Alcom-FT Technical Report Series
ALCOMFT-TR-02-100

Stability Issues in Heterogeneous and FIFO Networks under the Adversarial Queueing Model

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 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, 10], by using new techniques for adversary construction and tight analysis of the packet flow time evolution. We also show a graph for which FIFO is stable for any adversary with injection rate $r \leq 0.1428$, and, by generalizing, we present upper bounds for stability of any network under the FIFO protocol, answering partially an open question raised by Andrews et al. in [2]. The work presented here combines new and recent results of the authors.

1 Introduction, State-of-the-art and Our Results

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 stability problem has been investigated under various models of packet routing and in a number of overlapping areas, see for example [6, 7, 4, 5, 8, 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

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

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.

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 definition.

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. 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. 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, while it has been shown that FIFO, Last-In-First-Out (LIFO), Nearest-to-Go (NTG) and Farthest-from-Source (FFS) policies can be unstable. FIFO is proved unstable for a particular network for $r \geq 0.85$. Later, Goel [9], in a very interesting work, presented an even simpler network of only 3 queues for which FIFO is unstable (see Figure 1).

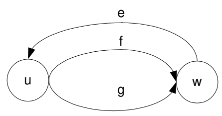

Fig. 1. Goel's network, N

In [10], 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 [10] by showing for the first time upper bounds on r for the stability of FIFO.

Summary of results: In this work,

1. 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, 10] FIFO instability lower bound by presenting a FIFO network where $r \geq 0.749$ leads to instability.
5. 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.
6. We present here upper bounds on r for stability in FIFO networks. A first version of this result has appeared in [10].

Note that our results on combinations of queueing disciplines 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.

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 a lemma presented in Andrews et al. [2].

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 FTG is Unstable

Theorem 1. *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 2.

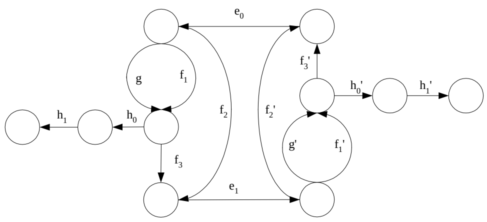

Fig. 2. A Network that uses LIS and FTG as queueing disciplines.

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^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 arrive in queue e_1 and $X_{e_1} = r^2s$ packets of X arrive in 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, $rs + r^3s > s \Rightarrow r \geq 0.683$. □

2.2 Mixing of LIS and SIS, LIS and NTS is Unstable

Theorem 2. *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. See full paper in [11]. □

Theorem 3. *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 full paper in [11]. □

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 FTG and NTS is Universally Stable

Lemma 1. *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 2. *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 4. *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. □

3.2 Mixing of SIS and FTG, SIS and NTS is Universally Stable

Theorem 5. *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. See full paper ([11]). □

Theorem 6. *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 full paper ([11]). □

4 An Improved Lower Bound for Instability in FIFO Networks

We present a new adversary construction that significantly lowers the best previous known FIFO instability injection rate bounds to 0.749.

Theorem 7. *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.*

Proof. (Sketch, for details see [11]).

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 $e_0, f_1, f_3, f_5,$

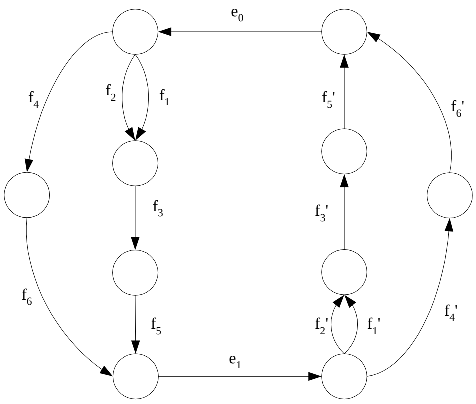

Fig. 3. Network G_1.

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.

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 $|S_4| = r^2 s$.

By the proportions of the packet stream sizes, we can estimamte the remaining packets of each type, X_{rem,f_2} and S_{2,rem,f_2}.

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 $X_{pass,f_2} - flow$, which mixes with $S_{2,pass,f_2} - flow$, the $S_{rem} - flow$, and the S_3 single-injected packets.

The total number of packets in the three flows is $|T| = \frac{r^3 s + r^2 s + 2rs}{r+1}$. We can estimate the remaining packets in queue f_3 from each flow at the end of round 2. Note that during round 2 the stream arriving to f_5 contains 3 different flows of packets: the S_4 single-injected packets, the $S_{pass,f_3} - flow$ and the $X_{pass,f_3} - flow$. Note also that the S_3 packets that traverse f_3 are absorbed after they pass f_3.

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 remaining packets in f_5 queue from each flow at the end of round 2 are: X_{pass,f_3}, S_{pass,f_3} and S_4.

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. Call the packets Y, S_5 that remain in queue f_4 respectively, Y_{rem} and $S_{5,rem}$.

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}. The remaining packets in f_6 queue are: Y_{rem,f_6} and Z_{rem,f_6}.

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 $t_{rem} = r^2 s - T_1$. 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 $|X_{pass,f_3,f_5}| = |S_{absorb,f_3,f_5}|$. 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 $|T_3| = |T_2| - r^2 s$. From this number of packets, the number of packets that belong to $S_{3,rem}$ is $|S_{3,rem,f_3}| = |T_3| \frac{r^2+r}{r^2+r+2}$.

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

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

For instability, we must have $s^{'} > s$. Therefore, $r \geq 0.749$, by MATLAB.

In order to conclude the proof we should also show that $Q(f_3) + Q(f_5) \leq Q(f_4) + Q(f_6)$. 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

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

Using MATLAB, $Q(f_3) + Q(f_5) \leq Q(f_4) + Q(f_6) \Longrightarrow r \geq 0.743$.

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.

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.

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

$$|T_4| = |Z_{rem,f_6}|+|Y_{rem,f_6}|+(|Z_{pass,f_6}|+|Y_{pass,f_6}|+|X_{pass,f_3}|+|X_{pass,f_3,f_5}|-T_{round-3}$$

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. □

5 An Upper Bound for the Stability in FIFO Networks

Let us consider the network N in Figure 1. We prove here, that given any initial packet configuration for N , and for any adversary A with injection rate $r \leq r_0$, for a given r_0 to be computed, the number of packets in the system remains bounded. So we can conclude that FIFO is stable for N, under any adversary with rate $r \leq r_0$.

Theorem 8. *For the network N, given any initial configuration, FIFO is stable against any adversary with injection rate $0 < r \leq 0.1428$.*

Proof. See full paper ([11]). □

The previous argument, can be extended to work in full generality, for any network N with k edges, maximum in-degree α, and maximum directed path length β.

Theorem 9. *For any network N, given any initial configuration, there exist an $0 < r_N < 1$, such that FIFO is stable with respect any adversary with injection rate $r \leq r_N$, where r_N is independent of the initial configuration and it is a function of the maximum in-degree, the maximum directed path length and the number of edges.*

Proof. See full paper ([11]). □

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

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

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

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

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

[9] 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.

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

[11] D. Koukopoulos, S. Nikoletseas and P. Spirakis. Stability Issues in Heterogeneous and FIFO Networks under the Adversarial Queueing Model (full paper), CTI Technical Report, July 2001, available at www.cti.gr/RD1/nikole/stability.ps