I. MULTISERVER SYSTEMS AND PERFORMANCE TRACKING

Here we treat multi-server systems that share a common buffer. We begin with the following simple example.

A. Example

Compare a single server, work conserving queue with constant server rate $\mu = 2$ to a system of two parallel servers with individual service rates $\mu = 1$. Consider an input stream consisting of exactly two packets, both of length equal to 1 unit: Packet $A$ arrives at time $t = 1$, and packet $B$ arrives at time $t = 1.5$. The packets are served in FIFO order in the single server system. The packets are also served in FIFO order in the 2-server system: Packet $A$ begins its service in either of the rate-1 servers immediately upon arrival, and packet $B$ begins its service immediately upon its arrival by entering the alternative rate-1 server (note that the first server is busy with packet $A$ at this time).

![Figure 1](image1.png)

Fig. 1. An example of the unfinished work functions associated with two arriving packets entering a multi-server system versus entering a single-server system, illustrating the multiplexing inequality.

The above figure illustrates the corresponding unfinished work functions $U_{\text{single}}(t)$ and $U_{\text{multi}}(t)$. In this example we observe that $U_{\text{single}}(t) \leq U_{\text{multi}}(t)$ for all time $t$. This can be considered as a special case of the multiplexing inequality by treating the multi-queue input functions $X_1(t)$ and $X_2(t)$ as the streams consisting of the packet entering the first server and the second server, respectively. It is clear from the above example that the multi-server system suffers from inefficiencies when it is not fully loaded, that is, when there are some servers that sit idle. In this lecture we describe the worst case backlog increase incurred by these inefficiencies.

B. Multi-Server, Single Buffer Queues

A multi-server, single buffer queue is a queueing system with a single shared buffer for storing incoming packets, together with a set of servers that process these packets (see Figure below). A packet waiting in the queue can be processed by any one of the servers. This system is conceptually similar to a system of parallel queues, with the exception that parallel queues often have separate storage buffers. Indeed, without a shared buffer, it is difficult or impossible for a packet currently contained in one queue to switch to the buffer space of another queue that it prefers. Shared buffering allows such freedom.

**Definition 1:** A multi-server, single buffer queueing system is work conserving if it never holds a packet in its buffer while there is an idle server available.
This definition is analogous to the “work conserving” definition for single server queues, as it implies that servers are utilized whenever possible. However, a multi-server, work conserving queue does not always process packets at the full output rate, because a single packet can be in at most one server.\footnote{More general strategies may allow a single packet to be “split” amongst several servers for simultaneous processing. The performance tracking theorem equally holds for these more general strategies.}

Note that the example system given in Section I-A can be viewed as a multi-server, single buffer system. It can be shown that in such a system with constant, equal rate servers and with all packets having the same length, the unfinished work function \( U(t) \) is a well defined function of the input stream \( X(t) \). However, if packets have different lengths and/or if there are heterogeneous server rates, then both the priority ordering of packets and the policy used to choose the next server are important in determining \( U(t) \). For example, suppose there are two servers, one having a service rate that is twice that of the other. Then if a single packet arrives to an empty system, the strategy of immediately routing to the fast server, and the alternate strategy of routing it to the slow server, are both consistent with the work conserving definition. However (in this special case when the input stream consists of a single packet) routing the packet to the faster server will allow the system to empty sooner.

C. Performance Tracking

Consider a work-conserving, shared buffer queueing system with \( K \) servers, and let \( \mu_1(t), \mu_2(t), \ldots, \mu_K(t) \) represent the server rate processes. Choose any particular policy for ordering the packets in their buffers, and for determining which server to choose whenever such a choice is available (provided that the policy is work conserving). Let \( U_{\text{single}}(t) \) represent the total unfinished work in this system for a given input stream \( X(t) \). Let \( U_{\text{multi}}(t) \) represent the corresponding unfinished work in a work conserving, single server system with the same input stream and with server rate function \( \mu(t) = \sum_{k=1}^{K} \mu_k(t) \).

**Theorem 1:** (Tracking Theorem) If both systems are work conserving and are initially empty, and if all packet lengths are upper bounded by a fixed maximum of \( B_{\text{max}} \) bits, then for all time \( t \), we have:

\[
U_{\text{single}}(t) \leq U_{\text{multi}}(t) \leq U_{\text{single}}(t) + (K - 1)B_{\text{max}}
\]

Thus, the work conserving nature of the multi-server system enables it to closely track the performance of the optimal single-server queue. The proof of the theorem relies on the notion of a \textit{fully loaded interval}, defined as an interval of time \([t_1, t_2]\) during which all servers of the multi-server system are busy. We first present a simple but useful lemma.

**Lemma 1:** If a time \( t \) is not within a fully loaded interval, then \( U_{\text{multi}}(t) \leq (K - 1)B_{\text{max}} \).

**Proof:** If the multi-server system is not fully loaded at time \( t \), then there must be at least one idle server. Because the system is work conserving, it follows that there are no packets in the buffer at this time. Hence, there are at most \( (K - 1) \) non-empty servers, each with at most one packet. As all packets have size less than or equal to \( B_{\text{max}} \), it follows that \( U_{\text{multi}}(t) \leq (K - 1)B_{\text{max}} \).

**Proof:** (Theorem 1) Note that the first inequality \( U_{\text{single}}(t) \leq U_{\text{multi}}(t) \) follows as a special case of the multiplexing inequality given in Lecture Notes #1. Here we prove that \( U_{\text{multi}}(t) \leq U_{\text{single}}(t) + (K - 1)B_{\text{max}} \).

Fix any time \( t \).

**Case 1:** Time \( t \) is not within a fully loaded interval. In this case, by Lemma 1 we know that \( U_{\text{multi}}(t) \leq (K - 1)B_{\text{max}} \), and so \( U_{\text{multi}}(t) \leq (K - 1)B_{\text{max}} + U_{\text{single}}(t) \), yielding the result.
Case 2: Time \( t \) is within a fully loaded interval. In this case, there must be some time \( t_f \) that initiated the current fully loaded interval, where \( t_f \leq t \). We thus have:

\[
U_{\text{multi}}(t) = U_{\text{multi}}(t_f^-) + X[t_f, t] - Y_{\text{multi}}[t_f, t]
\]  

(2)

where \( Y_{\text{multi}}[t_f, t] \) represents the bit departures of the multi-server system during the interval \([t_f, t]\). However, because this is a fully loaded interval, we have:

\[
Y_{\text{multi}}[t_f, t] = \int_{t_f}^{t} \sum_{k=1}^{K} \mu_k(\tau) d\tau = \int_{t_f}^{t} \mu(\tau) d\tau
\]

(3)

Using the above equality in (2) yields:

\[
U_{\text{multi}}(t) = U_{\text{multi}}(t_f^-) + X[t_f, t] - \int_{t_f}^{t} \mu(\tau) d\tau
\]  

(4)

\[
\leq (K - 1)B_{\text{max}} + X[t_f, t] - \int_{t_f}^{t} \mu(\tau) d\tau
\]  

(5)

\[
\leq (K - 1)B_{\text{max}} + X[t_f, t] - Y_{\text{single}}[t_f, t]
\]

(6)

\[
= (K - 1)B_{\text{max}} + U_{\text{single}}(t_f^-) + X[t_f, t] - Y_{\text{single}}[t_f, t]
\]

(7)

where the inequality (4) follows by using Lemma 1 (noting that the multi-server system was not fully loaded just before time \( t_f \)), and (5) follows because the bit departures from the single server queue over any interval must be less than or equal to the integral of the server rate over this interval. This proves the theorem.

The above result is useful for understanding the performance of a multi-server system in terms of the performance of a single-server system, and demonstrates the utility of a shared buffer.

II. A Functional Equation for Unfinished Work

The unfinished work of a single-server, work conserving queue is fully determined by the \( X(t) \) arrival process and the \( \mu(t) \) server process. Thus far, we have defined unfinished work as the difference between the \( X(t) \) process and the \( Y(t) \) process, where \( Y(t) \) is the integrated service rate over all busy periods. However, the busy periods themselves are functions of the arrival and server processes. We would like to have a simple mathematical expression that establishes the value of \( U(t) \) simply in terms of \( X(t) \) and \( \mu(t) \).

**Theorem 2:** (Functional Equation) The unfinished work in a single-server, work conserving queue that is empty at time 0 is given for all \( t \geq 0 \) by the formula:

\[
U(t) = \max_{0 \leq \tau \leq t} \left[ X[\tau, t] - \int_{\tau}^{t} \mu(v) dv \right]
\]

(6)

where \( X(t) \) and \( \mu(t) \) are the corresponding arrival and server processes, and \( X[\tau, t] = X(t) - \lim_{\delta \to 0^+} X(\tau - \delta) \).

The maximum operator in (6) is used to take a maximum of the function \( X[\tau, t] - \int_{\tau}^{t} \mu(v) dv \) over all of the (infinite) values of \( \tau \) such that \( 0 \leq \tau \leq t \). In the proof below, we show that this maximum is achieved at a particular value \( \tau \) that exactly corresponds to the start of the current busy period. Before proving the theorem, we first formally define \( X[t, t] \) to be zero if there is no packet arrival at time \( t \), and to be equal to the total number of arriving bits if there is an arrival at time \( t \). Therefore, considering the particular time \( \tau = t \), we have that

\[
0 \leq X[t, t] = X[t, t] - \int_{t}^{t} \mu(v) dv \leq \max_{0 \leq \tau \leq t} \left[ X[\tau, t] - \int_{\tau}^{t} \mu(v) dv \right]
\]

(7)

and so the right hand side of (6) is indeed non-negative.

**Proof:** (Theorem 2) We prove the theorem with the following two claims, which hold for any time \( t \):

**Claim 1:** \( U(t) \leq \max_{0 \leq \tau \leq t} \left[ X[\tau, t] - \int_{\tau}^{t} \mu(v) dv \right] \).

To prove the claim, note that if \( U(t) = 0 \), then the inequality holds (recall inequality (7)). Otherwise, if \( U(t) > 0 \), then there must exist a time \( t_b \) that started the current busy period (where \( 0 \leq t_b \leq t \)), such that:

\[
U(t) = X[t_b, t] - \int_{t_b}^{t} \mu(v) dv
\]
As \( t_b \) is simply a particular time within the interval \( 0 \leq \tau \leq t \), the right hand side of the above inequality is less than or equal to \( \max_{0 \leq \tau \leq t} \left[ X[\tau, t] - \int_{\tau}^{t} \mu(v)dv \right] \), proving Claim 1.

**Claim 2:** \( U(t) \geq \max_{0 \leq \tau \leq t} \left[ X[\tau, t] - \int_{\tau}^{t} \mu(v)dv \right] \).

To prove this claim, recall that for any time \( \tau \) such that \( 0 \leq \tau \leq t \), we have:

\[
U(t) = U(\tau^-) + X[\tau, t] - Y[\tau, t] \geq X[\tau, t] - Y[\tau, t]
\]

However, \( Y[\tau, t] \leq \int_{\tau}^{t} \mu(v)dv \), and hence:

\[
U(t) \geq X[\tau, t] - \int_{\tau}^{t} \mu(v)dv \tag{8}
\]

Because inequality (8) holds for all values \( \tau \) such that \( 0 \leq \tau \leq t \), it must also hold for the particular value of \( \tau \) that maximizes the expression on the right hand side. This proves Claim 2.

Exercise: Prove the scaling equality and the multiplexing equality (from lecture notes #1) directly from the functional equation (6). (Hint: Let \( X(t) = \sum_{k=1}^{K} X_k(t) \), where \( X_k(t) \) is defined as the bit arrival process associated with bits that are processed by server \( k \).)

### III. Departures

From equation (6), we again observe that the unfinished work satisfies the following unfinished work conservation property: The unfinished work in a work conserver, single server queue does not depend on the order in which packets are served. Indeed, serving packets in any order, or even preempting service according to an arbitrary pre-emption strategy, does not change any sample path of the \( U(t) \) process. Likewise, the bit-departure function \( Y(t) \) is invariant to any service order as long as service is work conserving. However, the same is not true for the packet departure function \( D(t) \).

Example: Consider a single server, work conserving queue that is initially empty. The server is time varying with rate function \( \mu(t) \). The input process consists of two packets, one of length \( B_1 \) and the other of length \( B_2 \), where \( B_1 < B_2 \). Both packets arrive at time 0. The following figure illustrates the corresponding unfinished work functions and departure functions for the case where we serve the shortest packet first (SPF), and the case when we serve the longest packet first (LPF).

Both service disciplines yield a final departure at the end of the busy period, which occurs at the time \( t_2 \) such that \( \int_{0}^{t_2} \mu(t)dt = B_1 + B_2 \) (indeed, this final departure time is the same for all pre-emptive service strategies, provided they are work conserving). In the case of the SPF service discipline, the first departure occurs at time \( t_{SPF} \) such that \( \int_{0}^{t_{SPF}} \mu(t)dt = B_1 \). In the case of LPF service, the first departure occurs at time \( t_{LPF} \), where \( \int_{0}^{t_{LPF}} \mu(t)dt = B_2 \). It is not difficult to see that \( t_{SPF} < t_{LPF} \).

Exercise: Consider all (potentially preemptive) work conserving service disciplines for this example of two packets with sizes \( B_1 \) and \( B_2 \) arriving at time 0. What is the set of feasible departure times for the first departure?

### IV. The Notion of Rate

A fundamental issue in queueing systems is the issue of stability, which involves determining whether or not the time average service rate of a queue is sufficient to handle the rate at which data arrives. Here we establish the simple notion of an input rate and an average service rate:

**Definition 2:** An input process \( X(t) \) has input rate \( \lambda \) if \( \lim_{t \to \infty} \frac{X(t)}{t} = \lambda \) (with probability 1).

**Definition 3:** A server process \( \mu(t) \) has time average rate \( \mu_{av} \) if \( \lim_{t \to \infty} \frac{1}{t} \int_{0}^{t} \mu(v)dv = \mu_{av} \) (with probability 1).

Our use of the phrase “with probability 1,” implicitly assumes we are treating the input and server processes in a probabilistic context. The corresponding definition in the deterministic context simply removes the “with probability 1” phrase. For example, in the purely deterministic context, an input stream of rate \( \lambda \) is simply a function \( X(t) \) that is guaranteed to satisfy the condition \( X(t)/t \to \lambda \) as \( t \to \infty \).

When the phrase “with probability 1” appears in probability problems, it is almost always the case that some form of the Law of Large Numbers (LLN) has been invoked (Theorem 2.1, page 79 in Ross text). Here we recite the LLN result:
5

Theorem 3: (Strong Law of Large Numbers) Let \( \{T_i\} \) be a sequence of i.i.d. random variables with mean \( \mathbb{E} \{ T \} = \mathbb{E} \{ T \} \). Then \( \frac{1}{n} \sum_{i=1}^{n} T_i \to \mathbb{E} \{ T \} \) with probability 1.

This is a fundamental probability result, and sets the foundation for stochastic analysis of queueing systems and Renewal Theory. As an example, we apply the result to calculate the data rate of a renewal process. The analytical technique used below is one of the three most important techniques used in queueing theory.

Example: Consider an input stream \( X(t) \) that is a renewal process. Assume all packets have fixed bit lengths equal to \( B \) units, and interarrival times \( \{T_i\} \) are i.i.d. and distributed with density \( p_T(t) \) and expectation \( \mathbb{E} \{ T \} \), so that \( \int_{0}^{\infty} tp_T(t) = \mathbb{E} \{ T \} \).

Claim: For the process of the above example, with probability 1 we have:

\[
\lim_{t \to \infty} \frac{X(t)}{t} = \frac{B}{\mathbb{E} \{ T \}}
\]

Proof: (Also given in Section 7.3 of the Ross text) Note that \( X(t) = BN(t) \), where \( N(t) \) is the number of packet arrivals during the interval \([0, t]\). It suffices to show that \( \frac{N(t)}{t} \to 1/\mathbb{E} \{ T \} \) with probability 1.

(a) Claim 1: With probability 1, \( N(t) \to \infty \) as \( t \to \infty \). The statement in Claim 1 is intuitive, and a simple proof is given on page 403 (Section 7.1) of the Ross text.

(b) Let \( T_i \) represent the packet interarrival times (See figure below). Then at any time \( t \), we have:

\[
\sum_{i=1}^{N(t)} T_i \leq t \leq \sum_{i=1}^{N(t)+1} T_i
\]

(9)

It follows that:

\[
\frac{N(t)}{\sum_{i=1}^{N(t)} T_i} \geq \frac{N(t)}{t} \geq \frac{N(t)}{\sum_{i=1}^{N(t)+1} T_i}
\]

(10)

However, because (with probability 1) \( N(t) \to \infty \) as \( t \to \infty \), it follows by the law of large numbers that

\[
\lim_{t \to \infty} \frac{N(t)}{\sum_{i=1}^{N(t)} T_i} = \lim_{n \to \infty} \frac{n}{\sum_{i=1}^{n} T_i} = \frac{1}{\mathbb{E} \{ T \}}.
\]

Likewise, \( \lim_{t \to \infty} \frac{N(t)+1}{\sum_{i=1}^{N(t)+1} T_i} = \frac{1}{\mathbb{E} \{ T \}} \).

We therefore have:

\[
\lim_{t \to \infty} \frac{N(t)}{\sum_{i=1}^{N(t)} T_i} \geq \lim_{t \to \infty} \frac{N(t)}{t} \geq \left( \lim_{t \to \infty} \frac{N(t)+1}{\sum_{i=1}^{N(t)+1} T_i} \right) \left( \lim_{t \to \infty} \frac{N(t)}{N(t)+1} \right)
\]

(11)
Substituting the limiting values into the inequality above and using the fact that \( \lim_{t \to \infty} \frac{N(t)}{N(t)+1} = 1 \) (which also follows because \( N(t) \to \infty \) with probability 1), we obtain:

\[
\frac{1}{\mathbb{E}\{T\}} \geq \lim_{t \to \infty} \frac{N(t)}{t} \geq \frac{1}{\mathbb{E}\{T\}}
\]  

(12)

which holds with probability 1, proving the result.