Greedy Packet Scheduling

Israel Cidon* Shay Kutten† Yishay Mansour‡ David Peleg§

Abstract

Scheduling packets to be forwarded over a link is an important subtask of the routing process both in parallel computing and in communication networks. This paper investigates the simple class of *greedy* scheduling algorithms, namely, algorithms that always forward a packet if they can. It is first proved that for various "natural" classes of routes, the time required to complete the transmission of a set of packets is bounded by the number of packets, k, and the maximal route length, d, for any greedy algorithm (including the arbitrary scheduling policy). Next, tight time bounds of $d+k-1$ are proved for a specific greedy algorithm on the class of shortest paths in n-vertex networks. Finally it is shown that when the routes are arbitrary, the time achieved by various "natural" greedy algorithms can be as bad as $\Omega(d\sqrt{k}+k)$, for any k, and even for $d=\Omega(n)$.

*IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, and Faculty of Electrical Engineering, The Technion, Haifa 32000, Israel. cidon@ee.technion.ac.il

†IBM T.J. Watson Research Center P.O. Box 704, Yorktown Heights, NY 10598, kutten@watson.ibm.com

‡Laboratory for Computer Science, MIT, Cambridge, MA 02139. Partially supported by IBM graduate fellowship. mansour@theory.lcs.mit.edu

§Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot 76100, Israel. peleg@wisdom.weizmann.ac.il. Supported in part by an Allon Fellowship, by a Walter and Elise Haas Career Development Award and by a Bantrell Fellowship. Part of the work was done while visiting IBM T.J. Watson Research Center.

1 Introduction

The task of managing the delivery of packets in a distributed communication network is intricate and complex. Consequently, many routing strategies incorporate design choices directed at simplifying the process. One prime example for this type of choice is the decision to create a clear distinction between two subtasks, namely, *route selection* and *packet scheduling*. The first subtask involves selecting for each packet the route it should use from its source to its destination. This selection is done in advance, before the packet actually leaves its source. The second subtask concerns the transmission stage itself, and involves deciding on the schedule by which the different packets are to be forwarded over each edge along their routes. At this stage, the packets are restricted to their predetermined routes, and cannot deviate from them. This paper concentrates on routing strategies adopting this separation, henceforth referred to as *fixed-route strategies*, and in particular on the scheduling subtask.

A second type of design choice, aimed at simplifying the scheduling process considerably, is to make scheduling decisions *locally* and per packet, rather than globally. The scheduling policy is thus restricted to the selection of local rules for managing the queues on outgoing links, namely, resolving the conflicts between the different packets that need to be advanced on the same outgoing edge. Intuitively, a *local* algorithm has the property that the rules used by a vertex in order to schedule awaiting packets rely only on information concerning these packets (typically contained in the packets' headers), such as the identity of the source and destination, the distance traversed by the packet so far, the arrival time at the current vertex etc. In contrast, a global algorithm can base its decisions on additional global information on the status of the network, such as the current distribution of packets in the network and the routes of these packets.

Although the two design decisions discussed above may not generally lead to a globally optimal algorithm, they are both widely used. In fact, one of the main distributed network strategies for packet routing is *virtual circuit* [CGK88, Mar82, BG87], which is based on fixing a predetermined *logical circuit* from the given end user to the given destination, and transmitting *all* packets between them on this circuit. Nonetheless, similar considerations apply also for the second common routing strategy, known as *datagram routing* [MRR80].

Fixed-route strategies are employed in communication networks such as SNA [Mar82], APPN [BGGJP85] and TYMNET [BG87]. In fact all the emerging integrated high-speed networks proposals such as the international Broadband ISDN (ATM) [CCITT90] and the IBM plaNET Gigabit network [CGGG92], are based on fixed routing strategies. As for the scheduling policy, most networks use a combination of FIFO, certain priority parameters, and flow control information, to determine the next packet to be forwarded. All of these mechanisms are "approximately" local (although flow control adds some global flavor). The main reasons for these choices are based on their advantages from an engineering point of view, namely, their simplicity and low complexity (compared to the global approach), which make them suitable for hardware implementation.

Packet scheduling algorithms for fixed-route strategies were studied by Leighton, Maggs and Rao [LMR88]. Although motivated by routing problems in specific networks realizing parallel machines, their paper studies the problem on networks of arbitrary topology. The first result of [LMR88] is a proof that there exists a schedule that terminates in $O(d + c)$ time, where d is the maximal route length and c is the maximal congestion, i.e., the maximal number of packets that traverse any edge. However, it seems that determining this schedule requires a complex *centralized* computation, relying on global information. The paper provides also some *randomized* distributed protocols for the problem. These protocols are simple, online and local (in the sense discussed above). The first applies to arbitrary sets of paths and requires $O(c + d \log |V|)$ time. The second protocol applies to the case when the paths are *leveled* with l levels. Informally, a set of paths is *leveled* if the vertices of the network can be partitioned into l levels in such a way that each edge of the paths connects two consecutive levels. The protocol completes the routing in $O(c + l + \log |V|)$ time. Both protocols of [LMR88] assume all packets start at the same time.

In contrast with both types of algorithms considered in [LMR88], in this paper we consider the complexity of *deterministic distributed* algorithms. In fact, we concentrate on a class of very simple on-line scheduling algorithms, termed *greedy* algorithms. A *greedy algorithm* is an algorithm satisfying the property that at each time unit, the set of packets that are forwarded is maximal, i.e., if there are messages waiting to be forwarded on some link then one of these messages is forwarded. Note that this includes also an algorithm that selects the message to be forwarded next on each link arbitrarily from among the waiting messages (or alternatively, allows an adversary to decide which packet will be sent next). The class of greedy policies is very natural [Ko78], and in fact, all packet scheduling policies used in practical packet switching networks of which we are aware fall in this class. It is interesting to note that the efficient algorithm of [LMR88] is not greedy.

In the sequel we present several results concerning the behavior of greedy scheduling algorithms. We first look at some restricted path classes. To begin with, in Section 3 we show that for a *leveled* set of paths $\mathcal{P}$, the time required for delivering packet p_i by any greedy algorithm is bounded by $d_i + k - 1$, where d_i is the length of the route of packet p_i and k is the total number of packets. Note that in practical networks k might be much larger than c, hence the randomized algorithm of [LMR88] may perform better.

Our result for the leveled case implies the same result for the natural case of *unique subpaths*. A route collection $\mathcal{P}$ obeys the *unique subpaths* property if for every pair of vertices u and v, all the *subpaths* connecting them in any path of $\mathcal{P}$ are identical.

In Section 4 we consider the class of *shortest paths*. For this class, we present a strategy that guarantees a bound of $d_i + k - 1$ assuming all packets start at the same time. This strategy is based on advancing the packet that has progressed the least so far. A different strategy, based on fixed priorities, was independently proposed in [RVN90], and shown to yield the same bound. In an earlier version of this paper [CKMP90] we conjectured that the same bound is true for any greedy algorithm. This conjecture has recently been resolved in the affirmative [MP91].

We then turn to general route classes. In contrast with the special cases discussed above, we show in Section 5 that greedy algorithms might behave badly for an *arbitrary* set of paths. This is true even when we consider natural greedy schedulers, like fixed priority, FIFO, or preferring the packet that traversed the minimum (or maximum) distance so far. We show that in such a case the time may be $\Omega(d\sqrt{k} + k)$. These negative results hold even for the case where both $k = \Theta(|V|)$ and $d = \Theta(|V|)$. This strengthens the counter-examples given in [LMR88] for the case of long routes and a large number of packets.

2 Model

We view the communication network as a directed graph, $G = (V, E)$, where an edge (u, v) represents a bidirectional link connecting the processors u and v. We assume synchronous communication, i.e., the system maintains a global clock, characterized by the property that a packet sent at time t is received by time $t + 1$.

Next let us define formally the routing problem and its relevant parameters. The input to the problem is a collection $\mathcal{P}$ of k packets p_i and k associated routes ρ_i, $1 \leq i \leq k$. Packet p_i, marked by an identifier I_i, is originated at vertex A_i, its destination is B_i, and it is transmitted along the route ρ_i. We deal with vertex-simple (or, loop-free) routes. The length of the route ρ_i is d_i, and we denote $d(\mathcal{P}) = \max_i\{d_i\}$.

Two packets are said to *collide* at time t if they are currently waiting at the same vertex to be sent over the same link. The scheduling algorithm has to decide at each time t which packet to forward at time t. (Note that the paths are fixed, and hence the algorithm has no choice with respect to the edges that a packet traverses.) Let τ_i^A denote the time at which packet p_i was sent from its originator A_i, and let τ_i^B denote the arrival time of p_i at its destination B_i. Let T_i denote the time elapsing from τ_i^A until τ_i^B, i.e., $T_i = \tau_i^B - \tau_i^A$. The *schedule time* of $\mathcal{P}$ is $T(\mathcal{P}) = \max_i\{T_i\}$.

Some of our results apply only to special path types. Below we characterize these route classes.

A set of paths $\mathcal{P}$ is leveled if there exists an assignment $level : V \to [1, \cdots, |V|]$, such that for each path $\rho = (v_1 \ldots v_l)$, $level(v_j) = level(v_{j-1}) + 1$. A directed graph is leveled if there exists an assignment $level$, such that for every directed edge (u, v), $level(u) + 1 = level(v)$. In a leveled directed graph, every set $\mathcal{P}$ of routes is leveled.

The path ρ_i is a *shortest path* if its length equals the distance between its endpoints A_i and B_i. A set of paths $\mathcal{P}$ is *shortest* if every path $\rho_i \in \mathcal{P}$ is a shortest path.

A set of paths $\mathcal{P}$ has the *unique subpaths* property if for every pair of vertices u and v, all the *subpaths* connecting them in any path of $\mathcal{P}$ are identical; that is, if both the routes ρ_i and ρ_j go through u and v, then the segments of the paths connecting u and v are identical.

3 Leveled routing

In this section we prove our first result, concerning greedy scheduling on leveled paths.

Theorem 3.1 *Let $\mathcal{P}$ be a set of k leveled paths. Then for any greedy algorithm used for routing $\mathcal{P}$,*

1. *every packet p_i arrives within $T_i \leq d_i + k - 1$ time units, and*
2. *the algorithm has schedule time $T(\mathcal{P}) \leq d(\mathcal{P}) + k - 1$.*

Proof: For each packet p_i and $t \geq 0$, let $level(p_i, t)$ denote the number of the level where p_i resides at time t. A level L is said to be *occupied* at time t if there exists a packet p_i such that $level(p_i, t) = L$. The proof is based on considering, for any $t \geq 0$, the set $\mathcal{L}(t)$ of levels that are occupied at time t. We shall argue that at every time unit there is some progress, in the sense that either the number of occupied levels grows, or the lowest occupied level (the one whose number is the smallest) becomes unoccupied.

For uniformity of presentation, we adopt the convention that at any time $t > \tau_i^B$ (the time p_i reaches its destination), $level(p_i, t)$ is incremented by one. This can be thought of as if the packet continues progressing indefinitely along some path ρ_i' extended from the destination

B_i and dedicated to it, and hence never collides afterwards. This does not restrict generality in any way, since such an extension ρ_i' of the packet's route has no influence on the routes of other packets, and the arrival time of the packet is still considered to be τ_i^B, the time it has reached its original destination B_i.

Consider the collection $\mathcal{L}(t)$ of occupied levels at time t. We break this collection into "blocks" of consecutive levels (separated by unoccupied levels). We define the following parameters for each packet p_i:

- $B(p_i, t)$ is the *block* of p_i at time t (i.e., the block containing $level(p_i, t)$).

Suppose that $B(p_i, t) = \{L, L+1, \ldots, H\}$. Then

- $min(p_i, t) = L$.
- $max(p_i, t) = H$.
- $width(p_i, t) = |B(p_i, t)| - 1 = max(p_i, t) - min(p_i, t)$ $(= H - L)$.

Note that the number of occupied levels at any given time t is bounded by the number of packets, $|\mathcal{L}(t)| \le k$, and therefore the maximum block size satisfies

$$width(p_i, t) \le k - 1. \tag{1}$$

Claim 3.2 *$max(p_i, t+1) - max(p_i, t) \ge 1$ for every $t \ge 0$.*

Proof: Since the algorithm is greedy, we are guaranteed that if the levels $L, L+1, \ldots, H$ are occupied at time t, then the levels $L+1, \ldots, H+1$ are occupied at time $t+1$. Also, $L \le level(p_i, t) \le H$ implies $L \le level(p_i, t+1) \le H+1$, and therefore $L+1, \ldots, H+1 \in B(p_i, t+1)$. This implies that $max(p_i + 1, t) \ge H + 1$. ∎

Corollary 3.3 *$T_i \le max(p_i, \tau_i^B) - max(p_i, \tau_i^A)$.*

This corollary is complemented by the following claim which bounds the increase in $max(p_i, t)$ from above.

Claim 3.4 *$max(p_i, \tau_i^B) - max(p_i, \tau_i^A) \le d_i + k - 1$.*

Proof: Consider a packet p_i whose origin A_i is at level $L_A = level(p_i, \tau_i^A)$ and whose destination B_i is at level $L_B = level(p_i, \tau_i^B) = L_A + d_i$. Initially, $max(p_i, \tau_i^A) \ge L_A$. On the other hand, upon arrival at the destination, $max(p_i, \tau_i^B) = min(p_i, \tau_i^B) + width(p_i, \tau_i^B) \le$

$L_B + width(p_i, \tau_i^B)$. Hence by (1) we have that $max(p_i, \tau_i^B) - max(p_i, \tau_i^A) \leq L_B + k - 1 - L_A = d_i + k - 1$ ∎

Combining Corollary 3.3 and Claim 3.4, we get $T_i \leq d_i + k - 1$. This completes the proof of Part (1) of the Theorem. Part (2) follows immediately from Part (1). ∎

The natural class of paths with the unique subpaths property can be analyzed using the above theorem.

Corollary 3.5 *Let $\mathcal{P}$ be a set of k paths satisfying the unique subpaths property. Then for any greedy algorithm used for routing $\mathcal{P}$,*

1. *every packet p_i arrives within $T_i \leq d_i + k - 1$ time units, and*
2. *the algorithm has schedule time $T(\mathcal{P}) \leq d(\mathcal{P}) + k - 1$.*

Proof: We prove that the delay suffered by any packet p_i is no greater than in an execution on a leveled graph (with the same k and d_i). Consider subgraph G_i induced by the route of a particular packet p_i, and consider the subpaths of this single route which are traversed by other packets. Clearly, because of the unique subpath property each subpath is a consecutive segment of the original route, therefore, G_i is leveled. Consider an execution of of the schedule in the original graph ($\mathcal{P}$) and observe the subgraph G_i. Let τ_j^A in G_i be the time at which packet p_j arrived to G_i (or p_i's route) in the above execution. Note, that the schedules of p_i in G_i and in $\mathcal{P}$ are identical. Thus, by Part (1) of Theorem 3.1 the delay suffered by p_i in the unique subpaths case is the same as the one in the leveled paths case we have constructed. ∎

4 Shortest path routing

In this section we consider a scheduling algorithm for the case in which each route ρ_i in the set $\mathcal{P}$ uses a shortest path from its origin to its destination. We shall assume that all packets start at the same time, i.e., $\tau_i^A = 0$ for $1 \leq i \leq k$. For every time $0 \leq t \leq T_i$, let $d_i(t)$ denote the distance traversed by p_i by time t (note that in particular, $d_i(T_i) = d_i$). If p_i and p_j "collide" at time t, the algorithm resolves the collision based on the distance traversed by the packets so far, breaking ties by packet identifiers. Thus the algorithm will prefer p_i iff

$$d_i(t) < d_j(t) \text{ or } (d_i(t) = d_j(t) \text{ and } I_i < I_j).$$

We refer to this algorithm as the *Min Went* algorithm. The rest of this section is devoted to proving the following theorem.

Theorem 4.1 *If the set of paths $\mathcal{P}$ consists of shortest paths and $\tau_i^A = 0$ for $1 \le i \le k$ (i.e., all the packets start at the same time) then the Min Went scheduling algorithm guarantees*

1. *Every packet p_i arrives at time $T_i \le d_i + k - 1$.*
2. *the schedule time is $T(\mathcal{P}) \le d(\mathcal{P}) + k - 1$.*

We begin the proof by pointing out the following trivial fact regarding the relationship between packets in consecutive collisions.

Fact 4.2 *If p_i and p_j collide twice (at times t_1 and t_2), then the relation between $d_i(t)$ and $d_j(t)$ are the same at both times.* ∎

Definition 4.3 *Given an execution of the algorithm the* collision relation C *is defined as the collection of all triples $\langle p_i, p_j, t\rangle$ such that at time t packets p_i and p_j collide (i.e., they are at the same vertex, waiting for the same edge), and p_i wins the collision resolution and gets to use the edge (at time t).*

Since only one packet can go on a specific edge at a time t we can deduce the following fact.

Fact 4.4 *For every p, t there is at most one triple $\langle p', p, t\rangle$ in C.*

Consider some packet p, without loss of generality we term it p_0. If this packet is never delayed, then $T_0 = d_0$ and we are done. Hence suppose the packet was delayed along its route. We now define a *delay sequence* for p_0. Let t_0 be the last time that packet p_0 was delayed. (Note that such a time exists since the delays are finite; a bound of $T(\mathcal{P}) \le k \cdot d(\mathcal{P})$ on the scheduling time of any greedy algorithm is trivial.) Namely, there is a triple $\langle p_1, p_0, t_0\rangle$ in C, and there is no such triple for p_0 in later times $t > t_0$. (Recall that by Fact 4.4 there is only one such p_1.) Let t_1 be the last time p_1 was delayed before time t_0. Namely, there is a triple $\langle p_2, p_1, t_1\rangle$ and no such triple for p_1 in any time between t_1 and t_0. Continue the sequence in this way until reaching a packet p_ℓ that was not delayed prior to time $t_{\ell-1}$.

It is convenient to define also $t_{-1} = T_0$ (the arrival time of p_0) and $t_\ell = 0$ (the start time of p_ℓ).

We get a sequence $\mathcal{DS}$ of triples

$$\mathcal{DS} = \langle p_1, p_0, t_0\rangle,\ \langle p_2, p_1, t_1\rangle,\ \ldots,\ \langle p_\ell, p_{\ell-1}, t_{\ell-1}\rangle,$$

where $T_0 = t_{-1} > t_0 > t_1 > \ldots > t_{\ell-1} > t_\ell = \tau_\ell^A = 0$.

Lemma 4.5 $T_0 \le d_0 + \ell$.

Proof: By definition of the relation C and the Min Went scheduling, we have the inequalities

$(X_j) \quad d_{j+1}(t_j) \le d_j(t_j), \qquad$ for $j = 0, 1, \ldots, \ell - 1$.

For $j = 0, \ldots, \ell$ let θ_j denote the segment of the route ρ_j traversed by p_j between the times t_j (when it "lost" in the collision resolution) and t_{j-1} (when it won), and let $\Delta_j = |\theta_j| = d_j(t_{j-1}) - d_j(t_j)$. Substitute this definition in the inequalities (X_j) to get

$(Y_j) \quad d_{j+1}(t_{j+1}) + \Delta_{j+1} \le d_j(t_j), \qquad$ for $j = 0, \ldots, \ell - 1$.

We also have

$(Y_{-1}) \quad d_0(t_0) + \Delta_0 = d_0(t_{-1}) = d_0$.

Summing the inequalities (Y_j) for $j = -1, 0, \ldots, \ell - 1$, and recalling that $d_\ell(t_\ell) = 0$, we get

$$\Delta_0 + \Delta_1 + \ldots + \Delta_{\ell-1} + \Delta_\ell \le d_0 \tag{2}$$

We also construct a chain of equalities for the times involved in these collisions. Since packet p_j (for $0 \le j \le \ell - 1$) was delayed at time t_j but never delayed since that time until time t_{j-1}, we have

$(Z_j) \quad t_{j-1} = t_j + 1 + \Delta_j, \qquad$ for $j = 0, \ldots, \ell - 1$.

We also have

$(Z_\ell) \quad t_{\ell-1} = d_\ell(t_{\ell-1}) = \Delta_\ell + t_\ell = \Delta_\ell$

Combining the equalities (Z_j) for $j = 0, \ldots, \ell$ we get

$$T_0 = t_{-1} = \Delta_0 + \Delta_1 + \ldots + \Delta_{\ell-1} + \Delta_\ell + \ell. \tag{3}$$

Combining Eq. (2) and (3), we get that $T_{i_0} \le d_{i_0} + \ell$, and the lemma follows. ∎

In order to complete the proof of the theorem, it therefore remains to bound the length of the sequence $\mathcal{DS}$. This is done by proving the following claim.

Lemma 4.6 *The packets p_j appearing in the triples of the sequence $\mathcal{DS}$ are all distinct.*

Proof:

The proof is by a contradiction. Assume that some packet occurs twice in the sequence, for instance $p_m = p_r$ for $m > r$. (See Figure 1.) By the structure of the sequence, every two consecutive packets are distinct, so necessarily $m \ge r + 2$. This means that the sequence contains a subcycle

$\langle p_{r+1}, p_r, t_r \rangle, \ \langle p_{r+2}, p_{r+1}, t_{r+1} \rangle, \ \ldots, \ \langle p_{m-1}, p_{m-2}, t_{m-2} \rangle, \ \langle p_m, p_{m-1}, t_{m-1} \rangle = \langle p_r, p_{m-1}, t_{m-1} \rangle$

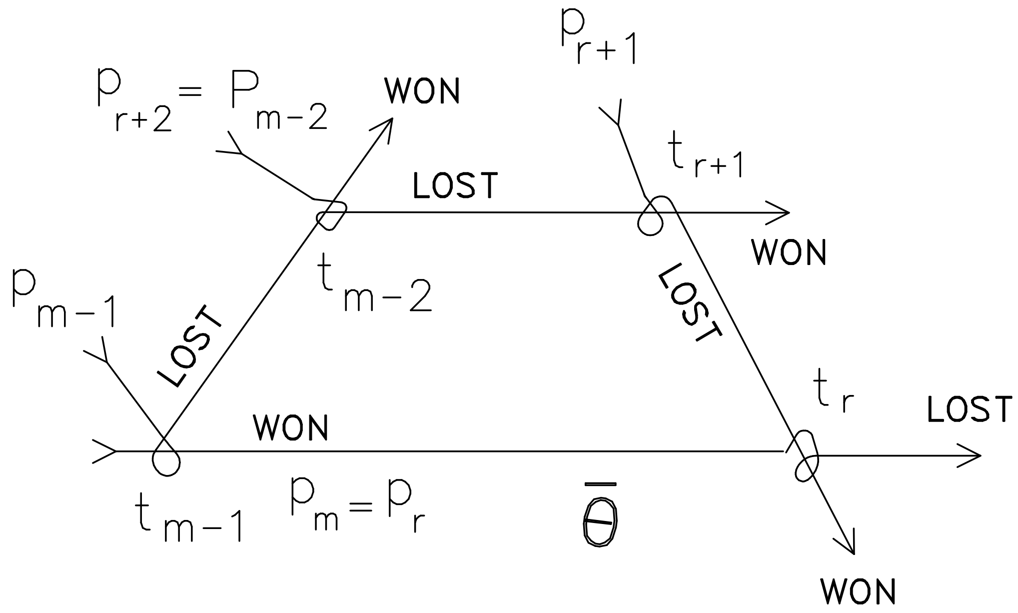

Figure 1: Cycle in delay sequence

where $t_r > t_{r+1} > \ldots > t_{m-1}$, and $m \geq r + 2$.

We argue that among the inequalities (X_j), for $r \leq j \leq m - 1$, at least one of the inequalities is strict. Otherwise, all the collision resolutions in the cycle were made on the basis of packet identities, so $I_r = I_m < I_{m-1} < \ldots < I_{r+1} < I_r$; contradiction. It follows that among the corresponding inequalities (Y_j), for $r \leq j \leq m - 2$, plus (X_{m-1}), at least one is strict. Combine these inequalities in a chain, to get

$$d_m(t_{m-1}) + \Delta_{m-1} + \ldots + \Delta_{r+1} < d_r(t_r). \tag{4}$$

Finally, let $\bar{\theta}$ denote the segment of the route ρ_r traversed by p_r between the times t_{m-1} (when it won) and t_r (when it lost), and let $\bar{\Delta} = |\bar{\theta}| = d_r(t_r) - d_r(t_{m-1})$. We get that

$$\Delta_{m-1} + \ldots + \Delta_{r+1} < d_r(t_r) - d_r(t_{m-1}) = \bar{\Delta},$$

or in other words, the segment $\bar{\theta}$ of ρ_r is not shortest; contradiction to the assumption that all the paths in $\mathcal{P}$ are shortest paths. ∎

Corollary 4.7 *The sequence $\mathcal{DS}$ is of length $\ell \leq k - 1$.* ∎

Combining this corollary with Lemma 4.5 completes the proof of Part (1) of the theorem. Part (2) follows immediately. ∎

5 Greedy algorithms in the general case

The purpose of this section is to demonstrate the fact that, unlike the case of leveled routes, for general route classes not every greedy algorithm delivers the messages fast. We start by constructing a directed graph and a set of paths, that will be used as building blocks for our main lower bound construction.

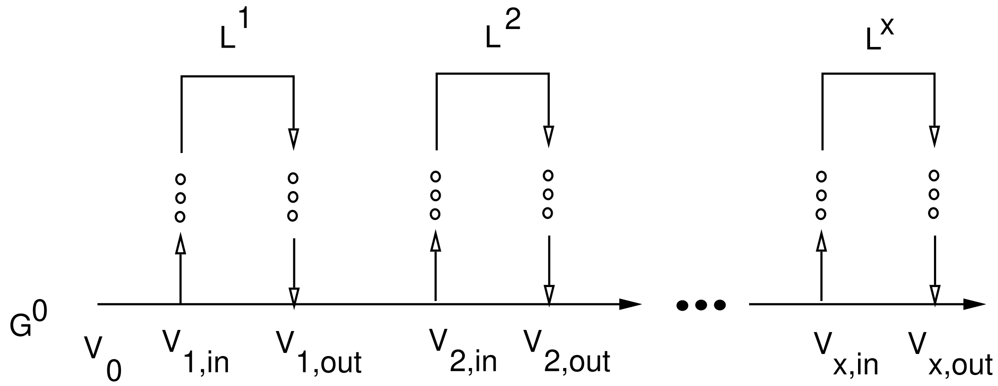

Figure 2: The graph $G_{x,y}$.

The construction is parameterized by two integers x and y. The constructed directed graph $G_{x,y}$ (see Figure 2) is composed of a line of $2x+1$ vertices, denoted by v_0, $v_{<1,\mathtt{in}>}$, $v_{<1,\mathtt{out}>}$, $\cdots$, $v_{<x,\mathtt{in}>}$, $v_{<x,\mathtt{out}>}$. In addition, each "in" vertex (i.e., $v_{<i,\mathtt{in}>}$) is connected to its corresponding "out" vertex (i.e., $v_{<i,\mathtt{out}>}$) also via a "detour" path of y vertices (denoted by L^i). Intuitively, the line is the "main" route; the L_i subgraphs are used to "side track" packets and delay them, thus causing them to collide repeatedly.

A more formal construction follows. Define the graph $G_{x,y} = (V,E)$ to be the union of the subgraphs $G^0, L^1, ..., L^x, C$.

The vertices of the subgraph G^0 are

$$V(G^0) \quad = \quad \{1,\cdots,x\} \times \{\mathtt{in},\mathtt{out}\} \quad \cup \quad \{v_0\},$$

and the edges are

$$\begin{aligned} E(G^0) \quad &= \quad \{(v_{<i,\mathtt{out}>}, v_{<i+1,\mathtt{in}>}) \;:\; 1 \le i \le x-1\} \\ &\cup \quad \{(v_{<i,\mathtt{in}>}, v_{<i,\mathtt{out}>}) \;:\; 1 \le i \le x\} \\ &\cup \quad \{(v_0, v_{<1,\mathtt{in}>})\}. \end{aligned}$$

The graph L is a straight line of y vertices and $y-1$ edges, i.e.,

$$V(L) = \{l_1, \cdots, l_y\} \quad \text{and} \quad E(L) = \{(l_i, l_{i+1}) \;:\; 1 \le i \le y-1\}.$$

For $1 \leq i \leq x$, the subgraph L^i is a copy of the graph L, with superscript i.

Finally, the edges in C connect the endpoints of each subgraph L^i to the corresponding vertices $v_{<i,\mathtt{in}>}$ and $v_{<i,\mathtt{out}>}$ on the path G^0.

$$C = \{(v_{<i,\mathtt{in}>}, l_1^i) \; : \; 1 \leq i \leq x\} \cup \{(l_y^i, v_{<i,\mathtt{out}>}) \; : \; 1 \leq i \leq x\}$$

Let

$$G_{x,y} = G^0 \cup C \cup \bigcup_{1 \leq i \leq x} L^i.$$

The paths $\mathcal{P}_{x,y}$ are the following. All the paths start at vertex v_0, and proceed through $v_{<i,\mathtt{in}>}$ and $v_{<i,\mathtt{out}>}$, for $i = 1, \ldots, x$, ending at $v_{<x,\mathtt{out}>}$. There are two types of paths: a "long" path, pl, consisting of $x(y+2)$ edges, and a "short" one, ps, consisting of $2x$ edges. The "short" path ps travels directly through the graph G^0. The "long" path pl takes all the "detours" L^i, i.e., it travels from each "in" vertex $v_{<i,\mathtt{in}>}$ to the corresponding "out" vertex $v_{<i,\mathtt{out}>}$, via the path L_i. Formally, $ps = E(G^0)$; and

$$pl = \bigcup_{1 \leq i \leq x} E(L^i) \cup \{(v_{<i,\mathtt{out}>}, v_{<i+1,\mathtt{in}>}) \; : \; 1 \leq i \leq x-1\} \cup \{(v_0, v_{<1,\mathtt{in}>})\} \cup C.$$

Let us first illustrate the use of this construction by proving a weaker lower bound, and then proceed to our main lower bound. The scheduling policy that we use for the queue scheduling is a fixed priority. This lemma will later be used to prove our main lower bound.

Lemma 5.1 *For any x and y there exist a graph $G_{x,y}$ and a collection of $y+1$ paths $\mathcal{P}_{x,y}$, such that there is a packet that traverses a path of length $O(x)$ and requires $\Omega(xy)$ time under the fixed priority queueing policy.*

Proof: Given x and y construct $G_{x,y}$ as above. The set $\mathcal{P}_{x,y}$ consists of $y+1$ paths, of which y are identical to the "long" path pl, and the remaining one is the "short" path ps. The last packet has the lowest priority. Note that whenever the low priority packet reaches an "out" vertex it is delayed y times, once by each other packet. It is also delayed y times in vertex v_0. Therefore, the total delay of this packet is $2x + xy$. ∎

The above lemma shows that for some packet the delay may be $\Omega(xy)$, even though the number of edges in its route is only $2x$. This lower bound can be strengthened in two respects. Note that in the above construction, some packets have routes of $\Omega(xy)$ edges, which is as high as the lower bound. Also, the number of vertices in the constructed graph is large (i.e., $\Omega(xy)$). In the setting of the next theorem, both the graph size and the route lengths are small relative to the derived lower bound.

Theorem 5.2 *For every d and k there exist a graph $\mathcal{G}_{d,k}$ and a collection of k paths $\mathcal{P}$ whose schedule time under the fixed priority algorithm is $T(\mathcal{P}) = \Omega(d\sqrt{k})$, and $\mid V(\mathcal{G}_{d,k}) \mid = O(d)$.*

Proof: First we create a graph that achieves the delay bound, but has more vertices, and then we show how to reduce the number of vertices. Consider the graph $G_{x,y}$ defined in Lemma 5.1, with the parameters $x = d/\sqrt{k}$ and $y = \sqrt{k}$. (We assume for simplicity that x and y are integral; if $x < 1$ then $k > d\sqrt{k}$, in which case the claim is trivial.) We take $\sqrt{k}$ copies of $G_{x,y}$, denoted by $G^i_{x,y}$, and connect the ith copy to the $(i+1)$st copy by identifying the last vertex of $G^i_{d,k}$ with the first vertex of $G^{i+1}_{d,k}$ (i.e., $v^i_{<x,\mathtt{out}>} \equiv v^{i+1}_0$).

We partition the packets into $\sqrt{k}$ groups of decreasing priorities, each group is of $\sqrt{k}$ packets. All packets go from $v^1_{<1,\mathtt{in}>}$ to $v^{\sqrt{k}}_{<\sqrt{k},\mathtt{out}>}$. The paths of all the packets within the same group are identical. The ith group traverses in G^j, $j \neq i$, the shortest path from the first vertex to the last vertex (i.e., ps). In G^i the packets traverse all the loops L^j (i.e., use pl). Note that all the paths are of the same length, which is $O(d)$.

Note that the packets in the ith group delay the packets in groups $j > i$ by $\Omega(d)$ time while traversing G^1 (and a similar thing happens as they traverse G^l for $l = 2, 3, \ldots, i-1$). This means that the last packet reaches its destination after $\Omega(d\sqrt{k})$ time.

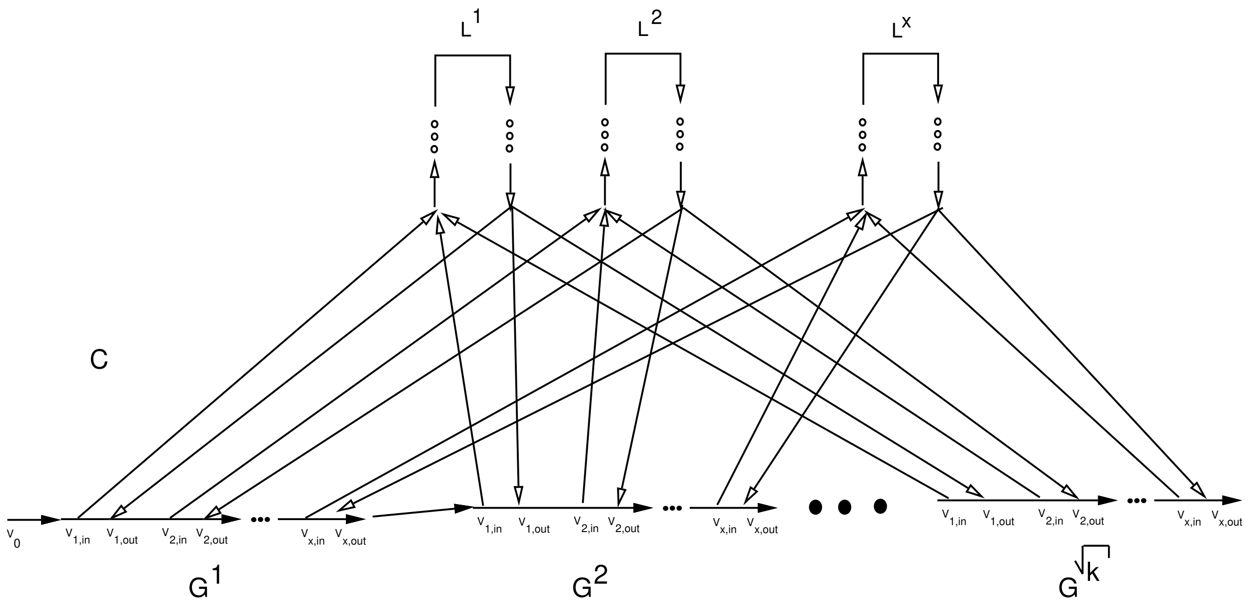

Figure 3: The graph $\mathcal{G}_{d,k}$.

The number of vertices in the construction so far is $O(d\sqrt{k})$. In order to reduce the number of vertices to $O(d)$ we modify the way in which we combine the $\sqrt{k}$ copies of $G_{x,y}$. For every i, we collapse the $\sqrt{k}$ subgraphs L^i in the graphs $G^1, \ldots, G^{\sqrt{k}}$ into a single copy, namely, L^i of G^1. See Figure 3. ∎

A similar, although somewhat more complicated proof can be constructed for other greedy algorithms, specifically the Min Went policy of Section 4, the analogous *Max Went* policy, or the *FIFO policy*, namely, the algorithm that resolves a collision between two packets in vertex v by sending the first to have arrived at vertex v. Details can be found in [CKMP90].

Acknowledgment

It is a pleasure to thank Amotz Bar-Noy for working with us in the early stages of this research. We are also grateful to the anonymous reviewers for their excellent comments and for simplifying some of the proofs, notably in section 3.

References

[BGGJP85] A. E. Baratz, J.P. Gray, P.E. Green, J.M. Jaffe, and D.P. Pozenski, SNA Networks of Small Systems, *IEEE Trans. on Comm.* **sac-3**, May 1985, 416–426.

[BG87] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 1987.

[CCITT90] CCITT SG XVIII, Draft Recommendation I.121: Broadband Aspects of ISDN, Geneva, January 1990

[CGGG92] I. Cidon, I. S. Gopal, P. M. Gopal, R. Guerin, J. Janniello and M. Kaplan, The plaNET/ORBIT High Speed Network, IBM Research Technical Report, RC18270, Aug. 1992.

[CKMP90] I. Cidon, S. Kutten, Y. Mansour and D. Peleg, Greedy Packet Scheduling, *Proc. 4th WDAG*, Bari, Italy, Sept. 1990. LNCS Vol. 486, Springer Verlag, pp. 169–184.

[CGK88] I. Cidon, I. Gopal and S. Kutten, New Models and Algorithms for Future Networks, *Proc. 7th Annual ACM Symp. on Principles of Distributed Computing*, Toronto, Canada, August 1988, 74–89.

[Ko78] H. Kobayashi, *Modeling and Analysis*, Addison-Wesley, 1978.

[LMR88] T. Leighton, B. Maggs, and S. Rao, Universal Packet Routing Algorithms, *Proc. 29th IEEE Symp. on Foundations of Computer Science*, White Plains, NY, October 1988, 256–269.

[MP91] Y. Mansour and B. Patt-Shamir, Greedy packet scheduling on shortest paths, In *Proceedings of the* 10^{th} *Annual ACM Symposium on Principles of Distributed Computing*, August 1991.

[Mar82] J. Martin, *SNA: IBM's Networking Solution*, Prantice Hall, Englewood Cliffs, NJ, 1982

[MRR80] J. McQuillan, I. Richer and E.C. Rosen, The New Routing Algorithm for the ARPANET, *IEEE Trans. on Commun.* **com-28**, May 1980, 711–719.

[RVN90] P. I. Rivera-Vega, R. Varadarajan, and S. B. Navathe, The file redistribution scheduling problem, In *Data Eng. Conf.*, pages 166–173, 1990.