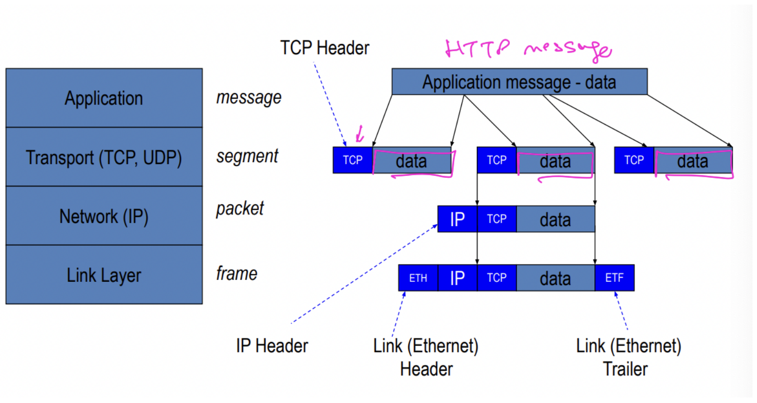
CAS CS 357

InClass Note 19

1. Data Formats

    a. 

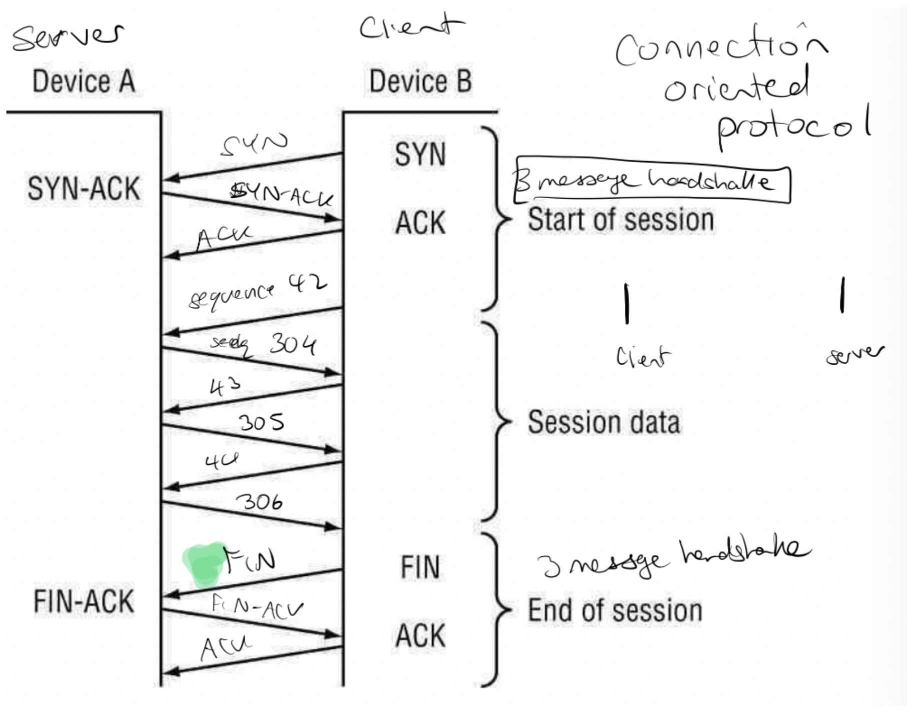
    b. TCP is the protocol that connects the client to server and makes sure that traffic arrives in the correct order and without being lost (controls flow of traffic)

    c. TCP has acknowledgement number and sequence number in the same packet → when client responds back, the acknowledgement number for the packet (also includes sequence number) is included

    d. Source IP, destination IP → server where the packet is being sent, server where the packet is received

    e. NAT is changing the source IP when it is sending to the public internet over the port (the NAT is going to translate the source IP and use the port number to disambiguate between the private IPs in the same system)

2. TCP

a. 

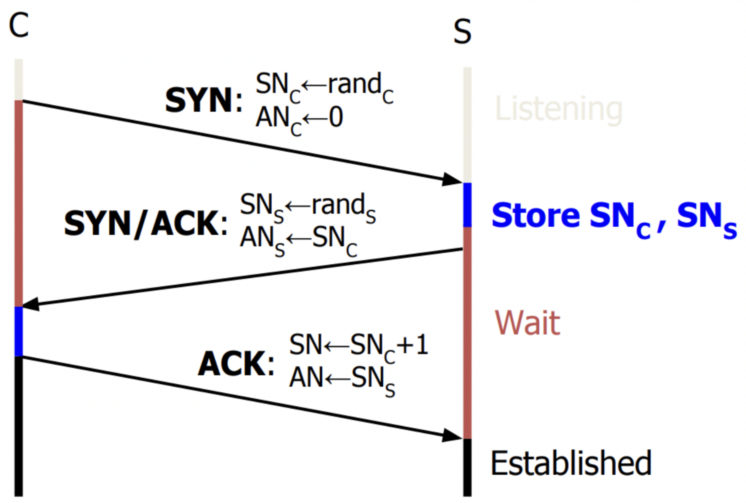
b. TCP handshake → SYN, SYN-ACK, ACK section

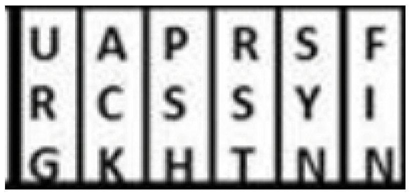
c. TLS handshake → client hello, server hello, g^x, g^y, MAC, etc.

d. TLS record layer – data

e. TCP handshake → FIN, FIN-ACK, ACK

3. TCP Handshake

a. 

b. 

c. Client sends a SYN/ACK packet → SYN/ACK bit in the TCP/IP packet (the SYN/ACK bit is set to one), choose a sequence number for a packet (random)

d. The server sees the sequence number sent by the client and responses sequence number chosen randomly and ACK

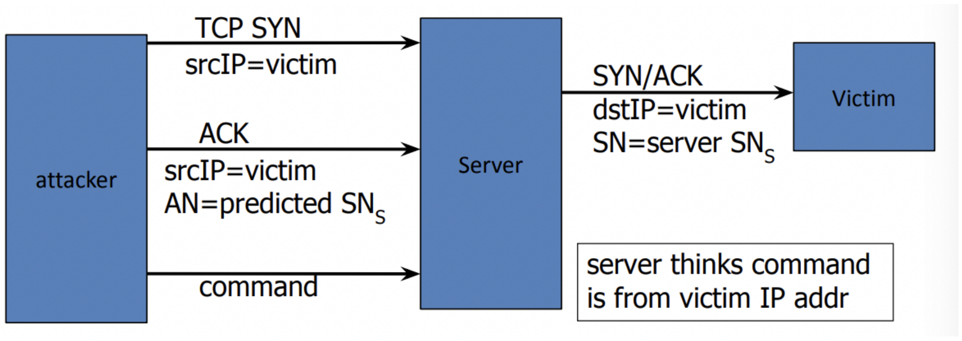
e. Client proves to the server that it knows the two random numbers generated

f. ex) randc = 42, rands = 343 → SNc = 42, SNs = 343

g. Off-path Adversary cannot know the numbers that are generated randomly and therefore prevents attackers from injecting packets (provides some kind of protection → you need to actually be on the path to do so since the packets with wrong sequence number and acknowledgement number will be ignored)

h. TCP → gives prevention from Off-path adversary from adding packets

4. If the numbers are not random

a. 

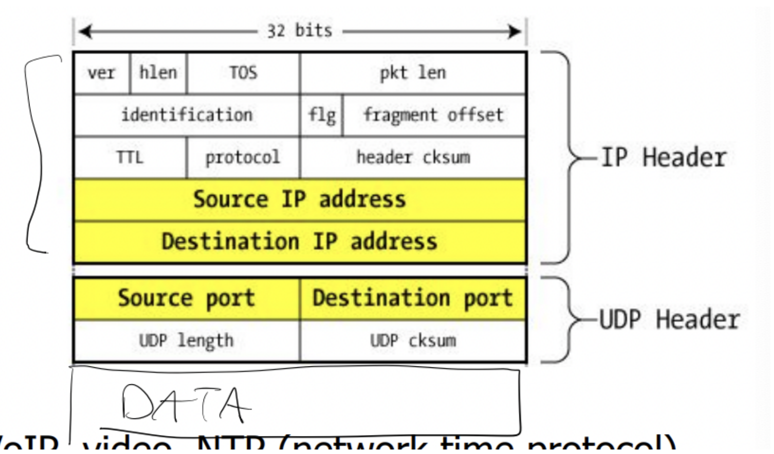
b. Spoofing an IP = altering source IP on a packet (not part of a legit NAT)

c. Attacker (spoofing a source IP on a packet) → attacker pretend that the packet came from the victim by srcIP=victim

d. Server gets the SYN/ACK to victim and if the adversary guesses the server SNs (the acknowledgement number and the sequence number) correct, the adversary can connect to the server as if he/she is the victim (if the numbers are random, such attack can be prevented → connection cannot be made to the server if the numbers are random and therefore adversary cannot guess the two numbers)

e. Connection exhaustion attack → when adversary sends the SYN to server, the server responds with SYN/ACK → the server stores the state and waits until the server receives ACK from the client (however, it will never come since the adversary will never get the SYN/ACK since numbers are generated randomly) → the memory and wait time continues to pile to the extent that server collapses → rate limiting occurs since the server is not accepting any more SYNs due to the vast number of already received connections that did not end

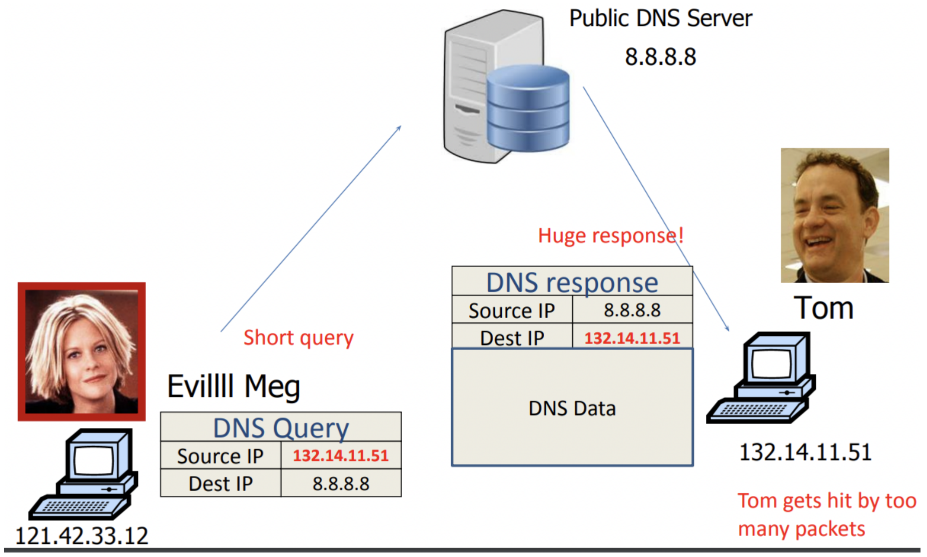
5. User Datagram Protocol (UDP)



a. VoIP, video, NTP (network time protocol)

b. UDP does not guarantee that packets arrive in order and no guarantee that every packet will arrive (get grainy, get gaps but do not care since you don't want latency) → loose packets that can arrive at any order

c. Spoof UDP (no sequence number, acknowledgement number) → can send as many UDP packets as you want, wherever you want

d. 

e. Public server → server not behind a NAT (server that is exposed to the Internet)

f. Denial of service attack → The adversary (Evil Meg) wants Tom's computer to go down → she sends a packet that has the source IP as Tom (132.14.11.51) and send the packet to the public server → the public server will return back the response to Tom → Tom gets hit by too many packets → Tom falls down (shuts down) or Tom would not be able to get actual real data that comes from the Internet due to the excess traffic on the wire

g. What makes it worse is that the response is bigger than the query

h. For example, if the data is 7 times more information than the query, Tom receives 7 times more data (7 times more traffic)

i. This is DDos Attack (Distributed Denial of Service) Attack

j. There are companies (such as Cloud Fare) that sell DDos protection (filter the queries) → most servers are not vulnerable due to many DDos protection

6. DNS

a. DNS → run DNS query to understand what the IP address is for the url of the domain that I need to visit

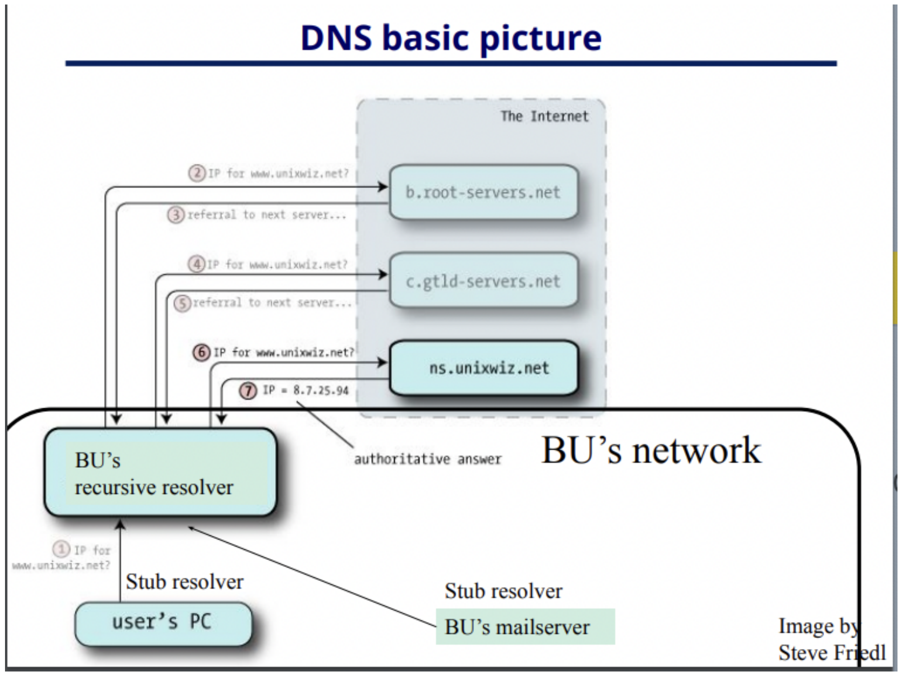
b. When you type google as your url to visit, you are making an HTTP query to google → get HTTP response (broken up into pieces and get TCP and IP header) → receive them and reorder them to produce them in the right order and send them back to the browser

c. What happens before you do any of this is that the operating system will send a DNS query → DNS server asks the IP of google.com → get back the IP → connection is made here (I know my IP and google's IP is known to public)

d. DNS → not encrypted (you can know that somebody visited the bank website through DNS)

e. TLS → protect actual data (sending money from bank → certificate, log in)

7. DNS Basic Picture

a. 

b. When you make query → go to recursive resolver (don't actually go to internet right away), reason it exists is that we can select all the queries from everyone (remembers the mapping of domains to IPs)→ the recursive resolver will cache information received from authoritative name servers. When a client requests the

IP address of a domain name that was recently requested by another client, the resolver can circumvent the process of communicating with the nameservers, and just deliver the client the requested record from its cache.

c. There exists three servers above → root server, gtld-servers, and unixwiz
d. Hierarchy → when there is website as bu.edu → root → edu (top level domain → .edu, .com, .cn, etc) → bu.edu (second level domain → bu.edu, google.com, etc)
e. DNS → start at the very top of the hierarchy with a query → go to the root → find (.net, etc) → return the IP address for requested hostname back to DNS Recursor (bu.edu)
f. Root servers → ones that know how to find the name servers for all the top level domains (.edu will know how to tell you how to find name servers for bu.edu) (whole point of root server is to tell you how to find the .net)
g. Top domain server's job is to know the address of the name servers that are second level domain servers