

# Direct Connection Networks

## Acknowledgements

These Slides have been adapted from the originals made available by J. Kurose and K. Ross  
All material copyright 1996-2009  
J.F Kurose and K.W. Ross, All Rights Reserved

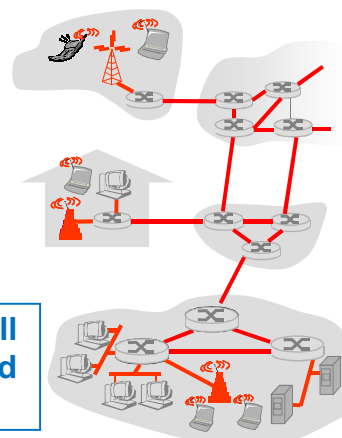
1

## Overview

### Some terminology:

- hosts and routers are **nodes**
- communication channels that connect adjacent nodes along communication path are **links**
  - wired/wireless links
  - Point-to-point/shared links

In this part of the course we will look at how data are transferred between adjacent nodes





## Goals

- Introducing direct connection networks
- understanding principles behind Data Link layer services:
  - reliable data transfer
    - error detection, correction
    - Acknowledgement, timeout, and re-transmission
  - flow control
  - sharing a broadcast channel
    - multiple access
    - link layer addressing
- instantiation and implementation of various link layer technologies

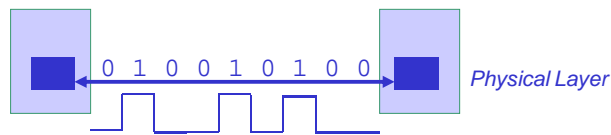


## Direct Connection Networks

- **Introduction**
- Error detection and correction
- Reliable Data Transfer
- PPP
- Multiple access protocols
- Local Area Networks (LAN)
- Ethernet

# Introduction

## Physical Link

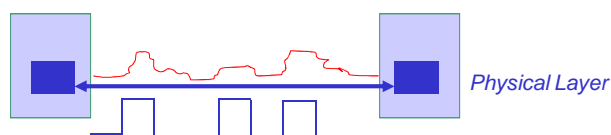


## Encoding

Bits are coded through an electric/electromagnetic/light signal and send over the physical link

# Real-life Problems

- ❑ The transmission channel is not ideal
  - Signal attenuation
  - Noise
    - Interferences, fading, ...
- ❑ The received data sequence may be different from the transmitted one

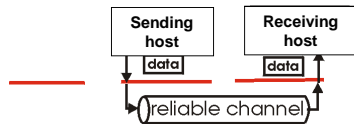


010010100

011010110 Direct Connection Networks

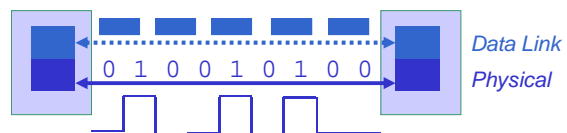
## Data Link Layer

- Reliable delivery between adjacent nodes



## Data-Link Layer Services

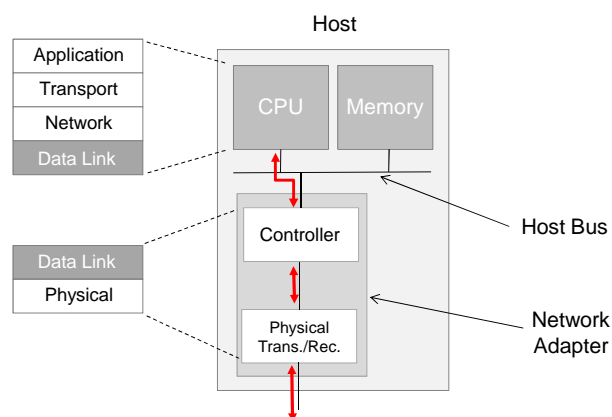
- Framing
  - encapsulate datagram into frame, adding header, trailer



## Data-Link Layer Services

- **Error detection:**
  - receiver detects presence of errors
- **Error correction:**
  - receiver identifies *and corrects* bit error(s)
- **Reliable Data Transfer**
  - Through acknowledgements and retransmissions
- **Flow control:**
  - pacing between adjacent sending and receiving nodes
- **Half-duplex and full-duplex**
  - with half duplex, nodes at both ends of link can transmit, but not at same time

## Where is the link layer implemented?





## Direct Connection Networks

- Introduction
- Error detection and correction
- Reliable Data Transfer
- PPP
- Multiple access protocols
- Local Area Networks (LAN)
- Ethernet

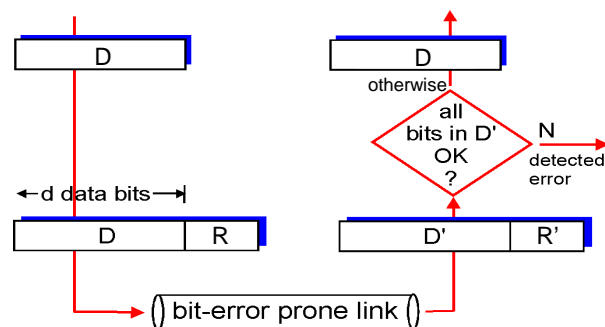


## Error Detection

R= Redundancy bits

D= Data protected by error checking, may include header fields

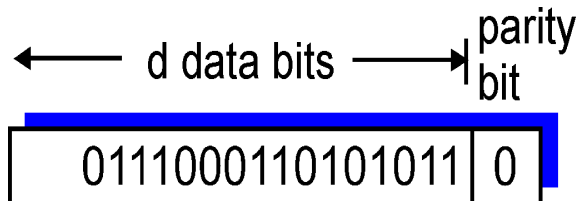
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger R field yields better detection





## Parity Checking

**Single Bit Parity:**  
Detect single bit errors



## Checksum

**Goal:** detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer *only*)

**Sender:**

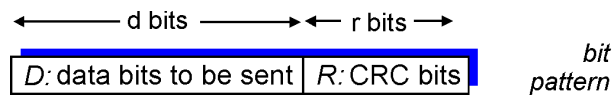
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into packet checksum field

**Receiver:**

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected.  
*But maybe errors nonetheless?*

# Cyclic Redundancy Check (CRC)

- widely used in practice (Ethernet, 802.11 WiFi, ...)
- view data bits,  $D$ , as a binary number
- choose  $r+1$  bit pattern  $G$  (*Generator*)
- goal: choose  $r$  CRC bits,  $R$ , such that
  - $\langle D, R \rangle$  exactly divisible by  $G$  (using modulo-2 arithmetic)
  - receiver knows  $G$ , divides  $\langle D, R \rangle$  by  $G$ .
  - If non-zero remainder: **error detected!**
- can detect all burst errors less than  $r+1$  bits



$D \cdot 2^r \text{ XOR } R$ 
*mathematical formula*

# CRC – How to derive R?

Want  $R$  such that:

$$D \cdot 2^r \text{ XOR } R = nG$$

*equivalently:*

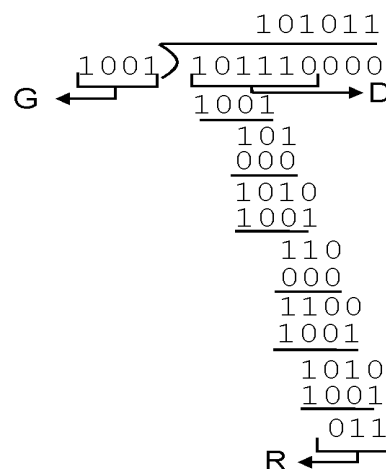
$$D \cdot 2^r = nG \text{ XOR } R$$

*equivalently:*

if we divide  $D \cdot 2^r$  by  $G$ ,  
the remainder is equal  
to  $R$

$$R = \text{remainder} \left[ \frac{D \cdot 2^r}{G} \right]$$

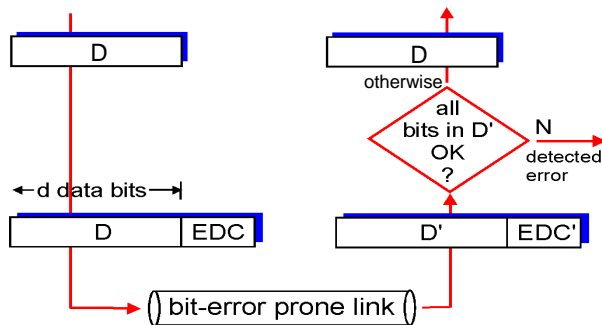
## Example





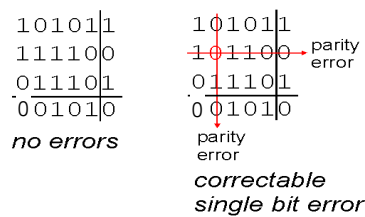
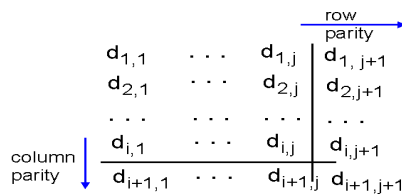
# Forward Error Correction (FEC)

EDC Error Detection and Correction (redundancy bits)  
 D Data protected by error checking, may include header fields



# Two-Dimensional Bit Parity

□ Detects and correct single bit values





## Direct Connection Networks

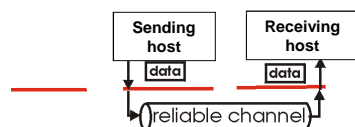
- Introduction
- Error detection and correction
- **Reliable Data Transfer**
- PPP
- Multiple access protocols
- Local Area Networks (LAN)
- Ethernet



## Principle of Reliable Data Transfer

- Important in Data Link, Transport, and Application layers
- Top-10 list of important networking topics!

What we would like to get



## Reliable data transfer: overview

*`rdt_send()`: called from above, (e.g., by network). Passed data to deliver to receiver upper layer*

*`deliver_data()`: called by `rdt` to deliver data to upper*



*`udt_send()`: called by `rdt`, to transfer packet over unreliable channel to receiver*

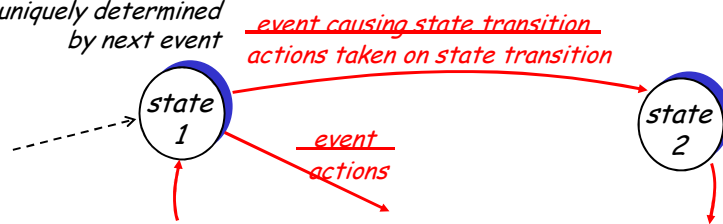
*`rdt_rcv()`: called when packet arrives on rcv-side of channel*

## Reliable data transfer: getting started

- incrementally develop sender, receiver sides of reliable data transfer protocol (`rdt`)
- consider only unidirectional data transfer
  - but control info will flow on both directions!
- use finite state machines (FSM) to specify sender, receiver

# Reliable Data Transfer: FSM

*state: when in this "state" next state uniquely determined by next event*



## Rdt1.0: reliable transfer over a reliable channel

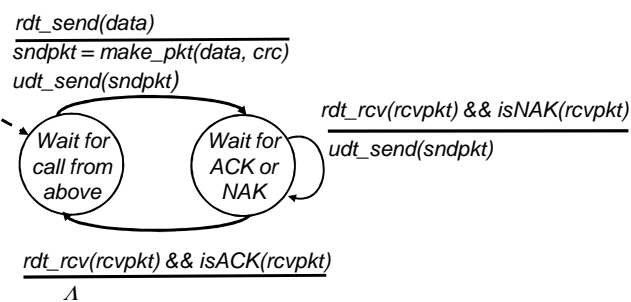
- underlying channel perfectly reliable
  - no bit errors
  - no loss of packets
- separate FSMs for sender, receiver:
  - sender sends data into underlying channel
  - receiver read data from underlying channel



## Rdt2.0: channel with bit errors

- Underlying channel may flip bits in packet
  - Error detection to detect bit errors
    - CRC (Data Link layer)
    - Checksum (Transport layer)
- How to recover from errors?
  - *acknowledgements (ACKs)*
    - receiver explicitly tells sender that pkt received OK
  - *negative acknowledgements (NAKs)*
    - receiver explicitly tells sender that pkt had errors
    - sender retransmits pkt on receipt of NAK
- Automatic Repeat reQuest (ARQ) protocol
  - error detection (receiver side)
  - receiver feedback
    - control msgs (ACK,NAK) sent from receiver to sender
  - retransmission (sender side)

## rdt2.0: FSM specification



sender

stop and wait  
 Sender sends one packet,  
 then waits for receiver  
 response



# rdt2.0 has a fatal flaw!

## What happens if ACK/NAK corrupted?

- ❑ sender doesn't know what happened at receiver!
- ❑ Error correction on ACKs/NAKs
  - Makes the channel error-free
  - Does not work on lossy channels where packets may get lost
- ❑ Re-transmission
  - sender retransmits current packet if ACK/NAK garbled
  - Possible Duplicates

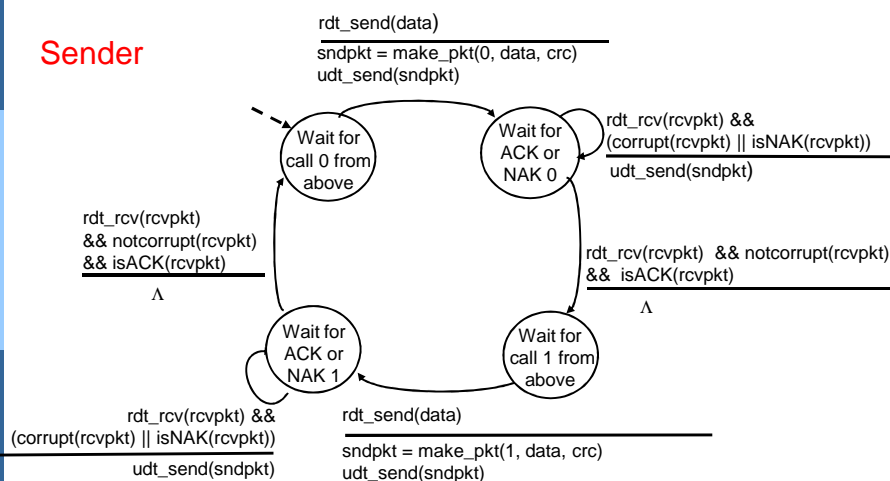
## Handling duplicates:

- ❑ sender adds *sequence number* to each packet
- ❑ receiver discards (doesn't deliver up) duplicate packets
- ❑ For a Stop-and-Wait protocol a 1-bit sequence number is enough



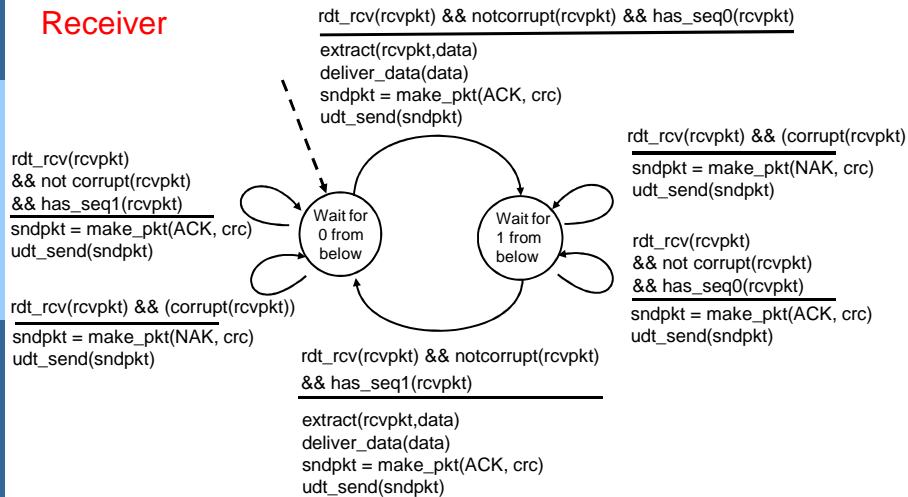
# rdt2.1: Handling of garbled ACK/NAKs

## Sender



## rdt2.1: Handling of garbled ACK/NAKs

### Receiver



## rdt2.1: discussion

### Sender:

- seq # added to pkt
- two seq. #'s (0,1) will suffice. Why?
- must check if received ACK/NAK corrupted
- twice as many states
  - state must "remember" whether "current" pkt has 0 or 1 seq. #

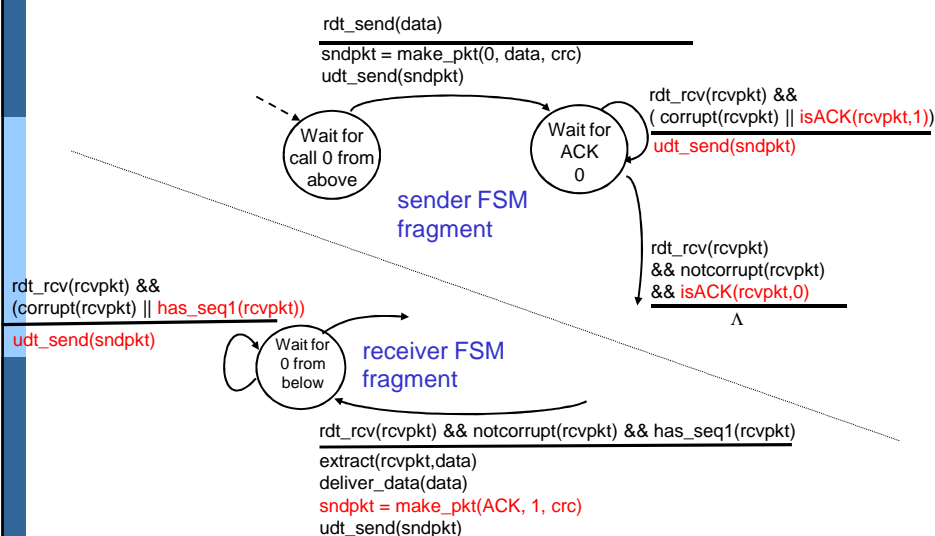
### Receiver:

- must check if received packet is duplicate
  - state indicates whether 0 or 1 is expected pkt seq #
- note: receiver can *not* know if its last ACK/NAK received OK at sender

## rdt2.2: a NAK-free protocol

- ❑ Same functionality as rdt2.1, using ACKs only
- ❑ Instead of NAK, receiver sends ACK for the last packet received OK
  - receiver must *explicitly* include the seq # of the packet being ACKed
- ❑ duplicate ACK at sender results in same action as NAK: *retransmit the current packet*

## rdt2.2: sender, receiver FSMs







## rdt3.0: channels with errors *and* loss

### New assumption:

underlying channel can also lose packets (data or ACKs)

- Error detection, seq. #, ACKs, retransmissions will be of help, *but not enough*

### New Problem:

How to detect a packet loss?

### Approach:

sender waits "reasonable" amount of time (time-out) for ACK

- retransmits if no ACK received in this time
- receiver must specify seq # of pkt being ACKed
- requires countdown timer

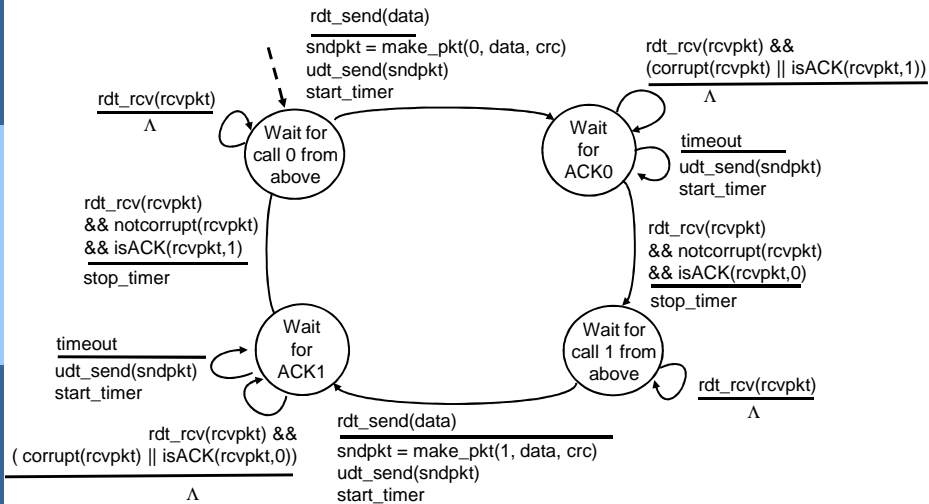


## rdt3.0: channels with errors *and* loss

### How long to wait?

- If the time-out is too long
  - The data transfer process is made slower
- If the time-out is too short
  - if pkt (or ACK) just delayed (not lost), retransmission will produce duplicates at the receiver
    - but use of seq. #'s already handles this
- The time-out should be tailored to the Round Trip Time (RTT)

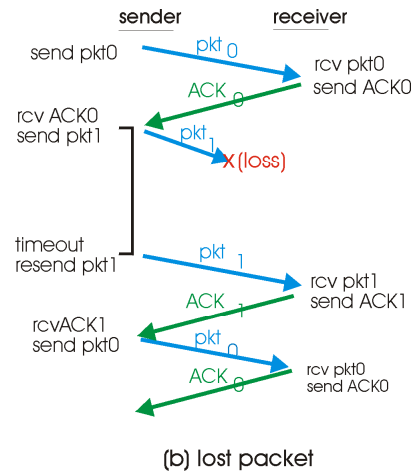
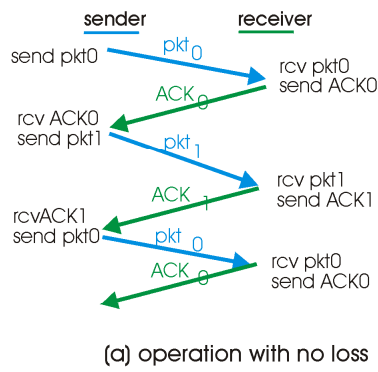
## rdt3.0 sender



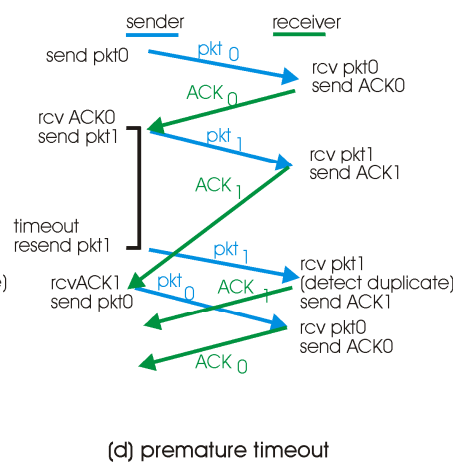
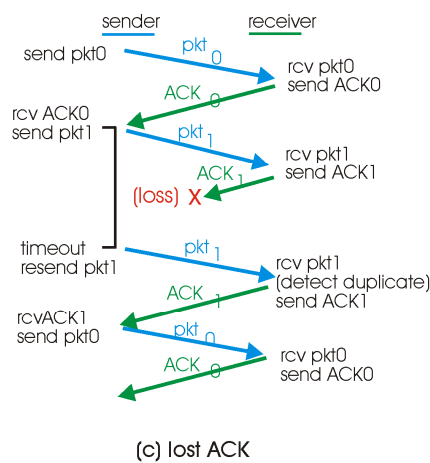
## rdt3.0 receiver

- Left to students as a homework
- Define the receiver FSM
  - Like the sender FSM shown in the previous slide

## rdt3.0 in action



## rdt3.0 in action



## Performance of rdt3.0

- rdt3.0 works, but performance stinks
- ex: 1 Gbps link, 15 ms prop. delay, 8000 bit packet:

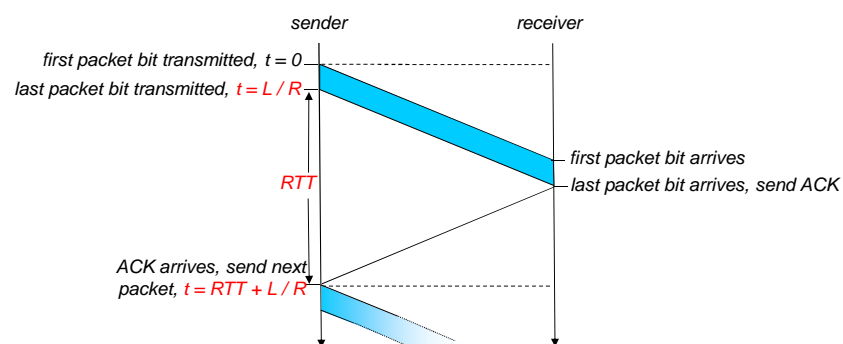
$$d_{trans} = \frac{L}{R} = \frac{8000\text{bits}}{10^9 \text{bps}} = 8\text{microseconds}$$

- $U_{sender}$ : *utilization* - fraction of time sender busy sending

$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

- 1KB pkt every 30 msec → 267 Kbps *throughput* over 1 Gbps link
- network protocol limits use of physical resources!

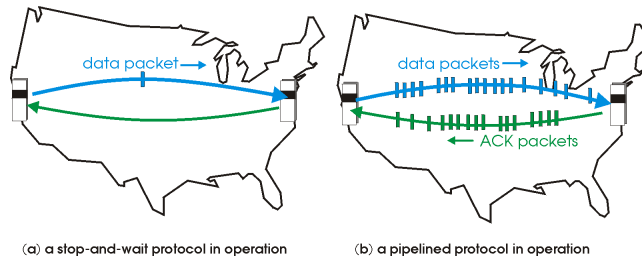
## rdt3.0: stop-and-wait operation



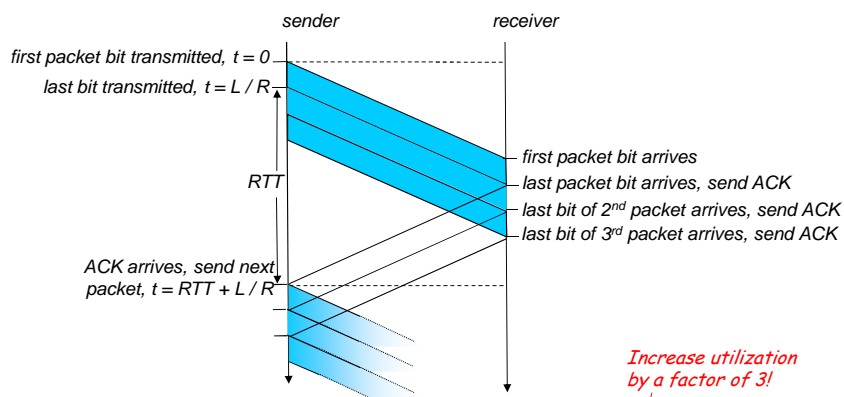
$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

# Pipelining

**Pipelining:** sender allows multiple, "in-flight", yet-to-be-acknowledged pkts



# Pipelining: increased utilization



$$U_{\text{sender}} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$



## Pipelining: Additional Mechanisms

- Range of sequence numbers must be increased
- Buffering at sender and/or receiver
  - The sender must buffer all packets not yet acknowledged
  - The receiver may buffer out-of-order packets
- The range of seq numbers and buffer size depend on how the protocol manages lost, corrupted, and delayed packets
- Error recovery strategies
  - Go-back-N
  - Selective Repeat



## Pipelining Protocols

### Go-back-N

- *sender*: up to N unACKed pkts in pipeline
- *receiver*: only sends cumulative ACKs
  - doesn't ACK pkt if there's a gap
- *sender*: has timer for oldest unACKed pkt
  - if timer expires: retransmit all unACKed packets

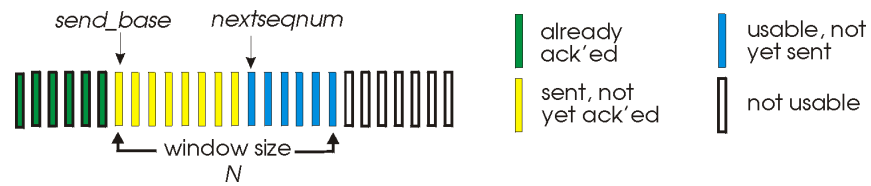
### Selective Repeat

- *sender*: up to N unACKed packets in pipeline
- *receiver*: ACKs individual pkts
- *sender*: maintains timer for each unACKed pkt
  - if timer expires: retransmit only unACKed packets

# Go-Back-N

## Sender:

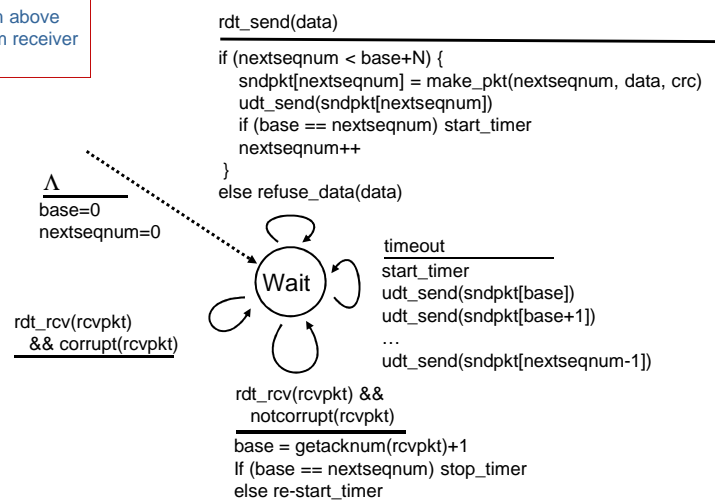
- k-bit seq # in pkt header
- "window" of up to N, consecutive unACKed pkts allowed



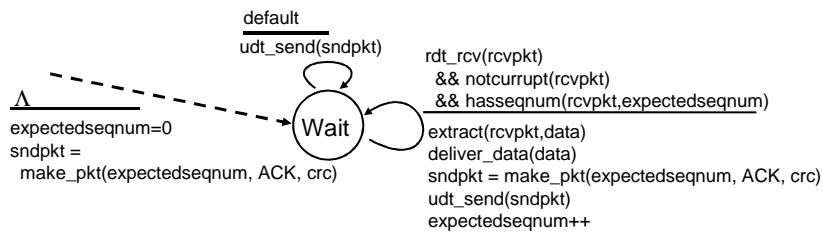
- ACK(n): ACKs all pkts up to, including seq # n ("cumulative ACK")
  - may receive duplicate ACKs (see receiver)
- timer for the oldest packet only (send base)
- timeout: retransmit pkt sendbase and all higher seq # pkts in window

# GBN: sender extended FSM

- Events:**
- Data from above
  - ACK from receiver
  - Timeout



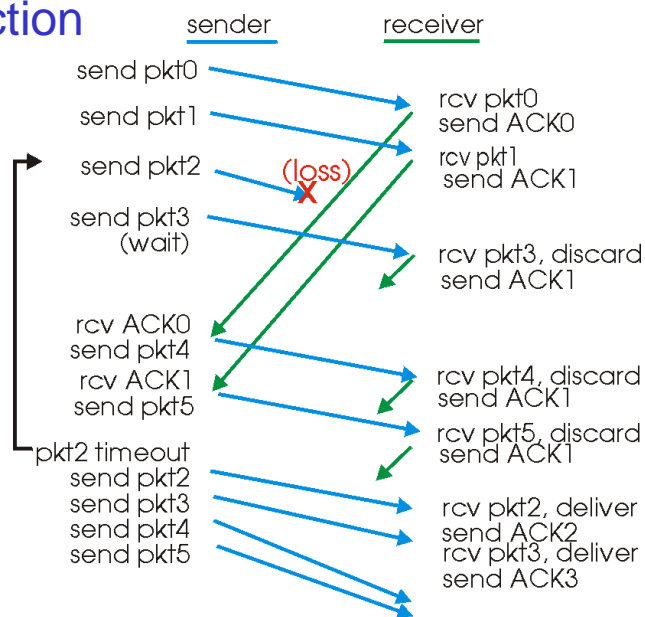
## GBN: receiver extended FSM



**ACK-only: always send ACK for correctly-received pkt with highest *in-order* seq #**

- may generate duplicate ACKs
- need only remember **expectedseqnum**
- **out-of-order pkt:**
  - discard (don't buffer) -> **no receiver buffering!**
  - Re-ACK pkt with highest in-order seq #

## GBN in action







## Limits of Go-back-N

- ❑ Packets are acked on a *cumulative* base
- ❑ Upon experiencing a time-out the sender retransmits *all* packets since the last received in order
  - Un-necessary re-transmissions
    - Consume bandwidth
    - Consume energy
  - The receiver does not need to buffer out-of-order packets
- ❑ Complexity is shifted at the sender side
  - The receiver only needs to know *expectedseqnum*

DataLink Layer 49

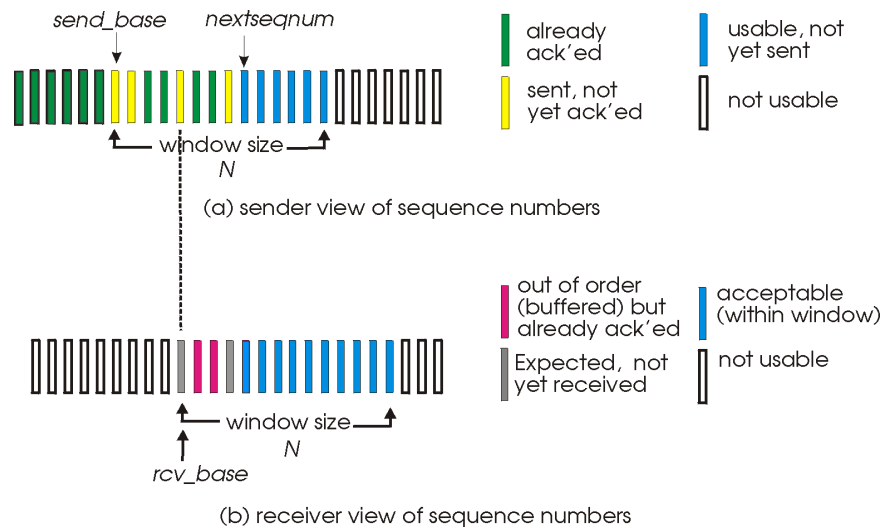


## Selective Repeat (SR)

- ❑ receiver
  - *individually* acknowledges all correctly received pkts
  - buffers pkts, as needed, for eventual in-order delivery to upper layer
- ❑ sender
  - only resends pkts for which ACK not received
  - sender timer for each unACKed pkt

Direct Connection Networks 50

## SR: sender, receiver windows



## Selective Repeat

### —sender—

#### data from above :

- if next available seq # in window, send pkt

#### timeout(n):

- resend pkt n, restart timer(n)

#### ACK(n) in [sendbase, sendbase+N]:

- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

### —receiver—

#### pkt n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- If out-of-order: buffer pkt n
- If in-order: deliver pkt n also deliver buffered, in-order pkts, advance rcv\_base to next not-yet-received pkt

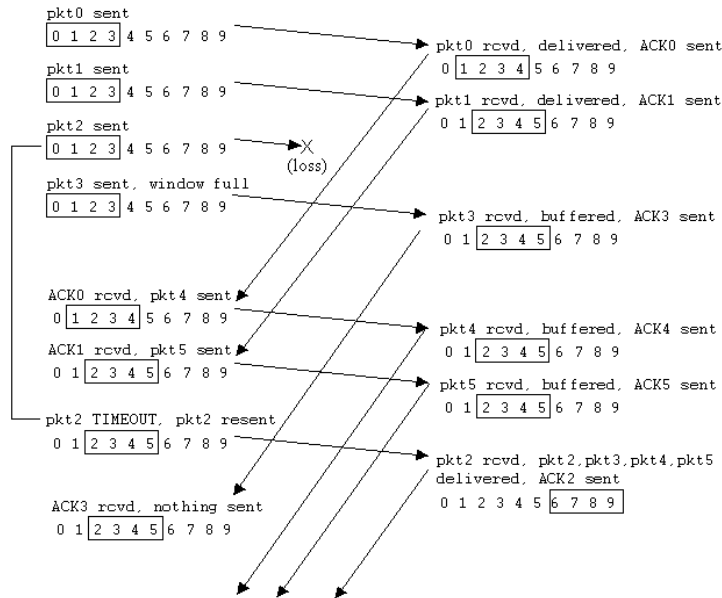
#### pkt n in [rcvbase-N, rcvbase-1]

- ACK(n)

#### otherwise:

- ignore

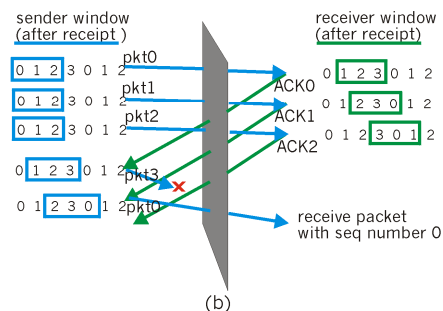
# Selective Repeat in action



# SR: dilemma

Example:

- seq #'s: 0, 1, 2, 3
- window size=3
- receiver sees no difference in two scenarios!
- incorrectly passes duplicate data as new in (a)





## Window Sizing

- **Question**
- What relationship between window size and sequence number space?
  
- The window size must be less than or equal to half of the sequence number space



## Window Sizing

- **Performance**
  - The window size should allow the sender to fill the pipe
- **Flow Control**
  - The window size should also avoid buffer overflow at the receiver
  - In a Point-to-Point link the window size can be defined based on
    - Round Trip Time (RTT)
    - Receiver Buffer Size



## Reliable Data Transfer: Summary

- Error detection (e.g., CRC)
- Acknowledgements (ACKs)
- Negative Acknowledgements (NAKs)
- Retransmission
- Sequence Number
- Retransmission Timer (Timeout)
- Pipelining (window)



## Direct Connection Networks

- Introduction
- Error detection and correction
- Reliable Data Transfer
- PPP
- Multiple access protocols
- Local Area Networks (LAN)
- Ethernet



## Point to Point Data-Link Protocols

- one sender, one receiver, one link
  - e.g., dialup link, ISDN line, ADSL, ...
- Popular point-to-point DLC protocols:
  - SLIP (Serial Link IP)
  - PPP (Point-to-Point Protocol)
  - HDLC: High level Data Link Control
    - Data Link used to be considered "high layer" in protocol stack!



## SLIP

- Ideato nel 1984 (RFC 1055)
  - Per interconnettere SUN ws a Internet tramite rete telefonica
- Nessuna gestione degli errori
  - I livelli superiori devono farsene carico
- Supporta solo IP
- Assegnazione statica di indirizzi IP
  - Data la limitatezza degli indirizzi IP è un grosso limite
- Nessuna autenticazione
  - Va bene per linee dedicate ma non per collegamenti telefonici
- Molte versioni (spesso incompatibili)
  - Non è uno standard Internet approvato



## Point to Point Protocol (PPP) [RFC 1547]

- ❑ **packet framing:** encapsulation of network-layer datagram in Data Link frame
  - carry network layer data of any network layer protocol (not just IP) *at same time*
  - ability to demultiplex upwards
- ❑ **bit transparency:** must carry any bit pattern in the data field
- ❑ **error detection** (no correction)
- ❑ **connection liveness:** detect and signal link failure to network layer
- ❑ **network layer address negotiation:** endpoint can learn/configure each other's network address



## PPP non-requirements

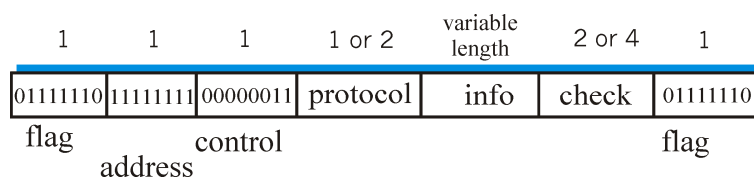
- ❑ No error correction/recovery
- ❑ No flow control
- ❑ Possible out of order delivery
- ❑ No support for point-to-multi-point communication
  - Other DL protocols supports this feature (e.g., HDLC)

**Error recovery, flow control, data re-ordering  
all relegated to higher layers (e.g., TCP)!**



## PPP Data Frame

- **Flag:** delimiter (framing)
- **Address:** does nothing (only one option)
- **Control:** does nothing; in the future possible multiple control fields
- **Protocol:** upper layer protocol to which frame delivered (eg, PPP-LCP, IP, IPCP, etc)

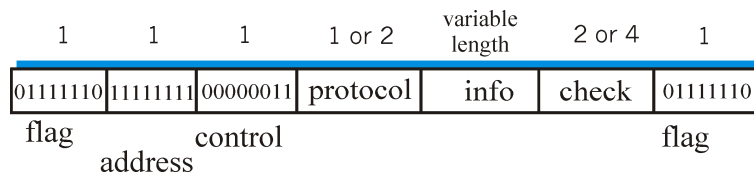


Direct Connection Networks 63



## PPP Data Frame

- **info:** upper-layer data being carried
- **check:** cyclic redundancy check for error detection



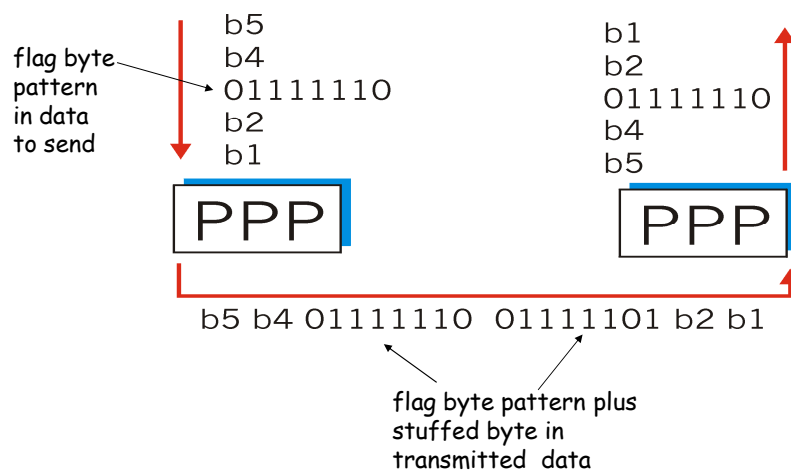
Direct Connection Networks 64



## Byte Stuffing

- "data transparency" requirement: data field must be allowed to include flag pattern <01111110>
  - Q: is received <01111110> data or flag?
- **Sender:**
  - adds ("stuffs") extra < 01111101 > byte after each <01111110> *data* byte
    - < 01111101 > byte = *escape byte*
- **Receiver:**
  - *Whenever receives 01111101* 01111110 discards the escape byte

## Byte Stuffing/Unstuffing





## Byte Stuffing/Unstuffing (More)

- Sender (byte stuffing)
  - 01111110 → 01111101 01111110
  - 01111101 → 01111101 01111101
- Receiver (byte unstuffing)
  - 01111101 01111110 → 01111110
  - 01111101 01111101 → 01111101



## PPP Data Control Protocol

Before exchanging network-layer data, Data Link peers must

- **configure PPP link** (max. frame length, authentication)
  - Through Link Control Protocol (LCP)
- **learn/configure network** layer information
  - for IP: carry IP Control Protocol (IPCP) msgs (protocol field: 8021) to configure/learn IP address



## Esempio: Attivazione di una connessione PPP via modem

1. Il PC chiama il router del provider via modem
2. Il modem del provider risponde
  - Si stabilisce un collegamento fisico tra PC e router del provider
3. Negoziazione dei parametri di link (protocollo LCP)
  - Utilizzo dei campi Address e Control, Lunghezza max frame, Protocollo di autenticazione,
4. Negoziazione parametri di rete
  - Compressione pacchetti IP?, ...
  - Viene effettuata tramite una serie di pacchetti IPCP (inviati mediante frame PPP)
5. Viene assegnato un indirizzo IP al PC
6. Il PC è ora collegato a Internet



## Esempio: Chiusura di una connessione PPP

1. Protocollo IPCP
  - Rilascio dell'indirizzo IP
  - Rilascio della connessione di livello rete
2. Protocollo LCP
  - Rilascio della connessione di livello Direct Connection Networks
3. Viene rilasciato il collegamento telefonico



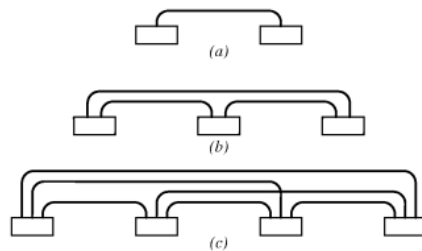
## Direct Connection Networks

- Introduction
- Error detection and correction
- Reliable Data Transfer
- PPP
- **Multiple access protocols**
- Local Area Networks (LAN)
- Ethernet

Direct Connection Networks 71



## Limits of Point-to-Point Links



*N*: Number of Nodes

Required number of links

$$\frac{N(N-1)}{2}$$

**Doesn't scale!!**

## Multiple Access Links and Protocols

Two types of "links":

- point-to-point links
  - PPP protocol
  - HDLC protocol
- **broadcast** (shared wire or medium)
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN



shared wire (e.g., cabled Ethernet)



shared RF (e.g., 802.11 WiFi)



shared RF (satellite)



humans at a cocktail party (shared air, acoustical)

## Multiple Access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - **collision** if node receives two or more signals at the same time

### Multiple Access Protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination



## Ideal Multiple Access Protocol

### Broadcast channel of rate $R$ bps

1. Fully Utilization
  - when one node wants to transmit, it should send at rate  $R$
2. Fairness
  - when  $M$  nodes want to transmit, each should send at average rate  $R/M$
3. Fully decentralization
  - no special node to coordinate transmissions
  - no synchronization of clocks, slot assignment, ...
4. Simplicity



## MAC Protocols: a taxonomy

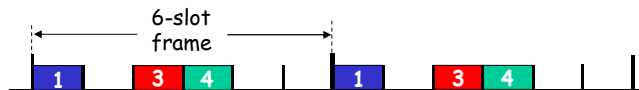
Three broad classes:

- **Channel Partitioning**
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- **Random Access**
  - channel not divided, allow collisions
  - "recover" from collisions
- **"Taking turns"**
  - nodes take turns, but nodes with more to send can take longer turns

## Channel Partitioning MAC protocols: TDMA

### TDMA: Time Division Multiple Access

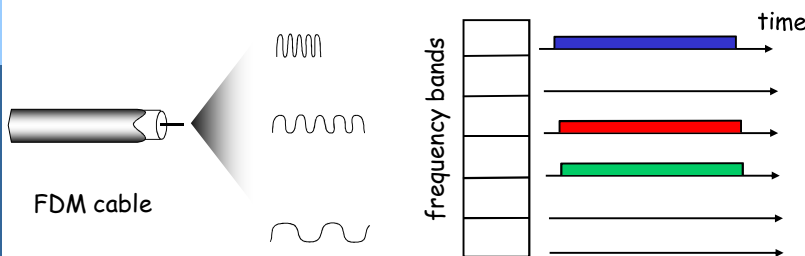
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



## Channel Partitioning MAC protocols: FDMA

### FDMA: Frequency Division Multiple Access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



## Channel Partitioning MAC protocols: CDMA

### CDMA: Code Division multiple access

- ❑ Each pair of nodes assigned with different code
  - Unique code used to encode transmitted data
- ❑ Simultaneous transmissions
- ❑ Each receiver can correctly decode
  - In spite of interferences from other nodes
- ❑ Mainly used in military applications and cellular telephony



## Random Access Protocols

- ❑ When node has packet to send
  - transmit at full channel data rate  $R$ .
  - no *a priori* coordination among nodes
- ❑ two or more transmitting nodes → "collision",
- ❑ **random access MAC protocol** specifies:
  - how to avoid/detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- ❑ Examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA



## Slotted ALOHA

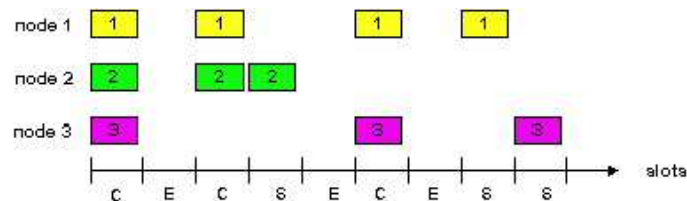
### Assumptions:

- ❑ all frames same size
- ❑ time divided into equal size slots (time to transmit 1 frame)
- ❑ nodes start to transmit only at slot beginning
- ❑ nodes are synchronized
- ❑ if 2 or more nodes transmit in the same slot, all nodes detect collision

### Operation:

- ❑ when node obtains fresh data, transmits in next slot
  - if no collision: node can send new frame in next slot
  - if collision: node retransmits frame in each subsequent slot with probability  $p$  until success

## Slotted ALOHA



### Pros

- ❑ single active node can continuously transmit at full rate of channel
- ❑ highly decentralized: only nodes need to be in sync
- ❑ simple

### Cons

- ❑ collisions, wasting slots
- ❑ idle slots
- ❑ nodes may be able to detect collision in less than time to transmit packet
- ❑ clock synchronization

## Slotted Aloha efficiency

**Efficiency** : long-run fraction of successful slots (many nodes, all with many frames to send)

- suppose:  $N$  nodes with many frames to send, each transmits in slot with probability  $p$
- prob that given node has success in a slot =  $p(1-p)^{N-1}$
- prob that *any* node has a success =  $Np(1-p)^{N-1}$

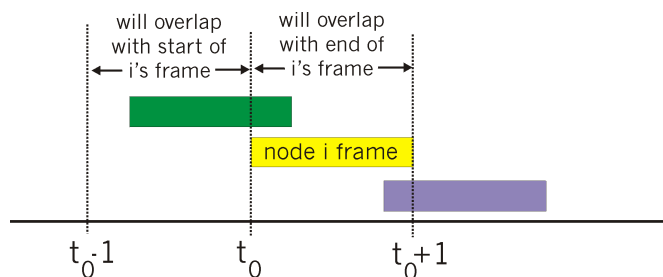
- max efficiency: find  $p^*$  that maximizes  $Np(1-p)^{N-1}$
- for many nodes, take limit of  $Np^*(1-p^*)^{N-1}$  as  $N$  goes to infinity, gives:  
Max efficiency =  $1/e = .37$

**At best:** channel used for useful transmissions 37% of time!



## Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives transmit immediately
  - After a collision transmit with probability  $p$
- collision probability increases:
  - frame sent at  $t_0$  collides with other frames sent in  $[t_0-1, t_0+1]$





## Pure Aloha efficiency

$P(\text{success by given node}) = P(\text{node transmits}) \cdot$

$P(\text{no other node transmits in } [t_0-1, t_0]) \cdot$

$P(\text{no other node transmits in } [t_0, t_0+1])$

$$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$$

$$= p \cdot (1-p)^{2(N-1)}$$

... choosing optimum  $p$  and then letting  $n \rightarrow \infty$  ...

$$= 1/(2e) = .18$$

*even worse than slotted Aloha!*



## CSMA (Carrier Sense Multiple Access)

**CSMA:** listen before transmit:

If channel sensed idle: transmit entire frame

If channel sensed busy, defer transmission

human analogy: don't interrupt others!

## CSMA collisions

**collisions can still occur:**

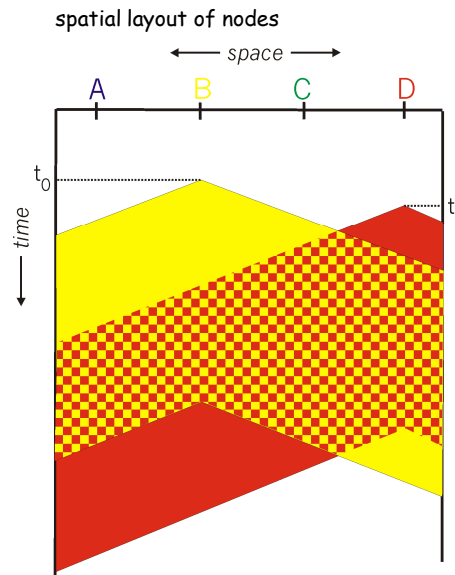
propagation delay means  
two nodes may not hear  
each other's transmission

**collision:**

entire packet transmission  
time wasted

**note:**

role of distance & propagation  
delay in determining collision  
probability

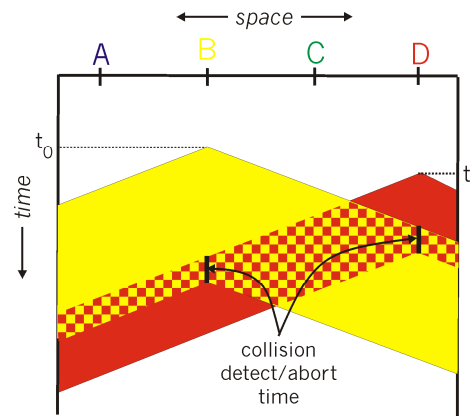


## CSMA/CD (Collision Detection)

**CSMA/CD:** carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage
- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
- human analogy: the polite conversationalist

## CSMA/CD collision detection



## “Taking Turns” MAC protocols

### channel partitioning MAC protocols:

- share channel efficiently and *fairly* at high load
- *inefficient at low load*: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

### Random access MAC protocols

- *efficient at low load*: single node can fully utilize channel
- high load: *collision* overhead

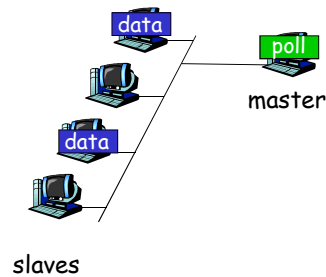
### “taking turns” protocols

look for best of both worlds!

## “Taking Turns” MAC protocols

### Polling:

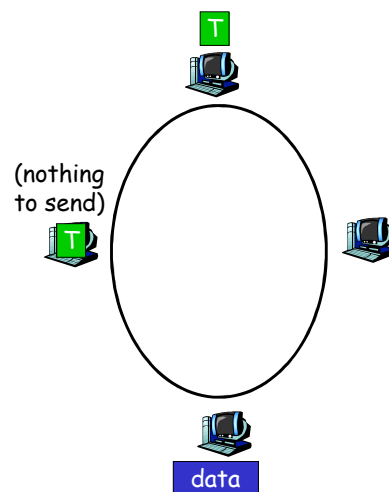
- ❑ master node  
“invites” slave nodes to transmit in turn
- ❑ typically used with “dumb” slave devices
- ❑ concerns:
  - polling overhead
  - latency
  - single point of failure (master)



## “Taking Turns” MAC protocols

### Token passing:

- ❑ control **token** passed from one node to next sequentially.
- ❑ token message
- ❑ concerns:
  - token overhead
  - latency
  - single point of failure (token)





## Summary of MAC protocols

- *channel partitioning*, by time, frequency or code
  - Time Division, Frequency Division, CDMA
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - Bluetooth, FDDI, IBM Token Ring



## Addressing

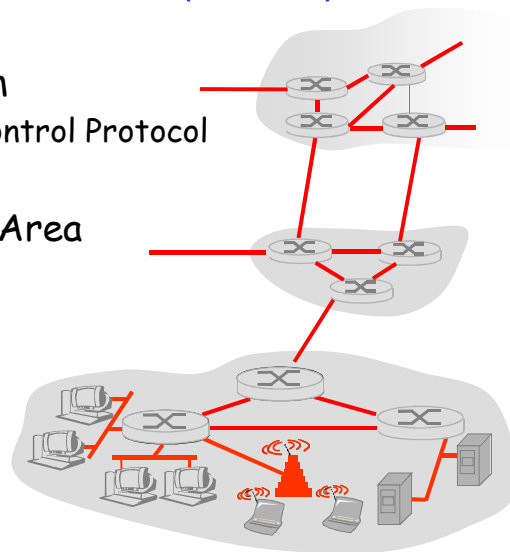
- **Hardware Address**
  - Also called Physical address, link-layer address, or MAC address:
  - function: *get frame from one interface to another physically-connected interface (same network)*
- **IEEE Addressing Scheme (LAN)**
  - 48 bit link-layer address
    - burned in NIC ROM, also sometimes software settable

## Direct Connection Networks

- Introduction
- Error detection and correction
- Reliable Data Transfer
- PPP
- Multiple access protocols
- Local Area Networks (LAN)
- Ethernet

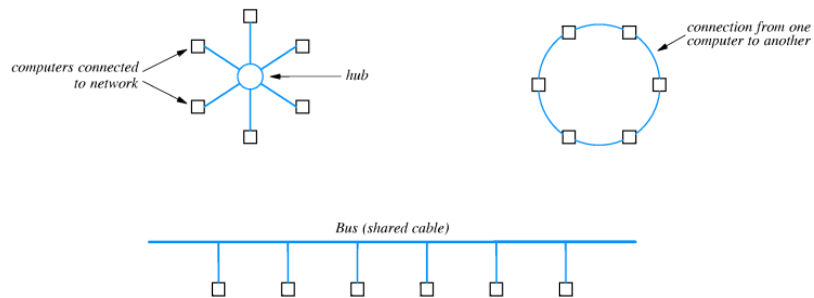
## Local Area Networks (LANs)

- Broadcast Medium
  - Medium Access Control Protocol
  - MAC addressing
- Limited Coverage Area
  - Building, Campus
- High Bit Rate
  - 10 Mbps - 10 Gbps





# Local Area Networks (LANs)

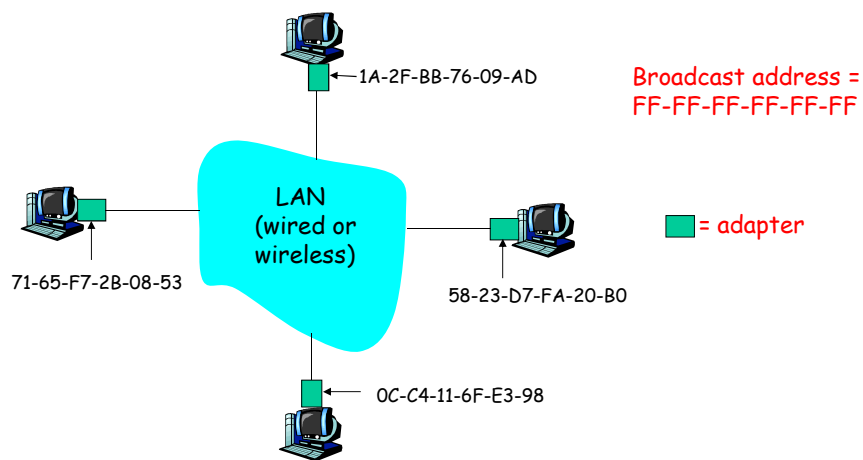


## □ Broadcast Medium

- Medium Access Control (MAC) Protocol for channel access
- MAC addressing

# MAC Addresses

Each adapter on LAN has unique MAC address





## MAC Address (more)

- ❑ MAC address allocation administered by IEEE
- ❑ manufacturer buys portion of MAC address space (to assure uniqueness)
- ❑ MAC flat address → portability
  - can move LAN card from one LAN to another
- ❑ IP hierarchical address NOT portable
  - address depends on IP subnet to which node is attached
- ❑ analogy:
  - (a) MAC address: like Social Security Number
  - (b) IP address: like postal address



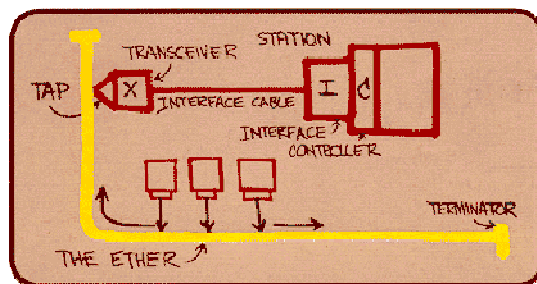
## Direct Connection Networks

- ❑ Introduction
- ❑ Error detection and correction
- ❑ Reliable Data Transfer
- ❑ PPP
- ❑ Multiple access protocols
- ❑ Local Area Networks (LAN)
- ❑ Ethernet

# Ethernet

"dominant" wired LAN technology:

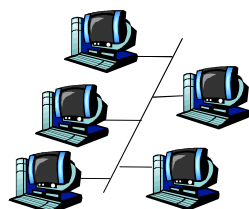
- ❑ cheap \$20 for NIC
- ❑ first widely used LAN technology
- ❑ simpler, cheaper than token LANs and ATM
- ❑ kept up with speed race: 10 Mbps - 10 Gbps



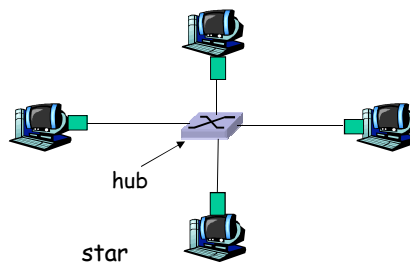
Metcalfe's Ethernet sketch

# Topologies

- ❑ bus topology
  - Based on a bus
  - all nodes in same collision domain (can collide with each other)
- ❑ star topology
  - Based on a central *hub*
  - all nodes in same collision domain (just as in the bus topology)



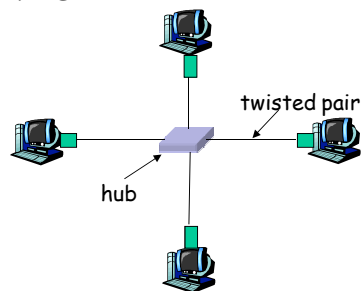
bus: coaxial cable



## Hubs

... physical-layer ("dumb") repeaters:

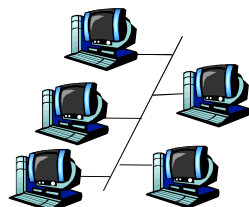
- bits coming in one link go out *all* other links at same rate
- all nodes connected to hub can collide with one another
- no frame buffering
- no CSMA/CD at hub: host NICs detect collisions



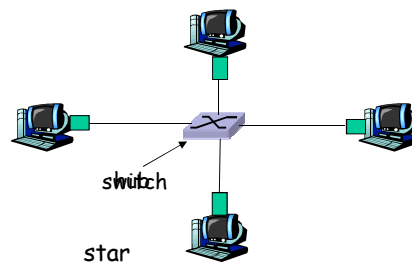
Packet Switched Networks 103

## Switched Star topology

- bus topology popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- today: star topology prevails
  - active *switch* in center
  - each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)



bus: coaxial cable

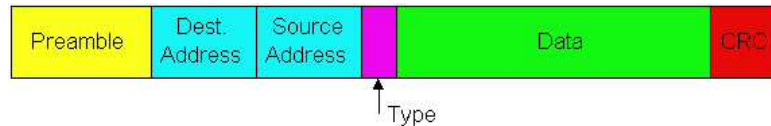


star

Direct Connection Networks 104

## Ethernet Frame Structure

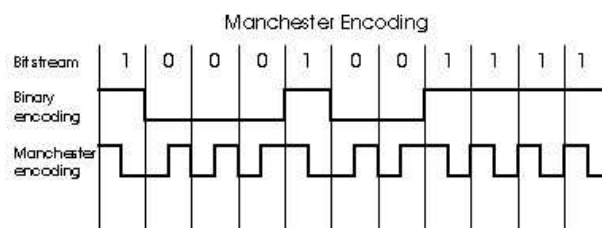
Sending adapter encapsulates IP datagram (or other network layer protocol packet) in **Ethernet frame**



**Preamble (8 bytes):**

- ❑ 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- ❑ used to synchronize receiver, sender clock rates

## Manchester encoding

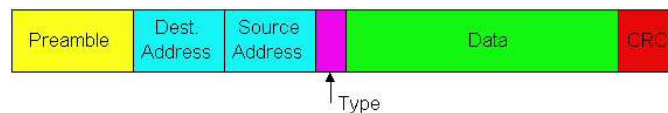


- ❑ used in 10BaseT
- ❑ each bit has a transition
- ❑ allows clocks in sending and receiving nodes to synchronize to each other
  - no need for a centralized, global clock among nodes!
- ❑ Hey, this is physical-layer stuff!



## Ethernet Frame Structure (more)

- **Addresses:** 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- **Type:** indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- **CRC:** checked at receiver, if error is detected, frame is dropped



Direct Connection Networks 107



## Ethernet: Service Type

- **connectionless:** No handshaking between sending and receiving NICs
- **unreliable:** receiving NIC doesn't send acks or nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP
  - otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted **CSMA/CD**

Direct Connection Networks 108



## Ethernet CSMA/CD algorithm

1. NIC receives data from network layer, creates frame
2. If NIC senses channel idle for **96 bits**, starts frame transmission  
If NIC senses channel busy, waits until channel idle for **96 bits**, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends **48-bit jam signal**
5. After aborting, NIC enters **exponential backoff**: after the  $n$ -th collision, NIC chooses  $K$  at random from  $\{0,1,2,\dots,2^m-1\}$ ,  
 $m = \min(n,10)$   
NIC waits  $K \cdot 512$  bit times, returns to Step 2

Direct Connection Networks 109



## Ethernet's CSMA/CD (more)

**Jam Signal:** make sure all other transmitters are aware of collision; 48 bits

**Bit time:** .1 microsec for 10 Mbps Ethernet ;  
for  $K=1023$ , wait time is about 50 msec

**Exponential Backoff:**

- **Goal:** adapt retransmission attempts to estimated current load
  - heavy load: random wait will be longer
- first collision: choose  $K$  from  $\{0,1\}$ ; delay is  $K \cdot 512$  bit transmission times
- after second collision: choose  $K$  from  $\{0,1,2,3\}$ ...
- after ten collisions, choose  $K$  from  $\{0,1,2,3,4,\dots,1023\}$

Direct Connection Networks 110

## CSMA/CD efficiency

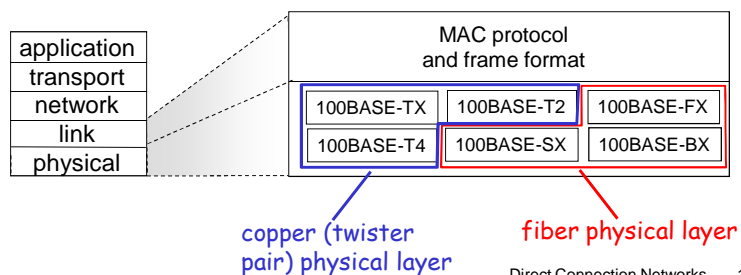
- $T_{prop}$  = max prop delay between 2 nodes in LAN
- $t_{trans}$  = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- efficiency goes to 1
  - as  $t_{prop}$  goes to 0
  - as  $t_{trans}$  goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!

## 802.3 Ethernet Standards: Link & Physical Layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - different physical layer media: fiber, cable







## Summary

- Direct connection networks
- principles behind Data Link layer services:
  - error detection, correction
  - reliable data transfer
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies
  - PPP
  - Ethernet