Efficiency of Oblivious Versus Non-Oblivious Schedulers for Optimistic, Rate-Based Flow Control

Panagiota Fatourou
University of Patras & Computer Technology Institute, Patras, Greece

Marios Mavronicolas
University of Cyprus, Nicosia, Cyprus

Paul Spirakis
University of Patras & Computer Technology Institute, Patras, Greece

(February 28, 2000)

A preliminary version of this work appears in the *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing* (Santa Barbara, California, August 1997), pp. 139–148, ACM, New York, 1997.

This work has been supported in part by the European Union's ESPRIT Long Term Research Projects ALCOM-IT (contract # 20244) and GEPPCOM (contract # 9072), by the Greek Ministry of Education, by funds for the promotion of research at University of Cyprus, and by AT&T/NCR research funds.

Part of the work of P. Fatourou was done while visiting Department of Computer Science, University of Cyprus, Nicosia, Cyprus. Part of the work of M. Mavronicolas was done while visiting Computer Technology Institute, Patras, Greece, and while at AT&T Labs – Research, Florham Park, NJ, as a visitor to the DIMACS Special Year on Networks, organized by the DIMACS Center for Discrete Mathematics and Theoretical Computer Science, Piscataway, NJ.

Authors' addresses: P. Fatourou, Department of Computer Engineering and Informatics, University of Patras, Rion, 265 00 Patras, Greece, & Computer Technology Institute, P. O. Box 1122, 261 10 Patras, Greece; M. Mavronicolas, Department of Computer Science, University of Cyprus, P. O. Box 537, Nicosia CY-1678, Cyprus; P. Spirakis, Department of Computer Engineering and Informatics, University of Patras, Rion, 265 00 Patras, Greece, & Computer Technology Institute, P. O. Box 1122, 261 10 Patras, Greece.

Abstract. Two important performance parameters of *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 till *max-min fairness* is reached. *Optimistic* such algorithms may go through transient states in which any *session* may receive more than its final, max-min fair rate; *bottleneck* such algorithms "terminate" a session only if its rate 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. We assume a set of n sessions laid out on a network of *session dependency* d; thus, an update operation on one session may influence, either directly or indirectly, at most d sessions.

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, as our main result, a fundamental lower bound of $dn/4+n/2$ on the convergence complexity of *any* oblivious algorithm. Moreover, say that an algorithm is *partially oblivious* if its scheduling mechanism uses information about session rates, but is otherwise unaware of network topology. We observe that, perhaps surprisingly, the lower bound of $dn/4 + n/2$ on convergence complexity still holds for partially oblivious algorithms; this lower bound is matched by the convergence complexities of previously proposed algorithms that use knowledge of session rates.

On the positive side, we prove that the lower bound for oblivious algorithms is tight (up to a constant factor of two) as well. We do so by presenting an *optimal,* deterministic, oblivious algorithm, which converges after $dn/2 + n/2$ update operations are performed in the *worst* case; this algorithm is simple and schedules sessions for an increase in a *round-robin* fashion. In addition, we argue that randomness can be exploited to yield an even simpler oblivious algorithm at the cost of a small increase in convergence complexity. We present a *randomized,* oblivious algorithm, where each session S_i is scheduled with some positive probability. In the case of uniform probabilities, this algorithm converges to the max-min fair rates after an *expected* number of $O(dn \ln n)$ update operations. Finally, 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 "counter-example", *non-oblivious* algorithm, which schedules a session only if it traverses a certain, highly congested link in the network. We show that this algorithm converges to the max-min fair rates after an *optimal* number of n update operations in *every* case.

Our results imply a convergence complexity "collapse" between oblivious and partially oblivious algorithms, and, in sharp contrast, a convergence complexity separation between partially oblivious (in particular, oblivious) and non-oblivious algorithms for optimistic, bottleneck, rate-based flow control. Thus, in conclusion, knowledge of session rates cannot suffice by itself to improve the efficiency of oblivious algorithms, while it does so in combination with knowledge of the topology of the network.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design– *distributed networks, network communications, network topology*; C.2.2 [**Computer-Communication Networks**]: Network Protocols– *protocol architecture*; C.2.3 [**Computer-Communication Networks**]: Network Operations– *network management*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems– *distributed applications*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation– *parallelism and concurrency, probabilistic computation*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems– *computations on discrete structures, sequencing and scheduling*; G.2.1 [**Discrete Mathematics**]: Combinatorics– *combinatorial algorithms*; G.2.2 [**Discrete Mathematics**]: Graph Theory– *graph algorithms, network problems, path and circuit problems*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: distributed computing, lower bounds, randomization, rate-based flow control, convergence complexity, locality, oblivious, partially oblivious, and non-oblivious schedulers, generalized coupon collector's problem.

Contents

1 Introduction **6**

1.1 Motivation-Overview 6

1.2 Detailed Description 9

1.3 Organization 11

2 Mathematical Foundations **11**

2.1 Notation and Miscellaneous 11

2.2 The Update Function 12

2.3 The Generalized Coupon Collector's Problem 13

3 Definitions and Preliminaries **14**

3.1 Network, Sessions and Rate Vectors 14

3.2 Active Sessions and Active Edges 15

3.3 Fair Share and Minimum Fair Share 16

3.4 Bottleneck Edges and Minimum Fair Share Edges 17

3.5 The Update Operation 17

3.6 Schedulers and Terminators 19

3.6.1 Schedulers 19

3.6.2 Terminators 20

3.7 Executions 21

3.8 Convergence Complexity 23

4 Bottleneck Algorithms **24**

4.1 Properties of Bottleneck Edges 25

4.2 Properties of Minimum Fair Share Edges 26

4.3 Termination Properties 29

5 A Network Construction **30**

5.1 Construction 31

5.2 Example 33

5.3 Simple Properties 35

6 Lower Bounds **37**

6.1 Deterministic, Oblivious Algorithms 37

7 Deterministic, Partially Oblivious Algorithms **102**

8 Upper Bounds **102**

8.1 Deterministic, Oblivious Algorithms 103

8.2 Randomized, Oblivious Algorithms 103

8.3 Non-Oblivious Algorithms 105

9 Discussion and Directions for Further Research **105**

9.1 Review 105

9.2 Further Research 106

1 Introduction

1.1 Motivation-Overview

In most commercial *communication networks,* a connection between different users is established by a *session,* a virtual circuit involving a fixed path between a source and a destination.* Since such networks are usually designed with average traffic demands in mind, situations sometimes arise where the externally injected load is larger than what can be handled even with optimal routing. In these situations, queue sizes at bottleneck links may grow indefinitely and eventually exceed the buffer space at the linked nodes; in such event, packets arriving at nodes with no available buffer space will have to be discarded and later retransmitted, thereby wasting communication resources. Moreover, buffer overflow has been one of the main causes of process failures in distributed systems (see, e.g., [31]). The function of *flow control algorithms* is to alleviate throughput degradation, unfairness, deadlocks, and failures by preventing such situations from arising (see, e.g., [3, Chapter 6] or [4, 10, 15, 17, 22, 24, 25, 32]).

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, 3, 7, 16, 19, 20, 24]). The rate-based approach has been particularly attractive due to its simplicity and its modest hardware requirements per virtual circuit (as opposed to those of *credit-based* [29] or *virtual-channel-based* [11] approaches). Indeed, the Asynchronous Transfer Mode (ATM) Forum on Traffic Management has adopted rate-based flow control as the prime mechanism for flow control of Available Bit Rate (ABR) traffic in its networks (see, e.g., [5, 12, 30, 36, 38, 41]).

A widely accepted fairness criterion for rate-based flow control is *max-min fairness* [1, 3, 7, 8, 9, 19, 20, 21, 22, 24, 27], requiring that it be impossible for any session to receive a larger rate on the account of a session with a smaller or equal rate. Call *max-min rates* those achieving max-min fairness.

Any rate-based flow control algorithm can be classified into one of two broad classes, *conservative* and *optimistic,* according to the way in which rates of sessions are adjusted. Conservative algorithms converge without ever decreasing rates; thus, the rate of a session is never larger than its final rate. In contrast, optimistic algorithms allow decreases so that a session may intermediately receive a rate larger than its final. In a recent work, Afek *et al.* [1] introduced optimistic algorithms and advocated them to fit better than the conservative ones [3, Chapter 6] into "real" dynamic networks, where there is often need to decrease the rates of some sessions in order to accomodate new entering sessions; moreover, the optimistic approach seems to get closer to specifications for rate-based flow control algorithms suggested in the ATM Forum on Traffic Management (see, e.g., [26, 39]).

A significant component of any rate-based flow control algorithm, whether conservative or optimistic, is its *scheduler,* the abstract mechanism it uses to decide which session to increase

*In the model we consider, it is assumed that each session's lifetime is infinite; this assumption is suitable for applications such as ftp, batch jobs, and Internet navigation tools.

next. In favor of efficient and robust distributed implementations of the scheduler abstraction, it is desirable that the scheduler does not require *global* knowledge of either the session rates or the network topology. (See also [25, Section IV] for similar motivations regarding locality for rate-based flow control algorithms.) Clearly, such "decentralized" schedulers adjust more easily to dynamic changes in network topology; also, they are implementable more efficiently in a distributed environment in terms of both communication and computation, since they save both the communication between different sessions and the computation needed to derive which session to adjust next. So, one important parameter of a rate-based flow control algorithm, from the point of view of network performance, is its *locality,* characterized 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; apparently, an oblivious scheduler enjoys nice properties of efficiency, robustness, and portability. In particular, oblivious schedulers avoid both the problem of communication in a distributed algorithm, and the problem of the ambiguity in the meaning of rate for interactive traffic (cf. [3, Section 6.4.2]). On the opposite extreme, a *non-oblivious* scheduler requires full knowledge of both session rates and network topology; clearly, a non-oblivious scheduler is more amenable to "centralized", rather than distributed, implementations.

There is, in addition, an important middle ground between oblivious and non-oblivious schedulers: schedulers which, although unaware of network topology, do have access to session rates; call these schedulers *partially oblivious.* A partially oblivious scheduler is superior to a non-oblivious scheduler in terms of robustness to dynamic changes in network topology that leave rates unaffected; however, it is surpassed by an oblivious scheduler in being ambiguous with respect to the meaning of rate for interactive traffic. Afek *et al.* [1, Sections 4 & 5] present 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 significant component of any rate-based flow control algorithm is its *terminator,* the mechanism it uses to decide which sessions to "terminate" at each specific instant. *Bottleneck* terminators [24] finalize the rate of a session, thereby terminating the session, only if it cannot be further increased on some particular link of the network that allows for the least possible share of bandwidth to each session traversing it; such a link is called a *bottleneck edge* [24].

Once a session has been scheduled for an increase, a control message loops around its path and calculates on its way the minimum share of bandwidth the session may take from the excess capacities of links along the path; this may require possible adjustments to the rates of conflicting sessions (see, e.g., [7, 8, 9, 26, 27]). Roughly speaking, the *convergence complexity* of a rate-based flow control algorithm is the number of rate adjustments performed in the *worst* case till all sessions are terminated and max-min fairness is reached. Since reaching max-min fairness fast is vital for efficient utilization of the network resources, convergence complexity is another significant performance parameter of rate-based flow control algorithms. We measure convergence complexity in terms of a simple abstraction of *update* operations introduced by

Afek *et al.* [1, Section 2.1]; in short, an update operation adjusts the rates of a group of conflicting sessions in a both fair and optimistic way.[†]

In this work, we present 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 lower and upper bounds on convergence complexity for oblivious, partially oblivious, and non-oblivious such algorithms. Our lower bounds demonstrate that achieving oblivious, or even partially oblivious, scheduling, hence, locality, necessarily imposes a non-constant, multiplicative overhead on convergence complexity; the overhead is a function of the particular network. These are the first general lower bounds ever shown on the convergence of rate-based flow control algorithms. Moreover, our algorithms span a wide spectrum of convergence complexity bounds and locality properties, and they improve significantly upon the upper bounds shown by Afek *et al.* [1], with respect to the combination of these two performance measures.

The convergence complexities of the optimistic algorithms proposed in [1] have been expressed in terms of the total number n of sessions laid out on the network. We argue that this number does not accurately reflect characteristics that determine more critically the convergence properties of a rate-based algorithm. Consider, for example, one network with n "independent" sessions, and another network with n sessions any two of which share an edge. Clearly, adjusting the rate of one session in the first network does not influence any other session; thus, such adjustment cannot "retard" the convergence of any other session. On the other hand, adjusting the rate of one session in the second network results in corresponding changes to the rates of all other sessions; thus, sessions may retard each other from converging in a "chain-like" fashion. However, bounds on the convergence complexity expressed only in terms of n fail to capture this discrimination in a quantitative way. A significant contribution of our work is the identification of a parameter other than n and the derivation of more meaningful bounds on convergence complexity in terms of this parameter. Specifically, let d be the maximum number of sessions that share an edge either directly or indirectly;[‡] call d the *session dependency*. Thus, d is the maximum number of sessions affected by a single update operation on any session; clearly, $1 \leq d \leq n$. It turns out that the session dependency d, together with the number of sessions n, pins down the optimal convergence complexity achievable by oblivious or partially oblivious algorithms.

One major contribution of our work is the introduction of a novel methodology for a combinatorial construction, given any oblivious or partially oblivious scheduler, of a network that incurs high convergence complexity when sessions are scheduled according to this particular scheduler; our lower bounds are shown using this construction. Moreover, to establish our up-

[†]Roughly speaking, an update operation increases, if possible, the rate of the scheduled 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 sharing that 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.

[‡]More accurately, d is the maximum size of an equivalence class induced by the transitive closure of the "share-an-edge" relation on the set of sessions.

per bounds, we offer several new properties and tools for the design and analysis of optimistic, bottleneck, rate-based flow control algorithms. We use these tools in showing correctness and analyzing the convergence complexity of the specific algorithms we propose. However, some of the properties we show are used in the proofs of the lower bounds as well. In the sequel, we will focus on optimistic and bottleneck algorithms; for simplicity, we shall sometimes refer to them simply as algorithms (and avoid repeatedly recalling that they are optimistic and bottleneck).

1.2 Detailed Description

Our first major result is a fundamental lower bound on the convergence complexity of oblivious, optimistic, and bottleneck algorithms that compute the max-min rates. This relies on a novel technique for constructing a network on which some particular oblivious, optimistic, and bottleneck algorithm must incur high convergence complexity. More specifically, we provide a general methodology for constructing, given any fixed but arbitrary oblivious algorithm, a specific network, as a function of the algorithm's scheduler, so that if sessions are scheduled for increase on this network according to the scheduler and the algorithm indeed computes the max-min rates, then $\Theta(dn)$ update operations are required before convergence to the max-min rates. The construction is purely combinatorial and, therefore, of independent interest; it uses the sequence of sessions specified by the scheduler for appropriately defining the edges and assigning capacities to them, in a way that "retards" sessions from reaching their final (max-min) rates. We rely heavily on the algorithm being optimistic and bottleneck for showing that its convergence is slow.

Perhaps surprisingly, we observe that the lower bound of $\Omega(dn)$ we have shown on the convergence complexity of oblivious algorithms applies also to partially oblivious algorithms. We describe how to simultaneously "hand-craft," in a step-by-step fashion, both the sequence of sessions produced by *any* partially oblivious algorithm and the network for which $\Omega(dn)$ update operations are required for convergence to the max-min rates. It turns out that this lower bound is, indeed, tight for partially oblivious algorithms. As we observe, the upper bound of $\Theta(n^2)$ on convergence complexity shown by Afek *et al.* [1] for the partially oblivious algorithms GlobalMin and LocalMin, and for the case where $d = n$, implies a corresponding upper bound of $\Theta(dn)$ for these algorithms, for the case of general d. Thus, although intuition may suggest that knowledge of session rates can be crucial to performance, our results imply that this is *not* the case: our bounds for partially oblivious schedulers show that knowledge of session rates cannot by itself be of much help for reducing convergence complexity below the lower bound of $\Theta(dn)$ achievable by oblivious schedulers.

Moreover, we show a corresponding, matching upper bound on the convergence complexity of oblivious algorithms. We present an oblivious, deterministic algorithm, called RoundRobin, with convergence complexity bounded above by $dn/2 + n/2$. This algorithm borrows the very simple idea of scheduling different sessions on a "round-robin" basis;[§] the scheduler conducts d "rounds," in each of which all sessions are scheduled in a "round-robin" fashion.

[§]We remark that "round-robin" techniques have already been used in several (switching) algorithms for rate-

The algorithms GlobalMin and LocalMin presented by Afek *et al.*[1, Sections 4 & 5] achieve convergence complexity $\Theta(dn)$ as well; however, RoundRobin improves over them with respect to locality, since it is oblivious, while GlobalMin and LocalMin are partially oblivious.

Is it possible to use randomization for obtaining optimistic, rate-based flow control algorithms that are even simpler than RoundRobin, yet still oblivious? We answer this question in the affirmative by presenting an oblivious, randomized algorithm, called Random; indeed, Random is the first randomized algorithm ever proposed for rate-based flow control (either conservative or optimistic). A probability p_i is associated with each session S_i, $1 \leq i \leq n$, which is the probability the scheduler selects the session for an increase.

To determine the convergence complexity of Random, we consider and analyze an interesting generalization of the classical *coupon collector's problem* [37], where each coupon is drawn with a certain probability (as opposed to uniformly); call it the *generalized coupon collector's problem*. We study the probability distribution of the number of trials required to collect at least one of each type of coupon; more specifically, we show upper bounds on the expectation of this number. We use properties of bottleneck algorithms, and we exploit a correspondence with the *generalized coupon collector's problem* in order to show that the expectation of the number of update operations performed in any run of Random is $O(d \ln n / p_{min})$. where $p_{min} = \min_{1 \leq i \leq n} p_i$. We remark that the bound on the expectation of the convergence complexity of Random becomes $O(dn \ln n)$ in the *uniform* case, where all probabilities are equal to $1/n$; this bound is only slightly inferior to the corresponding bound shown for the still oblivious but somehow less simple algorithm RoundRobin.

At this point, it is natural to ask whether it is possible to beat the $\Theta(dn)$ bound on convergence complexity achievable by oblivious or partially oblivious algorithms, at the cost, possibly, of sacrificing locality. Perhaps not too 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 describe a "counter-example" (deterministic) algorithm Linear that achieves an *exact* bound of n on convergence complexity; however, Linear is neither oblivious nor partially oblivious. In a nutshell, Linear follows the old and natural idea of selecting for an increase a session on a certain, highly congested link of the network (see, e.g., [3, Section 6.4.2] or [22]).[¶] Clearly, Linear is *optimal* with respect to convergence complexity, since there are situations forcing each of n sessions to have to be updated at least once by any algorithm (whether oblivious or not). This result answers in the affirmative the main question left open by the work of Afek *et al.* [1] of whether or not there exists an optimistic, bottleneck, rate-based flow control algorithm achieving convergence complexity that is linear in the number of sessions n.

based flow control (see, e.g., [19, 20, 21]); however, none of these algorithms performs any kind of rate-adjustment operations of the kind we study in this paper.

[¶]This idea forms the basis of a simple and classical iterative algorithm to compute the max-min fairness rates [3, Section 6.4.2]. In each iteration, the rates of sessions that use the "most congested" link are fixed. In the next iteration, the fixed rates are subtracted from the corresponding link capacities, and the procedure repeats itself. We emphasize that this algorithm is not optimistic.

Our lower and upper bounds exhibit interesting trade-offs between convergence complexity and locality for optimistic, bottleneck, rate-based flow control algorithms. Among our three proposed algorithms, Random is the "simplest", hence the most "distributed", but attains the highest (expected) convergence complexity of the three. Alternatively, RoundRobin, though somehow less "simple" than Random, still enjoys locality but yet manages to achieve convergence complexity which is asymptotically optimal for the class of oblivious algorithms. On the other hand, the "centralized" algorithm Linear achieves optimal convergence complexity, but does not enjoy locality.

We emphasize that our work deviates from previous work on rate-based flow control (e.g., [3, 4, 15, 16, 17, 19, 20, 21, 22, 24, 25, 32]) carried out in the data networks community in that it adopts the optimistic approach, put forward in the pioneering work by Afek *et al.* [1]; this approach emerges around the update operation, which allows sessions to intermediately receive rates larger than their final (max-min fair) rates. We have managed to show that certain "classical" scheduling policies, such as round-robin [19, 20, 21] or scheduling a session using a bottleneck link [22], still work in the optimistic framework; most interestingly, we have shown that some of these policies are provably superior, in terms of either convergence complexity or locality (or both), to some other "classical" policies, such as scheduling the session with the globally smallest rate [3], that have been already studied within the optimistic framework by Afek *et al.* [1].

1.3 Organization

The rest of this paper is organized as follows. Section 2 introduces some mathematical machinery. Section 3 includes definitions for the formal model, while Section 4 exhibits fundamental properties of bottleneck algorithms. Our generic network construction is presented in Section 5. Sections 6 and 8 include our lower and upper bounds, respectively. We conclude, in Section 9, with a discussion of our results and some open problems.

2 Mathematical Foundations

This section is organized as follows. Section 2.1 reviews some preliminary notation and facts. The update function is treated in Section 2.2, while Section 2.3 summarizes material on a generalization of the coupon collector's problem.

2.1 Notation and Miscellaneous

We use $\Re$ to denote the set of real numbers. Let λ denote the empty sequence. For any sequence σ, denote $|\sigma|$ the *length* of σ; an infinite sequence has infinite length. For each integer $m \geq 1$, denote $[m] = \{1, \ldots, m\}$, and define the *m-th harmonic number* H_m to be $\sum_{l=1}^{m} 1/l$.

H_m grows with m in a logarithmic fashion; to be precise, $H_m \approx \ln m + \gamma$, where $\gamma = 0.577\ldots$ is *Euler's constant*. We continue with a simple combinatorial fact.

Lemma 2.1 *Consider any finite sum $n = \sum_{j \geq 1} n_j$. Then, $\sum_{j \geq 1} n_j^2 \leq n \cdot \max_{j \geq 1} n_j$.*

Proof: Clearly, $\sum_{j \geq 1} n_j^2 \leq \sum_{j \geq 1} (n_j \cdot \max_{j \geq 1} n_j) = \sum_{j \geq 1} n_j \cdot \max_{j \geq 1} n_j = n \cdot \max_{j \geq 1} n_j$, as needed. ■

We will use boldface, lower-case symbols to represent vectors. Consider any vector $\mathbf{r} = \langle r_1, r_2, \ldots, r_m \rangle^{\mathrm{T}}$. We will refer to the norms $\|\mathbf{r}\|_\infty = \max_{1 \leq l \leq m} |r_l|$, $\|\mathbf{r}\|_{-\infty} = \min_{1 \leq l \leq m} |r_l|$, and $\|\mathbf{r}\|_1 = \sum_{l=1}^{m} |r_l|$. For any integer $m \geq 1$, $\mathbf{0}^{(m)}$ denotes the vector with all m entries equal to zero.

For a random variable X, $\mathcal{E}(X)$ stands for the *expectation* of X. Recall that expectation distributes over any linear combination of random variables. In case X is *geometrically distributed with parameter ρ*, $\mathcal{E}(X) = 1/\rho$ (cf. [33, Appendix C]).

2.2 The Update Function

The following is a restatement of [1, Appendix A, Definition A.1] in the form of an abstract, mathematical function.

Fix any integer $m \geq 2$. The *update function* is a function `update` that takes as input a triple of a vector $\mathbf{r} \in \Re^m$, an integer $i \in [m]$, and a real number $c \in \Re$ such that $\|\mathbf{r}\|_1 \leq c$, and returns as output a pair $\langle \mathbf{r}', \Delta \rangle = \texttt{update}(\langle \mathbf{r}, i, c \rangle)$ of a vector $\mathbf{r}'$ and a real number $\Delta \geq 0$, so that the following conditions are satisfied:

(1) $\|\mathbf{r}'\|_1 = c$;

(2) $r'_i = r_i + \Delta$;

(3) for each $l \in [m]$, $l \neq i$, if $r_l \leq r_i + \Delta$, then $r'_l = r_l$;

(4) for each $l \in [m]$, $l \neq i$, if $r_l > r_i + \Delta$, then $r'_l = r_i + \Delta$.

Whenever $\langle \mathbf{r}', \Delta \rangle = \texttt{update}(\langle \mathbf{r}, i, c \rangle)$, call $\mathbf{r}'$ a *reform* for $\langle \mathbf{r}, i, c \rangle$ that corresponds to *reform increase* Δ (cf. [1, Appendix A, Definition A.1]). Intuitively, a reform "saturates" c (condition (2/a)) by increasing entry r_i by Δ (condition (2/b)), while entries that do not exceed the increased entry are not affected (condition (2/c)); moreover, the reform preserves "fairness" in some sense since, in addition, there can be left no entries larger than the increased entry (condition (2/d)). (We refer the reader to [13] for a formal proof that `update` is indeed a function.)

Consider, for example, the case where $m = 4$; take the vector $\mathbf{r} = \langle 3, 1, 4, 2 \rangle^{\mathrm{T}}$, and assume that $i = 2$ and $c = 12$. The reader is encouraged to verify that the vector $\mathbf{r}' = \langle 3, 7/2, 7/2, 2 \rangle^{\mathrm{T}}$ is a reform for $\langle \langle 3, 1, 4, 2 \rangle^{\mathrm{T}}, 2, 12 \rangle$ that corresponds to reform increase $\Delta = 5/2$.

2.3 The Generalized Coupon Collector's Problem

Our presentation borrows from [33, Section 3.6], which considers the "standard" coupon collector's problem [37].[‖] The reader may prefer to skip this section for now, returning to it later when its results are required.

In the *generalized coupon collector's problem,* there are m coupon types denoted $1, \ldots, m$; at each trial, a coupon is chosen at random. Each randomly chosen coupon is likely to be of type i, $1 \leq i \leq m$, with probability $p_i > 0$, where $\sum_{i=1}^{m} p_i = 1$. Denote $p_{min} = \min_{1 \leq i \leq m} p_i$; notice that $0 < p_{min} \leq 1/m$. The random choices of coupons are mutually independent. In the "standard" *coupon collector's problem* [37], $p_1 = \ldots = p_m = 1/m$. Let X be the random variable defined as the number of trials required to collect at least one of each type of coupon. We show:

Proposition 2.2 $\mathcal{E}(X) \leq (\ln m + \gamma)/p_{min}$

Proof: Our analysis generalizes the analysis for the "standard" coupon collector's problem given in [33, Section 3.6].

Let $c_1, c_2, \ldots, c_X$ denote the sequence of trials, where $c_i \in [m]$ denotes the type of the coupon drawn in the ith trial. Call the ith trial a *success* if either $i = 1$ or the type c_i were not drawn in any of the first $i - 1$ trials.

We divide the sequence $c_1, c_2, \ldots, c_X$ into *epochs,* where epoch i, $0 \leq i \leq m - 1$, begins with the trial following the ith success and ends with the trial on which we obtain the $(i+1)$st success. For each i, $0 \leq i \leq m - 1$, define the random variable X_i to be the number of trials in epoch i, so that $X = \sum_{i=0}^{m-1} X_i$.

Let ρ_i denote the probability of success on any trial in epoch i; that is, ρ_i is the probability of drawing one of the $m - i$ remaining coupon types, denoted $c_1, c_2, \ldots, c_{m-i}$. These coupon types are selected with probabilities $p_{c_1}, p_{c_2}, \ldots, p_{c_{m-i}}$, respectively. Clearly, $\rho_i = p_{c_1} + p_{c_2} + \ldots + p_{c_{m-i}} \geq (m - i)\, p_{min}$.

It holds that the random variable X_i is geometrically distributed with parameter ρ_i (cf. [33, Section 3.6.1]), so that $\mathcal{E}(X_i) = 1/\rho_i \leq 1/((m - i)p_{min})$.

Hence,

$$\begin{aligned} \mathcal{E}(X) &= \mathcal{E}\Big(\sum_{i=0}^{m-1} X_i\Big) \\ &= \sum_{i=0}^{m-1} \mathcal{E}(X_i) \qquad \text{(by linearity of expectation)} \end{aligned}$$

[‖] For several interesting variants of this classical problem and their analysis, see [23, Chapter 3, Exercises 21-25].

$$
\begin{aligned}
&\leq \sum_{i=0}^{m-1} \frac{1}{(m-i)\, p_{min}} \\
&= \frac{1}{p_{min}} \sum_{i=0}^{m-1} \frac{1}{m-i} \\
&= \frac{1}{p_{min}} \sum_{i=1}^{m} \frac{1}{i} \\
&= \frac{1}{p_{min}} H_m \\
&\approx \frac{1}{p_{min}} \left(\ln m + \gamma\right),
\end{aligned}
$$

as needed. ■

3 Definitions and Preliminaries

In this section, we present formal definitions and some preliminary facts. Many of our definitions formalize and refine corresponding ones in [1] and [3, Chapter 6].

This section is organized as follows. The basic entities of a network, and sessions and their rates are introduced in Section 3.1. The notions of active sessions and active edges are described in Section 3.2. Section 3.3 introduces the concepts of fair share and minimum fair share, which are used, in turn, in Section 3.4 to define bottleneck edges and minimum fair share edges. Section 3.5 formally defines the update operation. Schedulers and terminators are treated in Section 3.6. Executions are defined in Section 3.7, while Section 3.8 introduces convergence complexity.

3.1 Network, Sessions and Rate Vectors

A *communication network* is a directed graph $G = (V, E)$, where each vertex $v \in V$ represents a network node, and each edge $e \in E$ represents a link in the network. Associated with each edge $e \in E$ is a *capacity* $cap(e)$, which is a real number greater than zero.

A *session* is a sequence of links, which is a simple path in G between a *source* and a *destination*; a set of n sessions $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ is laid out on G, where $n \geq 2$. For each edge $e \in E$, denote $sessions(e)$ the set of sessions passing through e. For any set of sessions $\mathcal{S}' \subseteq \mathcal{S}$, denote $\mathcal{S}' \mid e$ the set $\mathcal{S}' \cap sessions(e)$. We will often abuse notation by identifying a session S_i with its index i. Moreover, 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 $e \in S_i$.

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 there exists an edge $e \in E$ such that both $S_i \in sessions(e)$ and

$S_j \in \mathit{sessions}(e)$; that is, $S_i \parallel_{\mathcal{S}} S_j$ if S_i and S_j share an edge "in parallel". Notice that $\parallel_{\mathcal{S}}$ is reflexive and symmetric. The transitive closure of $\parallel_{\mathcal{S}}$, denoted $\parallel^{\star}_{\mathcal{S}}$, is an equivalence relation on $\mathcal{S}$, which partitions $\mathcal{S}$ into equivalence classes $\mathcal{S}_1, \ldots, \mathcal{S}_c$, called *session clusters* or *clusters* for short, where $1 \leq c \leq n$ and $|\mathcal{S}_1| + \ldots + |\mathcal{S}_c| = n$; note that for all indices j and k, $1 \leq j, k \leq c$ and $j \neq k$, $\mathcal{S}_j \cap \mathcal{S}_k = \emptyset$, while $\cup_{1 \leq j \leq c} \mathcal{S}_j = \mathcal{S}$. Denote $d = \max\{|\mathcal{S}_1|, \ldots, |\mathcal{S}_c|\}$; that is, d is the maximum size of a session cluster. Call d the *session dependency*; clearly, $1 \leq d \leq n$.

For a sequence σ of session indices, and for any set of sessions $\mathcal{S}' \subseteq \mathcal{S}$, denote $\sigma \mid \mathcal{S}'$ the *restriction* of σ to indices of sessions in $\mathcal{S}'$. Denote $\sigma \uparrow \mathcal{S}'$ the shortest prefix of $\sigma \mid \mathcal{S}'$ that includes the indices of all sessions in $\mathcal{S}'$, in the order they appear in this prefix of $\sigma \mid \mathcal{S}'$ and with no repetitions, or $\sigma \mid \mathcal{S}'$, with no repetitions, however, while preserving the order of indices in σ, in case no such prefix exists. Denote $\sigma \downarrow \mathcal{S}'$ the remaining suffix of $\sigma \mid \mathcal{S}'$.

Allocated to each session S is a *rate* or *bandwidth,* denoted $r(S)$, which is a non-negative number. A *rate vector* is a vector $\mathbf{r} = \langle r(S_1), r(S_2), \ldots, r(S_n) \rangle^{\mathrm{T}}$. For any edge e, denote $\mathbf{r} \mid e$ the restriction of $\mathbf{r}$ to entries of sessions traversing e. The *capacity constraint* requires that for each edge $e \in E$, $\|\mathbf{r} \mid e\|_1 \leq \mathit{cap}(e)$; that is, the sum of rates of sessions sharing an edge does not exceed the edge capacity. A *feasible rate vector* is a rate vector that satisfies the capacity constraint. Say that an edge e is *saturated* if $\|\mathbf{r} \mid e\|_1 = \mathit{cap}(e)$.

A *maximal rate vector* is a feasible rate vector for which it is not possible to increase the rate of any session, while still maintaining feasibility, without decreasing the rate of some other session. A *max-min rate vector* is a maximal rate vector such that for each session S_i, $r(S_i)$ cannot be increased, while still maintaining feasibility, without decreasing $r(S_j)$ for some session S_j such that $r(S_j) \leq r(S_i)$; that is, an increase to the rate of any session requires a decrease to the rate of a session with equal or lower rate. Thus, a max-min rate vector is a maximal rate vector, but not vice versa. Clearly, the assumption that all capacities exceed zero implies that all entries of a max-min rate vector must exceed zero as well.

3.2 Active Sessions and Active Edges

The communication network is abstracted as a (possibly infinite) state machine. Each state Q of the network consists of two components: a *feasible rate vector* $\mathbf{r} \in \Re^n$, and an *active set* $\mathcal{A} \subseteq \mathcal{S}$, which is a set of *active sessions in state* Q; that is, $Q = \langle \mathbf{r}, \mathcal{A} \rangle$. We will sometimes use Q as an index in writing $\mathbf{r}_Q$ and $\mathcal{A}_Q$, in order to declare the state of reference. Intuitively, an active session is one that has not yet reached its "final" (max-min fair) rate. Denote $\mathcal{D}_Q = \mathcal{S} \setminus \mathcal{A}_Q$; that is, $\mathcal{D}$ is the set of *done sessions in state* Q, which are sessions whose rates have been finalized and will not change any further. Call a cluster $\mathcal{S}_j$ an *active cluster in state* Q if it contains at least one session that remains active in state Q; otherwise, $\mathcal{S}_j$ is called a *done cluster in state* Q. 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 entries of active and done sessions, respectively, in state Q.

In the *initial state* $Q^{in} = \langle \mathbf{r}^{in}, \mathcal{A}^{in} \rangle$, $\mathbf{r}^{in} = \mathbf{0}^{(n)}$ and $\mathcal{A}^{in} = \mathcal{S}$; that is, all sessions are initially active and have zero rates. A state Q is *final* if $\mathcal{A}_Q = \emptyset$; that is, in a final state, all sessions have reached their "final" rates.

The set of *active edges of the session cluster $\mathcal{S}_j$ in state Q,* denoted $AE_Q(\mathcal{S}_j)$, is defined to be $AE_Q(\mathcal{S}_j) = \bigcup_{S \in \mathcal{S}_j \cap \mathcal{A}_Q} \{e \mid e \in S\}$; that is, $AE_Q(\mathcal{S}_j)$ contains all edges of the network traversed by at least one active session in state Q that belongs to cluster $\mathcal{S}_j$. The *set of active edges of the network in state Q,* denoted AE_Q, is defined to be $AE_Q = \bigcup_{1 \le j \le c} AE_Q(\mathcal{S}_j)$; so, AE_Q contains all edges of the network traversed by at least one active session in state Q.

For any edge $e \in E$ and state Q, the *residual capacity of e in state Q,* denoted $residual_Q(e)$, is defined to be the difference between the capacity of edge e and the total rate of sessions traversing e in state Q; that is, $residual_Q(e) = cap(e) - \sum_{S \in sessions(e)} r_Q(S)$.

For any edge $e \in E$ and state Q, the *allotted capacity of e in state Q* [1, Section 2.1], denoted $allotted_Q(e)$, is defined to be the total rate already allocated to done sessions passing through the edge; that is, $allotted_Q(e) = \sum_{S \in \mathcal{D}_Q | e} r_Q(S)$.

3.3 Fair Share and Minimum Fair Share

For any state Q, fix any edge $e \in AE_Q$; thus, $|\mathcal{A}_Q \mid e| > 0$. The *fair share of e in state Q* [1, Section 2.1], denoted $FS_Q(e)$, is defined to be

$$FS_Q(e) \quad = \quad \frac{cap(e) - allotted_Q(e)}{|\mathcal{A}_Q \mid e|};$$

roughly speaking, $FS_Q(e)$ is the per session "share" of that portion of the capacity of edge e that has not yet been allocated to sessions traversing the edge that are done in state Q.

Roughly speaking, the next simple claim establishes that for any particular active edge, if each active session traversing it "receives" rate less than the fair share of the edge, then the edge is not saturated.

Lemma 3.1 *Fix any state Q and edge $e \in AE_Q$. Assume that for each $i \in \mathcal{A}_Q \mid e$, $r_Q(S_i) < FS_Q(e)$. Then, e is not saturated in state Q.*

Proof: Clearly,

$$\begin{aligned}
\sum_{i \in \mathcal{A}_Q | e} r_Q(S_i) \quad &< \quad \sum_{i \in \mathcal{A}_Q | e} FS_Q(e) && \text{(by assumption)} \\
&= \quad \sum_{i \in \mathcal{A}_Q | e} \frac{cap(e) - allotted_Q(e)}{|\mathcal{A}_Q \mid e|} && \text{(by definition of fair share)} \\
&= \quad |\mathcal{A}_Q \mid e| \ \cdot \frac{cap(e) - allotted_Q(e)}{|\mathcal{A}_Q \mid e|} \\
&= \quad cap(e) - allotted_Q(e)\,,
\end{aligned}$$

so that e is not saturated. ■

For each $i \in [n]$, the *minimum fair share of session S_i in state Q* [1, Definition 3.6], denoted $MFS_Q(S_i)$, is defined to be $MFS_Q(S_i) = \min_{e \in S_i} FS_Q(e)$; so, $MFS_Q(S_i)$ is the least among all fair shares attained in state Q by edges traversed by S_i.

3.4 Bottleneck Edges and Minimum Fair Share Edges

For any state Q, a *bottleneck edge for* Q [24] is an edge $e \in E$ such that for each session $S \in \mathcal{A}_Q \mid e$, $MFS_Q(S) = FS_Q(e)$; that is, for each active session passing through e, e is the one, among all edges traversed by the session, that attains the minimum fair share of the session in state Q. We continue to show that the fair shares of any two bottleneck edges shared by any single session are equal.

Lemma 3.2 *Let e and e' be bottleneck edges for state Q such that $\mathcal{A}_Q \mid e \cap \mathcal{A}_Q \mid e' \neq \emptyset$. Then, $FS_Q(e) = FS_Q(e')$.*

Proof: Take any session $S \in \mathcal{A}_Q \mid e \cap \mathcal{A}_Q \mid e'$. Since e and e' are bottleneck edges for state Q, both $MFS_Q(S) = FS_Q(e)$ and $MFS_Q(S) = FS_Q(e')$. It follows that $FS_Q(e) = FS_Q(e')$, as needed. ■

For each session cluster $\mathcal{S}_j$, the set of *edges with minimum fair share for cluster $\mathcal{S}_j$ in state* Q is defined to be

$$MFSE_Q(\mathcal{S}_j) \quad = \quad \{e \in AE_Q(\mathcal{S}_j) \mid FS_Q(e) \leq FS_Q(e') \text{ for each } e' \in AE_Q(\mathcal{S}_j)\}\,.$$

The next simple claim is a direct consequence of the definitions of a bottleneck edge and a minimum fair share edge for some particular session cluster.

Lemma 3.3 *For any state Q and any session cluster $\mathcal{S}_j$, consider any edge $e \in MFSE_Q(\mathcal{S}_j)$. Then, e is a bottleneck edge for Q.*

Proof: Take any session $S \in \mathcal{A}_Q \mid e$. Since $e \in MFSE_Q(\mathcal{S}_j)$, for each $e' \in AE_Q(\mathcal{S}_j)$, $FS_Q(e) \leq FS_Q(e')$. Thus, in particular, for any edge e' traversed by S, $FS_Q(e) \leq FS_Q(e')$, so that $FS_Q(e) \leq \min_{e' \in S} FS_Q(e')$. Since S traverses e, $FS_Q(e) \geq \min_{e' \in S} FS_Q(e')$. It follows that $FS_Q(e) = \min_{e' \in S} FS_Q(e')$. Since S was chosen arbitrarily, this implies that e is a bottleneck edge for Q, as needed. ■

The set of *edges with minimum fair share in state* Q, denoted $MFSE_Q$, is defined to be $MFSE_Q = \bigcup_{1 \leq j \leq c} MFSE_Q(\mathcal{S}_j)$; that is, an edge e is in the set of edges with minimum fair share if it is an edge with minimum fair share for some session cluster.

3.5 The Update Operation

For any session S_i, the *increase for S_i in state Q imposed by edge $e \in S_i$* [1, Section 2.1], denoted $\Delta_Q(i, e)$, is either 0 if S_i is not active in state Q, or equal to the reform increase for $\langle \mathbf{r}_{\mathcal{A}_Q} \mid e, i, cap(e) - allotted_Q(e) \rangle$; that is, $\Delta_Q(i, e)$ is the real number $\Delta > 0$ such that

$$\langle \mathbf{r}', \Delta \rangle \quad = \quad \texttt{update}(\langle \mathbf{r}_{\mathcal{A}_Q} \mid e, i, cap(e) - allotted_Q(e) \rangle)\,,$$

for some rate vector $\mathbf{r}'$. (Note that, by definition of update function, $\mathbf{r}'$ is "compatible" with $\mathbf{r}_{\mathcal{A}_Q} \mid e$; so, it contains only entries that correspond to sessions active in state Q that traverse edge e.) In more detail, $\Delta_Q(i,e)$ is the *unique* possible increase to the rate of session S_i in state Q that "saturates" edge e, while it possibly decreases down to the increased rate of S_i the rates of other active sessions passing through e that exceed that increased rate, but it does not affect the active sessions traversing e whose rates are smaller than the increases rate of S_i, neither, of course, the done sessions traversing e, or any other session in the network. Thus, $\Delta_Q(i,e)$ 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.

The *increase for S_i in state Q* [1, Section 2.1], denoted $\Delta_Q(i)$, is either 0 if S_i is not active in state Q, or the *minimum,* over all edges $e \in S_i$, of the increase for S_i in state Q imposed by edge e; that is, $\Delta_Q(i) = \min_{e \in S_i} \Delta_Q(i,e)$; thus, assuming S_i is active in state Q, $\Delta_Q(i)$ is the minimum among all possible increases to the rate of S_i that are imposed by the edges it traverses. Since the rate of S_i can be increased by a certain amount only if it can be increased by this amount, while still maintaining feasibility, on every edge it traverses, it follows that $\Delta_Q(i)$ is the *maximum* possible increase to the rate of S_i in state Q that maintains feasibility. We remark that $\Delta_Q(i)$ can be computed from information that is "local" to session S_i (cf. [25, Section IV]).

Roughly speaking, an update operation defines a "procedure" to compute new rates for a set a sessions, on the basis of their old rates. In more detail, an *update operation* [1, Section 2.1] is a set of specific instantiations of the update function, which refer to some particular session index i and state Q, followed by a corresponding computation of new rates for some sessions. In such update operation, each edge e traversed by S_i is associated with a particular instantiation of the update function. In the instantiation of the update function associated with edge e, $\Delta_Q(i,e)$ is computed by $\langle \mathbf{r}', \Delta_Q(i,e) \rangle = \texttt{update}(\langle \mathbf{r}_{\mathcal{A}_Q} \mid e, i, \mathit{cap}(e) - \mathit{allotted}_Q(e) \rangle)$. Finally, new rates for session S_i and for active sessions that "interfere" with S_i are computed, using $\Delta_Q(i)$, as shown below, where we use assignment notation to denote the new rates.

(1) $r(S_i) := r(S_i) + \Delta_Q(i)$;

(2) for each $l \in \mathcal{A}_Q$, $l \neq i$, such that $S_l \cap S_i \neq \emptyset$,

$$r(S_l) \quad := \quad \min\{r(S_l), r(S_i) + \Delta_Q(i)\}\,.$$

Notice that each instantiation of the update function computes an increase $\Delta_Q(i,e)$ that "saturates" edge e (according to the definition of the update function). However, an increase of $\Delta_Q(i)$ for any edge e traversed by S_i is, by definition of $\Delta_Q(i)$, no more (and possibly less) than the increase $\Delta_Q(i,e)$ that would "saturate" it (according to the definition of the update function). It follows that the final rates computed by the update operation "saturate" only the edge(s) "realizing" $\Delta_Q(i)$, and no other edges.

3.6 Schedulers and Terminators

Schedulers and terminators are introduced in Sections 3.6.1 and 3.6.2, respectively.

3.6.1 Schedulers

We consider both deterministic and randomized schedulers.

Deterministic Schedulers

A *deterministic scheduler* is a function Sched that maps a triple of a pair $\langle G, \mathcal{S} \rangle$ of a network G and a set of sessions laid out on G, a network state Q, and a state index $l \geq 1$ to a session index $i = \mathsf{Sched}(\langle \langle G, \mathcal{S} \rangle, Q, l \rangle)$.

Say that Sched is *oblivious* if for all pairs $\langle G, \mathcal{S} \rangle$ and Q, and $\langle \widehat{G}, \widehat{\mathcal{S}} \rangle$ and $\widehat{Q}$, and for any integer $l \geq 1$, $\mathsf{Sched}(\langle G, \mathcal{S} \rangle, Q, l) = \mathsf{Sched}(\langle \widehat{G}, \widehat{\mathcal{S}} \rangle, \widehat{Q}, l)$; loosely speaking, an oblivious scheduler uses no "knowledge" of either the topology of the network, or the rates and status (active or done) of sessions in choosing the session to schedule next for an increase. Thus, an oblivious scheduler may be identified with a (finite or infinite) sequence $\mathsf{Sched} = i_1, i_2, \ldots$, where for each $l \geq 1$, $i_l \in [n]$. As a particular example of an oblivious scheduler, consider the "elevator" scheduler

$$\mathsf{ElevSched} \quad = \quad 1, 2, \ldots, n, n, \ldots, 2, 1, 1, 2, \ldots, n, n, \ldots, 2, 1, \ldots, 1, 2, \ldots, n, n, \ldots, 2, 1, \ldots \; .$$

Say that Sched is *partially oblivious* if for all pairs $\langle G, \mathcal{S} \rangle$ and $\langle \widehat{G}, \widehat{\mathcal{S}} \rangle$, for any network state Q, and for any state index $l \geq 1$, $\mathsf{Sched}(\langle G, \mathcal{S} \rangle, Q, l) = \mathsf{Sched}(\langle \widehat{G}, \widehat{\mathcal{S}} \rangle, Q, l)$; loosely speaking, a partially oblivious scheduler uses no "knowledge" of the topology of the network in choosing the session to schedule next, while, however, it may use "knowledge" of the rates (and status) of sessions. Thus, a partially oblivious scheduler Sched may be identified with a (finite or infinite) sequence of functions Sched_l, one for each integer $l \geq 1$, where each function maps a network state Q to a session index $i = \mathsf{Sched}_l(Q)$.

As an example of a partially oblivious scheduler consider the scheduler $\mathsf{GlobalMinSched}$, introduced by Afek *et al.* [1, Section 4], where $\mathsf{GlobalMinSched}_l(Q)$ is defined to be the index of the active session with the *least* rate in state Q.

Notice that any oblivious scheduler is also partially oblivious, but not vice versa. A *non-oblivious* scheduler is a scheduler that is not partially oblivious. Intuitively, a non-oblivious scheduler uses "global" knowledge of both the rates and status of sessions and the topology of the network.

Randomized Schedulers

A *randomized scheduler* is a function Sched that maps a triple of a pair $\langle G, \mathcal{S}\rangle$ of a network G and a set of sessions $\mathcal{S}$ laid out on G, a network state Q, and a state index $l \geq 1$ to a (discrete) probability distribution $f = \mathsf{Sched}(\langle G, \mathcal{S}\rangle, Q, l)$ on $\mathcal{S}$. Roughly speaking, a randomized scheduler chooses the session to schedule next for an increase on the basis of a probability distribution it determines.

The notion of oblivious scheduler can be extended to randomized schedulers in the natural way. Say that Sched is *oblivious* if for all pairs $\langle G, \mathcal{S}\rangle$ and Q, and $\langle \widehat{G}, \widehat{\mathcal{S}}\rangle$ and $\widehat{Q}$, and for any integer $l \geq 1$, $\mathsf{Sched}(\langle G, \mathcal{S}\rangle, Q, l) = \mathsf{Sched}(\langle \widehat{G}, \widehat{\mathcal{S}}\rangle, \widehat{Q}, l)$; roughly speaking, a randomized oblivious scheduler uses no "knowledge" of either the topology of the network, or the status and rates of sessions for assigning scheduling probabilities to sessions. Thus, an oblivious randomized scheduler may be identified with a (finite or infinite) sequence $\mathsf{Sched} = f_1, f_2, \ldots$, where for each $l \geq 1$, f_l is a probability distribution on $\mathcal{S}$. Extending now the notion of partially oblivious scheduler to randomized schedulers is a straightforward task.

3.6.2 Terminators

A *terminator* is a function Term that maps a pair of a pair $\langle G, \mathcal{S}\rangle$ of a network G and a set of sessions $\mathcal{S}$ laid out on G, and a network state Q to a set of sessions $\mathcal{T} = \mathsf{Term}(\langle\langle G, \mathcal{S}\rangle, Q\rangle)$ such that $\mathcal{T} \subseteq \mathcal{A}_Q$; intuitively, Term decides which sessions among those still active should be "terminated"** For sake of simplicity, we shall sometimes omit the argument $\langle G, \mathcal{S}\rangle$ of Term when such is clear from context.

Say that Term is *bottleneck* [24] 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, where S traverses e, such that (1) e is a bottleneck edge for Q, and (2) $r_Q(S) = FS_Q(e)$. Whenever such an edge e exists and Term is bottleneck, we will say that e *causes the termination of session S in state Q*.

As an example of a bottleneck terminator, we mention the terminator $\mathsf{GlobalMinTerm}$, defined by Afek *et al.* [1, Section 4], which "terminates" an active session whenever both its rate is the least among the rates of all active sessions, and its rate cannot be further increased. Although the definition of $\mathsf{GlobalMinTerm}$ is apparently different from that of a bottleneck terminator, Afek *et al.* [1, Observation 4.1] prove that $\mathsf{GlobalMinTerm}$ is, in fact, a bottleneck terminator.

An *algorithm* is a pair $\mathsf{Alg} = \langle \mathsf{Sched}, \mathsf{Term}\rangle$. Say that Alg is *oblivious* if Sched is; say that Alg is *bottleneck* if Term is. In all of our discussion, we will refer to algorithms that are both oblivious and bottleneck.

**In all of our discussion, a "terminated" session is meant to be one whose rate is finalized.

3.7 Executions

We model a computation "manipulating" the rates and status of sessions laid out on the network as a sequence of update operations, each of which increases the rate of some scheduled session, while possibly decreasing the rates of other sessions that "interfere" with the one being increased. Formally, an *execution of* Alg *on network* G *with session set* $\mathcal{S}$ 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) $Q_0 = Q^{in}$;

(2) for each integer $l \geq 1$, $i_l = \mathsf{Sched}(\langle G, \mathcal{S} \rangle, Q_{l-1}, l-1)$;

(3) for each integer $l \geq 1$, 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) $r_{Q_l}(S_{i_l}) := r_{Q_{l-1}}(S_{i_l}) + \Delta_{Q_{l-1}}(i_l)$;
(ii) for each session $S \in \mathcal{A}_{Q_l}$, $S \neq S_{i_l}$, such that $S \cap S_{i_l} \neq \emptyset$,

$$r_{Q_l}(S) \quad = \quad \min\{r_{Q_{l-1}}(S), r_{Q_{l-1}}(S_{i_l}) + \Delta_{Q_{l-1}}(i_l)\};$$

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

We continue with an informal explanation of execution. Condition (1) asserts that the starting state in execution α is the initial state Q^{in}. Condition (2) guarantees that the session indices in α are determined according to Sched. Condition (3) specifies how the rates and status of sessions change from a state to the next. If the scheduled session is done or cannot be further increased, then nothing happens; otherwise, the rate of the scheduled session is increased by $\Delta_{Q_{l-1}}(i_l)$ (condition (3/a/i)), while rates of interfering session that exceed the increased rate are decreased down to it (condition (3/a/ii)). Finally, condition (3/b) provides for any possible termination of sessions in the subsequent state according to Term.

Call each state Q_l, $l \geq 1$, a *reachable state for execution* α. For any integer $l \geq 1$, define the *preceding state of state* Q_l *in execution* α, denoted $\overleftarrow{Q_l}$, to be the state Q_{l-1}; for any integer $l \geq 0$, define the *successor state of state* Q_l *in execution* α, denoted $\overrightarrow{Q_l}$, to be the state Q_{l+1}. For any state indices l and $l' \geq l$, write $Q_l \xrightarrow{\alpha} Q_{l'}$ to denote that Q_l *precedes* $Q_{l'}$ *in execution* α. For any state index $l \geq 1$, we will sometimes abuse language and say that *session* S_l *is scheduled in front of state* Q_l.

Naturally, we are interested in algorithms for which a final state is reachable for each possible execution. The class of bottleneck algorithms enjoys a somehow related, interesting property.

Proposition 3.4 (Bertsekas and Gallager [3]) *Assume that* Alg *is a bottleneck algorithm. Then, for any reachable final state* Q *of* Alg, $\mathbf{r}_Q$ *is a max-min rate vector.*

Say that Alg *computes the max-min rate vector* if a final state Q is reachable for each execution of Alg such that $\mathbf{r}_Q$ is a max-min rate vector. Proposition 3.4 implies that in order to show that any given bottleneck algorithm computes the max-min rate vector, it suffices to prove that it reaches a final state. (We will frequently use Proposition 3.4 in our later proofs of upper bounds.)

We continue with some properties that specify how several quantities of interest change during an execution. All of these properties refer to the execution α. We first state an immediate consequence of the definitions of allotted capacity and execution.

Claim 3.5 *For each integer $l \geq 1$ and edge $e \in E$,*

$$allotted_{Q_l}(e) \;=\; allotted_{Q_{l-1}}(e) + \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l})|e} r_{Q_{l-1}}(S_i)\,.$$

The next simple claim is an immediate implication of the definitions of fair share and execution. Intuitively, it establishes that the fair share of any particular edge is determined, in the course of an execution, by the set of sessions that are active in each state.

Claim 3.6 *For any edge $e \in E$, for any states Q and Q' such that $Q \xrightarrow{\alpha} Q'$, if $\mathcal{A}_Q \mid e = \mathcal{A}_{Q'} \mid e$, then, $FS_Q(e) = FS_{Q'}(e)$.*

Afek *et al.* [1, Lemma 3.4] show an interesting "monotonicity" property of fair share for any execution of a bottleneck algorithm.

Proposition 3.7 (Afek, Mansour and Ostfeld [1]) *Assume that* Alg *is a bottleneck algorithm. Then, for any integer $l \geq 1$ and edge $e \in AE_{Q_l}$, $FS_{Q_l}(e) \geq FS_{Q_{l-1}}(e)$.*

Roughly speaking, Proposition 3.7 implies that the fair share of an edge may not decrease, as long as there are still active sessions traversing the edge, while a bottleneck algorithm is "running".

We continue with some definitions that will be helpful later. For an execution α, for any indices l_1 and l_2, $0 \leq l_1 < l_2 < \infty$, define the set $\mathcal{S} \mid_\alpha (Q_{l_1}, Q_{l_2}] \subseteq \mathcal{S}$ to be $\mathcal{S} \mid_\alpha (Q_{l_1}, Q_{l_2}] = \{i_l \mid l_1 < l \leq l_2 \text{ and } i_l \in \mathcal{A}_{Q_{l_1}}\}$; 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 for an increase in any state following Q_{l_1} but not Q_{l_2}. For an execution α, for any index l_1, $0 \leq l_1 < \infty$, define the set $\mathcal{S} \mid_\alpha (Q_{l_1}, \infty] \subseteq \mathcal{S}$ to be $\mathcal{S} \mid_\alpha (Q_{l_1}, \infty] = \{i_l \mid l_1 < l \text{ and } i_l \in \mathcal{A}_{Q_{l_1}}\}$; roughly speaking, $\mathcal{S} \mid_\alpha (Q_{l_1}, \infty]$ contains indices of all sessions active in Q_{l_1} that are scheduled for an increase in 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.

For an execution α, for any state Q_l, the *least schedule for Q_l in α*, denoted $\widehat{l}$, is the least integer $l' \geq l$ such that $\mathcal{S} \mid_\alpha (Q_l, Q_{l'}] = \mathcal{A}_{Q_l}$ if $\mathcal{S} \mid_\alpha (Q_l, \infty] = \mathcal{A}_{Q_l}$, or ∞ if no such integer

exists. Intuitively, the least schedule for Q_l in α 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.

Denote $\alpha^{(l)} = i_{l+1}, Q_{l+1}, \ldots, i_{\widehat{l}}, Q_{\widehat{l}}$. Note that in case $\widehat{l} = \infty$, $\alpha^{(l)}$ is the suffix of α following state Q_l. Every execution α can be written as $\alpha = Q_0, \alpha^{(l_1)}, \alpha^{(l_2)}, \ldots$, where the sequence $l_1, l_2, \ldots$ is defined inductively as follows. For the basis case, $l_1 = 0$. Assume inductively that for any $r \geq 2$, we have defined $l_1, \ldots, l_{r-1}$. For the induction step, define $l_r = \widehat{l_{r-1}}$. Call each $\alpha^{(l_r)}$, $r \geq 1$, an *execution epoch in* α. Note that in case α is infinite, the number of execution epochs in α can still be finite if (and only if) there exists some index $r_0 \geq 0$ such that $l_{r_0} = \infty$.

An *execution fragment of* Alg *on network* G *with session set* $\mathcal{S}$ is any finite subsequence $\alpha' = Q_{i_0}, i_1, Q_{i_0+1}, \ldots, i_l, Q_{i_0+l}$ of some execution α of Alg on network G with session set $\mathcal{S}$. Say that a session index i *appears* in the execution fragment α' if $i = i_r$ for some integer r such that $1 \leq r \leq l$.

3.8 Convergence Complexity

Intuitively, an update operation is "executed" in execution α each time it holds, for any integer $l \geq 1$, that $\Delta_{Q_{l-1}}(i_l) > 0$;[††] recall that these are the cases in which there are changes to the rates of S_{i_l} and its "interfering" sessions that remain active in state Q_{l-1}. Moreover, we are naturally interested in the *shortest* possible prefix (if any) of an execution that ends with a final state. Thus, we define the *number of update operations in execution* α to be the number of integers $l \geq 1$ such that both $\Delta_{Q_{l-1}}(i_l) > 0$ and none of $Q_0, \ldots, Q_{l-1}$ is a final state.

The *convergence complexity of* Alg *on network* G *with session set* $\mathcal{S}$, denoted $\mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S} \rangle)$, is defined to be the maximum, over all executions of Alg on network G with session set $\mathcal{S}$, of the number of update operations in any execution. We shall often determine $\mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S} \rangle)$ by breaking it up into $\mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S} \rangle) = \sum_{j \geq 1} \mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S}_j \rangle)$, where $\mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S}_j \rangle)$ is defined in the natural way. The *convergence complexity of* Alg, denoted $\mathcal{U}_{\mathsf{Alg}}$, is defined to be $\mathcal{U}_{\mathsf{Alg}} = \max_{\langle G, \mathcal{S} \rangle} \mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S} \rangle)$; that is, $\mathcal{U}_{\mathsf{Alg}}$ is the maximum, over all pairs of a network G and a session set $\mathcal{S}$ laid out on G, of the convergence complexity of Alg on network G with session set $\mathcal{S}$. Clearly,

$$\begin{aligned} \mathcal{U}_{\mathsf{Alg}} &= \max_{\langle G, \mathcal{S} \rangle} \sum_{j \geq 1} \mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S}_j \rangle) \\ &= \sum_{j \geq 1} \max_{\langle G, \mathcal{S}_j \rangle} \mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S}_j \rangle) \,. \end{aligned}$$

We continue to mention a simple lower bound on $\mathcal{U}_{\mathsf{Alg}}$. Recall that the rate of each session is initially zero; however, by max-min fairness, it must attain a non-zero value in any final state since all edge capacities are greater than zero. Since each update operation increases the

[††] By definition of $\Delta_{Q_{l-1}}(i_l)$, this implies that S_{i_l} is active in state Q_{l-1}.

rate of exactly one session, it follows that at least n update operations are needed, so that $\mathcal{U}_{\mathsf{Alg}} \geq n$.

By way of example, [1, Theorem 4.3] implies an upper bound on $\mathcal{U}_{\mathsf{GlobalMin}}(\langle G, \mathcal{S}_j \rangle)$, for any particular cluster $\mathcal{S}_j$.

Proposition 3.8 (Afek, Mansour and Ostfeld [1]) $\mathcal{U}_{\mathsf{GlobalMin}}(\langle G, \mathcal{S}_j \rangle) \leq |\mathcal{S}_j| \, (|\mathcal{S}_j| + 1)/2$

We observe that the convergence complexity of GlobalMin is, in fact, no more than $dn/2 + n/2$.

Proposition 3.9 $\mathcal{U}_{\mathsf{GlobalMin}} \leq dn/2 + n/2$

Proof: Clearly,

$$\begin{aligned}
\mathcal{U}_{\mathsf{GlobalMin}} &= \sum_{j \geq 1} \max_{\langle G, \mathcal{S}_j \rangle} \mathcal{U}_{\mathsf{GlobalMin}}(\langle G, \mathcal{S}_j \rangle) \\
&\leq \sum_{j \geq 1} \frac{|\mathcal{S}_j| \, (|\mathcal{S}_j| + 1)}{2} && \text{(by Proposition 3.8)} \\
&= \frac{1}{2} \sum_{j \geq 1} |\mathcal{S}_j|^2 + \frac{1}{2} \sum_{j \geq 1} |\mathcal{S}_j| \\
&\leq \frac{\max_{j \geq 1} |\mathcal{S}_j| \, .n}{2} + \frac{n}{2} && \text{(by Lemma 2.1)} \\
&\leq \frac{dn}{2} + \frac{n}{2},
\end{aligned}$$

as needed. ■

We finally consider the case where Sched is randomized. Then, the number of update operations in any execution of Alg on network G with session set $\mathcal{S}$ is a random variable, induced by Sched, which is called the *convergence complexity of* Alg *on network* G *with session set* $\mathcal{S}$, and denoted $\mathcal{U}_{\mathsf{Alg}}(\langle G, \mathcal{S} \rangle)$. The convergence complexity of Alg, denoted $\mathcal{U}_{\mathsf{Alg}}$, is defined in the natural way. We shall deal with the probability distribution of $\mathcal{U}_{\mathsf{Alg}}$.

4 Bottleneck Algorithms

In this section, we present basic properties of bottleneck algorithms. All of these properties refer to an execution $\alpha = Q_0, i_1, Q_1, \ldots, i_l, Q_l, \ldots$ of any bottleneck algorithm; we shall omit reference to α when such is clear from context. Properties of bottleneck edges and minimum fair share edges appear in Sections 4.1 and 4.2, respectively. Section 4.3 builds on these properties to establish fundamental termination properties of bottleneck algorithms.

4.1 Properties of Bottleneck Edges

We strengthen Proposition 3.7 for the special case of bottleneck edges. We present a collection of "stability" properties for any edge that becomes bottleneck in the course of an execution of a bottleneck algorithm. Roughly speaking, we establish that no change in the fair share of an edge may occur once the edge becomes bottleneck and as long as it remains active, so that the edge remains bottleneck; moreover, the final rate of any active session traversing the edge is equal to this constant fair share. Formally, we show:

Proposition 4.1 (Stability properties of bottleneck edges) *Assume that* Alg *is a bottleneck algorithm. For any integer $l_0 \geq 0$, fix any edge $e \in E$ 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 $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_{l+1}}) \mid e$, $r_{Q_l}(S_i) = FS_{Q_{l_0}}(e)$.*

Proof: By induction on l. For the basis case where $l = l_0$, (1) holds trivially, (2) holds by assumption, and (3) holds by definition of a bottleneck algorithm. Assume inductively that for some integer $l \geq l_0$, the following hold for any integer l' such that $l_0 \leq l' < l$:

(1) $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$;

(2) e is a bottleneck edge for $Q_{l'}$;

(3) for any $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_{l'+1}}) \mid e$, $r_{Q_{l'}}(S_i) = FS_{Q_{l_0}}(e)$.

For the induction step, we show that:

(1) $FS_{Q_l}(e) = FS_{Q_{l_0}}(e)$;

(2) e is a bottleneck edge for Q_l;

(3) for any $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_{l+1}}) \mid e$, $r_{Q_l}(S_i) = FS_{Q_{l_0}}(e)$.

We start by proving (1). Clearly,

$$\begin{aligned} FS_{Q_l}(e) &= \frac{cap(e) - allotted_{Q_l}(e)}{|\mathcal{A}_{Q_l} \mid e|} \\ &= \frac{cap(e) - allotted_{Q_{l-1}}(e) - \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e} r_{Q_{l-1}}(S_i)}{|\mathcal{A}_{Q_l} \mid e|} \quad \text{(by Claim 3.5)} \\ &= \frac{|\mathcal{A}_{Q_{l-1}} \mid e| \, FS_{Q_{l-1}}(e) - \sum_{i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e} r_{Q_{l-1}}(S_i)}{|\mathcal{A}_{Q_l} \mid e|}. \end{aligned}$$

Consider any session index $i \in (\mathcal{A}_{Q_{l-1}} \setminus \mathcal{A}_{Q_l}) \mid e$. 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 induction hypothesis (condition (3)), $r_{Q_{l-1}}(S_i) = FS_{Q_{l_0}}(e)$. It follows that

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

which completes the proof of (1).

We continue to show (2). Take any session index $i \in \mathcal{A}_{Q_l} \mid e$. Clearly, $i \in \mathcal{A}_{Q_{l_0}} \mid e$. Since e is a bottleneck edge for Q_{l_0}, $FS_{Q_{l_0}}(e) = MFS_{Q_{l_0}}(S_i) = \min_{e' \in S_i} FS_{Q_{l_0}}(e')$. Consider any edge $e' \in S_i$. Since $S_i \in \mathcal{A}_{Q_l} \mid e$, it follows that $e' \in AE_{Q_l}$; thus, by Proposition 3.7, $FS_{Q_l}(e') \geq FS_{Q_{l_0}}(e')$, so 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 (1), 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, completing the proof of (2).

We finally show (3). Take any session index $i \in (\mathcal{A}_{Q_{l_0}} \setminus \mathcal{A}_{Q_{l+1}}) \mid e$. 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 *(i)* e' is a bottleneck edge for $Q_{l'}$, and *(ii)* $r_{Q_{l'}}(S_i) = FS_{Q_{l'}}(e')$.

Either $l_0 < l' < l$ or $l' = l$; thus, it follows either by induction hypothesis (condition (2)) or by (2) that e is a bottleneck edge for $Q_{l'}$ in either case. Together this and condition *(i)* imply, by Lemma 3.2, that $FS_{Q_{l'}}(e) = FS_{Q_{l'}}(e')$. Together this and condition *(ii)* imply that $r_{Q_{l'}}(S_i) = FS_{Q_{l'}}(e)$. Since $l \geq l'$ and $i \in \mathsf{Term}(Q_{l'})$, $r_{Q_l}(S_i) = r_{Q_{l'}}(S_i)$. Either $l_0 < l' < l$ or $l' = l$; thus, it follows either by induction hypothesis (condition (1)) or by (1) that $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$. It follows that $r_{Q_l}(S_i) = FS_{Q_{l_0}}(e)$, which completes the proof of (3). ■

4.2 Properties of Minimum Fair Share Edges

We start by showing a simple "stability" 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. Roughly speaking, we establish that the edge remains a minimum fair share edge.

Proposition 4.2 (Stability property of minimum fair share edge) *Assume that* Alg *is a bottleneck algorithm. 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 $e \in MFSE_{Q_l}(\mathcal{S}_j)$, this implies that $FS_{Q_{l_0}}(e) \leq FS_{Q_{l_0}}(e')$. By Lemma 3.3, e is a bottleneck edge for Q_{l_0}; thus, by Proposition 4.1(2), $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$. By Proposition 3.7, $FS_{Q_{l_0}}(e') \leq FS_{Q_{l'}}(e')$. It follows that $FS_{Q_{l'}}(e) \leq FS_{Q_{l'}}(e')$. Since e' was chosen arbitrarily, this implies that $e \in MFSE_{Q_{l'}}(\mathcal{S}_j)$, as needed. ■

We continue to show some kind of a "safety" property for any edge that becomes a minimum fair share edge for any particular cluster. Roughly speaking, we consider any session traversing a minimum fair share edge; we establish that no decrease of its rate below this particular minimum fair share is possible if, in the first place, the rate is no less than the minimum fair share. Formally, we show:

Proposition 4.3 (Safety property of minimum fair share edge) *Assume that* Alg *is a bottleneck algorithm. 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 $S \in \mathcal{A}_{Q_{l_0}} \mid e$ such that $r_{Q_{l_0}}(S) \geq FS_{Q_{l_0}}(e)$. Then, for any integer $l \geq l_0$, $r_{Q_l}(S) \geq FS_{Q_{l_0}}(e)$.*

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

Assume first that $r_{Q_l}(S) \geq r_{Q_{l-1}}(S)$. By induction hypothesis, this implies that $r_{Q_l}(S) \geq FS_{Q_{l_0}}(e)$, as needed. So assume that $r_{Q_l}(S) < r_{Q_{l-1}}(S)$; that is, the rate of S decreases from Q_{l-1} to Q_l. By definition of execution and update operation, this implies that $S \cap S_{i_l} \neq \emptyset$ and $r_{Q_l}(S) = r_{Q_l}(S_{i_l})$; fix any edge $e' \in S \cap S_{i_l}$. Let e'' be an edge such that $\Delta_{Q_{l-1}}(i_l) = \Delta_{Q_{l-1}}(i_l, e'')$; roughly speaking, the edge e'' "realizes" the least, among all edges traversed by S_{i_l}, of all maximum reform increases to the rate of S_{i_l} in state Q_{l-1}. By definition of execution and update operation, e'' is saturated in Q_l. We prove:

Lemma 4.4 $r_{Q_l}(S_{i_l}) \geq FS_{Q_l}(e'')$

Proof: Assume, by way of contradiction, that $r_{Q_l}(S_{i_l}) < FS_{Q_l}(e'')$. By definition of execution and update operation, for any session index $k \in \mathcal{A}_{Q_{l_0}} \mid e''$, $r_{Q_l}(S_{i_l}) \geq r_{Q_l}(S_k)$; thus, $FS_{Q_l}(e'') > r_{Q_l}(S_k)$. Lemma 3.1 implies that e'' is not saturated in Q_l. A contradiction. ■

Since $r_{Q_l}(S) = r_{Q_l}(S_{i_l})$, Lemma 4.4 implies that $r_{Q_l}(S) \geq FS_{Q_l}(e'')$. By Proposition 3.7, $FS_{Q_l}(e'') \geq FS_{Q_{l_0}}(e'')$, so that $r_{Q_l}(S) \geq FS_{Q_{l_0}}(e'')$. Since $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, $FS_{Q_{l_0}}(e'') \geq FS_{Q_{l_0}}(e)$. It follows that $r_{Q_l}(S) \geq FS_{Q_{l_0}}(e)$, as needed. ■

The next claim complements Proposition 4.3 by providing some kind of a "liveness" property for any edge that becomes a minimum fair share edge for any particular cluster. Roughly speaking, we consider any session traversing a minimum fair share edge; we establish that no decrease of its rate below this particular minimum fair share is possible once all sessions traversing this minimum fair share edge have been scheduled at least once. Formally, we show:

Proposition 4.5 (Liveness property of minimum fair share edge) *Assume that* Alg *is a bottleneck algorithm. 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* $S \in \mathcal{A}_{Q_{l_0}} \mid e$*. Then, for any integer* $l \geq \widehat{l}_0 \mid e$*,* $r_{Q_l}(S) \geq FS_{Q_{l_0}}(e)$*.*

Proof: We start with an informal outline of our proof. We consider the point of the execution where S is being scheduled, that comes no later than when all sessions have been scheduled at least once. We establish that, at this point, the rate of S is no less than the fair share of e in state Q_{l_0}. We argue that e remains a minimum fair edge beyond 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 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 3.3 implies that e is a bottleneck edge for Q_{l_0}. Since $S \in \mathcal{A}_{Q_{l_0}} \mid e$, it follows by definition of $\widehat{l}_0 \mid e$ that $S = \mathsf{Sched}(\langle G, \mathcal{S} \rangle, Q_{l'}, l')$ for some integer l' such that $l_0 < l' \leq \widehat{l}_0 \mid e$. By Proposition 4.1(1), $FS_{Q_{l'}}(e) = FS_{Q_{l_0}}(e)$, while by Proposition 4.1(2), e is a bottleneck edge for $Q_{l'}$. We continue to prove:

Lemma 4.6 $r_{Q_{l'}}(S) \geq FS_{Q_{l_0}}(e)$

Proof: Assume, by way of contradiction, that $r_{Q_{l'}}(S) < FS_{Q_{l_0}}(e)$. Let e' be an edge such that $\Delta_{Q_{l'}}(S) = \Delta_{Q_{l'}}(S, e')$; roughly speaking, the edge e' "realizes" the least, among all edges traversed by S, of all maximum reform increases to the rate of S in state $Q_{l'}$. By definition of execution and update operation, e' is saturated in $Q_{l'}$.

Since e is a bottleneck edge for $Q_{l'}$, and S 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 $r_{Q_{l'}}(S) < FS_{Q_{l_0}}(e)$, it follows that $r_{Q_{l'}}(S) < FS_{Q_{l'}}(e')$. By definition of execution and update operation, for any session index $k \in \mathcal{A}_{Q_{l'}} \mid e$, $r_{Q_{l'}}(S_k) \leq r_{Q_{l'}}(S)$, so that $r_{Q_{l'}}(S_k) < FS_{Q_{l'}}(e')$. It follows by Lemma 3.1 that e' is not saturated in $Q_{l'}$. A contradiction. ■

Take any integer $l \geq \widehat{l}_0 \mid e$. Clearly, $l \geq l'$. By Proposition 4.3, Proposition 4.2 and Lemma 4.6 imply that $r_{Q_l}(S) \geq FS_{Q_{l_0}}(e)$, as needed. ■

4.3 Termination Properties

In this section, we establish fundamental termination properties of bottleneck algorithms. 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. Formally, we show:

Theorem 4.7 (Termination of all sessions) *Assume that* Alg *is a bottleneck algorithm. 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* $S \in \mathcal{A}_{Q_{l_0}} \mid e$*,* $S \in \mathcal{D}_{Q_{\widehat{l_0|e}}}$*.*

Proof: We start with an informal outline of our proof. We consider any session that remains 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, these will be sufficient to deduce that the session has reached its final rate. We now present the details of the formal proof.

Fix any $S \in \mathcal{A}_{Q_{l_0}} \mid e$. We start by proving:

Lemma 4.8 $r_{Q_{\widehat{l_0|e}}}(S) = FS_{Q_{l_0}}(e)$

Proof: Assume, by way of contradiction, that $r_{Q_{\widehat{l_0|e}}}(S) \neq FS_{Q_{l_0}}(e)$. By Proposition 4.5, $r_{Q_{\widehat{l_0|e}}}(S) \geq FS_{Q_{l_0}}(e)$. It follows that $r_{Q_{\widehat{l_0}}|e}(S) > FS_{Q_{l_0}}(e)$. Thus,

$$\begin{aligned}
\sum_{i \in \mathcal{A}_{Q_{l_0}}|e} r_{Q_{\widehat{l_0|e}}}(S_i) &= r_{Q_{\widehat{l_0|e}}}(S) + \sum_{\substack{i \in \mathcal{A}_{Q_{l_0}}|e, \\ S_i \neq S}} r_{Q_{\widehat{l_0|e}}}(S_i) \\
&> FS_{Q_{l_0}}(e) + \sum_{\substack{i \in \mathcal{A}_{Q_{l_0}}|e, \\ S_i \neq S}} r_{Q_{\widehat{l_0|e}}}(S_i) \\
&\geq FS_{Q_{l_0}}(e) + \sum_{\substack{i \in \mathcal{A}_{Q_{l_0}}|e, \\ S_i \neq S}} FS_{Q_{l_0}}(e) \quad \text{(by Proposition 4.5)} \\
&= \sum_{i \in \mathcal{A}_{Q_{l_0}}|e} FS_{Q_{l_0}}(e) \\
&= |\mathcal{A}_{Q_{l_0}} \mid e| \, FS_{Q_{l_0}}(e) \\
&= cap(e) - allotted_{Q_{l_0}}(e)\,,
\end{aligned}$$

a contradiction to the capacity constraint. ■

Since $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, Lemma 3.3 implies that e is a bottleneck edge for Q_{l_0}. By definition of $\widehat{l_0} \mid e$, $e \in AE_{Q_{\widehat{l_0}|e}}$. Thus, by Proposition 4.1(2), e is a bottleneck edge for $Q_{\widehat{l_0}|e}$; however, by Proposition 4.1(1), $FS_{Q_{\widehat{l_0}|e}}(e) = FS_{Q_{l_0}}(e)$, so that, by Lemma 4.8, $r_{Q_{\widehat{l_0}|e}}(e) = FS_{Q_{\widehat{l_0}|e}}(e)$. Since Term is bottleneck, these imply that $S \in \mathcal{D}_{Q_{\widehat{l_0}|e}}$. ■

Our final termination property is a direct consequence of Theorem 4.7; roughly speaking, we establish that scheduling any sequence of sessions that includes all active ones results in allocating a final rate to at least one active session per cluster.

Theorem 4.9 (Termination of at least one session per cluster) *Assume that* Alg *is a bottleneck algorithm. For any integer $l_0 \geq 0$, fix any active cluster $\mathcal{S}_j$ in Q_{l_0}. Then, there exists some session $S \in \mathcal{S}_j \cap \mathcal{A}_{Q_{l_0}}$ such that $S \in \mathcal{D}_{Q_{\widehat{l_0}}}$.*

Proof: Fix any edge $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, and consider any session $S \in \mathcal{A}_{Q_{l_0}} \mid e$. Clearly, $S \in \mathcal{S}_j \cap \mathcal{A}_{Q_{l_0}}$. By Theorem 4.7, $S \in \mathcal{D}_{Q_{\widehat{l_0}}}$, as needed. ■

We remark that all proofs of properties of bottleneck algorithms in this section have relied on the definition of a bottleneck terminator; however, these proofs have made no assumption on the scheduler, since they started with any arbitrary execution. Thus, these properties of bottleneck algorithms apply to both deterministic and randomized schedulers.

5 A Network Construction

In this section, we present a generic, combinatorial construction of a capacitated network associated with any (possibly infinite) sequence $\mathsf{Seq} = \sigma_1, \sigma_2, \ldots$, where for each $i \geq 1$, $\sigma_i \in [n]$. Fix any even integer d, and choose any integer n that is a multiple of d.[‡‡] 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.

In our construction, we will use, for assigning capacities to network edges, two (finite) sequences of real numbers, b and p (for "bottom" and "potential," respectively), each of length $d/2$, defined recursively as follows.

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

The next claim establishes a preliminary, combinatorial property of the sequences b_r and p_r that will be used in a later proof.

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

Proposition 5.1 *For any index $r \geq 1$,*

$$b_r + \frac{p_r}{4} \quad < \quad b_{r-1} + \frac{3}{4} p_{r-1} \, .$$

Proof: Clearly,

$$\begin{aligned} b_r + \frac{p_r}{4} &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{1}{4} \cdot \frac{p_{r-1}}{4} \quad \text{(by recursive definitions of } b_r \text{ and } p_r\text{)} \\ &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{16} \\ &= b_{r-1} + \frac{9}{16} p_{r-1} \\ &< b_{r-1} + \frac{3}{4} p_{r-1} \, , \end{aligned}$$

as needed. ■

The rest of this section is organized as follows. The main body of the construction appears in Section 5.1, while Section 5.2 provides an illustrating example. Finally, Section 5.3 establishes simple properties of the construction.

5.1 Construction

Partition $\mathcal{S}$ into n/d clusters $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n/d}$ so that for each j, $1 \leq j \leq n/d$, $\mathcal{S}_j$ 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 $G(\mathsf{Seq}) = (\cup_{1 \leq j \leq n/d} V_j(\mathsf{Seq} \mid \mathcal{S}_j), \cup_{1 \leq j \leq n/d} E_j(\mathsf{Seq} \mid \mathcal{S}_j))$; thus, each individual network G_j is a function of the sequence $\mathsf{Seq} \mid \mathcal{S}_j$, and the network G is the "parallel composition" of the individual networks. The construction of the network G_j proceeds in a sequence of no more than $d/2$ *epochs*; in epoch r, $1 \leq r \leq d/2$, the network $G_j^{(r)} = (V_j^{(r)}, E_j^{(r)})$ is constructed, so that $G_j = (\cup_{1 \leq r \leq d/2} V_j^{(r)}, \cup_{1 \leq r \leq d/2} E_j^{(r)})$; thus, the network G_j is the "parallel composition" of the individual networks $G_j^{(r)}$.

For each r, $1 \leq r \leq d/2$, the construction of the network $G_j^{(r)}$ 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, respectively, indices of $\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}$, or arbitrary indices from $\mathcal{I}_j^{(r)}$, in case $|\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}| < 2$.

These sets and sequences are 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 through the following case analysis.

1. Assume first that $\mathsf{Seq}_j^{(1)} = \lambda$, 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. Assume now 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. assume that $|\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)}| = 1$; 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. assume that $|\mathsf{Seq}_j^{(1)} \uparrow \mathcal{I}_j^{(1)}| > 1$, 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 Seq or arbitrary indices if Seq misses any of 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, $2 \leq r < d/2$. 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 the following case analysis.

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 indices in $\mathcal{I}_j^{(r)}$.

2. Assume now that $\mathsf{Seq}_j^{(r)} \neq \lambda$, so that $\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)} \neq \lambda$; there are again two cases to consider:

   a. assume that $|\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}| = 1$; then, $i_f^{(r)} := \mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}$, and fix $i_l^{(r)}$ to be any index in $\mathcal{I}_j^{(r)}$;

   b. assume that $|\mathsf{Seq}_j^{(r)} \uparrow \mathcal{I}_j^{(r)}| > 1$, 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

restring 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 arbitrary indices in case this suffix misses any such indices.

Note that since two different sessions are "extracted" from $\mathcal{S}_j$ in each of the $d/2$ epochs, all sessions in $\mathcal{S}_j$ will eventually participate in the construction.

We proceed to describe the construction of the network $G_j^{(r)}$, for any integer r, $1 \leq r \leq d/2$.

- Sessions $i_f^{(r)}$ and $i_l^{(r)}$ share an edge $e_{i_l}^{(r)}$ with $cap(e_{i_l}^{(r)}) = 2b_r + p_r/2$;
- for each $i \in \mathcal{I}_j^{(r)} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, sessions $i_f^{(r)}$ and i share an edge $e_i^{(r)}$ with $cap(e_i^{(r)}) = 2b_r + p_r$.

Informally, $i_f^{(r)}$ shares an edge with every other session in $\mathcal{I}_j^{(r)}$; however, the capacity of the edge shared with $i_l^{(r)}$ is the smallest, while all other capacities, are equal.

5.2 Example

We work out a detailed example to demonstrate the generic network construction. Fix $d = 6$ and choose $n = 12$. Consider the (infinite) "elevator" sequence

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

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 $12/6 = 2$ 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

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

Thus, $\mathsf{ElevSched}_1^{(1)} \downarrow \mathcal{S}_1^{(1)} = 1, 2, \ldots, 6$, so that $f_1^{(1)} = 1$ and $l_1^{(1)} = 6$. We continue to describe the construction of the network $G_1^{(1)}$:

- sessions 1 and 6 share an edge $e_6^{(1)}$ with $cap(e_6^{(1)}) = 2b_1 + p_1/2 = 512$.
- for each $i \in \{2, 3, 4, 5\}$, sessions 1 and i share an edge $e_i^{(1)}$ with $cap(e_i^{(1)}) = 2b_1 + p_1 = 1024$.

In addition, $\mathcal{I}_2^{(1)} = \mathcal{S}_2$, and

$$\begin{aligned}\mathsf{ElevSched}_2^{(1)} &= \mathsf{ElevSched} \mid \mathcal{S}_2 \\ &= 7, 8, \ldots, 12, 12, 11, \ldots, 7, 7, 8, \ldots, 12, 12, 11, \ldots, 7, \ldots, 7, 8, \ldots, 12, \ldots\ .\end{aligned}$$

Thus, $\mathsf{ElevSched}_2^{(1)} \downarrow \mathcal{S}_2^{(1)} = 7, 8, \ldots, 12$, so that $f_2^{(1)} = 7$ and $l_2^{(1)} = 12$. The construction of the network $G_2^{(1)}$ is similar:

- sessions 7 and 12 share an edge $e_{12}^{(1)}$ with $cap(e_{12}^{(1)}) = 2b_1 + p_1/2 = 512$.
- for each $i \in \{8, 9, 10, 11\}$, sessions 7 and i share an edge $e_i^{(1)}$ with $cap(e_i^{(1)}) = 2b_1 + p_1 = 1024$.

We proceed to the case where $r = 2$, which corresponds to the second epoch. Clearly, $\mathcal{I}_1^{(2)} = \mathcal{I}_1^{(1)} \setminus \{1, 6\} = \{2, 3, 4, 5\}$, and

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

Thus, $\mathsf{ElevSched}_1^{(2)} \uparrow \mathcal{I}_1^{(2)} = 5, 4, 3, 2$, so that $f_1^{(2)} = 5$ and $l_1^{(2)} = 2$. We continue to describe the construction of the network $G_1^{(2)}$:

- sessions 5 and 2 share an edge $e_2^{(2)}$ with $cap(e_2^{(2)}) = 2b_2 + p_2/2 = 1152$.
- for each $i \in \{3, 4\}$, sessions 5 and i share an edge $e_i^{(2)}$ with $cap(e_i^{(2)}) = 2b_2 + p_2 = 1280$.

In addition, $\mathcal{I}_2^{(2)} = \mathcal{I}_2^{(1)} \setminus \{7, 12\} = \{8, 9, 10, 11\}$, and

$$\begin{aligned}\mathsf{ElevSched}_2^{(2)} &= (\mathsf{ElevSched}_2^{(1)} \downarrow \mathcal{I}_2^{(1)}) \mid \mathcal{I}_2^{(2)} \\ &= 11, 10, 9, 8, 8, 9, 10, 11, 11, 10, 9, 8, 8, 9, 10, 11, \ldots, 11, 10, 9, 8, 8, 9, 10, 11, \ldots\ .\end{aligned}$$

Thus, $\mathsf{ElevSched}_2^{(2)} \uparrow \mathcal{I}_2^{(2)} = 11, 10, 9, 8$, so that $f_2^{(2)} = 11$ and $l_2^{(2)} = 8$. We continue to describe the construction of the network $G_2^{(2)}$:

- sessions 11 and 8 share an edge $e_8^{(2)}$ with $cap(e_8^{(2)}) = 2b_2 + p_2/2 = 1152$.
- for each $i \in \{9, 10, \}$, sessions 11 and i share an edge $e_i^{(2)}$ with $cap(e_i^{(2)}) = 2b_2 + p_2 = 1280$.

We proceed to the case where $r = 3$, which corresponds to the third epoch. Clearly, $\mathcal{I}_1^{(3)} = \mathcal{I}_1^{(2)} \setminus \{2, 5\} = \{3, 4\}$, and

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

Thus, $\mathsf{ElevSched}_1^{(3)} \uparrow \mathcal{I}_1^{(3)} = 3, 4$, so that $f_1^{(3)} = 3$ and $l_1^{(3)} = 4$. We continue to describe the construction of the network $G_1^{(3)}$:

- sessions 3 and 4 share an edge $e_4^{(3)}$ with $cap(e_4^{(3)}) = 2b_3 + p_3/2 = 2 \cdot 640 + 64/2 = 1312$.

In addition, $\mathcal{I}_2^{(3)} = \mathcal{I}_2^{(2)} \setminus \{8, 11\} = \{9, 10\}$, and

$$\begin{aligned} \mathsf{ElevSched}_2^{(3)} &= (\mathsf{ElevSched}_2^{(2)} \downarrow \mathcal{I}_2^{(2)}) \mid \mathcal{I}_2^{(3)} \\ &= 9, 10, 10, 9, 9, 10, 10, 9, \ldots, 9, 10, 10, 9, \ldots \ . \end{aligned}$$

Thus, $\mathsf{ElevSched}_2^{(3)} \uparrow \mathcal{I}_2^{(3)} = 9, 10$. We continue to describe the construction of the network $G_2^{(3)}$:

- sessions 9 and 10 share an edge $e_{10}^{(3)}$ with $cap(e_{10}^{(3)}) = 2b_3 + p_3/2 = 2 \cdot 640 + 64/2 = 1312$.

The network $G(\mathsf{ElevSched})$ is shown in Figure 1.

5.3 Simple Properties

Clearly, for each index r, $1 < r \leq d/2$, $\mathcal{I}_j^{(r)}$ contains neither $i_f^{(r-1)}$ nor $i_l^{(r-1)}$. Since $\mathcal{I}_j^{(d/2)} \subset \mathcal{I}_j^{(d/2-1)} \subset \ldots \subset \mathcal{I}_j^{(1)}$, it follows that for any index r', $r < r' \leq d/2$, $\mathcal{I}_j^{(r')}$ contains neither $i_f^{(r-1)}$ nor $i_l^{(r-1)}$. Thus, by construction of the network $G_j^{(r')}$, $r \leq r' \leq d/2$, it immediately follows.

Observation 5.1 *For each index r where $1 < r \leq d/2$, for each index r' such that $r \leq r' \leq d/2$, neither $i_f^{(r-1)}$ nor $i_l^{(r-1)}$ traverses an edge in $E_j^{(r')}$.*

We continue to show that the capacity of any edge in some particular epoch exceeds the capacity of every edge in the previous epoch.

Lemma 5.2 *For each index r where $1 < r \leq d/2$, let $e^{(r)}$ and $e^{(r-1)}$ be any edges in $E_j^{(r)}$ and $E_j^{(r-1)}$, respectively. Then, $cap(e^{(r)}) > cap(e^{(r-1)})$.*

Proof: By construction, $cap(e^{(r)}) \geq 2b_r + p_r/2$, and $cap(e^{(r-1)}) \leq 2b_{r-1} + p_{r-1}$. Thus, it suffices to prove that $2\,b_r + p_r/2 > 2\,b_{r-1} + p_{r-1}$. Clearly,

$$\begin{aligned} 2b_r + \frac{p_r}{2} &= 2\left(b_{r-1} + \frac{p_{r-1}}{2}\right) + \frac{p_r}{2} && \text{(by definition of } b_r) \\ &= 2b_{r-1} + p_{r-1} + \frac{p_r}{2} \\ &> 2b_{r-1} + p_{r-1} && \text{(since } p_r \geq 0)\,, \end{aligned}$$

as needed. ■

Lemma 5.2 immediately implies that the capacity of any edge in some particular epoch exceeds the capacity of every edge in all previous epochs:

Corollary 5.3 *For each index r where $1 < r \leq d/2$, for each index r' such that $1 \leq r' < r$, let $e^{(r)}$ and $e^{(r')}$ be any edges in $E_j^{(r)}$ and $E_j^{(r')}$, respectively. Then, $cap(e^{(r)}) > cap(e^{(r')})$.*

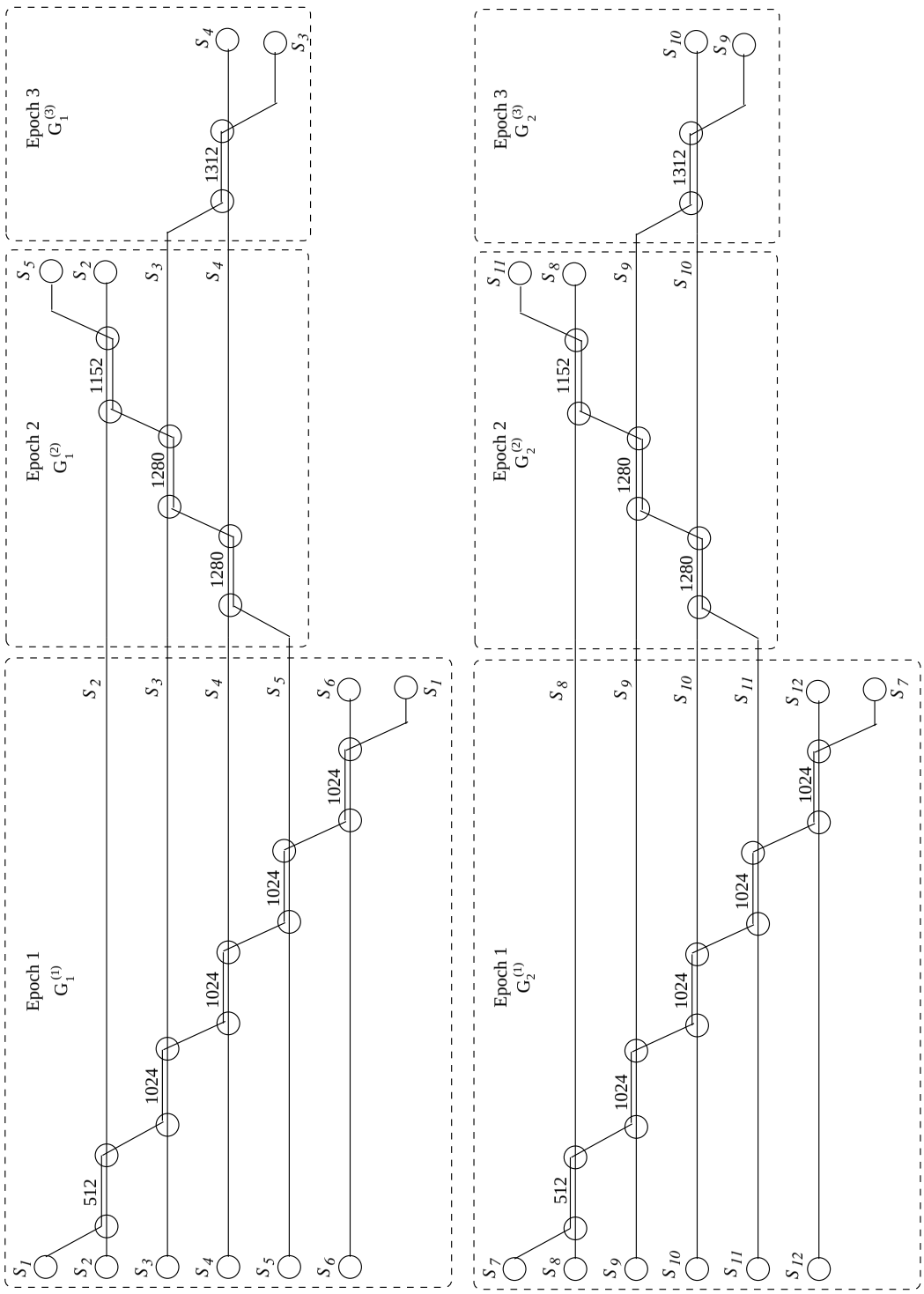

Figure 1: The network $G(\mathsf{ElevSched})$

6 Lower Bounds

Our main lower bound for deterministic, oblivious algorithms appears in Section 6.1. Section 7 discusses the extention to deterministic, partially oblivious algorithms.

6.1 Deterministic, Oblivious Algorithms

In this section, we present a lower bound of $\Omega(dn)$ on the convergence complexity of any deterministic, oblivious, bottleneck algorithm that computes the max-min rate vector. The proof uses the network constructed in Section 5.

Theorem 6.1 *Assume that* Alg *is a deterministic, oblivious, bottleneck algorithm that computes the max-min rate vector. Then,*

$$\mathcal{U}_{\mathsf{Alg}} \geq \frac{dn}{4} + \frac{n}{2}.$$

Proof: We will show that

$$\mathcal{U}_{\mathsf{Alg}}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle) = dn/4 + n/2.$$

To do so, we will 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 will add up the result over all clusters. Henceforth, we restrict attention to the execution α of Alg on the network $G_j(\mathsf{Sched} \mid \mathcal{S}_j)$; for simplicity, however, we shall omit, throughout our analysis, the cluster index j.

Define the sequence of states $Q^{(0)}, Q^{(1)}, \ldots$ as follows. $Q^{(0)} = Q_0$; for each index $r \geq 1$ such that $l_r < \infty$, $Q^{(r)} = Q_{l_r}$. Roughly speaking, $Q^{(r)}$ is the last state in execution epoch $\alpha^{(l_r)}$ of α. For each index $r \geq 1$ such that $l_r < \infty$, define $Q_f^{(r)}$ to be the earliest state in execution epoch $\alpha^{(l_r)}$ such that the session scheduled in front of $Q_f^{(r)}$ is an active session in state $\overrightarrow{Q^{(r-1)}}$. Note that the construction of the network $G(\mathsf{Sched})$ implies that for each execution epoch $\alpha^{(l_r)}$, the session $i_f^{(r)}$ is scheduled in front of the state $Q_f^{(r)}$.

The backbone of our proof is the following technical lemma.

Lemma 6.2 *For each integer r, $1 \leq r \leq d/2$, the following hold:*

(1) $l_r < \infty$;

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

(a) *if $i \in \{i_f^{(r)}, i_l^{(r)}\}$, then $r_{Q^{(r)}}(S_i) = b_r + p_r/4$;*

(b) *if* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, *then* $r_{Q^{(r)}}(S_i) = b_r + p_r/2$;

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

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

(a) $residual_{Q^{(r)}}(e_i^{(r')}) \geq p_r/4$;

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

Lemma 6.2 proves that state $Q^{\overrightarrow{(r-1)}}$ is defined for any index r such that $1 \leq r \leq d/2$, and deals with sessions active in state $Q^{\overrightarrow{(r-1)}}$. Roughly speaking, condition (1) asserts that all such sessions are scheduled for an increase in execution epoch $\alpha^{(l_r)}$, while condition (2) specifies their rates upon completion of execution epoch $\alpha^{(l_r)}$. Moreover, condition (3) determines which of these sessions are terminated upon completion of execution epoch $\alpha^{(l_r)}$, while condition (4) provides lower bounds on the residual capacity and fair share of edges related to all these sessions, except for the two distinguished ones $i_f^{(r)}$ and $i_l^{(r)}$, which become done, when execution epoch $\alpha^{(l_r)}$ is completed. We continue with the proof of Lemma 6.2.

Proof: By induction on r. We start with the basis case where $r = 1$.

Basis Case (Lemma 6.2)

We start by proving (1). Assume, by the way of contradiction, that $l_1 = \infty$. By construction of network $G(\mathsf{Sched})$, for each edge $e \in E(\mathsf{Sched})$, $cap(e) > 0$. By definition of l_1, there exists some session $i \in \mathcal{A}_{Q_0} = \mathcal{S}$ such that i does not appear in α. Thus, session S_i is not scheduled for an increase in α. By definition of update operation, this implies that the final rate of S_i remains zero in α, which contradicts max-min fairness. It follows that $l_1 < \infty$, as needed.

We continue to prove the rest of the conditions. We start with a technical claim that describes the state $Q_f^{(1)}$.

Lemma 6.3 *The following hold for the state* $Q_f^{(1)}$:

(1) $r_{Q_f^{(1)}}(S_{i_f^{(1)}}) = 2b_1 + p_1/2$;

(2) *for each session* $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}\}$, $r_{Q_f^{(1)}}(S_i) = 0$;

(3) $FS_{Q_f^{(1)}}(e^{(1)}_{i_l^{(1)}}) = b_1 + p_1/4$;

(4) *for each session* $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, $FS_{Q_f^{(1)}}(e_i^{(1)}) = b_1 + p_1/2$;

(5) $e^{(1)}_{i_l^{(1)}}$ *is a bottleneck edge for* $Q_f^{(1)}$;

(6) *for each session* $i \in \mathcal{A}_{Q_0}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_f^{(1)})$.

Roughly speaking, conditions (1) and (2) determine rates of active sessions in state $Q_f^{(1)}$, while conditions (3) and (4) determine the corresponding fair shares of edges in epoch 1 that are traversed by these sessions. Condition (5) establishes that $e^{(1)}_{i_f^{(1)}}$ is a bottleneck edge for $Q_f^{(1)}$, while condition (6) guarantees that no corresponding termination of sessions takes place. Note that since all sessions are active in the initial state Q_0, the first session to be scheduled is active, so that $Q_f^{(1)} = Q_1$. We continue with the proof of Lemma 6.3.

Proof: We start by proving (1). By construction of the network $G(\mathsf{Sched})$, session $i_f^{(1)}$ traverses edge $e^{(1)}_{i_l^{(1)}}$, and for each index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, it traverses edge $e_i^{(1)}$; moreover, by Observation 5.1, $i_f^{(1)}$ traverses no other edge of the network. We proceed to calculate the increases for $i_f^{(1)}$ in state Q_0 that are imposed by the edges it traverses.

Clearly,

$$\begin{aligned}\Delta_{Q_0}(i_f^{(1)}, e^{(1)}_{i_l^{(1)}}) &= cap(e^{(1)}_{i_l^{(1)}}) \\ &= 2b_1 + p_1/2\,,\end{aligned}$$

while for any session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$,

$$\begin{aligned}\Delta_{Q_0}(i_f^{(1)}, e_i^{(1)}) &= cap(e_i^{(1)}) \\ &= 2b_1 + p_1\,.\end{aligned}$$

Thus, by definition of $\Delta_{Q_0}(i_f^{(1)})$, $\Delta_{Q_0}(i_f^{(1)}) = \min\{2b_1 + p_1/2, 2b_1 + p_1\} = 2b_1 + p_1/2$. Hence, by definition of update operation, $r_{Q_f^{(1)}}(i_f^{(1)}) = r_{Q_0}(i_f^{(1)}) + \Delta_{Q_0}(i_f^{(1)}) = 0 + 2b_1 + p_1/2 = 2b_1 + p_1/2$. This completes the proof of (1).

To prove (2), take any session $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}\}$. By definition of update operation, $r_{Q_f^{(1)}}(i) = \min\{r_{Q_0}(i), r_{Q_f^{(1)}}(i_f^{(1)})\} = \min\{0, 2b_1 + p_1/2\} = 0$, as needed.

We continue to prove (3). Clearly,

$$\begin{aligned}
FS_{Q_f^{(1)}}(e^{(1)}_{i_l^{(1)}}) &= \frac{cap(e^{(1)}_{i_l^{(1)}}) - allot_{Q_f^{(1)}}(e^{(1)}_{i_l^{(1)}})}{|\mathcal{A}_{Q_f^{(1)}} \mid e^{(1)}_{i_l^{(1)}}|} \\
&= \frac{cap(e^{(1)}_{i_l^{(1)}})}{|\mathcal{A}_{Q_0} \mid e^{(1)}_{i_l^{(1)}}|} && \text{(since } \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_0) = \emptyset) \\
&= \frac{2b_1 + p_1/2}{2} && \text{(by construction of } G(\mathsf{Sched})) \\
&= b_1 + \frac{p_1}{4}\,,
\end{aligned}$$

which proves (3).

We continue to prove (4). Take any session $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$. Clearly,

$$\begin{aligned}
FS_{Q_f^{(1)}}(e_i^{(1)}) &= \frac{cap(e_i^{(1)}) - allot_{Q_f^{(1)}}(e_i^{(1)})}{|\mathcal{A}_{Q_f^{(1)}} \mid e_i^{(1)}|} \\
&= \frac{cap(e_i^{(1)})}{|\mathcal{A}_{Q_0} \mid e_i^{(1)}|} && \text{(since } \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_0) = \emptyset) \\
&= \frac{2b_1 + p_1}{2} && \text{(by construction of } G(\mathsf{Sched})) \\
&= b_1 + \frac{p_1}{2}\,,
\end{aligned}$$

which proves (4).

We continue to prove (5). By construction of $G(\mathsf{Sched})$, there are two sessions traversing edge $e^{(1)}_{i_l^{(1)}}$, namely $i_f^{(1)}$ and $i_l^{(1)}$. Take first session $i_f^{(1)}$. By construction of $G(\mathsf{Sched})$, besides edge $e^{(1)}_{i_l^{(1)}}$, $i_f^{(1)}$ traverses, for each index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, edge $e_i^{(1)}$. Clearly,

$$\begin{aligned}
MFS_{Q_f^{(1)}}(S_{i_f^{(1)}}) &= \min\{FS_{Q_f^{(1)}}(e^{(1)}_{i_l^{(1)}}), \min \bigcup_{i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}} \{FS_{Q_f^{(1)}}(e_i^{(1)})\}\} \\
&= \min\{b_1 + \frac{p_1}{4}, b_1 + \frac{p_1}{2}\} \\
&\quad \text{(by (3) and (4))} \\
&= b_1 + \frac{p_1}{4}\,,
\end{aligned}$$

so that $MFS_{Q_f^{(1)}}(S_{i_f^{(1)}}) = FS_{Q_f^{(1)}}(e^{(1)}_{i_l^{(1)}})$.

Take now session $i_l^{(1)}$. By construction of $G(\mathsf{Sched})$, $i_l^{(1)}$ traverses only edge $e_{i_l^{(1)}}^{(1)}$, so that $MFS_{Q_f^{(1)}}(S_{i_l^{(1)}}) = FS_{Q_f^{(1)}}(e_{i_l^{(1)}}^{(1)})$.

Since $MFS_{Q_f^{(1)}}(S_{i_f^{(1)}}) = FS_{Q_f^{(1)}}(e_{i_l^{(1)}}^{(1)})$ and $MFS_{Q_f^{(1)}}(S_{i_l^{(1)}}) = FS_{Q_f^{(1)}}(e_{i_l^{(1)}}^{(1)})$, it follows that $e_{i_l^{(1)}}^{(1)}$ is a bottleneck edge for state $Q_f^{(1)}$, which completes the proof of (5).

We continue to prove (6). We check first that no edge traversed by session $i_f^{(1)}$ causes its termination in state $Q_f^{(1)}$.

Take first edge $e_{i_l^{(1)}}^{(1)}$. By (1), $r_{Q_f^{(1)}}(i_f^{(1)}) = 2b_1 + p_1/2$, while by (3), $FS_{Q_f^{(1)}}(e_{i_l^{(1)}}^{(1)}) = b_1 + p_1/4$. Thus, $r_{Q_f^{(1)}}(i_f^{(1)}) - FS_{Q_f^{(1)}}(e_{i_l^{(1)}}^{(1)}) = b_1 + p_1/2$. Since both $b_1 \neq 0$ and $p_1 \neq 0$, it follows that $r_{Q_f^{(1)}}(i_f^{(1)}) \neq FS_{Q_f^{(1)}}(e_{i_l^{(1)}}^{(1)})$. Since Alg is bottleneck, this implies that $e_{i_l^{(1)}}^{(1)}$ does not cause the termination of $i_f^{(1)}$ in state $Q_f^{(1)}$.

Take now edge $e_i^{(1)}$, for any session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$. Recall that $MFS_{Q_f^{(1)}}(S_{i_f^{(1)}}) = b_1 + p_1/4$, while by (4), $FS_{Q_f^{(1)}}(e_i^{(1)}) = b_1 + p_1/2$. Since $p_1 \neq 0$, it follows that $MFS_{Q_f^{(1)}}(S_{i_f^{(1)}}) \neq FS_{Q_f^{(1)}}(e_i^{(1)})$. By definition of bottleneck edge, this implies that $e_i^{(1)}$ is not a bottleneck edge for state $Q_f^{(1)}$. Since Alg is bottleneck, this implies that $e_i^{(1)}$ does not cause the termination of session $i_f^{(1)}$ in state $Q_f^{(1)}$.

However, by Observation 5.1, session $i_f^{(1)}$ traverses no other edge. Hence, it follows that $i_f^{(1)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q_f^{(1)})$.

By (2), rates of all other sessions remain 0 in state $Q_f^{(1)}$. Hence, none of the remaining sessions is terminated in state $Q_f^{(1)}$ either, since it would otherwise retain its initial zero rate, contradicting max-min fairness. This completes the proof of (6). ■

We return to the proof of Lemma 6.2.

Define $Q_{ff}^{(1)}$ to be the earliest state such that a session active in the initial state Q_0 but different from $i_f^{(1)}$ is scheduled for an increase in front of state $Q_{ff}^{(1)}$ in execution α. Note that since $l_1 < \infty$, $Q_{ff}^{(1)}$ precedes $Q^{(1)}$ in the execution epoch $\alpha^{(l_1)}$; moreover, $Q_f^{(1)}$ precedes $Q_{ff}^{(1)}$ in the execution epoch $\alpha^{(l_1)}$. The next simple claim shows that, following state $Q_f^{(1)}$ in the execution α, no change in the rate of status of any session is possible prior to reaching state $Q_{ff}^{(1)}$.

Lemma 6.4 *Consider any session S_i. Then, for any state Q such that $Q_f^{(1)} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q_{ff}^{(1)}$, $r_Q(S_i) = r_{Q_f^{(1)}}(S_i)$, and $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q)$.*

Proof: By definition of $Q_{ff}^{(1)}$, the session that is scheduled in front of state Q is $S_{i_f^{(1)}}$. We proceed by induction on Q.

We start with the basis case where $Q = \overrightarrow{Q_f^{(1)}}$. By Lemma 6.3(1), $r_{Q_f^{(1)}}(S_{i_f^{(1)}}) = 2b_1 + p_1/2$. Moreover, by construction of $G(\mathsf{Sched})$, $cap(e_{i_l^{(1)}}^{(1)}) = 2b_1 + p_1/2$, it follows by definition of update function (conditions (1) and (2)), that $\Delta_{Q_f^{(1)}}(i_f^{(1)}, e_{i_l^{(1)}}^{(1)}) = 0$, so that $\Delta_{Q_f^{(1)}}(i_f^{(1)}) = 0$. Thus, by definition of update operation, for each session $i \in \mathcal{A}_{Q_f^{(1)}}$, $r_Q(S_i) = r_{Q_f^{(1)}}(S_i)$, as needed. Moreover, since Alg is bottleneck and the termination condition for bottleneck algorithms is a condition that depends solely on session rates, Lemma 6.3(6) immediately implies that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q)$, as needed.

Assume inductively that for any state Q such that $Q_f^{(1)} \xrightarrow{\alpha} Q \xrightarrow{\alpha} \overleftarrow{Q_{ff}^{(1)}}$, $r_Q(S_i) = r_{Q_f^{(1)}}(S_i)$, and $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q)$. For the induction step, consider state $\overrightarrow{Q}$.

By induction hypothesis and Lemma 6.3(1), $r_Q(S_{i_f^{(1)}}) = r_{Q_f^{(1)}}(S_{i_f^{(1)}}) = 2b_1 + p_1/2$. Moreover, by construction of $G(\mathsf{Sched})$, $cap(e_{i_l^{(1)}}^{(1)}) = 2b_1 + p_1/2$, it follows by definition of update function (conditions (1) and (2)), that $\Delta_Q(i_f^{(1)}, e_{i_l^{(1)}}^{(1)}) = 0$, so that $\Delta_Q(i_f^{(1)}) = 0$. Thus, by definition of update operation, for each session $i \in \mathcal{A}_Q$, $r_{\overrightarrow{Q}}(S_i) = r_Q(S_i)$. However, by induction hypothesis, $\mathcal{A}_Q = \mathcal{A}_{Q_f^{(1)}}$ and $r_Q(S_i) = r_{Q_f^{(1)}}(S_i)$. It follows that for each session $i \in \mathcal{A}_{Q_f^{(1)}}$, $r_{\overrightarrow{Q}}(S_i) = r_{Q_f^{(1)}}(S_i)$, as needed. Moreover, since Alg is bottleneck and the termination condition for bottleneck algorithms is a condition that depends solely on session rates, Lemma 6.3(6) immediately implies that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, \overrightarrow{Q})$, which completes the proof of the induction step. ■

Roughly speaking, the next technical claim describes the "fragment" of the execution epoch $\alpha^{(l_1)}$ starting with scheduling the session right in front of $Q_{ff}^{(1)}$ and ending with the state immediately preceding $Q^{(1)}$, the last state in $\alpha^{(l_1)}$.

Lemma 6.5 *Consider any state Q such that $\overleftarrow{Q_{ff}^{(1)}} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q^{(1)}$. Then, the following hold for state Q:*

(1) *for each session $i \in \mathcal{S} \mid_\alpha (Q_0, Q)$, $r_Q(S_i) = b_1 + p_1/2$;*

(2) *for each session* $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q)$, $r_Q(S_i) = 0$;

(3) *edge* $e^{(1)}_{i^{(1)}_l}$ *is a bottleneck edge for* Q;

(4) *for any session* $i \in \mathcal{A}_{Q_0} \setminus \{i^{(1)}_f, i^{(1)}_l\}$, *edge* $e^{(1)}_i$ *is not a bottleneck edge for* Q;

(5) *for any integer* $r' > 1$ *and edge* $e^{(r')} \in E^{(r')}$, $e^{(r')}$ *is not a bottleneck edge for* Q;

(6) *for any session* $i \in \mathcal{A}_{Q_0}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$.

Roughly speaking, conditions (1) and (2) determine rates of active sessions in state Q. Conditions (3), (4) and (5) establish that edge $e^{(1)}_{i^{(1)}_l}$ is the only bottleneck edge for state Q, while condition (6) guarantees that no corresponding termination of sessions takes place. We continue with the proof of Lemma 6.5.

Proof: By induction on Q. We start with the basis case where $Q = Q^{(1)}_{ff}$.

Basis Case (Lemma 6.5)

We start by determining the rates and status of all sessions in state $Q^{(1)}_{ff}$.

Denote $i^{(1)}_{ff}$ the session that is scheduled in front of state $Q^{(1)}_{ff}$. By definition of state $Q^{(1)}_{ff}$, $i^{(1)}_{ff} \neq i^{(1)}_f$, while by definition of state $Q^{(1)}$, $i^{(1)}_{ff} \neq i^{(1)}_l$.

By construction of $G(\mathsf{Sched})$, $i^{(1)}_{ff}$ traverses the epoch one edge $e^{(1)}_{i^{(1)}_{ff}}$, which is also traversed by session $i^{(1)}_f$. By construction of $G(\mathsf{Sched})$, $cap(e^{(1)}_{i^{(1)}_{ff}}) = 2b_1 + p_1$. By Lemma 6.4, $r^{\leftarrow}_{Q^{(1)}_{ff}}(S_{i^{(1)}_f}) = r_{Q^{(1)}_f}(S_{i^{(1)}_f})$ and $r^{\leftarrow}_{Q^{(1)}_{ff}}(S_{i^{(1)}_{ff}}) = r_{Q^{(1)}_f}(S_{i^{(1)}_{ff}})$. Lemma 6.3(1) and (2) imply that $r^{\leftarrow}_{Q^{(1)}_{ff}}(S_{i^{(1)}_f}) = 2b_1 + p_1/2$ and $r^{\leftarrow}_{Q^{(1)}_{ff}}(S_{i^{(1)}_{ff}}) = 0$. Thus, by definition of update operation,

$$\begin{aligned}
\Delta^{\leftarrow}_{Q^{(1)}_{ff}}(i^{(1)}_{ff}, e^{(1)}_{i^{(1)}_{ff}}) &= \frac{1}{2}\, cap(e_{i^{(1)}_{ff}}) - r^{\leftarrow}_{Q^{(1)}_{ff}}(S_{i^{(1)}_{ff}}) \\
&= \frac{1}{2}\, cap(e_{i^{(1)}_{ff}}) - 0 \\
&= \frac{1}{2}\,(2\,b_1 + p_1) \qquad \text{(by construction of } G(\mathsf{Sched})) \\
&= b_1 + \frac{p_1}{2}\,.
\end{aligned}$$

By construction of $G(\mathsf{Sched})$, $S_{i_{ff}^{(1)}}$ traverses also at some epoch r' edge $e^{(r')} \in E^{(r')}$, for any integer $r' > 1$. Moreover, by construction of $G(\mathsf{Sched})$, edge $e^{(r')}$ is also some other session $i \notin \{i_f^{(1)}, i_l^{(1)}\}$. By Lemma 6.4, $r_{\overleftarrow{Q_{ff}^{(1)}}}(S_i) = r_{Q_f^{(1)}}(S_i)$, while by Lemma 6.3(2), $r_{Q_f^{(1)}}(S_i) = 0$. It follows that $r_{\overleftarrow{Q_{ff}^{(1)}}}(S_i) = 0$. Hence, by definition of increase for S_i in state $\overleftarrow{Q_{ff}^{(1)}}$ imposed by edge $e^{(r')}$ and the update function, $\Delta_{\overleftarrow{Q_{ff}^{(1)}}}(i_{ff}^{(1)}, e^{(r')}) = cap(e^{(r')}) - r_{\overleftarrow{Q_{ff}^{(1)}}}(S_{i_{ff}^{(1)}}) = cap(e^{(r')})$. By Proposition 5.3, $cap(e^{(r')}) > cap(e_{i_{ff}^{(1)}}^{(1)}) = 2b_1 + p_1$. Thus, $\Delta_{\overleftarrow{Q_{ff}^{(1)}}}(i_{ff}^{(1)}, e^{(r')}) > 2b_1 + p_1$. It follows that

$$\begin{aligned}
\Delta_{\overleftarrow{Q_{ff}^{(1)}}}(i_{ff}^{(1)}, e^{(r')}) - \Delta_{\overleftarrow{Q_{ff}^{(1)}}}(i_{ff}^{(1)}, e_{i_{ff}^{(1)}}^{(1)}) &> 2b_1 + p_1 - (b_1 + \frac{p_1}{2}) \\
&= b_1 + \frac{p_1}{2} \\
&= 2^{p-1} && \text{(by definition of } b_1 \text{ and } p_1) \\
&> 0\,.
\end{aligned}$$

Hence, $\Delta_{\overleftarrow{Q_{ff}^{(1)}}}(i_{ff}^{(1)}) = \Delta_{\overleftarrow{Q_{ff}^{(1)}}}(i_{ff}^{(1)}, e_{i_{ff}^{(1)}}^{(1)}) = b_1 + p_1/2$. So, by definition of update operation,

$$\begin{aligned}
r_{Q_{ff}^{(1)}}(S_{i_{ff}^{(1)}}) &= r_{\overleftarrow{Q_{ff}^{(1)}}}(S_{i_{ff}^{(1)}}) + \Delta_{\overleftarrow{Q_{ff}^{(1)}}}(i_{ff}^{(1)}) \\
&= 0 + b_1 + \frac{p_1}{2} \\
&= b_1 + \frac{p_1}{2}\,.
\end{aligned}$$

Moreover, by definition of update operation,

$$r_{\overleftarrow{Q_{ff}^{(1)}}}(S_{i_f^{(1)}}) = r_{Q_{ff}^{(1)}}(S_{i_{ff}^{(1)}})\,,$$

so that,

$$r_{Q_{ff}^{(1)}}(S_{i_f^{(1)}}) = b_1 + \frac{p_1}{2}\,.$$

Clearly, by definition of states $Q_f^{(1)}$ and $Q_{ff}^{(1)}$, $\mathcal{S}\mid_\alpha (Q_f^{(1)}, Q_{ff}^{(1)}) = \{i_f^{(1)}, i_{ff}^{(1)}\}$, so that for each session $i \in \mathcal{S}\mid_\alpha (Q_f^{(1)}, Q_{ff}^{(1)})$, $r_{Q_{ff}^{(1)}}(S_i) = b_1 + p_1/2$, as needed for (1).

We proceed to show (2). Consider any session $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S}\mid_\alpha (Q_f^{(1)}, Q_{ff}^{(1)}) = \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_{ff}^{(1)}\}$. Clearly,

$$r_{\overleftarrow{Q_{ff}^{(1)}}}(S_i) = r_{Q_f^{(1)}}(S_i) \qquad \text{(by Lemma 6.4)}$$

$$= \quad 0 \qquad \text{(by Lemma 6.3)}.$$

Since $i \neq i_{ff}^{(1)}$, the definition of update operation immediately implies that $r_{Q_{ff}^{(1)}}(S_i) = 0$, as needed for (2).

We proceed to prove (3).

Claim 6.6 *$e_{i_l^{(1)}}^{(1)}$ is a bottleneck edge for state $Q_{ff}^{(1)}$.*

Proof: Lemma 6.3(6) implies that $\mathcal{A}_{\underset{Q_f^{(1)}}{\rightarrow}} = \mathcal{A}_{Q_0}$. By Lemma 6.4, for any state Q such that $Q_f^{(1)} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q_{ff}^{(1)}$, for each session $i \in \mathcal{A}_{Q_0}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$. It follows that $\mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{\underset{Q_f^{(1)}}{\rightarrow}} = \mathcal{A}_{Q_0}$. Thus, $i_l^{(1)} \in \mathcal{A}_{Q_{ff}^{(1)}}$. Since by construction of $G(\mathsf{Sched})$, $i_l^{(1)}$ traverses edge $e_{i_l^{(1)}}^{(1)}$, it follows that $e_{i_l^{(1)}} \in AE_{Q_{ff}^{(1)}}$. However, by Lemma 6.3, $e_{i_l^{(1)}}^{(1)}$ is a bottleneck edge for state $Q_f^{(1)}$. Hence, Proposition 4.1(2) implies that $e_{i_l^{(1)}}^{(1)}$ is a bottleneck edge for state $Q_{ff}^{(1)}$, as needed. ■

We continue to prove (4).

Claim 6.7 *For any session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, edge $e_i^{(1)}$ is not a bottleneck edge for state $Q_{ff}^{(1)}$.*

Proof: Lemma 6.3(6) implies that $\mathcal{A}_{\underset{Q_f^{(1)}}{\rightarrow}} = \mathcal{A}_{Q_0}$. By Lemma 6.4, for any state Q such that $Q_f^{(1)} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q_{ff}^{(1)}$, for each session $i \in \mathcal{A}_{Q_0}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$. It follows that $\mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{\underset{Q_f^{(1)}}{\rightarrow}} = \mathcal{A}_{Q_0}$. Hence, both $i \in \mathcal{A}_{Q_{ff}^{(1)}}$ and $i_f^{(1)} \in \mathcal{A}_{Q_{ff}^{(1)}}$. By construction of $G(\mathsf{Sched})$, edge $e_i^{(1)}$ is traversed by sessions S_i and $S_{i_f^{(1)}}$. It follows that $\mathcal{A}_{Q_{ff}^{(1)}} \mid e_i^{(1)} = \{i, i_f^{(1)}\}$, so that

$$\begin{aligned}
FS_{Q_{ff}^{(1)}}(e_i^{(1)}) &= \frac{cap(e_i^{(1)}) - allot_{Q_{ff}^{(1)}}(e_i^{(1)})}{|\mathcal{A}_{Q_{ff}^{(1)}} \mid e_i^{(1)}|} \\
&= \frac{cap(e_i^{(1)}) - 0}{2} && \text{(since } \mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{Q_0}\text{)} \\
&= b_1 + \frac{p_1}{2} && \text{(by construction of } G(\mathsf{Sched})\text{)}.
\end{aligned}$$

Since $\mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{Q_0}$, both $i_f^{(1)} \in \mathcal{A}_{Q_{ff}^{(1)}}$ and $i_l^{(1)} \in \mathcal{A}_{Q_{ff}^{(1)}}$. By construction of $G(\mathsf{Sched})$, edge $e_{i_l^{(1)}}^{(1)}$ is traversed by sessions $S_{i_f^{(1)}}$ and $S_{i_l^{(1)}}$. It follows that $\mathcal{A}_{Q_{ff}^{(1)}} \mid e_{i_l^{(1)}}^{(1)} = \{i_f^{(1)}, i_l^{(1)}\}$, so that

$$\begin{aligned} FS_{Q_{ff}^{(1)}}(e_{i_l^{(1)}}^{(1)}) &= \frac{cap(e_{i_l^{(1)}}^{(1)}) - allot_{Q_{ff}^{(1)}}(e_{i_l^{(1)}}^{(1)})}{|\mathcal{A}_{Q_{ff}^{(1)}} \mid e_{i_l^{(1)}}^{(1)}|} \\ &= \frac{cap(e_{i_l^{(1)}}^{(1)}) - 0}{2} && \text{(since } \mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{Q_0}\text{)} \\ &= b_1 + \frac{p_1}{4} && \text{(by construction of } G(\mathsf{Sched})\text{)}. \end{aligned}$$

By construction of network $G(\mathsf{Sched})$, session $i_f^{(1)}$ traverses every edge $e_i^{(1)}$, for each session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, and no other (higher epoch) edge. Thus,

$$\begin{aligned} MFS_{Q_{ff}^{(1)}}(S_{i_f^{(1)}}) &= \min\{FS_{Q_{ff}^{(1)}}(e_{i_l^{(1)}}^{(1)}), \bigcup_{i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}} FS_{Q_{ff}^{(1)}}(e_i^{(1)})\} \\ &= \min\{b_1 + \frac{p_1}{4}, b_1 + \frac{p_1}{2}\} \\ &= b_1 + \frac{p_1}{4}. \end{aligned}$$

Hence, $S_{i_f^{(1)}} \in \mathcal{A}_{Q_{ff}^{(1)}} \mid e_i^{(1)}$ while

$$\begin{aligned} FS_{Q_{ff}^{(1)}}(e_i^{(1)}) - MFS_{Q_{ff}^{(1)}}(S_{i_f^{(1)}}) &= b_1 + \frac{p_1}{2} - (b_1 + \frac{p_1}{4}) \\ &= \frac{p_1}{4} \\ &= 2^{p-2} && \text{(by definition of } p_1\text{)} \\ &\neq 0, \end{aligned}$$

which implies that $e_i^{(1)}$ is not a bottleneck edge for state $Q_{ff}^{(1)}$, as needed. ■

We continue to prove (5).

Claim 6.8 *For any integer $r' > 1$ and edge $e^{(r')} \in E^{(r')}$, $e^{(r')}$ is not a bottleneck edge for state $Q_{ff}^{(1)}$.*

Proof: By construction of network $G(\mathsf{Sched})$, edge $e^{(r')}$ is traversed by some session $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, which also traverses edge $e_i^{(1)}$. We proceed to calculate the fair shares of edges $e_i^{(1)}$ and $e^{(r')}$.

Lemma 6.3(6) implies that $\mathcal{A}_{\underset{Q_f^{(1)}}{\rightarrow}} = \mathcal{A}_{Q_0}$. By Lemma 6.4, for any state Q such that $Q_f^{(1)} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q_{ff}^{(1)}$, for each session $i \in \mathcal{A}_{Q_0}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$. It follows that $\mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{\underset{Q_f^{(1)}}{\rightarrow}} = \mathcal{A}_{Q_0}$. Hence, both $i \in \mathcal{A}_{Q_{ff}^{(1)}}$ and $i_f^{(1)} \in \mathcal{A}_{Q_{ff}^{(1)}}$. By construction of $G(\mathsf{Sched})$, edge $e_i^{(1)}$ is traversed by sessions S_i and $S_{i_f^{(1)}}$. It follows that $\mathcal{A}_{Q_{ff}^{(1)}} \mid e_i^{(1)} = \{i, i_f^{(1)}\}$, so that

$$\begin{aligned} FS_{Q_{ff}^{(1)}}(e_i^{(1)}) &= \frac{cap(e_i^{(1)}) - allot_{Q_{ff}^{(1)}}(e_i^{(1)})}{|\mathcal{A}_{Q_{ff}^{(1)}} \mid e_i^{(1)}|} \\ &= \frac{cap(e_i^{(1)}) - 0}{2} \qquad \text{(since } \mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{Q_0}\text{)} \\ &= \frac{cap(e_i^{(1)})}{2}. \end{aligned}$$

By construction of $G(\mathsf{Sched})$, edge $e^{(r')}$ is traversed by sessions S_i and $S_{i'}$, for some session index i'. Clearly, both $i \in \mathcal{A}_{Q_{ff}^{(1)}}$ and $i' \in \mathcal{A}_{Q_{ff}^{(1)}}$, so that $\mathcal{A}_{Q_{ff}^{(1)}} \mid e^{(r')} = \{i, i'\}$. Thus,

$$\begin{aligned} FS_{Q_{ff}^{(1)}}(e') &= \frac{cap(e') - allot_{Q_{ff}^{(1)}}(e')}{|\mathcal{A}_{Q_{ff}^{(1)}} \mid e^{(l)}|} \\ &= \frac{cap(e' - 0}{2} \qquad \text{(since } \mathcal{A}_{Q_{ff}^{(1)}} = \mathcal{A}_{Q_0}\text{)} \\ &= \frac{cap(e')}{2}. \end{aligned}$$

By Proposition 5.3, $cap(e_i^{(1)}) < cap(e^{(r')})$. It follows that $FS_{Q_{ff}^{(1)}}(e_i^{(1)}) < FS_{Q_{ff}^{(1)}}(e^{(r')})$. By definition of minimum fair share, this implies that $MFS_{Q_{ff}^{(1)}}(S_i) < FS_{Q_{ff}^{(1)}}(e^{(r')})$, so that edge $e^{(r')}$ is not a bottleneck edge for state $Q_{ff}^{(1)}$, as needed. ■

We continue to prove (6). We consider separately each of the sessions active in Q_0. We start with session $S_{i_f^{(1)}}$.

Claim 6.9 $i_f^{(1)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_{ff}^{(1)})$.

Proof: By construction of $G(\mathsf{Sched})$, $i_f^{(1)}$ traverses edge $e_{i_l^{(1)}}^{(1)}$; moreover, for each session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, $i_f^{(1)}$ traverses edge $e_i^{(1)}$ and no other edge. We consider separately each of these edges.

Take first edge $e^{(1)}_{i^{(1)}_l}$. Since $\mathcal{A}_{Q^{(1)}_{ff}} = \mathcal{A}_{Q_0}$, both $i^{(1)}_f \in \mathcal{A}_{Q^{(1)}_{ff}}$ and $i^{(1)}_l \in \mathcal{A}_{Q^{(1)}_{ff}}$. By construction of $G(\mathsf{Sched})$, edge $e^{(1)}_{i^{(1)}_l}$ is traversed by sessions $S_{i^{(1)}_f}$ and $S_{i^{(1)}_l}$. It follows that $\mathcal{A}_{Q^{(1)}_{ff}} \mid e^{(1)}_{i^{(1)}_l} = \{i^{(1)}_f, i^{(1)}_l\}$, so that

$$\begin{aligned} FS_{Q^{(1)}_{ff}}(e^{(1)}_{i^{(1)}_l}) &= \frac{cap(e^{(1)}_{i^{(1)}_l}) - allot_{Q^{(1)}_{ff}}(e^{(1)}_{i^{(1)}_l})}{|\mathcal{A}_{Q^{(1)}_{ff}} \mid e^{(1)}_{i^{(1)}_l}|} \\ &= \frac{cap(e^{(1)}_{i^{(1)}_l}) - 0}{2} && \text{(since } \mathcal{A}_{Q^{(1)}_{ff}} = \mathcal{A}_{Q_0}\text{)} \\ &= b_1 + \frac{p_1}{4} && \text{(by construction of } G(\mathsf{Sched})\text{)}. \end{aligned}$$

However, by (1), $r_{Q^{(1)}_{ff}}(S_{i^{(1)}_f}) = b_1 + p_1/2$, so that

$$\begin{aligned} r_{Q^{(1)}_{ff}}(S_{i^{(1)}_f}) - FS_{Q^{(1)}_{ff}}(e^{(1)}_{i^{(1)}_l}) &= b_1 + \frac{p_1}{2} - (b_1 + \frac{p_1}{4}) \\ &= \frac{p_1}{4} \\ &= 2^{p-2} && \text{(by definition of } p_1\text{)} \\ &\neq 0\,. \end{aligned}$$

Since Alg is bottleneck, it follows that edge $e^{(1)}_{i^{(1)}_l}$ does not cause the termination of session $S_{i^{(1)}_f}$ in state $Q^{(1)}_{ff}$.

Take now edge $e^{(1)}_i$ for any session index $i \in \mathcal{A}_{Q_0} \setminus \{i^{(1)}_f, i^{(1)}_l\}$. By Claim 6.7, $e^{(1)}_i$ is not a bottleneck edge for state $Q^{(1)}_{ff}$. Since Alg is bottleneck, it follows that edge $e^{(1)}_i$ does not cause the termination of session $S_{i^{(1)}_f}$ in state $Q^{(1)}_{ff}$.

Since none of the edges traversed by session $S_{i^{(1)}_f}$ causes its termination in state $Q^{(1)}_{ff}$, it follows that $i^{(1)}_f \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(1)}_{ff})$, as needed. ■

We continue with session $S_{i^{(1)}_l}$.

By definition of session $S_{i^{(1)}_l}$, $i^{(1)}_l \notin \mathcal{S} \mid_\alpha (Q^{(1)}_f, Q^{(1)}_{ff})$, so that by (2), $r_{Q^{(1)}_{ff}}(S_{i^{(1)}_l}) = 0$. Assume, by the way of contradiction, that $i^{(1)}_l \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(1)}_{ff})$. By definition of update operation, this implies that the final rate of $S_{i^{(!)}_l}$ remains zero in α, which contradicts max-min fairness. Hence, $i^{(1)}_l \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(1)}_{ff})$, as needed.

We finally consider all remaining sessions. Fix any session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$. By construction of $G(\mathsf{Sched})$, S_i traverses edge $e_i^{(1)}$ and any edge $e^{(r')} \in E^{(r')}$, for some integer $r' > 1$. By Claims 6.7 and 6.8, respectively, neither edge $e_i^{(1)}$ nor edge $e^{(r')}$ is a bottleneck edge for state $Q_{ff}^{(1)}$. Since Alg is bottleneck, this implies that neither edge $e_i^{(1)}$ nor edge $e^{(r')}$ causes the termination of session S_i in state $Q_{ff}^{(1)}$. Since none of the edges traversed by session S_i causes its termination in state $Q_{ff}^{(1)}$, it follows that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q_{ff}^{(1)})$.

The proof of the basis case of Lemma 6.5 is now complete.

Induction Hypothesis (Lemma 6.5)

Assume inductively that for some state Q such that $Q_{ff}^{(1)} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q^{(1)}$, for each state Q' such that $Q_{ff}^{(1)} \xrightarrow{\alpha} Q' \xrightarrow{\alpha} Q$, the following hold:

(1) for each session $i \in \mathcal{S} \mid_\alpha (Q_0, Q')$, $r_{Q'}(S_i) = b_1 + p_1/2$;

(2) for each session $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q')$, $r_{Q'}(S_i) = 0$;

(3) edge $e_{i_l^{(1)}}^{(1)}$ is a bottleneck edge for state Q';

(4) for any session $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, edge $e_i^{(1)}$ is not a bottleneck edge for state Q';

(5) for any integer $r' > 1$ and edge $e^{(r')} \in E^{(r')}$, $e^{(r')}$ is not a bottleneck edge for state Q';

(6) for any session $i \in \mathcal{A}_{Q_0}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q')$.

Roughly speaking, we assume that the claim holds for all states starting from state $Q_{ff}^{(1)}$ through but not including state Q, for some distinguished state $Q \neq Q_{ff}^{(1)}$.

Induction Step (Lemma 6.5)

We prove that the claim holds for state Q, namely that:

(1) for each session $i \in \mathcal{S} \mid_\alpha (Q_0, Q)$, $r_Q(S_i) = b_1 + p_1/2$;

(2) for each session $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q)$, $r_Q(S_i) = 0$;

(3) edge $e_{i_l^{(1)}}^{(1)}$ is a bottleneck edge for Q;

(4) for any session $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, edge $e_i^{(1)}$ is not a bottleneck edge for Q;

(5) for any integer $r' > 1$ and edge $e^{(r')} \in E^{(r')}$, $e^{(r')}$ is not a bottleneck edge for Q;

(6) for any session $i \in \mathcal{A}_{Q_0}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$.

We start by proving (1) and (2) simultaneously.

Claim 6.10 *The following hold:*

(1) *For each session* $i \in \mathcal{S} \mid_\alpha (Q_0, Q)$, $r_Q(S_i) = b_1 + p_1/2$;

(2) *for each session* $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q)$, $r_Q(S_i) = 0$.

Proof: Denote i_Q the session scheduled in front of state Q. We proceed by case analysis on i_Q.

1. Assume first that $i_Q = i_f^{(1)}$.

   Since $i_f^{(1)}$ is scheduled in front of state $Q_f^{(1)}$ and $Q_f^{(1)} \xrightarrow{\alpha} Q$, it follows that $\mathcal{S} \mid_\alpha (Q_0, Q) = \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$, so that also $\mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q) = \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$. Consider session $i_{ff}^{(1)}$. By construction of network $G(\mathsf{Sched})$, $i_{ff}^{(1)}$ and $i_f^{(1)}$ share edge $e^{(1)}_{i_{ff}^{(1)}}$ with $cap(e^{(1)}_{i_{ff}^{(1)}}) = 2b_1 + p_1$. Clearly, both $i_{ff}^{(1)} \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$ and $i_f^{(1)} \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$. Thus, by induction hypothesis (1), both $r_Q(S_{i_{ff}^{(1)}}) = b_1 + p_1/2$ and $r_Q(S_{i_f^{(1)}}) = b_1 + p_1/2$, so that $r_Q(S_{i_{ff}^{(1)}}) + r_Q(S_{i_f^{(1)}}) = 2b_1 + p_1 = cap(e^{(1)}_{i_{ff}^{(1)}})$. By definition of update operation, it follows that $\Delta_{\overleftarrow{Q}}(i_{ff}^{(1)}, e^{(1)}_{i_{ff}^{(1)}}) = 0$, so that also $\Delta_{\overleftarrow{Q}}(i_{ff}^{(1)}) = 0$. By definition of update operation, this implies that for each session index $i \in \mathcal{A}_{Q_0}$, $r_Q(S_i) = r_{\overleftarrow{Q}}(S_i)$. Thus, induction hypothesis (1) and (2) imply that for each session index $i \in \mathcal{A}_{Q_0}$, $r_Q(S_i) = b_1 + p_1/2$ if $i \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q}) = \mathcal{S} \mid_\alpha (Q_0, Q)$, or 0 if $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q}) = \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q)$. This establishes (1) and (2).

2. Assume now that $i_Q \neq i_f^{(1)}$.

   Since $Q \xrightarrow{\alpha} Q^{(1)}$ it follows by definition of $Q^{(1)}$ that $i_Q \neq i_l^{(1)}$. There are two cases to consider.

   Assume first that $i_Q \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$ (while still $i_Q \neq i_f^{(1)}$); that is, i_Q has been scheduled for an increase in front of at least one state up to and including state $\overleftarrow{Q}$. It follows that $\mathcal{S} \mid_\alpha (Q_0, Q) = \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$, so that also $\mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q) = \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$.

   By construction of network $G(\mathsf{Sched})$, i_Q and $i_f^{(1)}$ share edge $e^{(1)}_{i_Q}$ with $cap(e^{(1)}_{i_Q}) = 2b_1 + p_1$. Clearly, both $i_Q \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$ and $i_f^{(1)} \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$. Thus, by induction hypothesis

(1), both $r_Q(S_{i_Q}) = b_1 + p_1/2$ and $r_Q(S_{i_f^{(1)}}) = b_1 + p_1/2$, so that $r_Q(S_{i_Q}) + r_Q(S_{i_f^{(1)}}) = 2b_1 + p_1 = cap(e_{i_Q}^{(1)})$. By definition of update operation, it follows that $\Delta_{\overleftarrow{Q}}(i_Q, e_{i_Q}^{(1)}) = 0$, so that also $\Delta_{\overleftarrow{Q}}(i_Q) = 0$. By definition of update operation, this implies that for each session index $i \in \mathcal{A}_{Q_0}$, $r_Q(S_i) = r_{\overleftarrow{Q}}(S_i)$. Thus, induction hypothesis (1) and (2) imply that for each session index $i \in \mathcal{A}_{Q_0}$, $r_Q(S_i) = b_1 + p_1/2$ if $i \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q}) = \mathcal{S} \mid_\alpha (Q_0, Q)$, or 0 if $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q}) = \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q)$. This establishes (1) and (2).

3. Assume finally that $i_Q \notin \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$. By construction of network $G(\mathsf{Sched})$, i_Q and $i_f^{(1)}$ share edge $e_{i_Q}^{(1)}$ with $cap(e_{i_Q}^{(1)}) = 2b_1 + p_1$. Since $i_Q \notin \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$ while $i_f^{(1)} \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$, induction hypothesis (1) and (2) imply that $r_Q(S_{i_Q}) = 0$ and $r_Q(S_{i_f^{(1)}}) = b_1 + p_1/2$, respectively.

   Thus, by definition of iupdate function,

$$\begin{aligned}
\Delta_{\overleftarrow{Q}}(i_Q, e_{i_Q^{(1)}}) &= cap(e_{i_Q}^{(1)}) - r_{\overleftarrow{Q}}(S_{i_f^{(1)}}) - r_{\overleftarrow{Q}}(S_{i_Q}) \\
&= 2b_1 + p_1 - (b_1 + \frac{p_1}{2}) - 0 \\
&= b_1 + \frac{p_1}{2}\,.
\end{aligned}$$

   By construction of network $G(\mathsf{Sched})$, S_{i_Q} traverses also at some epoch r' edge $e^{(r')}$, for any integer $r' > 1$. Moreover, by construction of $G(\mathsf{Sched})$, edge $e^{(r')}$ is also traversed by some other session $i \notin \{i_f^{(1)}, i_l^{(1)}\}$. By induction hypothesis (1) and (2), either $r_{\overleftarrow{Q}}(S_i) = b_1 + p_1/2$ or $r_{\overleftarrow{Q}}(S_i) = 0$. Thus, by definition of update operation,

$$\begin{aligned}
\Delta_{\overleftarrow{Q}}(i_Q, e^{(r')}) &= cap(e^{(r')}) - r_{\overleftarrow{Q}}(S_{i_Q}) - r_{\overleftarrow{Q}}(S_i) \\
&\geq cap(e^{(r')}) - 0 - (b_1 + \frac{p_1}{2}) \\
&\quad \text{(by induction hypothesis (1) and (2))} \\
&> cap(e_{i_Q}^{(1)}) - (b_1 + \frac{p_1}{2}) \\
&\quad \text{(by Proposition 5.3)} \\
&= 2b_1 + p_1 - (b_1 + \frac{p_1}{2}) \\
&\quad \text{(by construction of } G(\mathsf{Sched})) \\
&= b_1 + \frac{p_1}{2}\,.
\end{aligned}$$

   Hence, $\Delta_{\overleftarrow{Q}}(i_Q) = \Delta_{\overleftarrow{Q}}(i_Q, e_{i_Q}^{(1)}) = b_1 + p_1/2$. So, by definition of update operation,

$$r_Q(S_{i_Q}) = r_{\overleftarrow{Q}}(S_{i_Q}) + \Delta_{\overleftarrow{Q}}(S_{i_Q})$$

$$
\begin{aligned}
&= 0 + b_1 + \frac{p_1}{2} \\
&= b_1 + \frac{p_1}{2}\,.
\end{aligned}
$$

By induction hypothesis (1) and (2), for each session index $i \in \mathcal{A}_{Q_0}$, $r_{\overleftarrow{Q}}(S_i) \leq b_1 + p_1/2$. It follows that for each session index $i \in \mathcal{A}_{Q_0}$, $r_{\overleftarrow{Q}}(S_i) \leq r_Q(S_i)$, so that by update operation, $r_Q(S_i) = r_{\overleftarrow{Q}}(S_i)$. Thus, by induction hypothesis (1) and (2), for each session index $i \in \mathcal{A}_{Q_0}$, $r_Q(S_i) = b_1 + p_1/2$ if $i \in \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$, or 0 if $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q})$. Since $\mathcal{S} \mid_\alpha (Q_0, Q) = \mathcal{S} \mid_\alpha (Q_0, \overleftarrow{Q}) \cup \{i_Q\}$ and $r_Q(S_{i_Q}) = b_1 + p_1/2$, it follows that for each session index $i \in \mathcal{A}_{Q_0}$, $r_Q(S_i) = b_1 + p_1/2$ if $i \in \mathcal{S} \mid_\alpha (Q_0, Q)$, or 0 if $i \in \mathcal{A}_{Q_0} \setminus \mathcal{S} \mid_\alpha (Q_0, Q)$. This establishes (1) and (2).

■

We continue to prove (3).

Claim 6.11 *Edge $e^{(1)}_{i^{(1)}_l}$ is a bottleneck edge for state Q.*

Proof: By construction of network $G(\mathsf{Sched})$, edge $e^{(1)}_{i^{(1)}_l}$ is traversed by sessions $i^{(1)}_f$ and $i^{(1)}_l$. By induction hypothesis(6), $\mathcal{A}_Q = \mathcal{A}_{Q_0}$, so that both $i^{(1)}_f \in \mathcal{A}_Q$ and $i^{(1)}_l \in \mathcal{A}_Q$. This implies that $e^{(1)}_{i^{(1)}_l} \in AE_Q$.

By induction hypothesis(3), edge $e^{(1)}_{i^{(1)}_l}$ is a bottleneck edge for state $\overleftarrow{Q}$. Thus, Proposition 4.1(2) implies that edge $e^{(1)}_{i^{(1)}_l}$ is a bottleneck edge for state Q, as needed. ■

We continue to calculate the fair share of edge $e^{(1)}_{i^{(1)}_l}$ for state Q.

Lemma 6.12 $FS_Q(e^{(1)}_{i^{(1)}_l}) = b_1 + p_1/4$.

Proof: By induction hypothesis (7), $\mathcal{A}_Q = \mathcal{A}_{Q_0}$. Hence, both $i^{(1)}_f \in \mathcal{A}_Q$ and $i^{(r)}_l \in \mathcal{A}_Q$. It follows that $\mathcal{A}_Q \mid e^{(1)}_{i^{(1)}_l} = \{i^{(1)}_f, i^{(1)}_l\}$, so that

$$
FS_Q(e^{(1)}_{i^{(1)}_l}) = \frac{cap(e^{(1)}_{i^{(1)}_l}) - allot_Q(e^{(1)}_{i^{(1)}_l})}{|\mathcal{A}_Q \mid e^{(1)}_{i^{(1)}_l}|}
$$

$$
\begin{aligned}
&= \frac{cap(e^{(1)}_{i^{(1)}_l})}{2} \\
&= b_1 + \frac{p_1}{4} \qquad \text{(by construction of } G(\mathsf{Sched})\,,
\end{aligned}
$$

as needed. ■

We continue to calculate the fair share of other edges of the first epoch for state Q.

Lemma 6.13 *For any session index $i \in \mathcal{A}_{Q_0} \setminus \{i^{(1)}_f, i^{(1)}_l\}$,*

$$FS_Q(e^{(1)}_i) \;=\; b_1 + \frac{p_1}{2}\,.$$

Proof: Induction hypothesis(7) implies that $\mathcal{A}_Q = \mathcal{A}_{Q_0}$. Hence, both $i \in \mathcal{A}_Q$ and $i^{(1)}_f \in \mathcal{A}_Q$. By construction of $G(\mathsf{Sched})$, edge $e^{(1)}_i$ is traversed by sessions S_i and $S_{i^{(1)}_f}$. It follows that $\mathcal{A}_Q \mid e^{(1)}_i = \{i, i^{(1)}_f\}$, so that

$$
\begin{aligned}
FS_Q(e^{(1)}_i) &= \frac{cap(e^{(1)}_i) - allot_Q(e^{(1)}_i)}{|\mathcal{A}_Q \mid e^{(1)}_i|} \\
&= \frac{cap(e^{(1)}_i)}{2} \\
&= b_1 + \frac{p_1}{2} \qquad \text{(by construction of } G(\mathsf{Sched})\,,
\end{aligned}
$$

as needed. ■

We next prove (4).

Claim 6.14 *For any session $i \in \mathcal{A}_{Q_0} \setminus \{i^{(1)}_f, i^{(1)}_l\}$, edge $e^{(1)}_i$ is not a bottleneck edge for Q.*

Proof: Lemma 6.13 implies that $FS_Q(e^{(1)}_i) = b_1 + p_1/2$, while by Lemma 6.12, $FS_Q(e^{(1)}_{i^{(1)}_l}) = b_1 + p_1/4$.

By construction of network $G(\mathsf{Sched})$, session $i^{(1)}_f$ traverses every edge $e^{(1)}_i$, for each session index $i \in \mathcal{A}_{Q_0} \setminus \{i^{(1)}_f\}$, and no other higher epoch edge. Thus,

$$
\begin{aligned}
MFS_Q(S_{i^{(1)}_f}) &\le \min\{FS_Q(e^{(1)}_{i^{(1)}_l}), \bigcup_{i \in \mathcal{A}_{Q_0} \setminus \{i^{(1)}_f, i^{(1)}_l\}} \{FS_Q(e^{(1)}_i)\}\} \\
&= \min\{b_1 + \frac{p_1}{4}, b_1 + \frac{p_1}{2}\} \\
&= b_1 + \frac{p_1}{4}\,.
\end{aligned}
$$

Thus, $S_{i_f^{(1)}} \in \mathcal{A}_Q \mid e_i^{(1)}$ while

$$\begin{aligned} FS_Q(e_i^{(1)}) - MFS_Q(S_{i_f^{(1)}}) &= b_1 + \frac{p_1}{2} - (b_1 + \frac{p_1}{4}) \\ &= \frac{p_1}{4} \\ &\neq 0\,, \end{aligned}$$

which implies that $e_i^{(1)}$ is not a bottleneck edge for state Q, as needed. ■

We continue to prove (5).

Claim 6.15 *For any integer r' such that $1 < r' \leq d/2$, edge $e^{(r')} \in E^{(r')}$ is not a bottleneck edge for state Q.*

Proof: By construction of network $G(\mathsf{Sched})$, $e^{(r')}$ is traversed by some session $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, which also traverses edge $e_i^{(1)}$.

We proceed to calculate the fair shares of edges $e_i^{(1)}$ and $e^{(r')}$. Lemma 6.13 implies that

$$\begin{aligned} FS_Q(e_i^{(1)}) &= b_1 + p_1/2 \\ &= \frac{cap(e_i^{(1)})}{2} \qquad \text{(by construction of } G(\mathsf{Sched}))\,. \end{aligned}$$

By construction of $G(\mathsf{Sched})$, edge $e^{(r')}$ is traversed by sessions S_i and $S_{i'}$, for some session index $i' \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$. Induction hypothesis(7) implies that $\mathcal{A}_Q = \mathcal{A}_{Q_0}$. Hence, both $i \in \mathcal{A}_Q$ and $i' \in \mathcal{A}_Q$, so that $\mathcal{A}_Q \mid e^{(r')} = \{i, i'\}$. Thus,

$$\begin{aligned} FS_Q(e^{(r')}) &= \frac{cap(e^{r'}) - allot_Q(e^{(r')})}{|\mathcal{A}_Q \mid e^{(r')}|} \\ &= \frac{cap(e^{(r')})}{2}\,. \end{aligned}$$

By Proposition 5.3, $cap(e_i^{(1)}) < cap(e^{(r')})$. It follows that $FS_Q(e_i^{(1)}) < FS_Q(e^{(r')})$. By definition of minimum fair share, this implies that $MFS_Q(S_i) < FS_Q(e^{(r')})$, so that edge $e^{(r')}$ is not a bottleneck edge for state Q, as needed. ■

We continue to prove (7). We consider seperately each of the sessions active in Q_0. We start with session $S_{i_f^{(1)}}$.

By construction, $i_f^{(1)}$ traverses edge $e^{(1)}_{i_l^{(1)}}$; moreover, for each index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, $i_f^{(1)}$ traverses edge $e_i^{(1)}$. We consider each of these edges seperately.

Take first edge $e^{(1)}_{i_l^{(1)}}$. By construction of $G(\mathsf{Sched})$, edge $e^{(1)}_{i_l^{(1)}}$ is traversed by sessions $S_{i_f^{(1)}}$ and $S_{i_l^{(1)}}$. Lemma 6.12 implies that $FS_Q(e^{(1)}_{i_l^{(1)}}) = b_1 + p_1/4$.

By (1), $r_Q(S_{i_f^{(1)}}) = b_1 + p_1/2$, so that

$$\begin{aligned} r_Q(S_{i_f^{(1)}}) - FS_Q(e^{(1)}_{i_l^{(1)}}) &= b_1 + \frac{p_1}{2} - (b_1 + \frac{p_1}{4}) \\ &= \frac{p_1}{4} \\ &\neq 0 \qquad \text{(by definition of } p_1). \end{aligned}$$

Since Alg is bottleneck, it follows that edge $e^{(1)}_{i_l^{(1)}}$ does not cause the termination of session $S_{i_f^{(1)}}$ in state Q.

Take now the edge $e_i^{(1)}$ for any session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$. Claim 6.14 implies that $e_i^{(1)}$ is not a bottleneck edge for state Q. Since Alg is bottleneck, it follows that edge $e_i^{(1)}$ does not cause the termination of session $S_{i_f^{(1)}}$ in state Q.

Since none of the edges traversed by session $S_{i_f^{(1)}}$ causes its termination in state Q, it follows that $i_f^{(1)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$, as needed.

We continue with session $S_{i_l^{(1)}}$. By construction of $G(\mathsf{Sched})$, session $S_{i_l^{(1)}}$ traverses edge $e^{(1)}_{i_l^{(1)}}$ and no other edge. By Lemma 6.12, $FS_Q(e^{(1)}_{i_l^{(1)}}) = b_1 + p_1/4 > 0$. By (2), $r_Q(S_{i_l^{(1)}}) = 0$. Thus,

$$FS_Q(e^{(1)}_{i_l^{(1)}}) \neq r_Q(S_{i_l^{(1)}}).$$

Since Alg is bottleneck, it follows that edge $e^{(1)}_{i_l^{(1)}}$ does not cause the termination of session $S_{i_l^{(1)}}$ in state Q.

Since session $S_{i_l^{(1)}}$ does not traverse any other edge, it follows that

$$i_l^{(1)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q),$$

as needed.

We finally consider all remaining sessions. By construction of network $G(\mathsf{Sched})$, session S_i traverses edge $e_i^{(1)}$; moreover, S_i also traverses edge $e^{(r')} \in E^{(r')}$, for some integer r' such that $1 < r' \leq d/2$.

By Claims 6.14 and 6.15, none of the edges $e_i^{(1)}$, and $e^{(r')}$, respectively, is a bottleneck edge for state Q. Since Alg is bottleneck, it follows that none of these edges causes the termination of session S_i in state Q, which implies that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$, as needed.

The proof of Lemma 6.5 is now complete. ■

So far we have determined the rates and status of all sessions till state $\overleftarrow{Q^{(1)}}$. Consider now state $Q^{(1)}$. Clearly, session $S_{i_l^{(1)}}$ is scheduled in front of $Q^{(1)}$. By construction of G, $S_{i_l^{(1)}}$ traverses only edge $e_{i_l^{(1)}}$ of capacity $cap(e^{(1)}_{i_l^{(1)}}) = 2b_1 + p_1/2$, which is also traversed by session $S_{i_f^{(1)}}$. By Lemma 6.5(1) and (2), $r_{\overleftarrow{Q^{(1)}}}(S_{i_f^{(1)}}) = b_1 + p_1/2$ and $r_{\overleftarrow{Q^{(1)}}}(S_{i_l^{(1)}}) = 0$. Thus, by definition of update operation,

$$\begin{aligned}
\Delta_{\overleftarrow{Q^{(1)}}}(e^{(1)}_{i_l^{(1)}}, i_l^{(1)}) &= \Delta_{\overleftarrow{Q^{(1)}}}(i_l^{(1)}) \\
&= \frac{1}{2}\, cap(e^{(1)}_{i_l^{(1)}}) - r_{\overleftarrow{Q^{(1)}}}(S_{i_l^{(1)}}) \\
&= \frac{1}{2} \cdot (2b_1 + \frac{p_1}{4}) - 0 \\
&= b_1 + \frac{p_1}{4}\,,
\end{aligned}$$

so that

$$\begin{aligned}
r_{Q^{(1)}}(S_{i_l^{(1)}}) &= r_{\overleftarrow{Q^{(1)}}}(S_{i_l^{(1)}}) + \Delta_{\overleftarrow{Q^{(1)}}} \\
&= 0 + b_1 + \frac{p_1}{4} \\
&\quad \text{(by induction hypothesis)} \\
&= b_1 + \frac{p_1}{4}\,.
\end{aligned}$$

Moreover, by definition of update operation,

$$\begin{aligned}
r_{Q^{(1)}}(S_{i_f^{(1)}}) &= r_{Q^{(1)}}(S_{i_l^{(1)}}) \\
&= b_1 + \frac{p_1}{4}\,,
\end{aligned}$$

as needed by (2a).

Moreover, for any index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, S_i does not share an edge with $S_{i_l}^{(1)}$, which by definition of update operation implies that,

$$\begin{aligned}
r_{Q^{(1)}}(S_i) &= r_{\overleftarrow{Q^{(1)}}}(S_i) \\
&\quad \text{(by definition of update operation)} \\
&= b_1 + \frac{p_1}{2}\,,
\end{aligned}$$

as needed by (2b).

We proceed to determine the status of each session in state $Q^{(1)}$.

Claim 6.16 *For each index $i \in \mathcal{A}_{Q_0}$, $i \in \{i_f^{(1)}, i_l^{(1)}\}$ if and only if*

$$i \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(1)}).$$

Proof: Lemma 6.5(3), (4), and (5) imply that the only bottleneck edge for state $\overleftarrow{Q^{(1)}}$ is edge $e^{(1)}_{i_l^{(1)}}$). By Lemma 6.5 (6), $\mathcal{A}_{\overleftarrow{Q^{(1)}}} = \mathcal{A}_{Q^{(1)}}$. Thus, the only bottleneck edge for state $Q^{(1)}$ is edge $e^{(1)}_{i_l^{(1)}}$. By construction of $G(\mathsf{Sched})$, edge $e^{(1)}_{i_l^{(1)}}$ is traversed by sessions $i_f^{(1)}$ and $i_l^{(1)}$. Lemma 6.5(6) implies that both $i_f^{(1)} \in \mathcal{A}_{Q^{(1)}}$ and $i_l^{(1)} \in \mathcal{A}_{Q^{(1)}}$. Thus,

$$\begin{aligned} FS_{Q^{(1)}}(e^{(1)}_{i_l^{(1)}}) &= \frac{cap(e^{(1)}_{i_l^{(1)}}) - allot_{\overrightarrow{Q^{(1)}}}(e^{(1)}_{i_l^{(1)}})}{|\mathcal{A}_{Q^{(1)}} \mid e^{(1)}_{i_l^{(1)}}|} \\ &= \frac{2b_1 + p_1/2}{2} \\ &= b_1 + \frac{p_1}{4}. \end{aligned}$$

Moreover, by (1), $r_{Q^{(1)}}(S_{i_f^{(1)}}) = r_{Q^{(1)}}(S_{i_l^{(1)}}) = b_1 + p_1/4$, so that $r_{Q^{(1)}}(S_{i_f^{(1)}}) = r_{Q^{(1)}}(S_{i_l^{(1)}}) = FS_{Q^{(1)}}(e^{(1)}_{i_l^{(1)}})$. Since Alg is bottleneck, it follows that both $i_f^{(1)} \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(1)})$, and $i_l^{(1)} \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(1)})$. Moreover, for each session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, session S_i does not traverse any bottleneck edge. Since Alg is bottleneck, session S_i is not terminated in state $Q^{(1)}$. Thus, for each session index $i \in \mathcal{A}_{Q_0}$, $i \in \{i_f^{(1)}, i_l^{(1)}\}$, if and only if $i \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(1)})$, as needed. ■

So far we have proved that claims (1), (2) and (3) of Lemma 6.2 hold for the basis case, where $r = 1$. We continue to prove (4).

By construction of network $G(\mathsf{Sched})$, for any session index $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, edge $e_i^{(1)}$ is traversed by sessions $S_{i_f^{(1)}}$ and S_i. Recall that $r_{Q^{(1)}}(S_{i_f^{(1)}}) = b_1 + p_1/4$, while $r_{Q^{(1)}}(S_i) = b_1 + p_1/2$. Thus,

$$\begin{aligned} residual_{Q^{(1)}}(e_i^{(1)}) &= cap(e_i^{(1)}) - \sum_{S \in \mathcal{A}_{Q^{(1)}}|e} r_{Q^{(1)}}(S) \\ &= cap(e) - r_{Q^{(1)}}(S_{i_f^{(1)}}) - r_{Q^{(1)}}(S_i) \\ &= 2b_1 + p_1 - (b_1 + \frac{p_1}{4}) - (b_1 + \frac{p_1}{2}) \\ &= \frac{p_1}{4}, \end{aligned}$$

as needed by (4a).

By Lemma 6.16, $i_f^{(1)} \notin \mathcal{A}_{\underset{Q^{(1)}}{\rightarrow}}$, while for each $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$, $i \in \mathcal{A}_{\underset{Q^{(1)}}{\rightarrow}}$. Thus, for each $i \in \mathcal{A}_{Q_0} \setminus \{i_f^{(1)}, i_l^{(1)}\}$,

$$\begin{aligned} FS_{\underset{Q^{(1)}}{\rightarrow}}(e_i^{(1)}) &= \frac{cap(e_i^{(1)}) - allot_{\underset{Q^{(1)}}{\rightarrow}}(e_i^{(1)})}{|\mathcal{A}_{\underset{Q^{(1)}}{\rightarrow}} \mid e_i^{(1)}|} \\ &= \frac{2b_1 + p_1 - (b_1 + p_1/4)}{1} \\ &= b_1 + \frac{3}{4}\, p_1\,, \end{aligned}$$

as needed by (4b).

The proof of the basis case of Lemma 6.2 is now complete.

Induction Hypothesis (Lemma 6.2)

Assume inductively that for some integer r such that $1 < r \leq d/2$, the following hold for any integer r'' such that $1 \leq r'' < r$:

(1) $l_{r''} < \infty$;

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

  (a) if $i \in \{i_f^{(r'')}, i_l^{(r'')}\}$, then $r_{Q^{(r'')}}(S_i) = b_{r''} + p_{r''}/4$;

  (b) if $i \in \mathcal{A}_{\underset{Q_{(r''-1)}}{\rightarrow}} \setminus \{i_f^{(r'')}, i_l^{(r'')}\}$, then $r_{Q^{(r'')}}(S_i) = b_{r''} + p_{r''}/2$;

(3) for each session $i \in \mathcal{A}_{\underset{Q^{(r''-1)}}{\rightarrow}}$, $i \in \{i_f^{(r'')}, i_l^{(r'')}\}$ if and only if $i \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r'')})$;

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

  (a) $residual_{Q^{(r'')}}(e_i^{(r')}) \geq p_{r''}/4$;

  (b) $FS_{\underset{Q^{(r'')}}{\rightarrow}}(e_i^{(r')}) \geq b_{r''} + 3\, p_{r''}/4$.

Induction Step (Lemma 6.2)

For the induction step, we will show:

(1) $l_r < \infty$;

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

  (a) if $i \in \{i_f^{(r)}, i_l^{(r)}\}$, then $r_{Q^{(r)}}(S_i) = b_r + p_r/4$;

  (b) if $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, then $r_{Q^{(r)}}(S_i) = b_r + p_r/2$;

(3) for each session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $i \in \{i_f^{(r)}, i_l^{(r)}\}$ if and only if $i \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r)})$;

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

  (a) $residual_{Q^{(r)}}(e_i^{(r')}) \geq p_r/4$;

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

We start by proving a technical lemma, stating that the available capacity on any edge of a higher than the rth epoch is at least $p_{r-1}/4$.

Lemma 6.17 *Let r' be any integer such that $r \leq r' \leq d/2$, and consider any edge $e^{(r')} \in E^{(r')}$. Then,*

$$residual_{Q^{(r-1)}}(e^{(r')}) \quad \geq \quad \frac{p_{r-1}}{8}.$$

Proof: By induction on r'. We start with the basis case where $r' = r$.

By construction of network $G(\mathsf{Sched})$, $e^{(r')}$ is traversed by two sessions i, j such that $\{i, j\} \subseteq \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. Clearly,

$$residual_{Q^{(r-1)}}(e^{(r')}) \quad = \quad cap(e^{(r')}) - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_j).$$

By Proposition 5.3, $cap(e^{(r')}) \geq cap(e^{(r)})$ for any edge $e^{(r)} \in E^{(r)}$. By construction of network $G(\mathsf{Sched})$, either $cap(e^{(r)}) = 2\,b_r + p_r/2$ or $cap(e^{(r)}) = 2\,b_r + p_r$, so that $cap(e^{(r)}) \geq 2\,b_r + p_r/2$.

Thus, $cap(e^{(r')}) \geq 2\,b_r + p_r/2$. Hence,

$$\begin{aligned} residual_{Q^{(r-1)}}(e^{(r')}) &\geq 2\,b_r + \frac{p_r}{2} - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_j) \\ &= 2\,b_r + \frac{p_r}{2} - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\ &\quad \text{(by induction hypothesis (2b))} \\ &= 2 \cdot \left(b_{r-1} + \frac{p_{r-1}}{2}\right) + \frac{1}{2} \cdot \frac{p_{r-1}}{4} - 2\,b_{r-1} - p_{r-1} \\ &\quad \text{(by recursive definition of } b_r \text{ and } p_r) \\ &= \frac{p_{r-1}}{8}, \end{aligned}$$

as needed.

Assume inductively that for every integer r' such that $r \leq r' \leq d/2$ the following hold for any integer r'' such that $r \leq r'' < r'$ and any edge $e^{(r'')} \in E^{(r'')}$:

$$residual_{Q^{(r-1)}}(e^{(r'')}) \geq \frac{p_{r-1}}{8}.$$

For the induction step we prove that

$$residual_{Q^{(r-1)}}(e^{(r')}) \geq \frac{p_{r-1}}{8}.$$

By construction of $G(\mathsf{Sched})$, edge $e^{(r')}$ is traversed by two sessions S_i and S_j, such that both $S_i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$ and $S_j \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. Clearly, $residual_{Q^{(r-1)}}(e^{(r')}) = cap(e^{(r')}) - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_j)$. Moreover, by construction of $G(\mathsf{Sched})$, session S_i traverses also edge $e_i^{(r'-1)}$, which is also traversed by session $i_f^{(r'-1)}$. Since $r' > r$, $r' - 1 \geq r$, so that induction hypothesis(3) implies that $i_f^{(r'-1)} \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$. Thus, by induction hypothesis (2b), $r_{Q^{(r-1)}}(S_{i_f^{(r'-1)}}) = b_{r-1} + p_{r-1}/2$. Moreover, induction hypothesis (2b) implies that $r_{Q^{(r-1)}}(S_j) = b_{r-1} + p_{r-1}/2$, so that $r_{Q^{(r-1)}}(S_{i_f^{(r'-1)}}) = r_{Q^{(r-1)}}(S_i)$. Thus,

$$\begin{aligned} residual_{Q^{(r-1)}}(e^{(r'-1)}) &= cap(e^{(r'-1)}) - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_{i_f^{(r'-1)}}) \\ &= cap(e^{(r'-1)}) - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_j). \end{aligned}$$

Hence,

$$\begin{aligned} & residual_{Q^{(r-1)}}(e^{(r')}) - residual_{Q^{(r-1)}}(e^{(r'-1)}) \\ = \; & (cap(e^{(r')}) - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_j)) - (cap(e^{(r'-1)}) - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_j)) \\ = \; & cap(e^{(r')}) - cap(e^{(r'-1)}). \end{aligned}$$

By Propositon 5.3, $cap(e^{(r')}) > cap(e^{(r'-1)})$. Thus,

$$residual_{Q^{(r-1)}}(e^{(r')}) \quad > \quad residual_{Q^{(r-1)}}(e^{(r'-1)}) .$$

Induction hypothesis 4(a) implies that $residual_{Q^{(r-1)}}(e_i^{(r'-1)}) \geq p_{r-1}/8$, so that

$$residual_{Q^{(r-1)}}(e_i^{(r')}) \geq p_{r-1}/8 ,$$

as needed. ■

We continue to prove (1). Asume, by the way of contradiction that $l_r = \infty$. We proceed by case analysis.

Assume first that $S \mid_\alpha (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\} = \emptyset$. By induction hypothesis (3), for each session index $i \in \mathcal{A}_{Q^{(r-1)}}$ only sessions $i_f^{(r-1)}, i_l^{(r-1)}$ become done in state $Q^{(r-1)}$. All other active sessions remain active. However, none of these sessions will be further scheduled for a increase. Thus, it suffices to prove that $|\mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}| > 0$ and at least one session $i \in \mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$ has not yet received its max-min fair rate.

By induction hypothesis (3), for each integer $r' \leq r-1$, only two sessions, namely $i_f^{(r')}, i_l^{(r')}$ becomes done between states $Q^{(r')}$ and $Q^{(r'+1)}$. These two sessions are done from state $\overrightarrow{Q^{(r')}}$ on. Thus, $|\mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}| = d - 2(r-1)$. Since $r \leq d/2$, it follows that $|\mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}| \geq d - 2(d/2 - 1)$, so that $|\mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}| \geq 2$. Thus, $\mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$ contains at least two elements.

Consider any arbitrary index $i \in \mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$. By construction of $G(\mathsf{Sched})$, S_i traverses edge $e_i^{(r')}$ for each integer r' such that $1 \leq r' < r$. Induction hypothesis (4a) imples that $residual_{Q^{(r-1)}}(e_i^{(r')}) \geq p_{r-1}/4$. Thus, by definition of update operation, $\Delta_{Q^{(r-1)}}(i, e_i^{(r')}) > 0$.

Moreover, by construction of $G(\mathsf{Sched})$, S_i traverses at least one edge $e^{(r'')} \in E^{(r'')}$ for some integer r'' such that $r \leq r'' \leq d/2$. Lemma 6.17 implies that $residual_{Q^{(r-1)}}(e^{(r'')}) \geq p_{r-1}/8$. Thus, by definition of update operation, $\Delta_{Q^{(r-1)}}(i, e_i^{(r'')}) > 0$. By construction of $G(\mathsf{Sched})$, S_i traverses no other edges, so that $\Delta_{Q^{(r-1)}}(i) > 0$. Thus, by definition of update operation, $r_{Q^{(r-1)}}(S_i)$ can be increased without decreasing the rate of any other session. A contradiction to max-min fairness.

Assume now that $S \mid_\alpha (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\} \neq \emptyset$. Clearly, at least one session $i_f^{(r)} \in \mathcal{A}_{\overrightarrow{Q^{r-1}}}$ is scheduled for an increase during the execution epoch $\alpha^{(l_r)}$. By construction of $G(\mathsf{Sched})$, $i_f^{(r)}$ shares an edge $e_{i_l^{(r)}}$ with some other session $i_l^{(r)} \in \mathcal{A}_{\overrightarrow{Q^{r-1}}}$, of capacity $cap(e_{i_l^{(r)}}) =$

$2\,b_r + p_r/2$. Moreover, by construction of $G(\mathsf{Sched})$, session $S_{i_f^{(r)}}$ shares an edge $e_i^{(r)}$ with session S_i, for any session index $i \in \mathcal{A}_{\underset{Q^{r-1}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$.

By definition of $Q_f^{(r)}$ no active session in $\overrightarrow{Q^{(r-1)}}$ is scheduled for an increase between states $Q^{(r-1)}$ and $\overleftarrow{Q_f^{(r)}}$. As an immediate consequence the rates and status of sessions do not change between these two states, as stated by the following observation.

Observation 6.1 *The following hold for the state* $\overleftarrow{Q_f^{(r)}}$*:*

(1) *for any session index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_i) = r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_i)$;

(2) *for any session index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, \overleftarrow{Q_f^{(r)}})$

(3) *for any session index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}\}$, $residual_{\underset{Q_f^{(r)}}{\leftarrow}}(e_i^{(r)}) = residual_{Q^{(r-1)}}(e_i^{(r)})$.

Roughly speaking, (1) implies that the rates of active sessions do not change between states $\overrightarrow{Q^{(r-1)}}$ and $\overleftarrow{Q_f^{(r)}}$, while (2) guarantees that no session is terminated on any state between these two states. Moreover, the available capacity on edges of epoch r remain unchanged between states $\overrightarrow{Q^{(r-1)}}$ and $\overleftarrow{Q_f^{(r)}}$.

We continue with the following technical claim that describes the state $Q_f^{(r)}$.

Lemma 6.18 *The following hold for the state* $Q_f^{(r)}$*:*

(1) $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$;

(2) *for each session index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}\}$, $r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$;

(3) $FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) = b_r + p_r/4$;

(4) *for each index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, $FS_{Q_f^{(r)}}(e_i^{(r)}) = b_r + p_r/2$;

(5) *edge* $e^{(r)}_{i_l^{(r)}}$ *is a bottleneck edge for* $Q_f^{(r)}$;

(6) *for each index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, $e_i^{(r)}$ *is not a bottleneck edge for* $Q_f^{(r)}$;

(7) *for any integer* r' *such that* $r < r' \leq d/2$, *and edge* $e^{(r')} \in E^{(r')}$, $e^{(r')}$ *is not a bottleneck edge for* $Q_f^{(r)}$;

(8) *for any session index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$ *and for any integer* r' *such that* $1 \leq r' < r$, *edge* $e_i^{(r')}$ *is not a bottleneck edge for* $Q_f^{(r)}$;

(9) *for each index* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_f^{(r)})$.

Roughly speaking, conditions (1) and (2) determine rates of active sessions in state $Q_f^{(r)}$, while conditions (3) and (4) determine the fair shares of edges in $Q_f^{(r)}$ that are traversed by these sessions. Condition (5) establishes that edge $e_{i_l^{(r)}}^{(r)}$ is a bottleneck edge for $Q_f^{(r)}$), while conditions (6), (7) and (8) guarantee that no other edge in the network is a bottleneck edge for $Q_f^{(r)}$. Finally, condition (10) claims that no corresponding termination of sessions takes place. We continue with the proof of Lemma 6.18.

Proof: We start by proving (1). By construction of the network $G(\mathsf{Sched})$, session $i_f^{(r)}$ traverses edge $e_{i_l^{(r)}}^{(r)}$; moreover, for each $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, $S_{i_f^{(r)}}$ traverses edge $e_i^{(r)}$; finally, $S_{i_f^{(r)}}$ traverses edge $e_{i_f^{(r)}}^{(r')}$ for each index r' such that $1 \leq r' < r$ and no other edge. We proceed to calculate the increases for $i_f^{(r)}$ in state $\overleftarrow{Q_f^{(r)}}$ imposed by the edges it traverses. We consider seperately each of these edges.

Consider first edge $e_{i_l^{(r)}}^{(r)}$. By construction of $G(\mathsf{Sched})$, edge $e_{i_l^{(r)}}^{(r)}$ is traversed by sessions $S_{i_f^{(r)}}$ and $S_{i_l^{(r)}}$, while $cap(e_{i_l^{(r)}}^{(r)}) = 2b_r + p_r/2$. By Observation 6.1, $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) = r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_{i_f^{(r)}})$, and $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_l^{(r)}}) = r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_{i_l^{(r)}})$. Induction hypothesis (2b) implies that

$$\begin{aligned} r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_{i_f^{(r)}}) &= r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_{i_l^{(r)}}) \\ &= b_{r-1} + p_{r-1}/2\,, \end{aligned}$$

so that $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) = b_{r-1} + p_{r-1}/2$ and $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$.

Thus, by definition of iupdate function,

$$\Delta_{\underset{Q_f^{(r)}}{\leftarrow}}(i_f^{(r)}, e_{i_l^{(r)}}^{(r)}) = cap(e_{i_l^{(r)}}^{(r)}) - r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) - r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}})$$

$$
\begin{aligned}
&= 2b_r + \frac{p_r}{2} - 2\,(b_{r-1} + \frac{p_{r-1}}{2}) \\
&\quad \text{(by Observation 6.1 and induction hypothesis (2b))} \\
&= 2\,(b_r - b_{r-1}) + \frac{p_r}{2} - p_{r-1} \\
&= 2\,\frac{p_{r-1}}{2} + \frac{1}{2}\cdot\frac{p_{r-1}}{4} - p_{r-1} \\
&\quad \text{(by recursive definition of } b_r \text{ and } p_r) \\
&= \frac{p_{r-1}}{8}\,.
\end{aligned}
$$

Consider now any edge $e_i^{(r)}$, $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By construction of $G(\mathsf{Sched})$, $e_i^{(r)}$ is traversed by sessions $S_{i_f^{(r)}}$ and S_i. By Observation 6.1, $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) = r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_{i_f^{(r)}})$, and $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_i) = r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_i)$. Induction hypothesis (2b) implies that $r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_{i_f^{(r)}}) = r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_i) = b_{r-1} + p_{r-1}/2$, so that $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) = b_{r-1} + p_{r-1}/2$ and $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_i) = b_{r-1} + p_{r-1}/2$. Thus, by definition of iupdate function,

$$
\begin{aligned}
\Delta_{\underset{Q_f^{(r)}}{\leftarrow}}(i_f^{(r)}, e_i^{(r)}) &= cap(e_i^{(r)}) - r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) - r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_i) \\
&= 2b_r + p_r - 2\,(b_{r-1} + \frac{p_{r-1}}{2}) \\
&\quad \text{(by Observation 6.1 and induction hypothesis 2(b))} \\
&= 2\,(b_r - b_{r-1}) + p_r - p_{r-1} \\
&= 2\,\frac{p_{r-1}}{2} + \frac{p_{r-1}}{4} - p_{r-1} \\
&\quad \text{(by recursive definition of } b_r \text{ and } p_r) \\
&= \frac{p_{r-1}}{4}\,.
\end{aligned}
$$

Consider now edge $e^{(r')}_{i_f^{(r)}}$ for each index r' such that $1 \le r' < r$. Observation 6.1(3) implies that $\mathit{residual}_{\underset{Q_f^{(r)}}{\leftarrow}}(e^{(r')}_{i_f^{(r)}}) = \mathit{residual}_{Q^{(r-1)}}(e^{(r')}_{i_f^{(r)}})$, so that induction hypothesis(4a) implies that $\mathit{residual}_{\underset{Q_f^{(r)}}{\leftarrow}}(e^{(r')}_{i_f^{(r)}}) \ge p_{r-1}/4$. Thus, by definition of update operation,

$$
\begin{aligned}
\Delta_{\underset{Q_f^{(r)}}{\leftarrow}}(i_f^{(r)}) &= \min\{p_{r-1}/8, p_{r-1}/4\} \\
&= p_{r-1}/8\,.
\end{aligned}
$$

Moreover, by definition of update operation,

$$
r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) + \Delta_{\underset{Q_f^{(r)}}{\leftarrow}}(i_f^{(r)})
$$

$$
\begin{aligned}
& b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} \\
& \text{(by Observation 6.1 and induction hypothesis(2b))} \\
= \quad & b_{r-1} + \frac{5p_{r-1}}{8}\,.
\end{aligned}
$$

This completes the proof of (1).

To prove (2), take any session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}\}$. By Observation 6.18, $r_{\underset{Q_f^{(r)}}{\leftarrow}}(S_i) = r_{Q^{(r-1)}}(S_i)$. By induction hypothesis(1b), $r_{Q^{(r-1)}}(S_i) = b_{r-1} + p_{r-1}/2$. Clearly,

$$
\begin{aligned}
r_{Q_f^{(r)}}(S_{i_f^{(r)}}) - r_{Q_f^{(r)}}(S_i) &= b_{r-1} + \frac{5p_{r-1}}{8} - b_{r-1} - \frac{p_{r-1}}{2} \\
&= \frac{p_{r-1}}{8} \\
&> 0\,.
\end{aligned}
$$

Thus, by definition of update operation, $r_{Q_f^{(r)}}(S_i) = r_{Q^{(r-1)}}(S_i) = b_{r-1} + p_{r-1}/2$, as needed.

We continue to prove (3).

Claim 6.19 $FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) = b_r + p_r/4$.

Proof: By construction of network $G(\mathsf{Sched})$, edge $e^{(r)}_{i_l^{(r)}}$ is traversed by sessions $i_f^{(r)}$ and $i_l^{(r)}$. Observation 6.1(2) implies that $\mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} = \mathcal{A}_{Q_f^{(r)}}$. Thus, by induction hypothesis (3), both $i_f^{(r)} \in \mathcal{A}_{Q_f^{(r)}}$ and $i_l^{(r)} \in \mathcal{A}_{Q_f^{(r)}}$, so that $\mathcal{A}_{Q_f^{(r)}} \mid e^{(r)}_{i_l^{(r)}} = \{i_f^{(r)}, i_l^{(r)}\}$. Hence,

$$
\begin{aligned}
FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) &= \frac{cap(e^{(r)}_{i_l^{(r)}}) - allot_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}})}{|\mathcal{A}_{Q_f^{(r)}} \mid e^{(r)}_{i_l^{(r)}}|} \\
&= \frac{cap(e^{(r)}_{i_l^{(r)}})}{2} \\
&\quad \text{(by Observation 6.1 and induction hypothesis (3))} \\
&= \frac{2b_r + p_r/2}{2} \\
&\quad \text{(by construction of } G(\mathsf{Sched})) \\
&= b_r + \frac{p_r}{4}\,,
\end{aligned}
$$

as needed. ■

At this point the proof of (3) is complete. We proceed to prove (4).

Claim 6.20 *For each session* $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$,

$$FS_{Q_f^{(r)}}(e_i^{(r)}) \;=\; b_r + \frac{p_r}{2}\,.$$

Proof: Take any session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By construction of network $G(\mathsf{Sched})$, edge $e_i^{(r)}$ is traversed by sessions $i_f^{(r)}$ and i. Observation 6.1(2) implies that $\mathcal{A}_{\overrightarrow{Q^{(r-1)}}} = \mathcal{A}_{Q_f^{(r)}}$. Thus, by induction hypothesis (3), both $i_f^{(r)} \in \mathcal{A}_{Q_f^{(r)}}$ and $i \in \mathcal{A}_{Q_f^{(r)}}$, so that $\mathcal{A}_{Q_f^{(r)}} \mid e_i^{(r)} = \{i_f^{(r)}, i\}$. Thus,

$$\begin{aligned}
FS_{Q_f^{(r)}}(e_i^{(r)}) &= \frac{cap(e_i^{(r)}) - allot_{Q_f^{(r)}}(e_i^{(r)})}{|\mathcal{A}_{Q_f^{(r)}} \mid e_i^{(r)}|} \\
&= \frac{cap(e_i^{(r)})}{2} \\
&\quad \text{(by Observation 6.1 and induction hypothesis (3))} \\
&= \frac{2b_r + p_r}{2} \\
&\quad \text{(by construction of } G(\mathsf{Sched})) \\
&= b_r + \frac{p_r}{2}\,,
\end{aligned}$$

as needed. ■

We continue to prove (5). By construction of $G(\mathsf{Sched})$, edge $e_{i_l^{(r)}}^{(r)}$ is traversed by two sessions, namely $i_f^{(r)}$ and $i_l^{(r)}$. We consider seperately each of these sessions.

Take first session $i_f^{(r)}$. The following technical claim states that session $i_f^{(r)}$ receives its minimum fair share on edge $e_{i_l^{(r)}}^{(r)}$ for state $Q_f^{(r)}$.

Lemma 6.21 $MFS_{Q_f^{(r)}}(S_{i_f^{(r)}}) = FS_{Q_f^{(r)}}(e_{i_l^{(r)}}^{(r)})$.

Proof: By construction of $G(\mathsf{Sched})$, appart from edge $e_{i_l^{(r)}}^{(r)}$, for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, session $i_f^{(r)}$ traverses also edge $e_i^{(r)}$; moreover, for each integer r' such that $1 \le r' < r$, session $i_f^{(r)}$ traverses edge $e_{i_f^{(r)}}^{(r')}$.

By Claim 6.19, $FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) = b_r + p_r/4$, while by Claim 6.20, for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, $FS_{Q_f^{(r)}}(e_i^{(r)}) = b_r + p_r/2$.

Consider next edge $e^{(r')}_{i_f^{(r)}}$, for any integer r' such that $1 \leq r' < r$. Observation 6.1 (2) implies that $\mathcal{A}_{Q_f^{(r)}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Thus, $FS_{Q_f^{(r)}}(e^{(r')}_{i_f^{(r)}}) = FS_{\overrightarrow{Q^{(r-1)}}}(e^{(r')}_{i_f^{(r)}})$. By induction hypothesis (4b), $FS_{\overrightarrow{Q^{(r-1)}}}(e^{(r')}_{i_f^{(r)}}) \geq p_{r-1} + 3p_{r-1}/4$, so that $FS_{Q_f^{(r)}}(e^{(r')}_{i_f^{(r)}}) \geq p_{r-1} + 3p_{r-1}/4$. Thus, by definition of minimum fair share,

$$\begin{aligned} MFS_{Q_f^{(r)}}(S_{i_f^{(r)}}) &= \min\{b_r + \frac{p_r}{4}, b_r + \frac{p_r}{2}, b_{r-1} + \frac{3}{4}p_{r-1}\} \\ &= b_r + \frac{p_r}{4}, \end{aligned}$$

so that $MFS_{Q_f^{(r)}}(S_{i_f^{(r)}}) = FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}})$, as needed. ■

Take now session $i_l^{(r)}$. The following technical claim states that session $i_l^{(r)}$ receives its minimum fair share on edge $e^{(r)}_{i_l^{(r)}}$ for state $Q_f^{(r)}$.

Lemma 6.22 $MFS_{Q_f^{(r)}}(S_{i_l^{(r)}}) = FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}})$.

Proof: By construction of $G(\mathsf{Sched})$, session $i_l^{(r)}$ traverses edge $e^{(r)}_{i_l^{(r)}}$; moreover, for each integer r' such that $1 \leq r' < r$, session $i_l^{(r)}$ traverses edge $e^{(r')}_{i_l^{(r)}}$.

Take first edge $e^{(r)}_{i_l^{(r)}}$. Claim 6.19 implies that $FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) = b_r + p_r/4$.

Take next edge $e^{(r')}_{i_l^{(r)}}$, for some integer r' such that $1 \leq r' < r$. Observation 6.1 (2) implies that all active sessions in $\mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$ are also active in state $Q_f^{(r)}$, so that $\mathcal{A}_{Q_f^{(r)}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Thus, by definition of fair share, $FS_{Q_f^{(r)}}(e^{(r')}_{i_l^{(r)}}) = FS_{\overrightarrow{Q^{(r-1)}}}(e^{(r')}_{i_l^{(r)}})$. Induction hypothesis 4(b) implies that $FS_{\overrightarrow{Q^{(r-1)}}}(e^{(r')}_{i_l^{(r)}}) \geq p_{r-1} + 3p_{r-1}/4$, so that $FS_{Q_f^{(r)}}(e^{(r')}_{i_l^{(r)}}) \geq p_{r-1} + 3p_{r-1}/4$. Thus, by definition of minimum fair share,

$$\begin{aligned} MFS_{Q_f^{(r)}}(S_{i_l^{(r)}}) &= \min\{b_r + \frac{p_r}{4}, b_{r-1} + \frac{3}{4}\,p_{r-1}\} \\ &= b_r + \frac{p_r}{4}, \end{aligned}$$

so that $MFS_{Q_f^{(r)}}(S_{i_l^{(r)}}) = FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}})$, as needed. ■

Lemmas 6.21 and 6.22 imply that both sessions, traversing edge $e^{(r)}_{i^{(r)}_l}$, namely $i^{(r)}_f$ and $i^{(r)}_l$, receive minimum fair share on edge $e^{(r)}_{i^{(r)}_l}$. Thus, by definition of a bottleneck edge, it follows that $e^{(r)}_{i^{(r)}_l}$ is a bottleneck edge for state $Q^{(r)}_f$, which completes the proof of (5).

We continue to prove (6). By construction of network $G(\mathsf{Sched})$, for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i^{(r)}_f, i^{(r)}_l\}$, edge $e^{(r)}_i$ is traversed by two sessions, namely $i^{(r)}_f$ and i. Claim 6.20 implies that $FS_{Q^{(r)}_f}(e^{(r)}_i) = b_r + p_r/2$. Claim 6.21 implies that $MFS_{Q^{(r)}_f}(S_{i^{(r)}_f}) = FS_{Q^{(r)}_f}(e^{(r)}_{i^{(r)}_l})$. By Claim 6.19, $FS_{Q^{(r)}_f}(e^{(r)}_{i^{(r)}_l}) = b_r + p_r/4$. Since both $b_r \neq 0$ and $p_r \neq 0$, it follows that $b_r + p_r/4 < b_r + p_r/2$, so that $MFS_{Q^{(r)}_f}(S_{i^{(r)}_f}) < FS_{Q^{(r)}_f}(e^{(r)}_i)$. Thus, by definition of bottleneck edge, $e^{(r)}_i$ is not a bottleneck edge for state $Q^{(r)}_f$. This completes the proof of (6).

We proceed to prove (7). By construction of network $G(\mathsf{Sched})$, for each integer r' such that $r < r' \leq d/2$, every edge $e^{(r')} \in E^{(r')}$ is traversed by two sessions S_i, S_j such that $\{i,j\} \subseteq \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i^{(r)}_f, i^{(r)}_l\}$. Observation 6.1 (2) implies that $\{i,j\} \subseteq \mathcal{A}_{Q^{(r)}_f}$. Thus, $FS_{Q^{(r)}_f}(e^{(r')}) = cap(e^{(r')})/2$. By construction of network $G(\mathsf{Sched})$, S_i traverses also edge $e^{(r)}_i$ of capacity $cap(e^{(r)}_i) = 2b_r + p_r$. Claim 6.20 implies that $FS_{Q^{(r)}_f}(e^{(r)}_i) = b_r + p_r/2$. Clearly, $FS_{Q^{(r)}_f}(e^{(r)}_i) = cap(e^{(r)}_i)/2$. Proposition 5.3 implies that $cap(e^{(r)}_i) < cap(e^{(r')})$, so that

$$cap(e^{(r)}_i)/2 < cap(e^{(r')})/2\,.$$

Thus, $FS_{Q^{(r)}_f}(e^{(r)}_i) < FS_{Q^{(r)}_f}(e^{(r')})$, so that, by definition of minimum fair share,

$$MFS_{Q^{(r)}_f}(S_i) \;\; < \;\; FS_{Q^{(r)}_f}(e^{(r')})\,,$$

so that $e^{(r')}$ is not a bottleneck edge for $Q^{(r)}_f$, which completes the proof of (7).

We next prove (8). Take any session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. By construction of network $G(\mathsf{Sched})$, for each integer r' such that $1 \leq r' < r$, session S_i traverses edges $e^{(r')}_i$. Induction hypothesis (6b) implies that $FS_{\overrightarrow{Q^{(r-1)}}}(e^{(r')}_i) \geq b_{r-1} + 3p_{r-1}/4$. By Observation 6.1, $\mathcal{A}_{Q^{(r)}_f} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$ so that, by definition of fair share, $FS_{Q^{(r)}_f}(e^{(r')}_i) = FS_{Q^{(r-1)}}(e^{(r')}_i)$. Thus, $FS_{Q^{(r)}_f}(e^{(r')}_i) \geq b_{r-1} + 3p_{r-1}/4$.

We proceed by case analysis on i. Assume first that $i \notin \{i^{(r)}_f, i^{(r)}_l\}$. By construction of network $G(\mathsf{Sched})$, session S_i traverses edge $e^{(r)}_i$. Claim 6.20 implies that

$$FS_{Q^{(r)}_f}(e^{(r)}_i) \;\; = \;\; b_r + \frac{p_r}{2}$$

$$
\begin{aligned}
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} \\
&\quad \text{(by recursive definition of } b_r,\, p_r) \\
&= b_{r-1} + \frac{5}{8}\, p_{r-1}\,.
\end{aligned}
$$

Since $b_{r-1}+5p_{r-1}/8 < b_{r-1}+3p_{r-1}/4$, it follows that $FS_{Q_f^{(r)}}(e_i^{(r)}) < FS_{Q_f^{(r)}}(e_i^{(r')})$. By definition of minimum fair share, $MFS_{Q_f^{(r)}}(S_i) < FS_{Q_f^{(r)}}(e_i^{(r')})$, so that $e_i^{(r')}$ is not a bottleneck edge for state $Q_f^{(r)}$.

Assume now that $i \in \{i_f^{(r)}, i_l^{(r)}\}$. By construction of network $G(\mathsf{Sched})$, sesion S_i traverses edge $e_{i_l^{(r)}}^{(r)}$. Claim 6.19 implies that

$$
\begin{aligned}
FS_{Q_f^{(r)}}(e_{i_l^{(r)}}^{(r)}) &= b_r + \frac{p_r}{4} \\
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{16} \\
&\quad \text{(by recursive definition of } b_r,\, p_r) \\
&= b_{r-1} + \frac{9}{16}\, p_{r-1}\,.
\end{aligned}
$$

Since $b_{r-1} + 9p_{r-1}/16 < b_{r-1} + 3p_{r-1}/4$, it follows that $FS_{Q_f^{(r)}}(e_{i_l^{(r)}}^{(r)}) < FS_{Q_f^{(r)}}(e_i^{(r')})$. By definition of minimum fair share, it follows that $MFS_{Q_f^{(r)}}(S_i) < FS_{Q_f^{(r)}}(e_i^{(r')})$, so that $e_i^{(r')}$ is not a bottleneck edge for state $Q_f^{(r)}$.

This completes the proof of (8). We continue to prove (9). We consider seperately each of the sessions active in $Q_f^{(r)}$. We start with session $S_{i_f^{(r)}}$.

By construction of $G(\mathsf{Sched})$, session $i_f^{(r)}$ traverses edge $e_{i_l^{(r)}}^{(r)}$; moreover, for each index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, it traverses edge $e_i^{(r)}$; finally, for each integer r' such that $1 \le r' < r$, $i_f^{(r)}$ traverses edge $e_{i_f^{(r)}}^{(r')}$ and no other edge. We consider seperately each of these edges.

Take first edge $e_{i_l^{(r)}}^{(r)}$. By (3),

$$
\begin{aligned}
FS_{Q_f^{(r)}}(e_{i_l^{(r)}}^{(r)}) &= b_r + \frac{p_r}{4} \\
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{1}{4} \cdot \frac{p_{r-1}}{4} \\
&= b_{r-1} + \frac{9p_{r-1}}{16}\,.
\end{aligned}
$$

However, by (1), $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$, so that

$$\begin{aligned} r_{Q_f^{(r)}}(S_{i_f^{(r)}}) - FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) &= b_{r-1} + \frac{5p_{r-1}}{8} - (b_{r-1} + \frac{9p_{r-1}}{16}) \\ &= \frac{1}{16}\, p_{r-1} \qquad \text{(since } p_r \neq 0\text{)} \\ &\neq 0\,, \end{aligned}$$

so that $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) \neq FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}})$.

Since Alg is bottleneck, it follows that edge $e^{(r)}_{i_l^{(r)}}$ does not cause the termination of session $S_{i_f^{(r)}}$ in state $Q_f^{(r)}$.

Take now edge $e_i^{(r)}$ for any index $i \in \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By (6), $e_i^{(r)}$ is not a bottleneck edge for state $Q_f^{(r)}$. Since Alg is bottleneck, it follows that edge $e_i^{(r)}$ does not cause the termination of session $S_{i_f^{(r)}}$ in state $Q_f^{(r)}$.

Take finally edge $e^{(r')}_{i_f^{(r)}}$ for each integer r' such that $1 \leq r' < r$. By (8), $e^{(r')}_{i_f^{(r)}}$ is not a bottleneck edge for state $Q_f^{(r)}$. Since Alg is bottleneck, it follows that edge $e^{(r')}_{i_f^{(r)}}$ does not cause the termination of session $S_{i_f^{(r)}}$ in state $Q_f^{(r)}$.

Since none of the edges traversed by session $S_{i_f^{(r)}}$ causes its termination in state $Q_f^{(r)}$, it follows that $i_f^{(r)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_f^{(r)})$, as needed.

We continue with session $S_{i_l^{(r)}}$. By construction of network $G(\mathsf{Sched})$, session $S_{i_l^{(r)}}$ traverses edge $e^{(r)}_{i_l^{(r)}}$; moreover, for each integer r' such that $1 \leq r' < r$, $i_l^{(r)}$ traverses edge $e^{(r')}_{i_l^{(r)}}$ and no other edge. We consider seperately each of these edges.

Take first edge $e^{(r)}_{i_l^{(r)}}$. By (3),

$$\begin{aligned} FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) &= b_r + p_r/4 \\ &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{16} \\ &= b_{r-1} + \frac{9p_{r-1}}{16}\,. \end{aligned}$$

However, by (2), $r_{Q_f^{(r)}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$, so that

$$FS_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}) - r_{Q_f^{(r)}}(S_{i_l^{(r)}}) = b_{r-1} + \frac{9p_{r-1}}{16} - (b_{r-1} + \frac{p_{r-1}}{2})$$

$$
\begin{aligned}
&= \frac{p_{r-1}}{16} \qquad \text{(since } p_{r-1} \neq 0\text{)} \\
&\neq 0\,,
\end{aligned}
$$

so that $r_{Q_f^{(r)}}(S_{i_l^{(r)}}) \neq FS_{Q_f^{(r)}}(e_{i_l^{(r)}}^{(r)})$.

Since Alg is bottleneck, it follows that edge $e_{i_l^{(r)}}^{(r)}$ does not cause the termination of session $S_{i_l^{(r)}}$ in state $Q_f^{(r)}$.

Take now edge $e_{i_l^{(r)}}^{(r')}$ for any integer r' such that $1 \leq r' < r$. By (8), edge $e_{i_l^{(r)}}^{(r')}$ is not a bottleneck edge for state $Q_f^{(r)}$. Since Alg is bottleneck, it follows that edge $e_{i_l^{(r)}}^{(r')}$ does not cause the termination of session $S_{i_l^{(r)}}$ in state $Q_f^{(r)}$.

Since none of the edges traversed by session $S_{i_l^{(r)}}$ causes its termination in state $Q_f^{(r)}$, it follows that $i_l^{(r)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_f^{(r)})$, as needed.

We finally consider all remaining sessions.

Fix any session $i \in \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By construction of network $G(\mathsf{Sched})$, session S_i traverses edge $e_i^{(r)}$; moreover, for each integer r' such that $1 \leq r' < r$, S_i traverses edge $e_i^{(r')}$, while S_i traverses also edge $e^{(r'')} \in E^{(r'')}$, for some integer r'' such that $r < r'' \leq d/2$.

By (6), (7) and (8), none of the edges $e_i^{(r)}$, $e_i^{(r')}$ and $e^{(r'')}$ is a bottleneck edge for state $Q_0^{(r)}$. Since Alg is a bottleneck algorithm, it follows that none of these edges cause the termination of session S_i in state $Q_0^{(r)}$, as needed.

The proof of Lemma 6.18 is now complete. ■

We continue to prove that all edges of a lower than rth epoch have available capacity for state $Q_f^{(r)}$.

Lemma 6.23 *For each session index $i \in \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}}$ and any integer r', $1 \leq r' < r$,*

$$
\mathit{residual}_{Q_f^{(r)}}(e_i^{(r')}) \;\geq\; \frac{p_{r-1}}{8}\,.
$$

Proof: By construction of $G(\mathsf{Sched})$, edge $e_i^{(r')}$ is traversed by sessions S_i and $S_{i_f^{(r')}}$. Induction hypothesis (3) implies that $i_f^{(r')} \notin \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}}$, so that $r_{Q_f^{(r)}}(S_{i_f^{(r')}}) = r_{Q^{(r-1)}}(S_{i_f^{(r')}})$. We proceed by case analysis on i.

Assume first that $i \neq i_f^{(r)}$. Lemma 6.18(2) implies that $r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$. By induction hypothesis (2b), $r_{Q^{(r-1)}}(S_i) = b_{r-1} + p_{r-1}/2$, so that $r_{Q_f^{(r)}}(S_i) = r_{\underset{Q^{(r-1)}}{\rightarrow}}(S_i)$. By definition of *residual*,

$$\begin{aligned} \mathit{residual}_{Q_f^{(r)}}(e_i^{(r')}) = &= \mathit{cap}(e_i^{(r')}) - r_{Q_f^{(r)}}(S_i) - r_{Q_f^{(r)}}(S_{i_f^{(r')}}) \\ &= \mathit{cap}(e_i^{(r')}) - r_{Q^{(r-1)}}(S_i) - r_{Q^{(r-1)}}(S_{i_f^{(r')}}) \\ &= \mathit{residual}_{Q^{(r-1)}}(e_i^{(r')})\,. \end{aligned}$$

Induction hypothesis (4a) implies that $\mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) \geq p_{r-1}/4$, so that

$$\begin{aligned} \mathit{residual}_{Q_f^{(r)}}(e_i^{(r')}) &\geq p_{r-1}/4 \\ &> p_{r-1}/8\,, \end{aligned}$$

as needed.

Assume now that $i = i_f^{(r)}$. Induction hypothesis (2b) implies that $r_{Q^{(r-1)}}(S_i) = b_{r-1} + p_{r-1}/2$. By Lemma 6.18 (1),

$$\begin{aligned} r_{Q_f^{(r)}}(S_i) &= b_{r-1} + \frac{5p_{r-1}}{8} \\ &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} \\ &= r_{Q^{(r-1)}}(S_i) + \frac{p_{r-1}}{8}\,, \end{aligned}$$

so that

$$\begin{aligned} \mathit{residual}_{Q_f^{(r)}}(e_i^{(r')}) &= \mathit{cap}(e_i^{(r')}) - r_{Q_f^{(r)}}(S_i) - r_{Q_f^{(r)}}(S_{i_f^{(r')}}) \\ &= \mathit{cap}(e_i^{(r')}) - r_{Q^{(r-1)}}(S_i) - \frac{p_{r-1}}{8} - r_{Q^{(r-1)}}(S_{i_f^{(r')}}) \\ &= \mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) - \frac{p_{r-1}}{8}\,. \end{aligned}$$

Induction hypothesis (4a) implies that $\mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) \geq p_{r-1}/4$, so that

$$\begin{aligned} \mathit{residual}_{Q_f^{(r)}}(e_i^{(r')}) &\geq p_{r-1}/4 - p_{r-1}/8 \\ &= p_{r-1}/8\,, \end{aligned}$$

which completes the proof of the claim. ■

Assume now, by the way of contradiction, that $|S\ |_\alpha\ (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}| = 1$. Clearly, there is no session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}\}$ such that S_i is scheduled for an increase during execution epoch $\alpha^{(l_r)}$.

Since $|S\ |_\alpha\ (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}| = 1$, and $i_f^{(r)} \in S\ |_\alpha\ (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$, it follows that $S\ |_\alpha\ (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\} = \{i_f^{(r)}\}$. Thus, the only active session that is scheduled for an increase after state $Q_f^{(r)}$ is session $S_{i_f^{(r)}}$. We prove that scheduling session $S_{i_f^{(r)}}$ for an increase in any state Q such that $Q_f^{(r)} \xrightarrow{\alpha} Q$ does not cause any change to the rate and status of all active sessions in $\overrightarrow{Q^{(r-1)}}$. Assume that Q, $Q_f^{(r)} \xrightarrow{\alpha} Q$, is the first state that $i_f^{(r)}$ is scheduled for another increase. By construction of network $G(\mathsf{Sched})$, $i_f^{(r)}$ and $i_l^{(r)}$ share the edge $e^{(r)}_{i_l^{(r)}}$ with capacity $cap(e^{(r)}_{i_l^{(r)}}) = 2\,b_r + p_r/2$. Lemma 6.18(1) implies that $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$, while by Lemma 6.18(1), $r_{Q_f^{(r)}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$. Clearly, $r_Q(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$ and $r_Q(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$. However,

$$\begin{aligned} r_Q(S_{i_f^{(r)}}) - r_Q(S_{i_l^{(r)}}) &= b_{r-1} + \frac{5p_{r-1}}{8} - (b_{r-1} + \frac{p_{r-1}}{2}) \\ &= \frac{p_{r-1}}{8} \qquad (\text{since } p_{r-1} \neq 0) \\ &> 0\,, \end{aligned}$$

so that $r_Q(S_{i_l^{(r)}}) < r_Q(S_{i_f^{(r)}})$. Thus, by definition of iupdate function, it follows that:

$$\begin{aligned} \Delta_Q(e^{(r)}_{i_l^{(r)}}, i_l^{(r)}) &= cap(e^{(r)}_{i_l^{(r)}}) - r_Q(S_{i_f^{(r)}}) - r_Q(S_{i_l^{(r)}}) \\ &= 2b_r + \frac{p_r}{2} - \left(b_{r-1} + \frac{5}{8}\,p_{r-1}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\ &= 2\left(b_{r-1} + \frac{p_{r-1}}{2}\right) + \frac{1}{2} \cdot \frac{p_{r-1}}{4} - 2b_{r-1} - p_{r-1} - \frac{p_{r-1}}{8} \\ &= 2b_{r-1} + p_{r-1} + \frac{p_{r-1}}{8} - 2b_{r-1} - p_{r-1} - \frac{p_{r-1}}{8} \\ &= 0\,, \end{aligned}$$

which means that $\Delta_Q(i_l^{(r)}) = 0$. Hence, by definition of update operation, $r_{\overrightarrow{Q}}(S_{i_f^{(r)}}) = r_Q(S_{i_f^{(r)}})$.

The same arguments are applied inductively for any state Q' such that $Q \xrightarrow{\alpha} Q'$ and $i_f^{(r)}$ is scheduled for an increase in Q'. Thus, the rate and status of all sessions remain the same as for state $Q_f^{(r)}$, for the rest of the execution.

Consider any arbitrary session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. We proceed to prove that the rate of session S_i can be increased in state $Q_f^{(r)}$ without causing any decrease to the rate of any other session.

By construction of network $G(\mathsf{Sched})$, S_i traverses edge $e_i^{(r)}$, and for each integer r' such that $1 \leq r' \leq r$, S_i traverses $e_i^{(r')}$, while it also traverses at least one edge $e^{(r'')} \in E^{(r'')}$ for some r'' such that $r < r'' \leq d/2$. We consider seperately each of these edges.

Consider first edge $e_i^{(r)}$. By construction of $G(\mathsf{Sched})$, $cap(e_i^{(r)}) = 2b_r + p_r$. By Lemma 6.18 (2), $r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$, while, by Lemma 6.18 (1), $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$, so that

$$\begin{aligned} residual_{Q_f^{(r)}}(e_i^{(r)}) &= cap(e_i^{(r)}) - r_{Q_f^{(r)}}(S_i) - r_{Q_f^{(r)}}(S_{i_f^{(r)}}) \\ &= 2b_r + p_r - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) - \left(b_{r-1} + \frac{5p_{r-1}}{8}\right) \\ &= 2 \cdot \left(b_{r-1} + \frac{p_{r-1}}{2}\right) + \frac{p_{r-1}}{4} - 2b_{r-1} - p_{r-1} - \frac{p_{r-1}}{8} \\ &\quad \text{(by definition of } b_r \text{ and } p_r) \\ &= \frac{p_{r-1}}{8}. \end{aligned}$$

Thus, by definition of update operation, $\Delta_{Q_f^{(r)}}(e_i^{(r)}, i) \geq p_{r-1}/8$.

Consider next edge $e_i^{(r')}$ for any integer r' such that $1 \leq r' \leq r$. By Lemma 6.23 we have $residual_{Q_f^{(r)}}(e_i^{(r')}) \geq p_{r-1}/8$. Thus, by definition of update operation, $\Delta_{Q_f^{(r)}}(e_i^{(r')}, i) \geq p_{r-1}/8$.

Finally, consider edge $e^{(r'')} \in E^{(r'')}$ for some r'' such that $r < r'' \leq d/2$. Lemma 6.17 implies that

$$residual_{Q^{(r-1)}}(e^{(r'')}) \geq \frac{p_{r-1}}{8}.$$

By construction of network $G(\mathsf{Sched})$, edge $e^{(r'')}$ is traversed by session S_i and also by some other session S_j such that $j \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By Lemma 6.18(2), $r_{Q_f^{(r)}}(S_i) = r_{Q_f^{(r)}}(S_j) = b_{r-1} + p_{r-1}/2$, while by induction hypothesis (2b), $r_{Q^{(r-1)}}(S_i) = r_{Q^{(r-1)}}(S_j) = b_{r-1} + p_{r-1}/2$, so that $r_{Q_f^{(r)}}(S_i) = r_{Q^{(r-1)}}(S_i)$ and $r_{Q_f^{(r)}}(S_j) = r_{Q^{(r-1)}}(S_j)$. Thus, $residual_{Q_f^{(r)}}(e_i^{(r'')}) = residual_{Q^{(r-1)}}(e_i^{(r'')})$, so that $residual_{Q_f^{(r)}}(e_i^{(r'')}) \geq \frac{p_{r-1}}{8}$. Hence, by definition of iupdate function, $\Delta_{Q_f^{(r)}}(e_i^{(r'')}, i) \geq p_{r-1}/8$.

By definition of update operation, $\Delta_{Q_f^{(r)}}(i) \geq p_{r-1}/8$, so that the rate of S_i can be increased without causing any decrease to the rate of any other session. Thus, session S_i has not yet received its max-min rate. A contradiction to max-min fairness.

To complete the proof of (1) we next prove that $S \mid_\alpha (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$ can not be a proper subset of $\mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$. Assume, by the way of contradiction, that $S \mid_\alpha (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\} \subset \mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\}$. We consider session $S_{i_l^{(r)}}$. Clearly,

$S_{i_l^{(r)}}$ is not scheduled for another increase from state $\mathcal{A}_{Q^{(r-1)}}$ on. We prove that session $S_{i_l^{(r)}}$ has not yet received its max-min fair rate.

By construction of network $G(\mathsf{Sched})$, session $S_{i_l^{(r)}}$ shares the edge $e^{(r)}_{i_l^{(r)}}$ with session $i_f^{(r)}$; moreover, for each integer r' such that $1 \leq r' < r$, it traverses edges $e^{(r')}_{i_l^{(r)}}$ and no other edge. We consider seperately each of these edges.

Take first edge $e^{(r)}_{i_l^{(r)}}$. By construction of network $G(\mathsf{Sched})$, $cap(e^{(r)}_{i_l^{(r)}}) = 2b_r + p_r/2$. Moreover, Lemma 6.18 (2) implies that $r_{Q_f^{(r)}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$. Thus, by definition of iupdate function,

$$\begin{aligned}
\Delta_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}, i_l^{(r)}) &= \frac{1}{2} \cdot cap(e^{(r)}_{i_l^{(r)}}) - r_{Q_f^{(r)}}(S_{i_l^{(r)}}) \\
&= \frac{1}{2} \cdot \left(2\, b_r + \frac{p_r}{2}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\
&= b_r + \frac{p_r}{4} - b_{r-1} - \frac{p_{r-1}}{2} \\
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{16} - b_{r-1} - \frac{p_{r-1}}{2} \\
&= \frac{p_{r-1}}{16}\,.
\end{aligned}$$

Take now edge $e^{(r')}_{i_l^{(r)}}$ for each integer r' such that $1 \leq r' < r$. Lemma 6.23 implies that $residual_{Q_f^{(r)}}(e^{(r')}_{i_l^{(r)}}) \geq p_{r-1}/8$. Thus, by definition of the update function

$$\Delta_{Q_f^{(r)}}(e^{(r)}_{i_l^{(r)}}, i_l^{(r)}) \;\geq\; p_{r-1}/8\,.$$

Moreover, by defintion of update operation,

$$\begin{aligned}
\Delta_{Q_f^{(r)}}(i_l^{(r)}) &= \min\{\frac{p_{r-1}}{8}, \frac{p_{r-1}}{16}\} \\
&= \frac{p_{r-1}}{16}\,,
\end{aligned}$$

so that

$$\begin{aligned}
r_{Q_f^{(r)}}(S_{i_l^{(r)}}) + \Delta_{Q_f^{(r)}}(i_l^{(r)}) &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{16} \\
&= b_{r-1} + \frac{9p_{r-1}}{16}\,.
\end{aligned}$$

Lemma 6.18 (1) implies that $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$. Clearly,

$$\begin{aligned}
r_{Q_f^{(r)}}(S_{i_f^{(r)}}) - r_{Q_f^{(r)}}(S_{i_l^{(r)}}) &= b_{r-1} + \frac{5p_{r-1}}{8} - (b_{r-1} + \frac{9p_{r-1}}{16}) \\
&= \frac{p_{r-1}}{16} \qquad (\text{since } p_{r-1} > 0) \\
&> 0\,.
\end{aligned}$$

Thus, by definition of update operation, increasing the rate of session $S_{i_l^{(r)}}$ would cause a decrease to the rate of session $S_{i_f^{(r)}}$ to $b_{r-1} + 9p_{r-1}/16$, while the rate of all other sessions remain the same. Hence, the rate of session $S_{i_l^{(r)}}$ can be increased by $p_{r-1}/16$, without causing a decrease to the rate of some other session with equal or lower rate, so that $r_{Q_f^{(r)}}(S_{i_l^{(r)}})$ is not the max-min rate of session $S_{i_l^{(r)}}$. A contradiction to max-min fairness.

Therefore, we conclude that $S \mid_\alpha (Q^{(r-1)}, \infty) \setminus \{i_f^{(r-1)}, i_l^{(r-1)}\} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, which means that $l_r < \infty$. This completes the proof of (1).

We continue to prove the rest of the claims. Since $l_r < \infty$, define $Q_{ff}^{(r)}$ to be the earliest state such that a session active in state $\overrightarrow{Q^{(r-1)}}$ but different from $i_f^{(r)}$ is scheduled for an increase in front of state $Q_{ff}^{(r)}$ in execution α. Note that since $l_r < \infty$, $Q_{ff}^{(r)}$ precedes $\overrightarrow{Q^{(r)}}$ in the execution epoch $\alpha^{(l_r)}$; moreover, $Q_f^{(r)}$ precedes $Q_{ff}^{(r)}$ in the execution epoch $\alpha^{(l_r)}$. The next simple claim shows that, following state $Q_f^{(r)}$ in the execution α, no change in the rate or status of any active session is possible prior to reaching state $Q_{ff}^{(r)}$.

Lemma 6.24 *Consider any session $S_i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Then, for any state Q such that $\overleftarrow{Q_f^{(r)}} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q_{ff}^{(r)}$, $r_Q(S_i) = r_{Q_f^{(r)}}(S_i)$, and $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}, \mathcal{S}\rangle, Q)$.*

Proof: By definition of $Q_{ff}^{(r)}$, the only active session that can be scheduled in front of state Q is $S_{i_f^{(r)}}$. We proceed by induction on Q.

For the basis case where $Q = Q_f^{(r)}$ it vacously holds that $r_Q(S_i) = r_{Q_f^{(r)}}(S_i)$. Moreover, Lemma 6.18(9) implies that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}, \mathcal{S}\rangle, Q)$, which completes the proof for the basis case.

Assume inductively that for any state $\overleftarrow{Q}$ such that $Q_f^{(r)} \xrightarrow{\alpha} \overleftarrow{Q} \xrightarrow{\alpha} Q_{ff}^{(r)}$, $r_{\overleftarrow{Q}}(S_i) = r_{Q_f^{(r)}}(S_i)$, and $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}, \mathcal{S}\rangle, \overleftarrow{Q})$. For the induction step, consider state Q.

Clearly, the only active session that may be scheduled for an increase in front of state Q is $S_{i_f^{(r)}}$. Notice that if a done session is scheduled for an increase infront of state Q, induction hypothesis implies that the claim holds. Thus, consider the case that session $S_{i_f^{(r)}}$ is scheduled for an increase in front of state Q. By construction of network $G(\mathsf{Sched})$, session $S_{i_f^{(r)}}$ traverses edge $e_{i_l^{(r)}}^{(r)}$. By induction hypothesis $r_{\overleftarrow{Q}}(S_{i_f^{(r)}}) = r_{Q_f^{(r)}}(S_{i_f^{(r)}})$ and $r_{\overleftarrow{Q}}(S_{i_l^{(r)}}) = r_{Q_f^{(r)}}(S_{i_l^{(r)}})$.

Lemma 6.18 (1), (2) imply that $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$, and $r_{Q_f^{(r)}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$, respectively, so that $r^{\leftarrow}_{Q}(S_{i_l^{(r)}}) = b_{r-1} + 5p_{r-1}/8$ and $r^{\leftarrow}_{Q}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$. Thus,

$$\begin{aligned} r^{\leftarrow}_{Q}(S_{i_f^{(r)}}) + r^{\leftarrow}_{Q}(S_{i_l^{(r)}}) &= b_{r-1} + \frac{p_{r-1}}{8} + b_{r-1} + \frac{p_{r-1}}{2} \\ &= 2b_{r-1} + \frac{9}{8}\,p_{r-1} \\ &= 2\left(b_{r-1} + \frac{p-r-1}{2}\right) + \frac{1}{2}\,\frac{p_{r-1}}{4} \\ &= 2b_r + \frac{p_r}{2}\,. \end{aligned}$$

By construction of network $G(\mathsf{Sched})$, $cap(e^{(r)}_{i_l^{(r)}}) = 2b_r + p_2/2$, so that $r^{\leftarrow}_{Q}(S_{i_f^{(r)}}) + r^{\leftarrow}_{Q}(S_{i_l^{(r)}}) = cap(e^{(r)}_{i_l^{(r)}})$. Thus, it follows by definition of update operation, that $\Delta^{\leftarrow}_{Q}(i_f^{(r)}, e^{(r)}_{i_l^{(r)}}) = 0$, so that $\Delta^{\leftarrow}_{Q}(i_f^{(r)}) = 0$. Thus, by definition of update operation, $r_Q(S_i) = r^{\leftarrow}_{Q}(S_i)$ for each session $i \in \mathcal{A}_Q$. However, by induction hypothesis, $\mathcal{A}^{\leftarrow}_{Q} = \mathcal{A}_{Q_f^{(r)}}$ and $r^{\leftarrow}_{Q}(S_i) = r_{Q_f^{(r)}}(S_i)$, so that $r_Q(S_i) = r_{Q_f^{(r)}}(S_i)$. Moreover, since Alg is bottleneck and the termination condition for bottleneck algorithms is a condition that depends solely on session rates, Lemma 6.18 (9) immediately implies that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}, \mathcal{S}\rangle, Q)$, which completes the proof of the induction step. ■

Roughly speaking, the next technical claim describes the "fragment" of the execution epoch $\alpha^{(l_r)}$ starting with scheduling the session right in front of $Q^{(r)}_{ff}$ and ending with the state immediately preceding $Q^{(r)}$, the last state in $\alpha^{(l_r)}$.

Lemma 6.25 *Consider any state Q such that $\overleftarrow{Q^{(r)}_{ff}} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q^{(r)}$. Then, the following hold for state Q:*

(1) *for each session $i \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \cup \{i_f^{(r)}\}$, $r_Q(S_i) = b_r + p_r/2$;*

(2) *for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \setminus \{i_f^{(r)}\}$, $r_Q(S_i) = r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$;*

(3) *edge $e^{(r)}_{i_l^{(r)}}$ is a bottleneck edge for Q;*

(4) *for any session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, edge $e_i^{(r)}$ is not a bottleneck edge for Q;*

(5) *for any session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$ and any integer r' such that $1 \le r' < r$, edge $e_i^{(r')}$ is not a bottleneck edge for Q;*

(6) *for any integer* r' *such that* $r < r' \leq d/2$, *and edge* $e^{(r')} \in E^{(r')}$, $e^{(r')}$ *is not a bottleneck edge for* Q;

(7) *for any session* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\longrightarrow}}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$.

Roughly speaking, conditions (1) and (2) determine rates of active sessions in state Q. Conditions (3), (4), (5) and (6) establish that edge $e^{(r)}_{i^{(r)}_l}$ is the only bottleneck edge for state Q, while condition (7) guarantees that no corresponding termination of sessions takes place. We continue with the proof of Lemma 6.25.

Proof: By induction on Q. We start with the basis case where $Q = Q^{(r)}_{ff}$.

Basis Case (Lemma 6.25)

We start by determining the rates and status of all sessions in state $Q^{(r)}_{ff}$.

Denote $i^{(r)}_{ff}$ the session that is scheduled in front of state $Q^{(r)}_{ff}$. By definition of state $Q^{(r)}_{ff}$, $i^{(r)}_{ff} \neq i^{(r)}_f$, while by definition of state $Q^{(r)}$, $i^{(r)}_{ff} \neq i^{(r)}_l$.

By construction of $G(\mathsf{Sched})$, session $i^{(r)}_{ff}$ traverses the epoch r edge $e^{(r)}_{i^{(r)}_{ff}}$, which is also traversed by session $i^{(r)}_f$. Moreover, by construction of $G(\mathsf{Sched})$, $cap(e^{(r)}_{i^{(r)}_{ff}}) = 2b_r + p_r$. By Lemma 6.24, $r \underset{Q^{(r)}_{ff}}{\leftarrow}(S_{i^{(r)}_f}) = r_{Q^{(r)}_f}(S_{i^{(r)}_f})$ and $r \underset{Q^{(r)}_{ff}}{\leftarrow}(S_{i^{(r)}_{ff}}) = r_{Q^{(r)}_f}(S_{i^{(r)}_{ff}})$. Thus, Lemma 6.18(1) and (2) imply that $r \underset{Q^{(r)}_{ff}}{\leftarrow}(S_{i^{(r)}_f}) = b_{r-1} + 5p_{r-1}/8$ and $r \underset{Q^{(r)}_{ff}}{\leftarrow}(S_{i^{(r)}_{ff}}) = b_{r-1} + p_{r-1}/2$. Thus, by definition of iupdate function,

$$\begin{aligned}
\Delta \underset{Q^{(r)}_{ff}}{\leftarrow}(e^{(r)}_{i^{(r)}_{ff}}, i^{(r)}_{ff}) &= \frac{1}{2}\, cap(e^{(r)}_{i^{(r)}_{ff}}) - r \underset{Q^{(r)}_{ff}}{\leftarrow}(S_{i^{(r)}_{ff}}) && \text{(by construction of network } G(\mathsf{Sched})\\
&= b_r + \frac{p_r}{2} - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) && \text{(by definition of } b_r \text{ and } p_r)\\
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} - b_{r-1} - \frac{p_{r-1}}{2}\\
&= \frac{p_{r-1}}{8}.
\end{aligned}$$

By construction of $G(\mathsf{Sched})$, session $S_{i^{(r)}_{ff}}$ traverses also edge $e^{(r')}_{i^{(r)}_{ff}}$, for every integer r' such that $1 \leq r' < r$. Lemma 6.24 and definition of $\mathit{residual} \underset{Q^{(r)}_{ff}}{\leftarrow}(e^{(r')}_{i^{(r)}_{ff}})$ imply that

$$\mathit{residual} \underset{Q^{(r)}_{ff}}{\leftarrow}(e^{(r')}_{i^{(r)}_{ff}}) \;=\; \mathit{residual}_{Q^{(r)}_f}(e^{(r')}_{i^{(r)}_{ff}}),$$

so that Lemma 6.23 implies that $\mathit{residual}_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(e_{i_{ff}^{(r)}}^{(r')}) \geq p_{r-1}/8$. Thus, $\Delta_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(i_{ff}^{(r)}, e_{i_{ff}^{(r)}}^{(r')}) \geq p_{r-1}/8$.

Finally, session $S_{i_{ff}^{(r)}}$ traverses also some epoch r'' edge $e^{(r'')} \in E^{(r'')}$ for any integer r'' such that $r < r'' \leq d/2$. Moreover, by construction of $G(\mathsf{Sched})$, $e^{(r'')}$ is also traversed by some session S_i such that $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}, i_{ff}^{(r)}\}$. Lemma 6.24 implies that $r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_i) = r_{Q_f^{(r)}}(S_i)$ and $r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_{i_{ff}^{(r)}}) = r_{Q_f^{(r)}}(S_{i_{ff}^{(r)}})$. Since $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}, i_{ff}^{(r)}\}$, Lemma 6.18(2) and induction hypothesis(2b) (of Lemma 6.2) imply that $r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_{i_{ff}^{(r)}}) = r_{Q^{(r-1)}}(S_{i_{ff}^{(r)}})$ and $r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_i) = r_{Q^{(r-1)}}(S_i)$, so that

$$\mathit{residual}_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(e^{(r'')}) \quad = \quad \mathit{residual}_{Q^{(r-1)}}(e^{(r'')})\,.$$

However, Lemma 6.23 implies that

$$\mathit{residual}_{\underset{Q^{(r-1)}}{\rightarrow}}(e^{(r'')}) \quad \geq \quad \frac{p_{r-1}}{8}\,.$$

Thus,

$$\mathit{residual}_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(e^{(r'')}) \quad \geq \quad \frac{p_{r-1}}{8}\,,$$

so that by definition of iupdate function, $\Delta_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(i_{ff}^{(r)}, e^{(r'')}) \geq p_{r-1}/8$. Hence, by definition of update operation,

$$\Delta_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(i_{ff}^{(r)}) \quad = \quad \frac{p_{r-1}}{8}\,,$$

so that

$$\begin{aligned}
r_{Q_{ff}^{(r)}}(S_{i_{ff}^{(r)}}) &= r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_{i_{ff}^{(r)}}) + \Delta_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(i_{ff}^{(r)}) \\
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} \qquad \text{(by recursive definition of } b_r\text{)} \\
&= b_r + \frac{1}{2} \cdot \frac{p_{r-1}}{4} \\
&= b_r + \frac{p_r}{2}\,.
\end{aligned}$$

Lemma 6.18(1) implies that $r_{Q_f^{(r)}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$. By Lemma 6.24, $r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) = r_{Q_f^{(r)}}(S_{i_f^{(r)}})$, so that $r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) = b_{r-1} + 5p_{r-1}/8$. Clearly,

$$r_{\underset{Q_{ff}^{(r)}}{\leftarrow}}(S_{i_f^{(r)}}) - r_{Q_{ff}^{(r)}}(S_{i_{ff}^{(r)}}) \quad = \quad b_{r-1} + \frac{5p_{r-1}}{8} - (b_r + \frac{p_r}{2})$$

$$
\begin{aligned}
& \text{(by recursive definition of } b_r,\ p_r) \\
= \ & b_{r-1} + \frac{5p_{r-1}}{8} - (b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8}) \\
= \ & \frac{5p_{r-1}}{8} - \frac{5p_{r-1}}{8} \\
= \ & 0\,,
\end{aligned}
$$

so that

$$
r_{Q_{ff}^{(r)}}(S_{i_f^{(r)}}) \ = \ b_r + \frac{p_r}{2}\,.
$$

Moreover, for each $i \in \mathcal{A}_{Q^{(r-1}} \setminus \{i_f^{(r)}, i_{ff}^{(r)}\}$, definition of update operation implies that $r_{Q_{ff}^{(r)}}(S_i) = r_{\overleftarrow{Q_{ff}^{(r)}}}(S_i)$. By Lemma 6.24, $r_{\overleftarrow{Q_{ff}^{(r)}}}(S_i) = r_{Q_f^{(r)}}(S_i)$. Lemma 6.18(2) implies that $r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$. Clearly, $\mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \subseteq \{i_f^{(r)}, i_{ff}^{(r)}\}$, which completes the proof of (1) and (2).

We proceed to prove (3).

Claim 6.26 *$e_{i_l^{(r)}}^{(r)}$ is a bottleneck edge for state $Q_{ff}^{(r)}$.*

Proof: By construction of $G(\mathsf{Sched})$, edge $e_{i_l^{(r)}}^{(r)}$ is traversed by sessions $i_f^{(r)}$ and $i_l^{(r)}$. Lemma 6.18(9) implies that both $i_f^{(r)} \in \mathcal{A}_{Q_f^{(r)}}$ and $i_l^{(r)} \in \mathcal{A}_{Q_f^{(r)}}$. By Lemma 6.24 we have that $\mathcal{A}_{Q_{ff}^{(r)}} = \mathcal{A}_{Q_f^{(r)}}$, so that $i_f^{(r)} \in \mathcal{A}_{Q_{ff}^{(r)}}$ and $i_l^{(r)} \in \mathcal{A}_{Q_{ff}^{(r)}}$. It follows that $\mathcal{A}_{Q_{ff}^{(r)}} \mid e_{i_l^{(r)}}^{(r)} = \{i_f^{(r)}, i_l^{(r)}\} \neq \emptyset$. Lemma 6.18(5) implies that edge $e_{i_l^{(r)}}^{(r)}$ is a bottleneck edge for state $Q_f^{(r)}$. Thus, Proposition 4.1 implies that edge $e_{i_l^{(r)}}^{(r)}$ is a bottleneck edge for state $Q_{ff}^{(r)}$, as needed. ■

We next prove (4). We first prove that the set of active sessions does not change between states $\overrightarrow{Q^{(r-1)}}$ and $Q_{ff}^{(r)}$.

Lemma 6.27 $\mathcal{A}_{Q_{ff}^{(r)}} = \mathcal{A}_{Q_f^{(r)}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$.

Proof: Lemma 6.18(6) implies that $\mathcal{A}_{Q_f^{(r)}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. By Lemma 6.24, for any state Q such that $\overleftarrow{Q_f^{(r)}} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q_{ff}^{(r)}$, for each session $i \in \mathcal{A}_{Q_f^{(r)}}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$. It follows that $\mathcal{A}_{Q_{ff}^{(r)}} = \mathcal{A}_{Q_f^{(r)}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, as needed.

■

We continue to calculate the fair share of edge $e^{(r)}_{i_{l_r}}$.

Lemma 6.28 *$FS_{Q^{(r)}_{ff}}(e^{(r)}_{i^{(r)}_l}) = b_r + p_r/4$.*

Proof: By Lemma 6.27, $\mathcal{A}_{Q^{(r)}_{ff}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Hence, both $i^{(r)}_f \in \mathcal{A}_{Q^{(r)}_{ff}}$ and $i^{(r)}_l \in \mathcal{A}_{Q^{(r)}_{ff}}$. It follows that $\mathcal{A}_{Q^{(r)}_{ff}} \mid e^{(r)}_{i^{(r)}_{l_r}} = \{i^{(r)}_f, i^{(r)}_l\}$, so that

$$\begin{aligned} FS_{Q^{(r)}_{ff}}(e^{(r)}_{i^{(r)}_l}) &= \frac{cap(e^{(r)}_{i^{(r)}_l}) - allot_{Q^{(r)}_{ff}}(e^{(r)}_{i^{(r)}_l})}{|\mathcal{A}_{Q^{(r)}_{ff}} \mid e^{(r)}_{i^{(r)}_l}|} \\ &= \frac{cap(e^{(r)}_{i^{(r)}_l})}{2} \\ &= b_r + \frac{p_r}{4} \qquad \text{(by construction of } G(\mathsf{Sched})\text{,} \end{aligned}$$

as needed. ■

We continue to calculate the fair share of other edges of epoch r.

Lemma 6.29 *For any session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i^{(r)}_f, i^{(r)}_l\}$,*

$$FS_{Q^{(r)}_{ff}}(e^{(r)}_i) = b_r + \frac{p_r}{2}\,.$$

Proof: By Lemma 6.27, $\mathcal{A}_{Q^{(r)}_{ff}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Hence, both $i \in \mathcal{A}_{Q^{(r)}_{ff}}$ and $i^{(r)}_f \in \mathcal{A}_{Q^{(r)}_{ff}}$. By construction of $G(\mathsf{Sched})$, edge $e^{(r)}_i$ is traversed by sessions S_i and $S_{i^{(r)}_f}$. It follows that $\mathcal{A}_{Q^{(r)}_{ff}} \mid e^{(r)}_i = \{i, i^{(r)}_f\}$, so that

$$\begin{aligned} FS_{Q^{(r)}_{ff}}(e^{(r)}_i) &= \frac{cap(e^{(r)}_i) - allot_{Q^{(r)}_{ff}}(e^{(r)}_i)}{|\mathcal{A}_{Q^{(r)}_{ff}} \mid e^{(r)}_i|} \\ &= \frac{cap(e^{(r)}_i)}{2} \\ &= b_r + \frac{p_r}{2} \qquad \text{(by construction of } G(\mathsf{Sched})\text{,} \end{aligned}$$

as needed. ■

Claim 6.30 *For any session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, edge $e_i^{(r)}$ is not a bottleneck edge for state $Q_{ff}^{(r)}$.*

Proof: Lemma 6.29 implies that $FS_{Q_{ff}^{(r)}}(e_i^{(r)}) = b_r + p_r/2$. By Lemma 6.28, $FS_{Q_{ff}^{(r)}}(e_{i_l^{(r)}}^{(r)}) = b_r + p_r/4$.

By construction of network $G(\mathsf{Sched})$, session $i_f^{(r)}$ traverses every edge $e_i^{(r)}$, for each session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}\}$. Thus,

$$\begin{aligned} MFS_{Q_{ff}^{(r)}}(S_{i_f^{(r)}}) &\leq \min\{FS_{Q_{ff}^{(r)}}(e_{i_l^{(r)}}^{(r)}), \bigcup_{i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}} \{FS_{Q_{ff}^{(r)}}(e_i^{(r)})\}\} \\ &= \min\{b_r + \frac{p_r}{4}, b_r + \frac{p_r}{2}\} \\ &= b_r + \frac{p_r}{4}\,. \end{aligned}$$

Thus, $S_{i_f^{(r)}} \in \mathcal{A}_{Q_{ff}^{(r)}} \mid e_i^{(r)}$ while

$$\begin{aligned} FS_{Q_{ff}^{(r)}}(e_i^{(r)}) - MFS_{Q_{ff}^{(r)}}(S_{i_f^{(r)}}) &\geq b_r + \frac{p_r}{2} - (b_r + \frac{p_r}{4}) \\ &= \frac{p_r}{4} \\ &\neq 0\,, \end{aligned}$$

which implies that $e_i^{(r)}$ is not a bottleneck edge for state $Q_{ff}^{(r)}$, as needed. ■

We next prove (5).

Claim 6.31 *For any integer r' such that $1 \leq r' < r$ and any session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, edge $e_i^{(r')}$ is not a bottleneck edge for state $Q_{ff}^{(r)}$.*

Proof: By induction hypothesis (4b),

$$FS_{\overrightarrow{Q^{(r-1)}}}(e_i^{(r')})(e_i^{(r')}) \geq b_{r-1} + 3p_{r-1}/4\,.$$

By Lemma 6.27, $\mathcal{A}_{Q_{ff}^{(r)}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Thus, by definition of fair share, $FS_{Q_{ff}^{(r)}} = FS_{\overrightarrow{Q^{(r-1)}}}(e_i^{(r')})$, so that

$$FS_{Q_{ff}^{(r)}}(e_i^{(r')}) \geq b_{r-1} + 3p_{r-1}/4\,.$$

Moreover, by Lemma 6.29

$$\begin{aligned} FS_{Q_{ff}^{(r)}}(e_i^{(r)}) &= b_r + \frac{p_r}{2} \\ &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} \\ &\quad \text{(by definition of } b_r,\ p_r) \\ &= b_{r-1} + \frac{5p_{r-1}}{8}\,. \end{aligned}$$

Clearly,

$$\begin{aligned} FS_{Q_{ff}^{(r)}}(e_i^{(r')}) - FS_{Q_{ff}^{(r)}}(e_i^{(r)}) &\geq b_{r-1} + \frac{3p_{r-1}}{4} - (b_{r-1} + \frac{5p_{r-1}}{8}) \\ &= \frac{p_{r-1}}{8} \\ &> 0\,. \end{aligned}$$

By construction of network $G(\mathsf{Sched})$, session S_i traverses both edges $e_i^{(r)}$ and $e_i^{(r')}$. By definition of minimum fair share, this implies that $MFS_{Q_{ff}^{(r)}}(S_i) < FS_{Q_{ff}^{(r)}}(e_i^{(r')})$, so that edge $e_i^{(r')}$ is not a bottleneck edge for state $Q_{ff}^{(r)}$, as needed. ■

We continue to prove (6).

Claim 6.32 *For any integer r' such that $r < r' \leq d/2$ and edge $e^{(r')} \in E^{(r')}$, edge $e^{(r')}$ is not a bottleneck edge for state $Q_{ff}^{(r)}$.*

Proof: By construction of network $G(\mathsf{Sched})$, $e^{(r')}$ is traversed by some session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, which also traverses edge $e_i^{(r)}$.

We proceed to calculate the fair shares of edges $e_i^{(r)}$ and $e^{(r')}$. Lemma 6.29 implies that

$$\begin{aligned} FS_{Q_{ff}^{(r)}}(e_i^{(r)}) &= b_r + p_r/2 \\ &= \frac{cap(e_i^{(r)})}{2} \qquad \text{(by construction of } G(\mathsf{Sched}))\,. \end{aligned}$$

By construction of $G(\mathsf{Sched})$, edge $e^{(r')}$ is traversed by sessions S_i and $S_{i'}$, for some session index $i' \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. Lemma 6.27 implies that $\mathcal{A}_{Q_{ff}^{(r)}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Hence, both $i \in \mathcal{A}_{Q_{ff}^{(r)}}$

and $i' \in \mathcal{A}_{Q_{ff}^{(r)}}$, so that $\mathcal{A}_{Q_{ff}^{(r)}} \mid e^{(r')} = \{i, i'\}$. Thus,

$$\begin{aligned} FS_{Q_{ff}^{(r)}}(e^{(r')}) &= \frac{cap(e^{r'}) - allot_{Q_{ff}^{(r)}}(e^{(r')})}{|\mathcal{A}_{Q_{ff}^{(r)}} \mid e^{(r')}|} \\ &= \frac{cap(e^{(r')})}{2}. \end{aligned}$$

By Proposition 5.3, $cap(e_i^{(r)}) < cap(e^{(r')})$. It follows that $FS_{Q_{ff}^{(r)}}(e_i^{(r)}) < FS_{Q_{ff}^{(r)}}(e^{(r')})$. By definition of minimum fair share, this implies that $MFS_{Q_{ff}^{(r)}}(S_i) < FS_{Q_{ff}^{(r)}}(e^{(r')})$, so that edge $e^{(r')}$ is not a bottleneck edge for state $Q_{ff}^{(r)}$, as needed. ■

We continue to prove (7). We consider seperately each of the sessions active in $\overrightarrow{Q^{(r-1)}}$. We start with session $S_{i_f^{(r)}}$.

Claim 6.33 $i_f^{(r)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_{ff}^{(r)})$.

Proof: By construction, $i_f^{(r)}$ traverses edge $e_{i_l^{(r)}}^{(r)}$; moreover, for each index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, it traverses edge $e_i^{(r)}$; finally, for each integer r' such that $1 \leq r' < r$, $i_f^{(r)}$ traverses edge $e_{i_f^{(r)}}^{(r')}$ and no other edge. We consider seperately each of these edges.

Take first edge $e_{i_l^{(r)}}^{(r)}$. By construction of $G(\mathsf{Sched})$, edge $e_{i_l^{(r)}}^{(r)}$ is traversed by sessions $S_{i_f^{(r)}}$ and $S_{i_l^{(r)}}$. Lemma 6.28 implies that $FS_{Q_{ff}^{(r)}}(e_{i_l^{(r)}}^{(r)}) = b_r + p_r/4$.

Recall (claim (1) of basis case) that $r_{Q_{ff}^{(r)}}(S_{i_f^{(r)}}) = b_r + p_r/2$, so that

$$\begin{aligned} r_{Q_{ff}^{(r)}}(S_{i_f^{(r)}}) - FS_{Q_{ff}^{(r)}}(e_{i_l^{(r)}}^{(r)}) &= b_r + \frac{p_r}{2} - (b_r + \frac{p_r}{4}) \\ &= \frac{p_r}{4} \\ &\neq 0 \qquad \text{(by definition of } p_r). \end{aligned}$$

Since Alg is bottleneck, it follows that edge $e_{i_l^{(r)}}^{(r)}$ does not cause the termination of session $S_{i_f^{(r)}}$ in state $Q_{ff}^{(r)}$.

Take now the edge $e_i^{(r)}$ for any session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. Claim 6.30 implies that $e_i^{(r)}$ is not a bottleneck edge for state $Q_{ff}^{(r)}$. Since Alg is bottleneck, it follows that edge $e_i^{(r)}$ does not cause the termination of session $S_{i_f^{(r)}}$ in state $Q_{ff}^{(r)}$.

Take finally edge $e^{(r')}_{i_f^{(r)}}$. Claim 6.30 implies that edge $e^{(r')}_{i_f^{(r)}}$ is not a bottleneck edge for state $Q^{(r)}_{ff}$. Since Alg is bottleneck, it follows that edge $e^{(r')}_{i_f^{(r)}}$ does not cause the termination of session $S_{i_f^{(r)}}$ in state $Q^{(r)}_{ff}$.

Since none of the edges traversed by session $S_{i_f^{(r)}}$ causes its termination in state $Q^{(}_{ff}r)$, it follows that $i_f^{(r)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r)}_{ff})$, as needed. ■

We continue with session $S_{i_l^{(r)}}$.

Lemma 6.34 $i_l^{(r)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r)}_{ff})$.

Proof: By construction of $G(\mathsf{Sched})$, session $S_{i_l^{(r)}}$ traverses edge $e^{(r)}_{i_l^{(r)}}$; moreover, for each integer r' such that $1 \leq r' < r$, $i_l^{(r)}$ traverses edge $e^{(r')}_{i_l^{(r)}}$ and no other edge. We consider each of these edges seperately.

Take first edge $e^{(r)}_{i_l^{(r)}}$. By Lemma 6.28, $FS_{Q^{(r)}_{ff}}(e^{(r)}_{i_l^{(r)}}) = b_r + p_r/4$. Recall that $r_{Q^{(r)}_{ff}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$. Thus,

$$\begin{aligned} FS_{Q^{(r)}_{ff}}(e^{(r)}_{i_l^{(r)}}) - r_{Q^{(r)}_{ff}}(S_{i_l^{(r)}}) &= b_r + \frac{p_r}{4} - (b_{r-1} + \frac{p_{r-1}}{4}) \\ &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} - (b_{r-1} + \frac{p_{r-1}}{4}) \\ &\quad \text{(by definition of } b_r \text{ and } p_r) \\ &= \frac{3p_{r-1}}{8} \\ &\neq 0 \qquad \text{(by defintion of } p_{r-1}). \end{aligned}$$

Since Alg is bottleneck, it follows that edge $e^{(r)}_{i_l^{(r)}}$ does not cause the termination of session $S_{i_l^{(r)}}$ in state $Q^{(r)}_{ff}$.

Take now edge $e^{(r')}_{i_l^{(r)}}$ for any integer r' such that $1 \leq r' < r$. By Claim 6.31, edge $e^{(r')}_{i_l^{(r)}}$ is not a bottleneck edge for state $Q^{(r)}_{ff}$. Since Alg is bottleneck, it follows that edge $e^{(r')}_{i_l^{(r)}}$ does not cause the termination of session $S_{i_l^{(r)}}$ in state $Q^{(r)}_{ff}$.

Since none of the edges traversed by session $S_{i_l^{(r)}}$ causes its termination in state $Q^{(r)}_{ff}$, it follows that $i_l^{(r)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r)}_{ff})$, as needed. ■

We finally consider all remaining sessions.

Claim 6.35 *For any session* $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_{ff}^{(r)})$.

Proof: By construction of network $G(\mathsf{Sched})$, session S_i traverses edge $e_i^{(r)}$; moreover, for each integer r' such that $1 \leq r' < r$, session S_i traverses edge $e_i^{(r')}$, while S_i also traverses edge $e^{(r'')} \in E^{(r'')}$, for some integer r'' such that $r < r'' \leq d/2$.

By Claims 6.30, 6.31 and 6.32, respectively, none of the edges $e_i^{(r)}$, $e_i^{(r')}$ and $e^{(r'')}$ is a bottleneck edge for state $Q_{ff}^{(r)}$. Since Alg is bottleneck, it follows that none of these edges causes the termination of session S_i in state $Q_{ff}^{(r)}$, which implies that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q_{ff}^{(r)})$, as needed. ■

The proof of the basis case of Lemma 6.25 is now complete.

Induction Hypothesis (Lemma 6.25)

Assume inductively that for some state Q such that $\overset{\leftarrow\!\!-}{Q}_{ff}^{(r)} \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q^{(r)}$, for each state Q' such that $\overset{\leftarrow\!\!-}{Q}_{ff}^{(r)} \xrightarrow{\alpha} Q' \xrightarrow{\alpha} \overset{\leftarrow}{Q}$, the following hold:

(1) for each session $i \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q') \cup \{i_f^{(r)}\}$, $r_{Q'}(S_i) = b_r + p_r/2$;

(2) for each session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q') \setminus \{i_f^{(r)}\}$, $r_{Q'}(S_i) = r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$;

(3) edge $e_{i_l^{(r)}}^{(r)}$ is a bottleneck edge for state Q';

(4) for any session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, edge $e_i^{(r)}$ is not a bottleneck edge for state Q';

(5) for any session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$ and for any integer r' such that $1 \leq r' < r$, edge $e_i^{(r')}$ is not a bottleneck edge for state Q';

(6) for any integer r' such that $r < r' \leq d/2$, and edge $e^{(r')} \in E^{(r')}$, edge $e^{(r')}$ is not a bottleneck edge for state Q';

(7) for any session $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q')$.

Roughly speaking, we assume that the claim holds for all states starting from state $Q_{ff}^{(r)}$ through but not including state Q, for any distinguished state Q'.

Induction Step (Lemma 6.25)

We prove that the claim holds for state Q, namely:

(1) for each session $i \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \cup \{i_f^{(r)}\}$, $r_Q(S_i) = b_r + p_r/2$;

(2) for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \setminus \{i_f^{(r)}\}$, $r_Q(S_i) = r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$;

(3) edge $e^{(r)}_{i_l^{(r)}}$ is a bottleneck edge for state Q;

(4) for any session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, edge $e_i^{(r)}$ is not a bottleneck edge for state Q;

(5) for any session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$ and for any integer r' such that $1 \leq r' < r$, edge $e_i^{(r')}$ is not a bottleneck edge for state Q;

(6) for any integer r' such that $r < r' \leq d/2$, and edge $e^{(r')} \in E^{(r')}$, edge $e^{(r')}$ is not a bottleneck edge for state Q;

(7) for any session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$.

Denote by i_Q the session scheduled in front of state Q. We proceed by case analysis on i_Q.

Assume first that $i_Q \notin \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Then the claims hold trivially.

Assume next that $i_Q = i_f^{(r)}$. Since $i_f^{(r)}$ is scheduled in front of state $Q_f^{(r)}$ and $Q_f^{(r)} \xrightarrow{\alpha} Q$, it follows that $\mathcal{S} \mid \alpha(Q_f^{(r)}, Q) = \mathcal{S} \mid \alpha(Q_f^{(r)}, \overleftarrow{Q})$, so that also $\mathcal{A}_{Q_f^{(r)}} \setminus \mathcal{S} \mid \alpha(Q_f^{(r)}, Q) = \mathcal{A}_{Q_f^{(r)}} \setminus \mathcal{S} \mid \alpha(Q_f^{(r)}, \overleftarrow{Q})$.

Consider session $i_{ff}^{(r)}$. By construction of network $G(\mathsf{Sched})$, sessions $i_{ff}^{(r)}$ and $i_f^{(r)}$ share edge $e^{(r)}_{i_{ff}^{(r)}}$ with $cap(e^{(r)}_{i_{ff}^{(r)}}) = 2b_r + p_r$. Clearly, both $i_{ff}^{(r)} \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$ and $i_f^{(r)} \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$. Thus, by induction hypothesis (1), both $r_{\overleftarrow{Q}}(S_{i_f^{(r)}}) = b_r + p_r/2$ and $r_{\overleftarrow{Q}}(S_i) = b_r + p_r/2$, so that $r_{\overleftarrow{Q}}(S_i) + r_{\overleftarrow{Q}}(S_{i_f^{(r)}}) = 2b_r + p_r/2 = cap(e^{(r)}_{i_{ff}^{(r)}})$. By definition of iupdate function, it follows that $\Delta_{\overleftarrow{Q}}(i_f^{(r)}, e^{(r)}_{i_{ff}^{(r)}}) = 0$, so that also $\Delta_{\overleftarrow{Q}}(i_f^{(r)}) = 0$. Moreover, by definition of update operation, this implies that for each session $i \in \mathcal{A}_{\overleftarrow{Q}}$, $r_Q(S_i) = r_{\overleftarrow{Q}}(S_i)$. Induction hypothesis (7) implies that $\mathcal{A}_{\overleftarrow{Q}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Thus, induction hypothesis (1) and (2) imply that for each session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $r_Q(S_i) = b_r + p_r/2$ if $i \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \cup \{i_f^{(r)}\}$, or $r_Q(S_i) = r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$ if $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \setminus \{i_f^{(r)}\}$. This establishes (1) and (2).

Assume now that $i_Q \neq i_f^{(r)}$. Since $Q \xrightarrow{\alpha} Q^{(r)}$ it follows that $i_Q \neq i_l^{(r)}$. There are two cases to consider.

Assume first that $i_Q \in \mathcal{S} \mid_\alpha (Q_0^{(r)}, \overleftarrow{Q}) \setminus \{i_f^{(r)}\}$; that is, i_Q has been scheduled for an increase in front of at least one state up to and including state $\overleftarrow{Q}$. It follows that $\mathcal{S} \mid \alpha(Q_f^{(r)}, Q) = \mathcal{S} \mid \alpha(Q_f^{(r)}, \overleftarrow{Q})$, so that also $\mathcal{A}_{Q_f^{(r)}} \setminus \mathcal{S} \mid \alpha(Q_f^{(r)}, Q) = \mathcal{A}_{Q_f^{(r)}} \setminus \mathcal{S} \mid \alpha(Q_f^{(r)}, \overleftarrow{Q})$.

By construction of network $G(\mathsf{Sched})$, i_Q and $i_f^{(r)}$ share edge $e_{i_Q}^{(r)}$ with capacity $cap(e_{i_Q}^{(r)}) = 2b_r + p_r$. Clearly, both $i_Q \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$ and $i_f^{(r)} \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$. Thus, by induction hypothesis (1), both $r_{\overleftarrow{Q}}(S_{i_Q}) = b_r + p_r/2$ and $r_{\overleftarrow{Q}}(S_{i_f^{(r)}}) = b_r + p_r/2$, so that $r_{\overleftarrow{Q}}(S_{i_Q}) + r_{\overleftarrow{Q}}(S_{i_f^{(r)}}) = 2b_r + p_r/2 = cap(e_{i_Q}^{(r)})$. By definition of iupdate function, it follows that $\Delta_{\overleftarrow{Q}}(i_Q, e_{i_Q}^{(r)}) = 0$, so that also $\Delta_{\overleftarrow{Q}}(i_Q) = 0$. Moreover, by definition of update operation, this implies that for each session $i \in \mathcal{A}_{\overleftarrow{Q}}$, $r_Q(S_i) = r_{\overleftarrow{Q}}(S_i)$. Induction hypothesis (7) implies that $\mathcal{A}_{\overleftarrow{Q}} = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Thus, induction hypothesis (1) and (2) imply that for each session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $r_Q(S_i) = b_r + p_r/2$ if $i \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \cup \{i_f^{(r)}\}$, or $r_Q(S_i) = r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$ if $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \setminus \{i_f^{(r)}\}$. This establishes (1) and (2).

Assume finally that $i_Q \notin \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$. By construction of network $G(\mathsf{Sched})$, sessions i_Q and $i_f^{(r)}$ share edge $e_{i_Q}^{(r)}$ with capacity $cap(e_{i_Q}^{(r)}) = 2b_r + p_r$. Since $i_Q \notin \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$ while $i_f^{(r)} \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$, induction hypothesis (1) and (2) imply that $r_{\overleftarrow{Q}}(S_{i_Q}) = b_{r-1} + p_{r-1}/2$ and $r_{\overleftarrow{Q}}(S_{i_f^{(r)}}) = b_r + p_r/2$, respectively. Thus, by definition of update operation,

$$\begin{aligned}
\Delta_{\overleftarrow{Q}}(e_{i_Q}^{(r)}, i_Q) &= cap(e_{i_Q}^{(r)}) - r_{\overleftarrow{Q}}(S_{i_f^{(r)}}) - r_{\overleftarrow{Q}}(S_{i_Q}) \\
&= cap(e_{i_Q}^{(r)}) - \left(b_r + \frac{p_r}{2}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\
&= 2b_r + p_r - \left(b_r + \frac{p_r}{2}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\
&= 2b_r + p_r - b_r - \frac{p_r}{2} - b_r \\
&= \frac{p_r}{2}\,.
\end{aligned}$$

By construction of network $G(\mathsf{Sched})$, session S_{i_Q} traverses also some epoch r' edge $e_{i_Q}^{(r')}$, for any integer $1 \leq r' < r$. Moreover, by construction of $G(\mathsf{Sched})$, edge $e_{i_Q}^{(r')}$ is also traversed by session $i_f^{(r')} \notin \{i_f^{(r)}, i_l^{(r)}\}$. By induction hypothesis(4a) (of Lemma 6.2),

$$residual_{Q^{(r-1)}}(e_{i_Q}^{(r')}) \geq \frac{p_{r-1}}{4}\,.$$

By induction hypothesis (3) (Lemma 6.2), $i_f^{(r')} \notin \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, so that $i_f^{(r')} \notin \mathcal{A}_{\overleftarrow{Q}}$. By induction hypothesis (2) (Lemma 6.2), $r_{Q^{(r-1)}}(S_{i_Q}) = b_{r-1} + p_{r-1}/2$, while by induction hypothesis (2), $r_{\overleftarrow{Q}}(S_{i_Q}) = b_{r-1} + p_{r-1}/2$, so that $r_{Q^{(r-1)}}(S_{i_Q}) = r_{\overleftarrow{Q}}(S_{i_Q})$. Thus, $residual_{Q^{(r-1)}}(e_{i_Q}^{(r')}) = residual_{\overleftarrow{Q}}(e_{i_Q}^{(r')}) \geq p_{r-1}/4$. Thus, by definition of update operation, $\Delta_{\overleftarrow{Q}}(e_{i_Q}^{(r')}, i_Q) \geq p_{r-1}/4 = p_r$.

Finally, i_Q traverses at least one edge $e^{(r'')} \in E^{(r'')}$, for some integer r'' such that $r < r'' \leq d/2$. By Lemma 6.17, $residual_{Q^{(r-1)}}(e^{(r'')}) \geq p_{r-1}/8$. By construction of network $G(\mathsf{Sched})$, $e^{(r'')}$ is traversed by S_{i_Q} and some other session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. We proceed by case analysis.

Assume first that $S_i \notin \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$. Induction hypothesis(2) implies that $r_{\overleftarrow{Q}}(S_i) = b_{r-1} + p_{r-1}/2$. Induction hypothesis(2) (of Lemma 6.2) implies that $r_{Q^{(r-1)}}(S_i) = b_{r-1} + p_{r-1}/2$, so that $r_{\overleftarrow{Q}}(S_i) = r_{Q^{(r-1)}}(S_i)$. Moreover, since $S_{i_Q} \notin \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$, induction hypothesis (2) implies that $r_{\overleftarrow{Q}}(S_{i_Q}) = b_{r-1} + p_{r-1}/2$. Induction hypothesis(2) (of Lemma 6.2) implies that $r_{Q^{(r-1)}}(S_{i_Q}) = b_{r-1} + p_{r-1}/2$, so that Lemma 6.24 implies that $r_{\overleftarrow{Q}}(S_{i_Q}) = r_{Q^{(r-1)}}(S_{i_Q})$. Thus, $residual_{\overleftarrow{Q}}(e^{(r'')}) = residual_{Q^{(r-1)}}(e^{(r'')})$, so that $residual_{\overleftarrow{Q}}(e^{(r'')} \geq p_{r-1}/8$. By definition of update operation, $\Delta_{\overleftarrow{Q}}(e^{(r'')}, i_Q) \geq p_{r-1}/8$.

Assume now that $S_i \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q})$. Induction hypothesis (1) implies that $r_{\overleftarrow{Q}}(S_i) = b_r + p_r/2$. By definition of update operation,

$$\begin{aligned}
\Delta_{\overleftarrow{Q}}(e^{(r'')}, i_Q) &= cap(e^{(r'')}) - r_{\overleftarrow{Q}}(S_i) - r_{\overleftarrow{Q}}(S_{i_Q}) \\
&= cap(e^{(r'')}) - \left(b_r + \frac{p_r}{2}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) .
\end{aligned}$$

Recall that

$$\Delta_{\overleftarrow{Q}}(e_{i_Q}^{(r)}, i_Q) = cap(e_{i_Q}^{(r)}) - \left(b_r + \frac{p_r}{2}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) .$$

By Proposition 5.3, $cap(e^{(r'')}) > cap(e_{i_Q}^{(r)})$, so that

$$cap(e^{(r'')}) - (b_r + p_r/2) - (b_{r-1} + p_{r-1}/2) > cap(e_{i_Q}^{(r)}) - (b_r + p_r/2) - (b_{r-1} + p_{r-1}/2) .$$

Thus,

$$\begin{aligned}
\Delta_{\overleftarrow{Q}}(e^{(r'')}, i_Q) &> \Delta_{\overleftarrow{Q}}(e_{i_Q}^{(r)}, i_Q) \\
&= p_r/2 .
\end{aligned}$$

Hence, by definition of update operation, $\Delta_{\overleftarrow{Q}}(i_Q) = p_r/2$. Thus, definition of update operation implies that $r_Q(S_{i_Q}) = b_{r-1} + p_{r-1}/2 + p_r/2 = b_r + p_r/2$. Moreover, by induction hypothesis, for each $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, either $r_{\overleftarrow{Q}}(S_i) = b_r + p_r/2$ or $r_{\overleftarrow{Q}}(S_i) = b_{r-1} + p_{r-1}/2$. Clearly,

$$\begin{aligned} b_r + \frac{p_r}{2} - (b_{r-1} + \frac{p_{r-1}}{2}) &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} - (b_{r-1} + \frac{p_{r-1}}{2}) \\ &= \frac{p_{r-1}}{8} \\ &> 0 \qquad \text{(by definition of } p_{r-1}\text{)}, \end{aligned}$$

so that $r_{\overleftarrow{Q}}(S_i) < r_Q(S_{i_Q})$. Thus, by definition of update operation, it follows that for each session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_Q\}$, $r_Q(S_i) = r_{\overleftarrow{Q}}(S_i)$.

Cleary, $\mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) = \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q}) \cup \{i_Q\}$, so that $r_Q(S_i) = b_r + p_r/2$ if $i \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \cup \{i_f^{(r)}\}$, and $r_Q(S_i) = r_{Q_f^{(r)}}(S_i) = b_{r-1} + p_{r-1}/2$ if $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \mathcal{S} \mid_\alpha (Q_f^{(r)}, Q) \setminus \{i_f^{(r)}\}$. This establishes (1) and (2).

We continue to prove (3).

Claim 6.36 *$e^{(r)}_{i_l^{(r)}}$ is a bottleneck edge for Q.*

Proof: By construction of $G(\mathsf{Sched})$, edge $e^{(r)}_{i_l^{(r)}}$ is traversed by sessions $i_f^{(r)}$ and $i_l^{(r)}$. Induction hypothesis (7) implies that $\mathcal{A}_Q = \mathcal{A}_{\overleftarrow{Q}}$, so that both $i_f^{(r)} \in \mathcal{A}_Q$ and $i_l^{(r)} \in \mathcal{A}_Q$, so that $\mathcal{A}_Q \mid e^{(r)}_{i_l^{(r)}} = \{i_f^{(r)}, i_l^{(r)}\} \neq \emptyset$. Induction hypothesis (3) implies that edge $e^{(r)}_{i_l^{(r)}}$ is a bottleneck edge for state $\overleftarrow{Q}$. Thus, Proposition 4.1 implies that edge $e^{(r)}_{i_l^{(r)}}$ is a bottleneck edge for state Q, as needed. ■

We continue to calculate the fair share of edge $e^{(r)}_{i_{l_r}}$ for state Q.

Lemma 6.37 *$FS_Q(e^{(r)}_{i_l^{(r)}}) = b_r + p_r/4$.*

Proof: By induction hypothesis (7), $\mathcal{A}_Q = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Hence, both $i_f^{(r)} \in \mathcal{A}_{Q_{ff}^{(r)}}$ and $i_l^{(r)} \in \mathcal{A}_Q$. It follows that $\mathcal{A}_Q \mid e^{(r)}_{i_{l_r}^{(r)}} = \{i_f^{(r)}, i_l^{(r)}\}$, so that

$$FS_Q(e^{(r)}_{i_l^{(r)}}) = \frac{cap(e^{(r)}_{i_l^{(r)}}) - allot_Q(e^{(r)}_{i_l^{(r)}})}{|\mathcal{A}_Q \mid e^{(r)}_{i_l^{(r)}}|}$$

$$
\begin{aligned}
&= \frac{cap(e^{(r)}_{i^{(r)}_l})}{2} \\
&= b_r + \frac{p_r}{4} \qquad \text{(by construction of } G(\mathsf{Sched})\,,
\end{aligned}
$$

as needed. ■

We continue to calculate the fair share of other edges of epoch r for state Q.

Lemma 6.38 *For any session index $i \in \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}} \setminus \{i^{(r)}_f, i^{(r)}_l\}$,*

$$FS_Q(e^{(r)}_i) \;=\; b_r + \frac{p_r}{2}\,.$$

Proof: Induction hypothesis (7) implies that $\mathcal{A}_Q = \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}}$. Hence, both $i \in \mathcal{A}_Q$ and $i^{(r)}_f \in \mathcal{A}_Q$. By construction of $G(\mathsf{Sched})$, edge $e^{(r)}_i$ is traversed by sessions S_i and $S_{i^{(r)}_f}$. It follows that $\mathcal{A}_Q \mid e^{(r)}_i = \{i, i^{(r)}_f\}$, so that

$$
\begin{aligned}
FS_Q(e^{(r)}_i) &= \frac{cap(e^{(r)}_i) - allot_Q(e^{(r)}_i)}{|\mathcal{A}_Q \mid e^{(r)}_i|} \\
&= \frac{cap(e^{(r)}_i)}{2} \\
&= b_r + \frac{p_r}{2} \qquad \text{(by construction of } G(\mathsf{Sched})\,,
\end{aligned}
$$

as needed. ■

We next prove (4).

Claim 6.39 *For any session index $i \in \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}} \setminus \{i^{(r)}_f, i^{(r)}_l\}$, edge $e^{(r)}_i$ is not a bottleneck edge for Q.*

Proof: Lemma 6.38 implies that $FS_Q(e^{(r)}_i) = b_r + p_r/2$, while by Lemma 6.37, $FS_Q(e^{(r)}_{i^{(r)}_l}) = b_r + p_r/4$.

By construction of network $G(\mathsf{Sched})$, session $i^{(r)}_f$ traverses every edge $e^{(r)}_i$, for each session index $i \in \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}} \setminus \{i^{(r)}_f\}$, and no other higher epoch edge. Thus,

$$
\begin{aligned}
MFS_Q(S_{i^{(r)}_f}) &\leq \min\{FS_Q(e^{(r)}_{i^{(r)}_l}), \bigcup_{i \in \mathcal{A}_{\stackrel{\rightarrow}{Q^{(r-1)}}} \setminus \{i^{(r)}_f, i^{(r)}_l\}} \{FS_Q(e^{(r)}_i)\}\} \\
&= \min\{b_r + \frac{p_r}{4}, b_r + \frac{p_r}{2}\} \\
&= b_r + \frac{p_r}{4}\,.
\end{aligned}
$$

Thus, $S_{i_f^{(r)}} \in \mathcal{A}_Q \mid e_i^{(r)}$ while

$$\begin{aligned} FS_Q(e_i^{(r)}) - MFS_Q(S_{i_f^{(r)}}) &= b_r + \frac{p_r}{2} - (b_r + \frac{p_r}{4}) \\ &= \frac{p_r}{4} \\ &\neq 0\,, \end{aligned}$$

which implies that $e_i^{(r)}$ is not a bottleneck edge for state Q, as needed. ■

We continue to prove (5).

Claim 6.40 *For any integer r' such that $r < r' \leq d/2$, edge $e_i^{(r')} \in E^{(r')}$ is not a bottleneck edge for state Q.*

Proof: By construction of network $G(\mathsf{Sched})$, $e^{(r')}$ is traversed by some session $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, which also traverses edge $e_i^{(r)}$.

We proceed to calculate the fair shares of edges $e_i^{(r)}$ and $e^{(r')}$. Lemma 6.38 implies that

$$\begin{aligned} FS_Q(e_i^{(r)}) &= b_r + p_r/2 \\ &= \frac{cap(e_i^{(r)})}{2} \qquad \text{(by construction of } G(\mathsf{Sched}))\,. \end{aligned}$$

By construction of $G(\mathsf{Sched})$, edge $e^{(r')}$ is traversed by sessions S_i and $S_{i'}$, for some session index $i' \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. Induction hypothesis (7) implies that $\mathcal{A}_Q = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Hence, both $i \in \mathcal{A}_Q$ and $i' \in \mathcal{A}_Q$, so that $\mathcal{A}_Q \mid e^{(r')} = \{i, i'\}$. Thus,

$$\begin{aligned} FS_Q(e^{(r')}) &= \frac{cap(e^{r'}) - allot_Q(e^{(r')})}{|\mathcal{A}_Q \mid e^{(r')}|} \\ &= \frac{cap(e^{(r')})}{2}\,. \end{aligned}$$

By Proposition 5.3, $cap(e_i^{(r)}) < cap(e^{(r')})$. It follows that $FS_Q(e_i^{(r)}) < FS_Q(e^{(r')})$. By definition of minimum fair share, this implies that $MFS_Q(S_i) < FS_Q(e^{(r')})$, so that edge $e^{(r')}$ is not a bottleneck edge for state Q, as needed. ■

We next prove (6).

Claim 6.41 *For any session index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$ and for any integer r' such that $1 \leq r' < r$, edge $e_i^{(r')}$ is not a bottleneck edge for state Q.*

Proof: By induction hypothesis(4b) (of Lemma 6.2), $FS_{\overrightarrow{Q^{(r-1)}}}(e_i^{(r')}) \geq b_{r-1} + 3p_{r-1}/4$. By induction hypothesis (7), $\mathcal{A}_Q = \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Thus, by definition of fair share, $FS_Q(e_i^{(r')} = FS_{\overrightarrow{Q^{(r-1)}}}(e_i^{(r')})$, so that $FS_Q(e_i^{(r')}) \geq b_{r-1} + 3p_{r-1}/4$. Moreover, by Lemma 6.38,

$$\begin{aligned} FS_Q(e_i^{(r)}) &= b_r + \frac{p_r}{2} \\ &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} \\ &\quad \text{(by recursive definition of } b_r \text{ and } p_r) \\ &= b_{r-1} + \frac{5p_{r-1}}{8}. \end{aligned}$$

Clearly,

$$\begin{aligned} FS_Q(e_i^{(r')}) - FS_Q(e_i^{(r)}) &\geq b_{r-1} + \frac{3p_{r-1}}{4} - (b_{r-1} + \frac{5p_{r-1}}{8}) \\ &= \frac{p_{r-1}}{8} \\ &> 0. \end{aligned}$$

By construction of network $G(\mathsf{Sched})$, session S_i traverses both edges $e_i^{(r)}$ and $e_i^{(r')}$. By definition of minimum fair share, this implies that $MFS_Q(S_i) < FS_Q(e_i^{(r')})$, so that edge $e_i^{(r')}$ is not a bottleneck edge for state Q, as needed. ■

We continue to prove (7). We consider seperately each of the sessions active in $\overrightarrow{Q^{(r-1)}}$. We start with session $S_{i_f^{(r)}}$.

By construction, $i_f^{(r)}$ traverses edge $e_{i_l^{(r)}}^{(r)}$; moreover, for each index $i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, it traverses edge $e_i^{(r)}$; finally, for each integer r' such that $1 \leq r' < r$, $i_f^{(r)}$ traverses edge $e_{i_f^{(r)}}^{(r')}$ and no other edge. We consider seperately each of these edges.

Take first edge $e_{i_l^{(r)}}^{(r)}$. By construction of $G(\mathsf{Sched})$, edge $e_{i_l^{(r)}}^{(r)}$ is traversed by sessions $S_{i_f^{(r)}}$ and $S_{i_l^{(r)}}$. Lemma 6.37 implies that $FS_Q(e_{i_l^{(r)}}^{(r)}) = b_r + p_r/4$.

By (1), $r_Q(S_{i_f^{(r)}}) = b_r + p_r/2$, so that

$$r_Q(S_{i_f^{(r)}}) - FS_Q(e_{i_l^{(r)}}^{(r)}) = b_r + \frac{p_r}{2} - (b_r + \frac{p_r}{4})$$

$$
\begin{aligned}
&= \frac{p_r}{4} \\
&\neq 0 \qquad \text{(by definition of } p_r\text{)}.
\end{aligned}
$$

Since Alg is bottleneck, it follows that edge $e^{(r)}_{i^{(r)}_l}$ does not cause the termination of session $S_{i^{(r)}_f}$ in state Q.

Take now the edge $e^{(r)}_i$ for any session index $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i^{(r)}_f, i^{(r)}_l\}$. Claim 6.39 implies that $e^{(r)}_i$ is not a bottleneck edge for state Q. Since Alg is bottleneck, it follows that edge $e^{(r)}_i$ does not cause the termination of session $S_{i^{(r)}_f}$ in state Q.

Take finally edge $e^{(r')}_{i^{(r)}_f}$. Claim 6.39 implies that edge $e^{(r')}_{i^{(r)}_f}$ is not a bottleneck edge for state Q. Since Alg is bottleneck, it follows that edge $e^{(r')}_{i^{(r)}_f}$ does not cause the termination of session $S_{i^{(r)}_f}$ in state Q.

Since none of the edges traversed by session $S_{i^{(r)}_f}$ causes its termination in state Q, it follows that $i^{(r)}_f \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q)$, as needed.

We continue with session $S_{i^{(r)}_l}$. By construction of $G(\mathsf{Sched})$, session $S_{i^{(r)}_l}$ traverses edge $e^{(r)}_{i^{(r)}_l}$; moreover, for each integer r' such that $1 \leq r' < r$, $i^{(r)}_l$ traverses edge $e^{(r')}_{i^{(r)}_l}$ and no other edge. We consider each of these edges seperately.

Take first edge $e^{(r)}_{i^{(r)}_l}$. By Lemma 6.37, $FS_Q(e^{(r)}_{i^{(r)}_l}) = b_r + p_r/4$. By (2), $r_Q(S_{i^{(r)}_l}) = b_{r-1} + p_{r-1}/2$. Thus,

$$
\begin{aligned}
FS_Q(e^{(r)}_{i^{(r)}_l}) - r_Q(S_{i^{(r)}_l}) &= b_r + \frac{p_r}{4} - (b_{r-1} + \frac{p_{r-1}}{4}) \\
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} - (b_{r-1} + \frac{p_{r-1}}{4}) \\
&\quad \text{(by definition of } b_r \text{ and } p_r\text{)} \\
&= \frac{3p_{r-1}}{8} \\
&\neq 0 \qquad \text{(by definition of } p_{r-1}\text{)}.
\end{aligned}
$$

Since Alg is bottleneck, it follows that edge $e^{(r)}_{i^{(r)}_l}$ does not cause the termination of session $S_{i^{(r)}_l}$ in state Q.

Take now edge $e^{(r')}_{i^{(r)}_l}$ for any integer r' such that $1 \leq r' < r$. By Claim 6.41, edge $e^{(r')}_{i^{(r)}_l}$ is not a bottleneck edge for state Q. Since Alg is bottleneck, it follows that edge $e^{(r')}_{i^{(r)}_l}$ does not cause the termination of session $S_{i^{(r)}_l}$ in state Q.

Since none of the edges traversed by session $S_{i_l^{(r)}}$ causes its termination in state Q, it follows that $i_l^{(r)} \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q)$, as needed.

We finally consider all remaining sessions. By construction of network $G(\mathsf{Sched})$, session S_i traverses edge $e_i^{(r)}$; moreover, for each integer r' such that $1 \leq r' < r$, session S_i traverses edge $e_i^{(r')}$, while S_i also traverses edge $e^{(r'')} \in E^{(r'')}$, for some integer r'' such that $r < r'' \leq d/2$.

By Claims 6.39, 6.41 and 6.40, respectively, none of the edges $e_i^{(r)}$, $e_i^{(r')}$ and $e^{(r'')}$ is a bottleneck edge for state Q. Since Alg is bottleneck, it follows that none of these edges causes the termination of session S_i in state Q, which implies that $i \notin \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S} \rangle, Q)$, as needed.

At this point we complete the proof of Lemma 6.25. ■

So far, we have determined the rates and status of all sessions till state $\overleftarrow{Q^{(r)}}$. Consider now state $Q^{(r)}$. Clearly, session $S_{i_l^{(r)}}$ is scheduled in front of state $Q^{(r)}$. By construction of $G(\mathsf{Sched})$, session $S_{i_l^{(r)}}$ traverses edge $e^{(r)}_{i_l^{(r)}}$; moreover, for each index $1 \leq r' < r$, $S_{i_l^{(r)}}$ traverses edge $e^{(r')}_{i_l^{(r)}}$ and no other edge. We consider each of these edges seperately.

Take first edge $e^{(r)}_{i_l^{(r)}}$. By construction of $G(\mathsf{Sched})$, edge $e^{(r)}_{i_l^{(r)}}$ is traversed by sessions $i_f^{(r)}$ and $i_l^{(r)}$. Moreover, by construction of $G(\mathsf{Sched})$, $cap(e^{(r)}_{i_l^{(r)}}) = 2b_r + p_r/2$. Since $i_f^{(r)} \in \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q^{(r)}})$, Lemma 6.25 (1) implies that $r_{\overleftarrow{Q^{(r)}}}(S_{i_f^{(r)}}) = b_r + p_r/2$. Clearly, $S_{i_l^{(r)}} \notin \mathcal{S} \mid_\alpha (Q_f^{(r)}, \overleftarrow{Q^{(r)}})$, so that Lemma 6.25 (2) implies that $r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$. Thus,

$$\begin{aligned}
\mathit{residual}_{\overleftarrow{Q^{(r)}}}(e^{(r)}_{i_l^{(r)}}) &= cap(e^{(r)}_{i_l^{(r)}}) - r_{\overleftarrow{Q^{(r)}}}(S_{i_f^{(r)}}) - r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}}) \\
&= 2b_r + \frac{p_r}{2} - \left(b_r + \frac{p_r}{2}\right) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\
&= 2b_r + \frac{p_r}{2} - b_r - \frac{p_r}{2} - b_r \\
&\quad \text{(by recursive definition of } b_r\text{)} \\
&= 0\,.
\end{aligned}$$

Clearly,

$$\begin{aligned}
r_{\overleftarrow{Q^{(r)}}}(S_{i_f^{(r)}}) - r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}}) &= b_r + \frac{p_r}{2} - (b_{r-1} + \frac{p_{r-1}}{2}) \\
&= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_{r-1}}{8} - (b_{r-1} + \frac{p_{r-1}}{2}) \\
&\quad \text{(by recursive definition of } b_r,\ p_r\text{)}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{p_{r-1}}{8} \\
&> 0 \qquad\qquad \text{(by definition of } p_{r-1}\text{)}.
\end{aligned}
$$

Thus, by definition of iupdate function,

$$
\begin{aligned}
\Delta_{\overleftarrow{Q^{(r)}}}(e^{(r)}_{i_l^{(r)}}, i_l^{(r)}) &= \frac{1}{2}\, cap(e^{(r)}_{i_l^{(r)}}) - r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}}) \\
&= \frac{1}{2}\,(2b_r + \frac{p_r}{2}) - \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\
&= b_r + \frac{p_r}{4} - b_r \\
&\quad \text{(by recursive definition of } b_r\text{)} \\
&= \frac{p_r}{4} \\
&= \frac{1}{4} \cdot \frac{p_r}{4} \\
&\quad \text{(by recursive definition of } p_r\text{)} \\
&= \frac{p_{r-1}}{16}\,.
\end{aligned}
$$

Consider now edge $e^{(r')}_{i_l^{(r)}}$, for each r' such that $1 \leq r' < r$. By construction of network $G(\mathsf{Sched})$, $e^{(r')}_{i_l{}^{(r)}}$ is traversed by sessions $i_l^{(r)}$ and $i_f^{(r')}$. By induction hypothesis(3), $i_f^{(r')} \notin \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$. Claim 6.25 (2) implies that $r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$. By induction hypothesis (2), $r_{\overrightarrow{Q^{(r-1)}}}(S_{i_l^{(r)}}) = b_{r-1} + p_{r-1}/2$, so that $r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}}) = r_{\overrightarrow{Q^{(r-1)}}}(S_{i_l^{(r)}})$. Thus, by definition of *residual*, $residual_{\overleftarrow{Q^{(r)}}}(e^{(r')}_{i_l^{(r)}}) = residual_{\overrightarrow{Q^{(r-1)}}}(e^{(r')}_{i_l^{(r)}})$. Moreover, since $i_f^{(r')} \notin \mathcal{A}_{\overrightarrow{Q^{(r-1)}}}$, $r_{\overleftarrow{Q^{(r-1)}}}(S_{i_f^{(r')}}) = r_{Q^{(r-1)}}(S_{i_f^{(r')}})$. Clearly, $r_{\overleftarrow{Q^{(r-1)}}}(S_{i_l^{(r)}}) = r_{Q^{(r-1)}}(S_{i_f^{(r)}})$. Thus,

$$
residual_{\overrightarrow{Q^{(r-1)}}}(e^{(r')}_{i_l^{(r)}}) = residual_{Q^{(r-1)}}(e^{(r')}_{i_l^{(r)}})\,.
$$

Induction hypothesis (4a) implies that $residual_{Q^{(r-1)}}(e^{(r')}_{i_l^{(r)}}) \geq p_{r-1}/4$. By definition of iupdate function,

$$
\Delta_{\overleftarrow{Q^{(r)}}}(e^{(r')}_{i_l^{(r)}}, i_l^{(r)}) \geq residual_{\overleftarrow{Q^{(r)}}}(e^{(r')}_{i_l^{(r)}})\,.
$$

Therefore,

$$
\Delta_{\overleftarrow{Q^{(r)}}}(e^{(r')}_{i_l^{(r)}}, i_l^{(r)}) \geq \frac{p_{r-1}}{4}\,.
$$

Thus, by definition of update operation,

$$\begin{aligned}\Delta_{\overleftarrow{Q^{(r)}}}(i_l^{(r)}) &= \min\{\frac{p_{r-1}}{16}, \frac{p_{r-1}}{4}\} \\ &= \frac{p_{r-1}}{16} \\ &= \frac{p_r}{4} \qquad \text{(by recursive definition of } p_r\text{)},\end{aligned}$$

so that

$$\begin{aligned}r_{Q^{(r)}}(S_{i_l^{(r)}}) &= r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}}) + \Delta_{\overleftarrow{Q^{(r)}}}(i_l^{(r)}) \\ &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{p_r}{4} \\ &= b_r + \frac{p_r}{4}.\end{aligned}$$

Moreover, since $r_{\overleftarrow{Q^{(r)}}}(S_{i_f^{(r)}}) > r_{\overleftarrow{Q^{(r)}}}(S_{i_l^{(r)}})$, definition of update operation implies that

$$\begin{aligned}r_{Q^{(r)}}(S_{i_f^{(r)}}) &= r_{Q^{(r)}}(S_{i_l^{(r)}}) \\ &= b_r + \frac{p_r}{4}.\end{aligned}$$

This establishes (2a).

By construction of $G(\mathsf{Sched})$, session $S_{i_l^{(r)}}$ does not share any edge with any session $S_i \in \mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \setminus \{i_f^{(r)}\}$. Thus, by definition of update operation, $r_{Q^{(r)}}(S_i) = r_{\overleftarrow{Q^{(r)}}}(S_i)$. Clearly, $S_i \in S \mid_\alpha (Q^{(r-1)}, \overleftarrow{Q^{(r)}})$, so that Lemma 6.25 (1) implies that $r_{\overleftarrow{Q^{(r)}}}(S_i) = b_r + p_r/2$, so that $r_{Q^{(r)}}(S_i) = b_r + p_r/2$. This establishes (2b).

We continue to prove (3). Lemma 6.25 (3), (4), (5) and (6) imply that the only bottleneck edge for state $\overleftarrow{Q^{(r)}}$ is edge $e^{(r)}_{i_l^{(r)}}$). By Lemma 6.25 (7), $\mathcal{A}_{\overleftarrow{Q^{(r)}}} = \mathcal{A}_{Q^{(r)}}$. Thus, the only bottleneck edge for state $Q^{(r)}$ is edge $e^{(r)}_{i_l^{(r)}}$. By construction of $G(\mathsf{Sched})$, edge $e^{(r)}_{i_l^{(r)}}$ is traversed by sessions $i_f^{(r)}$ and $i_l^{(r)}$. Lemma 6.25 (7) implies that both $i_f^{(r)} \in \mathcal{A}_{Q^{(r)}}$ and $i_l^{(r)} \in \mathcal{A}_{Q^{(r)}}$. Thus,

$$\begin{aligned}FS_{Q^{(r)}}(e^{(r)}_{i_l^{(r)}}) &= \frac{cap(e^{(r)}_{i_l^{(r)}}) - allot_{\overrightarrow{Q^{(r)}}}(e^{(r)}_{i_l^{(r)}})}{|\mathcal{A}_{\overrightarrow{Q^{(r-1)}}} \mid e^{(r)}_{i_l^{(r)}}|} \\ &= \frac{2b_r + p_r/2}{2} \\ &= b_r + \frac{p_r}{4}.\end{aligned}$$

Moreover, by (1), $r_{Q^{(r)}}(S_{i_f^{(r)}}) = r_{Q^{(r)}}(S_{i_l^{(r)}}) = b_r + p_r/4$, so that $r_{Q^{(r)}}(S_{i_f^{(r)}}) = r_{Q^{(r)}}(S_{i_l^{(r)}}) = FS_{Q^{(r)}}(e^{(r)}_{i_l^{(r)}})$. Since Alg is bottleneck, it follows that both $i_f^{(r)} \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r)})$, and $i_l^{(r)} \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r)})$. Moreover, for each session index $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$, session S_i does not traverse any bottleneck edge. Since Alg is bottleneck session S_i is not terminated in state $Q^{(r)}$. Thus, for each session index $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, $i \in \{i_f^{(r)}, i_l^{(r)}\}$, if and only if $i \in \mathsf{Term}(\langle G(\mathsf{Sched}), \mathcal{S}\rangle, Q^{(r)})$, which proves (3).

We continue to prove (4). We start by (4a).

Take first any edge $e_i^{(r)}$, $i \in \mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By construction of $G(\mathsf{Sched})$, edge $e_i^{(r)}$ is traversed by sessions $i_f^{(r)}$ and i. By (2a), $r_{Q^{(r)}}(S_{i_f^{(r)}}) = b_r + p_r/4$, while by (2b), $r_{Q^{(r)}}(S_i) = b_r + p_r/2$. Thus,

$$\begin{aligned} \mathit{residual}_{Q^{(r)}}(e_i^{(r)}) &= \mathit{cap}(e_i^{(r)}) - r_{Q^{(r)}}(S_{i_f^{(r)}}) - r_{Q^{(r)}}(S_l) \\ &= 2b_r + p_r - (b_r + \frac{p_r}{4}) - (b_r + \frac{p_r}{2}) \\ &= \frac{p_r}{4}\,. \end{aligned}$$

Consider now any edge $e_i^{(r')}$ for any integer r', $1 \le r' < r$, such that $i \in \mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By construction of network $G(\mathsf{Sched})$, $e_i^{(r')}$ is traversed by i and $i_f^{(r')}$. By definition of *residual*,

$$\begin{aligned} & \mathit{residual}_{Q^{(r)}}(e_i^{(r')}) \\ = \; & \mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) - (r_{Q^{(r)}}(S_{i_f^{(r')}}) - r_{Q^{(r-1)}}(S_{i_f^{(r')}})) - (r_{Q^{(r)}}(S_i) - r_{Q^{(r-1)}}(S_i))\,. \end{aligned}$$

By induction hypothesis (3), $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, so that $r_{Q^{(r)}}(S_{i_f^{(r')}}) = r_{Q^{(r-1)}}(S_{i_f^{(r')}})$. Lemma 6.25 (1) implies that $r_{Q^{(r)}}(S_i) = b_r + p_r/2$, while by induction hypothesis(2b), $r_{Q^{(r-1)}}(S_i) = b_{r-1} + p - r - 1/2$. Thus,

$$\begin{aligned} \mathit{residual}_{Q^{(r)}}(e_i^{(r')}) &= \mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) - 0 - \left(b_r + \frac{p_r}{2}\right) + \left(b_{r-1} + \frac{p_{r-1}}{2}\right) \\ &= \mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) - \left(b_r + \frac{p_r}{2} - b_r\right) \\ &\quad \text{(by recursive definition of } b_r) \\ &= \mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) - \frac{p_r}{2}\,. \end{aligned}$$

By induction hypothesis 4(1),

$$\mathit{residual}_{Q^{(r-1)}}(e_i^{(r')}) \;\ge\; \frac{p_{r-1}}{4}\,,$$

so that

$$\begin{aligned} residual_{Q^{(r)}}(e_i^{(r')}) &\geq \frac{p_{r-1}}{4} - \frac{p_r}{2} \\ &= p_r - \frac{p_r}{2} \\ &\quad \text{(by recursive definition of } p_r) \\ &= \frac{p_r}{2} \\ &\geq \frac{p_r}{4}\,. \end{aligned}$$

The proof of (4a) is now complete.

We continue to prove (4b).

Consider first any edge $e_i^{(r)}$, such that $i \in \mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By construction of $G(\mathsf{Sched})$, edge $e_i^{(r)}$ is traversed by sessions $i_f^{(r)}$ and i. By (3), $i_f^{(r)} \notin \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$, while $i \in \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$. Thus, by (2a), $r_{\underset{Q^{(r)}}{\rightarrow}}(S_{i_f^{(r)}}) = r_{Q^{(r)}}(S_{i_f^{(r)}}) = b_r + p_r/4$, so that $allot_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r)}) = b_r + p_r/4$. Thus,

$$\begin{aligned} FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r)}) &= \frac{cap(e_i^{(r)}) - allot_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r)})}{|\mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}} \mid e_i^{(r)}|} \\ &= \frac{2b_r + p_r - (b_r + p_r/4)}{1} \\ &= 2b_r + p_r - b_r - \frac{p_r}{4} \\ &= b_r + \frac{3}{4}\,p_r\,. \end{aligned}$$

Consider now edge $e_i^{(r')}$ for any integer r', $1 \leq r' < r$, and $i \in \mathcal{A}_{Q^{(r-1)}} \setminus \{i_f^{(r)}, i_l^{(r)}\}$. By construction of network $G(\mathsf{Sched})$, $e_i^{(r')}$ is traversed by i and $i_f^{(r')}$. By induction hypothesis (3), $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$, so that $i_f^{(r')} \notin \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$. By (3), $i \in \mathcal{A}_{\underset{Q^{(r)}}{\rightarrow}}$. Clearly, $i \in \mathcal{A}_{\underset{Q^{(r-1)}}{\rightarrow}}$ Thus, $FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e_i^{(r')}) = FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r')})$. By induction hypothesis (4b), $FS_{\underset{Q^{(r-1)}}{\rightarrow}}(e_i^{(r')}) \geq b_{r-1} + 3p_{r-1}/4$, so that $FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r')}) \geq b_{r-1} + 3p_{r-1}/4$. Clearly,

$$\begin{aligned} b_r + \frac{3}{4}\,p_r &= b_{r-1} + \frac{p_{r-1}}{2} + \frac{3}{4} \cdot \frac{p_{r-1}}{4} \qquad \text{(by recursive definitions of } b_r \text{ and } p_r) \\ &= b_{r-1} + \frac{11\,p_{r-1}}{16} \\ &< b_{r-1} + \frac{3\,p_{r-1}}{4}\,. \end{aligned}$$

Since $FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r')}) \geq b_{r-1} + 3p_{r-1}/4$, it follows that

$$FS_{\underset{Q^{(r)}}{\rightarrow}}(e_i^{(r')}) \;\; > \;\; b_r + \frac{3p_r}{4}\,.$$

This completes the proof of (4b). ■

For each pair of indices j and r, $1 \leq j \leq n/d$ and $1 \leq r \leq d/2$, denote $\mathcal{U}^{(r)}_{\mathsf{Alg}}(G_j, \mathcal{S}_j)$ the number of update operations performed on sessions in cluster $\mathcal{S}_j$ during epoch r. Clearly,

$$\mathcal{U}^{(r)}_{\mathsf{Alg}}(G_j^{(r)}, \mathcal{S}_j) \;\; = \;\; d - 2(r-1)\,.$$

So,

$$\mathcal{U}_{\mathsf{Alg}}(G_j, \mathcal{S}_j) \;\; = \;\; \sum_{r=1}^{d/2} \mathcal{U}^{(r)}_{\mathsf{Alg}}(G_j^{(r)}, \mathcal{S}_j)\,,$$

and

$$\begin{aligned}
\mathcal{U}_{\mathsf{Alg}}(G, \mathcal{S}) &= \sum_{j=1}^{n/d} \mathcal{U}_{\mathsf{Alg}}(G_j, \mathcal{S}_j) \\
&= \sum_{j=1}^{n/d} \sum_{r=1}^{d/2} \mathcal{U}^{(r)}_{\mathsf{Alg}}(G_j^{(r)}, \mathcal{S}_j) \\
&= \sum_{j=1}^{n/d} \sum_{r=1}^{d/2} (d - 2(r-1)) \\
&= \frac{n}{d} \sum_{r=1}^{d/2} (d + 2 - 2r) \\
&= \frac{n}{d} \left((d+2)\,\frac{d}{2} - 2 \sum_{r=1}^{d/2} r \right) \\
&= \frac{n}{d} \left((d+2)\,\frac{d}{2} - \frac{d}{2}\,(\frac{d}{2} + 1) \right) \\
&= \frac{n}{d} (\frac{d^2}{4} + \frac{d}{2}) \\
&= \frac{nd}{4} + \frac{n}{2}\,,
\end{aligned}$$

as needed. ■

We continue with a demonstration of the technique employed in the proof of Theorem 6.1 for constructing, given an oblivious scheduler, a network which is "bad" for Scheduler, We

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0
512	0	0	0	0	0	0	0	0	0	0	0
512	**512**	0	0	0	0	0	0	0	0	0	0
512	512	**512**	0	0	0	0	0	0	0	0	0
512	512	512	**512**	0	0	0	0	0	0	0	0
512	512	512	512	**512**	0	0	0	0	0	0	0
256	512	512	512	512	256	0	0	0	0	0	0
256	512	512	512	512	256	**512**	0	0	0	0	0
256	512	512	512	512	256	512	**512**	0	0	0	0
256	512	512	512	512	256	512	512	**512**	0	0	0
256	512	512	512	512	256	512	512	512	**512**	0	0
256	512	512	512	512	256	512	512	512	512	**512**	0
256	512	512	512	512	256	256	512	512	512	512	**256**
256	512	512	512	512	256	256	512	512	512	**640**	256
256	512	512	512	512	256	256	512	512	**640**	640	256
256	512	512	512	512	256	256	512	**640**	640	640	256
256	512	512	512	512	256	256	**576**	640	640	576	256
256	512	512	512	**640**	256	256	576	640	640	576	256
256	512	512	**640**	640	256	256	576	640	640	576	256
256	512	**640**	640	640	256	256	576	640	640	576	256
256	**576**	640	640	576	256	256	576	640	640	576	256
256	576	**672**	640	576	256	256	576	640	640	576	256
256	576	656	**656**	576	256	256	576	640	640	576	256
256	576	656	656	576	256	256	576	**672**	640	576	256
256	576	656	656	576	256	256	576	656	**656**	576	256

Figure 2: Steps of the algorithm Elev on the network G(Elev) (shown in Figure 1)

demonstrate the construction of a network which incurs $\frac{dn}{4} + \frac{n}{2} = 24$ update operations for Elevator to converge to the max-min vector. The steps of the algorithm Elevator on the constructed network are depicted in Figure 2. Bold entries in the table are those corresponding to increased rates, while underlined entries correspond to decreased rates.

Clearly, the number of update operations incurred is $12 + 8 + 4 = 24$, as claimed in the beginning.

7 Deterministic, Partially Oblivious Algorithms

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

Theorem 7.1 *Assume that* Alg *is a deterministic, partially oblivious, bottleneck algorithm that computes the max-min rate vector. Then,*

$$\mathcal{U}_{\mathsf{Alg}} \geq \frac{dn}{4} + \frac{n}{2}.$$

The proof is similar to that of Theorem 6.1; the main complication is due to the fact that the "complete" sequence of sessions produced by *any* partially oblivious scheduler is not known a priori, as has been the case for oblivious schedulers. Hence, both the sequence of sessions produced by any arbitrary but fixed partially oblivious scheduler (by using all intermediate rates) and the network for which $\Omega(dn)$ update operations are required for convergence to the max-min vector (whose capacities determine the intermediate rates) are simultaneously "hand-crafted" in a step-by-step fashion. As an example, we present the construction of the session sequence $i_1, i_2, \ldots, i_d$ for the first epoch. This sequence is defined inductively. For the basis case, where $l = 1$, set $i_1 = \mathsf{Scheduler}_1(\mathbf{0})$; that is, S_{i_1} is the session scheduled first by Scheduler. Assume inductively that we have defined $i_1, i_2, \ldots, i_{l-1}$; let $Q_1, Q_2, \ldots, Q_{l-1}$ be the sequence of network states right after the scheduling of each of these sessions. Define $i_l = \mathsf{Scheduler}_l(Q_{l-1})$ such that i_l is different from each of $i_1, i_2, \ldots, i_{l-1}$ (we continue applying Scheduler till we reach an index different than those seen so far). Denote $i_1, i_2, \ldots, i_d = \mathsf{Scheduler} \uparrow \mathcal{S}_j$. The rest of the proof is similar to that of Theorem 6.1.

8 Upper Bounds

Upper bounds for deterministic and randomized, oblivious algorithms, and for non-oblivious algorithms are shown in Sections 8.1, 8.2 and 8.3, respectively.

8.1 Deterministic, Oblivious Algorithms

In this section, we present the algorithm RoundRobin and show:

Theorem 8.1 RoundRobin *computes the max-min rate vector within* $dn/2 + n/2$ *update operations.*

The scheduler of RoundRobin conducts d scheduling "rounds". In each round, each of the n sessions is scheduled in "round-robin" order; the algorithm owes its name to the resemblance of its scheduler to the classical algorithm of round-robin scheduling of tasks on a processor (see, e.g., [40, Section 2.4.1]). Moreover, RoundRobin is bottleneck. Formally, define RoundRobin $= \langle \mathsf{RRSched}, \mathsf{RRTerm} \rangle$, where for each l, $0 \leq l \leq dn - 1$, $\mathsf{RRSched}(l+1) = l \bmod n + 1$, and RRTerm is bottleneck.

By definition of RRSched, each session is scheduled once in each round. Thus, Theorem 4.9 implies that at least one session per cluster becomes done in each of the d rounds. Since each cluster contains at most d sessions, it follows that all sessions are done after d rounds are conducted, whence the network enters a final state. Hence, Proposition 3.4 immediately implies that RoundRobin computes the max-min rate vector.

We continue to establish an upper bound on the convergence complexity of RoundRobin. Since at least one session per cluster becomes done in each round, it follows that for any cluster $\mathcal{S}_j$, $|\mathcal{S}_j| - l + 1$ update operations occur in the lth round, $1 \leq l \leq |\mathcal{S}_j|$, on sessions in $\mathcal{S}_j$. Hence,

$$\begin{aligned}
\mathcal{U}_{\mathsf{RoundRobin}} &= \sum_{j \geq 1} \mathcal{U}_{\mathsf{RoundRobin}}(\mathcal{S}_j) \\
&\leq \sum_{j \geq 1} \sum_{1 \leq l \leq |\mathcal{S}_j|} (|\mathcal{S}_j| - l + 1) \\
&= \sum_{j \geq 1} \sum_{1 \leq l \leq |\mathcal{S}_j|} l \\
&= \sum_{j \geq 1} \frac{|\mathcal{S}_j| \, (|\mathcal{S}_j| + 1)}{2} \\
&= \frac{1}{2} \sum_{j \geq 1} |\mathcal{S}_j|^2 + \frac{1}{2} \sum_{j \geq 1} |\mathcal{S}_j| \\
&\leq \frac{\max_{j \geq 1} |\mathcal{S}_j| \, n}{2} + \frac{n}{2} \quad \text{(by Lemma 2.1)} \\
&= \frac{dn}{2} + \frac{n}{2}.
\end{aligned}$$

8.2 Randomized, Oblivious Algorithms

In this section, we present the algorithm Random and show:

Theorem 8.2 Random *computes the max-min rate vector within expected* $O(d \ln n / p_{min})$ *update operations.*

Fix non-zero probabilities $p_1, p_2, \ldots, p_n$, where $\sum_{i=1}^{n} p_i = 1$. Denote $p_{min} = \min_{1 \leq i \leq n} p_i$; notice that $0 < p_{min} \leq 1/n$. The scheduler of Random is a randomized, oblivious scheduler; it schedules session S_i with probability p_i, $1 \leq i \leq n$. Translating this informal description into a function (from integers to probability distributions) is a straightforward task. Moreover, Random is bottleneck.

Fix any execution α of Random. Consider a sequence of indices $l_0, l_1, \ldots, l_d$ defined inductively as follows. For the basis case, set $l_0 = 0$. Assume inductively that we have defined l_{r-1} for some integer $r \geq 1$. For the induction step, define $l_r = \widehat{l}_{r-1}$; that is, l_r is the index of the earliest state following $Q_{l_{r-1}}$ by which all sessions that remain active in $Q_{l_{r-1}}$ will have been scheduled at least once, or infinite if no such index exists.

By definition of l_r and Theorem 4.9, for any cluster $\mathcal{S}_j$ that remains active in $Q_{l_{r-1}}$, there exists some session $S \in \mathcal{S}_j \cap \mathcal{A}_{Q_{l_{r-1}}}$ such that $S \in \mathcal{D}_{Q_{l_r}}$; since $|\mathcal{S}_j| \leq d$, it follows that all n sessions are done in state Q_{l_d}, whence the network enters a final state. Hence, Proposition 3.4 immediately implies that Random computes the max-min rate vector in any execution α for which all indices $l_1, l_2, \ldots, l_d$ are finite.

We continue to show an upper bound on the expectation of the convergence complexity of Random.

Proposition 8.3 $\mathcal{E}(\mathcal{U}_{\mathsf{Random}}) \leq d(\ln n + \gamma)/p_{min}$

Proof: For any index r, $1 \leq r \leq d$, define the random variable $\mathcal{U}_{\mathsf{Alg}}^{(r)}$ to be the number of update operations executed in the execution fragment $\alpha^{(l_r)} = i_{l_{r-1}+1}, Q_{l_{r-1}+1}, \ldots, i_{l_r}, Q_{l_r}$. Since all n sessions are done in state Q_{l_d}. it follows that $\mathcal{U}_{\mathsf{Alg}} = \sum_{r=1}^{d} \mathcal{U}_{\mathsf{Alg}}^{(r)}$, so that $\mathcal{E}(\mathcal{U}_{\mathsf{Alg}}) = \mathcal{E}(\sum_{r=1}^{d} \mathcal{U}_{\mathsf{Alg}}^{(r)}) = \sum_{r=1}^{d} \mathcal{E}(\mathcal{U}_{\mathsf{Alg}}^{(r)})$, by linearity of expectation. Clearly, $\mathcal{U}_{\mathsf{Alg}}^{(r)} \leq l_r - l_{r-1}$, so that $\mathcal{E}(\mathcal{U}_{\mathsf{Alg}}^{(r)}) \leq \mathcal{E}(l_r - l_{r-1})$.

Notice that by definition of the sequence $l_0, l_1, \ldots, l_d$, $l_r - l_{r-1}$ is the least number of states following $Q_{l_{r-1}}$ to reach a state where all sessions that remain active in $Q_{l_{r-1}}$ have been updated at least once; this is no more than the least number of states following $Q_{l_{r-1}}$ to reach a state where all n sessions have been scheduled at least once. It follows by Proposition 2.2 that $\mathcal{E}(\mathcal{U}_{\mathsf{Alg}}^{(r)}) \leq (\ln n + \gamma)/p_{min}$. Hence, $\mathcal{E}(\mathcal{U}_{\mathsf{Alg}}) \leq \sum_{r=1}^{d} (\ln n + \gamma)/p_{min} = d(\ln n + \gamma)/p_{min}$, as needed. ■

An immediate consequence of Proposition 8.3 follows.

Corollary 8.4 *Assume that* $p_{min} \geq 1/n^k$ *for some constant* $k \geq 1$. *Then,* $\mathcal{E}(\mathcal{U}_{\mathsf{Random}}) \leq dn^k(\ln n + \gamma)$.

Clearly, the upper bound on $\mathcal{E}(\mathcal{U}_{\mathsf{Random}})$ established in Corollary 8.4 attains its lowest values when $k = 1$; in this case, for each $i \in [n]$, $p_i \geq p_{min} \geq 1/n$. Since, however, $p_{min} \leq 1/n$, it follows that this case is the *uniform* one where $p_i = 1/n$ for each $i \in [n]$. Thus, we obtain:

Corollary 8.5 *In the uniform case,* $\mathcal{E}(\mathcal{U}_{\mathsf{Random}}) \leq dn(\ln n + \gamma)$.

8.3 Non-Oblivious Algorithms

In this section, we present the algorithm Linear and show:

Theorem 8.6 Linear *computes the max-min rate vector within exactly n update operations.*

Roughly speaking, the scheduler of Linear maintains an edge of minimum fair share that is currently active, and schedules all active sessions traversing it in any arbitrary order. Once it finishes with all of them, it chooses some other (currently active) edge of minimum fair share, and so on. Translating this informal description into a function is straightforward. Moreover, Linear is bottleneck.

Consider any state Q_{l_0} such that $e \in MFSE_{Q_{l_0}}$. Clearly, $e \in MFSE_{Q_{l_0}}(\mathcal{S}_j)$, for some cluster $\mathcal{S}_j$. By definition of the scheduler of Linear, each session traversing e is scheduled exactly once, so that the state $Q_{\widehat{(l)_0|e}}$ is reached; by Theorem 4.7, each such session is done in state $Q_{\widehat{(l)_0|e}}$. By definition of the scheduler of Linear, it follows that all sessions eventually become done, and the network enters a final state. Hence, Proposition 3.4 immediately implies that Linear computes the max-min rate vector

Clearly, the total number of update operations incurred by Linear is exactly n in every execution, which is optimal.

We note that the algorithms RoundRobin and Random, and Linear rely for correctness on combining, in a modular way, a termination property of bottleneck algorithms (Theorems 4.9 and 4.7, respectively) together with Proposition 3.4, which assures that, upon termination, the output of a bottleneck algorithm is a max-min rate vector.

9 Discussion and Directions for Further Research

9.1 Review

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 schedulers, which have no knowledge of either the network topology or the session rates, partially oblivious schedulers, which have knowledge of the session

rates but not of the network topology, and schedulers called non-oblivious, which enjoy full knowledge of the network.

We have shown that the classes of oblivious algorithms and partially oblivious algorithms "collapse" with respect to convergence complexity; most interestingly, our results imply a convergence complexity separation between partially oblivious algorithms (in particular, oblivious) and non-oblivious algorithms for optimistic, bottleneck, rate-based flow control.

Our algorithms demonstrate remarkable combinations of moderate convergence complexity with interesting locality properties; these algorithms significantly improve on the ones in [1]. On the other hand, our lower bounds identify fundamental limitations on the performance of oblivious or even partially oblivious algorithms. Both lower and upper bounds on convergence complexity we have shown have been expressed in terms of session dependency, a network parameter identified by our research, and introduced, as a measure of locality, into complexity considerations.

Both RoundRobin and Random admit simple distributed implementations. The implementation of RoundRobin makes use of a set of underlying "logical rings," one per session cluster, built on top of the communication network. Each ring is designed to traverse at least one edge, called "logical edge," of each session in the cluster and be as short as possible. A "token" is circulated on each ring to simulate the round-robin scheduling of sessions in the cluster; the scheduling order is determined by the sequence of logical edges in the ring. Each time the token traverses a logical edge, an update operation on the corresponding session is initiated; once the operation is completed, the token advances to the next logical edge, and so on. The implementation of Random relies on a randomized, distributed protocol to toss a multiple-sided, biased coin among sessions in each cluster; the communication between different sessions uses shortest paths on the underlying communication network. The choice of which one between RoundRobin and Random to implement can take into consideration issues of efficiency in routing or minimizing global delay, as tailored to each particular application.

Our work has envisioned the problem of optimistic, rate-based flow control as one of *distributed decision-making with incomplete information* [34, 35] and has provided yet another demonstration of the economic value of information [2] as a key computational resource in a distributed system. Beyond the specific results we have derived, we feel that a significant contribution of our work is the formalization of a simple, combinatorial framework within which further research on optimistic, rate-based flow control can be pursued.

For sake of completeness and comparison, we summarize in Table 3 all known (polynomial) lower and upper bounds, established in this work and in the preceding work by Afek *et al.* [1], on the convergence complexity of optimistic, bottleneck algorithms for rate-based flow control.

9.2 Further Research

Our work leaves open several important questions. On the most concrete level, it would be interesting to close the gap between the lower bound of $dn/4 + n/2$ and the upper bound

Scheduler types	Lower bounds	Upper bounds
Oblivious	$dn/4 + n/2$	$dn/2 + n/2$ (RoundRobin) $O(d \ln n / p_{min})$ expected (Random) $O(dn \ln n)$ expected (Random, uniform case)
Partially oblivious	$dn/4 + n/2$	$dn/2 + n/2$ (LocalMin or GlobalMin [1])
Non-oblivious	n	n (Linear)

Figure 3: Summary of known (polynomial) lower and upper bounds on convergence complexity for optimistic, bottleneck rate-based flow control algorithms

of $dn/2 + n/2$ we have shown on the convergence complexity of oblivious algorithms. We conjecture that the true bound is closer to the upper bound than to the lower.

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 still completely untouched by our model and analysis. In the first place, consider the max-min fairness criterion, which, roughly speaking, requires that sessions get as near to equal amounts of bandwidth as possible. We feel that this criterion may be undue in some realistic situations: consider, for example, two conflicting sessions that need to run at 10 Kb/sec and 10 Mb/sec, respectively; clearly, a "fair" arbitration of bandwidth between these two sessions would be undesirable. Thus, an extension of the classical max-min fairness criterion to consider sessions with different demands, which may be dynamically varying, suggests itself. (Some results in this direction have already been obtained by Fatourou *et al.* [14].) Second, the limitation to "static" sets of sessions is somehow restrictive; it would be extremely significant to extend our model and techniques to handle set-up/take-down of sessions.

Jaffe [25] investigates the possibility of maximizing *flow control power* [18] with a class of *decentralized* rate-based flow control algorithms, that work on the design principles of dynamic behavior, use of local information, and decentralized execution; this class abstracts similar motivations with our class of oblivious algorithms. Jaffe [25, Section IV] shows that, under the assumption that communication links behave like M/M/1 servers, flow control power *cannot* be maximized by decentralized algorithms; in sharp contrast, we have shown that oblivious algorithms *can* converge to max-min fairness, albeit no faster than performing $\Omega(dn)$ update operations in the worst case. It would be extremely interesting to further investigate the capabilities and limitations of suitable abstractions of rate-based flow control algorithms, such as oblivious or decentralized, and understand what they can or cannot achieve, and at which possible cost.

Acknowledgments:

Our thanks go to the anonymous PODC'97 reviewers for their helpful comments.

References

[1] Y. Afek, Y. Mansour and Z. Ostfeld, "Convergence Complexity of Optimistic Rate Based Flow Control Algorithms," *Proceedings of the 28th Annual ACM Symposium on Theory of Computing,* pp. 89–98, May 1996.

[2] K. Arrow, *The Economics of Information,* Harvard University Press, 1984.

[3] D. P. Bertsekas and R. G. Gallager, *Data Networks,* second edition, Prentice Hall, Englewood Cliffs, NJ, 1992.

[4] K. Bharath-Kumar and J. M. Jaffe, "A New Approach to Performance-Oriented Flow Control," *IEEE Transactions on Communications,* Vol. COM-29, No. 4 (Special Issue on Congestion Control in Computer Networks), pp. 427–435, April 1981.

[5] F. Bonomi and K. Fendick, "The Rate-Based Flow Control for Available Bit Rate ATM Service," *IEEE Network,* Vol. 9, No. 2, pp. 24–39, March/April 1995.

[6] L. S. Brakmo and L. Peterson, "TCP Vegas: End-to-End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communication,* Vol. 13, No. 8, pp. 1465–1480, October 1995.

[7] A. Charny, "An Algorithm for Rate Allocation in a Packet-Switching Network with Feedback," Technical Report MIT/LCS/TR-601, Laboratory for Computer Science, Massachusetts Institute of Technology, April 1994.

[8] A. Charny, D. D. Clark, and R. Jain, "Congestion Control with Explicit Rate Indication," *Proceedings of the International Conference on Communications (ICC'95),* June 1995.

[9] A. Charny, K. K. Ramakrishnan, and A. Lauck, "Time Scale Analysis and Scalability Issues for Explicit Rate Allocation in ATM Networks," *IEEE/ACM Transactions on Networking,* Vol. 4, No. 4, August 1996.

[10] L. Cherkasova, A. Davis, R. Hodgson, V. Kotov, I. Robinson and T. Rokicki, "Components of Congestion Control," *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures,* pp. 208–210, June 1996.

[11] W. Dally, "Virtual-Channel Flow Control," *Proceedings of the 17th International Symposium on Computer Architecture,* pp. 60–68, May 1990.

[12] Z. Dziong, M. Juda and L. G. Mason, "A Framework for Bandwidth Management in ATM Networks – Aggressive Equivalent Bandwidth Estimation Approach," *IEEE/ACM Transactions on Networking,* Vol. 5, No. 1, pp. 134–147, February 1997.

[13] P. Fatourou, M. Mavronicolas and P. Spirakis, "The Global Efficiency of Distributed, Rate-Based Flow Control Algorithms," *Proceedings of the 5th Colloqium on Structural*

Information and Communication Complexity, June 1998; also, brief announcement in *Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing,* p. 311, June/July 1998.

[14] P. Fatourou, M. Mavronicolas and P. Spirakis, "Max-Min Fair Flow Control Sensitive to Priorities," *Proceedings of the 2nd International Conference on Principles of Distributed Systems,* Amiens, France, December 1998.

[15] E. Gafni, "The Integration of Routing and Flow Control for Voice and Data in Integrated Packet Networks," Ph. D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1982.

[16] E. Gafni and D. Bertsekas, "Dynamic Control of Session Input Rates in Communication Networks," *IEEE Transactions on Automatic Control,* Vol. AC-29, No. 11, pp. 1009–1016, November 1984.

[17] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey," *IEEE Transactions on Communications,* Vol. COM-28, pp. 553–574, 1980.

[18] A. Giessler, J. Hanle, A. Konig, and F. Pade, "Free Buffer Allocation – an Investigation by Simulation," *Computer Networks,* Vol. 2, 1978.

[19] E. Hahne, "Round-Robin Scheduling for Fair Flow Control in Data Communication Networks," Ph. D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1986.

[20] E. Hahne, "Round-Robin Scheduling for MaxMin Fairness in Data Networks," *IEEE Journal on Selected Areas in Communications,* Vol. 9, No. 7, September 1991.

[21] E. Hahne and R. G. Gallager, "Round-Robin Scheduling for Fair Flow Control in Data Communication Networks," *Proceedings of the IEEE International Conference on Communications,* pp. 103–107, June 1986.

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

[23] M. Hofri, *Analysis of Algorithms: Computational Methods and Mathematical Tools,* Oxford University Press, 1995.

[24] J. M. Jaffe, "Bottleneck Flow Control," *IEEE Transactions on Communications,* Vol. COM-29, No. 7, pp. 954–962, July 1981.

[25] J. M. Jaffe, "Flow Control Power is Decentralizable," *IEEE Transactions on Communications,* Vol. COM-29, No. 9, pp. 1301–1306, September 1981.

[26] R. Jain, S. Kalyanaraman and R. Viswanathan, "The OSU Scheme for Congestion Avoidance Using Explicit Rate Indication," Technical Report ATM-FORUM/95-0883, The Ohio State University, September 1994.

[27] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "An Efficient Rate Allocation Algorithm for ATM Networks Providing Max-Min Fairness," *Proceedings of the 6th IFIP International Conference on High Performance Networking (HPN'95),* September 1995.

[28] G. Kesidis, *ATM Network Performance,* Kluwer Academic Publishers, 1996.

[29] H. T. Kung and R. Morris, "Credit-Based Flow Control for ATM Networks," *IEEE/ACM Transactions on Networking,* pp. 40–48, March/April 1995.

[30] J. LeBoudec, "The Asynchronous Transfer Mode: A Tutorial," *Computer Networks and ISDN Systems,* Vol. 24, No. 4, pp. 279–309, 1992.

[31] S. Mishra and L. Wu, "An Evaluation of Flow Control in Group Communication," *IEEE/ACM Transactions on Networking,* Vol. 6, No. 5, pp. 571–587, October 1998.

[32] J. Mosley, "Asynchronous Distributed Flow Control Algorithms," Ph. D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1984.

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

[34] C. H. Papadimitriou, *"The Value of Information,"* invited talk at the World Congress of Economics, Moscow, August 1992.

[35] C. H. Papadimitriou and M. Yannakakis, "On the Value of Information in Distributed Decision Making," *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing,* pp. 61–64, August 1991.

[36] C. Partridge, *Gigabit Networking,* Addison-Wesley, 1994.

[37] G. Polya, "Eine Wahrscheinlichkeitsaufgabe zur Kundenwerbung," *Zeitschrift Fur Angewandte Mathematik und Mechanik,* Vol. 10, pp. 96-97, 1930.

[38] M. D. Prycher, *Asynchronous Transfer Mode, Solution for Broadband ISDN,* Ellis Horwood, 1992.

[39] K. S. Sin and H. H. Tzeng, "Adaptive Proportional Rate Control (APRC) with Intelligent Congestion Indication," Technical Report ATM-FORUM/94-0888, University of California at Irvine, September 1994.

[40] A. S. Tanenbaum, *Modern Operating Systems,* Prentice-Hall, 1992.

[41] J. Walrand and P. Varaiya, *High-Performance Communication Networks,* Morgan Kaufmann Publishers, 1997.