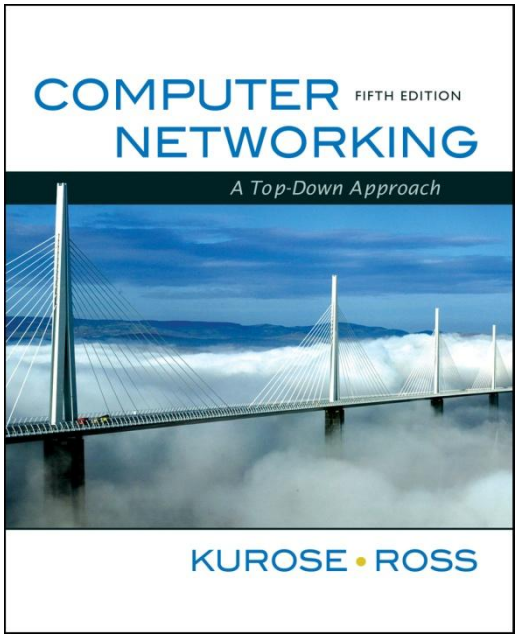
Chapter 5
Link Layer and LANs



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Link Layer

5.1 Introduction and services

5.2 Error detection and correction

5.3Multiple access protocols

New: WiFi

5.4 Link-layer Addressing

5.5 Ethernet

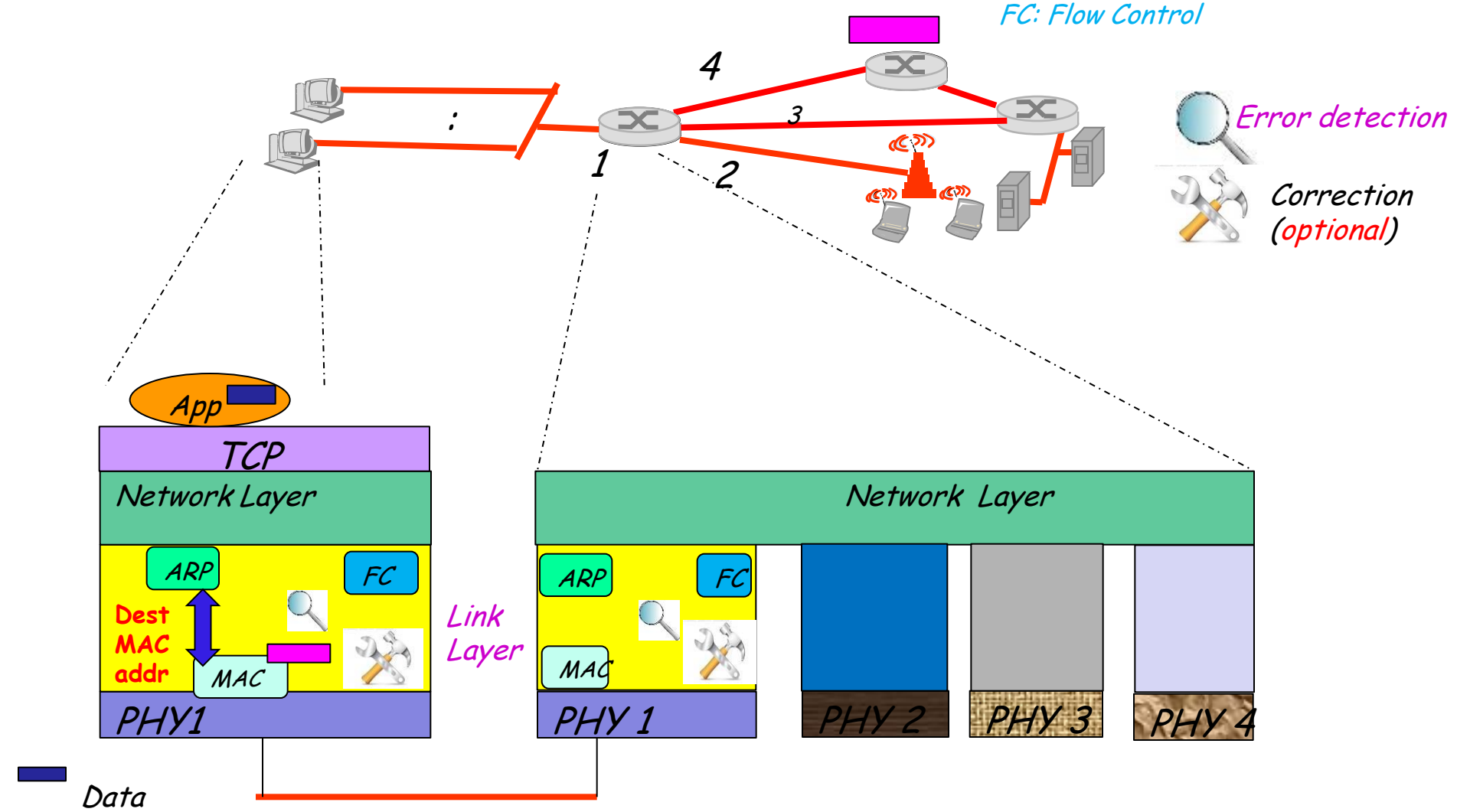
5.6 Link-layer switches

3.4 New: Reliable data transfer + flow control

Link layer: context

MAC: Medium Access Control

FC: Flow Control



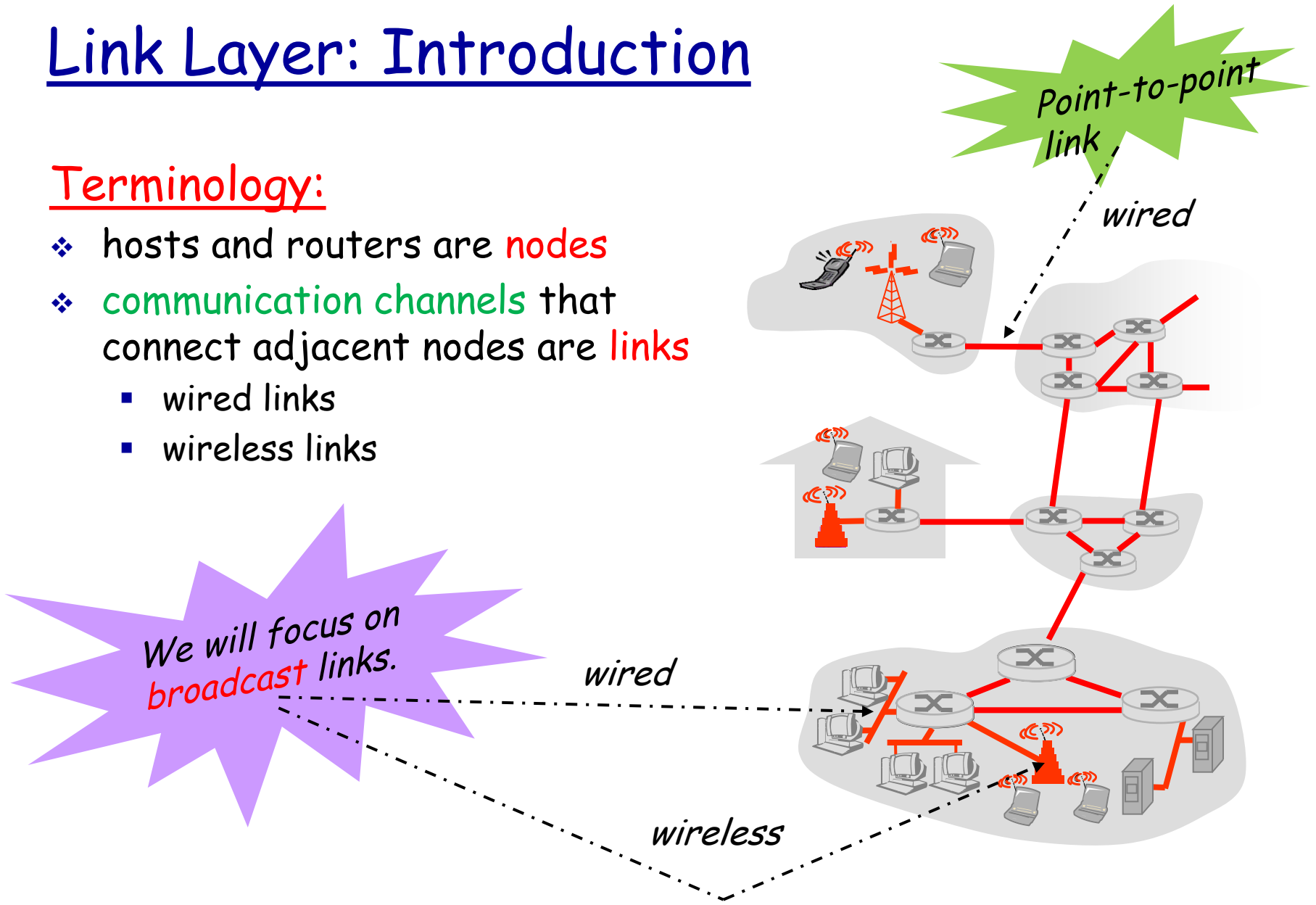
ARP (Address Resolution Protocol): IP address ➔ MAC address
(%ipconfig /all ← for MAC addresses.)

PHY: Physical layer ← hardware

Link Layer: Introduction

Terminology:

- hosts and routers are nodes
- communication channels that connect adjacent nodes are links
  - wired links
  - wireless links

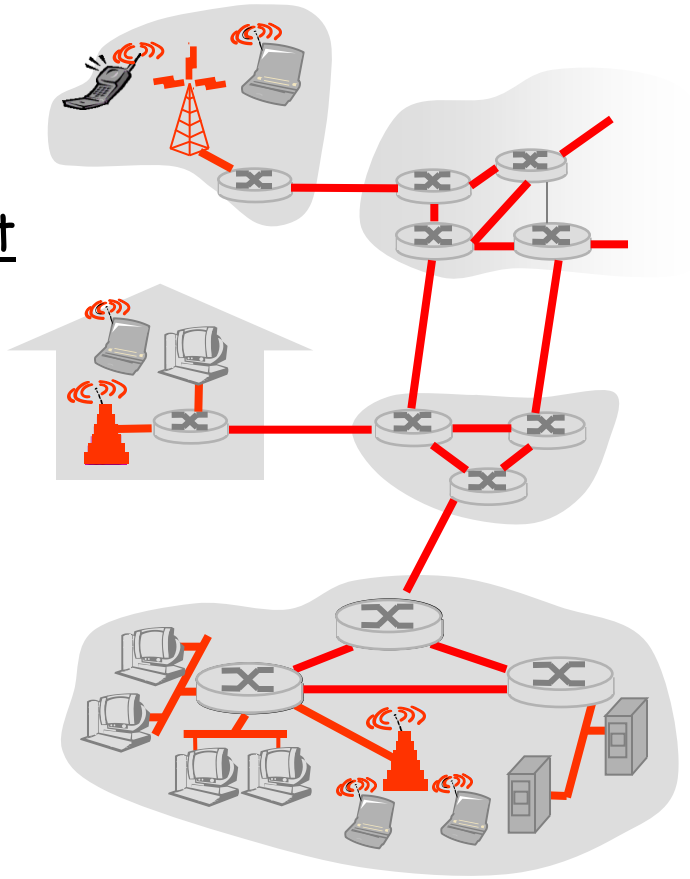


Chapter 5: The Data Link Layer

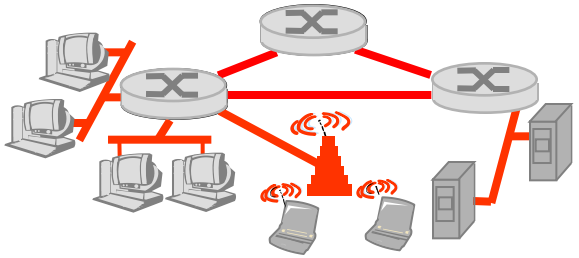
Our goals:

- understand principles behind data link layer services:

1. link access by sharing a broadcast channel: multiple access
   - Instruct the hardware (PHY layer) **when** to transmit (… MAC protocols)
2. link layer addressing



Link Layer Services (more)



- *error detection*:
  - errors caused by signal attenuation and noise
  - receiver detects presence of errors

3

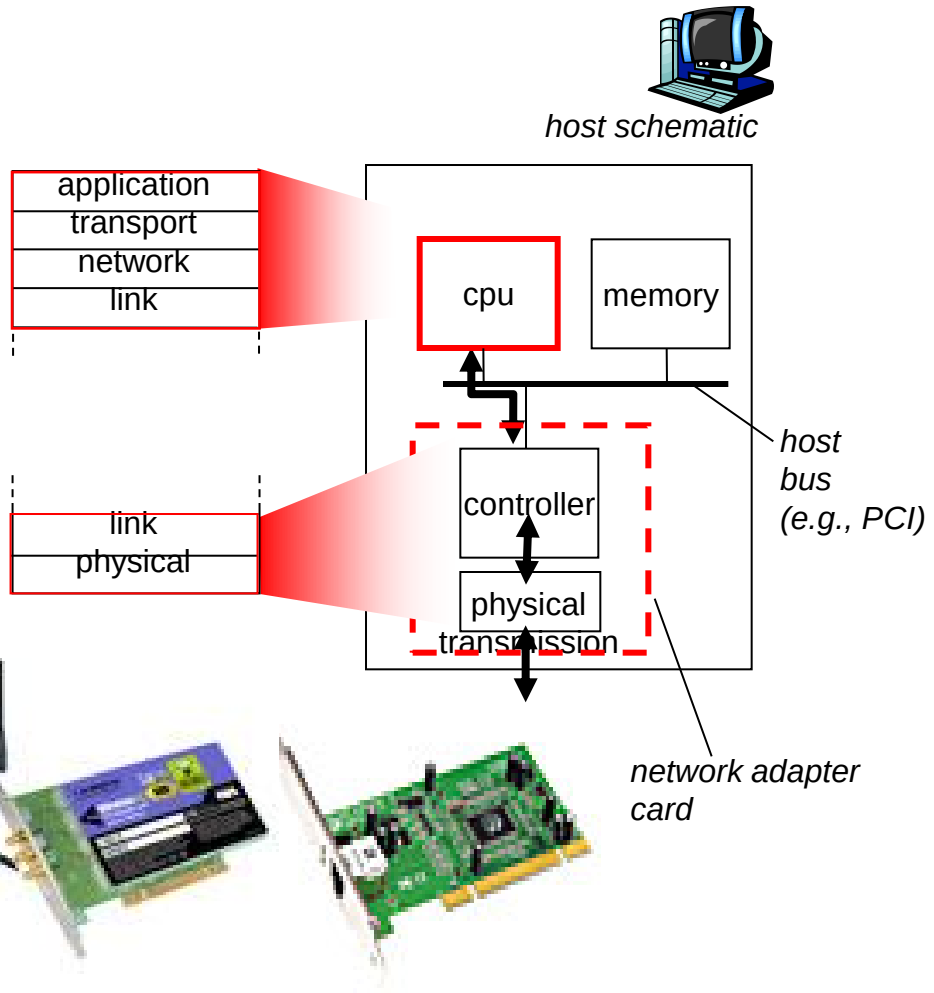
- error correction:
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
  - receiver signals sender for retransmission

4

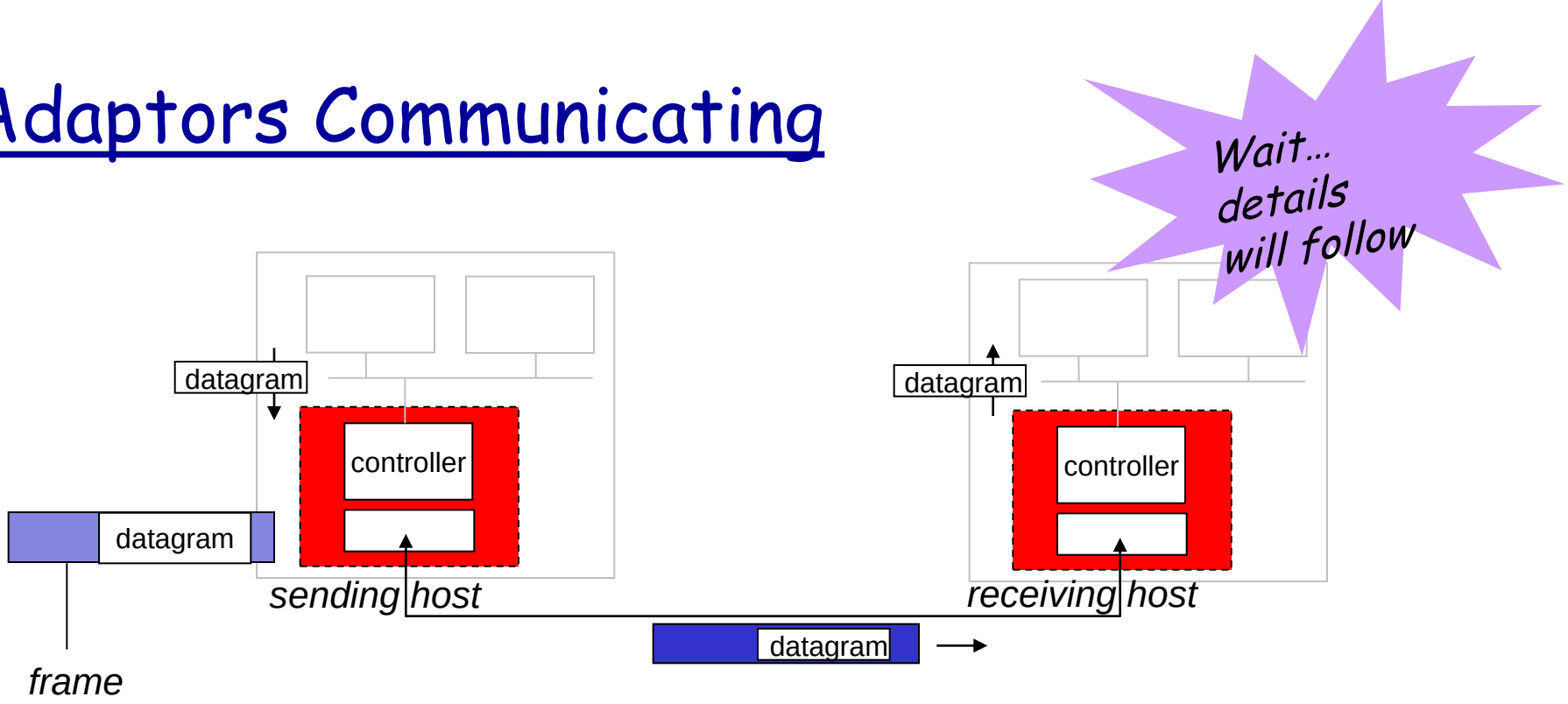
- *flow control:*
  - pacing between (adjacent) sending and receiving nodes

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC)
  - Ethernet card, PCMCI card, 802.11 card
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Adaptors Communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, and sequence # for flow control
- receiving side
  - looks for errors
  - extracts datagram, passes to upper layer at receiving side

Link Layer

5.1 Introduction and services

5.2 Error detection and correction

5.3Multiple access protocols

5.4 Link-layer Addressing

5.5 Ethernet

5.6 Link-layer switches

5.7 PPP

5.8 Link virtualization: MPLS

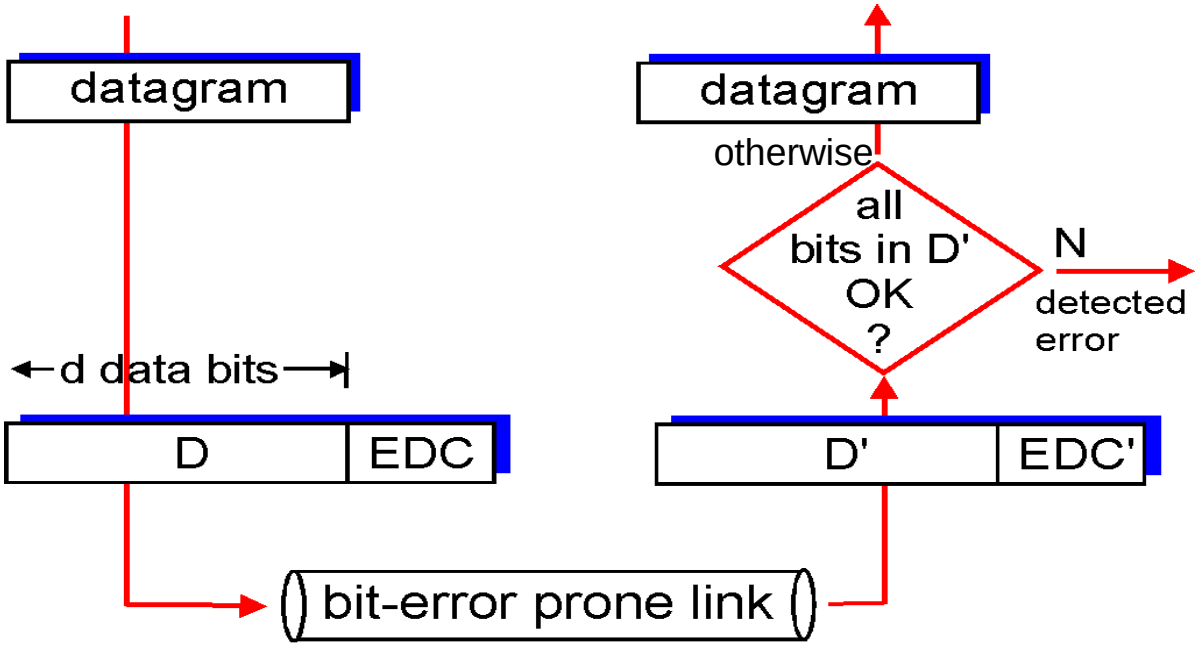
5.9 A day in the life of a web request

3.4 Reliable data transfer

Error Detection

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

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



What are all the *error detection mechanisms* that I will learn

Parity bits (today …)

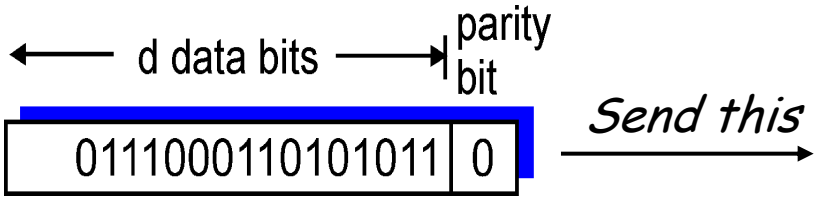
You will learn …

CRC:
Cyclic Redundancy Check (Link layer)

Internet Checksum
(Network Layer, Transport Layer)

Parity Checking

Single Bit Parity:
Detect single bit errors



(Example of odd parity)

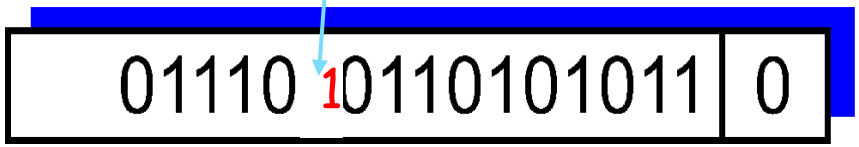
Two kinds of parity:

* **Odd parity:** Set the parity bit to 1 or 0 to make the total # of 1's an odd number.

* **Even parity:** Set the parity bit to 1 or 0 to make the total # of 1's an even number

You find the frame to be erroneous, but you cannot locate the bit error.

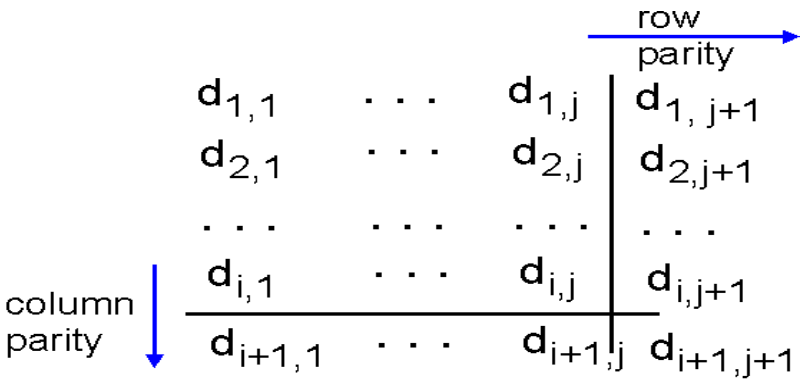
..... But, this is received



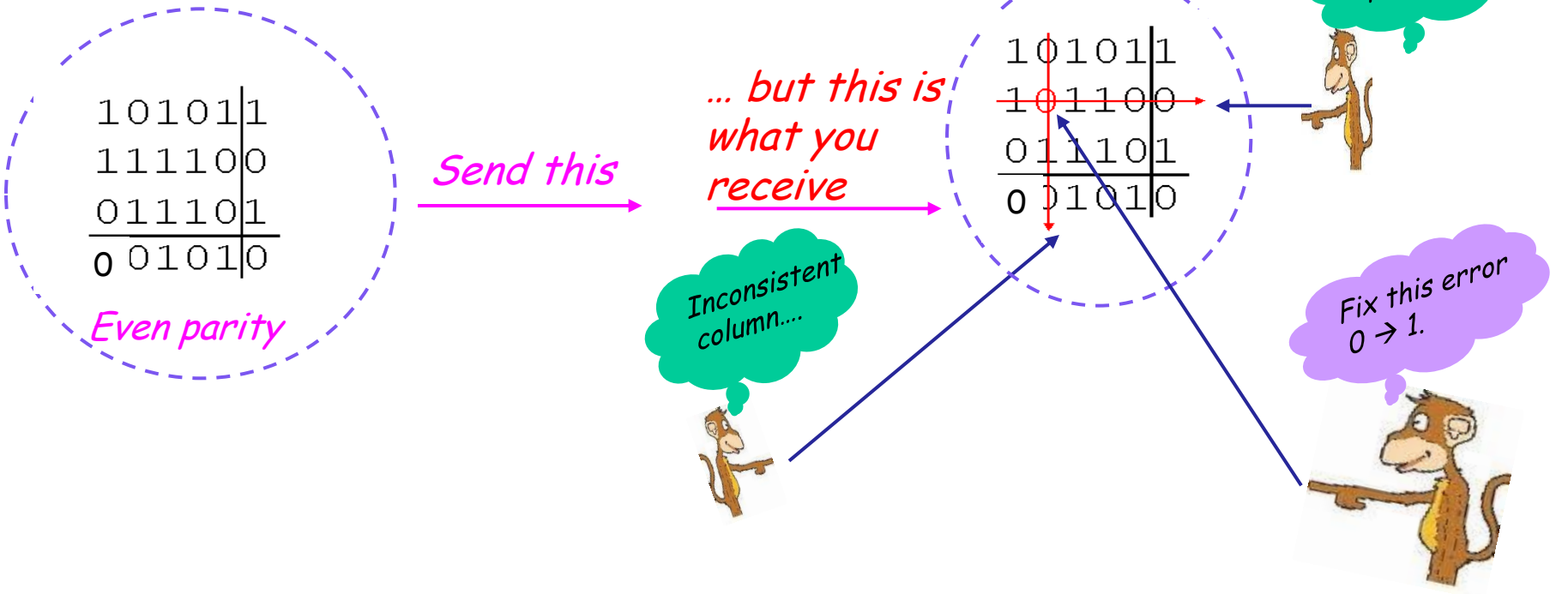
When you compute the parity bit, you know that it should be a '1'.

Use two-dimensional bit parity to **detect and correct single bit errors**

Parity Checking (Two dimensional)



Detect ***and*** correct **single bit errors**



Applications of Parity Checking

Where is parity checking used?

Not generally in network protocols these days...

But, it is used here and here and here ....

RAID: Redundant Array of Inexpensive Drives

DES: Data Encryption Standard for symmetric-key cryptography

Media access is a noisy process..... because of imperfect fabrication and hardware aging.

What kinds of questions can be asked on the exams?

ECE358 Mid-term S'12

For the following **four bytes** of data, compute the two dimensional *odd* parity bits.

1 1 0 0 1 1 0 1 |
1 0 1 1 1 0 1 1 |
0 1 1 0 1 0 1 1 |
1 0 0 0 0 1 0 1 |

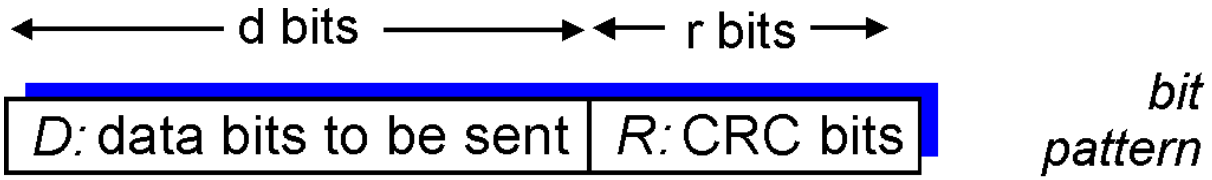
ECE358 Mid-term F'12

For the data and parity bits generated above, *introduce four bit errors* that the *receiver* will not be able to detect…

Checksumming: Cyclic Redundancy Check

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- **goal:** choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)

  ← zero remainder
- receiver knows G:
  - Divides <D,R> by G. If non-zero remainder: error detected!
- can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



$D * 2^r$ XOR R

mathematical formula

CRC Example:

Want:

$D \cdot 2^r$ XOR R = nG

equivalently:

$D \cdot 2^r$ = nG XOR R

equivalently:

if we divide $D \cdot 2^r$ by G, we get remainder R

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

Properties of XOR:

A XOR A = 0

A XOR 0 = A

(A XOR B) XOR C = A XOR (B XOR C)

Process: Calculation of CRC

If the **input** bit above the leftmost divisor bit is 1,
the divisor is XORed into the input.
Else (the input bit above the leftmost divisor bit is 0) do nothing.
The divisor is then shifted one bit to the right (➔)
The process is repeated until
the divisor reaches the right-hand end of the input row.

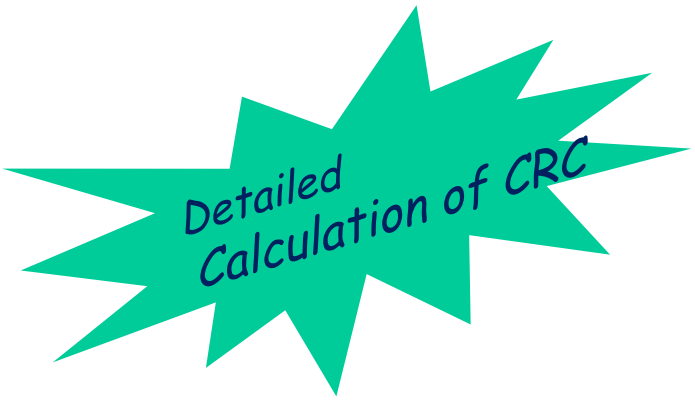
Input: $D.2^r$ ⟶
Divisor: G ⟶
Align input and divisor on MSB

```
1 0 1 1  1 0  0 0 0
1 0 0 1
-----------------
0 0 1 0 10  0 0 0
  1 00 1
-----------------
0 0 1 0 10  0 0 0
    1 0 01
-----------------
0 0 0 0 11  0 0 0
      1 00  1
-----------------
0 0 0 0 11  0 0 0
        10  0 1
-----------------
0 0 0 0 0 1  0 1 0
          1  0 0 1
-----------------
0 0 0 0 0 0 0 1 1
```

R

D = 101110

G = 1001



What questions can be asked on the exams?

ECE358 Mid-term S'12

For the data bits D = 11001101 and generator bits G = 1011, compute the CRC bits.

What is the disadvantage of making the CRC field as long as the data portion of a frame?

D.2^r: 1 1 0 0 1 1 0 1 0 0 0 ?
G: 1 0 1 1

What is the length of the CRC field?

Link Layer

5.1 Introduction and services

5.2 Error detection and correction

5.3Multiple access protocols

5.4 Link-layer Addressing

5.5 Ethernet

5.6 Link-layer switches

5.7 PPP

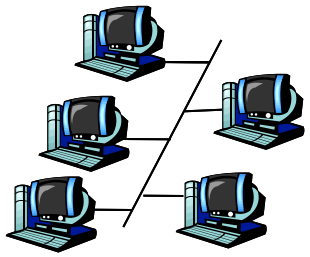
5.8 Link virtualization: MPLS

5.9 A day in the life of a web request

Multiple Access Links and Protocols

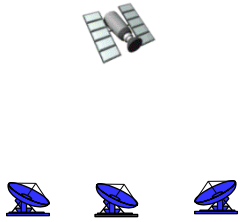
Two types of "links":

- point-to-point
  - PPP for dial-up access
- broadcast (shared wire or medium)
  - old-fashioned Ethernet
  - 802.11 wireless LAN



shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)

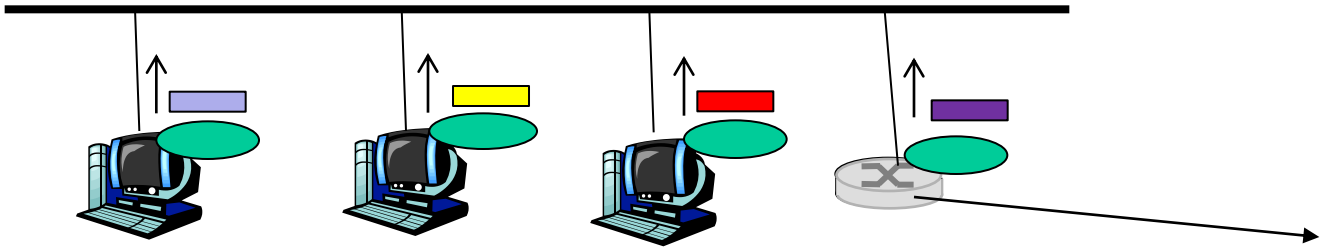


shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

Multiple Access protocols

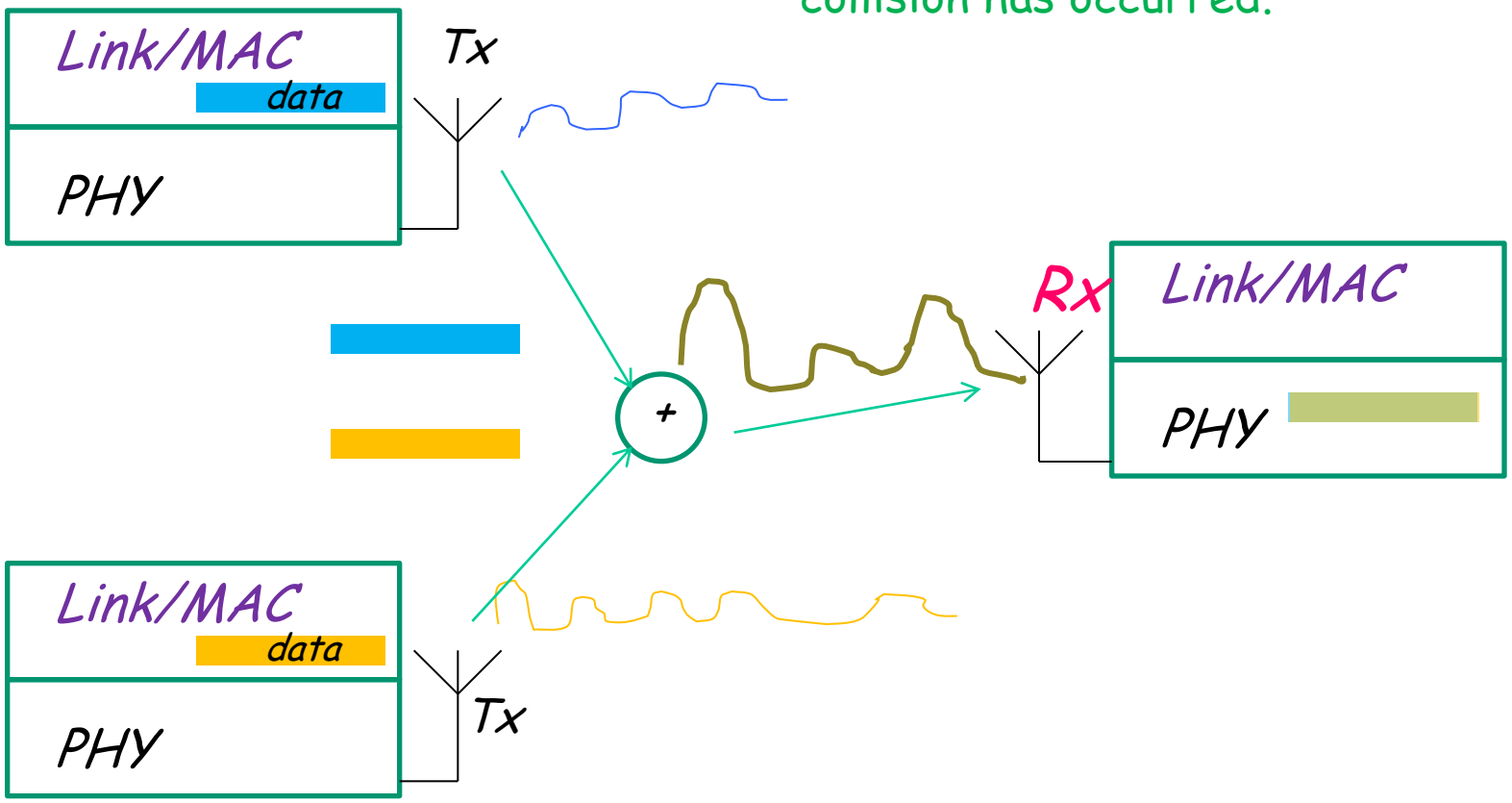
- single shared broadcast channel
- two or more simultaneous transmissions cause collision
- *multiple access protocol*
  - distributed algorithm that determines **when** a node can transmit



Packet Collision

Note

Packet collision occurs at the receiver, but transmitters must know that collision has occurred.



Ideal Multiple Access Protocol

Broadcast channel of rate R bps

1. when one node wants to transmit, it can send at rate R.

2. when M nodes want to transmit, each can send at average rate R/M

3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots

4. simple (plug-and-play, no complex hardware, …)

MAC Protocols: a taxonomy

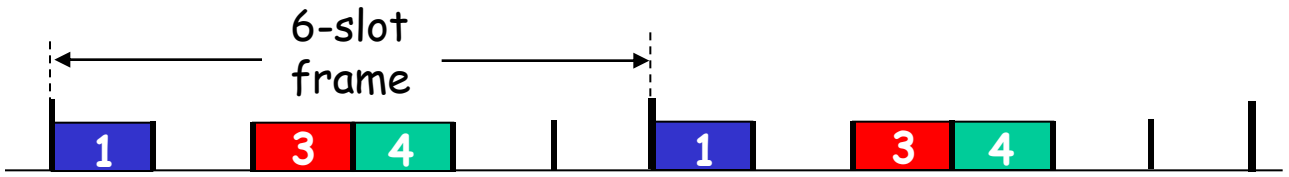
Three broad classes:

- Channel Partitioning (Commonly done in cellular networks)
  - divide channel into smaller "pieces" (time slots, frequency)
  - allocate piece to node for exclusive use
- Random Access
  - channel not divided, allow collisions
  - "recover" from collisions
- "Taking turns"
  - nodes take turns

Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

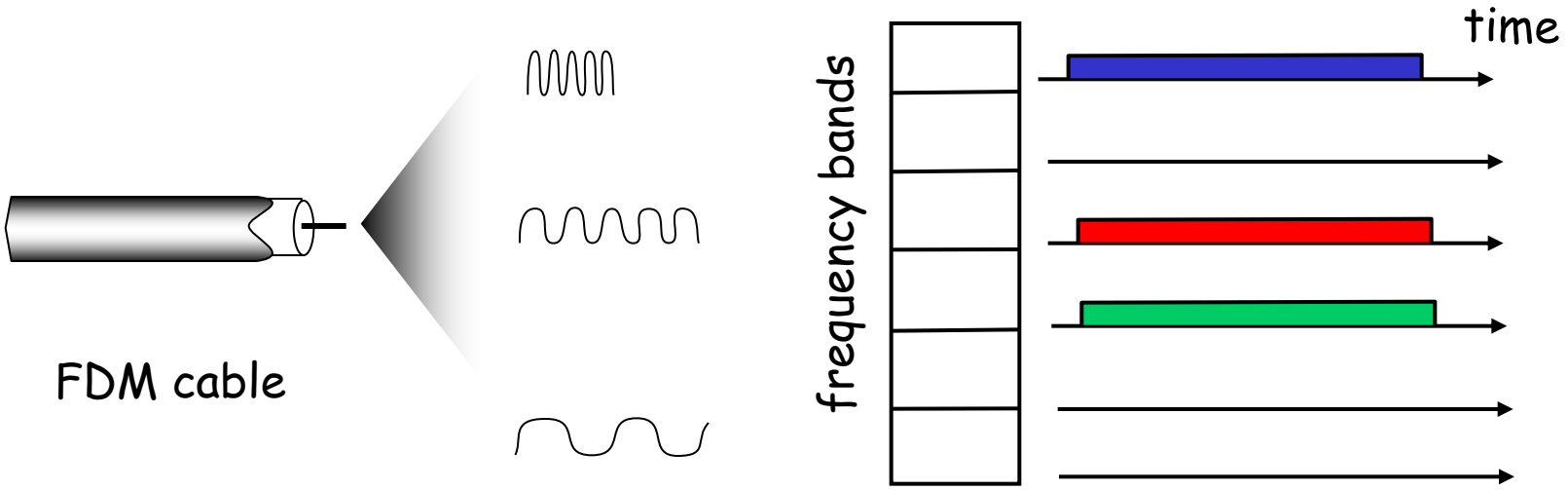
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



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



Random Access Protocols

- When node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes (…partial exception in WiFi)
- two or more transmitting nodes ➜ "collision",
- random access MAC protocol specifies:
  - **how** to detect collisions
  - **how** to recover from collisions
- Examples of random access MAC protocols:
  - ALOHA and slotted ALOHA
  - CSMA/CD, CSMA/CA

CSMA: Carrier Sense Multiple Access

  - CD: Collision **Detection** ← **Reactive scheme**
  - CA: Collision **Avoidance** ← **Preventive scheme**

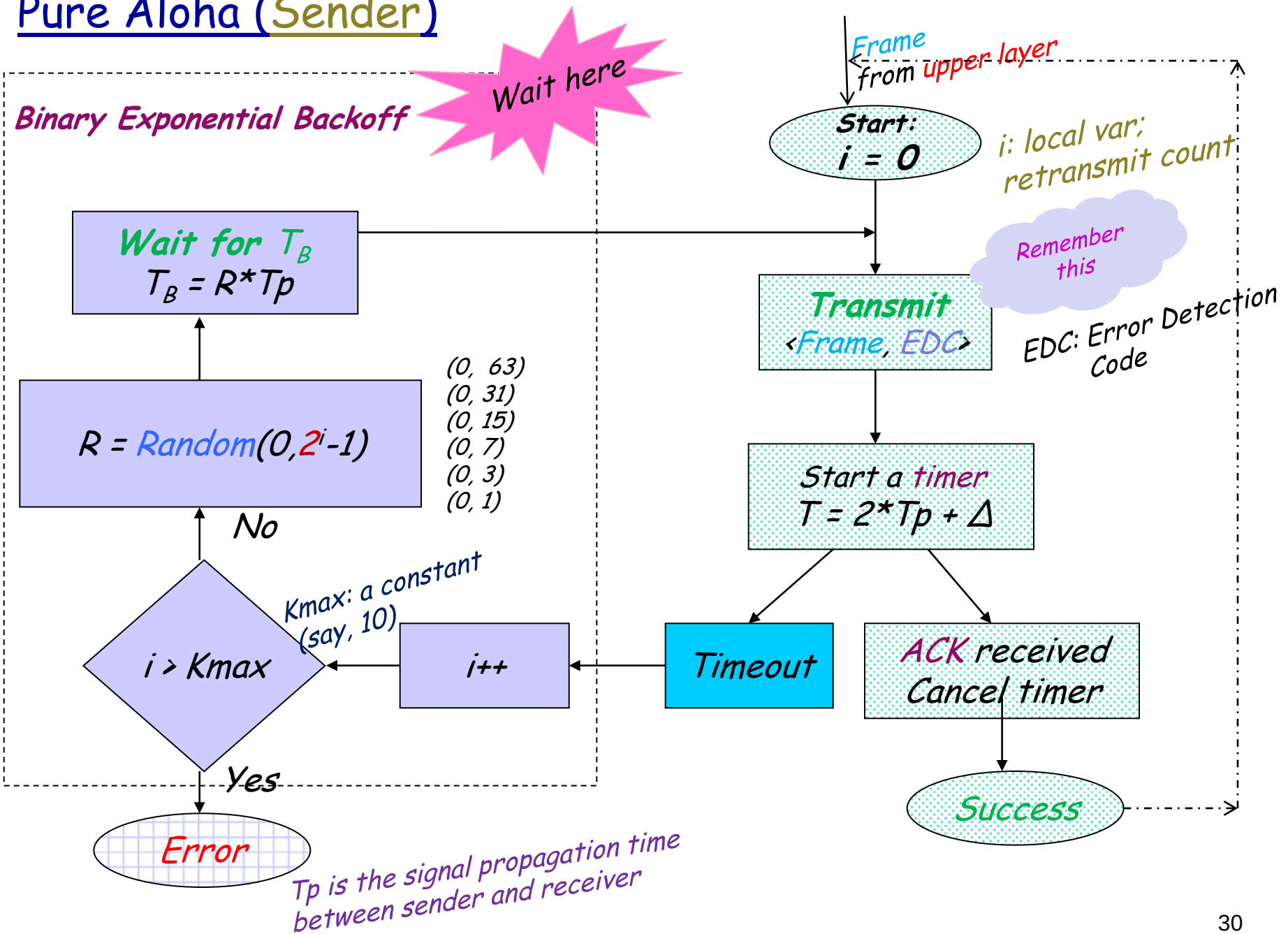
Aloha Protocol

- Developed in the 1970s at U of Hawaii
- To interconnect terminals with mainframes
- LAN/ WLAN: Possible, but not used

- GSM: Cell phones use this protocol to **request a channel** from the base stations

- Two types
  - Pure Aloha (Continuous time)
  - Slotted Aloha

Pure Aloha (Sender)



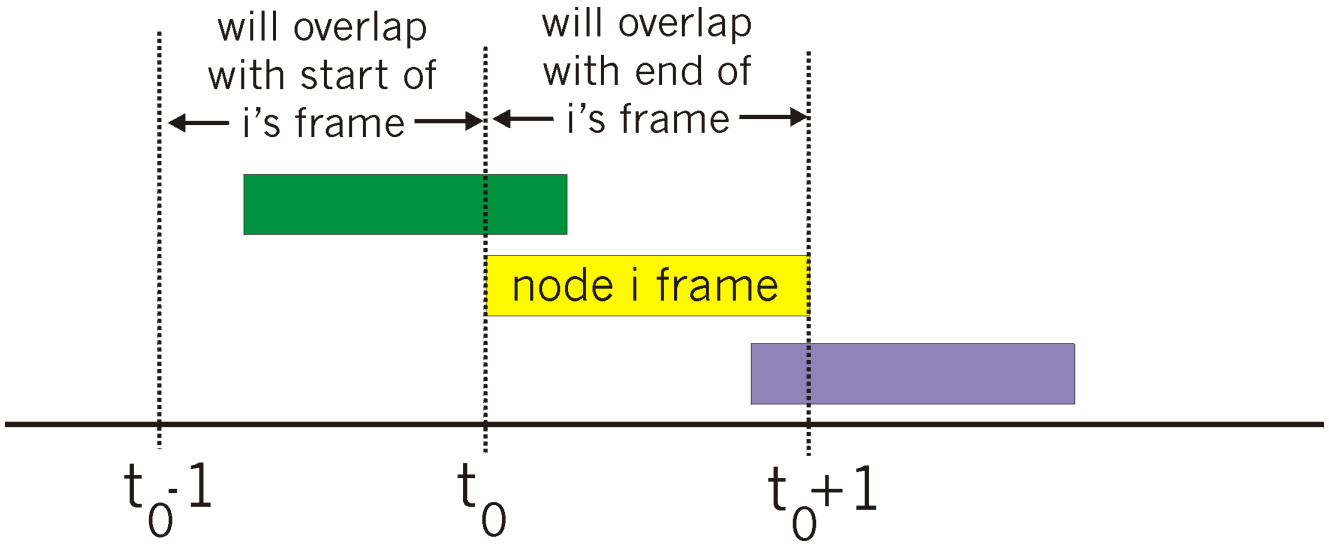
Pure Aloha

What does the receiver do?

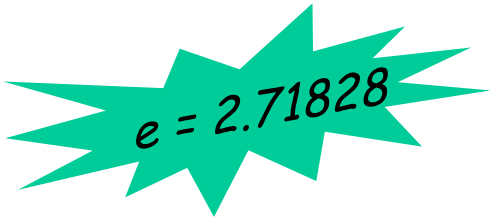
OK…
This is my homework…

Pure (unslotted) ALOHA

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



Pure Aloha efficiency



Throughput of Pure Aloha =

Total input rate (G) * Prob. of successful Packet Trans.

$= G * e^{-2G}$

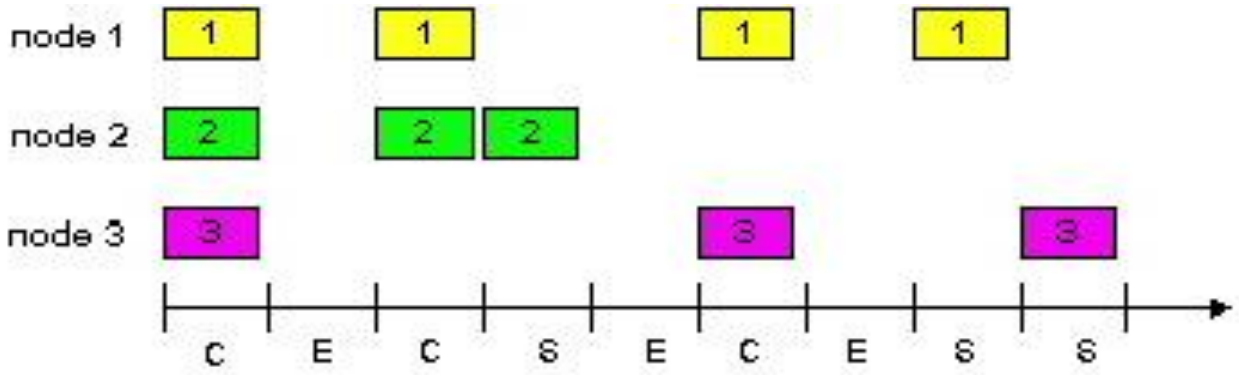
G: Number of packets in X sec.

where X is the packet Tx time.

Max throughput occurs at G = 0.5.

Max Throughput = 1/(2e) = .18

Slotted ALOHA



Assumptions:

- all frames are same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only at slot beginning
- nodes are synchronized

Operation:

- when node obtains fresh frame, transmit in next slot
- if Tx is not successful, retransmit in following slots after some wait..

Slotted ALOHA

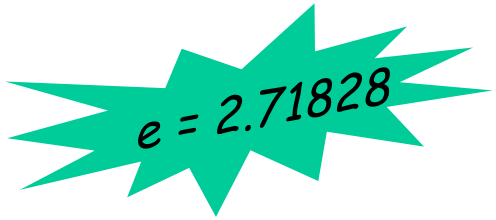
Pros

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

Cons

- collisions, wasting slots
- idle slots
- Keep transmitting even if there is collision ….
- clock synchronization among nodes ….

Slotted Aloha efficiency



Throughput of Slotted Aloha =

Total input rate (G) * Prob. of successful Packet Trans.

$= G * e^{-G}$

G: Number of packets in X sec.

where X is the packet Tx time.

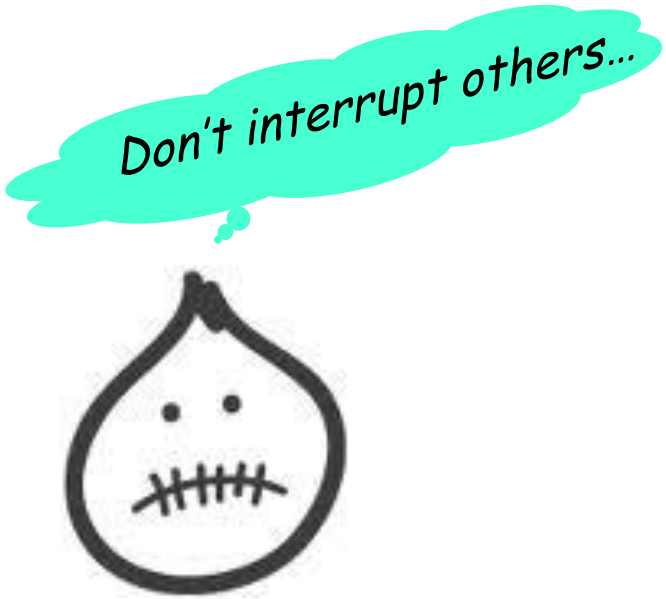
Max throughput occurs at G = 1.

Max Throughput = 1/e = .37

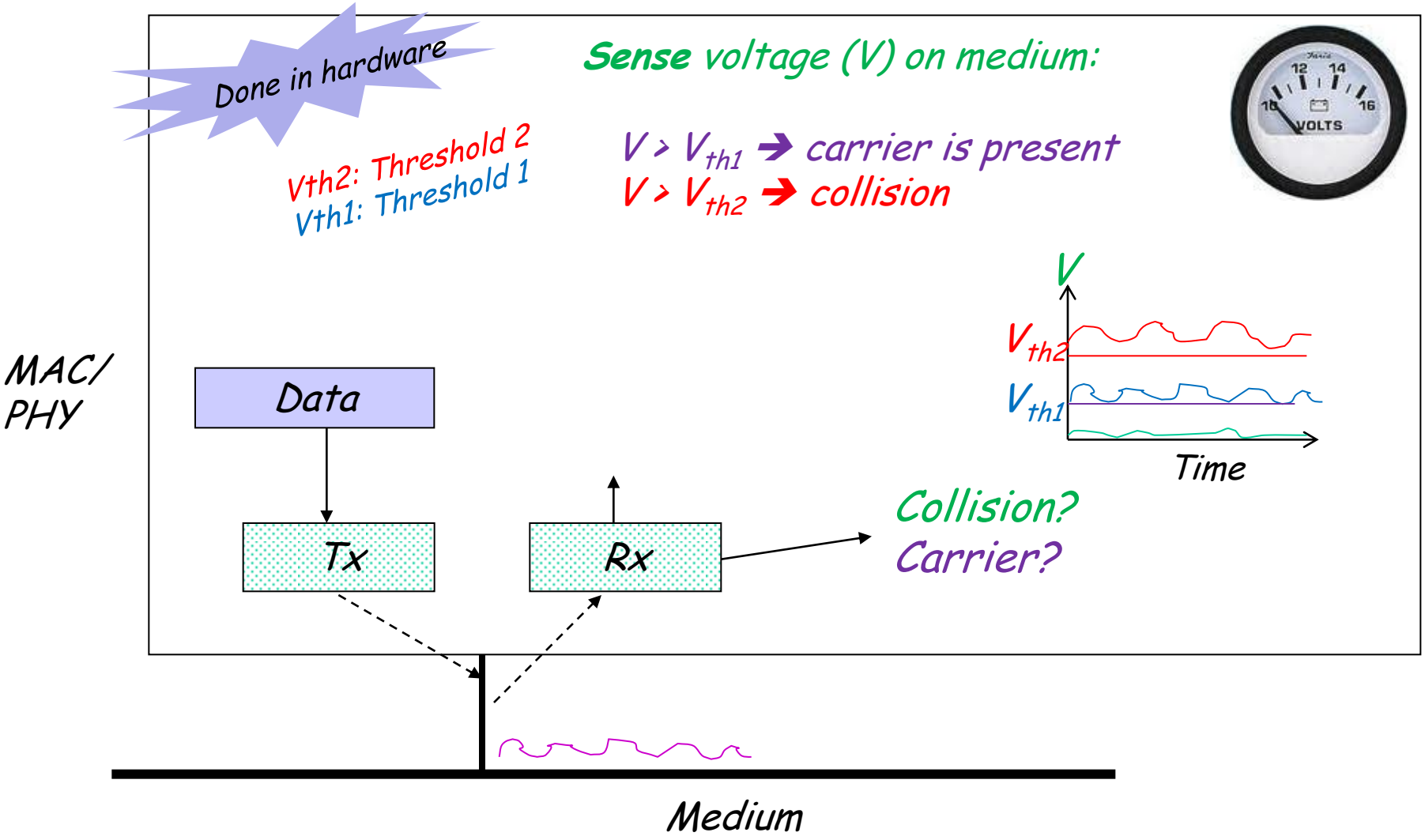
CSMA (Carrier Sense Multiple Access)

CSMA: listen before you transmit:

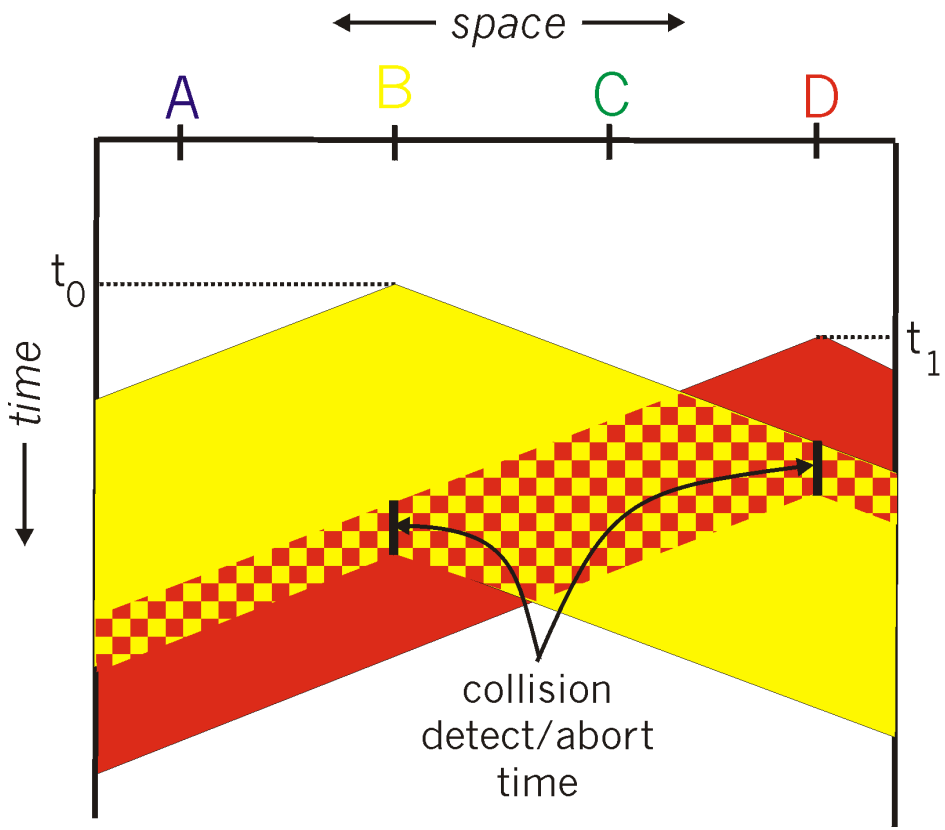
- If channel sensed idle: transmit entire frame
- If channel sensed busy, defer transmission



CSMA/CD Concepts of Carrier Sense and Collision Detection

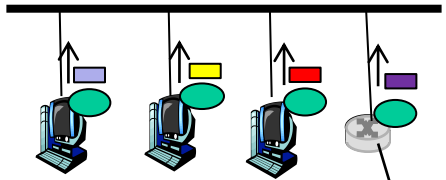


CSMA/CD collision detection



Remember this

Signal from all nodes must reach all others



*Therefore, **collision** occurring at a receiving node is detected by the frame's sender.... after some delay..*

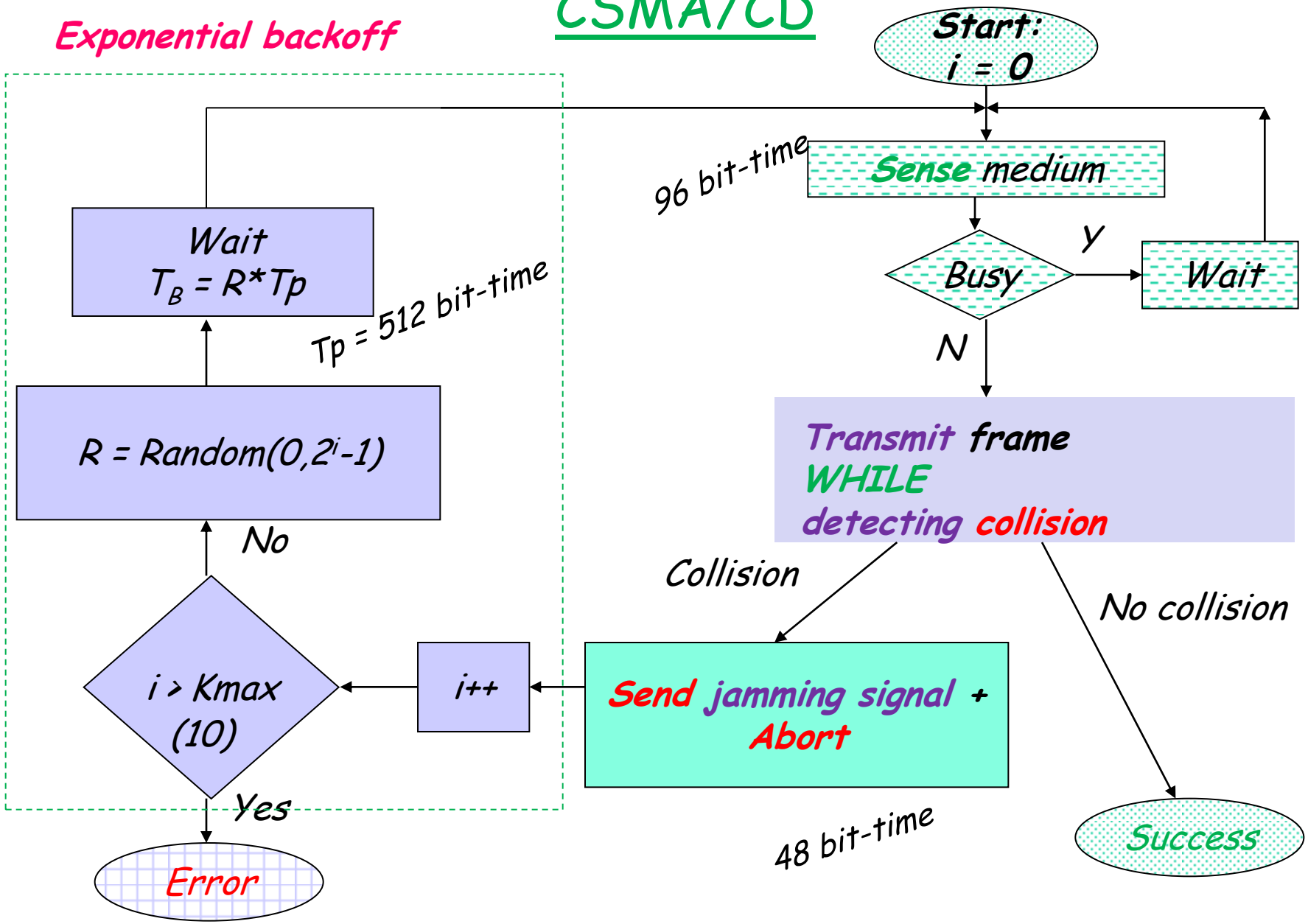
CSMA/CD (Collision Detection)

CSMA/CD: Carrier Sense Multiple Access with CD

- Tx attempts multiple times (multiple access)
- Collisions are *detected* within short time
- A colliding transmission is aborted ← don't keep wasting channel resource

- Collision detection:
  - Easy in wired LANs: explained before …
  - Not possible in wireless LANs: will be explained …

CSMA/CD

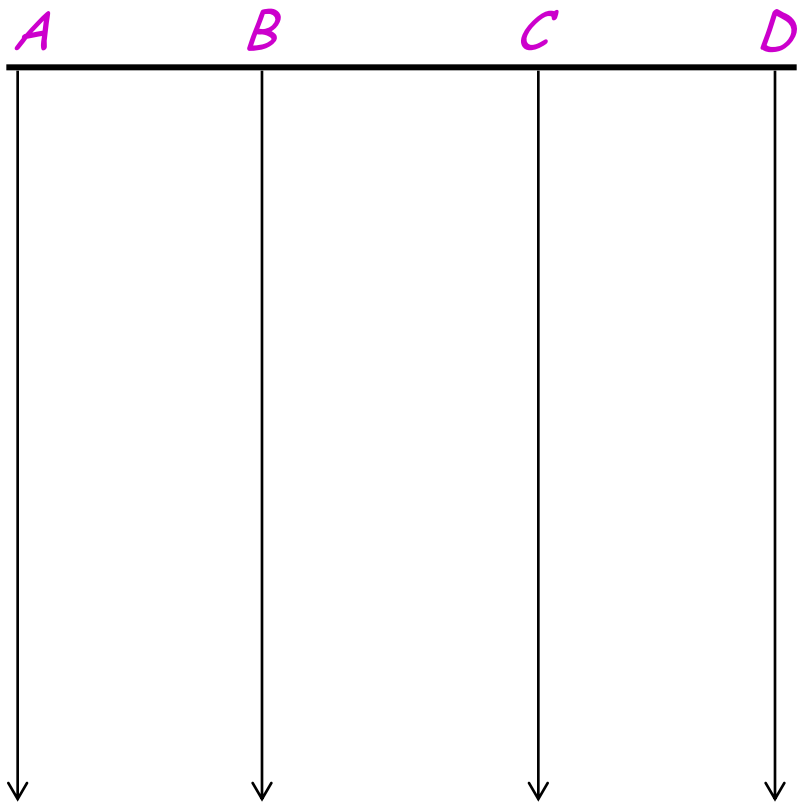

Exponential backoff
Start:
i = 0
96 bit-time
Sense medium
Busy
Y
Wait
N
Transmit frame
WHILE
detecting collision
Collision
No collision
Send jamming signal +
Abort
48 bit-time
Success
i++
i > Kmax
(10)
Yes
Error
No
R = Random(0,2^i-1)
Tp = 512 bit-time
Wait
T_B = R*Tp

CSMA/CD

- Medium sensing is done for 96 bit-times.
- Jamming signal length is 48 bits. Jamming signal creates enough energy on the medium for collision detection.
- Tp is equated with 512 bit-times.
- "i" saturates at 10.

I started transmitting
after sensing no-carrier....
Why did it collide with another frame.....?
Are they not listening...

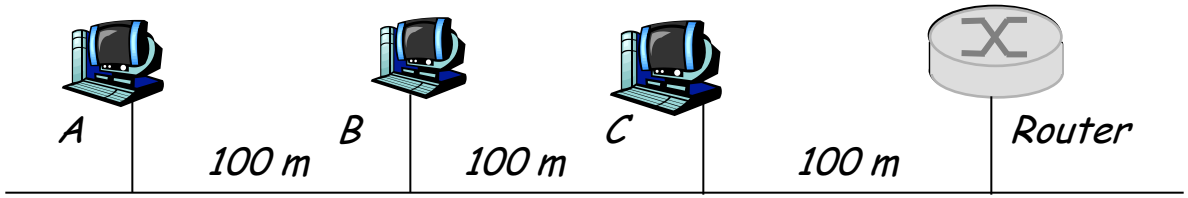
Past final exam



I have learned CSMA/CD… What kinds of questions should I expect on the exams

F'12 Mid-term

Computers *A, B, and C and one* ***router*** *are connected to an Ethernet bus* ***100 m****etres apart to make a LAN using the CSMA/CD protocol. Assume that signal propagates on the Ethernet bus at a speed of 2 x 10^8 m/sec, and all nodes can transmit data at the rate of 100 Mbps .*



- *After finding the bus to be idle for a little while, A and the router start transmitting their frames exactly at the same time.* ***How many bits of data*** *can node A transmit before detecting collision?*
- *What is the* ***maximum time gap*** *between the start of transmission and detection of collision by A?*
- *What is the* ***length of the smallest frame*** *that A can transmit while knowing whether or not it collided with a transmission from the router?*

Another question ....

Justify the use of binary exponential backoff in MAC protocols.

Mid-term F'12

Compare the **slotted Aloha** protocol with the **CSMA/CD** protocol by identifying *five* **comparison criteria.**

Criteria	Slotted Aloha	CSMA/CD

"Taking Turns" MAC protocols

channel partitioning MAC protocols:

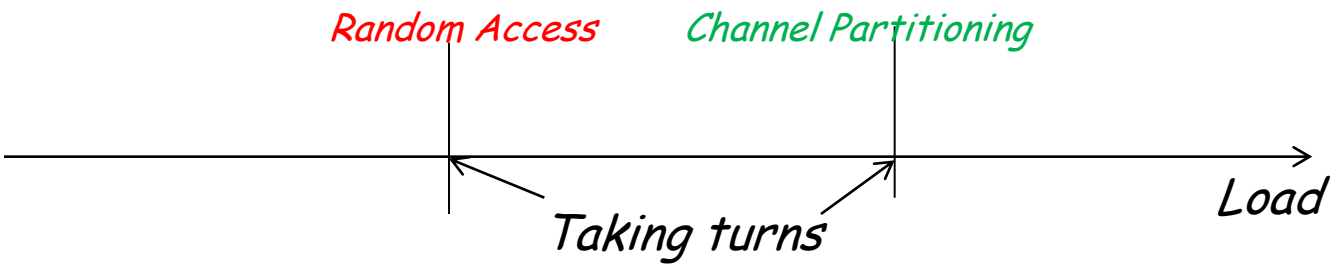
- share channel *efficiently* and *fairly* at high load
- inefficient at low load: 1/N bandwidth allocated even if only 1 active node!

random access MAC protocols:

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

"taking turns" protocols

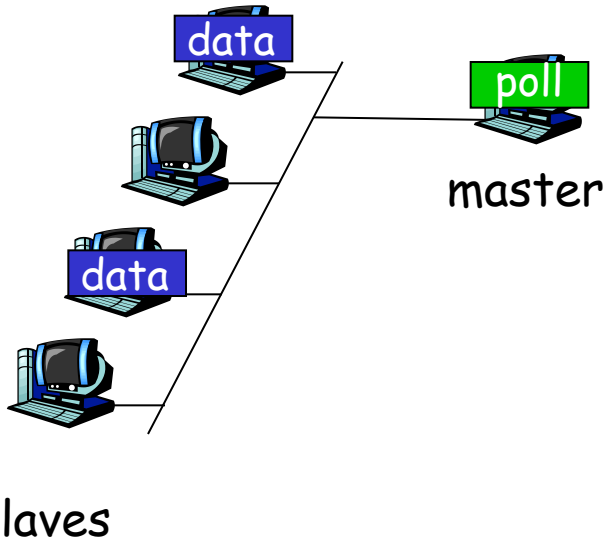
look for best of both worlds!



"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)

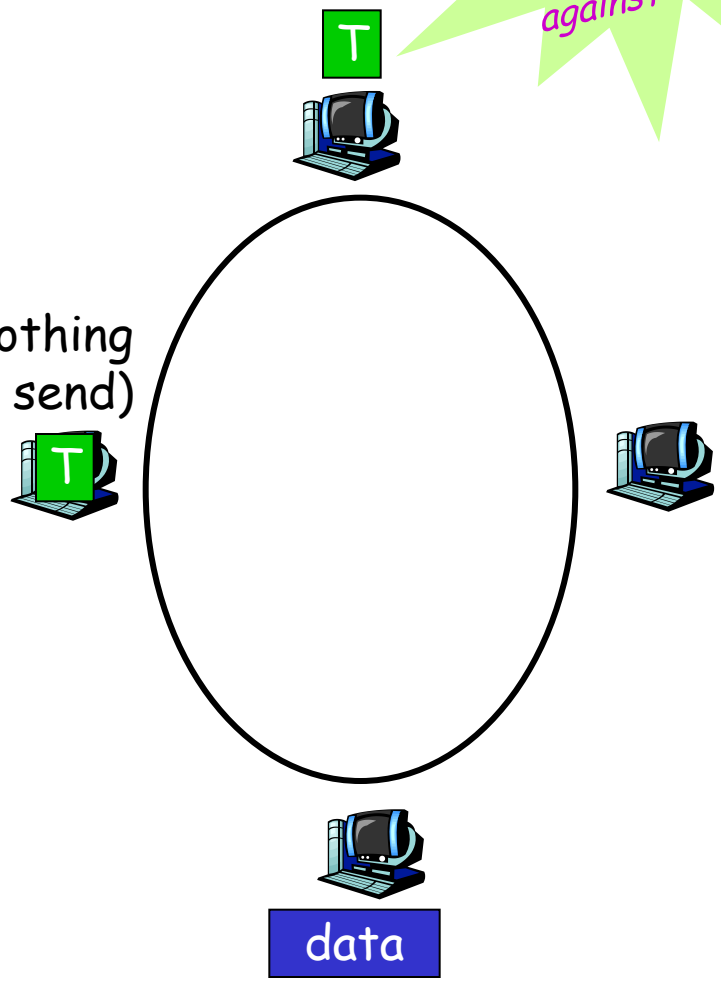


"Taking Turns" MAC protocols

Token passing:

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

IBM Token Ring was developed at IBM Zurich to compete against CSMA/CD.



Where are all those different kinds of MAC protocols used

Ch. Partitioning MAC

- Cellphone networks
- Bluetooth

Random access MAC

- LAN (Local Area Network)
- WLAN (Wireless LAN)

Taking turns

- Bluetooth

Link Layer

5.1 Introduction and services

5.2 Error detection and correction

5.3Multiple access protocols

5.4 Link-Layer Addressing

5.5 Ethernet

5.6 Link-layer switches

5.7 PPP

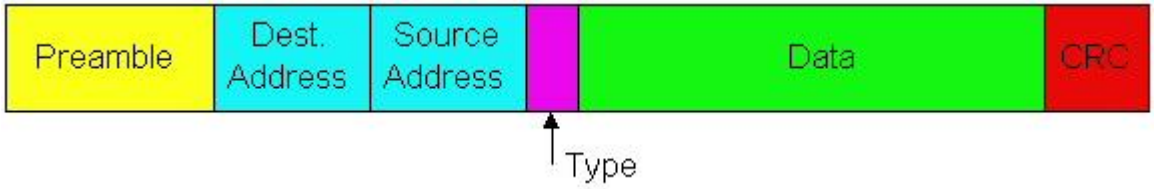
5.8 Link virtualization: MPLS

5.9 A day in the life of a web request

3.4 Reliable data transfer

Ethernet Frame Structure

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

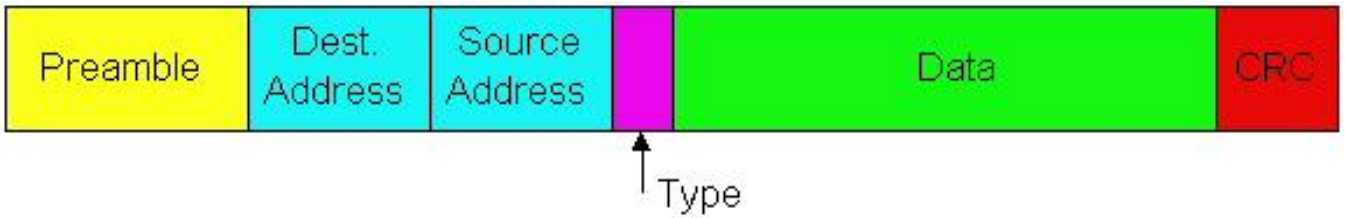


Preamble:

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

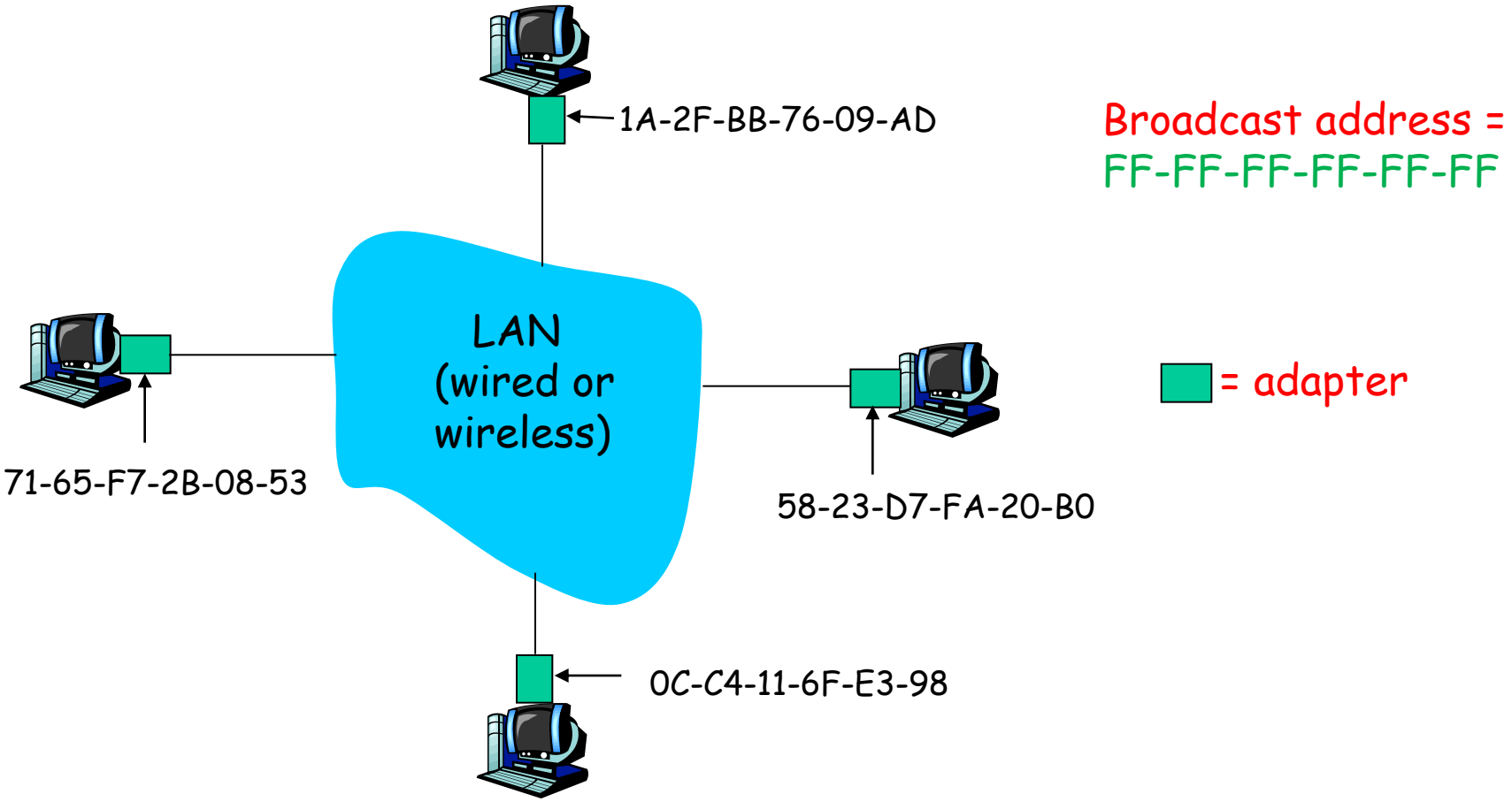
Ethernet Frame Structure (more)

- Addresses: 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to upper 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



LAN Addresses and ARP

Each adapter on LAN has unique LAN address



ARP: Address Resolution Protocol

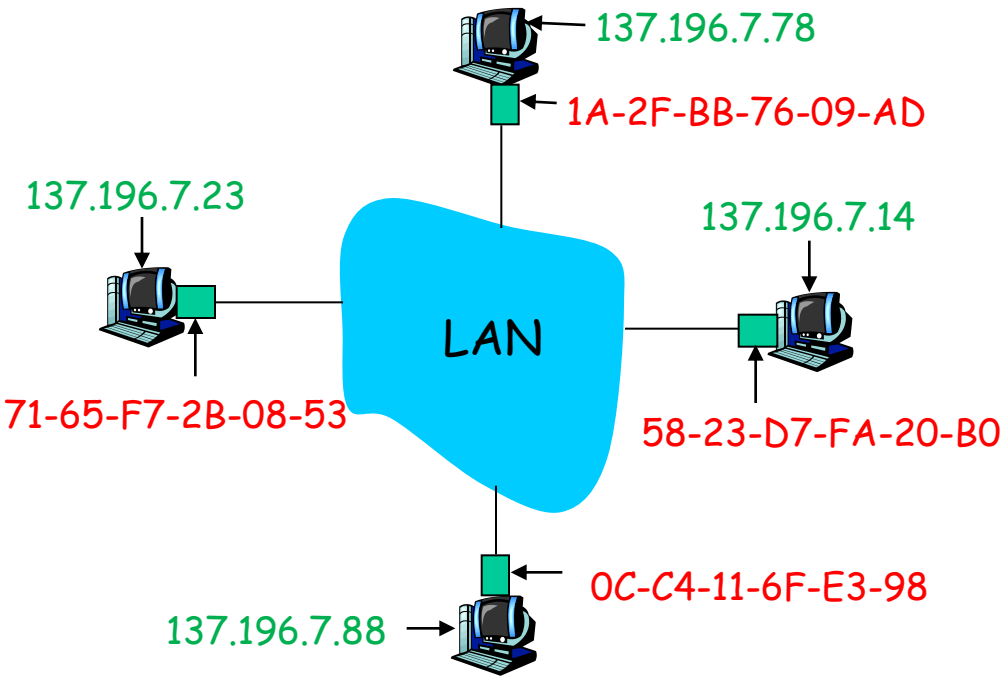
Question:
how to determine B's MAC address knowing B's IP address?

- Each IP node (host, router) on LAN has ARP table

- *ARP table: IP/MAC addr. mappings for LAN nodes*

< IP address; MAC address; TTL>

  - *TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)*



ARP protocol: Same LAN (network)

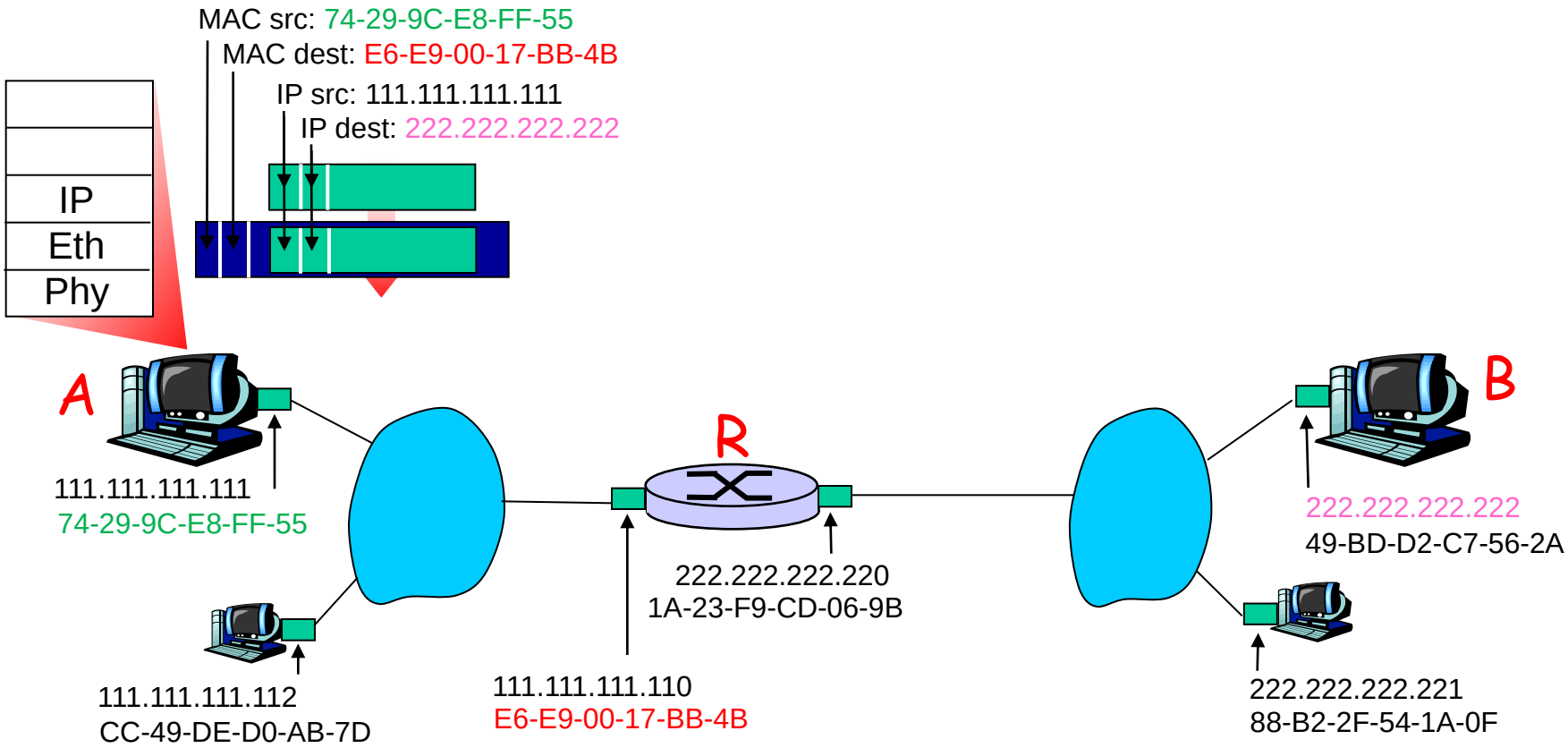
- A wants to send datagram to B, and B's MAC address not in A's ARP table.
  - A broadcasts ARP query pkt containing B's IP addr
    - dest MAC address = FF-FF-FF-FF-FF-FF
    - all machines on LAN receive ARP query
  - B receives ARP query, replies to A with its (B's) MAC addr
    - frame sent to A's MAC address (unicast)

- A caches (saves) IP-to-MAC addr pair in its ARP table until this becomes old

- ARP is "plug-and-play":
  - nodes create their ARP tables without intervention from net administrator

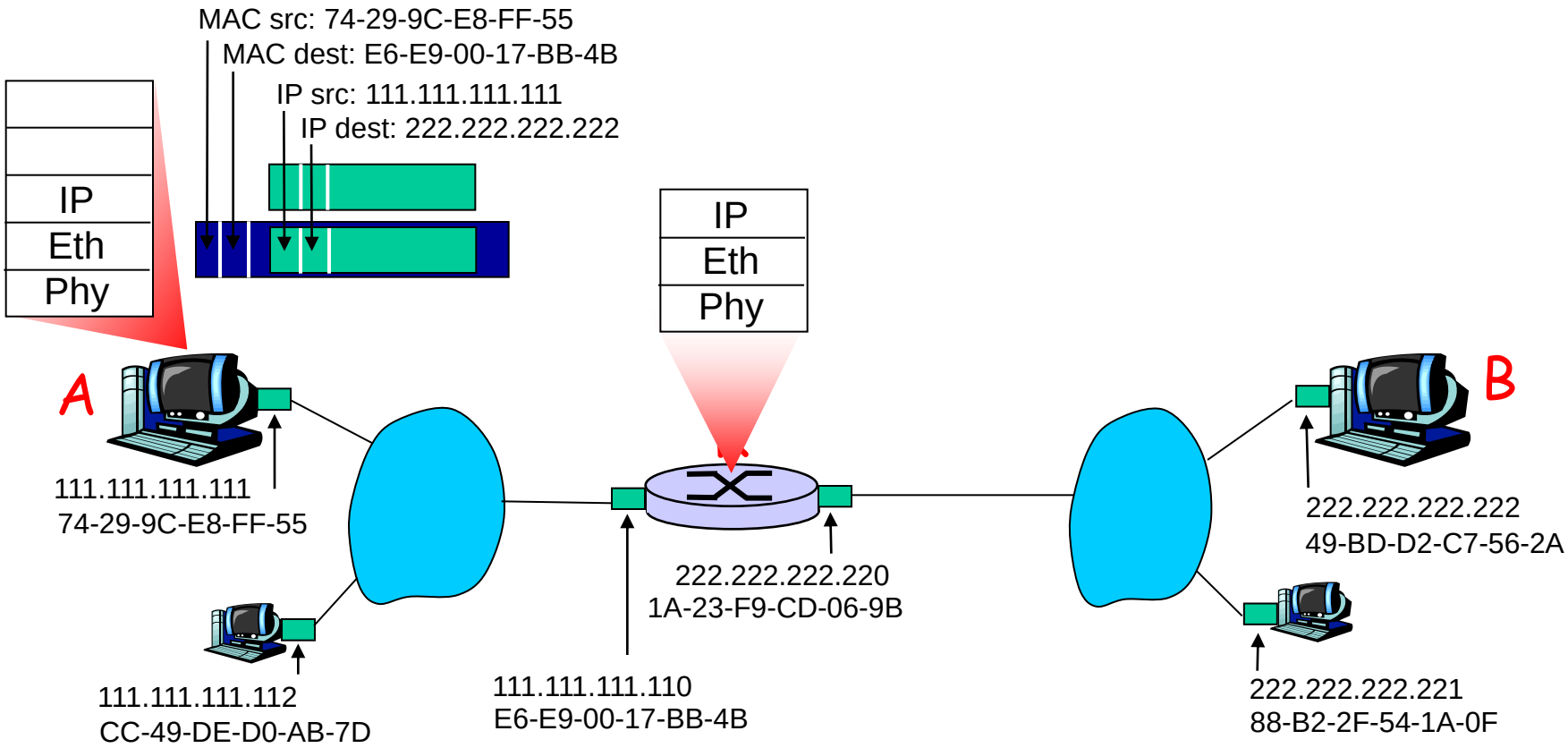
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



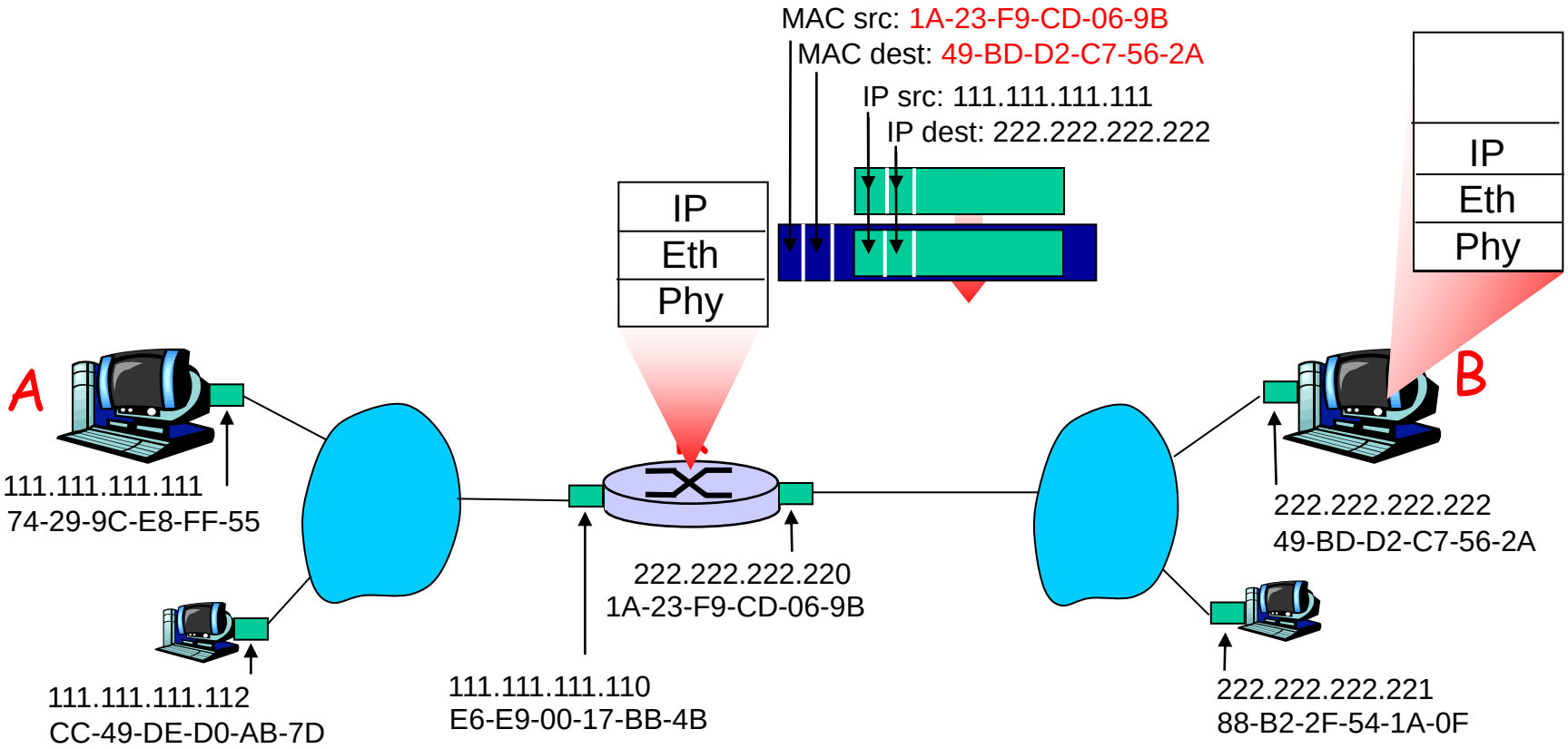
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



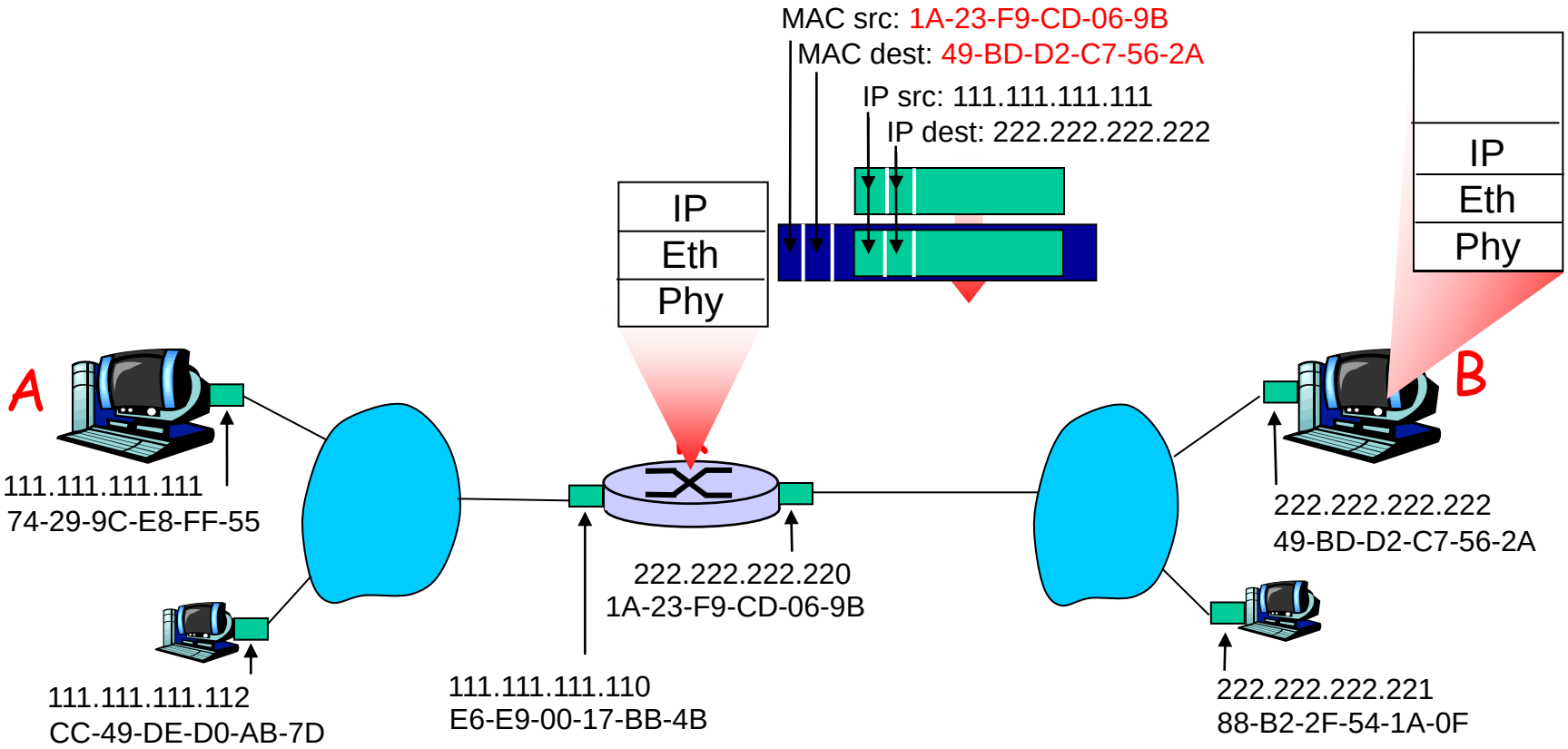
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



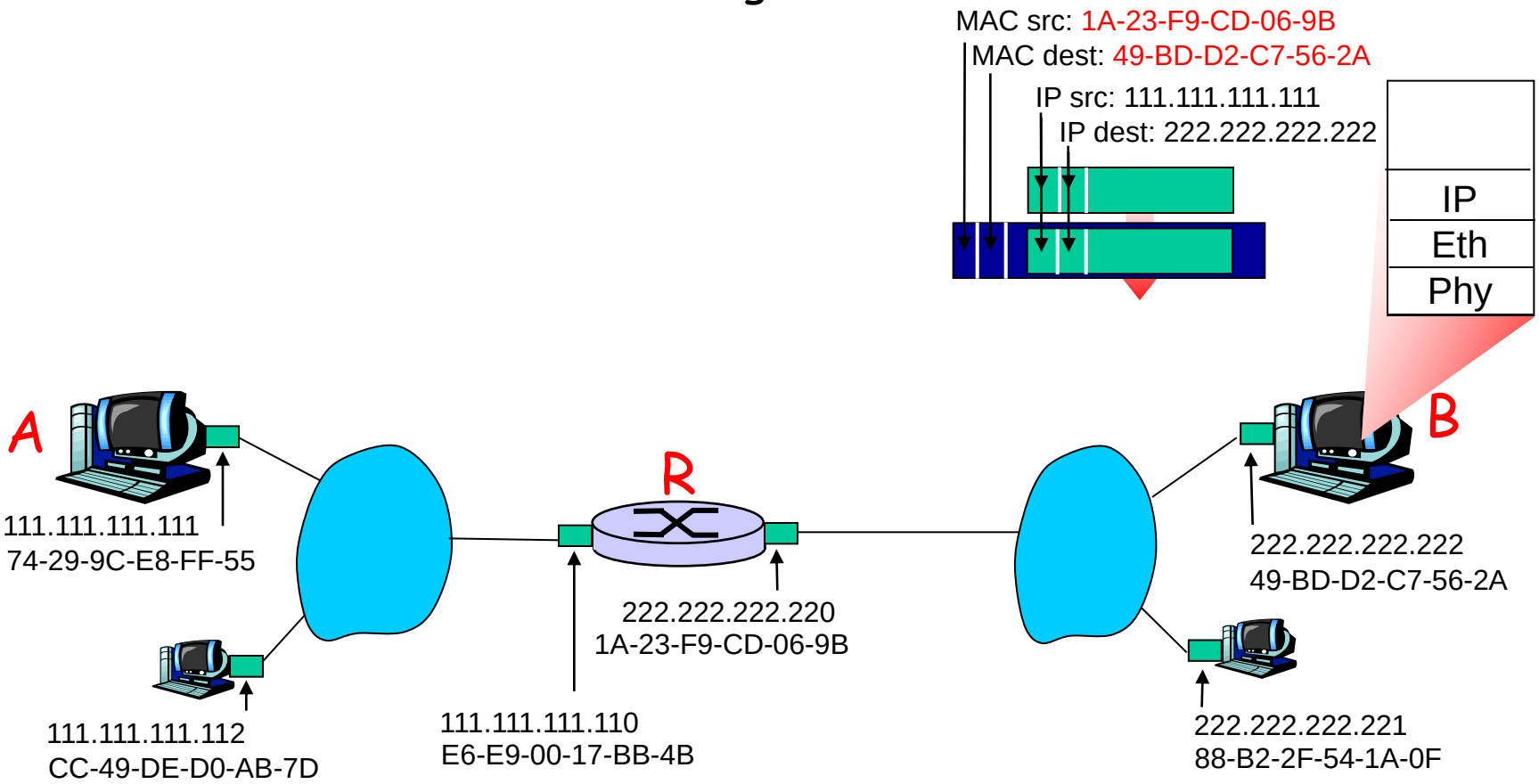
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Link Layer

5.1 Introduction and services

5.2 Error detection and correction

5.3 Multiple access protocols

5.4 Link-layer Addressing

5.5 Ethernet

5.6 Link-layer switches, LANs, VLANs

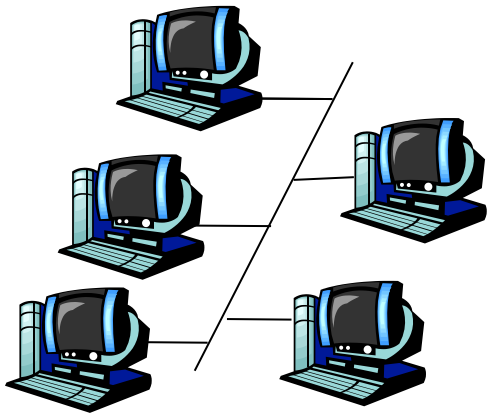
5.7 PPP

5.8 Link virtualization: MPLS

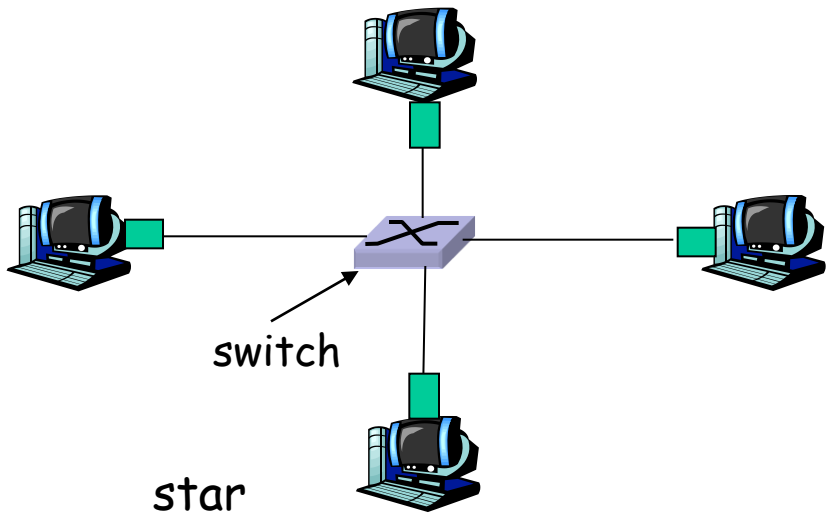
5.9 A day in the life of a web request

LAN with 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

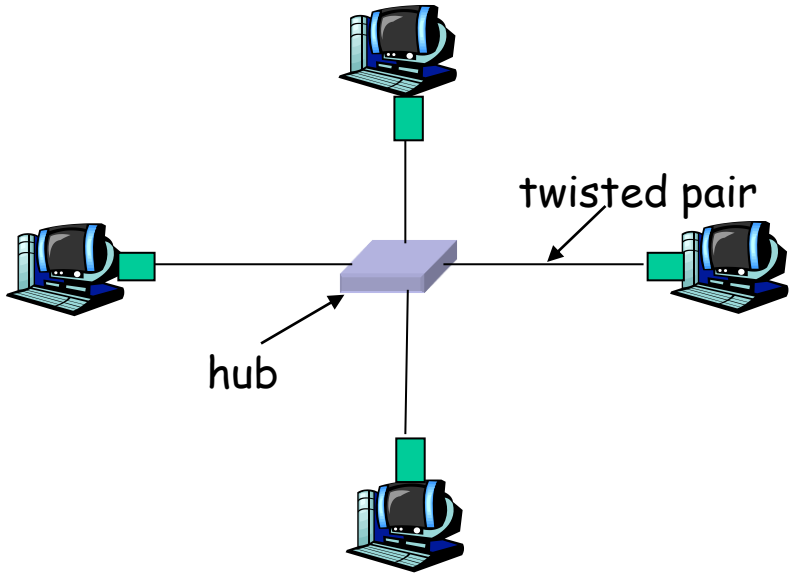


star

Hubs

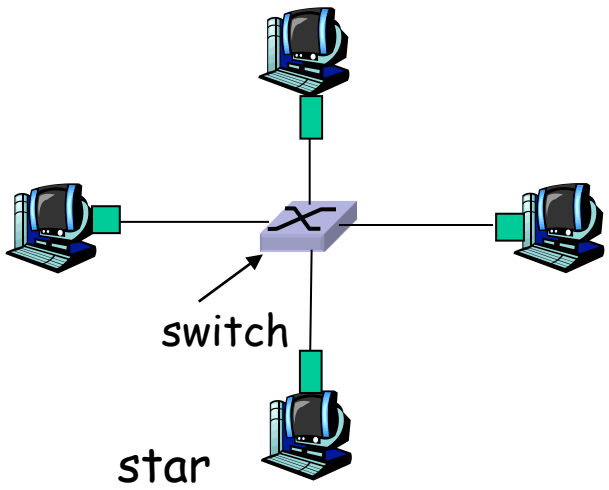
… physical-layer ("dumb") repeaters:

- bits coming in on 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



Switch

- smarter than hubs, take *active* role
  - Store and forward frames
  - Examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, using CSMA/CD to access segment

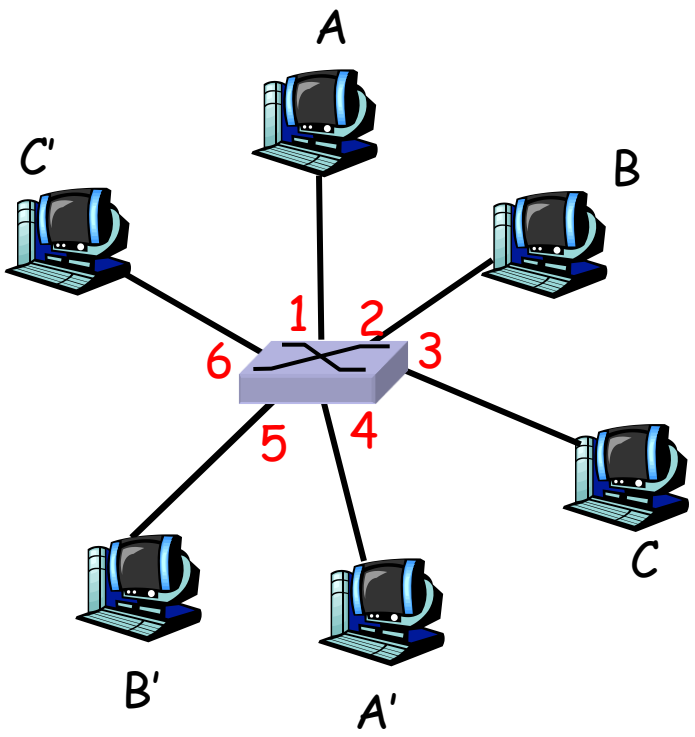


- transparent
  hosts are unaware of presence of switches

- plug-and-play, self-learning
  switches do not need to be configured

Switch: allows *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on each link
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions
  - not possible with dumb hub



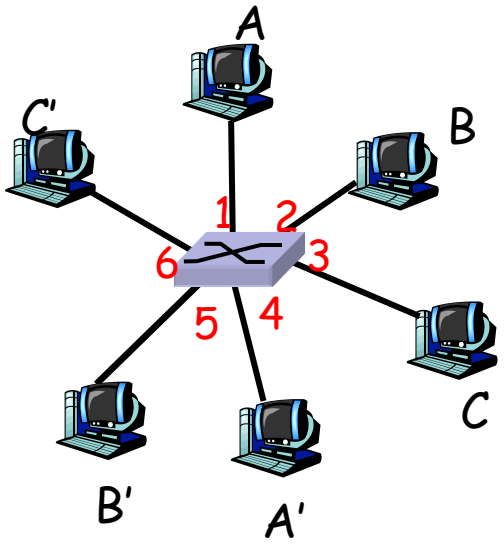
switch with six interfaces (1,2,3,4,5,6)

Switch Table

- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?

- *A:* each switch has a *switch table*

MAC addr of host	Interface #	Time To Live (TTL)

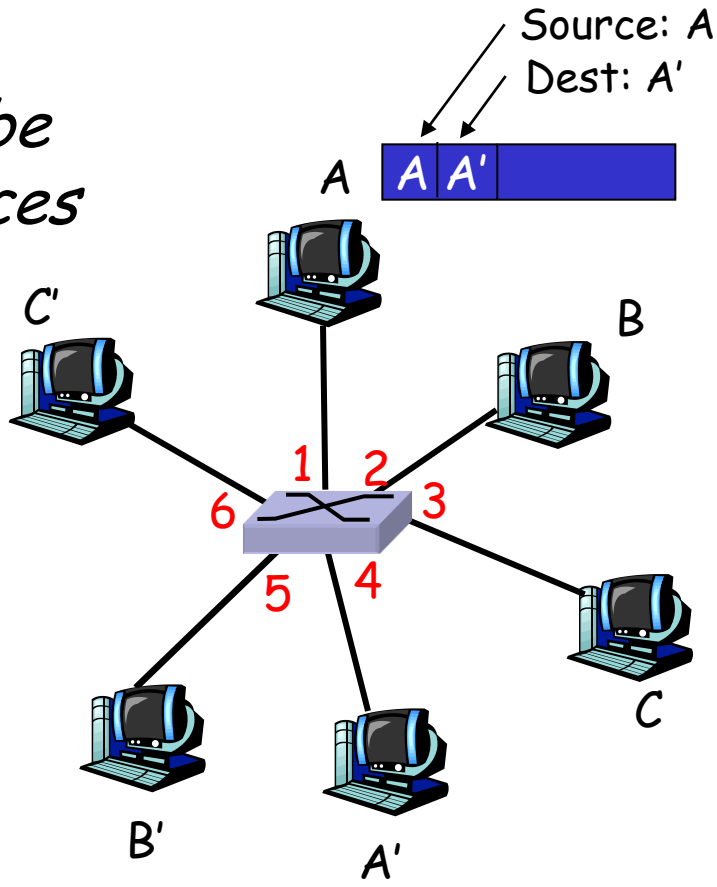


switch with six interfaces (1,2,3,4,5,6)

- *Q:* How are entries created and maintained in switch table?

Switch: self-learning

- *Switch learns which hosts can be reached through which interfaces*
- When frame received
  - Switch "learns" location of sender: incoming LAN segment
  - Records sender/location pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Switch: frame filtering/forwarding

When frame received:

1. record link associated with sending host

2. index switch table using MAC dest address

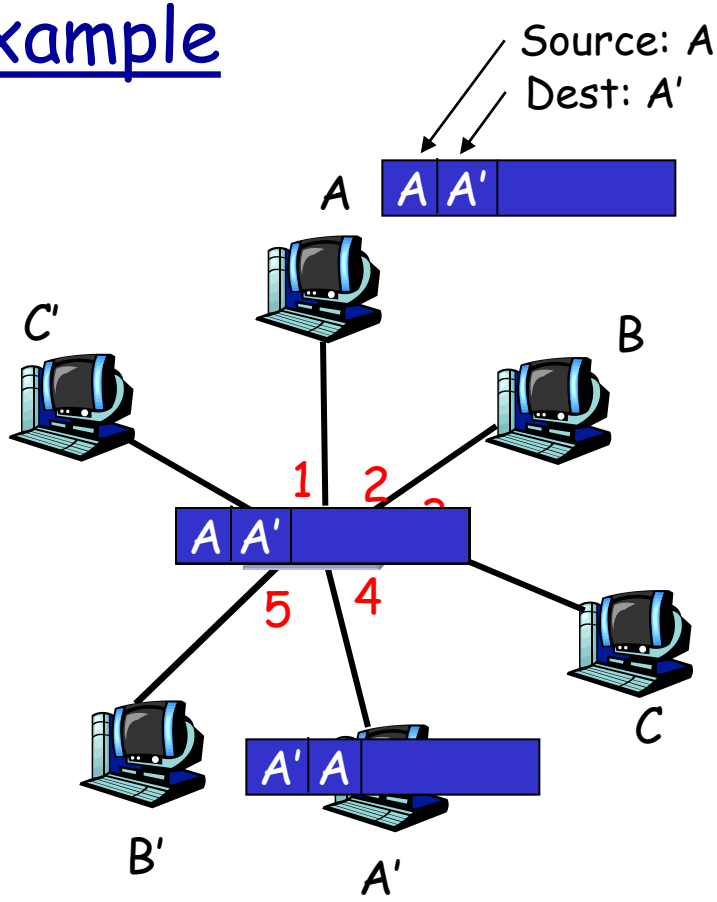
3. if entry found for destination then {
   if dest on segment from which frame arrived
   then drop the frame
   else forward the frame on interface indicated
   }

else flood

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

- frame destination unknown: *flood*
- destination A location known: *selective send*

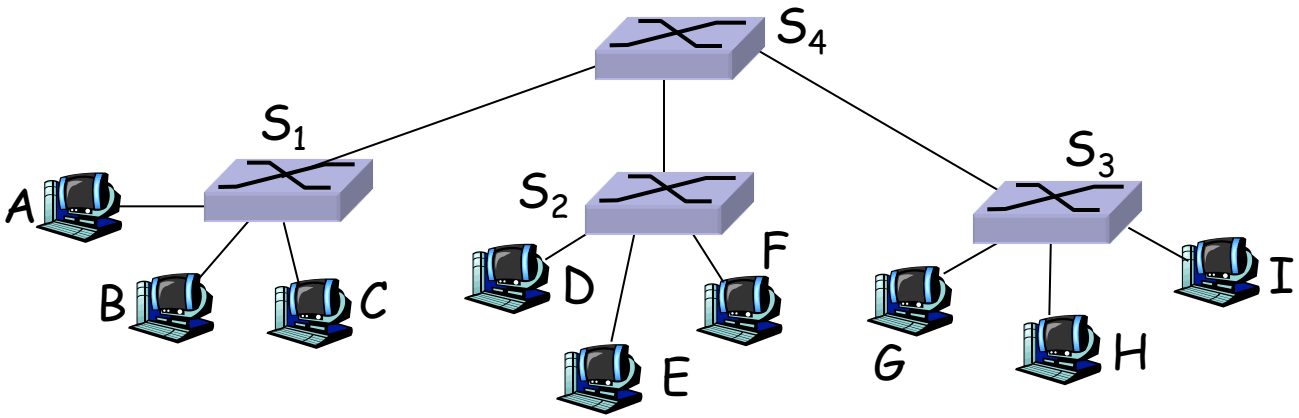


MAC addr	interface	TTL
A	1	60
A'	4	60

Switch table (initially empty)

Interconnecting switches

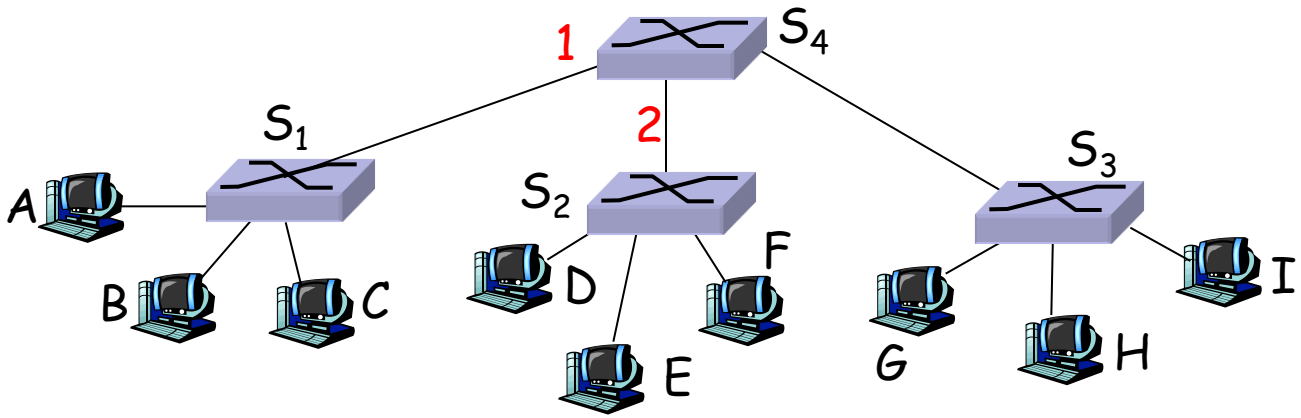
- switches can be connected together



- *Q:* sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?
- *A:* self learning! (works exactly the same as in single-switch case!)

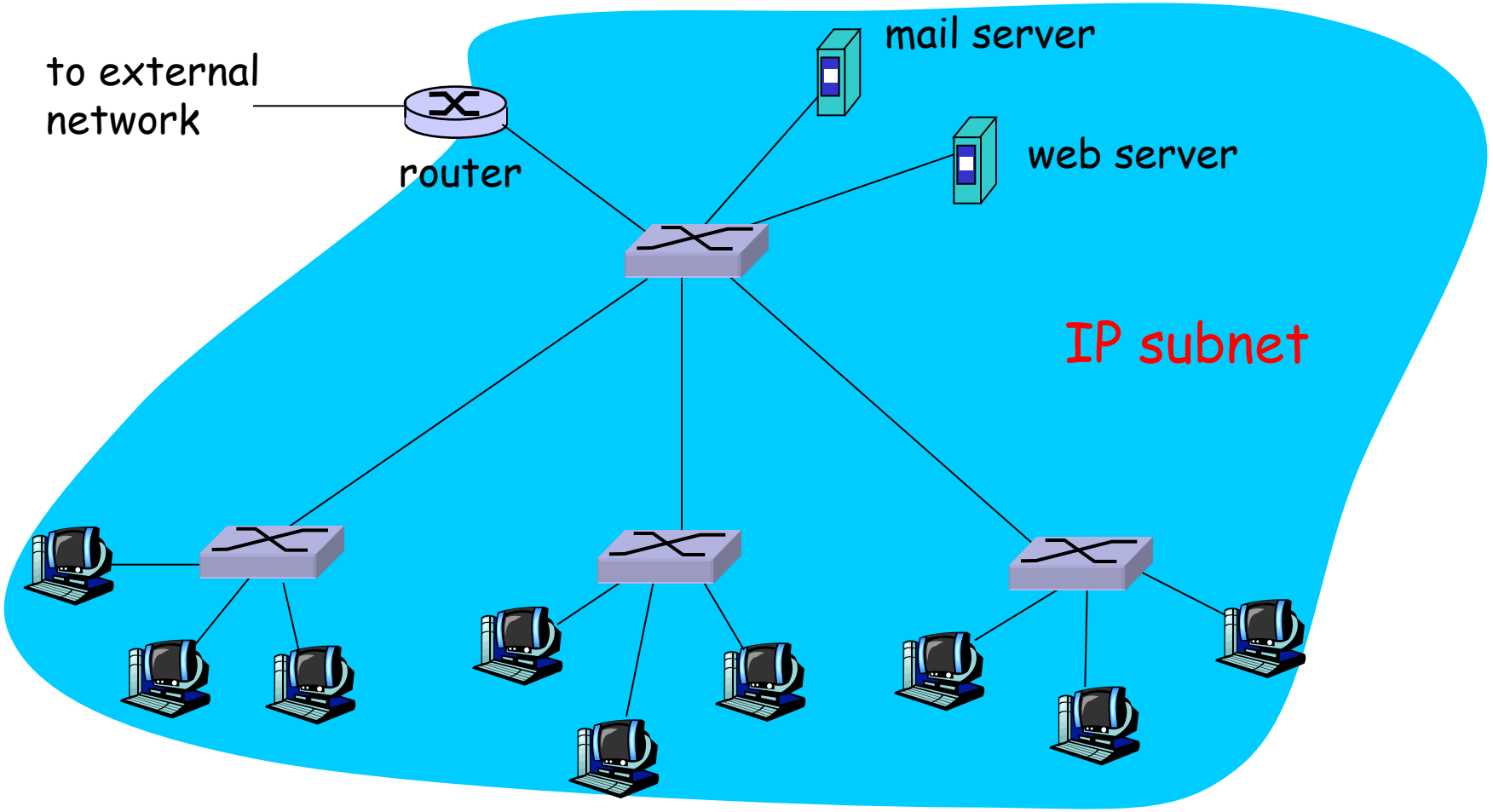
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



- *Q:* show switch tables and packet forwarding in S_1, S_2, S_3,S_4

Institutional network



Fall 2012 Final Exam Question

[3] Consider the institutional local area network shown in Figure 1.

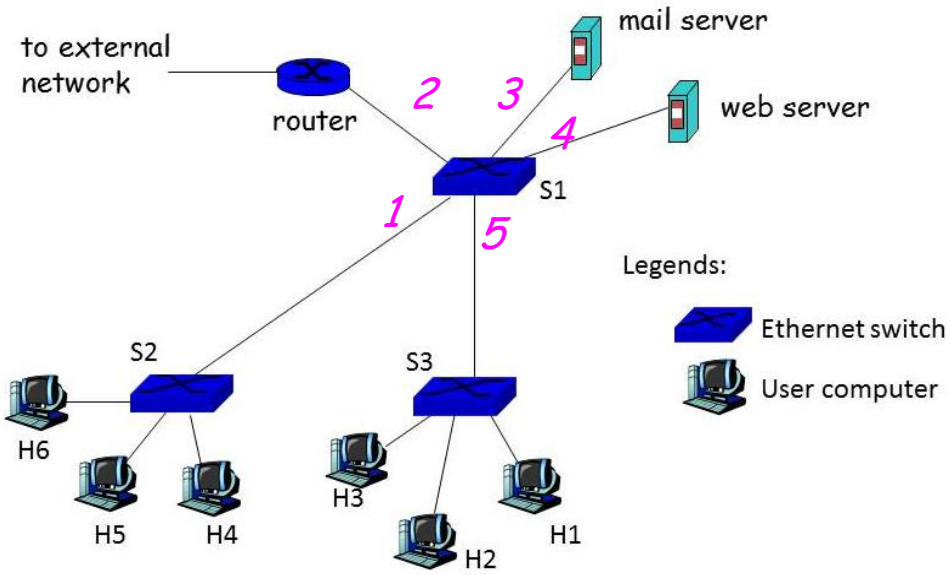


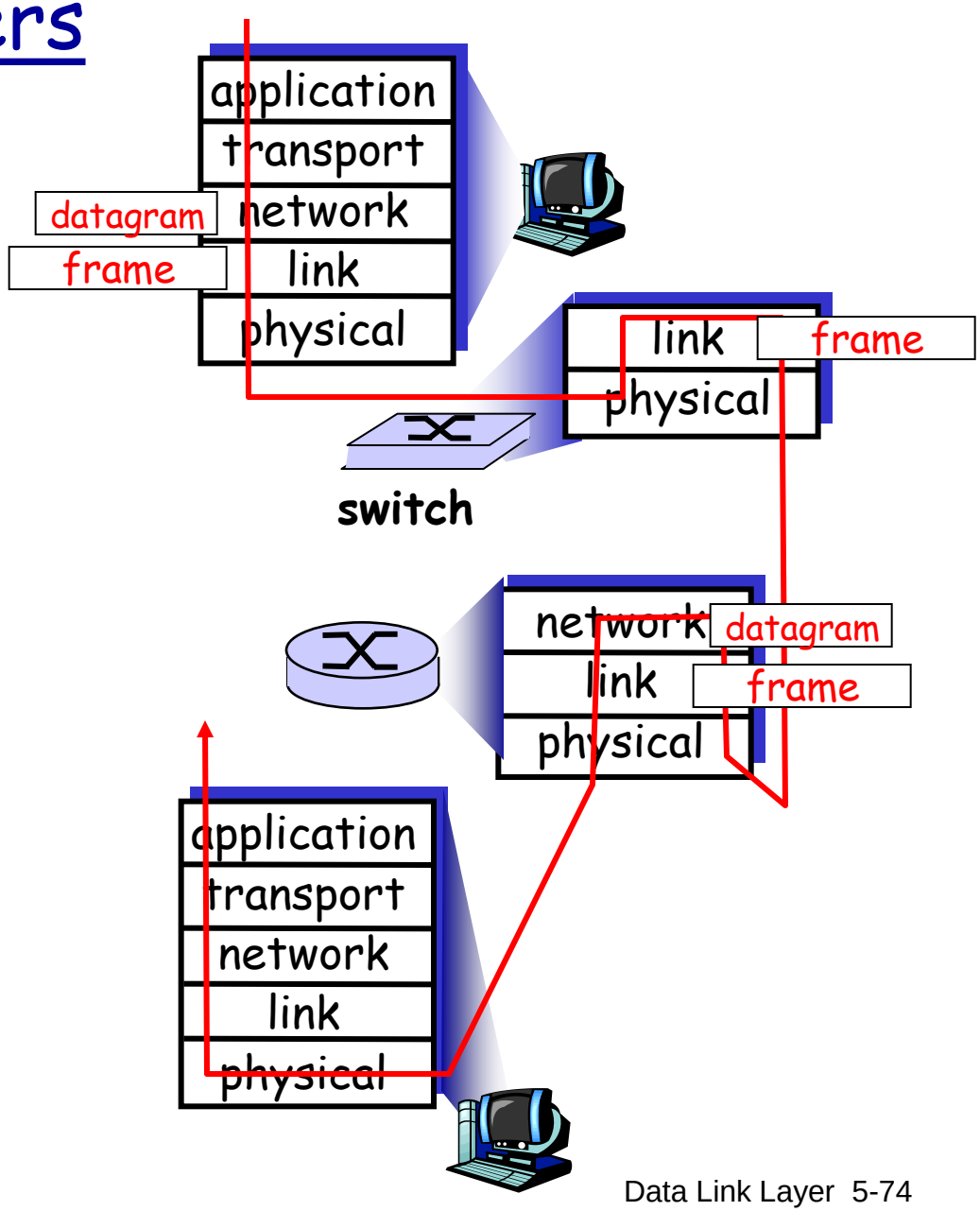
Figure 1: An institutional network

All the computing entities, except H1 and H4, are fully functional and sending and receiving data packets. H1 is switched off, whereas H4's Ethernet interface is broken.

Complete the following switching table at Ethernet switch S1.
An arbitrary number of rows have been given. Add more rows, if you need them.
Make assumptions if there is a need.

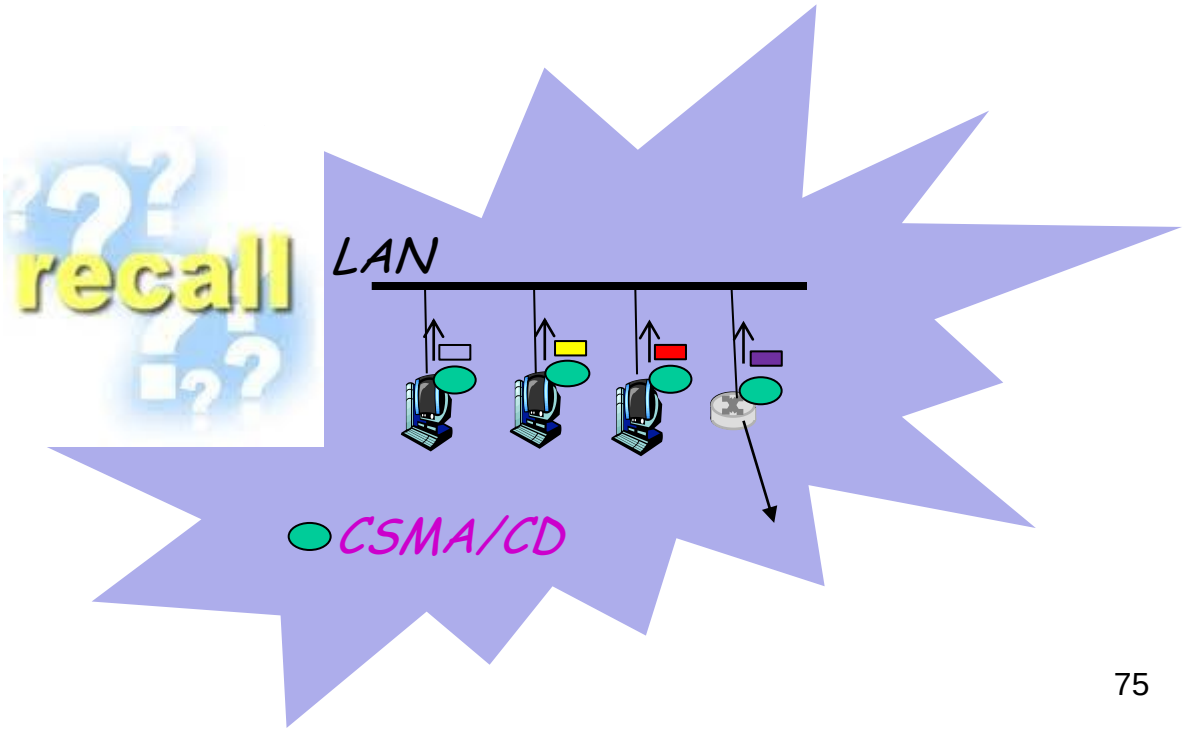
Switches vs. Routers

- both store-and-forward devices
  - routers: network-layer devices (examine network-layer headers)
  - switches are link-layer devices (examine link-layer headers)
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms



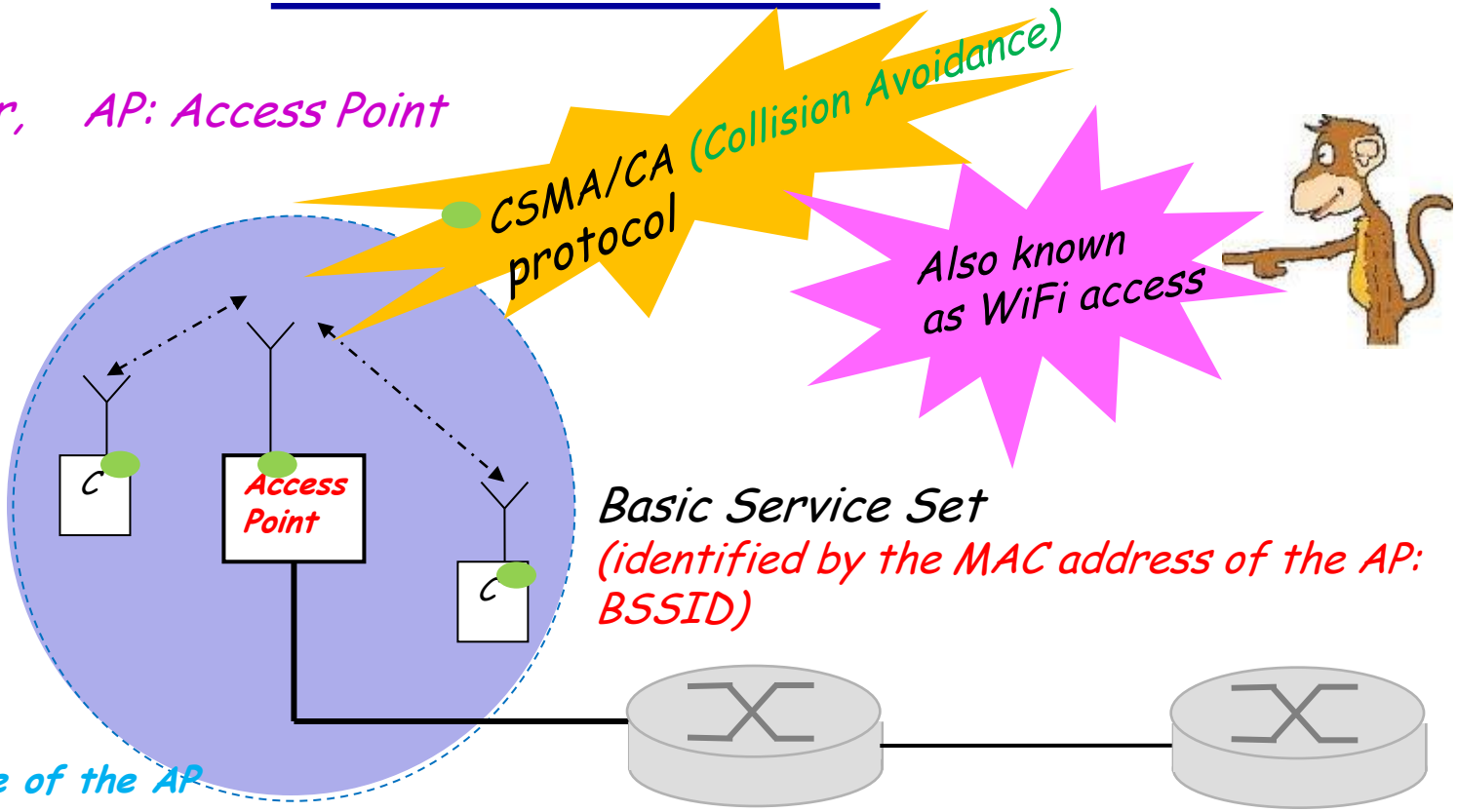
Wireless LAN

Standard: IEEE 802.11/a/b/g



WLAN: Basic idea

C: Computer, AP: Access Point



Independent BSS (IBSS)= BSS – AP

Extended Service Set (ESS): A collection of BSS connected by a Distribution System

Example: The UW WiFi network is one ESS.

IEEE 802.11/a/b/g/n Family

IEEE	Signal Transmission	Frequency Band	Rate **(Mbps)**
802.11	DSSS FHSS	2.4 GHz 2.4 GHz	1 and 2 1 and 2
802.11a	OFDM	5 GHz	6--54
802.11b	DSSS	2.4 GHz	5.5 and 11
802.11g **802.11n** **802.11ac** **(Draft/Nov. 2011)**	OFDM OFDM OFDM	2.4 GHz 2.4/5 GHz 5 GHz	22 and 54 72 and **150** **6.9 Gbps**

DSSS: Direct Sequence Spread Spectrum
FHSS: Frequency Hopping Spread Spectrum
OFDM: Orthogonal Frequency Division Multiplexing

Are there different MAC protocols in IEEE 802.11/a/b/g/n standards

They have all different PHY layers ……

But, all of them have the same MAC protocol …..

First ....

Different **Modes** of Operation of MAC in IEEE 802.11

Modes of IEEE 802.11 MAC

- There is no need for an AP.
- If AP is there, it behaves like an **ordinary** node.
- Nodes make their own decisions to Tx.

Distributed Coordination Function (DCF) mode

Point Coordination Function (PCF) mode

The AP is the controller AP **decides** who transmit and when.

Handshake mode

Without Handshake mode

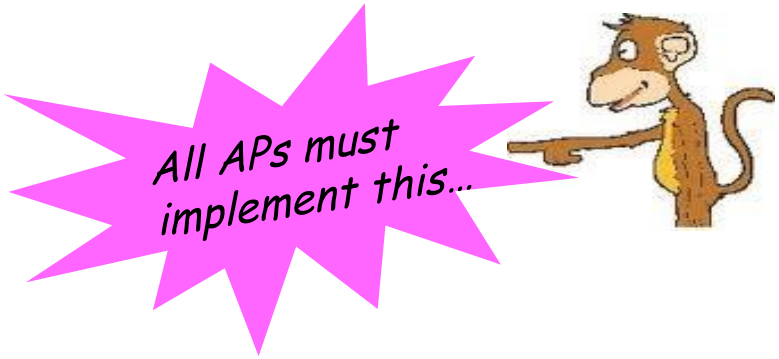
Seek permission from receiver before sending **data**

No prior permission is needed.

PCF Mode

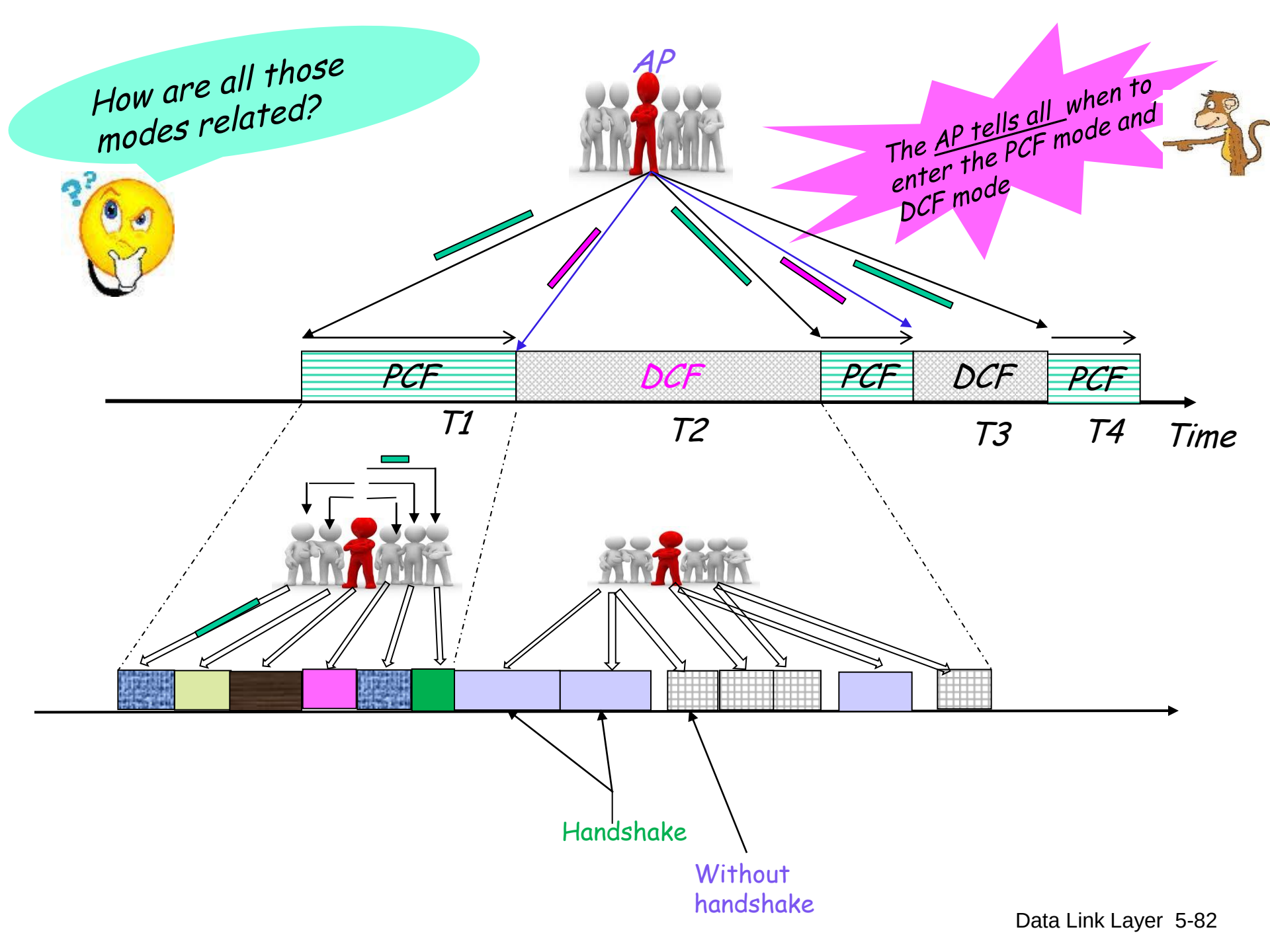
- The AP
  - Operates as the **central controller** in the BSS.
  - Decides who transmits and when.
  - There is no contention for medium access
  - Can follow a round-robin policy to allocate slots.
- This mode
  - Leads to waste of bandwidth if a scheduled node has no traffic.

DCF Mode



- An AP
  - Need not be used.
    - Computers can directly communicate among themselves <= Ad hoc.
  - Is used to provide connectivity to the Internet.

- In DCF
  - All nodes, including the AP, compete for medium access.
  - The AP does not operate as a central controller.


How are all those modes related?
AP
The AP tells all when to enter the PCF mode and DCF mode
PCF
DCF
PCF
DCF
PCF
T1
T2
T3
T4
Time
Handshake
Without handshake

How do I know when to use handshake and when to use without-handshake

The MAC management database holds a variable: **dotRTSThreshold** (integer in bytes)

Handshake mode

Frame length >= dotRTSThreshold

Without-Handshake mode

Frame length < dotRTSThreshold

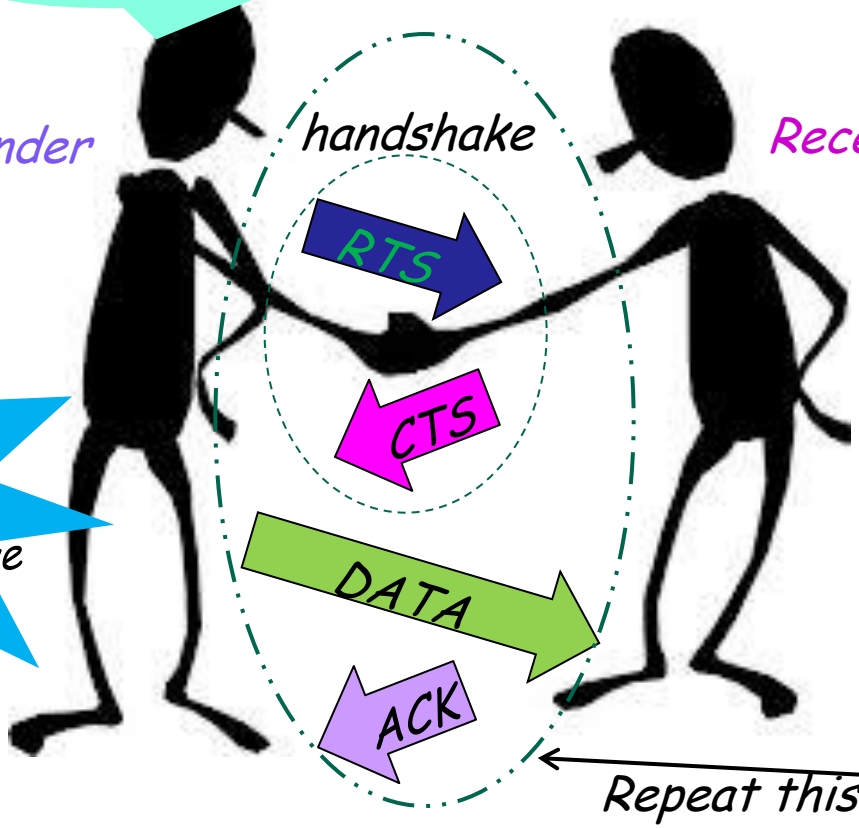
For "long" frames, you want to reduce the prob. of collision... with some overhead

DCF with *handshake*

How is handshake implemented

Under a certain condition ….
(to be explained soon ….)

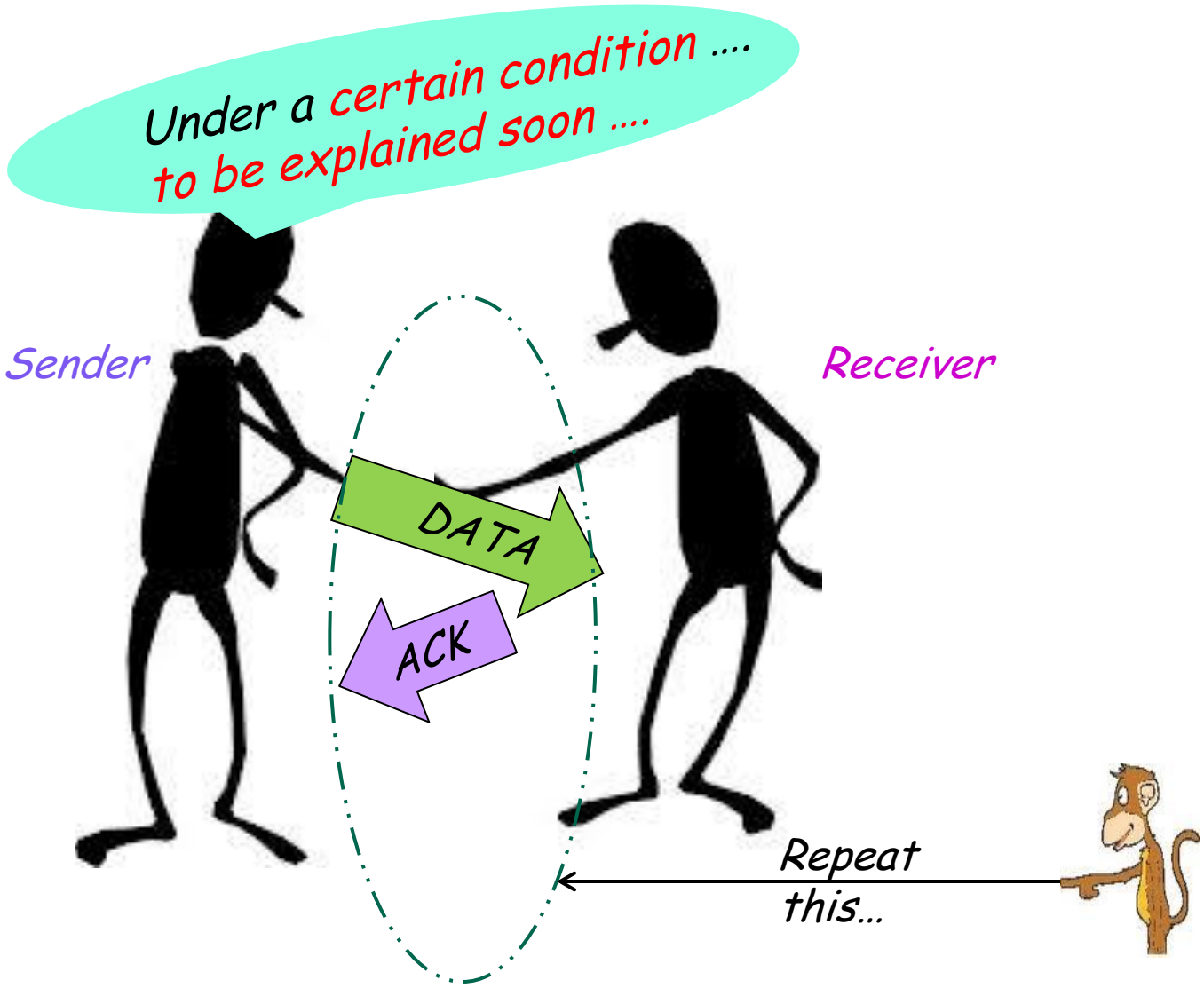
Control Frames:
RTS: Request To Send
CTS: Clear To Send



Note: There is strict timing relationship between successive frames

Receiver is under no obligation to send CTS.

DCF without handshake

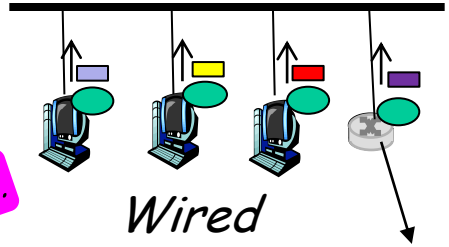


What is the big deal in a WLAN?

What happens to my RTS, CTS, DATA, ACK,.....
Do they collide with Tx from other nodes

Note the diff. between a wired medium & a wireless medium

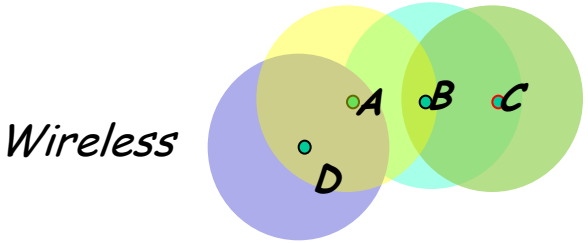
Signal from all nodes must reach all others



Remember this diff...

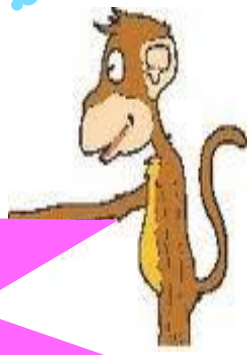
This diff. leads to some problems

Signal from some nodes may not reach all others



What problems can arise in a wireless net

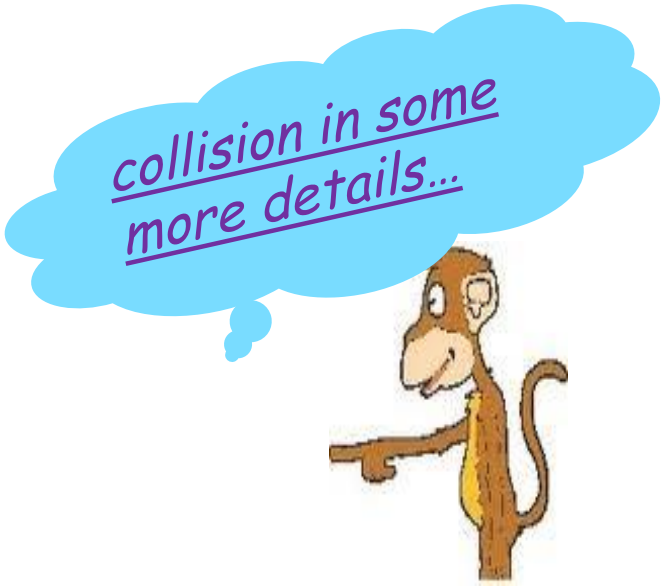
Collision detection is not always possible in a wireless LAN.



*You do not receive the **signal** received by the intended receiver of your frame.*

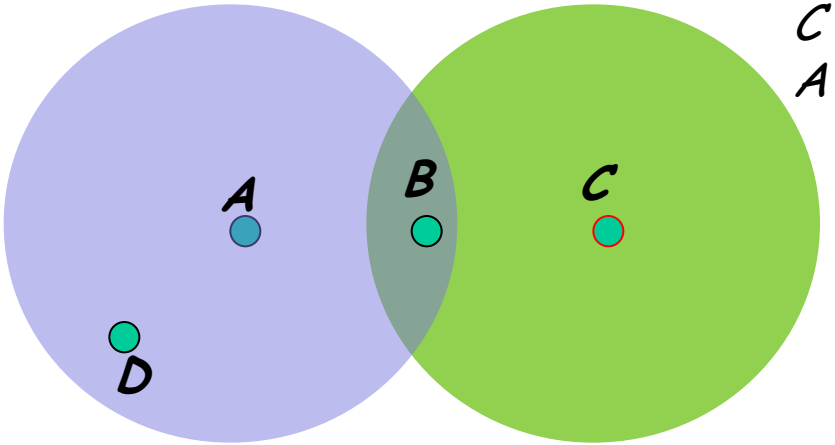
As a result, if your frame encountered collision at the receiver, you are NOT aware of that.

Two Problems in WLAN



- Hidden Terminal Problem
- *Exposed Terminal Problem*

Hidden Terminal Problem



C is transmitting a frame to B.
A is unaware of C's Tx.

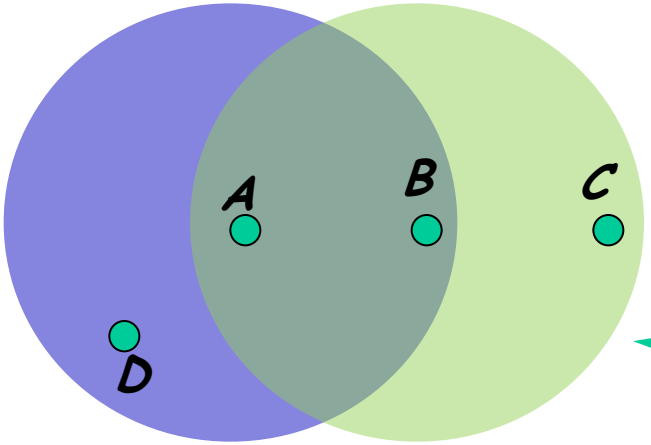
Now, if A transmits, A's frame will collide with C's at B

- *This problem is due to C being **hidden** from A.*
  - *Hidden means being "far away" …*

Collision occurs at B, but A cannot detect it…

Solution exists: CSMA/CA (Collision Avoidance)

Exposed Terminal Problem



No solution has been provided so far…..

A is transmitting a frame to D.

B knows that someone is transmitting.

If B transmits a frame to C, it does not collide with A's at D.

However, B does not transmit because it is unaware of D's location.

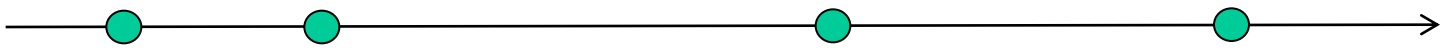
Problem: Loss of opportunity to transmit ➔ Loss of bandwidth ….

- The above problem is due to B being **exposed** to A's Tx.

WLAN MAC: CSMA/CA (basic idea)

- Collision is avoided by using two techniques:
  - **PHY-level carrier sensing**: *Done in receiver hardware*
  - **Virtual carrier sensing**:
    - An integer variable in each node - called NAV (Network Allocation Vector) -- indicates whether or not a nearby node is likely to be transmitting.
      - NAV > 0 ➔ Most likely another node is transmitting.
      - NAV = 0 ➔ No one nearby is transmitting.
      - ~~NAV < 0: This condition does not occur.~~
- *Transmit condition: Medium is idle at the receiver.*

(Carrier is absent) ***AND*** *(NAV = 0)*



CSMA/CA: NAV in detail

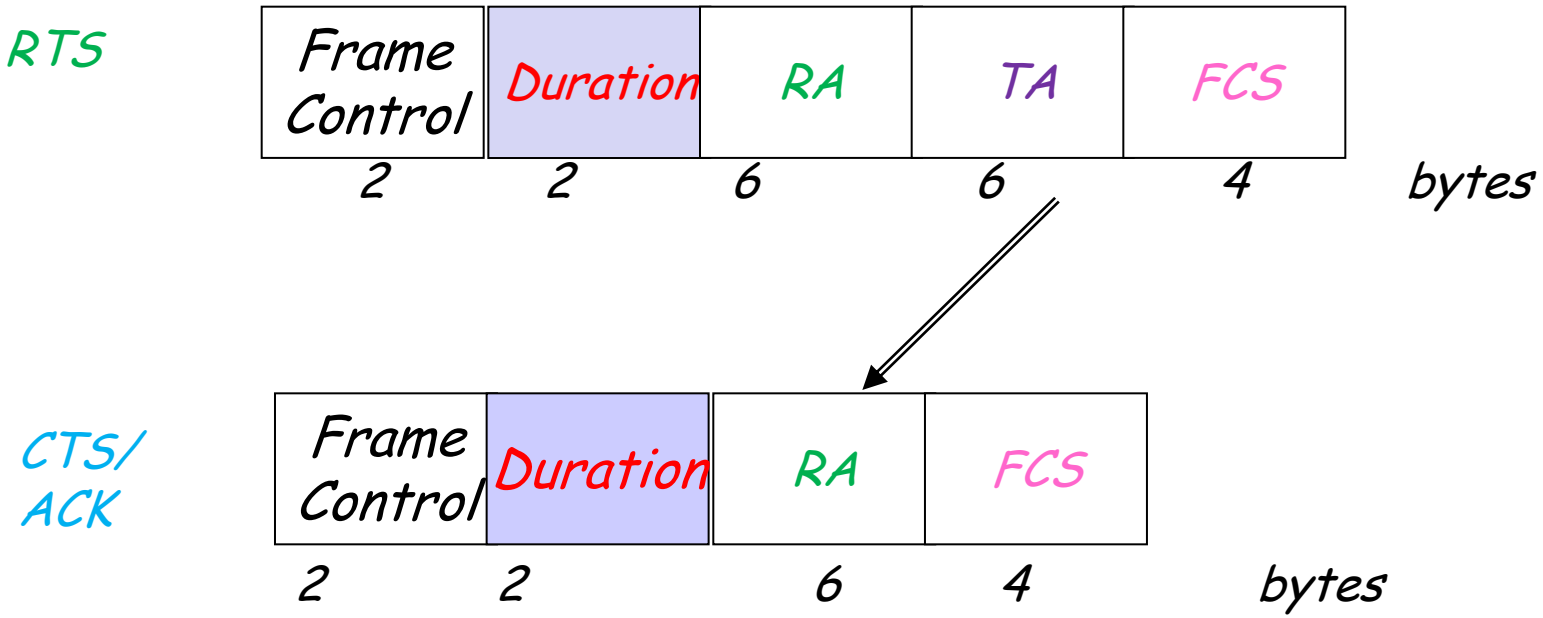
- *Recall the handshake mechanism with RTS and CTS*

- A ***duration*** field in frame header indicates the length of time the sender of the frame may use the medium.
- *All nodes which receive RTS and CTS (and DATA and ACK) update their NAV as follows*
  - *Events that update NAV: Initially NAV = 0*

With each passing μs (micro second)
*If (NAV > 0) **NAV = NAV – 1***
Else stop decrementing NAV

*If a frame (e.g. RTS, CTS) with **duration** field is received*
NAV = Max(NAV, duration)

RTS and CTS Frames

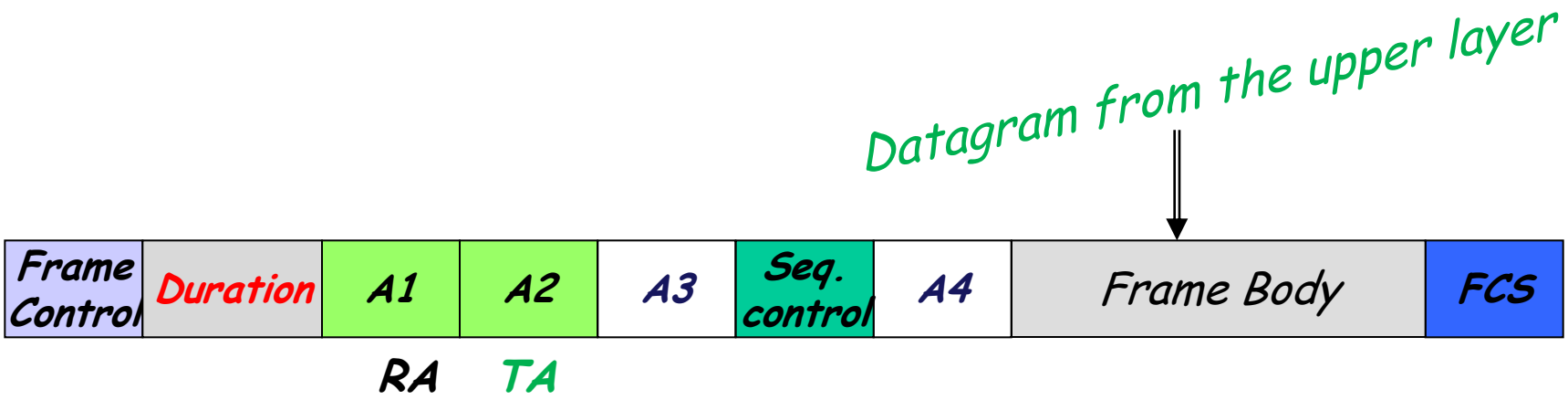


FCS: Frame Check Sequence ← CRC

RA: Receiver Address

TA: Transmitter Address

DATA Frame

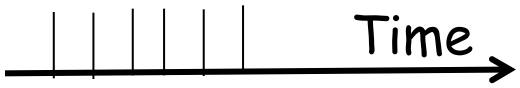


TA: Physically transmitting the frame.

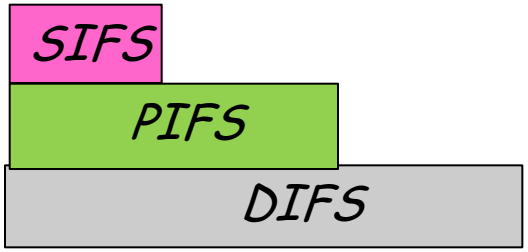
Timing Intervals

- The IEEE 802.11 MAC defines 4 timing intervals
  - *2 at the PHY level*
    - *SIFS: Short Inter-Frame Space (10 micro-sec) between successive frames (RTS, CTS, DATA, ACK, …)*
    - *aSlot (20 micro-sec)*
  - *2 at the MAC level*
    - *PIFS: Priority (in PCF) IFS (SIFS + aSlot)*
    - *DIFS: Distributed IFS (PIFS + aSlot)*

***aSlot** is chosen s.t. a node can determine if another node initiated a Tx aSlot time before (= propagation time + some hardware delay)*



Timing intervals are used to control priority

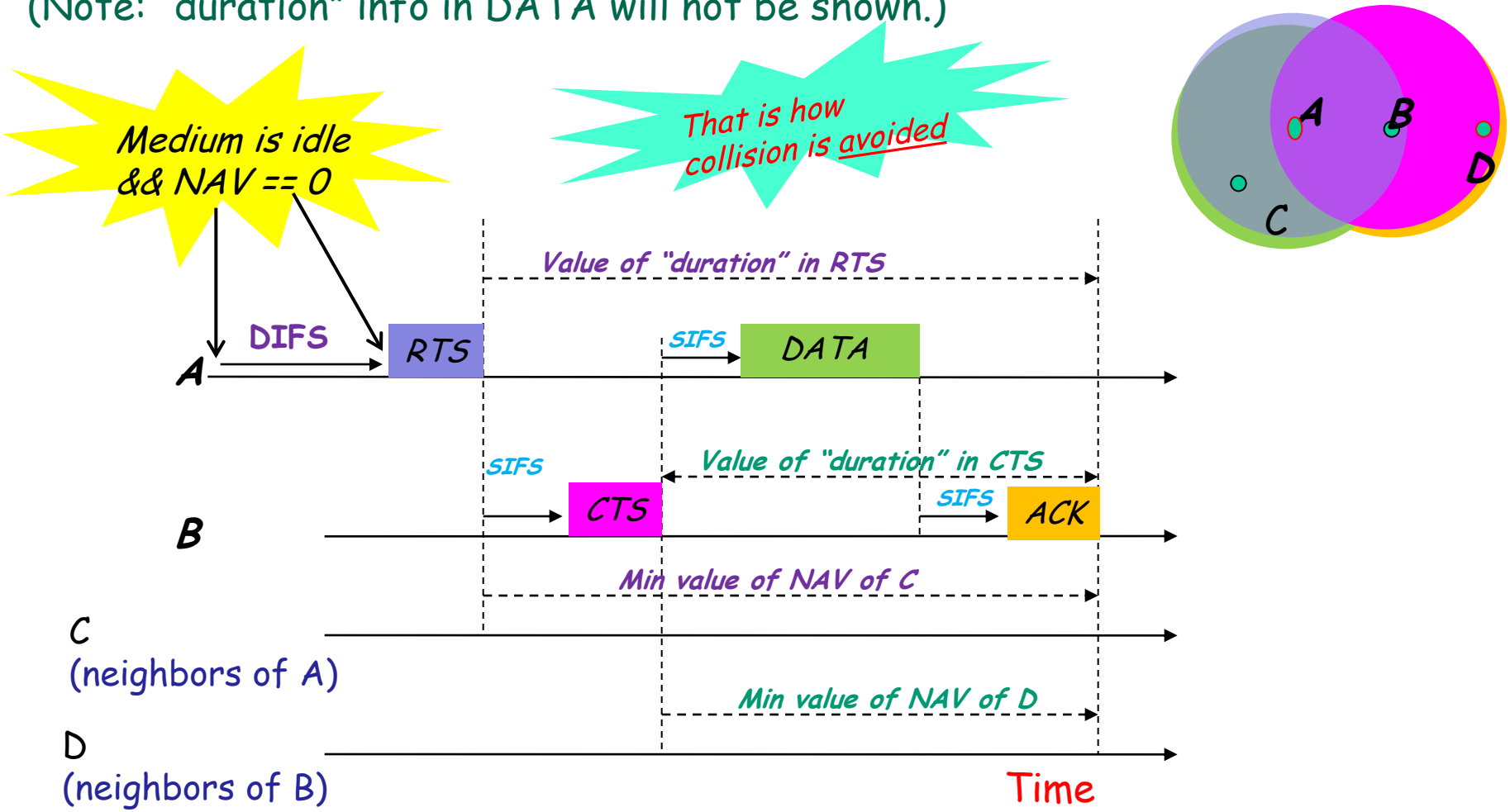


By keeping SIFS shortest, it is ensured that an ongoing cycle of Tx (with handshake or without handshake) is not disturbed ….
No one else can Tx between RTS, CTS, DATA, ACK.

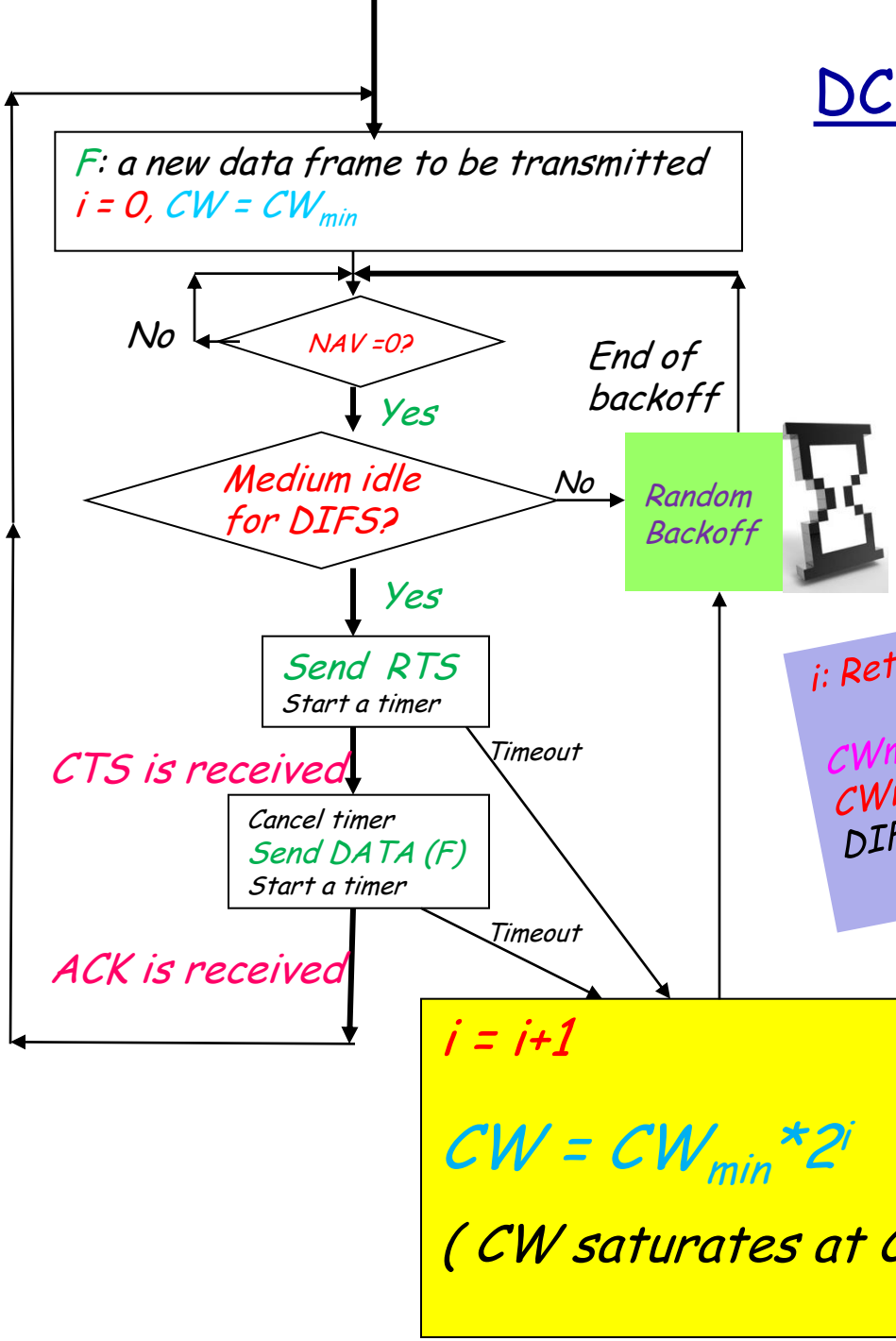
PIFS < DIFS enables an AP to become the controller of a BSS …

Handshake using RTS/CTS (A wants to send data to B)

(Note: "duration" info in DATA will not be shown.)



DCF with Hand-shake: Tx



i: Retry count, CW: Contention Window

CWmin: Min. value of CW (typical: 32)
CWmax: Max. value of CW (typical : 256)
DIFS: Distributed Interframe Space

Backoff Mechanism

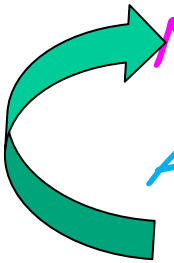
- **Initialize** a counter: Backoff Time Counter (BTC)
  - BTC = Random(0,CW-1) ……time unit of BTC is **aSlot**

- *As time passes, BTC is decremented as follows*

  At the start, let ch. remain idle for DIFS.

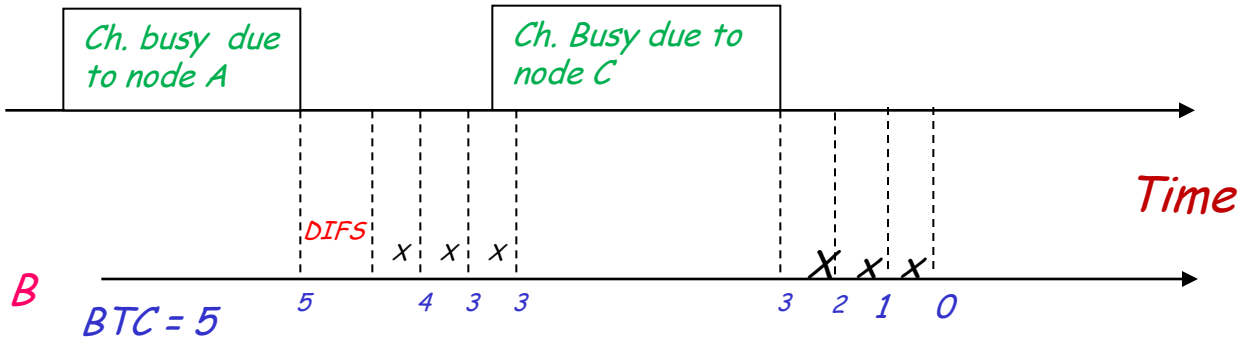
  Next, if ch. is idle for aSlot: BTC = BTC -1 ← *Repeat this*

  Anytime the medium is busy: Freeze BTC



- *BTC == 0* ➔ *End of backoff*

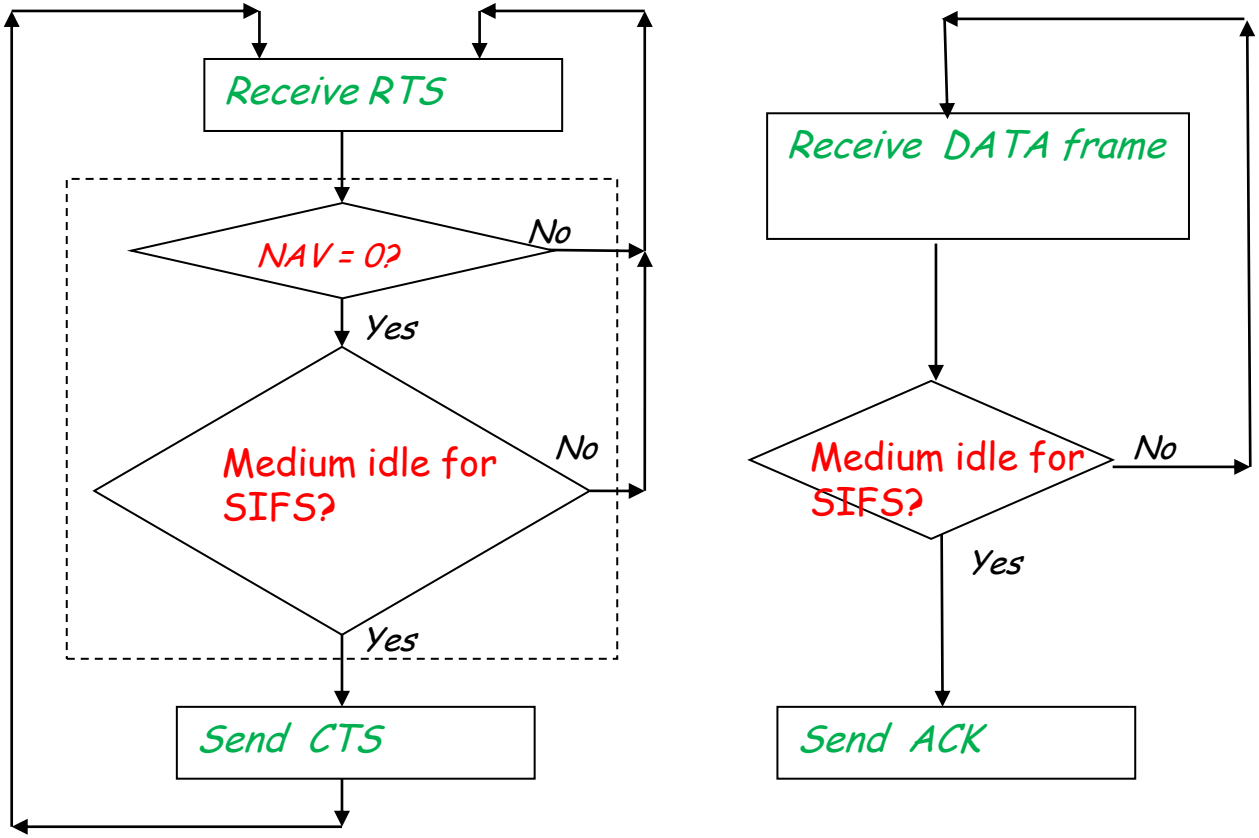
Backoff Mechanism



B is executing backoff

X = aSlotTime

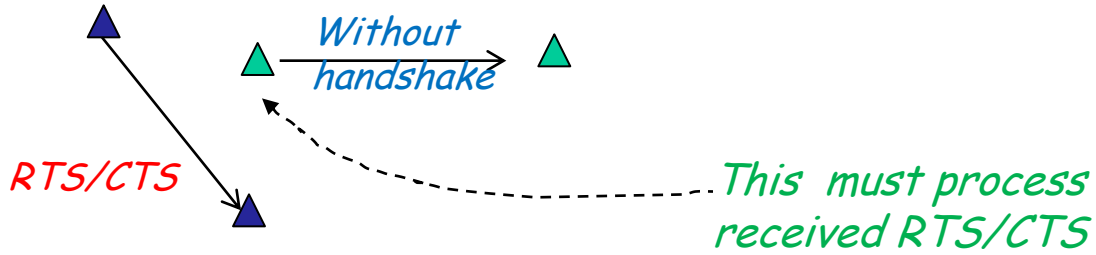
DCF with Handshake: Rx



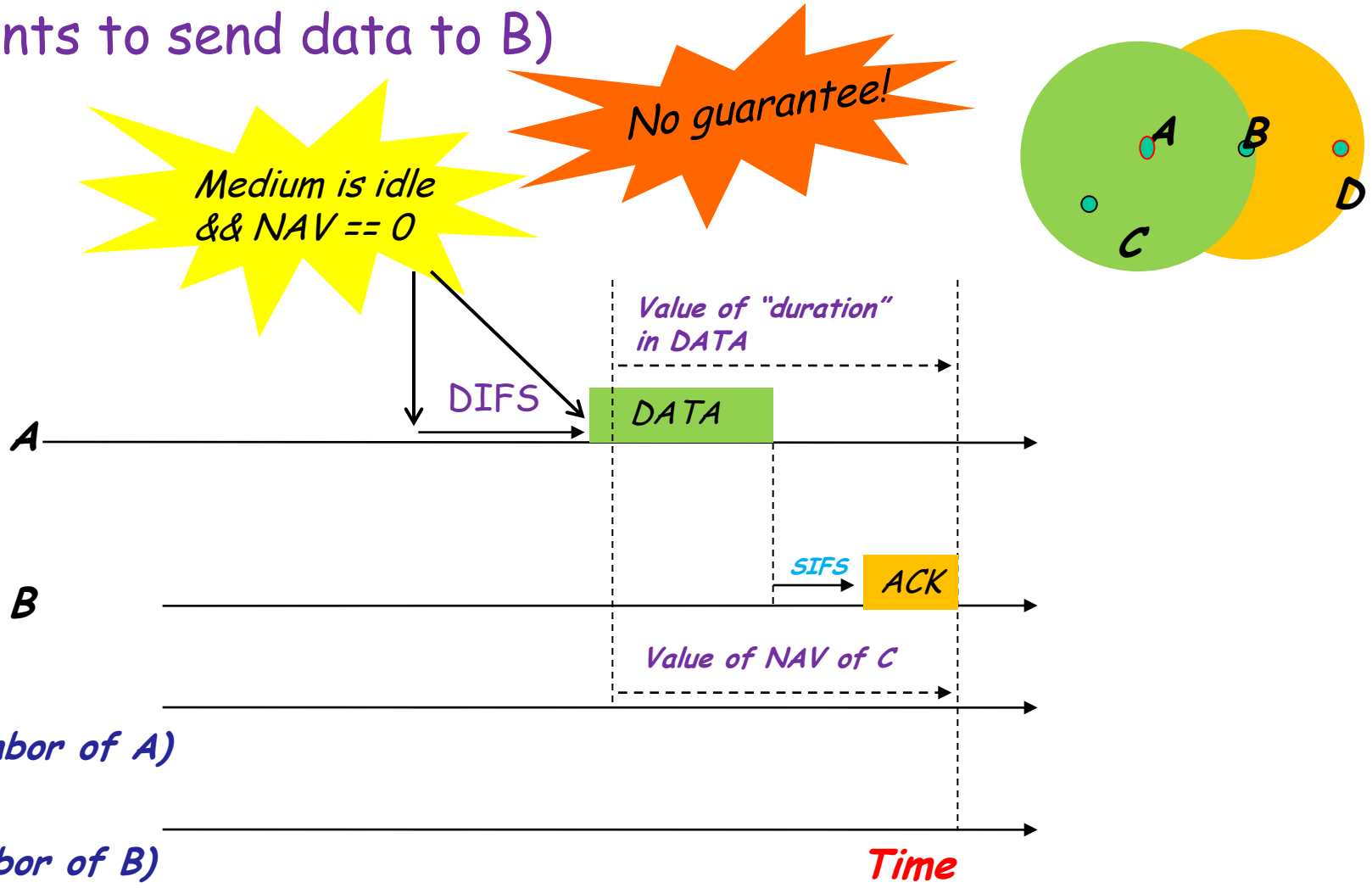
Note: The above two fragments of flow-charts can be easily merged.

DCF Mode without Handshake

- A special case of DCF with handshake
  - RTS/CTS frames are **not** exchanged.
- The idea of NAV is still used in this mode
  - Recall: All nodes process the received RTS/CTS of others

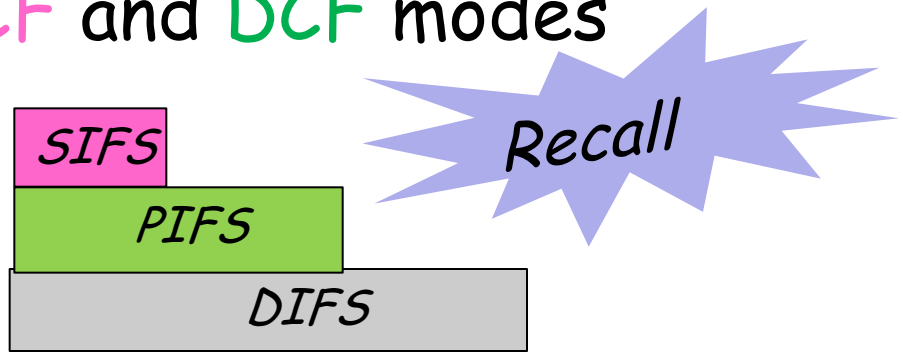


Data transmission without handshake
(A wants to send data to B)



PCF Mode: AP becomes the controller: How?

- AP alternates between PCF and DCF modes



- *AP operates as the **controller** as follows*

*If AP finds medium to be idle **for PIFS**, it transmits a beacon frame*

*Beacon contains a **CFPMaxDuration** field*

*Nodes receiving a beacon update their NAV to **CFPMaxDuration***

These nodes perceive the medium to be busy for CFPMaxDuration
They do not transmit unless asked to do so by the AP.

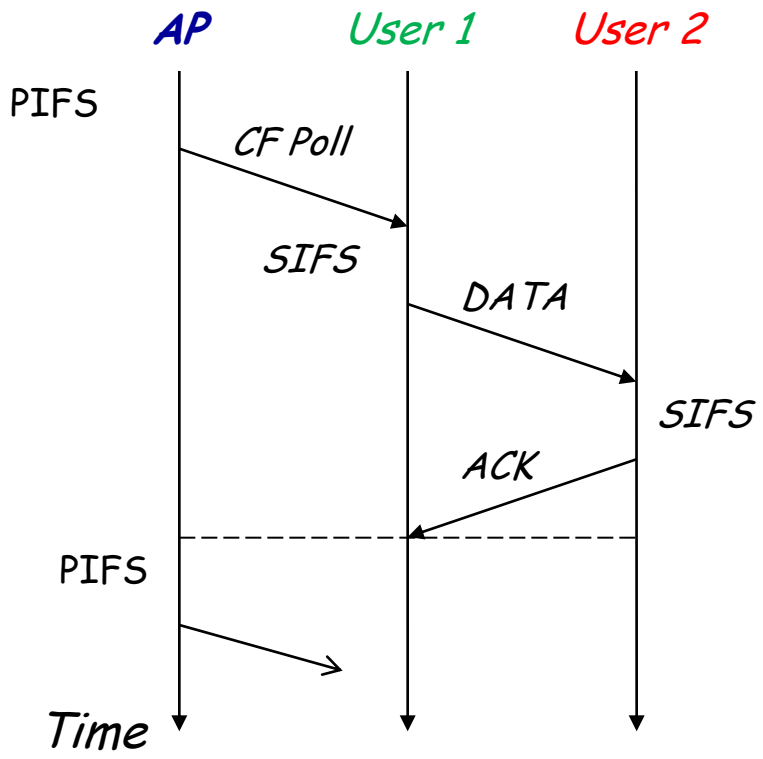
PCF Mode of Operation (Contd.)

- After **transmitting a beacon**, AP waits for PIFS before transmitting one of the following
  - DATA frame
  - CF Poll frame
  - DATA+CF Poll frame
  - ACK frame
  - CF End frame

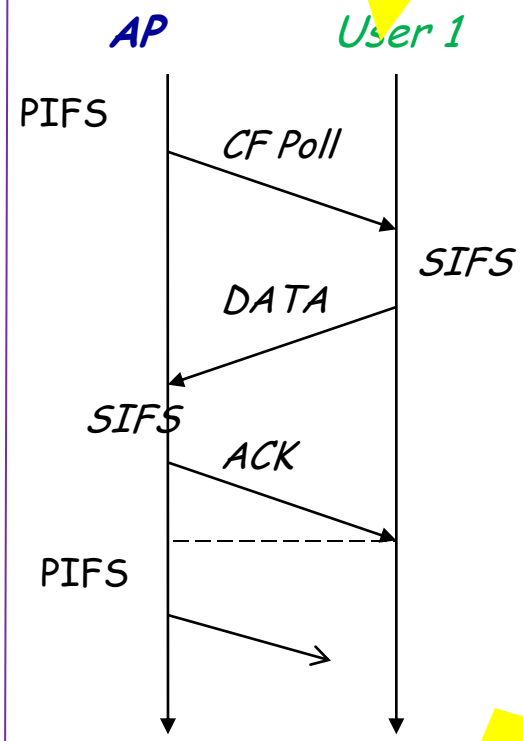
Contention Free

PCF Mode of Operation (Contd.)

CF Poll frame



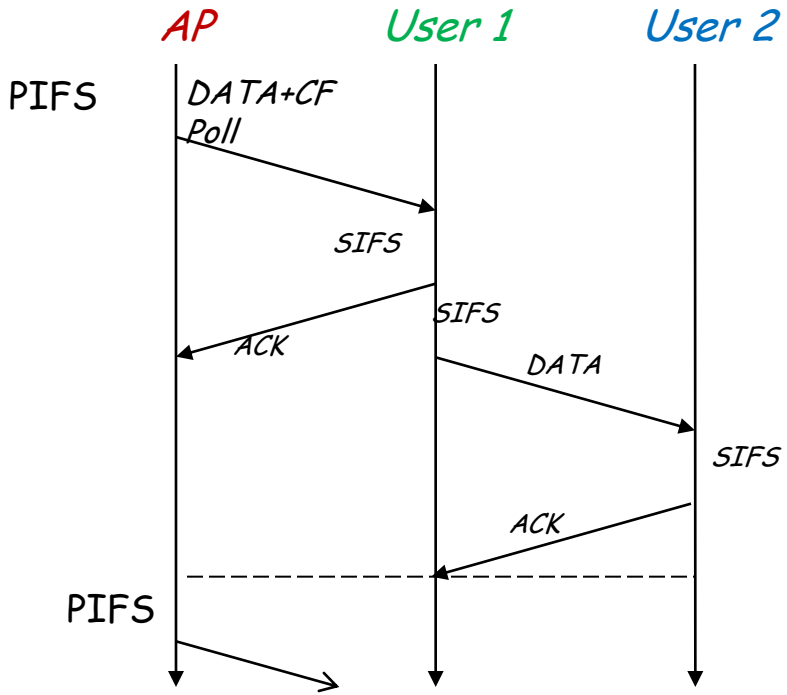
If a user does not have data, it sends a null DATA frame (A DATA frame with no actual data)



If ACK is not received, user 1 waits until it is polled again.

PCF Mode of Operation (Contd.)

DATA + CF Poll frame



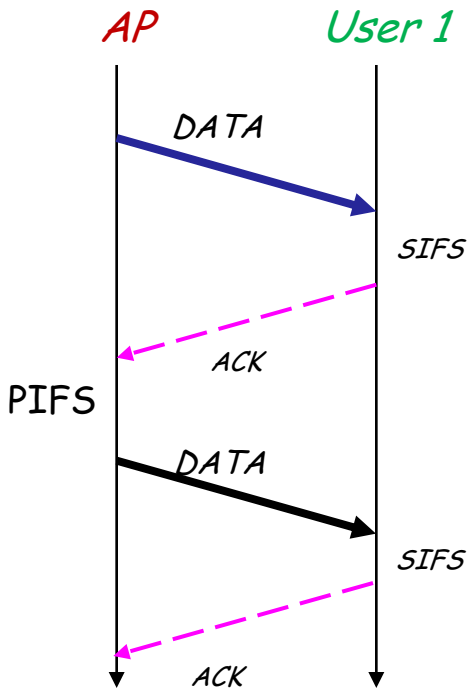
Note 1: If AP does not receive an ACK, it retransmits data after PIFS.

Note 2: If User 1 does not receive ACK, it does not retransmit data.

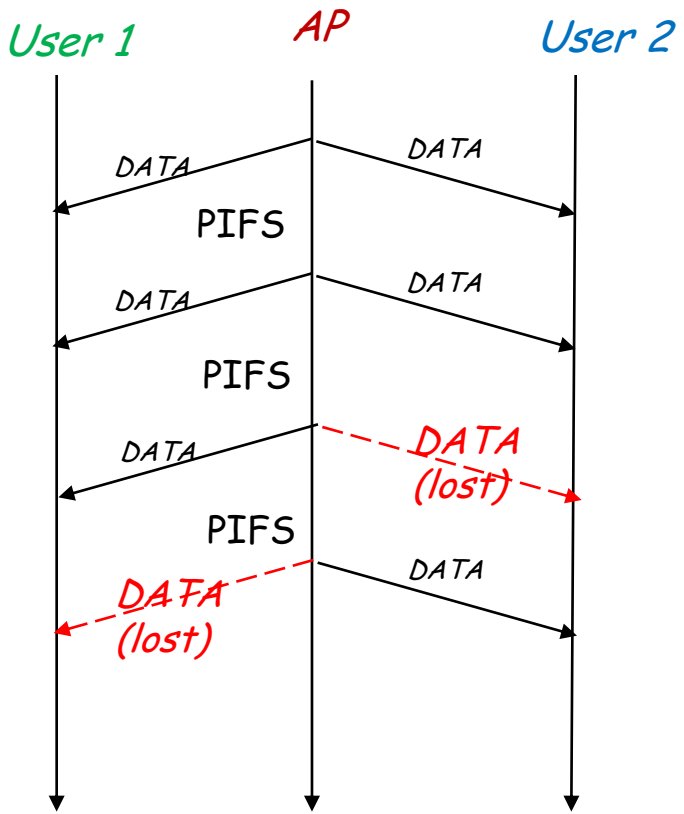
The polled user receives data from the AP and sends data to another user.

PCF Mode of Operation (Contd.)

DATA from AP (unicast): 1 → 1



DATA from AP (broadcast): 1 → all

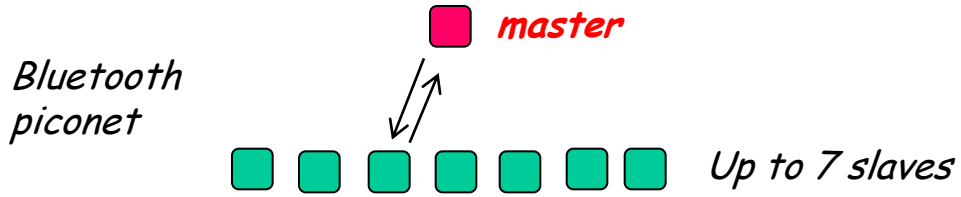


PCF Mode of Operation (Contd.)

- CF End frame
  - Identifies the end of CF period
    - Receiving nodes set NAV = 0.

Summary of MAC protocols

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



Could I see some past exam questions …

F'12 Mid-term

Identify five differences between the CSMA/CD and the CSMA/CA protocols by identifying five comparison criteria.

Use the following table to answer the question.

Criteria	CSMA/CD	CSMA/CA

More past exam questions …

F'12 Mid-term

Clearly explain why collision detection is not possible in wireless local area networks.

Clearly explain why the timing intervals SIFS, PIFS, and DIFS have been ordered as SIFS < PIFS < DIFS in the CSMA/CA protocol.

Clearly explain the hidden terminal problem

S'12 Mid-term

Link Layer

5.1 Introduction and services

5.2 Error detection and correction

5.3Multiple access protocols

5.4 Link-Layer Addressing

5.5 Ethernet

5.6 Link-layer switches

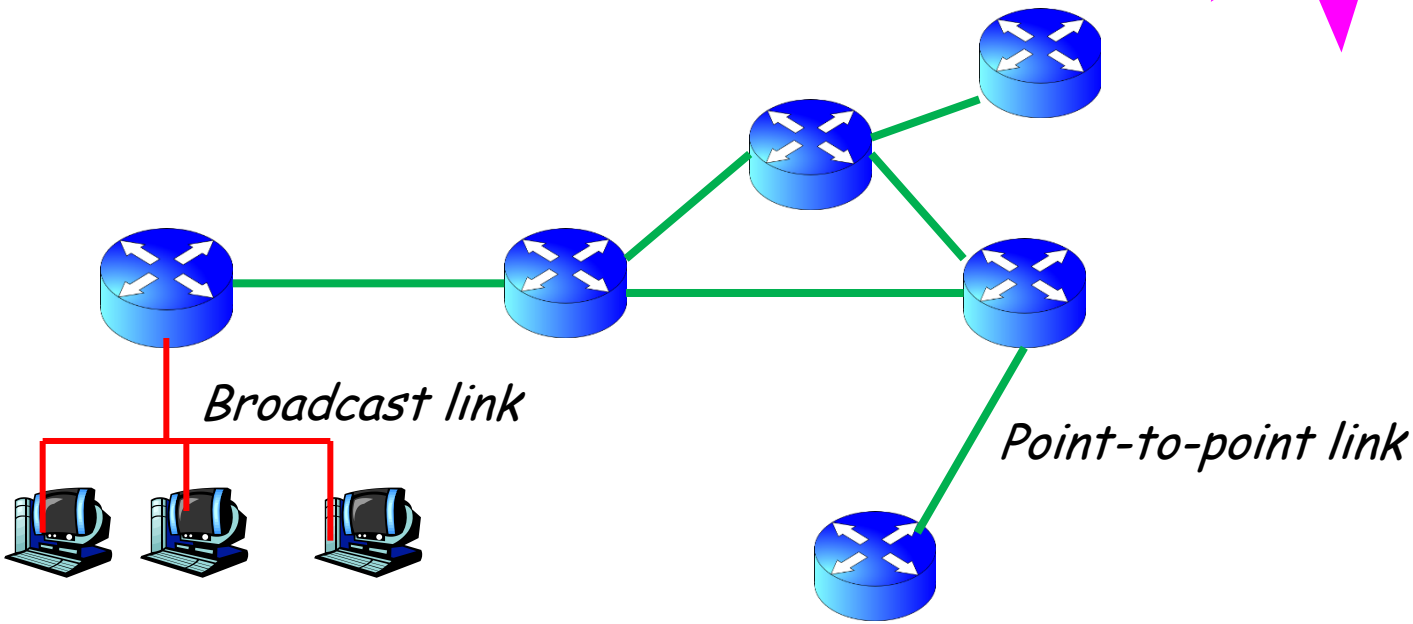
5.7 PPP

3.4 Reliable Data Transfer

I learned about accessing broadcast links … Ethernet …… and …. WiFi …..

Are there other kinds of links??

There are point-to-point links



Point to Point Data Link Control

- one sender, one receiver, one link: easier than broadcast link:
  - no Media Access Control
  - no need for explicit MAC addressing

- popular point-to-point DLC protocols:
  - PPP (point-to-point protocol)
  - HDLC: High level data link control (Data link used to be considered "high layer" in protocol stack!

PPP Design Requirements [RFC 1557]

- 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, signal link failure to network layer
- network layer address negotiation: endpoint can learn/configure each other's network address

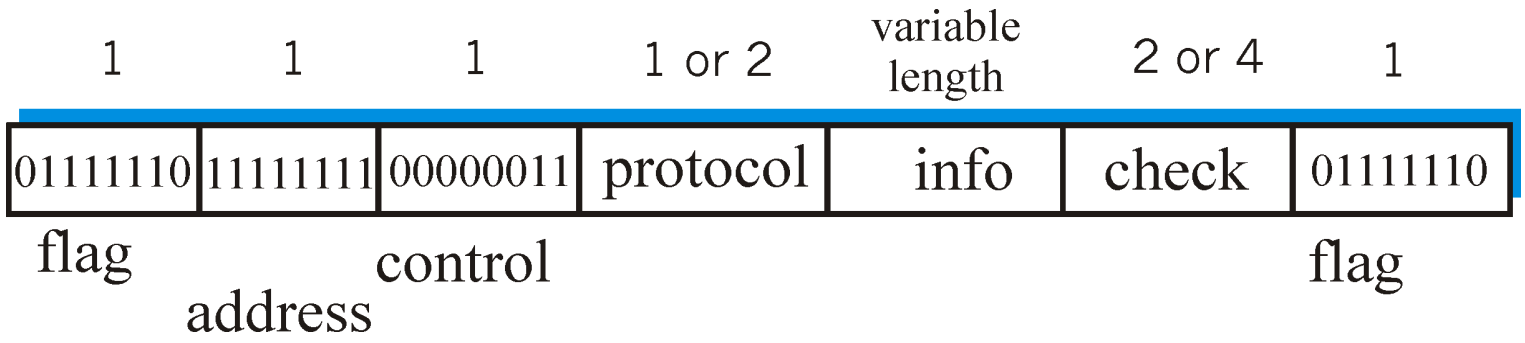
PPP non-requirements

- no error correction/recovery
- no flow control
- out of order delivery OK
- no need to support multipoint links (e.g., polling)

Error recovery, flow control, data re-ordering
all relegated to higher layers!

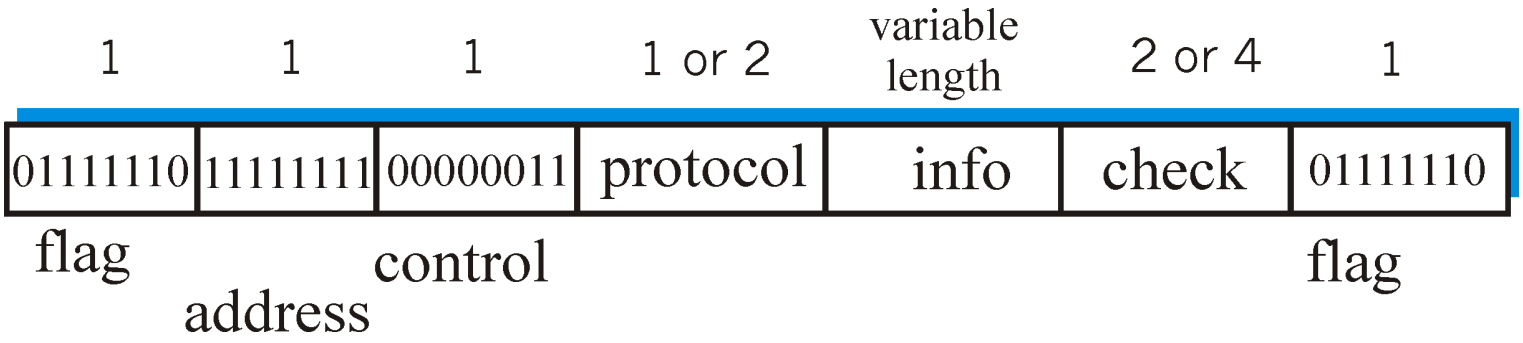
PPP Data Frame

- 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



PPP Data Frame

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

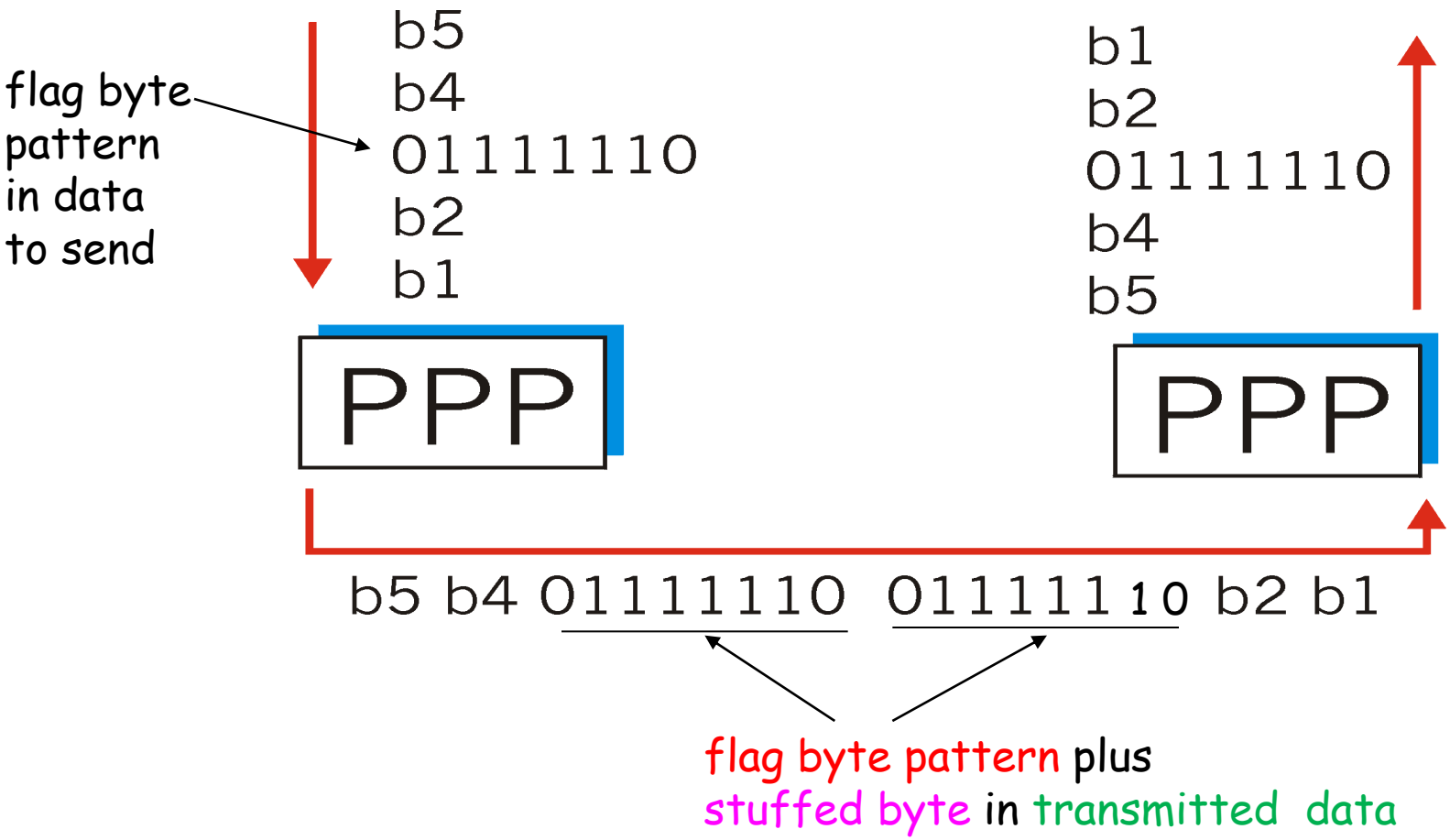


Byte Stuffing

- "data transparency" requirement: data field must be allowed to include flag pattern <01111110>
  - Q: is received <01111110> data or flag?

- Sender: adds ("stuffs") extra < 01111110> byte after each < 01111110> *data* byte
- Receiver:
  - two 01111110 bytes in a row: discard first byte, continue data reception
  - single 01111110: flag byte

Byte Stuffing



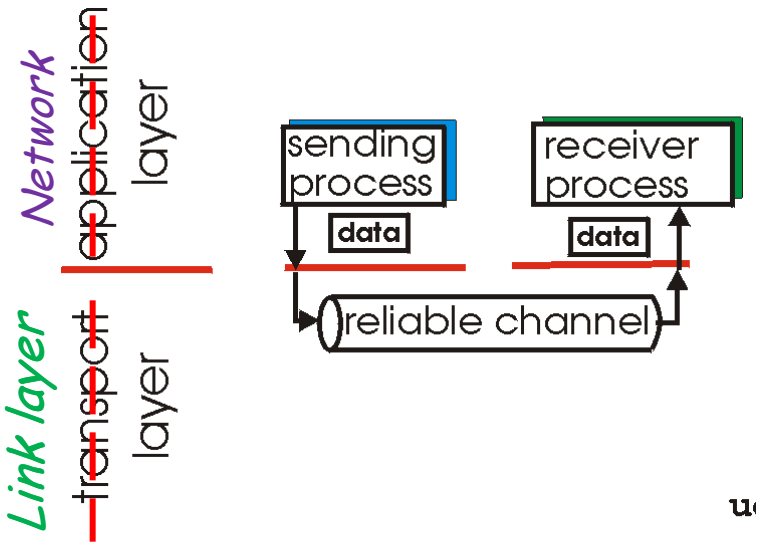
Link Layer

- 5.1 Introduction and services
- 5.2 Error detection and correction
- 5.3Multiple access protocols
- 5.4 Link-Layer Addressing
- 5.5 Ethernet
- 5.6 Link-layer switches
- 5.7 PPP

- 3.4 Reliable data transfer

3.4 Principles of Reliable data transfer

- important in app., transport, link layers

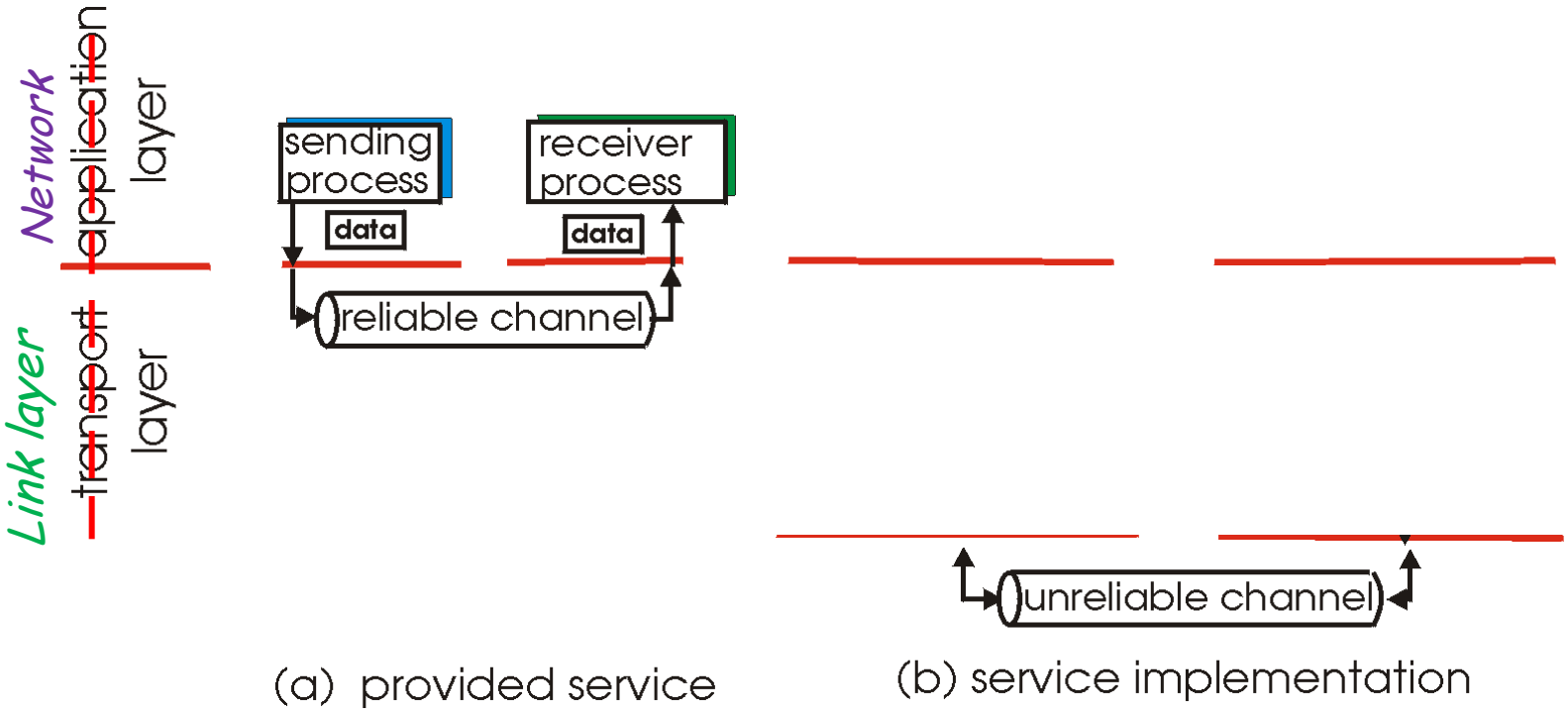


(a) provided service (b) service implementation

- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of Reliable data transfer

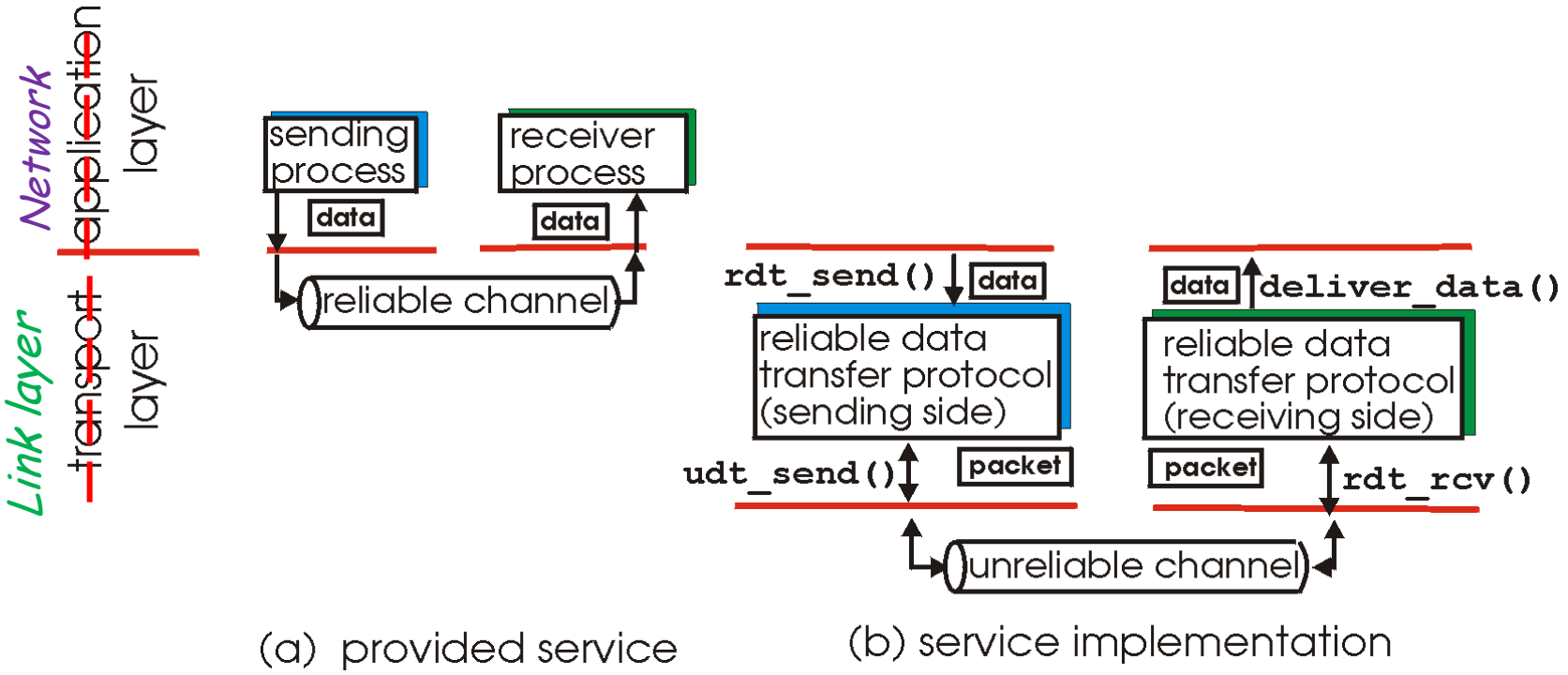
- important in app., transport, link layers



- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

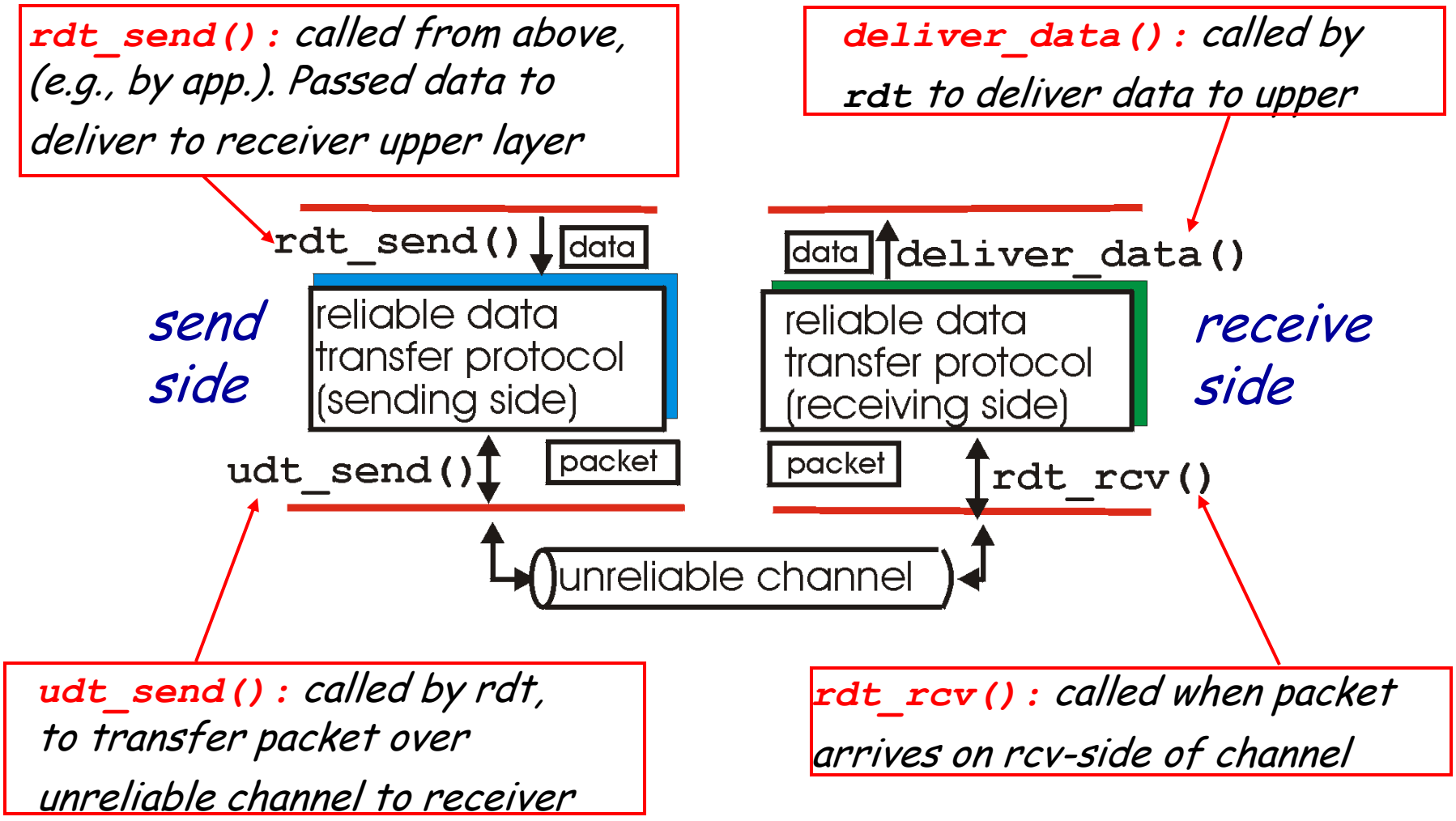
Principles of Reliable data transfer

- important in app., transport, link layers



- characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

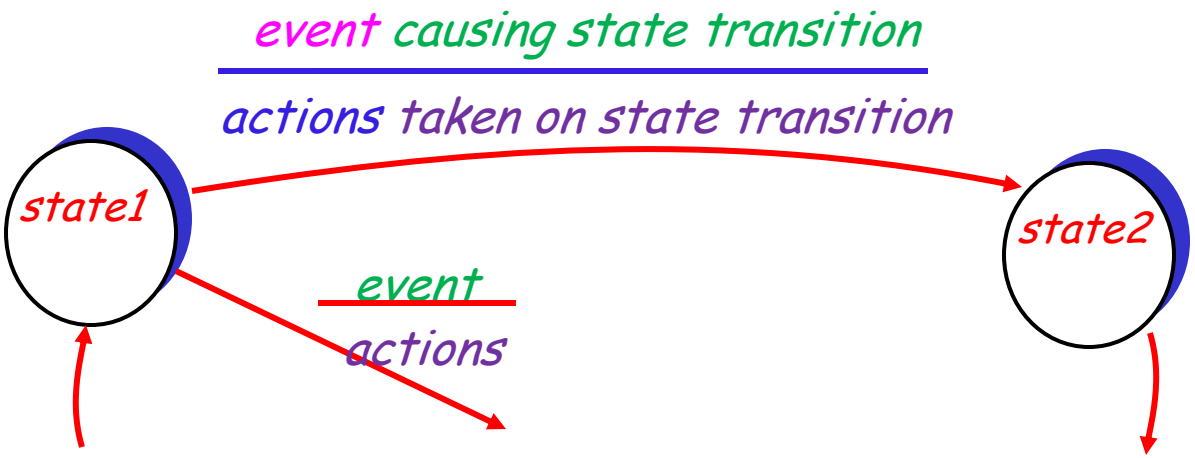
Reliable data transfer: getting started



Reliable data transfer: getting started

We will:

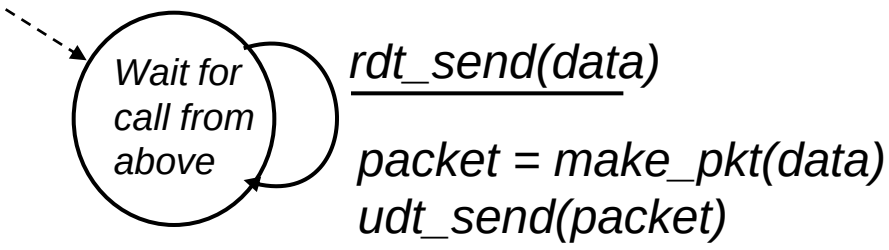
- incrementally develop sender, receiver sides of reliable data transfer protocol (rdt)
- consider only unidirectional data transfer
  - but control info will flow in both directions!
- use finite state machines (FSM) to specify sender, receiver



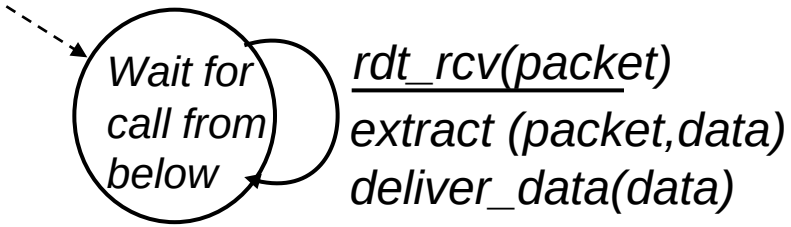
Rdt1.0: reliable transfer over a reliable channel

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

Worse can happen
Packet duplication
Packet reordering



sender



receiver

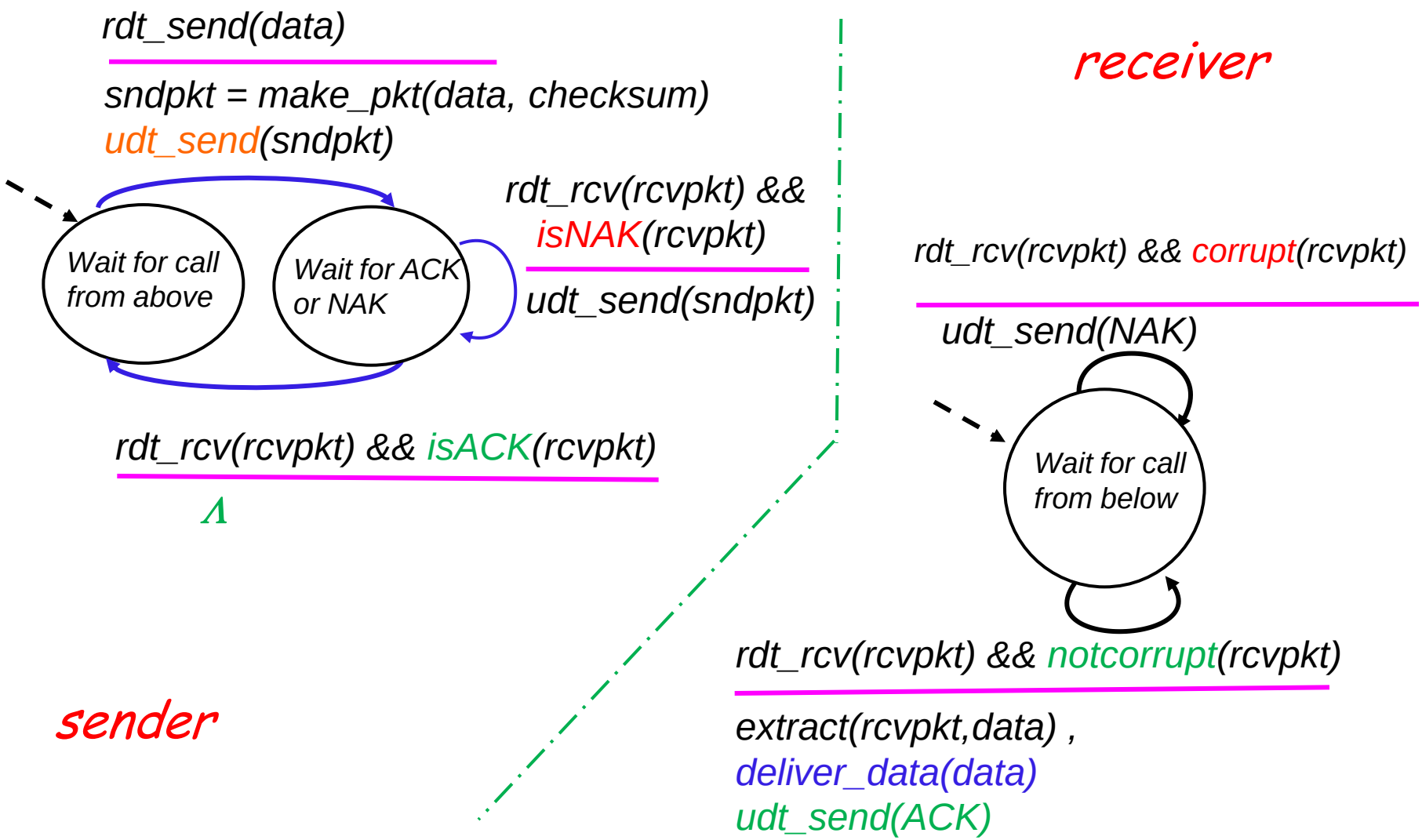
Rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
  - checksum to detect bit errors

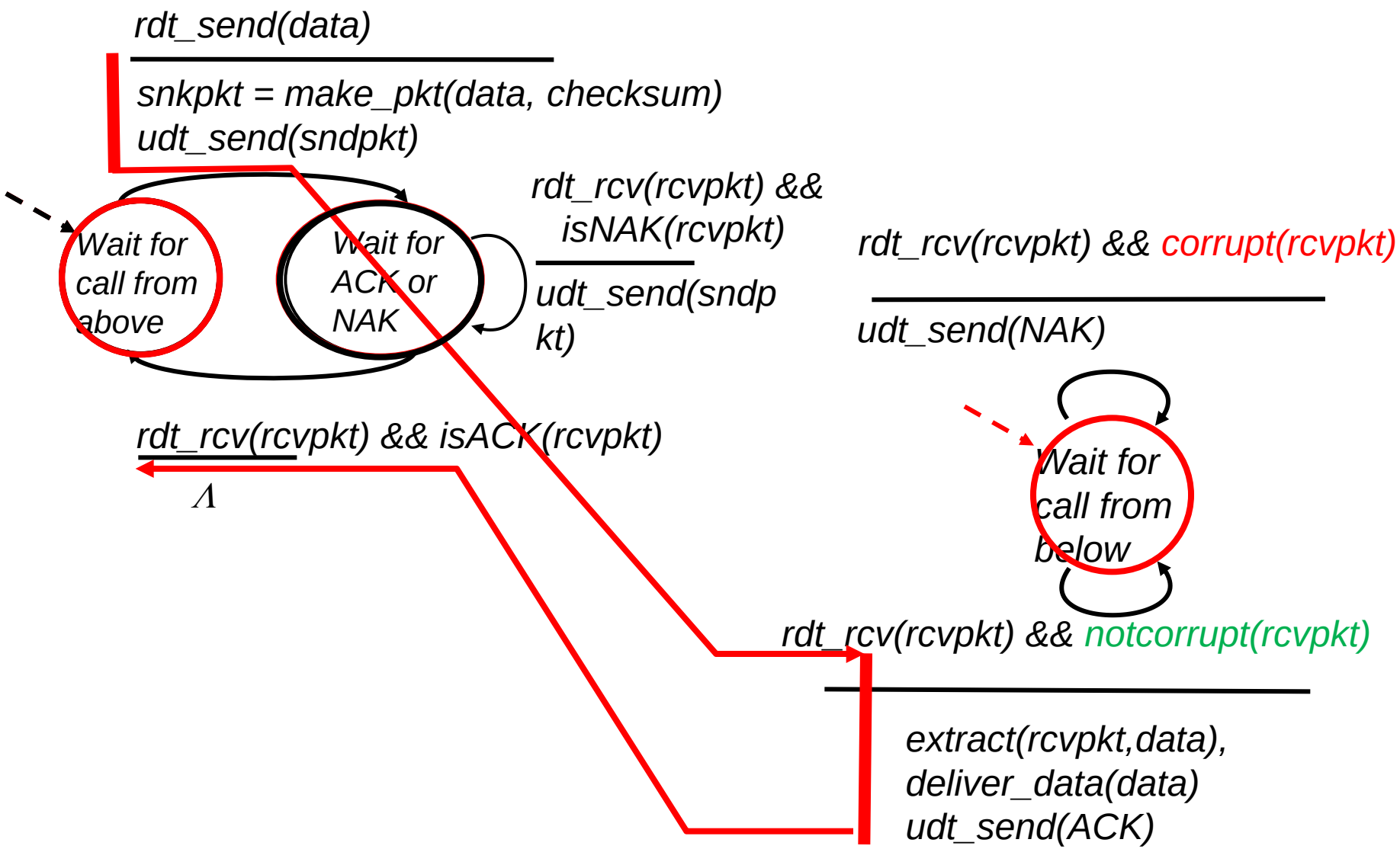
- the question: how to recover from errors:
  (Note: Correction by receiver is very expensive ….)
  - acknowledgements (ACKs): receiver explicitly tells sender that pkt received is OK
  - negative acknowledgements (NAKs): receiver explicitly tells sender that pkt had errors
  - sender retransmits pkt on receipt of NAK
  - sender transmits next pkt on receipt of ACK

- new mechanisms in `rdt2.0` (beyond `rdt1.0`):
  - error detection
  - receiver feedback: control msgs (ACK,NAK)

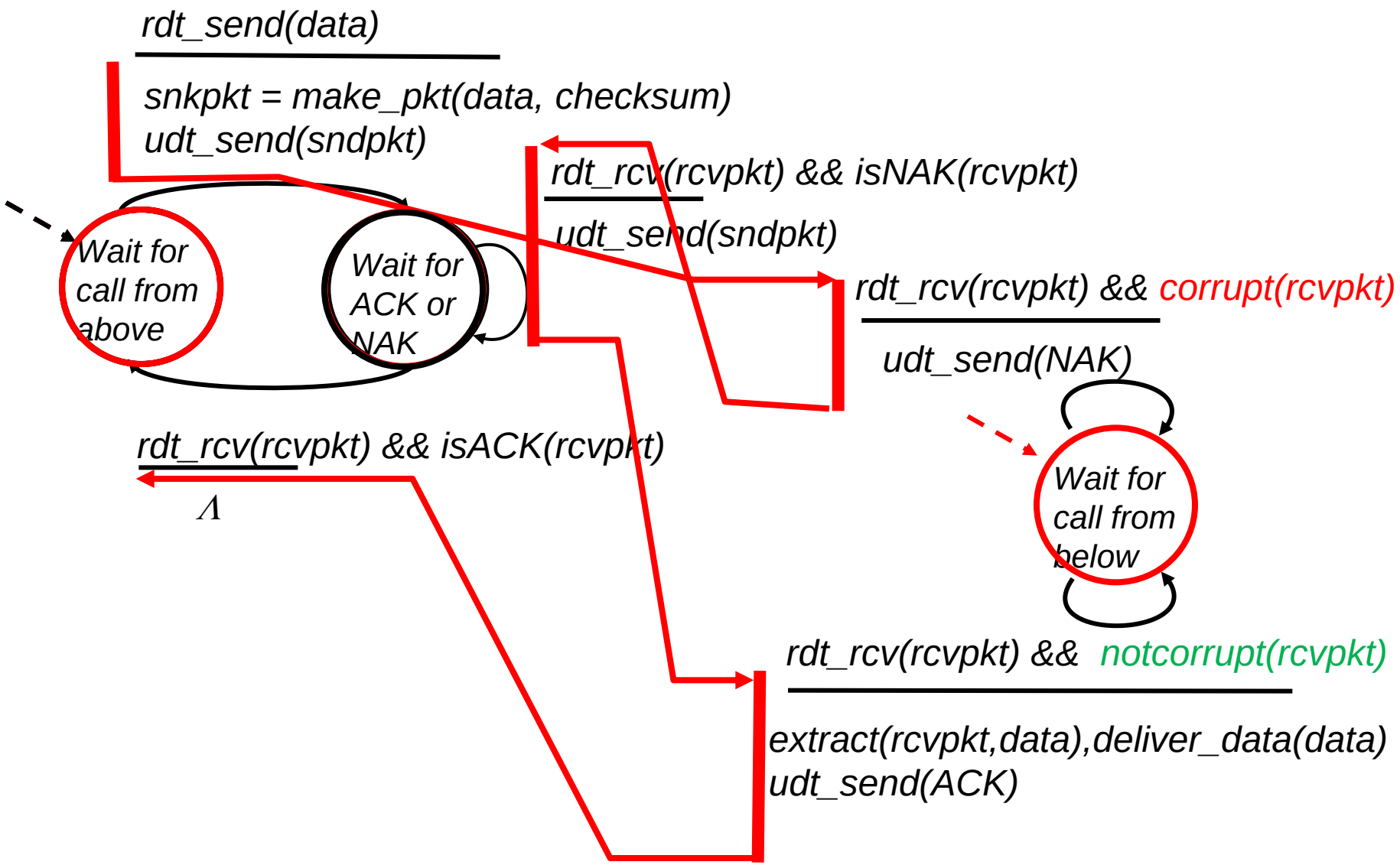
rdt2.0: FSM specification



rdt2.0: operation with no errors



rdt2.0: error scenario



rdt2.0 has a fatal flaw!

What if ACK/NAK corrupted?

- sender doesn't know what happened at receiver!
- can't just retransmit: possible duplicate

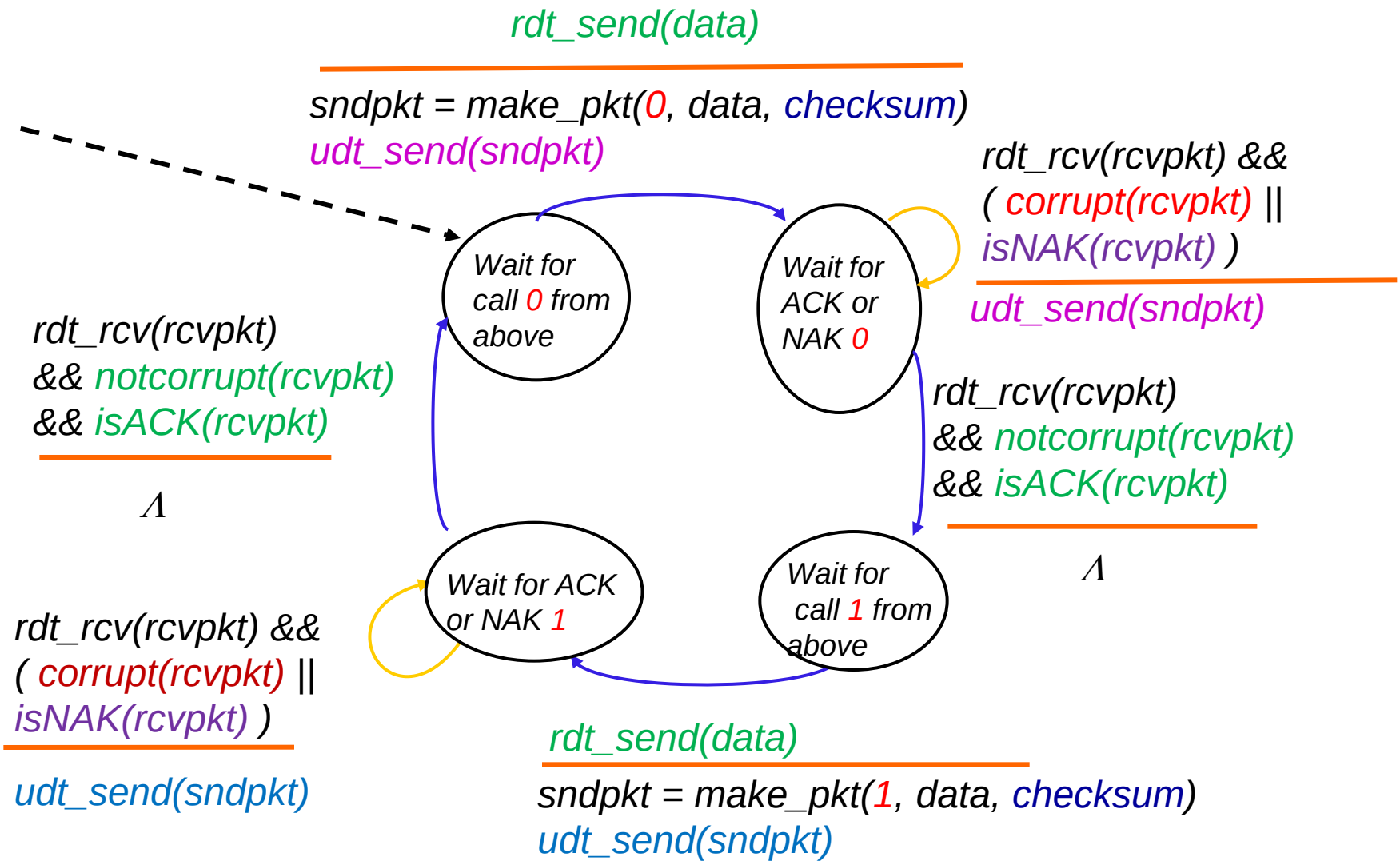
Handling duplicates:

- sender retransmits current pkt if ACK/NAK garbled
- sender adds ***sequence number*** to each pkt
- receiver discards duplicate pkt

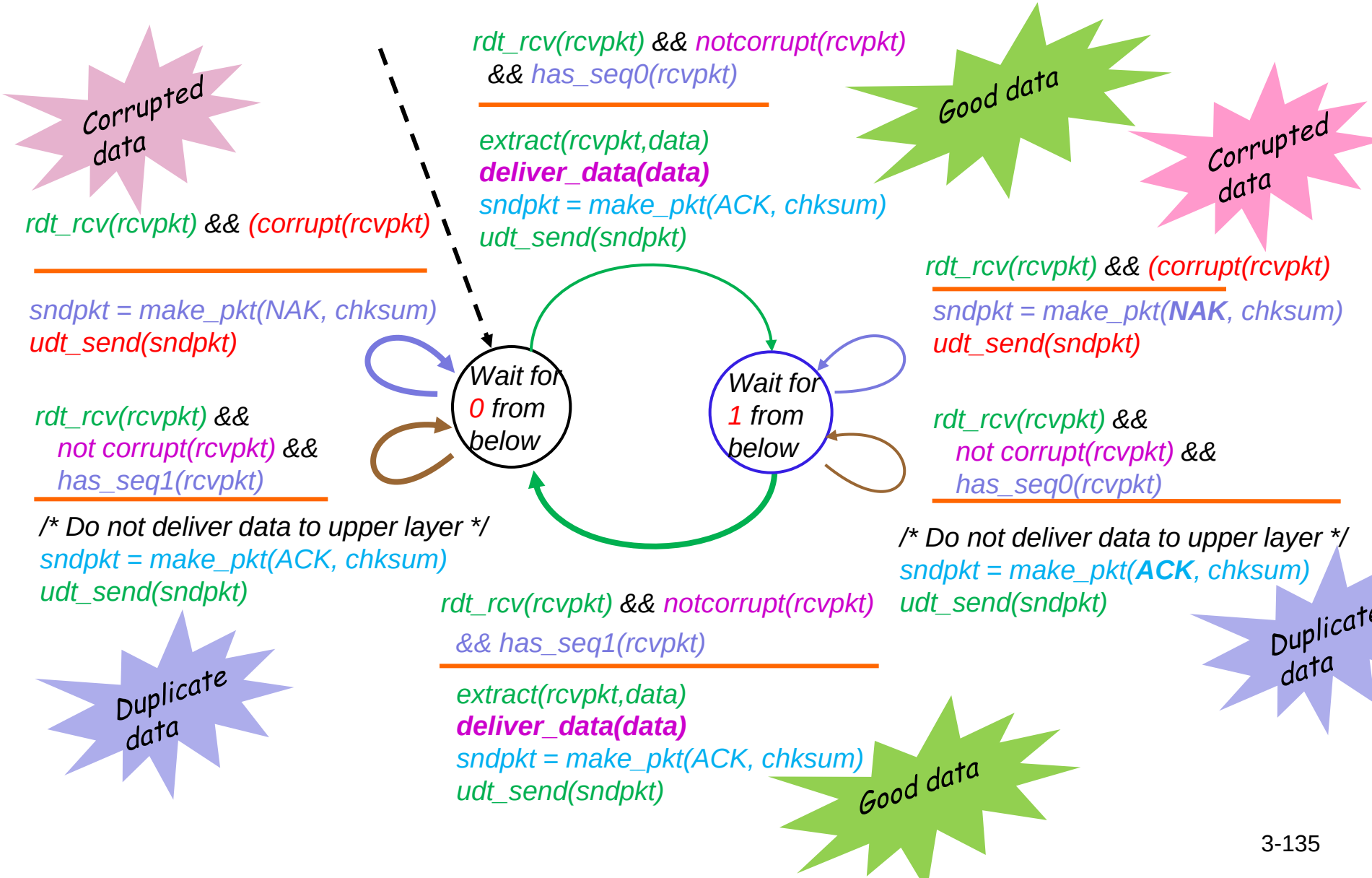
stop and wait
Sender sends one packet, then waits for receiver response

Note: Because of stop-and-wait, you need just two sequence numbers …… {0, 1} …..
Less than 2 is not a sequence number ….

rdt2.1: sender, handles garbled ACK/NAKs



rdt2.1: **receiver**, handles (garbled ACK/NAKs)



rdt2.1: discussion

Sender:

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

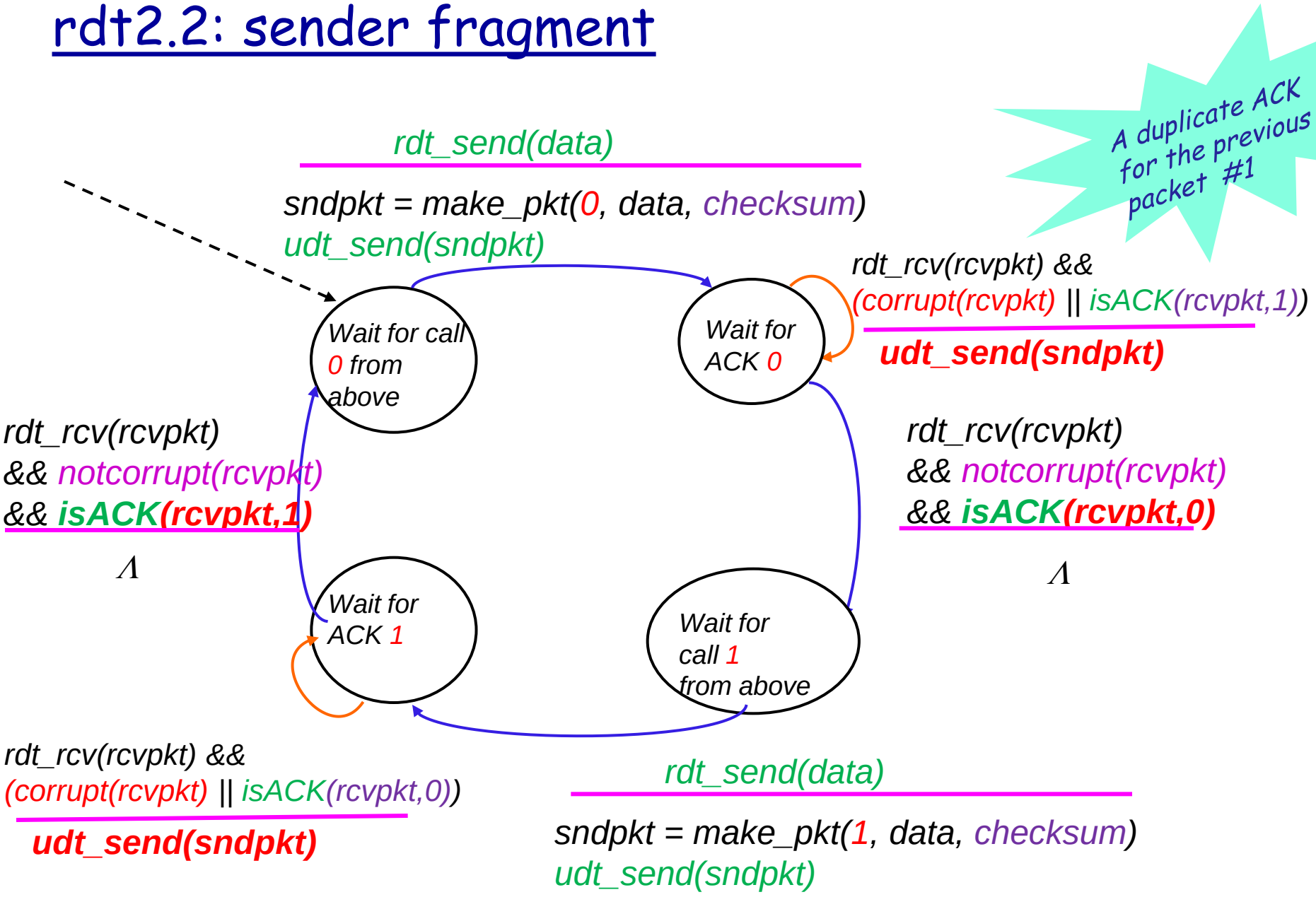
Receiver:

- 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 was received OK at sender

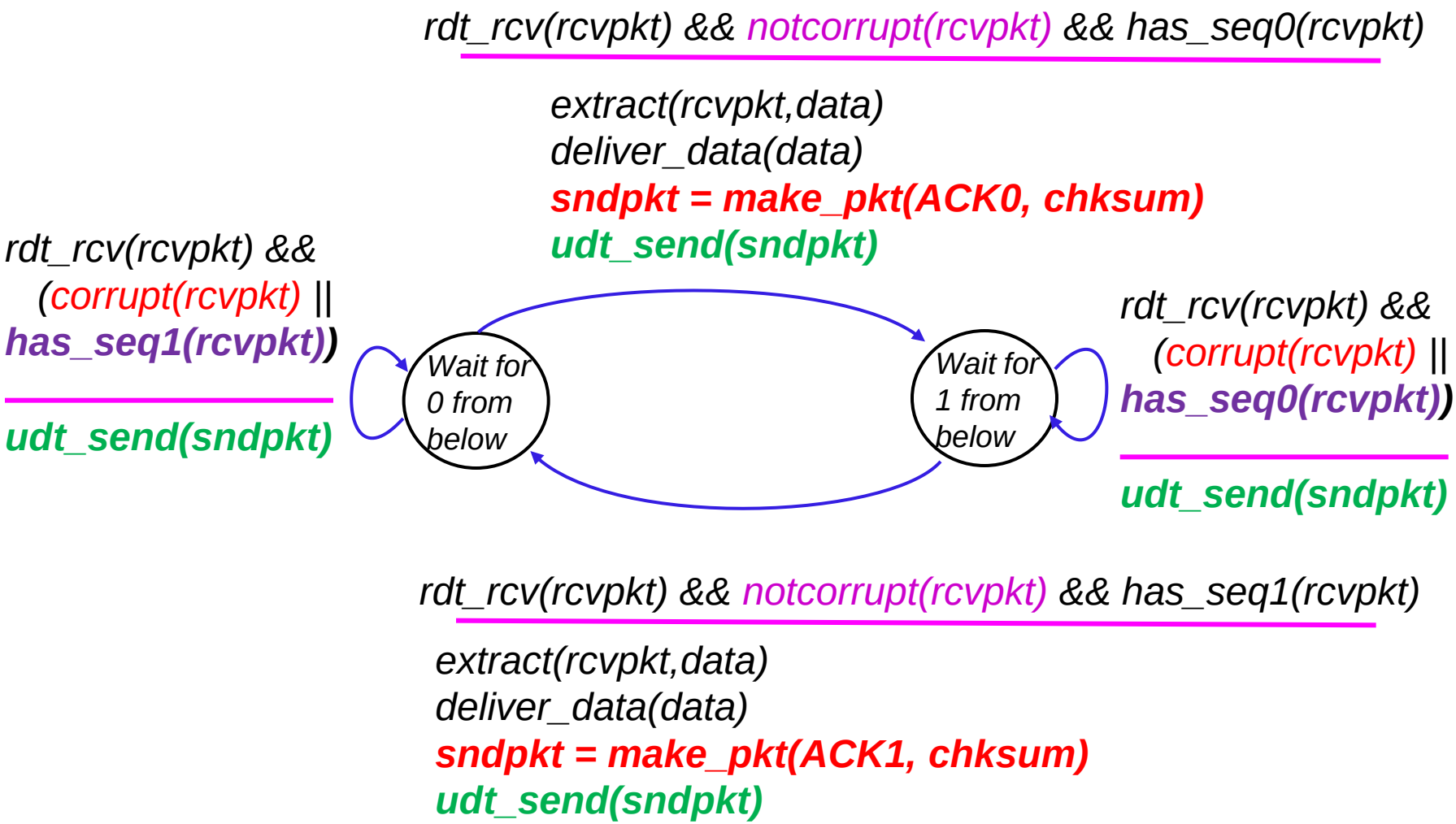
rdt2.2: a NAK-free protocol

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

rdt2.2: sender fragment



rdt2.2: receiver fragment



rdt3.0: channels with errors *and* loss

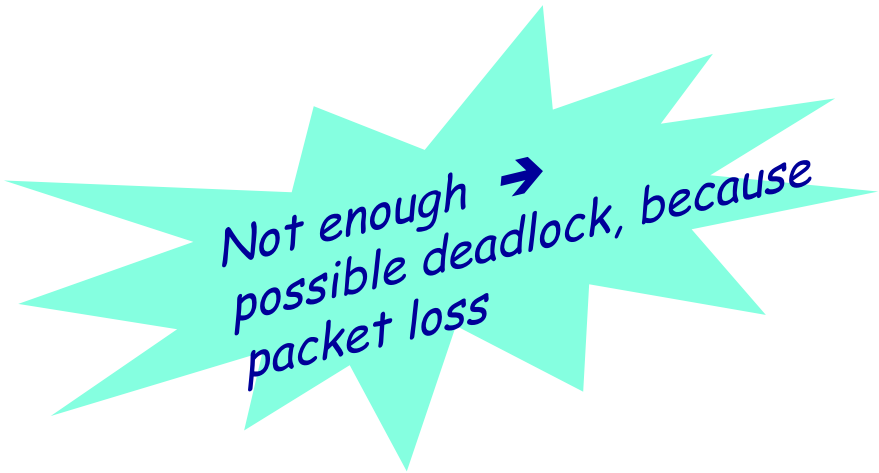
New assumption:

Underlying channel can also lose packets (data or ACKs)

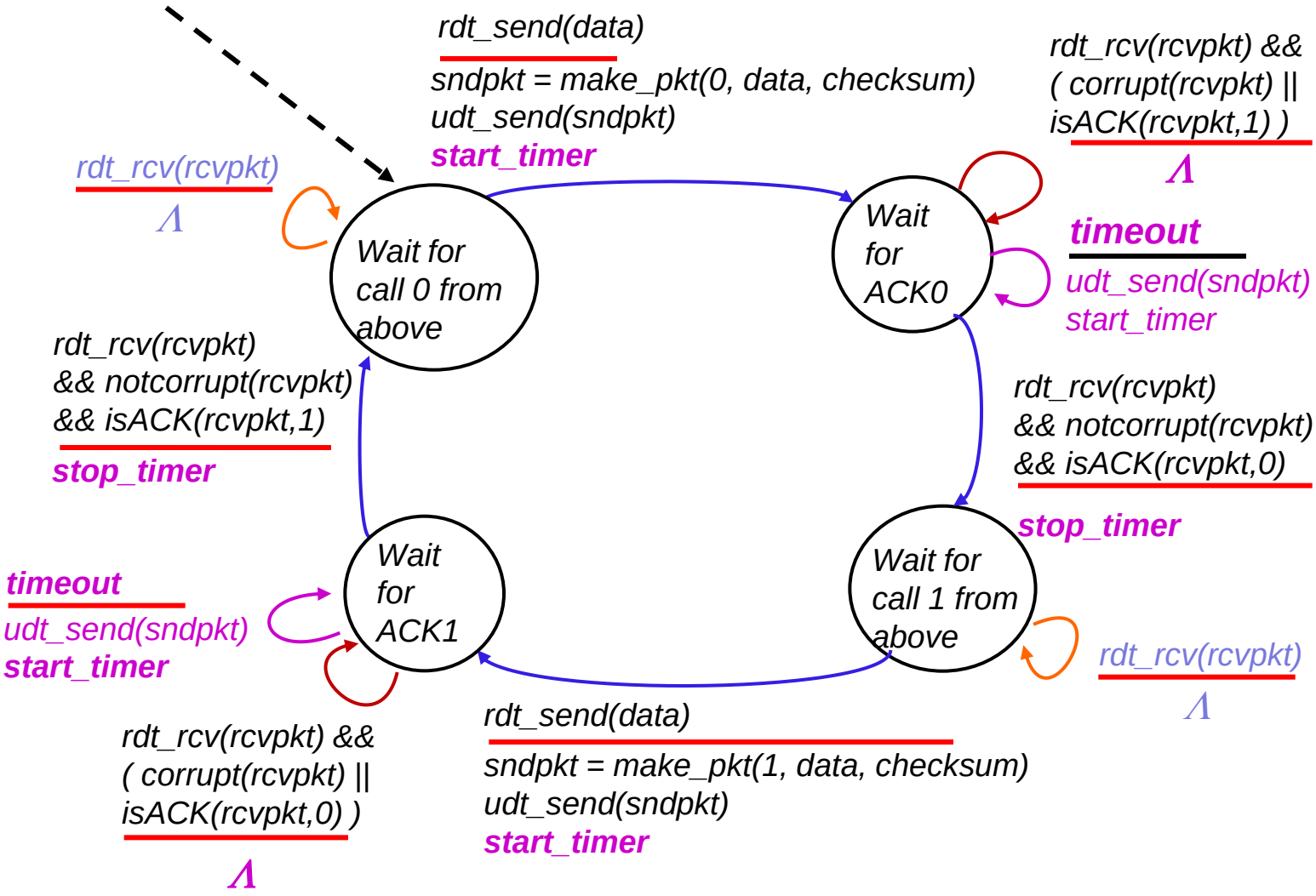
- checksum, seq. #, ACKs, retransmissions will be of help, but not enough

Approach: sender waits "reasonable" amount of time for ACK

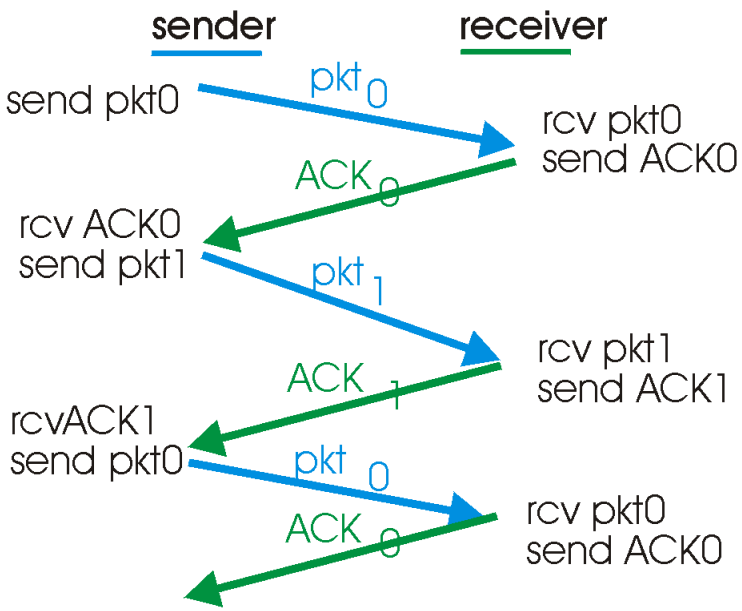
- retransmits if no ACK is received in this time
- if pkt (or ACK) just delayed (not lost):
  - retransmission will be duplicate, but use of seq. #'s already handles this



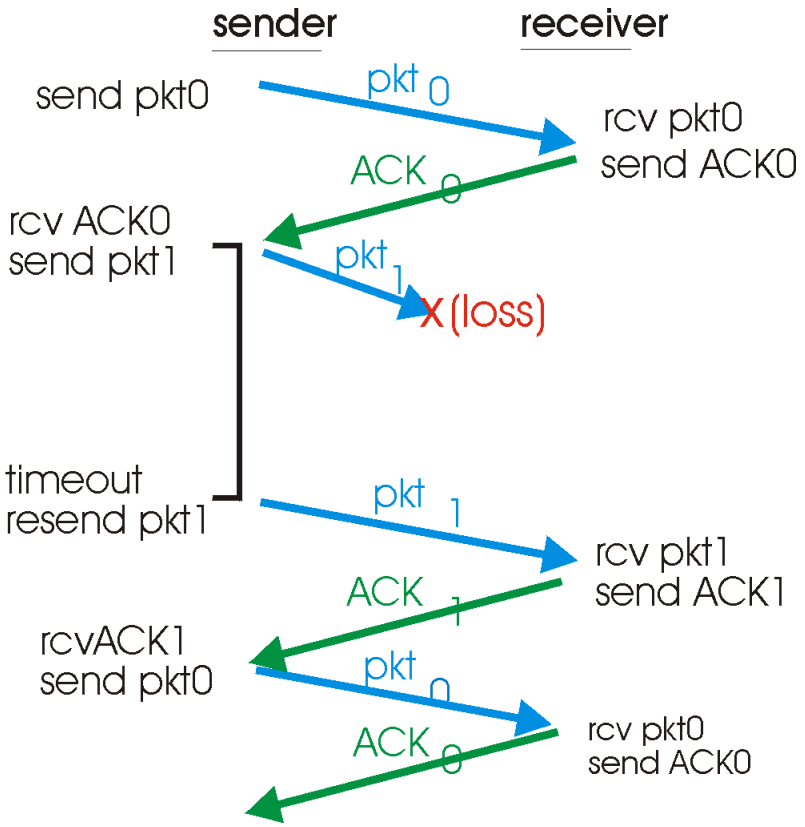
rdt3.0 sender (rdt2.2 + Timer + minor changes)



rdt3.0 in action

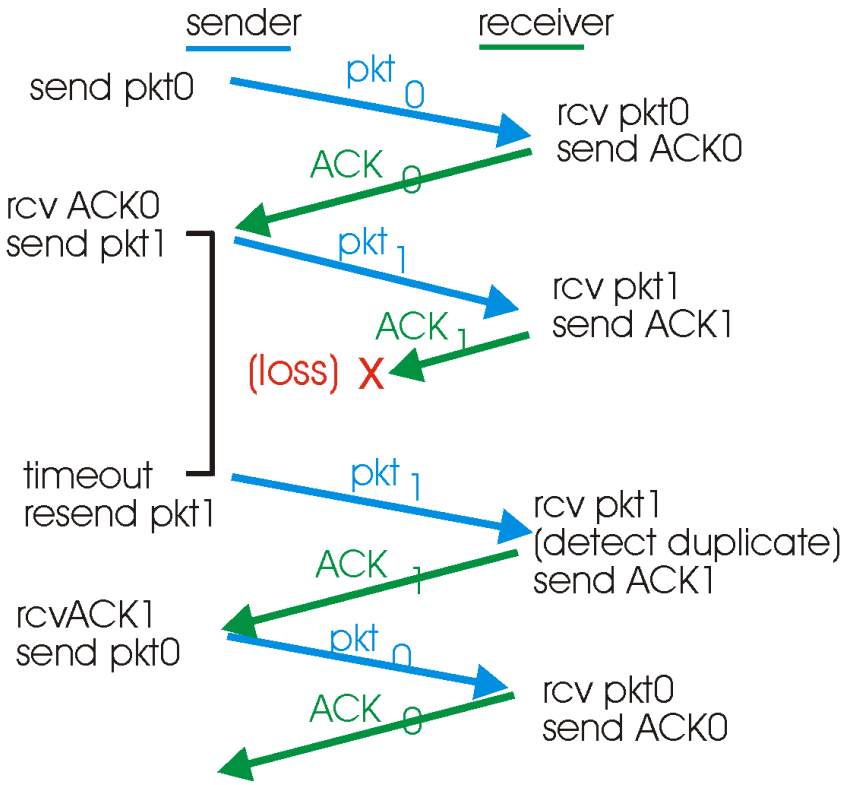


(a) operation with no loss

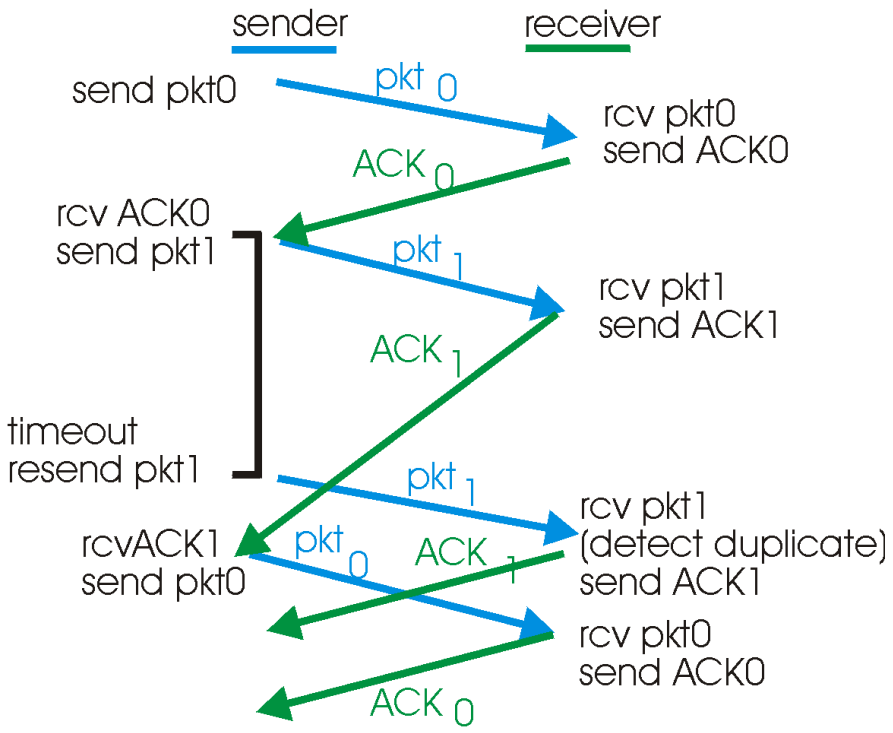


(b) lost packet

rdt3.0 in action



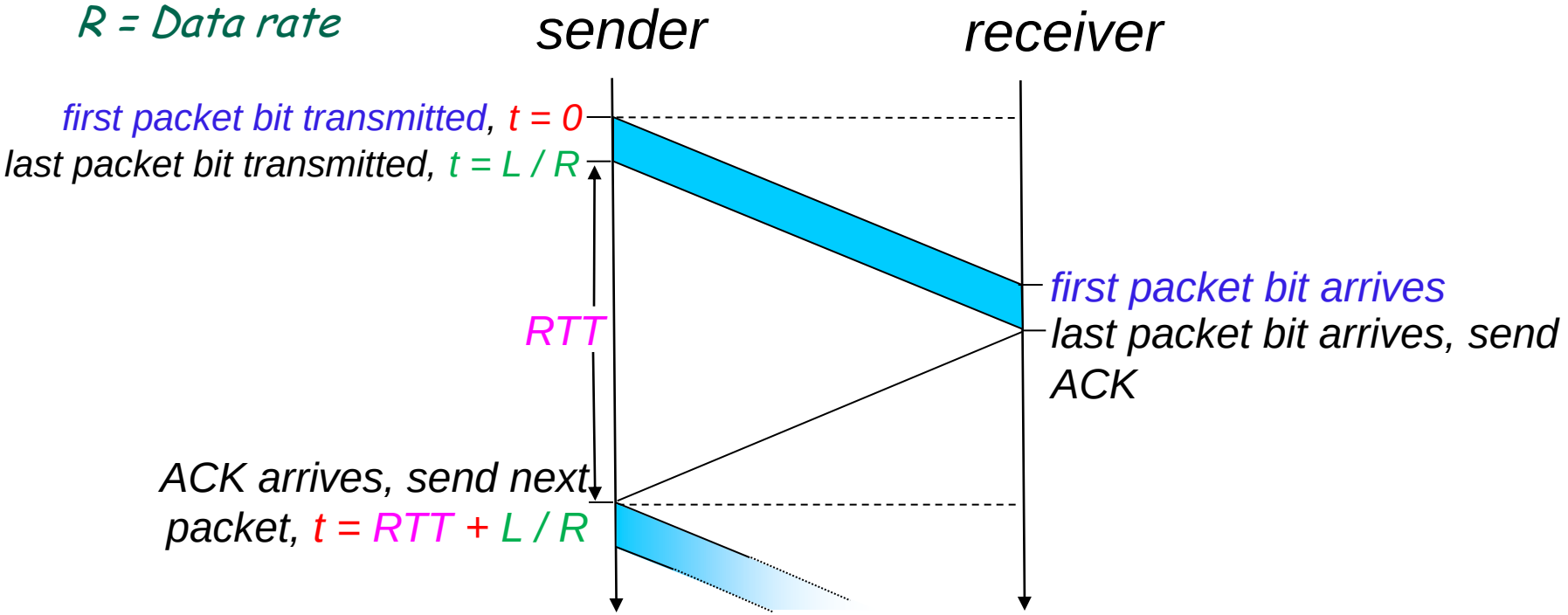
(c) lost ACK



(d) premature timeout

rdt3.0: stop-and-wait operation

L = Packet length
R = Data rate



RTT: Round-Trip Time = 2 x propagation delay.

Performance of rdt3.0

- ex: 1 Gbps link, 15 ms prop. delay, 8000 bit packet:

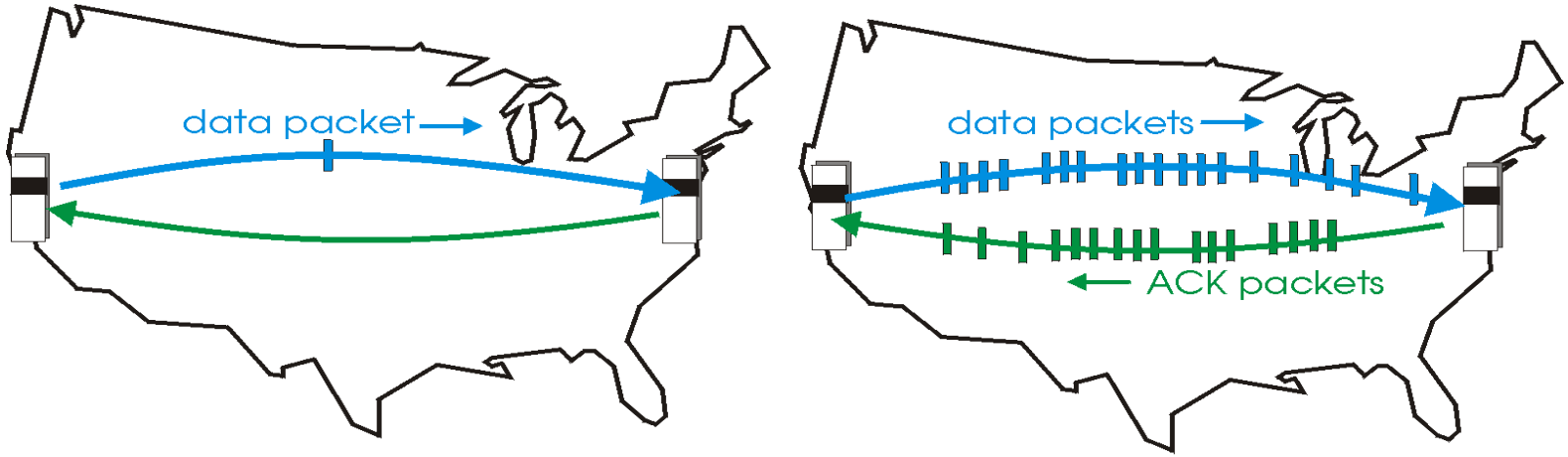
$$d_{trans} = \frac{L}{R} = \frac{8000\text{bits}}{10^9\text{bps}} = 8\,\text{microseconds}$$

- U_{sender}: utilization (= fraction of time sender is busy sending)

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

  - *if RTT=30 msec, 1KB pkt every 30 msec ➔ 33KB/sec throughput over 1 Gbps link*
  - network protocol limits use of physical resources!

Pipelined protocols



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

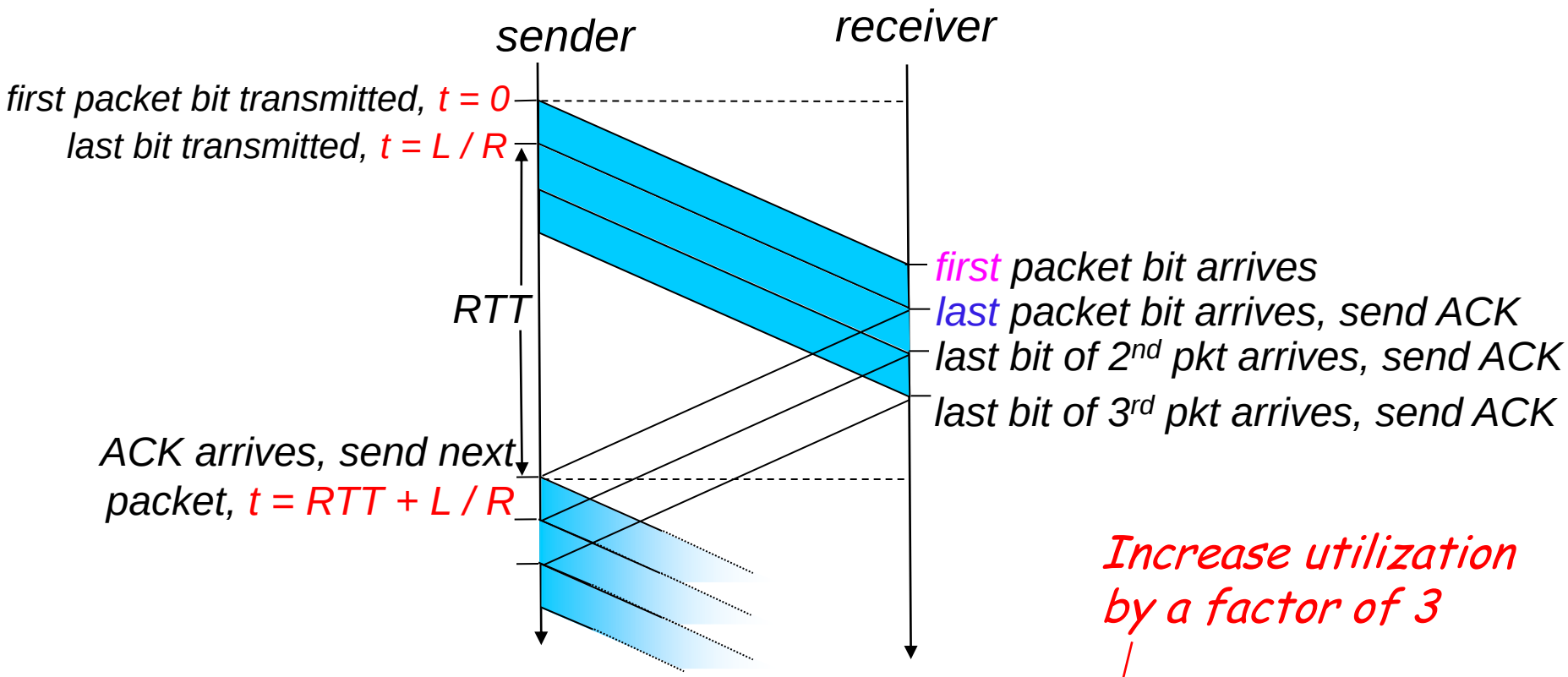
pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged pkts

- range of sequence numbers must be increased (sequence # {0, 1} are not enough.)

- two generic forms of pipelined protocols:

  go-Back-N and selective repeat

Pipelining: increases utilization



Increase utilization by a factor of 3

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

Pipelined Protocols (Sliding Window Protocols)

Go-back-N: big picture:

- sender can have up to N unACKed pkts in pipeline
- rcvr sends individual ack for each packet
- Sender interprets acks as cumulative acks

(Cumulative ACK: If ACK for pkt #5 is received, all pkts including and up to #5 have been correctly received)

- sender has a timer for oldest unACKed packet
  - if timer expires, retransmit all unACKed pkts

Selective Repeat: big pic

- sender can have up to N unACKed pkts in pipeline
- rcvr sends individual ack for each packet
- sender has one timer for each unACKed packet
  - when a timer expires, retransmit only one pkt corresponding to the timeout

Pipelined Protocols

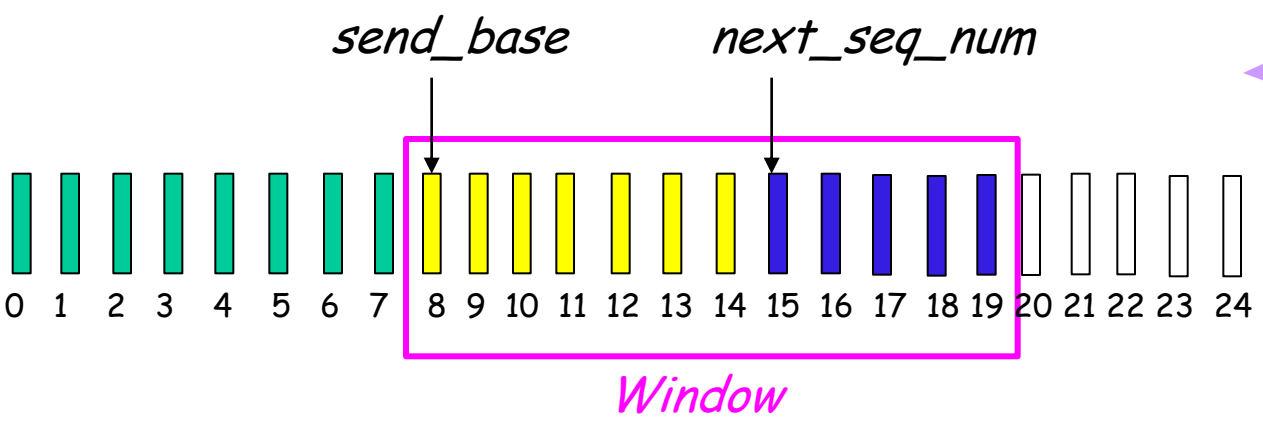
Remember:
The idea of cumulative ACK
is protocol specific.....

In TCP,
if the sender has received ACK
for **data byte #125**, it means
all bytes up to and including byte #124
have been received correctly and
the receiver is **expecting byte #125**.

Go-Back-N: Sender

Already ACKed | *Sent, but ACK not received* | Usable, not sent yet | Not usable

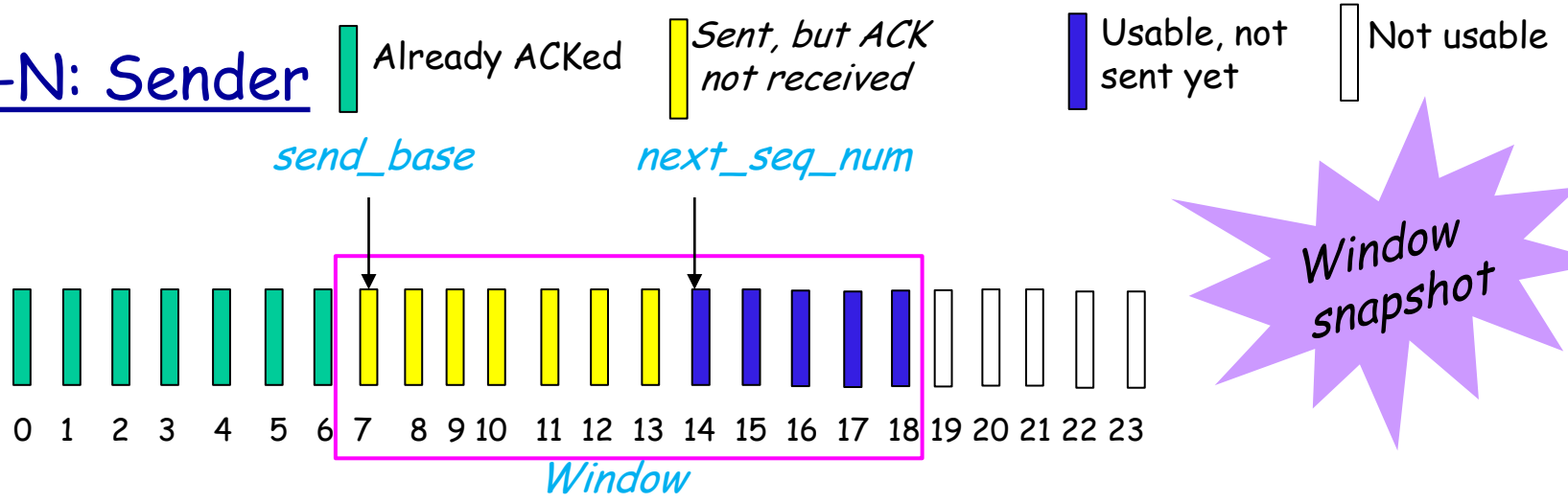
- k-bit seq # in pkt header: for k = 5, seq.# are: 0, 1, 2, 3, ... 31.
- "window" of up to N, consecutive unACKed pkts allowed

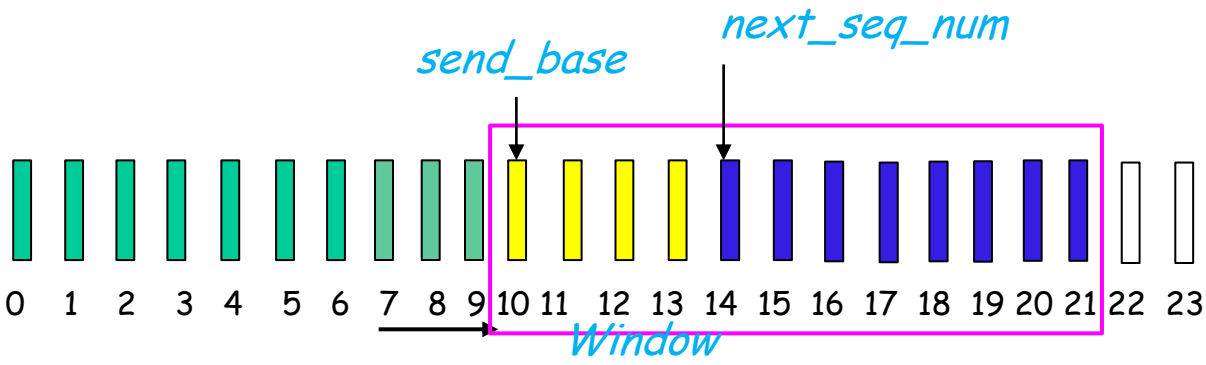


Window snapshot

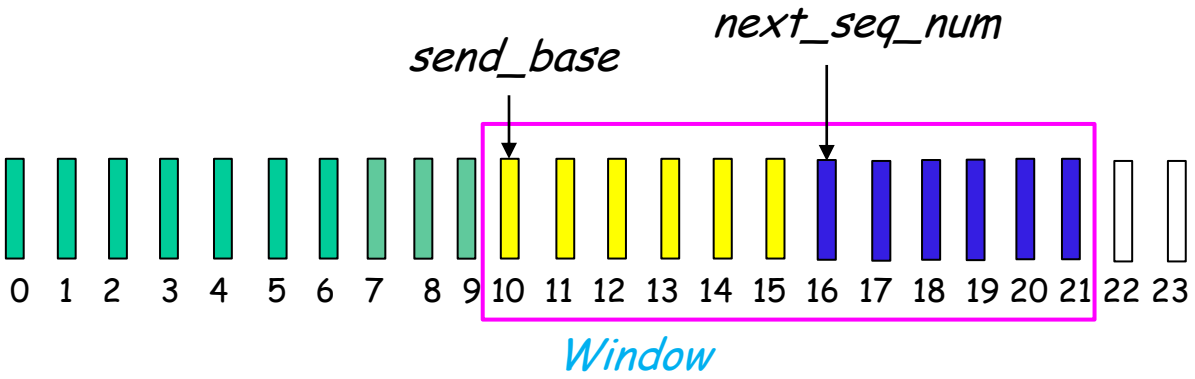
- timeout(n): retransmit pkt n and all higher seq # pkts in window

Go-Back-N: Sender



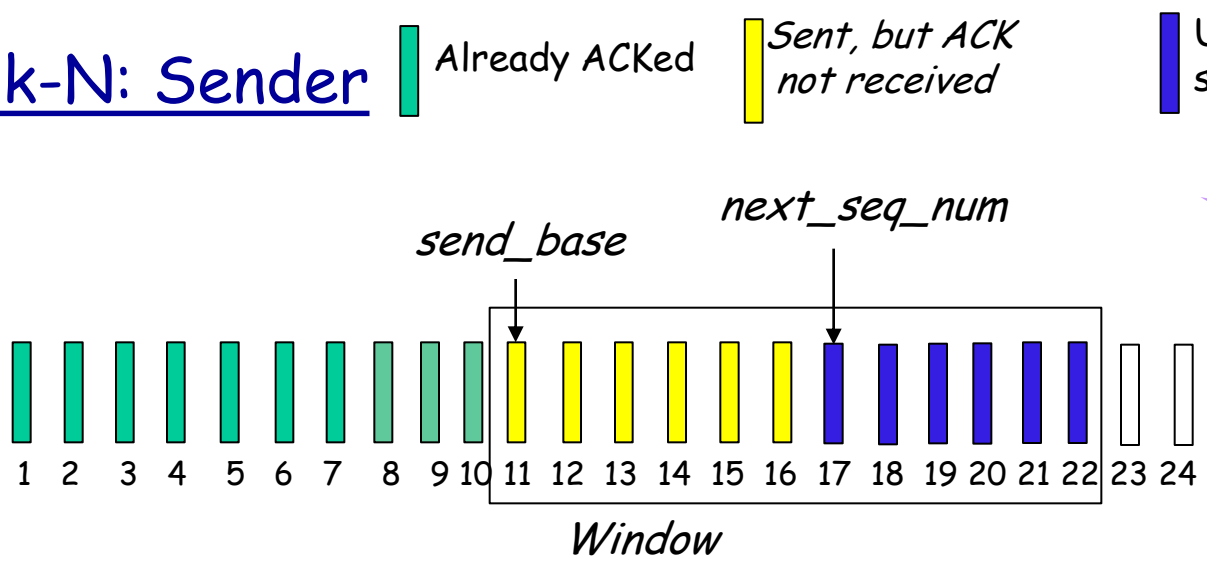


The sender receives ACKs for 3 pkts



The sender receives 2 data pkts from upper layer

Go-Back-N: Sender

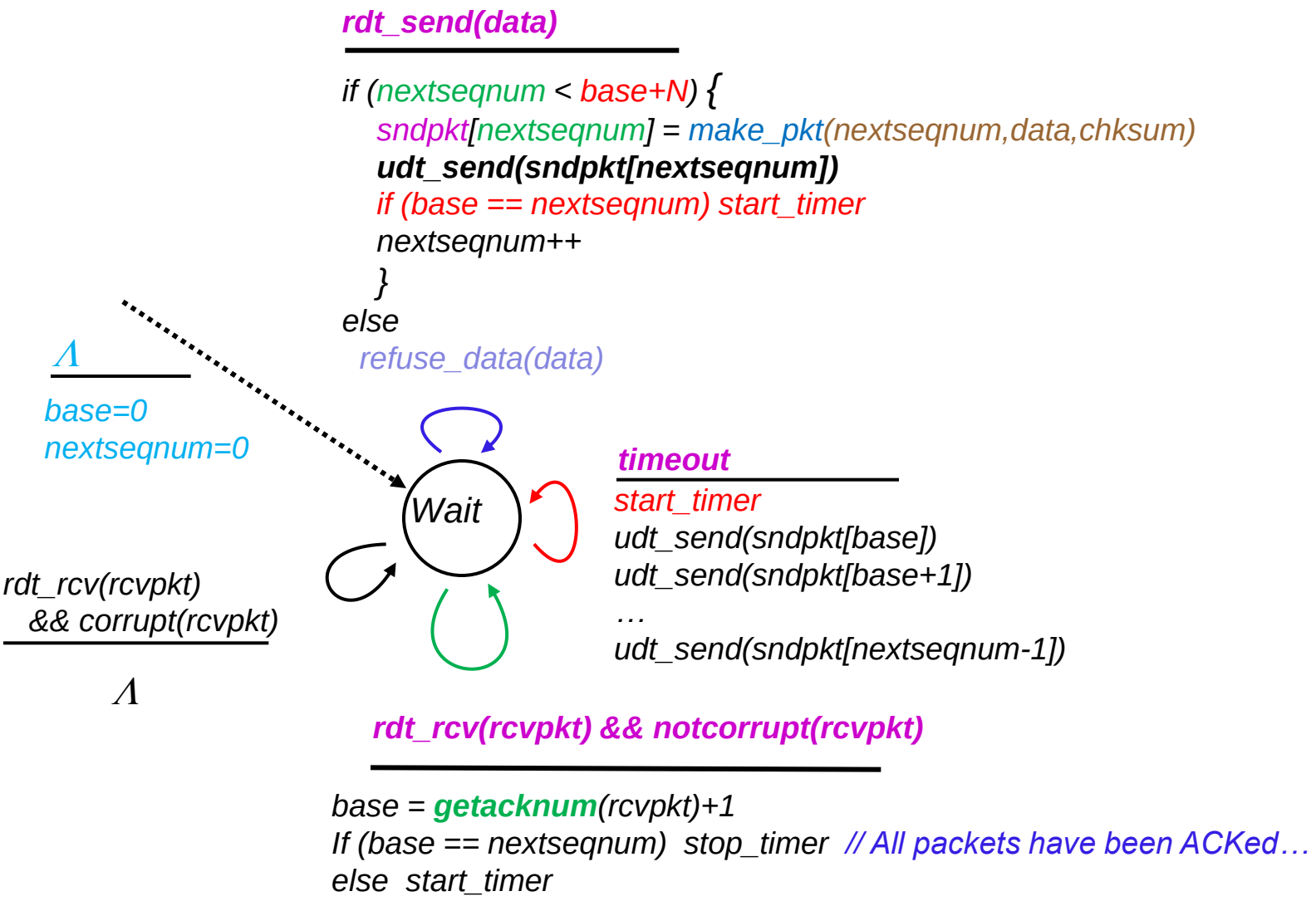


Window snapshot

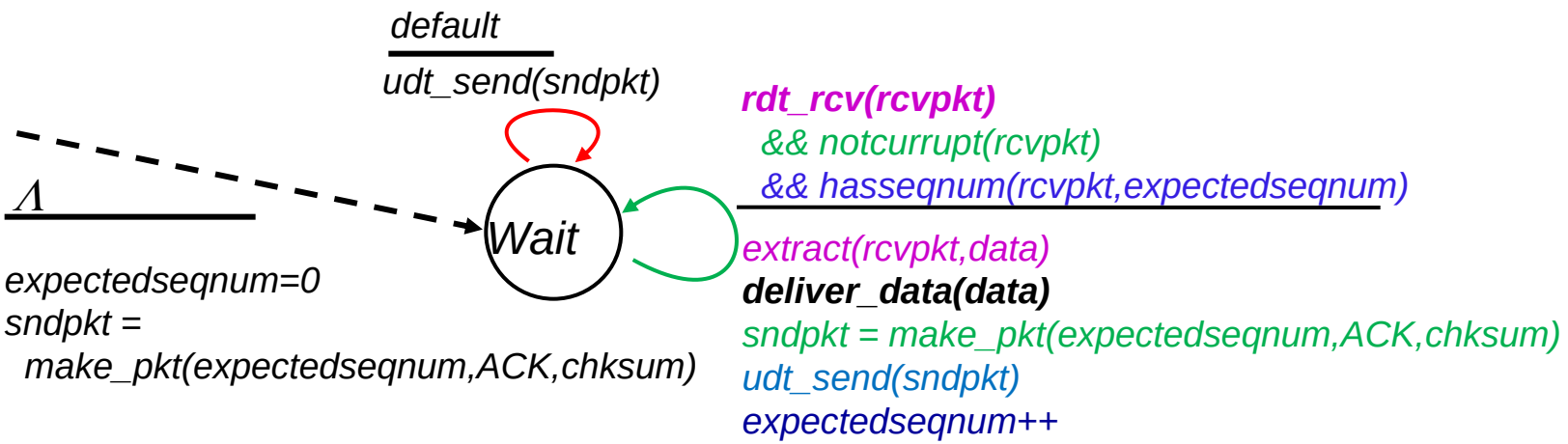
Timeout occurs

Retransmit

GBN: sender extended FSM



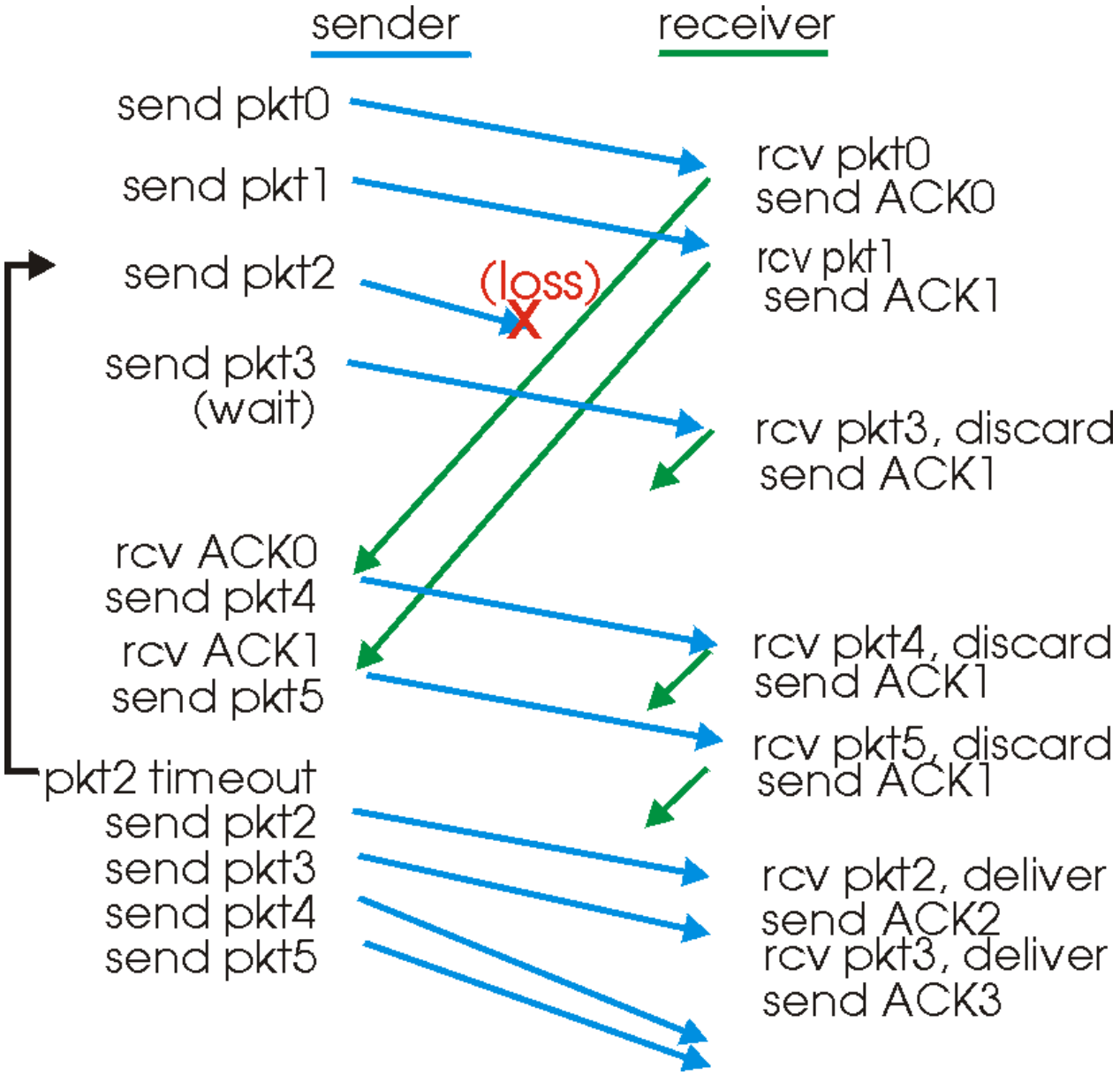
GBN: receiver extended FSM



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

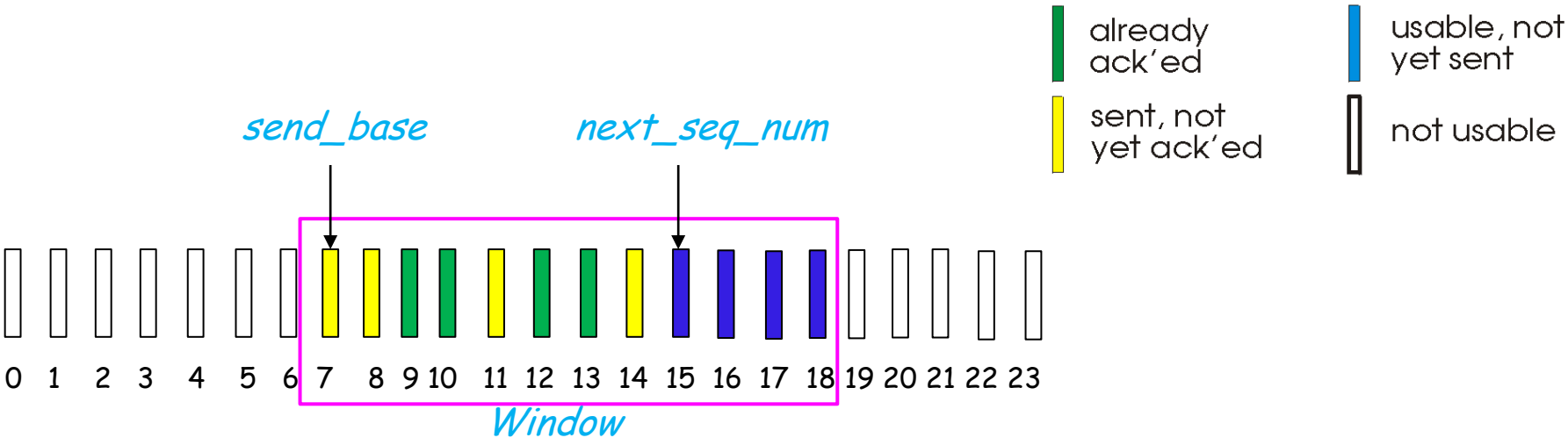


Selective Repeat

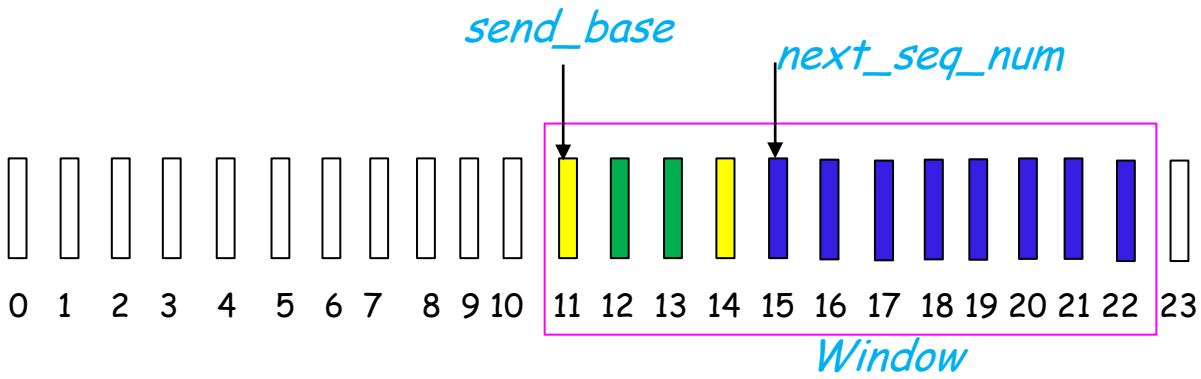
- receiver *individually* ACKs **all** correctly received pkts
  - buffers pkts, as needed,
    for eventual in-order delivery to upper layer

- *sender only resends pkts for which ACK is not received and timeout occurs*

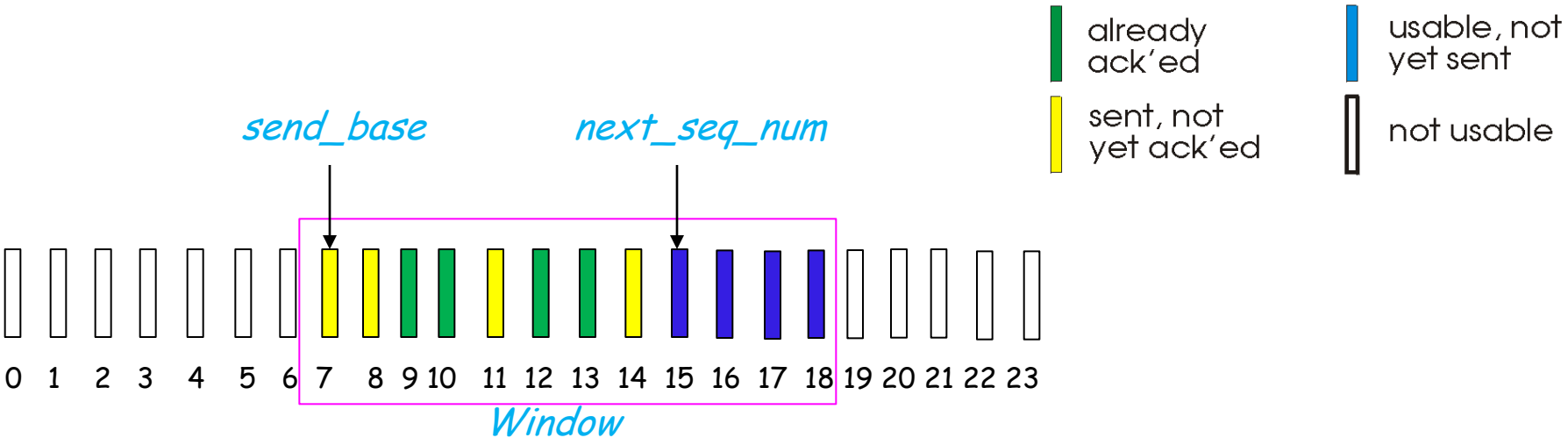
Selective repeat: sender window



ACK received for 7 and 8.



Selective repeat: sender window

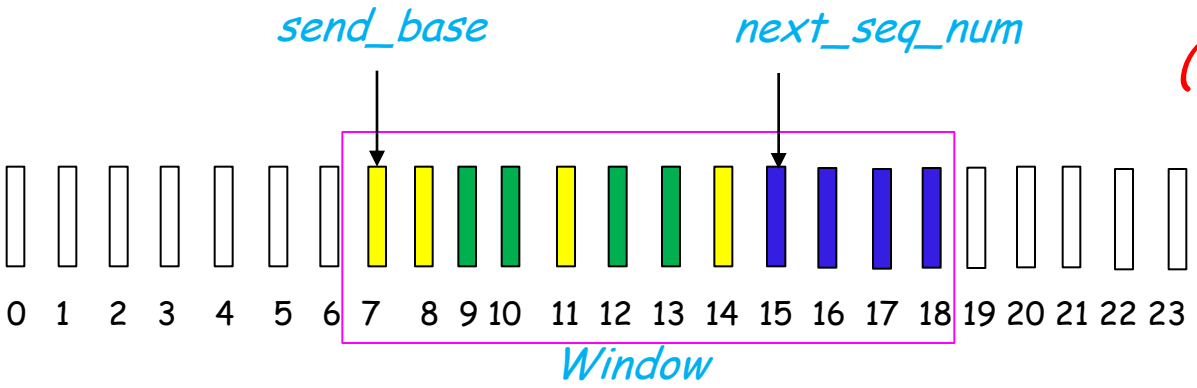


Retransmit

Timeout occurs for #7

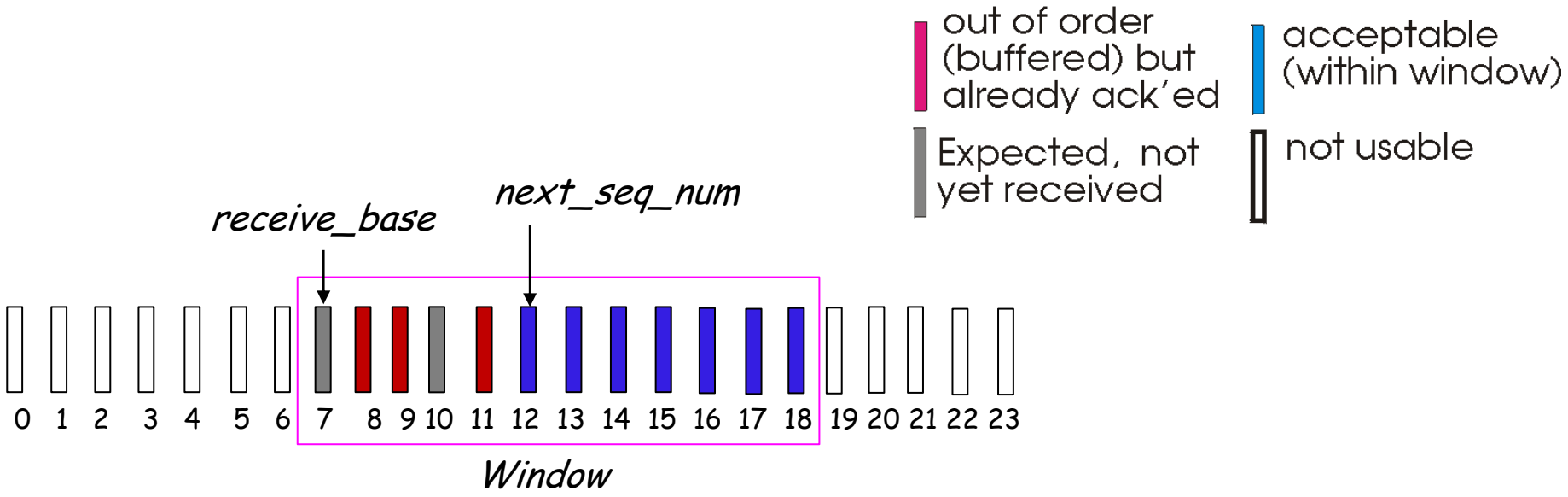
---

2 new packets arrive from upper layer

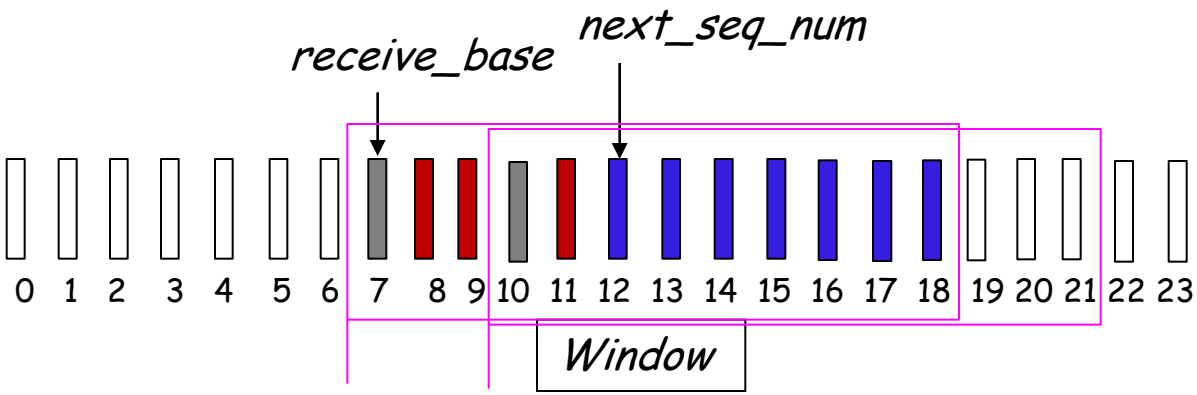


(Don't blink now.....)

Selective repeat: receiver window



Pkt #7 is received



Selective repeat

sender

data from above :

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

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)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

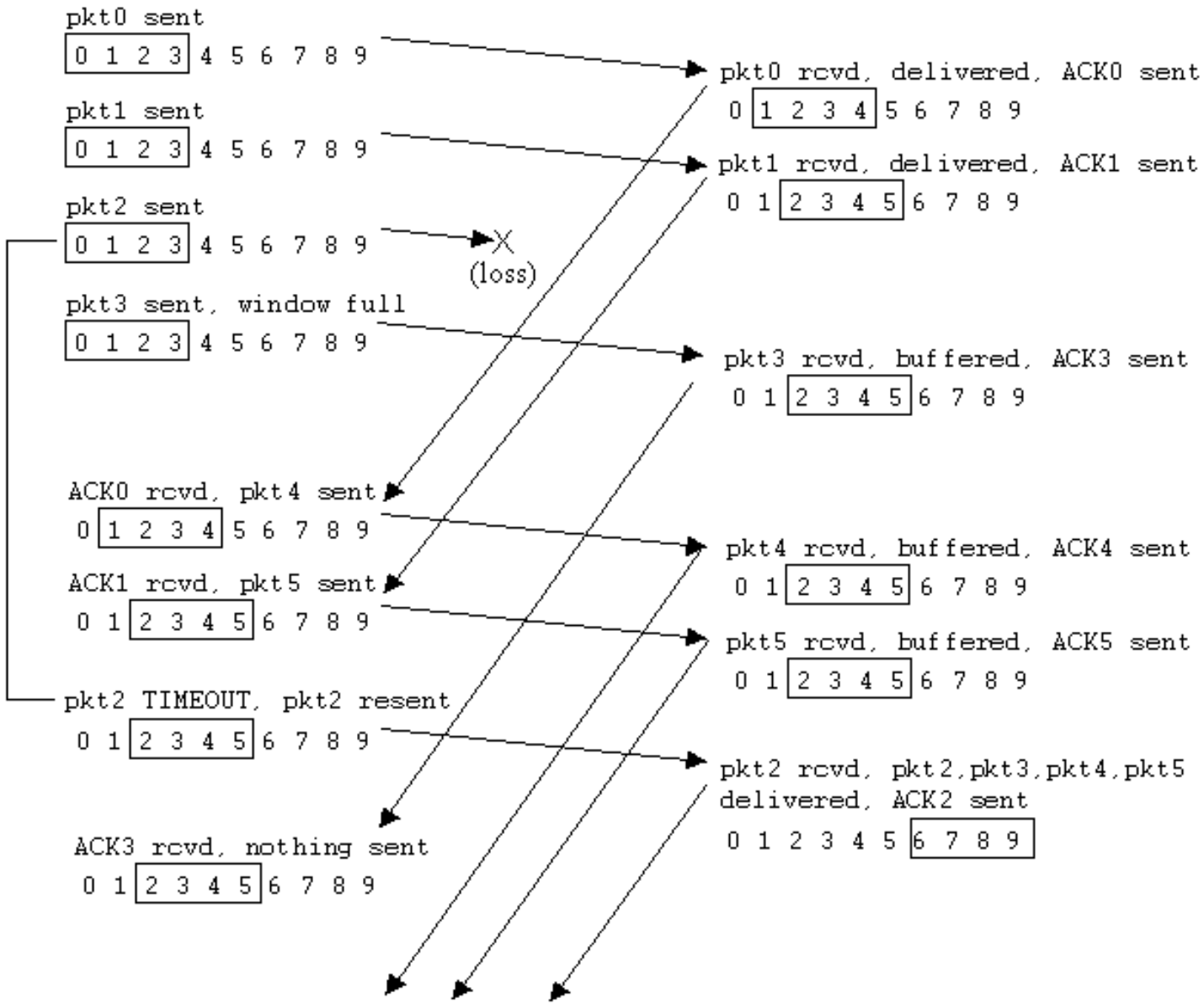
pkt n in [rcvbase-N,rcvbase-1]

- *ACK(n)*

otherwise:

- *ignore*

Selective repeat in action



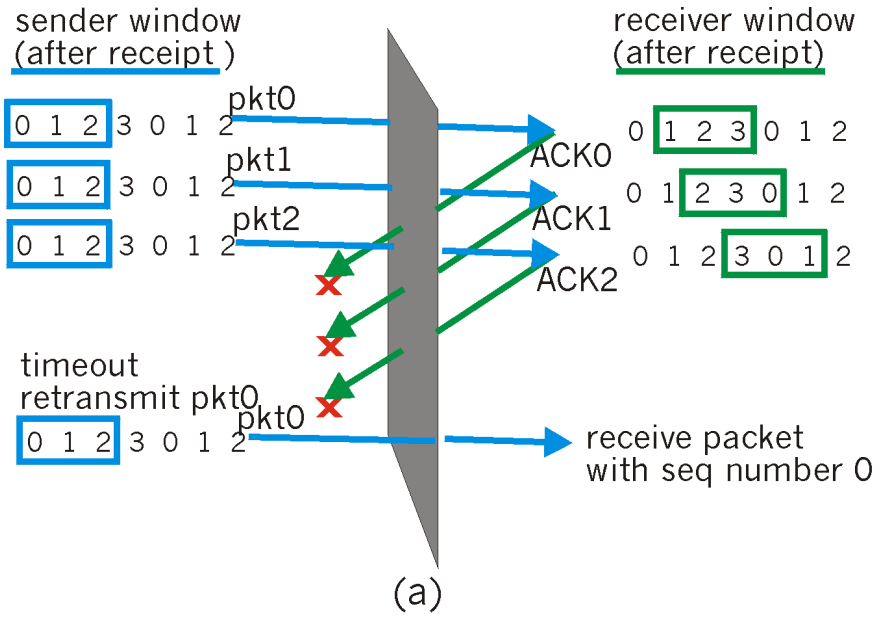
Selective repeat: dilemma

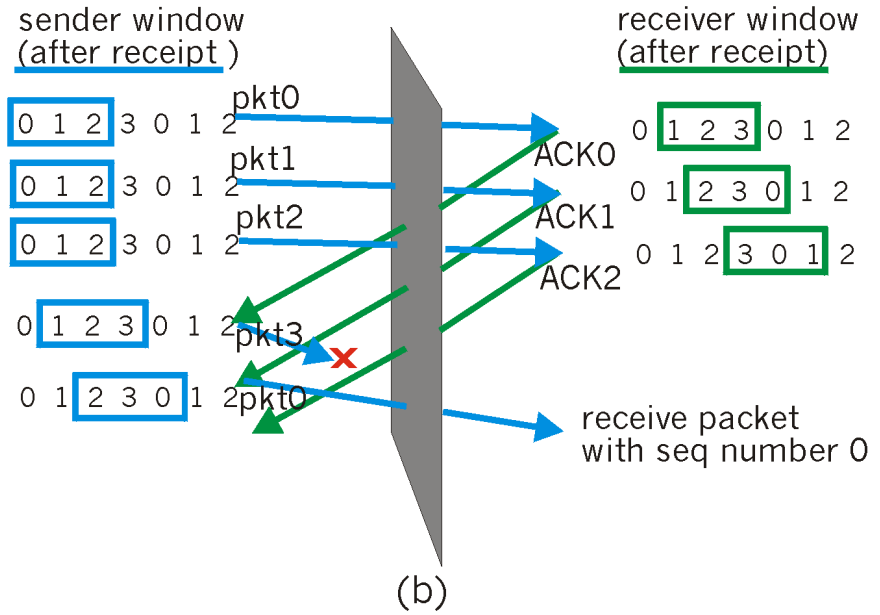
Example:

- seq #'s: 0, 1, 2, 3
- window size = 3

incorrectly passes duplicate data as new in (a)

receiver sees no difference in two scenarios!





End of Link Layer