Introduction to Computer Networks
Midterm solution

1.

TDM (Time-Division Multiplexing) or FDM (Frequency-Division Multiplexing)

2.

(a) Packet switching: The resource can be utilized more efficiently for multiple users.

(b) Circuit switching: The end-to-end performance can be guaranteed.

3. (a) $\frac{3\text{Mbps}}{1\text{Mbps}} = 3 \text{ users}$

(b) $\sum_{k=4}^{5} C_k^5 0.5^k 0.5^{5-k} = 0.5^5(5+1) = 0.1875$

4.

(a) $\mathrm{R} * \mathrm{T}_{prop} = 1 \text{ Mbps} * \frac{10000*10^3}{2.5*10^8} = 10^6 * 0.04 = 4 * 10^4$

(b) $4 * 10^4$

5.

Min(R_S, R_C, R/M)

6.

(a) F
(b) F
(c) F
(d) F
(e) F
(f) T
(g) F
(h) F
(i) T
(j) F
(k) F
(l) F

7.

(a)

$$\Delta = \frac{900000}{15 \times 10^6} = 0.06\,(\mathrm{s})$$

$$\text{access delay}: d_a = \frac{0.06}{1 - 0.06 \times 15} = 0.6\,(\mathrm{s})$$

$$\text{total time} = 0.6 + 0.2 = 0.8\,(\mathrm{s})$$

(b)

$$\lambda = 15 * 0.6 = 9$$

$$\text{access delay}: d_a = \frac{0.06}{1 - 0.06 * 9} = \frac{1}{8}\,(\mathrm{s})$$

$$\text{total time} = 0.4 * 0 + 0.6 * (0.2 + 1/8) = 0.195\,(\mathrm{s})$$

8.

(a) 8RTT

(b) 4RTT

(c) 3RTT

9.

(a)

Suppose the receiver has received packet which sequence number is up to k-1 and has ACKed those packets, all of possible cases will be bounded by the following cases:

Case1:

If all of those ACKs have been received by sender, then the window is [k, k-1+n]

Case2:

If none of those ACKs have been received by sender, then the window is [k-n, k-1]

Therefore, the possible sequence number is [k-n, k-1+n]

(The window begins somewhere in [k-n, k] and the window size is n)

(b)

Because the receiver is waiting for packet k and the sender has sent packets [k-n, k-1], the range of possible ACK values is from k-1-n to k-1

10.

(a)

Condition: ACK/NAK may be corrupted

What will happen: The receiver can’t tell whether the retransmitted packet is a duplicate packet or a new packet

(b)

Sender adds sequence number to each packet, and receiver discards the duplicate packet.

11.

(a)

In order to avoid the occurrence of this problem, we want to avoid having the leading edge of the receiver's window (i.e., the one with the “highest” sequence number) wrap around in the sequence number space and overlap with the trailing edge (the one with the "lowest" sequence number in the sender's window). That is, the sequence number space must be large enough to fit the entire receiver window and the entire sender window without this overlap condition. So - we need to determine how large a range of sequence numbers can be covered at any given time by the receiver and sender windows.

Suppose that the lowest-sequence number that the receiver is waiting for is packet m. In this case, its window is [m, m+w-1] and it has received (and ACKed) packet m-1 and the w-1 packets before that, where w is the size of the window. If none of those w ACKs have been yet received by the sender, then ACK messages with values of [m-w,m-1] may still be propagating back. If no ACKs with these ACK numbers have been received by the sender, then the sender's window would be [m-w, m-1].

Thus, the lower edge of the sender's window is m-w, and the leading edge of the receivers window is m+w-1. In order for the leading edge of the receiver's window to not overlap with the trailing edge of the sender's window, the sequence number space must thus be big enough to accommodate 2w sequence numbers. That is, the sequence number space must be at least twice as large as the window size, $k \geq 2w$.

12.

(a) 249

13.

It takes 1 microsecond (or 0.001 milliseconds) to send a packet. In order for the sender to be busy 90 percent of the time, we must have

$$util = \frac{0.001n}{10+0.001} > 0.9$$

⇨ $n \geqq 9001$

14.

(a) $u_s \leq (u_1+\ldots+u_N+u_s)/N$

Now that the sufficient condition cannot hold, each peer cannot fully utilize its upload link and the server cannot provide additional rate. The server needs to divide its upload rate into $r_1,\ldots,r_N$ for each peer, where $r_i = u_s u_i/(u_1+\ldots+u_N)$. (Note that $r_1+\ldots+r_N=u_s$ and $(N\text{-}1)r_i \leq u_i$.) The distribution scheme is designed as follows:

The file is broken into N parts, with each part having size r_i/u_s. The server transmits bits from part i to peer i at rate r_i. Each peer forwards the bits received from the server to each of the other peers at rate r_i. As a result, each peer can concurrently receive the file at rate $r_1+\ldots+r_N=u_s$. Finally, the total distribution time is F/u_s.

(b) $u_s \geq (u_1+\ldots+u_N+u_s)/N$

In P2P, a good idea is that each peer has part of the file and transmits it to each of the other peers at the same rate. Thus, peer i may transmit the part of file to other peer at rate $r_i=u_i/(N\text{-}1)$. The problem is whether the server can provide each part of file for each peer in time or not.

Since peer i can provide rate r_i for other peers, if the server can provide rate $r_1+\ldots+r_N$ in total, then each peer can fully utilize its upload link. Thus the sufficient condition is $u_s \geq r_1+\ldots+r_N = (u_1+\ldots+u_N)/(N\text{-}1)$. If the condition holds, the required distribution time can be NF/$(u_1+\ldots+u_N+u_s)$ since the total upload rate is $(u_1+\ldots+u_N+u_s)$ and N copies of the file will be transmitted.

In this problem, $u_s \geq (u_1+\ldots+u_N+u_s)/N$ is equivalent to this condition; therefore, each peer can fully utilize its upload link. Moreover, the server can provide additional rate $r_{N+1}=(u_s\text{-}(r_1+\ldots+r_N))/N$ for each peer. The distribution scheme is designed as follows:

The file is broken into N+1 parts, with the i_{th} part having size $(r_i/u_s)F$ bits. The server transmits bits from part i to peer i at rate r_i and transmits bits from part N+1 to each peer at rate r_{N+1}. Each peer forwards the bits received from the server to each of the other peers at rate r_i. As a result, each peer can concurrently receive the file at rate $r_1+\ldots+r_N+r_{N+1}=(u_1+\ldots+u_N+u_s)/N$. Finally, the total distribution time is $NF/(u_1+\ldots+u_N+u_s)$.

15.

E(T_{sojourn}) = $p_0(T_{s_0}) + p_1(T_{s_1}) + p_2(T_{s_2}) + \ldots$

$$= \sum_{k=0}^{\infty} p_k (T_{s_k})$$

$$= \sum_{k=0}^{\infty} p_k (T_{q_k} + T_{tran\ k})$$

$$= \sum_{k=0}^{\infty} p_k \left(\frac{k}{\mu} + \frac{1}{\mu}\right)$$

$$= \sum_{k=0}^{\infty} \left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\mu}\right)^k \left(\frac{k+1}{\mu}\right)$$

$$= \left(1 - \frac{\lambda}{\mu}\right) \left(\frac{1}{\mu}\right) \sum_{k=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^k (k+1)$$

$$= \left(1 - \frac{\lambda}{\mu}\right) \left(\frac{1}{\mu}\right) \frac{1}{(1-\frac{\lambda}{\mu})^2}$$

$$= \frac{1}{\mu - \lambda}$$