Packet Routing and Information Gathering in Lines, Rings and Trees *

Yossi Azar † Rafi Zachut ‡

Abstract

We study the problem of online packet routing and information gathering in lines, rings and trees. A network consists of n nodes. At each node there is a buffer of size B. Each buffer can transmit one packet to the next buffer at each time step. The packets injection is under adversarial control. Packets arriving at a full buffer must be discarded. In information gathering all packets have the same destination. If a packet reaches the destination it is absorbed. The goal is to maximize the number of absorbed packets. Previous studies have shown that even on the line topology this problem is difficult to handle by online algorithms. A lower bound of $\Omega(\sqrt{n})$ on the competitiveness of the Greedy algorithm was presented by Aiello et al in [2]. All other known algorithms have a polynomial competitive ratio. In this paper we give the first $O(\log n)$ competitive deterministic algorithm for the information gathering problem in lines, rings and trees. We also consider multi-destination routing where the destination of a packet may be any node. For lines and rings we show an $O(\log^2 n)$ competitive randomized algorithms. Both for information gathering and for the multi-destination routing our results improve exponentially the previous results.

1 Introduction

Overview: Packet routing networks, have become dominant platform for carrying data. In this paper we investigate a packet routing and information gathering in lines, rings and trees. In information gathering all injected packets have the same destination. Information gathering is widely used in many networks (e.g sensor networks). We also consider the multi destination routing in which the destination of a packet might be any node in the network.

We model the problem of packet routing on a unidirectional line or ring or tree as follows. A network has n nodes. At each node a buffer of size B. At each time step, new packets may be injected to the buffers, each has a destination node. Then a buffer can transmit one packet to its successor. A packet can only be stored in a buffer if there is enough space. Since the n nodes have bounded capacity, packet loss may occur. When a packet reaches its destination it is absorbed. All packets have the same value. The goal is to maximize the number of absorbed packet. The definitions can be extended to rings, trees and general graphs.

Traditionally, the performance of a packet routing algorithm was measured within the stability analysis. In such framework either a probabilistic model ([19, 31]) or an adversarial model([4, 22]) for

*A preliminary version of this paper appears in the proceedings of the 13th Annual European Symposium on Algorithms (ESA), 2005, pp. 484-495.

†azar@tau.ac.il. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel. Research supported in part by the German-Israeli Foundation and by the Israel Science Foundation.

‡Zachutra@post.tau.ac.il. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel.

the packet injection is given, and the goal is to bound the buffer size needed to prevent packet drop. Since it seems impossible to avoid packet drop in practice, approximation analysis, which avoids any a priori assumption on the input and compares the performance of algorithms to the optimal solution in the context of throughput has been adopted recently. In particular competitive analysis in which one has to deal with dropped packets becomes a common approach ([2, 17]).

To the best of our knowledge, even for a simple topology as the line, all known algorithm either have a polynomial competitive ratio or they must use buffers which are much larger than those of the optimal solution, in order to obtain a good competitive ratio. In [2] Aiello *et al.* showed the poor performance of the greedy algorithm for information gathering by proving a lower bound of $\Omega(\sqrt{n})$ on its competitive ratio (actually the upper bound is $n+1$ for FIFO model (see [17]) and n for non-FIFO model (see [25])). They also showed that for the multi-destination routing, algorithm NTG (Nearest To Go) is $O(n^{\frac{2}{3}})$-competitive. There were no algorithms with poly-logarithmic competitive ratio given the same condition as the optimal solution even for lines, rings and trees.

In this paper we provide the first logarithmic competitive algorithm for information gathering improving exponentially the previous results for lines, rings and trees. For multi destination routing we provide the first poly-logarithmic randomized algorithms for lines and rings. Our results hold for small buffers of constant size as well as for large buffers. The competitiveness of our algorithms is independent of the buffer size.

Our results:

- Our main contribution is an $O(\log n)$ competitive deterministic algorithms for information gathering in lines, rings and trees. For lines and rings we require $B \geq 2$ and for trees we require that B is larger than the maximum degree. We note that for $B = 1$ there is an easy deterministic lower bound of n for the line.[1]

- We provide an $O(\log^2 n)$ competitive randomized algorithm for the multi-destination routing in lines and unidirectional and bidirectional rings topology for any $B \geq 2$.

We use two tools which are of their own interest:

- We present a generic technique to transform any *fractional* algorithm for information gathering into a discrete algorithm. Specifically, we show that given a fractional algorithm for information gathering with buffers of size $B \geq 2$ in a line, we can construct a discrete algorithm whose competitive ratio is larger by the small factor of $\frac{B}{B-1}$.

- We present a generic technique to construct a fractional algorithm for large buffers from an algorithm for smaller buffers. Specifically, we show that given a fractional algorithm for information gathering for buffers of size n we can construct a fractional algorithm for buffers of size $B > n$ with competitive ratio larger by a factor of 16.

Recently, independently of our work Angelov *et al.* [5] achieved a slightly weaker result of $O(\log^2 n)$ competitiveness for information gathering in lines and trees and a randomized $O(\log^3 n)$ algorithm for multi-destination routing in lines.

[1]**A lower bound of $\Omega(n)$ for $B = 1$ (see [2]).** We describe a repeated process in which *Online* delivers at most one packet while the adversary delivers at least n. All packets are destined to node $n-1$. The adversary injects a packet to node 0. *Online* must accept this packet as it might be the only packet. Whenever *Online* forwards this packet, the adversary injects packets to the new location until *Online* forwards it on. *Online* must reject the new packets. If *Online* doesn't forward before n packets were injected, the process can be easily completed. Otherwise by the time *Online* delivers its packet, the adversary buffers n packets which will be delivered during the next n time steps.

Our techniques: We start by studying the online fractional call admission and circuit routing problem. By a small modification to the algorithm of Awerbuch *et al.* [7] for the discrete version of this problem with small bandwidth requests, we obtain a fractional $O(\log D)$ competitive algorithm, where D is a bound on the allowed maximum length of paths used. We next construct an *online* reduction from fractional packet routing in a line to fractional call admission and circuit routing and obtain an $O(\log(nB))$ competitive algorithm for fractional information gathering, which immediately supplies an $O(\log n)$ competitiveness for buffers of size polynomial in n. For larger buffers we use a reduction to buffers of size n in order to obtain the $O(\log n)$ competitiveness. Next we use a technique to transform any fractional algorithm for information gathering into a discrete algorithm, and obtain an $O(\log n)$ discrete algorithm for information gathering in a line network. We construct an $O(\log^2 n)$ randomized algorithm for multi-destination routing in a line network using the "Classify and randomly Select" technique with the algorithm for information gathering. We extend our techniques to rings and trees.

Related results for throughput packet routing:

- **Line and tree topologies:** Aiello *et al.*[2] investigated the unit packet routing on the line topology and proved a lower bound of $\Omega(\sqrt{n})$ on the competitiveness of the greedy algorithm for information gathering in a line. Algorithm *NTG* (Nearest To Go) is shown in [2] to be $O(n^{\frac{2}{3}})$ competitive for multi-destination routing in a line. Azar and Richter [17] showed that the greedy algorithm for multi-destination routing is $(n+1)$ competitive. In [27] Kesselman *et al.* investigated the routing problem under the *work conserving* assumption on directed lines and directed trees where packets are injected at the leaves and are destined to the root.
- **General graphs:** In [9] Awerbuch *et al.* presented a load balance algorithm for anycasting packet routing in general topologies. For the line problem this algorithm is $\frac{1}{1-\epsilon}$ competitive using buffers which are larger by factor of $O(\frac{n}{\epsilon})$ than those of the optimal solution. In [2] Aiello *et al.* proved that algorithm *NTG* (Nearest To Go) is $O(md)$ competitive for any network, where m is the number of edges in the network and d is the maximal length of a path traversed by any packet. They also showed that on DAGs any greedy algorithm is $O(md)$ competitive.
- **Packet switching:** Switches are fundamental part of most networks and thus studied extensively during recent years in the context of throughput. A multi-queue input switch receives packets from m input ports and transmits packets from a single output port. The best algorithm for the multi-queue input switch of unit value packets is 1.58-competitive (see [15]) when the switch input queues are larger than $\log m$. For smaller queues there is a 1.89-competitive algorithm (see [3]). The best algorithm for a multi-queue input switch of packets with general value is 3-competitive (see [17]). The best algorithm for maximizing the total weighted throughput of *CIOQ* switches is 8-competitive (see [16]). More work on other models of switches can be found in [26, 28].

Other approaches for packet routing: Works under the assumption that the adversary never overload the system with packets are as follows: [4, 22, 23, 33] deals with queuing policies alone (the path of each packet is given). [1, 8, 10, 11, 23] are studies of adversarial models in the context of routing (i.e the adversary does not reveal the paths). In [30] Kothapalli *et al.* investigated information gathering in lines and rings. They proved that in order to avoid packets drop in such networks, an online algorithm should have buffers larger by at least a logarithmic factor than the buffers of the optimal solution.

Online call admission and circuit routing: Call admission and circuit routing have been studied in a variety of contexts. Gary *et al.* [24] have studied these problems on a line topology where preemption is allowed. Awerbuch *et al.* [7] investigated the problem on general graphs in the case that the bandwidth of each request is relatively small compared to the capacity of the edges. Awerbuch *et al.* [12] and Awerbuch *et al.* [13] considered admission control on trees using randomization. Blum *et al.* [20] and Kleinberg *et al.* [29] investigated these problems on some special topologies such as trees and meshes. Work on the load model, where all requests are accepted can be found in [6, 32, 18, 14].

Paper structure: For simplicity of the presentation we consider in sections 2 up to 6 the line topology. In section 7 we show how to extend the results to rings and trees. Section 2 includes formal definitions and notation. In Section 3 we consider the online fractional call admission and circuit routing problem. In Section 4 we construct an $O(\log n)$ competitive algorithm for the fractional information gathering in a line network. In Section 5 we transform the outcome fractional algorithm from Section 4 into a discrete algorithm for information gathering with competitiveness of $O(\log n)$. In Section 6 we construct an $O(\log^2 n)$ competitive randomized algorithm for the multi-destination routing. In Section 7 we extend our results to trees and rings. In appendix A we give an $O(\log n)$ competitive algorithm for information gathering in trees.

2 Problem Definition and Notations

There are two major routing types in communication networks: packet routing and virtual circuit routing. In circuit routing paths connections are constructed while in packet routing packets are traversed in the network. In this paper we consider packet routing defined below. Interestingly, our results are based also on virtual circuit routing defined in Section 3.

In the online Packet Routing problem on a line we have a network organized in a line topology of length n, i.e. node $i = 0, \dots, n-2$ is connected to node $i+1$ via a unidirectional link with unit capacity. Node $i = 0, \dots, n-1$ contains a buffer of size B, which is initially empty, to buffer the packets waiting to be transmitted via its outgoing link. In information gathering we may assume without loss of generality that node $n-1$ is the destination of all packets while in the multi-destination routing the destination of a packet may be any node. We assume time proceeds in discrete steps, and each time step $t \geq 0$ is divided into two phases: at the first phase new packets may be injected to nodes $i = 0, \dots, n-1$, each packet is associated with a destination node. During the second phase of time t, node $i = 0, \dots, n-2$ may transmit a packet from its buffer to node $i+1$. If a packet reaches its destination it is absorbed. Otherwise, online arriving packets (from both phase 1 and phase 2) can be buffered without exceeding the buffers capacities. Remaining packets must be discarded. The goal is to maximize the number of packets that reach their destination.

We use the term *load* of buffer i to refer to the number of packets residing in that buffer. Given an online algorithm A we denote by $A(\sigma)$ the value of A given the sequence σ. We denote the optimal (offline) algorithm by OPT, and use similar notation for it.

Remark 1 *Packet arrives at a full buffer (at the injection phase or the transmission phase) must be discarded. However our algorithms discard packets only at the injection phase, which make them suitable also for a non-preemptive model in which accepted packets must reach their destination.*

3 Online Fractional Call Admission and Circuit Routing

Our routing algorithm for information gathering is based on an online fractional call admission and circuit routing algorithm. Thus we start by considering the online version of the call admission and circuit routing problem, which is defined as follows. A network is represented by a capacitated graph $G(V, E, u)$ and a bound D on the allowed maximum length of paths used. The capacity $u(e)$ assigned to each edge $e \in E$ represents the bandwidth available on this edge. The online input sequence consists of call requests for paths: $\beta_1, \beta_2, \dots, \beta_k$, where the ith call is represented by: $\beta_i = \{s_i, t_i, r_i\}$. Node

s_i is the origin of the call β_i, node t_i is its destination, and r_i is the bandwidth it requires. Upon receiving a call request β_i an algorithm either routes it by assigning it a path of length at most D from s_i to t_i with r_i bandwidth, or rejects it. If it routes the call then the available bandwidth of each edge on the path decreases by r_i and the throughput of the algorithm increases by r_i. The goal of an algorithm is to maximize its throughput while maintaining the capacity constraints.

The fractional version of the problem is defined as follows: an algorithm can split a call into smaller bandwidth calls and treat each one of them as a separate call. Thus an algorithm can route different fractions of a call β_i in different paths from s_i to t_i, and reject the remaining fraction. The throughput of the algorithm is the total bandwidth of routed fractions.

Awerbuch *et al.* [7] investigated the integral version of the problem. They proved $O(\log D)$ competitive algorithm for the case that the bandwidth of each request is relatively small compared to the capacity of the edges. Specifically they assumed that for each call β_i , $r_i \leq \frac{min_e\{u(e)\}}{\log(2D+2)}$. We call their algorithm AAP.

The low bandwidth assumption is required to achieve a poly-logarithmic competitive algorithm. We show how to easily overcome this assumption by modifying the AAP algorithm and allowing it to route fractional calls. We call this fractional algorithm $FAAP$ (presented in Figure 1).

Algorithm $FAAP$

Upon arrival of the call β_i :

1. Split β_i into calls of bandwidth $\frac{min_e\{u(e)\}}{\log(2D+2)}$.(The last fraction of β_i might be of smaller bandwidth).
2. Run AAP in sequence on β_i fractions , until it rejects a fraction or all fractions have been routed.

Figure 1: Algorithm $FAAP$.

Theorem 3.1 *Algorithm $FAAP$ is $O(\log D)$ competitive for maximizing the throughput (even compared to a fractional OPT).*

Proof: For low bandwidth calls, algorithm AAP is $O(\log D)$ competitive even against a fractional OPT (see [21],exercise 13.3). By splitting the call, the low bandwidth requirement of AAP applies. ■

4 Fractional Packet Routing

In this section we consider a fractional version of the packet routing problem described in Section 2. I.e., we allow an algorithm to accept fractional packets as well as transmit fractional packets from one node to its successor. Each packet fraction that reaches its destination increases the algorithm throughput by its size. The purpose of this section is to construct an $O(\log n)$ competitive fractional algorithm for information gathering. We also assume that input sequence σ consists of *integral* packets. However, this restriction is not obligatory for this section. The restriction is relevant for the transformation of a fractional algorithm for the packet routing problem into a discrete routing algorithm (see Subsection 5.1).

4.1 Fractional packet routing with bounded delay

In this section we consider a fractional variant of the multi-destination routing in which a packet fraction must not stay more than T time steps in the network.

We begin by introducing a translation of this problem into the problem of fractional call admission and circuit routing which was described in Section 3. The graph $G = (V, E, u)$ for the fractional call admission problem is described in Figure 2.

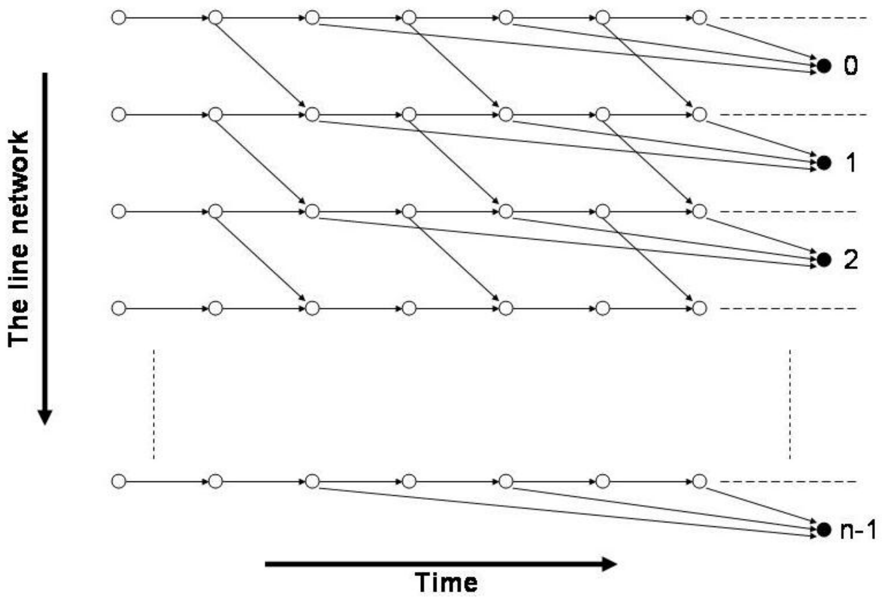

Figure 2: The graph G for the call admission and circuit routing problem

Exploiting the matrix shape in which the white nodes in Figure 2 are positioned, we refer to a white node according to its coordinates in the matrix. The node at the left top corner is node $\{0, 0\}$.

The i'th row in the graph represents the i'th buffer over all times. Each time step is represented by two consequent columns. The beginning of the injection phase of time step t is represented by column number $2t$. The consequent column represents the beginning of the transmission phase of time step t. We assign a capacity B to each edge $(\{2t, i\},\{2t + 1, i\})$ for every $t \geq 0$ and $0 \leq i \leq n - 1$. Those edges represent the buffers resources. We assign a capacity 1 to all diagonal edges between two white nodes. They represent the links resources. All other edges capacity is ∞.

We set the bound D on the allowed maximum length of a path used in the translated fractional call admission and circuit routing problem to be $2T + 1$ (since each time step corresponds to two consequent edges and the last edge that reach the destination node has no time meaning).

Based on the graph description, we translate the input sequence σ. Suppose a packet $p \in \sigma$ was injected to node i at time step t and its destination is node j. We translate p to the following call request c on G: The origin of c is the white node $\{2t, i\}$. The destination of c is the black node j. The bandwidth c requires is 1.

Next we show the equivalence between the fractional packet routing with bounded delay problem and the translated fractional call admission and circuit routing problem.

Claim 4.1 *Every fractional solution $\mathcal{A}$ to the translated fractional call admission and circuit routing problem can be mapped, in an online fashion, to a fractional solution, with the same throughput, to the original packet routing with bounded delay problem.*

Proof: We construct a solution to the original packet routing with bounded delay problem according to solution $\mathcal{A}$. Let p be a packet and c its translation to a call. If $\mathcal{A}$ splits c, we split p in a way each call fraction has a corresponding packet fraction with size equal to the call fraction bandwidth. If $\mathcal{A}$ rejects a call fraction, we reject its corresponding packet fraction. Otherwise the path $\mathcal{A}$ assigns to the call fraction dictates the routing of its corresponding packet fraction as follows. Let $\{2t+1, i\}$ be a white node on the assigned path. Then the path includes either the horizontal edge ($\{2t+1, i\}$,$\{2t+2, i\}$) or the diagonal edge ($\{2t+1, i\}$,$\{2t+2, i+1\}$). In the first case we hold up the packet fraction at buffer i at time step t. In the second case we transmit the packet fraction from buffer i to buffer $i+1$ at the transmission phase of time step t. The capacities of the edges of graph G and the feasibility of solution $\mathcal{A}$, assure that the constructed solution to the fractional packet routing with bounded delay problem doesn't require buffers larger than B, and links capacity greater than 1. Since we set the bound D on the allowed maximum length of path assigned to a call fraction to be $2T+1$ and since each time step is represented by two consequent columns of G, no packet fraction in the constructed solution stays more than T time steps in the network. Clearly, the constructed solution throughput is equal to the throughput of solution $\mathcal{A}$. ■

Claim 4.2 *Every fractional solution $\mathcal{S}$ to the packet routing with bounded delay problem can be mapped to a solution, with the same throughput, to the translated fractional call admission and circuit routing problem.*

Proof: We construct a solution to the translated fractional call admission and circuit routing problem according to solution $\mathcal{S}$. Let p be a packet and c its translation to a call. A solution to the packet routing with bounded delay problem can split p several times during its delay in the network. W.L.O.G we assume $\mathcal{S}$ splits p into its final fractions at its injection phase. We split c in a way each packet fraction has a corresponding call fraction with bandwidth equal to the fraction size. If $\mathcal{S}$ rejects a packet fraction, we reject its corresponding call fraction. Otherwise, we assign a path to the call fraction according to the routing of its corresponding packet fraction in solution $\mathcal{S}$. If $\mathcal{S}$ hold up a packet fraction at buffer i at time step t, we assign the edge ($\{2t+1, i\}$,$\{2t+2, i\}$) for the path of its corresponding call fraction. Otherwise, if $\mathcal{S}$ transmits the packet fraction from node i to node $i+1$ at time step t, we assign the edge ($\{2t+1, i\}$,$\{2t+2, i+1\}$) for the path of its corresponding call fraction. After those even edges on the path were assigned, the assignment of the odd edges is obvious. The feasibility of the constructed solution to the translated call admission and circuit routing problem is derived from the feasibility of solution $\mathcal{S}$. Since in $\mathcal{S}$ no packet fraction stays more than T time steps in the network, no path length in the constructed solution is greater than $2T+1$. Clearly, the constructed solution throughput is equal to the throughput of solution $\mathcal{S}$. ■

Corollary 4.3 *For any sequence σ, the throughput of the optimal solution in the translated call admission and circuit routing problem is equal to the optimal solution in the fractional packet routing with bounded delay problem.*

In Figure 3 we present the online algorithm BPR for the fractional packet routing with bounded delay problem.

Algorithm BPR (Bounded delay Packet Routing)

- Maintain a running simulation of $FAAP$ on G with $D = 2T+1$ on the translated input sequence σ.
- Construct a solution to the fractional packet routing with bounded delay problem from the solution of the simulated $FAAP$ as described in claim 4.1.

Figure 3: Algorithm BPR.

Theorem 4.4 *Algorithm BPR is feasible and $O(\log T)$ competitive for fractional packet routing with bounded delay.*

Proof: The theorem follows immediately from Claim 4.1, Corollary 4.3 and Theorem 3.1. ■

Remark 2 *The technique presented in Subsection 4.1 can be generalized to reduce fractional packet routing in general graphs to circuit routing.*

4.2 An $O(\log(nB))$ algorithm for fractional information gathering

In this subsection we consider the fractional version of information gathering with no bound on the delay. We show that given a solution with unbounded delay, we can construct a solution with bounded delay $T = 2nB$ with halved throughput. This implies $O(\log(nB))$ competitive fractional algorithm PR for information gathering (presented in Figure 4) based on the BPR algorithm.

Algorithm PR (Packet Routing)
Apply algorithm BPR with bounded delay $T = 2nB$.

Figure 4: The PR algorithm

Definition 4.1 *A buffer is applying a* greedy transmission *if it transmits the minimum between the link bandwidth and the sum of the fractions it holds.*

Definition 4.2 *A packet is* new *until the first divisible by nB time step that follows its injection, and* old *afterwards.*

Claim 4.5 *For every unbounded routing solution S for a sequence σ, there is a solution with a bounded delay $T = 2nB$ and a throughput of $\frac{1}{2} \cdot S(\sigma)$.*

Proof: We denote by S_{2nB}^{frac} the fractional solution derived from S, which doesn't keep any packet fraction more than $2nB$ time steps in the network, but yields a throughput of $\frac{1}{2} \cdot S(\sigma)$. In S_{2nB}^{frac} we virtually partition the network resources (i.e. buffers and links bandwidth) into two equal parts. In the first part we run a simulation of S on σ, while halving each accepted/transmitted packet fraction. Every nB time steps we transfer the remaining fractions in the first part of each buffer to its second part. Thus, the first part of the network contains only new packet fractions, while the second part contains only old packet fractions. We apply a greedy transmission in the second part of the network. Clearly the transfer process is harmless to the simulation upon new fractions in the first part of the network as it only frees the space taken by fractions that became old. Since the sum of the transfered old fractions never exceeds $\frac{1}{2} \cdot nB$ and due to the greedy transmission, no old fraction stays more than nB time steps in the second part of the network. ■

Corollary 4.6 *For every input sequence σ, $OPT^{frac}(\sigma) \leq 2 \cdot OPT_{2nB}^{frac}(\sigma)$*

Theorem 4.7 *Algorithm PR is feasible and $O(\log(nB))$ competitive for fractional information gathering with no bounded delay assumptions.*

Proof: Since algorithm PR actually uses algorithm BPR in the model described in Subsection 4.1 where the delay bound T is equal to $2nB$, we get:

$$\begin{aligned} OPT^{frac}(\sigma) &\leq 2 \cdot OPT_{2nB}^{frac}(\sigma) \\ &\leq 2 \cdot O(\log(2nB)) \cdot BPR_{2nB}(\sigma) = O(\log(nB)) \cdot PR(\sigma) \end{aligned}$$

where the first inequality is obtained from Corollary 4.6 and the second from Theorem 4.4. ■

4.3 Reduction from buffers of size $B > n$ to buffers of size n

In this subsection we give a generic technique to construct a fractional algorithm for information gathering with large buffers from an algorithm for small buffers. Specifically, given a c-competitive algorithm for buffers of size n, we can construct a $16c$-competitive fractional algorithm for buffers of size $B > n$. We call this technique GR (Generic Reduction). We assume throughout this subsection that $n|B$. We get rid of this assumption at the end of this subsection.

Note that given a c-competitive algorithm for buffers of size n immediately implies a $2c$-competitive fractional algorithm for buffers of size $\frac{n}{2}$. This is done by running the first algorithm while halving each accepted/transmitted packet fraction. Furthermore, given a $2c$-competitive algorithm Alg for buffers of size $\frac{n}{2}$ immediately implies a $2c$-competitive algorithm for buffers of size $\frac{B}{2}$ and links of bandwidth $\frac{B}{n}$. This is done by running algorithm Alg on the input sequence scaled down by factor $\frac{B}{n}$, and adopt its decisions while scaling them up by the same factor. Thus for simplicity we assume throughout this subsection we are given the $2c$-competitive algorithm A for fractional information gathering with buffers of size $\frac{B}{2}$ and bandwidth of $\frac{B}{n}$. We will show that applying the GR technique on algorithm A generates a $16c$-competitive algorithm for information gathering with buffers of size $B > n$.

By the following definition we combine every $\frac{B}{n}$ consecutive time steps.

Definition 4.3 *We define the l'th* time interval *$(l \geq 0)$ as time steps $l \cdot \frac{B}{n}$ upto $(l+1) \cdot \frac{B}{n} - 1$.*

Definition 4.4 *We define the l'th* border time *as the time between the end of the l'th time interval and the beginning of time interval $l+1$.*

We denote by $\hat{\sigma}$ the input sequence σ in which for each buffer we concatenate packets injected during the same time interval. Informally, the idea of the technique is to simulate algorithm A which runs in time intervals on the sequence $\hat{\sigma}$, by transmitting in the original sequence σ during the time steps contained in the time interval. We denote by $R_{i,l}$ the quantity of the packet fractions which was injected to buffer i at time interval l and accepted by the simulation of algorithm A on $\hat{\sigma}$. We denote by $T_{i,l}$ the quantity of packet fractions which was transmitted from buffer i at time interval l by the simulation of algorithm A on $\hat{\sigma}$. In Figure 5 we give the exact definition of GR with algorithm A as a parameter.

Algorithm GR^A

- Virtually partition each buffer of GR^A to *upper* buffer and *lower* buffer. Each of size $\frac{B}{2}$.
- Run a simulation of algorithm A in time intervals on the sequence $\hat{\sigma}$.
- For each node $i = 0, \ldots, n-1$ at every time step t:
  Let $l = \lfloor \frac{t}{(\frac{B}{n})} \rfloor$.
  1. **Injection phase:** Accept packet fractions into the upper buffer unless it's full.
  2. **Transmission phase:** Transmit $\frac{T_{i,l-1}}{(\frac{B}{n})}$ fractions of packets from the lower buffer to the lower buffer of the next node.
- At border time l move $R_{i,l}$ fractions of packets from the upper buffer to the lower buffer.

Figure 5: Algorithm GR^A

We will show that GR^A is $16c$-competitive. We start with the following theorem.

Theorem 4.8 *Algorithm GR^A on σ is feasible and yields the same throughput as algorithm A on $\hat{\sigma}$, i.e., $GR^A(\sigma) = A(\hat{\sigma})$.*

Proof: We begin with the feasibility. Clearly at the l'th border time, the load of the upper part of buffer i of GR^A is at least $R_{i,l}$, since at least $R_{i,l}$ packets were injected at that phase and $R_{i,l} \leq \frac{B}{2}$. Hence there are $R_{i,l}$ fractions of packets to move from the upper buffer to the lower buffer. Since $\frac{T_{i,l-1}}{(\frac{B}{n})} \leq 1$ the bandwidth constraint is maintained in the transmission phase. By a simple induction which uses the feasibility of algorithm A on $\hat{\sigma}$, at the l'th border time the load of the lower part of buffer i of GR^A on σ is equal to the load of buffer i of A on $\hat{\sigma}$ before the injection phase of time interval l, i.e the lower part of buffer i can accept $R_{i,l}$ fractions of packets from the upper part of buffer i, and transmit fractions of packets in a total quantity of $T_{i,l}$ during time interval $l+1$. This also implies that the throughput of algorithm GR^A on σ is equal to the throughput of algorithm A on $\hat{\sigma}$. ■

Next we will show that the fractional optimal solution with bandwidth $\frac{B}{n}$ and buffers of size $\frac{B}{2}$ on $\hat{\sigma}$ yields at least a quarter of the throughput of the fractional optimal solution which uses buffers of size B and links of bandwidth 1 on σ. Before we proceed we introduce some notation. We denote by $OPT^{frac}_{B,1}$ the fractional optimal (offline) algorithm which runs in original time steps on σ and uses buffers of size B and links of bandwidth 1. We denote by $\mathcal{R}_{i,l}$ the total quantity of packet fractions that was injected to buffer i during the l'th time interval and was accepted by $OPT^{frac}_{B,1}$. We further denote by $\mathcal{T}_{i,l}$ the total quantity of packet fractions $OPT^{frac}_{B,1}$ has transmitted from buffer i during the l'th time interval. Next we describe the fractional strategy S on $\hat{\sigma}$ with bandwidth $\frac{B}{n}$ and buffers of size $2B$, which imitate $OPT^{frac}_{B,1}$ on σ. (see Figure 6).

Strategy S
For each node $i = 0, \ldots, n-1$ at every time interval l:

1. **Injection phase:** accepts packet fractions in a quantity equal to $\mathcal{R}_{i,l}$.
2. **Transmission phase:** transmit packet fractions in a quantity equal to $\mathcal{T}_{i,l-i}$ (if $l < i$ transmit nothing).

Figure 6: Strategy S

Theorem 4.9 *For any finite sequence σ Strategy S on $\hat{\sigma}$ is feasible with buffers of size $2B$ and bandwidth of $\frac{B}{n}$ and $S(\hat{\sigma}) = OPT^{frac}_{B,1}(\sigma)$.*

Proof: We begin with the feasibility of the transmission phase.

Claim 4.10 *Assume the buffers of strategy S are unbounded, then the transmission phase of strategy S is feasible.*

Proof: Since $\mathcal{T}_{i,l-i} \leq \frac{B}{n}$, and since strategy S uses bandwidth of $\frac{B}{n}$, buffer i of S can transmit this quantity if it is available. We prove that this quantity of packet fractions is available for buffer i of strategy S by induction on the time intervals. For time intervals $0 \ldots i-1$, buffer i of strategy S is not required to transmit. We assume correctness until time interval $l \geq i-1$ and prove for time interval $l+1$. Note that the load of a buffer i is always equal to the subtraction of the accumulated sum of transmitted fractions of packets from the accumulated sum of accepted fractions of packets from both buffer $i-1$ and the injection to buffer i. By the induction hypothesis on buffer $i-1$ and buffer i, together with the definition of strategy S at the injection phase, the load of buffer i of strategy S before the transmission phase of time interval $l+1$ is equal to:

$$\begin{aligned} \sum_{k=0}^{l-i+1} \mathcal{T}_{i-1,k} + \sum_{k=0}^{l+1} \mathcal{R}_{i,k} - \sum_{k=0}^{l-i} \mathcal{T}_{i,k} &\geq \\ \sum_{k=0}^{l-i+1} \mathcal{T}_{i-1,k} + \sum_{k=0}^{l-i+1} \mathcal{R}_{i,k} - \sum_{k=0}^{l-i} \mathcal{T}_{i,k} &\geq T_{i,l-i+1} \end{aligned}$$

where the first inequality follows from a decreased summation range (on non negative numbers), and the second inequality follows from the fact that algorithm $OPT^{frac}_{B,1}$ on σ has transmitted $\mathcal{T}_{i,l-i+1}$ fractions of packets from a quantity which is equal to the middle expression. ■

Corollary 4.11 *If the buffers of strategy S are unbounded, then for every finite sequence σ, $S(\hat{\sigma}) = OPT^{frac}_{B,1}(\sigma)$.*

Now we bound the load of the buffers of strategy S.

Claim 4.12 *The load of buffer i of strategy S never exceeds $B + (i+1) \cdot \frac{B}{n}$.*

Proof: The load of buffer i of strategy S after the injection phase of the l'th time interval is equal to: $\sum_{k=0}^{l-i} \mathcal{T}_{i-1,k} + \sum_{k=0}^{l} \mathcal{R}_{i,k} - \sum_{k=0}^{l-i-1} \mathcal{T}_{i,k}$. Note that $OPT^{frac}_{B,1}$ can transmit at most $\frac{B}{n}$ fractions of packets during each time interval. Hence $\sum_{k=0}^{l} \mathcal{T}_{i,k} \leq \sum_{k=0}^{l-i-1} \mathcal{T}_{i,k} + (i+1) \cdot \frac{B}{n}$. By this and by increasing a summation range of non negative numbers, the load of buffer i of strategy S after the injection phase of the l'th time interval is at most:

$$\sum_{k=0}^{l} \mathcal{T}_{i-1,k} + \sum_{k=0}^{l} \mathcal{R}_{i,k} - \left[\sum_{k=0}^{l} \mathcal{T}_{i,k} - (i+1) \cdot \frac{B}{n}\right] = \left[\sum_{k=0}^{l} \mathcal{T}_{i-1,k} + \sum_{k=0}^{l} \mathcal{R}_{i,k} - \sum_{k=0}^{l} \mathcal{T}_{i,k}\right] + (i+1) \cdot \frac{B}{n}$$

which is the load of buffer i of $OPT^{frac}_{B,1}$ after the l'th time interval plus $(i+1) \cdot \frac{B}{n}$. Similarly the load of buffer i of strategy S after the transmission phase of the l'th time interval is at most the load of buffer i of $OPT^{frac}_{B,1}$ after time interval $l+1$ plus $(i+1) \cdot \frac{B}{n}$. ■

Corollary 4.13 *The load of the buffers of strategy S never exceeds $B + n \cdot \frac{B}{n} = 2B$.*

The proof of Theorem 4.9 follows immediately from Corollary 4.13, Claim 4.10 and Corollary 4.11. ■

Denote by $OPT^{frac}_{\frac{B}{2},\frac{B}{n}}$ the fractional optimal (offline) algorithm which uses buffers of size $\frac{B}{2}$ and links of bandwidth $\frac{B}{n}$.

Theorem 4.14 *For every $B > n$ and every finite sequence σ, $OPT^{frac}_{B,1}(\sigma) \leq 4 \cdot OPT^{frac}_{\frac{B}{2},\frac{B}{n}}(\hat{\sigma})$.*

Proof: By Theorem 4.9, running a simulation of strategy S on $\hat{\sigma}$, while shrinking each accepted and transmitted packet fraction by $\frac{1}{4}$ gives a solution, not necessary optimal, which uses buffers of size $\frac{B}{2}$ and links of bandwidth $\frac{B}{n}$ (actually this solution requires link bandwidth of only $\frac{B}{4n}$) and yields a throughput of $\frac{1}{4} \cdot OPT^{frac}_{B,1}(\sigma)$. ■

Now we are ready to prove that GR^A is $16c$ competitive. In proving so we also get rid of the assumption that $n|B$.

Theorem 4.15 *Let $B' \leq B$ the largest number such that $n|B'$ and let A be a $2c$-competitive algorithm for fractional information gathering with bandwidth $\frac{B'}{n}$ and buffers of size $\frac{B'}{2}$, then algorithm GR^A is $16c$-competitive for fractional information gathering with buffers of size B (and links of bandwidth 1).*

Proof: For every sequence σ:

$$\begin{aligned} OPT^{frac}_{B,1}(\sigma) &\leq 2 \cdot OPT^{frac}_{B',1}(\sigma) \\ &\leq 2 \cdot 4 \cdot OPT^{frac}_{\frac{B'}{2},\frac{B'}{n}}(\hat{\sigma}) \\ &\leq 8 \cdot 2c \cdot A^{frac}_{\frac{B'}{2},\frac{B'}{n}}(\hat{\sigma}) \\ &\leq 8 \cdot 2c \cdot GR^A(\sigma) = 16c \cdot GR^A(\sigma) \end{aligned}$$

Where the first inequality is obtained from the fact $B' > \frac{B}{2}$, the second inequality is obtained from Theorem 4.14 and the last inequality is obtained from Theorem 4.8. Note that GR^A uses only $B' \leq B$ buffer space and links of bandwidth 1. ■

Corollary 4.16 *Theorem 4.15 implies that given a c-competitive algorithm for buffers of size n, we can construct a $16c$-competitive algorithm for fractional information gathering with buffers of size $B > n$ by applying GR^A as above.*

4.4 Fractional information gathering - an $O(\log n)$ competitive algorithm

In this section we present an $O(\log n)$ competitive algorithm for fractional information gathering. For that purpose we use algorithm PR for information gathering, presented in Subsection 4.2, and the reduction to information gathering with size of buffers n presented in Subsection 4.3. Algorithm FIG for fractional information gathering is presented in Figure 7.

Algorithm FIG (Fractional Information Gathering)

- **if** $B \leq n$:
  Use algorithm PR.
- **else**
  1. Let A be algorithm PR for buffers of size $\frac{n}{2}$ scaled up by $\lfloor \frac{B}{n} \rfloor$.
  2. Use algorithm GR^A.

Figure 7: Algorithm FIG.

Theorem 4.17 *Algorithm FIG is feasible and $O(\log n)$ competitive for fractional information gathering.*

Proof: By Theorem 4.7 the fractional algorithm PR for information gathering is feasible and $O(\log(nB))$ competitive. This immediately implies an $O(\log n)$ competitiveness for buffers of size $B \leq n$. Next we consider $B > n$. Clearly A has buffers of size $\frac{n}{2} \cdot \lfloor \frac{B}{n} \rfloor \leq \frac{B}{2}$. Hence GR^A is feasible and by corollary 4.16 GR^A is $16 \cdot O(\log n) = O(\log n)$ competitive. ■

5 Discrete Information Gathering

In this section we consider discrete information gathering. Given a sequence σ which consists of *integral* packets, we present a generic local technique for buffers of size $B \geq 2$ to transform any online fractional algorithm for information gathering into a discrete algorithm with competitiveness multiplied by $\frac{B}{B-1}$. In particular, we use the fractional algorithm FIG which was presented in Subsection 4.4, to construct a discrete algorithm with $O(\log n)$ competitiveness.

5.1 Discretization of a fractional packet routing algorithm

Given a sequence σ which consists of *integral* packets, we present a generic local technique to transform any fractional algorithm A for information gathering, that uses buffers of size B, into a discrete algorithm with the same throughput, but uses buffers of size $B+1$. Alternatively we will show that for $B \geq 2$ this technique can be used to transform any c-competitive fractional algorithm for the problem into a discrete algorithm with a competitive ratio of $c \cdot \frac{B}{B-1}$ which doesn't require additional buffer space. Before we proceed we introduce some notations. Given a sequence σ which consists of *integral* packets and a fractional (or discrete) algorithm Alg, we denote by $T_i^{Alg}(t)$ the accumulated sum of packet fractions Alg has transmitted from node i until time step t inclusive. We denote by $R_i^{Alg}(t)$ the accumulated sum of packet fractions that were injected to node i until time step t inclusive and were accepted by Alg. In Figure 8 we present the definition of RU with algorithm A as a parameter. This technique rounds up quantities from algorithm A. Let $A' = RU^A$.

Algorithm $A' = RU^A$

Run a simulation of algorithm A (in the fractional model) with the input sequence σ.
For each node $i = 0, \ldots, n-1$ at every time step t:

1. **Injection phase:** Accept packets until $R_i^{A'}(t) = \lceil R_i^A(t) \rceil$.
2. **Transmission phase:** Transmit a packet only if it is necessary to hold the equality : $T_i^{A'}(t) = \lceil T_i^A(t) \rceil$.

Figure 8: Algorithm $A' = RU^A$.

Theorem 5.1 *Suppose the buffers of algorithm A' are larger than those of algorithm A by one slot each. Then for every input sequence σ, algorithm A' is feasible and $A'(\sigma) \geq A(\sigma)$.*

Proof: We begin with the feasibility of algorithm A'. Our first goal is to show it can maintain the equality $R_i^{A'}(t) = \lceil R_i^A(t) \rceil$ at the injection phase of time step t, and the equality $T_i^{A'}(t) = \lceil T_i^A(t) \rceil$ at the transmission phase of time step t.

Lemma 5.2 *Assume the buffers of algorithm A' are unbounded, then algorithm A' can keep the two equalities at each time step.*

Proof: We prove the lemma by induction on the time steps. Before the first time step the equalities clearly hold. We assume correctness for time step $t-1$, and prove that algorithm A' can keep the equalities at time step t. We first consider the injection phase at buffer i. According to the induction hypothesis $R_i^{A'}(t-1) = \lceil R_i^A(t-1) \rceil$. Thus the number of *integral* packets algorithm A' needs to accept, in order to keep the equality at time step t, doesn't exceed the number of packets algorithm A accepted fractions from at time step t. Next, consider the transmission phase at buffer i. The sum of transmitted packet fractions by algorithm A at time step t is at most 1. By the induction hypothesis $T_i^{A'}(t-1) = \lceil T_i^A(t-1) \rceil$. Then either the equality still holds or algorithm A' must transmit a single *integral* packet in order to keep the equality at time step t. So it remains to show that if $T_i^{A'}(t-1) < \lceil T_i^A(t) \rceil$ then algorithm A' has a packet to transmit. The total sum of packet fractions algorithm A has received at buffer i so far is $T_{i-1}^A(t-1) + R_i^A(t)$. The total sum of packet fractions algorithm A has transmitted from buffer i so far is $T_i^A(t)$. Thus $T_{i-1}^A(t-1) + R_i^A(t) - T_i^A(t) \geq 0$. By the induction hypothesis on the previous transmission phase at buffer $i-1$ we have $T_{i-1}^{A'}(t-1) \geq T_{i-1}^A(t-1)$. Based on what we proved regarding the injection phase at buffer i, $R_i^{A'}(t) \geq R_i^A(t)$. Thus if $T_i^{A'}(t-1) < \lceil T_i^A(t) \rceil$, then:

$$T_{i-1}^{A'}(t-1) + R_i^{A'}(t) - T_i^{A'}(t-1) > T_{i-1}^A(t-1) + R_i^A(t) - T_i^A(t) \geq 0$$

which implies that algorithm A' has a packet to transmit. ■

Lemma 5.3 *The load of buffer i of algorithm A' is always smaller than the load of buffer i of algorithm A plus two.*

Proof: We prove the claim for an arbitrary time step t. The load of buffer i is always equal to the subtraction of the accumulated sum of transmitted packets from the accumulated sum of accepted packets from both buffer $i-1$ and the injection to buffer i. By this and by lemma 5.2 the load of buffer i of algorithm A' after the injection phase of time step t is equal to:

$$\begin{array}{c} T_{i-1}^{A'}(t-1) + R_i^{A'}(t) - T_i^{A'}(t-1) = \\ \lceil T_{i-1}^A(t-1) \rceil + \lceil R_i^A(t) \rceil - \lceil T_i^A(t-1) \rceil \quad < \\ T_{i-1}^A(t-1) + R_i^A(t) - T_i^A(t-1) + 2 \end{array}$$

where the inequality is obtained from the ceiling properties. The last expression is equal to two plus the load of buffer i of algorithm A after the injection phase of time step t. A similar argument holds for the transmission phase. ■

Claim 5.4 *Let B be the size of the buffers of algorithm A, then algorithm A' is feasible using buffers of size $B+1$.*

Proof: The load of a buffer of A doesn't exceed B. Thus by lemma 5.3 the load of a buffer of A' is always smaller than $B+2$. Since A' is a discrete algorithm the load of its buffers is at most $B+1$. ■

Claim 5.5 *For every input sequence σ , $A'(\sigma) \geq A(\sigma)$.*

Proof: By lemma 5.2, for every buffer i at every time step t, algorithm A' holds the equalities $T_i^{A'}(t) = \lceil T_i^A(t) \rceil$ and $R_i^{A'}(t) = \lceil R_i^A(t) \rceil$. In particular for $i = n-1$ it proves the claim. ■

Theorem 5.1 follows immediately from claims 5.4 and 5.5. ■

Next we show how to use *RU* to transform any c-competitive fractional algorithm for buffers of size B into a discrete algorithm with buffers of size B whose competitive ratio is $c \cdot \frac{B}{B-1}$ for $B \geq 2$. We call this transformation D (Discretization). Let A be a c-competitive fractional algorithm for information gathering, when using buffers of size B. In Figure 9 we present the definition of D with algorithm A as a parameter.

Theorem 5.6 *Algorithm D^A uses buffers of size B and is $c \cdot \frac{B}{B-1}$ competitive for information gathering.*

Algorithm D^A

1. Run a simulation of algorithm A with the input sequence σ.
2. Let S be the algorithm obtained by shrinking by $\frac{B-1}{B}$ each accepted/transmitted packet fraction of the simulation of A .
3. Apply RU^S.

Figure 9: Algorithm D^A

Proof: Clearly, algorithm S uses buffers of size $B-1$ and yields a throughput of $\frac{B-1}{B} \cdot A(\sigma)$. Therefore for every $B \geq 2$:

$$\begin{aligned} OPT(\sigma) &\leq c \cdot A(\sigma) \\ &= c \cdot \frac{B}{B-1} \cdot S(\sigma) \\ &\leq c \cdot \frac{B}{B-1} \cdot RU^S(\sigma) = c \cdot \frac{B}{B-1} \cdot D^A(\sigma) \end{aligned}$$

where the last inequality is obtained from Theorem 5.1. Note that D^A is a discrete algorithm which uses buffers of size B. ■

5.2 An $O(\log n)$ algorithm for information gathering

In this subsection we apply the discretization technique D from Subsection 5.1 on algorithm FIG presented in Subsection 4.4 in order to construct an $O(\log n)$ competitive discrete algorithm for information gathering. The discrete algorithm IG for information gathering for buffers of size $B \geq 2$ is presented in Figure 10.

Algorithm IG (Information Gathering)

Use algorithm D^{FIG}.

Figure 10: Algorithm IG.

Theorem 5.7 *Algorithm IG is feasible and $O(\log n)$ competitive for information gathering (even against a fractional OPT).*

Proof: The proof follows immediately from Theorem 4.17 and Theorem 5.6. ■

Remark 3 *In information gathering, packets at a specific buffer are exchangeable. Thus transmitting the packet at the head of the buffer, instead of transmitting an arbitrary packet, will modify our information gathering algorithm to hold in a model in which FIFO queues are imposed.*

6 Multi-Destination Routing

In this section we give an $O(\log^2 n)$ randomized algorithm for multi-destination routing. For this algorithm we use the known technique "Classify and Randomly Select" (e.g [12]) and our result from Subsection 5.2. We classify a packet p according to its source to destination distance d_p , and its source location s_p , by the following rule: let $l = \lfloor \log(d_p) \rfloor$ then classify p into class number $(l, \lfloor \frac{s_p}{2^l} \rfloor \ mod\ 3)$. The above rule classifies the packets into $3 \log n$ disjoint classes.

Definition 6.1 *Nodes $j, \ldots, j+3d-1$ are an* interval *of class (l, a) if $d = 2^l$ and $j \bmod 3d = da$.*

By the classification rule and the interval definition the following fact immediately follows:

Fact 6.1 *A packet of a class is injected at the first third of an interval of the class, and absorbed somewhere in its next two-thirds.*

Definition 6.2 *A* super exit *of an interval of a class is the first node after the first third of the interval.*

In Figure 11 we present algorithm MD for the multi-destination routing:

Algorithm MD (Multi-Destination)

1. Randomly select class i^* uniformly from the $3\log n$ classes.
2. Reject all packets which are not from class i^*.
3. For each interval of the class i^*:
   - Run a simulation of IG in the first third of the interval while changing the destination of all the packets to be the super exit of the interval.
   - Apply the actions of the simulation in the first third of the interval.
   - Apply a greedy transmission in the last two-thirds of the interval.

Figure 11: Algorithm MD.

Theorem 6.1 *The randomized algorithm MD is $O(\log^2 n)$ competitive for multi-destination routing.*

Proof: We begin with the performance of MD within the selected class i^*. We denote by C_{i^*} the throughput of MD within class i^*, and by O_i^* the throughput of the optimum for packets in class i^*.

Claim 6.2 *MD is $O(\log n)$ competitive against the optimum for packets in class i^*.*

Proof: From fact 6.1 the throughput obtained in class i^* is the sum of throughputs obtained at each interval of the class. Thus it's enough to show that on each interval of class i^*, MD is $O(\log n)$ competitive against the optimum for packets in class i^*. Clearly, the throughput of MD on an interval is the throughput of the simulated IG. From fact 6.1, packets from class i^* are injected only at the first third of an interval, thus changing their destination to be the super exit, would not change the throughput of the optimum for packets in class i^* on the interval. Then by theorem 5.7, on each interval of class i^*, MD is $O(\log n)$ competitive against the optimum for packets in class i^*. ■

Therefore for every σ:

$$\begin{aligned} E[MD(\sigma)] &= \sum_{i=0}^{3\log n - 1} Pr[MD \ chooses \ class \ i] \cdot C_i \geq \sum_{i=0}^{3\log n - 1} \frac{1}{3\log n} \cdot \frac{1}{O(\log n)} \cdot O_i \\ &= \frac{1}{O(\log^2 n)} \cdot \sum_{i=0}^{3\log n - 1} O_i \geq \frac{1}{O(\log^2 n)} OPT(\sigma) \end{aligned}$$

Where the first inequality is obtained from Claim 6.2, and the second from the fact that the throughput of OPT within class i can not be more than O_i. ■

Remark 4 *We can easily modify our multi-destination routing algorithm so that it uses FIFO queues with the same competitive ratio (see remark 3).*

7 Extension to Rings and Trees

In this section we extend our results to rings and trees.

1. Information gathering in unidirectional ring is the same as in the line. For the multi-destination routing in a unidirectional ring we can also get $O(\log^2 n)$ competitive algorithm. This can be done in a similar way described in Section 6 with the small modification that the definition of the interval should be considered under the modulo operation.

2. For the multi-destination routing on a bidirectional ring we can also get $O(\log^2 n)$ competitive algorithm. We pick one direction with probability of $\frac{1}{2}$ and apply the algorithm for the unidirectional case. Since the sum of throughputs of the two unidirectional optimal solutions is at least the throughput of the bidirectional optimal solution, the competitiveness of this algorithm is twice the competitiveness of the algorithm for the unidirectional case, i.e $O(\log^2 n)$.

3. Information gathering in trees, i.e., routing packets to a single node of a tree, can be done in the same way described in Section 4 and Section 5. The discretization in the case of a tree results in the multiplication of the competitiveness of the fractional algorithm by a factor of $\frac{B}{B-d+1}$ (instead of $\frac{B}{B-1}$ as shown for the line topology), where d is the maximum input degree of a node. Thus for $B \geq (1+\epsilon)(d-1)$ we obtain an $O(\log n)$ competitive algorithm. For example for binary tree ($d = 3$) we need $B \geq 3$. See appendix A for details.

References

[1] W. Aiello, E. Kushilevitz, and R. Ostrovsky. Adaptive packet routing for bursty adversarial traffic. In *Proc. of the 30th ACM Symp. on Theory of Computing (STOC)*, pages 359–368, 1998.

[2] W. Aiello, R. Ostrovsky, E. Kushilevitz, and A. Rosén. Dynamic routing on networks with fixed-size buffers. In *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms*, pages 771–780, 2003.

[3] S. Albers and M. Schmidt. On the performance of greedy algorithms in packet buffering. In *Proc. 36th ACM Symp. on Theory of Computing*, pages 35–44, 2004.

[4] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proc. 37th IEEE Symp. on Found. of Comp. Science*, pages 380–389, 1996.

[5] S. Angelov, S. Khanna, and K. Kunal. The network as a storage device: Dynamic routing with bounded buffers. In *APPROX*, 2005.

[6] Jim Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 623–631, May 1993.

[7] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In *Proc. 34th IEEE Symp. on Found. of Comp. Science*, pages 32–40, November 1993.

[8] B. Awerbuch, P. Berenbrink, A. Brinkmann, and C. Scheideler. Simple online strategies for adversarial systems. In *Proc. of the 42nd IEEE Symp. on Foundation of Comupter Science (FOCS)*, 2001.

[9] B. Awerbuch, A. Brinkmann, and C. Scheideler. Anycasting and multicasting in adversarial systems: Routing and admission control. In *Proc. 30th ICALP*, pages 1153–1168, 2003.

[10] B. Awerbuch and F. Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *Proc. of the 26th ACM Symp. on Theory of Computing (STOC)*, pages 487–496, 1994.

[11] B. Awerbuch, Y. Mansour, and N. Shavit. End-to-end communication with polynomial overhead. In *Proc. of the 30th IEEE Symp. on Foundation of Comupter Science (FOCS)*, pages 358–363, 1989.

[12] Baruch Awerbuch, Yair Bartal, Amos Fiat, and Adi Rosén. Competitive non-preemptive call control. In *Proc. 5'th ACM-SIAM Symp. on Discrete Algorithms*, pages 312–320, 1994.

[13] Baruch Awerbuch, Rainer Gawlick, Tom Leighton, and Yuval Rabani. On-line admission control and circuit routing for high performance computation and communication. In *Proc. 35th IEEE Symp. on Found. of Comp. Science*, pages 412–423, 1994.

[14] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Raecke. Optimal oblivious routing in polynomial time. In *Proc. of the 35th STOC (San Diego)*, pages 383–388, 2003.

[15] Y. Azar and A. Litichevskey. Maximizing throughput in multi-queue switches. In *Proc. 12th Annual European Symposium on Algorithms*, pages 53–64, 2004.

[16] Y. Azar and Y. Richter. An improved algorithm for CIOQ switches. In *Proc. 12th Annual European Symposium on Algorithms*, pages 65–76, 2004.

[17] Y. Azar and Y. Richter. The zero-one principle for switching networks. In *Proc. 36th ACM Symp. on Theory of Computing*, 2004. 64–71.

[18] M. Bienkowski, M. Korzeniowski, and H. Raecke. A practical algorithm for constructing oblivious routing schemes. In *Proc. of the 15th SPAA (San Diego)*, pages 24–33, 2003.

[19] A. Birman, H. R. Gail, S. L. Hantler, Z. Rosberg, and M. Sidi. An optimal service policy for buffer systems. *Journal of the Association Computing Machinery (JACM)*, 42(3):641–657, 1995.

[20] A. Blum, A. Fiat, H. Karloffa, and Y. Rabani. Personal communication. 1993.

[21] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.

[22] A. Borodin, J.Kleinberg, P. Raghavan, M. Sudan, and D. Williamson. Adversarial queuing theory. In *Proc. 28th ACM Symp. on Theory of Computing*, pages 376–385, 1996.

[23] D. Gamarnik. Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks. In *Proc. of the 31st ACM Symp. on Theory of Computing (STOC)*, pages 206–214, 1999.

[24] Juan Garay, Inder Gopal, Shay Kutten, Yishay Mansour, and Moti Yung. Efficient on-line call control algorithms. *Journal of Algorithms*, 23:180–194, 1997. Also in *Proc. 2'nd Annual Israel Conference on Theory of Computing and Systems*, 1993.

[25] E. Gordon and A. Rosen. Competitive weighted throughput analysis of greedy protocols on dags. In *PODC*, 2005.

[26] E. L. Hahne, A. Kesselman, and Y. Mansour. Competitive buffer management for shared-memory switches. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 53–58, 2001.

[27] A. Kesselman, Z. Lotker, Y. Mansour, and B. Patt-Shamir. Buffer overflows of merging streams. In *Proc. 11th Annual European Symposium on Algorithms*, pages 349–360, 2003.

[28] A. Kesselman and Y. Mansour. Harmonic buffer management policy for shared memory switches. In *INFOCOM*, 2002.

[29] J. Kleinberg and E. Tardos. Disjoint paths in densely embedded graphs. In *Proc. of the 34th Annual IEEE Symposium on the Foundations of Computer Science*, pages 52–61, 1995.

[30] K. Kothapalli and C. Scheideler. Information gathering in adversarial systems: Lines and cycles. In *Proc. 15th ACM Symposium on Parallel Algorithms and Architectures*, 2003.

[31] M. May, J. C. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentiated services for the internet. In *Proceedings of the IEEE INFOCOM 1999*, pages 1385–1394.

[32] H. Raecke. Minimizing congestion in general networks. In *Proc. of the 43rd FOCS (Vancouver)*, pages 43–52, 2002.

[33] C. Scheideler and B. Vocking. From static to dynamic routing: efficient transformations of store-and-forward protocols. In *Proc. of the 31st ACM Symp. on Theory of Computing (STOC)*, pages 215–224, 1999.

A Information Gathering in Trees

In this section we construct an information gathering algorithm for trees. We achieve an $O(\log n)$ competitiveness for size of buffers $B \geq (1+\epsilon)(d-1)$, where d is the maximum input degree of a node. We briefly go through the stages in the construction of the algorithm for the line while emphasizing the needed modifications.

A.1 Fractional Information Gathering with Bounded Delay

In this subsection we develop an $O(\log T)$ competitive algorithm for the problem of fractional information gathering with bounded delay, where T is the bound on the delay of a packet fraction in the network. As in the line case we first translate the problem to the problem of call admission and circuit routing. We first elaborate on the structure of the graph G for the translated call admission and circuit routing problem. Each buffer is represented by an infinite directed line. We assign a B capacity for all odd edges of this line. The capacity of the even edges is unbounded. We connect between those lines according to the tree structure. If buffer i is next to buffer j in the tree, then for all $k \geq 1$ we place a directed edge with a capacity 1 from node $2k-1$ on line j to node $2k$ on line i. The graph G also includes a single node v which represents the destination node. If node r is the root of the tree, then for all $k \geq 0$ we place a directed edge with an unbounded capacity from node $2k$ on line r to node v. Except for node v the nodes of G are positioned in a matrix shape in which each two consequent columns represent one time step. Next we translate the input sequence σ. If packet p was injected to buffer j at time step t, we translate it to a call request of bandwidth 1 from node $2t$ on line j to node v. We set the bound D on the allowed maximum length of a path used in the translated fractional call admission and circuit routing problem to be $2T+1$. Algorithm BIG for fractional information gathering in trees is represented in Figure 12.

Algorithm BIG (Bounded delay Information Gathering)

- Maintain a running simulation of $FAAP$ on G with the translated input sequence σ.
- If $FAAP$ accepted a call fraction, accept its corresponding packet fraction and route it according to the path of the call fraction.

Figure 12: Algorithm BIG.

In a similar way to the one presented in Subsection 4.1 it can be proved that algorithm BIG is $O(\log T)$ competitive for the problem of fractional information gathering with bounded delay.

A.2 An $O(\log(nB))$ algorithm for fractional information gathering

Based on algorithm BIG from Subsection A.1 we construct an $O(\log(nB))$ competitive algorithm for information gathering in trees. Recall that BIG is $O(\log T)$ competitive in a fractional model in which a packet fraction must not stay more than T time steps in the network. In figure 13 we present the fractional algorithm QIG for information gathering on trees.

Algorithm QIG (Quick Information Gathering)

Apply algorithm BIG with bounded delay $T = 2nB$.

Figure 13: The QIG algorithm

In a similar way to the one presented in Subsection 4.2 it can be proved that algorithm QIG is $O(\log(nB))$ competitive for the fractional information gathering problem.

A.3 An $O(\log n)$ competitive algorithm for fractional information gathering

Clearly for $B \le n$ an $O(\log n)$ competitiveness can be achieved by using algorithm QIG directly. As in the line case for larger buffers we use the GR technique presented in Subsection 4.3. Algorithm $FTIG$ for fractional information gathering in trees is presented in Figure 14.

Algorithm $FTIG$ (Fractional Tree Information Gathering)

- **if** $B \le n$:
  Use algorithm QIG.
- **else**
  1. Let A be algorithm QIG for buffers of size $\frac{n}{2}$ scaled up by $\lfloor \frac{B}{n} \rfloor$.
  2. Use algorithm GR^A.

Figure 14: Algorithm $FTIG$.

It can be easily proved that using GR when $B > n$ results in the multiplication of the competitiveness of algorithm A in Figure 14 by a constant. Thus by the competitiveness of algorithm QIG for $B \le n$, algorithm $FTIG$ is $O(\log n)$ competitive.

A.4 Discretization of a fractional Information Gathering algorithm

For size of buffers $B \ge d$, we use the RU technique described in Subsection 5.1 to transform any c-competitive fractional algorithm for information gathering in trees into a $c \cdot \frac{B}{B-d+1}$ - competitive discrete algorithm. Note that the overhead of one slot in each buffer when applying RU in the line case is derived from the fact that the input degree of each node is 2 (see lemma 5.3 and claim 5.4). From this the overhead will not exceed $d-1$ slots when a tree of maximum input degree d is considered. In Figure 15 we present the definition of transformation I with the fractional algorithm A as a parameter.

Algorithm I^A

1. Run a simulation of algorithm A with the input sequence σ.
2. Let S be the algorithm obtained by shrinking by $\frac{B-d+1}{B}$ each accepted/transmitted packet fraction of the simulation of A .
3. Apply RU^S.

Figure 15: Algorithm I^A

By the properties of algorithm A, algorithm S uses buffers of size $B-d+1$ and is $c \cdot \frac{B}{B-d+1}$ competitive against the optimal solution with buffers of size B. Thus by the properties of RU, the discrete algorithm $I^A = RU^S$ uses buffers of size B and is $c \cdot \frac{B}{B-d+1}$ competitive.

A.5 An $O(\log n)$ algorithm for information gathering in trees

We give our information gathering algorithm for trees in Figure 16.

Algorithm TIG **(Tree Information Gathering)**
Use algorithm I^{FTIG}.

Figure 16: Algorithm TIG.

From Subsection A.4, transformation I increases the competitiveness by $\frac{B}{B-d+1}$. Thus for size of buffers $B \geq (1+\epsilon)(d-1)$, it increases the competitiveness by at most $\frac{1+\epsilon}{\epsilon}$. Since the competitiveness of algorithm $FTIG$ is $O(\log n)$, algorithm TIG is $O(\log n)$ competitive, for $B \geq (1+\epsilon)(d-1)$.