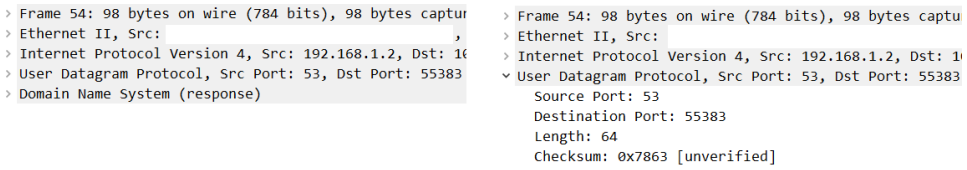
The Transport Layer and UDP

Packets and header information

Below is a screenshot from the useful tool Wireshark, which can be used to see all the network traffic happening on your computer. The screenshot shows the headers of a packet on the left, with the details of the UDP header shown on the right.



The internet is what's called a *packet-switched* network. The same lines of communication are shared by many devices. The data sent over the network is broken into units called *packets*. To make sure those packets get to their proper locations, each packet has information, called headers, added to it. Layers 2, 3, and 4 in particular, all add headers to the packet. That header information is used for sending packets to the right places and many other things that we will see.

This is sort of like how when you mail a letter, you have to put addressing information on the envelope. The difference is that instead of a single envelope, there are multiple envelopes in networking. You can think of it as the Layer 4 envelope sitting inside a Layer 3 envelope, which in turn sits inside a Layer 2 envelope.

This section will focus on parts of the Layer 4 header. The two main protocols at this layer are UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). The two protocols both provide the following four pieces of information: length, source port, destination port, and a checksum. TCP provides a lot more, which we'll talk about later. The length is the number of bytes of the header and packet data. The other fields will be covered below.

Ports

At the Transport Layer, a port is a number from 0 to 65535 that indicates which application a packet is associated with. For instance, port 80 is a standard port for HTTP web servers. If a packet arrives with port number 80 set, the networking software uses that to know to deliver the packet to the web server application running on the machine.

To stick with the postal analogy from above, IP addresses are sort of like street addresses, while ports are like names or apartment numbers. Once the mail gets to the right building, the name or apartment number is used to deliver it to the right person. Similarly, once a packet arrives at a computer, the port number is used to deliver it to the right application running on that computer.

In a networked communication there are two ports: a source port and a destination port. Suppose A and B are communicating, with A using port 54321 and B using port 80. When A is sending things to B, the source port is 54321 and the destination port is 80. When B is sending things to A, things flip, with the source port being 80 and the destination port being 54321.

Here a few common port numbers that are good to know.

20, 21	FTP	80	HTTP
22	SSH	110	POP3
23	Telnet	143	IMAP
25	SMTP	443	HTTPs
53	DNS		

These numbers are only suggestions. There's nothing stopping someone from running HTTP at a port other than 80 or even from running some other service at port 80. Quite often people run HTTP at unusual port numbers,

especially if port 80 is in use by a different service on the same machine.

Someone I know was having issues with unwanted people trying to SSH into a machine at the standard port of 22, so he moved it to some obscure port number. That stopped the majority of the unwanted traffic. But it won't stop all of it. A very common tool is a *port scanner*. This is a program that systematically tries to make network connections at a variety of port numbers and reports which ones replied. They are commonly available and easy to use. People use them all the time to find vulnerable services running on servers.

Moving the SSH server to an obscure port number is a little like getting rid of your front door, and installing a door around the back of your house. It will keep away most ordinary people, but not someone who goes looking around your house.

Wikipedia has a nice list of which port numbers are assigned to which services at https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers. Port numbers are usually distinguished in the following ranges:

- 0–1023: These are called *well-known port numbers*. Many of them, like those listed above, are assigned to established services. For instance, web servers typically run on ports 80 and 443. Most sites agree to run their web servers at these ports, which makes it easier for everyone than if each site ran them at whatever port it wanted to. You can run things on your system at port numbers in this range if you want to, but if you pick something assigned to a standard service, it might lead to confusion.
- 1024–49151: These are called *registered port numbers*. Some of them are assigned to specific services, and many are open.
- 49152–65535: These are called *dynamic* or *private* port numbers. None of them will officially be assigned to some specific service. They are available for the operating system to use when it needs a port for something. Often they are chosen randomly by the OS. For example, if you visit a web page, your computer will make a request to port 80 or 443 on the remote web server. That's the destination port. The source port on your computer will be a randomly chosen port in this range. Different ports are used to distinguish the different connections your computer makes, so that for each page you download something from, your browser will use a different random port number. Future downloads from the same page will use different port numbers as well, once the original connection has ended.

Checksums

The purpose of a checksum is to try to catch if a packet has been corrupted in between the sender and receiver. Network communication is subject to physical problems or software glitches that can cause a bit or two (or more) to get flipped from the correct value. A checksum is a quick calculation done on the bits of a packet. The sender does that calculation and sends the value of the calculation (the checksum) along with the rest of the packet. When the receiver gets it, they recompute the checksum and see if it agrees with what the sender got. If it does, then we assume the packet is okay, and if it does not agree, then we know the packet was corrupted in transmission.

The UDP checksum is computed as follows: the packet data and parts of the header are collected and broken into blocks 16 bits in size. Then those blocks are all added together, using ordinary binary addition, with the exception that if there is a carry in the last (leftmost) digit, then it carries around to the front. Once everything is added up, flip all the bits (called the one's complement). That 16-bit result is the checksum. Wireshark and other tools will often display it in hexadecimal.

Here is an example. Suppose the data and headers come out to 10100010100111110101001110101000. We break it into two blocks of 16 bits, as below, and add those blocks. The rules are that $0+0=0$, $0+1=1+0=1$ and $1+1=10$. In the last case the 1 is treated like a carry, just like in the grade school addition algorithm. See below (carries are shown in the first line).

```
      111   111
1010 0010 1001 1111
0101 0011 1010 1000
-------------------
1111 0110 0100 0111
```

Then we flip all the bits of the sum to get 0000 1001 1011 1000, which is 0x09B8 in hex. Binary addition is painful for humans, but very fast for computers. The checksum needs to be fast since network connections can transfer millions or billions of bytes a second, all of which may need to be checksummed.

This checksum will always catch if a single bit gets flipped. However, if more than one bit is flipped, it might happen that the two flips end up canceling each other out in the checksum. In the figure above, if the two bits in the leftmost column got flipped from 1 and 0 into 0 and 1, the checksum would still come out the same. So checksums are not perfect.

Checksums are used all over in networking and data storage. They are also used in things like UPC codes, ISBN numbers, and credit card numbers to catch mistakes in entry and scanning.

UDP vs. TCP

The UDP header contains only the port information, the length, and a checksum. The TCP header has that and a lot more. TCP's job is to recognize when data has been lost and arrange for it to be resent. It also performs congestion control to keep the internet from being overwhelmed with traffic. UDP does none of this. It's job is to be as simple as possible. It's basically almost a direct connection from the higher OSI layers to the lower layers, without adding any additional overhead.

People often treat the U in UDP as meaning "unreliable". This is a helpful way to remember that UDP does nothing to deal with packet loss. If packets are lost because someone stepped on a network cable or because an upstream router got overwhelmed with traffic, TCP will work to make sure that traffic is resent, while UDP will not. Because of the work it does to ensure reliability, TCP is slower than UDP. It will also more or less pause information transfer until missing data arrives. UDP, on the other hand, keeps going regardless.

For this reason, TCP is used in situations such as file transfers, where you would not want to lose any part of the file. UDP is used for things like streaming video, where you could tolerate losing a few frames here and there.

Consider, for instance, a video conference call on a bad connection. If it is done with TCP, there will be long pauses while TCP waits for data to be resent. You won't miss any part of the call, but you will end up out of sync, possibly several minutes behind. With UDP, you will always be in sync, but you may miss some parts of the call.