Randomization does not Reduce the Average Delay in Parallel Packet Switches

Hagit Attiya
Department of Computer Science
Technion
hagit@cs.technion.ac.il

David Hay
Department of Computer Science
Technion
hdavid@cs.technion.ac.il

ABSTRACT

Switching cells in parallel is a common approach to build switches with very high external line rate and a large number of ports. A prime example is the *parallel packet switch* (in short, PPS) in which a demultiplexing algorithm sends cells, arriving at rate R on N input-ports, through one of K intermediate slower switches, operating at rate $r < R$.

This paper presents lower bounds on the *average* queuing delay introduced by the PPS relative to an optimal work-conserving FCFS switch, for demultiplexing algorithms that does not have full and immediate information about the switch status. The bounds hold even if the algorithm is *randomized*.

These lower bounds are shown to be asymptotically optimal through a new methodology for analyzing the *maximal* relative queuing delay; this clearly upper bounds their average relative queuing delay. The methodology is used to devise a new algorithm that relies on slightly out-dated global information on the switch status. It is also used to provide, for the first time, a complete proof of the maximum relative queuing delay provided by the *fractional traffic dispatch* [19, 22] algorithm.

Categories and Subject Descriptors

B.4.3 [**Hardware**]: INPUT/OUTPUT AND DATA COMMUNICATIONS—*Interconnections*; C.2.6 [**Computer Systems Organization**]: COMPUTER-COMMUNICATION NETWORKS—*Internetworking*

General Terms

Algorithms, Performance, Theory

Keywords

Load balancing, Inverse multiplexing, Packet switching, Clos networks, Queuing delay, Randomization, Adaptive Adversary, Oblivious Adversary

1. INTRODUCTION

Parallelism often increases the throughput of a system, by distributing tasks among several processing entities. Careful *load balancing* is required to ensure even distribution and guarantee small delay for each task. *Randomization* is an attractive paradigm for balancing load on the average [4, 26]: even a very simple strategy ensures with high probability maximum load close to the optimal distribution [17].

Load balancing has recently been employed in packet switching architectures with high line-rates and large number of ports [6, 20]. A successful example of such a switch is the *parallel packet switch (PPS)* [18], which is the core of several contemporary switches (e.g., [1, 11, 25, 28]). Like *inverse multiplexing* systems [2, 9, 14, 15], an $N \times N$ PPS demultiplexes cells, arriving at rate R, through K slower switches (*planes*) operating at rate $r < R$ (see Figure 1).

The PPS is evaluated by the *queuing delay* it introduces *relative* to an *optimal work-conserving shadow* switch that receives the same incoming traffic.[1] This comparison is not burdened by unsubstantiated probabilistic assumptions on the incoming traffic [13] and reveals inherent strengths and weaknesses of the PPS. As we shall prove, the relative queuing delay is determined by the balancing of cells among the planes. Given the successful application of randomization in traditional load balancing settings (discussed above) and in other high-bandwidth switches [16, 32], it is tempting to employ randomization to reduce the average imbalance between planes and by that reduce the average relative queuing delay.

This paper shows that randomization does not help to decrease the average relative queuing delay. This surprising result holds because of the *per-flow FCFS* property required in most switches; switches must respect the arrival order of cells with the same input-port and the same output-port, since the common practice is that switches should not missequence cells [21]. This property allows an *adaptive* adversary [27] to exploit a transient increase of the relative queuing delay and perpetuate it sufficiently long to increase the *average* relative queuing delay.

Specifically, we devise traffic that exhibits with high probability a large average relative queuing delay. The exact bounds depend on the locality of information used for cell

[1]A work-conserving switch guarantees that an output-port is never idle unnecessarily, and by that, maximizes the switch throughput and minimizes its average cell delay [10, 23, 24]. This generalizes the common practice of comparing with an output-queued switch [8, 18, 19, 22, 24, 29, 30, 31].

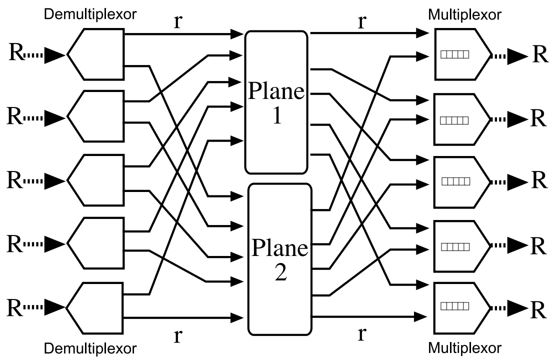


Figure 1: A 5×5 PPS with 2 planes in its center stage, without buffers in the input-ports

demultiplexing and are equal to the lower bounds known for maximum relative queuing delay [3]. The lower bound is $\Omega(\min\{u, \frac{R}{r}\} \cdot N)$ time-slots for algorithms that use global information older than u time-slots (namely, u real-time distributed algorithms [3]), and $\Omega(\frac{R}{r}N)$ time-slots for algorithms that only use local information.

The second lower bound holds also with an *oblivious* adversary [5], if the PPS obeys a *global* FCFS policy (that is, all cells to the same destination should leave the switch in their arrival order).

To prove that these bounds are tight, we present a new methodology for evaluating the relative queuing delay under *global* FCFS policies. We show a general upper bound that depends on the difference between the number of cells with the same destination that are sent through a specific plane, and the total number of cells with this destination.

We devise a new algorithm that uses global information that is 1 time-slot old, and show that its maximum relative queuing delay is $O(N)$ time-slots. This bound asymptotically matches the lower bound on the average relative queuing delay for this class of demultiplexing algorithms (even with randomization).

Finally, we employ our new methodology to prove that the relative queuing delay of the *fractional traffic dispatch (FTD)* algorithm [19, 22] is $O(N\frac{R}{r})$ time-slots. (Previous attempts to bound the relative queuing delay of FTD [19, 22] turned out to be flawed.) This matches the lower bound on the average relative queuing delay introduced by fully-distributed demultiplexing algorithms (even with randomization).

2. THE BUFFERLESS PPS MODEL

A *switch* handles fixed-size *cells* that arrive and leave at rate R, in discrete *time-slots*: Each cell c arrives at time $ta(c)$ to input-port $orig(c)$, and it is destined for output-port $dest(c)$. We assume the switch does not drop cells.

A *traffic* T is a finite collection of cells, and a *flow* (i, j) is the collection of cells sent from input-port i to output-port j. The *projection* of a traffic T on a set of input-ports I, denoted by $T|_I$, is $\{c \in T \mid orig(c) \in I\}$. A traffic T is *legal* if for any input-port i, there are no two cells $c_1, c_2 \in T|_i$ such that $ta(c_1) = ta(c_2)$; in this case, the arrival times of cells in $T|_i$ induce a total order on them. It is sometime necessary to compose in parallel two traffics, T_1 and T_2, from disjoint sets of input-ports: Namely, if $I_1 \cap I_2 = \emptyset$ then $T_1|_{I_1} \parallel T_2|_{I_2} = T_1 \cup T_2$.

For any cell c, shift(c, t) is a cell with the same origin and destination such that $ta(\text{shift}(c, t)) = ta(c) + t$. The *shift* operation is used for concatenating two traffics, T_1 and T_2, so that T_2 starts after the last cell of traffic T_1. Formally, $T_1 \circ T_2$ is the traffic $T_1 \cup \{\text{shift}(c, t) \mid c \in T_2\}$, where $t = 1 + \max\{ta(c) \mid c \in T_1|_{I_1}\}$.

An $N \times N$ *PPS* is a three-stage Clos network [12], with $K < N$ planes. Each plane is an $N \times N$ switch operating at rate $r < R$, and is connected to all the input-ports on one side, and to all the output-ports on the other side (see Figure 1). The *speedup* $S = \frac{Kr}{R}$ captures the switch over-capacity.

A *bufferless* PPS has no buffers at its input-ports but can store pending cells in its planes and in its output-ports. Each cell arriving at input-port i is immediately sent to one of the planes; the plane through which the cell is sent is determined by a randomized state machine with state set $\mathbb{S}_i$, following some algorithm.

Definition 2.1. *The demultiplexing algorithm of a bufferless input-port i is a function*

$$\texttt{ALG}_i : \{1, \ldots, N\} \times \mathbb{S}_i \times \texttt{COINSPACE} \rightarrow \{1, \ldots, K\} \times \mathbb{S}_i$$

which gives a plane number and the next state, according to the incoming cell destination, the current state, and the result of a coin-toss that is taken out of a finite and uniform coin-space COINSPACE. *(For a deterministic algorithm $|\texttt{COINSPACE}| = 1$.)*

$E_{PPS}(\texttt{ALG}, \sigma, T)$ is the *execution* of the PPS using demultiplexing algorithm ALG in response to incoming traffic T, and coin-tosses sequence σ; namely, the cells in T and the planes they are sent through: $\{\langle c, plane(c)\rangle \mid c \in T\}$.

A state $s \in \mathbb{S}_i$ is *reachable* if there is a sequence of coin tosses σ and a traffic T, such that the state-machine reaches state s in execution $E_{PPS}(\texttt{ALG}, \sigma, T)$. A switch *configuration* consists of the states of all state-machines, and the contents of all the buffers in the switch at a given time. A configuration is *reachable* if it is reached in an execution of the switch. Since the switch does not have a predetermined initial configuration, we assume that for every pair of reachable configurations C_1, C_2, there is a finite incoming traffic that causes the switch to transit from C_1 to C_2.

The internal lines of the switch operate at a lower rate $r < R$. For simplicity, we assume that $r' \triangleq \frac{R}{r} = \lceil \frac{R}{r} \rceil$. This lower rate r enforces an *input constraint* on the demultiplexing algorithm [18]: For any two cells c_1, c_2 such that $orig(c_1) = orig(c_2)$ and $plane(c_1) = plane(c_2)$, $|ta(c_1) - ta(c_2)| > r'$. Since the PPS has no buffers in its input-ports, cells are immediately sent to one of the planes; that is, a cell c traverses the internal link between $orig(c)$ and $plane(c)$ at time $ta(c)$ (see Figure 2).

We assume that both the planes and output buffers are FCFS and work-conserving. Let $tp(c)$ be the time-slot cell c leaves $plane(c)$, and denote $tl_{PPS}(c)$ the time-slot it leaves the PPS. The lower rate of the internal links between the planes to the output ports induces an *output-constraint* [18]: For every two cells $c_1, c_2 \in T$, if $dest(c_1) = dest(c_2)$ and $plane(c_1) = plane(c_2)$ then $|tp(c_1) - tp(c_2)| > r'$. To neglect delays caused by the additional stage of the PPS, a cell can leave the PPS at the same time-slot it arrives at the output-

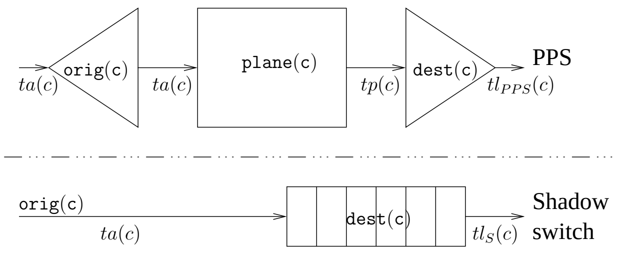


Figure 2: Illustration of the time-points associated with a cell c.

port, provided that no other cell is leaving at this time-slot, i.e., $tl_{PPS}(c) \geq tp(c)$. Note however that $tp(c) \geq ta(c) + 1$.

The PPS is compared to a work-conserving shadow switch that receives the same traffic T, and obeys per-flow FCFS discipline. We denote the time a cell $c \in T$ leaves the shadow switch by $tl_S(c)$, and the execution of the shadow switch in response to traffic T by $E_S(T)$. Note that $tl_S(c) \geq ta(c)+1$.

The *relative queuing delay* of a cell c under a certain demultiplexing algorithm ALG and a coin-tosses sequence σ is $\mathcal{R}(\texttt{ALG}, \sigma, c) = tl_{PPS}(c) - tl_S(c)$.

Definition 2.2. *For traffic T, demultiplexing algorithm ALG and coin-tosses sequence σ, the* maximum relative queuing delay *is $\mathcal{R}_{max}(ALG, \sigma, T) = \max_{c \in T}\{\mathcal{R}(ALG, \sigma, c)\}$, and the* average relative queuing delay *is $\mathcal{R}_{avg}(ALG, \sigma, T) = \frac{1}{|T|}\sum_{c \in T} \mathcal{R}(ALG, \sigma, c)$.*

3. LOWER BOUNDS ON THE AVERAGE RELATIVE QUEUING DELAY

In this section we prove that the *average* relative queuing delay asymptotically equals the maximum relative queuing delay, even if randomization is used.

We first show lower bounds that use an adaptive adversary, which sends cells to the switch at each time-slot based on the algorithm actions at previous slots. Then, we show that under certain assumptions the lower bounds can be extended to hold with an oblivious adversary, which must choose the entire traffic in advance, knowing only the demultiplexing algorithm [5].

The maximum and average relative queuing delay of algorithm ALG against an adaptive adversary are denoted $\mathcal{R}_{max}(\texttt{ALG})$ and $\mathcal{R}_{avg}(\texttt{ALG})$, respectively.

We prove yet stronger results, and show that the lower bounds hold even for traffics that are restricted by the (R, B)-*leaky-bucket* model. This model restricts the traffic from flooding the switches by requiring that the combined rate of flows sharing the same input-port or the same output-port does not exceed the external rate R of that port by more than a fixed bound B, which is independent of time [7].

A key observation is that if the last cell of a traffic attains a relative queuing delay $\mathcal{R}$, then this traffic can be continued so that every added cell attains *at least* the relative queuing delay $\mathcal{R}$, regardless of the results of the coin-tosses:

Lemma 3.1. *For any demultiplexing algorithm ALG, coin-tosses sequence σ, and finite traffic T whose last cell is c_1, if c_2 is a cell with origin $orig(c_1)$ and destination $dest(c_1)$ then $\mathcal{R}(ALG, \sigma\theta, c_2) \geq \mathcal{R}(ALG, \sigma, c_1)$ for any coin-toss θ.*

Proof. Since the shadow switch is per-flow FCFS and work-conserving, and because cell c_2 is in the flow from $orig(c_1)$ to $dest(c_1)$ and arrives immediately after cell c_1, cell c_2 leaves the shadow switch exactly at time-slot $tl_S(c_1) + 1$. Since the PPS is also per-flow FCFS, cell c_2 leaves the PPS after cell c_1, thus $tl_{PPS}(c_2) \geq tl_{PPS}(c_1) + 1$. Hence,

$$\begin{aligned}\mathcal{R}(\texttt{ALG}, \sigma\theta, c_2) &\geq tl_{PPS}(c_2) - tl_S(c_2)\\ &\geq (tl_{PPS}(c_1) + 1) - (tl_S(c_1) + 1)\\ &= \mathcal{R}(\texttt{ALG}, \sigma, c_1)\end{aligned}$$

□

An adaptive adversary can construct a traffic that exhibits an average relative queuing delay that matches the maximum relative queuing delay. It first waits for a cell c that attains $\mathcal{R}_{max}$ and then sends many cells (whose number depends on the number of cells that arrived before c) from $orig(c)$ to $dest(c)$, one cell at each time-slot. By repeatedly applying Lemma 3.1 we get:

Lemma 3.2. *If there is a demultiplexing algorithm ALG, a coin-tosses sequence σ and a finite traffic T whose last cell c has $\mathcal{R}(ALG, \sigma, c) = \mathcal{R}_{max}(ALG)$, then $\mathcal{R}_{avg}(ALG) = \mathcal{R}_{max}(ALG) - \varepsilon$, where ε can be made arbitrarily small.*

Proof. Let ℓ be the number of cells in traffic T. Applying Lemma 3.1 $\left\lceil \ell \frac{\mathcal{R}_{max}(\texttt{ALG}) - \varepsilon}{\varepsilon} \right\rceil$ times constructs a traffic $T \circ T'$ such that for every cell b in T' and any coin-tosses sequence σ_b, $\mathcal{R}(\texttt{ALG}, \sigma\sigma_b, b) = \mathcal{R}(\texttt{ALG}, \sigma, c) = \mathcal{R}_{max}(\texttt{ALG})$. Hence,

$$\begin{aligned}\mathcal{R}_{avg}(\texttt{ALG}) &\geq \frac{1}{\ell + \left\lceil \ell \frac{\mathcal{R}_{max}(\texttt{ALG}) - \varepsilon}{\varepsilon} \right\rceil} \left\lceil \ell \frac{\mathcal{R}_{max}(\texttt{ALG}) - \varepsilon}{\varepsilon} \right\rceil \mathcal{R}_{max}(\texttt{ALG})\\ &\geq \mathcal{R}_{max}(\texttt{ALG}) - \varepsilon\end{aligned}$$

□

High relative queuing delay is exhibited when cells that are supposed to leave the shadow switch one after the other, are concentrated in a single plane. An execution $E_{PPS}(\texttt{ALG}, \sigma, T)$ is (f, s)-*concentrating* if there is a time-slot t and output-port j such that:

1. Output-port j's buffer of the shadow switch is empty at time-slot t;
2. At least f cells destined for output-port j arrive to the switch during time-interval $[t, t+s)$, and f out of these cells are sent through the same plane; and
3. Traffic T ends at time-slot $t + s$.

The following lemma bounds the relative queuing delay exhibited in these executions, extending [3, Lemma 4]:

Lemma 3.3. *For any (R, B) leaky-bucket traffic T, coin-tosses sequence σ, and (f, s)-concentrating execution $E_{PPS}(ALG, \sigma, T)$ for output-port j and plane k, the last cell c that is sent from plane k to output-port j in $E_{PPS}(ALG, \sigma, T)$ attains $\mathcal{R}(ALG, \sigma, c) \geq f \cdot r' - (s + B)$.*

Proof. We compare the queuing delay of the cells in the PPS and in the shadow switch. Since the shadow switch is work-conserving, all f cells leave the switch exactly f time-slots after the first cell is dispatched. On the other hand, a PPS completes this execution after at least fr' time-slots, because f cells are sent to the same plane, and only one cell can be sent from this plane to the output-port every r' time-slots. Let c be the last of these cells sent from the plane to the output-port. Hence, the relative queuing delay c attains is at least $fr' - f$ time-slots. Since the incoming traffic is (R, B) leaky-bucket, $f \leq s + B$, and therefore $\mathcal{R}(\texttt{ALG}, \sigma, c) \geq fr' - f \geq fr' - (s + B)$ time-slots. $\square$

For every traffic T we examine the probability $\Pr_\sigma [E_{PPS}(\texttt{ALG}, \sigma, T)$ is (f, s)-concentrating$]$, taken over all coin-tosses sequences σ, that the execution of $\texttt{ALG}$ given T and σ is (f, s) concentrating.

The second observation is that if there is traffic T such that its execution is (f, s)-concentrating with small but non-negligible probability, an adaptive adversary can construct another execution that is *almost always* (f, s)-concentrating:

LEMMA 3.4. *If from every configuration C there is an (R, B) leaky-bucket traffic T such that $\Pr_\sigma [E_{PPS}(\texttt{ALG}, \sigma, T)$ is (f, s)-concentrating$] \geq \widetilde{p} > 0$, then an adaptive adversary can construct an (R, B) leaky-bucket traffic T' from C such that $\Pr_\sigma [E_{PPS}(\texttt{ALG}, \sigma, T')$ is (f, s)-concentrating$] \geq 1 - \delta$, where δ can be made arbitrarily small.*

PROOF. Fix a configuration C; the adaptive adversary constructs the executions from C iteratively: Denote $C^0 \triangleq C$. Let C^i be the configuration just before iteration $i \geq 0$, and denote by T^i the traffic such that from configuration C^i, $\Pr_\sigma \left[E_{PPS}(\texttt{ALG}, \sigma, T^i) \text{ is } (f, s)\text{-concentrating}\right] \geq \widetilde{p}$. The adversary stops if the last execution is indeed an (f, s)-concentrating. Otherwise, it concatenates an empty traffic of B time-slots (denoted T_e) and continues to the next iteration.

Since in each iteration the adversary stops with probability of at least $\widetilde{p}$ independently of previous iterations, it stops with (f, s) concentrating execution before iteration $\lceil \log_{1-\widetilde{p}} \delta \rceil$ with probability $1 - \delta$. Let ℓ be the iteration the adversary stops. Since there are B empty time-slots between the arrival of the last cell of traffic T^i and the arrival of the first cell in T^{i+1}, $T' = T^0 \circ T_e \circ \ldots \circ T_e \circ T^\ell$ has burstiness factor B, and its corresponding execution starting from C is (f, s)-concentrating with probability $1 - \delta$. $\square$

The next corollary follows immediately from Lemmas 3.1–3.4:

COROLLARY 3.5. *If from every configuration C there is an (R, B) leaky-bucket traffic T such that $\Pr_\sigma [E_{PPS}(\texttt{ALG}, \sigma, T)$ is (f, s)-concentrating$] \geq \widetilde{p} > 0$, then with probability $1 - \delta$, $\mathcal{R}_{avg}(\texttt{ALG}) \geq f \cdot r' - (s + B) - \varepsilon$, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small*

A *fully-distributed demultiplexing algorithm* [3] demultiplexes a cell, arriving at time-slot t, according to the input-port's local information in time interval $[0, t]$. The relative queuing delay of a PPS with fully-distributed demultiplexing algorithm strongly depends on the number of input-ports that can send a cell, destined for the same output-port, through the same plane:

DEFINITION 3.1. *A demultiplexing algorithm is* d-partitioned *if there is a plane k, an output-port j, and a set of input-ports I, such that $|I| \geq d$ and the following property holds: For every input-port $i \in I$ and state $s_i \in \mathbb{S}_i$, if at least n_i cells destined for output-port j arrive at input-port i after it is in state s_i, then with probability $p_i > 0$, i sends at least one cell destined for output-port j through plane k.*

It is possible to construct a traffic with no bursts that causes a deterministic d-partitioned fully distributed demultiplexing algorithm to concentrate d cells in a single plane during a time-interval of d time-slots [3, Theorem 6]. Under the randomized generalization of d-partitioning, it is clear that the same traffic results in a (d, d)-concentrating execution with probability of at least $\prod_{i=1}^{N} p_i > 0$, and therefore Corollary 3.5 implies the following lower bound:

THEOREM 3.6. *Any randomized d-partitioned fully distributed demultiplexing algorithm $\texttt{ALG}$ has, with probability $1 - \delta$, $\mathcal{R}_{max}(\texttt{ALG}) \geq d(r' - 1)$ time-slots and $\mathcal{R}_{avg}(\texttt{ALG}) \geq d(r' - 1) - \varepsilon$ time-slots, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small.*

The FTD algorithm (Section 5) can easily be made d-partitioned to provide a matching upper bound.

Another interesting class includes u *real-time distributed (u-RT)* demultiplexing algorithms [3], which demultiplex a cell arriving at time-slot t, according to the input-port's local information in time interval $[0, t]$, and to the switch's global information in time interval $[0, t - u]$. An input-port state transition may depend on other input-ports' state transitions, and on incoming flows to other input-ports, as long as they occurred more than u time-slots earlier.

Let $\overline{u} = \min\{u, \frac{r'}{2}\}$, the minimum between the lag in gathering global information, and the rate of the planes relative to the external rate. The next theorem, which is based on Corollary 3.5 and some observations from [3], gives a lower bound on the average relative queuing delay of u-RT demultiplexing algorithms:

THEOREM 3.7. *Any randomized u-RT demultiplexing algorithm $\texttt{ALG}$ has, with probability $1 - \delta$, $\mathcal{R}_{max}(\texttt{ALG}) \geq \frac{\overline{u}N}{S}(1 - \frac{\overline{u}}{r'})$ time-slots and $\mathcal{R}_{avg}(\texttt{ALG}) \geq \frac{\overline{u}N}{S}(1 - \frac{\overline{u}}{r'}) - \varepsilon$ time-slots, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small.*

PROOF. Consider an arbitrary configuration C. Denote by t_0 the time-slot in which the PPS is in configuration C, by x_0 the number of cells arrived to the PPS until time-slot t_0, and by n_0 the number of cells stored in one of the PPS' buffers at time-slot t_0.

Consider now the empty traffic T_e, in which no cells arrive to the switch at all. Denote by C_1 the switch configuration at time-slot $t_1 = t_0 + n_0 + \frac{\overline{u}Nx_0}{S} + 1$. If there are still cells stored in one of the buffers at time-slot t_1, then these cells have relative queuing delay of at least $\frac{\overline{u}Nx_0}{S} + 1$ time-slots; therefore the average relative queuing delay is more than $\frac{\overline{u}N}{S}$ time-slots, and the claim follows.

Otherwise, all the buffers are empty in configuration C_1. Fix an output-port j, and consider the traffic $\overline{T}$ in which cells destined for j arrive simultaneously to all input-ports at each time-slot in the interval $[t_1, t_1 + \overline{u})$. Note that $\overline{T}$ is an $(R, \overline{u}N - \overline{u})$ leaky-bucket traffic, since for any $\tau \geq 1$ and time-interval $[t, t + \tau)$, the total number of cells arriving to the switch is bounded by $\tau + (\overline{u}N - \overline{u})$.

Since $\overline{u} \leq \frac{1}{2}\frac{R}{r} \leq \frac{R}{r}$, the input constraint implies that two cells that arrive to the same input-port are not sent through the same plane. Hence, for every coin-tosses sequence σ, there is a plane k used by a set I of at least $\frac{\overline{u}}{K}N$ input-ports in the execution $E_{PPS}(\texttt{ALG}, \sigma, \overline{T})$; note that since the PPS speedup is at least 1, $\frac{\overline{u}}{K}N < \frac{R}{rK}N \leq N$.

For every input-port $i \in I$, let $c_i \in \overline{T}|_i$ be the cell such that $plane(c_i) = k$. Consider the traffic $T|_i = \{c | c \in$

$\overline{T}|_i$ and $ta(c) \leq ta(c_i)\}$; that is $T|_i$ is the traffic comprised of the cells in $\overline{T}|_i$ that arrived to the switch until cell c_i.

Now consider traffic $T|_I = \bigcup_{i \in I} T|_i$, which is the parallel composition of traffics $T|_i$. Note that both $T|_I$ and $T_e \circ T|_I$ are $(R, \overline{u}^2 \frac{N}{K} - \overline{u})$ leaky-bucket traffics.

For every input-port $i \in I$, $ta(c_i) \leq t_1 + \overline{u} < t_1 + u$, which implies that i does not use global information on the switch status after time-slot t_1. Hence, the executions $E_{PPS}(\texttt{ALG}, \sigma, \overline{T})$ and $E_{PPS}(\texttt{ALG}, \sigma, T|_I)$ are equivalent. Therefore, with probability of at least $\prod_{i \in I} \left(\frac{1}{|\texttt{COINSPACE}|}^{|T|_i|} \right) \geq \left(\frac{1}{|\texttt{COINSPACE}|}^{\overline{u}} \right)^{|I|} \geq \left(\frac{1}{|\texttt{COINSPACE}|}^{\overline{u}} \right)^{\frac{\overline{u}N}{K}} = \widetilde{p} > 0$, taken over the coin-tosses sequences σ, all the input-ports $i \in I$ send their last cell to plane k in $E_{PPS}(\texttt{ALG}, \sigma, T_e \circ T|_I)$ starting at configuration C. Hence, configuration C holds the conditions of Corollary 3.5 and the claim follows. $\square$

We now consider oblivious adversaries, which must choose the entire traffic in advance, knowing only the demultiplexing algorithm ALG [5]: $\mathcal{R}^{obl}_{max}(\texttt{ALG})$ and $\mathcal{R}^{obl}_{avg}(\texttt{ALG})$ denote the maximum and average queuing delay of algorithm ALG against such an adversary. We assume that the PPS and the shadow switch obey a *global FCFS* policy, i.e., cells that share the same output-port should leave the switch in the order of their arrival (ties are broken arbitrarily). Under this assumption, the following variant of Lemma 3.1 holds:

LEMMA 3.8. *If the PPS and the shadow switch obey a global FCFS policy, for any demultiplexing algorithm ALG, coin-tosses sequence σ, and finite traffic T whose last cell is c_1, if c_2 is a cell with destination $dest(c_1)$ then in execution $E_{PPS}(ALG, \sigma\theta, T \circ \{c_2\})$, $\mathcal{R}(ALG, \sigma\theta, c_2) \geq \mathcal{R}(ALG, \sigma, c_1)$ for any coin-toss θ.*

Note that unlike Lemma 3.1, in Lemma 3.8 the cells c_1 and c_2 are required to share only the same destination and not the same origin.

We next extend the lower bound presented in Theorem 3.6 to hold against oblivious adversaries if global FCFS discipline must be provided.

THEOREM 3.9. *Any randomized d-partitioned fully distributed demultiplexing algorithm ALG has, with probability $1 - \delta$, $\mathcal{R}^{obl}_{max}(ALG) \geq d(r' - 1)$ time-slots and $\mathcal{R}^{obl}_{avg}(ALG) \geq d(r' - 1) - \varepsilon$ time-slots, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small.*

PROOF. Given ALG, the adversary pre-computes the set $I = \{i_1, \ldots, i_d\}$ of d input ports, the output-port j and the plane k, and for each input port $i \in I$ the values n_i and p_i, for which the conditions presented in Definition 3.1 hold.

For any input port $i \in I$, let x_i be a value chosen uniformly at random from $\{1, \ldots, n_i\}$. Let T_i be a traffic comprised of x_i cells from input port i to outpout port j, and let c_i be the last cell of T_i. Traffic T' is defined as follows: $T' = (T_{i_1} \setminus \{c_{i_1}\}) \circ \ldots \circ (T_{i_d} \setminus \{c_{i_d}\}) \circ \{c_{i_1}\} \ldots \circ \{c_{i_d}\}$ (see Figure 3).

We first prove that with non-negligible probability, taken over all coin-tosses sequences σ, the execution of ALG on traffic T' is (d, d)-concentrating:

CLAIM. $\Pr_\sigma [E_{PPS}(ALG, \sigma, T')$ *sends the last d cells through plane k*$] \geq \prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} \triangleq \widetilde{p} > 0$.

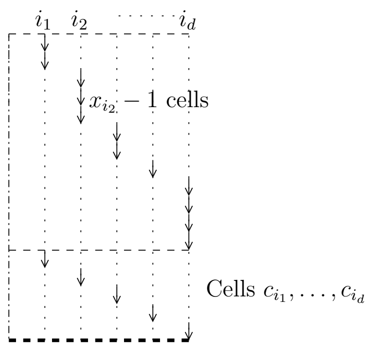


Figure 3: Illustration of traffic T' in the proof of Theorem 3.9.

PROOF. Denote by $\widetilde{T}_i$ the traffic comprised of n_i cells from input port i to output port j. By the definition of n_i, with probability p_i at least one cell in $\widetilde{T}_i$ is sent through plane k. Since x_i is chosen uniformly at random from the values $\{1, \ldots, n_i\}$, this cell is the x_i-th cell (that is, the cell c_i) with probability of at least $\frac{1}{n_i}$. Note that traffic T_i is a prefix of traffic $\widetilde{T}_i$; since the demultiplexor is bufferless, the decision through which plane to send the cell c_i is based only on cells arriving at the switch prior to c_i, which implies that cell c_i is sent through k with probability of at least $\frac{1}{n_i} \cdot p_i$.

In traffic T', for each input port $i \in I$, no cells arrive to input port i between $(T_i \setminus \{c_i\})$ and c_i. Thus, for each input port $i \in I$ and coin-tosses sequence σ, the cells c_i are sent through the same plane in executions $E_{PPS}(\texttt{ALG}, \sigma, T')$ and $E_{PPS}(\texttt{ALG}, \sigma, T_{i_1} \circ \ldots \circ T_{i_d}\})$. Since the demultiplexors are independent, the probablity, taken over all coin-tosses sequences σ, that execution $E_{PPS}(\texttt{ALG}, \sigma, T')$ sends the last d cells through plane k is at least $\prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} \triangleq \widetilde{p} > 0$. $\square$

Let T'' be a concatenation of $\lceil \log_{1-\widetilde{p}} \delta \rceil$ such instances of traffic T' (each with different and independent random choices of the values $x_{i_1}, \ldots x_{i_d}$). Each instance is (d, d)-concentrating with probability $\widetilde{p}$ regardless of the first configuration at the beginning of the instance. Therefore, a (d, d)-concentration occurs in traffic T'' with probability of at least $1 - \delta$. Denote the last d cells of the instance in which a concentration occurs by $c^{m'}_{i_1}, \ldots, c^{m'}_{i_d}$. Since all these cells are sent through plane k, $\mathcal{R}(\texttt{ALG}, \sigma, c^{m'}_{i_d}) \geq d(r' - 1)$, by Lemma 3.3.

Let $\ell = |T''|$, and $\ell' = \ell \left\lceil \frac{d(r'-1)-\varepsilon}{\varepsilon} \right\rceil$. Consider the traffic $\widetilde{T} = \{c_1\} \circ \ldots \circ \{c_{\ell'}\}$, such that each cell c_i is sent from an arbitrary input port i to output port j, and the traffic $T = T'' \circ \widetilde{T}$. During each interval $[ta(c^{m'}_{i_d}), ta(c^{m'}_{i_d}) + t]$, $t > 0$, exactly t cells arrive at the PPS and destined for the same output-port j (i.e., there are no stalls between the cells comprising traffic T). Therefore, Lemma 3.8 implies that for each cell c such that $ta(c) > ta(c^{m'}_{i_d})$, $\mathcal{R}(\texttt{ALG}, \sigma, c) \geq \mathcal{R}(\texttt{ALG}, \sigma, c^{m'}_{i_d}) \geq d(r' - 1)$. Similar to the proof of Lemma 3.2, with probability $1 - \delta$, $\mathcal{R}^{obl}_{avg}(\texttt{ALG}) \geq \mathcal{R}_{avg}(\texttt{ALG}, \sigma, T) \geq d(r' - 1) - \varepsilon$, and $\mathcal{R}^{obl}_{max}(\texttt{ALG}) \geq \mathcal{R}_{max}(\texttt{ALG}, \sigma, T) \geq d(r' - 1)$. $\square$

4. BOUNDING THE RELATIVE QUEUING DELAY

This section presents a methodology for bounding $\mathcal{R}_{max}(\texttt{ALG}, \sigma, T)$ for an arbitrary traffic T and coin-tosses sequence σ. We fix some traffic T and omit the notations $\texttt{ALG}, \sigma$ and T. For simplicity assume T begins after time-slot 0, and that at time-slot 0 (i.e., at "the beginning of time"), no cells arrive at the switch, and therefore all the queues are empty. We also assume the PPS is *global* FCFS.

Cells are queued in a bufferless PPS either within the planes or within the multiplexors residing at the output-ports. A simple situation in which queuing in a multiplexor happens is when the output-port is flooded, but in this case cells also suffer from high queuing delay in the shadow switch, and the relative queuing delay is small. A more complicated situation is when a cell arrives at the multiplexor out of order, and should wait for previous cells to arrive from their planes. In this case, the relative queuing delay is an indirect result of queuing within the other planes (of some preceding cell), as we prove in the following lemma:

LEMMA 4.1. *There is a cell c such that $tl_{PPS}(c) = tp(c)$ and $\mathcal{R}(c) = \mathcal{R}_{max}$.*

PROOF. Let c be the first cell to leave the PPS such that $\mathcal{R}(c) = \mathcal{R}_{max}$. Assume that $tl_{PPS}(c) > tp(c)$; since the multiplexor buffer is work-conserving, in time-slot $tl_{PPS}(c) - 1$ another cell c' leaves the PPS from output-port $dest(c)$. Hence $tl_{PPS}(c') = tl_{PPS}(c) - 1$, and therefore $\mathcal{R}(c') = tl_{PPS}(c') - tl_S(c') = tl_{PPS}(c) - 1 - tl_S(c')$. Since c' leaves the PPS before c and the shadow switch is FCFS, $tl_S(c') \leq tl_S(c) - 1$. Hence the relative queuing delay of c' is $\mathcal{R}(c') \geq tl_{PPS}(c) - tl_S(c) = \mathcal{R}(c) = \mathcal{R}_{max}$, contradicting the minimality of c. $\square$

By the lemma, the maximal relative queuing delay is due to queuing in the planes and not in the multiplexors. This points to the weakness of a previous attempt [19] to bound the relative queuing delay, which calculated only queuing in the multiplexors.

Consider a single cell c, and focus on the queuing within *plane*(c), caused by the lower rate on the link between *plane*(c) and *dest*(c). Since both the PPS and the shadow switch are FCFS, cells arriving at the switch after cell c cannot prohibit c from being transmitted on time. We present an upper bound that depends only on the disproportion of the number of cells sent through *plane*(c) to *dest*(c). Relating this quantity and the queue lengths at time-slot $ta(c)$ is not immediate, since it is possible that the shadow switch is busy while the plane is idle, and vice versa.

Let $A_j(t_1, t_2)$ be the number of cells destined for output-port j that arrive at the switch during time interval $[t_1, t_2]$, and $A_j^k(t_1, t_2)$ be the number of these cells that are sent through plane k. The following definition captures the imbalance between planes:

DEFINITION 4.1. *For a plane k, output-port j and time-slots $0 \leq t_1 \leq t_2$:*

1. *The* imbalance factor during time interval $[t_1, t_2]$ *is* $\Delta_j^k(t_1, t_2) = A_j^k(t_1, t_2) - \frac{1}{r'} A_j(t_1, t_2)$.
2. *The* imbalance factor by time-slot t_2 *is* $\Delta_j^k(t_2) = \max_{t_1 \leq t_2}\{\Delta_j^k(t_1, t_2)\}$.
3. *The* maximum imbalance factor *is* $\Delta_j^k = \max_{t_2}\{\Delta_j^k(t_2)\}$.

It can be shown that the imbalance factor is superadditive; that is, for every output-port j, plane k and time-slots t_1, t_2: $\Delta_j^k(t_2) \geq \Delta_j^k(t_1 - 1) + \Delta_j^k(t_1, t_2)$.

Let $Q_j(t)$ be the size of the j^{th} buffer in the shadow switch after time-slot t; similarly, $Q_j^k(t)$ is the size of j^{th} buffer of plane k of the PPS after time-slot t. Let $L_j^k(t_1, t_2)$ be the number of cells destined for output-port j that leave plane k during time interval $[t_1, t_2]$. Note that $Q_j^k(t) = A_j^k(0, t) - L_j^k(0, t)$.

Time-slot t_1 is the *beginning of a (k, j) busy period* for time-slot $t_2 \geq t_1$, if it is the last time-slot before t_2, such that $Q_j^k(t_1 - 1) = 0$. Since $Q_j^k(t_1) > Q_j^k(t_1 - 1)$, a cell c arrives at the switch at time-slot t_1, and therefore exactly one cell destined for j leaves plane k in time-interval $(t_1 + 1 - r', t_1 + 1]$. This is either cell c itself, or another cell that prohibits c from using the link, and therefore is sent at most r' time-slot before time-slot $t_1 + 1$. Since the queue is never empty until time-slot t_2, one cell is sent to j exactly every r' time-slots after the first cell. This implies that the number of cells sent from k, $L_j^k(t_1, t_2) \geq \left\lfloor \frac{(t_2 - t_1) + 1}{r'} \right\rfloor$.

The following lemma bounds how badly a plane can perform relative to the shadow switch, by comparing their busy periods:

LEMMA 4.2. *If $Q_j(t - 1) = 0$ then for every plane k, $Q_j^k(t - 1) \leq \max\{0, \lceil \Delta_j^k(t - 1) \rceil\}$.*

PROOF. The claim follows immediately if Q_j^k is empty after time-slot $t - 1$. Otherwise, let t' be the beginning of a (k, j) busy period for time-slot $t - 1$. During time interval $[t', t - 1]$ plane k sends a cell to output j every r' time-slots, therefore:

$$L_j^k(t', t - 1) \geq \left\lfloor \frac{t - t'}{r'} \right\rfloor \tag{1}$$

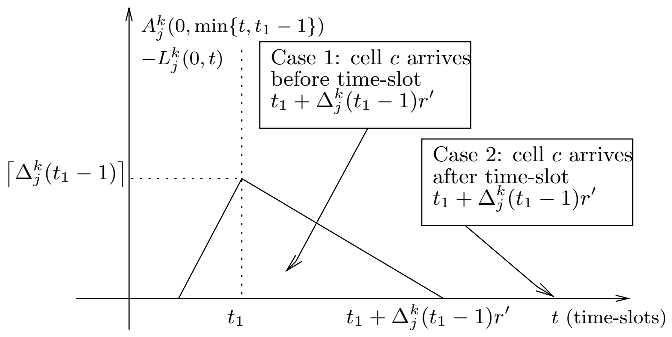
$Q_j(t - 1) = 0$ implies that for every time-slot $\ell \leq t - 1$, $A_j(\ell, t - 1) \leq t - \ell$ (otherwise the j^{th} buffer of the shadow switch is not empty after time-slot $t - 1$). In particular:

$$A_j(t', t - 1) \leq t - t' \tag{2}$$

Using these inequalities we bound $Q_j^k(t - 1)$:

$$\begin{aligned}
Q_j^k(t-1) &= A_j^k(0, t - 1) - L_j^k(0, t - 1) \\
&= \left(A_j^k(0, t' - 1) + A_j^k(t', t - 1)\right) - \\
&\quad \left(L_j^k(0, t' - 1) + L_j^k(t', t - 1)\right) \\
&= A_j^k(t', t - 1) - L_j^k(t', t - 1) \\
&\qquad \text{since } A_j^k(0, t'-1) - L_j^k(0, t'-1) = Q_j^k(t'-1) = 0 \\
&\leq A_j^k(t', t - 1) - \left\lceil \frac{t - t'}{r'} \right\rceil \qquad \text{by (1)} \\
&\leq A_j^k(t', t - 1) - \left\lceil \frac{A_j(t', t - 1)}{r'} \right\rceil \qquad \text{by (2)} \\
&= A_j^k(t', t - 1) + \left\lceil \Delta_j^k(t', t-1) - A_j^k(t', t-1) \right\rceil \\
&\leq \left\lceil \Delta_j^k(t - 1) \right\rceil
\end{aligned}$$

$\square$



$A_j^k(0,\min\{t,t_1-1\})$
$-L_j^k(0,t)$
Case 1: cell c arrives before time-slot $t_1+\Delta_j^k(t_1-1)r'$
Case 2: cell c arrives after time-slot $t_1+\Delta_j^k(t_1-1)r'$
$\lceil\Delta_j^k(t_1-1)\rceil$
t_1
$t_1+\Delta_j^k(t_1-1)r'$
t (time-slots)

Figure 4: Illustration for the different cases in the proof of Theorem 4.3

We complete the proof by bounding the lag between the time a cell leaves its plane and the time it should leave the shadow switch:

THEOREM 4.3. *The maximum relative queuing delay of cells destined for output-port j and sent through plane k is bounded by* $\max\{0, r'(\Delta_j^k+1)+B_j\}$, *where B_j is the maximum number of cells destined for output-port j that arrived at the switch in the same time-slot.*

PROOF. By Lemma 4.1, it suffices to bound $tp(c)-tl_S(c)$ for every cell c. Since $tp(c)-tl_S(c) = (tp(c)-ta(c)) - ((tl_S(c)-ta(c))$, it suffices to bound only the difference between the time a cell spends in the plane (i.e., $tp(c)-ta(c)$) and the time it spends in the shadow switch (i.e., $tl_S(c)-ta(c)$). Since both switches operate under FCFS policy, these values solely depend on the corresponding queues lengths when cell c arrives.

Let t_1 be the earliest time-slot, such that the buffer of output-port j in the shadow switch is never empty during time-interval $[t_1, ta(c)]$; if no such time-slot exists let $t_1 = ta(c)$.

First, we bound $tl_S(c)-ta(c)$ from below. The buffer in the shadow switch is empty at time-slot t_1-1, and then the switch is continuously busy during time-interval $[t_1, ta(c)–1]$, transmitting exactly one cell at each time-slot to output-port j. This implies that $Q_j(ta(c)-1) = A_j(t_1, ta(c)-1) - (ta(c)-t_1)$. All the cells in the queue, should leave the switch after time-slot $ta(c)$ and before $tl_S(c)$, therefore:

$$tl_S(c)-ta(c) > A_j(t_1, ta(c)-1) - (ta(c)-t_1)$$

Since $A_j(ta(c), ta(c)) \le B_j$, and $tl_S(c)-ta(c)$ is an integer, it follows that:

$$tl_S(c)-ta(c) \ge A_j(t_1, ta(c)) - B_j + t_1 - ta(c) + 1 \qquad (3)$$

Recall that by Lemma 4.2, $Q_j^k(t_1–1) \le \max\{0, \lceil\Delta_j^k(t_1–1)\rceil\}$. There are two cases to consider, depending on whether the queue in plane k was or was not drained before cell c's arrival, ignoring cells that arrive to plane k after cell c (see Figure 4):

Case 1. $ta(c) \le t_1 + \Delta_j^k(t_1-1)r'$. Since plane k is FCFS and work-conserving, it transfers every cell in its queue in exactly r' time-slots, except cell c which is considered as transfered in the first time-slot of its transmission:

$$\begin{aligned}
tp(c)-ta(c) &\le r'Q_j^k(ta(c))+1 \\
&\le r'\left(Q_j^k(t_1-1) - \left(\left\lfloor\frac{ta(c)-t_1}{r'}\right\rfloor - A_j^k(t_1, ta(c))\right)\right)+1 \\
&\qquad \text{since } Q_j^k \text{ is not drained} \\
&\le r'\left(\left\lceil\Delta_j^k(t_1-1)\right\rceil - \left\lfloor\frac{ta(c)-t_1}{r'}\right\rfloor + A_j^k(t_1, ta(c))\right)+1 \qquad \text{by Lemma 4.2} \\
&\le r'\Delta_j^k(t_1-1) - ta(c) + t_1 + 2r' + A_j(t_1, ta(c)) + r'\Delta_j^k(t_1, ta(c)) + 1 \\
&\le A_j(t_1, ta(c)) + r'\left(\Delta_j^k(ta(c))+2\right) - ta(c) + t_1 + 1 \\
&\qquad \text{since } \Delta \text{ is superadditive} \quad (4)
\end{aligned}$$

By (4) and (3), $tp(c)-tl_S(c) \le r'(\Delta_j^k(ta(c))+2)+B_j$. More fine-grained calculations, which are omitted for brevity, yield that $tp(c)-tl_S(c) \le r'(\Delta_j^k(ta(c))+1)+B_j$, since in this case $\lceil\Delta_j^k(t_1-1)\rceil - \left\lfloor\frac{ta(c)-t_1}{r'}\right\rfloor$ is at most $\Delta_j^k(t_1-1) - \frac{ta(c)-t_1}{r'}+1$.

Case 2. $ta(c) > t_1+\Delta_j^k(t_1-1)r'$. If $Q_j^k(ta(c)) = 0$ then cell c is immediately delivered to the output-port, i.e., $tp(c) = ta(c)+1 \le tl_S(c)$ and the claim holds since $tp(c)-tl_S(c) \le 0$.

If $Q_j^k(ta(c)) > 0$, let t_2 be the beginning of a (k,j) busy period for $ta(c)$. Note that by the choice of t_2, $L_j^k(t_2, ta(c)) \ge \left\lfloor\frac{ta(c)-t_2+1}{r'}\right\rfloor$. Hence, we have:

$$\begin{aligned}
tp(c)-ta(c) &\le r'Q_j^k(ta(c))+1 \\
&= r'\left(A_j^k(t_2, ta(c)) - L_j^k(t_2, ta(c))\right)+1 \\
&\qquad \text{since } Q_j^k(t_2-1)=0 \\
&\le r'\left(A_j^k(t_2, ta(c)) - \left\lceil\frac{ta(c)-(t_2-1)}{r'}\right\rceil\right)+1 \\
&\qquad \text{since plane } k \text{ is continuously busy} \\
&\le A_j(t_2, ta(c)) + r'\Delta_j^k(t_2, ta(c)) - r'\left\lceil\frac{ta(c)-(t_2-1)}{r'}\right\rceil+1 \\
&\le A_j(t_1, ta(c)) + r'\left(\Delta_j^k(t_2, ta(c))+1\right) + t_1 - ta(c) + (t_2-t_1) - A_j(t_1, t_2-1)+1 \quad (5)
\end{aligned}$$

By the choice of t_1, the output-buffer of the shadow switch is empty at time-slot t_1-1, and not during time-interval $[t_1, t_2-1]$. This implies that $(t_2-t_1) \le A_j(t_1, t_2-1)$, and therefore (5) implies:

$$tp(c)-ta(c) \le A_j(t_1, ta(c)) + r'(\Delta_j^k(ta(c))+1) + t_1 - ta(c) + 1 \qquad (6)$$

By (6) and (3), $tp(c)-tl_S(c) \le r'(\Delta_j^k(ta(c))+1)+B_j$. □

5. DEMULTIPLEXING ALGORITHMS WITH OPTIMAL RQD

The *fractional traffic dispatch algorithm (FTD)* [19, 22] is an example of a fully-distributed demultiplexing algorithm. In FTD, there is a *window* of size r' time-slots that slides over the sequence of cells in each flow (i, j). The algorithm maintains window constraint that ensures that two cells, which are in the same window, are not sent through the same plane. An equivalent variation of the algorithm divides each flow to static blocks of size r' [19, 22].

The demultiplexing algorithm chooses the plane to which a cell is sent through arbitrarily from the set of planes that do not violate the window constraint and the input-constraint described at Section 2. A speedup of $S \geq 2$ suffices for the algorithm to work correctly [19].

A simple use of Theorem 4.3 and the fact that always $B_j \leq N$ shows:

THEOREM 5.1. $\mathcal{R}_{avg}(FTD) \leq \mathcal{R}_{max}(FTD) \leq (N+1)r'$.

PROOF. Let $A_{i\to j}(t_1, t_2)$ be the number of cells in flow (i, j) that arrive at the switch during time-interval $[t_1, t_2]$, and $A^k_{i\to j}(t_1, t_2)$ be the number of these cells that are sent through plane k.

Due to the window constraint on each flow and because $A_{i\to j}$, r' are integers, for every output-port j plane k and time-slots t_1, t_2:

$$\begin{aligned}
\Delta^k_j(t_1,t_2) &= A^k_j(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'} \\
&= \sum_{i=1}^{N} A^k_{i\to j}(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'} \\
&\leq \sum_{i=1}^{N} \left\lceil \frac{A_{i\to j}(t_1,t_2)}{r'} \right\rceil - \frac{A_j(t_1,t_2)}{r'} \\
&\leq \sum_{i=1}^{N} \left(\frac{A_{i\to j}(t_1,t_2)}{r'} + \frac{r'-1}{r'} \right) - \frac{A_j(t_1,t_2)}{r'} \\
&= N\frac{r'-1}{r'}
\end{aligned}$$

By Theorem 4.3, $\mathcal{R}_{max}(FTD) \leq (N+1)r'$, since always $B_j \leq N$. □

We now show a 1-RT demultiplexing algorithm that matches the lower bounds presented in Section 3. Informally, Algorithm 1 divides the set of planes into two equal-size sets (V_0, V_1), and its operations with respect to cells destined for a specific output-port into two phases. In each phase, the algorithm sends cells destined for a specific output-port through different set of planes (i.e., V_0 or V_1). After every time-slot each input-port collects the global information of the switch, and uses it to calculate the imbalance factor for each plane k and each output-port j. Then, in the next time-slot each input-port sends a cell to output-port j only through planes with low (or zero) imbalance factor. A phase ends when there are no balanced planes to use.

To avoid situations in which all the input-ports send cells through the same plane, we divide the input-ports into $\frac{N}{r'}$ sets of size r', and assure that under no circumstances two input-ports in the same set send a cell destined for the same output-port through the same plane. This is done by calculating the actions of other input-ports in the same set as if they indeed get a cell destined for the same output-port.

Algorithm 1 1-RT Algorithm

```
Constants:
    $V_0=\{1,\ldots,\frac{K}{2}\}$; $V_1=\{\frac{K}{2}+1,\ldots,K\}$

Shared:
    $F[N]$: $N$ sets of planes, initially all $V_0$    ▷ Cells for $j$
        can be sent only through $F[j]$
    $R[N][r']$: matrix of values in $\{1,\ldots,K,\perp\}$, initially
        all $\perp$                     ▷ holding input-constraints
    $t$: value in $\{0,\ldots,r'-1\}$, initially 0  ▷ Cyclic pointer
        to matrix R
    $Q[N], L[N]$: $N$ sets of planes, initially all $\emptyset$
    $M[N]$: $N$ sets of planes, initially all $\{1,\ldots,K\}$
    $phase[N]$: vector of values in $\{0,1\}$, initially all 0

At the beginning of each time-slot:
    For every $j \in \{1,\ldots,N\}$: CALCULATE($j$)
    For every $j \in \{1,\ldots,N\}$: $F[j] \leftarrow$ UPDATE($j$)
    Update the matrix $R[N][r']$ according to global infor-
        mation
    $t \leftarrow (t+1)$ mod $r'$

 1: int procedure DISPATCH(cell $c$) at demutliplexor $i$
 2:     $j \leftarrow dest(c)$
 3:     $p \leftarrow \lfloor \frac{i}{r'} \rfloor$
 4:     set $B \leftarrow \emptyset$
 5:     for $x \leftarrow r'p$ to $i$ do
 6:         $E \leftarrow \{k \in \{1,\ldots,K\} \mid \exists a \in \{0,\ldots,r'-1\}, R[x][a]=k\}$
 7:         $k \leftarrow \min(F[j] \setminus (B \cup E))$
 8:         $B \leftarrow B \cup \{k\}$
 9:     end for
10:     $R[i][t] \leftarrow k$  ▷ can be read by other input-ports only
            in the next time-slot
11:     return $k$
12: end procedure

 1: set procedure UPDATE(int $j$)
 2:     set $S \leftarrow F[j]$
 3:     $Q[j] \leftarrow Q[j] \setminus M[j]$
 4:     if $Q[j] = \emptyset$ then                ▷ change phase
 5:         $Q[j] \leftarrow \{1,\ldots,K\} \setminus M[j]$
 6:         $phase[j] \leftarrow (1 - phase[j])$
 7:         $S \leftarrow V_{phase[j]}$
 8:     else
 9:         $S \leftarrow S \setminus (L[j])$
10:     end if
11:     return $S$
12: end procedure

 1: void procedure CALCULATE(int $j$)
 2:     set $A \leftarrow \{k \mid \Delta^k_j(t) > \frac{N}{r'}\}$  ▷ using global information
 3:     $M[j] \leftarrow \{k \mid \Delta^k_j(t) \leq 0\}$     ▷ using global information
 4:     $L[j] \leftarrow (L[j] \cup A) \setminus M[j]$
 5: end procedure
```

Before proving the correctness of Algorithm 1, we first show two properties of this algorithm.

The first lemma shows that the imbalance factor between each plane and each output-port is bounded under this algorithm:

LEMMA 5.2. *In Algorithm 1, for every plane $k \in V_0 \cup V_1$ and output-port j, $\Delta_j^k < \frac{2N}{r'}$.*

PROOF. Clearly, if $\Delta_j^k(t_3) > \frac{N}{r'}$ then $k \in L[j]$ in the beginning of time-slot $t_3 + 1$ (procedure CALCULATE, line 4). Therefore $k \notin F[j]$ in the beginning of time-slot $t_3 + 1$ (procedure UPDATE, line 9)[2], and cells are not sent through plane k until a time-slot ${t_3}' > t_3 + 1$ in which $\Delta_j^k({t_3}' - 1) \leq 0$.

For every two input-ports i_1, i_2 if $\left\lfloor \frac{i_1}{r'} \right\rfloor = \left\lfloor \frac{i_2}{r'} \right\rfloor$, then i_1 and i_2 do not send cells destined for the same output-port through the same plane in the same time-slot (procedure DISPATCH). This implies that the maximum number of cells destined for the same output-port and sent through the same plane in a single time-slot is $\frac{N}{r'}$.

By Definition 4.1, if a plane does not receive cells destined for output-port j in time-slot t_1 then $\Delta_j^k(t_1) \leq \Delta_j^k(t_1 - 1)$. This implies that there exists a time-slot t_1, in which plane k receives cells destined for j, and $\Delta_j^k(t_1) = \Delta_j^k$. In the worst-case $\Delta_j^k(t_1 - 1) = \frac{N}{r'}$, and k receives $\frac{N}{r'}$ cells destined for j.

Assume towards a contradiction, that $\Delta_j^k(t_1) \geq \frac{2N}{r'}$. Then there is a time-slot t_2 such that $\Delta_j^k(t_2, t_1) \geq \frac{2N}{r'}$. Note that $\Delta_j^k(t_1, t_1) < \frac{N}{r'}$, since $A_j^k(t_1, t_1) \leq \frac{N}{r'}$, and $A_j(t_1, t_1) \geq A_j^k(t_1, t_1)$. This implies that $t_2 < t_1$, and therefore by Definition 4.1:

$$\begin{aligned}
\Delta_j^k(t_2, t_1) &= A_j^k(t_2, t_1) - \frac{1}{r'} A_j(t_2, t_1) \\
&= A_j^k(t_2, t_1 - 1) + A_j^k(t_1, t_1) - \\
&\quad \frac{1}{r'} A_j(t_2, t_1 - 1) - \frac{1}{r'} A_j(t_1, t_1) \\
&= \Delta_j^k(t_2, t_1 - 1) + \Delta_j^k(t_1, t_1) \\
&< \frac{2N}{r'}
\end{aligned}$$

This contradicts the choice of t_2, and the claim follows. □

The second property is a simple conclusion from Lemma 5.2:

LEMMA 5.3. *If a PPS is operating under Algorithm 1, $2N$ cells destined for output j arrive at it during time-interval $[t_1, t_2]$, and none of them is sent through plane k, then $\Delta_j^k(t_2) \leq 0$.*

PROOF. By Definition 4.1, there is a time-slot t_3 such that $\Delta_j^k(t_2) = \Delta_j^k(t_3, t_2)$.

If $t_3 \geq t_1$ then $\Delta_j^k(t_3, t_2) \leq 0$ since $A_j^k(t_3, t_2) = 0$. Otherwise,

$$\begin{aligned}
\Delta_j^k(t_3, t_2) &= \Delta_j^k(t_3, t_1 - 1) + \Delta_j^k(t_1, t_2) \\
&= \Delta_j^k(t_3, t_1 - 1) + A_j^k(t_1, t_2) - \frac{1}{r'} A_j(t_1, t_2)
\end{aligned}$$

By Lemma 5.2 and Definition 4.1 $\Delta_j^k(t_3, t_1 - 1) \leq \Delta_j^k \leq \frac{2N}{r'}$. Since $A_j^k(t_1, t_2) = 0$ and $A_j(t_1, t_2) \geq 2N$, it follows that $\Delta_j^k(t_3, t_2) \leq 0$ also in this case. □

[2]The claim holds also if the phase changes in the beginning of time-slot t_3+1 since $Q[j] = \emptyset$ at line 4 yields that $V_{phase} \subseteq M[j]$ at line 7.

We conclude by showing that a speedup of 8 suffices for this demultiplexing algorithm to achieve optimal relative queuing delay:

THEOREM 5.4. *Speedup $S = 8$ suffices for Algorithm 1 to work correctly with maximum relative queuing delay of $3N + r'$ time-slots.*

PROOF. In order to prove Theorem 5.4, it suffices to show that every time line 7 of procedure DISPATCH is executed, $F[j] \setminus (B \cup E) \neq \emptyset$, and a plane can be chosen. Clearly, at each step $|B| \leq r'$ and $|E| < r'$; therefore the claim follows if $|F[j]| > 2r'$. Since $F[j]$ is changed only by procedure UPDATE(j), it suffices to show that after any execution of UPDATE(j), $|F[j]| > 2r'$.

Assume, without loss of generality, that *phase* $= 0$ after an execution of procedure UPDATE(j) at time-slot t_1. Assume, by way of contradiction, that $|F[j]| \leq 2r'$ at time-slot t_1. Clearly, from line 7 and the fact that $|V_0| = |V_1| = \frac{K}{2} = \frac{Sr'}{2} = 4r' > 2r'$, it follows that *phase* $= 0$ after time-slot $t_1 - 1$. This implies that $|V_0 \cap L[j]| \geq 2r'$.

Denote by t_2 the last time-slot in which *phase* was changed from 1 to 0 ($t_2 = 0$ if no such time-slot exists). At time-slot t_2, when executing line 4, $Q[j]$ is empty and therefore all planes $k \in V_0$ are at $M[j]$ at time-slot t_2. This implies that for every $k \in V_0$, $\Delta_j^k(t_2) \leq 0$.

Let k be a plane in $V_0 \cap L[j]$. By the definition of $L[j]$ there is a time-slot $t_3 \in [t_2, t_1]$ such that $\Delta_j^k(t_3) > \frac{N}{r'}$. Let t_4 be the last time-slot such that $\Delta_j^k(t_3) = \Delta_j^k(t_4, t_3)$. If $t_4 < t_2$ then

$$\begin{aligned}
\Delta_j^k(t_4, t_3) &= \Delta_j^k(t_4, t_2) + \Delta_j^k(t_2 + 1, t_3) \\
&\leq \Delta_j^k(t_2) + \Delta_j^k(t_2 + 1, t_3) \ \leq \ \Delta_j^k(t_2 + 1, t_3)
\end{aligned}$$

and therefore t_4 is not minimal. Hence $t_4 \geq t_2$ and $[t_4, t_3] \subseteq [t_2, t_1]$. Since $\Delta_j^k(t_4, t_3) = A_j^k(t_4, t_3) - \frac{1}{r'} A_j(t_4, t_3) > \frac{N}{r'}$, and $A_j(t_4, t_3) \geq A_j^k(t_4, t_3)$, it follows that $A_j^k(t_4, t_3) > \frac{N}{r'-1}$.

Because $|V_0 \cap L[j]| \geq 2r'$, then the number of cells arrived at the switch and destined for j during time-interval $[t_2, t_1]$ is at least $(2r')\frac{N}{r'-1} > 2N$. Since during time-interval $[t_2, t_1]$ no cells are sent to any plane in V_1, by Lemma 5.3 all planes in $k \in V_1$ has $\Delta_j^k(t_1) \leq 0$, and in particular all planes in $Q[j]$. This yields that $Q[j]$ becomes empty, and the phase changes at least once during time-interval $[t_2, t_1]$ which contradicts the choice of t_1 and t_2.

Lemma 5.2 and Theorem 4.3 imply that the relative queuing delay of the algorithm is at most $3N + r'$. □

6. DISCUSSION

This paper presents lower bounds on the average relative queuing delay, which hold with high probability even if randomization is used. This generally implies that unlike other load-balancing problems, randomization is not a silver bullet for reducing the relative queuing delay. Our lower bounds rely on the fact that switches are FCFS, but can be generalized to rely on other priority-based policies.

We show that these lower bounds are tight by presenting a methodology for bounding the relative queuing delay. We bound the relative queuing delay of the FTD algorithm [19, 22] by $(N + 1)r'$ time-slots, exactly matching the lower bound for fully-distributed algorithms. We also design a 1-RT demultiplexing algorithm with relative queuing delay of $3N + r' < 4N$ time-slots; for the common case of constant speedup, this asymptotically matches the lower bound of $\frac{N}{S}\left(1-\frac{1}{r'}\right)$ time-slots for 1-RT algorithms.

A natural design choice is to transmit global control information on the internal links of the PPS, at rate r. Substituting $u \approx r'$ in our lower bounds for u-RT algorithms yields relative queuing delay approximately equal to the delay when no information is gathered at all. This casts a doubt on the benefit of sharing information.

Acknowledgments. We would like to thank Ilan Newman and Fabian Kuhn for helpful discussions about randomized demultiplexing algorithms.

7. REFERENCES

[1] 3Com Corporation. *Inverse Multiplexing over ATM (IMA): A Breakthrough WAN Technology for Corporate Networks*, 1997. http://www.mcoecn.org/WhitePapers/3COM-Inverse-Multiplexing-ATM.pdf.

[2] The ATM Forum. *Inverse Multiplexing for ATM (IMA) specification*, March 1999. Version 1.1, AF-PHY-0086.001.

[3] H. Attiya and D. Hay. The inherent queuing delay of parallel packet switches. In *the 3rd IFIP International Conference on Theoretical Computer Science (TCS)*, pages 139–152, 2004. Revue paper appeared at *the 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2004.

[4] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal of Computing*, 29(1):180–200, 1999.

[5] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in online algorithms. *Algorithmica*, 11(1):2–14, 1994.

[6] C.S. Chang, D.S. Lee, and Y.S. Jou. Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering. *Computer Communications*, 25:611–622, 2002.

[7] A. Charny. *Providing QoS guarantees in input buffered crossbar switches with speedup*. PhD thesis, Massachusetts Institute Of Technology, September 1998.

[8] A. Charny, P. Krishna, N.S. Patel, and R.J. Simcoe. Algorithms for providing bandwidth and delay guarantees in iput-buffered crossbars with speedup. In *Sixth IEEE/IFIP International Workshop on Quality of Service*, pages 235–244, 1998.

[9] F.M. Chiussi, D.A. Khotimsky, and S. Krishnan. Generalized inverse multiplexing of switched ATM connections. In *IEEE Globecom*, 1998.

[10] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input output queued switch. In *IEEE INFOCOM*, pages 1169–1178, 1999.

[11] Cisco Systems, Inc. *ATM Switch Router Software Configuration Guide*, 2001. 12.1(6)EY.

[12] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, pages 406–424, 1953.

[13] R. L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, pages 114–31, 1991.

[14] J. Duncanson. Inverse multiplexing. *IEEE Communications Magazine*, 32(4):34–41, April 1994.

[15] P. Fredette. The past, present, and future of inverse multiplexing. *IEEE Communications Magazine*, 32(4):42–46, April 1994.

[16] P. Giaccone, B. Prabhakar, and D. Shah. Randomized scheduling algorithms for high-aggregate bandwidth switches. *IEEE Journal on Selected Area in Communications*, 21(4):546–559, May 2003.

[17] G. Gonnet. Expected length of the longest probe sequence in hash coding searching. *Journal of the ACM*, 28(2):289–304, April 1981.

[18] S. Iyer, A.A. Awadallah, and N. McKeown. Analysis of a packet switch with memories running slower than the line rate. In *IEEE INFOCOM*, pages 529–537, 2000.

[19] S. Iyer and N. McKeown. Making parallel packet switches practical. In *IEEE INFOCOM*, pages 1680–1687, 2001.

[20] I. Keslassy. *Load Balanced Router*. PhD thesis, Stanford University, June 2004.

[21] I. Keslassy and N. McKeown. Maintaining packet order in two-stage switches. In *INFOCOM*, 2002.

[22] D. Khotimsky and S. Krishnan. Stability analysis of a parallel packet switch with bufferless input demultiplexors. In *IEEE International Conference on Communications (ICC)*, pages 100–106, 2001.

[23] L. Kleinrock. *Queuing Systems, Volume II.* John Wiley&Sons, 1975.

[24] P. Krishna, N.S. Patel, A. Charny, and R.J. Simcoe. On the speedup required for work-conserving crossbar switches. *IEEE Journal on Selected Areas in Communications*, 17(6):1057–1066, June 1999.

[25] Lucent Technologies. *Inverse Multiplexing for ATM, Expanding the Revenue Opportunities for Converged Services over ATM*, 2001.

[26] M.D. Mitzenmacher. On the analysis of randomized load balancing schemes. *Theory of Computer Systems*, 32:361–386, 1999.

[27] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[28] PMC-Sierra, Inc. *Inverse multiplexing over ATM works today*, 2002. http://www.electronicstalk.com/news/pmc/pmc121.html.

[29] B. Prabhakar and N. McKowen. On the speedup required for combined input and output queued switching. *Automatica*, 35(12):1909–1920, December 1999.

[30] D.C. Sthephens and H. Zhang. Implementing distributed packet fair queueing in a scalable architecture. In *IEEE INFOCOM*, pages 282–290, 1998.

[31] I. Stoica and H. Zhang. Exact emulation of an output queueing switch by a combined input output queueing switch. In *Sixth IEEE/IFIP International Workshop on Quality of Service*, pages 218–224, 1998.

[32] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *INFOCOM*, pages 533–539, 1998.