Chapter 2 Link Layer and LANs

Our goals:

- understand principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control: *later!*
- instantiation and implementation of various link layer technologies

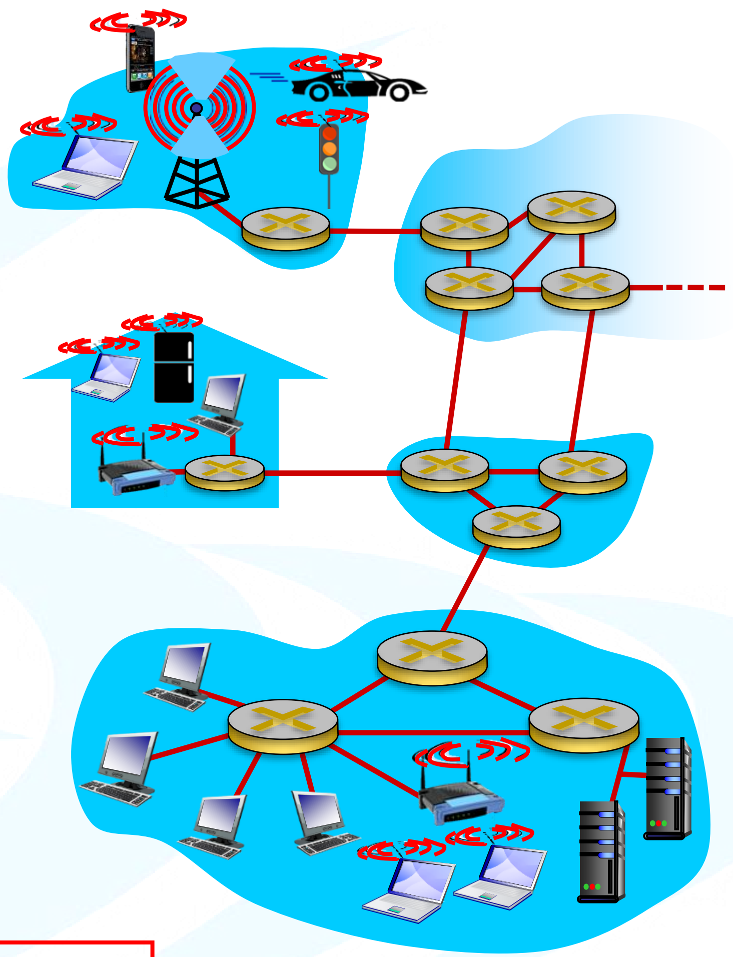
Chapter 2: Roadmap

- 2.1 Introduction and services of Link Layer
- 2.2 Error detection and correction
- 2.3 Multiple access Protocols
  - Channel Partitioning
  - Random Access
  - Taking Turns
- 2.4 Addressing
- 2.5 Ethernet
- 2.6 Interconnections & VLAN
- 2.7 PPP

- 2.1 Introduction and services of Link Layer

Some terminology:

- hosts and routers are nodes
- communication channels that connect adjacent nodes along communication path are links
  - wired links
  - wireless links
  - LANs
- layer-2 packet is a **frame**, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to adjacent node over a link

- Datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- Each link protocol provides different services
  - e.g., may or may not provide RDT over link

<u>transportation analogy</u>

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

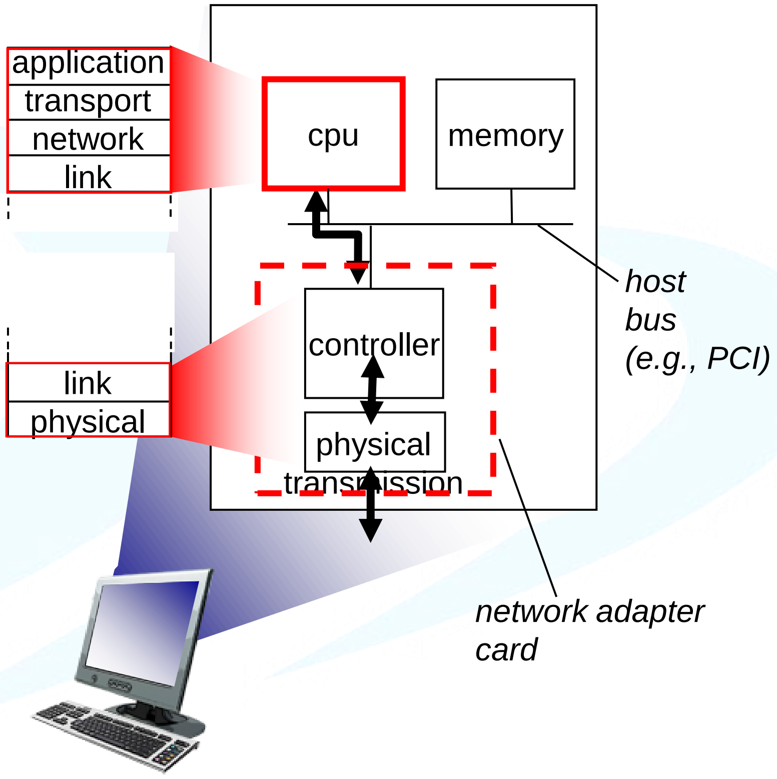
Link Layer Services

- Framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- Reliable delivery between adjacent nodes
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit error link (fiber, some twisted pair)
  - wireless links: high error rates
    - Q: why both link-level and end-end reliability?

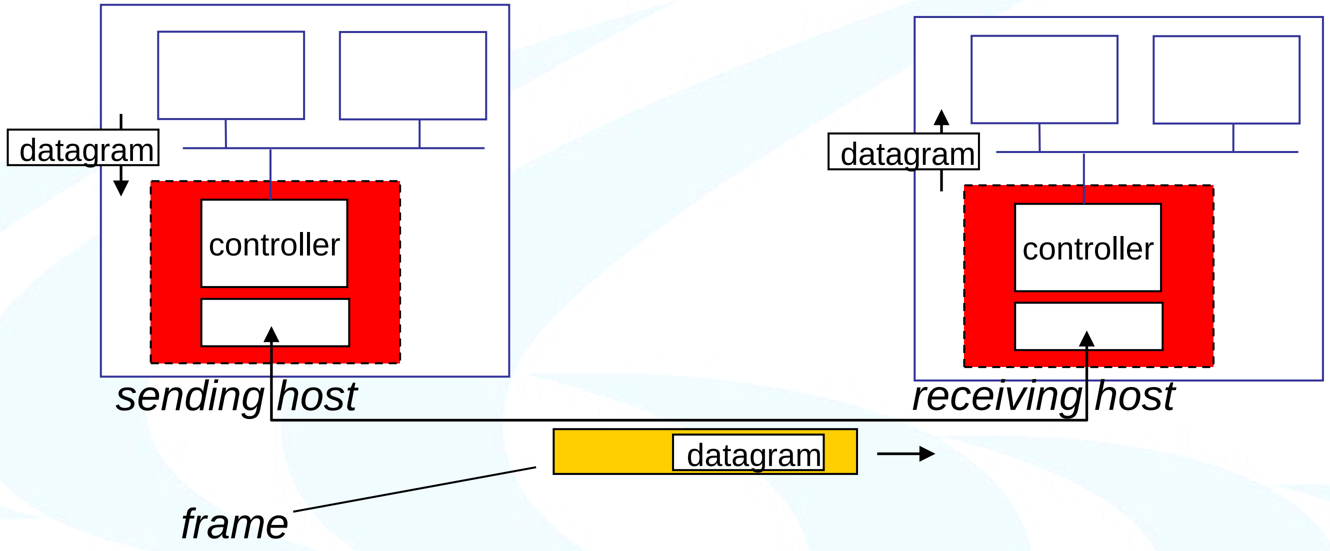
- ***Flow Control:***
  - pacing between adjacent sending and receiving nodes
- ***Error Detection:***
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- **Error Correction:**
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- ***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?

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



Adaptors communicating

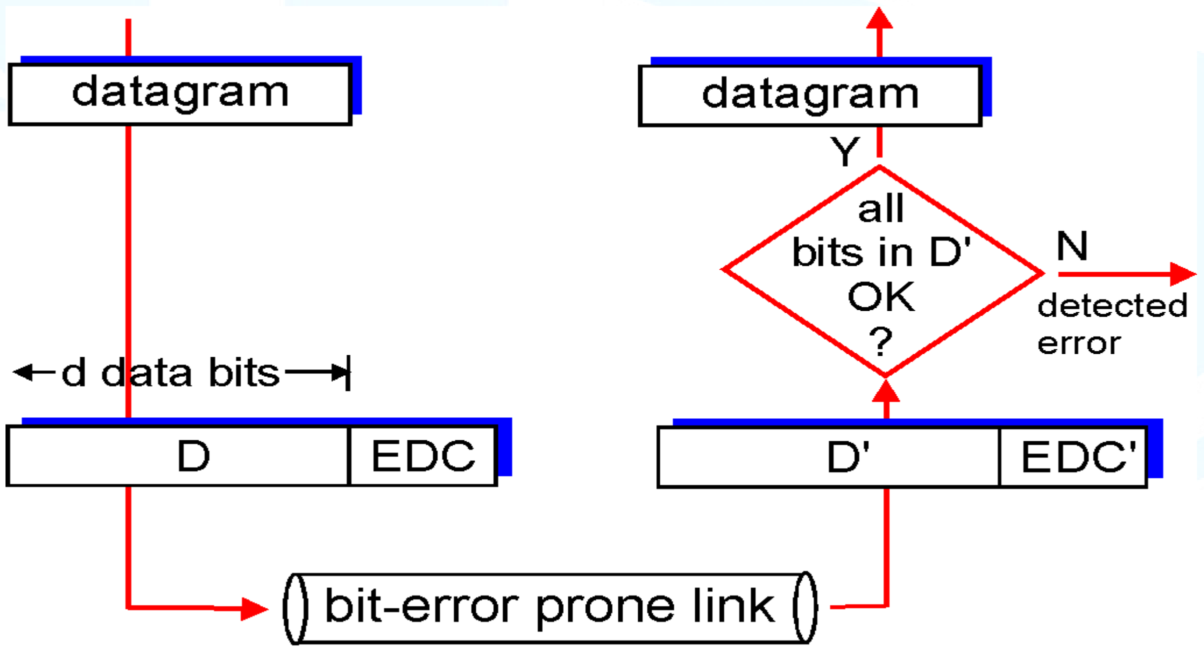


- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagram, passes to upper layer at receiving side

2.2 Error detection and correction

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

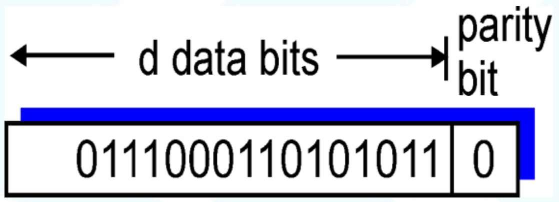
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



1.Parity Checking

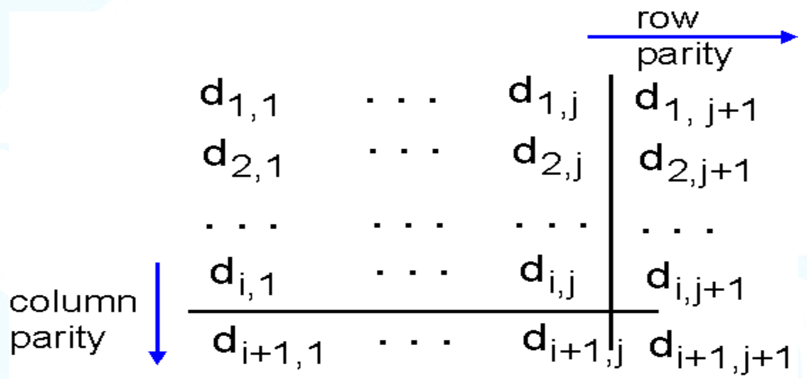
Single Bit Parity:

Detect single bit errors

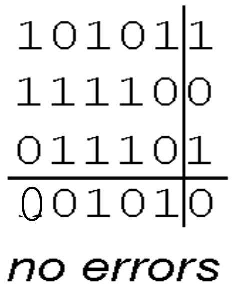


Two Dimensional Bit Parity:

Detect *and correct* single bit errors

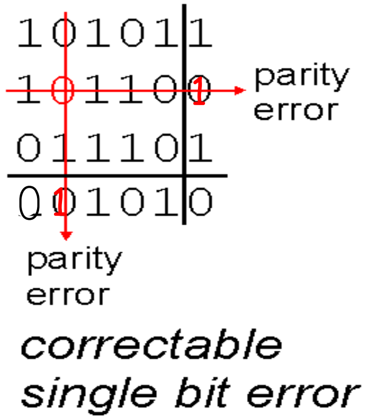


$$\begin{array}{cccc|c} d_{1,1} & \cdots & d_{1,j} & & d_{1,j+1} \\ d_{2,1} & \cdots & d_{2,j} & & d_{2,j+1} \\ \cdots & \cdots & \cdots & & \cdots \\ d_{i,1} & \cdots & d_{i,j} & & d_{i,j+1} \\ \hline d_{i+1,1} & \cdots & d_{i+1,j} & & d_{i+1,j+1} \end{array}$$



$$\begin{array}{ccccc|c} 1&0&1&0&1&1 \\ 1&1&1&1&0&0 \\ 0&1&1&1&0&1 \\ \hline 0&0&1&0&1&0 \end{array}$$

no errors



$$\begin{array}{ccccc|c} 1&0&1&0&1&1 \\ 1&0&1&1&0&0 \\ 0&1&1&1&0&1 \\ \hline 0&0&1&0&1&0 \end{array}$$

correctable single bit error

2.Internet checksum

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

Sender:

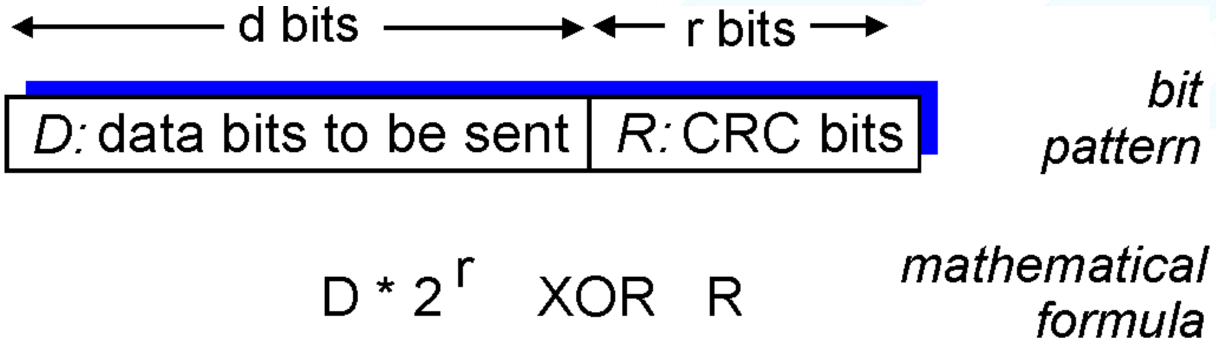
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

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? More* later ….

3.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)
  - 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 (ATM, HDCL)



$$D * 2^r \quad \text{XOR} \quad R$$

CRC Example

- **Want:**
  - **D.2^r XOR R = nG**
- ***equivalently:***
  - **D.2^r = nG XOR R**
- ***equivalently:***
- **if we divide D.2^r by G, want remainder R to satisfy:**

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

```
                     1101010110 ← Q 商
除数 G → 110101 ) 101000110100000 ← 2^rD 被除数
                  110101
                   111011
                   110101
                     111010
                     110101
                       111110
                       110101
                         101100
                         110101
                          110010
                          110101
                            01110 ← R 余数
```

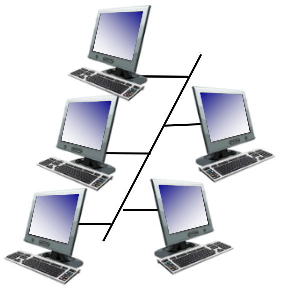
- 国际标准已经定义了**8-**、**16-**、**32-**位生成多项式**G**；**8-**位**CRC**用于**ATM**头部**5**字节的保护；**32-CRC**用于大量链路层**IEEE**协议。
- 每个**CRC**标准能够检测少于**r+1**位的猝发错误和任意的奇数个比特错误……
- 其他检错和纠错方法不常用，故不作专门介绍
- 校验和通常应用于传输层，要求简单快速的软件实现方式，而**CRC**通常应用于链路层，可以适配器硬件实现复杂的算法。

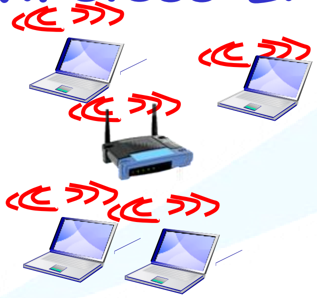
2.3 Multiple access Protocols

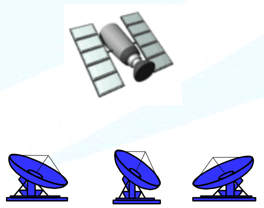
Multiple Access Links and Protocols

Two types of "links":

- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch and host
- broadcast (shared wire or medium)
  - traditional 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 Mulitple 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

MAC Protocols: a taxonomy

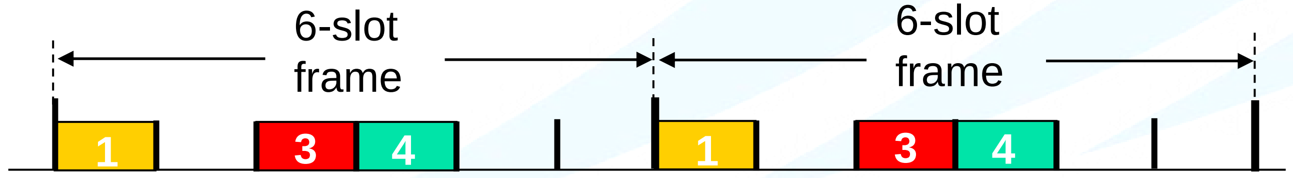
Three broad classes:

- 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

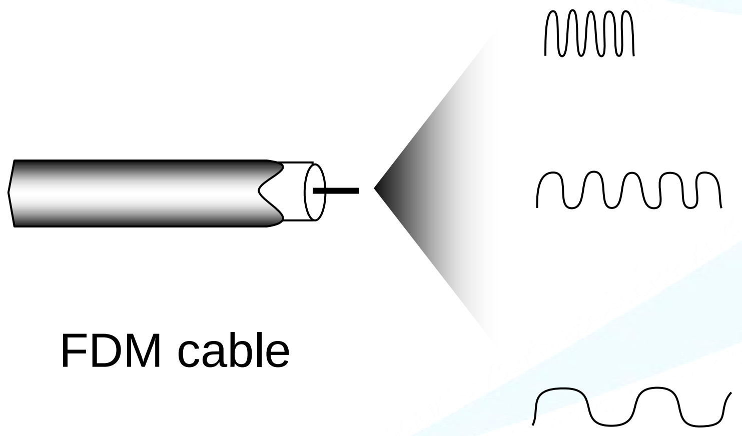
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle

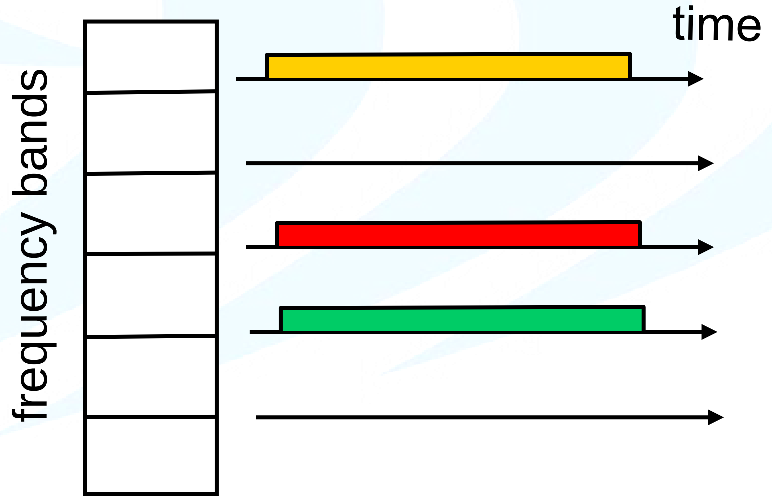


Channel Partitioning MAC protocols: FDMA

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





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 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

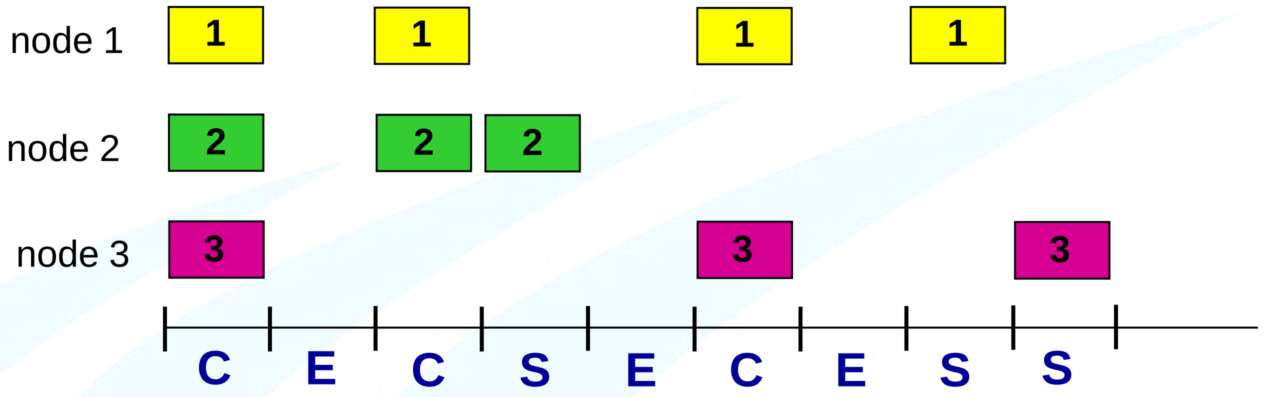
Slotted ALOHA

Assumptions

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

Operation

- when node obtains fresh frame, it 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 prob. p until success



Pros

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in 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, each with many frames to send)

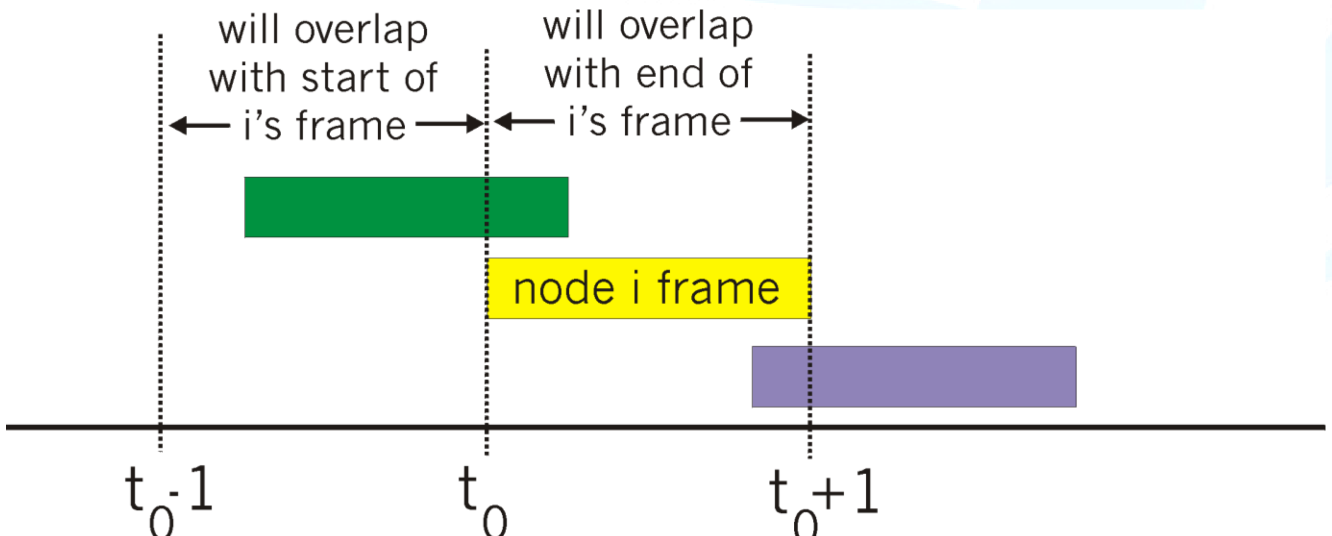
- Suppose N nodes with many frames to send, each transmits in slot with probability *p*
- prob that node 1 has success in a slot = $p(1-p)^{N-1}$
- prob that any node has a success = $Np(1-p)^{N-1}$

- For max efficiency with N nodes, 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 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
- collision probability increases:
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$


will overlap with start of i's frame
will overlap with end of i's frame
node i frame
t_0-1
t_0
t_0+1

Pure Aloha efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in [p_0-1,p_0] ·

P(no other node transmits in [p_0-1,p_0]

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

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

… choosing optimum p and then letting n -> ∞

= 1/(2e) = .18

Even worse !

CSMA (Carrier Sense Multiple Access)

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

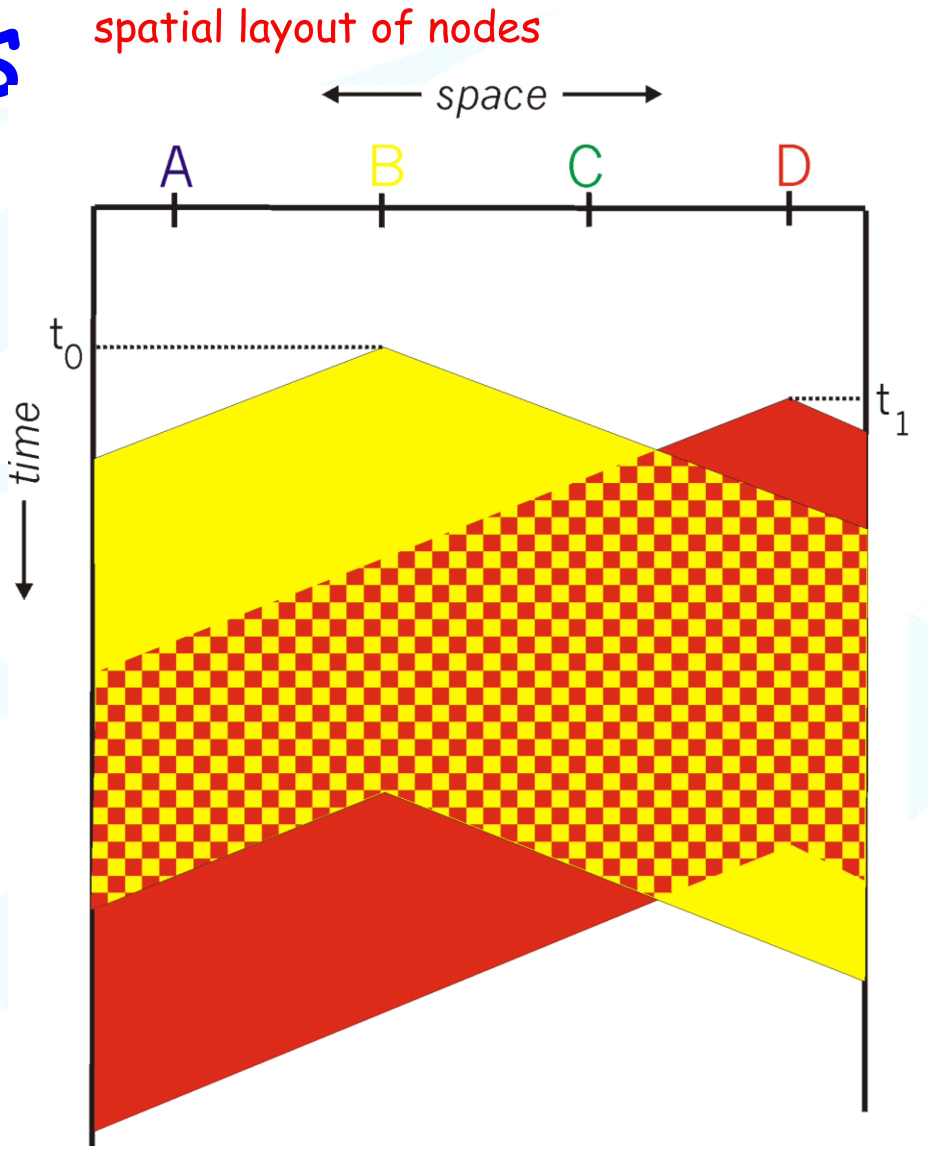
- Human analogy: don't interrupt others!

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



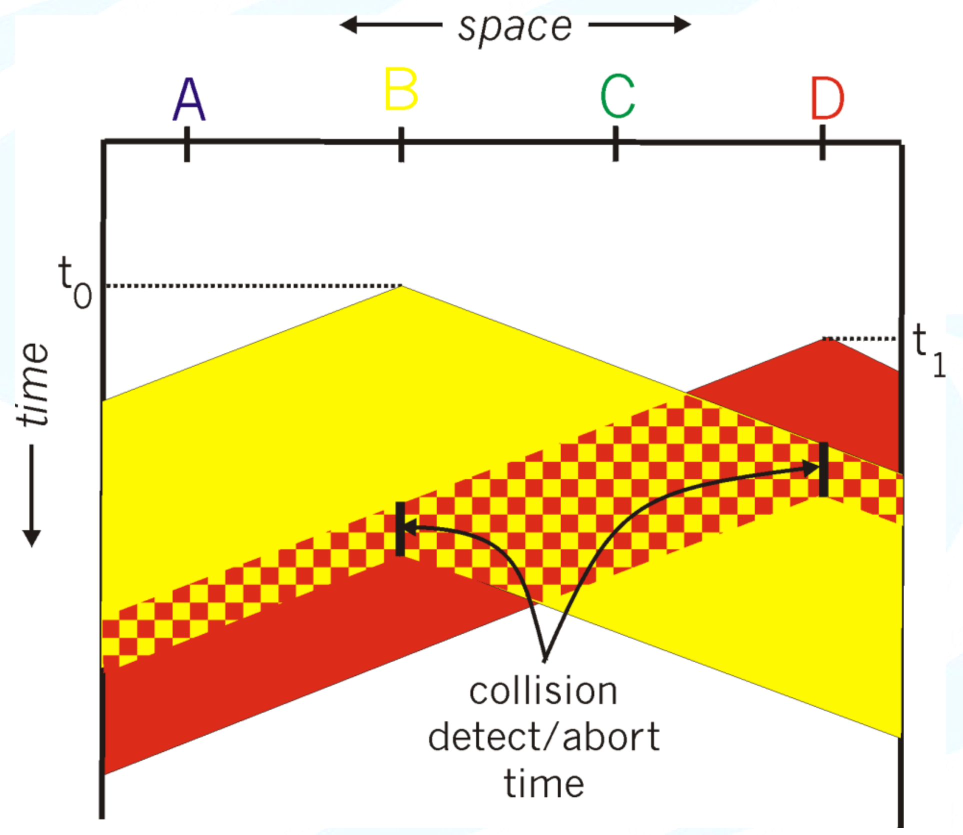
CSMA/CD (Collision Detection)

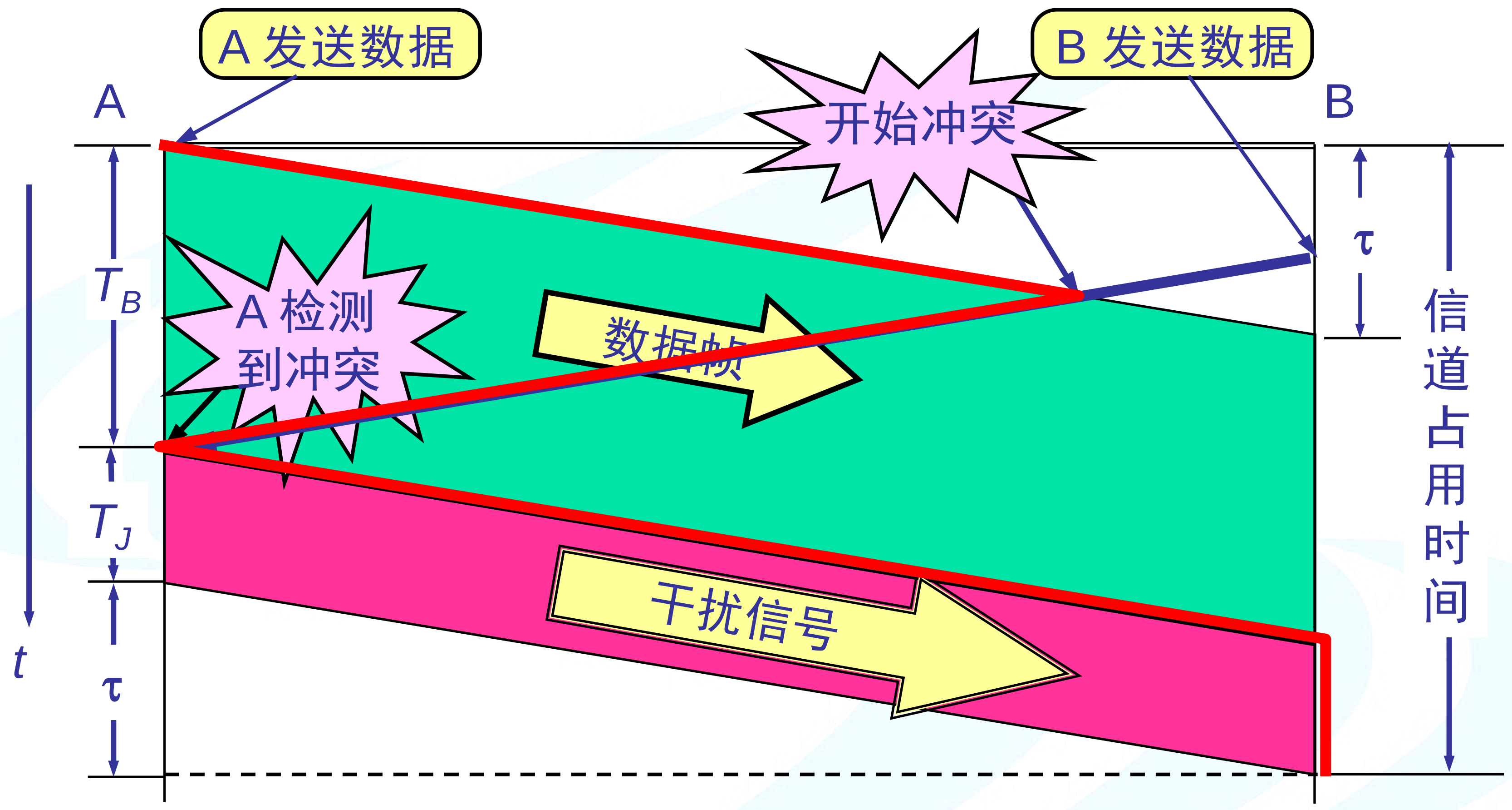
CSMA/CD: carrier sensing, deferral as in CSMA

- 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: receiver shut off while transmitting
- human analogy: the polite conversationalist

CSMA/CD collision detection





B也能够检测到冲突，并立即停止发送数据帧，接着就发送干扰信号。这里为了简单起见，只画出A发送干扰信号的情况。

"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

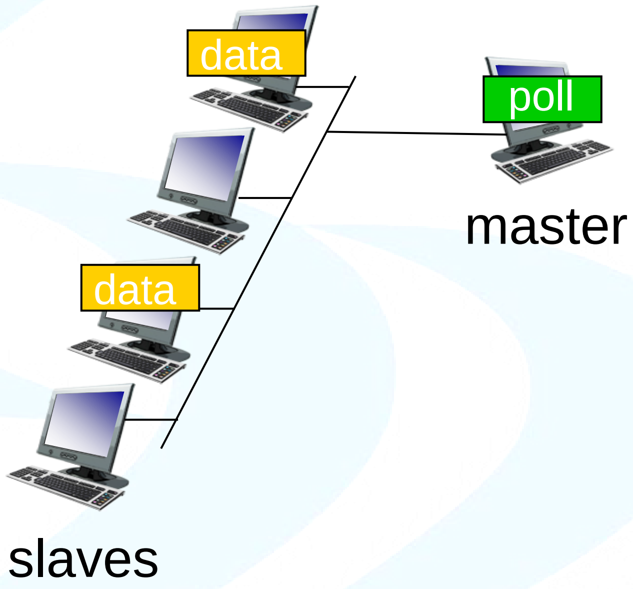
"taking turns" protocols

look for best of both worlds!

"Taking turns" MAC protocols

polling:

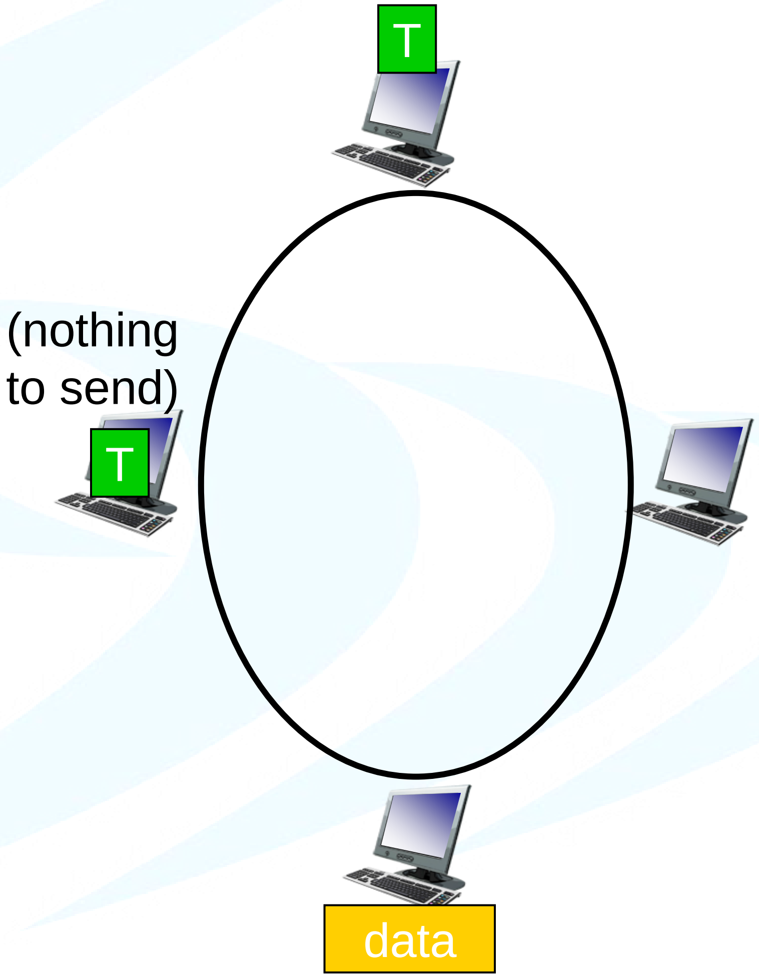
- **master node "invites" slave nodes to transmit in turn**
- **typically used with "dumb" slave devices**
- **concerns:**
  - **polling overhead**
  - **latency**
  - **single point of failure (master)**



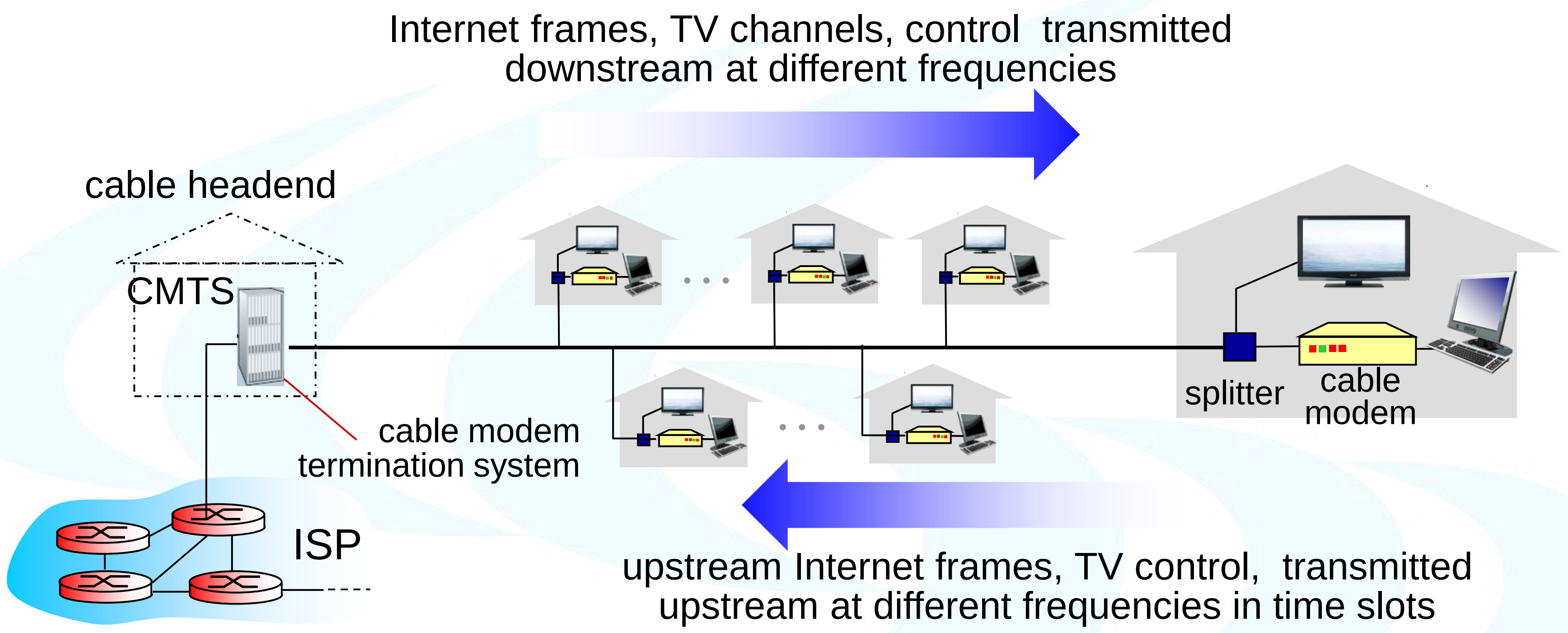
“Taking turns” MAC protocols

token passing:

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

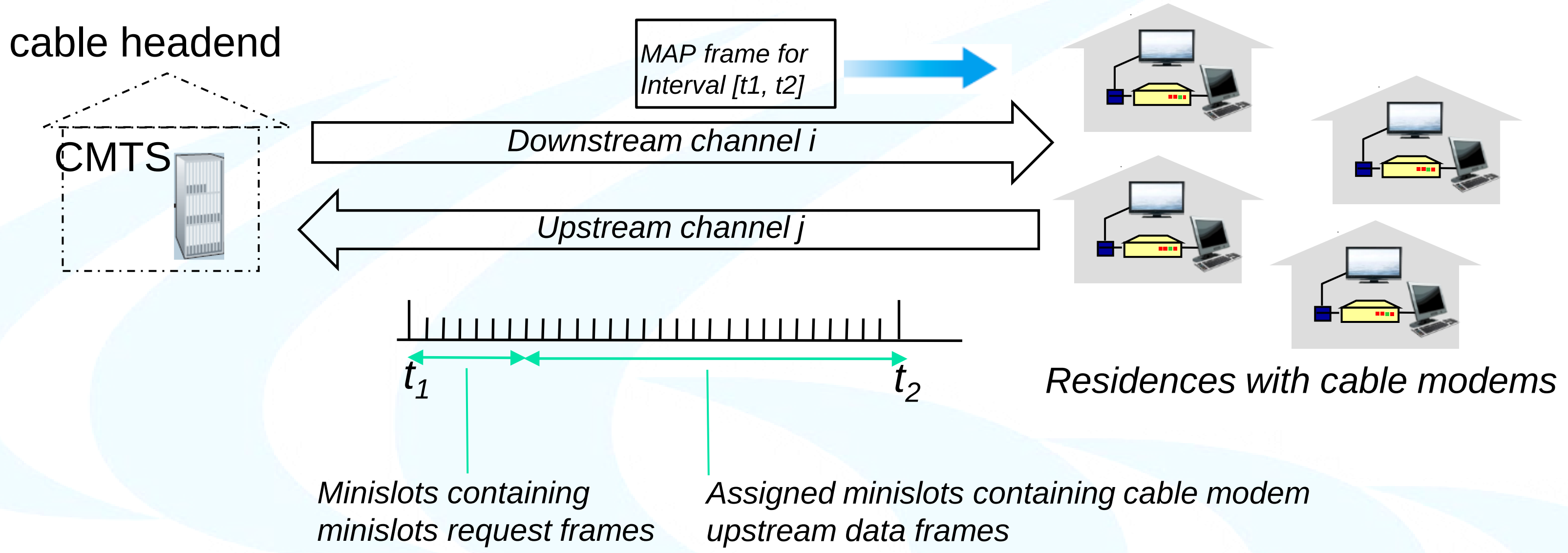


Cable access network


Internet frames, TV channels, control transmitted downstream at different frequencies
cable headend
CMTS
cable modem termination system
ISP
splitter
cable modem
upstream Internet frames, TV control, transmitted upstream at different frequencies in time slots

- multiple 40Mbps downstream (broadcast) channels
  - single CMTS transmits into channels
- multiple 30 Mbps upstream channels
  - multiple access: all users contend for certain upstream channel time slots (others assigned)

Cable access network


cable headend
CMTS
MAP frame for Interval [t1, t2]
Downstream channel i
Upstream channel j
t1
t2
Residences with cable modems
Minislots containing minislots request frames
Assigned minislots containing cable modem upstream data frames

DOCSIS: data over cable service interface spec

- FDM over upstream, downstream frequency channels
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

Summary of MAC protocols

- ***channel partitioning,*** **by time, frequency or code**
  - **Time Division, Frequency Division**
- ***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, token ring**

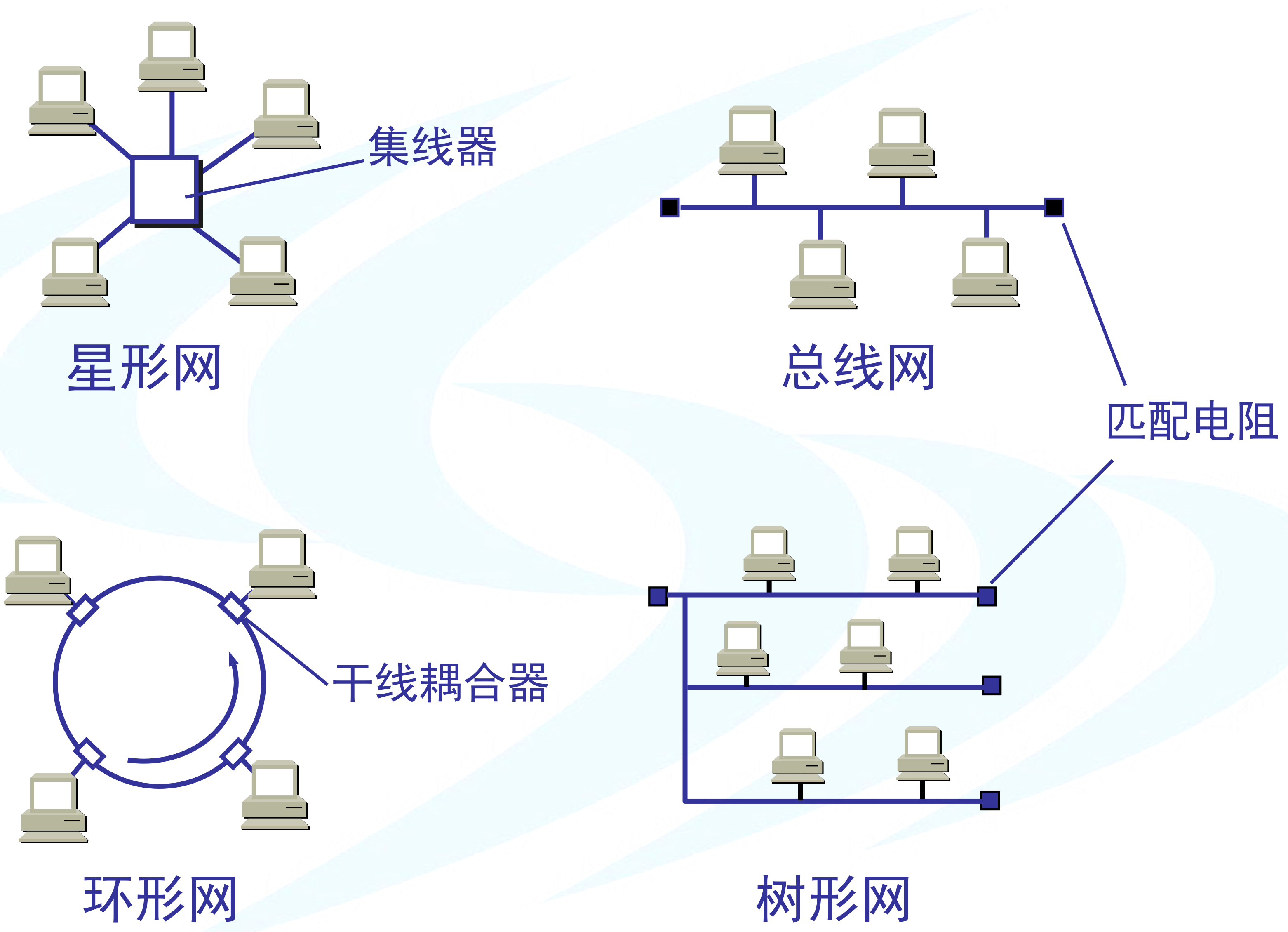
LAN technologies

Data link layer so far:

- services, error detection/correction, multiple access

Next: LAN technologies

- addressing
- Ethernet
- hubs, switches
- PPP


集线器
星形网
总线网
匹配电阻
干线耦合器
环形网
树形网

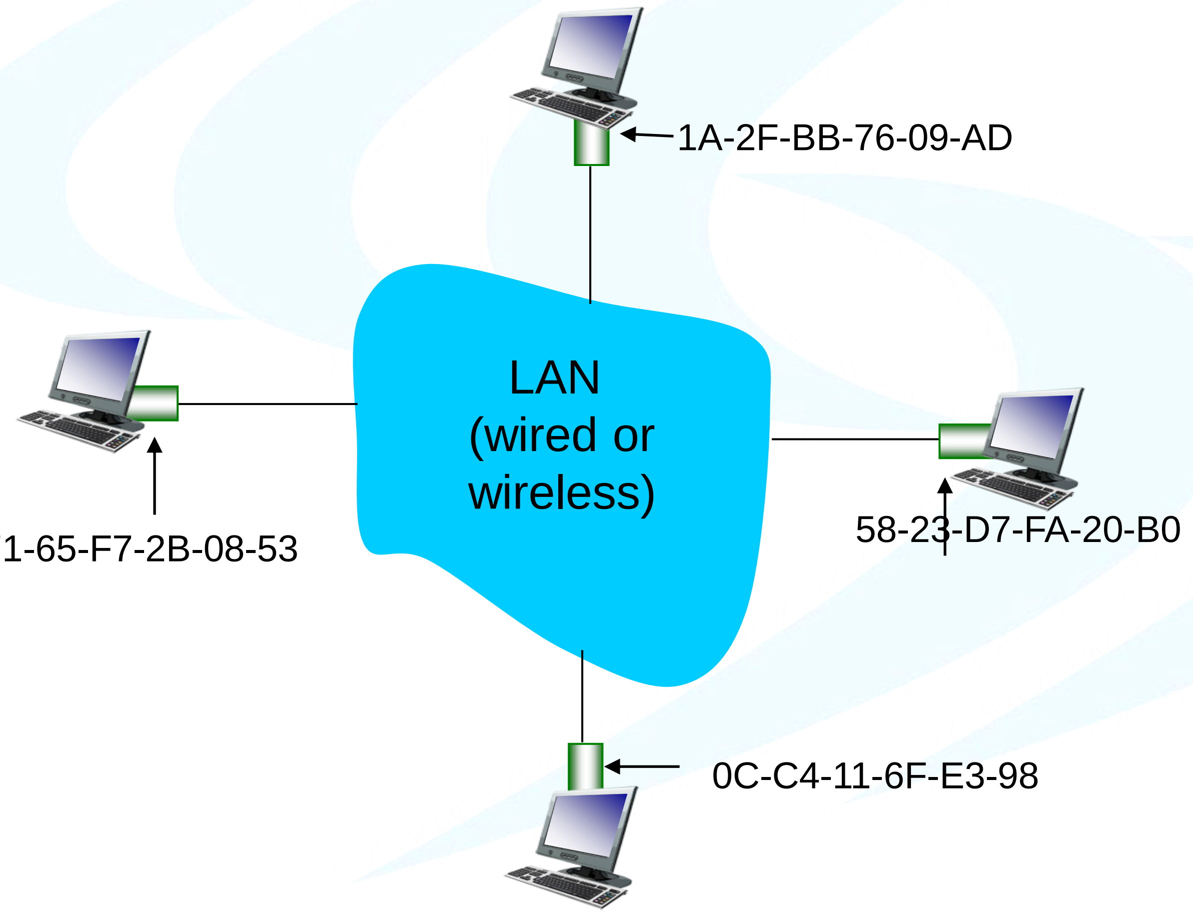
2.4 Addressing

MAC Addresses and ARP

- **32-bit IP address:**
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- **MAC (or LAN or physical or Ethernet) address:**
  - function: *used 'locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

LAN Addresses and ARP

Each adapter on LAN has unique LAN address



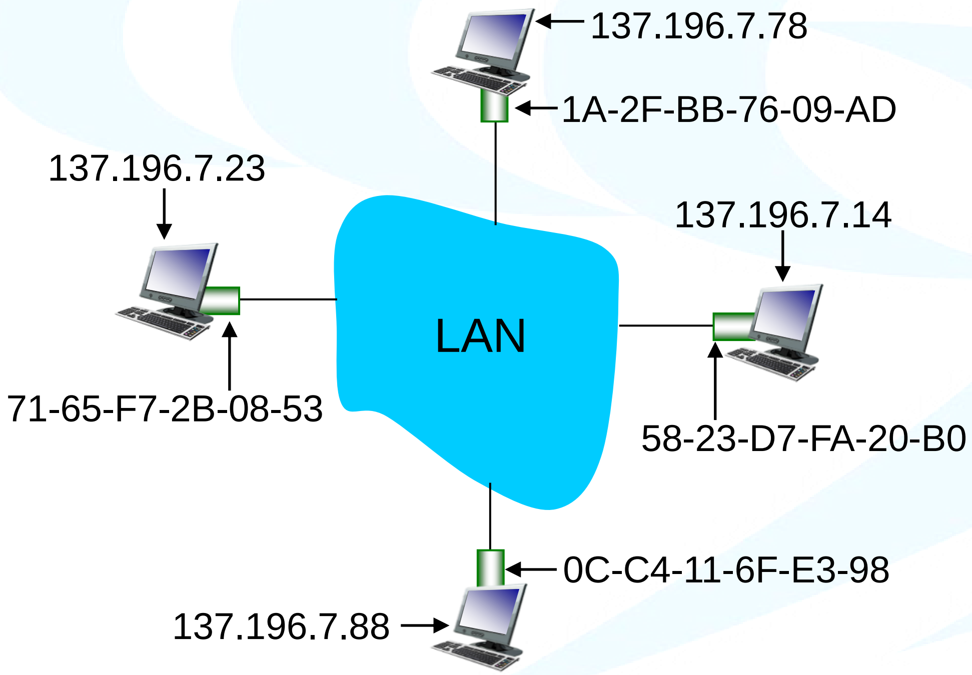
Broadcast address =
FF-FF-FF-FF-FF-FF

🟩 adapter

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

ARP: Address Resolution Protocol

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



- Each IP node (Host, Router) on LAN has **ARP** table
- ARP Table: IP/MAC address mappings for some LAN nodes

  < IP address; MAC address; TTL>

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

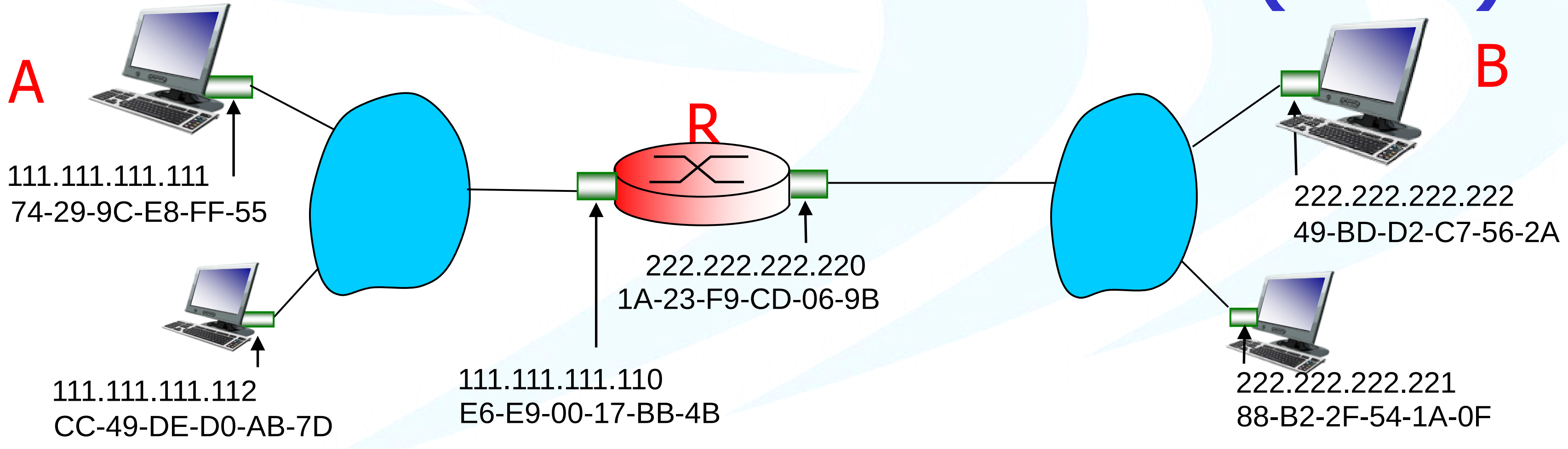
ARP protocol: Same LAN (network)

- A wants to send datagram to B, and B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - Dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables without intervention from net administrator

Addressing: routing to another LAN

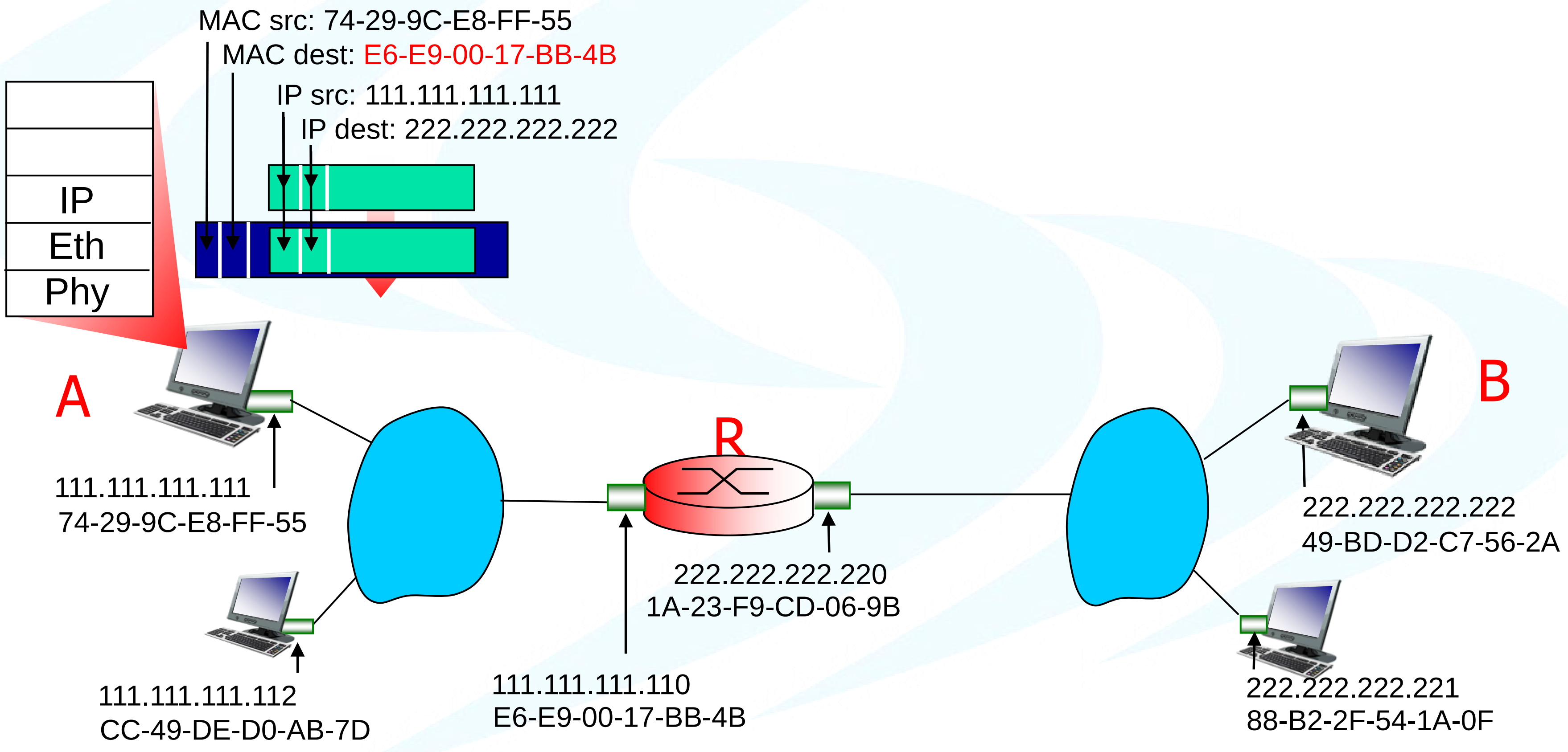
walkthrough: **send datagram from A to B via R**

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B’ s IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R’ s MAC address (how?)



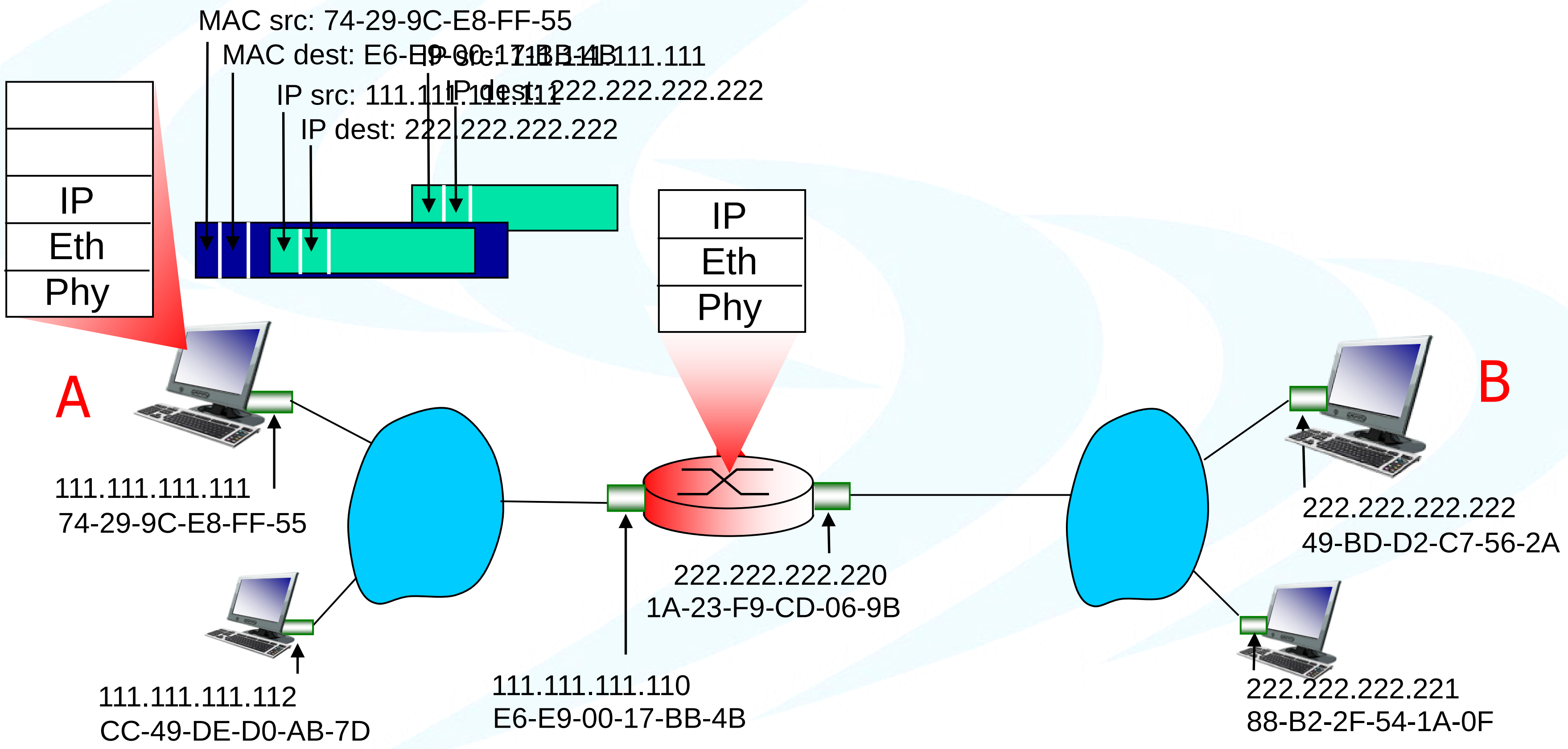
Addressing: routing to another LAN

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



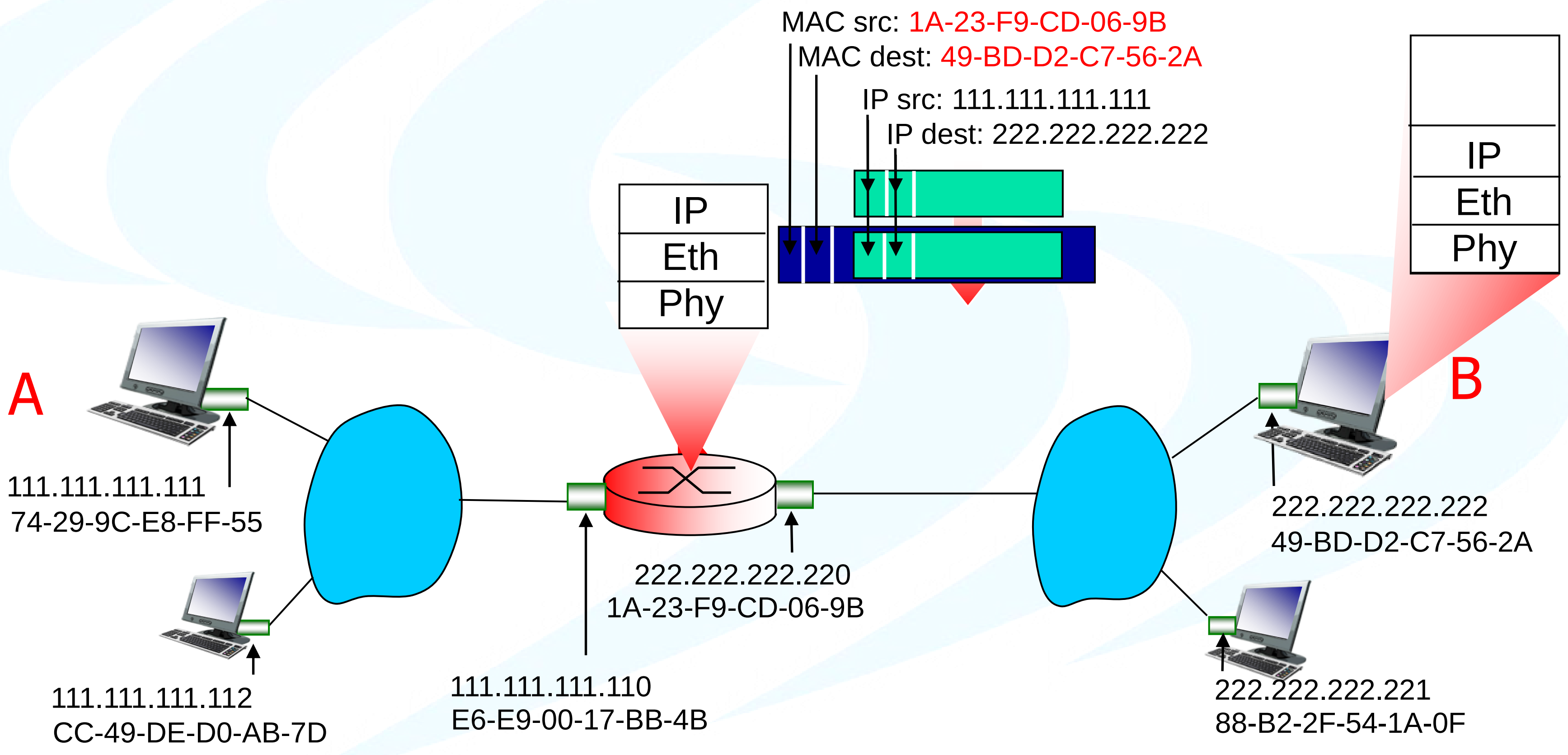
Addressing: routing to another LAN

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



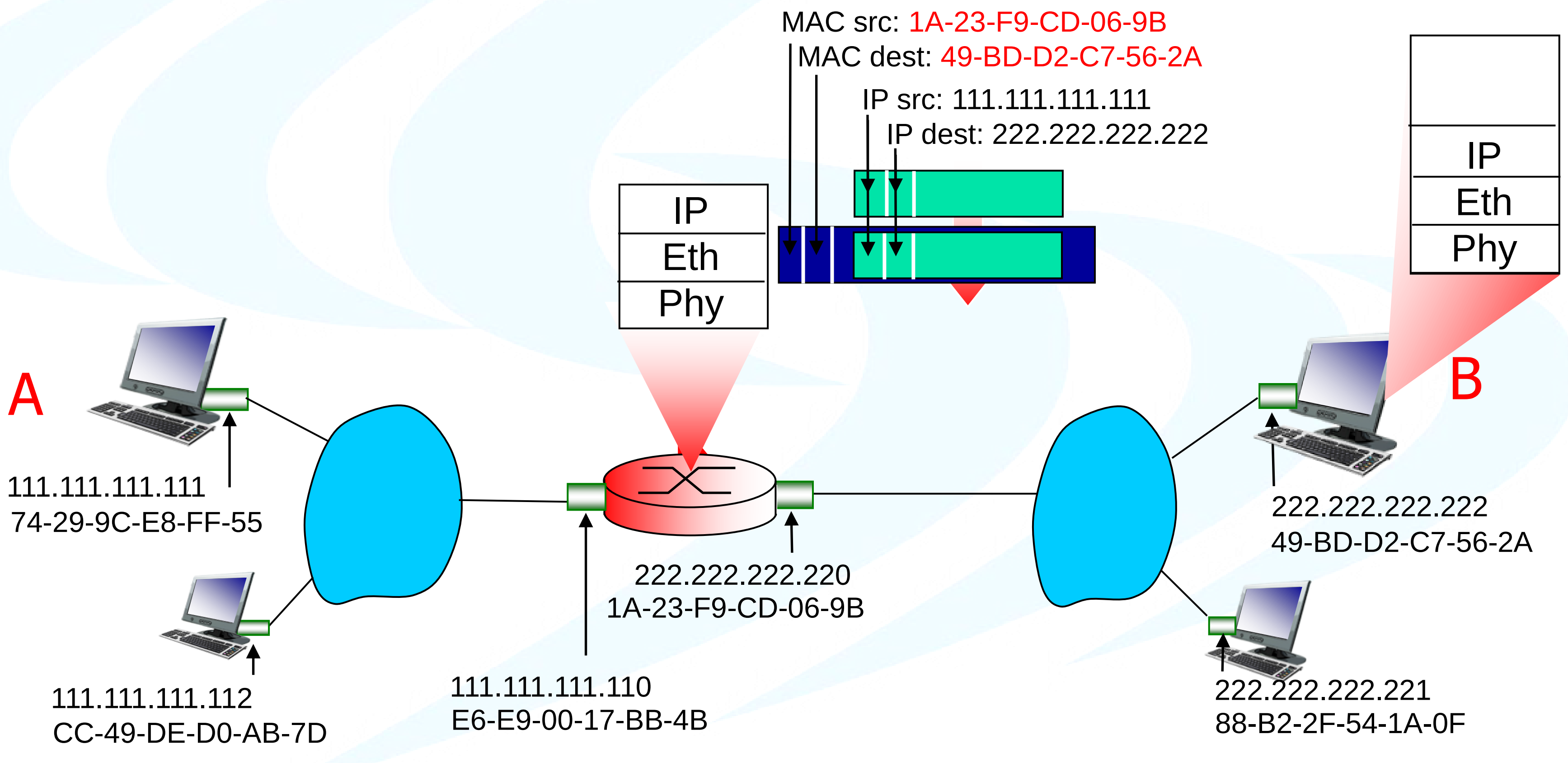
Addressing: routing to another LAN

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



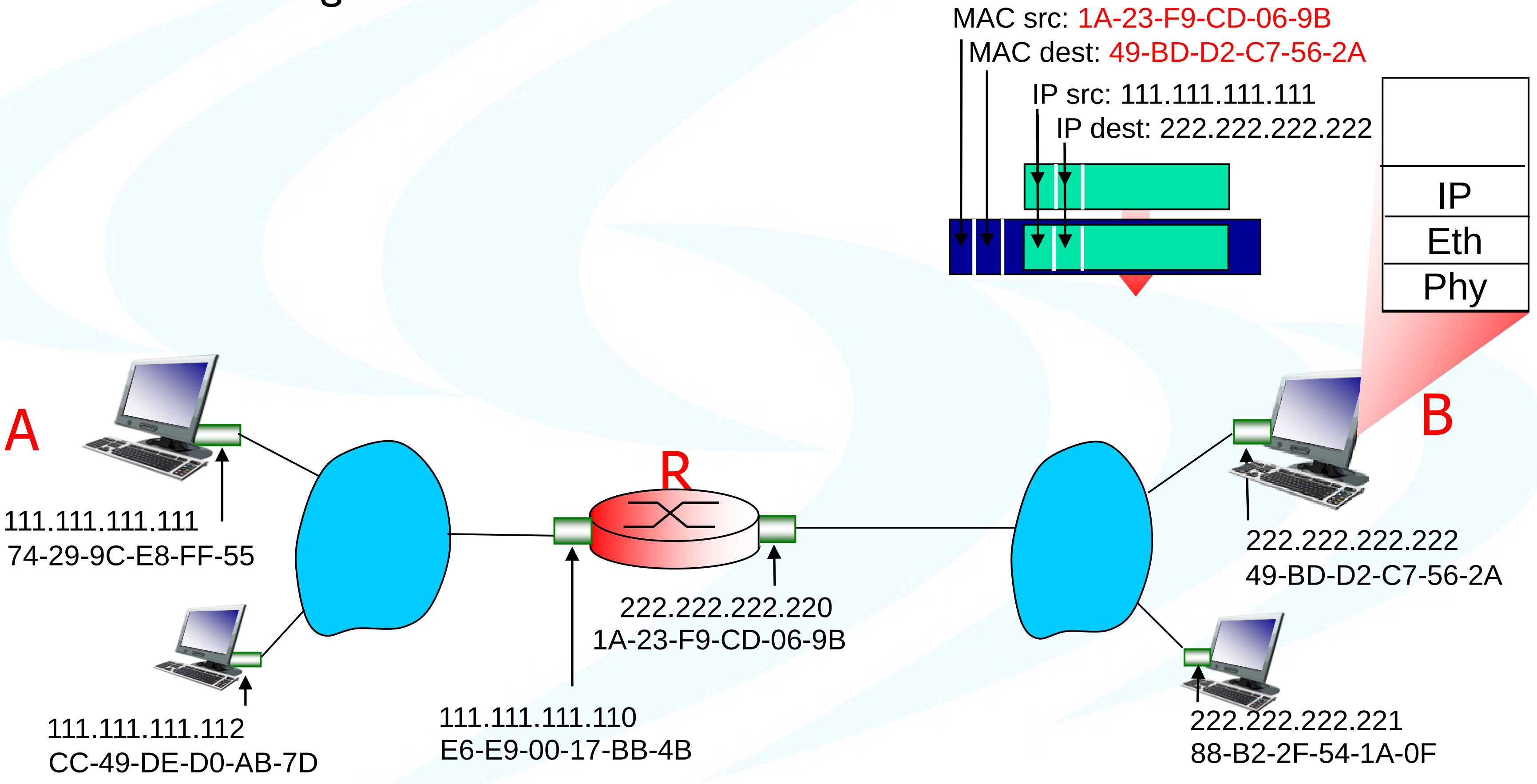
Addressing: routing to another LAN

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



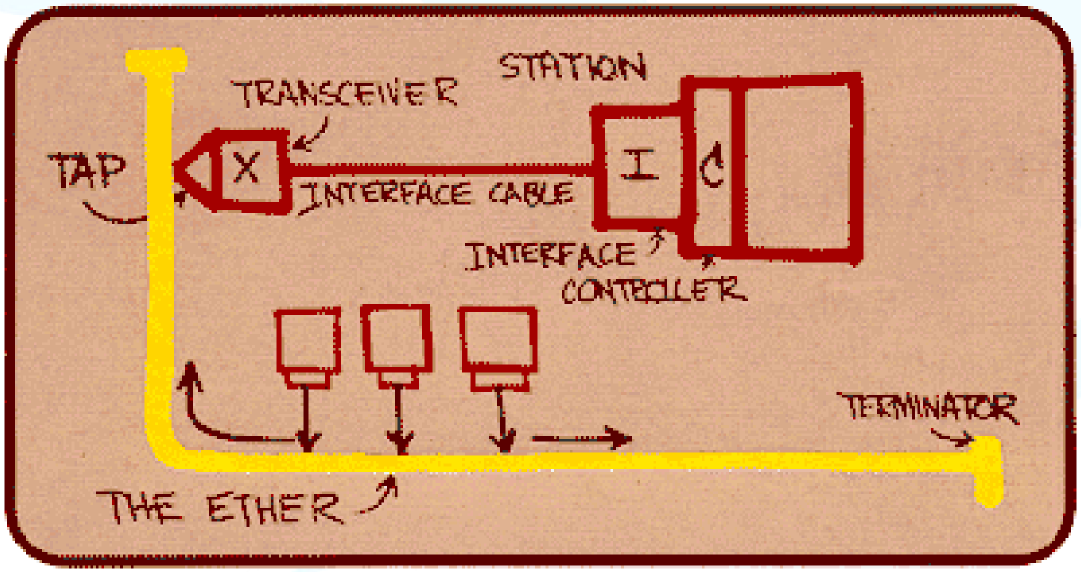
* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

2.5 Ethernet

2.5 Ethernet

"dominant" wired LAN technology:

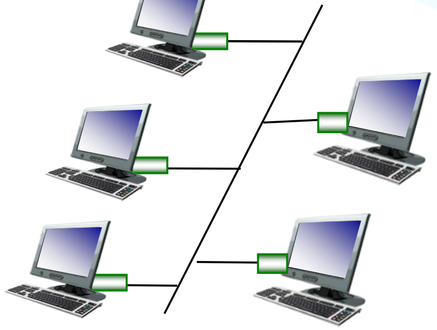
- cheap $20 for 100Mbs!
- first widely used LAN technology
- Simpler, cheaper than token LANs and ATM
- Kept up with speed race: 10 Mbps – 10 Gbps

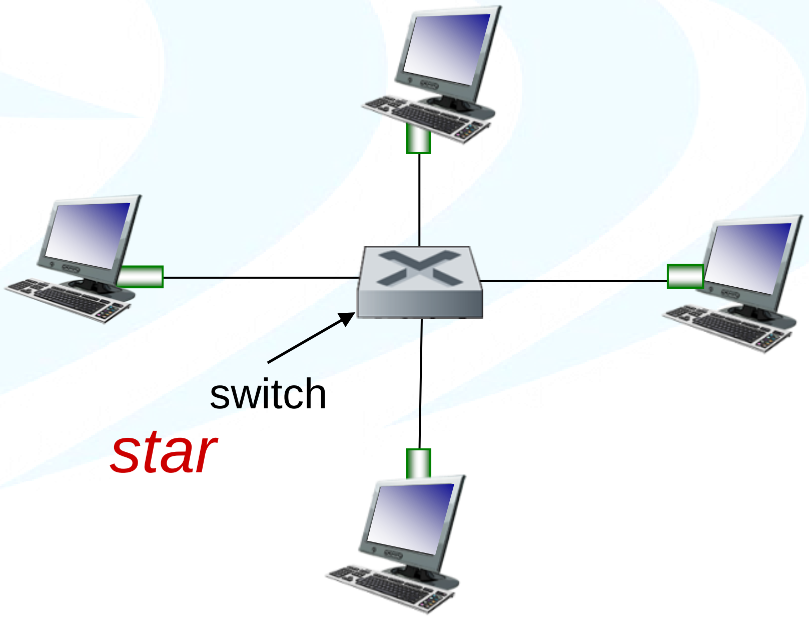


Metcalfe's Ethernet sketch

Ethernet: physical topology

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

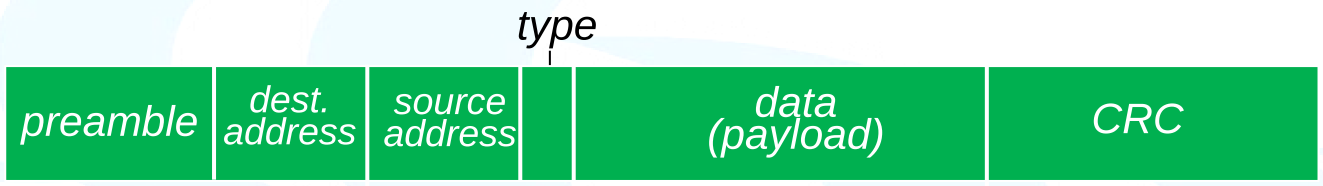

bus: coaxial cable



star

Ethernet Frame Structure

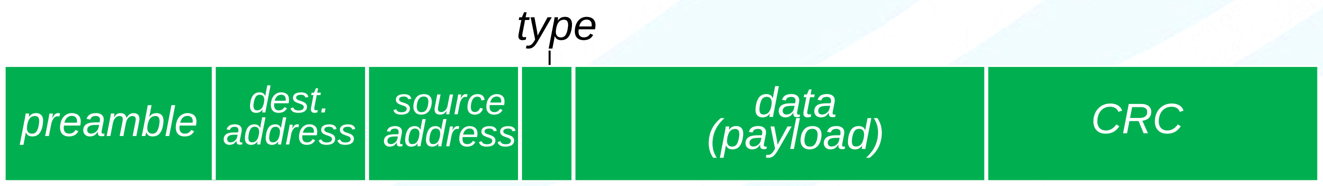
Sending adapter encapsulates IP datagram (or other network layer protocol packet) in **Ethernet frame**



Preamble:

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

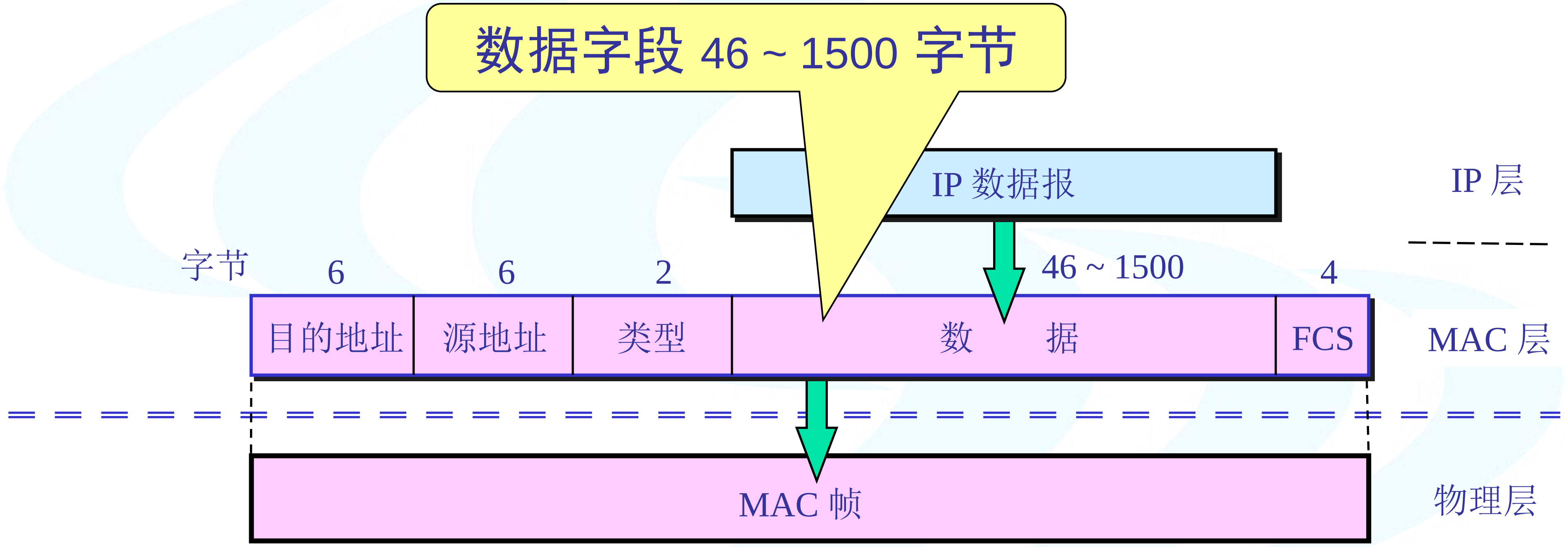
- **Addresses:** 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (eg ARP packet), it passes data in frame to net-layer protocol
  - otherwise, adapter discards frame
- **Type:** indicates the higher layer protocol (mostly IP but others may be supported such as Novell IPX and AppleTalk)
- **CRC:** checked at receiver, if error is detected, the frame is simply dropped



以太网 V2 的 MAC 帧格式

数据字段的正式名称是 MAC 客户数据字段

最小长度 64 字节 – 18 字节的首部和尾部 = 数据字段的最小长度



无效的 MAC 帧

- 帧的长度不是整数个字节；
- 用收到的帧检验序列 **FCS** 查出有差错；
- 数据字段的长度不在 **46 ~ 1500** 字节之间。
- 有效的 **MAC** 帧长度为 **64 ~ 1518** 字节之间。
- 对于检查出的无效 **MAC** 帧就简单地丢弃。以太网不负责重传丢弃的帧。

Unreliable, connectionless service

- *connectionless:* **no handshaking between sending and receiving NICs**
- *unreliable:* **receiving NIC doesn't send acks or nacks to sending NIC**
  - **data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost**
- **Ethernet’s MAC protocol: unslotted *CSMA/CD with binary backoff***

Ethernet uses CSMA/CD

- No slots
- adapter doesn't transmit if it senses that some other adapter is transmitting, that is, carrier sense
- transmitting adapter aborts when it senses that another adapter is transmitting, that is, collision detection
- Before attempting a retransmission, adapter waits a random time, that is, random access

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· 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,…,1023}

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, 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 jam signal
5. After aborting, NIC enters ***binary (exponential) backoff***:
   - after *m*th collision, NIC chooses *K* at random from {*0,1,2, …, 2^m-1*}. NIC waits K·512 bit times, returns to Step 2
   - longer backoff interval with more collisions

CSMA/CD efficiency

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

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

- Efficiency goes to 1 as t_{prop} goes to 0
- Goes to 1 as t_{trans} goes to infinity
- Much better than ALOHA, but still decentralized, simple, and cheap

载波监听多点接入/碰撞检测 CSMA/CD

- **CSMA/CD 表示 Carrier Sense Multiple Access with Collision Detection。**
- “多点接入”表示许多计算机以多点接入的方式连接在一根总线上。
- “载波监听”是指每一个站在发送数据之前先要检测一下总线上是否有其他计算机在发送数据，如果有，则暂时不要发送数据，以免发生碰撞。
- 总线上并没有什么“载波”。因此，“载波监听”就是用电子技术检测总线上有没有其他计算机发送的数据信号。

碰撞检测

- “碰撞检测”就是计算机边发送数据边检测信道上的信号电压大小。
- 当几个站同时在总线上发送数据时，总线上的信号电压摆动值将会增大（互相叠加）。
- 当一个站检测到的信号电压摆动值超过一定的门限值时，就认为总线上至少有两个站同时在发送数据，表明产生了碰撞。
- 所谓“碰撞”就是发生了冲突。因此“碰撞检测”也称为“冲突检测”。

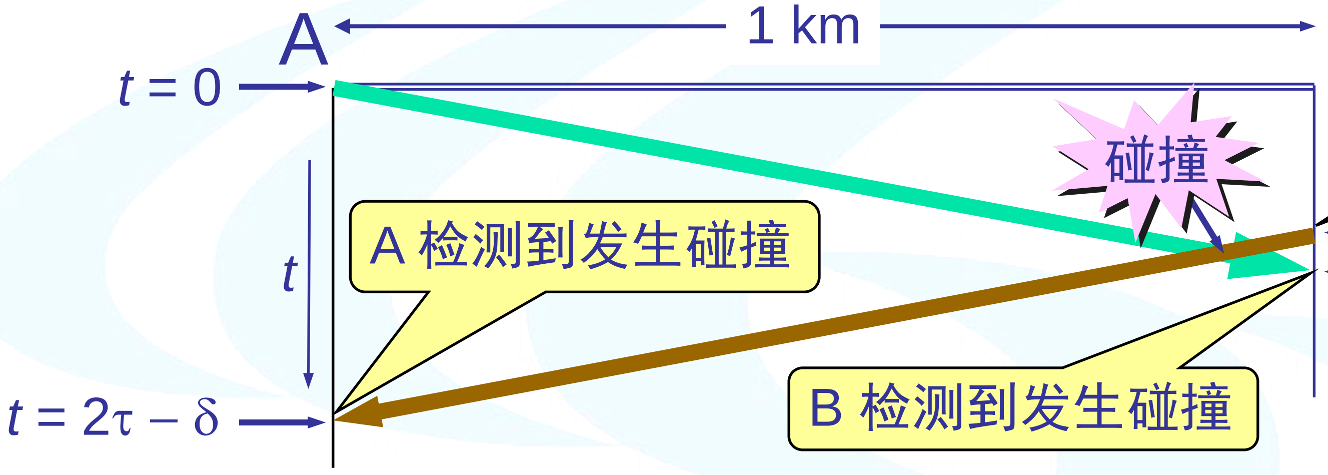
检测到碰撞后

- 在发生碰撞时，总线上传输的信号产生了严重的失真，无法从中恢复出有用的信息来。
- 每一个正在发送数据的站，一旦发现总线上出现了碰撞，就要立即停止发送，免得继续浪费网络资源，然后等待一段随机时间后再次发送。

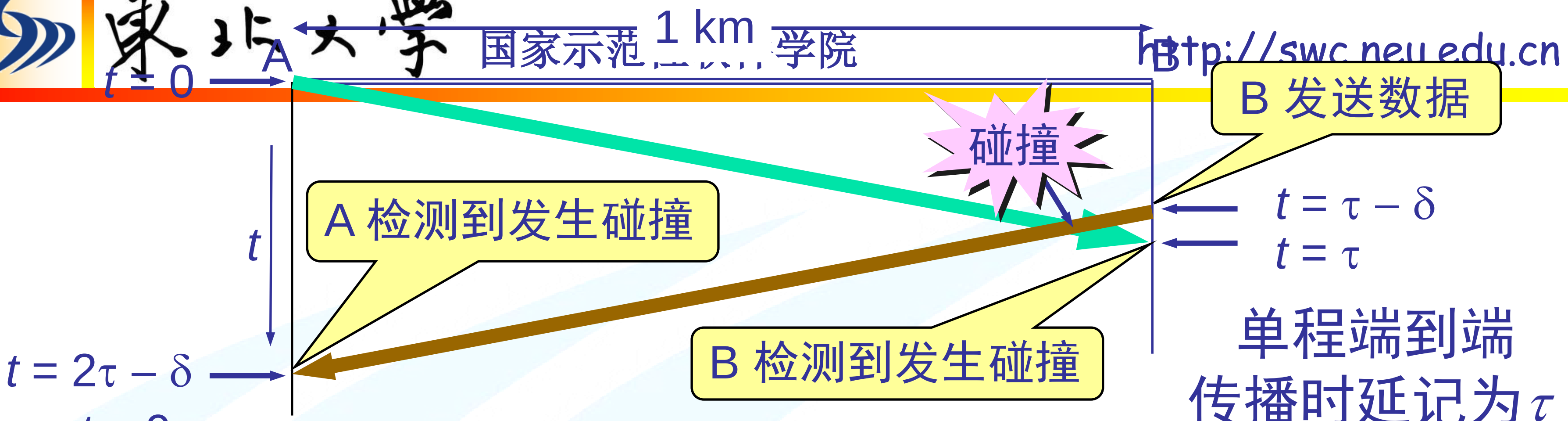
电磁波在总线上的有限传播速率的影响

- 当某个站监听到总线是空闲时，也可能总线并非真正是空闲的。
- **A** 向 **B** 发出的信息，要经过一定的时间后才能传送到 **B**。
- **B** 若在 **A** 发送的信息到达 **B** 之前发送自己的帧(因为这时 **B** 的载波监听检测不到 **A** 所发送的信息)，则必然要在某个时间和 **A** 发送的帧发生碰撞。
- 碰撞的结果是两个帧都变得无用。

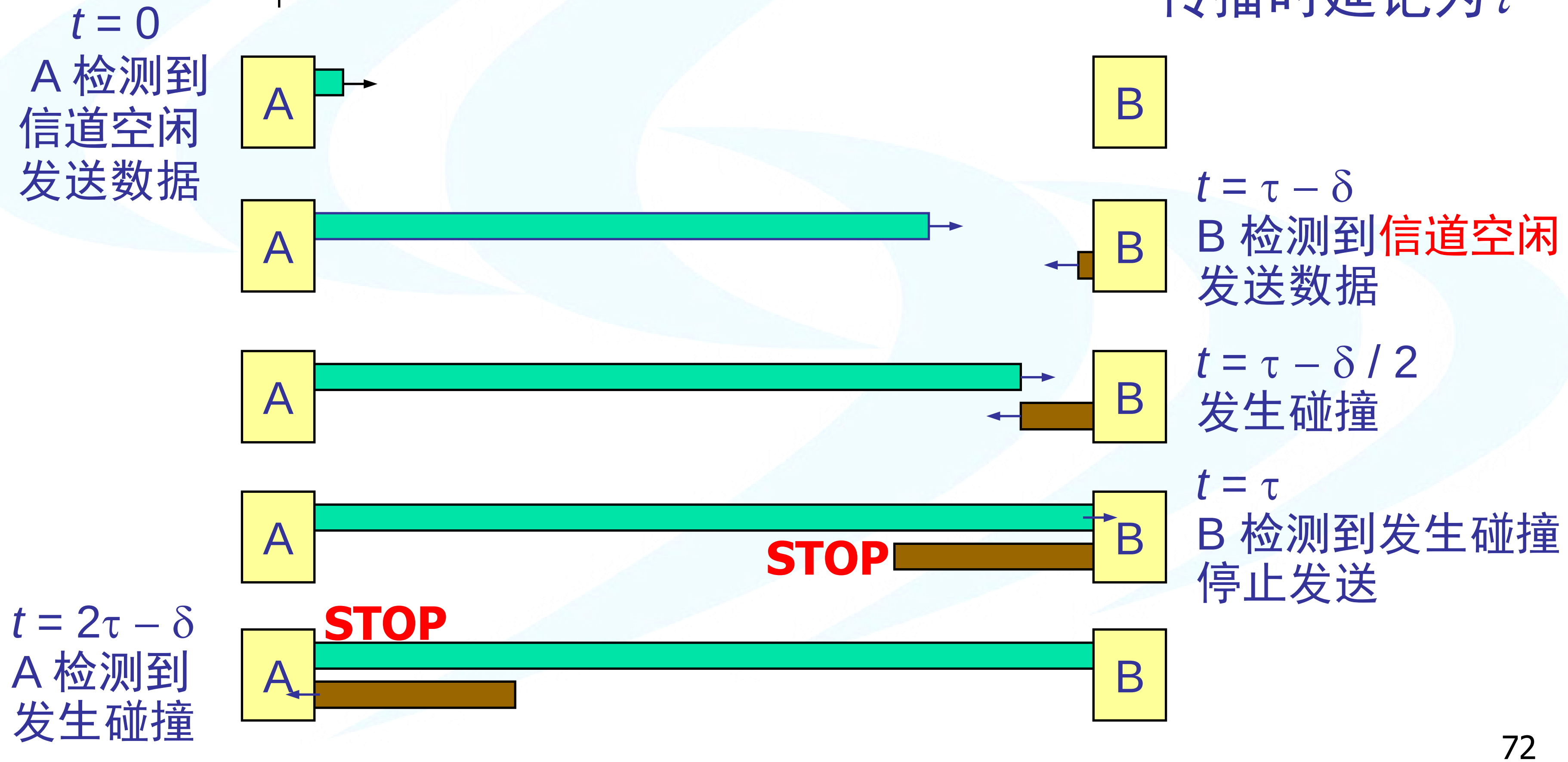
传播时延对载波监听的影响



单程端到端
传播时延记为 τ



单程端到端传播时延记为τ



重要特性

- 使用 **CSMA/CD** 协议的以太网不能进行全双工通信而只能进行双向交替通信（半双工通信）。
- 每个站在发送数据之后的一小段时间内，存在着遭遇碰撞的可能性。
- 这种发送的不确定性使整个以太网的平均通信量远小于以太网的最高数据率。

争用期

- 最先发送数据帧的站，在发送数据帧后至多经过时间 **2**τ（两倍的端到端往返时延）就可知道发送的数据帧是否遭受了碰撞。
- 以太网的端到端往返时延 **2**τ 称为争用期，或碰撞窗口。
- 经过争用期这段时间还没有检测到碰撞，才能肯定这次发送不会发生碰撞。

二进制指数类型退避算法 (truncated binary exponential type)

- 发生碰撞的站在停止发送数据后，要推迟（退避）一个随机时间才能再发送数据。
  - **确定基本退避时间，一般是取为争用期 2τ。**
  - **定义重传次数 *k*， *k* ≤ 10，即**

    k = Min[重传次数, 10]
  - **从整数集合[0,1,…, (2^k –1)]中随机地取出一个数，记为 *r*。重传所需的时延就是 *r* 倍的基本退避时间。**
  - **当重传达 16 次仍不能成功时即丢弃该帧，并向高层报告。**

争用期的长度

- 以太网取 **51.2 μs** 为争用期的长度。
- 对于 **10 Mb/s** 以太网，在争用期内可发送**512 bit**，即 **64** 字节。
- 以太网在发送数据时，若前 **64** 字节没有发生冲突，则后续的数据就不会发生冲突。

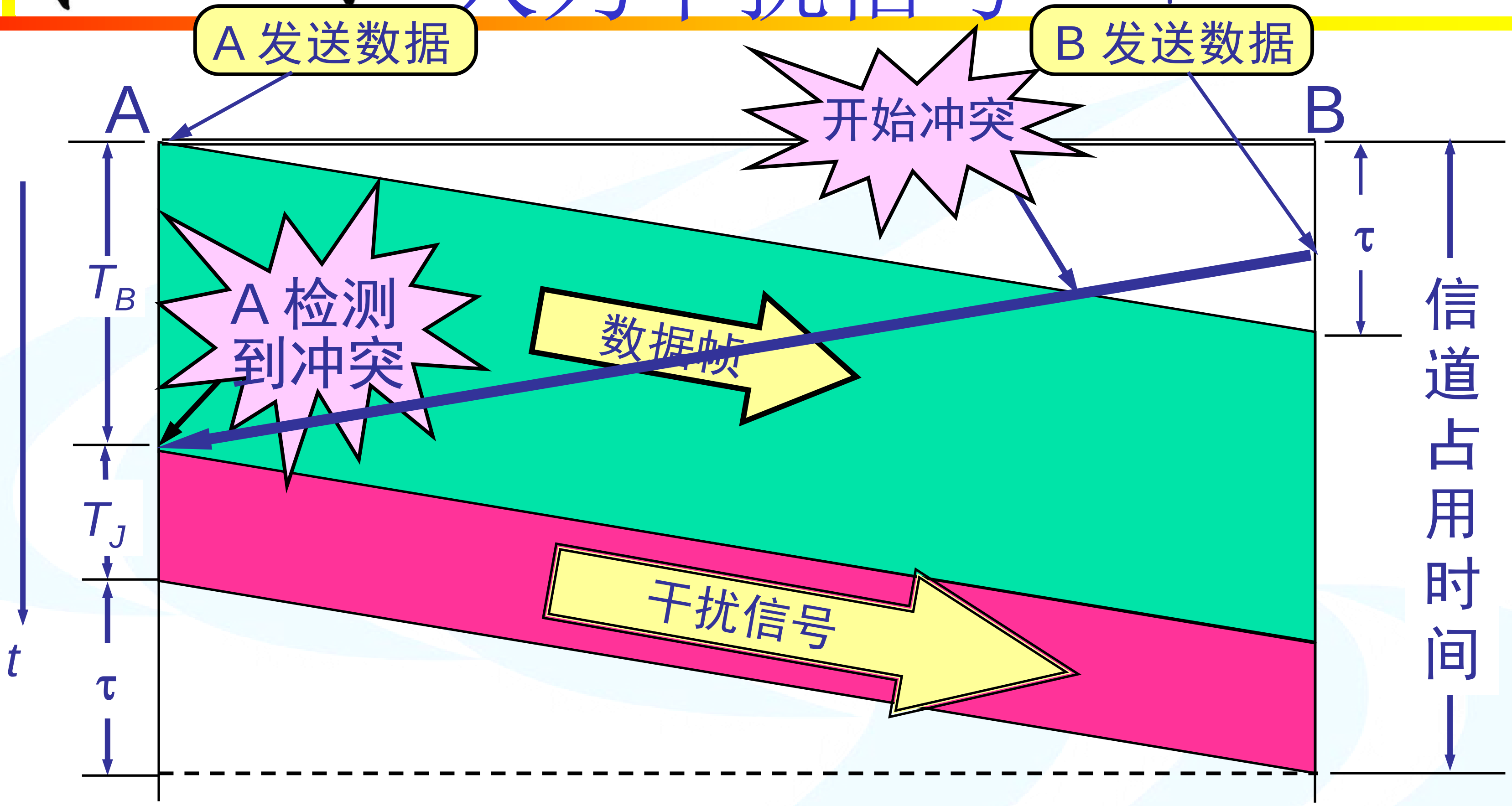
最短有效帧长

- 如果发生冲突，就一定是在发送的前 **64** 字节之内。
- 由于一检测到冲突就立即中止发送，这时已经发送出去的数据一定小于 **64** 字节。
- 以太网规定了最短有效帧长为 **64** 字节，凡长度小于 **64** 字节的帧都是由于冲突而异常中止的无效帧。

强化碰撞

- 当发送数据的站一旦发现发生了碰撞时：
  - **立即停止发送数据；**
  - **再继续发送若干比特的人为干扰信号 (jamming signal)，以便让所有用户都知道现在已经发生了碰撞。**

人为干扰信号



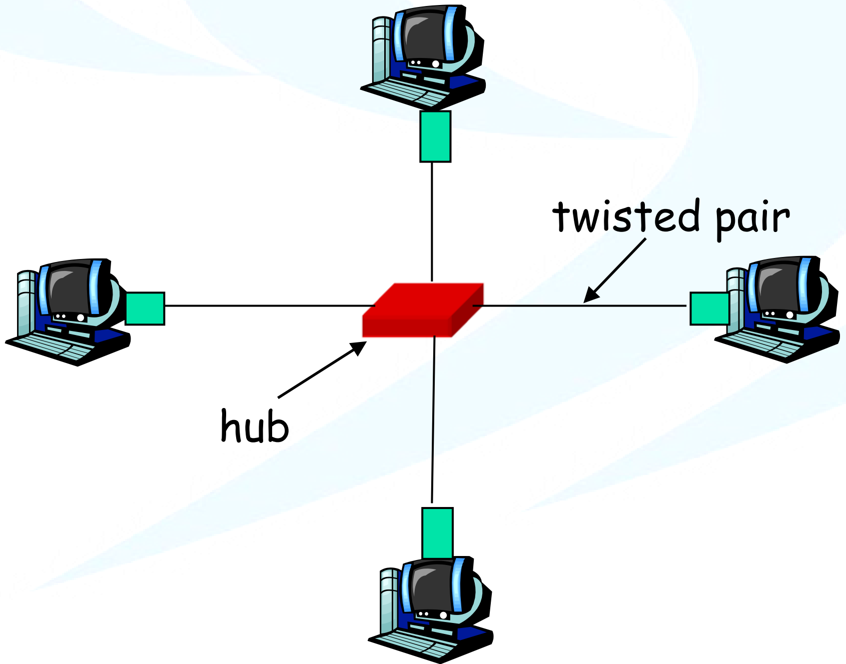
B 也能够检测到冲突，并立即停止发送数据帧，接着就发送干扰信号。这里为了简单起见，只画出 A 发送干扰信号的情况。

帧间最小间隔

- 帧间最小间隔为 **9.6 μs**，相当于 **96 bit** 的发送时间。
- 一个站在检测到总线开始空闲后，还要等待 **9.6 μs** 才能再次发送数据。
- 这样做是为了使刚刚收到数据帧的站的接收缓存来得及清理，做好接收下一帧的准备。

10BaseT and 100BaseT

- 10/100 Mbps rate; latter called “fast ethernet”
- T stands for Twisted Pair
- Nodes connect to a hub: “star topology”; 100 m max distance between nodes and hub



Gbit Ethernet

- uses standard Ethernet frame format
- allows for point-to-point links and shared broadcast channels
- in shared mode, CSMA/CD is used; short distances between nodes required for efficiency
- uses hubs, called here "Buffered Distributors"
- Full-Duplex at 1 Gbps for point-to-point links
- 10 Gbps now !

2.6 Interconnections: Hubs and switches

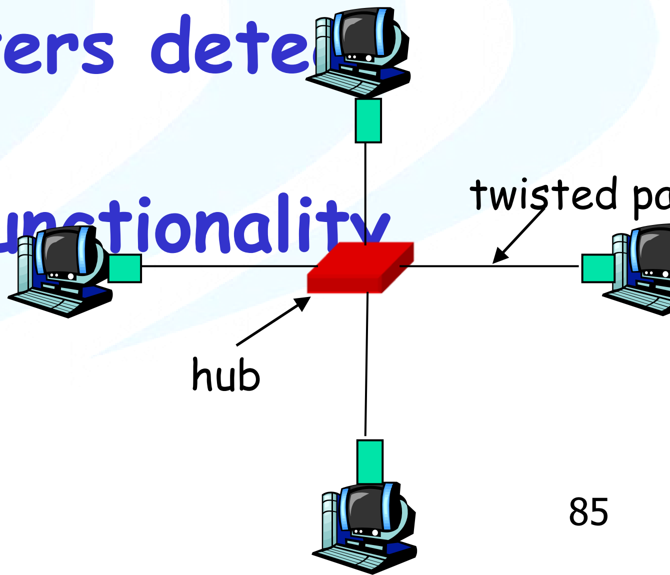
互连局域网网段的网络设备

- 中继器：（物理层）
- 转换器：信号转换的中继器
  - 如光电转换, **10base-2**转换到**10BaseT**
- 集线器：（物理层）
  - 集线器是中继器的一种形式，也称为多端口中继器。
- 交换机：（链路层）
  - 交换机是网桥的一种形式，也称为多端口网桥。

Hubs

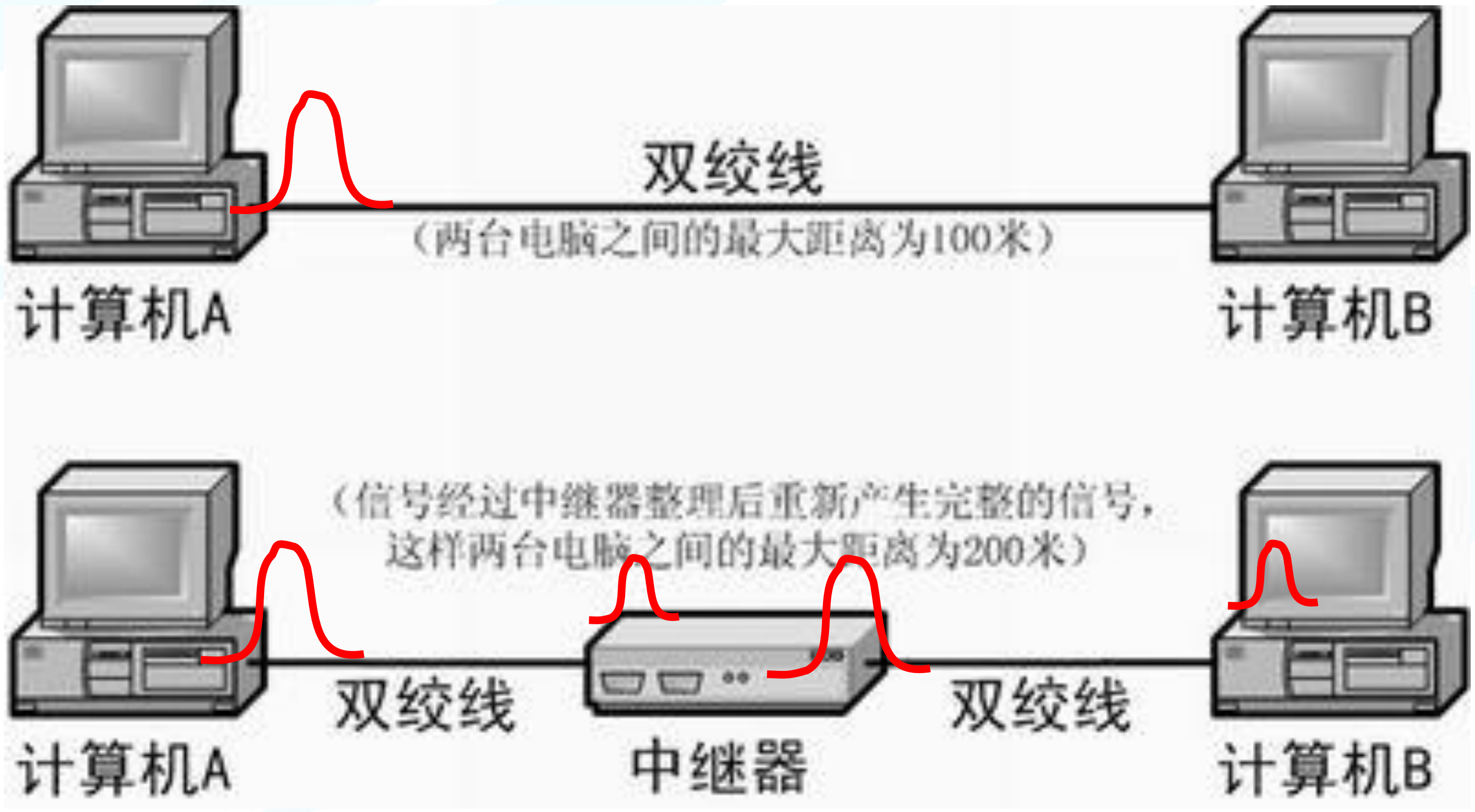
Hubs are essentially physical-layer repeaters:

- bits coming from one link go out all other links
- at the same rate
- no frame buffering
- no CSMA/CD at hub: adapters detect collisions
- provides net management functionality

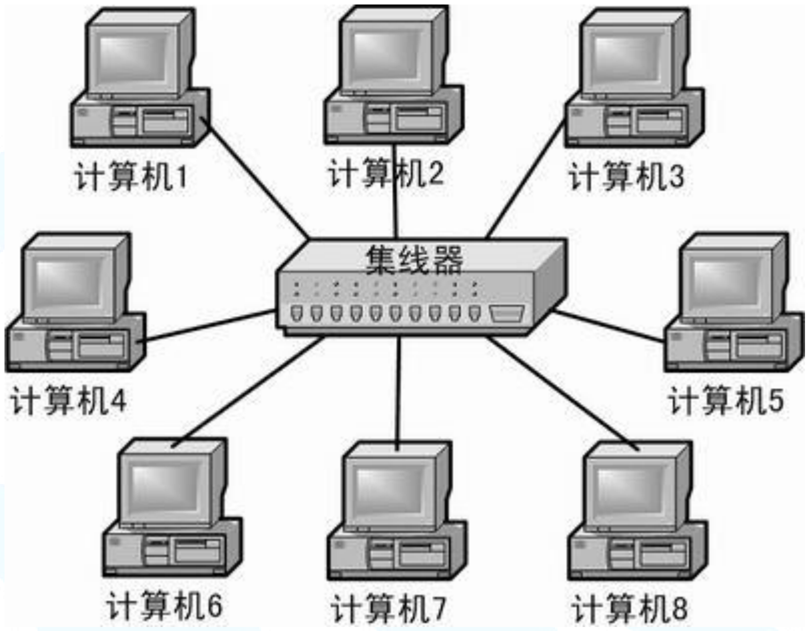


The working principle of HUB

A、peer to peer


双绞线
（两台电脑之间的最大距离为100米）
计算机A
计算机B
（信号经过中继器整理后重新产生完整的信号，这样两台电脑之间的最大距离为200米）
双绞线
中继器
双绞线
计算机A
计算机B

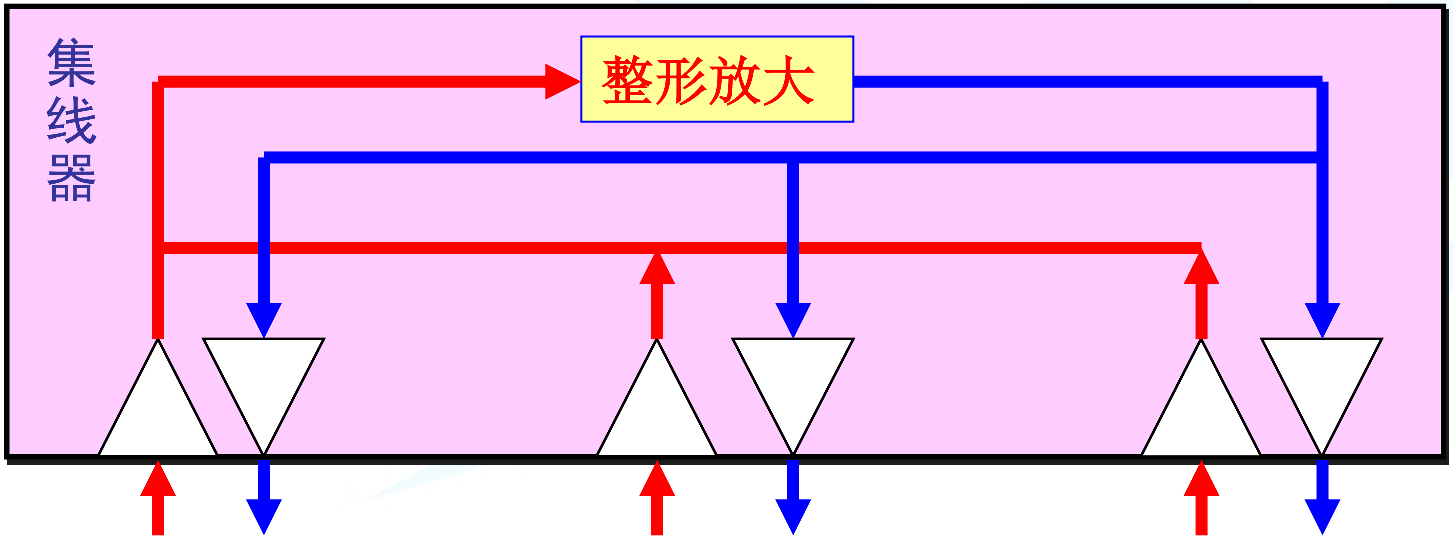
B、点到多点通信



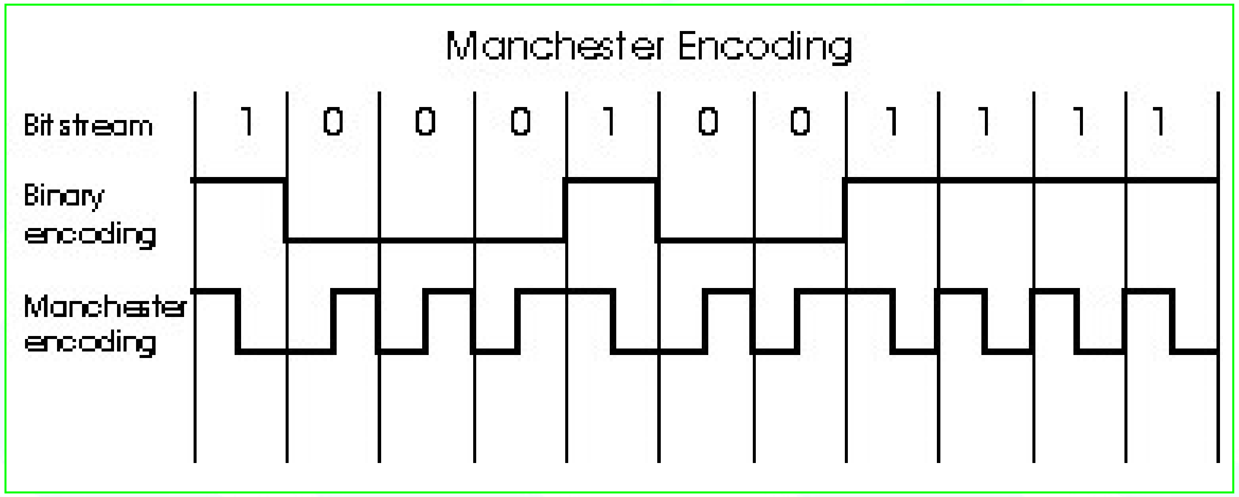
The function of HUB：

Forwarding

整形放大



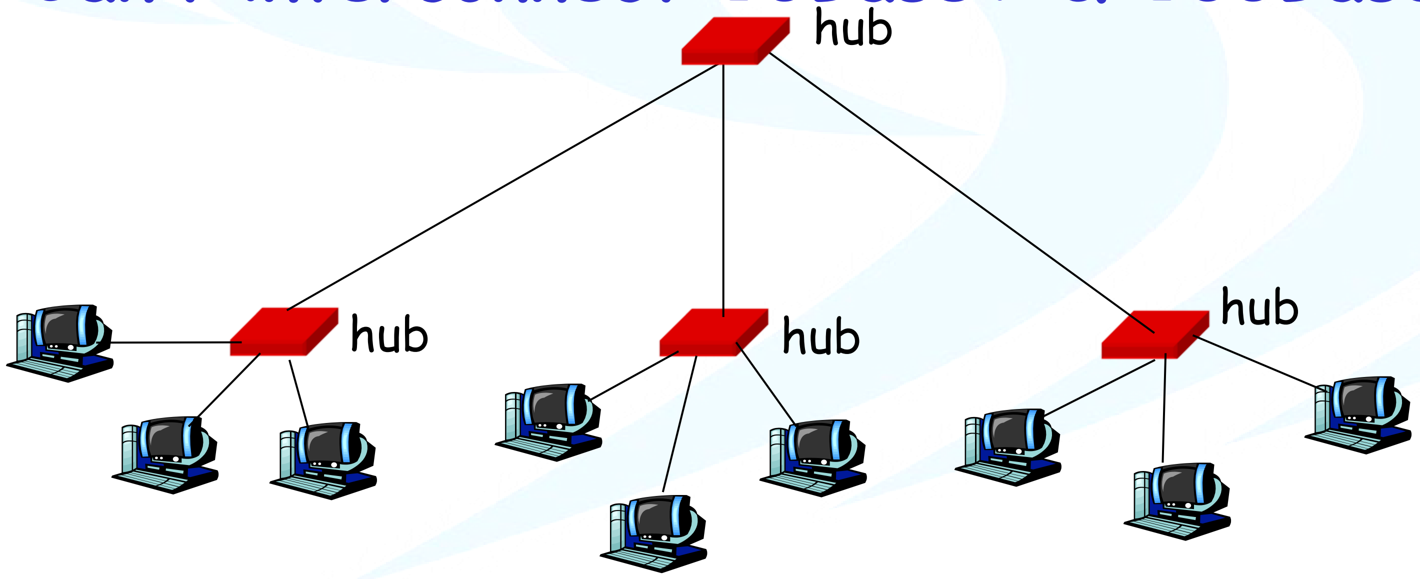
Manchester encoding



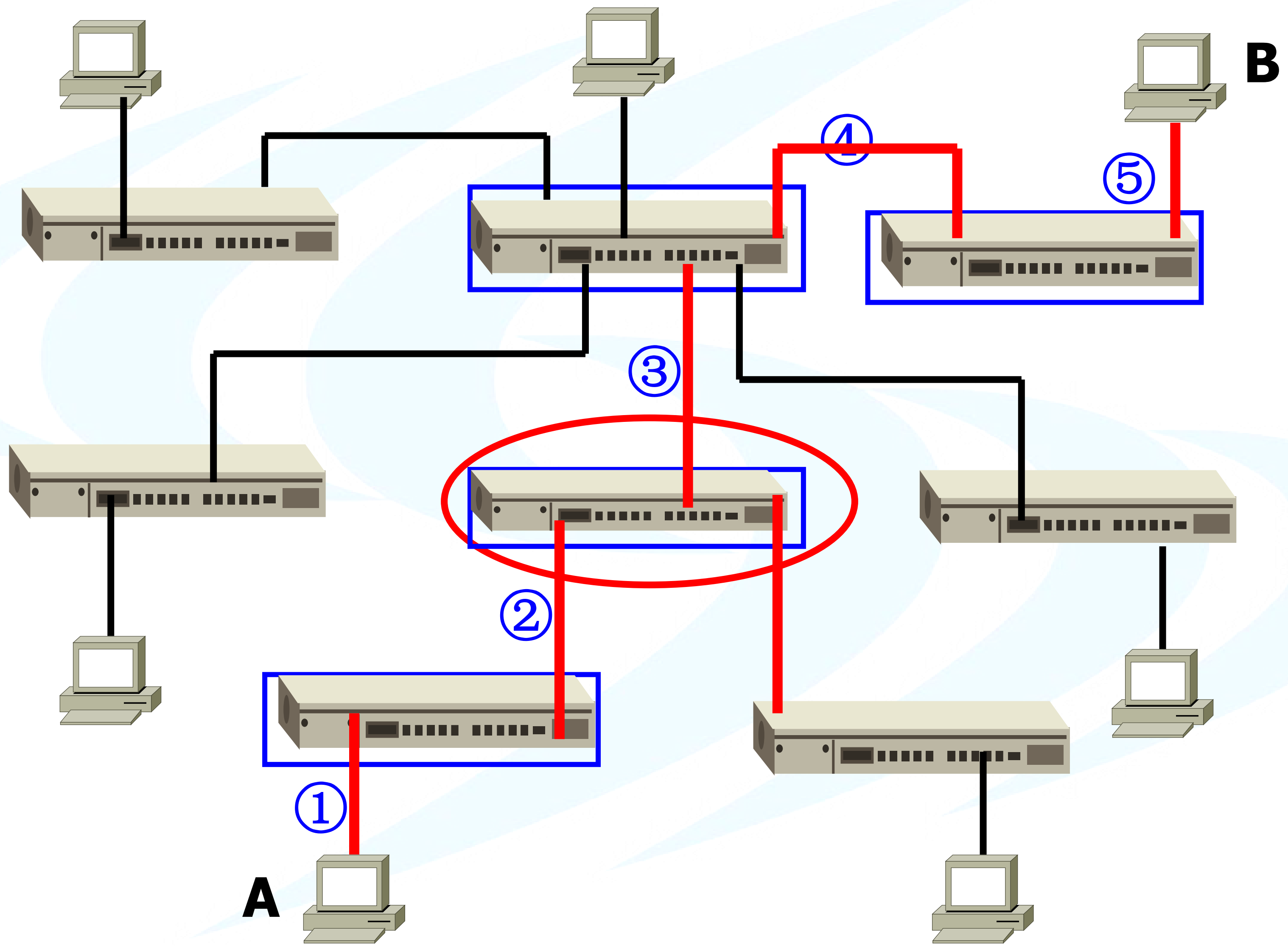
- Used in 10BaseT
- Each bit has a transition
- Allows clocks in sending and receiving nodes to synchronize to each other
  - no need for a centralized, global clock among nodes!
- Hey, this is physical-layer stuff!

Interconnecting with hubs

- Backbone hub interconnects LAN segments
- Extends max distance between nodes
- But individual segment collision domains become one large collision domain
- Can't interconnect 10BaseT & 100BaseT



5-4-3-2-1 Rule

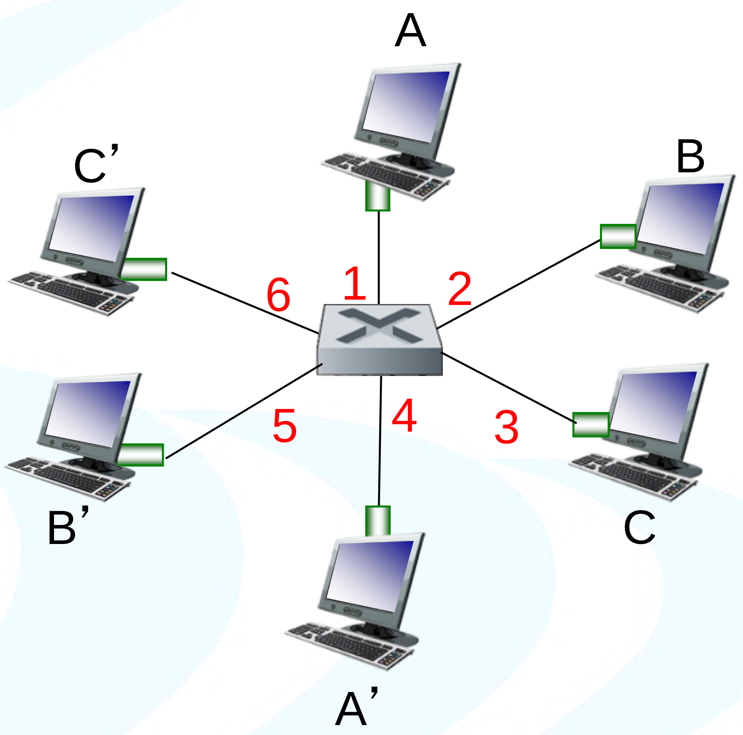


Ethernet switch

- **link-layer device: takes an *active* role**
  - **store, forward Ethernet frames**
  - **examine incoming frame’s MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses 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: *multiple* simultaneous transmissions

- **hosts have dedicated, direct connection to switch**
- **switches buffer packets**
- **Ethernet protocol used on *each* incoming link, but no collisions; full duplex**
  - **each link is its own collision domain**
- ***switching:* A-to-A' and B-to-B' can transmit simultaneously, without collisions**



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

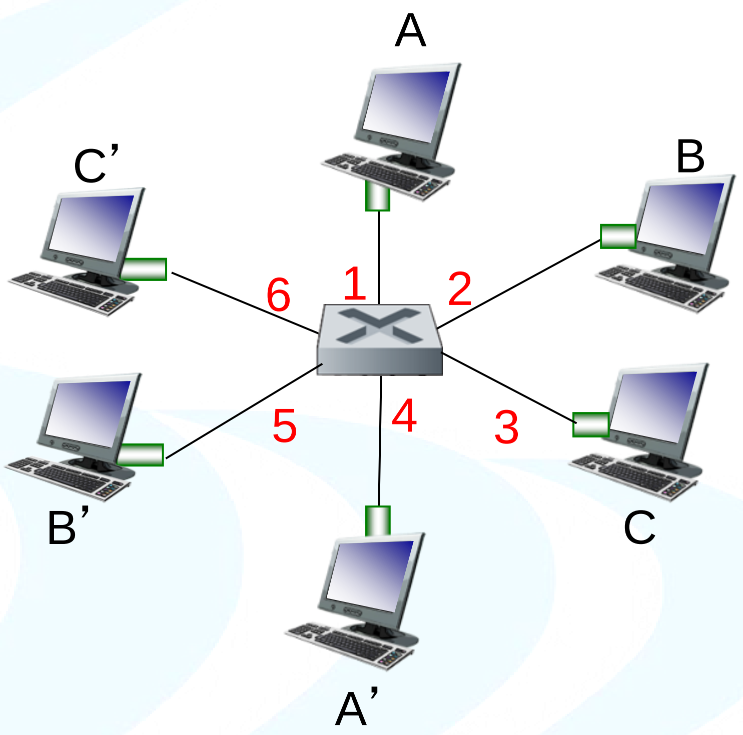
Switch forwarding table

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

- A: each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
  - looks like a routing table!

Q: how are entries created, maintained in switch table?

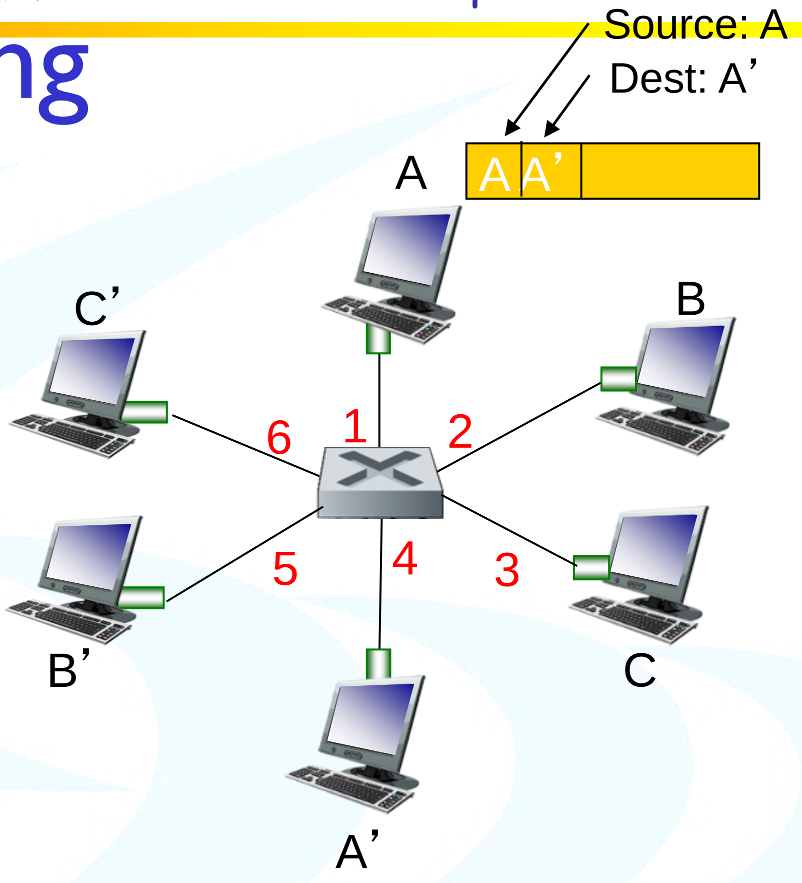
- something like a routing protocol?



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

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*

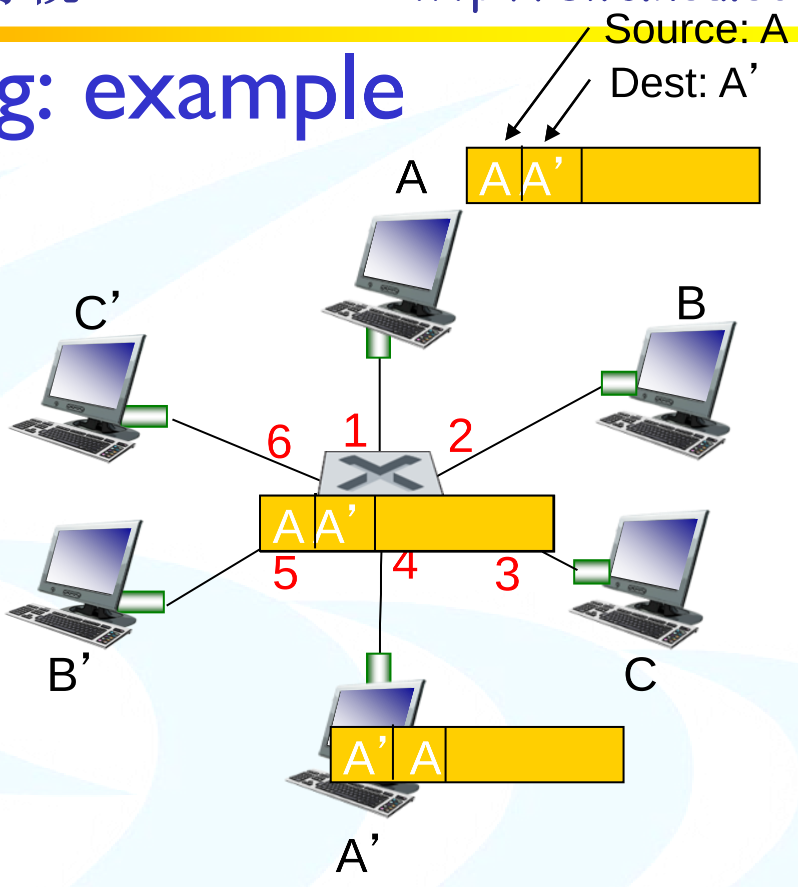
Switch table (initially empty)

Switch: frame filtering/forwarding

```
when  frame received at switch:
   1. record incoming link, MAC address of sending host
   2. index switch table using MAC destination address
   3. if entry found for destination
        then {
        if destination on segment from which frame
      arrived
            then drop frame
            else forward frame on interface indicated by
      entry
         }
        else flood  /* forward on all interfaces except
      arriving  interface */
```

Self-learning, forwarding: example

- **frame destination, A', location unknown:** *flood*
  - destination A location known: selectively send on just one link

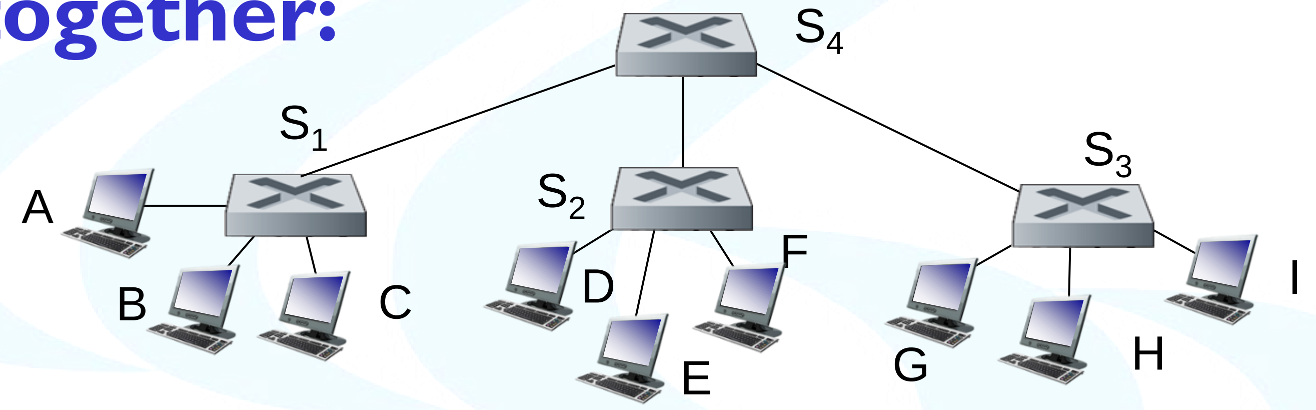


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

self-learning switches can be connected together:

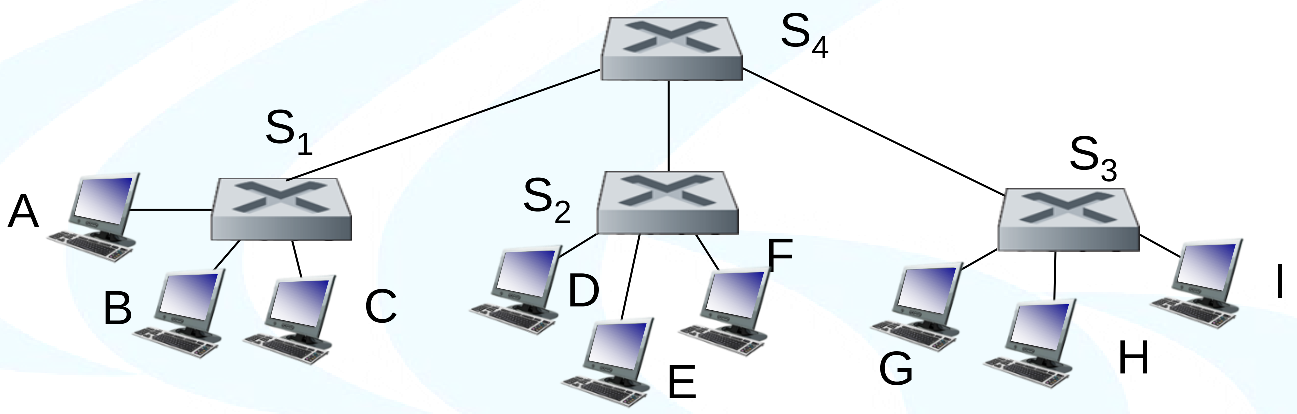


Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?

- A: self learning! (works exactly the same as in single-switch case!)

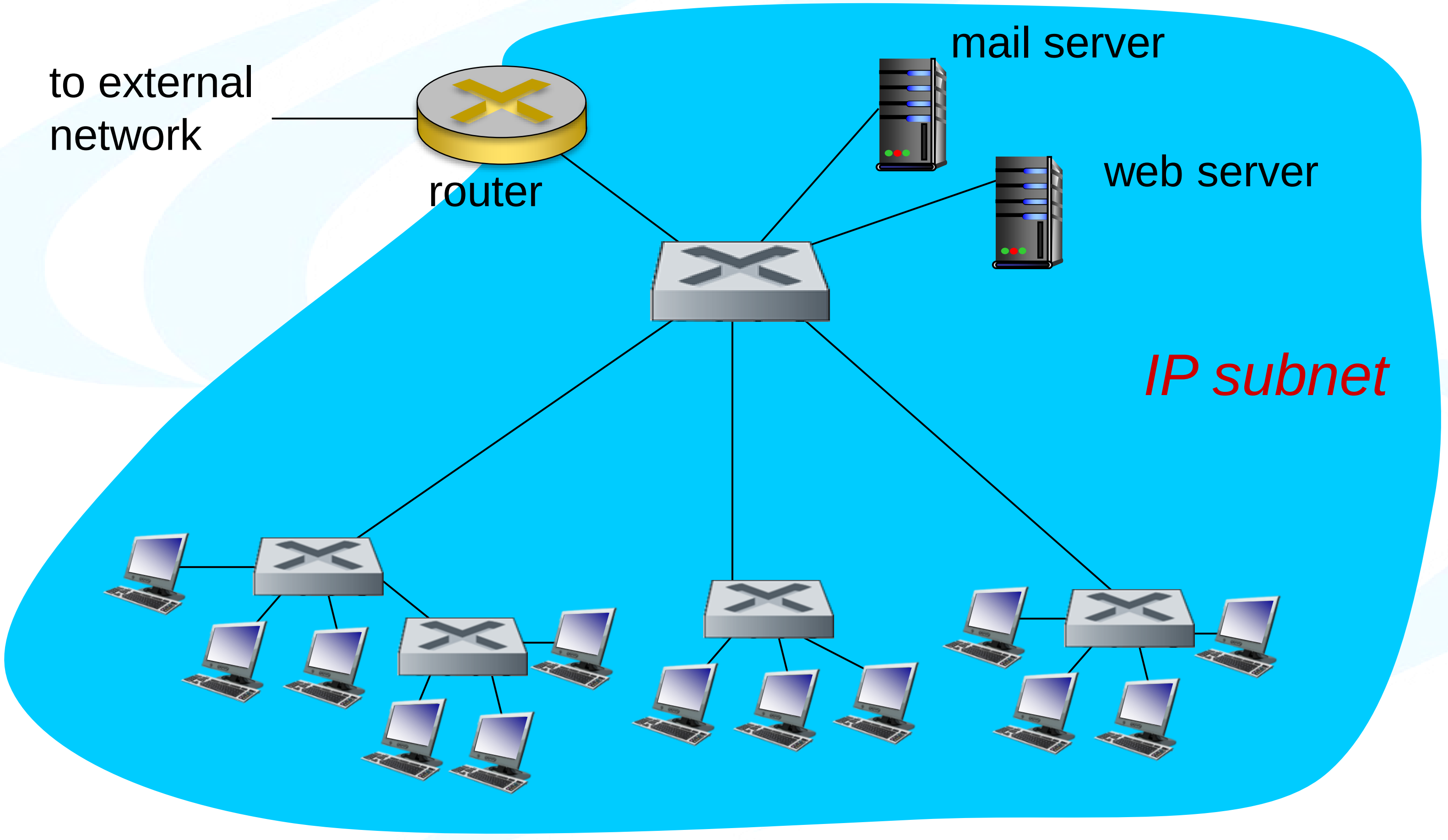
Self-learning multi-switch example

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



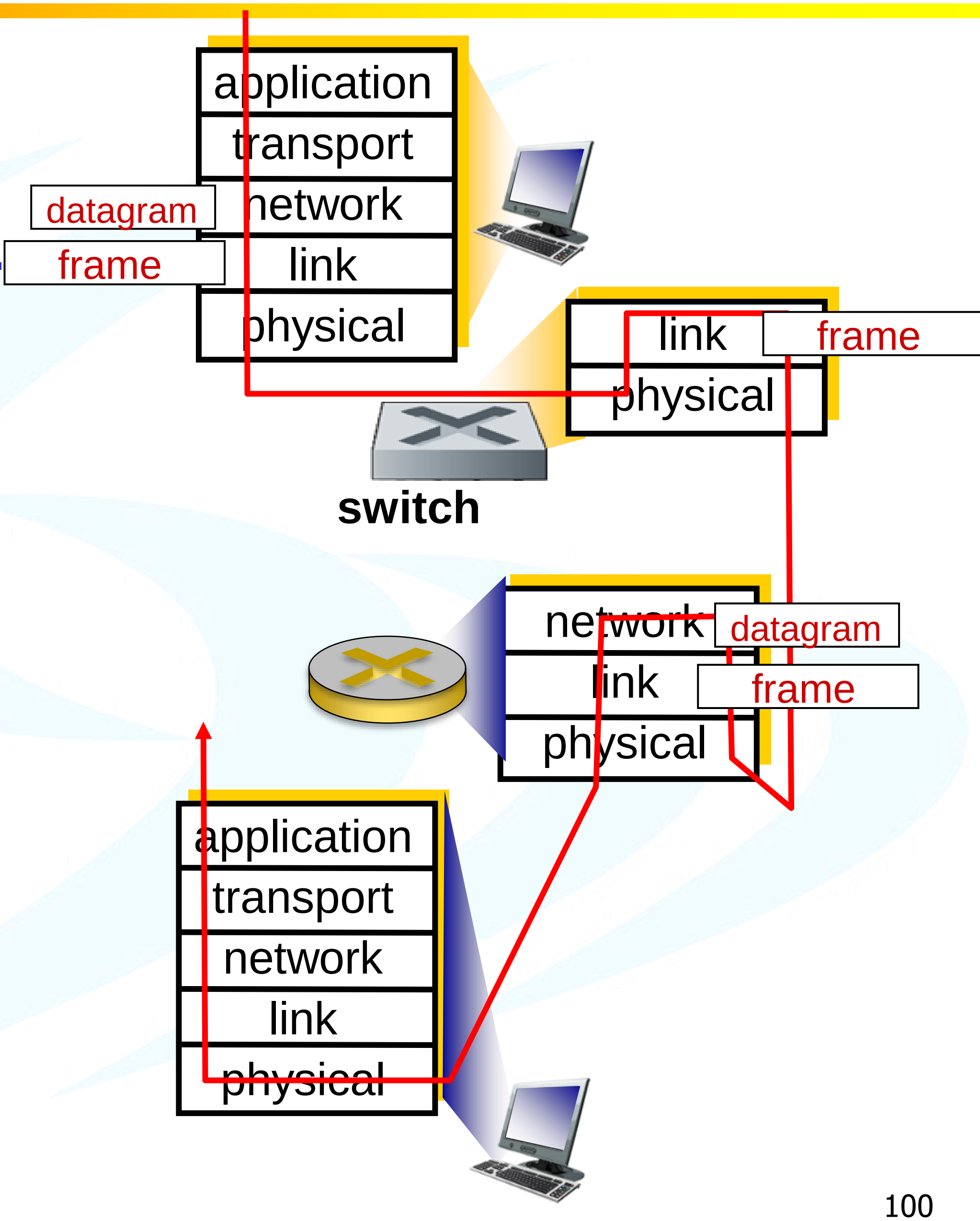
Switches vs. routers

both are store-and-forward:

- ***routers:*** **network-layer devices (examine network-layer headers)**
- ***switches:*** **link-layer devices (examine link-layer headers)**

both have forwarding tables:

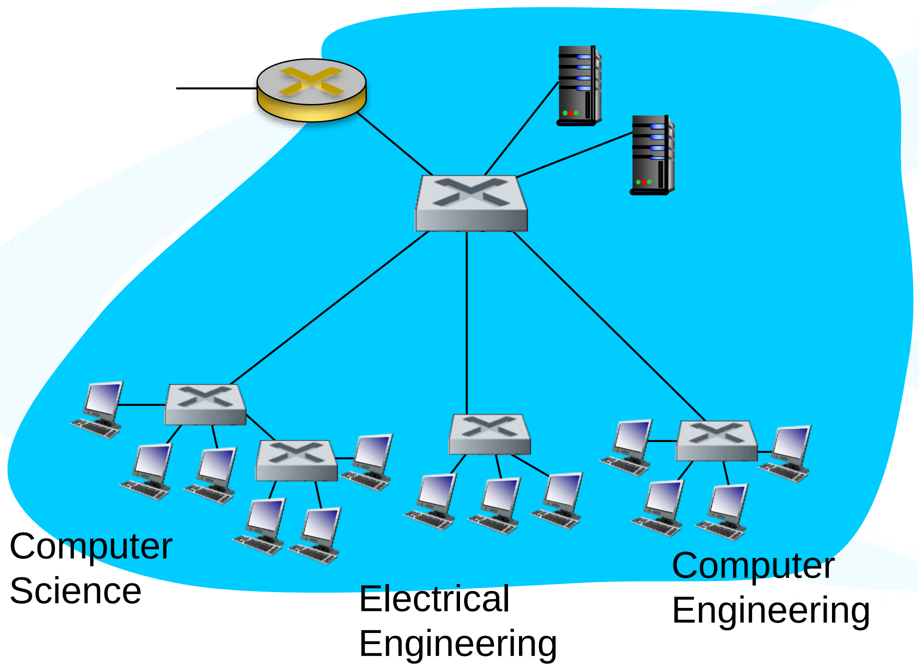
- ***routers:*** **compute tables using routing algorithms, IP addresses**
- ***switches:*** **learn forwarding table using flooding, learning, MAC addresses**



Summary comparison

	hubs	routers	switches
traffic isolation	no	yes	yes
plug & play	yes	no	yes
optimal routing	no	yes	no
cut through	yes	no	yes

VLANs: motivation



consider:

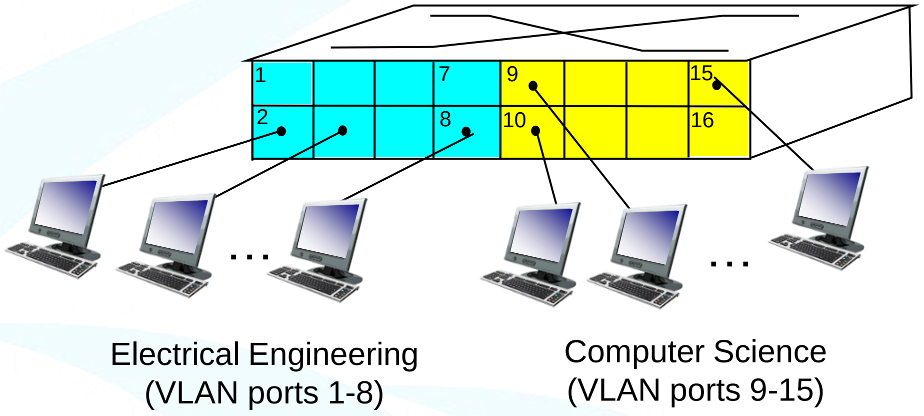
- **CS user moves office to EE, but wants connect to CS switch?**
- **single broadcast domain:**
  - **all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN**
  - **security/privacy, efficiency issues**

VLANs

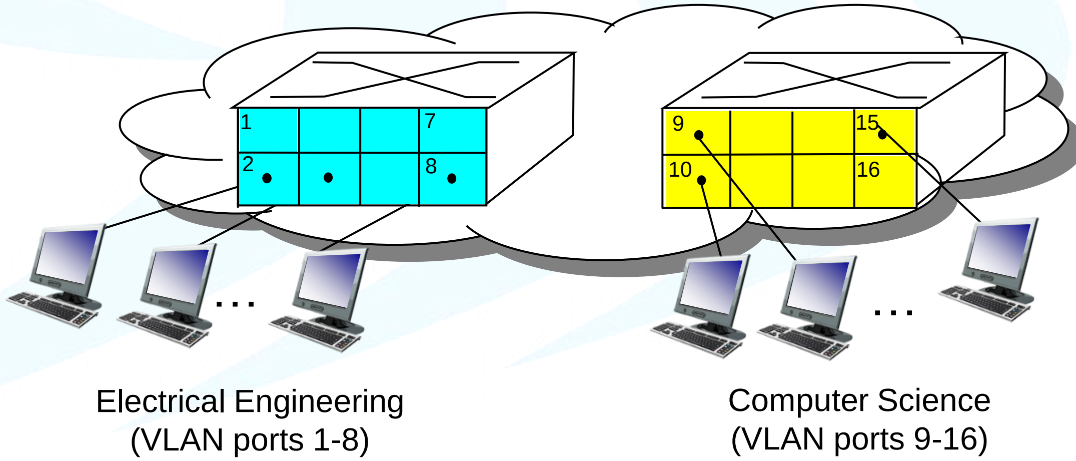
Virtual Local Area Network

switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch ……


1
2
7
8
9
10
15
16
…
…
Electrical Engineering (VLAN ports 1-8)
Computer Science (VLAN ports 9-15)

… operates as multiple virtual switches

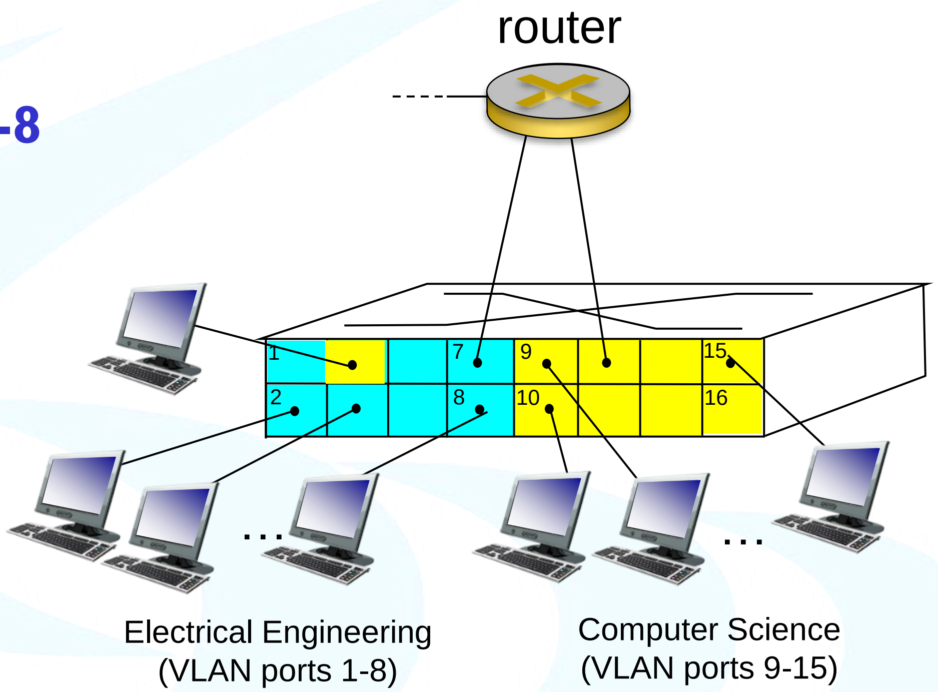

1
2
7
8
9
10
15
16
…
…
Electrical Engineering (VLAN ports 1-8)
Computer Science (VLAN ports 9-16)

Port-based VLAN

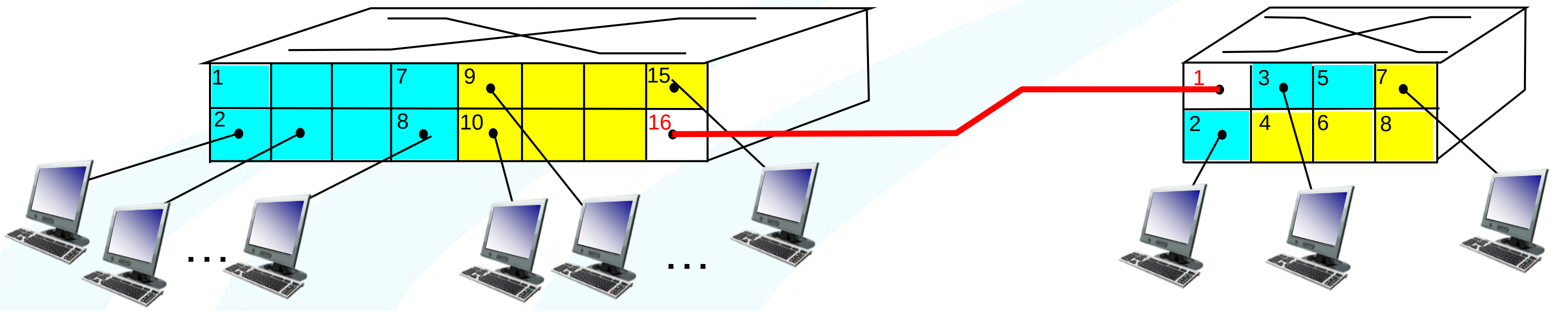
- ***traffic isolation:*** **frames to/from ports 1-8 can *only* reach ports 1-8**
  - **can also define VLAN based on MAC addresses of endpoints, rather than switch port**

- dynamic membership: ports can be dynamically assigned among VLANs

- forwarding between VLANS: done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers


router
1
2
7
8
9
10
15
16
…
Electrical Engineering (VLAN ports 1-8)
Computer Science (VLAN ports 9-15)

VLANS spanning multiple switches



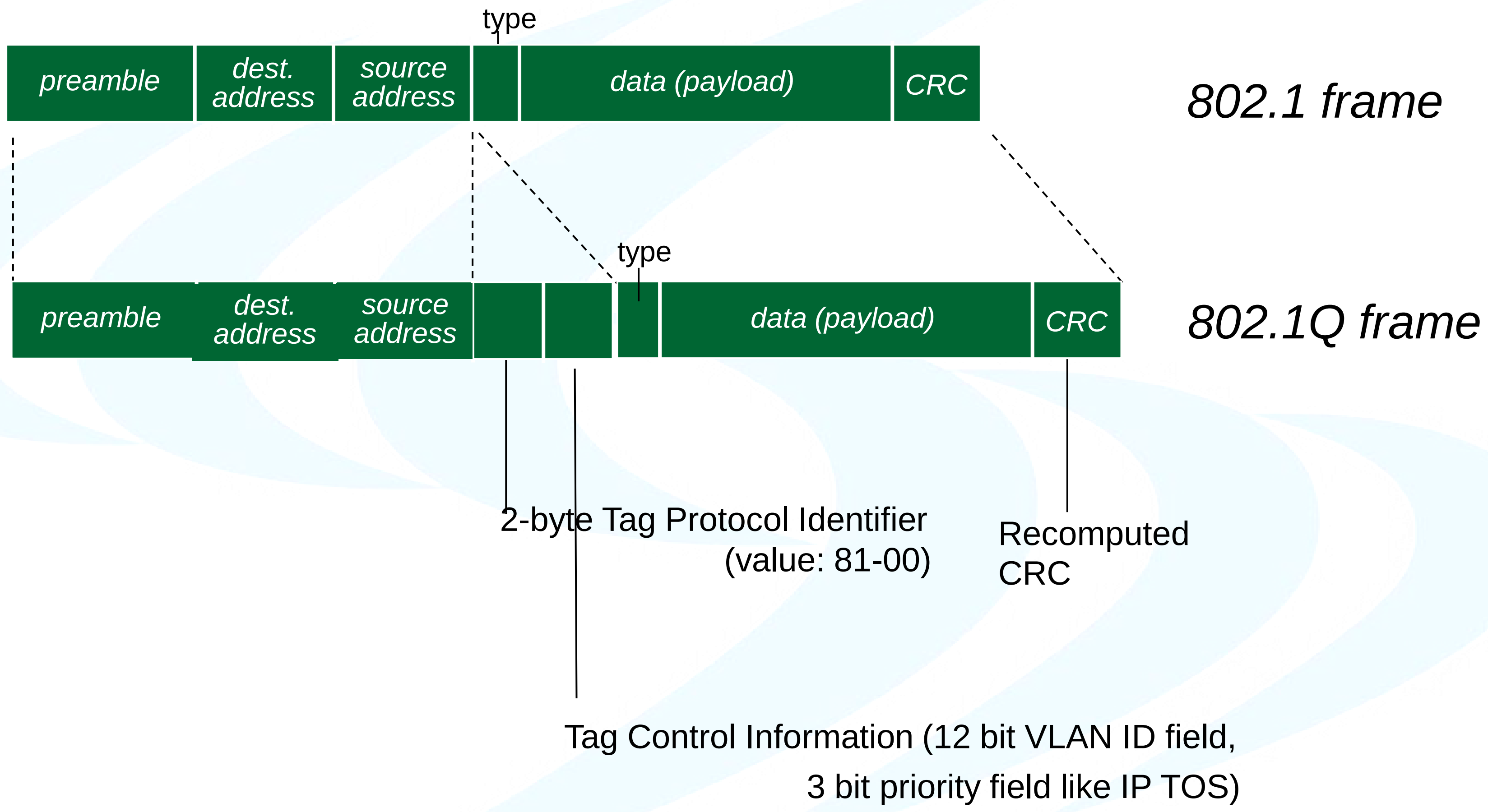
Electrical Engineering (VLAN ports 1-8)

Computer Science (VLAN ports 9-15)

Ports 2,3,5 belong to EE VLAN
Ports 4,6,7,8 belong to CS VLAN

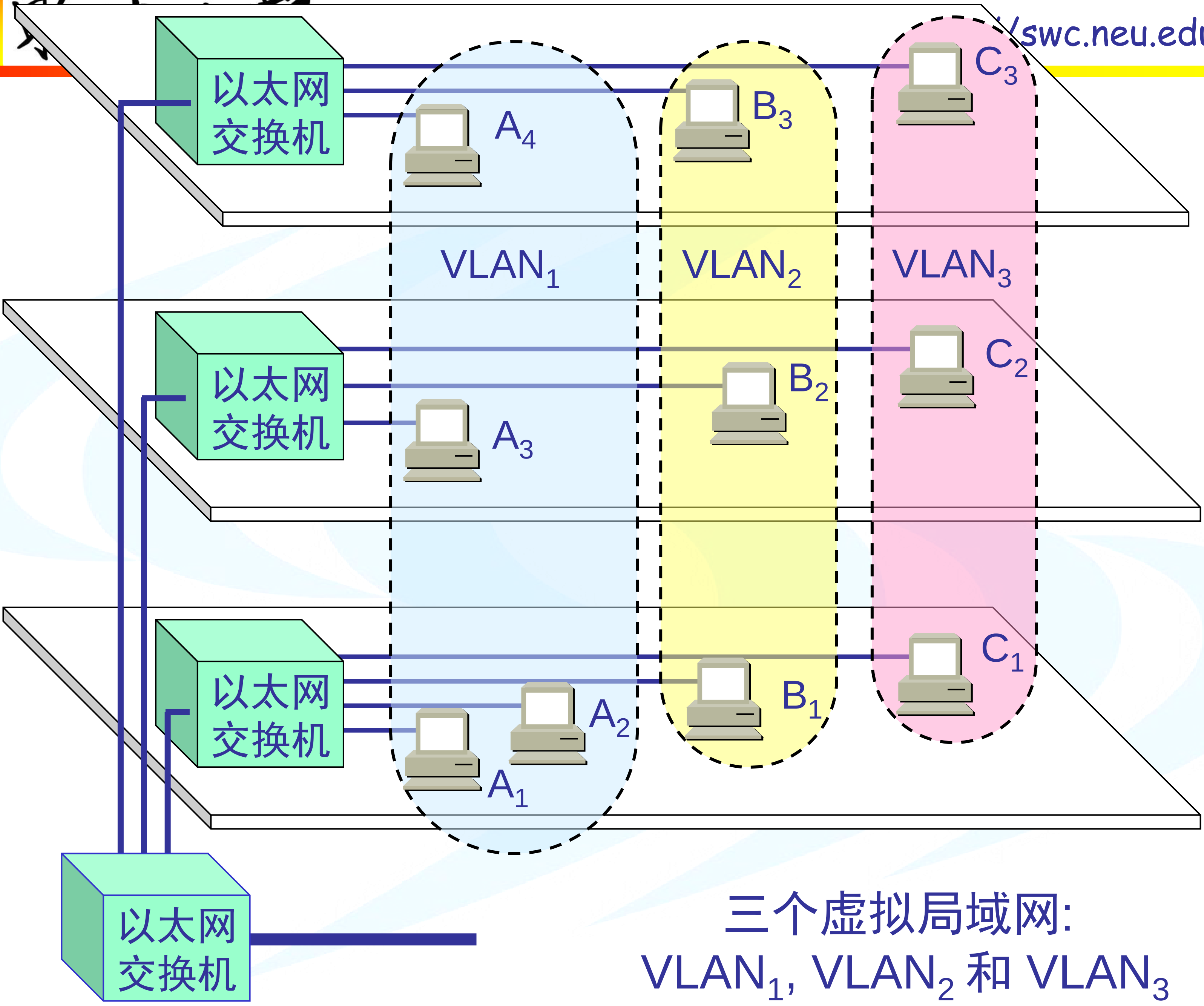
- ***trunk port:*** **carries frames between VLANS defined over multiple physical switches**
  - **frames forwarded within VLAN between switches can’t be vanilla 802.1 frames (must carry VLAN ID info)**
  - **802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports**

802.1Q VLAN frame format

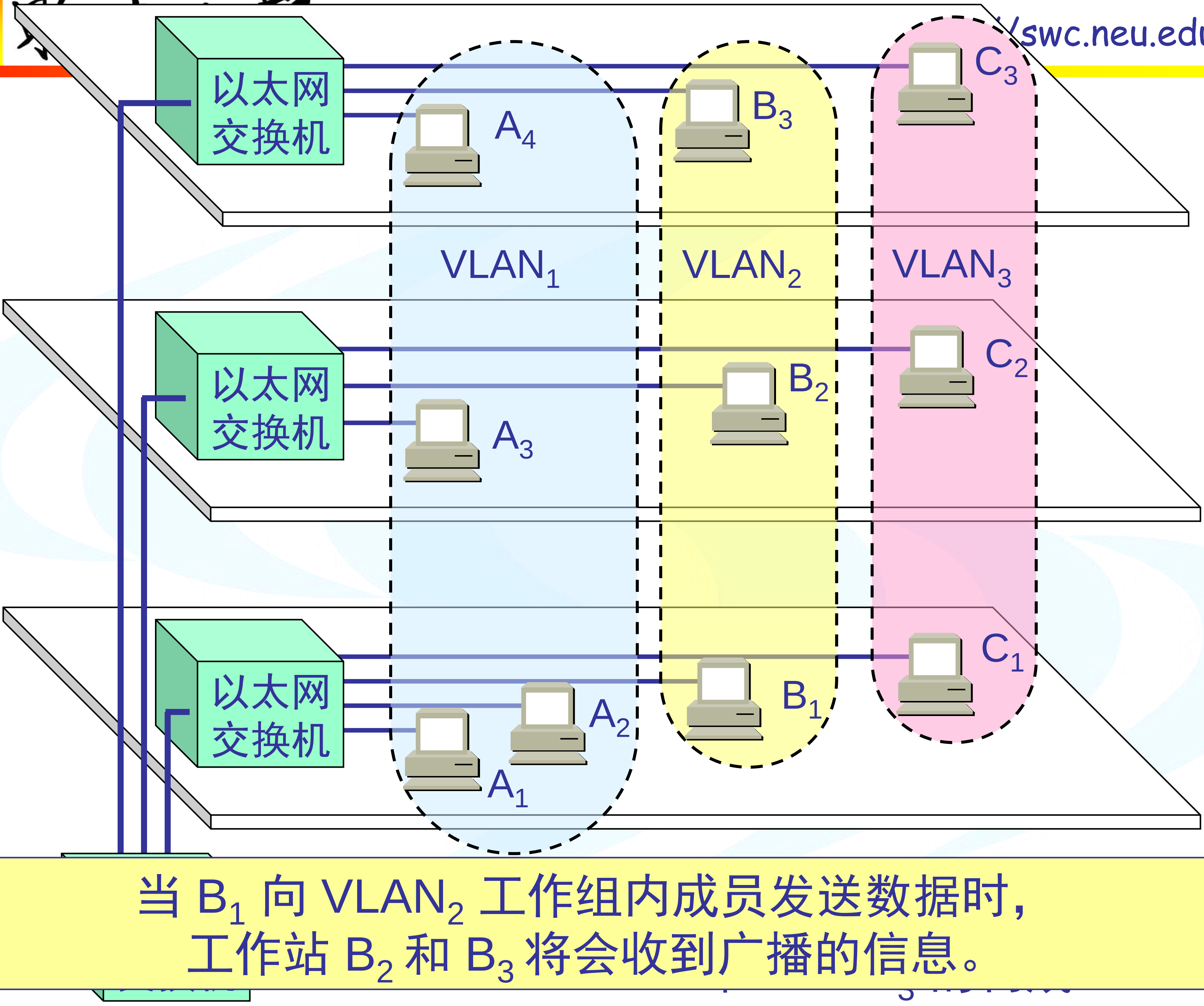


利用以太网交换机可以很方便地实现虚拟局域网

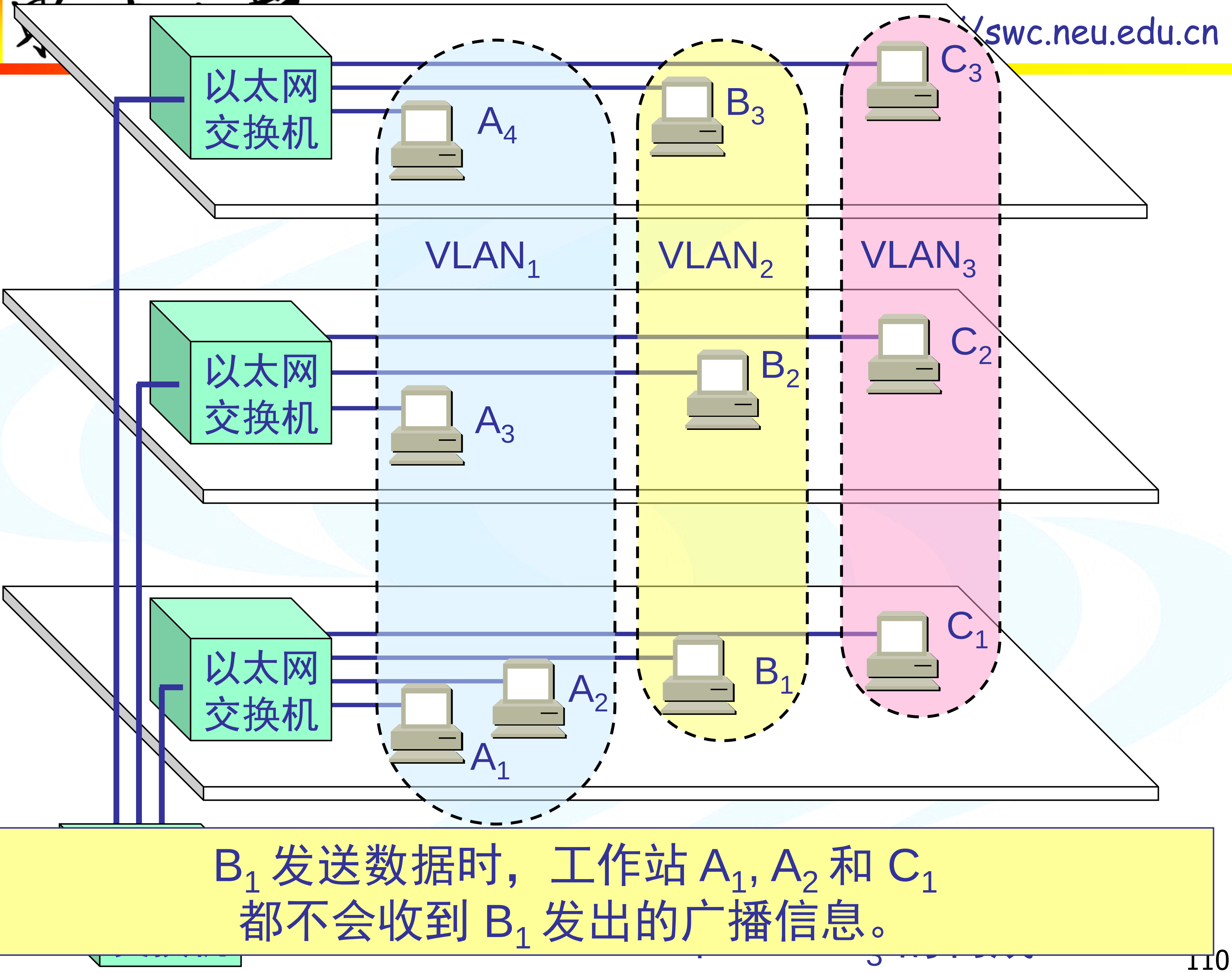
- 虚拟局域网 **VLAN** 是由一些局域网网段构成的与物理位置无关的逻辑组。
  - **这些网段具有某些共同的需求。**
  - **每一个 VLAN 的帧都有一个明确的标识符，指明发送这个帧的工作站是属于哪一个 VLAN。**
- 虚拟局域网其实只是局域网给用户提供的一种服务，而并不是一种新型局域网。



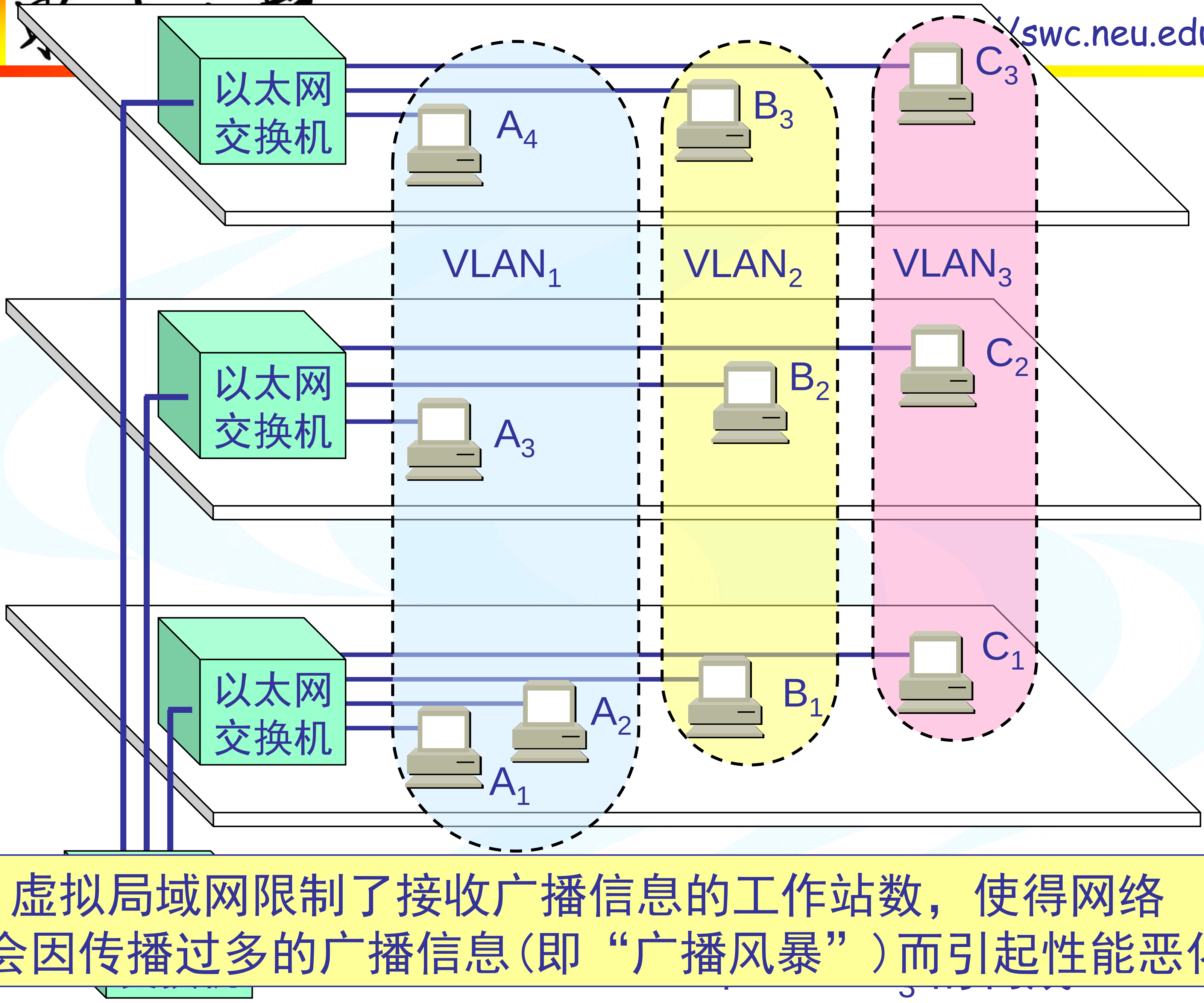
三个虚拟局域网:
$VLAN_1$, $VLAN_2$ 和 $VLAN_3$



当 B_1 向 $VLAN_2$ 工作组内成员发送数据时，
工作站 B_2 和 B_3 将会收到广播的信息。



B_1 发送数据时，工作站 A_1, A_2 和 C_1 都不会收到 B_1 发出的广播信息。



虚拟局域网限制了接收广播信息的工作站数，使得网络不会因传播过多的广播信息(即“广播风暴”)而引起性能恶化。

高速以太网
100BASE-T 以太网

- 速率达到或超过 **100 Mb/s** 的以太网称为高速以太网。
- 在双绞线上传送 **100 Mb/s** 基带信号的星型拓扑以太网，仍使用 **IEEE 802.3** 的**CSMA/CD** 协议。**100BASE-T** 以太网又称为快速以太网**(Fast Ethernet)**。

100BASE-T 以太网的特点

- 可在全双工方式下工作而无冲突发生。因此，不使用 **CSMA/CD** 协议。
- **MAC** 帧格式仍然是 **802.3** 标准规定的。
- 帧间时间间隔从原来的 **9.6 μs** 改为现在的 **0.96 μs**。

三种不同的物理层标准

- **100BASE-TX**
  - **使用 2 对 UTP 5 类线或屏蔽双绞线 STP。**
- **100BASE-FX**
  - **使用 2 对光纤。**
- **100BASE-T4**
  - **使用 4 对 UTP 3 类线或 5 类线。**

吉比特以太网

- 允许在 **1 Gb/s** 下全双工和半双工两种方式工作。
- 使用 **802.3** 协议规定的帧格式。
- 在半双工方式下使用 **CSMA/CD** 协议（全双工方式不需要使用 **CSMA/CD** 协议）。
- 与 **10BASE-T** 和 **100BASE-T** 技术向后兼容。

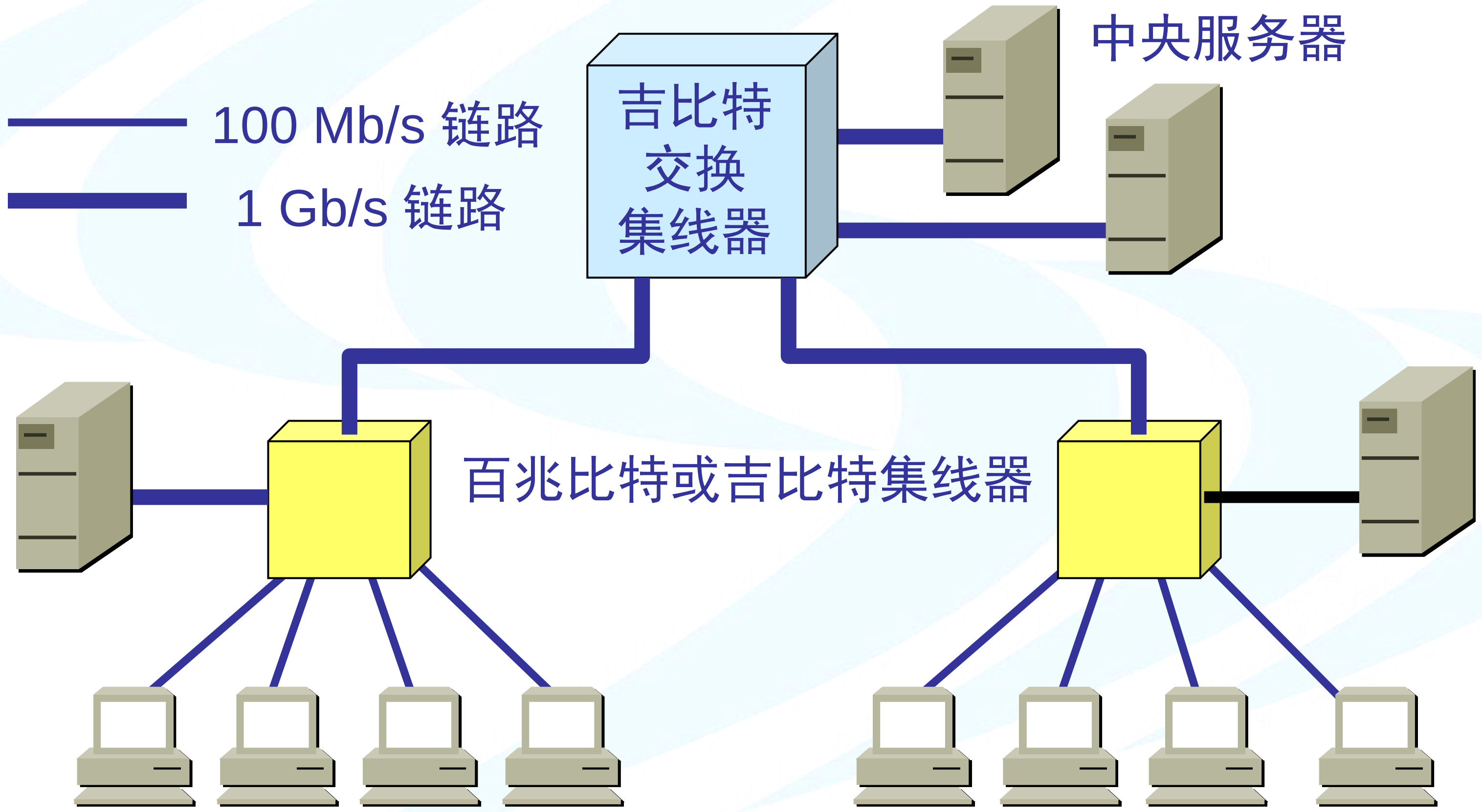
吉比特以太网的物理层

- **1000BASE-X　　基于光纤通道的物理层：**
  - **1000BASE-SX　SX表示短波长**
  - **1000BASE-LX　LX表示长波长**
  - **1000BASE-CX　CX表示铜线**
- **1000BASE-T**
  - **使用 4对 5 类线 UTP**

全双工方式

- 当吉比特以太网工作在全双工方式时（即通信双方可同时进行发送和接收数据），不使用载波延伸和分组突发。

吉比特以太网的配置举例



10 吉比特以太网

- **10** 吉比特以太网与 **10 Mb/s**，**100 Mb/s** 和 **1 Gb/s** 以太网的帧格式完全相同。
- **10** 吉比特以太网还保留了 **802.3** 标准规定的以太网最小和最大帧长，便于升级。
- **10** 吉比特以太网不再使用铜线而只使用光纤作为传输媒体。
- **10** 吉比特以太网只工作在全双工方式，因此没有争用问题，也不使用 **CSMA/CD** 协议。

吉比特以太网的物理层

- 局域网物理层 **LAN PHY**。局域网物理层的数据率是 **10.000 Gb/s**。
- 可选的广域网物理层 **WAN PHY**。广域网物理层具有另一种数据率，这是为了和所谓的 “**Gb/s**”的 **SONET/SDH**（即**OC-192/STM-64**）相连接。
  - 为了使 10 吉比特以太网的帧能够插入到 OC-192/STM-64 帧的有效载荷中，就要使用可选的广域网物理层，其数据率为 9.95328 Gb/s。

端到端的以太网传输

- **10 吉比特以太网的出现，以太网的工作范围已经从局域网（校园网、企业网）扩大到城域网和广域网，从而实现了端到端的以太网传输。**
- **这种工作方式的好处是：**
  - **成熟的技术**
  - **互操作性很好**
  - **在广域网中使用以太网时价格便宜。**
  - **统一的帧格式简化了操作和管理。**

以太网从 10 Mb/s 到 10 Gb/s 的演进

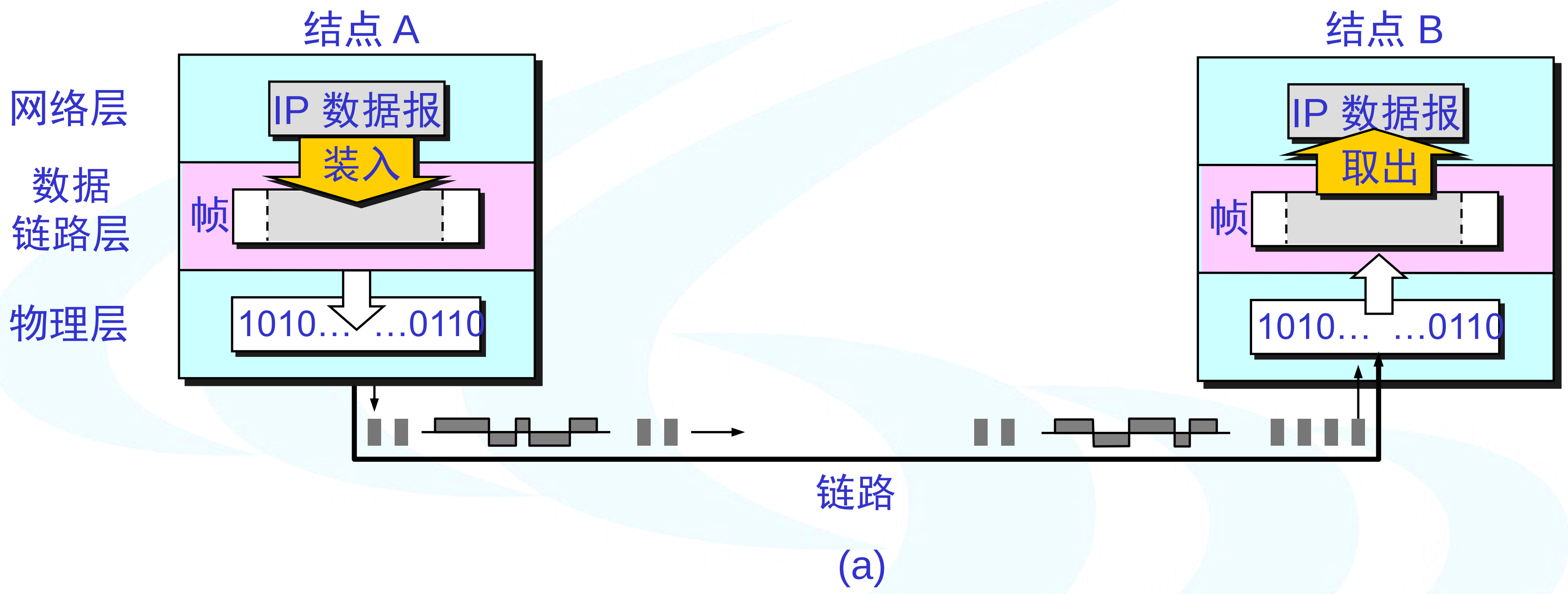
- 以太网从 **10 Mb/s** 到 **10 Gb/s** 的演进证明了以太网是：
- 可扩展的（从 **10 Mb/s** 到 **10 Gb/s**）。
- 灵活的（多种传输媒体、全/半双工、共享/交换）。
- 易于安装。
- 稳健性好。

2.7 PPP

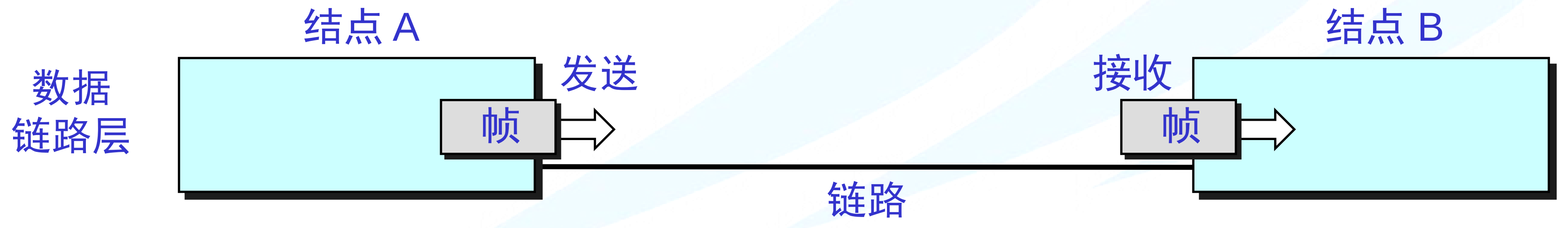
Point to Point Data Link Control

- one sender, one receiver, one link: easier than broadcast link:
  - no Media Access Control
  - no need for explicit MAC addressing
  - e.g., dialup link, ISDN line
- popular point-to-point DLC protocols:
  - PPP (point-to-point protocol)
  - HDLC: High level data link control

数据链路层传送的是帧



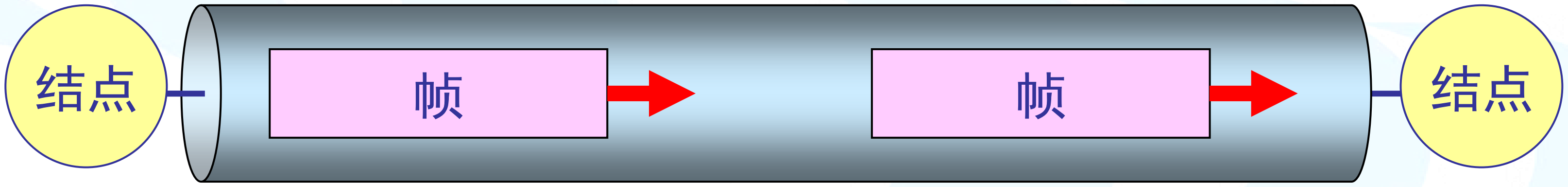
(a)



(b)

数据链路层像个数字管道

- 常常在两个对等的数据链路层之间画出一个数字管道，而在这条数字管道上传输的数据单位是帧。



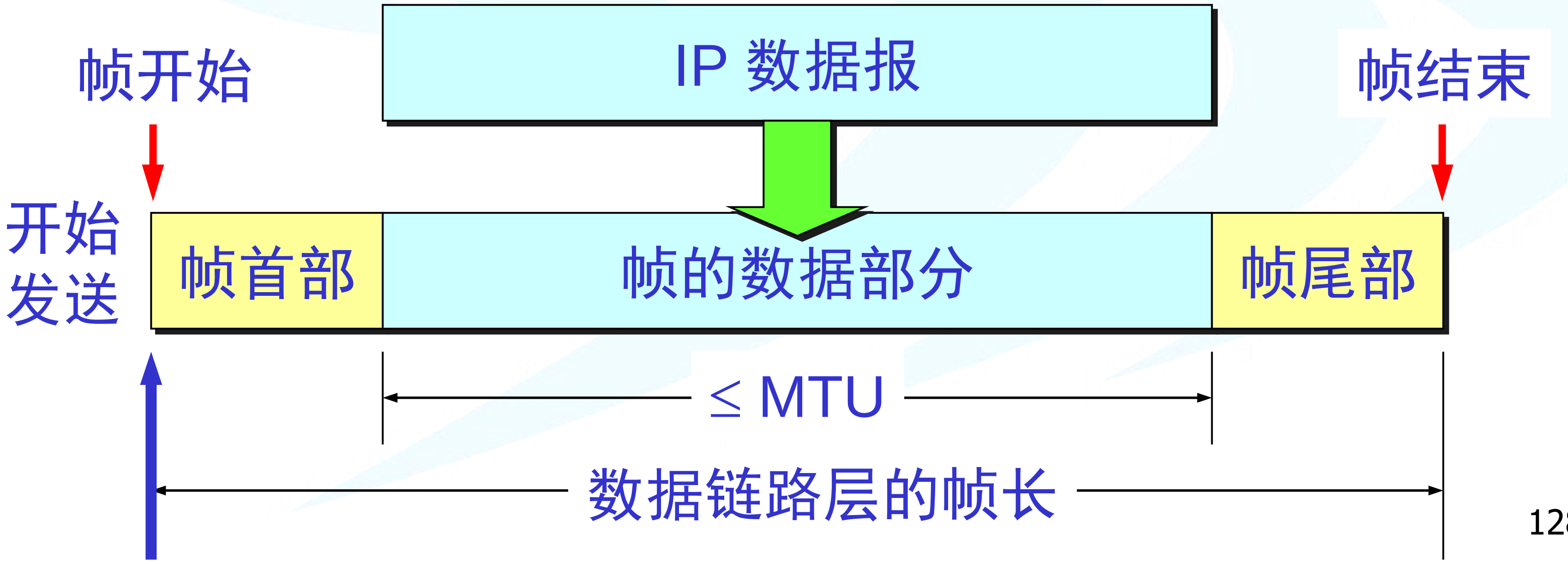
- 早期的数据通信协议曾叫作通信规程(procedure)。因此在数据链路层，规程和协议是同义语。

数据链路层：三个基本问题

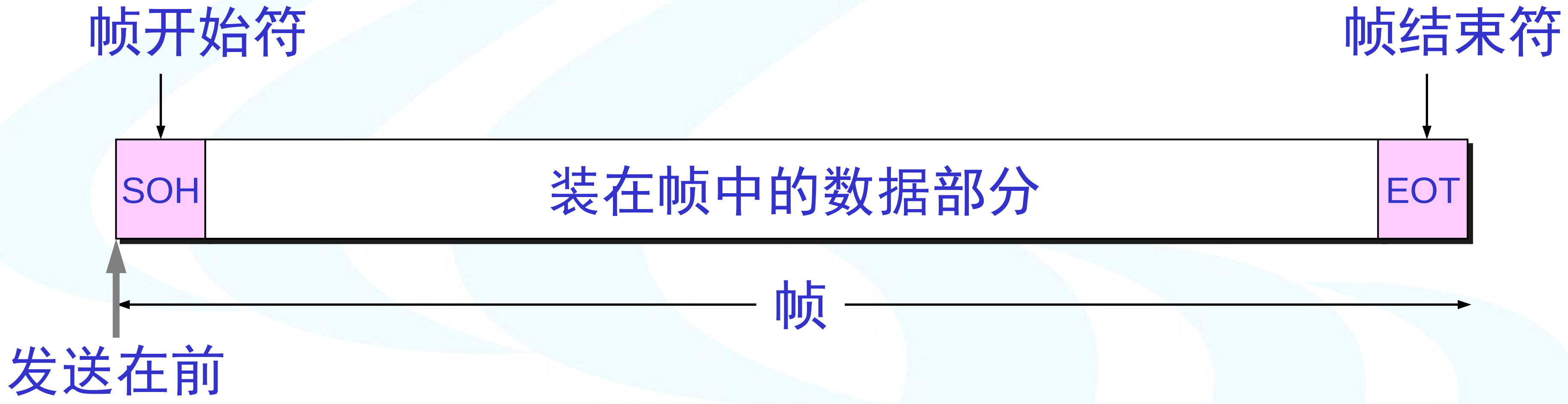
(1) 封装成帧
(2) 透明传输
(3) 差错控制

1. 封装成帧

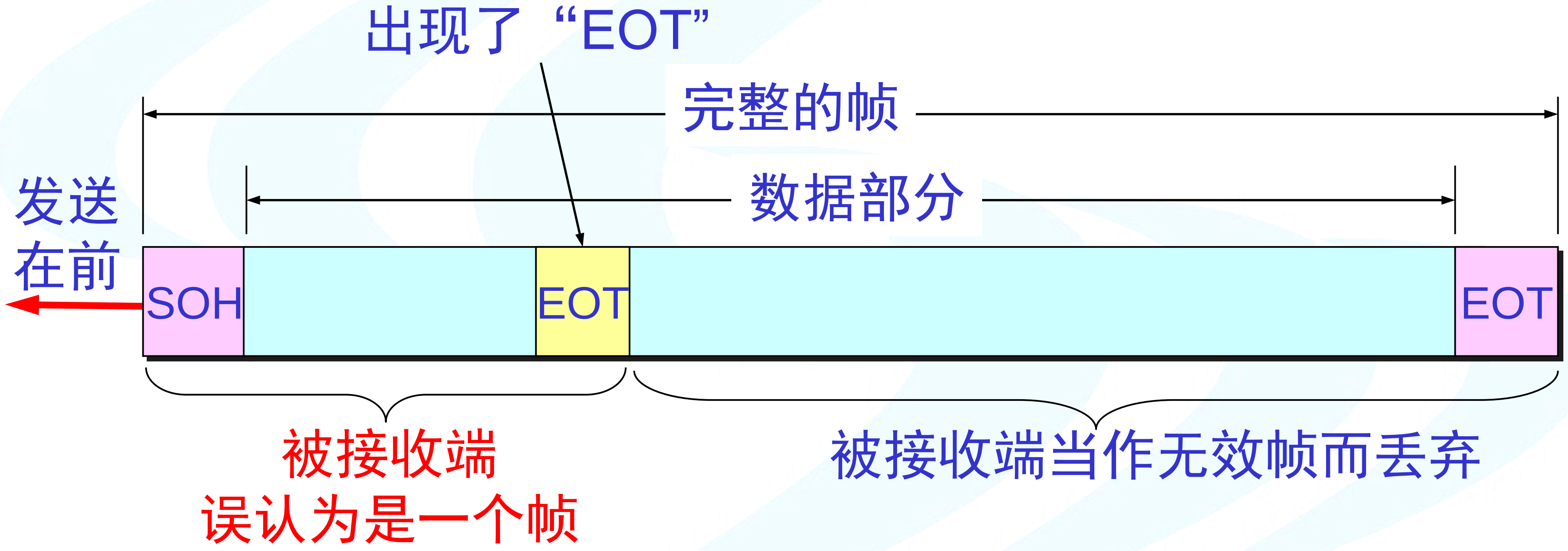
- 封装成帧(**framing**)就是在一段数据的前后分别添加首部和尾部，然后就构成了一个帧。确定帧的界限。
- 首部和尾部的一个重要作用就是进行帧定界。



用控制字符进行帧定界的方法举例



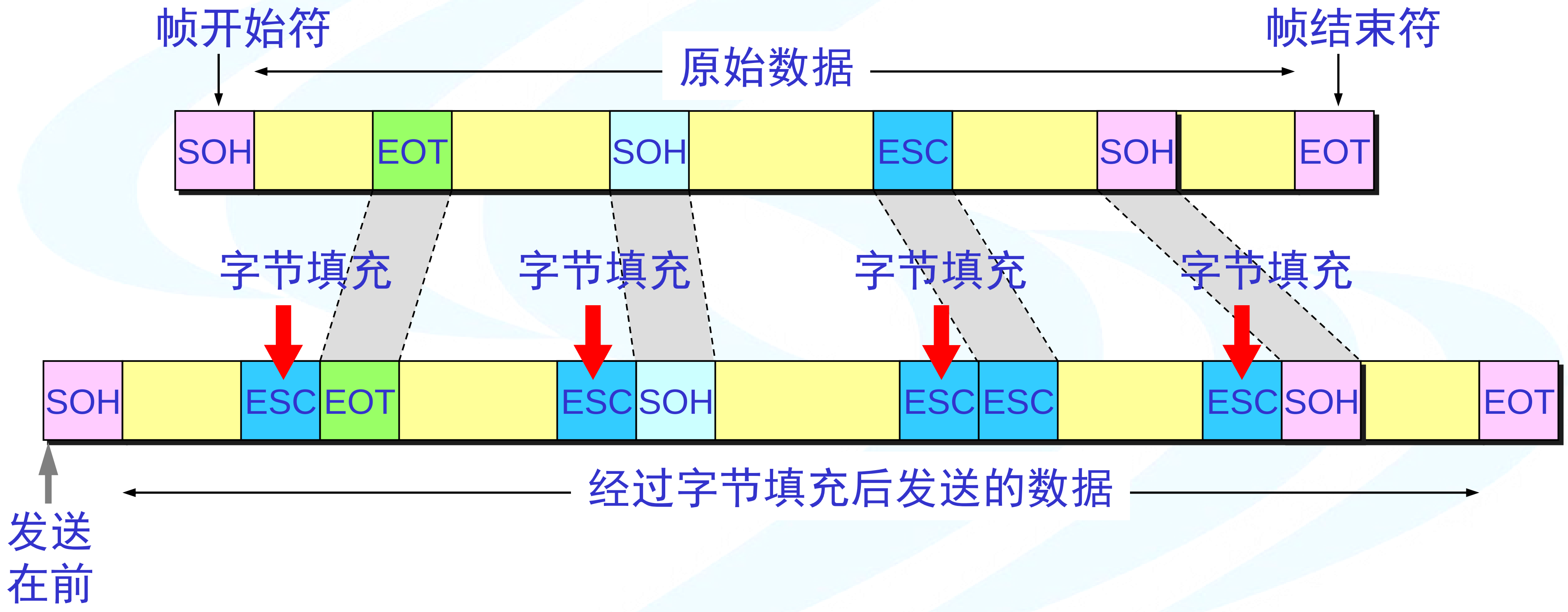
2. 透明传输


出现了“EOT”
完整的帧
数据部分
发送在前
SOH
EOT
EOT
被接收端误认为是一个帧
被接收端当作无效帧而丢弃

解决透明传输问题

- 发送端的数据链路层在数据中出现控制字符 “SOH” ”(其十六进制编码是 01)或 “EOT” (其十六进制编码是 04)的前面插入一个转义字符 “ESC”(其十六进制编码是 1B)。
- 字节填充(byte stuffing)或字符填充(character stuffing)——接收端的数据链路层在将数据送往网络层之前删除插入的转义字符。
- 如果转义字符也出现数据当中，那么应在转义字符前面插入一个转义字符。当接收端收到连续的两个转义字符时，就删除其中前面的一个。

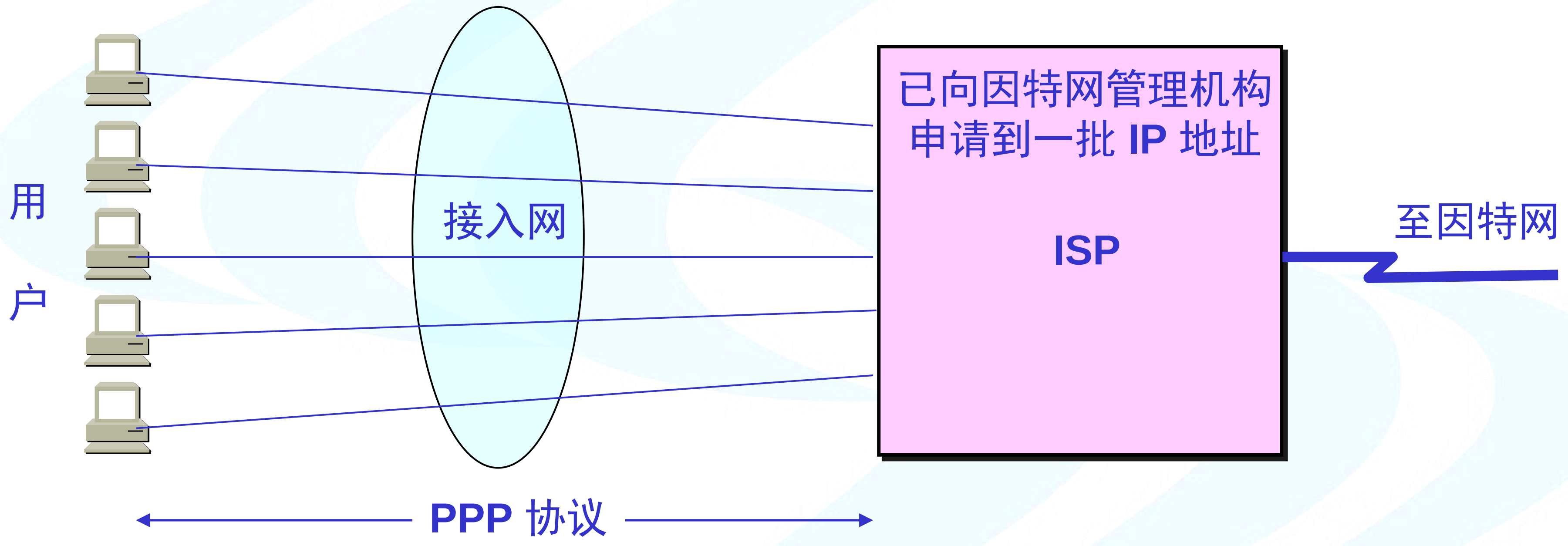
用字节填充法解决透明传输的问题



3. 差错检测

- 在传输过程中可能会产生比特差错：**1** 可能会变成 **0** 而 **0** 也可能变成 **1**。
- 在一段时间内，传输错误的比特占所传输比特总数的比率称为误码率 **BER (Bit Error Rate)**。
- 误码率与信噪比有很大的关系。
- 为了保证数据传输的可靠性，在计算机网络传输数据时，必须采用各种差错检测措施。

用户到 ISP 的链路使用 PPP 协议



PPP 协议的组成

- **1992 年制订了 PPP 协议。经过 1993 年和 1994 年的修订，现在的 PPP 协议已成为因特网的正式标准[RFC 1661]。**
- **PPP 协议有三个组成部分**
  - **一个将 IP 数据报封装到串行链路的方法。**
  - **链路控制协议 LCP (Link Control Protocol)。**
  - **网络控制协议 NCP (Network Control Protocol)。**

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!

标识帧开始 1个字节7E

该字段目前无作用

传输的数据

标识帧结束 1个字节7E

标志域	地址域	控制域	协议域	信息域	校验和	标志域
01111110	11111111	11111100	协议	信息	校验和	01111110

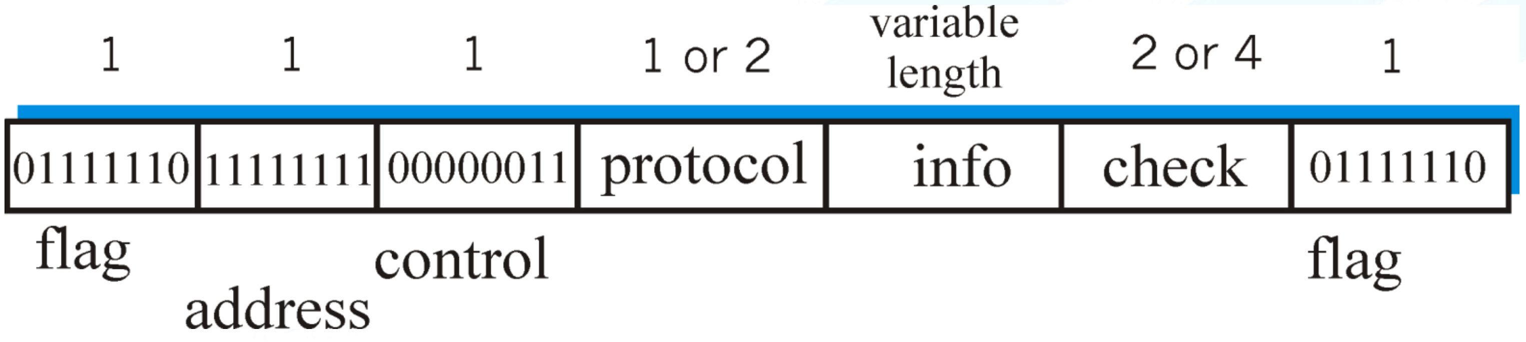
标识目的地址，不起作用

上层协议，1/2字节：LCP-C021， IPCP-8021，IP-0021, DECnet-027 , AppleTalk-0029

差错检查：16/32-CRC，

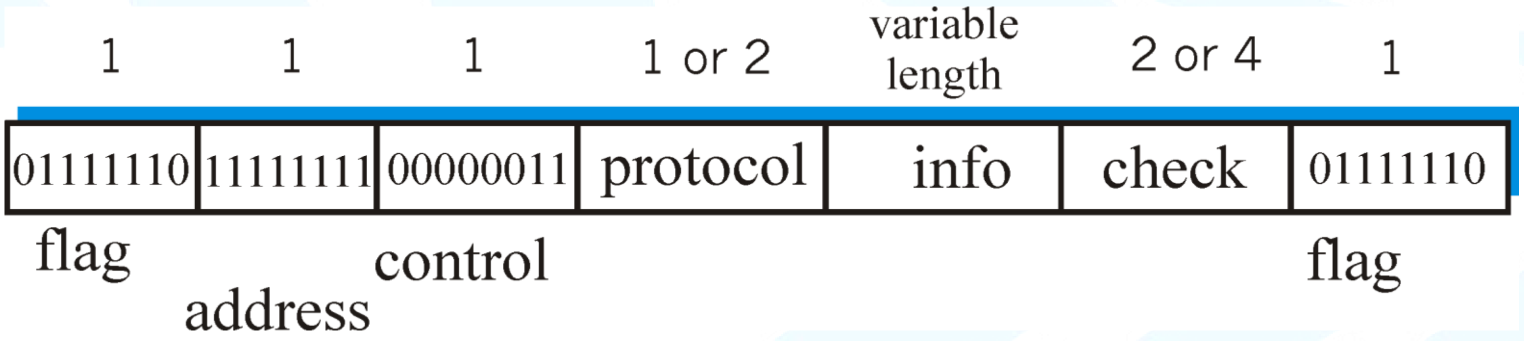
PPP Data Frame

- 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)



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

字节填充举例

标志域	地址域	控制域	协议域	信息域	校验和	标志域
01111110	11111111	11111100	协议	信息	校验和	01111110

01111110	11111111	11111100	00100001	01111110	校验和	01111110

发送前

字节填充

01111110	11111111	11111100	00100001	01111101 01111110	校验和	01111110

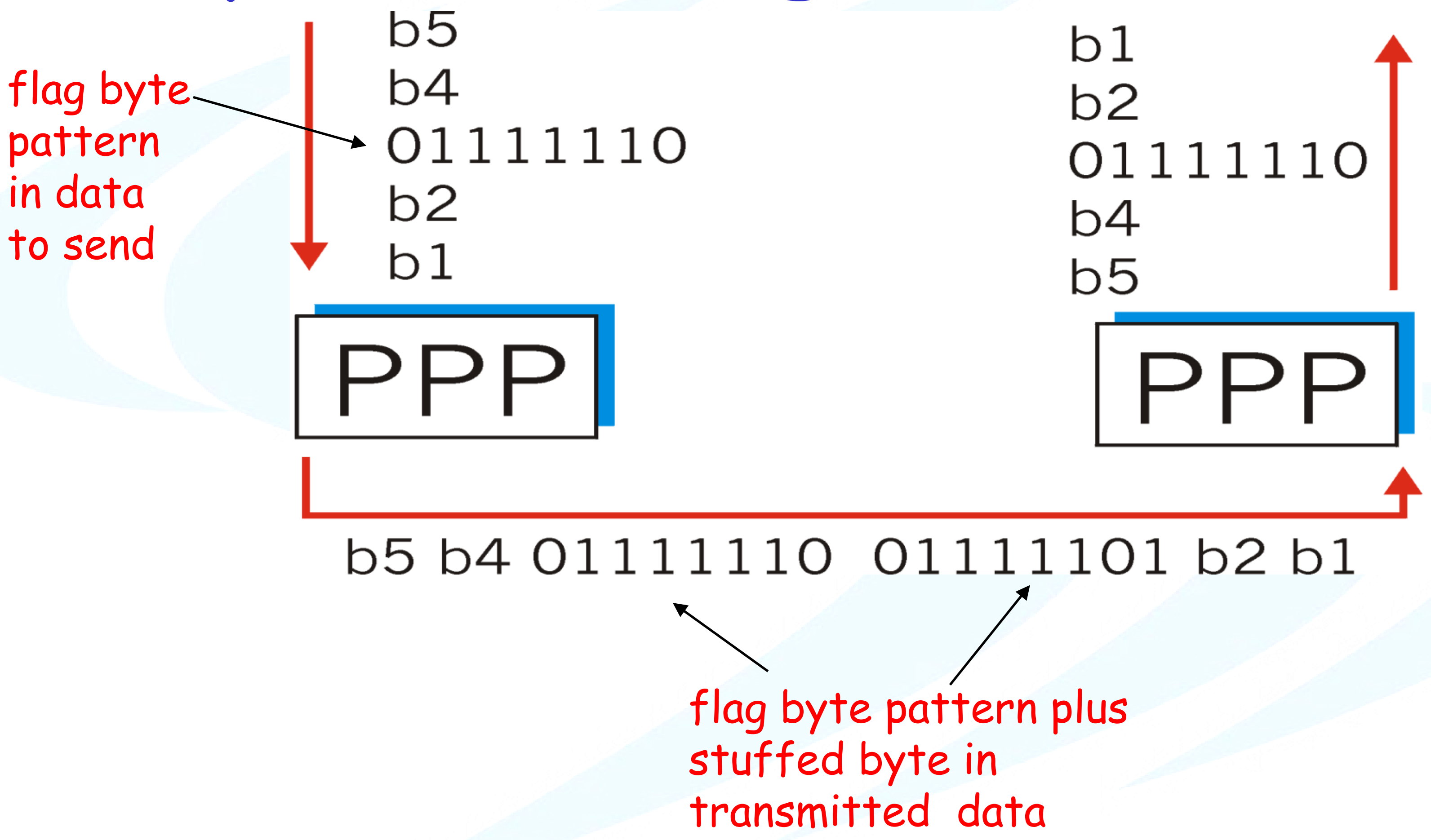
发送数据

01111110	11111111	11111100	00100001	01111101 01111110	校验和	01111110

接收数据处理

01111110	11111111	11111100	00100001	01111110	校验和	01111110

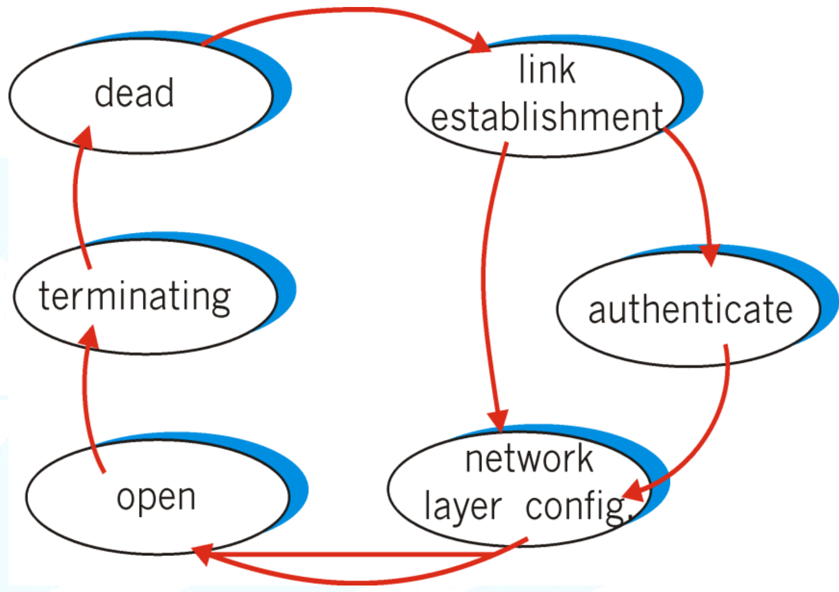
Byte Stuffing


flag byte pattern in data to send
b5
b4
01111110
b2
b1
PPP
b1
b2
01111110
b4
b5
PPP
b5 b4 01111110 01111101 b2 b1
flag byte pattern plus stuffed byte in transmitted data

PPP Data Control Protocol

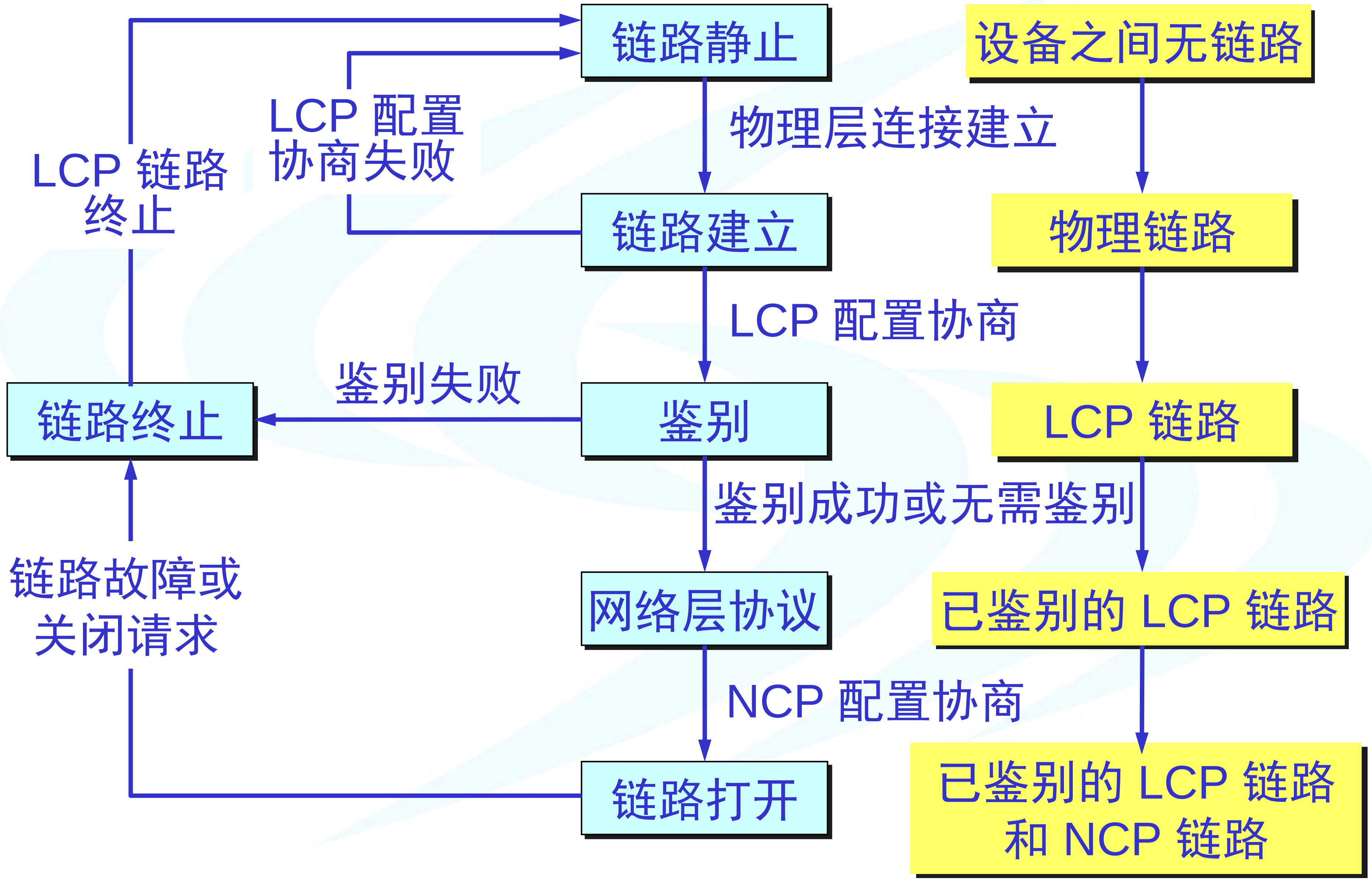
Before exchanging network-layer data, data link peers must

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



PPP 协议的工作状态

- 当用户拨号接入 **ISP** 时，路由器的调制解调器对拨号做出确认，并建立一条物理连接。
- **PC** 机向路由器发送一系列的 **LCP** 分组（封装成多个 **PPP** 帧）。
- 这些分组及其响应选择一些 **PPP** 参数，和进行网络层配置，**NCP** 给新接入的 **PC**机分配一个临时的 **IP** 地址，使 **PC** 机成为因特网上的一个主机。
- 通信完毕时，**NCP** 释放网络层连接，收回原来分配出去的 **IP** 地址。接着，**LCP** 释放数据链路层连接。最后释放的是物理层的连接。


链路静止
物理层连接建立
链路建立
LCP 配置协商
鉴别
鉴别成功或无需鉴别
网络层协议
NCP 配置协商
链路打开
LCP 配置协商失败
LCP 链路终止
鉴别失败
链路终止
链路故障或关闭请求
设备之间无链路
物理链路
LCP 链路
已鉴别的 LCP 链路
已鉴别的 LCP 链路和 NCP 链路