Chapter 4: The Network Layer

ATNLP (layers) (PLaNeT-A)
Application
Transport
Network
Link
Physical

4.1 Introduction

- Primary role of routers → forward datagrams ('network-layer packets') from input links to output links
- Routers do not run Application or Transport-layer protocols, only network-layer
- Role of network-layer → 'move packets from sending host to receiving host'

4.1.1 Forwarding and Routing

- 2 NB network-layer functions: **forwarding** and **routing**
  - **Forwarding:** forwarding of a packet from a single router's incoming link to the outgoing link (of that same router) which corresponds to the appropriate path required to get to the next router on the transmission journey
    - The action of forwarding happens locally within a single router
    - = *"changing lanes on the highway during a roadtrip"*
      - Or *"taking a specific circle exit during a roadtrip".* Nice to think of circles as routers imo.
  - **Routing:** network-layer must determine appropriate route/path taken by packet as it's passed from router to router on its transmission journey. *Routing Algorithms* calculate these routes/paths (ie: path from host 1 to host 2).
    - The action of routing is a network-wide/collective process involving many routers.
    - = *"planning the entire overall route of a roadtrip"*
  - Router algorithm can be *centralised* (algorithm executes from central location and routers download info from this location) or *decentralised* (piece of algorithm executes locally on each router)

- Every router has a **forwarding table** that it uses in process of forwarding packets:
  - (1) examine packet header
  - (2) use header field to index forwarding table
  - (3) value at that index in forwarding table = local outgoing link within this router which the packet needs to be forwarded through
  - *Note:* the header field can either index an exact destination address, or an indication of the connection to which the packet belongs → **clarify**

How are forwarding tables in router configured?

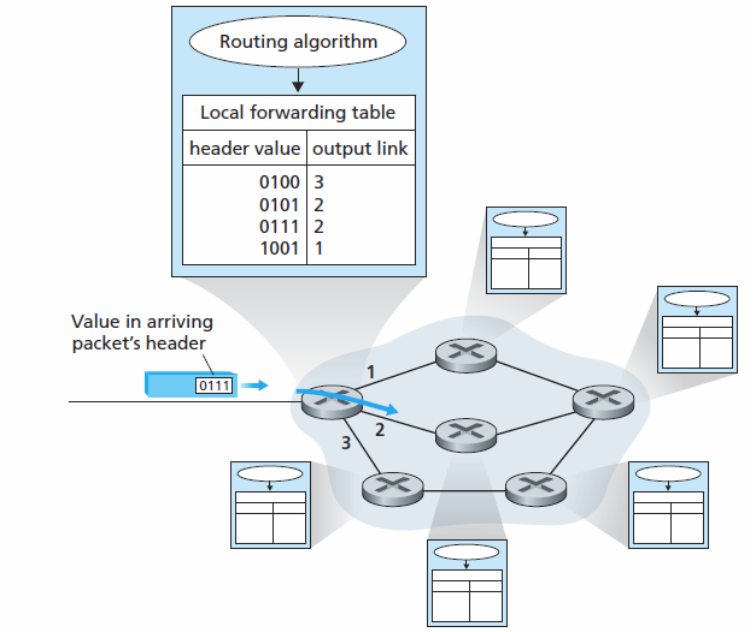


Figure 4.2 ♦ Routing algorithms determine values in forwarding tables

← Router receives *routing protocol messages* from Routing Algorithm, which it uses to configure forwarding table.

- Some more NB terminology:
  - **Packet switch:**
    - general device which transfers packet from input link interface to output link interface, based on packet header field
  - *Routers* and *link-layer switches* are <u>two different types of packet switch</u>
    - **Link-layer switch:**
      - forwarding decision based on link-layer field
      - therefore, <u>switches = link-layer (layer 2) devices.</u>
    - **Router:**
      - forwarding decision based on network-layer field
      - therefore, <u>routers = network-layer (layer 3) devices</u>
        - **But** must still implement link-layer (layer 2) protocols, since layer 3 depends on layer 2.
  - In this chapter, we use 'router' in place of 'packet-switch'

Connection Setup

- Apart from forwarding and routing, *connection setup* is an important network-layer functions in <u>some</u> computer networks
- Just like TCP requires handshake before data can be exchanged, *some network-layer architectures require routers to handshake → = connection setup*

4.1.2 Network Service Models

- What services does the network-layer offer us?

- **Network service model:** defines characteristics of end-to-end packet transport between end systems
    - Network service model for *the internet* (below):
    -

Network Architecture	Service Model	Bandwidth Guarantee	No-Loss Guarantee	Ordering	Timing	Congestion Indication
Internet	Best Effort	None	None	Any order possible	Not maintained	None

    - ^ there are advantages to this minimalistic design (discussed later)

4.2 Virtual Circuit and Datagram Networks

- Like transport-layer, Network-layer offer *connectionless* and *connection-oriented* services.
    - Network-layer connection-oriented service: *requires handshake*
    - Network-layer connectionless service: *no handshake required*
- Note: crucial differences between connection services offered by network and transport layers

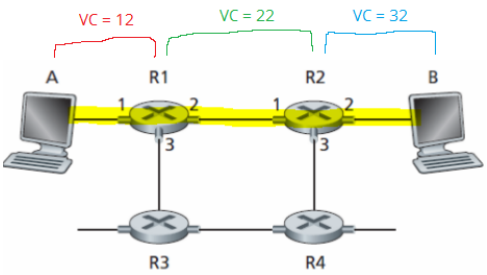
Network-layer connection services	**Transport-layer connection services**
Are host-host services for use by transport-layer	Are process-process services for use by application-layer
Connection-oriented service implemented in both routers and hosts	Connection-oriented service implemented at 'edge' of network in hosts

- **Virtual-Circuit (VC) network:** only provides connection-oriented service at network layer
- **Datagram network:** only provides connectionless service at network layer
    - Internet = datagram network

4.2.1 Virtual-Circuit (VC) network

- Only provides connection-oriented service at network-layer
- Uses network-layer connections called **virtual-circuits (VCs)**
- **What does a VC service consist of?:**
    - (1) a path → series of links & routers
    - (2) VC numbers → one for each router/link along path
    - (3) entries in forwarding table of each router along path
- VC packet has a VC number field in its header
- Since each router has a different VC number, each router must replace a packet's *VC number field* with a new VC number from its forwarding table

Simple VC network (below):



← numbers = *link interface numbers*

← A establishes VC with B
← path = A-R1-R2-B
← The 3 links on this path have VC numbers 12, 22, 32 respectively
← R1 updates initial VC from 12 to 22, R2 updates VC from 22 to 32

Incoming Interface	Incoming VC #	Outgoing Interface	Outgoing VC #
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

^ in this simple VC, R1 might have the following forwarding table (below):

- In a VC network, routers have to maintain **connection-state information** for ongoing connections.
    - Whenever a new VC is established across a router, new entry is added to forwarding table
    - Whenever a VC connection terminates, all the entries along its path are deleted

Why can't a packet keep the same VC number across routers?:

- **(1)** Replacing VC number across links reduces length of VC header field
- **(2)** Simplifies VC setup → prevents routers having to check with each other that the common VC isn't being used by another VC in the network.

3 phases in a VC:

- **(1) VC Setup**
    - Sending transport-layer passes receiver address to network-layer
    - Then, Network-layer:
        - plans out transmission path
        - determines VC number for each link along path
        - adds an entry in each router's forwarding table along path
        - Reserves resources (buffers etc) along path
- **(2) Data transfer**
    - Packets flow along VC
- **(3) VC teardown**
    - Forwarding tables updated → entries for VC removed

Connection setup: TCP vs VCs

- In TCP handshake, only 2 hosts involved
- In VC connection setup, every router along path (between 2 hosts) involved
    - In a VC, hosts send **signaling messages** to initiate/terminate VCs

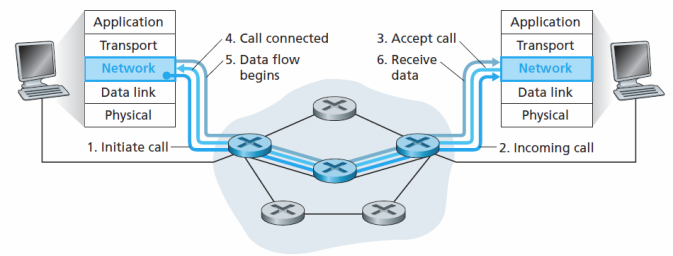
- 

  Figure 4.4 ♦ Virtual-circuit setup

4.2.2 Datagram Networks

- Each time a host sends a packet, it first stamps the packet with the address of the destination host (as a header field), then pops the packet into the network
- As a packet is transmitted from src to dest host, it passes through a series of routers
  - Each router has a forwarding table, which maps destination addresses to link addresses, which it uses to forward packets
- No VC setup and no VC **connection** state information kept by routers, since no VCs involved…
  - BUT routers still maintain **forwarding** state information in their forwarding tables
    - This happens very slowly/inconsistently though → every 1-5 min or so
- Since datagram network forwarding tables can change at any time, packets can arrive **out-of-order**

Datagram network lookup operation:

- Suppose destination addresses = 32 bits (= length of dest address in IP datagram)
  - = 4 billion possible addresses → not possible for forwarding table to have an entry for each one
- Now, suppose router has 4 links: **0, 1, 2 and 3**
- Packets can be forwarded to link interfaces as follows:
  -

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

  - *Clearly we don't need a separate table entry for each possible address!*

- We can simplify the above forwarding table to:

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

- With this style of forwarding table, the router matches a **prefix** of the packet's destination address with a table entry, and forwards the packet to the outgoing link which corresponds the match.
    - **Problem:** Destination address could match more than 1 entry in the table → which link to forward packet through?
    - **Solution:** *longest prefix matching rule* → the longest matching entry is the one whose link needs to be forwarded through

Datagram or VC network: why?

Internet (datagram)

- data exchange among computers
    - "elastic" service, no strict timing req.
- many link types
    - different characteristics
    - uniform service difficult
- "smart" end systems (computers)
    - can adapt, perform control, error recovery
    - ***simple inside network, complexity at "edge"***

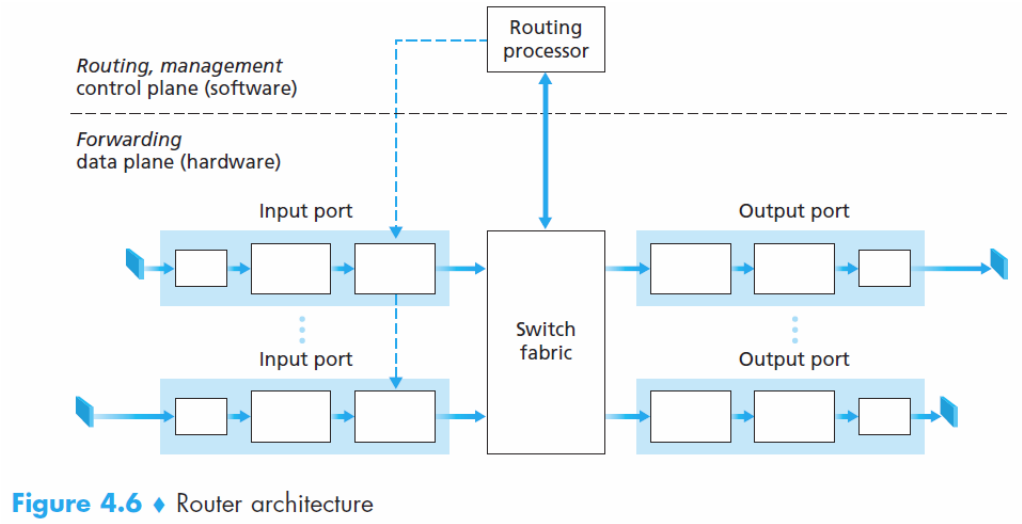
ATM (VC)

- evolved from telephony
- human conversation:
    - strict timing, reliability requirements
    - need for guaranteed service
- "dumb" end systems
    - telephones
    - ***complexity inside network***

4.3 What's inside a router?

- Namely, how does a router's forwarding function work?
    - (recall: forwarding function = transfer of packets from router's incoming links to appropriate outgoing links in that router)
- We'll use *forwarding* and *switching* interchangeably

Generic router architecture (below):



- Figure 4.6 ♦ Router architecture
    - **Input ports** (traffic circle entrance):
        - Most importantly, lookup function is performed here
            - Control packets tell the routing processor to consult the forwarding table when packets enter input ports
        - Also performs some physical & link-layer functions
    - **Switch fabric** (traffic circle):
        - Connects input ports to output ports → 'network inside the router'
    - **Output ports** (traffic circle exit):
        - Receives packets from switch fabric and forwards them through appropriate outgoing link
        - If a link carries traffic bidirectionally, input/output ports are on same little circuit-card
    - **Routing processor:**
        - Executes routing protocols (follows instructions from control packets)
        - Maintains forwarding tables
        - Network management functions

Router's forwarding function:

- Uses input ports, output ports, switch fabric
- Router's physical forwarding functions = **router forwarding plane** → hardware
- Routing processor commands the router forwarding plane based on control packets it receives
- **Input port =** traffic circle entrance
- **Switch fabric =** traffic circle
- **Outgoing port =** traffic circle exit

4.3.1 Input Processing

- Router receives packet at input port
- Router processor observes packet and determines (using forwarding table, which it's also in charge of updating) which outgoing port it should go through (via the switch fabric)
- If switch fabric is too busy, packets may have to be queued in a buffer in input port

- Usually, input ports store shallow/shadow copies of the forwarding table, so the input ports can quickly determine the correct outgoing port without having to consult the router processor
    - → helps to avoid centralised processing bottleneck

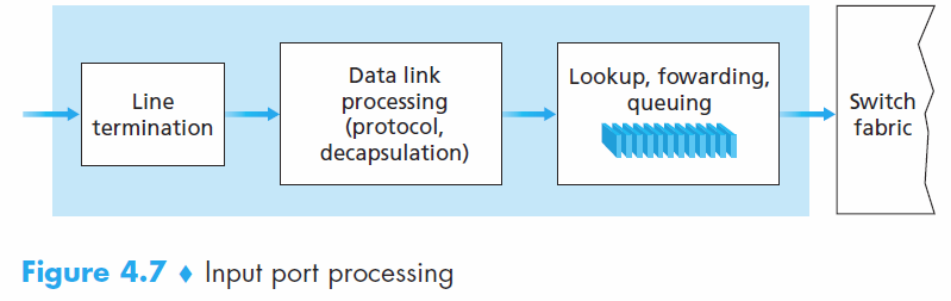
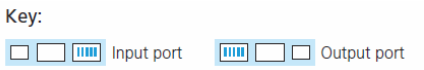
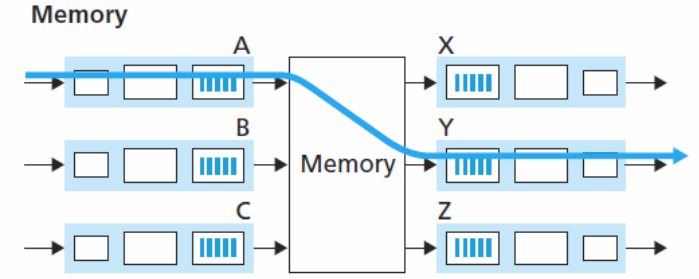


Figure 4.7 ♦ Input port processing

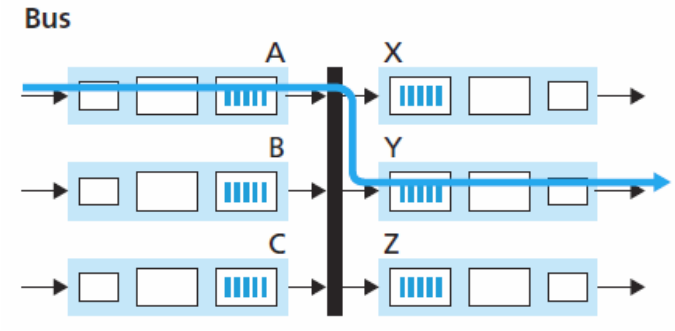
- As alternative to DRAM/SRAM, Ternary Content Address Memories (TCAMs) are often used for super-fast lookup
    - → can return a 32-bit address from forwarding table in essentially constant-time
- Remember, router processor must also update relevant header fields of packets which pass through it (ie: time-to-live, checksum, etc.)
- **"Match plus action":** abstraction of forwarding function → many network devices follow similar logic

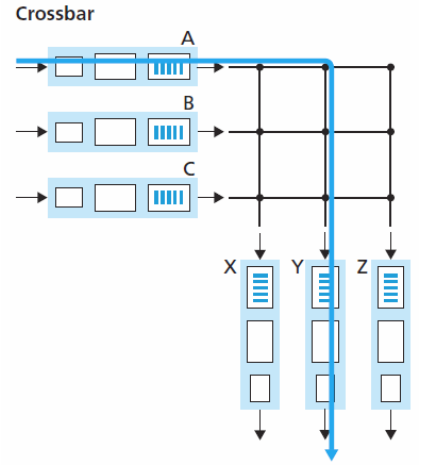
4.3.2 Switching

- Packets are switched/forwarded through switch-fabric
- 'Heart of router'
- **Switching rate**: rate at which packets can be transferred from input ports to output ports, via switch-fabric
- **Methods**:
    - 

    - **Switching via memory:**
        - Old days: CPU directly switches between input & output ports using memory to temp store the packet
        - Today: processing takes place on line cards using port-buffers to temp store the packet
        - 

    - **Switching via a shared bus:**
        - Inport port transfers packet to output port over shared bus

        - No router processor intervention
        - 

    - **Switching via an interconnection network (crossbar):**
        - Overcome limitations of shared bus
        - Packets can be forwarded in parallel → less queueing
        - 

4.3.3 Output Processing

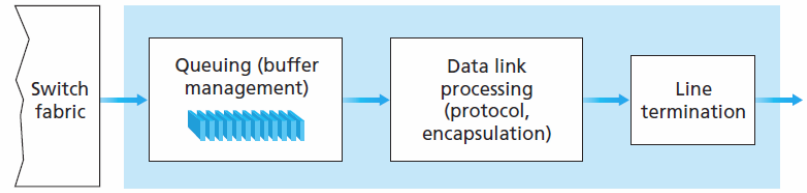
- Must transmit packets from output port buffers to the appropriate output link
- Includes selecting & de-qeueing
- 

  Figure 4.9 ♦ Output port processing

4.3.4 Where/when does queueing occur?

- Packet queues can form at input/output ports
- **Outport port queueing**: When arrival rate of packets into switch-fabric exceeds output line speed of switch-fabric, packets must be buffered
    - → switch-fabric can't send out data fast enough
    - **Packet scheduler** must choose one packet from the queue/buffer to send to the outgoing link
        - Can be done FCFS or WFQ (weighted fair queueing)
        - Packet scheduler makes *quality-of-service guarantees*
    - If buffer/queue is full, must either drop packet (**drop-tail** policy) or remove another from queue

- Can also drop packets before buffer is full → congestion signal to sender
- **Active Queue Management (AQM) →** packet dropping and marking policies
  - Famous AQM = **Random Early Detection (RED)**
  - → maintains threshold for length of output queue

- **Input port queueing:**
  - → switch-fabric can't receive data from inport ports fast enough (and run out of buffer memory)
  - **Head-of-the line (HOL) blocking:** a queued packet in an input queue must wait for transfer thru switch-fabric (even though its output port is free) because it is blocked by another packet at the head/front of the line.

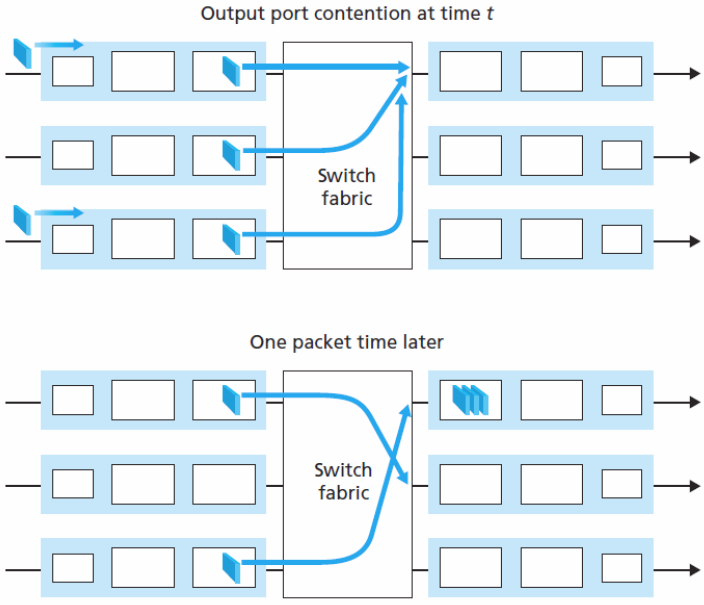


Figure 4.10 ♦ Output port queuing

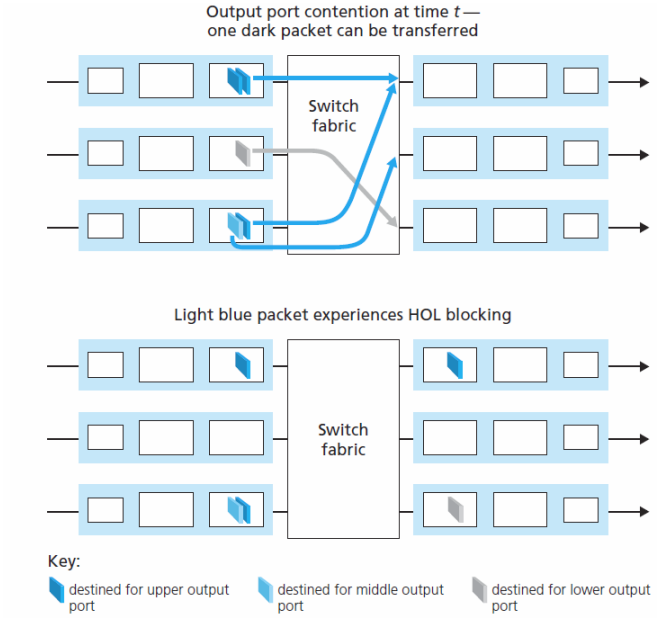


Figure 4.11 ♦ HOL blocking at an input queued switch

4.3.5 The Routing Control Plane

- Blah blah ending off section nothing important

4.4 The Internet Protocol (IP): Forwarding and Addressing in the Internet

- *Internet addressing & forwarding* = important components of Internet Protocol (IP)
- Two versions of IP in use today: *IPv4* and *IPv6*
- 3 main components of Internet
  - IP protocol
  - Routing Protocols
  - ICMP protocol

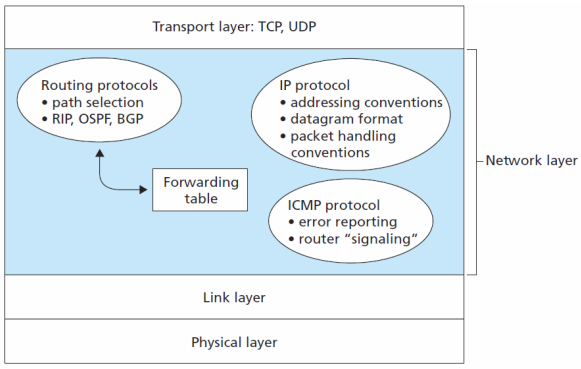


Figure 4.12 ♦ A look inside the Internet's network layer

- IP datagram has 20 bytes of header
- TCP segment has 20 bytes of header
- TCP/IP datagram = 40 bytes of header, plus data/msg to be sent

4.4.1 Datagram Format

- Datagram = network-layer packet
- IPv4 datagram (below):

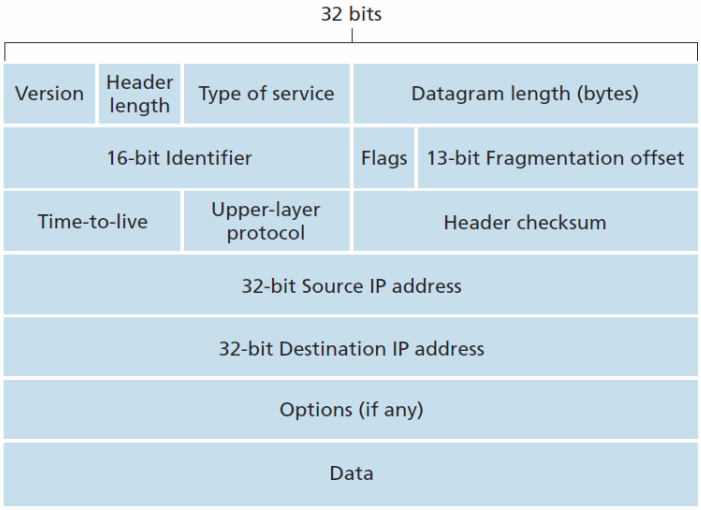


Figure 4.13 ♦ IPv4 datagram format

- Version number:
  - 4 bits
  - Specifies IP protocol version
  - Tells router how to determine rest of datagram
- Header length:
  - Specifies where in the datagram the 'Options' data should begin
- Type of service (TOS):
  - Specifies whether datagram requires low delay, high throughput, etc.
- Datagram length:
  - Total length (in bytes) of datagram → header + data
  - Since 16 bit field → max length = 65535 bytes, but usually not larger than 1500 bytes

- Identifier, flags, frag. Offset:
    - Pertain to ‘IP fragmentation’
    - Not in IPv6
- Time-to-live (TTL):
    - Ensure datagrams do not circulate forever
    - Decrement counter each time processed by new router → datagram dropped when TTL = 0
- Protocol:
    - Specifies which transport-layer protocol the data portion should be passed to
    - 6 = TCP, 17 = UDP
    - *‘Protocol number connects network and transport layer’*
        - *(and port number connects transport and application layer)*
- Header checksum:
    - Used by router to detect bit errors in received headers of received IP datagrams
    - Header checksum computed by summing bits in header using 1’s complement arithmetic
    - If router finds that this sum != checksum, discards the datagram
    - *Why is there error detection at network and transport layer levels?*
        - Answer = modularity. ie, if implement TCP into a different network architecture which doesn’t have error detection at network-layer, TCP would luckily have its own (transport-layer) error detection
- Source and destination IP addresses:
    - Self-explanatory
- Options:
    - Allow IP header to be extended
    - But, complicated since variable length
    - Thus not put in IPv6
- Data (payload):
    - Contains messages (ie: TCP or UDP transport-layer segment) to be delivered to destination

IP Datagram Fragmentation:

- Different link-layer protocols can carry different-sized network-layer packets
- Each IP (network-layer) datagram is encapsulated within a link-layer frame, so the
- **Maximum Transmission Unit (MTU) =** Max amount of data that a link-layer frame can carry.
- The MTU of a link-layer protocol places a hard-limit on the amount of data an IP datagram can hold, since IP datagrams have to fit into link-layer frames
- **What does a router do if it finds that an IP datagram is bigger than MTU?**
    - It can’t just squeeze the big IP datagram into a link-layer frame
    - **Solution:** *fragment the IP datagram into multiple smaller IP datagrams* ***(fragments)****, encapsulate each one of the smaller datagrams in a separate link-layer frame, and send each frame through the appropriate outgoing link*
    - Fragments must be reassembled before they reach the transport-layer at the destination.
        - TCP and UDP expect to receive complete, unfragmented segments from the network-layer
    - Reassembling of datagrams takes place in the end systems (hosts) - NOT in the router

        - When a host receives a series of datagrams from the same source, it must find out whether they are fragments or originals
    - Hosts use the *identification, flag* and *fragmentation offset* fields from the IP datagram header to reassemble fragments into the original datagram.
    - When a router fragments a segment, each fragment is stamped with the src address, dest address, and identification number of the original datagram
    - Then, hosts can see that if multiple datagrams have the same identification number, they must be fragments of an original datagram
    - Since IP is unreliable, some fragments might not arrive.
    - To ensure host has received up to the last fragment:
        - Flag bit of last segment set to 0, all others = 1
        - Offset field tells the host if the segment is the right size, and helps with correct ordering of the fragments
    - At dest host, payload of datagram is only passed to transport layer (TCP or UDP) when the original datagram has been fully reassembled. If it is not assembled properly, some transport-layer protocols will discard it. TCP will simply tell the source to retransmit the original data

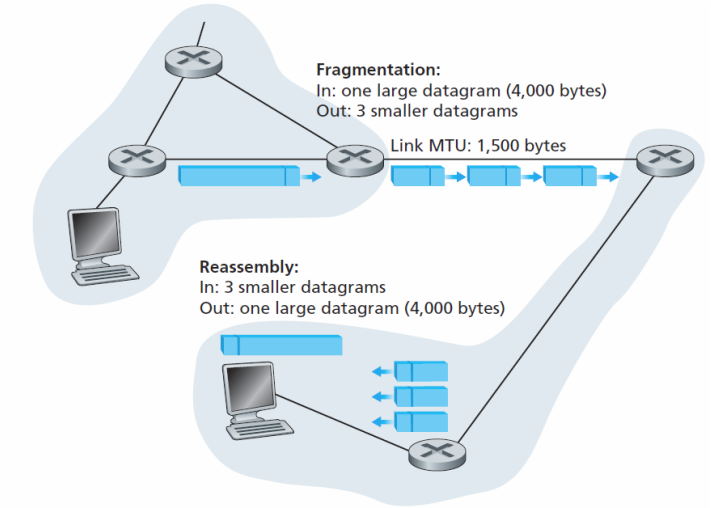


Figure 4.14 ♦ IP fragmentation and reassembly

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

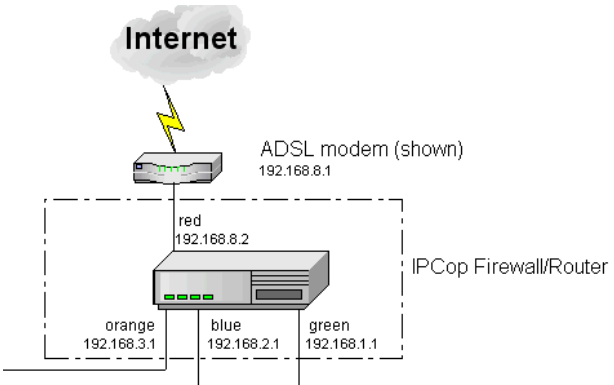
Table 4.2 ♦ IP fragments

- Cons of fragmentation:
    - Complicated
    - Opens vulnerability to DoS attacks

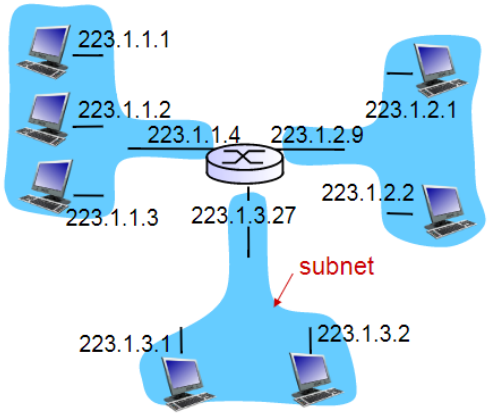
- IPv6 does not include fragmentation → streamlines IP packet processing and makes IP less vulnerable to attack

4.4.2 IPv4 Addressing

- **Interface** = boundary between host (or one of router's links) and physical link



- Routers have multiple interfaces since multiple in/outgoing links
- IP requires every *interface* of every host and router to have an IP address
- Each IP address 32-bits long (4 bytes)
    - 2^(32) possible IP addresses (around 4 bil)
- IP addresses written in **dotted-decimal notation:**
    - Each byte written in decimal form and separated by dot
    - ie: 193.32.216.9 → 193 is first 8 bits (byte), 32 is next 8 bits (byte), etc.
- **Subnet:**
    - Devices within a network that can physically reach each other without consulting the router
- IP addressing assigns addresses to subnets:
    - ie: if IP address is 223.1.1.0/24, '24' tells us that the left-most 24 bits of the 32-bit address define the subnet address
    - So for an interface to be part of this subnet, its IP address must be of the form 223.1.1.*xxx*



network consisting of 3 subnets

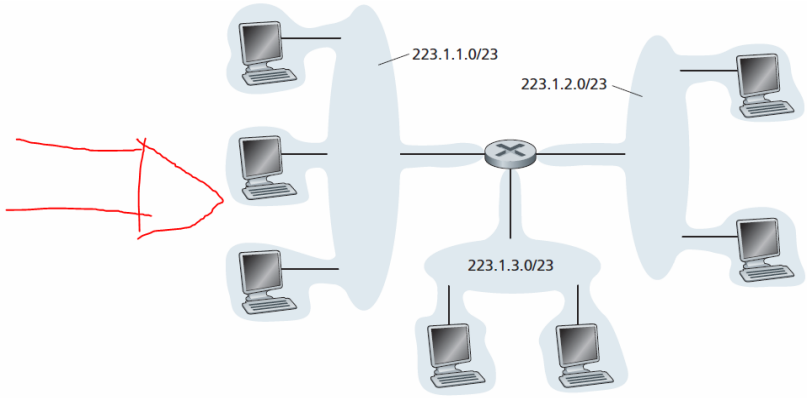


Figure 4.16 ♦ Subnet addresses

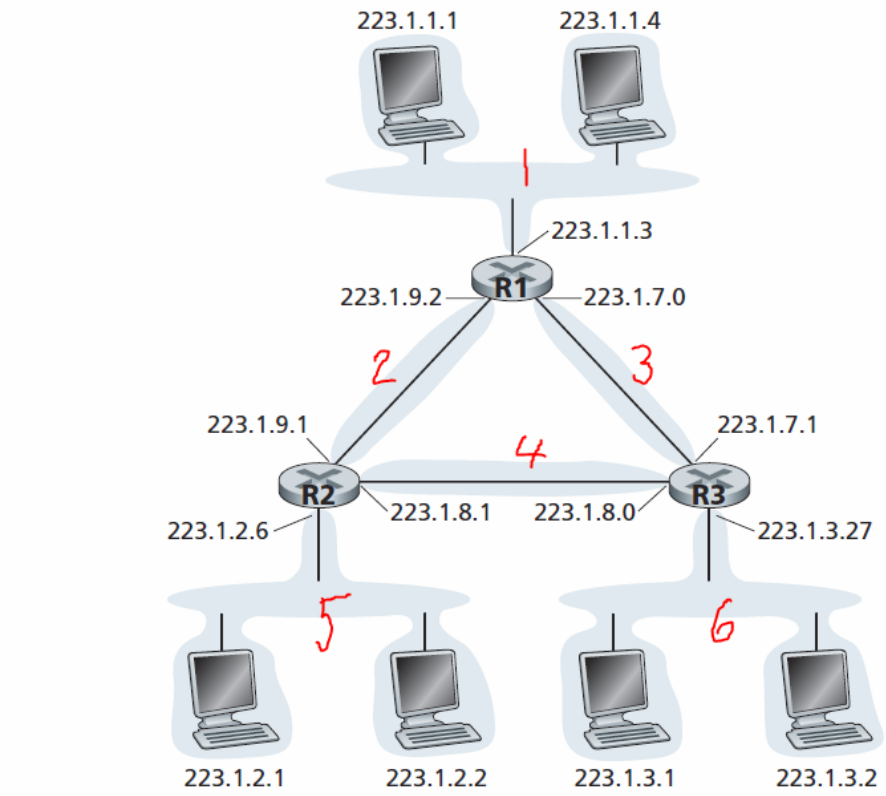


Figure 4.17 ♦ Three routers interconnecting six subnets

- **Classless Interdomain Routing (CIDR):**
  - Internet's address assignment strategy
  - Supports subnet addressing:
    - If IP address if a.b.c.d/x, the first x bits of the 32-bit address indicate the subnet address within the organisation, while the leftover (32-x) bits uniquely identify each interface in the subnet
      - These first x bits = most significant bits = **network prefix**
      - Router on other side of world only needs to consider the network prefix
- **Classful addressing:**
  - Before CIDR, network prefixes had to be either 8 (class A), 16 (class B), or 24 (class C)

- **IP broadcast address:**
  - = 255.255.255.255
  - If a host sends a datagram to this address, the datagram gets delivered to all hosts on the subnet

Obtaining a block of IP addresses:

- Organisation can obtain block of addresses from ISP in order to create its own subnet
- eg:

ISP's block	200.23.16.0/20	11001000 00010111 00010000 00000000
Organization 0	200.23.16.0/23	11001000 00010111 00010000 00000000
Organization 1	200.23.18.0/23	11001000 00010111 00010010 00000000
Organization 2	200.23.20.0/23	11001000 00010111 00010100 00000000
. . .	. . .	. . .
Organization 7	200.23.30.0/23	11001000 00010111 00011110 00000000

- **But how do ISPs get blocks of addresses?**
  - Internet Corporation for Assigned Names and Numbers (ICANN) allocates blocks of IP addresses (and also manages root DNS servers)

Obtaining a Host Address: the Dynamic Host Configuration Protocol (DHCP):

- Once organisation has a block of addresses, it can assign individual IP addresses to its host and router interfaces
- DHCP automatically allocates IP addresses to hosts (can be done manually too tho)
- Allows for **temporary IP addresses** (host gets new IP each time it connects to network)
- DHCP = 'plug-and-play' protocol

DHCP 4 step process:

- **(1) DHCP server discovery:**
  - Host connects to network and sends **DHCP discover message** to 255.255.255.255 (ie to entire subnet/network) via UDP port 67
- **(2) DHCP server offers:**
  - DHCP servers on network receive discover message, send **DHCP offer messages** to 255.255.255.255, containing proposed IP addresses for new host
- **(3) DHCP request:**
  - Host chooses from the proposed IP addresses, sends **DHCP request message** to chosen DHCP server, containing config parameters
- **(4) DHCP ACK:**
  - Server responds to DHCP request with ACK, confirming config parameters

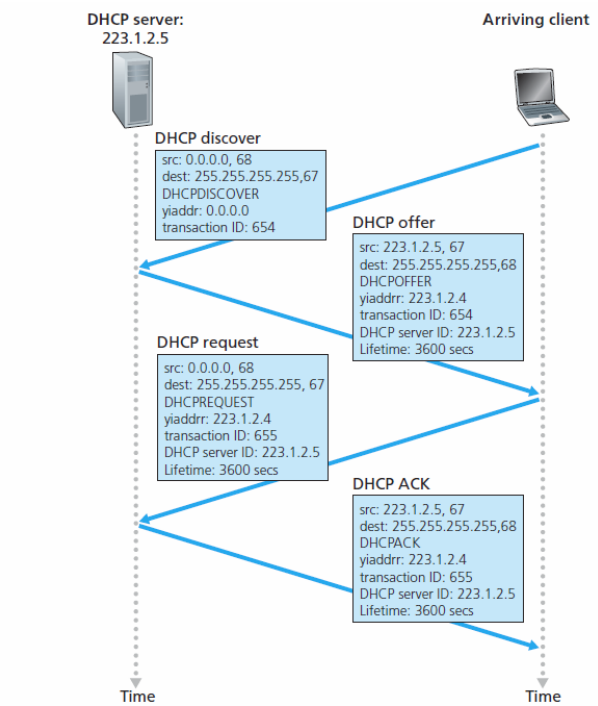


Figure 4.21 ♦ DHCP client-server interaction

Network Address Translation (NAT):

- So we know how an organisation can obtain a block of IP addresses from an ISP and then add different hosts to different subnets in its internet network
- But what about a household with a variable number of devices entering/leaving the network. Not feasible to make every homeowner worry about management of IP addresses.
- **Solution = NAT**
- Operation of a NAT-enabled router (below):

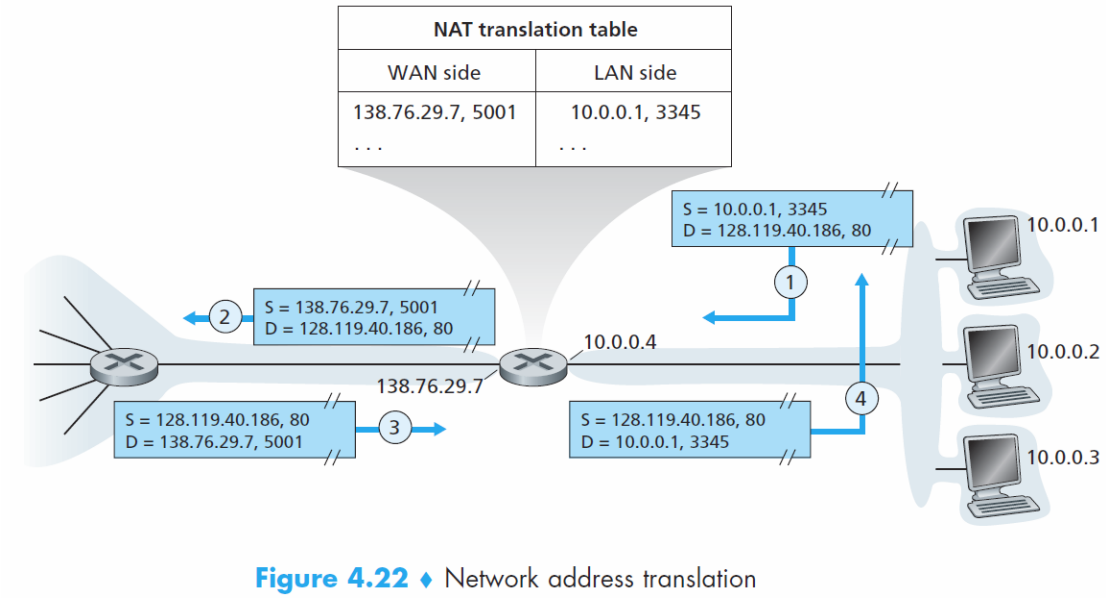


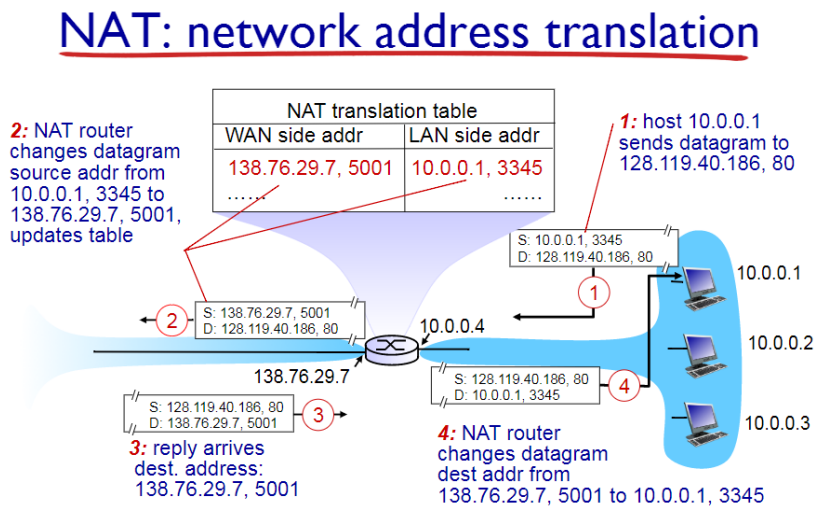
Figure 4.22 ♦ Network address translation

-
- ^ NAT-enabled router has an interface which is connected to the home network
  - Each of the 4 interfaces in the network have a subnet address of 10.0.0.xxx/24
  - 10.0.0.0/8 → (first 8 bits) reserved for private networks (**realms**)
    - In a realm with private addresses, the addresses only have meaning to other devices in the realm
- Devices within home network can talk to each other using 10.0.0.0/24 addressing
  - (Allocated internally using DHCP)
  - But not outside of network! Since every private network uses this address space internally.
- NAT-enabled router doesn't 'look like a router' to outside world
  - → behaves like a single device with 1 IP address
  - Hides the details of the home network
  - NAT gets this single IP address from ISP's DHCP server
- When datagrams are sent to a household's NAT-router's 'external' IP, **NAT translation table** is used to forward the messages to the appropriate devices inside the network, using translation table
    - **NAT translation table** entries have port numbers and IP addresses

NAT problems:

- (1) uses port numbers to address hosts, instead of processes
- (2) forces routers to process packets beyond layer 3
- (3) host-host communication should be direct → no modifying of IPs by other devices

- (4) should just use IPv6 to solve shortage of IP addresses, instead of patching problem with NAT
- (5) interferes with P2P applications
    - If a peer is behind a NAT, it cannot act as a server and accept TCP connections
    - Hacky solution if peer A is not behind a NAT: **connection reversal**
    - If A and B behind NATs, = trickier → must use application relays (like what Skype does)



UPnP: Universal Plug and Play

- Protocol that allows a host to discover and configure a nearby NAT
- Host and NAT must be UPnP-compatible

2012]. UPnP requires that both the host and the NAT be UPnP compatible. With UPnP, an application running in a host can request a NAT mapping between its (*private IP address, private port number*) and the (*public IP address, public port number*) for some requested public port number. If the NAT accepts the request and creates the mapping, then nodes from the outside can initiate TCP connections to (*public IP address, public port number*). Furthermore, UPnP lets the application know the value of (*public IP address, public port number*), so that the application can advertise it to the outside world.

As an example, suppose your host, behind a UPnP-enabled NAT, has private address 10.0.0.1 and is running BitTorrent on port 3345. Also suppose that the public IP address of the NAT is 138.76.29.7. Your BitTorrent application naturally wants to be able to accept connections from other hosts, so that it can trade chunks with them. To this end, the BitTorrent application in your host asks the NAT to create a "hole" that maps (10.0.0.1, 3345) to (138.76.29.7, 5001). (The public port number 5001 is chosen by the application.) The BitTorrent application in your host could also advertise to its tracker that it is available at (138.76.29.7, 5001). In this manner, an external host running BitTorrent can contact the tracker and learn that your BitTorrent application is running at (138.76.29.7, 5001). The external host can send a TCP SYN packet to (138.76.29.7, 5001). When the NAT receives the SYN packet, it will change the destination IP address and port number in the packet to (10.0.0.1, 3345) and forward the packet through the NAT.

In summary, UPnP allows external hosts to initiate communication sessions to NATed hosts, using either TCP or UDP. NATs have long been a nemesis for P2P applications; UPnP, providing an effective and robust NAT traversal solution, may be their savior. Our discussion of NAT and UPnP here has been necessarily brief. For more detailed discussions of NAT see [Huston 2004, Cisco NAT 2012].

4.4.3 Internet Control Message Protocol (ICMP)

- Recall: internet has 3 main components:
    - IP protocol

    - Routing protocols
    - ICMP
- ICMP = used by hosts to communicate network-layer info to each other
- Mainly used for error reporting
    - ie: “destination network unreachable”
- Lies just above IP
- ICMP messages:
    - Type field
    - Code field
    - Contains header and first 8-bytes of datagram that caused the error
- Used in *ping* and *traceroute*

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

- **Figure 4.23 ♦** ICMP message types