cycle edge
    having multiplicity bigger than 1 in G,
      then G is not simple-path universally stable.
Otherwise, G is simple-path universally stable.

The following algorithm checks the simple-path universal stability of a given strongly connected digraph G.

Digraph simple-path stability
If G does not have a directed k-cycle with $k \geq 3$,
  then G is simple-path universally stable.
Otherwise,
  compute a directed cycle $C = (v_k, e_1, v_1, \dots, v_{k-1} e_k, v_k)$ in G $(k \geq 3)$.
  If some of the cycle arcs has multiplicity bigger than one,
    then G is not simple-path universally stable.

Let e'_i be the arc opposite to e_i, $1 \leq i \leq k$.
If all the arcs e'_i are present in G and
some of them has multiplicity bigger than one,
then G is not simple-path universally stable.
Let G' be the digraph obtained from G by setting
the multiplicity of all arcs in G to one, and
removing the arcs in C and all the opposite arcs e'_i (if any).
If there are two cycle vertices connected by a directed path in G',
then G is not simple-path universally stable.
Otherwise,
Compute the strongly connected components of G'.
If a strongly connected component of G' contains a
directed k-cycle with $k \geq 3$,
then G is not simple-path universally stable.
Otherwise G is simple-path universally stable.

All the algorithms presented in this section run in polynomial time. Note that the most costly operation is the computation of the (strongly) connected components of a (di)graph.