5 OSI Layer 3 – The Network Layer

5.1 Goal of this Section Chapter

The goal of this section is for readers to learn what the Network layer of the OSI model is and what "interfaces" it uses to higher and lower layers in the OSI model. We'll learn what matters about the Network Layer and what services it provides. We'll also cover Internet Protocol (IP) which is *the* Layer 3 technology, and some basic tools for monitoring and debugging Layer 3 issues.

5.2 Purpose

The network layer is about getting coherent clumps of bits (packets) delivered between known destinations. So far this is similar to the Data Link layer. The key difference is that while the Data Link layer does this at a local small scale, the network layer conveys packets of data between destinations at a global level.

5.2.1 Interfaces Up and Down

The interface to the Transport Layer (above) is a coherent block of data (which may be larger than will fit in a single packet) that shares fate, and the layer 3 addressing information to cover where to send it.

The interface to the Data Link Layer (below) is a packet of data, but now with the Layer 3 addressing information included (ready to become the payload of a frame) and the Layer 2 addressing information** for where to send it.

The strict layering of the model is breaking down again under close inspection. The interesting question this time is: "Layer 2 has an addressing system and Layer 3 has an addressing system, so which layer is responsible for mapping between the two?" The answer for IP and Ethernet is the **Address Resolution Protocol (ARP) – covered later in this chapter. ARP runs at layer 2 (it uses MAC addresses for source and destination identification) but its job sits between Layer 2 and 3.

5.3 Services

Layer 3 provides the following services:

5.3.1 Global packet delivery

Layer 3 delivers packets to any destination, globally, including a standardised global addressing system and also route discovery and route selection.

5.3.2 Packet splitting and packet level error detection

Layer 3 allows higher layers to provide more data than fits within a single Layer 2 packet, and handles splitting and recombining the data (called fragmentation). Layer 3 also provides some additional packet level error detection.

5.3.3 Quality of service and prioritisation flow control

Layer 3 provides some simple ways that packets can be flagged as more/less important in order for the system to prioritise getting more important packets to their destination.

5.3.4 Redundancy across layer 2 networks

Layer 3 creates a logical link between a global source and destination. In the same manner as we saw Layer 2 provide redundancy across individual physical links, Layer 3 provides redundancy across layer 2 networks.

5.4 What does "Better" mean for Layer 3?

The goal of the Network layer is to get packets to the right place. IP delivers packets fairly reliably (relying on the protocols underneath), but still does *not* guarantee delivery. IP is fundamentally a global addressing system, and relies on other layers to handle jitter/bandwidth/etc. Given that, for layer 3, better is that packets are routed across the network to their destination efficiently, making the maximal use of underlying resources.

5.5 Internet Protocol (IP) (RFC 791 +1349 +2474 +6864)

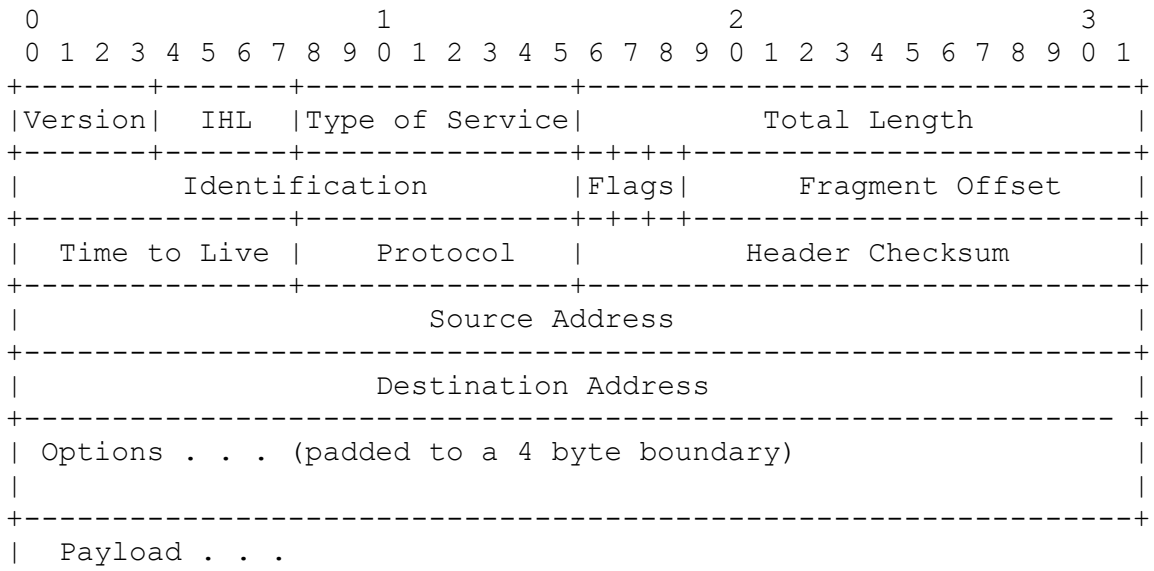
The original IP RFC came out in 1981 and defined a standard packet header format, which includes a version number. The Version reached at the time was 4, and IPv4 is the ubiquitous protocol in the internet.

In 1995, a new version, IPv6 was proposed in RFC 1883 (since updated several times and as of 2017 is defined in RFC 8200, obsoleting earlier RFCs). IPv6 solves a lot of problems that IPv4 has (mainly around scaling) and will likely *eventually* take over. However, it has been due to take over "within the next 10 years" since 1995 – don't hold your breath.

We'll cover IPv6 later in this course. For now, we'll focus on IPv4.

5.6 IPv4

The Ubiquitous IPv4 packet consists of a header and payload data. Let's unpack the header:



- **Version** (4 bits) is the version of IP. This is **4**. (15 is the maximum version number – We've got from 4 to 6 in 30 years, so I think we're OK at least until after we're dead).
- **IP Header Length** (4 bits) is the length of the IP header, measured in 32-bit words, allowing for a variable length header with different options and enabling something parsing the packet to jump straight to the payload.
- **Type of Service** (1 byte) This field is a set of bits that is intended to allow Layer 3 routers to identify and prioritise "important" packets. It has been used and reused in non-backwards-compatible (!) ways over the years, eventually being replaced with the "Differentiated Services" field in newer augmentations of the protocol . The ~globally ~consistent usage here is for the first 3 bits to denote priority (0-7 with higher priorities being more important). **Usually, this is set to 0.**
- **Total Length** (2 bytes) The Total length is the total length of the IP packet (header + payload), measured in bytes (so the maximum size of an IPv4 packet is 65,535 bytes).

- Fragmentation Information – the next 3 fields handle fragmentation of packets. IP can be handed a payload that is too large to fit into one packet (remember that IP packets are encapsulated in Ethernet Frames that have a maximum length!). IP breaks the data up into several packets and uses these fields to recombine the packets at the destination.
    - **Identification** (2 bytes) Set by the fragmenting source to a unique value for the source+destination pair, so that the destination can recognise parts of the same datagram.
    - **Flags & Fragment Offset** (2 bytes) The first flag is set to 1 if this is not the last fragment, and otherwise is set to the value in the header of the packet being fragmented (which allows for recursive fragmentation). The Fragment Offset is set to the offset of the fragment from the start (measured in number-of-64bit-words). Note that this means that IP packet payloads are always fragmented along 8 byte block boundaries.
- **Time To Live** (1 byte) This is set by the source, and decremented (decreased by 1) by every IP router that forwards the packet on (not by Layer 2 switches, that don't look at the Layer 3 info!). When the TTL hits 0, the packet is dropped – a simple way to prevent infinite routing loops.
- **Protocol** (1 byte) The protocol identifier identifies the protocol in use (and hints how to parse the payload) for a bunch of well-defined IP protocols. For example, Open Shortest Path First (OSPF) that we'll look at next section is protocol 0x59. These numbers are formally assigned by IANA.
- **Header Checksum –** This is a checksum over the IPv4 Header only. It is the 16-bit one's-complement of the one's-complement sum of the other 16-bit words in the header – meaning that it is set such that summing all the 16-bit words in the IPv4 header including the checksum and then adding the carry bits to the sum as single bits gives 0xFFFF. Note that in actuality, this field is not that useful, and is often not used in order to save processing time and instead just set to 0x0000.
- Addressing information
    - **Source Address –** The source IPv4 address of the packet.
    - **Destination Address** – The destination IPv4 address of the packet.
- **Options** – There are various optional additional pieces of information, each defined by an initial byte that indicates the option (where each option is of a defined fixed length). We don't cover any of these in this course.

5.7 IPv4 Addressing and subnets

5.7.1 IPv4 addresses

IPv4 addresses are 4 byte addresses (32 bits). As humans we write these by standard as each byte written out in decimal with dots between them, for example: 172.19.9.63. This is called dotted decimal format.

5.7.2 Subnets and subnet masks

IPv4 addresses are arranged carefully into subnets along Layer 2 network boundaries in order to make global routing easier. An IP subnet is simply a set of addresses that are the same for the most significant (left-most) part of the address. The length of the common-to-all-addresses-on-the-subnet part is called the subnet mask.

Subnets are notated either with the length of the mask in bits or with the mask written out separately. For example: 172.19.0.0/16 is the same as 172.19.0.0, mask 255.255.0.0.

Note that non-byte-aligned masks are also fine (For example: 172.19.16.0/20 or 172.19.16.0, mask 255.255.240.0).

5.7.3 Arranging Addresses into Subnets

For any given layer 2 network (subnet):

- Every IP address in that subnet must match the subnet (up to the mask).
- Every IP address in the world that matches the subnet up to the mask must be on that subnet.

Following this rule makes broader routing packets to destinations convenient (possible!). Rather than every place on the internet needing to know exactly how to reach every single address, you can now know that all addresses that start '123.' are over in that direction… We'll cover how routing works in more detail later.

5.7.4 Special subnets

When initially parcelling out subnets, IANA used byte boundaries, having Class A subnets (X.0.0.0/8), Class B subnets, (X.Y.0.0/16) and Class C subnets (X.Y.Z.0/24). The byte-boundary restriction is long gone, but the names are still around for those subnets.

A side note, IANA was handing out these back before the internet really took off at scale, and 1/255 or 1/65535 of the whole internet seemed like reasonable portions to hand out. Winners of Class A subnets include the Ford Motor Company (19.<anything>) and the US Department of Defense (which has 13 different Class As, so around 5% of the entire internet's address space).

There are some special subnets used within IPv4.

- **Private addresses.** These may not be publicly routed. You can use these yourself to build networks where you can route internally, but you may not advertise these outside your network.
  - 10.0.0.0/8 (Class A)
  - 172.16.0.0/12
  - 192.168.0.0/16 (Class B)
- **Multicast addresses** – used for multicasting (see below)
  - 224.0.0.0/4
- **Future reserved**. These are reserved by IANA for future use.
  - 240.0.0.0/4
- **Loopback addresses**. These are used by a host to refer to itself.
  - 127.0.0.0/8
- ~ Everything else is a global unicast address.
  - This isn't quite true. There are actually a load of other small subnets reserved for other things (test networks, usage in documentation so you don't actually put someone's real IP address in your docs, etc.)

5.8 Unicast/Multicast/Broadcast

Layer 2 is by default broadcast and then reduced down. IP is by default unicast (single destination), with specific addresses and protocols used for multicast/broadcasting.

5.9 Broadcast

How do you broadcast at Layer 3? Very, very carefully. True broadcast (send to every destination) is not a thing that people want to allow at a global scale! Broadcast at Layer 3 only occurs at a local subnet level (routers will not forward on broadcast packets). Each subnet has an associated broadcast address, which is the highest possible address number in the subnet (the IP address formed by taking the subnet mask and filling the remaining bits of the address with 1s). For example, for the subnet 172.19.9.0/24, the broadcast IP address is 172.19.9.255.

5.10 Multicast

There are circumstances where you want to broadcast one to many at global scale. For example an IP TV station does not want to have a single unicast stream to every single user, as their initial links will be flooded with millions of identical packets bound for different destinations.

For IP multicast, endpoints subscribe to Groups (identified by a particular multicast destination address) using multicast routing protocols. Endpoint hosts use Internet Group Membership Protocol (IGMP) to (un)subscribe for groups for a particular source at their local router. Routers use Reverse Path Forwarding or other multicast protocols to join/prune themselves upstream to the multicast forwarding tree root. We don't cover IP multicast in detail in this course.

5.11 (Unicast) IP Routing

Every IP enabled device also has a set of IP routing rules that cover where to send IP packets. No router needs to know the entire route through the network – just where to go *next* (the next hop).

Most rules are of the format "To reach destinations in subnet X, send to destination with IP address Y." Each host or router will also have some rules of the form "To reach destinations in subnet X (*that I am also a part of)* then go out a particular physical interface" More precise rules override less precise rules, and each device also has a catch-all rule (default route) that covers where to send anything that isn't matched by another rule.

IP configuration on a typical computer or endpoint physical interface therefore typically consists of:

- IP address (my IP address on this interface)
- Subnet mask (the mask of the local subnet on this interface – which gives me the set of IP addresses that I can reach using Layer 2 out of this interface)
- Default gateway (the IP address of the local "default" router that I can send everything else to at Layer 3, and it will sort out sending it onwards).

The default router will itself have a default router upstream, and so on and so on, until you reach the core of the internet, where there are a set of huge routers that know how to direct all traffic (to each other or back down to more precise areas).

You can see some examples of routing table, by looking at the ones on your computer. To view your routing table, use the **route print** command from a Windows command prompt, the **netstat -rn** command from a Mac, or the **route** command on Linux.

Before we move on to an example of how Ethernet and IP routing works, we need to cover one more protocol.

5.12 Address Resolution Protocol (ARP) (RFC 826)

Address Resolution Protocol (ARP) is a protocol that is used to determine the MAC address (Data Link layer) associated with a particular IP address. As such, while ARP operates at the Data Link layer (ARP packets are MAC addressed, not IP addressed), it sits between the layers and doesn't fit nicely in either.

In abstract, ARP works as follows:

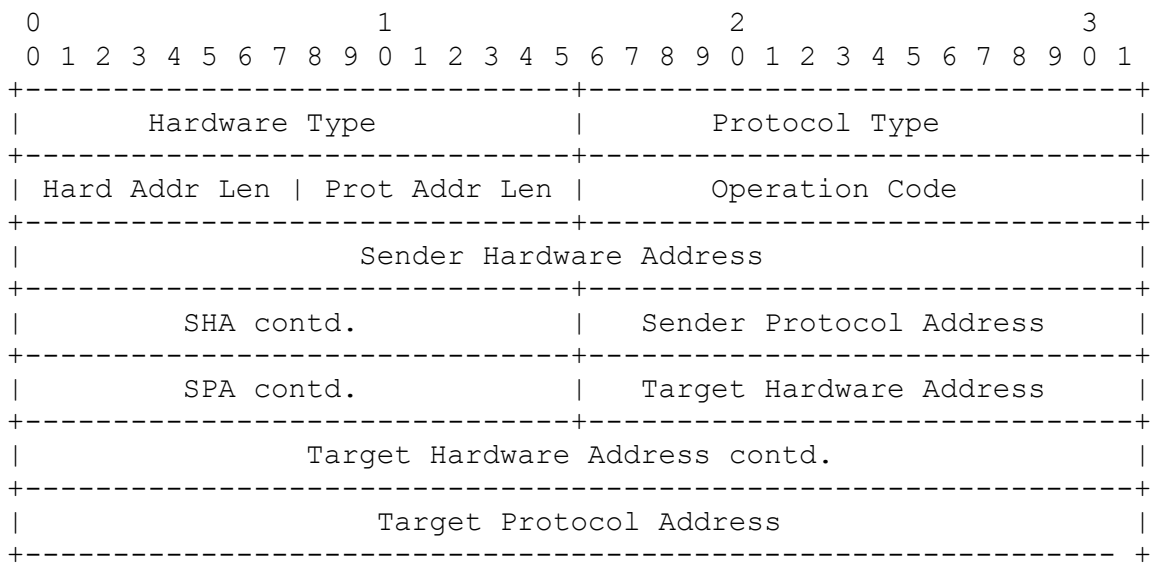
1) A node that wants to send a packet to an IP address that is on a local subnet broadcasts a request at layer 2 asking which MAC address owns that IP address. The node that owns the IP address responds broadcasting the mapping to everyone (not just the requester).
2) When a node is new onto a layer 2 network, it broadcasts a response (a **gratuitous** ARP) with its IP addresses without waiting for a request.

In addition, to avoid having to resend ARP messages all the time, hosts maintain a local ARP cache with the latest mappings.

ARP messages are broadcast on a layer 2 network, with the Ethernet header values

- Destination MAC set to FF:FF:FF:FF:FF:FF (Broadcast)
- Source MAC set to the source MAC address
- Protocol Type set to 0x0806 (ARP).

After the Ethernet Frame header, the ARP message has the following fields.



As a side note: ARP is a good example of a protocol that was built to solve an immediate problem (converting IPv4 addresses to MAC addresses), but tried to be future compatible and generic. This happens a lot – engineers try to predict what *might* be useful. Sometimes it's *really helpful*. Other times, not and you end up with a bunch of pointless fields.

- **Hardware Type** (2 bytes). Always set to 0x0001 (Ethernet)
- **Protocol type** (2 bytes). Always set to 0x0800 (IPv4)
- **Hardware address size** (1 byte). The hardware address size in bytes. Always set to 6 (MAC addresses are 6 bytes long).
- **Protocol address size** (1 byte). The protocol address size in bytes. Always set to 4 (IPv4 addresses are 4 bytes long).
- **Op Code** (2 byte). Set to 1 to indicate a query and 2 to indicate a response.
- **Source Hardware Address** (6 bytes). The source's MAC address. In a response, this is the answer!
- **Source Protocol Address** (4 bytes). The source's IP address.
- **Target Hardware Address** (6 bytes). Ignored on a request. On a response, the fields are copied from the Source values in the request.
- **Target Protocol Address** (4 bytes) On a request, this is the IP address being looked up. On the response, the IP address from the Source in the request.

The packet capture **05_ARP.pcapng** stored alongside these notes contains an ARP request and response.

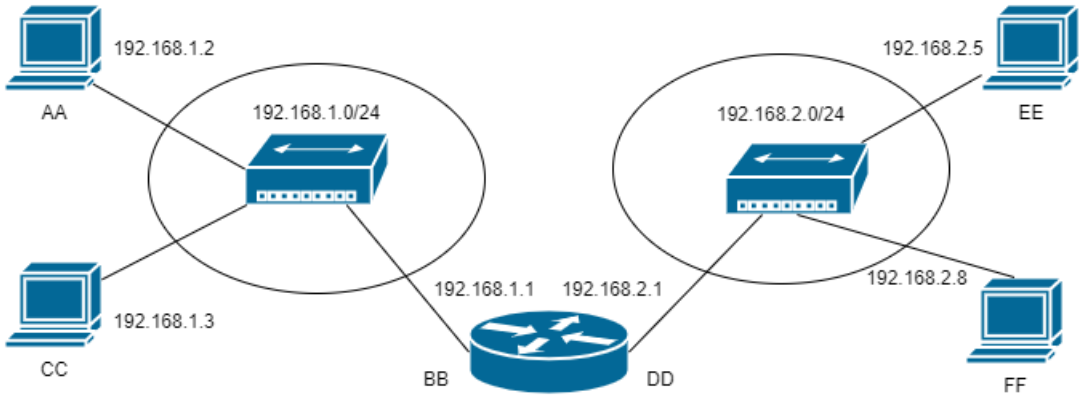
You can look at the arp-IP mappings cache on your computer using the arp command. **arp -a** displays the arp tables on Windows, Linux and Mac.

5.13 IP Routing Example.

The following worked example shows how IP routing rules, ARP, and Ethernet Switching work to send a payload of data across a network. This example goes through in quite a lot of detail as while

each step is simple, there are quite a large number of them going on which adds up to quite a complex whole.

5.13.1 The Network



In this network, there are 4 hosts connected to one router via two switches. The two Layer 2 switching domains are the subnets 192.168.1.0/24 and 192.168.2.0/24. Every interface has its IP address and MAC address shown (MAC addresses are shown as double-letters for ease of viewing – in reality they would be random 6-byte numbers.

5.13.2 The Routing rules

Host AA has the following routing table:

- Destination: 192.168.2.0/24 Next Hop: 192.168.1.1
- Destination 192.168.1.0/24 Directly connected – interface AA

The Router has the following routing table:

- Destination: 192.168.1.0/24 Directly connected – interface BB
- Destination: 192.168.2.0/24 Directly connected – interface DD

A good exercise is to work out what the routing tables on the other hosts look like.

5.13.3 The Data

The host at 192.168.1.2 wants to send a packet of data to the host at 192.168.2.8. We'll track the contents of the packet at each point as follows. (To keep things simpler, this example will only cover cover IP address and MAC address usage – ignoring the other IP and Ethernet header fields.)

Payload	IP Header: Source IP	IP Header: Destination IP	Eth Header: Source MAC	Eth Header: Destination MAC

5.13.4 The routing process

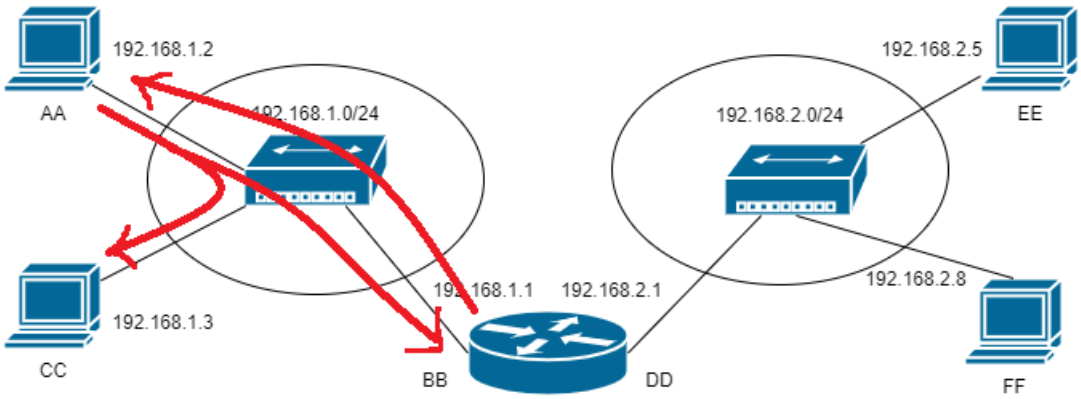
1. An application on Host AA wants to send a Payload of data to the application on FF. It passes it down to Layer 3 on AA.

Payload				

2. Layer 3 for Host AA adds the source and destination IP headers to the packet.

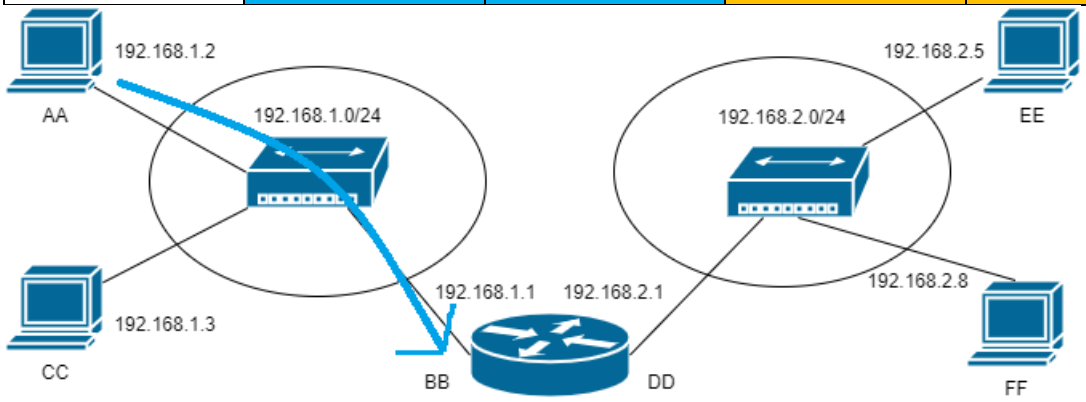
Payload	Src: 192.168.1.2	Dest: 192.168.2.8		

3. Layer 3 at AA looks up in the routing table to see where to send the packet.
    - Destination 192.168.2.8 ➔ Next Hop 192.168.1.1
    - Destination 192.168.1.1 ➔ Directly connected out Interface AA.
    - Ok – so we want to send this packet to 192.168.1.1 next, and to do that we send this packet out from interface AA.
4. What MAC address do we want to send this to? We don't know. Let's ARP!



5. AA broadcasts ARP asking who owns 192.168.1.1. CC ignores this as it does not. BB responds indicating that it owns 192.168.1.1.
6. Layer 2 at AA updates the MAC addresses into the packet (now frame) and sends it.
    - **Note:** The Layer 3 Destination is still 192.168.2.8. The Data Link Layer 2 destination is the Destination on this Layer 2 network. The destination MAC does not match the destination IP. This is fine and normal.

Payload	Src: 192.168.1.2	Dest: 192.168.2.8	Src: AA	Dest: BB

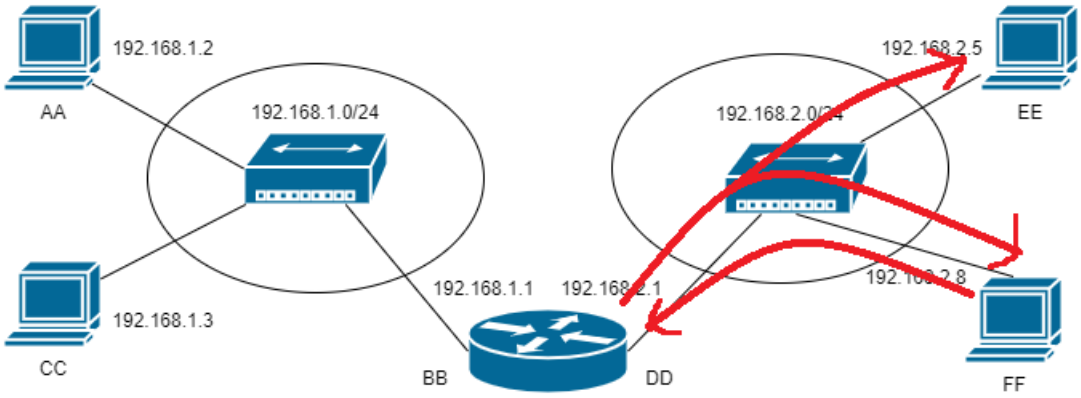


7. The packet transits network A and arrives at BB.
    - Note that it may also arrive at CC (the switch may broadcast it). However, CC will drop this at Layer 2 as the destination MAC address does not match.

8. Layer 2 at the router, interface BB receives the packet and it is addressed to BB. It passes the packet up to Layer 3

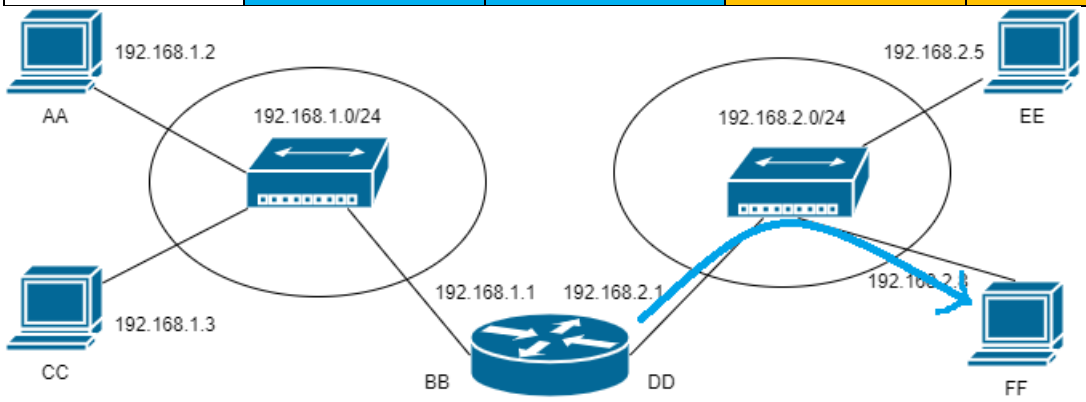
Payload	Src: 192.168.1.2	Dest: 192.168.2.8		

9. Layer 3 at the router looks at the IP address. It is **not** addressed to one of its IP addresses, so it needs to look up where to send it in its routing table.
    - Destination 192.168.2.8 ➔ Directly connected out Interface DD.
    - Ok – so we want to send this packet to 192.168.2.8 next, and to do that we send it out interface DD.
10. What MAC address do we want to send this to? We don't know. Lets ARP!



11. DD Broadcasts out an ARP request for 192.168.2.8. EE ignores this as it does not match. FF returns a response.
    - Note that the router is ARPing out interface DD only, as this is the interface that the IP address is in the subnet for.
12. Layer 2 at DD updates the packet (now frame) with the MAC addresses and sends it.

Payload	Src: 192.168.1.2	Dest: 192.168.2.8	Src: DD	Dest: FF



13. The packet transits the network and arrives at FF (and again possibly at EE, which would drop it.
14. Layer 2 at FF recognises the packet is addressed to it, and passes it up to Layer 3.

Payload	Src: 192.168.1.2	Dest: 192.168.2.8		

15. Layer 3 at FF recognises the packet is addressed to it, and passes it on up…

Payload				

16. The payload has arrived.

5.14 Problems / Troubleshooting / Tools

The most common failures at layer 3 are down to misconfiguration – using the wrong IP addresses, subnet masks, or routing rules. Note that at Layer 3 routing loops are bad, but not disastrous due to the Time-to-live (TTL) field.

The three classic network troubleshooting tools are:

- Ping – checks connectivity to an IP address
- Traceroute – checks each network hop in turn across a network to an IP destination
- Wireshark / tcpdump (all packets coming in/out of an interface).

You've already been using wireshark with these lectures. Trying using ping and traceroute from your own PC to see how you are connected to other IP addresses across the world.