Appendix A

Convexity in Queueing Systems

In this chapter we examine a work conserving $*/*/1$ queue and develop fundamental monotonicity and convexity properties of unfinished work and packet waiting time in the queue as a function of the packet arrival rate λ. The "*/*" notation refers to the fact that the input process has arbitrarily distributed and correlated interarrival times and packet lengths. (This differs from the standard GI/GI description, where interarrival times and packet lengths are independent and identically distributed).[1] The arrival process consists of the superposition of two component streams: an arbitrary and uncontrollable background input of the $*/*$ type, and a rate-controllable packet stream input (Fig. A-1). The rate-controllable stream contains a collection of indistinguishable $*/*$ substreams, and its rate is varied in discrete steps by adding or removing these substreams as inputs to the queue. We show that any moment of unfinished work is a convex function of this input rate. Under the special case of FIFO service, we show that waiting time moments are also convex.

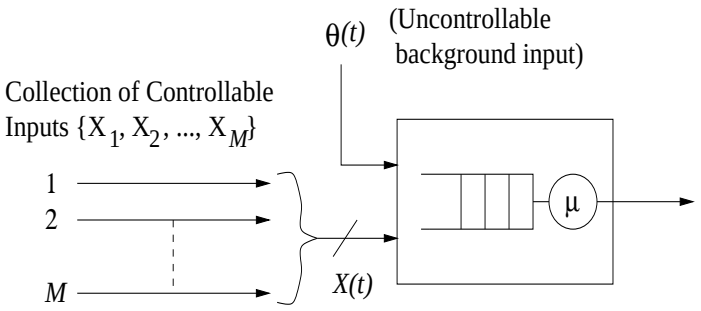

Figure A-1: A work conserving queue with server linespeed μ, a $*/*$ background input $\theta(t)$, and rate-controllable $*/*$ inputs $X(t) = \{X_1(t), \ldots, X_M(t)\}$.

[1] We avoid the ambiguous notation $G/G/1$, as different texts use this to mean either $*/*/1$ or $GI/GI/1$.

We then extend the convexity result to address the problem of optimally routing input streams over a parallel collection of N queues with different linespeeds $(\mu_1, ..., \mu_N)$. We show that cost functions consisting of convex combinations of unfinished work moments in each of the queues are convex in the N-dimensional rate tuple $(\lambda_1, ..., \lambda_N)$. In the symmetric case where the N queues are weighted equally in the cost function and have identical background processes, this convexity result implies that the uniform rate allocation minimizes cost. In the case of an asymmetric collection of N parallel queues, we present a sequentially greedy routing algorithm that is optimal.

The convexity results and optimization methods are extended to treat queues with time-varying linespeeds $(\mu_1(t), ..., \mu_N(t))$. We show that the amount of unprocessed bits in the multi-queue system remains convex in the input rate vector $(\lambda_1, ..., \lambda_N)$. However, we demonstrate that waiting times are not necessarily convex for general time varying linespeed problems. For simplicity of exposition, we postpone the time-varying analysis until Section A.6.

Convexity properties of single and parallel collections of queues have been developed previously with respect to various parameters and with various assumptions about the nature of the input processes and service time processes. In [82], queues with a large number of i.i.d. input streams are analyzed in the large deviations regime, and the buffer requirement to meet a specified overflow rate is shown to be convex in the server rate μ. The result is extended in [83] to address more general input processes. In [59] a convexity theory of "multi-modular functions" is developed and used to establish an optimal admission control sequence for arrivals to an exponential server, given that a fixed ratio of packets must be accepted. Extensions to queues and tandems with fixed batch arrivals are treated in [4] [2].

In [112] [101], the authors analyze the expected packet occupancy in tree networks of deterministic service time queues. It is shown that expected occupancy of any interior queue of the tree is a concave function of the multiple exogenous input rates, while occupancy in queues on the edge of the network is shown to be convex. Convexity properties of parallel $GI/GI/1$ queues with packet-based probabilistic "Bernoulli" routing are developed in [56] [25] [66], where it is shown under various models and independence assumptions that backlog moments in each queue are convex functions of the Bernoulli splitting probability, and hence uniform Bernoulli splitting minimizes expected backlog in homogeneous systems

among the class of all Bernoulli splittings. A related result for homogeneous systems in [80] shows that uniform splitting is optimal for *arbitrary arrivals* in a system of parallel queues with equal exponential servers (i.e., $*/M$ inputs), and stochastically minimizes the total workload at every instant of time.

Our treatment of the convexity problem for streams of $*/*$ inputs is an important feature, since packets from a single source often must be routed together to maintain predictability and to prevent out-of-order delivery. Such situations occur, for example, when we have N streams carrying voice traffic from individual users. In this case, convexity properties are considered with respect to the integer number of streams routed to a queue.[2] However, we also treat the packet-based routing method of [56] [25] [66] [80] in a more general (yet simpler) context. Rather than emphasizing the differences between packet-based and stream-based routing, we discover a fundamental similarity. We consider packet-based routing of a general $*/*$ input stream whose rate can be split according to a continuous rate parameter, using a splitting method such as the probabilistic "Bernoulli" splitting in [56] [25] [66] [80]. We find that convexity for this general packet-based routing problem is a consequence of our stream-based results.

Our analysis is carried out by introducing a new function of the superposition of two input streams that we call the blocking function. Properties of the blocking function are developed by examining sample paths of work conserving queues, and each property corresponds to an intuitive comparison of two different queueing system configurations. These properties are then used to establish the convexity and optimal routing results. As a special case of this analysis applied to the problem of routing over a homogeneous set of queues, we obtain the uniform splitting optimality result of [56] [25] [66] under the more general context of $*/*$ inputs (as well as time varying server rates). The analytical techniques used to prove the results in this chapter are novel and can likely be used in other contexts.

In the next section, we review related work in the area of convexity and monotonicity in queueing systems. In Section A.2 we define the blocking function. In Section A.3 we establish convexity properties of unfinished work and waiting time with respect to a discrete rate parameter corresponding to adding an integer number of indistinguishable $*/*$ streams as inputs to a queue. Convexity properties in terms of a continuous rate splitting parameter

[2]This problem is related to general admission control problems, where convexity properties are essential to establishing the structural properties of an optimal policy, see [128].

are developed in Section A.4, and in Section A.5 we consider example applications to the problem of optimal routing over a parallel set of queues. Time varying server rates are treated in Section A.6.

A.1 Related Work on Monotonicity and Convexity in Queueing Systems

It is often possible to use stochastic inequalities to compare the relative performance of two different queueing systems, even in cases where the exact performance of both systems is unknown or has no closed form analytical expression. For example, Stoyan develops a simple monotonicity result in [129], where it is shown that if a Markov process $X_1(t)$ has a step-by-step transition probability distribution that is *stochastically larger*[3] than the corresponding transition distribution for another process $X_2(t)$, and if the initial value $X_1(0)$ for the first process is greater than or equal to the initial value of the second, then at *any particular instant of time* t^* the random variable $X_1(t^*)$ is stochastically greater than or equal to the random variable $X_2(t^*)$. For queueing systems, such Markov processes correspond to queues with Poisson inputs and independent service times (so called "$M/GI/1$" queues), and the monotonicity theorem establishes the classical result that if two queues with identical exponential servers and service rates λ_1 and λ_2 are compared, then *at every instant of time* the number of packets in the first queue is stochastically less than the number in the second whenever $\lambda_1 \leq \lambda_2$ and the initial backlog of the first queue is less than or equal to that of the second.

Ross develops a similar monotonicity result in [119] by using the $\max(a-b, 0)$ operator to propogate waiting time distributions in a $GI/GI/1$ queue that is initially empty. It is shown that if two queues with the same arrival and service rates are compared, the waiting times in the first queue will be stochastically less than the waiting times in the second whenever the interarrival times and service times of the first queue are stochastically less bursty than the second. Baccelli and Brémaud present similar monotonicity results with respect to the unfinished work in a $GI/GI/1$ queue [see Ex. 4.2.6, pp.283 in [7]], and to the waiting time process in a queue where successive arrival and service time vectors (α_n, σ_n)

[3] A random variable X is said to be *stochastically greater* than a random variable Y if $Pr[X > z] \geq Pr[Y > z]$ for all z. See [119] for an illuminating treatment of the subject of stochastic dominance and convex ordering relations.

are independent but perhaps differently distributed for all n [see Ex. 4.4.7, pp.306 in [7]]. Similar statements for systems driven by doubly-stochastic Poisson processes are given in [26]. These results demonstrate that *regular* (less bursty) sequences are better in terms of reducing queue congestion. Indeed, in [64] it is shown that any moment of waiting time in a general $*/*/1$ queue, that is, a queue *general ergodic arrivals and packet lengths*, is greater than the corresponding moment if the arrival process is replaced by a periodic stream of the same rate, or if the packet lengths are replaced by fixed lengths equal to the sequence average.

A related result by Hajek in [59] establishes the optimal packet acceptance rule for an arrival stream (with independent interarrival times) to a queue with an exponential server, in the case when a fixed ratio of packets must be accepted. The result is generalized to non-exponential servers in [4] [2].

The effect of regularizing traffic is used in [45] to show that round robin routing is the optimal "backlog-blind" strategy for routing an *arbitrary arrival sequence* to two parallel queues with homogeneous exponential servers (the authors also show that *Join the Shortest Queue* routing is optimal for backlog-aware strategies). Similar round robin optimality results are shown in [90] for routing to a parallel set of homogeneous queues with i.i.d. service times that have an "increasing failure rate" property, and in [89] for queues with random and homogeneous server outages. Likewise, round robin is shown to be optimal in [3] for routing to parallel homogeneous tandems under a service time independence assumption—namely, that the service times of a packet routed to tandem 1 are independent of the service times it would experience if routed to tandem 2. We note that round robin is not optimal for general input processes with potentially correlated packet lengths. Indeed, consider a Poisson stream arriving to a system of two queues with identical server rates. If packet lengths alternate between short packets and long packets, then round robin is among the worst of all possible routing strategies. Similar counterexamples can be constructed for stationary service time processes where successive packet lengths are negatively correlated.

Discussions of various routing policies, including backlog-unaware probabilistic routing and backlog-aware Join-the-Shortest-Queue routing are provided in [17]. Shortest queue results similar to [45] are developed in [145] [146] for multiple homogeneous queues, in [135] for homogeneous queues with finite buffers, and in [110] for heterogeneous queues with finite buffers. Load balancing techniques for various symmetric systems with memoryless

properties of either service times, interarrival times, or both are considered in [145] [25] [80] [56]. Approximation techniques are developed in [32] for heterogeneous systems, and approaches to optimal routing using convex optimization are described in [14] [21] [130] [19] [22].

The prior works that are most directly related this chapter are the convexity results given in [25] [56] [66] for queues with independence assumptions, and the uniform Bernoulli routing result of [80] developed for general arrival streams to homogeneous exponential servers. These results are based on the theory of majorization and Schur-convex functions, described, for example, in [123] [24]. Our approach is quite different and enables general analysis of both stream based routing as well as a large class of probabilistic splitting methods (including Bernoulli splitting), and does not require independence assumptions on the input traffic. Furthermore, our analysis is self-contained and is based on easily-understood properties of queueing systems.

A.2 The Blocking Function For $*/*/1$ Queues

Consider a work conserving queue with a single server that can process packets at a constant line speed of μ bits/second (Fig. A-2). Variable length packets from input stream X flow into the queue and are processed at the single server according to any work-conserving service discipline (e.g., FIFO, LIFO, Shortest Packet First, GPS, etc.). The input stream is characterized according to two random processes: (i) The sequence $\{a_k\}$ of inter-arrival times, and (ii) The sequence $\{l_k\}$ of packet lengths.

We assume the processes $\{a_k\}$ and $\{l_k\}$ are ergodic with arrival rate λ and average packet length $\mathbb{E}\{L\}$, respectively. In general, inter-arrival times may be correlated with each other as well as jointly correlated with the packet length process. We maintain this generality by describing the input to the queue by the single random process $X(t)$, which represents the amount of bits brought into the queue as a function of time. As shown in Fig. A-2, a particular input $X(t)$ is a non-decreasing staircase function. Jumps in the $X(t)$ function occur at packet arrival epochs, and the amount of increase at these times is equal to the length of the entering packet.

For a given queue with input process $X(t)$, we define the *unfinished work process* $U_X(t)$ as the total amount of unprocessed bits in the queueing system (buffer plus server) as

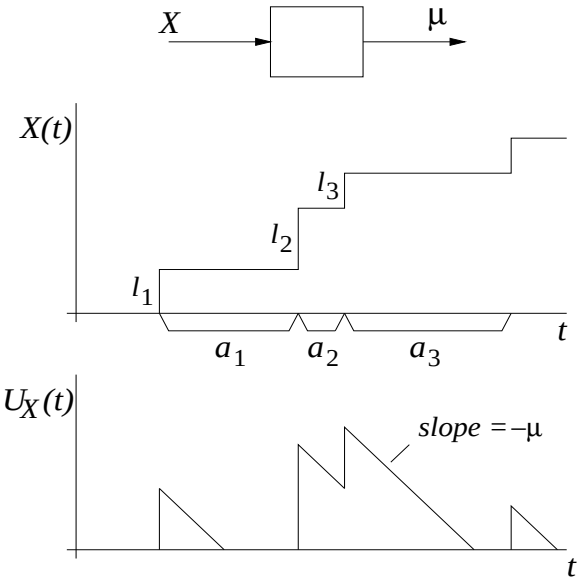

Figure A-2: A work conserving $*/*/1$ queue, and typical sample paths of accumulated and unfinished work.

a function of time. Note that for a system with a processor of rate μ and an amount of unfinished work $U_X(t)$, the quantity $U_X(t)/\mu$ represents the amount of time required for the system to empty if no other packets were to arrive. We assume the queue is initially empty at time $t = 0$. It is clear that $U_X(t)$ is the same for all work conserving service disciplines. It is completely determined by $X(t)$ as well as the server linespeed μ. An example unfinished work function $U_X(t)$ is shown in Fig. A-2. Notice the triangular structure and the fact that each new triangle emerges at packet arrival times and has a downward slope of $-\mu$.

We define the *superposition* of two input streams $X_1(t)$, $X_2(t)$ as the sum process $X_1(t)+X_2(t)$. We make the following sample path observation, which holds for any arbitrary set of sample paths:

Observation 1: For all times t, we have:

$$U_{X_1+X_2}(t) \geq U_{X_1}(t) + U_{X_2}(t) \tag{A.1}$$

Thus, for any two inputs X_1 and X_2, the amount of unfinished work in a work conserving queueing system with the superposition process $X_1 + X_2$ is always greater than or equal to the sum of the work in two identical queues with these same processes X_1 and X_2 entering them individually. This is illustrated in Fig. A-3.

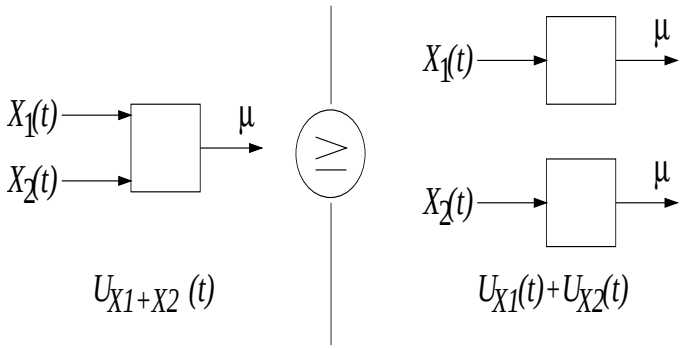

Figure A-3: A queueing illustration of the non-negativity property of the blocking function.

Proof. (Observation 1) We compare the two system configurations of Fig. A-3. Since $U_{X_1+X_2}(t)$ is the same for all work conserving service disciplines, we can imagine that packets from the X_1 stream have preemptive priority over X_2 packets. The queueing dynamics of the X_1 packets are therefore unaffected by any low priority packets from the X_2 stream. Thus, the $U_{X_1+X_2}(t)$ function can be written as $U_{X_1}(t)$ plus an extra amount $extra\_X_2(t)$ due to the X_2 packets, as shown in Fig. A-4. This extra amount (represented as the striped region in Fig. A-4) can be thought of as the amount of unfinished work remaining in a queue with the X_2 input stream alone, where the server goes on idle "vacations" exactly at times when $U_{X_1}(t)$ is nonzero. Clearly, this unfinished work is greater than or equal to the unfinished work there would be if the server did not go on vacations—which is $U_{X_2}(t)$. Thus:

$$U_{X_1+X_2}(t) = U_{X_1}(t) + extra\_X_2(t) \geq U_{X_1}(t) + U_{X_2}(t)$$

□

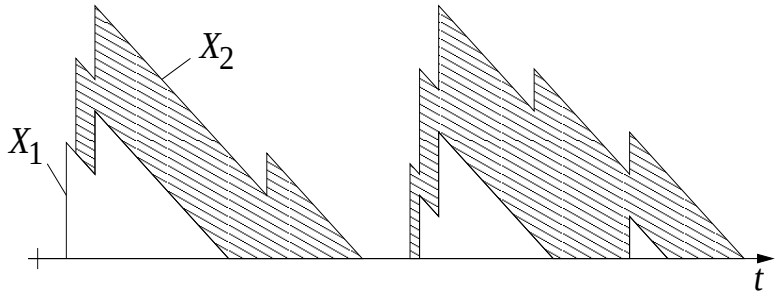

Figure A-4: An example sample path of the unfinished work function $U_{X_1+X_2}(t)$ in a system where X_1 packets have preemptive priority.

This observation places a lower bound on the unfinished work $U_{X_1+X_2}(t)$ in terms of its

component streams. We note that Baccelli et. al. give an *upper* bound in terms of different component streams [see prop. 2.11.6, pp. 164 in [7]]. Specifically, in [7] it is shown that the unfinished work at the time of the n^{th} arrival is less than or equal to the unfinished work at the time of the k^{th} arrival plus the unfinished work at the time of the n^{th} arrival in a system where the first $n-k$ original arrivals are removed, for any $k \in \{1, \ldots, n\}$.

The simple observation (A.1) motivates the following definition:

Definition 1. *The* Blocking Function $\beta_{X_1,X_2}(t)$ *between two streams* X_1 *and* X_2 *is the function:*

$$\beta_{X_1,X_2}(t) \triangleq U_{X_1+X_2}(t) - U_{X_1}(t) - U_{X_2}(t) \tag{A.2}$$

Thus, the blocking function is a random process that represents the extra amount of unfinished work in the system due to the blocking incurred by packets from the X_1 stream mixing with the X_2 stream.

Lemma 1. *The blocking function has the following properties for all times* t*:*

$$\begin{aligned}
&\beta_{X_1,X_2}(t) \geq 0 && \textit{(Non-negativity)}\\
&\beta_{X_1,X_2}(t) = \beta_{X_2,X_1}(t) && \textit{(Symmetry)}\\
&\beta_{X_1+X_2,X_3}(t) \geq \beta_{X_1,X_3}(t) && \textit{(Monotonicity)}
\end{aligned}$$

The non-negativity lemma above is just a re-statement of (A.1), while the symmetry property is obvious from the blocking function definition. Below we prove the monotonicity property.

Proof. (Monotonicity) From the definition of the blocking function in (A.2), we find that the monotonicity statement is equivalent to the following inequality at every time t:

$$U_{X_1+X_2+X_3}(t) - U_{X_1+X_2}(t) - U_{X_3}(t) \geq U_{X_1+X_3}(t) - U_{X_1}(t) - U_{X_3}(t)$$

Cancelling and shifting terms, it follows that we must prove:

$$U_{X_1+X_2+X_3}(t) + U_{X_1}(t) \geq U_{X_1+X_2}(t) + U_{X_1+X_3}(t) \tag{A.3}$$

We have illustrated (A.3) in Fig. A-5. We thus prove that the sum of the unfinished work in Systems A and B of Fig. A-5 is greater than or equal to the sum in A' and B'.

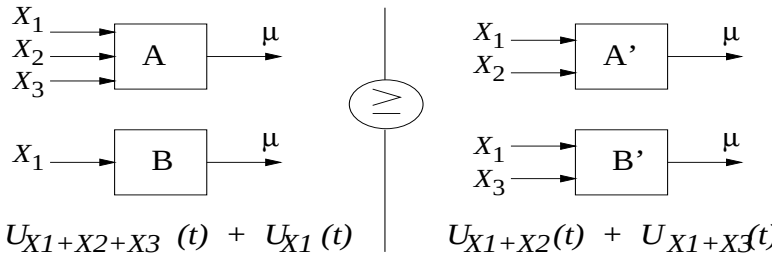

Figure A-5: A queueing illustration of the monotonicity property of the blocking function.

In a manner similar to the proof of Observation 1, we give packets from both the X_1 and X_2 streams preemptive priority over X_3 packets. The queues of Fig. A-5 can thus be treated as having servers that take "vacations" from serving X_3 packets during busy periods caused by the other streams. Comparing the A and A' systems, as well as the B and B' systems, we have:

$$U_{X_1+X_2+X_3}(t) = U_{X_1+X_2}(t) + extra\_in\_System\_A(t) \tag{A.4}$$

$$U_{X_1+X_3}(t) = U_{X_1}(t) + extra\_in\_System\_B'(t) \tag{A.5}$$

where $extra\_in\_System\_A(t)$ represents the amount of unfinished work from X_3 packets in a queue whose server takes vacations during busy periods caused by the X_1 and X_2 streams. Likewise, $extra\_in\_System\_B'(t)$ represents the amount of unfinished work from X_3 packets when vacations are only during X_1 busy periods. Since busy periods caused by the X_1 stream are subintervals of busy periods caused by the combined $X_1 + X_2$ stream, the X_3 packets in System A experience longer server vacations, and we have:

$$extra\_in\_System\_A(t) \geq extra\_in\_System\_B'(t) \tag{A.6}$$

Using (A.4)-(A.6) verifies (A.3) and concludes the proof. □

Intuitively interpreted, the monotonicity property means that the amount of blocking incurred by the (X_1+X_2) process intermixing with the X_3 process is larger than the amount incurred by the X_1 process alone mixing with the X_3 process.

These three lemmas alone are sufficient to develop some very general convexity results for unfinished work in $*/*/1$ queues. It seems reasonable to suspect that the same three lemmas can be re-formulated in terms of packet occupancy (rather than unfinished work)

when all packets have FIFO service. More precisely, suppose that $N_X(t)$ represents the number of packets in a FIFO queueing system with input stream $X(t)$. We can define the *Occupancy Blocking Function* $\alpha_{X_1,X_2}(t)$ in a manner similar to (A.2):

Definition 2. *The* occupancy blocking function $\alpha_{X_1,X_2}(t)$ *between two streams* X_1 *and* X_2 *is the function:*

$$\alpha_{X_1,X_2}(t) \triangleq N_{X_1+X_2}(t) - N_{X_1}(t) - N_{X_2}(t) \tag{A.7}$$

With this new definition of blocking in terms of packet occupancy, it can be shown that, for FIFO service, the non-negativity and symmetry properties still hold ($\alpha_{X_1,X_2}(t) \geq 0, \alpha_{X_1,X_2}(t) = \alpha_{X_2,X_1}(t)$). However, in Chapter Appendix A.B we furnish a counterexample demonstrating that, even under FIFO service, the occupancy monotonicity property does not hold for general variable length service time systems.

Such an example relies heavily on the fact that we have variable length packets. Indeed, in Chapter Appendix A.D it is shown that if all packets have fixed lengths L and service is non-preemptive work conserving, then the packet occupancy blocking function $\alpha_{X_1,X_2}(t)$ satisfies the non-negativity, symmetry, and monotonicity properties for all time.

A.3 Exchangeable Inputs and Convexity

In this section and the next, we use the non-negativity, symmetry, and monotonicity properties to show that any moment of unfinished work in a $*/*/1$ queue is a convex function of the input rate λ. To do this, we must first specify how an arbitrary input process can be parameterized by a single rate value. The parameterization should be such that an input stream of rate 2λ can be viewed as being composed of two similar streams of rate λ. Otherwise, it is clear that the convexity result may not hold. Indeed, consider an input stream $X_1(t)$ delivering bursty data at rate λ, and another stream $X_2(t)$ also delivering data at rate λ according to some other, less bursty process. If the $X_1(t)$ and $X_2(t)$ processes are sequentially added as inputs to a queue, the expected increment in unfinished work due to the additional $X_2(t)$ input may not be as large as the initial increment due to the $X_1(t)$ input. This happens if the $X_2(t)$ process is much smoother than $X_1(t)$, or if it is constructed to have packet arrivals precisely at idle periods of the queue with the $X_1(t)$ input alone.

Here, we consider the input rate λ as a discrete quantity which is varied by adding or removing substreams of the same "type" from the overall input process. We begin by

developing the notion of exchangeable random variables.

Definition 3. *A collection of M random variables are* exchangeable *if:*

$$p_{X_1,X_2,...,X_M}(x_1,\ldots,x_M) = p_{\tilde{X}_1,\tilde{X}_2,...,\tilde{X}_M}(x_1,\ldots,x_M) \tag{A.8}$$

for every $(\tilde{X}_1,\ldots,\tilde{X}_M)$ permutation of $(X_1,...,X_M)$, where $p_{X_1,X_2,...,X_M}(x_1,\ldots,x_M)$ is the joint density function.

Thus, exchangeable random variables exhibit a simple form of symmetry in their joint distribution functions.[4] Definitions for random variables to be *conditionally exchangeable* given some event ω can be similarly defined: The distributions in (A.8) are simply replaced by conditional distributions. It is clear that any set of independent and identically distributed (i.i.d.) random variables are exchangeable. Thus, exchangeable variables form a wider class than i.i.d. variables, and hence statements which apply to exchangeable variables are more general. Unlike i.i.d. variables, however, it can be seen that if random variables $(X_1,...,X_M)$ are conditionally exchangeable given some other random variable θ, then they are exchangeable.

We can extend this notion of exchangeability to include random *processes* that represent packet arrival streams. The following definition captures the idea that for any sample path realization of exchangeable processes $(X_1(t),...,X_M(t))$, the permuted sample path $(\tilde{X}_1(t),\ldots,\tilde{X}_M(t))$ is "equally likely":[5]

Definition 4. *Random processes $(X_1(t),...,X_M(t))$ are* exchangeable *if for any permutation $(\tilde{X}_1(t),...,\tilde{X}_M(t))$, we have $\mathbb{E}\{\Phi(X_1,\ldots,X_M)\} = \mathbb{E}\left\{\Phi(\tilde{X}_1,\ldots,\tilde{X}_M)\right\}$ for every operator $\Phi()$ which maps the processes to a single real number.*

Definition 5. *Random processes $(X_1(t),...,X_M(t))$ are* conditionally exchangeable given process $\theta(t)$ *if for every permutation $(\tilde{X}_1(t),...,\tilde{X}_M(t))$, we have $\mathbb{E}\{\Phi(X_1,\ldots,X_M,\theta)\} = \mathbb{E}\left\{\Phi(\tilde{X}_1,\ldots,\tilde{X}_M,\theta)\right\}$ for every real valued operator $\Phi()$ which acts on the processes.*

Hence, random processes are exchangeable if their joint statistics are invariant under every permutation. Note that the $\Phi()$ operator maps a set of *sample paths* to a real number.

[4] See [119] for an interesting treatment of the properties of exchangeable variables.

[5] The ideas developed here are closely related to the theory of *stochastic coupling*. See [119] for a formal treatment of the theory, and [133] for an application to optimal scheduling in symmetric queueing systems.

For example, it could correspond to the mapping of the input process $X(t)$ to its unfinished work at a particular time t^*. Exchangeable processes have the same properties as their random variable counterparts. In particular, if processes $(X_1, ..., X_M)$ are exchangeable given a process θ, then:

- Processes $(X_1, ..., X_M)$ are exchangeable.
- Processes $(X_k, ..., X_M)$ are exchangeable given processes $X_1, ..., X_{k-1}, \theta$.
- If $\Psi()$ is an operator that maps processes $X_1(t), X_2(t)$ and $\theta(t)$ to another process $Z(t) = \Psi(X_1, X_2, \theta)$, then $\Psi(X_1, X_2, \theta)$ and $\Psi(X_2, X_1, \theta)$ are exchangeable processes given $\theta(t)$.

The above properties are simple consequences of the definitions, where the last property follows by defining the operator $\tilde{\Phi}(X_1, X_2, \theta) \triangleq \Phi(\Psi(X_1, X_2, \theta), \Psi(X_2, X_1, \theta), \theta)$. Below we provide three examples of exchangeable input processes that can act as input streams to a queueing system:

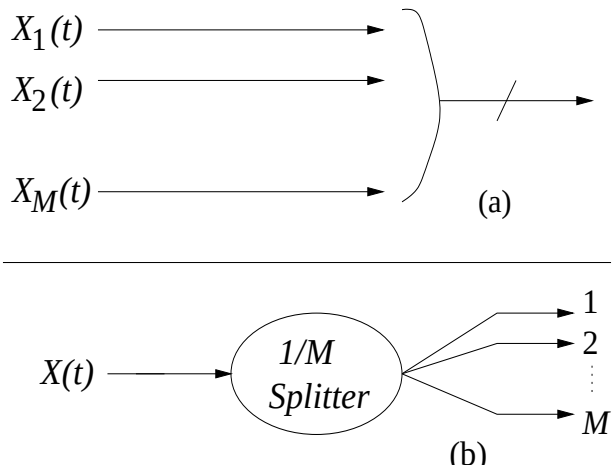

Figure A-6: M exchangeable inputs in the case of (a) the collection of i.i.d. $*/*$ processes $\{X_i\}$ and (b) probabilistic splitting of a $*/*$ process X into M substreams.

Example 1: Any general $*/*$ processes $\{X_i(t)\}$ independent and identically distributed over M input lines (Fig. A-6a).

Example 2: Any general $*/*$ process $X(t)$ which is split into M streams by routing each packet to stream $i \in \{1, \ldots, M\}$ with equal probability (Fig. A-6b).

Example 3: Any arbitrary collection of M processes $(X_1(t), ..., X_M(t))$ which are randomly permuted (with each permutation equally likely).

Notice that Example 1 demonstrates the fact that i.i.d. inputs are exchangeable. However, Example 2 illustrates that exchangeable inputs form a more general class of processes by providing an important set of input streams which are not independent yet are still exchangeable. Notice that this probabilistic routing can be extended to include "state-dependent" routing where the probability of routing to queue i depends on where the last packet was placed. The third example shows that an exchangeable input assumption is a good a-priori model to use when an engineer is given simply a "collection of wires" from various sources, and has no a-priori way of distinguishing the process running over "wire 1" from the process running over "wire 2."

We now examine how the unfinished work in a queue changes when a sequence of exchangeable inputs are added. Let $\theta(t)$ be an arbitrary background input process, and let $X_1(t)$ and $X_2(t)$ be two processes which are exchangeable given $\theta(t)$. Let $U_X(t)$ represent the unfinished work process as a function of time in a queue with an input process $X(t)$ running through it. Furthermore, let $f(u)$ represent any convex, non-decreasing function of the real number u for $u \geq 0$. We assume that the expected value of $f(U_X)$ is well defined. (Note that expectations over functions of the form $f(u) = u^k$ represent k^{th} moments of unfinished work). The following theorem shows that incremental values of queue cost are non-decreasing with each additional input.

Theorem 1. *For any particular time t^*, we have:*

$$\mathbb{E}f\left(U_{\theta+X_1+X_2}(t^*)\right) - \mathbb{E}f\left(U_{\theta+X_1}(t^*)\right) \geq \mathbb{E}f\left(U_{\theta+X_1}(t^*)\right) - \mathbb{E}f\left(U_{\theta}(t^*)\right) \tag{A.9}$$

Proof. Define the following processes:

$$\Delta_1(t) \triangleq U_{\theta+X_1}(t) - U_{\theta}(t) \tag{A.10}$$

$$\Delta_2(t) \triangleq U_{\theta+X_1+X_2}(t) - U_{\theta+X_1}(t) \tag{A.11}$$

We then find, by using the blocking function properties developed in the previous section:

$$\begin{aligned} \Delta_2(t) = U_{X_2}(t) + \beta_{\theta+X_1,X_2}(t) &\geq U_{X_2}(t) + \beta_{\theta,X_2}(t) \\ &= U_{\theta+X_2}(t) - U_{\theta}(t) \triangleq \tilde{\Delta}_1(t) \end{aligned} \tag{A.12}$$

where we have defined $\tilde{\Delta}_1(t) \triangleq U_{\theta+X_2}(t) - U_{\theta}(t)$. Because $X_2(t)$ and $X_1(t)$ are exchangeable

given $\theta(t)$, and because the $\tilde{\Delta}_1(t)$ and $\Delta_1(t)$ processes are derived from the same mapping of inputs to unfinished work, it follows that $\tilde{\Delta}_1(t)$ and $\Delta_1(t)$ are exchangeable processes given $\theta(t)$. Thus, for any time t^*, inequality (A.12) states that $\Delta_2(t^*)$ is a random variable that is always greater than or equal to another random variable which has the same distribution as $\Delta_1(t^*)$.

We now use an increasing increments property of non-decreasing, convex functions $f(u)$.

Fact: For non-negative real numbers a, b, x, where $a \geq b$, we have:

$$f(a+x) - f(a) \geq f(b+x) - f(b) \tag{A.13}$$

Using this fact and defining $a \triangleq U_{\theta+X_1}(t^*)$, $x \triangleq \Delta_2(t^*)$, and $b \triangleq U_\theta(t^*)$, we have:

$$\begin{aligned} f(U_{\theta+X_1}(t^*) + \Delta_2(t^*)) - f(U_{\theta+X_1}(t^*)) &\geq f(U_\theta(t^*) + \Delta_2(t^*)) - f(U_\theta) & \text{(A.14)} \\ &\geq f(U_\theta(t^*) + \tilde{\Delta}_1(t^*)) - f(U_\theta(t^*)) & \text{(A.15)} \end{aligned}$$

Inequality (A.14) follows from from (A.13) and the fact that $U_{\theta+X_1}(t^*) \geq U_\theta(t^*)$. Inequality (A.15) follows because $\Delta_2(t^*) \geq \tilde{\Delta}_1(t^*)$ (from (A.12)) and from the non-decreasing property of $f()$. Taking expectations of the inequality above, we find:

$$\mathbb{E}f\left(U_{\theta+X_1}(t^*) + \Delta_2(t^*)\right) - \mathbb{E}f\left(U_{\theta+X_1}(t^*)\right) \geq \mathbb{E}f\left(U_\theta(t^*) + \tilde{\Delta}_1(t^*)\right) - \mathbb{E}f\left(U_\theta(t^*)\right) \tag{A.16}$$

Using the fact that $\tilde{\Delta}_1(t)$ and $\Delta_1(t)$ are exchangeable given $\theta(t)$, we can replace the $\mathbb{E}f\left(U_\theta(t^*) + \tilde{\Delta}_1(t^*)\right)$ term on the right hand side of (A.16) with $\mathbb{E}f\left(U_\theta(t^*) + \Delta_1(t^*)\right)$, which yields the desired result. □

The theorem above can be used to immediately establish a convexity property of unfinished work in a work conserving queue with a collection of exchangeable inputs. Assume we have such a collection of M streams $(X_1, ..., X_M)$ which are exchangeable given another background stream $\theta(t)$. Assume that each of the streams X_i has rate λ_δ. The total input process to the queue can then be viewed as a function of a discrete set of rates $\lambda = n\lambda_\delta$ for $n \in \{0, 1, \ldots, M\}$. Let $\mathbb{E}f\left(U[n\lambda_\delta]\right)$ represent the expectation of a function $f()$ of the unfinished work (at some particular time t^*, which is suppressed for notational simplicity) when the input process consists of stream $\theta(t)$ along with a selection of n of the M exchangeable

streams. Hence:

$$\mathbb{E}f\left(U[n\lambda_\delta]\right) \triangleq \mathbb{E}f\left(U_{\theta+X_1+\ldots+X_n}(t^*)\right) \quad (0 \leq n \leq M) \tag{A.17}$$

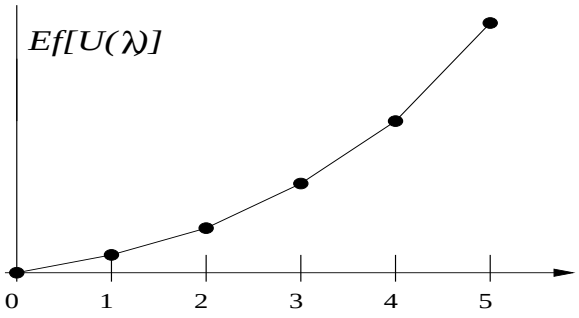

Figure A-7: Convexity of unfinished work as a function of the discrete rate parameter λ.

Theorem 2. *At any specific time t^*, the function $\mathbb{E}f\left(U[\lambda]\right)$ is monotonically increasing and convex in the discrete set of rates λ ($\lambda = n\lambda_\delta$, $n \in \{0, 1, \ldots, M\}$). In particular, any moment of unfinished work is convex (see Fig. A-7).*

Proof. Convexity of a function on a discrete set of equidistant points is equivalent to proving successive increments are monotonically increasing (Fig. A-7). Hence, the statement is equivalent to:

$$\mathbb{E}f\left(U[(n+2)\lambda_\delta]\right) - \mathbb{E}f\left(U[(n+1)\lambda_\delta]\right) \geq \mathbb{E}f\left(U[(n+1)\lambda_\delta]\right) - \mathbb{E}f\left(U[n\lambda_\delta]\right) \tag{A.18}$$

Defining the 'background stream' $\phi(t) = \theta(t) + X_1(t) + \ldots + X_n(t)$, we find that inequality (A.18) follows immediately from Theorem 19. □

A.3.1 Waiting Times

Notice that in Theorems 19 20, expectations were taken at any particular time t^*. It is not difficult to show that this property implies steady state unfinished work is convex, whenever such steady state limits exist. Moreover, we can allow t^* to be a time of special interest, such as the time when a packet from the X_1 stream enters the system. In FIFO queues, the unfinished work in the system at this special time represents the amount of waiting time W that the entering packet spends in the queue before receiving service. In this way, we show that waiting time increments are convex after the first stream is added. Specifically, for a

system with a background input $\theta(t)$ and M inputs $\{X_1, ..., X_M\}$ which are exchangeable given $\theta(t)$, we define the following steady state moments (which are functions of the discrete set of input rates $\lambda \in \{0, \lambda_\delta, 2\lambda_\delta, \ldots, M\lambda_\delta\}$):

$\mathbb{E}f\left(W_\theta^{(q)}[\lambda]\right), \mathbb{E}f\left(W_\theta[\lambda]\right)$ = Steady state waiting time moment corresponding to the time a packet from background stream $\theta(t)$ spends in the *queue* and in the *system*, respectively, when the controllable input rate is λ

$\mathbb{E}f\left(W_X^{(q)}[\lambda]\right), \mathbb{E}f\left(W_X[\lambda]\right)$ = Steady state waiting time moment corresponding to the time a packet from a controllable input stream spends in the *queue* and in the *system*, respectively, when the controllable input rate is λ

$\mathbb{E}f\left(N[\lambda]\right)$ = Steady state moment of the number of packets in the system (from both the background and controllable input streams) when the controllable input rate is λ

Formally, the steady state waiting time moments are defined:

$$\mathbb{E}f\left(W\right) \triangleq \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}f\left(W_k\right)$$

where W_k represents the waiting time of the k^{th} packet of the appropriate input stream. Likewise, the steady state occupancy moment is defined:

$$\mathbb{E}f\left(N\right) \triangleq \lim_{t\to\infty} \frac{1}{t} \int_0^t \mathbb{E}f\left(N(\tau)\right) d\tau$$

Assuming such steady state moments exist for the convex increasing function $f()$ of interest, we have:

Corollary 1. *In FIFO queueing systems:*

(a) $\mathbb{E}f\left(W_\theta^{(q)}[\lambda]\right)$ *and* $\mathbb{E}f\left(W_\theta[\lambda]\right)$ *are non-decreasing and convex in the discrete set of rates* $\lambda \geq 0$ *(i.e.,* $\lambda = n\lambda_\delta, n \in \{0, 1, \ldots, M\}$*).*

(b) $\mathbb{E}f\left(W_X^{(q)}[\lambda]\right)$ *and* $\mathbb{E}f\left(W_X[\lambda]\right)$ *are non-decreasing and convex in the discrete set of rates* $\lambda > 0$*.*

(c) $\mathbb{E}\left\{N[\lambda]\right\}$ *is non-decreasing and convex in the discrete set of rates* $\lambda \geq 0$*.*

Caveat: Note that in (b), waiting times for packets from the controllable input streams are not defined when $\lambda = 0$. Thus, convexity in this case is defined only for $\lambda > 0$. Further note that in (c), the function $f()$ is intentionally absent from the expectation, as we can only establish convexity of the first moment of packet occupancy in this general setting with variable length packets.

Proof. To prove (a), let p be a certain packet from the θ input stream which arrives to the system at time t_p. When n of the controllable inputs are applied to the system, $U_{\theta+X_1+...+X_n}(t_p^-)/\mu$ represents the amount of time packet p is buffered in the queue, and $U_{\theta+X_1+...+X_n}(t_p^+)/\mu$ is the total system time for packet p. From Corollary 20, $\mathbb{E}f\left(U[\lambda]\right)$ is a non-decreasing, convex function of $\lambda \geq 0$ when unfinished work is evaluated at any time t^*, including times $t^* = t_p^-$ and $t^* = t_p^+$. Hence, the expected waiting time of packet p in the queue and in the system is a non-decreasing convex function of the controllable input rate. Because this holds for any packet p from the θ stream, the expected waiting time averaged over all θ packets is also convex, completing the proof.

To prove (b), now let packet p represent a packet from the first controllable input stream X_1. Considering the sum process $\theta(t) + X_1(t)$ as a combined background stream with respect to inputs $\{X_2, ..., X_n\}$ (and noting that $\{X_2, ..., X_n\}$ remain exchangeable given $\theta(t) + X_1(t)$), from (a) we know that the expected queueing time and system time of packet p is a non-decreasing convex function of $\lambda \geq \lambda_\delta$. Because inputs $\{X_1, ..., X_M\}$ are exchangeable, the expected waiting time of a packet from stream X_1 is not different from the expected waiting time of a packet from stream X_k (provided that stream X_k is also applied to the system), and the result follows.

To prove (c), let $f(x) = x$. From (b) we know that $\mathbb{E}\left\{W_X[\lambda]\right\}$ is non-decreasing and convex for $\lambda > 0$. It is straightforward to verify that for any such function, the function $\lambda\mathbb{E}\left\{W_X[\lambda]\right\}$ is non-decreasing and convex for $\lambda \geq 0$, where $\lambda\mathbb{E}\left\{W_X[\lambda]\right\}$ is defined to be 0 at $\lambda = 0$. Let λ_θ represent the rate of the $\theta(t)$ stream. By Little's Theorem, it follows that $\mathbb{E}\left\{N[\lambda]\right\} = \lambda\mathbb{E}\left\{W_X(\lambda)\right\} + \lambda_\theta\mathbb{E}\left\{W_\theta[\lambda]\right\}$ is non-decreasing and convex in λ, as this is the sum of non-decreasing convex functions. □

One might expect the waiting time $\overline{W}_{av}$ averaged over packets from both the controllable and uncontrollable input streams to be convex. However, note that $\overline{W}_{av}[\lambda] = \left(\frac{\lambda_\theta}{\lambda+\lambda_\theta}\right)\mathbb{E}\left\{W_\theta[\lambda]\right\} + \left(\frac{\lambda}{\lambda+\lambda_\theta}\right)\mathbb{E}\left\{W_X[\lambda]\right\}$ is not necessarily convex even though both $\mathbb{E}\left\{W_\theta[\lambda]\right\}$

and $\mathbb{E}\{W_X[\lambda]\}$ are. Indeed, the following simple example shows that $\overline{W}_{av}[\lambda]$ may even *decrease* as λ increases:

Example: Let background input $\theta(t)$ periodically produce a new packet of service time 10 at times $t = \{0, 100, 200, \ldots\}$. Let input $X_1(t)$ consist of packets of service time 2 occurring periodically at times $t = \{50, 150, 250, \ldots\}$. Hence, packets from $\theta(t)$ and $X_1(t)$ never interfere with each other. We thus have $\overline{W}_{av}(0) = 10$ and $\overline{W}_{av}[\lambda_\delta] = (10 + 2)/2 = 6$.

A.3.2 Packet Occupancy $N(t)$

Notice that the non-negativity, symmetry, and monotonicity properties of the blocking function $\beta_{X_1,X_2}(t)$ were the only queueing features needed to establish convexity of unfinished work $U(t)$. Now suppose that all packets have fixed lengths L, and let $N(t)$ represent the number of packets in the queueing system at time t for some arbitrary arrival process. If service in the queue is work conserving and non-preemptive, it can be shown that the occupancy blocking function $\alpha_{X_1,X_2}(t)$ satisfies the non-negativity, symmetry, and monotonicity properties (see Chapter Appendix A.D for a complete proof). We can thus reformulate Theorems 19 and 20 in terms of packet occupancy. Suppose again that input streams $\{X_1, ..., X_M\}$ are exchangeable given background stream θ. We find:

Corollary 2. *If all packets have fixed lengths L and service is non-preemptive work conserving, then at any particular time t^*, the expectation $\mathbb{E}f(N[\lambda])$ is a convex function of the discrete rates $\lambda \in \{0, \lambda_\delta, 2\lambda_\delta, \ldots, M\lambda_\delta\}$.* $\square$

A.4 Convexity over a Continuous Rate Parameter

In the previous section we dealt with streams of inputs and demonstrated convexity of unfinished work and waiting time moments as streams are removed or added. Here, we extend the theory to include input processes that are parameterized by a continuous rate variable λ. The example to keep in mind in this section is packet-by-packet probabilistic splitting, where individual packets from an arbitrary packet stream are sent to the queue with some probability p. However, the results apply to any general "infinitely splittable" input, which are inputs that can be split into *substreams* according to some splitting method, as described below:

Definition 6. *A packet input process $X(t)$ together with a splitting method is said to be* infinitely splittable *if:*

(1) $X(t)$ or any of its substreams *can be split into disjoint substreams of arbitrarily small rate. Any superposition of disjoint substreams of $X(t)$ is considered to be a substream.*

(2) Any two (potentially non-disjoint) substreams that have the same rate are conditionally exchangeable given the rest of the process.

We emphasize that the above definition incorporates both the input process $X(t)$ and the method of splitting. Notice that any $*/*$ process $X(t)$ is infinitely splittable when using the probabilistic splitting method of independently including packets in a new substream i with some probability p_i. Likewise, probabilistic splitting of the lead packet in systems where blocks of K sequential packets must be routed together can be shown to satisfy the conditions of infinite splittability. However, not all splitting methods are valid. Consider for example the "divide by 2" splitting method, where an input stream is split into two by alternately routing packets to the first stream and then the second. Suppose the base input stream $X(t)$ has rate λ and consists of fixed length packets of unit size. Under this splitting method, any substream of rate $\lambda\frac{k}{2^n}$ can be formed by collecting superpositions of disjoint substreams of rate $\lambda/2^n$ (where k and n are any integers such that $k \leq 2^n$).[6] Thus, the first condition of infinite splittability is satisfied. However, it is not clear how a substream $\tilde{X}(t)$ of rate $\lambda/2$ is distributed. For example, the original stream $X(t)$ could be split into two substreams, one of which is $\tilde{X}(t)$ and represents the sequence of every other packet arrival from $X(t)$. (The odd substream may be differently distributed from the even substream, but this can be taken care of by randomly permuting the two so that it is impossible to know if $\tilde{X}(t)$ contains the odd samples or the even samples.) Alternately, the "divide by 2" splitting method might be used to form $\tilde{X}(t)$ by iteratively splitting $X(t)$ into eight substreams of rate $\lambda/8$, a random four of which are grouped together to form the rate $\lambda/2$ substream. Clearly the two approaches to building a rate $\lambda/2$ substream do not generally lead to identically distributed processes, as the first approach leads to a rate $\lambda/2$ substream that never contains two successive packets from the original stream, while the second approach leads to a $\lambda/2$ substream that might contain two successive packets. Thus, the divide-by-2 splitting method satisfies the first condition of the above definition

[6]Formally, streams of arbitrary irrational rates $\tilde{\lambda} < \lambda$ can be formed from the "divide by 2" splitting method by a countably infinite superposition of substreams, where the substreams that are added have progressively smaller rates.

but not the second.

With the above definition, it can be seen that an infinitely splittable input process $X(t)$ can be written as the sum of a large number of exchangeable substreams. Specifically, it has the property that for any $\epsilon > 0$, there exists a large integer M such that:

$$X(t) = \sum_{i=1}^{M} x_i(t) + \tilde{x}(t)$$

where $(x_1(t), ..., x_M(t))$ are exchangeable substreams, each with rate λ_δ, $\tilde{x}(t)$ has rate $\tilde{\lambda}_\delta$, and $\tilde{\lambda}_\delta < \lambda_\delta < \epsilon$.

We now use the blocking function to establish continuity of expected moments of unfinished work as a function of the continuous rate parameter λ. As before, these results also apply to waiting times in FIFO systems.

Again we assume that $f(u)$ is a non-decreasing convex function over $u \geq 0$. Suppose $X(t)$ is an infinitely splittable input process with total rate λ_{tot}. Suppose also that all exchangeable component processes of $X(t)$ are also exchangeable given the background input process $\theta(t)$. Let $\mathbb{E}f\left(U[\lambda_{tot}]\right)$ represent the expectation of a function of unfinished work at a particular time t^* in a queue with this input and background process. We assume here that $\mathbb{E}f\left(U[\lambda_{tot}]\right)$ is finite.

Theorem 3. *$\mathbb{E}f\left(U[\lambda]\right)$ can be written as a pure function of the continuous rate parameter λ, where $\lambda \in [0, \lambda_{tot}]$ is a rate achieved by some substream of the infinitely splittable $X(t)$ input. Furthermore, $\mathbb{E}f\left(U[\lambda]\right)$ is a monotonically increasing and continuous function of λ.*

Proof. The proof is given in Chapter Appendix A.A. We note that the proof of continuity uses a simple ϵ, δ argument (in the manner of the classic proof that the function $f(x) = x^2$ is continuous) together with the machinery of the blocking function. □

The continuity property of Theorem 21 allows us to easily establish the convexity of any moment of unfinished work (and packet waiting time) in a $*/*/1$ queue as a function of the continuous input rate λ. Let $X(t)$ be an infinitely splittable input process, and suppose that every collection of exchangeable components of $X(t)$ are also exchangeable given the background process $\theta(t)$. Then:

Theorem 4. *At any particular time t^*, the function $\mathbb{E}f\left(U[\lambda]\right)$ is convex over the continuous variable $\lambda \in [0, \lambda_{tot}]$. Likewise, if service is FIFO, then $\mathbb{E}f\left(W[\lambda]\right)$ is also convex.*

Proof. We wish to show that the function $\mathbb{E}f\left(U[\lambda]\right)$ always lies below its chords. Thus, for any three rates $\lambda_1 < \lambda_2 < \lambda_3$, we must verify that:

$$\mathbb{E}f\left(U[\lambda_2]\right) \leq \mathbb{E}f\left(U[\lambda_1]\right) + (\lambda_2 - \lambda_1)\frac{\left(\mathbb{E}f\left(U[\lambda_3]\right) - \mathbb{E}f\left(U[\lambda_1]\right)\right)}{(\lambda_3 - \lambda_1)} \tag{A.19}$$

We know from Theorem 20 in Section A.3 that the unfinished work function is convex over a discrete set of rates when the input process is characterized by a finite set of M exchangeable streams $(x_1, ..., x_M)$. We therefore consider a discretization of the rate axis by considering the sub-processes $(x_1, ..., x_M)$ of the infinitely splittable process $X(t)$, where each x_i has a small rate δ. In this discretization, we have rates:

$$\tilde{\lambda}_1 = k_1\delta, \tilde{\lambda}_2 = k_2\delta, \tilde{\lambda}_3 = k_3\delta \tag{A.20}$$

where the rates $(\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3)$ can be made arbitrarily close to their counterparts $(\lambda_1, \lambda_2, \lambda_3)$ by choosing an appropriately small value of δ. Now, from the discrete convexity result, we know:

$$\mathbb{E}f\left(U[\tilde{\lambda}_2]\right) \leq \mathbb{E}f\left(U[\tilde{\lambda}_1]\right) + (\tilde{\lambda}_2 - \tilde{\lambda}_1)\frac{\mathbb{E}f\left(U[\tilde{\lambda}_1]\right) - \mathbb{E}f\left(U[\tilde{\lambda}_1]\right)}{(\tilde{\lambda}_3 - \tilde{\lambda}_1)} \tag{A.21}$$

By continuity of the $\mathbb{E}f\left(U[\lambda]\right)$ function, we can choose the discretization unit δ to be small enough so that the right hand side of (A.21) is arbitrarily close to the right hand side of the (currently unproven) inequality (A.19). Simultaneously, we can ensure that the left hand sides of the two inequalities are arbitrarily close. Thus, the known inequality (A.21) for the discretized inputs implies inequality (A.19) for the infinitely splittable input. We thus have convexity of unfinished work at any point in time, which also implies convexity of waiting time in FIFO systems. □

A.5 Multiple Queues in Parallel

We now consider the system of N queues in parallel as shown in Fig. A-8. The servers of each queue k have linespeeds μ_k and arbitrary background packet input processes $\theta_k(t)$. An arbitrary input process $X(t)$ also enters the system, and $X(t)$ is rate-controllable in that a router can split $X(t)$ into substreams of smaller rate. These substreams can be distributed according to an N-tuple rate vector $(\lambda_1, ..., \lambda_N)$ over the multiple queues.

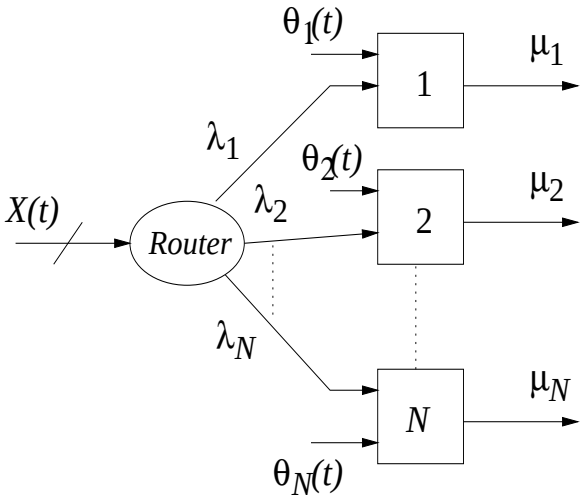

Figure A-8: Multiple queues in parallel with different background processes $\theta_i(t)$ and server rates μ_i.

We consider both the case when $X(t)$ is an infinitely splittable process (as in packet-based probabilistic splitting), and the case when $X(t)$ is composed of a finite collection of M exchangeable streams. The problem in both cases is to route the substreams by forming an optimal rate vector that minimizes some network cost function. We assume the cost function is a weighted summation of unfinished work and/or waiting time moments in the queues. Specifically, we let $\{f_k(u)\}$ be a collection of convex, non-decreasing functions on $u \geq 0$. Suppose that the queues reach some steady state behavior, and let $\mathbb{E}f\left(U_k[\lambda_k]\right)$ represent the steady state moment of unfinished work in queue k when an input stream of rate λ_k is applied. Let $\mathbb{E}f\left(W_k[\lambda_k]\right)$ represent the steady moment of waiting time for queue k.

Theorem 5. *If queues are work conserving and $X(t)$ is either a finitely*[7] *or infinitely rate splittable process given $\{\theta_k(t)\}$, then:*

(a) Cost functions of the form

$$C(\lambda_1, \ldots, \lambda_N) = \sum_{k=1}^{N} \mathbb{E}f_k(U_k[\lambda_k]) \tag{A.22}$$

are convex in the multivariable rate vector $(\lambda_1, \ldots, \lambda_N)$.

[7]We note that convexity on a discrete set of points is equivalent to convexity of the piecewise linear interpolation, see Fig. A-7.

(b) If service is FIFO, then cost functions of the form

$$C(\lambda_1, \ldots, \lambda_N) = \sum_{k=1}^{N} \lambda_k \mathbb{E} f_k(W_k[\lambda_k]) \tag{A.23}$$

are convex (where the $W_k[\lambda_k]$ values represent waiting times of packets from the controllable inputs).

(c) If service is FIFO and $N_k(\lambda_k)$ represents the number of packets in queue k in steady state, then cost functions of the following form are convex:

$$C(\lambda_1, \ldots, \lambda_N) = \sum_{k=1}^{N} \lambda_k c_k \mathbb{E}\left\{N_k[\lambda_k]\right\} \quad (\{c_k\} \geq 0) \tag{A.24}$$

Proof. Since $\mathbb{E} f_k(W_k[\lambda_k])$ is convex and non-decreasing for $\lambda_k > 0$, the function $\lambda_k \mathbb{E} f_k(W_k[\lambda_k])$ is convex on $\lambda_k \geq 0$. Thus, the cost functions in (a) and (b) are summations of convex functions, so they are convex. Part (c) follows from (b) by noting that, from Little's Theorem, $\mathbb{E}\{N\} = \lambda \mathbb{E}\{W\}$. □

Convexity of the cost function $C(\lambda_1, \ldots, \lambda_N)$ can be used to develop optimal rate distributions $(\lambda_1^o, \ldots, \lambda_N^o)$ over the simplex constraint $\lambda_1 + \ldots + \lambda_N = \lambda_{tot}$. For symmetric cost functions, which arise when the background processes $\{\theta_k(t)\}$ and the linespeeds $\{\mu_k\}$ are the same for all queues $k \in \{1, \ldots, N\}$, the optimal solution is particularly simple. It is clear in this case that the uniform distribution (or as near to this as can be achieved) is optimal and minimizes cost. Thus, in the symmetric case we want to spread the total input stream evenly amongst all of the queues in order to take full advantage of the bandwidth that each queue provides. In the asymmetric case when background streams and linespeed processes are not the same, the optimal rate vector deviates from the uniform allocation to favor queues with faster linespeeds and/or less background traffic.

Let us suppose the cost function $C(\lambda_1, \ldots, \lambda_N)$ is known. Convexity of $C()$ tells us that any local minimum of the cost function we find must also be a global minimum. It also tells us a great deal more [15], as we illustrate for both finitely and infinitely splittable inputs $X(t)$ below.

A.5.1 (Stream-Based) Finitely Splittable $X(t)$

Here we assume that the input $X(t)$ is a finite collection of M streams, which are exchangeable given the background processes $\{\theta_k(t)\}$. We want to distribute the streams over the N queues available. We can thus write the cost function $C(M_1, \ldots, M_N)$ as a function of an integer N-tuple $(M_1, \ldots, M_N)$ (rather than a rate N-tuple) which describes the number of streams we route to each queue.

If the weight functions $f_k()$ are identical for all k, and all queues have identical linespeeds and exchangeable background inputs, then the cost function $C(M_1, \ldots, M_N)$ is convex symmetric and the optimal solution is to allocate according to the load balanced strategy of assigning $\lceil M/n \rceil$ streams to $(M) \mod (N)$ of the queues, and $\lfloor M/N \rfloor$ streams to the remaining queues. In the non-symmetric case, we must consider other allocations and test them by evaluating the cost function.

Lemma 2. *Given a convex cost function $C(M_1, \ldots, M_N)$ of the form specified in Theorem 23, the optimal allocation vector can be obtained by sequentially adding streams, greedily choosing at each iteration the queue which increases the total cost $C()$ the least. This yields a cost-minimizing vector $(M_1^o, \ldots, M_N^o)$ after $M+N-1$ evaluations/estimations of the cost function.*

Proof. This lemma follows as a special case of a theory of integer optimization over separable convex functions (see [46]). We provide a simplified and independent proof in Chapter Appendix A.E for completeness. □

In the special case of routing M exchangeable inputs over N queues with the same server rate μ, it is clear that the load-balanced strategy minimizes all symmetric cost functions, and in particular it minimizes the expected unfinished work in all of the queues at every instant of time, which extends the result of [56] [25] [66] to $*/*$ inputs and stream based routing. It is tempting to conjecture that load balancing also stochastically minimizes the total workload at every instant of time, as is proved in [80] for homogeneous queues, Bernoulli splitting, and $*/M$ inputs. However, this is not the case, as illustrated by a simple counterexample given in Chapter Appendix A.C.

Example: We consider the system of Fig. A-8 when there are $N = 4$ queues and the input $X(t)$ consists of 200 independent streams that produce packets periodically every P seconds. Packets have a fixed length of L bits. The streams are unsynchronized, and hence

200 packets arrive in a uniformly distributed fashion over any time interval of length P. Such input streams are models for continuous bit rate traffic, such as digital voice or video data.

The problem is to distribute the streams over the 4 queues while minimizing the cost function, which we take to be the total expected number of packets in the system (this is the cost function of Theorem 23c). We first assume all server rates μ_i are the same, and all background streams $\theta_i(t)$ are exchangeable. In this case, we immediately know the optimal stream allocation vector is $(50, 50, 50, 50)$. Likewise for this symmetric example, if there are 201 streams, then the optimal vector is $(51, 50, 50, 50)$ (or any of the three other near-symmetric allocations).

Now suppose that we have server rates $(2, 1, 1, 1)$ and that there are 10 background streams of the same type at queue 4. In this asymmetric case, we must use the cost function to determine optimal allocation using the sequentially greedy method. The complementary occupancy distribution in a single queue with M inputs of period P, length L, and server rate μ has been derived explicitly in [118]. The result is:

$$Q_n\left(\frac{L}{\mu P}, M\right) = \sum_{i=1}^{M-n} \begin{pmatrix} M \\ i+n \end{pmatrix} \left(\frac{iL}{\mu P}\right)^{i+n} \left(1 - \frac{iL}{\mu P}\right) \left(\frac{\mu P/L - M + n}{\mu P/L - i}\right) \quad (0 \le n \le M-1)$$

The expected number of packets is hence:

$$\overline{N}\left(\frac{L}{\mu P}, M\right) = \sum_{n=0}^{M-1} Q_n\left(\frac{L}{\mu P}, M\right)$$

We therefore use the greedy algorithm with cost function:

$$C(M_1, M_2, M_3, M_4) = \sum_{i=1}^{4} \overline{N}\left(\frac{L}{\mu_i P}, M_i\right)$$

Using $L = 1, P = 75$, we find that the optimal allocation vector is: $(100, 37, 37, 26)$. From this solution, we see that—as expected because of the 10 background streams in queue 4—there are approximately 10 more streams allocated to queues 2 and 3 than queue 4. Interestingly, the server rate of queue 1 is twice that of the other queues, although, due to statistical multiplexing gains, the number of streams it is allocated is more than twice the number allocated to the others.

A.5.2 (Packet Based) Infinitely Splittable $X(t)$

Here we consider an infinitely splittable process $X(t)$ with total rate λ_{tot} as an input to the system of Fig. A-8. The problem is to optimally distribute the total rate over the N queues so as to minimize a cost function $C(\lambda_1, \ldots, \lambda_N)$. We assume the cost function is of one of the forms specified in Theorem 23. Each of these had the structure:

$$C(\lambda_1, \ldots, \lambda_N) = g_1(\lambda_1) + \ldots + g_N(\lambda_N)$$

for some convex, non-decreasing functions $g_i(\lambda_i)$.

If background inputs are exchangeable, and if all queues are weighted the same in the cost function, then $C()$ is convex symmetric and the optimal rate allocation is $(\lambda_{tot}/N, ..., \lambda_{tot}/N)$. Otherwise, we can take advantage of the convex structure of the $C()$ function to determine the optimal solution [13].

Each of the functions $g_i(\lambda_i)$ is non-decreasing and convex on some open interval $(0, \tilde{\lambda}_i)$. It can be shown that these properties ensure $g_i(\lambda_i)$ is right-differentiable. They are also sufficient to establish the correctness of a Lagrange Multipliers approach to cost minimization. Given the Lagrangian:

$$L(\lambda_1, \ldots, \lambda_N, \gamma) = C(\lambda_1, \ldots, \lambda_N) + \gamma \left(\lambda_{tot} - \sum_{i=1}^{N} \lambda_i \right)$$

where γ is the Lagrange Multiplier, we differentiate (from the right) with respect to λ_i to obtain

$$\frac{d}{d\lambda_i} g_i(\lambda_i) = \gamma \text{ for all } i \in \{1, \ldots, N\}$$

subject to the simplex constraint $\lambda_1 + \ldots + \lambda_N = \lambda_{tot}$. Fig. A-9 illustrates this solution. If we define:

$$\lambda_i(\gamma) = \text{Largest } \lambda \text{ such that } g_i'(\lambda) \leq \gamma$$

then from Fig. A-9 we see that we increase the value of γ until $\lambda_1(\gamma) + \ldots + \lambda_N(\gamma) = \lambda_{tot}$. The resulting rate vector yields the optimal solution. Although the form of the solution appears different from that of the discrete rate scenario, in fact the two can be viewed as exactly the same. Indeed, the continuous rate solution corresponds to sequentially allocating an infinitesimal portion of the total rate in a greedy manner.

Notice that a fast bisection type algorithm can be developed to find this optimal rate vector. First, two bracketing values γ_{Low} and γ_{Hi} are found which yield $\lambda_1(\gamma)+\ldots+\lambda_N(\gamma)$ values above and below λ_{tot}, respectively. The bisection routine proceeds as usual until the rate vector converges to a solution within acceptable error limits.

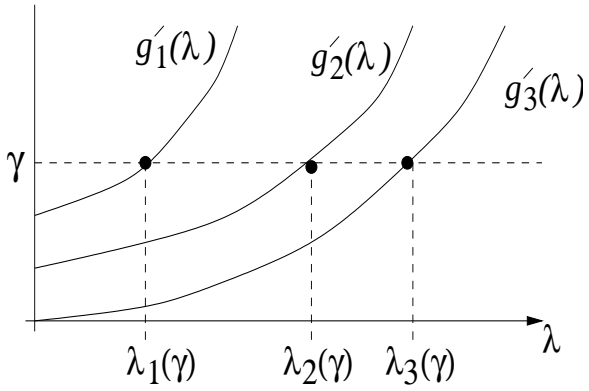

Figure A-9: A simple set of $\frac{d}{d\lambda_i}g_i(\lambda_i)$ curves.

Example: The following simple example for systems with memoryless arrivals and packet lengths demonstrates the method. The example fits into the optimal routing framework of [14] [22] [19], and the solution we obtain can be verified using the techniques developed there. Suppose, for simplicity, that there are no background arrivals $\{\theta_i(t)\}$. Let the arrival process be Poisson with rate λ, and the packet length process be i.i.d. and memoryless with an average packet length of 1 unit. We assume that we are using probabilistic Bernoulli routing where each packet is routed independently of the next. Let $\overline{N}_i$ represent the average number of packets in queue i, and define cost function:

$$C(\lambda_1,\ldots,\lambda_N)=\sum_i c_i\overline{N}_i[\lambda_i]$$

where

$$g_i(\lambda_i)=c_i\overline{N}_i(\lambda_i)$$

The expected number of packets in an M/M/1 queue is:

$$\overline{N}_i=\frac{\lambda_i/\mu_i}{1-\lambda_i/\mu_i}$$

Multiplying the above equation by c_i, taking derivatives with respect to λ_i, and setting the result equal to γ for all i, as well as considering the constraint $\lambda_1+\ldots+\lambda_N=\lambda_{tot}$, we find

for all $k \in \{1, \ldots, N\}$:

$$\lambda_k = \mu_k + \frac{\sqrt{c_k \mu_k}\,(\lambda_{tot} - \sum_i \mu_i)}{\sum_i \sqrt{c_i \mu_i}}$$

The above equation is a bit deceptive, in that the summations are taken over all i for which λ_i is positive. The positive λ_i's are determined by first assuming that all are positive and applying the above equation. If any of the λ_i's found are zero or negative, these λ_i's are set to zero and the calculation is repeated using the remaining subset of λ_i's.

This approach to optimal rate allocation is similar to the convex optimization routines described in [14]. There, the authors address packet routing in general mesh networks when input streams can be continuously split according to a rate parameter. They presuppose some convex cost function at each node of the network, which (as an idealization) is completely a function of the overall rate routed to that node. Here, we have considered cost functions that reflect the actual queuing congestion at each node when a general $*/*$ input is applied. We have established that, for a simple network consisting of N parallel queues, the cost functions at each queue are continuous and non-decreasing in the rate parameter, and they are indeed convex.

A.6 Time-Varying Server Rates

Here we consider the system of Fig. A-1 when the constant server of rate μ is replaced by a time varying server of rate $\mu(t)$. Characteristics of an unfinished work sample path $U_X(t)$ for time varying servers are similar to those illustrated in Fig. A-2 for constant server systems, with the exception that the $U_X(t)$ function decreases with a time varying slope $-\mu(t)$. This presents a slight problem for waiting time analysis, as waiting times may no longer be convex. Recall that for constant processing rate servers, the unfinished work $U_X(t)$ is proportional to the amount of time it would take the system to empty if no other packets were to arrive (specifically, this time is $U_X(t)/\mu$). However, this emptying time is not causally known from the system state for time varying servers. Hence, the unfinished work in the system at the time of a packet arrival no longer indicates the amount of time this packet will wait.

On the other hand, convexity analysis of the unfinished work $U_X(t)$ and the number of packets $N_X(t)$ can be performed for this time varying context in a manner similar to the treatment for fixed processing rate systems. Indeed, we can define the unfinished work

blocking function $\beta_{X_1,X_2}(t)$ and the packet occupancy blocking function $\alpha_{X_1,X_2}(t)$ as before:

$$\begin{aligned}\beta_{X_1,X_2}(t) &\triangleq U_{X_1+X_2}(t) - U_{X_1}(t) - U_{X_2}(t)\\ \alpha_{X_1,X_2}(t) &\triangleq N_{X_1+X_2}(t) - N_{X_1}(t) - N_{X_2}(t)\end{aligned}$$

By literally repeating the arguments of Section A.2, we can establish that the non-negativity, symmetry, and monotonicity properties still hold for $\beta_{X_1,X_2}(t)$ in this time-varying context. Likewise, if all packets have fixed lengths L and service is non-preemptive, it can be shown for time varying servers that the occupancy blocking function $\alpha_{X_1,X_2}(t)$ also satisfies these three properties (see Chapter Appendix A.D).

Consequently, given a collection of N queues with background input processes $\{\theta_i(t)\}$ and server rate processes $\{\mu_i(t)\}$, together with a (finitely or infinitely distributable) input $X(t)$, we can establish:

Theorem 6. *If the exchangeable components of $X(t)$ are exchangeable given $\{\theta_i(t)\}$ and $\{\mu_i(t)\}$, then $\sum \mathbb{E} f_i(U_i[\lambda_i])$ is convex in the rate vector $(\lambda_1, \ldots, \lambda_N)$. If all packets have a fixed length of L bits and service is non-preemptive, then $\sum \mathbb{E} f_i(N_i[\lambda_i])$ is convex in the rate vector.*

Recall from Little's Theorem that if the expected waiting time $\mathbb{E}\{W(\lambda)\}$ is convex in λ, then so is the expected packet occupancy $\mathbb{E}\{N(\lambda)\}$. However, the converse implication does not follow. Indeed, below we provide a (counter) example which illustrates that—even for fixed length packets under FIFO service—waiting times are not necessarily convex for time varying servers.

(Counter) Example: Consider identical input processes X_1, X_2, X_3 which produce a single packet of length $L = 1$ periodically at times $\{0, 3, 6, 9, ...\}$. Let the server rate be periodic of period 3 with $\mu(t) = 1$ for and $\mu(t) = 100$ for $t \in (2, 3)$. Then $\mathbb{E}\{W_{X_1}\} = 1$, $\mathbb{E}\{W_{X_1+X_2}\} = 1.5$, and $\mathbb{E}\{W_{X_1+X_2+X_3}\} = 1.67$. Clearly the increment in average waiting time when stream X_2 is added is larger than the successive increment when stream X_3 is added. Hence, waiting time is not convex in this time-varying server setting. □

Although waiting times are not necessarily convex, notice that minimizing $\overline{W}_{tot}$ in a parallel queue configuration (Fig. A-8) is accomplished by minimizing $\overline{N}_{tot}$ (since $\overline{N}_{tot} = \lambda_{tot}\overline{W}_{tot}$). For fixed length packets, Theorem 24 ensures this is a convex optimization even for time varying servers. Indeed, notice that expected occupancies $\mathbb{E}\{N_{X_1}\}$, $\mathbb{E}\{N_{X_1+X_2}\}$,

and $\mathbb{E}\{N_{X_1+X_2+X_3}\}$ for the above example can be obtained by multiplying $\mathbb{E}\{W_{X_1}\}$, $\mathbb{E}\{W_{X_1+X_2}\}$, and $\mathbb{E}\{W_{X_1+X_2+X_3}\}$ by $\lambda = 1/3$, $2/3$, and $3/3$ respectively, and the resulting values are convex. Indeed, the non-convex values $1, 1.5, 1.67$ become $\frac{1}{3}$, 1, and 1.67, which are just on the borderline of convexity.

A.7 Summary

We have developed general convexity results for $*/*/1$ queues using a new function of two packet stream inputs called the blocking function. Non-negativity, Symmetry, and Monotonicity properties of the blocking function were established. These properties proved to be valuable tools for establishing convexity of unfinished work and waiting time moments in terms of both a discrete and a continuous input rate λ.

We then addressed both stream-based and packet-based routing of general inputs over a collection of N parallel queues with arbitrary background inputs and different linespeeds. Optimal routing algorithms were developed utilizing the convexity results.

This convexity theory can be extended to address more complex variants of the parallel queue problem. One might consider a case when we have a collection of K sets of exchangeable inputs, where each set categorizes a different type of input process. For example, set S_1 could contain multiple exchangeable inputs of the "bursty" type, while set S_2 contains exchangeable inputs of the "continuous bit rate" type. This variation of the problem is closely related to the NP-complete bin packing problem. It would also be interesting to explore convexity and optimal routing in more general mesh networks using these techniques. Such an approach could perhaps establish the validity of known convex optimization routines, as well as provide insights into developing new ones.

Appendix A.A — Continuity of Unfinished Work

Here we show that for any particular time t^*, $\mathbb{E}f\left(U[\lambda]\right)$ is a continuous, monotonically increasing function of λ (Theorem 21 of Section A.4). We utilize the following facts about convex, non-decreasing functions:

Fact 1: If $f(u)$ is non-decreasing and convex, then for any fixed $a \geq 0$ there is a function $g(a,x)$ such that $f(a+x) = f(a)+g(a,x)$, where $g(a,x)$ is a convex, non-decreasing function of x for $x \geq 0$.

Fact 2: Any convex, non-decreasing function $g(x)$ with $g(0) = 0$ has the property that $g(x_1 + x_2) \geq g(x_1) + g(x_2)$ for any $x_1, x_2 \geq 0$.

Note that $\mathbb{E}f\left(U[\lambda]\right) \triangleq \mathbb{E}f\left(U_{X_\lambda}(t^*)\right)$, where $X_\lambda(t)$ is any substream of $X(t)$ with rate λ. This is a well defined function of λ because, by the properties of infinitely splittable inputs, all substreams with the same rate are identically distributed. The fact that $\mathbb{E}f\left(U[\lambda]\right)$ is monotonically increasing in λ follows as a simple consequence of the non-negativity property of the blocking function. Indeed, consider a substream $X_\delta(t)$ of rate δ. We have:

$$\mathbb{E}f\left(U[\lambda+\delta]\right) \triangleq \mathbb{E}f\left(U_{X_\lambda + X_\delta}(t^*)\right) \geq \mathbb{E}f\left(U_{X_\lambda}(t^*)\right) \triangleq \mathbb{E}f\left(U[\lambda]\right)$$

proving monotonicity. Below we prove the continuity property.

Proof. (Continuity) Here we prove that the function $\mathbb{E}f\left(U[\lambda]\right)$ is continuous from the right with respect to the λ parameter. Left continuity can be proven in a similar manner.

Take any λ in the set of achievable rates. We show that:

$$\lim_{\delta \to 0} \mathbb{E}f\left(U[\lambda+\delta]\right) = \mathbb{E}f\left(U[\lambda]\right) \tag{A.25}$$

where δ is the rate of a component process of $X(t)$ which make arbitrarily small. By monotonicity, if δ decreases to zero, then $\mathbb{E}f\left(U[\lambda+\delta]\right) - \mathbb{E}f\left(U[\lambda]\right)$ decreases toward some limit ϵ, where $\epsilon \geq 0$. Suppose now that this inequality is strict, so that $\epsilon > 0$. We reach a contradiction.

Consider disjoint component streams $\{x_1, \ldots, x_M\}$, each x_i of rate δ, for some yet-to-be-determined δ and M. We assume that these M sub-streams are disjoint from another substream θ of rate λ, all of which are components of the entire process $X(t)$. Let $U_{\theta+x_1+\ldots+x_M}$ represent the unfinished work in the system at some particular time t^*, with input processes

$\{\theta, x_1, \ldots, x_M\}$. From the definition of the blocking function, we have:

$$U_{\theta+x_1+\ldots+x_M} = U_{\theta+x_1+\ldots,+x_{M-1}} + U_{x_M} + \beta_{\theta+x_1+\ldots+x_{M-1},x_M} \tag{A.26}$$

By recursively iterating (A.26), we find:

$$U_{\theta+x_1+\ldots+x_M} = U_\theta + \sum_{i=1}^{M} U_{x_i} + \sum_{k=0}^{M-1} \beta_{\theta+x_1+\ldots+x_k,x_{k+1}} \tag{A.27}$$

Applying the monotonicity property of the blocking function to (A.27), we obtain:

$$U_{\theta+x_1+\ldots+x_M} \geq U_\theta + \sum_{i=1}^{M} [U_{x_i} + \beta_{\theta,x_i}] \tag{A.28}$$

Now applying the monotonically increasing, convex function $f(u)$ to both sides of (A.28) and writing $f(U_\theta + x) = f(U_\theta) + g(U_\theta, x)$ (from Fact 1), we have:

$$\begin{aligned} f(U_{\theta+x_1+\ldots+x_M}) &\geq f(U_\theta) + g\left(U_\theta, \sum_{i=1}^{m} [U_{x_i} + \beta_{\theta,x_i}]\right) && \text{(A.29)} \\ &\geq f(U_\theta) + \sum_{i=1}^{M} g\left(U_\theta, [U_{x_i} + \beta_{\theta,x_i}]\right) && \text{(A.30)} \end{aligned}$$

Inequality (A.30) holds by application of Fact 2, as $g(U, x)$ is a convex function of x that is zero at $x = 0$. Now notice that $\mathbb{E}f\left(U[\lambda + \delta]\right) - \mathbb{E}f\left(U[\lambda]\right) = \mathbb{E}f\left(U_\theta + x_i\right) - \mathbb{E}f\left(U_\theta\right) = \mathbb{E}\left\{g(U_\theta, U_{x_i} + \beta_{\theta,x_i})\right\}$ for any θ substream of rate λ, and any disjoint x_i substream of rate δ. Hence, by assumption:

$$\mathbb{E}\left\{g(U_\theta, U_{x_i} + \beta_{\theta,x_i})\right\} \geq \epsilon > 0 \tag{A.31}$$

Taking expectations of (A.30) and using (A.31), we find:

$$\mathbb{E}f\left(U_{\theta+x_1+\ldots+x_M}\right) \geq \mathbb{E}f\left(U_\theta\right) + M\epsilon \tag{A.32}$$

Inequality (A.32) above holds whenever $\theta + x_1 + \ldots + x_M$ is a substream of the entire, infinitely splittable process $X(t)$. We now choose M large enough so that $M\epsilon$ is greater than the expectation of $f(U)$ when the entire input $X(t)$ is applied, i.e., $M\epsilon > \mathbb{E}f\left(U_X\right)$. However, we choose a rate δ for each of the x_i substreams that is small enough so that we can still find M such substreams to ensure $\theta + x_1 + \ldots + x_M$ is still a component process

of $X(t)$. This implies that, from (A.32), $\mathbb{E}f\left(U_{\theta+x_1+\ldots+x_M}\right) > \mathbb{E}f\left(U_X\right)$, which contradicts monotonicity. Hence, $\epsilon = 0$, (A.25) holds, and the theorem is proven. □

Appendix A.B — A simple counterexample for packet occupancy

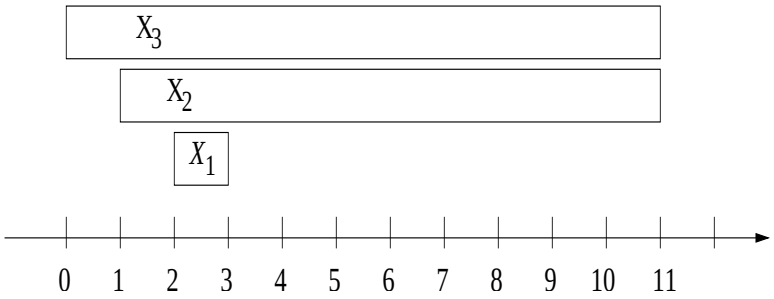

Figure A-10: A timing diagram with packets X_3, X_2, X_1 arriving at times $0, 1, 2$, respectively, illustrating that the occupancy blocking function $\alpha_{X_1,X_2}(t)$ does not have the monotonicity property when packets have variable lengths.

Here we show that under FIFO service with variable length packets, the monotonicity property of the packet occupancy blocking function does not hold, i.e., it is *not* true that $\alpha_{X_1+X_2,X_3}(t) \geq \alpha_{X_1,X_3}(t)$ for all time t.[8]

The counterexample, illustrated in Fig. A-10, is to consider streams X_1, X_2, X_3 consisting only of one packet each, where:

-The X_3 packet enters at time 0 with service time 11.

-The X_2 packet enters at time 1 with service time 10.

-The X_1 packet enters at time 2 with service time 1.

We look at time $t = 4$. At this time, we have: $N_{X_1}(4) = 0$. When X_1 and X_2 are combined, the X_2 packet blocks the X_1 packet from being served, hence $N_{X_1+X_2}(4) = 2$. Likewise, $N_{X_1+X_3}(4) = 2$, since the X_2 and X_3 packets are both long in comparison to the X_1 packet. However, because of this, when the X_3 packet is applied to a queue with the X_1 and X_2 packets, it will -The X_1 packet enters at time 2 with service time 1.

We look at time $t = 4$. At this time, we have: $N_{X_1}(4) = 0$. When X_1 and X_2 are combined, the X_2 packet blocks the X_1 packet from being served, hence $N_{X_1+X_2}(4) = 2$.

[8]Note that in cases of non-FIFO service, simple counterexamples can be constructed to show that the occupancy blocking function satisfies neither the monotonicity property nor the non-negativity property.

Likewise, $N_{X_1+X_3}(4) = 2$, since the X_2 and X_3 packets are both long in comparison to the X_1 packet. However, because of this, when the X_3 packet is applied to a queue with the X_1 and X_2 packets, it will not generate any extra packets due to blocking. Hence, $N_{X_1+X_2+X_3}(4) = 3$, and:

$$N_{X_1+X_2+X_3}(4) + N_{X_1}(4) = 3 < 4 = N_{X_1,X_2}(4) + N_{X_1,X_3}(4) \tag{A.33}$$

Thus, $\alpha_{X_1+X_2,X_3}(4) < \alpha_{X_1,X_3}(4)$, completing the example.

Appendix A.C — A Simple Counterexample for Load Balancing

Here we show that allocating exchangeable inputs to homogeneous queues according to the load balanced strategy does not necessarily stochastically minimize the total workload at every instant of time. That is, while it is true that the expected sum of moments is minimized, it is *not generally true* that for all values z and at every instant of time, the probability the total unfinished work in a load balanced system is greater than z is less than or equal to the corresponding probability in an unbalanced system.

Indeed, consider the following example of routing four streams to two queues with unit server rates. All streams deliver fixed length packets with unit packet lengths. Three of the streams have periodic packet arrivals with rates 0.2, while the fourth stream has periodic arrivals with rate 0.9. The first packet from all four streams arrives at time 0. The streams are randomly permuted so that it is impossible to know which is the high-rate stream, and hence the four streams are exchangeable.

The load balanced strategy is to allocate two streams to each queue, while the unbalanced strategy is to allocate three streams to the first queue and the remaining stream to the second queue. Under the balanced strategy, there must be one queue which is unstable, receiving periodic arrivals of total rate 1.1. It follows that the total unfinished work in this system is at least $0.1t$ at every instant of time. Thus, with probability 1, the unfinished work is greater than 5 whenever $t > 50$. However, the corresponding probability in the unbalanced system is at most 3/4. This is true because, in the case of the probability 1/4 event that the high-rate stream is placed alone in the second queue, the total number of packets in the first queue never exceeds three while the total number of packets in the second queue never exceeds one. However, one can directly verify that the expected workload in the balanced system is less than the expected workload in the unbalanced system at every instant of time, as proven for the general case by Lemma 24.

Appendix A.D — The occupancy blocking function

Here we prove the non-negativity, symmetry, and monotonicity properties of the packet occupancy blocking function $\alpha_{X_1,X_2}(t)$ when packets have fixed lengths L and service is non-preemptive work conserving. The general case of time varying processing speeds $\mu(t)$ is considered. We begin with a few simple lemmas comparing the systems of Fig. A-11.

D1. Consider the systems shown in Fig. A-11. All Systems A, B, and C have identical time-varying processing speeds $\mu(t)$. Two arbitrary input streams $X_1(t)$ and $X_2(t)$ together enter System A, and are applied individually to Systems B and C. We assume throughout that packets from the X_1 input have non-preemptive priority over X_2 packets, and that packets from the same stream are served in FIFO order. Define:

$$\begin{aligned} U^A_{X_1}(t) &= \text{Unfinished work in System } A \text{ due to } X_1 \text{ packets} \\ U^A_{X_2}(t) &= \text{Unfinished work in System } A \text{ due to } X_2 \text{ packets} \end{aligned}$$

Observation 1*:* $U^A_{X_1}(t) \geq U_{X_1}(t)$ for all time t.

Observation 2*:* $U^A_{X_2}(t) \geq U_{X_2}(t)$ for all time t.

Proof. (Observations 1 and 2) Note that $U^A_{X_1}(t)$ can be viewed as the unfinished work due to packets in a system with time varying processing rate $\mu(t)$ and a single input stream $X_1(t)$ with server vacations taking place at intervals corresponding to service intervals of X_2 packets in System A. This unfinished work is clearly less than the work $U_{X_1}(t)$ in a system with no vacations, and proves Observation 1. Observation 2 is proved similarly. □

We now consider individual packets p_1 and p_2 chosen arbitrarily from the X_1 and X_2 input streams of Fig. A-11, respectively. At any given time, we have:

Lemma D.1: If a packet p_1 from the X_1 stream is in System B, it is also in System A.

Lemma D.2: If a packet p_2 from the X_2 stream is in System C, it is also in System A.

Lemma D.3: There is at most 1 more packet from the X_1 stream in System A than in System B. In the case when there is 1 more, a packet from the X_2 stream must have been completely served during the current busy period of System A.

Proof. (Lemmas $D.1$ and $D.2$) To prove Lemma $D.1$, suppose packet p_1 enters from the X_1

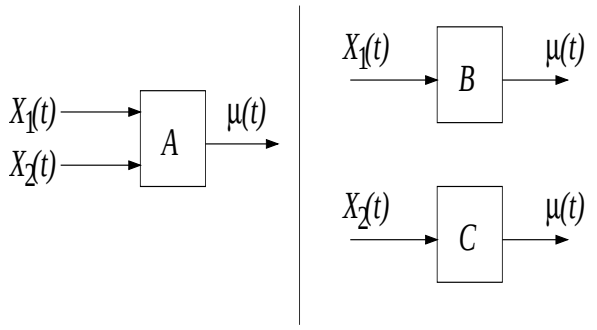

Figure A-11: Queueing systems A, B, and C with inputs $X_1(t)$ and $X_2(t)$ and time-varying processing rate $\mu(t)$.

stream at time t. In System A, this packet sees an amount of unfinished work $U^A_{X_1}(t^-) + R_{X_2}(t^-)$ in front of it (where R_{X_2} represents the residual service time if an X2 packet is in the server at time t). In System B, it sees $U_{X_1}(t^-)$. The packet exits System A at time $t + \tau_A$, and exits System B at time $t + \tau_B$, where τ_A and τ_B satisfy:

$$\begin{aligned} \int_t^{t+\tau_A} \mu(v)dv &= U^A_{X_1}(t^-) + R_{X_2}(t^-) + L \\ \int_t^{t+\tau_B} \mu(v)dv &= U_{X_1}(t^-) + L \end{aligned}$$

From Observation 1, we know $U^A_{X_1}(t^-) + R_{X_2}(t^-) \geq U_{X_1}(t^-)$, and hence $\tau_A \geq \tau_B$, proving Lemma D.1. Lemma D.2 is proved similarly. □

Proof. (Lemma D.3) Suppose there are more X_1 packets in System A than in System B at time t. Consider time t_1, the beginning of the current busy period in System A. If this busy period was completely composed X_1 packets, then System B must have an identical busy period and hence must have the same number of X_1 packets within it—which yields a contradiction.

Thus, the System A busy period must have contained an X_2 packet. Let t_2 be the time the latest X_2 packet started service in System A. Because of the priority scheme, no X_1 packets were in System A at this time (and hence—from Observation 1—System B must be empty at this time). It follows that System B can serve at most 1 more X_1 packet than System A during the interval $[t_1, t]$. Because System A has more X_1 packets at time t, it follows that it has exactly one more, that at least one departure from System B has occurred

during $[t_1, t]$, and hence that the X_2 packet that started service at time t_1 in System A has already departed. □

D2. Recall the packet occupancy blocking function is defined: $\alpha_{X_1,X_2}(t) \triangleq N_{X_1+X_2}(t) - N_{X_1}(t) - N_{X_2}(t)$

Theorem 7. *For time varying server speeds $\mu(t)$, if all packets have fixed lengths and service is non-preemptive and work conserving, the occupancy blocking function satisfies the non-negativity, symmetry, and monotonicity conditions, i.e., for all time t:*

(a) $\alpha_{X_1,X_2}(t) \geq 0$ *(Non-negativity)*

(b) $\alpha_{X_1,X_2}(t) = \alpha_{X_2,X_1}(t)$ *(Symmetry)*

(c) $\alpha_{X_1+X_2,X_3}(t) \geq \alpha_{X_1,X_3}(t)$ *(Monotonicity)*

Proof. Note that if packets are fixed in length, the number of packets in a queueing system is independent of the service order as long as service is non-preemptive and work conserving. Hence, without loss of generality, throughout we assume non-preemptive priority service with priority ordering $X_1 \rightarrow X_2 \rightarrow X_3$ (where X_1 packets receive the highest priority, etc.). To prove (a), it suffices to show that $N_{X_1+X_2}(t) \geq N_{X_1}(t) + N_{X_2}(t)$. This follows immediately from Lemmas $D.1$ and $D.2$. The symmetry property (b) is clear from the definition of $\alpha_{X_1,X_2}(t)$. Note that these conditions also hold for variable length packets when service is FIFO.

To prove the monotonicity property (c), it suffices to show that for all time t:

$$N_{X_1+X_2+X_3}(t) + N_{X_1}(t) \geq N_{X_1+X_2}(t) + N_{X_1+X_3}(t) \tag{A.34}$$

and hence that the combined number of packets in Systems A and B of Fig. A-12 below is greater than or equal to the number in Systems A' and B'. To show this, we have from Lemmas $D.1$ and $D.2$:

- If a packet p_1 from X_1 is in System A', it is also in System A
- If a packet p_2 from X_2 is in System A', it is also in System A.
- If a packet p_3 from X_3 is in System B', it is also in System A.

From Lemma $D.3$, there is at most one more packet from X_1 in System B' than in System B. If there are no more in System B' than in System B, we are done. Otherwise,

let p_1 represent the single packet from the X_1 stream which is in System B' and is not in System B. By Lemma $D.1$, we know this packet is also in System A. If it is not in System A', we are done. Otherwise, packet p_1 is also in System A', and hence by Lemma $D.3$ we know there was a packet p_2 from X_2 that was served in the current busy period of System A', as well as a packet p_3 from X_3 that was served in the current busy period of System B'. Because both p_2 and p_3 precluded service of p_1 (in Systems A' and B', respectively), if either is served in System A, the server of A must thereafter serve X_1 packets, and hence the other must be currently contained in A. Thus, System A contains either p_2 or p_3—both of which are absent from Systems A' and B'. This extra packet in System A makes up for the 1 packet deficit in System B and hence preserves inequality (A.34). □

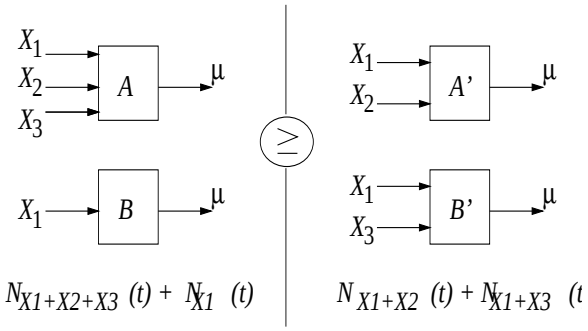

Figure A-12: A queueing illustration of the monotonicity property of the occupancy blocking function $\alpha_{X_1,X_2}(t)$ for fixed length packets.

Appendix A.E — Optimality of the Greedy Policy

Here we prove Lemma 24: Given a convex cost function $C(M_1, \ldots, k_N)$ of the form specified in Theorem 23, the optimal allocation vector can be obtained by sequentially adding streams, greedily choosing at each iteration the queue which increases the total cost $C()$ the least. This yields a cost-minimizing vector $(k_1^o, \ldots, k_N^o)$ after $M + N - 1$ evaluations/estimations of the cost function.

Proof. The lemma is clearly true for $M = 1$ stream. We assume that it is true for $M = k$ streams, and by induction prove it holds for $M = k + 1$.

Let $(k_1^o, \ldots, k_N^o)$ represent the optimal allocation vector for $M = k$ streams, which is obtained by the sequentially greedy algorithm. We thus have $\sum_i k_i^o = k$.

Now we add an additional stream in a greedy manner by placing it in the queue which increases the cost function the least. Without loss of generality, we assume this queue is queue 1, and we have a new allocation vector $(k_1^o + 1, \ldots, k_N^o)$. Suppose there is some other vector $(\tilde{k}_1^o, \ldots, \tilde{k}_N^o)$ whose elements sum to $k + 1$, such that $C(k_1^o + 1, \ldots, k_N^o) > C(\tilde{k}_1^o, \ldots, \tilde{k}_N^o)$.

Case 1 ($\tilde{k}_1 \geq \tilde{k}_1^o + 1$): In this case, we take away an input stream from the first entry of both vectors. This effects only the queue 1 term $\mathbb{E}f_1[k]$ in the cost function. Because this function is convex and non-decreasing, $\mathbb{E}f_1[k_1^o + 1]$ decreases by less than or equal to the amount that $\mathbb{E}f_1[\tilde{k}_1]$ decreases. Hence, it must be that $C(k_1^o, \ldots, k_N^o) > C(\tilde{k}_1^o - 1, \ldots, \tilde{k}_N^o)$, which contradicts the fact that the $(k_1^o, \ldots, k_N^o)$ vector is optimal for $M = k$ streams.

Case 2 ($\tilde{k}_1 < \tilde{k}_1^o + 1$): In this case, there exists some queue j such that $\tilde{k}_j > k_j^o$. We take the additional input we added to queue 1 and move it to queue j, forming a new vector $k_1^o, \ldots, k_j^o + 1, \ldots, k_N^o)$. Notice that this change cannot decrease the cost function, since this input was originally added greedily to queue 1. We now have $\tilde{k}_j \geq k_j^o + 1$, which reduces the problem to Case 1.

Thus, the sequentially greedy algorithm is optimal. It can be implemented with $M + N - 1$ evaluations of the cost function by keeping a record of the $N - 1$ queue increment values for the $N - 1$ queues not chosen at each step. □

Appendix B

Routing in Finite Buffer Queues

In this appendix, we consider the problem of routing packets from an arbitrary input stream $X(t)$ over a collection of heterogeneous queues in parallel. When the processing rates $(\mu_1, \ldots, \mu_N)$ of the queues are constant, a simple work conserving routing strategy π_{WC} is shown to hold total system backlog within a fixed upper bound from the resulting backlog of any other policy. Similar results apply to systems with time varying processing rates $(\mu_1(t), ..., \mu_N(t))$ when routing decisions can be postponed by placing packets in a pre-queue.

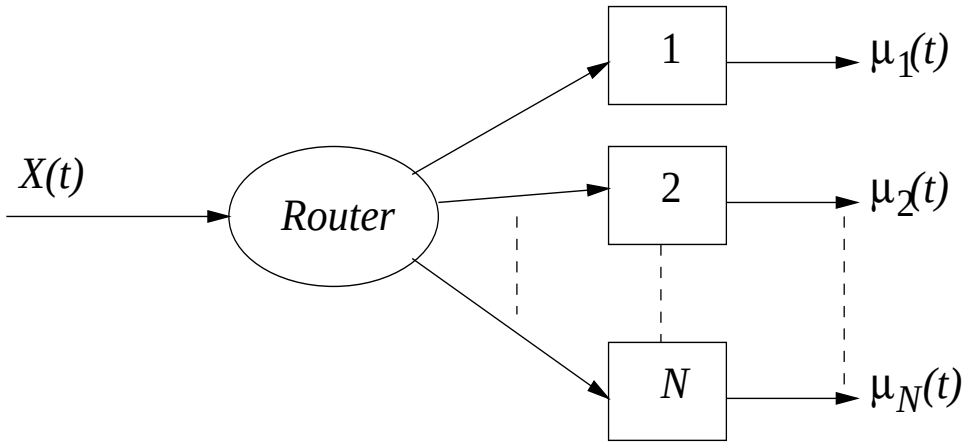

Figure B-1: Routing over a set of parallel queues with time-varying processing speeds $\mu_i(t)$.

For the case when routing decisions must be made immediately upon arrival, it is impossible to bound the number of packets or the amount of unprocessed bits in the queues unless complete knowledge of future events is known. However, we demonstrate that the simple non-predictive policy of routing packets to the shortest queue ensures system sta-

bility whenever possible. Of particular interest in this appendix is our treatment of finite buffer queueing analysis. We consider the *Join-the-Shortest-Queue* policy when all queues have finite buffers, and establish upper and lower bounds on packet drop rates. Stability properties are proven under the following new notion of stability: A finite buffer system is *stable* if the packet drop rate can be made arbitrarily small by increasing the buffer size.

We apply these results to a joint problem of routing and power allocation, where for each time varying channel state $S_i(t)$, the rate of each queue i can be varied by adjusting a power parameter P_i (subject to power constraints) according to a rate-power curve $\mu_i(S_i, P_i)$. A throughput maximizing algorithm is developed for this joint problem. This work supplements our analysis presented in Chapter 3. Indeed, in Chapter 3 we developed a power allocation scheme for a satellite downlink and proved stability of the *Join-the-Shortest-Queue* routing strategy using a Lyapunov function for infinite buffer systems. Our approach to the finite buffer problem uses a completely different technique.

Previous work on routing and queue control policies is found in [147] [45] [142] [60] [145] [146] [135] [17] [110] [108] [105]. In [147] an exact analysis of the *Join-the-Shortest-Queue* policy for parallel queues is developed for M/M/1 systems. In [45] the *Join-the-Shortest-Queue* strategy is shown to be optimal for minimizing average backlog in a system with an arbitrary packet arrival process entering a system of two queues with homogeneous exponential servers. An extension to the case when the two servers have different service rates is treated in [142], where an optimal threshold policy is developed using dynamic programming. Shortest queue optimality results similar to [45] are developed in [145] [146] for multiple homogeneous queues, and in [135] for homogeneous queues with finite buffers. Static or 'backlog-blind' routing policies such as round robin or probabilistic splitting are considered with various assumptions on the inputs in [45] [90] [89] [3] [25] [80] [56] [32] [14] [21] [130] [19] [22] [105]. In Appendix A, we develop convexity properties of queueing systems with general $*/*$ inputs and time varying server rates, and use them to establish an optimal static policy for routing multiple data streams over N parallel queues.

A related NP-complete problem of batch packet arrivals is considered in [52] [30] where the goal is to route packets to parallel queues to minimize the total delay for transmitting one batch. In [126] [9] [74] an online job scheduling problem is considered where N identical servers work on K jobs which arrive randomly over an interval of time $[0, T]$. Simple algorithms which finish within twice the minimum possible completion time are developed.

The main contribution in this paper is to treat stochastic queueing systems and to provide tight, worst case bounds on system performance for arbitrary input processes. Using sample path techniques, we develop additive bounds on the amount of unfinished work in the queues for the entire timeline $t \geq 0$. For finite buffer systems, we present additive bounds on packet drops. Such bounds directly translate into statements about queue stability and buffer requirements in the system. This approach differs significantly from the dynamic programming techniques which address systems with particular types of input and server processes.

In the next section we introduce the routing problem by comparing two natural routing strategies: a greedy routing strategy and a work conserving routing strategy. In Section B.2 a simple queueing inequality is developed for time-varying systems and is used as a tool for comparing routing schemes and proving performance bounds. In Sections B.3 and B.4 we treat routing in systems with a pre-queue and without a pre-queue, respectively, and in Section B.5 we apply these results to a power allocation problem. To provide the most general results while facilitating intuition, all buffers are assumed to be infinite until the finite buffer analysis is presented in Section B.4.

B.1 The Greedy and Work Conserving Algorithms

Consider the system of Fig. B-1 with an arbitrary arrival stream $X(t)$ sending packets to be routed over the N queues. Assume all processing rates $(\mu_1, \ldots, \mu_N)$ are constant. The goal is to route packets in a manner that ensures an acceptably low level of unfinished work $U(t)$ and number of packets $N(t)$ in the system for all time. It can be shown that a policy π_{greedy} (described below) is optimal in minimizing $N(t)$ at every instant of time if all packets have fixed lengths and arrive in a single burst at time zero. However, for general streams $X(t)$ with arrivals occurring over the timeline $t \in [0, \infty)$, it is not possible to minimize $N(t)$ or $U(t)$ at every instant of time—even if the entire future is known in advance. Here we seek a robust strategy, one whose performance at every instant of time t is sufficiently close to that of a system optimized to minimize backlog at that particular time instant.

Two natural routing strategies emerge, the greedy strategy π_{greedy} and the work conserving strategy π_{WC}:

1. The greedy strategy routes the current packet i to the queue that allows it to exit

first:

π_{greedy}: Choose queue k such that $k = \arg\min_{j\in\{1,\ldots,N\}} \left\{ \frac{L_i + U_j(t)}{\mu_j} \right\}$

(where L_i is the length of the current packet i, and $U_j(t)$ is the unfinished work in queue j at time t).

2. The work conserving strategy routes to the queue which will empty first:

π_{WC}: Choose queue k such that $k = \arg\min_{j\in\{1,\ldots,N\}} \left\{ \frac{U_j(t)}{\mu_j} \right\}$

Notice that policies π_{greedy} and π_{WC} are identical if all server speeds μ_j are the same, although they may differ considerably under heterogeneous server rates. Because the greedy strategy uses the length of the current packet when making a routing decision, one would expect this policy to offer better performance. However, for suitable choices of the linespeeds $(\mu_1, \ldots, \mu_N)$, a system under the greedy strategy can have arbitrarily more unfinished work within it than the same system operating under the work conserving strategy, as the following example illustrates.

Suppose a burst of B packets arrive to the system at time 0. After this initial burst, a single packet arrives periodically at times $\{1, 2, 3, \ldots\}$. Assume all packets have fixed lengths $L = 1$, and that $(\mu_1, \mu_2, \ldots, \mu_N) = (1, \epsilon, ..., \epsilon)$. Suppose that $\epsilon > 0$ is sufficiently small to ensure that all packets from the initial burst as well as all packets thereafter are routed to queue 1 under strategy π_{greedy}. Thus, under the greedy strategy, there are always B packets in the system.

Now consider a different set of routing decisions (which we represent as policy π): route the B packets from the initial burst amongst the $N-1$ queues of rate ϵ, and route all packets from the periodic stream thereafter to queue 1. Under this policy, queue 1 always has exactly one packet within it. However, the B packets from the initial burst eventually depart from queues $\{2, 3, \ldots, N\}$, leaving these queues empty after some finite time T. Hence, after time T the greedy strategy π_{greedy} results in $B-1$ *more packets* in the system than policy π, where $B-1$ can be made arbitrarily large. Alternatively, in Section B.2 it is shown that the work conserving strategy π_{WC} is fundamental in that it never produces more than $N-1$ extra packets in the system compared to any other policy.

B.2 A Multiplexing Inequality

Here we develop a queueing inequality useful for establishing performance bounds on routing policies. Let $X(t)$ represent an arbitrary packet arrival process on the interval $[0, \infty)$. A particular $X(t)$ sample path is a non-decreasing staircase function representing the total number of bits that have arrived to the system during $[0, t]$. Jumps in the $X(t)$ function occur at packet arrival epochs and have magnitudes equal to the length of the arriving packet. We assume there is a maximum packet length L_{max}.

Consider the single server and multi-server queueing systems of Fig. B-2. The linespeed processes $\mu(t)$ and $\{\mu_i(t)\}$ represent instantaneous server processing rates (in units of bits per second). Here we assume that the rate of the single server queue of Fig. B-2a is equal to the sum of the individual server rates in Fig. B-2b, i.e., $\mu(t) = \mu_1(t) + \ldots + \mu_N(t)$. Assume both systems of Fig. B-2 are initially empty and the same input process $X(t)$ is applied to each. Packets are immediately processed in the single server system according to a non-idling service policy. However, in the multi-server system packets are routed to the N queues using any conceivable routing mechanism. All buffers in the queues and in the router device are assumed to be infinite, so that no packets are lost. Let $U_{single-server}(t)$ represent the unfinished work (in bits) in the single server queue, and let $U_{multi-server}(t)$ represent the total amount of unfinished bits in the multi-queue system.

Lemma 3. *(Multiplexing Inequality): For all time $t \geq 0$:*

$$U_{single-server}(t) \leq U_{multi-server}(t) \tag{B.1}$$

Proof. Let $D_{single-server}(t)$ and $D_{multi-server}(t)$ represent the amount of processed bits or "departures" from the single server system and multi-server system, respectively, during $[0, t]$. Clearly we have $D_{single-server}(t) = X(t) - U_{single-server}(t)$ and $D_{multi-server}(t) = X(t) - U_{multi-server}(t)$, and it suffices to show that $D_{single-server}(t) \geq D_{multi-server}(t)$. Notice that the departure functions are both initially 0 at time $t = 0$. Whenever the single-server system is empty, $D_{single-server}(t) = X(t) \geq D_{multi-server}(t)$, and hence the inequality is satisfied at such times. If the single-server queue is not empty, then it is processing packets at the instantaneous rate $\mu(t)$, which is greater than or equal to the instantaneous departure rate of bits from the multi-server system. Hence, the departures from the multi-server

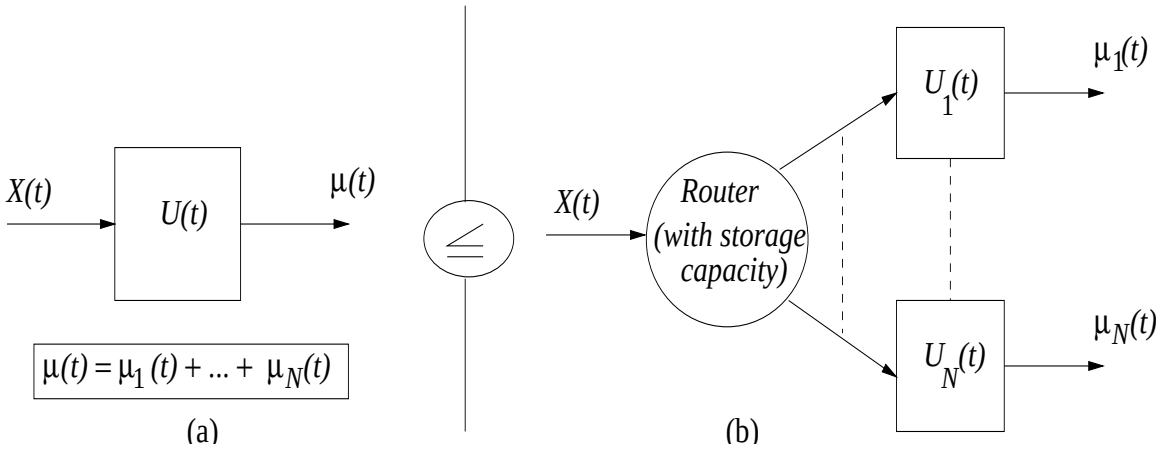

Figure B-2: A single server queue and a set of multi-server queues with an arbitrary router device. The sum of the time varying server rates of the servers in (b) equals the rate $\mu(t)$ in (a).

system can never overtake the departures from the single-server queue. □

A finite buffer version of this statement is given in Section B.4. This multiplexing inequality demonstrates that it is always better to multiplex data streams from individual queues to a single queue whose rate is equal to the sum of the individual processing rates. It is useful to consider such a virtual single-server queue to provide a baseline for measuring the performance of routing policies. Strategies yielding unfinished work functions close to the $U_{single-server}(t)$ lower bound are desirable.

Consider now a particular work conserving routing strategy π_{WC} that operates on the multi-queue system of Fig. B-2b. The policy π_{WC} places all incoming packets in a shared buffer device or "pre-queue." Whenever any server becomes available, the pre-queue instantaneously routes the next packet to that server. If there is more than one server available, the choice is made arbitrarily. Thus, the policy π_{WC} is "work conserving" in that no servers are idle whenever there are buffered packets waiting to be processed. Let $U_{WC}(t)$ represent the total unfinished work at time t in the multi-server system of Fig. B-2b under this policy.

Lemma 4. *(Performance Tracking): At every instant of time t:*

$$U_{single-server}(t) \leq U_{WC}(t) \leq U_{single-server}(t) + (N-1)L_{max} \tag{B.2}$$

Proof. The first inequality is just a particular case of the multiplexing inequality (Lemma 25). To prove the second inequality, we compare the departed bits $D_{WC}(t)$ and $D_{single-server}(t)$.

It suffices to show that $D_{single-server}(t) \leq D_{WC}(t) + (N-1)L_{max}$. For simplicity, assume that $\mu_i(t) < \infty$ for all time t and all i, so that departures drain continuously from the queues. For the above departure inequality to be violated, there must be some crossing time t^* such that $D_{single-server}(t^*) = D_{WC}(t^*) + (N-1)L_{max}$. If the single-server system is empty at this time, the departure function $D_{single-server}(t)$ cannot increase, and hence t^* cannot be a crossing time. Otherwise, the multi-server system holds strictly more than $(N-1)L_{max}$ bits, and hence contains at least N distinct packets. By the nature of the work conserving policy π_{WC}, all servers of the multi-server system must be actively processing these packets. The departure functions $D_{single-server}(t)$ and $D_{WC}(t)$ are thus increasing at the same rate at time t^*, and the departures from the single-server system cannot overtake the bound. □

Notice that the bounds of Lemma 26 imply that the work conserving routing strategy π_{WC} is stable (in the usual sense of infinite buffer systems) if and only if the single queue system with the same inputs is stable. Stability issues for finite buffers systems are addressed further in Section B.4.

B.3 Systems with Pre-Queues

The results of the previous section allow routing policy π_{WC}—which is implemented with a pre-queue—to be compared to any other policy π which operates on the same multi-server system. Here we show π_{WC} is *minimax optimal*. Let $U_{WC}(t)$, $N_{WC}(t)$, $U_\pi(t)$, and $N_\pi(t)$ represent the unfinished work and number of packets in the multi-queue system of Fig. B-2b under policies π_{WC} and some other (arbitrary) policy π, respectively.

B.3.1 Variable Length Packets

Here we treat systems with variable length packets. All packets are bounded by a maximum packet length L_{max}. We show that the π_{WC} routing strategy provides the best worst-case performance guarantee of all policies for routing packets non-preemptively over multiple time varying servers.

A policy π is *non-preemptive* if it does not interrupt a packet that has initiated processing at a server. If the policy does not require knowledge of the future, we say it is *non-predictive*. If a policy π has full knowledge of future events, it can be designed to minimize unfinished

work at some particular time instant τ. Let $\Phi_{X,\vec{\mu}}(\tau)$ represent this minimum value of unfinished work in a system with input stream $X(t)$ and linespeeds $\vec{\mu}(t) = (\mu_1(t), \ldots, \mu_N(t))$.

Consider a game where a scheduler makes routing decisions (according to some non-predictive policy π) while an adversary dynamically creates an input stream $X(t)$ and varies the linespeeds $(\mu_1(t), \ldots, \mu_N(t))$ in order to maximize the difference between $U_\pi(t)$ and $\Phi_{X,\vec{\mu}}(\tau)$. The goal of the scheduler is to minimize the worst case deviation from optimality, i.e., minimize the value of $\max_{t\geq 0}\left\{U_\pi(t) - \Phi_{X,\vec{\mu}}(t)\right\}$.

Theorem 8. *(a) For all policies π (possibly predictive and preemptive), we have:*

$$U_{WC}(t) \leq U_\pi(t) + (N-1)L_{max} \tag{B.3}$$

(b) The work conserving policy π_{WC} is the minimax optimal non-predictive, non-preemptive routing strategy over the set of all possible inputs and linespeed variations.

Proof. Part (a) follows immediately from Lemmas 25 and 26. To establish that the policy π_{WC} is minimax optimal, it suffices to show that any non-predictive, non-preemptive policy π can be forced to meet the $(N-1)L_{max}$ bound. The idea is for an adversary to force policy π to route maximum length packets to distinct servers, and then to trap these packets by setting their server rates to 0. Specifically, the adversary sends $(N-1)$ maximum length packets at time 0. The adversary then maintains a constant output rate of $\mu_i(t) = \mu$ for all servers i that have no packets within them. Whenever a packet is routed to a server under policy π, that server rate is set to 0. After $(N-1)L_{max}/\mu$ seconds have elapsed, no unfinished work has been processed, and there must be some server j^* that has remained empty for the full time interval, with an unused processing rate $\mu_j(t) = \mu$. Hence, given this particular sample path of server rates and packet arrivals, an alternative routing scheme that sends all packets to server j^* would have allowed the system to be empty at this time, and so the $(N-1)L_{max}$ bound is met with equality. □

B.3.2 Fixed Length Packets

In the case when all packets have fixed lengths, a version of inequality (B.3) can be established in terms of the number of packets $N(t)$ in the system.

Theorem 9. *If all packets have fixed lengths L, then:*

$$N_{WC}(t) \leq N_{\pi}(t) + N - 1 \tag{B.4}$$

Proof. Define a *completely busy period* as an interval of time when all servers of the parallel queue system are busy. Because π_{WC} never buffers packets if a server is idle, the inequality holds whenever t is not within a completely busy period. Suppose now that t lies within a completely busy period, and let τ_B be the beginning time of this period. We have:

$$N_{WC}(t) = N_{WC}(\tau_B^-) + A(\tau_B, t) - D_{WC}(\tau_B, t) \tag{B.5}$$

$$N_{\pi}(t) = N_{\pi}(\tau_B^-) + A(\tau_B, t) - D_{\pi}(\tau_B, t) \tag{B.6}$$

where τ_B^- represents the time just before the arrival initiating the completely busy period, $A(\tau_B, t)$ is the number of arrivals during the interval $[\tau_B, t]$, and $D_{WC}(\tau_B, t)$, $D_{\pi}(\tau_B, t)$ respectively represent the number of packet departures from the π_{WC} system and the π system during this interval. The above departure functions are composed of individual terms D_{WC}^i and D_{π}^i representing departures from queue i:

$$D_{WC}(\tau_B, t) = \sum_{i=1}^{N} D_{WC}^i \ , \ D_{\pi}(\tau_B, t) = \sum_{i=1}^{N} D_{\pi}^i \tag{B.7}$$

During the time interval $[\tau_B, t]$, each queue of the π_{WC} system continuously processes fixed length packets. Thus, $D_{WC}^i \geq D_{\pi}^i - 1$ for all queues i, and equality holds only if there was a packet being processed by queue i under the π routing policy at time τ_B^-. Suppose there are k such cases, so that $N_{\pi}(\tau_B^-) = k$ (where $k \leq N$). Thus, $D_{WC}(\tau_B, t) \geq D_{\pi}(\tau_B, t) - k$, and we have:

$$N_{WC}(t) \leq N_{WC}(\tau_B^-) + A(\tau_B, t) - D_{\pi}(\tau_B, t) + k \tag{B.8}$$

$$= N_{WC}(\tau_B^-) + N_{\pi}(t) - N_{\pi}(\tau_B^-) + k \tag{B.9}$$

$$\leq (N-1) + N_{\pi}(t) - k + k \tag{B.10}$$

where (B.9) follows from (B.6), and (B.10) follows because the work conserving strategy contains fewer than N packets just before the completely busy period. □

A technique similar to the one used in the proof of Theorem 26 shows that this $(N-1)$ bound is tight and is minimax optimal over all non-predictive, non-preemptive strategies.

Similar routing and scheduling problems have been treated in [45] [142] [60] [144] for systems with two heterogeneous servers ($N = 2$) with memoryless assumptions on the server or arrival processes. With such a formulation applied to the problem of routing Poisson inputs with fixed-length packets in a two-queue system, it is possible to prove that the optimal routing strategy for minimizing expected occupancy in the system has a threshold structure. A complex dynamic program can be developed to numerically compute the exact threshold function. However, here we find that—with arbitrary input processes $X(t)$—the simple work conserving strategy π_{WC} ensures no more than one extra packet in the system compared to any other strategy at any time.

B.4 Systems Without a Pre-Queue

B.4.1 Constant Rate Servers

The implementation of the work conserving policy π_{WC} for time varying servers uses a pre-queue to store packets until the next server becomes available. In many systems it is undesirable or even impossible to implement a pre-queue. For example, in a satellite network, queues might be aboard different satellites, which may require routing decisions to be made immediately upon packet arrival. Here we show that the same results can be obtained in systems without a pre-queue if the server rates $(\mu_1, \ldots, \mu_N)$ are known constants.

Observation: For constant server rates $(\mu_1, \ldots, \mu_N)$, the strategy of routing an incoming packet to the queue k with the smallest value of $U_k(t)/\mu_k$ (the π_{WC} strategy as described in Section B.1) is the same as the π_{WC} strategy described for time varying servers in Section B.2. Thus, a pre-queue is not needed.

Proof. The strategy always routes a new packet to the queue that will empty first. Thus, if there is ever an empty server i, there can be no more than 1 packet in each of the $(N-1)$ other queues. □

Thus, the bounds in Theorems 26 and 27 apply to heterogeneous, constant rate servers when this routing method is used.

B.4.2 Time Varying Servers

Consider routing over a multi-queue system with time-varying processing rates when no pre-queue is available and all routing decisions are made immediately upon packet arrival. We examine the *Join-the-Shortest-Queue* (*JSQ*) policy: route the current packet to the queue i with the smallest value of $U_i(t)$. Intuitively, this strategy is the closest match to the work conserving strategy given that we cannot predict future values of server rates.

We seek to prove that this strategy stabilizes the multi-queue system whenever it is stabilizable. This is done for general ergodic input sources and linespeed processes by introducing a new notion of stability defined in terms of finite buffer systems. Consider the single queue system of Fig. B-3a with an ergodic input process $X(t)$ of bit rate λ, a linespeed process $\mu(t)$ with ergodic rate μ_{av}, and assume this system has a finite buffer capacity of M bits. We assume that a full packet of size L is dropped if it arrives when $M - U(t) < L$. Let $G^M(t)$ represent the total bits dropped (or placed into the *Garbage*) during $[0, t]$ when the buffer size is M. Further let $DR(M)$ represent the *drop rate* (measured in bits) of the system as a function of buffer space M:

$$DR(M) = \limsup_{t \to \infty} \frac{G^M(t)}{t}$$

Definition 7. *A system is* loss rate stable *if the drop rate can be made arbitrarily small by increasing buffer capacity, i.e., if* $DR(M) \to 0$ *as* $M \to \infty$.

For the remainder of this appendix, we consider only finite buffer systems, and use the term *stable* to mean *loss-rate-stable*.

It can be shown that a necessary condition for loss rate stability is $\lambda \leq \mu_{av}$. Furthermore, if the input stream and server rate processes evolve according to an underlying finite state ergodic Markov chain, a sufficient condition for loss rate stability is $\lambda < \mu_{av}$. This notion of stability is closely tied to the standard notion defined in terms of a vanishing complementary occupancy distribution for infinite buffer capacity queues [98] [6] [81] [88] [132] [95].

Before analyzing the *JSQ* strategy, we present a finite buffer version of the multiplexing inequality. Consider the two systems of Fig. B-3. The server rates $\mu_1(t), \ldots, \mu_N(t)$ of the multi-queue system sum to the single-server linespeed $\mu(t)$. Suppose the single queue system has buffer size M, while the multi-queue system has buffer sizes $M_1, \ldots, M_N$. Let $G_{single}(t)$ represent the total bits dropped by the single server system during $[0, t]$, and let

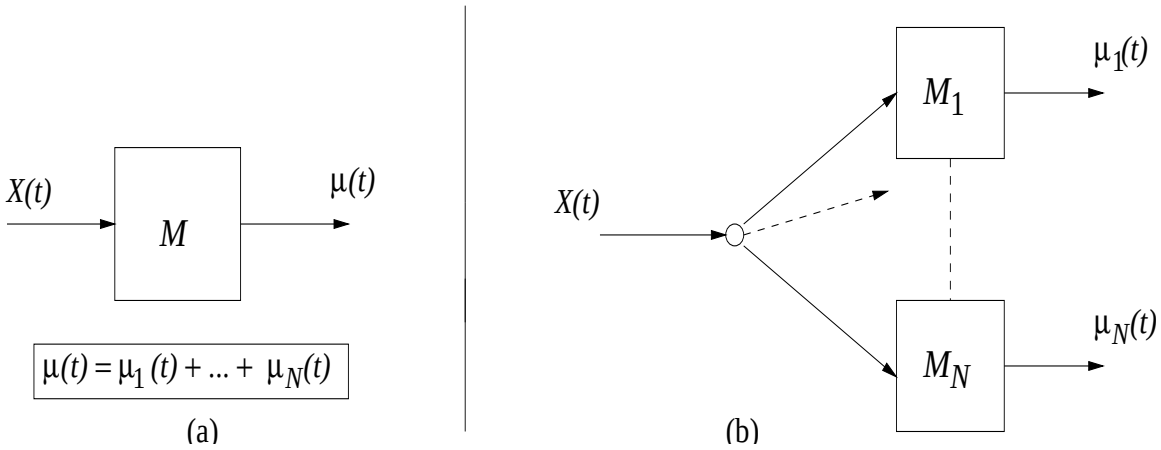

Figure B-3: A single stage system with a finite buffer of size M and an aggregated server processing rate $\mu(t)$ compared to a multi-queue system with finite buffers.

$G_{multi}(t)$ represent the bits dropped by the multi-server system with the same inputs (using any arbitrary set of routing decisions). Both systems are assumed empty at time 0.

Theorem 10. *(Finite Buffer Multiplexing Inequality): For arbitrary inputs $X(t)$, if $M \geq M_1 + \ldots + M_N + L_{max}$, then for all t:*

(a) Departures satisfy: $D_{single}(t) \geq D_{multi}(t)$

(b) Packet drops satisfy: $G_{single}(t) \leq G_{multi}(t)$

Proof. See Appendix B.A. □

Thus, a single-queue system in which all buffer slots and linespeeds are aggregated (with an additional buffer slot of size L_{max}) always outperforms the multi-queue system. We now consider the particular routing strategy *JSQ*. Let $DR_{JSQ}(M)$ represent the drop rate of the multi-queue system (operating under the *JSQ* policy) when all queues have finite buffer storage M.

Theorem 11. *For all buffer sizes M:*

$$DR_{JSQ}(M + NL_{max}) \leq DR_{single-queue}(M) \tag{B.11}$$

and hence a multi-queue system under the JSQ *strategy with a buffer size of $M + NL_{max}$ in each queue drops fewer packets than the single queue with buffer size M.*

Proof. : We prove a stronger result: $G_{JSQ}(t) \leq G_{single}(t)$ for all $t \geq 0$, where $G_{JSQ}(t)$ and $G_{single}(t)$ respectively represent the total bits dropped by the multi-queue system (under

JSQ) and the single queue system. Suppose this inequality is first violated at some time τ (we reach a contradiction). It follows that an arriving packet must have been dropped by the *JSQ* system at this time, and thus all servers of the multi-queue system are busy. Let t_B represent the start of this completely busy period, so that bits depart from the *JSQ* system at the full service rate during $[t_B, \tau]$. Let $U_{JSQ}(t)$ represent the unfinished work in the *JSQ* system at time t. Further define:

$$\begin{aligned} a &\triangleq \text{Arrivals during } [t_B, \tau] \\ d_{JSQ} &\triangleq \text{Bits processed by the } JSQ \text{ system during } [t_B, \tau] \\ g_{JSQ} &\triangleq \text{Bits dropped by the } JSQ \text{ system during } [t_B, \tau] \end{aligned}$$

Define d_{single}, g_{single}, and $U_{single}(t)$ similarly for the single queue system. Note that $g_{JSQ} > g_{single}$ because the packet drop inequality $G_{JSQ}(t) \leq G_{single}(t)$ is first violated at time τ. The following bit-conservation equalities hold:

$$U_{JSQ}(\tau) = U_{JSQ}(t_B^-) + a - d_{JSQ} - g_{JSQ} \tag{B.12}$$

$$U_{single}(\tau) = U_{single}(t_B^-) + a - d_{single} - g_{single} \tag{B.13}$$

Just before the completely busy period, at least one queue of the multi-server system is empty, and hence:

$$U_{JSQ}(t_B^-) \leq (N-1)\left[M + NL_{max}\right] \tag{B.14}$$

Because a packet is dropped by the *JSQ* system at time τ, all queues must have more than $[M + (N-1)L_{max}]$ unfinished work within them, and hence:

$$U_{JSQ}(\tau) > N\left[M + (N-1)L_{max}\right] \tag{B.15}$$

Using (B.14) and (B.15) in (B.12), we have:

$$N\left[M + (N-1)L_{max}\right] < (N-1)\left[M + NL_{max}\right] + a - d_{single} - g_{single}$$

and hence

$$a - d_{JSQ} > M + g_{JSQ} \tag{B.16}$$

The unfinished work in the single queue can thus be bounded:

$$\begin{aligned} U_{single}(\tau) &\geq a - d_{single} - g_{single} && \text{(B.17)} \\ &\geq a - d_{JSQ} - g_{single} && \text{(B.18)} \\ &> M + g_{JSQ} - g_{single} && \text{(B.19)} \end{aligned}$$

where (B.17) follows from (B.13), (B.18) follows because the *JSQ* system processes packets at the full rate $\mu(t)$ during $[t_B, \tau]$, and (B.19) follows from (B.16). Now, because of the finite buffer constraint, $M \geq U_{single}(\tau)$, and hence (B.19) yields $g_{JSQ} < g_{single}$, a contradiction. □

Theorems 28 and 29 imply that

$$DR_{single}(MN + (N^2+1)L_{max}) \leq DR_{JSQ}(M + NL_{max}) \leq DR_{single}(M)$$

and hence the multi-queue system under the *JSQ* routing strategy is stable if and only if the corresponding single queue system is stable. Furthermore, (B.11) provides a simple and useful bound on the packet drop rate in the multi-queue system in terms of a single queue with a finite buffer. Inequality (B.11) can be used together with the finite buffer multiplexing inequality to bound the performance of *JSQ* in terms of any other routing policy π:

$$DR_{JSQ}(M) \leq DR_{\pi}\left(\frac{M}{N} - L_{max}\frac{N+1}{N}\right)$$

where $DR_{\pi}(V)$ represents the drop rate in the multi-queue system with the same input stream but with any arbitrary (possibly anticipatory) routing policy π, for the case when all queues have buffer size V.

B.5 Joint Routing and Power Allocation

Suppose that the transmission rates $(\mu_1, \ldots, \mu_N)$ of the system can be controlled by adjusting power levels P_i allocated to each server. Specifically, suppose that each channel $i \in \{1, 2, \ldots, N\}$ has an associated channel state S_i which takes values on a finite set of states ($S_i \in \{C_1^i, C_2^i, \ldots, C_{M_i}^i\}$). Let $\mu_i(P_i, S_i)$ represent a concave rate-power curve for each channel state (see Fig. B-4). We assume that the individual queues belong to a collection

of J sub-units, and let V_j represent the set of queues $i \in \{1, \ldots, N\}$ belonging to sub-unit j. The sub-units could represent distinct satellites of a satellite network, or different basestations in a wireless network. Each sub-unit j has its own power resource with total power $P_{tot}^{(j)}$.

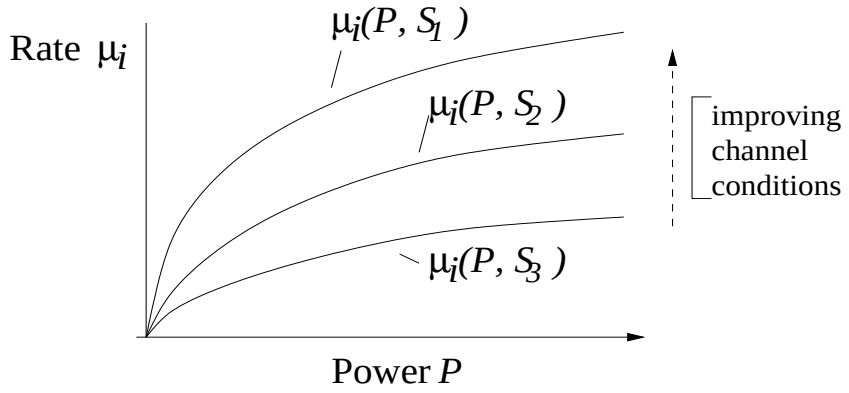

Figure B-4: A set of concave power curves $\mu_i(P_i, S)$ for channel states S_1^i, S_2^i, S_3^i.

Let $\vec{S}(t) = (S_1(t), S_2(t), \ldots, S_N(t))$ represent the vector of channel states at time t, and assume that $\vec{S}(t)$ varies according to a discrete time Markov chain with timeslots of length T. As before, packets enter the system according to an input process $X(t)$ and routing decisions are made immediately upon arrival. In addition to making routing decisions, a controller must choose a power allocation $\vec{P}(t) = (P_1(t), \ldots, P_N(t))$ for each instant of time, subject to the total power constraints of each sub-unit: $\sum_{i \in V_j} P_i(t) \leq P_{tot}^{(j)}$ for all $j \in \{1, \ldots, J\}$. The problem is to design a joint routing and power allocation strategy that maximizes throughput and stabilizes the system whenever the system is stabilizable. Such a policy makes decisions using the observable system state vectors $\vec{U}(t)$ and $\vec{S}(t)$.

In general, both state vectors $\vec{U}(t)$ and $\vec{S}(t)$ are important in both the routing and power allocation decisions. For example, clearly any power allocated to an empty queue is wasted and should be re-allocated to improve processing rates amongst the non-empty queues. Likewise, a router is inclined to place packets in faster queues, especially if the rates of those queues are guaranteed to operate at high levels for one or more timeslots. However, below we show that the routing and power allocation problem can be decoupled into two policies: a routing policy which considers only $\vec{U}(t)$, and a power allocation policy which considers only $\vec{S}(t)$. The power allocation policy is distributed, so that each sub-unit makes independent control decisions using only the local channel state information for each

queue it contains. The resulting strategy maximizes total system throughput even when the underlying Markov chain describing $\vec{S}(t)$ is unknown.

Let $\beta_{\vec{S}}$ represent the steady-state probability that the channel vector is in state $\vec{S}$. Define the following rate $\overline{\mu}$:

$$\overline{\mu} = \sum_{\vec{S}} \beta_{\vec{S}} \left[\max_{\sum_{i \in V_j} P_i = P_{tot}^{(j)} \, \forall j} \sum_i \mu_i(P_i, S_i) \right] \tag{B.20}$$

The value of $\overline{\mu}$ is the average total rate offered by the system when power is allocated to maximize total rate at every instant of time. Assume that the input process $X(t)$ generates packets according to a fine state, ergodic Markov chain, and let λ represent the total bit rate.

Theorem 12. *The capacity of the multi-queue system with joint routing and power allocation is $\overline{\mu}$, i.e., a necessary condition for stability is $\lambda \leq \overline{\mu}$, and a sufficient condition is $\lambda < \overline{\mu}$.*

The fact that $\lambda \leq \overline{\mu}$ is necessary follows by application of Theorems 28 and 29. Indeed, suppose a stabilizing algorithm exists, and let $\{p_i(t)\}$ represent the stabilizing power functions. (Note that these functions are not necessarily ergodic). The sum rate of all servers in the multi-queue system is hence $\mu(t) = \mu_1(P_1(t), S_1(t)) + \ldots + \mu_N(P_N(t), S_N(t))$. By Theorem 28, the system is stable only if a single queue with the same inputs and server rate $\mu(t)$ is stable. But $\mu(t) \leq \mu^*(t)$ for all t, where $\mu^*(t)$ is the result when power is allocated to maximize instantaneous sum rate. Thus, a system with input $X(t)$ and server process $\mu^*(t)$ is also stable. But $X(t)$ is ergodic with rate λ and $\mu^*(t)$ is ergodic with rate $\overline{\mu}$, so $\lambda \leq \overline{\mu}$ must hold.

Sufficiency is established by design of the following decoupled policy π^* which stabilizes the system whenever $\lambda < \overline{\mu}$:

Power Allocation: At every new timeslot, each sub-unit j observes the entries of the channel state vector $\vec{S}(t)$ corresponding to the queues it contains (given by the set V_j). The sub-unit then allocates power $\{P_i(t)\}$ (for $i \in V_j$) to maximize $\sum_{i \in V_j} \mu_i(P_i, S_i(t))$ subject to $\sum_{i \in V_j} P_i = P_{tot}^{(j)}$. This power allocation is held constant until the next timeslot.

Routing: Whenever a new packet enters the system, we observe the value of $\vec{U}(t) = (U_1(t), \ldots, U_N(t))$ and route to the shortest queue.

Note that this strategy is simply an application of *JSQ* routing in reaction to the rates determined by the power allocation decisions. Thus, from Theorem 29 we know that the multi-queue system is stable whenever the single queue system is stable, which is ensured when $\lambda < \overline{\mu}$. This establishes Theorem 30.

B.6 Summary

The problem of routing packets from an arbitrary stream over a set of parallel queues has been considered in the context of constant and time-varying processing rates. Using sample path analysis, a simple work conserving strategy was developed to provide fundamental performance bounds on the unfinished work in the system at any instant of time. In time varying systems, this strategy can be implemented with a pre-queue and guarantees that performance closely follows the performance of a superior single-queue system with an aggregated data rate.

The pre-queue was shown to be unnecessary when service rates are constant. In the case of time-varying rates, removing the pre-queue precludes the design of a routing strategy which meets the tight performance bounds on unfinished work guaranteed by a work conserving policy. However, a general stability result was established for the *Join-the-Shortest-Queue* policy, and the result was extended to treat a joint problem of routing and power allocation. This analysis was performed using a new and useful notion of stability defined in terms of finite buffer systems. Performance bounds for the *JSQ* strategy were given by showing that if all queues have buffer size $M + NL_{max}$, the drop rate is less than or equal to the drop rate of a single queue with an aggregate rate and buffer size M.

Our approach differs significantly from other approaches in that we provide tight, worst case bounds on system performance with arbitrary input and linespeed processes, rather than analyzing systems with particular stochastic inputs and linespeeds. We believe this approach can be applied to more complex queueing structures in satellite and wireless networks to provide performance bounds and stability guarantees for systems with very general input processes and control laws.

Appendix B.A — Proof of Finite Buffer Multiplexing Inequality

Here we prove the finite buffer multiplexing inequality (Theorem 28), which compares a single queue with a time varying server rate $\mu(t)$ and a buffer of size M to a parallel system of N queues with server rates $\mu_1(t), \mu_2(t), \ldots, \mu_N(t)$ with buffer sizes $M_1, M_2, \ldots, M_N$.

Theorem: If $\sum_i \mu_i(t) = \mu(t)$ and if $L_{max} + \sum_i M_i \leq M$, then for an arbitrary input stream $X(t)$:

(a) Departures satisfy: $D_{single}(t) \geq D_{multi}(t)$ for all t

(b) Packet drops satisfy: $G_{single}(t) \leq G_{multi}(t)$ for all t

Proof. The single-queue and multi-queue systems satisfy the following bit conservation equalities for all time:

$$U_{single}(t) = X(t) - D_{single}(t) - G_{single}(t) \tag{B.21}$$

$$U_{multi}(t) = X(t) - D_{multi}(t) - G_{multi}(t) \tag{B.22}$$

Claim 1: If $D_{single}(t) \geq D_{multi}(t)$ for all $t \in [0, t^*]$ for some time t^*, then $G_{single}(t) \leq G_{multi}(t)$ on the same interval.

Pf: It suffices to check times t when the single-server system loses a packet, and the multi-server system retains that same packet (otherwise, $G_{single}(t) - G_{multi}(t)$ cannot increase). At such times, $U_{single}(t) > M - L_{max}$, and $U_{multi} \leq M_1 + M_2 + \ldots + M_N \leq M - L_{max}$. Furthermore, we have:

$$\begin{aligned} G_{single}(t) &= X(t) - D_{single}(t) - U_{single}(t) \\ &\leq X(t) - D_{multi}(t) - U_{single}(t) \\ &< X(t) - D_{multi}(t) - (M - L_{max}) \\ &= U_{multi}(t) + G_{multi}(t) - (M - L_{max}) \\ &\leq G_{multi}(t) \end{aligned}$$

which proves the claim. □

Claim 2: $D_{single}(t) \geq D_{multi}(t)$ for all time $t \geq 0$.

Pf: For simplicity, assume $\mu(t) < \infty$ so that packets drain continuously from the queues.

The departure inequality is true at time 0. If it is ever violated, there must be some first crossing time t^* where $D_{multi}(t^*) - D_{single}(t^*)$. At such a time, from (B.21) and (B.22) we have:

$$U_{multi}(t^*) + G_{multi}(t^*) = U_{single}(t^*) + G_{single}(t^*)$$

However, from Claim 1, we know $G_{single}(t^*) \leq G_{multi}(t^*)$, and hence $U_{multi}(t^*) \leq U_{single}(t^*)$. Thus, if the single-server system is empty at time t^*, the multi-server system is also empty, no bits are being processed, and the departure function cannot overtake the bound. Otherwise, the single-server system is busy and departures are draining from it at the fastest possible rate $\mu(t^*)$—so again the departure function for the multi-server system cannot cross the $D_{single}(t^*)$ bound. □

Appendix C

The Jitter Theorem

Consider a queue in continuous time with an input process $A(t)$ and a time varying server process $\mu(t)$ with an average rate $\mu_{av} = \mathbb{E}\{\mu(t)\}$. In this appendix, we show that if the server process is stationary and independent of the arrival process, then any moment of unfinished work in the queue is lower bounded by the corresponding moment if the time varying server is replaced by a constant rate server of rate μ_{av}.

Suppose the queue is empty at time zero, when the input stream $A(t)$ is applied. Note that the unfinished work $U(t)$ in the queue at any instant of time is given by

$$U(t) = \max_{\tau \geq 0} \left[A(t) - A(t-\tau) - \int_{t-\tau}^{t} \mu(v)dv \right] \tag{C.1}$$

The above equation is easily verified by noting that the unfinished work at any time instant t is at least as large as the total arrivals minus the total offered service over any interval ending at time t, and the bound is met with equality over the interval $[t_b, t]$, where t_b is the start of the current busy period.

Define $\hat{U}(t)$ as the unfinished work in the system with the same arrival process but with a constant server rate of μ_{av}. An expression for $\hat{U}(t)$ in terms of the input $A()$ is given by using $\mu(v) = \mu_{av}$ in the above equality (C.1).

Theorem 13. *(Jitter Theorem) For any convex, non-decreasing function $f(u)$, we have at every time instant t:*

$$\mathbb{E}f\left(U(t)\right) \geq \mathbb{E}f\left(\hat{U}(t)\right)$$

The following proof uses convexity of the max[] operator in a manner similar to the

technique used in [64] to show that fixed length packets minimize delay over all packet input sequences with a given mean packet length.

Proof. For any convex increasing function $f(u)$, we have:

$$\begin{aligned}
\mathbb{E}f\left(U(t)\right) &= \mathbb{E}f\left(\max_{\tau\geq 0}\left[A(t)-A(t-\tau)-\int_{t-\tau}^{t}\mu(v)dv\right]\right) && \text{(C.2)}\\
&= \mathbb{E}_{A()}\mathbb{E}\left\{f\left(\max_{\tau\geq 0}\left[A(t)-A(t-\tau)-\int_{t-\tau}^{t}\mu(v)dv\right]\right)|A()\right\} && \text{(C.3)}\\
&\geq \mathbb{E}_{A()}f\left(\max_{\tau\geq 0}\left[\mathbb{E}\left\{A(t)-A(t-\tau)-\int_{t-\tau}^{t}\mu(v)dv\,|A()\right\}\right]\right) && \text{(C.4)}\\
&= \mathbb{E}_{A()}f\left(\max_{\tau\geq 0}\left[A(t)-A(t-\tau)-\int_{t-\tau}^{t}\mathbb{E}\left\{\mu(v)\,|A()\right\}dv\right]\right) && \text{(C.5)}\\
&= \mathbb{E}_{A()}f\left(\max_{\tau\geq 0}\left[A(t)-A(t-\tau)-\int_{t-\tau}^{t}\mu_{av}dv\right]\right) && \text{(C.6)}
\end{aligned}$$

where (C.3) follows because we have broken the original expectation into an iterated expectation, (C.4) holds by Jensen's inequality together with the fact that the $f(\max[])$ operator is convex,[1] and (C.6) holds because the server process is stationary and independent of the arrival process. The final expression is by definition equal to $\mathbb{E}\left\{f(\hat{U}(t))\right\}$, proving the theorem. □

Thus, any time varying jitter in the linespeed process creates extra queue congestion. We indirectly apply this result in [104] to show any $N \times N$ packet switch scheduler that does not consider queue backlog has an average delay of at least $O(N)$.

C.1 Upper Bound Conjecture

Define $\Phi_X(\mu)$ as the steady state expectation of unfinished work in a queue with an input process $X(t)$ and a constant server rate of μ, and let $\mu(t)$ represent a time varying server process that is stationary and independent of $X(t)$. Let $p(\mu)$ represent the steady state distribution of the $\mu(t)$ process, and let μ_{av} be the average rate. Let $\overline{U}$ represent the average unfinished work in a queue with input $X(t)$ and server process $\mu(t)$.

[1] Indeed, using the non-decreasing convex property of the $f(u)$ function, it is not difficult to show that $f\left(\max_{\tau\geq 0}\left[p_1 g_1(\tau)+p_2 g_2(\tau)\right]\right) \leq p_1 f\left(\max_{\tau\geq 0}\left[g_1(\tau)\right]\right)+p_2 f\left(\max_{\tau\geq 0}\left[g_2(\tau)\right]\right)$ for any functions $g_1()$, $g_2()$, and for any probabilities p_1, p_2 such that $p_1+p_2=1$.

Conjecture: $\Phi_X(\mu_{av}) \leq \overline{U} \leq \mathbb{E}_{p(\mu)}\{\Phi_X(\mu)\}$

The lower bound follows as a special case of the Jitter Theorem, so the conjecture only concerns the upper bound. Thus, we conjecture that the average unfinished work in a queue with a time varying server process that is stationary and independent of the inputs is less than or equal to the expectation of $\Phi_X(\mu)$, where μ is treated as a random variable with a distribution equal to the steady state distribution of the $\mu(t)$ process.

To give intuition about why we expect the upper bound to hold, we note that it trivially holds in the case when the $\mu(t)$ process has a positive steady state probability of being strictly larger than λ, as $\mathbb{E}_{p(\mu)}\{\Phi_X(\mu)\} = \infty$ in this case. Further note that both the upper and lower bounds are tight when the speed of server variation is decreased to zero or increased to infinity, respectively, while the same steady state distribution is maintained (i.e., for a fixed $\mu(t)$ process, we can consider $\mu(Vt)$ where $V \to 0$ or $V \to \infty$). Indeed, in the case when server variations are very rapid, the average server rate converges to μ_{av} over the course of just a single packet transfer, so that effective service rates are constant and $\overline{U} \to \Phi_X(\mu_{av})$. Alternately, when variations are very slow, the queue reaches steady state behavior for each different channel state, and hence $\overline{U} \to \mathbb{E}_{p(\mu)}\{\Phi_X(\mu)\}$.

C.2 Alternate Proof of Jitter Theorem

Here we develop an alternate and more intuitive proof of the Jitter Theorem that is based on the multiplexing inequality for queueing systems (Lemma 25 of Appendix B). Consider an input stream $X(t)$ (representing the amount of bits that arrive during the interval $[0, t]$), and a time varying server process $\mu(t)$, representing the instantaneous server rate in units of bits/second. Let $U(t)$ represent the unfinished work in the queue as a function of time (assuming the queue is initially empty). Further let $U_{mutli-server}(t)$ represent the sum unfinished work in a multi-server system with rates $\mu_1(t), \mu_2(t), \ldots, \mu_N(t)$ such that $\sum_i \mu_i(t) = \mu(t)$. We make use of the following basic properties of all queueing systems:

- *Multiplexing Inequality:* $U(t) \leq U_{multi-server}(t)$ for all time $t \geq 0$.
- *Unit Scaling Equality:* For any constant $V > 0$, $VU(t) = \tilde{U}(t)$ for all time $t \geq 0$, where $\tilde{U}(t)$ represents the unfinished work process in a single queue system with an

input stream $VX(t)$ and a server rate $V\mu(t)$. That is, $\tilde{U}(t)$ is the resulting unfinished work when both the input and server rate are scaled by V.

The multiplexing inequality is proved in Appendix B (Lemma 25). The unit scaling property holds because scaling both the input and server process by a constant V scales the $U(t)$ sample path by V at every instant of time.[2]

The Jitter Theorem follows directly from these two properties together with the picture shown in Fig. C-1. For simplicity of exposition, we illustrate only the fact that $\mathbb{E}\{U(t)\} \geq \mathbb{E}\left\{\hat{U}(t)\right\}$, proving Theorem 31 for the special case where $f(u) = u$.

From the figure, we note by the unit scaling property that the unfinished work $U(t)$ at every instant of time is equal to the sum unfinished work when the system is duplicated M times, with scaled inputs and server rates $\frac{1}{M}X(t)$ and $\frac{1}{M}\mu(t)$. Next, note that the expected unfinished work in each duplicate system m is equal to the expected unfinished work in a modified system m' with the same input process $\frac{1}{M}X(t)$ but with the server process replaced by $\frac{1}{M}\mu_m(t)$, where $\mu_m(t)$ is an independent but identically distributed version of the original $\mu(t)$ process. This holds because the original $\mu(t)$ process is independent of the input stream $X(t)$. However, applying the multiplexing inequality, we find that the sum unfinished work in all modified queues is greater than or equal to the unfinished work in a single queue with an input stream $X(t) = \frac{1}{M}\sum_{m=1}^{M} X(t)$ and a server process $\frac{1}{M}\sum_{m=1}^{M}\mu_m(t)$. Thus, the expected unfinished work in the original system is greater than or equal to the expected unfinished work in a new system with the original input stream but with a server process consisting of a sum of M i.i.d. processes $\mu_m(t)$. This holds for all positive integers M, and hence we can take limits as $M \to \infty$. Because all processes are stationary and independent, it follows by the law of large numbers (applied to processes) that $\frac{1}{M}\sum_{m=1}^{M}\mu_m(t) \to \mu_{av}$.

[2]The operation of scaling both $X(t)$ and $\mu(t)$ by the same constant $V > 0$ can be viewed simply as expressing the backlog in units other than bits.

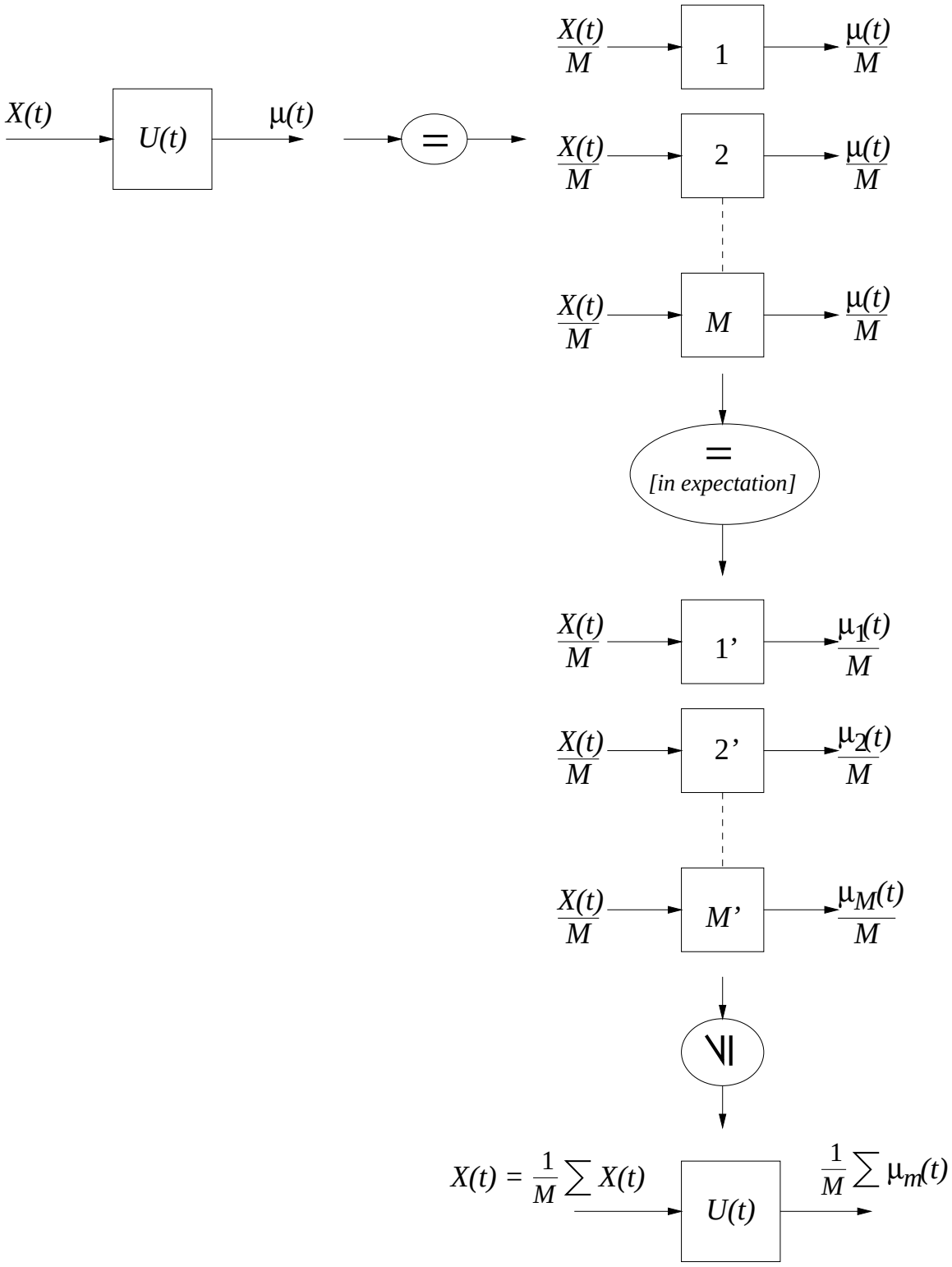

Figure C-1: An illustration proving the Jitter Theorem.