SIAM J. COMPUT.
Vol. 37, No. 5, pp. 1613–1636

RANDOMIZATION DOES NOT REDUCE THE AVERAGE DELAY IN PARALLEL PACKET SWITCHES*

HAGIT ATTIYA† AND DAVID HAY†

Abstract. Switching cells in parallel is a common approach to building switches with very high external line rates and a large number of ports. A prime example is the *parallel packet switch* (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 $r < R$. In order to utilize the parallelism of the PPS, a key issue is to balance the load among the planes; since *randomization* is known as a successful paradigm to solve load balancing problems, it is tempting to design randomized demultiplexing algorithms that balance the load on the average. This paper presents lower bounds on the *average* queuing delay introduced by the PPS relative to an optimal work-conserving first-come first-serve (FCFS) switch for *randomized* demultiplexing algorithms that do not have full and immediate information about the switch status. These lower bounds are shown to be asymptotically optimal through a methodology for analyzing the *maximal* relative queuing delay by measuring the imbalance between the middle stage switches; clearly, this also bounds (from above) the *average* relative queuing delay. The methodology is used to devise new algorithms that rely on slightly outdated global information on the switch status.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* algorithm [S. Iyer and N. McKeown, in *Proceedings of IEEE INFOCOM*, IEEE Communications Society, New York, NY, 2001, pp. 1680–1687; D. Khotimsky and S. Krishnan, in *Proceedings of the IEEE International Conference on Communications*, IEEE Communications Society, New York, NY, 2001, pp. 100–106]. These optimal algorithms are deterministic, proving that randomization does not reduce the relative queuing delay of the PPS.

Key words. load balancing, randomization, packet switching, Clos networks, queuing delay, inverse multiplexing

AMS subject classifications. 68W10, 68W15, 68W20

DOI. 10.1137/050648250

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 the load on the average [4, 29]: 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 numbers of ports [6, 35, 34, 33, 22]. A successful example of such a switch is the *parallel packet switch (PPS)* [19], which is the core of several contemporary switches (e.g., [1, 11, 28, 31]). 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).

A PPS is evaluated by the *queuing delay* it introduces *relative* to an *optimal work-conserving shadow* switch that receives the same incoming traffic. A work-conserving switch guarantees that an output-port is never idle unnecessarily and, by

*Received by the editors December 22, 2005; accepted for publication (in revised form) July 11, 2007; published electronically February 14, 2008. Some of the results appeared, in a preliminary form, in *Proceedings of the* 17*th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA* 2005*)*, pp. 11–20.

http://www.siam.org/journals/sicomp/37-5/64825.html

†Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (hagit@cs.technion.ac.il, hdavid@cs.technion.ac.il).

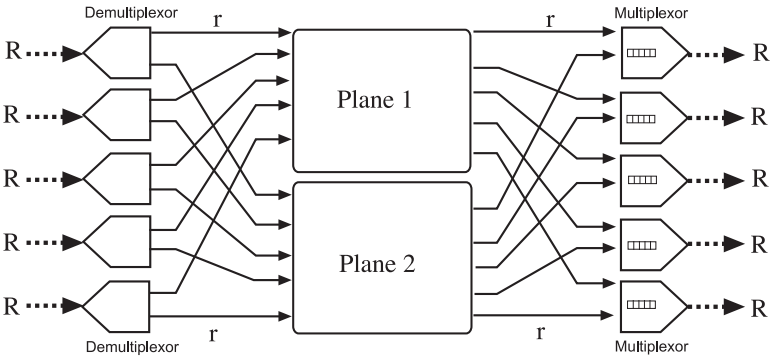

FIG. 1. *A* 5×5 *PPS with two planes in its center stage, without buffers in the input-ports.*

that, maximizes the switch throughput and minimizes its average cell delay [10, 26, 27]. This generalizes the common practice of comparing with an output-queued switch [8, 19, 20, 25, 27, 32, 36, 37]. This comparison is not burdened by unsubstantiated probabilistic assumptions on the incoming traffic [13] and reveals inherent strengths and weaknesses of the PPS architecture.

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 [4, 17, 29] and in other high-bandwidth switches [16, 38], 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 result holds due to the requirement that switches should not mis-sequence cells [23]. This property allows an adversary to exploit a transient increase of the relative queuing delay and perpetuate it sufficiently to increase the *average* relative queuing delay.

Specifically, we show that an adversary can 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 demultiplexing, the type of the adversary, and the exact restriction on the order of cells the switch should respect. The bounds are equal to the lower bounds known for maximum relative queuing delay [3]: If a PPS respects the arrival order of cells with the same input-port and the same output-port and the adversary is *adaptive* [30], 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 use only local information. The latter lower bound also holds with an *oblivious* adversary [5] if a PPS obeys a *global* first-come first-serve (FCFS) policy (that is, all cells to the same destination should leave the switch according to their arrival order).

To prove that these bounds are tight, we devise a 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.

Our methodology is employed to prove that the maximal relative queuing delay of the *fractional traffic dispatch (FTD)* algorithm [20] is $O(N\frac{R}{r})$ time-slots. This matches the lower bound on the average relative queuing delay introduced by fully

distributed demultiplexing algorithms (even when randomization can be used). This is the first formal and complete correctness proof for this algorithm. Iyer and McKeown [21, 20] outline an approach for bounding the relative queuing delay of FTD but leave a number of details missing [18]; a previous attempt [25] to complete the formal proof and precisely bound the relative queuing delay of FTD turned out to be flawed [24].

By precisely capturing the crucial factors affecting the relative queuing delay, our methodology leads to new algorithms that use global information that is u time-slot old. Their maximum relative queuing delay is $O(N)$ time-slots, asymptotically matching the lower bound on the average relative queuing delay for this class of demultiplexing algorithms (even when randomization can be used).

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, such that no two cells arrive at the same input-port at the same time-slot. 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 $\{c \in T \mid orig(c) \in I\}$. Since for any input-port i and traffic T, there are no two cells $c_1, c_2 \in T|_i$ such that $ta(c_1) = ta(c_2)$, the arrival times of cells in $T|_i$ induce a total order on them.

For any cell c, $\text{shift}(c, t)$ 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 finite 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 input-ports on one side and to all 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*

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

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,* $|\texttt{COINSPACE}| = 1$*.)*

$\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$ is the *execution* of a PPS using demultiplexing algorithm ALG in response to incoming traffic T and coin-toss sequence σ; for all cells in T, the execution indicates the planes the cells are sent through: $\{\langle c, plane(c, T)\rangle \mid c \in T\}$.

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 $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$. 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

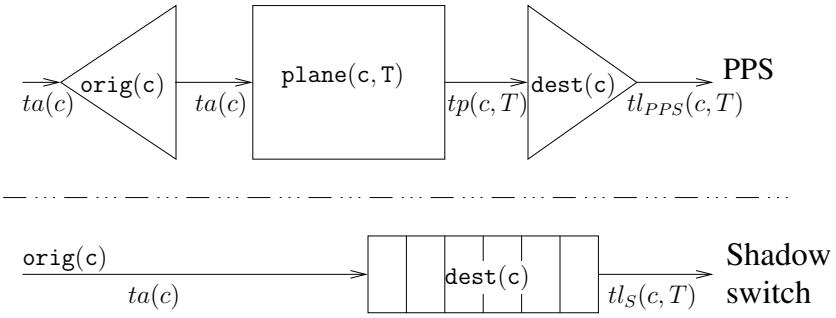

FIG. 2. *Time-points associated with a cell* $c \in T$.

is a finite incoming traffic that causes the switch a transition from C_1 to C_2.

The internal lines of the switch operate at rate $r < R$. For simplicity, we assume that $r' \triangleq \frac{R}{r} = \lceil \frac{R}{r} \rceil$. This lower rate r imposes an *input constraint* on the demultiplexing algorithm [19].

For any two cells c_1, c_2 in traffic T such that $orig(c_1) = orig(c_2)$ and $plane(c_1, T) = plane(c_2, T)$, $|ta(c_1) - ta(c_2)| > r'$.

Since a 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 $plane(c, T)$ at time $ta(c)$ (see Figure 2).

We assume that both the planes and the output buffers are FCFS and work-conserving. Let $tp(c, T)$ be the time-slot in which a cell $c \in T$ leaves $plane(c, T)$, and denote $tl_{PPS}(c, T)$ the time-slot it leaves the PPS. The lower rate of the internal links between the planes to the output-ports imposes an *output-constraint* [19].

For every two cells $c_1, c_2 \in T$, if $dest(c_1) = dest(c_2)$ and $plane(c_1, T) = plane(c_2, T)$ then $|tp(c_1, T) - tp(c_2, T)| > r'$.

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

A PPS is compared to a work-conserving shadow switch that receives the same traffic T and obeys the *per-flow FCFS* discipline; that is, cells with the same origin and the same destination should leave the switch in their arrival order. We denote the execution of the shadow switch in response to traffic T by $\mathcal{E}_S(T)$ and the time a cell $c \in T$ leaves the shadow switch by $tl_S(c, T)$. Note that $tl_S(c, T) \geq ta(c) + 1$.

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

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

The maximum and the average relative queuing delays of an algorithm ALG *against an adversary* A *are denoted* $\mathcal{R}^A_{max}(\texttt{ALG})$ *and* $\mathcal{R}^A_{avg}(\texttt{ALG})$*, respectively.*

When it is clear from the context, we omit the traffic T from the notation $plane(c, T)$, $tp(c, T)$, $tl_{PPS}(c, T)$, $tl_S(c, T)$, and $\mathcal{R}(\texttt{ALG}, \sigma, c, T)$.

3. Lower bounds on the average relative queuing delay. In this section, we prove lower bounds on the *average* relative queuing delay even when randomization is used. Our first lower bounds use an adaptive adversary that sends cells to the switch

at each time-slot based on the algorithm actions at previous slots. Then, we show that, under reasonable assumptions, the lower bounds can be extended to hold with an oblivious adversary [5] that chooses the entire traffic in advance, knowing only the demultiplexing algorithm.

We prove yet stronger results and show that the lower bounds hold even when the traffic is restricted by the (R, B)-*leaky-bucket* model [7, 13]. 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, called the burstiness factor of the traffic, which is independent of time. Specifically, a traffic T is (R, B)-leaky-bucket if, for any two time-slots $t_1 \le t_2$ and any output-port j, $|\{c \in T \mid t_1 \le ta(c) \le t_2 \text{ and } dest(c) = j\}| \le (t_2 - t_1) + B$.

3.1. General techniques and observations. A key observation is that if the last cell of a traffic attains relative queuing delay $\mathcal{R}$, then this traffic can be continued so that every added cell attains *at least* relative queuing delay $\mathcal{R}$, regardless of the random choices made by the demultiplexing algorithm.

We first define how a traffic is continued. A cell $c_2 \in T$ is the *immediate successor* of cell $c_1 \in T$ in demultiplexing algorithm $\mathtt{ALG}$, denoted $c_2 = succ(c_1, T)$, if $tl_S(c_2, T) = tl_S(c_1, T) + 1$, and for every coin tosses sequence σ, $tl_{PPS}(c_2, T) > tl_{PPS}(c_1, T)$ in the execution $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T)$. Namely, a PPS cannot change the order in which c_1 and c_2 are delivered; this happens, for example, when a PPS follows a per-flow FCFS policy and c_1, c_2 share the same input-port and the same output-port.

Let c be the last cell in a traffic T; i.e., $tl_S(c, T) = \max_{c' \in T}\{tl_S(c', T)\}$. A traffic $T' = \{c_0, \ldots, c_n\}$ is a *proper continuation* of T if, in the execution of the shadow switch in response to traffic $T \circ T'$, all the cells of T' are delivered one time-slot after the other without any stalls, and the delivery times of the cells of T remain unchanged. Formally, T' is a proper continuation of T if, in execution $\mathcal{E}_S(T \circ T')$, $c_0 = succ(c, T \circ T')$, $c_i = succ(c_{i-1}, T \circ T')$ for every i, and for every $c' \in T$, $tl_S(c', T) = tl_S(c', T \circ T')$ and $tl_{PPS}(c', T) = tl_{PPS}(c', T \circ T')$.

We first examine proper continuations by a single cell.

LEMMA 3.1. *For any demultiplexing algorithm* $\mathtt{ALG}$*, coin-toss sequence* σ*, and finite traffic* T*, if* c_1 *is the last cell of* T*, and* $T' = \{c_2\}$ *is a proper continuation of* T*, then* $\mathcal{R}(\mathtt{ALG}, \sigma\theta, c_2, T \circ T') \ge \mathcal{R}(\mathtt{ALG}, \sigma, c_1, T)$ *for any coin-toss* θ*.*

Proof. Since T' is a proper continuation of T, cell c_2 leaves the shadow switch exactly at time-slot $tl_S(c_1, T \circ T') + 1$, and in addition $tl_{PPS}(c_2, T \circ T') \ge tl_{PPS}(c_1, T \circ T') + 1$. Hence,

$$\begin{aligned}\mathcal{R}(\mathtt{ALG}, \sigma\theta, c_2, T \circ T') &\ge tl_{PPS}(c_2, T \circ T') - tl_S(c_2, T \circ T') \\ &\ge (tl_{PPS}(c_1, T \circ T') + 1) - (tl_S(c_1, T \circ T') + 1) \\ &= tl_{PPS}(c_1, T) - tl_S(c_1, T) \;=\; \mathcal{R}(\mathtt{ALG}, \sigma, c_1, T). \qquad \square\end{aligned}$$

If the adversary can construct, for every traffic, a proper continuation that is arbitrarily long, then it can construct a traffic that exhibits an average relative queuing delay that matches the maximum relative queuing delay. Intuitively, the adversary waits for a cell c that attains $\mathcal{R}_{max}$ and then sends many cells, which form a proper continuation (whose length depends on the number of cells that arrived before c).

LEMMA 3.2. *Fix an adversary* A*, demultiplexing algorithm* $\mathtt{ALG}$*, a coin-toss sequence* σ*, and a finite traffic* T *whose last cell* c *has* $\mathcal{R}(\mathtt{ALG}, \sigma, c, T) = x$*. If the adversary* A *can construct a proper continuation of traffic* T*, whose size is at least* $\left\lceil |T| \frac{x-\varepsilon}{\varepsilon} \right\rceil$ *(*ε *is an arbitrarily small constant), then* $\mathcal{R}^A_{avg}(\mathtt{ALG}) \ge x - \varepsilon$*.*

Proof. Let ℓ be the number of cells in traffic T, and let T' be a proper continuation of T such that $|T'| = \left\lceil \ell \frac{x-\varepsilon}{\varepsilon} \right\rceil$. Applying Lemma 3.1 $|T'|$ times implies that for every cell b in T' and any coin-toss sequence σ_b, $\mathcal{R}(\mathtt{ALG}, \sigma\sigma_b, b) \geq \mathcal{R}(\mathtt{ALG}, \sigma, c) \geq x$. Hence,

$$\mathcal{R}^A_{avg}(\mathtt{ALG}) \;\geq\; \frac{1}{\ell + \left\lceil \ell \frac{x-\varepsilon}{\varepsilon} \right\rceil} \left\lceil \ell \frac{x-\varepsilon}{\varepsilon} \right\rceil x \;\geq\; x - \varepsilon. \qquad \square$$

The specific construction of the proper continuation depends on the type of the adversary. Lemma 3.4 and Step 2 of Theorem 3.9 show such constructions.

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 $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T)$ is (f, s)-*concentrating for output-port* j *and plane* k if there is a time-slot t 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 plane k; and
3. traffic T ends at time-slot $t + s$.

We call an execution an (f, s)-*concentrating* execution when the plane k and the output-port j are clear from the context.

The following lemma bounds the relative queuing delay exhibited in (f, s)-concentrating executions, extending [3, Lemma 4] for randomized demultiplexing algorithms.

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

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 that c attains is at least $fr' - f$ time-slots. Since the incoming traffic is (R, B)-leaky-bucket, $f \leq s + B$, and therefore $\mathcal{R}(\mathtt{ALG}, \sigma, c, T) \geq fr' - f \geq fr' - (s + B)$ time-slots. $\square$

We concentrate now on an adaptive adversary, denoted *adp*, which sends cells to the switch based on the algorithm actions.

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

Another key observation is that if there is traffic T such that its execution is (f, s)-concentrating with small but nonnegligible 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 $\Pr_\sigma[\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T)$ 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 $\Pr_\sigma[\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T')$ 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,

$\Pr_\sigma\left[\mathcal{E}_{PPS}(\texttt{ALG},\sigma,T^i) \text{ is } (f,s)\text{-concentrating}\right] \geq \widetilde{p}$. The adversary stops if the last execution is indeed (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 at least $\widetilde{p}$ independently of previous iterations, it stops with an (f,s)-concentrating execution at iteration $\ell \leq \left\lceil \log_{1-\widetilde{p}} \delta \right\rceil$ with probability $1-\delta$. 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}, $T' = T^0 \circ T_e \circ \cdots \circ T_e \circ T^\ell$ has burstiness factor B, and its corresponding execution starting from C is (f,s)-concentrating with probability $1-\delta$. ∎

Since both the shadow switch and the PPS are per-flow FCFS, an adaptive adversary can always construct an arbitrarily long proper continuation of some traffic T. Therefore, we have the following lemma.

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

Proof. By Lemma 3.4, an adaptive adversary can construct a traffic T' from configuration C, such that $\Pr_\sigma\left[\mathcal{E}_{PPS}(\texttt{ALG},\sigma,T') \text{ is } (f,s)\text{-concentrating}\right] \geq 1-\delta$.

Let c be the last cell of T'. Lemma 3.3 implies that with probability $1-\delta$, the relative queuing delay of c is at least $f \cdot r' - (s+B)$.

The adaptive adversary continues with traffic T'', which consists of $\left\lceil |T'| \frac{f \cdot r' - (s+B) - \varepsilon}{\varepsilon} \right\rceil$ cells from $orig(c)$ to $dest(c)$, one cell at each time-slot. T'' is a proper continuation of traffic T', because both the PPS and the shadow switch obey a per-flow FCFS policy and all cells in T'' share the same input-port and the same output-port.

Hence, Lemma 3.2 implies that $\mathcal{R}^{adp}_{avg}(\texttt{ALG}) \geq f \cdot r' - (s+B) - \varepsilon$, with probability $1-\delta$. ∎

3.2. Lower bound for fully distributed algorithms with an adaptive adversary. 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 $[0,t]$. Since no information is shared between input-ports, we assume that the state $s_i \in \mathbb{S}_i$ of demultiplexor i does not change, unless a cell arrives at input-port i. Note that demultiplexing algorithms that change their state even without receiving a cell are not considered fully distributed, because a common clock-tick is shared among all input-ports. (Such algorithms are covered in section 3.3.)

The relative queuing delay of a PPS with a 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. The following definition captures this switch characteristic.

DEFINITION 3.6. *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.*

We later show that a static partition of the planes among the demultiplexors may reduce the relative queuing delay. However, since such partitioning is failure-prone, most existing fully distributed algorithms are N-partitioned, meaning that each demultiplexor may use each plane in order to send cells to each output-port. All our results hold for this class of algorithms by substituting $d = N$.

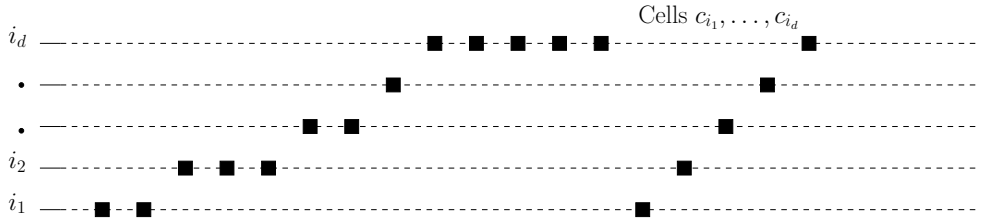

FIG. 3. *Illustration of traffic T in the proof of Theorem* 3.7.

We next prove a lower bound for d-partitioned fully distributed demultiplexing algorithms by showing that it is possible to construct a traffic with no bursts that causes, with nonnegligible probability, the algorithm to concentrate d cells in a single plane during a time interval of d time-slots; the proof follows the lower bound for the deterministic case [3, Theorem 6].

THEOREM 3.7. *Any randomized d-partitioned fully distributed demultiplexing algorithm* ALG *has, with probability $1-\delta$, $\mathcal{R}^{adp}_{max}(\texttt{ALG}) \geq d(r'-1)$ time-slots and $\mathcal{R}^{adp}_{avg}(\texttt{ALG}) \geq d(r'-1)-\varepsilon$ time-slots, where $\varepsilon > 0$ and $\delta > 0$ can be made arbitrarily small.*

Proof. Given ALG, the adversary 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 $p_i > 0$, for which the conditions presented in Definition 3.6 hold.

Fix a configuration C, and for every $i \in I$, let T'_i be a traffic consisting of n_i cells destined for output-port j that arrive one after the other to input-port i. By the definition of n_i, with probability at least p_i, there is at least one cell in T'_i that is sent through plane k. Let c_i be the first such cell; it follows that $\Pr_\sigma[\text{in } \mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T'_i) \text{ cell } c_i \text{ is sent through plane } k] \geq p_i$. Let T_i be the prefix of T'_i that ends with cell c_i; that is, $T_i = \{c \in T'_i | ta(c) \leq ta(c_i)\}$. Since the probability to send c_i through plane k in execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T'_i)$ depends only on cells that arrive at the switch before cell c_i, it follows that for prefix T_i, $\Pr_\sigma[\text{in } \mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T_i) \text{ cell } c_i \text{ is sent through plane } k] \geq p_i$.

Traffic T is defined as follows: $T = (T_{i_1} \setminus \{c_{i_1}\}) \circ \cdots \circ (T_{i_d} \setminus \{c_{i_d}\}) \circ \{c_{i_1}\} \cdots \circ \{c_{i_d}\}$ (see Figure 3). We next show that, with nonnegligible probability, taken over all coin-toss sequences σ, all cells $c_{i_1} \ldots c_{i_d}$ are sent through plane k in the execution of ALG on traffic T.

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-toss sequence σ, $plane(c_i, T) = plane(c_i, T_{i_1} \circ \cdots \circ T_{i_d}\}$. Since the demultiplexors are independent, the probability, taken over all coin-toss sequences σ that the last d cells are sent through plane k in execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$ is at least $\prod_{a=1}^{d} p_{i_a} > 0$.

This implies that execution $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T)$ is (d,d)-concentrating with nonnegligible probability. Since T has no bursts, the claim follows immediately from Lemma 3.5. ∎

3.3. Lower bound for u-RT algorithms with an adaptive adversary. 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 $[0, t-u]$. In this manner, 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. A prominent example of 1-RT demultiplexing algorithms (that is, with $u = 1$) are demultiplexing algorithms that share only a common clock-tick among input-ports. Note that a 1-RT demultiplexing algorithm may change its state even if no cell arrives at its input-port.

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

THEOREM 3.8. *Any randomized u-RT demultiplexing algorithm* ALG *has, with probability* $1-\delta$, $\mathcal{R}^{adp}_{max}(\texttt{ALG}) \geq \frac{\overline{u}N}{S}(1-\frac{\overline{u}}{r'})$ *time-slots and* $\mathcal{R}^{adp}_{avg}(\texttt{ALG}) \geq \frac{\overline{u}N}{S}(1-\frac{\overline{u}}{r'})-\varepsilon$ *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 that arrived at the PPS until time-slot t_0, and by n_0 the number of cells stored in one of the PPS's buffers at time-slot t_0.

Consider now the empty traffic T_e, in which no cells arrive at the switch at all. We first argue that if T_e is long enough, all the buffers of the switch become empty. Specifically, 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 theorem follows.

Assume now that 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 at the switch is bounded by $\tau + (\overline{u}N - \overline{u})$.

Since $\overline{u} \leq \frac{1}{2}\frac{R}{r} < \frac{R}{r}$, the input constraint implies that two cells arriving at the same input-port are not sent through the same plane. Hence, for every coin-toss 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 $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, \overline{T})$; note that since a 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 a cell such that $plane(c_i, \overline{T}|_i) = k$. Consider the traffic $T|_i = \{c | c \in \overline{T}|_i \text{ and } ta(c) \leq ta(c_i)\}$; that is, $T|_i$ consists of the cells in $\overline{T}|_i$ that arrive at the switch before cell c_i.

Consider the parallel composition of traffics $T|_i$, $T|_I = \bigcup_{i \in I} T|_i$ (see Figure 4). 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 $i \in I$, $ta(c_i) < t_1 + \overline{u} \leq t_1 + u$, which implies that input-port i does not have global information on the switch status after time-slot t_1. Hence,

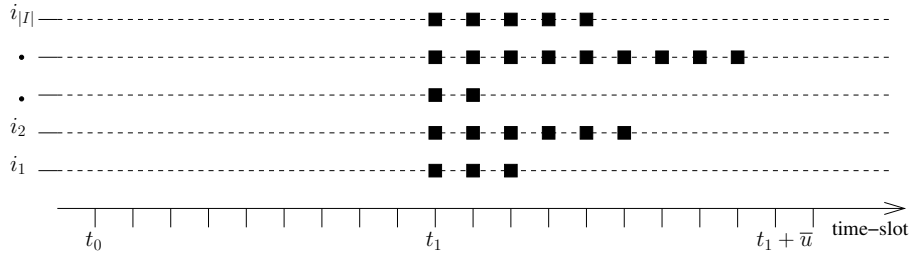

FIG. 4. *Illustration of traffic* $T_e \circ T|_I$ *in the proof of Theorem* 3.8.

the executions $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, \overline{T})$ and $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T|_I)$ are equivalent. Therefore, with probability of at least

$$\prod_{i \in I} \left(\frac{1}{|\texttt{COINSPACE}|}^{|T|_i|} \right) \geq \left(\frac{1}{|\texttt{COINSPACE}|}^{\overline{u}} \right)^{|I|} \geq \left(\frac{1}{|\texttt{COINSPACE}|}^{\overline{u}} \right)^{\frac{\overline{u}N}{K}} > 0,$$

taken over the coin-toss sequences σ, all the input-ports $i \in I$ send their last cell to plane k in $\mathcal{E}_{PPS}(\texttt{ALG}, \sigma, T_e \circ T|_I)$, starting at configuration C. Hence, configuration C satisfies the conditions of Lemma 3.5, and the claim follows. □

3.4. Lower bound for fully distributed algorithms with an oblivious adversary. We now consider oblivious adversaries, *obl*, that 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 (with ties broken arbitrarily). Unlike per-flow FCFS policy, global FCFS policy requires cells to leave in order even if they do not share the same origin.

We next extend Theorem 3.7 to hold with an oblivious adversary, under a global FCFS discipline.

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

Proof. Given ALG, the adversary precomputes the set $I = \{i_1, \ldots, i_d\}$ of d input-ports, the output-port j, the plane k, and for each input-port $i \in I$ the values n_i and p_i, for which the conditions of Definition 3.6 hold. Let $\widetilde{p} \triangleq \prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} > 0$.

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 consisting of x_i cells from input-port i to output-port j, and let c_i be the last cell of T_i. Traffic T' is defined as follows: $T' = (T_{i_1} \setminus \{c_{i_1}\}) \circ \cdots \circ (T_{i_d} \setminus \{c_{i_d}\}) \circ \{c_{i_1}\} \cdots \circ \{c_{i_d}\}$. Note that traffic T' is similar to traffic T in the proof of Theorem 3.7 (illustrated in Figure 3).

Using traffic T', the adversary constructs a traffic T (illustrated in Figure 5) whose average relative queuing delay is at least $d(r'-1) - \varepsilon$ time-slots with probability $1-\delta$ (the constants $\delta, \varepsilon > 0$ can be made arbitrarily small). The construction has two steps.

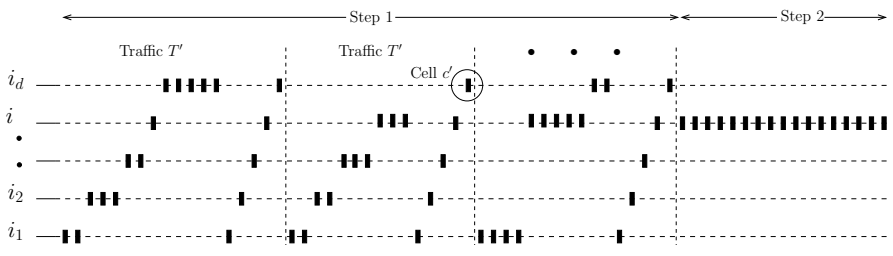

FIG. 5. *Illustration of traffic T in the proof of Theorem* 3.9.

Step 1. Concatenate $\lceil \log_{1-\widetilde{p}} \delta \rceil$ instances of traffic T'. For each instance, choose independently and uniformly at random the values for x_{i_m}, $1 \leq m \leq d$, from $\{1, \ldots n_{i_m}\}$. Let ℓ be the total size of these instances.

Step 2. Concatenate a traffic of size $\ell \lceil \frac{d(r'-1)-\varepsilon}{\varepsilon} \rceil$ cells, such that each cell is sent from an arbitrary input-port i to output-port j.

We first prove that with nonnegligible probability, taken over all coin-toss sequences σ, the execution of ALG on any instance of traffic $T' = (T_{i_1} \setminus \{c_{i_1}\}) \circ \cdots \circ (T_{i_d} \setminus \{c_{i_d}\}) \circ \{c_{i_1}\} \cdots \circ \{c_{i_d}\}$ is (d,d)-concentrating, regardless of the initial configuration.

CLAIM 3.10. $\Pr_\sigma [\mathcal{E}_{PPS}(\mathit{ALG}, \sigma, T')$ *sends the last* d *cells through plane* $k] \geq \prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} \triangleq \widetilde{p} > 0$.

Proof of claim. For any input-port i, denote by $\widetilde{T}_i$ the traffic consisting 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_ith cell (that is, the cell c_i) with probability 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 of 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 at input-port i between $T_i \setminus \{c_i\}$ and c_i. Thus, for each input-port $i \in I$ and coin-toss sequence σ, $\mathit{plane}(c_i, T') = \mathit{plane}(c_i, T_{i_1} \circ \cdots \circ T_{i_d})$ Since the demultiplexors are independent, the probability, taken over all coin-toss sequences σ, that execution $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T')$ sends the last d cells through plane k is at least $\prod_{a=1}^{d} \frac{p_{i_a}}{n_{i_a}} \triangleq \widetilde{p} > 0$. ∎

In Step 1, the random choices of the T' instances are independent. Therefore, (d,d)-concentration occurs at least once in Step 1 with probability at least $1 - \delta$. Let c' be last cell of the first instance in which (d,d)-concentration occurs and T_1 be the traffic $\{c \in T \mid ta(c) \leq ta(c')\}$. Since $\mathcal{E}_{PPS}(\mathtt{ALG}, \sigma, T_1)$ is (d,d)-concentrating, Lemma 3.3 implies that $\mathcal{R}(\mathtt{ALG}, \sigma, c', T_1) \geq d(r'-1)$.

Let $T_2 = T \setminus T_1$. We next show that T_2 is a proper continuation of T_1. Intuitively, this is due to the fact that the switches are work-conserving with FCFS policy and during each interval of size τ, exactly τ cells arrive at the switch and are destined for the same output-port j (i.e., there are no stalls between cells in traffic $T = T_1 \circ T_2$).

Formally, consider two cells $c_1, c_2 \in T$ such that $ta(c_2) = ta(c_1) + 1$. The FCFS policy implies that $tl_S(c_2, T) > tl_S(c_1, T)$ and $tl_{PPS}(c_2, T) > tl_{PPS}(c_1, T)$. In addition, by the construction of traffic T, there is no cell $c_3 \in T$ such that $ta(c_1) \leq ta(c_3) \leq ta(c_2)$. Therefore, the FCFS policy and the work-conservation of the shadow switch imply that $tl_S(c_2, T) = tl_S(c_1, T) + 1$. Hence, for every two cells $c_1, c_2 \in T$, if $ta(c_2) = ta(c_1) + 1$, then $c_2 = \mathit{succ}(c_1, T)$; in particular, the first cell of T_2 is the successor of cell c'. Moreover, since the switches follow an FCFS policy, cells of traffic T_2 do not prohibit cells of traffic T_1 from being delivered on time; namely, for any cell $c \in T_1$, $tl_S(c, T_1) = tl_S(c, T_1 \circ T_2)$ and $tl_{PPS}(c, T_1) = tl_{PPS}(c, T_2)$.

Since $\mathcal{R}(\mathtt{ALG}, \sigma, c', T_1) \geq d(r'-1)$ and $|T_2| \geq \lceil |T_1| \frac{d(r'-1)-\varepsilon}{\varepsilon} \rceil$, Lemma 3.2 implies that $\mathcal{R}^{obl}_{avg}(\mathtt{ALG}) \geq d(r'-1) - \varepsilon$ and $\mathcal{R}^{obl}_{max}(\mathtt{ALG}) \geq d(r'-1)$, with probability $1-\delta$. ∎

The question of whether the lower bound for the u-RT demultiplexing algorithm (described in Theorem 3.8) can be extended to hold with an oblivious adversary is left open. The proof technique described in this section will most likely fail to provide such an extension, since the worst-case traffics that are used in order to prove lower bounds for u-RT algorithms have bursts. Unfortunately, the burstiness accumulates

when concatenating bursty traffics, unless there is a gap of a certain number of time-slots in which no cells arrive at the overloaded output-port. Large bursts may justify a high queuing delay of cells and hence result in low *relative* queuing delay. On the other hand, a gap in which no cells arrive at the overloaded output reduces the relative queuing delay of cells that arrive immediately after it. This implies that the adversary should identify the concentration and then choose to continue the traffic without a gap (as in Lemma 3.2).

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-toss sequence σ. We fix some traffic T and omit the notation $\texttt{ALG}, \sigma$, and T. For simplicity assume the execution 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. Our analysis depends on the realistic assumption that the PPS obeys the *global* FCFS policy.

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)—the relative queuing delay of the waiting cell is at most the relative queuing delay of some preceding cell that was queued only in the planes. This observation is captured 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. ∎

Consider a single cell c, and focus on the queuing within $plane(c)$, caused by the lower rate on the link from $plane(c)$ to $dest(c)$. 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 when 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 let $A_j^k(t_1, t_2)$ be the number of these cells that are sent through plane k. The following definition captures the imbalance between planes.

DEFINITION 4.2. *For a plane k, output-port j and time-slots $0 \leq t_1 \leq t_2$ the following are defined.*

1. *The* imbalance of 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 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 *is* $\Delta_j^k = \max_{t_2}\{\Delta_j^k(t_2)\}$.

Clearly, $\Delta_j^k \geq \Delta_j^k(t_2) \geq \Delta_j^k(t_1, t_2)$ for every output-port j, plane k, and time-slots $t_1 > t_2$. In addition, the imbalance is superadditive.

PROPERTY 4.3. *For every output-port j, plane k, and time-slots $t_1 > t_2$,*

$$\Delta_j^k(t_2) \geq \Delta_j^k(t_1 - 1) + \Delta_j^k(t_1, t_2).$$

Proof. By Definition 4.2, there is a time-slot $t_1' \leq t_1$ such that $\Delta_j^k(t_1 - 1) = \Delta_j^k(t_1', t_1 - 1) = A_j^k(t_1', t_1 - 1) - \frac{1}{r'} A_j(t_1', t_1 - 1)$. Since $\Delta_j^k(t_1, t_2) = A_j^k(t_1, t_2) - \frac{1}{r'} A_j(t_1, t_2)$, we have

$$\begin{aligned}\Delta_j^k(t_1, t_2) + \Delta_j^k(t_1 - 1) &= A_j^k(t_1', t_1 - 1) + A_j^k(t_1, t_2) - \tfrac{1}{r'}(A_j(t_1', t_1 - 1) + A_j(t_1, t_2)) \\ &= \Delta_j^k(t_1', t_2) \;\leq\; \Delta_j^k(t_2). \qquad \square\end{aligned}$$

Let $Q_j(t)$ be the size of the jth buffer in the shadow switch after time-slot t; similarly, $Q_j^k(t)$ is the size of jth 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 $Q_j^k(t_1 - 1) = 0$. Note that this expression is well defined because in time-slot 0 all the queues are empty. 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-slots 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, $L_j^k(t_1, t_2) \geq \lfloor \frac{(t_2 - t_1) + 1}{r'} \rfloor$.

Remark 4.1. Khotimsky and Krishnan [25] defined busy periods only with respect to an output-port j. This points to a flaw in their proof, which ignores situations when the optimal shadow switch is busy sending cells to output-port j, while a specific plane in the PPS is idle part of the time [24]. These situations are the main source of difficulty in our proof.

The following lemma, which is illustrated in Figure 6, bounds how badly a plane can perform relative to the shadow switch, by comparing their busy periods.

LEMMA 4.4. *If $Q_j(t - 1) = 0$, then for every plane k and for every $\delta \in \{0, \ldots, \Delta_j^k(t - 1)r'\}$,*

$$L_j^k(0, (t - 1) + \delta) \geq A_j^k(0, t - 1) - \left\lceil \frac{\Delta_j^k(t - 1)r' - \delta}{r'} \right\rceil.$$

Proof. If there is a time-slot $t_1 \in [t - 1, t - 1 + \delta]$ such that $Q_j^k(t_1) = 0$, then by time-slot t_1 no cells destined for j are waiting in plane k. That is, $L_j^k(0, (t - 1) + \delta) \geq L_j^k(0, t_1) = A_j^k(0, t_1) \geq A_j^k(0, t - 1)$, and the lemma follows.

Otherwise, let t_2 be the beginning of a (k, j) busy period for time-slot $(t - 1) + \delta$. During time interval $[t_2, (t - 1) + \delta]$ plane k sends a cell to output j every r' time-slots; therefore,

$$L_j^k(t_2, (t - 1) + \delta) \geq \left\lfloor \frac{t + \delta - t_2}{r'} \right\rfloor. \tag{1}$$

On the other hand, $Q_j(t - 1) = 0$ implies that for every time-slot $t_3 \leq t - 1$, $A_j(t_3, t - 1) \leq t - t_3$; otherwise, the jth buffer of the shadow switch is not empty after time-slot $t - 1$. In particular,

$$A_j(t_2, t - 1) \leq t - t_2. \tag{2}$$

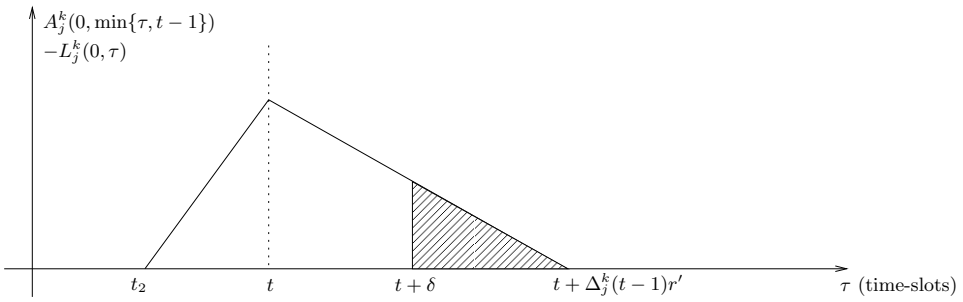

FIG. 6. *Number of cells arriving until time-slot $t-1$ and still queued in plane k by time-slot τ.*

Using these inequalities, we bound $L_j^k(0,(t-1)+\delta)$:

$$\begin{aligned}
L_j^k(0,(t-1)+\delta) &= L_j^k(0,t_2-1) + L_j^k(t_2,(t-1)+\delta) \\
&\geq L_j^k(0,t_2-1) + \left\lfloor \tfrac{t+\delta-t_2}{r'} \right\rfloor && \text{by (1)} \\
&\geq A_j^k(0,t_2-1) + \left\lfloor \tfrac{t+\delta-t_2}{r'} \right\rfloor && \text{since } Q_j^k(t_2-1)=0 \\
&\geq A_j^k(0,t_2-1) + \left\lfloor \tfrac{\delta}{r'} + \tfrac{A_j(t_2,t-1)}{r'} \right\rfloor && \text{by (2)} \\
&= A_j^k(0,t_2-1) + A_j^k(t_2,t-1) + \left\lfloor \tfrac{\delta}{r'} - \Delta_j^k(t_2,t-1) \right\rfloor \\
& && \text{by Definition 4.2} \\
&\geq A_j^k(0,t-1) - \left\lceil \tfrac{r'\Delta_j^k(t-1)+\delta}{r'} \right\rceil . \qquad \square
\end{aligned}$$

By substituting $\delta = 0$ in Lemma 4.4, we get the following corollary, demonstrating the relation between the imbalance and the queue size in the beginning of a busy period.

COROLLARY 4.5. *If $Q_j(t-1) = 0$, then for every plane k,*

$$Q_j^k(t-1) \leq \max\left\{0, \left\lceil \Delta_j^k(t-1) \right\rceil\right\}.$$

We complete the proof by bounding the lag between the time a cell leaves the plane it is sent through and the time it should leave the shadow switch.

THEOREM 4.6. *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 arrive 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, $tp(c) - ta(c)$, and the time it spends in the shadow switch, $tl_S(c) - ta(c)$. Since both switches operate under the FCFS policy, these values depend solely 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

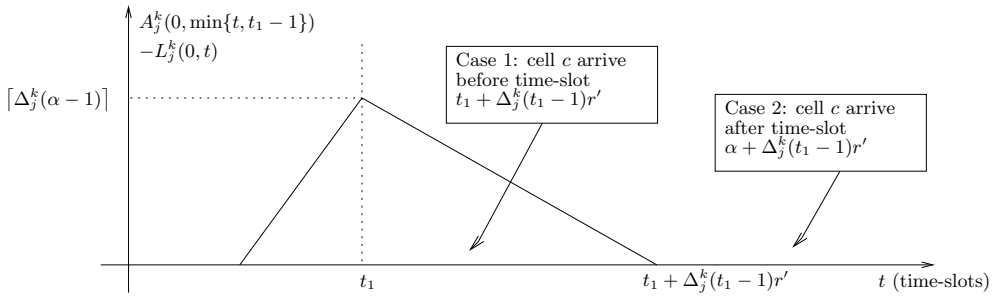

FIG. 7. *Illustration for the different cases in the proof of Theorem* 4.6.

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

$$tl_S(c) - ta(c) \ge A_j(t_1, ta(c)) - B_j + t_1 - ta(c) + 1. \tag{3}$$

Recall that by Corollary 4.5, $Q_j^k(t_1-1) \le \max\{0, \lceil \Delta_j^k(t_1-1) \rceil\}$. There are two cases to consider, depending on whether all the cells that were queued in plane k at time-slot t_1 left the plane before the arrival of cell c (see Figure 7).

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 transferred in the first time-slot of its transmission:

$$\begin{aligned}
tp(c) - ta(c) &\le r' Q_j^k(ta(c)) + 1 \\
&= r'(A_j^k(0, ta(c)) - L_j^k(0, ta(c))) + 1 \qquad \text{by the definition of } Q_j^k(ta(c)) \\
&\le r' \left(A_j^k(0, ta(c)) - A_j^k(0, t_1 - 1) + \left\lceil \frac{r'\Delta_j^k(t_1 - 1) + t_1 - ta(c)}{r'} \right\rceil \right) + 1 \\
&\qquad \text{by Lemma 4.4, since } ta(c) \in [t_1, t_1 + \Delta_j^k(t_1 - 1)r'] \\
&\qquad \text{and } L_j^k(0, ta(c)) \ge L_j^k(0, ta(c) - 1) \\
&\le r' \left(A_j^k(0, ta(c)) - A_j^k(0, t_1 - 1) + \frac{r'\Delta_j^k(t_1 - 1) + t_1 - ta(c)}{r'} + 1 \right) + 1 \\
&\le r' A_j^k(t_1, ta(c)) + r'\Delta_j^k(t_1 - 1) + t_1 - ta(c) + r' + 1 \\
&= A_j(t_1, ta(c)) + r'\Delta_j^k(t_1, ta(c)) + r'\Delta_j^k(t_1 - 1) - ta(c) + t_1 + r' + 1 \\
&\qquad \text{by Definition 4.2} \\
&\le A_j(t_1, ta(c)) + r'(\Delta_j^k(ta(c)) + 1) - ta(c) + t_1 + 1 \qquad \text{by Property 4.3.}
\end{aligned}$$

Together with (3), this implies that $tp(c) - tl_S(c) \le r'(\Delta_j^k(ta(c)) + 1) + B_j$.

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, $L_j^k(t_2, ta(c)) \geq \lfloor \frac{ta(c)-t_2+1}{r'} \rfloor$. Hence, we have

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

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

$$tp(c) - ta(c) \leq A_j(t_1, ta(c)) + r'(\Delta_j^k(ta(c)) + 1) + t_1 - ta(c) + 1.$$

Together with (3), this implies that $tp(c) - tl_S(c) \leq r'(\Delta_j^k(ta(c)) + 1) + B_j$. ∎

5. Demultiplexing algorithms with optimal relative queuing delay. This section presents several demultiplexing algorithms and uses the methodology described in section 4 in order to bound their relative queuing delay.

First, we revisit the *fractional traffic dispatch (FTD) algorithm* [20] and show that its relative queuing delay is $(N+1)r'$ time-slots. For a PPS with speedup $S > 2$, we introduce a variant of the FTD algorithm that is $2N/S$-partitioned; its relative queuing delay is at most $(2N/S + 1)\, r' + N(1 - 2/S)$ time-slots, matching the lower bound for fully distributed demultiplexing algorithms (Theorem 3.7).

Then, we present novel 1-RT and u-RT demultiplexing algorithms with relative queuing delays of $3N + r'$ time-slots (sections 5.2 and 5.3). Both algorithms have optimal relative queuing delays when the speedup of the PPS is constant.

5.1. Optimal fully distributed demultiplexing algorithm. The *fractional traffic dispatch (FTD) algorithm* [20] is the best-known example of a fully distributed demultiplexing algorithm. In the FTD algorithm, there is a *window* of size r' time-slots that slides over the sequence of cells in each flow (i,j). The algorithm maintains a window constraint that ensures that two cells in the same window are not sent through the same plane. An equivalent variation of the algorithm statically divides each flow to blocks of size r' [20, 25].

The demultiplexing algorithm chooses the plane through which a cell is sent 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 [20].

A simple application of Theorem 4.6 and the fact that $B_j \leq N$ shows the following.

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 let $A_{i \to j}^k(t_1, t_2)$ be the number of these cells

that are sent through plane k.

$$\begin{aligned}
\Delta_j^k(t_1,t_2) &= A_j^k(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'} && \text{by Definition 4.2}\\
&= \sum_{i=1}^{N} A_{i\to j}^k(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'}\\
&\le \sum_{i=1}^{N} \left\lceil \frac{A_{i\to j}(t_1,t_2)}{r'} \right\rceil - \frac{A_j(t_1,t_2)}{r'} && \text{due to the window constraint}\\
&\le \sum_{i=1}^{N} \left(\frac{A_{i\to j}(t_1,t_2)}{r'} + \frac{r'-1}{r'} \right) - \frac{A_j(t_1,t_2)}{r'} && \text{since } A_{i\to j}, r' \text{ are integers}\\
&= N\frac{r'-1}{r'}.
\end{aligned}$$

By Theorem 4.6, $\mathcal{R}_{max}(FTD) \le (N+1)r'$, since $B_j \le N$. ∎

For a PPS with speedup $S > 2$, a $\frac{2N}{S}$-partitioned variant of the FTD algorithm yields a better relative queuing delay, matching the lower bounds described in Theorems 3.7 and 3.9. In this algorithm, denoted PART-FTD, demultiplexor i uses only planes from the set $\{2r'\lfloor \frac{i}{2N/S} \rfloor, \ldots, 2r'(\lfloor \frac{i}{2N/S} \rfloor + 1) - 1\}$. This implies that each demultiplexor uses exactly $2r'$ planes, as required for the correctness of the FTD algorithm, but each plane is used only by at most $\frac{2N}{S}$ demultiplexors.

THEOREM 5.2. $\mathcal{R}_{avg}(PART\text{-}FTD) \le \mathcal{R}_{max}(PART\text{-}FTD) \le \left(\frac{2N}{S}+1\right) r' + N\left(1-\frac{2}{S}\right)$.

Proof. We use the same notation as in the proof of Theorem 5.1. The only difference is that

$$\Delta_j^k(t_1,t_2) = A_j^k(t_1,t_2) - \frac{A_j(t_1,t_2)}{r'} \le \sum_{i=1}^{N} \left\lceil \frac{A_{i\to j}(t_1,t_2)}{r'} \right\rceil - \frac{A_j(t_1,t_2)}{r'} \le \frac{2N}{S}\frac{r'-1}{r'}$$

since at most $\frac{2N}{S}$ demultiplexors can send cells through plane k. Therefore, by Theorem 4.6, $\mathcal{R}_{max}(\text{PART-FTD}) \le \left(\frac{2N}{S}+1\right) r' + N\left(1-\frac{2}{S}\right)$. ∎

5.2. Optimal 1-RT demultiplexing algorithm. We describe a 1-RT demultiplexing algorithm that matches the lower bound presented in Theorem 3.8. Informally, Algorithm 1 divides the set of planes into two equal-size sets, V_0 and 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 global information about the switch and uses it to calculate the imbalance for each plane k and each output-port j. In the next time-slot, each input-port sends a cell to output-port j only through planes with low (or zero) imbalance. Intuitively, a phase i ends when there are no balanced planes in V_i to use. In the next phase, the demultiplexors use the planes of the set V_{1-i}.

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 ensure that under no circumstances can 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

```
1: void procedure ADVANCE-CLOCK( )                ▷ invoked at the beginning of each time-slot
2:     For every j ∈ {1,...,N}: CALCULATE(j)
3:     For every j ∈ {1,...,N}: F[j] ← UPDATE(j)
4:     Update the matrix R[N][r'] according to global information
5:     t ← (t + 1) mod r'
6: end procedure
```

```
1: int procedure DISPATCH(cell c) at demultiplexor i
2:     j ← dest(c)
3:     p ← ⌊i/r'⌋
4:     set B ← ∅
5:     for x ← r'p to i do
6:         E←{k∈{1,...,K} | ∃a∈{0,...,r'−1},R[x][a]=k}
7:         k ← min(F[j] \ (B ∪ E))
8:         B ← B ∪ {k}
9:     end for
10:    R[i][t] ← 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 ← F[j]
3:     Q[j] ← Q[j] \ M[j]
4:     if Q[j] = ∅ then                                              ▷ change phase
5:         Q[j] ← {1,...,K} \ M[j]
6:         phase[j] ← (1 − phase[j])
7:         S ← V_phase[j]
8:     else
9:         S ← S \ (L[j])
10:    end if
11:    return S
12: end procedure
```

```
1: void procedure CALCULATE(int j)
2:     set A ← {k|Δ_j^k(t) > N/r'}                          ▷ using global information
3:     M[j] ← {k|Δ_j^k(t) ≤ 0}                              ▷ using global information
4:     L[j] ← (L[j] ∪ A) \ M[j]
5: end procedure
```

With respect to each output-port j, planes are divided into three levels according to their imbalance (see Definition 4.2): *balanced* planes with imbalance $\Delta_j^k(t) \leq 0$, *extremely imbalanced* planes with imbalance $\Delta_j^k(t) \geq \frac{N}{r'}$, and *slightly imbalanced* planes whose imbalance satisfies $0 < \Delta_j^k(t) < \frac{N}{r'}$. At the beginning of each time-slot, a set of *eligible* planes, denoted by $F[j]$, is calculated for every destination j: A plane is eligible for output-port j if it is balanced with respect to output-port j or if it was never extremely imbalanced with respect to output-port j since the last phase change. Phase i is changed to phase $1-i$ when all planes $k \in V_{1-i}$ become balanced, as maintained by the set $Q[j]$.

TABLE 5.1

Illustration of Example 5.1. *The plane number through which each demultiplexor would have sent a cell destined for output-port* 0*, if such a cell arrived at the switch. Actual arrivals are marked in framed boxes. No cells arrive at different output-ports in this time interval.*

	Time-slot								
	0	1	2	3	4	5	6	7	8
Demultiplexor 0		[1]	2	1	[2]	3	[9]	10	1
Demultiplexor 1		2	[1]	2	[3]	2	[10]	9	2
Demultiplexor 2		[1]	2	1	2	2	[9]	10	1
Demultiplexor 3		2	[1]	2	3	3	[10]	9	2
Demultiplexor 4		[1]	2	[1]	2	[2]	9	9	1
Demultiplexor 5		2	[1]	[2]	3	3	10	10	2
Demultiplexor 6		1	[1]	2	2	2	9	[9]	1
Demultiplexor 7		2	2	[1]	3	3	10	10	2
$\Delta_0^1(t)$		1.5	3.5	4	3	2.5	0.5	0	0
$\Delta_0^9(t)$	6.5	5	3	1.5	0.5	0	0	0.5	0.5

We demonstrate the operations of the algorithm with the following example for an 8×8 PPS with speedup $S = 8$ and $r' = \frac{R}{r} = 2$.

Example 5.1. Suppose that at time-slot $t = 0$, *phase*$[0]$ changed from 1 to 0, $\Delta_0^9(0) = 6.5$, and all other planes in V_1 have imbalance at most 6.5. In addition, we assume that planes 1 and 2 did not receive any cells before time-slot 0.

The demultiplexors are divided into 4 sets: $\{0,1\}, \{2,3\}, \{4,5\}, \{6,7\}$. Upon receiving a cell, each demultiplexor calculates the behavior of all demultiplexors in its set that have smaller index and ensures that it will not send the cell through the same plane as them. Table 5.1 shows the plane number through which each demultiplexor would have sent a cell destined for output-port 0 if such a cell arrives at the switch.

The actual arrivals are marked in framed boxes and are taken into account in the following time-slots.

At time-slot 1, demultiplexor 0 will send a cell through the first plane in V_0 (that is, plane 1). On the other hand, demultiplexor 1 must avoid sending its cell through plane 1, and therefore it will use plane 2. Similarly, demultiplexors $2, 4$, and 6 will use plane 1, and demultiplexors $3, 5$, and 7 will use plane 2.

At time-slot 2, demultiplexor 0 cannot use plane 1 due to the input-constraint. Therefore, it will use plane 2 and demultiplexor 1 will use plane 1. Plane 1 becomes extremely imbalanced after time-slot 2, and therefore it is not eligible to receive cells for output-port 0 in the following time-slots. Although plane 1 becomes slightly imbalanced after time-slot 3, Algorithm 1 dictates that it is still not eligible for output-port 0, since the phase has not changed yet.

The phase changes after time-slot 5 because, for every plane $k \in V_1$, $\Delta_0^k(5) \leq 0$. This implies that planes from the set V_1 are used for sending cells destined for output-port 0 in the following time-slots. At this time, $Q[0] = \{1,2\}$ since $\Delta_0^1(5) = 2.5$ and $\Delta_0^2(5) = 0.5$. The phase changes again after time-slot 7, since both $\Delta_0^1(7)$ and $\Delta_0^2(7)$ are not positive.

To prove the correctness of Algorithm 1, we start with two lemmas.

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

LEMMA 5.3. *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), and cells are not sent through plane k until a time-slot $t_3' > t_3 + 1$ in which $\Delta_j^k(t_3' - 1) \leq 0$. This observation also holds if the phase changes in the beginning of time-slot $t_3 + 1$, since the condition $Q[j] = \emptyset$ in line 4 of procedure UPDATE implies that $V_{phase} \subseteq M[j]$ in line 7.

For every two input-ports i_1 and i_2, if $\lfloor \frac{i_1}{r'} \rfloor = \lfloor \frac{i_2}{r'} \rfloor$, 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 4.2, 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 is 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, toward 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 4.2

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

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

The second property is a simple conclusion from Lemma 5.3.

LEMMA 5.4. *If $2N$ cells destined for output j arrive at a PPS operating under Algorithm 1 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 4.2, 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 5.3 and Definition 4.2, $\Delta_j^k(t_3, t_1 - 1) \leq \Delta_j^k \leq \frac{2N}{r'}$. 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. ∎

The final theorem shows that a speedup of 8 suffices for this demultiplexing algorithm to achieve optimal relative queuing delay. Note that such a high speedup is considered impractical for real switches; nevertheless, Algorithm 1 demonstrates that the lower bound presented in Theorem 3.8 is tight for $u = 1$.

THEOREM 5.5. *Algorithm 1 works correctly with speedup $S = 8$ and maximum relative queuing delay of $3N + r'$ time-slots.*

Proof. 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 $|F[j]| > 2r'$ after any execution of UPDATE(j).

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 $|V_0 \cap L[j]| \geq 2r'$, the number of cells arriving at the switch, destined for output-port j, during time interval $[t_2, t_1]$, is at least $(2r')\frac{N}{r'-1} > 2N$. During time interval $[t_2, t_1]$, no cells are sent to any plane in V_1, and Lemma 5.4 implies that for every plane $k \in V_1$, $\Delta_j^k(t_1) \leq 0$; in particular, this holds for all planes in $Q[j]$. This implies that $Q[j]$ becomes empty, and the phase changes at least once during time interval $[t_2, t_1]$, contradicting the choice of t_1 and t_2.

By Lemma 5.3 and Theorem 4.6, the relative queuing delay of the algorithm is at most $3N + r'$. ∎

5.3. Optimal u-RT demultiplexing algorithm. Algorithm 1 can be used as a building block for u-RT algorithms with $u > 1$. Algorithm 2 runs u instances of Algorithm 1 in cycles, such that in each time-slot only one instance is active (that is, the ith instance is active on time-slots $i, i + u, i + 2u, i + 3u$ etc.). Since there are u time-slots between two consecutive times in which the same instance is active, global information on the previous time the instance was active can be shared among the demultiplexors. In addition, each instance of Algorithm 1 has its own set of $8r'$ planes; hence Algorithm 2 needs a speedup $S = 8u$.

Algorithm 2 u-RT Algorithm

Shared:
- $ALG[u]$: u instances of Algorithm 1 ▷ Each instance with its own planes and shared variables
- x: value in $\{0, \ldots, u-1\}$, initially $u - 1$ ▷ cyclic pointer to array ALG

1: void **procedure** ADVANCE-CLOCK( ) ▷ invoked at the beginning of each time-slot
2: $x \leftarrow (x + 1) \bmod u$
3: $ALG[x]$.ADVANCE-CLOCK() ▷ invoke procedure ADVANCE-CLOCK on the xth instance
4: **end procedure**

1: int **procedure** DISPATCH(cell c) at demultiplexor i
2: **return** $ALG[x]$.DISPATCH(c) ▷ invoke procedure DISPATCH on the xth instance
3: **end procedure**

We next bound the imbalance under Algorithm 2.

LEMMA 5.6. *In Algorithm* 2*, for every plane* k *and output-port* j*,* $\Delta_j^k < \frac{2N}{r'}$.

Proof. Assume, toward a contradiction, that there is a traffic T, a plane k, and an output-port j such that $\Delta_j^k \geq \frac{2N}{r'}$. Let t be the first time-slot in which $\Delta_j^k(t) \geq \frac{2N}{r'}$, and let $x = t \bmod u$. The choice of t and Algorithm 2 imply that a cell is sent through plane k at time-slot t by instance x.

Let T' be the traffic consisting of the cells of traffic T handled by the instance x; that is, $T' = \{c \mid c \in T, ta(c) - x \bmod u = 0\}$. Let $round(c) = \frac{ta(c)-x}{u}$ be the number of times instance x was active until cell c arrived to the switch.

Consider traffic $\widetilde{T}$ in which each cell c of traffic T' arrives at the switch at time-slot $round(c)$; that is, $\widetilde{T} = \{\text{shift}(c, round(c) - ta(c)) \mid c \in T'\}$. Let $\widetilde{A}_j(t_1, t_2)$ be the number of cells in traffic $\widetilde{T}$ destined for output-port j that arrive at the switch during time interval $[t_1, t_2]$, and let $\widetilde{A}_j^k(t_1, t_2)$ be the number of these cells that are sent through plane k by Algorithm 1. Similarly, following Definition 4.2, $\widetilde{\Delta}_j^k(t_1, t_2) = \widetilde{A}_j^k(t_1, t_2) - \frac{1}{r'}\widetilde{A}_j(t_1, t_2)$, and $\widetilde{\Delta}_j^k(t_2) = \max_{t_1 \leq t_2} \widetilde{\Delta}_j^k(t_2)$.

Since only instance x sends cells to plane k, and the dispatching decisions of instance x in response to traffic T are the same as the decisions of Algorithm 1 in response to traffic $\widetilde{T}$, it follows that for every time $t' < t$, $A_j^k(t', t) = \widetilde{A}_j^k(\lceil \frac{t'-x}{u} \rceil, \frac{t-x}{u})$. On the other hand, in $\widetilde{T}$ there is a subset of T's cells destined for output-port j, and therefore $A_j(t', t) \geq \widetilde{A}_j(\lceil \frac{t'-x}{u} \rceil, \frac{t-x}{u})$. This implies that $\widetilde{\Delta}_j^k(\frac{t-x}{u}) \geq \Delta_j^k(t) \geq \frac{2N}{r'}$, contradicting Lemma 5.3. ∎

Lemma 5.6, Theorem 5.5, and Theorem 4.6 immediately imply the following theorem.

THEOREM 5.7. *For any* $u \geq 1$ *and a PPS with speedup* $S = 8u$*, there is a* u*-RT demultiplexing algorithm* ALG *such that* $\mathcal{R}_{max}(\texttt{ALG}) \leq 3N + r'$.

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 does not reduce the relative queuing delay. Our lower bounds use the fact that switches are FCFS but can be generalized to rely on other priority-based policies.

We also present a methodology for bounding the relative queuing delay and use it to show that the lower bounds are tight. In particular, we prove that the relative queuing delay of the FTD algorithm [20] is $(N+1)r'$ time-slots, exactly matching the lower bound for fully distributed algorithms. We also design u-RT demultiplexing algorithms with relative queuing delay of $3N + r' < 4N$ time-slots; for the common case of constant speedup—independent of the size of the switch and its rates—this asymptotically matches the lower bound of $\frac{uN}{S}\left(1 - \frac{u}{r'}\right)$ time-slots for u-RT algorithms, when $1 \leq u \leq \frac{r'}{2}$.

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 a relative queuing delay approximately equal to the delay when no information is gathered at all. This implies that without dedicated control links working at a rate much larger than r, it is probably unprofitable to share information between the input-ports.

Acknowledgments. We would like to thank Ilan Newman and Fabian Kuhn for helpful discussions about randomized demultiplexing algorithms. We would also like to thank the anonymous referees for their comments and suggestions.

REFERENCES

[1] 3Com Corporation, *Inverse Multiplexing over ATM (IMA): A Breakthrough WAN Technology for Corporate Networks*, Marlborough, MA, 1997.

[2] The ATM Forum, *Inverse Multiplexing for ATM (IMA) Specification, Version* 1.1, 1999, AF-PHY-0086.001; available online from http://www.ipmplsforum.org/tech/atm_specs.shtml.

[3] H. Attiya and D. Hay, *The inherent queuing delay of parallel packet switches*, IEEE Trans. Parallel Distrib. Systems, 17 (2006), pp. 1048–1056.

[4] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal, *Balanced allocations*, SIAM J. Comput., 29 (1999), pp. 180–200.

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

[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 (2002), pp. 611–622.

[7] A. Charny, *Providing QoS Guarantees in Input Buffered Crossbar Switches with Speedup*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[8] A. Charny, P. Krishna, N.S. Patel, and R.J. Simcoe, *Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup*, in Proceedings of the 6th IEEE/IFIP International Workshop on Quality of Service, IEEE Communications Society, New York, NY, 1998, pp. 235–244.

[9] F.M. Chiussi, D.A. Khotimsky, and S. Krishnan, *Generalized inverse multiplexing of switched ATM connections*, in Proceedings of IEEE Globecom, IEEE Communications Society, New York, NY, 1998, pp. 3134–3140.

[10] S. Chuang, A. Goel, N. McKeown, and B. Prabhakar, *Matching output queueing with a combined input output queued switch*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 1999, pp. 1169–1178.

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

[12] C. Clos, *A study of non-blocking switching networks*, Bell System Tech. J., 32 (1953), pp. 406–424.

[13] R.L. Cruz, *A calculus for network delay, part* I*: Network elements in isolation*, IEEE Trans. Inform. Theory, 37 (1991), pp. 114–131.

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

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

[16] P. Giaccone, B. Prabhakar, and D. Shah, *Randomized scheduling algorithms for high-aggregate bandwidth switches*, IEEE J. Selected Areas in Communications, 21 (2003), pp. 546–559.

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

[18] S. Iyer, *Personal communication*, 2006.

[19] S. Iyer, A.A. Awadallah, and N. McKeown, *Analysis of a packet switch with memories running slower than the line rate*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2000, pp. 529–537.

[20] S. Iyer and N. McKeown, *Making parallel packet switches practical*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2001, pp. 1680–1687.

[21] S. Iyer and N. McKeown, *Analysis of the parallel packet switch architecture*, IEEE/ACM Trans. Networking, 11 (2003), pp. 314–324.

[22] I. Keslassy, *Load Balanced Router*, Ph.D. thesis, Stanford University, Palo Alto, CA, 2004.

[23] I. Keslassy and N. McKeown, *Maintaining packet order in two-stage switches*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2002.

[24] D. Khotimsky, *Personal communication*, 2004.

[25] D. Khotimsky and S. Krishnan, *Stability analysis of a parallel packet switch with bufferless input demultiplexors*, in Proceedings of the IEEE International Conference on Communications (ICC), IEEE Communications Society, New York, NY, 2001, pp. 100–106.

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

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

[28] Lucent Technologies, *Inverse Multiplexing for ATM, Expanding the Revenue Opportunities for Converged Services over ATM*, Murray Hill, NJ, 2001.

[29] M.D. Mitzenmacher, *On the analysis of randomized load balancing schemes*, Theory Comput. Syst., 32 (1999), pp. 361–386.
[30] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.
[31] PMC-Sierra, Inc., *Inverse Multiplexing Over ATM Works Today*, available online at http://www.electronicstalk.com/news/pmc/pmc121.html, 2002.
[32] B. Prabhakar and N. McKeown, *On the speedup required for combined input and output queued switching*, Automatica J. IFAC, 35 (1999), pp. 1909–1920.
[33] A. Prakash, A. Aziz, and V. Ramachandran, *A near optimal schedule for switch-memory-switch routers*, in Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), ACM, New York, 2003, pp. 343–352.
[34] A. Prakash, A. Aziz, and V. Ramachandran, *Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2004.
[35] S. Sharif, A. Aziz, and A. Prakash, *An $O(\log^2 N)$ parallel algorithm for output queuing*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 2002.
[36] D.C. Sthephens and H. Zhang, *Implementing distributed packet fair queueing in a scalable architecture*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 1998, pp. 282–290.
[37] I. Stoica and H. Zhang, *Exact emulation of an output queueing switch by a combined input output queueing switch*, in Proceedings of the 6th IEEE/IFIP International Workshop on Quality of Service, IEEE Communications Society, New York, NY, 1998, pp. 218–224.
[38] L. Tassiulas, *Linear complexity algorithms for maximum throughput in radio networks and input queued switches*, in Proceedings of IEEE INFOCOM, IEEE Communications Society, New York, NY, 1998, pp. 533–539.