ECE 158A 2019 Fall Midterm Practice Questions

Problem 1 (packet delay)

A file of size x is divided into n packets of equal sizes (ignoring header size) and transmitted over a path from node S to node D. The path consists of m links. The rate and the physical length of the i-th link are known to be r_i and l_i.The intermediate nodes separating the links are store-and-forward devices, each of which introduces a processing delay d_{proc}. Determine the following:

(a) The time difference between the arrivals of two successive packets at the destiny D.

Ans: The time difference equals

$$\frac{x/n}{\min_{i\in\{1,\ldots,m\}} r_i}$$

where x/n is the packet size, $\min_{i\in\{1,\ldots,m\}} r_i$ is the rate of the bottleneck link.

(b) The total time required to transfer the file.

Ans: The delay experienced by the first packet on the i-th link and the node following it equals $\frac{l_i}{v} + \frac{x/n}{r_i} + d_{proc}$. Thus the total delay it experiences is

$$\frac{1}{v}\sum_{i=1}^{m} l_i + \frac{x}{n}\cdot\sum_{i=1}^{m}\frac{1}{r_i} + m\cdot d_{proc}$$

The answer to part (a) is also the time difference between the reception of the the last bit of two succesive packets. So the time difference between the completions of receiving the first and the last packets is $n-1$ times the answer to (a). The total delay is the delay experienced by the first packet plus this time difference.

$$\frac{1}{v}\sum_{i=1}^{m} l_i + \frac{x}{n}\cdot\left(\sum_{i=1}^{m}\frac{1}{r_i} + \frac{n-1}{\min_{i\in\{1,\ldots,m\}} r_i}\right) + m\cdot d_{proc}$$

Problem 2 (Queueing)

Consider two different links that can be modeled as M/M/1 queues, denoted respectively as Q1 and Q2. Q1 has a server with service rate $\mu_1 = 200$ packets per second (in this problem, we assume all packets have the same length). Q2 has a server with service rate $\mu_2 = 300$ packets per second. There is a single incoming packet flow with arrival rate $\lambda = 400$ packets per second, and we need to split it among Q1 and Q2 in a fair way. λ_1 will denote the flow sent to Q1, while λ_2 will denote the flow sent to Q2.

(a) Observing that $\mu_2 = 3\mu_1/2$, we try to split the incoming flow in the same proportion, that is, $\lambda_2 = 3\lambda_1/2$. We get $\lambda_1 = 160$ packets per second, $\lambda_2 = 240$ packets per second.

1) Compute the load factors ρ_1 and ρ_2 of Q1 and Q2.

$$\rho_1 = \frac{\lambda_1}{\mu_1} = 160/200 = 0.8$$

$$\rho_2 = \frac{\lambda_2}{\mu_2} = 240/300 = 0.8$$

2) Compute the average time T_1 that a packet spends in Q1, and the average time T_2 that a packet spends in Q2.

$$T_1 = \frac{1}{\mu_1 - \lambda_1} = \frac{1}{200 - 160} = 25ms$$
$$T_2 = \frac{1}{\mu_2 - \lambda_2} = \frac{1}{300 - 240} = 16.7ms$$

3) In your opinion, was the traffic flow split in a fair way between Q1 and Q2?

No, even if the load factor is the same in the two queue, the average times that packets spends in them are different.

(b) We want to split the traffic flow (λ_1 to Q1, λ_2 to Q2) in such a way that the average times T_1 and T_2 are equal.

4) Write T_2 as a function of μ_2 and λ_1.

$$T_2 = \frac{1}{\mu_2 - (\lambda - \lambda_1)}$$

5) Compute the value of λ_1 such that $T_1 = T_2$.

$$\frac{1}{\mu_1 - \lambda_1} = \frac{1}{\mu_2 - (\lambda - \lambda_1)}$$
$$\mu_2 - (\lambda - \lambda_1) = \mu_1 - \lambda_1$$
$$\lambda_1 = (\mu_1 - \mu_2 + \lambda)/2 = 150\, packets\ per\ second$$

6) Compute the corresponding value of λ_2.

$$\lambda_2 = \lambda - \lambda_1 = 250\, packets\ per\ second$$

7) Compute the load factors ρ_1 and ρ_2 of Q1 and Q2.

$$\rho_1 = \frac{\lambda_1}{\mu_1} = 150/200 = 0.75$$
$$\rho_2 = \frac{\lambda_2}{\mu_2} = 250/300 = 0.833$$

8) Compute the average time T_1 that a packets spends in Q1, and the average time T_2 that a packets spends in Q2.

$$T_1 = \frac{1}{\mu_1 - \lambda_1} = \frac{1}{200 - 150} = 20ms$$
$$T_2 = \frac{1}{\mu_2 - \lambda_2} = \frac{1}{300 - 250} = 20ms$$

9) In terms of average time a packet spends in the system, is it better to have the traffic split between Q1 and Q2 (as we just computed), or to have the entire flow λ sent to a queue Q3 with service rate $\mu_3 = \mu_1 + \mu_2 = 500$ packets per second. As part of your answer, you need to compute the average time T_3 a packets would spend in Q3.

$$T_3 = \frac{1}{\mu_3 - \lambda} = \frac{1}{500 - 400} = 10ms. This\ is\ better$$

(c) We want to split the traffic flow (λ_1 to Q1, λ_2 to Q2) in a way that $T_1 = T_2/2$.

10) Find $\lambda_1, \lambda_2, \rho_1, \rho_2, T_1, T_2$ in this case.

$$\frac{1}{\mu_1 - \lambda_1} = \frac{1}{2} \cdot \frac{1}{\mu_2 - (\lambda - \lambda_1)}$$
$$2\mu_2 - 2(\lambda - \lambda_1) = \mu_1 - \lambda_1$$

$$\lambda_1 = (\mu_1 - 2\mu_2 + 2\lambda)/3 = 133\, packets\ per\ second$$

$$\lambda_2 = \lambda - \lambda_1 = 267\, packets\ per\ second$$

$$\rho_1 = \lambda_1/\mu_1 = 133/200 = 0.665$$

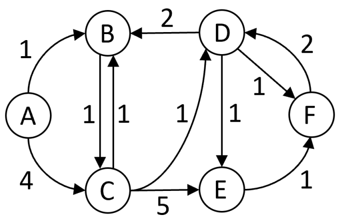
$$\rho_2 = \lambda_2/\mu_2 = 267/300 = 0.89$$

$$T_1 = \frac{1}{\mu_1 - \lambda_1} = \frac{1}{200 - 133} = 14.9ms$$

$$T_2 = \frac{1}{\mu_2 - \lambda_2} = \frac{1}{300 - 267} = 30ms$$

Problem 3 (Dijkstra's algorithm)

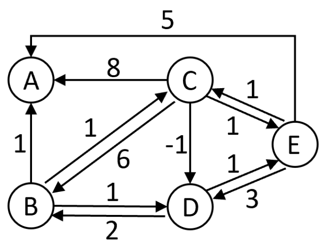
Perform Dijkstra's algorithm on the network shown below to determine the shortest path from each node to node F.



Ans:

Step	Set S	$D(A), p(A)$	$D(B), p(B)$	$D(C), p(C)$	$D(D), p(D)$	$D(E), p(E)$
1	F	∞	∞	∞	1,F	$\underline{\text{1,F}}$
2	F,E	∞	∞	6,E	$\underline{\text{1,F}}$	
3	F,E,D	∞	∞	$\underline{\text{2,D}}$		
4	F,E,D,C	6,C	$\underline{\text{3,C}}$			
5	F,E,D,C,B	$\underline{\text{4,B}}$				

Problem 4 (Bellman-Ford algorithm)



Consider the network described by the directed graph above.

(a) Apply Bellman-Ford algorithm to determine the shortest path from each node to node A.

Sol: Nodes whose shortest path length changes in one iteration send update messages to their successor nodes in the next iteration. Assume all messages in each iteration are sent simutaneous, so the messages all contain the information from the previous iteration, not affected by any change made in the current iteration. The algorithm converges when no node's shortest path length changes in an iteration. In the table below, the shortest path's length from each node is underlined.

Iteration	**B**			**C**				**D**		**E**			*nodes whose shortest*
number	A	C	D	A	B	D	E	B	E	A	C	D	*path length changed*
1	$\underline{1}$	∞	∞	$\underline{8}$	∞	∞	∞	∞	∞	$\underline{5}$	∞	∞	B($\infty \to 1$), C($\infty \to 8$), E($\infty \to 5$)
2	$\underline{1}$	9	∞	8	7	∞	$\underline{6}$	$\underline{3}$	6	$\underline{5}$	9	∞	C($8 \to 6$), D($\infty \to 3$)
3	$\underline{1}$	7	4	8	7	$\underline{2}$	6	$\underline{3}$	6	$\underline{5}$	7	6	C($6 \to 2$)
4	$\underline{1}$	3	4	8	7	$\underline{2}$	6	$\underline{3}$	6	5	$\underline{3}$	6	E($5 \to 3$)
5	$\underline{1}$	3	4	8	7	$\underline{2}$	4	$\underline{3}$	4	5	$\underline{3}$	6	

(b) Assume that after the algorithm in (a) converges, the cost of the link from E to A suddenly drops from 5 to 1. Show how the Bellman-Ford algorithm adapts to this change.

Ans: Copy the final state of part (a), change the length of the direct path from E to A from 5 to 1, and use this as the initial state.

Iteration	**B**			**C**				**D**		**E**			*nodes whose shortest*
number	A	C	D	A	B	D	E	B	E	A	C	D	*path length changed*
1	$\underline{1}$	3	4	8	7	$\underline{2}$	4	$\underline{3}$	4	$\underline{1}$	3	6	E($3 \to 1$)
2	$\underline{1}$	3	4	8	7	$\underline{2}$	2	3	$\underline{2}$	$\underline{1}$	3	6	D($3 \to 2$)
3	$\underline{1}$	3	3	8	7	$\underline{1}$	2	3	$\underline{2}$	$\underline{1}$	3	5	C($2 \to 1$)
4	$\underline{1}$	2	3	8	7	$\underline{1}$	2	3	$\underline{2}$	$\underline{1}$	2	5	

(c) Assume the link cost from E to A resumes its previous value after the algorithm coverges in (b). How many iterations does it take the Bellman-Ford algorithm to return to the old result?

Ans: Copy the final state of part (b), change the length of the direct path from E to A back to 5, and use this as the initial state. It can be seen that after 5 iterations the algorithm converges and the old result is resumed.

Iteration	**B**			**C**				**D**		**E**			*nodes whose shortest*
number	A	C	D	A	B	D	E	B	E	A	C	D	*path length changed*
1	$\underline{1}$	2	3	8	7	$\underline{1}$	2	3	$\underline{2}$	5	$\underline{2}$	5	E($1 \to 2$)
2	$\underline{1}$	2	3	8	7	$\underline{1}$	3	3	$\underline{3}$	5	$\underline{2}$	5	D($2 \to 3$)
3	$\underline{1}$	2	4	8	7	$\underline{2}$	3	3	$\underline{3}$	5	$\underline{2}$	6	C($1 \to 2$)
4	$\underline{1}$	3	4	8	7	$\underline{2}$	3	3	$\underline{3}$	5	$\underline{3}$	6	E($2 \to 3$)
5	$\underline{1}$	3	4	8	7	$\underline{2}$	4	$\underline{3}$	4	5	$\underline{3}$	6	

Problem 5 (FEC)

We want to multicast N packets $p_1, p_2, ..., p_N$ across a link, but unfortunately packets can get lost. We decide to implement a FEC strategy, which consists in generating and transmitting $M > N$ packets $c_1, ..., c_M$. Each packet c_i is obtained as a combination (modulo-2 sum) of a subset of $p_1, p_2, ..., p_N$. For example, fixed $a_{i,1}, a_{i,2}, ..., a_{i,N} \in \{0, 1\}$

$$c_i = \begin{bmatrix} a_{i,1} & a_{i,2} & \cdots & a_{i,N} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}$$

means that c_i is obtained as the (bit-by-bit) modulo-2 sum of the packets p_j corresponding to all coeffcients $a_{i,j}$ that are equal to 1. Out of all M transmitted packets, $M' \leq M$ are received. Let these packets be $c_{i_1}, c_{i_2}, ..., c_{i_{M'}}$, where $i_1, i_2, ..., i_{M'}$ are distinct indices in $\{1, ..., M\}$. The received packets $c_{i_1}, ..., c_{i_{M'}}$ and the original packets $p_1, ..., p_N$ are related as

$$\begin{bmatrix} c_{i_1} \\ c_{i_2} \\ \vdots \\ c_{i_{M'}} \end{bmatrix} = \begin{bmatrix} a_{i_1,1} & a_{i_1,2} & \cdots & a_{i_1,N} \\ a_{i_2,1} & a_{i_2,2} & \cdots & a_{i_2,N} \\ \vdots & \vdots & & \vdots \\ a_{i_{M'},1} & a_{i_{M'},2} & \cdots & a_{i_{M'},N} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix} \triangleq A \cdot \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix}$$

Note that A is a $M' \times N$ matrix. We have that $p_1, ..., p_N$ can be recovered from $c_{i_1}, c_{i_2}, ..., c_{i_{M'}}$ if and only if A has rank N.

(a) What is the minimum M' below which it is not possible to recover $p_1, ..., p_N$? If A has rank N, we can select N linearly independent rows of it and obtain a $N \times N$ matrix A' of rank N (note that A' is therefore invertible).

Ans: We need to receive $M' \geq N$ packets (so the minimum is N). If $M' < N$ we cannot build a $N \times N$ matrix A' which is invertible and allows to recover the original packets. If $M' \geq N$, then the packets can be recovered if and only if there is a $N \times N$ submatrix A' of A (obtained choosing N linearly independent rows of A) that is invertible.

(b) How can we use A' to recover $p_1, ..., p_N$ from the received packets $c_{i_1}, c_{i_2}, ..., c_{i_{M'}}$ corresponding to the chosen rows of A'?

Ans: $p = (A')^{-1}c$

Assume now that we want to transmit packets p_1, p_2, p_3 and that we choose to adopt a FEC strategy sending 5 packets $c_1, ..., c_5$ obtained as follows

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

(c) If we receive c_1, c_2, c_4, can we recover p_1, p_2, p_3? If so, explain how, if not, explain why.

Ans: The matrix A corresponding to the received packets c_1, c_2, c_4 is

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

The matrix has rank 2, and thus it is not invertible. Therefore, p_1, p_2, p_3 cannot be recovered.

(d) If we receive c_1, c_2, c_4, c_5, can we recover p_1, p_2, p_3? If so, explain how, if not, explain why. The matrix A corresponding to the received packets c_1, c_2, c_4, c_5 is

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

The matrix has rank 3. We can choose the rows of A the received packets c_1, c_2, c_5, obtaining

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

which is, of course, invertible, and can be used to recover p_1, p_2, p_3.

(e) Assume that each of the packets $c_1, ..., c_5$ get lost with probability ρ independently of the other. What is the probability that p_1, p_2, p_3 can be recovered? Consider the case of 3 received packets out of $c_1, ..., c_5$. Out of the $\left(\begin{array}{c} 5 \\ 3 \end{array} \right) = 10$ possibilities, only 2 choices do not allow to build a matrix with rank 3 and therefore do not allow to recover p_1, p_2, p_3. These cases are c_1, c_2, c_4 and c_2, c_3, c_5. The other 8 possibilities lead to correct recovery p_1, p_2, p_3. Given a set of 3 packets, the event of receiving these 3 packets and not receiving the other 2 packets has probability $(1-\rho)^3\rho^2$. Therefore the probability of the 8 possibilities is $8(1-\rho)^3\rho^2$.

Consider the case of 4 received packets out of $c1, ..., c5$. Out of the $\left(\begin{array}{c} 5 \\ 4 \end{array} \right) = 5$ possibilities, all lead to correct recovery p_1, p_2, p_3. Given a set of 4 packets, the event of receiving these 4 packets and not receiving the other packet has probability $(1-\rho)^4\rho$. Therefore the probability of the 8 possibilities is $5(1-\rho)^4\rho$.

Consider the case of 5 received packets out of $c_1, ..., c_5$. This possibility leads to correct recovery p_1, p_2, p_3, and has probability $(1-\rho)^5$. The probability of successful recovery is therefore

$$P_{recovery} = 8(1-\rho)^3\rho^2 + 5(1-\rho)^4\rho + (1-\rho)^5$$