CSE 3231
Computer Networks

Chapter 5
The Network Layer
part 1

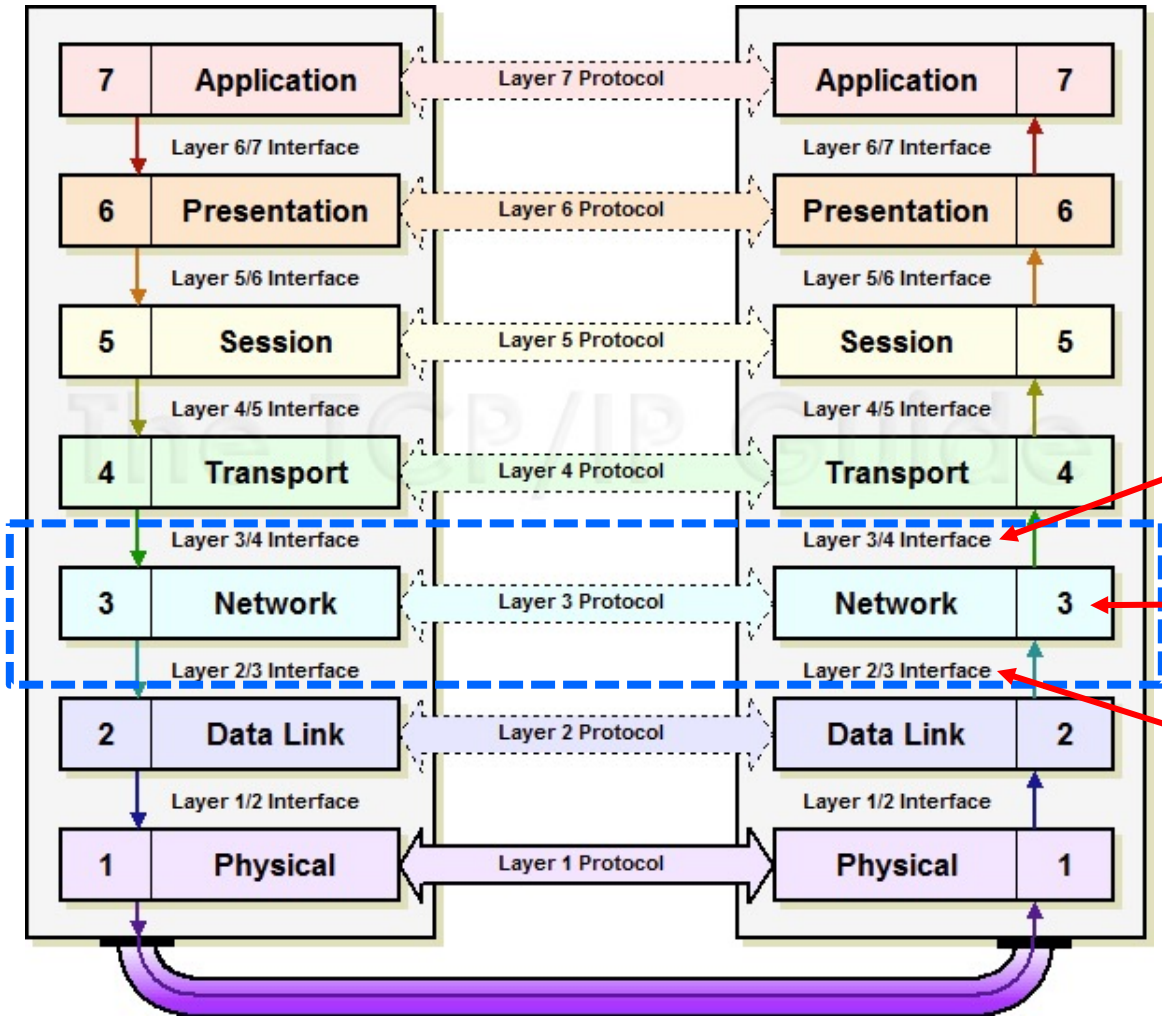
William Allen, PhD

Spring 2022

Networking Beyond the Local Link

- We have seen how the Data Link layer delivers frames from one node to another within the *same* Local Area Network (LAN)
- In this chapter, we look at the way that data is delivered *from one network to another*
  - This will require different protocols and additional header information to support delivery across multiple networks (including the Internet)
- This is the purpose of the *Network Layer*
  - The network layer assembles data into *packets* that are carried as the *payload* in Data Link layer frames

Network Layer



Includes the interface between Network and Transport layers

the Network layer

the interface between the Network and Data Link layers

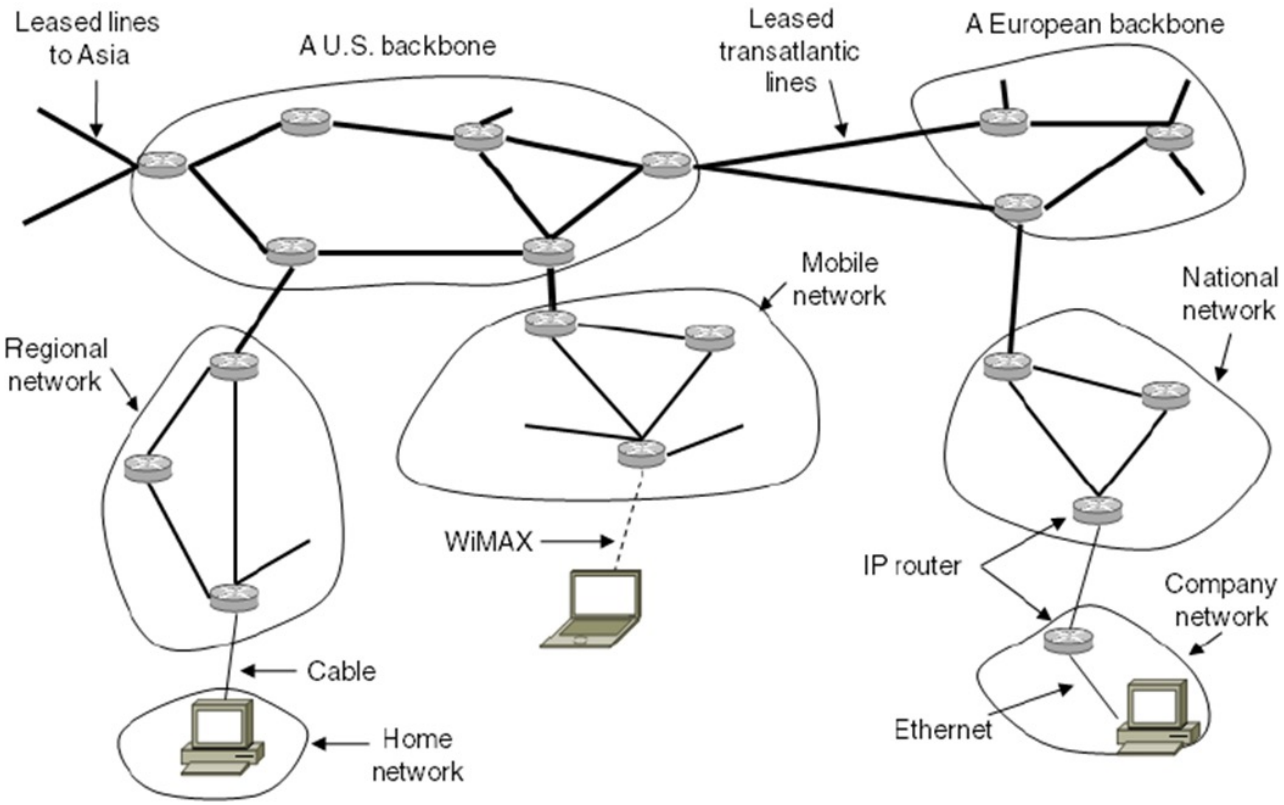
Network Layer in the Internet

The Network Layer's Internet Protocol (IP) was shaped by these guiding principles:

- Make sure it works
- Keep it simple
- Make clear choices
- Exploit modularity
- Expect heterogeneity
- Avoid static options and parameters
- Look for good design (not perfect)
- Strict sending, tolerant receiving
- Think about scalability
- Consider performance and cost

Network Layer in the Internet

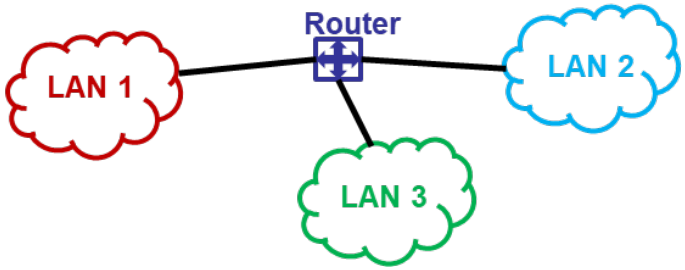
The Internet is an interconnected collection of many networks held together by the IP protocol



Store and Forward

The links between networks are connected through devices called *switches* and *routers*

- Each router or switch has a buffer where it holds the packets until the outbound link is clear for transmission
  - This process is called *store and forward*



- The difference between routers and switches is based on *how they forward* the packets
  - Switches are used to create *virtual circuits* – a pre-determined dedicated path between two nodes
    - these switches are different from the Ethernet switches we saw earlier
  - Routers use a packet's IP address to decide which output link the packet should be sent through to reach its destination

Networking Beyond the Local Link

- How does a switch or a router determine whether the packet's destination is a node in the same network or is a node in another network?
  - In both cases, it uses information in the *network-layer header* of the packet to make that decision
  - The two most common approaches:
    - *Virtual Circuit* or *Connection-oriented* approach creates a path through the network in advance
    - *Datagram* or *Connectionless* approach routes packets dynamically through the network
      - The terms connection-oriented and connectionless are used in several different ways in networking

Switching and Forwarding

Virtual Circuit Switching

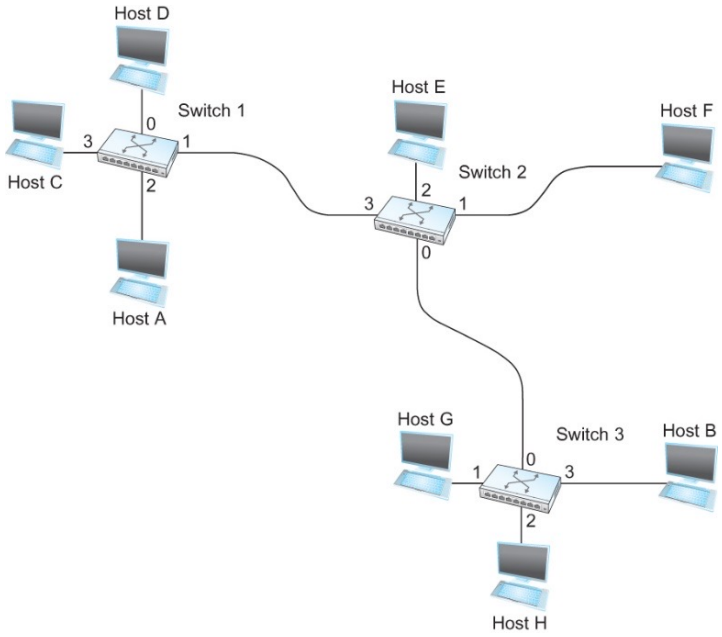
- Creates a *virtual circuit* (VC) from source to destination before transmitting any data packets

Connection-Oriented data delivery:

- Connection setup
  - Establish the virtual circuit through each of the switches between the source and destination hosts
  - The virtual circuit for a single connection consists of an entry in the *Virtual Circuit (VC) forwarding table* in each switch through which the connection passes
- Transfer data until connection is no longer needed
- Remove forwarding table entries for this connection

Connection-Oriented – Virtual Circuits

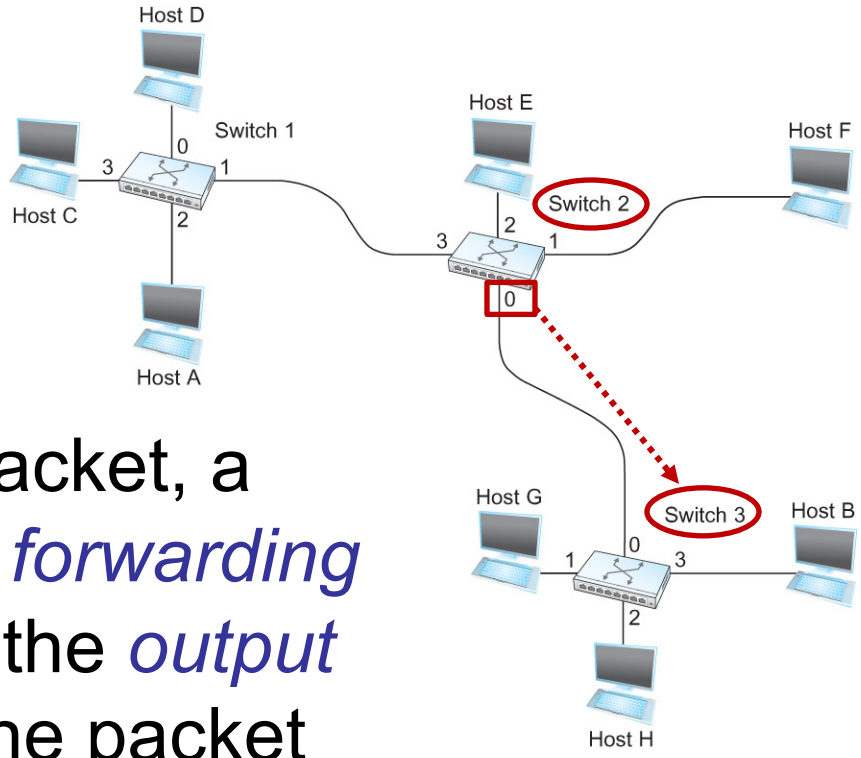
- Packets are forwarded along a virtual circuit using an *identifier* in the packet's header
  - A *Virtual Circuit* (VC) is set up ahead of time, i.e. a *connection* is established before data is transmitted by creating *forwarding tables* in each switch the packet will travel through



  - An example network:
  - Assume we have already calculated a spanning tree for this network

Switching and Forwarding

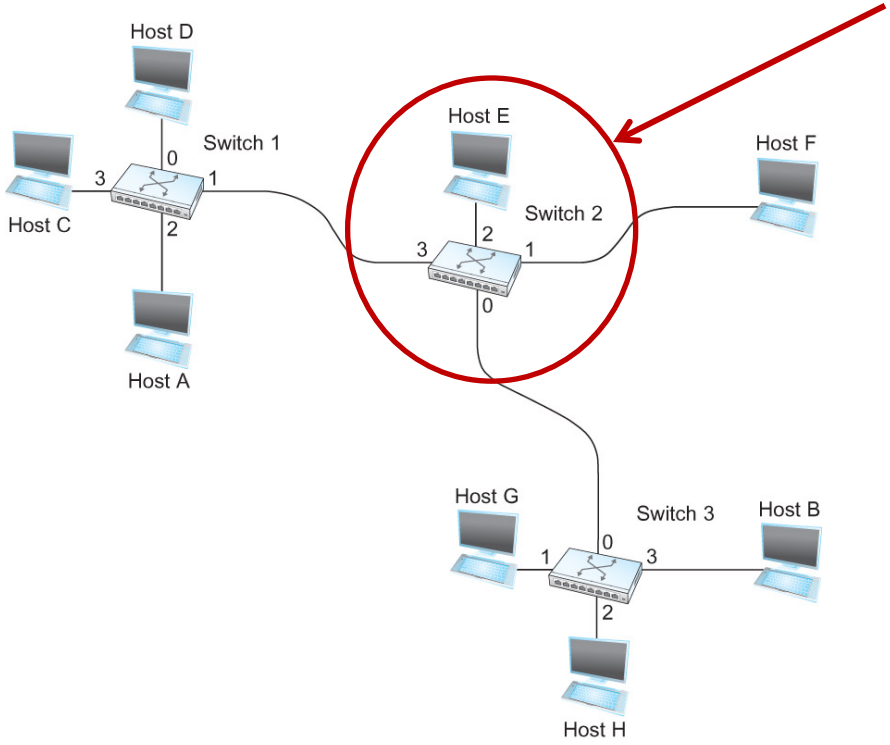
An example network:



- Before sending a packet, a switch consults the *forwarding table* that specifies the *output port* for delivering the packet to the next switch or node
  - The *forwarding table* could be based on the network's spanning tree

For example, Switch 2 uses port 0 to deliver frames to Switch 3

Switching and Forwarding

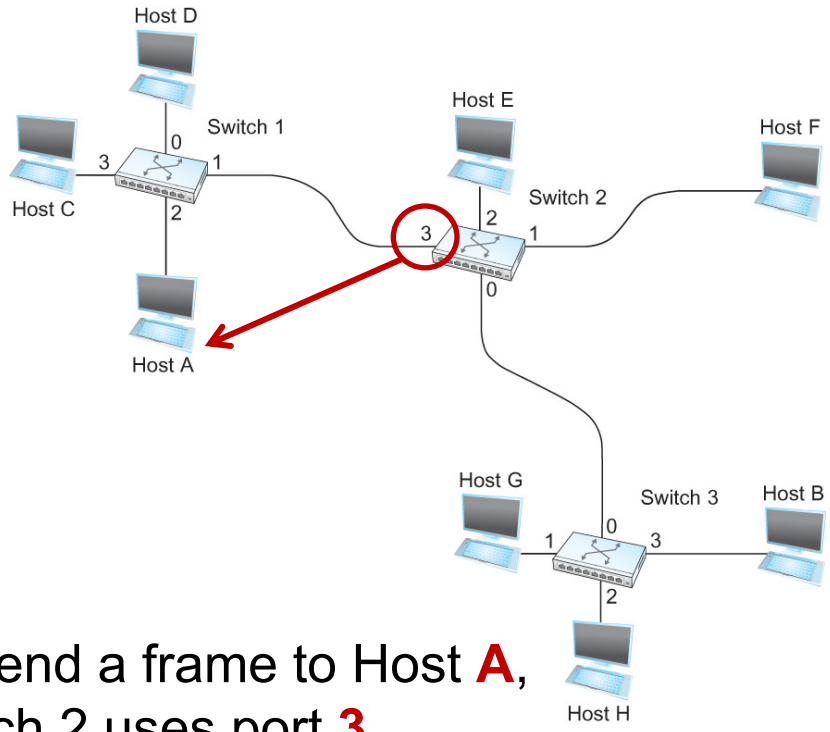


Destination	Port
A	?
B	?
C	?
D	?
E	?
F	?
G	?
H	?

Forwarding Table for Switch 2

Each switch has its own Forwarding Table

Switching and Forwarding



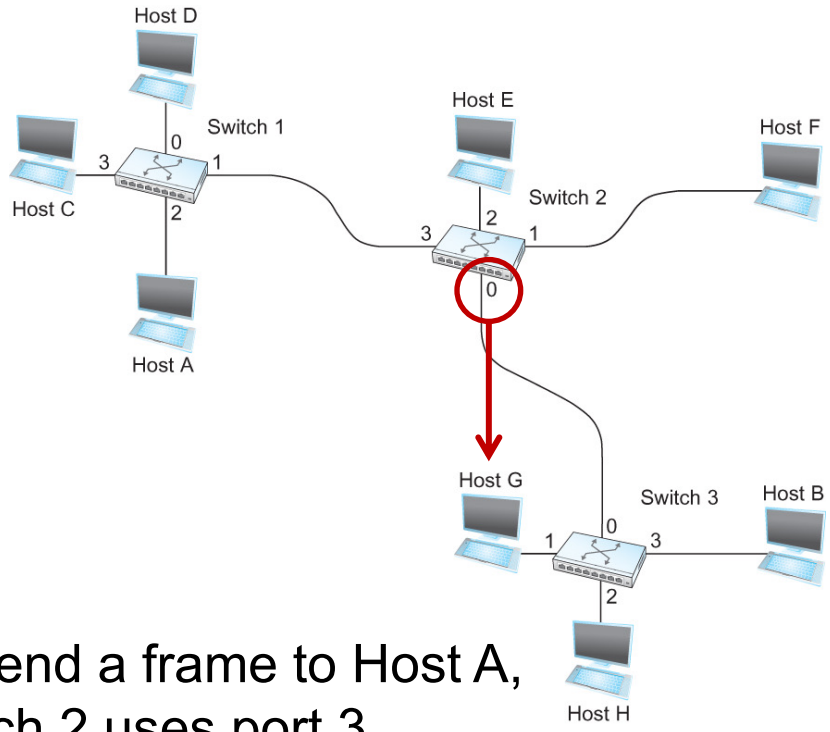
- To send a frame to Host **A**, switch 2 uses port **3**

Destination	Port
A	**3**
B	?
C	?
D	?
E	?
F	?
G	?
H	?

Forwarding Table for Switch 2

Each switch has its own Forwarding Table

Switching and Forwarding



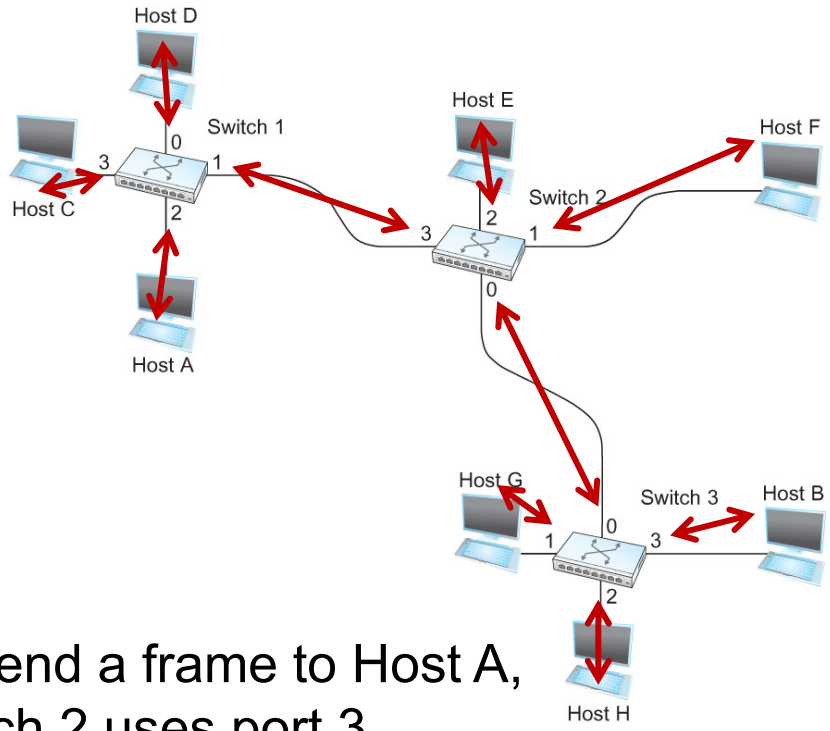
Destination	Port
A	3
B	?
C	?
D	?
E	?
F	?
G	**0**
H	?

Forwarding Table for Switch 2

- To send a frame to Host A, switch 2 uses port 3
- To send a frame to Host **G**, switch 2 uses port **0**

Each switch has its own Forwarding Table

Switching and Forwarding



Destination	Port
A	3
B	0
C	3
D	3
E	2
F	1
G	0
H	0

Forwarding Table for Switch 2

- To send a frame to Host A, switch 2 uses port 3
- To send a frame to Host G, switch 2 uses port 0

We can easily fill in the rest of the ports

Each switch has its own Forwarding Table

Switching and Forwarding

One entry in the VC table on a single switch contains:

- A *virtual circuit identifier* (VCI) that uniquely identifies the connection at this switch and will be carried inside the header of the packets that belong to this connection
- An *incoming interface* (port) where packets for this VC arrive at the switch
- An *outgoing interface* (port) where packets for this VC leave the switch
- A potentially different VCI that will be used for the packets going out over the next virtual circuit
  - chosen to avoid collisions with other virtual circuits

Switching and Forwarding

- If a packet arrives on the designated incoming interface and that packet contains the designated VCI value in its header, then the packet should be sent out the specified outgoing interface with the specified outgoing VCI value first having been placed in its header
- The combination of the VCI of the packets as they are received at the switch and the interface on which they are received uniquely identifies the virtual connection
  - There may be many virtual connections established in the same switch at one time, each with a different VCI

Switching and Forwarding

- Incoming and outgoing VCI values are not generally the same
  - VCI is not a globally significant identifier for the connection; it only has to be unique on a given link
- Whenever a new connection is created, we need to assign a unique VCI for that connection on each link that the connection will traverse
  - We also need to ensure that the chosen VCI on a given link is not currently in use on that link by some existing connection. But they only have to be unique for that link, they can be reused over another link.

Switching and Forwarding

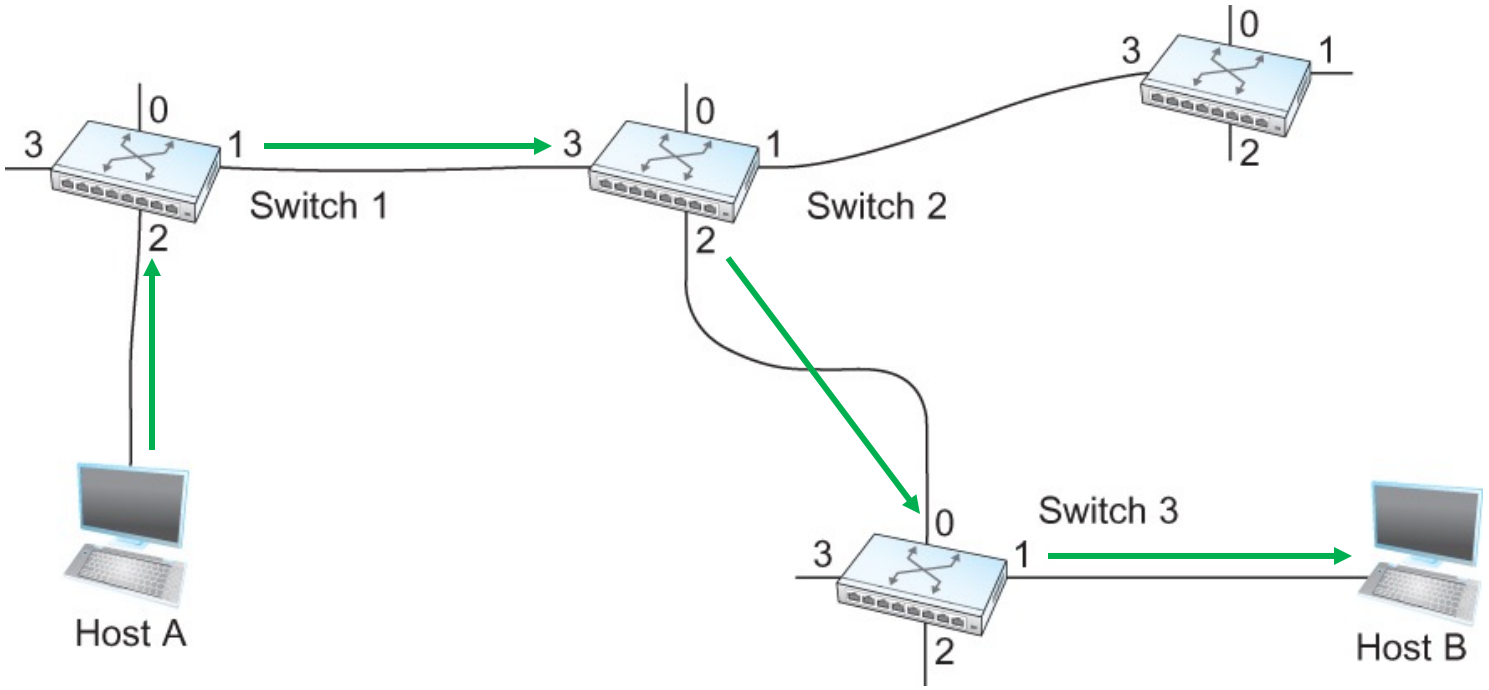
Two broad classes of approach to establishing connection state

- Network Administrator could configure the tables
  - The virtual circuit is a Permanent Virtual Circuit (PVC)
  - The network administrator can update this later
  - Can be thought of as a long-lived or administratively configured VC
- A host can send messages through the network to create the entries in the tables remotely
  - This is referred as *signalling* and the resulting virtual circuit is said to be a Switched Virtual Circuit (SVC)
  - A host may set up and delete such an SVC dynamically without the involvement of a network administrator

Switching and Forwarding

Assume that a network administrator wants to manually create a new virtual connection from host A to host B

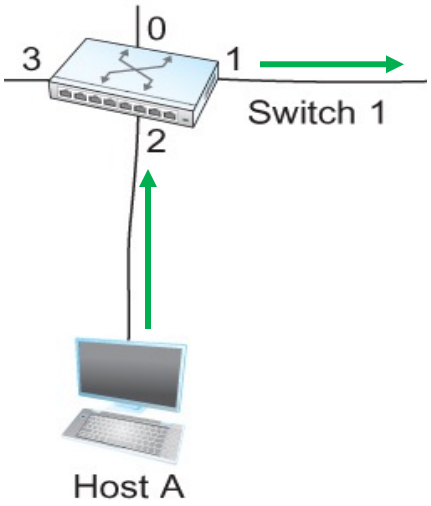
- First the administrator identifies a path through the network from A to B



Switching and Forwarding

The administrator then picks a VCI value that is currently unused on each link for the connection

- For our example,
  - Suppose the VCI value **5** is chosen for the link from host A to switch 1 and VCI value **11** for the link from switch 1 to switch 2
  - Determine the incoming interface for the link from host A
  - Determine the outgoing interface for switch 1



Any packet arriving on interface 2 with VCI=5 should exit interface 1 with VCI=11

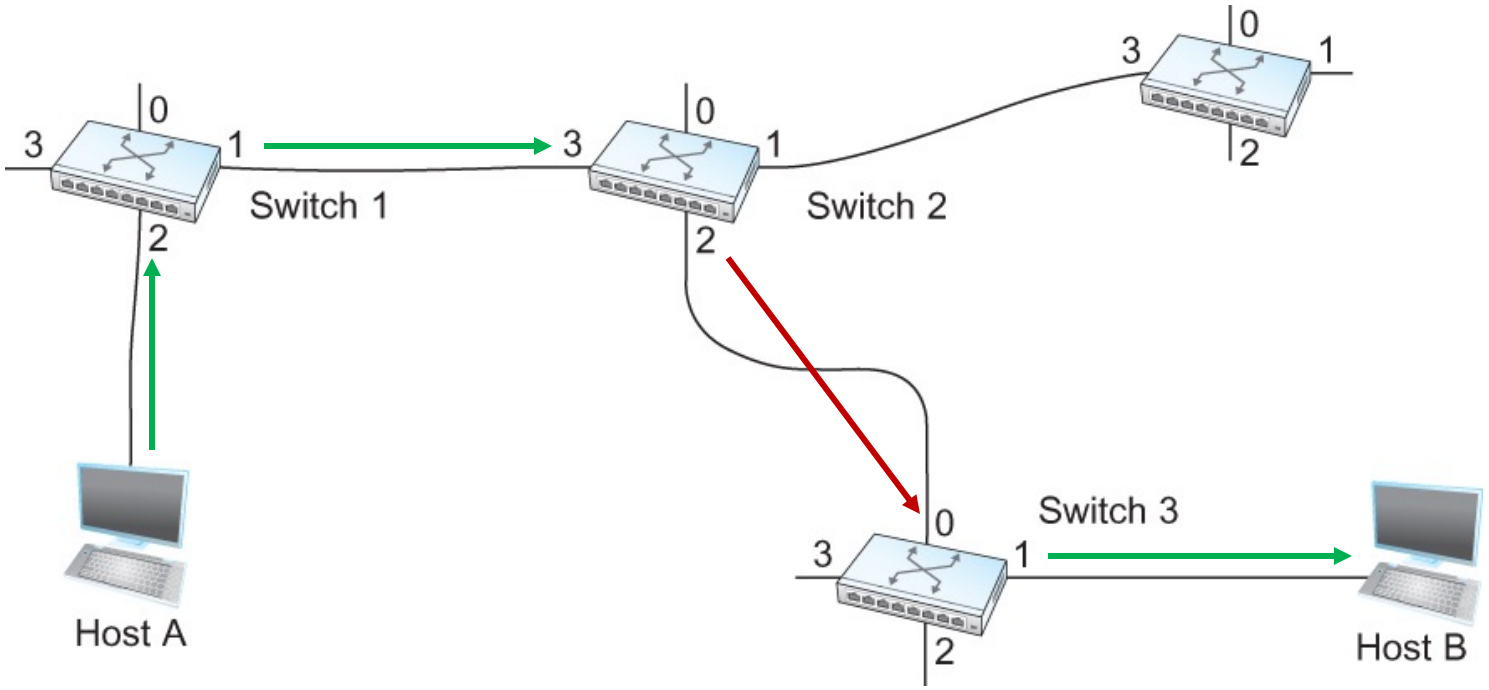
Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
2	**5**	**1**	**11**

Table for Switch 1

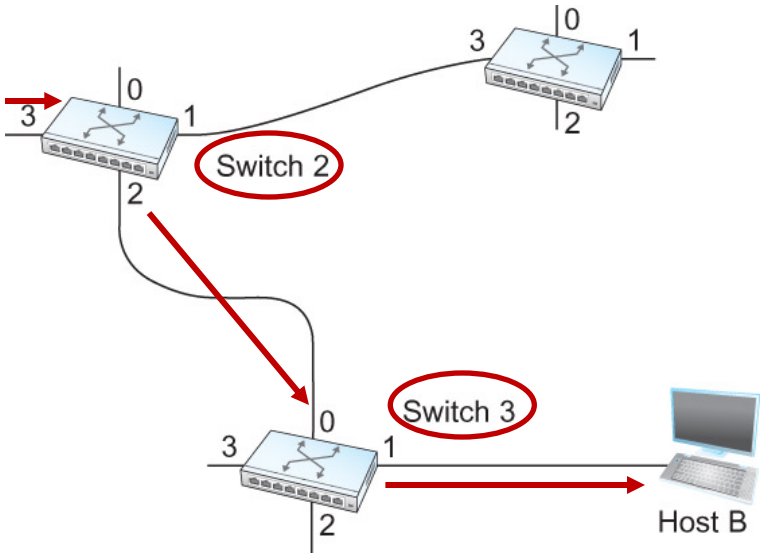
Switching and Forwarding

Continuing to manually create a new virtual connection from host A to host B

- We looked at the link from Host A to Switch 1, now let's look at the link from Switch 2 to Switch 3



Switching and Forwarding



Continuing:

- VCI of **7** is chosen to identify this connection on the link from Switch 2 to Switch 3
- VCI of **4** is chosen for the link from Switch 3 to host B

Incoming & Outgoing Interfaces on Switch 2 & Switch 3

Switch 2

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
3	11	2	7

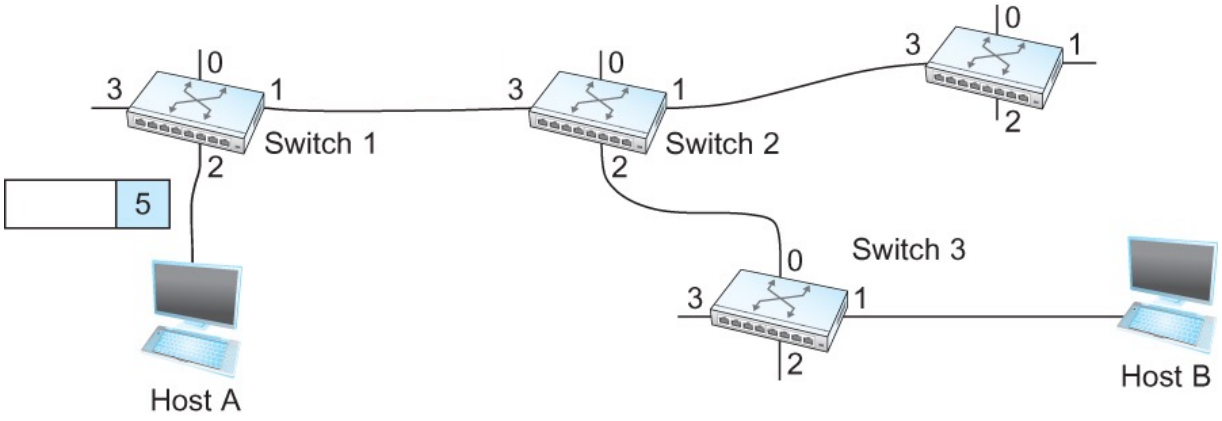
Switch 3

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
0	7	1	4

Switching and Forwarding

- For any packet that A wants to send to B, A first puts the VCI value 5 in the header of the packet and sends it to Switch 1
- Switch 1 receives any such packet on interface 2, and it uses the combination of the interface and the VCI in the packet header to find the appropriate VC table entry.

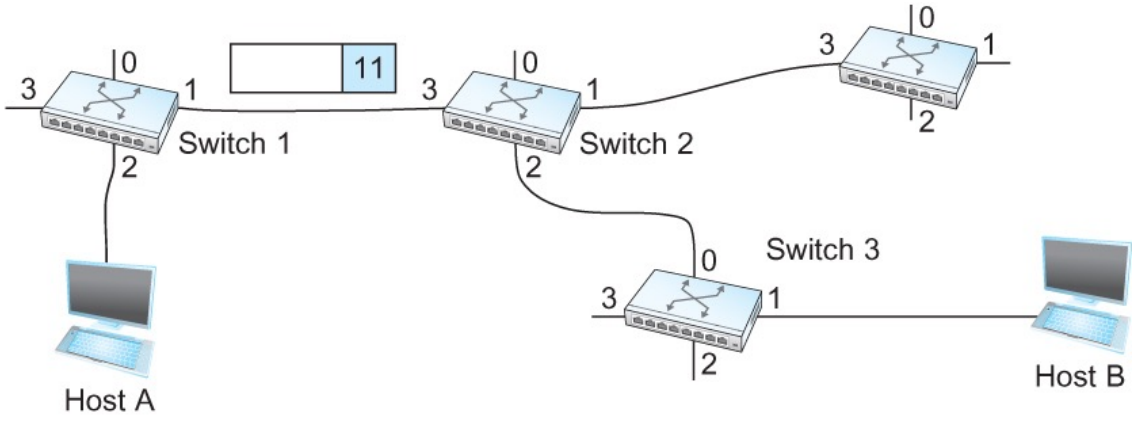
Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
2	5	1	11



Switching and Forwarding

- For any packet that A wants to send to B, A first puts the VCI value 5 in the header of the packet and sends it to Switch 1
- Switch 1 receives any such packet on interface 2, and it uses the combination of the interface and the VCI in the packet header to find the appropriate VC table entry.
- The table entry on Switch 1 tells the switch to forward the packet out of interface 1 and to put the VCI value 11 in the header

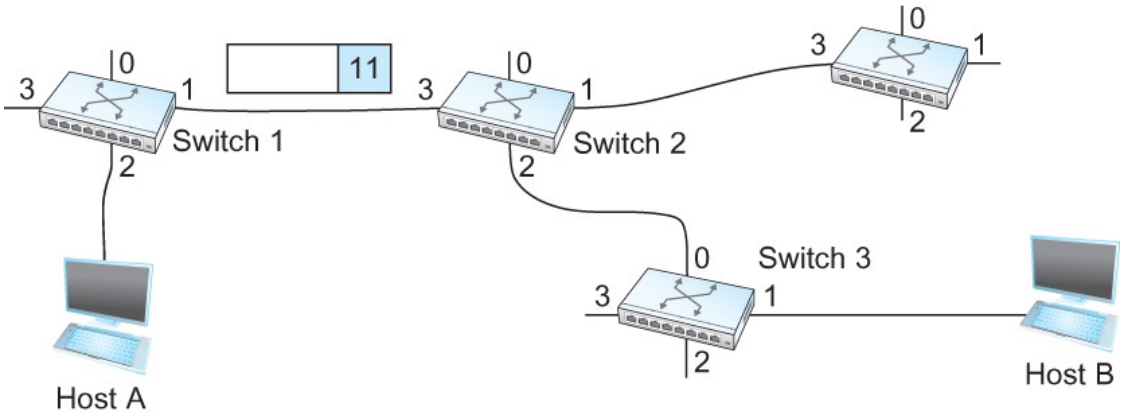
Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
2	5	1	11



Switching and Forwarding

- Packet will arrive at Switch 2 on interface 3 bearing VCI 11
- Switch 2 looks up interface 3 and VCI 11 in its VC table and sends the packet on to Switch 3 after updating the VCI value appropriately

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
3	11	2	7
Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
0	7	1	4

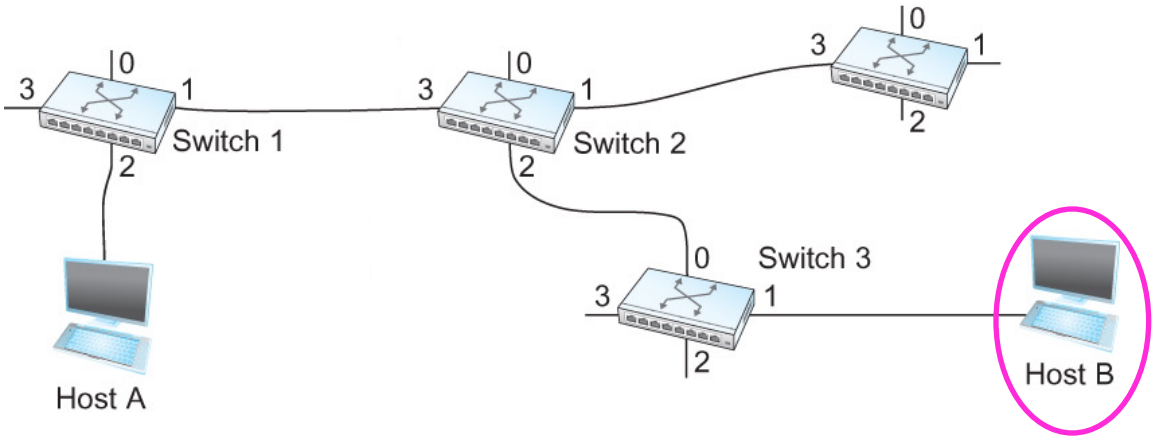


Switching and Forwarding

- Packet will arrive at Switch 2 on interface 3 bearing VCI 11
- Switch 2 looks up interface 3 and VCI 11 in its VC table and sends the packet on to Switch 3 after updating the VCI value appropriately
- This process continues until it arrives at host B with the VCI value of 4 in the packet
- To host B, this identifies the packet as having come from host A

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
3	11	2	7

Incoming Interface	Incoming VCI	Outgoing Interface	Outgoing VCI
0	7	1	4



IP Service Model

- Common packet delivery model in the Internet
  - *Connectionless* model for data delivery
  - *Best-effort* delivery (unreliable service)
    - packets can be lost
    - packets can be delivered out of order
    - duplicate copies of a packet can be delivered
    - packets can be delayed for a long time
- Uses a *global addressing scheme*
  - Provides a way to identify all hosts in the network

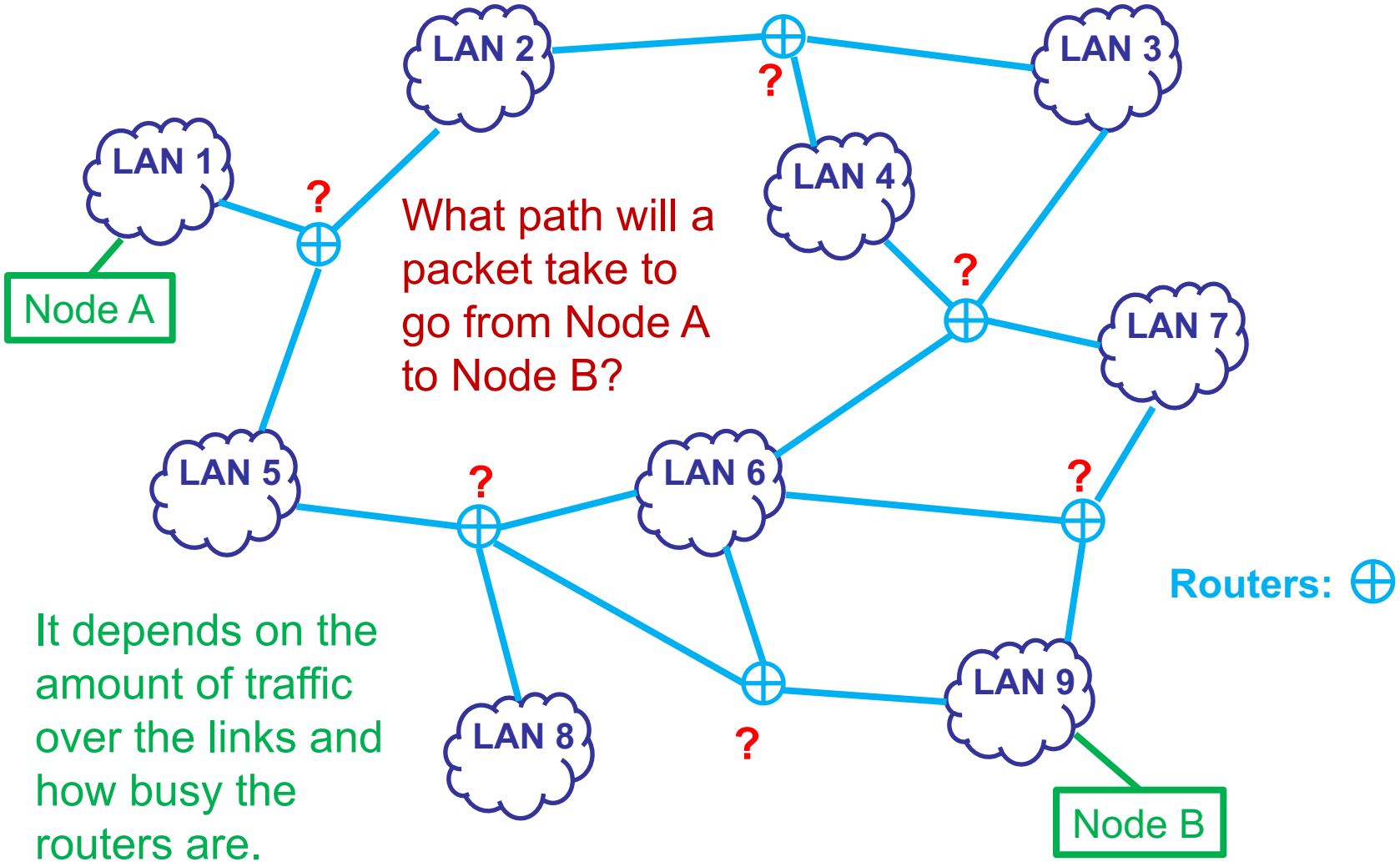
Connectionless Delivery

- Packet headers contain information that routers use to forward the packet to their destination
  - address of the source node and the destination node
- Routers “know” which output port will send the packet *towards* the destination, but they may not “know” the entire path to get it there
  - The packet will move from router to router until it arrives at the destination
- The path it will take was *not pre-determined*
  - thus, this is referred to as a *connectionless* delivery and packets sent this way are called *datagrams*

Connectionless (Datagram) Networks

- Hosts can transmit packets without delay since no connection is needed before sending data
- However, when a host sends a packet, it has no way of knowing if the network can deliver it, how long it will take or if the destination host is even up and running
  - no guarantee of delivery or order of arrival or time of arrival
- Each packet is forwarded independently of previous packets that have been sent to the same destination.
  - Thus two successive packets from host A to host B may follow completely different paths
- A switch or link failure might not result in a lost packet because it may be possible to find an alternate route around the failure - routers will be updated frequently

Connectionless (Datagram) Networks



Comparison of Virtual-Circuits & Datagrams

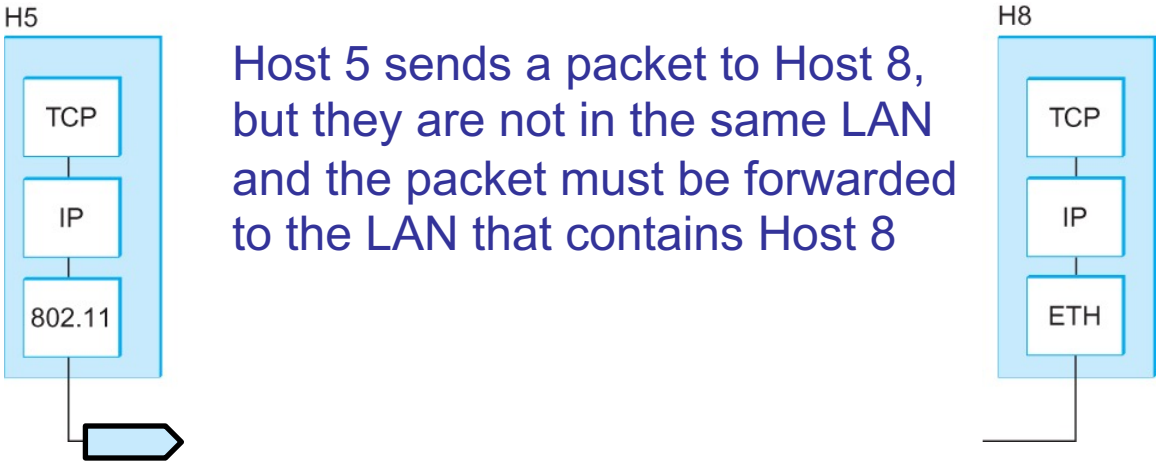
VC = *Virtual Circuit*

Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Internetworking

Internet Protocol (IP)

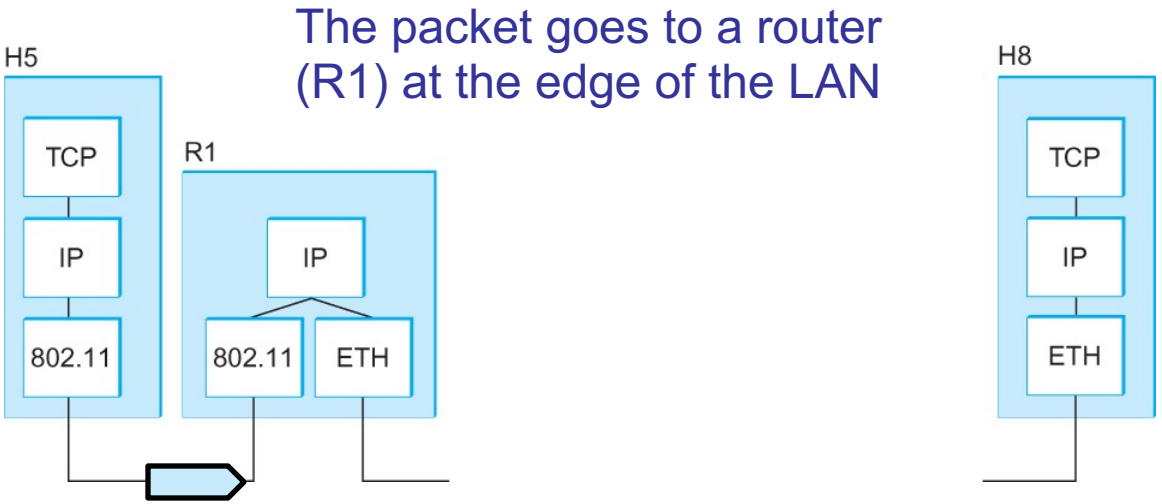
- Supports scalable, heterogeneous internetworks
  - *internetwork*: group of networks that are all connected
- It runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork



Internetworking

Internet Protocol (IP)

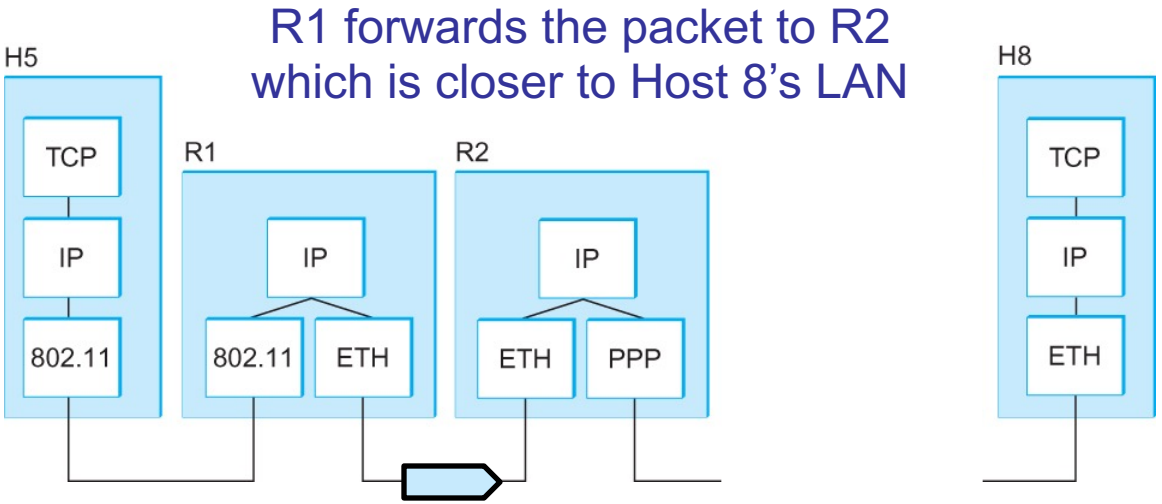
- Supports scalable, heterogeneous internetworks
  - *internetwork*: group of networks that are all connected
- It runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork



Internetworking

Internet Protocol (IP)

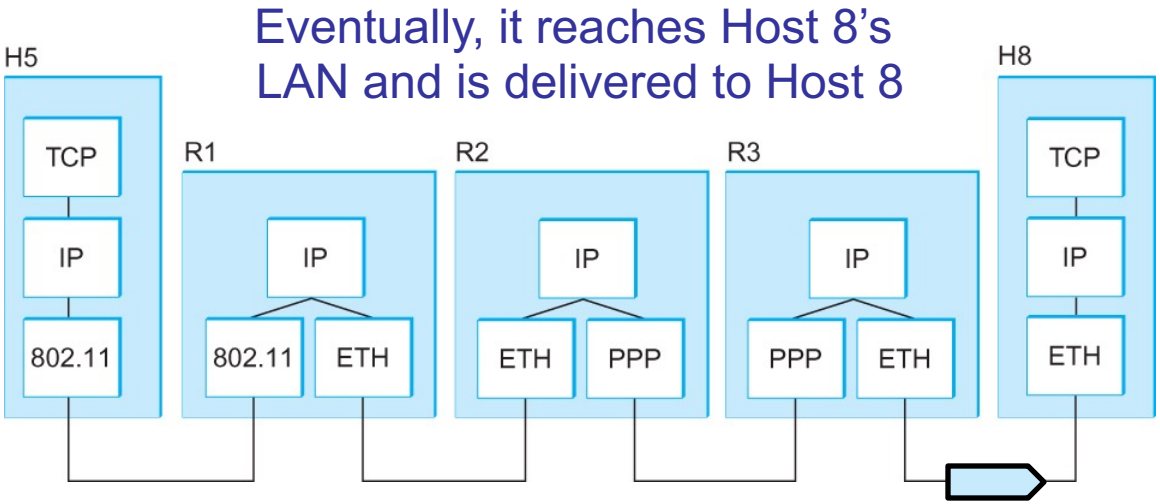
- Supports scalable, heterogeneous internetworks
  - *internetwork*: group of networks that are all connected
- It runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork



Internetworking

Internet Protocol (IP)

- Supports scalable, heterogeneous internetworks
  - *internetwork*: group of networks that are all connected
- It runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork

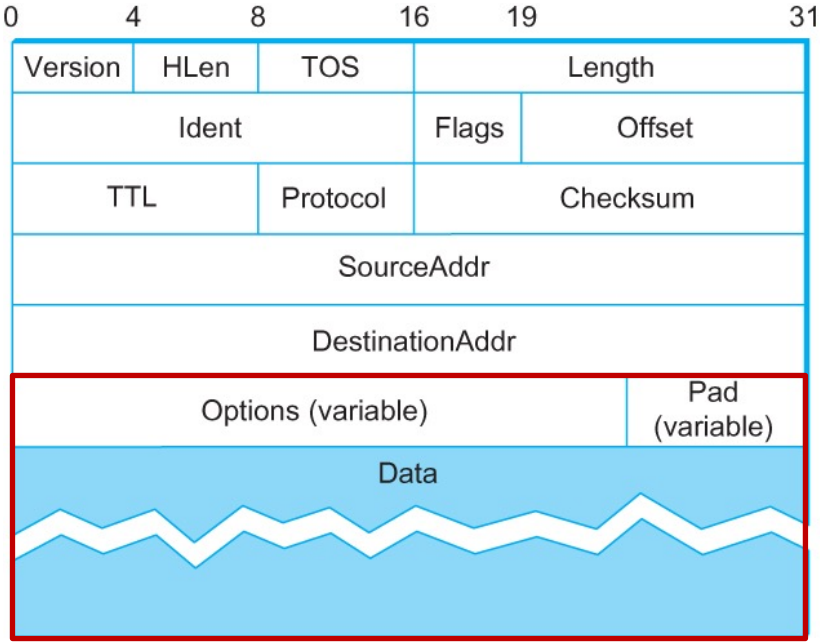


Internet Protocol Versions

- There are currently two active versions of the Internet Protocol (IP)
  - Version 4 (IPv4) has been in use since 1981 (RFC 791) and has been updated many times
  - Version 6 (IPv6) was released in 2006 to address limitations in IPv4, but it still hasn't been fully adopted across the Internet
    - the most recent version of IPv6 is in RFC 8200
- We will start with IPv4 (which is deployed everywhere in the Internet) and later take a brief look at IPv6
  - v1, v2, v3 and v5 were experimental

IP Packet Format

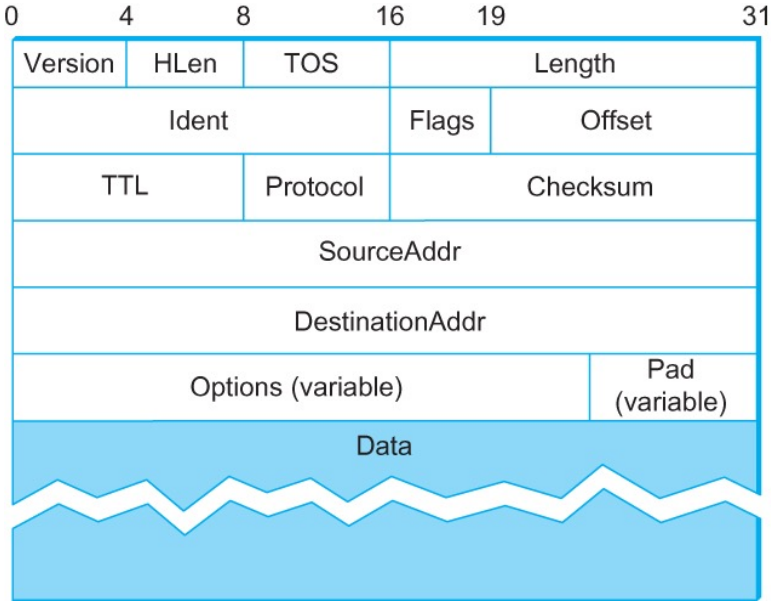
- Version (4 bits): most often v4
- Hlen (4 bits): number of 32-bit words in the header
- TOS (8 bits): type of service (not widely used)
- Length (16 bits): number of bytes in this datagram
- Ident (16 bits) and Flags/Offset (16 bits): supports fragmentation
- TTL (8 bits): number of hops this datagram has traveled
- Protocol (8 bits): demultiplex key (TCP=6, UDP=17)
- Checksum (16 bits): (header only)
- SrcAddr, DestAddr (32 bits each)



- Options/Pad (variable length)
- Data (variable length)

IP Header Fields

- *Version*: all IPv4 packets have the value 0x4 (`0100`)
- *HLen*: this field shows the number of *4-byte words* that the header uses. The default value is 0x5 (`0101`) because the default header size is 20 bytes (0x5 * 4 bytes)
- *TOS*: Type of Service, allows the sender to specify priority for this packet (not used often)
- *Length*: the length of the entire packet (min: 20, max: 65,535)



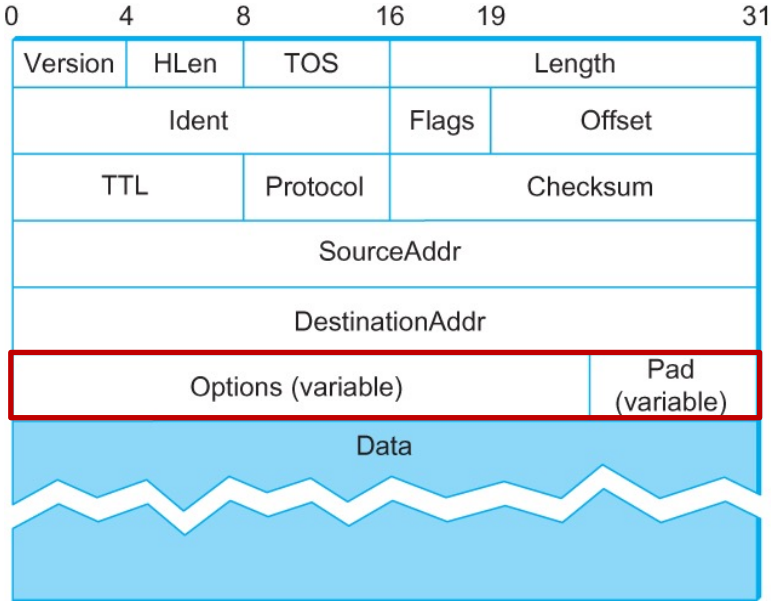
Note: there is no trailer field, the packet ends when the last byte of data is reached

IP Header Fields

- Ident (16 bits)
- Flags (3 bits)
- Offset (13 bits)

together, these fields support fragmentation

- TTL (8 bits): number of hops this datagram can pass thru (prevents infinite loops)
- Protocol (8 bits): next layer (TCP=6, UDP=17, etc.)
- Checksum (16 bits): (covers the header only)
- Source Addr (32 bits)
- Destination Addr (32 bits)
- Optional parameters
  - depends on the transport protocol



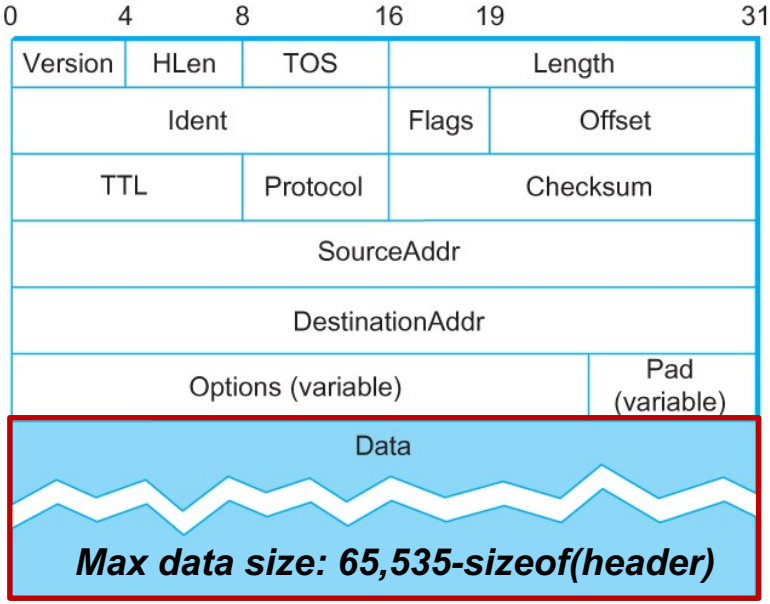
Each time a packet is processed by a router, the TTL value is decremented, when TTL == 0, the packet is discarded

IP Header Fields

- Ident (16 bits)
- Flags (3 bits)
- Offset (13 bits)

together, these fields support fragmentation

- TTL (8 bits): number of hops this datagram can pass thru (prevents infinite loops)
- Protocol (8 bits): next layer (TCP=6, UDP=17, etc.)
- Checksum (16 bits): (covers the header only)
- Source Addr (32 bits)
- Destination Addr (32 bits)
- Optional parameters
  - depends on the transport protocol



Each time a packet is processed by a router, the TTL value is decremented, when TTL == 0, the packet is discarded

Dotted Decimal Notation

- IP addresses are written in what is often called ***dotted decimal* notation**
  - Each byte is represented by a decimal number in the range [0..255]: