SIAM J. COMPUT.
Vol. 0, No. 0, pp. 000–000

EFFICIENCY OF OBLIVIOUS VERSUS NONOBLIVIOUS SCHEDULERS FOR OPTIMISTIC, RATE-BASED FLOW CONTROL*

PANAGIOTA FATOUROU†, MARIOS MAVRONICOLAS‡, AND PAUL SPIRAKIS§

Abstract. Two important performance parameters of distributed, *rate-based flow control algorithms* are their *locality* and *convergence complexity*. The former is characterized by the amount of global knowledge that is available to their scheduling mechanisms, while the latter is defined as the number of *update operations* performed on rates of individual *sessions* until *max-min fairness* is reached. *Optimistic* algorithms allow any *session* to intermediately receive a rate larger than its max-min fair rate; *bottleneck* algorithms finalize the rate of a session only if it is restricted by a certain, highly congested link of the network. In this work, we present a comprehensive collection of lower and upper bounds on convergence complexity, under varying degrees of locality, for optimistic, bottleneck, rate-based flow control algorithms.

Say that an algorithm is *oblivious* if its scheduling mechanism uses no information of either the session rates or the network topology. We present a novel, combinatorial construction of a capacitated network, which we use to establish a fundamental lower bound of $\frac{dn}{4} + \frac{n}{2}$ on the convergence complexity of *any* oblivious algorithm, where n is the number of sessions laid out on a network, and d, the *session dependency*, is a measure of topological dependencies among sessions. Moreover, we devise a novel *simulation proof* to establish that, perhaps surprisingly, the lower bound of $\frac{dn}{4} + \frac{n}{2}$ on convergence complexity still holds for any *partially oblivious* algorithm, in which the scheduling mechanism is allowed to use information about session rates, but is otherwise unaware of network topology.

On the positive side, we prove that the lower bounds for oblivious and partially oblivious algorithms are both *tight*. We do so by presenting *optimal* oblivious algorithms, which converge after $\frac{dn}{2} + \frac{n}{2}$ update operations are performed in the *worst* case. To complete the picture, we show that *linear* convergence complexity can indeed be achieved if information about both session rates and network topology is available to schedulers. We present a counterexample, a *nonoblivious* algorithm, which converges within an *optimal* number of n update operations.

Our results imply a surprising convergence complexity collapse of oblivious and partially oblivious algorithms, and a convergence complexity separation between (partially) oblivious and nonoblivious algorithms for optimistic, bottleneck rate-based flow control.

Key words. distributed algorithms; lower bounds; rate-based flow control; max-min fairness; convergence complexity; oblivious, partially oblivious, and nonoblivious algorithms; bottleneck algorithms

---

*Received by the editors August 4, 2003; accepted for publication (in revised form) December 7, 2004; published electronically DATE. A preliminary version of this work appeared in the Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing (Santa Barbara, California, 1997), ACM, New York, 1997, pp. 139–148. This work was partially supported by the ESPRIT Program of the European Union under the Long Term Research Project ALCOM-IT (contract 20244), by the IST Program of the European Union under Projects ALCOM-FT (contract IST-1999-14186), FLAGS (contract IST-2001-33116), and DELIS (contract 001907), by funds from the Program E.Π.E.A.E.K. II of the Greek Ministry of Education, by research funds at University of Cyprus, and by AT&T/NCR research funds.

http://www.siam.org/journals/sicomp/x-x/43275.html

†Department of Computer Science, University of Ioannina, P.O. Box 1186, Ioannina GR-45110, Greece (faturu@cs.uoi.gr). Part of this research was performed while this author was at the Department of Computer Engineering and Informatics, University of Patras, the Research and Academic Computer Technology Institute, the Max-Planck Institute für Informatik, and while visiting the Department of Computer Science, University of Cyprus.

‡Department of Computer Science, University of Cyprus, P.O. Box 20537, Nicosia CY-1678, Cyprus (mavronic@ucy.ac.cy). Part of this research was performed while this author was at the Department of Computer Science and Engineering, University of Connecticut, while visiting the University of Patras and Research and Academic Computer Technology Institute, and while at AT&T Labs—Research, as a visitor to the DIMACS Special Year on Networks.

§Department of Computer Engineering and Informatics, University of Patras, Rion, Patras GR-26500, Greece, and Research and Academic Computer Technology Institute, P.O. Box 1122, Patras GR-26110, Greece (spirakis@cti.gr).

AMS subject classifications. 68M10, 68M12, 68M14, 68Q25, 68R05, 68W15, 68W40, 90B18

DOI. 10.1137/S009753970343275X

1. Introduction. In many modern *communication networks*, a connection between different users is established by a *session*, a virtual circuit of infinite duration that involves a fixed path between a *source* and a *destination*. The role of *flow control algorithms* is to alleviate throughput degradation, unfairness, deadlocks, and failures by preventing buffer overflow from arising in situations where the externally injected load is larger than what can be accommodated even with optimal routing (see, e.g., [2, Chapter 6] or [3, 6, 12, 14, 15, 16, 17, 22]). In particular, *rate-based* flow control algorithms adjust transmission rates of different sessions in an end-to-end manner, with the objective to optimize network utilization while still maintaining fairness between different sessions (see, e.g., [1, 2, 4, 5, 11, 13, 16, 18]).

For a range of settings including both high-speed networks and Internet applications, *max-min fairness* [1, 2, 5, 6, 7, 13, 14, 15, 16, 17, 19] has emerged as a widely accepted formulation of fairness for flow control; roughly speaking, max-min fairness requires that it be impossible for any session to receive an infinitesimally larger rate on the account of a session with a smaller or equal rate [15, 16, 17, 21]. Call *max-min fair rates* those achieving max-min fairness.

Any rate-based flow control algorithm can be classified as one of two broad classes, *conservative* and *optimistic*, according to the way in which rates of sessions are adjusted. Conservative algorithms (see, e.g., [2, Chapter 6]) do not provide for decreases to the rates of sessions; in contrast, optimistic algorithms allow decreases, so that rates may go up and down and a session can intermediately receive a rate larger than its final. Afek, Mansour, and Ostfeld [1] introduced optimistic algorithms and advocated them to fit better than the conservative ones into real dynamic networks; indeed, in such networks, new sessions may join in, so that it is desirable to incrementally adjust rates by decreasing the rates of some of them.

A crucial component of a rate-based flow control algorithm is its *scheduler*, the abstract mechanism it uses to decide which session's rate to adjust next. Apparently, it is desirable that the scheduler does not require *global* knowledge of either the session rates or the network topology. Clearly, no-knowledge schedulers are not only more efficient in terms of communication and computation, but they also adjust more easily to dynamic changes in network topology. So, one important performance parameter of a rate-based flow control algorithm is its *locality*, measured by the amount of global knowledge required by the scheduler.

Call a scheduler that uses no information of either the session rates or the network topology an *oblivious* scheduler. On the opposite extreme, a *nonoblivious* scheduler requires full knowledge of both session rates and network topology. There is, in addition, an important middle ground between oblivious and nonoblivious schedulers: schedulers which, although unaware of network topology, do have access to session rates; call these schedulers *partially oblivious*. Clearly, a partially oblivious scheduler is superior to a nonoblivious scheduler in terms of robustness to dynamic changes in network topology, while it is surpassed by an oblivious scheduler. Afek et al. [1, sections 4 and 5] presented two interesting, partially oblivious schedulers called GlobalMinSched and LocalMinSched, respectively; these schedulers always choose for an increase the session whose rate is the *globally* and *locally* smallest, respectively.

A second crucial component of a rate-based flow control algorithm is its *terminator*, the abstract mechanism it uses to decide which sessions to terminate at each

specific instant.[1] *Bottleneck* terminators [16] finalize the rate of a session, thereby terminating the session, only if its rate is restricted on some particular network link that allows for the least possible share of bandwidth to each session traversing it; call such a link a *bottleneck edge* [16].

When a session is scheduled for an increase, its rate is increased by the minimum possible share of bandwidth along its path; this may involve possible decreases to the rates of crossing sessions (see, e.g., [5, 6, 7, 18, 19]). The *convergence complexity* of a rate-based flow control algorithm is the number of individual rate adjustments performed in the worst case until the algorithm quiesces and rates have reached max-min fairness (so that they do not change any further). Thus, convergence complexity models the number of rounds of communication and local computation needed for convergence to max-min fairness, so that it is another significant performance parameter of such distributed algorithms.

We measure convergence complexity in terms of a particular, simple abstraction of rate adjustments, called *update operation*, which was introduced in [1, section 2.1]. In essence, an `update` operation atomically adjusts the rates of a group of neighboring sessions in a fair and optimistic way.[2] We note that some of the essential intricacies encountered when designing practical, distributed flow-control algorithms [4, 5, 6, 7, 18, 19, 22] include scheduling the rate adjustments, minimizing the communication, and converging to fairness quickly. Although we do not address in this work implementation issues for the model used, we feel that it captures most of these intricacies. (For a discussion of such issues for this model, see [9].)

This work presents a comprehensive collection of bounds on convergence complexity under varying degrees of locality for optimistic, bottleneck, rate-based flow control algorithms; more specifically, we show upper and lower bounds (sections 5 and 7, respectively) on convergence complexity for oblivious, partially oblivious, and nonoblivious such algorithms. The lower bounds demonstrate that achieving oblivious, or even partially oblivious, scheduling, and therefore locality, necessarily imposes a nonconstant, multiplicative overhead on convergence complexity. These are the first general lower bounds ever shown on the convergence of rate-based flow control algorithms. In addition, our algorithms span a wide spectrum of convergence complexity bounds and locality properties, while they improve significantly upon all previous optimistic algorithms with respect to the combination of these two performance measures. To establish these upper bounds, we offer several new basic properties and tools for the design and analysis of optimistic, bottleneck, rate-based flow control algorithms (section 4); these properties significantly extend and strengthen the corresponding ones shown in [1].

Our bounds are expressed in terms of n, the total number of sessions laid out on the network, and a new parameter d, called *session dependency*, through which we derive more accurate bounds on convergence complexity. Specifically, d is the maximum number of sessions that share an edge either directly or indirectly. In the

[1] In all of our discussion, a terminated session is meant to be one that has reached its final (max-min fair) rate, so that its rate value will not change any further.

[2] In more detail, an `update` operation increases, if possible, the rate of one session, so that it becomes the session with the maximum rate on some particular link that allows the minimum possible increase. The new value is a function of the link capacity and the rates of other sessions traversing the link. Conflicting sessions whose rates exceed the new value have their rates decreased to the new value, while rates of other sessions remain unchanged. These adjustments saturate that particular link.

rest of this paper, we will focus on optimistic and bottleneck algorithms; we sometimes refer to them simply as algorithms.

Our first major result is a fundamental lower bound on the convergence complexity of oblivious algorithms. Its proof relies on constructing, given any arbitrary oblivious algorithm, a specific network, as a function of the algorithm's scheduler, so that if sessions are scheduled on this network according to the scheduler and the algorithm computes the max-min fair rates, then $\Omega(dn)$ `update` operations are required before convergence (section 6). The construction is novel, purely combinatorial, and of independent interest. We rely on the algorithm being optimistic and bottleneck for showing that convergence is slow.

Although intuition may suggest that knowledge of session rates can be crucial to performance, we surprisingly establish that the lower bound of $\Omega(dn)$ that we show on the convergence complexity of oblivious algorithms applies also to partially oblivious algorithms. We use a powerful simulation proof to simulate any partially oblivious scheduler on some network by an oblivious scheduler on the same network. The simulation inherits the $\Omega(dn)$ lower bound shown for oblivious algorithms down to partially oblivious algorithms.

We show a matching upper bound on the convergence complexity of oblivious and partially oblivious algorithms. We present a class of oblivious algorithms, called d-Epoch, with convergence complexity $\Theta(dn)$; an example is algorithm RoundRobin, which uses the simple and natural idea of scheduling sessions in a round-robin order. We note that the partially oblivious algorithms GlobalMin and LocalMin from [1, sections 4 and 5] also achieve convergence complexity $\Theta(dn)$, which implies the tightness of our lower bound for partially oblivious algorithms; however, any d-Epoch algorithm improves over these two algorithms since it is oblivious.

At this point, it is only natural to ask whether it is possible to overcome the $\Theta(dn)$ barrier on convergence complexity achievable by oblivious or partially oblivious algorithms, possibly at the cost of sacrificing locality. Perhaps not very surprisingly, it turns out that the locality enjoyed by oblivious and partially oblivious algorithms comes at a multiplicative in d overhead on convergence complexity. We present a counterexample—nonoblivious and optimistic algorithm called Linear—that achieves an exact bound of n on convergence complexity; Linear follows the earlier idea of selecting and conservatively increasing the rate of any session that traverses the most highly congested link in the network (see, e.g., [2, section 6.4.2] or [15]). We emphasize that Linear, although theoretically efficient, is clearly impractical.

Our work differs at first from earlier work on rate-based flow control carried out in the data networks community (see, e.g., [2, 3, 11, 12, 15, 16, 17, 22]) in adopting the optimistic approach introduced in [1]. We have been able to show that certain classical scheduling policies, such as round-robin [13, 14] or scheduling a session that traverses the most congested link [15], are correctly applicable in the optimistic framework. Most interestingly, we have shown the first general lower bounds on the convergence complexity of rate-based flow control algorithms; these imply that some scheduling policies are provably superior over some other policies, in terms of either convergence complexity or locality (or both), and even optimal.

2. Model. Many of our definitions formalize and refine those from [1] and [2, Chapter 6]. A *communication network* is a graph $G = (V, E)$, where vertices and edges represent switches and links, respectively. A set $\mathcal{S}$ of $n \geq 1$ *sessions* is laid out on G. Each session S_i (also denoted as i) passes through a sequence of edges and has a *rate* $r(S_i)$. The vector $\mathbf{r} = \langle r(S_1), \ldots, r(S_n) \rangle$ is the *rate vector*. Each

edge has a *capacity* $c(e) > 0$, which the sum of the rates of the sessions traversing it cannot exceed; when this sum equals the capacity, the edge is *saturated*. In a *max-min fair rate vector* [15, 16, 17, 21], an increase to the rate of any session requires either exceeding an edge capacity or decreasing the rate of another session with equal or lower rate.

The communication network is abstracted as a state machine. Each state $Q = \langle \mathbf{r}_Q, \mathcal{A}_Q \rangle$ consists of a *rate vector* $\mathbf{r}_Q$ and a set $\mathcal{A}_Q \subseteq \mathcal{S}$ of *active sessions*. Intuitively, an active session is one that has not yet reached its max-min fair rate. Denote by $\mathcal{D}_Q = \mathcal{S} \setminus \mathcal{A}_Q$ the set of *done sessions* in state Q. In the *initial state* Q^{in}, all sessions are active and have zero rates. A state is *final* if all sessions have reached their max-min fair rates.

An operation defines a procedure to compute new rates for a set of sessions on the basis of their old rates. Formally, an *operation* is a function `operation` that takes as input a session S_i and a state Q, and outputs new rates for S_i and all sessions S_j sharing edges with S_i. Say that `operation` is *conservative* if it decreases no individual rate; else `operation` is *optimistic*.

We will consider a specific optimistic operation, called `update`, which we describe below. For a state Q and each edge e traversed by S_i, $\Delta_Q(i, e)$ is the maximum amount by which $r_Q(S_i)$ can be increased (without exceeding the capacity of e) while decreasing down to the increased rate of S_i the rates of other active sessions passing through e and exceeding the increased rate. Notice that these rate adjustments saturate e. Intuitively, $\Delta_Q(i, e)$ is the maximum amount by which $r_Q(S_i)$ can be increased in a fair manner if edge e were the only edge constraining S_i. Finally, S_i's rate is increased by $\Delta_Q(i)$, called the *increase* for S_i in Q, which is the minimum of these maximum amounts over all edges that S_i passes through. In addition, each (active) session S_j sharing an edge with S_i has its rate decreased down to the new rate $r(S_i)$ (unless it is already less). We remark that the rates computed by `update` saturate the edge(s) realizing $\Delta_Q(i)$, but no other edges. Notice also that the `update` operation uses only local information with respect to session S_i [17, section IV]. From now on, we will consider only optimistic algorithms that employ the `update` operation.

A scheduler decides the order of sessions on which to perform the `update` operation. Formally, a *scheduler* (cf. [1]) is a function Sched that maps a pair $\langle G, \mathcal{S} \rangle$, a state Q, and a state index $l \geq 1$ to a session $i = \mathsf{Sched}\left(\langle G, \mathcal{S} \rangle, Q, l\right)$. A *terminator* is a function Term that maps a pair $\langle G, \mathcal{S} \rangle$ and a state Q to a set of sessions $\mathcal{T} = \mathsf{Term}\left(\langle G, \mathcal{S} \rangle, Q\right) \subseteq \mathcal{A}_Q$; intuitively, Term decides which active sessions will be *terminated* (and their rates will not change any further). An *algorithm* is a pair $\mathsf{Alg} = \langle \mathsf{Sched}, \mathsf{Term} \rangle$. The classification of operations into conservative and optimistic leads to the corresponding classification of algorithms in a natural way.

An *oblivious* scheduler uses no knowledge of either the topology of the network or the rates and *statuses* (active or done) of sessions. Thus, an oblivious scheduler is a (finite or infinite) sequence $\mathsf{Sched} = i_1, i_2, \ldots$, where for each $l \geq 1$, $i_l \in [n]$. A *partially oblivious* scheduler uses no knowledge of the topology of the network, while it may use knowledge of the rates (and statuses) of sessions. Notice that any oblivious scheduler is also partially oblivious, but not vice versa. A *nonoblivious* scheduler is a scheduler that is not partially oblivious. The classification of schedulers into oblivious, partially oblivious, and nonoblivious leads to the corresponding classification of algorithms in a natural way.

For any edge $e \in E$ and state Q, the *allotted capacity* of e in Q [1, section 2.1], denoted $allot_Q(e)$, is the total rate already allocated to done sessions passing through

the edge. Clearly, $\sum_{S \in \mathcal{A}_Q | e} r_Q(S) \leq c(e) - allot_Q(e)$, where $\mathcal{A}_Q \mid e$ denotes the set of active sessions traversing e in Q. For any state Q and for any edge e with $|\mathcal{A}_Q \mid e| > 0$ active sessions, the *fair share* of e in state Q [1, section 2.1], denoted $FS_Q(e)$, is defined to be $FS_Q(e) = \frac{c(e) - allot_Q(e)}{|\mathcal{A}_Q \mid e|}$; intuitively, $FS_Q(e)$ is the per session share of the portion of the capacity of edge e which has not yet been allocated to sessions done in state Q that traverse the edge. For any state Q, a *bottleneck edge* for Q [16] is an edge e such that for each active session passing through e, $FS_Q(e)$ is the minimum fair share over all edges traversed by the session.

A terminator Term is *bottleneck* [16] if for any state Q and session S, $S \in \mathsf{Term}(\langle G, \mathcal{S} \rangle, Q)$ if (and only if) there exists an edge e traversed by S such that (1) e is a bottleneck edge for Q, and (2) $r_Q(S) = FS_Q(e)$. Whenever such an edge e exists for state Q and Term is bottleneck, we will say that e causes the termination of session S in Q. Say that an algorithm Alg is *bottleneck* if Term is.

The *execution* α of Alg on $\langle G, \mathcal{S} \rangle$ is an infinite sequence of alternating states and session indices $\alpha = Q_0, i_1, Q_1, \ldots, i_l, Q_l, \ldots$, satisfying the following conditions:

(1) $Q_0 = Q^{in}$ and $\mathcal{A}_{Q_0} = \mathcal{A}_{Q_1} = \mathcal{S}$;
(2) for each $l \geq 1$, $i_l = \mathsf{Sched}(\langle G, \mathcal{S} \rangle, Q_{l-1}, l-1)$;
(3) for each $l \geq 1$, if $i_l \in \mathcal{D}_{Q_l}$ or $\Delta_{Q_{l-1}}(i_l) = 0$, then $\mathbf{r}_{Q_l} = \mathbf{r}_{Q_{l-1}}$ and $\mathcal{A}_{Q_{l+1}} = \mathcal{A}_{Q_l}$; else
   (a) (i) $r_{Q_l}(i_l) := r_{Q_{l-1}}(i_l) + \Delta_{Q_{l-1}}(i_l)$;
   (ii) for each session $i \in \mathcal{A}_{Q_l}$, $i \neq i_l$, such that $S_i \cap S_{i_l} \neq \emptyset$,

$$r_{Q_l}(i) := \min\{r_{Q_{l-1}}(i), r_{Q_{l-1}}(i_l) + \Delta_{Q_{l-1}}(i_l)\};$$

   (b) $\mathcal{A}_{Q_{l+1}} = \mathcal{A}_{Q_l} \setminus \mathsf{Term}(\langle G, \mathcal{S} \rangle, Q_l)$.

Call each state Q_l in an execution α, where $l \geq 1$, a *reachable state*. Say that Alg *computes the max-min fair rate vector* if for each execution of Alg a final state Q is reachable such that $\mathbf{r}_Q$ is a max-min fair rate vector.

The number of `update` operations in any execution α of Alg is the number of state indices $l \geq 1$ such that $i_l \in \mathcal{A}_{Q_{l-1}}$. The *convergence complexity* of Alg on network G with session set $\mathcal{S}$, denoted $\mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S} \rangle)$, is the number of operations in the execution of Alg on G with $\mathcal{S}$. The *convergence complexity of* Alg, denoted $\mathcal{U}_{\mathsf{Alg}}$, is the maximum, over all pairs $\langle G, \mathcal{S} \rangle$, of the convergence complexity of Alg on G with $\mathcal{S}$.

3. Notation. We collect here some notation that will be used in most of the following sections. For each edge e and for any set of sessions $\mathcal{S}' \subseteq \mathcal{S}$, denote $\mathcal{S}' \mid e$ to be the set of sessions in $\mathcal{S}'$ traversing e. We will sometimes treat a session S_i as the set of its links; so, for any edge e traversed by S_i, we will write $e \in S_i$. For an edge e and for a rate vector $\mathbf{r}$, denote by $\mathbf{r} \mid e$ the restriction of $\mathbf{r}$ to sessions traversing e. We will sometimes abuse notation by writing $\mathbf{r}_{\mathcal{A}_Q}$ and $\mathbf{r}_{\mathcal{D}_Q}$ to denote the restriction of $\mathbf{r}_Q$ to active and done sessions, respectively, in Q.

Define the *share-an-edge* relation on $\mathcal{S}$, denoted $\|_{\mathcal{S}}$, as follows. For any pair of sessions $S_i, S_j \in \mathcal{S}$, $S_i \|_{\mathcal{S}} S_j$ if S_i and S_j traverse a common edge. The transitive closure of $\|_{\mathcal{S}}$ is an equivalence relation on $\mathcal{S}$, which partitions $\mathcal{S}$ into equivalence classes $\mathcal{S}_1, \ldots, \mathcal{S}_c$, called *clusters*, where $1 \leq c \leq n$. The *session dependency* d is the maximum size of a cluster. Call a cluster $\mathcal{S}_j$ an *active cluster* in state Q if it contains at least one session that remains active in Q.

The set of *active edges* of the cluster $\mathcal{S}_j$ in state Q, denoted $AE_Q(\mathcal{S}_j)$, contains all edges of the network traversed by at least one active session in Q that belongs to $\mathcal{S}_j$.

The set of *active edges* of the network in state Q, denoted AE_Q, contains all edges of the network traversed by at least one active session in Q. The *residual capacity* of edge e in state Q, denoted $resid_Q(e)$, is the difference between the capacity of e and the total rate of sessions traversing e in Q.

The set of *edges with minimum fair share* for cluster $\mathcal{S}_j$ in state Q is defined as $MFSE_Q(\mathcal{S}_j) = \{e \in AE_Q(\mathcal{S}_j) \mid FS_Q(e) \leq FS_Q(e') \text{ for each } e' \in AE_Q(\mathcal{S}_j)\}$. The set of *edges with minimum fair share* in state Q, denoted $MFSE_Q$, is defined to be $MFSE_Q = \bigcup_{1 \leq j \leq c} MFSE_Q(\mathcal{S}_j)$.

An *execution fragment* α of Alg is a contiguous subsequence of some execution of Alg starting with the state $first(\alpha)$; if α is finite, it ends with the state $last(\alpha)$. For each execution (resp., execution fragment) α of Alg, the *schedule* $\sigma(\alpha)$ is the sequence of session indices in α. If α is a finite execution fragment and α' is any execution fragment such that $first(\alpha') = last(\alpha)$, then $\alpha \cdot \alpha'$ is the concatenation of α and α', eliminating the duplicate occurrence of $last(\alpha) = first(\alpha')$.

For any index $l \geq 1$, the *preceding state* of Q_l in execution α, denoted $\overleftarrow{Q_l}$, is the state Q_{l-1}; for any index $l \geq 0$, the *successor state* of Q_l in α, denoted $\overrightarrow{Q_l}$, is the state Q_{l+1}. For any indices l and $l' > l$, write $Q_l \xrightarrow{\alpha} Q_{l'}$ to denote that Q_l precedes $Q_{l'}$; moreover, write $Q_l \xmapsto{\alpha} Q_{l'}$ if additionally Q_l and $Q_{l'}$ may coincide. For any index $l \geq 1$, we will sometimes abuse language and say that session l is *scheduled in front* of state Q_l. For any state Q, denote i_Q the session scheduled in front of Q. The *least schedule* for Q_l in α, denoted $\widehat{l}$, is the index of the earliest state following Q_l by which all sessions that remain active in Q_l have been scheduled at least once, or infinite if no such state exists. Define $\widehat{l} \mid e$ in the natural way.

4. Bottleneck algorithms. In this section, we present basic properties of bottleneck algorithms, which will be useful in what follows. These properties refer to an execution $\alpha = Q_0, i_1, Q_1, \ldots, i_l, Q_l, \ldots$ of any bottleneck algorithm. To prove these properties, some more general properties are also proved in section 4.1.

4.1. Preliminaries. We study how several quantities of interest change during an execution. We first state an immediate consequence of the definitions of allotted capacity and execution; we then prove a similar simple fact that will be helpful in later proofs.

LEMMA 4.1. *For each integer $l \geq 1$, and for any edge e,*

$$allot_{Q_l}(e) = allot_{Q_{l-1}}(e) + \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e} r_{Q_{l-1}}(i).$$

LEMMA 4.2. *For any integers l_0 and l, $0 \leq l_0 < l$, and for any edge e,*

$$\sum_{i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_l}) \mid e} r_{Q_{l_0}}(i) = allot_{Q_l}(e) - allot_{Q_{l_0}}(e).$$

Proof. Clearly, $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_l}) \mid e$ if and only if $i \in (\mathcal{D}_{Q_l} \setminus \mathcal{D}_{Q_{l_0}}) \mid e$. Hence, it follows that

$$\sum_{i\in(\mathcal{A}_{Q_{l_0}}\setminus\mathcal{A}_{Q_l})|e} r_{Q_l}(i)$$

$$= \sum_{i\in(\mathcal{D}_{Q_l}\setminus\mathcal{D}_{Q_{l_0}})|e} r_{Q_l}(i)$$

$$= \sum_{i\in\mathcal{D}_{Q_l}|e} r_{Q_l}(i) - \sum_{i\in\mathcal{D}_{Q_{l_0}}|e} r_{Q_l}(i) (\text{since } \mathcal{D}_{Q_{l_0}} \subseteq \mathcal{D}_{Q_l})$$

$$= allot_{Q_l}(e) - allot_{Q_{l_0}}(e),$$

as needed. □

We continue to prove that the saturation of an edge depends in a critical way on how rates of sessions traversing the edge compare to each other.

LEMMA 4.3. *For any integer $l_0 \geq 0$, assume that edge e is active in state Q_{l_0}. Then, for each integer $l \geq l_0$, the following hold:*

1. *if for each session $i \in \mathcal{A}_{Q_{l_0}} \mid e$, $r_{Q_l}(i) = FS_{Q_{l_0}}(e)$, then e is saturated in Q_l;*
2. *if for each session $i \in \mathcal{A}_{Q_{l_0}} \mid e$, $r_{Q_l}(i) < FS_{Q_{l_0}}(e)$, then e is not saturated in Q_l;*
3. *there exists no session $k \in \mathcal{A}_{Q_{l_0}} \mid e$ such that $r_{Q_l}(k) > FS_{Q_{l_0}}(e)$, while for each session $i \in \mathcal{A}_{Q_{l_0}} \mid e$, $i \neq k$, $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$.*

Proof. We start by proving (1). Clearly,

$$\sum_{i\in\mathcal{A}_{Q_l}|e} r_{Q_l}(i)$$

$$= \sum_{i\in\mathcal{A}_{Q_{l_0}}|e} r_{Q_l}(i) - \sum_{i\in(\mathcal{A}_{Q_{l_0}}\setminus\mathcal{A}_{Q_l})|e} r_{Q_l}(i) \qquad (\text{since } \mathcal{A}_{Q_l} \mid e \subseteq \mathcal{A}_{Q_{l_0}} \mid e)$$

$$= \sum_{i\in\mathcal{A}_{Q_{l_0}}|e} FS_{Q_{l_0}}(e) - \big(allot_{Q_l}(e) - allot_{Q_{l_0}}(e)\big) \qquad (\text{by assumption and Lemma 4.2})$$

$$= \big|\mathcal{A}_{Q_{l_0}} \mid e\big| \cdot FS_{Q_{l_0}}(e) - (allot_{Q_l}(e) - allot_{Q_{l_0}}(e))$$

$$= c(e) - allot_{Q_{l_0}}(e) - (allot_{Q_l}(e) - allot_{Q_{l_0}}(e)) \qquad (\text{by definition of fair share})$$

$$= c(e) - allot_{Q_l}(e),$$

as needed to establish that e is saturated in state Q_l.

Condition (2) is proved in an almost identical way. (The only difference is that now the assumption for (2) and Lemma 4.2 imply that $\sum_{i\in\mathcal{A}_{Q_{l_0}}|e} r_{Q_l}(S_i) - \sum_{i\in(\mathcal{A}_{Q_{l_0}}\setminus\mathcal{A}_{Q_l})|e} r_{Q_l}(S_i) < \sum_{i\in\mathcal{A}_{Q_{l_0}}|e} FS_{Q_{l_0}}(e) - (allot_{Q_l}(e) - allot_{Q_{l_0}}(e).)$

We finally prove (3). Assume otherwise; so, there exists some session $k \in \mathcal{A}_{Q_{l_0}} \mid e$ such that $r_{Q_l}(k) > FS_{Q_{l_0}}(e)$, while for each $i \in \mathcal{A}_{Q_{l_0}} \mid e$, $i \neq k$, $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$.

Then

$$
\begin{aligned}
&\sum_{i\in\mathcal{A}_{Q_l}|e} r_{Q_l}(i)\\
&= \sum_{i\in\mathcal{A}_{Q_{l_0}}|e} r_{Q_l}(i) - \sum_{i\in(\mathcal{A}_{Q_{l_0}}\setminus\mathcal{A}_{Q_l})|e} r_{Q_l}(i) && \text{(since } \mathcal{A}_{Q_l}|e \subseteq \mathcal{A}_{Q_{l_0}} \mid e)\\
&= \sum_{i\in\mathcal{A}_{Q_{l_0}}|e} r_{Q_l}(i) - (allot_{Q_l}(e) - allot_{Q_{l_0}}(e)) && \text{(by Lemma 4.2)}\\
&= r_{Q_l}(k) + \sum_{i\in\mathcal{A}_{Q_{l_0}}|e, i\neq k} r_{Q_l}(i) - allot_{Q_l}(e) + allot_{Q_{l_0}}(e)\\
&> FS_{Q_{l_0}}(e) + \sum_{i\in\mathcal{A}_{Q_{l_0}}|e, i\neq k} FS_{Q_{l_0}}(e) - allot_{Q_l}(e) + allot_{Q_{l_0}}(e) && \text{(by assumption)}\\
&= \sum_{i\in\mathcal{A}_{Q_{l_0}}|e} FS_{Q_{l_0}}(e) - allot_{Q_l}(e) + allot_{Q_{l_0}}(e))\\
&= \left|\mathcal{A}_{Q_{l_0}} \mid e\right| \cdot FS_{Q_{l_0}}(e) - allot_{Q_l}(e) + allot_{Q_{l_0}}(e)\\
&= c(e) - allot_{Q_{l_0}}(e) - allot_{Q_l}(e) + allot_{Q_{l_0}}(e) && \text{(by definition of fair share)}\\
&= c(e) - allot_{Q_l}(e),
\end{aligned}
$$

so that $\sum_{i\in\mathcal{A}_{Q_l}|e} r_{Q_l}(i) > c(e) - allot_{Q_l}(e)$. This is a contradiction. □

The next claim follows directly from the definitions of bottleneck edge and fair share.

LEMMA 4.4. *Let e and e' be bottleneck edges for state Q such that $\mathcal{A}_Q \mid e \bigcap \mathcal{A}_Q \mid e' \neq \emptyset$. Then $FS_Q(e) = FS_Q(e')$.*

The following (easy to prove) claim is a direct consequence of the definitions of a bottleneck edge and a minimum fair share edge for some particular cluster.

LEMMA 4.5. *For any state Q and cluster $\mathcal{S}_j$, consider an edge $e \in MFSE_Q(\mathcal{S}_j)$. Then e is a bottleneck edge for Q.*

We are interested in algorithms for which a final state is reachable for each possible execution. Bottleneck algorithms (whether conservative or optimistic) enjoy a related, interesting property [15].

PROPOSITION 4.6 (Hayden [15]). *Assume that* Alg *is a bottleneck algorithm. Then, for any reachable final state Q of* Alg, $\mathbf{r}_Q$ *is a max-min fair rate vector.*

Proposition 4.6 implies that in order to show that any given bottleneck algorithm computes the max-min fair rate vector, it suffices to prove that it reaches a final state. We continue with an interesting monotonicity property of fair share.

LEMMA 4.7 (Afek, Mansour, and Ostfeld [1]). *Assume that* Alg *is a bottleneck algorithm. Then, for each integer $l \geq 1$ and for any edge $e \in AE_{Q_l}$, $FS_{Q_l}(e) \geq FS_{Q_{l-1}}(e)$.*

4.2. Properties of bottleneck edges. We strengthen Lemma 4.7 for the special case of bottleneck edges. We present a collection of invariant properties for any edge that becomes bottleneck in the course of an execution of a bottleneck algorithm (whether conservative or optimistic).

PROPOSITION 4.8 (invariants of bottleneck edge). *Assume that* Alg *is bottleneck. For any integer $l_0 \geq 0$, fix any edge e that is a bottleneck edge for Q_{l_0}. Then, for any integer $l \geq l_0$ such that $e \in AE_{Q_l}$, the following hold:*

(1) $FS_{Q_l}(e) = FS_{Q_{l_0}}(e)$;
(2) *e is a bottleneck edge for* Q_l;
(3) *for any session* $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_{l+1}}) \mid e$, $r_{Q_l}(i) = FS_{Q_{l_0}}(e)$.

Roughly speaking, Proposition 4.8 establishes that no change in the fair share of an active edge occurs once the edge has become bottleneck, so that the edge remains bottleneck; moreover, the final rate of any active session traversing it is equal to this constant fair share.

Proof. The proof is by induction on l. For the basis case where $l = l_0$, condition (1) holds trivially, condition (2) holds by assumption, and condition (3) holds by definition of a bottleneck algorithm. Assume inductively that for some integer $l \geq l_0$, the claims hold for any integer l' where $l_0 \leq l' < l$. For the induction step, we show that the claims hold for integer l. We start by proving condition (1). Clearly,

$$\begin{aligned}
&FS_{Q_l}(e)\\
&= \frac{c(e) - allot_{Q_l}(e)}{|\mathcal{A}_{Q_l} \mid e|}\\
&= \frac{c(e) - allot_{Q_{l-1}}(e) - \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e} r_{Q_{l-1}}(i)}{|\mathcal{A}_{Q_l} \mid e|} && \text{(by Lemma 4.1)}\\
&= \frac{\left|\mathcal{A}_{Q_{l-1}} \mid e\right| FS_{Q_{l-1}}(e) - \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e} r_{Q_{l-1}}(i)}{|\mathcal{A}_{Q_l} \mid e|} && \text{(by definition of fair share).}
\end{aligned}$$

Consider any session $i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e$. Since $\mathcal{A}_{Q_{l-1}} \subseteq \mathcal{A}_{Q_{l_0}}$, this implies that $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_l}) \mid e$, so that by the induction hypothesis (condition (3)), $r_{Q_{l-1}}(i) = FS_{Q_{l_0}}(e)$. So,

$$\begin{aligned}
&FS_{Q_l}(e)\\
&= \frac{\left|\mathcal{A}_{Q_{l-1}} \mid e\right| FS_{Q_{l-1}}(e) - \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e} FS_{Q_{l_0}}(e)}{|\mathcal{A}_{Q_l} \mid e|}\\
&= \frac{\left|\mathcal{A}_{Q_{l-1}} \mid e\right| FS_{Q_{l_0}}(e) - \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e} FS_{Q_{l_0}}(e)}{|\mathcal{A}_{Q_l} \mid e|} && \text{(by the induction hypothesis)}\\
&= \frac{\left|\mathcal{A}_{Q_{l-1}} \mid e\right| FS_{Q_{l_0}}(e) - \left|(\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e\right| FS_{Q_{l_0}}(e)}{|\mathcal{A}_{Q_l} \mid e|}\\
&= \frac{(\left|\mathcal{A}_{Q_{l-1}} \mid e\right| - \left|(\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e\right|) FS_{Q_{l_0}}(e)}{|\mathcal{A}_{Q_l} \mid e|}\\
&= \frac{|\mathcal{A}_{Q_l} \mid e| \; FS_{Q_{l_0}}(e)}{|\mathcal{A}_{Q_l} \mid e|}\\
&= FS_{Q_{l_0}}(e),
\end{aligned}$$

which completes the proof of condition (1).

We continue to prove condition (2). Take any session $i \in \mathcal{A}_{Q_l} \mid e$. Clearly, $i \in \mathcal{A}_{Q_{l_0}} \mid e$. Since e is a bottleneck edge for Q_{l_0}, $FS_{Q_{l_0}}(e) = MFS_{Q_{l_0}}(i) = \min_{e' \in S_i} FS_{Q_{l_0}}(e')$. Consider any edge $e' \in S_i$. Since $i \in \mathcal{A}_{Q_l} \mid e$, it follows that $e' \in AE_{Q_l}$; thus, by Lemma 4.7, $FS_{Q_l}(e') \geq FS_{Q_{l_0}}(e')$. Since e' was chosen arbitrarily, this implies that $\min_{e' \in S_i} FS_{Q_l}(e') \geq \min_{e' \in S_i} FS_{Q_{l_0}}(e')$. It follows that $\min_{e' \in S_i} FS_{Q_l}(e') \geq FS_{Q_{l_0}}(e)$. By condition (1) shown above, this implies that

$\min_{e' \in S_i} FS_{Q_l}(e') \geq FS_{Q_l}(e)$. Since $e \in S_i$, $\min_{e' \in S_i} FS_{Q_l}(e') \leq FS_{Q_l}(e)$. It follows that $FS_{Q_l}(e) = \min_{e' \in S_i} FS_{Q_l}(e')$. Since i was chosen arbitrarily, this implies that e is a bottleneck edge for Q_l, which completes the proof of condition (2).

We finally prove condition (3). Take any session $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_{l+1}}) \mid e$. Since $i \notin \mathcal{A}_{Q_{l+1}}$, there exists some integer l', $l_0 < l' \leq l$, such that $i \in \mathsf{Term}(Q_{l'})$. Since Term is bottleneck, there exists some edge $e' \in S_i$ such that e' is a bottleneck edge for $Q_{l'}$, and $r_{Q_{l'}}(i) = FS_{Q_{l'}}(e')$.

Either $l_0 < l' < l$ or $l' = l$. If $l_0 < l' < l$, then the induction hypothesis (condition (2)) implies that e is a bottleneck edge for $Q_{l'}$; if, on the other hand, $l' = l$, then condition (2) shown above implies that e is a bottleneck edge for $Q_{l'}$. Thus, in either case, e is a bottleneck edge for $Q_{l'}$. Since both e and e' are bottleneck edges for state Q'_l and $S_i \in (\mathcal{A}_{Q_{l'}} \mid e) \bigcap (\mathcal{A}_{Q_{l'}} \mid e')$, Lemma 4.4 implies that $FS_{Q_{l'}}(e) = FS_{Q_{l'}}(e')$. Since $r_{Q_{l'}}(i) = FS_{Q_{l'}}(e')$, this implies that $r_{Q_{l'}}(i) = FS_{Q_{l'}}(e)$. Since $l \geq l'$ and $i \in \mathsf{Term}(Q_{l'})$, $r_{Q_l}(i) = r_{Q_{l'}}(i)$. Either $l_0 < l' < l$ or $l' = l$. If $l_0 < l' < l$, then the induction hypothesis (condition (1)) implies that $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$; if, on the other hand, $l' = l$, then condition (1) shown above implies that $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$. Thus, in either case, $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$. Hence, it follows that $r_{Q_l}(i) = FS_{Q_{l_0}}(e)$, which completes the proof of condition (3). □

4.3. Properties of minimum fair share edges. We start by proving a simple invariant property for any edge that becomes a minimum fair share edge for any particular cluster in the course of an execution of a bottleneck algorithm. We establish that the edge remains a minimum fair share edge (as long as it is active).

PROPOSITION 4.9 (invariant of minimum fair share edge). *Assume that* Alg *is bottleneck. For any integer* $l_0 \geq 0$*, fix any edge* $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$ *for some active cluster* $\mathcal{S}_j$ *in* Q_{l_0}*. Then, for any integer* $l \geq l_0$ *such that* $e \in AE_{Q_l}$*,* $e \in MFSE_{Q_l}(\mathcal{S}_j)$*.*

Proof. Consider any edge $e' \in AE_{Q_l}(\mathcal{S}_j)$; clearly, $e' \in AE_{Q_{l_0}}(\mathcal{S}_j)$. Since $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, it follows that $FS_{Q_{l_0}}(e) \leq FS_{Q_{l_0}}(e')$. By Lemma 4.5, e is a bottleneck edge for Q_{l_0}; thus, by Proposition 4.8 (condition (2)), $FS_{Q_l}(e) = FS_{Q_{l_0}}(e)$. By Lemma 4.7, $FS_{Q_{l_0}}(e') \leq FS_{Q_l}(e')$. So, $FS_{Q_l}(e) \leq FS_{Q_l}(e')$. Since e' is arbitrary, it follows that $e \in MFSE_{Q_l}(\mathcal{S}_j)$. □

Similarly to Proposition 4.8, Proposition 4.9 holds for any bottleneck algorithm (whether conservative or optimistic) as well. However, the rest of the properties established in this section require the assumption of optimistic, bottleneck algorithms. We first prove a safety property for any edge that becomes a minimum fair share edge for any particular cluster during the execution of an optimistic, bottleneck algorithm.

PROPOSITION 4.10 (safety property of minimum fair share edge). *Assume that* Alg *is optimistic and bottleneck. For any integer* $l_0 \geq 0$*, fix any edge* $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$ *for some active cluster* $\mathcal{S}_j$ *in* Q_{l_0}*. Consider any session* $i \in \mathcal{A}_{Q_{l_0}} \mid e$ *such that* $r_{Q_{l_0}}(i) \geq FS_{Q_{l_0}}(e)$*. Then, for any integer* $l \geq l_0$*,* $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$*.*

Proposition 4.10 considers any (active) session traversing a minimum fair share edge; roughly speaking, it establishes that no decrease to its rate below this particular minimum fair share is possible if the rate is initially no less than the minimum fair share.

Proof. The proof is by induction on l. For the basis case where $l = l_0$, the claim holds by our assumption. Assume inductively that for some integer $l > l_0$, $r_{Q_{l-1}}(i) \geq FS_{Q_{l_0}}(e)$. For the induction step, we show that $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$.

Assume first that $r_{Q_l}(i) \geq r_{Q_{l-1}}(i)$. By the induction hypothesis, this implies that $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$, as needed. So assume that $r_{Q_l}(i) < r_{Q_{l-1}}(i)$. By definition of execution and `update` operation, this implies that S_i intersects the session S_{i_l},

scheduled in front of state Q_l; moreover, $r_{Q_l}(i) = r_{Q_l}(i_l)$. Let e' be an edge such that $\Delta_{Q_{l-1}}(i_l) = \Delta_{Q_{l-1}}(i_l, e')$. We prove the following.

LEMMA 4.11. $r_{Q_l}(i_l) \geq FS_{Q_l}(e')$.

Proof. Assume otherwise; so, $r_{Q_l}(i_l) < FS_{Q_l}(e')$. By definition of the `update` operation, e' is saturated in Q_l; moreover, for any session $k \in \mathcal{A}_{Q_{l_0}} \mid e'$, $r_{Q_l}(i_l) \geq r_{Q_l}(k)$. Thus, $FS_{Q_l}(e') > r_{Q_l}(k)$. Lemma 4.3 (condition (2)) implies that e' is not saturated in Q_l—a contradiction. □

Since $r_{Q_l}(i) = r_{Q_l}(i_l)$, Lemma 4.11 implies that $r_{Q_l}(i) \geq FS_{Q_l}(e')$. Also, by Lemma 4.7, $FS_{Q_l}(e') \geq FS_{Q_{l_0}}(e')$, so that $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e')$. Since $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, $FS_{Q_{l_0}}(e') \geq FS_{Q_{l_0}}(e)$. It follows that $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$, as needed. □

The next claim complements Proposition 4.10 by giving a corresponding liveness property.

PROPOSITION 4.12 (liveness property of minimum fair share edge). *Assume that* Alg *is optimistic and bottleneck. For any integer $l_0 \geq 0$, fix any edge $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$ for some active cluster $\mathcal{S}_j$ in Q_{l_0}, such that $\widehat{l_0} \mid e < \infty$. Consider any session $i \in \mathcal{A}_{Q_{l_0}} \mid e$. Then, for any integer $l \geq \widehat{l_0} \mid e$, $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$.*

Proposition 4.12 considers any active session traversing a minimum fair share edge; roughly speaking, it establishes that eventually, once all active sessions traversing this minimum fair share edge have been scheduled at least once, the rate of the session will be no less than this particular minimum fair share.

Proof. We start with an informal outline of the proof. We consider the point of the execution following state Q_{l_0} where session i is scheduled; clearly, that point comes no later than when all sessions have been scheduled at least once. We establish that at this point, the rate of S_i is no less than the fair share of edge e in state Q_{l_0}. We also argue that e remains a minimum fair share edge beyond state Q_{l_0}; this allows us to exploit the safety property of minimum fair share edges in order to argue that the rate of S_i will subsequently remain no less than the fair share of e in state Q_{l_0}. We now present the details of the formal proof.

Since $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, Lemma 4.5 implies that e is a bottleneck edge for state Q_{l_0}. Since $i \in \mathcal{A}_{Q_{l_0}} \mid e$, it follows by definition of $\widehat{l_0} \mid e$ that there exists a least index l', $l_0 < l' \leq \widehat{l_0} \mid e$, such that i is scheduled in front of state $Q_{l'}$. We proceed by case analysis.

1. Assume first that i is not active in state $Q_{l'}$. Since $i \in \mathcal{A}_{Q_{l_0}}$, there exists an index l'', $l_0 \leq l'' < l'$, such that $i \in \mathsf{Term}(Q_{l''})$. Since Term is bottleneck, it follows that there exists some edge e' traversed by session i that is a bottleneck edge for state $Q_{l''}$, and $r_{Q_{l''}}(i) = FS_{Q_{l''}}(e')$. Since $i \in \mathcal{A}_{Q_{l''}}$ and i traverses e, e is an active edge at $Q_{l''}$. By Proposition 4.8 (conditions (1) and (2)), $FS_{Q_{l''}}(e) = FS_{Q_{l_0}}(e)$, and e is a bottleneck edge for $Q_{l''}$. By Lemma 4.4, $FS_{Q_{l''}}(e') = FS_{Q_{l''}}(e)$, so that $FS_{Q_{l''}}(e') = FS_{Q_{l_0}}(e)$. It follows that $r_{Q_{l''}}(i) = FS_{Q_{l_0}}(e)$. Now take any integer $l \geq \widehat{l_0} \mid e$. Clearly, $l \geq l''$. Since $i \in \mathsf{Term}(Q_{l''})$, $r_{Q_l}(i) = r_{Q_{l''}}(i) = FS_{Q_{l_0}}(e)$, which establishes the claim in this case.
2. Assume now that i is active in state $Q_{l'}$. Since i traverses edge e, it follows that e is active in state $Q_{l'}$. By Proposition 4.8 (conditions (1) and (2)), $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$, and e is a bottleneck edge for $Q_{l'}$. We prove the following.

   LEMMA 4.13. $r_{Q_{l'}}(i) \geq FS_{Q_{l_0}}(e)$.

   Proof. Assume, by way of contradiction, that $r_{Q_{l'}}(i) < FS_{Q_{l_0}}(e)$. Let e' be an edge such that $\Delta_{Q_{l'}}(i) = \Delta_{Q_{l'}}(i, e')$. By definition of `update` operation,

e' is saturated in state $Q_{l'}$. Since e is a bottleneck edge for state $Q_{l'}$, and i traverses both e and e', $FS_{Q_{l'}}(e) \leq FS_{Q_{l'}}(e')$. Since $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$, this implies that $FS_{Q_{l_0}}(e) \leq FS_{Q_{l'}}(e')$. Since $r_{Q_{l'}}(i) < FS_{Q_{l_0}}(e)$, it follows that $r_{Q_{l'}}(i) < FS_{Q_{l'}}(e')$. By definition of the `update` operation, for any session $k \in \mathcal{A}_{Q_{l'}} \mid e$, $r_{Q_{l'}}(k) \leq r_{Q_{l'}}(i)$, so that $r_{Q_{l'}}(k) < FS_{Q_{l'}}(e')$. It follows by Lemma 4.3 (condition (2)) that e' is not saturated in state $Q_{l'}$. This is a contradiction. □

Now take any integer $l \geq \widehat{l_0} \mid e$. Clearly, $l \geq l'$. Since e is a minimum fair share edge for Q_{l_0}, Proposition 4.9 implies that e is a minimum fair share edge for $Q_{l'}$ as well. Moreover, by Lemma 4.13, $r_{Q_{l'}}(i) \geq FS_{Q_{l_0}}(e)$. Since $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$, this implies that $r_{Q_{l'}}(i) \geq FS_{Q_{l'}}(e)$. It follows by Proposition 4.10 (taking l' for l_0) that $r_{Q_l}(i) \geq FS_{Q_{l_0}}(e)$, which establishes the claim in this case.

The proof of Proposition 4.12 is now complete. □

4.4. Termination properties. The first property considers active sessions in any particular cluster that traverse a minimum fair share edge; it is established that once each such session has been scheduled at least once, all of these sessions must have become done.

PROPOSITION 4.14 (termination of all sessions). *Assume that* Alg *is optimistic and bottleneck. For any integer $l_0 \geq 0$, fix any edge $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$ for some active cluster $\mathcal{S}_j$ in Q_{l_0} such that $\widehat{l_0} \mid e < \infty$. Then, for any session $i \in \mathcal{A}_{Q_{l_0}} \mid e$, $i \in \mathcal{D}_{\underset{Q_{\widehat{l_0}|e}}{\longrightarrow}}$.*

Proof. We start with an informal outline of the proof. We consider any session active in state Q_{l_0}, and we argue that after all sessions have been scheduled at least once, the session will receive rate equal to the fair share of e in Q_{l_0}. We will exploit the fact that e is a bottleneck edge for Q_{l_0} in order to argue that e remains bottleneck subsequently, and that its fair share does not change. Since Alg is a bottleneck algorithm, this will be sufficient for deducing that the session has reached its final rate. We now present the details of the formal proof.

Fix any session $i \in \mathcal{A}_{Q_{l_0}} \mid e$. We start by proving the following.

LEMMA 4.15. $r_{Q_{\widehat{l_0}|e}}(i) = FS_{Q_{l_0}}(e)$.

Proof. Assume, by way of contradiction, that $r_{Q_{\widehat{l_0}|e}}(i) \neq FS_{Q_{l_0}}(e)$. By Proposition 4.12, $r_{Q_{\widehat{l_0}|e}}(i) \geq FS_{Q_{l_0}}(e)$. It follows that $r_{Q_{\widehat{l_0}|e}}(i) > FS_{Q_{l_0}}(e)$. We proceed by case analysis.

Assume first that there exists no session $k \in \mathcal{A}_{Q_{l_0}} \mid e$ with $k \neq i$; thus, $\mathcal{A}_{Q_{l_0}} \mid e = \{i\}$. Denote by Q_l the latest state in execution α, such that $Q_{l_0} \overset{\alpha}{\longmapsto} Q_l \overset{\alpha}{\longmapsto} Q_{\widehat{l_0}|e}$, and $\mathcal{A}_{Q_l} \neq \emptyset$. Thus, $\mathcal{A}_{Q_l} \mid e = \mathcal{A}_{Q_{l_0}} \mid e$ and $allot_{Q_l}(e) = allot_{Q_{l_0}}(e)$. Clearly,

$$
\begin{aligned}
&\sum_{k \in \mathcal{A}_{Q_l} | e} r_{Q_l}(k) \\
&\quad = r_{Q_l}(i) && \text{(since } \mathcal{A}_{Q_{l_0}} \mid e = \{i\}\text{)} \\
&\quad = r_{Q_{\widehat{l_0}|e}}(i) && \text{(by definition of state } Q_l\text{)} \\
&\quad > FS_{Q_{l_0}}(e) && \text{(by assumption)} \\
&\quad = c(e) - allot_{Q_{l_0}}(e) && \text{(by definition of fair share and since } \left|\mathcal{A}_{Q_{l_0}} \mid e\right| = 1\text{)} \\
&\quad = c(e) - allot_{Q_l}(e).
\end{aligned}
$$

This is a contradiction.

Assume now that there exists some session $k \in \mathcal{A}_{Q_{l_0}} \mid e$ with $k \neq i$. By Proposition 4.12, $r_{Q_{\widehat{l_0}|e}}(k) \geq FS_{Q_{l_0}}(e)$. Together with $r_{Q_{\widehat{l_0}|e}}(i) > FS_{Q_{l_0}}(e)$, this implies a contradiction to Lemma 4.3 (condition (3)), and the proof is now complete. □

We continue with the proof of Proposition 4.14. In case $i \in \mathcal{D}_{Q_l}$ for some state Q_l in α such that $Q_{l_0} \xrightarrow{\alpha} Q_l \overset{\alpha}{\longmapsto} Q_{\widehat{l_0}|e}$, the definition of execution implies that $i \in \mathcal{D}_{\underset{Q_{\widehat{l_0}|e}}{\longrightarrow}}$. So, assume that for each state Q_l in execution α such that $Q_{l_0} \xrightarrow{\alpha} Q_l \overset{\alpha}{\longmapsto} Q_{\widehat{l_0}|e}$, $i \in \mathcal{A}_{Q_l}$. Denote by Q_l the latest state in execution α such that both $Q_{l_0} \xrightarrow{\alpha} Q_l \overset{\alpha}{\longmapsto} Q_{\widehat{l_0}|e}$ and $r_{Q_l}(i) = r_{Q_{\widehat{l_0}|e}}(i)$.

Since $e \in MFSE_{Q_{l_0}}$, Lemma 4.5 implies that e is a bottleneck edge for Q_{l_0}. Since $i \in \mathcal{A}_{Q_l}$ and i traverses edge e, it follows that $\mathcal{A}_{Q_l} \mid e \neq \emptyset$. Hence, Proposition 4.8 (conditions (1) and (2)) implies that $FS_{Q_l}(e) = FS_{Q_{l_0}}(e)$, and e is a bottleneck edge for Q_l. By Lemma 4.15, $r_{Q_{\widehat{l_0}|e}}(i) = FS_{Q_{l_0}}(e)$. It follows that $r_{Q_l}(i) = FS_{Q_l}(e)$.

In total, e is a bottleneck edge for state Q_l, traversed by session i for which $r_{Q_l}(i) = FS_{Q_l}(e)$. Since Alg is bottleneck, it follows that $i \in \mathcal{D}_{\underset{Q_l}{\longrightarrow}}$. Since $Q_l \rightarrow \overrightarrow{Q_{\widehat{l_0}|e}}$, it follows that $i \in \mathcal{D}_{\underset{Q_{\widehat{l_0}|e}}{\longrightarrow}}$. □

The final termination property is a direct consequence of Proposition 4.14. In essence, we establish that scheduling any sequence of sessions that includes all currently active ones must result in finalizing the rate of *at least one* active session per cluster.

PROPOSITION 4.16 (termination of at least one session per cluster). *Assume that* Alg *is optimistic and bottleneck. For any integer* $l_0 \geq 0$ *such that* $\mathcal{A}_{Q_{l_0}} \neq \emptyset$ *and* $\widehat{l_0} < \infty$*, fix any active cluster* $\mathcal{S}_j$ *in* Q_{l_0}*. Then there exists some session* $S_i \in \mathcal{S}_j \cap \mathcal{A}_{Q_{l_0}}$ *such that* $S_i \in \mathcal{D}_{\underset{Q_{\widehat{l_0}}}{\longrightarrow}}$.

Proof. Since $\mathcal{S}_j$ is active in $Q_{\mathcal{A}_{Q_{l_0}}}$, it follows that $MFSE_{Q_{l_0}}(\mathcal{S}_j) \neq \emptyset$. Fix any edge $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, and consider any session $i \in \mathcal{A}_{Q_{l_0}} \mid e$. By Proposition 4.14, $i \in \mathcal{D}_{\underset{Q_{\widehat{l_0}}}{\longrightarrow}}$. □

5. Upper bounds.

5.1. Oblivious algorithms. This section presents the algorithm RoundRobin and shows the following.

THEOREM 5.1 (upper bound for oblivious algorithms). RoundRobin *computes the max-min fair rate vector within* $\frac{dn}{2} + \frac{n}{2}$ `update` *operations.*

The scheduler of RoundRobin conducts scheduling rounds. In each round, each of the n sessions is scheduled in round-robin order. Moreover, RoundRobin is bottleneck. By definition of RoundRobin, each session is scheduled once in each round. Thus, Proposition 4.16 implies that at least one session per cluster becomes done in each round. Since each cluster contains at most d sessions, all sessions are done after d rounds, whence the network enters a final state. So, Proposition 4.6 immediately implies that RoundRobin computes the max-min fair rate vector.

We now establish an upper bound on the convergence complexity of RoundRobin. Since at least one session per cluster becomes done in each round, at most $|\mathcal{S}_j| - l + 1$ `update` operations are executed in round l, $1 \leq l \leq |\mathcal{S}_j|$, on sessions in any cluster $\mathcal{S}_j$.

Summing up over all clusters and rounds yields that

$$\begin{aligned}\mathcal{U}_{\mathsf{RoundRobin}} &\le \sum_{j\ge 1}\sum_{1\le l\le d} \min\{0, (|\mathcal{S}_j| - l + 1)\} \\ &\le \sum_{j\ge 1}\sum_{1\le l\le |\mathcal{S}_j|} (|\mathcal{S}_j| - l + 1) \\ &= \sum_{j\ge 1} \frac{|\mathcal{S}_j|(\mathcal{S}_j + 1)}{2} \\ &\le \frac{dn}{2} + \frac{n}{2}.\end{aligned}$$

RoundRobin extends naturally to a class of bottleneck algorithms d-Epoch. The scheduler d-EpochSched $= d$-EpochSched$_1 \ldots d$-EpochSched$_d$ d-EpochSched$'$, of algorithm d-Epoch is a sequence such that for each index r, $1 \le r \le d$, all sessions are included in d-EpochSched$_r$. An argument identical to the one applied to RoundRobin immediately implies the following.

THEOREM 5.2 (upper bound for oblivious algorithms). *d-Epoch computes the max-min fair rate vector within $\frac{dn}{2} + \frac{n}{2}$ `update` operations.*

5.2. Nonoblivious algorithms. This section presents the algorithm Linear and shows the following.

THEOREM 5.3 (upper bound for nonoblivious algorithms). *Linear computes the max-min fair rate vector within exactly n `update` operations.*

The scheduler of Linear maintains an active edge of minimum fair share and schedules all active sessions traversing it in any order. Once it finishes, it chooses any other (active) edge of minimum fair share, and so on. Moreover, Linear is bottleneck.

Consider any state Q_{l_0} and an arbitrary edge $e \in MFSE_{Q_{l_0}}$. Clearly, $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$ for some cluster $\mathcal{S}_j$. By definition of Linear, each session traversing e is scheduled exactly once, so that the state $Q_{\widehat{l_0|e}}$ is reached; by Proposition 4.14, each such session is done in state $Q_{\widehat{l_0|e}}$. It follows that all sessions eventually become done and a final state is reached. Hence, Proposition 4.6 immediately implies that Linear computes the max-min fair rate vector.

Linear incurs n `update` operations. Recall that all rates are initially zero. Since all capacities exceed zero, all rates in a max-min fair rate vector exceed zero as well. Since each `update` increases the rate of exactly one session, it follows that at least n `update` operations are needed, so that $\mathcal{U}_{\mathsf{Alg}} \ge n$ for every Alg. Thus, Linear is optimal.

6. Network construction. We present a generic, combinatorial construction of a network associated with any sequence $\mathsf{Seq} = i_1, i_2, \ldots$, where for each $l \ge 1$, $i_l \in [n]$. For any sequence Seq, denote $|\mathsf{Seq}|$ to be the *length* of Seq; an infinite sequence has infinite length. For a sequence Seq of session indices, and for any set of sessions $\mathcal{S}' \subseteq \mathcal{S}$, denote by $\mathsf{Seq} \mid \mathcal{S}'$ the *restriction* of Seq to indices of sessions in $\mathcal{S}'$. Denote by $\mathsf{Seq} \uparrow \mathcal{S}'$ the shortest prefix of $\mathsf{Seq} \mid \mathcal{S}'$ that includes the indices of all sessions in $\mathcal{S}'$, in the order they appear in this prefix of $\mathsf{Seq} \mid \mathcal{S}'$ and with no repetitions; in case no such prefix exists, $\mathsf{Seq} \uparrow \mathcal{S}'$ results from $\mathsf{Seq} \mid \mathcal{S}'$ by removing repetitions, while, however, preserving the order of the indices. Denote by $\mathsf{Seq} \downarrow \mathcal{S}'$ the remaining suffix of $\mathsf{Seq} \mid \mathcal{S}'$. For example, if $\mathsf{Seq} = 1, 5, 4, 2, 1, 3, 3, 3, 5, 4$ and $\mathcal{S}' = \{1, 3, 4\} \subset \{1, 2, 3, 4, 5\}$, then $\mathsf{Seq} \mid \mathcal{S}' = 1, 4, 1, 3, 3, 3, 4$, $\mathsf{Seq} \uparrow \mathcal{S}' = 1, 4, 3$, and $\mathsf{Seq} \downarrow \mathcal{S}' = 3, 3, 4$.

Fix any even integer d, and choose any integer n that is a multiple of d.[3] We construct a network $G = G(\mathsf{Seq}) = (V(\mathsf{Seq}), E(\mathsf{Seq}))$, as a function of Seq, with a set of sessions $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ laid out on G. For assigning capacities to network edges, we will use two (finite) sequences of real numbers, b and p (for *bottom* and *potential*, resp.), each of length $\frac{d}{2}$, defined recursively as follows.

- $b_0 = p_0 = 0$, $b_1 = 0$, and $p_1 = 2^p$ for some integer $p \geq d$;
- for each index r, $1 < r \leq \frac{d}{2}$, $b_r = b_{r-1} + \frac{p_{r-1}}{2}$ and $p_r = \frac{p_{r-1}}{4}$.

Partition $\mathcal{S}$ into $\frac{n}{d}$ clusters $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n/d}$ so that for each j, $1 \leq j \leq \frac{n}{d}$, $\mathcal{S}_j$ contains sessions $S_{(j-1)d+1}, S_{(j-1)d+2}, \ldots, S_{(j-1)d+d}$; notice that $|\mathcal{S}_j| = d$. For each cluster $\mathcal{S}_j$, we construct a network $G_j = G_j(\mathsf{Seq} \mid \mathcal{S}_j) = (V_j(\mathsf{Seq} \mid \mathcal{S}_j), E_j(\mathsf{Seq} \mid \mathcal{S}_j))$, with $\mathcal{S}_j$ laid out on $G_j(\mathsf{Seq} \mid \mathcal{S}_j)$, so that

$$G(\mathsf{Seq}) = \left(\bigcup_{1 \leq j \leq n/d} V_j(\mathsf{Seq} \mid \mathcal{S}_j), \bigcup_{1 \leq j \leq n/d} E_j(\mathsf{Seq} \mid \mathcal{S}_j) \right);$$

thus, each individual network G_j is a function of the sequence $\mathsf{Seq} \mid \mathcal{S}_j$, and the network G is the resulting composition.

The construction of the network G_j proceeds in a sequence of $\frac{d}{2}$ *epochs*; in epoch r, $1 \leq r \leq \frac{d}{2}$, the network $G_j^{(r)} = (V_j^{(r)}, E_j^{(r)})$ is constructed, so that $G_j = (\bigcup_{1 \leq r \leq d/2} V_j^{(r)}, \bigcup_{1 \leq r \leq d/2} E_j^{(r)})$; thus, the network G_j is the composition of the individual networks $G_j^{(r)}$.

For each r, $1 \leq r \leq \frac{d}{2}$, the construction of $G_j^{(r)}$ uses b_r and p_r as parameters. It also uses the following sets and sequences:

- a set of indices $\mathcal{I}_j^{(r)} \subseteq \mathcal{S}_j$ such that $|\mathcal{I}_j^{(r)}| = d - 2(r-1)$;
- a sequence $\mathsf{Seq}_j^{(r)}$, which is a suffix of $\mathsf{Seq} \mid \mathcal{S}_j$;
- a set $\{i_f^{(r)}, i_l^{(r)}\} \subseteq \mathcal{I}_j^{(r)}$; roughly speaking, $i_f^{(r)}$ and $i_l^{(r)}$ will be defined to be the first and last indices, respectively, of $\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}$, or some of them will be set to arbitrary indices from $\mathcal{I}_j^{(r)}$ in case $|\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}| < 2$.

These sets and sequences are inductively defined as follows. For the basis case where $r = 1$, $\mathcal{I}_j^{(1)} := \mathcal{S}_j$, $\mathsf{Seq}_j^{(1)} := \mathsf{Seq} \mid \mathcal{S}_j$, and the set $\{i_f^{(1)}, i_l^{(1)}\}$ is defined as follows:

(1) Assume first that $\mathsf{Seq}_j^{(1)} = \lambda$, the *empty* sequence, so that $\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)} = \lambda$; then fix $i_f^{(1)}$ and $i_l^{(1)}$ to be any arbitrary indices in $\mathcal{I}_j^{(1)}$.

(2) Now assume that $\mathsf{Seq}_j^{(1)} \neq \lambda$, so that $\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)} \neq \lambda$; there are two cases to consider.
   (a) First, take $|\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)}| = 1$; then $i_f^{(1)} := \mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)}$ and fix $i_l^{(1)}$ to be any arbitrary index in $\mathcal{I}_j^{(1)} \setminus \{i_f^{(1)}\}$.
   (b) Finally, take $|\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)}| > 1$, so that $\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)} = i_f, \ldots, i_l$; then $i_f^{(1)} := i_f$ and $i_l^{(1)} := i_l$.

Informally, $\mathcal{I}_j^{(1)}$ contains indices of all sessions in cluster $\mathcal{S}_j$, while $\mathsf{Seq}_j^{(1)}$ is the restriction of Seq to indices of sessions in cluster $\mathcal{S}_j$; moreover, $i_f^{(1)}$ and $i_l^{(1)}$ are the indices of sessions in cluster $\mathcal{S}_j$ appearing first and last, respectively, in $\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)}$

[3] Standard "padding" techniques can be used to handle the case where n is not a multiple of d.

or some of them will be set to be arbitrary indices if $\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)}$ misses any such indices.

Assume inductively that we have defined $\mathcal{I}_j^{(r-1)}$, $\mathsf{Seq}_j^{(r-1)}$, and $\{i_f^{(r-1)}, i_l^{(r-1)}\}$ for some integer r, where $2 \leq r \leq \frac{d}{2}$. For the induction step, we show how to construct $\mathcal{I}_j^{(r)}$, $\mathsf{Seq}_j^{(r)}$, and $\{i_f^{(r)}, i_l^{(r)}\}$. Define $\mathcal{I}_j^{(r)} := \mathcal{I}_j^{(r-1)} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$, $\mathsf{Seq}_j^{(r)} := (\mathsf{Seq}_j^{(r-1)} \downarrow \mathcal{I}_j^{(r-1)}) \mid \mathcal{I}_j^{(r)}$, and the set $\{i_f^{(r)}, i_l^{(r)}\}$ is defined through a case analysis identical to the one for the basis case.

(1) Assume first that $\mathsf{Seq}_j^{(r)} = \lambda$, so that $\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)} = \lambda$; then fix $i_f^{(r)}, i_l^{(r)}$ to be any arbitrary indices in $\mathcal{I}_j^{(r)}$.

(2) Now assume that $\mathsf{Seq}_j^{(r)} \neq \lambda$, so that $\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)} \neq \lambda$; there are two cases to consider.
   (a) First, take $|\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}| = 1$; then $i_f^{(r)} := \mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}$ and fix $i_l^{(r)}$ to be any arbitrary index in $\mathcal{I}_j^{(r)}$;
   (b) Finally, take $|\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}| > 1$, so that $\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)} = i_f, \ldots, i_l$; then $i_f^{(r)} := i_f$ and $i_l^{(r)} := i_l$.

Informally, $\mathcal{I}_j^{(r)}$ is obtained by removing $i_f^{(r-1)}$ and $i_l^{(r-1)}$ from $\mathcal{I}_j^{(r-1)}$, while $\mathsf{Seq}_j^{(r)}$ results from $\mathsf{Seq}_j^{(r-1)}$ by chopping off its shortest prefix that includes all indices in $\mathcal{I}_j^{(r-1)}$, and restringing the remaining suffix to indices in $\mathcal{I}_j^{(r)}$; moreover, $i_f^{(r)}$ and $i_l^{(r)}$ are the indices of sessions in $\mathcal{I}_j^{(r)}$ that appear first and last, respectively, in this suffix, or some of them will be set to arbitrary indices in case this suffix misses any such indices.

Since two different sessions are extracted from $\mathcal{S}_j$ in each of the $\frac{d}{2}$ epochs, all d sessions in $\mathcal{S}_j$ will eventually be extracted. We now describe the construction of $G_j^{(r)}$, $1 \leq r \leq \frac{d}{2}$:

- sessions $i_f^{(r)}$ and $i_l^{(r)}$ traverse some edge $e_{i_l}^{(r)}$ with $c(e_{i_l}^{(r)}) = 2b_r + \frac{p_r}{2}$;
- for each $i \in \mathcal{I}_j^{(r)} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, sessions $i_f^{(r)}$ and i traverse some edge $e_i^{(r)}$ with $c(e_i^{(r)}) = 2b_r + p_r$.

Informally, $i_f^{(r)}$ shares an edge with every other session in $\mathcal{I}_j^{(r)}$; the capacity of the edge shared with $i_l^{(r)}$ is the smallest, while all other capacities are equal. All other sessions traverse only the edge shared with $i_f^{(r)}$. We finally state an easy to prove property of the construction.

LEMMA 6.1. *For each integer r, where $1 < r \leq \frac{d}{2}$, for each index r' where $1 \leq r' < r$, let $e^{(r)}$ and $e^{(r')}$ be any edges in $E_j^{(r)}$ and $E_j^{(r')}$, respectively. Then $c(e^{(r)}) > c(e^{(r')})$.*

Example. Fix $d = 6$ and choose $n = 12$. Consider the (infinite) elevator sequence

$$\mathsf{ElevSched} = 1, 2, \ldots, 11, 12, 12, 11, \ldots, 2, 1, \ldots, 1, 2, \ldots, 11, 12, 12, 11, \ldots, 2, 1, \ldots .$$

We construct the network $G = G(\mathsf{ElevSched}) = (V(\mathsf{ElevSched}), E(\mathsf{ElevSched}))$ as a function of $\mathsf{ElevSched}$, with a set of sessions $\mathcal{S} = \{S_1, \ldots, S_{12}\}$ laid out on G. Partition $\mathcal{S}$ into $\frac{12}{6} = 2$ clusters $\mathcal{S}_1$ and $\mathcal{S}_2$, each containing six sessions, so that $\mathcal{S}_1 = \{S_1, \ldots, S_6\}$ and $\mathcal{S}_2 = \{S_7, \ldots, S_{12}\}$. Fix $p = 10$, so that $b_1 = 0$ and $p_1 = 2^{10} = 1024$.

For the basis case where $r = 1$, which corresponds to the first epoch, $\mathcal{I}_1^{(1)} = \mathcal{S}_1$, and

$$\begin{aligned}\mathsf{ElevSched}_1^{(1)} &= \mathsf{ElevSched} \mid \mathcal{S}_1 \\ &= 1, 2, \ldots, 6, 6, 5, \ldots, 1, \ldots, 1, 2, \ldots, 6, 6, 5, \ldots, 1, \ldots .\end{aligned}$$

Thus, $\mathsf{ElevSched}_1^{(1)} \uparrow \mathcal{S}_1^{(1)} = 1, 2, \ldots, 6$, so that $i_f^{(1)} = 1$ and $i_l^{(1)} = 6$. We continue to describe the construction of the network $G_1^{(1)}$:

• sessions 1 and 6 traverse edge $e_6^{(1)}$ with $c(e_6^{(1)}) = 2b_1 + \frac{p_1}{2} = 512$;

• for each $i \in \{2, 3, 4, 5\}$, sessions 1 and i traverse $e_i^{(1)}$ with $c(e_i^{(1)}) = 2b_1 + p_1 = 1024$.

The construction of the network $G_2^{(1)}$ is similar; it can be found in Figure 1. We proceed to the case $r = 2$, corresponding to the second epoch, where $\mathcal{I}_1^{(2)} = \mathcal{I}_1^{(1)} \setminus \{1, 6\} = \{2, 3, 4, 5\}$ and

$$\begin{aligned}\mathsf{ElevSched}_1^{(2)} &= \left(\mathsf{ElevSched}_1^{(1)} \downarrow \mathcal{I}_1^{(1)}\right) \mid \mathcal{I}_1^{(2)} \\ &= 5, 4, 3, 2, 2, 3, 4, 5, \ldots, 5, 4, 3, 2, 2, 3, 4, 5, \ldots .\end{aligned}$$

Thus, $\mathsf{ElevSched}_1^{(2)} \uparrow \mathcal{I}_1^{(2)} = 5, 4, 3, 2$, so that $i_f^{(2)} = 5$ and $i_l^{(2)} = 2$. We continue to describe the construction of $G_1^{(2)}$:

• sessions 5 and 2 traverse edge $e_2^{(2)}$ with $c(e_2^{(2)}) = 2b_2 + \frac{p_2}{2} = 1152$;

• for each $i \in \{3, 4\}$, sessions 5 and i traverse edge $e_i^{(2)}$ with $c(e_i^{(2)}) = 2b_2 + p_2 = 1280$.

The construction of the network $G_2^{(2)}$ is similar; it can be found in Figure 1. We proceed to the case $r = 3$, corresponding to the third epoch, where $\mathcal{I}_1^{(3)} = \mathcal{I}_1^{(2)} \setminus \{2, 5\} = \{3, 4\}$ and

$$\begin{aligned}\mathsf{ElevSched}_1^{(3)} &= \left(\mathsf{ElevSched}_1^{(2)} \downarrow \mathcal{I}_1^{(2)}\right) \mid \mathcal{I}_1^{(3)} \\ &= 3, 4, 4, 3, \ldots, 3, 4, 4, 3, \ldots .\end{aligned}$$

Thus, $\mathsf{ElevSched}_1^{(3)} \uparrow \mathcal{I}_1^{(3)} = 3, 4$, so that $i_f^{(3)} = 3$ and $i_l^{(3)} = 4$. We continue to describe the construction of $G_1^{(3)}$:

• sessions 3 and 4 traverse edge $e_4^{(3)}$ with $c(e_4^{(3)}) = 2b_3 + \frac{p_3}{2} = 2 \cdot 640 + \frac{64}{2} = 1312$.

The construction of the network $G_2^{(3)}$ is similar; it can be found in Figure 1, which also depicts the complete network $G(\mathsf{ElevSched})$. □

7. Lower bounds.

7.1. Oblivious algorithms. We present a lower bound of $\Omega(dn)$ on the convergence complexity of any optimistic, oblivious, and bottleneck algorithm $\mathsf{Alg} = \langle \mathsf{Sched}, \mathsf{Term} \rangle$ that computes the max-min fair rate vector. The proof uses the network $G(\mathsf{Sched})$ constructed in section 6. We start with two immediate technical lemmas that quantify $\Delta_Q(i, e)$ in case edge e is traversed by only two sessions that remain active in state Q. (These lemmas will be used for Proposition 7.3.)

LEMMA 7.1. *For an edge e traversed only by sessions $i, i' \in \mathcal{A}_Q \mid e$, $\Delta_Q(i, e) = c(e) - r_Q(i) - \min\{\frac{c(e)}{2}, r_Q(i')\}$.*

Since in the setting of Claim 7.1, $c(e) - r_Q(i) - r_Q(i') = resid_Q(e)$, Lemma 7.2 follows.

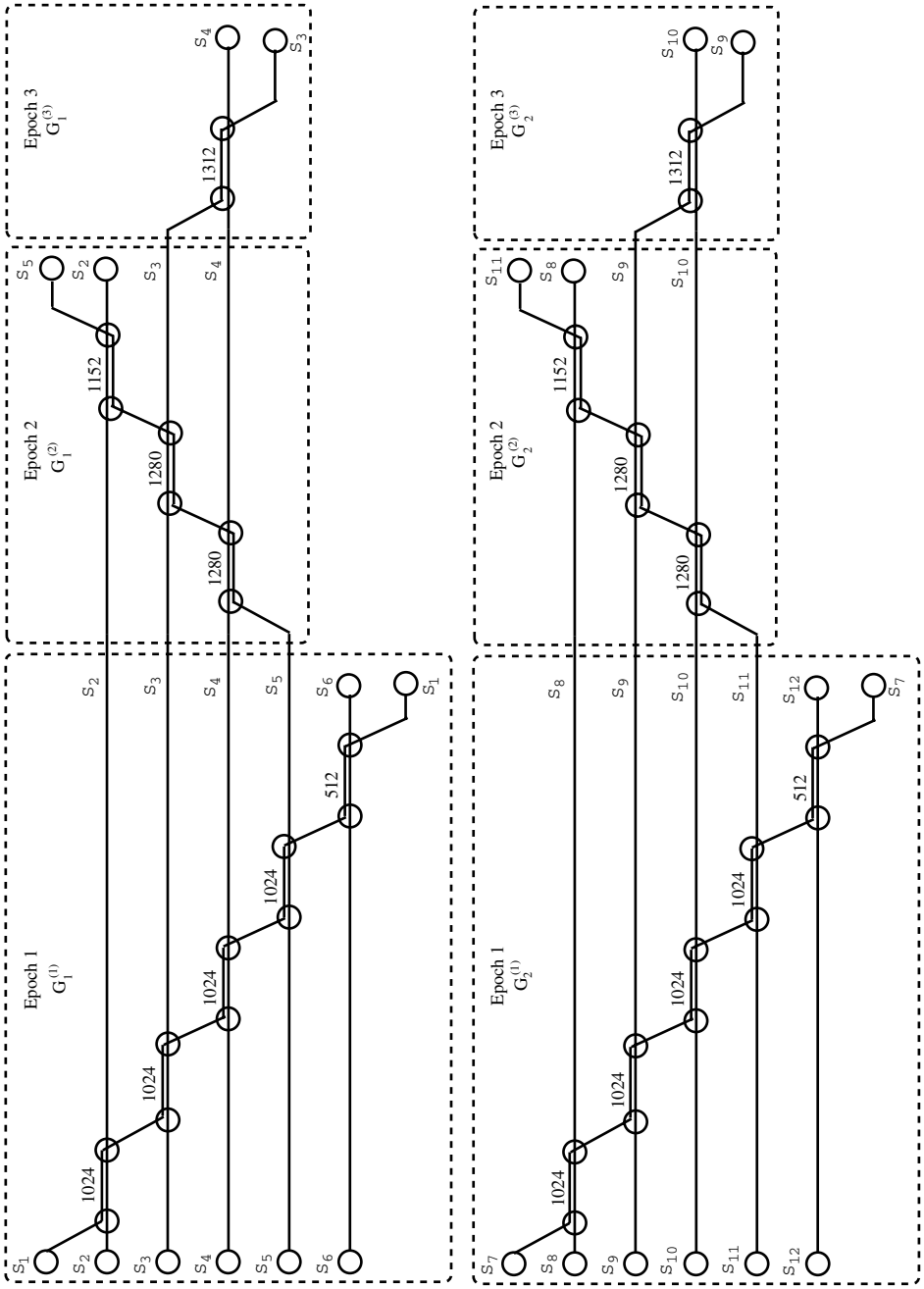

FIG. 1. *The network* $G(\mathsf{ElevSched})$.

LEMMA 7.2. *For an edge* e *traversed only by sessions* $i, i' \in \mathcal{A}_Q \mid e$, $\Delta_Q(i,e) \geq resid_Q(e)$.

We restrict our attention to the execution α of Alg on the network $G(\mathsf{Sched} \mid \mathcal{S}_j)$ with any particular cluster $\mathcal{S}_j$; for notational simplicity, we shall abuse notation and use $G(\mathsf{Sched})$ to denote $G(\mathsf{Sched} \mid \mathcal{S}_j)$ and $\mathcal{S}$ to denote $\mathcal{S}_j$.

For an execution α, for any indices l_1 and l_2, $0 \leq l_1 \leq l_2$, define the set $\mathcal{S} \mid_\alpha (Q_{l_1}, Q_{l_2}] \subseteq \mathcal{S}$ to be $\mathcal{S} \mid_\alpha (Q_{l_1}, Q_{l_2}] = \left\{ i_l \mid l_1 < l \leq l_2 \text{ and } i_l \in \mathcal{A}_{Q_{l_1}} \right\}$; roughly speaking, $\mathcal{S} \mid_\alpha (Q_{l_1}, Q_{l_2}]$ contains indices of all sessions active in Q_{l_1} that are scheduled in front of any state following Q_{l_1} and up to and including Q_{l_2}. Notice that if $l_1 = l_2$, $\mathcal{S} \mid_\alpha (Q_{l_1}, Q_{l_2}] = \emptyset$. For an execution α, for any index $l_1 \geq 0$, define the set $\mathcal{S} \mid_\alpha (Q_{l_1}, \infty) \subseteq \mathcal{S}$ to be $\mathcal{S} \mid_\alpha (Q_{l_1}, \infty) = \left\{ i_l \mid l_1 < l \text{ and } i_l \in \mathcal{A}_{Q_{l_1}} \right\}$; roughly speaking, $\mathcal{S} \mid_\alpha (Q_{l_1}, \infty)$ contains indices of all sessions active in Q_{l_1} that are scheduled in front of any state following Q_{l_1}. Define $\mathcal{S} \mid_\alpha (Q_{l_1}, Q_{l_2}] \mid e$ and $\mathcal{S} \mid_\alpha (Q_{l_1}, \infty) \mid e$ in the natural way. Recall that for any state Q_l, $\widehat{l}$ is the least integer $l' \geq l$ such that $\mathcal{S} \mid_\alpha (Q_l, Q_{l'}] = \mathcal{A}_{Q_l}$, or infinite if no such integer exists.

Define inductively the index sequence $l_0, l_1, \ldots$ as follows. For the basis case, $l_0 = 0$. Assume inductively that for any integer $r \geq 1$, we have defined $l_1, \ldots, l_{r-1}$. For the induction step, define $l_r = \widehat{l_{r-1}}$. Define also the execution fragments $\alpha^{(1)}, \alpha^{(2)}, \ldots$, where for any integer $r \geq 1$, $\alpha^{(r)} = Q_{l_{r-1}}, \ldots, i_{l_r}, Q_{l_r}$. Call each $\alpha^{(r)}$, $r \geq 1$, an *execution epoch in* α. Note that in case $l_r = \infty$, for any integer $r \geq 1$, $\alpha^{(r)}$ is the infinite suffix of α following state $Q_{l_{r-1}}$. Thus, write $\alpha = \alpha^{(1)} \cdot \alpha^{(2)} \ldots$. We remark that in case α is infinite, the number of execution epochs in α can still be finite if (and only if) there exists some integer $r \geq 0$ such that $l_r = \infty$. To simplify notation, denote each state Q_{l_r}, $r \geq 0$, as $Q^{(r)}$. Thus, $Q^{(r)}$ is the latest state in execution epoch $\alpha^{(r)}$ of α. (Note that $Q^{(r)}$ exists if and only if $l_r < \infty$.)

The backbone of our analysis is a technical proposition (Proposition 7.3) that describes the states of execution α. The first part of Proposition 7.3 (part (A)) deals with each state starting from $Q^{(r-1)}$, $1 \leq r \leq \frac{d}{2}$, such that not all active sessions in $\overrightarrow{Q^{(r-1)}}$ have yet been scheduled until this state. Thus, $Q^{(r)}$ would be the state immediately following this sequence of states; we will later show that $Q^{(r)}$ is well defined. Observe that the states considered in part (A) for any particular integer r, $1 \leq r \leq \frac{d}{2}$, are precisely the states in execution epoch $\alpha^{(r)}$ excluding state $Q^{(r)}$. Part (B) explores properties of state $Q^{(r)}$.

PROPOSITION 7.3 (properties of execution α). *For each integer r, $1 \leq r \leq \frac{d}{2}$, the following hold for states in execution epoch $\alpha^{(r)}$:*

(A) (properties of states from $Q^{(r-1)}$ to $\overleftarrow{Q^{(r)}}$) *Consider any state Q such that $Q^{(r-1)} \stackrel{\alpha}{\longmapsto} Q$ and $\mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \not\subseteq \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$. Then the following conditions hold:*

(1) *for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$,*

$$r_Q(i) = \begin{cases} b_r + \frac{p_r}{2} & \text{if } i \in \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right], \\ b_{r-1} + \frac{p_{r-1}}{2} & \text{otherwise;} \end{cases}$$

(2) *for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$,*

$$FS_Q(e_i^{(r)}) = \begin{cases} b_r + \frac{p_r}{2} & \text{if } i \notin \left\{ i_f^{(r)}, i_l^{(r)} \right\}, \\ b_r + \frac{p_r}{4} & \text{if } i = i_l^{(r)}; \end{cases}$$

(3) *for the edges of $G(\mathsf{Sched})$, it holds, for $Q \neq Q^{(r-1)}$, that*
   (a) *edge $e_{i_l^{(r)}}^{(r)}$ is a bottleneck edge for state Q;*

(b) *for each session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$,

$$FS_Q(e_i^{(r)}) \neq MFS_Q(i_f^{(r)}),$$

so that edge $e_i^{(r)}$ *is not a bottleneck edge for state* Q*;*

(c) *for any integer* r', $1 \leq r' < r$, *and for any edge* $e^{(r')} \in E^{(r')}$ (Sched) *traversed by session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $FS_Q(e^{(r')}) \neq MFS_Q(i)$, *so that edge* $e^{(r')}$ *is not a bottleneck edge for state* Q*;*

(d) *for any integer* r', $r < r' \leq \frac{d}{2}$, *and for any edge* $e^{(r')} \in E^{(r')}$ (Sched) *traversed by session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$,

$$FS_Q(e^{(r')}) \neq MFS_Q(i),$$

so that $e^{(r')}$ *is not a bottleneck edge for state* Q*;*

(4) *for each session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $i \notin \mathsf{Term}(Q)$*;*

(5) *for each session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$ *scheduled in front of state* $\overrightarrow{Q}$,

$$\Delta_Q(i) = \begin{cases} \frac{p_r}{2} & \text{if } i \neq i_l^{(r)}, \\ \frac{p_r}{4} & \text{otherwise.} \end{cases}$$

(B) (properties of state $Q^{(r)}$) *The following conditions hold for state* $Q^{(r)}$*:*

(1) $l_r < \infty$*;*

(2) *for each session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$,

$$r_{Q^{(r)}}(i) = \begin{cases} b_r + \frac{p_r}{4} & \text{if } i \in \left\{i_f^{(r)} i_l^{(r)}\right\}, \\ b_r + \frac{p_r}{2} & \text{if } i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \left\{i_f^{(r)}, i_l^{(r)}\right\}; \end{cases}$$

(3) *for each session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $i \in \{i_f^{(r)}, i_l^{(r)}\}$ *if and only if* $i \in \mathsf{Term}(Q^{(r)})$*;*

(4) *for each session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$ *and for any integer* r', $1 \leq r' \leq r$, *for any edge* $e_i^{(r')}$,

(a) $resid_{Q^{(r)}}(e_i^{(r')}) \geq \frac{p_r}{4}$*;*

(b) $FS_{\overrightarrow{Q^{(r)}}}(e_i^{(r')}) \geq b_r + \frac{3p_r}{4}$.

Proposition 7.3 deals mainly with sessions active in state $\overrightarrow{Q^{(r-1)}}$ for any index r, $1 \leq r \leq \frac{d}{2}$. We start with an informal description of the conditions in part (A). Condition (A/1) determines rates of active sessions in state Q. Condition (A/2) determines the fair shares of all edges in epoch r; condition (A/3) establishes that edge $e_{i_l^{(r)}}^{(r)}$ is the only bottleneck edge for state Q. Condition (A/4) guarantees that no session is terminated in state Q. Finally, condition (A/5) determines the increase in state Q for sessions active in $\overrightarrow{Q^{(r-1)}}$ that are not yet scheduled. We now continue with the conditions in part (B). Condition (B/1) asserts that all active sessions in

state $\overrightarrow{Q^{(r-1)}}$ must be scheduled in execution epoch $\alpha^{(r)}$; condition (B/2) specifies their rates upon completion of $\alpha^{(r)}$. Moreover, condition (B/3) determines which of these sessions are terminated upon completion of $\alpha^{(r)}$. Condition (B/4) provides lower bounds on the residual capacity and fair share of some edges from previous epochs upon completion of $\alpha^{(r)}$.

We note that conditions (B/1) and (B/3) will suffice by themselves to imply the lower bound. However, the rather technical remaining conditions are necessary to assume inductively in the proof of conditions (B/1) and (B/3). To simplify notation, we will denote $e^{(r)}_{i_l^{(r)}}$ as $e_l^{(r)}$.

Proof. The proof is by induction on r. For the sake of shortening the proof, we merge the proof for the basis case (where $r = 1$) and the proof for the induction step; thus, the case $r = 1$ will be treated separately (where needed) along the proof of the induction step.

We assume, as our induction hypothesis, that the claims hold for all integers less than some fixed integer r, $1 \leq r \leq \frac{d}{2}$. Notice that if $r = 1$, the induction hypothesis is empty. We proceed to the induction step, where we prove the claims for r.

Proof of part (A). The proof is by induction on Q. For the sake of shortening the proof, we merge again the proof for the basis case (where $Q = Q^{(r-1)}$) and the proof for the induction step; thus, the case $Q = Q^{(r-1)}$ will be treated separately (where needed) along the proof for the induction step (on states).

Fix any state Q such that $Q^{(r-1)} \overset{\alpha}{\longmapsto} Q$ and $\mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \not\subseteq \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$, and assume that for each state Q' such that $Q^{(r-1)} \overset{\alpha}{\longmapsto} Q' \xrightarrow{\alpha} Q$, the claims of part (A) hold for Q'; thus, we assume, as our induction hypothesis, that the claims of part (A) hold for all states from $Q^{(r-1)}$ through but not including state Q. Notice that if $Q = Q^{(r-1)}$, the induction hypothesis is empty. We now proceed with the induction step, where we prove the claims for Q.

Proof of (A/1). There are two cases to consider.

(1) Assume first that $Q = Q^{(r-1)}$. Then $\mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right] = \emptyset$. In case $r = 1$, $Q = Q_0$, and condition (A/1) holds trivially since all session rates are initially zero and $b_0 = p_0 = 0$. So assume $r > 1$. By the induction hypothesis of induction on r (condition (B/3)), $\mathcal{A}_{\overrightarrow{Q^{(r-1)}}} = \mathcal{A}_{\overrightarrow{Q^{(r-2)}}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$. Hence, condition (A/1) follows from the induction hypothesis of induction on r (condition (B/2)).

(2) Now assume that $Q \neq Q^{(r-1)}$. We proceed by case analysis on i_Q (the session scheduled in front of state Q).

   (a) Assume first that $i_Q \notin \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. (Notice that this case need not be considered when $r = 1$, since all sessions are active in $\overrightarrow{Q^{(0)}} = Q_1$.) Then $\mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right] = \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}\,]$. Since, in addition, no session rates change from $\overleftarrow{Q}$ to Q, the claim follows inductively. So we proceed to the cases where $i_Q \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$.

   (b) Next assume that $i_Q = i_f^{(r)}$. There are two subcases to consider.

      (i) First, take $i_f^{(r)} \notin \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}\,]$; that is, $i_f^{(r)}$ has not been scheduled in front of any state between $Q^{(r-1)}$ and Q. Recall that

$i_f^{(r)}$ is the session scheduled first (following state $Q^{(r-1)}$) among active sessions in $\overrightarrow{Q^{(r-1)}}$; so, for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $i \notin \mathcal{S} \mid_\alpha (Q^{(r-1)}, Q]$. Thus, by the induction hypothesis of induction on states (condition (A/1)), $r_{\overleftarrow{Q}}(i) = b_{r-1} + \frac{p_{r-1}}{2} = b_r$ (by recursive definition of b_r). In particular, $r_{\overleftarrow{Q}}(i_f^{(r)}) = b_r$. By the induction hypothesis of induction on states (condition (A/5)), $\Delta_{\overleftarrow{Q}}(i_f^{(r)}) = \frac{p_r}{2}$. Thus, by the `update` operation, $r_Q(i_f^{(r)}) = r_{\overleftarrow{Q}}(i_f^{(r)}) + \Delta_{\overleftarrow{Q}}(i_f^{(r)}) = b_r + \frac{p_r}{2}$.

Since $p_r \neq 0$, it follows that for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}\}$, $r_{\overleftarrow{Q}}(i) < r_Q(i_f^{(r)})$. Thus, by the `update` operation, $r_Q(i) = r_{\overleftarrow{Q}}(i) = b_{r-1} + \frac{p_{r-1}}{2}$. Since $\mathcal{S} \mid_\alpha (Q^{(r-1)}, Q] = \{i_f^{(r)}\}$, it follows that for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$,

$$r_Q(i) = \begin{cases} b_r + \frac{p_r}{2} & \text{if } i \in \mathcal{S} \mid_\alpha \big(Q^{(r-1)}, Q\big], \\ b_{r-1} + \frac{p_{r-1}}{2} & \text{otherwise.} \end{cases}$$

(ii) Now take $i_f^{(r)} \in \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}]$. Then $\mathcal{S} \mid_\alpha (Q^{(r-1)}, Q] = \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}]$. Clearly, $i_l^{(r)} \notin \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}]$, while $i_f^{(r)} \in \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}]$. By construction of $G(\mathsf{Sched})$, session $i_f^{(r)}$ traverses edge $e_l^{(r)}$ with $c(e_l^{(r)}) = 2b_r + \frac{p_r}{2}$, as does session $i_l^{(r)}$. By the induction hypothesis of induction on states (condition (A/1)), $r_{\overleftarrow{Q}}(i_f^{(r)}) = b_r + \frac{p_r}{2}$, while $r_{\overleftarrow{Q}}(i_l^{(r)}) = b_{r-1} + \frac{p_{r-1}}{2} = b_r$. By Lemma 7.1,

$$\begin{aligned} &\Delta_{\overleftarrow{Q}}(i_f^{(r)}, e_l^{(r)}) \\ &\quad = c(e_l^{(r)}) - r_{\overleftarrow{Q}}(i_f^{(r)}) - \min\left\{ \frac{c(e_l^{(r)})}{2}, r_{\overleftarrow{Q}}(i_l^{(r)}) \right\} \\ &\quad = 2\,b_r + \frac{p_r}{2} - \left(b_r + \frac{p_r}{2}\right) - \min\left\{b_r + \frac{p_r}{2}, b_r\right\} \\ &\quad = 0, \end{aligned}$$

so that $\Delta_{\overleftarrow{Q}}(i_f^{(r)}) = 0$. Thus, by the `update` operation, for each session $i \in \mathcal{A}_{\overleftarrow{Q}}$, $r_Q(i) = r_{\overleftarrow{Q}}(i)$. By the induction hypothesis of induction on states (condition (A/4)), it follows that $\mathcal{A}_{\overleftarrow{Q}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Hence, the induction hypothesis of induction on states (condition (A/1)) implies that for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$,

$$r_Q(i) = \begin{cases} b_r + \frac{p_r}{2} & \text{if } i \in \mathcal{S} \mid_\alpha \big(Q^{(r-1)}, Q\big], \\ b_{r-1} + \frac{p_{r-1}}{2} & \text{otherwise.} \end{cases}$$

(c) Assume now that $i_Q \neq i_f^{(r)}$. There are two subcases to consider.

(i) First consider the case where $i_Q \notin \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}]$. By the induction hypothesis of induction on states (condition (A/1)), it follows that $r_{\overleftarrow{Q}}(i_Q) = b_{r-1} + \frac{p_{r-1}}{2} = b_r$; moreover, by condition (A/5), $\Delta_{\overleftarrow{Q}}(i_Q) = \frac{p_r}{2}$. Thus, by the `update` operation, $r_Q(i_Q) = r_{\overleftarrow{Q}}(i_Q) + \Delta_{\overleftarrow{Q}}(i_Q) = b_r + \frac{p_r}{2}$.
By induction hypothesis of induction on states (condition (A/1)), it follows that for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_Q\}$, $r_{\overleftarrow{Q}}(i) \leq b_r + \frac{p_r}{2}$; hence, $r_{\overleftarrow{Q}}(i) \leq r_Q(i_Q)$. Thus, by the `update` operation, $r_Q(i) = r_{\overleftarrow{Q}}(i)$. Since $\mathcal{S} \mid_\alpha (Q^{(r-1)}, Q] = \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}] \bigcup \{i_Q\}$, the induction hypothesis of induction on states (condition (A/1)) implies now that for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$,

$$r_Q(i) = \begin{cases} b_r + \frac{p_r}{2} & \text{if } i \in \mathcal{S} \mid_\alpha \big(Q^{(r-1)}, Q\big], \\ b_{r-1} + \frac{p_{r-1}}{2} & \text{otherwise.} \end{cases}$$

(ii) Finally, consider the case where $i_Q \in \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}] \setminus \{i_f^{(r)}\}$. Then $\mathcal{S} \mid_\alpha (Q^{(r-1)}, Q] = \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}]$. Clearly, both $i_Q, i_f^{(r)} \in \mathcal{S} \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q}]$. Thus, by the induction hypothesis of induction on states (condition (A/1)), $r_{\overleftarrow{Q}}(i_Q) = r_{\overleftarrow{Q}}(i_f^{(r)}) = b_r + \frac{p_r}{2}$. By construction of $G(\mathsf{Sched})$, $c(e_{i_Q}^{(r)}) = 2\,b_r + p_r$. It follows that $r_{\overleftarrow{Q}}(i_Q) = r_{\overleftarrow{Q}}(i_f^{(r)}) = \frac{c(e_{i_Q}^{(r)})}{2}$. Hence, Lemma 7.1 implies that $\Delta_{\overleftarrow{Q}}(i_Q, e_{i_Q}^{(r)}) = 0$, so that $\Delta_{\overleftarrow{Q}}(i_Q) = 0$. Thus, by the `update` operation, for each session $i \in \mathcal{A}_{\overleftarrow{Q}}$, $r_Q(i) = r_{\overleftarrow{Q}}(i)$. By the induction hypothesis of induction on states (condition (A/4)), it follows that $\mathcal{A}_{\overleftarrow{Q}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Hence, the induction hypothesis of induction on states (condition (A/1)) implies that for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$,

$$r_Q(i) = \begin{cases} b_r + \frac{p_r}{2} & \text{if } i \in \mathcal{S} \mid_\alpha \big(Q^{(r-1)}, Q\big], \\ b_{r-1} + \frac{p_{r-1}}{2} & \text{otherwise.} \end{cases}$$ □

Proof of (A/2). First consider edge $e_i^{(r)}$ for any session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, with capacity $c(e_i^{(r)}) = 2\,b_r + p_r$, which is traversed by sessions $i_f^{(r)}$ and i. Either by definition of Q_0 in case $Q = Q^{(r-1)}$ and $r = 1$, or by the induction hypothesis of induction on r (condition (B/3)) in case $Q = Q^{(r-1)}$ and $r \neq 1$, or by the induction hypothesis of induction on states (condition (A/4)) in case $Q \neq Q^{(r-1)}$, it follows that

both $i_f^{(r)}, i \in \mathcal{A}_Q$, so that $\mathcal{A}_Q \mid e_i^{(r)} = \{i_f^{(r)}, i\}$. Hence,

$$\begin{aligned} FS_Q(e_i^{(r)}) &= \frac{c(e_i^{(r)}) - allot_Q(e_i^{(r)})}{|\mathcal{A}_Q \mid e_i^{(r)}|} \\ &= \frac{c(e_i^{(r)})}{2} \\ &= b_r + \frac{p_r}{2}. \end{aligned}$$

Now consider edge $e_l^{(r)}$ with capacity $c(e_l^{(r)}) = 2b_r + \frac{p_r}{2}$, which is traversed by sessions $i_f^{(r)}$ and $i_l^{(r)}$. Either by definition of Q_0 in case $Q = Q^{(r-1)}$ and $r = 1$, or by the induction hypothesis of induction on r (condition (B/3)) in case $Q = Q^{(r-1)}$ and $r \neq 1$, or by the induction hypothesis of induction on states (condition (A/4)) in case $Q \neq Q^{(r-1)}$, it follows that both $i_f^{(r)}, i_l^{(r)} \in \mathcal{A}_Q$, so that $\mathcal{A}_Q \mid e_l^{(r)} = \{i_f^{(r)}, i_l^{(r)}\}$. Hence,

$$\begin{aligned} FS_Q(e_l^{(r)}) &= \frac{c(e_l^{(r)}) - allot_Q(e_l^{(r)})}{\left|\mathcal{A}_Q \mid e_l^{(r)}\right|} \\ &= \frac{c(e_l^{(r)})}{2} \\ &= b_r + \frac{p_r}{4}. \qquad \square \end{aligned}$$

Proof of (A/3/a). By construction of $G(\mathsf{Sched})$, edge $e_l^{(r)}$ is traversed by sessions $i_f^{(r)}$ and $i_l^{(r)}$. We prove that each of them receives its minimum fair share on edge $e_l^{(r)}$.

(1) First, take session $i_f^{(r)}$, which traverses the following:
- edge $e_l^{(r)}$; by condition (A/2) shown above, $FS_Q(e_l^{(r)}) = b_r + \frac{p_r}{4}$;
- edge $e_i^{(r)}$ for each session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$; by condition (A/2) shown above, $FS_Q(e_i^{(r)}) = b_r + \frac{p_r}{2}$;
- edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ for each integer r', $1 \leq r' < r$, in case $r > 1$. By the induction hypothesis of induction on r (condition (B/4/b)), $FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e^{(r')}) \geq b_{r-1} + \frac{3\,p_{r-1}}{4} = b_r + p_r$ (by recursive definition of b_r and p_r). Since $Q \neq Q^{(r-1)}$, the induction hypothesis of induction on states (condition (A/4)) is nonempty and implies that $\mathcal{A}_Q = \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. It follows that $FS_Q(e^{(r')}) = FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e^{(r')})$, so that $FS_Q(e^{(r')}) \geq b_r + p_r$.

Hence, $MFS_Q(i_f^{(r)}) = b_r + \frac{p_r}{4}$, so that $MFS_Q(i_f^{(r)}) = FS_Q(e_l^{(r)})$.

(2) Now take session $i_l^{(r)}$, which traverses the following:
- edge $e_l^{(r)}$; by condition (A/2) shown above, $FS_Q(e_l^{(r)}) = b_r + \frac{p_r}{4}$;
- edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ for each integer r', $1 \leq r' < r$, in case $r > 1$. By the induction hypothesis of induction on r (condition (B/4/b)), $FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e^{(r')}) \geq b_{r-1} + \frac{3\,p_{r-1}}{4} = b_r + p_r$. Since $Q \neq Q^{(r-1)}$, the induction hypothesis of induction on states (condition (A/4)) implies that

$\mathcal{A}_Q = \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. It follows that $FS_Q(e^{(r')}) = FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e^{(r')})$, so that $FS_Q(e^{(r')}) \geq b_r + p_r$.

These imply that $MFS_Q(i_l^{(r)}) = b_r + \frac{p_r}{4}$, so that $MFS_Q(i_l^{(r)}) = FS_Q(e_l^{(r)})$. Hence $MFS_Q(i_f^{(r)}) = MFS_Q(i_l^{(r)}) = FS_Q(e_l^{(r)})$, so that $e_l^{(r)}$ is a bottleneck edge for Q. □

Proof of (A/3/b). Consider edge $e_i^{(r)}$ for any session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, which is traversed by sessions $i_f^{(r)}$ and i. By condition (A/3/a) shown above, edge $e_l^{(r)}$, traversed by session $i_f^{(r)}$, is a bottleneck edge for state Q, so that $MFS_Q(i_f^{(r)}) = FS_Q(e_l^{(r)})$. By condition (A/2) shown above, $FS_Q(e_l^{(r)}) = b_r + \frac{p_r}{4}$ and $FS_Q(e_i^{(r)}) = b_r + \frac{p_r}{2}$. Since $p_r \neq 0$, it follows that $FS_Q(e_l^{(r)}) < FS_Q(e_i^{(r)})$, so that $MFS_Q(i_f^{(r)}) < FS_Q(e_i^{(r)})$. It follows that $e_i^{(r)}$ is not a bottleneck edge for Q. □

Proof of (A/3/c). The claim holds vacuously in case $r = 1$. So assume $r > 1$. Consider any edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ for any integer r', $1 \leq r' < r$, traversed by some session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. By the induction hypothesis of induction on r (condition (B/4/b)), $FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e^{(r')}) \geq b_{r-1} + \frac{3\,p_{r-1}}{4}$. Since $Q \neq Q^{(r-1)}$, the induction hypothesis of induction on states (condition (A/4)) is nonempty and implies that $\mathcal{A}_Q = \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. It follows that $FS_Q(e^{(r')}) = FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e^{(r')})$, so that $FS_Q(e^{(r')}) \geq p_{r-1} + \frac{3\,p_{r-1}}{4} = b_r + p_r$. Session i also traverses edge $e_i^{(r)}$. By condition (A/2) shown above, $FS_Q(e_i^{(r)}) \leq b_r + \frac{p_r}{2}$.

Since $p_r \neq 0$, it follows that $FS_Q(e^{(r')}) > FS_Q(e_i^{(r)})$, so that $FS_Q(e^{(r')}) > MFS_Q(i)$. It follows that $e^{(r')}$ is not a bottleneck edge for Q. □

Proof of (A/3/d). Consider any edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ for any integer r', $r < r' \leq \frac{d}{2}$. Take any session i traversing $e^{(r')}$; by construction of $G(\mathsf{Sched})$, $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, and i traverses also $e_i^{(r)}$ with $c(e_i^{(r)}) = 2\,b_r + p_r$. By condition (A/2) shown above, $FS_Q(e_i^{(r)}) = b_r + \frac{p_r}{2} = \frac{c(e_i^{(r)})}{2}$. Since $Q \neq Q^{(r-1)}$, the induction hypothesis of induction on states (condition (A/4)) is nonempty and implies that both sessions traversing $e^{(r')}$ are active in Q. So, $FS_Q(e^{(r')}) = \frac{c(e^{(r')})}{2}$.

By Lemma 6.1, $c(e_i^{(r)}) < c(e^{(r')})$. So, $FS_Q(e_i^{(r)}) < FS_Q(e^{(r')})$. By definition of minimum fair share, $MFS_Q(i) < FS_Q(e^{(r')})$. Hence, $e^{(r')}$ is not a bottleneck edge for Q. □

Proof of (A/4). If $Q = Q^{(r-1)}$, the claim holds trivially. So assume $Q \neq Q^{(r-1)}$. By condition (A/3/b) shown above, any edge $e_i^{(r)}$ with $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$ is not a bottleneck edge for Q. Moreover, by conditions (A/3/c) and (A/3/d) shown above, neither is any edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ with $1 \leq r' < r$ or $r < r' \leq \frac{d}{2}$. Since Alg is bottleneck, none of these edges causes the termination of a session in Q. Thus, it remains to prove only that edge $e^{(r)}_{i_l^{(r)}}$ does not cause either the termination of any of the sessions $i_f^{(r)}$ and $i_l^{(r)}$ traversing it.

Since $p_r \neq 0$, conditions (A/1) and (A/2) shown above imply that $r_Q(i_f^{(r)}) \neq FS_Q(e_l^{(r)})$ and $r_Q(i_l^{(r)}) \neq FS_Q(e_l^{(r)})$. Since Alg is bottleneck, it follows that $i_f^{(r)}, i_l^{(r)} \notin \mathsf{Term}(Q)$. □

Proof of (A/5). The proof is obtained by case analysis on i.

1. Assume first that $i = i_f^{(r)}$. We first prove by case analysis on Q that, in this case, for each session $k \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $r_Q(k) = b_r$. (This will be useful for some later calculations.)
   - First take $Q = Q^{(r-1)}$. In case $r = 1$ where $Q = Q_0$, all session rates are zero in Q_0. Since $b_0 = p_0 = 0$, it follows that $r_Q(k) = b_{r-1} + \frac{p_{r-1}}{2}$. Assume now that $r > 1$. Since $k \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, by construction of $G(\mathsf{Sched})$ and by the induction hypothesis of induction on r (condition (B/3)), $k \notin \{i_f^{(r-1)}, i_l^{(r-1)}\}$. Thus, the induction hypothesis of induction on r (condition (B/2)) implies that $r_Q(k) = b_{r-1} + \frac{p_{r-1}}{2}$.
   - Now take $Q \neq Q^{(r-1)}$. Since $i_f^{(r)} \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$, $k \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$ as well. So, by condition (A/1) shown above, $r_Q(k) = b_{r-1} + \frac{p_{r-1}}{2}$.

   So, in all cases, $r_Q(k) = b_{r-1} + \frac{p_{r-1}}{2} = b_r$. By construction of $G(\mathsf{Sched})$, $i_f^{(r)}$ traverses
   - edge $e_l^{(r)}$ with $c(e_l^{(r)}) = 2b_r + \frac{p_r}{2}$;
   - edge $e_k^{(r)}$ for each session $k \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, with $c(e_k^{(r)}) = 2b_r + p_r$;
   - edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ for each index r', $1 \leq r' < r$, in case $r > 1$.

   We next calculate separately the increases for $i_f^{(r)}$ in Q allowed by these edges.

   (a) First consider edge $e_l^{(r)}$, which is also traversed by $i_l^{(r)} \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. By Lemma 7.1,

$$\begin{aligned}\Delta_Q(i_f^{(r)}, e_l^{(r)}) &= c(e_l^{(r)}) - r_Q(i_f^{(r)}) - \min\left\{\frac{c((e_l^{(r)})}{2}, r_Q(i_l^{(r)})\right\} \\ &= 2b_r + \frac{p_r}{2} - b_r - \min\left\{b_r + \frac{p_r}{4}, b_r\right\} \\ &= \frac{p_r}{2}.\end{aligned}$$

   (b) Next consider any edge $e_k^{(r)}$ for any session $k \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, which is also traversed by session k. By Lemma 7.1,

$$\begin{aligned}\Delta_Q(i_f^{(r)}, e_k^{(r)}) &= c(e_k^{(r)}) - r_Q(i_f^{(r)}) - \min\{c(e_k^{(r)}), r_Q(k)\} \\ &= 2\,b_r + p_r - b_r - \min\left\{b_r + \frac{p_r}{2}, b_r\right\} \\ &= p_r.\end{aligned}$$

   (c) Finally, consider edge $e^{(r')}$ for any integer r', $1 \leq r' < r$, in case $r > 1$. By the induction hypothesis of induction on r (condition (B/4/a)), $resid_{Q^{(r-1)}}(e^{(r')}) \geq \frac{p_{r-1}}{4}$. Recall that $i_f^{(r)}$ is the session scheduled first (following state $Q^{(r-1)}$) among active sessions in $\overrightarrow{Q^{(r-1)}}$. Since $i_f^{(r)} \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$, it follows that for each session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$. Thus, rates of all sessions are preserved from

state $Q^{(r-1)}$ to Q. Hence, $resid_Q(e^{(r')}) = resid_{Q^{(r-1)}}(e^{(r')}) \geq \frac{p_{r-1}}{4} = p_r$. By Lemma 7.2, it follows that $\Delta_Q(i_f^{(r)}, e^{(r')}) \geq resid_Q(e^{(r')}) \geq p_r$.

Thus, by definition of increase, it follows that $\Delta_Q(i_f^{(r)}) = \frac{p_r}{2}$.

2. Now assume that $i \notin \{i_f^{(r)}, i_l^{(r)}\}$. Since $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$, by condition (A/1) shown above, we have that $r_Q(i) = b_{r-1} + \frac{p_{r-1}}{2}$. By construction of $G(\mathsf{Sched})$ i traverses
   - edge $e_i^{(r)}$ with $c(e_i^{(r)}) = 2\,b_r + p_r$;
   - edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ for each integer r', $1 \leq r' < r$, in case $r > 1$;
   - edge $e^{(r'')} \in E^{(r'')}(\mathsf{Sched})$ for any integer r'', $r < r'' \leq \frac{d}{2}$.

   We next calculate separately the increases for i in Q allowed by these edges.
   (a) First consider edge $e_i^{(r)}$, which is also traversed by $i_f^{(r)}$. Since i is scheduled in front of state $\overrightarrow{Q}$, it follows that $i_f^{(r)} \in \mathcal{S} \mid_\alpha \left(Q^{(r-1)}, Q\right]$. Thus, condition (A/1) shown above implies that $r_Q(i_f^{(r)}) = b_r + \frac{p_r}{2} = \frac{c(e_i^{(r)})}{2}$. By Lemma 7.1,

$$\begin{aligned}\Delta_Q(i, e_i^{(r)}) &= c(e_i^{(r)}) - r_Q(i) - \min\left\{\frac{c(e_i^{(r)})}{2}, r_Q(i_f^{(r)})\right\} \\ &= c(e_i^{(r)}) - r_Q(i) - \frac{c(e_i^{(r)})}{2} \\ &= \frac{c(e_i^{(r)})}{2} - \Big(b_{r-1} + \frac{p_{r-1}}{2}\Big).\end{aligned}$$

   Alternatively, we derive that $\Delta_Q(i, e_i^{(r)}) = b_r + \frac{p_r}{2} - (b_{r-1} + \frac{p_{r-1}}{2}) = \frac{p_r}{2}$.[4]

   (b) Next consider edge $e_i^{(r')}$ for any integer r', $1 \leq r' < r$, in case $r > 1$, which is also traversed by $i_f^{(r')} \notin \{i_f^{(r)}, i_l^{(r)}\}$. Induction hypothesis of induction on r (condition (B/3)) implies that $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$; hence, $i_f^{(r')} \notin \mathcal{A}_Q$. So, $r_{Q^{(r-1)}}(i_f^{(r')}) = r_Q(i_f^{(r')})$. Recall that $r_Q(i) = b_{r-1} + \frac{p_{r-1}}{2}$. On the other hand, by the induction hypothesis of induction on r (condition (B/2)), $r_{Q^{(r-1)}}(i) = b_{r-1} + \frac{p_{r-1}}{2}$. Hence, $r_Q(i) = r_{Q^{(r-1)}}(i)$. It now follows that $resid_Q(e_i^{(r')}) = resid_{Q^{(r-1)}}(e_i^{(r')})$. By the induction hypothesis of induction on r (condition (B/4/a)), $resid_{Q^{(r-1)}}(e_i^{(r')}) \geq \frac{p_{r-1}}{4}$, so that $resid_Q(e_i^{(r')}) \geq \frac{p_{r-1}}{4}$. By Lemma 7.2, $\Delta_Q(i, e_i^{(r')}) \geq \frac{p_{r-1}}{4} = p_r$.

   (c) Finally, consider edge $e^{(r'')}$ for any integer r'', $r < r'' \leq \frac{d}{2}$, which is also traversed by some other session $k \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By condition (A/1) shown above and by recursive definition of b_r, $r_Q(k) \leq b_r + \frac{p_r}{2} = \frac{c(e_i^{(r)})}{2}$. By Lemma 6.1, $c(e^{(r'')}) > c(e_i^{(r)})$. By Lemma 7.1,

[4] Both expressions provided for $\Delta_Q(i, e_i^{(r)})$ will be needed in the rest of the proof.

$$\begin{aligned}\Delta_Q(i, e^{(r'')}) &= c(e^{(r'')}) - r_Q(i) - \min\left\{\frac{c(e^{(r'')})}{2}, r_Q(k)\right\} \\ &\geq c(e^{(r'')}) - r_Q(i) - \min\left\{\frac{c(e^{(r'')})}{2}, \frac{c(e_i^{(r)})}{2}\right\} \\ &\geq c(e^{(r'')}) - r_Q(i) - \frac{c(e^{(r'')})}{2} \\ &= \frac{c(e^{(r'')})}{2} - \Big(b_{r-1} + \frac{p_{r-1}}{2}\Big).\end{aligned}$$

Since $c(e^{(r'')}) > c(e_i^{(r)})$, comparing the first expression for $\Delta_Q(i, e_i^{(r)})$ to $\Delta_Q(i, e^{(r'')})$ implies that $\Delta_Q(i, e^{(r'')}) > \Delta_Q(i, e_i^{(r)})$.
Thus, by definition of increase, it follows that $\Delta_Q(i) = \frac{p_r}{2}$.

3. Finally, assume that $i = i_l^{(r)}$. Since $i_l^{(r)} \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \mathcal{S} \mid_\alpha \big(Q^{(r-1)}, Q\big]$, by condition (A/1) shown above, we have that $r_Q(i_l^{(r)}) = b_{r-1} + \frac{p_{r-1}}{2} = b_r$. By construction of $G(\mathsf{Sched})$, $i_l^{(r)}$ traverses
   - edge $e_l^{(r)}$ with capacity $c(e_l^{(r)}) = 2\,b_r + \frac{p_r}{2}$;
   - edge $e^{(r')} \in E^{(r')}(\mathsf{Sched})$ for each integer r', $1 \leq r' < r$, in case $r > 1$.

   We next calculate separately the increases for $i_l^{(r)}$ in Q allowed by these edges.
   (a) First consider edge $e_l^{(r)}$, which is also traversed by session $i_f^{(r)}$. Since $i_l^{(r)}$ is scheduled in front of $\overrightarrow{Q}$, it follows that $i_f^{(r)} \in \mathcal{S} \mid_\alpha \big(Q^{(r-1)}, Q\big]$. Thus, condition (A/1) shown above implies that $r_Q(i_f^{(r)}) = b_r + \frac{p_r}{2}$. By Lemma 7.1,

$$\begin{aligned}\Delta_Q(i_l^{(r)}, e_l^{(r)}) &= c(e_l^{(r)}) - r_Q(i_l^{(r)}) - \min\left\{\frac{c(e_l^{(r)})}{2}, r_Q(i_f^{(r)})\right\} \\ &= 2\,b_r + \frac{p_r}{2} - b_r - \min\Big\{b_r + \frac{p_r}{4}, b_r + \frac{p_r}{2}\Big\} \\ &= \frac{p_r}{4}.\end{aligned}$$

   (b) Finally, consider any edge $e^{(r')}$ for some integer r', $1 \leq r' < r$, in case $r > 1$, which is also traversed by session $i_f^{(r')} \notin \{i_f^{(r)}, i_l^{(r)}\}$. Induction hypothesis of induction on r (condition (B/3)) implies that $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, so that $i_f^{(r')} \notin \mathcal{A}_Q$. It follows that $r_{Q^{(r-1)}}(i_f^{(r')}) = r_Q(i_f^{(r')})$. Recall that $r_Q(i_l^{(r)}) = b_{r-1} + \frac{p_{r-1}}{2}$. On the other hand, by the induction hypothesis of induction on r (condition (B/2)), we have that $r_{Q^{(r-1)}}(i_l^{(r)}) = b_{r-1} + \frac{p_{r-1}}{2}$. Hence, $r_Q(i_l^{(r)}) = r_{Q^{(r-1)}}(i_l^{(r)})$. It now follows that $\mathit{resid}_Q(e^{(r')}) = \mathit{resid}_{Q^{(r-1)}}(e^{(r')})$. By the induction hypothesis of induction on r (condition (B/4/a)), $\mathit{resid}_{Q^{(r-1)}}(e^{(r')})$

$\geq \frac{p_{r-1}}{4}$, so that $resid_Q(e^{(r')}) \geq \frac{p_{r-1}}{4}$ as well. By Lemma 7.2, it follows that $\Delta_Q(i_l^{(r)}, e^{(r')}) \geq \frac{p_{r-1}}{4} = p_r$.

Thus, by definition of increase, it follows that $\Delta_Q(i_l^{(r)}) = \frac{p_r}{4}$. □

The proof of part (A) is now complete. We continue with the proof of part (B).

Proof of part (B).

Proof of (B/1). Assume, by way of contradiction, that $l_r = \infty$. Then there exists some session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$ which is not scheduled in front of any state following $Q^{(r-1)}$. By condition (A/5) shown above, it follows that for any state Q such that $Q^{(r-1)} \overset{\alpha}{\longmapsto} Q$, $\Delta_Q(i) \geq \frac{p_r}{4} > 0$ (since $p_r > 0$). Thus, by the `update` operation, the rate of session i can be increased without causing any decrease to the rate of any other session with smaller rate. This is in contradiction to max-min fairness. □

Condition (B/1) establishes that state $Q^{(r)}$ is well defined. Recall that in part (A) we considered all states from $Q^{(r-1)}$ through $\overleftarrow{Q^{(r)}}$. We now explore further properties of $Q^{(r)}$.

Proof of (B/2). Recall that session $i_l^{(r)}$ is scheduled in front of $Q^{(r)}$. By condition (A/1) shown above, $r_{\underset{Q^{(r)}}{\leftarrow}}(i_f^{(r)}) = b_r + \frac{p_r}{2}$, while $r_{\underset{Q^{(r)}}{\leftarrow}}(i_l^{(r)}) = b_{r-1} + \frac{p_{r-1}}{2} = b_r$. Moreover, by condition (A/5) shown above, $\Delta_{\underset{Q^{(r)}}{\leftarrow}}(i_l^{(r)}) = \frac{p_r}{4}$. Thus, by the `update` operation, $r_{Q^{(r)}}(i_l^{(r)}) = r_{\underset{Q^{(r)}}{\leftarrow}}(i_l^{(r)}) + \Delta_{\underset{Q^{(r)}}{\leftarrow}}(i_l^{(r)}) = b_r + \frac{p_r}{4}$; moreover, $r_{Q^{(r)}}(i_f^{(r)}) = \min\{r_{Q^{(r)}}(i_l^{(r)}), r_{\underset{Q^{(r)}}{\leftarrow}}(i_f^{(r)})\} = \min\left\{b_r + \frac{p_r}{4}, b_r + \frac{p_r}{2}\right\} = b_r + \frac{p_r}{4}$.

Consider now any session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By definition of $Q^{(r)}$, $i \in \mathcal{S}\,|_\alpha (Q^{(r-1)}, \overleftarrow{Q^{(r)}}]$. Thus, condition (A/1) shown above implies that $r_{\underset{Q^{(r)}}{\leftarrow}}(i) = b_r + \frac{p_r}{2}$. Since, by construction of $G(\mathsf{Sched})$, $i \cap i_l^{(r)} = \emptyset$, the `update` operation implies that $r_{Q^{(r)}}(i) = r_{\underset{Q^{(r)}}{\leftarrow}}(i) = b_r + \frac{p_r}{2}$. □

Proof of (B/3). By condition (A/4) shown above, it follows that fair shares of all active edges are preserved from $\overleftarrow{Q^{(r)}}$ to $Q^{(r)}$. Since Alg is bottleneck and the termination condition for bottleneck algorithms is a predicate on session rates and fair shares, it follows that the only sessions that are candidates to be terminated in state $Q^{(r)}$ are those whose rates are changed in $Q^{(r)}$; by conditions (A/1) and (B/1) shown above, these are sessions $i_f^{(r)}$ and $i_l^{(r)}$.

By condition (A/3/a) shown above, edge $e_l^{(r)}$ is a bottleneck edge for $\overleftarrow{Q^{(r)}}$. Condition (A/4) shown above implies that both $i_f^{(r)}, i_l^{(r)} \in \mathcal{A}_{Q^{(r)}}$, so that $e_l^{(r)} \in AE_{Q^{(r)}}$. Thus, Proposition 4.8 (condition (2)) implies that $e_l^{(r)}$ is a bottleneck edge for $Q^{(r)}$. Recall that $FS_{Q^{(r)}}(e_l^{(r)}) = FS_{\underset{Q^{(r)}}{\leftarrow}}(e_l^{(r)})$. Thus, by condition (A/2) shown above, it follows that $FS_{Q^{(r)}}(e_l^{(r)}) = b_r + \frac{p_r}{4}$. By condition (B/2) shown above, this implies that $r_{Q^{(r)}}(i_f^{(r)}) = r_{Q^{(r)}}(i_l^{(r)}) = FS_{Q^{(r)}}(e_l^{(r)})$. Since Alg is bottleneck, it follows that $i_f^{(r)}, i_l^{(r)} \in \mathsf{Term}\left(Q^{(r)}\right)$. □

Proof of (B/4). Take any session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, and fix any integer r', $1 \leq r' \leq r$. The proof is by case analysis on r'.

1. First consider the case where $r' = r$. By construction of $G(\mathsf{Sched})$, edge $e_i^{(r)}$ with capacity $c(e_i^{(r)}) = 2\,b_r + p_r$ is traversed by sessions $i_f^{(r)}$ and i. So,

$$\begin{aligned}
&\mathit{resid}_{Q^{(r)}}(e_i^{(r)}) \\
&\quad = c(e_i^{(r)}) - r_{Q^{(r)}}(i_f^{(r)}) - r_{Q^{(r)}}(i) \\
&\quad = 2\,b_r + p_r - \left(b_r + \frac{p_r}{4}\right) - \left(b_r + \frac{p_r}{2}\right) \quad \text{(by condition (B/2) shown above)} \\
&\quad = \frac{p_r}{4},
\end{aligned}$$

which establishes condition (B/4/a) in this case.
We continue with condition (B/4/b). By condition (B/3) shown above, $i_f^{(r)} \notin \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$, while $i \in \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$, so that $|\mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}} \mid e| = 1$. Thus,

$$\begin{aligned}
&FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r)}) \\
&\quad = \frac{c(e_i^{(r)}) - \mathit{allot}_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r)})}{\left|\mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \mid e_i^{(r)}\right|} \\
&\quad = \frac{2b_r + p_r - (b_r + \frac{p_r}{4})}{1} \text{ (by conditions (B/1) and (B/3) shown above)} \\
&\quad = b_r + \frac{3}{4}\,p_r,
\end{aligned}$$

which establishes condition (B/4/b) in this case.
2. Now consider the case where $r' < r$. Clearly, $r > 1$ in this case, so that the induction hypothesis of induction on r is nonempty. Fix any edge $e_i^{(r')} \in E^{(r')}(\mathsf{Sched})$. Condition (B/3) shown above implies that $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, so that $r_{Q^{(r)}}(i_f^{(r')}) = r_{Q^{(r-1)}}(i_f^{(r')})$. By condition (B/2) shown above, $r_{Q^{(r)}}(i) = b_r + \frac{p_r}{2}$, while by the induction hypothesis of induction on r (condition (B/2)), $r_{Q^{(r-1)}}(i) = b_{r-1} + \frac{p_{r-1}}{2} = b_r$. Thus,

$$\begin{aligned}
&\mathit{resid}_{Q^{(r-1)}}(e_i^{(r')}) - \mathit{resid}_{Q^{(r)}}(e_i^{(r')}) \\
&\quad = (c(e_i^{(r')}) - r_{Q^{(r-1)}}(i_f^{(r')}) - r_{Q^{(r-1)}}(i)) \\
&\qquad - (c(e_i^{(r')}) - r_{Q^{(r)}}(i_f^{(r')}) - r_{Q^{(r)}}(i)) \\
&\quad = (r_{Q^{(r)}}(i) - r_{Q^{(r-1)}}(i)) - (r_{Q^{(r)}}(i_f^{(r')}) - r_{Q^{(r-1)}}(i_f^{(r')})) \\
&\quad = \left(b_r + \frac{p_r}{2}\right) - b_r - 0 \\
&\quad = \frac{p_r}{2}.
\end{aligned}$$

By the induction hypothesis of induction on r (condition (B/4/a)), it follows that $\mathit{resid}_{Q^{(r-1)}}(e_i^{(r')}) \geq \frac{p_{r-1}}{4} = p_r$. Thus, $\mathit{resid}_{Q^{(r)}}(e_i^{(r')}) \geq p_r - \frac{p_r}{2} > \frac{p_r}{4}$, which establishes condition (B/4/a) in this case.
We continue with condition (B/4/b). Recall that $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, so that $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$, and $r_{\underset{Q^{(r-1)}}{\rightarrow}}(i_f^{(r')}) = r_{\underset{Q^{(r)}}{\rightarrow}}(i_f^{(r')})$; on the other hand, $i \in \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$,

so that $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. It follows that $FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e_i^{(r')}) = FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r')})$. By the induction hypothesis of induction on r (condition (B/4/b)), it follows that $FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e_i^{(r')}) \geq b_{r-1} + \frac{3p_{r-1}}{4} = b_r + p_r$. Thus, $FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r')}) \geq b_r + p_r > b_r + \frac{3p_r}{4}$, which establishes condition (B/4/b) in this case. □

The proof of Proposition 7.3 is now complete. □

We are now ready to prove the following.

THEOREM 7.4 (lower bound for oblivious algorithms). *Assume that* Alg *is optimistic, oblivious, and bottleneck, and that it computes the max-min fair rate vector. Then* $\mathcal{U}_{\mathsf{Alg}} \geq \frac{dn}{4} + \frac{n}{2}$.

Proof. We will show that $\mathcal{U}_{\mathsf{Alg}}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle) = \frac{dn}{4} + \frac{n}{2}$, where $\mathsf{Alg} = \langle \mathsf{Sched}, \mathsf{Term}\rangle$ and $G(\mathsf{Sched})$ is the network constructed in section 6. To do so, we calculate $\mathcal{U}_{\mathsf{Alg}}(\langle G(\mathsf{Sched}), \mathcal{S}_j\rangle)$ for any particular cluster $\mathcal{S}_j$, $j \geq 1$, and we add up over all $\frac{n}{d}$ clusters. By Proposition 7.3 (condition (B/1)), the execution of Alg on $G(\mathsf{Sched} \mid \mathcal{S}_j)$ is divided into $\frac{d}{2}$ execution epochs $\alpha^{(1)}, \ldots, \alpha^{(d/2)}$ such that all active sessions are scheduled at least once in each epoch; condition (B/3) implies that only two such sessions are terminated in each epoch. Thus, at least $d - 2(r-1)$ `update` operations are executed in $\alpha^{(r)}$, $1 \leq r \leq \frac{d}{2}$. Summing up over all clusters and execution epochs yields that $\mathcal{U}_{\mathsf{Alg}}(G, \mathcal{S}) \geq \sum_{j=1}^{n/d} \sum_{r=1}^{d/2} (d - 2(r-1)) = \frac{nd}{4} + \frac{n}{2}$, as needed. □

7.2. Partially oblivious algorithms. In this section, we present a lower bound of $\Omega(dn)$ on the convergence complexity of any optimistic, partially oblivious, and bottleneck algorithm that computes the max-min rate vector.

For the sake of clarity of presentation, we first consider the special case where $d = n$. (We remark that this is the case where the lower bound to be shown is maximum over all possible values of d, $1 \leq d \leq n$.) The backbone of our proof for this case will be a simulation lemma that establishes a correspondence between the executions of any partially oblivious algorithm and a suitable oblivious algorithm on the network constructed from the latter as in section 6 (assuming $d = n$). We will then use the simulation lemma to prove a lower bound of $\Omega(n^2)$ for this case. We finally consider the general case of arbitrary d; we state the $\Omega(dn)$ lower bound for it and discuss its proof, which relies on a generalization of the simulation lemma to general d. We start with the simulation lemma for the case $d = n$.

PROPOSITION 7.5 (simulation lemma). *Assume that* Alg *is optimistic, partially oblivious, and bottleneck, and that it computes the max-min fair rate vector. Then there exists some optimistic, oblivious, and bottleneck algorithm* $\mathsf{OAlg} = \langle \mathsf{OSched}, \mathsf{OTerm}\rangle$ *such that there exist execution prefixes* $\alpha = \alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{d/2}$ *of* OAlg *and* $\beta = \beta_1 \cdot \beta_2 \cdot \ldots \cdot \beta_{d/2}$ *of* Alg*, on network* $G(\mathsf{OSched})$*, such that for each integer* r*,* $1 \leq r \leq \frac{d}{2}$*, the following conditions hold:*

1. α_r *and* β_r *are identical;*
2. *all sessions active in first* (β_r) *are scheduled at least once in* β_r *and only two of them terminate in* β_r*.*

We first provide an informal, high-level outline of our proof. Given any partially oblivious algorithm, we derive some oblivious algorithm such that the executions of the two algorithms on the network constructed from the oblivious one in section 6 are identical. The execution of the oblivious algorithm is described by Proposition 7.3; since the two executions are shown to be identical, this will eventually imply the claimed lower bound for the partially oblivious algorithm. In order to derive the oblivious algorithm with the required properties, we start with the set of all n-epoch, oblivious

algorithms; we inductively restrict this set until it contains only algorithms that are compatible with the partially oblivious algorithm (with respect to their schedulers) and perform sufficiently many `update` operations. All intermediate sets of oblivious algorithms contain only algorithms that induce executions identical to the partially oblivious algorithm up to each intermediate point. In more detail, the partial execution of each oblivious algorithm in any intermediate set on the network constructed from the algorithm in section 6 is identical to the partial execution of the partially oblivious algorithm on the same network. Each individual intermediate step is extended progressively by considering the session scheduled next by the partially oblivious algorithm and restricting the intermediate set of oblivious algorithms to those that schedule the same session next. (The construction follows an outer induction on execution epochs and an inner induction on states within each epoch.)

Proof. For each index r, $1 \leq r \leq \frac{d}{2}$, we will construct a set $\mathbf{A}^{(r)}$ of optimistic, oblivious, and bottleneck algorithms that compute the max-min fair rate vector such that for each algorithm $\mathsf{OAlg} \in \mathbf{A}^{(r)}$, there exist execution prefixes $\alpha(\mathsf{OAlg}) = \alpha_1(\mathsf{OAlg}) \ldots \alpha_r(\mathsf{OAlg})$ of OAlg and $\beta(\mathsf{OAlg}) = \beta_1(\mathsf{OAlg}) \ldots \beta_r(\mathsf{OAlg})$ of Alg on network $G(\mathsf{OSched})$ so that

(1) for each algorithm $\mathsf{OAlg} \in \mathbf{A}^{(r)}$ for each integer r', $1 \leq r' \leq r$, $\alpha_{r'}(\mathsf{OAlg})$ and $\beta_{r'}(\mathsf{OAlg})$ are identical;
(2) $\alpha(\mathsf{OAlg})$ is identical over all algorithms $\mathsf{OAlg} \in \mathbf{A}^{(r)}$;
(3) $\beta(\mathsf{OAlg})$ is identical over all algorithms $\mathsf{OAlg} \in \mathbf{A}^{(r)}$;
(4) for each algorithm $\mathsf{OAlg} \in \mathbf{A}^{(r)}$ for each integer r', $1 \leq r' \leq r$, all sessions active in state $\mathit{first}(\beta_{r'}(\mathsf{OAlg}))$ are scheduled at least once in $\beta_{r'}(\mathsf{OAlg})$, but only two of them terminate in $\beta_{r'}(\mathsf{OAlg})$.

Any algorithm OAlg in the final set $\mathbf{A}^{(d/2)}$ will satisfy the claim due to conditions (1) and (4) guaranteed by the construction.

The construction will be by induction on r, $1 \leq r \leq \frac{d}{2}$. The construction employs the set of all optimistic, oblivious, bottleneck, n-epoch algorithms, which we denote as $\mathbf{A}^{(0)}$. By Theorem 5.2, any algorithm from $\mathbf{A}^{(0)}$ computes the max-min fair rate vector. Our construction will progressively restrict the set $\mathbf{A}^{(0)}$ so that $\mathbf{A}^{(d/2)} \subseteq \cdots \subseteq \mathbf{A}^{(r)} \subseteq \cdots \subseteq \mathbf{A}^{(0)}$. Hence, for each integer r, $1 \leq r \leq \frac{d}{2}$, any algorithm from $\mathbf{A}^{(r)}$ computes the max-min fair rate vector. This property will be needed later, whenever we use Proposition 7.3, which assumes it. For each algorithm $\mathsf{OAlg} \in \mathbf{A}^{(0)}$, consider the initial state $Q_0(\mathsf{OAlg})$ of network $G(\mathsf{OSched})$ (with all rates zero and all sessions active). Set $\alpha_0(\mathsf{OAlg}) := Q_0(\mathsf{OAlg})$ and $\beta_0(\mathsf{OAlg}) := Q_0(\mathsf{OAlg})$. For the sake of shortening the construction, we merge the construction for the basis case (where $r = 1$) and the construction for the induction step. Thus, the case $r = 1$ will be treated separately (where needed) along the construction for the induction step.

Fix any integer r, $1 \leq r \leq \frac{d}{2}$, and assume inductively that we have constructed a set $\mathbf{A}^{(r-1)}$ with the required properties. Notice that if $r = 1$, then $\mathbf{A}^{(r-1)} = \mathbf{A}^{(0)}$ and the induction hypothesis is empty.

For the induction step, we construct a set $\mathbf{A}^{(r)} \subseteq \mathbf{A}^{(r-1)}$ with the required properties. This construction is progressive and uses induction on states, starting with $\mathit{last}(\beta(\mathsf{OAlg}))$ for any algorithm $\mathsf{OAlg} \in \mathbf{A}^{(r-1)}$. We will prove that conditions (1), (2), and (3) are preserved along the inductive construction on states, while (4) holds when the construction of $\mathbf{A}^{(r)}$ is complete.

For the basis case (of induction on states), take $\mathbf{A}^{(r)} := \mathbf{A}^{(r-1)}$; for any algorithm $\mathsf{OAlg} = \langle \mathsf{OSched}, \mathsf{OTerm} \rangle \in \mathbf{A}^{(r-1)}$, set $\alpha_r(\mathsf{OAlg}) = \mathit{last}(\alpha_{r-1}(\mathsf{OAlg}))$ and $\beta_r(\mathsf{OAlg}) = \mathit{last}(\beta_{r-1}(\mathsf{OAlg}))$. In case $r = 1$, by definition of initial state ($Q_0(\mathsf{OAlg})$),

$\alpha_1(\mathsf{OAlg})$ (resp., $\beta_1(\mathsf{OAlg})$) is identical for all algorithms $\mathsf{OAlg} \in \mathbf{A}^{(1)}$, implying (2) and (3). By construction, for each algorithm $\mathsf{OAlg} \in \mathbf{A}^{(1)}$, it trivially holds that $\alpha_1(\mathsf{OAlg})$ and $\beta_1(\mathsf{OAlg})$ are identical, which implies (1). In case $r > 1$, by the induction hypothesis of induction on r, conditions (1), (2), and (3) hold.

We now continue with the induction step (induction on states) of the inductive construction of $\mathbf{A}^{(r)}$. Take any algorithm $\mathsf{OAlg} \in \mathbf{A}^{(r)}$ and consider $\alpha(\mathsf{OAlg})$. (By the induction hypothesis of induction on states (condition (2)), the choice of OAlg does not matter.) In case $r = 1$, all sessions are active in state $last(\beta_{r-1}(\mathsf{OAlg})) = Q_0(\mathsf{OAlg})$. In case $r > 1$, since $r - 1 < \frac{d}{2}$, Proposition 7.3 (condition (B/3)) implies that the set of active sessions in state $last(\alpha_{r-1}(\mathsf{OAlg}))$ is nonempty; the induction hypothesis of induction on states (condition (1)) implies now that the set of active sessions in state $last(\beta_{r-1}(\mathsf{OAlg}))$ is nonempty as well. If all such sessions have been scheduled at least once in $\alpha_r(\mathsf{OAlg})$, then the inductive construction of $\mathbf{A}^{(r)}$ is complete and conditions (1), (2), and (3) hold by the induction hypothesis of induction on states. (Condition (4) will be shown later.) So assume that there exists at least one such active session $i(\mathsf{OAlg})$ that has not been scheduled in $\beta_r(\mathsf{OAlg})$. By the induction hypothesis of induction on states (condition (1)), $\alpha_r(\mathsf{OAlg})$ and $\beta_r(\mathsf{OAlg})$ are identical. It follows that $i(\mathsf{OAlg})$ has not been scheduled in $\alpha_r(\mathsf{OAlg})$ either. Proposition 7.3 (condition (A/5)) implies that the increase for session $i(\mathsf{OAlg})$ in state $last(\alpha_r(\mathsf{OAlg}))$ is nonzero. By the induction hypothesis of induction on states (condition (1)), it follows that the increase for session $i(\mathsf{OAlg})$ in state $last(\beta_r(\mathsf{OAlg}))$ is nonzero as well. By the `update` operation, it follows that an increase to the rate of $i(\mathsf{OAlg})$ in $last(\beta_r(\mathsf{OAlg}))$ is possible without decreasing the rate of any other session. So, max-min fairness has not been reached yet and some session rate must change. However, a session rate changes only when an active session is scheduled. Since Alg computes the max-min fair rate vector, it follows that at least one active session in $last(\beta_r(\mathsf{OAlg}))$ is scheduled after $last(\beta_r(\mathsf{OAlg}))$. An inductive application of this argument implies that all sessions active in $first(\beta_r(\mathsf{OAlg}))$ are scheduled at least once in $\beta_r(\mathsf{OAlg})$. It follows that the inductive construction of $\mathbf{A}^{(r)}$ eventually terminates.

Denote by $i'(\mathsf{OAlg})$ the session scheduled by Alg immediately after $last(\beta_r(\mathsf{OAlg}))$ on network $G(\mathsf{OSched})$. By the induction hypothesis of induction on states (condition (3)), it follows that $last(\beta_r(\mathsf{OAlg}))$ is identical for all $\mathsf{OAlg} \in \mathbf{A}^{(r)}$. Since Alg is partially oblivious, this implies that $i'(\mathsf{OAlg})$ is identical for all $\mathsf{OAlg} \in \mathbf{A}^{(r)}$ as well; so, denote it i'. Fix any algorithm $\mathsf{OAlg} \in \mathbf{A}^{(r)}$ and restrict $\mathbf{A}^{(r)}$ to the set of all optimistic, oblivious, bottleneck, n-epoch algorithms whose schedulers have prefix $\sigma(\alpha_r(\mathsf{OAlg}))\, i'$.

We now argue that $\mathbf{A}^{(r)}$ is nonempty and unique. Any sequence of sessions $\sigma(\beta_r(\mathsf{OAlg}))\, i'$ can be extended to an n-epoch scheduler (by appropriate padding). This implies that $\mathbf{A}^{(r)}$ is nonempty. By the induction hypothesis of induction on states (condition (3)), $\beta_r(\mathsf{OAlg})$ is identical over all algorithms $\mathsf{OAlg} \in \mathbf{A}^{(r)}$, so that $\sigma(\beta_r(\mathsf{OAlg}))\, i'$ is also identical over all $\mathsf{OAlg} \in \mathbf{A}^{(r)}$. It follows that the constructed $\mathbf{A}^{(r)}$ is unique.

Take any algorithm $\mathsf{OAlg} = \langle \mathsf{OSched}, \mathsf{OTerm} \rangle \in \mathbf{A}^{(r)}$.

- Define $Q(\mathsf{OAlg})$ to be the state that results when Alg, starting from state $last(\beta_r(\mathsf{OAlg}))$ of the network $G(\mathsf{OSched})$, schedules session i' on $G(\mathsf{OSched})$ and set $\beta_r(\mathsf{OAlg}) := \beta_r(\mathsf{OAlg}),\, i',\, Q(\mathsf{OAlg})$.
- Similarly, define $Q'(\mathsf{OAlg})$ to be the state that results when OAlg, starting from $last(\alpha_r(\mathsf{OAlg}))$ of the network $G(\mathsf{OSched})$, schedules session i' on $G(\mathsf{OSched})$ and set $\alpha_r(\mathsf{OAlg}) := \alpha_r(\mathsf{OAlg}),\, i',\, Q'(\mathsf{OAlg})$.

We now prove the required properties for the sets

$$\alpha(\mathsf{OAlg}) = \left\{ \alpha_1(\mathsf{OAlg}) \cdot \ldots \cdot \alpha_r(\mathsf{OAlg}) \;\mid\; \mathsf{OAlg} \in \mathbf{A}^{(r)} \right\}$$

and

$$\beta(\mathsf{OAlg}) = \left\{ \beta_1(\mathsf{OAlg}) \cdot \ldots \cdot \beta_r(\mathsf{OAlg}) \;\mid\; \mathsf{OAlg} \in \mathbf{A}^{(r)} \right\}$$

of execution prefixes.

We start by proving (1). Take any algorithm $\mathsf{OAlg} = \langle \mathsf{OSched}, \mathsf{OTerm} \rangle \in \mathbf{A}^{(r)}$. By the induction hypothesis of induction on states (condition (1)), Alg and OAlg start from identical states of the network $G(\mathsf{OSched})$; hence, scheduling the same session (i') on the network $G(\mathsf{OSched})$ results in identical states, which inductively extends the claim.

We now treat (2) and (3). Since $i'(\mathsf{OAlg})$ is identical over all $\mathsf{OAlg} \in \mathbf{A}^{(r)}$, the induction hypothesis of induction on states (condition (2)) implies that the set of sessions scheduled in $\alpha_r(\mathsf{OAlg})$ is identical over all $\mathsf{OAlg} \in \mathbf{A}^{(r)}$. So, Proposition 7.3 (conditions (A/1) and (A/4)) implies that $Q'(\mathsf{OAlg})$ is identical over all $\mathsf{OAlg} \in \mathbf{A}^{(r)}$. Induction hypothesis of induction on states (condition (2)) and the fact that $i'(\mathsf{OAlg})$ is identical over all $\mathsf{OAlg} \in \mathbf{A}^{(1)}$ imply that $\alpha_r(\mathsf{OAlg})$ is identical over all $\mathsf{OAlg} \in \mathbf{A}^{(r)}$, which proves (2). Now (3) follows from (1) and (2).

We finally prove (4) for the set $\mathbf{A}^{(r)}$ constructed when the induction on states is complete. Fix any algorithm $\mathsf{OAlg} \in \mathbf{A}^{(r)}$. Recall that all sessions active in $\mathit{first}(\beta_r(\mathsf{OAlg}))$ are scheduled at least once in $\beta_r(\mathsf{OAlg})$; moreover, the termination condition used in the construction of $\mathbf{A}^{(r)}$ implies that $\beta_r(\mathsf{OAlg})$ is the shortest such fragment. By condition (1) shown above, this implies that $\alpha_r(\mathsf{OAlg})$ is the shortest execution fragment such that all sessions active in $\mathit{first}(\alpha_r(\mathsf{OAlg}))$ are scheduled at least once in $\alpha_r(\mathsf{OAlg})$. So, by Proposition 7.3 (condition (B/3)), only two sessions terminate in $\alpha_r(\mathsf{OAlg})$. Hence, the induction hypothesis of induction on states (condition (1)) implies that only two sessions terminate in $\beta_r(\mathsf{OAlg})$, as needed.

The proof of the simulation lemma is now complete. □

We are now ready to prove the lower bound for the case where $d = n$.

THEOREM 7.6 (lower bound for partially oblivious algorithms). *Assume that* Alg *is optimistic, partially oblivious, and bottleneck, and that it computes the max-min rate vector. Then* $\mathcal{U}_{\mathsf{Alg}} \geq \frac{n^2}{4} + \frac{n}{2}$.

Proof. Proposition 7.5 implies that there exists some optimistic, oblivious, and bottleneck algorithm $\mathsf{OAlg} = \langle \mathsf{OSched}, \mathsf{OTerm} \rangle$ such that there exists an execution prefix $\beta = \beta_1 \cdot \beta_2 \cdot \ldots \cdot \beta_{d/2}$ of Alg on network $G(\mathsf{OSched})$ such that for each integer r, $1 \leq r \leq \frac{d}{2}$, all sessions active in $\mathit{first}(\beta_r)$ are scheduled at least once in β_r and only two of them terminate in β_r. Hence, at least $n - 2(r-1)$ `update` operations are executed in β_r, $1 \leq r \leq \frac{n}{2}$. Summing up over all r epochs, $1 \leq r \leq \frac{n}{2}$, yields that $\mathcal{U}_{\mathsf{Alg}}(G(\mathsf{OSched}), \mathcal{S}) \geq \sum_{r=1}^{n/2} (n - 2(r-1)) = \frac{n^2}{4} + \frac{n}{2}$. □

As a direct generalization of the simulation lemma for the case $d = n$, we obtain the following.

PROPOSITION 7.7 (simulation lemma). *Assume that* Alg *is optimistic, partially oblivious, and bottleneck, and that it computes the max-min fair rate vector. Partition* $\mathcal{S}$ *into disjoint clusters* $\mathcal{S}_1, \ldots, \mathcal{S}_{n/d}$ *with* d *sessions each. Then there exists some optimistic, oblivious, and bottleneck algorithm* $\mathsf{OAlg} = \langle \mathsf{OSched}, \mathsf{OTerm} \rangle$ *such that there exist execution prefixes* α *of* OAlg *and* β *of* Alg*, on network* $G(\mathsf{OSched})$*, such that for each cluster* $\mathcal{S}_j$*,* $1 \leq j \leq \frac{n}{d}$*,* α *and* β *can be written as* $\alpha = \alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{\frac{d}{2}}$

and $\beta = \beta_1 \cdot \beta_2 \cdot \ldots \cdot \beta_{\frac{d}{2}}$ *so that for each integer* r, $1 \leq r \leq \frac{d}{2}$, *the following conditions hold:*

(1) α_r *and* β_r *are identical;*
(2) *all sessions from* $\mathcal{S}_j$ *active in first* (β_r) *are scheduled at least once in* β_r *and only two of them terminate in* β_r.

The execution prefixes α and β in Proposition 7.7 are constructed in a similar inductive manner to the corresponding prefixes in Proposition 7.5. The additional complications stem from the fact that the execution fragments α_r and β_r, $1 \leq r \leq \frac{d}{2}$, may now be different for each cluster; thus, the inductive construction in the proof of Proposition 7.5 needs to be adjusted in order to accommodate chopping off α and β into the suitable execution fragments α_r and β_r, $1 \leq r \leq \frac{d}{2}$, for each particular cluster. We finally use Proposition 7.7 to show our final lower bound; its proof is similar to the one of Theorem 7.6 that used Proposition 7.5.

THEOREM 7.8 (lower bound for partially oblivious algorithms). *Assume that* Alg *is optimistic, partially oblivious, and bottleneck, and that it computes the max-min rate vector. Then* $\mathcal{U}_{\mathsf{Alg}} \geq \frac{dn}{4} + \frac{n}{2}$.

Consider the partially oblivious algorithm GlobalMinSched introduced by Afek, Mansour, and Ostfeld [1, section 4], whose scheduler chooses, for each state Q, the active session with the minimum rate. We note that [1, Theorem 4.3] implies an upper bound $\frac{|\mathcal{S}_j|\,(|\mathcal{S}_j|+1)}{2}$ on the number of `update` operations executed by GlobalMin on network G with session set $\mathcal{S}_j$ for any particular cluster $\mathcal{S}_j$. So, $\mathcal{U}_{\mathsf{GlobalMin}} \leq \sum_{j\geq 1} \frac{|\mathcal{S}_j|\,(|\mathcal{S}_j|+1)}{2} \leq \frac{\max_{j\geq 1}|\mathcal{S}_j|+1}{2} \sum_{j\geq 1} |\mathcal{S}_j| = \frac{dn}{2} + \frac{n}{2}$. This implies that the lower bound established in Theorem 7.8 is tight (within a factor of 2).

8. Discussion and directions for further research. We have presented a comprehensive collection of lower and upper bounds on the convergence complexity of optimistic, bottleneck, rate-based flow control algorithms, under varying degrees of the knowledge used by the scheduling component of the algorithms. In particular, we have defined and studied oblivious, partially oblivious, and nonoblivious algorithms. We have shown that, perhaps surprisingly, the classes of oblivious algorithms and partially oblivious algorithms collapse with respect to convergence complexity; we have also shown a convergence complexity separation between (partially) oblivious algorithms and nonoblivious algorithms. A more complete presentation of results for the model studied in this paper (and extensions of it) can be found in [8].

For the sake of completeness and comparison, we summarize in Table 1 all known lower and upper bounds on the convergence complexity of optimistic, bottleneck algorithms for rate-based flow control, established in this work and in the preceding work by Afek, Mansour, and Ostfeld [1]. We remark that RoundRobin represents an exponential improvement over the previous algorithm Arbitrary [1, section 6] for the class of oblivious algorithms we introduced. (The algorithm Arbitrary schedules sessions in any arbitrary way.)

TABLE 1
Summary of known lower and upper bounds on convergence complexity for optimistic, bottleneck rate-based flow control algorithms.

Scheduler types	Lower bounds	Upper bounds
Oblivious	$\frac{dn}{4} + \frac{n}{2}$	$\frac{dn}{2} + \frac{n}{2}$ (RoundRobin) $\Theta(2^n)$ (Arbitrary [1])
Partially oblivious	$\frac{dn}{4} + \frac{n}{2}$	$\frac{dn}{2} + \frac{n}{2}$ (LocalMin or GlobalMin [1])
Nonoblivious	n	n (Linear)

Our work leaves open several important questions. The most obvious open question would be to close the gap between the lower bound of $\frac{dn}{4} + \frac{n}{2}$ and the upper bound of $\frac{dn}{2} + \frac{n}{2}$ we have shown on the convergence complexity of oblivious algorithms.

The model considered in this work is simple and elegant, yet structured enough to capture several significant ingredients of distributed rate-based flow control; there remain, however, a number of significant practical issues untouched by our model and analysis. In the first place, we feel that the max-min fairness criterion may be undue in some realistic situations, where sessions have different demands. (Some results in this direction have been obtained in [10].) Second, the limitation to static sets of sessions is somehow restrictive; it would be significant to extend our model and techniques to handle set-up and take-down of sessions. Third, practical considerations may demand that rate-based flow control algorithms avoid too small or too large adjustments to session rates. Encompassing such practical considerations, and analyzing their impact on convergence complexity, into the framework of rate-based flow control algorithms is an interesting research problem.

Kleinberg, Rabani, and Tardos [20] formulated some natural approximations to max-min fairness and advocated them as suitable fairness conditions for certain routing and load balancing applications. It would be interesting to study the convergence complexity of such approximations within the framework of rate-based flow control algorithms.

Acknowledgments. Our thanks go to the anonymous *PODC* 1997 and *SIAM Journal on Computing* reviewers for their helpful comments.

REFERENCES

[1] Y. Afek, Y. Mansour, and Z. Ostfeld, *Convergence complexity of optimistic rate based flow control algorithms*, J. Algorithms, 30 (1999), pp. 106–133.

[2] D. P. Bertsekas and R. G. Gallager, *Data Networks*, 2nd ed., Prentice-Hall, London, 1992.

[3] K. Bharath-Kumar and J. M. Jaffe, *A new approach to performance-oriented flow control*, IEEE Trans. Comm., 29 (1981), pp. 427–435.

[4] F. Bonomi and K. Fendick, *The rate-based flow control for available bit rate ATM service*, IEEE Network, 9 (1995), pp. 24–39.

[5] A. Charny, *An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback*, Tech. Report MIT/LCS/TR-601, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1994.

[6] A. Charny, D. D. Clark, and R. Jain, *Congestion control with explicit rate indication*, in Proceedings of the IEEE 30th International Conference on Communications, 1995, pp. 1954–1963.

[7] A. Charny, K. K. Ramakrishnan, and A. Lauck, *Time scale analysis and scalability issues for explicit rate allocation in ATM networks*, IEEE/ACM Trans. Networking, 4 (1996), pp. 569–581.

[8] P. Fatourou, *Algorithmic Foundations of Fair Rate-Based Flow Control*, Ph.D. thesis, Department of Computer Engineering and Informatics, University of Patras, Greece, 1999.

[9] P. Fatourou, M. Mavronicolas, and P. Spirakis, *The global efficiency of distributed, rate-based flow control algorithms*, in Proceedings of the 5th Colloquium on Structural Information and Communication Complexity, Carleton Scientific, Waterloo, ON, Canada, 1998, pp. 244–258.

[10] P. Fatourou, M. Mavronicolas, and P. Spirakis, *Max-min fair flow control sensitive to priorities*, in Proceedings of the 2nd International Conference on Principles of Distributed Systems, 1998, pp. 45–59.

[11] E. Gafni and D. Bertsekas, *Dynamic control of session input rates in communication networks*, IEEE Trans. Automat. Control, 29 (1984), pp. 1009–1016.

[12] M. Gerla and L. Kleinrock, *Flow control: A comparative survey*, IEEE Trans. Comm., 28 (1980), pp. 553–574.

[13] E. Hahne, *Round-robin scheduling for maxmin fairness in data networks*, IEEE J. Selected Areas in Commun., 9 (1991), pp. 1024–1039.

[14] E. Hahne and R. G. Gallager, *Round-robin scheduling for fair flow control in data communication networks*, in Proceedings of the IEEE 21st International Conference on Communications, 1986, pp. 103–107.

[15] H. Hayden, *Voice Flow Control in Integrated Packet Networks*, Tech. Report MIT/LIDS/TH/TR-601, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, 1981.

[16] J. M. Jaffe, *Bottleneck flow control*, IEEE Trans. Comm., 29 (1981), pp. 954–962.

[17] J. M. Jaffe, *Flow control power is nondecentralizable*, IEEE Trans. Comm., 29 (1981), pp. 1301–1306.

[18] R. Jain, S. Kalyanaraman, and R. Viswanathan, *The OSU Scheme for Congestion Avoidance Using Explicit Rate Indication*, Tech. Report atm-forum/95-0883, Ohio State University, 1994.

[19] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, *An efficient rate allocation algorithm for ATM networks providing max-min fairness*, in Proceedings of the 6th IFIP International Conference on High Performance Networking, Chapman and Hall, London, 1995, pp. 143–154.

[20] J. Kleinberg, Y. Rabani, and É. Tardos, *Fairness in routing and load balancing*, in Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999, pp. 568–578.

[21] H. Luss and D. R. Smith, *Resource allocation among competing entities: A lexicographic minimax approach*, Oper. Res. Lett., 5 (1986), pp. 227–231.

[22] J. Mosely, *Asynchronous Distributed Flow Control Algorithms*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1984.