Optimal Online Ring Routing

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

Kevin R. Hutson
Department of Mathematics
Furman University
Greenville, SC 29613
kevin.hutson@furman.edu

Abstract

We study how to route online splittable flows in bidirectional ring networks to minimize maximum load. We show that the competitive ratio of any deterministic online algorithm for this problem is at least $2 - 2/n$, where n is the size of the ring, and that the online algorithm that splits each flow inversely proportionally to the length of the flow's shortest path achieves this competitive ratio for all integers $n \geq 2$.

1 Introduction

In recent years, there has been renewed interest in bidirectional ring networks, especially for Metropolitan Area Networks (MANs) [1, 14]. The ring topology offers two primary benefits. First, the connectivity of the cycle protects against failure of either a node or a link and increases the survivability of the network. In addition, if there is high demand congesting a link, the ring offers an alternative route for communication. Second, the ring topology allows the network resources to be shared in a cost-effective way. There are currently two prevalent modern MAN ring technologies: the Synchronous Optical Network (SONET) ring [14], which is virtual circuit based, and the newer Resilient Packet Ring (RPR) [1], which is packet based but allows for reserved bandwidth transmissions. Both technologies allow for bidirectional traffic in either the clockwise or counterclockwise direction. The ring capacity is equal in both directions and fixed in the design phase. Therefore, it is important to route requested demand efficiently to avoid congestion (load) on any particular link in the system.

Here we consider the problem of routing online splittable flows in ring networks to minimize maximum load. We define a problem instance $(n, \mathcal{F})$ as follows. Let $R = (V, E)$ be a directed ring network with node set $V = \{0, 1, \ldots, n-1\}$ and arc set $E = \{(i, (i+1) \bmod n), ((i+1) \bmod n, i) : i \in V\}$. Let $\mathcal{F} = f_1, f_2, \ldots, f_k$ be a sequence of flow requests on R. Each flow f_j is a triple (s_j, t_j, l_j) where $s_j \in V$ is the source of the flow, $t_j \in V$ is the destination of the flow, and l_j is the bandwidth (or demand) required for the flow. Flows may also have arrival and departure times assigned to them. For each flow f_j, an algorithm must decide what fraction of l_j to route from s_j to t_j in the clockwise direction (involving arcs of type $e_i = (i, (i+1) \bmod n)$) and what fraction to route in the counter-clockwise direction (involving arcs of type $\overline{e_i} = ((i+1) \bmod n, i)$). In this online problem, an algorithm must irrevocably make the decision for each flow f_j independent of knowledge about future flows f_i where $i > j$ (but may use knowledge of previous flows f_i where $i < j$).

For an online algorithm A and flow sequence $\mathcal{F}$, we define the load on arc e at time t, denoted $A_e(\mathcal{F}, t)$, to be the sum of the fractional flows that an algorithm A has assigned to e at time t. The goal of an online algorithm A is to minimize the maximum load $A(\mathcal{F}, t) = \max_{e \in E} A_e(\mathcal{F}, t)$. Let $OPT(\mathcal{F}, t)$ be the optimal maximum load for flow sequence $\mathcal{F}$ at time t. We evaluate the performance of an online algorithm by its competitive ratio, which is the supremum, over all possible request sequences, of the ratio of the maximum load achieved by the online algorithm to the maximum load achieved by an optimal offline algorithm. Formally, we say A is c-competitive if and only if, for all $\mathcal{F}$ and t, $A(\mathcal{F}, t) \leq c\, OPT(\mathcal{F}, t)$. The competitive ratio of A is then $\sup_{\mathcal{F},t} A(\mathcal{F}, t)/OPT(\mathcal{F}, t)$.

In this paper, we present the following results. We begin, in Section 2, by reviewing research related to our problem. In Section 3 we define cuts and establish a lower bound on the optimal load in any problem instance. In Section 4, we show that the competitive ratio of any deterministic online algorithm is at least $2 - 2/n$, where n is the size of the ring. In Section 5, we show that the competitive ratio of the online algorithm that splits each flow inversely proportionally to the length of the flow's shortest path, which we will call Proportional Split (PS), is $2 - 2/n$, for all $n \geq 2$, and is hence optimal. This result applies to both temporary and permanent flows. We find it remarkable that no better algorithm exists considering that PS is oblivious, in the sense that it does not take into account the current load when making routing decisions. This characteristic also obviates the need for any centralized network controller. In contrast, we show that the natural greedy algorithm Balance, which balances the resulting maximum load on each path, has competitive ratio $\Omega(n)$.

2 Related research

Research in the area of bidirectional SONET Rings has focused on the Ring Loading Problem (RLP) [2, 3, 4, 5, 6, 7, 8, 9, 11, 12], introduced by Cosares and Saniee [2]. Here, the concern is to set the initial capacity of the ring based on forecasted traffic demands d_{st} from each source s to each destination t. In the original RLP, the entire demand between two given nodes must be routed in one of the two possible directions, and the objective is to minimize the maximum load on the cycle, where the load between two adjacent nodes is the sum of the demands routed over the edges connecting them in *both directions*. Note that this differs from our problem since we count the load in each direction separately. Also, we handle each flow request separately rather than combining all flows between s and t into a single demand. The more general Relaxed RLP [8, 9, 12] allows demands to be split arbitrarily in two directions. The Relaxed RLP with Integer Demand Splitting [6, 7, 12] also allows demands to be split, but the amount of demand assigned to each direction must be an integer.

Cosares and Saniee [2] proved that the RLP is NP-hard. However, the problem with unit demands is solvable in polynomial time [4]. Dell'Amico et al. [3] were the first to give an exact algorithm to solve the RLP. Schrijver et al. [11] used a rounding algorithm on solutions to the Relaxed RLP to provide an approximation algorithm that achieves a load that is at most 3/2 times the maximum demand. Khanna [5] improved on this result by showing that a polynomial time approximation scheme (PTAS) exists for the RLP. Myung et al. [9] gave an algorithm to solve the Relaxed RLP that runs in $O(nk)$ time, improving upon previous algorithms given by Dell'Amico et al. [3] and Schrijver et al. [11]. Myung and Kim [8] and Wang [12] later gave improved algorithms with running times $O(\min\{nk, n^2\})$ and $O(k)$ when $k \geq n^\epsilon$ for some $\epsilon > 0$, respectively. Lee and Chang [6], Myung [7], and Wang [12] have provided increasingly efficient algorithms for the Relaxed RLP with Integer Demand Splitting. We show in Section 4 how the lower bound for our problem can be modified to also hold for the online Relaxed RLP.

The Directed Ring Routing Problem (DRRP) [13], in contrast to the RLP, allows for multiple, individually routed, non-splittable unit flow demands between each pair of nodes, and link load is measured individually on each of the $2n$ arcs (as in our problem). Wilfong and Winkler [13] showed that DRRP is a tractable case of the integer multicommodity flow problem. They gave a polynomial-time algorithm which uses the solution to the relaxed LP version along with a method to unsplit demands. Our problem is an online version of the relaxed DRRP.

3 Cuts and optimal load

An optimal (offline) solution to our problem can be found by solving the LP given by Wilfong and Winkler [13]. A natural lower bound on the optimal solution can be obtained by looking at cuts on the ring. We define a *cut* K in a bidirectional ring to be a pair $[e, E']$ with the following two properties:

1. E' is a nonempty set of edges oriented in the same direction.
2. e is a single edge that is oriented in the direction opposite that of the edges in E' and is different than (j, i) for every $(i, j) \in E'$.

Let $|K|$ denote the number of edges in the cut, i.e., if $K = [e, E']$, then $|K| = |E'| + 1$. Note that $|E'| \in \{1, 2, \ldots, n-1\}$. A cut $K = [e, E']$ induces a set of $|E'|$ unique *cutsets* $\{[e, e'] : e' \in E'\}$. A flow f_j *crosses* a cutset $[e, e']$ if it crosses edge e on one path from its source to its destination and edge e' on its other path. A flow f_j *crosses a cut* K if it crosses at least one of the cutsets induced by K. Let $\mathcal{F}_t(K)$ denote the set of indices of flows in request sequence $\mathcal{F}$ that cross cut K at time t. Also, let $\sigma(\mathcal{F}_t(K)) = \sum_{j \in \mathcal{F}_t(K)} l_j$.

Theorem 1 *In any solution for an instance $(n, \mathcal{F})$, the maximum load is at least $\max_{K,t}\{\sigma(\mathcal{F}_t(K))/|K|\}$.*

Proof Consider an arbitrary instance $(n, \mathcal{F})$, time t, and cut $K = [e, E']$, where $E' = \{e'_1, e'_2, \ldots, e'_m\}$. Now consider an arbitrary solution for this instance and let $L(e'_i)$ denote the load on edge $e'_i \in E'$ at time t from flows in $\mathcal{F}_t(K)$. Every flow in $\mathcal{F}_t(K)$ must cross edge e in one direction and at least one of the edges in E' in the opposite direction. Therefore, in order to minimize the maximum load from the flows in $\mathcal{F}_t(K)$ on the edges in K, we want $L(e'_1) = L(e'_2) = \cdots = L(e'_m) = L(e)$ subject to $\sum_{i=1}^{m} L(e'_i) + L(e) \geq \sigma(\mathcal{F}_t(K))$. Therefore, $\max_{\hat{e} \in E' \cup \{e\}}\{L(\hat{e})\} \geq \sigma(\mathcal{F}_t(K))/(m+1) = \sigma(\mathcal{F}_t(K))/|K|$. □

4 General lower bound

We will prove the following result.

Theorem 2 *The competitive ratio of any deterministic online algorithm for routing splittable flows on a ring is at least $2 - 2/n$ for any $n \geq 2$.*

Our proof uses a standard adversary argument in which an adversary issues flow requests to an arbitrary online algorithm A in order to make it perform as badly as possible. The adversary

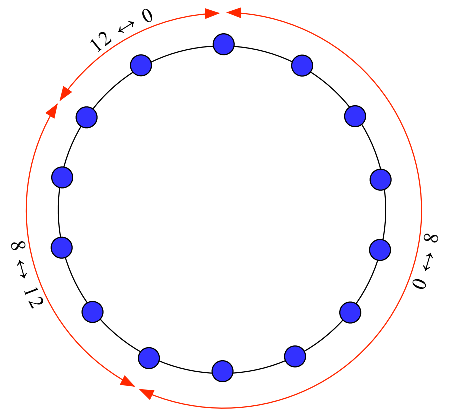

Figure 1: An illustration of the ring segments when $n = 14$.

will request pairs of opposing flows of the form $\{(s,t,f),(t,s,f)\}$, which we will denote $(s \leftrightarrow t, f)$ for simplicity. Also, suppose $s \to v_1 \to v_2 \to \cdots \to v_p \to t$ is a path for the flow (s,t,f) and $t \to v_p \to v_{p-1} \to \cdots \to v_1 \to s$ is the corresponding path for the flow (t,s,f). Then we will denote this set of two opposing paths as $s \leftrightarrow v_1 \leftrightarrow v_2 \leftrightarrow \cdots \leftrightarrow v_p \leftrightarrow t$. The adversary behaves differently depending upon whether n is even or odd. We will present the even case in detail followed by a description of how the odd case differs. Because the adversary sequence is somewhat complex, we first offer examples of a particular even case ($n = 14$) and a particular odd case ($n = 15$).

4.1 An even example ($n = 14$)

We will think of n as a sum of powers of 2; in particular, 14 is the sum of 8, 4, and 2. The adversary's request sequence will be based on a partition of the ring into segments with these lengths, as illustrated in Figure 1.

The adversary begins by sending two pairs of flows $(0 \leftrightarrow 8, 1)$.[1] Consider the fraction of these flows' total demand that A chooses to send on the two paths with length 8 $(0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 8)$.

Case 1: If A sends a total of $24/14 = 4 - 32/14$ or more of the demand from the first four flows on $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 8$, then the adversary proceeds to send more flow on this segment of the ring, as illustrated on the left side of Figure 2. The adversary begins by sending four additional flows: $(0 \leftrightarrow 4, 2)$ and $(4 \leftrightarrow 8, 2)$. A must choose to send a fraction of the flows $(0 \leftrightarrow 4, 2)$ on their shortest paths $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 4$ and the remaining fraction on their longest paths which include $4 \leftrightarrow 5 \leftrightarrow \cdots \leftrightarrow 8$. Similarly, A must choose to send a fraction of the flows in $(4 \leftrightarrow 8, 2)$ on their shortest paths $4 \leftrightarrow 5 \leftrightarrow \cdots \leftrightarrow 8$ and the remaining fraction on their longest paths which include $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 4$. Therefore, A must choose to send at least half of the total demand from the flows $(0 \leftrightarrow 4, 2)$ and $(4 \leftrightarrow 8, 2)$ on either the paths in $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 4$ or the paths in $4 \leftrightarrow 5 \leftrightarrow \cdots \leftrightarrow 8$. Without loss of generality, suppose that A chooses the latter. Then the adversary sends four more flows whose shortest paths are contained in $4 \leftrightarrow 5 \leftrightarrow \cdots \leftrightarrow 8$: $(4 \leftrightarrow 6, 4)$ and $(6 \leftrightarrow 8, 4)$. Note that these four flows are similar to the previous four, but have half the length and twice the demand. Similar to the previous step, A must choose to send at least half of the total demand from these flows on either the paths in $4 \leftrightarrow 5 \leftrightarrow 6$ or $6 \leftrightarrow 7 \leftrightarrow 8$. Without loss of generality, suppose that A chooses the former. Because these paths have length 2, the adversary sends two final flows that

[1]Sending two pairs instead of one preserves consistency between the even and odd cases, as we will show later.

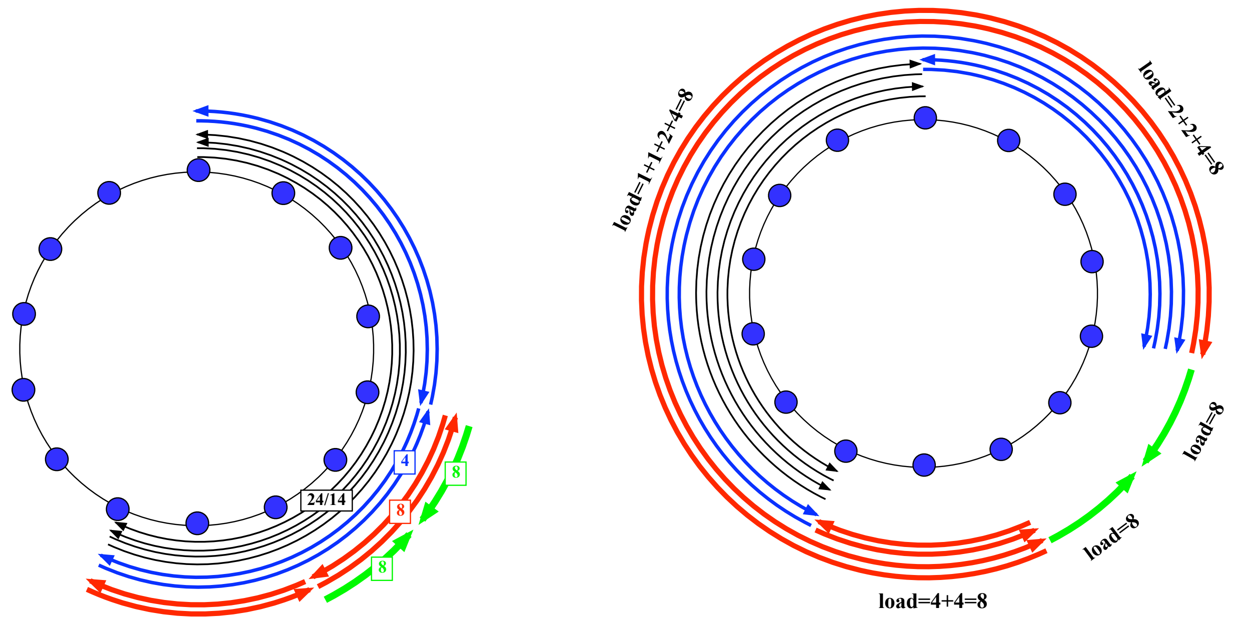

Figure 2: An illustration of a possible flow sequence in Case 1 when $n = 14$. Line widths and colors indicate the flow amounts (1 is black, 2 is blue, 4 is red, and 8 is green). On the left, one path of each flow is shown with lower bounds on the algorithm's load in rectangles. On the right, the corresponding optimal solution with loads is shown.

cross the cut $[(4,5),(6,5)]$: $(4,5,8)$ and $(6,5,8)$. Notice that the adversary has forced A to send total flow $24/14 + 4 + 8 + 16 = 416/14$ over this cut. Therefore, A can do no better than to split this demand between the two cut edges.

But an optimal solution can send every flow but the last two on a path that does not cross the edges in the cut $[(4,5),(6,5)]$. In particular, an optimal solution sends the two pairs of flows $(0 \leftrightarrow 8, 1)$ on the paths $8 \leftrightarrow 9 \leftrightarrow \cdots \leftrightarrow 0$, the flows $(0 \leftrightarrow 4, 2)$ and $(4 \leftrightarrow 8, 2)$ on paths $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 4$ and $8 \leftrightarrow 9 \leftrightarrow \cdots \leftrightarrow 4$, respectively, and the flows $(4 \leftrightarrow 6, 4)$ and $(6 \leftrightarrow 8, 4)$ on paths $6 \leftrightarrow 7 \leftrightarrow \cdots \leftrightarrow 4$ and $6 \leftrightarrow 7 \leftrightarrow \cdots \leftrightarrow 8$, respectively. This assignment results in a load of 8 on every edge but those between nodes 4 and 5 and nodes 5 and 6. See the right side of Figure 2 for an illustration. Then the optimal solution can send the last two flows $(4,5,8)$ and $(6,5,8)$ on their shortest paths without increasing the maximum load, resulting in a competitive ratio at least $(208/14)/8 = 2 - 2/14$ for algorithm A.

Case 2: If A sends less than $24/14$ of the demand from the first four flows on $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 8$, the adversary sends the following four additional flows: $(8 \leftrightarrow 12, 2)$ and $(12 \leftrightarrow 0, 2)$. Consider the total flow that A chooses to send on the paths $8 \leftrightarrow 9 \leftrightarrow \cdots \leftrightarrow 12$.

Case 2(a): If A chooses to send at least $48/14 = 8 - 64/14$ of the total demand from these four flows on $8 \leftrightarrow 9 \leftrightarrow \cdots \leftrightarrow 12$, then the adversary next issues a sequence of flows that is analogous to the sequence in Case 1, as illustrated in Figure 3. Specifically, the adversary first sends the four flows $(8 \leftrightarrow 10, 4)$ and $(10 \leftrightarrow 12, 4)$. A must choose to send at least half of the total demand from these four flows on either the paths $8 \leftrightarrow 9 \leftrightarrow 10$ or the paths $10 \leftrightarrow 11 \leftrightarrow 12$. Without loss of generality, suppose that A chooses the former. Then the adversary sends two final flows: $(8,9,8)$ and $(10,9,8)$. So the adversary has forced A to send total flow at least $32/14 + 48/14 + 8 + 16 = 416/14$ over the cut $[(8,9),(10,9)]$ and at least half this amount over one edge of the cut. As in Case 1, the optimal

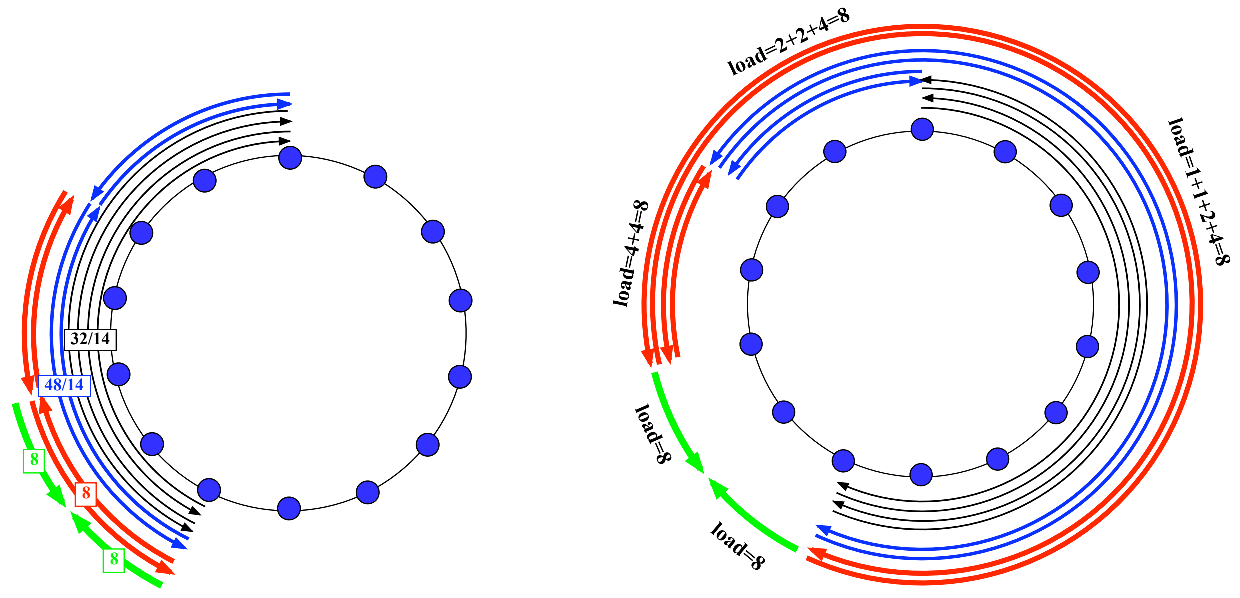

Figure 3: An illustration of a possible flow sequence in Case 2(a) when $n = 14$. Line widths and colors indicate the flow amounts (1 is black, 2 is blue, 4 is red, and 8 is green). On the left, one path of each flow is shown with lower bounds on the algorithm's load in rectangles. On the right, the corresponding optimal solution with loads is shown.

load is 8, giving competitive ratio at least $2 - 2/14$.

Case 2(b): Finally, if A chooses to send less than $48/14$ of the total demand from these four flows on $8 \leftrightarrow 9 \leftrightarrow \cdots \leftrightarrow 12$, then the adversary sends two last flows: $(12, 13, 4)$ and $(0, 13, 4)$, as illustrated in Figure 4. Now the adversary has forced A to send total flow at least $32/14 + 64/14 + 8 = 208/14$ across the cut $[(12, 13), (0, 13)]$. Since A must send at least half this amount over one of the cut edges and optimal load is 4, the competitive ratio of A is at least $2 - 2/14$.

4.2 An odd example ($n = 15$)

When n is odd, the adversary needs a slightly different request sequence based on the even $n - 1$ case. We will illustrate the idea with $n = 15$.

The adversary begins by sending two flows: $(0 \leftrightarrow 1, 1)$. If A sends at least $14/15$ of either of these flows on its shortest path, the adversary stops. Since the optimal maximum load at this point is $1/2$, the competitive ratio of A is at least $2 - 2/15$. Otherwise, the adversary proceeds as in the $n = 14$ case, except that the flow sources and destinations are increased by one[2] and the thresholds between cases differ slightly. Specifically, the adversary begins by requesting the flows $(1 \leftrightarrow 9, 1)$ and $(9 \leftrightarrow 0, 1)$. If A chooses to send at least $26/15 = 4 - 34/15$ (instead of $24/14$ for $n = 14$) of the total demand from these flows on the paths $1 \leftrightarrow 2 \leftrightarrow \cdots \leftrightarrow 9$, then the adversary next requests the flows in Case 1, beginning with $(0 \leftrightarrow 4, 2)$ and $(4 \leftrightarrow 8, 2)$, resulting in total load at least $2/15 + 26/15 + 4 + 8 + 16 = 448/15$ over a cut and hence competitive ratio at least $(224/15)/8 = 2 - 2/15$. Otherwise, the adversary issues the flows in Case 2: $(9 \leftrightarrow 13, 2)$ and $(13 \leftrightarrow 0, 2)$. If A chooses to send at least $52/15 = 8 - 68/15$ (instead of $48/14$) of the total demand on

[2]There is one exception: when the destination of a flow is node 0 in the $n = 14$ case, it remains node 0 in the $n = 15$ case.

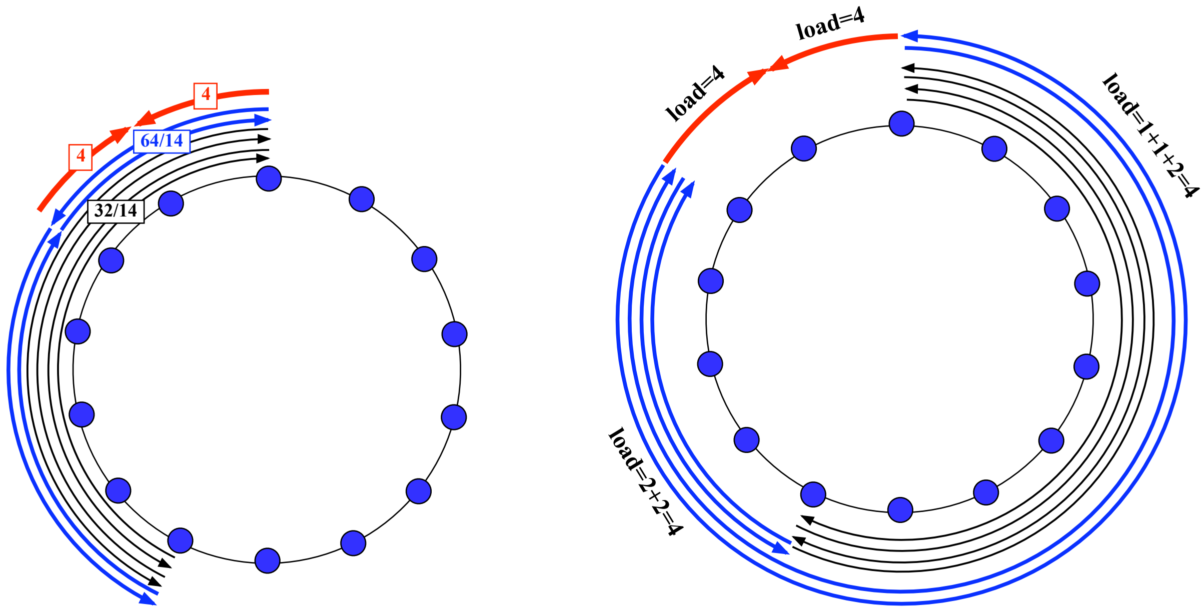

Figure 4: An illustration of the flow sequence in Case 2(b) when $n = 14$. Line widths and colors indicate the flow amounts (1 is black, 2 is blue, and 4 is red). On the left, one path of each flow is shown with lower bounds on the algorithm's load in rectangles. On the right, the corresponding optimal solution with loads is shown.

the paths $9 \leftrightarrow 10 \leftrightarrow \cdots \leftrightarrow 13$, the adversary requests the flows in Case 2(a), beginning with $(8 \leftrightarrow 10, 4)$ and $(10 \leftrightarrow 12, 4)$, resulting in total load at least $2/15 + 34/15 + 52/15 + 8 + 16 = 448/15$ over a cut, and hence competitive ratio $(224/15)/8 = 2 - 2/15$. Otherwise, the adversary requests the flows in Case 2(b) — $(12, 13, 4)$ and $(0, 13, 4)$ — forcing total load at least $2/15+34/15+68/15+8 = 224/15$ over a cut, and hence competitive ratio at least $(112/15)/4 = 2 - 2/15$.

4.3 Even n

We now show how to apply this idea to a general instance $(n, \mathcal{F})$, where n is even.

First, let $\text{Binary}(n) = b_m b_{m-1} \cdots b_0$, where each $b_i \in \{0, 1\}$, denote the binary representation of n. Then let $B_n = \{j : j \in \{0, 1, \ldots, m\} \text{ and } b_j = 1\} = \{\beta_0, \beta_1, \ldots, \beta_l\}$ where $\beta_i < \beta_j$ iff $i < j$. So $n = \sum_{j=0}^{l} 2^{\beta_j}$.

We now define in Figure 5 a request sequence $\sigma(n, i, \alpha)$, for $i \in \{0, 1, 2, \ldots, l\}$ and $0 < \alpha \le 1$, that the adversary will call upon in its full request sequence to attain the lower bound.[3] To illustrate, the sequence $\sigma(n, i, 1)$ begins with the four flows

$$\left(s_1 \leftrightarrow s_1 + 2^{\beta_{l-i}-1},\ 2^{i+1}\right) \text{ and } \left(s_1 + 2^{\beta_{l-i}-1} \leftrightarrow s_1 + 2^{\beta_{l-i}},\ 2^{i+1}\right)$$

where $s_1 = \sum_{j=l-i+1}^{l} 2^{\beta_j}$. Since these flows have combined demand 2^{i+3}, an arbitrary online

[3]Hereafter we adopt the convention that any summation over an empty range evaluates to 0. In particular, when $i = 0$, $\sum_{j=l-i+1}^{l} 2^{\beta_j} = \sum_{j=l+1}^{l} 2^{\beta_j} = 0$.

```
$\sigma(n,\ i,\ \alpha)$
    $s_1 \leftarrow \sum_{j=l-i+1}^{l} 2^{\beta_j}$
    $p \leftarrow 1$
    while $p \le \beta_{l-i} - 1$ do
        request flows in $F_p = \left(s_p \leftrightarrow s_p + 2^{\beta_{l-i}-p},\ \alpha \cdot 2^{i+p}\right) \cup \left(s_p + 2^{\beta_{l-i}-p} \leftrightarrow s_p + 2^{\beta_{l-i}-p+1},\ \alpha \cdot 2^{i+p}\right)$
        if $A$ sends at least half of the total demand of $F_p$ on $s_p \leftrightarrow s_p + 1 \leftrightarrow \cdots \leftrightarrow s_p + 2^{\beta_{l-i}-p}$ then
            $s_{p+1} \leftarrow s_p$
        else
            $s_{p+1} \leftarrow s_p + 2^{\beta_{l-i}-p}$
        $p \leftarrow p + 1$
    request flows $\left(s_p, s_p + 1,\ \alpha \cdot 2^{i+\beta_{l-i}}\right)$ and $\left((s_p + 2) \bmod n, s_p + 1,\ \alpha \cdot 2^{i+\beta_{l-i}}\right)$
```

Figure 5: Adversary request sequence $\sigma(n, i, \alpha)$.

algorithm A must choose to send at least 2^{i+2} of this demand on either the paths

$$s_1 \leftrightarrow s_1 + 1 \leftrightarrow \cdots \leftrightarrow s_1 + 2^{\beta_{l-i}-1}$$

or the paths

$$s_1 + 2^{\beta_{l-i}-1} \leftrightarrow s_1 + 2^{\beta_{l-i}-1} + 1 \leftrightarrow \cdots \leftrightarrow s_1 + 2^{\beta_{l-i}} \ .$$

On the set of paths on which A chooses to assign the most load, the adversary subsequently sends a similar sequence of four flows, but with half the length and twice the demand. For example, suppose that A chooses the former set of opposing paths (so $s_2 \leftarrow s_1$). Then the adversary sends four more flows:

$$\left(s_2 \leftrightarrow s_2 + 2^{\beta_{l-i}-2},\ 2^{i+2}\right) \text{ and } \left(s_2 + 2^{\beta_{l-i}-2} \leftrightarrow s_2 + 2^{\beta_{l-i}-1},\ 2^{i+2}\right).$$

Now A must send at least 2^{i+3} of the total demand from these flows on either

$$s_2 \leftrightarrow s_2 + 1 \leftrightarrow \cdots \leftrightarrow s_2 + 2^{\beta_{l-i}-2}$$

or

$$s_2 + 2^{\beta_{l-i}-2} \leftrightarrow s_2 + 2^{\beta_{l-i}-2} + 1 \leftrightarrow \cdots \leftrightarrow s_2 + 2^{\beta_{l-i}-1} \ .$$

The adversary continues this process, choosing the paths on which A chooses to send the most flow, halving the length and doubling the demand, until it sends flows with shortest path length 2 and demand $2^{i+\beta_{l-i}-1}$. Let $p = \beta_{l-i}$. Then $s_p \leftrightarrow s_p + 1 \leftrightarrow (s_p + 2) \bmod n$ is the set of paths on which A chose to send the most (at least $2^{\beta_{l-i}+i}$) total demand from the last set of four flows. Now the adversary sends two last flows — $(s_p, s_p+1, 2^{\beta_{l-i}+i})$ and $((s_p+2) \bmod n, s_p+1, 2^{\beta_{l-i}+i})$ — to cross the cut $K = [(s_p, s_p+1), ((s_p+2) \bmod n, s_p+1)]$.

Lemma 1 *Let $n \ge 2$ be an even integer. Then, for all $i \in \{0, 1, \ldots, l\}$, any online algorithm A assigns to cut K at least $2^{i+2}(2^{\beta_{l-i}} - 1)\alpha$ of the demand from the flow requests in sequence $\sigma(n, i, \alpha)$.*

```
Adversary-even($n$)
1   $i \leftarrow 0$
2   while ($i < l$) do
3       request flows in $P_i = \left(\sum_{j=l-i+1}^{l} 2^{\beta_j} \leftrightarrow \sum_{j=l-i}^{l} 2^{\beta_j},\, 2^i\right)$ and $Q_i = \left(\sum_{j=l-i}^{l} 2^{\beta_j} \leftrightarrow 0,\, 2^i\right)$
4       if $A$ sends at least $2^{i+2} - f(i)/n$ of the demand from $P_i \cup Q_i$ on $R_{l-i}$ then
5           request $\sigma(n,\, i,\, 1)$ and halt
6       $i \leftarrow i + 1$
7   request $\sigma(n,\, l,\, 1/2)$
```

Figure 6: The adversary sequence for even n.

Proof During each iteration $p \in \{1, 2, \ldots, \beta_{l-i} - 1\}$, A sends at least $\alpha \cdot 2^{i+p+1}$ of the demand from F_p across the cut K. Therefore, the total load on cut K from these flows is at least

$$\sum_{p=1}^{\beta_{l-i}-1} \alpha \cdot 2^{i+p+1} = 2^{i+2}\left(2^{\beta_{l-i}-1} - 1\right)\alpha.$$

Since the last two flows must cross the cut, they add an additional load of $\alpha \cdot 2^{i+\beta_{l-i}+1}$, resulting in total load at least $2^{i+2}\left(2^{\beta_{l-i}} - 1\right)\alpha$ crossing the cut. □

Now we are ready to define the full adversary request sequence, presented in Figure 6. The function f is defined as follows:

$$f(i) = \begin{cases} 2^{\beta_l+2} & \text{if } i = 0 \\ \sum_{j=0}^{i-1} f(j) + 2^{\beta_{l-i}+i+2} & \text{if } 1 \le i \le l-1 \end{cases} = \sum_{j=0}^{i-1} 2^{\beta_{l-j}+i+1} + 2^{\beta_{l-i}+i+2}\ .$$

This function helps to define the threshold between continuing in the while loop and calling upon $\sigma(n, i, 1)$ to conclude the sequence. We also let R_h, where $h \in \{0, 1, \ldots, l\}$, denote the bidirectional ring interval

$$\sum_{j=h+1}^{l} 2^{\beta_j} \leftrightarrow \sum_{j=h+1}^{l} 2^{\beta_j} + 1 \leftrightarrow \cdots \leftrightarrow \left(\sum_{j=h}^{l} 2^{\beta_j}\right) \bmod n\ .$$

Let $\overline{R_h}$ denote the bidirectional ring with interval R_h omitted. For example, in Figure 1, R_2 is the interval $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow 8$, R_1 is the interval $8 \leftrightarrow 9 \leftrightarrow \cdots \leftrightarrow 12$, and R_0 is the interval $12 \leftrightarrow 13 \leftrightarrow 0$.

Before we analyze the load that the full adversary sequence forces on A, we will examine the load in an optimal solution. We begin by quantifying the optimal load in an instance corresponding to the flows in Adversary-even(n) after P_i and Q_i are requested, but before $\sigma(n, i, 1)$ is called.

Lemma 2 *Let $n \ge 2$ be an even integer. Then, for all $i \in \{0, 1, \ldots, l-1\}$, after requesting the flows in P_i and Q_i in Adversary-even(n), there exists a solution in which the load on every edge in interval R_{l-i} is 0 and the load on every edge in intervals R_j, for $j \ne l - i$, is 2^{i+1}.*

Proof Notice that the flow requests in P_i can be assigned to either R_{l-i} or $\overline{R_{l-i}}$. The flow requests in Q_i can be assigned to either $\bigcup_{j=0}^{l-i-1} R_j$ or $\bigcup_{j=l-i}^{l} R_j$. Consider the solution in which, for $h \in \{0, 1, \ldots, i-1\}$, the flows in P_h are assigned to $\overline{R_{l-h}}$ and the flows in Q_h are assigned to $\bigcup_{j=l-h}^{l} R_j$. Also, assign the flows in P_i to $\overline{R_{l-i}}$ and the flows in Q_i to $\bigcup_{j=0}^{l-i-1} R_j$. In this solution, each interval R_h, for $h \in \{0, 1, \ldots, l-i-1\}$, is assigned only the demand from the flow requests in P_i and Q_i, resulting in load $2 \cdot 2^i = 2^{i+1}$ on each edge in the interval. Each interval R_h, for $h \in \{l-i+1, l-i+2, \ldots, l\}$, is assigned the demand from the flow requests in P_{l-h} and $\bigcup_{j=l-h}^{i} Q_j$, resulting in load $2^{l-h} + \sum_{j=l-h}^{i} 2^j = 2^{i+1}$ on each edge in the interval. The load on the edges in interval R_{l-i} is 0. □

We now incorporate the flow requests in $\sigma(n, i, \alpha)$ into the optimal load. Notice that the shortest paths of the flows in $\sigma(n, i, \alpha)$ are all contained in the interval R_{l-i} which has length $2^{\beta_{l-i}}$. We partition R_{l-i} into smaller intervals B_p, for $p \in \{1, 2, \ldots, \beta_{l-i}\}$, based on the actions of A as follows:

$$B_p = \begin{cases} s_p \leftrightarrow s_p + 1 \leftrightarrow \cdots \leftrightarrow s_p + 2^{\beta_{l-i}-p} & \text{if } 1 \leq p < \beta_{l-i} \text{ and } s_{p+1} \neq s_p \\ s_p + 2^{\beta_{l-i}-p} \leftrightarrow s_p + 2^{\beta_{l-i}-p} + 1 \leftrightarrow \cdots \leftrightarrow s_p + 2^{\beta_{l-i}-p+1} & \text{if } 1 \leq p < \beta_{l-i} \text{ and } s_{p+1} = s_p \\ s_p \leftrightarrow s_p + 1 \leftrightarrow (s_p + 2) \bmod n & \text{if } p = \beta_{l-i} \end{cases}$$

Intuitively, B_p is the interval containing the paths on which A chose to send the least demand from the flows in F_p. Notice that $\bigcup_{h=1}^{\beta_{l-i}} B_h = R_{l-i}$.

Lemma 3 *Let $n \geq 2$ be an even integer. If Adversary-even(n) terminates during iteration $i < l$ of the while loop, then there exists a solution with maximum load $2^{i+\beta_{l-i}}$.*

Proof Before terminating in iteration i, the adversary sequence calls $\sigma(n, i, 1)$ from line 5. We first prove by induction that, before iteration p of $\sigma(n, i, 1)$, there exists a solution in which the intervals in $\bigcup_{h=1}^{p-1} B_h \cup \overline{R_{l-i}}$ have load 2^{i+p} and the intervals in $\bigcup_{h=p}^{\beta_{l-i}} B_h$ have load 0. Before iteration $p = 1$, this statement holds by Lemma 2. Assume inductively that the statement holds before iteration $p-1$. Then, during iteration $p-1$, assign the flows in F_{p-1} to the paths that intersect interval B_{p-1}. This increases the load on the edges in interval B_{p-1} from 0 to $2 \cdot 2^{i+p-1} = 2^{i+p}$ and increase the load on the edges in intervals $\bigcup_{h=1}^{p-2} B_h \cup \overline{R_{l-i}}$ from 2^{i+p-1} to 2^{i+p}.

Therefore, after the adversary has requested all the flows in $\sigma(n, i, 1)$ (i.e., when $p = \beta_{l-i}$), there exists a solution in which the edges in interval $B_{\beta_{l-i}}$ have load 0 while all other edges have load $2^{i+\beta_{l-i}}$. Assigning the final flow requests to their shortest paths gives us a solution with maximum load $2^{i+\beta_{l-i}}$. □

Lemma 4 *Let $n \geq 2$ be an even integer. If Adversary-even(n) terminates on line 7, then there exists a solution with maximum load $2^{l+\beta_0-1}$.*

Proof On line 7, the adversary sequence calls $\sigma(n,\ l,\ 1/2)$. We first prove by induction that, before iteration p of $\sigma(n,\ l,\ 1/2)$, there exists a solution in which all intervals in $\bigcup_{h=1}^{p-1} B_h \cup \overline{R_0}$ have load 2^{l+p-1} and the intervals in $\bigcup_{h=p}^{\beta_0} B_h$ have load 0. When $p = 1$ and $l > 0$, the statement holds by Lemma 2 since the last flows requested were those in P_{l-1} and Q_{l-1}. When $l = 0$, the while loop in Adversary-even(n) is skipped, so the statement holds for $p = 1$ because $\bigcup_{h=p}^{\beta_0} B_h = R_0$, which is the entire ring, and $\bigcup_{h=1}^{p-1} B_h \cup \overline{R_0} = \emptyset$. Assume inductively that the statement holds before iteration $p-1$. Then, during iteration $p-1$, assign the flows in F_{p-1} to the paths that intersect interval

B_{p-1}. This increases the load on the edges in interval B_{p-1} from 0 to $2 \cdot 2^{l+p-2} = 2^{l+p-1}$ and on the edges in intervals $\bigcup_{h=1}^{p-2} B_h \cup \overline{R_0}$ from 2^{l+p-2} to 2^{l+p-1}.

Therefore, after the adversary has requested all the flows in $\sigma(n,\, l,\, 1/2)$, there exists a solution in which interval B_{β_0} has load 0 and all other intervals have load $2^{l+\beta_0-1}$. Assigning the final flow requests in $\sigma(n,\, l,\, 1/2)$ to their shortest paths gives us a solution with maximum load $2^{l+\beta_0-1}$. □

Finally, we use these results to prove the lower bound for even n.

Lemma 5 *The competitive ratio of any deterministic online algorithm for routing splittable flows on a ring is at least $2 - 2/n$ for even $n \geq 2$.*

Proof Let A by an arbitrary deterministic online algorithm. We consider two cases. First, if the adversary stops at line 5, then the load crossing the cut K in A's solution is at least

$$\frac{1}{n}\sum_{j=0}^{i-1} f(j) + \left(2^{i+2} - \frac{f(i)}{n}\right) + 2^{i+2}\left(2^{\beta_{l-i}} - 1\right) = 2^{\beta_{l-i}+i+2} - \frac{2^{\beta_{l-i}+i+2}}{n} . \tag{1}$$

The first term in (1) is the minimum load that A must have sent across the cut during iterations 0 to $i-1$ of the while loop in the adversary sequence, the second term is the minimum load that A must have sent across the cut during iteration i of the adversary sequence, and the third term is the minimum load crossing the cut due to $\sigma(n,i,1)$ by Lemma 1. Since the maximum load in A's solution must be at least half the amount in (1) and the optimal load is at most $2^{\beta_{l-i}+i}$ by Lemma 3, the competitive ratio of A is at least $2 - 2/n$.

If the adversary stops at line 7, then the load crossing the cut K in A's solution is at least

$$\begin{aligned}\frac{1}{n}\sum_{j=0}^{l-1} f(j) + 2^{l+1}\left(2^{\beta_0} - 1\right) &= 2^{l+1}\left(\frac{1}{n}\sum_{j=0}^{l-1} 2^{\beta_{l-j}} + 2^{\beta_0} - 1\right)\\ &= 2^{l+1}\left(\frac{1}{n}\left(\sum_{j=0}^{l-1} 2^{\beta_{l-j}} - \sum_{j=0}^{l} 2^{\beta_j}\right) + 2^{\beta_0}\right) = 2^{l+1}\left(2^{\beta_0} - \frac{2^{\beta_0}}{n}\right) .\end{aligned} \tag{2}$$

The first term in (2) is the minimum load that A must have sent across the cut during the l iterations of the while loop in the adversary sequence and the second term is the minimum load crossing the cut due to $\sigma(n,\, l,\, 1/2)$ by Lemma 1. Since the maximum load in A's solution must be at least half the amount in (2) and the optimal load is at most 2^{β_0+l-1} by Lemma 4, the competitive ratio of A is at least $2 - 2/n$. □

4.4 Odd n

For an instance $(n, \mathcal{F})$, where n is odd, the argument is very similar to the even case. We will just focus on the key differences here.

First, we define $B_n = B_{n-1} = \{\beta_0, \beta_1, \ldots, \beta_l\}$. So $n = \sum_{j=0}^{l} 2^{\beta_j} + 1$. Second, we modify $\sigma(n,i,\alpha)$ by adding 1 to each source and destination (except that we leave the destination the same when it is 0). We will call this modified sequence $\sigma'(n,i,\alpha)$. Third, we define f slightly differently:

$$f(i) = \begin{cases} 2^{\beta_l+2} + 2 & \text{if } i = 0 \\ \sum_{j=0}^{i-1} f(j) + 2^{\beta_{l-i}+i+2} + 2 & \text{if } 1 \leq i \leq l-1 \end{cases} = \sum_{j=0}^{i-1} 2^{\beta_{l-j}+i+1} + 2^{\beta_{l-i}+i+2} + 2^{i+1} .$$

```
Adversary-odd($n$)
1   request $(0 \leftrightarrow 1, 1)$
2   if $A$ sends at least $(n-1)/n$ of either flow on line 1 on its shortest path then
3       halt
4   $i \leftarrow 0$
5   while $(i < l)$ do
6       request flows in $P_i = \left(1 + \sum_{j=l-i+1}^{l} 2^{\beta_j} \leftrightarrow 1 + \sum_{j=l-i}^{l} 2^{\beta_j},\, 2^i\right)$ and $Q_i = \left(1 + \sum_{j=l-i}^{l} 2^{\beta_j} \leftrightarrow 0,\, 2^i\right)$
7       if $A$ sends at least $2^{i+2} - f(i)/n$ of the demand from $P_i \cup Q_i$ on $R_{l-i}$ then
8           request $\sigma'(n-1,\, i,\, 1)$ and halt
9       $i \leftarrow i + 1$
10  request $\sigma'(n-1,\, l,\, 1/2)$
```

Figure 7: The adversary sequence for odd n.

Finally, we shift by 1 each interval R_h so that interval R_l starts at 1. With these modifications in mind, the adversary sequence for odd n is presented in Figure 7. Lemmas 1, 2, 3, and 4 can be easily modified to apply to the odd case:

Lemma 6 *Let $n \geq 3$ be an odd integer. Then, for all $i \in \{0, 1, \ldots, l\}$, algorithm A assigns to cut K at least $2^{i+2}(2^{\beta_{l-i}} - 1)\alpha$ of the demand from the flow requests in sequence $\sigma'(n-1, i, \alpha)$.*

Lemma 7 *Let $n \geq 3$ be an odd integer. Then, for all $i \in \{0, 1, \ldots, l-1\}$, after requesting the flows in P_i and Q_i in Adversary-odd(n), there exists a solution in which the load on every edge in interval R_{l-i} is 0 and the load on every edge in intervals R_j, for $j \neq l-i$, is 2^{i+1}.*

Lemma 8 *Let $n \geq 3$ be an odd integer. If Adversary-odd(n) terminates during iteration $i < l$ of the while loop, then there exists a solution with maximum load $2^{i+\beta_{l-i}}$.*

Lemma 9 *Let $n \geq 3$ be an odd integer. If Adversary-odd(n) terminates on line 10, then there exists a solution with maximum load $2^{l+\beta_0-1}$.*

The arguments regarding optimal load in the last three lemmas are identical to the even case, except that the initial flow requests $(0 \leftrightarrow 1, 1)$ are assigned to their shortest paths in an optimal solution.

Lemma 10 *The competitive ratio of any deterministic online algorithm for routing splittable flows on a ring is at least $2 - 2/n$ for odd $n \geq 3$.*

Proof Let A be an arbitrary online algorithm. We consider three cases. First, if the adversary stops at line 3, then A has maximum load at least $(n-1)/n$ while the optimal load is $1/2$. Therefore, the competitive ratio of A is at least $2 - 2/n$.

If the adversary stops at line 8, then the load crossing the cut K in A's solution is at least

$$\frac{2}{n} + \frac{1}{n}\sum_{j=0}^{i-1} f(j) + \left(2^{i+2} - \frac{f(i)}{n}\right) + 2^{i+2}\left(2^{\beta_{l-i}} - 1\right) = 2^{\beta_{l-i}+i+2} - \frac{2^{\beta_{l-i}+i+2}}{n}\ . \tag{3}$$

All terms on the left hand side of (3) except the first are identical to the terms in (1). Specifically, the first term in (3) is a lower bound on the load from $(0 \leftrightarrow 1, 1)$ that A must have sent across K, the second term is the minimum load that A must have sent across the cut during iterations 0 to $i-1$ of the while loop in the adversary sequence, the third term is the minimum load that A must have sent across the cut during iteration i of the adversary sequence, and the fourth term is the minimum load crossing the cut due to $\sigma'(n-1, i, 1)$ by Lemma 6. Since the maximum load in A's solution must be at least half the amount in (3) and the optimal load is at most $2^{\beta_{l-i}+i}$ by Lemma 8, the competitive ratio of A is at least $2 - 2/n$.

If the adversary stops at line 10, then the load crossing the cut K in A's solution is at least

$$\begin{aligned}\frac{2}{n} + \frac{1}{n}\sum_{j=0}^{l-1} f(j) + 2^{l+1}\left(2^{\beta_0} - 1\right) &= 2^{l+1}\left(\frac{1}{n}\left(\sum_{j=0}^{l-1} 2^{\beta_{l-j}} + 1\right) + 2^{\beta_0} - 1\right)\\ &= 2^{l+1}\left(\frac{1}{n}\left(\sum_{j=0}^{l-1} 2^{\beta_{l-j}} - \sum_{j=0}^{l} 2^{\beta_j}\right) + 2^{\beta_0}\right)\\ &= 2^{l+1}\left(2^{\beta_0} - \frac{2^{\beta_0}}{n}\right). \end{aligned} \tag{4}$$

All terms on the left hand side of (4) except the first are identical to the terms in (2). The first term in (4) is a lower bound on the load from $(0 \leftrightarrow 1, 1)$ that A must have sent across K, the second term is the minimum load that A must have sent across the cut during the l iterations of the while loop in the adversary sequence, and the third term is the minimum load crossing the cut due to $\sigma'(n-1,\ l,\ 1/2)$ by Lemma 6. Since the maximum load in A's solution must be at least half the amount in (4) and the optimal load is at most 2^{β_0+l-1} by Lemma 9, the competitive ratio of A is at least $2 - 2/n$. □

Together, Lemmas 5 and 10 prove Theorem 2.

4.5 Relaxed RLP

Recall that, in the (relaxed) RLP, the load on an edge (u, v) is defined to be the sum of the loads assigned to both (u, v) and (v, u). We note that the result in Theorem 2 can be adapted to apply to this definition of load as well.

Corollary 1 *The competitive ratio of any deterministic online algorithm for the Relaxed RLP is at least* $2 - 2/n$.

Proof In the proofs of Lemmas 5 and 10, we are concerned with the total load on pairs of opposing paths, but this is identical to the definition of load on a single path in the RLP. Therefore, the adversary can substitute flow (s, t, l) for each pair of opposing flows $(s \leftrightarrow t, l)$ in the proofs to force an online algorithm A for the Relaxed RLP to get the same loads. The individual flows (s, t, l) in the adversary sequences for our problem remain the same. □

5 The proportional split algorithm

It is well known that the algorithm that assigns each flow to its shortest path has competitive ratio 2 on the ring [10]. Therefore, the shortest path algorithm is optimal for $n \to \infty$. We next show that the PROPORTIONAL SPLIT (PS) algorithm, which splits each flow inversely proportionally to the length of the flow's shortest path, has competitive ratio $2-2/n$ and is thus an optimal deterministic algorithm for all n. The algorithm is defined as follows:

Algorithm PROPORTIONAL SPLIT (PS): If the length of the shortest path for a flow $f_j = (s_j, t_j, l_j)$ is d, then send $(1 - d/n)l_j$ units of the flow on the flow's shortest path and the remaining $(d/n)l_j$ units of the flow on its longest path.

It is remarkable that PS is an optimal online algorithm given that it does not use past information when making current routing decisions. Since PS makes decisions independently of the current load, routing decisions can be made without a centralized controller. Also, this allows us, in the proof that follows, to ignore flow arrivals and departures. Thus, we assume permanent flows for simplicity, but we could also consider a set of temporary flows at any particular point in time.

Theorem 3 *The competitive ratio of PS is* $2 - 2/n$.

Proof For an arbitrary instance $(n, \mathcal{F})$, let $\tilde{e}$ denote the edge with maximum load in the solution constructed by PS. Let L denote the load on edge $\tilde{e}$ in the solution constructed by PS. Also, let L_S and L_L denote the load on edge $\tilde{e}$ that is contributed by flows that are assigned to their shortest and longest paths, respectively, in the solution constructed by PS. Let $\mathcal{S}$ denote the set of indices of flows whose shortest paths contain edge $\tilde{e}$. Also, let $\mathcal{S}_d$ denote the set of indices of flows whose shortest paths contain edge $\tilde{e}$ and have length d. Let $\mathcal{S}(e)$ denote the set of indices of flows whose shortest paths contain both edges $\tilde{e}$ and e. Similarly, let $\mathcal{L}_d$ denote the set of indices of flows whose longest paths both contain edge $\tilde{e}$ and have length $n-d$ and let $\mathcal{L}(e)$ denote the set of indices of flows whose longest paths contain edge $\tilde{e}$ but *not* edge e (shortest paths contain edge e). For any set of flow indices $\mathcal{A}$, let $\sigma(\mathcal{A}) = \sum_{j\in\mathcal{A}} l_j$.

First, by the definition of PS, we know that

$$L_S = \sum_{d=1}^{\lfloor n/2\rfloor} \left(1 - \frac{d}{n}\right) \sigma(\mathcal{S}_d) = \frac{1}{n} \sum_{d=1}^{\lfloor n/2\rfloor} (n-d)\, \sigma(\mathcal{S}_d)\ . \tag{5}$$

and

$$L_L = \sum_{d=1}^{\lfloor n/2\rfloor} \left(\frac{d}{n}\right) \sigma(\mathcal{L}_d) = \frac{1}{n} \sum_{d=1}^{\lfloor n/2\rfloor} d\, \sigma(\mathcal{L}_d)\ . \tag{6}$$

By Theorem 1, we know that the optimal maximum load is

$$\begin{aligned} OPT &\geq \frac{1}{2}\left(\max_{K=[\tilde{e},e']} \{\sigma(\mathcal{F}(K))\}\right) \\ &= \frac{1}{2}\left(\max_{e'} \left\{\sigma\left(\mathcal{L}\left(e'\right)\right) + \sigma\left(\mathcal{S} \setminus \mathcal{S}\left(\overline{e'}\right)\right)\right\}\right) \\ &= \frac{1}{2}\left(\sigma(\mathcal{S}) + \max_{e'} \left\{\sigma\left(\mathcal{L}\left(e'\right)\right) - \sigma\left(\mathcal{S}\left(\overline{e'}\right)\right)\right\}\right) \end{aligned} \tag{7}$$

where the maximums are taken over all edges e' that can form a cutset with $\tilde{e}$ (i.e., edges $e' \neq \overline{\tilde{e}}$ with direction opposite of $\tilde{e}$). By averaging over all of these edges e', we can say that

$$\max_{e'} \left\{ \sigma\left(\mathcal{L}\left(e'\right)\right) - \sigma\left(\mathcal{S}\left(\overline{e'}\right)\right) \right\} \geq \frac{1}{n-1} \left(\sum_{e'} \left\{ \sigma\left(\mathcal{L}\left(e'\right)\right) - \sigma\left(\mathcal{S}\left(\overline{e'}\right)\right) \right\} \right). \tag{8}$$

Now, by summing over shortest path length rather than edges, we can rewrite this load as

$$\sum_{e'} \left\{ \sigma\left(\mathcal{L}\left(e'\right)\right) - \sigma\left(\mathcal{S}\left(\overline{e'}\right)\right) \right\} = \sum_{d=1}^{\lfloor n/2 \rfloor} \left(d\, \sigma(\mathcal{L}_d) - (d-1)\, \sigma(\mathcal{S}_d) \right). \tag{9}$$

Note that the second term in the summation has factor $d-1$ because, in the previous sum, $e' \neq \overline{\tilde{e}}$. Now by substituting (8) and (9) into (7), we have

$$OPT \geq \frac{1}{2(n-1)} \left(\sum_{d=1}^{\lfloor n/2 \rfloor} d\, \sigma(\mathcal{L}_d) + \sum_{d=1}^{\lfloor n/2 \rfloor} (n-d)\, \sigma(\mathcal{S}_d) \right). \tag{10}$$

Finally, by substituting (5), (6) and (10), we have $PS/OPT = (L_S + L_L)/OPT \leq 2 - 2/n$. □

As mentioned above, it is somewhat remarkable that PS is optimal considering that it ignores the current load on the network. It is interesting to contrast this with the following natural greedy algorithm that attempts to minimize the maximum load on the network by making locally optimal decisions. Balance splits each flow so that the resulting maximum cumulative loads on the flow's two paths are as close to equal as possible.

Formally, let $B_j(P)$ denote the maximum load over the edges on path P after Balance has assigned flows f_i, $i \in \{1, 2, \ldots, j\}$. Let S_j and L_j denote the shortest and longest paths for flow f_j and let α_j denote the fraction of l_j that Balance assigns to shortest path S_j.

Algorithm Balance: Choose α_j to minimize $|(B_{j-1}(S_j) + \alpha_j l_j) - (B_{j-1}(L_j) + (1-\alpha_j) l_j)|$.

Theorem 4 *The competitive ratio of* Balance *is at least* $n/2$.

Proof Suppose an adversary issues the following sequence of flows to Balance: $f_0 = (0,1,1), f_1 = (1,0,1), f_2 = (1,2,1), f_3 = (2,1,1), \ldots, f_{2n-2} = (n-1,0,1), f_{2n-1} = (0,n-1,1)$. When every flow f_i arrives, the maximum load on both of its paths will be $\lfloor i/2 \rfloor / 2$, so Balance will split every flow evenly between its two paths. After the last flow is routed, the algorithm's maximum load is therefore $n/2$. An optimal solution with load 1 can be constructed by assigning all of the flows to their mutually disjoint shortest paths. □

6 Conclusion

We have studied an online version of the Directed Ring Routing Problem introduced in [13]. In the case of splittable flows, we have shown that the Proportional Split (PS) algorithm is an optimal deterministic algorithm with competitive ratio $2 - 2/n$. This is remarkable since PS makes routing decisions independently of past decisions. In contrast, Balance, which tries to balance the resulting load along a flow's two paths, has competitive ratio $\Omega(n)$.

Acknowledgments

We wish to thank the anonymous referees for carefully reading the manuscript and making suggestions that improved the presentation of the paper.

References

[1] *IEEE Std 802.17-2004: Resilient packet ring (RPR) access method and physical layer specifications.* IEEE Computer Society, 2004.

[2] S. Cosares and I. Saniee. An optimization problem related to balancing loads on SONET rings. *Telecommunications Systems* 3 (1994), 165–181.

[3] M. Dell'Amico, M. Labbé, and F. Maffioli. Exact solution of the SONET ring loading problem. *Operations Research Letters* 25 (1999), 119–129.

[4] A. Frank, T. Nishizeki, N. Saito, H. Suzuki, and E. Tardos. Algorithms for routing around a rectangle. *Discrete Applied Mathematics* 40 (1992), 363–378.

[5] S. Khanna. A polynomial time approximation scheme for the SONET ring loading problem. Bell Labs Technical Journal 2 (1997), pp. 36-41.

[6] C. Y. Lee and S. G. Chang. Balancing loads on SONET rings with integer demand splitting. *Computers and Operations Research* 24 (1997), 221–229.

[7] Y.-S. Myung. An efficient algorithm for the ring loading problem with integer demand splitting. *SIAM Journal on Discrete Mathematics* 14 (2001), 291–298.

[8] Y.-S. Myung and H.-G. Kim. On the ring loading problem with demand splitting. *Operations Research Letters* 32 (2004), 167–173.

[9] Y.-S. Myung, H.-G. Kim, and D.-W. Tcha. Optimal load balancing on SONET bidirectional rings. *Operations Research* 45 (1997), 148–153.

[10] R. Ramaswami and G. H. Sasaki. Multiwavelength optical networks with limited wavelength conversion. In *Proceedings of IEEE INFOCOM*, 1997, pp. 489-498.

[11] A. Schrijver, P. Seymour, and P. Winkler. The ring loading problem. *SIAM Review* 41 (1999), 777–791.

[12] B.-F. Wang. Linear time algorithms for the ring loading problem with demand splitting. *Journal of Algorithms* 54 (2005), 45–57.

[13] G. Wilfong and P. Winkler. Ring routing and wavelength translation. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1999, pp. 333–341.

[14] T. Wu. *Fiber Network Service Survivability.* Artech House, Boston, 1992.