ECE Communication Networks 1
Midterm Exam Solutions
Fall 2006

Instructions: The exam is a closed book. You are to work on the exam alone. Please clearly indicate which problem you are solving at the top of each page in the blue book. There will be 2 hours for the exam. The exam consists of varying levels of difficulty, and so it is advised that you look through the entire exam first. Should you get stuck on part of a problem, remember that there may be "easier points" elsewhere in the exam.

1. (**Stop, ARQ Time!**) In an automatic repeat request (ARQ) protocol, the primary P (sender) transmits data frames to the secondary S, who in turn responds with acknowledgement (ACK) or negative-acknowledgement (NAK) frames. Let $I(N)$ be the N-th data frame to be sent by S.

   (a) (**5 points**) Draw two rounds (N and $N+1$ data frames) of an error free SW-ARQ. Be sure to indicate data frames $I(N)$, the corresponding ACK frames, and the initiation and resetting of the sender-side timers. Your diagram should clearly label sender (P) and secondary (S) timelines.

   (b) (**10 points**) Draw one round of SW-ARQ where the data frame $I(N)$ is corrupted en route from $P \rightarrow S$. Again, be sure to indicate the initiation and resetting of timers, as well as necessary retransmissions.

   (c) (**10 points**) Draw one round of SW-ARQ where the ACK frame $ACK(N)$ is lost en route from $S \rightarrow P$. Again, be sure to indicate the initiation and resetting of timers, as well as necessary retransmissions.

   Solution: See last page of file.

2. (**A Tiny Bit of TinyOS!**) In this problem, you will revisit CRC and TinyOS.

   (a) (**10 points**) Explain how CRC implemented in TinyOS, and identify the code polynomial used in TinyOS? Your answer should include an explanation of the process. To assist, the TinyOS code used to calculate the crc is listed as follows:

```
uint16_t crcByte(uint16_t crc, uint8_t b) {   uint8_t i;

    crc = crc ^ b << 8;
    i = 8;
    do
                if (crc & 0x8000)
            crc = crc << 1 ^ 0x1021;
                else
            crc = crc << 1;
    while (--i);
    return crc;
}
```

   (b) (**15 points**) The checksummed message is the original message plus the crc result. If the checksummed message received is $0x88109387$, can this message pass the CRC checking, using the function listed above? Why?

   Solution:

   (a) In Tinyos, for each byte, the function *crcByte* is called iteratively until the last byte. The final crc remainder is appended to the original message packet. The code polynomial is $x^{16}+x^{12}+x^5+1$
   (b) No, the binary representation of $0x88109387$ is 10001000000100001001001110000111, and the crc calculation is shown in the following figure. Since the remainder is non-zero, the checksummed message does not pass the crc checking.

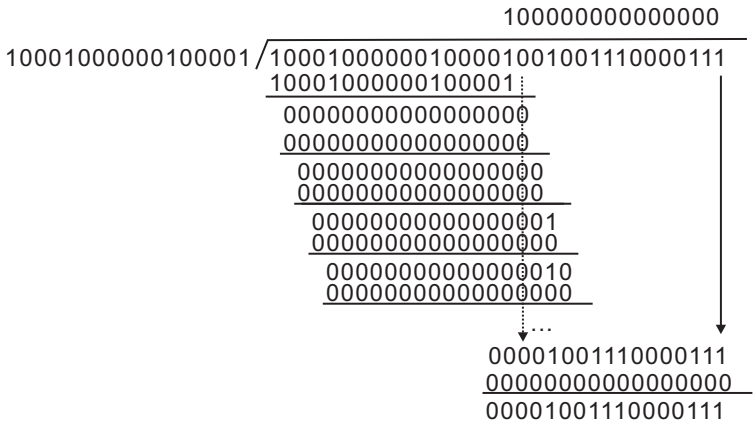


Figure 1: CRC calculation.

3. (**Engset Loss or Alot of M's (M/M/m/K/m):**) Consider a queuing system with K input sources and m parallel servers. Each source spends a random time interval U before it generates a call, where U is exponentially distributed with mean $1/\nu$. When the source has generated a call, it switches into a "being served" mode. The service time of each server is exponentially distributed with mean $1/\mu$. When all m servers are busy, and a source generates a new call, that call is lost and the source returns to its generation mode.

   (a) (**5 points**) Identify the state-dependent birth and death rates for the system.

   (b) (**5 points**) Draw a state transition diagram for the $M/M/m/K/m$ queue.

   (c) (**10 points**) Let $\rho = \nu/\mu$. Using the birth-death formulation, calculate the steady-state distribution $p_n(K)$ for the number of calls being serviced at an arbitrary time. This distribution is known as the Engset distribution.

   (d) (**10 points**) Using your result from part (b) of this problem, consider letting $K \to \infty$ while keeping $K\nu = \lambda$ fixed. Show that the Engset distribution converges to the Erlang distribution:

$$q_n = \frac{\frac{a^n}{n!}}{\sum_{i=0}^{m} \frac{a^i}{1!}},$$

   where $a = \lambda/\mu$.

Solution:

(a) Clearly $\lambda(n) = (K-n)\nu$ for $0 \le n \le m$, and 0 otherwise. Also, the death rate is the same as in the Erlang loss model $\mu(n) = n\mu$ for $1 \le n \le m$.

(b) Just draw the figure corresponding to part (a).

(c) From our derivations in class, the normalization constant is

$$G = \sum_{n=0}^{\infty} \prod_{i=0}^{n-1} \frac{\lambda(i)}{\mu(i+1)}$$

$$= \sum_{n=0}^{m} \frac{K(K-1)(K-2)\cdots(K-n+1)\nu^n}{n!\mu^n}$$

$$= \sum_{n=0}^{m} \binom{K}{n} r^n$$

where $r = \nu/\mu$. Thus, the distribution follows a truncated binomial distribution

$$p_n(K) = \frac{\binom{K}{n} r^n}{\sum_{i=0}^{m} \binom{K}{i} r^i}$$

for $0 \leq n \leq m$.

(c) To see the convergence, let $K\nu = \lambda$ be held constant and let $K \to \infty$. Then

$$\binom{K}{n} r^n = \frac{K(K-1)\cdots(K-n+1)(Kr)^n}{K^n n!} \to \frac{(\lambda/\mu)^n}{n!}.$$

Substitution gives the convergence to

$$q_n = \frac{\frac{a^n}{n!}}{\sum_{i=0}^{m} \frac{a^i}{1!}},$$

where $a = \lambda/\mu$.

4. (**Which Queue Would You Choose?:**) In this problem, we will examine two different queues and compare their performance based on their average throughput and delay.

   (a) (**10 points**) Consider the two following queues: Show that these two different queues have

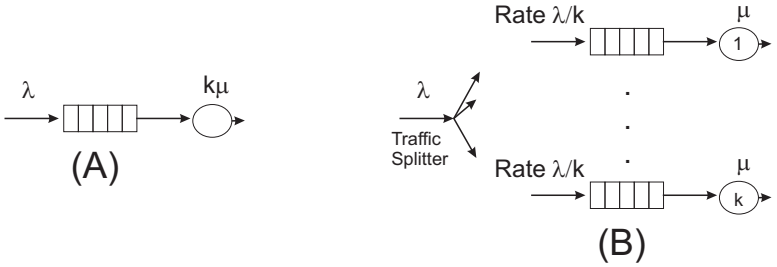


Figure 2: Two different queues: (A) an $M/M/1$ queue with service rate $k\mu$; (B) k separate $M/M/1$ queues with service rate μ (note: traffic splitting yields k Poisson streams of parameter λ).

   the same traffic intensity (and hence throughput).

   (b) (**10 points**) Although these two queues have the same throughput, they have different delay characterizations. Examine the average time that a packet/job spends in each queueing system (i.e. the whole system). Which system has better delay QoS behavior?

   Solution:

   (a) Call the first system A, and the second system B. The traffic intensity for system A is $\rho = \frac{\lambda}{k\mu}$. System B also has traffic intensity $\rho = \frac{\lambda}{k\mu}$ as the total traffic rate entering is λ and the net rate leaving is $k\mu$.

   (b) The average system time for system A is

$$E[T_A] = \frac{1}{k\mu(1-\rho)}$$

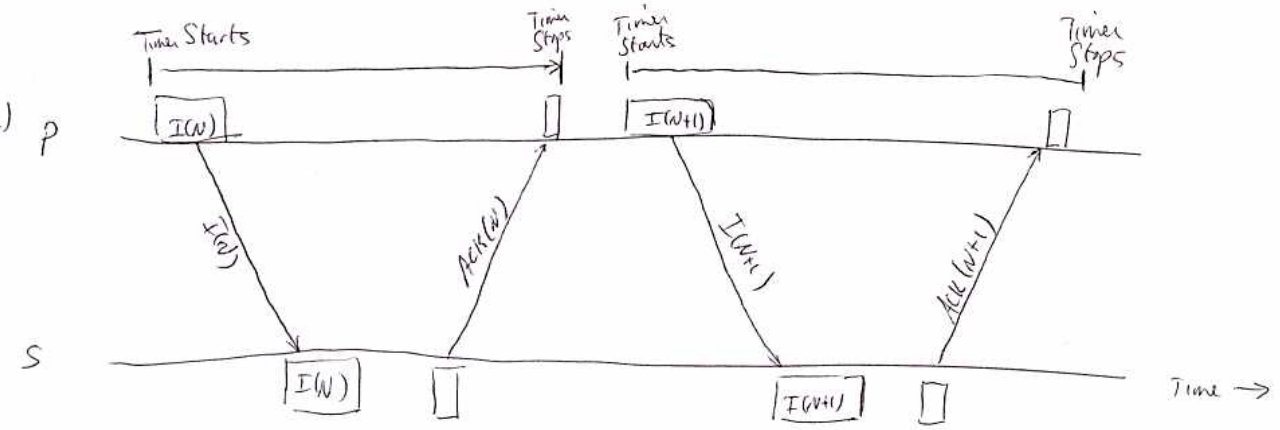
   while for system B it is

$$E[T_B] = \frac{1}{\mu(1-\rho)}.$$

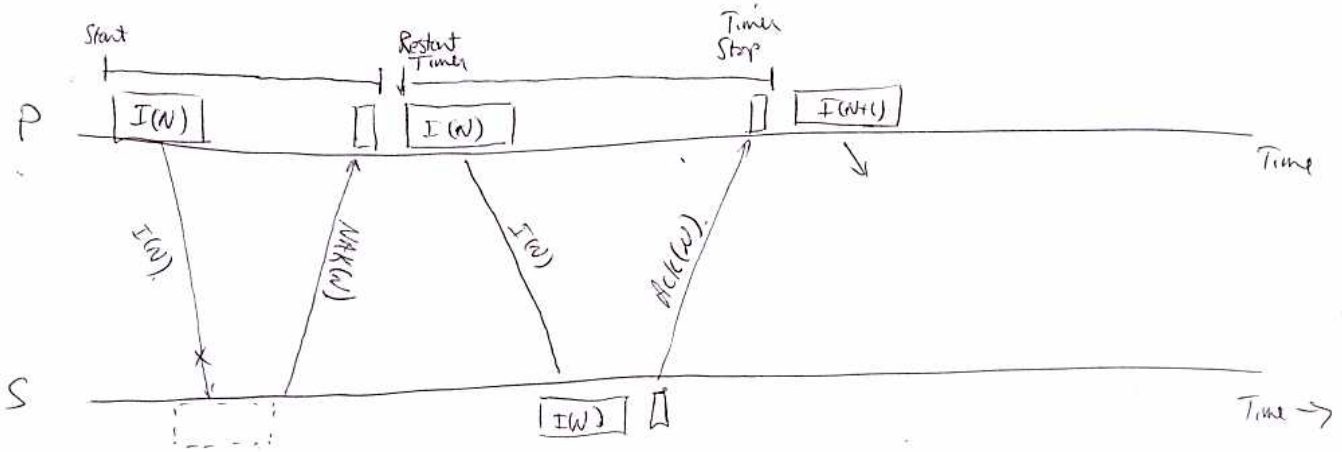
   Clearly, $E[T_B] = kE[T_A]$ implies that replacing k smaller systems by one with the same processing capability leads to a reduction in the system time by a factor of k. System A has better processing time.

(1)

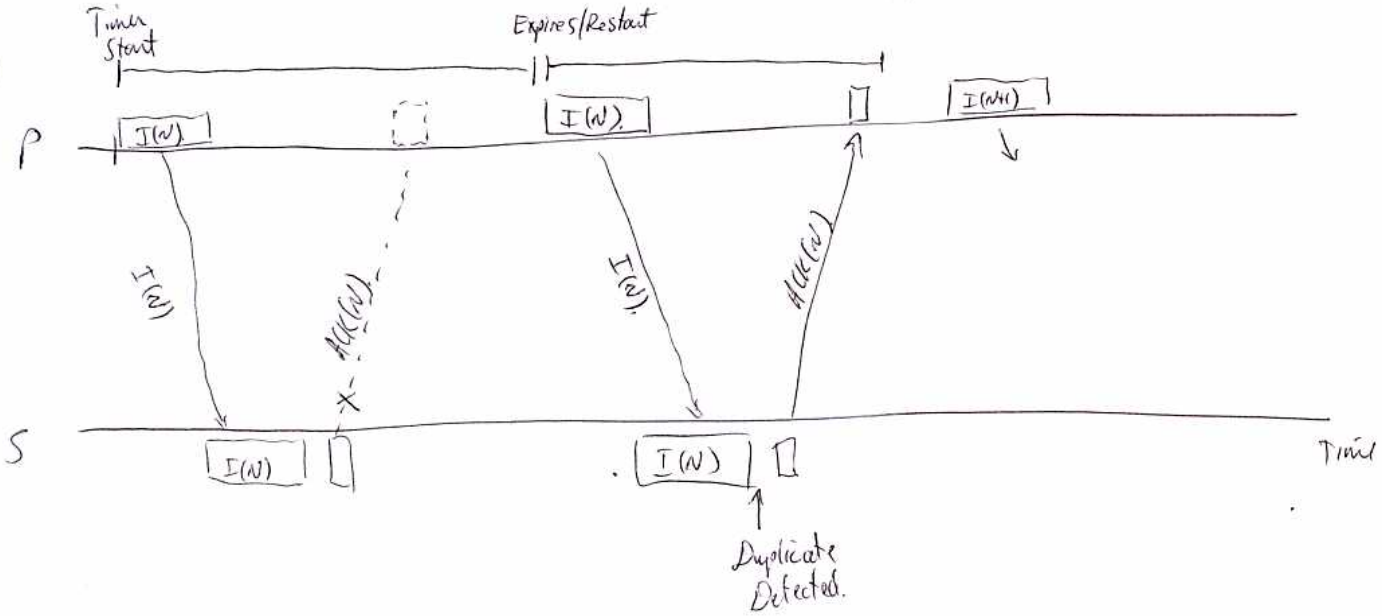
(a)


Timer Starts
Timer Stops
Timer Starts
Timer Stops
P
I(N)
I(N+1)
I(N)
ACK(N)
I(N+1)
ACK(N+1)
S
I(N)
I(N+1)
Time →

(b)


Start
Restart Timer
Timer Stops
P
I(N)
I(N)
I(N+1)
Time
I(N)
NAK(N)
I(N)
ACK(N)
S
I(N)
Time →

(c)


Timer Start
Expires/Restart
P
I(N)
I(N)
I(N+1)
I(N)
ACK(N)
I(N)
ACK(N)
S
I(N)
I(N)
Time
Duplicate Detected.