

# Online Packet Routing on Linear Arrays and Rings

Jessen T. Havill

Department of Mathematics and Computer Science  
Denison University  
Granville, OH 43023 USA  
havill@denison.edu

**Abstract.** In contrast to classical offline  $k$ - $k$  routing, the online packet routing problem allows for an arbitrary number of packets with arbitrary end points and release times. We study this problem on linear array and ring networks. We generalize an earlier result for the offline problem by showing that FARTHEST FIRST (FF) scheduling is optimal with respect to makespan on linear arrays. We also show that two other algorithms (LONGEST IN SYSTEM (LIS) and MOVING PRIORITY (MP)) have competitive ratio 2 with respect to makespan on linear arrays. For bidirectional rings, we show that, the competitive ratio of shortest path routing combined with LIS or MP scheduling is in  $[2.5, 3)$  and the competitive ratio of shortest path routing combined with FF scheduling is 2. The latter algorithm is optimal among deterministic memoryless algorithms and all algorithms of which we are aware in the literature.

## 1 Introduction

The problem of efficiently moving packets of data among network nodes has classically been studied in the context of  $k$ - $k$  routing where it is assumed that all packets are known to an algorithm before any packets are sent, and each node sends and receives exactly  $k$  packets. The traditional goal of an algorithm is to construct a schedule for any instance with makespan (maximum completion time) at most equal to the optimal makespan for a worst case instance. In contrast, we are interested in a generalized online problem that differs from the classical problem in three respects. First, we allow an arbitrary number of packets in an instance. Second, we allow any number of packets to originate at or be delivered to any node. Third, we allow packet release times to be arbitrary. We are interested in *oblivious online* algorithms that assign routes at the source before future packets are known. Our algorithms are also *distributed* in the sense that packets are scheduled locally at each intermediate node without regard for packets that have not yet passed through that node.

We will evaluate our algorithms using the competitive ratio (see below) with respect to makespan. We choose makespan for a couple of reasons. First, we wish to compare algorithms for the online problem with known algorithms for

the offline problem, using a similar objective function, to expose interesting similarities and differences between the two problems. Second, even though we allow for arbitrary release times, makespan is still an appropriate measure for many situations, especially if release times are close together. For instance, suppose we want to minimize the completion time of a job on a message passing parallel computer when that job is characterized by a burst of packets. Then minimizing the makespan of the packets may be more important to minimizing the job completion time than minimizing, say, the maximum or average flow time of the packets. Of course, maximum and average flow time are also important criteria to study. We will mention some preliminary observations on these criteria later in the paper and leave further work as an important area for future research.

### 1.1 Problem Definition

In this paper, we consider full-duplex linear array and ring interconnection networks (and, by extension, networks with in-degree one). A linear array network contains  $n$  nodes labeled  $\{0, 1, \dots, n-1\}$  and  $m = 2(n-1)$  directed links  $\{(i, i+1), (i+1, i) : i = 0, 1, \dots, n-2\}$ . A ring network has  $n$  nodes and  $m = 2n$  directed links  $\{(i, (i+1) \bmod n), ((i+1) \bmod n, i) : i = 0, 1, \dots, n-1\}$ . The input to the problem is a sequence of packets  $\sigma = p_1, p_2, \dots, p_k$ , ordered by release time. Each  $p_j = (s_j, t_j, a_j)$ , where  $s_j$  is the packet's source node,  $t_j$  is the packet's destination node, and  $a_j$  is the packet's release time. Let  $P_j$  be the (monotonic) route assigned to packet  $p_j$ , and let  $P_j(i)$  denote the  $i^{\text{th}}$  link in the route.

Each network node with incoming links has a queue for each of its outgoing links to temporarily store packets being forwarded over that link. During each discrete time step  $t \geq 1$ , which represents the continuous time interval  $(t-1, t]$ , a node must decide whether each packet in a link's queue should wait in the queue or be one of at most  $w$  packets forwarded over that link during the next time step. If a packet arrives at a node during time step  $t$ , it will cross the next link on its path no earlier than during time step  $t+1$ . A schedule for packet  $p_j$  is a function  $S_j$  where  $S_j(i)$  is the time step during which packet  $p_j$  will cross link  $P_j(i)$ . We assume that no packet is delayed due to a full queue at its next link. Rather, packets are delayed only when there are already  $w$  packets assigned to the next link or the algorithm decides to hold the packet for another reason. The overall schedule must obey the capacity constraints of the links during each time step.

Let  $C_j$  denote the completion time of packet  $p_j$ . The goal of an algorithm  $A$  is to minimize the makespan of its schedule,  $C_A(\sigma) = \max_j C_j$ , given a request sequence  $\sigma$ . Let  $C^*(\sigma)$  be the optimal makespan for the same request sequence. Then  $A$  is  $c$ -competitive if and only if, for all  $\sigma$ ,  $C_A(\sigma) \leq c \cdot C^*(\sigma) + a$ , where  $a$  is a constant. The competitive ratio of  $A$  is defined to be  $\sup_{\sigma} C_A(\sigma)/C^*(\sigma)$ .

We observe some trivial lower bounds on the optimal makespan for an instance. First, let  $\delta = \max_j \{a_j + |P_j^*|\}$ , where  $P_j^*$  is the route assigned to packet  $p_j$  by an optimal algorithm, and let  $\mu^* = \max_{e \in E} |\{j : e \in P_j^*\}|/w$  be the congestion of the set of routes assigned by an optimal algorithm. Then we know that the

makespan of any optimal schedule must be at least  $\Delta = \max \{\lceil \mu^* \rceil, \delta\}$ . Similarly, define  $\mu$  to be the congestion of the set of routes assigned by a particular online algorithm. If each packet has only one route, then clearly  $\delta = \max_j \{a_j + d_j\}$ , where  $d_j$  is the distance between  $s_j$  and  $t_j$ , and  $\Delta = \max \{\lceil \mu \rceil, \delta\}$ .

## 1.2 Packet Scheduling Algorithms

We will study three packet scheduling algorithms that are all *greedy* in the sense that a packet is delayed at a link  $e$  during time step  $t$  only if  $w$  other packets are already traversing  $e$  during  $t$ . First, LIS (LONGEST IN SYSTEM) [2] is the online scheduling algorithm that schedules each packet as early as possible on each link on its path, in the order it appears in the sequence. In other words, a packet  $p_j$  is given priority over packets  $p_i$  with  $i > j$ . (We note that this algorithm requires a global clock.) The second algorithm, MP (MOVING PRIORITY), gives packets passing through a node priority over packets originating at the node. Unlike LIS, MP does not rely upon a global clock. MP is a relaxed version of LIS on networks with in-degree one since every schedule constructed by LIS could have also been constructed by MP. Lastly, we study the FF (FARTHEST FIRST) algorithm, which gives priority to packets that have the farthest distance yet to travel to their destinations.

## 1.3 Past Research and Our Results

Mansour and Patt-Shamir [8] proved that when packets follow shortest paths, *any* greedy scheduling algorithm guarantees that each  $p_j$  will arrive at its destination within  $d_j + \lfloor (k-1)/w \rfloor$  steps, even in online instances. Cidon, et al. [3] showed that greedy policies cannot achieve this bound on an arbitrary set of paths. LIS was previously studied by Mao and Simha [9] and Rivera-Vega, et al. [10] who showed that LIS achieves the bound in [8] on shortest paths when  $w = 1$ . Valiant and Brebner [12] showed that MP delays each packet at most  $k-1$  times on a linear array. We show that, on networks with in-degree one, LIS ensures that any packet  $p_j$  arrives at its destination by time  $a_j + |P_j| + \lceil \mu_j \rceil - 1$ , where  $\mu_j$  is the maximum number of packets with smaller indices that traverse a link on  $P_j$ , normalized by  $w$ . Similarly, we show that MP guarantees a completion time of  $a_j + |P_j| + \lceil \mu \rceil - 1$ . We use these results to show that each of LIS and MP has a tight competitive ratio of  $2 - 1/\Delta$  on linear arrays. Kaufmann and Sibeyn [5] showed that FF optimally solves the  $k$ - $k$  routing problem on a linear array by constructing a schedule with makespan  $\max_{u < v} \{v - u + h(u, v)\} - 1$  for packets moving either left or right, where  $h(u, v)$  is the number of packets passing through nodes  $u$  and  $v$  from a particular direction. We generalize this result for the online problem by showing that FF constructs an optimal schedule with makespan  $\max_{v, t \leq T_v} \{\lceil N_{\geq}(v, t)/w \rceil + t\} - 1$ , where  $N_{\geq}(v, t)$  is the number of packets that would arrive at node  $v$  at time  $t$  or later if they were not delayed and  $T_v$  is the latest time a packet would arrive at node  $v$  if it were not delayed.

For a related offline problem with deadlines, Lui and Zaks [6] showed a necessary and sufficient condition for an instance to have a schedule that meets all

deadlines on bottleneck-free networks with at most one path connecting any pair of nodes. Furthermore, they showed that if this condition holds, the algorithm that forwards packets with the closest deadlines finds that schedule. Adler, et al. [1] designed a distributed online algorithm that meets at least  $O(1/\log k)$  of the deadlines met in an optimal schedule. For some special cases, their algorithm comes within a constant fraction of optimal.

The well-known worst case lower bound on makespan for the general  $k$ - $k$  routing problem on a ring is  $\max\{(n-1)/2, kn/4\}$ . For the special case in which all  $k$  packets originating at a node have the same destination, Makedon and Symonis [7] designed an algorithm that constructs a schedule with makespan at most  $kn/4 + 5n/2$ . In contrast, they showed that the greedy algorithm that assigns every packet to its shortest path requires at least  $k\lfloor n/2 \rfloor$  steps in the worst case. Kaufmann and Sibeyn [4] designed an algorithm for the general  $k$ - $k$  routing problem on a ring which schedules packets with FF and guarantees a makespan of  $kn/4 + \sqrt{n}$ . Sibeyn [11] designed an algorithm for the online problem with worst case makespan at most  $kn/3 + n/3$  for a  $k$ - $k$  distribution. The key to these ring routing algorithms is to always send some packets on their long paths. However, for the online problem, we show that *none* of these algorithms can have a better competitive ratio than the algorithm that sends every packet on its shortest path and schedules the packets with FF. This algorithm has competitive ratio 2. More specifically, we show that the competitive ratio of an online algorithm is at least 2 if it always assigns a request to its shortest path if the shortest path length is bounded by some fraction of the ring size. Therefore, in order to do better, a deterministic online algorithm must make routing decisions adaptively based on packets seen so far. We also show that the competitive ratio of the online algorithm that uses shortest paths and schedules packets with LIS or MP is in  $[2.5, 3 - 1/\Delta]$ . For the special case in which all packets have the same source and release time, we show that this algorithm is 2 competitive.

## 2 Scheduling with LIS and MP

We begin by proving properties of LIS and MP on full-duplex linear array networks (and more generally, any network with in-degree one) and then use these results to prove bounds on the competitive ratio of algorithms that combine LIS or MP scheduling with shortest path routing on rings.

### 2.1 Linear Array Networks

We will prove the following theorem regarding LIS and MP on linear arrays:

**Theorem 1.** *The competitive ratio of each of LIS and MP is  $2 - \frac{1}{\Delta}$  on networks with in-degree one.*

First, we give two lemmas, one for each algorithm, which state upper bounds on the completion time of any packet. We omit the proof of the second lemma to conserve space. To simplify notation, let  $\mu_j = \max_{e \in P_j} |\{i : i \leq j, e \in P_i\}| / w$ .

**Lemma 1.** LIS guarantees that  $C_j \leq a_j + d_j + \lceil \mu_j \rceil - 1$ , for all  $j$ .

*Proof.* For any  $j$ ,  $C_j = a_j + d_j + x$ ,  $x \geq 0$ . For contradiction, assume  $x > \lceil \mu_j \rceil - 1$ . First, consider  $P_j(l)$ , at the tail of which  $p_j$  is delayed last, and notice that

$$S_j(l) = a_j + l + x. \quad (1)$$

Also notice that our assumption implies that  $l > 1$ . If this were not so then, since LIS is greedy,  $w$  packets with indices less than  $j$  must cross  $P_j(1)$  during each time step  $a_j + 1, a_j + 2, \dots, a_j + x$ , implying that  $\lceil \mu_j \rceil \geq \lceil (xw + 1)/w \rceil = x + 1$ .

Now let  $X(\tau) = \{p_i : i < j \text{ and } p_i \text{ crosses } P_j(l) \text{ during time step } \tau\}$ . Also, to simplify notation, let  $t = S_j(l - 1) + 1$ . Then  $X(t), X(t + 1), \dots, X(S_j(l) - 1)$  are the sets of packets (each with cardinality  $w$ ) that delay  $p_j$  on link  $P_j(l)$ . Additionally, there may be sets  $X(r), X(r + 1), \dots, X(t - 1)$  with cardinality  $w$ . If these additional sets do not exist, then we let  $r = t$ . Formally, we define  $r = \min\{\tau : 2 \leq \tau \leq t, |X(\tau)| = w \text{ for all } \tau, \tau + 1, \dots, t\}$ .

We now prove the following lemma under the assumption that  $x > \lceil \mu_j \rceil - 1$ .

**Lemma 2.** If  $x > \lceil \mu_j \rceil - 1$  then, for all  $p \in X(\tau)$ ,  $r \leq \tau \leq S_j(l) - 1$ ,  $p$  must have crossed link  $P_j(l - 1)$  previously.

*Proof.* Suppose, for contradiction, that there is a packet  $p_i \in X(\tau)$ , for some  $r \leq \tau \leq S_j(l) - 1$ , that did not cross link  $P_j(l - 1)$ . Then,  $P_i(1) = P_j(l)$ . Therefore, by the definition of LIS, if  $p_i$  was delayed, then  $w$  earlier packets must have crossed  $P_j(l)$  during each time step  $a_i + 1, \dots, \tau - 1$ . But since  $w$  packets also cross  $P_j(l)$  during each time step  $\tau, \dots, S_j(l) - 1$  (if  $p_i$  was not delayed, then  $\tau = a_i + 1$ ) and  $p_j$  crosses  $P_j(l)$  during time step  $S_j(l)$ ,

$$\begin{aligned} \lceil \mu_j \rceil &\geq \left\lceil \frac{((S_j(l) - 1) - (a_i + 1) + 1)w + 1}{w} \right\rceil \\ &= (S_j(l) - 1) - (a_i + 1) + 2 \\ &= (a_j + l + x) - a_i && \text{by (1)} \\ &> x + 1 && \text{since } a_j \geq a_i \text{ and } l > 1. \quad \square \end{aligned}$$

Continuing with the proof of Lemma 1, let  $X = \bigcup_{r \leq \tau \leq t} X(\tau) \cup \{p_j\}$ . By Lemma 2 and the fact that  $l > 1$ , the  $(t - r + 1)w + 1$  packets in  $X$  crossed link  $P_j(l - 1)$  before time step  $t$ . Thus, at least one packet  $p \in X$  must have crossed link  $P_j(l - 1)$  before time step  $r - 1$ . Since  $p$  crossed link  $P_j(l)$  during a time step greater than  $r - 1$ ,  $p$  must have been delayed at the tail of link  $P_j(l)$  by  $w$  packets crossing link  $P_j(l)$  during time step  $r - 1$ . This set of packets must be  $X(r - 1)$ . But, by definition,  $X(r - 1)$  contains strictly less than  $w$  packets.  $\square$

**Lemma 3.** MP guarantees that  $C_j \leq a_j + d_j + \lceil \mu \rceil - 1$ , for all  $j$ .

We now use Lemmas 1 and 3 to bound the competitive ratios from above.

**Lemma 4.** The competitive ratio of each of LIS and MP is at most  $2 - \frac{1}{\Delta}$  on networks with in-degree one.

*Proof.* We consider only LIS. The proof for MP is very similar. Let  $p_j$  denote the last packet to complete in the schedule constructed by LIS. By Lemma 1, we know that  $C_j \leq a_j + d_j + \lceil \mu_j \rceil - 1$ . On the other hand, we know that the optimal makespan is at least  $\Delta$ . Therefore, the competitive ratio of LIS is at most  $(a_j + d_j + \lceil \mu_j \rceil - 1)/\Delta \leq (\delta + \lceil \mu \rceil - 1)/\Delta \leq 2 - 1/\Delta$ .  $\square$

We conclude by bounding the competitive ratios from below.

**Lemma 5.** *The competitive ratio of each of LIS and MP is at least  $2 - \frac{1}{\Delta}$  on networks with in-degree one.*

*Proof.* Let  $w = 1$ ,  $k = n - 1$ , and  $p_j = (0, j, 0)$ , for all  $1 \leq j \leq k$ . Note that  $\delta = n - 1$  and  $\mu = \mu^* = n - 1$  since all packets must cross link  $(0, 1)$ . LIS and MP schedule the packets in this instance in order, so that the last packet arrives at node  $n - 1$  at time  $(n - 2) + (n - 1) = 2n - 3$ . On the other hand, an optimal algorithm will schedule the packets in reverse order, achieving a makespan of  $n - 1$ . Thus, the competitive ratio of both algorithms is at least  $(2n - 3)/(n - 1) = 2 - 1/(n - 1) = 2 - 1/\Delta$ .  $\square$

## 2.2 Ring Networks

We define RING1 to be the ring algorithm that routes each packet on its shortest path and schedules packets with either LIS or MP. We prove that the competitive ratio of RING1 is between 2.5 and  $3 - 1/\Delta$ . For the special case where all packets originate at the same node, we can show that the algorithm is 2 competitive. (We will omit the proof of this theorem to conserve space.)

We first prove the general upper bound. Notice that every simple path on a full duplex ring is contained in one of two disjoint simplex rings. We will call the simplex ring with links pointing clockwise the *right ring* and the simplex ring with links pointing counter-clockwise the *left ring*. Let  $A = \{j : P_j = P_j^*\}$  and  $B = \{j : P_j \neq P_j^*\}$ . Furthermore, partition  $A$  into  $A_l$  and  $A_r$  where  $A_l$  is the subset of indices in  $A$  of packets which the online algorithm routes in the left ring and  $A_r$  is the subset of indices in  $A$  of packets which the online algorithm routes in the right ring. Similarly, partition  $B$  into  $B_l$  and  $B_r$ . Therefore, RING1 routes the packets with indices in  $A_l \cup B_l$  in the left ring and the optimal algorithm routes the packets with indices in  $A_l \cup B_r$  in the left ring. A symmetric property holds for the right ring. For a set of requests  $S$ , let  $\mu_S = \max_{e \in E} |\{p_j \in S : e \in P_j\}|/w$ .  $\mu_S^*$  is defined analogously for the optimal route assignment.

**Lemma 6.**  $\mu_{B_l} \leq \mu_{B_l}^*$  and  $\mu_{B_r} \leq \mu_{B_r}^*$ .

*Proof.* We will show that  $\mu_{B_r} \leq \mu_{B_r}^*$ . The argument for  $B_l$  is symmetric. Let  $(i, (i + 1) \bmod n)$  denote a link that satisfies the definition of  $\mu_{B_r}$ . Let  $S \subseteq B_r$  be the subset of requests that are assigned to  $(i, (i + 1) \bmod n)$  in the algorithm's route assignment. So  $\mu_{B_r} = |S|/w$ . Now partition the nodes of the ring into two sets: let  $X = \{\iota : \iota = (i + \lceil n/2 \rceil + 1) \bmod n, (i + \lceil n/2 \rceil + 2) \bmod n, \dots, i\}$  and let  $Y = \{\iota : \iota = (i + 1) \bmod n, (i + 2) \bmod n, \dots, (i + \lceil n/2 \rceil) \bmod n\}$ . Notice that the source of every request in  $S$  is in  $X$  and the destination of every

$ji$	$a_{ji}$	$\alpha(j, i)$	$\frac{n}{2} - j$	$l_{ji}$	$(s_{ji}, t_{ji})$
11	0	3	3	3	(0, 3)
12	0	2	3	3	(0, 3)
13	0	—	—	—	(0, 4)
14	0	—	—	—	(0, 4)
21	1	2	2	3	(1, 4)
22	1	1	2	2	(1, 3)
23	1	—	—	—	(1, 5)
24	1	—	—	—	(1, 5)

$p_{ji}$	$a_{ji}$	$\alpha(j, i)$	$\frac{n}{2} - j$	$l_{ji}$	$(s_{ji}, t_{ji})$
31	2	1	1	4	(2, 6)
32	2	1	1	3	(2, 5)
33	2	—	—	—	(2, 6)
34	2	—	—	—	(2, 6)
41	3	1	0	4	(3, 7)
42	3	1	0	4	(3, 7)
43	3	—	—	—	(3, 7)
44	3	—	—	—	(3, 7)

**Fig. 1.** A small lower bound instance with  $n = 8$ .

request in  $S$  is in  $Y$ , since the requests are following shortest paths in the right ring. Therefore, in the optimal route assignment, since all the requests in  $S$  are sent in the opposite direction, every request in  $S$  must be assigned to the link  $((i + \lceil n/2 \rceil + 1) \bmod n, (i + \lceil n/2 \rceil) \bmod n)$ . Thus,  $\mu_{B_r}^* \geq |S|/w = \mu_{B_r}$ .  $\square$

**Lemma 7.**  $\mu \leq 2\mu^*$ .

*Proof.* The congestion incurred by RING1 is  $\mu \leq \max\{\mu_{A_l} + \mu_{B_l}, \mu_{A_r} + \mu_{B_r}\}$ . Without loss of generality, suppose  $\mu_{A_l} + \mu_{B_l} \geq \mu_{A_r} + \mu_{B_r}$ . Then, by Lemma 6,  $\mu \leq \mu_{A_l} + \mu_{B_l} \leq \mu_{A_l} + \mu_{B_l}^* \leq 2\mu^*$ .  $\square$

**Theorem 2.** RING1 is  $3 - \frac{1}{\Delta}$  competitive on a full duplex ring.

*Proof.* Let  $p_j$  denote the packet that arrives at its destination last in the online schedule. Since the schedule on the full duplex ring is equivalent to two disjoint schedules, each on a simplex ring, we know from Lemmas 1 and 3 that  $C_j \leq a_j + |P_j| + \lceil \mu \rceil - 1 \leq \max_i \{a_i + |P_i^*|\} + \lceil \mu \rceil - 1 \leq \delta + 2\lceil \mu^* \rceil - 1 \leq 3\Delta - 1$ . The second inequality follows because packets are assigned to their shortest routes by the online algorithm. The third inequality follows from Lemma 7 and the definition of  $\delta$ . Thus, the competitive ratio of RING1 is at most  $(3\Delta - 1)/\Delta = 3 - 1/\Delta$ .  $\square$

Finally, we bound the competitive ratio from below by 2.5 as  $n \rightarrow \infty$ .

**Theorem 3.** The competitive ratio of RING1 is at least 2.5.

*Proof.* We define an instance consisting of  $k = 2n$  packets  $p_{ji}$ , where  $1 \leq j \leq \frac{n}{2}$  and  $1 \leq i \leq 4$ . First, let  $\alpha(j, i) = \lfloor (n - (j + i) + 1)/3 \rfloor$  and let

$$l_{ji} = \begin{cases} \min\{j - i + 2, n/2\}, & \alpha(j, i) \geq n/2 - j \\ \alpha(j, i) + 1, & \text{otherwise} \end{cases}.$$

Then, for  $i = 1, 2$ , we define  $p_{ji} = (j - 1, j + l_{ji} - 1, j - 1)$ , and for  $i = 3, 4$ , we define  $p_{ji} = (j - 1, j + n/2, j - 1)$ . In general, RING1 constructs a schedule with makespan  $(5/2)n - 1$  for this instance and there is always a better schedule with makespan  $n$ . For arbitrarily large values of  $n$ , the instance demonstrates a lower bound of  $5/2$  for RING1. For example, consider the instance in Fig. 1 with  $n = 8$ . The schedule created by RING1, displayed in Fig. 2, has makespan 19. However, a better schedule, also displayed in Fig. 2, has makespan 8.  $\square$

	(0,1)	(1,2)	(2,3)	(3,4)	(4,5)	(5,6)	(6,7)	(7,0)
1	$p_{11}$							
2	$p_{12}$	$p_{11}$						
3	$p_{13}$	$p_{12}$	$p_{11}$					
4	$p_{14}$	$p_{13}$	$p_{12}$	$p_{41}$				
5		$p_{14}$	$p_{13}$	$p_{42}$	$p_{41}$			
6		$p_{21}$	$p_{14}$	$p_{13}$	$p_{42}$	$p_{41}$		
7		$p_{22}$	$p_{21}$	$p_{14}$		$p_{42}$	$p_{41}$	
8		$p_{23}$	$p_{22}$	$p_{21}$			$p_{42}$	
9		$p_{24}$	$p_{23}$	$p_{43}$				
10			$p_{24}$	$p_{23}$	$p_{43}$			
11			$p_{31}$	$p_{24}$	$p_{23}$	$p_{43}$		
12			$p_{32}$	$p_{31}$	$p_{24}$		$p_{43}$	
13			$p_{33}$	$p_{32}$	$p_{31}$			
14			$p_{34}$	$p_{33}$	$p_{32}$	$p_{31}$		
15				$p_{34}$	$p_{33}$			
16				$p_{44}$	$p_{34}$	$p_{33}$		
17					$p_{44}$	$p_{34}$		
18						$p_{44}$		
19							$p_{44}$	

	(0,1)	(1,2)	(2,3)	(3,4)	(4,5)	(5,6)	(6,7)	(7,0)	(0,7)	(7,6)	(6,5)	(5,4)	(4,3)	(3,2)	(2,1)	(1,0)
1	$p_{11}$								$p_{13}$							
2	$p_{12}$	$p_{21}$							$p_{14}$	$p_{13}$						$p_{23}$
3		$p_{22}$	$p_{31}$						$p_{23}$	$p_{14}$	$p_{13}$				$p_{33}$	$p_{24}$
4		$p_{11}$	$p_{32}$	$p_{41}$					$p_{24}$	$p_{23}$	$p_{14}$	$p_{13}$		$p_{43}$	$p_{34}$	$p_{33}$
5		$p_{12}$	$p_{21}$	$p_{42}$	$p_{41}$				$p_{33}$	$p_{24}$	$p_{23}$	$p_{14}$		$p_{44}$	$p_{43}$	$p_{34}$
6			$p_{22}$	$p_{31}$	$p_{42}$	$p_{41}$			$p_{34}$	$p_{33}$	$p_{24}$				$p_{44}$	$p_{43}$
7			$p_{11}$	$p_{32}$	$p_{31}$	$p_{42}$	$p_{41}$		$p_{43}$	$p_{34}$						$p_{44}$
8			$p_{12}$	$p_{21}$	$p_{32}$	$p_{31}$	$p_{42}$		$p_{44}$							

Fig. 2. RING1's schedule for the instance in Fig. 1 (top) and a better schedule (bottom).

As stated above, we can show that the algorithm is sometimes 2 competitive.

**Theorem 4.** *If all packets have the same arrival time and source, then any greedy algorithm that uses shortest paths is 2 competitive on a full duplex ring.*

### 3 Scheduling with FF

In this section, we generalize for the online case a result of Kaufmann and Sibeyn [5] by showing that scheduling with FF is optimal on linear arrays. We also show that the algorithm which combines FF with shortest path routing on rings is 2 competitive and optimal among all deterministic online algorithms that always send a packet on its shortest path if it is shorter than some constant fraction of  $n$ . This class of algorithms includes all of those of which we are aware in the literature, including the online algorithm of Sibeyn [11].

#### 3.1 Linear Array Networks

We will say that a packet  $p$  will *ideally* arrive at node  $v$  at time  $t$  if  $t = a_j + |v - s_j|$ . Let  $N(S, v, t) = |\{p_j \in S : v \in P_j \text{ and } a_j + |v - s_j| = t\}|$ , the number

of packets in a set  $S$  that would ideally arrive at node  $v$  at time  $t$ . Also, let  $N_{\geq}(S, v, t) = \sum_{\tau=t}^{\infty} N(S, v, \tau)$  and  $T(S, v) = \max\{t: N(S, v, t) > 0\}$ . When  $S$  is clear from the context, we will let  $N(v, t) = N(S, v, t)$ ,  $N_{\geq}(v, t) = N_{\geq}(S, v, t)$ , and  $T_v = T(S, v)$ . We first improve our trivial lower bound for scheduling on a linear array. We then show that FF achieves this bound and is therefore optimal.

**Theorem 5.** *Any scheduling algorithm constructs a schedule with makespan at least  $\max_{v, t \leq T_v} \{\lceil N_{\geq}(v, t)/w \rceil + t\} - 1$  on a linear array.*

*Proof.* Consider an arbitrary node  $v$ . During each time unit in  $\{1, 2, \dots, T_v\}$ , at most  $w$  packets will reach node  $v$ . For any time unit  $t \leq T_v$ , if there are more than  $(T_v - t + 1)w$  packets that would ideally reach  $v$  at time  $t$  or greater then they could not all have reached  $v$  by time  $T_v$ . Rather,  $N_{\geq}(v, t) - (T_v - t + 1)w$  packets must reach  $v$  after time  $T_v$ . Since this is true for any  $v$  and any  $t \leq T_v$ , the time at which the final packet reaches its destination must be at least

$$\max_v \left\{ T_v + \max_{t \leq T_v} \left\lceil \frac{N_{\geq}(v, t) - (T_v - t + 1)w}{w} \right\rceil \right\} = \max_{v, t \leq T_v} \left\{ \left\lceil \frac{N_{\geq}(v, t)}{w} \right\rceil + t \right\} - 1. \quad \square$$

**Theorem 6.** *The makespan of a schedule constructed by FF on a linear array is equal to  $\max_{v, t \leq T_v} \{\lceil N_{\geq}(v, t)/w \rceil + t\} - 1$ .*

*Proof.* By the previous theorem, the lower bound holds. To prove the upper bound, consider an arbitrary node  $v$  and let  $t_{\max}$  satisfy

$$\left\lceil \frac{N_{\geq}(v, t_{\max})}{w} \right\rceil + t_{\max} - 1 = \max_{t \leq T_v} \left\{ \left\lceil \frac{N_{\geq}(v, t)}{w} \right\rceil + t \right\} - 1. \quad (2)$$

The following inequalities follow from this definition:

$$\sum_{\tau=t}^{t_{\max}-1} N(v, \tau) \leq (t_{\max} - t)w, \text{ for all } t \in \{1, 2, \dots, t_{\max} - 1\} \quad (3)$$

$$\sum_{\tau=t_{\max}}^t N(v, \tau) \geq (t - t_{\max} + 1)w, \text{ for all } t \in \{t_{\max}, t_{\max} + 1, \dots, T_v\} \quad (4)$$

If (3) were not true, then a smaller value of  $t_{\max}$  would satisfy (2). If (4) were not true, then a larger value of  $t_{\max}$  would satisfy (2).

We first show that, by the definition of  $t_{\max}$  and (3), the packets counted in  $\sum_{\tau=1}^{t_{\max}-1} N(v, \tau)$  will arrive at node  $v$  by time unit  $t_{\max} - 1$ . Notice that we can safely ignore any packets not counted in  $N(v, t)$ , for any  $t$ , since packets that will pass through node  $v$  will have priority over those that do not. If at most  $w$  packets would ideally arrive at node  $v$  at some time  $t$ , then they will all arrive at node  $v$  at their ideal time since no other packets will delay them. Now consider a group of  $p > w$  packets that all want to arrive at node  $v$  at some time  $t \in \{1, 2, \dots, t_{\max} - 1\}$ . By (3), for each such group, there must be at least

$p - w$  additional available slots between  $t + 1$  and  $t_{\max} - 1$ , inclusive, at which no packets would ideally arrive. Therefore, each of these  $p$  packets can arrive at node  $v$  at a unique time step at most  $t_{\max} - 1$ .

Next we show that the remaining packets, those that would ideally arrive at  $v$  after time  $t_{\max} - 1$ , will all arrive at  $v$  by time  $t_{\max} - 1 + \lceil N_{\geq}(v, t_{\max})/w \rceil$ . Notice that, by (4), we can adjust the release time of each of these packets so that it will arrive at  $v$  no earlier than in the original schedule, and at a unique time step in  $\{t_{\max}, t_{\max} + 1, \dots, t_{\max} - 1 + \lceil N_{\geq}(v, t_{\max})/w \rceil\}$ . Note that the packets cannot finish any earlier with this modification in any schedule. Therefore, all packets that travel through  $v$  arrive there by time step  $\lceil N_{\geq}(v, t_{\max})/w \rceil + t_{\max} - 1$  and the makespan of the schedule is at most  $\max_{v, t \leq T_v} \{\lceil N_{\geq}(v, t)/w \rceil + t\} - 1$ .  $\square$

This result reduces to that of Kaufmann and Sibeyn [5] if all  $a_j=0$  and  $w=1$ :

$$\max_{v, t \leq T_v} \{N_{\geq}(v, t) + t\} - 1 = \max_{v, t \leq v} \{h(v-t, v) + t\} - 1 = \max_{u < v} \{h(u, v) + (v-u)\} - 1 .$$

### 3.2 Ring Networks

Let RING2 be the ring algorithm that sends each packet on its shortest path and schedules packets with FF. We show that RING2 has a tight competitive ratio of 2 and that this is better than any algorithm that routes a packet on its shortest path if it is shorter than a constant fraction of the ring size.

**Theorem 7.** *The competitive ratio of RING2 is at most 2 on a ring.*

*Proof.* Let  $S_l$  denote the set of packets routed in the left ring by RING2. Without loss of generality, suppose a packet in  $S_l$  has the latest completion time. Then  $C_{FF} = \max_{v, t \leq T_v} \{\lceil N_{\geq}(S_l, v, t)/w \rceil + t\} - 1$ . Let  $v_{\max}, t_{\max} \leq T_{v_{\max}}$  be values that satisfy  $C_{FF}$ . We consider two cases (henceforth omitting  $S_l$  from notation):

**Case 1:**  $\lceil N_{\geq}(v_{\max}, t_{\max})/w \rceil \leq t_{\max} + 1$ .

In this case,  $C_{FF} = \lceil N_{\geq}(v_{\max}, t_{\max})/w \rceil + t_{\max} - 1 \leq 2t_{\max} \leq 2T_{v_{\max}} \leq 2C^*$ .

**Case 2:**  $\lceil N(v_{\max}, t_{\max})/w \rceil > t_{\max} + 1$ .

For contradiction, suppose there exists an instance such that  $C_{FF}/C^* > 2$ .

**Case 2a:** If the optimal schedule directs  $X \leq N(v_{\max}, t_{\max}) - 1$  of the packets that are counted in  $N(v_{\max}, t_{\max})$  to the right then we know that  $C^* \geq \lceil (N(v_{\max}, t_{\max}) - X)/w \rceil + t_{\max} - 1$ . Since  $C^* < (\lceil N(v_{\max}, t_{\max})/w \rceil + t_{\max} - 1)/2$  by assumption, this means that  $X > (\lceil N(v_{\max}, t_{\max})/w \rceil + t_{\max} - 1)/2$ . (Notice that this is valid since, by assumption,  $(\lceil N(v_{\max}, t_{\max})/w \rceil + t_{\max} - 1)/2 < \lceil N(v_{\max}, t_{\max})/w \rceil - 1$ .) But, since all of the packets in  $X$  must traverse the edge  $((v + \lceil n/2 \rceil) \bmod n, (v + \lceil n/2 \rceil - 1) \bmod n)$  in the right ring in the optimal schedule, it must be the case that  $C^* \geq X > (\lceil N(v_{\max}, t_{\max})/w \rceil + t_{\max} - 1)/2$  in the right ring. But this implies that  $C_{FF}/C^* < 2$ , a contradiction.

**Case 2b:** If the optimal schedule directs all of the packets that are counted in  $N(v_{\max}, t_{\max})$  to the right then we know that  $C^* \geq \lceil N(v_{\max}, t_{\max})/w \rceil$  in the right ring in the optimal schedule. This also implies a contradiction:

$$\frac{C_{FF}}{C^*} \leq \frac{\lceil N(v_{\max}, t_{\max})/w \rceil + t_{\max} - 1}{\lceil N(v_{\max}, t_{\max})/w \rceil} < 1 + \frac{t_{\max} - 1}{t_{\max} + 1} < 2 . \quad \square$$

**Theorem 8.** *The competitive ratio of any routing algorithm is at least  $2 - \epsilon$ , for arbitrarily small positive  $\epsilon$ , if it always assigns a request to its shortest path if the shortest path has length at most  $\beta n$ , for any  $\beta \in (0, 1/2]$ .*

*Proof.* Let  $w = 1$ . Consider a sequence of  $k \geq (1 - 2\beta)n$  packets  $(0, \lfloor \beta n \rfloor, 0)$ . The algorithm will assign all these requests to their shortest path, resulting in a schedule with makespan at least  $\lfloor \beta n \rfloor + k - 1$ . On the other hand, a better schedule assigns  $\lceil \alpha k \rceil$  packets to the shortest path and  $\lfloor (1 - \alpha)k \rfloor$  packets to the long path, where  $\alpha = 1/2 + (1 - 2\beta)n/(2k)$ . By Lemma 1, a LIS schedule on these routes will have makespan  $\max\{\lfloor \beta n \rfloor + \lceil \alpha k \rceil - 1, (n - \lfloor \beta n \rfloor) + \lfloor (1 - \alpha)k \rfloor - 1\} \leq (n + k)/2$ . Therefore, the makespan of an optimal schedule will be at most  $(n + k)/2$ , and the competitive ratio for this instance is at least  $2((\lfloor \beta n \rfloor + k - 1)/(n + k))$ , which approaches 2 for an arbitrarily large value of  $k$ .  $\square$

**Theorem 9.** *The competitive ratio of RING2 is exactly 2 on a ring and RING2 is optimal among algorithms which always assign a packet to its shortest path if the shortest path has length at most  $\beta n$ , for any  $\beta \in (0, 1/2]$ .*

### 3.3 Other Objective Functions

The consideration of other objective functions such as maximum flow time ( $\max_j(C_j - a_j)$ ) and total flow time ( $\sum_j(C_j - a_j)$ ) is an important direction for future research. Here we observe that FF is not optimal for these objective functions on linear arrays.

**Observation 10.** *FF is not optimal with respect to total flow time (or total completion time) on a linear array.*

*Proof.* Consider the following instance with  $n = 3$  and  $w = 1$ :  $p_0 = (0, 1, 0)$ ,  $p_1 = (0, 2, 0)$ , and  $p_3 = (1, 2, 1)$ . FF will assign  $p_1$  to  $(0, 1)$  during step 1,  $p_1$  and  $p_2$  to  $(1, 2)$  during steps 2 and 3, and  $p_0$  to  $(0, 1)$  during step 2, giving a total flow time of 6 and total completion time of 7. On the other hand, LIS will assign  $p_0$  to  $(0, 1)$  during step 1,  $p_1$  to  $(0, 1)$  and  $(1, 2)$  during steps 2 and 3, and  $p_2$  to  $(1, 2)$  during step 2, giving a total flow time 5 and total completion time 6.  $\square$

**Observation 11.** *The competitive ratio of FF is arbitrarily poor with respect to maximum flow time on a linear array.*

*Proof.* Consider the following instance on a 2 node linear array with  $w = 1$ :  $p_0 = (0, 1, 0)$  and  $p_j = (0, 1, j - 1)$  for  $j = \{1, 2, \dots, k - 1\}$ . FF may schedule the packets in the following order:  $p_1, p_2, \dots, p_k, p_0$ . The flow time of  $p_0$  is  $k$  in this case, while the optimal maximum flow time is 2.  $\square$

## 4 Conclusions

We have studied how to route and schedule online packet instances on linear arrays and rings. We bounded the delay of two simple scheduling algorithms

(LIS and MP) and proved that each has a competitive ratio of  $2 - 1/\Delta$  on linear arrays. We also generalized for the online case a previous result by showing that FF is optimal on linear arrays. We used these results to analyze the corresponding online algorithms that route packets on their shortest paths on rings. We showed that the ring algorithm that schedules with LIS or MP has a competitive ratio between 2.5 and  $3 - 1/\Delta$ , and the ring algorithm that schedules with FF has a tight competitive ratio of 2. The latter is optimal in the class of “memoryless” algorithms that always send a packet on its shortest path if its length is less than some constant fraction of the ring size. This class of algorithms includes all of which we are aware in the literature. In order to do better, a deterministic online algorithm would need to examine the network state and history, and assign a route and schedule accordingly. The investigation of such algorithms is an interesting area for future research. The study of other objective functions is also an important research direction. Since FF is not optimal with respect to maximum or total flow time, a study of other algorithms is required for these objectives. In addition, it would be interesting to extend this analysis to routing and scheduling on meshes and tori, and networks with arbitrary capacities.

## References

- [1] M. Adler, A. L. Rosenberg, R. K. Sitaraman, and W. Unger. Scheduling time-constrained communication in linear networks. In *Proc. ACM Symp. on Parallel Algorithms and Architectures*, pages 269–278, 1998.
- [2] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. In *Proc. ACM Symp. on Theory of Computing*, pages 376–385, 1996.
- [3] I. Cidon, S. Kutten, Y. Mansour, and D. Peleg. Greedy packet scheduling. *SIAM Journal on Computing*, 24(1):148–157, 1995.
- [4] M. Kaufmann and J. F. Sibeyn. Deterministic routing on circular arrays. In *Proc. IEEE Symp. on Parallel and Distributed Processing*, pages 376–383, 1992.
- [5] M. Kaufmann and J. F. Sibeyn. Randomized multipacket routing on sorting on meshes. *Algorithmica*, 17:224–244, 1997.
- [6] K.-S. Lui and S. Zaks. Scheduling in synchronous networks and the greedy algorithm. In *Proc. Int. Workshop on Distributed Algorithms*, pages 66–80, 1997.
- [7] F. Makedon and A. Symvonis. Optimal algorithms for multipacket routing problems on rings. *Journal of Parallel and Distributed Computing*, 22(1):37–43, 1994.
- [8] Y. Mansour and B. Patt-Shamir. Greedy packet scheduling on shortest paths. *Journal of Algorithms*, 14(3):449–465, 1993.
- [9] W. Mao and R. Simha. Routing and scheduling file transfers in packet-switched networks. *Journal of Computing and Information*, 1(1):559–574, 1994.
- [10] P. I. Rivera-Vega, R. Varadarajan, and S. B. Navathe. Scheduling data redistribution in distributed databases. In *Proc. IEEE Int. Conf. on Data Engineering*, pages 166–173, 1990.
- [11] J. F. Sibeyn. Deterministic routing and sorting on rings. In *Proc. IEEE Int. Parallel Processing Symp.*, pages 406–410, 1994.
- [12] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proc. ACM Symp. on Theory of Computing*, pages 263–277, 1981.