

Average-Case Analysis of Greedy Packet Scheduling*

Zvi Lotker Boaz Patt-Shamir

zvilo@eng.tau.ac.il boaz@eng.tau.ac.il

Dept. of Electrical Engineering

Tel-Aviv University

Tel-Aviv 69978

Israel

June 3, 2002

Abstract

We study the average number of delays suffered by packets routed using greedy (work conserving) scheduling policies. We obtain tight bounds on the worst-case average number of delays in a few cases as follows. First, we show that the worst-case average number of delays is a function of the number of sources of packets, which is interesting in case a node may send many packets. Then, using a new concept we call *delay race*, we prove a tight bound on the average number of delays in a leveled graph. Finally, using delay races in a more involved way, we prove nearly-tight bounds on the average number of delays in directed acyclic graphs (DAGs). The upper bound for DAGs is expressed in terms of the underlying topology, and as a result it holds for any acyclic set of routes, even if they are not shortest paths. The lower bound for DAGs, on the other hand, holds even for shortest paths routes.

*A preliminary version of this paper appeared in Proc. *19th ACM Symp. on Principles of Distributed Computing*, July 2000.

1 Introduction

Packet routing schemes can be generally classified as follows. In one model, motivated by parallel machines, the chief objective is to minimize the time to route a given set of packets, in the sense of minimizing the arrival time of the last packet. Many variants of this model exist: see, e.g., [10, 15]. Another avenue of research is motivated by packet-switching data networks, where the main goals are to minimize buffer space, maximize throughput, minimize the number of packets dropped, etc. The former model is called “static routing,” or “one-shot routing,” and the latter is called “dynamic routing,” or “ongoing routing.” Standard networking texts provide many results for this model: see, e.g., [5, 9].

In this work we take a third approach: our main motivation is to better understand the average-case behavior of a packet from a given set. More specifically, we are interested in the average number of delays a packet suffers while being routed to its destination, under very mild assumptions on the way routing is done. The model we consider is the following. The system is synchronous, i.e., there is a global notion of “steps”, such that in each time step, a single packet may cross each link. At the ingress of each link, there is a buffer, where packets are stored until submitted to the link by a link scheduler. We consider the broad class of *greedy* (a.k.a. *work conserving*) link scheduling policies, i.e., policies that always forward a packet over a link if its buffer is not empty.

Our starting point is the work of Mansour and Patt-Shamir [14], where it is proved for the static case that if the paths traversed are shortest paths, then for any scheduling policy, no packet suffers more than $k - 1$ delays, where k is the total number of packets to be routed. Note that $k - 1$ is the best bound in general: if all k packets are simultaneously waiting to the same link, then the last packet to cross the link is delayed $k - 1$ times. Trivially, the $k - 1$ bound on the maximal number of delays implies that the average number of delays is also bounded by $k - 1$. However, it is not clear whether this bound is tight: in the simple one-link scenario just mentioned, the average number of delays is $(k - 1)/2$.

Our Results. Somewhat surprisingly, it turns out that $(k - 1)/2$ is not the worst-case: the average number of delays can approach $k - 1$ arbitrarily closely. In this work, our goal is to relate the average number of delays to the topology of the route set. We present tight bounds in a few cases of interest as follows.

- First, we show that if we assume only that packets traverse shortest paths and that the schedule is greedy, then the average number of delays is $k(1 - 1/2s) - O(s)$, where s is the number of sources (i.e., start nodes of packet routes) in the system.
- For the case where the set of routes induces a leveled graph, we show that the average

number of delays is never more than $(k - 1)/2$. This bound is tight by the one-link example.

- Our most involved result is for the case where the set of routes induces a directed acyclic graph (DAG). In this case we show that even if the routes traversed are *not* shortest paths, the average number of delays per packet under any greedy schedule is bounded by $(k - 1)/2 + O(1)$, where the additive constant depends only on the graph topology, regardless of k . This result is particularly interesting in the dynamic model, where k may be much larger than the size of the graph. Finally, we provide a lower bound that demonstrates that the upper bound is essentially tight, even if the routes are shortest paths.

To analyze leveled and acyclic graphs, we develop a formal mechanism called *delay race*, which may be of independent interest in its own right. In a delay race, each delay creates a token that traverses the network piggy-backed on packets; to bound the average number of delays, we bound the total number of tokens based on their final locations.

Related Work. The work most closely related to this paper is by Patt-Shamir and Mansour [14] mentioned above (which generalizes the results of Cidon *et al.* [7]). A fundamental result for one-shot routing was given by Leighton, Maggs and Rao [11, 12], where they show that the last packet can arrive at its destination in $O(d + c)$ time units, where d is the length of the longest route, and c is the maximal number of routes that use the same edge. Greedy schedules were also considered for packet routing with deadlines by Lui and Zaks [13], and in the context of on-line scheduling by Adler *et al.* [2, 1]. From the dynamic routing viewpoint, it is worth mentioning the model of adversarial queuing theory [6], where the goal is to keep the buffer size bounded under high packet arrival rate. Many results were recently obtained for this model: see, for example, [4, 3]. It seems that adversarial queuing theory is closely related to the average number of delays, since they both count the number of buffer occupancy slots. Many average-case results are known under the assumption that the behavior of the packets is governed by a probability distribution, which is a completely different model (see, e.g., [8] for a modern treatment).

Paper Organization. The remainder of this paper is organized as follows. In Section 2 we describe the model, define basic notation, and give preliminary results. In Section 3 we consider shortest-paths greedy schedules in terms of the number of sources. In Section 4 we describe the delay race mechanism. In Section 5 we consider greedy schedules for leveled graphs. In Section 6 we consider greedy schedule for arbitrary DAGs. We list a few open problems in Section 7.

2 Model, Notation, and Preliminaries

We model the communication network as an unweighted directed graph $G = (V, E)$, where an edge (u, v) represents a unidirectional link from processor u to v . We use $dist(u, v)$ to denote the distance between two nodes in the graph, defined to be the number of edges in a shortest path from u to v . In the scenarios we consider, there is a set of k packets $P = \{p_1, \dots, p_k\}$, and a *schedule*, which maps packets and time steps to nodes in the graph. Intuitively, a schedule describes the location of all packet at all time steps during execution of the packet routing task. In each time step, a packet may progress over a link, under the condition that at most one packet may use each link at each step (i.e., we consider a synchronous model with unit-capacity links). The sequence of edges traversed by a packet in a given schedule is referred to as its *route*. The first node of a route is called the *source* node, and the last node is called the *destination* node.

For a given a schedule, we define the following notation.

- $v(p, t)$: location of packet p at time t .
- $R_p(t, t')$: route traversed by packet p in the time interval $[t, t']$.
- $R_p(v, v')$: route traversed by p from node v to node v' .
- $D_p(t, t')$: number of delays suffered by p in the time interval $[t, t']$, i.e., $D_p(t, t') = t' - t - |R_p(t' - t)|$.
- $p(e, t)$: packet crossing edge e at time $[t, t + 1)$ (undefined if no packet crosses e at that time).
- $ne(p, t)$: next edge on the route of packet p at time t (undefined if p reached its destination by time t).

A schedule is called *greedy* if a packet is never delayed unless another packet is traversing its next edge. Formally, if $v(p, t) = v(p, t + 1)$, then there exists another packet p' with $ne(p, t) = ne(p', t)$ and $v(p', t + 1) \neq v(p', t)$. In this case we say that p' *delayed* p at time t . A schedule is said to be *shortest-paths* schedule if the route traversed by each packet is a shortest path in the underlying graph (for our purposes, given a schedule, we may assume that the graph consists only of edges that were actually traversed).

The following result is key to the current paper.

Theorem 2.1 ([14]) *The number of delays suffered by any packet in any shortest-paths greedy schedule is at most $k - 1$, where k is the total number of packets.*

Theorem 2.1 holds regardless of the initial starting times of the packets. Consider *strict*

priority scheduling: under this policy, each packet is assigned a priority, and the schedule is such that a packet never waits for another packet with lower priority. For this greedy policy, we have the following simple corollary.

Corollary 2.2 *The average number of delays in any shortest-paths strict priority schedule is at most $\frac{k-1}{2}$, where k is the total number of packets.*

Proof: Fix a schedule, and let p_i be the i -th highest priority packet. Consider the schedule obtained by removing all packets of priority lower than p_i : there will be no change at all in the way packets of priority i and higher progress, and in addition, this is still a shortest-paths greedy schedule. By Theorem 2.1, p_i suffers at most $i - 1$ delays in the new schedule, and hence $i - 1$ delays in the original schedule as well. Since this observation holds for any i , we conclude that the average number of delays in the given schedule is at most

$$\frac{1}{k} \sum_{i=1}^k (i - 1) = \frac{\binom{k}{2}}{k} = \frac{k - 1}{2} . \quad \blacksquare$$

3 Average Delay and the Number of Sources

In this section we relate the average number of delays to the number of sources (i.e., nodes from which packets start their routes) in the network. It is shown that the fewer sources there are in the network, a better upper bound exists on the maximal average delay. The bound is proven under the assumption that all packets start moving at the same time. We also prove a lower bound, demonstrating that the upper bound is essentially tight.

3.1 An Upper Bound

We prove the following theorem.

Theorem 3.1 *Suppose that k packets with shortest-path routes start their routes at the same time from s distinct nodes. Then the average number of delays suffered by a packet is at most*

$$k - \frac{k}{2s} - \frac{1}{2} .$$

The proof of the theorem relies on the bound for the worst-case (Theorem 2.1), extended by a simple counting argument. Our first step in the proof is to apply a few simplifications. We use the following lemma.

Lemma 3.2 *Let a shortest-paths greedy schedule S for a graph G be given. Then there exists another shortest-paths greedy schedule S' for a graph G' whose total number of delays is at*

least as big as in S , such that for any two packets p, p' with the same source in S , in S' p, p' have the same source and also share the same last edge on their routes.

We first prove the following technical lemma.

Lemma 3.3 *Let a shortest-paths greedy schedule S be given for a graph G , and suppose that in S the packets p, p' have the same source node. Then there exists another shortest-paths greedy schedule S' in a graph G' with the same number of total delays such that in S' , all packets have the same source and destination nodes as in S , except that p and p' have the same destination node.*

Proof: We construct the schedule S' and its underlying graph as follows. Let d, d' be the lengths of the routes traversed by p, p' in S , respectively.

If p and p' have the same destination in S , then we are done. Otherwise, suppose first that $d \neq d'$, and assume w.l.o.g. that $d > d'$. We introduce a directed path of length $d - d'$ outgoing from the old destination of p' , and set the new destination of p' to be the end of that path. Thus, we henceforth assume that $d = d'$.

Finally, we consider the case that p, p' have distinct destinations, but $d = d'$. In this case we introduce a new node v , and a directed edge from each of the destinations of p, p' to v . To complete the construction, we set v to be the destinations of p, p' . Clearly, in the new schedule the total number of delays, all the sources, and all the destinations with the possible exceptions of the destinations of p, p' remain unchanged. It remains to show that the resulting schedule is a shortest-paths greedy schedule. The greedy nature of the schedule is obvious from the fact that all packets retain their old schedule in the old portion of the graph, and p and p' are not delayed in the new portion of the graph. For the shortest-paths property, first notice that the new edges lead only to new nodes, so that no new routes to destinations of packets other than p, p' are introduced. Finally, note that any path to a new destination can be composed to a path to the old destination, followed by a path to the new destination. Since the new part is unique, and since the path taken to the old destination was shortest, the new route is also shortest. ■

We now prove Lemma 3.2.

Proof of Lemma 3.2: First, note that by inductive application of Lemma 3.3, there exists a schedule S'' such that for any two packets with the same source in S there is a common destination in S'' . Secondly, note that any route can be extended by a single edge outgoing from its destination without decreasing the number of total delays in the schedule. The resulting schedule is shortest-paths and greedy for reasoning similar to the one in Lemma 3.3. ■

We also use the following technical lemma, which is a variant of the Cauchy-Schwarz inequality.

Lemma 3.4 *Let A be any real number, and consider a sequence of positive real numbers $X = (x_1, \dots, x_n)$ with a given sum $K = \sum_{i=1}^n x_i$. Then the sum $\sum_{i=1}^n x_i(A - x_i)$ is maximized when $x_i = K/n$ for all $1 \leq i \leq n$.*

To prove Lemma 3.4, we need the following result.

Lemma 3.5 *Let x, y and A be any real numbers. Then $y(A - y) + x(A - x) \leq (y + x)(A - \frac{y + x}{2})$, with equality iff $x = y$.*

Proof: The inequalities in the following series are equivalent.

$$\begin{aligned} (x - y)^2 &\geq 0 \\ y^2 + x^2 &\geq 2xy \\ -y^2 - x^2 &\leq \frac{-(x + y)^2}{2} \\ y(A - y) + x(A - x) &\leq (y + x) \left(A - \frac{y + x}{2} \right), \end{aligned}$$

with equality if and only if $x = y$. ■

Proof of Lemma 3.4: First, note that since $f(X) \stackrel{\text{def}}{=} \sum_{i=1}^n x_i(A - x_i)$ is concave, f obtains a unique maximum within any closed region. Let X_0 be the sequence of n positive numbers with sum K which maximizes $f(X)$, and suppose, for contradiction, that for two elements x, y in X_0 we have that $x \neq y$. Consider the sequence X_1 obtained from X_0 by replacing x, y with two instances of $\frac{x + y}{2}$. Then we have by Lemma 3.5 that

$$\begin{aligned} f(X_1) - f(X_0) &= (y + x) \left(A - \frac{y + x}{2} \right) (y(A - y) + x(A - x)) \\ &> 0, \end{aligned}$$

contradicting the maximality assumption for X_0 . ■

We can now prove Theorem 3.1.

Proof of Theorem 3.1: We bound the sum of delays of packets by summing separately for each set of packets with a common source. Let v be any node, and let $P(v)$ be the set of packets that v is their source node. By Lemma 3.2, we can assume w.l.o.g. that all packets in $P(v)$ go through the same last edge. From Theorem 2.1 we have that any packet, and in particular the last packet to reach its destination, cannot have been delayed more than $k - 1$ time units. Since all packets in $P(v)$ start at the same time, and since all these packet traverse the same number of edges (because their routes are shortest and have the

same source and destination), it follows that the second-to-last packet of $P(v)$ to reach its destination was delayed at most $k - 2$ time units, the one before it at most $k - 3$ time units etc. It follows that if $T(v)$ is the total number of delays suffered by packets in $P(v)$, then

$$\begin{aligned} T(v) &\leq (k - 1) + (k - 2) + \cdots + (k - |P(v)|) \\ &= \frac{|P(v)|(2k - 1 - |P(v)|)}{2}. \end{aligned}$$

Therefore, we have that the total number of delays is

$$\begin{aligned} \sum_{v \in V} T(v) &\leq \sum_{v \in V} \frac{|P(v)|(2k - 1 - |P(v)|)}{2} \\ &\leq s \cdot \frac{\frac{k}{s}(2k - 1 - \frac{k}{s})}{2} && \text{by Lemma 3.4} \\ &= k \left(k - \frac{k}{2s} - \frac{1}{2} \right). \end{aligned}$$

It follows that the average number of delays is at most $k - \frac{k}{2s} - \frac{1}{2}$, as required. \blacksquare

3.2 A Lower Bound

We conclude this section with a lower bound on the average number of delays which nearly matches the upper bound proved in Theorem 3.1.

Theorem 3.6 *For any $s \geq 2$ there are infinitely many $k \geq s$ such that there exists a shortest-paths greedy schedule with s source nodes and k packets, and such that the average delay suffered by a packet is*

$$k - \frac{3k}{2s} - \Theta(s).$$

Proof: Consider a directed ring with s nodes, where each node is the source for $z \geq 1$ packets, whose destination is $s - 1$ edges away (i.e., if the ring is oriented counter-clockwise, then the destination of each packet is one edge away from its source, clockwise). Note that the total number of packets is $k = sz$. Consider the greedy scheduling policy “furthest to go,” where packets have priority if they are further from their destinations, and ties are broken arbitrarily. In this case, in the first z steps each packet progresses one edge, in the next z time steps each packet progresses another edge etc. In each of the steps $(s - 2)z + 1, \dots, (s - 1)z$, s packets arrive at their destination. It follows that the average number of delays suffered by a packet in the schedule is

$$(z - 1)(s - 2) + \frac{z - 1}{2} = k \left(1 - \frac{3}{2s} \right) - s + \frac{3}{2}. \quad \blacksquare$$

Note that for $s = \sqrt{k}$, the average number of delays is more than $k(1 - \frac{5}{2s})$, very close to the upper bound of Theorem 3.1.

4 Delay Races

Our main tool in analyzing the number of delays in schedules in leveled and acyclic graphs is the concept of *delay race*. The idea is as follows. Each time a packet is delayed, a *delay token* is created. Delay tokens are characterized by the identity of the packet delayed and the time step in which the delay occurred. The packet whose delay created the token is called the *token generator*. Delay tokens move in the system piggy-backed on packets, or may also not move in some time steps, according to certain rules that will be explained shortly. It may be helpful to visualize packets as trains, and tokens as passengers that either progress on a train, or wait for a train in a station. In our proofs, we count the tokens in the system based on their locations; since each delay creates a token, bounding the number of tokens allows us to bound the total (and hence the average) number of delays in a given schedule.

We remark that delay races resemble the concepts of *time path* [14], and *time sequence* [11], but there is an important difference: delay races can be computed on-line, which makes them usable in actual protocols, while time paths and time sequences can be computed only off-line, and thus they can be useful only for analysis. We do not explore this direction further in the current paper.

The rules guiding delay tokens are as follows.

- If a packet p is delayed at time t , then a token $\delta = (t, p)$ is created. The packet p is called the *generator* of δ , denoted $p(\delta)$, and the time step t is called the *creation time* of δ , denoted $t(\delta)$.
- A token progresses on its next edge (see below) only when a packet is progressing on that edge. If a token δ progresses on edge e at time t , the packet that progresses on e at t is called the *carrier* of δ , denoted $c(\delta, t)$.
- The *next edge* to be traversed by a token is the next edge in the route of its latest carrier. The next edge of a token δ at time t is denoted $ne(\delta, t)$. The next edge of a newly generated token is the next edge of its generator packet, i.e., $ne(\delta, t(\delta)) = ne(p(\delta), t(\delta))$. If the last carrier of δ has reached its destination at time t , then $ne(\delta, t')$ is undefined for all $t' \geq t$, and δ does not progress after t .
- At each time step, a packet may be the carrier of at most one delay token for each generator, i.e., if $c(\delta, t) = c(\delta', t)$ then $p(\delta) \neq p(\delta')$.

- At each time step, among the (possibly empty) set of tokens with the same generator and next edge (and hence the same location), only the token with the smallest creation time progresses.

It is convenient to generalize to delay tokens some of the notation originally introduced for packets.

- $v(\delta, t)$ is the location of a delay token δ at time t .
- The route traversed by δ between a and b is denoted $R_\delta(a, b)$, where a and b are either both time points or both nodes.

We define the following concepts.

Definition 4.1 *Let δ be a delay token in a given schedule.*

- *The rank of δ is*

$$\text{rank}(\delta) \stackrel{\text{def}}{=} |\{\delta' \mid p(\delta') = p(\delta) \text{ and } t(\delta') \leq t(\delta)\}| .$$

- *The number of delays suffered by δ by time t is*

$$D_\delta(t) \stackrel{\text{def}}{=} t - t(\delta) - |R_\delta([t(\delta), t])| .$$

Note that the paths traversed by delay tokens are not necessarily shortest paths. The following definition quantifies by how much does a token path deviate from the shortest path.

Definition 4.2 *The length of the bypass done by a delay token δ by time t is*

$$B_\delta(t) \stackrel{\text{def}}{=} |R_\delta([t_0, t])| - \text{dist}(v_0, v) ,$$

where $t_0 = t(\delta)$ is the creation time of δ , $v_0 = v(\delta, t_0)$ is the location of the creation of δ , and $v = v(\delta, t)$ is the location of δ at time t .

We now state a few simple properties we use later. The following lemma relates the number of delays and the bypass length of a delay token.

Lemma 4.1 *Let δ be a delay token with creation time t_0 . Then $B_\delta(t) = t - t(\delta) - D_\delta(t_0, t) - \text{dist}(v(\delta, t_0), v(\delta, t))$.*

Proof: Follows from the fact that $|R_\delta(t_0, t)| = t - t(\delta) - D_\delta(t_0, t)$. ■

The lemma below says that the bypass length is a monotonically increasing function of time.

Lemma 4.2 *For all delay tokens δ and all times t, t' , we have that if $t \leq t'$ then $B_\delta(t) \leq B_\delta(t')$.*

Proof: Let $t_0 = t(\delta)$, and denote $v_0 = v(\delta, t_0)$, $v = v(\delta, t)$ and $v' = v(\delta, t')$. Let $R = R_\delta(t_0, t)$ and $R' = R_\delta(t, t')$. Then we have

$$\begin{aligned}
B_\delta(t') &= |R_\delta(t_0, t')| - \text{dist}(v_0, v') \\
&\geq |R| + |R'| - (\text{dist}(v_0, v) + \text{dist}(v, v')) \\
&= B_\delta(t) + (|R'| - \text{dist}(v, v')) \\
&\geq B_\delta(t) . \quad \blacksquare
\end{aligned}$$

The following lemma expresses intermediate route lengths in terms of distances and bypass lengths.

Lemma 4.3 *Let δ be a any delay token, and denote $t(\delta) = t_0$ and $v(\delta, t_0) = v_0$. For any times $t \geq t' \geq t_0$ with $v(\delta, t) = v$ and $v(\delta, t') = v'$, we have $|R_\delta(t', t)| = \text{dist}(v_0, v) + B_\delta(t) - \text{dist}(v_0, v') - B_\delta(t')$.*

Proof: Follows from the fact that $|R_\delta(t', t)| = |R_\delta(t_0, t)| - |R_\delta(t_0, t')|$, and Def. 4.2. \blacksquare

5 Levelled Graphs

In this section we consider the case where the set of routes induces a leveled graph, i.e., the nodes can be partitioned into sets called *levels*, and numbered $0, 1, 2, \dots$ such that a packet may only progress from a node in level i to a node in level $i + 1$ for some i . We prove that in this case, the average number of delays is at most $\frac{k-1}{2}$, just like the simplest case, where the routes of all packets is the same single edge.

The main idea in the proof of this section is that in a delay race on a leveled graph, tokens are never delayed. To facilitate the proof, we first make the following simplification: we assume that all packets arrive at their respective destinations at the same time t^* . This assumption is justified by the following transformation: Suppose that a packet p arrives at its destination node v , but there is still another packet in the schedule not yet in its destination. In this case we add a new node u and an edge (v, u) , and make u the new destination of p . We apply this transformation inductively, resulting in a larger graph, but preserving the leveled nature of the graph, the greedy nature of the schedule, and the total number of delays.

We start with the following lemma for greedy schedules over leveled graphs. Let $\ell_x(t)$ be the level number of x at time t , where x is either a packet or a delay token.

Lemma 5.1 *At any time $t \leq t^*$, for any packet p , there is at most one delay token generated by p at each level greater than $\ell_p(t)$. Moreover, each token has a carrier.*

Proof: By induction on time. For the base case, there are no delay tokens and the result is trivial. For the inductive step, assume that the lemma holds at time t and consider time $t + 1$. By the induction hypothesis, each token δ has a carrier at time t . Since no packet stops before time t^* , there must be a packet which traverses the next edge of δ at time t . Moreover, by the induction hypothesis, there is no other token to compete with δ over that carrier; it follows that δ will progress one level forward and will have a carrier. Consider now a packet p . There are two cases to consider. If p progresses, its level number increases and the result follows from induction. If p is delayed at time t , then it generates a new delay token δ' . Let δ'' be the “last” delay token generated by p . Since by induction, δ'' is at least (in fact, exactly) one level ahead of p in time t , and since δ'' progresses one level by time $t + 1$, it we have that the newly generated token δ' will not be in the same level as δ . Moreover, δ' will not be on the same level as p , by taking the packet that delayed p to be its carrier. ■

We can now prove the main theorem of this section.

Theorem 5.2 *The average number of delays suffered by a packet in a leveled graph is at most $\frac{k-1}{2}$.*

Proof: To prove the result, consider the system at time t^* . Denote

$$N(p) \stackrel{\text{def}}{=} |\{l > \ell_p(t^*) \mid l = \ell_q(t^*) \text{ for some packet } q\}|.$$

In words, $N(p)$ is the number of “populated” levels ahead of p at time t^* . By Lemma 5.1 we have that the total number of delay tokens, and hence the total number of delays suffered by a packet p is at most $N(p)$, since each token must have a carrier, and there can be at most one token in each level. It follows that the total number of delays over all packets is at most

$$\sum_{p \in P} N(p) \leq 0 + 1 + 2 \cdots + k - 1 = \binom{k}{2}. \quad \blacksquare$$

6 Directed Acyclic Graphs

In this section we consider the case where the routes traversed by the packets induce a directed acyclic graph (DAG) on the network. We first prove, using the delay race methodology, that the number of delays suffered by an average packet when traversing a DAG, under any greedy schedule, is never more than $\frac{k}{2} + O(1)$. The additive term depends only on the topology of the routes, and not on the number of packets using them. As a consequence, the result does *not* depend on the paths being shortest: acyclicity suffices. This result is particularly interesting in the context of dynamic routing problems, where packets are continuously

generated and delivered, but the graph remains fixed. We then prove a lower bound on the average delay in a DAG, which nearly matches the upper bound; the lower bound applies even for shortest-paths routes.

6.1 An Upper Bound

We prove the following theorem.

Theorem 6.1 *For any greedy schedule for k packets whose route set is acyclic (but not necessarily shortest paths), the average number of delays per packet is at most $\frac{k}{2} + |V|(|L| + 1) + |E|$, where L is the length of the longest path in the network.*

To prove the theorem we use a delay race, and split the delay tokens generated by each packet into two sets. One set of tokens will be counted similarly to the case of leveled graphs (Theorem 5.2), and the remainder will be charged against elements of the graph topology. The second part is essential because unlike the simple case of leveled graphs, in general DAGs it is possible for a delay token to be delayed.

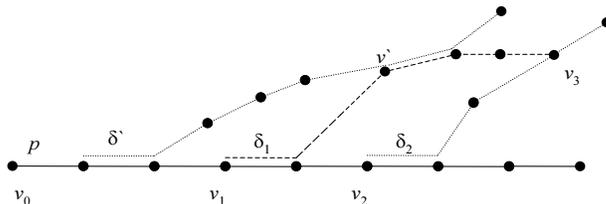


Figure 1: Scenario considered in the proof of Theorem 6.1. All progress is from left to right.

We first relate delayed delay tokens to the graph topology. Our main focus for this case is the following scenario (see Figure 1). A packet p , originating from node v_0 , suffers its r_1 -st delay at time t_1 in node v_1 , creating a delay token δ_1 . It then suffers its r_2 -nd delay at time $t_2 > t_1$ in node v_2 , creating a delay token δ_2 . The delay tokens later meet at time t_3 in node v_3 .

We use the following additional notation.

Notation 6.1

$$\Delta_{\delta_2}(\delta_1, t_3) \stackrel{\text{def}}{=} \text{rank}(\delta_2) - \text{rank}(\delta_1) + B_{\delta_2}(t_3) + D_{\delta_2}(t_3) .$$

Intuitively, $\Delta_{\delta_2}(\delta_1, t_3)$ measures “wasted steps:” first, of p between v_1 and v_2 (since by definition, the difference of the ranks is exactly the number delays in the interval); and then, of δ_2 between v_2 and v_3 .

We first prove the following lemma.

Lemma 6.2 *Let δ_1, δ_2 be with the same generator p and creation times $t_1 < t_2$ respectively. Suppose that δ_1 and δ_2 meet at time t_3 at node v_3 . Then*

$$|R_{\delta_1}(t_1, t_3)| \geq \Delta_{\delta_2}(\delta_1, t_3) + \text{dist}(v_1, v_3) - D_{\delta_1}(t_3) .$$

Proof:

$$\begin{aligned} |R_{\delta_1}(t_1, t_3)| &= t_3 - t_1 - D_{\delta_1}(t_3) \\ &= |R_p(t_1, t_2)| + r_2 - r_1 + |R_{\delta_2}(t_2, t_3)| + D_{\delta_2}(t_3) - D_{\delta_1}(t_3) \\ &= r_2 - r_1 + B_{\delta_2}(t_3) + D_{\delta_2}(t_3) - D_{\delta_1}(t_3) + |R_p(t_1, t_2)| + \text{dist}(v_2, v_3) \\ &\geq \Delta_{\delta_2}(\delta_1, t_3) - D_{\delta_1}(t_3) + \text{dist}(v_1, v_3) . \end{aligned}$$

■

Another property we prove is the following.

Lemma 6.3 *Let δ_1, δ_2 be with the same generator p and creation times $t_1 < t_2$ respectively. Suppose that δ_1 and δ_2 meet at time t_3 at node v_3 . Then $B_{\delta_1}(t_3) + D_{\delta_1}(t_3) \geq \Delta_{\delta_2}(\delta_1, t_3)$.*

Proof: By Lemma 4.1, we have that

$$t_3 - t_1 = B_{\delta_1}(t_3) + D_{\delta_1}(t_3) + \text{dist}(v_1, v_3) . \quad (1)$$

On the other hand, consider the path induced by the route of p from time t_1 to time t_2 , followed by the route of δ_2 from time t_2 to time t_3 . For this path, we have that

$$t_3 - t_1 = r_2 - r_1 + B_{\delta_2}(t_3) + D_{\delta_2}(t_3) + |R_p(v_1, v_2)| + \text{dist}(v_2, v_3) . \quad (2)$$

Equating the r.h.s. of Eq. (1) and the r.h.s. of Eq. (2), we get

$$\begin{aligned} B_{\delta_1}(t_3) + D_{\delta_1}(t_3) &= r_2 - r_1 + B_{\delta_2}(t_3) + D_{\delta_2}(t_3) + |R_p(v_1, v_2)| + \text{dist}(v_2, v_3) - \text{dist}(v_1, v_3) \\ &\geq r_2 - r_1 + B_{\delta_2}(t_3) + D_{\delta_2}(t_3) \\ &= \Delta_{\delta_2}(\delta_1, t_3) , \end{aligned}$$

where the last inequality follows from the fact that $|R_p(v_1, v_2)| + \text{dist}(v_2, v_3) \geq \text{dist}(v_1, v_3)$.

■

The following lemma provides the key argument in the proof of Theorem 6.1.

Lemma 6.4 *Let δ_1, δ_2 be with the same generator p and creation times $t_1 < t_2$ respectively. Suppose that δ_1 and δ_2 meet at time t_3 at node v_3 as in Figure 1. Then there exists a path of length at least $\text{rank}(\delta_2) - \text{rank}(\delta_1)$.*

Proof: We prove the lemma by proving the following stronger claim: There exists a path from v_0 to v_3 of length at least $\Delta_{\delta_2}(\delta_1, t_3) + \text{dist}(v_0, v_3)$.

This claim is proved by induction on the rank of δ_1 . Assume first that $\text{rank}(\delta_1) = 1$. In this case δ_1 is never delayed in the time interval $[t_1, t_3]$, and the claim follows from Lemma 6.2.

For the inductive step, assume that the claim holds for $r_1 = l$; we prove the claim for $r_1 = l + 1$. If $D_{\delta_1}(t_3) = 0$, then we are done by Lemma 6.2. Otherwise, δ_1 was delayed in the time interval $[t_1, t_3]$. Let δ' be the delay token which is the last to delay δ_1 before t_3 . Let us denote by t' the time in which δ_1 is delayed by δ' , and let $v' = v(\delta_1, t') = v(\delta', t')$. With this notation, we have that by the choice of δ' ,

$$D_{\delta_1}(t') = D_{\delta_1}(t_3) - 1 . \quad (3)$$

Also, since δ' delays δ_1 , we have that

$$\text{rank}(\delta') \leq r_1 - 1 . \quad (4)$$

Combining Eq. (3) and Eq. (4), we get

$$\text{rank}(\delta_1) - \text{rank}(\delta') + D_{\delta_1}(t') \geq D_{\delta_1}(t_3) . \quad (5)$$

We use this fact later.

Now, the induction hypothesis, applied to δ' and δ_1 , implies that there exists a path R from v_0 to v' of length at least $\Delta_{\delta_1}(\delta', t') + \text{dist}(v_0, v')$. Consider now the path $R \cdot R_{\delta_1}(v', v_3)$ obtained from concatenating R to the path traversed by δ_1 from v' to v_3 . By Lemma 4.3, we have that $|R_{\delta_1}(v', v_3)| = \text{dist}(v_0, v_3) - \text{dist}(v_0, v') + B_{\delta_1}(t_3) - B_{\delta_1}(t')$. Hence we get

$$\begin{aligned} |R \cdot R_{\delta_1}(v', v_3)| &\geq \Delta_{\delta_1}(\delta', t') + \text{dist}(v_0, v') + \text{dist}(v_0, v_3) - \text{dist}(v_0, v') + B_{\delta_1}(t_3) - B_{\delta_1}(t') \\ &= \text{rank}(\delta_1) - \text{rank}(\delta') + D_{\delta_1}(t') + \text{dist}(v_0, v_3) + B_{\delta_1}(t_3) \\ &\geq B_{\delta_1}(t_3) + D_{\delta_1}(t_3) + \text{dist}(v_0, v_3) \\ &\geq \Delta_{\delta_2}(\delta_1, t_3) + \text{dist}(v_0, v_3) . \end{aligned}$$

The second to last inequality follows from Eq. (5), and the last inequality follows from Lemma 6.3. \blacksquare

We can now prove the upper bound for DAGs.

Proof of Theorem 6.1: We bound the total number of delay tokens in the schedule by bounding the total number of delay tokens generated by each packet. Let p be any

packet, and suppose that it reached its destination at time t . Clearly, no more tokens will be generated by p after time t . We partition the tokens generated by p into two sets. Tokens of the *first kind* are the tokens generated by p that stopped progressing by time t , i.e., their last carrier has reached its own destination by that time. Tokens of the *second kind* are all the rest, i.e., tokens which haven't reached the destination of their last carrier. We denote by $\mathcal{F}(p)$ the set of tokens of the first kind generated by p . By the definition of delay race, when a packet reaches its destination, it may be the carrier of at most one token for each possible generator. Hence, if packet p is the i -th to reach its destination, then $|\mathcal{F}(p)| \leq i - 1 + |E|$, since at time t at most $|E|$ packets can reach their destinations. It follows that

$$\sum_{p \in P} |\mathcal{F}(p)| \leq \binom{k}{2} + k|E|, \quad (6)$$

and hence the contribution of tokens of the first kind to the average delay is $\frac{k}{2} + |E|$.

To deal with tokens of the second kind, we use Lemma 6.4 as follows. We count the number of tokens in each node separately. Consider any node v . Let the set of tokens generated by p stored at v at time t be denoted by $\mathcal{G}_p(v)$. We show that $|\mathcal{G}_p(v)| \leq L + 1$, where L is the length of the longest path in the DAG induced by the route set. If $|\mathcal{G}_p(v)| < 2$, we are done. Otherwise, let δ_1, δ_2 be the tokens with the smallest and largest rank in $\mathcal{G}_p(v)$, respectively. By the definition of rank, it follows that $|\mathcal{G}_p(v)| \leq \text{rank}(\delta_2) - \text{rank}(\delta_1) + 1$. On the other hand, by Lemma 6.4, we have that there exists a path in the graph of length at least $\text{rank}(\delta_2) - \text{rank}(\delta_1)$. Since any path in the graph has length at most L , it follows that $|\mathcal{G}_p(v)| \leq L + 1$. To conclude the proof, note that the total number of delays contributed by tokens of the second kind for each packet is therefore at most $|V|(L + 1)$. ■

6.2 A Lower bound

We now prove that the average number of delays in a DAG is quite close to the upper bound proved in Theorem 6.1, even if the paths traversed are shortest.

Theorem 6.5 *There exists shortest-paths greedy schedules for k packets with acyclic route set such that the average number of delays per packet is at $\frac{k}{2} + \Omega(|E|)$.*

Proof: We construct a graph consisting of many copies of a small graph. Consider first the graph depicted in on the left side of Figure 2.

- Packets $2, \dots, k - 1$ share the same horizontal route:

$$v_{1,1} \rightarrow v_{1,2} \rightarrow v_{1,3} \rightarrow v_{1,4} \rightarrow v_{1,5} \rightarrow v_{1,6} \rightarrow v_{1,7}.$$

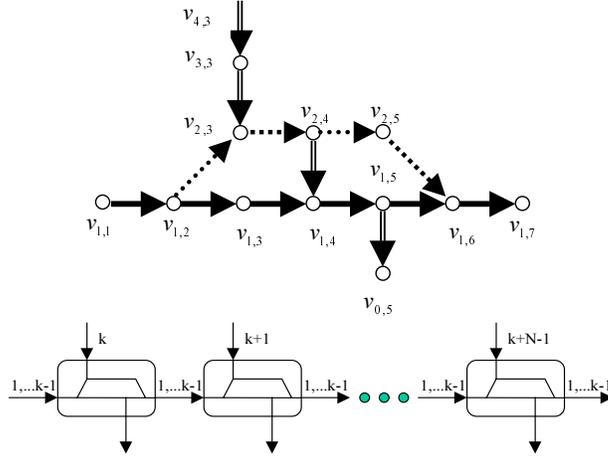


Figure 2: DAGs used in the proof of Theorem 6.5. Top: basic graph. Bottom: cascaded graph. Packet 1 goes over the dotted arrows, packet 2, \dots , $k-1$ go over the boldfaced arrows, and packet $k, \dots, k+N-1$ go over the double arrows (see text for details).

- Packet 1 uses a partly “parallel” route:
 $v_{1,1} \rightarrow v_{1,2} \rightarrow v_{2,3} \rightarrow v_{2,4} \rightarrow v_{2,5} \rightarrow v_{1,6} \rightarrow v_{1,7}$.
- Packet k uses a “cross” route:
 $v_{4,3} \rightarrow v_{3,3} \rightarrow v_{2,3} \rightarrow v_{2,4} \rightarrow v_{1,4} \rightarrow v_{1,5} \rightarrow v_{0,5}$.

Clearly, all routes are shortest paths.

Next, we define the schedule. All packets start together. Packets cross the edge $(v_{1,1}, v_{1,2})$, ordered by their number (contributing $\binom{k-1}{2}$ total delays). Packet k delays packet 1 on the edge $(v_{2,3}, v_{2,4})$, and is then delayed by each packet of $2, \dots, k-1$ on the edge $(v_{1,4}, v_{1,5})$ (thus packet k is involved in $k-1$ additional delays). Finally, on the edge $(v_{1,6}, v_{1,7})$, packet 1 delays packet 2, packet 2 then delays packet 3 etc. (adding $k-2$ delays). It follows that the total number of delays in this schedule is $\binom{k-1}{2} + 2k - 3$.

Next, based on the basic graph (depicted on the left of Figure 2) and the schedule described above, we define another graph and schedule (see Figure 2 right). The new graph is obtained by cascading a series of N copies of the original graph, where we identify node $v_{1,7}$ in copy j with node $v_{1,1}$ in copy $j+1$, for $j = 1, \dots, N-1$. We now define the packets in the new graph and the schedule. Packets $1, \dots, k-1$ are as before: when they reach node $v_{1,1}$ in any copy, their original route is replicated in this copy. To keep the schedule the same, we add a new “vertical” packet for each copy, i.e., a packet which traverses the same route in the local copy as packet k traverses in the original copy; we fix the start times

of the vertical packets so that they will delay packet 1 on their copy of the edge $(v_{2,3}, v_{2,4})$. This will guarantee that all delays of the first copy, except for the delays on edge $(v_{1,1}, v_{1,2})$, are replicated on all copies. It is straightforward to see that in the new schedule,

- the size of the new graph is $|V| + |E| = \Theta(N)$,
- the total number of packets in this setting is $K = k + N - 1$, and
- the total number of delays is $\binom{k-1}{2} + N(2k - 3)$.

Therefore, choosing $N = k/2$ (and hence the number of packets is $K = 3k/2$), we get that the average number of delays for a packet is

$$\begin{aligned} \frac{\binom{k-1}{2} + \frac{k}{2}(2k - 3)}{\frac{3k}{2} - 1} &> \frac{2 \left(\frac{(k-1)(k-2)}{2} + k^2 - \frac{3k}{2} \right)}{3k} \\ &= k + \frac{2}{3k} - 2 \\ &> \frac{2K}{3} - 2. \end{aligned}$$

Since $K = \Theta(N)$, we have that the average number of delays is $\frac{K}{2} + \Omega(|E|)$. ■

There is a gap between the upper bound of Theorem 6.1 and the lower bound of Theorem 6.5. We conjecture that the upper bound can be improved.

7 Conclusion

In this work we have showed that the average number of delay can get arbitrarily close to $k - 1$. On the positive side, we show that very simple means suffice to keep the average number of delays much lower, e.g., strict-priority scheduling, or acyclic routes. Shortest paths do not ensure low average delay, however. It seems interesting to capture the relation between average number of delays and size of the queues. It may also be interesting to investigate ways to exploit delay tokens by routing protocols.

References

- [1] M. Adler, S. Khanna, R. Rajaraman, and A. Rosén. Time-constrained scheduling of weighted packets on trees and meshes. In *Proc. of the 1999 ACM Symposium on Parallel Algorithms and Architecture*, pages 1–12, 1999.
- [2] M. Adler, R. K. Sitaraman, A. L. Rosenberg, and W. Unger. Scheduling time-constrained communication in linear networks. In *Proc. of the 1998 ACM Symposium on Parallel Algorithms and Architecture*, pages 269–278, 1998.
- [3] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosen. Adaptive packet routing for bursty adversarial traffic. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 359–368, 1998.
- [4] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *37th Annual Symposium on Foundations of Computer Science*, pages 380–389, 1996.
- [5] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, Englewood Cliffs, New Jersey, second edition, 1992.
- [6] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queueing theory. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 376–385, 1996.
- [7] I. Cidon, S. Kutten, Y. Mansour, and D. Peleg. Greedy packet scheduling. *SIAM J. Comput.*, 24(1):148–157, 1995.
- [8] F. P. Kelly. *Reversibility and Stochastic Networks*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, Chichester, 1979.
- [9] S. Keshav. *An Engineering Approach to Computer Networking*. Addison-Wesley Publishing Co., 1997.
- [10] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*. Morgan-Kaufman, 1991.
- [11] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 256–269, White Plains, NY, October 1988.

- [12] T. Leighton, B. Maggs, and S. Rao. Fast algorithms for finding $O(\text{congestion} + \text{dilation})$ packet routing schedules. In *Proc. 28th Hawaii International Conference on System Sciences*, January 1995.
- [13] K. S. Lui and S. Zaks. Scheduling in synchronous networks and the greedy algorithm. In *Proceedings of the 11th International Workshop on Distributed Algorithms (WDAG 97)*, Sept. 1997.
- [14] Y. Mansour and B. Patt-Shamir. Greedy packet scheduling on shortest paths. *J. of Algorithms*, 14:449–465, 1993. A preliminary version appears in the *Proc. of 10th Annual Symp. on Principles of Distributed Computing, 1991*.
- [15] L. G. Valiant. *Handbook of Theoretical Computer Science*, pages 943–971. Elsevier/MIT Press, 1990. Chapter 18: *General Purpose Parallel Architectures*.