The Range of Stability for Heterogeneous and FIFO Queueing Networks

Electronic Colloquium on Computational Complexity, Report No. 99 (2001)

D. K. Koukopoulos, S. E. Nikoletseas, and P. G. Spirakis *

Computer Technology Institute (CTI) and Patras University
Riga Fereou 61, P.O.Box 1122-26 110 Patras, Greece
e-mail:koukopou,nikole,spirakis@cti.gr

Abstract. In this paper, we investigate and analyze for the first time the stability properties of heterogeneous networks, which use a combination of different universally stable queueing policies for packet routing, in the Adversarial Queueing model. We interestingly prove that the combination of SIS and LIS policies, LIS and NTS policies, and LIS and FTG policies leads to instability for specific networks and injection rates that are presented. It is also proved that the combination of SIS and FTG policies, SIS and NTS policies, and FTG and NTS policies is universally stable. Furthermore, we prove that FIFO is non-stable for any $r \geq 0.749$, improving significantly the previous best known bounds of [2, 11], by using new techniques for adversary construction and tight analysis of the packet flow time evolution.

1 Introduction

In this paper, we study the behavior of packet-switched communication networks in which packets arrive dynamically at the nodes and are routed in discrete time steps across the edges. A crucial issue that arises in such a setting is that of *stability* - will the number of packets in the system remain bounded, as the system runs for an arbitrary long period of time? The answer to this question typically depends on the *rate* at which packets arrive into the system, and on the contention-resolution *protocol* that is used when more than one packet wants to cross a given edge in a single time step.

The stability problem has been investigated under various models of packet routing and in a number of overlapping areas, see for example [7, 8, 5, 6, 9, 1]. The *adversarial queueing model* of Borodin et al. [3], was developed as a robust model of queueing theory in network traffic, and replaces stochastic by worst case inputs. The underlying goal is to determine whether it is feasible to prove stability even when packets are injected by an adversary, rather than by an oblivious randomized process. Adversarial Queueing Theory considers the time evolution of a packet-routing network as a game between an adversary and a protocol. The adversary, at each time step, may inject a set of packets at some nodes. For each packet, the adversary specifies a simple network path that the packet must traverse and, when the packet arrives to its final destination, it is absorbed by the system. If more than one packet wish to cross an edge e in the current time step, then a contention resolution protocol is used to resolve the conflict. We use the equivalent term policy or service discipline for such a protocol.

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 $\lceil r|I| \rceil + b$ can have paths that contain any particular edge." Such a model allows for adversarial injection of packets that are "bursty".

In this paper, we consider only *greedy* (also known as *work-conserving*) protocols - those that advance a packet across an edge e whenever there is at least one packet waiting to use e. We. also, consider discrete time units $t = 0, 1, 2, \ldots$. In each time unit a queue can serve exactly one packet.

In particular, we study the stability properties of five greedy service disciplines. The *Shortest-in-System* (SIS) policy gives priority at every queue to the packet that was injected most recently in the system. On the contrary, the *Longest-in-System* (LIS) policy gives priority at every queue to the packet that has been in the system the longest. The *Farthest-to-Go* (FTG) policy gives priority to the packet that still has to traverse the largest number of queues, while the *Nearest-to-Source* (NTS) policy gives priority to the packet that has traversed the smallest number of edges. Finally, *First-In-First-Out* (FIFO) policy gives priority to the packet that has arrived first in the queue.

* The work of all the authors was partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT)

ISSN 1433-8092

We emphasize the fact that until now the stability of these protocols has been studied in isolation i.e. in networks where all queues obey a single service policy. In this work, we study for the first time, *combinations* of these policies. The issue of the stability of combinations of policies studied in this paper is highly motivated by the fact that modern communication networks (and the Internet) are heterogeneous in the sense that more than one disciplines are used on the network servers.

We also focus on FIFO, since FIFO is one of the simplest queueing policies and has been used to provide best-effort services in packet-switched networks.

In [2, 3], the authors highlight a very basic algorithmic question: when is a given contention-resolution protocol stable in a given network, against a given adversary? More specifically, these questions are based on the following definitions.

Definition 1. *We say that a protocol P is stable on a network G against an adversary A if there is a constant C (which may depend on G and A) such that, starting from an empty configuration, the number of packets in the system at all times is bounded by C.*

Definition 2. *We say that a graph G is universally stable if every greedy protocol is stable against every adversary of rate less than 1 on G.*

Definition 3. *We say that a protocol P is universally stable if it is stable against every adversary of rate less than 1, on every network.*

The current state-of-the-art mostly focuses on FIFO and also other greedy protocols on networks whose queues use a single service discipline. In particular, in a fundamental work in the field of stability, Andrews et al. [2] have proved that LIS, SIS, NTS, and FTG policies are universally stable and that FIFO is unstable for a particular network for $r \geq 0.85$.

Later, Goel [10], in a very interesting work, presented an even simpler network of only 3 queues (see Figure 1) for which FIFO is unstable, as a corollary of his main Theorem investigating structural conditions for the decidability of the question if a given network is stable for FIFO.

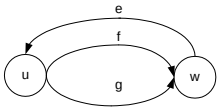

Fig. 1. Goel's network

In [11], the FIFO instability bound is slightly improved by our team by showing a network for which FIFO is unstable for $r \geq 0.8357$. Also, an open question raised by Andrews et al. [2] is *partially* answered in [11] by showing for the first time upper bounds on r for the stability of FIFO in networks with a finite number of queues. These bounds depend on the size of the network.

Summary of results: In this work,

1. *For the first time*, we investigate stability properties of *combinations* of queueing disciplines.
2. We demonstrate instability of certain combinations of universally stable protocols in the same network.
3. We prove universal stability of certain other combinations of universally stable service policies in the same network.
4. We significantly improve the best previously known [2, 11] FIFO instability lower bound by presenting a FIFO network where $r \geq 0.749$ leads to instability.
5. We present the first instability bound for $r < 0.708 = \frac{1}{\sqrt{2}}$, where $r = 0.708$ is the lower instability bound for which we have seen a constructive proof [2] (with the exception of the result of [4] for LIFO, NTG and FFS, where the authors show that there exist arbitrarily small rates for instability of these protocols).
6. In the second result above, we interestingly show that for instability it suffices *to have only two queues* with a policy different from all other queues in the network.

In particular, our results on combinations of queueing disciplines are summarized in the table below. These results seem to suggest the need for an extended definition of the notion of universal stability of protocols, taking into account their sensitivity to changing the discipline of some (even a small number) of the queues in the network.

Protocols Combination	*Stable?*
LIS-SIS	No
LIS-NTS	No
LIS-FTG	No
SIS-NTS	Yes
SIS-FTG	Yes
FTG-NTS	Yes

Table 1. Universal stability of combinations of universally stable protocols considered.

We also provide two new FIFO instability lower bounds, based on new approaches and techniques, which might independently significantly benefit possible future research. For the first improved bound ($r \geq 0.771$) we introduce a new type of inductive hypothesis, assuming that the initial packets are in more than one queues, taking also care of their relative spread among the queues so as to enable the packets to proceed fast enough to block possible new injections. For the second bound ($r \geq 0.749$), we perform in addition a tight analysis of the time evolution of packet flow by exactly estimating the number of the remaining packets in various queues as time proceeds in order to precisely measure the actual delay imposed. To allow for further delay of initial packets, we introduce a modified network with additional queues where appropriate injections are performed.

For simplicity, and in a way similar to that in [2], we omit floors and ceilings and sometimes count time steps and packets roughly. This only results to loosing small additive constants while we gain in clarity.

For lower bounds of the type we are interested in obtaining in this paper, it is advantageous to have an adversary that is as weak as possible. Thus, for these purposes, we say that an adversary A has rate r if for every $t \geq 1$, every interval I of t steps, and every edge e, it injects no more than $\lceil r|I| \rceil$ packets during I that require e at the time of their injection.

We will present our lower bounds for systems that start from a non-empty initial configuration. This implies instability results for systems with an empty initial configuration, by the following lemma presented in Andrews et al. [2].

Lemma 1. *Let G be a graph, P be a greedy protocol, and A an adversary of rate r, and suppose the system (G, A, P) is unstable starting with some non-empty initial configuration. Then, there exists a system (G', A', P) that is unstable starting with an empty initial configuration, where A' is an adversary of rate r.*

Organisation of the paper: In section 2 we present instability results for combinations of the (in isolation) universally stable protocols LIS and SIS, LIS and FTG, LIS and NTS. In the next section, we show universal stability of mixing universally stable policies, such as SIS and FTG, SIS and NTS, FTG and NTS. In section 4 we present the two improved FIFO instability bounds and discuss the techniques used. After the references an appendix is given.

2 Non-Stable Combinations of Universally Stable Policies

We split the time into periods. In each period we study the number of initial packets by considering corresponding time rounds. For each period, we inductively prove for instability that the number of packets in the system increases. This inductive argument can be applied repeatedly, thus showing instability. Our constructions use symmetric networks of two parts. The inductive argument has to be applied twice so that increased population appear in the same queues, since we use symmetric networks.

2.1 Mixing of LIS and SIS is Unstable

Theorem 1. *Let $r \geq 0.683$. There is a network G that uses LIS and SIS as queueing disciplines and an adversary A of rate r, such that the (G, A, LIS, SIS) system is unstable, starting from a non-empty initial configuration.*

Proof. Let's consider the network in Figure 2.

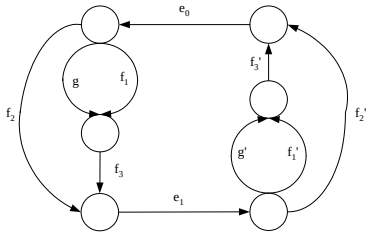

Fig. 2. A Network that uses LIS and SIS as queueing disciplines.

All the queues of this network use the LIS queueing discipline except from the queues that correspond to the edges $g, g^{'}$ that use the SIS queueing discipline.

Inductive Hypothesis: If at the beginning of phase j, there are s packets queued in the queues $e_0, f_3^{'}$ requiring to traverse edges e_0, g, f_3, then at the beginning of phase $j+1$ there will be more than s packets queued in the queues e_1, f_3 requiring to traverse edges $e_1, g^{'}, f_3^{'}$.

From the inductive hypothesis, initially, there are s packets (called $S - flow$) in the queues $e_0, f_3^{'}$ requiring to traverse edges e_0, g, f_3.

Phase j consists of 3 rounds. The sequence of injections is as follows:

Round 1:

For s steps, the adversary injects in $f_2^{'}$ queue a set X of rs packets wanting to traverse edges $f_2^{'}, e_0, f_1, f_3, e_1, g^{'}, f_3^{'}$. These packets are blocked by the $S - flow$ in queue e_0 because e_0 has LIS as queueing discipline and because for every arrival of an injected packet in e_0 there is at least one $S - flow$ packet there.

At the same time, the $S - flow$ is delayed by the adversary's single injections $S_1 = rs$ in queue g that require to traverse only queue g. This happens because g follows the SIS policy and S_1 is earlier than S in the system. Note that because of SIS, when a packet from $S_1 - flow$ is injected at g, it has priority over all packets from $S - flow$ that are queued at g. Thus, all the S_1 packets traverse g and some packets from $S - flow$ traverse g. Their number is $s - rs$. Therefore, at the end of this round there are rs packets of S in queue g.

Round 2:

For the next rs steps, the adversary injects a set Y of $r^2 s$ packets in queue $f_2^{'}$ requiring to traverse edges $f_2^{'}, e_0, f_2, e_1, g^{'}, f_3^{'}$. These packets are blocked by the set X in queue e_0 because e_0 uses LIS policy and the set X has arrived earlier in queue e_0.

At the same time, all X packets traverse e_0, f_1 but they are blocked in f_3 that uses LIS policy because of the remaining rs packets of $S - flow$ in g at the end of the previous round, that want to traverse f_3.

Round 3:

For the next $T = r^2 s$ steps, the adversary injects a set Z of $r^3 s$ packets requiring to traverse edges $e_1, g^{'}, f_3^{'}$. Moreover, the Y packets reach queue e_1 because they are not blocked by X packets any more. Also, $X_{e_1} = r^2 s$ packets of X reach queue e_1 traversing f_3, while $X_{f_3} = rs - r^2 s$ X packets remain in queue f_3.

At the end of this round the number of packets queued in queues f_3, e_1 requiring to traverse edges $e_1, g^{'}, f_3^{'}$ is

$$s^{'} = X_{e_1} + X_{f_3} + Y + Z - T = r^2 s + rs - r^2 s + r^2 s + r^3 s - r^2 s = rs + r^3 s$$

In order to have instability, we must have $s^{'} > s$. Therefore, we should have $rs + r^3 s > s$, i.e. $r \geq 0.683$. This concludes our proof.

□

2.2 Mixing of LIS and NTS is Unstable

Theorem 2. *Let $r \geq 0.683$. There is a network G that uses LIS and NTS as queueing disciplines and an adversary A of rate r, such that the (G, A, LIS, NTS) system is unstable, starting from a non-empty initial configuration.*

Proof. (See appendix)

2.3 Mixing of LIS and FTG is Unstable

Theorem 3. *Let $r \geq 0.683$. There is a network G that uses LIS and FTG as queueing disciplines and an adversary A of rate r, such that the (G, A, LIS, FTG) system is unstable, starting from a non-empty initial configuration.*

Proof. An interesting proof is presented in the appendix. In this proof we consider the network in Figure 3.

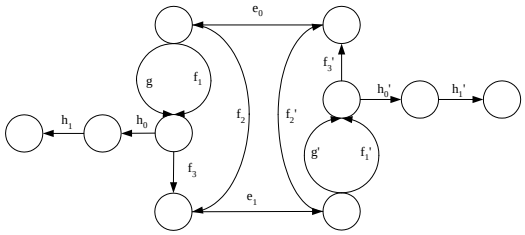

Fig. 3. A Network that uses LIS and FTG as queueing disciplines.

3 Universally Stable Combinations of Universally Stable Policies

Let $0 < \epsilon < 1$ be a real number. We assume that $r = 1 - \epsilon$, m is the number of network edges, and d is the length of the longest simple directed path in the network. Our techniques here are motivated by analogous techniques in Andrews et al. [2].

3.1 Mixing of SIS and FTG is Universally Stable

Lemma 2. *Let p be a packet waiting in a queue e at time t and suppose there are currently $k-1$ other packets in the system requiring e that have priority over p. Then p will cross e within the next $\frac{k+b}{\epsilon}$ steps if the queueing discipline in queue e is SIS or FTG.*

Proof. Let's assume that p does not cross e in the next $\frac{k+b}{\epsilon}$ steps from the moment it arrives in queue e. It will be proved that this assumption results in a contradiction. In order for packet p not to cross queue e in the next $\frac{k+b}{\epsilon}$ steps, other packets cross e in these steps (one distinct packet crosses queue e in each step because of the greedy nature of the protocol). These packets must either belong to the set of $k-1$ packets existing in the system at time t requiring edge e that have priority over p or belong to the (at most) $(1-\epsilon)\frac{k+b}{\epsilon} + b$ packets (from the definition of bounded adversary [2]) requiring queue e that can be injected in the system during the time period of $\frac{k+b}{\epsilon}$ steps.

Note that *all* new packets have priority over p if SIS policy is used. Also, if FTG is used, in the *worst case* all new injections have longer paths to cross than packet p. Therefore at most

$$k - 1 + (1-\epsilon)\frac{k+b}{\epsilon} + b < \frac{k+b}{\epsilon}$$

packets have priority over p during this time period. Hence, there is a contradiction. Thus, p will cross e within the next $\frac{k+b}{\epsilon}$ steps.

□

Let's define a sequence of numbers by the recurrence $k_j = \frac{mk_{j-1}+mb}{\epsilon}$, where $k_1 = \frac{mb}{\epsilon}$.

Lemma 3. *When a packet arrives at an edge e_{d-j+1} that has distance $d-j$ from the final edge on its path, there are at most $k_j - 1$ packets requiring to traverse e_{d-j+1} with priority over p, if the used queueing policy is SIS or FTG.*

Proof. Induction will be used to prove the claim of this lemma. If queue e_{d-j+1} uses SIS as service discipline then the claim holds for $j = 1$, since for any queue $e_{d-j+1} = e_d$ the only packets requiring to traverse it, which initially could have priority over p, are the (at most) $b-1$ packets injected in the same time step as p ($b - 1 \leq k_1 - 1$).

If queue e_{d-j+1} uses FTG as service discipline we will prove that the claim holds for $j = 1$. Let's define as $X_i(t)$ the set of packets in the queue e_{d-j+1} that still have to cross at least i edges at time t and let's assume that l_i is (at most) the number of packets in the system that still have to cross at least i edges ($i = d - j + 1$). Let t be the current time and let $t^{'}$ be the most recent time in which $X_i(t^{'})$ was empty. Any packet in $X_i(t)$ must either have had at least $i+1$ edges to cross at time $t^{'}$ or else it must have been injected after time $t^{'}$. But, at every step $t^{''}$ between times $t^{'}$ and t a packet from $X_i(t^{''})$ must have crossed edge e_{d-j+1}. Hence,

$$|X_i(t)| \leq l_{i+1} + (t - t^{'})(1 - \epsilon) + b - (t - t^{'}) = l_{i+1} - (t - t^{'})\epsilon + b$$

From the above inequality, we conclude that

$$t - t^{'} \leq \frac{l_{i+1} + b}{\epsilon}$$

Hence, because there are m queues the total number of packets in the system that have to cross i or more edges is always at most $\frac{ml_{i+1}+mb}{\epsilon}$.

In the case of $j = 1$ and FTG as used policy, we claim that $k_{j=1} = l_{d-1+1=d}$ because in the queue e_d packet p will have to cross d edges. Therefore, only packets that have to cross $d+1$ edges have priority over p in queue e_d. But, no packet has to cross $d+1$ edges because d is the length of the longest path in the network. So, $l_{d+1} = 0$. Therefore, $\frac{0+mb}{\epsilon} - 1 = k_1 - 1$ packets could have priority over p in queue e_d when FTG is used. Thus, the claim holds for $j = 1$ if FTG is used.

Now suppose that the claim holds for some j. Then by lemma 1, p will arrive at the tail of e_{d-j} on its path in at most another $\frac{k_j+b}{\epsilon}$ steps, during which at most $(1-\epsilon)\frac{k_j+b}{\epsilon} + b$ packets requiring edge e_{d-j} arrive with priority over p. Thus, when p arrives at the tail of e_{d-j} at most

$$k_j - 1 + (1 - \epsilon)\frac{k_j + b}{\epsilon} + b = \frac{k_j + b}{\epsilon} - 1 \leq \frac{mk_j + mb}{\epsilon} - 1 = k_{j+1} - 1$$

packets requiring queue e_{d-j} have priority over p and hence the claim holds.

□

Theorem 4. *The system (G, A, SIS, FTG) is stable, no queue ever contains more than k_d packets and no packet spends more than $\frac{1}{\epsilon}(db + \sum_{i=1}^{d} k_i)$ steps in the system, where d is the length of the longest simple directed path in G.*

Proof. Assume there are $k_d + 1$ packets at some time all requiring the same edge. Then, the packet with the lowest priority of the $k_d + 1$ packets contradicts the claim of lemma 2. Combining both lemmas, a packet p takes at most $\frac{k_j+b}{\epsilon}$ steps to cross the edge e_{d-j+1}, which distance from the final edge on its path is $d - j$, once it is in the queue for this edge. Therefore, the delay bound is $D = \frac{db+\sum_{i=1}^{d} k_i}{\epsilon}$. No packet spends more than D steps in the system.

□

3.2 Mixing of SIS and NTS is Universally Stable

Theorem 5. *The system (G, A, SIS, NTS) is stable, there are never more than k_d packets in the system and no queue contains more than $\frac{1}{\epsilon}(k_{d-1} + b)$ packets in the system where d is the length of the longest simple directed path in G.*

Proof. (See Appendix)

3.3 Mixing of FTG and NTS is Universally Stable

Theorem 6. *The system (G, A, FTG, NTS) is stable, there are never more than k_d packets in the system and no queue contains more than $\frac{1}{\epsilon}(k_{d-1}+b)$ packets in the system where d is the length of the longest simple directed path in G.*

Proof. (See Appendix)

4 Improved Lower Bounds for Instability in FIFO Networks

In [2], Andrews et al. proved the non-stability of FIFO giving a lower bound of 0.85 for the injection rate r. Our team in [11] achieved to slightly lower the injection rate bound for which FIFO is unstable to 0.8357. In this work, we present two adversary constructions that lower the injection rate bound to 0.771 on the network that we consider in Figure 7, and to 0.749 on the network that we consider in Figure 4 for which FIFO is unstable. Furthermore, the techniques that have been applied to achieve these new bounds are presented.

4.1 An Improved Lower Bound Using a New Adversary Construction

The basic technique introduced here is a "broader" inductive hypothesis, according to which we consider initial packets in more than one queues rather than in only one queue in previous approach. Thus, we are able to show bigger delays and get a better instability bound. Note that we additionally impose a restriction on the relative number of initial packets in the two queues, for symmetry reasons.

Theorem 7. *Let $r \geq 0.771$. There is a network G and an adversary A of rate r, such that the (G, A, FIFO) system is unstable, starting from a non-empty initial configuration.*

Proof. (See Appendix)

4.2 A Better Bound by a Sharper Time Analysis of Packet Flow

Theorem 8. *Let $r \geq 0.749$. There is a network G and an adversary A of rate r, such that the (G, A, FIFO) system is unstable, starting from a non-empty initial configuration.*

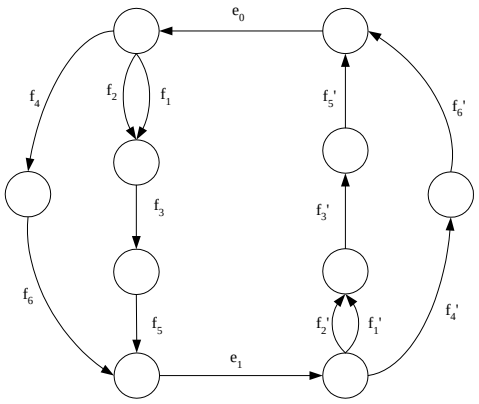

Fig. 4. Network G_1.

Proof. We consider the network G_1 in the above figure.

Inductive Hypothesis: If at the beginning of phase j, there are s packets queued in the queues $e_0, f_3', f_4', f_5', f_6'$ requiring to traverse edges e_0, f_1, f_3, f_5, all these packets manage to depart their initial edges to the symmetric part of the network (f_1, f_3, f_5) as a continuous flow in s time steps, and the

number of packets that are queued in queues $f_4^{'}, f_6^{'}$ is bigger than the number of packets queued in queues $f_3^{'}, f_5^{'}$ then at the beginning of phase $j+1$ there will be more than s packets ($s^{'}$ packets) queued in the queues f_3, f_5, f_4, f_6, e_1 requiring to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$, all of which will be able to depart their initial edges to the symmetric part of the network ($f_1^{'}, f_3^{'}, f_5^{'}$) in $s^{'}$ time steps as a continuous flow and the number of packets that are queued in queues f_4, f_6 is bigger than the number of packets queued in queues f_3, f_5.

Note that the second part of the inductive hypothesis, that claims that if at the beginning of phase j all s packets queued in the queues $e_0, f_3^{'}, f_4^{'}, f_5^{'}, f_6^{'}$ requiring to traverse edges e_0, f_1, f_3, f_5 manage to depart their initial edges to the symmetric part of the network as a continuous flow in s time steps then at the beginning of phase $j+1$ all $s^{'}$ packets that are queued in queues f_3, f_5, f_4, f_6, e_1 requiring to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$ will be able to depart their initial edges to the symmetric part of the network in $s^{'}$ time steps as a continuous flow guarantees the reproduction of the inductive hypothesis in queues f_3, f_5, f_4, f_6, e_1 but with some flows (in particular in queues f_3, f_4, f_5) that have packets that don't want to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$ regularly spread among the packets that want to traverse these edges. This argument implies the third part of the inductive hypothesis that claims that if at the beginning of phase j, the number of packets that are queued in queues $f_4^{'}, f_6^{'}$ is bigger than the number of packets queued in queues $f_3^{'}, f_5^{'}$ then at the beginning of phase $j+1$ the number of packets that are queued in queues f_4, f_6 is bigger than the number of packets queued in queues f_3, f_5 because in the first round of the adversary's construction we inject packets in queue $f_4^{'}$ and if the third part of the inductive hypothesis doesn't hold then we cannot guarantee that all the initial s packets will depart their edges to the edges f_1, f_3, f_5 in s time steps as a continuous flow. However, we include it to the inductive hypothesis for readability reasons.

From the inductive hypothesis, initially, there are s packets (called $S-flow$) in the queues $e_0, f_3^{'}, f_4^{'}, f_5^{'}, f_6^{'}$ requiring to traverse edges e_0, f_1, f_3, f_5.

Phase j consists of 3 rounds. The sequence of injections is as follows:

Round 1:

For s steps, the adversary injects in $f_4^{'}$ queue a set X of rs packets wanting to traverse edges $f_4^{'}, f_6^{'}, e_0, f_2, f_3, f_5, e_1, f_1^{'}, f_3^{'}, f_5^{'}$. These packets are blocked by the $S-flow$.

At the same time, the $S-flow$ is delayed by the adversary's single injections $S_1 = rs$ in queue f_1. The S_1 packets get mixed with the packets in the $S-flow$.

Notice that because of the FIFO policy, the packets of S, S_1 mix in consecutive blocks according to their initial proportion of their sizes (fair mixing property). Since $|S| = s$ and $|S_1| = rs$, these proportions are $\frac{1}{r+1}$ and $\frac{r}{r+1}$, respectively. Thus, during the s steps of round 1, the packets of S, S_1, which cross f_1 are, respectively, $s\frac{1}{r+1} = \frac{s}{r+1}$ and $s\frac{r}{r+1} = \frac{rs}{r+1}$.

Therefore, the remaining packets of each type in queue f_1 are for S_{rem}: $s - \frac{s}{r+1} = \frac{rs}{r+1}$ and for $S_{1,rem}$: $rs - \frac{rs}{r+1} = \frac{r^2 s}{r+1}$

Round 2:

For the next rs steps, the adversary injects a set Y of $r^2 s$ packets requiring edges $f_4^{'}, f_6^{'}, e_0, f_4, f_6, e_1, f_1^{'}, f_3^{'}, f_5^{'}$. These packets are blocked by the set X. At the same time, the adversary pushes a set S_2 of single injections in the queue f_2, where $|S_2| = r^2 s$, a set S_3 of single injections in the queue f_3, where $|S_3| = r^2 s$ and a set S_4 of single injections in the queue f_5, where $|S_4| = r^2 s$.

Because of the FIFO policy, the packets of X, S_2 mix in consecutive blocks according to their initial proportion of their sizes. Since $|X| = rs$ and $|S_2| = r^2 s$, these proportions are $\frac{1}{r+1}$ and $\frac{r}{r+1}$, respectively. Thus, during the rs steps of this round, the packets of X, S_2, that pass f_2 are, respectively, $X_{pass,f_2} = \frac{rs}{r+1}$, and $S_{2,pass,f_2} = \frac{r^2 s}{r+1}$.

Therefore, the remaining packets of each type are for X_{rem,f_2}: $rs - \frac{rs}{r+1} = \frac{r^2 s}{r+1}$ and for S_{2,rem,f_2}: $r^2 s - \frac{r^2 s}{r+1} = \frac{r^3 s}{r+1}$

Note that in queue f_1, there are the remaining $S-flow$ and the remaining S_1-flow packets. Since their total number is rs (which is equal to the duration of the round), the $S_{1,rem}-flow$ does not delay the $S_{rem}-flow$. Note also that, because the $S_{1,rem}$ packets are absorbed after they pass only f_1, only the S_{rem} packets require edge f_3. As a result the stream arriving from f_1 to f_3 contains empty spaces at the positions of the $S_{1,rem}$ packets. However, these empty spaces are uniformly spread in the system for the duration of the time period. Therefore, during round 2, three different flows of packets arrive to the f_3 queue:

- the $X_{pass,f_2}-flow$, where $|X_{pass,f_2}| = \frac{rs}{r+1}$. This flow is mixed with $S_{2,pass,f_2}-flow$. However, since their total number is rs (that is equal to the duration of the round), $S_{2,pass,f_2}-flow$ does not delay

the $X_{pass,f_2} - flow$. Note also that, because the $S_{2,pass,f_2} - flow$ is absorbed after it passes f_2, only the $X_{pass,f_2} - flow$ requires edge f_3. As a result the stream arriving from f_2 to f_3 contains empty spaces at the positions of the $S_{2,pass,f_2}$ packets.
- the $S_{rem} - flow$, where $|S_{rem}| = \frac{rs}{r+1}$.
- the S_3 single-injected packets, where $|S_3| = r^2 s$.

Since the total number of packets in the three flows is $|T| = \frac{r^3 s + r^2 s + 2rs}{r+1}$ the corresponding proportions are:

for X_{pass,f_2}: $\frac{X_{pass,f_2}}{T} = \frac{1}{r^2+r+2}$, for S_{rem}: $\frac{S_{rem}}{T} = \frac{1}{r^2+r+2}$, and for S_3: $\frac{S_3}{T} = \frac{r^2+r}{r^2+r+2}$

Thus, the remaining packets in queue f_3 from each flow at the end of round 2 are: for X_{rem,f_3}: $\frac{rs}{r+1} - \frac{rs}{r^2+r+2} = \frac{r^3 s + rs}{(r+1)(r^2+r+2)}$, for S_{rem,f_3}: $\frac{rs}{r+1} - \frac{rs}{r^2+r+2} = \frac{r^3 s + rs}{(r+1)(r^2+r+2)}$ and for $S_{3,rem}$: $r^2 s - rs\frac{r^2+r}{r^2+r+2} = \frac{r^4 s + r^2 s}{r^2+r+2}$.

The technique of proportions can still be used even if some flow has empty spaces since the empty spaces are uniformly spread.

Note that during round 2 the stream arriving to f_5 contains 3 different flows of packets: the S_4 single-injected packets, where $|S_4| = r^2 s$, the $S_{pass,f_3} - flow$, where $|S_{pass,f_3}| = \frac{rs}{r^2+r+2}$ and the $X_{pass,f_3} - flow$, where $|X_{pass,f_3}| = \frac{rs}{r^2+r+2}$.

Note also that the S_3 packets that traverse f_3 are absorbed after they pass f_3.

Since the total number of packets in the three flows is $|T^{'}| = \frac{r^4 s + r^3 s + 2r^2 s + 2rs}{r^2+r+2}$ the corresponding proportions are:

- for X_{pass,f_3}: $\frac{X_{pass,f_3}}{T'} = \frac{rs}{r^4 s + r^3 s + 2r^2 s + 2rs} = \frac{1}{r^3+r^2+2r+2}$
- for S_{pass,f_3}: $\frac{S_{pass,f_3}}{T'} = \frac{1}{r^3+r^2+2r+2}$ and
- for S_4: $\frac{S_4}{T'} = \frac{r^2 s(r^2+r+2)}{r^4 s + r^3 s + 2r^2 s + 2rs} = \frac{r^3+r^2+2r}{r^3+r^2+2r+2}$

Thus, the remaining packets in f_5 queue from each flow at the end of round 2 are:

- for X_{pass,f_3}: $\frac{rs}{r^2+r+2} - \frac{rs}{r^3+r^2+2r+2} = \frac{r^4 s + r^2 s}{(r^2+r+2)(r^3+r^2+2r+2)}$
- for S_{pass,f_3}: $\frac{rs}{r^2+r+2} - \frac{rs}{r^3+r^2+2r+2} = \frac{r^4 s + r^2 s}{(r^2+r+2)(r^3+r^2+2r+2)}$
- for S_4: $r^2 s - \frac{r^4 s + r^3 s + 2r^2 s}{r^3+r^2+2r+2} = \frac{r^5 s + r^3 s}{r^3+r^2+2r+2}$

Round 3:

For the next $|T_{round-3}| = r^2 s$ steps, the adversary injects a set S_5 of $r^3 s$ packets requiring to traverse edge f_4 (single injections). The set S_5 is mixed with the set Y in consecutive blocks according to their initial proportion of their sizes. These proportions are $Y : \frac{1}{r+1}$ and $S_5 : \frac{r}{r+1}$. Thus, during the $r^2 s$ steps of this round, the packets Y, S_5 that remain in queue f_4 are respectively, $|Y_{rem}| = r^2 s - \frac{r^2 s}{r+1} = \frac{r^3 s}{r+1}$ and $|S_{5,rem}| = r^3 s - \frac{r^3 s}{r+1} = \frac{r^4 s}{r+1}$.

Furthermore, the adversary injects a set Z of $r^3 s$ packets requiring to traverse edges $f_6, e_1, f_1^{'}, f_3^{'}, f_5^{'}$. The set Z is mixed with the set Y_{pass} in consecutive blocks according to their initial proportion of their sizes. These proportions are Y_{pass}: $\frac{\frac{r^2 s}{r+1}}{\frac{r^2 s}{r+1} + r^3 s} = \frac{1}{r^2+r+1}$ and Z: $\frac{r^3 s}{r^3 s + \frac{r^2 s}{r+1}} = \frac{r(r+1)}{r^2+r+1}$.

Thus, during the $r^2 s$ steps of this round, the packets that pass f_6 are respectively, $|Y_{pass,f_6}| = \frac{r^2 s}{r^2+r+1}$ and $|Z_{pass,f_6}| = \frac{r^3 s(r+1)}{r^2+r+1}$. Therefore, the remaining packets in f_6 queue are:

$|Y_{rem,f_6}| = \frac{r^2 s}{r+1} - \frac{r^2 s}{r^2+r+1} = \frac{r^4 s}{(r+1)(r^2+r+1)}$ and $|Z_{rem,f_6}| = r^3 s - \frac{r^3 s(r+1)}{r^2+r+1} = \frac{r^5 s}{r^2+r+1}$.

Note that the total number of packets in queue f_5 at the end of round 2 is $|T_1| = \frac{r^7 s + r^6 s + 3r^5 s + 3r^4 s + 2r^3 s + 2r^2 s}{(r^2+r+2)(r^3+r^2+2r+2)}$. However, it has been proved by MATLAB that $T_1 < r^2 s$, $\forall r > 0$. Therefore, the remaining time is

$t_{rem} = r^2 s - T_1 = \frac{r^6 s + 2r^5 s + 3r^4 s + 4r^3 s + 2r^2 s}{r^5 + 2r^4 + 5r^3 + 6r^2 + 6r + 4}$

In t_{rem} steps the number of X_{rem,f_3} packets that traverses f_5, that is equal with the number of S_{rem,f_3} packets that absorbed when they pass f_5 is

$|X_{pass,f_3,f_5}| = |S_{absorb,f_3,f_5}| = \frac{t_{rem}}{r^2+r+2} = \frac{r^6 s + 2r^5 s + 3r^4 s + 4r^3 s + 2r^2 s}{(r^2+r+2)(r^5+2r^4+5r^3+6r^2+6r+4)}$.

Note that the total number of packets in queue f_3 at the beginning of round 3 is $|T_2| = \frac{r^5 s + r^4 s + 3r^3 s + r^2 s + 2rs}{(r+1)(r^2+r+2)}$. However, $r^2 s < |T_2|$, $\forall r < 1$ (it has been proved by MATLAB). Thus, a number of $X_{rem,f_3}, S_{rem,f_3}, S_{3,rem}$

packets remain in f_3. This number is $|T_3| = |T_2| - r^2 s = \frac{2rs - r^2 s - r^4 s}{(r+1)(r^2+r+2)}$. From this number of packets, the number of packets that belong to $S_{3,rem}$ is $|S_{3,rem,f_3}| = |T_3| \frac{r^2+r}{r^2+r+2} = \frac{2r^2 s - r^3 s - r^5 s}{(r^2+r+2)^2}$.

Also, the total number of packets that are in queue f_2 at the end of round 2 is $|T_3| = r^2 s$. Thus, all the $X - flow$ packets in queue f_2 are queued in queue f_3.

At the end of this round the number of packets that are in queues f_3, f_4, f_5, f_6, e_1 requiring to traverse the edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$ is

$s^{'} = r^3 s + r^2 s + \frac{r^2 s}{r+1} + \frac{r^3 s + rs}{(r+1)(r^2+r+2)} + \frac{r^4 s + r^2 s}{(r^2+r+2)(r^3+r^2+2r+2)} - r^2 s$

In order to have instability, we must have $s^{'} > s$. Therefore,

$r^3 s + \frac{r^2 s}{r+1} + \frac{r^3 s + rs}{(r+1)(r^2+r+2)} + \frac{r^4 s + r^2 s}{(r^2+r+2)(r^3+r^2+2r+2)} > s \Longrightarrow r \geq 0.749$

The above inequality has been proved by MATLAB.

This concludes the first part of the proof. In order to conclude the proof we should also show that the number of packets that remain in queues f_3, f_5 $(Q(f_3), Q(f_5))$ should be less than the number of packets that remain in queues f_4, f_6 $(Q(f_4), Q(f_6))$, i.e. $Q(f_3) + Q(f_5) \leq Q(f_4) + Q(f_6)$. But,

$Q(f_3) + Q(f_5) = |X_{rem,f_2}| + (|X_{rem,f_3}| - |X_{pass,f_3,f_5}|) + (|S_{rem,f_3}| - |S_{absorb,f_3,f_5}|) + |S_{3,rem,f_3}|$ and

$Q(f_4) + Q(f_6) = (|Y_{rem}| + |S_{5,rem}|) + (|Y_{rem,f_6}| + |Z_{rem,f_6}|)$

Assigning the appropriate values to the above equations we take:

$Q(f_3) + Q(f_5) = \frac{r^2 s}{r+1} + 2(\frac{r^3 s + rs}{(r+1)(r^2+r+2)} - \frac{r^6 s + 2r^5 s + 3r^4 s + 4r^3 s + 2r^2 s}{(r^2+r+2)(r^5+2r^4+5r^3+6r^2+6r+4)}) + \frac{2r^2 s - r^3 s - r^5 s}{(r^2+r+2)^2}$ and

$Q(f_4) + Q(f_6) = r^3 s + \frac{r^6 s + r^5 s + r^4 s}{(r+1)(r^2+r+1)}$

Using MATLAB it has been proved that $Q(f_3) + Q(f_5) \leq Q(f_4) + Q(f_6) \Longrightarrow r \geq 0.743$. If this constraint holds, the third part of the inductive hypothesis is fulfilled.

Notice that we have, till now, managed to reproduce the inductive hypothesis in queues f_3, f_5, f_4, f_6, e_1 but with some flows (in particular in queues f_4, f_3, f_5) having empty spaces (packets that don't want to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$). In order for the induction step to work we must show that all these packets in these queues will manage to depart to the symmetric part of the network $(f_1^{'}, f_3^{'}, f_5^{'})$ in the $s^{'}$ time steps as a continuous flow. We now show this.

As we have shown above all the packets that are queued in e_1 queue want to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$ and their flow is continuous without empty spaces (packets that don't want to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$). Also, the number of packets queued in f_4, f_6 queues is bigger than the number of packets queued in f_3, f_5 queues. The packets queued in f_6 queue can be seen as a continuous flow that wants to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$, while the set of packets in f_4 queue consist of packets that want to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$ (Y_{rem}) and packets that are single injections $(S_{5,rem})$ that can be considered as empty spaces. Because of that we should show that all the (Y_{rem}) packets manage to leave their initial edge during $s^{'}$ time steps.

In order to show that, we should estimate the number of Y_{rem} packets that manage to traverse queue f_4 within the time steps the initial packets in queue f_6 need to traverse edges f_6, e_1. Then, if we show that this number is bigger than the number of $Y_{rem}, S_{5,rem}$ packets that remain in f_4 queue we prove that actually the $S_{5,rem}$ packets do not delay the Y_{rem} packets. Therefore, with this way it is proved that in $s^{'}$ time steps all the $s^{'}$ packets that want to traverse edges $e_1, f_1^{'}, f_3^{'}, f_5^{'}$ leave their initial edges to $f_1^{'}, f_3^{'}, f_5^{'}$.

The number of time steps needed for all the initial packets in queue f_6 to traverse edges f_6, e_1 is

$$\begin{aligned} |T_4| &= |Z_{rem,f_6}| + |Y_{rem,f_6}| + (|Z_{pass,f_6}| + |Y_{pass,f_6}| + |X_{pass,f_3}| + |X_{pass,f_3,f_5}| - T_{round-3}) \\ &= \frac{r^4 s}{r+1} + \frac{r^4 s + r^2 s}{(r^2+r+2)(r^3+r^2+2r+2)} + \frac{r^6 s + 2r^5 s + 3r^4 s + 4r^3 s + 2r^2 s}{(r^2+r+2)(r^5+2r^4 s+5r^3+6r^2+6r+4)} \end{aligned}$$

In T_4 time steps, the number of Y_{rem} packets that traverse f_4 is $A = |T_4| \frac{1}{r+1}$, and the number of $S_{5,rem}$ packets that traverse f_4 is $B = |T_4| \frac{r}{r+1}$, where $\frac{1}{r+1}, \frac{r}{r+1}$ are the mixing proportions of $Y_{rem}, S_{5,rem}$ correspondingly.

We should show that $A \geq |Y_{rem}| - A + |S_{5,rem}| - B$. It was proved by using MATLAB that this holds for $r \geq 0$.

Taking the maximum of the three values $\max\{0.749, 0.743, 0\} = 0.749$, the network G is unstable for $r \geq 0.749$. This concludes our proof.

□

Acknowledgements. We wish to thank Antonio Fernandez for his useful comments.

References

[1] M. Andrews. Instability of FIFO in session-oriented networks. In 11th. ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), pages 440-447, 2000.

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

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

[4] A. Borodin, J. Kleinberg, M. Sudan, and D. Williamson. Notes. June 1996

[5] M. Bramson. Instability of FIFO queuing networks. Annals of Applied Probability, 4:414-431, 1994.

[6] M. Bramson. Instability of FIFO queueing networks with quick service times. Annals of Applied Probability, 4(3):693-718, 1994.

[7] R. Cruz. A calculus for network delay. Part I: Network elements in isolation. IEEE Transactions on Information Theory, 37:114-131), 1991.

[8] R. Cruz. A calculus for network delay. Part II: Network analysis. IEEE Transactions on Information Theory, 37:132-141, 1991.

[9] D. Gamarnik. Stability of adversarial queues via models. In 39th. IEEE Symposium on Foundations of Computer Science (FOCS 1998), pages 60-76,1998.

[10] A. Goel. Stability of networks and protocols in the adversarial queueing model for packet routing. In the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA 1999) (short abstract), pages 315-324, 1999. Accepted for publication in Networks.

[11] J. Diaz, D. Koukopoulos, S. Nikoletseas, M. Serna, P. Spirakis and D. Thilikos. Stability and non-stability of the FIFO protocol. In 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2001), pages 48-52, Crete, Greece, July 2001.

Appendix

Mixing of LIS and NTS is Unstable

Theorem 2. Let $r \geq 0.683$. There is a network G that uses LIS and NTS as queueing disciplines and an adversary A of rate r, such that the (G, A, LIS, NTS) system is unstable, starting from a non-empty initial configuration.

Proof. Let's consider the network in Figure 5.

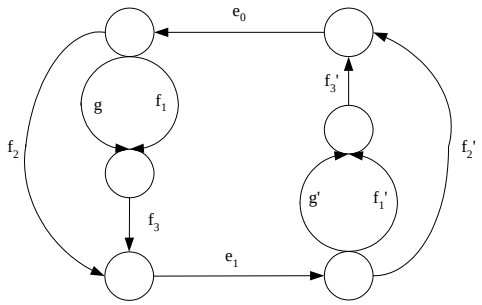

Fig. 5. A Network that uses LIS and NTS as queueing disciplines.

All the queues of this network use the LIS queueing discipline except from the queues that correspond to the edges $g, g^{'}$ that use the NTS queueing discipline.

Inductive Hypothesis: If at the beginning of phase j, there are s packets queued in the queues $e_0, f_3^{'}$ requiring to traverse edges e_0, g, f_3, then at the beginning of phase $j+1$ there will be more than s packets queued in the queues e_1, f_3 requiring to traverse edges $e_1, g^{'}, f_3^{'}$.

From the inductive hypothesis, initially, there are s packets (called $S - flow$) in the queues $e_0, f_3^{'}$ requiring to traverse edges e_0, g, f_3.

Phase j consists of 3 rounds. The sequence of injections is as follows:

Round 1:

For s steps, the adversary injects in $f_2^{'}$ queue a set X of rs packets wanting to traverse edges $f_2^{'}, e_0, f_1, f_3, e_1, g^{'}, f_3^{'}$. These packets are blocked by the $S - flow$ in queue e_0 because e_0 has LIS as queueing discipline and because for every arrival of an injected packet in e_0 there is at least one $S - flow$ packet there.

At the same time, the $S - flow$ is delayed by the adversary's single injection $S_1 = rs$ in queue g that require to traverse queue g. This happens because g follows the NTS policy and S_1 is nearest to its source than S. Thus, at the end of this round rs packets of S will remain in queue g because all the S_1 packets pass as they arrive in queue g and the size of S_1 is rs.

Round 2:

For the next rs steps, the adversary injects a set Y of r^2s packets in queue $f_2^{'}$ requiring to traverse edges $f_2^{'}, e_0, f_2, e_1, g^{'}, f_3^{'}$. These packets are blocked by the set X in queue e_0 because e_0 uses LIS policy and the set X has arrived earlier in queue e_0.

At the same time, all X packets traverse e_0, f_1 but they are blocked in f_3 that uses LIS policy because of the remaining rs packets of $S - flow$ in g that want to traverse f_3 and are longer time in the system than X.

Round 3:

For the next $T = r^2s$ steps, the adversary injects a set Z of r^3s packets requiring to traverse edges $e_1, g^{'}, f_3^{'}$. Moreover, the Y packets reach queue e_1 and $X_{e_1} = r^2s$ packets of X reach queue e_1, too. Also, $X_{f_3} = rs - r^2s$ X packets remain in queue f_3.

At the end of this round the number of packets queued in queues f_3, e_1 requiring to traverse edges $e_1, g, f_3^{'}$ is

$$s^{'} = X_{e_1} + X_{f_3} + Y + Z - T = r^2s + rs - r^2s + r^2s + r^3s - r^2s = rs + r^3s$$

In order to have instability, we must have $s^{'} > s$. Therefore,

$$\begin{aligned} & rs + r^3 s > s \\ \Longrightarrow\ & r \qquad\quad \geq 0.683. \end{aligned}$$

This concludes our proof.

□

Mixing of LIS and FTG is Unstable

Theorem 3. Let $r \geq 0.683$. There is a network G that uses LIS and FTG as queueing disciplines and an adversary A of rate r, such that the (G, A, LIS, FTG) system is unstable, starting from a non-empty initial configuration.

Proof. Let's consider the network in Figure 6.

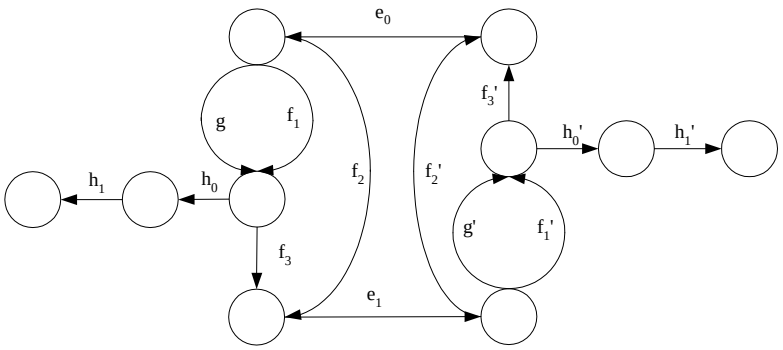

Fig. 6. A Network that uses LIS and FTG as queueing disciplines.

All the queues of this network use the LIS queueing discipline except from the queues that correspond to the edges $g, g^{'}$ that use the FTG queueing discipline.

Inductive Hypothesis: If at the beginning of phase j, there are s packets queued in the queues $e_0, f_3^{'}$ requiring to traverse edges e_0, g, f_3, then at the beginning of phase $j+1$ there will be more than s packets queued in the queues e_1, f_3 requiring to traverse edges $e_1, g^{'}, f_3^{'}$.

From the inductive hypothesis, initially, there are s packets (called $S - flow$) in the queues $e_0, f_3^{'}$ requiring to traverse edges e_0, g, f_3.

Phase j consists of 3 rounds. The sequence of injections is as follows:

Round 1:

For s steps, the adversary injects in $f_2^{'}$ queue a set X of rs packets wanting to traverse edges $f_2^{'}, e_0, f_1, f_3, e_1, g^{'}, f_3^{'}$. These packets are blocked by the $S - flow$ in queue e_0 because e_0 has LIS as queueing discipline and because for every arrival of an injected packet in e_0 there is at least one $S - flow$ packet there.

At the same time, the $S - flow$ is delayed by the adversary's injection $S_1 = rs$ packets in queue g that require to traverse queues g, h_0, h_1. This happens because g follows the FTG policy and S_1 has more edges to traverse than S. Thus, at the end of this round rs packets of S will remain in queue g because all the S_1 packets pass as they arrive in queue g and the size of S_1 is rs.

Round 2:

For the next rs steps, the adversary injects a set Y of $r^2 s$ packets in queue $f_2^{'}$ requiring to traverse edges $f_2^{'}, e_0, f_2, e_1, g^{'}, f_3^{'}$. These packets are blocked by the set X in queue e_0 because e_0 uses LIS policy and the set X has arrived earlier in queue e_0.

At the same time, all X packets traverse e_0, f_1 but they are blocked in f_3 that uses LIS policy because of the remaining rs packets of $S - flow$ in g that want to traverse f_3 and are longer time in the system than X.

Round 3:

For the next $T = r^2 s$ steps, the adversary injects a set Z of $r^3 s$ packets requiring to traverse edges $e_1, g^{'}, f_3^{'}$. Moreover, the Y packets arrive in queue e_1 and $X_{e_1} = r^2 s$ packets of X arrive in queue e_1, too. Also, $X_{f_3} = rs - r^2 s$ X packets remain in queue f_3.

At the end of this round the number of packets queued in queues f_3, e_1 requiring to traverse edges $e_1, g^{'}, f_3^{'}$ is

$$s^{'} = X_{e_1} + X_{f_3} + Y + Z - T = r^2 s + rs - r^2 s + r^2 s + r^3 s - r^2 s = rs + r^3 s$$

In order to have instability, we must have $s^{'} > s$. Therefore,

$$\begin{aligned} & rs + r^3 s > s \\ \Longrightarrow\ & r \qquad\quad \geq 0.683. \end{aligned}$$

This concludes our proof.

□

Mixing of SIS and NTS is Universally Stable

Lemma 4. Let p be a packet waiting in a queue e at time t and suppose there are currently $k-1$ other packets in the system requiring e that have priority over p. Then p will cross e within the next $\frac{k+b}{\epsilon}$ steps if the queueing discipline in queue e is SIS or NTS.

Proof. Let's assume that p does not cross e in the next $\frac{k+b}{\epsilon}$ steps from the moment it arrives in queue e. It will be proved that this assumption results in a contradiction. In order for packet p not to cross queue e in the next $\frac{k+b}{\epsilon}$ steps, other packets cross e in these steps (one distinct packet crosses queue e in each step because of the greedy nature of the protocol). These packets must either belong to the set of $k-1$ packets existing in the system at time t requiring edge e that have priority over p or belong to the (at most) $(1-\epsilon)\frac{k+b}{\epsilon} + b$ packets requiring queue e that can be injected in the system during the time period of $\frac{k+b}{\epsilon}$ steps.

Note that all new packets have priority over p if SIS policy is used. Also, if NTS is used in the worst case new injections are nearest to their source than packet p. Therefore at most

$$k - 1 + (1-\epsilon)\frac{k+b}{\epsilon} + b < \frac{k+b}{\epsilon}$$

packets have priority over p during this time period. Hence, there is a contradiction. Thus, p will cross e within the next $\frac{k+b}{\epsilon}$ steps.

□

Let's define a sequence of numbers by the recurrence $k_j = \frac{mk_{j-1}+mb}{\epsilon}$, where $k_1 = \frac{mb}{\epsilon}$.

Lemma 5. When a packet arrives at the queue of the j^{th} edge e_j on its path, there are at most $k_j - 1$ packets requiring to traverse e_j with priority over p if the used queueing policy is SIS or NTS.

Proof. Induction will be used to prove the claim of this lemma. If queue e_j uses SIS then the claim holds for $j = 1$, since for any queue e_j the only packets requiring to traverse it, which initially could have priority over p, are the (at most) $b-1$ packets injected in the same time step as p ($b - 1 \leq k_1 - 1$).

If queue e_j uses NTS then we will prove that the claim holds for $j = 1$. Let's define $X_i(t)$ the set of packets in the queue e_j that have crossed less than i edges at time t and let's assume that l_i is (at most) the number of packets in the system that have crossed less than i edges (i=j). Let t be the current time and let $t^{'}$ be the most recent time in which $X_i(t^{'})$ was empty. Any packet in $X_i(t)$ must either have had crossed less than $i-1$ edges at time $t^{'}$ or else it must have been injected after time $t^{'}$. But, at every step $t^{''}$ between times $t^{'}$ and t a packet from $X_i(t^{''})$ must have crossed edge e_j. Hence,

$$\begin{aligned} |X_i(t)| &\leq l_{i-1} + (t - t^{'})(1-\epsilon) + b - (t - t^{'}) \\ &= l_{i-1} - (t - t^{'})\epsilon + b \end{aligned}$$

From the above inequality, we conclude that

$$t - t^{'} \leq \frac{l_{i-1} + b}{\epsilon}$$

Hence, because there are m queues the total number of packets in the system that have crossed less than i edges is always at most $\frac{ml_{i-1}+mb}{\epsilon}$. We claim that, at most $\frac{ml_0+mb}{\epsilon} - 1$ packets could have priority

over p in queue e_1 when NTS is used because in the queue e_1 packet p hasn't crossed any edge. Therefore, no packet from the old ones have priority over p in queue e_1. So, $l_0 = 0$. Therefore, $\frac{b}{\epsilon} - 1 \leq k_1 - 1$ packets could have priority over p in queue e_1 when NTS is used. Thus, the claim holds for $j = 1$ if NTS is used.

Now suppose that the claim holds for some j. Then by lemma 3 p will arrive at the tail of e_{j+1} in at most another $\frac{k_j+b}{\epsilon}$ steps, during which at most $(1-\epsilon)\frac{k_j+b}{\epsilon} + b$ packets requiring edge e_{j+1} arrive with priority over p. Thus, when p arrives at the tail of e_{j+1} at most

$$k_j - 1 + (1-\epsilon)\frac{k_j+b}{\epsilon} + b = \frac{k_j+b}{\epsilon} - 1 \leq \frac{mk_j+mb}{\epsilon} - 1 = k_{j+1} - 1$$

packets requiring queue e_{j+1} have priority over p and hence the claim holds.

□

Theorem 5. The system (G, A, SIS, NTS) is stable, there are never more than k_d packets in the system and no queue contains more than $\frac{1}{\epsilon}(k_{d-1}+b)$ packets in the system where d is the length of the longest simple directed path in G.

Proof. Assume there are $k_d + 1$ packets at some time all requiring the same edge. Then, the packet with the lowest priority of the $k_d + 1$ packets contradicts the claim of lemma 4. Combining both lemmas, a packet p takes at most $\frac{k_j+b}{\epsilon}$ steps to cross the j^{th} edge on its path, once it is in the queue for this edge.

□

Mixing of FTG and NTS is Universally Stable

Lemma 6. Let p be a packet waiting in a queue e at time t and suppose there are currently $k-1$ other packets in the system requiring e that have priority over p. Then p will cross e within the next $\frac{k+b}{\epsilon}$ steps if the queueing discipline in queue e is FTG or NTS.

Proof. Let's assume that p does not cross e in the next $\frac{k+b}{\epsilon}$ steps from the moment it arrives in queue e. It will be proved that this assumption results in a contradiction. In order for packet p not to cross queue e in the next $\frac{k+b}{\epsilon}$ steps, other packets cross e in these steps (one distinct packet crosses queue e in each step because of the greedy nature of the protocol). These packets must either belong to the set of $k-1$ packets existing in the system at time t requiring edge e that have priority over p or belong to the (at most) $(1-\epsilon)\frac{k+b}{\epsilon} + b$ packets requiring queue e that can be injected in the system during the time period of $\frac{k+b}{\epsilon}$ steps.

Note that if FTG is used in the worst case new injections have more edges to traverse than packet p. Also, if NTS is used in the worst case new injections are nearest to their source than packet p. Therefore at most

$$k - 1 + (1-\epsilon)\frac{k+b}{\epsilon} + b < \frac{k+b}{\epsilon}$$

packets have priority over p during this time period. Hence, there is a contradiction. Thus, p will cross e within the next $\frac{k+b}{\epsilon}$ steps.

□

Let's define a sequence of numbers by the recurrence $k_j = \frac{mk_{j-1}+mb}{\epsilon}$, where $k_1 = \frac{mb}{\epsilon}$.

Lemma 7. When a packet arrives at an edge e_{d-j+1} that has distance $d-j$ from the final edge on its path, there are at most $k_j - 1$ packets requiring to traverse e_j with priority over p if the used queueing policy is FTG or NTS.

Proof. Induction will be used to prove the claim of this lemma. If queue e_{d-j+1} uses NTS then we will prove that the claim holds for $j = 1$. Let's define $X_i(t)$ the set of packets in the queue e_{d-j+1} that have crossed less than i edges at time t and let's assume that l_i is (at most at any time) the number of packets in the system that have crossed less than i edges ($i = j$). Let t be the current time and let t' be the most recent time in which $X_i(t')$ was empty. Any packet in $X_i(t)$ must either have had crossed less than $i-1$ edges at time t' or else it must have been injected after time t'. But, at every step t'' between times t' and t a packet from $X_i(t'')$ must have crossed edge e_{d-j+1}. Hence,

$$\begin{aligned} |X_i(t)| &\leq l_{i-1} + (t-t')(1-\epsilon) + b - (t-t') \\ &= l_{i-1} - (t-t')\epsilon + b \end{aligned}$$

From the above inequality, we conclude that

$$t - t^{'} \leq \frac{l_{i-1} + b}{\epsilon}$$

Hence, because there are m queues the total number of packets in the system that have crossed less than i edges is always at most $\frac{ml_{i-1}+mb}{\epsilon}$. We claim that, at most $\frac{ml_0+mb}{\epsilon} - 1$ packets could have priority over p in queue $e_{d-j+1} = e_d$, which distance from the final edge on p's path is $d-1$ when NTS is used, because in the queue e_d packet p hasn't crossed any edge. Therefore, no packet from the old ones have priority over p in queue e_d. So, $l_0 = 0$. Therefore, $\frac{b}{\epsilon} - 1 \leq k_1 - 1$ packets could have priority over p in queue e_d when NTS is used. Thus, the claim holds for $j = 1$ if NTS is used.

If queue e_{d-j+1} uses FTG then we will prove that the claim holds for $j = 1$. Let's define as $L_h(t)$ the set of packets in the queue e_{d-j+1} that still have to cross at least h edges at time t and let's assume that l_h is (at most) the number of packets in the system that still have to cross at least h edges ($h = d-j+1$). Let t be the current time and let $t^{'}$ be the most recent time in which $L_h(t^{'})$ was empty. Any packet in $L_h(t)$ must either have had at least $h+1$ edges to cross at time $t^{'}$ or else it must have been injected after time $t^{'}$. But, at every step $t^{''}$ between times $t^{'}$ and t a packet from $L_h(t^{''})$ must have crossed edge e_{d-j+1}. Hence,

$$\begin{aligned} |L_h(t)| &\leq l_{h+1} + (t - t^{'})(1-\epsilon) + b - (t - t^{'}) \\ &= l_{h+1} - (t - t^{'})\epsilon + b \end{aligned}$$

From the above inequality, we conclude that

$$t - t^{'} \leq \frac{l_{h+1} + b}{\epsilon}$$

Hence, the number of packets in the system that have to cross h or more edges is always at most $\frac{ml_{h+1}+mb}{\epsilon}$. In the case of $j = 1$ and FTG as used policy, we claim that $k_{j=1} = l_d$ because in the queue e_d packet p will have to cross d edges. Therefore, only packets that have to cross $d+1$ edges have priority over p in queue e_d. But, no packet have to cross $d+1$ edges because d is the length of the longest path in the network. So, $l_{d+1} = 0$. Therefore, $\frac{0+mb}{\epsilon} - 1 = k_1 - 1$ packets could have priority over p in queue e_1 when FTG is used. Thus, the claim holds for $j = 1$ if FTG is used.

Now suppose that the claim holds for some j. Then by lemma 5 p will arrive at the tail of e_{d-j} in at most another $\frac{k_j+b}{\epsilon}$ steps, during which at most $(1-\epsilon)\frac{k_j+b}{\epsilon} + b$ packets requiring edge e_{d-j} arrive with priority over p. Thus, when p arrives at the tail of e_{d-j} at most

$$k_j - 1 + (1-\epsilon)\frac{k_j + b}{\epsilon} + b = \frac{k_j + b}{\epsilon} - 1 \leq \frac{mk_j + mb}{\epsilon} - 1 = k_{j+1} - 1$$

packets requiring queue e_{d-j} have priority over p and hence the claim holds.

□

Theorem 6. The system (G, A, FTG, NTS) is stable, there are never more than k_d packets in the system and no queue contains more than $\frac{1}{\epsilon}(k_{d-1} + b)$ packets in the system where d is the length of the longest simple directed path in G.

Proof. Assume there are $k_d + 1$ packets at some time all requiring the same edge. Then, the packet with the lowest priority of the $k_d + 1$ packets contradicts the claim of lemma 6. Combining both lemmas, a packet p takes at most $\frac{k_j+b}{\epsilon}$ steps to cross the e_{d-j+1} edge that has distance $d-j$ from the final edge on its path, once it is in the queue for this edge.

□

An Improved Lower Bound Using a New Adversary Construction

Theorem 7. Let $r \geq 0.771$. There is a network G and an adversary A of rate r, such that the (G, A, FIFO) system is unstable, starting from a non-empty initial configuration.

Proof. We consider the network G in Figure 7.

Inductive Hypothesis: If at the beginning of phase j, there are s packets queued in queues $e_0, f_2^{'}, f_3^{'}$ requiring to traverse edges e_0, g, f_2, all the packets in queue $f_3^{'}$ want to traverse edges e_0, g, f_2, and the number of packets queued in queue $f_3^{'}$ is bigger than the number of packets queued in queue $f_2^{'}$, then at

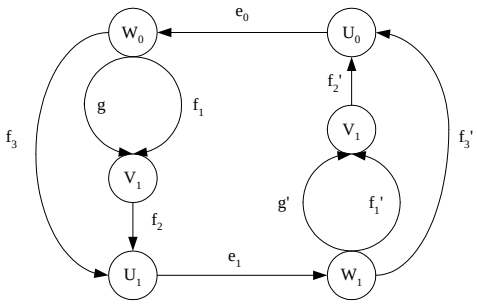

Fig. 7. Network G

the beginning of phase $j+1$ there will be more than s packets queued in queues f_2, f_3, e_1 requiring to traverse edges $e_1, g^{'}, f_2^{'}$, all the packets in queue $f_3^{'}$ will want to traverse edges $e_1, g^{'}, f_2^{'}$, and the number of packets queued in queue f_3 will be bigger than the number of packets queued in queue f_2.

Remark that the inductive hypothesis claim, that if at the beginning of phase j all the packets in queue $f_3^{'}$ want to traverse edges e_0, g, f_2, and the number of packets queued in queue $f_3^{'}$ is bigger than the number of packets queued in queue $f_2^{'}$ then at the beginning of phase $j+1$ all the packets in queue $f_3^{'}$ will want to traverse edges $e_1, g^{'}, f_2^{'}$, and the number of packets queued in queue f_3 will be bigger than the number of packets queued in queue f_2, guarantees that all the packets that want to traverse edges e_0, g, f_2 at the beginning of phase j and all the packets that want to traverse edges $e_1, g^{'}, f_2^{'}$ at the beginning of phase $j+1$ will manage to depart their initial edges to the symmetric part of the network in the s and $s^{'}$ time steps correspondingly as a continuous flow (g, f_2 and $g^{'}, f_2$ correspondingly). Thus, the inductive hypothesis can be reproduced in queues f_2, f_3, e_1 but with a flow (in particular in queue f_2) that has packets that don't want to traverse edges $e_1, g^{'}, f_2^{'}$ regularly spread among the packets that want to traverse these edges.

From the inductive hypothesis, initially, there are s packets (called $S-flow$) in the queues $e_0, f_2^{'}, f_3^{'}$ requiring to traverse edges e_0, g, f_2.

Phase j consists of 3 rounds. The sequence of injections is as follows:

Round 1:

For s steps, the adversary injects in $f_3^{'}$ queue a set X of rs packets wanting to traverse $f_3^{'}, e_0, f_1, f_2, e_1, g^{'}, f_2^{'}$. These packets are blocked by the $S-flow$.

Note that it has been assumed that the number of packets belonging to the $S-flow$ that are queued in $f_3^{'}$ queue at the beginning of this round is bigger than the number of $S-flow$ packets queued in $f_2^{'}$ queue.

At the same time, the $S-flow$ is delayed by the adversary's single injections $S_1 = rs$ in queue g. The S_1 packets get mixed with the packets in the $S-flow$.

Notice that because of the FIFO policy, the packets of S, S_1 mix in consecutive blocks according to their initial proportion of their sizes (fair mixing property). Since $|S| = s$ and $|S_1| = rs$, these proportions are $\frac{1}{r+1}$ and $\frac{r}{r+1}$, respectively. Thus, during the s steps of round 1, the packets of S, S_1, which cross g are, respectively,

$$s\frac{1}{r+1} = \frac{s}{r+1}, s\frac{r}{r+1} = \frac{rs}{r+1}$$

Therefore, the remaining packets of each type are:

- for S_{rem}: $s - \frac{s}{r+1} = \frac{rs}{r+1}$
- for $S_{1,rem}$: $rs - \frac{rs}{r+1} = \frac{r^2 s}{r+1}$

Round 2:

For the next rs steps, the adversary injects a set Y of $r^2 s$ packets requiring edges $f_3^{'}, e_0, f_3, e_1, g^{'}, f_2^{'}$. These packets are blocked by the set X. At the same time, the adversary pushes a set S_2 of single injections in the queue f_1, where $|S_2| = r^2 s$ and a set S_3 of single injections in the queue f_2, where $|S_3| = r^2 s$.

Because of the FIFO policy, the packets of X, S_2 mix in consecutive blocks according to their initial proportion of their sizes. Since $|X| = rs$ and $|S_2| = r^2 s$, these proportions are $\frac{1}{r+1}$ and $\frac{r}{r+1}$, respectively.

Thus, during the rs steps of this round, the packets of X, S_2, that pass f_1 are, respectively,

$$|X_{pass}| = \frac{rs}{r+1}, |S_{2,pass}| = \frac{r^2 s}{r+1}$$

Therefore, the remaining packets of each type are:

- for X_{rem}: $rs - \frac{rs}{r+1} = \frac{r^2 s}{r+1}$
- for $S_{2,rem}$: $r^2 s - \frac{r^2 s}{r+1} = \frac{r^3 s}{r+1}$

Note that in queue g, there are the remaining $S - flow$ and the remaining $S_1 - flow$ packets. Since their total number is rs (which is equal to the duration of the round), the $S_{1,rem} - flow$ does not delay the $S_{rem} - flow$. Note also that, because the $S_{1,rem}$ packets are absorbed after they pass only g, only the S_{rem} packets require edge f_2. As a result the stream arriving from g to f_2 contains empty spaces at the positions of the $S_{1,rem}$ packets. Therefore, during round 2, three different flows of packets arrive to the f_2 queue:

- the $X_{pass} - flow$, where $|X_{pass}| = \frac{rs}{r+1}$. This flow is mixed with $S_{2,pass} - flow$. However, since their total number is rs (that is equal to the duration of the round), $S_{2,pass} - flow$ does not delay the $X_{pass} - flow$. Note also that, because the $S_{2,pass} - flow$ is absorbed after they pass edge f_1, only the $X_{pass} - flow$ requires edge f_2. As a result the stream arriving from f_1 to f_2 contains empty spaces at the positions of the $S_{2,pass}$ packets.
- the $S_{rem} - flow$, where $|S_{rem}| = \frac{rs}{r+1}$.
- the S_3 single-injected packets, where $|S_3| = r^2 s$.

Since the total number of packets in the three flows is:

$$|T| = \frac{r^3 s + r^2 s + 2rs}{r+1}$$

the corresponding proportions are:

- for X_{pass}: $\frac{X_{pass}}{T} = \frac{1}{r^2+r+2}$
- for S_{rem}: $\frac{S_{rem}}{T} = \frac{1}{r^2+r+2}$
- for S_3: $\frac{S_3}{T} = \frac{r^2+r}{r^2+r+2}$

Thus, the remaining packets in f_2 queue from each flow at the end of round 2 are:

- for X_{pass}: $\frac{rs}{r+1} - \frac{rs}{r^2+r+2} = rs\frac{r^2+1}{(r+1)(r^2+r+2)}$
- for S_{rem}: $\frac{rs}{r+1} - \frac{rs}{r^2+r+2} = rs\frac{r^2+1}{(r+1)(r^2+r+2)}$
- for S_3: $r^2 s - rs\frac{r^2+r}{r^2+r+2} = rs\frac{r^3+r}{r^2+r+2}$

Round 3:

For the next $r^2 s$ steps, the adversary injects a set Z of $r^3 s$ packets requiring edges f_3, e_1, g', f_2'. The set Z is mixed with the set Y in consecutive blocks according to their initial proportion of their sizes. These proportions are $Y : \frac{1}{r+1}$ and $Z : \frac{r}{r+1}$. Thus, during the $r^2 s$ steps of this round, the packets Y, Z that pass f_3 are respectively,

$|Y_{pass}| = r^2 s\frac{1}{r+1} = \frac{r^2 s}{r+1}$

$|Z_{pass}| = r^2 s\frac{r}{r+1} = \frac{r^3 s}{r+1}$

Therefore, the remaining packets in f_3 queue are:

- for Y_{rem}: $r^2 s - \frac{r^2 s}{r+1} = \frac{r^3 s}{r+1}$
- for Z_{rem}: $r^3 s - \frac{r^3 s}{r+1} = \frac{r^4 s}{r+1}$

Notice that all the packets in queue f_3 share the same destination path, which fulfills a part of the inductive hypothesis.

During this period the number of $X_{pass} - flow$ packets that traverses f_2 is

$$r^2 s\frac{1}{r^2+r+2}$$

Thus, the remaining $X_{pass} - flow$ and S_{rem} packets that are still in f_2 queue at the end of this round are:

$$|S_{rem,f_2}| = |X_{pass,f_2}| = \frac{r^3 s + rs}{(r+1)(r^2+r+2)} - \frac{r^2 s}{r^2+r+2} = \frac{rs - r^2 s}{(r+1)(r^2+r+2)}$$

Also, the remaining $S_3 - flow$ packets that remain in f_2 queue at the end of this round are:

$$|S_{3,rem}| = \frac{r^4 s + r^2 s}{r^2+r+2} - \frac{r^4 s + r^3 s}{r^2+r+2} = \frac{r^2 s - r^3 s}{r^2+r+2}$$

Furthermore, all the X_{rem} packets that are queued in f_1 at the beginning of this round traverse f_1 and are queued in f_2 because the total size of packets in f_1 is

$$|X_{rem}| + |S_{2,rem}| = \frac{r^2 s}{r+1} + \frac{r^3 s}{r+1} = r^2 s$$

which is equal to the duration of this round.

From the inductive hypothesis, the assumption that the number of packets requiring to traverse edges $f_3, e_1, g^{'}, f_2^{'}$ is bigger than the number of packets requiring to traverse edges $f_2, e_1, g^{'}, f_2^{'}$ should be hold.

However in queue f_2, there is a number of S_{rem,f_2} and $S_{3,rem}$ packets at the end of this round, that are mixed with X_{pass,f_2} packets, while the X_{rem} packets are queued after S_{rem}, $S_{3,rem}$ and X_{pass,f_2} packets in queue f_2. Because of this mixture X_{pass,f_2} packets are delayed in the next phase. So, we should take them into account for the following comparison in order to guarantee that the first part of the inductive hypothesis is met.

$$\begin{array}{ll} & Q(f_3) > Q(f_2) \\ \Longrightarrow & \frac{r^3 s}{r+1} + \frac{r^4 s}{r+1} > 2\frac{rs - r^2 s}{(r+1)(r^2+r+2)} + \frac{r^2 s}{r+1} + \frac{r^2 s - r^3 s}{r^2+r+2} \\ \Longrightarrow & r \geq 0.755 \end{array}$$

where $Q(f_3)$ and $Q(f_2)$ are the number of packets in f_3 and f_2 queues respectively. The above inequality has been proved using MATLAB.

Thus, for $r \geq 0.755$, we have proved the part of the *inductive hypothesis*, which argues that if at the beginning of phase j, all the packets in queue $f_3^{'}$ want to traverse edges e_0, g, f_2 and the number of packets queued in queue $f_3^{'}$ is bigger than the number of packets queued in queue $f_2^{'}$, then at the beginning of phase $j+1$, all the packets in queue f_3 will want to traverse edges $e_0, g^{'}, f_2^{'}$ and the number of packets queued in f_3 queue is bigger than the number of packets queued in f_2 queue.

Now, we will find the appropriate lower bound for injection rate in order to prove that the second part of the *inductive hypothesis*, which argues that the $s^{'}$ packets queued in the queues f_2, f_3, e_1 requiring to traverse edges $e_1, g^{'}, f_2^{'}$ at the beginning of phase $j+1$ are more than the s packets queued in the queues $e_0, f_2^{'}, f_3^{'}$ requiring to traverse edges e_0, g, f_2 at the beginning of phase j, holds on.

At the end of the round, the number of packets that are in queues f_2, f_3, e_1 requiring to traverse edges $e_1, g^{'}, f_2^{'}$ is:

$$s^{'} = r^3 s + r^2 s + \frac{r^2 s}{r+1} + \frac{r^3 s + rs}{(r+1)(r^2+r+2)} - r^2 s$$

Note that in the above estimations, $s^{'}$ counts only packets from X, Y, Z flows that remain at the system at the end of round 3 and not packets from $S - flow$ and single injections. In fact, we can ignore packets causing empty spaces in the stream of f_2 queue, because the stream of f_3 queue is continuous and has bigger size than the stream of f_2 queue, even if empty spaces are included, while the packets queued at e_1 at the end of this round form a stream that is continuous.

In order to have instability $s^{'} > s$ should be hold.

Therefore,

$$\begin{array}{ll} & r^3 s + \frac{r^2 s}{r+1} + \frac{r^3 s + rs}{(r+1)(r^2+r+2)} > s \\ \Longrightarrow & r^6 + 2r^5 + 4r^4 + 3r^3 > 2r + 2 \end{array}$$

The above inequality has as a result $r \geq 0.771$ (it has been proved by using MATLAB). Thus, in order to fulfill the *inductive hypothesis*, we take the maximum of 0.771 and 0.755. Therefore, for $r \geq 0.771$ the network in Figure 7 is unstable. This concludes our proof.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'