1 The five layers of the Internet

1. What are the five layers of the Internet stack? Application, Transport, Routing (or Network), Link (or MAC), and Physical

2. What are each of the five layers responsible for?
   Application: The application layer is where the user meets the network through an application. It encapsulates any application-specific logic for networking, such as a web browser, a video conferencing client, or a gaming client. The application layer figures out what data to send or receive to or from the network.
   Transport: The transport layer is responsible for providing end-to-end services to the application layer such as an in-order reliable byte stream abstraction (TCP) or a datagram/message-based abstraction (UDP).
   Routing: The routing layer takes care of global delivery of data from one end host to another through a sequence of routers on the Internet.
   Link layer: The link layer takes care of local delivery of data within a local network, e.g., from your Ethernet switch to your desktop or WiFi AP to your laptop.
   Physical layer: The physical layer takes care of translating bits into a physical representation, e.g., electric voltages over the wire (Ethernet) or over the air (WiFi). This physical medium could take on a variety of different forms, e.g., a sound channel as in the case of SONAR.

3. Where are each of the 5 layers implemented?
   The application and transport layers are implemented on the end hosts alone because (1) the applications that generate data for the network reside on an end host, and (2) the required semantics for transport protocols (e.g., reliable delivery) can be implemented on end hosts alone without burdening routers.
   The routing layer is implemented by both end hosts and routers because the end hosts need to generate the source and destination addresses of packets and routers need to forward packets based on the destination address. The link layer and physical layer are implemented by both end hosts and switches (Ethernet or WiFi APs) because both ends of a local network need to cooperate to send and deliver data locally.

4. Give an example of a protocol or technology at each layer.
   Application layer: HTTP, Web browsers, Skype, RTP
   Transport layer: TCP, UDP
   Routing layer: IP
   Link layer: WiFi, Ethernet, LTE, WiMAX
   Physical layer: Ethernet cable (RJ45), twisted pair, coaxial cables, WiFi antennas.

If you say routers instead of switches that’s OK. Conceptually, the distinction between the two is that a switch operates at the link and physical layer, while a router operates at the routing layer. At an implementation level, they are very similar.
2 The difference between capacity and propagation delay

Indicate propagation delay (low vs. high) and capacity (low vs. high) characteristics of each of these links.

1. Bluetooth link: low propagation delay and low capacity. Low propagation delay because they are physically so close. Low capacity because audio only needs a few 10 kbit/s.

2. Transatlantic cable: high propagation delay and high capacity. High propagation delay because it crosses a large geographic distance. High capacity because it carries traffic for an entire company site.

3. In-flight WiFi: high propagation delay, low capacity. High propagation delay because it needs to get from your laptop to the plane’s WiFi AP, and then from the AP to the cellular towers that are pointing upward. Low capacity because many people on the flight have to share the modest-capacity network.

4. Network inside a company: low propagation delay, high capacity. Low propagation delay because it’s within the same building. High capacity because you may have to shuffle large amounts of network for data processing.

Other examples: Link between an NFC card reader and NFC card is low propagation delay and low capacity. Your 2G and 3G networks on your phone are an example of high propagation delay and low capacity. Your 4G and 5G networks on your phone are an example of higher capacity and lower propagation delay (relative to 2 and 3G).

3 Finding your public IP address

My laptop’s IP address is 192.168.1.153. It’s a private IP address because it belongs to the 192.168.*** range.

My laptop’s public IP address is 98.116.140.254. My cellphone’s public IP address is also 98.116.140.254 when my cellphone connects to the Internet over WiFi.

My laptop and cellphone share a public IP address because they are both connected to the same WiFi AP and hence are behind the same Network Address Translator (NAT). The NAT translates our private IP addresses into the same public IP address.

But, if I use the cellular network to access the Internet on my phone, its public IP address is different: 107.77.225.198. That’s because the cellphone is behind a different NAT when it connects to the cellular network.

4 Measuring throughput using wget

First question in 4.3. I see a throughput of between 800 KBytes/second and 3 MBytes/second. The throughput could vary for several reasons: (1) there could be other applications that start or stop at different times and which share a link along the path between my laptop and the server, and (2) the throughput between my laptop and the WiFi AP itself could be variable.

Second and third questions in 4.3. When I run it on a machine with a wired connection to the Internet, the throughput is much higher. It’s in the range of around 30 Megabytes per second because the link between my wired machine and the Internet has higher capacity (can carry more bits per second).

After I rate limit my network to 1 Mbit/s using NLC, the throughput is about 118 KByte/s. This translates to about 944 kbit/s, which is close to the rate limit of 1 Mbit/s.

5 Protocol overheads

The size of the data packets carrying data being downloaded is 1514 bytes. The size of the packet headers on my machine are: TCP: 32 bytes, IP: 20 bytes, Ethernet: 14 bytes. The protocol overhead is \[
\frac{14 + 32 + 20}{1514}.
\]
6 Latency measurements using ping

6.1 Measuring propagation delay
1. ping localhost: 0.089 ms round trip latency
2. ping www.cs.nyu.edu: 36 ms
3. pinging www.cs.nyu.edu from a machine with a wired connection to the Internet: 1.184 ms
4. ping 139.130.4.5: 323.227 ms

There is a 3–4 orders of magnitude variation (3631.76494382x variation) in latency between the lowest and highest latencies. Local host is on the same machine so it’s the fastest. Pinging a wired host on the same network takes longer, while pinging a wireless network takes even longer. Finally pinging a server in Australia needs you to cross a larger geographic area so the latency is the highest.

6.2 Measuring queueing delay
1. The average latency to ping www.cs.nyu.edu from my laptop is about 27 ms when I rate limit to 1 Mbit/s without cross traffic.
2. Once I introduce cross traffic the ping latency goes up to several 100 ms and is quite variable: 200 to 600 ms.
3. The increased latency is the result of queueing delay because of TCP packets from the wget transfer sharing the link with ping packets.
4. If I turn off wget, the latency is back down to the original few tens of ms value.
5. On a 500 kbit/s link, without cross traffic, the ping latency is similar: a few tens of ms (40 ms or so). With cross traffic, the queueing latencies are now around 800 or 900 ms, which is more than the 200 to 600 range we saw earlier. This is because the queueing delay depends on the product of two terms: the number of packets ahead of you in the queue and the time it takes to transmit each of those packets. The time it takes to transmit those packets is higher on a 500 kbit/s link because the link capacity is smaller.

7 DNS using Wireshark
1. DNS uses UDP.
2. It runs on port 53 at the server end and an arbitrary port on the client end.
3. It contacts 4 servers on my machine.
4. Their IP addresses are: 8.8.8.8, 216.239.36.10, 192.5.5.241, 192.48.79.30
5. dig +trace takes about 116 ms.
6. dig without +trace takes about 15 ms.
7. The response comes back from 8.8.8.8, my local name server.

8 TCP echo server
See code for solution.
9  UDP RPC server

See code for solution.

10  TCP relay

See code for solution.

The odd behavior that the question asks about is how bytes “clump” up at the relay and the receiver. In other words, you might send 2 strings, 10 bytes each, at the sender, but they show up as a single 20 byte string at the receiver and the relay. See README.relay for an example of this clumping.

This is because TCP is a byte-stream abstraction that is not required to preserve the boundary between different messages sent by the sender. So there needs to be no correspondence between different send calls and different receive calls. All that is guaranteed by TCP is that the sum total of all bytes sent across all send calls (before the connection is closed), will be received by the receiver in the same order in which they were sent by the sender.

Each receive call will attempt to read up to a certain number of bytes from the receiver’s buffer, regardless of which send calls these bytes came from. This is what leads to the clumping behavior.