Routing over Parallel Queues with Time Varying Channels with Application to Satellite and Wireless Networks

Michael J. Neely Eytan Modiano Charles E. Rohrs
MIT--LIDS: {mjneely@mit.edu, modiano@mit.edu, crohrs@mit.edu}

***Abstract* -- 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 $(\mu_1,..., \mu_n)$ 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. For the case when routing decisions must be made immediately upon arrival, we demonstrate that the non-predictive policy of routing packets to the shortest queue ensures system stability (and maintains low packet loss rates for finite buffer systems) whenever the system is stabilizable. Finally, we consider a joint problem of routing and power allocation where, for each time varying channel state $c_i(t)$, 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 $\mu_i(c_i, p_i)$. A throughput maximizing algorithm is developed for this joint problem.**

I. Introduction

This paper treats a problem of routing packets from an arbitrary input stream $X(t)$ over a set of n heterogeneous queues in parallel (Fig. 1). Routing decisions are made based upon the current state of the queues, and the goal is to maintain an acceptably low level of backlog for all time. Such a scenario occurs, for example, in a satellite network where packets arrive to a satellite and can reach the ground using one of several downlink channels. Packet transmission rates along different downlinks may vary as different channel codes are used to adjust to fluctuations in channel conditions (e.g., due to weather). Timescales of variation may be very long--in which case a constant transmission rate model is appropriate--or very short, in which case a time-varying channel model is used. Here, both the cases of constant and time-varying rates are considered from a queueing perspective. The instantaneous transmission rates are assumed to be known at the transmitter, although in the time varying case future rates are unknown and change according to arbitrary ergodic processes.

For constant processing rates, we develop a simple work conserving routing policy π_{WC} that ensures no more than a fixed amount of excess bits in the system above the amount there would be under any other routing strategy. Similar results hold when processing rates are time-varying and routing decisions can be postponed by storing packets in a pre-queue.

When no such pre-queue is available and packets must be routed immediately upon their 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 show that the simple, non-predictive strategy of placing an arriving packet in the queue with the least backlogged bits yields a stable system whenever the system is stabilizable. A new notion of stability is developed for this analysis, which is useful for addressing stability issues in general ergodic systems. The analysis yields simple performance bounds on queue occupancy and packet drop rates.

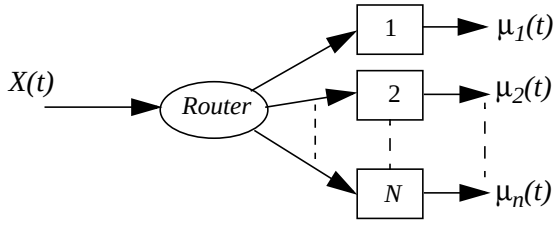

Figure 1: Routing over a set of parallel queues with time-varying processing speeds $\mu_i(t)$.

We extend these results to address a joint problem of routing and power allocation. A constrained resource of power is allocated to the multiple transmitters to control transmission rates in response to packet backlog and/or varying channel conditions. A throughput maximizing algorithm with performance bounds is developed.

Previous work on routing and queue control policies in satellite and wireless systems is found in [1-9]. In [1], stabilizing power control algorithms are developed for a multi-beam satellite downlink operating in discrete time. In [2] a server scheduling problem is formulated where a single server can be scheduled to serve one of n parallel queues, each queue having iid packet arrivals and iid channel outage states (ON or OFF) on each timeslot. In symmetric situations when all queues have the same arrival and outage processes, stochastic coupling is used to show optimality of the "serve-the-longest-connected-queue" policy. The problem of routing an arbitrary stream over two queues with homogeneous and exponential service rates is treated in [3], and the join-the-shortest-queue strategy is shown to minimize average delay. Routing over two heterogeneous queues is considered in [4], and an optimal threshold policy for minimizing average occupancy is developed using dynamic programming techniques. In [6] stable routing policies are developed for a network of queues using Lyapunov drift techniques. In [7] an exact analysis of the join-the-shortest-queue policy for parallel queues is developed for M/M/1 systems. In [9] convexity properties of queueing systems with general */* inputs are developed and used to establish optimal static policies for routing multiple data streams over n parallel queues.

A related NP-complete problem of batch packet arrivals is considered in [10, 11] where the goal is to route packets to parallel queues to minimize the total delay for transmitting one batch. In [12-14] 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 $t>0$. 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 and Lyapunov function techniques which address systems with particular types of input 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 III 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 IV and V we treat routing in systems with a pre-queue and without a pre-queue, respectively, and in Section VI we apply these results to a power allocation problem.

II. The Greedy and Work Conserving Algorithms

Consider the system of Fig. 1 with an arbitrary arrival stream $X(t)$ sending packets to be routed over the n queues. Assume all processing rates $(\mu_1,..., \mu_n)$ 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 each 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 $k = \underset{j \in \{1, ..., n\}}{argmin}\left\{\frac{L_i + U_j(t)}{\mu_j}\right\}$

(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 $k = \underset{j \in \{1, ..., n\}}{argmin}\left\{\frac{U_j(t)}{\mu_j}\right\}$.

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 $(\mu_1,..., \mu_n)$, 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 $\{1, 2, 3,...\}$. Assume all packets have fixed lengths $L=1$, and that $(\mu_1, \mu_2,..., \mu_n) = (1, \varepsilon,..., \varepsilon)$. Suppose that $\varepsilon>0$ 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 $\{2, 3,..., n\}$, 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 III 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.

III. 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. 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. 2a is equal to the sum of the individual server rates in Fig. 2b, i.e., $\mu(t) = \mu_1(t) + \ldots + \mu_n(t)$. Assume both systems of Fig. 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 $U_{single\text{-}server}(t)$ represent the unfinished work (in bits) in the single server queue, and let $U_{multi\text{-}server}(t)$ represent the total amount of unfinished bits in the multi-queue system.

Lemma 1 (Multiplexing Inequality): For all time $t \geq 0$:

$$U_{single\text{-}server}(t) \leq U_{multi\text{-}server}(t). \quad (1)$$

Proof: Let $D_{single\text{-}server}(t)$ and $D_{multi\text{-}server}(t)$ represent the amount of processed bits or "departures" from the single server system and multi-server system, respectively, during $[0, t]$. Clearly we have $D_{single\text{-}server}(t) = X(t) - U_{single\text{-}server}(t)$ and

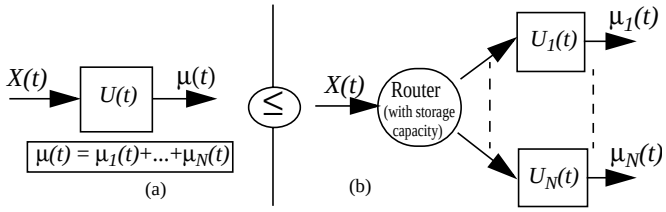

Figure 2: A single server queue and a set of multi-server queues with an arbitrary router device. The sum of the time varying server rates of the servers in (b) equals the rate $\mu(t)$ in (a).

$D_{multi\text{-}server}(t) = X(t) - U_{multi\text{-}server}(t)$, and it suffices to show that $D_{single\text{-}server}(t) \geq D_{multi\text{-}server}(t)$. Notice that the departure functions are both initially 0 at time t=0. Whenever the single-server system is empty, $D_{single\text{-}server}(t) = X(t) \geq D_{multi\text{-}server}(t)$, 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 system can never overtake the departures from the single-server queue. ❑

A finite buffer version of this statement is given in Section V. 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 $U_{single\text{-}server}(t)$ lower bound are desirable.

Consider now a particular work conserving routing strategy π_{WC} that operates on the multi-queue system of Fig. 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. 2b under this policy.

Lemma 2 (Performance Tracking): At every instant of time t:

$$U_{single\_server}(t) \leq U_{WC}(t) \leq U_{single\_server}(t) + (n\text{-}1)L_{max}. \quad (2)$$

Proof: The first inequality is just a particular case of the multiplexing inequality (Lemma 1). 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 $\mu_i < \infty$ for 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 $D_{single\_server}(t^*) = D_{WC}(t^*) + (n\text{-}1)L_{max}$. 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 $(n\text{-}1)L_{max}$ 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 2 imply that the work conserving routing strategy π_{WC} is stable if and only if the single queue system with the same inputs is stable. Stability issues are addressed further in Section V.

IV. 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. 2b under policies π_{WC} and some other (arbitrary) policy π, respectively.

A. Variable Packet Lengths: 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 $min_{X,\hat{\mu}}(\tau)$ represent this minimum value of unfinished work in a system with input stream $X(t)$ and linespeeds $(\mu_i(t))$.

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 $\mu_1(t)$,..., $\mu_N(t)$ in order to maximize the difference between $U_{\pi}(t)$ and $min_{X,\hat{\mu}}(\tau)$. The goal of the scheduler is to minimize the worst case deviation from optimality, i.e., minimize the value of $\max_{t \geq 0}\{U_{\pi}(t) - min_{X,\hat{\mu}}(t)\}$.

Theorem 1: (a) For all policies π (possibly predictive and preemptive), we have: $U_{WC}(t) \leq U_{\pi}(t) + (n-1)L_{max}$. (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 1 and 2. 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 $(n\text{-}1)L_{max}$ 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 $(n\text{-}1)$ maximum length packets at time 0. She 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, 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, an alternative routing scheme that sends all packets to this server would have allowed the system to be empty at this time, and so the $(n-1)L_{max}$ bound is met with equality. ❑

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

Theorem 2: If all packets have fixed lengths L, then:

$$N_{WC}(t) \le N_{\pi}(t) + n - 1. \quad (4)$$

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:

$$N_{WC}(t) = N_{WC}(\tau^-_B) + A(\tau_B, t) - D_{WC}(\tau_B, t) \quad (5)$$

$$N_{\pi}(t) = N_{\pi}(\tau^-_B) + A(\tau_B, t) - D_{\pi}(\tau_B, t) \quad (6)$$

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^i_{WC} and D^i_{π} representing departures from queue i in the π_{WC} and π systems:

$$D_{WC}(\tau_B, t) = \sum D^i_{WC} \quad , \quad D_{\pi}(\tau_B, t) = \sum D^i_{\pi}. \quad (7)$$

During the time interval $[\tau_B, t]$, each queue of the π_{WC} system continuously processes fixed length packets. Thus, $D^i_{WC} \ge D^i_{\pi} - 1$ for al0l 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 (where $k \le n$). We thus have:

$$N_{WC}(t) \le N_{WC}(\tau^-_B) + A(\tau_B, t) - D_{\pi}(\tau_B, t) + k \quad (8)$$

$$= N_{WC}(\tau^-_B) + N_{\pi}(t) - N_{\pi}(\tau^-_B) + k \quad (9)$$

$$\le (n-1) + N_{\pi}(t) - k + k \quad (10)$$

where (9) follows from (6), and (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 1 shows that this $(n-1)$ bound is tight and is minimax optimal over all non-predictive, non-preemptive strategies.

Similar routing problems have been treated in [3,4,5,16] for systems with two heterogeneous servers (n=2) when packet inputs are Poisson. With this stochastic formulation, 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 calculate 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.

V. Systems Without A Pre-Queue

A. 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 $(\mu_1,...,\mu_n)$ are known constants.

Observation: For constant server rates $(\mu_1,...,\mu_n)$, 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 II) is the same as the π_{WC} strategy described for time varying servers in Section III. 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 1 and 2 apply to heterogeneous, constant rate servers when this routing method is used.

B. 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. 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 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:

$$DR(M) = \lim_{t \to \infty} sup[G^M(t)/t].$$

Definition: 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 $M \to \infty$.

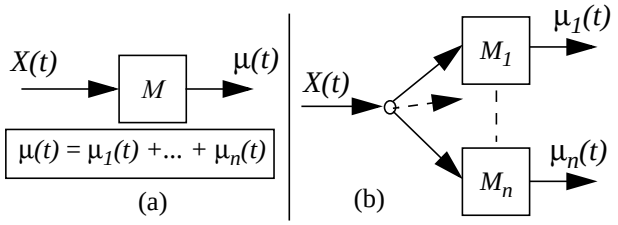

Figure 3: A single stage system with a finite buffer of size M and an aggregated server processing rate $\mu(t)$ compared to a multi-queue system with finite buffers.

It can be shown that a necessary condition for loss rate stability is $\lambda \le \mu_{av}$. Furthermore, if the input stream and server rate process evolve according to an underlying finite state 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 [6], [1].

Before analyzing the JSQ strategy, we present a finite buffer version of the multiplexing inequality. Consider the two systems of Fig. 3. The server rates $\mu_1(t),...,\mu_n(t)$ 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 $M_1,...,M_n$. Let $G_{single}(t)$ represent the total bits dropped by the single server system during $[0, t]$, and let $G_{multi}(t)$ 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 3 (Finite Buffer Multiplexing Inequality): For arbitrary inputs $X(t)$, if $M \ge M_1 + \ldots + M_n + L_{max}$, then for all t:

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

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

Proof: See Appendix. ❑

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 4: For all buffer sizes M:

$$DR_{JSQ}(M + nL_{max}) \le DR_{single\text{-}queue}(M) \qquad (11)$$

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: $G_{JSQ}(t) \le G_{single}(t)$ for all $t \ge 0$, 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:

a = Arrivals during $[t_B, \tau]$

d_{JSQ} = Bits processed by the JSQ system during $[t_B, \tau]$

g_{JSQ} = Bits dropped by the JSQ system during $[t_B, \tau]$

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 $G_{JSQ}(t) \le G_{single}(t)$ is first violated at time τ. The following bit-conservation equalities hold:

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

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

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

$$U_{JSQ}(t_B^-) \le (n-1)[M + nL_{max}]. \qquad (14)$$

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:

$$U_{JSQ}(\tau) > n[M + (n-1)L_{max}]. \qquad (15)$$

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

$$a - d_{JSQ} > M + g_{JSQ}. \qquad (16)$$

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

$$U_{single}(\tau) \ge a - d_{single} - g_{single} \qquad (17)$$

$$\ge a - d_{JSQ} - g_{single} \qquad (18)$$

$$> M + g_{JSQ} - g_{single} \qquad (19)$$

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

Theorems 3 and 4 imply that the multi-queue system under the JSQ routing strategy is stable if and only if the corresponding single queue system is stable. Furthermore, (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.

VI. Joint Routing And Power Allocation

Suppose that the transmission rates $(\mu_1,...,\mu_n)$ 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 c_i which takes values on a finite set of states ($c_i \in \{S^i_1, S^i_2, \ldots, S^i_{Mi}\}$). Let $\mu_i(p_i, c_i)$ represent a concave rate-power curve for each channel state (see Fig. 4). We assume that the individual queues belong to a collection of J sub-units, and let V_j represent the set of queues $i \in \{1, \ldots, n\}$

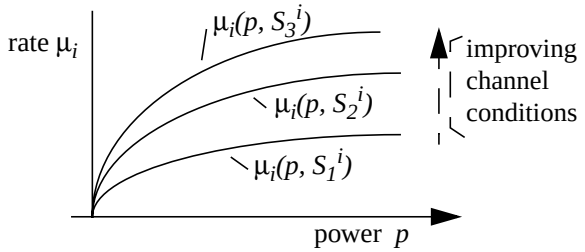

Figure 4: A set of concave power curves $\mu_i(p_i, c_i)$ for channel states $c_i = S_1^i, S_2^i, S_3^i$.

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 $P_{tot}^{(j)}$.

Let $\vec{C}(t) = (c_1(t), c_2(t), \ldots, c_n(t))$ represent the vector of channel states at time *t*, and assume that $\vec{C}(t)$ 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 $\vec{C}(t)$.

In general, both state vectors $\vec{U}(t)$ and $\vec{C}(t)$ 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 $\vec{C}(t)$. 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 $\vec{C}(t)$ is unknown.

Let $\xi_{\vec{C}}$ represent the steady-state probability that the channel vector is in state $\vec{C}$. Define the following rate $\bar{\mu}$:

$$\bar{\mu} = \sum_{\vec{C}} \xi_{\vec{C}} \left[\max_{\sum_{i \in V_j} p_i = P_{tot}^{(j)} \ \forall j} \sum \mu_i(p_i, c_i) \right]. \quad (20)$$

The value of $\bar{\mu}$ 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 5: The capacity of the multi-queue system with joint routing and power allocation is $\bar{\mu}$, i.e., a necessary condition for stability is $\lambda \leq \bar{\mu}$, and a sufficient condition is $\lambda < \bar{\mu}$.

The fact that $\lambda \leq \bar{\mu}$ is necessary follows by application of Theorems 3 and 4. 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 $\mu(t) = \mu_1(p_1(t), c_1(t)) + \ldots + \mu_n(p_n(t), c_n(t))$. The system is stable if and only if a single queue system 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 total instantaneous 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 $\bar{\mu}$, so $\lambda \leq \bar{\mu}$ must hold.

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

Power Allocation: At every new timeslot, each sub-unit *j* observes the entries of the channel state vector $\vec{C}(t)$ 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 $\sum_{i \in V_j} \mu_i(p_i, c_i(t))$ 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 4 we know that the multi-queue system is stable whenever the single queue system is stable, which is ensured when $\lambda < \bar{\mu}$. This establishes Theorem 5.

VII. Conclusions

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:

Here we prove the finite buffer multiplexing inequality.

Theorem 3: For arbitrary inputs $X(t)$, if:

$M \geq M_1 + \ldots + M_n + L_{max}$, then for all time t:

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

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

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

$$U_{single}(t) = X(t) - D_{single}(t) - G_{single}(t) \quad (21)$$

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

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 $U_{multi}(t) \leq M_1 + M_2 + \ldots + M_n \leq M - L_{max}$. Furthermore, we have:

$$G_{single}(t) = X(t) - D_{single}(t) - U_{single}(t) \quad (23)$$

$$\leq X(t) - D_{multi}(t) - U_{single}(t) \quad (24)$$

$$< X(t) - D_{multi}(t) - (M - L_{max}) \quad (25)$$

$$= U_{multi}(t) + G_{multi}(t) - (M - L_{max}) \quad (26)$$

$$\leq G_{multi}(t). \quad (27)$$

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 (21) and (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. ❑

REFERENCES:

[1] M.J.Neely, E. Modiano, and C. Rohrs, "Power and Server Allocation in a Multi-Beam Satellite with Time Varying Channels," *IEEE Proceedings of INFOCOM* 2002.

[2] L. Tassiulas and A. Ephremides, "Dynamic Server Allocation to Parallel Queues with Randomly Varying Connectivity," *IEEE Trans. on Information Theory*, vol.39, no.2, March 1993.

[3] A. Ephremides, P. Varaiya, and J. Walrand, "A Simple Dynamic Routing Problem," *IEEE Trans. on Automatic Control*, vol.25, no.4, Aug. 1980.

[4] I. Viniotis and A. Ephremides, "Extension of the Optimality of the Threshold Policy in Heterogeneous Multiserver Queueing Systems," *IEEE Trans. on Automatic Control*, vol.33, no.1, Jan. 1988.

[5] Bruce Hajek, "Optimal Control of Two Interacting Service Stations," *IEEE Trans. on Automatic Control*, vol.29, no.6, June 1984.

[6] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, Dec. 1992.

[7] P. Wu and M.J.M. Posner, "A Level-Crossing Approach to the Solution of the Shortest-Queue Problem," *Operations Research Letters* 21, 1997.

[8] M. Carr and Bruce Hajek, "Scheduling with Asynchronous Service Opportunities with Applications to Multiple Satellite Systems," *IEEE Trans. on Automatic Control*, vol.38, no.12, Dec. 1993.

[9] M.J.Neely and E. Modiano, "Convexity and Optimal Load Distributions in Work Conserving */*/1 Queues," *IEEE Infocom Proceedings*, 2001.

[10] R.L.Graham, "Bounds on Multiprocessing Timing Anomalies," *SIAM J. Appl. Math*, vol.17,no.2, March 1969.

[11] E.G. Coffman, M.R. Garey, and D.S.Johnson, "An Application of Bin-Packing to Multi-Processor Scheduling," *SIAM J. of Comput.*, vol.7, no.1, Feb. 1978.

[12] D. B. Shmoys, J. Wein, and D.P. Williamson, "Scheduling Parallel Machines On-line," *IEEE 32nd Ann. Symp. on Foundations of Comp Sci*, 1991.

[13] Y.Bartal, A.Fiat, H.Karloff, and R. Vohra, "New Algorithms for an Ancient Scheduling Problem," *Proc. of the 24th Annual ACM Symp. on the Theory of Computing*, 1992.

[14] D.R. Karger, S.J.Phillips, E.Torng, "A Better Algorithm for an Ancient Scheduling Problem," *Proc. of the 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, 1994.

[15] Tom Leighton, "Methods for Message Routing in Parallel Machines," *Proc. of the 24th Annual ACM Symp. on the Theory of Computing*, 1992.

[16] Jean Walrand. *An Introduction to Queueing Networks*. Prentice-Hall, Englewood Cliffs: NJ, 1988.