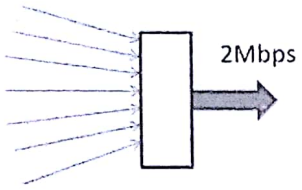
1. The edge link of an ISP has a 2Mbps (1Mbps = 10^6 bits per sec) transmission rate capacity, and the ISP wants to provide 200Kbps (1Kbps = 10^3 bits per sec) transmission rate to its users.



a. How many users can the link support under circuit switching? Explain your answer. [1]

Since the capacity will be divided equally... (½) among all the users, max # users supported = $\frac{2\,Mbps}{200\,Kbps} = 10$ ..... (½)

b. Give two drawbacks and one advantage of circuit switching? [3]

Disadv:
- Circuit establishment overhead before any transmission can begin (1)
- Unused capacity of all reserved capacity is not used, esp. true in web traffic (bursty) (1)

Adv:
- Guaranteed bw. & latency. either (1)
- Packet switching is best effort (1)
- Lower latency by avoiding trans. & queueing delays (nearly)

c. Write an equation to find the number of users the ISP can support under packet switching, if the probability of a user being active is 0.1 and the ISP wishes to provide the promised service to its users more than 95% of the time. You do not have to solve the equation. Explain your answer. [4]

- Prob. active = 0.1 ; Prob. inactive = 0.9
- Number of users that can be probabilistically supported = n (1)
- Max active users at any time = 10 (1)

- Prob. of 1 active user $= {}^{n}C_1 (0.1)(0.9)^{n-1}$
- Prob. of 2 " " $= {}^{n}C_2 (0.1)^2 (0.9)^{n-2}$

⋮

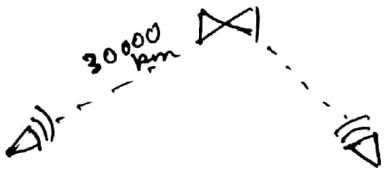
- Prob. of 10 " " $= {}^{n}C_{10} (0.1)^{10} (0.9)^{n-10}$

- Total prob. of at at most 10 active users $= \sum_{i=1}^{10} {}^{n}C_i (0.1)^i (0.9)^{n-i}$

> 0.95 ... (1)

... (2)

2. A satellite is 30,000km from Earth, signals travel at the speed of light (3 x 10^8 m/s), the uplink transmission rate is 25KBps (1 KBps = 10^3 bytes per sec), the downlink transmission rate is 100KBps. Assume queueing and processing delays to be negligible. What is the one way latency for an earth station to send a 1000 byte packet to another earth station via the satellite? Explain the steps in your calculation. [4]



(1) ... Prop. delay $= \frac{30000 + 30000 \text{ km}}{3 \times 10^8 \text{ m/s}} = \frac{60000 \times 10^3}{3 \times 10^8}$

$= .20 \text{ sec} = 200 \text{ ms}$

(1) ... Trans. delay on uplink $= \frac{1000 \text{ bytes}}{25 \text{ KBps}}$

$= 0.04 \text{ sec} = 40 \text{ ms}$

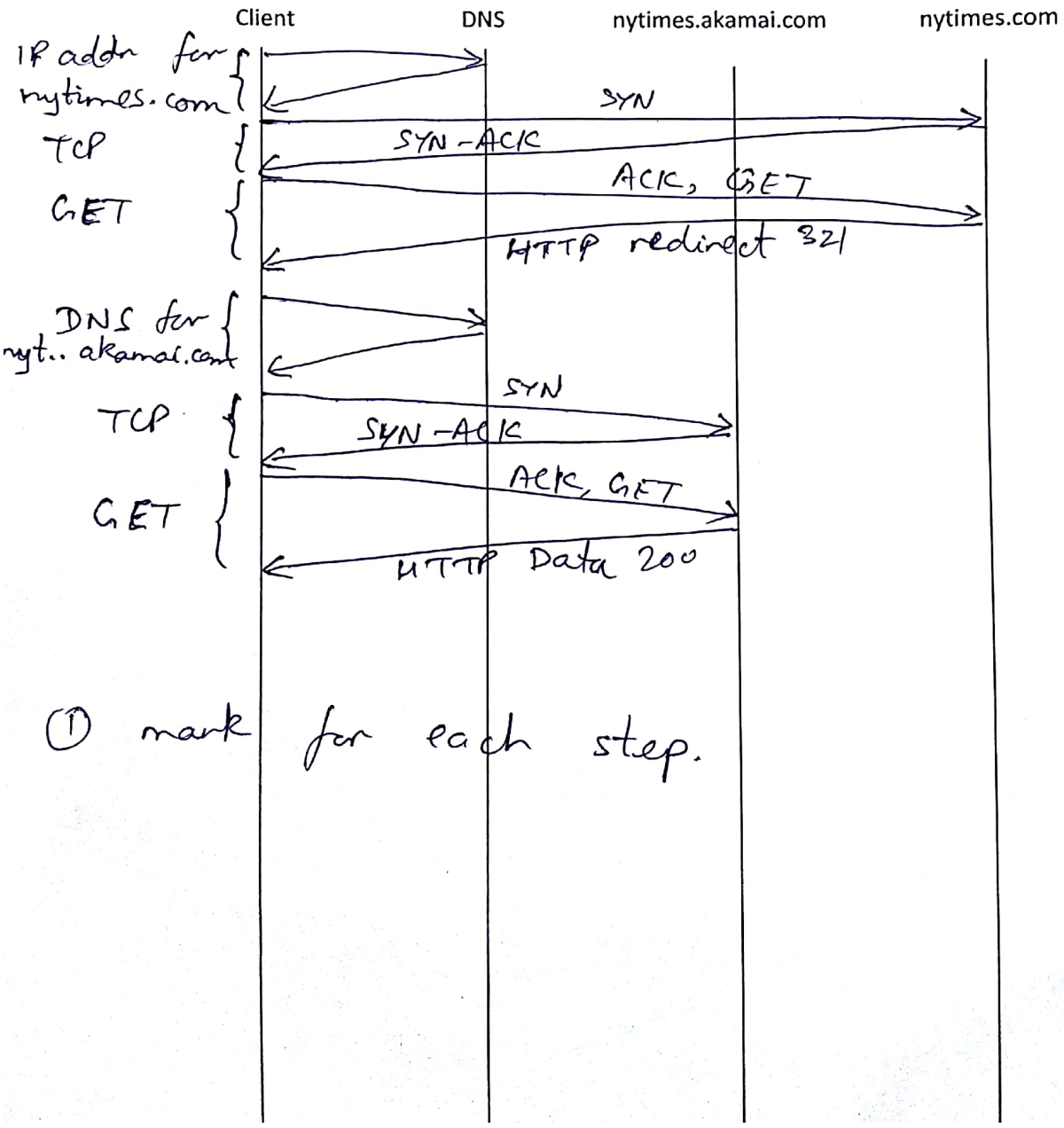
(1) ... Trans delay on downlink $= \frac{1000 \text{ bytes}}{100 \text{ KBps}}$

$= 0.01 \text{ sec} = 10 \text{ ms}$

(1) ... Total one-way latency $= 250 \text{ ms}$

3. Akamai is a CDN (Content Delivery Network) provider. It places its content servers inside the networks of regional ISPs. An ISP like Airtel may have an Akamai server inside its network, and similarly Reliance may also have an Akamai server inside its network.

a. Suppose a content provider like NYTimes wants to use Akamai. It does this by responding to the very first HTTP GET request to www.nytimes.com, with a redirection to nytimes.akamai.com. The client will then use DNS to resolve nytimes.akamai.com, and since Akamai would have populated the DNS to return the IP address for the Akamai CDN server that is closest to the client, therefore after the DNS resolution the client will be able to connect to the closest server to get the content. Trace out the TCP connection establishment, HTTP, and DNS requests, in the transaction diagram below. [6]



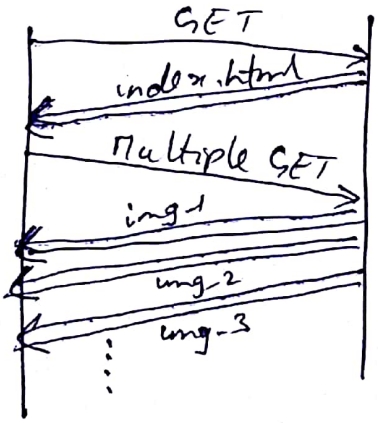
(1) mark for each step.

b. What is the key benefit of using content delivery networks? [1]

① Requests are served from CDN servers closer to the clients, to reduce latency.

c. Assume the page structure is as follows: an index.html file 100KB (1KB = 10^3 bytes) in size, referring to 10 images each of 200KB. Redirection to the closest CDN server has already happened, and DNS and TCP connection establishment is also complete. The client can now start sending GET requests to the CDN server to fetch the content. The round trip latency between the client and the CDN server is 100ms (milliseconds). The bandwidth is limited at the client to 100KBps (1KBps = 10^3 bytes per second). What will be the page load time if persistent HTTP is used with pipelining? Without pipelining? [5]

With pipelining



- GET: RTT = 100 ms
- index.html : Download = $\frac{100\,KB}{100\,KBps}$ = 1s

- Multiple GET: RTT = 100ms
- img-1 : Download = $\frac{200\,KB}{100\,KBps}$ = 2s
- img-2 : ... 2000ms = 2s
- img-10 : ... 2000ms = 2s

300ms + 200ms x 10
= 21.2 sec. page load time

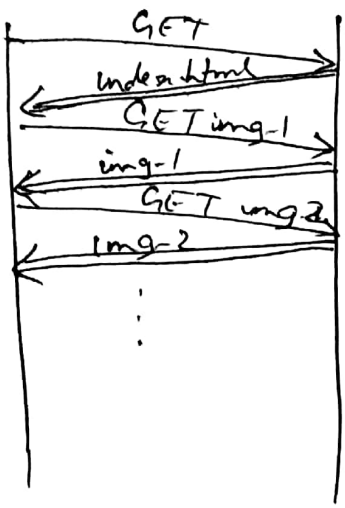
① for difference between pipelining & non-pipelining

② marks each for calculations

req-resp ① OK

① sum OK

Without pipelining:



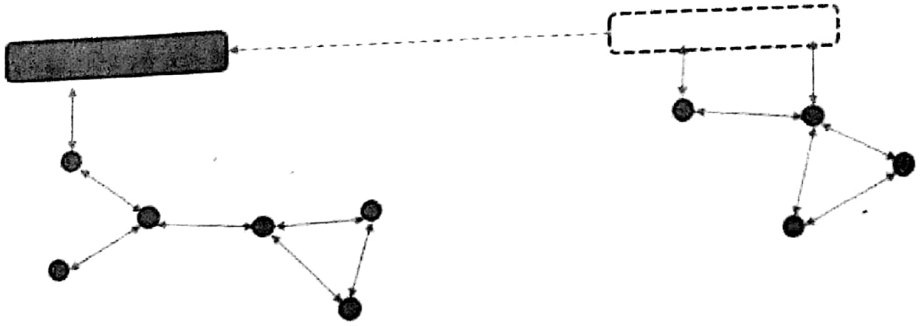
- GET: RTT = 100 ms
- index.html: Download = 1 sec
- GET img-1: 100 ms
- img-1 download = 2 sec
- GET img-2: 100 ms
- img-2 download = 2 sec

200 ms + 300 ms × 10

= 22.1 sec page load time

4. We want to set up a wireless sensor network, in the following setting:

   - Wireless sensors are dropped randomly on the ocean floor. All sensors are identical to each other in terms of their computation, storage, and networking capabilities.
   - Each sensor periodically monitors in its immediate environment the temperature and chemical concentration of different gases, it is assumed to know its (lat, long, depth) coordinates through an underwater GPS-equivalent system, and it has a unique 32-bit identifier assigned to identify itself.
   - Sensors within wireless range of each other can talk to each other and share data. The diagram below shows a sample layout with bi-directional links on which data transmission can happen between pairs of sensors.
   - An undersea capsule periodically makes its way over the ocean floor to collect data wirelessly from the sensors. It may not come in range of each and every sensor, and therefore the sensors should be able to send their data to neighbouring sensors so that the data can be picked up by the capsule from any sensors with which it may come in range. The diagram below shows the capsule traversing over two clusters of sensors to collect data. It is not necessary that the capsule will follow the exact same trajectory each time, ie. it could come in range of another set of sensors to pick up the data in its next traversal.
   - The process each sensor follows is that it periodically senses the environment, puts the data in a packet, and tries to then transmit the packet to nearby sensors in its range. The sensors that receive the packet, then try to transmit it to other neighbours, thereby broadcasting all packets everywhere in the cluster. Whenever the capsule comes in contact with any sensor in a cluster, it is therefore able to collect all the data produced in that cluster.



We want to design a network for such a setup. Answer the following questions:

a. Will a MAC protocol need to be in place to allow sensors to communicate with each other? Why or why not? [1]

(1) Yes, since the sensors could try and transmit at the same time, therefore a MAC protocol will be needed to help with coordination.

b. In the above setup where data produced by a sensor is forwarded in the entire network, how can we prevent a broadcast flood where the same packet could end up getting forwarded again and again forever in the network? Hint: You need to think of a way to uniquely distinguish each packet. [2]

(1) Each packet can be distinguished uniquely by (sensor identifier, time of transmisson or transmission counter or random identifier)

(1) Each sensor then keeps track of it has already forwarded the packet earlier, and does not forward again.

c. The sensors will have limited storage. To be able to effectively reuse the storage, once any packets have been picked up by the capsule they need not be retained on the sensors any more. Describe a protocol the sensors can follow to achieve this. [2]

Similar to packet broadcast, an ack-bcast protocol can be developed. (1)

For each packet picked up by the capsule, an ack can be broadcast to everybody to purge the packet from their storage. (1)

d. In a generic network with n sensors and e links, give an upper bound for the time a new packet produced by a sensor will take to propagate everywhere in the network? Assume that the (transmission + propagation) delay on each link is the same, d, and ignore any processing or queueing delays under the assumption that the rate at which a sensor monitors its environment and produces packets is very low. Explain your answer. [1]

Worst case if the sensors are laid out in a chain, assuming no partitions, it will take $(n-1)$ hops, and hence $(n-1).d$ time ..... (1)

e. Using your method of part (b), how many copies of a packet will need to be made for it to propagate everywhere in the network? Each time a sensor forwards a packet on one of its links, it is counted as a new copy. Explain your answer. [1]

Worst case the same packet could traverse twice on the same link, ~~but~~ since sensors will not forward again therefore the 2nd time or subsequently a sensor will not re-broadcast. Therefore $2e$ ..... (1)

f. The equivalent of congestion in this case is if the capsule gets delayed in its pickup and therefore a lot of packets get piled up at each sensor. Some of the packets will then need to be deleted. As an application programmer, what logic would you want the sensors to follow to decide which packet to drop? Keep in mind that the goal of the sensor network is to instrument as much of the ocean as possible. [2]

We need to select which packets to drop while preserving diversity. Therefore we will sort by # packets from each sensor & drop packets from the sensors with most packets. Further prioritization can be done on time.

D.. ①..

g. The equivalent of end-to-end reliability in this case is to ensure that all packets produced by a sensor are picked up by the capsule. Packets however could get dropped as described in part (f) above, and if you have a good solution for part (c) then the sensors will even get to know which of their packets successfully got collected by the capsule. Assume that sensors do get a notification for packets successfully picked up by the capsule. In that case, describe what kind of a logic the sensor could use to initiate a re-broadcast of the packet. [1]

Sensors could trigger re-transmission on a timeout, or if they got acks for packets sent after a particular packet but not for that packet itself.
That is, a logic similar to that used by TCP could be incorporated.

① -- either one is fine

h. We want to roughly calculate the storage space required at each sensor. Assume each packet to be of B bytes, at most N sensors in a cluster, rate of production of new packets per sensor of r, and average rate of visit of the capsule of R. Here, $r > R$, ie. rate at which the capsule visits the sensors is quite a bit lower than the rate at which new data is produced. What is the bare minimum storage required at each sensor? In practice, you would provision for at least 3-4x of the average rate. [2]

~~Rate of production = N.B.r~~

~~Pickup = N.B.R~~

# of production rounds between pickups

$= r/R$ ---- (1)

Data produced within each round

$= N.B.\frac{r}{R}$ ---- (1)

Should provision for 3-4x of that

------ Rough work ------