Assignment 1 for CNT 4700 Computer Networks: Internet Architecture & Application Layer
Leeson Chen - September 24th

Instructions: Be precise and to the point. Many questions require answers using a sentence or two). Some questions will ask you to elaborate, use visual aids or graphs, or show traces/code. Use your own words and phrases, do not copy from any other source.

Q1.

A. What is a network protocol?

A network protocol defines the behavior of messages sent via the internet. According to the lecture, a network protocol defines the following:

- format
- order of messages sent / received among network entitites (e.g. ISPs)
- action taken on message transmission

Protocols govern all communication sent through the Internet, whether it be streaming, VoIP, uploading or downloading. One example of an Internet protocol is TCP (transmission control protocol), which is the protocol through which hosts and servers establish connections to send and receive packets. The maintenance and upkeep of network protocols, as well as the creation of new protocols, is an area of important research between oversight organizations.

B. What is the body that standardizes the Internet protocols?

Internet protocols are standardized by the organizations IETF (Internet Engineering Task Force) and RFC (Request for Comments). This effort is also supplemented by the IEEE (Institute of Electronics and Electronics Engineers). Additionally, the organization ICANN (Internet Corporation for Assigned Names and Numbers, formerly IANA) is in charge of allocating IP addresses. The IAB (Internet Architecture Board) provides oversight for protocol architecture and other Internet procedures.

C. In which language are the protocol specifications largely written? What is the main problem with that format?

Protocol specifications are largely written in English. However, English has the drawback of being an indirect, descriptive language that often leaves space for exact details to be interpreted differently by different readers. This leads to the need for "legalese" language so that important points are not left vague and interpretable.

D. [Extra:] Write a simple protocol spec for the following task: consider two people playing a game where one has a secret number in 1 to 10 and the other is attempting to guess it. But the two people can only communicate by each alternatively saying a single number.

We will first define the two people participating in the game as following: the Guesser is the player who attempts to guess the secret number; the Holder is the player who thinks of the secret number. The Guesser will ask the Holder numbers from 1 to 10, and the Holder responds to the Guesser.

The difficult part of this game is how both Guesser and Holder are limited to only speaking in single numbers. While this is easy to integrate into the Guesser's role (as they only guess

numbers 1-10), it becomes more difficult for the Holder, who cannot explicitly state whether the Guesser's most recent guess was correct or not.
The simplest way to solve this problem is to assign right / wrong values to two numbers (e.g., if the Holder responds with 10 the guess was right; if the Holder responds 1 the guess was wrong.) However, because it was never explicitly stated that the Holder has to respond within the 1-10 range, it can return the negative guess if wrong, and positive guess if right.

Q2.

A. What are the basic paradigms (or models) of communication? (mention 3)

The basic paradigms / models of Internet communication are the client-server model, the peer-to-peer model (P2P), and the hybrid architecture which combines the two. In the client-server model, a client connects to an always-on host server and requests data. In the peer-to-peer model, peers link to neighboring users forming a web of connected peers that act as both clients (when downloading) and servers (when sharing). In the hybrid architecture, peers may intermittently connect to a centralized server which is used for indexing other peers and looking up the location of certain data in the network.

B. What are the processes are needed to support all these paradigms?

In order to support the above-mentioned architectures, the network uses TCP (for web / HTTP, file transfer / FTP, email / SMTP) and UDP (streaming media, teleconferencing, DNS).

Q3.

What are the advantages and disadvantages of:

A. Hierarchical network architectures

The Internet is a hierarchal network of networks, wherein many different users (hosts) connect to an access net, which then connects to a regional network, to regional ISPs then to global / tier 1 ISPs and content provider networks. At the top level, examples of tier 1 ISPs would be commercial ISPs that provide international coverage such as Sprint and AT&T, and content provider networks are large corporations that can afford the physical infrastructure to bypass tier 1 and regional ISPs such as Google and Amazon.
In this hierarchal architecture, a packet has to pass up from the access network, local ISP, Tier 3, Tier 2, Tier 1 ISP, then down again the reverse order to get to the other server. To navigate, the packet must engage in routing, which may be sub optimal at times, and necessitates other infrastructure. The hierarchy requires multiple "working parts" to be established and then maintained, operated by different organizations and companies that depend on the others to be working properly. However, despite these downsides, there are many advantages to this system: it is highly scalable and flexible, since the further to the end you get, the more customized organizations can make their network, and attach themselves to the larger global ISPs. When something does go down, the problem can often be isolated to a weak link within the hierarchy, preventing the error from propagating outward and shutting down multiple other systems, the way a single ISP architecture might.

B. Protocol layering

Protocol layering means that, in the Internet protocol stack, each layer (application / transport / network / link / physical) receives its own separate protocols, as opposed to consolidated

protocols that perform functionality at multiple layers. For example, the application layer uses FTP (file transfer), SMTP (email), and HTTP (web browsing); the transport layer uses TCP and UDP; and the network layer uses IP and routing protocols. This is helpful for modularity, such that changes at one layer don't affect others, so problems can be isolated. However, maintaining the interaction between these layers requires additional overhead.

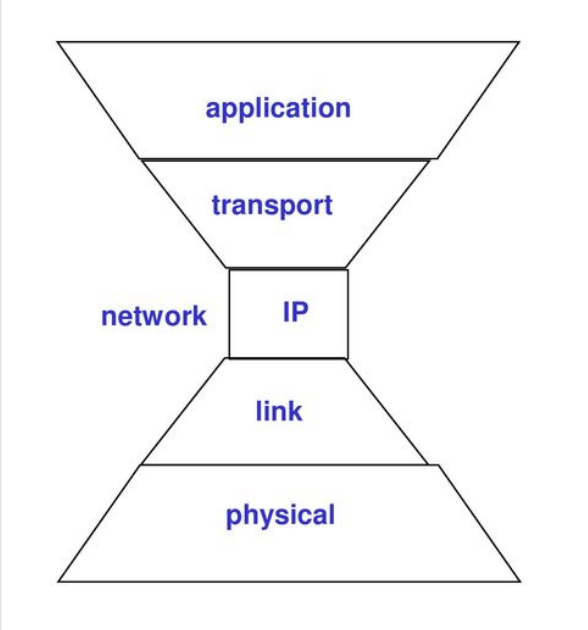
C. Stateless protocols

Stateless protocols do not remember the state, or the program's internal data relative to time and past changes. An example would be pure HTTP, which does not remember the user's past browsing patterns on the server side. These stateless protocols are easy to write and maintain, because they do not require additional memory space for past states, or algorithms for tracking such data. There is less of a security risk for inconsistency and reconciliation, and users are less trackable. Disadvantages of using stateless protocols include the loss of useful data, as well as the ability to optimize the reduction of sending redundant data.

Q4.

A. What is the *hour-glass* Internet model? (use simple drawing to illustrate)

The Internet hour-glass model is an abstracted structure of the Internet, wherein complexity and the computational intelligence to deal with that complexity is implemented at the edges of the network, while the central connectivity is relatively lightweight. The application layer is where most functionality gets written, under the end-to-end principle. This means apps such as web apps, gaming, online shopping, and streaming must configure their software so that the only data they need to pass through the Internet core is in the form of packets. In between these end layers is the Internet's "spanning layer," which is deliberately restricted in functionality such that it only transmits packets. This design allows scalability for diverse types of applications and their implementations, provided the application creators can figure out how to reduce their network dependency on just packet transmission.



B. What does the Internet *thin waist* mean?
The "thin waist" of the Internet is similar to how functionality and intelligence is pushed to the edges. The application layer is where most of the heavy work is done for different applications, so that the Internet connection itself is only used as the transmission of packets. Computational load at the middle transmission layer is relatively light and thin, which allows for the Internet to scale and adapt easily as new applications are built.

C. [Extra:] Give two examples of protocols at the *thin waist* of the hourglass? Why are these protocols at the *thin waist*?
The only protocol in the central thin waist layer of the hourglass is IP, Internet Protocol. The transport layer is "thicker" (i.e. more complex) than the network layer, but still "thinner" than the top application layer. The transport layer encompasses TCP and UDP, although in a pure sense, IP is the only protocol that exists at the true thin waist. Although not a protocol, the network layer also involves routing algorithms and tables.

Q5.
Can the Internet provide guarantees for message delivery and bandwidth? Provide reasons for your answer [hint: you may compare/contrast your answer with the telephone network]
The Internet is designed to provide no guarantee whatsoever that any single message will be delivered or that it will receive a specific amount or limit of bandwidth. Packets are delivered best-effort. TCP may retransmit lost information, but cannot prevent overflowed buffers or downed systems. Conversely, the telephone network employs a different system called circuit switching, wherein end-to-end resources are reserved for a single call. This guarantees that one call will get through and maintain for its duration without interruption, but if the network is at capacity then other calls attempting to join the network are denied. This circuit-like resource reservation is achieved through frequency and time multiplexing. On the other hand, the Internet does not allow general reservation of resources.

Q6.
A. How were the original Internet requirements met through its design?
The first Internet design goals were scalability and economic access, robustness, reliability, and evolvability. As it was designed to have a simple message-passing core of many ISPs with end-to-end complexity pushed to the edges, this allowed for it to be scaled to many diverse future applications, reroute around failures, and achieve high utilization through resource sharing and packet switching as opposed to circuit switching. TCP fulfills reliability via ACKs and timed retransmissions.

B. What are the two main requirements that you see missing from the original design that are much needed today?
The Internet was originally designed by and for researchers, so security was not an issue—it was designed for mutually trusting users in a transparent network. If security had been built-in to the design the Internet would likely have less vulnerabilities today. The best-effort delivery design can be argued as both a positive and a negative. There are downsides to not guaranteeing packet delivery, but it also allows the Internet to not get stuck reserving resources.

Q7.

How does the Internet scale its routing tables?

DNS, or domain name system, is a core Internet function implemented as an application layer protocol, wherein distributed name servers resolve names into IP addresses. These many interconnected servers are organized hierarchically from the top root DNS server to .com / .org etc servers, down to different websites. Routes are resolved using prefixes, numbers within the IP address separated by the hierarchal regions. Each packet has its destination address in its header.

Q8.

Give an example in which wireless mobile networks may be more secure than wired networks. [hint: you can use examples given in the class/lecture].

In a wireless mobile network, users constantly move and change locations and access points. This makes it difficult to assign permanent IP addresses to each user. By not affixing permanent IP addresses, users become harder to track through mobile networks. Additionally, due to the unique conditions of a mobile network, new designs had to be invented. This led to the propagation of new technologies to connect users of a mobile network with end to end encryption.

Q9.

A. What is statistical multiplexing and what is the value it provides over TDM? Give example with numbers to support your argument(s).

TDM (time-dimensional multiplexing) is used in circuit switching networks, wherein resources are reserved and allocated for specific users for every moment in time. However, this resource reservation allows for abuse of the network, if users reserve resources but do not use the network, while other users are waiting for resources to become available. It is also overall inefficient, as shown in the in-class example: sending a 640kb file over a 1.536Mpbs link using TDM with 24 slots/sec, with 500msec to establish an end-to-end circuit, results in a 10.5 second wait. On the other hand, statistical multiplexing assumes that at any period in time, most users will not be actively using network resources, so the network can allow more users to join. In a network with 1Mbps links and each user being active 10% of the time using 100kbps when active, circuit switching only allows for 10 users while the statistical multiplexing allows for 35 users, based on the probability of 10 users all being active at the same time is less than .0004.

B. What is the main disadvantage of statistical multiplexing?

Statistical multiplexing, or packet switching in practice, is optimal for bursty data that can be split up into packets. But circuit-like behavior is still better for the single user that gets the resources, and large amounts of data with time dependency like audio visual calls may prefer circuit like behavior. Furthermore, statistical multiplexing may suffer from too many users at once if they do not behave in the predicted pattern, which leads to congestion. This lack of admission control and congestion control necessitates additional protocols.

Q10.

A. What is a DDoS attack? And why is it harder to control than a DoS attack?

DoS stands for Denial of Service. A DoS attack means that a server receives too many requests for service (e.g. hosts trying to connect to the server and receive HTML web pages), such that its buffer overflows with pending requests and the attacked server is unable to function.
A DDoS attack is a Distributed Denial of Service. Whereas a DoS may be caused by a single client trying to connect, DDoS generally implies a vast number of hosts, often spread all across the world as part of a botnet. A DoS from a single client may easily be controlled by blocking the client's IP, a DDoS comes from many discrete IP addresses, which may not be similar in ISP / region / country. There is no uniform solution to prevent all of these clients' requests, or to filter them out from legitimate traffic.
The symptoms of a DDoS may be mistaken for a "flash crowd," a similar phenomenon where a server is suddenly made popular within a matter of hours (e.g., it is linked in a viral post on social media, or gains notoriety in the news). It may suddenly be flooded with thousands or millions of requests by legitimate internet traffic. However, these sites may run on small, local servers that are ill-equipped for large influxes of traffic, and shut down similar to DDoS behavior.

Q11.

[Extra:] What is the first Internet worm, and how did it harm the Internet? [hint: Watch video link posted on canvas]

The first Internet worm was called the Morris Worm, written by Cornell grad student Robert Morris in 1988 at MIT. The worm's stated purpose was to draw attention to security flaws, exploiting vulnerabilities in UNIX systems, rsh, and weak passwords. However, the worm was able to infect a single computer target multiple times, slowing the target further and further eventually rendering the computer functionless. As it propagated, it effectively became a DoS attack, costing damage worth $100,000 - 10,000,000, and infecting two thousand computers within fifteen hours. The worm infected roughly 10% of the existing network, which forced ISPs to partition the Internet, quarantine, clean, and reconnect to prevent recontamination. Morris was convicted to 3 years probation, and DARPA established a central network emergency response team at Carnegie Mellon.

Q12.

A. Why is UDP preferred over TCP for IP-telephony/VoIP (like Skype)?

An important distinction between TCP and UDP is that TCP ensures a packet will be delivered (if the packet fails to send completely, it resends) whereas UDP is closer to "best-effort," and dropped or failed packets will not be resent to ensure complete delivery.
Although at first glance, TCP seems like the obvious, best choice for every type of Internet communication, UDP is preferable for IP-telephony/VoIP (Voice over Internet Protocol) such as Skype. This is because real-world communication has variable and unexpected delays, but human-to-human voice conversations happen in real-time. If a Skype conversation suddenly has a 5-second delay due to network lag, a pure TCP connection would send the lost sound five seconds later. But this causes the 5 second delay to propagate through the rest of the conversation, so that one speaker is only talking to what the other speaker said five seconds ago. More delays will propagate similarly, causing a confusing overlap where neither person is speaking to the other in real-time.

With UDP, that 5-second delay simply "disappears," as the lagging data is dropped and never arrives. While this may cause a confusing moment for the speakers, any notification of network lag allows them to restate what information was lost. There is no delay propagation.

B. Why would Skype sometimes use TCP? Give two reasons.
Although the main functionality of Skype is VoIP which primarily uses UDP, there are many supplementary functions within Skype that might prefer TCP. For example, when initiating a call, the process of connecting to a supernode and receiving the callee's address then directly contacting that callee is done over TCP. UDP traffic may also be blocked by some enterprise firewalls due to its ability to DDoS servers, so TCP traffic may be used to allow Skype within these networks.

Q13.
Would an application that needs congestion control ever use UDP? Give two examples to support your argument.
Yes, because although UDP does not include built-in congestion control, some software may build its own proprietary congestion control at the application layer. For some companies, this may be preferable to TCP's congestion control, if it cannot be configured to their desired specifications. Skype has congestion control, but builds that over UDP. Streaming services like YouTube also use UDP, yet build their own congestion control over it.

Q14.
What are the identifiers needed for process communication across the network? Give example of a connection between two hosts. You can use a simple drawing to aid your answer.
In order for two processes separated by the Internet to communicate, they must share two identifiers: an IP address, and a port number, for both the source and destination. This link is established with the socket programming API, where sockets are basic connections for processes to connect to and send information. By specifying these values, a process from IP A port 9157 can go to destination IP B at port 80, then receive a message back.

Q15.
Someone suggested that 'HTTP' is a stateful protocol. Argue for or against this statement.
Pure HTTP is not stateful. Creating a stateful protocol on the server-side is a complicated algorithm with need for larger resource overhead. Furthermore, a stateful HTTP protocol may be insecure, and allow for exploitation and man-in-the-middle attacks. However, there are benefits to a stateful Internet protocol, such that websites "remember" user information (login credentials, payment information, interests for recommended products or entertainment). A pseudo-stateful protocol can be implemented with the use of cookies, which are initially generated through a separate server when credentials are entered. Future usage bypasses the cookie-generation step, but from the unique identifier stored on the client-side browser, servers remember who users are. From this, they can attach the appearance of stateful protocols by tracking webpages visited.

Q16.

How do web caches/proxy servers help Internet performance? Explain its benefits from the user and network perspectives. [hint: explain using your understanding of elementary queueing theory and delays, and use graphs as needed]

In an Internet without proxy servers and web caches, users would need to request data from the main servers every time. This would lead to congestion and bandwidth problems, due to heavy amounts of traffic converging at a few locations, requesting repetitive and similar data. However, if many users within a specific region all pass through a proxy server to reach the main server, it is faster to store a copy, or cache, of that often-requested data at the proxy. This way the users can access the data from a closer server, which improves performance and reduces congestion at the main server. Traffic at both the client and server ends of the network is reduced, improving performance throughout.

Q17.

What is the effect of customized content, video streaming and encrypted traffic (e.g., HTTPS) on proxy caching?

Proxy caching is not optimal for all sorts of data. For example, many types of data are highly time-sensitive (stocks, voting, weather, sports) and storing them in local caches can lead users to out-of-date information. Video streaming is also very data-heavy, and storing copies of video at proxy caches may be inefficient. Lastly, encrypted information is less useful to store copies of, as it is encrypted for the individual user and cannot be shared to other users accessing the proxy. For these types of data, the proxy may be avoided to maintain peak information integrity, while losing out on performance.

Q18.

A. When is peer-to-peer (p2p) network architecture needed or preferred?

P2P network architecture is preferable for Internet communication wherein the users do not need or want to connect to a central server. A common example of this is illicit downloading of files (music, video games, movies) through software such as Limewire, Napster, or BitTorrent. If several other users already have the file, it can be sent in parts through the P2P network to the downloader, without ever passing through a central server. Aside from illegal uses where the users do not want to be traced, the highly scalable nature of P2P architectures makes them efficient for networks with many users connecting to one another for audio and visual data, such as Skype.

B. What is the main problem in the p2p design?

Although P2P architecture is highly scalable, it faces the problem of search. If there is a particular file one user wants to receive, they need to know which of the other users has chunks of it. Some P2P software, such as Napster, fixed this issue by using a central server that contained a list of users with associated data. However, due to the illegal downloading of music Napster was used for, this central server was not only a point of congestion and failure, but also one of legal liability. The server was shut down by law enforcement. Other designs, such as BitTorrent and Skype, use a similar "tracker" or super nodes (respectively). Another potential solution is distributed hashes, where each user can use a hash function to determine the chunks of other users.

C. Suggest a solution to the problem in B above.

As explained in B, Napster used a centralized server for metadata, BitTorrent uses a similar tracker, and Skype uses super nodes which contain information about neighboring peers. The centralized server was a single point of failure, and shut down due to law enforcement. BitTorrent and Skype are still operational. The tracker is still a legal liability but is distributed instead of being completely centralized. The Skype super nodes contain maps of users to IP addresses, and communicate with neighboring super nodes to put users in contact.

Q19.

Use '*traceroute*' and '*ping*' commands/tools to measure and analyze delays in the Internet:

A. Use *traceroute* to measure delays between your location and an overseas location (e.g., www.eurecom.fr). Show the trace and annotate it showing the transoceanic link.

```
Last login: Tue Sep 24 12:45:43 on ttys001
Leesons-MacBook-Pro:~ leesonchen$ traceroute www.eurecom.fr
traceroute to www.eurecom.fr (193.55.113.240), 64 hops max, 52 byte packets
 1  192.168.0.1 (192.168.0.1)  2.961 ms  1.392 ms  1.324 ms
 2  host-232-1.flgagro.gainesville.fl.us.clients.pavlovmedia.net (68.234.232.1)  1.942 ms  3.666 ms  1.719 ms
 3  68.234.193.21 (68.234.193.21)  2.518 ms  4.565 ms  5.145 ms
 4  68.234.128.188 (68.234.128.188)  19.392 ms  15.232 ms  15.006 ms
 5  v212.core1.atl1.he.net (216.66.70.29)  18.199 ms  20.860 ms  25.347 ms
 6  100ge8-1.core1.ash1.he.net (184.105.213.70)  48.360 ms  35.001 ms  34.519 ms
 7  100ge9-1.core1.par2.he.net (184.105.213.174)  115.828 ms  120.390 ms  103.074 ms
 8  renater.par.franceix.net (37.49.236.19)  105.931 ms  104.439 ms  108.416 ms
 9  te2-2-lyon2-rtr-021.noc.renater.fr (193.51.177.43)  120.536 ms
    193.51.180.55 (193.51.180.55)  117.657 ms
    te2-6-lyon2-rtr-021.noc.renater.fr (193.51.177.145)  120.774 ms
10  te0-0-0-5-lyon1-rtr-001.noc.renater.fr (193.51.177.216)  133.896 ms  121.153 ms  117.550 ms
11  xe1-0-1-marseille1-rtr-131.noc.renater.fr (193.51.177.222)  116.281 ms  115.888 ms  127.015 ms
12  te1-2-sophia-rtr-021.noc.renater.fr (193.51.177.21)  117.728 ms  117.833 ms  116.127 ms
13  eurocom-valbonne-gi9-7-sophia-rtr-021.noc.renater.fr (193.51.187.17)  115.884 ms  121.011 ms  115.810 ms
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * *^C
Leesons-MacBook-Pro:~ leesonchen$ traceroute https://www.asf.alaska.edu/
```

```
Last login: Tue Sep 24 22:44:25 on ttys001
Leesons-MacBook-Pro:~ leesonchen$ ping www.eurecom.fr
PING www.eurecom.fr (193.55.113.240): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
Request timeout for icmp_seq 2
Request timeout for icmp_seq 3
Request timeout for icmp_seq 4
Request timeout for icmp_seq 5
Request timeout for icmp_seq 6
Request timeout for icmp_seq 7
Request timeout for icmp_seq 8
Request timeout for icmp_seq 9
Request timeout for icmp_seq 10
Request timeout for icmp_seq 11
Request timeout for icmp_seq 12
Request timeout for icmp_seq 13
Request timeout for icmp_seq 14
Request timeout for icmp_seq 15
Request timeout for icmp_seq 16
Request timeout for icmp_seq 17
Request timeout for icmp_seq 18
Request timeout for icmp_seq 19
Request timeout for icmp_seq 20
Request timeout for icmp_seq 21
```

B. Identify machines/routers along the way with:
1. less than1ms delay,
2. 2–10ms delay,
3. 11–100ms delay, more than 100ms delay
then *ping* those machines for 15seconds each and analyze their delays
There are no machines with less than 1 ms of delay, but the first two lines are my local machine and access net in Gainesville, with a low 2 ms delay. The delay slowly gains then suddenly jumps at line 7 to the hundreds of milliseconds, indicating the transoceanic link to France.

C. Identify the locations of the machines and reason about the differences in delays
[hints: look at the traceroute example in the lecture/book and perform something similar. *traceroute* is called *tracert* on windows. On some machines you need to be super user (sudo) to run traceroute. You may run the commands from your machine or from a UF machine (e.g., storm.cise.ufl.edu), so try and see what works for you.]
Line 7 is where the delay jumps to the hundreds of milliseconds, indicating the transoceanic link. The first line is simply the closest access net, and the second line is my apartment's wifi client, Pavlov media in Gainesville. After line 7, we see the servers located in France, indicated by their names and longer delays.

Q20.
[Extra:] Visit the wireshark website at wireshark.org, read the user's manual (https://www.wireshark.org/docs/wsug_html_chunked/), then answer these questions:
A. What is wireshark?
Wireshark is a type of software that falls under the category of packet sniffers. Using Wireshark, a person can monitor packets entering and leaving a network connection. Additionally, Wireshark displays certain details about each packet, such as: the intended IP address, the intended hostname and URL, and the sender's information.

B. What are some of intended purposes? (mention four)
Wireshark allows network admins to troubleshoot network problems, network security engineers to examine security problems, QA engineers to verify network applications, and developers to debug protocol implementations.

C. What are two unintended purposes?
[hints: install wireshark and start using it to prepare for future hwks. Read intro posted on canvas.]