49202 Communication Protocols

The Network Layer

Daniel R. Franklin

Faculty of Engineering & IT

April 11, 2023

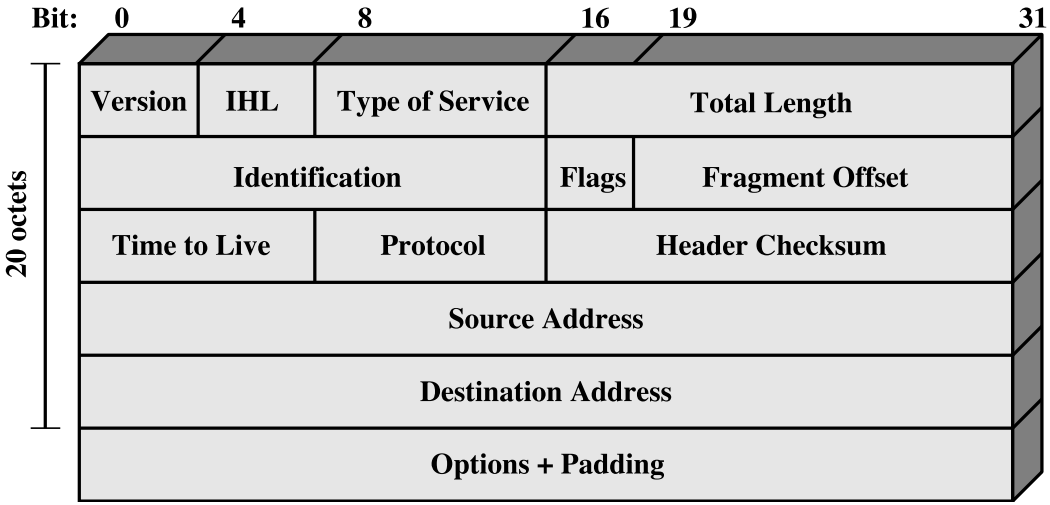
Role of the network layer

- The network layer has a seemingly simple task: **get datagrams from a host in one network to a host in another network**
- The network layer in TCP/IP (IPv4 or IPv6) is datagram-based - each datagram is handled independently (like sending letters through the post)
- We have already seen similar transport-layer behaviour with UDP - which is not surprising since UDP adds very little on top of IP
- IP is not responsible for, nor does it care about, reliability of delivery, order of delivery, or the integrity of the packets - it does its best but offers no guarantee
- This is known as a *best-effort* service
- However, we *do* have a limited ability to tell routers how we would *like* them to prioritise our datagrams

Routing

- **Routing** - finding a path through the network from source to destination - is the main responsibility of the network layer
- Each packet is routed **independently** between different layer 2 networks via a series of **routers**
- The decision about what to do with a given IP datagram is made by each router based on
  - The **destination IP address** in the datagram
  - The router's **routing table**
- In general, *nothing else matters* in making IP routing decisions.

IPv4 datagram header



IPv4 header structure

- 4-bit Version field - for IPv4 this will be 4 (0100_2)
- 4-bit Internet Header Length (IHL) field - length of the IPv4 header in blocks of 32-bit words.
- The minimum (and default) IP header length is 20 bytes; this would be equal to 5 (0101_2). Maximum IP header length is $1111_2 = 15_{10}$ 32-bit words or 60 bytes.

IPv4 header structure

- 8-bit Type of Service (ToS) field - used to inform routers how we would like this datagram to be prioritised for queueing. Its precise use has changed over the years - now it is interpreted as follows:
  - A 6-bit Differentiated services code point (DSCP) - various values are currently defined ranging from 0 to 56 to indicate the traffic class. 0 = best-effort, 48 or 56 mean routing control packets
  - A 2-bit Explicit Congestion Notification (ECN) field - can be set by router a router to inform the destination that congestion is present. 00_2 = endpoint doesn't support ECN; 01_2 or 10_2 = sender or receiver support ECN (treated the same by routers) and 11_2 = congestion is being flagged. ECN can be used to tell TCP (or other congestion-aware protocols) to slow down.
  - If a TCP receiver sees ECN = 11_2, a Congestion Experienced flag is set in the TCP header.
- **Usually** all these bits are set to zero.

IPv4 header structure

- 16-bit Total length field - entire length of the datagram (header + payload) in bytes; minimum = 20 bytes, maximum = 65535 bytes.
- This is not necessarily the length of the layer 2 payload - that is limited by the MTU. In this case, the datagram will be fragmented (more on this later).
- 16-bit Identification field - a number used to identify different fragments which are part of the same IP datagram. It isn't **strictly** necessary for this to be different for every successive datagram sent by a host, but usually it will be.

IPv4 header structure

- 3-bit Flags field - this is used to control fragmentation; only the upper two bits are used (the least-significant bit is zero);
  - Bit 1 - DF - don't fragment this datagram (drop it if the MTU of the next network is too small for this datagram)
  - Bit 2 - MF - this datagram is a fragment and there are more fragments coming. Set to zero if it is the last fragment of a multi-fragment sequence.
- 13-bit Fragment Offset - used to reassemble the fragments at the destination; indicates where this fragment should go in the overall sequence, in units of 8-byte octets (64 bit blocks).

IPv4 header structure

- 8-bit Time To Live (TTL) field - set to an initial value by the sender, and decremented when the datagram arrives at a router. If it reaches zero it will be dropped by the router and an ICMP TIME EXCEEDED error message sent back to the sender.
- 8-bit Protocol field - used to identify the payload. The Internet Assigned Numbers Authority (IANA) has a standard list, e.g. 1 = ICMP, 6 = TCP, 17 = UDP etc.
- `https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers`
- Header checksum - 16-bit checksum which is calculated ONLY on the header. If we care about the payload, the transport layer can provide additional checksumming capabilities (e.g. TCP or UDP).

IPv4 header structure

- Source and destimation IPv4 addresses - 32-bit address fields, these are probably the most important parts of the header - much more to come about these!
- Options field - rarely used; if it is, it is structured with the following sub-fields:
  - Copied (1 bit) - Set to 1 if the options need to be copied into all fragments of a fragmented packet.
  - Option Class (2 bits) - A general options category. 0 is for control options, and 2 is for debugging and measurement. 1 and 3 are reserved.
  - Option Number (5 bits) - Specifies an option identifier
  - Option Length (8 bits) - Indicates the size of the entire option (including this field). This field may not exist for simple options.
  - Option Data (variable length) - Option-specific data. This field may not exist for simple options.
- Depending on the options, padding (zeros) maybe used to make the total header length a multiple of 4 bytes (see the discussion on the IHL field)

IPv4 Address Basics

- Complete IPv4 *host* addresses are 32 bits (4 8-bit bytes) long (1 byte = 8 bits)
- The total address space size is a bit less than $2^{32} = 4,294,967,296$
- However, some 32-bit numbers are *not valid* IP addresses (more on this shortly)
- IPv4 addresses are normally written in “dotted decimal” form, for example:

$$123.10.200.5$$

- Each decimal number represents 8 bits and can be in the range $0 - 255$.

IPv4 Address Classes

- IP addresses belong to one of a number of *classes*:

Class	Leading Bits	From	To	Network bits	Networks	Host bits	Hosts
A	0	0.0.0.0	127.255.255.255	8(7)*	$2^7 = 128$	24	$2^{24} = 16,777,216$
B	10	128.0.0.0	191.255.255.255	16(14)*	$2^{14} = 16,384$	16	$2^{16} = 65,536$
C	110	192.0.0.0	223.255.255.255	24(21)*	$2^{21} = 2,097,152$	8	$2^8 = 256$
D	1110	224.0.0.0	239.255.255.255		N/A (Multicast groups)		
E	1111	240.0.0.0	255.255.255.255		N/A (Reserved for future use)		

*The leading 1, 2 or 3 bits are already 'used up', leaving the remaining bits for defining the network ID number. More on this point later!

Subnetting and Variable Length Subnet Masks (VLSM)

- When there were only a few networks and hosts on the Internet, classes were an adequate solution to dividing up the address space
- However, you might only need a small fraction of the addresses available within a network belonging to a particular class
- What do you do (for example) if you needed to support up to 1000 hosts?
  - A Class C (256-address block) is not enough
  - You have to allocate a whole Class B network and use up 65,536 addresses!
- The solution to this problem is to use a 32-bit *network mask*

The Network Mask (netmask)

- The network mask (or netmask, or generality mask or genmask) is a 32-bit binary number comprising a block of **contiguous binary ones** followed by a block of **contiguous zeros**
- It is written together with the IP address, either in dotted decimal notation or just as a slash followed by the number of ones - for example, the following IP addresses/netmasks are all equivalent:

$$01111011.00001010.11001000.00000101/11111111.11111111.11110000.00000000$$
$$123.10.200.5/255.255.240.0$$
$$123.10.200.5/20$$

The Network Mask (netmask)

- The **network ID** number (network address) is calculated by performing a bitwise AND between the IP address and the netmask
- The **host ID** number is calculated by performing a bitwise AND between the IP address and the complement (bitwise inversion) of the netmask
- This is known as **Variable Length Subnet Masking (VLSM)**.
- In this example, the network part of the address is:

$$\begin{aligned} &01111011.00001010.11001000.00000101 (= 123.10.200.5) \ \& \\ &\underline{11111111.11111111.11110000.00000000} (= 255.255.240.0) \\ &01111011.00001010.11000000.00000000 (= 123.10.192.0) \end{aligned}$$

- The host part is the remaining bits:

$$00000000.00000000.00001000.00000101 = 2053$$

Subnetting

- The basic idea is that we can split a given network into smaller sub-networks and allocate them in a way which better aligns with the needs of each network
- An organisation can be allocated a large block (e.g. one or two Class B networks) and, using VLSM, divide these up internally into smaller networks
- There will still be some waste (very rarely would you need exactly 2^N for an integer value of N in a single network), but it is much less than with classful IP addressing.
- The price paid for this convenience is that we now need to distribute the network masks along with IP addresses when we disseminate routing information or allocate IP addresses via DHCP.

Example

Checking the IP address and netmask of my laptop in a public WiFi network (on Linux, use the `ip addr` or `ifconfig` command; in Windows and Mac OSX use `ipconfig`), I see the following:

$$172.20.129.101/19$$

- Write the netmask in dotted decimal notation
- To what class does this IP address belong?
- What are the network and host ID numbers?
- What is the first and last VALID IP address in this network?
- How many valid addresses do we have in this network?

Example - Solution

$$172.20.129.101/19$$

- Write the netmask in dotted decimal notation:
  /19 means 19 ones followed by (32-19) zeros. In binary, we have:

  $$11111111.11111111.11100000.00000000$$

  Converting each group of 8 bits to decimal, we have **255.255.224.0**.
- To what class does this IP address belong?
  Writing the IP address 172.20.129.101 in binary, we have:

  $$10101100.00010100.10000001.01100101$$

  The first (left-most) bits of this address are **10**. Consulting the table previously shown, we can see that this must be a **class B** IP address.

Example - Solution

- What are the network and host ID numbers?
  Perform the bitwise AND operation:

$$\begin{array}{ll} 10101100.00010100.10000001.01100101 & \& \\ \underline{11111111.11111111.11100000.00000000} & = \\ 10101100.00010100.10000000.00000000 & \end{array}$$

  The result is **172.20.128.0** in dotted decmal format. The host ID part is the remaining 13 bits: 0000101100101, which is **357** as a decimal number.

Example - Solution

- What is the first and last VALID IP address in this network?
  If we set all the host field bits to zero, then add one, we have the first valid address; if we set them all to one, then subtract one, we have the last valid address. Doing this with our IP address will give **172.20.128.1** for the first, and **172.20.159.254** for the last.
- How many valid addresses do we have in this network?
  The host field is $32 - 19 = 13$ bits long. The total number of valid addresses is $2^{13} - 2 = \mathbf{8,190}$ (we don't use all-zeros or all-ones host IDs).

Example - Dimensioning Subnets

- UTS has been allocated the Class B network 138.25.0.0/16. Suppose that we wish to divide this address space between 8 faculties, each of which will receive an equal IP address allocation. What will be the network IDs, and how many hosts can be accommodated in each?
- We have 16 bits for the subnet + host ID field. We need $8 = 2^3$ subnets; therefore, the subnet field is 3 bits, leaving 13 bits for the host field in each subnet. This means that 8,192 - 2 = **8,190** hosts can be accommodated in each subnet (since all-zeros and all-ones addresses are not used).
- The network IDs will be **138.25.0.0/19**, **138.25.32.0/19**, **138.25.64.0/19**, **138.25.96.0/19**, **138.25.128.0/19**, **138.25.160.0/19**, **138.25.192.0/19** and **138.25.224.0/19**
- We could also perform further subnetting (e.g. one faculty's allocation could be split between schools). Subnetting can be performed hierarchically within an organisation.

Observations & Comments

- IPv4 is *not* classless, despite the term CIDR!
- Class still matters - but CIDR provides the ability to divide up the address space with finer granularity than via class alone.
- It is important to remember that **the first one, two, three or four bits** of the IP address determine its class.
- The network portion of the IP address *includes* these bits. This means that even though a Class X IP address is split into a network and host ID number, part of the network ID number is *fixed* for each class.
  - For Class A networks, the network part of the address is 8 bits, but the first bit is 0, leaving 7 bits for the network ID number: 128 Class A networks
  - For Class B, the network part is 16 bits and the first *two* bits are 10, leaving 14 bits for the network ID: 16,384 Class B networks
  - For Class C, the network part is 24 bits and the first *three* bits are 110, leaving 21 bits for the network ID: 2,097,152 Class C networks

Classless inter-domain routing (CIDR)

CIDR is a mechanism that allows routing tables to be simplified

- For example, suppose we have the following set of four stub networks connected to a router **R**

  192.168.104.0/24
  192.168.105.0/24
  192.168.106.0/24
  192.168.107.0/24

- Examining the third byte of each address (104, 105, 106, 107), we see that in binary, they look like this:

  01101000
  01101001
  01101010
  01101011

- All but the last two bits are the same! So we can describe the entire set of 4 networks as a single subnet **192.168.104.0/22**.

Address Aggregation / Route Summarisation

- Route summarisation reduces the number of network prefixes that the router needs to advertise - it can distribute a single prefix which will tell any peer that it can reach any of the four individual subnets by forwarding the packet to **R**.
- More generally, we can summarise a group of subnets provided that the following conditions are **ALL** satisfied:
  - We have exactly 2^N networks for integer values of N
  - The networks address blocks are **contiguous** - no gaps!
  - The lower and upper boundaries of the lowest and highest subnet address block are aligned to the lower and upper boundaries of the supernet
- The aggregate network prefix is called the *supernet* and this practice is referred to as **supernetting** (also route aggregation or route summarisation).

Route Summarisation Example

- Is it possible to summarise the following subnets, and if so, what is the summary route?
  1. 192.168.4.0/24
  2. 192.168.5.0/24
  3. 192.168.6.0/24
  4. 192.168.7.0/24
- We have $2^2 = 4$ subnets, and there are no gaps between the subnets (they are consecutive).
- Looking at the 3rd byte, we have 00000100, 00000101, 00000110 and 00000111. The first 6 bits are identical (000001), and the remaining 2 bits cover the 4 possible values (00, 01, 10, 11). So the subnets are aligned on a /22 supernet boundary!
- So the answer is yes; the summary route is **192.168.4.0/22**
- What about 192.168.2.0/24, 192.168.3.0/24, 192.168.4.0/24, 192.168.5.0/24?

Route Summarisation Example

- Is it possible to summarise the following subnets, and if so, what is the summary route?
  1. 125.16.64.0/25
  2. 125.16.64.128/25
  3. 125.16.65.0/25
  4. 125.16.65.128/25
  5. 125.16.66.0/25
  6. 125.16.66.128/25
  7. 125.16.67.0/25
  8. 125.16.67.128/25
- We have $2^3 = 8$ subnets, and there are no gaps. Eight /25 subnets (if summarisable) would be a single /22 subnet, which would be 125.16.64.0/22. The subnets are aligned to this boundary, so we can summarise these 8 networks as 125.16.64.0/22.

Route Summarisation Example

- Is it possible to summarise the following subnets, and if so, what is the summary route?
  1. 138.25.229.16/28
  2. 138.25.229.32/28
  3. 138.25.229.48/28
  4. 138.25.229.64/28
- Looking at the last byte (which is the location of the network/host ID boundary), we have 00010000, 00100000, 00110000, 01000000
- Unfortunately, while we DO have $2^2 = 4$ subnets, and there are no gaps, these four subnets do NOT align on a /26 supernet boundary (the nearest boundaries are at 138.25.229.0/26 or 138.25.229.64/26 - convert the last byte to binary to see this). Therefore, these subnets cannot be summarised by a single summary prefix. The middle two addresses could be summarised as a single /27 supernet, reducing the number of prefixes being advertised from 4 to 3 in total (the summary route would be 138.24.229.32/27).

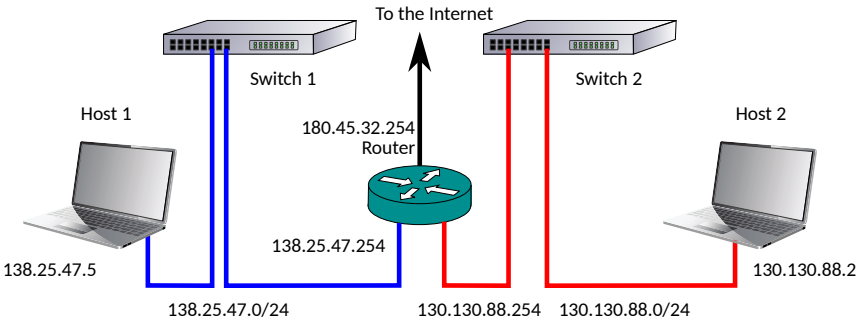
More Practice?

This is an online subnetting calculator you can play with for checking calculations:

`https://www.solarwinds.com/free-tools/advanced-subnet-calculator`

This may be useful in helping to check your binary calculations.

Routing



- Consider the internetwork shown above
- We have two separate Layer 2 (switched ethernet) networks. One is on the IP subnet 138.25.47.0/24, the other is on 130.130.88.0/24.
- What happens when Host 1 on the left wants to send an IP datagram to Host 2 on the right?

The routing process

- There are quite a few steps involved. First, we are going to assume that layer 2 (MAC) addresses are mutually known to all hosts in each network. The mechanism which accomplishes this will be discussed in Module 5.
- First, Host 1 constructs an IP datagram. Source/destination IP addresse will be 135.25.47.5 and 130.130.88.2, respectively
- Where should we send this frame? To decide, Host 1 will consult its **routing table**

The routing process

- The routing table is a list of rules. On host 1 it might look like this:

```
user@host1:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         138.25.47.254   0.0.0.0         UG    0      0        0 eth0
138.25.47.0     0.0.0.0         255.255.255.0   U     1000   0        0 eth0
```

- The first entry is the *default route* - anything which does not explicitly match any other entry should be sent to the specified router (gateway).
- The second entry is our own local network - we do not need a gateway to reach it.
- Host 1 searches this table for any rules which match our destination IP address

The routing process

- For each row in the table, the router performs a bitwise AND between the destination IP address and the netmask in the table
- The result is the network prefix or network address. The router then checks the destination field of the table; any matches are recorded
- Finally, we complete the search through the table:
  - If there are no matches, we discard the packet and send an ICMP error message back to the source (no route to destination)
  - If one entry matches, that will be the one that is used
  - If multiple entries match, the one with the longest network prefix (number of ones in the netmask) will be used as it is the most specific

The routing process

- In our example, the first entry matches, even without checking the destination - it is the default route, and ALL destination IP addresses will match.
- For the second entry, the result of the bitwise AND is 130.130.88.0; this does not match the destination in this entry (137.25.47.0).
- Therefore, the first entry is used to forward the datagram. It says we should send it using interface `eth0` to router `137.25.47.254`.
- So, Host 1 creates a Layer 2 frame with its MAC as the source address and the router's left-hand interface MAC address as the destination MAC address
- It then transmits the frame to the router, which accepts the frame (since its interface address is the destination MAC address)
- However, the destination IP address is not one that belongs to the router. So the route table lookup process now repeats on the router.

The routing process

- The routing table on the router looks like this:

```
user@router:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         180.45.32.254   0.0.0.0         UG    0      0        0 eth2
138.25.47.0     0.0.0.0         255.255.255.0   U     1000   0        0 eth0
130.130.88.0    0.0.0.0         255.255.255.0   U     1000   0        0 eth1
```

- Again, we search through for matches. The first and third entry will match in this case.
- The first prefix is 0 bits long while the third is 24 bits long - so the third is used.
- A new layer 2 frame is now constructed with the router's right-hand interface as the source MAC and Host 2's MAC as the destination.
- The frame is sent to Host 2, which accepts it (since the destination MAC address matches its own). The destination IP address of the decapsulated datagram matches Host 2's IP address; so Host 2 accepts the datagram