The Necessity of Timekeeping in Adversarial Queueing

Maik Weinard*

Institut für Informatik
Johann Wolfgang Goethe–Universität Frankfurt am Main
Robert-Mayer-Straße 11-15
60054 Frankfurt am Main, Germany

Abstract. We study queueing strategies in the adversarial queueing model. Rather than discussing individual prominent queueing strategies we tackle the issue on a general level and analyze classes of queueing strategies. We introduce the class of queueing strategies that base their preferences on knowledge of the entire graph, the path of the packet and its progress. This restriction only rules out time keeping information like a packet's age or its current waiting time.
We show that all strategies without time stamping have exponential queue sizes, suggesting that time keeping is necessary to obtain subexponential performance bounds. We further introduce a new method to prove stability for strategies without time stamping and show how it can be used to completely characterize a large class of strategies as to their 1-stability and universal stability.

1 Introduction

We study the problem of contention resolution for packet routing in networks. A network is represented as a graph with vertices representing the access points of the network (routers) and edges representing the established connections between routers. Users will insert data – organized in packets of roughly same size – into the access points of the network. Each packet has a destination and routing policies assign a simple path from its source to its destination.

This paper focuses on queueing strategies. Queueing strategies are used to decide which packet may proceed whenever more than one packet intends to traverse an edge. Throughout this paper we will concentrate on *greedy* strategies. These are strategies that allow one packet to cross an edge e whenever there is a packet ready to cross edge e.

We analyze queueing strategies in a distributed online model, i.e., decisions need to be made on the fly, independent of future input, with local information only, as a router is realistically neither aware of packets that will be inserted into the network in the future nor of packets that are currently stored in other nodes of the network.

* email: weinard@thi.informatik.uni-frankfurt.de

We work within the model of adversarial queueing theory [6], a worst-case setup which allows to establish performance guarantees. Packets are inserted by an adversary at arbitrary times and with arbitrary assigned paths. Of course the strength of the adversary needs to be restricted, since no queueing strategy can cope with an adversary, who continuously inserts more packets which seek to cross a specific edge e than edge e can handle.

Definition 1. *An adversary is an (r, b)-adversary, if for every time interval I and every edge e at most $r \cdot |I| + b$ packets with edge e in their path are inserted into the network during I. We call r the rate and b the burstiness.*

Hence for $r \leq 1$ an adversary cannot just simply overload an edge by new insertions. While for some queueing strategies like First-In-First-Out (FIFO) there exist graphs for which an upper bound for the total traffic in a network cannot be guaranteed even for arbitrarily small $r > 0$ [5, 8], others like Nearest-To-Source (NTS) have bounded total traffic in every graph even for $r = 1$ [7].

Definition 2. *1. A queueing strategy is r-stable, if for every graph G and every $b \in \mathbb{N}$ there exists a bound $c_{G,r,b}$ such that for every sequence of insertions by an (r, b)-adversary into G the total number of packets in G never exceeds $c_{G,r,b}$.*
2. A queueing strategy is universally stable, if it is r-stable for every $r < 1$.

Apart from the number of packets in the system, one is also interested in transportation times, i.e., the time between the insertion of a packet into the network and its arrival at the final node of its path. For $r < 1$ there is a straight forward connection between the number of packets in the system and transportation times [2]: if in a given network and against a given adversary the maximal size of a queue is bounded, so are the number of packets in the entire network and the transportation times. Hence for $r < 1$ we concentrate on analyzing the queue sizes of strategies.

It is crucial to see that universal stability is a necessary but by no means sufficient condition for a strategy in order to be "useful" in the $r < 1$ setup. A transportation time superpolynomial in the number of vertices of the graph is inacceptable if one has networks like the internet in mind and the fact, that this is a constant for a fixed network, offers little comfort.

A randomized protocol with polynomial queue size was introduced in [2]. This protocol however can only be derandomized in a centralized manner: total knowledge of all insertions is necessary. In [3] another randomized strategy, following similar high level ideas, with polynomial delay is introduced and derandomized in a distributed manner: each router can do the necessary computations only with the knowledge of the packets inserted into the network via this router. So also deterministic queueing strategies with polynomial queue size do exist.

The question remains as to whether there exist *simple* queueing strategies that achieve the same goal. Longest-In-System (LIS) is the only *prominent* strategy for which the upper bound of $2^{O(d)}$ could only be matched by a lower bound of $\Omega(d)$, where d is the diameter of the graph. It is known [1], that on directed acyclic graphs the queue size for LIS is indeed $O(d)$, whereas a proof for the general case requires new techniques [4]. Also in [4] it is shown, that the transportation time under LIS can be exponential in the path length of a packet. (These paths however turn out to be short in comparison to the diameter.) This result is then extended to a class of *vulnerable* queueing strategies.

We also seek results about entire classes of queueing strategies. We associate a queueing strategy S with a priority function that maps the *state of a packet* and available knowledge of the network to a priority. Among packets competing for the same edge, S then prefers a packet of highest priority. Classes arise by specifying which parameters a strategy bases its priority on. Our goal is a study of the large class of *strategies without time stamping.*

Definition 3. *We say that a queueing strategy S_f operates* without time stamping *if it assigns priorities $f(G, P, a)$, where G is the graph of the network, P is the path of the packet and a is the number of edges already traversed. We call strategies that operate $\underline{w}$ithout $\underline{t}$ime $\underline{s}$tamping WTS-strategies for short.*

Prominent WTS-strategies include Nearest-To-Source (NTS), Farthest-From-Source (FFS), Nearest-To-Go (NTG), and Farthest-To-Go (FTG) with priority functions $f_{NTS}(G, P, a) = -a$, $f_{FFS}(G, P, a) = a$, $f_{NTG}(G, P, a) = a - |P|$ and $f_{FTG}(G, P, a) = |P| - a$ respectively.

In Section 3 (Theorem 1) we show that each WTS-strategy has queue size $2^{\Omega(\sqrt{n})}$, where n is the number of vertices. Moreover the diameter d turns out to coincide asymptotically with $\sqrt{n}$ and hence the queue size is $2^{\Omega(d)}$.

This result suggests that time keeping is crucial to obtain good performance bounds, as the only *reasonable* quantities that WTS-strategies ignore, are times, such as the age of a packet – as used in LIS – or the current waiting time of a packet – as used in FIFO.

In [9] the term *eternal packet* is introduced for a packet that gets stuck in a network indefinitely. For greedy strategies this effect only arises at the critical arrival rate $r = 1$. Observe that there are strategies that avoid eternal packets at $r = 1$ but are instable for every $r < 1$ like FIFO, while others are even 1-stable but fail to avoid eternal packets at $r = 1$. In fact no strategy can avoid eternal packets *and* be 1-stable [9].

All WTS-strategies produce eternal packets at $r = 1$. For burstiness $b \geq 1$ this observation can be easily verified with a one edge network: in step one insert two packets that seek to traverse the edge and in every later step one more packet. As a WTS-strategy is incapable of distinguishing between any of these packets, one can be stuck forever.

In Section 4 we introduce the technique of push-around-cycles that can be used to prove 1-stability for WTS-strategies. Using this technique we provide a complete classification of 1-stable distance-based strategies: these strategies base their decision on the number of edges a packet has already crossed and the length of its path. It turns out that in this class of strategies 1-stability and universal stability coincide. Conclusions and open problems are discussed in Section 5.

2 Notation and Conventions

Throughout this paper we let G denote the graph of the network, $\mathcal{P}_G$ the set of all simple paths in G and n, m, d the number of vertices, number of edges and the diameter of the graph in question. We assume that the network operates in consecutive steps, where each step breaks down into three substeps:

1. Insertion: New packets with assigned paths are inserted by the adversary.
2. Transportation: Packets move along edges.
3. Clean Up: Packets that have crossed the last edge of their path are removed from the system.

Hence a packet that is inserted into the system may proceed in the same step. In each step each edge can be crossed by at most one packet. We use $Q_e(t)$ to denote the set of packets ready to cross edge e in step t. We occasionally use multiple edges; however, if desired, multiple edges can be eliminated by introducing extra nodes. Whenever two packets of same priority reside in the same queue we assume worst case tie resolution.

3 Queue Size of WTS-Strategies

We now see that WTS-strategies cannot avoid exponential queue size and hence exponential transportation time.

Theorem 1. *There is a family G_k of graphs with $n = 2k^2 + 6$ nodes and diameter $d = 4k$ so that every WTS-strategy requires queues of size $2^{\Theta(k)}$ for $r > 0.5$ and $b > \frac{2rk}{2r-1}$.*

Proof. We describe $G_k = (V_k, E_k)$. G_k basically consists of k^2 copies of the gadget G_{ij} (see Figure 1). G_k has $2k^2 + 6$ vertices and diameter $4k$.

Let R_i be the set of paths from X to Z traversing only the gadgets of row i. Let C_j be the set of paths from X' to Z' traversing only the gadgets of column j. We will only work with these row and column paths.

For each of the x-edges x_{ij}^l we determine a *dominant path*: i.e., a path in $R_i \cup C_j$ that uses x_{ij}^l and has a maximum priority in the queue of x_{ij}^l. Hence in the queue of x_{ij}^l a packet on the dominant path has priority

$$\max\left\{\max_{P\in R_i}\{f(G_k, P, 2j) | P \text{ uses } x_{ij}^l\}, \max_{P\in C_j}\{f(G_k, P, 2i) | P \text{ uses } x_{ij}^l\}\right\}.$$

Observe that a packet on a path in R_i $[C_j]$ has traversed $2j$ edges $[2i$ edges$]$ when reaching x_{ij}^l. If a dominant path of x_{ij}^l is in R_i we say that edge x_{ij}^l is *row dominated*, otherwise it is *column dominated*. Furthermore we say that gadget $G_{i,j}$ is row [column] dominated if at least $k + 1$ of its x-edges are row [column] dominated. Hence each gadget is either row or column dominated.

We now focus on a row in which at least half of the gadgets are column dominated or a column in which at least half of the gadgets are row dominated.

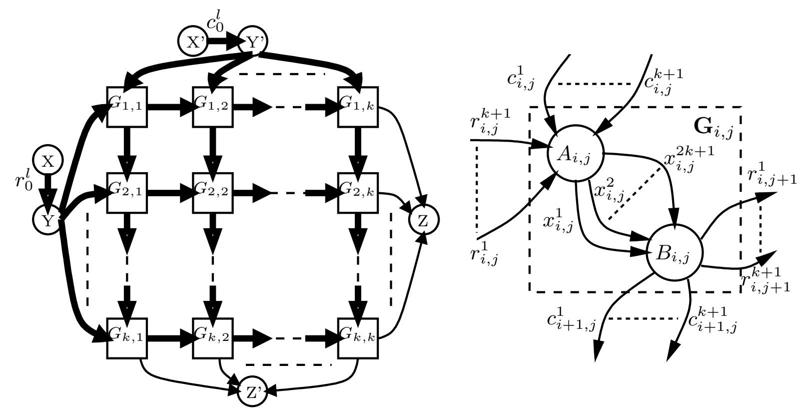

Fig. 1. Graph G_k contains k^2 gadgets $G_{i,j}$ and extra nodes X, Y, Z, X', Y' and Z' arranged and connected as indicated on the left. A thick arrow represents a set of $k+1$ multiple edges. Each gadget consists of two internal nodes $A_{i,j}$, where all incoming edges end, and $B_{i,j}$, where all outgoing edges originate, connected by $2k+1$ directed edges named $x_{i,j}^1, \ldots, x_{i,j}^{2k+1}$. The incoming row edges (coming from $G_{i,j-1}$ resp. Y) are labeled $r_{i,j}^l$ for $1 \leq l \leq k+1$. The incoming column edges (coming from $G_{i-1,j}$ resp. Y') are labeled $c_{i,j}^l$ for $1 \leq l \leq k+1$. The row edges [column edges] leaving $G_{i,k}$ [$G_{k,j}$] and entering Z [Z'] are named $r_{i,k+1}$ [$c_{k+1,j}$]. Finally we call the $k+1$ edges connecting X and Y [X' and Y'] r_0^l [c_0^l] for $1 \leq l \leq k+1$.

Observe that at least one such row or column must exist. W.l.o.g. asssume that there are $q \geq \frac{k}{2}$ column dominated gadgets in row i_0 and that in column dominated gadgets $G_{i_0,j}$ edges $x_{i_0,j}^1, \ldots, x_{i_0,j}^{k+1}$ are column dominated. The following algorithm carefully chooses paths from R_{i_0} that we will use to insert packets.

1. Initially let L consist of the edge $r_{i_0,k+1}$, the edges r_0^l and $r_{i_0,j}^l$ as well as $x_{i_0,j}^l$ for $1 \leq j \leq k$ and $1 \leq l \leq k+1$. We call edges in L *legal* and call a path legal, if it only uses legal edges. Set $z := q$.
2. For j from k to 1 in descending order repeat:
   (a) Choose $e \in \{r_{i_0,j}^l | 1 \leq l \leq k+1\} \cap L$ arbitrarily (a legal entrance to $G_{i_0,j}$).
   (b) IF $G_{i_0,j}$ is column dominated:
      i. Let S_z be a legal path that uses e and assigns a minimum priority (restricted to legal paths using e) in Q_e. (I.e., $f(G_k, S_z, 2j-1)$ is minimal). Remove the edges S_z traverses before e from L.
      ii. Let $e' \in \{x_{i_0,j}^l | 1 \leq l \leq k+1\}$ be a legal edge S_z does not use.
      iii. Let D_z be the dominant path of e'. Set $z := z - 1$.
   (c) ELSE: Choose $e' \in \{x_{i_0,j}^l | 1 \leq l \leq k+1\} \cap L$ arbitrarily.
   (d) Remove all edges $r_{i_0,j}^l \neq e$ and $x_{i_0,j}^l \neq e'$ with $1 \leq l \leq k+1$ from L.
3. Choose a legal path S_0 arbitrarily.

Observe that the choices of e and e' are always well defined, since at start there are for each j at least $k+1$ legal $r^l_{i_0,j}$ and $x^l_{i_0,j}$ edges. At most $k-1$ of them are removed in steps 2(b)i before e and e' are picked.

We are now ready to start our insertion of packets. The insertion scheme proceeds in q phases that correspond to the column dominated gadgets. Let these be $G_{i_0,j_1}, G_{i_0,j_2}, \ldots G_{i_0,j_q}$. To start off the process we exploit burstiness and launch b packets along path S_0 in one step. We call this set of packets X_0.

Phase t: Phase t starts, when the first packet of X_{t-1} is ready to cross $r^l_{i_0,j_t}$ and lasts for $|X_{t-1}|$ steps. During phase t we insert packets along S_t and D_t in parallel at rate r. Note that S_t and D_t are edge disjoint.

In the queue of $r^l_{i_0,j_t}$ the packets from S_t collide with the packets of X_{t-1} for the first time. They have a priority not greater than any packet in X_{t-1} as the paths, the packets of X_{t-1} are travelling on, were all legal when S_t was picked minimally. Hence none of the $r|X_{t-1}|$ packets from S_t is able to traverse G_{i_0,j_t} in phase t.

After at most $2k$ steps of phase t the first packet on D_t arrives in G_{i_0,j_t} and from then on the $x^l_{i_0,j_t}$ edge, the packets from X_{t-1} intend to use, is occupied $r\cdot(|X_{t-1}|-2k)$ steps by blocking packets. As D_t is a dominant path of the edge at least $r\cdot(|X_{t-1}|-2k)$ packets from X_{t-1} do not traverse G_{i_0,j_t} in phase t.

Let X_t be the union of these remaining X_{t-1} packets and the newly inserted packets from S_t. Then $|X_t| \geq 2r|X_{t-1}| - 2rk$ holds. Observe that for $r > \frac{1}{2}$ and sufficiently large b (i.e., $|X_0| = b > \frac{2rk}{2r-1}$) the size of X_t has increased by a multiplicative factor. As $q = \Theta(k) = \Theta(d)$, the last set X_q has size $2^{\Theta(d)}$. □

4 Stability of WTS-Strategies

In this section we provide a new method for proving 1-stability of WTS strategies, that can be used to unify the proofs for NTS and FTG as well as for proving the 1-stability of entire classes of strategies. Besides providing a sufficient criterion for 1-stability of WTS-strategies we also characterize universally stable distance-based strategies.

Crucial for this approach is the concept of push-around-cycles. A push-around-cycle is intuitively speaking a sequence of paths that intersect in a cyclic manner so that the priorities allow to push packets around this cycle indefinitely. Assume WTS-strategy S_f is used. We show that whenever the number of packets in a network G can be driven beyond any bound (instability) then G contains a push-around-cycle with respect to f. By contraposition we may then conclude 1-stability whenever a queueing strategy prevents push-around-cycles in every single graph. We introduce the concept of a path prefix.

Definition 4. *Let $P = (e_1, e_2, \ldots, e_z)$ be a path. Then the* path prefix $[P, a]$ *for $a \leq z$ consists of the edges $(e_1, e_2, \ldots, e_a)$.*

Let $Q^P_e(t)$ denote the set of packets in $Q_e(t)$ travelling along path P and $Q^P(t)$ be the set of packets travelling along path P at time t. (Hence $Q^P(t) = \bigcup_{i=1}^{z} Q^P_{e_i}(t)$.) Furthermore set $Q^{[P,a]}(t) = \bigcup_{i=1}^{a} Q^P_{e_i}(t)$ as the set of packets in path prefix $[P, a]$.

WTS-strategies define priorities between paths P and P' meeting in a common edge e: a packet reaching edge e on path P has the right of way over a packet reaching e on path P' if $f(G,P,j) > f(G,P',i)$ where e is the j-th (resp. i-th) edge of P (resp. P'). The Lemma 1 shows that for every heavily loaded path prefix $[P,a]$ there is another heavily loaded path prefix $[P',a']$ that has priority over P in an edge e. Later we apply this method repeatedly to find a necessary condition for instability at every $r \leq 1$. Figure 2 illustrates Lemma 1.

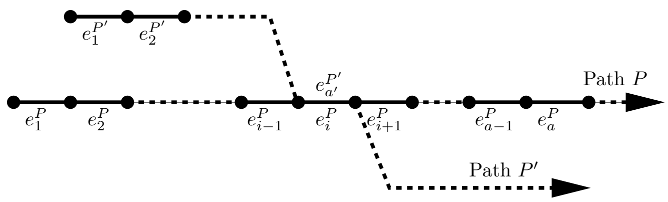

Fig. 2. An illustration for Lemma 1. The common edge $e_{a'}^{P'} = e_i^P$ is distinguished.

Lemma 1. *Assume WTS-strategy S_f (with priority function f) is used on a graph G. Moreover assume that after a sequence of insertions for t steps by an $(1,b)$ adversary there is a path prefix $[P,a]$ such that $|Q^{[P,a]}(t)| \geq c$, where c is a constant larger than b. Then there exists a path prefix $[P',a']$ such that*

- *P' has the right of way over P in a common edge $e = e_i^P = e_{a'}^{P'}$ and*
- *the path prefix of P' ending in e has at some moment in time before time t carried at least $\frac{c}{2\cdot 3^d \cdot |\mathcal{P}_\mathcal{G}|} - \frac{b}{|\mathcal{P}_\mathcal{G}|}$ packets.*

Proof. Assume that $|Q^{[P,a]}(t)| \geq c$ holds. Then choose t_0 minimal such that at time t_0 there exists $i \leq a$ such that

$$|Q_{e_i}^P(t_0)| > \frac{c}{3^{a-i+1}}. \tag{1}$$

Such a $t_0 \leq t$ is well defined, since otherwise we have $|Q_{e_j}^P(t)| \leq \frac{c}{3^{a-j+1}}$ for all $j \leq a$ and consequently

$$\begin{aligned} |Q^{[P,a]}(t)| &= \sum_{j=1}^{a} |Q_{e_j}^P(t)| \leq \sum_{j=1}^{a} \frac{c}{3^{a-j+1}} \\ &= c \cdot 3^{-a} \cdot \sum_{j=1}^{a} 3^{j-1} = c \cdot 3^{-a} \frac{3^a - 1}{2} < \frac{c}{2}, \text{ a contradiction.} \end{aligned}$$

Choose i to be a minimal index satisfying inequality (1). Furthermore pick t_1 maximally with $t_1 < t_0$ such that $Q_{e_i}^P(t_1) = \emptyset$. t_1 exists, since all queues are assumed to be empty in the beginning. Exploiting $t_1 < t_0$ and the minimality of t_0 we get $|Q^{[P,i-1]}(t_1)| = \sum_{j=1}^{i-1} |Q^P_{e_j}(t_1)| \le \sum_{j=1}^{i-1} \frac{c}{3^{a-j+1}} \le \frac{c}{2\cdot 3^{a-i+1}}$.

Since $|Q^P_{e_i}(t_0)| > \frac{c}{3^{a-i+1}}$, at most half of the packets in $Q^P_{e_i}(t_0)$ were already in the system at time t_1. We concentrate on the time interval $J = (t_1, t_0]$ and assume that y packets are inserted into path P during J. Let x denote the number of packets on path P that traverse e_i during J. We then conclude

$$y - x \ge \frac{c}{2\cdot 3^{a-i+1}}. \tag{2}$$

Since $Q^P_{e_i}(t)$ is nonempty during J, one packet traverses e_i in every step of J, and thus $t_0 - t_1 - 1 = |J|$ packets traverse e_i. Hence $(t_0 - t_1 - 1 - x)$ packets from paths that have priority over P in e_i traverse edge e_i during J. Due to the restriction on the adversary at most $(t_0 - t_1 - 1 + b)$ packets with e_i are inserted during J and at most $(t_0 - t_1 - 1 + b - y)$ of them travel on paths other than P. We may hence conclude – using (2) –, that at least

$$(t_0 - t_1 - 1 - x) - (t_0 - t_1 - 1 + b - y) = y - x - b \ge \frac{c}{2\cdot 3^{a-i+1}} - b \ge \frac{c}{2\cdot 3^{d}} - b$$

packets on paths with priority over P in e_i are in the system at time t_1 and that they are somewhere before or on edge e_i on their respective paths.

Finally we observe that at least one path with priority over P in e_i must carry at least $\frac{c}{2\cdot 3^d \cdot |\mathcal{P}_G|} - \frac{b}{|\mathcal{P}_G|}$ of these packets before or on e_i and hence if P' is this path and a' is picked so that $e^{P'}_{a'} = e_i$ we have verified the claim. □

A repeated application of Lemma 1 leads to the concept of push-around-cycles.

Definition 5. *A push-around-cycle with respect to a WTS-strategy with priority function f and a network G is a sequence of paths $P_1, P_2, \dots, P_r \in \mathcal{P}_G$ in G with two distinguished edges $e^{P_i}_{x_i}$ and $e^{P_i}_{y_i}$ for every path with the following properties:*

- $\forall_{1\le i\le r}\ x_i \le y_i$: *edge $e^{P_i}_{x_i}$ precedes edge $e^{P_i}_{y_i}$ on P_i,*
- $\forall_{1\le i\le r-1}\ e^{P_i}_{y_i} = e^{P_{i+1}}_{x_{i+1}}$ *and* $e^{P_r}_{y_r} = e^{P_1}_{x_1}$: *The second distinguished edge of P_i is the first distinguished edge of P_{i+1} and the second distinguished edge of the last path is the first distinguished edge of the first path.*
- $\forall_{1\le i\le r-1}\ f(G, P_i, y_i) \ge f(G, P_{i+1}, x_{i+1})$ *and* $f(G, P_r, y_r) \ge f(G, P_1, x_1)$: *path P_i has the right of way over P_{i+1} with respect to their common distinguished edge. Moreover P_r has the right of way over P_1 with respect to their common edge.*

Figure 3 illustrates the concept of a push-around-cycle. We are now ready to state the main result of this section.

Theorem 2. *If a WTS-strategy S_f with priority function f is instable at $r = 1$ on a graph G, then there exists a push-around-cycle among the paths of G with respect to S_f.*

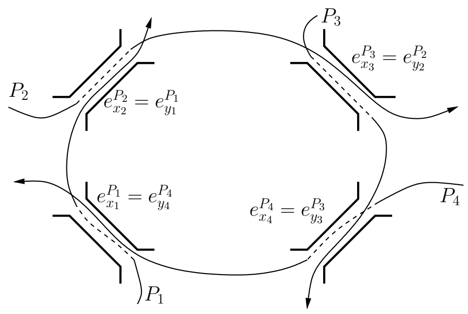

Fig. 3. An illustration for a push-around-cycle consisting of four paths and their four distinguished edges. A dashed line indicates that the respective path must yield priority.

Proof. We define the following recurence in order to use Lemma 1: $A(0) = b$ and $A(i) = \frac{A(i+1)}{2 \cdot 3^d \cdot |\mathcal{P}_G|} - \frac{b}{|\mathcal{P}_G|}$. As we assume instability, there is a sequence of insertions that causes G to accomodate $|\mathcal{P}_G| \cdot A(d \cdot |\mathcal{P}_G|)$ packets. Hence one path accomodates at least $A(d \cdot |\mathcal{P}_G|)$ packets. Using this entire path P_0 and its entire length $a_0 := |P_0|$ as a first path prefix $[P_0, a_0]$ we apply Lemma 1 iteratively: if we have a path prefix $[P_i, a_i]$ that can be forced to accomodate $A(d \cdot |\mathcal{P}_G| - i)$ packets, Lemma 1 provides a path P_{i+1} that intersects $[P_i, a_i]$ and has priority at that intersection. If we pick a_{i+1} according to Lemma 1, we know that the path prefix $[P_{i+1}, a_{i+1}]$ can be forced to contain $A(d \cdot |\mathcal{P}_G| - (i+1))$ packets.

As we have picked $A(d \cdot |\mathcal{P}_G|)$ large enough, we can iterate the application of Lemma 1 $d \cdot |\mathcal{P}_G|$ times, which is an upper bound on the number of path prefixes in G. Hence a cycle must be closed in the process and we have our theorem. □

By contraposition we get immediately that every WTS-strategy, that does not allow push-around-cycles in a network G, is 1-stable in G. The following corollary contains Nearest-To-Source and Farthest-To-Go as special cases.

Corollary 1. *Assume a WTS-strategy S_f with priority function f is given. If f is strictly decreasing along all paths P in all graphs G, i.e., $f(G, P, i) > f(G, P, i+1)$, then S_f is 1-stable.*

Proof. Assume G contains a push-around-cycle with respect to S_f, let $P_1, \ldots, P_r$, $x_1, \ldots, x_r$ and $y_1, \ldots, y_r$ be defined in accordance to Definition 5. We then have $f(G, P_1, x_1) > f(G, P_1, y_1) \geq f(G, P_2, x_2) > \ldots > f(G, P_r, y_r) \geq f(G, P_1, x_1)$ as a contradiction. Hence by Theorem 2 the strategy is stable at $r = 1$. □

Remark 1. Assume that we have a WTS-strategy that does not depend on the number of edges a packet has traversed so far. If the strategy assigns different priorities to different paths in every G, then this strategy is 1-stable.

We say that a queueing strategy is distance-based, if its priority function only depends on the the number x of traversed edges and the length y of the packet's path. We provide a complete classification of 1-stable distance-based strategies.

Theorem 3. *Let f be the priority function of a distance-based queueing strategy S_f. Then S_f is 1-stable if and only if*

$$\forall(x, y)\ 1 \leq x < y\ :\ f(x, y) < f(x-1, y). \tag{3}$$

It is not even universally stable otherwise.

Proof. The 1-stability given (3) follows immediatly from Corollary 1, since f is strictly decreasing along all paths. To complete the proof we need to carefully embed and adapt the *baseball graph* from [2] and come up with an elaborate insertion scheme.

Our proof exploits that S_f is only stable if it guarantees stability for every possible choice among packets of same priority. Hence if two packets with identical x and y values collide, we may pick the one to be prefered in a worst case manner. We consider the network of Figure 4.

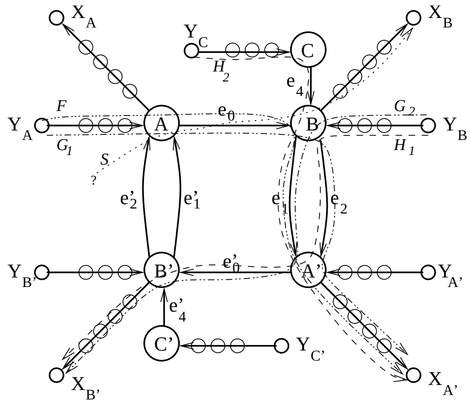

Fig. 4. The graph used to prove instability of $\mathbf{A_f}$. We have *central* nodes A, B, C and A', B', C'. Each central node has an *entrance path* starting at a node named $Y_A, \ldots, Y_{C'}$. Each of these entrance paths consists of $x-1$ nodes and $x-1$ edges. Hence for $x = 1$ these entrance paths disappear and the Y-node is identical with the corresponding central node. Also A, B, A' and B' have an *exit path* ending in a node named $X_A, \ldots, X_{B'}$. Each exit path has length $y - x - 1$. So for $y = x + 1$ these paths vanish and the X-nodes are identical with their respective central node.

We assume that there exists a time t such that if injection of new packets is completely stopped after step t , there is still – for each of the next s steps – a packet crossing e_0. These packets (we call them set S) will have crossed precisely x edges before e_0, have destination X_B and a total path of length y.

We intend to construct an injection process (Phases 1,2 and 3), such that there is a set of more than s packets waiting to cross edge e'_0, if injection is stopped after step $t' > t$. These packets will have crossed precisely x edges before

e'_0, have destination $X_{B'}$ and a total path of length y. This implies instability, since the process can then be repeated arbitrarily.

In the caption of Figure 4 we have introduced the notion of central node, entrance and exit path. All the packets that we are inserting in phases 1,2 and 3 have a path of length y. In the first central node of their paths they have crossed $x-1$ edges, in the second central node they have crossed x edges. The necessary insertions are listed in the following table along with references to the relevant observations.

Phase	Duration	Inserted Sets & Paths	Size	Observations
1	s	$F:(Y_A \xrightarrow{*} A \xrightarrow{e_0} B \xrightarrow{e_1} A' \xrightarrow{*} X_{A'})$	rs	(1),(2),(3),(4)
2	rs	$G_1:(Y_A \xrightarrow{*} A \xrightarrow{e_0} B \xrightarrow{e_2} A' \xrightarrow{*} X_{A'})$	r^2s	(2),(4),(5)
		$G_2:(Y_B \xrightarrow{*} B \xrightarrow{e_1} A' \xrightarrow{e'_0} B' \xrightarrow{*} X_{B'})$	r^2s	(3),(4),(6)
3	r^2s	$H_1:(Y_B \xrightarrow{*} B \xrightarrow{e_2} A \xrightarrow{e'_0} B' \xrightarrow{*} X_{B'})$	r^3s	(5)
		$H_2:(Y_C \xrightarrow{*} C \xrightarrow{e_4} B \xrightarrow{e_1} A' \xrightarrow{*} X_{A'})$	r^3s	(6)

(1) In phase 1 the set S of s packets we assume to cross e_0 complete their journey. When F reaches A, it will be blocked by S, since $f(x,y) \geq f(x-1,y)$ holds. Observe that $x-1$ steps pass before F's first packet reaches A. So in fact not all of the rs packets of F can get to A before phase 1 ends, i.e., the last one has just been inserted and needs another $x-1$ steps to get to A. For s large enough however this effect causes no problems.

After phase 1 S has vanished. (The last packets of S are actually still on their way from B to X_B, but they do not interfere with anything we are about to do and we will consider them gone. The same argument in the next steps allows us to regard every packet as *out of the way* once it's on its exit path.)

(2) Set G_1 collides with F in node A. Their path length and advance in the path at this point are identical, hence we may choose that F is advanced over e_0.

(3) Set G_2 collides with F in node B. Since F has traversed x edges and G_2 has traversed $x-1$ edges, F will be prefered.

(4) At the end of phase 2 F has vanished and both G_1 and G_2 are still in the first central node of their paths.

(5) G_1 completes its journey in phase 3. In node B the set G_1 will block H_1.

(6) In the first x steps of Phase 3 x packets of G_2 cross e_1 and are lost for our purpose. After these x steps the stream of H_2 packets has reached B. In B packets from H_2 will be prefered over G_2-packets. Of course, as $r < 1$ holds, the stream of H_2 packets occupies e_1 for $r \cdot (r^2s - x)$ of the remaining $r^2s - x$ steps of Phase 3. Hence $(1-r) \cdot (r^2s - x)$ more G_2 packets slip through and are lost.

We define the end of phase 3 as the time t'. Observe that $r(r^2s - x)$ packets from G_2 and the entire H_1 still need to cross e'_0. Their combined size is $|S'| = 2r^3s - rx$. So by having S and r sufficiently large we can guarantee, that $|S'| > |S|$ holds and we have successfully increased the number of packets in the system. To start out the process we have to use a sufficiently large burstiness. $\square$

5 Conclusion and Open Problems

We have introduced the class of WTS-strategies and obtained general results about this class itself and its subclass of distance-based strategies. Most importantly we have ruled out the existence of WTS-strategies with subexponential queue size indicating, that some form of timekeeping is necessary to achieve polynomial queue size. Furthermore we have introduced the concept of push-around-cycles and used it to classify the 1-stable and universally stable distance-based strategies.

In order to be of practical use, queueing strategies need to be as simple as possible. So the question remains whether queueing strategies simpler than the one introduced in [3] and with polynomial queue size exist. Such a candidate is Longest-In-System, as LIS is obviously one of the simplest queueing strategies that does use timekeeping.

6 Acknowledgements

I would like to thank Gregor Gramlich, Matthias Poloczek and Georg Schnitger, as well as an unknown referee for many helpfull comments.

References

1. Adler, Micah and Rosen, Adi, Tight Bounds for the Performance of Longest in System on DAGs, *Proc. of the 19th Symposium on Theoretical Aspects of Computer Science, 2002*, pp. 88-99
2. Andrews, M., Awerbuch, B., Fernández, A., Leighton, T., and Liu, Z., Universal-Stability Results and Performance Bounds for Greedy Contention-Resolution Protocols, *Journal of the ACM, Vol. 48, No 1, January 2001*, pp. 39-69
3. Andrews, M., Fernández, A., Goel, A., Zhang, L., Source Routing and Scheduling in Packet Networks, *Proc. of the 42nd Symposium on Foundations of Computer Science, 2001*, pp. 168-177
4. Andrews, M., Zhang, L., The Effects of Temporary Sessions on Network Performance, *SIAM Journal of Computation, Vol. 33, No 3*, pp. 659-673
5. Bhattacharjee, R. and Goel, A., Instability of FIFO at arbitrarily low rates in the adversarial queueing model, *Proc. of the 44th Symposium on Foundations of Computer Science, 2003*, pp. 160-167
6. Borodin, A., Kleinberg, J., Raghavan, P., Sudan, M., and Williamson, D. P. Adversarial queueing theory, *Journal of the ACM, Vol. 48, No 1, January 2001*, pp. 13-38
7. Gamarnik, David, Stability od Adaptive and Non-Adaptive Packet Routing Policies in Adversarial Queueing Networks, *SIAM Journal on Computing, Vol. 32, No 2, 2003*, pp. 371-385
8. Koukopoulos, D., Mavronicolas, M., Spirakis, P., FIFO is Unstable at Arbitrarily Low Rates (Even in Planar Networks), *Electronic Colloq. on Computational Complexity, 2003*
9. Rosén, Adi and Tsirkin, Michael S., On Delivery Times in Packet Networks under Adversarial Traffic, *Proceedings of the 16th ACM Symposium on Parallelism in Algorithms and Architectures, 2004*, pp. 1-10