

# Broadband & TCP/IP fundamentals

---

Sridhar Iyer

School of Information Technology

IIT Bombay

[sri@it.iitb.ac.in](mailto:sri@it.iitb.ac.in)

[www.it.iitb.ac.in/~sri](http://www.it.iitb.ac.in/~sri)

# About the course

---

- Session 1: Aug 30<sup>th</sup> (1st half)
  - Basics of TCP/IP networks: Issues in layering
- Session 2: Aug 30<sup>th</sup> (2nd half)
  - Switching and Scheduling: Medium access, switching, queueing, scheduling.
- Session 3: Aug 31<sup>st</sup> (1st half)
  - Routing and Transport: Addressing, routing, TCP variants, congestion control
- Session 4: Aug 31<sup>st</sup> (2nd half)
  - Applications and Security: Sockets, RPC, firewalls, cryptography.

# Some Texts/References

---

- A.S. Tanenbaum. Computer Networks. Prentice Hall India, 1998.
- S. Keshav. An Engineering Approach to Computer Networks. Addison Wesley, 1997.
- L.L. Peterson and B.S. Davie. Computer Networks: A Systems Approach. Morgan Kaufmann, 1996.
- W.R. Stevens. TCP/IP Illustrated, Vol 1: The Protocols. Addison Wesley, 1994.
- D.E. Comer. and D.L. Stevens. Internetworking with TCP/IP. Vol 1-3. Prentice Hall. 1993.

# More Text/References

---

- W.R. Cheswick and S.M. Bellovin. Firewalls and Internet Security. Addison Wesley, 1994.
- W. Stallings. Cryptography and Network Security. Prentice Hall, 1999.
- P.K. Sinha. Distributed Operating Systems: Concepts and Design. Prentice Hall, 1997.
- G. Coulouris, J. Dollimore and T. Kindberg. Distributed Systems: Concepts and Design. Addison Wesley, 1994.
- RFCs, source code of implementations etc.

---

# Introduction

# Layers in a computer

---

- Hardware: CPU, Memory...
- Architecture: x86, Sparc...
- Operating system: NT, Solaris...
- Language support: C, Java...
- Application: dbms...

# The network *is* the computer

---

- Hardware: Computers and communication media
- Architecture: *standard* protocols
- Operating system: heterogeneous
- Language support: C, Java, MPI
- Application: peer-peer...

Our focus: network architecture

# Perspectives

---

- **Network designers:** Concerned with cost-effective design
  - Need to ensure that network resources are efficiently utilized and fairly allocated to different users.



# Perspectives (contd.)

---

- **Network users:** Concerned with application services
  - Need guarantees that each message sent will be delivered without error within a certain amount of time.

# Perspectives (contd.)

---

- **Network providers:** Concerned with system administration
  - Need mechanisms for security, management, fault-tolerance and accounting.

# Connectivity

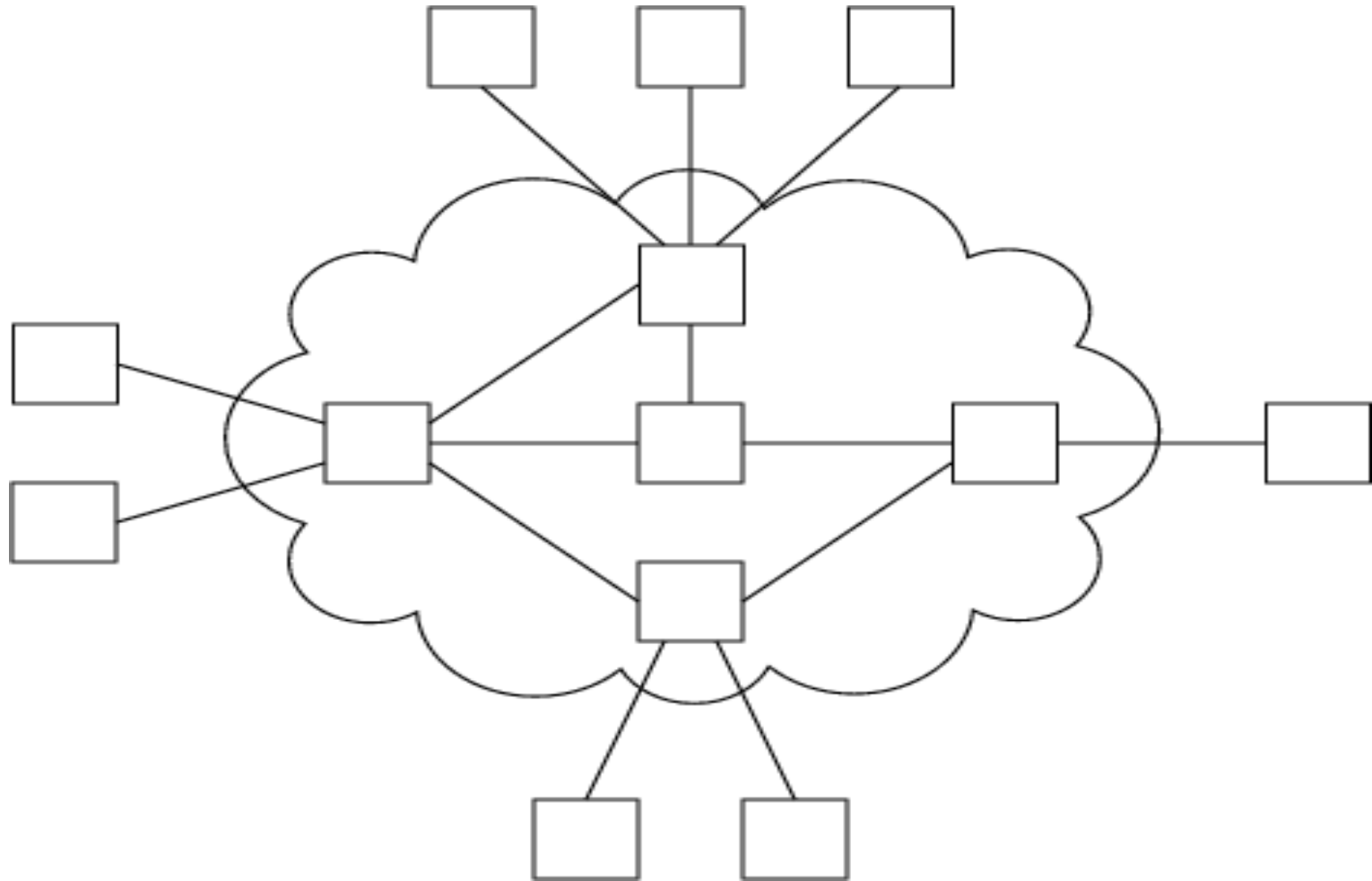
---

## Building Blocks:

- nodes: general-purpose workstations...
- links: coax cable, optical fiber...
  - Direct links: point-to-point
  - Multiple access: shared

# Switched networks

---



# Interconnection devices

---

Basic Idea: Transfer data from input to output

- Repeater
  - Amplifies the signal received on input and transmits it on output
- Modem:
  - Accepts a serial stream of bits as input and produces a modulated carrier as output (or vice versa)

# Interconnection devices (contd.)

---

- Hub
  - Connect nodes/segments of a LAN
  - When a packet arrives at one port, it is copied to all the other ports
- Switch:
  - Reads destination address of each packet and forwards appropriately to specific port
  - Layer 3 switches (IP switches) also perform routing functions

# Interconnection devices (contd.)

---

- Bridge:
  - “ignores” packets for same LAN destinations
  - forwards ones for interconnected LANs
- Router:
  - decides routes for packets, based on destination address and network topology
  - Exchanges information with other routers to learn network topology

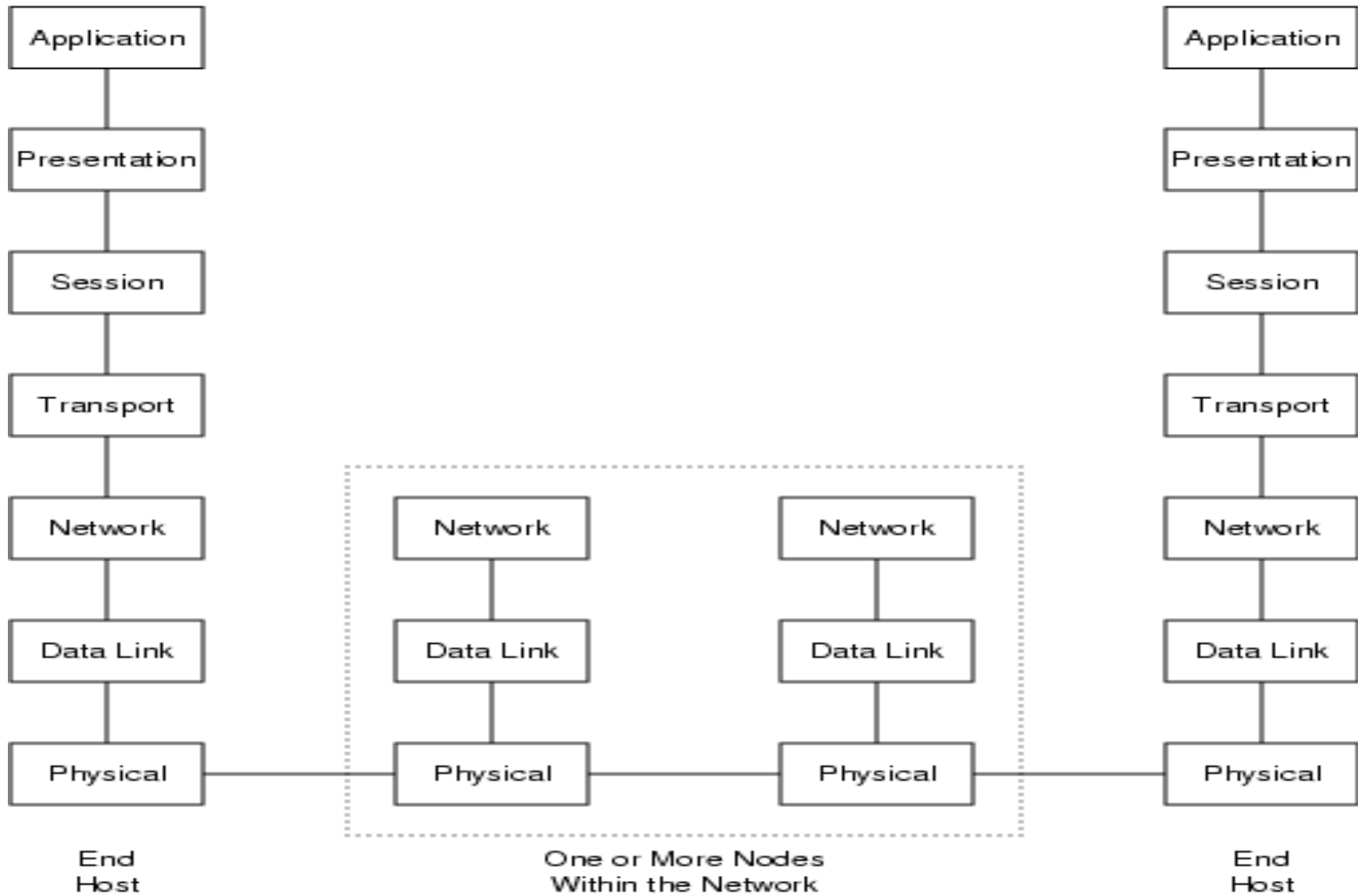
# Network architecture

---

- Layering used to reduce design complexity
  - Use abstractions for each layer
  - Can have alternative abstractions at each layer
  - If the service interface remains unchanged, implementation of a layer can be changed without affecting other layers.



# OSI architecture



# TCP/IP layers

---

- Physical Layer:
  - Transmitting bits over a channel.
  - Deals with electrical and procedural interface to the transmission medium.
- Data Link Layer:
  - Transform the raw physical layer into a 'link' for the higher layer.
  - Deals with framing, error detection, correction and multiple access.

# TCP/IP layers (contd.)

---

- Network Layer:
  - Addressing and routing of packets.
  - Deals with subnetting, route determination.
- Transport Layer:
  - end-to-end connection characteristics.
  - Deals with retransmissions, sequencing and congestion control.

# TCP/IP layers (contd.)

---

- Application Layer:
  - ``application" protocols.
  - Deals with providing services to users and application developers.
- Protocols are the building blocks of a network architecture.

# Protocols and Services

---

Each protocol object has two interfaces

- *service interface*: defines operations on this protocol.
  - Each layer provides a service to the layer Above.
- *peer-to-peer interface*: defines messages exchanged with peer.
  - Protocol of “conversation” between corresponding Layers in Sender and Receiver.

# Physical layer – Media dependent components

---

- **Copper: Coaxial/Twisted Pair**
  - Typically upto 100 Mbps
- **Fibre: Single/Multi Mode**
  - Can transmit in Gigabits/second
- **Satellite:**
  - Channels of 64 kbps, 128 kbps,...

# Physical layer – Media independent

---

- Connectors: Interface between equipment and link
- Control, clock and ground signals
- Protocols:
  - RS 232 (20 kbps, 10 ft)
  - RS 449 (2 Mbps, 60 ft)

# Data link layer functions

---

- Grouping of bits into frames
- Dealing with transmission errors
- Regulating the flow of frames
  - so that slow receivers are not swamped by fast senders
- Regulating multiple access to the medium



# Data link layer services

---

- **Unacknowledged connectionless service**
  - No acknowledgements, no connection
  - Error recovery up to higher layers
  - For low error-rate links or voice traffic
- **Acknowledged connectionless service**
  - Acknowledgements improve reliability
  - For unreliable channels. e.g.: wireless systems

# Data link layer services

---

- Acknowledged connection-oriented service
  - Equivalent of reliable bit-stream; in-order delivery
  - Connection establishment and release
  - Inter-router traffic
- Typically implemented by network adaptor
  - Adaptor fetches (deposits) frames out of (into) host memory

# Data link layer – Logical link control (LLC)

---

- Framing (start and stop)
- Error Detection
- Error Correction
- Optimal Use of Links (Sliding Window Protocol)
  - Examples: HDLC, LAP-B, LAP-D

# Data link layer – Medium access control (MAC)

---

- Multiple Access Protocols
- Channel Allocation
- Contention, Reservation, Round-robin
- Examples: Ethernet (IEEE 802.3),  
Token Ring (802.5)

# Network layer

---

- Need for network layer
  - All machines are not Ethernet!
  - Hide type of subnet (Ethernet, Token Ring, FDDI)
  - Hide topology of subnets
- Scheduling
- Addressing
- Routing

# Network layer functions

---

- Internetworking
  - uniform addressing scheme
- Routing
  - choice of appropriate paths from source to destination
- Congestion Control
  - avoid overload on links/routers

# Addressing

---

Address: byte-string that identifies a node

- physical address: device level
- network address: network level
- logical address: application level
  
- unicast: node-specific
- broadcast: all nodes on the network
- multicast: some subset of nodes

# Routing

---

- Mechanisms of forwarding messages towards the destination node based on its address
- Need to learn global information
- Queueing (buffering)
- Scheduling



# Connection Oriented service

---

- Network layer at sender must set up a connection to its peer at the receiver
- Negotiation about parameters, quality, and costing are possible
- Avoids having to choose routes on a per packet basis

# Connectionless service

---

- Network layer at sender simply puts the packet on the outgoing link without connection setup
- Intermediate nodes use routing tables to deliver the packet to destination
- Avoids connection setup delays

# Circuit switching

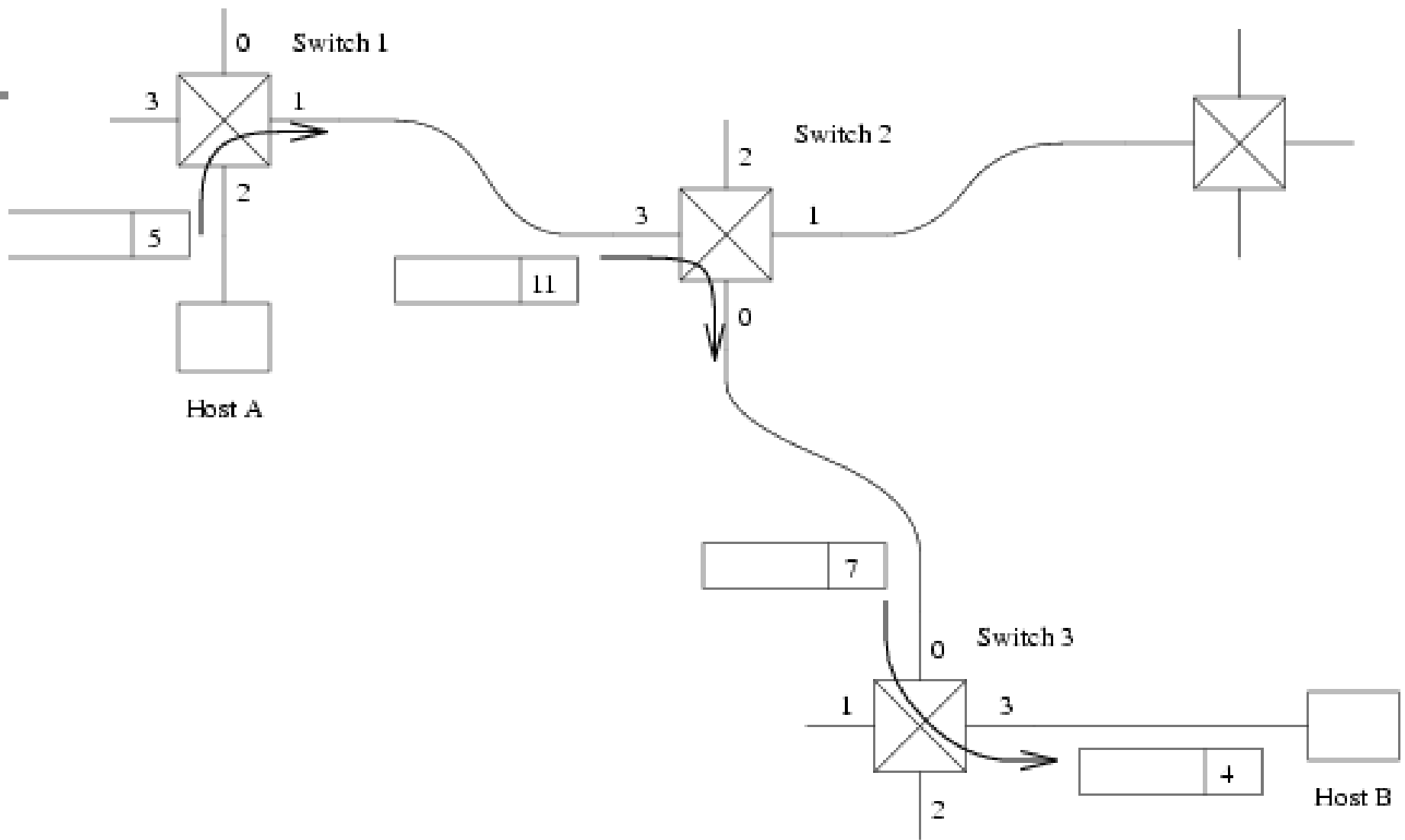
---

- dedicated circuit for sender-receiver.
- end-to-end path setup before actual communication.
- no congestion for an established circuit connection.
- resources are reserved; only propagation delays.
- unused bandwidth on an allocated circuit is wasted.

# Virtual Circuits

---

- Used in subnets whose primary service is connection-oriented
  - During connection setup, a route from the source to destination is chosen and remembered
  - Packets contain a circuit identifier rather than full destination address
- Disadvantages:
  - Connection setup overhead
  - If a link/node along the route fails all VCs are terminated



# Packet switching (datagrams)

---

- Used in subnets whose primary service is connectionless
- Routes are not worked out in advance
  - Successive packets may follow different routes
  - No connection setup overhead
- Disadvantages :
  - Packets carry full addresses and are larger
  - Routing decisions have to be made for every packet
  - typically ``best-effort" service; may face congestion.

# Transport layer

---

- Lowest end-to-end service
- Main Issues:
  - Reliable end-to-end delivery
  - Flow control
  - Congestion control
  - providing guarantees
- Depends on application requirements

# Application requirements

---

- Best-effort: FTP
- Bandwidth guarantees: Video
  - burst versus peak rate
- Delay guarantees: Video
  - jitter: variance in latency (inter-packet gap)



---

# Bandwidth and Multiplexing

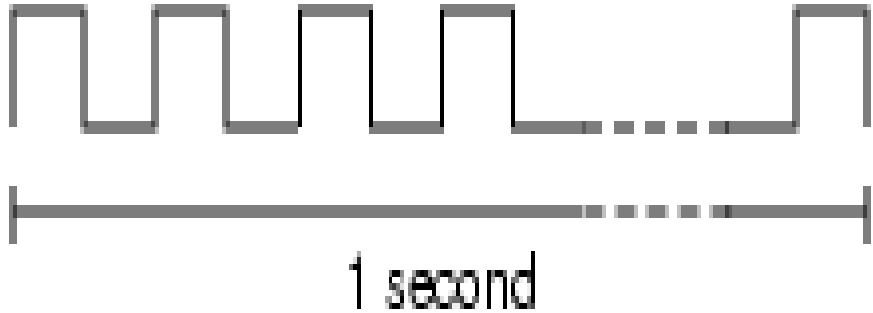
# Bandwidth

---

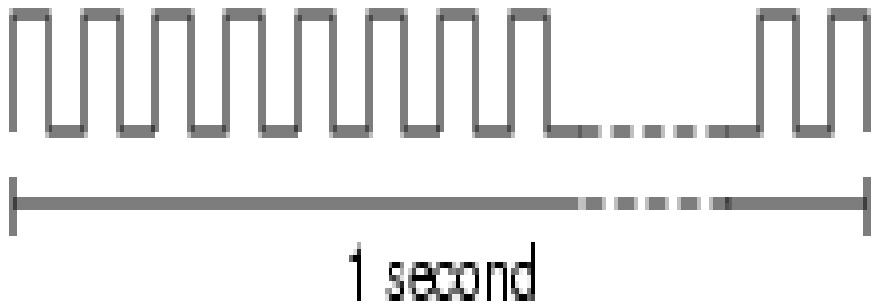
- Amount of data that can be transmitted per unit time
  - expressed in cycles per second, or Hertz (Hz) for analog devices
  - expressed in bits per second (bps) for digital devices
  - KB =  $2^{10}$  bytes; Mbps =  $10^6$  bps
- Link v/s End-to-End

# Bandwidth v/s bit width

---



1 Mbps  
(each bit 1 microseconds wide)



2 Mbps  
(each bit 0.5 microseconds wide)

# Latency (delay)

---

- Time it takes to send message from point A to point B
  - Latency = Propagation + Transmit + Queue
  - Propagation = Distance / SpeedOfLight
  - Transmit = Size / Bandwidth

# Latency

---

- Queueing not relevant for direct links
- Bandwidth not relevant if  $\text{Size} = 1$  bit
- Process-to-process latency includes software overhead
- Software overhead can dominate when Distance is small
  
- RTT: round-trip time

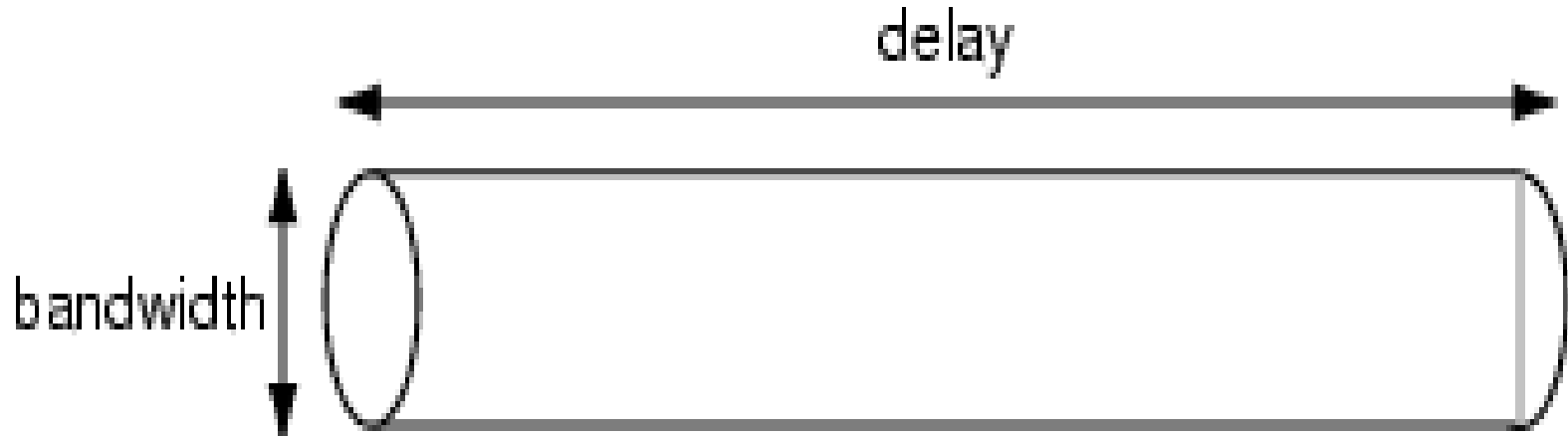
# Delay X Bandwidth product

---

- Relative importance of bandwidth and delay
- Small message: 1ms vs 100ms dominates 1Mbps vs 100Mbps
- Large message: 1Mbps vs 100Mbps dominates 1ms vs 100ms

# Delay X Bandwidth product

---

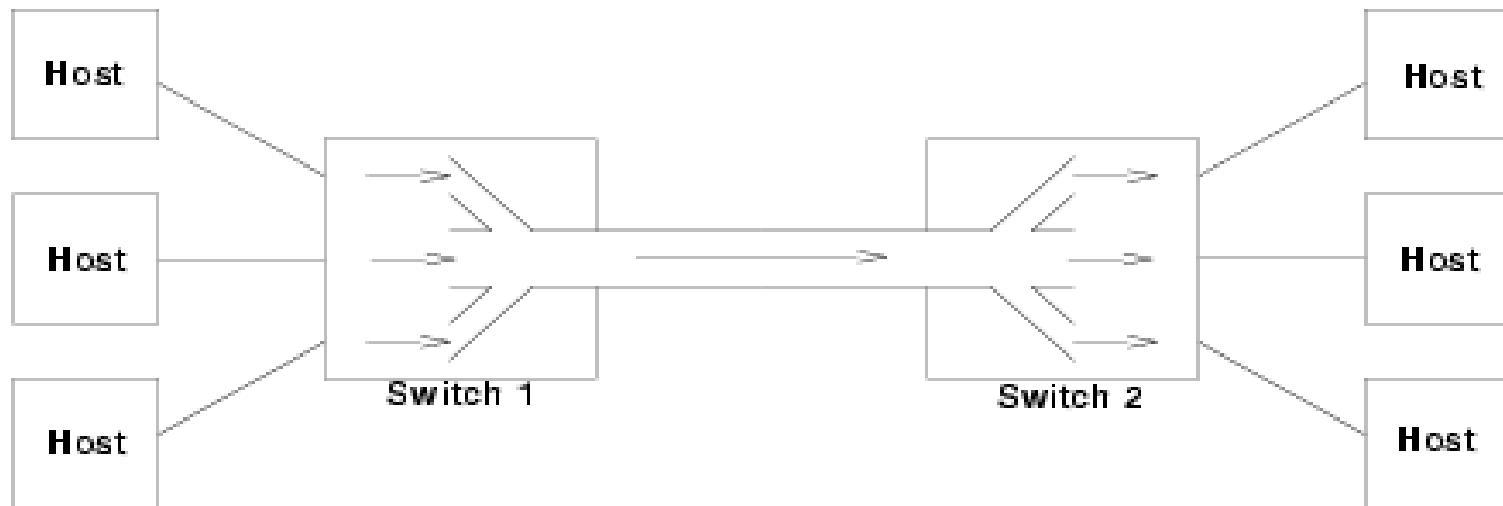


100ms RTT and 45Mbps Bandwidth =  
560 KB of data

# Effective resource sharing

---

Need to share (*multiplex*) network resources (nodes and links) among multiple users.





# Common multiplexing strategies

---

- Time-Division Multiplexing (TDM):
  - Each user periodically gets the entire bandwidth for a small burst of time.
- Frequency-Division Multiplexing (FDM):
  - Frequency spectrum is divided among the logical channels.
  - Each user has exclusive access to his channel.

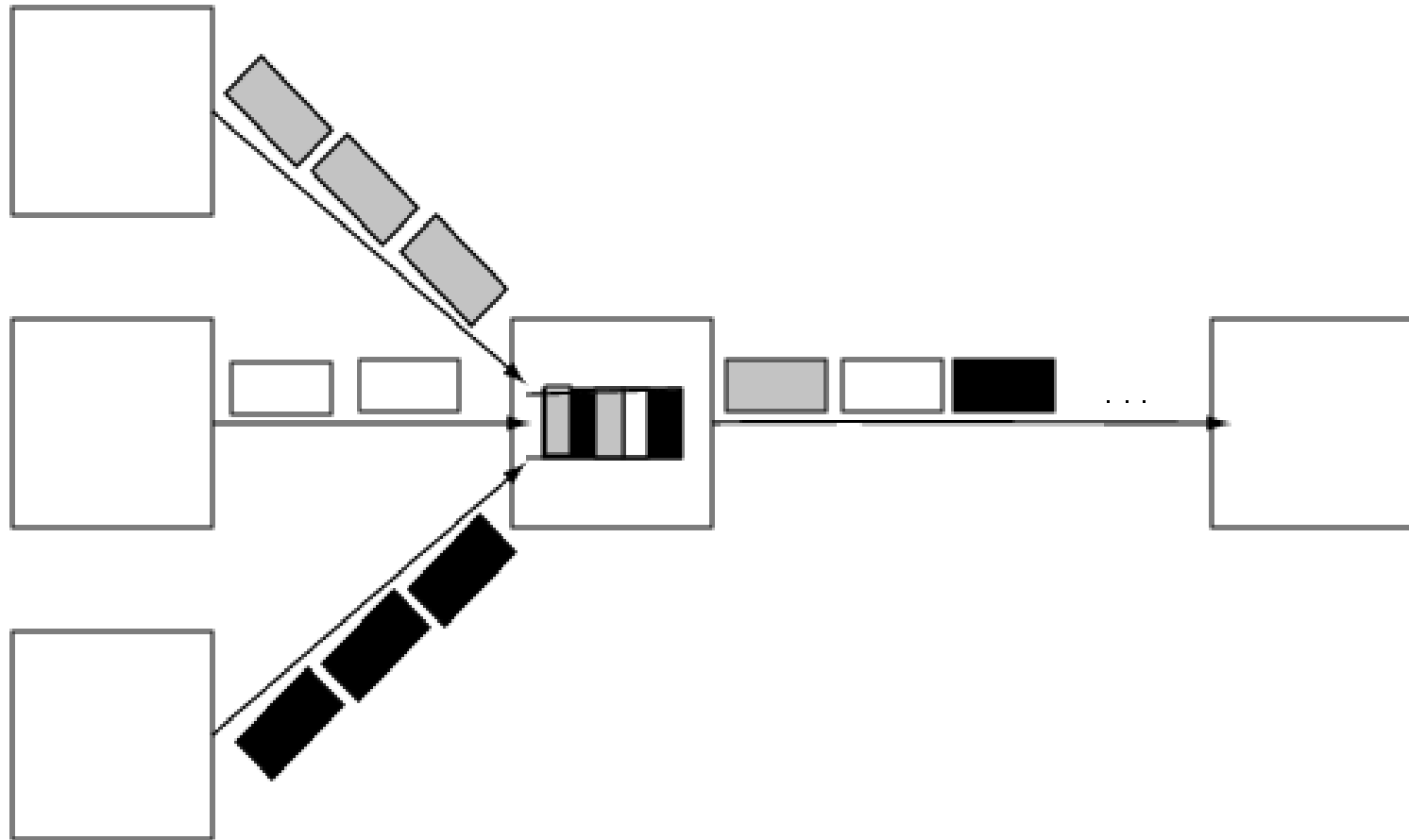
# Statistical multiplexing

---

- Time-division, but “on demand” (not fixed)
- Reschedule link on a per-packet basis
  - Packets from different sources are interleaved
  - Buffer packets that are contending for the link
  - Packet queue may be processed FIFO, but not necessarily
- Buffer overflow is called congestion

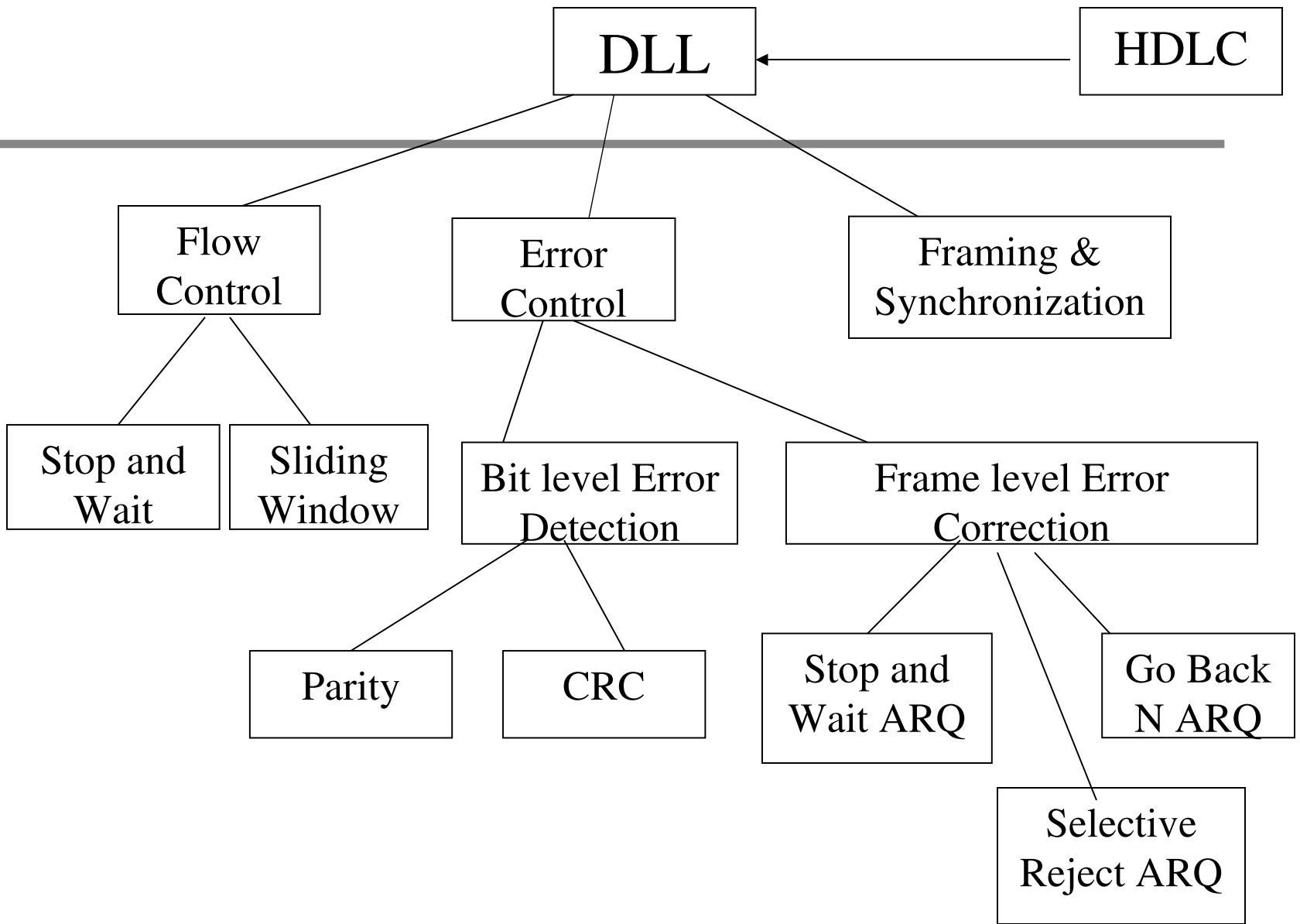
# Statistical multiplexing

---



---

# Error detection and correction



# Bit level error detection/correction

---

Single-bit, multi-bit or burst errors introduced due to channel noise.

- Detected using redundant information sent along with data.
- Full Redundancy:
  - Send everything twice
  - Simple but inefficient

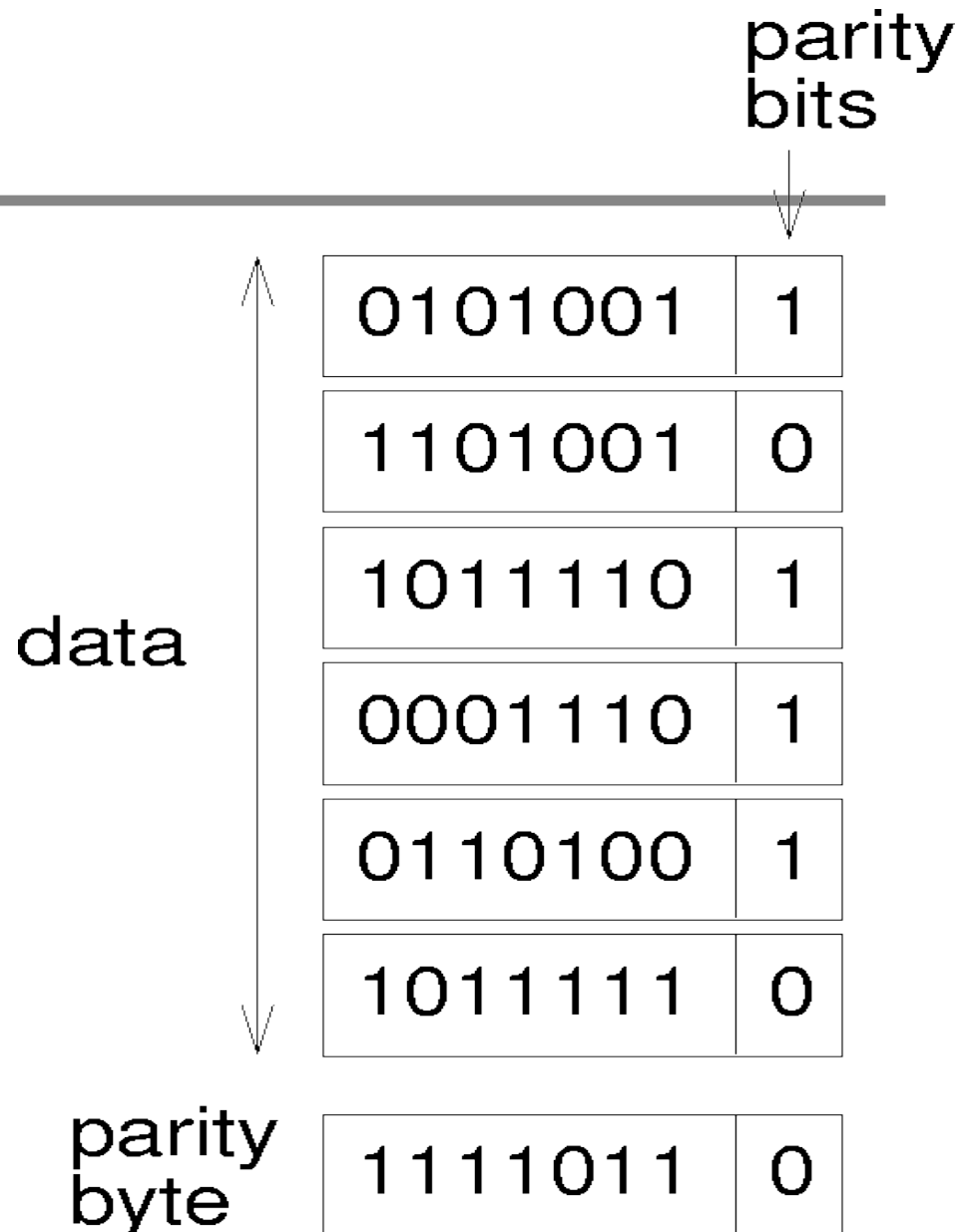
# Parity

## Parity (horizontal)

- 1 bit error detectable, not correctable
- 2 bit error not detectable

## Parity (rectangular)

- 1 bit error correctable
- 2 bit error detectable
- Slow, needs memory



# Cyclic Redundancy Check (CRC)

---

- Based on binary division instead of addition.
- Powerful and commonly used to ‘detect’ errors.
  - Rarely for ‘correction’
- Uses modulo 2 arithmetic:
  - Add/Subtract := XOR (no carries for additions or borrows for subtraction)
  - $2^k * M$  := shift M towards left by k positions and then pad with zeros
- Digital logic for CRC is fast. no delay, no storage



# CRC algorithm

---

To transmit message  $M$  of size of  $n$  bits

- Source and destination agree on a common bit pattern  $P$  of size  $k+1$  ( $k > 0$ )
- Source does the following:
  - Add (in modulo 2) bit pattern ( $F$ ) of size  $k$  to the message  $M$  ( $k < n$ ), such that
  - $2^k * M + F = T$  is evenly divisible (modulo 2) by pattern  $P$ .
- Receiver checks if above condition is true
  - i.e.  $(2^k * M + F) / P = 0$

# Example

- $M = 10011010$
- $P = 1101$

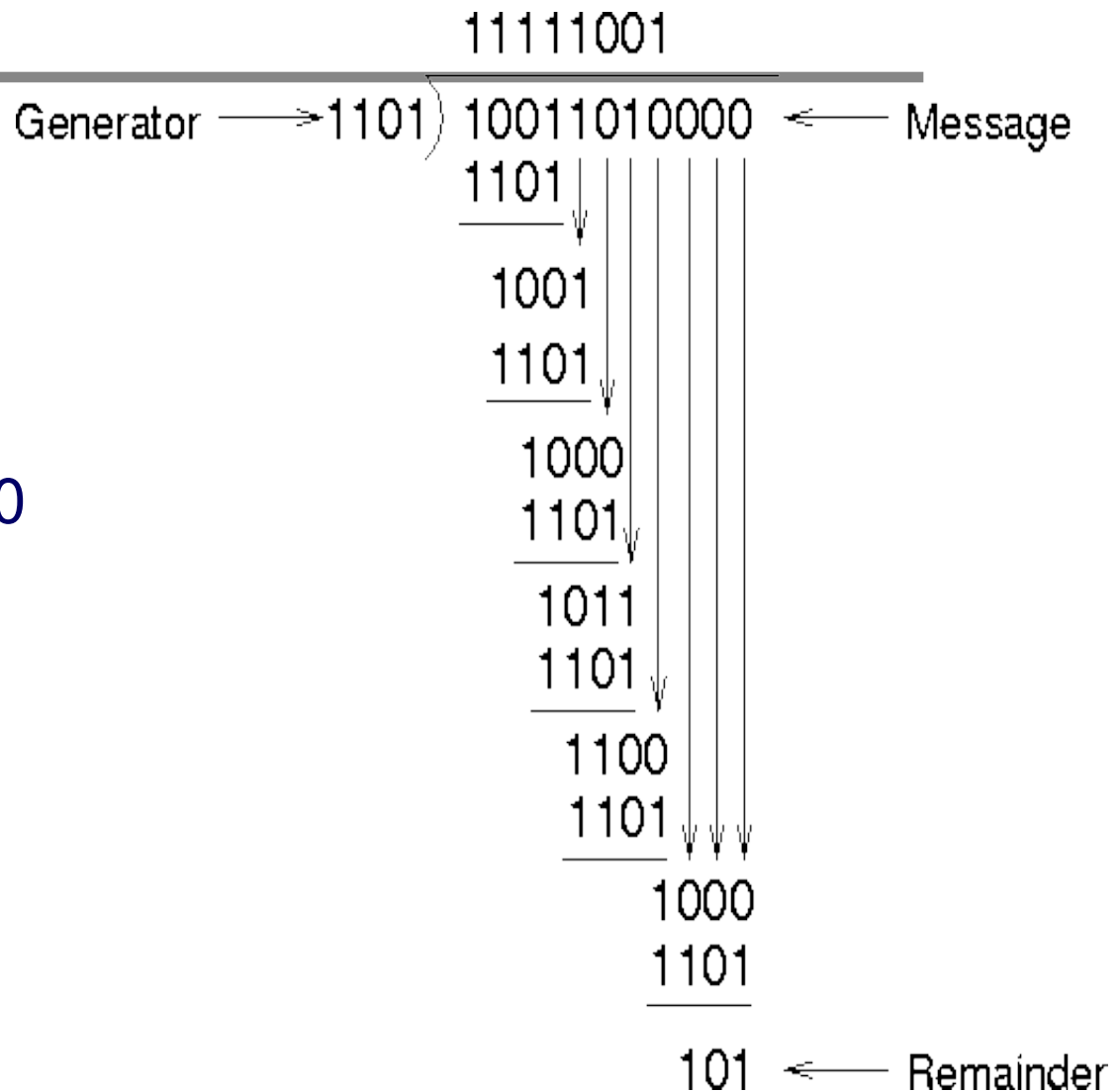
$$M * 2^3 = 10011010000$$

$$F = 10011010000 / 1101 = 101$$

$$T = 10011010000 + 101 = 10011010101$$

At receiver

- $T/P \Rightarrow$  No remainder



# Frame Check Sequence (FCS)

---

- Given  $M$  (message of size  $n$ ) and  $P$  (generator polynomial of size  $k+1$ ), find appropriate  $F$  (frame check sequence)
  1. Multiply  $M$  with  $2^k$  (add  $k$  zeros to end of  $M$ )
  2. Divide (in modulo 2) the product by  $P$ 
    - » The remainder  $R$  is the required FCS
  3. Add the remainder  $R$  to the product  $2^k * M$
  4. Transmit the resultant  $T$

# Polynomial representation

---

- Represent  $n$ -bit message as an  $n-1$  degree polynomial;
  - $M=10011010$  corresponds to  $M(x) = x^7 + x^4 + x^3 + x^1$ .
- Let  $k$  be the degree of some divisor polynomial  $C(x)$ ; (also called Generator Polynomial)
  - $P = 1101$  corresponds to  $C(x) = x^3 + x^2 + 1$ .
- Multiply  $M(x)$  by  $x^k$ ;
  - $10011010000: x^{10} + x^7 + x^6 + x^4$
- Divide result by  $C(x)$  to get remainder  $R(x)$ ;
  - $10011010000/1101 = 101$
- Send  $P(x); 10011010000 + 101 = 10011010101$

# Generator polynomials

---

- Receive  $P(x) + E(x)$  and divide by  $C(x)$ 
  - $E(x)$  represents the error with 1s in position of errors
- Remainder zero only if:
  - $E(x) = 0$  (no transmission error), or
  - $E(x)$  is exactly divisible by  $C(x)$ .
- Choose  $C(x)$  to make second case extremely rare.
  - **CRC-8**  $x^8 + x^2 + x^1 + 1$
  - **CRC-10**  $x^{10} + x^9 + x^5 + x^4 + x^1 + 1$
  - **CRC-12**  $x^{12} + x^{11} + x^3 + x^2 + 1$
  - **CRC-16**  $x^{16} + x^{15} + x^2 + 1$
  - **CRC-32**  $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

# Internet checksum

---

- IP header; TCP/UDP segment checksum.
  - View message as sequence of 16-bit integers.
  - Add these integers using 16-bit ones-complement arithmetic.
  - Take the ones-complement of the result.
  - Resulting 16-bit number is the checksum.
  - Receiver repeats the operation and matches the result with the checksum.
- Can detect all 1 bit errors.
- speed of operation; less erratic channels

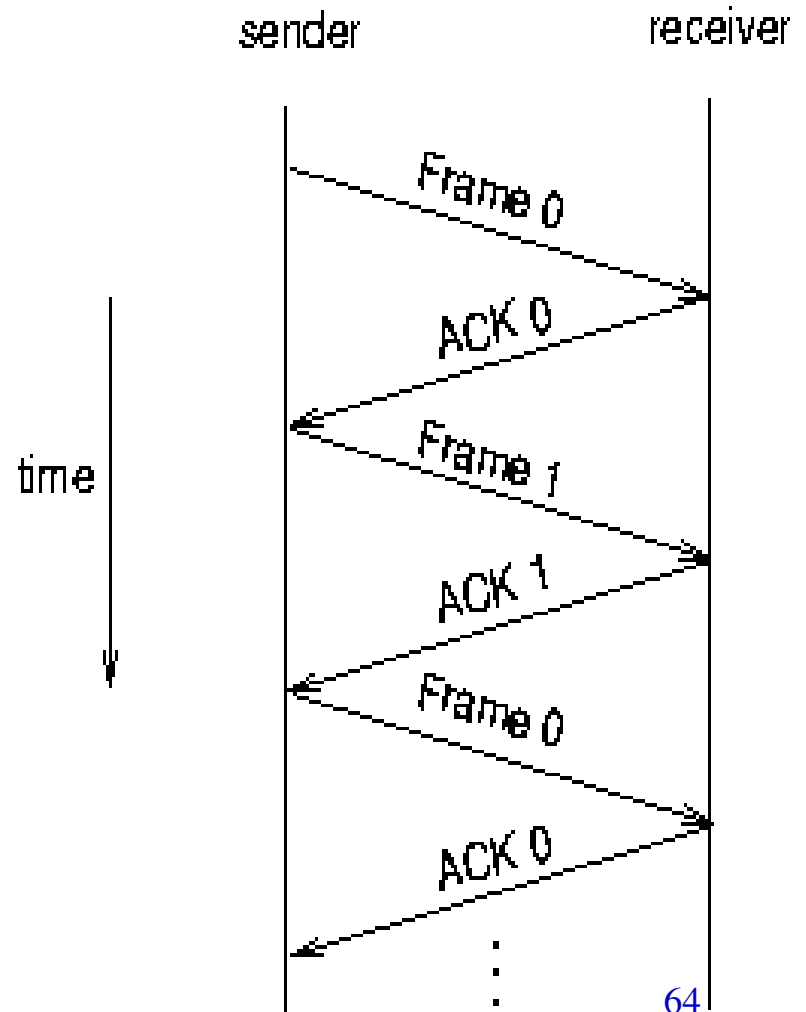
# Frame level error correction

---

- Problems in transmitting a sequence of frames over a lossy link
  - frame damage, loss, reordering, duplication, insertion
- Solutions:
  - Forward Error Correction (FEC)
    - » Use of redundancy for packet level error correction
  - Automatic Repeat Request (ARQ)
    - » Use of acknowledgements and retransmission

# Stop and Wait ARQ

- Sender waits for acknowledgement (ACK) after transmitting each frame; keeps copy of last frame.
- Receiver sends ACK if received frame is error free and NACK if received frame is in error.
- Sender retransmits frame if ACK/NACK not received before timer expires.





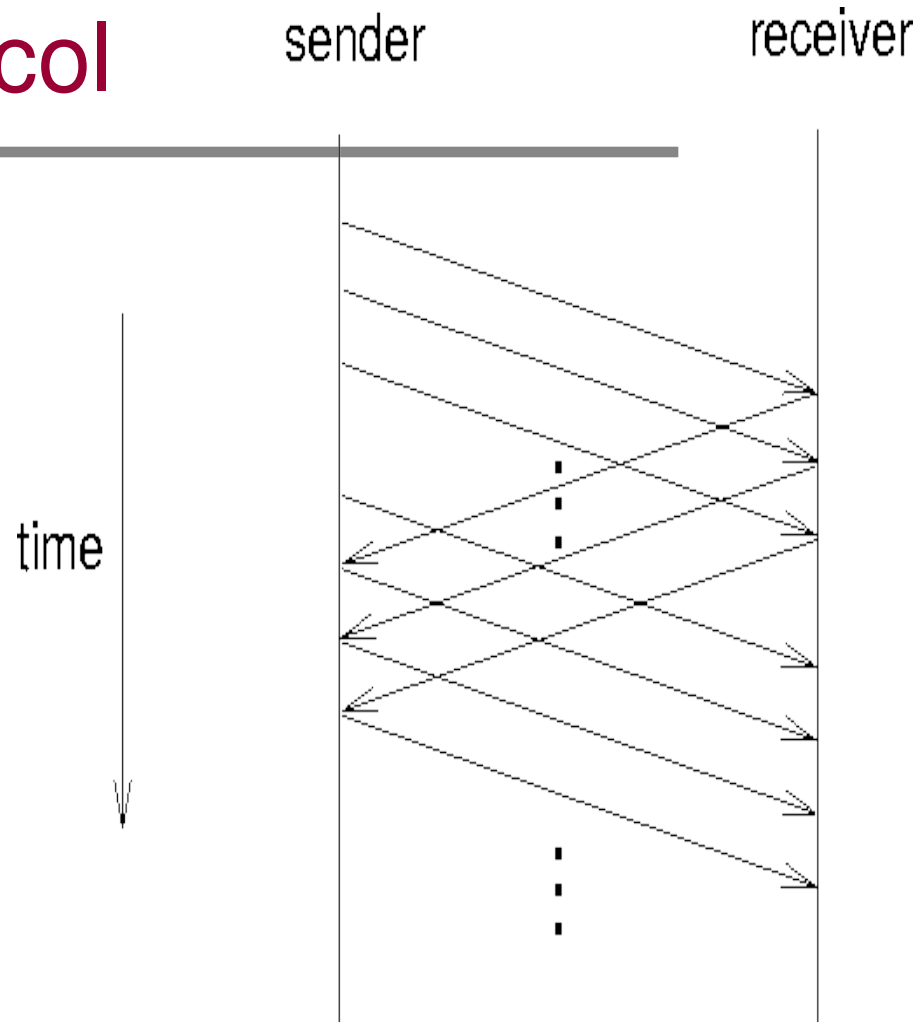
# Stop and Wait ARQ

---

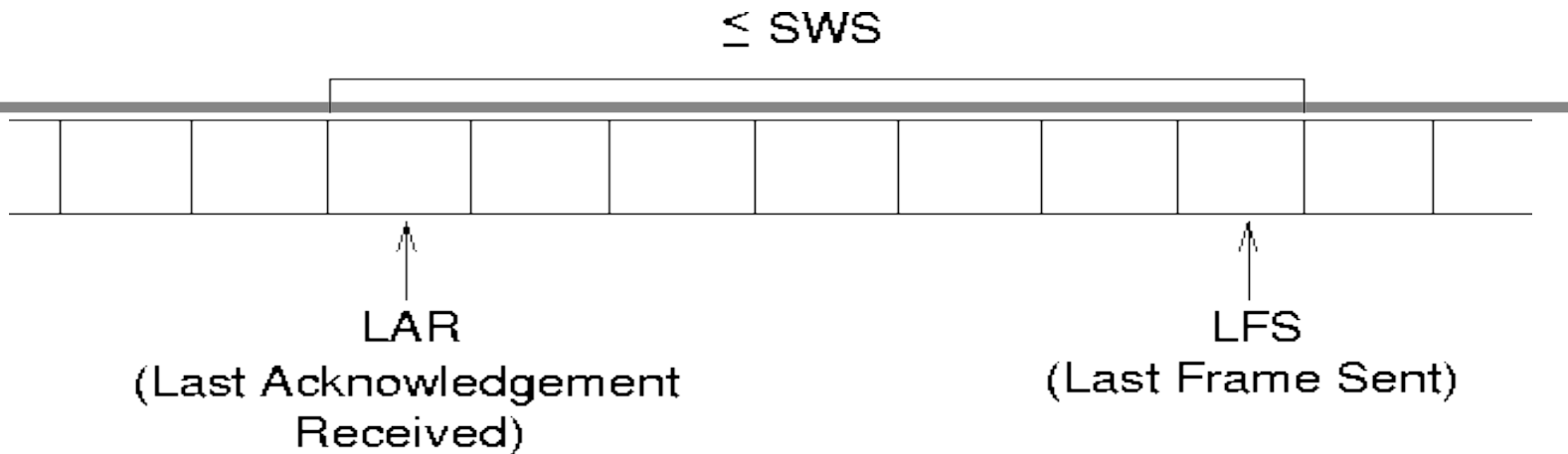
- Frames and ACKs need to be numbered for identifying duplicate transmissions
  - alternating 0 or 1.
- Simple to implement but may waste bandwidth;
  - Example: 1.5Mbps link 45ms RTT = 67.5Kb (8KB).
  - Assuming frame size of 1KB,
  - stop-and-wait uses one-eighth of the link's capacity.
  - Sender should be able to transmit up to 8 frames before having to wait for an ACK.

# Sliding Window Protocol

- Allows sender to transmit multiple frames before receiving an ACK.
- Upper limit on number of outstanding (un-ACKed) frames.
- Sender buffers all transmitted frames until they are ACKed.
- Receiver may send ACK (with SeqNum of next frame expected) or NACK (with SeqNum of damaged frame received).



# Sliding window sender



- Assign sequence number to each frame (SeqNum)
- Maintain three state variables:
  - send window size (SWS)
  - last acknowledgment received (LAR)
  - last frame sent (LFS)
  - Maintain invariant:  $LFS - LAR < SWS$
- When ACK arrives, advance LAR, thereby opening window
- Buffer up to SWS frames

# Sliding window receiver



- Maintain three state variables:
  - receive window size (RWS)
  - last frame accepted (LFA)
  - next frame expected (NFE)
  - Maintain invariant:  $LFA - NFE < RWS$
- Frame SeqNum arrives:
  - if SeqNum is in between NFE and LFA, accepted
  - if SeqNum is not in between NFE and LFA, discarded
- Send cumulative ACK.

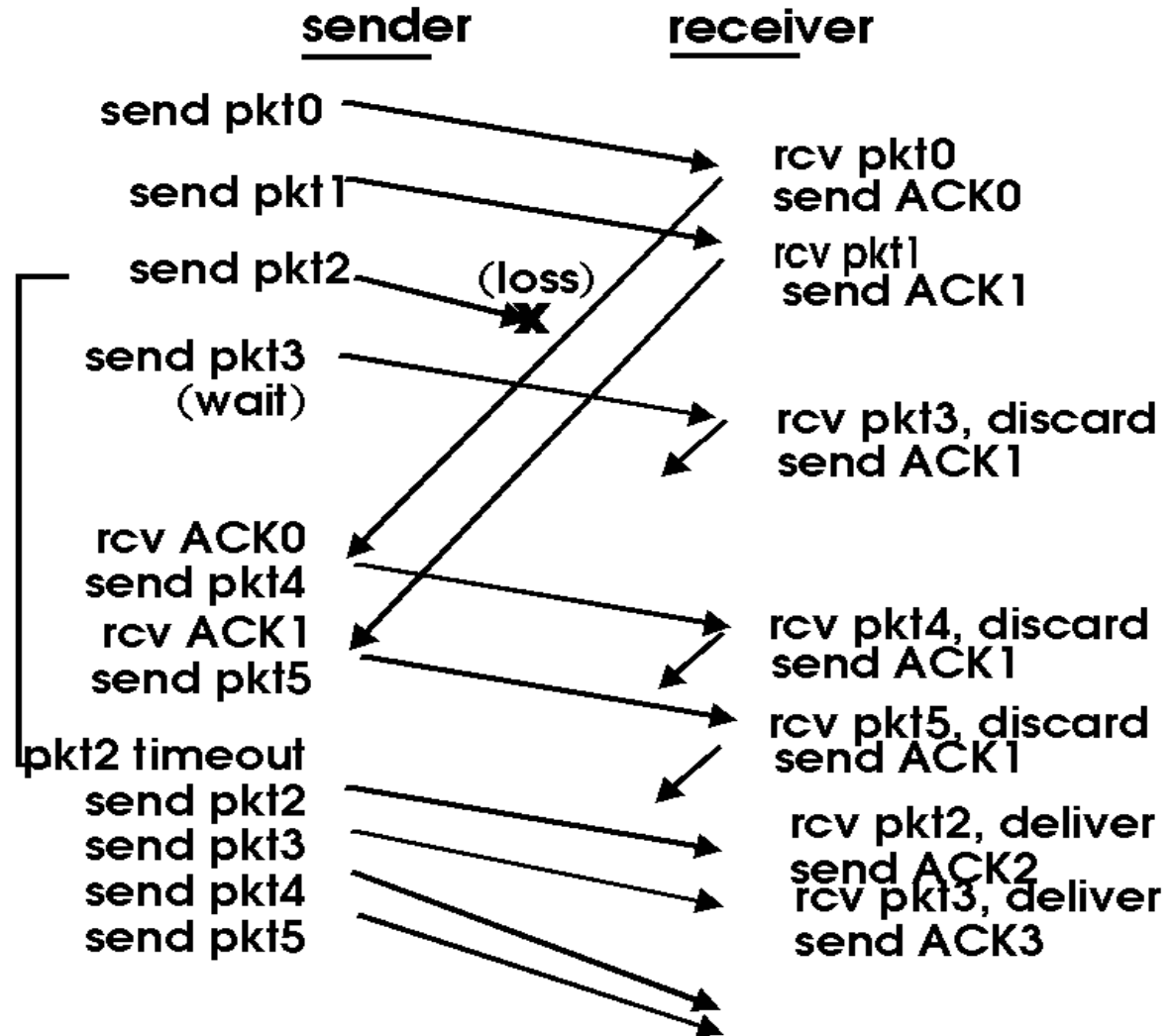
# Sliding window features

---

- ACKs may be cumulative.
  - ACK-6 implies all frames upto 5 received correctly;
  - NACK-4 implies frame 4 in error but frames upto 3 received correctly.
- SeqNum field is wrap around.
- Window size must be smaller than MaxSeqNum.

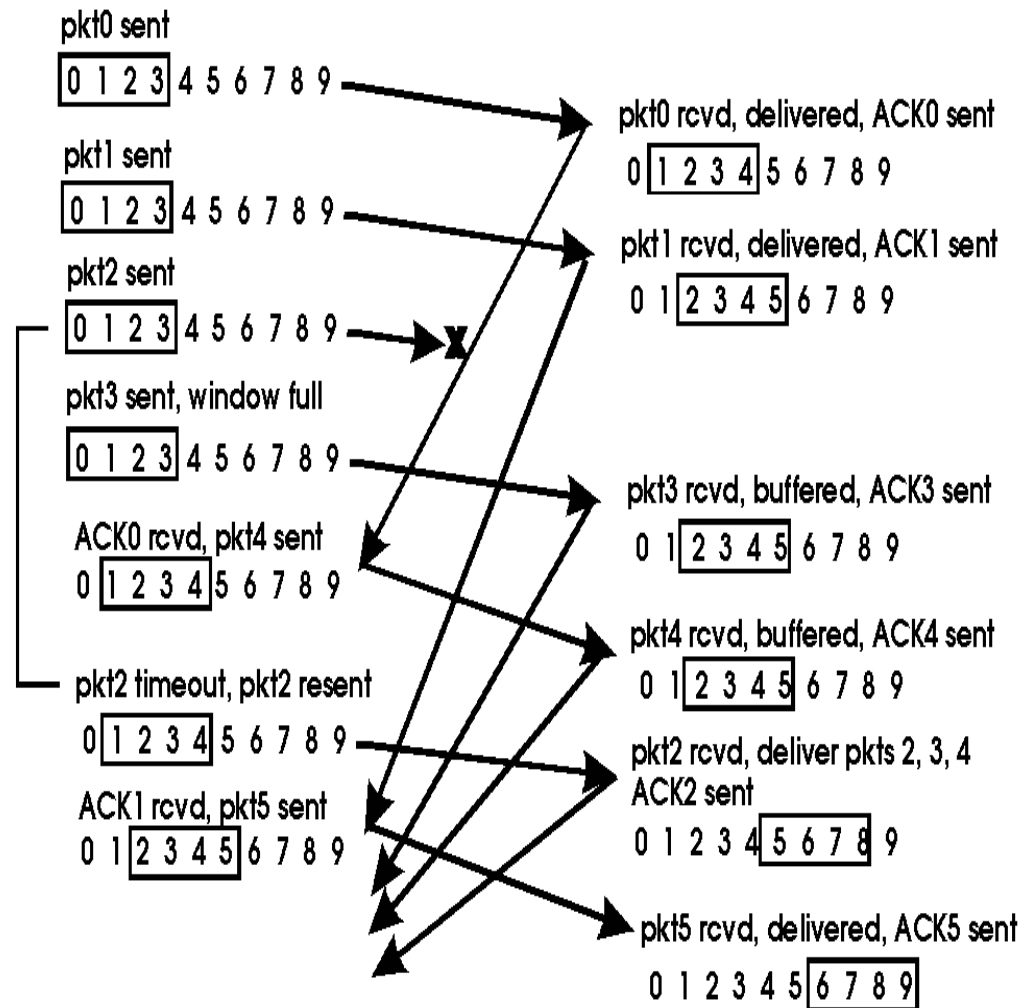
# Go-back-N ARQ

- Sliding window protocol
- Receiver discards out-of-seq pkt received and ACKs LFA.
- Simplicity in buffering & processing



# Selective Repeat ARQ

- Sliding window protocol
- Receiver ACKs correctly received out-of-sequence packets
- Sender retransmits packet upon ACK timeout or NACK (selective reject)



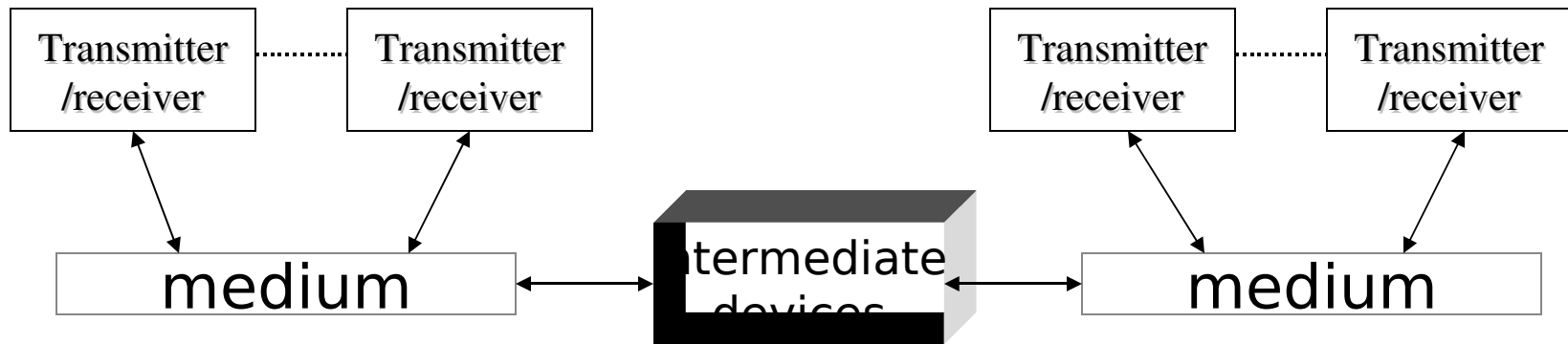
---

# Medium Access Control



# Multiple access

---



**problem:** control the access so that

- the number of messages exchanged per second is maximized
- time spent waiting for a chance to transmit is minimized

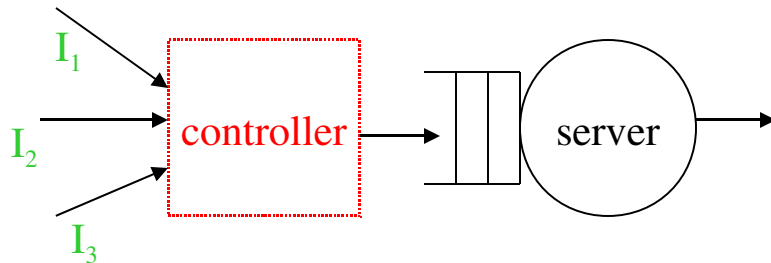
# Control methods

---

## Where ?

- **Centralized**

A controller grants access to the network



- **Distributed**

The stations collectively determine the order of transmission

## How ?

- **Synchronous**

Specific capacity dedicated to a connection

- **Asynchronous**

In response to immediate needs  
-> dynamic

- **Free for all**

Transmit freely

- **Scheduled**

Transmit only during reserved intervals

# Performance metrics

---

- Throughput (normalized) or *goodput*:
  - Fraction of link capacity devoted to carrying non-retransmitted packets
  - excludes time lost to protocol overhead, collisions etc.
  - Example: 1Mbps link can ideally carry 1000 packets/sec of size 125 bytes;
  - If a scheme reduces throughput to 250 packets/sec then goodput of scheme is 0.25.

# Performance metrics (contd.)

---

- Mean delay
  - amount of time a station has to wait before it successfully transmits a packet
- Stability
  - No/minimal decrease in throughput with increase in offered load (number of stations transmitting).
- Fairness
  - Every station should have an opportunity to transmit within a finite waiting time (*no-starvation*).

# ALOHA

---

Stations transmit whenever they have data to send

- Detect collision or wait for acknowledgment
- If no acknowledgment (or collision), try again after a random waiting time

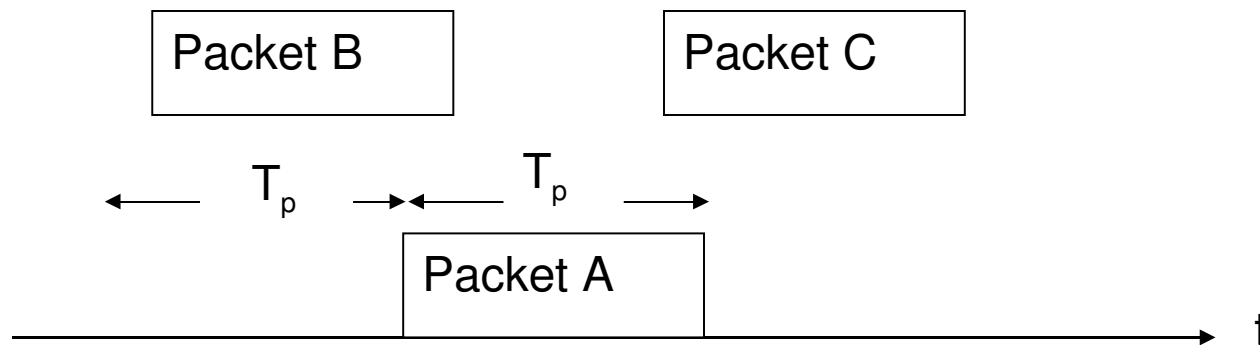
Collision: If more than one node transmits at the same time.

If there is a collision, all nodes have to re-transmit packets

# Vulnerable window

---

- For a given frame, the time when no other frame may be transmitted if a collision is to be avoided.
- Assume all packets have same length ( $L$ ) and require  $T_p$  seconds for transmission
- Each packet vulnerable to collisions for time  $V_p = ??$



# Vulnerable window

---

- Suppose packet A sent at time  $t_0$
- If pkt B sent any time in  $[t_0 - T_p$  to  $t_0]$ 
  - end of packet B collides with beginning of packet A
- If pkt C sent any time in  $[t_0$  to  $t_0 + T_p]$ 
  - start of packet C will collide with end of packet A
- Total vulnerable interval for packet A is  $2T_p$

# Slotted ALOHA

---

- Time is divided into slots
  - *slot = one packet transmission time at least*
- Master station generates *synchronization pulses* for time-slots.
- Station waits till beginning of slot to transmit.
- *Vulnerability Window* reduced from  $2T$  to  $T$ ; *goodput* doubles.



# ALOHA summary

---

- Fully distributed, S-Aloha – needs global sync
- Relatively cheap, simple to implement
- Good for sparse, intermittent communication.
- not a good LAN protocol because of
  - poor utilization (36%)
  - potentially infinite delay
  - stations have listening capability, but don't fully utilize it
- Still used in uplink cellular, GSM

# Carrier Sense Multiple Access (CSMA)

---

- Listen before you speak
- Check whether the medium is active before sending a packet (i.e. *carrier sensing*)
- If medium idle, then transmit
- If collision happens, then detect and resolve
  
- If medium is found busy, transmission follows:
  - 1- persistent
  - P- persistent
  - Non-persistent

# 1 - Persistent CSMA

---

1 - persistent CSMA is selfish

- Sense the channel.
- IF the channel is *idle*, THEN transmit.
- IF the channel is *busy*, THEN continue to listen until channel is *idle*.
- Now transmit immediately.

Collisions in case of several waiting senders

# P - Persistent CSMA

---

p - persistent CSMA is a slotted approximation.

- Sense the channel.
- IF the channel is *idle*, THEN
  - with probability  $p$  transmit and
  - with probability  $(1-p)$  delay for *one time slot* and start over.
- IF the channel is *busy*, THEN delay *one time-slot* and start over.

# Choice of p

---

- Time slot is usually set to the maximum propagation delay.
- as p decreases,
  - stations wait longer to transmit, but
  - the number of collisions decreases
- Considerations for the choice of p:
  - if  $np > 1$ : secondary transmission likely.
  - So  $p < 1/n$
  - Large n needs small p which causes delay

# Non-Persistent CSMA

---

nonpersistent CSMA is less greedy

- Sense the channel.
- IF the channel is *idle*, THEN transmit.
- If the channel is *busy*, THEN wait a *random amount of time* and start over.
  
- Random time needs to be chosen appropriately

# Collision detection (CSMA/CD)

---

- All aforementioned scheme can suffer from collision
- Device can detect collision
  - Listen while transmitting
  - Wait for  $2 * \text{propagation delay}$
- On collision detection wait for random time before retrying
- Binary Exponential Backoff Algorithm
  - Reduces the chances of two waiting stations picking the same random time

# Binary Exponential Backoff

---

1. On detecting 1<sup>st</sup> collision for packet x  
station A chooses a number r between 0 and 1.  
wait for  $r * \text{slot time}$  and transmit.

Slot time is taken as  $2 * \text{propagation delay}$

- k. On detecting k<sup>th</sup> collision for packet x  
choose r between  $0, 1, \dots, (2^k - 1)$
- When value of k becomes high (10), give up.
  - Randomization increase with larger window, but delay increases.



# Example: Ethernet (IEEE 802.3)

---

- Ethernet Address (48 bits)
  - Example: 08:00:0D:01:74:71
- Ethernet Frame Format

Pre- amble (7)	S F D	Destinatio n Address	Source Address (6)	L (2)	Data (46-1500)	FCS (4)
----------------------	-------------	----------------------------	--------------------------	----------	-------------------	------------

# 802.3 frame

---

- Preamble (7 bytes) - 0101...
- SFD - Start Frame Delimiter - 10101011
- Length (2 bytes) - length (in bytes) of data field
- Data (46-1500 bytes)
- FCS - Frame Check Sequence (4 bytes) - error checking
- May contain LLC header
- Minimum size of frame is 64 bytes (51.2 $\mu$ s)

Pre- amble (7)	S F D	Destinatio n Address	Source Address (6)	L (2)	Data (46-1500)	FCS (4)
----------------------	-------------	----------------------------	--------------------------	----------	-------------------	------------

# Collision free protocols

---

- For long cables, propagation delay is increased, decreasing the performance of CSMA/CD.
- Collision free protocols reserve time slots for nodes, thus avoiding collisions.
- Also called as reservation protocols.
  - Bit map reservation protocol
  - Adaptive tree walk protocol

---

# Bridging and Switching

# Bridges

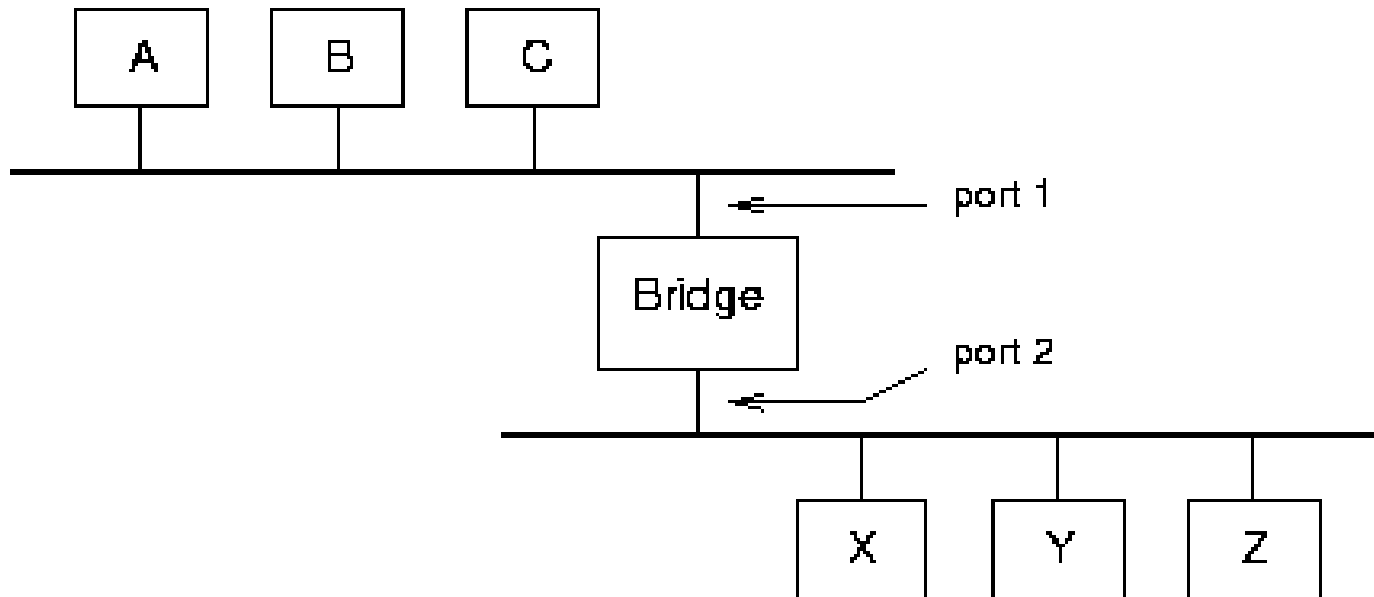
---

- connect 2 or more existing LANs
  - different organizations want to be connected
  - connect geographically separate LANs.
- split an existing LAN but stay connected
  - too many stations or traffic for one LAN
  - reduce collisions and increase efficiency
  - help restrict traffic to one LAN
- Support multiple protocols at MAC layer
- Cheaper than routers

# Bridge functioning

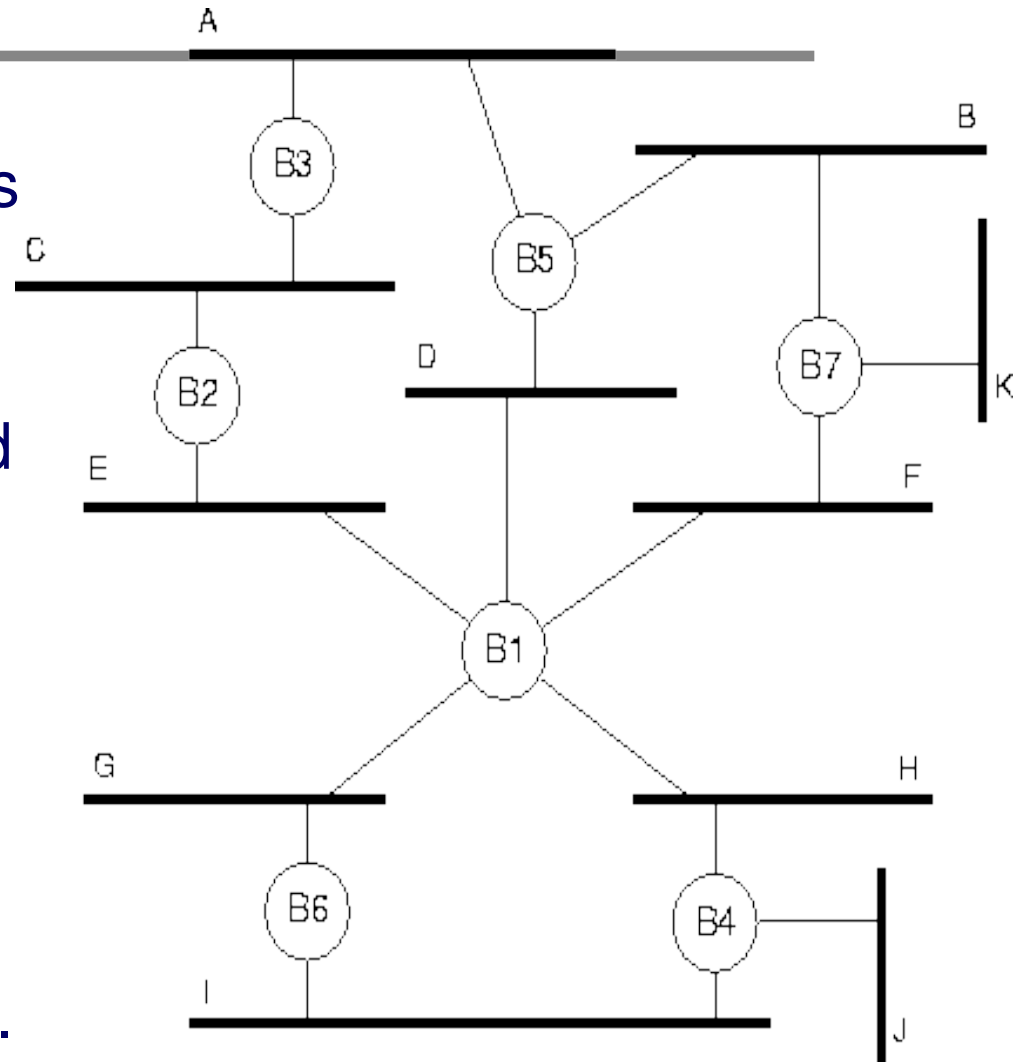
---

- Forwards to connected segments
- Learns MAC address to segment mapping
- Mapping table
- Maintains data in table till *timeout*



# Spanning tree algorithm

- Extended LANs may have loops due to parallel bridges
- Bridges run a distributed spanning tree algorithm.
- Each bridge has a unique id (e.g., B1, B2, B3).
- Select bridge with smallest id as root.
- Select bridge on each LAN that is closest to the root as that LAN's *designated bridge* (use id to break ties).



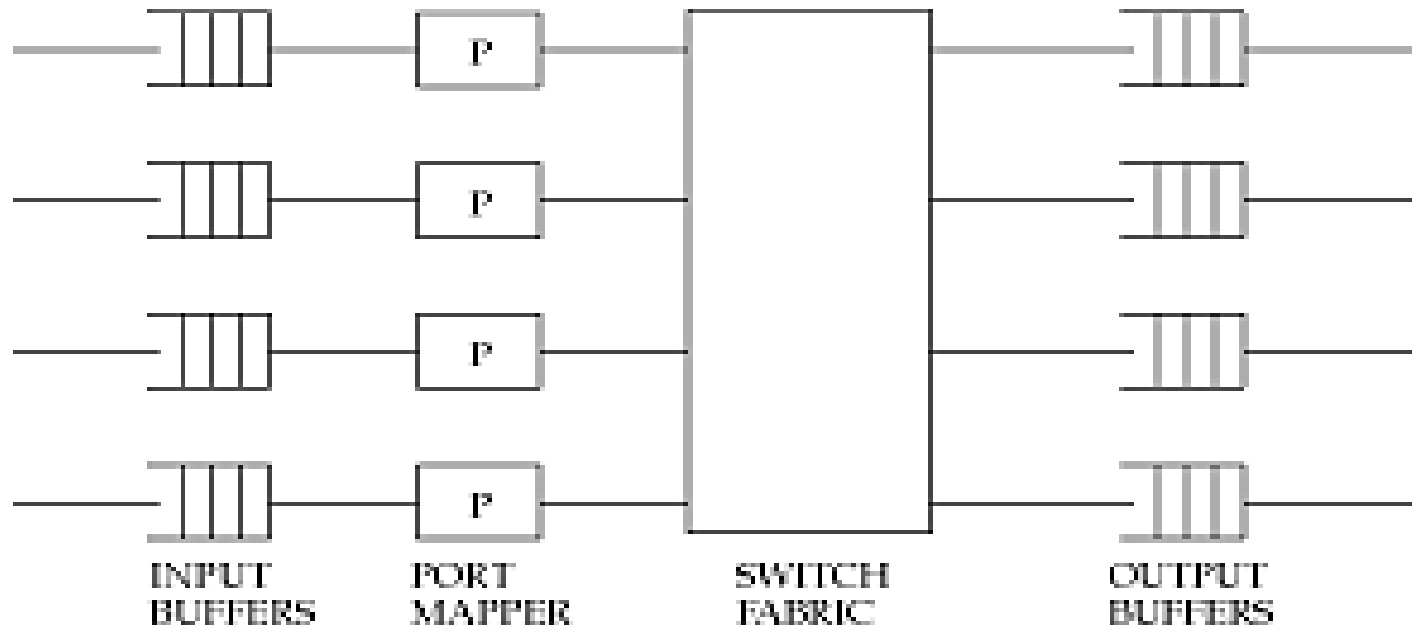
# Spanning tree protocol

---

- Bridges exchange configuration messages.
  - id for bridge sending the message.
  - id for what the sending bridge believes to be root bridge.
  - distance (hops) from sending bridge to root bridge.
- Each bridge records current best configuration message for each port.
- Initially, each bridge believes it is the root.
- When learn not root, stop generating configuration message.
- When learn not designated bridge, stop forwarding configuration messages.
- Root bridge continues to send configuration messages periodically.

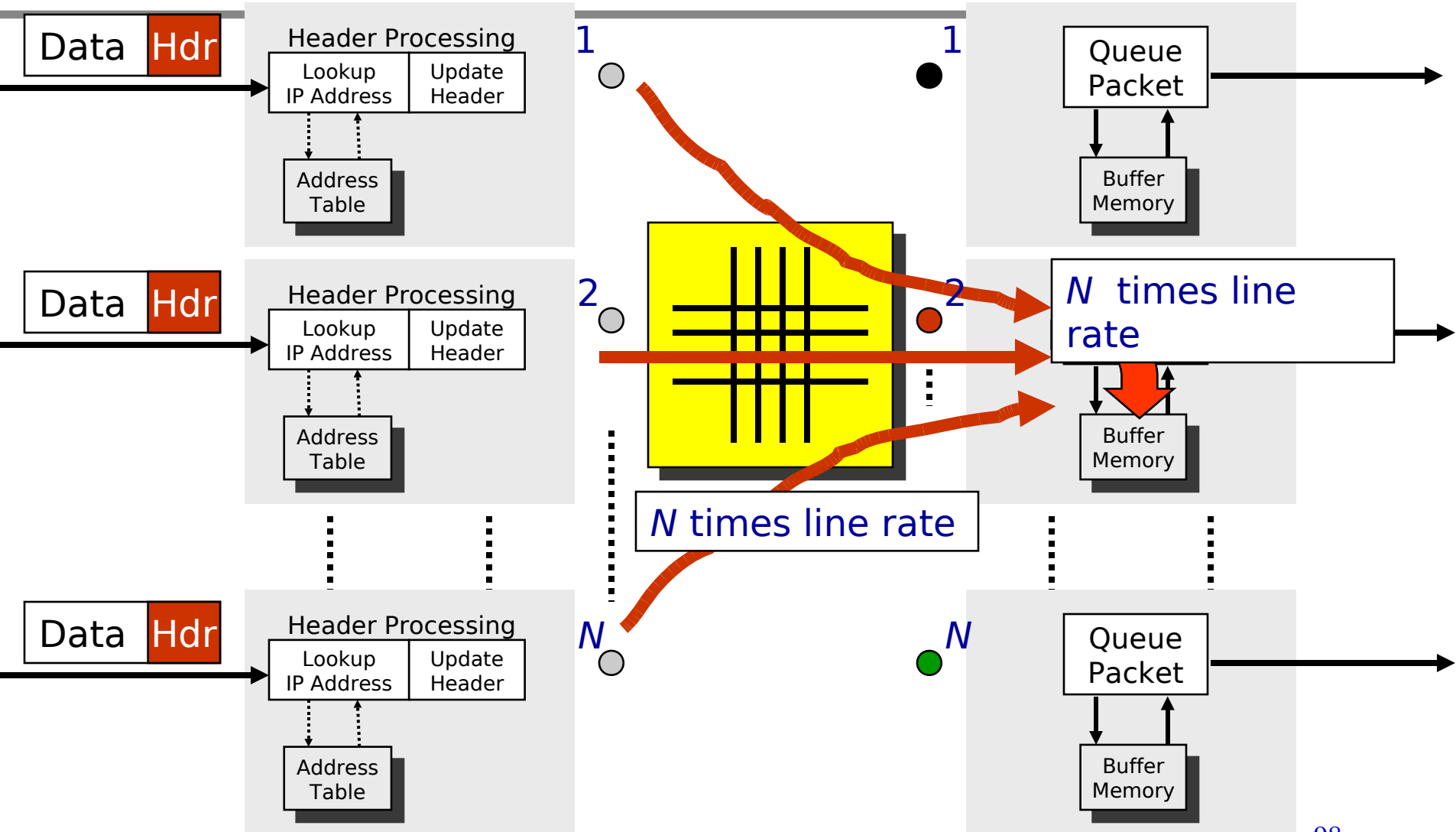


# Generic Switch



Latency: Time a switch takes to figure out where to forward a data unit

# Generic Router Architecture



# Blocking in packet switches

---

- Can have both internal and output blocking
  - Internal: no path to output
  - Output: link unavailable
- Unlike a circuit switch, cannot predict if packets will block
- If packet is blocked, must either buffer or drop it

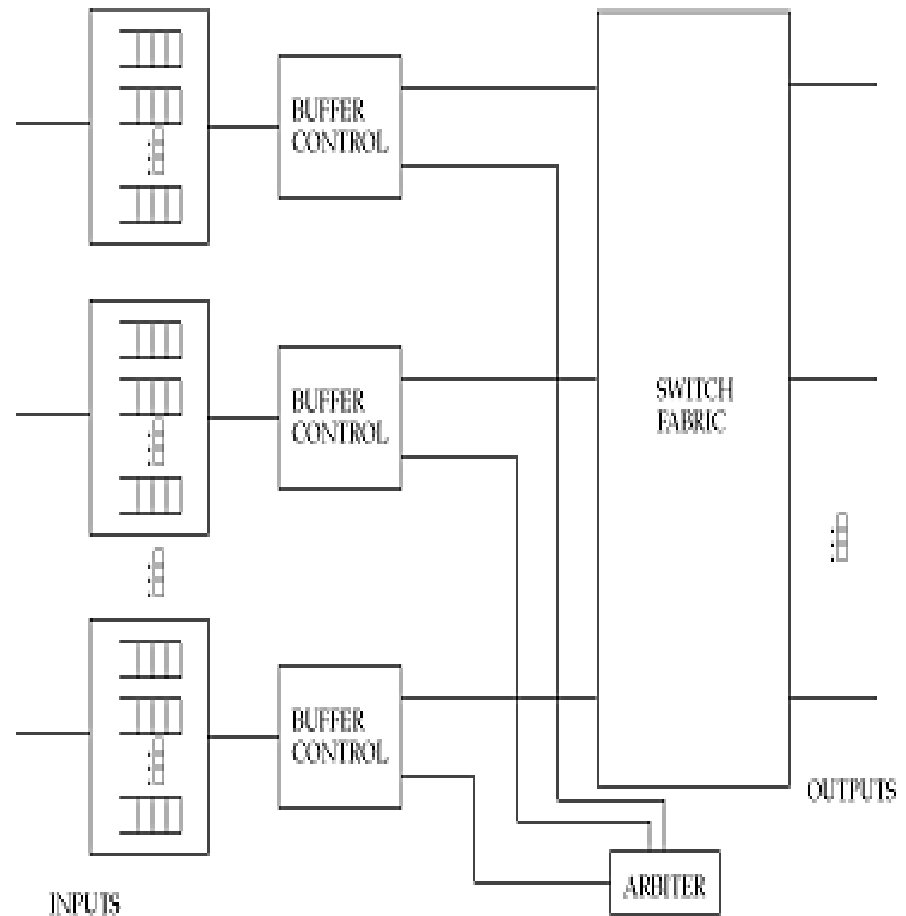
# Dealing with blocking

---

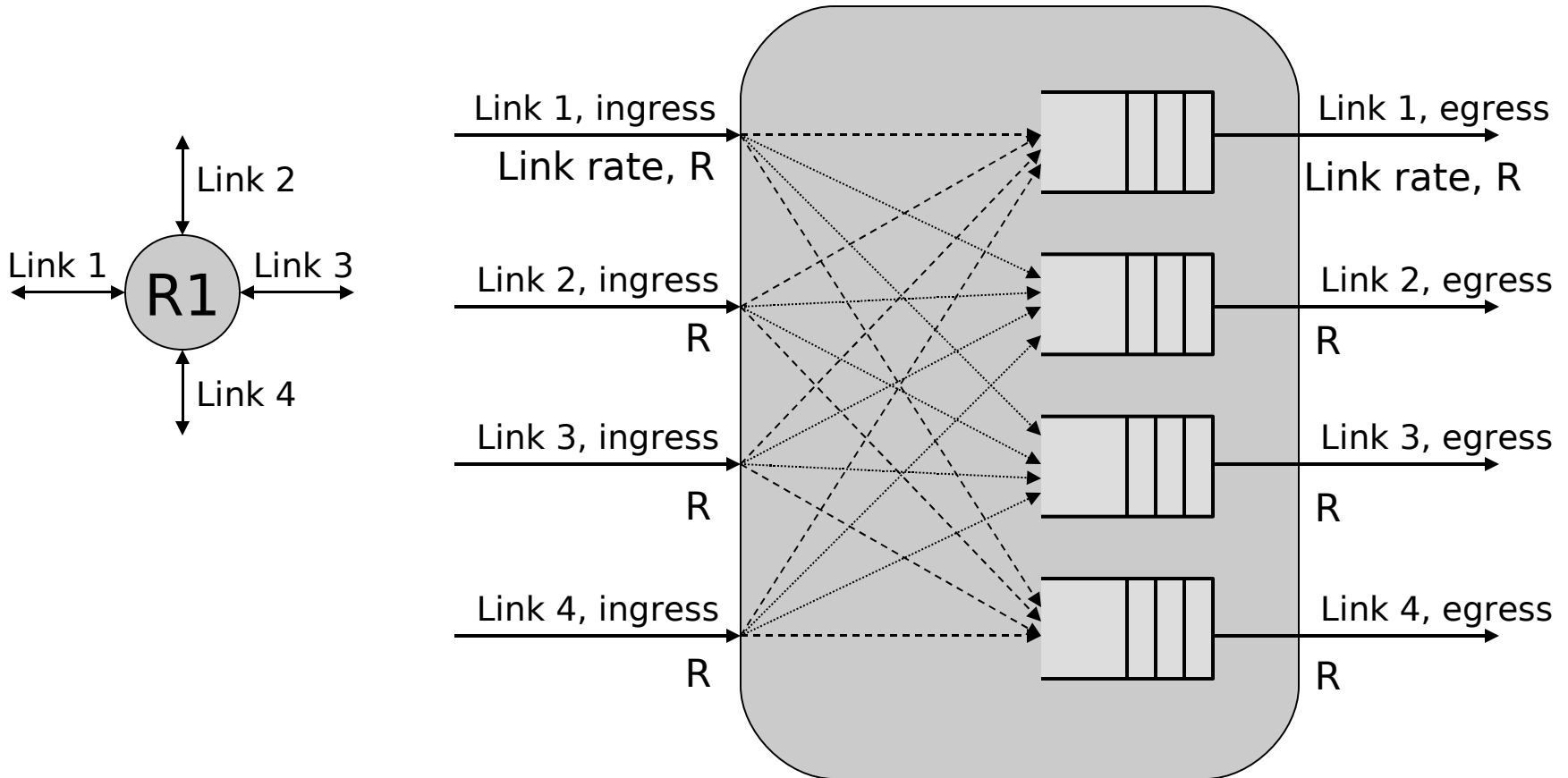
- Match input rate to service rate
  - Overprovisioning: internal links much faster than inputs
- Buffering:
  - » input port
  - » in the fabric
  - » output port

# Input buffering (input queueing)

- No speedup in buffers or trunks (unlike output queued switch)
- Needs arbiter
- Problem: *head of line blocking*



# Output queued switch

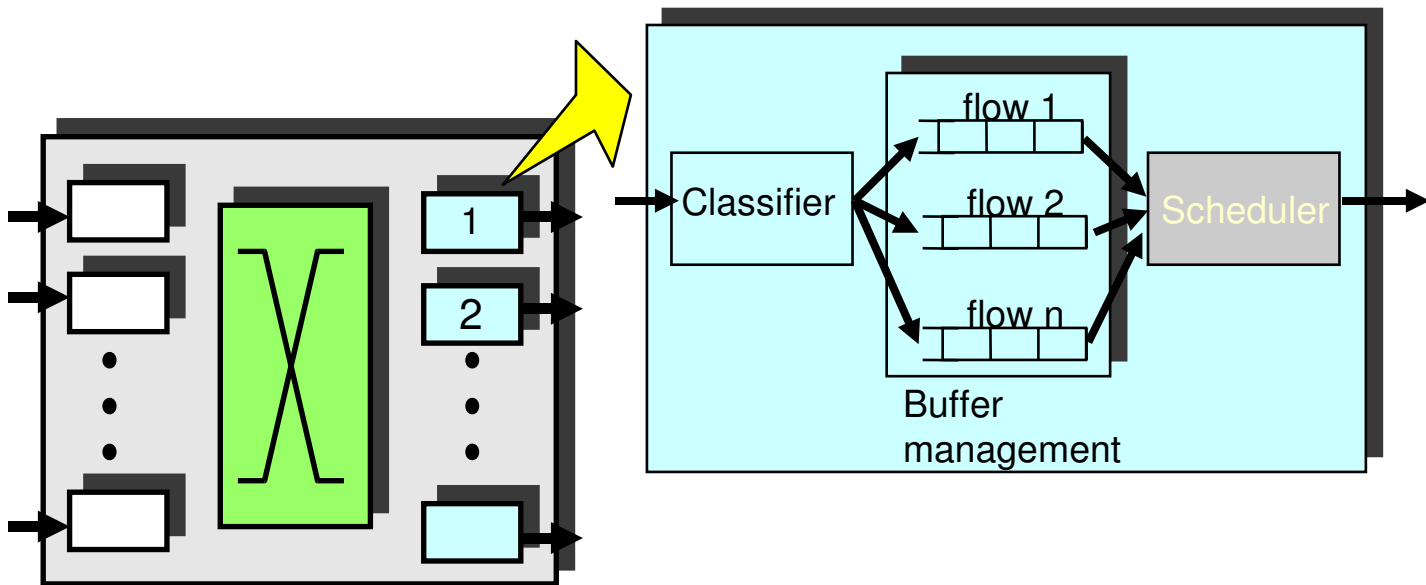


---

# Scheduling

# Packet scheduling

- Decide when and what packet to send on output link
  - Usually implemented at output interface

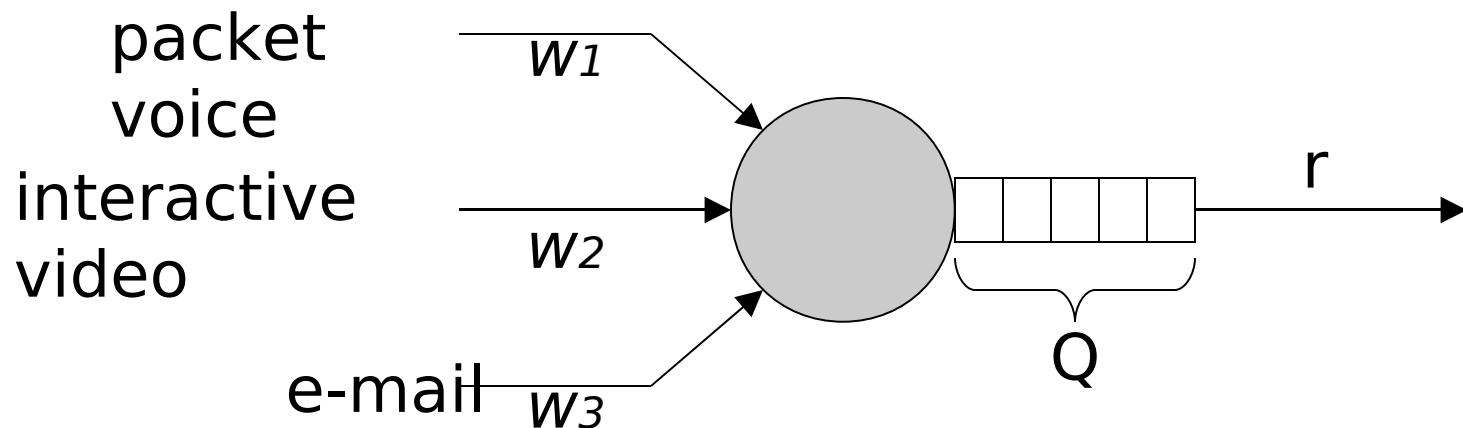




# Scheduling objectives

---

- Key to fairly sharing resources and providing performance guarantees.
- A scheduling discipline does two things:
  - decides service order.
  - manages queues of service requests.



# Scheduling disciplines

---

- Scheduling is used:
  - Wherever contention may occur
  - Usually studied at network layer, at output queues of switches
- Scheduling disciplines:
  - resolve contention
  - allocate bandwidth
  - Control delay, loss
  - determine the fairness of the network
  - give different qualities of service and performance guarantees

# Scheduling requirements

---

1. Easy to implement.
2. Min-Max Fairness.
3. Flexible with variable weights and packets length.
4. Provide performance bounds.
5. Allows easy admission control decisions.

# Problems with FIFO queues

---

Fairness

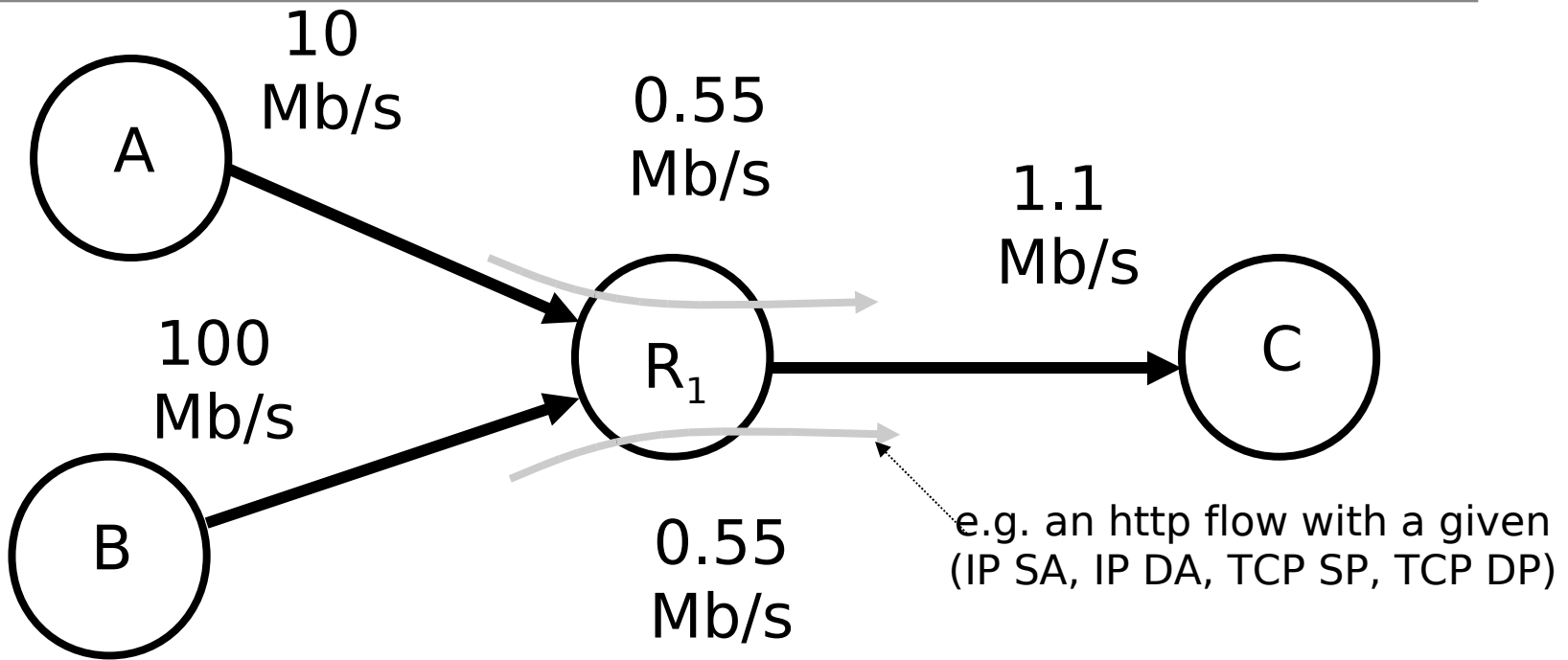
1. In order to maximize its chances of success, a source has an incentive to maximize the rate at which it transmits.
2. (Related to #1) When many flows pass through it, a FIFO queue is “unfair” – it favors the most greedy flow.

Delay  
Guarantees

3. It is hard to control the delay of packets through a network of FIFO queues.

# Fairness

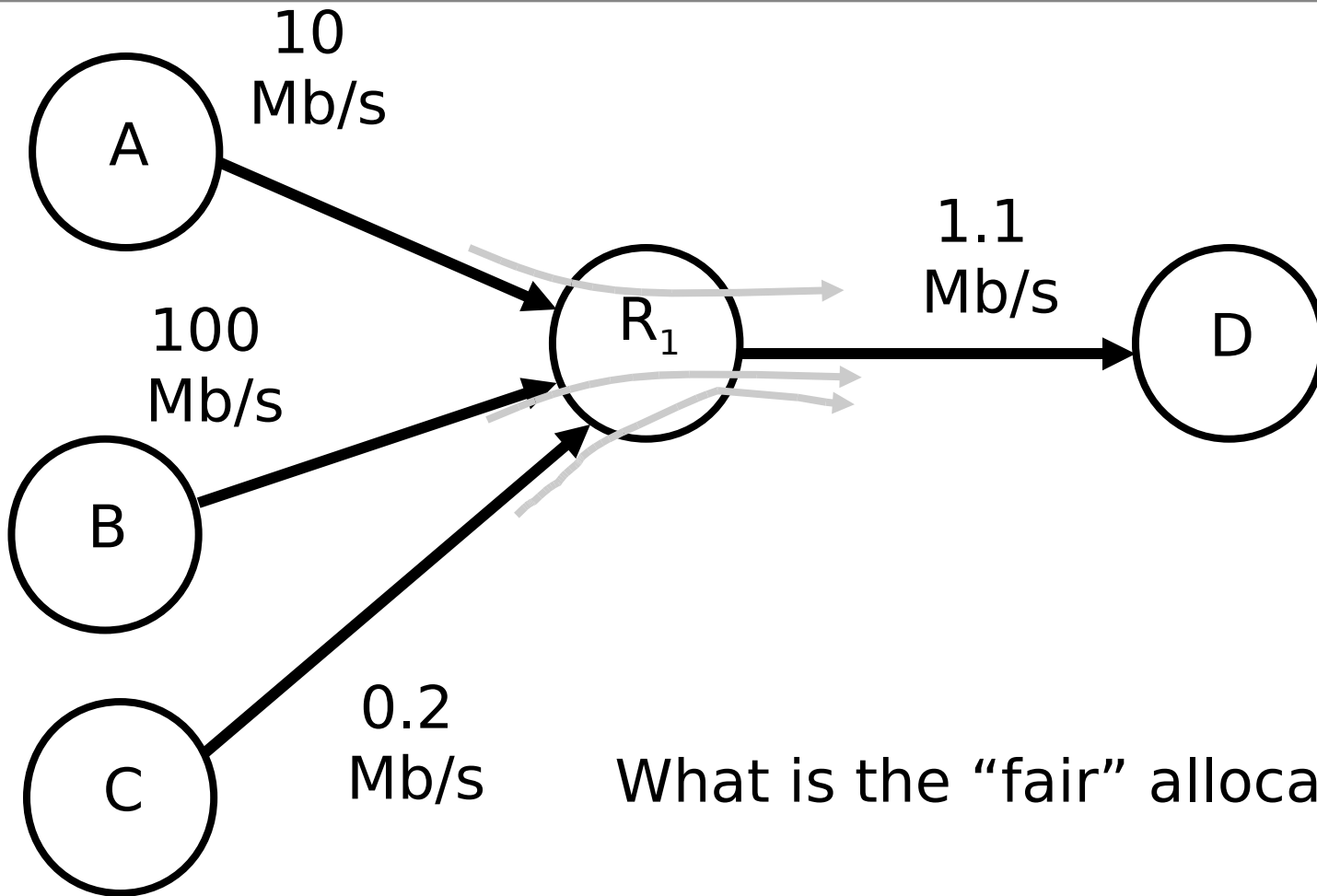
---



What is the “fair” allocation:  
(0.55Mb/s, 0.55Mb/s) or (0.1Mb/s, 1Mb/s)?

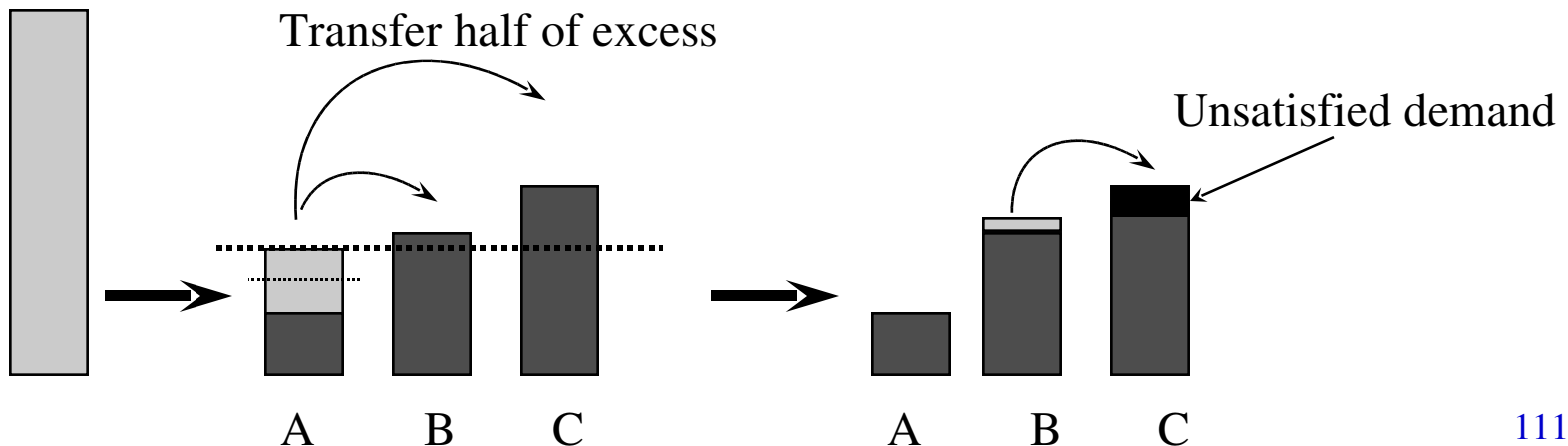
# Fairness

---



# Max-Min Fairness

- An allocation is fair if it satisfies *max-min fairness*
  - each connection gets no more than what it wants
  - the excess, if any, is equally shared



# Max-Min Fairness

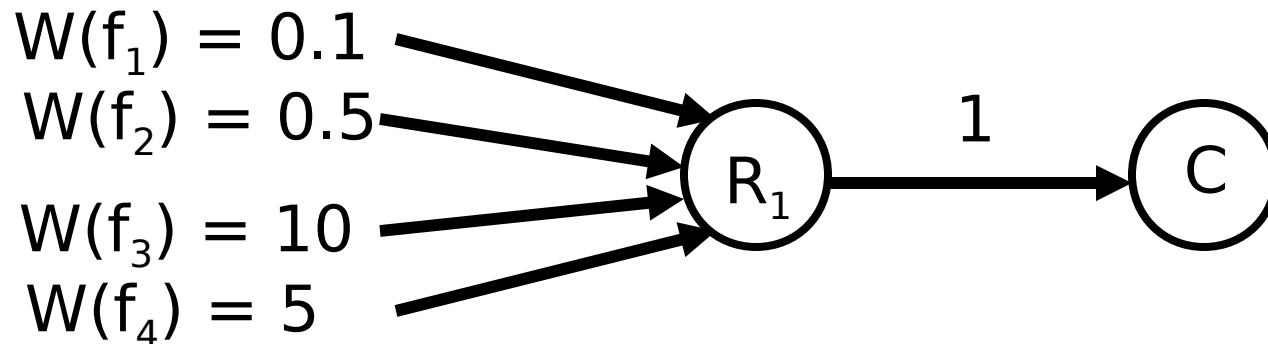
---

- $N$  flows share a link of rate  $C$ .
  - Flow  $f$  wishes to send at rate  $W(f)$ , and is allocated rate  $R(f)$ .
1. Pick the flow,  $f$ , with the smallest  $W(f)$ .
  2. If  $W(f) < C/N$ , then set  $R(f) = W(f)$ .
  3. If  $W(f) > C/N$ , then set  $R(f) = C/N$ .
  4. Set  $N = N - 1$ .  $C = C - R(f)$ .
  5. If  $N > 0$  goto 1.



# Max-Min Fairness: example

---



Round 1: Set  $R(f_1) = 0.1$

Round 2: Set  $R(f_2) = 0.9/3 = 0.3$

Round 3: Set  $R(f_4) = 0.6/2 = 0.3$

Round 4: Set  $R(f_3) = 0.3/1 = 0.3$

# Fair scheduling goals

---

- Max-Min fair allocation of resources among contending flows
- Protection (Isolate ill-behaved users)
  - Router does not send explicit feedback to source
  - Still needs e2e congestion control
- Work Conservation:
  - One flow can fill entire pipe if no contenders
  - Work conserving → scheduler never idles link if it has a packet

# Work conservation

---

- conservation law:  $\sum \rho_i q_i = \text{constant}$ ;
  - $\rho_i = \lambda_i x_i$ ;
  - $\lambda_i$  is traffic arrival rate
  - $x_i$  is mean service time for packet
  - $q_i$  is mean waiting time at the scheduler, for connection  $i$ ;
- sum of mean queueing delays received by a set of multiplexed connections, weighted by their share of the link, is independent of the scheduling discipline

# Round robin scheduling

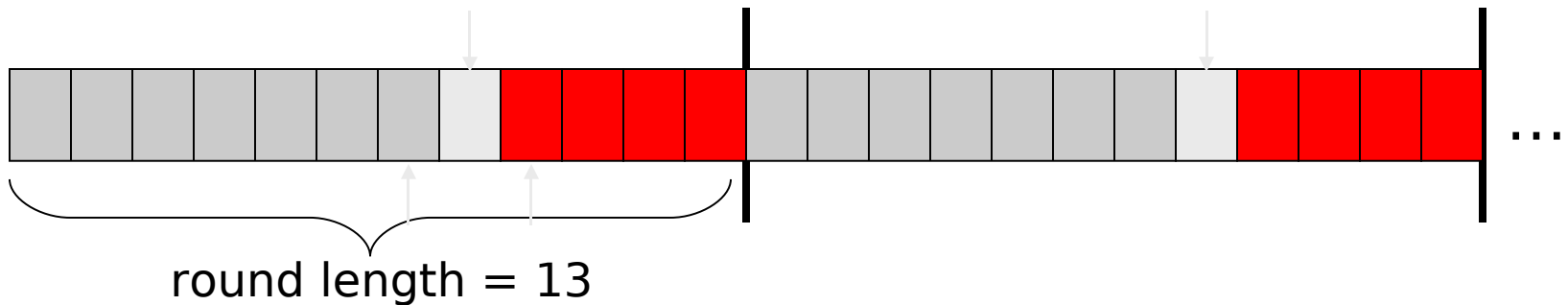
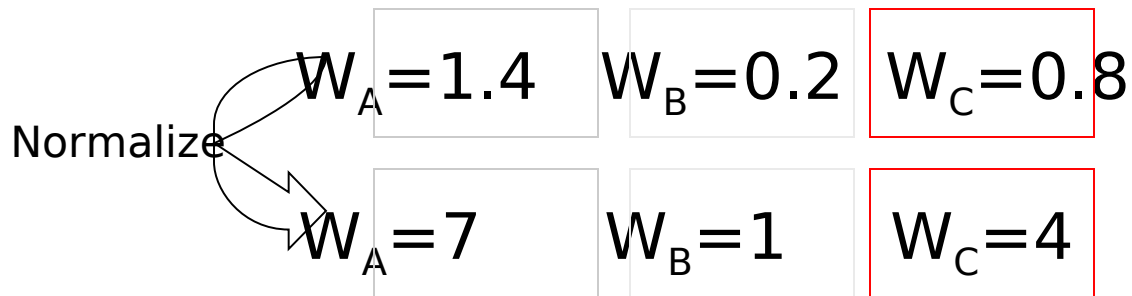
---

Scan class queues serving one from each class that has a non-empty queue

- Assumption: Fixed packet length
- Advantage:
  - Provides Min-Max fairness and Protection within contending flows
- Disadvantage:
  - More complex than FIFO: per flow queue/state
  - Unfair if packets are of different length or weights are not equal

# Weighted round robin

- Serve more than one packet per visit
- Number of packets are proportional to weights
- Normalize the weights so that they become integer



## Weighted RR - variable length packet

---

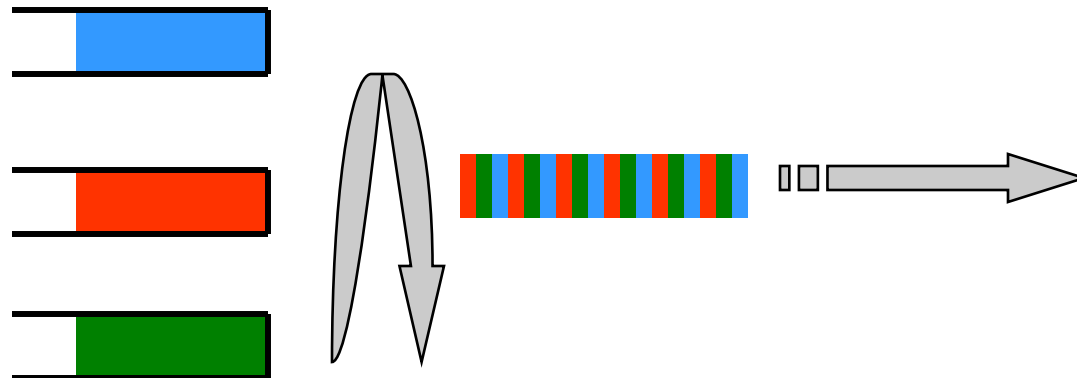
If different connection have different packet size, then

- WRR divides the weight of each connection with that connection's mean packet size and obtains a normalized set of weights
  - weights {0.5, 0.75, 1.0},
  - mean packet sizes {50, 500, 1500}
  - normalize weights:  $\{0.5/50, 0.75/500, 1.0/1500\} = \{0.01, 0.0015, 0.000666\}$ ,
  - normalize again {60, 9, 4}

# Generalized Processor Sharing (GPS)

---

- Main requirement is *fairness*
  - Visit each non-empty queue in turn
  - Serve infinitesimal from each
  - GPS is not implementable; we can serve only packets



# Weighted Fair Queueing (WFQ)

---

- Deals better with variable size packets and weights
- Also known as *packet-by-packet GPS* (PGPS)
- Find *finish time* of a packet, *had we been doing GPS*; serve packets in order of their finish times
- Uses round number and finish number



# WFQ details

---

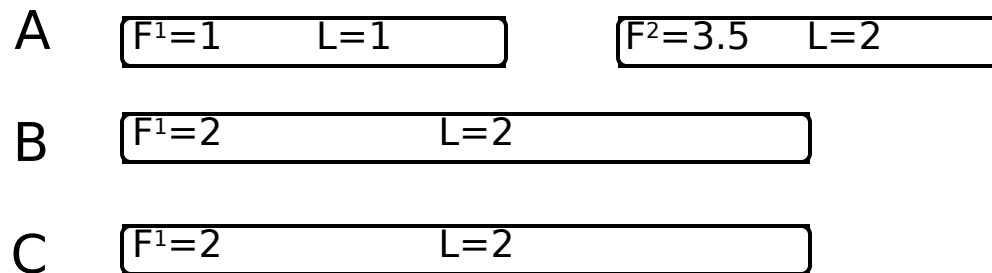
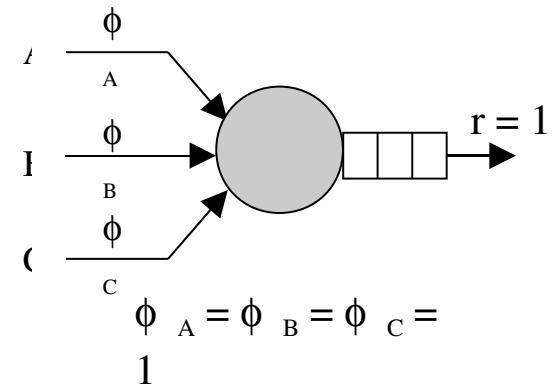
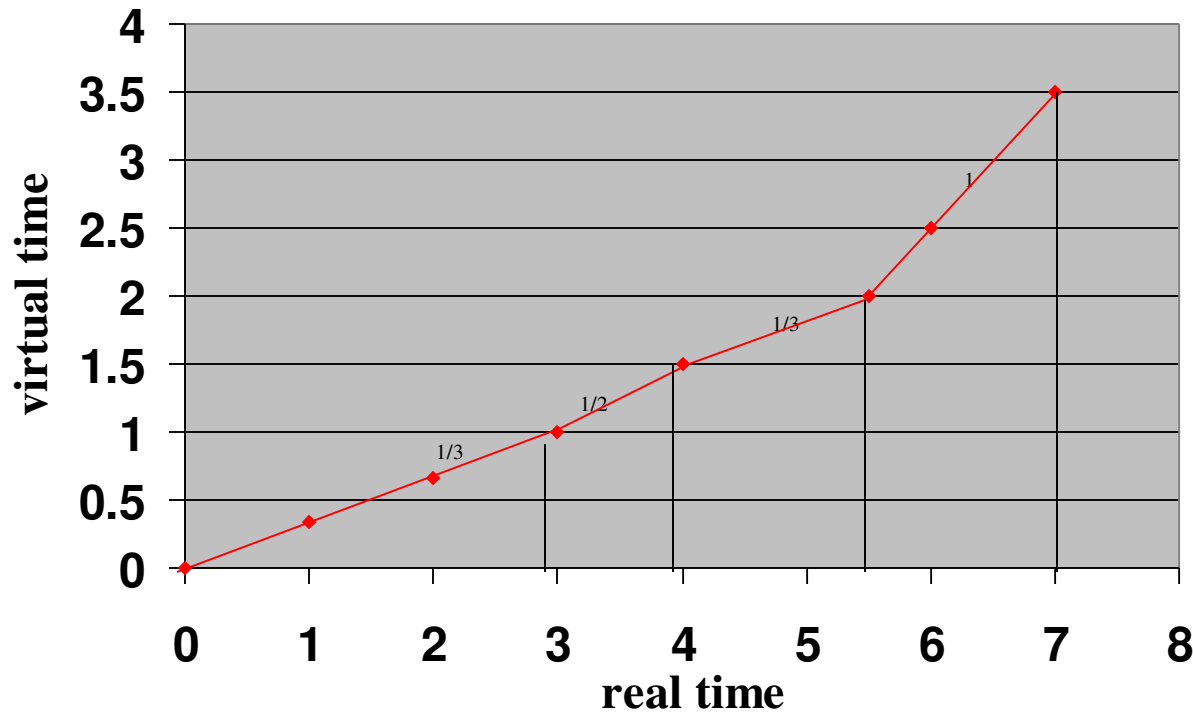
- Suppose, in each *round*, the server served one bit from each active connection
- *Round number* is the number of rounds already completed
  - can be fractional
- If a packet of length  $p$  arrives to an empty queue when the round number is  $R$ , it will complete service when the round number is  $R + p \Rightarrow$  *finish number* is  $R + p$ 
  - independent of the number of other connections!

# WFQ details

---

- If a packet arrives to a queue, and the previous packet has/had a finish number of  $f$ , then the packet's finish number is  $f+p$ 
  - Serve packets in order of finish numbers
- Finish time of a packet is not the same as the finish number

# WFQ Example



t=0: Packets of sizes 1,2,2 arrive at connections A, B, C.

t=4: Packet of size 2 arrives at connection A.

# Example (contd.)

---

- At time 0, slope of  $1/3$ ,
  - Finish number of A = 1, Finish number of B, C = 2
- At time 3,
  - connection A become inactive, slope becomes  $1/2$
- At time 4,
  - second packet at A gets finish number  $2 + 1.5 = 3.5$ ,  
Slope decreases to  $1/3$
- At time 5.5,
  - round number becomes 2 and connection B and C become inactive, Slope becomes 1
- At time 7,
  - round number becomes 3.5 and A becomes inactive.

# Guaranteed-service scheduling

---

- Delay-Earliest Due Date:
  - packet with earliest deadline selected
  - Delay-EDD prescribes how to assign deadlines
  - Source is required to send slower than its *peak rate*
  - Bandwidth at scheduler reserved at peak rate
  - Deadline = expected arrival time + delay bound
  - Delay bound is *independent* of bandwidth requirement
  - Implementation requires per-connection state and a priority queue

# Non work-conserving scheduling

---

- Non work conserving discipline may be idle even when packets await service
  - main idea: delay packet till *eligible*
  - Reduces delay-jitter => fewer buffers in network
  - Choosing eligibility time:
    - » rate-jitter regulator: bounds maximum outgoing rate
    - » delay-jitter regulator: compensates for variable delay at previous hop
  - Always punishes a misbehaving source
  - Increases mean delay; Wastes bandwidth

# Congestion control

---

- Congestion:
  - Performance degradation due to too many packets present in the subnet
- Causes:
  - Packets from several input lines needing the same output line
  - Bursty traffic, slow processors
  - Insufficient bandwidth/buffering

# Congestion control strategies

---

- Allocate resources in advance
- Packet discarding
  - aggregation: classify packets into classes and drop packet from class with longest queue
  - priorities: drop lower priority packets
- Choke the input
- Flow control at higher layers



# Early random drop

---

- Early drop => drop even if space is available
  - drop arriving packet with fixed drop probability if queue length exceeds threshold
  - signals endpoints to reduce rate
  - cooperative sources get lower overall delays,
  - uncooperative sources get severe packet loss

# Random early detection (RED)

---

- Metric is moving average of queue lengths
- Packet drop probability is a function of mean queue length
- Can mark packets instead of dropping them
- RED improves performance of a network of cooperating TCP sources
  - small bursts pass through unharmed
  - prevents severe reaction to mild overload

# Drop position

---

- Can drop a packet from head, tail, or random position in the queue
- Tail: easy; default approach
- Head: harder; lets source detect loss earlier
- Random: hardest; if no aggregation, hurts uncooperating sources the most

---

# IP Addressing

# Addressing

---

- Addresses need to be globally unique, so they are hierarchical
- Another reason for hierarchy: *aggregation*
  - reduces size of routing tables
  - at the expense of longer routes

# IP addressing

---

- Internet Protocol (IP)
  - Provides connectionless packet delivery and “best-effort” quality of service
  - No assurance that the packet will reach intended destination
- Every host interface has its own IP address
- Routers have multiple interfaces, each with its own IP address

# IPv4 addresses

---

- Logical address at network layer
- 32 bit address space
  - Network number, Host number
  - boundary identified with a *subnet* mask
  - can aggregate addresses within subnets
- Machines on the same "network" have same network number
- One address per interface

# Address classes

---

- Class A addresses - 8 bits network number
- Class B addresses - 16 bits network number
- Class C addresses - 24 bits network number
- Distinguished by leading bits of address
  - leading 0 => class A (first byte < 128)
  - leading 10 => class B (first byte in the range 128-191)
  - leading 110 => class C (first byte in the range 192-223)



# IP address notation

---

- Dotted decimal notation
  - 144.16.111.2 (Class B)
  - 202.54.44.120 (Class C)
  - Special Conventions
    - » All 0s -- this host
    - » All 1s -- limited broadcast (localnet)

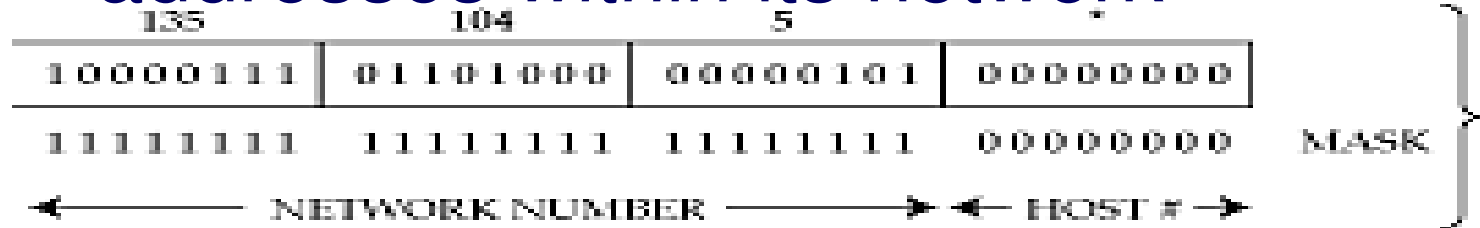
# IP address issues

---

- Inefficient: wasted addresses
- Inflexible: fixed interpretation
- Not scalable: Not enough network numbers
- IP addressing schemes
  - Sub-netting: Create sub networks within an address space
  - CIDR: Variable interpretations for the network number
  - Ipv6: 128 bit address space

# Subnetting

- Allows administrator to cluster IP addresses *within* its network



INCLUDES

135	104	5	1
10000111	01101000	00000101	00000001

135	104	5	6
1000 0111	0110 1000	0000 0101	0000 0110

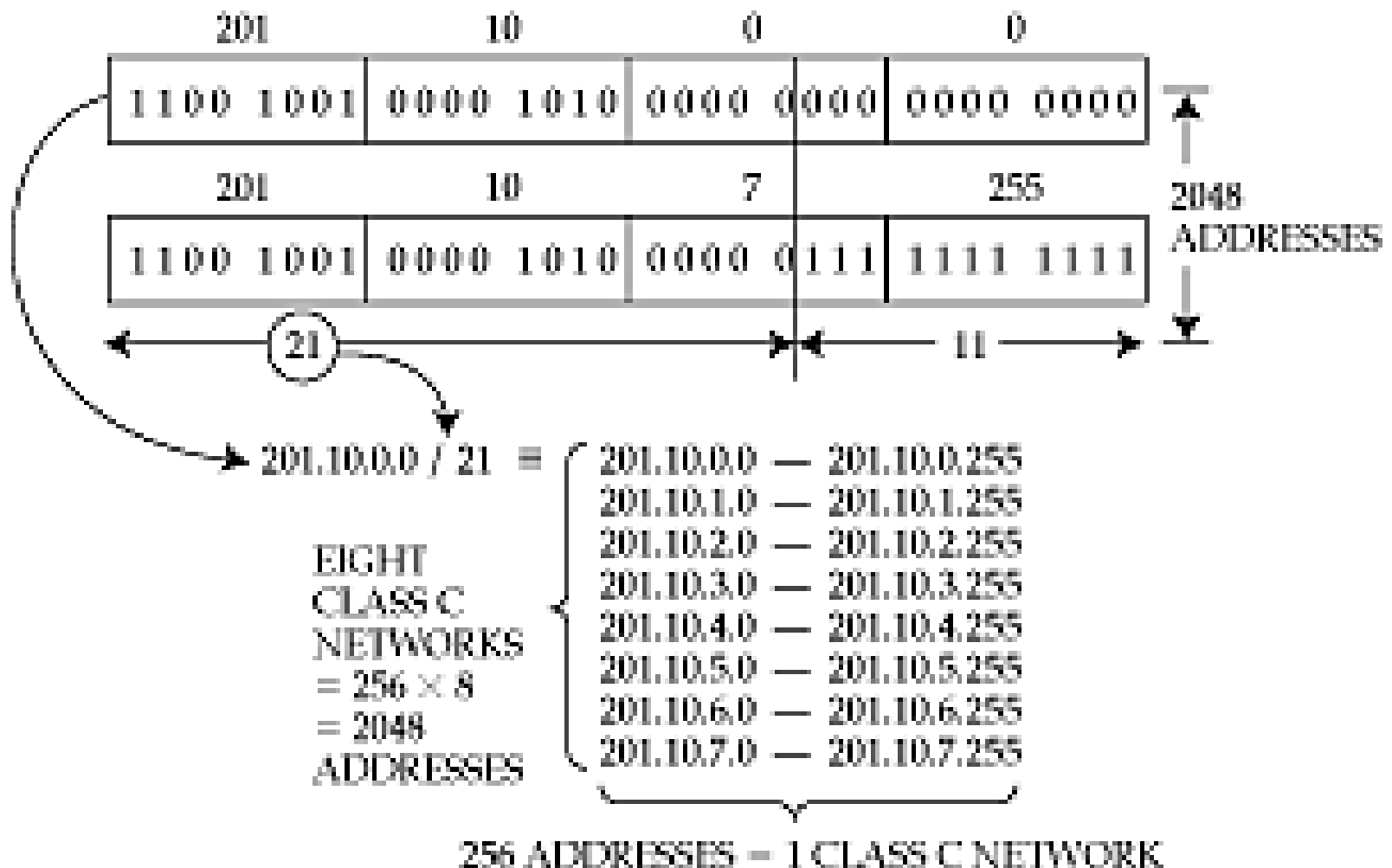
135	104	5	24
1000 0111	0110 1000	0000 0101	0001 1000

# Classless Inter Domain Routing (CIDR)

---

- Scheme forced medium sized nets to choose class B addresses, which wasted space
  - allow ways to represent a set of class C addresses as a block, so that class C space can be used
  - use a CIDR mask
  - idea is very similar to subnet masks, except that all routers must agree to use it

# CIDR (contd.)

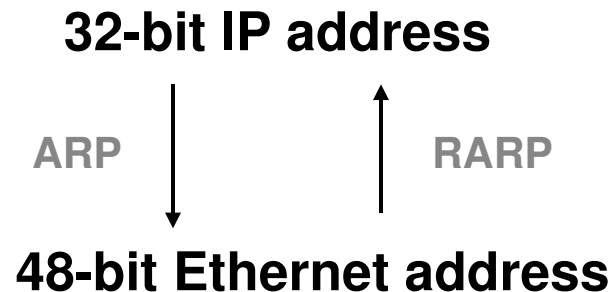


# Address Resolution Protocol (ARP)

## RFC 1010

---

- Address resolution provides mapping between IP addresses and datalink layer addresses



- point-to-point links don't use ARP, have to be configured manually with addresses

# ARP

---

- ARP requests are broadcasts
  - “Who owns IP address x.x.x.x.?”.
- ARP reply is unicast
- ARP cache is created and updated dynamically
  - arp -a displays entries in cache
- Every machine broadcasts its mapping when it boots

# RARP and Proxy ARP

---

- RARP: used by diskless workstations when booting.
  - Query answered by RARP server
- Proxy ARP: router responds to an ARP request on one of its networks for a host on another of its networks.
  - Router acts as proxy agent for the destination host. Fools sender of ARP request into thinking router is destination



# ICMP (Internet Control Message)

---

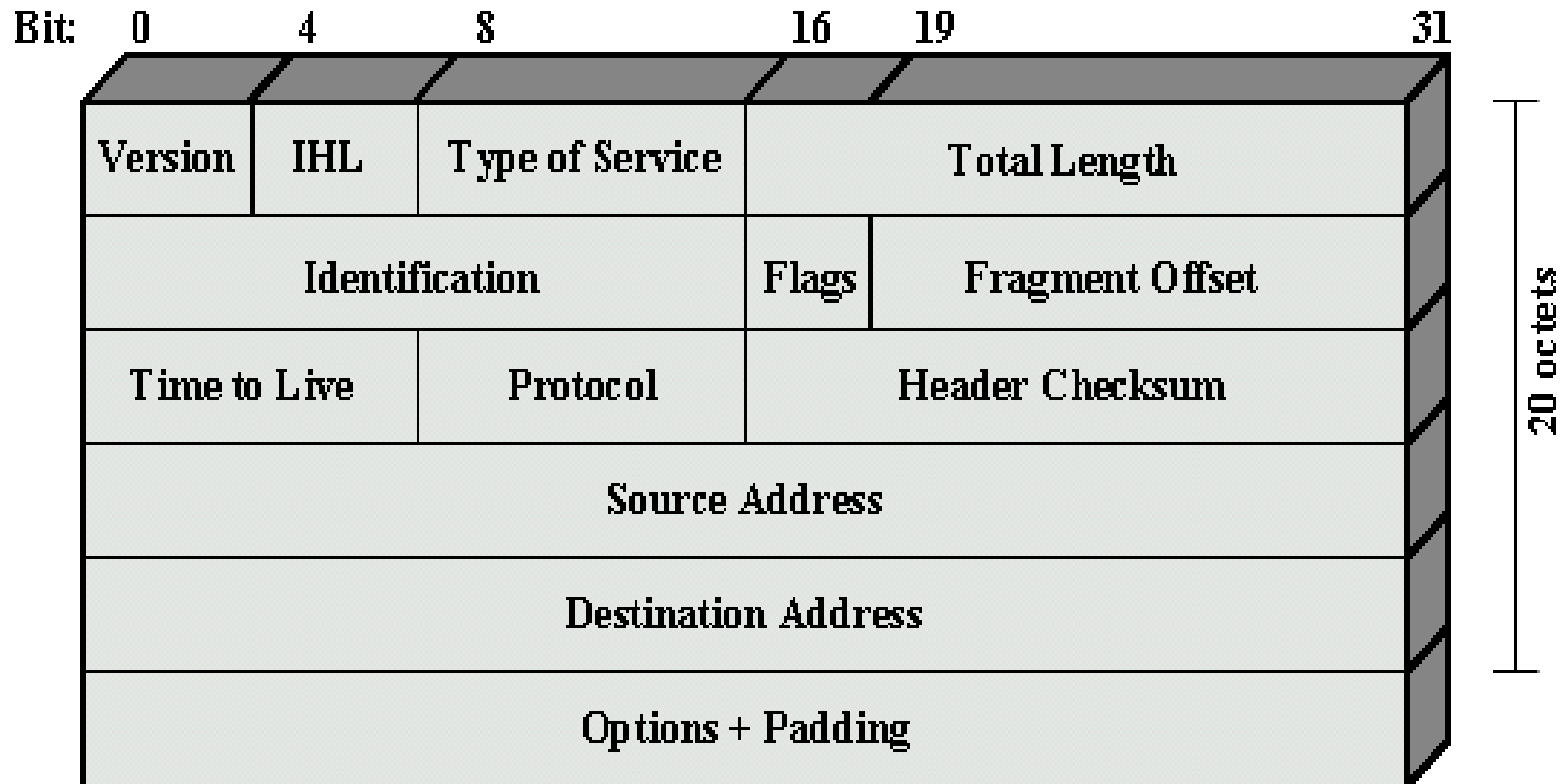
- Unexpected events are reported to the source by routers, using ICMP
- ICMP messages are of two types: query, error
- ICMP messages are transmitted within IP datagrams (layered above IP)
- ICMP messages, if lost, are not retransmitted

# Example ICMP messages

---

- “destination unreachable” (type 3)
  - can’t find destination network or protocol
- “time exceeded” (type 11)
  - expired lifetime (TTL reaches 0): symptom of loops or congestion . . .
- redirect
  - advice sending host of a better route
- echo request, echo-reply (query)
  - testing if destination is reachable and alive
- timestamp request, timestamp-reply
  - sampling delay characteristics

# IP header



# IP header

---

- Source and Destination IP addresses of 4 bytes each
- Version number: IPv4, next IPv6
- IHL: header length, can be max. 60 bytes.
- 20 byte fixed part and a variable length optional part
- Total length: max. 65 535 bytes presently (header + data)

# IP header

---

- Type of Service (ToS): to be used for providing quality of service
  - Low delay, high throughput, high reliability, low monetary costs are ToS metrics
- TTL: Time to Live, reduced by one at each router. Prevents indefinite looping.
- Checksum: over header, NOT data.
  - Implemented in software

# IP header

---

- Protocol: 1=ICMP, 6=TCP, 17=UDP
  - RFC 1700 for numbers of well known protocols
  - could also be IP itself, for encapsulation
- Identification, 3-bit flags and fragment offset (4 bytes) fields used for fragmentation and reassembly of packets
  - DF: “Don’t fragment” bit.
  - MF: More fragments bit.

---

# IP Routing

# IP forwarding

---

- At a Host:
  - Destination on my net?
  - If yes, use ARP and deliver directly
  - If not, give to default gateway
- At a Gateway:
  - Am I the destination IP?
  - If yes, deliver packet to higher layer
  - If not, which interface to forward on?
  - consult Routing Tables to decide



# Building routing tables

---

- Computed by routing protocols
  - Routing Information Protocol (RIP) [RFC 1058]
  - Open Shortest Path First (OSPF) [RFC 1131]
  - Border Gateway Protocol (BGP) [RFC 1105]
- Routing table contains the following information
  - destination IP address (host or network)
  - IP address of next Hop router
  - flags: which interface etc.

# Routing protocol issues

---

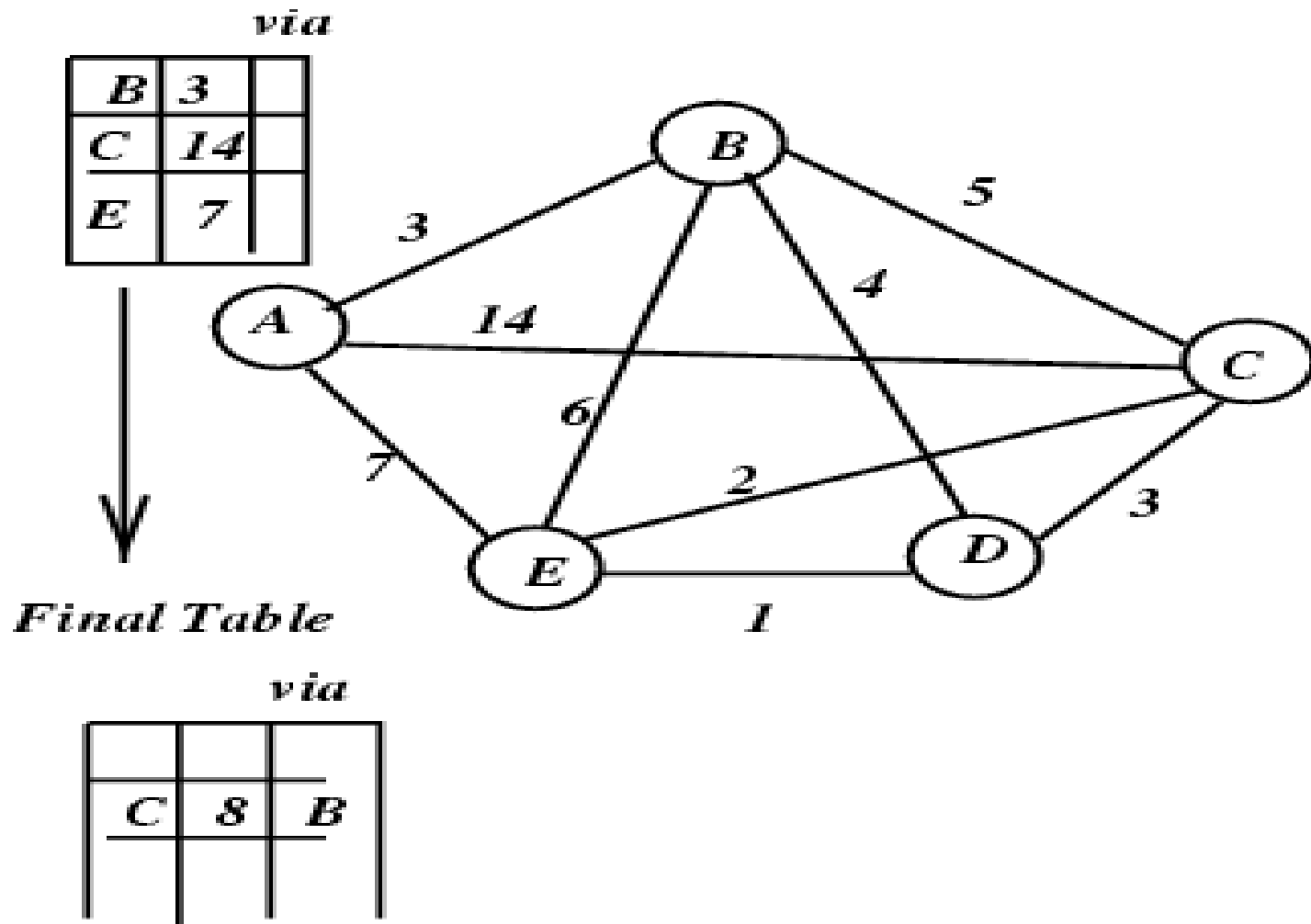
- **Simplicity and Performance:**
  - Size of the routing table should be kept small
  - Minimize number of control messages exchanged
- **Correctness and Robustness:**
  - Packet should be eventually delivered
  - Cope with changes in the topology and failures
  - No formation of routing loops or frequent toggling of routes

# Classification of routing protocols

---

- distance vector vs. link state
  - Both assume router knows
    - » address of each neighbor
    - » cost of reaching each neighbor
  - Both allow a router to determine global routing information by talking to its neighbors
- interior vs. exterior
  - Hierarchically reduce routing information

# DV Example: RIP



# DV problem: count to infinity

---

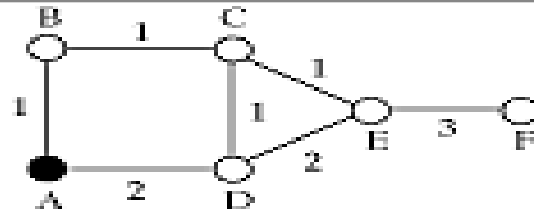
- Path vector
  - DV carries path to reach each destination
- Split horizon
  - never tell neighbor cost to X if neighbor is next hop to X
- Triggered updates
  - exchange routes on change, instead of on timer
  - faster count up to infinity

# Link state routing

---

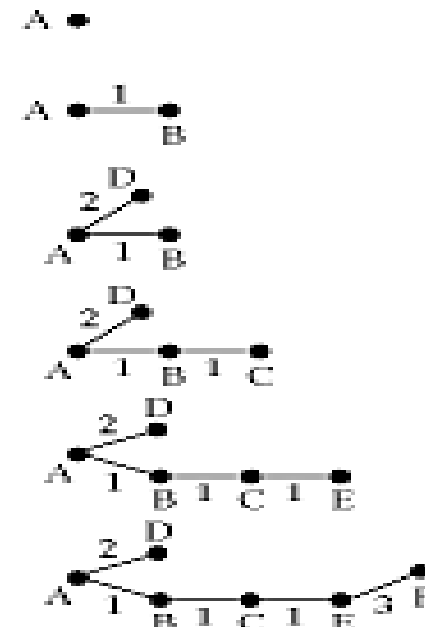
- A router describes its neighbors with a *link state packet (LSP)*
- Use *controlled flooding* to distribute this everywhere
  - store an LSP in an *LSP database*
  - if new, forward to every interface other than incoming one
- Sequence numbers in LSP headers
  - Greater sequence number is newer
  - Wrap around/purging: aging

# LS Example: OSPF



B(A,1) means B was reached by A, cost 1

PERMANENT	TEMPORARY	COMMENTS
A	B(A,1), D(A,2)	ROOT AND ITS NEIGHBORS
A, B(A,1)	D(A,2), C(B,2)	ADD C(B,2)
A, B(A,1) D(A,2)	E(D,4), C(B,2)	C(D,3) DIDN'T MAKE IT
A, B(A,1) D(A,2), C(B,2)	E(C,3)	E(D,4) TOO LONG
A, B(A,1) D(A,2), C(B,2) E(C,3)	F(E,6)	
A, B(A,1) C(B,2), D(A,2) E(C,3), F(E,6)	NULL	STOP



# RIP

---

- Distance vector
  - Cost metric is hop count
  - Infinity = 16
- RIPv1 defined in RFC 1058
  - uses UDP at port 520
- trigger for sending of distance vectors
  - 30-second intervals
  - routing table update
  - split horizon
- RIPv2 defined in RFC 1388
  - uses IP multicasting (224.0.0.9)



# OSPF

---

- Successor to RIP which used Link-State
- Using raw IP and IP multicasting
- LSP updates are acknowledged
- Complex
  - LSP databases to be protected
- Implementation: gated

# Exterior routing protocols

---

- Large networks need large routing tables
  - more computation to find shortest paths
  - more bandwidth wasted on exchanging DVs and LSPs
- Hierarchical routing
  - divide network into a set of domains
  - gateways connect domains
  - computers within domain unaware of outsiders
  - gateways know only about other gateways

# External and summary records

---

- If a domain has multiple gateways
  - *external* records tell hosts in a domain which one to pick to reach a host in an external domain
  - *summary* records tell backbone which gateway to use to reach an internal node
- External and summary records contain distance from gateway to external or internal node

# Border Gateway Protocol (BGP)

---

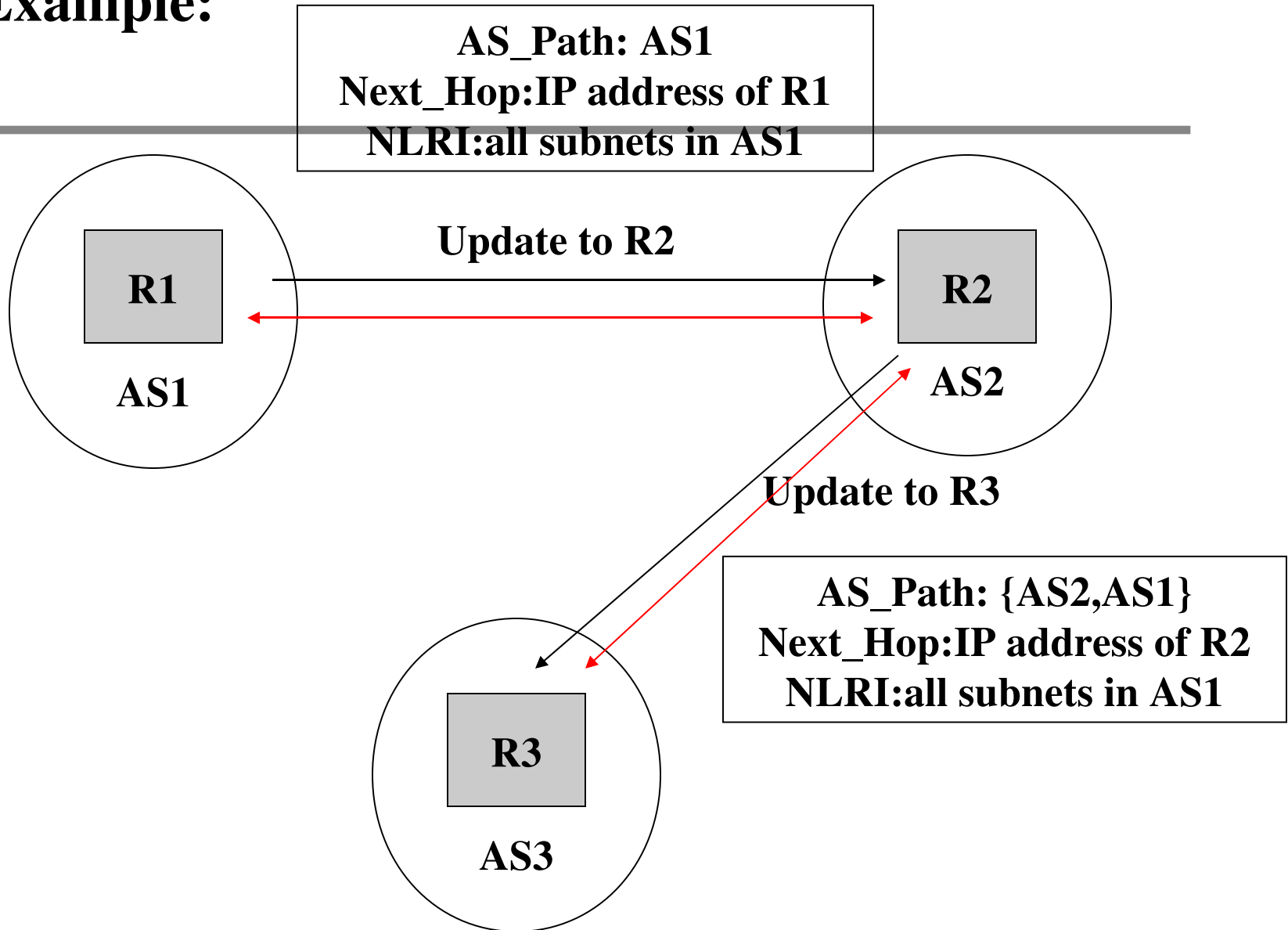
- Internet exterior protocol
- Path-vector
  - distance vector annotated with entire path
  - also with policy attributes
  - guaranteed loop-free
- Uses TCP to disseminate DVs
  - reliable
  - but subject to TCP flow control

# Functions of BGP

---

- Neighbor acquisition
  - open and keep-alive messages
- Neighbor reachability
  - keep alive and update messages
- Network reachability
  - Database of reachable internal subnets
  - Sends updates whenever this info changes
  - notification messages
  - NLRI: network layer reachability information

# Example:



# IP routing mechanism

---

- Steps for searching of routing table
  - search for a matching host address
  - search for a matching network address
  - search for a default entry
- a matching host address is always used before a matching network address (longest match)
- if none of above steps works, then packet is undeliverable

# Some IP networking tools

---

- netstat: info. about network interfaces
- ifconfig: configure/query a network interface
- ping: test if a particular host is reachable
- traceroute: obtain list of routers between source and destination
- tcpdump/sniffit: capture and inspect packets from network
- nslookup: address lookup
- arp: display/manage ARP cache



---

# End-to-End Transport

# User Datagram Protocol (UDP)

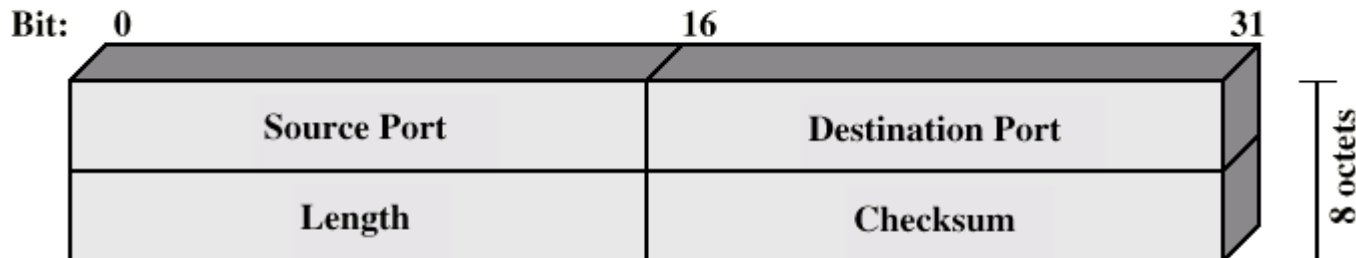
---

- Datagram oriented [RFC 768]
- Doesn't guarantee any reliability
- Useful for Applications such as voice and video, where
  - retransmission should be avoided
  - the loss of a few packets does not greatly affect performance
- each application “write” produces one UDP datagram, which causes one IP datagram to be sent

# UDP header

---

- Length of header and data in bytes.
- Checksum covers header and data.
  - Checksum uses a 12 byte pseudo-header containing some fields from the IP header
  - includes IP address of source and destination, protocol and segment length



# Transmission Control Protocol (TCP)

---

- Guaranteed service protocol [RFC 793]
  - ensures that a packet has been received by the destination by using acknowledgements and retransmission
- Connection oriented
  - applications need to establish a TCP connection prior to transfer
  - 3-way handshake

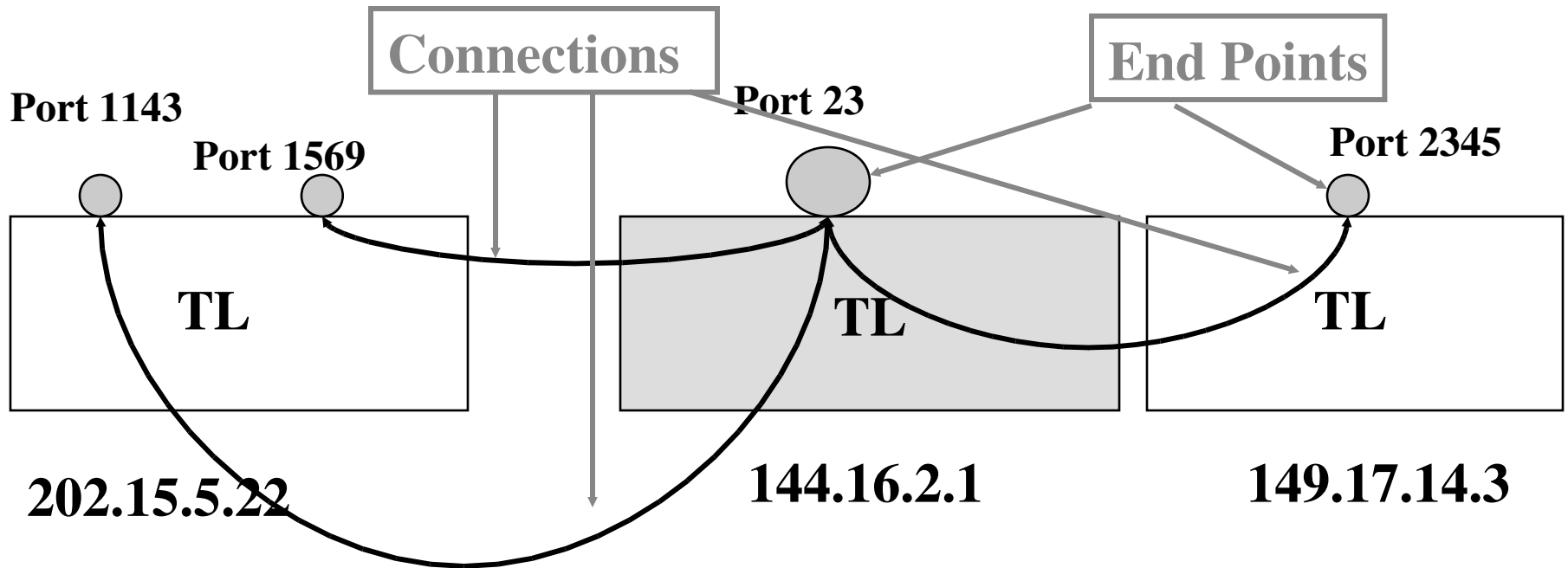
# More TCP features

---

- Full duplex
  - Both ends can simultaneously read and write
- Byte stream
  - Ignores message boundaries
- Flow and congestion control
  - Source uses feedback to adjust transmission rate

# Ports, Connections, End-points, Sockets

---



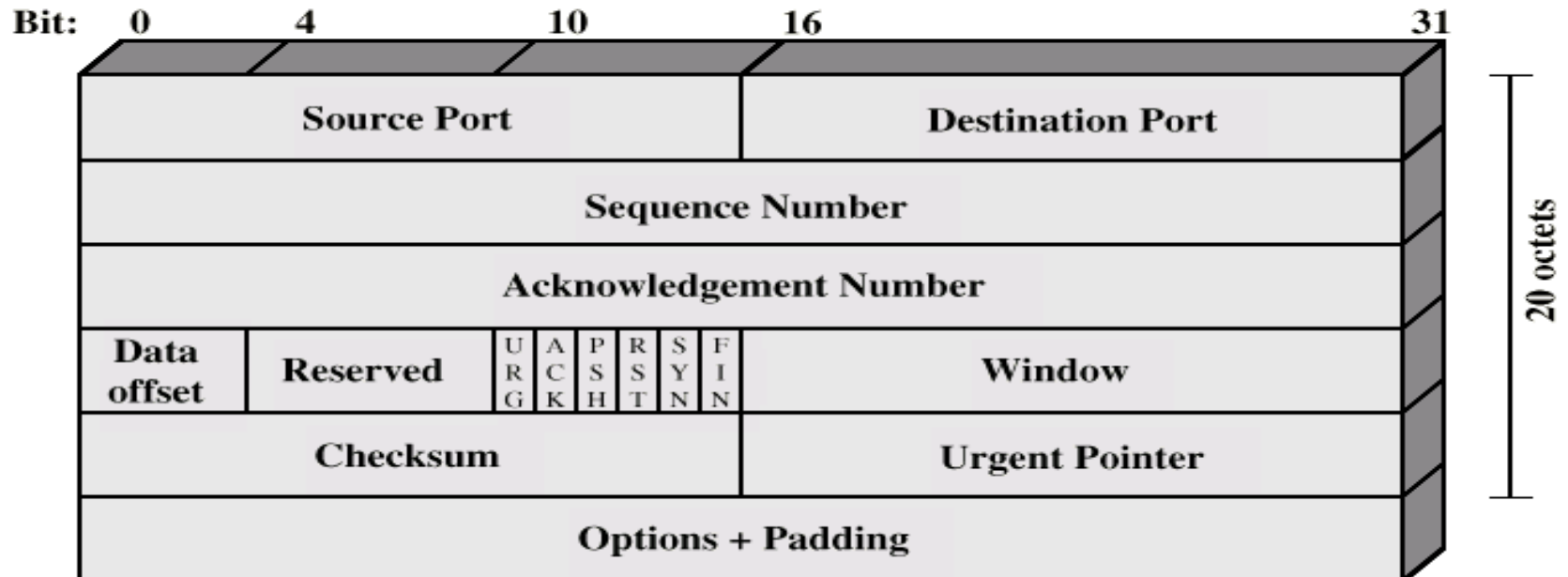
# Ports and Sockets

---

- **Port:** A number on a host assigned to an application to allow multiple destinations
- **Endpoint:** A pair, a destination host number and a port number on that host
- **Connection:** A pair of end points
- **Socket:** An abstract address formed by the IP address and port number (characterizes an endpoint)

# TCP connection and header

- Unique identifier for a TCP connection
  - Source IP address and port number
  - Destination IP address and port number





# Port numbers

---

- 16-bit port numbers- 0 to 65535
- 0 to 1023 set aside as *well-known* ports
  - assigned to certain common applications
  - telnet uses 23, SMTP 25, HTTP 80 etc.
- 1024 to 49151 are registered ports
  - 6000 through 6063 for X-window server
- 49152 through 65535 are *dynamic* or *private* ports, also called *ephemeral*.

# Sequence number and window size

---

- sequence number identifies the byte in the stream between sender & receiver.
- Sequence number wraps around to 0 after reaching  $2^{32} - 1$
- Window size controls how much data (bytes), starting with the one specified by the Ack number, that the receiver can accept
- 16-bit field limits window to 65535 bytes

# Flags

---

- URG: The *urgent pointer* is valid
- ACK: The *acknowledgement number* is valid (i.e. packet has a piggybacked ACK)
- PSH: (Push) The receiver should pass this data to the application as soon as possible
- RST: Reset the connection (during 3-way handshake)
- SYN: Synchronize sequence number to initiate a connection (during handshake)
- FIN: sender is finished sending data

# Piggybacking

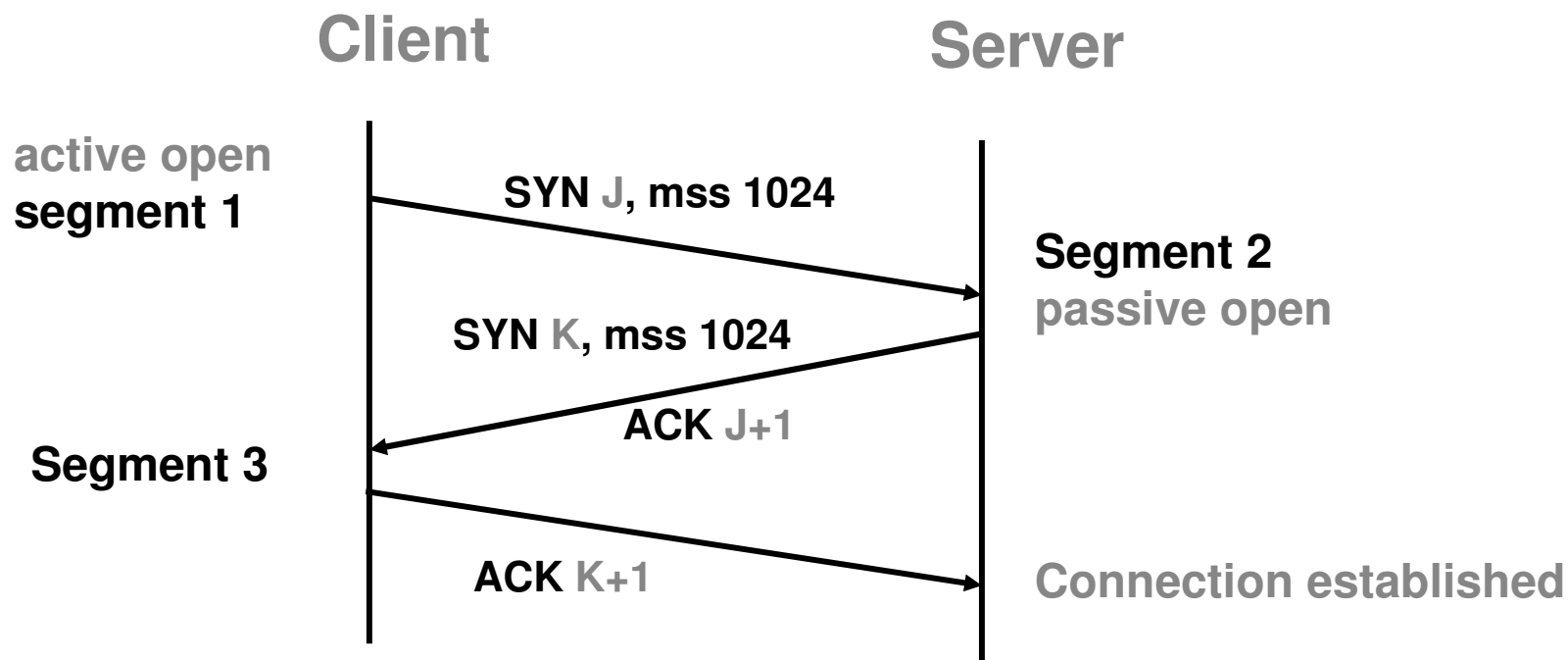
---

- ACKs usually contain the number of the next frame that is expected
- if a station has data to send, as well as ACKs, then it can send both together in one frame - piggybacking
- if a station has an ACK to send, but no data, it can send a separate ACK frame
- if a station has data but no new ACK to send, it must repeat the last ACK

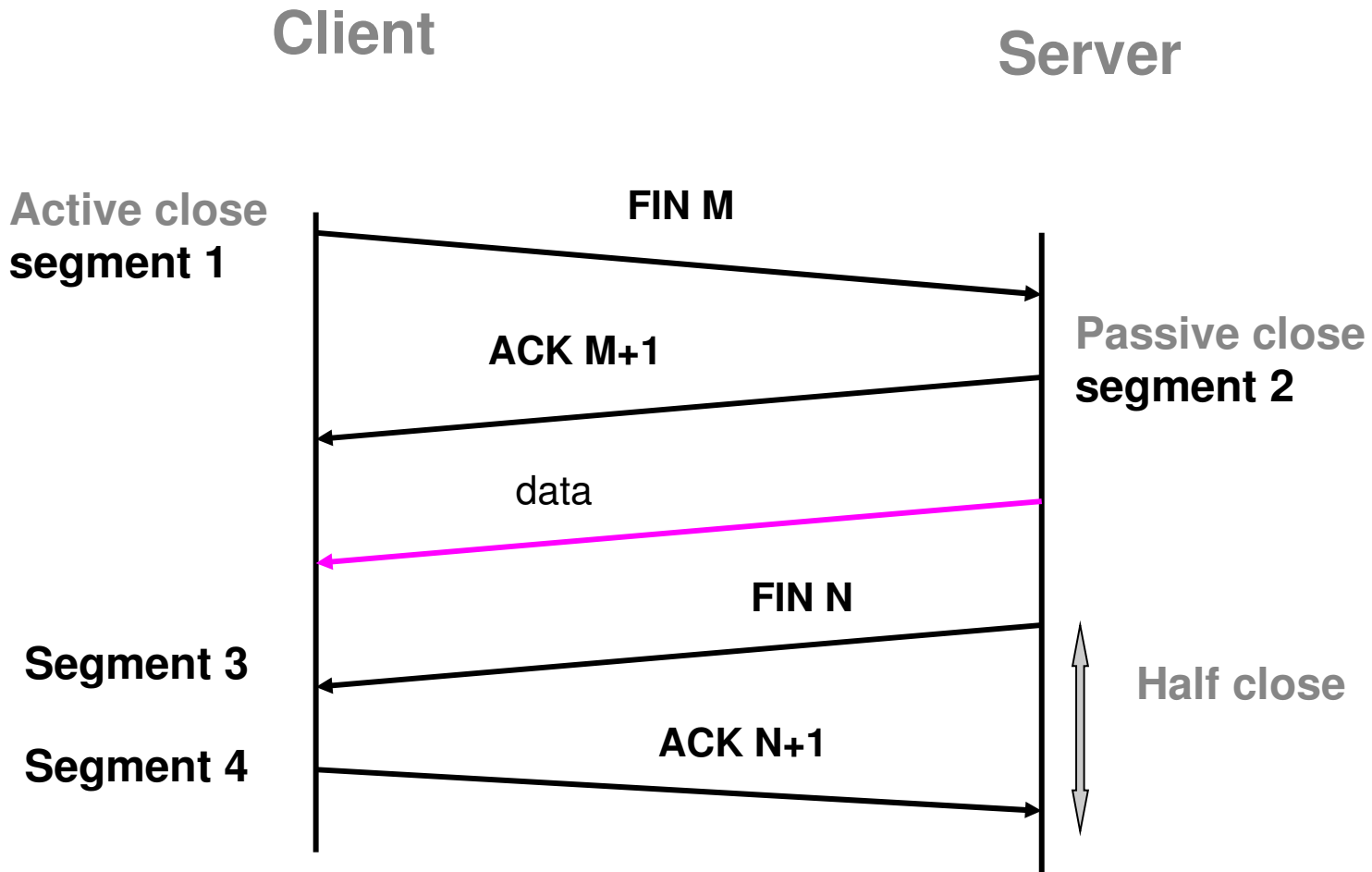
# Connection establishment

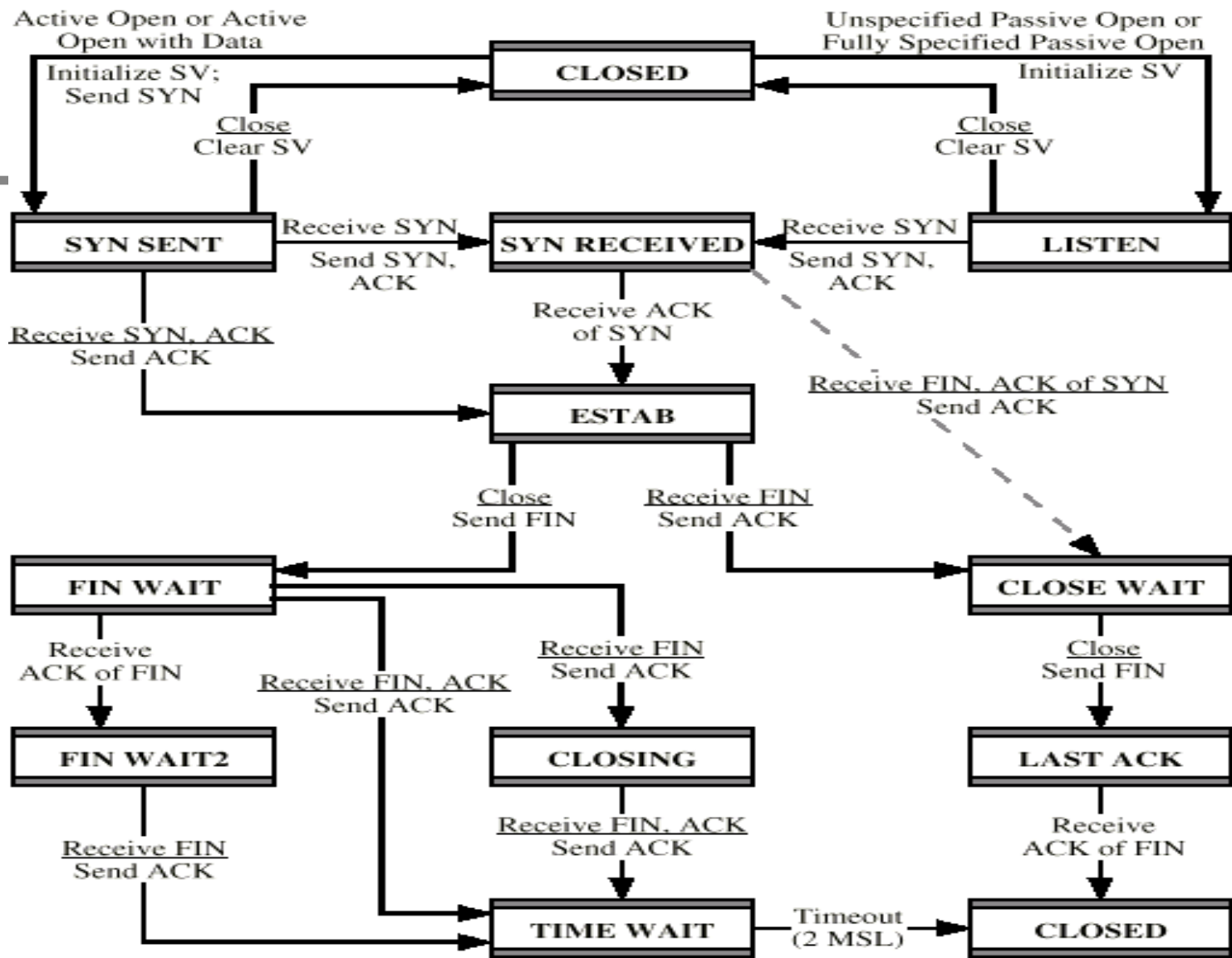
---

- Three-way handshake

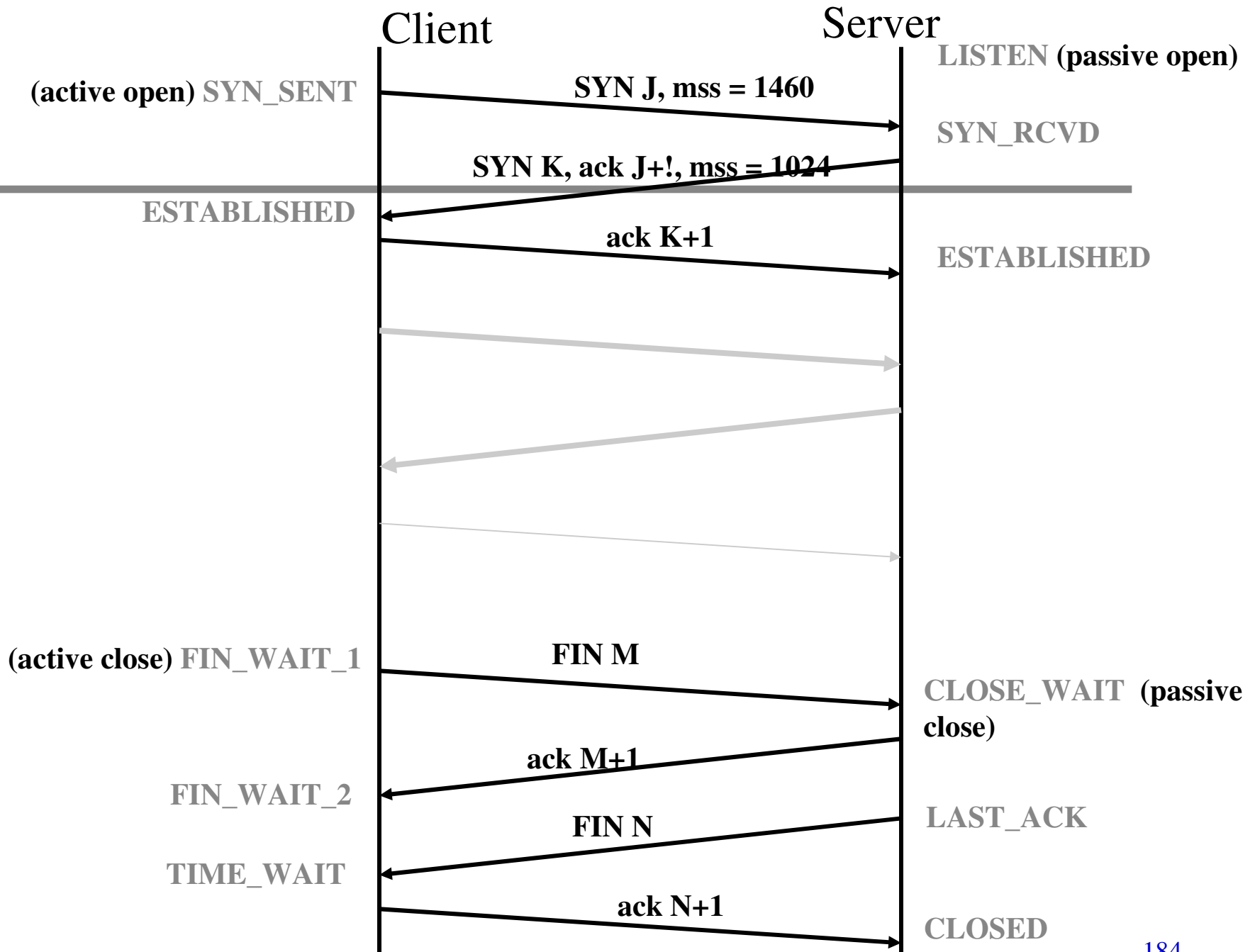


# Connection termination





SV = state vector  
 MSL = maximum segment lifetime





# Timeouts and retransmission

---

- TCP manages four different timers for each connection
  - retransmission timer: when awaiting ACK
  - persist timer: keeps window size information flowing
  - keepalive timer: when other end crashes or reboots
  - 2MSL timer: for the TIME\_WAIT state

# RTT estimation

---

- Accurate timeout mechanism is important for congestion control
- Fixed: Choose a timer interval apriori;
  - useful if system is well understood and variation in packet-service time is small.
- Adaptive: Choose interval based on past measurements of RTT;
  - Typically  $RTO = 2 * EstimatedRTT$

# Exponential averaging filter

---

- Measure SampleRTT for segment/ACK pair
- Compute weighted average of RTT
  - $\text{EstimatedRTT} = \alpha \text{PrevEstimatedRTT} + (1 - \alpha) \text{SampleRTT}$
- Choose
  - small  $\alpha$  if RTTs vary quickly; large  $\alpha$  otherwise
  - Typically  $\alpha$  between 0.8 and 0.9
- Optimizations
  - Jacobson-Karek; Karn-Partridge algorithms

# Flow control

---

- sliding window flow control: allows multiple frames to be in transit at the same time
  - Window size may grow or shrink depending on receiver and congestion feedback
- Receiver uses an AdvertisedWindow to keep sender from overrunning
  - Sender buffer size: MaxSendBuffer
  - Receive buffer size: MaxRcvBuffer

# Advertised window

---

- Receiving side
  - $\text{AdvertisedWindow} = \text{MaxRcvBuffer} - (\text{LastByteRcvd} - \text{NextByteRead})$
- Sending side
  - $\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$
- Sender uses persist timer to probe receiver when  $\text{AdvertisedWindow} = 0$ .

# Silly-Window syndrome

---

- slow application receiver
  - TCP advertises small windows
  - Sender sends small segments
- solution
  - Receiver advertises window only if size is MSS or half the buffer
  - Sender typically sends data only if a full segment can be sent

# Data Transfer using TCP

---

- On a packet count basis,
  - 50% segments bulk data (FTP, E-mail)
  - 50% segments interactive data (Rlogin)
- On a byte-count basis,
  - 90% bulk data and 10% interactive
  - bulk data segments tend to be full sized
- Different algorithms come into play for each

# Interactive Input

---

- Small segments cause congestion
- Delayed ACKs are often used to reduce number of segments
- Nagle algorithm (RFC 896)
  - If TCP connection has outstanding data that has not been ACKed, small segments cannot be sent until ACKs come in
  - algorithm is self-clocking: the faster the ACKs come back, the faster data is sent
  - May be turned off e.g. X Window server



# Bulk data

---

- TCP uses the sliding window protocol for efficient transfer of bulk data
- Issues
  - window size, slow start, congestion...
  - RTT measurements, congestion avoidance, fast retransmit and recovery...
- TCP variants
  - Tahoe, Reno, Vegas, SACK

# Additive increase

# Multiplicative decrease

---

- CongestionWindow: limits amount of data in transit
  - $\text{MaxWin} = \text{MIN}(\text{CongestionWindow}, \text{AdvertisedWindow})$   
 $\text{EffWin} = \text{MaxWin} - (\text{LastByteSent} - \text{LastByteAcked})$
- Increase CongestionWindow (linearly) when congestion goes down
- Decrease CongestionWindow (multiply) when congestion goes up
- Source infers congestion upon timeout

# Slow start and Congestion avoidance

---

- CongestionWindow: cwnd
- AIMD
  - Increment cwnd by one packet per RTT
  - Divide cwnd by two upon each timeout
- Slow Start
  - Increase cwnd exponentially upto a threshold (ssthresh)
- Congestion Avoidance
  - Increase cwnd linearly after ssthresh

# Fast retransmit and Fast recovery

---

- Waiting for TCP sender timeouts leads to idle periods
- Fast retransmit
  - use duplicate ACKs to trigger retransmission
- Fast recovery
  - remove the slow start phase;
  - go directly to half the last successful cwnd

# TCP Tahoe

---

- Slow start with congestion avoidance
- Detects congestion using timeouts
- Initialization
  - cwnd initialized to 1;
  - ssthresh initialized to  $1/2$  MaxWin.
- Upon timeout
  - set cwnd to 1,
  - ssthresh to  $1/2$  CurrentWindow,
  - enter slow start.

# TCP Reno

---

- Detects congestion loss using timeouts as well as duplicate ACKs.
- On timeout, TCP Reno behaves same as TCP Tahoe.
- On fast retransmit (receipt of 3 ACKs with same cumulative sequence number),
  - decreases `ssthresh` and `CurrentWindow` to half the previous values; skips exponential phase and goes directly into linear increase.

---

# Application Programming

# Application layer

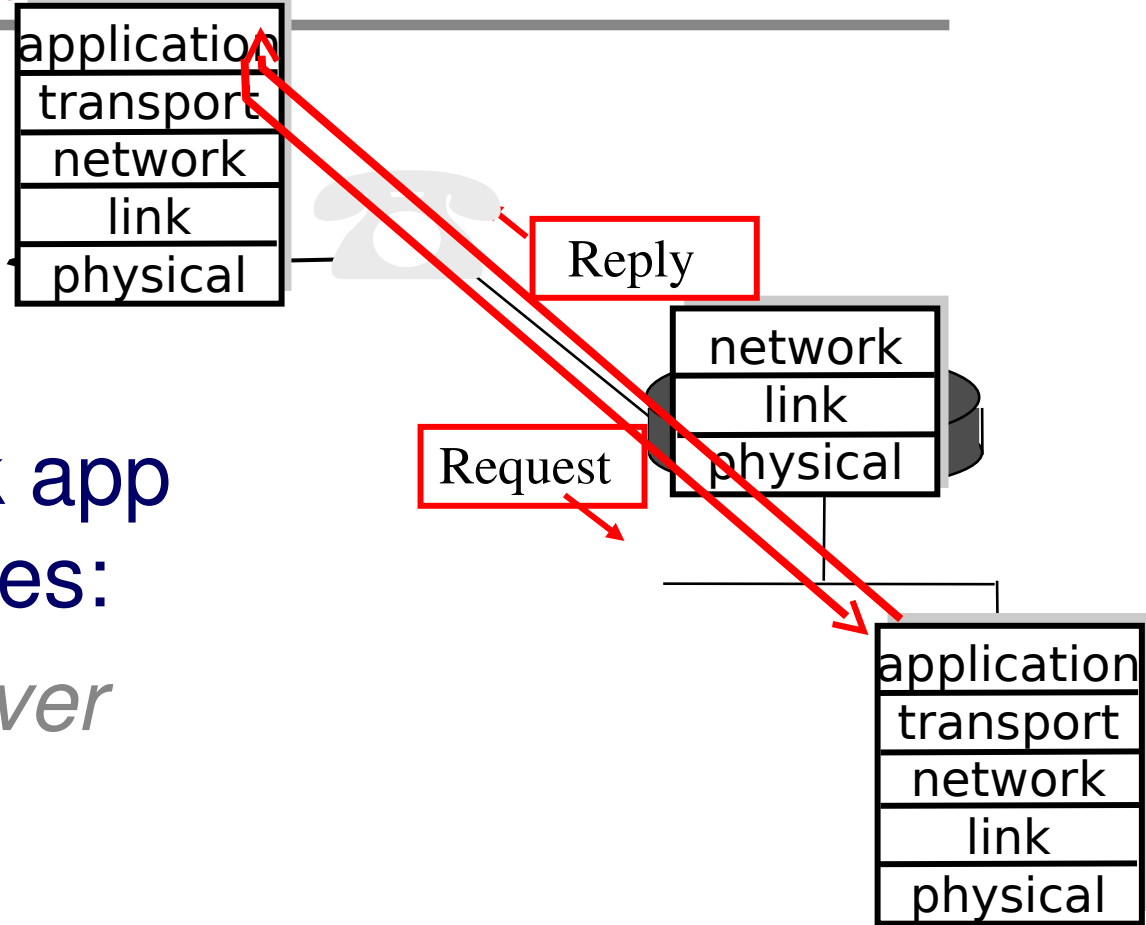
---

- communicating, distributed processes
  - exchange messages to implement application
  - e.g., email, file transfer, the Web
- Application layer protocols
  - one “piece” of an application
  - define messages exchanged by application components and actions taken
  - uses services provided by lower layer protocols



# Client-Server paradigm

Typical network app  
has two pieces:  
*client* and *server*

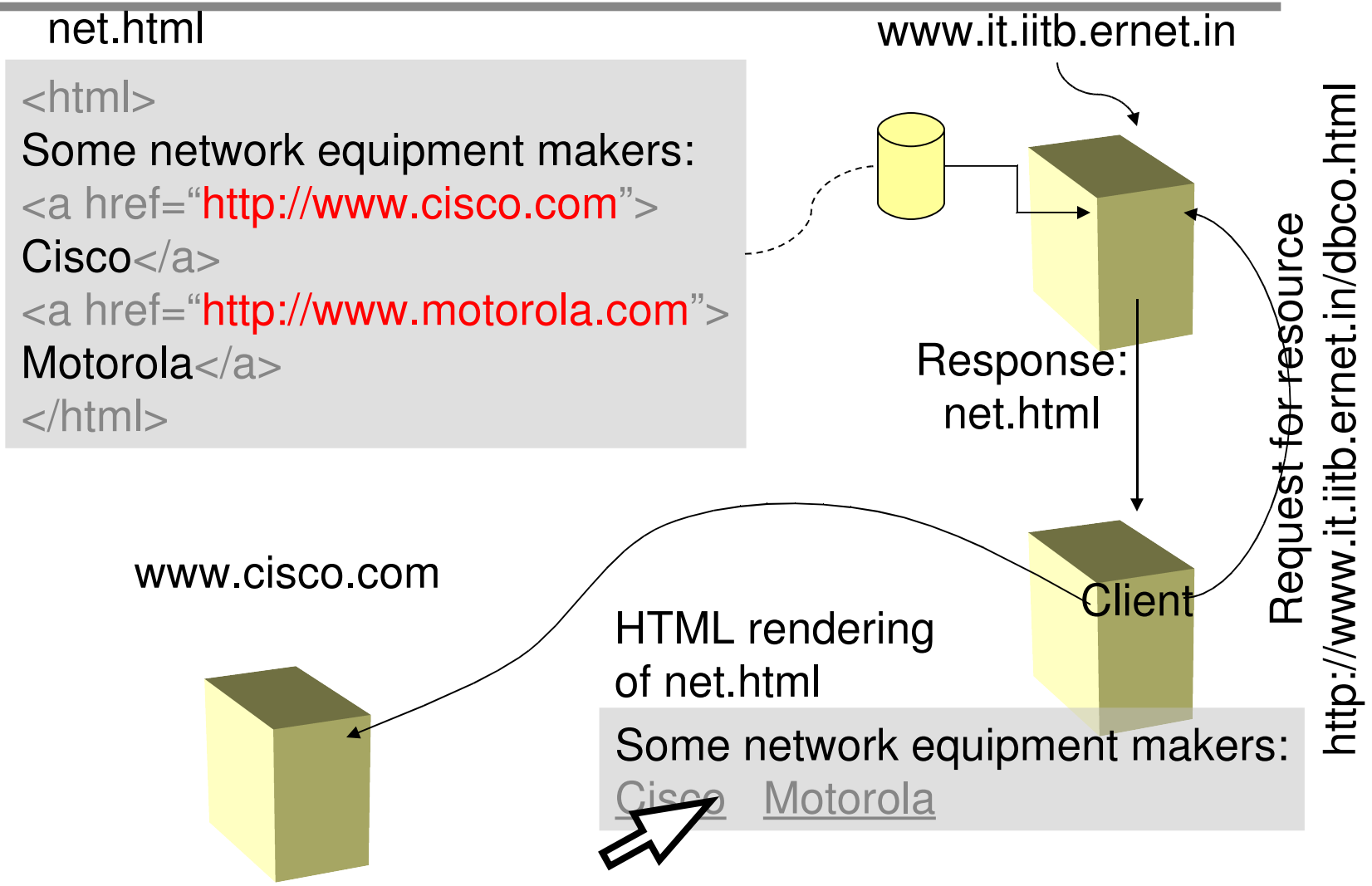


# Client-Server actions

---

- Client:
  - initiates contact with server (“speaks first”)
  - typically requests service from server
  - e.g.: sends request for Web page
- Server
  - provides requested service to client
  - e.g., sends requested Web page

# Example: Web access (HTTP)

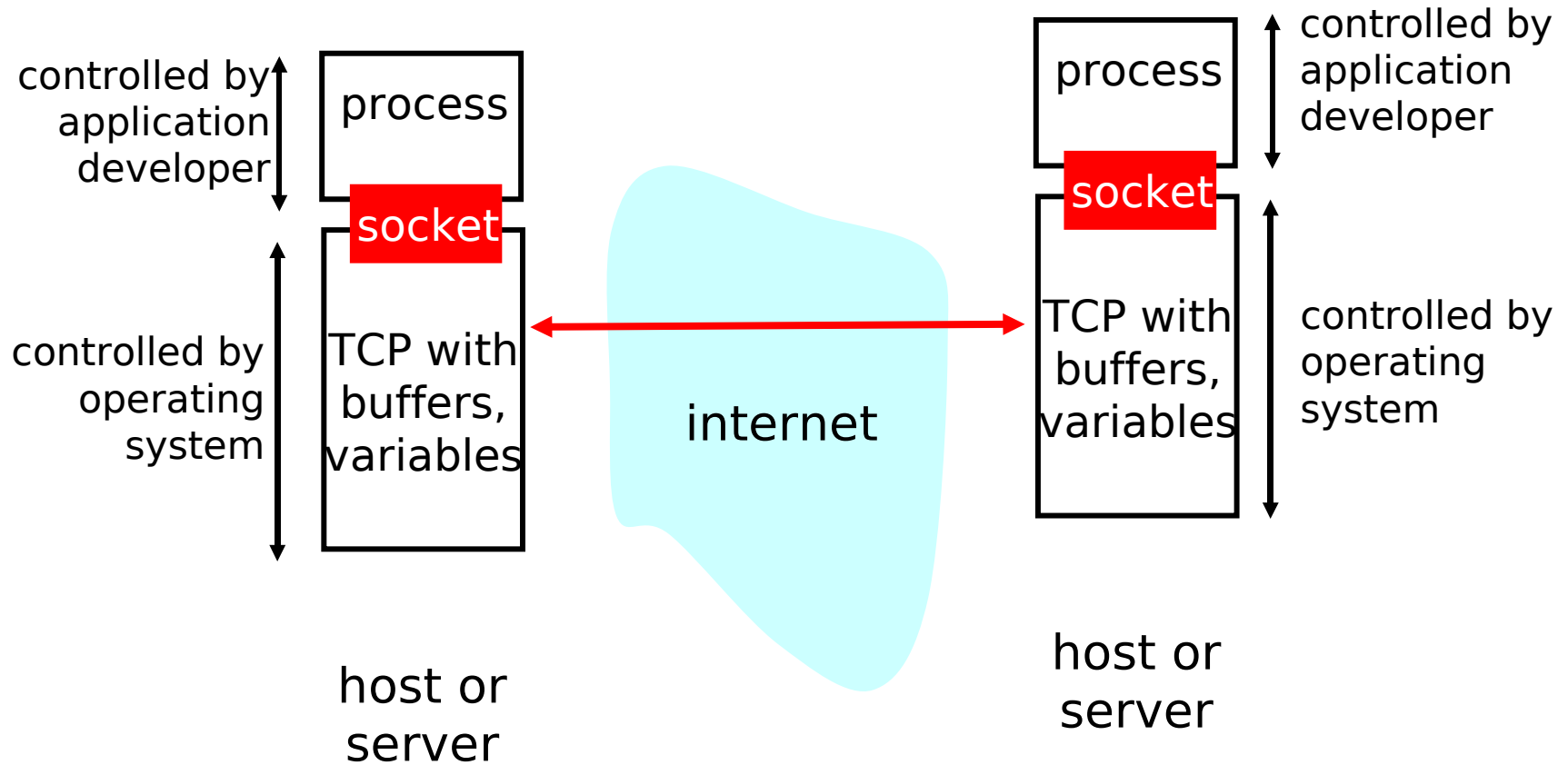


# Sockets API

---

- Interface between application and transport layer
  - two processes communicate by
  - sending data into a socket
  - reading data out of a socket
- Client “identifies” Server process using
  - IP address ; port number

# Sockets interface



# Client actions

---

- Create a socket (*socket*)
- Map server name to IP address (*gethostbyname*)
- Connect to a given port on the server address (*connect*)
  
- Client must contact server first!

# Server actions

---

- Create a socket (*socket*)
- Bind to one or more port numbers (*bind*)
- Listen on the socket (*listen*)
- Accept client connections (*accept*)
- Server process must be running!

# Client architecture

---

- **Simpler than servers**
  - Typically do not interact with multiple servers concurrently
  - Typically do not require special ports
- **Most client software executes as a conventional program**
- **Most clients rely on OS for security**



# Server architecture

---

Depends on requirements for

- Type of connection
  - **Connection-Oriented**: reliable but needs OS resources
  - **Connection-less**: less resources but less reliable
- Server state
  - **Stateless**: each transaction is independent
  - **Stateful**: server maintains state
- Servicing of requests
  - **Iterative**: accept requests one at a time
  - **Concurrent**: fork a new process for each client

# Remote Procedure Call (RPC)

---

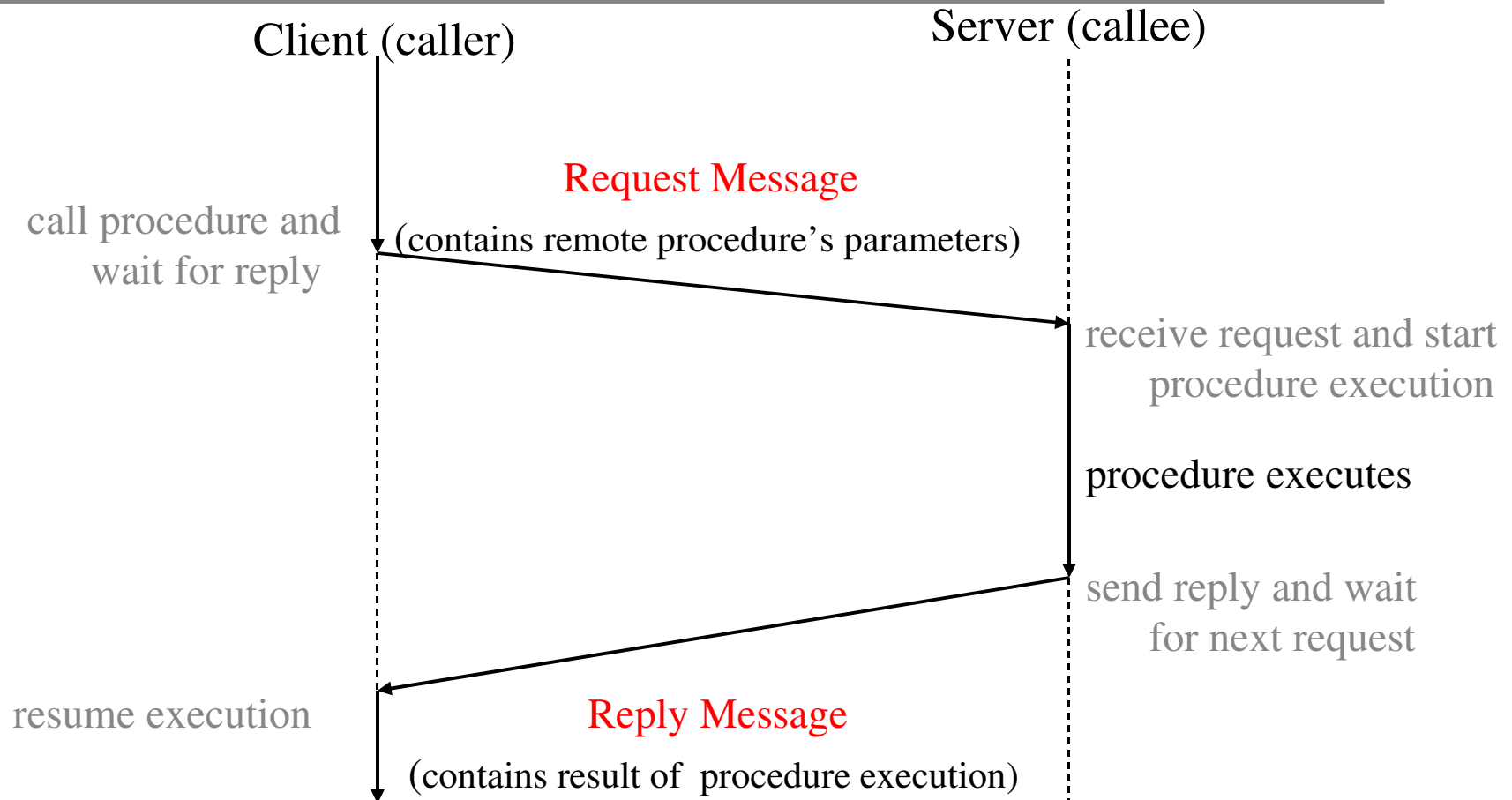
- **Goal:** make distributed programming as simple as possible
  - distributed programming should be similar to normal sequential programming
- Allow programs to call procedures located on other machines

# RPC: central idea

---

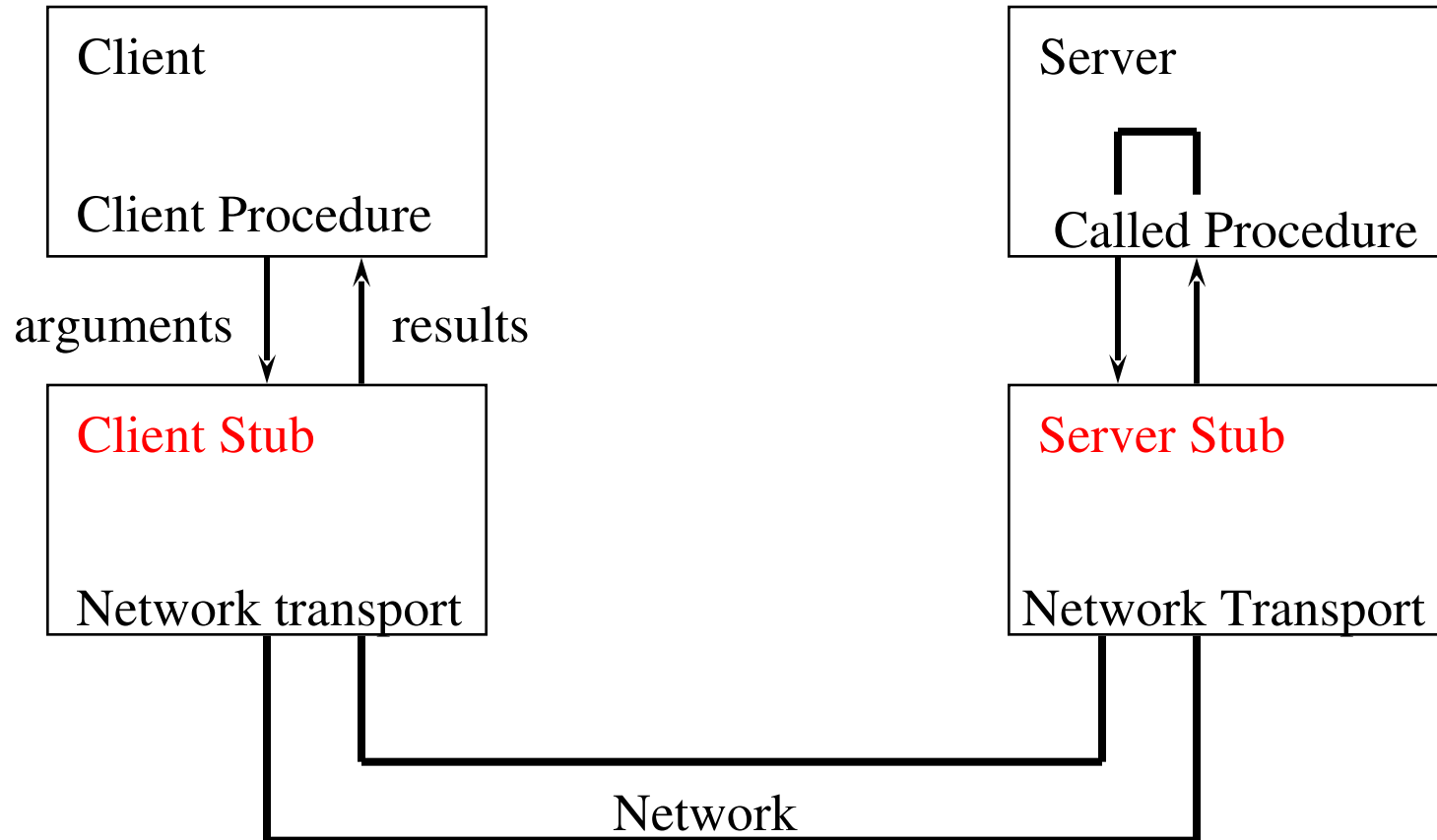
- when program on machine A calls a procedure on machine B, the calling process on A is suspended
- the execution of the called procedure takes place on B
- when the procedure returns, the process on A resumes
- Exchange takes place via procedure arguments and return value

# RPC functionality



# RPC design

---



# RPC advantages

---

- Extends the conventional procedure call to the client/server model
- Remote procedures accept arguments and return results
- Makes it easy to design and understand programs
- Helps programmer to focus on the application instead of the communication protocol

---

# Network Security

# Network security

---

- Security Plan (RFC 2196)
  - Identify assets
  - Determine threats
  - Perform risk analysis
  - Implement security mechanisms
  - Monitor events, handle incidents
- Cost of protecting should be less than the cost of recovering



# Security requirements

---

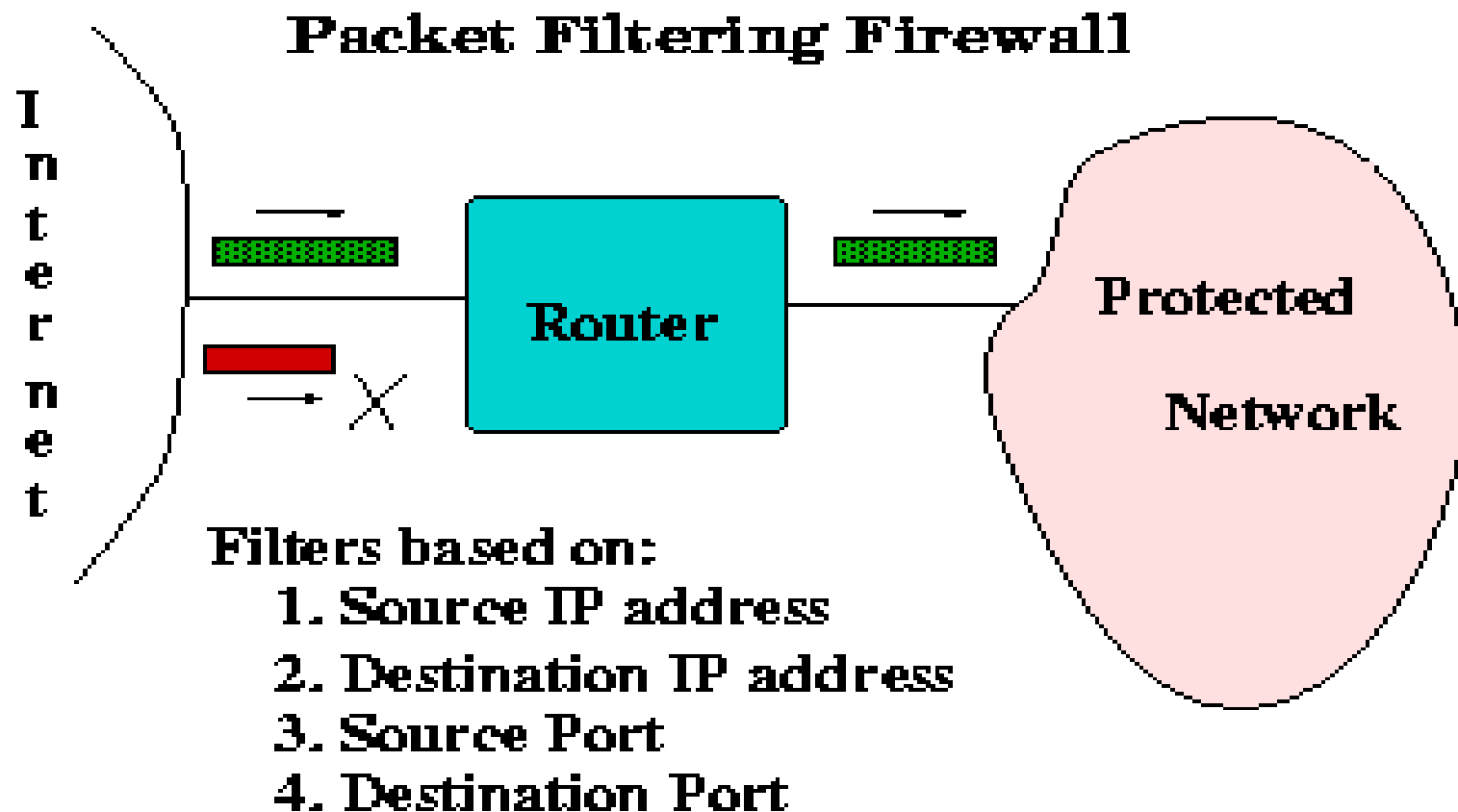
- Confidentiality:
  - No unauthorized disclosure
- Integrity:
  - No unauthorized modification
- Authentication:
  - Assurance of identity of originator
- Non-Repudiation:
  - Originator cannot deny sending the message

# Security threats and levels

---

- Threats
  - Host: Unauthorized access
  - Transmission: sniffing, masquerading
- Host level
  - Authentication and access control
- Network level
  - Firewalls and proxies
- Application level
  - Encryption and signatures

# Firewalls



***Default deny***      ***vs.***      ***Default allow***

# Sample filtering rules

---

- Permit incoming Telnet sessions only to a specific internal hosts
- Permit all outbound Telnet and FTP sessions
- Deny all incoming traffic from specific external networks

# Cryptography

---

## Components:

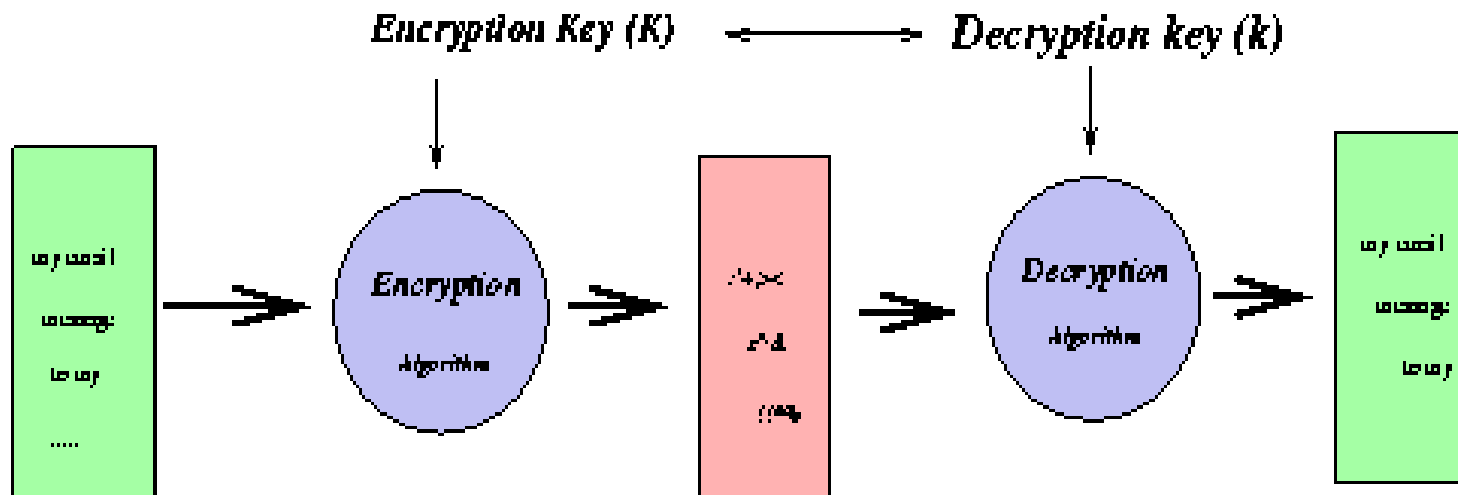
- Plain text
- Encryption/Decryption Algorithms
- Encryption/Decryption Keys
- Cipher text

# Techniques

---

- Symmetric/Private Key: (DES)
  - secure environment for key exchange
- Asymmetric/Public Key: (RSA)
  - private-public key pair
- Hash Algorithms: (MD5, SHA)
  - message integrity
- Digital Signatures:
  - integrity and authentication

# *SYMMETRIC/SHARED KEY Encryption*



## *Advantages*

- \* *Fast*
- \* *Special Hardware*
- \* *Built-in Authentication*

*Message Integrity*

## *Disadvantages*

- \* *How to Exchange Keys?*
- \* *Puzzle in Tannenbaum*

# Some security tools

---

- Password tools: crack, COPS
- Host-based auditing tools: Tripwire, rdist
- Virus detection software: Norton, McAfee
- Network-based auditing tools: SATAN, SAINT
- Network traffic analyzers: tcpdump, sniffer
- Firewall packages/products: Checkpoint, PIX
- Encryption utilities and libraries: PGP



# Summary

---

- Multiplexing
- Error detection and correction
- Medium access control
- Switching and scheduling
- Addressing and routing
- Flow and congestion control
- Application API
- Security