Computer Networking

谢 逸

中山大学·计算机学院

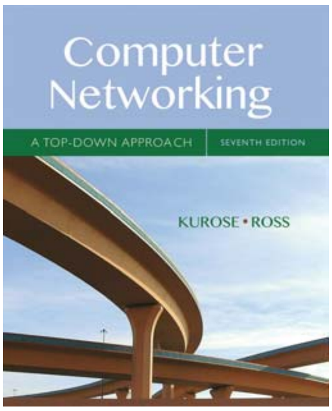
2023. Fall

Chapter 6 The Link Layer and LANs

A note on the use of these Powerpoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and 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) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides 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

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Homework

- Ch6 (ver7), 6, 11, 13, 14, 15, 17, 18, 21, 23, 29, 31,32
- Keywords: VLAN, CSMA/CD/CA, CRC, TDM, CDM, ALOHA, ARP, self-learning, ATM, MPLS, PPP, Ethernet, MAC protocols

Chapter 6: Link layer

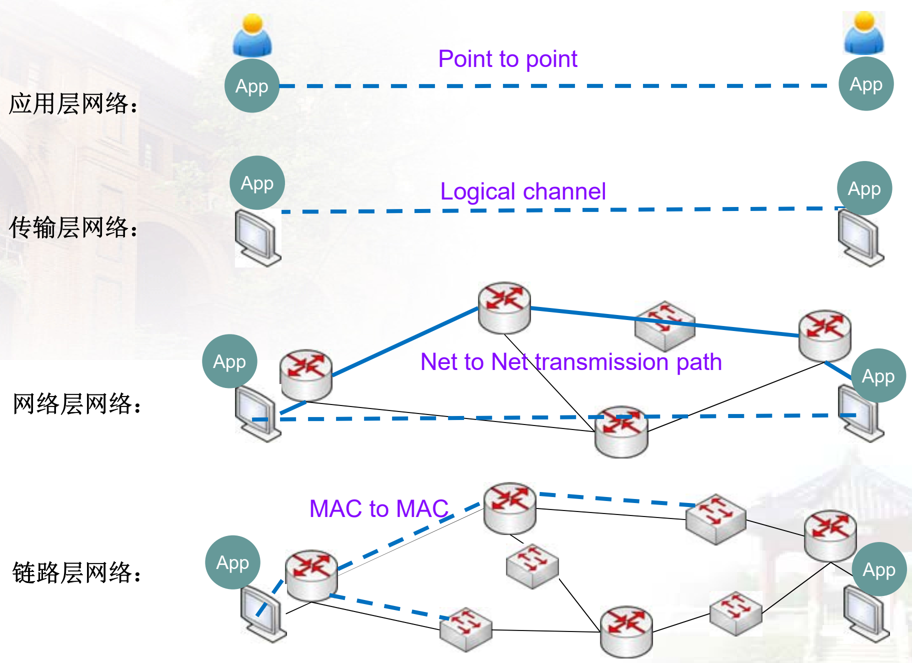
our goals:

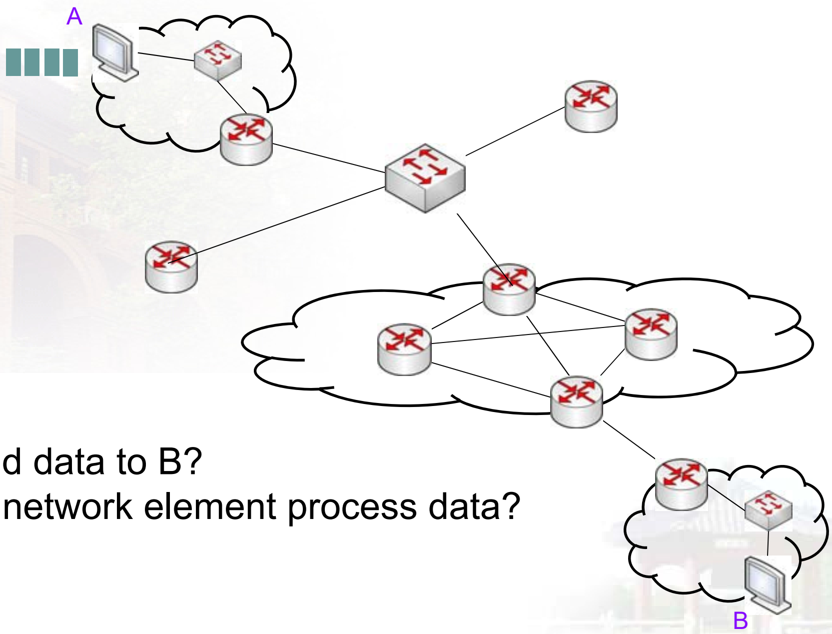
- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- instantiation, implementation of various link layer technologies

Link layer, LANs: outline

6.1 introduction, services
6.2 error detection, correction
6.3 multiple access protocols
6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS
6.6 data center networking
6.7 a day in the life of a web request

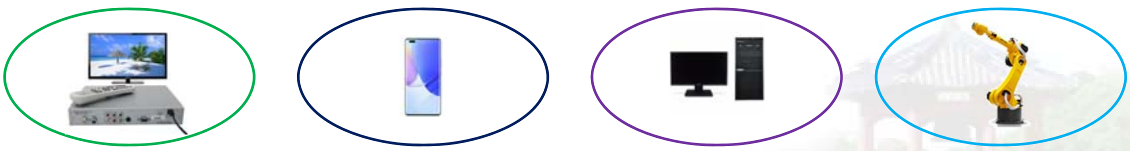




How does A send data to B?
How does each network element process data?

Link layer: introduction

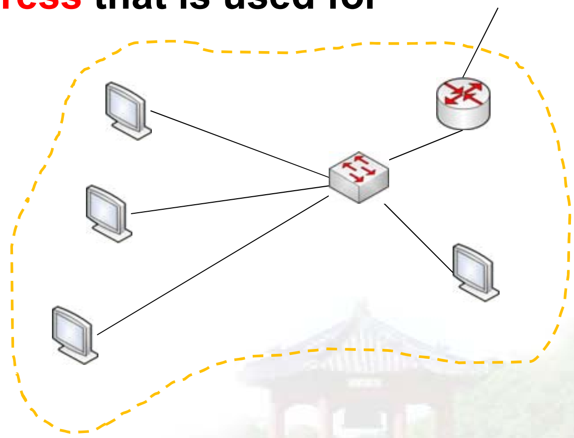
- Each network interface card has an address named physical address
  - TV-net, telephone-net, data-net, …
  - Physical address is different from IP address
  - Physical addresses do NOT have a uniform format and standard. It may be 12bits, 48bits, or others.



Link layer: introduction

- In a same subnet, the nodes are considered to have a uniform standard physical address that is used for addressing.

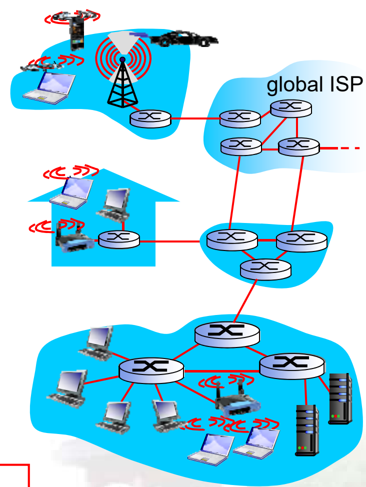
MAC address is used for addressing in the same subnet



Link layer: introduction

terminology:

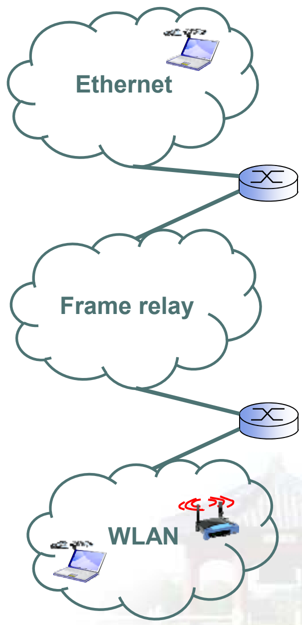
- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs
- layer-2 packet: frame, encapsulates datagram



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

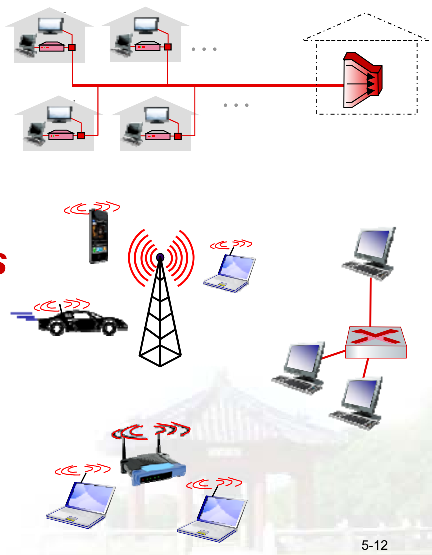
Link layer: context

- 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



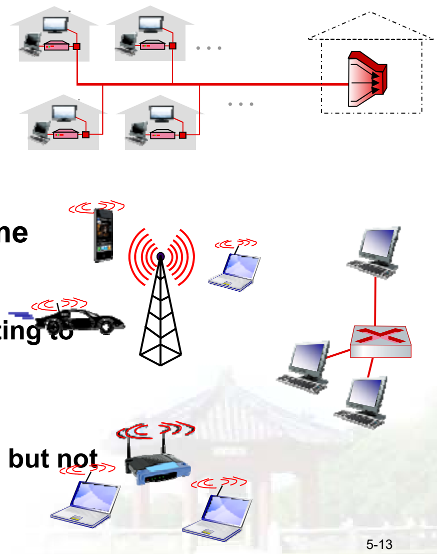
Link layer services

- *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?



Link layer services (more)

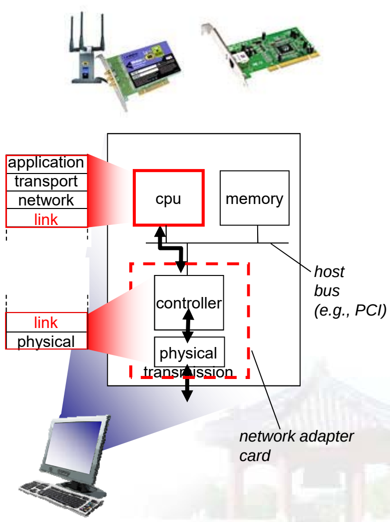
- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit 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



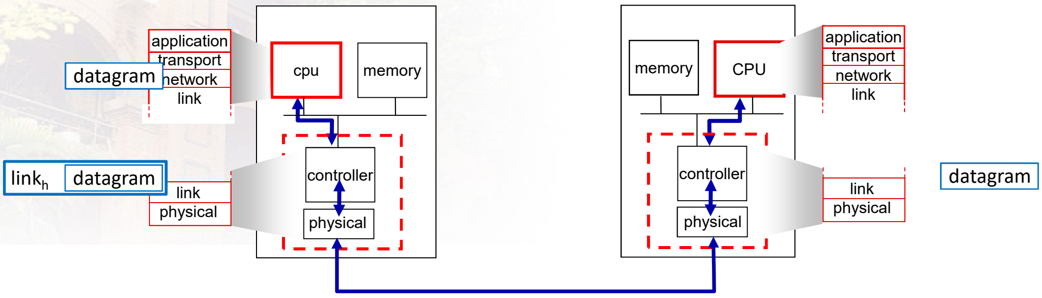
Why no congestion control?

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

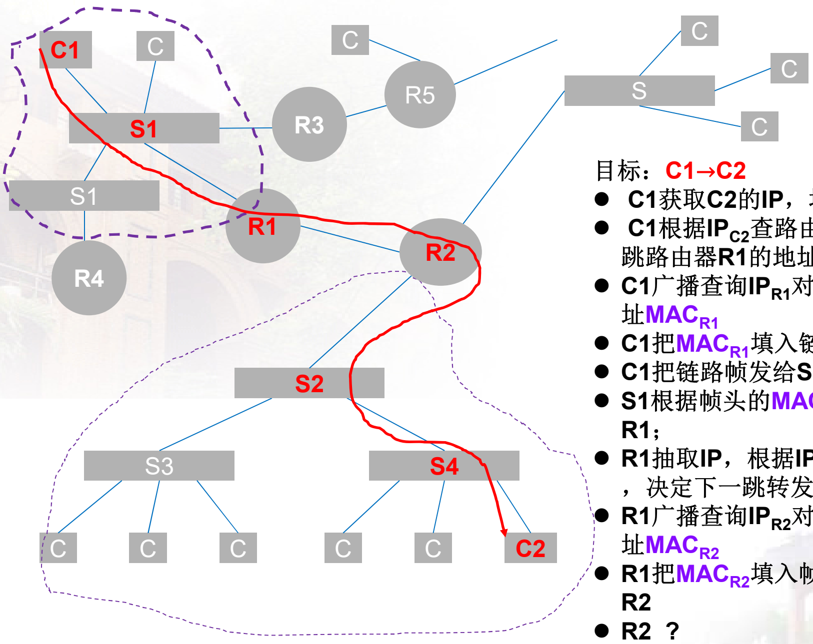


Interfaces communicating



sending side:
- encapsulates datagram in frame
- adds error checking bits, reliable data transfer, flow control, etc.

receiving side:
- looks for errors, reliable data transfer, flow control, etc.
- extracts datagram, passes to upper layer at receiving side



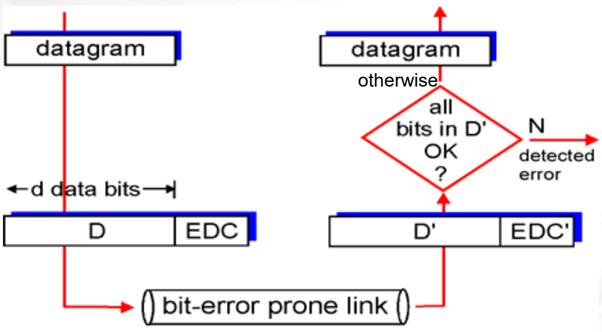
目标：**C1→C2**
- **C1获取C2的IP，填入IP报头**
- **C1根据IP_{C2}查路由表得到下一跳路由器R1的地址IP_{R1}**
- **C1广播查询IP_{R1}对应的物理地址MAC_{R1}**
- **C1把MAC_{R1}填入链路帧头；**
- **C1把链路帧发给S1;**
- **S1根据帧头的MAC_{R1}转发给R1；**
- **R1抽取IP，根据IP_{C2}查路由表，决定下一跳转发给IP_{R2};**
- **R1广播查询IP_{R2}对应的物理地址MAC_{R2}**
- **R1把MAC_{R2}填入帧头，转发给R2**
- **R2 ？**

Link layer, LANs: outline

Error detection

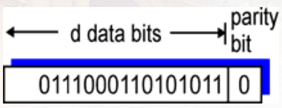
EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

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

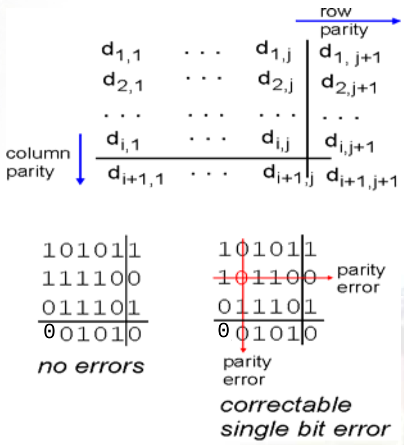


Parity checking

single bit parity:
- detect single bit errors



two-dimensional bit parity:
- detect and correct single bit errors



Internet checksum (review)

goal: detect “errors” (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

sender:
- **treat segment contents as sequence of 16-bit integers**
- **checksum: addition (1’s complement sum) of segment contents**
- **sender puts checksum value into 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?***

Cyclic redundancy check

- more powerful error-detection coding
- 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 (Ethernet, 802.11 WiFi, ATM)

R=f(D)

← d bits → ← r bits →

D data bits to be sent	R CRC bits

bit pattern

$D * 2^r$ XOR R

mathematical formula

Cyclic redundancy check

- **generator**

2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	1	1

$$\Rightarrow x^6 \cdot 1 + x^5 \cdot 0 + x^4 \cdot 1 + x^3 \cdot 0 + x^2 \cdot 1 + x^1 \cdot 1 + x^0 \cdot 1$$
$$= x^6 + x^4 + x^2 + x + 1$$
$$= G(x)$$

$$G(x) = x^4 + x^3 + x + 1$$
$$\Rightarrow ?$$

CRC example

want:

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

equivalently:

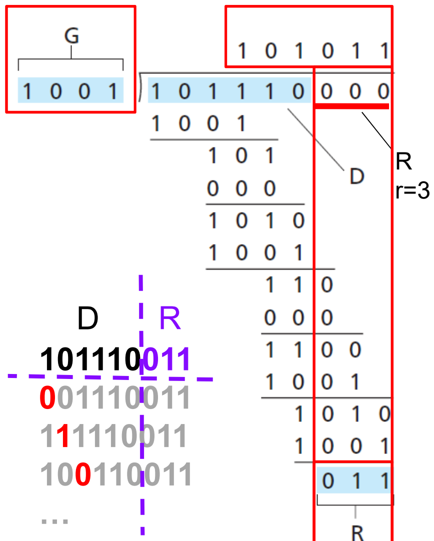
$D \cdot 2^r = nG$ XOR R

equivalently:

if we divide $D \cdot 2^r$ by G, want remainder R to satisfy:

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

Sender:



余数的位数一定要是比除数位数只能少一位，哪怕前面位是0，甚至是全为0（整除时）也都不能省略

Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

5.5 link virtualization: MPLS

5.6 data center networking

5.7 a day in the life of a web request

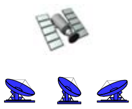
Multiple access links, protocols

two types of "links":

- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

Multiple access protocols

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

multiple access protocol

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

An ideal multiple access protocol

given: broadcast channel of rate R bps

desiderata:

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

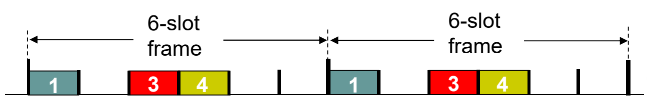
MAC protocols: taxonomy

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

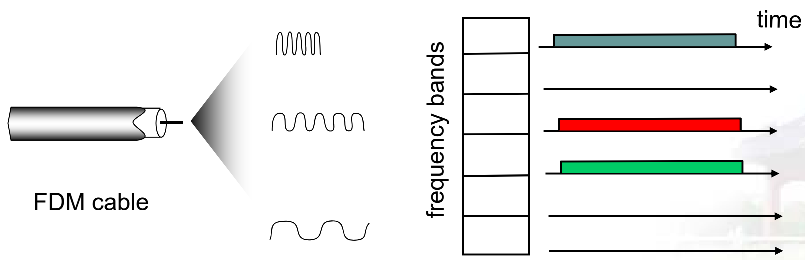
Channel partitioning MAC protocols: TDMA

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

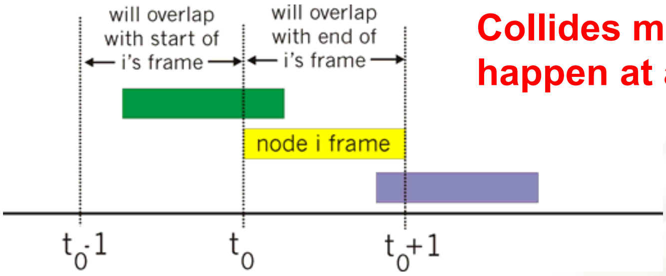


Random access protocols

- 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:
  - ALOHA
  - slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA

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 $[t_0-1, t_0+1]$



Collides may happen at any time

Pure ALOHA efficiency

Pr[success by given node] = Pr[node transmits] $\cdot$
Pr[no other node transmits in $[t_0-1,t_0]$] $\cdot$
Pr[no other node transmits in $[t_0,t_0+1]$]

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

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

Pr[a successful slots] = $Np(1-p)^{2N-1}$

… choosing optimum p and then letting $N \to \infty$

= 1/(2e) = .18

worse **than the following slotted Aloha!**

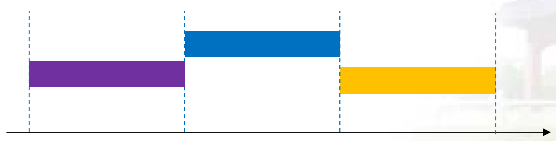
Slotted ALOHA

assumptions:

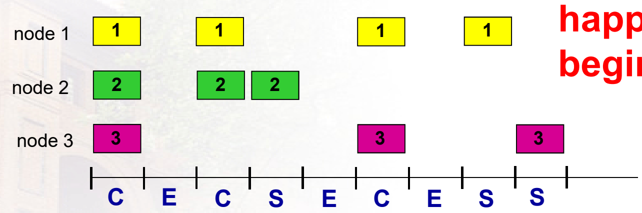
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit *only* slot **beginning**
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

operation:

- when node obtains fresh frame, 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



Slotted ALOHA



Collides only happen at the beginning

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, all with many frames to send)

- ***suppose: N*** **nodes with many frames to send, each transmits in slot with probability *p***
- **prob that given node has success in a slot = $p(1-p)^{N-1}$**
- **prob of a successful slots = $Np(1-p)^{N-1}$**
- **max efficiency: find p* that maximizes $Np(1-p)^{N-1}$**
- **for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:**

max efficiency = 1/e = .37

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

CSMA (carrier sense multiple access)

***CSMA*: listen before transmit:**

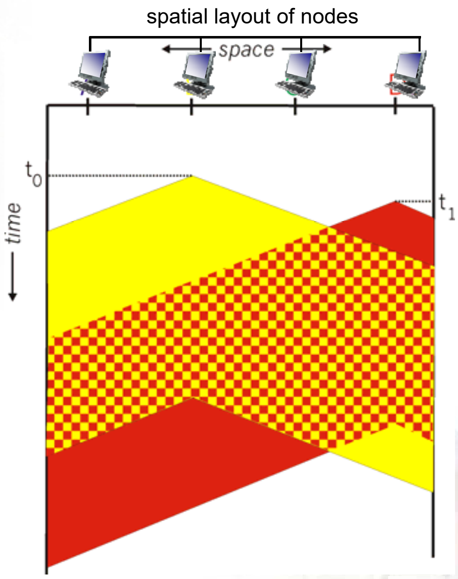
if channel sensed idle: transmit entire frame

- **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**
- **Nodes don't detect collision during transmission**
- **collision: entire packet transmission time wasted**
  - **distance & propagation delay play role in in determining collision probability**

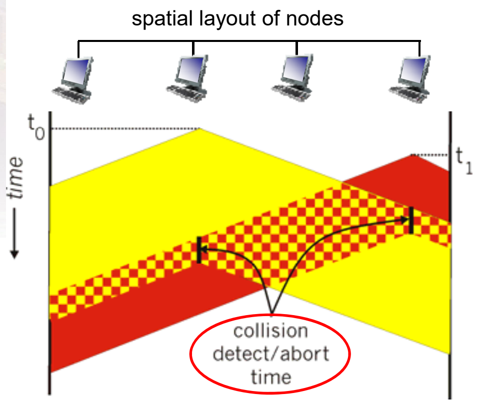


CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA
- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
- human analogy: the polite conversationalist

CSMA/CD (collision detection)



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

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

- efficiency goes to 1 (i.e., NO collision)
  - as t_{prop} goes to 0 → No time difference
  - as t_{trans} goes to infinity → Always occupying the channel
- better performance than ALOHA: and simple, cheap, decentralized!

"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!

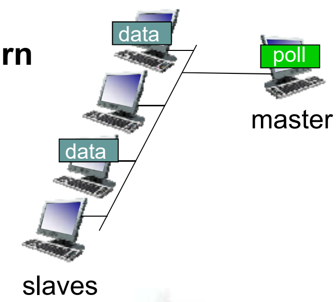
random access MAC protocols
- efficient at low load: single node can fully utilize channel
- high load: collision overhead

"taking turns" protocols

look for best of both worlds!

"Taking turns" MAC protocols

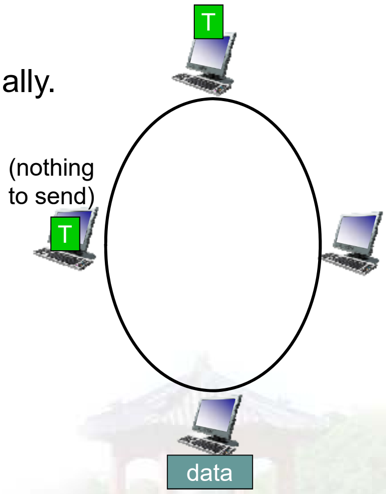
polling:
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



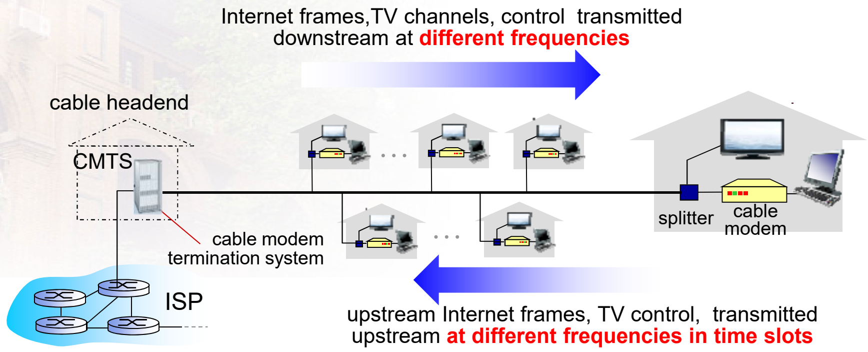
"Taking turns" MAC protocols

token passing:

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

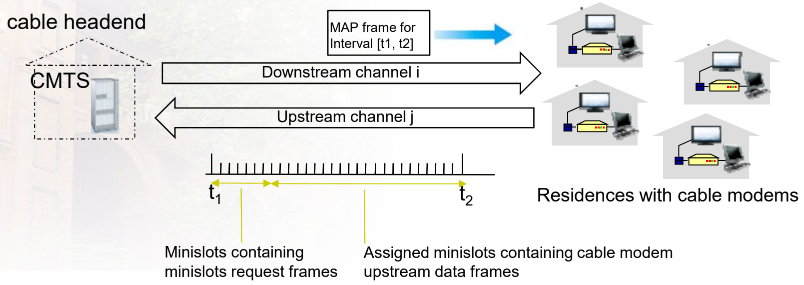


A Case: Cable access network



- 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)

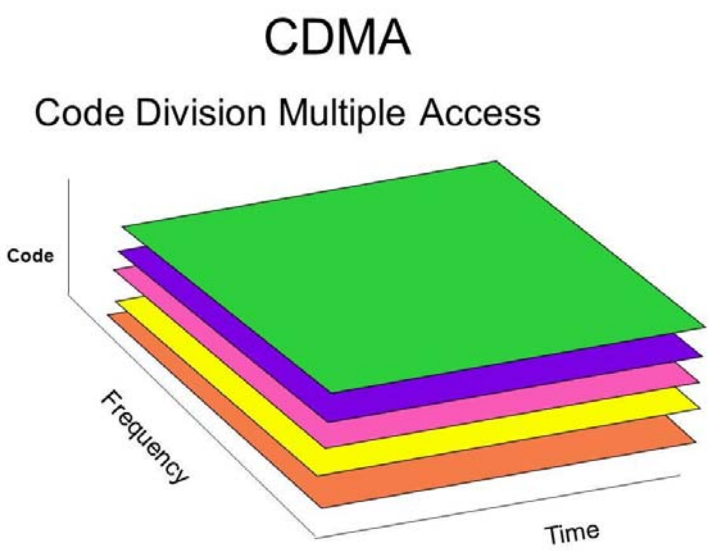
Cable access network



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

TDM+FDM+CDM



Summary of MAC protocols

2022.12.2

- *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

A question: why do we use CSMA/CD rather than TDMA/FDMA？

Link layer, LANs: outline

6.1 introduction, services
6.2 error detection, correction
6.3 multiple access protocols
6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS
6.6 data center networking
6.7 a day in the life of a web request

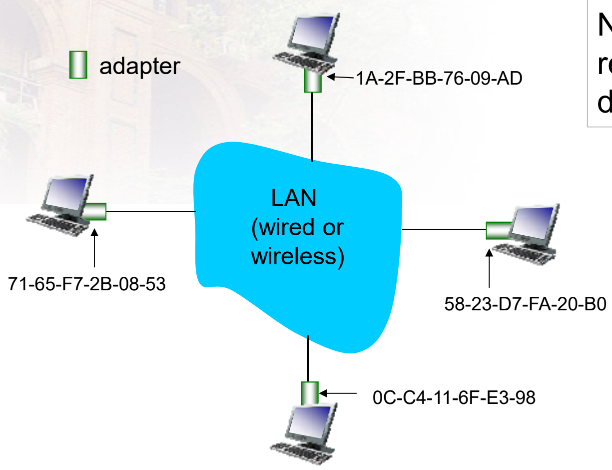
MAC addresses and ARP

After the packet enters the local network, it depends on its MAC address for addressing

- 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

hexadecimal (base 16) notation
(each "number" represents 4 bits)

LAN addresses and ARP



each adapter on LAN has unique *LAN* address

NIC decides the frame is received or not by its destination MAC address.

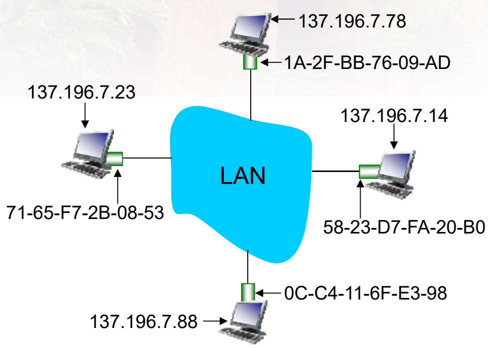
LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like identity card number (fixed)
  - IP address: like postal address (changeable)
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address depends on IP subnet to which node is attached

IP	MAC
逻辑地址	物理地址
全局有效	局部有效
层次化	平面型
网间寻址	网内寻址

ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has 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

- A wants to send datagram to B
  - 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 nodes 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*

ARP protocol in action

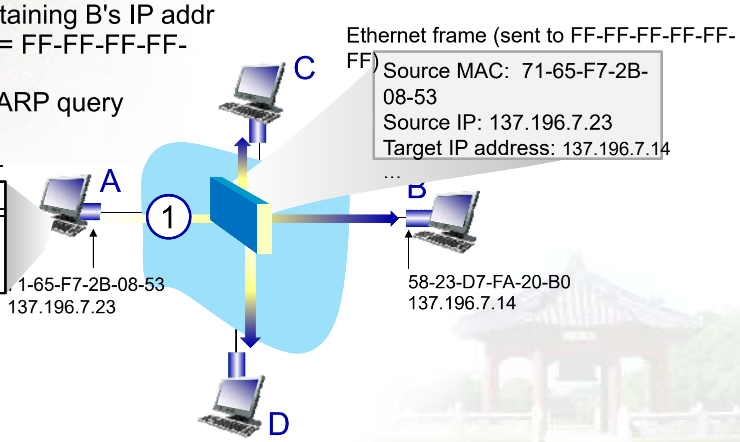
example: A wants to send datagram to B
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

A broadcasts ARP query, containing B's IP addr
1
- destination MAC address = FF-FF-FF-FF-FF-FF
- all nodes on LAN receive ARP query

ARP table in A

IP addr	MAC addr	TTL

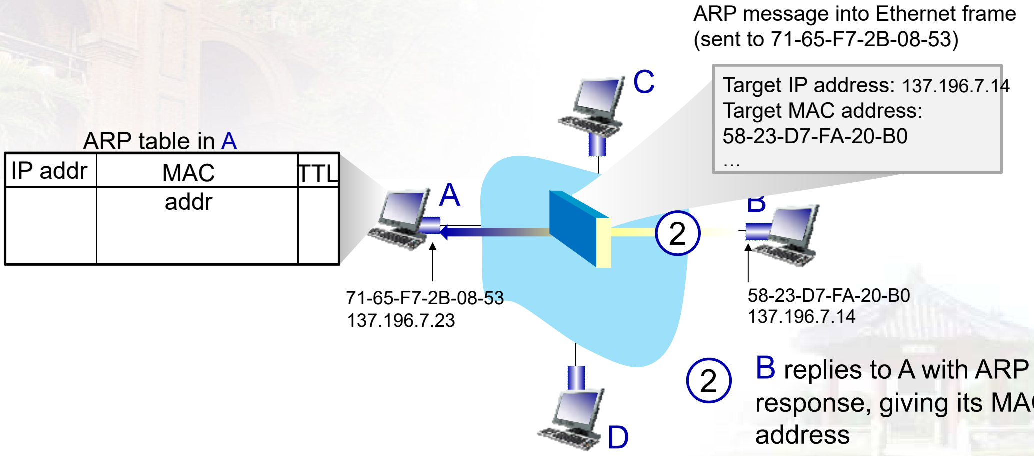
TTL



ARP protocol in action

example: A wants to send datagram to B

- B' s MAC address not in A's ARP table, so A uses ARP to find B's MAC address

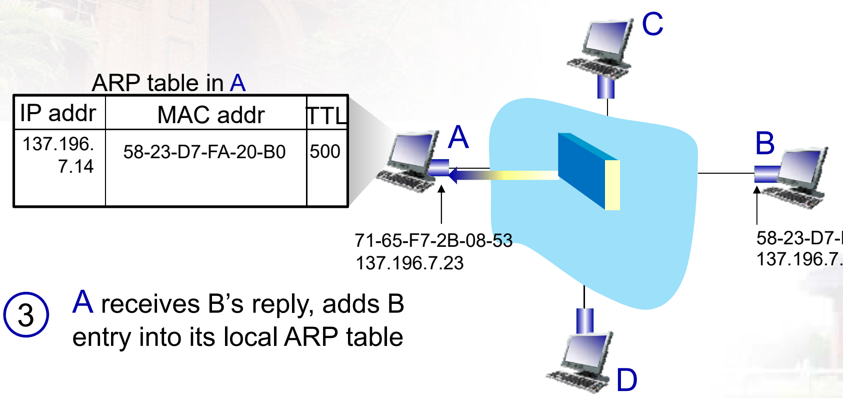


2 B replies to A with ARP response, giving its MAC address

ARP protocol in action

example: A wants to send datagram to B

- B' s MAC address not in A's ARP table, so A uses ARP to find B's MAC address

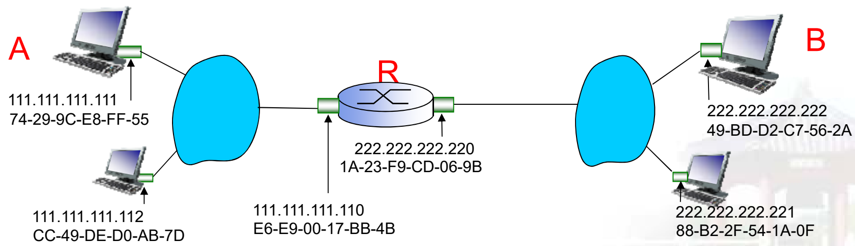


3 A receives B's reply, adds B entry into its local ARP table

Addressing: routing to another LAN

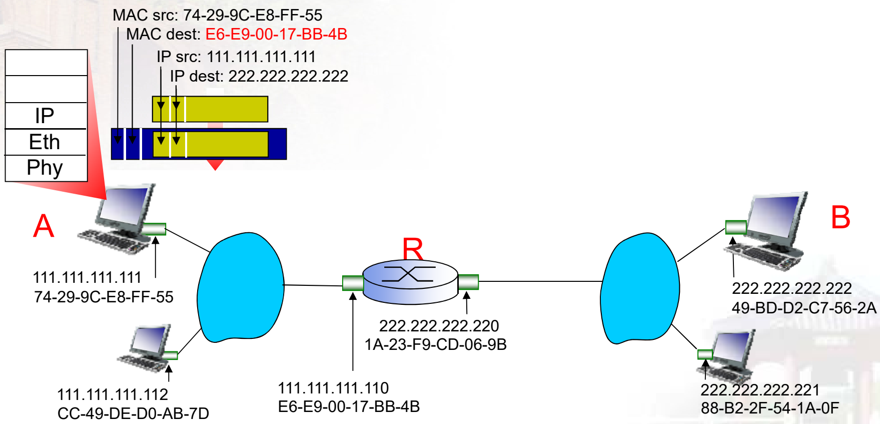
walkthrough: send datagram from A to B via R

- 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?)



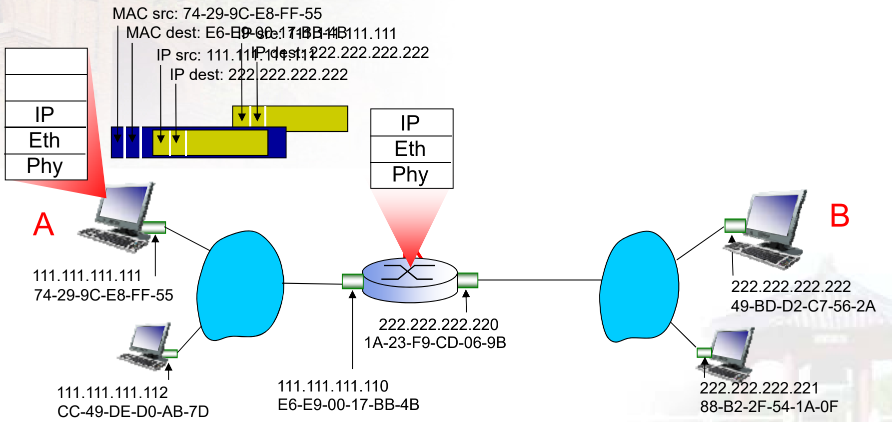
Addressing: routing to another LAN

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



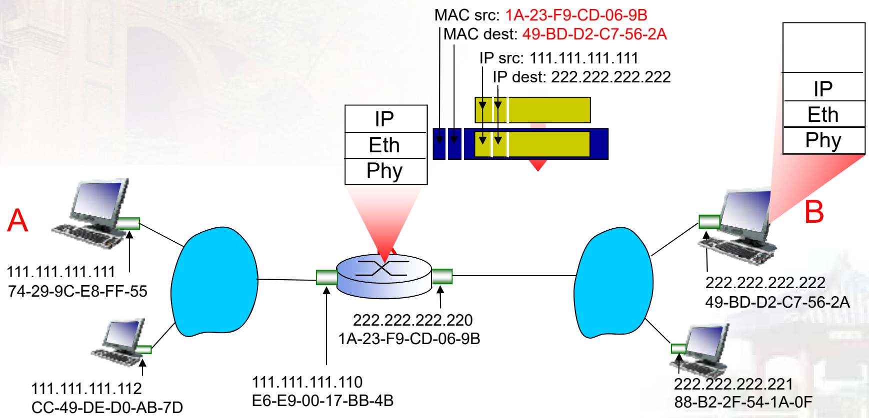
Addressing: routing to another LAN

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



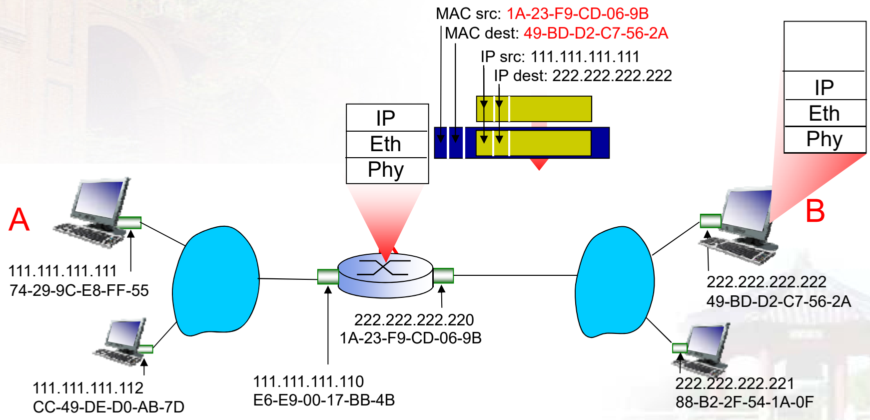
Addressing: routing to another LAN

- 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



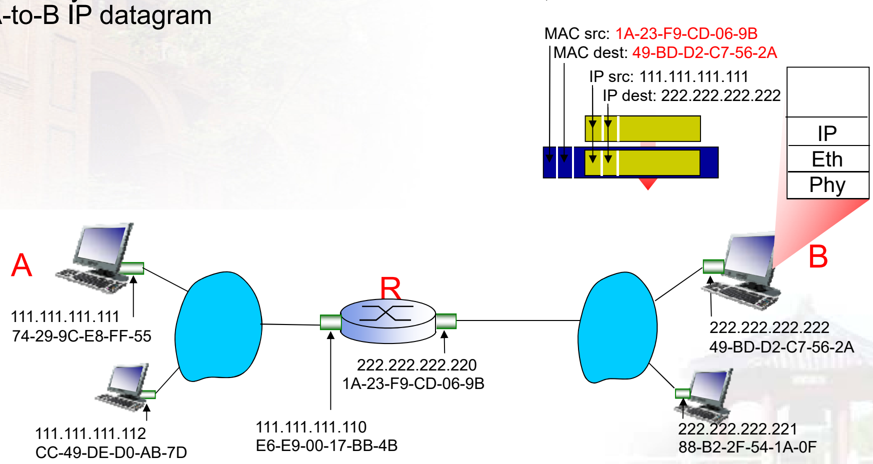
Addressing: routing to another LAN

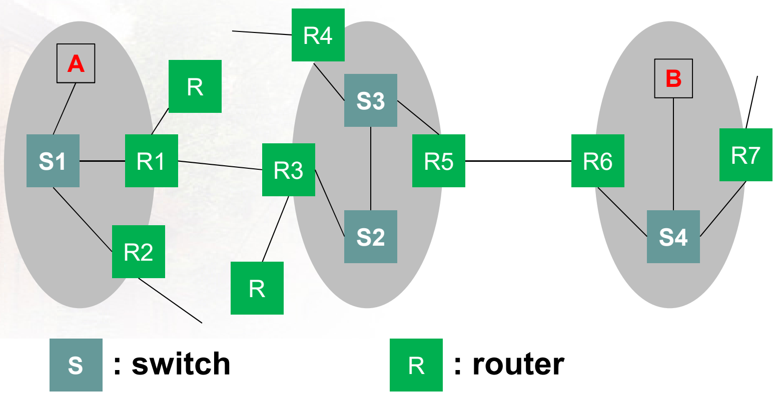
- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



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





A sends an IP to B, please describe the entire addressing process, including the IP and MAC.

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

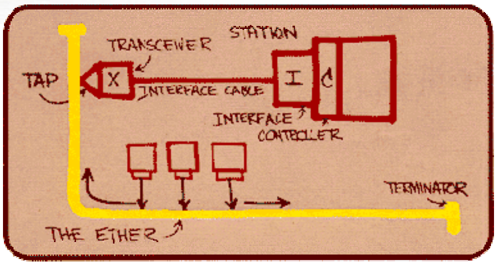
6.5 link virtualization: MPLS

6.6 data center networking

6.7 a day in the life of a web request

Ethernet

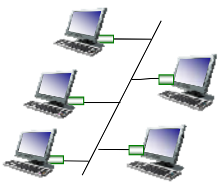
"dominant" wired LAN technology:
- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



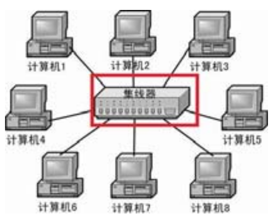
Metcalfe's Ethernet sketch

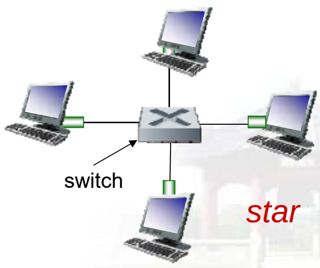
Ethernet: physical topology

- *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

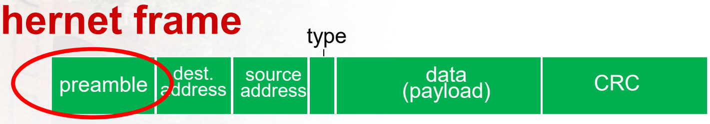




star

Ethernet frame structure

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

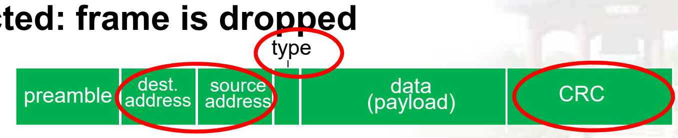


preamble:

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

Ethernet frame structure (more)

- ***addresses:*** 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with **broadcast address** (e.g. ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- ***type:*** indicates **higher layer protocol** (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- ***CRC:*** cyclic redundancy check at receiver
  - error detected: frame is dropped

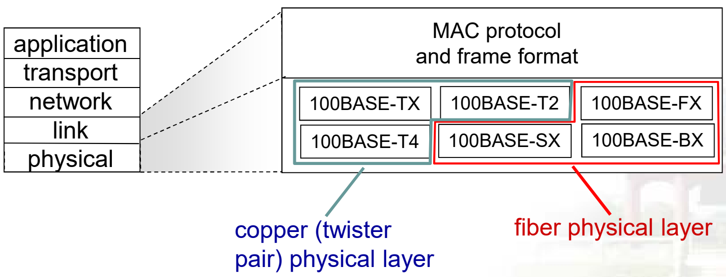


Ethernet: unreliable, connectionless

- *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 wth binary backoff*

802.3 Ethernet standards: link & physical layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - different physical layer media: fiber, cable



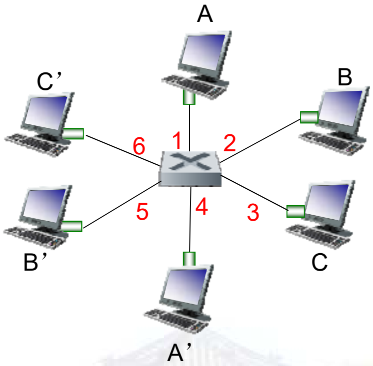
Link layer, LANs: outline

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 (No IP and MAC address)*
  - 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



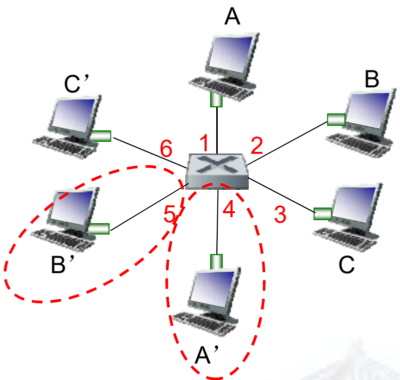
switch with six interfaces (1,2,3,4,5,6)

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!

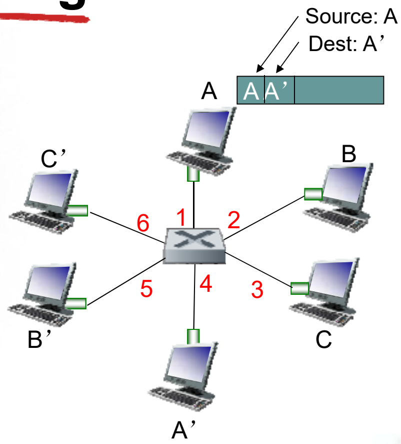
Q: how are entries created, maintained in switch table?
- something like a routing protocol?



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

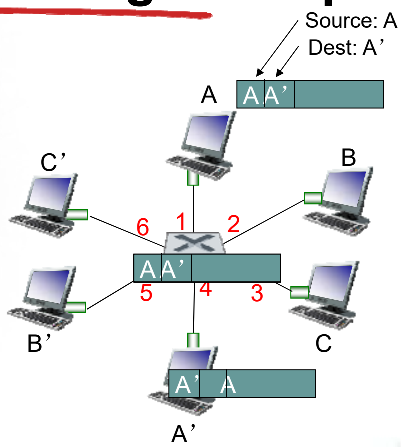


MAC addr	interface	TTL
A	1	*60*

Switch table (initially empty)

Self-learning, forwarding: example

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

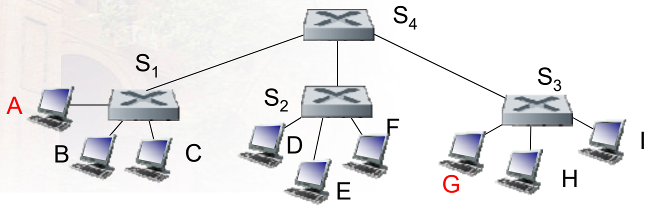


MAC addr	interface	TTL
A A'	1 4	*60* *60*

switch table (initially empty)

Interconnecting switches

- switches can be connected together

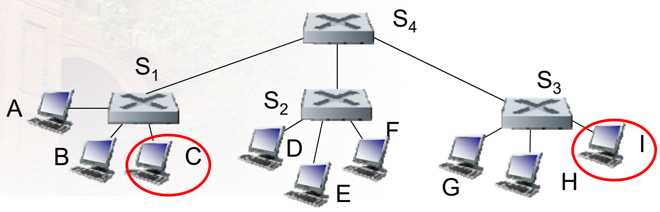


Q: sending from **A to G** - how does S_1 know to forward frame destined to F via S_4 and S_3?

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

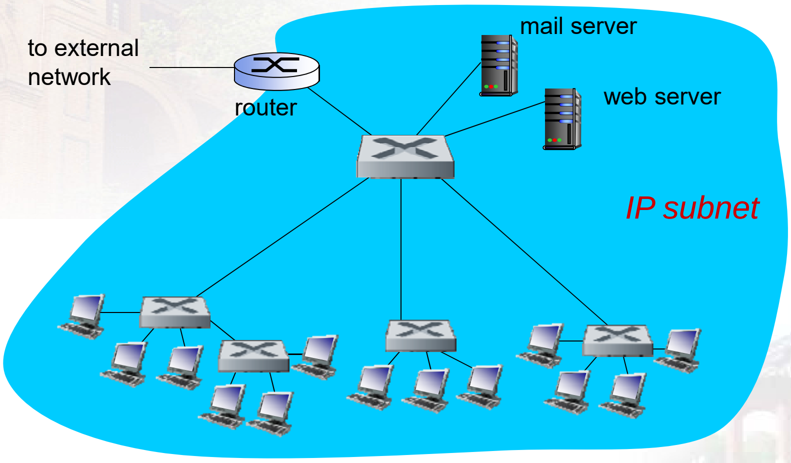
Self-learning multi-switch example

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



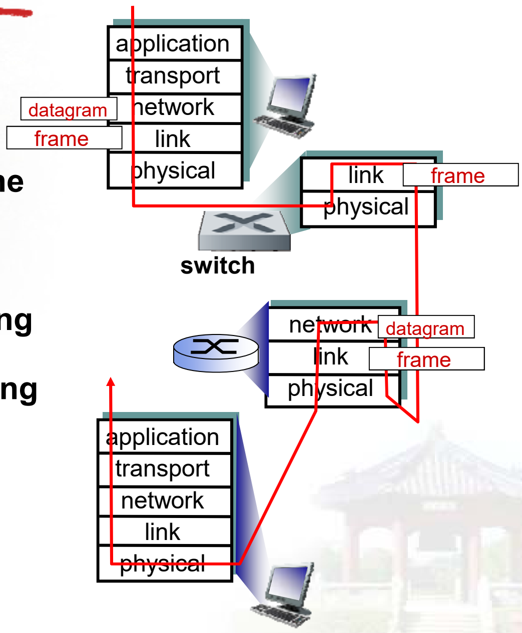
Switches vs. routers

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**

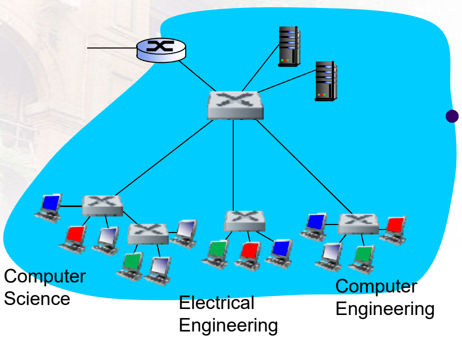


Link layer, LANs: outline

6.1 introduction, services
6.2 error detection, correction
6.3 multiple access protocols
6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS
6.6 data center networking
6.7 a day in the life of a web request

VLANs: motivation

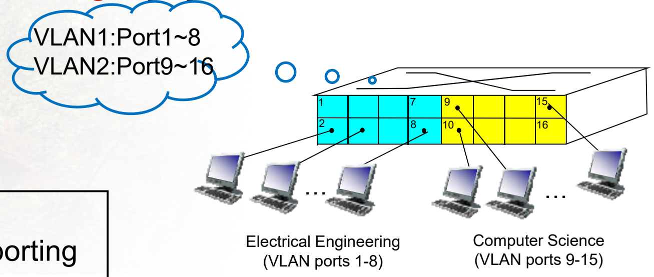


What is the difference between VLAN and the general LAN formed by router?

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

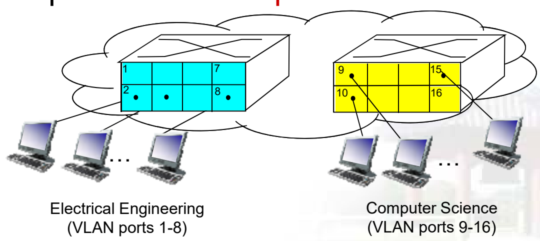
port-based VLAN: switch ports grouped (by switch management software) so that ***single*** physical switch ……



Virtual Local Area Network

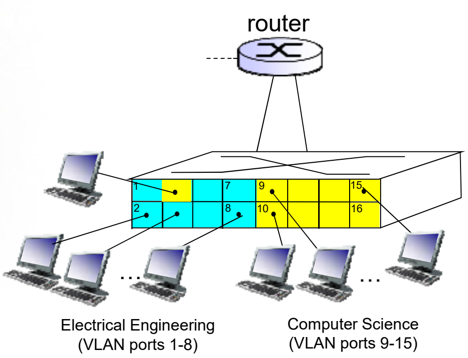
switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

… operates as multiple virtual switches

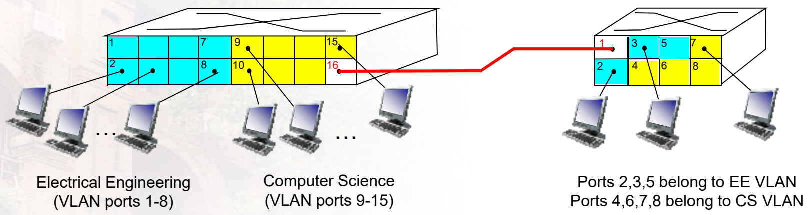


Port-based VLAN

- ***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

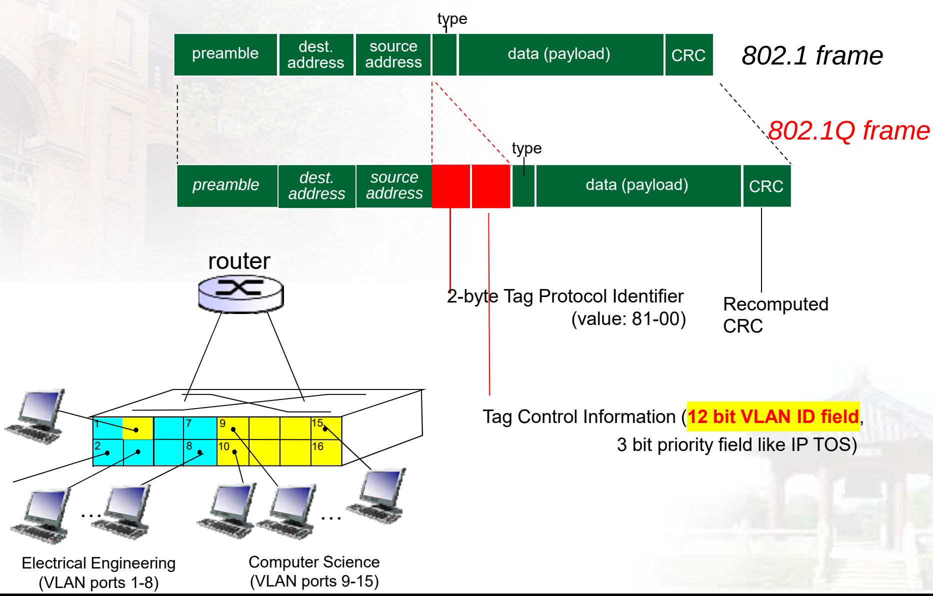


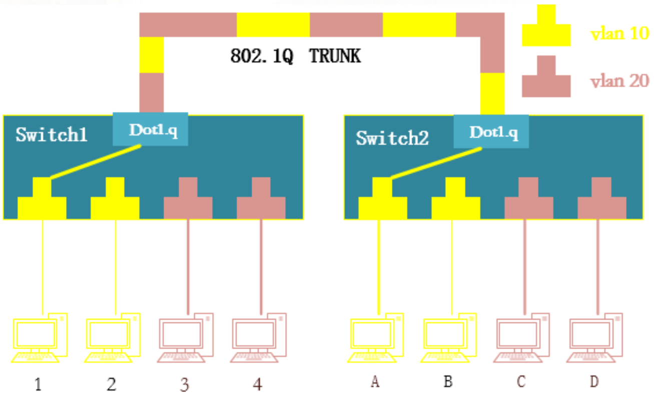
VLANS spanning multiple switches



- *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





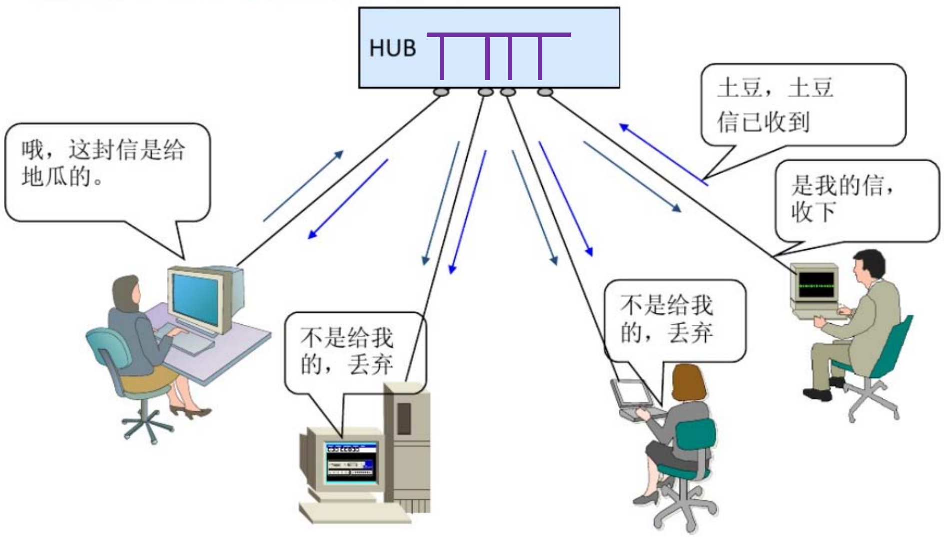
同一个交换机中，端口/MAC与VLAN是捆绑关系，因此，主机发送帧不需要带上VLAN ID。

在跨越不同交换机时，下一个交换机无法通过普通帧判断它所属的VLAN ID，所以，从Trunk口发出去前，必须插入VLAN ID。

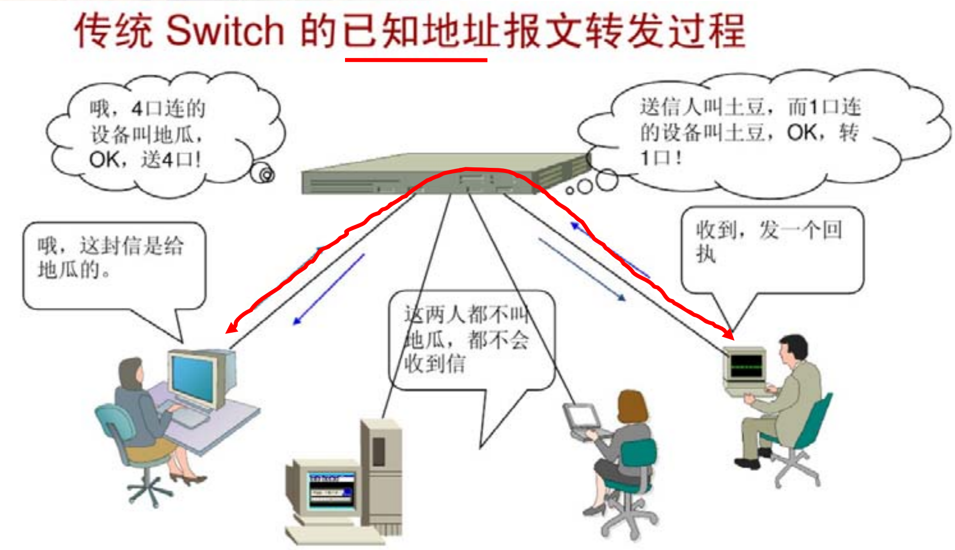
1).1主机发送普通的数据帧;
2).switch1收到此帧首先需要对其解封装，查看二层帧头部帧目的MAC地址;
3).从表中查找其目的MAC地址对应的VLAN ID与接收该帧的接口对应的VLAN ID 是否相同，如果相同则找到对应的出接口，如果不同则丢弃该帧;
4).找到出接口后，打上对应的VLAN 标签，封装成802.1Q的帧，从Trunk接口发送出去;
5).到达switch2后，解封装查看帧头部的目的MAC地址;
6).从表中查找其目的MAC地址对应的VLAN ID与接收该帧头部的VLAN ID是否匹配，如果匹配，则查找对应的出接口，如果不同则丢弃该帧;
7).找到出接口后，封装成原始的帧，从相应端口转发出去。

HUB、2-layer switch、 3-layer switch

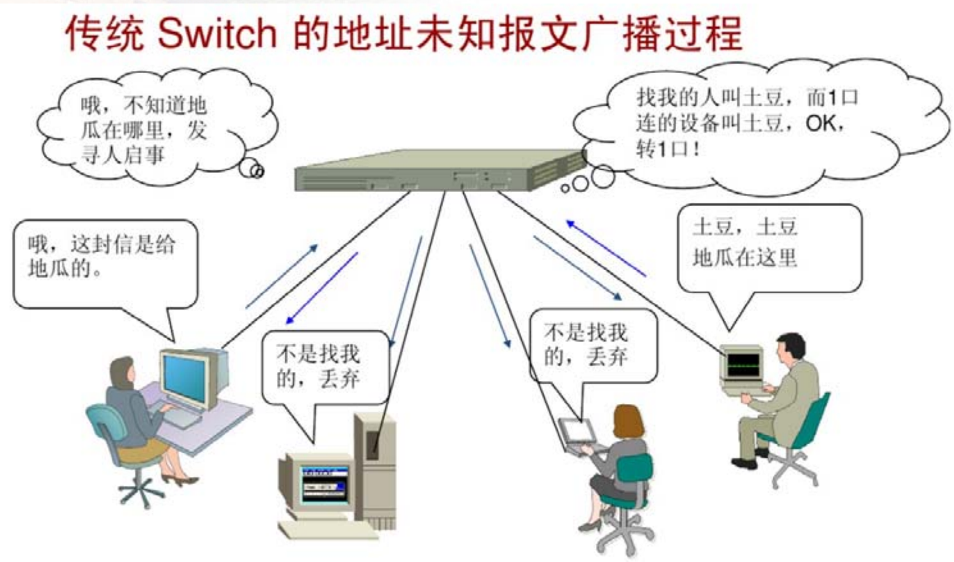
传统 HUB 的工作过程



HUB、2-layer switch、 3-layer switch

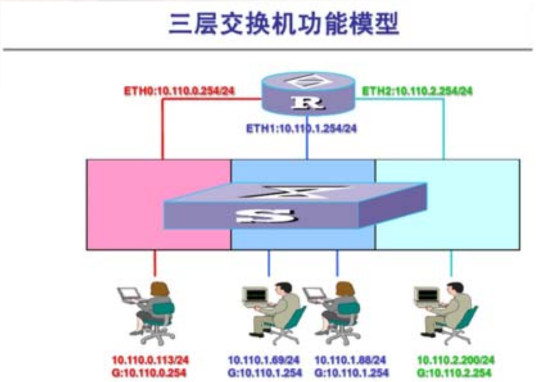


HUB、2-layer switch、 3-layer switch



HUB、2-layer switch、 3-layer switch

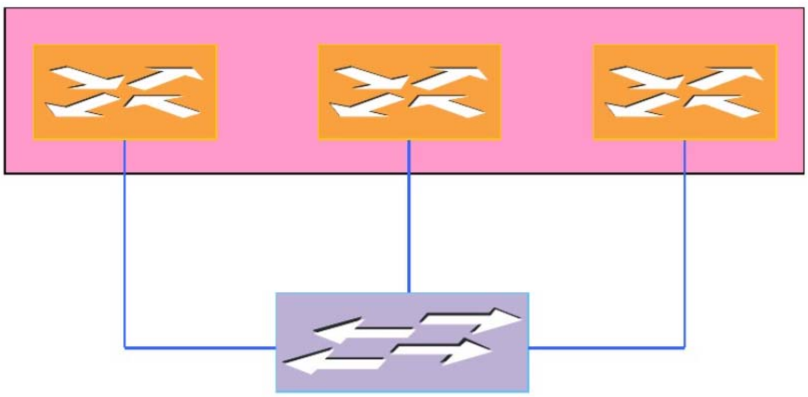
- 三层交换技术：二层交换技术+三层转发技术。解决局域网中网段划分之后，网段中子网必须依赖路由器进行管理的局面，解决了传统路由器低速、复杂所造成的网络瓶颈问题。



第三层交换、第四层交换、多层交换、多层数据包分类和路由交换机

HUB、2-layer switch、 3-layer switch

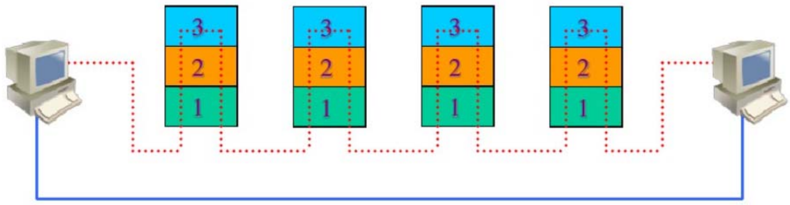
三层交换机中的路由和二层交换



- 二层交换引擎：实现同一网段内的快速二层转发
- 三层路由引擎：实现跨网段的三层路由转发

HUB、2-layer switch、 3-layer switch

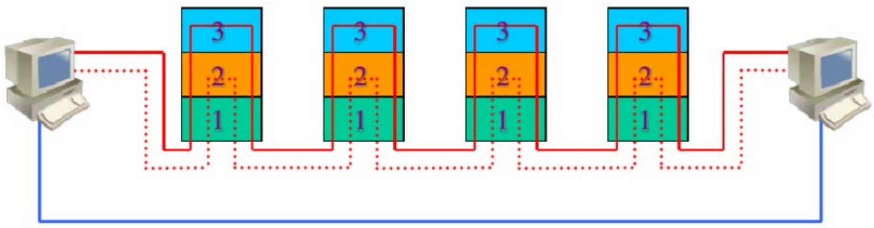
报文到报文的三层交换技术



- 传统三层技术对每个报文进行处理，并基于第三层地址转发报文。这一方法称为报文到报文（Px P）。

HUB、2-layer switch、 3-layer switch

基于流交换的三层交换技术



- 不在三层处理所有报文的的方法称之为流交换（FS）。

—— 第一个报文

⋯⋯ 后续报文

HUB、2-layer switch、 3-layer switch

假设两个使用IP协议的站点A、B通过第三层交换机进行通信

- 发送站点A在开始发送时，把自己的IP地址与B站的IP地址比较，判断B站是否与自己在同一子网内。
- 若目的站B与发送站A在同一子网内，则进行二层的转发。
- 若两个站点不在同一子网内，发送站A要向“缺省网关”发出ARP(地址解析)封包，而“缺省网关”的IP地址其实是三层交换机的三层交换模块。

HUB、2-layer switch、 3-layer switch

- 当发送站A向“缺省网关”的IP发送ARP请求时，如果三层交换模块在以前的通信过程中已经知道B站的MAC地址，则向发送站A回复B的MAC地址。否则三层交换模块根据路由信息向B站广播一个ARP请求;
- B站得到此ARP请求后向三层交换模块回复其MAC地址;
- 三层交换模块保存此地址并回复给发送站A,同时将B站的MAC地址发送到二层交换引擎的MAC地址表中。

HUB、2-layer switch、 3-layer switch

- 从这以后，当A向B发送的数据包便全部交给二层交换处理，信息得以高速交换。由于仅仅在路由过程中才需要三层处理，绝大部分数据都通过二层交换转发，因此三层交换机的速度很快，接近二层交换机的速度

方法二: 交换机直接把三层的目的IP映射到二层目的主机的MAC所对应的端口,三层交换机的交换表:

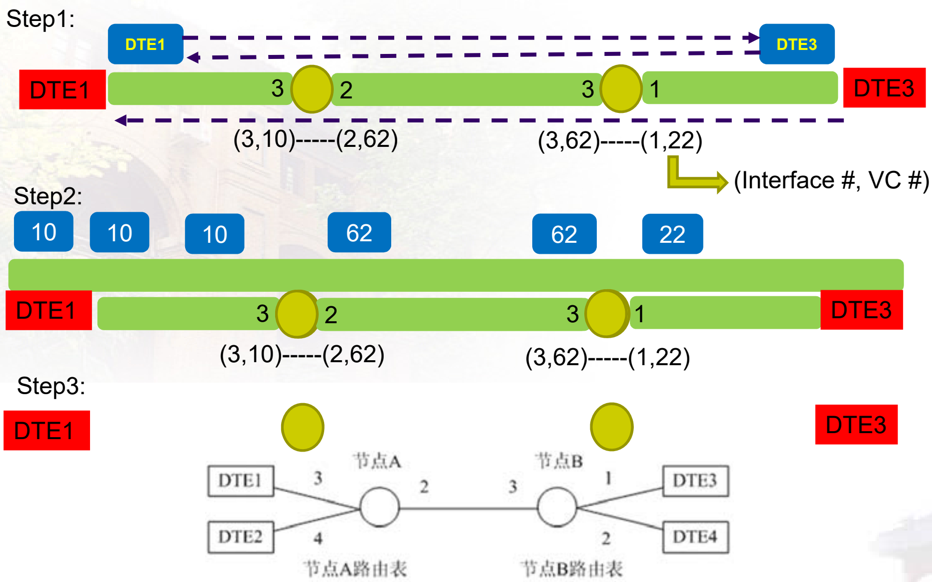
目的IP地址	目的主机MAC	输出端口

Link layer, LANs: outline

15周:2022.12.9

6.1 introduction, services
6.2 error detection, correction
6.3 multiple access protocols
6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS
6.6 data center networking
6.7 a day in the life of a web request



实质：电路交换与分组交换的结合体！

The VC mode

节点A路由表

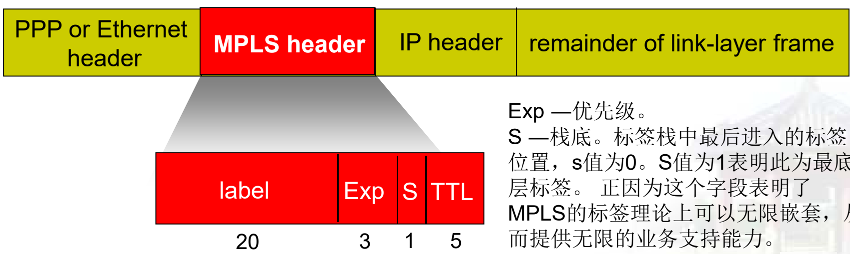
呼叫	入		出	
	端口号	LCN	端口号	LCN
1	3	10	2	62
2	4	12	2	63

节点B路由表

呼叫	入		出	
	端口号	LCN	端口号	LCN
1	3	62	1	22
2	3	63	2	23

Multiprotocol label switching (MPLS)

- initial goal: high-speed IP forwarding using fixed length label (instead of IP address)
  - fast lookup using fixed length identifier (rather than shortest prefix matching)
  - borrowing ideas from Virtual Circuit (VC) approach
  - but IP datagram still keeps IP address!

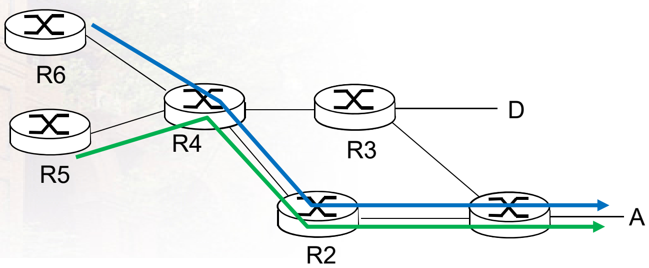


Exp —优先级。
S —栈底。标签栈中最后进入的标签位置，s值为0。S值为1表明此为最底层标签。 正因为这个字段表明了MPLS的标签理论上可以无限嵌套，从而提供无限的业务支持能力。

MPLS capable routers

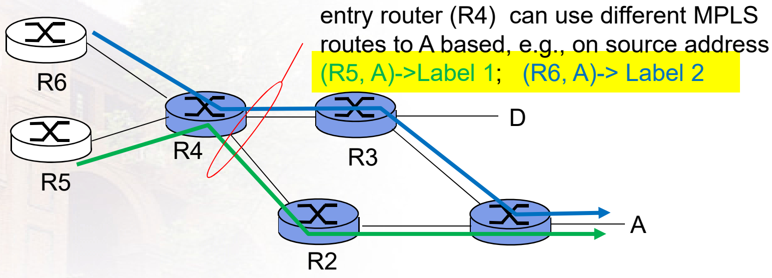
- a.k.a. label-switched router
- forward packets to outgoing interface based only on label value (*don't inspect IP address*)
  - MPLS forwarding table distinct from IP forwarding tables
- *flexibility:* MPLS forwarding decisions can *differ* from those of IP
  - use destination *and* source addresses to route flows to same destination differently (traffic engineering)
  - re-route flows quickly if link fails: pre-computed backup paths (useful for VoIP)

MPLS versus IP paths

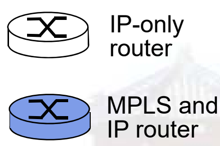


- **IP routing**: path to destination determined by destination address alone

MPLS versus IP paths

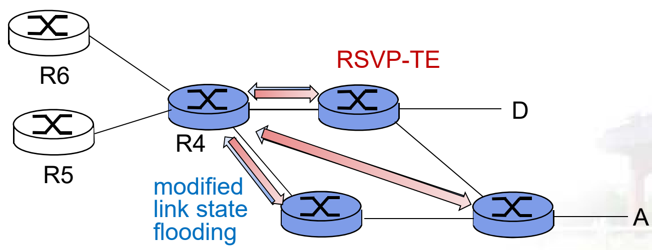


- IP routing: path to destination determined by destination address alone
- MPLS routing: path to destination can be based on source and dest. address
  - fast reroute: precompute backup routes in case of link failure



MPLS signaling

- **modify OSPF link-state flooding protocols to carry info used by MPLS routing,**
  - **e.g., link bandwidth, amount of “reserved” link bandwidth**
    - *entry MPLS router uses RSVP-TE signaling protocol to set up MPLS forwarding at downstream routers*



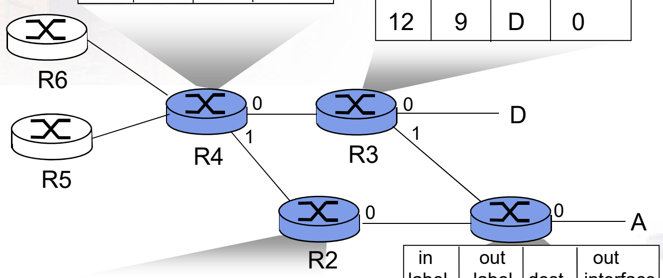
MPLS forwarding tables

R4:

in label	out label	dest	out interface
	10	A	0
	12	D	0
	8	A	1

R3:

in label	out label	dest	out interface
10	6	A	1
12	9	D	0

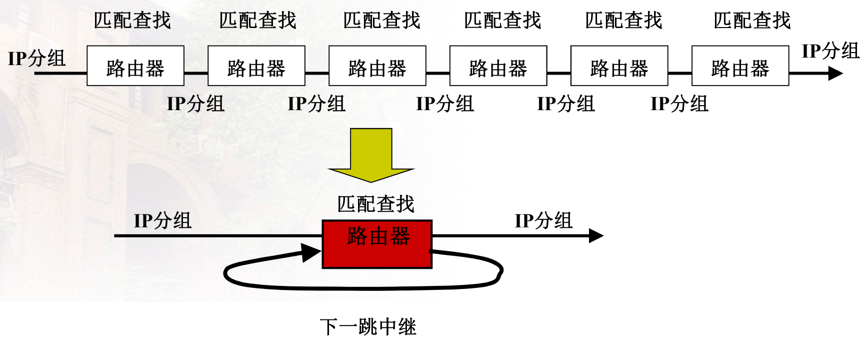


R2:

in label	out label	dest	out interface
8	6	A	0

R1:

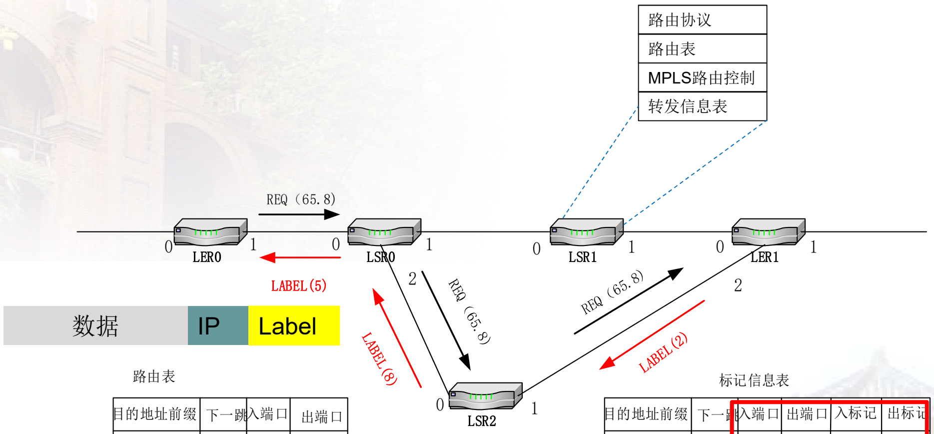
in label	out label	dest	out interface
6	-	A	0



IP网络的逐跳式分组转发

Step1: routing for the path;
Step2: labeling each link;
Step3: forwarding based on label

Replace the **global** routing table with the **local** label switching table



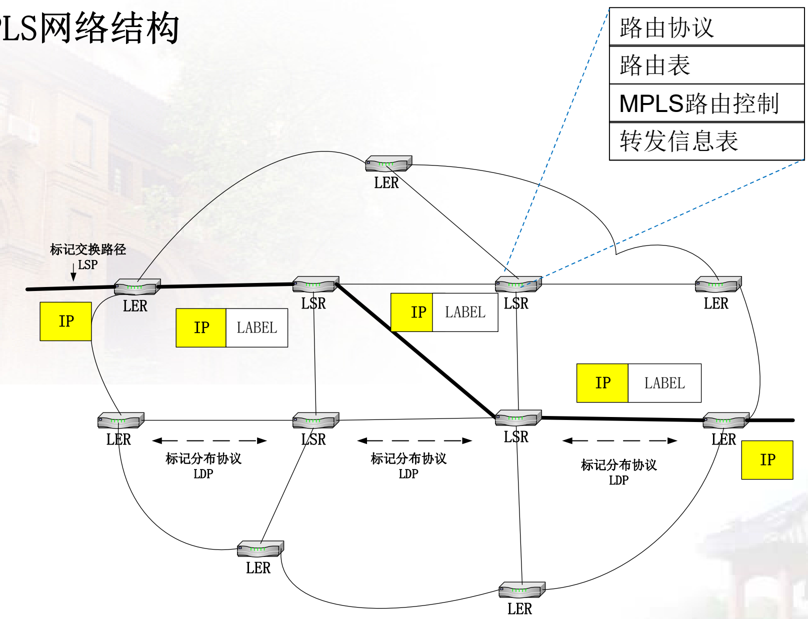
路由表

	目的地址前缀	下一跳	入端口	出端口
LER0	65.8	LSR0	0	1
LSR0	65.8	LSR2	0	2
LSR2	65.8	LER1	0	1
LER1	65.8	----	0	---

标记信息表

	目的地址前缀	下一跳	入端口	出端口	入标记	出标记
LER0	65.8	LSR0	0	1	---	5
LSR0	65.8	LSR2	0	2	5	8
LSR2	65.8	LER1	0	1	8	2
LER1	65.8	----	0	---	2	---

MPLS网络结构



Link layer, LANs: outline

6.1 introduction, services
6.2 error detection, correction
6.3 multiple access protocols
6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS
6.6 data center networking
6.7 a day in the life of a web request

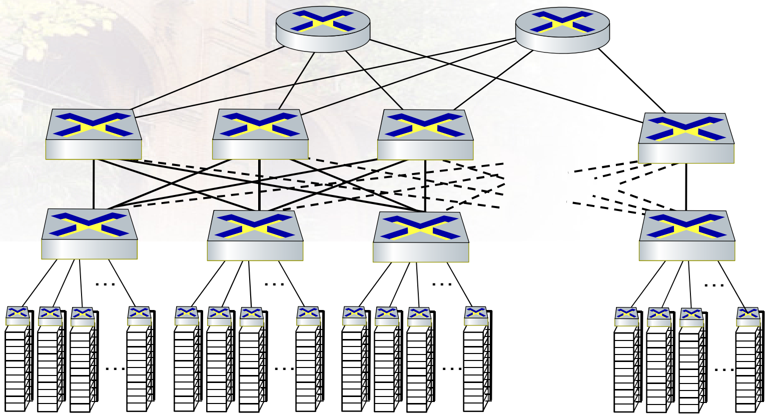
Data center networks

- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
  - e-business (e.g. Amazon)
  - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
  - search engines, data mining (e.g., Google)

- challenges:
  - multiple applications, each serving massive numbers of clients
  - managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Datacenter networks: network elements



Border routers
- connections outside datacenter

Tier-1 switches
- connecting to ~16 T-2s below

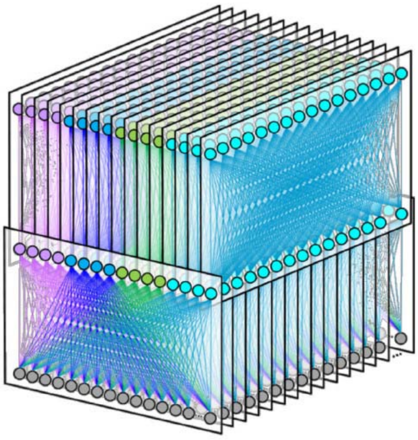
Tier-2 switches
- connecting to ~16 TORs below

Top of Rack (TOR) switch
- one per rack
- 40-100Gbps Ethernet to blades

Server racks
- 20- 40 server blades: hosts

Datacenter networks: network elements

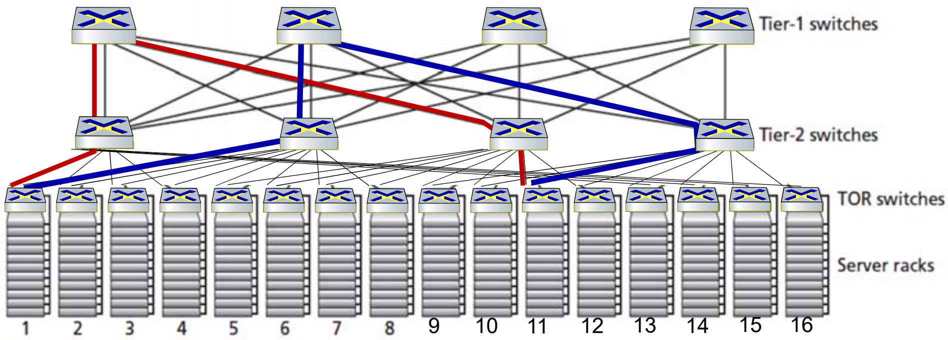
Facebook F16 data center network topology:



https://engineering.fb.com/data-center-engineering/f16-minipack/ (posted 3/2019)

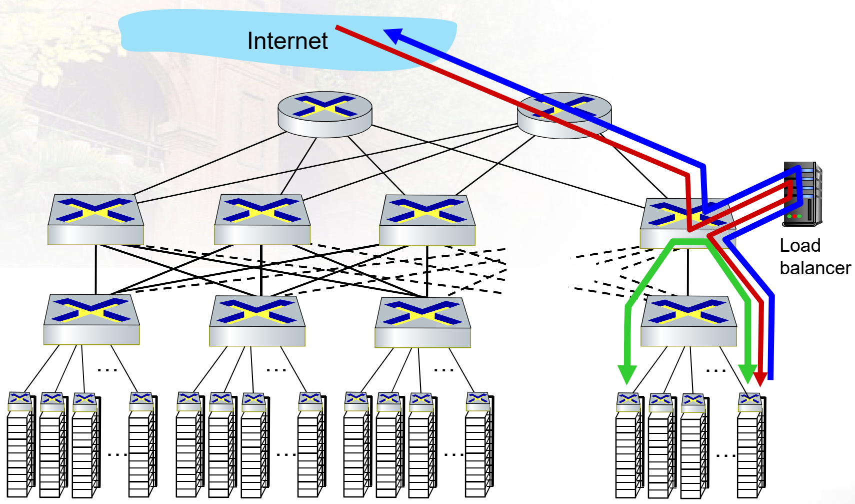
Datacenter networks: multipath

- rich interconnection among switches, racks:
  - increased throughput between racks (multiple routing paths possible)
  - increased reliability via redundancy



two disjoint paths highlighted between racks 1 and 11

Datacenter networks: application-layer routing



load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)

Datacenter networks: protocol innovations

- **link layer:**
  - RoCE: remote DMA (RDMA) over Converged Ethernet
- **transport layer:**
  - ECN (explicit congestion notification) used in transport-layer congestion control (DCTCP, DCQCN)
  - experimentation with hop-by-hop (backpressure) congestion control
- **routing, management:**
  - SDN widely used within/among organizations' datacenters
  - place related services, data as close as possible (e.g., in same rack or nearby rack) to minimize tier-2, tier-1 communication

Link layer, LANs: outline

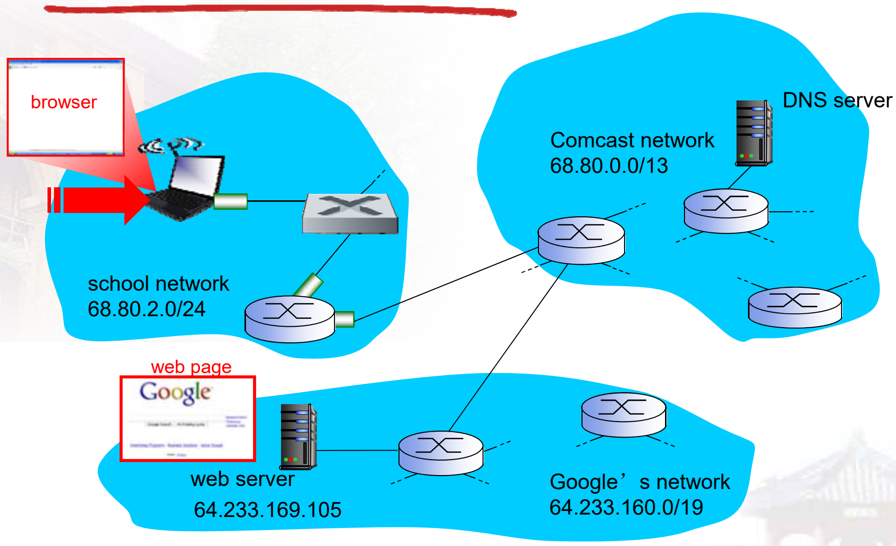
6.1 introduction, services
6.2 error detection, correction
6.3 multiple access protocols
6.4 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS
6.6 data center networking
6.7 a day in the life of a web request

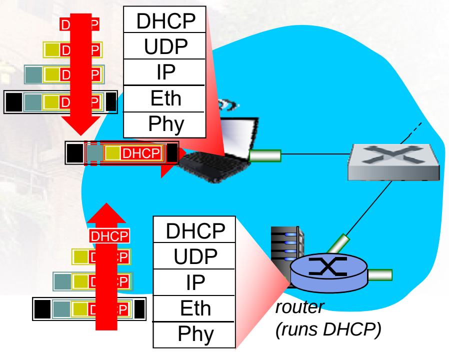
Synthesis: a day in the life of a web request

- journey down protocol stack complete!
  - application, transport, network, link
- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

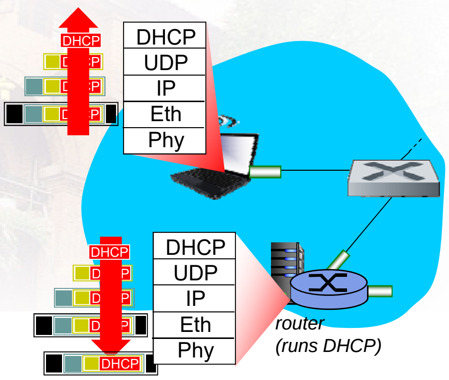


A day in the life… connecting to the Internet



- **connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP***
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.3 Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

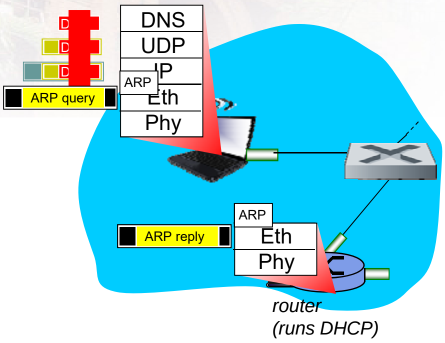
A day in the life… connecting to the Internet



- **DHCP server formulates *DHCP ACK* containing client’s IP address, IP address of first-hop router for client, name & IP address of DNS server**
- encapsulation at DHCP server, frame forwarded (switch learning) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

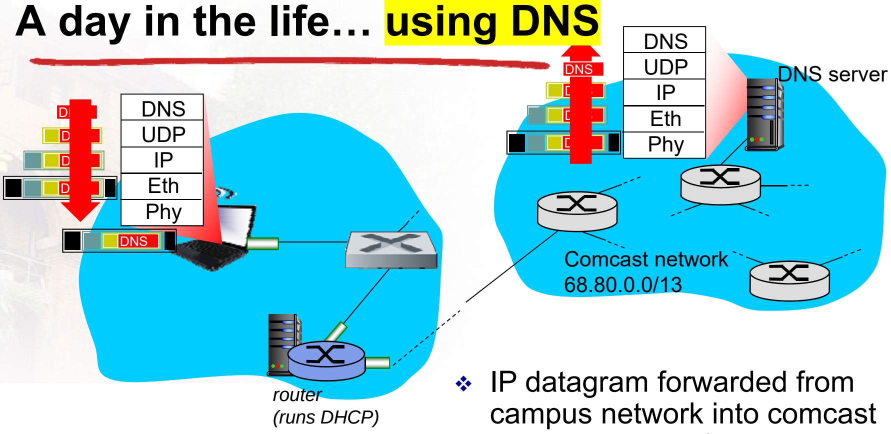
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life… ARP (before DNS, before HTTP)



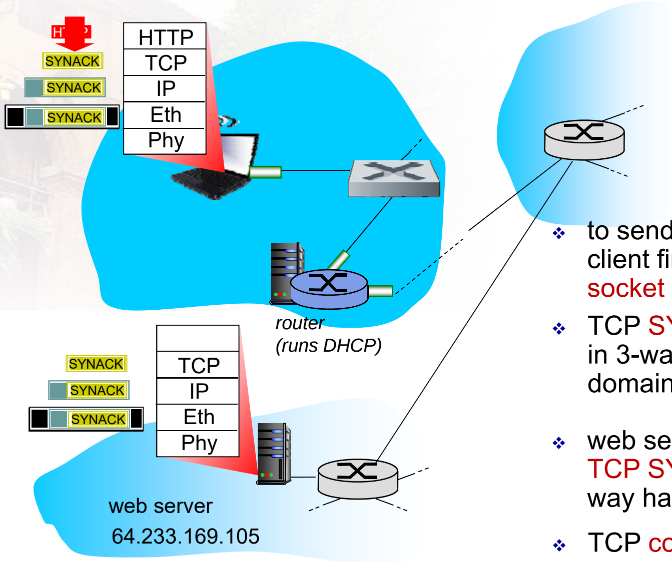
- **before sending *HTTP* request, need IP address of www.google.com: *DNS***
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP
- ARP query broadcast, received by router, which replies with ARP reply giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life… using DNS



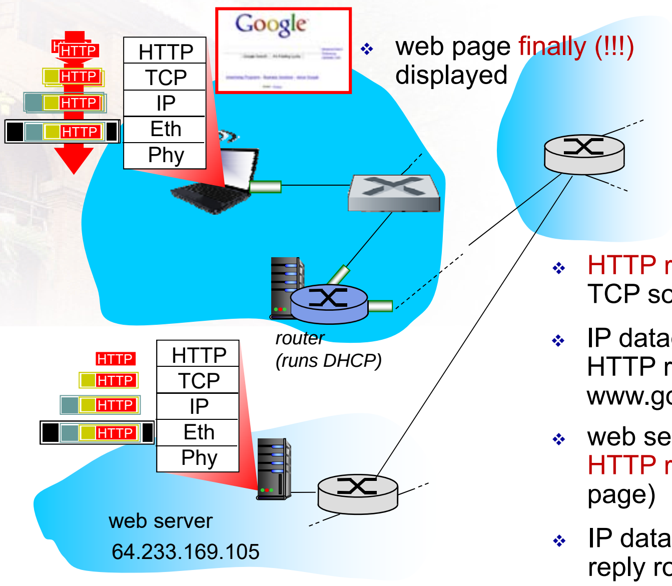
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- Demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



- to send HTTP request, client first opens TCP socket to web server
- TCP SYN segment (step 1 in 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in 3-way handshake)
- TCP connection established!

A day in the life… HTTP request/reply



- web page finally (!!!) displayed
- HTTP request sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with HTTP reply (containing web page)
- IP datagram containing HTTP reply routed back to client

Chapter 6: Summary

- principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies
  - Ethernet
  - switched LANS, VLANs
  - virtualized networks as a link layer: MPLS
- synthesis: a day in the life of a web request

Chapter 6: let' s take a breath

- journey down protocol stack *complete* (except PHY)
- solid understanding of networking principles, practice
- ….. could stop here …. but *lots* of interesting topics!
  - wireless
  - multimedia
  - security

The End of Chapter 6

Thanks

Q & A

Email: xieyi5@mail.sysu.edu.cn
https://cse.sysu.edu.cn/content/2462