CS 538: Advanced Computer Networks Spring 2018

CS 538 Assignment 2

Assignment 2 Due: 11:00 am CT, Friday April 20, 2018

Submission instructions. This assignment is due at the time listed above. In your submission, explain your answer (e.g., an answer with a numerical result should show how you derived it). 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 5 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 5 seconds starting at 1:01 pm. (Incidentally, 250 KB is roughly the size of an average web page on the Internet.) Define the *flow completion time* (FCT) of a 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). Note: Here, the prefixes giga-, mega-, kilo- are powers of 1000; i.e., 100 Mbps $= 100 \cdot 10^6$ bits per second, 5 GB $= 5 \cdot 10^9$ bytes.

   (a) What is the FCT of the elephant flows? What's the FCT of the 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 FCT of the elephants, and what's the FCT of the mice?

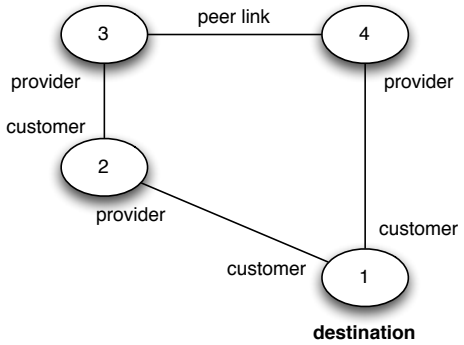
   (c) Assume now a real-world network which does not have prioritization, but instead relies on TCP. Compared with the idealized fair sharing used in (a) above, give one reason that with TCP in the real world, the mice might have *higher* FCT, and one reason the mice might have *lower* FCT.

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 discussed on February 26.

(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]** Suppose we build a cluster with a fat tree topology using 24-port switches, following the topology design of the Al-Fares paper (discussed April 4).

(a) How many distinct end-to-end shortest paths are there in the physical topology between a source server A and a destination server B in a different pod? (Here, shortest paths are those that have the minimum number of switches along the path. This question concerns the physical topology, not routing or forwarding.)

(b) Suppose the cluster runs BGP, with each switch having its own AS number and each edge switch announcing a distinct attached IP prefix covering all its servers. (No other prefixes are announced.) Switches run standard IP forwarding with ECMP across all shortest paths. How many forwarding rules are in each edge switch? How many are in each core switch? (Each forwarding rule specifies a single egress interface for a set of matching packets.)

(c) Suppose the switch hardware is capable of performing MPLS forwarding, with IP-in-MPLS encapsulation, and the hardware can also still perform IP forwarding. More specifically, a forwarding rule on a switch can either (1) perform IP/ECMP forwarding as above, (2) match a prefix of IP packets and direct it into an MPLS tunnel, with ECMP-style hashing if there are multiple matching rules; (3) perform MPLS label swap forwarding; or (4) match an MPLS label, pop the MPLS header and continue forwarding out an interface via IP. Can you use this hardware to deliver data along the same paths as ECMP, but with fewer forwarding rules? If so, describe your solution and how many forwarding rules it needs at each edge switch and at each core switch.

(d) Compare the resilience of your solution with that of ECMP. Specifically, suppose one core switch fails. What plausibly happens within about 1 millisecond, i.e., with only local reaction at each switch, in ECMP and in your solution? (Note: the goal isn't to produce a scheme which is necessarily better or worse than ECMP; the goal is just to compare.)

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

(a) When is a (transport-layer) port number used for the same purpose that an IP address was originally intended? When is an IP address used for the same purpose that a port number was intended? Give an example of each.

(b) Consider a cluster of database servers. The time taken for any server to respond to any request is 10 milliseconds 99.8% of the time, and 200 milliseconds 0.2% of the time. Assume these times are sampled independently at random for each request. We have a front end web server that, when a user request arrives, queries 100 database servers in parallel. When all responses are received, it can reply to the client. The web server itself is incredibly fast, so all its local processing always takes less than 1 millisecond. What is the probability that the web server needs ≥ 200 milliseconds to be ready to reply to the client? (As with all answers, briefly show how you calculated the answer.)

(c) In the setting of the previous question, suppose the web server needs to perform *two* phases of processing before replying to the client. Phase 1 queries 100 database servers as described above. But now, after receiving all responses from phase 1, it can begin the second phase, where it queries another 200 database servers in parallel. Finally, when all responses are received, it can reply to the client. What is the probability that the web server needs to wait ≥ 200 milliseconds for its requests to complete?