Tight Bounds for the Performance of Longest in System on DAGs *

Micah Adler† Adi Rosén‡

Abstract

A growing amount of work has been invested in recent years in analyzing packet-switching networks under worst-case scenarios rather than under probabilistic assumption. Most of this work makes use of the model of "adversarial queuing theory" proposed by Borodin et al. [9], under which an adversary is allowed to inject into the network any sequence of packets as long as – roughly speaking – it does not overload the network.

We show that the protocol Longest In System, when applied to directed acyclic graphs, uses buffers of only linear size (in the length of the longest path in the network). Furthermore, we show that any packet incurs only linear delay as well. These are, to the best of our knowledge, the first deterministic polynomial bounds on queue sizes and packet delays in the framework of adversarial queuing theory (other than on trees and the cycle). Furthermore these results separate Longest In System from other common universally stable protocols for which there exist exponential lower bounds that are obtained on DAGs. Our upper bounds are complemented by matching linear lower bounds on buffer sizes and packet delays.

*A preliminary version of this paper appeared in the proceedings of STACS 2002.

†Dept. of Computer Science, University of Massachusetts, Amherst, MA 01003. Email: micah@cs.umass.edu. Fax: 1-413-545-1249.

‡Dept. of Computer Science, Technion, Haifa 32000, Israel. Email: adiro@cs.technion.ac.il. Research supported in part by a grant from the Fund for the Promotion of Research at the Technion.

1 Introduction

The behavior of packet-switching networks, in which packets are injected into the network in a continuous manner, has been the subject of considerable amount of work in recent years. See e.g., [10, 8, 16, 17, 19, 18, 11, 9, 3, 7, 13, 15, 2, 5]. In such networks packets are transmitted between adjacent switches over links, in discrete time steps, a prescribed number of packets over each link in any time step. New packets are injected into the network dynamically at any time. Each packet is injected into a source node and has to reach a destination node over a prescribed path. The packets travel to their destinations in a "store-and-forward" manner, being stored in buffers at intermediate switches.

As the capacity of the links in the network is limited, two important questions arise in this setting: what are the delays incurred by the packets, and what are the sizes of the buffers used. A crucial question is the question of *stability*, i.e., whether the maximum buffer size grows with time. (In other words, is there a finite upper bound, which is independent of time, that bounds the size of the buffers in the network). The answers to these questions depend on the topology of the network, the injection pattern of the packets, and the contention-resolution protocol (used when more packets than the capacity of an edge attempt to cross this edge at a given time).

Considerable amount of research has been published on these questions under certain probabilistic assumptions on the injection of packets [10, 8, 16, 17, 20, 19, 18]. In particular the assumptions are usually that the packets are generated at the different nodes by independent Poisson or Bernoulli processes, and are destined to uniformly distributed destinations. More recently, a growing amount of research has been invested in an attempt to answer the same questions without resorting to probabilistic assumptions. Rather, the questions of stability, queue size, and packets delay are studied under worst-case scenarios [9, 3, 7, 15, 13, 14, 2, 5], in an effort to prove stability and small queue size even when the packets are injected by an adversary and not by an oblivious random process. To formulate this adversary we use the model of "adversarial queuing theory" introduced by Borodin et al. [9]. Informally, in this model an adversary can inject any sequence of packets, under the condition that the paths that the packets have to follow do not accumulate on any edge at a rate higher than the capacity of this edge.[1]

The question of stability under this model received rather detailed answers. A number of natural protocols are known to be universally stable (i.e., they are stable on any network topology), while others are known not to be always stable [9, 3]. Methods for determining if a network is universally stable (i.e., it is stable with any greedy protocol) have been developed [13], and the set of universally stable networks is well characterized [15, 14, 1]. While stability ensures that there are some upper bounds, independent of time, on the sizes

[1] Note that it is necessary to limit the power of the adversary. Otherwise an adversary can inject into the network a sequence of packet which is beyond the capacity of the network. The formulation above captures the condition that the adversary can injects packets up to the capacity of the network, but not beyond it, disallowing sequences for which stability is impossible. For a formal definition of the adversary see Section 1.2.

of the buffers, it does not guarantee small bounds on them. In contrast to the question on stability, the question of good upper bounds on queue sizes (and delays of packets) for specific and universally stable protocols remains rather open. The only known upper bounds on queue sizes on general networks are exponential in the length of the longest path in the network [9, 3]. The only known sub-exponential bounds on queue sizes hold only for trees and the cycle [9, 3]. Moreover, several natural and universally stable protocols have exponential lower bounds [3]. Thus, determining whether there exists a protocol with polynomial queue sizes (and packet delays) is an important question in this area [3]. Andrews et al. [4] gave recently a protocol with polynomial queue sizes for the case where upper bounds on the parameters of the adversary are known to the protocol. The protocol Longest In System (henceforth LIS), in which a packet has the highest priority to cross an edge if its time of injection is the earliest, has been suggested as a good candidate: no exponential lower bound is known for it, and there is a centralized polynomial algorithm that works in the spirit of LIS [3]. Determining the specific behavior of LIS was thus also raised as an important open question [9, 3].

In this paper we show that the protocol LIS, when applied to Directed Acyclic Graphs (henceforth DAGs) is polynomial. Previously, for DAGs, it was only known that any greedy protocol is stable, but only with exponential upper bounds. We show that on DAGs, LIS has linear (in the longest path in the network) buffer sizes and linear packet delays. This is the first sub-exponential deterministic upper bound on buffer sizes obtained in the framework of adversarial queuing theory, other than on trees and the cycle. Moreover, this result separates LIS from other natural stable protocols (such as Shortest In System), as these have exponential lower bounds that are obtained on DAGs [3].

In addition to our upper bounds, we show that these bounds are best (or almost best) possible, up to constant factors. We give a matching linear lower bound on the maximum delay incurred by the packets. We also give an almost matching linear lower bound on the maximum buffer size used by LIS on DAGs. [2] Our results thus establish the behavior of LIS on DAGs as being linear.

1.1 Related Work

Two papers within the literature on adversarial analysis of packet-switching networks are particularly close to the present work.

Andrews and Zhang [5] give a lower bound for LIS in the framework of adversarial queuing theory. They give a lower bound on the maximum delay of packets which is exponential in d, the length of the longest path followed by any packet. This lower bound is obtained on a tree of depth exponential in d. Thus, our upper bounds also show that there can be an exponential gap between a function giving the best possible upper bound on the delay of a packet in terms of the longest directed path in the network, and a function giving the same in terms of the longest path followed by any packet.

[2] This lower bound matches the upper bounds in terms of two of the three parameters involved, and has a small gap in terms of the third parameter. See Section 1.3 for the exact bounds.

Scheideler and Vöcking [21] consider layered graphs and a model related to the present model. In their model each node has buffers on *incoming* edges, and in addition a separate buffer for packets injected into the specific node. They give a protocol for layered graphs, which is in the spirit of LIS, but uses also additional control packets. This protocol achieves for layered graphs linear delay for the packets. Their protocol uses buffers of constant size (on any of the incoming edges), while they do not give upper bounds on the size of the buffers used for injected packets (although such bounds can be inferred from the upper bounds on the delay of the packets).

1.2 The Model

We model a communication network as a graph $G = (V, E)$, $|V| = n$, $|E| = m$. Each node $v \in V$ represents a communication switch, and each edge $e \in E$ represents a link between two switches. The switches store and forward packets; the packets are stored in the switches in *buffers*. Every switch has a buffer for each outgoing link, and stores in this buffer the packets to be sent on the corresponding link. We model here the network as a *directed graph*. Thus each edge $e \in E$ is a directed edge, capable of delivering packets in the direction in which it is oriented. The network is *synchronous*, and there is a global clock known to all switches. We number the time steps by $t \in \mathcal{N} = \{1, 2, 3, \ldots\}$. Each edge can transmit a single packet in each time step (in the direction it is oriented).[3]

New packets are injected into the network at any time step. Each packet p is injected into an arbitrary source node s_p, and has to reach an arbitrary destination node d_p. To reach d_p from s_p, the packet has to follow a prescribed path $\pi_p = (s_p = v_0, v_1, \ldots, v_k = d_p)$.

Each time step, in each node, is divided into two sub-steps: in the first sub-step packets are sent from the buffers in which they are stored (on the edge that corresponds to each buffer). At most one packet is sent per edge. In the second sub-step each nodes receives the packets sent to it over its incoming edges, and the packets injected into it from the outside. Those packets that reach their destination are absorbed. The other packets are placed in the buffers that correspond to the next edge on their path.

The injection of packets into the network is controlled by an *adversary*. We use the model of Adversarial Queuing Theory introduced by Borodin et al. [9].

Definition 1: We say that the adversary injecting packets is an $A(w, \varepsilon)$ adversary, for some $\varepsilon \geq 0$ and some integer $w > 1$, if the following holds: for any time $t \in \mathcal{N}$, let $\mathcal{I}^t$ be the set of packets injected during the w time steps from t to $t + w - 1$, inclusive. Let Π^t be the set of paths that the packets in $\mathcal{I}^t$ have to follow: $\Pi^t = \{\pi_p : p \in \mathcal{I}^t\}$. Then, every edge $e \in E$ is used by the paths of Π^t at most $\lfloor (1 - \varepsilon)w \rfloor$ times.

[3] In the literature, networks are many times modeled as *undirected* graphs, with edges capable of transmitting packets in both directions. However, the results in this paper hold in the case when the paths followed by the packets induce a DAG on the graph. We therefore simplify the presentation by defining in advance the graph as directed graph.

A *protocol* is a set of n algorithms, one algorithm corresponding to each switch, that control the sending of the packets from the buffers on the corresponding outgoing links. Each algorithm corresponding to a switch is run locally in it. In particular each algorithm uses only information available in its specific switch and decides if a packet is to be sent over an outgoing link and which packet it will be. A *greedy protocol* is a protocol that always sends a packet over an outgoing edge e from node v, if there is at least one packet in v that has e on its path. We consider here only greedy protocols.

1.3 Our Results

Our results concentrate on the performance of the Longest In System (LIS) protocol, run on networks which are DAGs. We show tight (or almost tight), up to constant factors, linear bounds on both the delay incurred by the packets, and the size of the buffers used by the protocol.

In particular, we show that the size of the buffers used by LIS on DAGs, when the packets are injected by an $A(w, \varepsilon)$ adversary, is bounded from above by $O(\ell w(1-\varepsilon))$, where ℓ is the length of the longest path in the network. More precisely, let the *level* of a node in a DAG be the length of the longest path that leads to it. We show that the buffers in a node v of level i never exceed $O(iw(1-\varepsilon))$. As mentioned above this to the best of our knowledge the first polynomial upper bound on buffer sizes in the adversarial queuing theory model, other than on trees and the cycle. Moreover, this upper bound separates LIS from other universally stable protocols such as Shortest In System, for which there are known exponential lower bounds on DAGs [3]. Our upper bound is complemented by a lower bound of $\Omega(iw(1-\varepsilon)^2)$ on the size of the buffers of LIS in a level-i node of a DAG. We observe that for a fixed adversary this lower bound matches our upper bound (up to constant factors), and is close to our upper bound in general. We also provide a linear upper bound of $O(iw(1-\varepsilon))$ on the delay incurred by any packet destined to a node of level i, and give matching lower bounds. We thus show that the maximum delay incurred by a packet destined to a node v of level i is $\Theta(iw(1-\varepsilon))$.

2 Upper Bounds for LIS on DAGs

We first give linear upper bounds on the maximum delay incurred by the packets, and the maximum size of the buffers in the nodes.

For the purpose of the proofs we order the nodes in the graph by their topological order as induced by the DAG, assigning each node a *level*, starting with level 0, and ending at level ℓ. A node of level i is a node such that the longest path leading to it is of length i. The property that will be useful for the proofs below is that for any edge leading from node u to node v, it holds that the level of u is smaller than the level of v.

The main theorem of this section is the following.

Theorem 1: When LIS is used on a DAG, and packets are injected by a $A(w, \varepsilon)$ adversary,

every packet is delivered to its destination within $\ell w(1-\varepsilon)$ time steps, where ℓ is the length of the longest path in the network.

This theorem immediately follows from the slightly stronger lemma below.

Lemma 2: Let p be a packet injected at time t, and let v be a node on its path, which is not the destination node. If node v is of level i, then p clears node v by time step $t + \lfloor (i+1)w(1-\varepsilon) \rfloor$.

Proof: We prove the lemma by double induction, on i (the level of the node), and on t (the time of the injection of the packet). Let $e = (v, u)$ be the edge on which p has to leave v.

For $i = 0$, any packet that has a node v of level 0 on its path, must be injected into node v, since no edge leads into this node. This includes the packet p under consideration, and any other packet. Thus, if p is injected into v at time t, it can be delayed in v only by other packets injected into v at times $t' \leq t$.

We prove the lemma for a node v of level 0 by induction on t. For $t = 1$ packet p can be delayed only by packets injected at time 1 and use e on their path. By the definition of the adversary there are at most $\lfloor w(1-\varepsilon) \rfloor$ such packets (including p itself). It follows that p will leave v by time step $1 + \lfloor w(1-\varepsilon) \rfloor$. For $t > 1$, consider any packet injected into v at time $t' \leq t - w$. By the induction hypothesis any such packet clears node v by time step $t' + \lfloor w(1-\varepsilon) \rfloor \leq t - w + \lfloor w(1-\varepsilon) \rfloor \leq t$. Thus, packet p can be delayed in v only by packets injected in the time interval $[t-w+1, t]$. There are at most $\lfloor w(1-\varepsilon) \rfloor$ such packets (including p itself). It follows that p will leave v by time step $t + \lfloor w(1-\varepsilon) \rfloor$.

For $i > 0$ we will use the induction hypothesis that for $k < i$, every packet injected at time $t' \geq 1$, and has a node of level k on its path, clears this node by time step $t' + \lfloor (k+1)w(1-\varepsilon) \rfloor$.

We now prove the claim for $i > 0$, and any $t \geq 1$, by induction on t. If a packet p is injected at time $t = 1$, then it can be delayed at v only by packets injected at time 1 (and that have e on their path). We distinguish between two cases. The first one is when p is injected into v and the other one is when p arrives to v over an edge. In the first case, p is clearly at v at the end of time step 1. Since there are at most $\lfloor w(1-\varepsilon) \rfloor$ packets that use e and are injected at time 1, packet p will clear v by time step $1 + \lfloor w(1-\varepsilon) \rfloor$. If p has to arrive to v over an edge, then it has to arrive over some edge $e' = (x, v)$, for a node x of level $k < i$. By the induction hypothesis (on i) packet p clears node x by time $1 + \lfloor (k+1)w(1-\varepsilon) \rfloor$. This means that it is in node v by the end of this time step. Since it can be delayed in v only by packets injected at time 1, it follows that packet p will clear v by time step $1 + \lfloor (k+1)w(1-\varepsilon) \rfloor + \lfloor w(1-\varepsilon) \rfloor \leq 1 + \lfloor (k+2)w(1-\varepsilon) \rfloor \leq 1 + \lfloor (i+1)w(1-\varepsilon) \rfloor$.

For packets injected at time $t > 1$, we again distinguish between the case where the packet is injected into node v, and the case where it has to arrive into v over an edge. If the packet is injected into node v, then clearly it is in v by the end of time step t. Otherwise it has to arrive into v over an edge $e' = (x, v)$, for a node x of level $k < i$. By the induction hypothesis (on i) packet p clears node x by time step $t + \lfloor (k+1)w(1-\varepsilon) \rfloor$, i.e., it arrives

into v by the end of this time step. In any case we know that packet p arrives in v by the end of time step $T = t + \lfloor iw(1-\varepsilon) \rfloor$.

Now consider any packet that uses edge e and is injected at time step $t' \leq t - w$. By the induction hypothesis on t, we have that any such packet clears node v by time step $t' + \lfloor (i+1)w(1-\varepsilon) \rfloor \leq t - w + \lfloor (i+1)w(1-\varepsilon) \rfloor \leq t + \lfloor iw(1-\varepsilon) \rfloor = T$. Therefore, if packet p does not clear node v by time T, it must be at v at this time, and can further be delayed in v only by packets injected in time interval $[t-w+1, t]$ (and use e). By the definition of the adversary there are at most $w(1-\varepsilon)$ such packets (including p). It follows that p clears v by time $T + \lfloor w(1-\varepsilon) \rfloor = t + \lfloor iw(1-\varepsilon) \rfloor + \lfloor w(1-\varepsilon) \rfloor \leq t + \lfloor (i+1)w(1-\varepsilon) \rfloor$. □

Proof of Theorem 1. Any packet injected at t with destination v of level i, has to arrive to v over an edge $e' = (x, v)$ for a node x of level $k < i$. Therefore, by Lemma 2 such packet clears node x (and arrives at v) by time step $t + \lfloor (k+1)w(1-\varepsilon) \rfloor \leq t + \lfloor iw(1-\varepsilon) \rfloor \leq t + \lfloor \ell w(1-\varepsilon) \rfloor$. □

Lemma 2 also provides upper bounds on the sizes of the buffers used by LIS on DAGs. We have the following corollary.

Corollary 3: Consider LIS when run on a DAG, where the packets are injected by a $A(w, \varepsilon)$ adversary. Then the buffer at the tail of any edge $e = (v, u)$, for node v of level i, stores at most $(i+1)w(1-\varepsilon)$ packets at any time.

Proof: Assume by way of contradiction that there is a time when the buffer at the tail of an edge $e = (v, u)$, for a node v of level i stores more than $(i+1)w(1-\varepsilon)$ packets. Then there is at least one packet that will clear node v (of level i) more than $(i+1)w(1-\varepsilon)$ time steps after its injection. A contradiction. □

3 Lower Bounds

We now give lower bounds on the maximum delay incurred by the packets when LIS is run on DAGs, and the maximum buffer size used by the nodes. We begin with a lower bound on the delay of the packets.

Theorem 4: For any i, w, and ε, there is a DAG and an $A(w, \varepsilon)$ adversary, such that some packet, destined to a node v of level i, is delayed $\Omega(iw(1-\varepsilon))$ time steps.

Proof: Consider the following network: start with a complete binary tree of height k, with all edges directed towards the root. We assign a binary string $s(v)$ to every node v of this network, where the m^{th} node in the left to right ordering at distance j from the root receives the j-bit binary representation of the number $m-1$. Thus, every internal node receives the longest common prefix of its children, and the root is assigned the empty string. We also modify the network as follows: for every internal node v of the tree, remove the edge

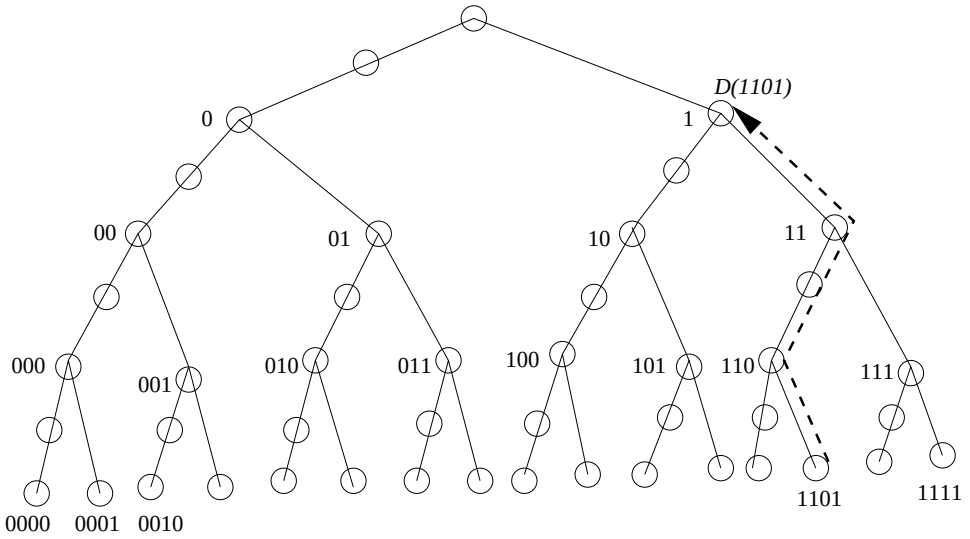

Figure 1: The tree network used for the lower bound.

pointing to v from its left child v_l, (v_l, v), and add a node v_l' and edges (v_l, v_l') and (v_l', v). The resulting network N is a subnetwork of the complete binary tree of height $2k$.

For any node v, let $Z(v)$ be the number of 0's in the string $s(v)$. Note that the distance in N of node v from the root of the tree is $|s(v)| + Z(v)$. Also, for any leaf node v, let $D(v)$ be the node on the path from v to the root such that $s(D(v))$ is obtained by removing least significant bits from $s(v)$ until either the empty string is obtained, or one bit after the first 0 is removed. Note that in the original binary tree, $D(v)$ is the first node for which v is a descendant of, or equal to, the left child of a child of $D(v)$, or the root, if no such node exists. (See Figure 1.)

We can now describe the adversary. The adversary injects $\frac{w(1-\epsilon)}{2}$ packets to each leaf node v at time $k - Z(v)$. For simplicity, we assume here that $\frac{w(1-\epsilon)}{2}$ is an integer, but this assumption is easy to remove.[4] Each such packet has a destination of $D(v)$. We now see that this is in fact a valid adversary. Note that for a leaf v to send a packet that, for nodes u_1 and u_2 of the original tree, traverses the edge (u_1, u_2), or the two edges (u_1, u_1') and (u_1', u_2), the $k - |s(u_1)| - 1$ least significant bits of $s(v)$ must all be 1's, and the $|s(u_1)|$ most significant bits of $s(v)$ must all match the corresponding bits of $s(u_1)$. Thus, at most two leaves send packets that traverse any edge, and therefore any edge is used by at most $w(1-\varepsilon)$ packets.

Claim 1: For any node v from the original tree such that $|s(v)| \leq k-2$, all packets that pass through v on their way to another node arrive between time $(k - |s(v)| - 1)\frac{w(1-\varepsilon)}{2} + 2k - Z(v) - |s(v)| + 1$ and time $(k - |s(v)|)\frac{w(1-\varepsilon)}{2} + 2k - Z(v) - |s(v)|$. During this time interval, exactly one packet arrives at v from each child of v during each time step.

[4] Without the assumption, the adversary injects $\lfloor \frac{w(1-\varepsilon)}{2} \rfloor$ packets to each leaf and the lower bound becomes $\Omega(i \lfloor \frac{w(1-\varepsilon)}{2} \rfloor)$.

Proof: We prove the claim by induction on $k - |s(v)|$, i.e., on the height of the node v. For the base case, let v be any node such that $|s(v)| = k-2$, let v_r be the right child of v, and let v_ℓ be the node other than v that is adjacent to the left child of v (i.e., the left child of v in the original tree). Note that $s(v_\ell) = s(v)0$. Let v''_ℓ be the right child of v_ℓ, and let v'_ℓ be the node other than v_ℓ that is adjacent to the left child of v_ℓ. We see that $\frac{w(1-\varepsilon)}{2}$ packets are injected by the adversary to v'_ℓ at time $k - Z(v'_\ell)$. These packets arrive, one per time step, to v_ℓ starting at time $k - Z(v'_\ell) + 2$. Since $s(v'_\ell) = s(v_\ell)0$, $Z(v'_\ell) = Z(v_\ell) + 1$, and so $k - Z(v'_\ell) + 2 = k - Z(v_\ell) + 1$. The packets that v_ℓ receives from v''_ℓ are injected by the adversary to v''_ℓ at time $k - Z(v''_\ell) = k - Z(v_\ell)$, and thus the packets from v''_ℓ arrive to v_ℓ at exactly the same time steps as the packets from v'_ℓ. However, since the packets from v'_ℓ have been in the system longer, when being sent on to v, they will all have priority over the packets from v''_ℓ. All of the packets from v'_ℓ have v as a final destination. Thus, the first of the packets that v receives from v_ℓ that will be forwarded on arrives at v at time $[k - Z(v_\ell) + 1] + \frac{w(1-\varepsilon)}{2} + 2$, and the remainder arrive one per time step for the next $\frac{w(1-\varepsilon)}{2} - 1$ time steps. Since $Z(v_\ell) = Z(v) + 1$ and $|s(v)| = k - 2$, this means that exactly one of those packets arrives at every time step between $(k - |s(v)| - 1)\frac{w(1-\varepsilon)}{2} + 2k - Z(v) - |s(v)| + 1$, and time $(k - |s(v)|)\frac{w(1-\varepsilon)}{2} + 2k - Z(v) - |s(v)|$. A similar argument shows the analogous fact for the packets that arrive to v from v_r and are forwarded onward by v.

For the inductive step, we assume the claim for all v' such that $k - |s(v')| \leq j$. Choose any v such that $k - |s(v)| = j + 1$, let v_r be the right child of v, and let v_ℓ be the node other than v that is adjacent to the left child of v. By induction, the packets that v_ℓ forwards to v arrive at v_ℓ between time $(j-1)\frac{w(1-\varepsilon)}{2} + 2k - Z(v_\ell) - |s(v_\ell)| + 1$ and time $j\frac{w(1-\varepsilon)}{2} + 2k - Z(v_\ell) - |s(v_\ell)|$. Of these packets, the packets that v_ℓ receives from its left child originated at a node v'_ℓ such that $s(v'_\ell) = s(v_\ell)01^{j-1}$, and the packets that v_ℓ receives from its right child originated at a node v''_ℓ such that $s(v''_\ell) = s(v_\ell)1^j$. Since $Z(v'_\ell) = Z(v''_\ell) + 1$, all the packets that v_ℓ receives from its left child have been in the system longer than the packets that v_ℓ receives from its right child.

Thus, all the packets that v_ℓ receives from its left child and passes on to v (which must also have v as their final destination) have priority over any packet that v_ℓ receives from its right child and passes on to v. Since v_ℓ receives one packet to forward per time step from each of its children, the effect of this is that all of the packets destined for v are forwarded before any packet that has a further destination. There are $\frac{w(1-\varepsilon)}{2}$ packets that v_ℓ forwards to v that have v as a final destination, and thus the packets that v receives from v_ℓ that v will forward arrive between times $j\frac{w(1-\varepsilon)}{2} + 2k - Z(v_\ell) - |s(v_\ell)| + 3$, and time $(j+1)\frac{w(1-\varepsilon)}{2} + 2k - Z(v_\ell) - |s(v_\ell)| + 2$. Since $|s(v_\ell)| = |s(v)| + 1$ and $Z(v_\ell) = Z(v) + 1$, these steps are between $(k - |s(v)| - 1)\frac{w(1-\varepsilon)}{2} + 2k - Z(v) - |s(v)| + 1$ and time $(k - |s(v)|)\frac{w(1-\varepsilon)}{2} + 2k - Z(v) - |s(v)|$. It is easy to see that exactly one of those packets arrives at v at each time step. A similar argument shows that at each time step where v receives a packet from v_ℓ that v will forward onward, v also receives a packet from v_r that it will forward. □

We can now conclude the proof of the theorem. Observe that the longest path that leads to any node v is $i = 2(k - |s(v)|)$. To conclude the proof of the theorem we identify for any node v a packet destined to v that is delayed $\Omega(iw(1-\epsilon))$ time steps. This will be one of the

packets destined to v that arrive to v from its right child v_r. By claim 1, packets destined for v reach v_r no earlier than time $(k-|s(v_r)|-1)\frac{w(1-\varepsilon)}{2}+2k-Z(v_r)-|s(v_r)|+1$. The last of these packets will reach v at time $(k-|s(v_r)|)\frac{w(1-\varepsilon)}{2}+2k-Z(v_r)-|s(v_r)|+2$. These packets are inserted at a leaf v' such that $s(v')=s(v_r)01^{k-|s(v_r)|-1}$, and thus are inserted at time step $k-Z(v')=k-Z(v_r)-1$. Thus, the total time spent in the system by the last of these packets is $(k-|s(v_r)|)\frac{w(1-\varepsilon)}{2}+k-|s(v_r)|+3$. Substituting $|s(v)|+1$ for $|s(v_r)|$, this is $(k-|s(v)|-1)(\frac{w(1-\varepsilon)}{2}+1)+3=\Omega(iw(1-\epsilon))$. □

For the adversary described in the above proof, the largest queue required is $\frac{w(1-\varepsilon)}{2}$. However, we can also demonstrate the following:

Theorem 5: For any w,ε, and $i=\Omega(\frac{1}{1-\epsilon})$, there is a DAG and an $A(w,\varepsilon)$ adversary, such that LIS requires queues of size $\Omega(iw(1-\varepsilon)^2)$, in nodes of level i.

Proof: Call the network N constructed in the previous proof an *adversary tree* of height k. Let s be the largest power of two such that $2s\lceil\frac{1}{1-\epsilon}\rceil+\log s-1\leq i$. For the lower bound on queue size, we shall use s adversary trees, one for each height that is a multiple of $\lceil\frac{1}{1-\varepsilon}\rceil$ between 1 and $m=\lceil\frac{1}{1-\varepsilon}\rceil s$. Each of the roots of these adversary trees also serves as a leaf of a complete binary tree of height $\log s$. Call this binary tree the *root tree*. As before, the adversary will inject $\frac{w(1-\varepsilon)}{2}$ packets to each leaf of every adversary tree.[5] Each such packet is destined for the same node in its respective adversary tree as before, unless that packet is injected at a leaf v for which $s(v)=1^*$, in which case the packet is now destined for the root of the root tree (instead of the root of the adversary tree). All the packets that are injected into a given leaf are injected at the same time step. Let v be a leaf in an adversary tree of height k and with $Z(v)=h$ in its adversary tree. Then the packets injected into v are injected at time step $m-h+(m-k)(\frac{w(1-\varepsilon)}{2}+2)$.

We first demonstrate that this is a valid adversary. We saw in the proof of Theorem 4 that this adversary does not inject more than $w(1-\varepsilon)$ packets that cross any edge of any adversary tree. As to edges of the root tree, observe that for every adversary tree, only packets from one leaf travel into the root tree. The adversary injects into each leaf $\frac{w(1-\varepsilon)}{2}$ packets. Moreover, the injection times of packets into two leaves belonging to two different adversary trees are more than $\lceil\frac{1}{1-\varepsilon}\rceil\frac{w(1-\varepsilon)}{2}\geq w/2$ time steps apart. It follows that for any edge of the root tree, and any consecutive w time steps, at most $w(1-\varepsilon)$ packets are injected.

We now consider an arbitrary leaf of the root tree, denoted x. This node is also a root of an adversary tree; denote the height of that adversary tree by k. We consider the packets that arrive to x and are forwarded to the root of the root tree. By construction, these packets arrive to x from the right child of the right child of x. Denote by y the right child of x and by y_r and y_ℓ the right child and left child of y, respectively. By construction, packets start to be injected into the adversary tree rooted at x at time $m-k+(m-k)(\frac{w(1-\varepsilon)}{2}+2)$. Therefore, by applying Claim 1, we have that for $\frac{w(1-\varepsilon)}{2}$ time steps starting at time $(k-2)\frac{w(1-\varepsilon)}{2}+2k+m-k+(m-k)(\frac{w(1-\varepsilon)}{2}+2)=(m-2)\frac{w(1-\varepsilon)}{2}+3m$, one packet arrives

[5] We assume again that $\frac{w(1-\varepsilon)}{2}$ is an integer; otherwise we use the value $\lfloor\frac{w(1-\varepsilon)}{2}\rfloor$ instead.

to y from each of y_r and y_ℓ in each time step. All these packets are to be forwarded to x. By the same arguments as those used in the proof of Claim 1, the packets arriving from y_ℓ have priority over the packets arriving from y_r. It follows that starting at time $(m-2)\frac{w(1-\varepsilon)}{2} + 3m + \frac{w(1-\varepsilon)}{2} + 1 = (m-1)\frac{w(1-\varepsilon)}{2} + 3m + 1$, $\frac{w(1-\varepsilon)}{2}$ packets destined to the root of the root tree arrive at x, one packet in each time step.

Observe that the arrival time of these packets to x is independent of the identity of the particular leaf, i.e., all leaves of the root tree start receiving such packets at the same time. Denote this time by T, i.e., $T = (m-1)\frac{w(1-\varepsilon)}{2} + 3m + 1$. It is easy to see that any node at distance d from the leaves of the root tree will then forward one packet per time step starting at time $T + d + 1$, and continue doing so until time $T + d + \frac{w(1-\varepsilon)}{2}2^d$. Let c be the child of the root of the root tree that has as a descendant the adversary tree of largest height. The node c is forwarded $\frac{w(1-\varepsilon)}{2}2^{\log s - 2} = \frac{w(1-\varepsilon)s}{8}$ packets destined for the root from each of its children. Thus, it receives $\frac{w(1-\varepsilon)s}{4}$ packets in an interval of time of length $\frac{w(1-\varepsilon)s}{8}$. Since c can only forward one packet per time step, there is a time step where it has a queue of size $\frac{w(1-\varepsilon)s}{8}$. The level of node c is $\ell = 2s\lceil\frac{1}{1-\epsilon}\rceil + \log s - 1$. Since $s = \Omega(\ell(1-\varepsilon))$, in terms of ℓ, the lower bound of $\frac{w(1-\varepsilon)s}{8}$ on the buffer size of c is $\Theta(\ell w(1-\varepsilon)^2)$. Using $i = \Omega(\frac{1}{1-\epsilon})$ and the definition of s, we have that $i/3 \leq \ell \leq i$. If $\ell < i$, we can add dummy nodes until the level of c is i, and thus, we obtain a node with level exactly i with queue size of $\Omega(iw(1-\varepsilon)^2)$. □

4 Conclusions

We considered in this paper the protocol Longest In System when run on DAGs. We obtained linear upper and lower bounds on the queue sizes and maximum delay of packets, when the packets are injected according to the adversarial queuing model. We thus prove the first polynomial bounds (other than on trees and the cycle) in the context of this model, and establish the behavior of LIS on DAGs to be linear. Furthermore, these results separate LIS from other universally stable protocols for which there are exponential lower bounds on DAGs. Comparing our results to the results in [5] also demonstrates that there is an exponential gap between measuring the performance of protocols in terms of the longest path in the network and in terms of the longest path followed by any packet.

As mentioned above, we obtain that LIS is linear on DAGs. It remains however open if LIS is polynomial on general networks.

Acknowledgments We thank Matthew Andrews for discussions on the results in [5]. We also thank Eyal Kushilevitz for useful discussions.

References

[1] C. Álvarez, M. J. Blesa, M. J. Serna, "Universal stability of undirected graphs in the adversarial queueing model.". In *proc. of SPAA 2002*, pp. 183–197.

[2] M. Andrews, Instability of FIFO in Session Oriented Networks. In *Proc. of the 11th SODA*, 2000. pp. 440–447.

[3] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu, "Universal Stability Results for Greedy Contention-Resolution Protocols", *Journal of the ACM*, Vol. 48, No. 1, pp. 39–69, 2001.

[4] M. Andrews, A. Fernández, A. Goel, L. Zhang, "Source Routing and Scheduling in Packet Networks". In *Proc. of 42nd FOCS*, pp. 168–177, 2001.

[5] M. Andrews, and L. Zhang, The Effect of Temporary Sessions on Network Performance. In *Proc. of the 11th SODA*, 2000, pp. 448–457.

[6] M. Andrews, A. Fernández, M. Harchol-Balter, and T. Leighton, L. Zhang, "General Dynamic Routing with Per-Packet Delay Guarantees of O(distance + 1/session rate)," *Siam Journal on Computing*. Vol. 30, No. 5, pp. 1594–1623, 2000.

[7] W. Aiello, E. Kushilevitz, R. Ostrovsky, A. Rosén. Adaptive packet Routing for Bursty Adversarial Traffic. *JCSS*, Vol. 60, No. 3, pp 482–509, 2000.

[8] A.Z. Broder, A.M. Frieze, and E. Upfal, "A General Approach to Dynamic Packet Routing with Bounded Buffers", *Journal of the ACM*, Vol. 48, No. 2, pp. 324–349, 2001.

[9] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Wiliamson, "Adversarial Queuing Theory", *Journal of the ACM*, Vol. 48, No. 1, pp. 13–38, 2001.

[10] A.Z. Broder, and E. Upfal, "Dynamic Deflection Routing on Arrays," In *Proc. of 28th STOC*, pp. 348–355, 1996.

[11] R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory*, pp. 114–131, 1991.

[12] R. Cruz, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Transactions on Information Theory*, pp. 132–141, 1991.

[13] D. Gamarnik, Stability of Adversarial Queues via Fluid Model. In Proc. of the 39th FOCS, pp. 60–70, 1998.

[14] D. Gamarnik. Stability of Adaptive and Non-Adaptive Packet Routing Policies in Adversarial Queuing Networks. In *Proc. of the 31st STOC*, pp. 206–214, 1999.

[15] A. Goel, Stability of Networks and Protocols in the Adversarial Queuing Model for Packet Routing. *Networks*, Vol. 37, No. 4, pp 219–224, 2001.

[16] M. Harchol-Balter and P. Black, "Queuing Analysis of Oblivious Packet-Routing Algorithms," In *Proc. of 5th SODA*, pp. 583–592, 1994.

[17] M. Harchol-Balter and D. Wolfe "Bounding Delays in Packet Routing Networks," *Proc. of 27th STOC*, pp. 248-257, 1995.

[18] M. Mitzenmacher, "Bounds on the Greedy Routing Algorithm for Array Networks", J. Comput. System Sci. 53 (1996), No. 3, pp. 317–327.

[19] G. Stamoulis and J. Tsitsiklis, "The Efficiency of Greedy Routing in Hypercubes and Butterflies," *IEEE Transactions on Communications*, 42 (11), pp. 3051–208, 1994.

[20] C. Scheideler and B. Vöcking, "Universal Continuous Routing Strategies," *Theory of Computing Systems*, Vol. 31, No. 4, pp. 425-2249, 1998.

[21] C. Scheideler and B. Vöcking, "From Static to Dynamic Routing: Efficient Transformations of Store-and-Forward Protocols", *SIAM journal on Computing.* Vol. 30, No. 4, pp 1126–1155, 2000.