

COMPUTER NETWORKS

9 August 2023

Dr Noor Mohammad Sk

Computer Networks

2

- Heterogeneous systems need to talk to each other:
 - Media to connect
 - Wired – twisted pair, coaxial cable, fibre
 - Wireless – radio
 - Topology of the Network
 - Protocols and Software

Network Requirements

3

- Connectivity
- Cost-Effective Resource Sharing
- Support for Common Services

4

Network Requirement

Connectivity

Connectivity

5

- A network must provide connectivity among a set of computers
- Sometimes it is enough to build a limited network that connects only a few select machines
 - For the reasons of privacy and security
 - Many corporate networks
- Networks are designed to grow (scale) in a way that allows them the potential to connect all the computers in the world
 - Internet

Connectivity – Links, Nodes

6

- At the lowest level
- A network can consist of two or more computers directly connected by some physical medium
 - ▣ A coaxial cable or an optical fiber
- We call such a physical medium a link
- Node: is a specialized piece of hardware rather than a computer
 - ▣ We often refer computer as a node

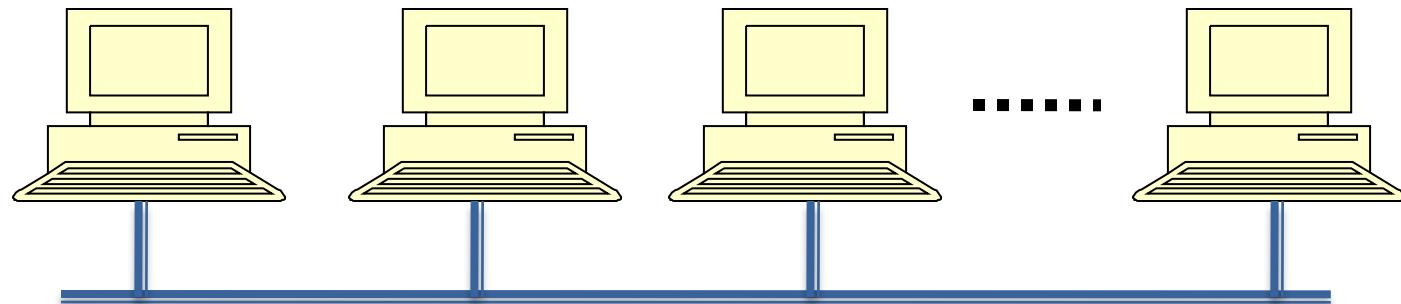
Connectivity – Direct Links

7

- *Point-to-point*: physical links are sometimes limited to a pair of nodes



- *Multiple Access*: more than two nodes may share a single physical link



Limitations of Direct Link

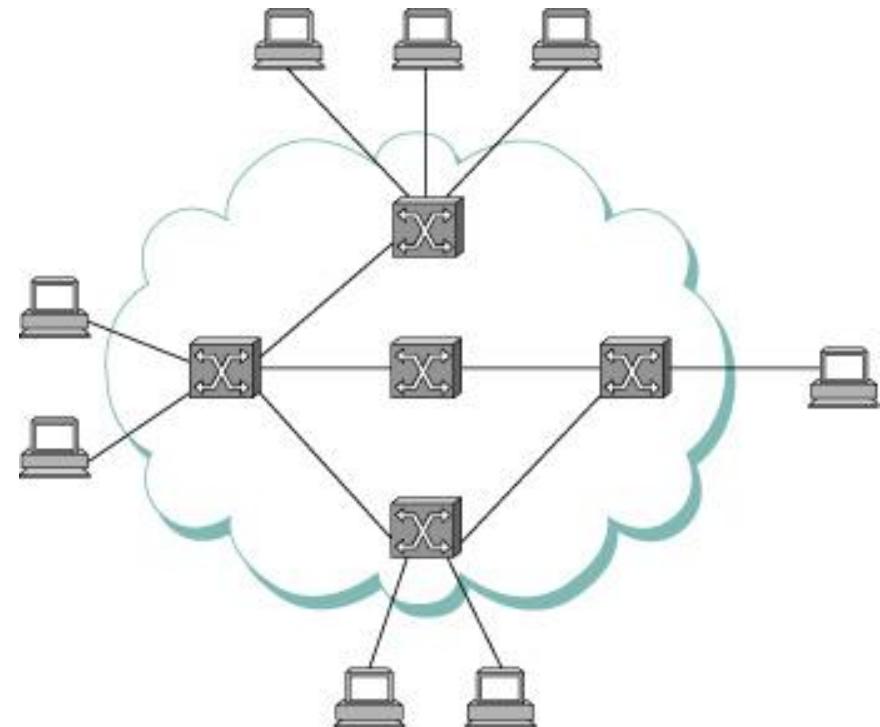
8

- If computer networks were limited to situations in which all nodes are directly connected to each other over a common physical medium
- Then networks would either be very limited in the number of computers they could connect
- Or the number of wires coming out of the back of each node would quickly become both unmanageable and very expensive
- Fortunately, connectivity between two nodes does not necessarily imply a direct physical connection between them
- Indirect connection may be achieved among a set of cooperating nodes

Connectivity – Indirectly connected links

9

- Figure shows a set of nodes, each of which is attached to one or more point-to-point links
- Those nodes that are attached to at least two links run software that forwards data received on one link out of another
- If organized in a systematic way, these forwarding nodes form a switched network



Switched Networks

10

- Two most common types are:
 - Circuit switched
 - Employed by the Telephone Systems
 - Packet switched
 - Computer networks
- Packet Switched Network
 - The nodes in such a network send discrete blocks of data to each other
 - These blocks of data corresponding to some piece application data such as a file, a piece of email, or an image
 - We call each block of data either a packet or a message

Packet Switched Networks

11

- Typically use a strategy called *store-and-forward*
- Each node in a store-and-forward network first receives a complete packet over some link
- Stores the packet in its internal memory, and then forwards the complete packet to the next node
- In contrast, a circuit switched network first establishes a dedicated circuit across a sequence of links
 - and then allows the source node to send a stream of bits across this circuit to a destination node
- The major reason for using packet switching rather than circuit switching in a computer network is efficiency

Connectivity

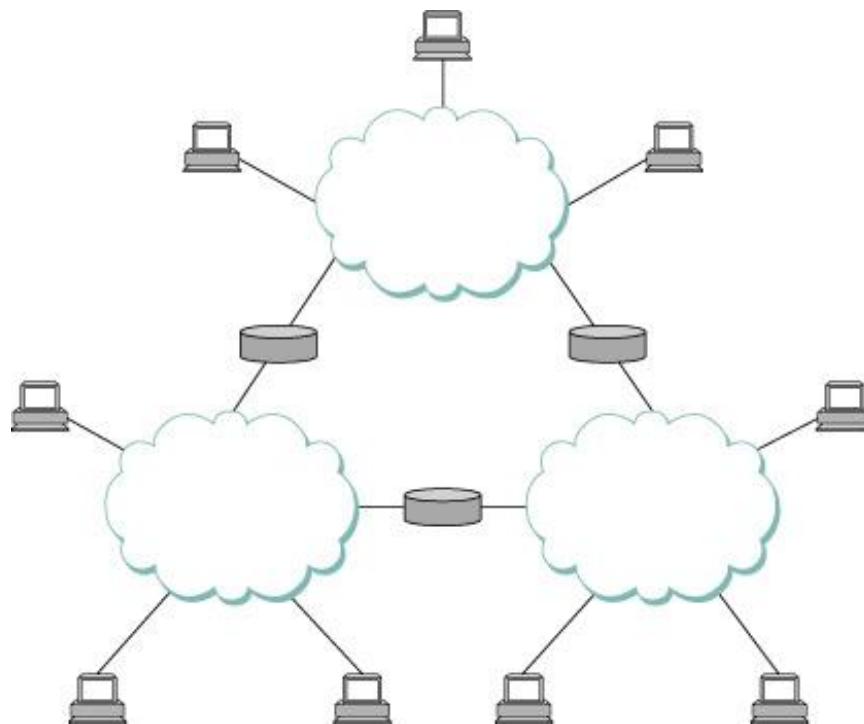
12

- Cloud is one of the important icons of computer networking
 - ▣ Commonly called switches, and their primary function is to store and forward packets
- In general we use a cloud to denote any type of network, whether it is a
 - ▣ Single point-to-point link
 - ▣ Multiple-access link
 - ▣ Or a switched network

Interconnection of Networks

13

- A set of independent networks(clouds) are interconnected to form an internetwork, or internet



Interconnection of Networks

14

- A set of computers can be indirectly connected
- A node that is connected to two or more networks is commonly called a router or gateway
 - ▣ It plays same role as a switch
 - ▣ It forwards messages from one network to another
- Internet itself can be viewed as another kind of networks
 - ▣ An internet built from an interconnection of internets
- Thus one can recursively build arbitrary large networks by interconnecting clouds to form larger cloud

Connectivity - Address

15

- A set of hosts are directly or indirectly connected to each other does not mean that we have succeeded in providing host-to-host connectivity
- The final requirement is that each node must be able to state which of the other nodes on the network it wants to communicate with
- This is done by assigning an *address* to each node
- An address is a byte string that identifies a node
- The network can use a node's address to distinguish it from the other nodes connected to the network

Connectivity – routing

16

- When a source node wants the network to deliver a message to a certain destination node
 - ▣ It specifies the address of the destination node
- If the sending and receiving nodes are not directly connected
 - ▣ Then the switches and routers of the network use this address to decide how to forward the message toward the destination
- The process of determining systematically how to forward messages toward the destination node based on its address is called routing

Unicast, Broadcast and Multicast

17

- Unicast:
 - The source node wants to send a message to a single destination node
- Broadcast:
 - The source node might want to broadcast a message to all the nodes on the network
- Multicast:
 - A source node might want to send a message to some other subset of the other nodes
 - But not all of them

Network Requirement

Cost-Effective Resource Sharing

How hosts share a Network

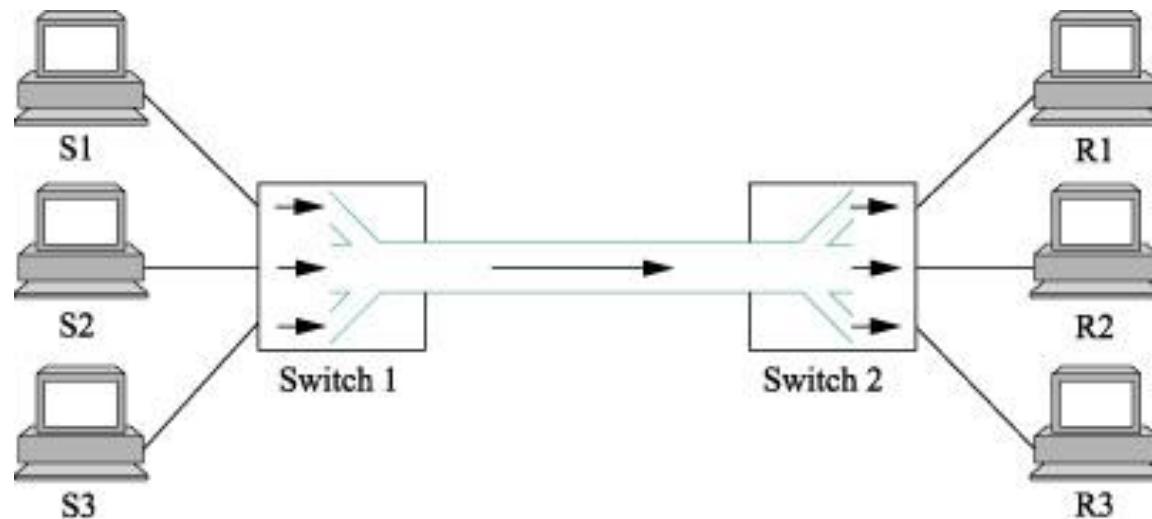
19

- Any one pair of hosts can exchange the messages across a sequence of links and nodes
- Will network support more than one pair of communication between the hosts
- Multiplexing:
 - A system resource is shared among multiple users
 - Analogy to a timesharing computer system
 - Data being sent by multiple users can be multiplexed over the physical links that make up a network

Multiplexing

20

- Three hosts on the left side of the network (senders S1 – S3) are sending data to the three hosts on the right (receivers R1 – R3) by sharing switched networks that contains only one physical link



Multiplexing

21

- Three flows of data – corresponding to the three pairs of hosts – are multiplexed onto a single physical link by switch1
- The demultiplexed back into separate flows by switch 2
- Multiplexing Methods
 - Synchronous time division multiplexing (STDM)
 - Frequency division multiplexing (FDM)

STDM

22

- Idea is to divide time into equal-sized quanta and, in a round-robin fashion, give each flow a chance to send its data over the physical link
- In other words, during
 - Time quantum 1, data from S1 to R1 is transmitted
 - Time quantum 2, data from S2 to R2 is transmitted
 - In time quantum 3, data from S3 to R3 is transmitted
 - In the next time quantum, the first flow (S1 to R1) gets go again,
 - and the process repeats

FDM

23

- The idea is to transmit each flow over the physical link at different frequency
- Much the same way that the signals for different TV stations are transmitted at a different frequency on a physical cable TV link

Limitations of STDM and FDM

24

- If one of the flows (host pairs) does not have any data to send,
 - ▣ Its share of the physical link
 - ▣ That is, its time quantum or its frequency – remain idle
 - ▣ Even if one of the other flows has data to transmit
- For computer communication, the amount of time that a link is idle can be very large
 - ▣ Example: consider the time you spend reading a web page (leave the link idle) compared to the time you spend fetching the page

Limitations of STDM and FDM

25

- The maximum number of flows is fixed and known ahead of time
 - It is not practical to resize the quantum or to add additional quanta in the case of STDM
 - or to add new frequencies in the case of FDM

Statistical Multiplexing

26

- It is like STDM in that the physical link is shared over time
 - ▣ First data from one flow is transmitted over the physical link, then data from another flow is transmitted, and so on.
- Data is transmitted from each flow on demand rather than during a predetermined time slot
 - ▣ If only one flow has data to send, it gets to transmit that data without waiting for its quantum to come around
 - ▣ Thus without having to watch the quanta assigned to the other flows go by unused
- It is avoidance of idle time that gives packet switching its efficiency

Limitations of Statistical Multiplexing

27

- No mechanism to ensure that all the flows eventually get their turn to transmit over the physical link
- Once flow begins sending a data, we need some way to limit the transmission, so that the other flows can have a turn
- An upper bound on the size of the block of data that each flow is permitted to transmit at a given time
- This limited size block of data is typically referred to as a packet

packets

28

- Packet switched network limits the maximum size of packets
- A host may not be able to send a complete message in one packet
- The source may need to fragment the message into several packets, with the receiver reassembling the packets back into the original message

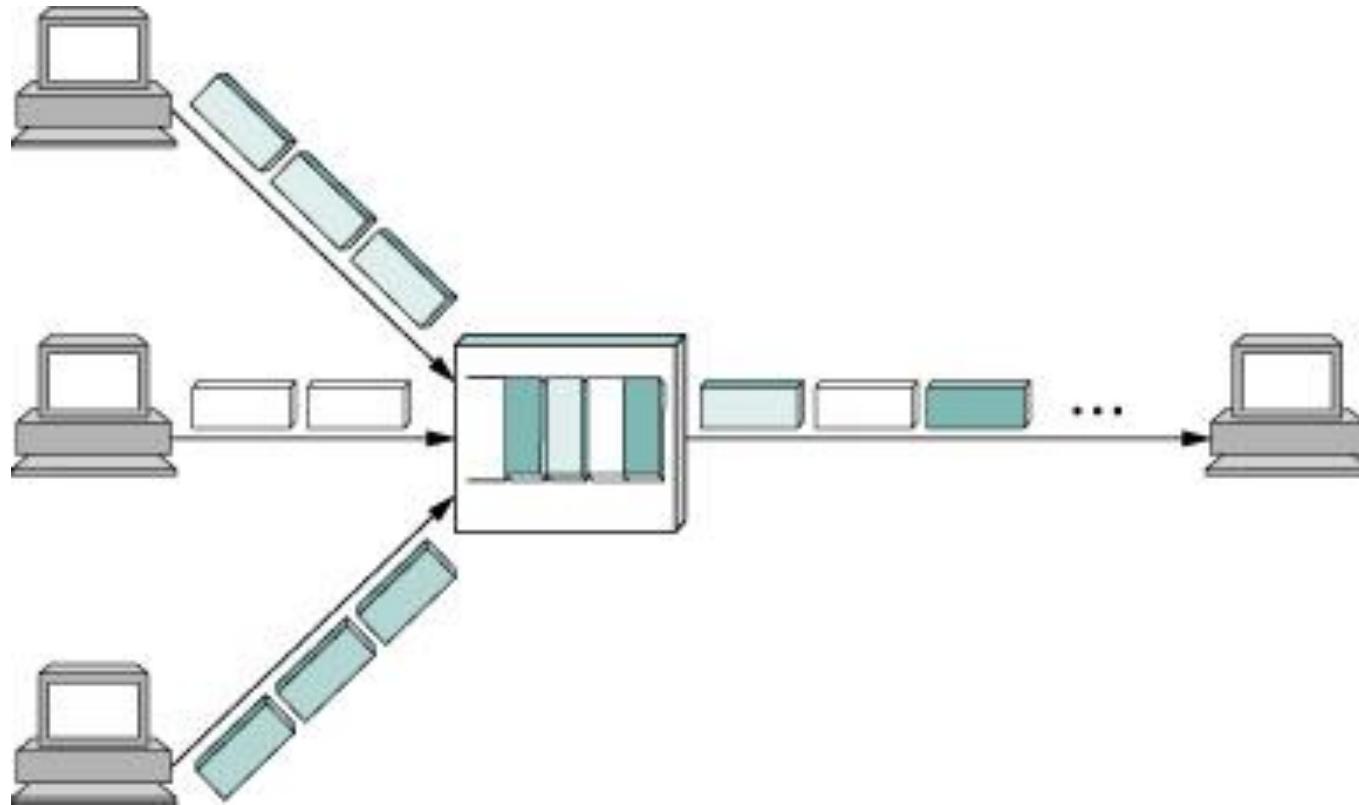
Single Shared Link

29

- Each flow sends a sequence of packets over the physical link, with a decision made on a packet-by-packet basis as to which flow's packet to send next
- If only one flow has data to send, then it can send a sequence of packets back-to-back
- If more than one of the flows have data to send, then their packets are interleaved on the link

Multiple flows on a Shared Link

30



A Switch multiplexing Packets from Multiple Sources onto one Shared Link

Packet Switching Decision

31

- Each switch in a packet switched network makes this decision independently, on a packet-by-packet basis
- One of the issues that faces a network designer is how to make decision in a fair manner
- A switch could be designed to service packets
 - On a first-in-first-out (FIFO) basis
 - Round-robin manner

QoS

32

- Switching might be done to ensure that certain flows receive a particular share of the link's bandwidth,
- Or that they never have their packets delayed in the switch for more than a certain length of time
- A network that attempts to allocate bandwidth to particular flows is sometimes said to support *quality of service (QoS)*

Congestion

33

- In the above example the switch has to multiplex three incoming packet streams onto one outgoing link
- It is possible that the switch will receive packets faster than the shared link
 - In this case, the switch is forced to buffer these packets in its memory
- Switch receive packets faster than it can send them for an extended period of time
- Switch will eventually run out of buffer space, and some packets will have to be dropped
- This state of switch operation is called as *congested*

Requirements

Support for Common Services

Applications Programs on Networks

35

- In simple the computer network is delivering packets among a collection of computers
- A network as providing the means for a set of application processes that are distributed over those computers to communicate
- The application programs running on the hosts connected to the network must be able to communicate in a meaningful way

Applications Programs on Networks

36

- When two applications need to communicate with each other
- There are a lot of complicated things that need to happen beyond simply sending a message from one host to another
- One option would be for application designers to build all that complicated functionality into each application program
- Many applications need common services, it is much more logical to implement those common services once
 - Such that application designer build the application using those services

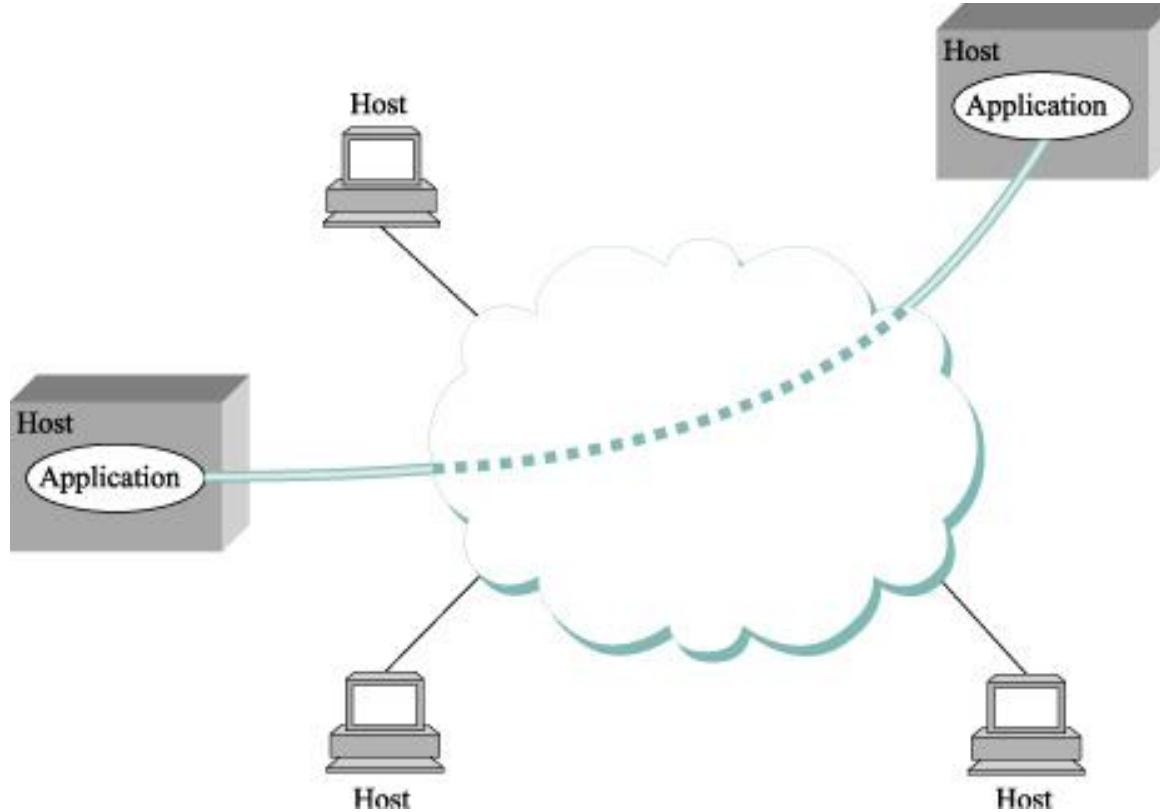
Network Common Services

37

- Network provides logical channels over which application-level processes can communicate with each other
- Each channel provides the set of services required by that application
- A cloud is abstractly represent connectivity among a set of computers
- A channel connects one process to another
- A channel is like a pipe connecting two applications
 - Sending application can put data in one end and expect that data to be delivered by the network to the application at the other end of the pipe

Channel

38



Processes communicating over an abstract channel

Identifying Common Communication Patterns

39

- Involves
 - Understanding the common needs of a representative collection of applications
 - Extracting their common communication requirements
 - Incorporating the functionality that meets these requirements in the network
- One of the earliest applications supported on any network is a file access program
 - FTP
 - NFS

Identifying Common Communication Patterns

40

- FTP/ NFS
 - Whole files are transferred across the network or only single blocks of the file are read/written at a given time
- The communication component of remote file access is characterized by a pair of processes
 - One that requests that a file be read or written
 - Client
 - A second process that honors this request
 - Server

Read and Write – Server & Client

41

- Reading a file involves
 - ▣ The client sending a small request message to a server
 - ▣ The server responding with a large message that contains the data in the file
- Writing works in opposite way
 - ▣ The client sends a large message containing the data to be written to the server
 - ▣ The server responds with a small message confirming that the write to disk has taken place

Video Applications

42

- Two types of applications
 - Videoconferencing
 - Video on demand
- Channels are
 - Request/reply channels
 - Message stream channels
- Request/reply channel would be used by the file transfer and digital library applications

Request/reply Channel

43

- It would guarantee that every message sent by one side is received by the other side
 - ▣ only one copy of each message is delivered
- The request/reply channel might also protect the privacy and integrity of the data that flows over it
 - ▣ Unauthorized parties cannot read or modify the data being exchanged between the client and server processes

Message Stream Channels

44

- Used by both the vide-on-demand and videoconferencing applications
- It is parameterized to support both one-way and two-way traffic and to support different delay properties
- The message stream channel might not need to guarantee that all messages are delivered
- A video application can operate adequately even if some video frame is not received
- The message stream channel might need to support multicast
 - So that multiple parties can participate in the teleconference or view the video

Channels/pipes

45

- A network designer to strive for the smallest number of abstract channel types that can serve the largest number of applications
 - ▣ There is danger in trying to get away with too few channel abstractions
- With change in application the network designers will probably be inventing new types of channels and adding options to existing channels
 - ▣ for as long as application programmers are inventing new applications

Bit Pipe

46

- It is easiest to view host-to-host connectivity of the underlying network as simply providing a bit pipe
 - With any high-level communication semantics provided at the end hosts
- The advantage of this approach is it keeps the switches in the middle of the network as simple as possible
 - They simply forward packets
 - But it requires the end hosts to take on much of the burden of supporting semantically rich process-to-process channels
- The alternative is to push additional functionality onto the switches, thereby allowing the end hosts to be “dumb” devices

Reliability

47

- Reliable message delivery is one of the most important functions that a network can provide
- It is difficult determine how to provide this reliability
 - Without understanding how networks can fail
- The computer networks do not exist in a perfect world
 - Physical Problems
 - Machine crash and later are rebooted
 - Fiber are cut
 - Electrical interference corrupts bits in the data being transmitted
 - Switches run out of buffer space
 - Software
 - The software that manages the hardware sometimes forwards packets into oblivion

Reliability – Bit Level Failure

48

- The major requirement of a network is to recover from certain kind of failures
- Three general cases of failures in the network is
- Bit Errors
 - ▣ The packet transmitted over a physical link, bit errors may be introduced into the data
 - i.e., 1 is turned into a 0 or vice versa
- Burst error – several consecutive bits are corrupted
- Bit errors typically occurs because the outside forces
 - ▣ Such as lightening strikes, power surges and microwave ovens, interface with the transmission of data

Reliability

49

- The good news is that bit errors are fairly rare
 - Affecting on average only one out of every 10^6 to 10^7 bits on a typical copper base cable
 - One out of every 10^{12} to 10^{14} bits on a typical optical fiber
- There are techniques that detect these bit errors with high probability
- It is sometimes possible to correct for such errors
- If damage is so bad, than it is necessary to discard the entire packet
 - In such a case, the sender may be expected to retransmit the packet

Reliability – Packet Failure

50

- Failure is at the packet, rather than the bit level
- i.e., a complete packet is lost by network
- One reason this can happen is that the packet contains an uncorrectable bit error and therefore has to be discarded
- A switch that is forwarding it from one link to another – is so over loaded that it has no place to store the packet, and therefore is forced to drop it (congestion)

Reliability – Packet Failure

51

- Software running on one of the nodes that handles the packet makes a mistake
 - ▣ It might incorrectly forward a packet out on the wrong link
 - ▣ So packet never finds its way to the ultimate destination
- One of the main difficulties in dealing with lost packets is distinguished between a packet that is indeed lost and one that is merely late in arriving at the destination

Reliability – Node and Link Failure

52

- Failure is at node and link level
- A physical link is cut or the computer it is connected to crashes
- This can be caused by software that crashes a power failure, or a reckless backhoe operator
- Failure due to misconfiguration of a network device are also common
- Any of these failures can eventually be corrected, they can have a dramatic effect on the network for an extended period of time

Reliability – Node and Link Failure

53

- In a packet switched network, it is sometimes possible to route around a failure node or link

NETWORK ARCHITECTURE

9 August 2023

Dr Noor Mohammad Sk

Network Architecture

55

- A computer network must provide general, cost-effective, fair, and robust connectivity among a large number of computers
- To deal with the network complexity, network designers have to develop a general blueprint – usually called network architectures
- Network architectures guide the design and implementation of network
- Two of the most widely referenced architectures –
 - The OSI architecture
 - The Internet architecture

Network Architecture

Layering and Protocols

Layering – Abstraction

57

- When a system gets complex, the system designer introduces another level of abstraction
- An abstraction is to define a unifying model that can capture some important aspect of the system
- An abstraction for applications that hides the complexity of the network from application writers
- Abstraction normally leads to layering in network systems

Layering

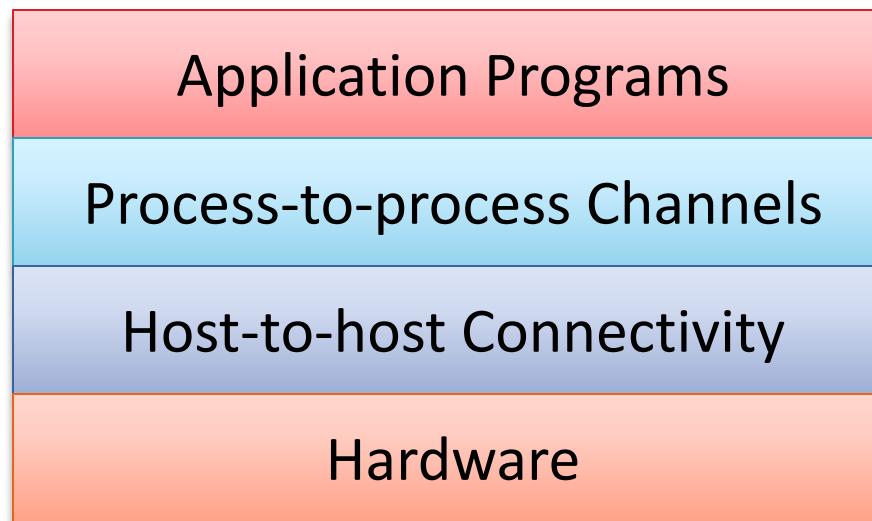
58

- The general idea is to start with the services offered by the underlying hardware and
- Add a sequence of layers, each providing a higher (more abstract) level of service.
- The services provided at the high layers are implemented in terms of the services provided by the low layers

Layering – Example

59

- A simple network as having two layers of abstraction sandwiched between the application program and underlying hardware



Example of a layered network system

Layering – Example

60

- The layer immediate above the hardware in this case might provide host-to-host connectivity
 - Abstracting away the fact that there may be an arbitrarily complex network topology between any two hosts
- The next layer up builds on the available host-to-host communication service and provides support for process-to-process channels

Layering

61

- Layering offers two features
- it decomposes the problem of building a network into more manageable components
 - ▣ One can implement several layers, each of which solves one part of the problem
- It provides a more modular design
 - ▣ If you want to add some new services, you may only need to modify the functionality at one layer
 - ▣ By reusing the functions provided at all the other layers

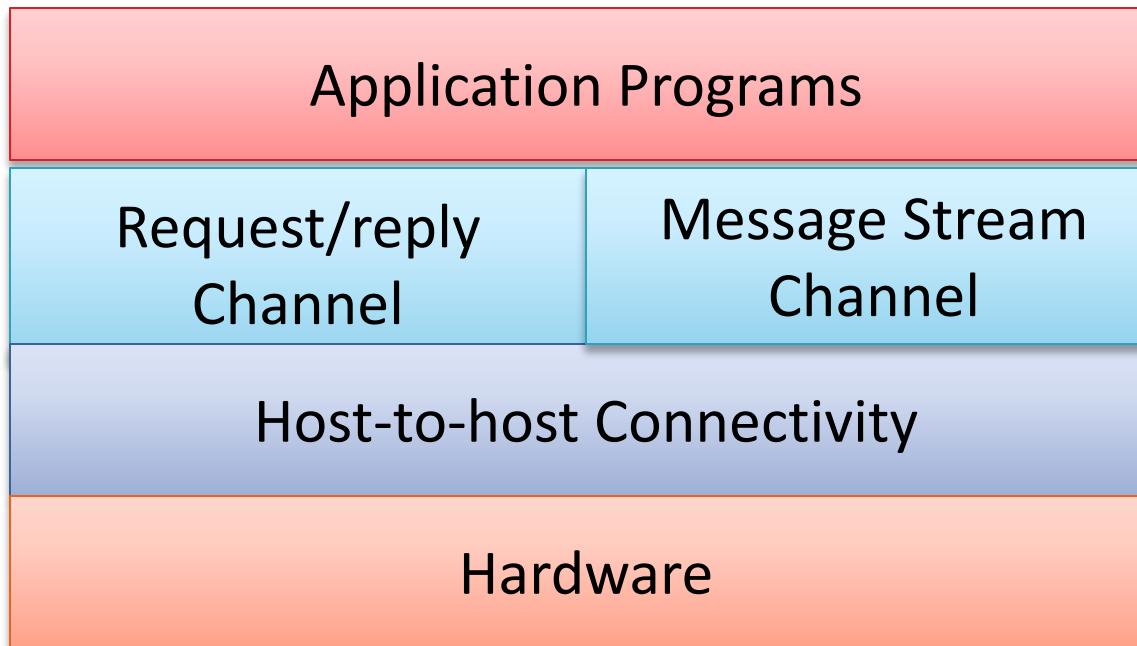
Layering – Multiple Abstraction

62

- Multiple abstractions are provided at any given level of the system
 - Each provides a different services to the higher layers but building on the same low-level abstractions
- Lets consider the two channels
 - A request/reply services
 - A message stream services
- These two channels might be alternative offerings at some level of a multilevel networking system

Layering – Multiple Abstraction

63



Layered System with alternative abstraction
available at a given layer

Protocol

64

- The abstract objects that make up the layers of a network system are called protocols
- A protocol provides a communication service that higher-level objects use to exchange the messages
- Example
 - Imagine a network that supports a request/reply protocol and a message stream protocol

Protocol – Interfaces

65

- Each protocol defines two different interfaces
- It defines a *service interface* to the other objects on the same computer that want to use its communication services
 - This service interface defines the operations that local objects can perform on the protocol
- Examples
 - A request/reply protocol would support operations by which an application can send and receive messages
 - An implementation of HTTP protocol support an operation to fetch a page of hypertext from a remote server

Protocol – Interfaces

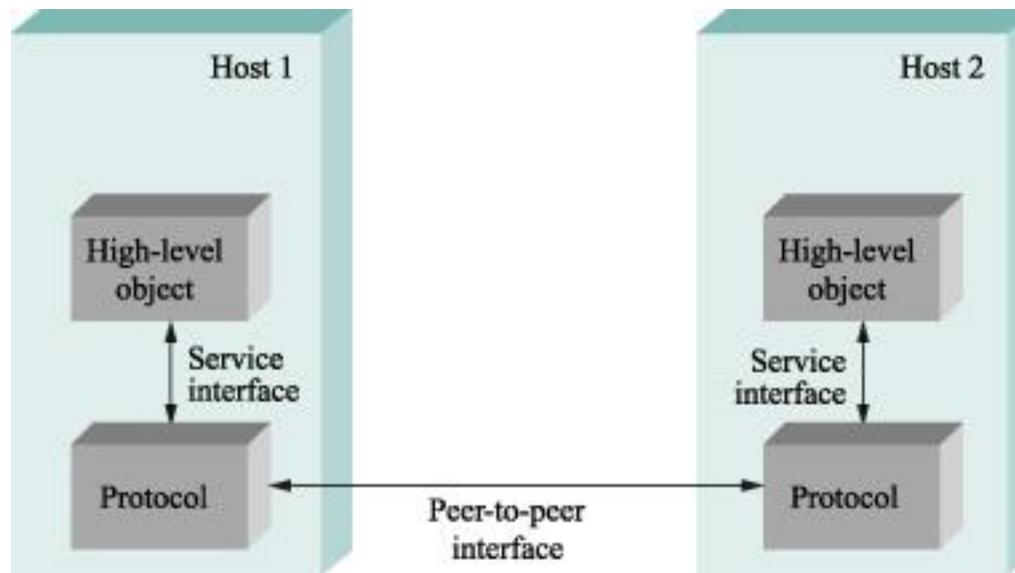
66

- A protocol defines a *peer interface* to its counterpart (peer) on another machine
- This interface defines the form and meaning of messages exchanged between protocol peers to implement the communication service
 - This would determine the way in which a request/reply protocol on one machine communicates with its peer on another machine
- Example: In the case of HTTP
 - The protocol specification defines in detail how a “GET” command is formatted
 - What arguments can be used with command

Protocol

67

- A protocol defines a communication services that it exports locally (the service interface)
- Along with a set of rules governing the messages that the protocol exchanges with its peer(s) to implement this service (the peer interface)



Protocol

68

- Peer-to-peer communication is indirect
 - Except at the hardware level where peers directly communicate with each other over a link
- Each protocol communicates with its peer by passing messages to some lower-level protocol
 - Which in turn delivers the message to its peers
- In addition, there are potentially multiple protocols at any given level, each providing a different communication service
- Therefore represent suite of protocols that make up a network system with a protocol graph

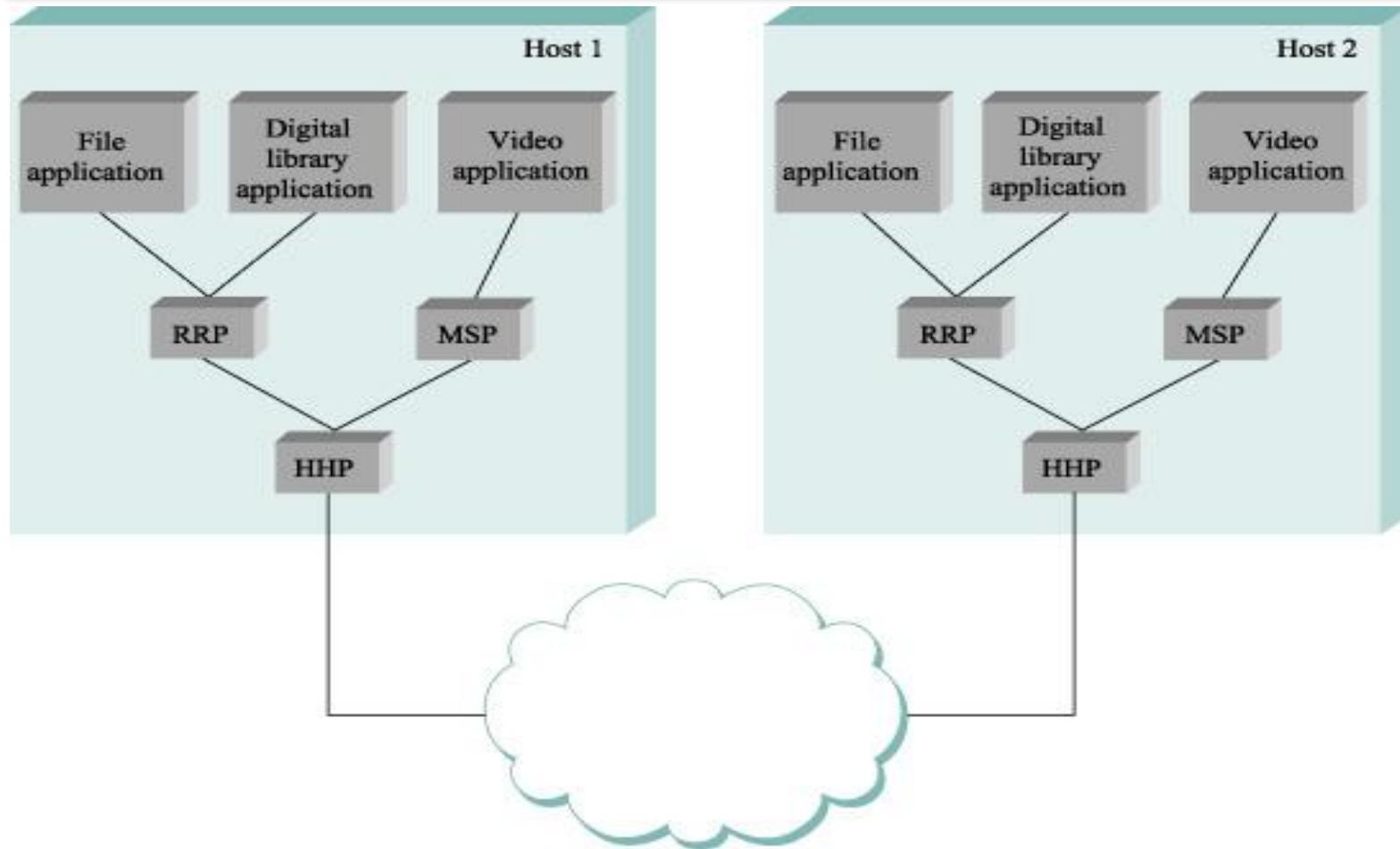
Protocol Graph

69

- The nodes of the graph correspond to protocols, and the edges represent a *depends on* relation
- Example
- The request/reply protocol (RRP) and message stream protocol (MSP) implement two different types of process-to-process channels
- Both depend on Host-to-host protocol (HHP), which provides a host-to-host connectivity service

Protocol Graph – Example

70



Protocol Graph – Example

71

- Host 1:
 - The file access program on host 1 wants to send a message to its peer on host 2 using communication services offered by protocol RRP
 - In this case, the file application asks RRP to send the message on its behalf
 - To communicate with its peer, RRP then invokes the services of HHP
 - which in turn transmits the message to its peer on the other machine
- Host 2:
 - Once the message has arrived at protocol HHP on host 2
 - HHP passes the message up to RRP, which in turn delivers the message to the file application

Protocol – Interface Vs Module

72

- Protocols is used in two different ways
 - ▣ **Interfaces** – the operations defined by the service interface and the form and meaning of messages exchanged between peers
 - ▣ **Modules** – that actually implements required interfaces
- Protocol specification distinguish the given protocol is an interface type or module type

Protocol – Specifications

73

- Generally expressed using combination of
 - prose, pseudocode, state transition diagrams, pictures of packet formats and other abstract notations
- A given protocol can be implemented in different ways by different programmers
- The challenge is to ensuring that two different implementations of the same specification can successfully exchange messages
- Two or more protocol modules that do accurately implement a protocol specification are said to *interoperate* with each other

Protocols

74

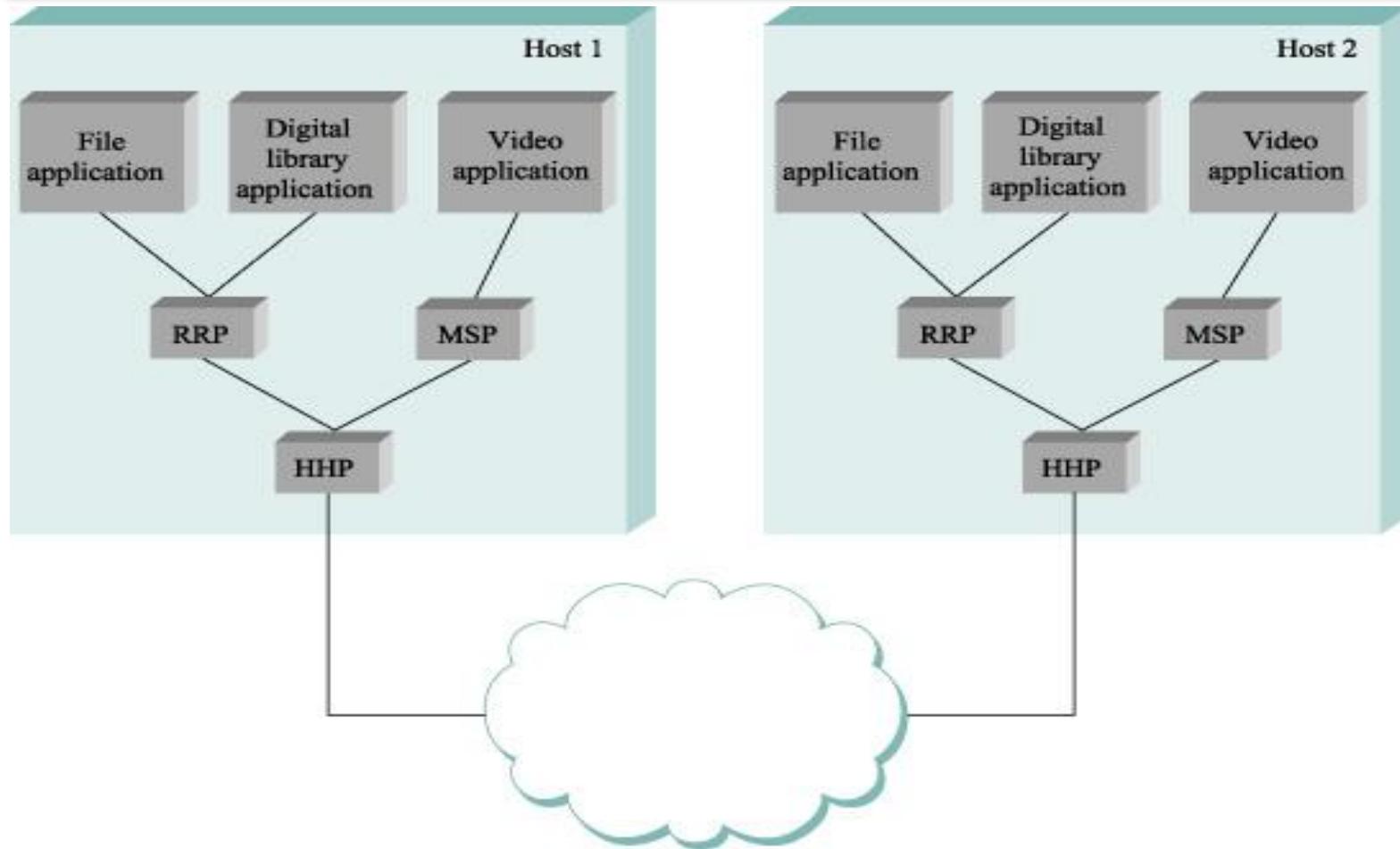
- We can imagine many different protocols and protocol graphs that satisfy the communication requirements of a collection of applications
- Standardization:
 - International Standard Organization (ISO)
 - Internet Engineering Task Force (IETF)
- Establish/defines policies for a particular protocol graph
- Network Architecture: The set of rules governing, the form and content of a protocol graph

Network Architecture

Encapsulation

Example – A protocol Graph

76



Encapsulation

77

- When one of the application programs sends a message to its peer by passing the message to protocol RRP
- From RRP perspective, the message it is given by the application is an uninterpreted string of bytes
- RRP does not care that these bytes represent an array of integers, an email message, a digital image, or whatever
 - ▣ It simply charged with sending them to its peer
- However, RRP must communicate control information to its peer, instructing it how to handle the message when it is received

Encapsulation

78

- RRP does this by attaching a header to the message
- A header is a small data structure
 - ▣ From a few bytes to a few dozen bytes
 - ▣ Used among peers to communicate with each other
- Headers are usually attached to the front of a message
- In some cases, this peer-to-peer control information is sent at the end of the message
 - ▣ Called as a trailer
- The exact format for the header attached by the RRP is defined by its protocol specification

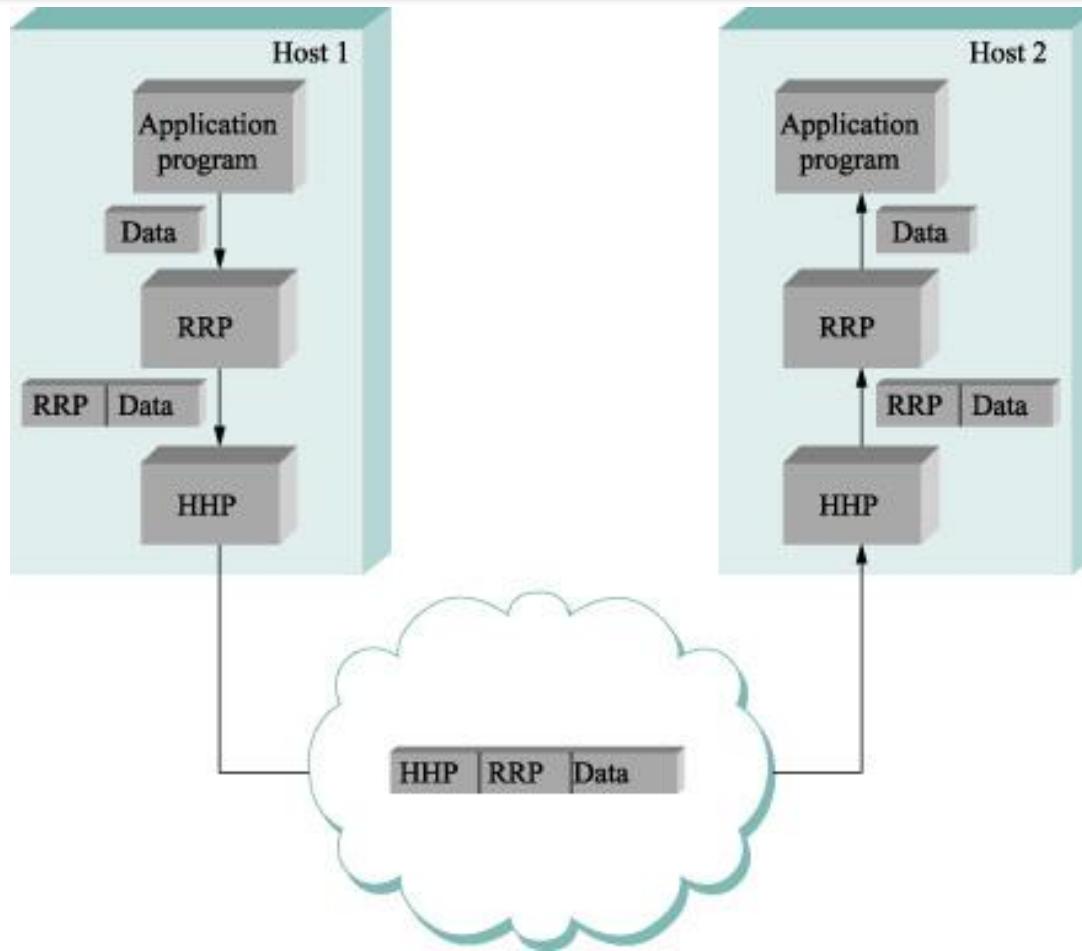
Encapsulation

79

- The rest of the message
 - ▣ The data being transmitted on behalf of the application is called the message's *body* or *payload*
 - ▣ It means application's data is encapsulated in the new message created by protocol RRP
- This process of encapsulation is then repeated at each level of the protocol graph

Encapsulation Example

80



High level messages are encapsulated inside of low-level message

Example

81

- HHP encapsulate RRP's message by attaching a header of its own
 - If we now assume that HHP sends the message to its peer over some network
 - Then when the message arrives at the destination host, it process it in the opposite order
- HHP first interprets the HHP header at the front of the message (i.e., takes whatever action is appropriate given the contents of the header),
- Passes the body of the message (but not the HHP header) up to the application program

Example

82

- The message passed up from RRP to the application on host 2 is exactly the same message as the application passed down to RRP on host 1
- The application does not see any of the headers that have been attached to it to implement the lower-level communication services
- **Node** in the network (e.g., switches and router) may inspect the HHP header at the front of the message

Encapsulation

83

- A low-level protocol does not interpret the message it is given by some high-level protocol
- It does not know how to extract any meaning from the data contained in the message
- Encryption
 - The low-level protocol applies some simple transformation to the data it is given, such as to compress or encrypt it
 - In this case, the protocol is transforming the entire body of the message, including both the original *application's data* and *all the headers* attached to that data by higher-level protocols

Network Architecture

Multiplexing and Demultiplexing

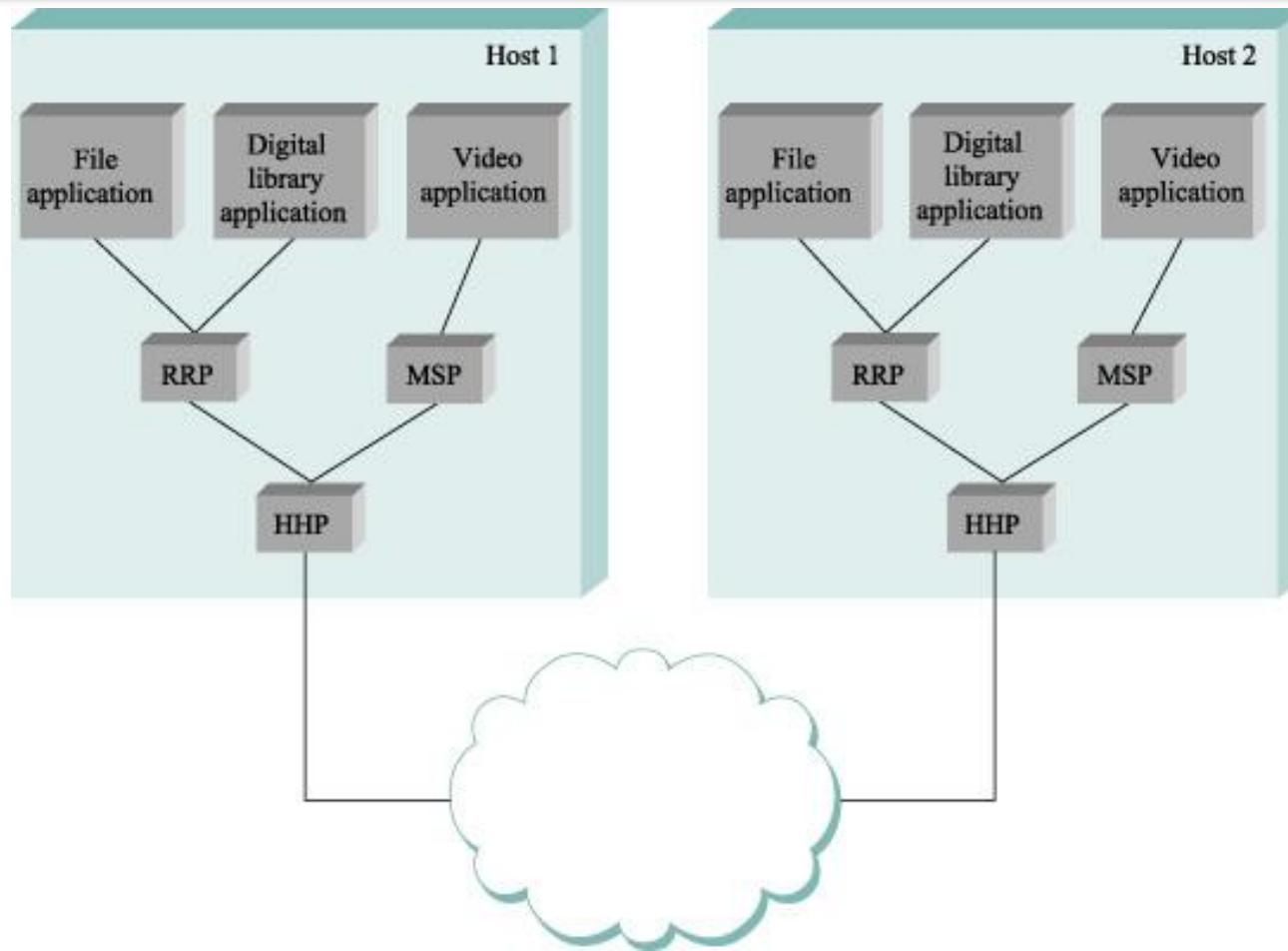
Multiplexing and Demultiplexing

85

- The fundamental idea of packet switching is to multiplex multiple flows of data over a single physical link
- The same idea applies up and down the protocol graph, not to switching nodes

Example

86



Multiplexing and Demultiplexing

87

- RRP implementing a logical communication channel, with message from two different applications multiplexed over this channel at the source host
- And then demultiplexed back to the appropriate application at the destination host
- Practically
 - A header that RRP attaches to its message contains an **identifier** that records the application to which the message belongs
 - This identifier is called as RRP's *demultiplexing key* or *demux key*

Multiplexing and Demultiplexing

88

- At the source host, RRP includes the appropriate demux key in its header
- When the message is delivered to RRP on the destination host
 - It strips its header, examines the demux key and demultiplexes the message to the correct application
- RRP is not unique in its support for multiplexing
 - Nearly every protocol implements this mechanism
- Example:
 - HHP has its own demux key to determine which messages to pass up to RRP and which to pass up to MSP

Multiplexing and Demultiplexing

89

- There is no uniform agreement among protocols
- Even those within a single network architecture
 - On exactly what constitutes a demux key
- Some protocols use 8-bit field (meaning they can support only 256 high-level protocols) and others use 16- or 32-bit fields
- Some protocols have **a single** demultiplexing field in their header
 - The same demux key is used on both sides of the communication
- Some protocols have **a pair of** demultiplexing fields in their header
 - Each side uses a different key to identify the high-level protocol (or application program) to which the message is to be delivered

90

Network Architecture

OSI Architecture

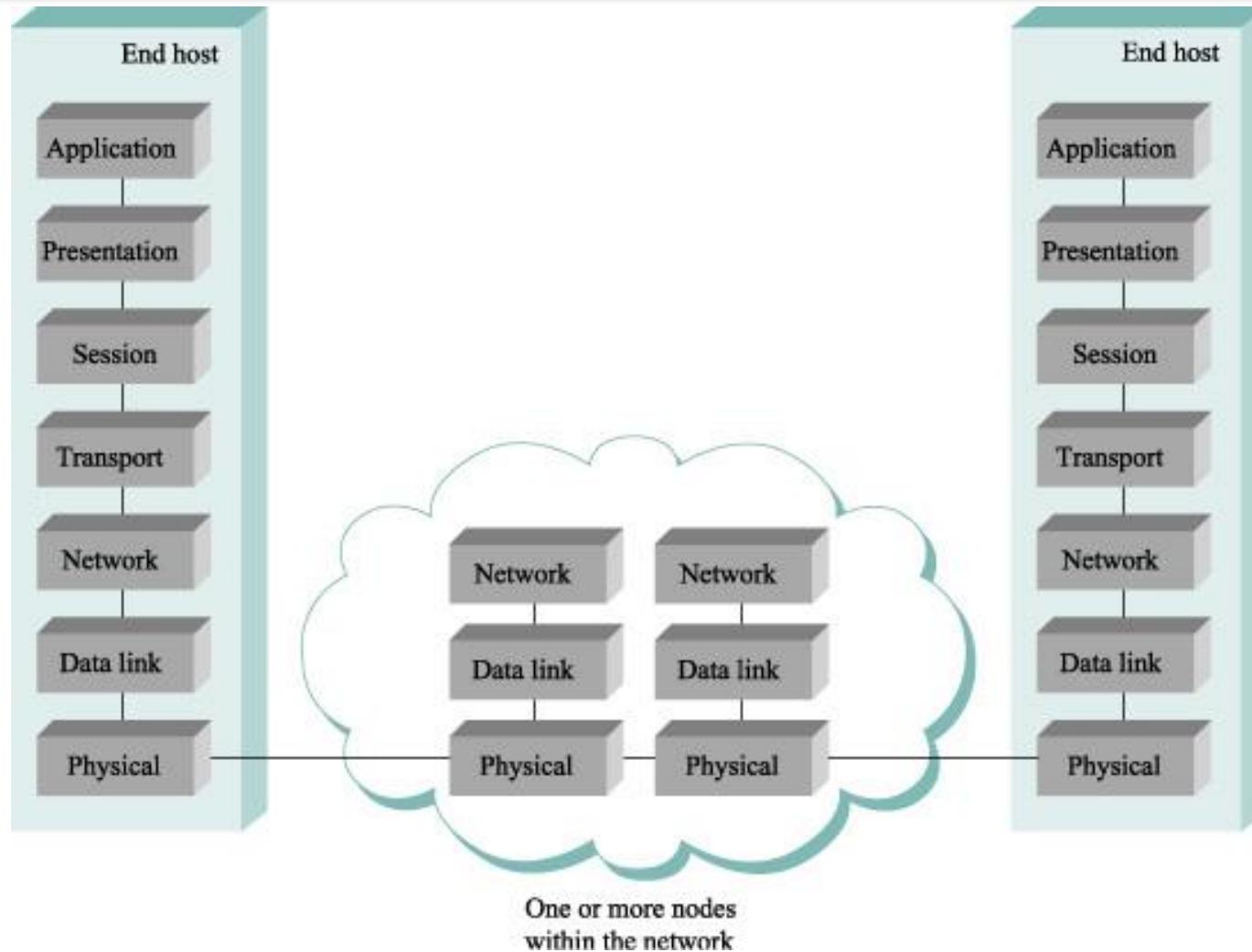
OSI Architecture

91

- The ISO formally define a common way to connect computers
 - ▣ OSI: Open System Architecture
- OSI defines a partitioning of network functionality into seven layers
 - ▣ Where one or more protocols to implement the functionality assigned to a given layer
- ISO and ITU publishes a series of protocol specifications based on the OSI architecture
 - ▣ This series sometimes called “Xdot” series
 - ▣ X.25, X.400, X.500 and so on

OSI Network Architecture

92



OSI Network Architecture

93

- Physical layer handles the transmission of raw bits over communication link
- Data link layer then collects a stream of bits into a large aggregate called a frame
 - ▣ Network adaptors, along with device drivers running in the node's OS, typically implement the data link level
 - ▣ This means that frames, not raw bits, are actually delivered to hosts
- The network layer handles routing among nodes within a packet-switched network
 - ▣ At this layer, the unit of data exchanged among nodes is typically called a packet rather than a frame

OSI Network Architecture

94

- The lower three layers are implemented on all network nodes, including switches within the network and hosts connected along the exterior the network
- Transport layer implements a process-to-process channel
 - ▣ Here, the unit of data exchanged is commonly called a **message** rather than a packet or a frame
 - ▣ The transport layer and higher layers typically run only on the end hosts and not on the intermediate switches or routers

OSI Architecture

95

- Application Layer is the top (seventh) layer
 - Application layer protocols include things like the File Transfer Protocol (FTP)
 - Which defines a protocol by which file transfer application can interoperate
- Presentation Layer is concerned with the format of exchanged between peers
- Example:
 - Whether an integer is 16, 32, or 64 bits long and whether the most significant byte is *transmitted* first or last or how video stream is formatted

OSI Architecture

96

- Session Layer provides name space that is used to tie together the potentially different transport streams that are part of a single application
 - **Example:** It might manage an audio stream and a video stream that are being combined in a teleconferencing application

Network Architecture

Internet Architecture

Internet Architecture

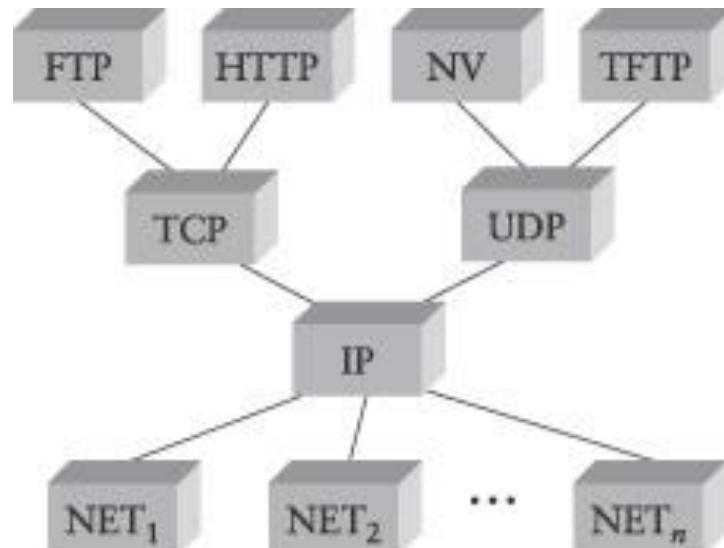
98

- Is also called as TCP/IP architecture
- Internet architecture evolved out of experiences with an earlier packet-switched network called the ARPANET
- Both the Internet and the ARPANET were funded by the Advanced Research Project Agency (ARPA) – US Defense
- The Internet and ARPANET were around before the OSI architecture

Internet Architecture

99

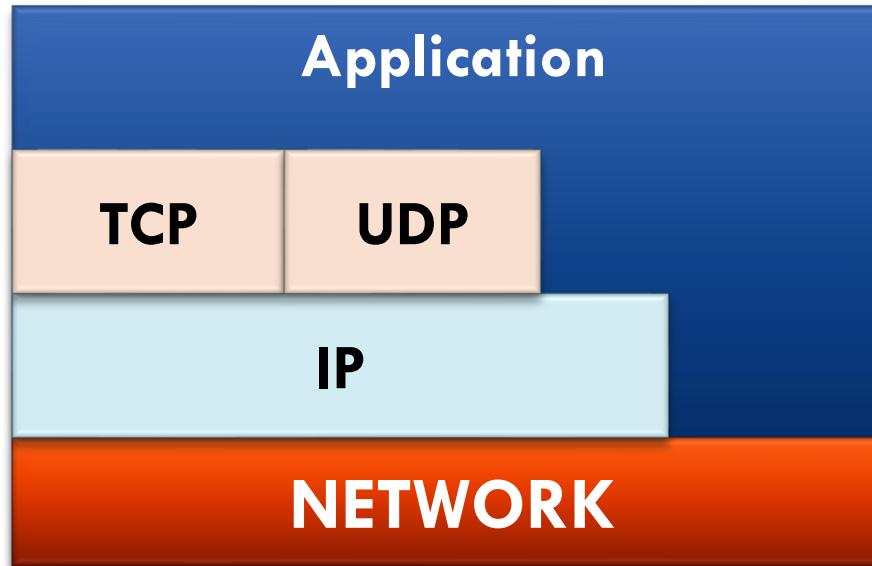
- A four layer model



Internet Protocol Graph

Internet Architecture

100



- An alternative view to the Internet architecture
- The “network” layer shown here is sometimes referred to as the “subnetwork” or “link” layer

Internet Architecture

101

- At the lowest level are a wide variety of network protocols, denoted NET_1 , NET_2 and so on
- In practice these protocols are implemented by a combination of hardware (e.g., network adaptor) and software (e.g., a network device driver)
- For example: Ethernet or Fiber Distributed Data Transfer (FDDI) protocols at this layer

Internet Architecture

102

- The second layer consists of a single protocol – Internet Protocol(IP)
 - This protocol supports the multiple networking technologies into a single logical internetwork
- The third layer contains two main protocols
 - The Transmission Control Protocol (TCP)
 - The User Datagram Protocol (UDP)
 - TCP and UDP provide alternative logical channels to application programs
 - TCP provides reliable byte-stream channel
 - UDP provides an unreliable datagram delivery channel
 - UDP and TCP protocols are also called as end-to-end protocols
 - We can refer this layer or protocol as Transfer protocol

Internet Architecture

103

- Above the transport layer/protocol is Application protocols
- Such as
 - FTP, TFTP (Trivial File Transport Protocol)
 - Telnet (remote login)
 - SMTP (Simple Mail Transfer Protocol, or electronic mail)
 - HTTP (Hyper Text Transfer Protocol)

Internet Architecture Features

104

- Three features
- The internet layering does not imply strict layering
 - ▣ The application is free to bypass the defined transport layers and to directly use IP or one of the underlying networks
 - ▣ Programmers are free to define new channel abstractions or applications that run on top of any of the existing protocols

Internet Architecture features

105

- Protocol graph looks like a hourglass shape
 - Wide at the top, narrow in the middle, and wide at the bottom
 - This shape actually reflects the central philosophy of the architecture
 - i.e., IP serves as the focal point for the architecture
 - It defines a common method for exchanging packets among a wide collection networks
 - Above IP can be arbitrarily many transport protocols, each offering different channel abstraction to application programs
 - Below IP, the architecture allows for arbitrarily many different network technologies ranging from Ethernet to wireless to single point-to-point links

Internet Architecture features

106

- A new protocol is officially (IETF) included in the architecture
 - ▣ There needs to be both a protocol specification and at least one (and preferably two) representative implementations of the specification
 - ▣ This IETF cultural assumption of the design community helps to ensure that the architecture's protocols can be efficiently implemented

Reference

107

- **Larry L Peterson & B S Davie, Computer Networks: A Systems Approach**, 4 Edn, Morgan Kauffman Publishers, 2007

THANK YOU!!

9 August 2023

Noor Mohammad Sk

IMPLEMENTING NETWORK SOFTWARE

25 September
2023

Dr Noor Mohammad Sk

Implementing Network Software

2

- Network architectures and protocol specifications are essential things to define a good blue print
- The number of computers connected to the Internet has roughly doubled every 12 to 18 months since 1981
- Number of Internet users are increasing exponentially over the year
- Great success of Internet contributed by
 - A good architecture
 - Major contribution is its functionality is provided by the software running in general purpose computers
 - The significance of this is that new functionality can be added readily with “just a small matter of programming”

Application Programming Interface (Sockets)

3

- Since most network protocols are implemented in software (especially those high in the protocol stack)
- All computers implement their network protocols as part of the operating system
- The interface that OS provides to its networking subsystem
 - Is called as the network Application Programming Interface (API)
- Each operating system is free to define its own network API
 - Over time certain of these APIs have become widely supported

Application Programming Interface (Sockets)

4

- The *socket interface* originally provided by the Berkeley distribution of Unix
- It is supported in virtually all popular operating systems
- The advantage of industry-wide support for a single API is that applications can be easily ported from one OS to another
 - ▣ Developers can easily write applications for multiple OSs
- The network application programs typically interact with many parts of the OS other than the network

Application Programming Interface (Sockets)

5

- Two systems support the same network API does not mean that their file system, process, or graphic interfaces are the same
- Each protocol provides a certain set of services
- The API provides a *syntax* by which those services can be invoked in this particular OS
- The socket is the point where a local application process attaches to the network
- The interface defines operations for
 - creating a socket,
 - attaching the socket to network
 - Sending/receiving message through the socket, and closing the socket

Application Programming Interface (Sockets)

6

- First step to create a socket
 - *int socket(int domain, int type, int protocol)*
- The domain argument specifies the protocol family that is going to be used
 - PF_INET denotes the internet family
 - PF_UNIX denotes the Unix pipe facility
 - PF_PACKET denotes direct access to network interface(i.e., it bypasses the TCP/IP protocol stack)
- The type argument indicates the semantic of the communication
 - SOCK_STREAM denotes a byte stream
 - SOCK_DGRAM is an alternative that denotes a message-oriented service, such as that provided by UDP

Application Programming Interface (Sockets)

7

- On a server machine the application process performs a passive open
 - ▣ The server says that it is prepared to accept connections,
 - ▣ but it does not actually establish a connection
 - ▣ The server does the connection by invoking the following three operations
 - int bind(int socket, struct sockaddr *address, int addr_len)
 - int listen(int socket, int backlog)
 - int accept(int socket, struct sockaddr *address, int *addr_len)
- The bind operation: binds the newly created socket to the specified address

Application Programming Interface (Sockets)

8

- The listen operation defines how many connections can be pending on the specified *socket*
- Accept operation carries out the passive open
 - ▣ It is a blocking operation that does not return until a remote participant has established a connection
 - ▣ When it does complete, it returns a new socket that corresponds to this just established connection
- The address argument contains the *remote* participant's address

Application Programming Interface (Sockets)

9

- On the client machine, the application process performs an *active* open;
 - It says who it wants to communicate
 - *int connect(int socket, struct sockaddr *address, int addr_len)*
 - This process does not return until TCP has successfully established a connection at which time the application is free to begin sending data
- In practice, the client usually, specifies only the remote participant's address and lets the system fill in the local information

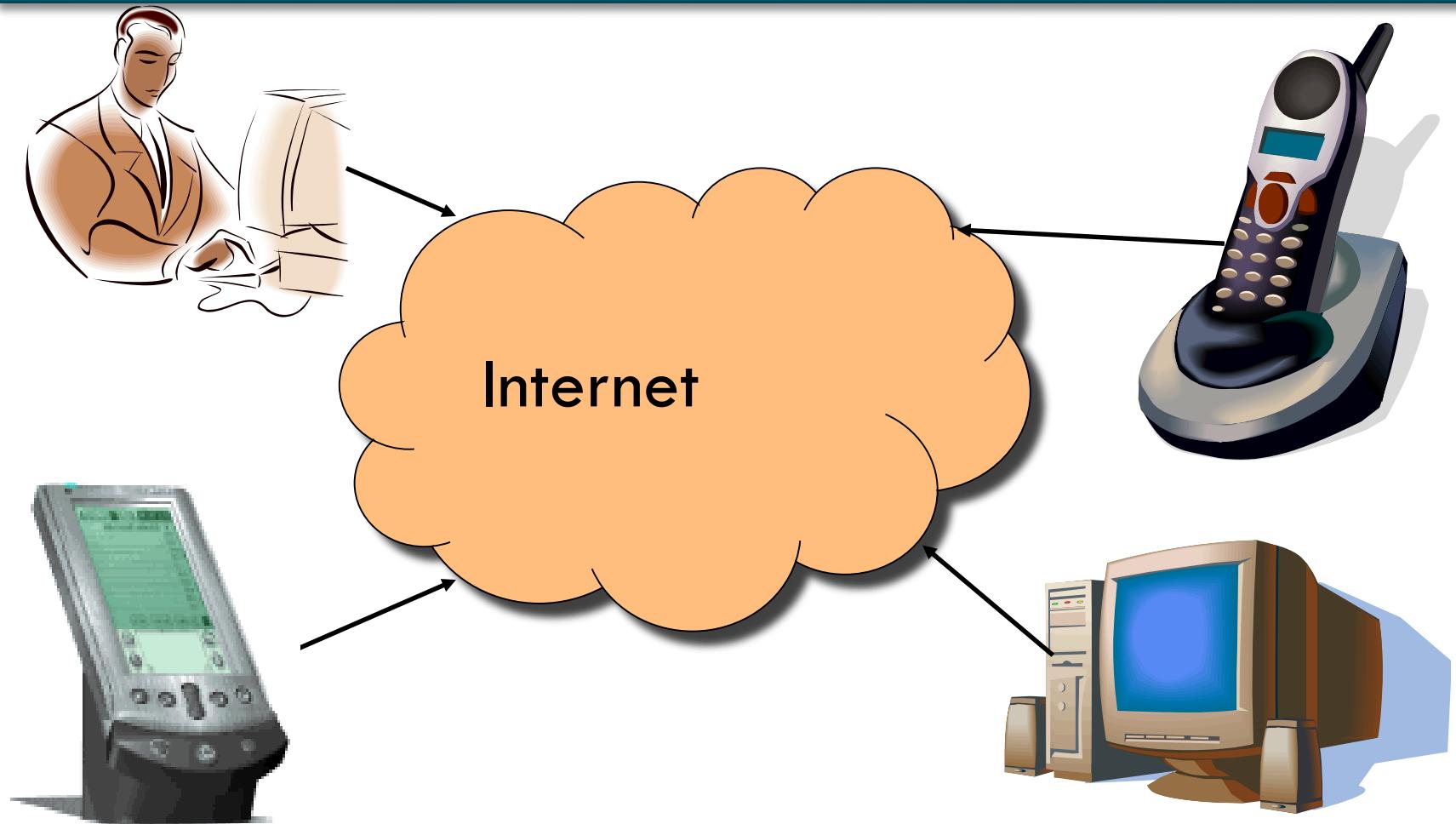
Application Programming Interface (Sockets)

10

- Server usually listens for messages on a well-known port, a client typically does not care which port it uses for itself
 - The OS simply selects an unused one
- Once connection is established, the application processes invoke the following two operations to send and receive data
 - *int send(int socket, char *message, int msg_len, int flags)*
 - Sends the given message over the specified socket
 - *int recv(int socket, char *buffer, int buf_len, int flags)*
 - Receives a message from the specified socket into the given buffer
 - Both the operations take a set of flags that control certain details of the operation

End System: Computer on the ‘Net

11

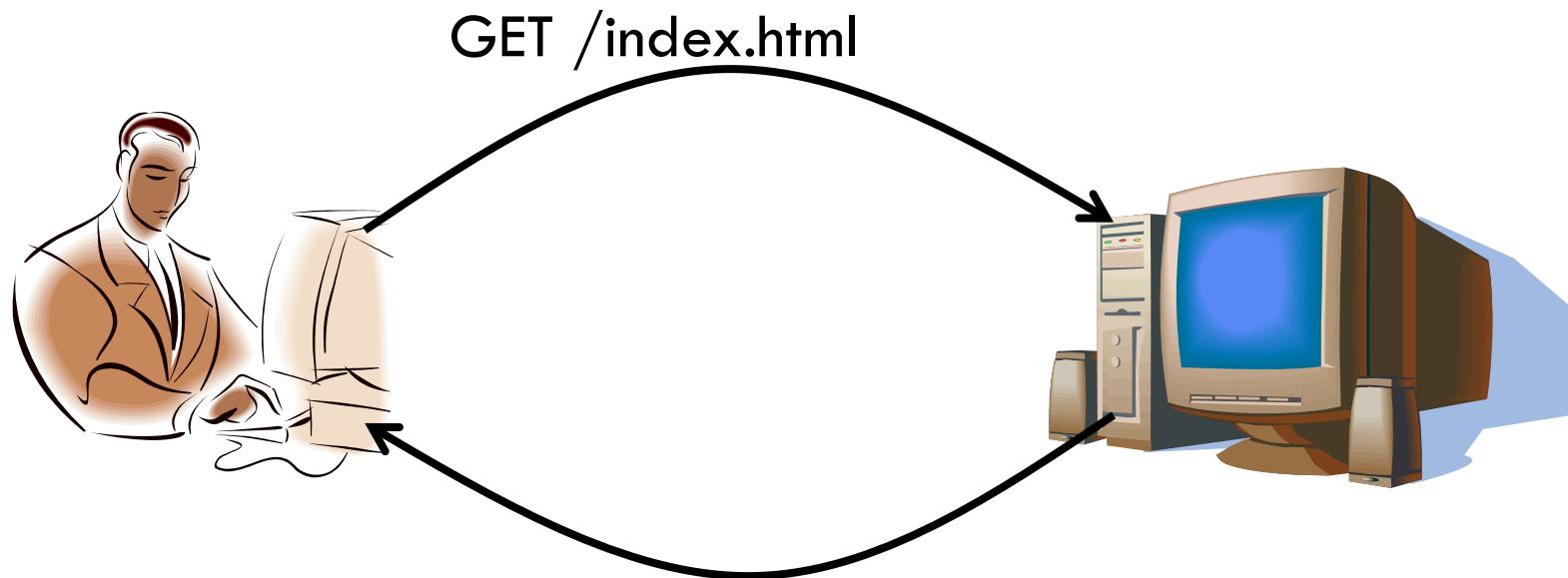


Also known as a “host”...

Clients and Servers

12

- Client program
 - Running on end host
 - Requests service
 - E.g., Web browser
- Server program
 - Running on end host
 - Provides service
 - E.g., Web server



Clients Are Not Necessarily Human

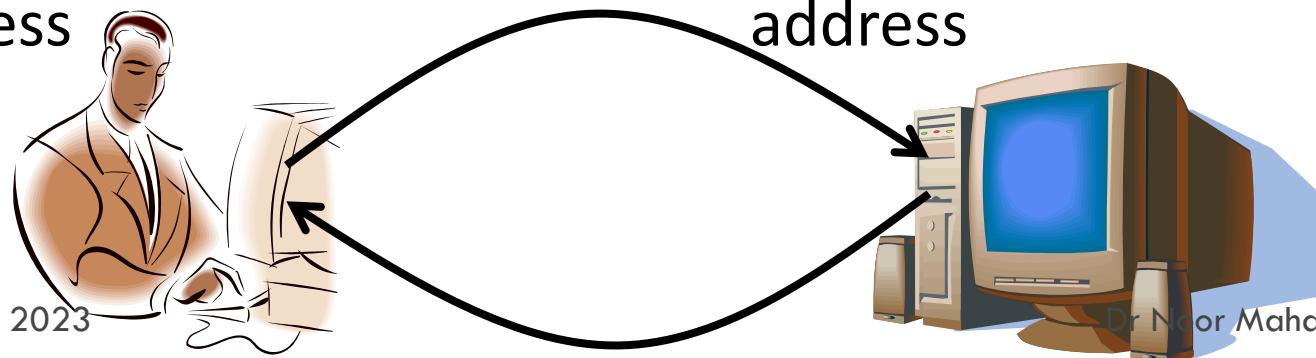
13

- Example: Web crawler (or spider)
 - Automated client program
 - Tries to discover & download many Web pages
 - Forms the basis of search engines like Google
- Spider client
 - Start with a base list of popular Web sites
 - Download the Web pages
 - Parse the HTML files to extract hypertext links
 - Download these Web pages, too
 - And repeat, and repeat, and repeat...

Client-Server Communication

14

- Client “sometimes on”
 - Initiates a request to the server when interested
 - E.g., Web browser on your laptop or cell phone
 - Doesn’t communicate directly with other clients
 - Needs to know server’s address
- Server is “always on”
 - Services requests from many client hosts
 - E.g., Web server for the www.iiitdm.ac.in Web site
 - Doesn’t initiate contact with the clients
 - Needs fixed, known address



Peer-to-Peer Communication

15

- No always-on server at the center of it all
 - ▣ Hosts can come and go, and change addresses
 - ▣ Hosts may have a different address each time
- Example: peer-to-peer file sharing
 - ▣ Any host can request files, send files, query to find a file's location, respond to queries, ...
 - ▣ Scalability by harnessing millions of peers
 - ▣ Each peer acting as both a client and server

Client and Server Processes

16

- Program vs. process
 - Program: collection of code
 - Process: a running program on a host
- Communication between processes
 - Same end host: inter-process communication
 - Governed by the operating system on the end host
 - Different end hosts: exchanging messages
 - Governed by the network protocols
- Client and server processes
 - Client process: process that initiates communication
 - Server process: process that waits to be contacted

Delivering the Data: Division of Labor

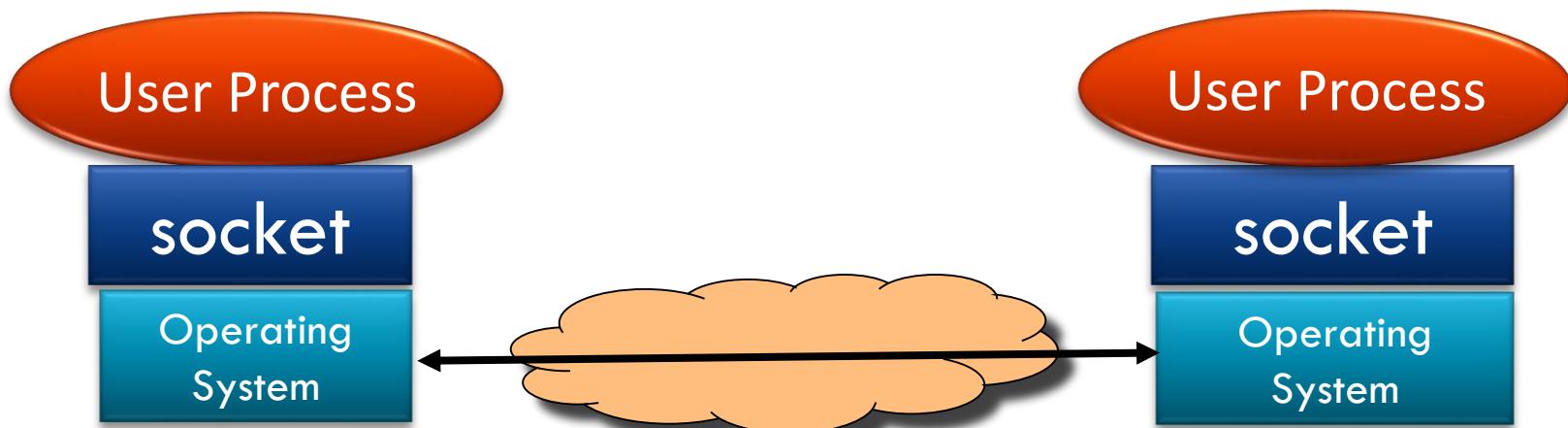
17

- Network
 - Deliver data packet to the destination host
 - Based on the destination IP address
- Operating system
 - Deliver data to the destination socket
 - Based on the destination port number (e.g., 80)
- Application
 - Read data from and write data to the socket
 - Interpret the data (e.g., render a Web page)

Socket: End Point of Communication

18

- Sending message from one process to another
 - Message must traverse the underlying network
- Process sends and receives through a “socket”
 - In essence, the doorway leading in/out of the house
- Socket as an Application Programming Interface
 - Supports the creation of network applications



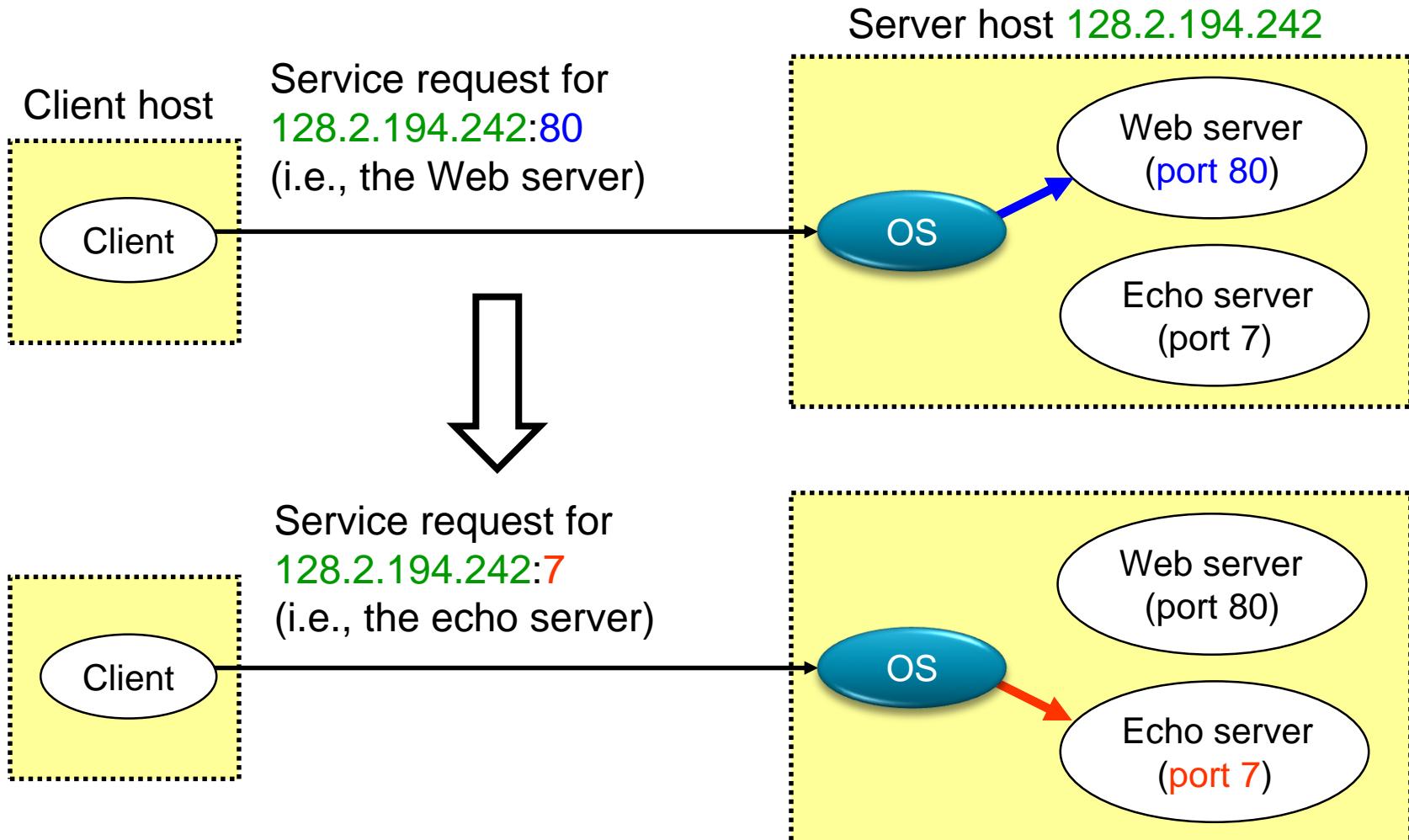
Identifying the Receiving Process

19

- Sending process must identify the receiver
 - ▣ The receiving end host machine
 - ▣ The specific socket in a process on that machine
- Receiving host
 - ▣ Destination address that uniquely identifies the host
 - ▣ An IP address is a 32-bit quantity
- Receiving socket
 - ▣ Host may be running many different processes
 - ▣ Destination port that uniquely identifies the socket
 - ▣ A port number is a 16-bit quantity

Using Ports to Identify Services

20



Knowing What Port Number To Use

21

- Popular applications have well-known ports
 - E.g., port 80 for Web and port 25 for e-mail
 - See <http://www.iana.org/assignments/port-numbers>
- Well-known vs. ephemeral ports
 - Server has a well-known port (e.g., port 80)
 - Between 0 and 1023 (requires root to use)
 - Client picks an unused ephemeral (i.e., temporary) port
 - Between 1024 and 65535
- Uniquely identifying traffic between the hosts
 - Two IP addresses and two port numbers
 - Underlying transport protocol (e.g., TCP or UDP)

Port Numbers are Unique per Host

22

- Port number uniquely identifies the socket
 - Cannot use same port number twice with same address
 - Otherwise, the OS can't demultiplex packets correctly
- Operating system enforces uniqueness
 - OS keeps track of which port numbers are in use
 - Doesn't let the second program use the port number
- Example: two Web servers running on a machine
 - They cannot both use port “80”, the standard port #
 - So, the second one might use a non-standard port #
 - E.g., <http://www.iiitdm.ac.in:8080>

UNIX Socket API

23

- Socket interface
 - Originally provided in Berkeley UNIX
 - Later adopted by all popular operating systems
 - Simplifies porting applications to different OSes
- In UNIX, everything is like a file
 - All input is like reading a file
 - All output is like writing a file
 - File is represented by an integer file descriptor
- API implemented as system calls
 - E.g., connect, read, write, close, ...

Typical Client Program

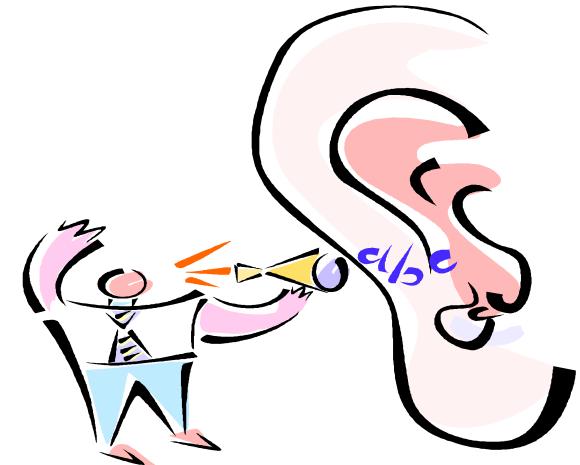
24

- Prepare to communicate
 - Create a socket
 - Determine server address and port number
 - Initiate the connection to the server
- Exchange data with the server
 - Write data to the socket
 - Read data from the socket
 - Do stuff with the data (e.g., render a Web page)
- Close the socket

Servers Differ From Clients

25

- Passive open
 - ▣ Prepare to accept connections
 - ▣ ... but don't actually establish
 - ▣ ... until hearing from a client
- Hearing from multiple clients
 - ▣ Allowing a backlog of waiting clients
 - ▣ ... in case several try to communicate at once
- Create a socket for each client
 - ▣ Upon accepting a new client
 - ▣ ... create a *new* socket for the communication



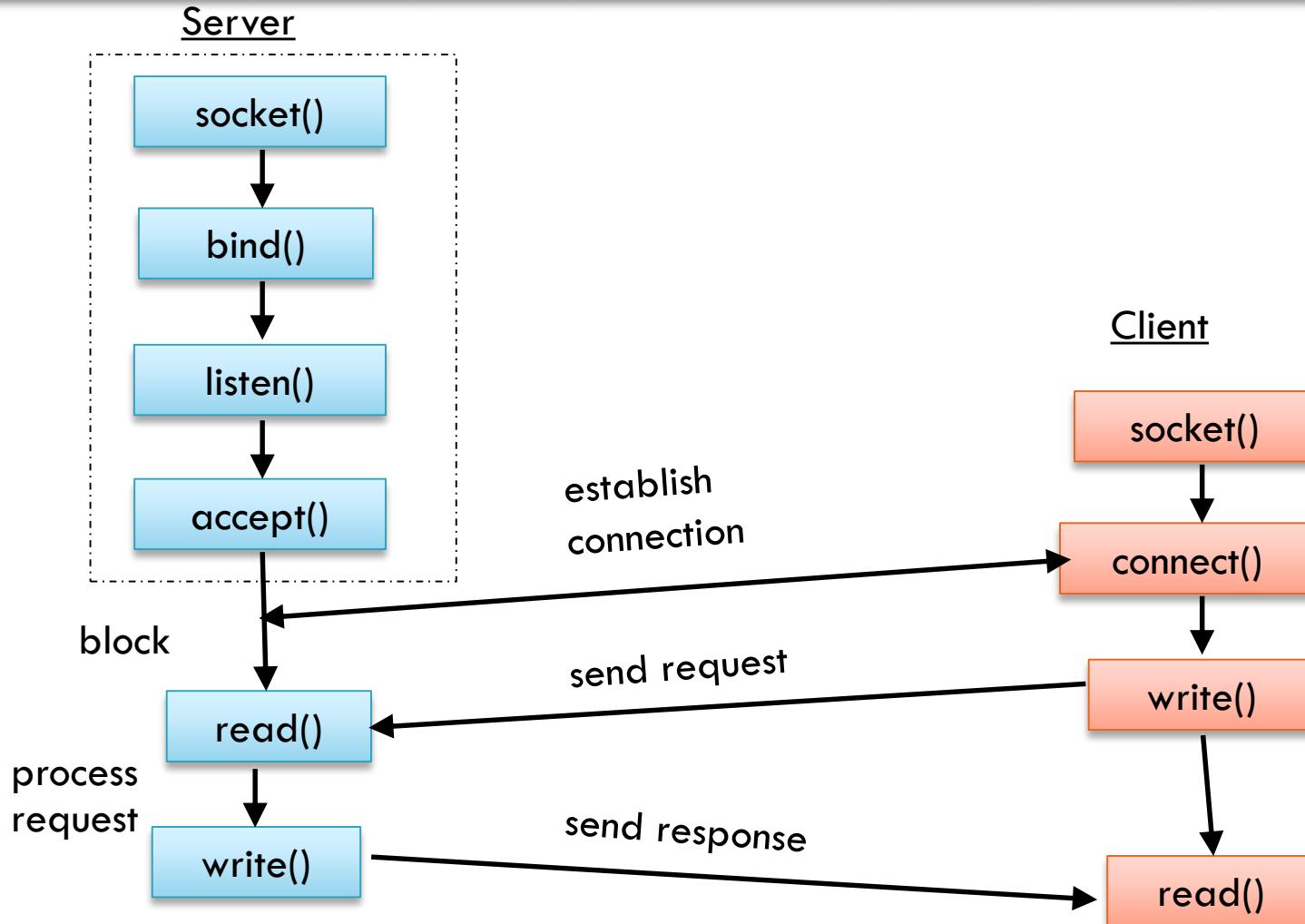
Typical Server Program

26

- Prepare to communicate
 - Create a socket
 - Associate local address and port with the socket
- Wait to hear from a client (passive open)
 - Indicate how many clients-in-waiting to permit
 - Accept an incoming connection from a client
- Exchange data with the client over new socket
 - Receive data from the socket
 - Do stuff to handle the request (e.g., get a file)
 - Send data to the socket
 - Close the socket
- Repeat with the next connection request

Putting it All Together

27



Client Creating a Socket: socket()

28

- Creating a socket
 - *int socket(int domain, int type, int protocol)*
 - Returns a file descriptor (or handle) for the socket
 - Originally designed to support any protocol suite
- Domain: protocol family
 - PF_INET for the Internet (IPv4)
- Type: semantics of the communication
 - SOCK_STREAM: reliable byte stream (TCP)
 - SOCK_DGRAM: message-oriented service (UDP)
- Protocol: specific protocol
 - UNSPEC: unspecified

Client: Learning Server Address/Port

29

- Server typically known by name and service
 - E.g., “www.cnn.com” and “http”
- Need to translate into IP address and port #
 - E.g., “64.236.16.20” and “80”
- Translating the server’s name to an address
 - ***struct hostent *gethostbyname(char *name)***
 - Argument: host name (e.g., “www.cnn.com”)
 - Returns a structure that includes the host address
- Identifying the service’s port number
 - ***struct servent *getservbyname(char *name, char *proto)***
 - Arguments: service (e.g., “ftp”) and protocol (e.g., “tcp”)
 - Static config in/etc/services

Client: Connecting Socket to the Server

30

- Client contacts the server to establish connection
 - Associate the socket with the server address/port
 - Acquire a local port number (assigned by the OS)
 - Request connection to server, who hopefully accepts
- Establishing the connection
 - *int connect (int sockfd,
 struct sockaddr *server_address,
 socketlen_t addrlen)*
 - Arguments: socket descriptor, server address, and address size
 - Returns 0 on success, and -1 if an error occurs

Client: Sending Data

31

- Sending data
 - ***ssize_t write (int sockfd, void *buf, size_t len)***
 - Arguments: socket descriptor, pointer to buffer of data to send, and length of the buffer
 - Returns the number of bytes written, and -1 on error

Client: Receiving Data

32

- Receiving data
 - ***ssize_t read(int sockfd, void *buf, size_t len)***
 - Arguments: socket descriptor, pointer to buffer to place the data, size of the buffer
 - Returns the number of characters read (where 0 implies “end of file”), and -1 on error
- Closing the socket
 - ***int close(int sockfd)***

Server: Server Preparing its Socket

33

- Server creates a socket and binds address/port
 - Server creates a socket, just like the client does
 - Server associates the socket with the port number (and hopefully no other process is already using it!)
 - Choose port “0” and let kernel assign ephemeral port
- Create a socket
 - *int socket (int domain, int type, int protocol)*
- Bind socket to the local address and port number
 - *int bind (int sockfd,
 struct sockaddr *my_addr,
 socklen_t addrlen)*
 - Arguments: sockfd, server address, address length
 - Returns 0 on success, and -1 if an error occurs

Server: Allowing Clients to Wait

34

- Many client requests may arrive
 - Server cannot handle them all at the same time
 - Server could reject the requests, or let them wait
- Define how many connections can be pending
 - ***int listen(int sockfd, int backlog)***
 - Arguments: socket descriptor and acceptable backlog
 - Returns a 0 on success, and -1 on error
- What if too many clients arrive?
 - Some requests don't get through
 - The Internet makes no promises...
 - And the client can always try again

Server: Accepting Client Connection

35

- Now all the server can do is wait...
 - Waits for connection request to arrive
 - Blocking until the request arrives
 - And then accepting the new request
- Accept a new connection from a client
 - *int accept(int sockfd,
 struct sockaddr *addr, socketlen_t *addrlen)*
 - Arguments: sockfd, structure that will provide client address and port, and length of the structure
 - Returns descriptor of socket for this new connection

Server: One Request at a Time?

36

- Serializing requests is inefficient
 - ▣ Server can process just one request at a time
 - ▣ All other clients must wait until previous one is done
- May need to time share the server machine
 - ▣ Alternate between servicing different requests
 - Do a little work on one request, then switch when you are waiting for some other resource (e.g., reading file from disk)
 - “Nonblocking I/O”
 - ▣ Or, use a different process/thread for each request
 - Allow OS to share the CPU(s) across processes
 - ▣ Or, some hybrid of these two approaches

Client and Server: Cleaning House

37

- Once the connection is open
 - ▣ Both sides can read and write
 - ▣ Two unidirectional streams of data
 - ▣ In practice, client writes first, and server reads
 - ▣ ... then server writes, and client reads, and so on
- Closing down the connection
 - ▣ Either side can close the connection
 - ▣ ... using the `close()` system call
- What about the data still “in flight”
 - ▣ Data in flight still reaches the other end
 - ▣ So, server can `close()` before client finishes reading

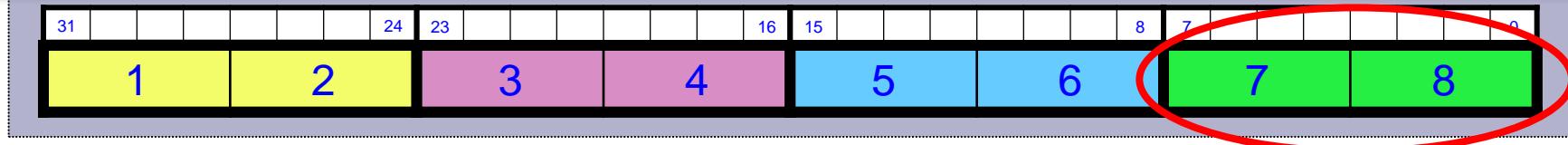
One Annoying Thing: Byte Order

38

- Hosts differ in how they store data
 - E.g., four-byte number (byte3, byte2, byte1, byte0)
- Little endian (“little end comes first”): Intel x86’s
 - Low-order byte stored at the lowest memory location
 - Byte0, byte1, byte2, byte3
- Big endian (“big end comes first”)
 - High-order byte stored at lowest memory location
 - Byte3, byte2, byte1, byte 0
- Makes it more difficult to write portable code
 - Client may be big or little endian machine
 - Server may be big or little endian machine

Endian Example: Where is the Byte?

39



8 bits memory

1000	78
1001	
1002	
1003	

16 bits Memory

1000	+1	+0
1002		78
1004		
1006		

32 bits Memory

1000	+3	+2	+1	+0
1004				78
1008				
100C				

Little-Endian

Big-Endian

+0 +1

+0 +1 +2 +3

1000	78
1001	
1002	
1003	

1000	+0	+1
1002		78
1004		
1006		

1000	+0	+1	+2	+3
1004				78
1008				
100C				

IP is Big Endian

40

- But, what byte order is used “on the wire”
 - That is, what do the network protocol use?
- The Internet Protocols picked one convention
 - IP is big endian (aka “network byte order”)
- Writing portable code require conversion
 - Use htons() and htonl() to convert to network byte order
 - Use ntohs() and ntohl() to convert to host order
- Hides details of what kind of machine you’re on
 - Use the system calls when sending/receiving data structures longer than one byte

Using htonl and htons

41

```
int sockfd = // connected SOCK_STREAM
u_int32_t my_val = 1234;
u_int16_t my_xtra = 16;

u_short bufsize = sizeof (struct data_t);
char *buf = New char[bufsize];
bzero (buf, bufsize);

struct data_t *dat = (struct data_t *) buf;
dat->value = htonl (my_val);
dat->xtra = htons (my_xtra);

int rc = write (sockfd, buf, bufsize);
```

Implementing Network Software

Protocol Implementation Issues

Protocol Implementation Issues

43

- A high level protocol interacts with a low-level protocol
 - Example: TCP needs an interface to send outgoing messages to IP
 - IP needs to be able to deliver incoming messages to TCP
- Process Model
 - Most operating systems provide an abstraction called a *process*, or alternatively, a *thread*
 - Each process runs largely independently of other processes
 - The OS is responsible for making sure that resources
 - Such as address space and CPU cycles, are allocated to all the current processes

Process Model

44

- The process abstraction makes it fairly straightforward to have a lot of things executing concurrently on one machine
 - Example: each user application might execute in its own process, and various things inside the OS might execute as other processes
- When the OS stops one process from executing on the CPU and starts up another one, we call the change a *context switch*

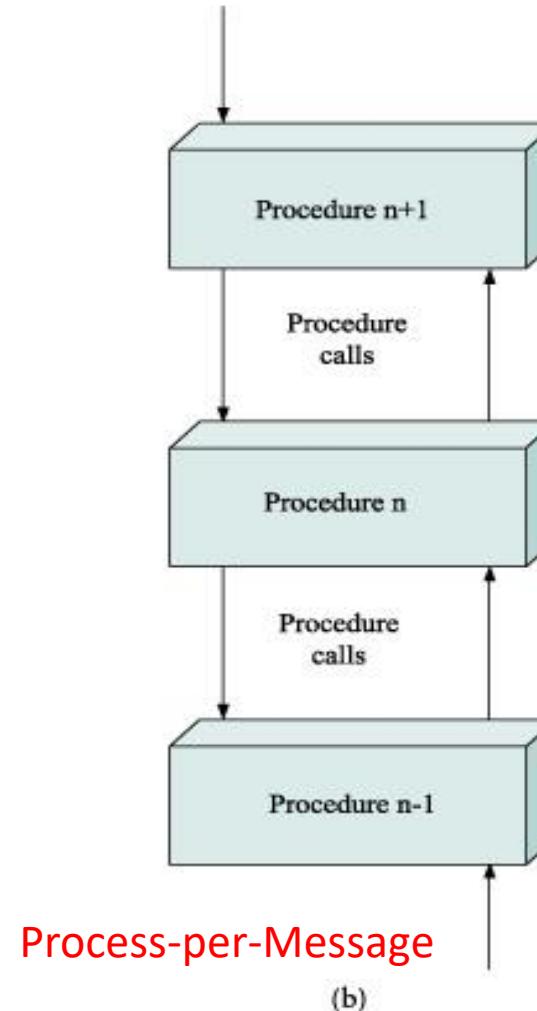
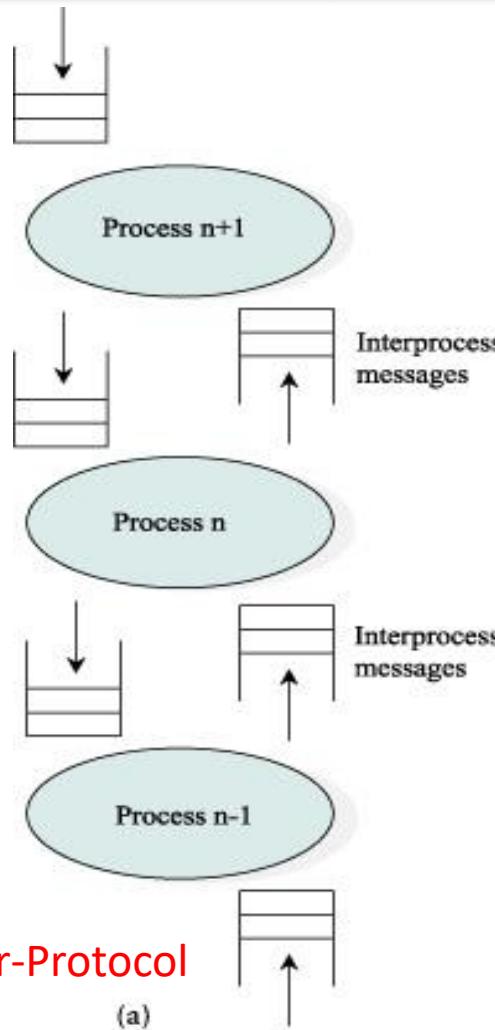
Network Process Models

45

- Two models
- Process-per-protocol model
 - ▣ Each protocol is implemented by a separate process
 - ▣ As a message moves up or down the protocol stack
 - It is passed from one process/protocol to another
 - Example: process that implements protocol i processes the message, then passes it to protocol i-1 and so on
 - ▣ Host OS provides interprocess communication
 - To support one process/protocol passes a message to the next process/protocol

Network Process Models

46



Process-per-Protocol

(a)

Process-per-Message

(b)

Network Process Models

47

- Process-per-message
 - Treats each protocol as a static piece of code and associates the processes with the message
 - When message arrives from the network, the OS dispatches a process that it makes responsible for the message as it moves up the protocol graph
 - At each level, the procedure that implements that protocol is invoked
 - Which eventually results in the procedure for the next protocol being invoked and so on
 - In both directions, the protocol graph is traversed in a sequence of procedure calls

Process-per-Model vs process-per-message

48

- Process-per-Model
 - ▣ A context switch is required at each level of the protocol graph – typically time consuming
 - ▣ Requires a context switch at each level
- process-per-message
 - ▣ The process-per-message model is generally more efficient
 - A procedure call is an order of magnitude more efficient than a context switch on most computers
 - ▣ A procedure call per level

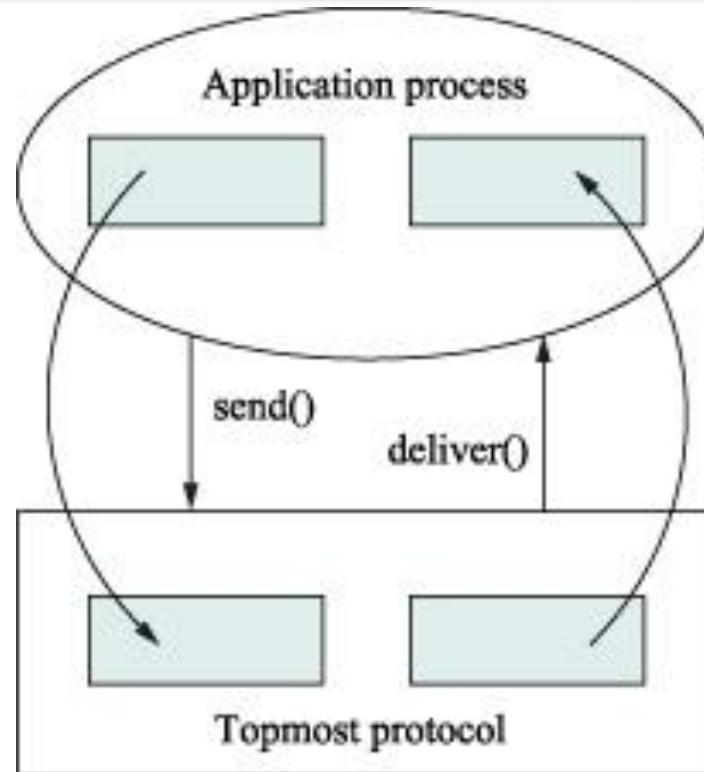
Message Buffers

49

- The application process provides the buffer that contains the outbound message
 - When calling *send*
- Similarly it provides the buffer into which an incoming message is copied
 - when invoking the *receive* operation
- This forces the topmost protocol to copy the message from the application's buffer into a network buffer, and vice versa

Message Buffers

50



Copying incoming/outgoing messages between application buffer and network buffer

Message Buffers –Drawback

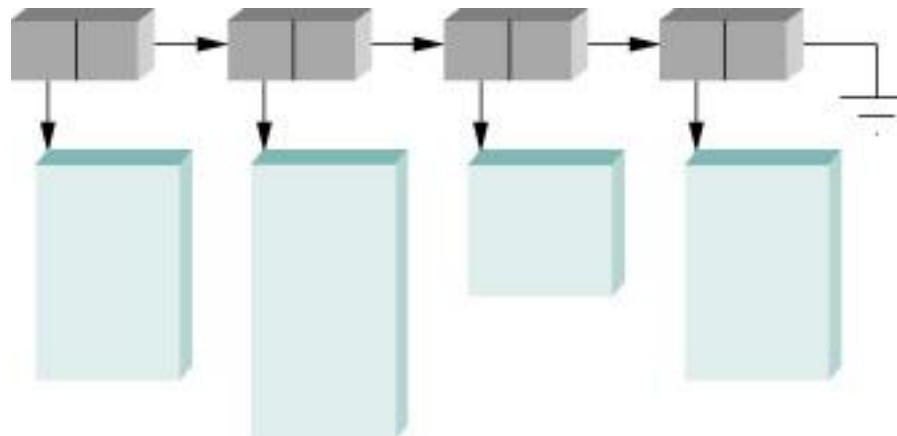
51

- Copying data from one buffer to another is one of the most expensive things a protocol implementation can do
 - Memory speed is slower when compared to processors
- Most of networks defines an abstract data type for messages that is shared by all protocols in the protocol graph
- This abstraction permits
 - Message to be passed up and down the protocol graph without copying
 - It also provides a copy free ways of manipulation of messages
 - Such as adding and stripping headers
 - Fragmenting large messages into a set of small messages
 - Reassembling a collection of small messages into a single large message

Message Buffers

52

- The exact form of message abstraction differs from OS to OS
- In generally involves a linked list of pointers to message buffers



Example message data Structure

PERFORMANCE

25 September
2023

Dr Noor Mohammad Sk

Performance

54

- The effectiveness of computations distributed over the network often depends directly on the efficiency with which the network delivers the computation's data
- Network performance is measured in two fundamental ways
 - Bandwidth (throughput)
 - Latency (delay)

Performance

Bandwidth and Latency

Bandwidth and Latency

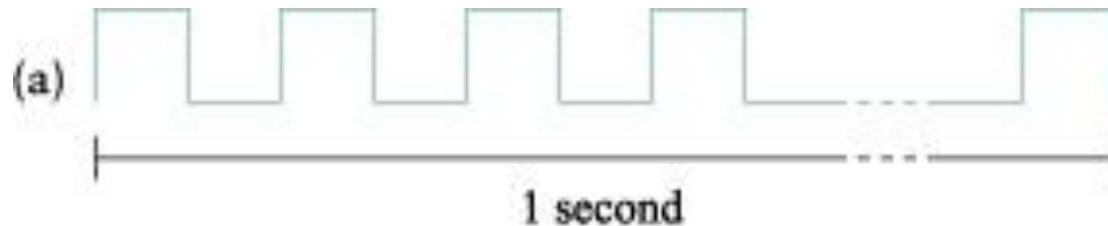
56

- Bandwidth: the number of bits that can be transmitted over the network in a certain period of time
 - ▣ Example: 10 millions bits per seconds
 - It is able to deliver 10 million bits every second
 - ▣ On 10 Mbps network, it takes 0.1 microsecond (μs) to transmit each bit
- Latency: corresponding to how long it takes a message to travel from one end of a network to the other
 - ▣ Latency is measured strictly in terms of time

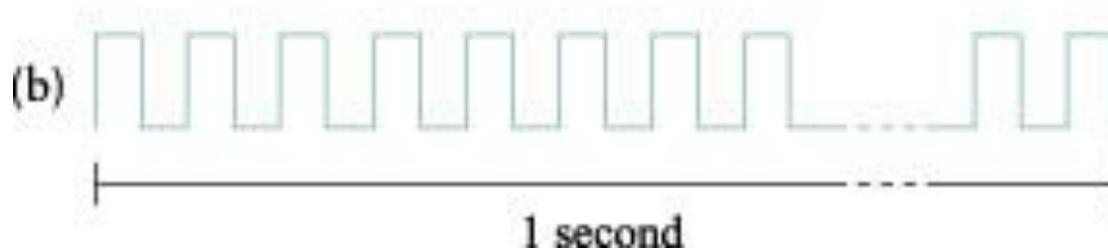
Bandwidth

57

- Bits transmitted at a particular bandwidth can be regarded as having some width



Bits transmitted at 1mbps (each bit 1us wide)



Bits transmitted at 2mbps (each bit 0.5us wide)

Latency

58

- **Round-trip time (RTT)**: the time it takes to send a message from one end of a network to the other and back
- Latency depends of three parameters
 - Propagation Delay
 - **Transmit Delay**: the amount of time it take to transmit a unit of data
 - **Queue Delays**: since packets switches generally need to store packets for some time before forwarding them on an outbound link
- *Latency* = *Propagation* + *Transmit* + *Queue*
- *Propagation* = *Distance*/*SpeedOfLight*
- *Transmit* = *Size*/*Bandwidth*

Latency

59

- **Distance**: Length of the wire over which the data will travel
- **SpeedOfLight**: effective speed of light over that wire
- **Size**: size of the packet
- **Bandwidth**: bandwidth at which the packet is transmitted

Performance

60

- Bandwidth and latency combine to define the performance characteristics of a given link or channel
- Latency dominates bandwidth for some applications
 - ▣ A client that sends a 1-byte message to a server and receives a 1-byte message in return is latency bound
 - ▣ The application performs much differently on different channels
 - A transcontinental channel with a 100ms RTT
 - An across the room channel with a 1ms RTT
 - ▣ The channel is 1Mbps or 100Mbps is relatively insignificant
 - To transmit a byte on a above will take 8us and 0.08us time.

Performance

61

- A digital library program that is being asked to fetch a 25MB image
- The more bandwidth that is available, the faster it will be able to return the image to the user
- The bandwidth of the channel dominates performance

Delay x Bandwidth Product

62

- A channel a pair of processes as a hollow pipe
 - The latency corresponds to the length of the pipe
 - Bandwidth gives the diameter of the pipe
 - The delay x bandwidth product gives the volume of the pipe
 - For example a transcontinental channel with a one-way latency of 50ms and a bandwidth of 45 Mbps is able to hold
 - $50 \times 10^{-3} \text{ sec} \times 45 \times 10^6 \text{ bits/sec}$
 - $2.25 \times 10^6 \text{ bits}$



Delay x Bandwidth Product

63

Link Type	Bandwidth (typical)	Distance (typical)	Round-trip Delay	Delay x BW
Dail – up	56 Kbps	10Km	87 μ s	5bits
Wireless LAN	54 Mbps	50m	0.33 μ s	18 bits
Satellite	45 Mbps	35,000 Km	230 ms	10Mb
Cross-country fiber	10Gbps	4,000 km	40 ms	400 Mb

- The delay x bandwidth product is important to know when constructing high-performance networks
- Because it corresponds to how many bits the sender must transmit before the first bit arrives at the receiver

Delay x Bandwidth Product

64

- Relative importance of bandwidth and latency depends on application
 - For large file transfer, bandwidth is critical
 - For small messages (HTTP, NFS, etc.), latency is critical
 - Variance in latency (jitter) can also affect some applications (*e.g.*, audio/video conferencing)

Delay x Bandwidth Product

65

- How many bits the sender must transmit before the first bit arrives at the receiver if the sender keeps the pipe full
- Takes another one-way latency to receive a response from the receiver
- If the sender does not fill the pipe—send a whole delay × bandwidth product's worth of data before it stops to wait for a signal—the sender will not fully utilize the network

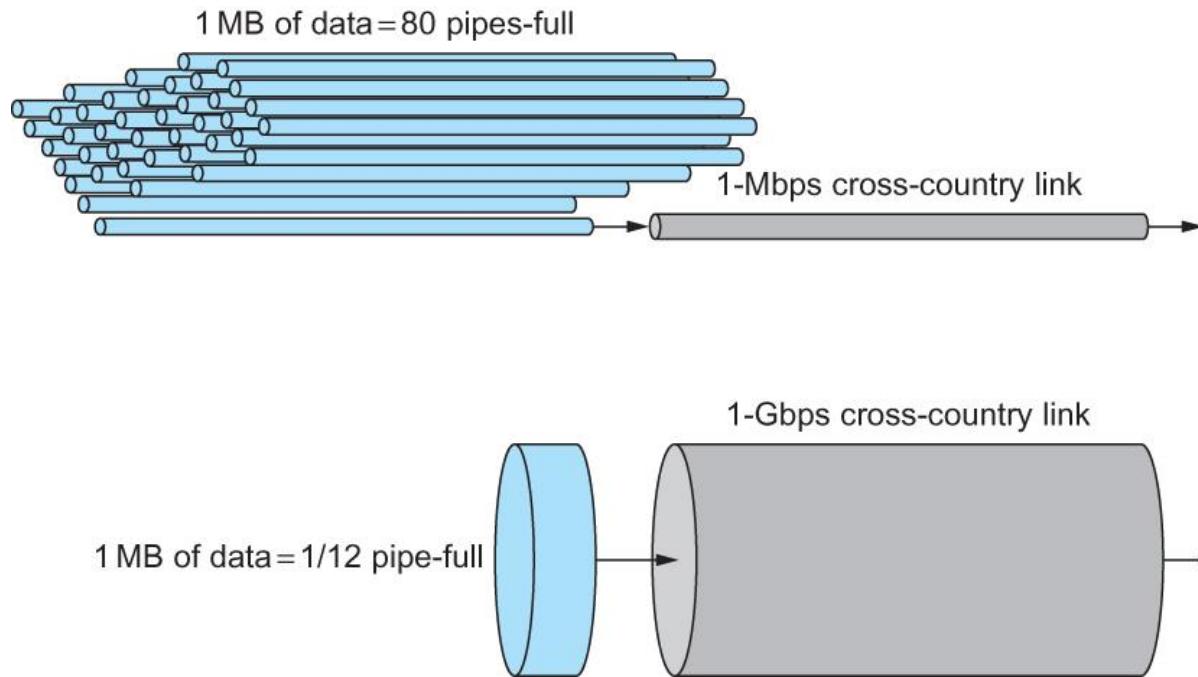
Delay x Bandwidth Product

66

- Infinite bandwidth
 - RTT dominates
 - $\text{Throughput} = \text{TransferSize} / \text{TransferTime}$
 - $\text{TransferTime} = \text{RTT} + (1/\text{Bandwidth}) \times \text{TransferSize}$
- Its all relative
 - 1-MB file to 1-Gbps link looks like a 1-KB packet to 1-Mbps link

Relationship between Bandwidth and Latency

67



A 1-MB file would fill the 1-Mbps link 80 times,
but only fill the 1-Gbps link 1/12 of one time

Throughput of Network

68

- $\text{Throughput} = \text{TransferSize} / \text{TransferTime}$
- $\text{TransferTime} = \text{RTT} + (1/\text{Bandwidth}) \times \text{TransferSize}$

Summary

69

- We have identified what we expect from a computer network
- We have defined a layered architecture for computer network that will serve as a blueprint for our design
- We have discussed the socket interface which will be used by applications for invoking the services of the network subsystem
- We have discussed two performance metrics using which we can analyze the performance of computer networks

THANK YOU!!

25 September
2023

Noor Mohammad Sk

DIRECT LINK NETWORKS

25 September
2023

Dr Noor Mohammad Sk

Outline

2

- Hardware Building Blocks
- Encoding
- Framing
- Error Detection
- Reliable Transmission
- Ethernet
- Rings
- Wireless

Objectives

3

- Exploring different communication medium over which we can send data
- Understanding the issue of encoding bits onto transmission medium so that they can be understood by the receiving end
- Discussing the matter of delineating the sequence of bits transmitted over the link into complete messages that can be delivered to the end node
- Discussing different technique to detect transmission errors and take the appropriate action

Objectives

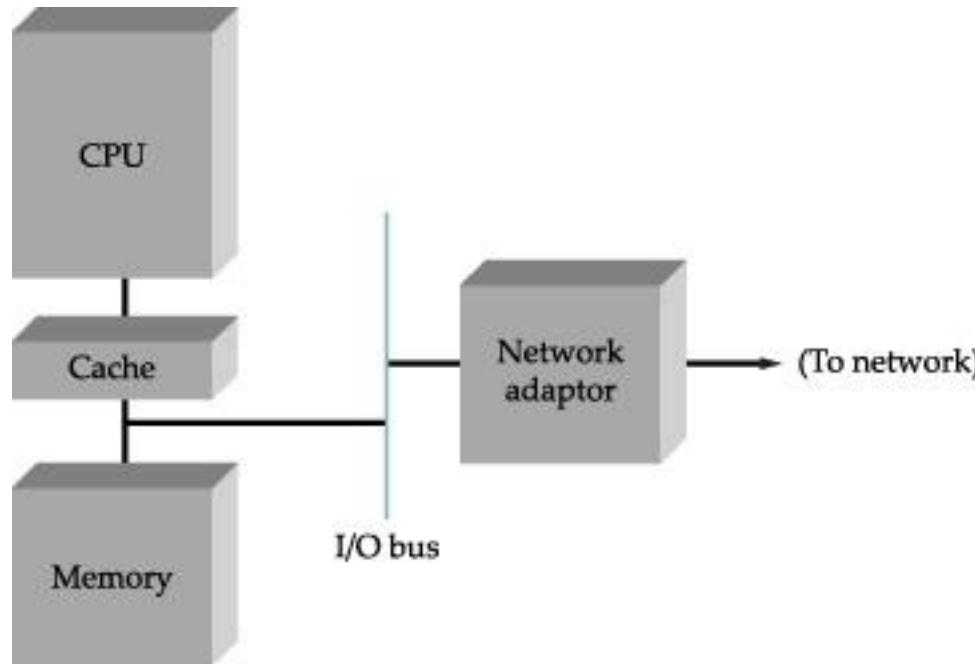
4

- Discussing the issue of making the links reliable in spite of transmission problems
- Introducing Media Access Control Problem
- Introducing Carrier Sense Multiple Access (CSMA) networks
- Introducing Wireless Networks with different available technologies and protocol

Nodes

5

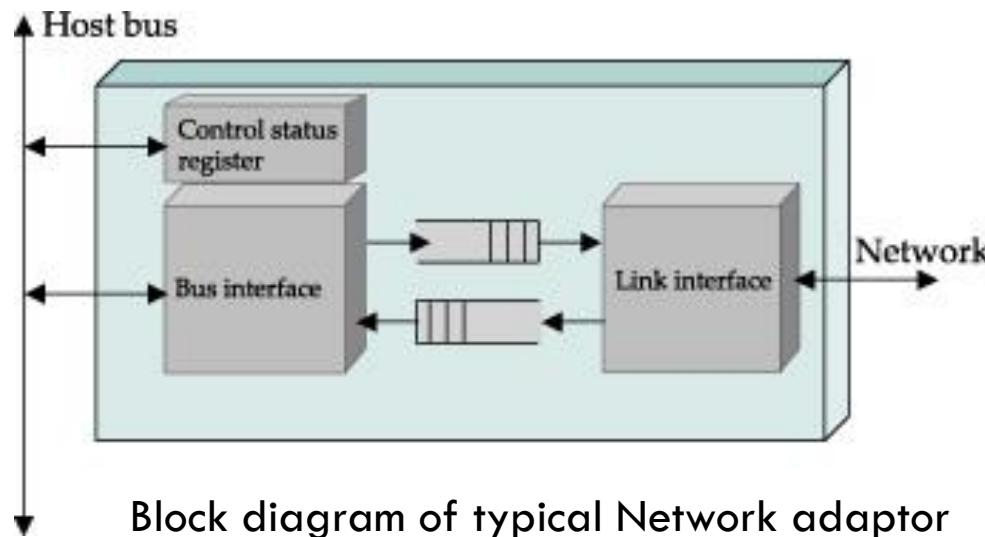
- Example of workstation architecture



Nodes

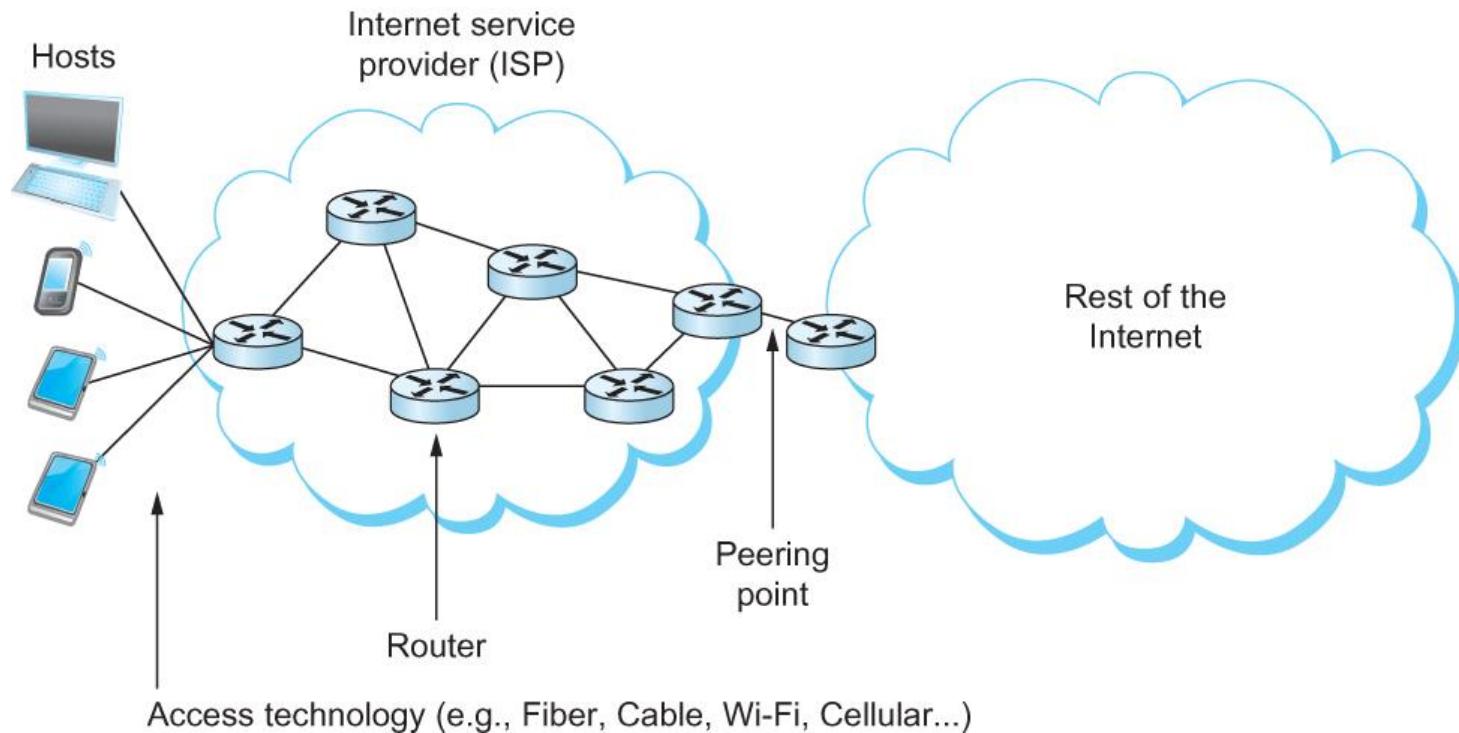
6

- CSR is readable and writable from CPU
- Software on the host
 - Writes to the CSR to instruct it to transmit and/or receive data
 - Reads from the CSR to learn the current status of the adaptor



Perspectives on Connecting

7



An end-user's view of the Internet

Link Capacity and Shannon-Hartley Theorem

8

- Gives the upper bound to the capacity of a link in terms of bits per second (bps) as a function of signal-to-noise ratio of the link measured in decibels (dB).
- $C = B \log_2(1+S/N)$
 - B is Bandwidth, frequency support range 300 to 3300 Hz.
 - Where $B = 3300 - 300 = 3000\text{Hz}$, S is the signal power, N the average noise.
 - The signal to noise ratio (S/N) is measured in decibels is related to $\text{dB} = 10 \times \log_{10}(S/N)$. If there is 30dB of noise then $S/N = 1000$.
 - Now $C = 3000 \times \log_2(1001) = 30\text{kbps}$.

Links

9

- All practical links rely on some sort of electromagnetic radiation propagating through a medium or, in some cases, through free space
- One way to characterize links, then, is by the medium they use
 - Typically copper wire in some form (as in Digital Subscriber Line (DSL) and coaxial cable),
 - Optical fiber (as in both commercial fiber-to-the home services and many long-distance links in the Internet's backbone), or
 - Air/free space (for wireless links)

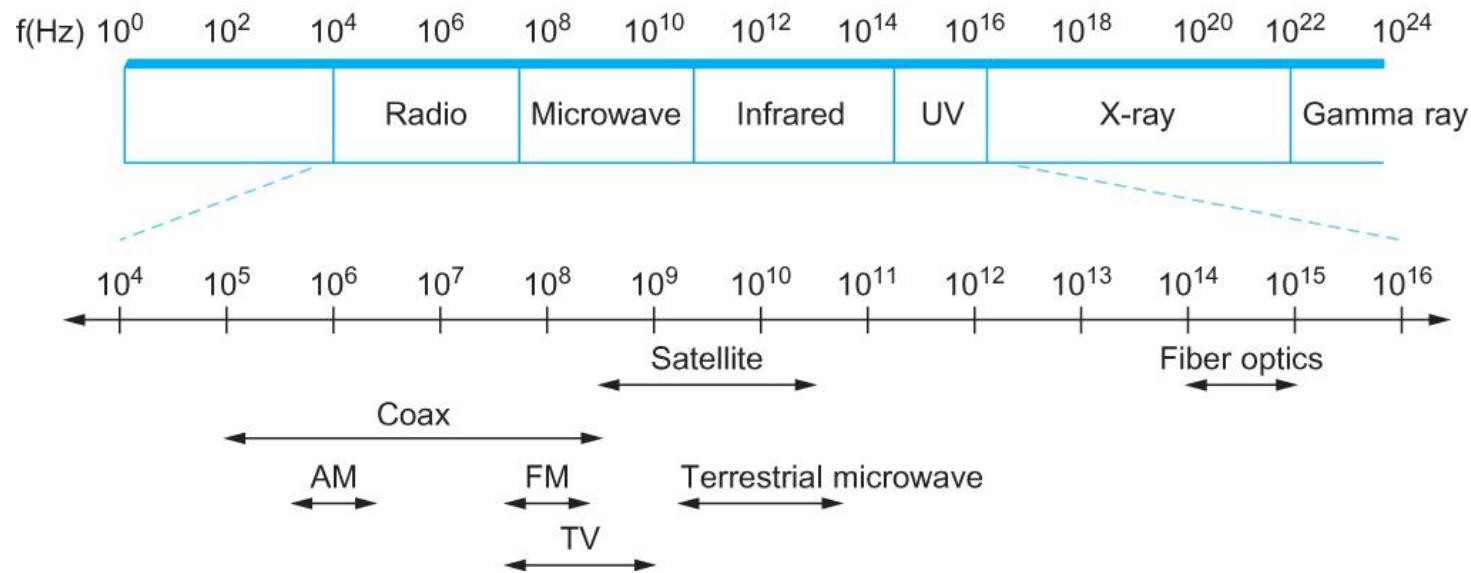
Links

10

- Another important link characteristic is the *frequency*
 - Measured in hertz, with which the electromagnetic waves oscillate
- Distance between the adjacent pair of maxima or minima of a wave measured in meters is called *wavelength*
 - Speed of light divided by frequency gives the wavelength.
 - Frequency on a copper cable range from 300Hz to 3300Hz; Wavelength for 300Hz wave through copper is speed of light on a copper / frequency
 - $(2/3) \times 3 \times 10^8 / 300 = 667 \times 10^3$ meters.
- Placing binary data on a signal is called *encoding*.
- Modulation involves modifying the signals in terms of their frequency, amplitude, and phase.

Links

11



Electromagnetic spectrum

Cables

12

Cable	Typical Bandwidth	Distance
Category 5 Twisted Pair	10 – 100 Mbps	100m
Thin – net coax	10 – 100 Mbps	200m
Thick – net coax	10 – 100 Mbps	500m
Multimode Fiber	100Mbps	2Km
Single – mode fiber	0.1 – 10 Gbps	40Km

Common types of cables and fibers available for Local Links

Links

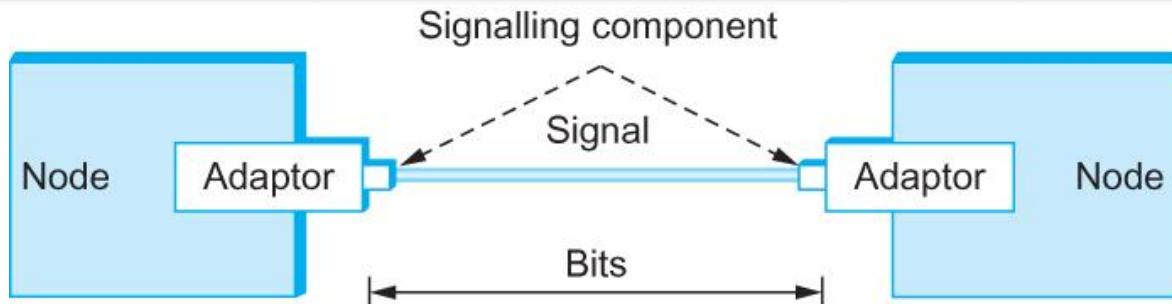
13

Service	Bandwidth (typical)
Dial-up	28–56 kbps
ISDN	64–128 kbps
DSL	128 kbps–100 Mbps
CATV (cable TV)	1–40 Mbps
FTTH (fibre to the home)	50 Mbps–1 Gbps

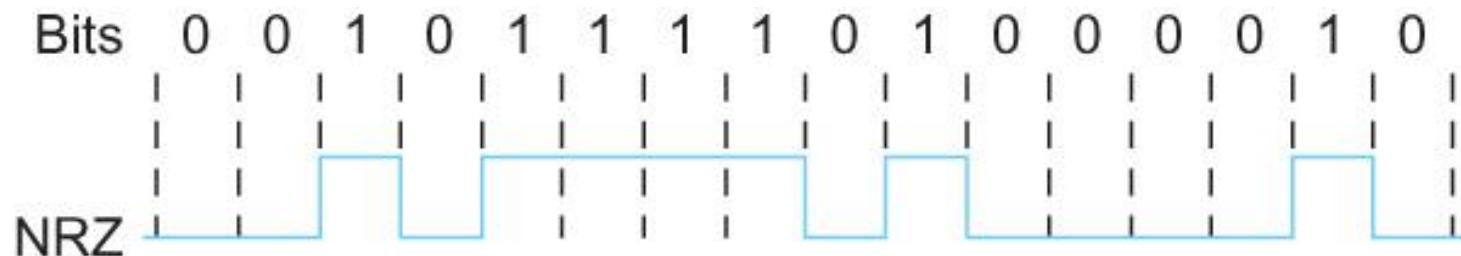
Common services available to connect your home

Encoding

14



Signals travel between signaling components; bits flow between adaptors



NRZ encoding of a bit stream

Encoding

15

- Problem with NRZ (*nonreturn to zero*)
 - Baseline wander
 - The receiver keeps an average of the signals it has seen so far
 - Uses the average to distinguish between low and high signal
 - When a signal is significantly low than the average, it is 0, else it is 1
 - Too many consecutive 0's and 1's cause this average to change, making it difficult to detect

Encoding

16

- Problem with NRZ
 - ▣ Clock recovery
 - Frequent transition from high to low or vice versa are necessary to enable clock recovery
 - Both the sending and decoding process is driven by a clock
 - Every clock cycle, the sender transmits a bit and the receiver recovers a bit
 - The sender and receiver have to be precisely synchronized

Encoding

17

- NRZI
 - Non Return to Zero Inverted
 - Sender makes a transition from the **current signal** to **encode 1** and stay at the current signal to **encode 0**
 - Solves for consecutive 1's

Encoding – Manchester

18

- Manchester encoding
 - Merging the clock with signal by transmitting Ex-OR of the *NRZ encoded data* and the *clock*
 - Clock is an internal signal that alternates from low to high, a low/high pair is considered as one clock cycle
 - In Manchester encoding
 - 0: **low → high** transition
 - 1: **high → low** transition

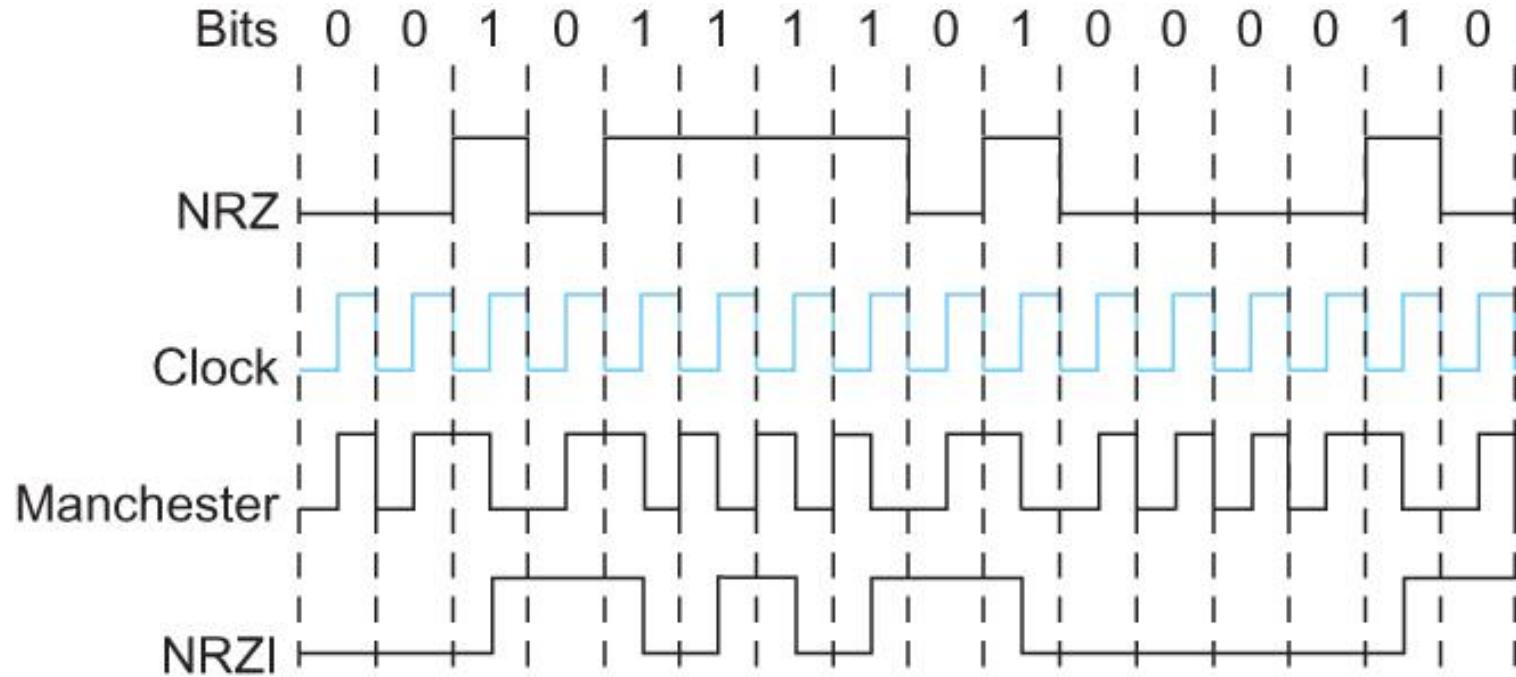
Encoding - Manchester

19

- Problem with Manchester encoding
 - Doubles the rate at which the signal transitions are made on the link
 - Which means the receiver has half of the time to detect each pulse of the signal
 - The rate at which the signal changes is called the link's baud rate
 - In Manchester the bit rate is half the baud rate

Encoding

20



Different encoding strategies

Encoding

21

- 4B/5B encoding
 - ▣ Insert extra bits into bit stream so as to break up the long sequence of 0's and 1's
 - ▣ Every 4-bits of actual data are encoded in a 5-bit code that is transmitted to the receiver
 - ▣ 5-bit codes are selected in such a way that each one has no more than one leading 0(zero) and no more than two trailing 0's.
 - ▣ No pair of 5-bit codes results in more than three consecutive 0's

4B/5B Encoding

22

4 – bit Data Symbol	5 – bit Code
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

Encoding

23

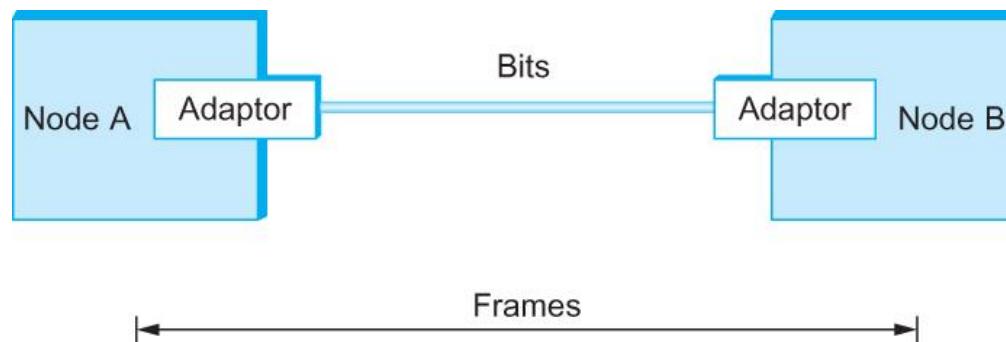
- 4B/5B encoding

- 16 left
- 11111 – when the line is idle
- 00000 – when the line is dead
- 00100 – to mean halt
- 13 left :
 - 7 invalid (no more than One leading ‘0’, two tailing ‘0s’),
 - 6 for various control symbols/signals

Framing

24

- We are focusing on packet-switched networks, which means that blocks of data (called *frames* at this level), not bit streams, are exchanged between nodes.
- It is the network adaptor that enables the nodes to exchange frames.



Bits flow between adaptors, frames between hosts

Framing

25

- When node A wishes to transmit a frame to node B, it tells its adaptor to transmit a frame from the node's memory. This results in a sequence of bits being sent over the link.
- The adaptor on node B then collects together the sequence of bits arriving on the link and deposits the corresponding frame in B's memory.
- Recognizing exactly what set of bits constitute a frame—that is, determining where the frame begins and ends—is the central challenge faced by the adaptor

Framing

26

- Byte-oriented Protocols (PPP)
 - To view each frame as a collection of bytes (characters) rather than bits
 - BISYNC (Binary Synchronous Communication) Protocol
 - Developed by IBM (late 1960)
 - DDCMP (Digital Data Communication Protocol)
 - Used in DECNet

Framing

27

- BISYNC – sentinel approach
 - Frames transmitted beginning with leftmost field
 - Beginning of a frame is denoted by sending a special SYN (synchronize) character
 - Data portion of the frame is contained between special sentinel character STX (start of text) and ETX (end of text)
 - SOH : Start of Header
 - DLE : Data Link Escape
 - CRC: Cyclic Redundancy Check

Framing

28

- Header: Link level reliable delivery algorithm



BISYNC Frame Format

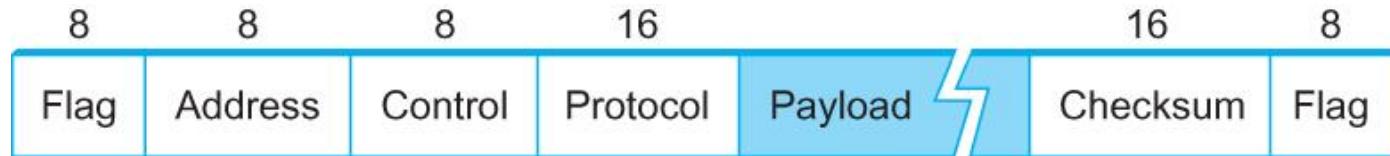
Framing

29

- Recent PPP which is commonly run over Internet links uses sentinel approach
 - Special start of text character denoted as Flag
 - 0 1 1 1 1 1 0
 - Address, control: default numbers
 - Protocol for demux: IP / IPX
 - Payload: negotiated (1500 bytes)
 - Checksum: for error detection

Framing

30



PPP Frame Format

Framing

31

- Byte-counting approach
 - DDCMP (Digital Data Communication Message Protocol)
 - *count* : how many bytes are contained in the frame body
 - If *count* is corrupted
 - Framing error



DDCMP Frame Format

Framing

32

- Bit-oriented Protocol
 - ▣ HDLC : High Level Data Link Control
 - Beginning and Ending Sequences

0 1 1 1 1 1 0



HDLC Frame Format

Framing

33

- HDLC Protocol
 - On the sending side, any time five consecutive 1's have been transmitted from the body of the message (i.e. excluding when the sender is trying to send the distinguished 01111110 sequence)
 - The sender inserts 0 before transmitting the next bit

Framing

34

- HDLC Protocol

- On the receiving side

- 5 consecutive 1's

- Next bit 0 : Stuffed, so discard it

- 1 : Either End of the frame marker

- Or Error has been introduced in the bitstream

- Look at the next bit

- If 0 (01111110) → End of the frame marker

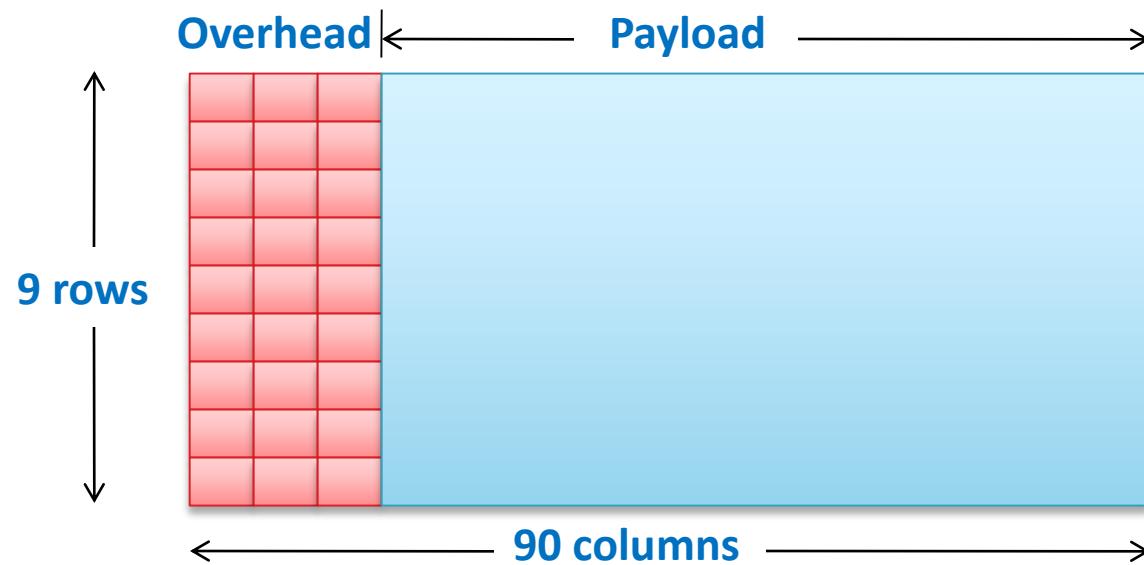
- If 1 (01111111) → Error, discard the whole frame

- The receiver needs to wait for next
01111110 before it can start
receiving again

Clock-Based Framing (SONET)

35

- Synchronous Optical Network (SONET) Standard is used for long distance transmission of data over optical network.
- It supports multiplexing of several low speed links into one high speed links
- An STS – 1 frame is used in this method



Clock-Based Framing (SONET)

36

- It is arranged as nine rows of 90 bytes each, and the first 3 bytes of each row are overhead, with the rest being available for data
- The first 2 bytes of the frame contain a special bit pattern, and it is these bytes that enable the receiver to determine where the frame starts
- The receiver looks for the special bit pattern consistently, once in every 810 bytes, since each frame is $9 \times 90 = 810$ bytes long

Clock-Based Framing (SONET)

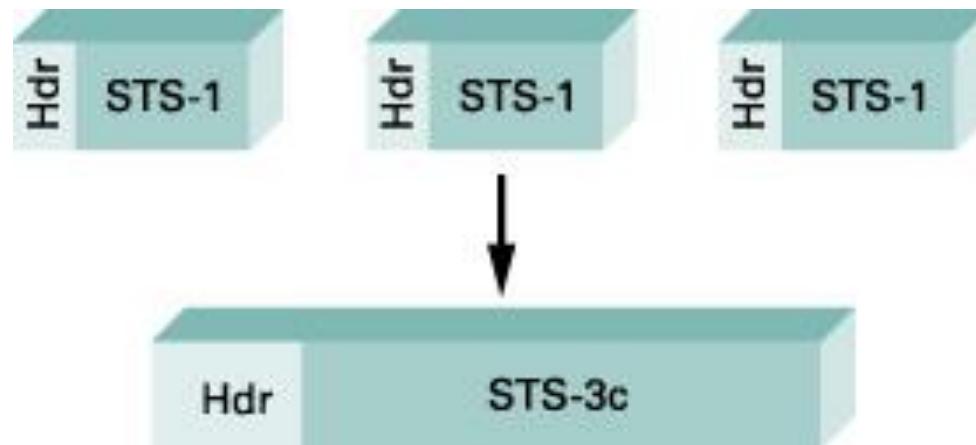
37

- The STS – N frame can be thought of as consisting of N STS – 1 frames, where the bytes from these frames are interleaved;
 - ▣ That is, a byte from the first frame is transmitted, then a byte from the second frame is transmitted, and so on
- Payload from these STS – 1 frames can be linked together to form a large STS – N payload, such a link is denoted STS – Nc. One of the bit in overhead is used for this purpose

Clock-Based Framing (SONET)

38

- Three STS – 1 frame multiplexed onto one STS – 3c frame



Error Detection

39

- Bit errors are introduced into frames
 - ▣ Because of electrical interference and thermal noises
- Detecting Error
- Correction Error
- Two approaches when the recipient detects an error
 - ▣ Notify the sender that the message was corrupted, so the sender can send again.
 - If the error is rare, then the retransmitted message will be error-free
 - ▣ Using some error detection and correction algorithm, the receiver reconstructs the message

Error Detection

40

- Common technique for detecting transmission error
 - CRC (Cyclic Redundancy Check)
 - Used in HDLC, DDCMP, CSMA/CD, Token Ring
 - Other approaches
 - Two Dimensional Parity (BISYNC)
 - Checksum (IP)

Error Detection

41

- Basic Idea of Error Detection
 - To add redundant information to a frame that can be used to determine if errors have been introduced
 - Imagine (Extreme Case)
 - Transmitting two complete copies of data
 - Identical → No error
 - Differ → Error
 - Poor Scheme ???
 - n bit message, n bit redundant information
 - Error can go undetected
 - In general, we can provide strong error detection technique
 - k redundant bits, n bits message, $k \ll n$
 - In Ethernet, a frame carrying up to 12,000 bits of data requires only 32-bit CRC

Error Detection

42

- Extra bits are redundant
 - They add no new information to the message
 - Derived from the original message using some algorithm
 - Both the sender and receiver know the algorithm

Sender

m	r
---	---

Receiver

m	r
---	---

Receiver computes r using m

If they match, no error

Two Dimensional Parity

43

- Two-dimensional parity is exactly what the name suggests
- It is based on “simple” (one-dimensional) parity, which usually involves adding one extra bit to a 7-bit code to balance the number of 1s in the byte. For example,
 - ▣ Odd parity sets the eighth bit to 1 if needed to give an odd number of 1s in the byte, and
 - ▣ Even parity sets the eighth bit to 1 if needed to give an even number of 1s in the byte

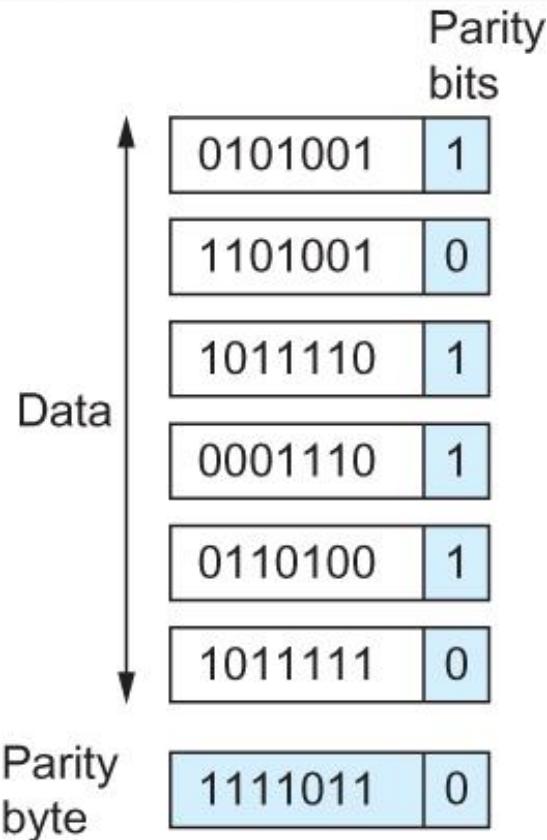
Two-dimensional parity

44

- Two-dimensional parity does a similar calculation for each bit position across each of the bytes contained in the frame
- This results in an extra parity byte for the entire frame, in addition to a parity bit for each byte
- Two-dimensional parity catches all 1-, 2-, and 3-bit errors and most 4-bit errors

Two-dimensional parity

45



Two Dimensional Parity

Internet Checksum Algorithm

46

- Not used at the link level
- *Add up all the words* that are transmitted and then transmit the result of that *sum*
 - The result is called the checksum
- The receiver performs the same calculation on the received data and compares the result with the received checksum
- If any transmitted data, including the checksum itself, is corrupted, then the results will not match, so the receiver knows that an error occurred

Internet Checksum Algorithm

47

- Consider the data being check-summed as a sequence of 16-bit integers.
- Add them together using 16-bit ones complement arithmetic (explained next slide) and then take the ones complement of the result.
- That 16-bit number is the checksum

Internet Checksum Algorithm

48

- In ones complement arithmetic, a negative integer $-x$ is represented as the complement of x ;
 - Each bit of x is inverted.
- When adding numbers in ones complement arithmetic, a carryout from the most significant bit needs to be added to the result.

Internet Checksum Algorithm

49

- Consider, for example, the addition of -5 and -3 in ones complement arithmetic on 4-bit integers
 - $+5$ is 0101, so -5 is 1010; $+3$ is 0011, so -3 is 1100
- If we add 1010 and 1100 ignoring the carry, we get 0110
- In ones complement arithmetic, the fact that this operation caused a carry from the most significant bit causes us to increment the result, giving 0111, which is the ones complement representation of -8 (obtained by inverting the bits in 1000), as we would expect

Evaluation of checksums

50

- Data overhead - 16 or 32 bits
- Computational overhead - simple additions
- Undetected errors - some periodic reversal of bits
(e.g., reversing one bit in each of four data items)

Example failure of checksums

51

Data Item (Binary)	Checksum Value	Data Item (Binary)	Checksum Value
0 0 0 1	1	0 0 1 1	3
0 0 1 0	2	0 0 0 0	0
0 0 1 1	3	0 0 0 1	1
0 0 0 1	1	0 0 1 1	3
Totals	7	Totals	7

Cyclic redundancy checks (CRCs)

52

- Detects more errors than checksums and only requires simple hardware
- Can be analyzed mathematically but is best presented in terms of the hardware design

Cyclic Redundancy Check (CRC)

53

- Reduce the number of extra bits and maximize protection
- Given a bit string 110001 we can associate a polynomial on a single variable x for it.

$$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^5 + x^4 + 1 \text{ and the degree is 5.}$$

A k -bit frame has a maximum degree of $k-1$

- Let $M(x)$ be a message polynomial and $C(x)$ be a generator polynomial.

Cyclic Redundancy Check (CRC)

54

- Let $M(x)/C(x)$ leave a remainder of 0.
- When $M(x)$ is sent and $M'(x)$ is received we have
$$M'(x) = M(x) + E(x)$$
- The receiver computes $M'(x)/C(x)$ and if the remainder is *nonzero*, then an error has occurred.
- The only thing the sender and the receiver should know is $C(x)$.

Cyclic Redundancy Check (CRC)

55

Polynomial Arithmetic Modulo 2

- Any polynomial $B(x)$ can be divided by a divisor polynomial $C(x)$ if $B(x)$ is of higher degree than $C(x)$.
- Any polynomial $B(x)$ can be divided once by a divisor polynomial $C(x)$ if $B(x)$ is of the same degree as $C(x)$.
- The remainder obtained when $B(x)$ is divided by $C(x)$ is obtained by subtracting $C(x)$ from $B(x)$.
- To subtract $C(x)$ from $B(x)$, we simply perform the exclusive-OR (XOR) operation on each pair of matching coefficients.

Cyclic Redundancy Check (CRC)

56

- Let $M(x)$ be a frame with m bits and let the generator polynomial have less than m bits say equal to r .
- Let r be the degree of $C(x)$. Append r zero bits to the low-order end of the frame, so it now contains $m+r$ bits and corresponds to the polynomial $x^r M(x)$.

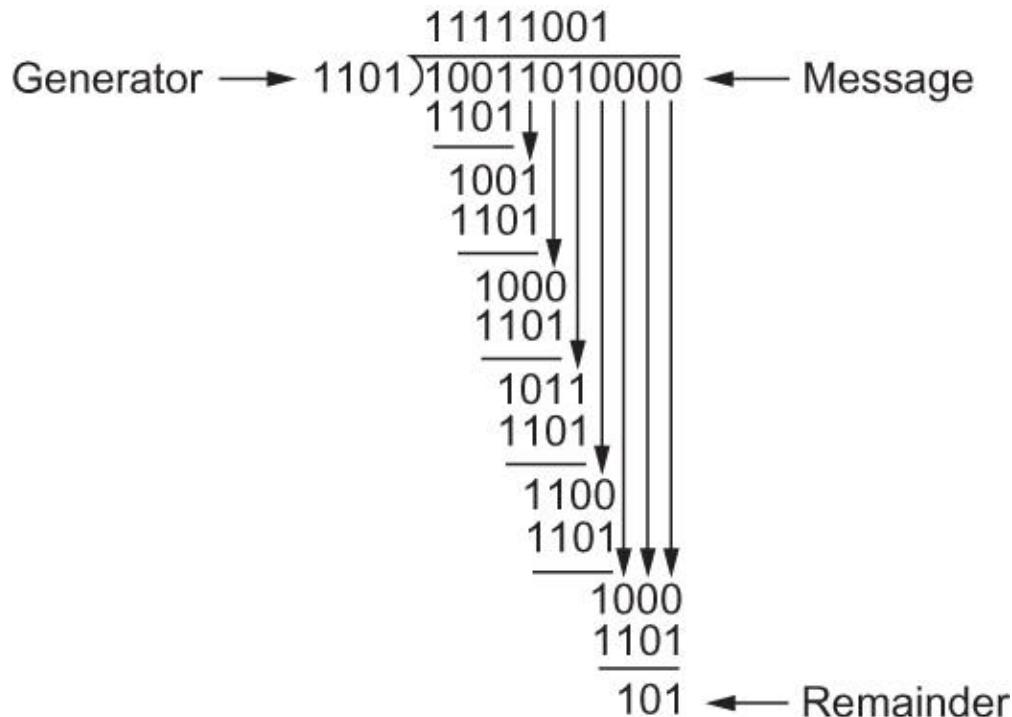
Cyclic Redundancy Check (CRC)

57

- Divide the bit string corresponding to $x^r M(x)$ by the bit string corresponding to $C(x)$ using modulo 2 division.
- Subtract the remainder (which is always r or fewer bits) from the string corresponding to $x^r M(x)$ using modulo 2 subtraction (addition and subtraction are the same in modulo 2).
- The result is the check summed frame to be transmitted. Call it polynomial $M'(x)$.

Cyclic Redundancy Check (CRC)

58



CRC Calculation using Polynomial Long Division

Cyclic Redundancy Check (CRC)

59

□ Properties of Generator Polynomial

- Let $P(x)$ represent what the sender sent and $P(x) + E(x)$ is the received string. A 1 in $E(x)$ represents that in the corresponding position in $P(x)$ the message the bit is flipped.
- We know that $P(x)/C(x)$ leaves a remainder of 0, but if $E(x)/C(x)$ leaves a remainder of 0, then either $E(x) = 0$ or $C(x)$ is factor of $E(x)$.
- When $C(x)$ is a factor of $E(x)$ we have problem; errors go unnoticed.
- If there is a single bit error then $E(x) = x^i$, where i determines the bit in error. If $C(x)$ contains two or more terms it will never divide $E(x)$, so all single bit errors will be detected.

Cyclic Redundancy Check (CRC)

60

□ Properties of Generator Polynomial

- In general, it is possible to prove that the following types of errors can be detected by a $C(x)$ with the stated properties
 - All single-bit errors, as long as the x^k and x^0 terms have nonzero coefficients.
 - All double-bit errors, as long as $C(x)$ has a factor with at least three terms.
 - Any odd number of errors, as long as $C(x)$ contains the factor $(x+1)$.
 - Any “burst” error (i.e., sequence of consecutive error bits) for which the length of the burst is less than k bits. (Most burst errors of larger than k bits can also be detected.)

Cyclic Redundancy Check (CRC)

61

- Six generator polynomials that have become international standards are:
 - $\text{CRC-8} = x^8 + x^2 + x + 1$
 - $\text{CRC-10} = x^{10} + x^9 + x^5 + x^4 + x + 1$
 - $\text{CRC-12} = x^{12} + x^{11} + x^3 + x^2 + x + 1$
 - $\text{CRC-16} = x^{16} + x^{15} + x^2 + 1$
 - $\text{CRC-CCITT} = x^{16} + x^{12} + x^5 + 1$
 - $\text{CRC-32} = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Reliable Transmission

Reliable Transmission

63

- CRC is used to detect errors.
- Some error codes are strong enough to correct errors.
- The overhead is typically too high.
- Corrupt frames must be discarded.
- A link-level protocol that wants to deliver frames reliably must recover from these discarded frames.
- This is accomplished using a combination of two fundamental mechanisms
 - ▣ Acknowledgements and Timeouts

Reliable Transmission

64

- An *acknowledgement* (ACK for short) is a small control frame that a protocol sends back to its peer saying that it has received the earlier frame.
 - A control frame is a frame with header only (no data).
- The receipt of an *acknowledgement* indicates to the sender of the original frame that its frame was successfully delivered.

Reliable Transmission

65

- If the sender does not receive an *acknowledgment* after a reasonable amount of time, then it retransmits the original frame.
- The action of waiting a reasonable amount of time is called a *timeout*.
- The general strategy of using *acknowledgements* and *timeouts* to implement reliable delivery is sometimes called **Automatic Repeat reQuest (ARQ)**.

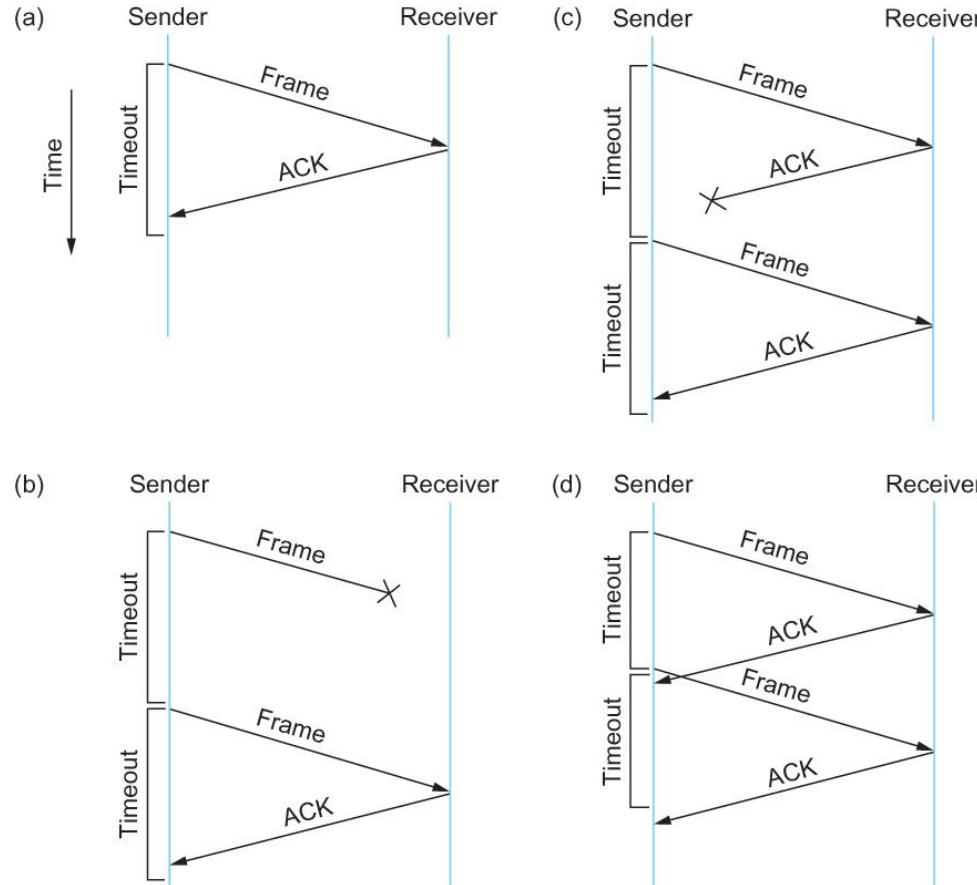
Stop and Wait Protocol

66

- Idea of stop-and-wait protocol is straightforward
- After transmitting one frame, the sender *waits* for an *acknowledgement* before transmitting the *next frame*.
- If the acknowledgement does not arrive after a certain period of time, the sender *times out* and *retransmits* the original frame

Stop and Wait Protocol

67



Timeline showing four different scenarios for the stop-and-wait algorithm.

(a) The ACK is received before the timer expires; (b) the original frame is lost; (c) the ACK is lost; (d) the timeout fires too soon

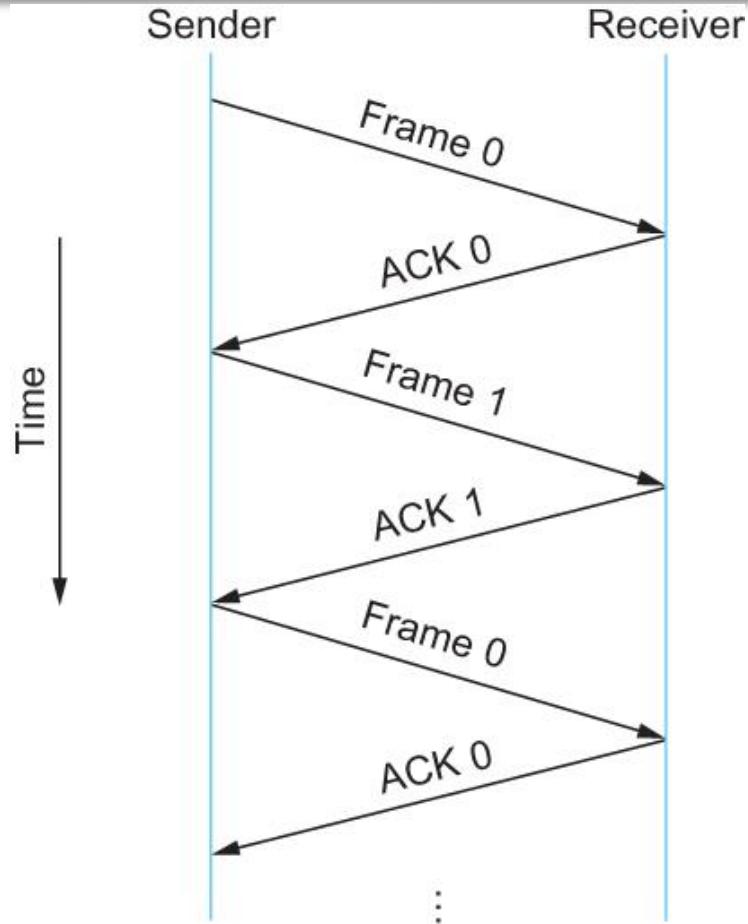
Stop and Wait Protocol

68

- If the acknowledgment is lost or delayed in arriving
 - The sender times out and retransmits the original frame, but the receiver will think that it is the next frame since it has correctly received and acknowledged the first frame
 - As a result, duplicate copies of frames will be delivered
- How to solve this
 - Use 1 bit sequence number (0 or 1)
 - When the sender retransmits frame 0, the receiver can determine that it is seeing a second copy of frame 0 rather than the first copy of frame 1 and therefore can ignore it (the receiver still acknowledges it, in case the first acknowledgement was lost)

Stop and Wait Protocol

69



Timeline for stop-and-wait with 1-bit sequence number

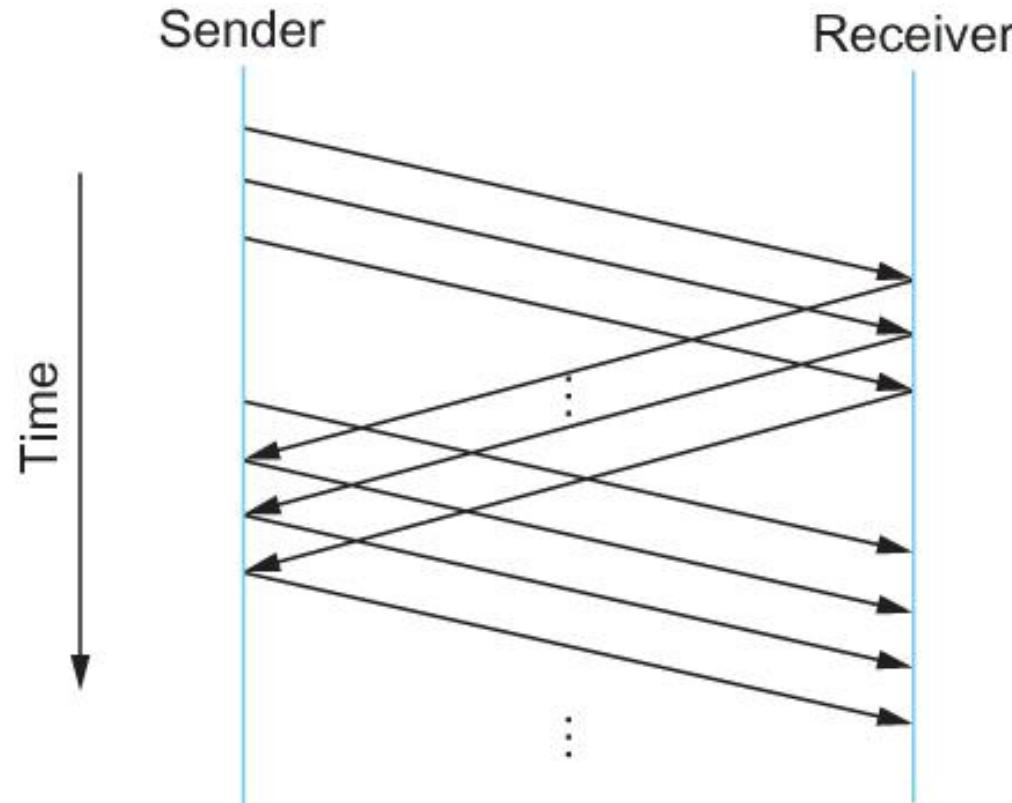
Stop and Wait Protocol

70

- The sender has only one outstanding frame on the link at a time
 - This may be far below the link's capacity
- Consider a 1.5 Mbps link with a 45 ms RTT
 - The link has a delay × bandwidth product of 67.5 Kb or approximately 8 KB
 - Since the sender can send only one frame per RTT and assuming a frame size of 1 KB
 - Maximum Sending rate
 - Bits per frame ÷ Time per frame = $1024 \times 8 \div 0.045 = 182 \text{ Kbps}$
Or about one-eighth of the link's capacity
 - *To use the link fully*, then sender should transmit up to **eight** frames before having to wait for an acknowledgement

Sliding Window Protocol

71



Timeline for Sliding Window Protocol

Sliding Window Protocol

72

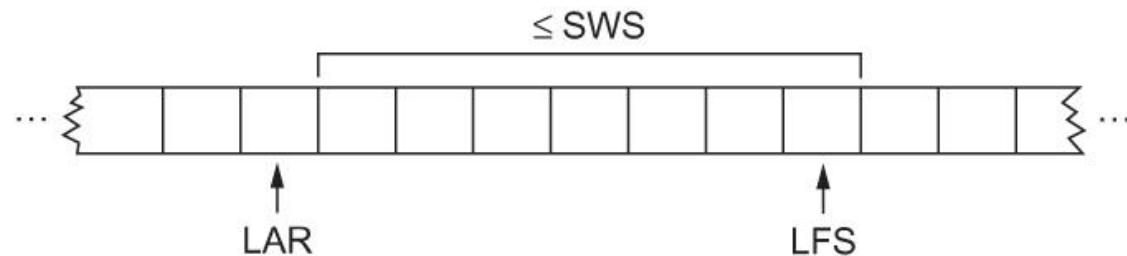
- Sender assigns a sequence number denoted as **SqNum** to each frame.
 - Assume it can grow infinitely large
- Sender maintains three variables
 - Sending Window Size (**SWS**)
 - Upper bound on the number of outstanding (unacknowledged) frames that the sender can transmit
 - Last Acknowledgement Received (**LAR**)
 - Sequence number of the last acknowledgement received
 - Last Frame Sent (**LFS**)
 - Sequence number of the last frame sent

Sliding Window Protocol

73

- Sender also maintains the following invariant

$$\text{LFS} - \text{LAR} \leq \text{SWS}$$



Sliding Window on Sender

Sliding Window Protocol

74

- When an acknowledgement arrives
 - the sender moves **LAR** to right, thereby allowing the sender to transmit another frame
- Also the sender associates a timer with each frame it transmits
 - It retransmits the frame if the timer expires before the **ACK** is received
- Note that the sender has to be willing to buffer up to **SWS** frames

Sliding Window Protocol

75

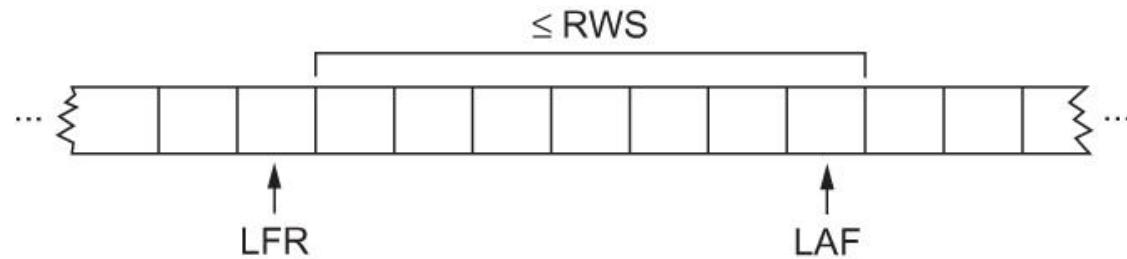
- Receiver maintains three variables
 - Receiving Window Size (**RWS**)
 - Upper bound on the number of out-of-order frames that the receiver is willing to accept
 - Largest Acceptable Frame (**LAF**)
 - Sequence number of the largest acceptable frame
 - Last Frame Received (**LFR**)
 - Sequence number of the last frame received

Sliding Window Protocol

76

- Receiver also maintains the following invariant

$$\text{LAF} - \text{LFR} \leq \text{RWS}$$



Sliding Window on Receiver

Sliding Window Protocol

77

- When a frame with sequence number **SeqNum** arrives, what does the receiver do?
 - If $\text{SeqNum} \leq \text{LFR}$ or $\text{SeqNum} > \text{LAF}$
 - Discard it (the frame is outside the receiver window)
 - If $\text{LFR} < \text{SeqNum} \leq \text{LAF}$
 - Accept it
 - Now the receiver needs to decide whether or not to send an ACK

Sliding Window Protocol

78

- Let **SeqNumToAck**
 - Denote the largest sequence number not yet acknowledged, such that all frames with sequence number less than or equal to SeqNumToAck have been received
- The receiver acknowledges the receipt of SeqNumToAck even if high-numbered packets have been received
 - This acknowledgement is said to be cumulative.
- The receiver then sets
 - $LFR = SeqNumToAck$ and adjusts
 - $LAF = LFR + RWS$

Sliding Window Protocol

79

For example, suppose LFR = 5 and RWS = 4

(i.e. the last ACK that the receiver sent was for seq. no. 5)

⇒ LAF = 9

If frames 7 and 8 arrive, they will be buffered because they are within the receiver window

But no ACK will be sent since frame 6 is yet to arrive

Frames 7 and 8 are out of order

Frame 6 arrives (it is late because it was lost first time and had to be retransmitted)

Now Receiver Acknowledges Frame 8
and bumps LFR to 8
and LAF to 12

Issues with Sliding Window Protocol

80

- When timeout occurs, the amount of data in transit decreases
 - ▣ Since the sender is unable to advance its window
- When the packet loss occurs, this scheme is no longer keeping the pipe full
 - ▣ The longer it takes to notice that a packet loss has occurred, the more severe the problem becomes
- How to improve this
 - ▣ Negative Acknowledgement (NAK)
 - ▣ Additional Acknowledgement
 - ▣ Selective Acknowledgement

Issues with Sliding Window Protocol

81

- Negative Acknowledgement (NAK)
 - Receiver sends NAK for frame 6 when frame 7 arrives (in the previous example)
 - However this is unnecessary since sender's timeout mechanism will be sufficient to catch the situation
- Additional Acknowledgement
 - Receiver sends additional ACK for frame 5 when frame 7 arrives
 - Sender uses duplicate ACK as a clue for frame loss
- Selective Acknowledgement
 - Receiver will acknowledge exactly those frames it has received, rather than the highest number frames
 - Receiver will acknowledge frames 7 and 8
 - Sender knows frame 6 is lost
 - Sender can keep the pipe full (additional complexity)

Issues with Sliding Window Protocol

82

How to select the window size

- SWS is easy to compute
 - Delay × Bandwidth
- RWS can be anything
 - Two common setting
 - RWS = 1
 - No buffer at the receiver for frames that arrive out of order
 - RWS = SWS
 - The receiver can buffer frames that the sender transmits
 - It does not make any sense to keep RWS > SWS

Issues with Sliding Window Protocol

83

- Finite Sequence Number
 - Frame sequence number is specified in the header field
 - Finite size
 - 3 bit: eight possible sequence number: 0, 1, 2, 3, 4, 5, 6, 7
 - It is necessary to wrap around

Issues with Sliding Window Protocol

84

- How to distinguish between different incarnations of the same sequence number?
 - Number of possible sequence number must be larger than the number of outstanding frames allowed
 - Stop and Wait: One outstanding frame
 - 2 distinct sequence number (0 and 1)
 - Let **MaxSeqNum** be the number of available sequence numbers
 - $SWS + 1 \leq \text{MaxSeqNum}$
 - Is this sufficient?

Issues with Sliding Window Protocol

85

$SWS + 1 \leq \text{MaxSeqNum}$

- Is this sufficient?
 - Depends on RWS
 - If RWS = 1, then sufficient
 - If RWS = SWS, then not good enough
- For example, we have eight sequence numbers

0, 1, 2, 3, 4, 5, 6, 7

RWS = SWS = 7

Sender sends 0, 1, ..., 6

Receiver receives 0, 1, ..., 6

Receiver acknowledges 0, 1, ..., 6

ACK (0, 1, ..., 6) are lost

Sender retransmits 0, 1, ..., 6

Receiver is expecting 7, 0, ..., 5

Issues with Sliding Window Protocol

86

To avoid this,

If $RWS = SWS$

$$SWS < (\text{MaxSeqNum} + 1)/2$$

Issues with Sliding Window Protocol

87

- Serves three different roles
 - ▣ Reliable
 - ▣ Preserve the order
 - Each frame has a sequence number
 - The receiver makes sure that it does not pass a frame up to the next higher-level protocol until it has already passed up all frames with a smaller sequence number
 - ▣ Frame control
 - Receiver is able to throttle the sender
 - Keeps the sender from overrunning the receiver
 - From transmitting more data than the receiver is able to process

88

Ethernet

Ethernet

89

- Most successful local area networking technology of last 20 years.
- Developed in the mid-1970s by researchers at the Xerox Palo Alto Research Centers (PARC).
- Uses CSMA/CD technology
 - Carrier Sense Multiple Access with Collision Detection.
 - A set of nodes send and receive frames over a shared link.
 - Carrier sense means that all nodes can distinguish between an idle and a busy link.
 - Collision detection means that a node listens as it transmits and can therefore detect when a frame it is transmitting has collided with a frame transmitted by another node.

Ethernet (802.3)

90

- Uses ALOHA (packet radio network) as the root protocol
 - Developed at the University of Hawaii to support communication across the Hawaiian Islands.
 - For ALOHA the medium was atmosphere, for Ethernet the medium is a coax cable.
- DEC and Intel joined Xerox to define a 10-Mbps Ethernet standard in 1978.
- This standard formed the basis for IEEE standard 802.3
- More recently 802.3 has been extended to include a 100-Mbps version called Fast Ethernet and a 1000-Mbps version called Gigabit Ethernet.

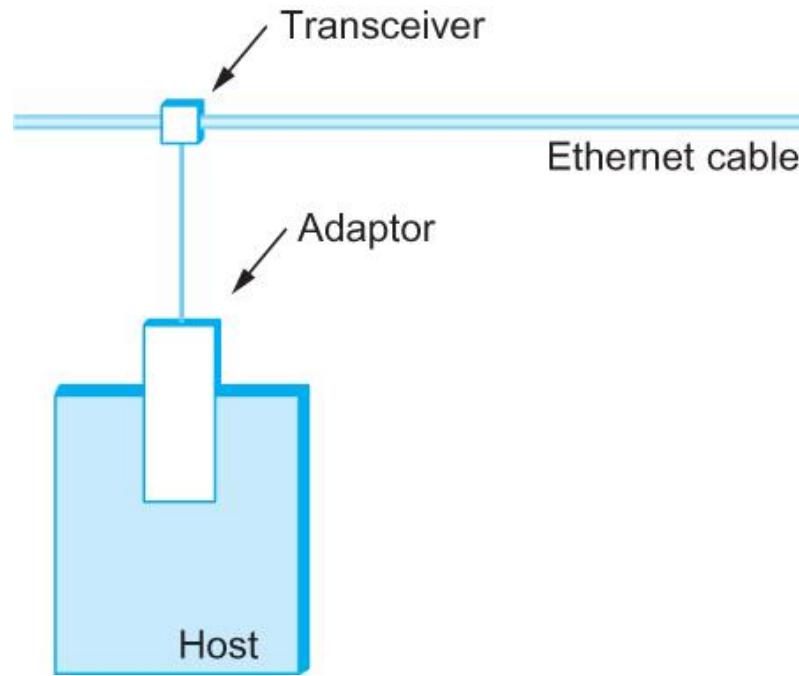
Ethernet – Physical Properties

91

- An Ethernet segment is implemented on a coaxial cable of up to 500 m.
 - This cable is similar to the type used for cable TV except that it typically has an impedance of 50 ohms instead of cable TV's 75 ohms.
- Hosts connect to an Ethernet segment by tapping into it.
- A transceiver (a small device directly attached to the tap) detects when the line is idle and drives signal when the host is transmitting.
- The transceiver also receives incoming signal.
- The transceiver is connected to an Ethernet adaptor which is plugged into the host.
- The protocol is implemented on the adaptor.

Ethernet – Physical Properties

92



Ethernet transceiver and adaptor

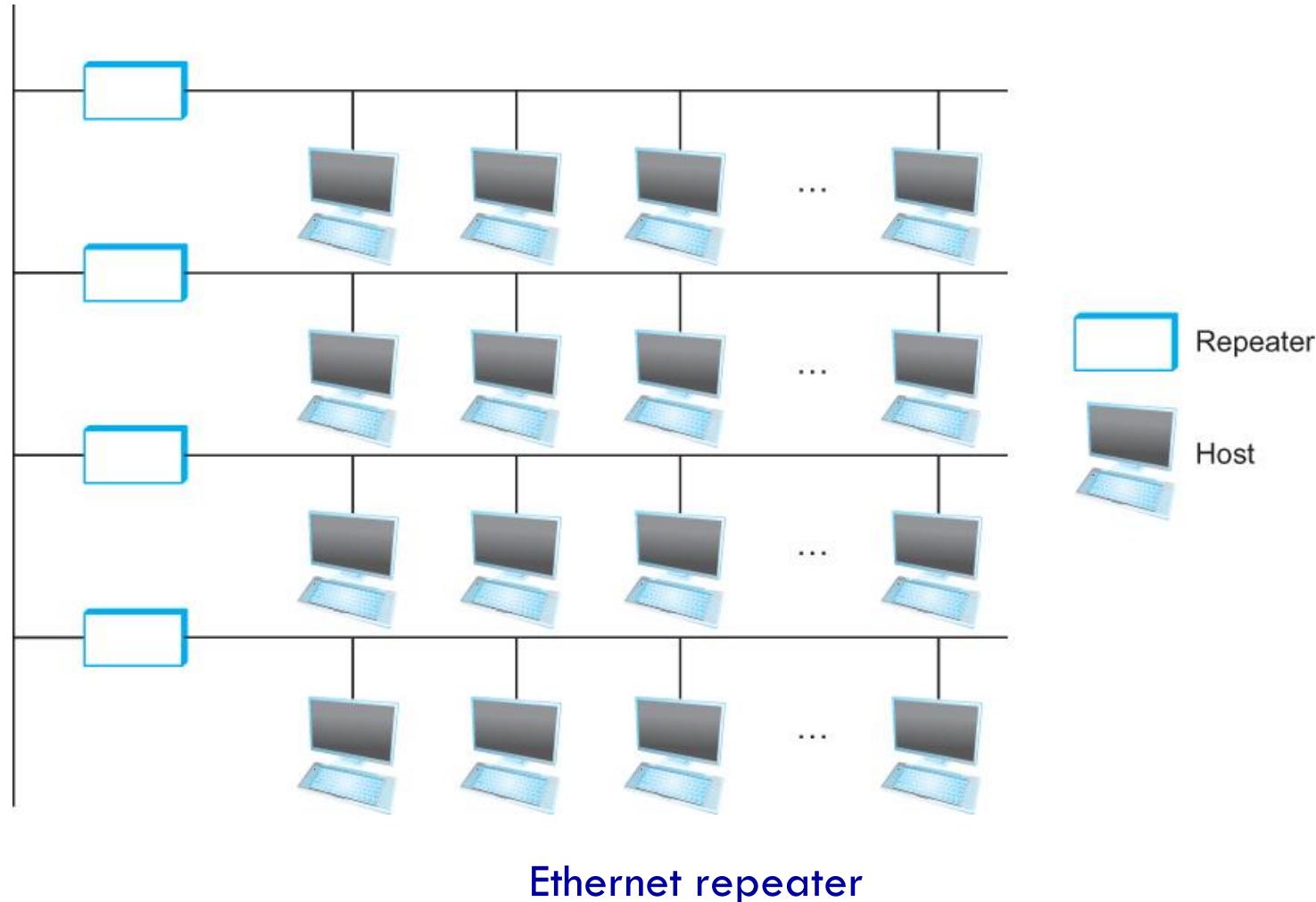
Ethernet – Physical Properties

93

- Multiple Ethernet segments can be joined together by *repeaters*.
- A *repeater* is a device that forwards digital signals.
- No more than four repeaters may be positioned between any pair of hosts.
 - ▣ An Ethernet has a total reach of only 2500 m.

Ethernet Repeater – Physical Properties

94



Ethernet – Physical Properties

95

- Any signal placed on the Ethernet by a host is broadcast over the entire network
 - Signal is propagated in both directions.
 - Repeaters forward the signal on all outgoing segments.
 - Terminators attached to the end of each segment absorb the signal.
- Ethernet uses Manchester encoding scheme.

Ethernet – Physical Properties

96

□ New Technologies in Ethernet

- Instead of using coax cable, an Ethernet can be constructed from a thinner cable known as 10Base2 (the original was 10Base5)
 - 10 means the network operates at 10 Mbps
 - Base means the cable is used in a baseband system
 - 2 means that a given segment can be no longer than 200 m

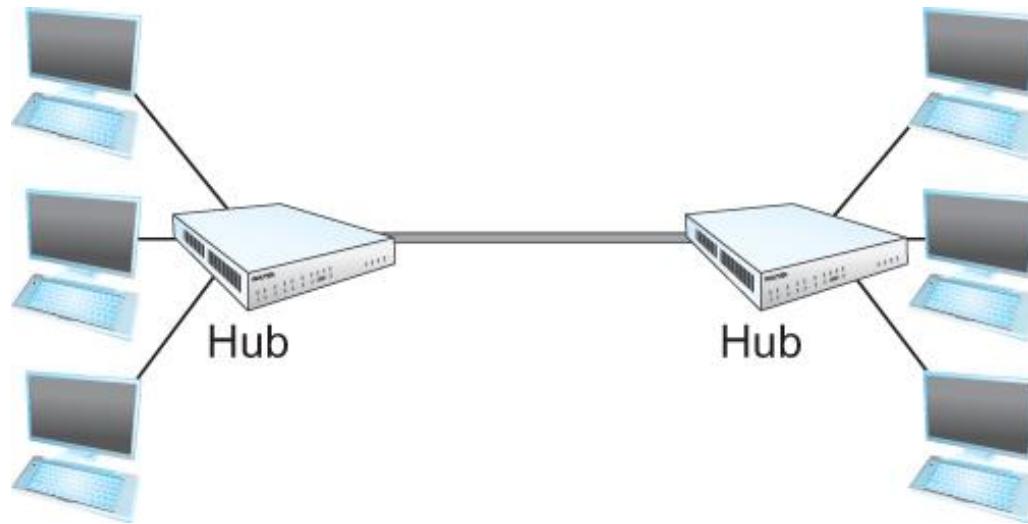
Ethernet – Physical Properties

97

- New Technologies in Ethernet
 - Another cable technology is 10BaseT
 - T stands for twisted pair
 - Limited to 100 m in length
 - With 10BaseT, the common configuration is to have several point to point segments coming out of a multiway repeater, called *Hub*

Ethernet Hub

98



Ethernet Hub

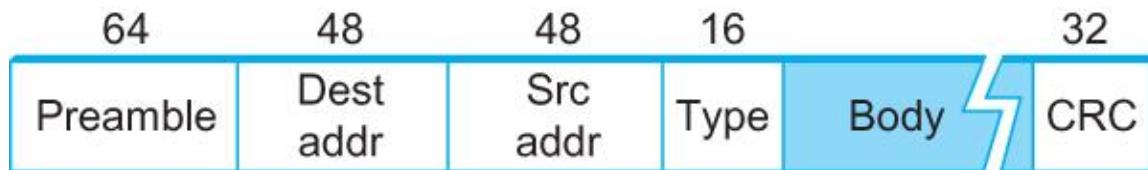
Access Protocol for Ethernet

99

- The algorithm is commonly called Ethernet's Media Access Control (MAC).
 - It is implemented in Hardware on the network adaptor.
- Frame format
 - Preamble (64bit): allows the receiver to synchronize with the signal (sequence of alternating 0s and 1s).
 - Host and Destination Address (48bit each).
 - Packet type (16bit): acts as demux key to identify the higher level protocol.
 - Data (up to 1500 bytes)
 - Minimally a frame must contain at least 46 bytes of data.
 - Frame must be long enough to detect collision.
 - CRC (32bit)

Ethernet Frame

100



Ethernet Frame Format

Ethernet Address

101

- Each host on an Ethernet (in fact, every Ethernet host in the world) has a unique Ethernet Address.
- The address belongs to the adaptor, not the host.
 - It is usually burnt into ROM.
- Ethernet addresses are typically printed in a human readable format
 - As a sequence of six numbers separated by colons.
 - Each number corresponds to 1 byte of the 6 byte address and is given by a pair of hexadecimal digits, one for each of the 4-bit nibbles in the byte
 - Leading 0s are dropped.
 - For example, 8:0:2b:e4:b1:2 is
 - 00001000 00000000 00101011 11100100 10110001 00000010

Ethernet Addresses

102

- To ensure that every adaptor gets a unique address, each manufacturer of Ethernet devices is allocated a **different prefix** that must be prepended to the address on every adaptor they build
 - AMD has been assigned the 24bit prefix 8:0:20

Ethernet Addresses

103

- Each frame transmitted on an Ethernet is received by every adaptor connected to that Ethernet.
- Each adaptor recognizes those frames addressed to its address and passes only those frames on to the host.
- In addition, to *unicast* address, an Ethernet address consisting of *all 1s* is treated as a **broadcast** address.
 - All adaptors pass frames addressed to the *broadcast* address up to the host.
- Similarly, an address that has the *first bit set to 1* but is not the *broadcast* address is called a **multicast** address.
 - A given host can program its adaptor to accept some set of *multicast* addresses.

Ethernet Addresses

104

- To summarize, an Ethernet adaptor receives all frames and accepts
 - Frames addressed to its own address
 - Frames addressed to the broadcast address
 - Frames addressed to a multicast address if it has been instructed

Ethernet Transmitter Algorithm

105

- When the adaptor has a frame to send and the line is idle, it transmits the frame immediately.
 - The upper bound of 1500 bytes in the message means that the adaptor can occupy the line for a fixed length of time.
- When the adaptor has a frame to send and the line is busy, it waits for the line to go idle and then transmits immediately.
- The Ethernet is said to be *1-persistent protocol* because an adaptor with a frame to send transmits with probability 1 whenever a busy line goes idle.

Ethernet Transmitter Algorithm

106

- Since there is no centralized control it is possible for two (or more) adaptors to begin transmitting at the same time,
 - ▣ Either because both found the line to be idle,
 - ▣ Or, both had been waiting for a busy line to become idle.
- When this happens, the two (or more) frames are said to be *collide* on the network.

Ethernet Transmitter Algorithm

107

- Since Ethernet supports collision detection, each sender is able to determine that a collision is in progress.
- At the moment an adaptor detects that its frame is colliding with another, it first makes sure to transmit a **32-bit jamming** sequence and then stops transmission.
 - Thus, a transmitter will minimally send **96 bits** in the case of collision
 - **64-bit preamble + 32-bit jamming sequence**

Ethernet Transmitter Algorithm

108

- One way that an adaptor will send only 96 bit (called a *runt frame*) is if the two hosts are close to each other.
- Had they been farther apart,
 - They would have had to transmit longer, and thus send more bits, before detecting the collision.

Ethernet Transmitter Algorithm

109

- The worst case scenario happens when the two hosts are at opposite ends of the Ethernet.
- To know for sure that the frame its just sent did not collide with another frame, the transmitter may need to send as many as 512 bits.
 - Every Ethernet frame must be at least 512 bits (64 bytes) long.
 - 14 bytes of header + 46 bytes of data + 4 bytes of CRC

Ethernet Transmitter Algorithm

110

- Why 512 bits?
 - Why is its length limited to 2500 m?
- The farther apart two nodes are, the longer it takes for a frame sent by one to reach the other, and the network is **vulnerable to collision** during this time

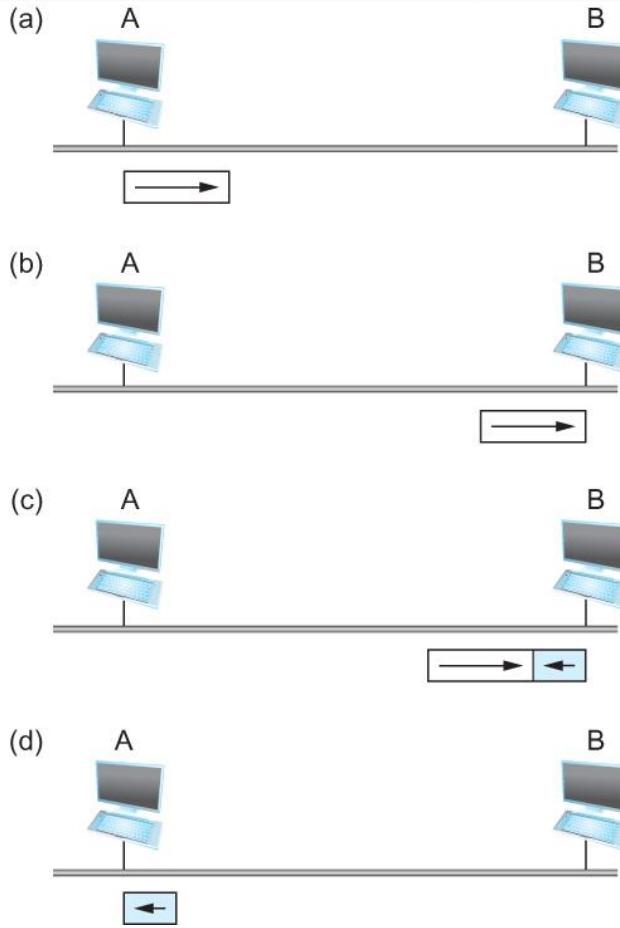
Ethernet Transmitter Algorithm

111

- Node A begins transmitting a frame at time t
- d denotes the one link latency
- The first bit of A's frame arrives at B at time $t + d$
- Suppose an instant before host A's frame arrives, host B begins to transmit its own frame
- B's frame will immediately collide with A's frame and this collision will be detected by host B
- Host B will send the 32-bit jamming sequence
- Host A will not know that the collision occurred until B's frame reaches it, which will happen at $t + 2 * d$
- Host A must continue to transmit until this time in order to detect the collision
 - Host A must transmit for $2 * d$ to be sure that it detects all possible collisions

Ethernet Transmitter Algorithm

112



Worst-case scenario:

- Host A sends a frame at time t ;
- Host A's frame arrives at B at time $t + d$;
- Host B begins transmitting at time $t + d$ and collides with A's frame;
- B's runt (32-bit) frame arrives at A at time $t + 2d$.

Ethernet Transmitter Algorithm

113

- Consider that a maximally configured Ethernet is 2500 m long, and there may be up to four repeaters between any two hosts, the round trip delay has been determined to be $51.2 \mu\text{s}$
 - ▣ Which on 10 Mbps Ethernet corresponds to 512 bits
- The other way to look at this situation,
 - ▣ We need to limit the Ethernet's maximum latency to a fairly small value ($51.2 \mu\text{s}$) for the access algorithm to work
 - Hence the maximum length for the Ethernet is on the order of 2500 m.

Ethernet Transmitter Algorithm

114

- Once an adaptor has detected a collision, and stopped its transmission, it waits a certain amount of time and tries again.
- Each time the adaptor tries to transmit but fails, it doubles the amount of time it waits before trying again.
- This strategy of doubling the delay interval between each retransmission attempt is known as *Exponential Backoff*.

Ethernet Transmitter Algorithm

115

- The adaptor first delays either 0 or 51.2 μ s, selected at random.
- If this effort fails, it then waits 0, 51.2, 102.4, 153.6 μ s (selected randomly) before trying again;
 - This is $k * 51.2$ for $k = 0, 1, 2, 3$
- After the third collision, it waits $k * 51.2$ for $k = 0...2^3 - 1$ (again selected at random).
- In general, the algorithm randomly selects a k between 0 and $2^n - 1$ and waits for $k * 51.2 \mu$ s, where n is the number of collisions experienced so far.

Experience with Ethernet

116

- Ethernets work best under lightly loaded conditions.
 - Under **heavy loads**, too much of the network's capacity is **wasted by collisions**.
- Most Ethernets are used in a conservative way.
 - Have fewer than 200 hosts connected to them which is far fewer than the maximum of 1024.
- Most Ethernets are far shorter than 2500m with a round-trip delay of closer to 5 μ s than 51.2 μ s.
- Ethernets are easy to administer and maintain.
 - There are no switches that can fail and no routing and configuration tables that have to be kept up-to-date.
 - It is easy to add a new host to the network.
 - It is inexpensive.
 - Cable is cheap, and only other cost is the network adaptor on each host.

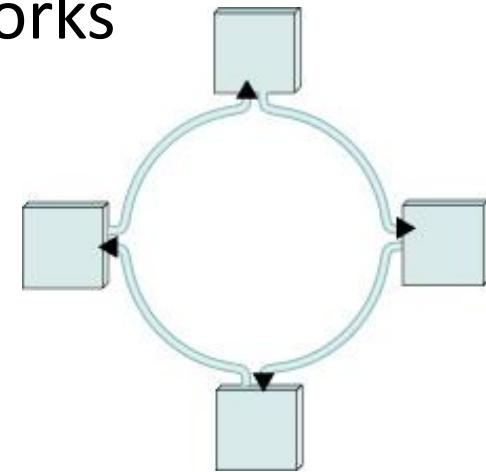
117

Rings (802.5, FDDI, RPR)

Ring Networks

118

- Like Ethernets, are shared media networks
- Token Ring Networks
 - ▣ PRONET: 10Mbps and 80 Mbps rings
 - ▣ IBM: 4Mbps token ring
 - ▣ 16Mbps IEEE 802.5/token ring
 - ▣ 100Mbps Fiber Distributed Data Interface (FDDI)
- A ring networks consists of a set of nodes connected in a ring



Ring Networks

119

- Basic Idea
 - frames flow in one direction: upstream to downstream
 - special bit pattern (token) rotates around ring
 - must capture token before transmitting
 - release token after done transmitting
 - immediate release
 - delayed release
 - remove your frame when it comes back around
 - stations get round-robin service

IEEE 802.5 Token Ring

120

- Consists of a set of nodes connected in a ring.
- Data flows in a particular direction only.
- Data received from upstream neighbour forwarded to downstream neighbour.
- Token – access to the shared ring
 - a special sequence of bits
 - circulates around the ring.

IEEE 802.5 Token Ring

121

- Each node receives and forwards token.
- Frame makes its way back to sender
 - frame removed by sender
 - sender reinsert token.
- As token circulates around ring, each station gets a chance to transmit
 - Service round - robin fashion

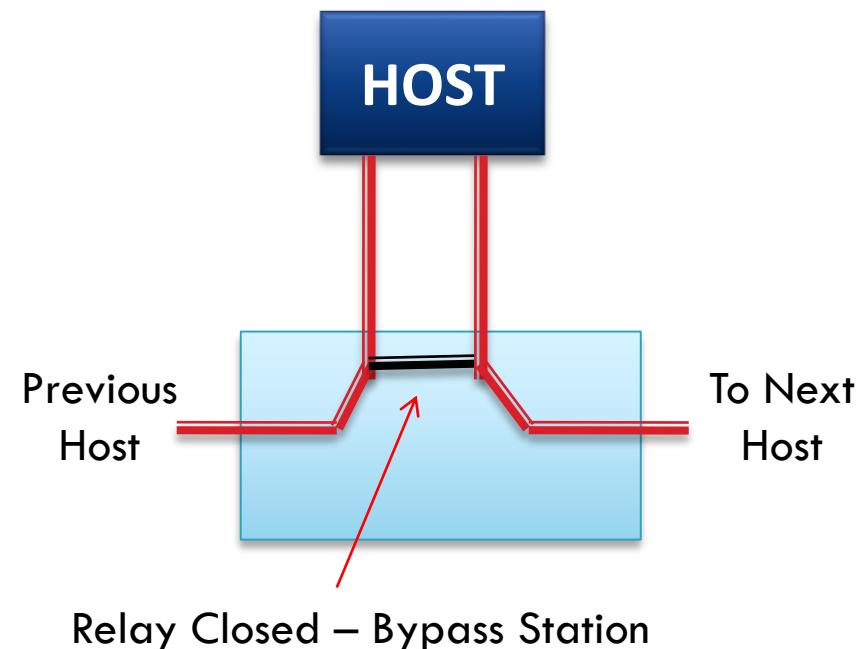
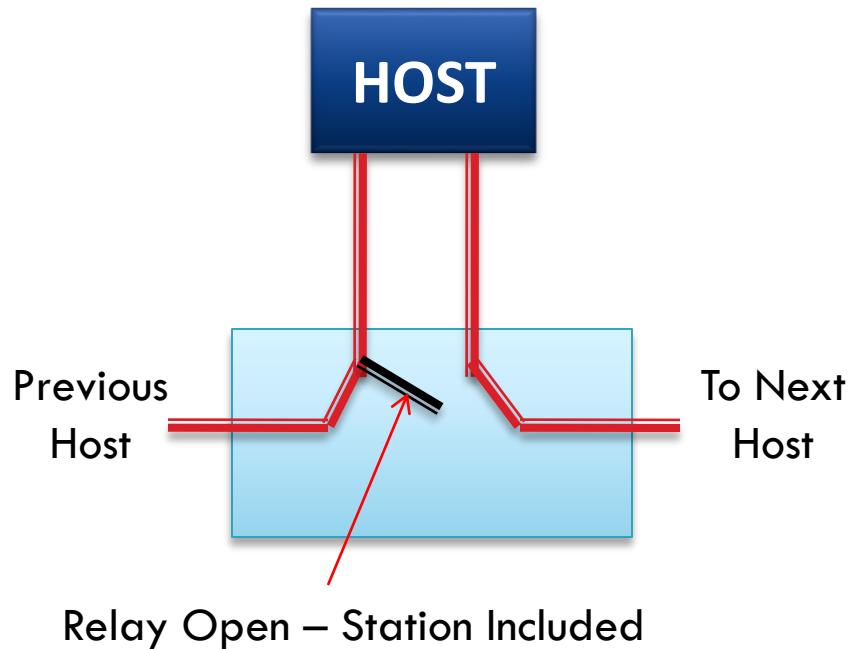
Token Ring Issues

122

- Any link or node failure
 - Network rendered useless
- Solution –
 - electromechanical relay
 - Station active relay is open and station included
 - Station is inactive
 - no power
 - relay closed
 - bypass station

Token Ring Issues

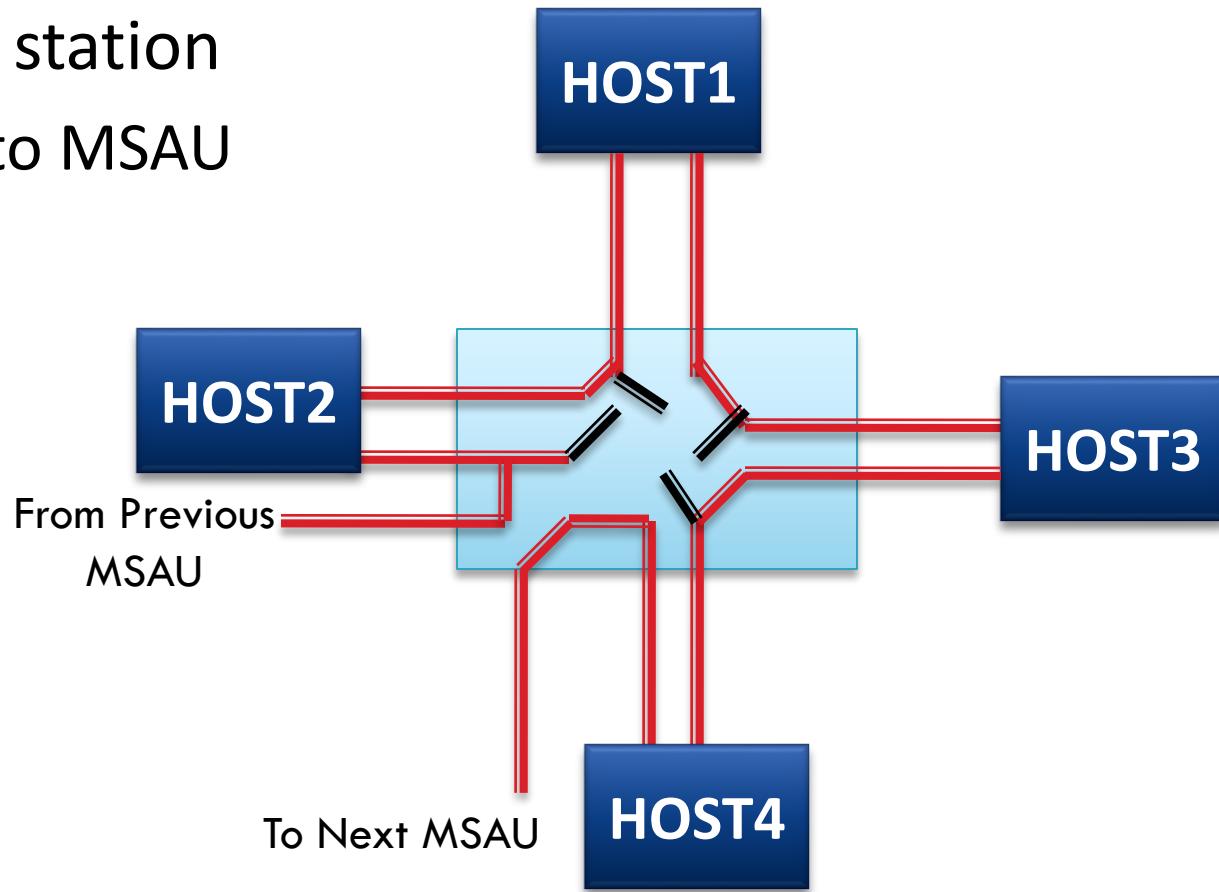
123



Multistation Access Unit (MSAU)

124

- Several relays in a box
- Add new station
 - Plug into MSAU



Token Ring (Characteristics)

125

- Data rate: 4 Mbps or 16 Mbps
- encoding: differential manchester
- 802.5 upto 250 station

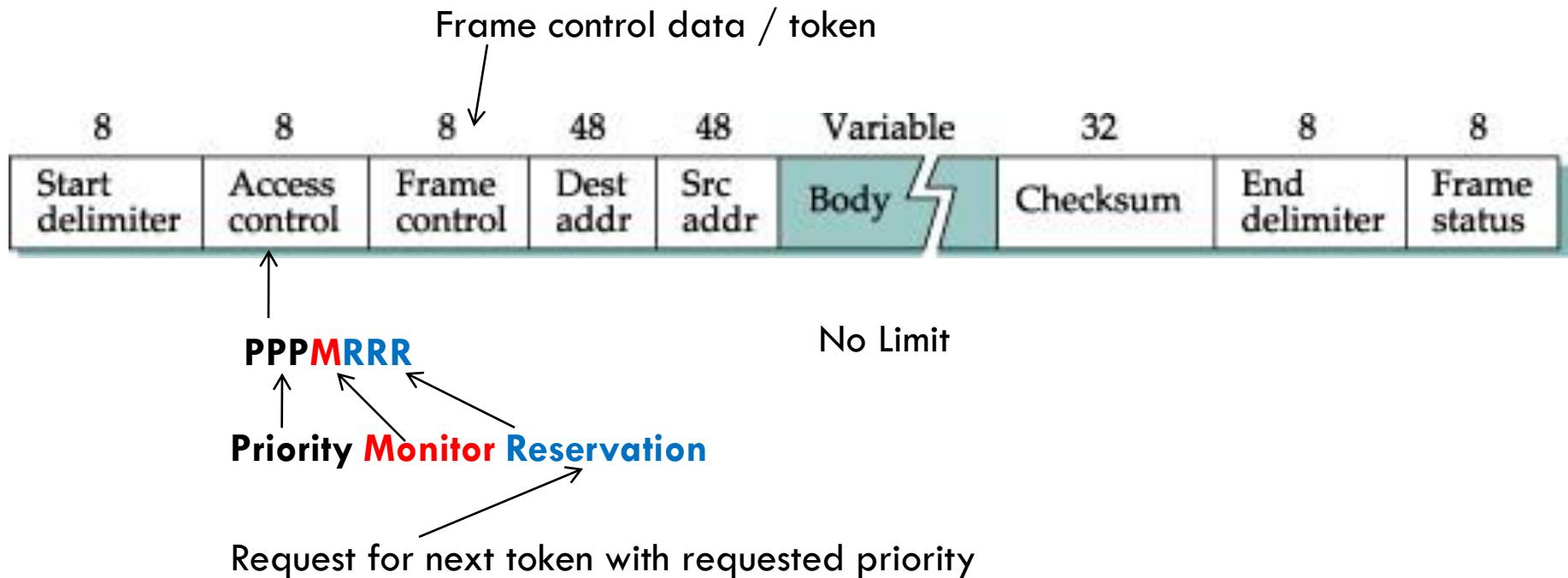
Token Ring Access Control

126

- Network adapter: receiver, and transmitter, and one or more bits of data storage between them.
- When no stations have anything to transmit token circulates
- Ring has enough storage capacity to hold an entire token.
 - 1 bit / station

Token Ring Frame Format

127



IEEE 802.5

128

- Token Size: 24 bits
 - Minimum number of stations is 24
 - Overcome this by including a monitor which adds the extra bits of delay
- Token operation
 - Token circulates
 - Station seizes a token

IEEE 802.5

129

- Modifies a bit in the second byte of token
- Station that has token transmits data
- Station drains token out of the ring
- Station sends data
- Each packet has destination address
- All stations downhill check destination address
- Destination copies packet
- Packet finds its way back to sending station

IEEE 802.5

130

- Sending station removes packet from ring
- Station reinserts token into the ring

- Size of packet stored in the ring
 - Larger/smaller than ring
 - Add/remove bits

IEEE 802.5

131

□ Issues

- Size of data that given node is allowed to transmit
- Token holding time (THT) = ∞ ?
 - Utilisation is 100%
 - Unfair to stations to other than the station holding the token
- THT affects ring performance

Token Holding Time

132

- Token Rotation Time (TRT):
- $\text{TRT} \leq \text{Active nodes} * \text{THT} + \text{Ring Latency}$
- Ring Latency – token circulation time

Reliable Transmission

133

- The 802.5 protocol provides a form of reliable delivery using 2 bits in the packet trailer
 - The **A** and **C** bits
- Initially **A** and **C** zero.
- Receiver sets **A** bit after seeing that it is the intended recipient
- Receiver sets **C** bit after copying frame
- If both **A** and **C** are not set – retransmit

Priorities in IEEE 802.5

134

- Supports different levels of priority
 - 3 bits
 - each station waiting to send, sets priority for packet
packet's priority as high current token
 - then token can be seized
 - Intending to send station – sets the priority on
currently passing data frame

Priorities in IEEE 802.5

135

- releasing station sets priority of token to n.
- Lower priority packets circulate for long in ring

- Token Release
 - Early release
 - After transmitting packet
 - Delayed release
 - After removing packet when it returns to the sender

Token Ring Maintenance

136

- Designated monitor
 - any station can become a monitor
 - defined procedures for becoming a monitor
 - healthy monitor announces that it is a monitor at periodic interval
 - if a station does not see that packet for some time – then it sends a “claim token”
 - if claim token comes back to station then it is monitor
 - if another wants to claim see other stations claim first some arbitration rule.

Token Ring Maintenance

137

- Role of monitor
 - insert additional delay in ring
 - ensure always that there is a token somewhere in the ring
 - regenerate a vanished token
 - no token seen for TRT => regenerate

Token Ring Maintenance

138

- Orphaned / corrupted packets – drain them if orphaned
 - (A and C bits set – parent dies)
 - A bit set C bit not set – parent dies
- bit is initially set to 1 by monitor
 - monitor notices back when packet passes by monitor a second time

Token Ring Maintenance

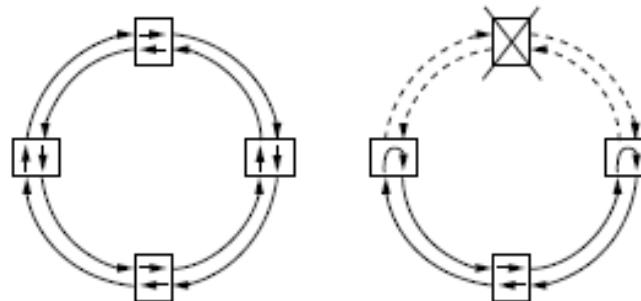
139

- Detection of dead stations
 - some problem un detected
 - suspecting station sends a beacon frame –
 - how far beacon goes decide which stations must be bypassed

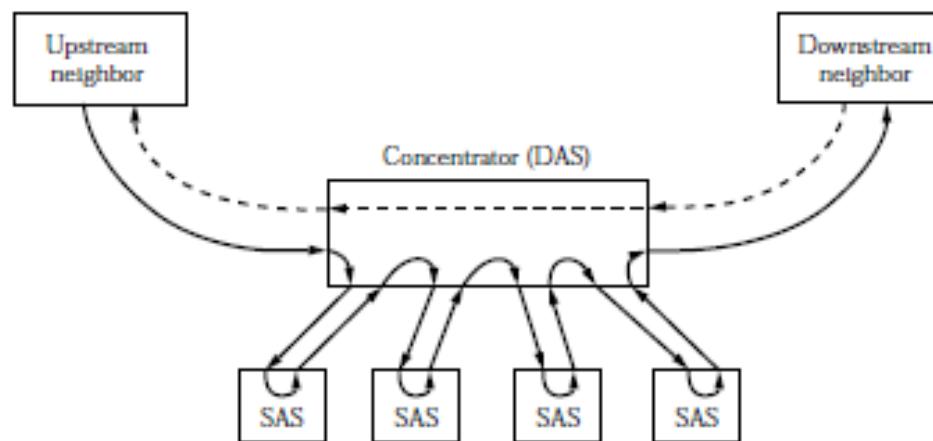
Physical Properties of FDDI

140

□ Dual Ring Configuration



□ Single and Dual Attachment Stations



Physical Properties of FDDI

141

- Each station imposes a delay (e.g., 50ns)
- Maximum of 500 stations
- Upper limit of 100km (200km of fiber)
- Uses 4B/5B encoding
- Can be implemented over copper (CDDI)

142

Wireless (802.11)

Wireless Links

143

- Wireless links transmit electromagnetic signals
 - Radio, microwave, infrared
- Wireless links all share the same “**wire**” (so to speak)
 - The challenge is to share it efficiently without unduly interfering with each other
 - Most of this sharing is accomplished by dividing the “wire” along the dimensions of frequency and space
- Exclusive use of a particular frequency in a particular geographic area may be allocated to an individual entity such as a corporation

Wireless Links

144

- These allocations are determined by government agencies such as FCC (Federal Communications Commission) in USA
- Specific bands (frequency) ranges are allocated to certain uses.
 - Some bands are reserved for government use
 - Other bands are reserved for uses such as AM radio, FM radio, televisions, satellite communications, and cell phones
 - Specific frequencies within these bands are then allocated to individual organizations for use within certain geographical areas.
 - Finally, there are several frequency bands set aside for “license exempt” usage
 - Bands in which a license is not needed

Wireless Links

145

- Devices that use license-exempt frequencies are still subject to certain restrictions
 - The first is a limit on transmission power
 - This limits the range of signal, making it less likely to interfere with another signal
 - For example, a cordless phone might have a range of about 100 feet.

Wireless Links

146

- The second restriction requires the use of **Spread Spectrum** technique
 - Idea is to spread the signal over a wider frequency band
 - So as to minimize the impact of interference from other devices
 - Originally designed for military use
 - ***Frequency hopping***
 - Transmitting signal over a random sequence of frequencies
 - First transmitting at one frequency, then a second, then a third...
 - The sequence of frequencies is not truly random, instead computed algorithmically by a pseudorandom number generator
 - The receiver uses the same algorithm as the sender, initializes it with the same seed, and is
 - Able to hop frequencies in sync with the transmitter to correctly receive the frame

Wireless Links

147

- A second spread spectrum technique called ***Direct sequence***
 - Represents each bit in the frame by multiple bits in the transmitted signal.
 - For each bit the sender wants to transmit
 - It actually sends the exclusive OR of that bit and n random bits
 - The sequence of random bits is generated by a pseudorandom number generator known to both the sender and the receiver.
 - The transmitted values, known as an ***n-bit chipping code***, spread the signal across a frequency band that is n times wider

Wireless Links

148



Data stream: 1010



Random sequence: 0100101101011001



XOR of the two: 1011101110101001

Example 4-bit chipping sequence

Wireless Links

149

- Wireless technologies differ in a variety of dimensions
 - How much bandwidth they provide
 - Distance – How far apart the communication nodes can be
- Four prominent wireless technologies
 - Bluetooth
 - Wi-Fi (more formally known as 802.11)
 - WiMAX (802.16)
 - 3G cellular wireless

Wireless Links

150

	Bluetooth (802.15.1)	Wi-Fi (802.11)	3G Cellular
Typical link length	10 m	100 m	Tens of kilometers
Typical data rate	2 Mbps (shared)	54 Mbps (shared)	Hundreds of kbps (per connection)
Typical use	Link a peripheral to a computer	Link a computer to a wired base	Link a mobile phone to a wired tower
Wired technology analogy	USB	Ethernet	DSL

Overview of leading wireless technologies

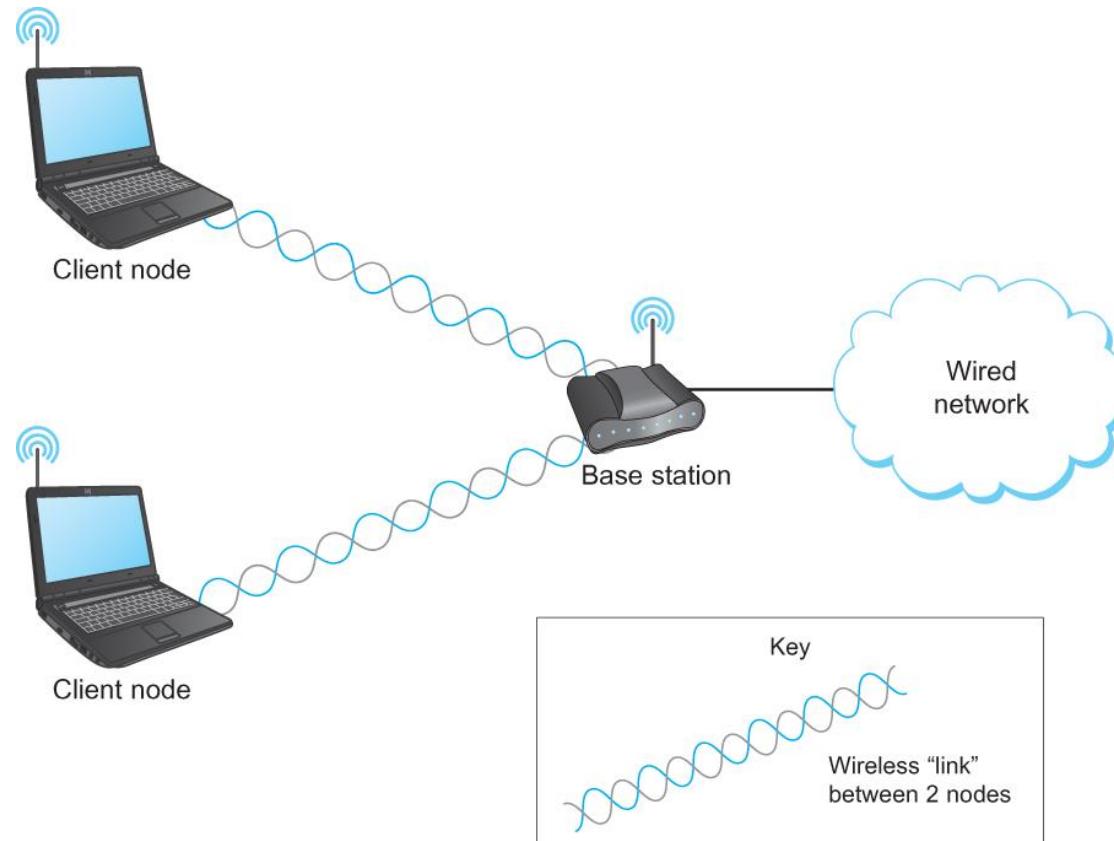
Wireless Links

151

- Mostly widely used wireless links today are usually asymmetric
 - Two end-points are usually different kinds of nodes
 - One end-point usually has no mobility, but has wired connection to the Internet (known as **base station**)
 - The node at the other end of the link is often mobile

Wireless Links

152



A wireless network using a base station

Wireless Links

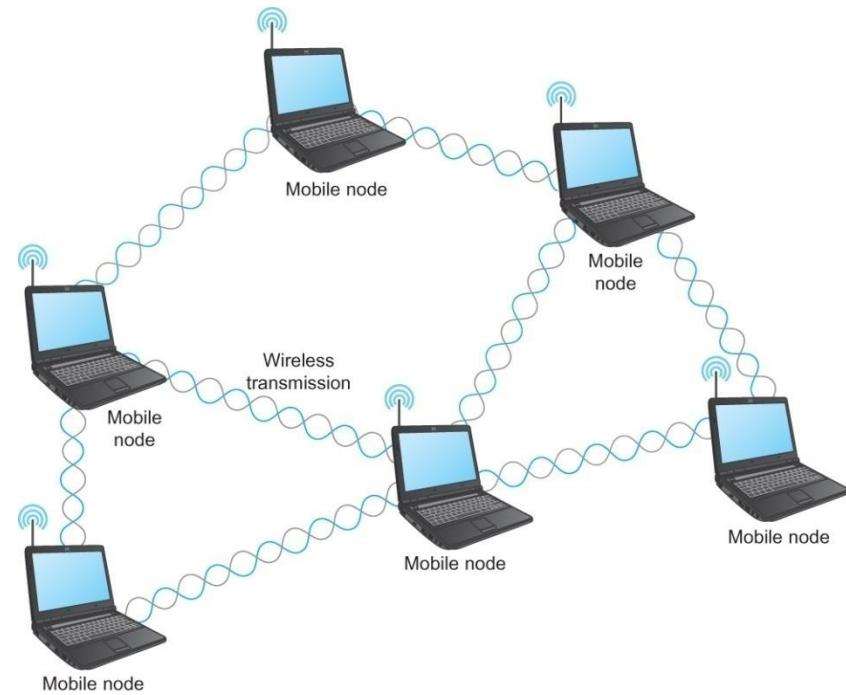
153

- Wireless communication supports point-to-multipoint communication
- Communication between non-base (client) nodes is routed via the base station
- Three levels of mobility for clients
 - No mobility: the receiver must be in a fix location to receive a directional transmission from the base station (initial version of WiMAX)
 - Mobility is within the range of a base (Bluetooth)
 - Mobility between bases (Cell phones and Wi-Fi)

Wireless Links

154

- Mesh or Ad-hoc network
 - Nodes are peers
 - Messages may be forwarded via a chain of peer nodes



A wireless ad-hoc or mesh network

IEEE 802.11

155

- Also known as Wi-Fi
- Like its Ethernet and token ring siblings, 802.11 is designed for use in a limited geographical area (homes, office buildings, campuses)
 - Primary challenge is to mediate access to a shared communication medium – in this case, signals propagating through space
- 802.11 supports additional features
 - power management and
 - security mechanisms

IEEE 802.11

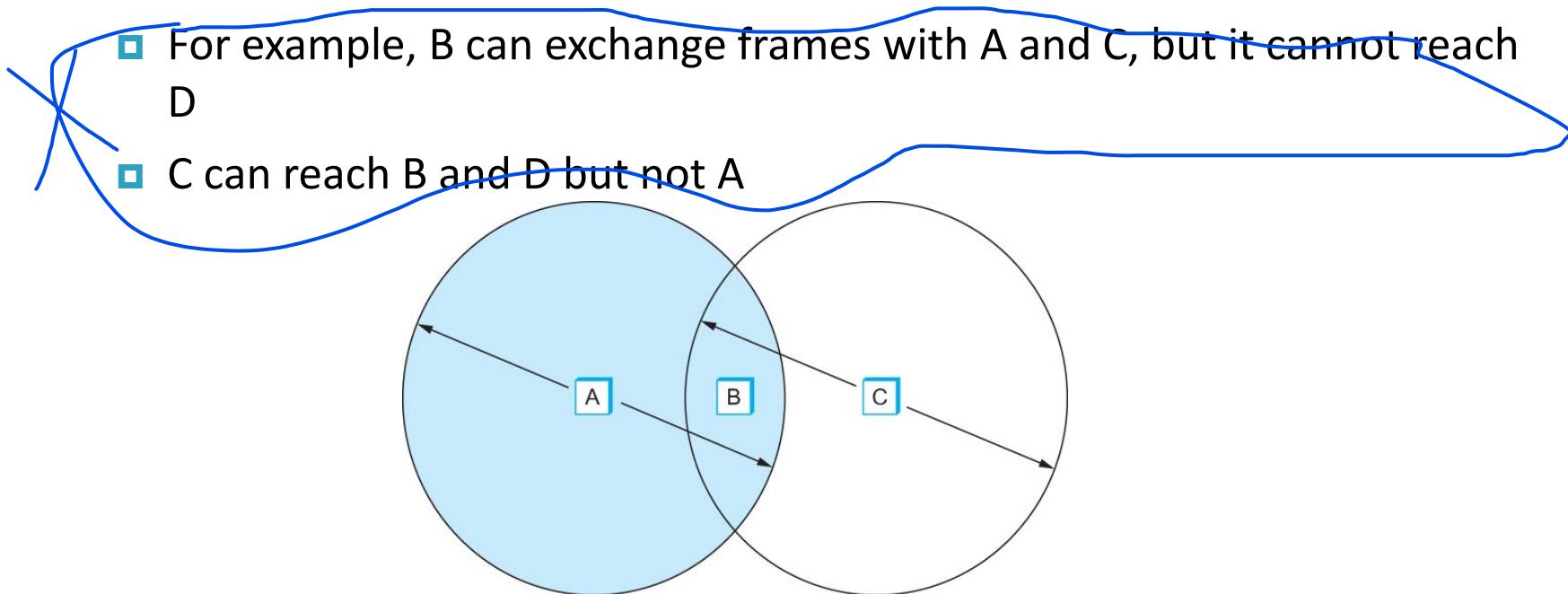
156

- Original 802.11 standard defined two radio-based physical layer standard
 - One using the frequency hopping
 - Over 79 1-MHz-wide frequency bandwidths
 - Second using direct sequence
 - Using 11-bit chipping sequence
 - Both standards run in the 2.4-GHz and provide up to 2 Mbps
- Then physical layer standard 802.11b was added
 - Using a variant of direct sequence 802.11b provides up to 11 Mbps
 - Uses license-exempt 2.4-GHz band
- Then came 802.11a which delivers up to 54 Mbps using OFDM
 - 802.11a runs on license-exempt 5-GHz band
- Most recent standard is 802.11g which is backward compatible with 802.11b
 - Uses 2.4 GHz band, OFDM and delivers up to 54 Mbps
- Latest – 802.11n uses 2.4GHz, MIMO (a signal processing and smart antenna technique for transmitting multiple data streams through multiple antennas) , 300 Mbps

IEEE 802.11 – Collision Avoidance

157

- Consider the situation in the following figure where each of four nodes is able to send and receive signals that reach just the nodes to its immediate left and right



Example of a wireless network

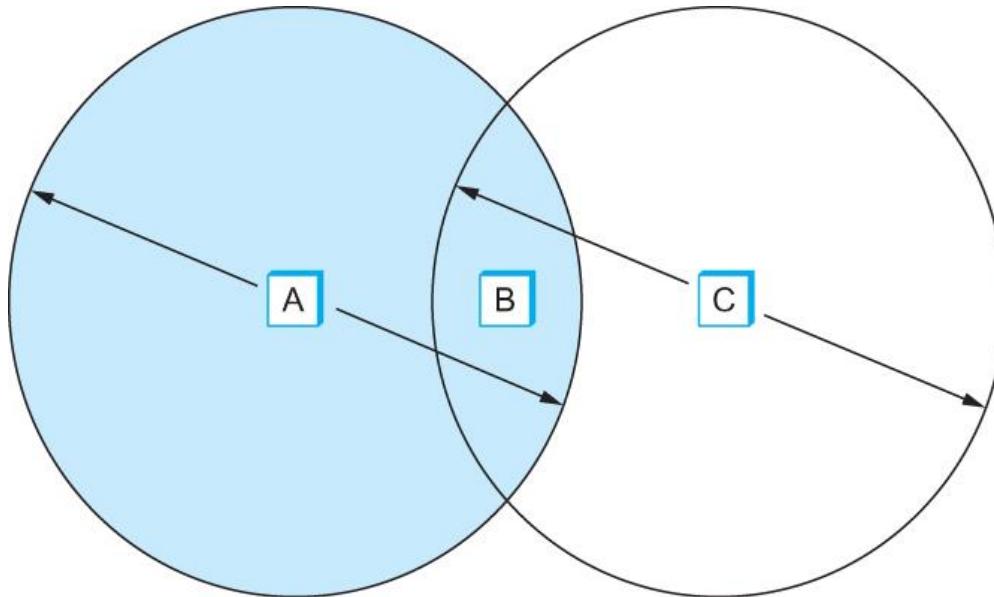
IEEE 802.11 – Collision Avoidance

158

- Suppose both A and C want to communicate with B and so they each send it a frame.
 - A and C are unaware of each other since their signals do not carry that far
 - These two frames collide with each other at B
 - But unlike an Ethernet, neither A nor C is aware of this collision
 - A and C are said to *hidden nodes* with respect to each other

IEEE 802.11 – Collision Avoidance

159



The “Hidden Node” Problem. Although A and C are hidden from each other, their signals can collide at B. (B’s reach is not shown.)

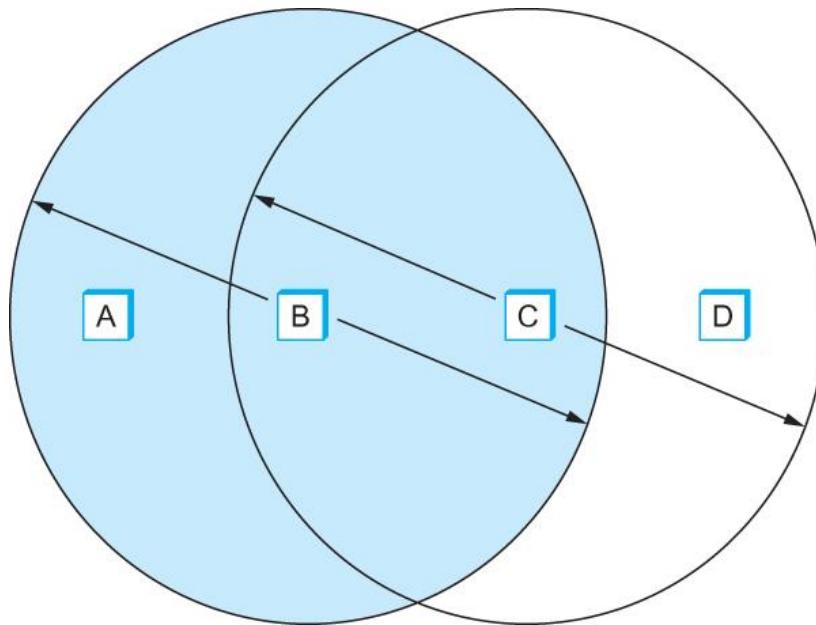
IEEE 802.11 – Collision Avoidance

160

- Another problem called *exposed node* problem occurs
 - Suppose B is sending to A. Node C is aware of this communication because it hears B's transmission.
 - It would be a mistake for C to conclude that it cannot transmit to anyone just because it can hear B's transmission.
 - Suppose C wants to transmit to node D.
 - This is not a problem since C's transmission to D will not interfere with A's ability to receive from B.

IEEE 802.11 – Collision Avoidance

161



Exposed Node Problem. Although B and C are exposed to each other's signals, there is no interference if B transmits to A while C transmits to D. (A and D's reaches are not shown.)

IEEE 802.11 – Collision Avoidance

162

- 802.11 addresses these two problems with an algorithm called Multiple Access with Collision Avoidance (**MACA**).
- Key Idea
 - Sender and receiver exchange control frames with each other before the sender actually transmits any data.
 - This exchange informs all nearby nodes that a transmission is about to begin
 - Sender transmits a *Request to Send* (**RTS**) frame to the receiver.
 - The RTS frame includes a field that indicates how long the sender wants to hold the medium
 - Length of the data frame to be transmitted
 - Receiver replies with a *Clear to Send* (**CTS**) frame
 - This frame echoes this length field back to the sender

IEEE 802.11 – Collision Avoidance

163

- Any node that sees the CTS frame knows that
 - it is close to the receiver, therefore
 - cannot transmit for the period of time it takes to send a frame of the specified length
- Any node that sees the RTS frame but not the CTS frame
 - is not close enough to the receiver to interfere with it, and
 - so is free to transmit

IEEE 802.11 – Collision Avoidance

164

- Using ACK in MACA
 - Proposed in MACAW: MACA for Wireless LANs
- Receiver sends an ACK to the sender after successfully receiving a frame
- All nodes must wait for this ACK before trying to transmit
- If two or more nodes detect an idle link and try to transmit an RTS frame at the same time
 - Their RTS frame will collide with each other
- 802.11 does not support collision detection
 - So the senders realize the collision has happened when they do not receive the CTS frame after **a period of time**
 - In this case, they each wait a random amount of time before trying again.
 - The amount of time a given node delays is defined by the same *exponential backoff* algorithm used on the Ethernet.

IEEE 802.11 – Distribution System

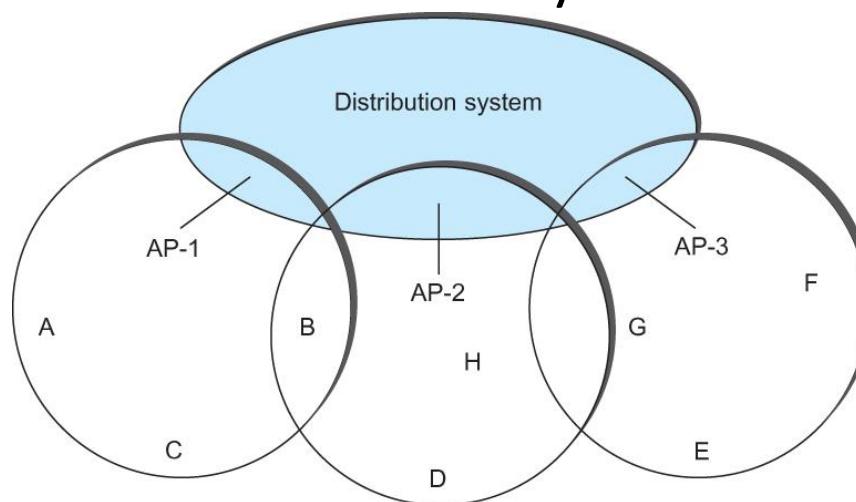
165

- 802.11 is suitable for an ad-hoc configuration of nodes that may or may not be able to communicate with all other nodes.
- Nodes are free to move around
- The set of directly reachable nodes may change over time
- To deal with this mobility and partial connectivity,
 - 802.11 defines additional structures on a set of nodes
 - Instead of all nodes being created equal,
 - some nodes are allowed to roam
 - some are connected to a wired network infrastructure
 - they are called *Access Points* (AP) and they are connected to each other by a so-called *distribution system*

IEEE 802.11 – Distribution System

166

- Following figure illustrates a distribution system that connects three access points, each of which services the nodes in the same region
- Each of these regions is analogous to a cell in a cellular phone system with the APs playing the same role as a base station
- The distribution network runs at layer 2 of the ISO architecture

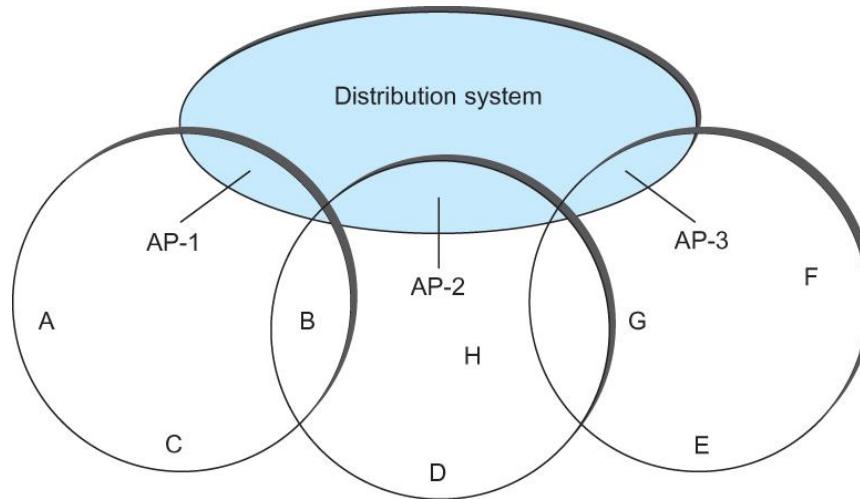


Access points connected to a distribution network

IEEE 802.11 – Distribution System

167

- Although two nodes can communicate directly with each other if they are within reach of each other, the idea behind this configuration is
 - Each node associates itself with one access point
 - For node A to communicate with node E, A first sends a frame to its AP-1 which forwards the frame across the distribution system to AP-3, which finally transmits the frame to E



Access points connected to a distribution network

IEEE 802.11 – Distribution System

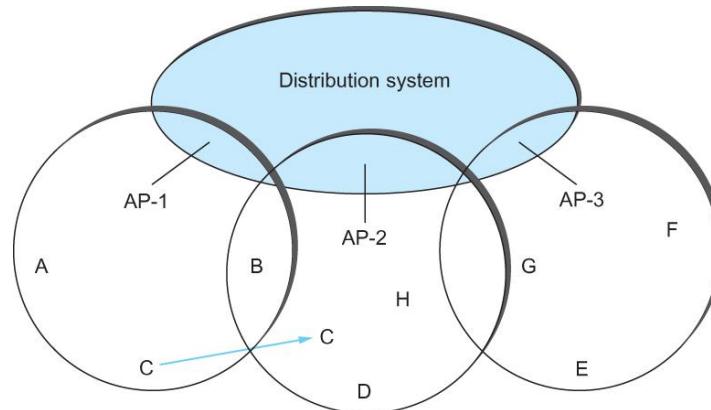
168

- How do the nodes select their access points
- How does it work when nodes move from one cell to another
- The technique for selecting an AP is called *scanning*
 - The node sends a *Probe* frame
 - All APs within reach reply with a *Probe Response* frame
 - The node selects one of the access points and sends that AP an *Association Request* frame
 - The AP replies with an *Association Response* frame
- A node engages this protocol whenever
 - it joins the network, as well as
 - when it becomes unhappy with its current AP
 - This might happen, for example, because the signal from its current AP has weakened due to the node moving away from it
 - Whenever a node acquires a new AP, the new AP notifies the old AP of the change via the distribution system

IEEE 802.11 – Distribution System

169

- Consider the situation shown in the following figure when node C moves from the cell serviced by AP-1 to the cell serviced by AP-2.
- As it moves, it sends *Probe* frames, which eventually result in *Probe Responses* from AP-2.
- At some point, C prefers AP-2 over AP-1 , and so it associates itself with that access point.
 - This is called *active scanning* since the node is actively searching for an access point

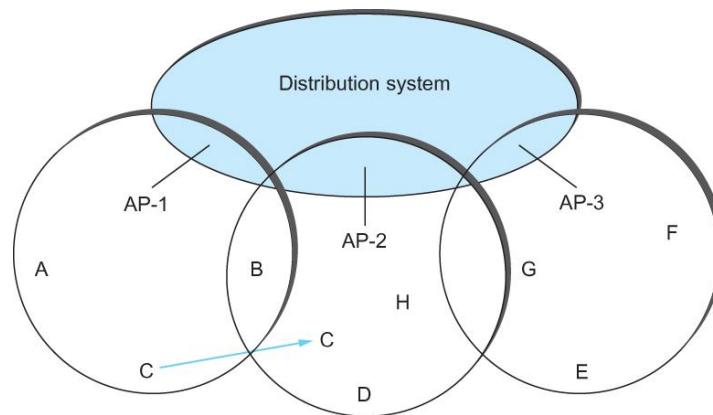


Node Mobility

IEEE 802.11 – Distribution System

170

- APs also periodically send a *Beacon* frame that advertises the capabilities of the access point; these include the transmission rate supported by the AP
 - This is called *passive scanning*
 - A node can change to this AP based on the *Beacon* frame simply by sending it an *Association Request* frame back to the access point.



Node Mobility

IEEE 802.11 – Frame Format

171

- Source and Destinations addresses: each 48 bits
- Data: up to 2312 bytes
- CRC: 32 bit
- Control field: 16 bits
 - Contains three subfields (of interest)
 - 6 bit **Type** field: indicates whether the frame is an RTS or CTS frame or being used by the scanning algorithm
 - A pair of 1 bit fields : called **ToDS** and **FromDS**



Frame Format

IEEE 802.11 – Frame Format

172

- Frame contains four addresses
- How these addresses are interpreted depends on the settings of the **ToDS** and **FromDS** bits in the frame's Control field
- This is to account for the possibility that the frame had to be forwarded across the distribution system which would mean that,
 - the original sender is not necessarily the same as the most recent transmitting node
- Same is true for the destination address
- Simplest case
 - When one node is sending directly to another, both the DS bits are 0, Addr1 identifies the target node, and Addr2 identifies the source node

IEEE 802.11 – Frame Format

173

- Most complex case
 - Both DS bits are set to 1
 - Indicates that the message went from a wireless node onto the distribution system, and then from the distribution system to another wireless node
 - With both bits set,
 - Addr1 identifies the ultimate destination,
 - Addr2 identifies the immediate sender (the one that forwarded the frame from the distribution system to the ultimate destination)
 - Addr3 identifies the intermediate destination (the one that accepted the frame from a wireless node and forwarded across the distribution system)
 - Addr4 identifies the original source
- Addr1: E, Addr2: AP-3, Addr3: AP-1, Addr4: A

Bluetooth

174

- Used for very short range communication between mobile phones, PDAs, notebook computers and other personal or peripheral devices
- Operates in the license-exempt band at 2.45 GHz
- Has a range of only 10 m
- Communication devices typically belong to one individual or group
 - Sometimes categorized as Personal Area Network (PAN)
- Version 2.0 provides speeds up to 2.1 Mbps
- Power consumption is low

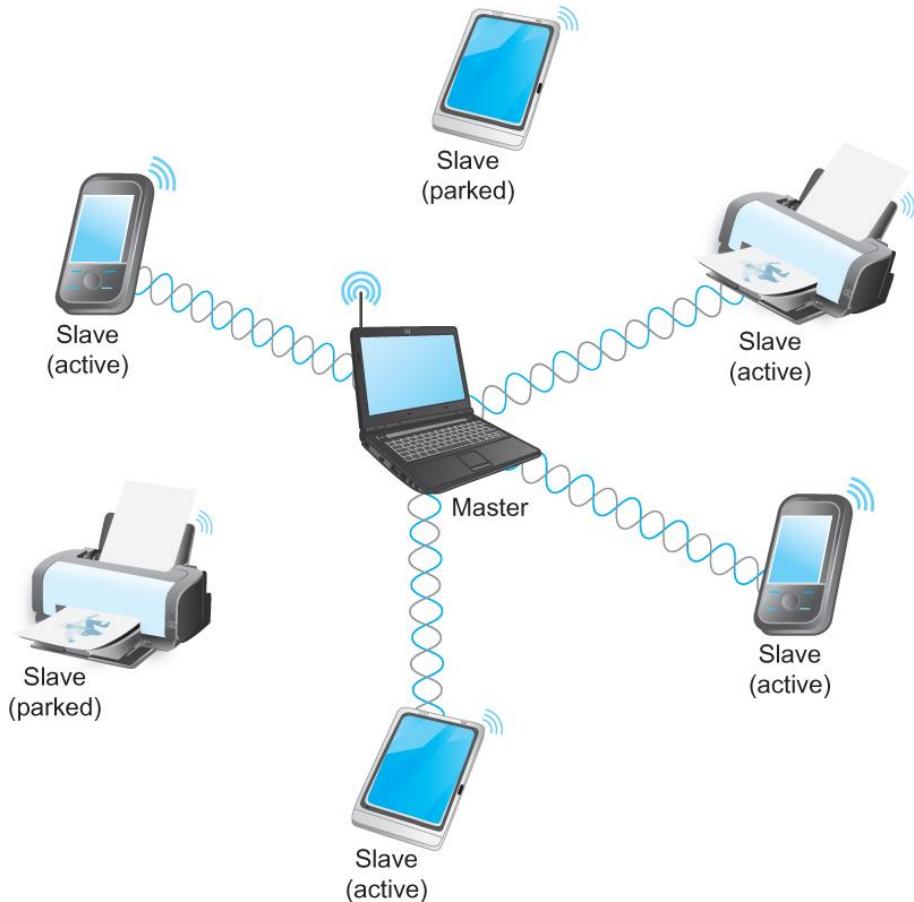
Bluetooth

175

- Bluetooth is specified by an industry consortium called the Bluetooth Special Interest Group
- It specifies an entire suite of protocols, going beyond the link layer to define application protocols, which it calls *profiles*, for a range of applications
 - There is a profile for synchronizing a PDA with personal computer
 - Another profile gives a mobile computer access to a wired LAN
- The basic Bluetooth network configuration is called a *piconet*
 - Consists of a master device and up to seven slave devices
 - Any communication is between the master and a slave
 - The slaves do not communicate directly with each other
 - A slave can be *parked*: set to an inactive, low-power state

Bluetooth

176



A Bluetooth Piconet

ZigBee

177

- ZigBee is a new technology that competes with Bluetooth
- Devised by the ZigBee alliance and standardized as IEEE 802.15.4
- It is designed for situations where the bandwidth requirements are low and power consumption must be very low to give very long battery life
- It is also intended to be simpler and cheaper than Bluetooth, making it financially feasible to incorporate in cheaper devices such as a wall switch that wirelessly communicates with a ceiling-mounted fan

Summary

178

- Introduced the many and varied type of links that are used to connect users to existing networks, and to construct large networks from scratch.
- Looked at the five key issues that must be addressed so that two or more nodes connected by some medium can exchange messages with each other
 - Encoding
 - Framing
 - Error Detecting
 - Reliability
 - Multiple Access Links
 - Ethernet
 - Wireless 802.11, Bluetooth

Reference

179

- Chapter – 2: *Computer Networks A Systems Approach* by Larry L. Peterson and Bruce S. Davie, 4Th Edition, Morgan Kaufmann Publications.

PACKET SWITCHING

20 October 2023

Dr Noor Mohammad Sk

Oultine

2

- Switching and Forwarding
- Bridges and LAN Switches
- Cell Switches (ATM)

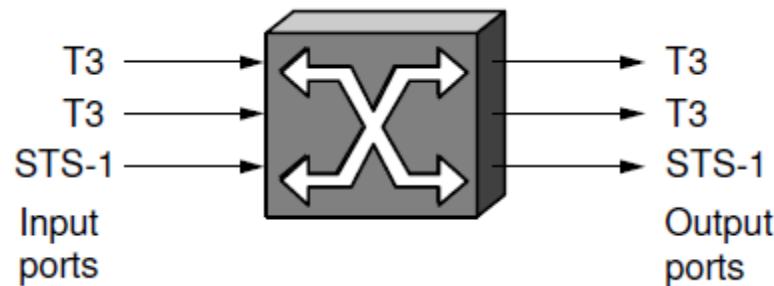
3

Switching and Forwarding

Switch

4

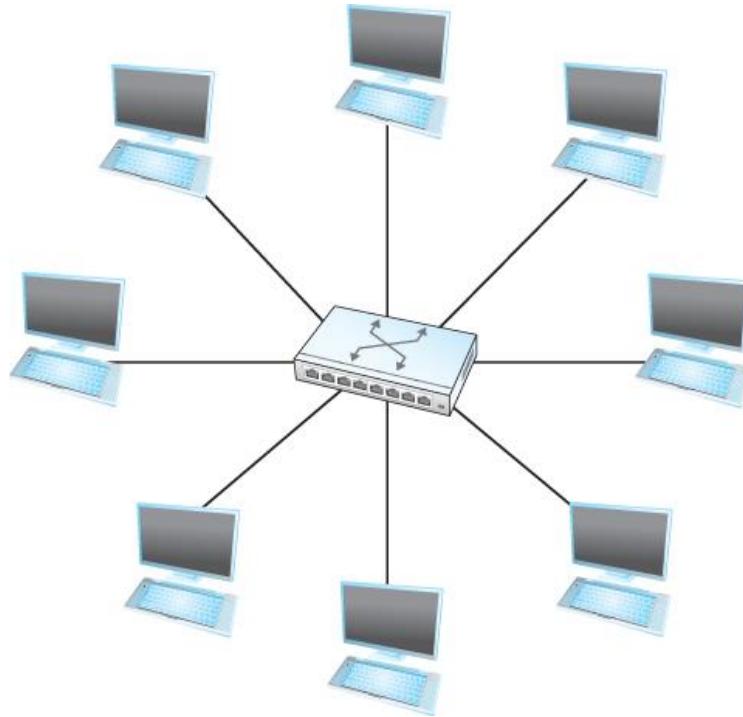
- A mechanism that allows us to interconnect links to form a large network
- A multi-input, multi-output device which transfer packets from an input to one or more outputs



Switching and Forwarding

5

- ❑ Adds the star topology to the point-to-point link, bus(Ethernet), and ring (802.5 and FDDI) topologies



Properties of Star Topology

6

- A switch has a fixed number of inputs and outputs
 - which limits the number of hosts that can be connected to a single switch
- Large networks can be build by interconnecting a number of switches
- We can connect switches to each other and to hosts using point-to-point links
 - which typically means that we can build network of large geographic scope
- Adding a new host to the network by connecting it to a switch does not necessarily mean that the hosts already connected will get worse performance from the network

Switching and Forwarding

7

- It is impossible for two hosts on the same Ethernet to transmit continuously at 10Mbps because they share the same transmission medium
- Every host on a switched network has its own link to the switch
 - So it may be entirely possible for many hosts to transmit at the full link speed (Bandwidth) provided that the switch is designed with enough aggregate capacity

Switching and Forwarding

8

- A switch is connected to a set of links and for each of these links, runs the appropriate data link protocol to communicate with that node
- A switch's primary job is to receive incoming packets on one of its links and to transmit them on some other link
 - This function is referred as *switching and forwarding*
 - According to OSI architecture this is the main function of the **network layer**

Switching and Forwarding

9

- How does the switch decide which output port to place each packet on?
 - ▣ It looks at the header of the packet for an identifier that it uses to make the decision
 - ▣ Two common approaches
 - Datagram or Connectionless approach
 - Virtual circuit or Connection-oriented approach
 - ▣ A third approach source routing is less common

Switching and Forwarding

10

- Assumptions
 - Each host has a globally unique address
 - There is some way to identify the input and output ports of each switch
 - We can use numbers
 - We can use names

Switching and Forwarding

11

- Datagrams

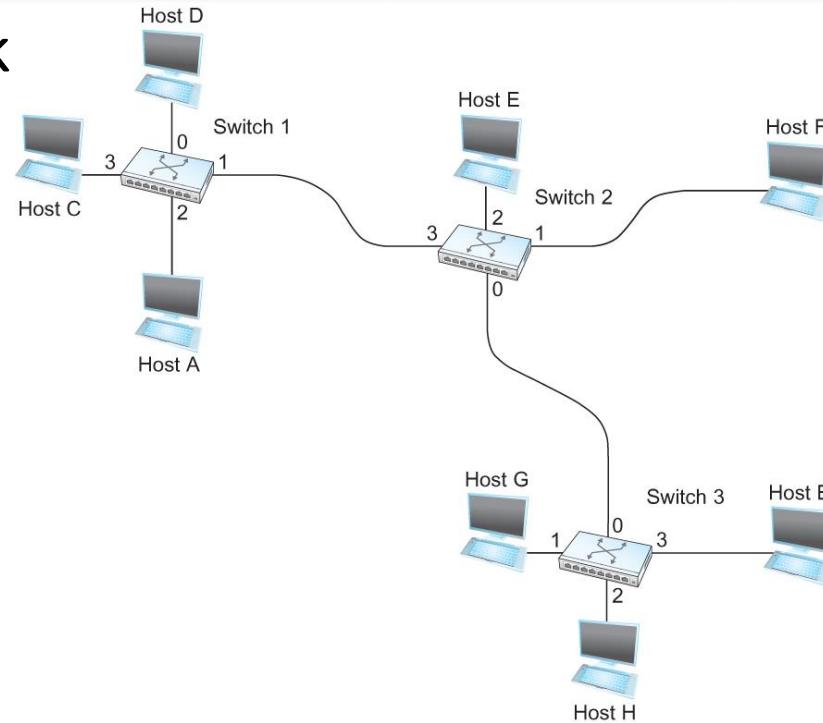
- Key Idea

- Every packet contains enough information to enable any switch to decide how to get it to destination
 - Every packet contains the complete destination address

Switching and Forwarding

12

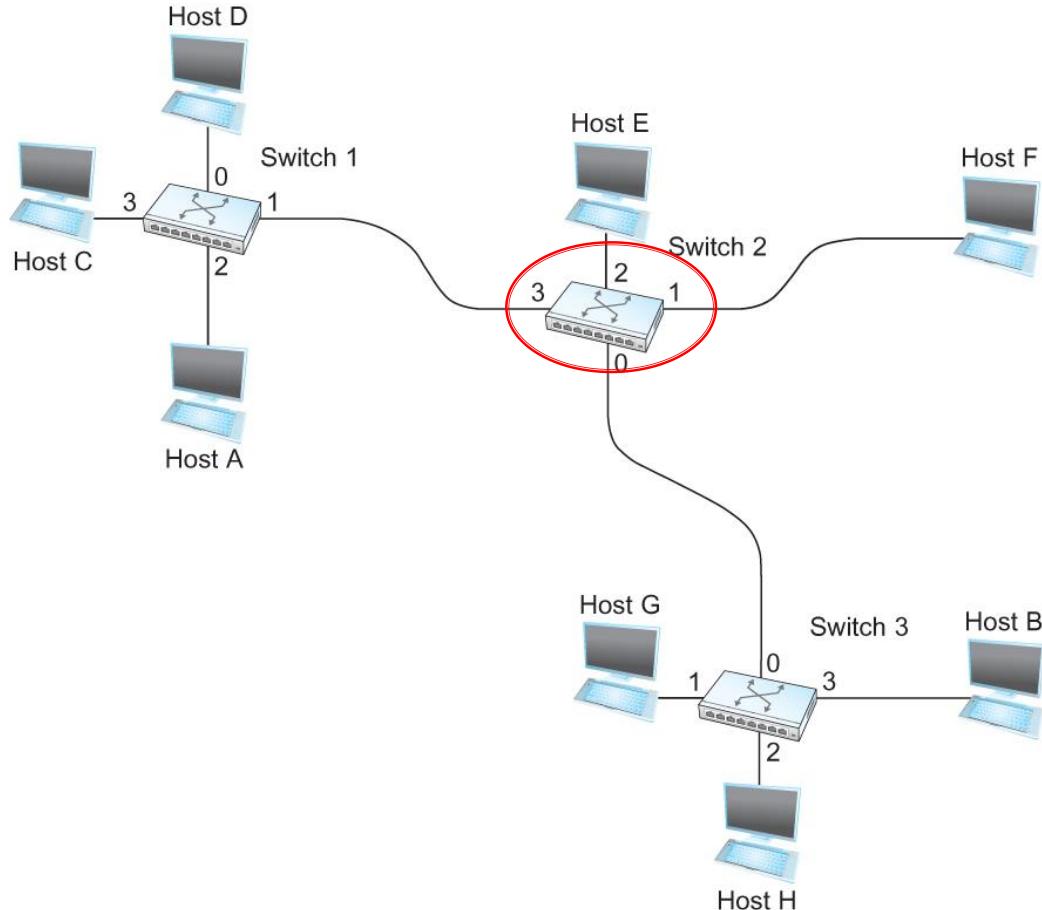
- An example network



- To decide how to forward a packet, a switch consults a *forwarding table* (sometimes called a *routing table*)

Switching and Forwarding

13



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

Forwarding Table for Switch 2

Characteristics of Connectionless (datagram) Network

14

- A host can send a packet anywhere at any time, since any packet that turns up at the switch can be immediately forwarded (assuming a correctly populated forwarding table)
- When a host sends a packet, it has no way of knowing if the network is capable of delivering it or if the destination host even up and running
- Each packet is forwarded independently of previous packets that might 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 have any serious effect on communication if it is possible to find an alternate route around the failure and update the forwarding table accordingly.

Switching and Forwarding

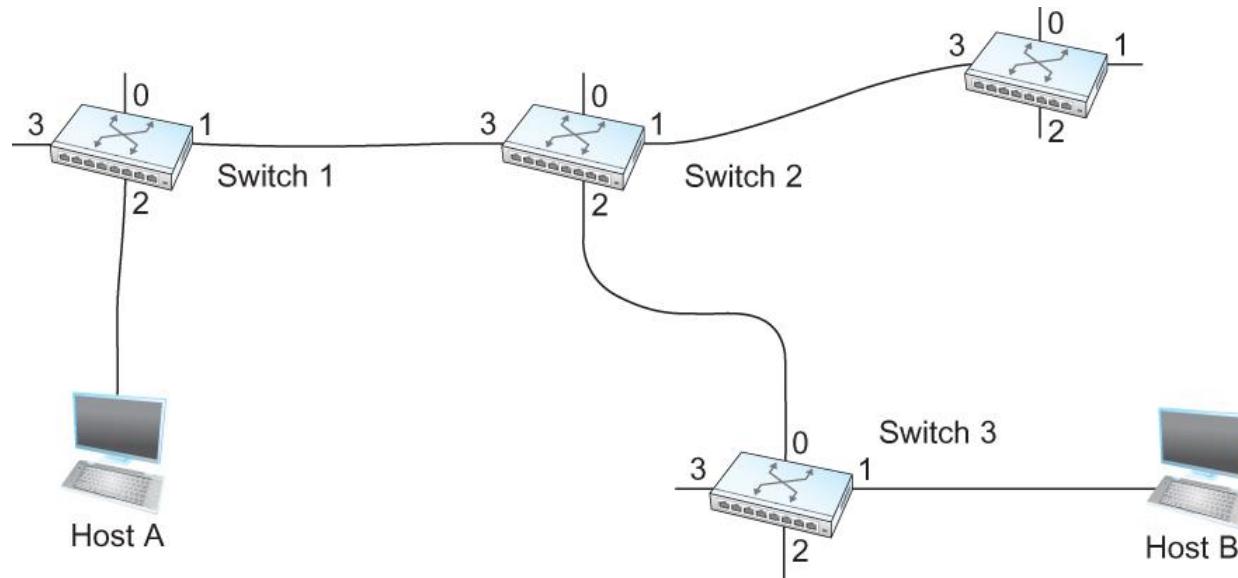
15

- Virtual Circuit Switching
 - Widely used technique for packet switching
 - Uses the concept of Virtual circuit (VC)
 - Also called a connection oriented model
 - First set up a virtual connection from the source host to the destination host and then send the data

Switching and Forwarding

16

- Host A wants to send packets to host B



Switching and Forwarding

17

- Two- stage process
 - ▣ Connection setup
 - ▣ Data Transfer
- Connection Setup
 - ▣ Establish “*connection state*” in each of the switches between the source and destination hosts
 - ▣ The connection state for a single connection consists of an entry in the “*VC table*” in each switch through which the connection passes

Switching and Forwarding

18

- 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 that will be carried inside the header of the packets that belong to this connection
 - ▣ An incoming interface on which packets for this VC arrive at the switch
 - ▣ An outgoing interface in which packets for this VC leave this switch
 - ▣ A potentially different VCI that will be used for outgoing packets

Switching and Forwarding

19

- The semantics for one such entry is
 - 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

Switching and Forwarding

20

- Note:
 - 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 switch at one time
 - Incoming and outgoing VCI values are not generally the same
 - VCI is not a globally significant identifier for the connection; rather it has significance only on a given link
 - Whenever a *new connection* is created, we need to assign *a new 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

Switching and Forwarding

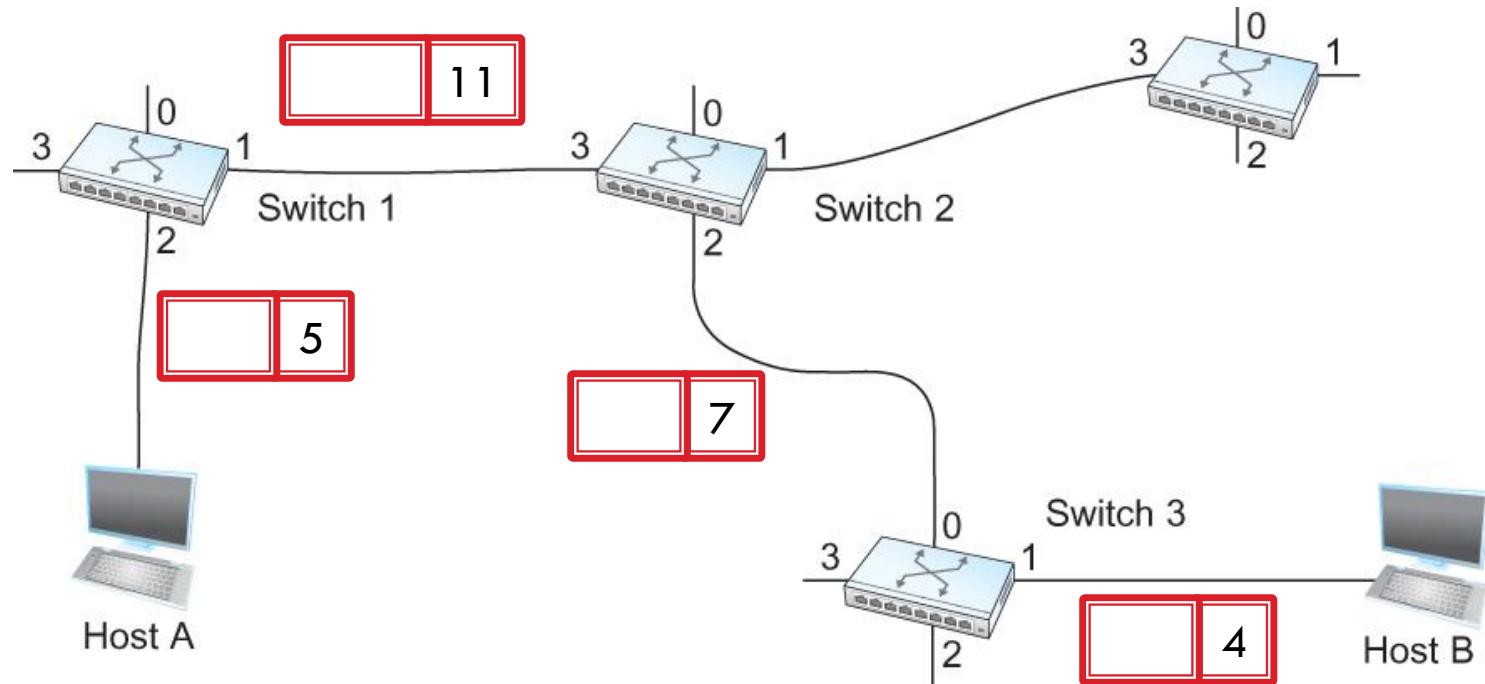
21

- Two broad classes of approach to establish connection state
 - Network Administrator will configure the state
 - The virtual circuit is **permanent** (PVC)
 - The network administrator can delete this
 - Can be thought of as a long-lived or administratively configured VC
 - A host can send message into the network to cause the state to be established
 - This is referred as **signalling** and the resulting virtual circuit is said to be **switched** (SVC)
 - A host may set up and delete such a VC dynamically without the involvement of a network administrator

Switching and Forwarding

22

- Let's 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

23

- 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
 - 11 is chosen for the link from switch 1 to switch 2
 - So the switch 1 will have an entry in the VC table

Incoming Interface	Incoming VC	Outgoing Interface	Outgoing VC
2	5	1	11

Switching and Forwarding

24

- Similarly, suppose
 - 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
 - Switches 2 and 3 are configured with the following VC tables

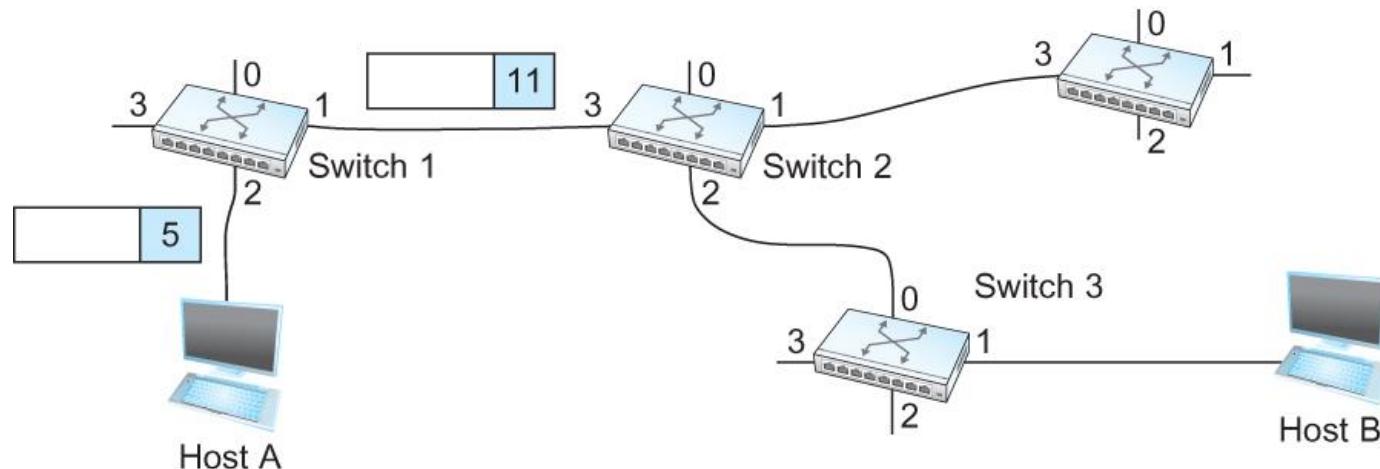
Incoming Interface	Incoming VC	Outgoing Interface	Outgoing VC
3	11	2	7

Incoming Interface	Incoming VC	Outgoing Interface	Outgoing VC
0	7	1	4

Switching and Forwarding

25

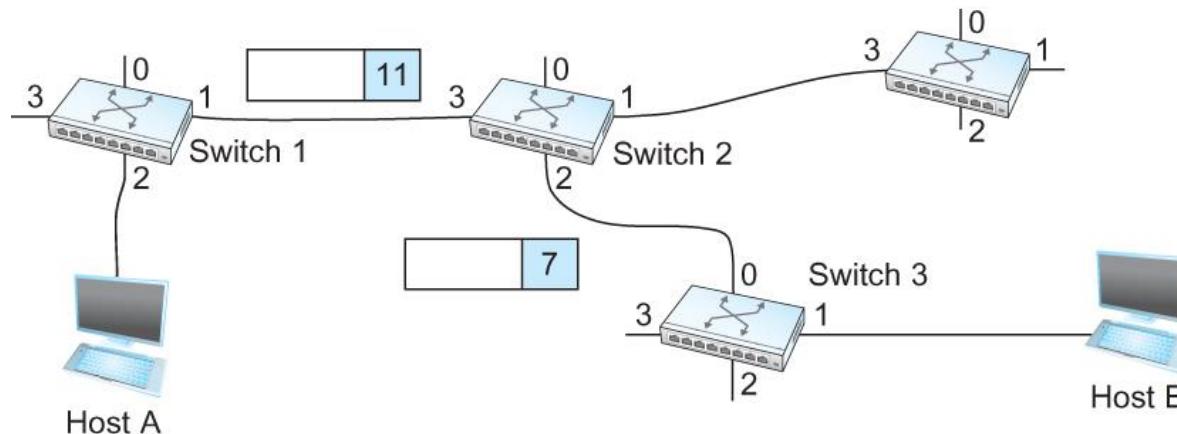
- For any packet that A wants to send to B, A puts the VCI value 5 in the header of the packet and sends it to switch 1
- Switch 1 receives any such packet on interface 2, and it uses the combination of the interface and the VCI in the packet header to find the appropriate VC table entry.
- The table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



Switching and Forwarding

26

- 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



Switching and Forwarding

27

- In real networks of reasonable size, the burden of configuring VC tables correctly in a large number of switches would quickly become excessive
 - Thus, some sort of signalling is almost always used, even when setting up “permanent” VCs
 - In case of PVCs, signalling is initiated by the network administrator
 - Switched Virtual Circuits (SVCs) are usually set up using signalling by one of the hosts

Switching and Forwarding

28

- How does the signalling work
 - To start the signalling process, host A sends a setup message into the network (i.e. to switch 1)
 - The setup message contains (among other things) the complete destination address of B.
 - The setup message needs to get all the way to B to create the necessary connection state in every switch along the way
 - It is like sending a datagram to B where every switch knows which output to send the setup message so that it eventually reaches B
 - Assume that every switch knows the topology to figure out how to do that
 - When switch 1 receives the connection request, in addition to sending it on to switch 2, it creates a new entry in its VC table for this new connection
 - The entry is exactly the same shown in the previous table
 - Switch 1 picks the value 5 for this connection

Switching and Forwarding

29

- How does the signalling work (contd.)
 - When switch 2 receives the setup message, it performs the similar process and it picks the value 11 as the incoming VCI
 - Similarly switch 3 picks 7 as the value for its incoming VCI
 - Each switch can pick any number it likes, as long as that number is not currently in use for some other connection on that port of that switch
 - Finally the setup message arrives at host B.
 - Assuming that B is healthy and willing to accept a connection from host A, it allocates an incoming VCI value, in this case 4.
 - This VCI value can be used by B to identify all packets coming from A

Switching and Forwarding

30

- Now to complete the connection, everyone needs to be told what their downstream neighbor is using as the VCI for this connection
 - Host B sends an acknowledgement of the connection setup to switch 3 and includes in that message the VCI value that it chose (4)
 - Switch 3 completes the VC table entry for this connection and sends the acknowledgement on to switch 2 specifying the VCI of 7
 - Switch 2 completes the VC table entry for this connection and sends acknowledgement on to switch 1 specifying the VCI of 11
 - Finally switch 1 passes the acknowledgement on to host A telling it to use the VCI value of 5 for this connection

Switching and Forwarding

31

- When host A **no longer wants to send data** to host B, it tears down the connection by sending a teardown message to switch 1
- The switch 1 **removes the relevant entry** from its table and forwards the message on to the other switches in the path which similarly delete the appropriate table entries
- At this point, if host A were to send a packet with a VCI of 5 to switch 1, it would be dropped as if the **connection had never existed**

Switching and Forwarding

32

- Characteristics of VC
 - Since host A has to wait for the connection request to reach the far side of the network and return before it can send its first data packet, there is **at least one RTT of delay** before data is sent
 - While the connection request contains the full address for host B (which might be quite large, being a global identifier on the network), each data packet contains only a small identifier, which is only unique on one link.
 - Thus the per-packet overhead caused by the header is reduced relative to the datagram model
 - If a switch or a link in a connection fails, the connection is broken and a new one will need to be established.
 - Also the old one needs to be torn down to free up table storage space in the switches
 - The issue of how a switch decides which link to forward the connection request on has similarities with the function of a routing algorithm

Switching and Forwarding

33

- Good Properties of VC
 - By the time the host gets the go-ahead to send data, it knows quite a lot about the network-
 - For example, that there is really a route to the receiver and that the receiver is willing to receive data
 - It is also possible to allocate resources to the virtual circuit at the time it is established

Switching and Forwarding

34

- For example, an X.25 network – a packet-switched network that uses the connection-oriented model – employs the following three-part strategy
 - Buffers are allocated to each virtual circuit when the circuit is initialized
 - The sliding window protocol is run between each pair of nodes along the virtual circuit, and this protocol is augmented with the flow control to keep the sending node from overrunning the buffers allocated at the receiving node
 - The circuit is rejected by a given node if not enough buffers are available at that node when the connection request message is processed

Switching and Forwarding

35

- Comparison with the Datagram Model
 - Datagram network has no connection establishment phase and each switch processes each packet independently
 - Each arriving packet competes with all other packets for buffer space
 - If there are no buffers, the incoming packet must be dropped
- In VC, we could imagine providing each circuit with a different quality of service (QoS)
 - The network gives the user some kind of performance related guarantee
 - Switches set aside the resources they need to meet this guarantee
 - For example, a percentage of each outgoing link's bandwidth
 - Delay tolerance on each switch
- Most popular examples of VC technologies are Frame Relay and ATM
 - One of the applications of Frame Relay is the construction of VPN

ATM

36

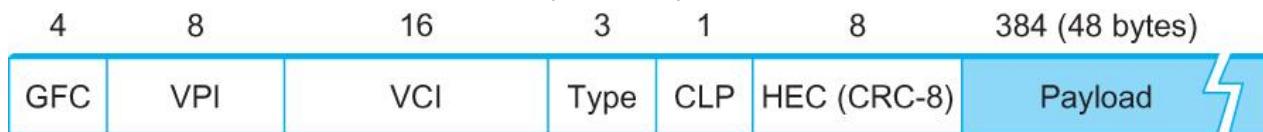
- ATM (Asynchronous Transfer Mode)
 - Connection-oriented packet-switched network
 - Packets are called cells
 - 5 byte header + 48 byte payload
 - Fixed length packets are easier to switch in hardware
 - Simpler to design
 - Enables parallelism

ATM

37

□ User – Network Interface (UNI)

- Host-to-switch format
- GFC: Generic Flow Control
- VPI: Virtual Path Identifier
- VCI: Virtual Circuit Identifier
- Type: management, congestion control
- CLP: Cell Loss Priority
- HEC: Header Error Check (CRC-8)



□ Network – Network Interface (NNI)

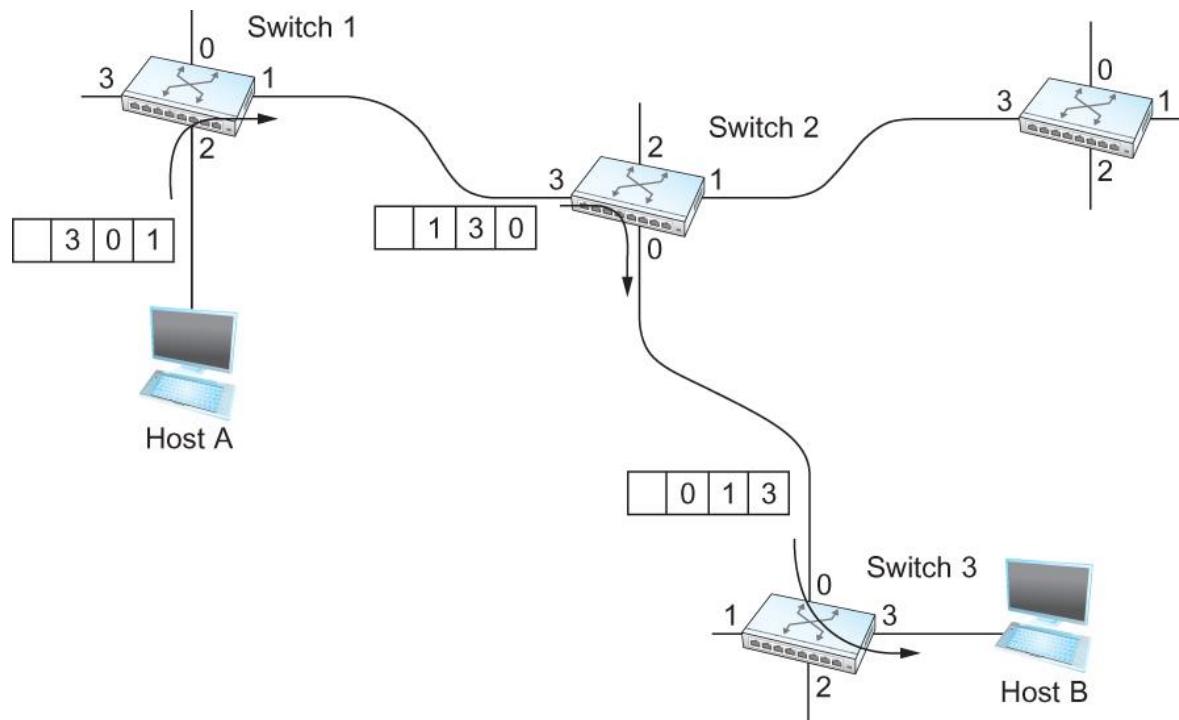
ATM Cell Format

- Switch-to-switch format
- GFC becomes part of VPI field

Source Routing

38

- All the information about network topology that is required to switch a packet across the network is provided by the source host



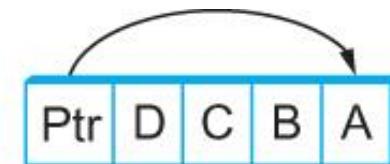
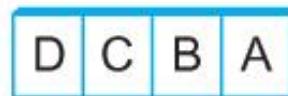
Source Routing

39

- Other approaches in source routing
- Three ways to handle headers for source routing
 - a) Rotation
 - b) Stripping
 - c) Pointer

The labels are read right to left

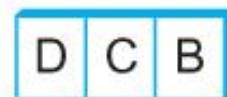
Header entering switch



Header leaving switch



(a)



(b)



(c)

Bridges and LAN Switches

Bridges and LAN Switches

41

- Class of switches that is used to forward packets between shared-media LANs such as Ethernets
 - Known as LAN switches
 - Referred to as Bridges
- Suppose you have a pair of Ethernets that you want to interconnect
 - One approach is put a repeater in between them
 - It might exceed the physical limitation of the Ethernet
 - No more than four repeaters between any pair of hosts
 - No more than a total of 2500m in length is allowed

Bridges and LAN Switches

42

- An alternative would be to put a node between the two Ethernets and have the node forward frames from one Ethernet to the other
 - This node is called a bridge
 - A collection of LANs connected by one or more bridges is usually said to form an **extended LAN**

Bridges and LAN Switches

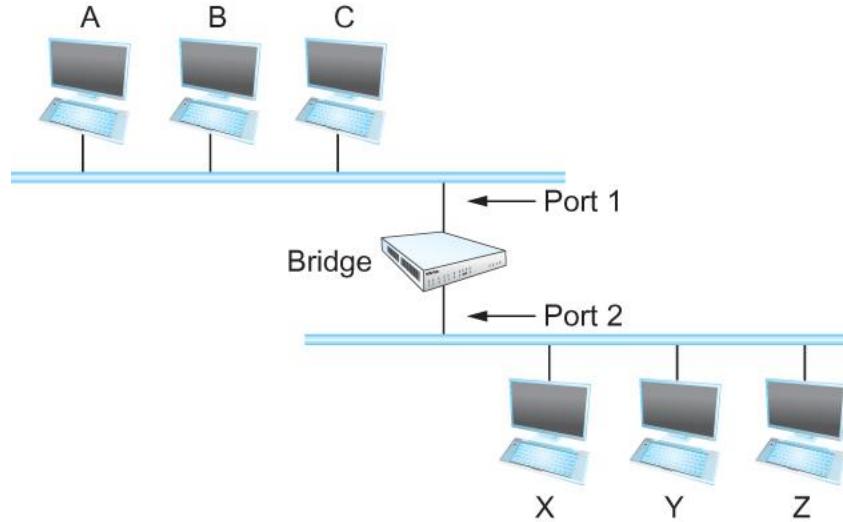
43

- Simplest Strategy for Bridges
 - ▣ Accept LAN frames on their inputs and forward them out to all other outputs
 - ▣ Used by early bridges
- Learning Bridges
 - ▣ Observe that there is no need to forward all the frames that a bridge receive

Bridges and LAN Switches

44

- Consider the following figure
 - When a frame from host A that is addressed to host B arrives on port 1 there is no need for the bridge to forward the frame out over port 2



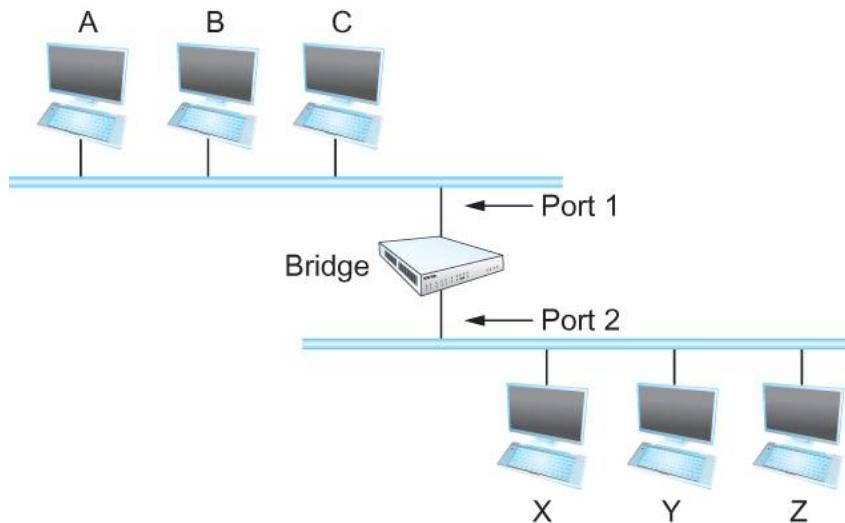
- How does a bridge come to learn on which port the various hosts reside?

Bridges and LAN Switches

45

- Solution

- Download a table into the bridge



Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

- Who does the download
 - Human
 - Too much work for maintenance

Bridges and LAN Switches

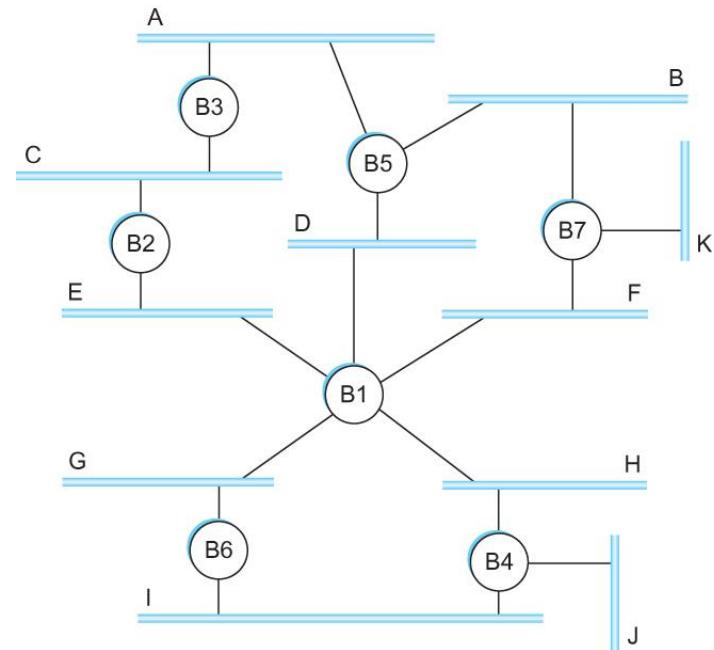
46

- Can the bridge learn this information by itself?
 - Yes
- How
 - Each bridge inspects the source address in all the frames it receives
 - Record the information at the bridge and build the table
 - Entries are added over time
 - A timeout is associated with each entry
 - The bridge discards the entry after a specified period of time
 - To protect against the situation in which a host is moved from one network to another

Bridges and LAN Switches

47

- If the bridge receives a frame that is addressed to host not currently in the table
 - ▣ Forward the frame out on all other ports
- Strategy works fine if the extended LAN does not have a loop in it
- Why?
 - ▣ Frames potentially loop through the extended LAN forever
 - ▣ Bridges B1, B4 and B6 form a loop



Bridges and LAN Switches

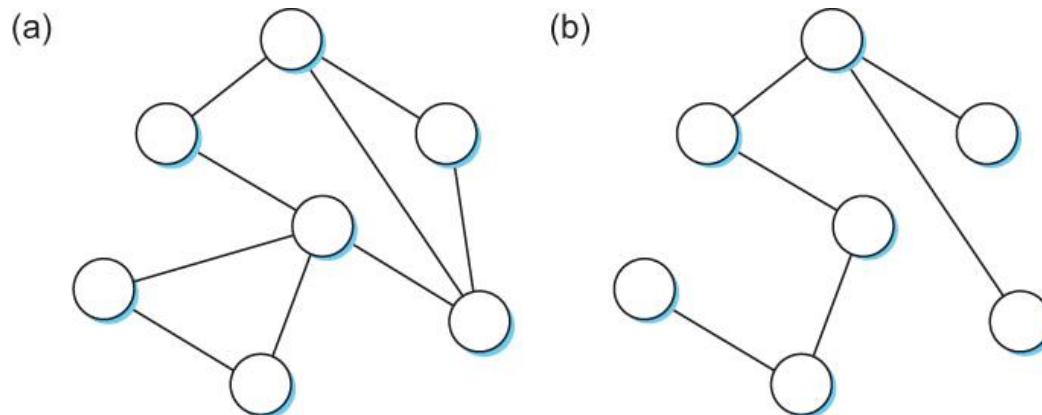
48

- How does an extended LAN come to have a loop in it?
 - Network is managed by more than one administrator
 - For example, it spans multiple departments in an organization
 - It is possible that no single person knows the entire configuration of the network
 - A bridge that closes a loop might be added without any one knowing
 - Loops are built into the network to provide redundancy in case of failures
- Solution
 - Distributed Spanning Tree Algorithm

Spanning Tree Algorithm

49

- Think of the extended LAN as being represented by a graph that possibly has loops (cycles)
- A spanning tree is a sub-graph of this graph that covers all the vertices but contains no cycles
 - Spanning tree keeps all the vertices of the original graph but throws out some of the edges



Example of (a) a cyclic graph; (b) a corresponding spanning tree.

Spanning Tree Algorithm

50

- Developed by Radia Perlman at Digital
 - ▣ a protocol used by a set of bridges to agree upon a spanning tree for a particular extended LAN
 - ▣ IEEE 802.1 specification for LAN bridges is based on this algorithm
 - ▣ Each bridge decides the ports over which it is and is not willing to forward frames
 - In a sense, it is by removing ports from the topology that the extended LAN is reduced to an acyclic tree
 - It is even possible that an entire bridge will not participate in forwarding frames

Spanning Tree Algorithm

51

- Algorithm is dynamic
 - The bridges are always prepared to reconfigure themselves into a new spanning tree if some bridges fail
- Main idea
 - Each bridge selects the ports over which they will forward the frames

Spanning Tree Algorithm

52

- Algorithm selects ports as follows:
 - Each bridge has a unique identifier
 - B1, B2, B3,...and so on.
 - Elect the bridge with the smallest id as the root of the spanning tree
 - The root bridge always forwards frames out over all of its ports
 - Each bridge computes the shortest path to the root and notes which of its ports is on this path
 - This port is selected as the bridge's preferred path to the root
 - Finally, all the bridges connected to a given LAN elect a single *designated bridge* that will be responsible for forwarding frames toward the root bridge

Spanning Tree Algorithm

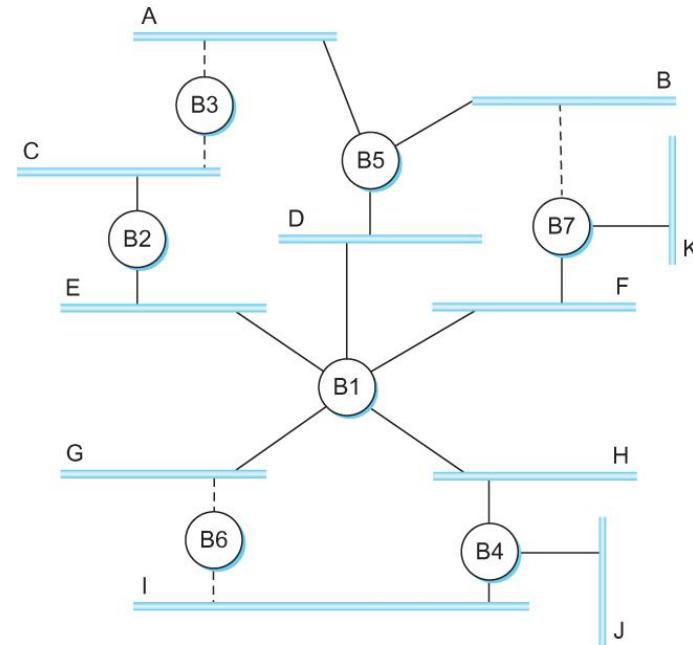
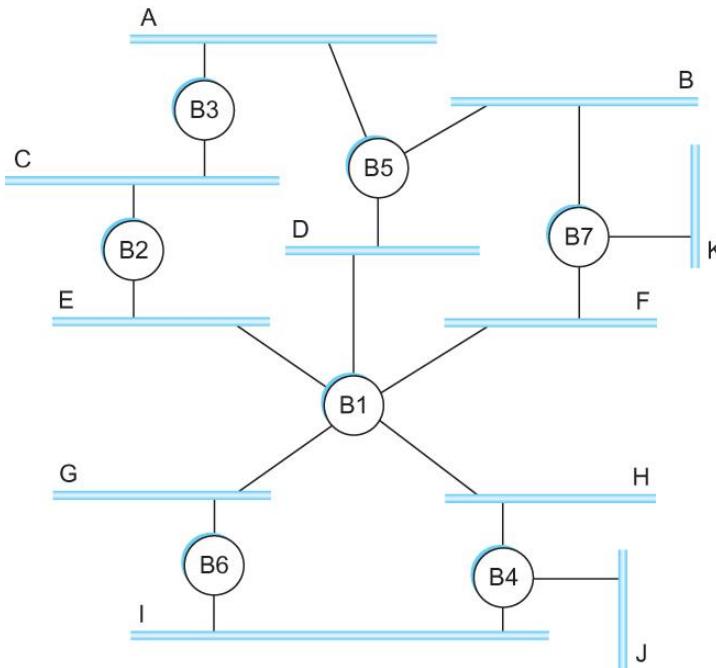
53

- Each LAN's designated bridge is the one that is closest to the root
- If two or more bridges are equally close to the root,
 - Then select bridge with the smallest id
- Each bridge is connected to more than one LAN
 - So it participates in the election of a designated bridge for each LAN it is connected to.
 - Each bridge decides if it is the designated bridge relative to each of its ports
 - The bridge forwards frames over those ports for which it is the designated bridge

Spanning Tree Algorithm

54

- B1 is the root bridge
- B3 and B5 are connected to LAN A, but B5 is the designated bridge
- B5 and B7 are connected to LAN B, but B5 is the designated bridge



Spanning Tree Algorithm

55

- Initially each bridge thinks it is the root, so it sends a configuration message on each of its ports identifying itself as the root and giving a distance to the root of 0
- Upon receiving a configuration message over a particular port, the bridge checks to see if the new message is *better* than the current best configuration message recorded for that port
- The new configuration is better than the currently recorded information if
 - It identifies a root with a smaller id or
 - It identifies a root with an equal id but with a shorter distance or
 - The root id and distance are equal, but the sending bridge has a smaller id

Spanning Tree Algorithm

56

- If the new message is better than the currently recorded one,
 - The bridge discards the old information and saves the new information
 - It first adds 1 to the distance-to-root field
- When a bridge receives a configuration message indicating that it is not the root bridge (that is, a message from a bridge with smaller id)
 - The bridge stops generating configuration messages on its own
 - Only forwards configuration messages from other bridges after 1 adding to the distance field

Spanning Tree Algorithm

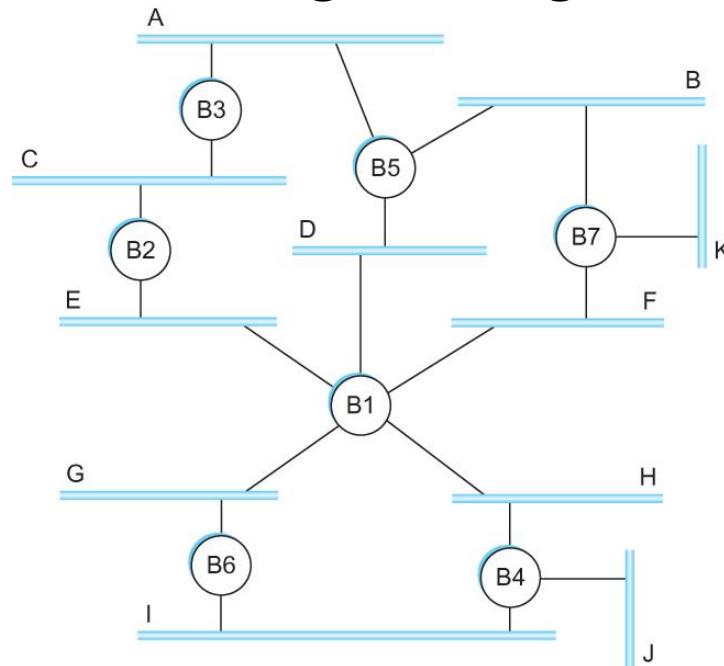
57

- When a bridge receives a configuration message that indicates it is not the designated bridge for that port
 - => a message from a bridge that is closer to the root or equally far from the root but with a smaller id
 - The bridge stops sending configuration messages over that port
- When the system stabilizes,
 - Only the root bridge is still generating configuration messages.
 - Other bridges are forwarding these messages only over ports for which they are the designated bridge

Spanning Tree Algorithm

58

- Consider the situation when the power had just been restored to the building housing the following network

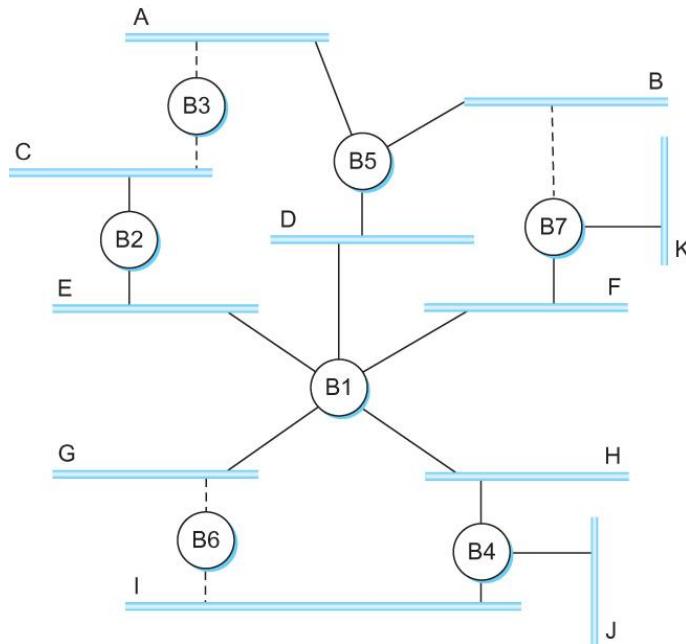


- All bridges would start off by claiming to be the root

Spanning Tree Algorithm

59

- Denote a configuration message from node X in which it claims to be distance d from the root node Y as (Y, d, X)

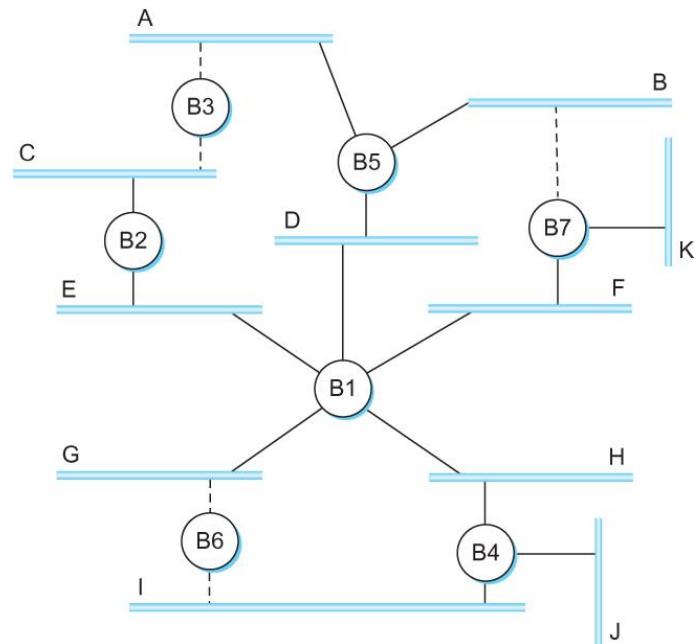


- Consider the activity at node B3

Spanning Tree Algorithm

60

- B3 receives (B2, 0, B2)
- Since $2 < 3$, B3 accepts B2 as root
- B3 adds 1 to the distance advertised by B2 and sends (B2, 1, B3) to B5
- Meanwhile B2 accepts B1 as root because it has the lower id and it sends (B1, 1, B2) toward B3
- B5 accepts B1 as root and sends (B1, 1, B5) to B3
- B3 accepts B1 as root and it notes that both B2 and B5 are closer to the root than it is.
 - Thus B3 stops forwarding messages on both its interfaces
 - This leaves B3 with both ports not selected



Spanning Tree Algorithm

61

- Even after the system has stabilized, the root bridge continues to send configuration messages periodically
 - Other bridges continue to forward these messages
- When a bridge fails, the downstream bridges will not receive the configuration messages
- After waiting a specified period of time, they will once again claim to be the root and the algorithm starts again
- Note
 - Although the algorithm is able to reconfigure the spanning tree whenever a bridge fails, it is not able to forward frames over alternative paths for the sake of routing around a congested bridge

Broadcast and Multicast

62

- Forwarding all broadcast/multicast frames
 - ▣ Current practice
- Learn when no group members downstream
- Accomplished by having each member of group G and send a frame to bridge multicast address with G in source field

Limitation of Bridges

63

- Do not scale
 - Spanning tree algorithm does not scale
 - Broadcast does not scale
- Do not accommodate heterogeneity

64

Cell Switching (ATM)

Cell Switching (ATM)

65

- 1980s and early 1990s; embraced by telephone industry
- Used in both WAN and LAN setting
- Specified by ATM forum
- Connection-oriented (virtual circuit switching), packet-switched network
 - ▣ Signalling (Connection setup) protocol: Q.2931
- Packets are called cells: 5-byte header + 48-byte payload
- Commonly transmitted over SONET
 - ▣ Other physical layers possible

Variable Vs Fixed-Length Packets

66

- No optimal length => variable-length packet
 - ▣ If small: high header-to-data overhead
 - ▣ If large: low utilization for small messages
- Fixed-Length Easier to Switch in Hardware
 - ▣ Simpler to implement
 - ▣ Enables parallelism (since length is known and fixed)

Big Vs Small Packets

67

- Small Improves Queue behavior
 - finer-grained preemption point for scheduling link
 - maximum packet = 4KB
 - Link speed = 100Mbps
 - transmission time = $4096 \times 8/100 = 327.68\text{us}$
 - High priority packet may sit in the queue 327.68us
 - In contrast, $53 \times 8/100 = 4.24\text{us}$ for ATM
 - near cut-through behavior
 - two 4KB packets arrive at same time
 - link idle for 327.68us while both arrive
 - Because the switch must wait to receive the whole first packet before starting transmitting it
 - at end of 327.68us, still have 8KB to transmit
 - in contrast, can transmit first cell after 4.24us
 - at end of 327.68us, just over 4KB left in queue

Big Vs Small Packets

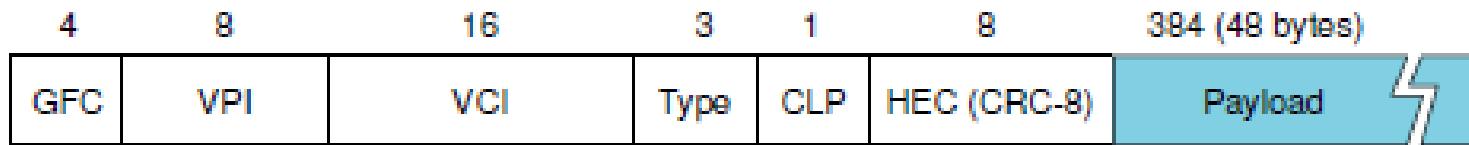
68

- Small Reduces Latency (for voice)
 - voice digitally encoded at 64KBps (8-bit samples at 8KHz)
 - need full cell's worth of samples before sending cell
 - example: 1000-byte cells implies 125ms per cell (too long)
 - smaller latency implies no need for echo cancellers (since a very small latency feels like “0 latency”)
- ATM Compromise: 48 bytes
 - US: would like it to be 64 bytes (has echo canceller, thus can afford large packets to reduce header-to-payload ratio)
 - Europe: advocates 32 bytes (no echo canceller, thus need small packet)
 - Compromise: $48 = (32+64)/2$

Cell Format

69

- User-Network Interface (UNI): host-to-switch format

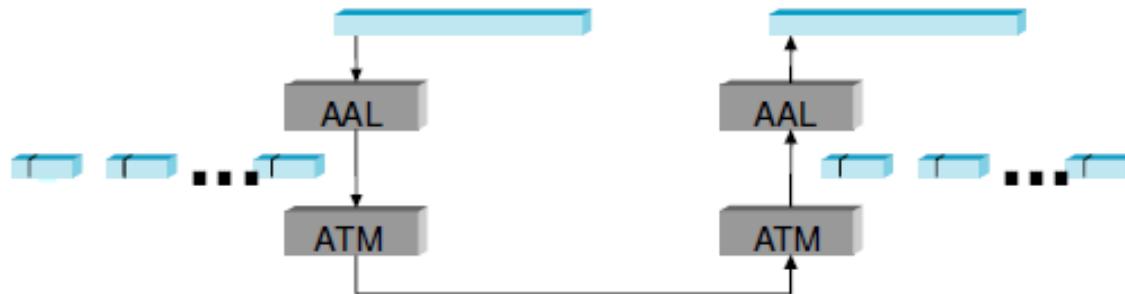


- GFC: Generic Flow Control (still being defined)
- VPI: Virtual Path Identifier
- VCI: Virtual Circuit Identifier
- Type: management, congestion control, AAL5 (later)
- CLP: Cell Loss Priority
- HEC: Header Error Check (CRC-8)
- Network-Network Interface (NNI): switch-to-switch format
 - GFC becomes part of VPI field

Segmentation and Reassembly

70

- Accomplished by ATM Adaptation Layer (AAL)



- Two sub layers:
 - Higher layer: Convergence Sub layer (CS)
 - responsible for packaging the higher layer PDU (protocol data unit) with additional information required for the adaptation necessary for specific service types (e.g., bit rate, connection-oriented or connectionless)
 - Lower layer: Segmentation and Reassembly (SAR)

Different Types of AALs

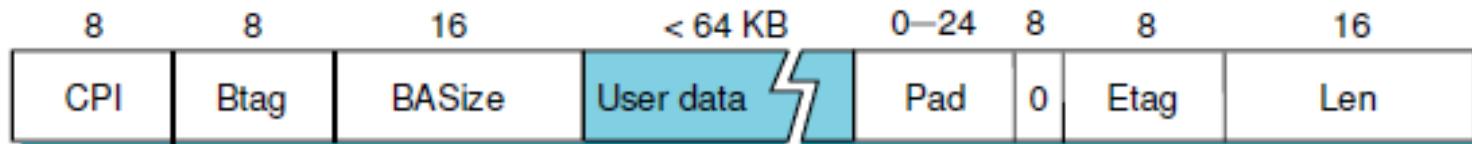
71

- AAL 1 and 2 designed for applications that need guaranteed rate (e.g., voice, video)
- AAL 3/4 designed for data packet
- AAL 5 is an alternative standard for data packet

AAL 3/4

72

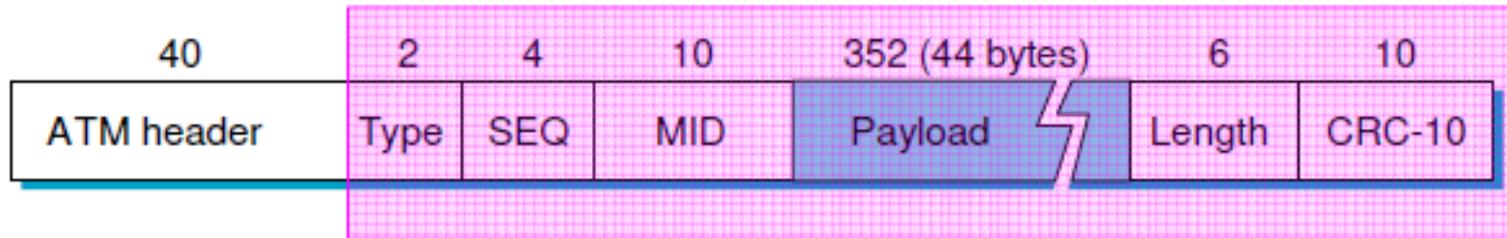
□ Convergence Sub layer Protocol Data Unit (CS-PDU)



- CPI: common part indicator (version field); not used yet
- Btag/Etag: beginning and ending tag (deal with cell corruption)
- BASize: hint on amount of buffer space to allocate
- Length: size of whole PDU

SAR: Cell Format --- payload

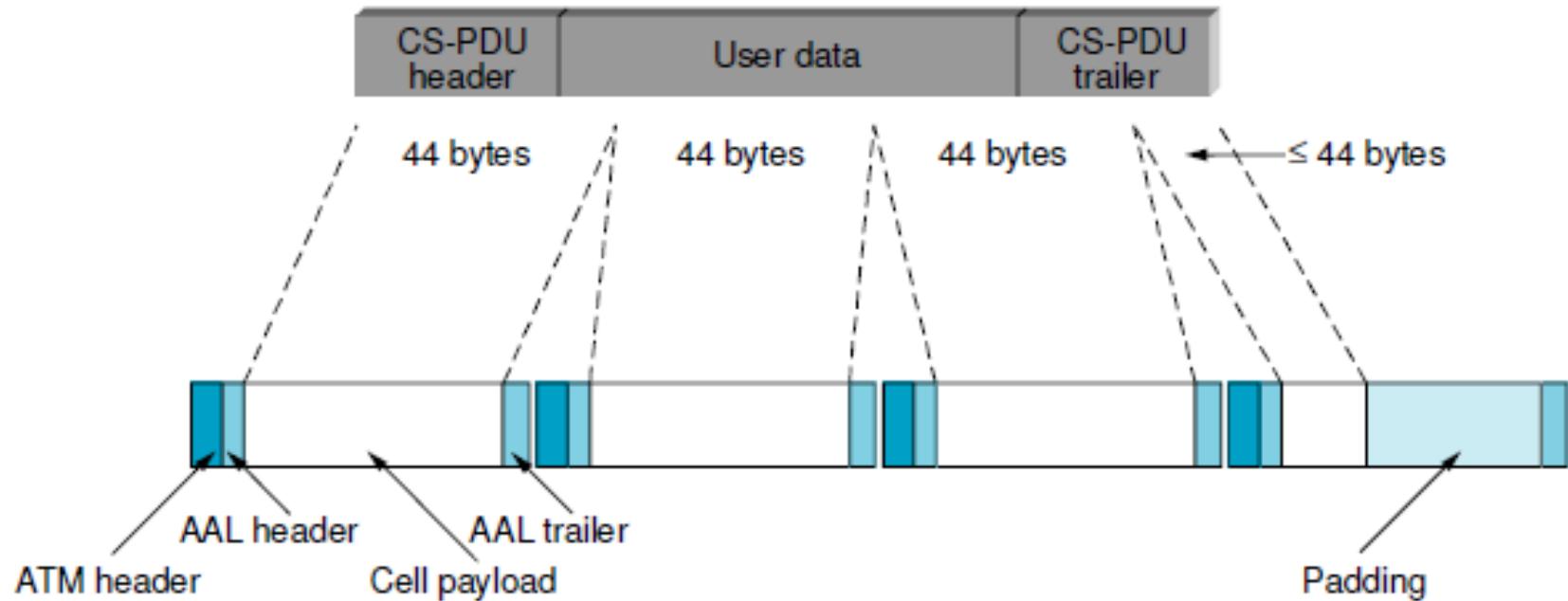
73



- Type
 - BOM: beginning of message
 - COM: continuation of message
 - EOM end of message
- SEQ: sequence number
- MID: multiplexing identifier
- Length: number of bytes of payload (from CS-PDU) in this cell; in bytes

Encapsulation & segmentation for AAL3/4

74

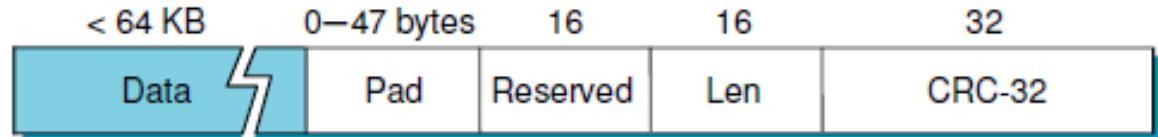


AAL5

75

- ❑ Simplifies the format of AAL $\frac{3}{4}$

- ❑ CS-PDU Format

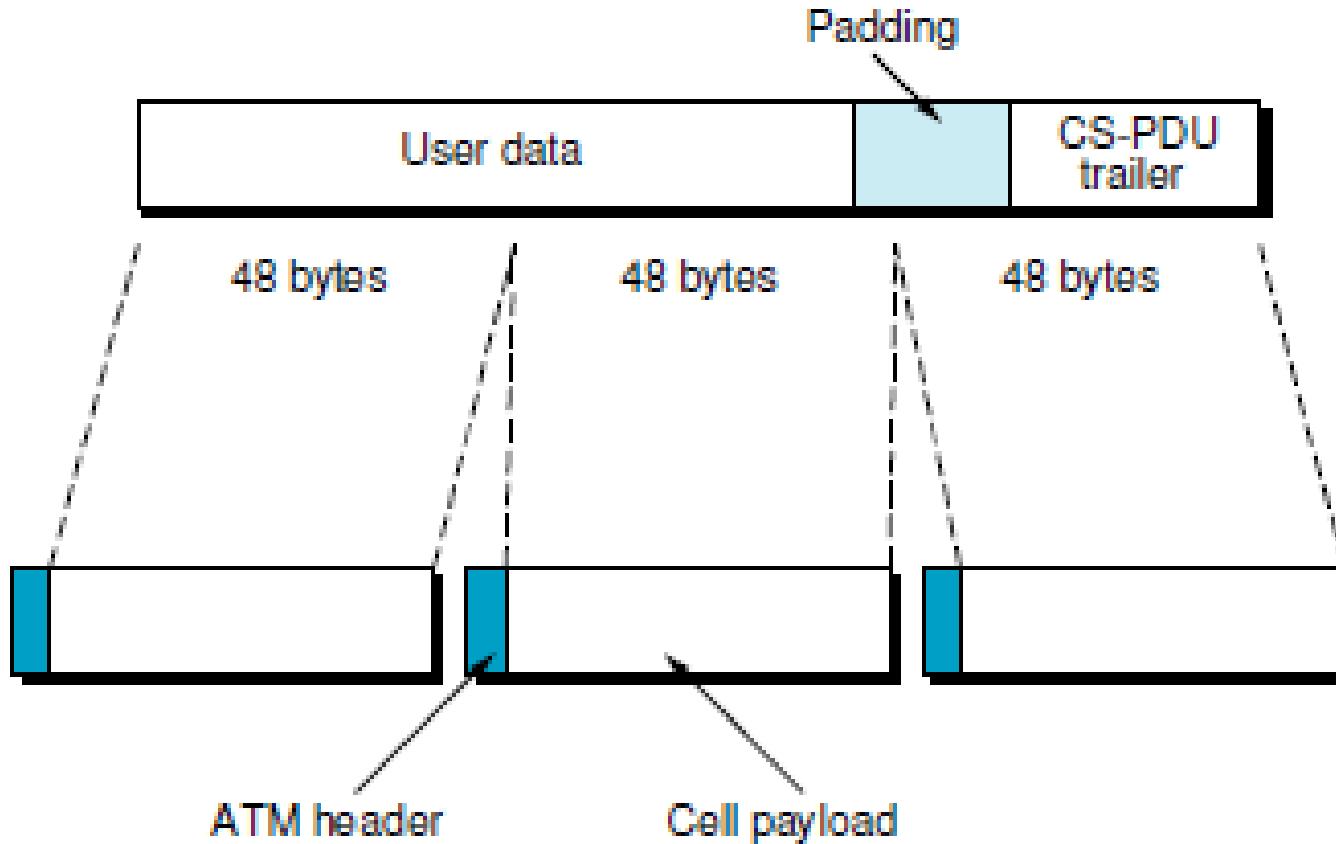


- ❑ Pad, so trailer always falls at end of ATM cell
- ❑ Len: size of PDU (data only); in bytes
- ❑ CRC-32 (detects missing or misordered cells)
- ❑ SAR: Cell Format --- payload
 - ❑ **end-of-PDU** bit in Type field of ATM header
 - ❑ Compared with AAL 3/4 , AAL 5 does not provide an additional level of multiplexing onto one virtual circuit (which was achieved via MID in AAL $\frac{3}{4}$)

Encapsulation & segmentation for AAL

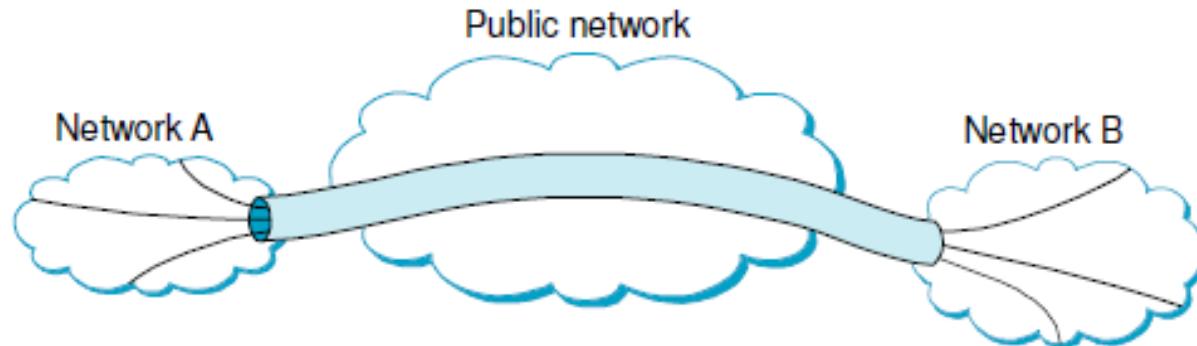
5

76



Virtual path

77



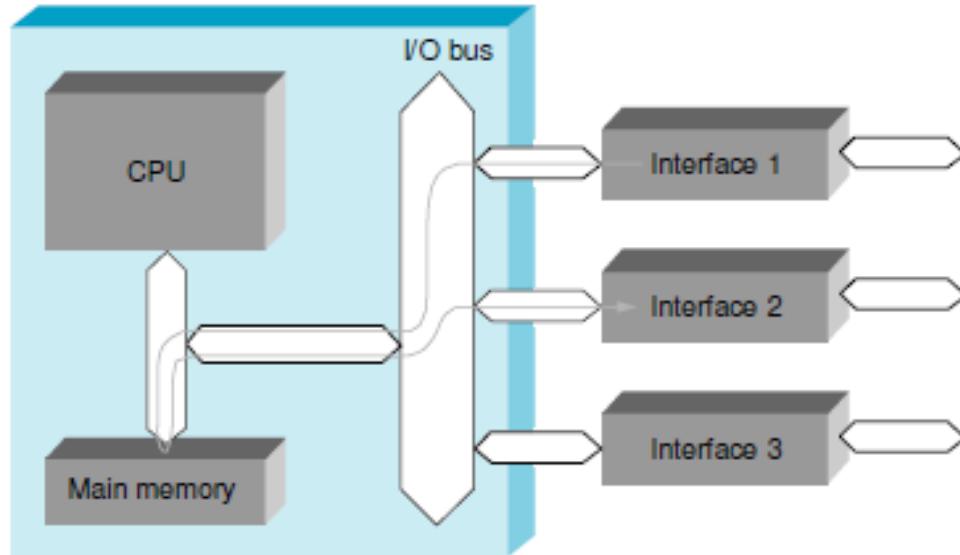
- Two-level hierarchy of virtual connections
 - Virtual path (VP)
 - Virtual circuit (VC)
- Switches in public network only maintains state about VPs, which is much fewer than the number of VCs
 - Thus, improves systems scalability

Cell Switching

Implementation and Performance

A workstation used as a packet switch

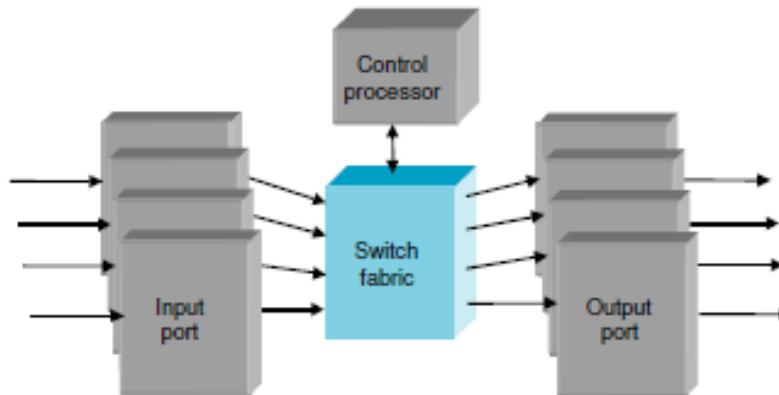
79



- Flexible in trying out different routing/switching strategies
 - Experimental systems
- (-) high switching overhead, and thus potentially low throughput

A 4×4 switch

80



- Input ports: switching logic control (e.g., which output ports to forward packets); usually do not buffer packets
- Switch fabric: forward packets from input ports to output ports; may have internal buffer space
- Output ports: buffering

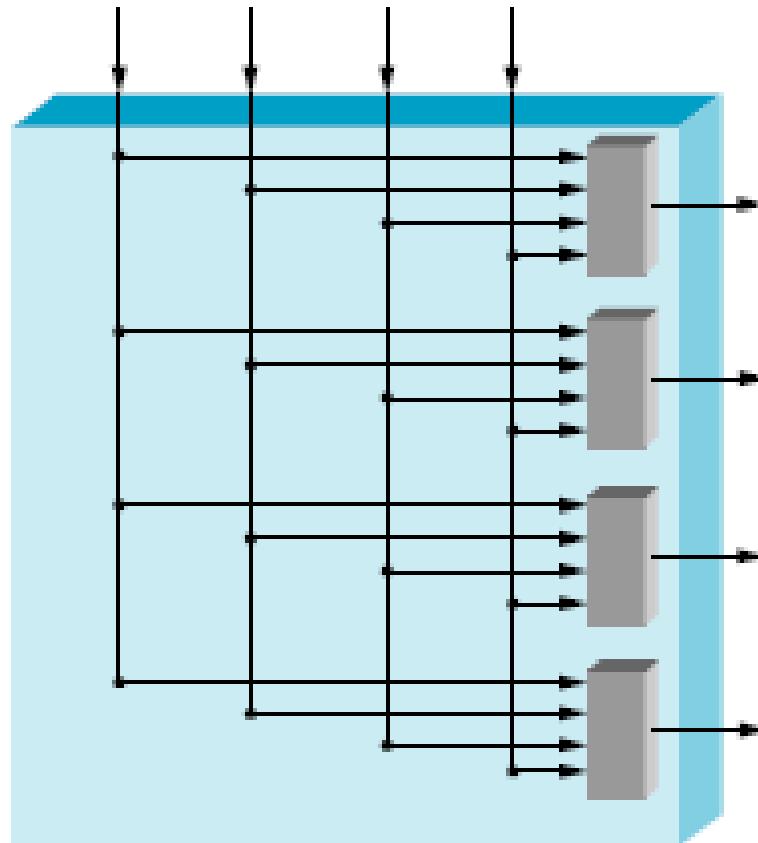
Switch Fabrics

81

- Shared Bus
 - This type of switches used in conventional workstation
 - Bandwidth determines the throughput of the switch
- Shared Memory
 - Packets are written into a memory location by an input port and then read from memory by the output ports
 - Memory bandwidth determines switch throughput
- Crossbar:
 - Switch is a matrix of pathways that can be configured to connect any input port to any output port

Crossbar Switch

82



A 4 x 4 Crossbar Switch

Switch Fabrics

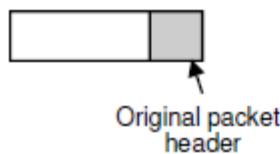
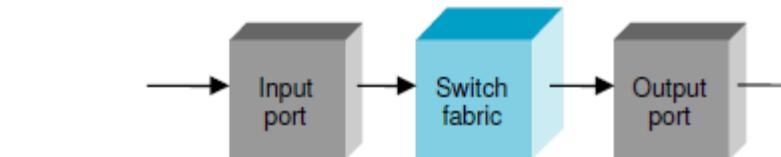
83

- Self-routing:
 - Rely on some information in the packet header to direct each packet to its correct output

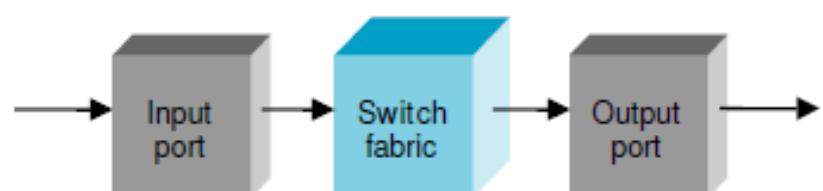
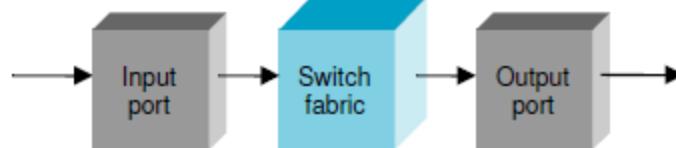
Switch fabrics (contd.)

84

- Self-routing header is applied to a packet at input to enable the fabric to send the packet to the correct output, where it is removed



Packet arrives at the input port



Input port attaches self-routing header to direct packet to correct output

20 October 2023

Self-routing header is removed at output port before packet leaves switch

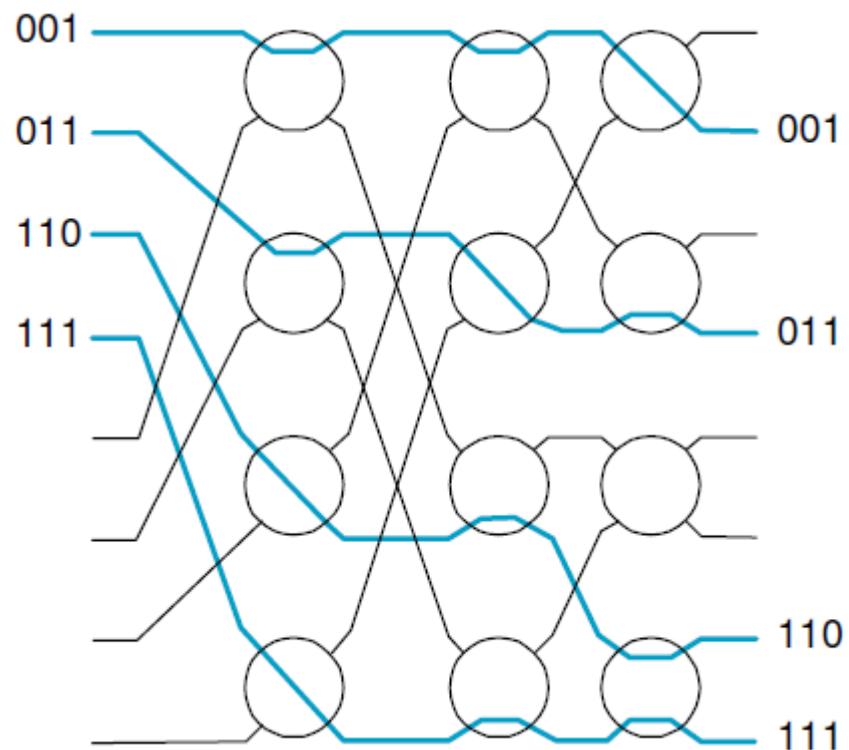
Dr Noor Muhammad Sk

Switch fabrics (contd.)

85

- Banyan Network: switch elements in 1st column looks at the most significant bit of output port number: 0 -> route packet to the top; 1 -> route to bottom ...

- Batcher-Banyan Switch design: Batcher network first sorts packets (for parallel tx.), then the packets are sent to Banyan network



Reference

86

- Chapter – 3: *Computer Networks A Systems Approach* by Larry L. Peterson and Bruce S. Davie, 4Th Edition, Morgan Kaufmann Publications.

INTERNETWORKING

20 October 2023

Dr Noor Mohammad Sk

Outline

2

- Simple Internetworking (IP)
- Routing
- Multicast

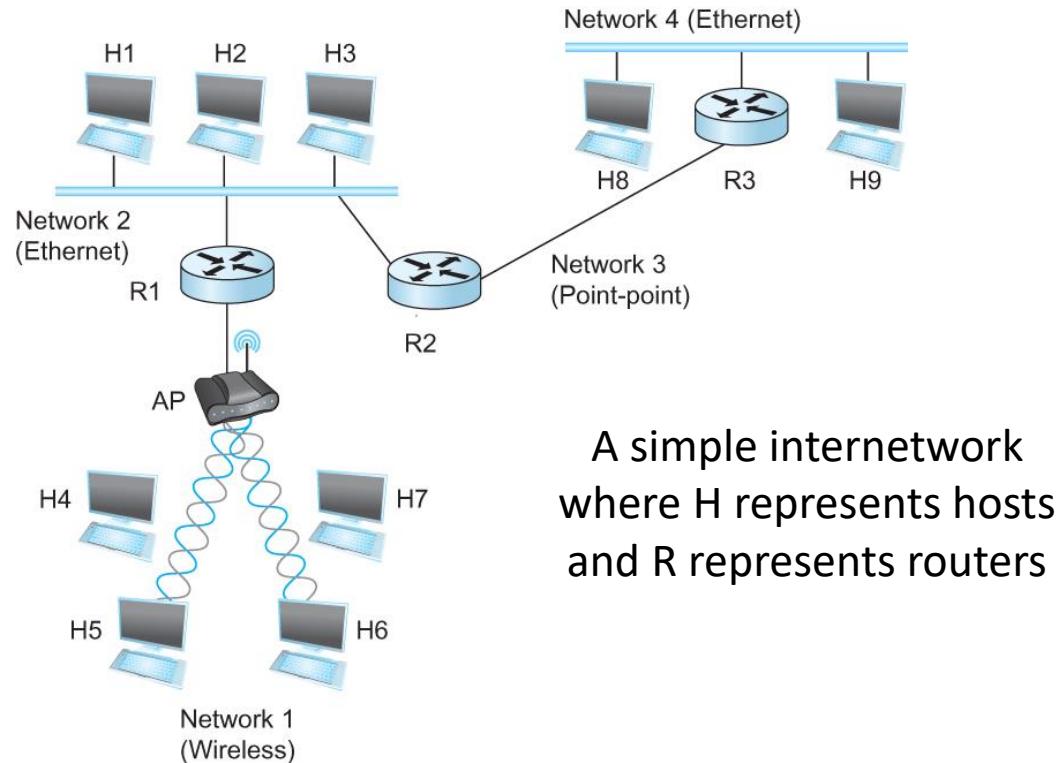
Simple Internetworking (IP)

What is an Internetwork

What is Internetworking

4

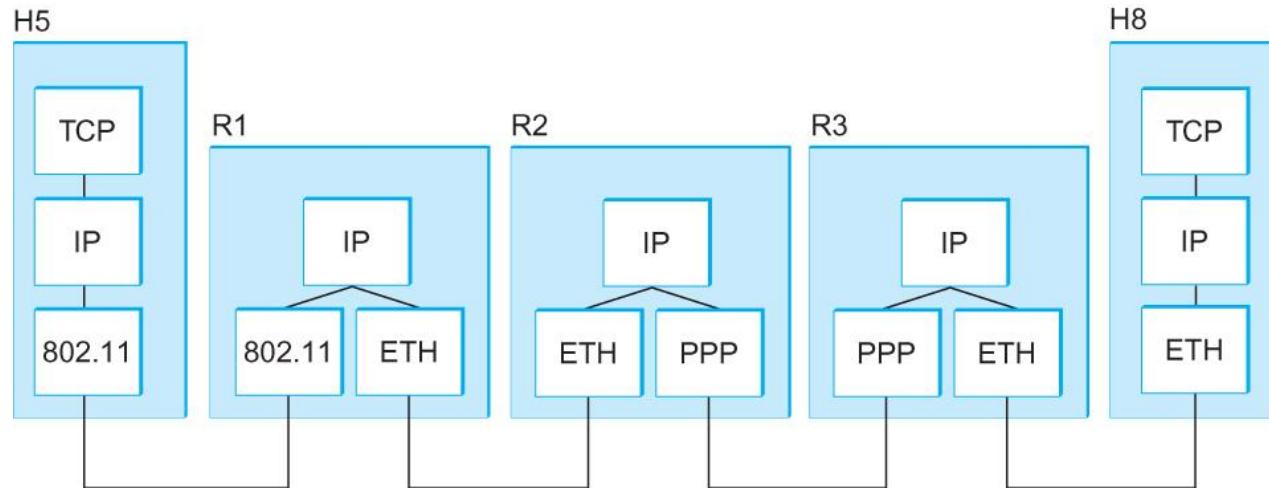
- An arbitrary collection of networks interconnected to provide some sort of host – host packet delivery service



What is IP

5

- IP stands for Internet Protocol
- Key tool used today to build scalable, heterogeneous internetworks
- 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



A simple internetwork showing the protocol layers

IP Service Model

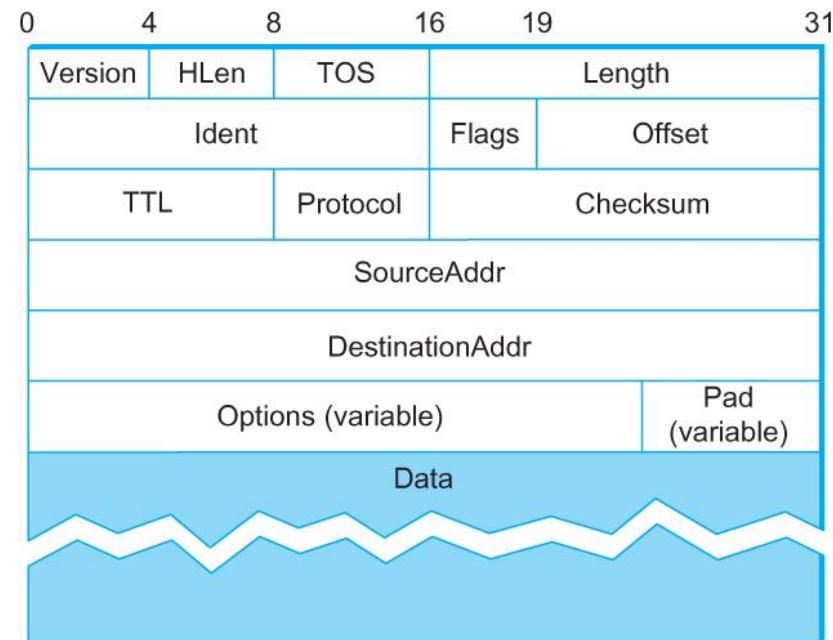
6

- Packet Delivery Model
 - ▣ Connectionless model for data delivery
 - ▣ Best-effort delivery (unreliable service)
 - Packets are lost
 - Packets are delivered out of order
 - Duplicate copies of a packet are delivered
 - Packets can be delayed for a long time
- Global Addressing Scheme
 - ▣ Provides a way to identify all hosts in the network

Packet Format (IPv4)

7

- Version (4)
- Hlen(4): number of 32-bit words in header
- TOS(8): Type of service (not widely used)
- Length(16): number of bytes in this datagram
- Idnet(16): used by fragmentation
- Flags/Offset(16): used by fragmentation
- TTL(8): number of hops this datagram has traveled
- Protocol(8): demux key (TCP=6, UDP=17)
- Checksum(16): of the header only
- DestAddr & SrcAddr (32)



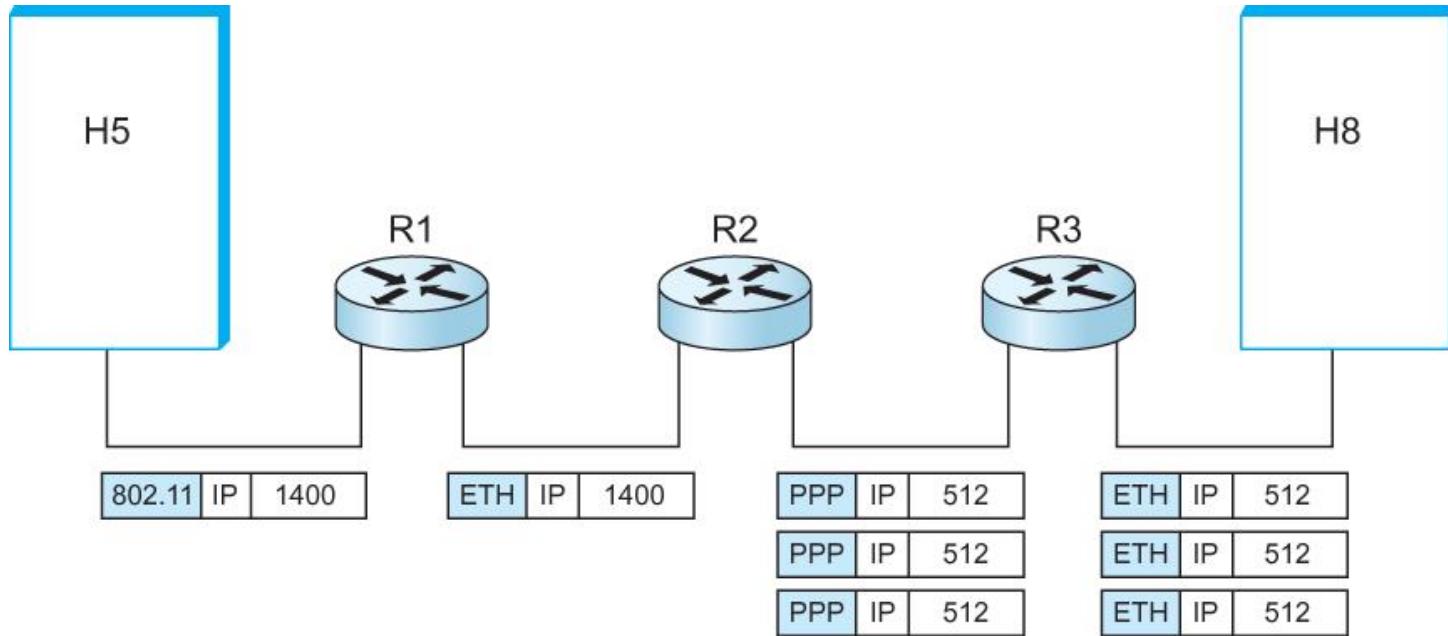
IP Fragmentation and Reassembly

8

- Each network has some MTU (Maximum Transmission Unit)
 - Ethernet (1500 bytes), FDDI (4500 bytes)
- Strategy
 - Fragmentation occurs in a router when it receives a datagram that it wants to forward over a network which has (MTU < datagram)
 - Reassembly is done at the receiving host
 - All the fragments carry the same identifier in the Ident field
 - Fragments are self-contained datagrams
 - IP does not recover from missing fragments

IP Fragmentation and Reassembly

9



IP datagrams traversing the sequence of physical networks

IP Fragmentation and Reassembly

10

Start of header			
Ident = x			0
Offset = 0			
Rest of header			
1400 data bytes			

Start of header			
Ident = x			1
Offset = 0			
Rest of header			
512 data bytes			

Start of header			
Ident = x			1
Offset = 64			
Rest of header			
512 data bytes			

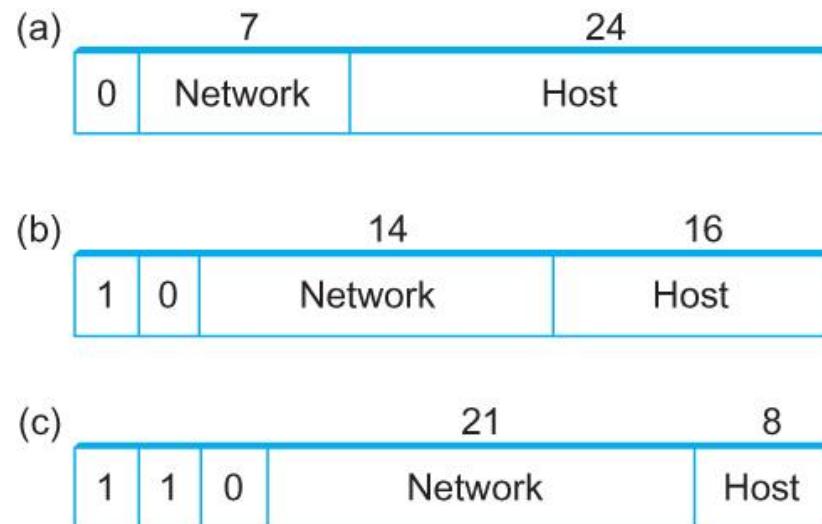
Start of header			
Ident = x			0
Offset = 128			
Rest of header			
376 data bytes			

Header fields used in IP fragmentation. (a)
Unfragmented packet; (b) fragmented packets.

Global Addresses

11

- Properties
 - Globally unique
 - Hierarchical: network + host
 - 4 Billion IP address, half are A type, $\frac{1}{4}$ is B type, and $\frac{1}{8}$ is the C type
- Format
- Dot notation
 - 10.3.2.4
 - 128.96.33.81
 - 192.12.69.77

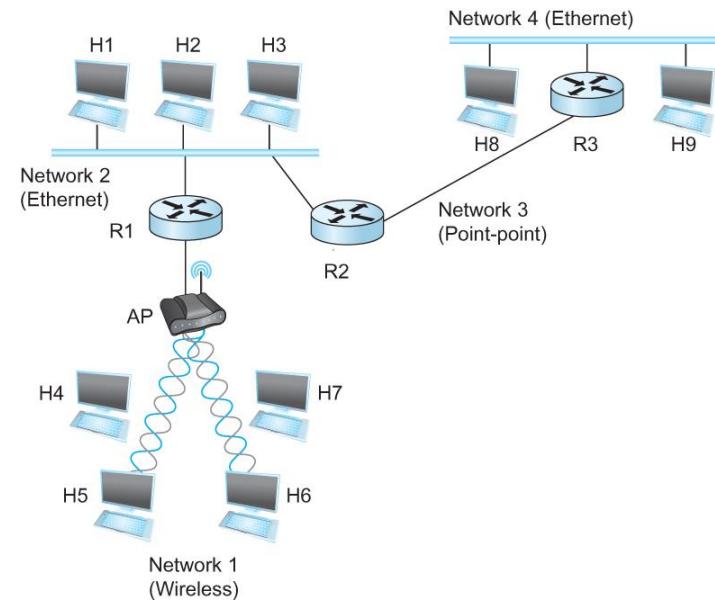


IP Datagram Forwarding

12

- Strategy
 - Every datagram contains destination's address
 - If directly connected to destination network, then forward to host
 - If not directly connected to destination network, then forward to some router
 - Forwarding table maps network number into next hop
 - Each host has a default router
 - Each router maintains a forwarding table
- Example (router R2)

NetworkNum	NextHop
1	R1
2	Interface 1
3	Interface 0
4	R3



IP Datagram Forwarding - Algorithm

13

```
if (NetworkNum of destination = NetworkNum of one of my interfaces) then
    deliver packet to destination over that interface
else
    if (NetworkNum of destination is in my forwarding table) then
        deliver packet to NextHop router
    else
        deliver packet to default router
```

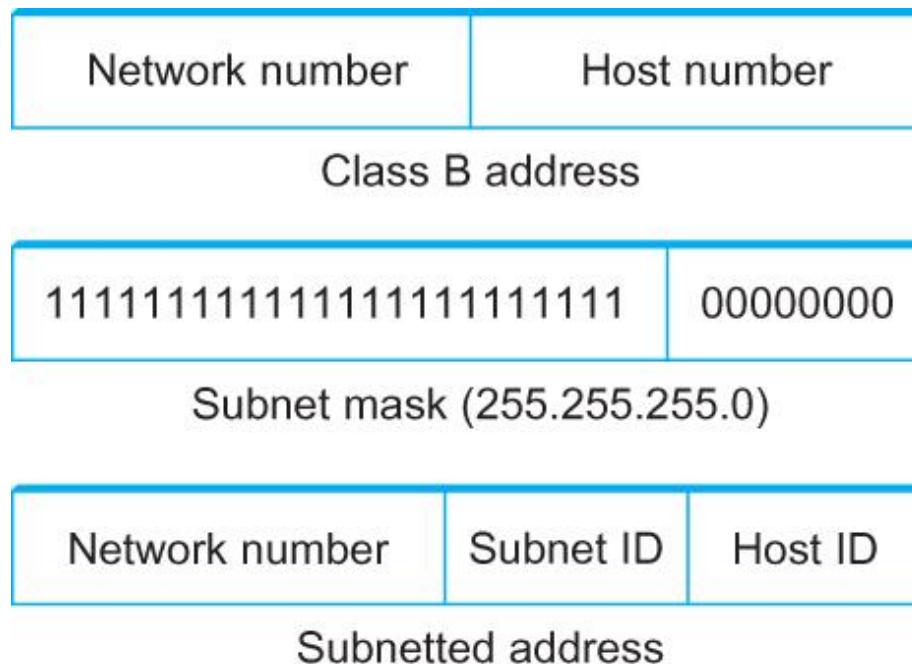
- For a host with only one interface and only a default router in its forwarding table, this simplifies to

```
if (NetworkNum of destination = my NetworkNum)then
    deliver packet to destination directly
else
    deliver packet to default router
```

Subnetting

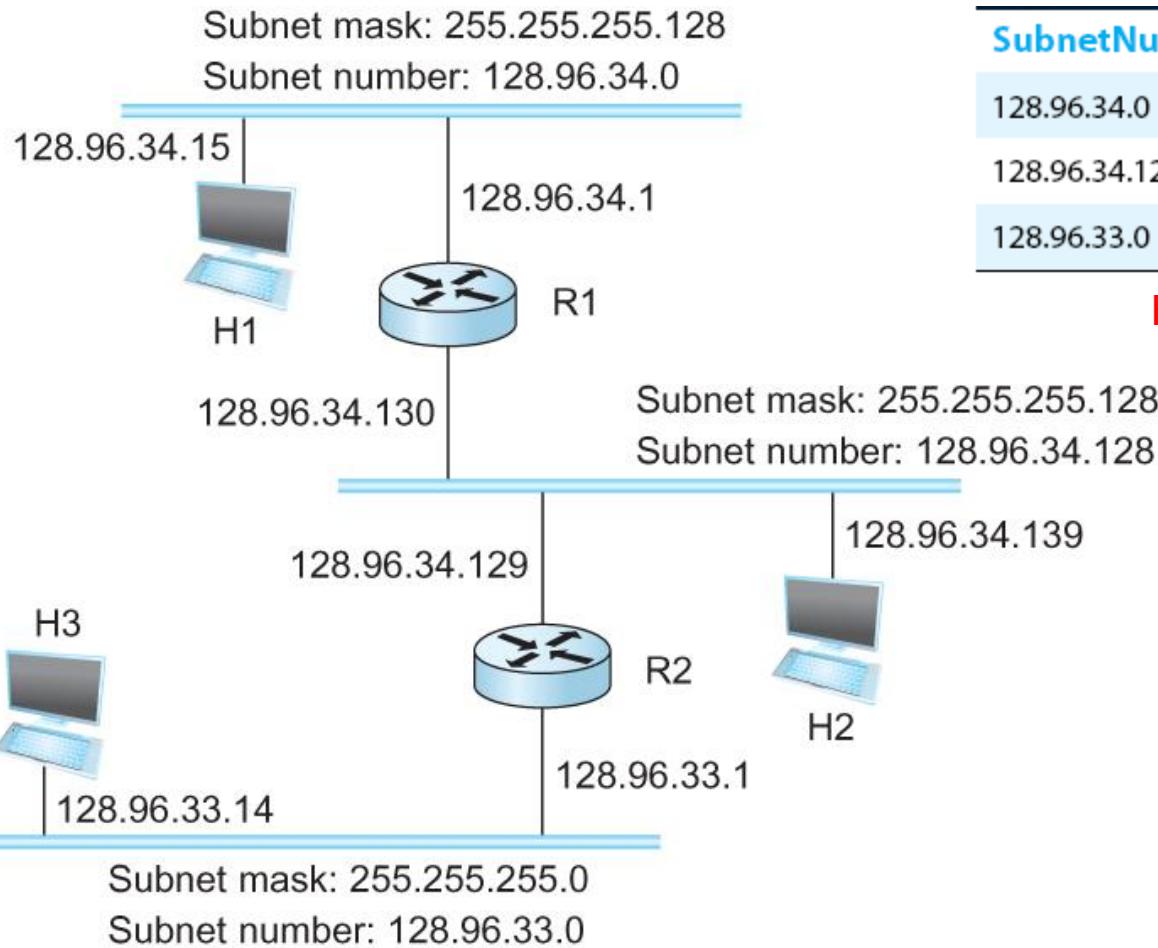
14

- Add another level to address/routing hierarchy: subnet
- Subnet masks define variable partition of host part of class A and B addresses
- Subnets visible only within site



Subnetting

15



SubnetNumber	SubnetMask	NextHop
128.96.34.0	255.255.255.128	Interface 0
128.96.34.128	255.255.255.128	Interface 1
128.96.33.0	255.255.255.0	R2

Forwarding Table at Router R1

Subnetting

16

□ Forwarding Algorithm

```
D = destination IP address
for each entry < SubnetNum, SubnetMask, NextHop>
    D1 = SubnetMask & D
    if D1 = SubnetNum
        if NextHop is an interface
            deliver datagram directly to destination
        else
            deliver datagram to NextHop (a router)
```

Subnetting

17

- Would use a default router if nothing matches
- Not necessary for all ones in subnet mask to be contiguous
- Can put multiple subnets on one physical network
- Subnets not visible from the rest of the internet

Classless Addressing

18

- Classless Inter-Domain Routing
 - A technique that addresses two scaling concerns in the Internet
 - The growth of backbone routing table as more and more network numbers need to be stored in them
 - Potential exhaustion of the 32-bit address space
 - Address assignment efficiency
 - Arises because of the IP address structure with class A, B and C addresses
 - Forces us to hand out network address space in fixed-size chunks of three very different sizes
 - A network with two hosts needs a class C address
 - Address assignment efficiency = $2/255 = 0.78$
 - A network with 256 hosts needs a class B address
 - Address assignment efficiency = $256/65535 = 0.39$

Classless Addressing

19

- Exhaustion of IP address space center on exhaustion of the class B network numbers
- Solution
 - ▣ Say “NO” to any autonomous system(AS) that requests a class B address unless they can show a need for something close to 64K addresses
 - ▣ Instead give them an appropriate number of class C addresses
 - ▣ For any AS with at least 256 hosts, we can guarantee an address space utilization of at least 50%

Classless Addressing

20

- Problem with this solution
 - Excessive storage requirement at the routers
- If a single AS has, say 16 class C network numbers assigned to it
 - Every Internet backbone router needs 16 entries in its routing tables for that AS
 - This is true, even if the path to every one of these networks is the same
- If we had assigned a class B address to the AS
 - The same routing information can be stored in one entry
 - Efficiency = $16 \times (255/65,536) = 6.2\%$

Classless Addressing

21

- CIDR tries to balance the desire to *minimize the number of routes that a router needs* to know against the need to hand out addresses efficiently

- CIDR uses aggregate routes
 - ▣ Uses a single entry in the forwarding table to tell the router how to reach a lot of different networks
 - ▣ Breaks the rigid boundaries between address classes

Classless Addressing

22

- Consider an AS with 16 class C network numbers.
- Instead of handing out 16 addresses at random, hand out a block of contiguous class C addresses
- Suppose we assign the class C network numbers from 192.4.16 through 192.4.31
- Observe that top 20 bits of all the addresses in this range are the same (11000000 00000100 0001)
 - ▣ We have created a 20-bit network number (which is in between class B network number and class C number)
- Requires to hand out blocks of class C addresses that share a common prefix

Classless Addressing

23

- Requires to hand out blocks of class C addresses that share a common prefix
- The convention is to place a /X after the prefix where X is the prefix length in bits
- For example, the 20-bit prefix for all the networks 192.4.16 through 192.4.31 is represented as 192.4.16/20

- By contrast, if we wanted to represent a single class C network number, which is 24 bits long, we would write it 192.4.16/24

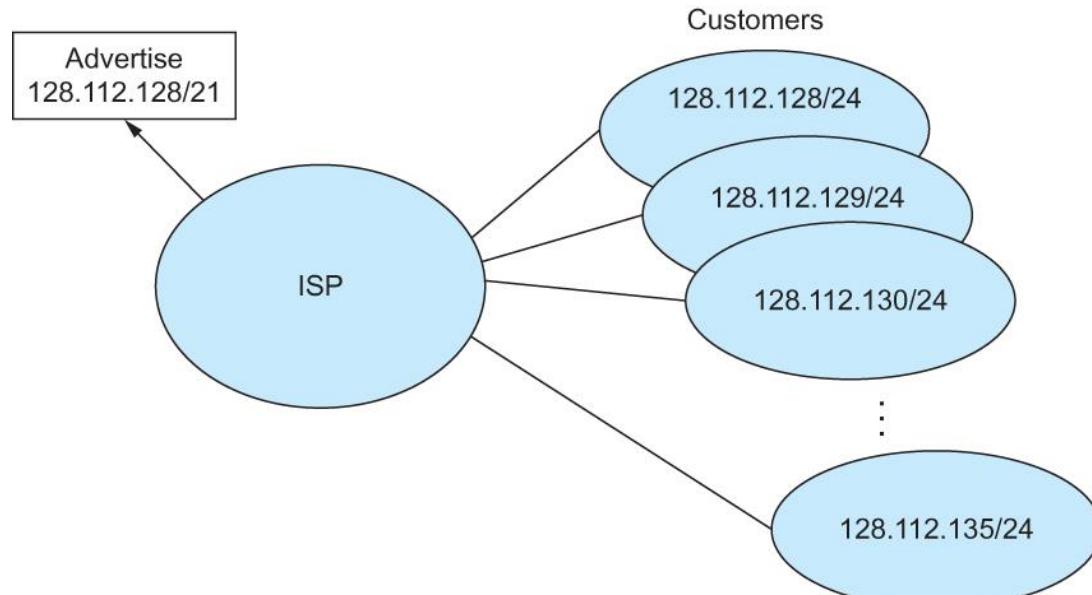
Classless Addressing

24

- How do the routing protocols handle this classless addresses
 - It must understand that the network number may be of any length
- Represent network number with a single pair
<length, value>
- All routers must understand CIDR addressing

Classless Addressing

25



Route aggregation with CIDR

IP Forwarding Revisited

26

- IP forwarding mechanism assumes that it can find the network number in a packet and then look up that number in the forwarding table
- We need to change this assumption in case of CIDR
- CIDR means that prefixes may be of any length, from 2 to 32 bits

IP Forwarding Revisited

27

- It is also possible to have prefixes in the forwarding tables that overlap
 - Some addresses may match more than one prefix
- For example, we might find both 171.69 (a 16 bit prefix) and 171.69.10 (a 24 bit prefix) in the forwarding table of a single router
- A packet destined to 171.69.10.5 clearly matches both prefixes.
 - The rule is based on the principle of “longest match”
 - 171.69.10 in this case
- A packet destined to 171.69.20.5 would match 171.69 and not 171.69.10

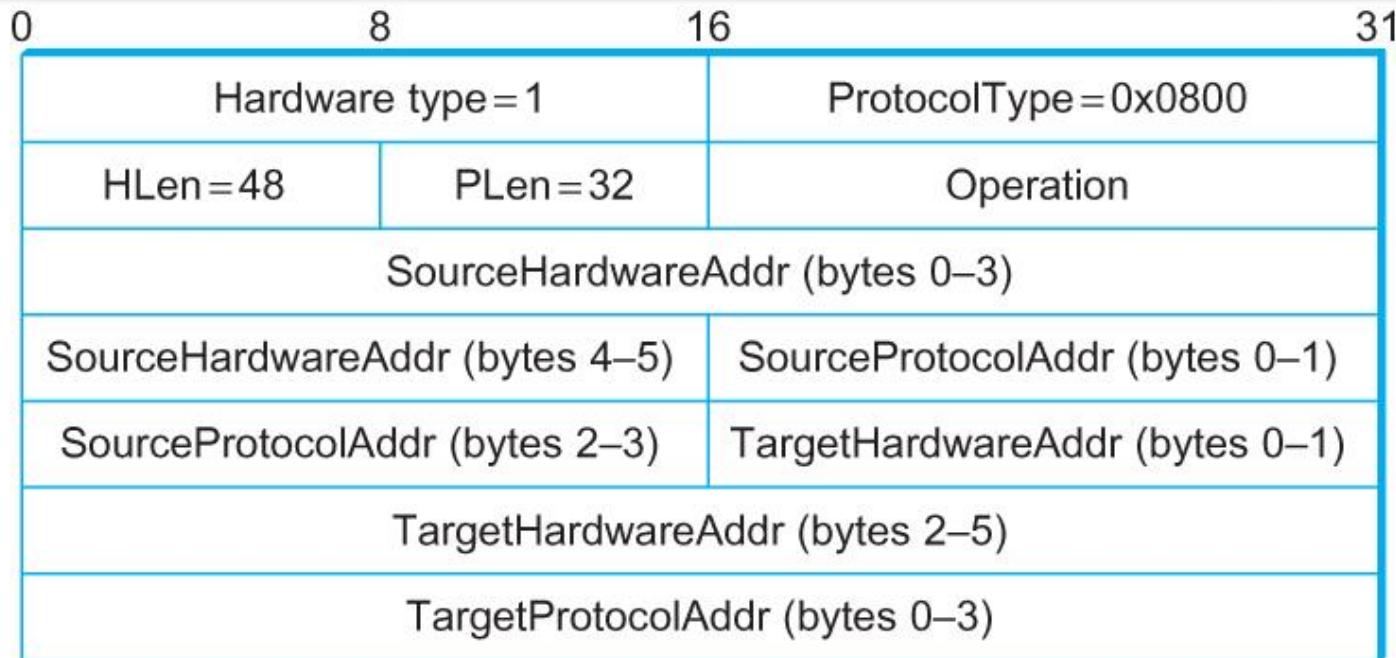
Address Translation Protocol (ARP)

28

- Map IP addresses into physical addresses
 - destination host
 - next hop router
- Techniques
 - encode physical address in host part of IP address
 - table-based
- ARP (Address Resolution Protocol)
 - table of IP to physical address bindings
 - broadcast request if IP address not in table
 - target machine responds with its physical address
 - table entries are discarded if not refreshed

ARP Packet Format

29



- HardwareType: type of physical network (e.g., Ethernet)
- ProtocolType: type of higher layer protocol (e.g., IP)
- HLEN & PLEN: length of physical and protocol addresses
- Operation: request or response
- Source/Target Physical/Protocol addresses

Host Configuration

30

- Ethernet addresses are configured into network by manufacturer and they are unique
- IP addresses must be unique on a given internetwork but also must reflect the structure of the internetwork
- Most host Operating systems provide a way to manually configure the IP information for the host
- Drawbacks of manual configuration
 - ▣ A lot of work to configure all the hosts in a large network
 - ▣ Configuration process is error-prone
- Automated configuration processes is required

Dynamic Host Configuration Protocol (DHCP)

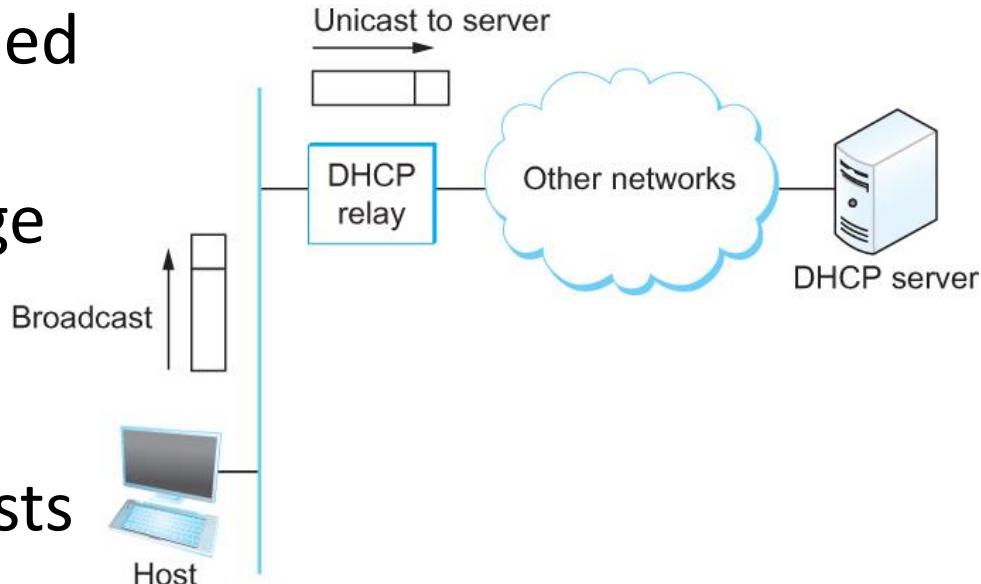
31

- DHCP server is responsible for providing configuration information to hosts
- There is at least one DHCP server for an administrative domain
- DHCP server maintains a pool of available addresses

DHCP

32

- Newly booted or attached host sends DHCPDISCOVER message to a special IP address (255.255.255.255)
- DHCP relay agent unicasts the message to DHCP server and waits for the response



Internet Control Message Protocol (ICMP)

33

- Defines a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully
 - Destination host unreachable due to link/node failure
 - Reassembly process failed
 - TTL had reached 0 (so datagram don't cycle forever)
 - IP header checksum failed
- ICMP – Redirect
 - From router to a source host
 - With a better route information

Routing

34

- Forwarding Versus Routing

- Forwarding

- To select an output port based on destination address and routing table

- Routing

- Process by which routing table is built

Routing – Forwarding Table Vs Routing Table

35

- Forwarding Table
 - Used when a packet is being forwarded and so must contain enough information to accomplish the forwarding function
 - A row in the forwarding table contains the mapping from a network number to an outgoing interface and some MAC information, such as Ethernet address of the next hop
- Routing Table
 - Built by the routing algorithm as a precursor to build the forwarding table
 - Generally contains mapping from network numbers to next hops

Routing

36

(a)

Prefix/Length	Next Hop
18/8	171.69.245.10

(b)

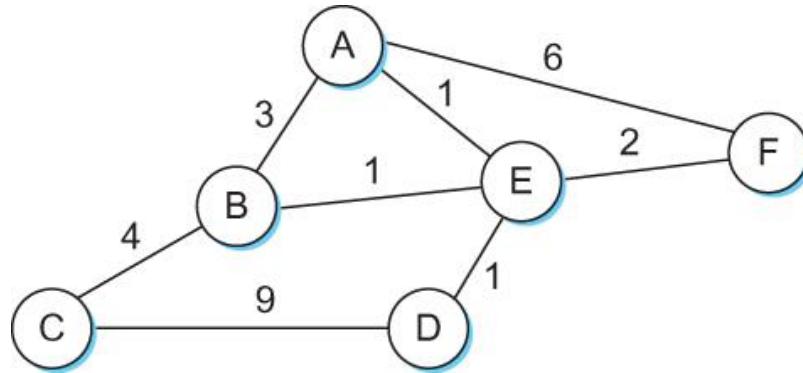
Prefix/Length	Interface	MAC Address
18/8	if0	8:0:2b:e4:b:1:2

Example rows from (a) routing and (b) forwarding tables

Routing

37

- Network as a Graph



- The basic problem of routing is to find the lowest-cost path between any two nodes
 - Where the cost of a path equals the sum of the costs of all the edges that make up the path

Routing

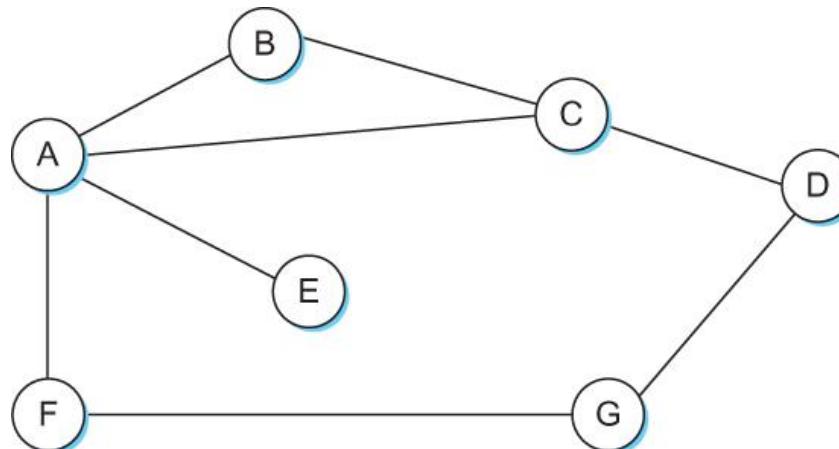
38

- For a simple network, we can calculate all shortest paths and load them into some nonvolatile storage on each node
- Such a static approach has several shortcomings
 - ▣ It does not deal with node or link failures
 - ▣ It does not consider the addition of new nodes or links
 - ▣ It implies that edge costs cannot change
- Why is the solution?
 - ▣ Need a distributed and dynamic protocol
 - ▣ Two main classes of protocols
 - Distance Vector
 - Link State

Distance Vector

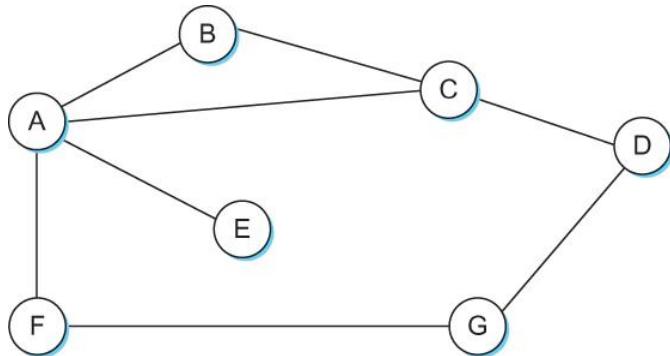
39

- Each node constructs a one dimensional array (a vector) containing the “distances” (costs) to all other nodes and distributes that vector to its immediate neighbors
- Starting assumption is that each node knows the cost of the link to each of its directly connected neighbors



Distance Vector

40

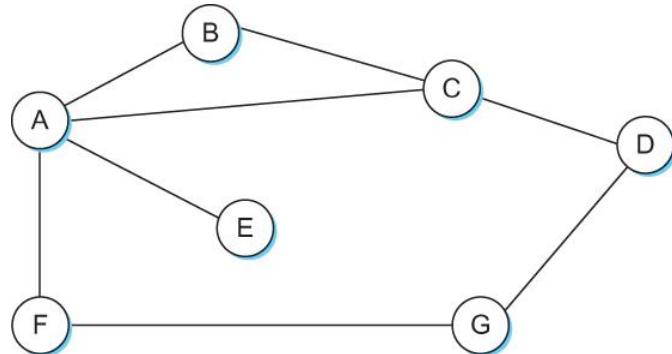


Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Initial distances stored at each node (global view)

Distance Vector

41

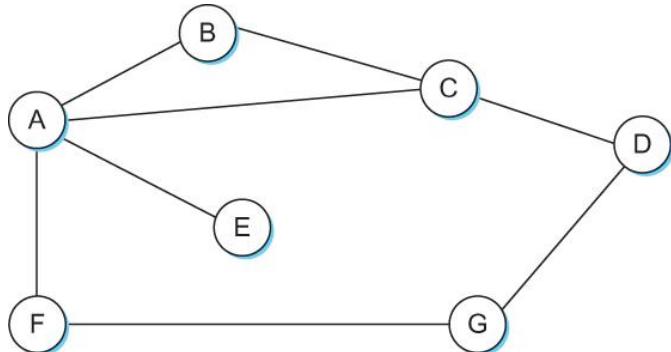


Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Initial routing table at node A

Distance Vector

42

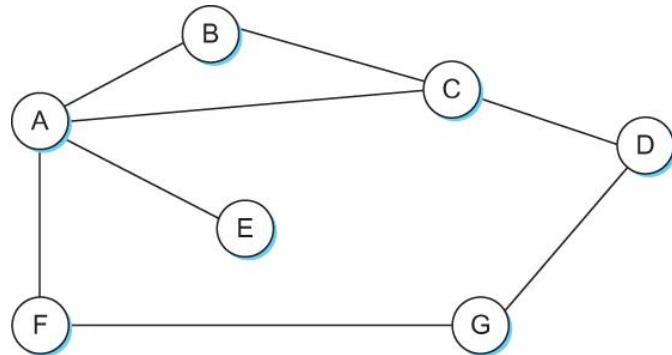


Destination	Cost	Nexthop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Final routing table at node A

Distance Vector

43



Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Final distances stored at each node (global view)

Distance Vector

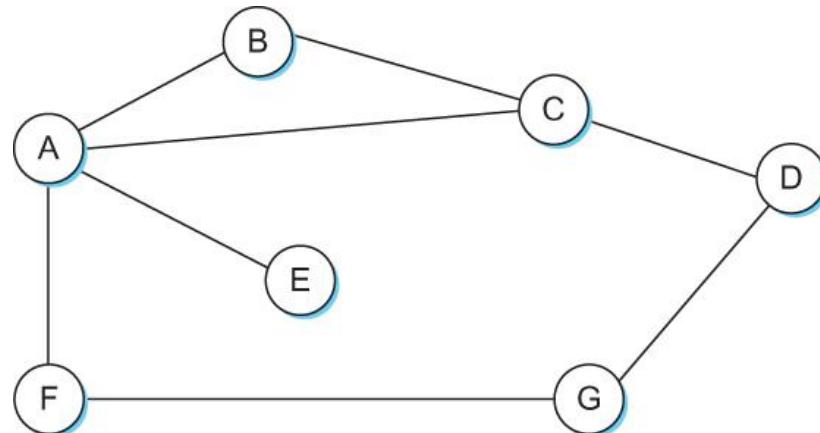
44

- The distance vector routing algorithm is sometimes called as Bellman-Ford algorithm
- Every T seconds each router sends its table to its neighbor each router then updates its table based on the new information
- Problems include fast response to good news and slow response to bad news. Also too many messages to update

Distance Vector

45

- When a node detects a link failure
 - F detects that link to G has failed
 - F sets distance to G to infinity and sends update to A
 - A sets distance to G to infinity since it uses F to reach G
 - A receives periodic update from C with 2-hop path to G
 - A sets distance to G to 3 and sends update to F
 - F decides it can reach G in 4 hops via A



Distance Vector

46

- Slightly different circumstances can prevent the network from stabilizing
 - Suppose the link from A to E goes down
 - In the next round of updates, A advertises a distance of infinity to E, but B and C advertise a distance of 2 to E
 - Depending on the exact timing of events, the following might happen
 - Node B, upon hearing that E can be reached in 2 hops from C, concludes that it can reach E in 3 hops and advertises this to A
 - Node A concludes that it can reach E in 4 hops and advertises this to C
 - Node C concludes that it can reach E in 5 hops; and so on
 - This cycle stops only when the distances reach some number that is large enough to be considered infinite
 - Count – to – infinity problem

Count-to-infinity Problem

47

- Use some relatively small number as an approximation of infinity
- For example, the maximum number of hops to get across a certain network is never going to be more than 16
- One technique to improve the time to stabilize routing is called split horizon
 - When a node sends a routing update to its neighbors, it does not send those routes it learned from each neighbor back to that neighbor
 - For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update

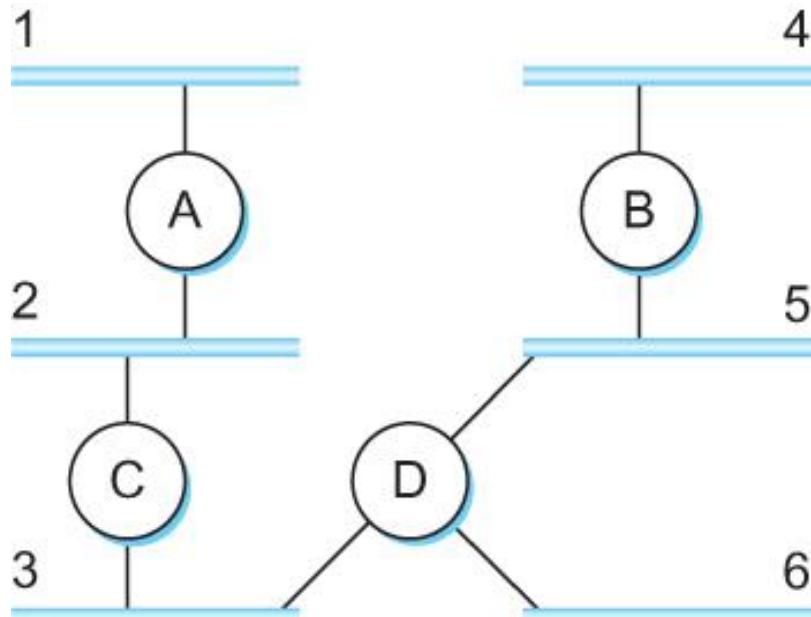
Count – to – infinity Problem

48

- In a stronger version of split horizon, called split horizon with poison reverse
 - ▣ B actually sends that back route to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E
 - ▣ For example, B sends the route (E, ∞) to A

Routing Information Protocol (RIP)

49



Example Network running RIP

0	8	16	31
Command	Version	Must be zero	
Family of net 1	Route Tags		
Address prefix of net 1			
Mask of net 1			
Distance to net 1			
Family of net 2	Route Tags		
Address prefix of net 2			
Mask of net 2			
Distance to net 2			

RIPv2 Packet Format

Link State Routing

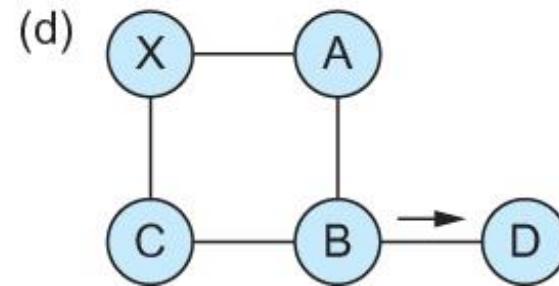
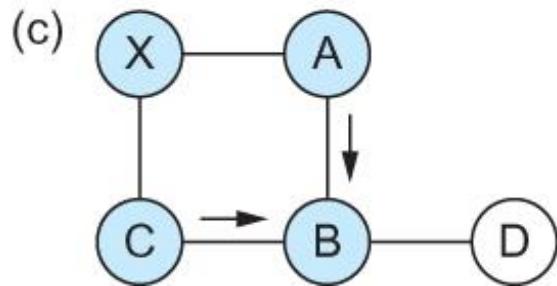
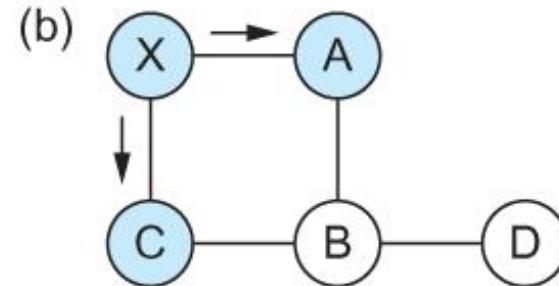
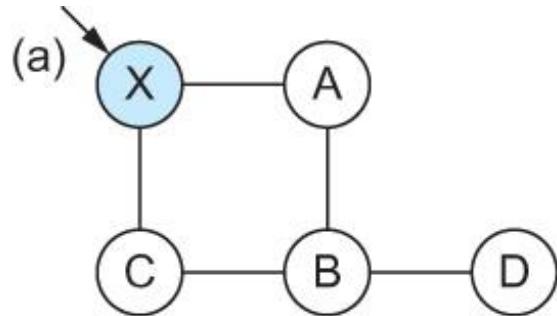
50

- Strategy: Send to all nodes (not just neighbors) information about directly connected links (not entire routing table)
- Link State Packet (LSP)
 - Id of the node that created the LSP
 - Cost of link to each directly connected neighbor
 - Sequence number (SEQNO)
 - Time-to-live (TTL) for this packet
- Reliable Flooding
 - Store most recent LSP from each node
 - Forward LSP to all nodes but one that send it
 - Generate new LSP periodically; increment SEQNO
 - Start SEQNO at 0 when reboot
 - Decrement TTL of each stored LSP; discard when TTL = 0

Link State

51

□ Reliable Flooding



Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) A and C flood LSP to B (but not X); (d) flooding is complete

Shortest Path Routing

52

- Dijkstra's Algorithm - Assume non-negative link weights
 - N : set of nodes in the graph
 - $l(i, j)$: the non-negative cost associated with the edge between nodes $i, j \in N$ and $l(i, j) = \infty$ if no edge connects i and j
 - Let $s \in N$ be the starting node which executes the algorithm to find shortest paths to all other nodes in N
 - Two variables used by the algorithm
 - M : set of nodes incorporated so far by the algorithm
 - $C(n)$: the cost of the path from s to each node n
 - The algorithm

```
M = {s}
For each n in N - {s}
    C(n) = l(s, n)
while ( N ≠ M)
    M = M ∪ {w} such that C(w) is the minimum
                                for all w in (N-M)
    For each n in (N-M)
        C(n) = MIN (C(n), C(w) + l(w, n))
```

Shortest Path Routing

53

- In practice, each switch computes its routing table directly from the LSP's it has collected using a realization of Dijkstra's algorithm called the forward search algorithm
- Specifically each switch maintains two lists, known as Tentative and Confirmed
- Each of these lists contains a set of entries of the form (Destination, Cost, NextHop)

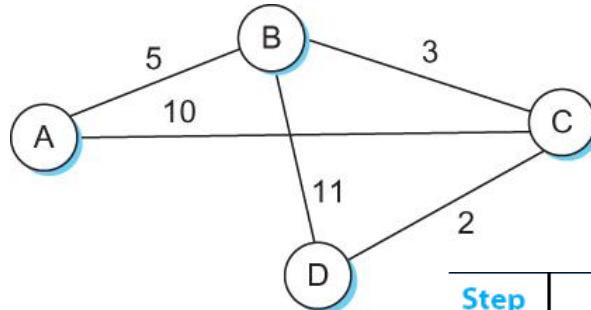
Shortest Path Routing

54

- The algorithm
 - Initialize the **Confirmed** list with an entry for myself; this entry has a cost of 0
 - For the node just added to the **Confirmed** list in the previous step, call it node **Next**, select its LSP
 - For each neighbor (Neighbor) of **Next**, calculate the cost (Cost) to reach this Neighbor as the sum of the cost from myself to Next and from Next to Neighbor
 - If Neighbor is currently on neither the **Confirmed** nor the **Tentative** list, then add (Neighbor, Cost, Nexthop) to the **Tentative** list, where Nexthop is the direction I go to reach Next
 - If Neighbor is currently on the **Tentative** list, and the Cost is less than the currently listed cost for the Neighbor, then replace the current entry with (Neighbor, Cost, Nexthop) where Nexthop is the direction I go to reach Next
 - If the **Tentative** list is empty, stop. Otherwise, pick the entry from the **Tentative** list with the lowest cost, move it to the **Confirmed** list, and return to Step 2.

Shortest Path Routing

55



Step	Confirmed	Tentative	Comments
1	(D,0,-)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,-)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,-) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,-) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

Open Shortest Path First (OSPF)

56

0 8 16 31

Version	Type	Message length		
SourceAddr				
AreaId				
Checksum	Authentication type			
Authentication				

OSPF Header Format

LS Age	Options	Type=1
Link-state ID		
Advertising router		
LS sequence number		
LS checksum	Length	
0	Flags	0
Number of links		
Link ID		
Link data		
Link type	Num_TOS	Metric
Optional TOS information		
More links		

OSPF Link State Advertisement

57

IPv6

Major Features

58

- 128 – bit addresses
- Multicast
- Real-time service
- Authentication and security
- Auto-configuration
- End-to-end fragmentation
- Enhanced routing functionality, including support for mobile hosts

IPv6 Addresses

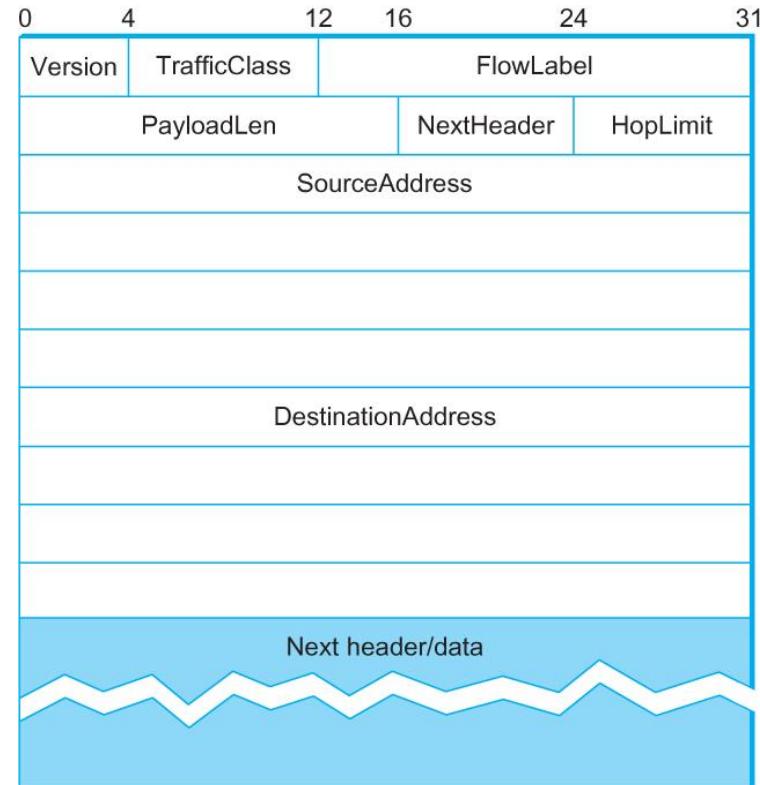
59

- Classless addressing/routing (similar to CIDR)
- Notation: x:x:x:x: x:x:x:x (x = 16-bit hex number)
 - ▣ Contiguous 0s are compressed: 47CD::A456:0124
 - ▣ IPv6 compatible IPv4 address: 128.42.1.87
- Address assignment
 - ▣ Provider-based
 - ▣ Geographic

IPv6 Header

60

- 40-byte “base” header
- Extension headers (fixed order, mostly fixed length)
 - Fragmentation
 - Source routing
 - Authentication and security
 - Other options



Internet Multicast

Multicast

62

- One-to-many
 - Radio station broadcast
 - Transmitting news, stock-price
 - Software updates to multiple hosts
- Many-to-many
 - Multimedia teleconferencing
 - Online multi-player games
 - Distributed simulations

Multicast

63

- Without support for multicast
 - ▣ A source needs to send a separate packet with the identical data to each member of the group
 - This redundancy consumes more bandwidth
 - Redundant traffic is not evenly distributed, concentrated near the sending host
 - ▣ Source needs to keep track of the IP address of each member in the group
 - Group may be dynamic
- To support many-to-many and one-to-many IP provides as IP-level multicast

Multicast

64

- Basic IP multicast model is many-to-many based on multicast groups
 - Each group has its own IP multicast address
 - Hosts that the members of a group receive copies of any packets sent to that group's multicast address
 - A host can be in multiple groups
 - A host can join and leave groups

Multicast

65

- Using IP multicast to send the identical packet to each member of the group
 - A host sends a single copy of the packet addressed to the group's multicast address
 - The sending host does not need to know the individual unicast IP address of each member
 - Sending host does not send multiple copies of the packet

Multicast

66

- IP's original many-to-many multicast has been supplemented with support for a form of one-to-many multicast
- One-to-many multicast
 - Source specific multicast (SSM)
 - A receiving host specifies both a multicast group and a specific sending host
- Many-to-many model
 - Any source multicast (ASM)

Multicast

67

- A host signals its desire to join or leave a multicast group by communicating with its local router using a special protocol
 - ▣ In IPv4, the protocol is Internet Group Management Protocol (IGMP)
 - ▣ In IPv6, the protocol is Multicast Listener Discovery (MLD)
- The router has the responsibility for making multicast behave correctly with regard to the host

Multicast Routing

68

- A router's unicast forwarding tables indicate for any IP address, which link to use to forward the unicast packet
- To support multicast, a router must additionally have multicast forwarding tables that indicate, based on multicast address, which links to use to forward the multicast packet
- Unicast forwarding tables collectively specify a set of paths
- Multicast forwarding tables collectively specify a set of trees
 - Multicast distribution trees

Multicast Routing

69

- To support source specific multicast, the multicast forwarding tables must indicate which links to use based on the combination of multicast address and the unicast IP address of the source

- Multicast routing is the process by which multicast distribution trees are determined

Distance-Vector Multicast

70

- Each router already knows that shortest path to source S goes through router N.
- When receive multicast packet from S, forward on all outgoing links (except the one on which the packet arrived), iff packet arrived from N.
- Eliminate duplicate broadcast packets by only letting
 - “parent” for LAN (relative to S) forward
 - shortest path to S (learn via distance vector)
 - smallest address to break ties

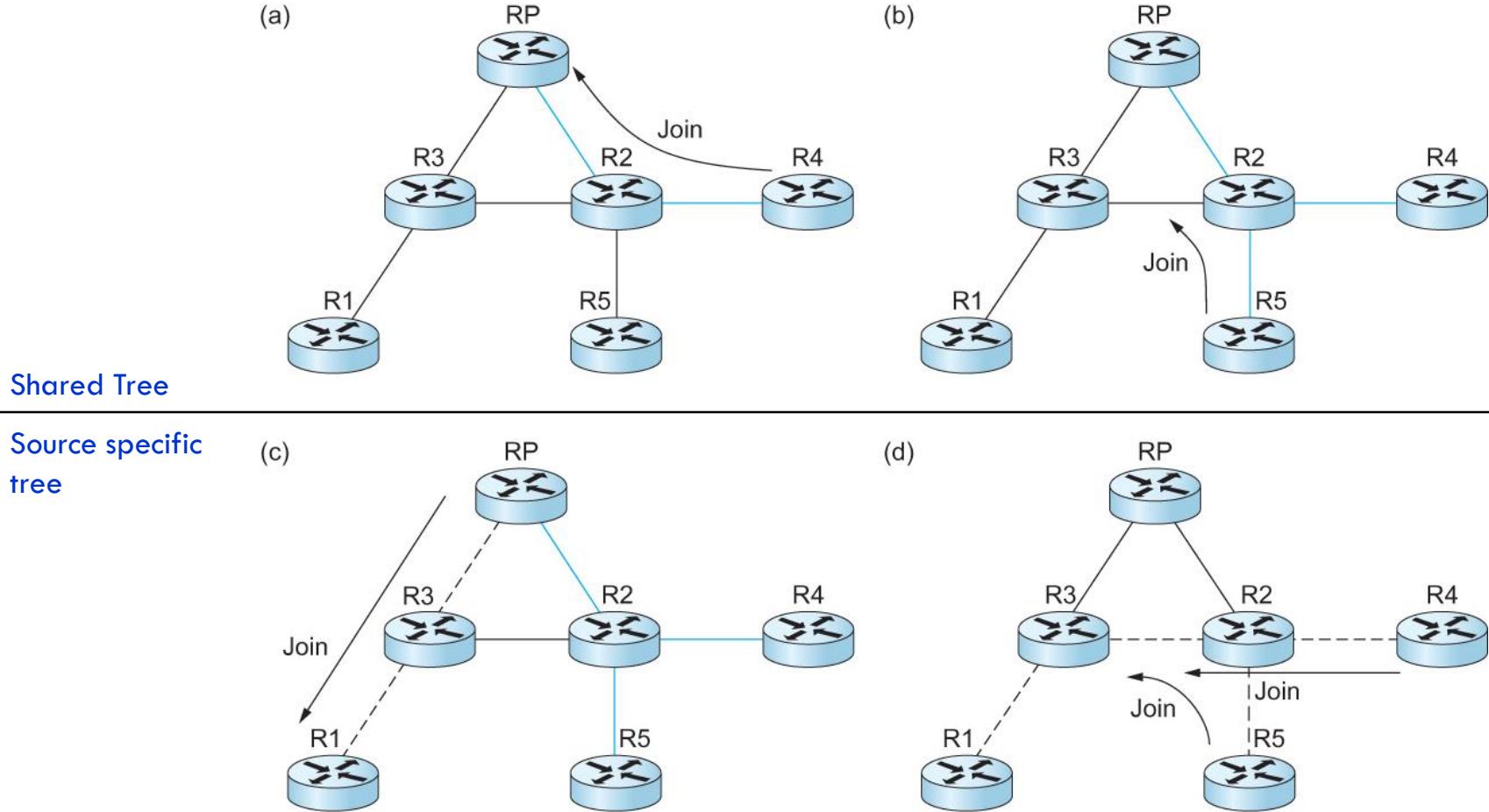
Distance Vector Multicast

71

- Reverse Path Broadcast (RPB)
- Goal: Prune networks that have no hosts in group G
- Step 1: Determine if LAN is a *leaf* with no members in G
 - leaf if parent is only router on the LAN
 - determine if any hosts are members of G using IGMP
- Step 2: Propagate “no members of G here” information
 - augment <**Destination, Cost**> update sent to neighbors with set of groups for which this network is interested in receiving multicast packets.
 - only happens when multicast address becomes active.

Protocol Independent Multicast (PIM)

72



RP = Rendezvous point

20 October 2023

— Shared tree

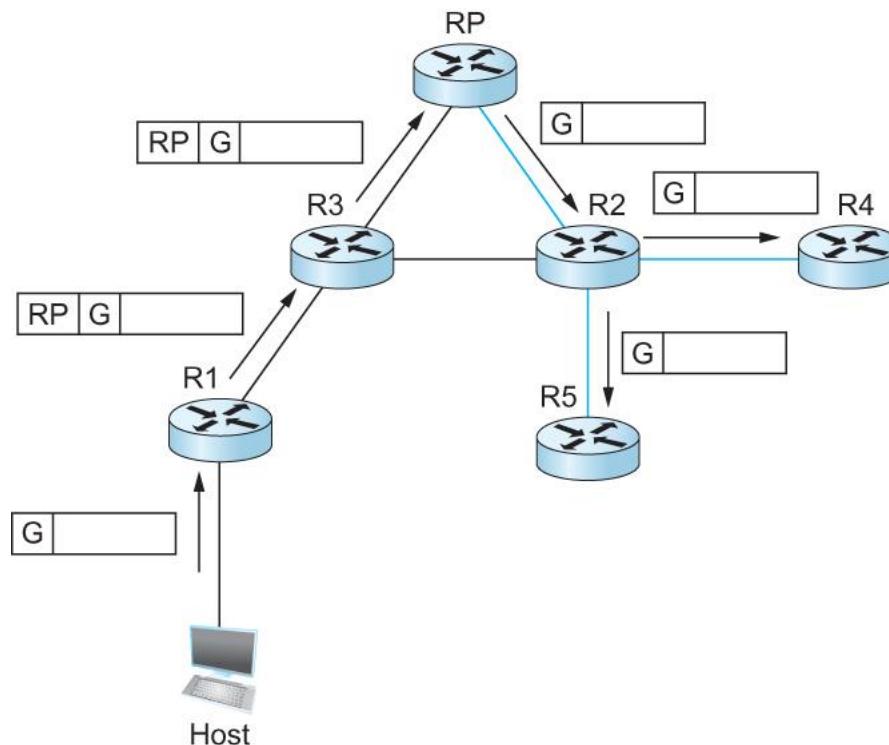
---- Source-specific tree for source R1

Dr Noor Mohammad Sk

Protocol Independent Multicast (PIM)

73

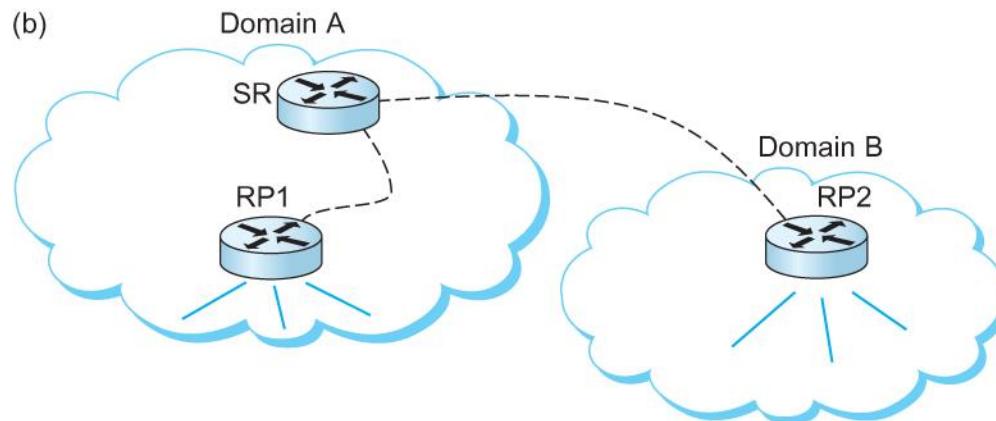
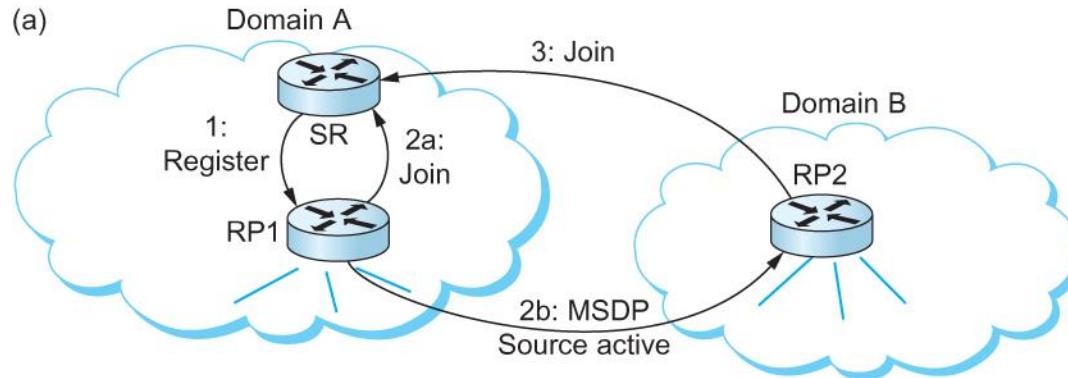
- Delivery of a packet along a shared tree. R1 tunnels the packet to the RP, which forwards it along the shared tree to R4 and R5.



Inter – domain Multicast

74

□ Multicast Source Discovery Protocol (MSDP)



— Shared tree

- - - Source-specific tree for source SR

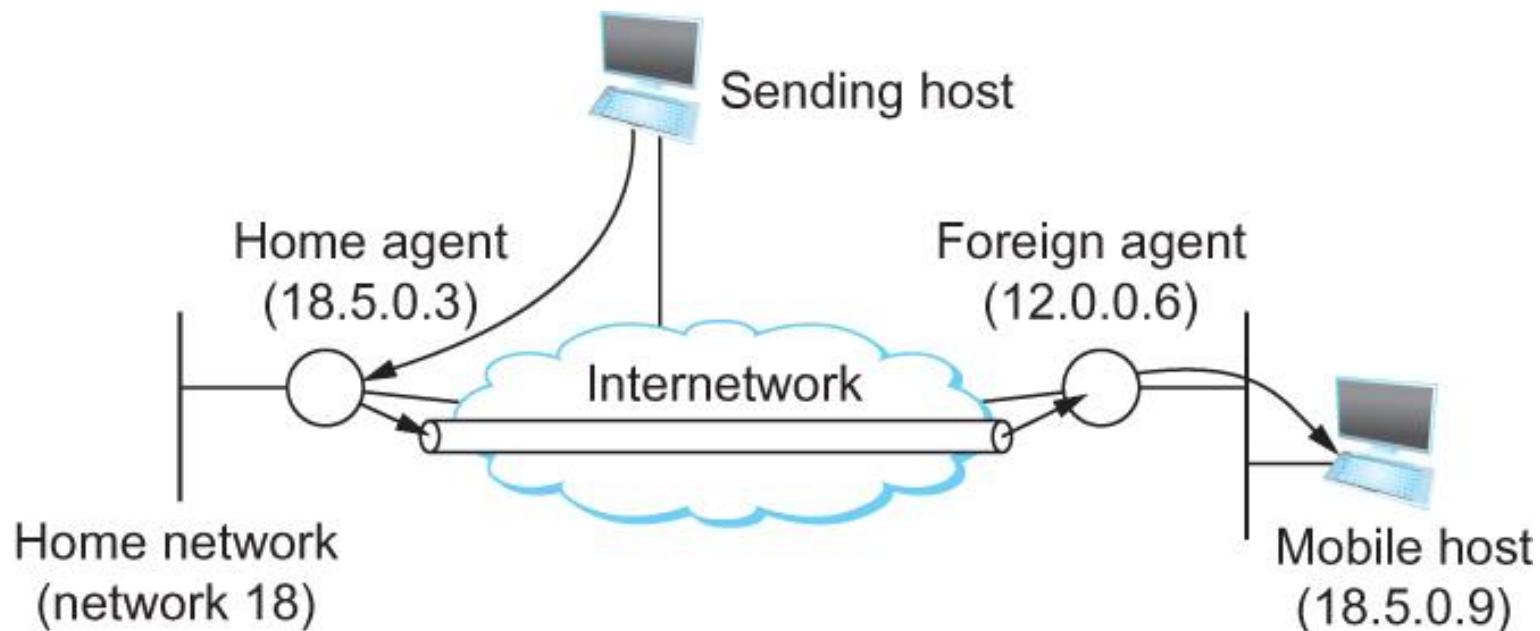
Routing for Mobile Hosts

75

- Mobile IP
 - Home Agent
 - Router located on the home network of the mobile hosts
 - Home Address
 - The permanent IP address of the mobile host
 - Has a network number equal to that of the home network and thus of the home agent
 - Foreign Agent
 - Router located on a network to which the mobile node attaches itself when it is away from its home network

Routing for Mobile Hosts

76



Routing for Mobile Hosts

77

- Problem of delivering a packet to the mobile node
 - How does the home agent intercept a packet that is destined for the mobile node?
 - Proxy ARP
 - How does the home agent then deliver the packet to the foreign agent?
 - IP tunnel
 - Care – of – address
 - How does the foreign agent deliver the packet to the mobile node?

Routing for Mobile Hosts

78

- Route optimization in Mobile IP
 - The route from the sending node to mobile node can be significantly sub-optimal
 - One extreme example
 - The mobile node and the sending node are on the same network, but the home network for the mobile node is on the far side of the internet
 - Triangle Routing Problem

Routing for Mobile Hosts

79

□ Solution

- Let the sending node know the care-of-address of the mobile node. The sending node can create its own tunnel to the foreign agent
- Home agent sends binding update message
- The sending node creates an entry in the binding cache
- The binding cache may become out-of-date
 - The mobile node moved to a different network
 - Foreign agent sends a binding warning message

Reference

80

- Chapter – 4: *Computer Networks A Systems Approach* by Larry L. Peterson and Bruce S. Davie, 4Th Edition, Morgan Kaufmann Publications.

END-TO-END PROTOCOLS

20 October 2023

Dr Noor Mohammad Sk

Outline

2

- Simple Demultiplexer (UDP)
- Reliable Byte Stream (TCP)
- Remote Procedure Call

End-to-end Protocols

3

- Common properties that a transport protocol can be expected to provide
 - ▣ Guarantees message delivery
 - ▣ Delivers messages in the same order they were sent
 - ▣ Delivers at most one copy of each message
 - ▣ Supports arbitrarily large messages
 - ▣ Supports synchronization between the sender and the receiver
 - ▣ Allows the receiver to apply flow control to the sender
 - ▣ Supports multiple application processes on each host

End-to-end Protocols

4

- Typical limitations of the network on which transport protocol will operate
 - Drop message
 - Reorder messages
 - Deliver duplicate copies of a given message
 - Limit messages to some finite size
 - Deliver messages after an arbitrarily long delay

End-to-end Protocols

5

- Challenge for Transport Protocols
 - Develop algorithms that turn the less-than-desirable properties of the underlying network into the high level of service required by application programs

Simple Demultiplexer (UDP)

6

- Extends host-to-host deliver service of the underlying network into a process-to-process communication service
- Add a level of demultiplexing which allows multiple application processes on each host to share the network

Simple Demultiplexer (UDP)

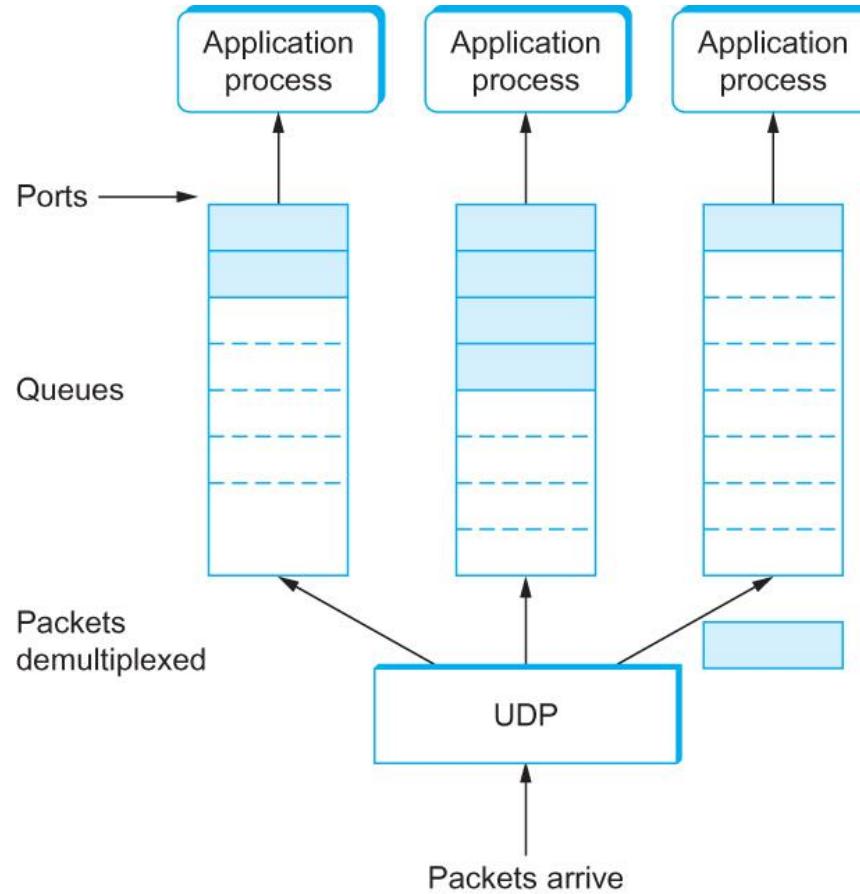
7



Format for UDP header (Note: length and checksum fields should be switched)

Simple Demultiplexer (UDP)

8



UDP Message Queue

Reliable Byte Stream (TCP)

9

- In contrast to UDP, Transmission Control Protocol (TCP) offers the following services
 - Reliable
 - Connection oriented
 - Byte-stream service

Flow Control Vs Congestion Control

10

- Flow control involves preventing senders from overrunning the capacity of the receivers
- Congestion control involves preventing too much data from being injected into the network, thereby causing switches or links to become overloaded

End-to-end Issues

11

- At the heart of the TCP is the sliding window algorithm
- As the TCP runs over the Internet rather than a point-to-point link, the following issues need to be addressed by the sliding window algorithm
 - ▣ TCP supports logical connections between processes that are running on two different computers in the Internet
 - ▣ TCP connections are likely to have widely different RTT time
 - ▣ Packets may get reordered in the Internet

End-to-end Issues

12

- TCP needs a mechanism using which each side of a connection will learn what resources the other side is able to apply to the connection
- TCP needs a mechanism using which the sending side will learn the capacity of the network

TCP Segment

13

- TCP is a byte-oriented protocol, which means that the sender writes bytes into a TCP connection and the receiver reads bytes out of the TCP connection

- Although “byte stream” describes the service TCP offers to application processes, TCP does not, itself, transmit individual bytes over the Internet

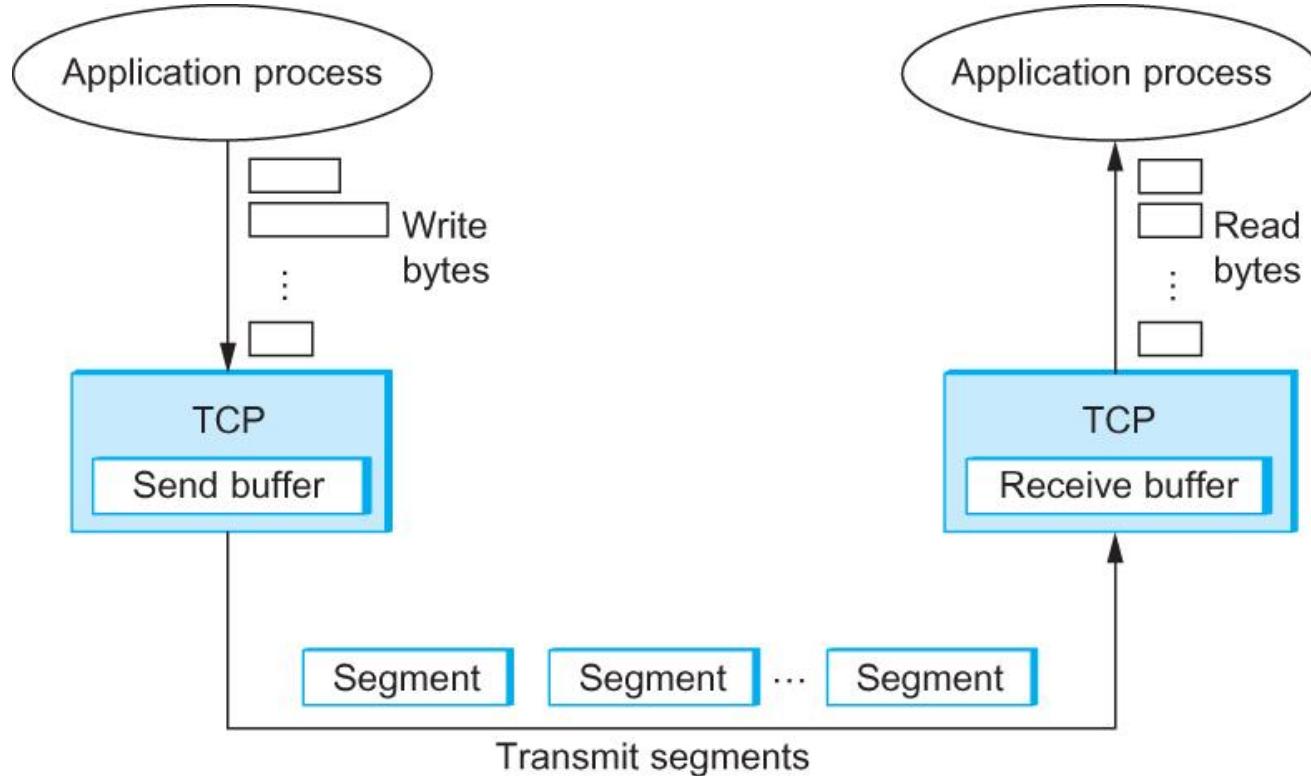
TCP Segment

14

- TCP on the source host buffers enough bytes from the sending process to fill a reasonably sized packet and then sends this packet to its peer on the destination host
- TCP on the destination host then empties the contents of the packet into a receive buffer, and the receiving process reads from this buffer at its leisure
- The packets exchanged between TCP peers are called segments.

TCP Segment

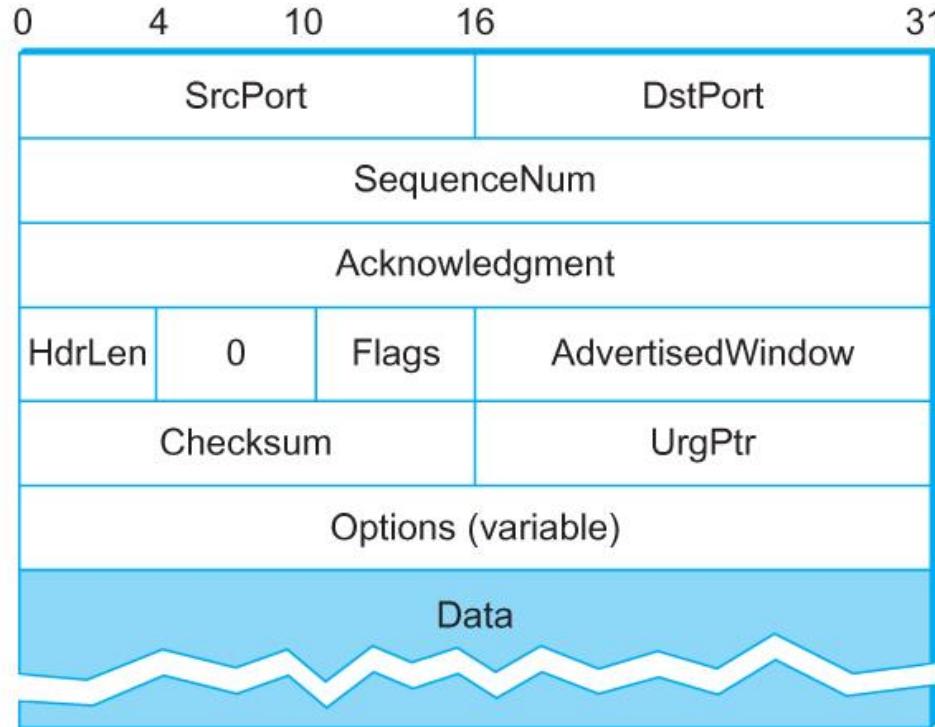
15



How TCP manages a byte stream.

TCP Header

16



TCP Header Format

TCP Header

17

- The SrcPort and DstPort fields identify the source and destination ports, respectively.
- The Acknowledgment, SequenceNum, and AdvertisedWindow fields are all involved in TCP's sliding window algorithm.
- Because TCP is a byte-oriented protocol, each byte of data has a sequence number; the SequenceNum field contains the sequence number for the first byte of data carried in that segment.
- The Acknowledgment and AdvertisedWindow fields carry information about the flow of data going in the other direction.

TCP Header

18

- The 6-bit Flags field is used to relay control information between TCP peers.
- The possible flags include SYN, FIN, RESET, PUSH, URG, and ACK.
- The SYN and FIN flags are used when establishing and terminating a TCP connection, respectively.
- The ACK flag is set any time the Acknowledgment field is valid, implying that the receiver should pay attention to it.

TCP Header

19

- The URG flag signifies that this segment contains urgent data. When this flag is set, the UrgPtr field indicates where the nonurgent data contained in this segment begins.
- The urgent data is contained at the front of the segment body, up to and including a value of UrgPtr bytes into the segment.
- The PUSH flag signifies that the sender invoked the push operation, which indicates to the receiving side of TCP that it should notify the receiving process of this fact.
- Finally, the RESET flag signifies that the receiver has become confused

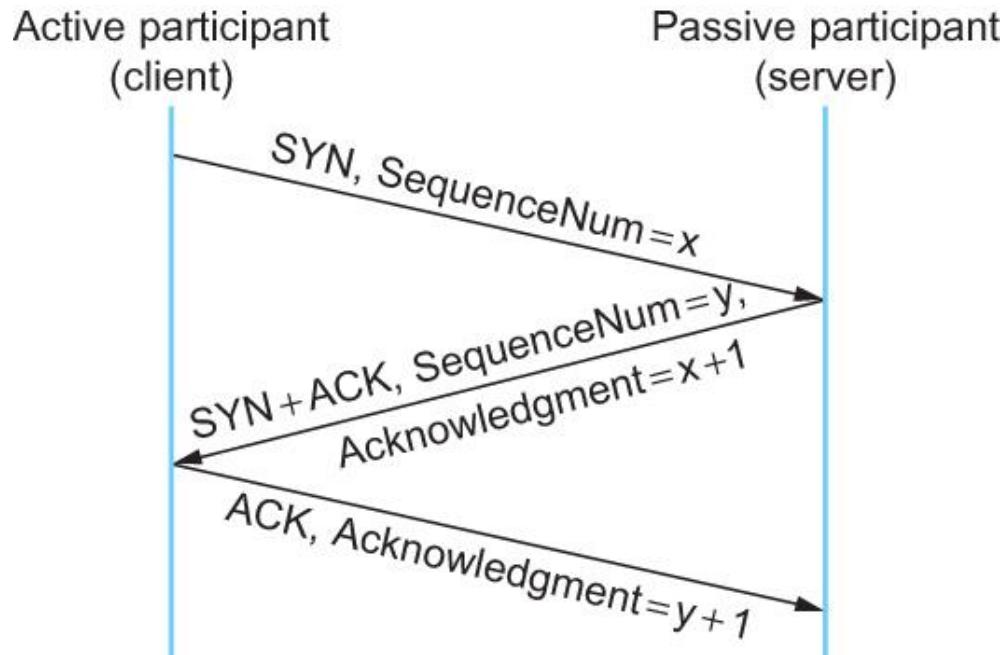
TCP Header

20

- Finally, the RESET flag signifies that the receiver has become confused, it received a segment it did not expect to receive—and so wants to abort the connection.
- Finally, the Checksum field is used in exactly the same way as for UDP—it is computed over the TCP header, the TCP data, and the pseudoheader, which is made up of the source address, destination address, and length fields from the IP header.

Connection Establishment/Termination in TCP

21



Timeline for three-way handshake algorithm

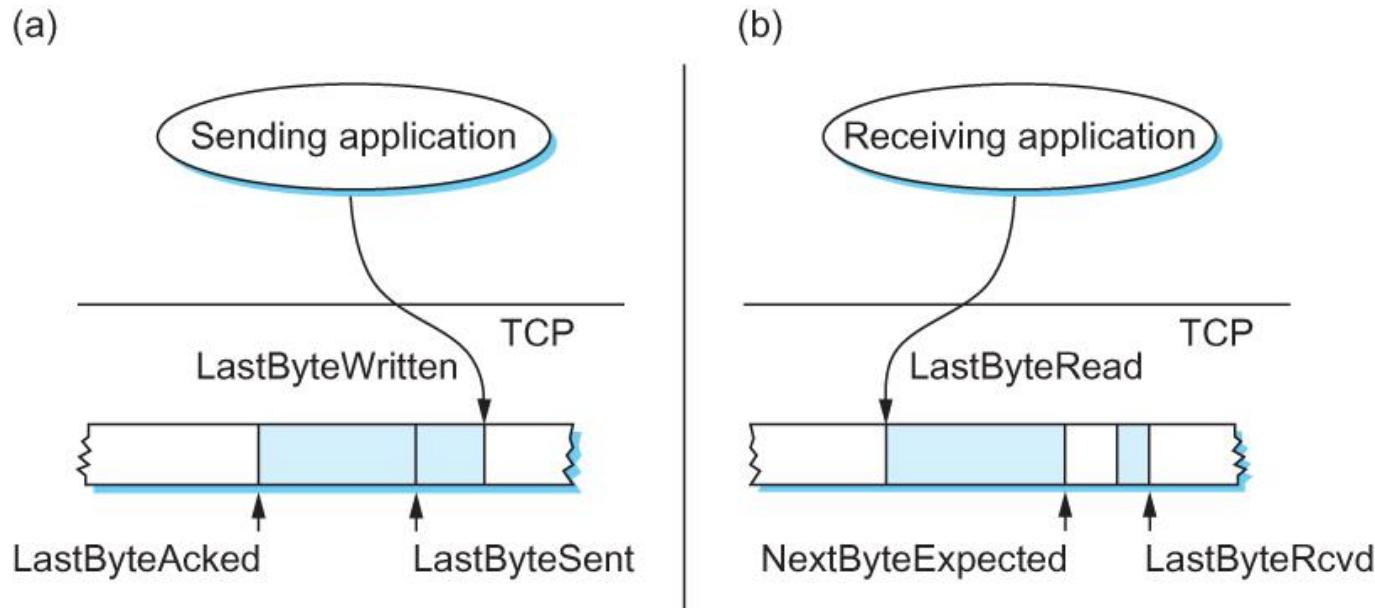
Sliding Window Revisited

22

- TCP's variant of the sliding window algorithm, which serves several purposes:
 - (1) It guarantees the reliable delivery of data
 - (2) It ensures that data is delivered in order, and
 - (3) It enforces flow control between the sender and the receiver

Sliding Window Revisited

23



Relationship between TCP send buffer (a) and receive buffer (b).

TCP Sliding Window

24

- Sending Side
 - $\text{LastByteAcked} \leq \text{LastByteSent}$
 - $\text{LastByteSent} \leq \text{LastByteWritten}$
- Receiving Side
 - $\text{LastByteRead} < \text{NextByteExpected}$
 - $\text{NextByteExpected} \leq \text{LastByteRcvd} + 1$

TCP Flow Control

25

- $\text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxRcvBuffer}$
- $\text{AdvertisedWindow} = \text{MaxRcvBuffer} - ((\text{NextByteExpected} - 1) - \text{LastByteRead})$
- $\text{LastByteSent} - \text{LastByteAcked} \leq \text{AdvertisedWindow}$
- $\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$
- $\text{LastByteWritten} - \text{LastByteAcked} \leq \text{MaxSendBuffer}$
- If the sending process tries to write y bytes to TCP, but $(\text{LastByteWritten} - \text{LastByteAcked}) + y > \text{MaxSendBuffer}$ then TCP blocks the sending process and does not allow it to generate more data.

Protecting against Wraparound

26

- SequenceNum: 32 bits longs
- AdvertisedWindow: 16 bits long
 - ▣ TCP has satisfied the requirement of the sliding Window algorithm that is the sequence number Space be twice as big as the window size
 - ▣ $2^{32} \gg 2 \times 2^{16}$

Protecting Against Wraparound

27

- Relevance of the 32-bit sequence number space
 - The sequence number used on a given connection might wraparound
 - A byte with sequence number x could be sent at one time, and then at a later time a second byte with the same sequence number x could be sent
 - Packets cannot survive in the Internet for longer than the MSL (maximum segment lifetime)
 - MSL is set to 120sec
 - We need to make sure that the sequence number does not wrap around within a 120-second period of time
 - Depends on how fast data can be transmitted over the Internet

Protecting against Wraparound

28

Bandwidth	Time until Wraparound
T1 (1.5 Mbps)	6.4 hours
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
Fast Ethernet (100 Mbps)	6 minutes
OC-3 (155 Mbps)	4 minutes
OC-12 (622 Mbps)	55 seconds
OC-48 (2.5 Gbps)	14 seconds

Time until 32-bit sequence number space wraps around.

Keeping the Pipe Full

29

- 16-bit advertised window field must be big enough to allow the sender to keep the pipe full
- Clearly the receiver is free not to open the window as large as the AdvertisedWindow field allows
- If the receiver has enough buffer space
 - The window needs to be opened far enough to allow a full
 - Delay x bandwidth product's worth of data
 - Assuming an RTT of 100 ms

Keeping the Pipe Full

30

Bandwidth	Delay × Bandwidth Product
T1 (1.5 Mbps)	18 KB
Ethernet (10 Mbps)	122 KB
T3 (45 Mbps)	549 KB
Fast Ethernet (100 Mbps)	1.2 MB
OC-3 (155 Mbps)	1.8 MB
OC-12 (622 Mbps)	7.4 MB
OC-48 (2.5 Gbps)	29.6 MB

Required window size for 100-ms RTT

Triggering Transmission

31

- How does TCP decide to transmit a segment?
 - TCP supports a byte stream abstraction
 - Application programs write bytes into streams
 - It is up to TCP to decide that it has enough bytes to send a segment

Triggering Transmission

32

- What factors governs this decision
 - Ignore flow control: window is wide open, as would be the case when the connection starts
 - TCP has three mechanism to trigger the transmission of a segment
 - (1) TCP maintains a variable MSS and sends a segment as soon as it has collected MSS bytes from the sending process
 - MSS is usually set to the size of the largest segment TCP can send without causing local IP to fragment
 - MSS: MTU of directly connected network – (TCP header + and IP header)
 - (2) Sending process has explicitly asked TCP to send it
 - TCP supports push operation
 - (3) When a timer fires
 - Resulting segment contains as many bytes as are currently buffered for transmission

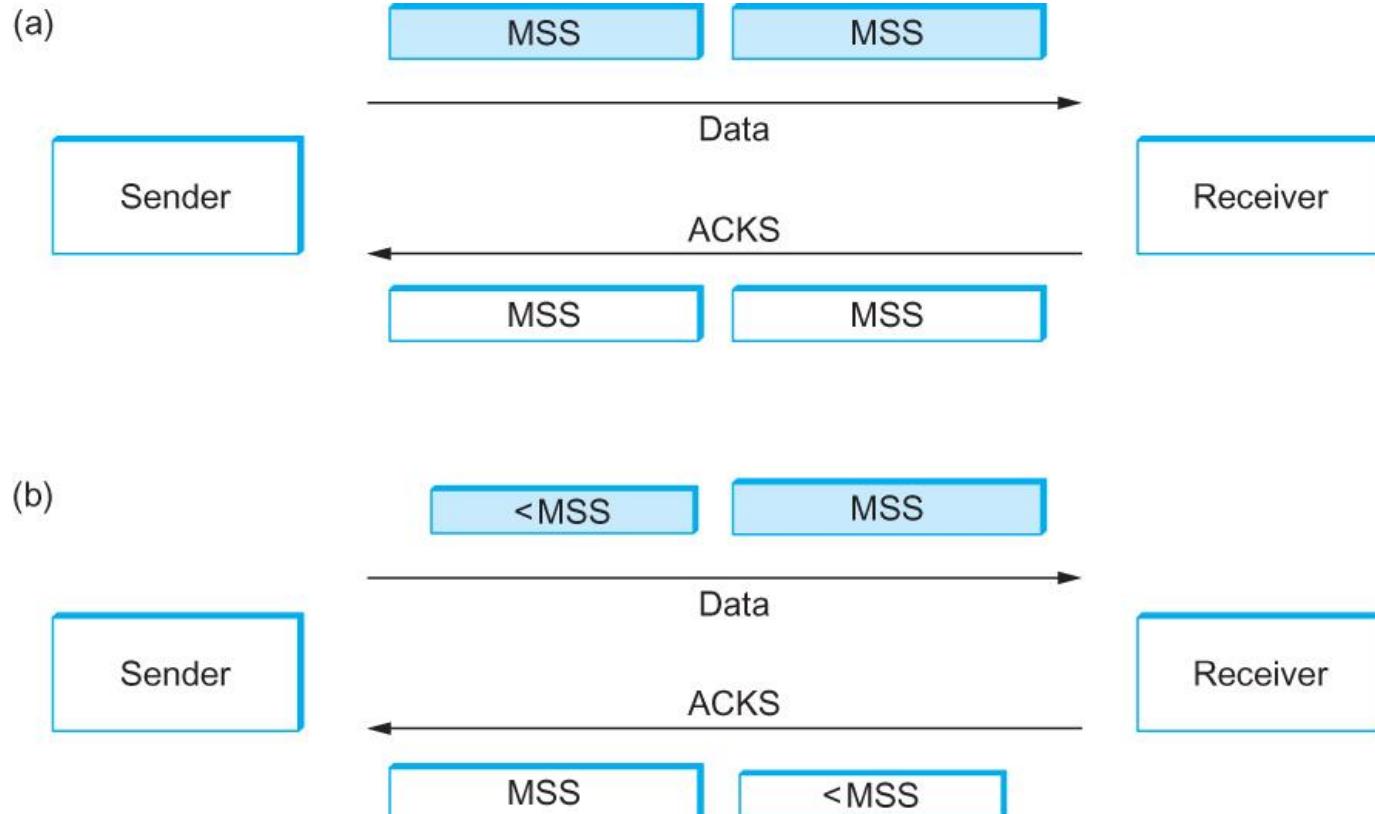
Silly Window Syndrome

33

- If you think of a TCP stream as a conveyer belt with “full” containers (data segments) going in one direction and empty container (ACKs) going in the reverse direction, then MSS-sized segments correspond to large container and 1-byte segments correspond to very small containers
- If the sender aggressively fills an empty container as soon as it arrives, then any small container introduced into the system remains in the system indefinitely
- That is, it is immediately filled and emptied at each end, and never coalesced with adjacent containers to create large containers

Silly Window Syndrome

34



Silly Window Syndrome

Nagle's Algorithm

35

- If there is data to send but the window is open less than MSS, then we may want to wait some amount of time before sending the available data
- But how long?
- If we wait too long, then we hurt interactive applications like Telnet
- If we don't wait long enough, then we risk sending a bunch of tiny packets and falling into the silly window syndrome
 - ▣ The solution is to introduce a timer and to transmit when the timer expires

Nagle's Algorithm

36

- We could use a clock-based timer, for example one that fires every 100ms
- Nagle introduced an elegant self-clocking solution
- Key Idea
 - ▣ As long as TCP has any data in flight, the sender will eventually receive an ACK
 - ▣ This ACK can be treated like a timer firing, triggering the transmission of more data

Nagle's Algorithm

37

- When the application produces data to send
 - If both the available data and the window \geq MSS
 - Send a full segment
 - Else
 - If there is unACKed data in flight
 - Buffer the new data until an ACK arrives
 - Else
 - Send all the new data now

Adaptive Retransmission

38

- Original Algorithm
 - Measure SampleRTT for each segment/ACK pair
 - Compute weighted average of RTT
 - $\text{EstRTT} = \alpha \times \text{EstRTT} + (1 - \alpha) \times \text{SampleRTT}$
 - α between 0.8 and 0.9
 - Set timeout based on EstRTT
 - TimeOut = 2 x EstRTT

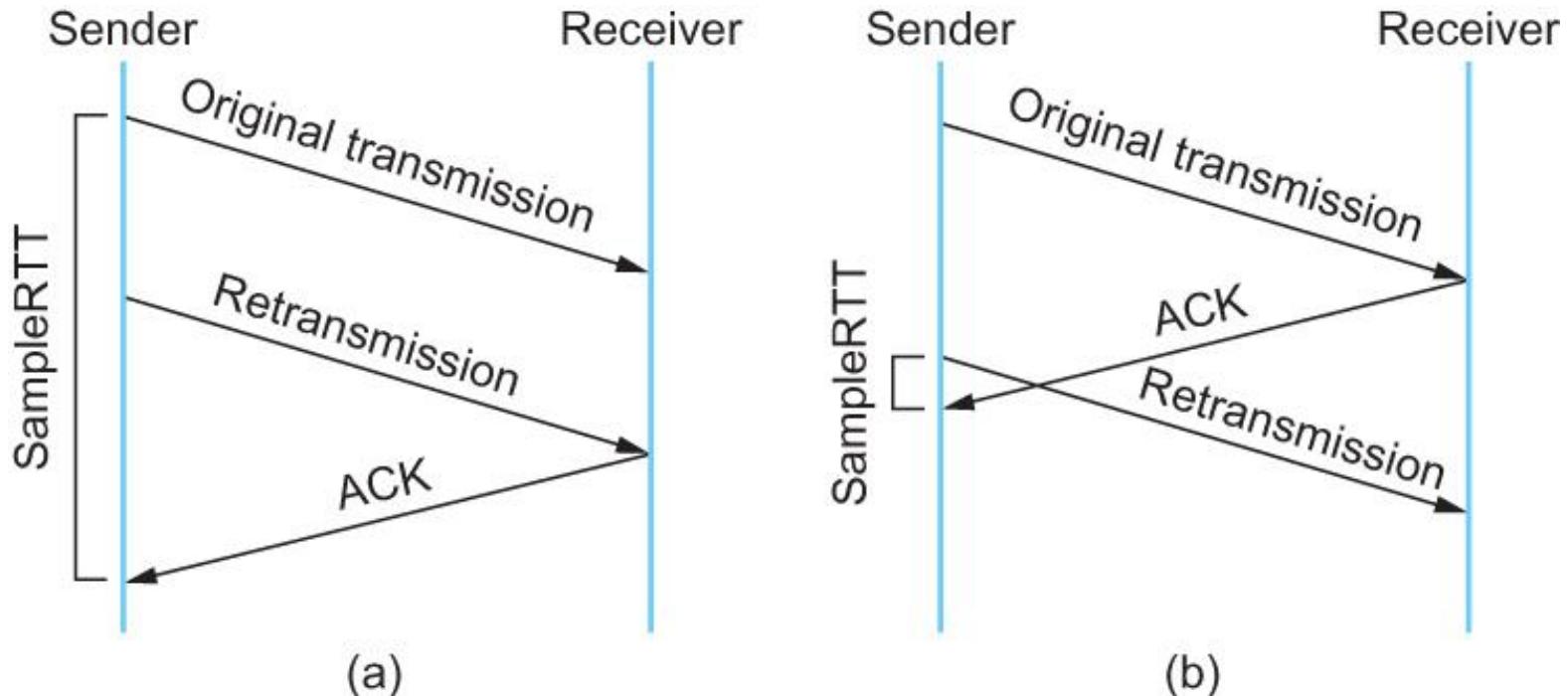
Original Algorithm

39

- Problem
 - ACK does not really acknowledge a transmission
 - It actually acknowledges the receipt of data
 - When a segment is retransmitted and then an ACK arrives at the sender
 - It is impossible to decide if this ACK should be associated with the first or the second transmission for calculating RTTs

Karn/Partridge Algorithm

40



Associating the ACK with (a) original transmission versus (b) retransmission

Karn/Partridge Algorithm

41

- Do not sample RTT when retransmitting
- Double timeout after each retransmission

Karn/Partridge Algorithm

42

- Karn-Partridge algorithm was an improvement over the original approach, but it does not eliminate congestion
- We need to understand how timeout is related to congestion
 - If you timeout too soon, you may unnecessarily retransmit a segment which adds load to the network

Karn/Partridge Algorithm

43

- Main problem with the original computation is that does not take variance of sample RTTs into consideration
- If the variance among sample RTTs is small
 - ▣ Then the estimated RTT can be better trusted
 - ▣ There is no need to multiply this by 2 to compute the timeout

Karn/Partridge Algorithm

44

- On the other hand, a large variance in the samples suggest that timeout value should not be tightly coupled to the Estimated RTT
- Jacobson/Karels proposed a new scheme for TCP retransmission

Jacobson/Karels Algorithm

45

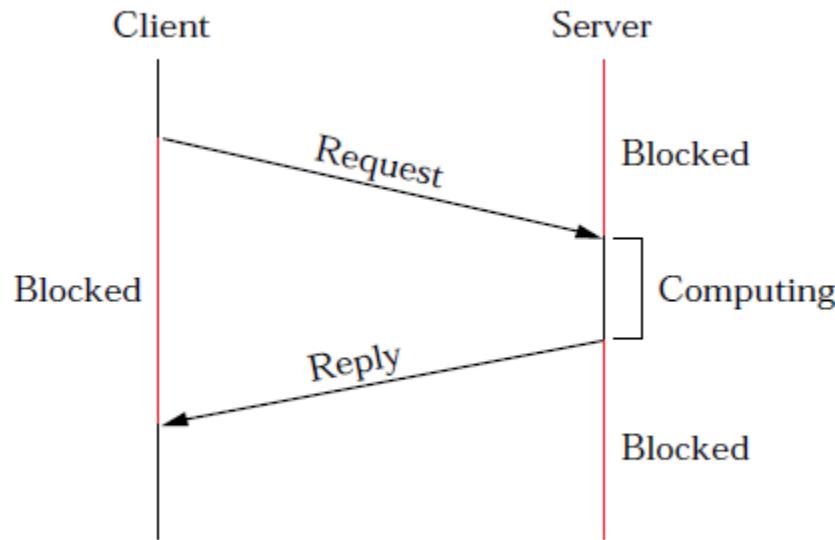
- Difference = SampleRTT – EstimatedRTT
- EstimatedRTT = EstimatedRTT + (x Difference)
- Derivation = Derivation + (|Difference| - Deviation)
- TimeOut = $\mu \times \text{EstimatedRTT} + x \times \text{Deviation}$
 - ▣ Where based on experience, μ is typically set to 1 and is set to 4. Thus, when the variance is small, TimeOut is close to EstimatedRTT; a large variance causes the deviation term to dominate the calculation

46

RPC

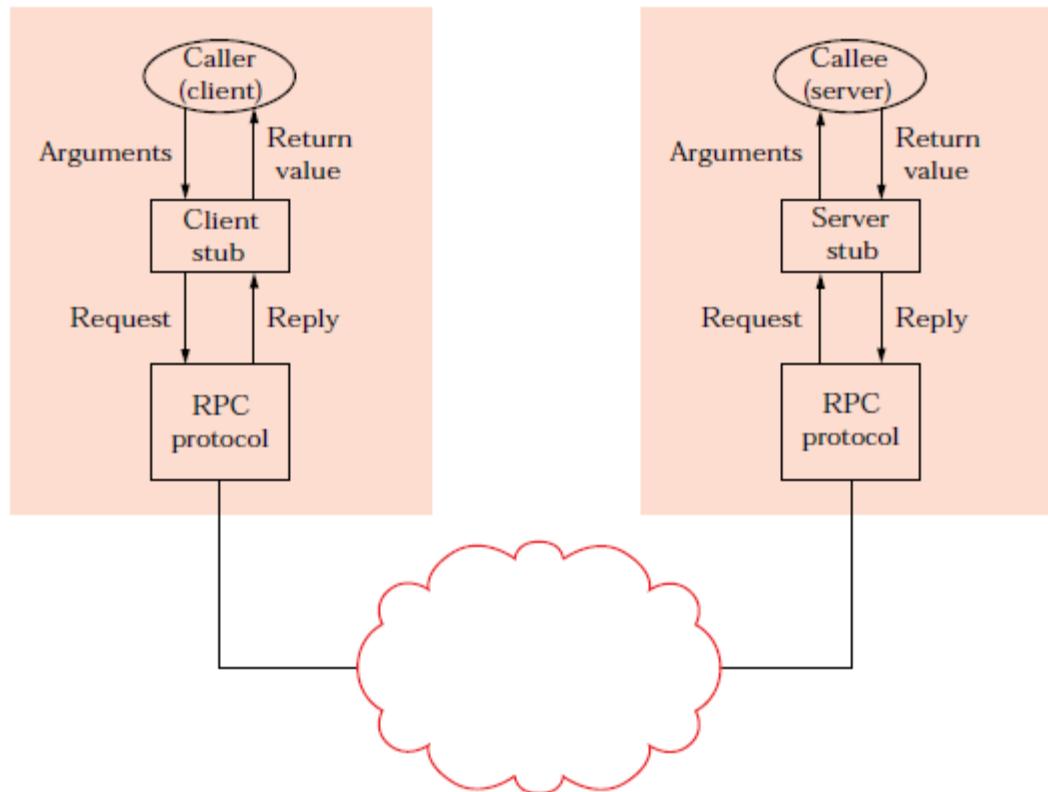
RPC

47



RPC

48



RPC Functions

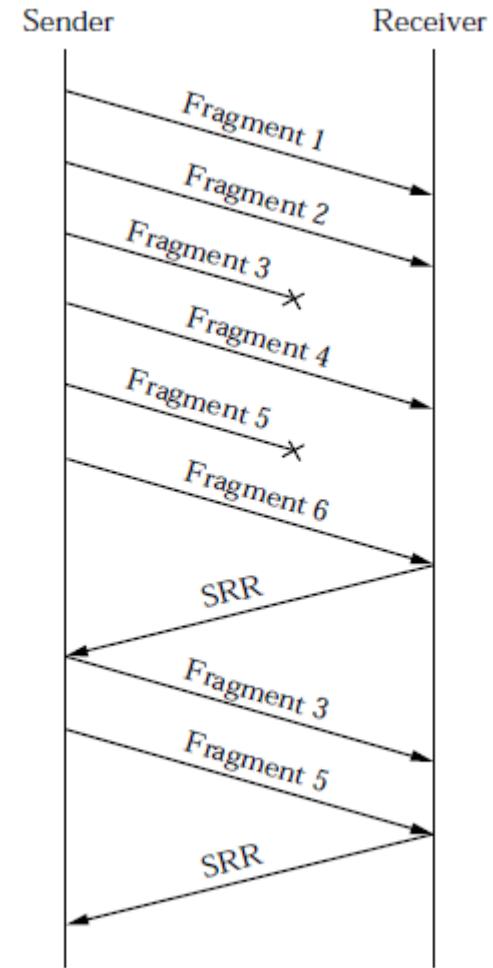
49

- RPC protocol consists of three basic functions
 - BLAST: fragments and reassembles large messages
 - CHAN: synchronizes request and reply messages
 - SELECT: dispatches request messages to the correct process

Bulk Transfer (BLAST)

50

- Unlike AAL and IP in that it tries to recover from lost fragments; persistent, but does not guarantee delivery. Strategy is to use *selective retransmission (partial acknowledgements)*



Bulk Transfer (BLAST)

51

- Sender:
 - After sending all fragments, set timer DONE
 - If receive SRR, send missing fragments and reset DONE
 - If timer DONE expires, free fragments
- Receiver:
 - When first fragment arrives, set timer LAST_FRAG
 - When all fragments present, reassemble and pass up

Bulk Transfer (BLAST)

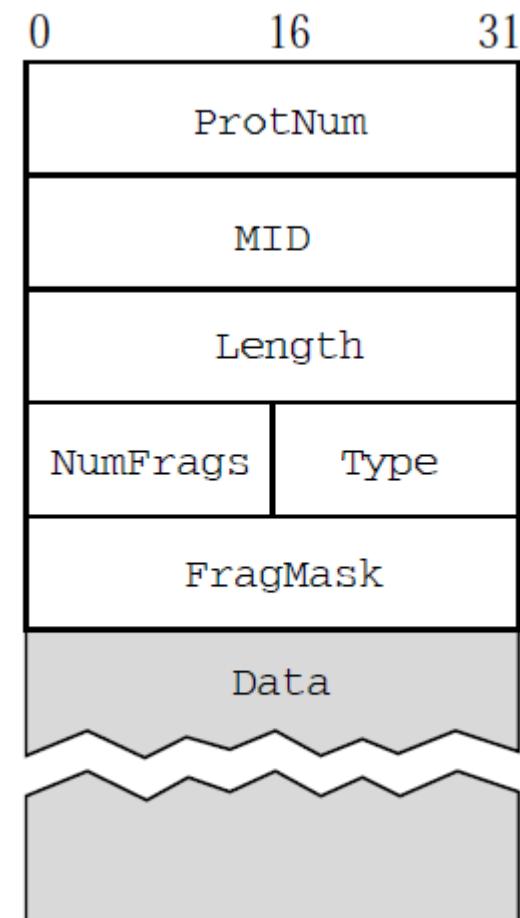
52

- Four exceptional conditions
 - if last fragment arrives but message not complete
 - send SRR and set timer RETRY
 - if timer LAST_FRAG expires
 - send SRR and set timer RETRY
 - if timer RETRY expires for first or second time
 - send SRR and set timer RETRY
 - if timer RETRY expires for third time
 - give up and free partial message

BLAST Header Format

53

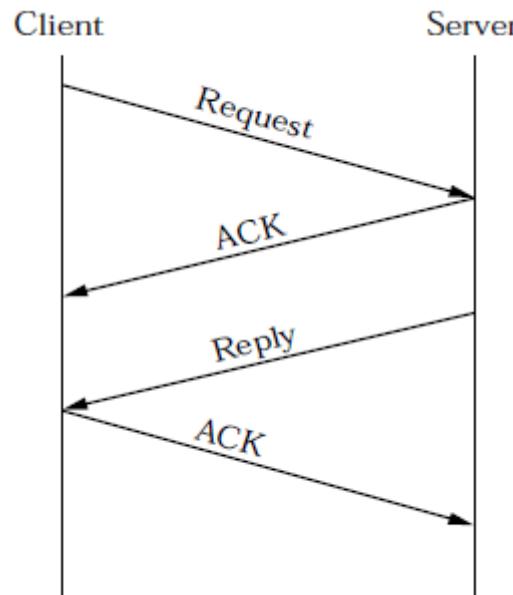
- MID must protect against wrap around
- Type = DATA or SRR
- NumFrags indicates number of fragments in message
- FragMask distinguishes among fragments:
 - ▣ if Type=DATA, identifies this fragment
 - ▣ if Type=SRR, identifies missing fragments



Request/Reply (CHAN)

54

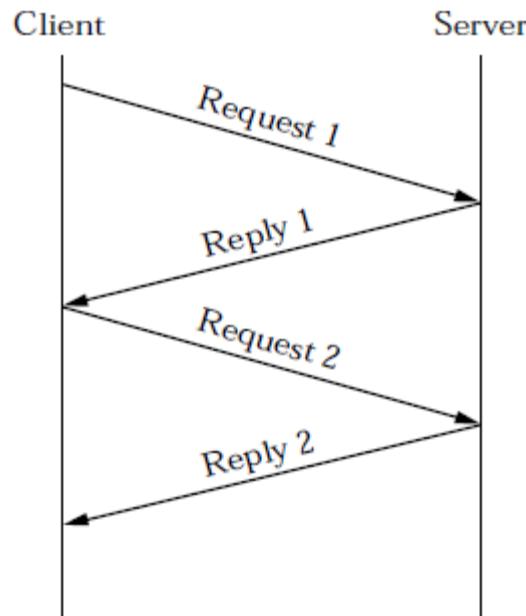
- ❑ Guarantees message delivery, and synchronizes client with server; i.e., blocks client until reply received. Supports *at-most-once semantics*
- ❑ Simple case



Request/Reply (CHAN)

55

□ Implicit Acknowledgements



CHAN – Complications

56

- Lost message (request, reply, or ACK)
 - set RETRANSMIT timer
 - use message id (MID) field to distinguish
- Slow (long running) server
 - client periodically sends “are you alive” probe, or
 - server periodically sends “I'm alive” notice
- Want to support multiple outstanding calls
 - use channel id (CID) field to distinguish
- Machines crash and reboot
 - use boot id (BID) field to distinguish

CHAN Header Format

57

```
typedef struct {  
    u_short Type;      /* REQ, REP, ACK, PROBE */  
    u_short CID;       /* unique channel id */  
    int MID;           /* unique message id */  
    int BID;            /* unique boot id */  
    int Length;         /* length of message */  
    int ProtNum;        /* high-level protocol */  
} ChanHdr;
```

CHAN Session State

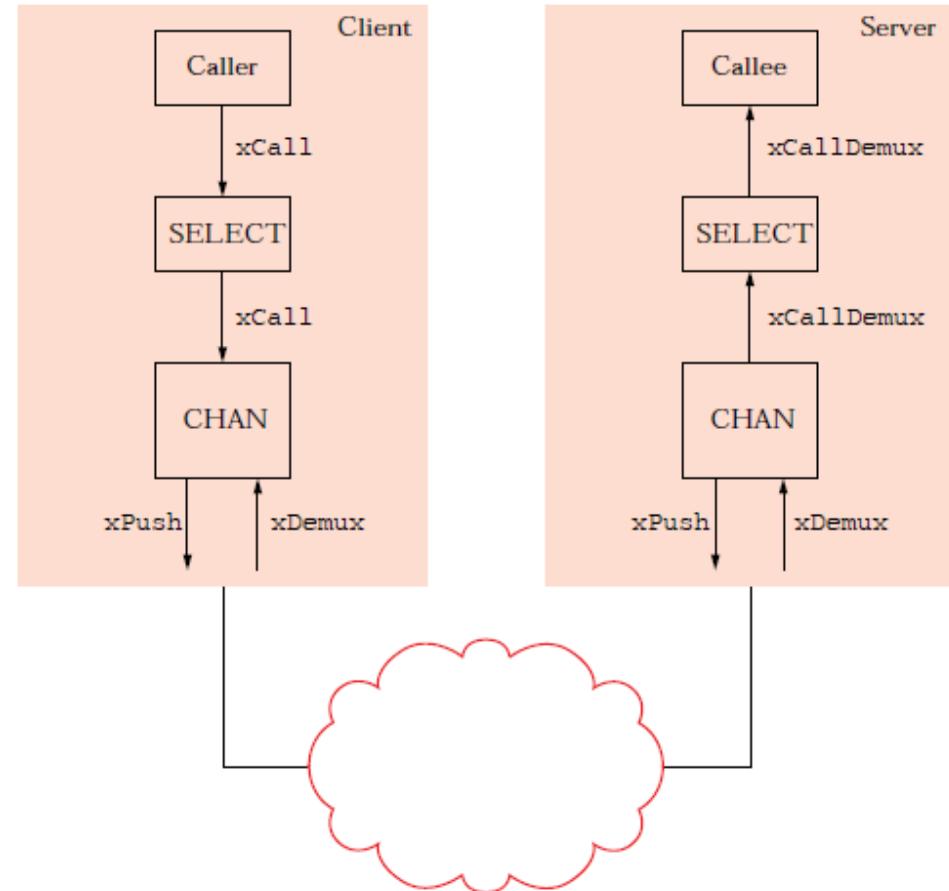
58

```
typedef struct {
    u_char type;          /* CLIENT or SERVER */
    u_char status;         /* BUSY or IDLE */
    int retries;           /* number of retries */
    int timeout;           /* timeout value */
    XkReturn ret_val;      /* return value */
    Msg *request;          /* request message */
    Msg *reply;             /* reply message */
    Semaphore reply_sem;   /* client semaphore */
    int mid;                /* message id */
    int bid;                /* boot id */
} ChanState;
```

Dispatcher (SELECT)

59

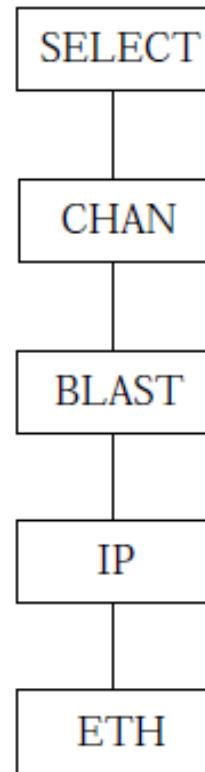
- Dispatches request messages to the appropriate procedure; fully synchronous counterpart to UDP
- Address Space for Procedures
 - Flat: unique id for each possible procedure
 - Hierarchical: program + procedure within program



Putting it All Together

60

□ Simple RPC Stack



Reference

61

- Chapter – 5: *Computer Networks A Systems Approach* by Larry L. Peterson and Bruce S. Davie, 4Th Edition, Morgan Kaufmann Publications.

CONGESTION CONTROL AND RESOURCE ALLOCATION

11 November
2023

Dr Noor Mohammad Sk

Congestion Control and Resource Allocation

2

- Resources
 - Bandwidth of the links
 - Buffers at the routers and switches
- Packets contend at a router for the use of a link, with each contending packet placed in a queue waiting for its turn to be transmitted over the link

Congestion Control and Resource Allocation

3

- When too many packets are contending for the same link
 - The queue overflows
 - Packets get dropped
 - Network is congested!
- Network should provide a congestion control mechanism to deal with such a situation

Congestion Control and Resource Allocation

4

- Congestion control and Resource Allocation
 - Two sides of the same coin
- If the network takes active role in allocating resources
 - The congestion may be avoided
 - No need for congestion control

Congestion Control and Resource Allocation

5

- Allocating resources with any precision is difficult
 - ▣ Resources are distributed throughout the network
- On the other hand, we can always let the sources send as much data as they want
 - ▣ Then recover from the congestion when it occurs
 - ▣ Easier approach but it can be disruptive because **many packets many be discarded** by the network before congestions can be controlled

Congestion Control and Resource Allocation

6

- Congestion control and resource allocations involve both hosts and network elements such as routers
- In network elements
 - Various queuing disciplines can be used to control the order in which packets get transmitted and which packets get dropped
- At the hosts' end
 - The congestion control mechanism paces how fast sources are allowed to send packets

Issues in Resource Allocation

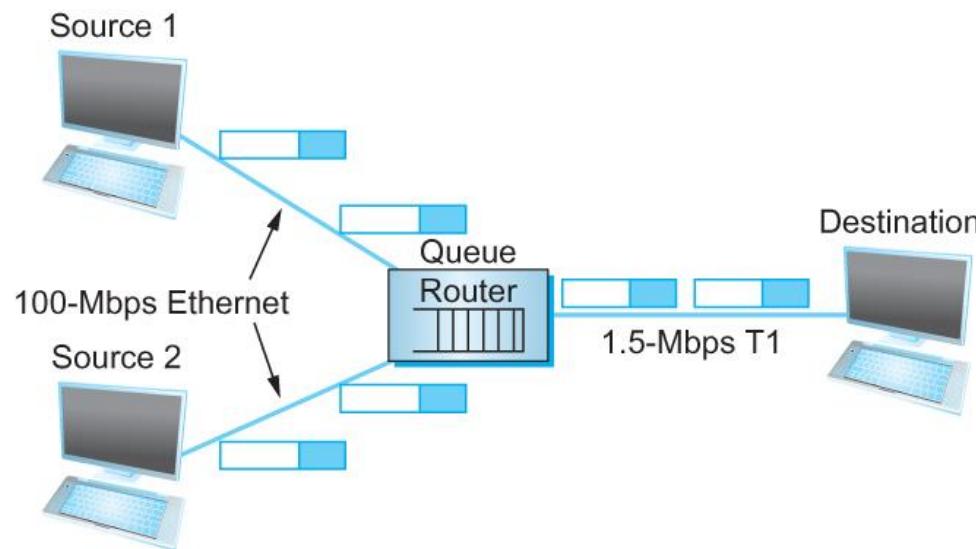
7

- Network Model – Packet Switched Network
 - We consider resource allocation in a packet switched network (or internet) consisting of multiple links and switches (or routers)
 - In such an environment
 - a given source may have more than enough capacity on the immediate outgoing link to send a packet,
 - but somewhere in the middle of a network, its packets encounter a link that is being used by many different traffic sources

Issues in Resource Allocation

8

□ Network Model – Packet Switched Network



A potential bottleneck router.

Issues in Resource Allocation

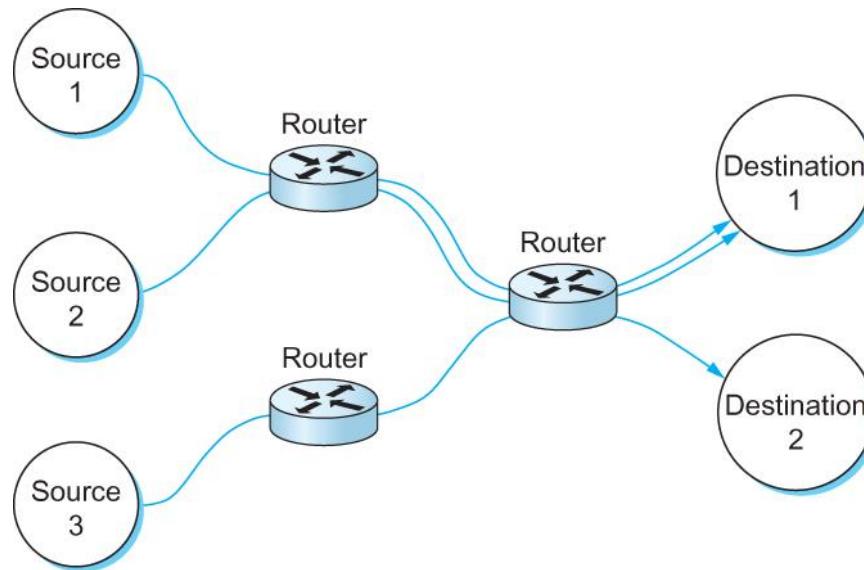
9

- **Network Model – Connectionless flows**
- For much of our discussion, we assume that the network is essentially connectionless, with any connection-oriented service implemented in the transport protocol that is running on the end hosts.
 - ▣ A connectionless network is too strong when all the datagrams are completely independent
 - ▣ The datagrams are certainly switched independently, but it is usually the case that a stream of datagrams between a particular pair of hosts flows through a particular set of routers

Issues in Resource Allocation

10

□ Network Model – Connectionless flows



Multiple flows passing through a set of routers

Issues in Resource Allocation

11

□ Network Model – Connectionless flows

- One of the powers of the flow abstraction is that flows can be defined at different granularities.
 - For example, a flow can be host-to-host (i.e., have the same source/destination host addresses) or process-to-process (i.e., have the same source/destination host/port pairs).
- In the latter case, a flow is essentially the same as a channel. The flow is visible to the routers inside the network, whereas a channel is an end-to-end abstraction.

Issues in Resource Allocation

12

□ Network Model – Connectionless flows

- Because multiple related packets flow through each router, it sometimes makes sense to maintain some state information for each flow, information that can be used to make resource allocation decisions about the packets that belong to the flow. This state is sometimes called **soft state**.
- The main difference between soft state and “hard” state is that soft state need not always be explicitly created and removed by signalling.

Issues in Resource Allocation

13

□ Network Model – Connectionless flows

- Soft state represents a middle ground between a purely connectionless network that maintains no state at the routers and a purely connection-oriented network that maintains hard state at the routers.
- In general, the correct operation of the network does not depend on soft state being present (**each packet is still routed correctly without regard to this state**), but when a packet happens to belong to a flow for which the router is currently maintaining soft state, then the router is better able to handle the packet.

Issues in Resource Allocation

14

- Taxonomy
- Router-centric versus Host-centric
 - In a router-centric design,
 - Each router takes responsibility for deciding when packets are forwarded and
 - Selecting which packets are to be dropped,
 - As well as for informing the hosts that are generating the network traffic, how many packets they are allowed to send.
 - In a host-centric design,
 - The end hosts observe the network conditions (e.g., how many packets they are successfully getting through the network) and
 - Adjust their behavior accordingly.
 - Note that these two groups are not mutually exclusive.

Issues in Resource Allocation

15

- Taxonomy
- Reservation-based versus Feedback-based
 - In a reservation-based system, some entity (e.g., the end host) asks the network for a certain amount of capacity to be allocated for a flow.
 - Each router then allocates enough resources (buffers and/or percentage of the link's bandwidth) to satisfy this request.
 - If the request cannot be satisfied at some router, because doing so would overcommit its resources, then the router rejects the reservation.
 - In a feedback-based approach, the end hosts begin sending data without first reserving any capacity and then adjust their sending rate according to the feedback they receive.
 - This feedback can either be *explicit* (i.e., a congested router sends a “please slow down” message to the host) or
 - It can be *implicit* (i.e., the end host adjusts its sending rate according to the externally observable behavior of the network, such as packet losses).

Issues in Resource Allocation

16

- Taxonomy
- Window-based versus Rate-based
 - Window advertisement is used within the network to reserve buffer space.
 - Control sender's behavior using a rate, how many bit per second the receiver or network is able to absorb

Issues in Resource Allocation

17

□ Evaluation Criteria – Effective Resource Allocation

- A good starting point for evaluating the effectiveness of a resource allocation scheme is to consider the two principal metrics of networking: throughput and delay.
 - Clearly, we want as much throughput and as little delay as possible.
 - Unfortunately, these goals are often somewhat at odds with each other.
 - One sure way for a resource allocation algorithm to increase throughput is to allow as many packets into the network as possible, so as to drive the utilization of all the links up to 100%.
 - We would do this to avoid the possibility of a link becoming idle because an idle link necessarily hurts throughput.
 - The problem with this strategy is that increasing the number of packets in the network also increases the length of the queues at each router. Longer queues, in turn, mean packets are delayed longer in the network

Issues in Resource Allocation

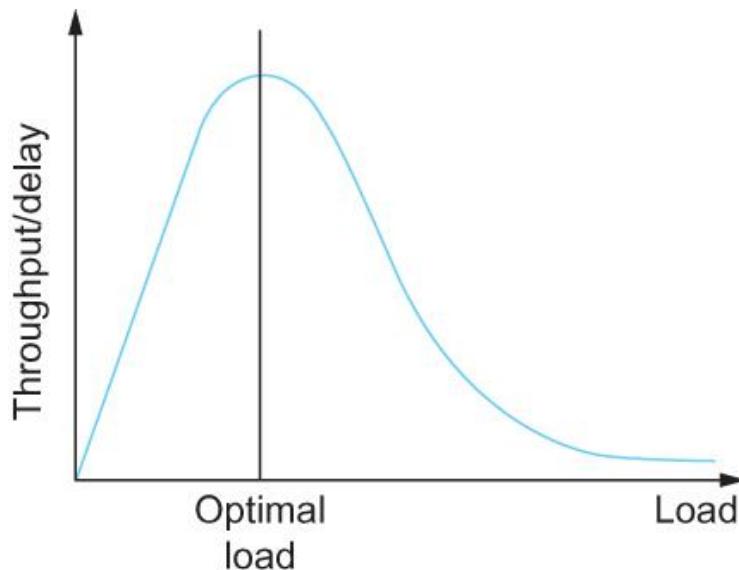
18

- Evaluation Criteria - Effective Resource Allocation
 - To describe this relationship, some network designers have proposed using the ratio of throughput to delay as a metric for evaluating the effectiveness of a resource allocation scheme.
 - This ratio is sometimes referred to as the *power of the network*.
 - Power = Throughput/Delay

Issues in Resource Allocation

19

□ Evaluation Criteria - Effective Resource Allocation



Ratio of throughput to delay as a function of load

Issues in Resource Allocation

20

□ Evaluation Criteria - Fair Resource Allocation

- The effective utilization of network resources is not the only criterion for judging a resource allocation scheme.
- We must also consider the **issue of fairness**. However, we quickly get into murky waters when we try to define what exactly constitutes fair resource allocation.
- For example, a reservation-based resource allocation scheme provides an explicit way to create controlled unfairness.
- With such a scheme, we might use reservations to enable a video stream to receive 1 Mbps across some link while a file transfer receives only 10 Kbps over the same link

Issues in Resource Allocation

21

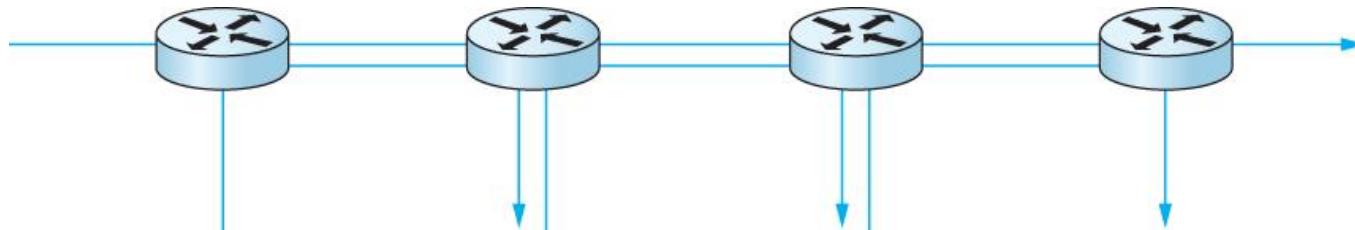
□ Evaluation Criteria - Fair Resource Allocation

- In the absence of explicit information to the contrary, when several flows share a particular link, we would like for each flow to receive an equal share of the bandwidth.
- This definition presumes that a *fair share of bandwidth means an equal share of bandwidth*.
- But even in the absence of reservations, equal shares may not equate to fair shares.
- Should we also consider the length of the paths being compared?
 - Consider the figure in next slide

Issues in Resource Allocation

22

- Evaluation Criteria
 - Fair Resource Allocation



One four-hop flow competing with three one-hop flows

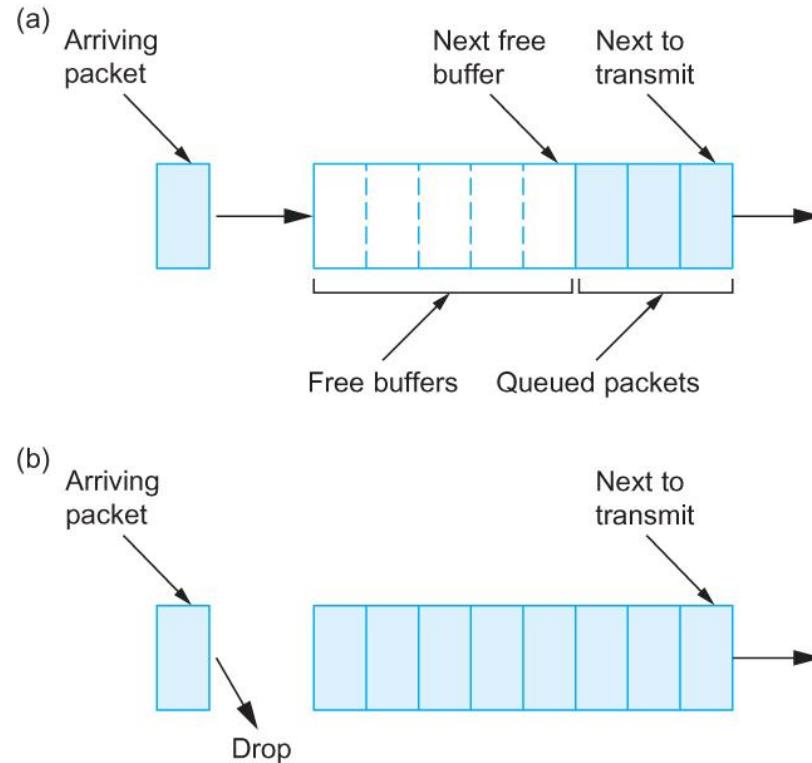
Queuing Disciplines

23

- The idea of FIFO queuing, also called first-come-first-served (FCFS) queuing, is simple:
 - The first packet that arrives at a router is the first packet to be transmitted
 - Given that the amount of buffer space at each router is finite, if a packet arrives and the queue (buffer space) is full, then the router discards that packet
 - This is done without regard to which flow the packet belongs to or how important the packet is. This is sometimes called *tail drop*, since *packets that arrive at the tail end of the FIFO are dropped*
 - Note that tail drop and FIFO are two separable ideas. FIFO is a *scheduling discipline*—it determines the order in which packets are transmitted. Tail drop is a *drop policy*—it determines which packets get dropped

Queuing Disciplines

24



(a) FIFO queuing; (b) tail drop at a FIFO queue.

Queuing Disciplines

25

- A simple variation on basic FIFO queuing is priority queuing.
 - ▣ The idea is to mark each packet with a priority; the mark could be carried, for example, in the IP header.
- The routers then implement multiple FIFO queues, one for each priority class.
 - ▣ The router always transmits packets out of the highest-priority queue if that queue is nonempty before moving on to the next priority queue.
- Within each priority, packets are still managed in a FIFO manner.

Queuing Disciplines

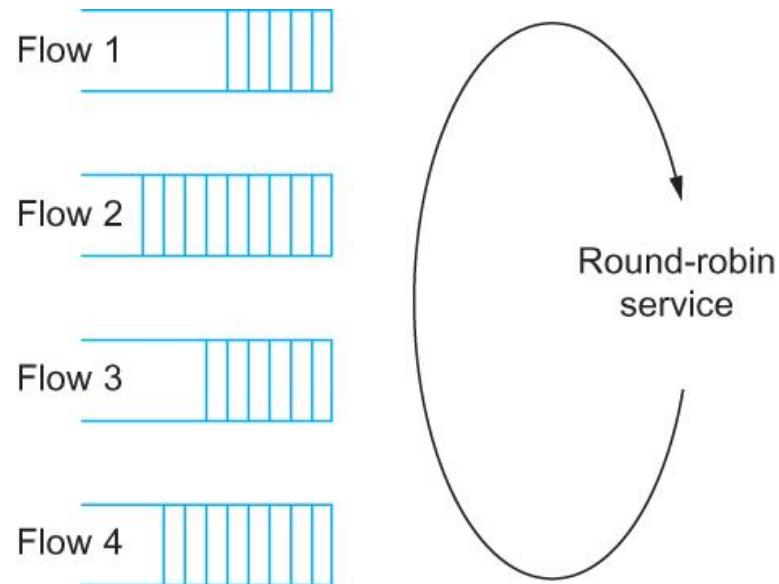
26

- Fair Queuing
 - ▣ The main problem with FIFO queuing is that
 - It does not discriminate between different traffic sources, or
 - It does not separate packets according to the flow to which they belong.
 - ▣ Fair queuing (FQ) is an algorithm that has been proposed to address this problem.
 - The idea of FQ is to maintain a separate queue for each flow currently being handled by the router.
 - The router then services these queues in a sort of round-robin

Queuing Disciplines

27

□ Fair Queuing



Round-robin service of four flows at a router

Queuing Disciplines

28

- Fair Queuing
 - ▣ The main complication with Fair Queuing is that the packets being processed at a router are not necessarily the same length.
 - ▣ To truly allocate the bandwidth of the outgoing link in a fair manner, it is necessary to take packet length into consideration.
 - For example, if a router is managing two flows, one with 1000-byte packets and the other with 500-byte packets (perhaps because of fragmentation upstream from this router), then a simple round-robin servicing of packets from each flow's queue will give the first flow two thirds of the link's bandwidth and the second flow only one-third of its bandwidth.

Queuing Disciplines

29

- Fair Queuing
 - What we really want is bit-by-bit round-robin; that is, the router transmits a bit from flow 1, then a bit from flow 2, and so on.
 - Clearly, it is not feasible to interleave the bits from different packets.
 - The FQ mechanism therefore simulates this behavior by first determining when a given packet would finish being transmitted if it were being sent using bit-by-bit round-robin, and then using this finishing time to sequence the packets for transmission

Queuing Disciplines

30

□ Fair Queuing

- To understand the algorithm for approximating bit-by-bit round robin, consider the behavior of a single flow
- For this flow, let

- P_i : denote the length of packet i
- S_i : time when the router starts to transmit packet i
- F_i : time when router finishes transmitting packet i
- Clearly, $F_i = S_i + P_i$

Queuing Disciplines

31

□ Fair Queuing

- When do we start transmitting packet i ?
 - Depends on whether packet i arrived before or after the router finishes transmitting packet $i-1$ for the flow
- Let A_i denote the time that packet i arrives at the router
- Then $S_i = \max(F_{i-1}, A_i)$
- $F_i = \max(F_{i-1}, A_i) + P_i$

Queuing Disciplines

32

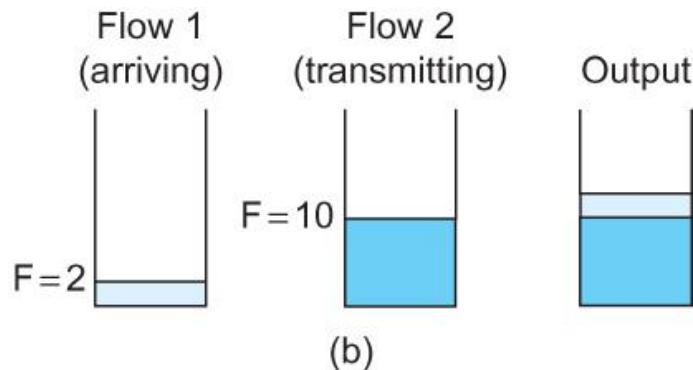
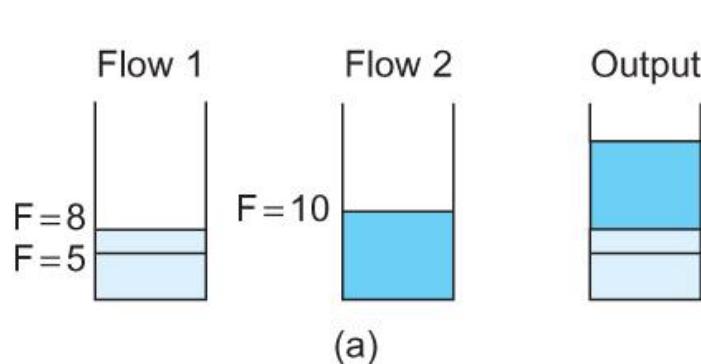
□ Fair Queuing

- Now for every flow, we calculate F_i for each packet that arrives using our formula
- We then treat all the F_i as timestamps
- Next packet to transmit is always the packet that has the lowest timestamp
 - The packet that should finish transmission before all others

Queuing Disciplines

33

□ Fair Queuing



Example of fair queuing in action: (a) packets with earlier finishing times are sent first; (b) sending of a packet already in progress is completed

TCP Congestion Control

34

- TCP congestion control was introduced into the Internet in the late 1980s by Van Jacobson, roughly eight years after the TCP/IP protocol stack had become operational.
- Immediately preceding this time, the Internet was suffering from congestion collapse—
 - Hosts would send their packets into the Internet as fast as the advertised window would allow,
 - Congestion would occur at some router (causing packets to be dropped), and
 - The hosts would time out and retransmit their packets, resulting in even more congestion

TCP Congestion Control

35

- The idea of TCP congestion control is
 - ▣ for each source to determine how much capacity is available in the network,
 - ▣ So that it knows how many packets it can safely have in transit.
 - Once a given source has this many packets in transit,
 - it uses the arrival of an ACK as a signal that one of its packets has left the network, and
 - it is therefore safe to insert a new packet into the network without adding to the level of congestion.
 - By using ACKs to pace the transmission of packets, TCP is said to be *self-clocking*.

TCP Congestion Control

36

- Additive Increase Multiplicative Decrease
 - TCP maintains a new state variable for each connection, called *CongestionWindow*,
 - which is used by the source to limit how much data it is allowed to have in transit at a given time.
 - The congestion window is congestion control's counterpart to flow control's advertised window.
 - TCP is modified such that the maximum number of bytes of unacknowledged data allowed is now the minimum of the congestion window and the advertised window

TCP Congestion Control

37

- Additive Increase Multiplicative Decrease
 - ▣ TCP's effective window is revised as follows:
 - $\text{MaxWindow} = \text{MIN}(\text{CongestionWindow}, \text{AdvertisedWindow})$
 - $\text{EffectiveWindow} = \text{MaxWindow} - (\text{LastByteSent} - \text{LastByteAcked})$.
 - ▣ That is, MaxWindow replaces AdvertisedWindow in the calculation of EffectiveWindow.
 - ▣ Thus, a TCP source is allowed to send no faster than the slowest component—the network or the destination host—can accommodate.

TCP Congestion Control

38

□ Additive Increase Multiplicative Decrease

- The problem, is how TCP comes to learn an appropriate value for CongestionWindow.
- Unlike the AdvertisedWindow, which is sent by the receiving side of the connection, there is no one to send a suitable CongestionWindow to the sending side of TCP.
 - The TCP source sets the CongestionWindow based on the **level of congestion it perceives to exist in the network**.
 - This involves decreasing the congestion window when the level of congestion goes up and increasing the congestion window when the level of congestion goes down. Taken together, the mechanism is commonly called *additive increase/multiplicative decrease (AIMD)*

TCP Congestion Control

39

- Additive Increase Multiplicative Decrease
 - The key question, then, is how does the source determine that the network is congested and that it should decrease the congestion window?
 - Based on the observation
 - that the main reason packets are not delivered, and a timeout results, is that a packet was dropped due to congestion.
 - It is rare that a packet is dropped because of an error during transmission.
 - Therefore, TCP interprets **timeouts as a sign of congestion** and reduces the rate at which it is transmitting.
 - Specifically, each time a timeout occurs, the source sets CongestionWindow to half of its previous value.
 - This halving of the CongestionWindow for each timeout corresponds to the “multiplicative decrease” part of AIMD.

TCP Congestion Control

40

- Additive Increase Multiplicative Decrease
 - Although CongestionWindow is defined in terms of bytes, it is easiest to understand multiplicative decrease if we think in terms of whole packets.
 - For example, suppose the CongestionWindow is currently set to 16 packets. If a loss is detected, CongestionWindow is set to 8.
 - Additional losses cause CongestionWindow to be reduced to 4, then 2, and finally to 1 packet.
 - CongestionWindow is not allowed to fall below the size of a single packet, or in TCP terminology, the *maximum segment size (MSS)*.

TCP Congestion Control

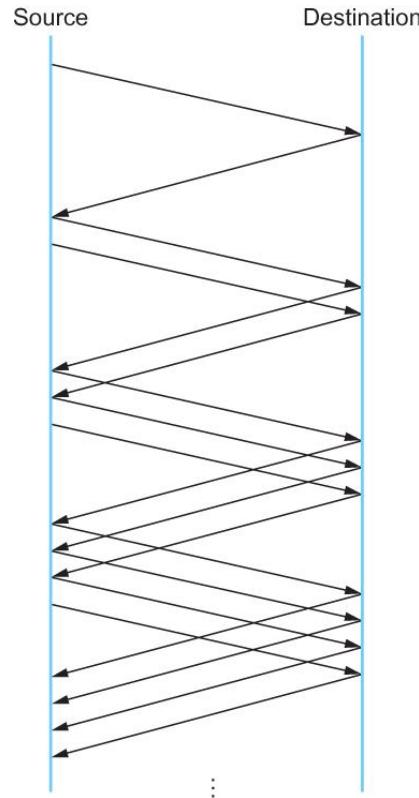
41

- Additive Increase Multiplicative Decrease
 - A congestion-control strategy that only decreases the window size is obviously too conservative.
 - We also need to be able to **increase the congestion window** to take advantage of newly available capacity in the network.
 - This is the “additive increase” part of AIMD, and it works as follows.
 - Every time the source successfully sends a CongestionWindow’s worth of packets—that is, each packet sent out during the last RTT has been ACKed—it **adds the equivalent of 1 packet** to CongestionWindow.

TCP Congestion Control

42

❑ Additive Increase Multiplicative Decrease



Packets in transit during
additive increase, with one
packet being added each RTT

TCP Congestion Control

43

- Additive Increase Multiplicative Decrease
 - Note that in practice, TCP does not wait for an entire window's worth of ACKs to add 1 packet's worth to the congestion window, but instead increments CongestionWindow by a little for each ACK that arrives.
 - Specifically, the congestion window is incremented as follows each time an ACK arrives:
 - Increment = $MSS \times (MSS/CongestionWindow)$
 - CongestionWindow+= Increment
 - That is, rather than incrementing CongestionWindow by an entire MSS bytes each RTT, we increment it by a fraction of MSS every time an ACK is received.
 - Assuming that each ACK acknowledges the receipt of MSS bytes, then that fraction is $MSS/CongestionWindow$.

TCP Congestion Control

44

□ Slow Start

- The additive increase mechanism just described is the right approach to use when the **source is operating close to the available capacity of the network**, but it takes too long to ramp up a connection when it is starting **from scratch**.
- TCP therefore provides a second mechanism, ironically called *slow start*, that is used to increase the congestion window rapidly from a cold start.
- Slow start effectively increases the congestion window exponentially, rather than linearly

TCP Congestion Control

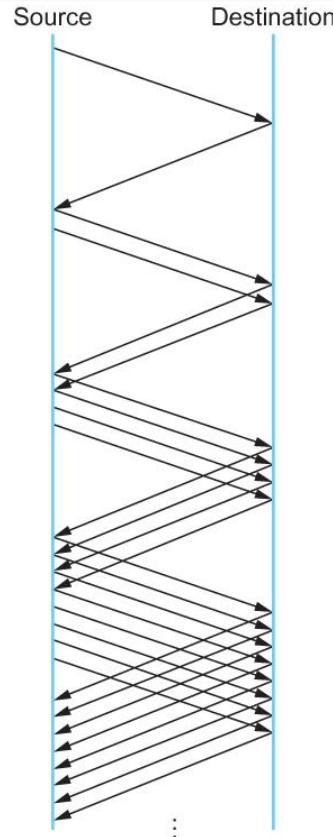
45

- Slow Start
 - Specifically, the source starts out by setting CongestionWindow to one packet.
 - When the ACK for this packet arrives, TCP adds 1 to CongestionWindow and then sends two packets.
 - Upon receiving the corresponding two ACKs, TCP increments CongestionWindow by 2—one for each ACK—and next sends four packets.
 - The end result is that TCP effectively doubles the number of packets it has in transit every RTT.

TCP Congestion Control

46

□ Slow Start



Packets in transit during slow start.

TCP Congestion Control

47

- Slow Start
 - There are actually two different situations in which slow start runs.
 - The first is at the very beginning of a connection, at which time the source has no idea how many packets it is going to be able to have in transit at a given time.
 - In this situation, slow start continues to double CongestionWindow each RTT until there is a loss, at which time a timeout causes multiplicative decrease to divide CongestionWindow by 2.

TCP Congestion Control

48

- Slow Start
 - There are actually two different situations in which slow start runs.
 - The second situation in which slow start is used is a bit more subtle; it occurs when the connection goes dead while waiting for a timeout to occur.
 - Recall how TCP's sliding window algorithm works—when a packet is lost, the source eventually reaches a point where it has sent as much data as the advertised window allows, and so it blocks while waiting for an ACK that will not arrive.
 - Eventually, a timeout happens, but by this time there are no packets in transit, meaning that the source will receive no ACKs to “clock” the transmission of new packets.
 - The source will instead receive a single cumulative ACK that reopens the entire advertised window, but as explained above, **the source then uses slow start to restart the flow** of data rather than dumping a whole window's worth of data on the network all at once.

TCP Congestion Control

49

- Slow Start
 - Although the source is using slow start again, it now knows more information than it did at the beginning of a connection.
 - Specifically, the source has a current (and useful) value of CongestionWindow; this is the value of CongestionWindow that existed prior to the last packet loss, divided by 2 as a result of the loss.
 - We can think of this as the “target” congestion window.
 - Slow start is used to rapidly increase the sending rate up to this value, and then additive increase is used beyond this point.

TCP Congestion Control

50

- Slow Start
 - ▣ Notice that we have a small bookkeeping problem to take care of, in that we want to remember the “target” congestion window resulting from multiplicative decrease as well as the “actual” congestion window being used by slow start.
 - ▣ To address this problem, TCP introduces a temporary variable to store the target window, typically called **CongestionThreshold**, that is set equal to the **CongestionWindow** value that results from multiplicative decrease.

TCP Congestion Control

51

- Slow Start
 - The variable CongestionWindow is then reset to one packet, and it is incremented by one packet for every ACK that is received until it reaches.
 - **CongestionThreshold**, at which point it is incremented by one packet per RTT.

TCP Congestion Control

52

□ Slow Start

- In other words, TCP increases the congestion window as defined by the following code fragment:

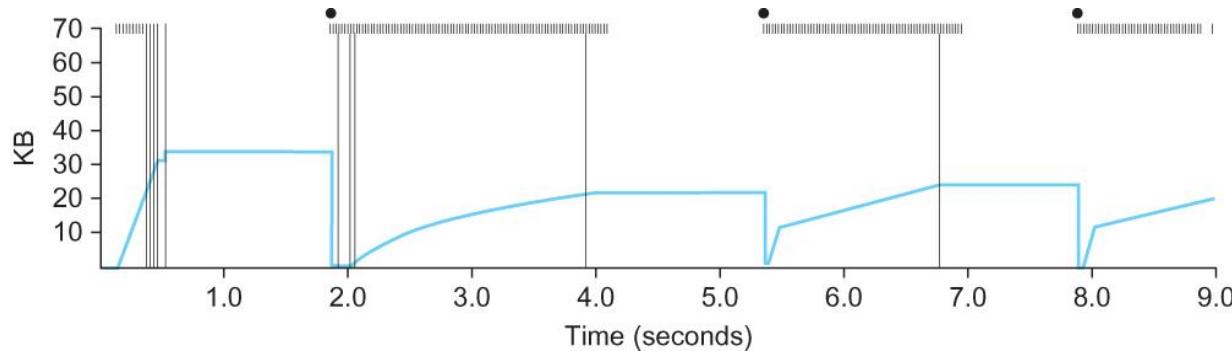
```
{  
    u_int cw = state->CongestionWindow;  
    u_int incr = state->maxseg;  
    if (cw > state->CongestionThreshold)  
        incr = incr * incr / cw;  
    state->CongestionWindow = MIN(cw + incr,  
TCP_MAXWIN);  
}
```

- where state represents the state of a particular TCP connection and TCP MAXWIN defines an upper bound on how large the congestion window is allowed to grow

TCP Congestion Control

53

□ Slow Start



Behavior of TCP congestion control. Colored line = value of CongestionWindow over time;
solid bullets at top of graph = timeouts;
hash marks at top of graph = time when each packet is transmitted;
vertical bars = time when a packet that was eventually retransmitted was first transmitted.

TCP Congestion Control

54

- Fast Retransmit and Fast Recovery
 - ▣ The mechanisms described so far were part of the original proposal to add congestion control to TCP.
 - ▣ It was soon discovered, however, that the coarse-grained implementation of TCP timeouts led to long periods of time during which the connection went dead while waiting for a timer to expire.
 - ▣ Because of this, a new mechanism called *fast retransmit* was added to TCP.
 - ▣ **Fast retransmit is a heuristic** that sometimes triggers the retransmission of a dropped packet sooner than the regular timeout mechanism

TCP Congestion Control

55

- Fast Retransmit and Fast Recovery
 - ▣ The idea of fast retransmit is straightforward. Every time a data packet arrives at the receiving side, the receiver responds with an acknowledgment, even if this sequence number has already been acknowledged.
 - ▣ Thus, when a packet arrives out of order— that is, TCP cannot yet acknowledge the data the packet contains because earlier data has not yet arrived—TCP resends the same acknowledgment it sent the last time.

TCP Congestion Control

56

- Fast Retransmit and Fast Recovery
 - This second transmission of the same acknowledgment is called a *duplicate ACK*.
 - When the sending side sees a duplicate ACK, it knows that the other side must have received a packet out of order, which suggests that an earlier packet might have been lost.
 - Since it is also possible that the earlier packet has only been delayed rather than lost, the sender waits until it sees some number of duplicate ACKs and then retransmits the missing packet. In practice, TCP waits until it has seen three duplicate ACKs before retransmitting the packet.

TCP Congestion Control

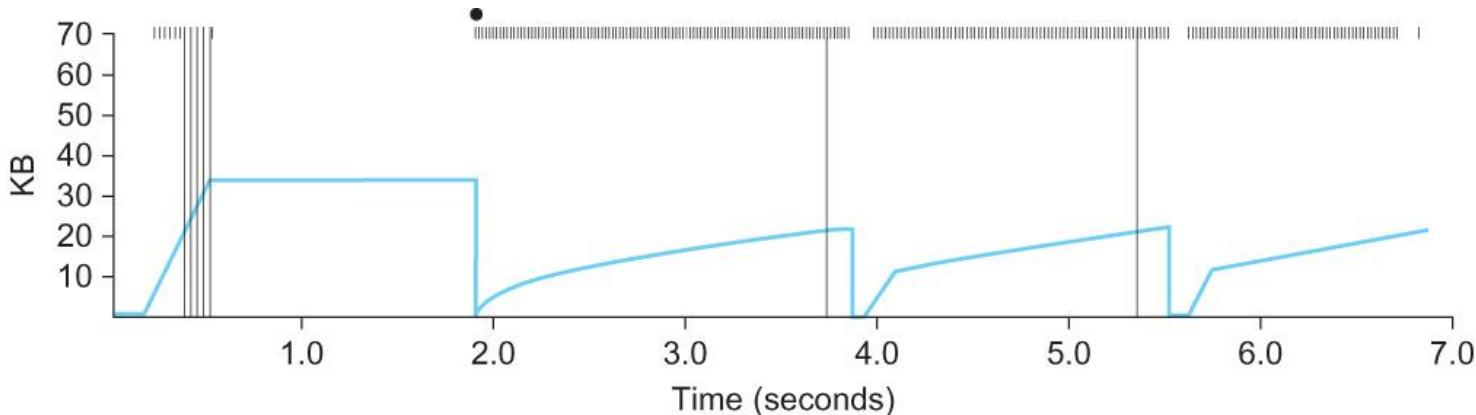
57

- Fast Retransmit and Fast Recovery
 - When the fast retransmit mechanism signals congestion, rather than drop the congestion window all the way back to one packet and run slow start, it is possible to use the ACKs that are still in the pipe to clock the sending of packets.
 - This mechanism, which is called *fast recovery*, effectively removes the slow start phase that happens between when fast retransmit detects a lost packet and additive increase begins.

TCP Congestion Control

58

□ Fast Retransmit and Fast Recovery



Trace of TCP with fast retransmit. Colored line = CongestionWindow;
solid bullet = timeout;
hash marks = time when each packet is transmitted;
vertical bars = time when a packet that was eventually retransmitted was first transmitted.

Congestion Avoidance Mechanism

59

- TCP's strategy is to control congestion once it happens
 - ▣ Opposite – to trying to avoid congestion in the first place.
- In fact, TCP repeatedly increases the load it imposes on the network in an effort to find the point at which congestion occurs, and then it backs off from this point.
- An appealing alternative – is to predict when congestion is about to happen and
 - ▣ Then to reduce the rate at which hosts send data just before packets start being discarded.
- We call such a strategy *congestion avoidance*, to distinguish it from *congestion control*

Congestion Avoidance Mechanism

60

- DEC Bit
 - The first mechanism was developed for use on the Digital Network Architecture (DNA), a connectionless network with a connection-oriented transport protocol.
 - The idea here is to more evenly split the responsibility for congestion control between the routers and the end nodes

Congestion Avoidance Mechanism

61

- DEC Bit
 - Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur.
 - This notification is implemented by **setting a binary congestion bit** in the packets that flow through the router; hence the name DECbit.
 - The destination host then copies this congestion bit into the ACK it sends back to the source.
 - Finally, the source adjusts its sending rate so as to avoid congestion

Congestion Avoidance Mechanism

62

- DEC Bit
 - A single congestion bit is added to the packet header.
 - A router sets this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives.
 - This average queue length is measured over a time interval that spans the last **busy + idle** cycle, plus the current busy cycle

Congestion Avoidance Mechanism

63

- DEC Bit
 - Essentially, the router calculates the area under the curve and divides this value by the time interval to compute the average queue length.
 - Using a queue length of 1 as the trigger for setting the congestion bit is a trade-off between significant queuing (and hence higher throughput) and increased idle time (and hence lower delay).
 - In other words, a queue length of 1 seems to optimize the power function

Congestion Avoidance Mechanism

64

- DEC Bit
 - The source records how many of its packets resulted in some router setting the congestion bit.
 - In particular, the source maintains a congestion window, just as in TCP, and watches to see what fraction of the last window's worth of packets resulted in the bit being set

Congestion Avoidance Mechanism

65

- DEC Bit
 - If less than 50% of the packets had the bit set, then the source increases its congestion window by one packet.
 - If 50% or more of the last window's worth of packets had the congestion bit set, then the source decreases its congestion window to 0.875 times the previous value.

Congestion Avoidance Mechanism

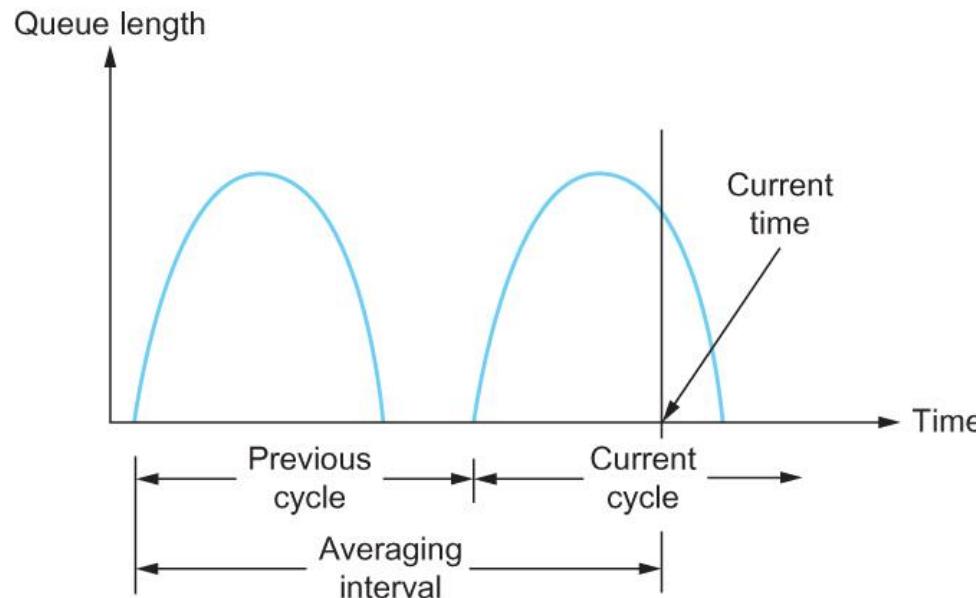
66

- DEC Bit
 - The value 50% was chosen as the threshold based on analysis that showed it to correspond to the peak of the power curve. The “increase by 1, decrease by 0.875” rule was selected because additive increase/multiplicative decrease makes the mechanism stable

Congestion Avoidance Mechanism

67

□ DEC Bit



Computing average queue length at a router

Congestion Avoidance Mechanism

68

- Random Early Detection (RED)
 - A second mechanism, called *random early detection (RED)*, is similar to the DECbit scheme
 - In that each router is programmed to monitor its own queue length, and when it detects that congestion is imminent, to notify the source to adjust its congestion window.
 - RED, invented by Sally Floyd and Van Jacobson in the early 1990s, differs from the DECbit scheme in two major ways:

Congestion Avoidance Mechanism

69

- Random Early Detection (RED)
 - The first is that rather than explicitly sending a congestion notification message to the source, RED is most commonly implemented such that it *implicitly notifies* the source of congestion by dropping one of its packets.
 - The source is, therefore, effectively notified by the subsequent timeout or duplicate ACK.
 - RED is designed to be used in conjunction with TCP, which currently detects congestion by means of timeouts (or some other means of detecting packet loss such as duplicate ACKs).

Congestion Avoidance Mechanism

70

- Random Early Detection (RED)
 - As the “early” part of the RED acronym suggests, the gateway drops the packet earlier than it would have to, so as to notify the source that it should decrease its congestion window sooner than it would normally have.
 - In other words, the router drops a few packets before it has exhausted its buffer space completely, so as to cause the source to slow down, with the hope that this will mean it does not have to drop lots of packets later on.

Congestion Avoidance Mechanism

71

- Random Early Detection (RED)
 - The second difference between RED and DECbit is in the details of how RED decides when to drop a packet and what packet it decides to drop.
 - To understand the basic idea, consider a simple FIFO queue. Rather than wait for the queue to become completely full and then be forced to drop each arriving packet, we could decide to drop each arriving packet with some *drop probability* whenever the queue length exceeds some *drop level*.
 - This idea is *called early random drop. The RED algorithm defines the details of how to monitor the queue length and when to drop a packet*

Congestion Avoidance Mechanism

72

- Random Early Detection (RED)
 - ▣ First, RED computes an average queue length using a weighted running average similar to the one used in the original TCP timeout computation. That is, AvgLen is computed as
 - $\text{AvgLen} = (1 - \text{Weight}) \times \text{AvgLen} + \text{Weight} \times \text{SampleLen}$
 - where $0 < \text{Weight} < 1$ and SampleLen is the length of the queue when a sample measurement is made.
 - ▣ In most software implementations, the queue length is measured every time a new packet arrives at the gateway.
 - ▣ In hardware, it might be calculated at some fixed sampling interval.

Congestion Avoidance Mechanism

73

□ Random Early Detection (RED)

- Second, RED has two queue length thresholds that trigger certain activity: MinThreshold and MaxThreshold.
- When a packet arrives at the gateway, RED compares the current AvgLen with these two thresholds, according to the following rules:
 - if $\text{AvgLen} \leq \text{MinThreshold}$
 - \rightarrow queue the packet
 - if $\text{MinThreshold} < \text{AvgLen} < \text{MaxThreshold}$
 - \rightarrow calculate probability P
 - \rightarrow drop the arriving packet with probability P
 - if $\text{MaxThreshold} \leq \text{AvgLen}$
 - \rightarrow drop the arriving packet

Congestion Avoidance Mechanism

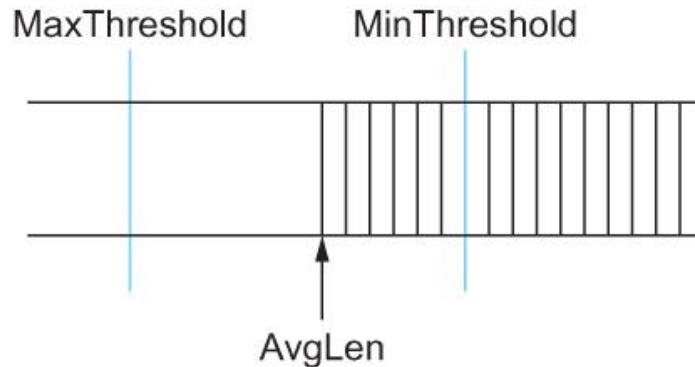
74

- Random Early Detection (RED)
 - ▣ P is a function of both AvgLen and how long it has been since the last packet was dropped.
 - ▣ Specifically, it is computed as follows:
 - $\text{TempP} = \text{MaxP} \times (\text{AvgLen} - \text{MinThreshold}) / (\text{MaxThreshold} - \text{MinThreshold})$
 - $P = \text{TempP} / (1 - \text{count} \times \text{TempP})$

Congestion Avoidance Mechanism

75

□ Random Early Detection (RED)

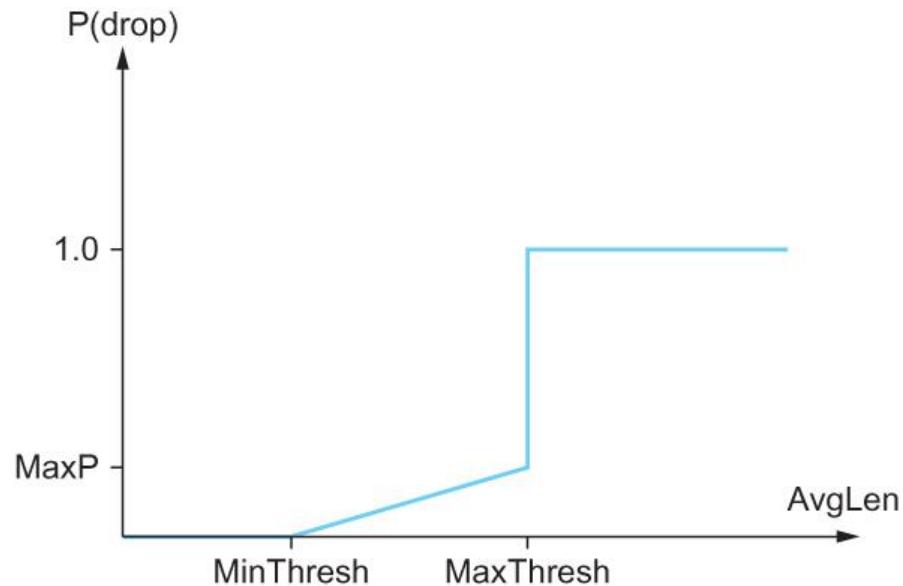


RED thresholds on a FIFO queue

Congestion Avoidance Mechanism

76

□ Random Early Detection (RED)



Drop probability function for RED

Congestion Avoidance Mechanism

77

- Source-based Congestion Avoidance
 - Watch for some sign from the network that some router's queue is building up and that congestion will happen soon if nothing is done about it.
 - For example, the source might notice that as packet queues build up in the network's routers, there is a measurable increase in the RTT for each successive packet it sends

Congestion Avoidance Mechanism

78

- Source-based Congestion Avoidance
 - One particular algorithm exploits this observation as follows:
 - The congestion window normally increases as in TCP, but every two round-trip delays the algorithm checks to see if the current RTT is greater than the average of the minimum and maximum RTTs seen so far.
 - If it is, then the algorithm decreases the congestion window by one-eighth.

Congestion Avoidance Mechanism

79

- Source-based Congestion Avoidance
 - A second algorithm does something similar.
 - The decision as to whether or not to change the current window size is based on changes to both the RTT and the window size.
 - The window is adjusted once every two round-trip delays based on the product
 - $(\text{CurrentWindow} - \text{OldWindow}) \times (\text{CurrentRTT} - \text{OldRTT})$
 - If the result is positive, the source decreases the window size by one-eighth;
 - if the result is negative or 0, the source increases the window by one maximum packet size.
 - Note that the window changes during every adjustment; that is, it oscillates around its optimal point.

Quality of Service

80

- For many years, packet-switched networks have offered the promise of supporting multimedia applications,
 - i.e., audio, video, and data.
- After all, once digitized, audio and video information become like any other form of data
 - A stream of bits to be transmitted.
 - One obstacle to the fulfillment of this promise has been the need for higher-bandwidth links.
- Recently,
 - The improvements in coding have reduced the bandwidth needs of audio and video applications
 - While at the same time link speeds have increased.

Quality of Service

81

- There is more to transmitting audio and video over a network than just providing sufficient bandwidth
 - Synchronization
- Participants in a telephone conversation, for example, expect to be able to converse in such a way that one person can respond to something said by the other and be heard almost immediately.
- Thus, the **timeliness** of delivery can be very important.
- When you refer to applications that are sensitive to the timeliness of data as *real-time applications*

Quality of Service

82

- Industrial control—you would like a command sent to a robot arm to reach it before the arm crashes into something.
- Even file transfer applications can have timeliness constraints
 - Such as a requirement that a database update complete overnight before the business that needs the data resumes on the next day

Quality of Service

83

- The distinguishing characteristic of real-time applications is that they need some sort of assurance *from the network that data is likely to arrive on time.*
- Whereas a non-real-time application can use an end-to-end retransmission strategy to make sure that data arrives *correctly, such a strategy cannot provide timeliness.*
- This implies that the network will treat some packets differently from others—something that is not done in the best-effort model.
- A network that can provide these different levels of service is often said to support quality of service (QoS).

Quality of Service

84

□ Real-Time Applications

- Data is generated by collecting samples from a microphone and digitizing them using an A →D converter
- The digital samples are placed in packets which are transmitted across the network and received at the other end
- At the receiving host the data must be played back at some appropriate rate

Quality of Service

85

□ Real-Time Applications

- For example, if voice samples were collected at a rate of one per $125 \mu\text{s}$, they should be played back at the same rate
- We can think of each sample as having a particular playback time
- The point in time at which it is needed at the receiving host
- In this example, each sample has a playback time that is $125 \mu\text{s}$ later than the preceding sample
- If data arrives after its appropriate playback time, it is useless

Quality of Service

86

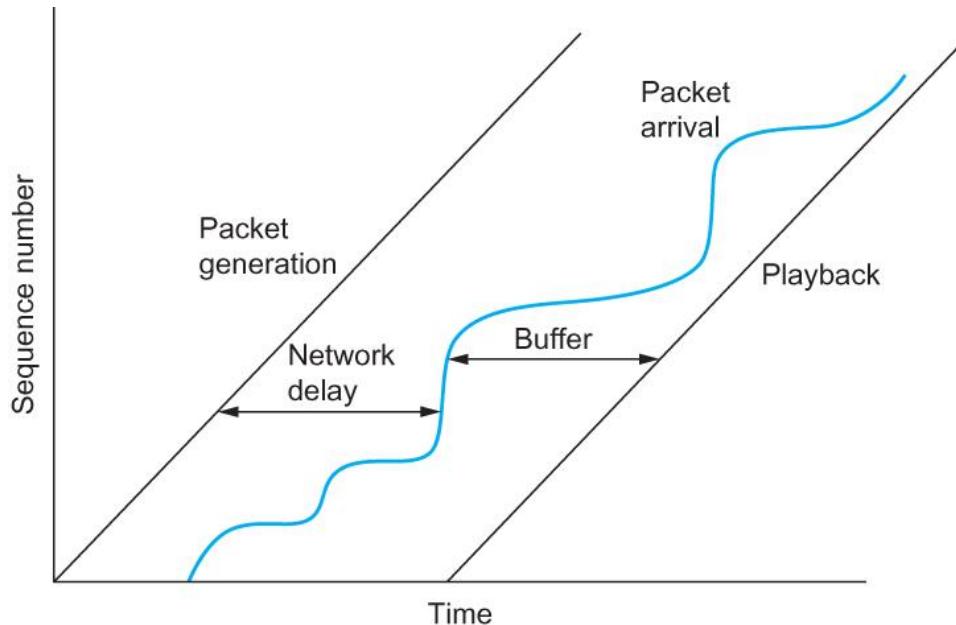
□ Real-Time Applications

- For some audio applications, there are limits to how far we can delay playing back data
- It is hard to carry on a conversation if the time between when you speak and when your listener hears you is more than 300 ms
- We want from the network a guarantee that all our data will arrive within 300 ms
- If data arrives early, we buffer it until playback time

Quality of Service

87

□ Real-Time Applications



A playback buffer

Quality of Service

88

- Taxonomy of Real-Time Applications
 - ▣ The first characteristic by which we can categorize applications is their tolerance of loss of data
 - Where “loss” might occur because a packet arrived too late to be played back as well as arising from the usual causes in the network.
 - ▣ On the one hand, one lost audio sample can be interpolated from the surrounding samples with relatively little effect on the perceived audio quality.
 - It is only as more and more samples are lost that quality declines to the point that the speech becomes incomprehensible.

Quality of Service

89

- Taxonomy of Real-Time Applications
 - On the other hand, a robot control program is likely to be an example of a real-time application that cannot tolerate loss—losing the packet that contains the command instructing the robot arm to stop is unacceptable.
 - Thus, we can categorize real-time applications as *tolerant* or *intolerant* depending on whether they can tolerate occasional loss

Quality of Service

90

- Taxonomy of Real-Time Applications
 - A second way to characterize real-time applications is by their adaptability.
 - For example, an audio application might be able to adapt to the amount of delay that packets experience as they traverse the network.
 - If we notice that packets are almost always arriving within 300 ms of being sent, then we can set our playback point accordingly, buffering any packets that arrive in less than 300 ms.
 - Suppose that we subsequently observe that all packets are arriving within 100 ms of being sent.
 - If we moved up our playback point to 100 ms, then the users of the application would probably perceive an improvement.
 - The process of shifting the playback point would actually require us to play out samples at an increased rate for some period of time.

Quality of Service

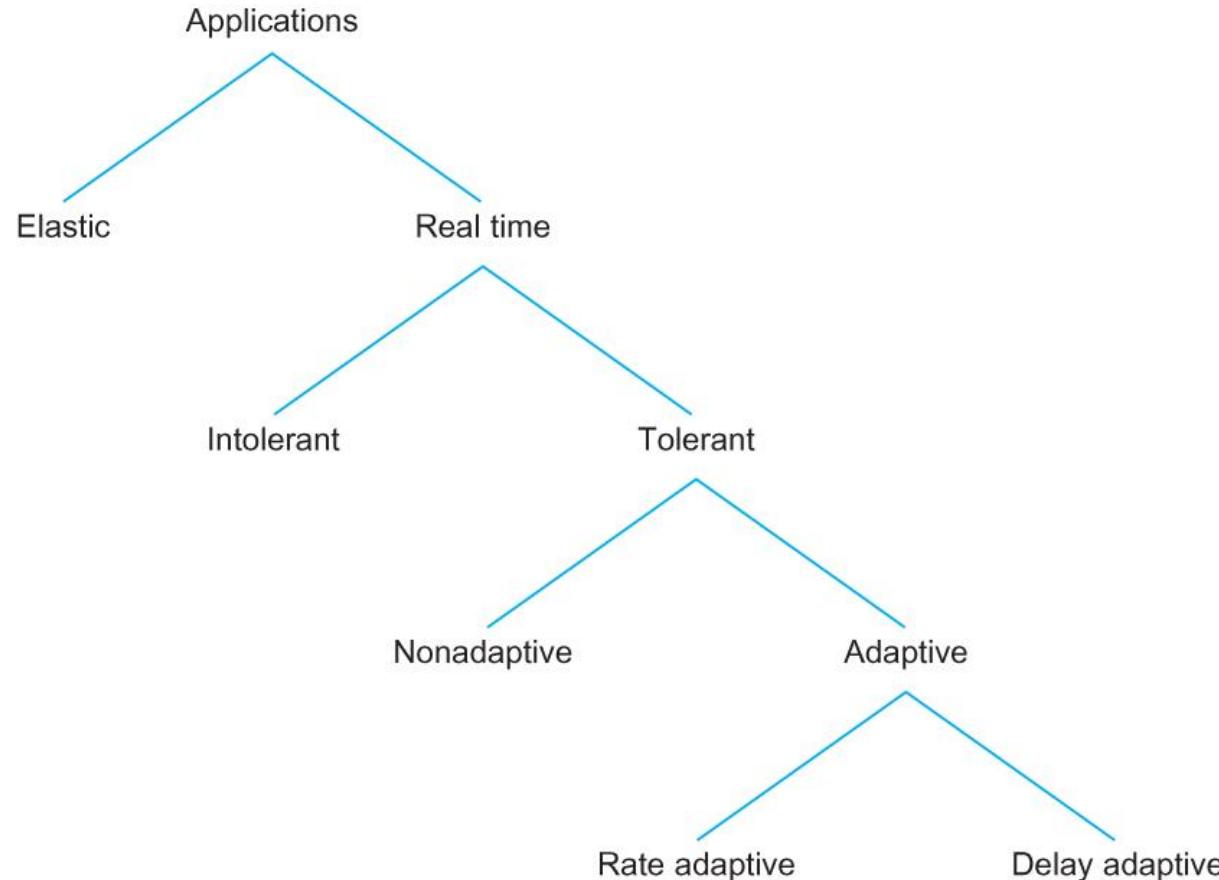
91

- Taxonomy of Real-Time Applications
 - *Delay-adaptive applications.*
 - The applications that can adjust their playback point
 - Another class of adaptive applications are *rate adaptive*.
 - *For example, many* video coding algorithms can trade off bit rate versus quality.
 - Thus, if we find that the network can support a certain bandwidth, we can set our coding parameters accordingly.
 - If more bandwidth becomes available later, we can change parameters to increase the quality.

Quality of Service

92

□ Taxonomy of Real-Time Applications



Quality of Service

93

- Approaches to QoS Support
 - *fine-grained approaches*
 - Which provide QoS to individual applications or flows
 - *coarse-grained approaches*
 - Which provide QoS to large classes of data or aggregated traffic
 - In the first category we find “Integrated Services,” a QoS architecture developed in the IETF and often associated with RSVP (Resource Reservation Protocol).
 - In the second category lies “Differentiated Services,” which is probably the most widely deployed QoS mechanism.

Quality of Service

94

- Integrated Services (RSVP)
 - The term “Integrated Services” (often called IntServ for short) refers to a body of work that was produced by the IETF around 1995–97.
 - The IntServ working group developed specifications of a number of *service classes designed to meet the needs of some of the application types described above.*
 - It also defined how RSVP could be used to make reservations using these service classes

Quality of Service

95

□ Integrated Services (RSVP)

□ Service Classes

- Guaranteed Service
 - The network should guarantee that the maximum delay that any packet will experience has some specified value
- Controlled Load Service
 - The aim of the controlled load service is to emulate a lightly loaded network for those applications that request the service, even though the network as a whole may in fact be heavily loaded

Quality of Service

96

□ Integrated Services (RSVP)

□ Overview of Mechanisms

■ Flowspec

- With a best-effort service we can just tell the network where we want our packets to go and leave it at that, a real-time service involves telling the network something more about the type of service we require
- The set of information that we provide to the network is referred to as a *flowspec*.

■ Admission Control

- When we ask the network to provide us with a particular service, the network needs to decide if it can in fact provide that service.
- The process of deciding when to say no is called *admission control*.

■ Resource Reservation

- We need a mechanism by which the users of the network and the components of the network itself exchange information such as requests for service, flowspecs, and admission control decisions.
- We refer to this process as *resource reservation*

Quality of Service

97

□ Integrated Services (RSVP)

□ Overview of Mechanisms

■ Packet Scheduling

- Finally, when flows and their requirements have been described, and admission control decisions have been made, the network switches and routers need to meet the requirements of the flows.
- A key part of meeting these requirements is managing the way packets are queued and scheduled for transmission in the switches and routers.
- This last mechanism is *packet scheduling*.

Quality of Service

98

□ Integrated Services (RSVP)

□ Flowspec

- There are two separable parts to the flowspec:
 - The part that describes the flow's traffic characteristics (called the *TSpec*) and
 - The part that describes the service requested from the network (the *RSpec*).

- The RSpec is very service specific and relatively easy to describe.
- For example, with a controlled load service, the RSpec is trivial: The application just requests controlled load service with no additional parameters.
- With a guaranteed service, you could specify a delay target or bound.

Quality of Service

99

□ Integrated Services (RSVP)

□ Flowspec

■ Tspec

- We need to give the network enough information about the bandwidth used by the flow to allow intelligent admission control decisions to be made
- For most applications, the bandwidth is not a single number
 - It varies constantly
- A video application will generate more bits per second when the scene is changing rapidly than when it is still
 - Just knowing the long term average bandwidth is not enough

Quality of Service

100

□ Integrated Services (RSVP)

□ Flowspec

- Suppose 10 flows arrive at a switch on separate ports and they all leave on the same 10 Mbps link
- If each flow is expected to send no more than 1 Mbps
 - No problem
- If these are variable bit applications such as compressed video
 - They will occasionally send more than the average rate
- If enough sources send more than average rates, then the total rate at which data arrives at the switch will be more than 10 Mbps
- This excess data will be queued
- The longer the condition persists, the longer the queue will get

Quality of Service

101

□ Integrated Services (RSVP)

□ Flowspec

- One way to describe the Bandwidth characteristics of sources is called a Token Bucket Filter
- The filter is described by two parameters
 - A token rate r
 - A bucket depth B
- To be able to send a byte, a token is needed
- To send a packet of length n , n tokens are needed
- Initially there are no tokens
- Tokens are accumulated at a rate of r per second
- No more than B tokens can be accumulated

Quality of Service

102

- Integrated Services (RSVP)

- Flowspec

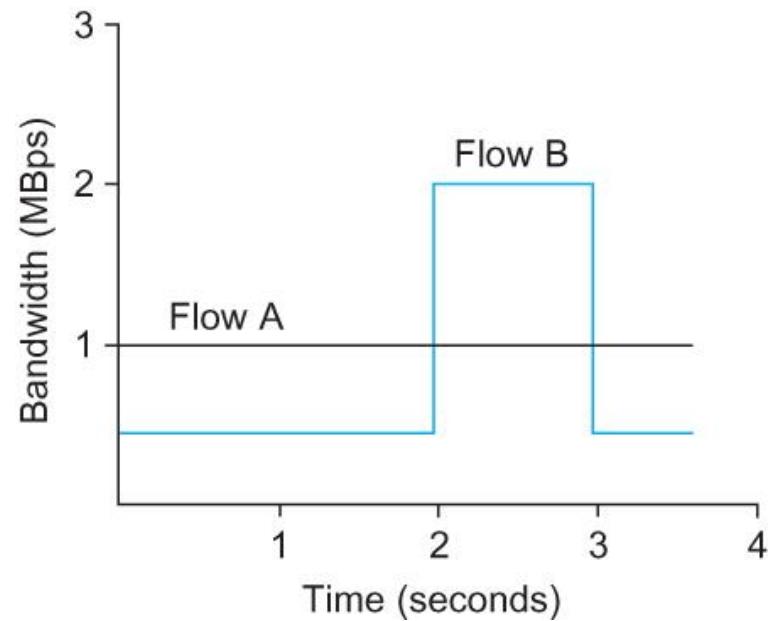
- We can send a burst of as many as B bytes into the network as fast as we want, but over significant long interval we cannot send more than r bytes per second
 - This information is important for admission control algorithm when it tries to find out whether it can accommodate new request for service

Quality of Service

103

□ Flowspec

- The figure illustrates how a token bucket can be used to characterize a flow's Bandwidth requirement
- For simplicity, we assume each flow can send data as individual bytes rather than as packets
- Flow A generates data at a steady rate of 1 MBps
 - So it can be described by a token bucket filter with a rate $r = 1$ MBps and a bucket depth of 1 byte
 - This means that it receives tokens at a rate of 1 MBps but it cannot store more than 1 token, it spends them immediately

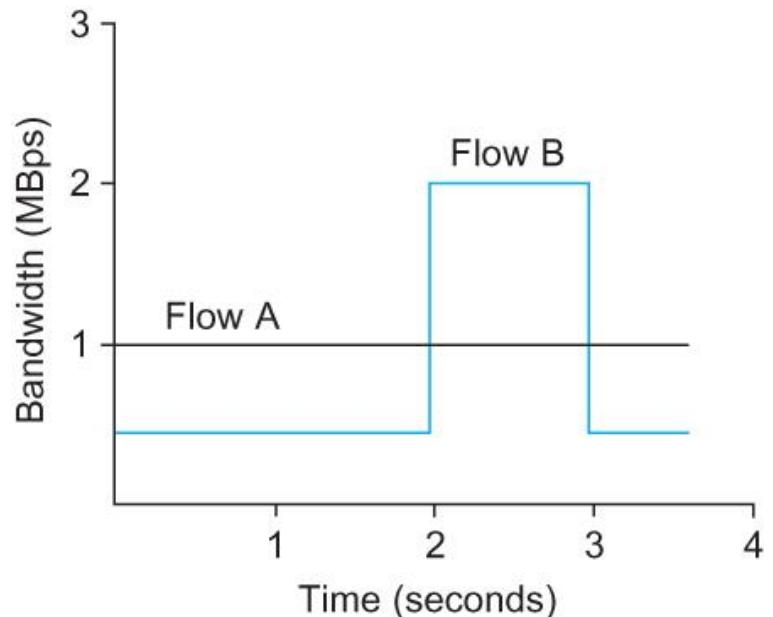


Quality of Service

104

□ Flowspec

- Flow B sends at a rate that averages out to 1 MBps over the long term, but does so by sending at 0.5 MBps for 2 seconds and then at 2 MBps for 1 second
- Since the token bucket rate r is a long term average rate, flow B can be described by a token bucket with a rate of 1 MBps
- Unlike flow A, however flow B needs a bucket depth B of at least 1 MB, so that it can store up tokens while it sends at less than 1 MBps to be used when it sends at 2 MBps

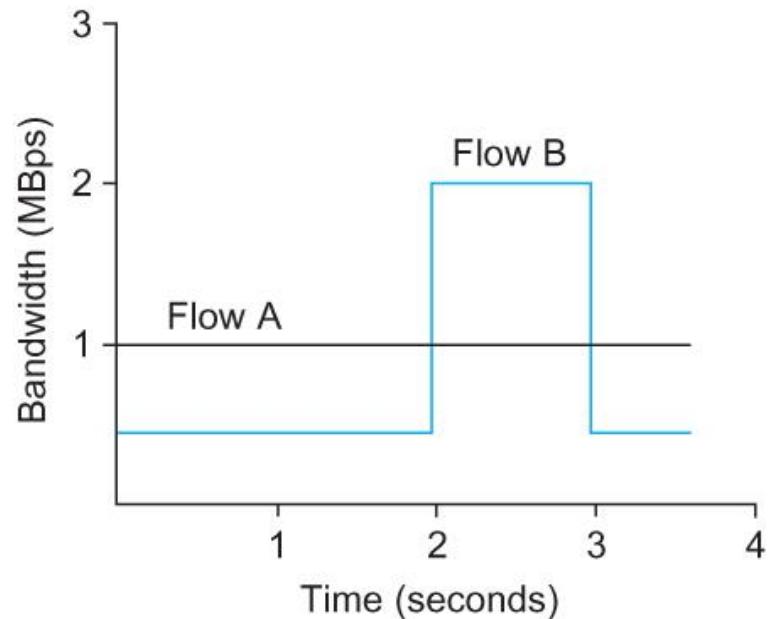


Quality of Service

105

□ Flowspec

- For the first 2 seconds, it receives tokens at a rate of 1 MBps but spends them at only 0.5 MBps,
 - So it can save up $2 \times 0.5 = 1$ MB of tokens which it spends at the 3rd second



Quality of Service

106

□ Integrated Services (RSVP)

□ Admission Control

- When some new flow wants to receive a particular level of service
- Admission control looks at the TSpec and RSpec of the flow and tries to decide if the desired service can be provided to that amount of traffic, given the currently available resources, without causing any previously admitted flow to receive worse service than it had requested.
 - If it can provide the service, the flow is admitted; if not, then it is denied.

Quality of Service

107

□ Integrated Services (RSVP)

□ Reservation Protocol

- While connection-oriented networks have always needed some sort of setup protocol to establish the necessary virtual circuit state in the switches, connectionless networks like the Internet have had no such protocols.
- However we need to provide a lot more information to our network when we want a real-time service from it.
- While there have been a number of setup protocols proposed for the Internet, the one on which most current attention is focused is called Resource Reservation Protocol (RSVP).

Quality of Service

108

□ Integrated Services (RSVP)

□ Reservation Protocol

- One of the key assumptions underlying RSVP is that it should not detract from the robustness that we find in today's connectionless networks.
- Because connectionless networks rely on little or no state being stored in the network itself, it is possible for routers to crash and reboot and for links to go up and down while end-to-end connectivity is still maintained.
- RSVP tries to maintain this robustness by using the idea of *soft state in the routers*.

Quality of Service

109

□ Integrated Services (RSVP)

□ Reservation Protocol

- Another important characteristic of RSVP is that it aims to support multicast flows just as effectively as unicast flows
- Initially, consider the case of one sender and one receiver trying to get a reservation for traffic flowing between them.
- There are two things that need to happen before a receiver can make the reservation.

Quality of Service

110

□ Integrated Services (RSVP)

□ Reservation Protocol

- First, the receiver needs to know what traffic the sender is likely to send so that it can make an appropriate reservation.
 - i.e., it needs to know the sender's TSpec.
- Second, it needs to know what path the packets will follow from sender to receiver, so that it can establish a resource reservation at each router on the path.
- Both of these requirements can be met by sending a message from the sender to the receiver that contains the TSpec.
- Obviously, this gets the TSpec to the receiver.
- Each router looks at this message (called a PATH message) as it goes past, and it figures out the *reverse path that will be used to send reservations from the receiver back to the sender* in an effort to get the reservation to each router on the path.

Quality of Service

111

□ Integrated Services (RSVP)

□ Reservation Protocol

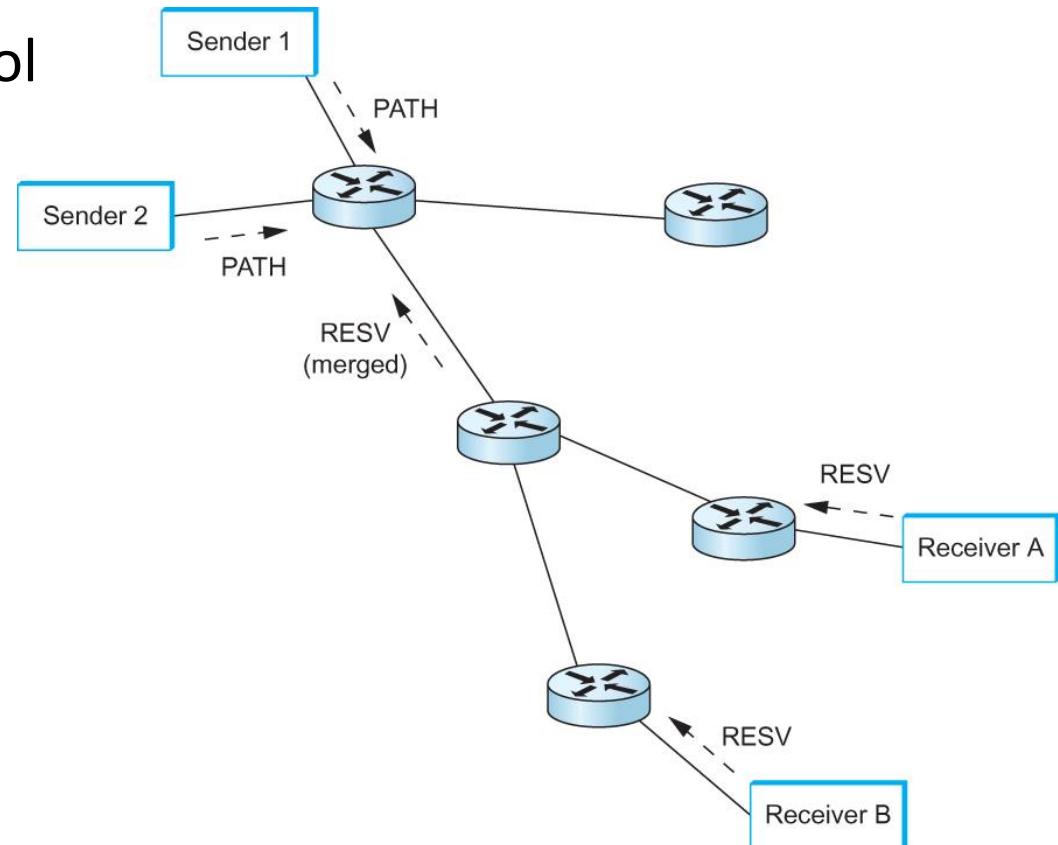
- Having received a PATH message, the receiver sends a reservation back “up” the multicast tree in a RESV message.
- This message contains the sender’s TSpec and an RSpec describing the requirements of this receiver.
- Each router on the path looks at the reservation request and tries to allocate the necessary resources to satisfy it.
- If the reservation can be made, the RESV request is passed on to the next router.
- If not, an error message is returned to the receiver who made the request.
- If all goes well, the correct reservation is installed at every router between the sender and the receiver.
- As long as the receiver wants to retain the reservation, it sends the same RESV message about once every 30 seconds.

Quality of Service

112

□ Integrated Services (RSVP)

□ Reservation Protocol



Making reservations on a multicast tree

Quality of Service

113

- Integrated Services (RSVP)
 - Packet Classifying and Scheduling
 - Once we have described our traffic and our desired network service and have installed a suitable reservation at all the routers on the path, the only thing that remains is for the routers to actually deliver the requested service to the data packets.
 - There are two things that need to be done:
 - Associate each packet with the appropriate reservation so that it can be handled correctly, a process known as *classifying packets*.
 - Manage the packets in the queues so that they receive the service that has been requested, a process known as *packet scheduling*.

Quality of Service

114

- Differentiated Services
 - Whereas the Integrated Services architecture allocates resources to **individual flows**
 - The Differentiated Services model (often called DiffServ for short) allocates resources to a small number of classes of traffic.
 - In fact, some proposed approaches to DiffServ simply divide traffic into two classes.

Quality of Service

115

□ Differentiated Services

- Suppose that we have decided to enhance the best-effort service model by adding just one new class, which we'll call "premium."
- Clearly we will need some way to figure out which packets are premium and which are regular old best effort.
- Rather than using a protocol like RSVP to tell all the routers that some flow is sending premium packets, it would be much easier **if the packets could just identify themselves to the router** when they arrive.
- This could obviously be done by using a bit in the packet header—if that bit is a 1, the packet is a premium packet; if it's a 0, the packet is best effort

Quality of Service

116

- Differentiated Services
 - With this in mind, there are two questions we need to address:
 - Who sets the premium bit, and under what circumstances?
 - What does a router do differently when it sees a packet with the bit set?

Quality of Service

117

□ Differentiated Services

- There are many possible answers to the first question, but a common approach is to set the bit at an administrative boundary.
- For example, the router at the edge of an Internet service provider's network might set the bit for packets arriving on an interface that connects to a particular company's network.
- The Internet service provider might do this because that company has paid for a higher level of service than best effort.

Quality of Service

118

- Differentiated Services
 - Assuming that packets have been marked in some way, what do the routers that encounter marked packets do with them?
 - Here again there are many answers.
 - In fact, the IETF standardized a set of router behaviors to be applied to marked packets.
 - These are called “per-hop behaviors” (PHBs), a term that indicates that they define the behavior of individual routers rather than end-to-end services

Quality of Service

119

□ Differentiated Services

□ The Expedited Forwarding (EF) PHB

- One of the simplest PHBs to explain is known as “expedited forwarding” (EF).
 - Packets marked for EF treatment should be forwarded by the router with minimal delay and loss.
- The only way that a router can guarantee this to all EF packets is if the arrival rate of EF packets at the router is strictly limited to be less than the rate at which the router can forward EF packets.

Quality of Service

120

□ Differentiated Services

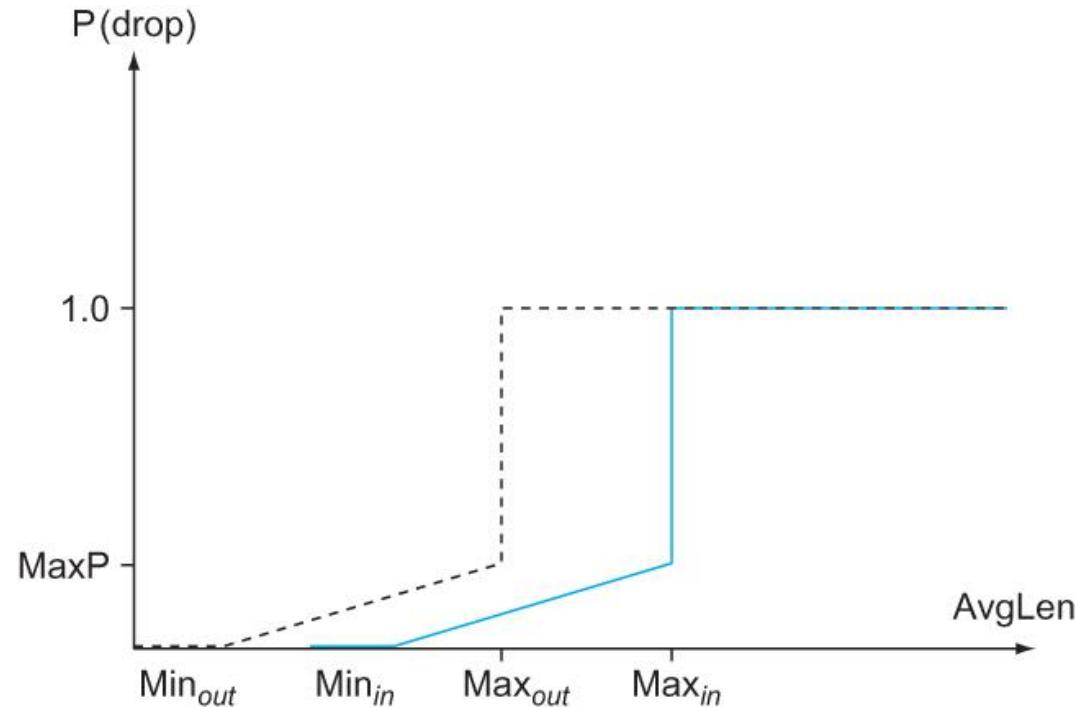
□ The Assured Forwarding (AF) PHB

- The “assured forwarding” (AF) PHB has its roots in an approach known as “RED with In and Out” (RIO) or “Weighted RED,” both of which are enhancements to the basic RED algorithm.
- For our two classes of traffic, we have two separate drop probability curves. RIO calls the two classes “in” and “out” for reasons that will become clear shortly.
- Because the “out” curve has a lower MinThreshold than the “in” curve, it is clear that, under low levels of congestion, only packets marked “out” will be discarded by the RED algorithm.
- If the congestion becomes more serious, a higher percentage of “out” packets are dropped, and then if the average queue length exceeds Min_{in} , RED starts to drop “in” packets as well.

Quality of Service

121

- Differentiated Services
 - ▣ The Assured Forwarding (AF) PHB



RED with In and Out drop probabilities

Reference

122

- Chapter – 6: *Computer Networks A Systems Approach* by Larry L. Peterson and Bruce S. Davie, 4Th Edition, Morgan Kaufmann Publications.