Pre-Midterm

0. Recall: RTT, bandwidth-delay product, encoding, forward ECC
1. Review all the quizzes so far. (Will be handed back in section.)

Security

0. What's the difference between symmetric and asymmetric key encryption?
Symmetric uses the same key to encrypt and decrypt, asymmetric key uses two different keys (usually a private and public key).
1. How does a digital signature ensure message **authenticity**? Message **integrity**?
Message authenticity- the sender is who you think it is.
Message integrity- the message was not tampered with.
The signature, which is provided with the original message, is the message cryptographically hashed (via a one-way function) and then encrypted (or similar*) with the private key. To verify the message you decrypt (or similar*) the signature with the public key and compare it to your version of the same cryptographic hash on the same message. The cryptographic hash and PKI encryption verify integrity. The PKI encryption + certification verify authenticity.
You're trusting: the CA where you got that public key, and your implementation of the hash function and other software in this process
*There are explanations of why signing is not actually the same as encryption/decryption in practice, but they're beyond scope for now.
2. What's the difference between a Message Authentication Code (MAC) and a signature?
MACs use symmetric keys (technically, even if one is stuck on a hardware key) and signatures use asymmetric keys.
3. What security protocols could fail if I decide to trust a malicious certificate authority? How could those attacks occur?
If a malicious certificate authority (trusted by your browser) gave a malicious server a fake certificate, if you visited a site on that server, TLS would allow that server to authenticate and would fail to protect you.

TCP & TCP Congestion Control

1. There are many many variants of TCP. Review TCP Tahoe, Reno, New Reno, & SACK. (See lecture slides on the Transport Layer.)
TCP Tahoe- immediate slow start upon packet loss (both 3 duplicate acks and timeout)
TCP Reno- "fast recovery" back to AI with half cwnd upon packet loss
TCP New Reno- improved "fast recovery" with window-refilling by sending a single new packet upon duplicate ack, "hole"-filling upon partial-progress ack that indicates another single packet loss within the sent window
TCP SACK- "selective ack" used to specify blocks of packets that were received correctly in addition to normal sequence number
2. What are:

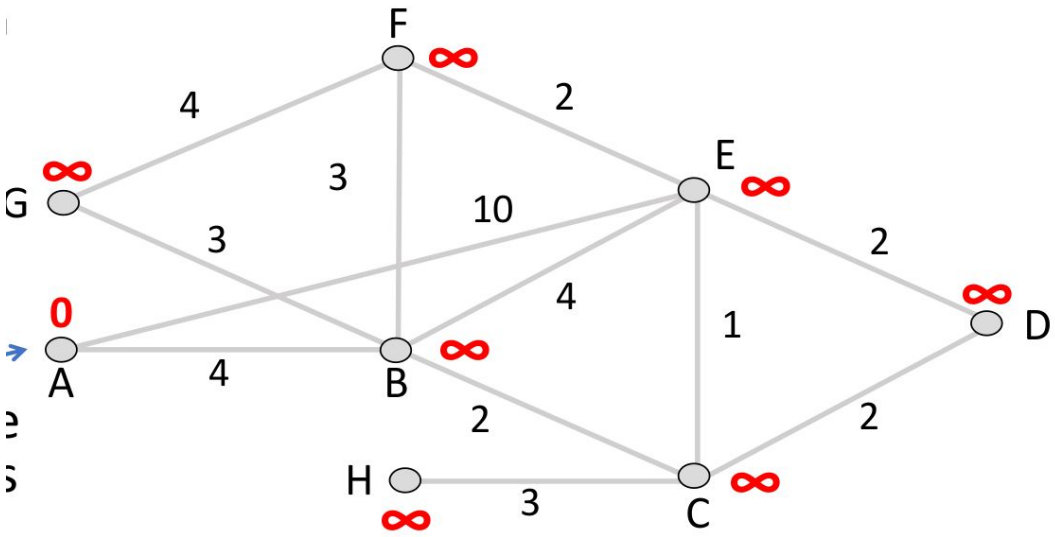
-Slow start?
In section
-Fast recovery?
Straight to AIMD with ssthresh set to half cwnd, instead of all the way back to slow start.
-The indicators of a lost packet?
Duplicate acks and timeouts
-AIMD?
In section
3. What is ECN?
Explicit congestion notification. In TCP, ECN is implemented through 3 header flags (plus some negotiation during connection setup). Not all TCP implementations support ECN but most do now.
4. What is Random Early Detection? (Also Random Early Discard/Drop)
When a router sees its input queue filling up (indicating congestion), it can choose to randomly (for fairness and other statistical properties) drop packets from the queue without sending them, so that the senders of those packets will slow their transmission rates in the near future.

Routing Protocols (Link State vs Distance Vector)

0. Review the Link State and Distance Vector sections from the lecture slides.

1. Big idea
Link State Routing:
a. Suppose the network is using link state routing. Briefly describe the link state update packets, the update process, and what each node knows about its network topology after a (fully converged) link state update.
Link state: Update packets from each node contain only the weights to the node's direct neighbors. The update process is by flooding all update packets to all other nodes. After the link state update has converged, each node knows the full network topology including weights and can run an algorithm such as Dijkstra's to find its own shortest path tree.
Distance Vector Routing:
b. Suppose the network is using distance vector routing. Briefly describe the distance vector update packets, the update process, and what each node knows about its network topology after a distance vector update (multiple exchanges until convergence).
Distance Vector: Update packets from each node contain a vector of full distances to all other nodes. They are sent only to the node's direct neighbors. When using "split horizon, poison reverse" to prevent loops, the vector sent to different neighbors may differ. For example, the vector sent from node A to B will have a distance value of infinity for the path from A to B, and a path from A to C that is only reachable through B. After routes have converged, each node does not know the full topology but only its distance to each other node and the next hop (the bare minimum needed for routing).

Consider the following network graph from the slides.



Back to **Link State.**

2. Run Dijkstra's algorithm on the above network topology from the source node G, showing all iterations.

In Section

3. Imagine that due to some bug (floating point precision errors? etc), some edge weights are recorded as negative. Would Dijkstra's algorithm still work? Why or why not?

No, because negative weight edges destroy the property that ensures Dijkstra's correctness- that the shortest path to a given node is made up of the shortest paths to intermediate nodes. Consider the example:

```
   -100
 A ----- C
 1 \     / 1
       B
```

Strangely, the shortest path from B to A is B-C-A and the shortest path from B to C is B-A-C (disregarding any paths involving infinite ping-ponging between A and C).

4. Dealing with failure:

a. Now suppose an arbitrary network with average degree d, node number |V| and edge number |E|. A node goes down, and this triggers a link state update. (Note: it does not have to work this way- link state updates can also follow a schedule.) In big O notation, what is the approximate number of messages that get sent out as a result of this update? O(d*|E|)

b. If a node is "flapping" (coming up and going down rapidly) at a rate of once per 30 seconds, give the approximate network bandwidth the updates take up in terms of update message size U. (U*d*|E|)/30 bits/sec

c. What would happen if a node claimed to have a path weight of 0 to all of its neighbors?

Redundant link state packets from other nodes in the same flooding round would contradict this info and allow the malicious node to be detected.

d. What would happen if an update packet was extremely delayed and arrived late? (**stale**, as opposed to **fresh**)

Link state protocols use sequence numbers to ensure freshness of link state packets, so a stale packet with an old sequence number would be rejected.

Now **Distance Vector**.

5. Suppose an arbitrary network with average degree d, node number |V| and edge number |E|. In big O notation, how many messages get sent out as a result of a single distance vector exchange? O(|E|)

6. Dealing with failure:

a. Suppose link G-B in the given network graph fails and this triggers an update. How many rounds of distance vector exchange until D's route to G converges? Assume use of "Split Horizon, Poison Reverse," a max hop count of 15, and "hold time" of 10x a single exchange time.
In section

b. What would happen if a node claimed to have a path weight of 0 to all of its neighbors?
A large amount of traffic (though perhaps not all, for a well-connected graph) would be redirected to the malicious node. With no additional security features, distance-vector protocols are vulnerable to malicious rumor-spreading due to lack of confirmation of topology between nodes, as done via flooding in link-state protocols.

c. What would happen if an exchange packet was extremely delayed and arrived late?
Some distance vector protocols such as DSDV use sequence numbers for freshness, but if not, the late packet could provide stale information and perturb the routing tables until reconvergence. Since distance-vector exchanges only occur between directly connected nodes, long delays are less likely than in link-state routing, but still possible in theory.

HTTP

1. What do the GET and CONNECT methods do?
Two of the most used HTTP requests - GET retrieves information from the server, CONNECT establishes a tunnel between the requestor and server
2. What are the steps for an HTTP request to a server?
    a. Resolve the server IP
    b. Setup TCP connection (port 443)
    c. SSL/TLS negotiation and key exchange
    d. Send Encrypted messages
    e. Teardown connection
3. How does DNS lookup work?
After querying your local name server, it will begin resolving the IP by starting from the outermost layers. For example: for cs.washington.edu, it will begin by querying the root

name server, we'll get an answer for edu, then we'll query the edu server, for which it will give the washington server, and so on until we have resolved the entire URL for an IP

BGP

1. What are the two relationships that define the protocol
   Peers and Transit (Customer)
2. What can you announce and to whom (depending on the relationship)?
   ISP's announce everything it can reach to its customer
   Customers only announce their customers to their provider
   ISP announce customers to peers
3. ISP's never announce peer relationships to other peers, why is this the case?
   Never route 4 free

Misc. Protocols

1. What is DNS poisoning? How does it work? (This question is more here for general interest than exam prep- you won't need to know quite so much detail for the exam.)

You have a malicious/owned DNS server and malicious web domain, and you're trying to redirect traffic from a target web domain to an IP address of your choosing.

Option 1: When someone (eg another DNS server) requests the malicious web domain, the malicious DNS server can send a response that sets the malicious web domain's authority to be the target domain's name server, and then in the "Additional" section, "glue" the IP address of the target name server to a malicious IP address. The requesting DNS server caches the target name server and its supposed IP, so when it gets requests for the target domain, those requests will go to the malicious DNS server at that IP. The requesting DNS server has been poisoned.

Option 2: When someone (eg another DNS server) requests the malicious web domain, the malicious DNS server can send a response that sets the target domain's authority to be the malicious domain's name server, and then in the "Additional" section, "glue" the IP address of the malicious domain's name server to a malicious IP address. The requesting DNS server caches the target domain's authority as the malicious DNS server, so when it gets requests for the target domain, those requests will go to the malicious DNS server at that IP. The requesting DNS server has been poisoned.

These can both be mitigated via TLS and certificates.