Optimal Node Routing

Yossi Azar * Yoel Chaiutin †

Abstract

We study route selection for packet switching in the competitive throughput model. In contrast to previous papers which considered competitive algorithms for packet scheduling, we consider the packet routing problem (output port selection in a node). We model the node routing problem as follows: a node has an arbitrary number of input ports and an arbitrary number of output queues. At each time unit, an arbitrary number of new packets may arrive, each packet is associated with a subset of the output ports (which correspond to the next edges on the allowed paths for the packet). Each output queue transmits packets in some arbitrary manner. Arrival and transmission are arbitrary and controlled by an adversary. The node routing algorithm has to route each packet to one of the allowed output ports, without exceeding the size of the queues. The goal is to maximize the number of the transmitted packets. In this paper, we show that all non-refusal algorithms are 2-competitive. Our main result is an almost optimal $\frac{e}{e-1} \approx 1.58$-competitive algorithm, for a large enough queue size. For packets with arbitrary values (allowing preemption) we present a 2-competitive algorithm for any queue size.

1 Introduction

Overview: A general network consists of nodes (routers) and communication links through which packets of information flow. Generally, a node consists of input ports, a switching module and an output buffer connected to each output port. A packet is received at an input port and then forwarded through the switching module to an appropriate output buffer. If the output buffer is full, the switching module must drop some packets. Most of previous works for competitive packet routing/switching considered the packet scheduling problem in the buffers assuming the path of the packet through the network is fixed and known. Moreover, to the best of our knowledge there are no results for the routing problem (i.e. path or output selection) in the competitive throughput model. In this paper we consider the simplest packet routing problem which is choosing a route in a node.

Traditionally, similar problems were analyzed while assuming either a specific distribution of the arrival rates (see [15, 22]), or some predefined structure of the sequence of arriving packets such as in the Adversarial Queueing Theory (AQT), in which the adversary injects packets that obey some capacity constraints, so that packet dropping is not necessary. It is interesting to note that the first papers on AQT assumed fixed paths [5, 16, 17, 23] and only later papers began considering the path selection problem [1, 6, 8, 9, 17]. Recently, throughput problems for various types of switches, in special graphs and arbitrary graphs were studied while avoiding any a-priori assumption on the input. As already mentioned, all these papers considered the packet scheduling but not the packet routing or output port selection. Here we consider the simplest routing (path

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

†yoel1717@yahoo.com. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel.

selection or output port selection) which is node routing. Packets arrive at the input ports, while each packet is associated with a subset of the output ports. The packet has to be routed into one of these output ports.

We model the problem of node routing as follows: a node has an arbitrary number of input ports and an arbitrary number of output queues (denoted by m). All the output queues are of size B. At each time unit, an arbitrary number of new packets may arrive, each packet is associated with a subset of the output ports (which corresponds to the next edges on the allowed paths for the packet). Each output queue transmits packets in some arbitrary manner. Arrival and transmission are arbitrary and controlled by an adversary. In contrast to the models where the path is fixed, and hence each packet needs to be routed into a unique output queue, in our model the output queue needs to be selected. In particular the main decision problem is into which output queue to send the packet (among the allowed destination output queues). If the buffers are full, the packet must be rejected. The goal of the routing algorithm is to maximize the number of the transmitted packets. We also consider the model of packets with arbitrary values. In this model we allow preemption.

We use competitive analysis to evaluate the performance of online algorithms compared to the optimal offline algorithm that knows the entire sequence in advance. Particulary, the competitive ratio of an online algorithm is the supremum, taken all over finite sequences, of the ratio between the online throughput and the optimal throughput on the same input.

Our results:

- We show that all non-refusal algorithms are 2-competitive for any queue size, and are optimal for queues of size 1. Our main result is an optimal deterministic $\frac{e}{e-1} \approx 1.58$-competitive algorithm for node routing, for a large enough size of the queues. This is done by designing an optimal $\frac{e}{e-1}$-competitive *fractional* algorithm and transforming it into a discrete algorithm with a competitive ratio of $\frac{e}{e-1}(1+\frac{2m+1}{B})$. We show that our algorithm is almost optimal by providing a lower bound of $\frac{e}{e-1} - \Theta(\frac{1}{m})$. We also show an optimal $\frac{e}{e-1} - o(1)$-competitive randomized algorithm for any B.
- Actually the discretization is more general. In fact, we present a generic technique for transforming any *fractional* algorithm for the node routing problem to a discrete algorithm, using the vector rounding algorithm of [3]. More specifically, given any c-competitive online fractional algorithm for the node routing problem, we construct an online discrete algorithm with a competitive ratio of $c(1+\frac{2m+1}{B})$.
- For the preemptive model with arbitrary values, we present a 2-competitive algorithm for any m and B. This is done using the zero-one principle presented in [13]. The algorithm is optimal for $B = 1$.

Side results: We also consider a variant of our problem (described later) where the algorithm needs to perform packet routing, admission control and packet scheduling. We show a 3-competitive algorithm for this problem, once again using the zero-one principle of [13]. We provide a lower bound of $3 - \Theta(\frac{1}{m})$ for queues of size 1.

Our techniques: We start by constructing an online reduction from the fractional node routing problem to the problem of finding a maximum fractional matching in a bipartite graph. Thus, using algorithm WL from [10] which is $\frac{e}{e-1}$-competitive, we obtain an $\frac{e}{e-1}$-competitive fractional algorithm. Then we present a generic technique for transforming any *fractional* algorithm for the node routing problem to a discrete algorithm. Specifically, we present an algorithm with larger queues and no packet loss that maintains a running simulation of the fractional algorithm. This is done using the vector rounding algorithm of [3]. We then transform this algorithm to an algorithm with queues of size B. For the model of packets with arbitrary values, we obtain the upper bound by using the zero-one principle presented in [13].

We use the following results and techniques from previous papers:

- **Vector rounding:** The n-dimensional vector rounding problem is defined as follows (the definition is taken from [3]): the input is a list of vectors $(V_1, V_2, ...)$ arriving online, where each $V_t = (v_t^1, ..., v_t^n)$ is a vector of length n over the reals which suffices $\sum_{i=1}^n v_t^i \in Z^+$. The output is a list of integer vectors $(Z_1, Z_2, ...)$ where $Z_t = (z_t^1, ..., z_t^n)$ is a rounding of V_t that preserves the sum, i.e., for all $1 \le i \le n$ we have that $z_t^i \in \{\lfloor v_t^i \rfloor, \lceil v_t^i \rceil\}$ and that $\sum_{i=1}^n z_t^i = \sum_{i=1}^n v_t^i$. The goal is to make the accumulated difference in each entry as small as possible, i.e., for every t we want $max_{1 \le i \le n} | \sum_{j=1}^t z_j^i - \sum_{j=1}^t v_j^i |$ to be as small as possible. In [3], the authors show an algorithm for the problem. They prove that the cost of their algorithm is at most n, i.e., for every t, $max_{1 \le i \le n} | \sum_{j=1}^t z_j^i - \sum_{j=1}^t v_j^i | \le n$.

- **Online matching:** The online unweighted matching problem (as appears in [10]) consists of a bipartite graph where vertices of one side arrive one by one and should be matched to one of its adjacent vertices. The goal is to maximize the number of matched vertices. It is easy to show that the greedy algorithm is 2-competitive. Karp *et al.* [19] introduced a randomized algorithm $RANKING$ with a competitive ratio of $\frac{e}{e-1} - o(1)$ and proved a lower bound of $\frac{e}{e-1} - o(1)$, thus proving the optimality of $RANKING$, up to lower order terms. The fractional model of the online unweighted matching problem was studied by Azar and Litichevsky in [10]. They showed an online algorithm *Water Level (WL)* with competitiveness of $\frac{e}{e-1}$. They also proved a lower bound of $\frac{e}{e-1} - \Theta(\frac{1}{m})$ for any deterministic algorithm, thus obtained the optimality of WL.

- **The zero-one principle:** Azar and Richter [13] presented the zero-one principle. They showed that for every comparison based switching algorithm, A is c-competitive if and only if A is c-competitive for all packet sequences whose values are restricted to 0/1, with arbitrary tie breaking.

Related results:

- **Packet switching:** Competitive analysis was used for single-queue, multi-queue and $CIOQ$ (Combined Input and Output Queue - a $m * m$ switch) switches. In these models each packet has a value and arrives to a specific queue. All packets are destined to the output port. The algorithm has to perform an admission control on the packets arrival and a packet scheduling at the transmission time, i.e., at transmission, the algorithm selects one queue and transmits one packet from the head of the queue. The goal is to maximize the total values of the transmitted packets. We emphasize that no routing decisions have to be made. In the single-queue model, the greedy algorithm is 2-competitive, as shown in [21], and a 1.75-competitive algorithm was presented in [14]. For the more general multi-queue model, there are various results: for unit value packets, the greedy algorithm is 2-competitive and a more sophisticated 1.89-competitive algorithm was presented in [4]. For large queues, a 1.58-competitive algorithm can be achieved [10]. For arbitrary values, the best known algorithm was presented in [13] with competitiveness of 3. For the more realistic $CIOQ$ switches, an 8-competitive algorithm for maximizing the total weighted throughput is presented in [12].

- **Packet switching in various graph types:** For line topology, Aiello *et al.*[2] presented algorithm Nearest To Go (NTG) which is $O(n^{\frac{2}{3}})$-competitive for unit value packets. Azar *et al.* [13] showed that the greedy algorithm is $(n+1)$-competitive for arbitrary value packets. A lower bound of $\Omega(\sqrt{n})$ on the competitiveness of the greedy algorithm for unit value packets was proved by Aiello *et al.*[2]. For directed trees, Kesselman *et al.* [20] investigated the routing *work conserving* algorithms where packets arrive to the leaves and are destined to the root. For general graphs, Awerbuch *et al.* [7] presented a load balance algorithm for packet routing. Their algorithm is $\frac{1}{1-\epsilon}$-competitive using buffers which are larger than those of the optimal solution by a factor of up to $O(\frac{n^2}{\epsilon})$ but at least $O(\frac{n}{\epsilon})$. In [2], Aiello *et al.* proved that algorithm NTG (Nearest To Go) is $O(md)$-competitive for any topology, 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 an $O(md)$ competitiveness for every greedy algorithm on DAGs. We emphasize that in all these papers, the paths of the packets through the graph are fixed and known, there are no results for the path selection problem.

Paper structure: Section 2 includes formal definitions and notations. In section 3 we introduce an optimal algorithm for the node routing problem. We show lower bounds for the problem in section 4. In section 5 we provide an algorithm for the arbitrary-value node routing problem. We address the multi queue switch routing problem in appendix A, where we present an algorithm for the problem and a lower bound. All omitted proofs appear in appendix B.

2 Problem Definition and Notations

We start with our main model.

The node routing model: We model the problem as follows: a node has an arbitrary number of input ports and m output queues $\{Q_1, ..., Q_m\}$ each of size B. At each time unit, an arbitrary number of new packets may arrive. Each packet p is associated with a subset of the output queues $Q^p \subseteq \{Q_1, ..., Q_m\}$ (which corresponds to the next edges on the allowed paths for the packet). All packets are of equal size and value. W.l.o.g. we assume that all m queues are empty at the beginning. Each time unit is divided into two phases: in the *arrival* phase a set of packets arrive, each packet p is associated with a specific subset of the output queues Q^p. For each packet p, the main decision problem is into which output queue to send it (among the allowed destination output queues Q^p). Each packet might be rejected at its arrival. If the queues Q^p are full, the packet must be rejected. Clearly, in this model there is no need for preemption since the preference of one packet over the other is pointless. At the *transmission* phase, a subset of the queues (could be also none or all the queues) is selected by the system, and a packet is transmitted from each of the heads of these queues (only from the non empty queues). Both arrival and transmission are controlled by an adversary, but we assume no starvation for each of the output queues, i.e., eventually all queues will transmit and become empty. The goal of the algorithm is to maximize the number of the transmitted packets.

We use the term *non-refusal algorithm* for algorithms that accept every packet which the queues have space for. Obviously, every algorithm can be transformed into a non-refusal algorithm without worsening its competitiveness.

The model above is the *synchronous model* in which all the packets of a single arrival phase arrive together (and all the transmissions in the transmission phase are made together), so the algorithm has full knowledge about the arrival phase before it has to make its routing decisions. We also consider the *event driven* model where arrivals and transmissions occur in arbitrary times. We denote by σ the sequence events. Each event in σ can be either an arrival event or a transmission event. The algorithm must respond after every arrival event and route the packet (or discard it). Note that the event driven model is stronger than the synchronous one. Nevertheless our upper bounds hold even for the event driven model, and the lower bounds hold even for the synchronous model (except for the arbitrary B lower bound).

We also consider the *arbitrary-value* model where each packet is associated with a non-negative value. The goal is to maximize the sum of values of the transmitted packets. In this model we allow preemption. A preempted packet is a packet which is dropped after residing in a queue. Our results hold for both FIFO and non FIFO queues (e.g. priority queues). Note that the throughput of the optimal algorithm is the same for FIFO and non FIFO models.

The multi queue switch routing model: We also consider a variant of our problem where the algorithm needs to perform packet routing, admission control and packet scheduling. In this model the output queues are parallel queues which are connected to a single output port. In a transmission event the algorithm has to choose *one* of the output queues to transmit. In this model, the transmission events in σ do not contain the transmitting queues. Here we consider only the arbitrary-value model, with preemption.

Notations: We use the terms insert, accept and assign for packet insertion into one of the queues. For packet rejection we use the terms drop and discard. We use the event driven model throughout the paper, except for the lower bound sections. We denote by *event* t the t-th event, which can be either an arrival or a transmission event. We use the term *arrival event* j to denote the j-th arrival event, i.e., the arrival of the j-th packet. Similarly we denote by *transmission event* k the k-th transmission event. For event t, we use the term *load* of queue Q_i to refer to the number of packets residing in that queue, at that time. For a given event t, the term *space* of queue Q_i is used to refer to its size minus its load, i.e., how many additional packets can be assigned to queue Q_i, at that event. Given an online routing algorithm A we denote by $A(\sigma)$ the value of A given the sequence σ, i.e., the total amount of transmitted packets after all the packets of the sequence have arrived and all the queues have been emptied. We denote the optimal (offline) algorithm by Opt. A deterministic online algorithm A is *c-competitive* for all maximization problems described in this paper iff for every packet sequence σ we have: $Opt(\sigma) \leq c \cdot A(\sigma)$.

3 Optimal Algorithm for Node Routing

First we discuss non-refusal algorithms. We note that every non-refusal algorithm is 2-competitive, for every $B \geq 1$ (this is optimal for $B = 1$, as shown in section 4). The general non-refusal algorithm is defined as follows:

Non-Refusal Algorithm: For each incoming packet p: insert p into any queue $Q_i \in Q^p$ which is not full. If such a queue does not exist, discard p.

It can be proved directly that every non-refusal algorithm is 2-competitive, but it also follows from Remark 3.1 or Theorem 5.4.

Next we describe our optimal algorithm which is $\frac{e}{e-1}(1+\frac{2m+1}{B})$-competitive (a lower bound of $\frac{e}{e-1} - \Theta(\frac{1}{m})$ is shown in section 4). Clearly, the competitive ratio of the algorithm asymptotically approaches $\frac{e}{e-1} \approx 1.58$ for large size queues. In subsection 3.1 we present algorithm FR for the fractional model, which is $\frac{e}{e-1}$-competitive. In subsection 3.2, we present a generic technique to transform any c-competitive online fractional algorithm for our problem into a discrete algorithm with a competitive ratio of $c(1+\frac{2m+1}{B})$. Specifically, we will take the fractional algorithm FR, and transform it into a $\frac{e}{e-1}(1+\frac{2m+1}{B})$-competitive discrete algorithm. We begin with the fractional routing algorithm in the following subsection.

3.1 Algorithm for the fractional version of node routing

In this subsection we consider the fractional model, which is a relaxation of the discrete model that was presented in section 2. In the fractional model we allow the online algorithm to accept fractional packets into the queues as well as to transmit fractional packets. More formally, the model is defined as follows: we have a sequence of events, each event can be either an arrival of packet p or a transmission from a queue. At arrival event t, one packet p associated with a subset of the output queues $Q^p \subseteq \{Q_1, ..., Q_m\}$ arrives. The algorithm may split the packet into fractions and insert them into the queues Q^p (only into queues with sufficient free space), and may also discard any of the fractions. The value of each fraction is equal to its size. We note by k_i^t the total amount of fractions of p inserted into queue Q_i, at arrival event t. The total amount of fractions inserted into all the queues, must not exceed the unit value, i.e., $\sum_{i=1}^{m} k_i^t \leq 1$. In addition, $Q_i \notin Q^p$ implies that $k_i^t = 0$. The second type of event is the transmission event, where some specific queue transmits a full packet from the head of that queue, if there are enough fractions in the queue. If the total fractions in the queue are less then a unit, they will all be transmitted. We assume that sequence σ consists of *integral* packets. In this subsection we show a fractional algorithm for the problem with a competitive ratio of $\frac{e}{e-1} \approx 1.58$, even against an optimal algorithm which is allowed to split incoming packets. We begin by introducing the problem of an online unweighted fractional matching in a bipartite graph. Then we introduce a translation of our problem (the fractional model) into the problem of the online unweighted fractional matching in a bipartite graph.

The online unweighted matching in a bipartite graph problem is defined as follows (as appears in [10]): first, consider an online version of the maximum bipartite matching on a graph $G = (S, R, E)$, where S and R are the disjoint vertex sets and E is the set of edges. We refer to sets S and R as the servers and the requests, respectively. The objective is to match a request to a server. At step t, a vertex $r_t \in R$ along with all of its incident edges, arrives online. Algorithm A can either reject r_t, or irreversibly match it to an unmatched vertex $s_i \in S$ adjacent to r_t. The goal of the online algorithm is to maximize the size of the matching.

The fractional version of the online unweighted matching is as follows: each request r_t has a size x_t which is the amount of work needed to service it. Algorithm A can match a fraction of size $k_i^t \in [0, x_t]$ to each vertex $s_i \in S$ adjacent to r_t. If request t is matched partially to some server i with weight k_i^t then $\sum_{i=1}^{|S|} k_i^t \leq x_t$. But the load of each server i, which is $\sum_{t=1}^{n} k_i^t$ where n is the length of σ, must be at most 1. We use the terms *match* and *assign* interchangeably. The goal of the online algorithm is to maximize the sum of matched fractions, i.e., to maximize $\sum_{t,i} k_i^t$.

Our translation of the fractional routing problem into the problem of online unweighted fractional matching in a bipartite graph, opposes, in some sense, the translations in [10, 11]. In our translation, the incoming packets are the requests and they are matched to the queues which are the servers, while in [10, 11] the requests are the transmission events and they are matched to the packets, which are the servers.

Given a sequence σ, we translate it into the bipartite graph $G^\sigma = (R, S, E)$, which is defined as follows:

- Let T denote the total number of packets. We define the set of requests as $R = \{r_1, ..., r_T\}$ all with unit sizes, i.e., $x_i = 1$ for each $1 \leq i \leq T$. Each request corresponds to a packet.
- For each queue Q_i we define a set of servers S_i, which represents the queue over time. Specifically, each S_i ($1 \leq i \leq m$) contains $T + B$ servers: $S_i = \{s_i^1, ..., s_i^{T+B}\}$. The S_i's are disjoint and the full set of servers is defined as $S = \bigcup_{i=1}^{m} S_i$.
- Let y_i^t denote the number of times queue Q_i was selected for transmission until *arrival event* t. We denote by S_i^t the B servers in S_i that represent queue Q_i at arrival event t. We define $S_i^t = \{s_i^{y_i^t+1}, ..., s_i^{y_i^t+B}\} \subseteq S_i$. Consider packet p arriving at arrival event t, associated with the subset Q^p. In the bipartite graph problem, at step t a request r_t arrives along with its incident edges, which are the edges connecting it to all servers in S_i^t for each i such that $Q_i \in Q^p$. More formally, r_t arrives with the following incident edges: $\{(s, r_t) | s \in S_i^t, Q_i \in Q^p\}$.

Definition 3.1 *A* route *RT for a sequence of arriving packets σ, for the fractional node routing problem, is a set of triplets of the form (t, Q_i, k_i^t) for each $1 \leq i \leq m$ and $1 \leq t \leq T$. Each triplet expresses that at arrival event t, queue Q_i gets the fraction $k_i^t \geq 0$ of the packet p. The* size of the route*, denoted by $|RT|$, is the total amount of fractions transmitted. Since all accepted fractions of packets are transmitted, clearly $|RT| = \sum_{t,i} k_i^t$.*

Definition 3.2 *A route RT for a sequence σ is called* legal *if for every triplet (t, Q_i, k_i^t), queue Q_i has free space of at least k_i^t at arrival event t (the empty space is a function of accepted fractions and transmissions which are in σ) and $\sum_{i=1}^{m} k_i^t \leq 1$, and $Q_i \notin Q^p$ implies that $k_i^t = 0$.*

The following lemmas connect bipartite fractional matching to our problem.

Lemma 3.1 *Every legal fractional route RT for the sequence σ can be mapped, in an online fashion, to a fractional matching M in G^σ such that $|RT| = |M|$.*

Lemma 3.2 *Every fractional matching M in G^σ can be translated, in an online fashion in polynomial time, to a legal fractional route RT for σ such that $|RT| = |M|$.*

The following corollary result directly from Lemmas 3.1 and 3.2.

Corollary 3.3 *For any sequence σ, the size of the optimal fractional route for σ is equal to the size of a maximum fractional matching in G^σ.*

Remark 3.1 *Actually, Lemmas 3.1 and 3.2 hold also for integral node routing and integral online matching. This implies that every non-refusal algorithm for node routing is 2-competitive. Note that every non-refusal node routing algorithm corresponds to a non-refusal online matching algorithm, which yields a maximal matching. Since every maximal matching is at least half of the maximum matching, we conclude that every non-refusal algorithm for node routing is 2-competitive.*

Remark 3.2 *We note that by using the above reduction from the integral node routing to the integral online matching, and applying the randomized algorithm $RANKING$ of Karp* et al. *[19], we obtain a randomized $\frac{e}{e-1} - o(1)$-competitive algorithm. Our main focus is on a deterministic algorithm.*

We use algorithm WL for the unweighted fractional matching problem, presented in [10], and its results.

Algorithm WL: For each request r_t adjacent to servers $\{s_1, ..., s_n\}$, match a fraction of size k_j^t for each adjacent s_j, where $k_j^t = (h - l_j)_+$ and $h \leq 1$ is the maximum number such that $\sum_{j=1}^{n} k_j^t \leq 1$. By $(f)_+$ we mean $\max\{f, 0\}$.

Theorem 3.4 [10] *Algorithm WL is $\frac{e}{e-1}$-competitive and optimal for the bipartite unweighted fractional matching problem.*

Now we present a fractional routing algorithm FR for the fractional node routing problem. Since the bipartite unweighted fractional matching problem is connected to the fractional node routing problem, we use algorithm WL. Algorithm FR intuitively bases its decisions on the matching constructed by algorithm WL, in the online constructed graph G^σ.

Algorithm Fractional Routing (FR):

- Maintain a running simulation of WL in the online constructed graph G^σ.
- **Routing**: For arrival event t, translate the matching of r_t to servers made by WL, using Lemma 3.2, to legal routing triplets (t, Q_i, k_i^t). Recall that each triplet corresponds to assigning a k_i^t fraction of p to queue Q_i.

Note that FR is not a non-refusal algorithm (because in G^σ, the total load of the servers in S_i^t might be bigger than the load of queue Q_i at arrival event t).

Theorem 3.5 *For every sequence σ, $Opt(\sigma) \leq \frac{e}{e-1} FR(\sigma)$ for the fractional node routing.*

3.2 The discretization of the fractional algorithm

In this subsection we introduce a generic technique for the discretization of any fractional algorithm for the node routing problem. In particular, we will use this technique for the discretization of algorithm FR, which was presented in subsection 3.1. We present in subsection 3.2.1 a general technique for the discretization of the routing decisions of any fractional algorithm A, for the *unbounded queues* version of the problem. We will use this technique in subsection 3.2.2 for the discretization of the routing of any fractional algorithm A, for our version of the node routing problem (bounded queues).

3.2.1 The unbounded queues node routing problem

Consider the node routing problem, but assume that the queues are unbounded. Thus all packets are accepted into the queues. In this case, we may consider the minimization of the maximum queue size (notice that it is a *cost* problem). More formally, we are given a node with m output queues, where queues have an unbounded size. At the beginning, all m queues are empty. We denote the online finite event sequence by σ. Each event is either an arrival event or a transmission event. In an arrival event one packet p arrives to the queues, associated with some subset of the queues Q^p. Packet p *must* be inserted by the algorithm into one of the queues Q^p. At a transmission event, the adversary selects one of the queues which in turn transmits the packet at the head of that queue (if such exists). We define the cost of the online algorithm A (denoted by $cost(A)$), given a finite sequence σ, as the maximum length of a queue taken over all queues and times. The goal is to minimize the maximum cost, i.e., minimize the maximum queue size over time.

We now turn to consider a relaxation of the model and allow an online algorithm to split packets and assign fractions of packets to the queues, provided that for every packet p, the total size of the accepted fractions of p into queues Q^p is exactly 1, i.e., the algorithm is not allowed to discard any fraction of the packet. We denote by A the online fractional algorithm. We now introduce a general technique for the discretization of the routing decisions of algorithm A, given a finite sequence of *integral* packets σ while adding an additive cost of at most $2m$ to the cost of A.

Now, we define a discrete algorithm M which gets the fractional algorithm A as a parameter and denote it by M^A. At each packet arrival, in order to decide which queue to serve, M^A computes the load seen by A and uses this information for making its decisions. Given arrival event t, let l_i^A and l_i be the simulated load of A and the actual load of M^A on queue Q_i at that event, respectively.

For the decisions of M^A we use a theorem for the vector rounding problem which applies to our problem. We address the vector rounding problem as presented in [3][1]. The m-dimensional vector rounding problem is this: the input is a list of vectors $(V_1, V_2, ...)$ arriving online, where each $V_t = (v_t^1, ..., v_t^m)$ is a vector of length m over the reals which suffices $\sum_{i=1}^m v_t^i \in Z^+$. The output is a list of integer vectors $(Z_1, Z_2, ...)$ where $Z_t = (z_t^1, ..., z_t^m)$ is a rounding of V_t that preserves the sum, i.e., for all $1 \le i \le m$ we have that $z_t^i \in \{\lfloor v_t^i \rfloor, \lceil v_t^i \rceil\}$ and that $\sum_{i=1}^m z_t^i = \sum_{i=1}^m v_t^i$. The goal is to make the accumulated difference in each entry as small as possible, i.e., for every t we want $max_{1 \le i \le m} | \sum_{j=1}^t z_j^i - \sum_{j=1}^t v_j^i |$ to be as small as possible. The cost of the algorithm is the unfairness, which is the maximum accumulated difference in each entry over all time units. The goal is to minimize the unfairness.

The major special case we will be interested in is when vectors V_t satisfy $0 \le v_t^i \le 1$ and $\sum_{i=1}^m v_t^i = 1$. Since the output vector Z_t satisfies $z_t^i \in \{\lfloor v_t^i \rfloor, \lceil v_t^i \rceil\}$ and $\sum_{i=1}^m z_t^i = \sum_{i=1}^m v_t^i = 1$, we conclude that Z_t will be all zeros except for one entry $z_t^{i'} = 1$ where i' satisfies $v_t^{i'} > 0$.

In [3], the authors show a way to build an algorithm (we call it VR) for the online vector rounding problem. Algorithm VR is based on another algorithm which is a greedy algorithm for a degenerate

[1]The vector rounding problem is a generalization of the car pool problem.

online vector rounding problem (also called 2-person carpool problem): each vector V_t consists of zeros, except for two entries: $v^{i_1} = v^{i_2} = 1/2$, $i_1 \neq i_2$.

Theorem 3.6 [3] *For the vector rounding problem, algorithm VR's unfairness is at most m, i.e., for every t, $max_{1\leq i\leq m}|\sum_{j=1}^{t} z_j^i - \sum_{j=1}^{t} v_j^i| \leq m$.*

Now we present the discrete algorithm M^A.

Algorithm M^A:

Maintain a running online simulation of algorithm A. For each arrival event t of a packet p let k_i^t denote the fractions inserted into the queue Q_i by A, i.e., $\sum_{Q_i \in Q^p} k_i^t = 1$, and $Q_i \notin Q^p$ implies that $k_i^t = 0$. For each arrival event t of a packet p do:

- Build an input vector V_t for the vector rounding problem $V_t = (v_t^1, ..., v_t^m)$ where $v_t^i = k_i^t$.
- Get the output vector $Z_t = (z_t^1, ..., z_t^m)$ from the simulated algorithm VR given the input vectors $(V_1, ..., V_t)$.
- Insert p into queue Q_i which satisfies $z_t^i = 1$.

Note that our algorithm is not affected by the transmissions. We also note that the insertion of p is legal, because all the input vectors of VR are of the form $V_t = (v_t^1, ..., v_t^m) = (k_1^t, ..., k_m^t)$ and satisfy $0 \leq k_i^t \leq 1$ and $\sum_{i=1}^{m} k_i^t = 1$, and as mentioned, this implies that Z_t consists of zeros except for one entry $z_t^{i'} = 1$, where i' satisfies $k_{i'}^t > 0$, this implies that $Q_{i'} \in Q^p$.

Now we introduce our main theorem for this subsection.

Theorem 3.7 *For every sequence σ, $cost(M^A) \leq cost(A) + 2m$. Alternatively, for sequences in which A needs queues of size B, M^A needs queues of size $B + 2m$.*

3.2.2 The discretization of the fractional algorithm

In this subsection we return to the standard node routing model (with bounded queues). We show a general technique to transform any c-competitive *fractional* algorithm for the node routing problem into a discrete algorithm with a competitive ratio of $c(1 + \frac{2m+1}{B})$. Specifically, we will apply this technique for the discretization of algorithm FR, which was presented in subsection 3.1.

For our discretization process we rely on the results of the unbounded problem, which were presented in subsection 3.2.1. We want to address the packets which were accepted by FR as the input sequence σ for the cost problem studied in subsection 3.2.1, in a way which we will present later. Recall that algorithm FR might accept fractional packets due to insufficient queue space. Since in the model of the unbounded problem we study the case where σ consists of integral packets, we want FR to only accept packets integrally, i.e., for every packet p, to accept k_i^t such that $\sum_{i=1}^{m} k_i^t = 1$. Hence, we continue by considering the following problem: assume we are given an online c-competitive algorithm A. We want to produce a competitive algorithm $\hat{A}$ which assigns only integral packets. We provide algorithm $\hat{A}$ which has queues of size $B+1$ (algorithm A maintains queues of size B); we will get rid of this assumption later on. Intuitively, $\hat{A}$ emulates the routing of A and accepts only integral packets.

Definition 3.3 *We denote fractional algorithms that accept integral packets whenever there is sufficient space for a whole packet, and otherwise discard the whole packet, as* discrete non-refusal *algorithms.*

We now define the transformation of a given algorithm A with queues of size B into algorithm $\hat{A}$ with queues of size $B+1$, which is a discrete non-refusal algorithm. Let l_i^t and $\hat{l}_i^t$ be the loads of queue Q_i at event t in A and $\hat{A}$, respectively.

Algorithm $\hat{A}$:

- Maintain a running simulation of A. Let k_i^t be the fraction size which was inserted by algorithm A at arrival event t into queue Q_i.
- If the queues in Q^p do not have sufficient space (the total free space is less than a unit), discard p.
- Otherwise, assign to queue Q_i amount of $min(B+1-\hat{l}_i, k_i^t)$, i.e., assign k_i^t if there is enough space or just fill the queue. After assigning to all the queues in Q^p, if the total inserted volume is less than a unit, insert the rest of p arbitrarily into any subset of the queues in Q^p which has sufficient empty space.

Obviously, $\hat{A}$ is a discrete non-refusal algorithm. Now we prove that $A(\sigma) \leq \hat{A}(\sigma)$ for every σ.

Theorem 3.8 *For a given algorithm A with queues of size B, algorithm $\hat{A}$ with queues of size $B+1$ has at least the same throughput as A, given the same sequence σ.*

Now, we consider algorithm M presented in subsection 3.2.1. Recall that M simulates some fractional algorithm which fully accepts every incoming packet (since the queues are unbounded). We want to use algorithm M on algorithm $\hat{A}$ (denoted by $M^{\hat{A}}$) which is a discrete non-refusal algorithm. Since $\hat{A}$ has queues of size $B+1$, algorithm $M^{\hat{A}}$ will use queues of size $B+1+2m$. Still, we cannot use M on $\hat{A}$ since $\hat{A}$ rejects packets when there is no sufficient free space in queues Q^p. Therefore we extend algorithm M to work on algorithms that reject whole packets. This is done by skipping the events in which packets are rejected. We now present the following lemma:

Lemma 3.9 *Algorithm $M^{\hat{A}}$ with queues of size $B+1+2m$ has the same throughput as algorithm $\hat{A}$ with queues of size $B+1$, given the same sequence σ.*

We now return to our original model where queues are of size B. By Theorem 3.8 and Lemma 3.9, if $M^{\hat{A}}$ had queues of size $B+1+2m$, then algorithm $M^{\hat{A}}$ would have had at least the same throughput as A. Unfortunately, this is not the case, so we continue by emulating an algorithm with large queues with an algorithm with small queues.

We use the emulation presented in [10]: assume we are given an online competitive discrete algorithm A with queues of size y and we want to produce a competitive algorithm E^A with queues of size $y' < y$. We present algorithm E^A, and the corresponding theorem from [10].

Algorithm E^A: Maintain a running simulation of algorithm A. Accept a packet into queue Q_i if A accepts it to queue Q_i and the queue is not full.

Theorem 3.10 [10] *Given two switching algorithms A and B with queue sizes y and y' respectively, and $y > y'$. If B accepts the same packets to the same queues as A when it has sufficient free space (otherwise it discards the packets), and B transmits from the same queues as A when they are not empty (otherwise B does not transmit) then $A(\sigma) \leq \frac{y}{y'}B(\sigma)$.*

Corollary 3.11 *Obviously, by applying Theorem 3.10, we conclude that $A(\sigma) \leq \frac{y}{y'}E^A(\sigma)$.*

We prove the main result of this section with the next theorem.

Theorem 3.12 *Given any c-competitive fractional algorithm A for the node routing problem, algorithm $E^{M^{\hat{A}}}$ is a $c(1 + \frac{2m+1}{B})$-competitive discrete algorithm for the node routing problem.*

Corollary 3.13 *Applying Theorem 3.12 on algorithm FR, produces algorithm $E^{M^{\hat{FR}}}$ which is $\frac{e}{e-1}(1 + \frac{2m+1}{B})$-competitive. For $B \gg m$ the competitive ratio of $E^{M^{\hat{FR}}}$ approaches $\frac{e}{e-1} \approx 1.58$.*

4 Lower Bounds

In this section we show some lower bounds for the node routing problem. We first show that an $\frac{e}{e-1}$-competitive algorithm for node routing is optimal. The lower bound holds even for randomized algorithms.

Theorem 4.1 *Every algorithm for the node routing problem is at least $\frac{e}{e-1} - \Theta(\frac{1}{m})$-competitive, for every B.*

Now we show an easy proof for lower bound of 2 for deterministic algorithms, even for the synchronous model, using queues of size 1. Note that w.l.o.g. we may assume that every algorithm is a non-refusal algorithm.

Theorem 4.2 *Every algorithm for the node routing problem is at least 2-competitive, for $B = 1$ and any $m \geq 2$.*

5 Algorithm for arbitrary-value node routing

In this section we consider the *arbitrary-value* node routing problem. We introduce algorithm *greedy routing* (GR) which is 2-competitive, for the problem.

Algorithm Greedy Routing (GR):

When packet p associated with the subset Q^p arrives:

- If there exists queue Q_i ($Q_i \in Q^p$) which is not full, insert p into Q_i.
- Otherwise, we note by p' a packet with the smallest value in Q^p and by queue $Q_{i'}$ ($Q_{i'} \in Q^p$) we note its queue.
  - If the value of p' is smaller than the value of p, preempt p' and insert p into queue $Q_{i'}$.
  - Otherwise, drop p.

In order to prove that GR is 2-competitive, we use the zero-one principle from [13]. We begin by presenting the principle.

Theorem 5.1 [13] *Let A be a comparison based deterministic switching algorithm. A is a c-competitive algorithm if and only if A is c-competitive for all packet sequences whose values are restricted to 0/1, breaking ties arbitrarily.*

The term *comparison based* means that the algorithm bases its decisions solely on the relative order between values (full definition is given in [13]). Clearly, algorithm GR is comparison based.

We analyze how algorithm GR acts on 0/1 sequences: it fills the queues Q^p if possible. If the queues Q^p are all full, there are some possible cases: if a 0-packet p arrives and the queues Q^p contain some 0-packets, then since ties may be broken arbitrarily, GR may discard p or preempt one of the 0-packets in Q^p. If a 0-packet p arrives and the queues Q^p contain only 1-packets, GR must discard p. If a 1-packet p arrives and the queues Q^p contain some 0-packets, GR must preempt one of the 0-packets in Q^p. If a 1-packet p arrives and the queues Q^p contain only 1-packets, then since ties may be broken arbitrarily, GR may discard p or preempt one of the 1-packets in Q^p.

According to the zero-one principle we need only show that the algorithm is 2-competitive for 0/1 sequences. With a slight abuse of notations, let $GR(\sigma)$ denote the set of 1-packets transmitted by GR, given the input sequence σ. Similarly, let $Opt(\sigma)$ denote the set of packets transmitted by Opt. Note that we may assume that Opt accepts and transmits only 1-packets, and never preempts packets, since it is the offline optimal algorithm. We show a matching from $(Opt(\sigma) \setminus GR(\sigma))$ to $GR(\sigma)$. First we prove the following claim.

Claim 5.2 *If there exists a matching from $(Opt(\sigma) \setminus GR(\sigma))$ to $GR(\sigma)$ in which each 1-packet from $GR(\sigma)$ is matched at most once, then GR is 2-competitive for 0/1 sequences.*

The matching is done by the following marking scheme.

Marking Scheme

For each incoming 1-packet p do:

1. If p is accepted by GR, consider it as an unmarked packet. In case that a marked 1-packet p' was preempted from queue Q_i because of p, mark the first unmarked 1-packet in queue Q_i of GR, starting from the head of the queue.
2. If p is not accepted by GR but accepted by Opt (into queue $Q_{i'}$ ($Q_{i'} \in Q^p$)), mark the first unmarked 1-packet in queue $Q_{i'}$ of GR, starting from the head of the queue.

Clearly, for each 1-packet in $(Opt(\sigma) \setminus GR(\sigma))$ we mark one 1-packet in $GR(\sigma)$, and each 1-packet in $GR(\sigma)$ can be marked at most once. Obviously, if a marking is needed in step 1, there exists at least one unmarked 1-packet (e.g. the new packet p). For the marking to be valid we have to show that in step 2, an unmarked 1-packet always exists. Now we prove the marking validity.

Lemma 5.3 *The marking scheme is valid, hence there is a matching from $(Opt(\sigma) \setminus GR(\sigma))$ to $GR(\sigma)$ in which each 1-packet from $GR(\sigma)$ is matched at most once.*

Now we present the main theorem of this section:

Theorem 5.4 *Algorithm GR is 2-competitive for the arbitrary-value node routing problem.*

Corollary 5.5 *For the unit-value model, algorithm GR includes all non-refusal algorithms, hence every non-refusal algorithm is 2-competitive in the unit-value model.*

References

[1] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. 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] Miklos Ajtai, James Aspnes, Moni Naor, Yuval Rabani, Leonard J. Schulman, and Orli Waarts. Fairness in scheduling. *Journal of Algorithms*, 29(2):306–357, November 1998.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. In *Proc. 35th ACM Symp. on Theory of Computing*, pages 82–89, 2003.

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

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

[14] N. Bansal, L. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko. Further improvements in competitive guarantees for QoS buffering. pages 196–207, 2004.

[15] 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.

[16] 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.

[17] 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.

[18] B. Kalyanasundaram and K. R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233:319–325, 2000.

[19] R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of 22nd Annual ACM Symposium on Theory of Computing*, pages 352–358, may 1990.

[20] 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.

[21] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 520–529, 2001.

[22] 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.

[23] 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 Appendix

Multi Queue Switch Routing In this section we analyze the multi queue switch routing model, in which the algorithm has to make the packet scheduling decisions (choose a queue for transmission whenever a transmission event occurs) in addition to the routing decisions. We consider the arbitrary-value model, allowing preemption. We present algorithm *transmit from largest head* (TLH) which is 3-competitive (similar to algorithm TLH in [13]). We also show a lower bound of $3 - \Theta(\frac{1}{m})$ for $B = 1$.

A.1 Algorithm for multi queue switch routing

In this subsection we present algorithm *transmit from largest head* (TLH) which is 3-competitive. The algorithm and the proof are based on algorithm TLH in [13]. The proof in this section is similar to the proof in section 5 (but slightly more complicated), hence we use the same notations. Now we introduce algorithm TLH.

Algorithm Transmit From Largest Head (TLH):

- Routing: Use algorithm GR (from section 5).
- Scheduling: At each transmission event, transmit a packet with the largest value among all packets at the heads of the queues.

We use again the zero-one principle [13], presented in section 5. Clearly, algorithm TLH is comparison based.

We analyze how algorithm TLH acts on 0/1 sequences:

- At packet arrivals: It acts the same as GR.
- At packet transmissions: If there exist 1-packets in any of the heads of the queues, one of them will be transmitted. Otherwise, a 0-packet will be transmitted (unless all queues are empty).

According to the zero-one principle we need only show that the algorithm is 3-competitive for 0/1 sequences. We show a matching from $(Opt(\sigma) \setminus TLH(\sigma))$ to $TLH(\sigma)$, using the matching we will prove that TLH is 3-competitive. First we prove the following claim.

Claim A.1 *If there is a matching from $(Opt(\sigma) \setminus TLH(\sigma))$ to $TLH(\sigma)$ in which each 1-packet from $TLH(\sigma)$ is matched at most twice, then TLH is 3-competitive for 0/1 sequences.*

Proof: It immediately follows from the matching that:

$$|Opt(\sigma)| \leq |Opt(\sigma) \setminus TLH(\sigma)| + |TLH(\sigma)| \leq 3 \cdot |TLH(\sigma)|$$

■

The matching is done using the same marking scheme as in [13].

Marking scheme

1. For each incoming 1-packet p associated with the subset Q^p do:
   (a) If p is accepted by TLH, consider it as an unmarked packet. In case that a marked 1-packet p' was preempted from queue Q_i because of p, mark the first unmarked 1-packet in queue Q_i of TLH, starting from the head of the queue.
   (b) If p is not accepted by TLH but accepted by Opt (into queue $Q_{i'}$ $(Q_{i'} \in Q^p)$), mark the first unmarked 1-packet in queue $Q_{i'}$ of TLH, starting from the head of the queue.
2. For a transmission event, whenever Opt transmits a packet do:
   (a) If Opt and TLH use the same queue for transmission, do nothing.
   (b) Otherwise, let Q_i and $Q_{i'}$ $(i \neq i')$ be the queues used by TLH and Opt, respectively. If queue $Q_{i'}$ of TLH contains marked packets, unmark the marked 1-packet closest to the tail in queue $Q_{i'}$ and mark the packet transmitted from queue Q_i.

Clearly, for each 1-packet in $(Opt(\sigma) \setminus TLH(\sigma))$ we mark one 1-packet in $TLH(\sigma)$. Each 1-packet in $TLH(\sigma)$ can be marked at most twice, once while it resides in the queue and once while it is transmitted. Obviously, if a marking is needed in step 1a, there exists at least one unmarked 1-packet (e.g. the new packet p). For the marking to be valid, we have to prove the invariants in the next two claims.

Claim A.2 Validity of incoming marking *(marking step 1b): Whenever incoming packet p is accepted by Opt into queue $Q_{i'}$ but not accepted by TLH, queue $Q_{i'}$ contains an unmarked 1-packet.*

Claim A.3 Validity of transmission marking *(marking step 2b): Whenever TLH and Opt transmit from different queues Q_i and $Q_{i'}$, respectively, and queue $Q_{i'}$ contains marked packets, TLH transmits a 1-packet.*

In proving the validation of the marking, we ignore the insertion, rejection or exchange of 0-packets since they do not interfere with the marking scheme. In addition, we assume that Opt does not preempt 1-packets as it is an offline algorithm. Now we prove the validity of the marking:

Lemma A.4 *The marking scheme is valid, hence there exists a matching from $(Opt(\sigma) \setminus TLH(\sigma))$ to $TLH(\sigma)$ in which each 1-packet from $TLH(\sigma)$ is matched at most twice.*

Proof: All we have to prove is the two invariants: the validity of incoming marking (Claim A.2) and the validity of transmission marking (Claim A.3). We start with the validity of transmission marking (Claim A.3).

Proof: We use the same claim as in section 5:

Claim A.5 *For each event t, if the packet in the head of queue Q_i is unmarked, then all the packets in queue Q_i are unmarked.*

Proof: Same as for Claim B.6. ■

The next claim is from [13], it concludes the proof of the validity of transmission marking.

Claim A.6 *For every transmission event t', if the queues hold any marked packets, then TLH transmits a 1-packet.*

Proof: From Claim A.5 we conclude that if queue Q_i holds a marked packet then the head of the queue is marked, i.e., it is a 1-packet. Since TLH transmits the largest head, a 0-packet cannot be transmitted. ■

This completes the proof of the validity of transmission marking (Claim A.3). ■

Now we prove the validity of incoming marking (Claim A.2):

Proof: We use the same notations as in section 5. The next claim proves the validity of incoming marking and thus concludes the proof.

Claim A.7 *For each state S_j, $(1 \leq j \leq r)$, and queue Q_i, $(1 \leq i \leq m)$, $U_i^j \geq TLH_i^j - Opt_i^j$, and the incoming marking is possible if required at that state.*

Proof: First we show that if the inequality holds for state S_j, incoming marking is possible if required at that state. Consider the arrival of 1-packet $p \in (Opt(\sigma) \setminus TLH(\sigma))$ (we note by $Q_{i'}$ the queue into which Opt inserted p). Since TLH did not accept p, we conclude that all the queues in Q^p must be full of 1-packets only. So $TLH_{i'} = B$ but $Opt_{i'} < B$ because Opt had place for p in queue $Q_{i'}$. From the inequality it follows that $U_{i'} > 0$; therefore the incoming marking is valid for state S_j.

We prove the inequality by induction on the states. For the initial state S_1 the inequality clearly holds. We assume correctness for state S_j and prove that the inequality holds for state S_{j+1}. We consider two types of events:

1. Consider the arrival of a 1-packet p associated with the subset Q^p. The proof for this case is the same as the proof for the arrival event in Claim B.7 (for algorithm GR).

2. Consider the transmission events. TLH and Opt transmit p and p' from queues Q_i and $Q_{i'}$, respectively (if the queues are not empty).

   Observation A.1 *For each queue whose head is unmarked, the inequality still holds after transmission. It holds since from Claim A.5 we conclude that all the packets in the queue are unmarked. After any transmission all the packets will be still unmarked. Therefore $U_i = TLH_i$, and since $Opt_i \geq 0$ the inequality holds.*

We have four possible cases, we consider only the cases when p is marked or when queue $Q_{i'}$ contains marked packets (because of Observation A.1):

(a) All the queues in TLH are empty (only Opt transmits): $U_{i'} = 0$, $TLH_{i'} = 0$ and $Opt_{i'} \geq 0$, therefore, the inequality holds.

(b) All the queues in Opt are empty (only TLH transmits): p is marked (Observation A.1), TLH_i is decreased, so the inequality holds.

(c) $i = i'$: p is marked (Observation A.1), both TLH_i and Opt_i are decreased by 1, so the inequality holds.

(d) $i \neq i'$: We prove two inequalities:

   i. For queue Q_i: p is marked (Observation A.1), TLH_i is decreased, so the inequality holds.

   ii. For queue $Q_{i'}$: Queue $Q_{i'}$ contains marked packets (Observation A.1), $Opt_{i'}$ is decreased, but $U_{i'}$ is increased because of the marking exchange (step 2b), so the inequality holds. This does not affect the inequality of queue Q_i.

This concludes the proof of Claim A.7. ■

The incoming marking validity (Claim A.2) follows immediately from Claim A.7. ■

This completes the proof of the validity of the marking (Lemma A.4). ■

Now we present the main theorem of this subsection.

Theorem A.8 *Algorithm TLH is 3-competitive for the arbitrary-value multi queue switch routing.*

Proof: Using the marking scheme with Lemma A.4 and Claim A.1 we conclude that TLH is 3-competitive for 0/1 sequences. Then by using the zero-one principle (Theorem 5.1) we complete the proof. ■

A.2 Lower bound for multi queue switch routing

We show a lower bound for this problem when $B = 1$, even for the unit-value, synchronous model. Note that w.l.o.g. we may assume that every algorithm is non-preemptive and non-refusal.

Theorem A.9 *Every algorithm for the unit-value multi queue switch routing problem is at least $(3 - \frac{6}{m})$-competitive.*

Proof: Consider queues of size 1. Given an algorithm A for our problem. We construct the following sequence σ_A:

1. At the first arrival phase ($t = 1$), $\frac{m}{2}$ packets arrive, every packet is associated with the whole set of queues $\{Q_1, ..., Q_m\}$.

   W.l.o.g we may assume that A inserts the $\frac{m}{2}$ packets into queues $Q_1, ..., Q_{\frac{m}{2}}$, and then transmits a packet from $Q_{\frac{m}{2}}$.

2. In the next arrival phase, ($t = 2$), $\frac{m}{2} - 1$ packets arrive, every packet associated with subset of the queues $\{Q_1, ..., Q_{\frac{m}{2}-1}\}$.

3. In the next $\frac{m}{2}-2$ arrival phases (time units $t=3,...,\frac{m}{2}$), one packet arrives to one full queue in A (such a queue always exists).

4. The above sequence is repeated many times.

We now analyze the competitive ratio of A, by looking on each sequence repetition. Clearly, after step 2, algorithm A has only queues $\{Q_1,...,Q_{\frac{m}{2}-1}\}$ full. While Opt can accept all the $m-1$ packets. In step 3, A transmits all its packets, but does not gain any new packets. On the other hand Opt accepts all the new $\frac{m}{2}-2$ packets by transmitting from the queue which will receive a packet in the next time unit. Hence:

$$\frac{Opt(\sigma_A)}{A(\sigma_A)} = \frac{\frac{3m}{2}-3}{\frac{m}{2}} = 3-\frac{6}{m}$$

■

B Appendix

Proof of Lemma 3.1. Let RT be a legal route for σ. For each $1 \le i \le m$ and $1 \le t \le T$ we have $(t, Q_i, k_i^t) \in RT$. We construct the desired matching M incrementally while moving ahead in time. In step t request r_t arrives, we match a fraction of size $\min\{k_i^t, 1-l_i^j\}$ from request r_t to server s_i^j, where $j=\min\{k|s_i^k \in S_i^t, l_i^k<1\}$ and l_i^j is the load on server s_i^j. If k_i^t is not matched completely on s_i^j we match the rest of k_i^t to a new server $s_i^{j'}$, where $j'=j+1$. With simple induction, it is easy to show that for each arrival event t and queue Q_i, the sum of matched fractions of servers in S_i^t is equal to the total load of queue Q_i at arrival event t, according to RT. Hence, every fraction assigned to a queue (and thus transmitted) can be mapped to a fraction matched in M. Clearly, by the construction, $|RT|=|M|$. □

Proof of Lemma 3.2. Let M be a matching in G^σ. We construct a legal route RT for σ incrementally, while going over the requests in R, starting from r_1. The set of servers fractionally matched to request r_t in M are denoted by D_{r_t}. Obviously from the construction of G^σ, $D_{r_t} \subseteq \bigcup_{1\le i\le m} S_i^t$. We define a set $I=\{i|S_i^t \cap D_{r_t} \ne \phi\}$. For each $i \in I$ we define k_i^t to be the total sum of fractions matched from request r_t to each $s_i^j \in S_i^t$. For each $1 \le i \le m$ such that $i \notin I$ we define $k_i^t=0$. We add the triplet (t, Q_i, k_i^t) to the route RT, for each i. With simple induction, it is easy to show that for every $1 \le i \le m$ and $1 \le t \le T$, the total load of servers in S_i^t is at least the load of queue Q_i at arrival event t according to RT (this is because in transmission the load of the queue is decreased by a full packet, while in G^σ, S_i^t's load might be decreased by less than a unit). Therefore, we can always translate matched fractions in M to a fraction assigned to a queue in RT and thus it is also transmitted. Clearly by construction $\sum_{i=1}^m k_i^t \le 1$, and $Q_i \notin Q^p$ implies that $k_i^t=0$, so our obtained route is legal. Obviously, this translation takes polynomial time and, by the construction, $|RT|=|M|$. □

Proof of Theorem 3.5. The theorem follows immediately from Corollary 3.3, the construction of FR and Lemma 3.2. □

Proof of Theorem 3.7. First we define:

Definition B.1 *Given an event t, let l_i^A and l_i be the simulated load of A and the actual load of M^A on queue Q_i at that event, respectively. The* residual load *of queue Q_i at event t is defined as $l_i^{res}=l_i-l_i^A$.*

Obviously, the next lemma completes the proof.

Lemma B.1 *For each* event t *and queue* Q_i, $l_i^{res} \leq 2m$.

Proof: Consider a relaxation to our problem in which packets arrive but are never transmitted.

Claim B.2 *If there are no transmissions then* $|l_i^{res}| \leq m$.

Proof: Since there is no packet loss, $\sum_{j=1}^{t} v_j^i$ and $\sum_{j=1}^{t} z_j^i$ express the load of queue Q_i at arrival event t for algorithms A and M^A, respectively. Using Theorem 3.6 we conclude that $|l_i^{res}| \leq m$. ■

Before we proceed with our proof, we present a few definitions.

Definition B.2 *We call a transmission as* A-loss *when A transmits more than M^A, i.e., A transmits a packet (or a fraction of a packet) but M^A does not (the queue is empty in M^A). Similarly, we call a transmission as* M^A-loss *when M^A transmits more than A. Finally, we call a transmission as* regular *when both A and M^A transmit a packet.*

Observation B.1 *For each queue i, an A-loss transmission is possible only when $l_i^{res} < 0$, and a M^A-loss transmission is possible only when $l_i^{res} > 0$.*

Definition B.3 *For arrival event t, let $Trans_i^t(A)$ and $Trans_i^t(M^A)$ denote the total sum of packets (and fractions of packets) transmitted so far from queue Q_i in A and M^A, respectively.*

Definition B.4 *The* residual input *of queue i at arrival event t is defined as $RI_i^t = \sum_{j=1}^{t} z_j^i - \sum_{j=1}^{t} v_j^i$. Similarly, the* residual transmission *of queue Q_i at arrival event t is defined as $RT_i^t = Trans_i^t(M^A) - Trans_i^t(A)$.*

Observation B.2 *For regular transmissions $\triangle RT_i = 0$. If there were only M^A-loss and regular transmissions from queue Q_i between two arrival events $t_1 < t_2$ then $\triangle RT_i \geq 0$ (i.e. $RT_i^{t_1} \leq RT_i^{t_2}$).*

Now we prove Lemma B.1 for the full model. We prove it for each arrival event. For transmission events it will follow easily, as we remark at the end of the proof. Obviously, for each arrival event t, $l_i = \sum_{j=1}^{t} z_j^i - Trans_i^t(M^A)$ and $l_i^A = \sum_{j=1}^{t} v_j^i - Trans_i^t(A)$. Therefore

$$l_i^{res} = \sum_{j=1}^{t} z_j^i - \sum_{j=1}^{t} v_j^i + Trans_i^t(A) - Trans_i^t(M^A) = RI_i^t - RT_i^t$$

Clearly, for arrival events in which l_i^{res} is non-positive, $l_i^{res} \leq 2m$. Therefore we look on a portion of the arrival events sequence $t_1 \leq t \leq t_2$ in which the series l_i^{res} begins being positive, i.e., for every arrival event t, $t_1 \leq t \leq t_2$ it holds $l_i^{res} > 0$ but for arrival event $t_1 - 1$ it holds $l_i^{res} \leq 0$.

Claim B.3 *For every arrival event t, $t_1 \leq t \leq t_2$, it holds: $RT_i^{t_1-1} - RT_i^t \leq 0$.*

Proof: Since $l_i^{res} > 0$ in the portion, using Observation B.1 we conclude that there are no A-loss transmissions in this portion, from Observation B.2 it follows that $RT_i^{t_1-1} \leq RT_i^t$. ■

Claim B.4 *For arrival event $t_1 - 1$ it holds: $RT_i^{t_1-1} \geq RI_i^{t_1-1}$.*

Proof: It simply follows from the fact that for arrival event $t_1 - 1$: $0 \geq l_i^{res} = RI_i^{t_1-1} - RT_i^{t_1-1}$. ■

Now it is easy to see that for every arrival event t, $t_1 \leq t \leq t_2$, it holds:

$$l_i^{res} = RI_i^t - RT_i^t = RI_i^t - RT_i^{t_1-1} + (RT_i^{t_1-1} - RT_i^t) \leq RI_i^t - RI_i^{t_1-1} \leq 2m$$

The first inequality follows from claims B.3 and B.4. The last inequality follows from Theorem 3.6 which implies that for every t, $|RI_i^t| = |\sum_{j=1}^t z_j^i - \sum_{j=1}^t v_j^i| \leq m$.

We have proved Lemma B.1 for every *arrival event* but since transmissions can only decrease $|l_i^{res}|$, the inequality holds for both arrival and transmission events. ■

This completes the proof of Theorem 3.7. □

Proof of Theorem 3.8. The theorem follows directly from the next lemma.

Lemma B.5 *For each queue Q_i and event t, $\hat{l}_i^t \geq l_i^t$.*

Proof: The proof is by induction on events for each queue Q_i. For $t = 0$ the claim is trivial. Suppose the claim is true for t, and consider $t+1$. If $t+1$ is a transmission event, then the inequality clearly holds after the transmission, because either the same amount was transmitted from the queues or $l_i^t = 0$ after the transmission, anyway the inequality holds. If $t+1$ is an arrival event of packet p, then after the arrival we consider a few possible cases:

- Packet p was discarded by $\hat{A}$: This means that the empty space in the queues Q^p is less than a unit, thus we conclude that all the queues Q^p in $\hat{A}$ have a load greater than B. Since the queues in A are only of size B the inequality holds.
- Packet p was accepted by $\hat{A}$, and $min(B + 1 - \hat{l}_i, k_i^{t+1}) = k_i^{t+1}$ in queue Q_i: Algorithm A inserted k_i^{t+1} while algorithm $\hat{A}$ inserted at least k_i^{t+1}, so the inequality holds for queue Q_i.
- Packet p was accepted, and $min(B + 1 - \hat{l}_i, k_i^{t+1}) < k_i^{t+1}$ in queue Q_i: This means that $\hat{A}$'s queue Q_i is filled, i.e., $\hat{l}_i = B + 1$, so the inequality holds for queue Q_i.

■

From Lemma B.5, at each transmission event, $\hat{A}$ transmits at least the same size as algorithm A. This completes the proof of Theorem 3.8. □

Proof of Lemma 3.9. Consider the sequence of packets which was accepted by $\hat{A}$ as a sequence for the unbounded node routing problem. Since the sequence of packets which is given to $M^{\hat{A}}$ consists **solely** of the packets which were accepted by $\hat{A}$, using Theorem 3.7 we conclude that if the queues were of size $B + 1 + 2m$ no packet loss may occur for $M^{\hat{A}}$. Thus the throughput of $M^{\hat{A}}$ equals the throughput of $\hat{A}$. □

Proof of Theorem 3.12. Let A_X denote an online algorithm A which works on queues of size X, where $X \geq 0$. Then:

$$\begin{aligned}
Opt_B(\sigma) &\leq c \cdot A_B(\sigma) \\
&\leq c \cdot \hat{A}_{B+1}(\sigma) \\
&= c \cdot M^{\hat{A}}_{B+2m+1}(\sigma) \\
&\leq c \cdot (1 + \frac{2m+1}{B}) E_B^{M^{\hat{A}}}(\sigma)
\end{aligned}$$

where the second inequality is obtained from Theorem 3.8, the equality from Lemma 3.9, and the last inequality from Corollary 3.11. Note that $E_B^{M\hat{A}}$ is a discrete algorithm. $\square$

Proof of Theorem 4.1. The online unweighted matching problem (defined in subsection 3.1) was generalized by Kalyanasundaram and Pruhs in [18] to the b-matching problem. In this model each vertex $r_i \in R$ can be matched to a *single* adjacent vertex $s_j \in S$, while every s_j can be matched up to b times. It is easy to see that in the node routing problem, if we consider only the sequences in which all the transmissions occur after all the packets have arrived, then the node routing problem is equivalent to the b-matching problem with $B = b$. Hence a lower bound for the b-matching is also a lower bound for the node routing problem. In [10] the authors show a lower bound for the unweighted *fractional* online matching (defined in subsection 3.1) of $\frac{e}{e-1} - \Theta(\frac{1}{m})$ even against an integral offline algorithm. This implies a deterministic and a randomized lower bound for the b-matching problem with an arbitrary b by duplicating each request b times. Hence, $\frac{e}{e-1} - \Theta(\frac{1}{m})$ is a lower bound for the node routing problem.

Remark B.1 *Actually, Kalyanasundaram* et al. *proved a tight upper and lower bound of* $\frac{(1+\frac{1}{b})^b}{(1+\frac{1}{b})^b-1}$. *We cannot use this lower bound since it only applies for* $m > b$.

$\square$

Proof of Theorem 4.2. Let A be an algorithm for our problem. We construct the following sequence σ_A: at first arrival phase, a packet arrives to queues $\{Q_1, Q_2\}$. W.l.o.g, algorithm A inserts the packet into Q_1. Then Q_2 transmits. In the second arrival phase a packet arrives to Q_1, and then Q_1 transmits. The above sequence is repeated many times. It is easy to see that in every sequence repetition, A can transmit one packet at most, while Opt can transmit two packets. $\square$

Proof of Claim 5.2. It immediately follows from the matching that:

$$|Opt(\sigma)| \leq |Opt(\sigma) \setminus GR(\sigma)| + |GR(\sigma)| \leq 2 \cdot |GR(\sigma)|$$

$\square$

Proof of Lemma 5.3. What is needed to be shown is that in step 2 an unmarked 1-packet always exists hence the marking is valid. Recall the way algorithm GR acts on 0/1 sequences. In the marking analysis, we can ignore the arrival of 0-packets since rejecting, inserting or exchanging 0-packets does not interfere with the marking scheme. We first prove the following simple claim.

Claim B.6 *For each event* t*, if the packet in the head of queue* Q_i *is unmarked, then all the packets in queue* Q_i *are unmarked.*

Proof: Let queue Q_i hold a marked packet p at event t. When p was marked, queue Q_i contained no 0-packets, hence all the packets closer to the head than p were 1-packets and marked (since the marking scheme marks the unmarked 1-packet which is closest to the head). Transmissions and preemptions might have decreased the number of packets closer to the head than p, but obviously they are still marked. In particular, the packet at the head of queue Q_i is marked. $\blacksquare$

Before we proceed we introduce some notations. We describe the state of the system at a certain point by the contents of the queues in GR and Opt. Let r denote the total number of states. Denote the sequence of states, starting from the initial state, by $[S_1, ..., S_r]$. A change of state can occur whenever a packet arrival or transmission takes place. We denote by $U_i^j (j = 1, ..., r)$ the number of unmarked *1-packets* in queue Q_i (of GR) at state S_j. Finally, we denote by GR_i^j and Opt_i^j the number of 1-packets in queue Q_i at state S_j in GR and Opt, respectively. For simplicity of notation we drop the superscript j whenever the state is clear from context.

To complete the proof all that is left to show is that whenever a 1-packet p associated with subset Q^p is rejected by GR but accepted by Opt into queue $Q_{i'}$ ($Q_{i'} \in Q^p$), queue $Q_{i'}$ contains unmarked packets.

Claim B.7 *For each state S_j, $(1 \leq j \leq r)$, and queue Q_i, $(1 \leq i \leq m)$, $U_i^j \geq GR_i^j - Opt_i^j$, and an unmarked 1-packet is available in queue Q_i if required at that state.*

Proof: First we show that if the inequality holds for state S_j, then an unmarked 1-packet is available if required at that state. Consider the arrival of 1-packet $p \in (Opt(\sigma) \setminus GR(\sigma))$ (we note by $Q_{i'}$ the queue into which Opt inserted p). Since GR did not accept p, we conclude that all the queues in Q^p are full of 1-packets only. So $GR_{i'}^j = B$ but $Opt_{i'}^j < B$ because Opt had place for p in queue $Q_{i'}$. From the inequality it follows that $U_{i'} > 0$; therefore the marking is possible.

We prove the inequality by induction on the states. For the initial state S_1 the inequality clearly holds. We assume correctness for state S_j and prove that the inequality holds for state S_{j+1}. We consider two types of events:

1. Consider the arrival of 1-packet p associated with the subset Q^p. We note by Q_i and $Q_{i'}$ the queues into which GR and Opt insert p, respectively (if such queues exist). In the case of a preempted 1-packet in queue Q_i of GR, both U_i and GR_i do not change, while $Opt_{i'}$ might be increased by 1 (if $p \in Opt(\sigma)$), so the inequality holds for all the queues (even if $i = i'$). Thus we ignore the preemption of 1-packets in the following possible cases:
   (a) $p \notin GR(\sigma)$ and $p \notin Opt(\sigma)$: The state of the system is not changed.
   (b) $p \in (GR(\sigma) \setminus Opt(\sigma))$: Both U_i and GR_i are increased by 1, so the inequality holds.
   (c) $p \in (Opt(\sigma) \setminus GR(\sigma))$: We have a marking in this case (we already showed that if the inequality is valid for state S_j, a marking is possible at that state). The value of $Opt_{i'}$ is increased by 1, while $U_{i'}$ is decreased by 1 because of the marking, hence the inequality holds.
   (d) $p \in (Opt(\sigma) \cap GR(\sigma))$: Here we have two cases:
      i. $i = i'$: GR_i, Opt_i and U_i are increased by 1, hence the inequality holds.
      ii. $i \neq i'$: In the queue Q_i, both GR_i and U_i are increased by 1. In the queue $Q_{i'}$ only $Opt_{i'}$ is increased. Hence, both inequalities hold.
2. Consider the transmission event. A packet p is transmitted from queue Q_i (if such exists). We have three possible cases:
   (a) Queue Q_i is empty in GR: $U_i = 0$, $GR_i = 0$ and $Opt_i \geq 0$, therefore, the inequality holds.
   (b) Packet p is marked: GR_i is decreased by 1, Opt_i might be decreased by 1 (if queue Q_i is not empty in Opt). In any case, the inequality holds.
   (c) Packet p is unmarked: Since the head of queue Q_i was unmarked, using Claim B.6 we conclude that all the packets in queue Q_i are unmarked, therefore $U_i = GR_i$. Since $Opt_i \geq 0$ the inequality holds.

This completes the proof of Claim B.7. ■

Obviously, Lemma 5.3 follows immediately from Claim B.7. □

Proof of Theorem 5.4. Using the marking scheme with Lemma 5.3 and Claim 5.2 we conclude that GR is 2-competitive for 0/1 sequences. Then by using the zero-one principle (Theorem 5.1) we complete the proof. □