The zero-one principle for switching networks

Yossi Azar * Yossi Richter †

Abstract

Recently, approximation analysis has been extensively used to study algorithms for routing weighted packets in various network settings. Although different techniques were applied in the analysis of diverse models, one common property was evident: the analysis of input sequences composed solely of two different values is always substantially easier, and many results are known only for restricted value sequences. Motivated by this, we introduce our zero-one principle for switching networks which characterizes a wide range of algorithms for which achieving c-approximation (as well as c-competitiveness) with respect to sequences composed of 0's and 1's implies achieving c-approximation. The zero-one principle proves to be very efficient in the design of switching algorithms, and substantially facilitates their analysis. We present three applications. First, we consider the Multi-Queue QoS Switching model and design a 3-competitive algorithm, improving the result from [6]. Second, we study the Weighted Dynamic Routing problem on a line topology of length k and present a $(k+1)$-competitive algorithm, which improves and generalizes the results from [1, 11]. As a third application, we consider the work of [14], that compares the performance of local algorithms to the global optimum in various network topologies, and generalize their results from 2-value sequences to arbitrary value sequences.

1 Introduction

Overview: Packet routing networks, most notably the Internet, have become the preferred platform for carrying data of all kinds. Due to the steady increase of network traffic, and the fact that Internet traffic tends to constantly fluctuate, Quality of Service (QoS) networks, which allow prioritization between different traffic streams have gained considerable attention within the networking community. As network overloads become frequent, intermediate switches have to cope with increasing amounts of traffic, while attempting to pass forward more "valuable" packets, where values correspond to the required quality of service for each packet. We can measure the quality of the decisions made within a network by considering the total value of packets that were delivered to their destination.

Traditionally, network routing algorithms were studied within the stability analysis framework, either by a probabilistic model for packet injection (queuing theory, see e.g. [7, 17]) or an adversarial model (adversarial queuing theory, see e.g. [5, 8]). In stability analysis we consider networks with unit-value packets, and the goal is to measure the largest amount of packets ever waiting for transmission on a link, as a function of the network topology and the packet injection model, thereby bounding the buffer size needed to prevent packet drop. Since it seems inevitable to drop

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

†yo@tau.ac.il. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel. Research supported in part by the Israeli Ministry of industry and trade.

packets in real-world networks, approximation analysis framework, which avoids any assumptions on the input sequence and compares the performance of algorithms to the optimal solution, has been adopted recently for studying throughput maximization problems. In particular, competitive analysis has been applied lately to investigate online routing algorithms. Initially, researchers have investigated single-queue switches in various settings [2, 4, 11, 12, 16], and later multi-queue switches [6, 13] and multiple-node networks [1] have been studied.

The zero-one principle: While different techniques were used to analyze algorithms in various switching models, there was one common property: analysis of 2-value sequences, in which packets can take only 2 distinct values, was always substantially easier compared with arbitrary packet sequences. Moreover, many results are known only for restricted value sequences, due to the fact that handling the state of a system containing packets with arbitrary values is significantly more involved. Motivated by this, we introduce the zero-one principle for switching networks. This principle applies to all *comparison-based* switching algorithms, that base their decisions on the relative order between packet values. The principle says that in order to prove that an algorithm achieves c-approximation it is sufficient to prove that it achieves c-approximation with respect to sequences composed solely of 0's and 1's, where ties between packets with equal values may be broken arbitrarily. We note that one can assume that without loss of generality there are no 0-value packets in the input sequence since such packets could have been dropped. Indeed, the optimal solution may ignore all the 0-value packets, however, the comparison-based algorithm may not, since it only regards the relative order between values.

A zero-one principle has been already introduced for sorting networks in the comparison model [3, 9, 15], and it has been proved that a sorting network, that sorts correctly all sequences that are composed of 0's and 1's, is guaranteed to sort correctly an arbitrary sequence. In a similar manner to the zero-one principle for sorting networks, our principle turns out to be very useful in the design of a wide range of approximation algorithms for different switching models, and considerably facilitates their analysis, due to the fact that it allows to focus on 0/1 sequences. We note that in contrast with sorting networks where sorting 0's and 1's is equivalent to sorting any two distinct values, the analysis of 0/1 sequences in switching networks is easier compared with arbitrary 2-value sequences. In addition, we note that there is a myriad of research papers on throughput maximization problems in networks transmitting unit-value packets. Our paper provides a linking step between previous analysis of unit-value networks and QoS networks, by allowing to modify previous analysis only for 0/1 sequences, which are in many cases not very different from uniform sequences. We present three applications to the zero-one principle.

Applications: We first consider the preemptive Multi-Queue QoS Switching model that was originally introduced in [6]. In this model we have a switch with m incoming FIFO queues with bounded capacities and one output port. At each time step new packets arrive to the queues, each associated with a value. Additionally, at each time step the switch selects one non-empty queue and transmits the packet at the head of the queue through the output port. The goal is to maximize the total value of transmitted packets. We present a 3-competitive algorithm for this problem, improving upon the 4-competitive algorithm that was shown in [6].

Our second application is the Weighted Dynamic Routing problem on a line. We consider a network with a topology of a line of length k , i.e. node i ($i = 1, \ldots, k$) is connected to node $i + 1$ by a unidirectional link, and contains a fixed-size FIFO queue to store the packets waiting to be transmitted. At each time step new packets may arrive online to the network nodes, each packet

is associated with a value and a destination node. Additionally, each node can transmit the packet at the head of its queue to the next node. The goal is to maximize the total value of packets that were delivered to their destination. Special cases of this model were studied in [1, 11]. Kesselman *et al.* [11] studied the single-queue model, i.e. $k = 1$, and proved that the natural greedy algorithm is asymptotically 2-competitive. The unweighted version of our model, in which all packets have unit value, was investigated by Aiello *et al.* [1] who proved that the greedy algorithm is $O(k)$-competitive. We prove that the natural greedy algorithm is $(k + 1)$-competitive for the weighted problem, therefore generalizing the results in [1, 11].

Kesselman *et al.* [14] analyzed the performance of local off-line and on-line algorithms compared with the performance of the global optimal solution, in different network topologies. Their strongest results applied only to 2-value sequences composed of 1 and α values. As a third application, we employ our zero-one principle to generalize their results to arbitrary value sequences.

Related results: The online problem of throughput maximization in switches supporting QoS has been studied extensively during recent years. Initially, single queue switches were investigated, for both arbitrary packet sequences, and 2-value sequences. The preemptive model, in which packets stored in the queue can be discarded, was studied in [11, 12, 16]. The non-preemptive model, in which a packet can be discarded only upon arrival, was initially studied by Aiello *et al.* [2], followed by Andelman *et al.* [4] who showed tight bounds. Recently, these results for a single queue were generalized for switches with arbitrary number of queues in [6], by a general reduction from the m queues model to the single queue model. An alternative model is the shared memory QoS switch, in which memory is shared among all queues. Hahne *et al.* [10] studied buffer management policies in this model while focusing on deriving upper and lower bounds for the natural Longest Queue Drop policy.

Aiello *et al.* [1] studied the Dynamic Routing problem with unit packets and investigated the performance of various greedy algorithms with comparison to their stability guarantees. In particular, they showed that the natural greedy algorithm is $O(k)$-competitive for a line topology. In fact, the original model suggested in [1] is non-FIFO, and packets can be transmitted from the queues in any order. Algorithm NTG (Nearest To Go) is shown to be $O(k^{2/3})$-competitive for the non-FIFO model.

Paper structure: The paper is organized as follows. Section 2 includes formal definitions and notations. In section 3 we prove our zero-one principle. We apply our zero-one principle to three different switching models in sections 4, 5 and 6.

2 Definitions and notations

In this section we formally define the three problems we consider throughout the paper and introduce some notations.

Multi-Queue QoS Switching: In the online Multi-Queue QoS Switching problem, originally introduced in [6], we are given a switch with m FIFO queues, where queue i has size B_i, and one output port. Packets arrive online, each packet is destined to one of the queues and is associated with a non-negative value. Initially, the m queues are empty. We assume that time is discrete, and each time step $t \geq 0$ is divided into two phases: at the beginning of the first phase of time t a set of packets arrive to the queues. Packets can be inserted to each queue without exceeding its capacity.

Remaining packets must be discarded. In the second phase of time t, the switching algorithm may select one of the non-empty queues and transmit the packet at the head of the queue. The goal is to maximize the total value of transmitted packets. We consider the preemptive case, in which previously stored packets can be discarded from the queues.

Weighted Dynamic Routing on a line: In the online Weighted Dynamic Routing on a line problem we have a network organized in a line topology of length k, i.e. node $i = 1, \ldots, k$ is connected to node $i+1$ via a unidirectional link with unit capacity. Node $i = 1, \ldots, k$ contains a FIFO queue of size B_i, which is initially empty, to buffer the packets waiting to be transmitted via its outgoing link. 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 arrive online to nodes $i = 1, \ldots, k$, each packet is associated with a value and a destination node. During the second phase of time t, node $i = 1, \ldots, k$ may transmit the packet at the head of its queue 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 enqueued without exceeding the queues capacities. Remaining packets must be discarded. The goal is to maximize the total value of packets that reach their destination.

The Weighted Dynamic Routing problem can be generalized to an arbitrary directed graph $G = (V, E)$, where nodes represent the network switches, and edges represent unidirectional links. In the general case each packet arrives with a predefined path from its source to destination, and each directed link $e \in E$ is associated with a capacity $c(e)$, and a FIFO queue Q_e of size $size(Q_e)$, to store the packets waiting for transmission on e. A special case of this problem was originally introduced in [1], where all packets had unit values, links had unit capacities, and the nodes contained buffers instead of FIFO queues, i.e. transmission was allowed in any order.

Work-Conserving Weighted Dynamic Routing: Kesselman *et al.* [14] studied the dynamic routing problem in the same model described in the previous paragraph, under the *work-conserving* assumption. In the following, we briefly review their definitions. A schedule is called *work-conserving* if for every time t and link e, the number of packets transmitted over e at time t is the minimum between $c(e)$ and the number of packets in Q_e. We consider two network topologies: a directed line topology, identical to the one defined in the previous paragraph, and a directed tree topology, where packets are only injected at the leaves and are destined to the root. An algorithm is called *local on-line* if its action at time t at node v depends only on the packets arriving to v until time t. An algorithm is called *local off-line* if its action at time t at node v depends only on the sequence of packets arriving at node v (possibly after time t). We denote by OptL_v the local off-line optimal solution, that maximizes the total value transmitted out of v. For any acyclic network we denote by OptL the algorithm that exercises OptL_v in topological order. We denote by Opt the global work-conserving optimal algorithm that knows all the information in the system (all nodes, all times).

We now introduce the notations we use for all problems. We denote by $\sigma = \{p_1, \ldots, p_n\}$ the finite sequence of incoming packets, and by $v_\sigma : \sigma \to \mathbb{R}$ the packets value function. Let $\mathcal{A}$ be a switching algorithm, we denote by $\mathcal{A}(\sigma)$ the set of packets delivered by $\mathcal{A}$ given the sequence σ (also referred to as the output sequence). Given a packet sequence σ, we denote by $V(\sigma)$ the total value of packets in the sequence, i.e. $V(\sigma) = \sum_{p \in \sigma} v_\sigma(p)$. The optimal solution is denoted by Opt.

A deterministic (resp. randomized) approximation algorithm $\mathcal{A}$ achieves c-approximation ($c \geq 1$) for a maximization problem iff for every packet sequence σ we have: $V(\mathsf{Opt}(\sigma)) \leq c \cdot V(\mathcal{A}(\sigma))$ (resp. $V(\mathsf{Opt}(\sigma)) \leq c \cdot E[V(\mathcal{A}(\sigma))]$). The definition of the competitive ratio with respect to online

algorithms is the same.

3 The zero-one principle

In this section we introduce our zero-one principle for switching networks. We begin with some additional definitions and notations. Given a function $f : \mathbb{R} \to \mathbb{R}$ we denote by $f(\sigma)$ the packet sequence σ with the modified value function $v_{f(\sigma)} = f \circ v_\sigma$. For a given switching algorithm $\mathcal{A}$ we use the notation $f(\mathcal{A}(\sigma))$ to denote the set of transmitted packets with modified values. In the following we define comparison-based algorithms for switching networks. Informally, an algorithm $\mathcal{A}$ is said to be comparison-based if it bases its decisions on relative order between values alone, and may break ties arbitrarily. We note that the latter means that $\mathcal{A}(\sigma)$ is the *set* of all possible output sequences, where we break ties between equal packet values in all possible ways (even if algorithm $\mathcal{A}$ breaks ties in a specific way).

Definition 3.1 *Let $\mathcal{A}$ be a switching algorithm that may break ties arbitrarily between packets with equal values. $\mathcal{A}$ is called* comparison-based *if and only if for every monotonically increasing function $f : \mathbb{R} \to \mathbb{R}$ we have $\mathcal{A}(\sigma) \subseteq \mathcal{A}(f(\sigma))$.*

For the remainder of this section we slightly abuse notation, and denote by $\mathcal{A}(\sigma)$ a *specific* output sequence in $\mathcal{A}(\sigma)$ and by $\mathcal{A}(f(\sigma))$ the *same* output sequence in $\mathcal{A}(f(\sigma))$ (where f is a monotonically increasing function), that exists according to definition 3.1.

Theorem 3.1 ***[zero-one principle]*** *Let $\mathcal{A}$ be a comparison-based switching algorithm (deterministic or randomized). $\mathcal{A}$ is a c-approximation algorithm if and only if $\mathcal{A}$ achieves c-approximation for all packet sequences whose values are restricted to 0/1.*

Proof: The "only if" direction is straightforward (even if $\mathcal{A}$ breaks ties between equal packet values in a specific way), thus it remains to prove the other direction. For simplicity of notation we prove the theorem for deterministic algorithms. Using linearity of expectation the theorem follows for randomized algorithms. We define the monotonically increasing step function $f_t(x)$ by: $f_t(x) = 1$ if $x \geq t$, and otherwise $f_t(x) = 0$. In the following claims we show that the set of packets delivered by an algorithm $\mathcal{A}$ can be broken into sequence of 0's and 1's by using f_t. If $\mathcal{A}$ is comparison-based we can apply the transformation directly to the input sequence.

Claim 3.2 *Let σ be any packet sequence. Then the following holds: $V(\sigma) = \int_{t=0}^{\infty} V(f_t(\sigma))dt$.*

Proof:

$$V(\sigma) = \sum_{p \in \sigma} v_\sigma(p) = \sum_{p \in \sigma} \int_{t=0}^{\infty} f_t(v_\sigma(p))dt = \int_{t=0}^{\infty} \sum_{p \in \sigma} f_t(v_\sigma(p))dt = \int_{t=0}^{\infty} V(f_t(\sigma))dt,$$

where the second equality follows from $\int_{t=0}^{\infty} f_t(x)dt = x$. ■

Claim 3.3 *Let $\mathcal{A}$ be a comparison-based switching algorithm and let f be any monotonically increasing function. Then the following holds for any packet sequence σ: $V(f(\mathcal{A}(\sigma))) = V(\mathcal{A}(f(\sigma)))$.*

Proof:

$$V(f(\mathcal{A}(\sigma))) = \sum_{p \in \mathcal{A}(\sigma)} f(v_\sigma(p)) = \sum_{p \in \mathcal{A}(f(\sigma))} f(v_\sigma(p)) = V(\mathcal{A}(f(\sigma))),$$

where the second equality follows since $\mathcal{A}$ is comparison-based. ■

We can now show for every sequence σ:

$$\begin{aligned} V(\mathcal{A}(\sigma)) &= \int_{t=0}^{\infty} V(f_t(\mathcal{A}(\sigma)))dt = \int_{t=0}^{\infty} V(\mathcal{A}(f_t(\sigma)))dt \\ &\geq \int_{t=0}^{\infty} \frac{1}{c} \cdot V(\mathsf{Opt}(f_t(\sigma)))dt \geq \frac{1}{c} \cdot \int_{t=0}^{\infty} V(f_t(\mathsf{Opt}(\sigma)))dt \\ &= \frac{1}{c} \cdot V(\mathsf{Opt}(\sigma)), \end{aligned}$$

where the first and last steps follow from claim 3.2, the second step follows from claim 3.3, the third step follows since $f_t(\sigma)$ is a 0/1 sequence, and the forth step follows since $\mathsf{Opt}(f_t(\sigma))$ maximizes the number of delivered packets with original value at least t. ■

Given Theorem 3.1, if we wish to prove that a comparison-based algorithm $\mathcal{A}$ achieves c-approximation, it is enough to show that for every 0/1 packet sequence σ, and for each possible output sequence $\mathcal{O} \in \mathcal{A}(\sigma)$, i.e. for every possible way of breaking ties between equal values, we have: $V(\mathsf{Opt}(\sigma)) \leq c \cdot V(\mathcal{O})$.

In many cases, the approximation ratio of a switching algorithm is given in terms of the ratio between the largest to smallest packet values, denoted α. In the following theorem we extend the zero-one principle to this case, and prove that it is sufficient to consider packet sequences composed solely of two values: 1 and α.

Theorem 3.4 *Let $\mathcal{A}$ be a comparison-based switching algorithm (deterministic or randomized). $\mathcal{A}$ is a $c(\alpha)$-approximation algorithm if and only if $\mathcal{A}$ achieves $c(\alpha)$-approximation for all packet sequences whose values are restricted to $1/\alpha$.*

Proof: The "only if" direction is straightforward, thus it remains to prove the other direction. Again, for simplicity of notation we prove the theorem for deterministic algorithms, while the randomized case follows easily. We define the monotonically increasing step function $g_t(x) : [1, \alpha] \rightarrow \{1, \alpha\}$ for $1 \leq t \leq \alpha$ as follows: $g_t(x) = \alpha$ if $1 \leq t \leq x \leq \alpha$, otherwise $g_t(x) = 1$. We begin by restating claim 3.2.

Claim 3.5 *Let σ be any packet sequence whose values lie in the range $[1, \alpha]$. Then the following holds: $V(\sigma) = \frac{1}{\alpha - 1} \int_{t=1}^{\alpha} V(g_t(\sigma))dt$.*

Proof:

$$V(\sigma) = \sum_{p \in \sigma} v_\sigma(p) = \sum_{p \in \sigma} \frac{1}{\alpha - 1} \int_{t=1}^{\alpha} g_t(v_\sigma(p))dt = \frac{1}{\alpha - 1} \int_{t=1}^{\alpha} \sum_{p \in \sigma} g_t(v_\sigma(p))dt = \frac{1}{\alpha - 1} \int_{t=1}^{\alpha} V(g_t(\sigma))dt,$$

where the second equality follows since $\int_{t=1}^{\alpha} g_t(x)dt = \alpha(x-1) + (\alpha - x) = (\alpha - 1)x$. ■

We can now show for every sequence σ whose values lie in $[1, \alpha]$:

$$\begin{aligned} V(\mathcal{A}(\sigma)) &= \frac{1}{\alpha - 1} \int_{t=1}^{\alpha} V(g_t(\mathcal{A}(\sigma)))dt = \frac{1}{\alpha - 1} \int_{t=1}^{\alpha} V(\mathcal{A}(g_t(\sigma)))dt \\ &\geq \frac{1}{\alpha - 1} \int_{t=1}^{\alpha} \frac{1}{c} \cdot V(\mathsf{Opt}(g_t(\sigma)))dt \geq \frac{1}{\alpha - 1} \cdot \frac{1}{c} \cdot \int_{t=1}^{\alpha} V(g_t(\mathsf{Opt}(\sigma)))dt \\ &= \frac{1}{c} \cdot V(\mathsf{Opt}(\sigma)), \end{aligned}$$

where the first and last steps follow from claim 3.5, the second step follows from claim 3.3, the third step follows since $g_t(\sigma)$ is a $1/\alpha$ sequence, and the forth step follows since $\mathsf{Opt}(g_t(\sigma))$ maximizes the total value (with respect to g_t) of the packets transmitted. ■

4 Application 1: Multi-Queue QoS Switching

In this section we present the first application of our zero-one principle. We consider the problem of online Multi-Queue QoS Switching and design a 3-competitive algorithm for the problem, improving the 4-competitive algorithm suggested in [6].

We begin by repeating the description of the greedy preemptive admission control algorithm for a single queue from [11] (figure 1) and then we present our algorithm **TransmitLargestHead** (abbreviated TLH) for the problem (figure 2).

Algorithm Greedy [**Single-Queue**]

Enqueue a new packet if:

- The queue is not full.
- Or the packet with the smallest value in the queue has a lower value than the current packet. In this case the smallest packet is discarded and the new packet is enqueued.

Figure 1: Algorithm Greedy.

Algorithm TLH [**Multi-Queue**]

1. **Admission control:** use algorithm Greedy for admission control in all m incoming queues.
2. **Scheduling:** at each time step, transmit the packet with the largest value among all packets at the head of the queues.

Figure 2: Algorithm TLH.

Theorem 4.1 *Algorithm* TLH *is 3-competitive for the Multi-Queue QoS Switching problem.*

Proof: Clearly, algorithm TLH bases its admission control and scheduling decisions solely on relative order between values, therefore is it comparison-based. According to our zero-one principle (Theorem 3.1) we need only show that the algorithm is 3-competitive for 0/1 sequences. We note, however, that since a comparison-based algorithm may break ties arbitrarily between packets with equal values, we should consider cases where a packet stored in the queues is discarded in favor of a packet with an equal value.

Let σ be a 0/1 sequence composed solely of packets with value 1 (abbreviated 1-packets) and packets with value 0 (0-packets in short). With slight abuse of notation we denote by $\mathsf{TLH}(\sigma)$ the set of 1-packets transmitted by the algorithm, and by $\mathsf{Opt}(\sigma)$ the set of packets transmitted by the optimal solution (note that we may assume that the optimal solution does not transmit any 0-packets). Theorem 4.1 directly follows from the next lemma.

Lemma 4.2 *For every 0/1 sequence σ we have:* $|\mathsf{Opt}(\sigma) \setminus \mathsf{TLH}(\sigma)| \leq 2 \cdot |\mathsf{TLH}(\sigma)|$.

Proof: We prove the lemma by providing a matching from $(\mathsf{Opt}(\sigma) \setminus \mathsf{TLH}(\sigma))$ to $\mathsf{TLH}(\sigma)$ in which each 1-packet from $\mathsf{TLH}(\sigma)$ is matched at most twice. This is done by a marking scheme that marks one of the 1-packets stored in the queues according to TLH operation, whenever a 1-packet

Marking scheme

For each time step t do:

1. For each incoming 1-packet to queue i do:
   (a) If the packet is accepted by TLH, consider it as an unmarked packet.
   (b) Otherwise, if the packet is accepted by Opt, look for the first unmarked 1-packet, starting from the head of the queue and moving to its tail, and mark it.
2. In the transmission phase, whenever Opt transmits a packet do:
   (a) If Opt and TLH use the same queue for transmission, do nothing.
   (b) Otherwise, let $i \neq j$ be the queues used by Opt and TLH, respectively. If queue i contains marked packets, unmark the marked 1-packet closest to the tail in queue i and mark the packet transmitted from queue j.

Figure 3: Marking scheme for TLH.

is accepted by Opt but rejected by TLH, and when transmissions take place. A description of the marking scheme follows (figure 3).

Clearly, each 1-packet in $\mathsf{TLH}(\sigma)$ can be marked at most twice, once while it resides in the queue and once while it is transmitted. The following claims prove that each 1-packet transmitted by Opt but not by TLH is matched through the marking scheme to a 1-packet in $\mathsf{TLH}(\sigma)$. We begin by proving that whenever we change a marking (step 2b) a 1-packet is transmitted, and therefore the change is possible.

Claim 4.3 *For each time step t, if the queues hold any marked packets, then a 1-packet is transmitted.*

Proof: Let queue i hold a marked packet p at time t. When p was marked, queue i contained no 0-packets, hence all the packets closer to the head than p are 1-packets. In particular, the packet at the head of queue i at time t is a 1-packet. Therefore, a 0-packet can not be transmitted at time t. ■

Before we proceed we introduce some notations. Let us describe the state of the system at a certain point by the contents of the queues in TLH and Opt. We denote the sequence of states, starting from the initial state, by $[S_1, \ldots, S_r]$. A change of state can occur whenever a packet arrives or transmission takes place. We denote by U_i^j $(j = 1, \ldots, r)$ the number of unmarked packets in queue i at state S_j. We further denote by TLH_i^j and Opt_i^j the number of 1-packets in queue i at state S_j in TLH and Opt, respectively. For simplicity of notation we drop the superscript j whenever the state is clear from context. To complete the proof of Lemma 4.2, it remains to show that whenever a 1-packet destined to queue i is rejected by TLH but accepted by Opt, queue i contains unmarked packets (step 1b).

Claim 4.4 *For each state S_j $(1 \leq j \leq r)$ and queue i, $(1 \leq i \leq m)$, $U_i^j \geq \mathsf{TLH}_i^j - \mathsf{Opt}_i^j$.*

Proof: First recall that TLH may breaks ties arbitrarily. However, exchanging 0-packets does not interfere with the marking scheme, and 1-packets can be exchanged only when the queue is full

with 1-packets, hence we may ignore this scenario as it does not change the values stored in the queue. We prove the claim 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}. Consider a 1-packet p arriving to queue i. If p is accepted by TLH, U_i increases and the inequality holds. Otherwise, if p is rejected by TLH and accepted by Opt, then by the induction hypothesis $U_i \geq \mathsf{TLH}_i - \mathsf{Opt}_i$ and since $\mathsf{TLH}_i = B_i$ and $\mathsf{Opt}_i < B_i$ we have an unmarked packet in queue i. After we mark a 1-packet both sides of the inequality decrease by 1. Now, consider the transmission phase. Let i and j be the queues used for transmission by Opt and TLH, respectively. The inequality clearly holds for queue j if it does not hold any marked packets. Otherwise, a marked packet is transmitted from queue j (since we mark packets in the direction from the head to the tail), U_j is unchanged while the right hand side of the inequality can only decrease. If $i \neq j$, we increase U_i by unmarking a packet (step 2b), and the inequality continues to hold for queue i.

This concludes the proof of Lemma 4.2 and Theorem 4.1. ■

The following lemma proves that our analysis of TLH is asymptotically tight. Proof can be found in appendix A

Lemma 4.5 *Algorithm* TLH *is at least* $(3 - \frac{1}{m})$*-competitive.*

5 Application 2: Weighted Dynamic Routing

In this section we present our second application of the zero-one principle. We consider the problem of dynamic routing on a line (see section 2 for formal definition) and present a $(k+1)$-competitive algorithm for the problem (figure 4).

Algorithm PF **(PushForward)**

For each node $i = 1, \ldots, k$ do:

1. **Arrival phase:** use the Greedy algorithm to accept packets into the queue.
2. **Transmission phase:** if the queue is not empty, transmit the packet at the head of the queue to the next node.

Figure 4: Algorithm PF.

Theorem 5.1 *Algorithm* PF *is* $(k+1)$*-competitive for the Weighted Dynamic Routing problem on a line of length* k*.*

Proof: Clearly, algorithm PF bases its admission control decisions solely on relative order between values, therefore it is comparison-based. According to our zero-one principle (Theorem 3.1) we need only show that the algorithm is $(k+1)$-competitive for 0/1 sequence.

Let σ be a 0/1 sequence. With slight abuse of notation we denote by $\mathsf{PF}(\sigma)$ the set of 1-packets that were delivered by PF, and by $\mathsf{Opt}(\sigma)$ the set of packets delivered by Opt. Theorem 5.1 directly follows from the next lemma.

Lemma 5.2 *For every 0/1 sequence* σ *we have:* $|\mathsf{Opt}(\sigma) \setminus \mathsf{PF}(\sigma)| \leq k \cdot |\mathsf{PF}(\sigma)|$.

Proof: We prove the lemma by providing a matching from $(\mathsf{Opt}(\sigma) \setminus \mathsf{PF}(\sigma))$ to $\mathsf{PF}(\sigma)$ in which each packet from $\mathsf{PF}(\sigma)$ is matched at most k times. This is done by a marking scheme that marks one of the 1-packets in PF queues whenever a 1-packet is accepted by Opt but rejected by PF. A description of the marking scheme follows (figure 5).

Marking scheme

For each node $i = 1, \ldots, k$ do:

1. For each incoming 1-packet do:
   (a) If the packet is accepted by PF, consider it as an unmarked packet.
   (b) Otherwise, if the packet is accepted by Opt, look for the first unmarked 1-packet, starting from the head of queue i and moving to its tail, and mark it.
2. In the transmission phase:
   If a 1-packet arrives from the previous node, it is considered to be unmarked.

Figure 5: Marking scheme for PF.

We observe that each 1-packet is marked at most k times (at most once in each of the intermediate nodes on its path), and that packets are not dropped during the transmission phase since all nodes that store packets push a packet forward. Therefore, it remains to prove that we always have an available unmarked packet when we reject a packet that is accepted by Opt (step 1b in the marking scheme). In the following we refer to system states $[S_1, \ldots, S_r]$ and use the notations U_i^j, PF_i^j and Opt_i^j $(i = 1, \ldots, k\ ,\ j = 1, \ldots, r)$ with similar meanings to those used in the proof of Claim 4.4.

Claim 5.3 *For each state S_j $(1 \leq j \leq r)$ and queue i $(1 \leq i \leq k)$, $U_i^j \geq \mathsf{PF}_i^j - \mathsf{Opt}_i^j$.*

Proof: First, we recall that algorithm *PF* may break ties arbitrarily. We may ignore these scenarios from the same considerations given in the proof of Claim 4.4. We prove the claim by induction on the states. At the initial state S_1 the queues are empty, hence the inequality clearly holds. We assume correctness for state S_j and prove that the inequality continues to hold for state S_{j+1}. We first consider the packet arrival phase. Consider a 1-packet p arriving to queue i. If p is accepted by PF, U_i increases and the inequality holds. Otherwise, if p is rejected by PF and is accepted by Opt, then by the induction hypothesis $U_i \geq \mathsf{PF}_i - \mathsf{Opt}_i$ and since $\mathsf{PF}_i = B_i$ and $\mathsf{Opt}_i < B_i$ we have an unmarked 1-packet in queue i. After we mark a packet both sides of the inequality decrease by 1. Next, consider the transmission phase. Recall that no packets are dropped during this phase. We examine the changes in both the transmitting end and the receiving end. At the transmitting end, note that if queue i does not contain marked packets then the inequality clearly holds. Otherwise, U_i remains unchanged (since a marked packet is transmitted) while the right hand side of the inequality can only decrease (if Opt chooses not to transmit). At the receiving end, we first observe that a packet that reaches its destination has no effect. A 1-packet enqueued into queue i in PF causes U_i to increase; A packet enqueued by Opt causes the right hand side of the inequality to decrease. In either case, the inequality continues to hold. ■

Theorem 5.1 can be generalized as follows. The proof follows the same lines and is omitted.

Theorem 5.4 *Let $G = (V, E)$ be a directed graph with uniform edge capacities such that $\forall v \in V\, d_{in}(v) = 1$. The competitive ratio of algorithm* PF *on networks with topology G is $\ell(G) + 1$ (where $\ell(G)$ is the length of the longest acyclic directed path in G).*

Observe that in particular Theorem 5.4 implies that algorithm PF is $(k+1)$-competitive for a cycle topology. The following lemma shows that our analysis of PF is tight up to a factor of 2. Proof can be found in appendix A

Lemma 5.5 *Algorithm* PF *is at least $k/2$-competitive on a line topology of length k.*

6 Application 3: Work-Conserving Weighted Dynamic Routing

Kesselman *et al.* [14] studied the dynamic routing problem under the *work-conserving* assumption. Specifically, they compared the performance of the local on-line Greedy algorithm and the local off-line OptL algorithm to the global work-conserving optimum, in networks with line and tree topologies (see section 2 for exact definitions). The strongest results in [14] are shown only for sequences restricted to two values: 1 and α, while the bounds for arbitrary sequences remain as open questions. Using our zero-one principle we can generalize their results to arbitrary sequence as follows. First, we observe that the local off-line algorithm, OptL_v, is in fact a comparison-based algorithm that sorts the sequence of packets according to their values and then attempts to schedule the packets starting from the largest value (see e.g. [16]). The local on-line Greedy algorithm is clearly comparison-based. With a few minor modifications, that are omitted here, we can modify the proofs shown in [14] to work for 0/1 sequences and algorithms that break ties arbitrarily. Before we proceed to restate the main results from [14] in their general form, we need to introduce some additional definitions and notations from [14].

Definition 6.1 *For a given link e in a given network, define the* delay *of e, denoted $D(e)$, to be the ratio $\lceil size(Q_e)/c(e) \rceil$. The delay of a given path is the sum of the edge delays on that path.*

Definition 6.2 *Let $e = (u, v)$ be any directed link in a given tree topology. The* height *of e, denoted $h(e)$, is the maximum path delay, over all paths starting at a leaf and ending at v. The* weakness *of e, denoted $\lambda(e)$, is defined to be $\lambda(e) = \frac{h(e)}{D(e)}$.*

We are now ready to restate the main results from [14] in their generalized form, for arbitrary sequences. We emphasize that all other results in [14] restricted to sequences with two values can be generalized as well.

Theorem 6.1 *The competitive ratio of* Greedy *for any given tree topology $G = (V, E)$ is $O(\max\{\lambda(e) \mid e \in E\})$.*

Theorem 6.2 *The approximation ratio of* OptL *for a complete binary tree of depth h is at most $O(h/\log h)$.*

Theorem 6.3 *The approximation ratio of* OptL *for a line of length h is $O(\sqrt{h})$.*

References

[1] 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.

[2] W. A. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosen. Competitive queue policies for differentiated services. In *Proceedings of the IEEE INFOCOM 2000*, pages 431–440.

[3] M. Ajtai, J. Komlós, and E. Szemerédi. An $o(n \log n)$ sorting network. In *Proc. 15th ACM Symp. on Theory of Computing*, pages 1–9, 1983.

[4] N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for QoS switches. In *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms*, pages 761–770, 2003.

[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] 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.

[7] 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.

[8] 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.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[10] 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.

[11] 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.

[12] A. Kesselman and Y. Mansour. Loss-bounded analysis for differentiated services. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, pages 591–600, 2001.

[13] A. Kesselman and A. Rosén. Scheduling policies for CIOQ switches. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 353–362, 2003.

[14] Alex Kesselman, Zvi Lotker, Yishay Mansour, and Boaz Patt-Shamir. Buffer overflows of merging streams. In *Proc. 11th Annual European Symposium on Algorithms*, pages 349–360, 2003.

[15] Donald E. Knuth. *The Art of Computer Programming, Sorting and Searching*, volume 3. Addison-Wesley, Reading, MA, 1973.

[16] Z. Lotker and B. Patt-Shamir. Nearly optimal fifo buffer management for diffserv. In *Proc. 21st ACM Symp. on Principles of Distrib. Computing*, pages 134–143, 2002.

[17] 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.

A Lower bounds

In this section we prove the lower bounds presented in the paper.

Proof: **[Lemma 4.5]** We construct the following sequence σ:

1. At $t = 0$, B_i packets with value ε arrive to queue i $(i = 1, \ldots, m)$.
2. For $1 \leq i \leq m-1$, during time steps $\sum_{j=1}^{i-1} B_j, \ldots, (\sum_{j=1}^{i} B_j) - 1$, B_i packets with value $1+\delta$ arrive to queue i, one packet at a time. Then, during the next B_m time steps, B_m packets with value 1 arrive to queue m, one packet at a time. Denote by P the set of packets arriving at this phase.
3. When phase 2 is completed, a set of packets P arrives (identical to the set of packets from phase 2, only they all arrive together).
4. During the next $\sum_{i=1}^{m-1} B_i$ time steps, a packet with value 1 arrives at each time step to queue m.
5. No more packets arrive.

We now analyze the competitive ratio of TLH while we assume w.l.o.g $B_m = \min_{1 \leq i \leq m} B_i$ and we take $\varepsilon, \delta \to 0$, thereby considering σ to consist of 0's and 1's. Algorithm TLH accepts all the packets that arrive at step 1. During phase 2 these ε-packets are transmitted while the queues are filled with 1-packets. All the packets that arrive in step 3 are thus discarded. During phase 4 TLH empties queues $[1, m-1]$ first, therefore discarding all the packets that arrive during this phase to queue m. In contrast, Opt discards all the ε-packets that arrive at the beginning, and can therefore transmit all subsequent packets. Hence:

$$\frac{\mathsf{Opt}(\sigma)}{\mathsf{TLH}(\sigma)} = \frac{3 \cdot \sum_{i=1}^{m} B_i - B_m}{\sum_{i=1}^{m} B_i} \geq 3 - \frac{1}{m}.$$

■

Proof: **[Lemma 5.5]** Consider the following packet sequence σ arriving to a network with uniform queues sizes:

1. At time $t = 0$, $2B$ packets are injected at node i $(i = 1, \ldots, k)$, the first B packets have value $1+\varepsilon$ and are destined to node $k+1$, the last B packets have value 1 and are destined to node $i+1$.
2. Packets with value 1 and destination $i+1$ are injected at node i $(i = 2, \ldots, k)$ during time steps $1, \ldots, i \cdot B - 1$.

Algorithm PF accepts the B more valuables packets injected at step 1 at each node and rejects all other packets in the sequence. In contrast, Opt rejects the valuable packets injected at step 1, and can therefore deliver all other packets. Taking $\varepsilon \to 0$ we obtain:

$$\frac{\mathsf{Opt}(\sigma)}{\mathsf{PF}(\sigma)} = \frac{\sum_{i=1}^{k-1} i \cdot B}{(k-1)B} = \frac{k}{2}.$$

■