Stability of Service under Time-of-Use Pricing

Shuchi Chawla* Nikhil R. Devanur† Alexander E. Holroyd‡ Anna R. Karlin§
James Martin¶ Balasubramanian Sivan||

Abstract

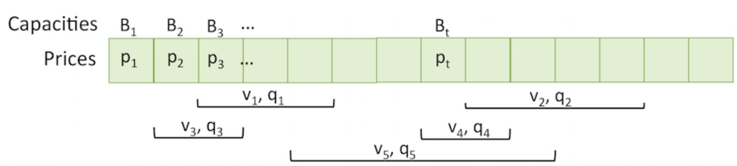
We consider "time-of-use" pricing as a technique for matching supply and demand of temporal resources with the goal of maximizing social welfare. Relevant examples include energy, computing resources on a cloud computing platform, and charging stations for electric vehicles, among many others. A client/job in this setting has a window of time during which he needs service, and a particular value for obtaining it. We assume a stochastic model for demand, where each job materializes with some probability via an independent Bernoulli trial. Given a per-time-unit pricing of resources, any realized job will first try to get served by the cheapest available resource in its window and, failing that, will try to find service at the next cheapest available resource, and so on. Thus, the natural stochastic fluctuations in demand have the potential to lead to cascading overload events. Our main result shows that setting prices so as to optimally handle the *expected* demand works well: with high probability, when the actual demand is instantiated, the system is stable and the expected value of the jobs served is very close to that of the optimal offline algorithm.

*University of Wisconsin-Madison. shuchi@cs.wisc.edu
†Microsoft Research, Redmond. nikdev@microsoft.com
‡Microsoft Research, Redmond. holroyd@microsoft.com
§University of Washington, Seattle. karlin@cs.washington.edu
¶Oxford University. martin@stats.ox.ac.uk
||Google Research. balusivan@google.com

1 Introduction

For many commodities of a temporal nature, demand and supply fluctuate stochastically over time. Demand for electricity changes over the course of a day as well as across different days of the week—home owners use more electricity during evenings and weekends, offices use more during normal working hours, and energy usage at factories can follow an altogether different cycle depending on workloads. On the other hand, supply from sources of renewable energy depends on weather conditions and can also vary significantly over time. Likewise, demand for computing resources on a cloud computing platform varies over time depending on users' workloads. Supply also varies stochastically, depending on scheduled and unscheduled downtime for servers and other kinds of outages. In this paper, we explore the effectiveness of "time-of-use" pricing as a method for efficiently and effectively matching supply and demand in such settings.

Online matching of temporal resources: Suppose that there are B_t units of resource available at time t and each potential client, a.k.a. "job", j has a window of time during which it would like to obtain "service", say, a unit allocation of the resource. Job j obtains a value of v_j from getting serviced at any time in its window. (See figure below.) We consider the following model of job arrival that is a hybrid of stochastic and adversarial models: job j is realized with probability q_j via an independent Bernoulli trial. Jobs arrive online in the system in an adversarial order that can depend on the set of realized jobs.



How should the supplier allocate the available resources to jobs so as to maximize the total expected value[1] of the jobs that are served? Perhaps the most natural approach is to use the stochastic information about demand to price the available resources on a per-time-unit basis. Such **time-of-use pricing** is an effective way for the supplier of a temporal commodity to balance supply and demand. During lean supply periods, advertising a high price suppresses demand, whereas during times of excess supply, advertising a low price encourages higher demand. Moreover, allowing each client to be a "price-taker", that is, making sure that each client is allocated the cheapest available resource that meets his requirements,[2] trivially guarantees that clients will be truthful about all of their parameters: there is no advantage to misreporting one's value or service window.

For a single time period in isolation, determining the right price to set is a *newsvendor* problem [21]. The optimal solution is to set the price so that the system is slightly overprovisioned with the expected supply matching the expected demand plus a small reserve. Even for this setting, if jobs arrive in adversarial order, $\Omega(\epsilon^{-2}\ln(\epsilon^{-1}))$ units of resource are needed to guarantee that the expected value of the jobs served is at least $1-\epsilon$ times the expected value of the jobs scheduled by the optimal offline algorithm.

For the general case, as a thought experiment, suppose that we only needed to satisfy supply constraints in expectation in every time period. This entails solving an "expected LP", which yields a set of prices, one for each period, and automatically matches potential jobs with a cheapest slot in their window. But how well does such a system work under the natural stochastic fluctuations that will necessarily occur? The concern

[1] Also called the efficiency or social welfare of the system.

[2] A resource at time t priced at p_t meets job j's requirement if $p_t \leq v_j$ and time slot t is within job j's window.

is that, because of variability in the realized demand relative to the expected demand, a client may show up and find that the cheapest slot in his window has already been allocated. This will cause him to try to take the next cheapest slot and so on. Such "overload" events, that is, events where demand exceeds supply causing excess demand to be forwarded, are positively correlated across time slots and can exhibit cascading behavior.

Our main theorem is that such time-of-use pricing works well with high probability: Suppose that B_t resources are available at each time t, where $B_t = \Omega(\epsilon^{-2}\ln(\epsilon^{-1}))$. (We call this the "large market assumption.") Given the model of job arrivals described above, there is a set of prices (p_t) such that if (a) realized jobs arrive online in an adversarial order, and (b) upon arrival, each job grabs his favorite available resource given the prices, the expected value of the jobs served is at least $1-\epsilon$ times the expected value of the jobs scheduled by the optimal offline algorithm.

Thus, despite the complex interaction between demand for time slots due to the forwarding of unmet demand and the adversarial arrival order of realized jobs, we can guarantee near-optimal expected performance without increasing capacity over what would be needed in a single time-unit setting.

Key ideas in the proof. The prices we set induce a *forwarding graph*: the nodes are time slots and an edge from time slot t to time slot t' means that $p_t \leq p_{t'}$ and some job might try to grab a resource at time t' immediately after failing to find an available resource at time t.

What properties of the forwarding graph determine whether or not overload cascades are likely? Perhaps unsurprisingly, the maximum in-degree of a node in the forwarding graph is key. Suppose, for example, that one time slot t has very high in-degree, meaning that it may receive forwarded jobs from many other time slots. Even if each of the latter time slots has a low probability of forwarding a job, the total expected number of jobs forwarded to t may be high, and may therefore lead to a high probability of overload at t. If all of the highest value jobs happen to have t as the only slot in their window, this could wreak havoc on our social welfare bounds.

What *is* perhaps somewhat surprising though is that maximum in-degree is the *only* relevant graph parameter. In particular, the size of the graph does not play a role. Showing that our theorem holds is easy if the forwarding graph is a line or even a bounded-degree tree; the analysis boils down to proving inductively that the number of jobs forwarded from one time slot to the next satisfies an exponential tail bound. However, once the graph has cycles, inductive arguments no longer apply. A key part of our proof consists of showing that, among bounded degree graphs, a bounded degree tree will maximize the probability of overload at any time slot. This requires the use of a "decorrelation lemma" that allows us to upper bound the probability of bad dependent events by the probability of bad independent events.

Unfortunately, though, the story doesn't end here, because the forwarding graph induced by our pricing does not in general yield a bounded degree forwarding graph.[3] Nonetheless, we show that the paths created by the forwarding of jobs across time slots possess a simple canonical form that allows us to modify them and obtain a new forwarding graph of in-degree at most 3, without reducing the load at any resource.

Beyond unit-length jobs. We extend the above result to the setting where each job j requires the use of the resource for some number of consecutive time units within its window. This is a significantly more complicated problem and, correspondingly, requires a stronger the large market assumption.[4]

[3] Figure 3 shows a concrete example where the in-degree of a time slot can be unbounded.

[4] We obtain bounds that match the unit-length case, except for an additional polynomial dependency of the supply requirement on the maximum length of a job.

Mechanism design for temporal resources. As an application of our main theorem, we develop a new online mechanism for selling cloud services when jobs are strategic, that achieves a number of desirable properties in addition to being near-optimal for social welfare. The problem of designing truthful mechanisms for scheduling jobs with deadlines has been studied with many variations in the worst case setting: the parameters the job can lie about (arrival, departure, value, etc.), deterministic vs. randomized, whether payments are determined immediately (prompt) or not (tardy), unit length vs. arbitrary (bounded) length jobs, and assuming certain *slackness* in the deadlines [3, 4, 7, 12, 15]. In the worst case setting, the underlying algorithmic problem (that is, without incentive constraints) already exhibits polylogarithmic hardness [5].[5] Lavi and Nisan [19] showed that no deterministic truthful mechanism (w.r.t. all the parameters) can approximate social welfare better than a factor T in the worst case, where T is the time horizon, even for unit length jobs on a single machine. Subsequent papers cope with this impossibility by weakening different assumptions. In contrast, we consider the Bayesian setting, where jobs are drawn from a known distribution. We give a simple *order oblivious posted pricing* mechanism (OPM), where the seller announces prices, and jobs purchase resources in a greedy first-come-first-served fashion. Our mechanism is truthful for jobs' values, requirements, and deadlines; is prompt in that jobs' allocations and payments are determined right at the time of their arrival; and in the stochastic setting, under the large market assumption described above, achieves near-optimal efficiency (a $1 - o(1)$ approximation). Determining the pricing requires the seller to know the demand distribution. When the demand distributions are cyclic, say with a period of a week or a month or a year, the optimal prices are also cyclic with the same periodicity. The seller can then use a polynomial size linear program to solve for the appropriate prices. If the demand distribution stays constant over time, then a constant price per unit of resource per unit of time suffices to provide near-optimal system efficiency.

OPMs have previously been shown to achieve constant-factor approximations to revenue and social welfare in many different settings. See, e.g., [6, 12], and references therein. Feldman et al. [12] show, in particular, that for settings with many items and many clients with fractionally subadditive values, there always exists an item pricing such that if clients purchase their favorite bundles of items sequentially in arbitrary order, the expected social welfare achieved is at least half of the optimum. For our setting with temporal resources, this implies that when all jobs have unit length, there exists a time-of-use pricing that obtains a half approximation to the optimal social welfare. In contrast, we obtain a $(1 - \epsilon)$ approximation via the same kind of selling mechanism under a large market assumption. Furthermore, while Feldman et al.'s approach can only guarantee an $O(\ell_{\max})$ approximation when jobs have lengths in $\{1, \cdots, \ell_{\max}\}$, we are able to use item pricing[6] to again obtain a $(1 - \epsilon)$ approximation under a large market assumption. To our knowledge, this is the first near-optimal result $(1 - o(1))$ achieved via OPMs that has no dependence on the length of time the system is running (or, in the setting considered by [12], the number of items being sold).

Other applications. While the main motivation for our work is to analyze pricing schemes for temporal resources, our analysis applies to other resource allocation settings where clients have varied preferences over different resources and greedily grab the first available resource on their preference list at their time of arrival. Consider, for example, a network of charging stations for electric vehicles. A client wishing to charge his EV strategically chooses which station to obtain service at, depending on the price, travel time,

[5] The algorithmic problem of stochastic online matching and its generalizations, under large budgets/capacities, are similar in spirit to the stochastic process we consider [1, 2, 8, 9, 11, 13, 18]. The temporal aspects of the two problems are very different, however, due to which standard models in that literature such as the random order model are not a good fit here.

[6] Indeed, Feldman et al. show that in the small market setting, bundle pricing is necessary to achieve an $o(l_{\max})$ approximation.

etc.; if that station is already at capacity, the client goes to his next favorite station, and so on. Depending on the geography of the area and traffic patterns, if the forwarding graph over charging stations formed by such a movement of clients has constant in-degree at every node, then our results apply, and near optimal efficiency can be achieved. While pricing had previously been studied in the context of EV charging (see, e.g., [24]), these works focus on optimizing the average case behavior of the system, rather than studying its stochastic behavior.

Connections to queueing theory. Special cases of our models are closely related to standard models in queuing theory, where the demand and supply are stationary (i.e., not changing with time). In particular, for unit length jobs, suppose that $B_t = B$ for all t, the advertised prices are all equal, and every client tries to obtain service at the first slot in its window, failing which it moves its demand to the next time slot, and then the next, and so on. This case corresponds to the standard M/D/B queueing model, with Markovian arrivals, deterministic processing time, and B servers, under the first-come first-served (FCFS) queuing discipline.[7] Our result matches the *optimal bound* for this model in the so-called *Halfin-Whitt regime* [16]: if the expected demand in every time period is $B - \Omega(\sqrt{B \log(1/\delta)})$, then the overload probability is at most δ. In other words, every arriving job obtains service at the first time slot in its window with probability $1 - \delta$.

In the more general model with different job lengths, the special case of stationary demand and supply corresponds to the more general M/GIB/B queueing model, where jobs have arbitrary but bounded processing times. Even though optimal bounds in the Halfin-Whitt regime for FCFS have been known for M/M/B and M/D/B queues [10], GI/M/B queues [16], GI/D/B queues [17], and GI/H_2^*/B queues [23], proving such bounds for M/GIB/B queues with FCFS queuing is a major open problem in queuing theory. We consider a variant of FCFS: we admit only a certain limited number of jobs of any particular length at every time slot. Our result in this setting matches the optimal bound in the Halfin-Whitt regime for this variant, albeit with a polynomial dependency on the maximum length. Our techniques might give a way to prove the same bound for FCFS, which would resolve the open problem regarding M/GIB/B queues mentioned above.

Organization of the paper. Our analysis is divided into four main parts. In Appendix D we describe properties of a time-of-use pricing that balances supply and demand in every time period in expectation; these are summarized in Lemma 2.1 in Section 2. In Section 3 we analyze the stochastic resource allocation process for a general forwarding graph over resources, and show that the overload probability is related to the in-degree of the forwarding graph. In Section 4 we analyze the temporal resource setting for unit-length jobs, and give a reduction from this setting to a low-degree-forwarding-graph setting. In Section 5 we extend this analysis to jobs of arbitrary length. A detailed discussion of related work appears in Appendix A.

2 Preliminaries and Main Results

Temporal resource allocation problem. We consider a setting where a seller has multiple copies of a reusable resource available to allocate over time. Clients, a.k.a. *jobs*, reserve a unit of the resource for some length of time, after which that unit again becomes available to be allocated to other jobs. A job j is described by a tuple consisting of a starting time, a deadline, a length, and a value, denoted by (s_j, d_j, l_j, v_j),

[7] The notation for different queuing models is as follows: an A/B/C queue is one where the inter job arrival times are drawn from distributions in family A, the job lengths distributions belong to family B, and there are C identical machines. D is the class of deterministic distributions, M is the class of exponential distributions, GI is the class of general, independent distributions, GIB is the same class with a bounded support, and an H_2^* distribution is a mixture of an exponential and a point mass.

with the first three elements in $\mathbb{Z}_+$ and the last in $\mathbb{R}_+$. The interpretation is that the job can be processed in the time interval $[s_j, d_j]$, and requires l_j consecutive units of time to complete. The value accrued by processing this job is v_j. Let $W_j = [s_j, d_j - l_j + 1]$ denote the "job's window" or the interval of time during which the job can be started so as to finish before its deadline. For each $t \in \mathbb{Z}_+$, at most $B_t \in \mathbb{Z}_+$ jobs can be processed in parallel.

We consider the following stochastic model of job arrival: there is a set of *potential* jobs J; associated with each job $j \in J$ is a probability q_j. A potential job j is realized with probability q_j via an independent Bernoulli trial. The order of arrivals of the realized jobs in the system is determined by an adversary who knows the set of realized jobs. [8]

A scheduling mechanism, at the time of each job's arrival, determines whether or not to accept a job. In the former case it allocates l_j consecutive units of time in the time interval $[s_j, d_j]$, and charges the job a payment p_j. Job j derives a *utility* of $v_j - p_j$ if it is accepted, and 0 otherwise.[9] The objective of the algorithm is to maximize the total value of the jobs processed, a.k.a. the *social welfare*. The mechanism is required to be *truthful* in dominant strategies, which means that a job j can not get a higher utility by misreporting any of its parameters.[10] The algorithm knows the set of potential jobs J (each defined by its associated 4-tuple as above), their arrival probabilities, and the capacities B_t ahead of time, but the realized job arrivals are learned as they happen. The scheduling decision and payment are determined at the time of the jobs' arrival and are irrevocable.

Time of Use pricing. We consider a particularly simple kind of mechanism that announces a "time of use pricing" $(p_t)_{t\in\mathbb{Z}_+}$ up front, where p_t is the price per unit of resource at time t. The mechanism then requires a job of length l_j starting at time t to pay a total price of $p_t(l_j) = \sum_{t'=t}^{t+l_j-1} p_{t'}$. For every job j, let $\textsc{Fav}_j = \arg\min_{t\in W_j : p_t(l_j)\le v_j}\{p_t(l_j)\}$ denote the job's least expensive options within its window, a.k.a. its "favorite" starting slots. A mechanism that assigns every arriving job to one of its favorite slots is trivially truthful. It follows from strong LP duality by a standard argument that with an appropriate choice of prices, such a mechanism obtains nearly the optimal social welfare, *if it is only required to satisfy the supply constraints in expectation*. See Appendix D for the LP and a proof. Let OPT denote the expected maximum social welfare achievable by any feasible (capacity respecting) assignment under this stochastic arrival model.

Lemma 2.1. **(Fractional assignment lemma)** *Fix any set of potential jobs J, their arrival probabilities, and the capacities B_t for all $t \in \mathbb{Z}_+$. Then for any $\epsilon > 0$, $\exists$ nonnegative prices $(p_t)_{t\in\mathbb{Z}_+}$ and a fractional assignment $X_{j,t} \in [0,1]$ from jobs $j \in J$ to their favorite slots $t \in \textsc{Fav}_j$, such that,*

1. *Every job that can afford to pay the price at its favorite slot is fully scheduled: for every j with $p_t(l_j) < v_j$ for $t \in \textsc{Fav}_j$, we have $\sum_{t\in\textsc{Fav}_j} X_{j,t} = 1$.*
2. *The expected allocation at time t is at most $(1-\epsilon)B_t$: $\forall t, \sum_{j\in J, t'\in[t-l_j+1,t]} q_j X_{j,t'} \le (1-\epsilon)B_t$.*
3. *The expected social welfare is at least $(1-\epsilon)$ times the optimum: $\sum_{j\in J, t\in\textsc{Fav}_j} v_j q_j X_{j,t} \ge (1-\epsilon)\mathrm{OPT}$.*

Further, if the distribution is periodic,[11] the prices are also periodic with the same period, and can be computed efficiently.

[8] For example, a job j that shows up well before s_j may make a reservation for resources in its window in advance.

[9] We assume that the utility of job j is $-p_j$ if it does not get at least l_j units of time within the interval $[s_j, d_j]$.

[10] We assume that the setting disallows Sybil attacks. That is, a job of length l cannot pretend to be multiple different "subjobs" of total length l trying to obtain service in consecutive time blocks.

[11] See formal definition in Appendix D.

The asynchronous allocation process. Of course, the actual allocation of slots to jobs happens in an online fashion and the capacity constraints are hard constraints that must be met regardless of which jobs are actually realized.The mechanism we analyze is a greedy first-come first-served[12] type mechanism: The slot prices $(p_t)_{t\in\mathbb{Z}_+}$ induce a preference ordering over slots for each job j; this is a list of slots t in j's window W_j with $p_t(l_j) < v_j$, in non-decreasing order of price. Let Π_j denote the preference ordering[13] of job j over time slots in its window. When job j arrives, it considers time slots in the order of Π_j, and gets served at the first one that has resources available (or doesn't get served if no slot in Π_j has leftover capacity). We emphasize that *which* jobs are realized is determined by the stochastic model described above, but *when* jobs arrive is determined adversarially, and can depend on which other jobs are realized. [14] For this reason, we call this an "asynchronous allocation process".

Our main theorem shows that with an appropriate choice of $\epsilon > 0$, the asynchronous allocation process corresponding to the price vector given by Lemma 2.1 obtains near-optimal social welfare.

Let $l_{\max} := \max_{j\in J} l_j$ and $B := \min_t B_t$. The case when $l_{\max} = 1$ is called the *unit length jobs* setting.

Theorem 2.2. (Stability of service theorem) *$\exists$ a universal constant c such that $\forall$ $\epsilon \in [0, 1/2]$, for prices determined by Lemma 2.1 for this ϵ, in the asynchronous allocation process for the temporal resource allocation problem, every arriving job that can afford the price at its favorite slot gets accepted at such a slot with probability $\geq 1 - \epsilon$, and the social welfare achieved is $\geq (1 - 2\epsilon)$ times* OPT*, if for the unit length jobs case and the general case respectively,*

$$B \geq c\frac{\log(1/\epsilon)}{\epsilon^2}, \quad \textit{and,} \quad B \geq c\frac{l_{\max}^6 \log(1/\epsilon)}{\epsilon^3}.$$

As a step towards proving this theorem, we will study a slightly more abstract setting without prices in the next section: Suppose that the time slots are nodes in a "forwarding graph" G, and that there is an edge from time slot t' to time slot t if there is some job j such that t follows t' in j's preference ordering Π_j. Jobs arrive at the various nodes in the graph[15] and move through this graph until they are successfully served. However, we relax the requirement that each job j must follow Π_j; rather, we allow each job to take an adversarially selected path in the forwarding graph G in its quest for service. We then present conditions on the arrival process, in terms of the maximum indegree of the graph, under which failures are unlikely to cascade.

3 Stability of service for a network of servers

We will analyze the temporal process described above by reducing it to the following *network of servers* setting:

- There is a set of n *servers*, which we identify with $[n]$. Server i can service a total of B_i jobs and then expires.

[12] Sometimes for jobs of length > 1 our mechanism artificially limits the capacity of a slot, that is, does not allocate a block of time slots even when available; however, it does so in a truthful manner. This detail is discussed in Section 5.

[13] When prices of time slots are not unique, this preference ordering is not unique. We need to impose a particular tie breaking rule among the job's favorite slots, but can break ties among other slots arbitrarily. Part of this tie-breaking is required already to satisfy the conclusion of Lemma 2.1, part of it is required to ensure the overall stability of this system. We detail the tie-breaking rule in Section 4. Note that the mechanism remains truthful regardless of the tie-breaking rule.

[14] This is a reversal of the random order model popular in online matching, where the set of arrivals is adversarial but the order is random.

[15] The prices from Lemma 2.1 will guarantee that the number of external arrivals to each node is slightly less than the capacity of that node.

- There is a directed forwarding graph G whose vertex set is the n servers $[n]$. Let $d_{\max}$ denote the maximum indegree in G. We will refer to the vertices of G as either servers or nodes.
- The number of jobs entering the network at each node is determined by a stochastic process[16] : Denote by A_i the number of jobs that enter the network at node $i \in [n]$. The random variables A_i are mutually independent.
- Each arriving job j is forwarded through the network G until the job reaches an *available* server. Server i is available if it has not yet served B_i jobs. Thus, if job j enters the network at i, and server i is available, i serves j and j leaves the network. If i has already served B_i jobs prior to j's arrival, then job j gets *"forwarded"* to some neighbor of i in G and tries to get service there, and so on. Job j leaves the network as soon as it is serviced, or it has tried all reachable servers, or it gives up, whichever happens first.
- All aspects of this process other than the external arrival process are assumed to be *adversarial*: the paths jobs take as they seek an available server, the timing of external arrivals, and the timing of forwarding events.

Our main theorem for this setting gives conditions under which the probability that any particular job gets served by the first server it tries is close to 1. The crucial point here is that this is independent of n.

Theorem 3.1. *Consider the network of servers setting as above. Fix an ϵ in $[0, 1/2]$. Suppose that for each node $i \in [n]$ the moment generating function of $A_i - B_i$ satisfies $\mathbb{E}\left[e^{\epsilon(A_i - B_i)}\right] \le \epsilon^2/ed_{\max}$. Then for any job j, $\Pr[j$ is not served at the first node on its path$] \le \epsilon$. In other words, failures don't cascade and each job is served with high probability at the node at which it enters.*

We introduce some more notation before we proceed. For a particular instantiation of the process (as determined by the stochastic job arrivals and the adversarial timing of arrivals and forwards), let P_j denote an arriving job j's *realized path* in G, the set of servers that j tries to get service from. This path begins of course at the node at which j enters the network. Let $\mathcal{P} = (P_j)$ denote the collection of all realized paths. Let $\ell_i(\mathcal{P})$ denote the number of jobs that attempt to get service at node i (external job arrivals to i as well as forwards), a.k.a. its "load". We say that node i is "overloaded" if $\ell_i(\mathcal{P}) \ge B_i$.

If a node i forwards a job, then node i must have already served B_i other jobs. Thus, the collection of realized paths $\mathcal{P}$ satisfies the following **min-work condition**: for every node i, the number of jobs forwarded is no more than the number of realized paths P_j containing i minus the capacity B_i.

We now proceed to sketch a proof of Theorem 3.1. A detailed proof can be found in Appendix B.

Consider the load on a single node and suppose that it has constant in-degree. If each of the forwards from its predecessors were *independent*, and these forwards were few and far between, as captured by a bound on the expectation of the moment generating function, then it can be argued that forwards from this node would also inductively satisfy a similar bound on its moment generating function. The forwards are not independent, so this simple approach does not work. Moreover, G is not necessarily acyclic, so there is not even an obvious order for induction. However, these conditions are satisfied when G is a tree, and our first lemma formalizes the above approach in this case.

Lemma 3.2. *Fix an ϵ in $[0, 1/2]$. Suppose that the network G is a finite directed tree, that is, it contains no directed cycles and every node has out-degree 1, and that the moment generating function of $A_i - B_i$ for each node i satisfies $\mathbb{E}\left[e^{\epsilon(A_i - B_i)}\right] \le \epsilon^2/ed_{\max}$. Then, for any i, $\Pr[\ell_i(\mathcal{P}) \ge B_i] \le \epsilon$.*

[16] E.g., Potential job j arrives with probability q_j.

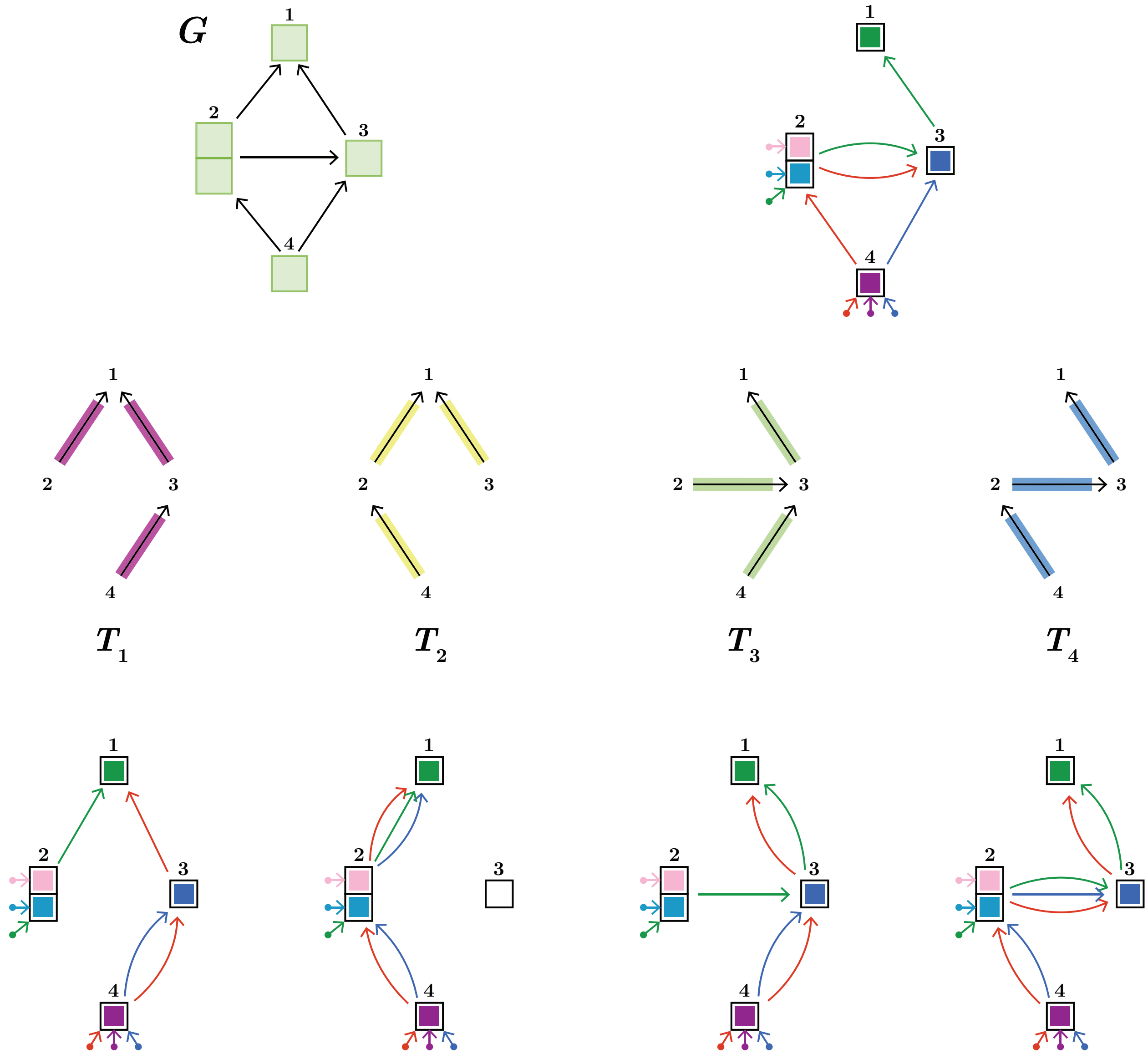


Figure 1: This figure illustrates the first step of the argument for the network of servers. The graph G is shown on the upper left. In this example, all nodes except for node 2 have a capacity of 1; node 2 has a capacity of 2. The upper right shows an example of what might happen with three jobs arriving at node 2 and three arriving at node 4. The blue job arriving at node 4 gets forwarded to node 3 where it gets served. The green job arriving at node 2 gets forwarded to node 3 and then to node 1 where it finally gets service, and so on. Notice that the load at node 1 for this set of job arrivals and paths is 1. The middle panel of the figure shows the four trees in $\mathcal{T}(1)$. The bottom panel shows how the packets might be routed and served, if all forwarding was done along edges of the associated tree (immediately above). In this example, the worst case load at node 1 is when all jobs are routed along edges of the tree T_2. This results in a load of 3 at node 1.

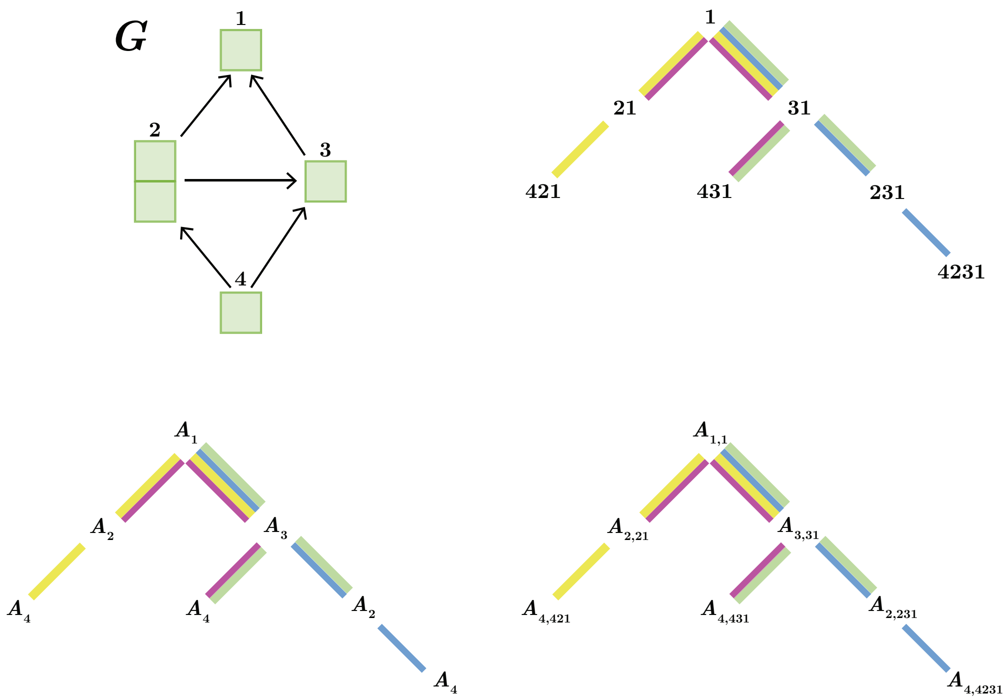


Figure 2: This figure illustrates the second step of argument for network of servers. Again on the upper left we see the graph G. On the upper right we see the tree of trees $\mathbf{T}_1$. Each node is labeled with a simple path in G to node 1. It is easy to see that each of the trees $\mathcal{T}(1) = \{T_1, \ldots, T_4\}$ from Figure 1 is a subtree of $\mathbf{T}_1$. (The edges are color-coded as in the corresponding tree in $\mathcal{T}(1)$.) The bottom left version of $\mathbf{T}_1$ indicates on each node the random variable which is the number of external job arrivals at that node. Lemma 3.3 implies that it suffices to bound the probability of overflow at the root of this tree. Lemma 3.5 shows that instead we can bound the probability of overflow at the root of the bottom right tree, where the external arrivals at different nodes are independent. For example, $A_{2,21}$ and $A_{2,231}$ are independent samples from the distribution of A_2.

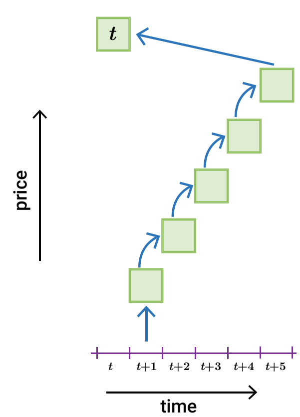


Figure 3: This figure illustrates that the temporal process can have high in-degree. Suppose that there are jobs with window $[t, t+1]$, $[t, t+2]$, $[t, t+3]$ and so on. Then all of these jobs would first try slot $t+1$. The final slot all of these jobs would try is slot t, so there would be an edge in the forwarding graph from each of $t+1, \ldots, t+5$ to t.

Our proof of the Theorem 3.1 will reduce the analysis in a general network to that in an appropriately defined tree network. The argument has two parts that we outline next. Throughout the proof, we will focus on a particular server u in G.

Part 1: Reducing to a tree for fixed arrivals

In the first part, we fix the set of realized paths $\mathcal{P}$ (as determined by the stochastic job arrivals and the adversarial timing of arrivals and forwards). This fixes the entries $\mathbf{a} = (a_i)_{i=1}^n$, where a_i is the number of jobs arriving at node i from outside the network. We then show that if the node u is overloaded for this fixed outcome, then there exists a subtree of the network G that is rooted at u, such that if jobs are forwarded exclusively along edges of this tree until service is received (or there is no where else to go), then node u is still overloaded.

More formally, let T be a directed tree rooted at the node u. For a vector of external arrivals $\mathbf{a} = (a_i)$ and node i, let $\ell_i^T(\mathbf{a})$ denote the load on node i (external arrivals plus forwards) when jobs are forwarded along the edges of the tree T until service is received.

Let $\mathcal{T}(u)$ denote the set of all directed subtrees of G rooted at node u. The following lemma captures the first part of our analysis.

Lemma 3.3. *If a fixed set of arrivals (resulting in a particular* $\mathbf{a}$*) and induced paths* $\mathcal{P}$ *overload a node* u *in the network* G*, then* $\exists$ *a tree* $T \in \mathcal{T}(u)$ *such that* u *is overloaded with the same set of arrivals* $\mathbf{a}$ *when requests are routed along* T*. Formally,*

$$\ell_u(\mathcal{P}) \geq B_u \quad \textit{implies that} \quad \max_{T \in \mathcal{T}(u)} \ell_u^T(\mathbf{a}) \geq B_u.$$

The first step in the proof consists of removing cycles in $\mathcal{P}$ while preserving the set of overloaded vertices. In the second step we reroute the paths so that they form a tree. The proof of this lemma is deferred to the appendix. An example is presented in Figure 1.

Part 2: Reducing to a tree of trees

Lemma 3.3 does not reduce the analysis of the network of servers setting to the analysis of a single tree, because the particular tree that gives the worst-case load on u depends on the realized arrivals $\mathbf{a} = (a_i)_{i=1}^n$. The lemma does show, however, that to complete the proof of Theorem 3.1 it suffices for us to bound the probability $\Pr\left[\max_{T \in \mathcal{T}(u)} \ell_u^T(\mathbf{A}) \geq B_u\right]$ for each node u, where $\mathbf{A} := (A_i)_{i=1}^n$. (Recall that random variable A_i denotes the number of external arrivals at node i, and that the different A_i's are mutually independent.)

In order to analyze this quantity, we will construct a new tree network[17] $\mathbf{T}_u$ over an expanded set of nodes that contains *every* tree $T \in \mathcal{T}(u)$ as a subtree. The tree $\mathbf{T}_u$ is defined as follows: There is a node v_P in $\mathbf{T}_u$ for each simple directed path P in G terminating at u, and there is an edge in $\mathbf{T}_u$ from v_P to $v_{P'}$, if $P = iP'$ for some node i in G. By construction, each tree T in $\mathcal{T}(u)$ has a unique isomorphic copy in $\mathbf{T}_u$. See Figure 2. [18]

We then consider the network of servers process on $\mathbf{T}_u$, under the assumption that for every node v_P such that $P = iP'$, the number of external arrivals at v_P is A_i, and also that as long as a job is not serviced it is forwarded along the next edge in the tree. Then for any tree $T \in \mathcal{T}(u)$, the load on the node corresponding

[17] We call this the *tree of trees*.

[18] This construction blows up the number of nodes exponentially, but this does not affect us since our bound (Lemma 3.2) is independent of the number of nodes in the tree.

to i in the isomorphic copy of T in $\mathbf{T}_u$ is no smaller than the load on i in T under the same set of arrivals. In particular,

$$\Pr\left[\ell_u^{\mathbf{T}_u}(\mathbf{A}) \geq B_u\right] \geq \Pr\left[\max_{T\in\mathcal{T}(u)} \ell_u^T(\mathbf{A}) \geq B_u\right]. \tag{3.1}$$

Unfortunately, we cannot analyze $\ell_u^{\mathbf{T}_u}(\mathbf{A})$ as in the proof of Lemma 3.2, since the external arrivals at different nodes are correlated. In particular, for each node i in G, there are n_i nodes in $\mathbf{T}_u$ at which the entries A_i are the same, where n_i is the number of different directed simple paths from i to u in G. The key step in the rest of the proof is to show that replacing these by independent draws from the same distribution can only (stochastically) increase the load at u. To this end, we require the following "decorrelation" lemma:

Lemma 3.4. (Decorrelation lemma for the max function) *Let $g_\ell : \Re \to \Re$, $\ell = 1, \ldots, k$ be any non-decreasing functions, X be any real valued random variable, and let $Y_1, \ldots, Y_k$ be independent and identically distributed random variables from the same distribution as X. Then,*

$$\max_\ell\{g_\ell(Y_\ell)\} \overset{\text{st}}{\geq} \max_\ell\{g_\ell(X)\},$$

where $\overset{\text{st}}{\geq}$ denotes stochastic dominance.

Applied to our setting, the decorrelation lemma gives us the following result.

Lemma 3.5. *For each i and directed simple path P from i to u, let $A_{i,P}$ be an independent draw from the distribution of A_i, let $P_i(T)$ be the unique path from i to u in tree T, and let $\overset{\text{st}}{\geq}$ denote stochastic dominance. Then*

$$\max_{T\in\mathcal{T}(u)} \ell_u^T\left((A_{i,P_i(T)})_{i\in T}\right) \overset{\text{st}}{\geq} \max_{T\in\mathcal{T}(u)} \ell_u^T(\mathbf{A}).$$

Theorem 3.1 now follows by observing that (3.1) holds also w.r.t. the arrivals $(A_{i,P})_{i\in[n],P\in\mathbf{T}_u}$. Then $\ell_u^{\mathbf{T}_u}\left((A_{i,P})_{i\in[n],P\in\mathbf{T}_u}\right)$ can be analyzed using Lemma 3.2 since we now have the required independence. Lemmas 3.3 and 3.5 complete the sequence of inequalities.

4 Stability of service for unit-length jobs

We now return to the temporal resource allocation problem and prove the stability of service theorem, Theorem 2.2, for the special case where each job has unit length, that is, $l_j = 1$ for all j. The non-unit length case is discussed in Section 5.

We fix ϵ as stated in the theorem, as well as the set of prices given by the fractional assignment lemma (Lemma 2.1). Then, for the asynchronous assignment process induced by these prices, we construct an instance of the network of servers setting discussed in Section 3 that satisfies the assumptions made in Theorem 3.1. Applying that theorem would then imply Theorem 2.2.

The obvious way to reduce from the temporal setting to the network of servers setting was described at the end of Section 2: construct a forwarding graph G over the set of all time slots $t \in \mathbb{Z}_+$ so that it contains all edges (t, t') that are in some job's preference order Π_j over time slots.[19] Unfortunately, the graph so defined can have unbounded in-degree. See Figure 3. Observe though that in this example, the path of every job that is forwarded to node t goes through the node $t + 1$. As such, each of these jobs is effectively forwarded from $t + 1$ to t. Taking inspiration from this example, we will proceed as follows. For every instantiation of job arrivals and preference orderings, we will define a canonical "shortcutting" of the

[19] In other words, t' is the next slot after t that some job j prefers, given that job's window and the slot prices.

jobs' paths, such that the overload status of every time slot is maintained. We will then show that the union of the shortcut paths over all possible instantiations defines a bounded degree graph. We can then apply Theorem 3.1.

We give a brief overview of this argument below. Details can be found in Appendix C.

The network of servers. We begin by defining a directed graph D on the set of all time slots $\mathbb{Z}_+$ as follows. For every time slot $t \in \mathbb{Z}_+$, define $\ell(t) = \max\{s < t : p_s \leq p_t\}$ and $r(t) = \min\{s > t : p_s < p_t\}$ to be the left and right "parents" of t. Let $E_F := \{(\ell(t), t) \cup (r(t), t) \quad \forall t \in \mathbb{Z}_+\}$; we call this the set of *forward* edges. Let $E_B := \{(b(t), t) \quad \forall t \in \mathbb{Z}_+\}$ where $b(t) = \min\{s > t : p_s = p_t\}$; we call this the set of *backward* edges. The directed graph D on vertex set $\mathbb{Z}_+$ is then defined as $D := (\mathbb{Z}_+, E_F \cup E_B)$. Observe that every node $t \in \mathbb{Z}_+$ in this graph has in-degree at most 3. Figure 4 illustrates the forward edges in this construction.

Let $\tilde{D}$ denote the graph formed by just the forward edges: $\tilde{D} = (\mathbb{Z}_+, E_F)$. For any $t \in \mathbb{Z}_+$, let $C(t)$ denote the *ancestors* of t in $\tilde{D}$, that is, $C(t) = \{s \text{ such that there is a path in } \tilde{D} \text{ from } s \text{ to } t\}$.

The reduction. We now consider the network of servers setting over the graph D, and describe a specific realization of jobs and paths for every realization of jobs and paths in the temporal setting. The set of arriving jobs and their entry nodes are the same in the two settings. We need to redefine the realized paths of the jobs to follow the edges in D.

Recall that in the temporal setting, each arriving job had a preference ordering Π_j over time slots in its window. We complete the description of Π_j by specifying how ties are broken: Π_j begins at the node $y_j \in \textsc{Fav}_j$ to which it is assigned in the fractional assignment[20] returned by Lemma 2.1. It then visits other nodes in $\textsc{Fav}_j$, if any, in a particular order: first, it visits all nodes $t \in \textsc{Fav}_j$ with $t < y_j$ in decreasing order of time, then it visits all nodes $t \in \textsc{Fav}_j$ with $t > y_j$ in increasing order of time. Having visited all of the least price slots in its window, the job then visits all slots of the next smallest price in its window in increasing order of time, and so on. See Figure 5 for an illustration.

Let P_j be the realized path of job j, namely, the prefix of Π_j from y_j to the node (call it z_j) where the job receives service or exits the process. We use P_j^1 to denote the prefix of this path which visits nodes $t \in \textsc{Fav}_j$ with $t < y_j$ in decreasing order of time; this always contains the node y_j. The remaining suffix of P_j, if non-empty, is denoted P_j^2. Observe that every edge in P_j^1 is a backward edge. However, edges in P_j^2 don't necessarily belong to D.

Let $\tilde{P}_j^2 = P_j^2 \cap C(z_j)$, in other words, we remove from P_j^2 all of the nodes that are not ancestors of z_j in the graph formed by the forward edges, $\tilde{D}$. The resulting path is a *short-cut* of the original path of the job. We now define $\tilde{P}_j$, a path from y_j to z_j, as follows. If $z_j \in P_j^1$, then $\tilde{P}_j := P_j^1$; otherwise, we define an appropriate prefix of P_j^1 called $\tilde{P}_j^1$, and set $\tilde{P}_j = \tilde{P}_j^1 \cup \tilde{P}_j^2$. Observe that $\tilde{P}_j$ is a short-cutting of P_j. Furthermore, every node that we short-cut in this process forwarded the job j, and is therefore overloaded. We can now prove the following two lemmas. See Figure 6 for an illustration of the short-cutting procedure.

Lemma 4.1. *Paths $\tilde{P}_j$ as defined above lie in the graph D.*

Lemma 4.2. *The collection of paths $\tilde{\mathcal{P}} = (\tilde{P}_j)$, as defined above, satisfies the min-work condition. Further, a node $t \in \mathbb{Z}_+$ is overloaded under the realized paths $\tilde{\mathcal{P}}$ if and only if it is overloaded under the realized paths $\mathcal{P}$. That is, $\ell_t(\mathcal{P}) \geq B_t$ if and only if $\ell_t(\tilde{\mathcal{P}}) \geq B_t$.*

[20] Note that this node may be a random variable, but is always among the favorite nodes of the job. The job is indifferent over all the nodes over which we tie-break.

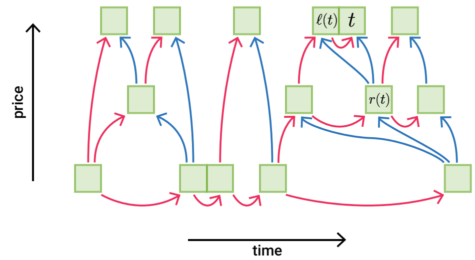


Figure 4: This figure shows the set of all forward edges in the directed, acyclic graph D on a set of time slots. Each time slot is represented by a green square, with its height indicating its price. Red edges go from $\ell(t)$ to t, and blue edges go from $r(t)$ to t for each t.

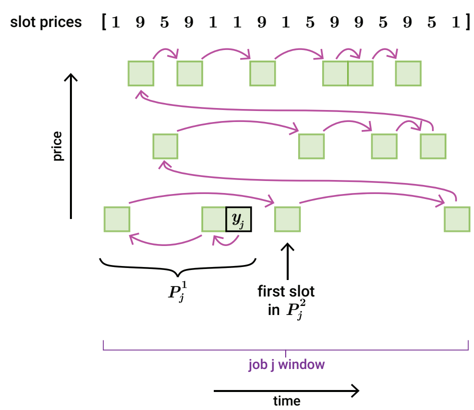


Figure 5: This figure shows the canonical path of a job over the time slots. The top line shows the prices of each of the time slots. The job enters at slot y_j. The decomposition of the path into P_j^1 and P_j^2 is also illustrated.

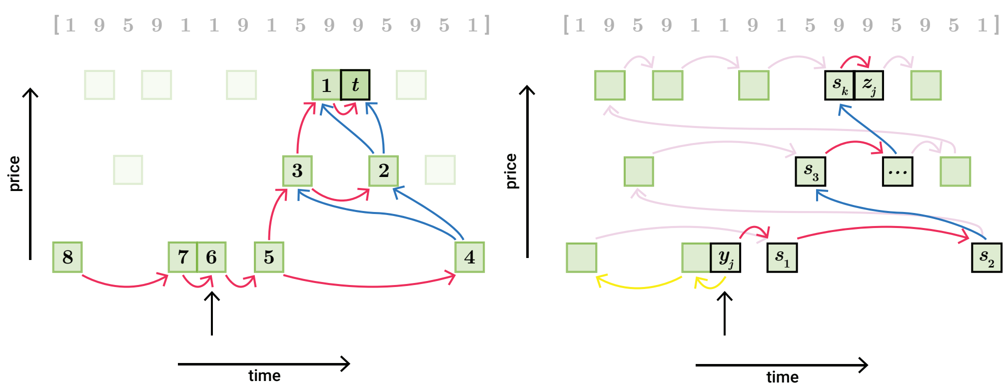


Figure 6: The figure on the left displays ancestors of t in $\tilde{D}$ numbered in reverse topological order. The figure on the right displays the shortcutting of the path from Figure 5. The yellow path is P_j^1, and the red and blue path is $\tilde{P}_j^2$. In this case, $\tilde{P}_j = \{y_j\} \cup \tilde{P}_j^2$.

We are now ready to prove the stability of service theorem for unit-length jobs. Observe that the instance of the network of items setting described above satisfies all of the properties required by Theorem 3.1. In particular, for every time slot t, the number of arrivals A_t is given by $\sum_j q_j \hat{X}_{j,t}$, where $\{X_{j,t}\}$ is the fractional assignment given by Lemma 2.1 and $\hat{X}_{j,t}$ is a Bernoulli random variable with expectation $X_{j,t}$. Therefore, it can be verified that $\mathbb{E}\left[e^{\epsilon(A_t - B_t)}\right] \leq \epsilon^2/3e$ for all t, and every job gets serviced with probability at least $(1-\epsilon)$ times its total fractional assignment.

5 Stability of service for arbitrary length jobs

We now turn to the temporal resource allocation problem for jobs of arbitrary lengths, and prove Theorem 2.2. As in Section 4, we fix any $\epsilon > 0$, and a set of prices for the time slots as given by Lemma 2.1 for this ϵ. Recall that $p_t(l)$ denotes the total price for l consecutive units of resource starting at time t. A job j of length l_j can choose to buy l_j or more consecutive units of resource depending on availability at these prices; we call these consecutive units "time blocks" and denote them by the pair (t, l) where t is the starting time of the block and l its length. The prices induce for each job j a preference ordering Π_j over time blocks (t, l) with $t \in W_j$ and $l \geq l_j$, and ties broken appropriately. As in the unit-length case, jobs search for the first available time block in their preference ordering in adversarial order. Lemma 2.1 guarantees that for every time slot t, the expected number of arriving jobs whose first block in their preference ordering starts at t is at most $(1-\epsilon)B_t$.

Correlation introduced by non-unit length jobs. As in Section 4 we can think of the movement of jobs as inducing paths in a graph over (starting) time slots. The challenge with non-unit length jobs is that when considering a block (t, l), they need to check the availability of the resource at each of l different slots; in other words, the forwarding decision for such jobs at slot t depends on loads at other neighboring slots, introducing extra correlations in the forwarding process. Alternately, we can think of the movement of jobs as inducing paths in a graph over *time blocks*. The challenge now is that we don't have a well defined notion of capacity; rather each time block shares capacity with other overlapping time blocks in a non-trivial manner.

Solution: capacity partitioning. We adopt the second approach. In order to overcome the challenge described above, we decouple capacity constraints at time blocks by artificially limiting the number of jobs assigned to any block. In particular, we assign a capacity of $\tilde{B}_{t,l}$ to time block (t, l). Once $\tilde{B}_{t,l}$ jobs have been assigned to block (t, l), even if there are available resources at all slots in the interval $[t, t+l-1]$, we admit no more jobs at this block. In order to respect the original capacity constraint at a time slot $t \in \mathbb{Z}_+$, the capacities $\tilde{B}_{t,l}$ must satisfy for all t the property that $\sum_{l \in [l_{\max}]} \sum_{t' \in [t-l+1, t]} \tilde{B}_{t',l} \leq B_t$.

Two issues remain: (1) How should the capacities be set to satisfy the above per-slot capacity constraints while obtaining good social welfare? (2) What process/graph does this induce over time blocks?

Setting the capacities. We set capacities based on the fractional assignment returned by Lemma 2.1. Let $\{X_{j,t}\}$ denote this fractional assignment. Then, we set $\tilde{B}_{t,l}$ to be equal to $\sum_{j:l_j=l} q_j X_{j,t}$ plus a reserve capacity of $\epsilon' B_t$ where $\epsilon' = \epsilon / l_{\max}^2$. It is immediate that the per-slot capacity constraints are satisfied: for all $t \in \mathbb{Z}_+$, we have,

$$\sum_{l \in [l_{\max}]} \sum_{t' \in [t-l+1, t]} \tilde{B}_{t',l} \leq \sum_j \sum_{t' \in [t-l_j+1, t]} q_j X_{j,t} + \epsilon' B_t l_{\max}^2 \leq (1-\epsilon)B_t + \epsilon B_t = B_t.$$

Furthermore, the fraction assignment of Lemma 2.1 gives a $(1-\epsilon)$-approximation to social welfare while respecting the block-wise capacity constraints in expectation.

Network over time blocks. We will think of the graph over time blocks as partitioned into $l_{\max}$ layers, with layer $\Gamma_l = \{(t,l)\}_{t\in\mathbb{Z}_+}$ corresponding to all blocks of length l. Within each layer, the induced subgraph is a graph over (starting) time slots. Each job's preference ordering, restricted to layer Γ_l, is identical to the preference ordering induced in the unit-length case when slot prices are given by $p_t(l)$. Accordingly, we define a network D_l over Γ_l in a manner analogous to the definition of network D in Section 4 with respect to prices $\{p_t(l)\}$: $D_l = (\Gamma_l, E_{F,l} \cup E_{B,l})$. Finally, let $E_L = \{((t,l),(t,l+1))\}_{t\in\mathbb{Z}_+, l\in[l_{\max}-1]}$ denote "inter-layer" edges that go from each block (t,l) to block $(t,l+1)$. Let $\mathcal{D} = (\cup_l \Gamma_l, \cup_l (E_{F,l}\cup E_{B,l}) \cup E_L)$.

Observe that the network $\mathcal{D}$ has maximum in-degree 4. We now argue that the realized path P_j of each job j can be "short-cut" into a path in the graph $\mathcal{D}$. Suppose that the realized path of a job j of length l_j starts at block (y_j, l_j) and terminates at block (z_j, l) for $l \ge l_j$. Observe that if $l > l_j$, prior to considering block (z_j, l), the job must have considered every block (z_j, l') with $l' \in [l_j, l-1]$; all of these blocks (z_j, l') are in P_j. Now, define the path $\tilde{P}_j$ in two parts as follows. The first part is a short-cut of the prefix of P_j from (y_j, l_j) to (z_j, l_j) defined over the layer Γ_{l_j} as in Section 4. The second part is a sequence of inter-layer edges connecting (z_j, l') to $(z_j, l'+1)$ for $l' \in [l_j, l-1]$.

It is easy to see that $\tilde{P}_j$ is a short-cut of P_j and lies in the graph $\mathcal{D}$. Corollary B.2 then implies that the collection of realized paths $\mathcal{P}' = (\tilde{P}_j)$ satisfies the min-work condition and Theorem 3.1 can be applied. It remains to argue that for every block (t,l), the moment generating function of $A_{t,l} - \tilde{B}_{t,l}$ is bounded, where $A_{t,l}$ is the random number of fresh arrivals at the block. Recall that $\tilde{B}_{t,l} = \sum_{j:l_j=l} q_j X_{j,t} + \epsilon' B_t$ where $\epsilon' = \epsilon/l_{\max}^2$. On the other hand, $A_{t,l} = \sum_{j:l_j=l} q_j \hat{X}_{j,t}$, where $\hat{X}_{t,l}$ is a Bernoulli variable with expectation $X_{j,t}$. So, we have

$$\mathbb{E}\left[e^{\epsilon'(A_{t,l}-\tilde{B}_{t,l})}\right] \le e^{-\frac{1}{2}\epsilon'^2 \tilde{B}_{t,l}} \le e^{-\frac{1}{2}\epsilon'^3 B_t} \le \epsilon^{c/2}$$

which for an appropriate constant c is at most $\epsilon^2/4e$. Here the second inequality used the fact that $\tilde{B}_{t,l} \ge \epsilon' B_t$, and the third used the lower bound on B_t from the statement of Theorem 2.2. Therefore, Theorem 3.1 applies and each job is accepted with probability at least $1-\epsilon'$. We achieve an approximation factor of $(1-\epsilon')(1-\epsilon) \ge 1-2\epsilon$ for social welfare.

This concludes the proof of Theorem 2.2.

Truthfulness and job payments. Truthfulness of the above mechanism is straightforward to argue: each job is allocated the cheapest block available that meets its requirements at the time of its arrival. Observe that a job of length l_j that is allocated block (t,l) for some $l > l_j$ must pay the price $p_t(l)$ (and not the cheaper price $p_t(l_j)$) in the above mechanism. It is, however, possible to modify our argument so that the theorem holds also when a job of length l_j can buy a slot (t,l) with $l > l_j$ at a price of $p_t(l_j)$. This change to the mechanism changes each job's preference ordering and realized path, but realized paths can once again be short-cut to form paths in $\mathcal{D}$, and we obtain the same conclusion as before. Finally, the new mechanism continues to be truthful with respect to jobs' lengths: a job paying $p_t(l_j)$ for some block (t,l) with $l > l_j$ is terminated after l steps, so it hurts to report a length smaller than the true length.

Acknowledgements

We are grateful to TJ Gilbrough for making almost all of the figures in this paper.

References

[1] Shipra Agrawal and Nikhil R Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1405–1424. SIAM, 2015. 5, A

[2] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014. 5, A

[3] Yossi Azar and Ety Khaitsin. Prompt mechanism for ad placement over time. In *International Symposium on Algorithmic Game Theory*, pages 19–30. Springer, 2011. 1, A

[4] Yossi Azar, Inna Kalp-Shaltiel, Brendan Lucier, Ishai Menache, Joseph Seffi Naor, and Jonathan Yaniv. Truthful online scheduling with commitments. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 715–732. ACM, 2015. 1, A

[5] Ran Canetti and Sandy Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal on Computing*, 27(4):993–1015, 1998. 1, A

[6] S. Chawla, J. Hartline, D. Malec, and B. Sivan. Sequential posted pricing and multi-parameter mechanism design. In *STOC*, 2010. 1

[7] Richard Cole, Shahar Dobzinski, and Lisa Fleischer. Prompt mechanisms for online auctions. In *International Symposium on Algorithmic Game Theory*, pages 170–181. Springer, 2008. 1, A

[8] Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *EC*, pages 71–78, 2009. 5, A

[9] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *EC*, pages 29–38, 2011. 5, A

[10] AK Erlang. On the rational determination of the number of circuits. *The life and works of AK Erlang*, pages 216–221, 1948. 1, A

[11] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *Proceedings of the 18th Annual European Conference on Algorithms: Part I*, ESA'10, pages 182–194, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15774-2, 978-3-642-15774-5. URL `http://dl.acm.org/citation.cfm?id=1888935.1888957`. 5, A

[12] Michal Feldman, Nick Gravin, and Brendan Lucier. Combinatorial auctions via posted prices. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 123–135. SIAM, 2015. 1, A

[13] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 982–991, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. 5, A

[14] David A Goldberg. On the steady-state probability of delay and large negative deviations for the $gi/gi/n$ queue in the halfin-whitt regime. *arXiv preprint arXiv:1307.0241*, 2013. A

[15] Mohammad T. Hajiaghayi, Robert D. Kleinberg, Mohammad Mahdian, and David C. Parkes. Online auctions with re-usable goods. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 165–174. ACM, 2005. 1, A

[16] Shlomo Halfin and Ward Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations research*, 29(3):567–588, 1981. 1, A

[17] Predrag Jelenković, Avishai Mandelbaum, and Petar Momčilović. Heavy traffic limits for queues with many deterministic servers. *Queueing Systems*, 47(1-2):53–69, 2004. 1, A

[18] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. In *STOC*, pages 303–312, 2014. 5, A

[19] Ron Lavi and Noam Nisan. Online ascending auctions for gradually expiring items. *Journal of Economic Theory*, 156:45–76, 2015. 1, A

[20] Konstantinos Psounis, Pablo Molinero-Fernández, Balaji Prabhakar, and Fragkiskos Papadopoulos. Systems with multiple servers under heavy-tailed workloads. *Performance Evaluation*, 62(1):456–474, 2005. A

[21] William J. Stevenson. *Operations Management*. McGraw-Hill Erwin, 2009. 1

[22] Ward Whitt. A diffusion approximation for the g/gi/n/m queue. *Operations Research*, 52(6):922–941, 2004. A

[23] Ward Whitt. Heavy-traffic limits for the g/h 2*/n/m queue. *Mathematics of Operations Research*, 30 (1):1–27, 2005. 1, A

[24] Yanhai Xiong, Jiarui Gan, Bo An, Chunyan Miao, and Yeng Chai Soh. Optimal pricing for efficient electric vehicle charging station management. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 749–757, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-4239-1. URL `http://dl.acm.org/citation.cfm?id=2937029.2937035`. 1

A Related Work

Mechanism design for online allocation. In online settings, a mechanism is called *prompt* if the payments are computed as soon as the job is scheduled, and is *tardy* if payments are computed at some later point in time (usually after the deadline of a job). The truthful online scheduling problem has been extensively studied in the worst case competitive analysis framework. Lavi and Nisan [19] introduced the problem of truthful online scheduling for unit length jobs on a single machine, with the social welfare objective, and showed that no deterministic mechanism that is truthful w.r.t. all the parameters can get an approximation ratio $< T$, where T is the time horizon. They proposed a weaker notion of truthfulness that they call set-Nash, and gave constant competitive mechanisms satisfying set-Nash. Hajiaghayi et al. [15] gave a tardy 2 approximation for unit length jobs. Their mechanism is truthful with the assumption of no early arrivals and late departures. They also extended this to an $O(\log l_{\max})$ approximation for jobs of different lengths, where $l_{\max}$ is the ratio of the maximum to minimum length of a job. Cole et al. [7] gave a prompt 2 approximation for unit length jobs, that is truthful only w.r.t. the value. They extended it to a prompt $O(\log l_{\max})$ approximation for different length jobs, that is truthful w.r.t the value and the deadline. Azar and Khaitsin [3] designed a prompt mechanism for unit length jobs with arbitrary *width*, on a single machine, that is a 6-approximation. The mechanism is truthful only w.r.t. the value. In a more recent work, Azar et al. [4] assumed that there is a lower bound s on the *slack* of each job, which is the ratio of the length of the [arrival, deadline] window to the job's length. They obtained a $2 + O(1/(\sqrt[3]{s} - 1)) + O(1/(\sqrt[3]{s} - 1)^3)$ approximation for arbitrary length jobs, via a mechanism that is truthful w.r.t all the parameters, under the assumption of no early arrival, no late departure, and no under-reporting of length. The mechanism is tardy, but can be modified to make decisions earlier with further assumptions on the slack. In the absence of slack, even algorithmically (i.e., with no truthfulness constraints), the online problem with arbitrary length jobs has a lower bound on the competitive ratio that is polylogarithmic in l or μ, where μ is the ratio of the largest to smallest possible values [5].

Other results: Although stated in terms of combinatorial auctions, the results of Feldman et al. [12] are relevant. They show how posted prices can achieve a truthful 2-approximation in combinatorial auctions with XOS biddders, in the *Bayesian* setting. This implies a 2-approximation for unit length jobs that is truthful w.r.t. all the parameters. The algorithmic problems of stochastic online matching and generalizations, under large budgets/capacities, are similar in spirit to the stochastic process we consider [1, 2, 8, 9, 11, 13, 18]. The temporal aspects of the two problems are very different, due to which standard models in that literature such as the random order model are not a good fit here.

Connections to queueing theory. Our models are closely related to standard models in queuing theory, when the demand and supply are stationary (i.e., not changing with time). In particular, for unit length jobs, suppose that $B_t = B$ for all t, the advertised prices are all equal, and every client tries to obtain service at the first slot in its window, failing which it moves its demand to the next time slot, and then the next, and so on. This case corresponds to the standard M/D/B queueing model, with Markovian arrivals, deterministic processing time, and B servers, under the first-come first-served (FCFS) queuing discipline.[21] We consider the regime where the rate of arrival of total work is $B - O(\sqrt{B})$. While we would like to analyze the probability of completion of any given job within its deadline, an easier quantity to compute, that is also an upper bound on this, is the probability that all B machines are busy, which is called the delay

[21] The notation for different queuing models is as follows: an A/B/C queue is one where the inter job arrival times are drawn from distributions in family A, the job lengths distributions belong to family B, and there are C identical machines. D is the class of deterministic distributions, M is the class of exponential distributions, GI is the class of general, independent distributions, GIB is the same class with a bounded support, and an H_2^* distribution is a mixture of an exponential and a point mass.

probability. This question was studied already by the seminal paper of Erlang [10], which initiated the study of queues. In particular Erlang's C model refers to an M/M/B queue and a closed form expression for the delay probability is derived. The importance of this regime was recognized by Halfin and Whitt [16], and is now called the Halfin-Whitt regime or the Quality and Efficiency Driven (QED) regime. This is because in this regime one can hope for high efficiency, which refers to a utilization ratio close to 1, and high quality, which refers to a delay probability close to 0. Halfin and Whitt [16] gave a formula for the delay probability of GI/M/B queues in this regime, and this was extended to H_2^* distributions (a mixture of exponential and a point mass) by [23]. Jelenković et al. [17] did the same for GI/D/B queues.

Good bounds on the delay probability for more general job length distributions are not known. In particular, it is open whether the delay probability is bounded above by δ for all job length distributions when the rate of work is at most $B - c\sqrt{B\log(1/\delta)}$ for some universal constant c. In fact it is not even known if such a bound holds for all distributions supported on $[0, L]$, when the rate of work is at most $B - \mathrm{poly}(L)\sqrt{B\log(1/\delta)}$. Whitt [22] gives heuristic approximations for the delay probability and other related quantities for GI/GI/B queues, and Psounis et al. [20] do the same for heavy tailed distributions, using an expression derived from a "bimodal" distribution. These are not proven theorems, but are rather shown to be good approximations via numerical analysis, or using simulations on traces of real workloads. The state of the art in this area is by Goldberg [14], who gives bounds on the delay probability as a limit of limits: the limit as $c \to \infty$, and as a function of c, the limit as $B \to \infty$, of the delay probability of GI/GI/queues with arrival rate of $B - c\sqrt{B}$. The convergence is not uniform, as the rate depends on both the arrival and job length distributions.

B Proofs for Section 3

B.1 The tree setting: proof of Lemma 3.2

We begin by proving that the stability of service theorem holds for the network of servers setting when the network is a tree.

Lemma 3.2. *Fix an ϵ in $[0, 1/2]$. Suppose that the network G is a finite directed tree, that is, it contains no directed cycles and every node has out-degree 1, and that the moment generating function of $A_i - B_i$ for each node i satisfies $\mathbb{E}\left[e^{\epsilon(A_i - B_i)}\right] \le \epsilon^2/ed_{\max}$. Then, for any i, $\Pr[\ell_i(\mathcal{P}) \ge B_i] \le \epsilon$.*

Proof. Recall that when the network is a tree, every node v, after processing the first B_v jobs that arrive at this node, forwards all of the remaining jobs to its parent. We call a node a *leaf* if it has no incoming edges. Order the nodes in the tree in topological order starting from the leaves. Let F_v be the number of jobs forwarded by node v to its parent, and let

$$F'_v := \max(A_v + \sum_{i=0}^{d} F_{u_i} - (B_v - 1), 0).$$

Clearly $\ell_v(\mathcal{P}) \ge B_v$ if and only if $F'_v > 0$. We will prove by induction over the topological ordering that

$$\mathbb{E}\left[e^{\epsilon F'_v}\right] \le \rho, \qquad \text{where} \qquad \rho = 1 + \frac{\epsilon^2}{d}.$$

The base case is a leaf (and follows from the argument below).

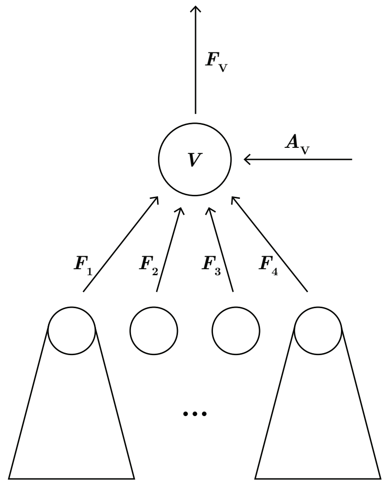


Figure 7: The induction step

For the induction step, consider a node v where up to d predecessors are $u_1, \ldots, u_d$. (See Figure 7). By the induction hypothesis and the fact that $F'_{u_i} \geq F_{u_i}$, we have

$$\mathbb{E}\Big[e^{\epsilon F_{u_i}}\Big] \leq \rho$$

Note that F_{u_i} and F_{u_j} are independent for each distinct i and j since the trees rooted at them are disjoint, and they are also independent of A_v. Thus, we have

$$\begin{aligned}
\mathbb{E}\Big[e^{\epsilon F'_v}\Big] &\leq \mathbb{E}\Big[e^{\epsilon \max(A_v + \sum_{i=0}^{d} F_{u_i} - B_v + 1, 0)}\Big] \\
&\leq 1 + e^{\epsilon}\mathbb{E}\Big[e^{\epsilon(A_v + \sum_{i=0}^{d} F_{u_i} - B_v)}\Big] \\
&\leq 1 + e^{\epsilon}\mathbb{E}\Big[e^{\epsilon(A_v - B_v)}\Big]\rho^d \\
&\leq 1 + \frac{e^{\epsilon}\epsilon^2}{ed}\rho^d \leq \rho,
\end{aligned}$$

for ρ defined as above. Here the last inequality follows by observing:

$$1 + \frac{e^{\epsilon}\epsilon^2}{ed}\left(1 + \frac{\epsilon^2}{d}\right)^d \leq 1 + \frac{e^{\epsilon}\epsilon^2}{ed} e^{d\ln\left(1+\frac{\epsilon^2}{d}\right)} \leq 1 + \frac{e^{\epsilon}\epsilon^2}{ed} e^{\epsilon^2} \leq 1 + \frac{\epsilon^2}{d},$$

as long as $\epsilon + \epsilon^2 \leq 1$. Letting

$$\eta_v := \mathbb{P}\big(F'_v > 0\big)\,,$$

and recalling that F'_v is integral, we have,

$$1 - \eta_v + \eta_v e^{\epsilon} \leq \mathbb{E}\Big[e^{\epsilon F'_v}\Big] \leq \rho.$$

Solving for η_v, we obtain

$$\eta_v(e^{\epsilon} - 1) \leq \rho - 1 = \frac{\epsilon^2}{d}$$

so

$$\eta_v \le \frac{\epsilon^2}{d(e^\epsilon - 1)} \le \frac{\epsilon^2}{d\epsilon} \le \frac{\epsilon}{d} \le \epsilon.$$

Therefore,

$$\mathbb{P}(\ell_v(\mathcal{P}) \ge B_v) = \mathbb{P}\big(F'_v > 0\big) \le \epsilon.$$

□

B.2 Reducing to a tree: proof of Lemma 3.3

We will now prove that for every instantiation of arrivals and forwards in the network of servers setting on G and every node u, we can find a subtree T of G rooted at u, such that the load at u becomes worse when the process is run over the tree T. Before we restate the main result of this section, let us recall some notation. Let a_i denote the realized number of jobs arriving at node i in G, and $\mathbf{a} = (a_i)$; let $\mathcal{P}$ denote the realized paths of jobs. Let $\mathcal{T}(u)$ denote the set of all directed subtrees of G rooted at node u, and for $T \in \mathcal{T}(u)$, let $\ell_i^T(\mathbf{a})$ denote the load on node i given the realized arrivals $\mathbf{a}$, when jobs that have not yet been served are routed along the tree T. (See Figure 1.)

Lemma 3.3. *If a fixed set of arrivals (resulting in a particular* $\mathbf{a}$*) and realized paths* $\mathcal{P}$ *overload a node* u *in the network* G*, then* $\exists$ *a tree* $T \in \mathcal{T}(u)$ *such that* u *is overloaded with the same set of arrivals when jobs are routed along* T*. Formally,*

$$\ell_u(\mathcal{P}) \ge B_u \quad \textit{implies that} \quad \max_{T\in\mathcal{T}(u)} \ell_u^T(\mathbf{a}) \ge B_u.$$

The proof of Lemma 3.3 proceeds in several steps.

Step 1: Remove cycles

Throughout the argument we will progressively modify the realized paths of jobs, while maintaining the invariant that every node i must process at least B_i jobs before forwarding any jobs. To this end, we say that a set of paths $\mathcal{P}' = (P'_j)$ is **valid for arrivals** $\mathbf{a}$ if there is an ordering of arrival and forwarding events for the realized jobs consistent with the arrivals $\mathbf{a}$, such that the realized path of each job j is exactly P'_j, and P'_j is a path in G.

For a directed multi-graph G', let $\mathrm{in}_{G'}(i)$ and $\mathrm{out}_{G'}(i)$ denote the in- and out-degrees, respectively, of node i in the multi-graph. We first show that a set of paths $\mathcal{P}'$ is valid for $\mathbf{a}$ if and only if the multi-graph given by the union of the paths, call it G', satisfies the following **min-work condition**:

$$\forall i, \quad \mathrm{out}_{G'}(i) \le \max(0, \mathrm{in}_{G'}(i) + a_i - B_i). \tag{B.1}$$

Claim 1. *A multi-graph* G' *can be decomposed into set of paths that is valid for arrivals* $\mathbf{a}$ *if and only if it satisfies the min-work condition* (B.1)*.*

Proof. The "only-if" direction of the statement follows trivially from the definition of valid paths. For the "if" direction, define

$$\mathrm{dep}_i := a_i + \mathrm{in}_{G'}(i) - \mathrm{out}_{G'}(i), \tag{B.2}$$

which is the number of departures at node i. (These departures can occur either because a job is processed at i or because its path terminates.) Clearly $\sum_i a_i = \sum_i \mathrm{dep}_i$. Construct an $s-t$ flow network, where there is an edge of capacity a_i from s to node i, an edge of capacity dep_i from node i to t, and an edge from i to

j of capacity equal to the number of edges in G' from i to j. Clearly, there is an integer flow in which each edge is filled to capacity, and therefore, this flow can be decomposed into paths.

Moreover, the paths are trivially consistent with the property that each node processes $\min(\text{dep}_i, B_i)$ jobs, since by (B.1), any i with positive outdegree satisfies

$$0 < \text{out}_{G'}(i) \leq \text{in}_{G'}(i) + a_i - B_i,$$

or in other words,

$$\text{dep}_i = a_i + \text{in}_{G'}(i) - \text{out}_{G'}(i) \geq B_i.$$

□

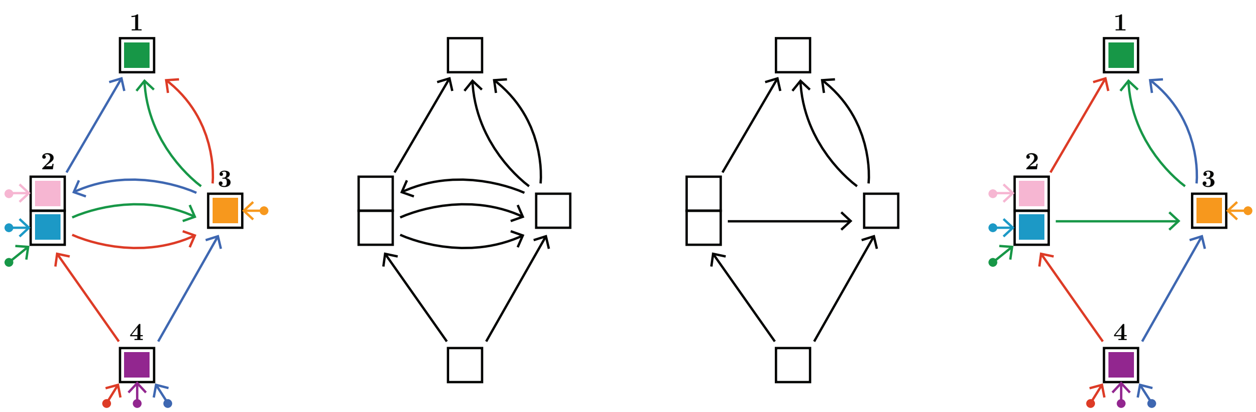
Corollary B.1. *Let* $\mathbf{a} = (a_i)$ *be a set of arrivals and* $\mathcal{P}$ *a set of paths that are valid for* $\mathbf{a}$*. Let* G' *be the multigraph obtained by taking the union of paths in* $\mathcal{P}$*. Successively remove directed cycles from* G' *to obtain a new acyclic multigraph* G''*. Then* G'' *can be decomposed into a set of paths* $\mathcal{P}'$ *valid for the arrivals* $\mathbf{a}$*, with the property that a node* i *is overloaded under* $\mathcal{P}$ *if and only if it is overloaded under* $\mathcal{P}'$*. That is, for all* i*,*

$$\ell_i(\mathcal{P}) \geq B_i \longrightarrow \ell_i(\mathcal{P}') \geq B_i \tag{B.3}$$

and

$$\ell_i(\mathcal{P}) < B_i \longrightarrow \ell_i(\mathcal{P}') < B_i.$$

Proof. Prior to the removal of cycles, the paths $\mathcal{P}$ satisfied the property (B.1). Since removing a cycle preserves the min-work property (B.1) and the resulting graph is acyclic, by Claim 1, its edges decompose into a valid set of realized paths. This is the set of paths $\mathcal{P}'$. See Figure 8.



(a) (b) (c) (d)

Figure 8: Figure (a) shows a set of arrivals and paths taken. Figure (b) is the induced multigraph G'. Figure (c) shows the multigraph G'' obtained after removing cycles from G' (arrivals are not shown). Figure (d) shows a decomposition of G'' into set of valid paths, as per Claim 1. (This decomposition is not unique.) All overloaded nodes are still overloaded.

Clearly the in-degree of each node is weakly decreasing. Furthermore, nodes with out-degree $= 0$ do not belong to any cycles, and so their loads don't change. So we only need to show that (B.3) holds for nodes with out-degree > 0 after removing cycles. This follows from the fact that the out-degree of a node in G' is an upper bound on the number of cycles removed that it is part of. By (B.1), $\text{out}_{G'}(i) \geq k$ implies that $\text{in}_{G'}(i) + a_i \geq B_i + k$.

Therefore, if k cycles through queue i are removed during the process of removing cycles, the final load at i (in-degree plus arrivals) is at least B_i, so (B.3) holds. □

Via a similar argument, we also obtain the following corollary that will be used in the proof of Lemma 4.2. It allows us to "short-cut" paths without affecting overload events.

Corollary B.2. *Let $\mathbf{a} = (a_i)$ be a set of arrivals and $\mathcal{P}$ a set of realized paths valid for $\mathbf{a}$. Consider a path $P \in \mathcal{P}$ of length at least 2, and let (u_1, u_2) and (u_2, u_3) be two consecutive edges in this path. Let P' be obtained by removing (short-cutting) the vertex u_2 from P. That is, $P' = P \setminus \{(u_1, u_2), (u_2, u_3)\} \cup \{(u_1, u_3\}$. Then, the new set of paths $\mathcal{P}' = \mathcal{P} \setminus \{P\} \cup \{P'\}$ is valid for $\mathbf{a}$. Furthermore, for all nodes i, $\ell_i(\mathcal{P}) \geq B_i$ iff $\ell_i(\mathcal{P}') \geq B_i$.*

Step 2: Modify paths to obtain tree

For the rest of this subsection, we will assume that we are given the arrivals $\mathbf{a}$, a set of valid paths $\mathcal{P}$ that form a directed acyclic graph, and a specific node u. Let G' denote the multi-graph formed by taking the union of the paths in $\mathcal{P}$. We complete the proof of Lemma 3.3 by modifying the paths so that they are directed along a tree rooted at u, without decreasing the load on u. The modified paths will remain valid.

To this end, we will repeatedly use the following two operations:

Operation 1: Remove an edge (i, j) if j has out-degree 0, and $j \neq u$.

Operation 2: Suppose that P_1 and P_2 are two edge-disjoint paths that start at i and end at j, and there is a path from j to u. Delete path P_2 and replace it by a duplicate copy of P_1.

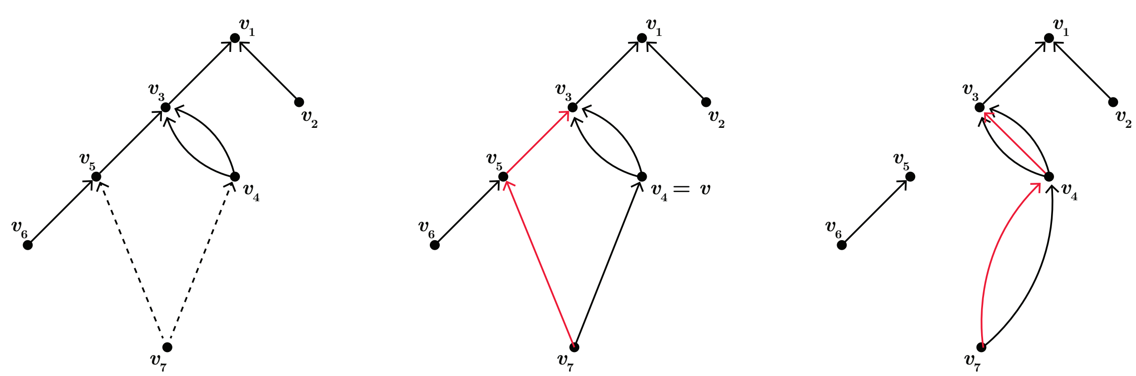


Figure 9: This figure shows an example of the application of inductive step when $i = 7$. Initially (left figure) all paths from $v_1, \ldots, v_6$ form a tree directed towards v_1, and we are about to process v_7 which has edges to v_4 and v_5. Operation 2 is applied to the paths $P_1 = (v_7, v_4, v_3)$ and $P_2 = (v_7, v_5, v_3)$ resulting in the graph shown on the right. Subsequently, an application of operation 1 removes the edge (v_6, v_5).

If we begin with a set of paths valid for $\mathbf{a}$, then by Claim 1, operations 1 and 2 preserve the existence of a set of realized paths that are valid for the arrivals $\mathbf{a}$. Indeed, operation 1 reduces the out-degree of a node. Operation 2 has the following properties:

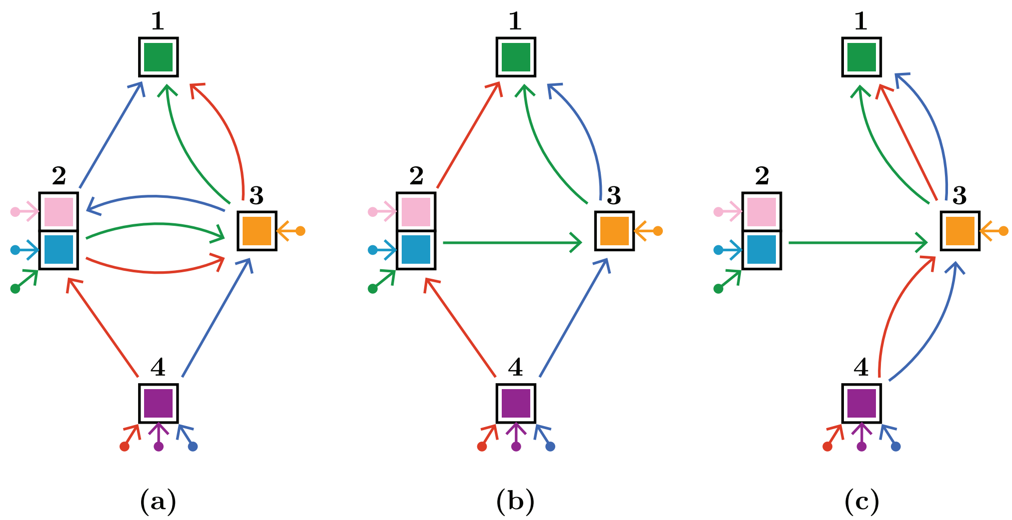


Figure 10: This figure shows the transformations applied to convert the paths into valid paths along a tree (in this case tree T_3 from Figure 1). Going from the left graph to the middle is the result of removing cycles. The right figure shows the paths obtained once we apply Step 2, which modifies paths to obtain a tree. In this example, the red path was rerouted to go through 3 rather than 2. The load at node 1 is preserved in the transformation from the middle routing to the routing on the right.

- It reduces both the in-degree and out-degree of some nodes by 1 (every node on P_2 except for i and j), which preserves (B.1).
- It increases both the in-degree and out-degree of some nodes by 1 (nodes on P_1 except for i and j), but only nodes that already had out-degree 1, which also preserves (B.1).
- It maintains the out-degree of i and in-degree of j.

We apply these two operations to get our tree as follows: Recall that G' is acyclic, and consider the nodes in G' in topological order (from sinks to sources), say $v_1, \ldots, v_n$. Let $S_i := \{v_1, \ldots, v_i\}$. We inductively apply the above operations so that the subgraph on S_i consists of a collection of paths terminating at u, for which the corresponding graph (not multigraph) is a tree directed towards and rooted at u. (This tree could be empty.)

The base case is $i = 0$ (or the empty set). To extend from S_{i-1} to S_i, we do the following: If all of the out-edges from v_i are to nodes with out-degree 0, then remove all of these edges (applying operation 1). Otherwise, suppose that v_i has an edge to some vertex $v \in S_{i-1}$ from which there is a path P to u. Pick such a v, and the associated path P. Repeat the following two steps until S_i satisfies the inductive hypothesis:

1. As long as there is an edge (v', v'') with $v'' \in S_i \setminus u$, and $\mathrm{out}_G(v'') = 0$, apply operation 1 to remove this edge.
2. If there is an edge (v_i, v') where $v' \neq v$ and there is a path P' from v' to u, find the first node j at which the paths P and P' intersect. Let $P_{v,j}$ be the prefix of P terminating at j, and let $P_{v',j}$ be the prefix of P' terminating at j. Apply operation 2 to the paths $P_1 := (v_i, P_{v,j})$,and $P_2 := (v_i, P_{v',j})$. See Figure 9. (Note that if there is no path P' from v' to u, then by the inductive hypothesis, v' has out-degree 0, which means the edge (v_i, v') will be removed.)

This argument completes the proof of Lemma 3.3.

B.3 The decorrelation lemma

We will now prove the decorrelation lemma that is needed in our analysis. We use the notation $\overset{\text{st}}{\geq}$ to denote stochastic dominance.

Lemma 3.4. (Decorrelation lemma for the max function) *Let $g_\ell : \Re \to \Re$, $\ell = 1, \ldots, k$ be any non-decreasing functions, X be any real valued random variable, and let $Y_1, \ldots, Y_k$ be independent and identically distributed random variables from the same distribution as X. Then,*

$$\max_\ell\{g_\ell(Y_\ell)\} \overset{\text{st}}{\geq} \max_\ell\{g_\ell(X)\}.$$

Proof. Since the Y_ℓ's are independent, for any a,

$$\Pr[\max_\ell\{g_\ell(Y_\ell)\} \leq a] = \prod_\ell \Pr[g_\ell(Y_\ell) \leq a], \tag{B.4}$$

whereas, recalling that g_ℓ is non-decreasing for each ℓ, and setting $x^*(a) = \min_\ell g_\ell^{-1}(a)$,[22]

$$\Pr[\max_\ell\{g_\ell(X)\} \leq a] = \Pr[X \leq x^*(a)] = \min_\ell \Pr[g_\ell(X) \leq a]. \tag{B.5}$$

Clearly, the RHS of (B.5) is larger for all a than the RHS of (B.4). Therefore, the lemma follows. □

We can further generalize the decorrelation lemma as follows.

Lemma B.3. *Let $h_k : \Re^n \to \Re$, $k = 1, \ldots, N$ be functions that are non-decreasing in each variable. Let $A_1, \ldots, A_n$ be independent (but not identically distributed random variables) and, for each $1 \leq i \leq n$, let $P_i : [N] \to [n_i]$, for nonnegative integers n_i. Then,*

$$\max_{1\leq k\leq N}\{h_k(\{A_{i,P_i(k)}\}_{i=1}^n)\} \overset{\text{st}}{\geq} \max_{1\leq k\leq N}\{h_k(\{A_i\}_{i=1}^n)\}. \tag{B.6}$$

where each $A_{i,P_i(k)}$ is an independent draw from the distribution of A_i.

Proof. We prove by induction on j that

$$\max_{1\leq k\leq N}\left[h_k\left(\{A_{i,P_i(k)}\}_{i=1}^j, \{A_i\}_{i=j+1}^n\right)\right] \overset{\text{st}}{\geq} \max_{1\leq k\leq N}\{h_k\left(\{A_{i,P_i(k)}\}_{i=1}^{j-1}, \{A_i\}_{i=j}^n\right)\}.$$

The base case of $j = 0$ is immediate. For the induction step, condition on all variables other than A_j (which are independent of A_j), and define

$$h'_k(X) := h_k\left(\{A_{i,P_i(k)}\}_{i=1}^{j-1}, X, \{A_i\}_{i=j+1}^n\right) |\{A_{i,P_i(k)}\}_{i=1}^{j-1} \text{ and } \{A_i\}_{i=j+1}^n.$$

Thus, it suffices to show that

$$\max_{1\leq k\leq N}\left[h'_k\left(A_{j,P_j(k)}\right)\right] \overset{\text{st}}{\geq} \max_{1\leq k\leq N}\{h'_k\left(A_j\right)\}. \tag{B.7}$$

Letting

$$f_\ell(A_j) := \max_{1 \le k \le N \text{ s.t. } P_j(k)=\ell} h'_k(A_j),$$

showing (B.7) is the same as showing that

$$\max_{1 \le \ell \le n_j} f_\ell(A_{j,\ell}) \overset{\text{st}}{\ge} \max_{1 \le \ell \le n_j} f_\ell(A_j),$$

which follows directly from Lemma 3.4. $\square$

B.4 Reducing to a tree of trees: proof of Lemma 3.5

Lemma 3.5. *For each i and directed simple path P from i to u, let $A_{i,P}$ be an independent draw from the distribution of A_i, $P_i(T)$ be the unique path from i to u in tree T, and $\overset{\text{st}}{\ge}$ denote stochastic dominance. Then*

$$\max_{T \in \mathcal{T}(u)} \ell_u^T\left((A_{i,P_i(T)})_{i \in T}\right) \overset{\text{st}}{\ge} \max_{T \in \mathcal{T}(u)} \ell_u^T(\mathbf{A}).$$

Proof. Apply Lemma B.3 with N equal to the number of distinct trees T rooted at u, n equal to the number of queues in the queueing network, $h_k(\mathbf{A}) := \ell_u^{T_k}(\mathbf{A})$, i.e., the load on u when the each job follows the routes given by tree T_k until it is processed, and n_i equal to the number of distinct simple paths from i to u. $\square$

C Proofs for Section 4

C.1 Properties of the graph D

We begin by proving properties of the directed graph D. Recall that D contains three types of edges. For every time slot $t \in \mathbb{Z}_+$, the "left forward" edges connect t's left parent $\ell(t)$ to t; the "right forward" edges connect t's right parent $r(t)$ to it. The "backward" edges are defined as $E_B := \{(b(t), t) \quad \forall t \in T\}$ where $b(t) = \min\{s > t : p_s = p_t\}$. Recall that every node $t \in \mathbb{Z}_+$ in this graph has in-degree at most 3. (See Figure 4.)

Recall that $C(t)$ denotes the ancestors of time slot t in $\tilde{D}$ i.e. $C(t) = \{s \text{ such that there is a path in } \tilde{D} \text{ from } s \text{ to } t\}$.

Lemma C.1. *(a) For any t, either there is an edge in $\tilde{D}$ from $\ell(t)$ to $r(t)$ or vice versa. That is, either $\ell(t) = \ell(r(t))$ or $r(t) = r(\ell(t))$.*

(b) For any t, the set $C(t)$ is totally ordered: if $t_1, t_2 \in C(t)$, then either $t_1 \in C(t_2)$ or $t_2 \in C(t_1)$.

Proof. (a) See proof in caption of Figure 11.

(b) Proof by induction on $k = |C(t)|$. For all t with $|C(t)| = 1$, the statement is immediate. For $k > 1$, suppose wlog that $\ell(t) = \ell(r(t))$. Then all ancestors of t (other than t itself) are ancestors of $r(t)$, and therefore, the total order on $C(t)$ must terminate with $r(t)$ followed by t. Applying the induction hypothesis to $C(r(t))$ completes the argument. See Figure 6 for an example of the total ordering. $\square$

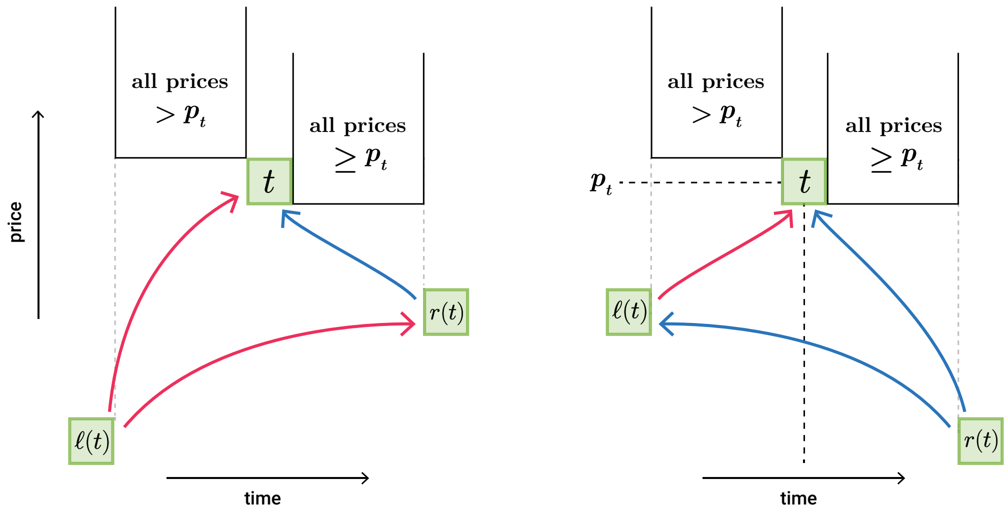


Figure 11: For a particular time slot t, $\ell(t)$ is the largest time less than t where the price is at most p_t, and $r(t)$ is the smallest time greater than t at which the price is strictly less than p_t. Notice that if $p_{\ell(t)} \leq p_{r(t)}$, then by definition $\ell(t) = \ell(r(t))$. The right hand side shows the case where $p_{\ell(t)} > p_{r(t)}$. When this happens, then by definition $r(t) = r(\ell(t))$.

C.2 Properties of the paths $\tilde{P}_j$: proof of Lemmas 4.1 and 4.2

Recall that we define job j's new path $\tilde{P}_j$ as follows. Let y_j and z_j denote the first and last nodes on the job's realized path P_j. Recall that P_j^1 denotes the prefix of P_j which visits nodes $t \in \text{FAV}_j$ with $t < y_j$, starting at y_j, and P_j^2 denotes the remaining suffix of the path. We begin with a simple property of the suffix P_j^2.

Lemma C.2. *Let t be a node in the path P_j^2 for some job j. Then, either $\ell(t)$ or $r(t)$ belongs to the job's window W_j, and appears before t on the realized path P_j.*

Proof. If neither $\ell(t)$ or $r(t)$ are in job j's window, $W_j = [s_j, d_j]$, then all prices in the window are at least p_t, and all prices in $[s_j, t-1]$ are strictly larger than p_t. See Figure 12. Since the path P_j starts at a cheapest slot in the window, it must start at a slot of price p_t, at a time t or later. That is, $t \in \text{FAV}_j$ with $y_j \geq t$. If the path doesn't start at t itself, then by the definition of P_j^1, t belongs to P_j^1. Finally, if one of $\ell(t)$ or $r(t)$ lies in W_j, then it is easy to see that job j must visit this slot before it visits t: either the price at this slot is smaller than that of t, or, in the case of $\ell(t)$, the prices are the same, but the job visits slots to the left of t with prices p_t before t. □

We now complete our description of the reduction from the temporal setting to the network setting by specifying the paths $\tilde{P}_j$ for each realized job j. Recall that we define $\tilde{P}_j^2 = P_j^2 \cap C(z_j)$ to be a "short-cutting" of the suffix P_j^2. Let s_1 be the first node on the path $\tilde{P}_j^2$. By Lemma C.2, one of the parents of s_1 lies in P_j. Call this parent s_0. Since s_0 appears before s_1 in P_j, it must be the case that $s_0 \in P_j^1$. Let $\tilde{P}_j^1$ be the prefix of P_j^1 from y_j to s_0. Define $\tilde{P}_j = \tilde{P}_j^1 \cup \tilde{P}_j^2$ if P_j^2 is non-empty, and $\tilde{P}_j = P_j^1$ otherwise.

We will now prove that the new paths $\tilde{P}_j$ lie in the graph D. (See Figure 6.)

Lemma 4.1. *Paths $\tilde{P}_j$ as defined above lie in the graph D.*

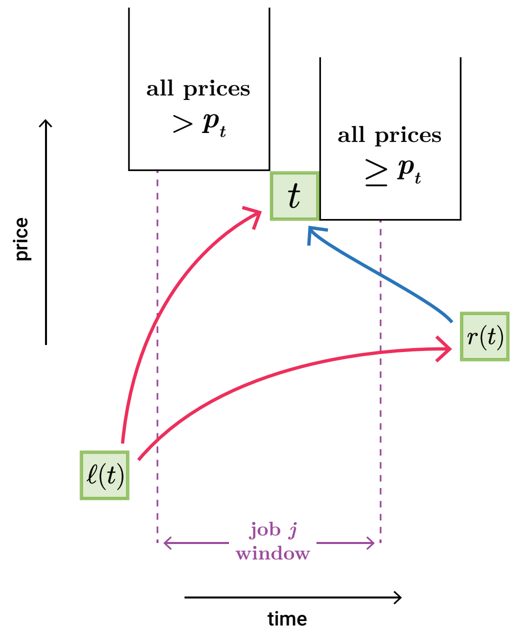


Figure 12: This figure illustrates the proof of Lemma C.2.

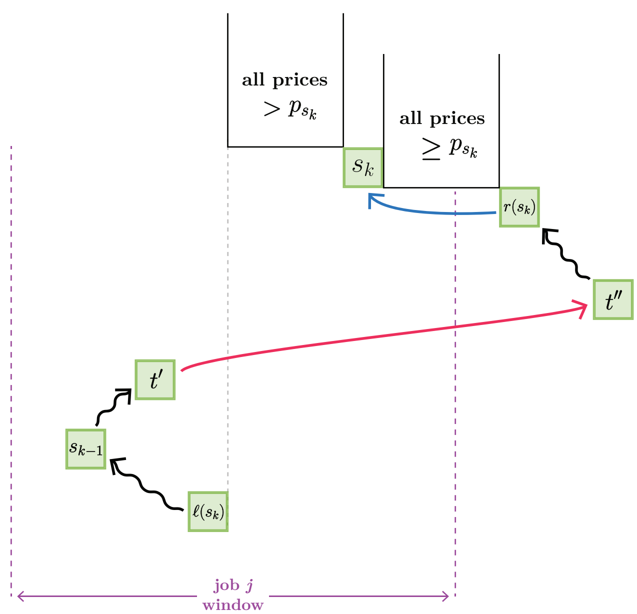


Figure 13: This figure illustrates the contradiction in the proof of Lemma 4.1. Time t' is less than time $\ell(s_k)$ and $p_{t'} > p_{\ell(s_k)}$. This contradicts the fact that $t' = \ell(t'')$.

Proof. Consider a job j. The path $\tilde{P}_j$ consists of three components: (1) a prefix of the path P_j^1, (2) the edge (s_0, s_1), where s_1 is the first node on $\tilde{P}_j^2$ and s_0 is its ancestor on P_j^1, and, (3) the path $\tilde{P}_j^2 = P_j^2 \cap C(z_j)$. Observe that the edges in path P_j^1 are all backward edges. Therefore, they lie in the graph D. The edge (s_0, s_1) lies in $\tilde{D}$ by construction, and by recalling that Lemma C.2 implies that one of the parents of s_1 lies in P_j^1. We will now focus on the path $\tilde{P}_j^2 = P_j^2 \cap C(z_j)$. Let $\tilde{P}_j^2 = \{s_1, s_2, \cdots, s_k\}$, for some $k \geq 1$ where $s_k = z_j$. We claim that for all $i \in \{1, \cdots, k-1\}$, (s_i, s_{i+1}) is an edge in $\tilde{D}$.

We prove the claim by induction on the length k of $\tilde{P}_j^2$: Consider the last node s_k in $\tilde{P}_j^2$. Suppose, without loss of generality, that $\ell(s_k) = \ell(r(s_k))$. We will prove that either $s_{k-1} = \ell(s_k)$ or $s_{k-1} = r(s_k)$. Once this is proved, we simply use the fact that there is an edge (s_{k-1}, s_k) in $\tilde{D}$, and complete the argument by applying the inductive hypothesis to $(s_1, \ldots, s_{k-1})$.

Suppose then that s_{k-1} is not $r(s_k)$ or $\ell(s_k)$. Since all ancestors of s_k except s_k itself are ancestors of $r(s_k)$, and P_j doesn't go through $r(s_k)$, it must be that $r(s_k)$ is outside the job's window W_j. Thus, we have:

- s_{k-1} is an earlier time than $\ell(s_k)$: this follows from the fact that all prices in $[\ell(s_k)+1, r(s_k)-1]$ are too high to come before s_k in P_j^2; see Figure 13.
- $\ell(s_k)$ is visited by j prior to s_{k-1}: by Lemma C.2, since $r(s_k) \notin P_j$, and $s_k \in P_j^2$, it must be that $\ell(s_k) \in P_j$ and is visited by P_j before s_k, and therefore also before s_{k-1}.
- The price at $\ell(s_k)$ is strictly smaller than that at s_{k-1}: if the prices at the two slots were equal, noting that $s_{k-1} < \ell(s_k)$ would imply $s_{k-1} \in P_j^1$, but we know that $s_{k-1} \in P_j^2$.
- There is a path Q in $\tilde{D}$ from $\ell(s_k)$ to s_{k-1} to $r(s_k)$ to s_k: this follows from the total order on ancestors of s_k, because all of these nodes (including s_{k-1} by virtue of it being in $\tilde{P}_j^2$) are ancestors of s_k.
- The path Q contains an edge (t', t'') where $t' < \ell(s_k)$ and $t'' \geq r(s_k)$.

But the final observation yields a contradiction: since the edge (t', t'') goes left to right, it would have to be that $t' = \ell(t'')$. But that can't be, since $\ell(t)$ also has price less than $p_{t''}$ and is further to the right. See Figure 13. This completes the proof. □

Lemma 4.2. *The collection of paths $\tilde{\mathcal{P}} = (\tilde{P}_j)$, as defined above, satisfies the min-work condition. Further, a node $t \in \mathbb{Z}_+$ is overloaded under the realized paths $\tilde{\mathcal{P}}$ if and only if it is overloaded under the realized paths $\mathcal{P}$. That is, $\ell_t(\mathcal{P}) \geq B_t$ if and only if $\ell_t(\tilde{\mathcal{P}}) \geq B_t$.*

Proof. This lemma follows immediately by repeatedly applying Corollary B.2.

□

C.3 Proof of the stability of service theorem for unit-length jobs

We are now ready to prove Theorem 2.2 for the special case of unit-length jobs. Lemmas 4.1 and 4.2 imply that the (random) collection of paths $\tilde{\mathcal{P}}$ forms a valid instance of the network of servers setting with graph D. It remains to argue that the moment generating function of $A_t - B_t$ for every node t is small. Let $\{X_{j,t}\}$ be the fractional assignment given by Lemma 2.1 for ϵ picked in the statement of the theorem. Recall that

$\sum_j q_j X_{j,t} \le (1-\epsilon)B_t$ and $A_t = \sum_j q_j \hat{X}_{j,t}$, where $\hat{X}_{j,t}$ is a Bernoulli random variable with expectation $X_{j,t}$. Then we have:

$$\begin{aligned}\mathbb{E}\left[e^{\epsilon(A_t-B_t)}\right] &\le e^{-\frac{1}{2}\epsilon^2 B_t}\\ &\le \epsilon^{c/2} < \epsilon^2/3e\end{aligned}$$

for an appropriate choice of c. We can therefore apply Theorem 3.1 and the unit-length case of Theorem 2.2 follows.

D The expected case LP: Proof of Lemma 2.1

Lemma 2.1. (Fractional assignment lemma) *Fix any set of potential jobs J, their arrival probabilities, and the capacities B_t for all $t \in \mathbb{Z}_+$. Then for any $\epsilon \ge 0$, $\exists$ nonnegative prices $(p_t)_{t\in\mathbb{Z}_+}$ and a fractional assignment $X_{j,t} \in [0,1]$ from jobs $j \in J$ to their favorite slots $t \in \text{FAV}_j$, such that,*

1. *Every job that can afford to pay the price at its favorite slot is fully scheduled: for every j with $p_t(l_j) < v_j$ for $t \in \text{FAV}_j$, we have $\sum_{t\in\text{FAV}_j} X_{j,t} = 1$.*
2. *The expected allocation at time t is at most $(1-\epsilon)B_t$: $\forall t, \sum_{j\in J, t'\in[t-l_j+1,t]} q_j X_{j,t'} \le (1-\epsilon)B_t$.*
3. *The expected social welfare is at least $(1-\epsilon)$ times the optimum: $\sum_{j\in J, t\in\text{FAV}_j} v_j q_j X_{j,t} \ge (1-\epsilon)\text{OPT}$.*

Further, if the distribution is periodic, the prices are also periodic with the same period, and can be computed efficiently.

Proof. We begin by writing a linear program for the fractional assignment problem. The variables in the program correspond to the fractional assignment of jobs j to slots t, x_{jt}, with the interpretation that if job j arrives, it is assigned with probability x_{jt} to the interval of time $[t, t+l_j-1]$. Fractional assignments must satisfy two constraints: (1) every job is assigned with total probability at most 1; (2) the expected number of jobs assigned to a slot t is at most B_t. Together these constraints ensure that the optimal solution to the LP satisfies the last two requirements of the lemma. The prices we select are based on writing a dual for the program.

The linear program and its dual are given below. In the description below, we assume that the time horizon for the allocation process is given by $[H]$ for some large H. In the case (discussed below) where jobs are drawn from a periodic distribution, we allow H to go to infinity. Recall that W_j denotes the window of starting times for a job; taking the time horizon into account, it is defined as $W_j = [s_j, \min(d_j, H) - l_j + 1]$.

Primal LP

$$\text{Maximize} \sum_{j, t\in W_j} v_j q_j x_{jt} \text{ s.t.}$$

$$\begin{aligned}
&\sum_j \sum_{t'\in[t-l_j+1,t]\cap W_j} q_j x_{jt'} \le B_t(1-\epsilon) && \forall t\in[H]\\
&\sum_{t\in W_j} x_{jt} \le 1 && \forall j \in J\\
&x_{jt} \ge 0 && \forall j\in J, t\in[H]
\end{aligned}$$

Dual LP

$$\text{Minimize} \sum_t \lambda_t B_t(1-\epsilon) + \sum_j \mu_j \text{ s.t.}$$

$$\begin{aligned}
&q_j \sum_{t'\in[t,t+l_j-1]} \lambda_{t'} + \mu_j \ge v_j q_j && \forall j\in J, t\in W_j\\
&\mu_j \ge 0 && \forall j\in J\\
&\lambda_t \ge 0 && \forall t\in[H]
\end{aligned}$$

Complementary Slackness conditions (CS). Let (x^*, λ^*, μ^*) denote the optimal solutions to the Primal and Dual programs. Define $p_t := \lambda_t^*$, and observe that $p_t(l) = \sum_{t' \in [t, t+l-1]} \lambda_{t'}^*$. The following complementary slackness conditions hold:[23]

1. For all $j \in J, t \in W_j$, either $x_{jt}^* = 0$ or $p_t(l_j) + \mu_j^*/q_j = v_j$.
2. For all $j \in J$, either $\mu_j^* = 0$ or $\sum_{t \in W_j} x_{jt}^* = 1$.

The first CS condition, along with the fact that $p_{t'}(l_j) + \mu_j^*/q_j \geq v_j$ for all $t' \in W_j$ and $j \in J$, implies that if a job is assigned (partially) to a starting slot t, i.e., $x_{jt}^* > 0$, then $p_t(l_j) \leq p_{t'}(l_j)$ for all $t' \in W_j$. In other words, jobs can only be assigned to one or more of their favorite slots.

Furthermore, if for a job $j \in J$, there exists a time $t \in W_j$ with $p_t(l_j) < v_j$, then we must have $\mu_j^* > 0$. Thus, the second CS condition implies that $\sum_{t \in W_j} x_{jt}^* = 1$ and the job is fully scheduled.

This completes the proof of the lemma. □

Periodicity. We now prove that for periodic instances, we can efficiently find periodic prices satisfying the conditions of the lemma.

We begin by defining periodic instances. A periodic instance with period $k \in \mathbb{Z}_+$ is given by a core set J_0 of potential jobs and their probabilities. Let the set J_i be obtained by shifting all the jobs in J_0 by ki time units:

$$\forall i \in \mathbb{Z}_+, J_i := \{(s_j + ki, d_j + ki, l_j, v_j) : j \in J_0\}.$$

The full set of potential jobs is defined to be

$$J = \bigcup_{i=0}^{\infty} J_i.$$

The associated probability for each job is the same as that for the corresponding job in the core set J_0. Furthermore, supply is also periodic with period k: for all $t = t' \pmod{k}$, $B_t = B_{t'}$. A special case is that of *i.i.d. distributions*, which are simply periodic distributions with period 1.

We will show next that for periodic instances, the above LP can be simplified into a compact form. The compact LP assigns jobs in J_0 to time slots in $[k]$, with jobs and their windows "wrapping around" the interval k. In particular, if $d_j - l_j + 1 \leq k$, we define the window of a job, as before, to be $\tilde{W}_j = [s_j, d_j - l_j + 1]$. Otherwise, if $d_j - l_j + 1 > k$ and the length of the window, $|W_j| = d_j - l_j - s_j + 2$, is smaller than k, we define $\tilde{W}_j = [s_j, k] \cup [1, s_j + |W_j| - k]$. Finally, if $|W_j| \geq k$, we define $\tilde{W}_j = [k]$.

Likewise, for a job j scheduled fractionally at a time $t \in [k]$, if $t + l_j - 1 > k$, the job "wraps around" the interval $[k]$ (potentially multiple times), and places load on slots $t' \in [0, t + l_j - 1 - k]$. Specifically, a job with $l_j \geq k$ places a load of $\lfloor \frac{l_j}{k} \rfloor$ (times its fractional allocation) on *every* slot in $[k]$, and an extra unit of load on $t' \in [k]$ such that $(t' - t) \bmod k$ is less than or equal to $(l_j - 1) \bmod k$. For example, if a job of length 5 in a setting with period $k = 3$ starts at time slot 1, it places a load of 2 units on each of slots 1 and 2, and a load of 1 unit on slot 3; if instead it starts at time slot 3, then it places a load of 2 units on each of slots 3 and 1, and a load of 1 unit on slot 2.

Accordingly, we obtain the following LP. To understand the capacity constraint in this LP, consider the example of a setting with period 3, and a job j of length 5 with window $[1, 3]$. Then, the load placed by this

[23] The condition corresponding to λ_t^* is not relevant to the proof of the lemma.

job on slot 1 is $2q_jx_{j1} + q_jx_{j2} + 2q_jx_{j3}$.

$$\text{Maximize } \frac{H}{k} \sum_{j \in J_0, t \in \tilde{W}_j} v_j q_j x_{jt} \text{ s.t.}$$

$$\sum_{j \in J_0} \sum_{t' \in [k] \cap \tilde{W}_j} q_j \left\lfloor \frac{l_j}{k} \right\rfloor x_{jt'} + \sum_{j \in J_0} \sum_{\substack{t' \in \tilde{W}_j: \\ (t-t') \bmod k < l_j \bmod k}} q_j x_{jt'} \le B_t(1-\epsilon) \qquad \forall t \in [k]$$

$$\sum_{t \in \tilde{W}_j} x_{jt} \le 1 \qquad \forall j \in J_0$$

$$x_{jt} \ge 0 \qquad \forall j \in J_0, t \in [k]$$

Let OPT_P and OPT_A be the optimal values of the periodic and aperiodic LPs given above, respectively. We now show that as H goes to infinity, $OPT_P \ge OPT_A$, and therefore using OPT_P as a benchmark can only make our approximation factor worse than what it should actually be. Observe also that the dual of the periodic LP has a price variable for every slot $t \in [k]$ — the interpretation is that prices repeat with periodicity k and $p_{t'} = p_t$ for all $t' = t \pmod k$.

Claim 2. $OPT_P \ge OPT_A$.

Proof. For simplicity we will assume that H is a multiple of k. For a job $j \in J_0$ and $j' \in J_i$ for some $i \in \mathbb{Z}_+$, we say that j is congruent to j', written $j \cong j'$, if j' is obtained by shifting the arrival time and deadline of j by a multiple of k. For $t \in [k]$, let $S_t = \{t' \in [H] : t' = t \pmod k\}$. Observe that $|S_t| = H/k$.

Let x^*_{jt} be the optimal solution to the aperiodic LP for the set of all jobs J. Consider the following solution $\forall t \in [k]$ and $j \in J_0$: $x^\dagger_{jt} = \frac{k}{H} \sum_{j' \cong j} \sum_{t' \in S_t} x^*_{j't'}$.

The value obtained by the solution $x^\dagger$ in the periodic LP is exactly equal to the value obtained by x^* in the aperiodic LP. We will now prove that $x^\dagger$ is feasible for the periodic LP, which implies the lemma.

Since x^* is feasible for the aperiodic LP, we have for all $j \in J_0$:

$$\sum_{t \in \tilde{W}_j} x^\dagger_{jt} = \sum_{t \in \tilde{W}_j} \frac{k}{H} \sum_{j' \cong j} \sum_{t' \in S_t} x^*_{j't'} = \frac{k}{H} \sum_{j' \cong j} \sum_{t \in \tilde{W}_j} \sum_{t' \in S_t} x^*_{j't'} = \frac{k}{H} \sum_{j' \cong j} \sum_{t' \in W_{j'}} x^*_{j't'} \le \frac{k}{H} \sum_{j' \cong j} 1 = 1.$$

Likewise, we have for all $t \in [k]$:

$$\sum_{t' \in S_t} \sum_{j} \sum_{t'' \in [t'-l_j+1, t'] \cap W_j} q_j x^*_{jt''} \le \frac{H}{k} B_t(1-\epsilon)$$

Rearranging the sum, we get,

$$\sum_{j \in J_0} q_j \frac{k}{H} \sum_{j' : j' \cong j} \sum_{t' \in S_t} \sum_{t'' \in [t'-l_j+1, t'] \cap W_j} x^*_{j't''} \le B_t(1-\epsilon)$$

Consider the inner sum $\sum_{t' \in S_t} \sum_{t'' \in [t'-l_j+1, t'] \cap W_j} x^*_{j't''}$. If $l_j < k$, this sum is exactly equal to the sum over $t' \in \tilde{W}_j$ with $(t-t') \bmod k < l_j \bmod k$ of $\sum_{t'' \in S_{t'}} x^*_{j't''}$. If $l_j > k$, then each t'' belongs to the interval $[t' - l_j + 1, t']$ for $\left\lfloor \frac{l_j}{k} \right\rfloor$ additional different $t' \in S_t$, and so, we get an additional term of $\sum_{t' \in [k]} \sum_{t'' \in S_{t'}} \left\lfloor \frac{l_j}{k} \right\rfloor x^*_{j't''}$. Putting these expressions together, we get,

$$\sum_{j \in J_0} q_j \frac{k}{H} \sum_{j' : j' \cong j} \left(\sum_{\substack{t' \in \tilde{W}_j : \\ (t-t') \bmod k < l_j \bmod k}} \sum_{t'' \in S_{t'}} x^*_{j't''} + \sum_{t' \in [k]} \sum_{t'' \in S_{t'}} \left\lfloor \frac{l_j}{k} \right\rfloor x^*_{j't''} \right) \leq B_t(1 - \epsilon),$$

or,

$$\sum_{j \in J_0} \sum_{t' \in [k] \cap \tilde{W}_j} q_j \left\lfloor \frac{l_j}{k} \right\rfloor x^\dagger_{jt'} + \sum_{j \in J_0} \sum_{t' \in [t - (l_j \bmod k) + 1, t] \cap \tilde{W}_j} q_j x^\dagger_{jt'} \leq B_t(1 - \epsilon)$$

Thus, the $x^\dagger_{jt}$'s form a feasible solution to the periodic LP and the lemma holds. □