The Internet Protocol

The Internet Protocol (IP) is the most important protocol at the network layer. Below we will look at the different parts of the IPv4 header and how they work.

Ver.	IHL	TOS/DS	Total Length	
Identification			Flags	Fragment Offset
TTL		Protocol	Checksum	
Source Address				
Destination Address				
Options				

The header consists of 5 rows of 32 bits, followed by some options that are rarely included. So the IP header is typically 20 bytes. Let's look at each part of the header.

Version The first four bits are for the version number. This is always set to 4 since this is IPv4.

IHL IHL is short for internet header length. It takes up four bits of the header. It indicates how many rows long the header is. The purpose of this is to help networking software know where the header ends and the data begins.

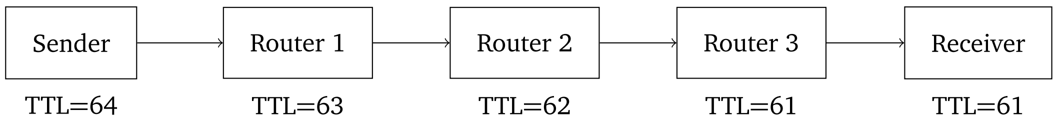
TOS/DS This 8-bit field has changed meaning over the years. Originally, TOS stood for Type of Service, and it has now morphed into Differentiated Services and Explicit Congestion Notification (ECN). The basic idea of Differentiated Services is that it gives a way to prioritize certain types of traffic over others. For instance, streaming video should get higher priority over some OS updates that are being downloaded in the background.

The last two bits of this field has been redefined for ECN, which is used together with an ECN flag in the TCP header. If a router is experiencing congestion, it can set the ECN bits in the IP header to let the let the receiver know about the congestion. The receiver then sets the ECN flag in the TCP header in its ACK back to the sender so that the sender knows to slow down. This is a means of congestion control that avoids packet loss.

Total length The last part of the first row is the total length of the IP header and data combined. It is a 16-bit field, which means the maximum length of an IP packet is $2^{16} - 1$ =65,535 bytes. However, most internet traffic runs at some point over ethernet, which has a 1500-byte limit, so most IP packets will be no larger than 1500 bytes.

Row 2 of the header We will cover this a little later when we talk about fragmentation.

TTL This is short for time to live. Each router decreases the value of the TTL by 1 before forwarding the packet to the destination. This is shown in the example below.



If the TTL ever reaches 0, the router that the packet is at will drop the packet, and it may send back an error message to the sender indicating that the packet's TTL expired. This message is an ICMP "Time Exceeded" message. We will cover ICMP at a later time.

Why do this? Sometimes packets get caught in routing loops, where they could bounce around endlessly between the same routers. TTL is a way to kill packets that are on the network for too long. Without it, the network would eventually fill up with these packets.

On the internet, most packets take no more than around 30 to 40 hops to get to their destination, so to be safe, the sender should make sure the TTL is larger than this. The TTL field is 8 bits, which means the maximum TTL is $2^8 - 1 = 255$. Most Windows systems set the TTL to 128 and most Mac and Linux systems set it to 64.

Protocol This is an 8-bit value that indicates what type of traffic is being carried by this packet. Each type has a corresponding numerical code that goes in this field. The most common types are TCP, UDP, and ICMP.

Checksum This is a 16-bit checksum, taken over only the content of the IP header. It is computed in the same way as the TCP and UDP checksums.

Addresses The source and destination addresses are each 32 bits in the header, meaning there are $2^{32} \approx 4$ billion addresses, theoretically.

Options There are a variety of options, but most of them are not used, often for security reasons. One interesting one is source routing, which allows a sender to specify the sequence of routers they want the packet to go through.

Just for fun, here is a capture from Wireshark showing the entire contents of the IP header for a particular packet.

```
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.1.116
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 377
    Identification: 0x0000 (0)
    Flags: 0x4000, Don't fragment
        0... .... .... .... = Reserved bit: Not set
        .1.. .... .... .... = Don't fragment: Set
        ..0. .... .... .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0xb5ae [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.1.1
    Destination: 192.168.1.116
```

Fragmentation

Row 2 of the IP header is dedicated to fragmentation. The purpose of fragmentation is if a packet needs to be forwarded on a line which can't handle a packet of that size, then the packet is broken into pieces small enough to fit, called *fragments*. The stuff in row 2 of the header is mostly there to help with reassembling those fragments.

The three key parts of the header for this purpose are Identification, Fragment Offset, and the MF (more fragments) flag. The Identification field is used to indicate what packet the current fragment belongs to. It's possible that fragments from multiple different fragments could arrive, and this field is used to help know which fragments go with which packets.

The Fragment Offset field is used to determine where the current fragment fits in the overall set of fragments for a packet. It is a byte number, sort of like the TCP sequence number.

The MF flag is used to indicate if there are more fragments coming or not. If there are, it's set to 1, and if there aren't then it's set to 0. This is how the receiver will know what the last fragment is; the MF flag will be 0 for it.

There are two other bits in row 2 of the IP header. One of those bits is a reserved bit that senders must set to 0. The other bit is the DF (don't fragment) flag. It tells a router not to fragment a packet. If the packet is too large, and the DF flag is set, then it is dropped and an ICMP error message may be sent back to the sender indicating this. It is used for something called *path MTU discovery*, covered a little later, which is a way for a sender to figure out the largest packet they can send without it having to be fragmented.

Fragmentation was an important part of IP when it was first created, but it's not much used in the modern internet. Fragmentation is also used by some people to evade firewalls. Some firewalls determine whether or not to accept a packet by looking at things like port numbers and TCP flags. A packet can be cleverly fragmented so that the header info is broken across fragments, making the fragments able to sneak by the firewall.

Ping

Ping is a simple and extremely useful networking tool. Its most common use is to see if a host is reachable. Here is the output of running a simple ping in Windows PowerShell.

```
Pinging google.com [172.217.13.78] with 32 bytes of data:
Reply from 172.217.13.78: bytes=32 time=26ms TTL=112
Reply from 172.217.13.78: bytes=32 time=26ms TTL=112
Reply from 172.217.13.78: bytes=32 time=26ms TTL=112
Reply from 172.217.13.78: bytes=32 time=28ms TTL=112

Ping statistics for 172.217.13.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 26ms, Maximum = 28ms, Average = 26ms
```

Ping sends a small amount of data in an ICMP message to the destination. If the destination is reachable and is configured to respond to pings, then it sends back a reply. On Windows, ping automatically sends four pings. On Mac and Linux, the pings are generated continuously until you tell it to stop (via Ctrl+C). Ping also provides timing information, which is useful for diagnosing networking problems. Here are a few of the most useful options.

Description	Windows	Mac	Linux
number of pings	-n	-c	-c
size of ping	-l	-s	-s
TTL	-i	-T	-t
do not fragment	-f	-D	-m do

For instance, on Windows, to send two 500-byte pings with TTL 255, we could use the following:

```
ping google.com -n 2 -l 500 -i 255~.
```

Ping is one of the first things I try if I'm having network connectivity problems. I'll try to ping Google and my router to see which ones, if any, I'm able to reach. Note that some network administrators disable their machines from responding to pings. This is to prevent people from discovering those machines.

Traceroute

Traceroute is a useful networking tool for seeing the path a packet takes from source to destination. Here is an example ping I ran from my house to Google.

```
  1     1 ms     2 ms    <1 ms  192.168.1.1
  2    15 ms    13 ms    12 ms  96.120.9.9
  3    38 ms    16 ms    27 ms  162.151.69.213
  4    24 ms    24 ms    14 ms  96.108.5.201
  5    21 ms    22 ms    23 ms  be-34-ar01.mckeesport.pa.pitt.comcast.net [69.139.168.141]
  6    26 ms    23 ms    25 ms  be-31641-cs04.pittsburgh.pa.ibone.comcast.net [96.110.42.173]
  7    24 ms    22 ms    24 ms  be-1411-cr11.pittsburgh.pa.ibone.comcast.net [96.110.38.142]
  8    42 ms    28 ms    29 ms  be-302-cr12.ashburn.va.ibone.comcast.net [96.110.32.101]
  9    29 ms    28 ms    40 ms  be-1112-cs01.ashburn.va.ibone.comcast.net [96.110.32.201]
 10    28 ms    36 ms    28 ms  be-2111-pe11.ashburn.va.ibone.comcast.net [96.110.32.122]
 11    34 ms    30 ms    27 ms  50.248.119.106
 12    27 ms    27 ms    27 ms  108.170.229.246
 13    30 ms    31 ms    32 ms  209.85.251.83
 14    26 ms    26 ms    27 ms  iad23s63-in-f14.1e100.net [172.217.15.78]
```

We see that the first hop is my home router at 192.168.1.1. Then second hop is 96.120.9.9. A little while later we see some addresses that are part of the internet backbone, which is sort of like the internet equivalent of a major interstate highway. After 14 hops, the packet reaches its destination. The names you see for some of the hops are gotten by a reverse DNS lookup (using the PTR resource record). Not all the routers on the network have names. This name lookup is the slowest part of tracert. See a little later for how to disable it and just see the IP addresses.

Sometimes, when you do a traceroute, you may see a hop that gives no information. It will look like `* * * Request timed out`. That is often because the router at which the packet's TTL hit 0 is configured for security reasons to not send back ICMP Time Exceeded messages.

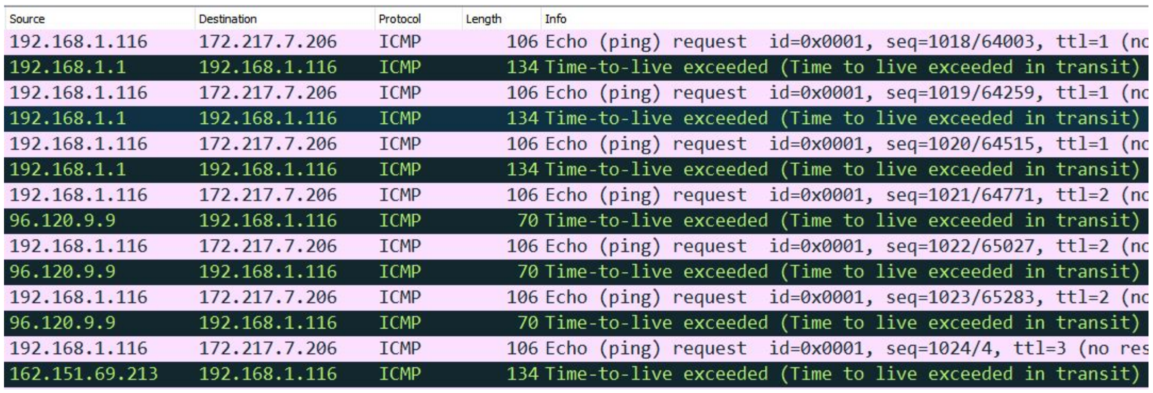
How Traceroute works Traceroute works by making clever use of the TTL field in the IP header. The first three steps are shown below.

1. Send a packet with TTL=1. This packet will get to the router at the first hop, and the packet's TTL will drop to 0. The packet is dropped by the router, and the router sends an ICMP Time Exceeded message back to us. The IP address of the router will be visible in that message.

2. Send a packet with TTL=2. This packet will get to the router at the second hop before its TTL reaches 0. It's dropped there, and we'll get a Time Exceeded message from that router.

3. Send a packet with TTL=3. This packet gets to the router at the third hop before its TTL reaches 0. It's dropped, and the router at that third hop sends us a Time Exceeded message.

Traceroute continues this by sending packets with TTL 4, 5, 6, etc. Here is a Wireshark screenshot showing the first few steps of the process. Notice near the right the TTL increasing from 1 to 2 to 3. Traceroute by default repeats each step 3 times.



Source	Destination	Protocol	Length	Info
192.168.1.116	172.217.7.206	ICMP	106	Echo (ping) request id=0x0001, seq=1018/64003, ttl=1 (nc
192.168.1.1	192.168.1.116	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.116	172.217.7.206	ICMP	106	Echo (ping) request id=0x0001, seq=1019/64259, ttl=1 (nc
192.168.1.1	192.168.1.116	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.116	172.217.7.206	ICMP	106	Echo (ping) request id=0x0001, seq=1020/64515, ttl=1 (nc
192.168.1.1	192.168.1.116	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.116	172.217.7.206	ICMP	106	Echo (ping) request id=0x0001, seq=1021/64771, ttl=2 (nc
96.120.9.9	192.168.1.116	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.116	172.217.7.206	ICMP	106	Echo (ping) request id=0x0001, seq=1022/65027, ttl=2 (nc
96.120.9.9	192.168.1.116	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.116	172.217.7.206	ICMP	106	Echo (ping) request id=0x0001, seq=1023/65283, ttl=2 (nc
96.120.9.9	192.168.1.116	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
192.168.1.116	172.217.7.206	ICMP	106	Echo (ping) request id=0x0001, seq=1024/4, ttl=3 (no res
162.151.69.213	192.168.1.116	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)

We can do a manual traceroute using pings. Here is a short example using the Windows version of ping, with a few lines removed from the output for clarity. The “Reply from. . . ” lines tell us the IP addresses of the various routers each packet reaches.

```
> ping google.com -n 1 -i 1
Pinging google.com [172.217.15.78] with 32 bytes of data:
Reply from 192.168.1.1: TTL expired in transit.

> ping google.com -n 1 -i 2
Pinging google.com [172.217.9.206] with 32 bytes of data:
Reply from 96.120.9.9: TTL expired in transit.

> ping google.com -n 1 -i 3
Pinging google.com [172.217.9.206] with 32 bytes of data:
Reply from 162.151.69.213: TTL expired in transit.
```

Using Traceroute On Mac and Linux, the tool is called `traceroute`, but on Windows it’s called `tracert`. The Windows version sends out ICMP pings by default, while the Mac/Linux version uses UDP and has options to use ICMP or TCP. A useful option is `-d` on windows and `-n` on Mac/Linux, which tells traceroute to not try to find the domain names associated with the IP addresses. This considerably speeds up the result. There are also options that allow you to set a limit so that it never searches past a certain number of hops. On Mac and Linux (but not Windows), you can set the TTL value to skip some hops, to start right at hop 5, for instance. The Mac/Linux version has some other interesting options that aren’t available on the Windows version.

Path MTU discovery

The *maximum transmission unit* (MTU) of a network is the largest packet size that can get through from a sender to a receiver. Often there are multiple types of network media that a packet will pass through on its way to its destination. It might first travel over Wifi to a router, then over ethernet to another router, then maybe over a fiber optic line to another router, etc. Some media can handle larger packet sizes than others. The MTU is limited by whatever the smallest of these sizes is.

A sender can find the MTU of a path to a receiver by sending packets (usually pings) of varying sizes and seeing what gets through. The trick is to set the DF flag to true so that if a packet is too large, it will be dropped and the router dropping it will send back an ICMP error message. Note that for security reasons, some people set their network devices to not respond to pings or send ICMP error messages, which makes path MTU discovery more difficult.

For example, maybe the sender tries to send a 600-byte packet and it gets through. Then they try a 2000-byte packet and maybe it doesn’t get through. So they know the path MTU must be somewhere between 600 and 2000 bytes. A reasonable next try would be the middle value of these, which is 1300 bytes. They continue this process to determine the exact value. A very common value is 1472 bytes. This is due to the 1500-byte ethernet MTU. Specifically, 1472 bytes of data plus 20 bytes of the IP header, plus 8 bytes of the ICMP header. Here is an example of the first two steps of the process using Windows, with a few lines removed from the output for clarity.

```
> ping google.com -n 1 -l 600 -f
Pinging google.com [172.217.9.206] with 600 bytes of data:
Reply from 172.217.9.206: bytes=68 (sent 600) time=6ms TTL=114

> ping google.com -n 1 -l 2000 -f
Pinging google.com [172.217.9.206] with 2000 bytes of data:
Packet needs to be fragmented but DF set.
```

A final note, there you can use `netsh interface ipv4 show subinterfaces` on Windows and `sudo ifconfig | grep MTU` on Mac/Linux to see the MTU of the various interfaces on your machine.