CIS 553: Networked Systems April 21, 2020

Problem Set 2 - Solutions

Instructor: Prof. Vincent Liu *Department of CIS, University of Pennsylvania*

Turn in your solutions in on **April 27, 2020** by 10:00 p.m. via Canvas. Grading will be lenient, as long as you give an honest effort to answer every question. **Please work alone** and do not discuss answers with other students.

Note that while all of these questions are taken from prior exams, you should NOT assume this is what the exam will look like. For one thing, there are topics here that will not be on the exam, and topics on the exam that are not included here. Instead, treat this as an opportunity to review some of the material and debug any basic misunderstandings about the basic ideas covered so far.

1. True or False

   (a) The 5-tuple that identifies distinct flows is usually defined as (src IP, dst IP, src port, dst port, protocol #).

   T

   (b) One of the primary reasons why TCP is used for video streaming is that many firewalls block UDP packets.

   T

   (c) The Pakistan hijacking case occurred because Pakistans ISP spoofed low-hop advertisements to remote ISPs.

   F

   (d) Streaming video is video that can begin playout before downloading the entire file.

   T

   (e) AIMD's Multiplicative Decrease is a form of BEB.

   T (but debatable)

   (f) Explicit proxies are proxies that act on behalf of a few servers, but handle all clients.

   F

   (g) In wireless networks, multipath effects cause self-interference.

   T

   (h) Users in censoring countries can defend against TCP RST injection by modifying their laptop network configurations.

   F

(i) One of the enablers of DDoS attacks is the fact that the Internet implements best-effort routing.

T

2. Imagine that you are watching streaming video from `unoptimizedvideo.com` over HTTP/1.0 and are experiencing video playback stuttering. For each of the following optimizations:

   i. Do you expect the technique to improve the smoothness of video playback? If so, how? If not, why not?
   ii. Do you expect the technique to be more effective on video than a typical webpage? Why or why not?

   Note: For each of the following, a case could be made for other answers. The reasoning is the most important part of this question.

   (a) Pipelined/persistent connections

   (i) No or very little. Videos are typically large in size, and therefore bottlenecked by raw bandwidth. Pipelining can reduce a few RTTs (˜50−100ms per 6 seconds in the case of Netflix) and persistence can eliminate unnecessary slow starts, but overall, the effect will likely be small.

   (ii) No, webpages are typically composed of many smaller files, so they would benefit more from the pipelining and persistence.

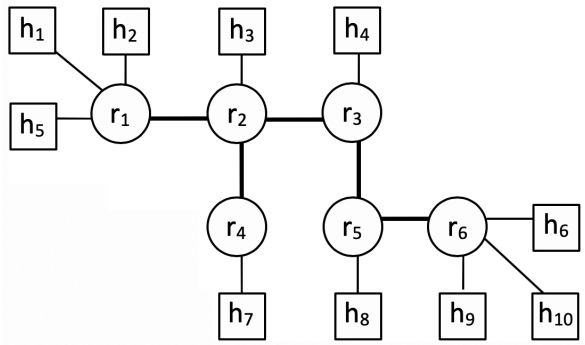
   (b) CDNs

   (i) Yes, CDNs can alleviate bottlenecks and congestion that can occur in the core of the Internet and near the servers. They also reduce RTT, which allows TCP to ramp up faster and take more bandwidth.

   (ii) In terms of bandwidth from the RTT advantage of CDNs, video fares better as it is typically bandwidth limited. In terms of the benefits to jitter, videos also benefit more than web traffic as users don't directly perceive that in web pages. In terms of the benefits to latency, webpages benefit more as PLT is sensitive to RTTs and the sequential download of dependencies.

   (c) HTTP/2

   (i) No, in fact it might be worse as HTTP/2 pushes for a single connection instead of m, which would lead to a factor of m decrease in bandwidth. Server push would not help either as the client is purposely in charge of fetching all chunks.
   (ii) No, see above.



3. Consider the network above. Each host-to-router link has capacity 100 Mb/s and each router-to-router link has capacity 200 Mb/s.

   Given the following concurrent TCP connections (src→dst; demand), what are the steady-state max-min fair share allocations of each of them on the given network? What is the bottleneck link of the connection (e.g., $r_5 \rightarrow r_6$)?

   (a) $h_{10} \rightarrow h_5$; 80 Mb/s

   75 Mb/s; $r_5 \rightarrow r_3$

   (b) $h_8 \rightarrow h_3$; 50 Mb/s

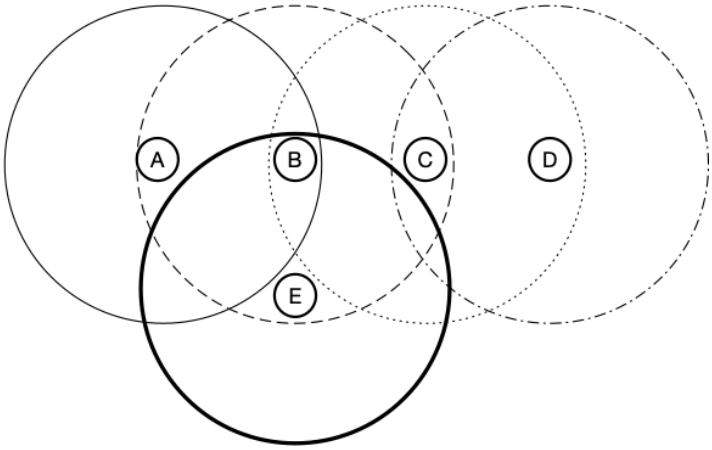
   50 Mb/s; none

   (c) $h_9 \rightarrow h_1$; 200 Mb/s

   75 Mb/s; $r_5 \rightarrow r_3$

   (d) $h_7 \rightarrow h_2$; 40 Mb/s

   40 Mb/s; none



4. For the following questions, consider the wireless topology above, comprised of 5 nodes: A, B, C, D, and E. All of them have an effective transmission range as shown. For example, D can only speak to C (C is the only node in its circle), and can also only hear from C (D is in Cs circle and no others).

(a) Why are acknowledgements used in 802.11, but not in wired Ethernet?

In wired Ethernet, we know when collisions occur because we can see them. In Wi−Fi, we can't see them, and thus we need another mechanism. Detecting collisions is important because it helps us estimate how many other nodes are contending for the medium (through BEB). Otherwise, we would need a fixed backoff period that would likely be too long (inefficient) or too short (high chance of collisions and therefore inefficient).

(b) Assume A is about to transmit to B. Which terminals transmissions can cause hidden terminal problems? List in terms of src→dst. What about exposed terminal problems?

hidden: E→B, C→B
exposed: none

(c) Repeat (a) for B→C. Which terminals transmissions can cause hidden or exposed terminal problems?

hidden: D→C
exposed: none

5. Fair Queuing

(a) Why does even bit-level fair queuing fail to provide max-min fairness in some scenarios?

While bit−level fair queuing provides max−min fairness at the level of a single router, this doesn't generalize to a network of fair−queuing routers. See Lecture 17, Slide 11 for an example of this.

(b) Using python-like pseudocode, write an implementation of packet-level fair queuing. You can base your implementation around two functions: `recv(packet, size, flowID)`, which is called on every new packet arrival, and `send()`, which is called whenever the output port is idle. `send()` should return the next packet to send or NULL if there are no packets queued. You can define any global state you need.

```
// Assuming a single output port

// One queue per ingress.
// Each element on the queues are of the form:
//  <packet, finish time in a bit−level FQ scheme
perIngQueues[numPorts]

recv(packet, size, flowID):
        ingressPort = fowardingTable[packet.dstAddr]
        virtualNow = max(now(), queues.back.second)
        virtualFinish = virtualNow + len(packet)/BW
        perIngQueues[numPorts].enqueue([packet, virtualFinish])

send():
        packet = NULL
        minFinish = −1
```

```
for q in perIngQueues:
        if minFinish == -1 or q.front.second < minFinish:
                minFinish = q.front.second
                packet = q.front.first

return packet
```