CS 538: Advanced Computer Networks Spring 2017

CS 538 Assignment 2

Assignment 2 Due: 11:00 am CT, Monday April 24, 2017

Submission instructions. This assignment is due at the time listed above. Send your submission via email to the instructor using:

- Subject: CS 538 Assignment 2
- Attachment format: PDF
- Attachment filename: *YourNetID*.pdf

Collaboration policy. You're encouraged to discuss the assignment and solution strategies with your classmates. However, your solution and submission must be written yourself, in your own words. Please see the policy on academic honesty and cheating stated in the course syllabus.

1. **[15 points]** Consider a shared 100 Mbps (that's mega*bits* per second) link. Several flows are sharing this link. There are four "elephant" flows that each want to download 4 GB (that's giga*byte*) files; these flows all start at 1:00 p.m. There are 100 "mouse" flows which want to download 250 KB each; one such flow arrives once every 10 seconds starting at 1:01 pm. (Incidentally, 250 KB is roughly the size of an average web page on the Internet.) Define the *delay* of the flow as its completion time minus its start time. Assume an idealized fair sharing model where if there are n concurrent flows, each flow gets $\frac{1}{n} \cdot 100$ Mbps (this could be implemented by an idealized steady-state TCP, or fair queueing). Assume for simplicity of calculation that all prefixes (giga-, mega-, kilo-) are powers of 1000.

   (a) What is the delay of the elephant and mouse flows?

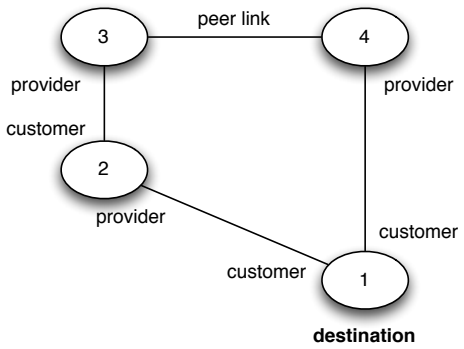
   (b) Now suppose mouse flows are always given priority over elephants (i.e., if any mice want to use the link, then none of the elephants get any bandwidth). In this case, what is the delay of the elephant and mouse flows?

   (c) Assume now a real-world network which does not have prioritization, but instead relies on TCP. Compared with the idealized fair sharing used in (a) above, give one reason that with TCP in the real world, the mice might have *higher* delay, and one reason the mice might have *lower* delay.

2. **[10 points]** Routes between A and B are *asymmetric* when the $A \rightsquigarrow B$ path is not the same as the $B \rightsquigarrow A$ path. How can asymmetric routing occur in the Internet, even if all ASes use BGP with common business relationship policies (prefer customer over peer over provider for route selection, and valley-free export)? Give two examples of how this could happen.

3. **[20 points]** We saw how BGP routing can be formulated as a game where the selfish players are autonomous systems (ASes), and we saw a case where this game has no Nash equilibrium (stable state). That is, the control plane will keep switching routes even though the physical network is stable. In this problem, we'll see an example where there are *two* equilibria, and different sequences of events could lead to one or the other.

   Consider the network below:



For simplicity, assume AS 1 is the only destination: it is the only AS that will originate an announcement message. The ASes have provider/customer/peer business relationships as shown. They follow the common route selection and export policies ... except that AS 1 is using AS 2 as a *backup provider*. This means AS 1 has instructed AS 2 to route to AS 1 via the link $2 \rightarrow 1$ only when no other path is available. (Incidentally, this is possible with BGP's community attribute.) The effect is that AS 2 prefers to route through AS 3 in order to reach AS 1. Assume that at time 0, no routes have been announced by anyone. Shortly after time 0, AS 1 will begin announcing its IP prefix.

(a) Describe a sequence of events (BGP announcement messages and path selection decisions) that lead to one stable state.

(b) Describe a different sequence of events that lead to a *different* stable state.

(c) Suppose the network is now stabilized in state (a). A link fails; the BGP routers reconverge; the link recovers; BGP reconverges again; but now the network is in state (b) instead of state (a)! (This problem is sometimes known as a "BGP wedgie" because the system has gotten stuck in a bad state.) What sequence of events causes this story to happen? That is, which link failed, and which messages get sent?

4. **[20 points]** This question deals with the B4 paper which we read for March 1.

(a) Describe two possible failures and how B4 protects itself from each type of failure.

(b) In the original OpenFlow paper, a centralized controller received the first packet of each flow and installed forwarding rules to handle that flow. However, a significant concern for any centralized design is scalability. Describe two features of the B4 SDN design which allow it to scale to a large number of network devices and flows.

5. **[20 points]** At several points this semester, we've seen examples of network systems which put intelligence (i.e., functionality) at the edge of the network (for some definition of "edge"), which lets core routers be simpler. This is essentially an application of the end-to-end principle.

The following questions ask for more examples of this approach of placing intelligence at the edges. For each example, describe three specific items: (1) what the "edge" means in that example, (2) what functionality resides only at the edge, and (3) how that design choice simplifies or benefits the rest of the system.

(a) Describe how this approach is applied in the classic Internet architecture. (One example is enough.)

(b) Describe how this approach is applied in either MPLS traffic engineering or TeXCP (discussed Feb 20).

(c) Describe how this approach is applied in the Fabric paper.

(d) Describe how this approach is applied in either NVP or VL2 (discussed March 6).

6. **[15 points]** Short questions.

(a) Consider a cluster of database servers. The time taken for any server to respond to any request is 10 milliseconds 97% of the time, and 200 milliseconds 3% of the time. Assume these times are sampled independently at random for each request. If a front-end web server queries 20 database servers for a user request, what is the probability that the web server needs to wait $\geq$ 200 milliseconds for its requests to complete? Ignore processing delays at the front-end web server. Briefly show how you calculated the answer.

(b) In the setting of the previous question, if the web server needs to query 100 database servers instead of 20, what is the probability that the web server needs to wait $\geq$ 200 milliseconds for its requests to complete?

(c) Suppose we build a fat tree data center topology using 24-port switches, following the design of the Al-Fares paper (discussed April 5), and using ECMP routing within the topology. How many distinct end-to-end paths are there between a source server A and a destination server B in a different pod?

(d) In the setting of the previous question, let C be one of the core switches. How many distinct paths are there from $A \rightarrow C$?

(e) In the setting of the previous question, how many distinct paths are there from $C \rightarrow B$?