SOFTENG 364: Computer Networks

Link Layer

Kurose and Ross, chapter 6

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

ENGINEERING

Learning Outcomes

By the end of this module you should be able to:

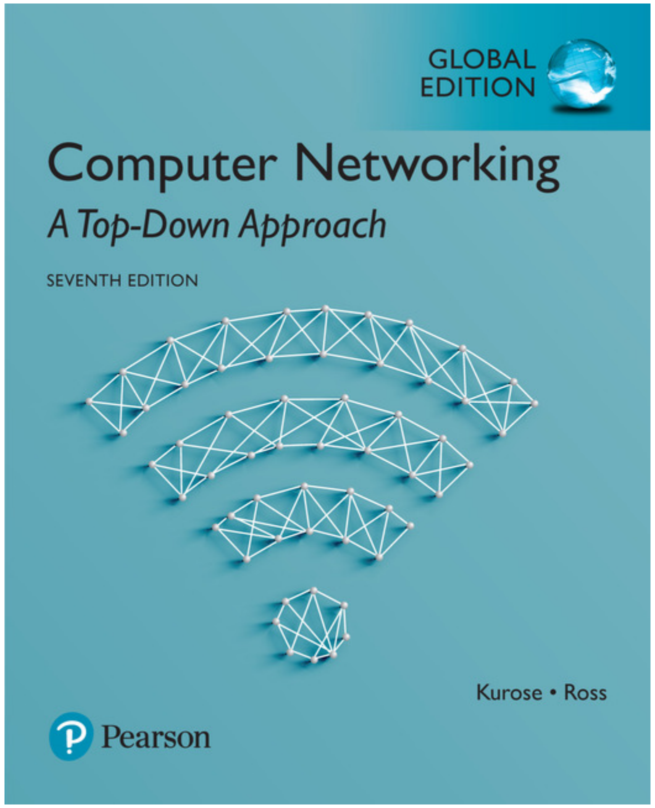
- Describe what the link layer is and its purpose
- Explain what error detection and correction is and how it works
- Explain what the two types of "links" are and what are the different concerns for each
- Describe how channel partitioning protocols work
- Describe how random access protocols work
- Describe how "taking turns" protocols work
- For each protocol, explain the advantages and the disadvantages of the protocol
- Explain how the Ethernet protocols work, including the underlying technologies such as ARP
- Demonstrate how switches are self-learning
- Describe how data centres work and what are the differences from a typical network
- Demonstrate how a machine can connect to a webserver

ENGINEERING

References

Computer Networking: A Top Down Approach, 7th edition (2016). *By J. Kurose & K. Ross.*

- Chapter 6

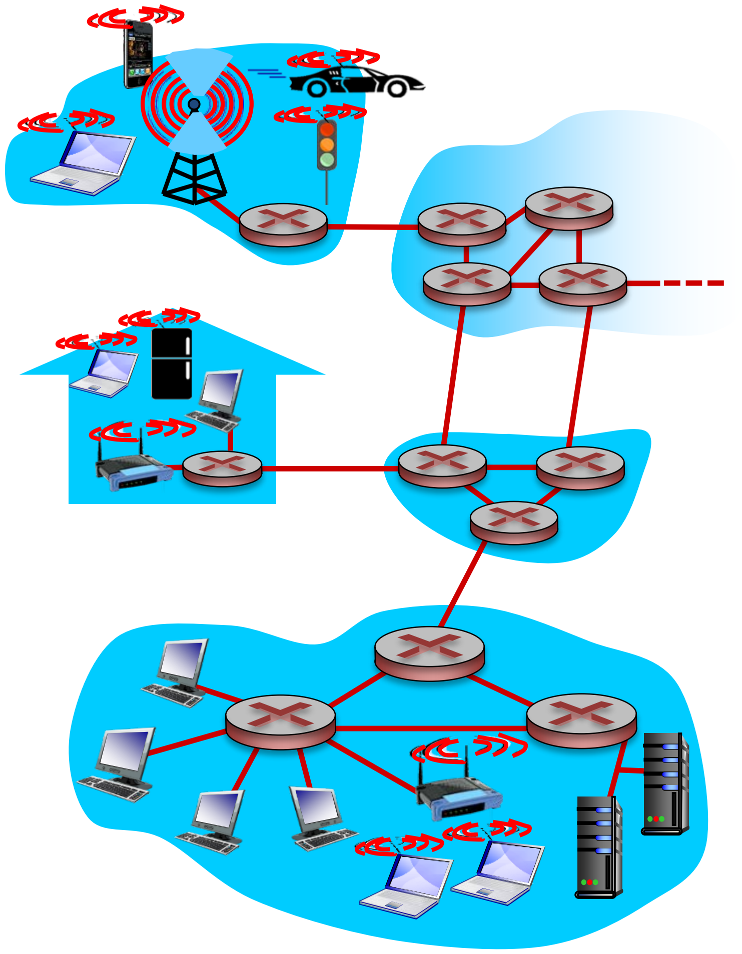


Link layer: introduction

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

Transportation analogy: 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, destination
    - 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

Where is the link layer implemented?

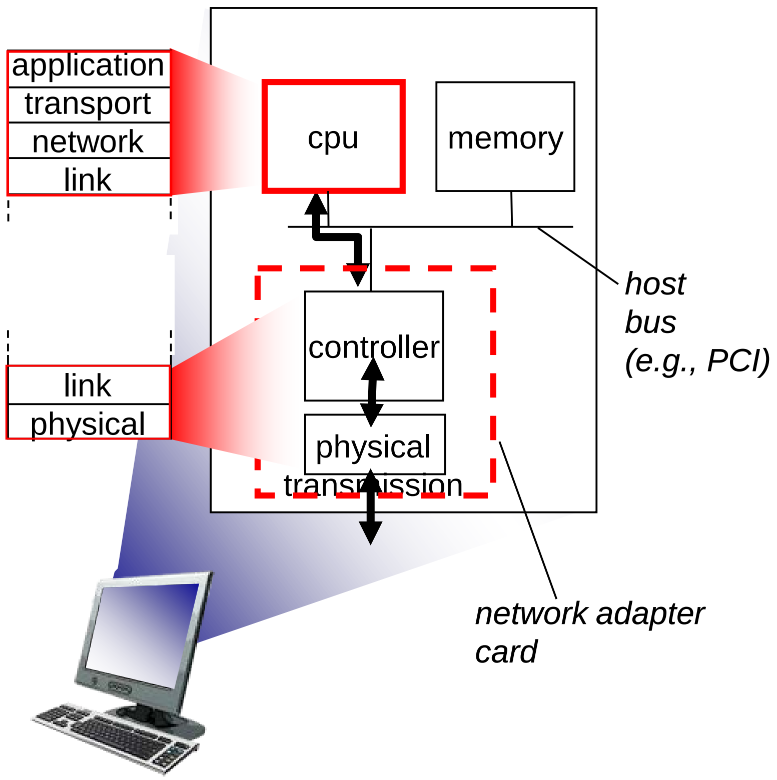
In each and every host

Link layer implemented in “adaptor” (aka *network interface card* NIC) or on a chip

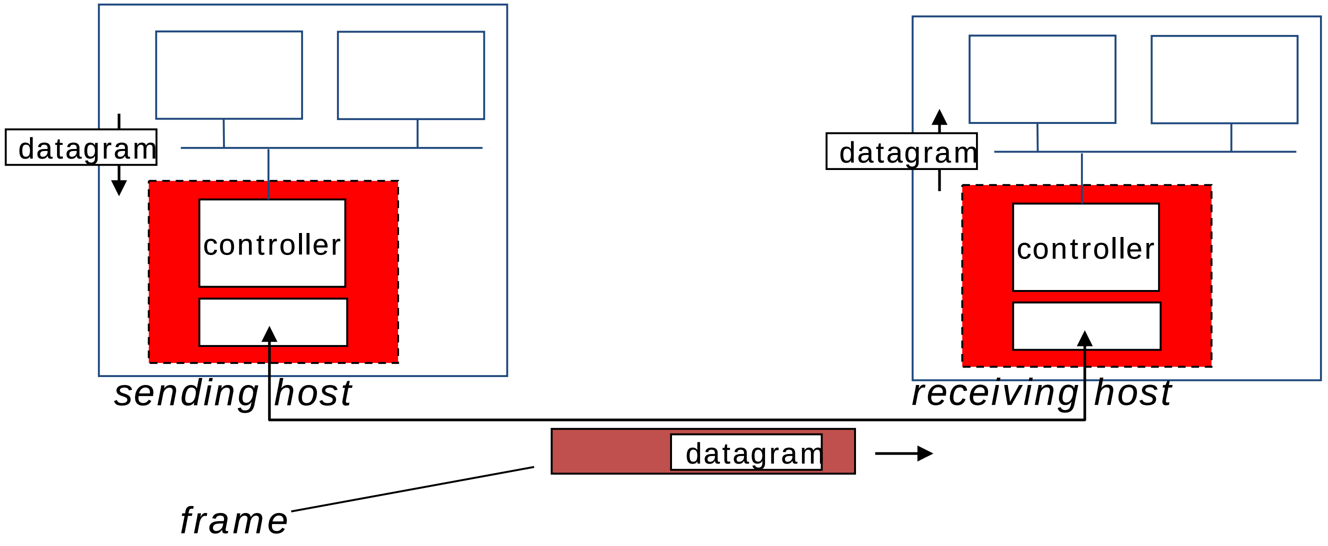
- Ethernet card, 802.11 card; Ethernet chipset
- Implements link, physical layer

Attaches into host’s system buses

Combination of hardware, software, firmware



Adaptors communicating



Sending side:

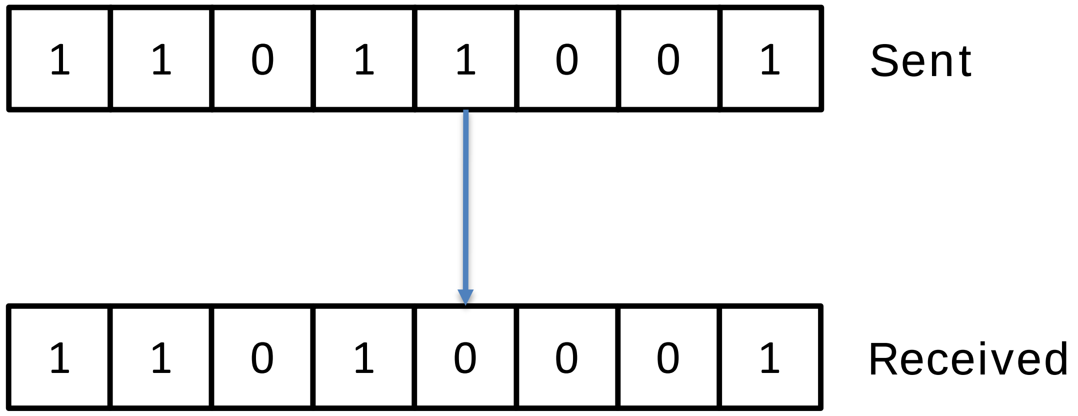
- Encapsulates datagram in frame
- Adds error checking bits, rdt, flow control, etc.

Receiving side

- Looks for errors, rdt, flow control, etc.
- Extracts datagram, passes to upper layer at receiving side

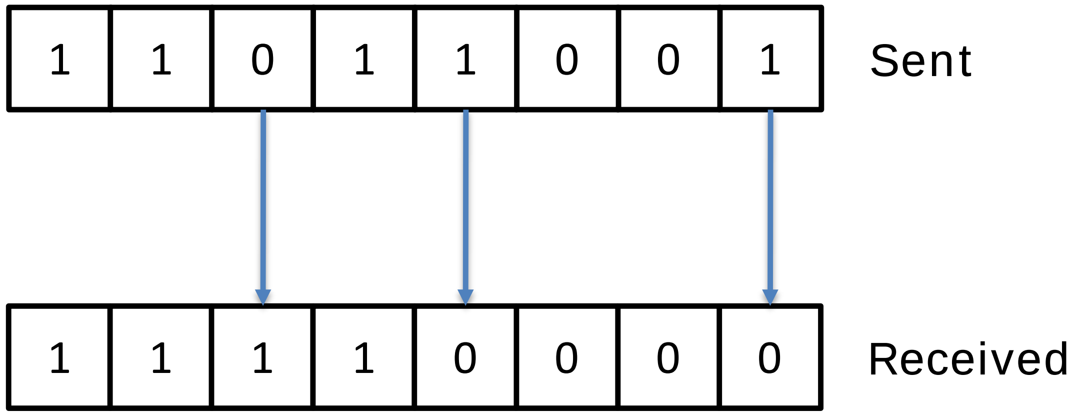
Error Types

Single bit error



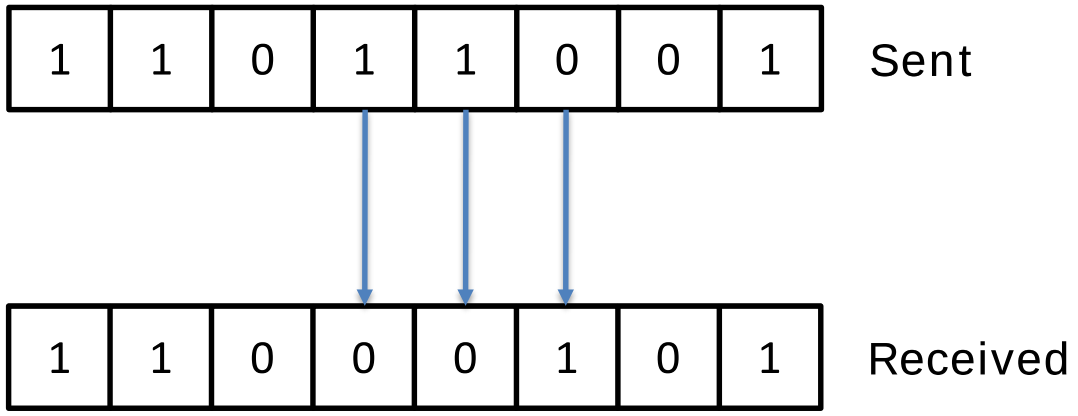
Error Types

Multiple bits error



Error Types

Burst error



How do we know we have an error?
How do we know which bits are in error?

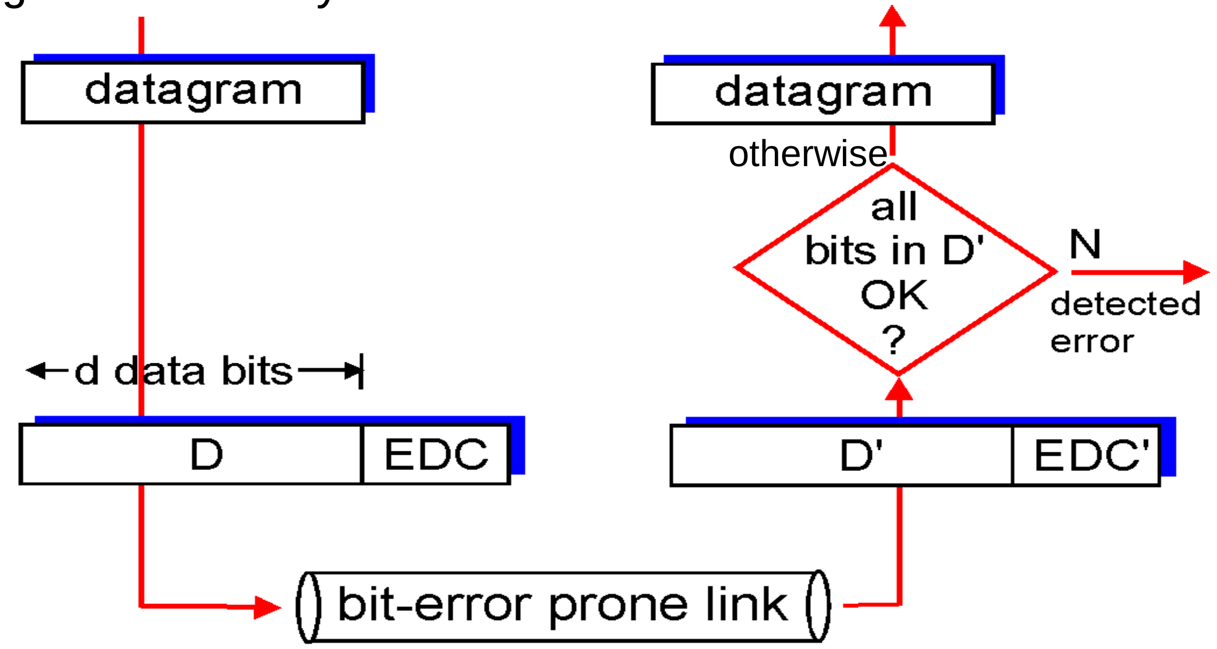
Error detection

EDC = Error Detection and Correction bits (redundancy)

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

Error detection not 100% reliable!

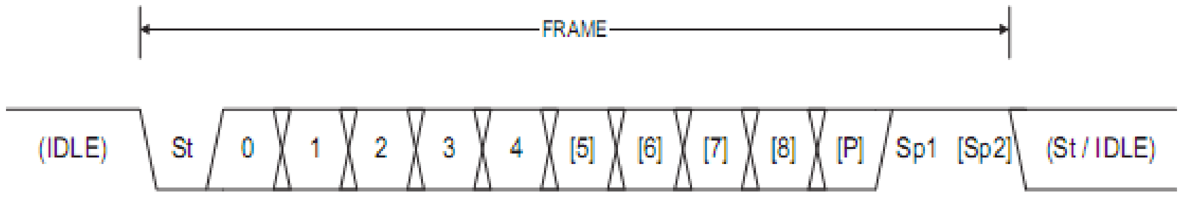
- Protocol may miss some errors, but rarely
- Larger EDC field yields better detection and correction



Frame Format for Parity Checks

USART (a serial protocol) has the following frame format:

- 1 start bit *[Beginning of frame]*
- 5, 6, 7, 8, or 9 data bits
    - This will limit the data ranges i.e., 25, 26, 27, 28, 29
- no, even or odd parity bit
- 1 or 2 stop bits *[End of frame]*



St	Start bit, always low
(n)	Data bits (0 to 8)
P	Parity bit. Can be odd or even
Sp	Stop bit, always high
IDLE	No transfers on the communication line (RxD or TxD). An IDLE line must be high

Error Checking: Parity

A parity bit is used for error checking

- The data is discarded if the revised parity bit does not match the parity computed from the received data

A parity bit is a bit added to a string of binary to ensure that the total number of 1-bits in the string is even or odd.

- Even: sets to ‘1’ if count odd
- i.e. if five ‘1’ bits then the parity bit is set to ‘1’ to make six ‘1’ bits
- Odd: sets to ‘1’ if count even

The Parity bit is calculated by an exclusive-or (XOR) of all data bits. If odd parity is used, the result is inverted.

- If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

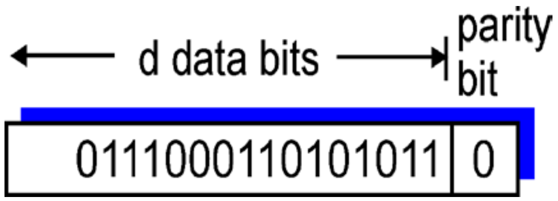
$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even}	Parity bit using even parity
P_{odd}	Parity bit using odd parity
d_n	Data bit n of the character

Parity checking

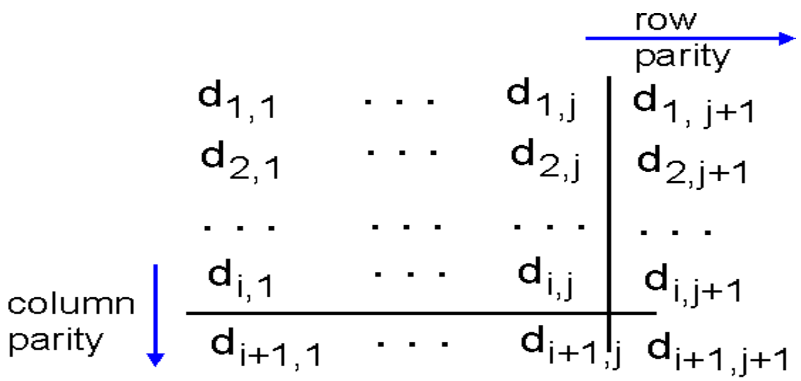
Single bit parity:

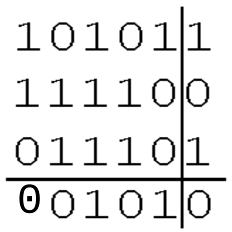
- Detect single bit errors



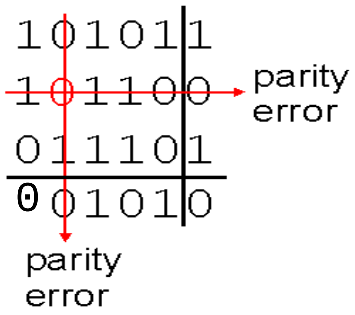
Two-dimensional bit parity:

- Detect and correct single bit errors


row parity
column parity
$d_{1,1}$ ... $d_{1,j}$ $d_{1,j+1}$
$d_{2,1}$ ... $d_{2,j}$ $d_{2,j+1}$
... ...
$d_{i,1}$ ... $d_{i,j}$ $d_{i,j+1}$
$d_{i+1,1}$ ... $d_{i+1,j}$ $d_{i+1,j+1}$



no errors



correctable single bit error

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

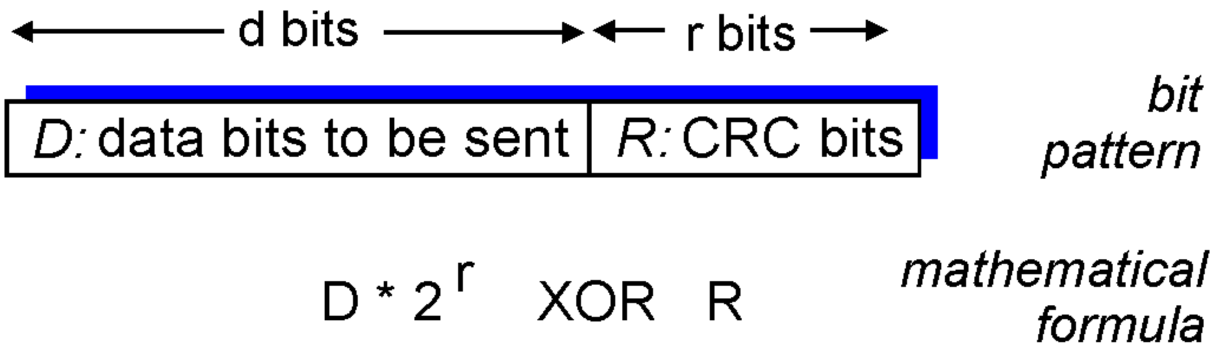
View data bits, D, as a binary number

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)



THE UNIVERSITY OF AUCKLAND Te Whare Wananga o Tamaki Makaurau NEW ZEALAND | ENGINEERING

CRC example

Want:

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

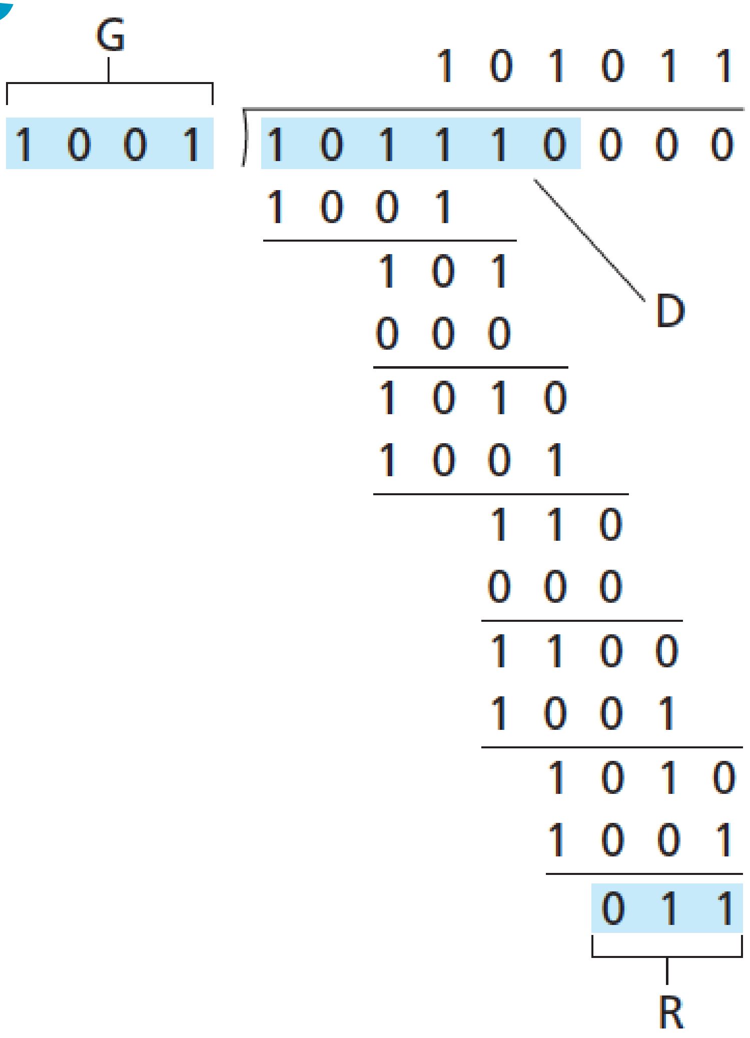
Equivalently:

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

Equivalently:

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

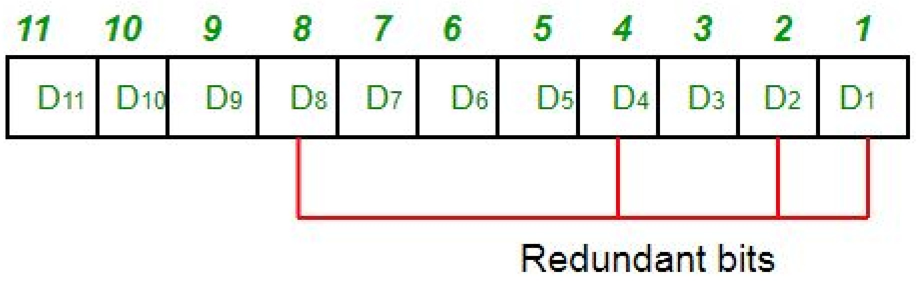
$$R = remainder[\frac{D \cdot 2^r}{G}]$$



Error correction techniques

Some approaches:

- Hamming Codes
- Binary Convolution Code
- Reed – Solomon Code
- Low-Density Parity-Check Code



Multiple access links

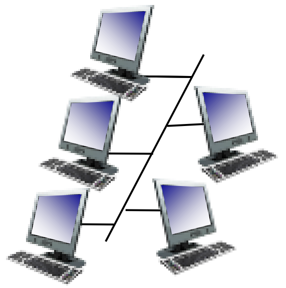
Two types of "links":

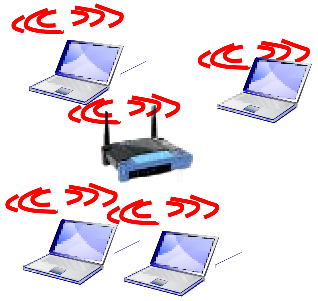
Point-to-point

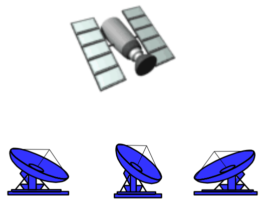
- PPP for dial-up access
- Point-to-point link between Ethernet switch, host

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 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

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

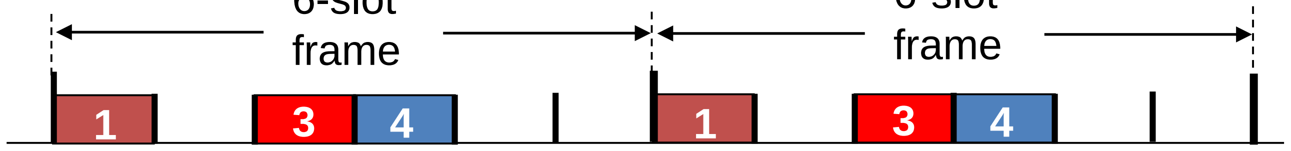
Time Division Multiple Access (TDMA)

Access to channel in "rounds"

Each station gets fixed length slot (length = packet transmission time) in each round

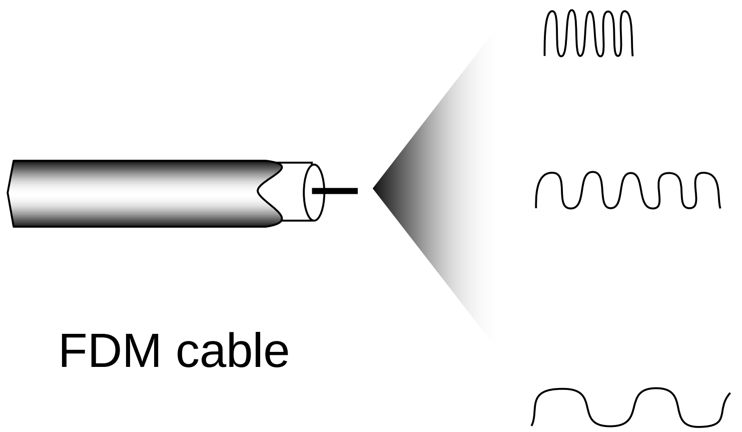
Unused slots go idle

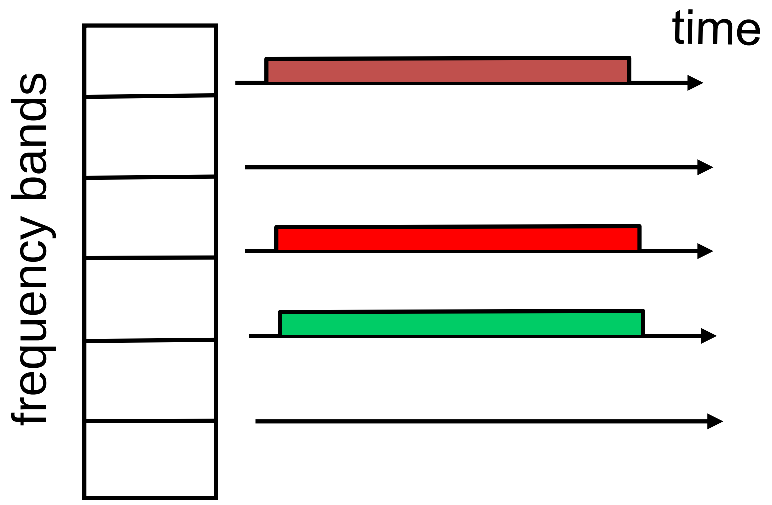
Example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



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 packet to send, 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

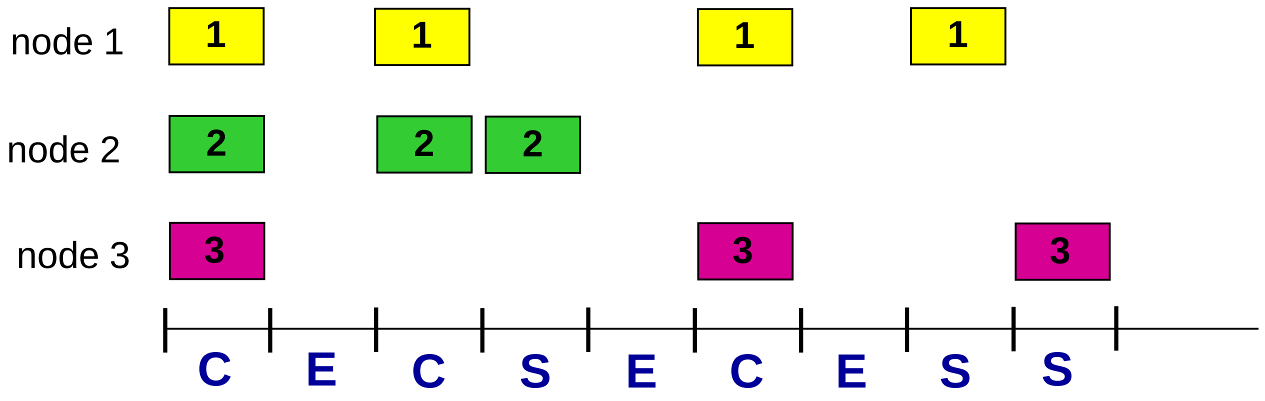
Assumptions:

- All frames same size
- Time 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 probability *p* until success

Slotted ALOHA



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
- Probability that given node has success in a slot $= p(1-p)^{N-1}$
- Probability that *any* node has a success $= Np(1-p)^{N-1}$

- Max efficiency: find p^* that maximizes $Np(1-p)^{N-1}$
- For many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

Max efficiency = 1/e = .37

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

!

Pure (unslotted) ALOHA

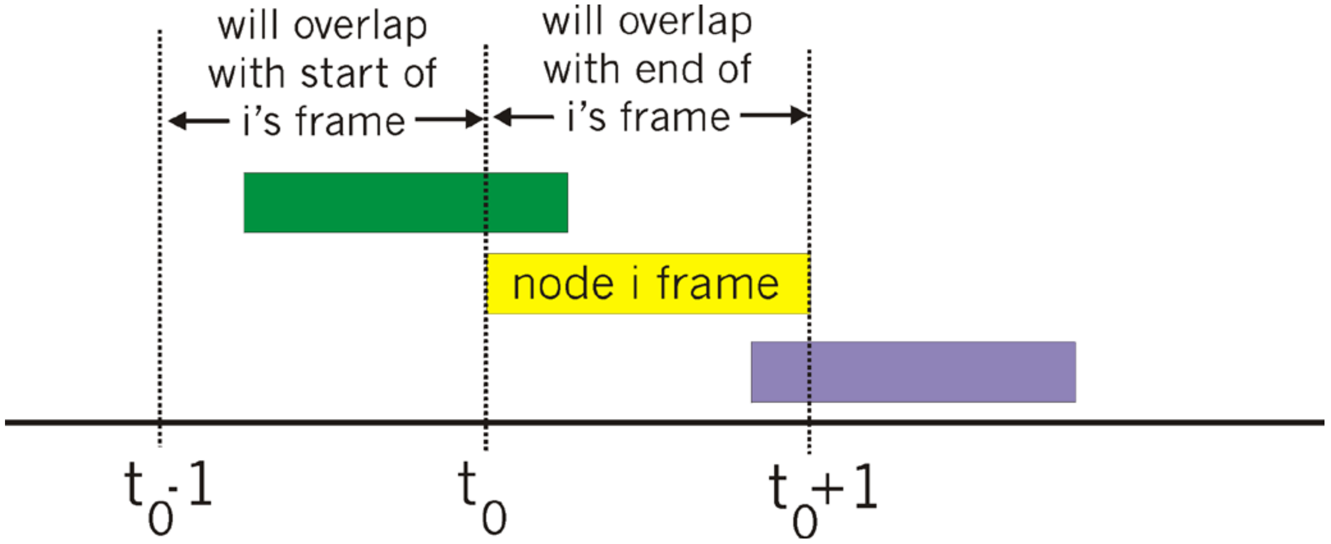
Unslotted Aloha: simpler, no synchronization

When frame first arrives

- Transmit immediately

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 $[t_0-1,t_0]$ ·

P(no other node transmits in $[t_0-1,t_0]$

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

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

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

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

even *worse* than slotted Aloha!

Carrier Sense Multiple Access (CSMA)

CSMA: listen before transmit:

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

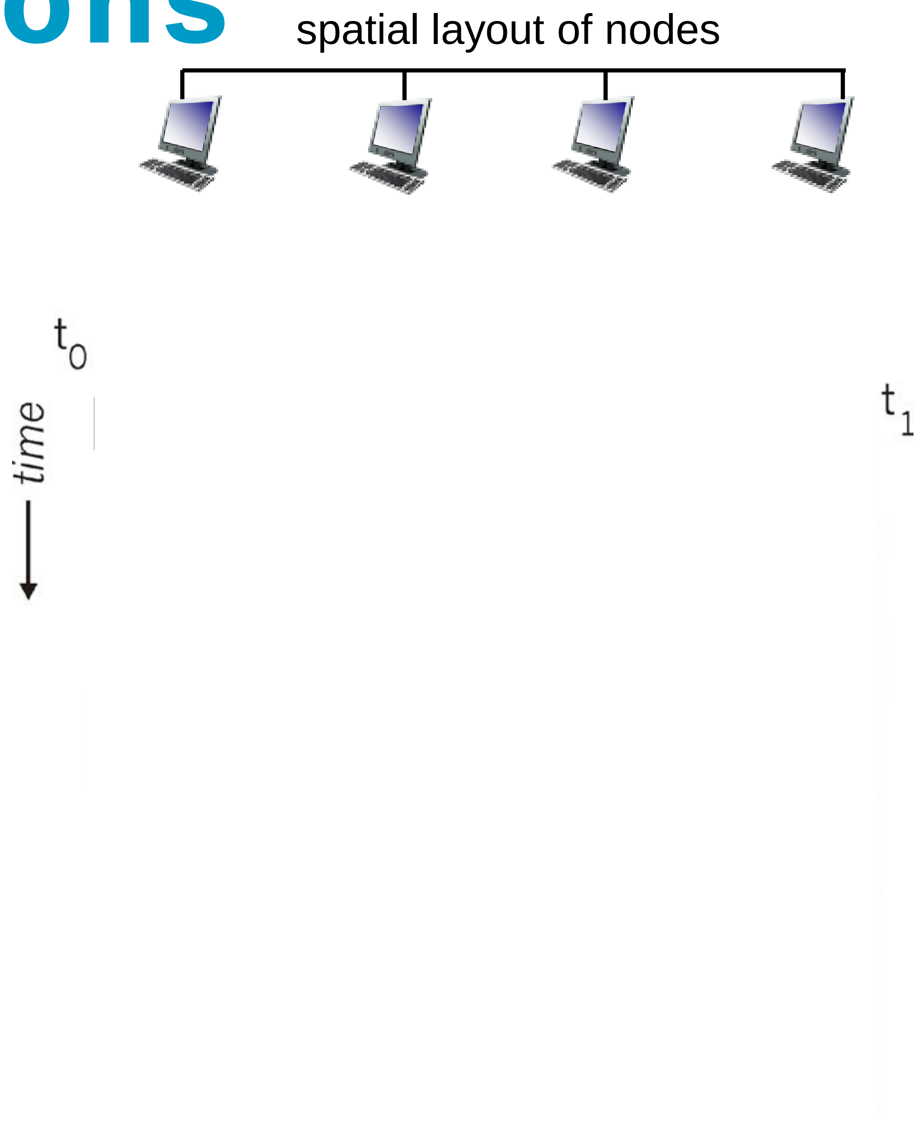
Human analogy: don’t interrupt others!

CSMA collisions

Collisions can still occur: propagation delay means two nodes may not hear each other's transmission

Collision: entire packet transmission time wasted

- Distance & propagation delay play role in in determining collision probability



CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA

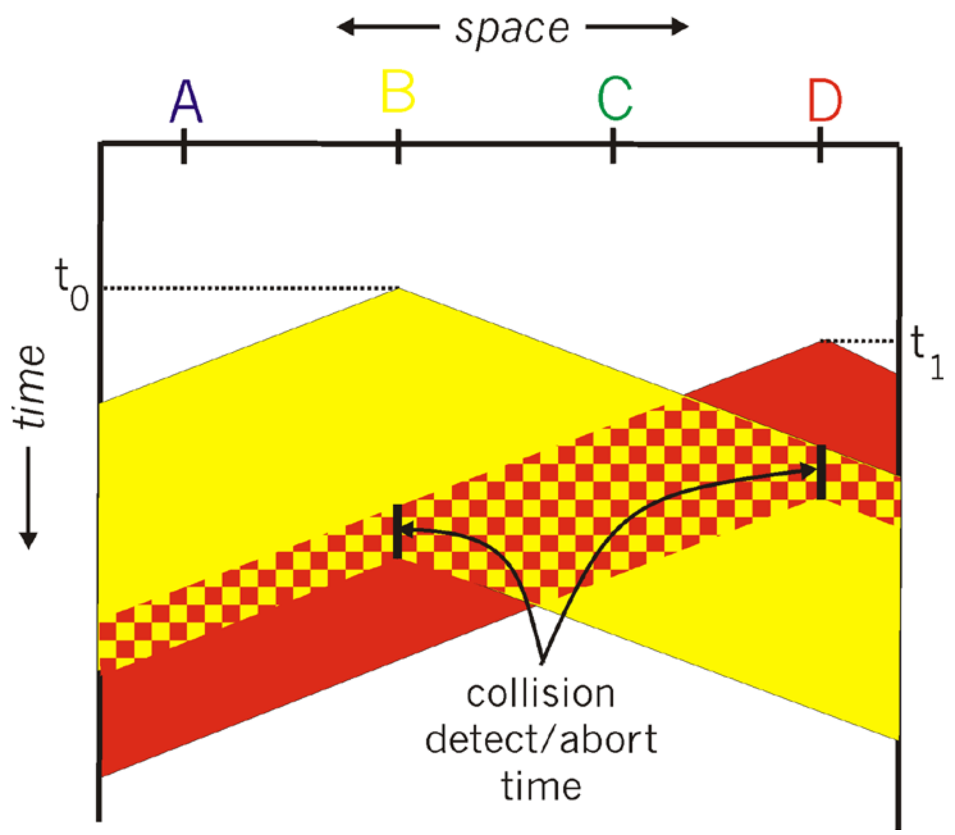
- Collisions *detected* within short time
- Colliding transmissions aborted, reducing channel wastage

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



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

- as t_{prop} goes to 0
- as t_{trans} goes to infinity

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

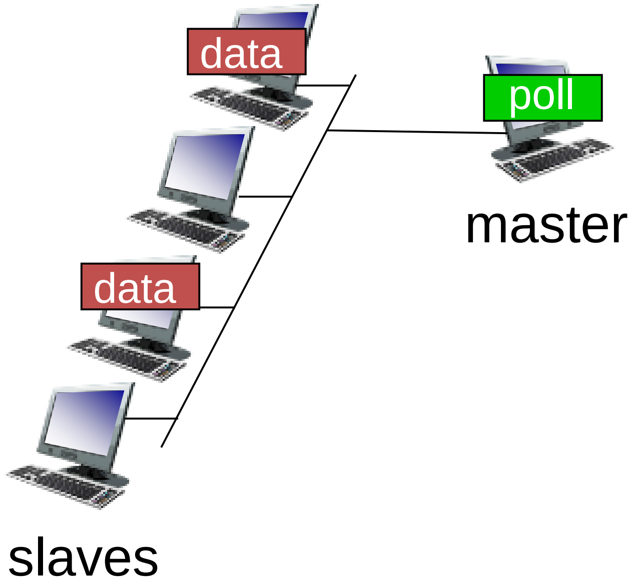
“Taking turns” protocols

- Look for best of both worlds!

“Taking turns” MAC protocols

Polling:

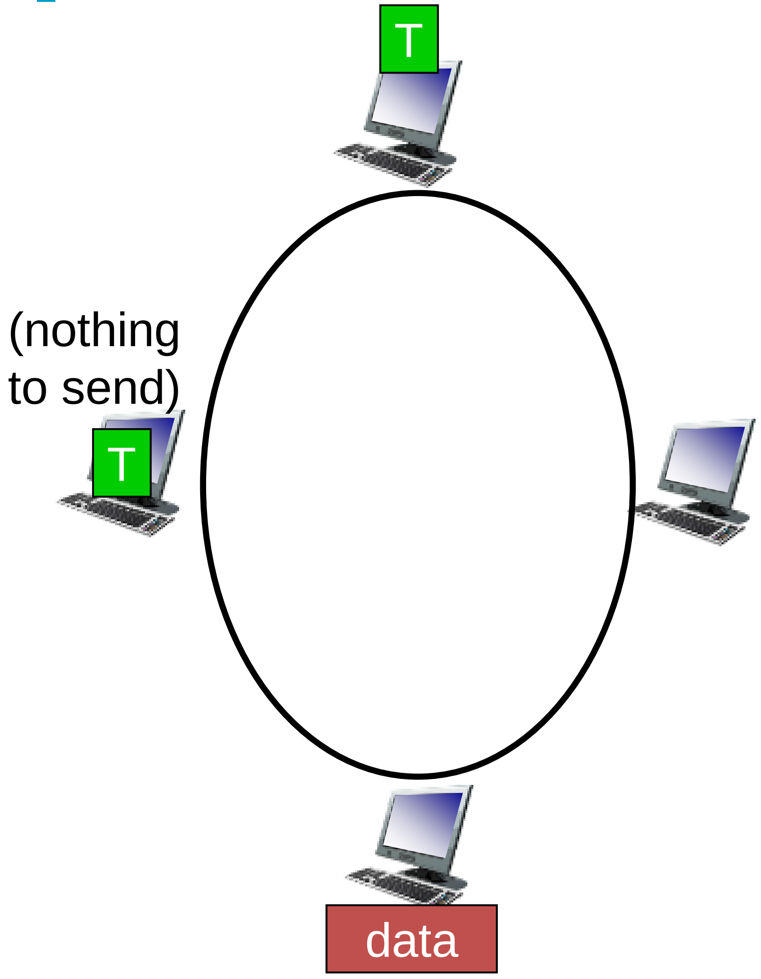
- Master node “invites” slave nodes to transmit in turn
- Typically used with “dumb” slave devices
- Concerns:
  - Polling overhead
  - Latency
  - Single point of failure (master)



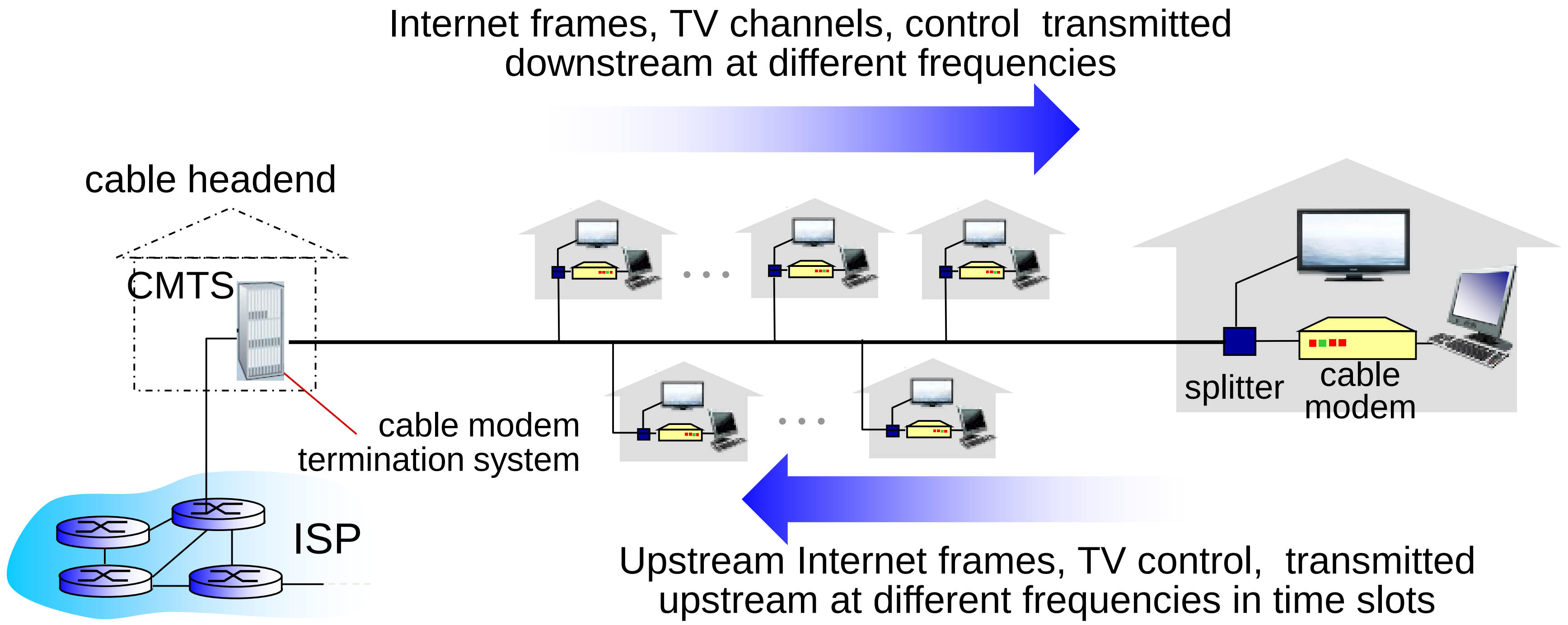
"Taking turns" MAC protocols

Token passing:

- Control *token* passed from one node to next sequentially.
- Token message
- Concerns:
  - Token overhead
  - Latency
  - Single point of failure (token)

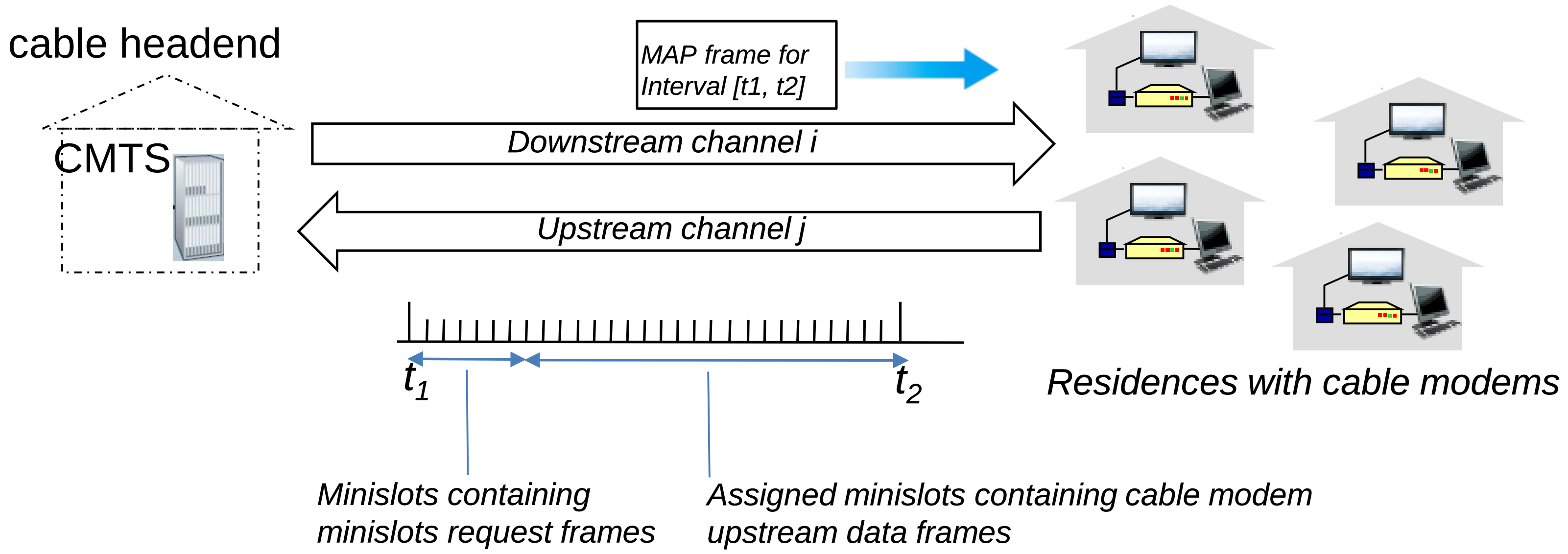


Cable access network


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
t_1
t_2
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

MAC addresses and ARP

32-bit IP address:

- *Network-layer* address for interface
- Used for layer 3 (network layer) forwarding

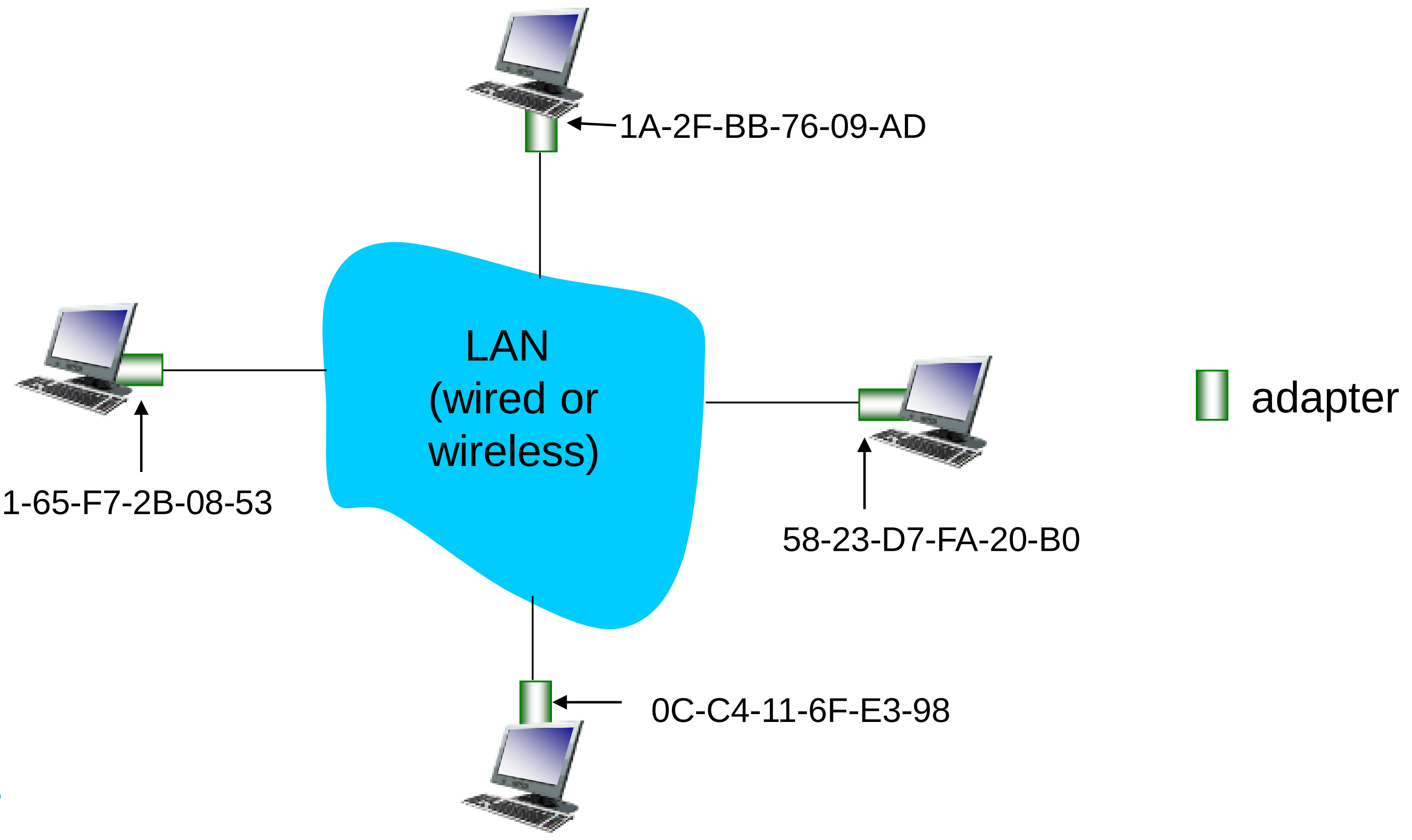
MAC (or LAN or physical or Ethernet) address:

- 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

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

LAN addresses and ARP

Each adapter on LAN has unique *LAN* address



LAN addresses (more)

MAC address allocation administered by IEEE

- Manufacturer buys portion of MAC address space (to assure uniqueness)

Analogy:

- MAC address: like Social Security Number
- IP address: like postal address

MAC flat address → portability

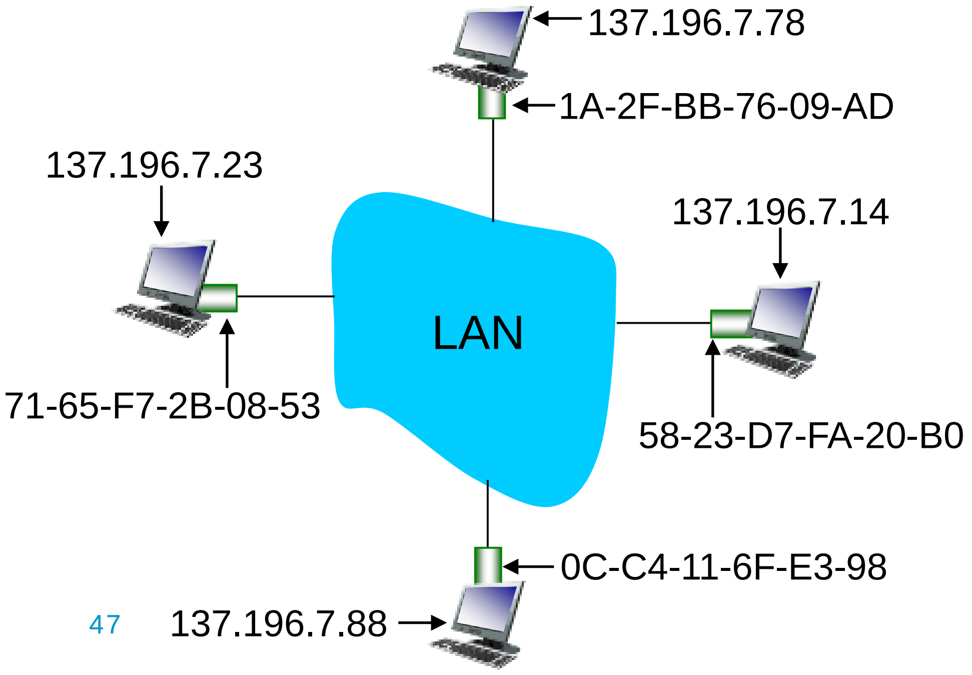
- Can move LAN card from one LAN to another

IP hierarchical address not portable

- Address depends on IP subnet to which node is attached

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 mapsome LAN nodes: < IP address; MAC address; TTL>
- TTL (Time pings for 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

- Destination 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

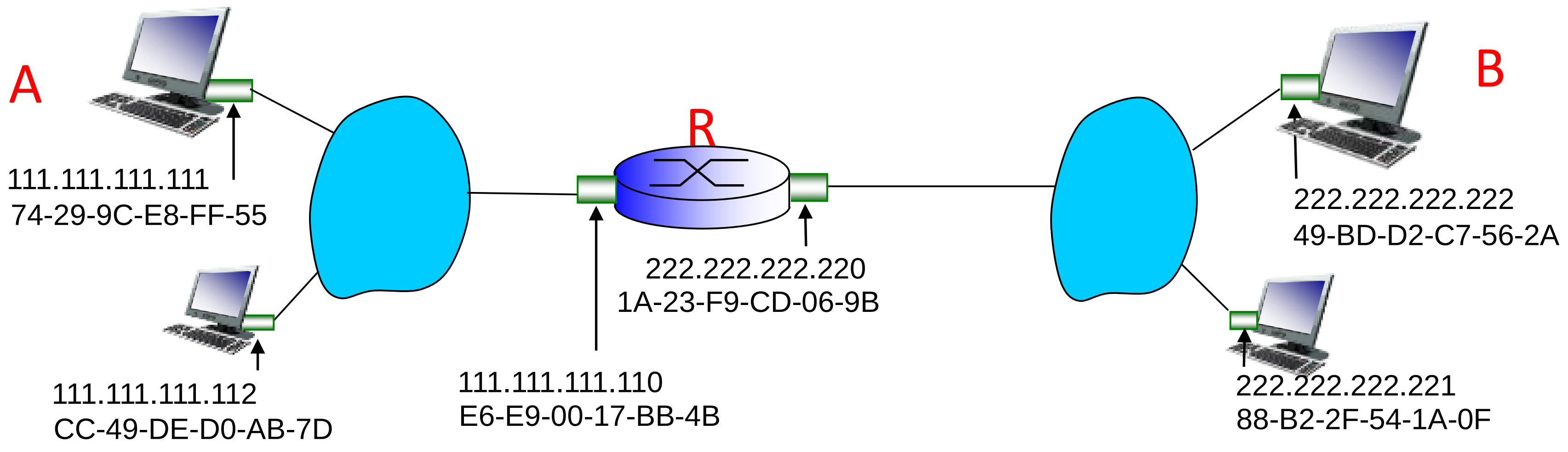
ARP is "plug-and-play":

- Nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

Walkthrough: send datagram from A to B via R

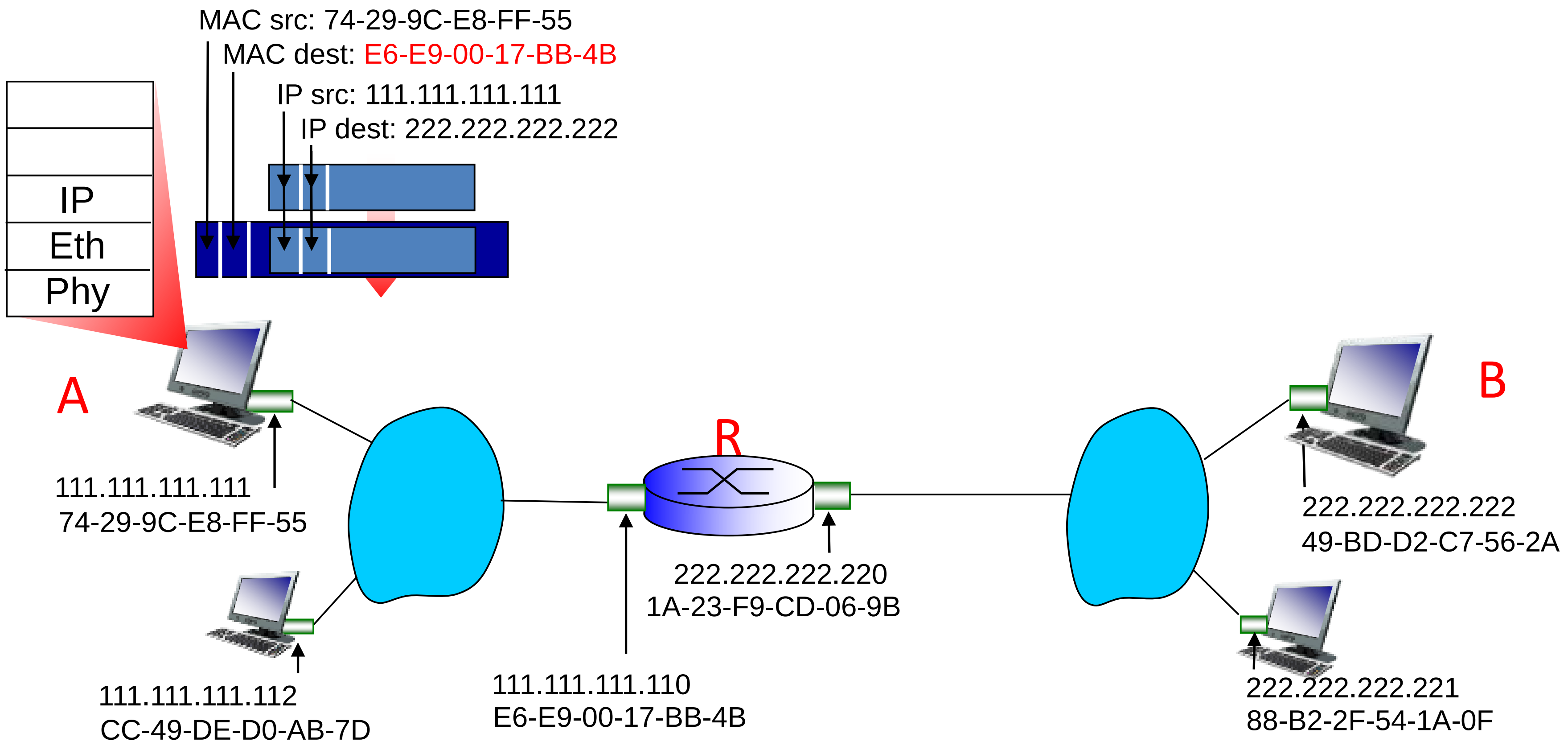
- Focus on addressing – at IP (datagram) and MAC layer (frame)
- Assume A knows B's IP address
- Assume A knows IP address of first hop router, R (how?)
- Assume A knows R's MAC address (how?)



Addressing: routing to another LAN

A creates IP datagram with IP source A, destination B

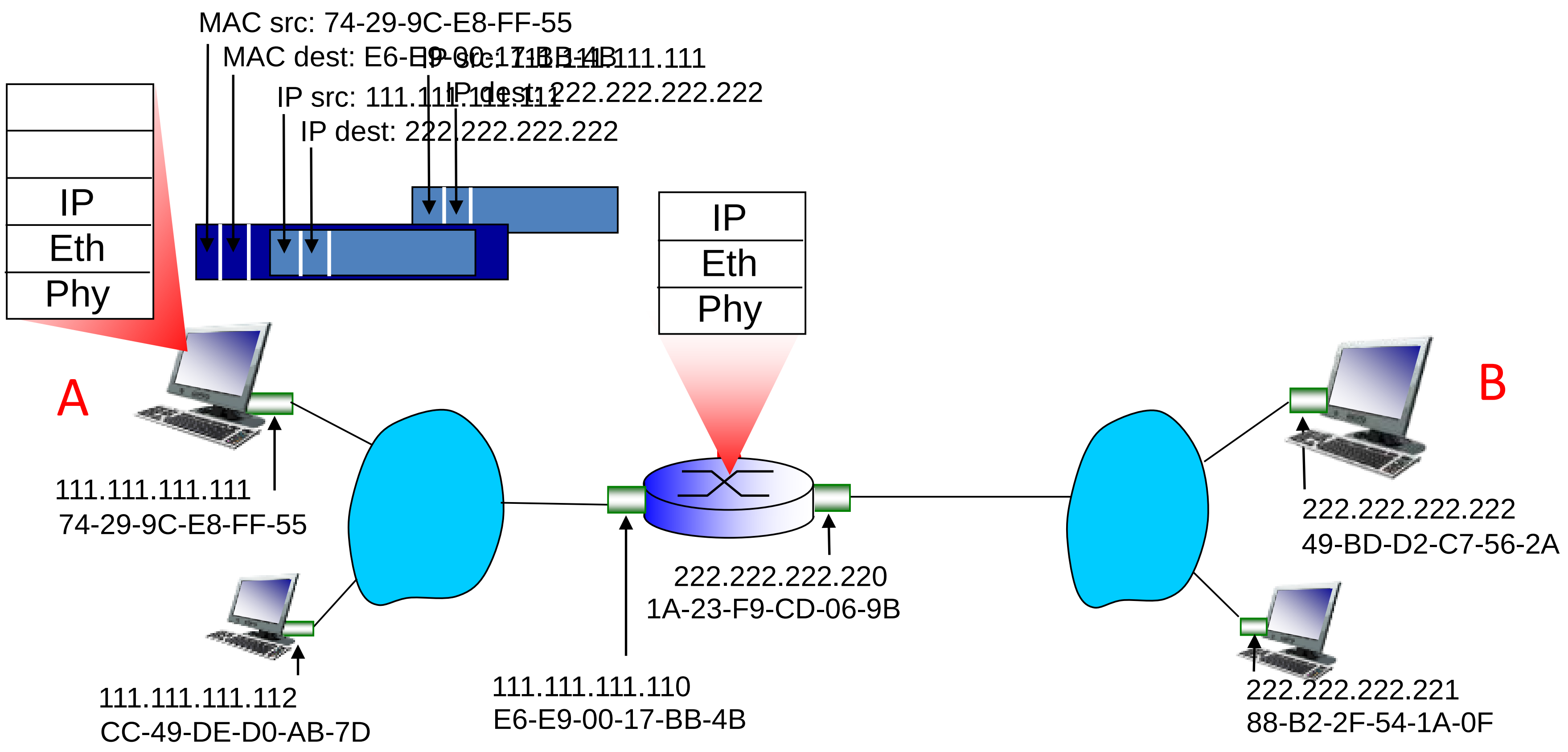
A creates link-layer frame with R's MAC address as destination address, frame contains A-to-B IP datagram



Addressing: routing to another LAN

Frame sent from A to R

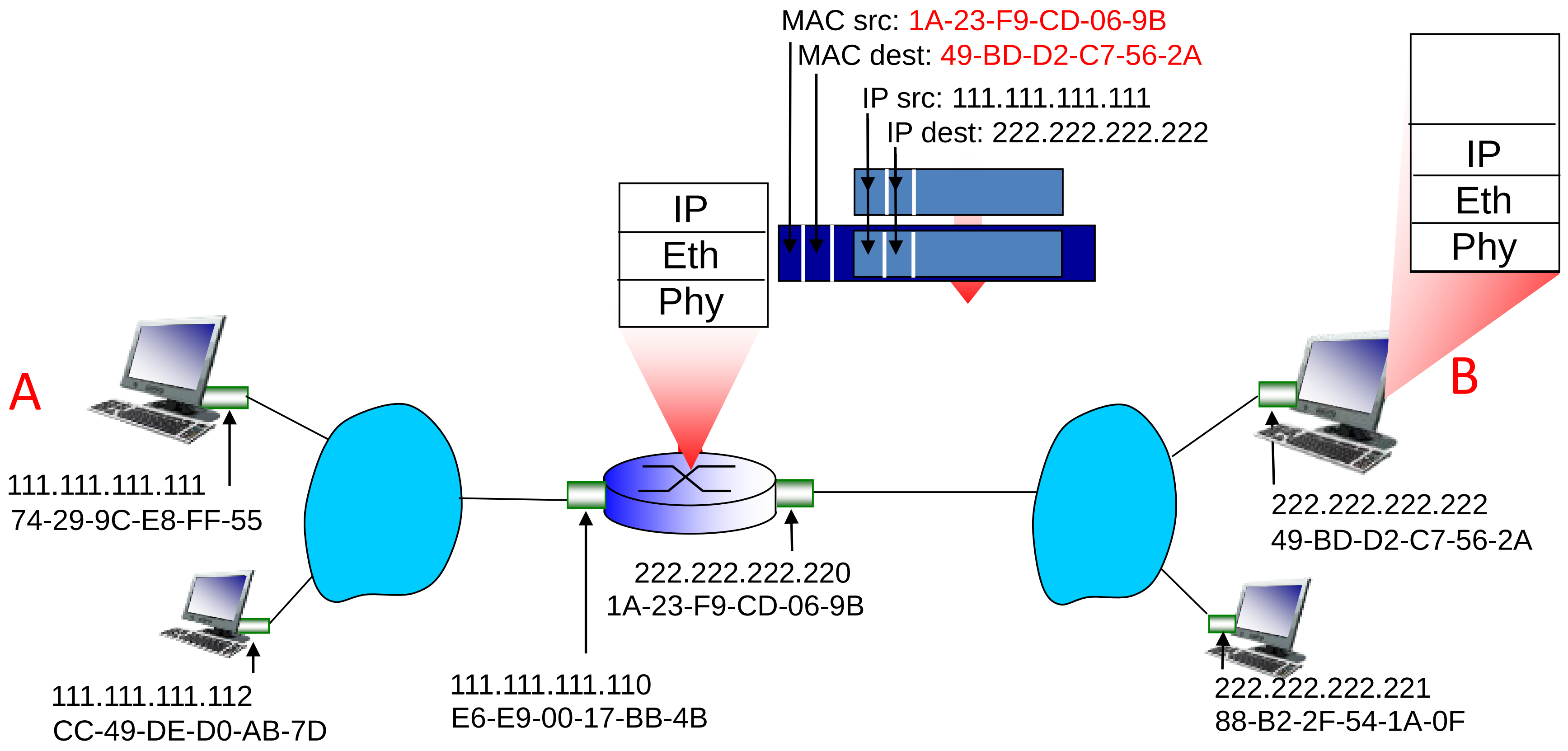
Frame received at R, datagram removed, passed up to IP



Addressing: routing to another LAN

R forwards datagram with IP source A, destination B

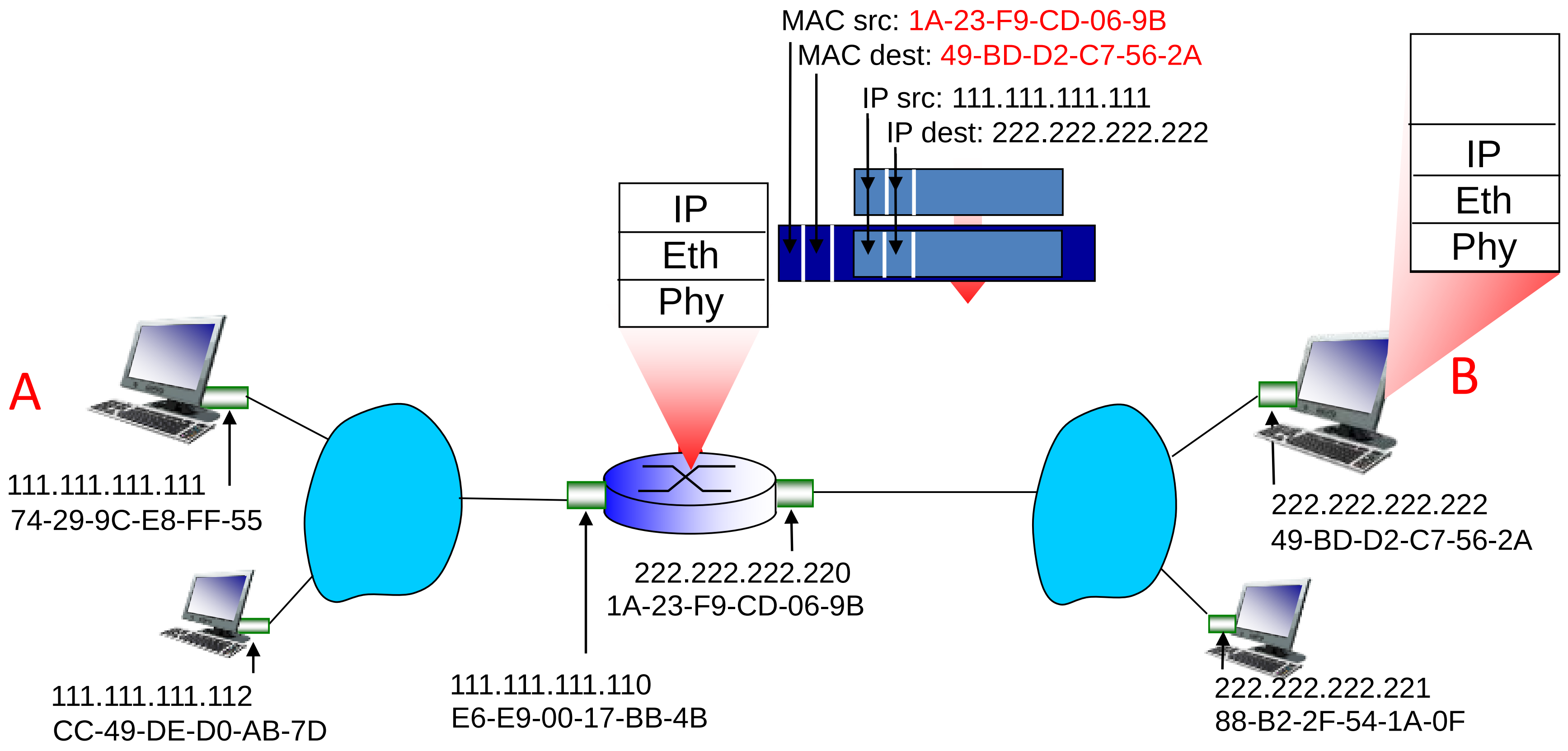
R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram



Addressing: routing to another LAN

R forwards datagram with IP source A, destination B

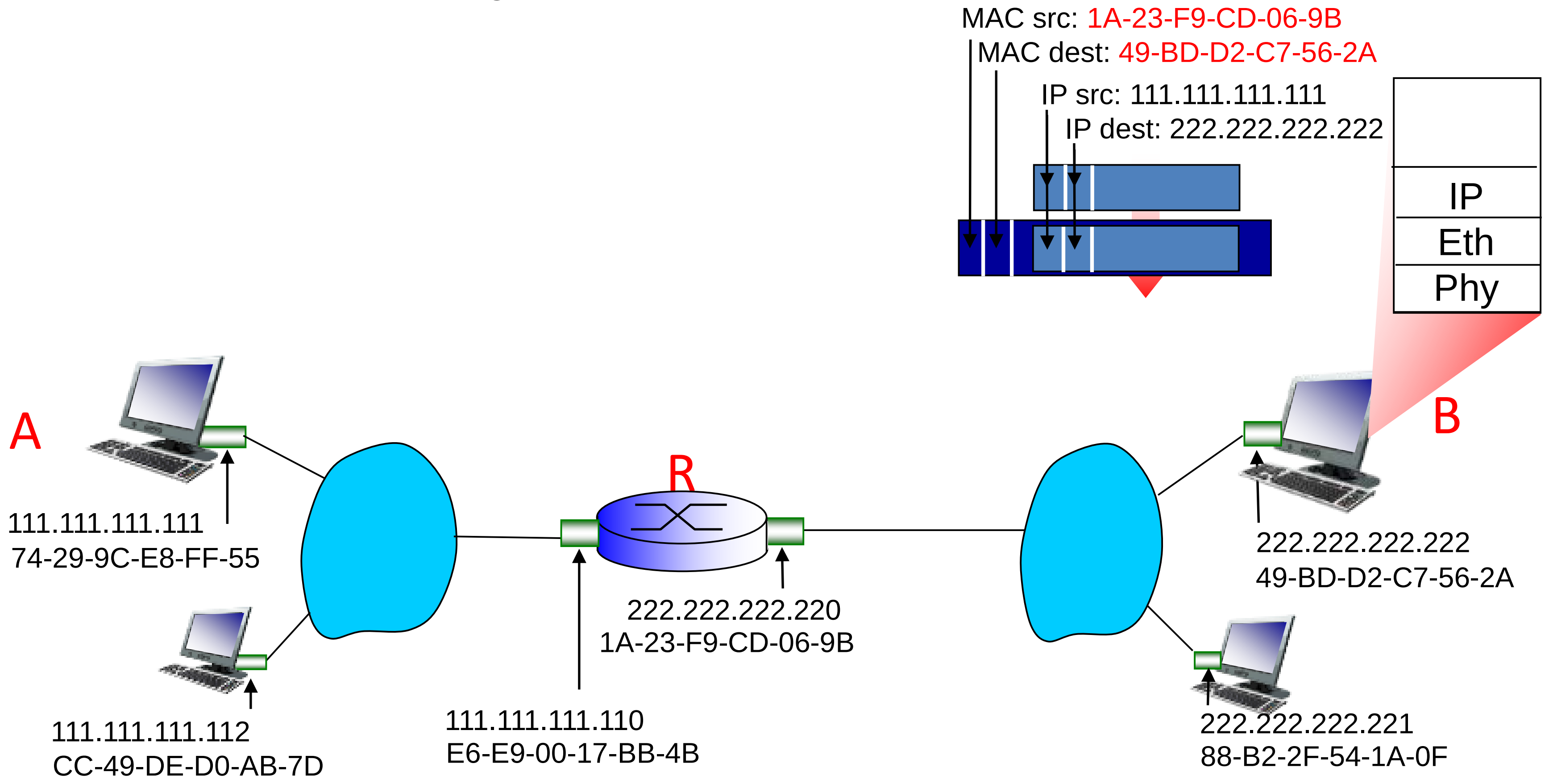
R creates link-layer frame with B's MAC address as destination address, frame contains A-to-B IP datagram



Addressing: routing to another LAN

R forwards datagram with IP source A, destination B

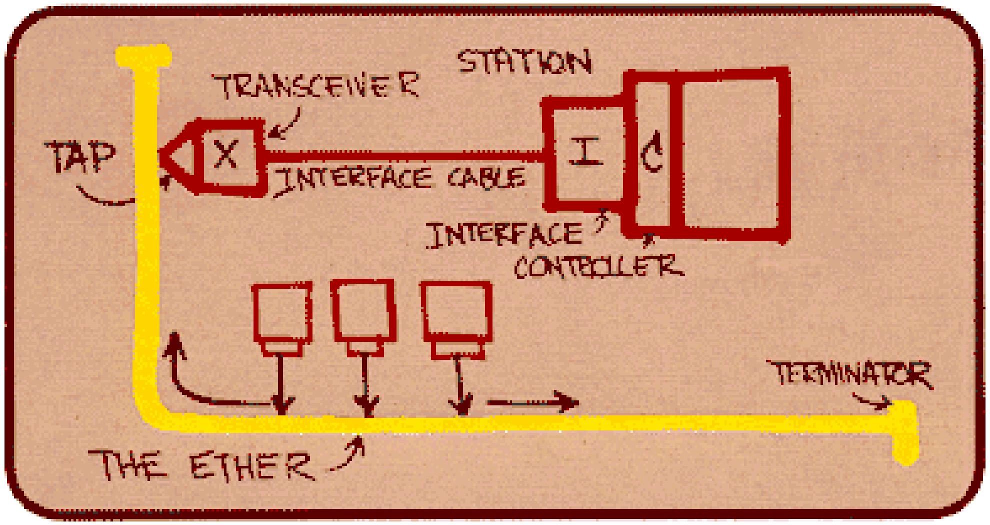
R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Ethernet

"Dominant" wired LAN technology:

- Single chip, multiple speeds (e.g., Broadcom BCM5761)
- First widely used LAN technology
- Simpler, cheap
- Kept up with speed race: 10 Mbps – 10 Gbps



Metcalfe's Ethernet sketch

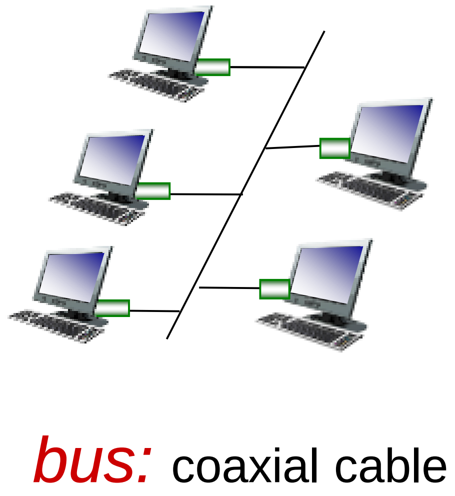
Ethernet: physical topology

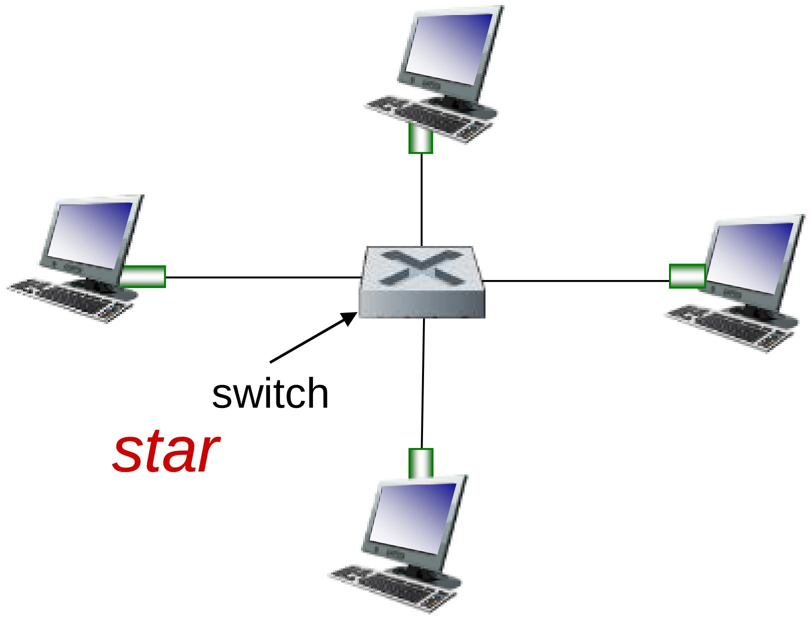
Bus: popular through mid 90s

- All nodes in same collision domain (can collide with each other)

Star: prevails today

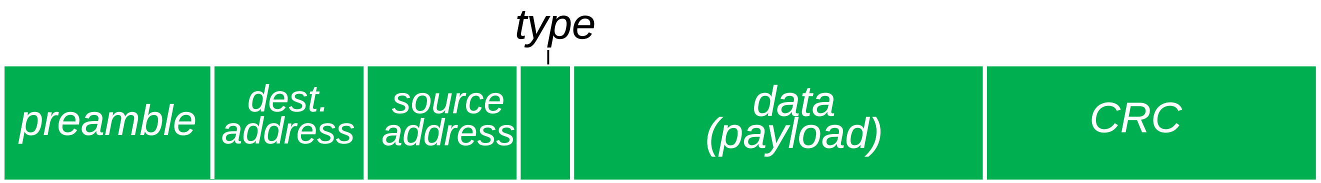
- Active switch in center
- Each "spoke" runs a (separate) Ethernet protocol (nodes do not collide with each other)





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)

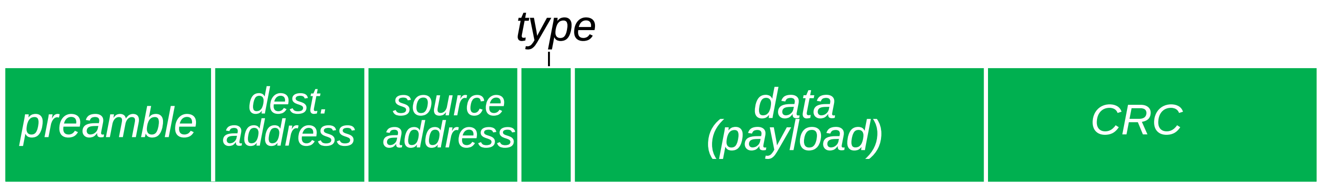
Addresses: 6 byte source, destination MAC addresses

- 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

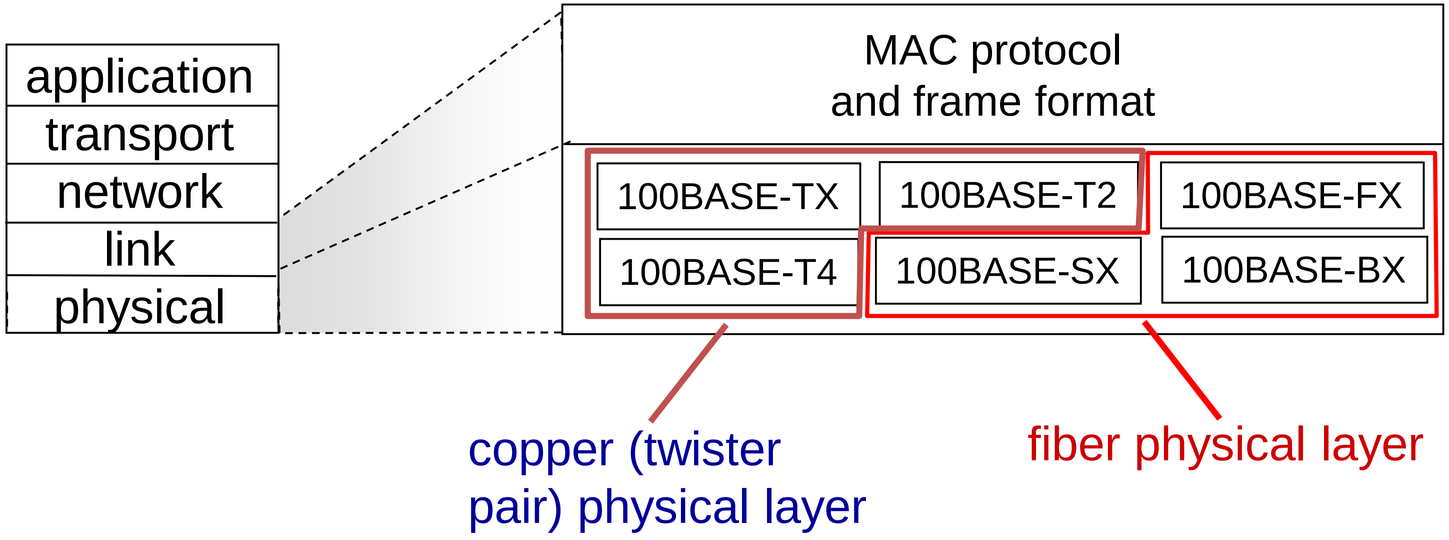
- Data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost

Ethernet’s MAC protocol: unslotted CSMA/CD with binary backoff

802.3 Ethernet standards

Many different Ethernet standards

- Common MAC protocol and frame format
- Different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10 Gbps, 40 Gbps
- Different physical layer media: fiber, cable



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

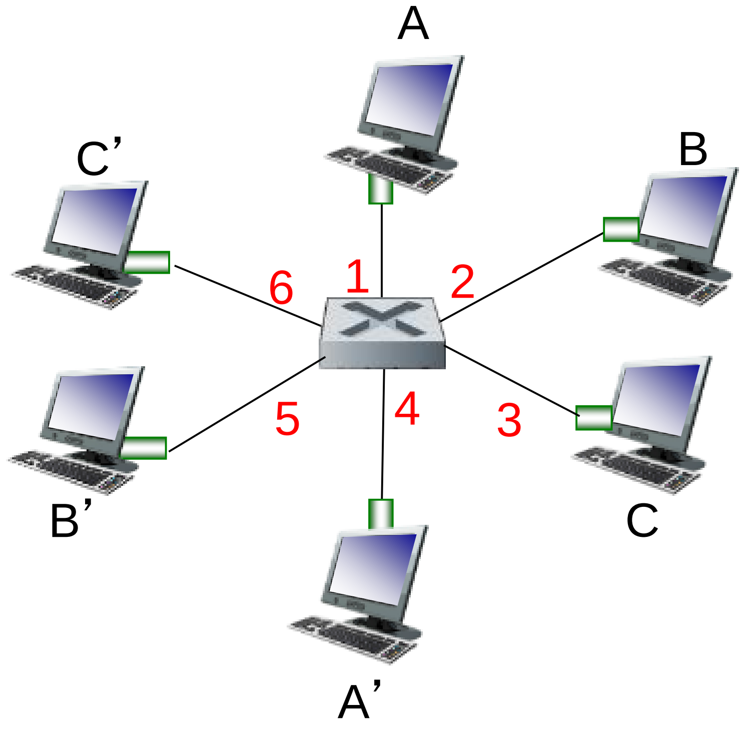
- Switches do not need to be configured

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

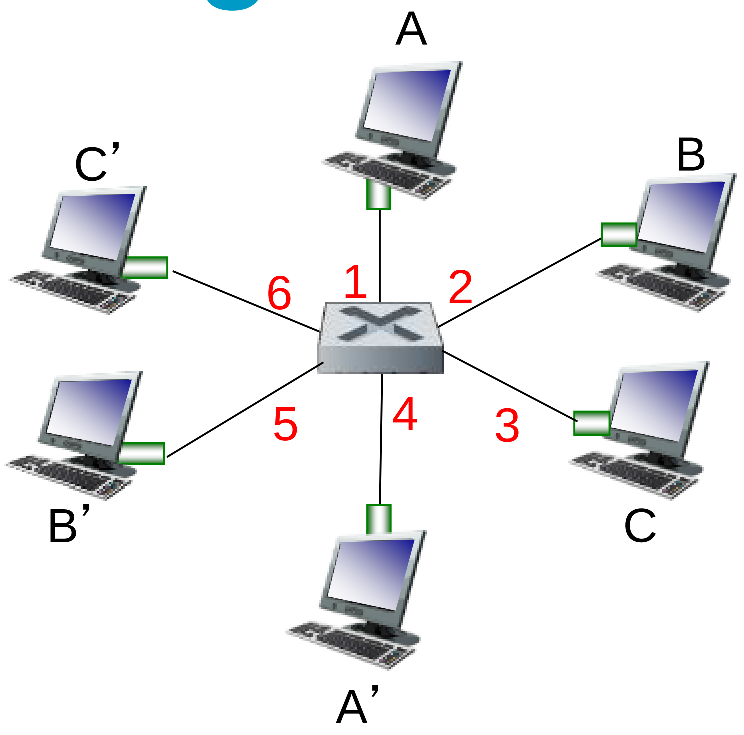
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

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?

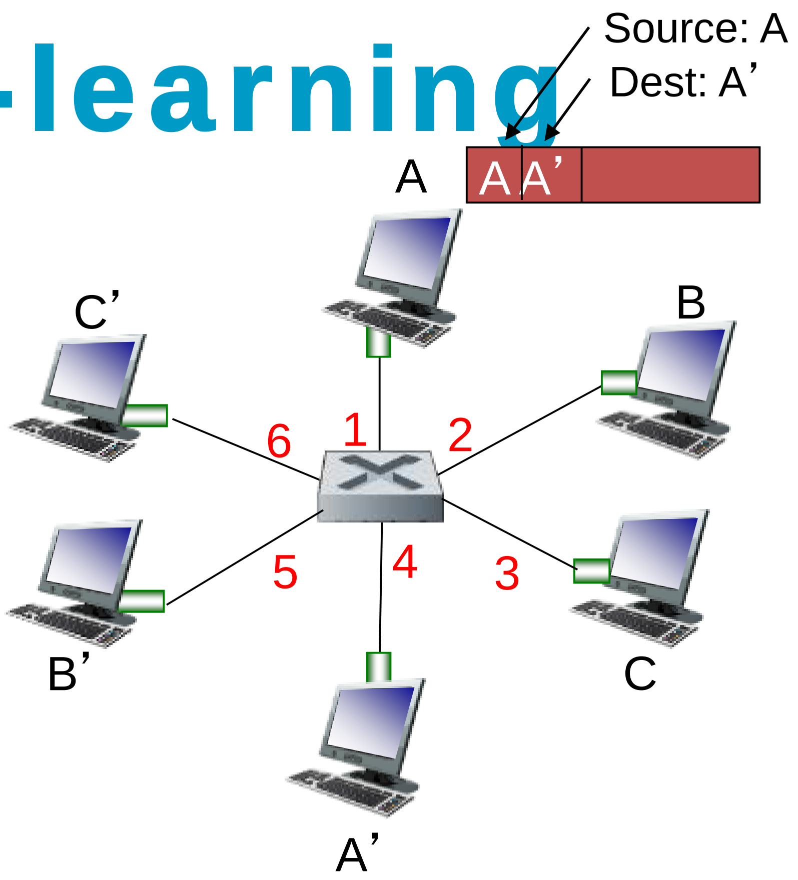


switch with six interfaces *(1,2,3,4,5,6)*

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*

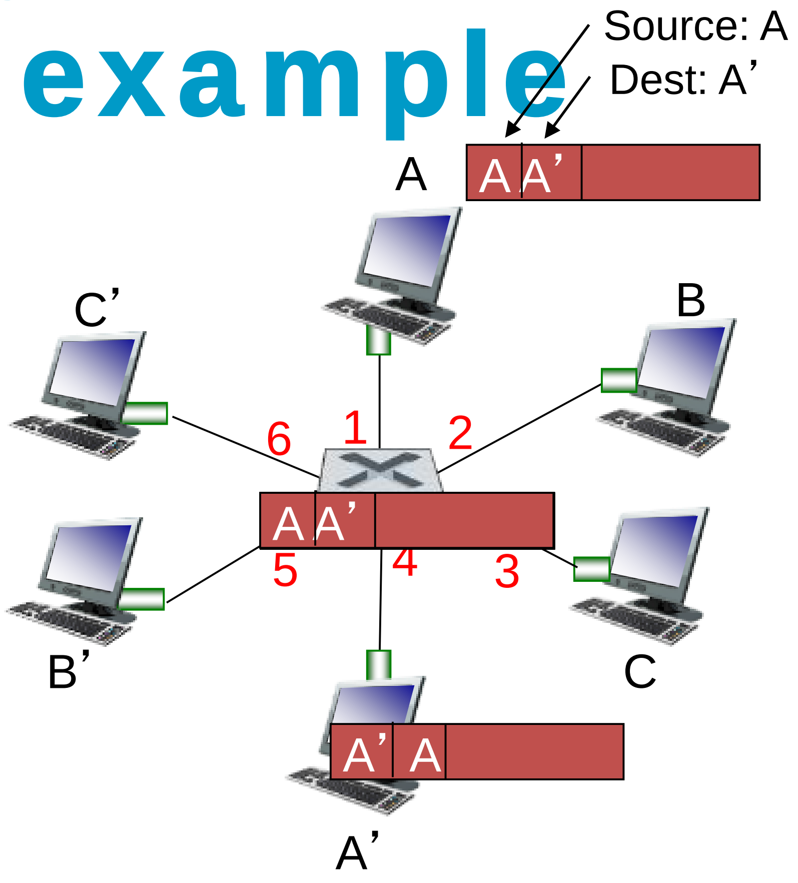
Switch table (initially empty)

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:

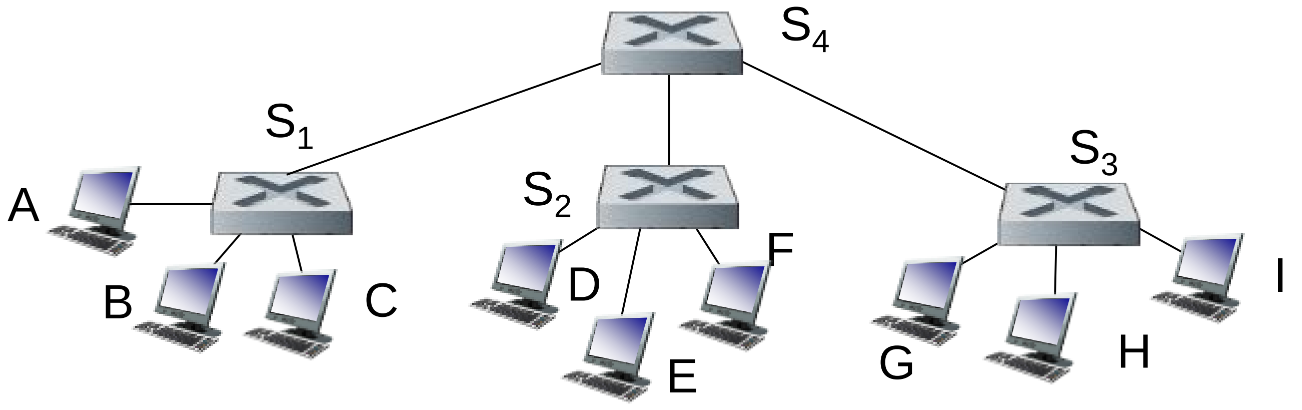
selectively send on just one link

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

switch table (initially empty)

Interconnecting switches

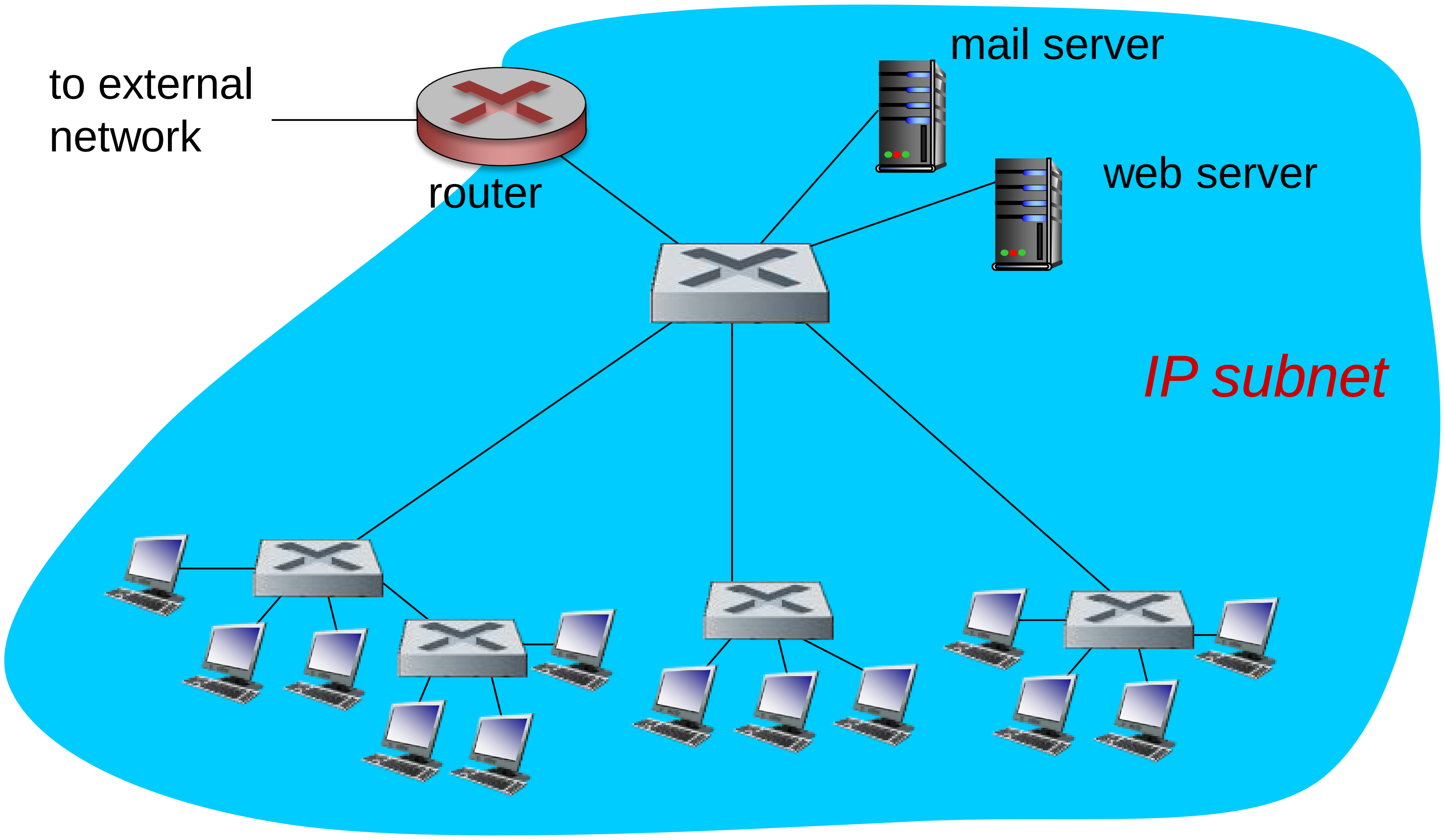
Self-learning switches can be connected together:



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!)

Institutional network



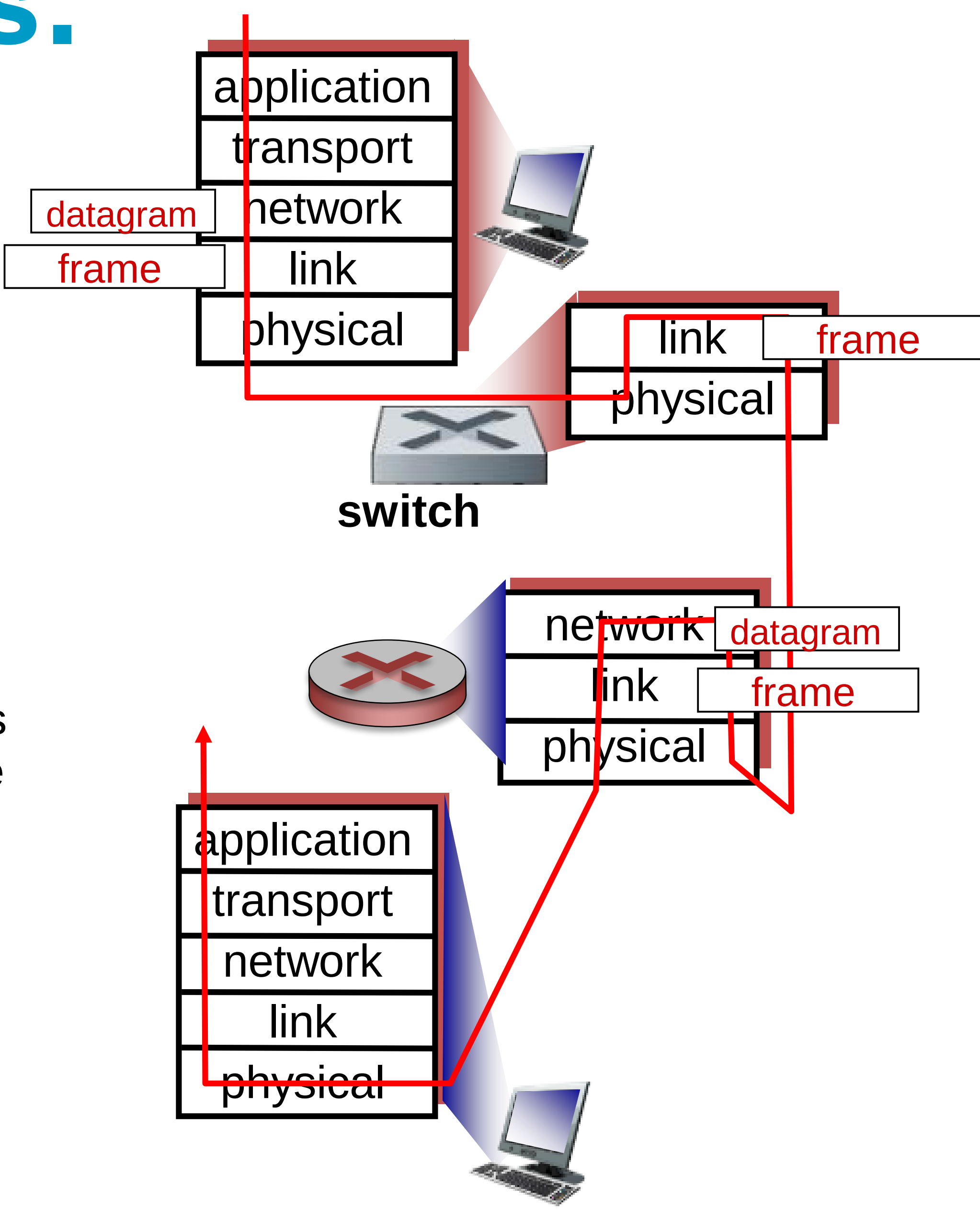
Switches vs. Routers

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



THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND
ENGINEERING

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:

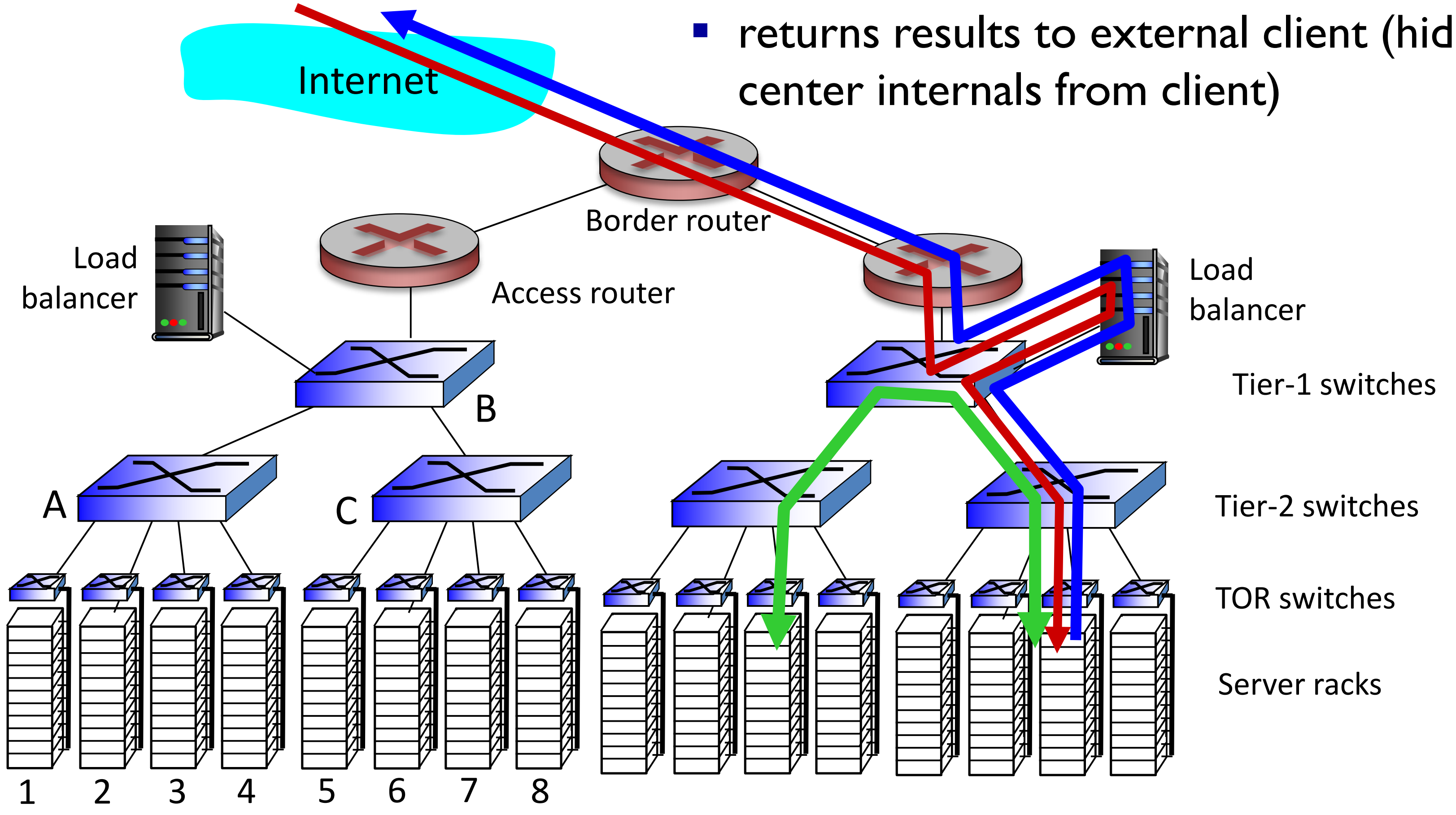
- Multiple applications, each serving massive numbers of clients
- Managing/Balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Data center networks

load balancer: application-layer routing

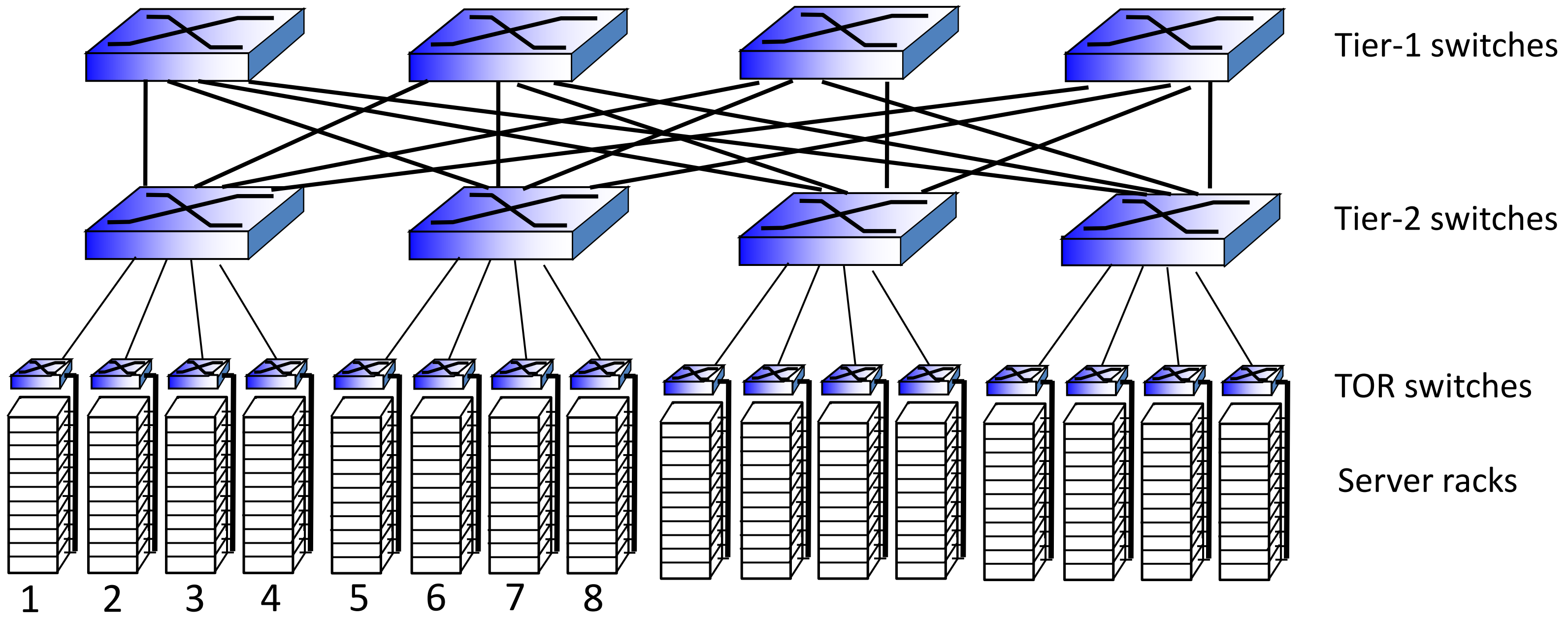
- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



Data center networks

Rich interconnection among switches, racks:

- Increased throughput between racks (multiple routing paths possible)
- Increased reliability via redundancy



Synthesis: a day in the life of a web request

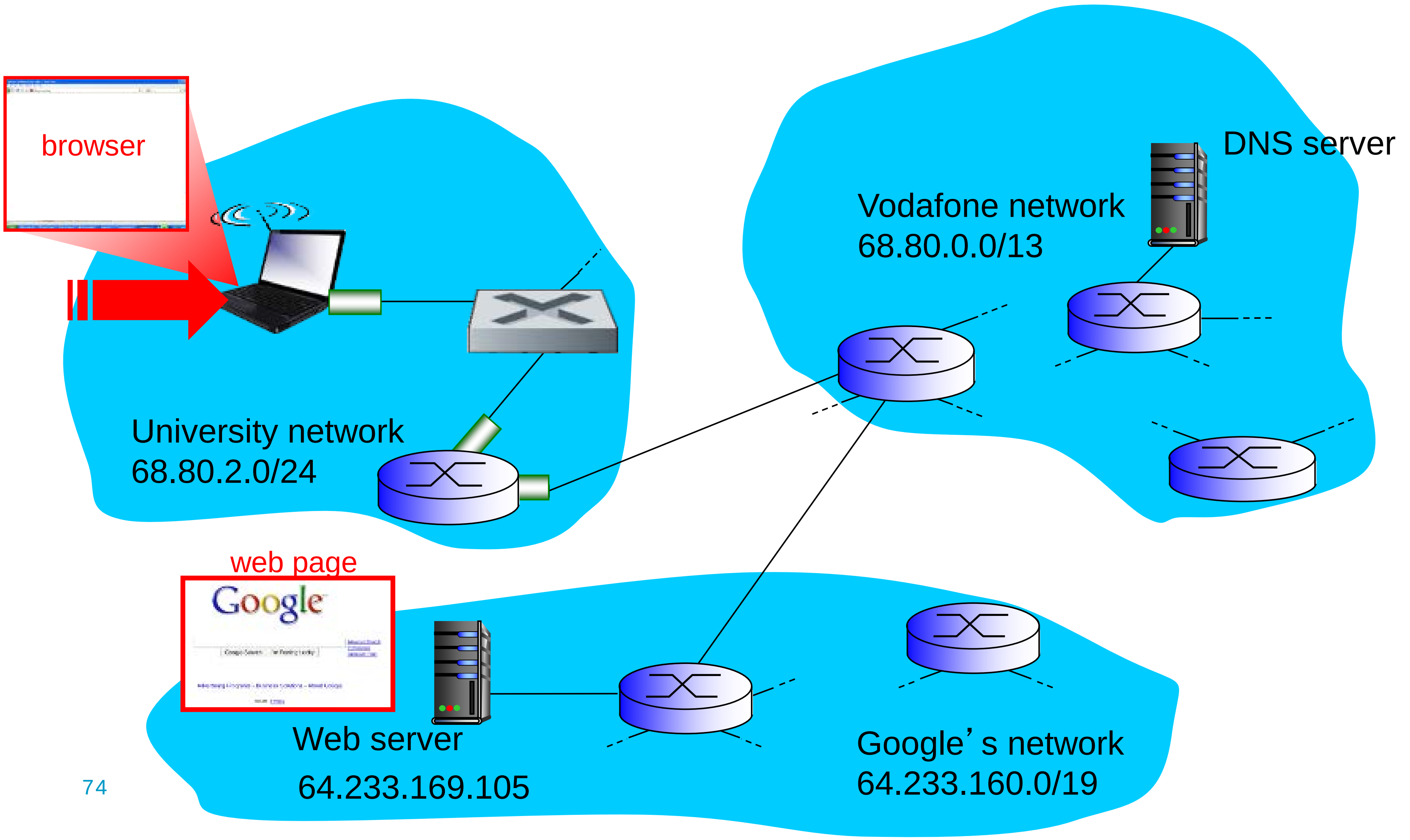
Journey down protocol stack complete!

- Application, transport, network, link

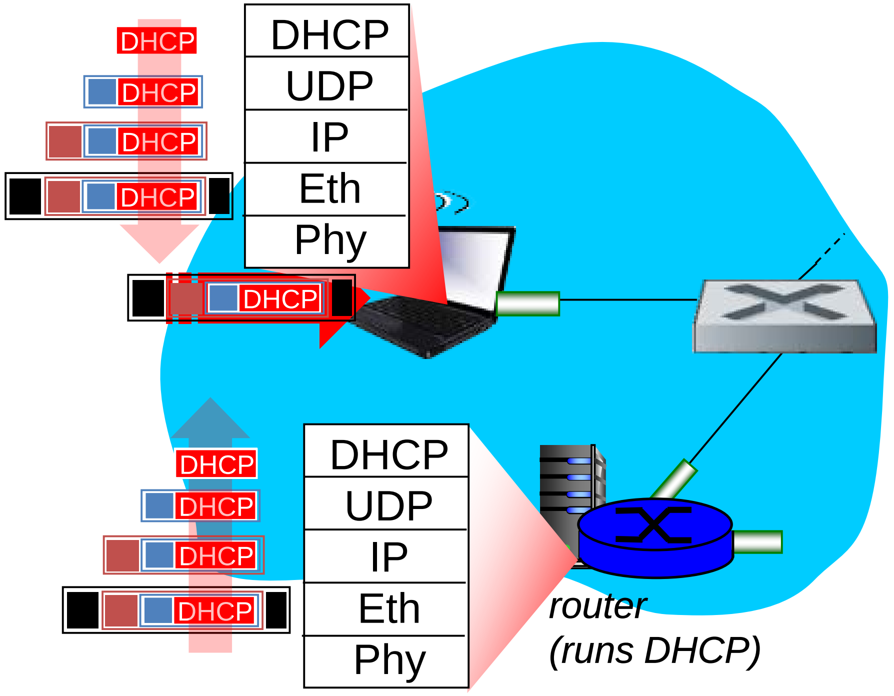
Putting-it-all-together: synthesis!

- *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

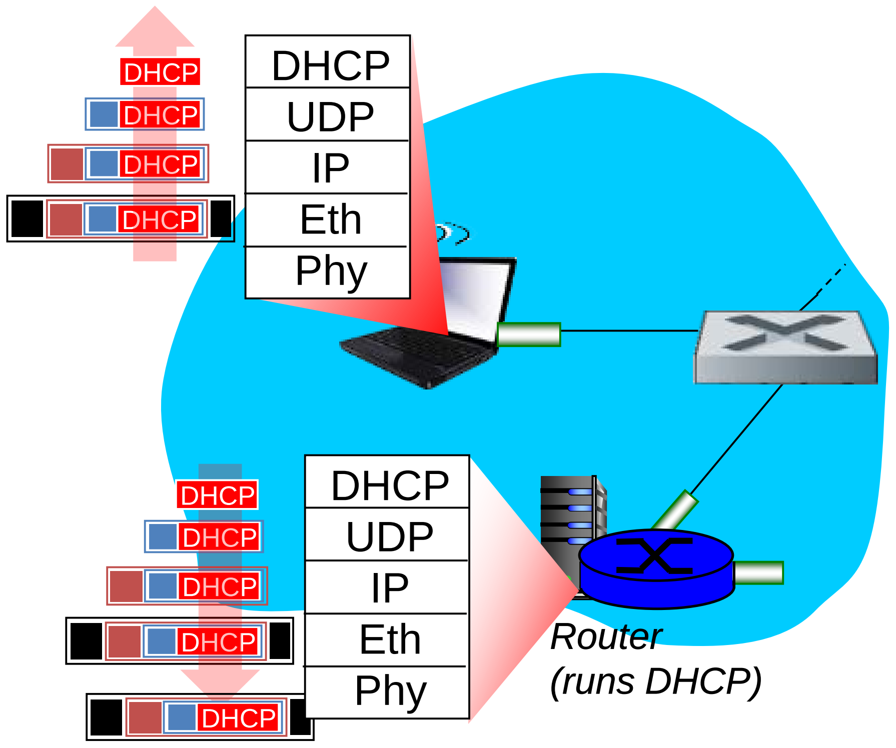


Connecting to the Internet



- 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

Connecting to the Internet



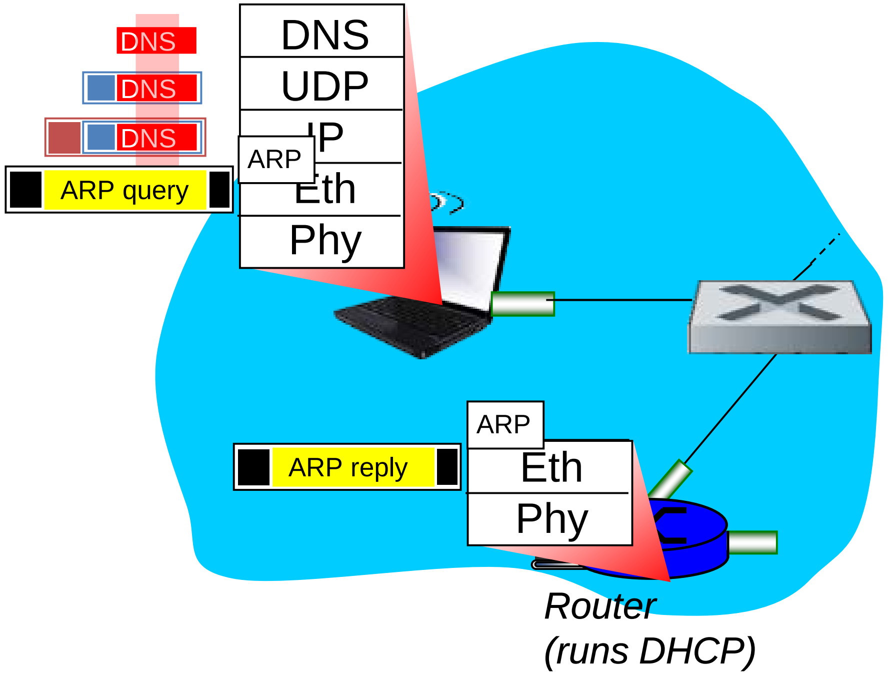
- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- 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

ARP (before DNS, before HTTP)



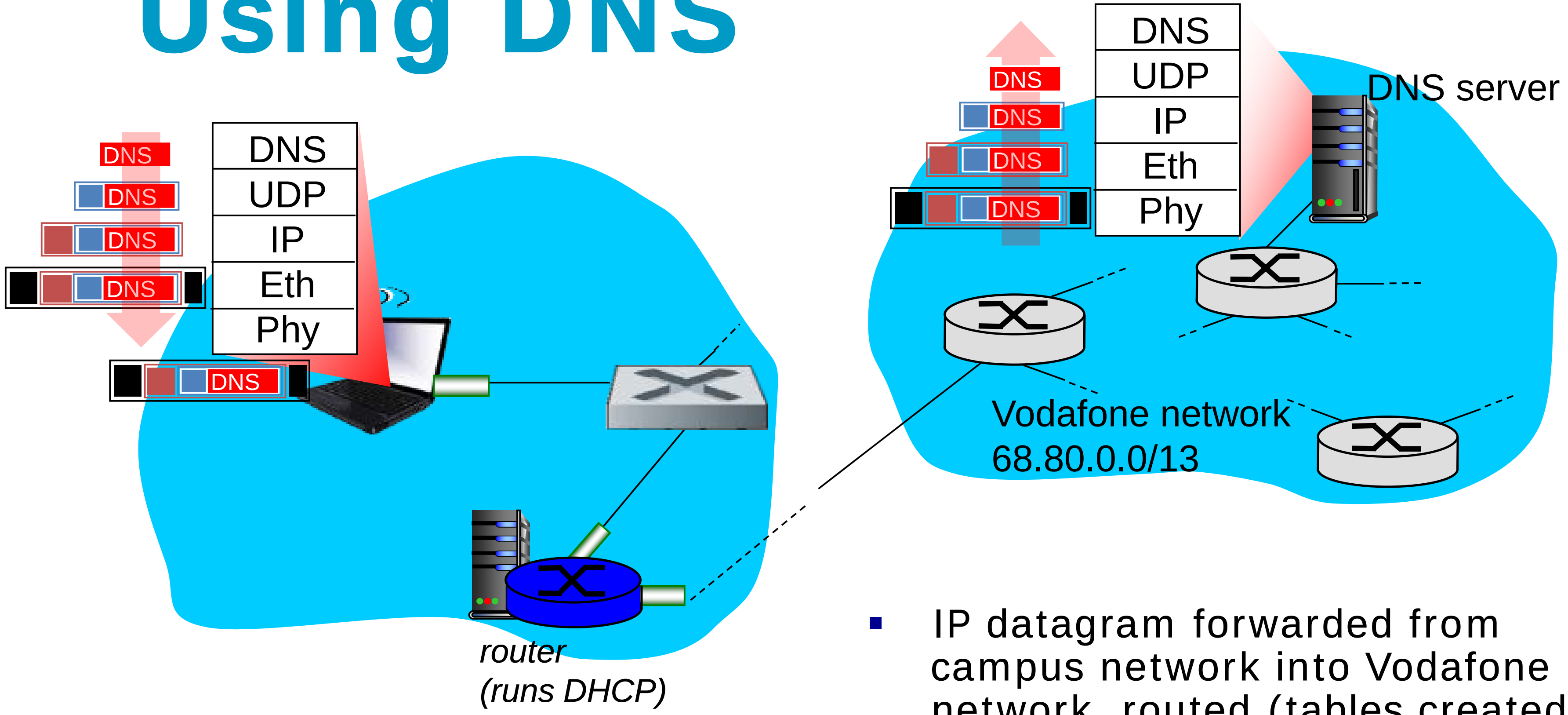
- Before sending *HTTP* request, need IP address of www.google.com: *DNS*

- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet. 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

Using DNS

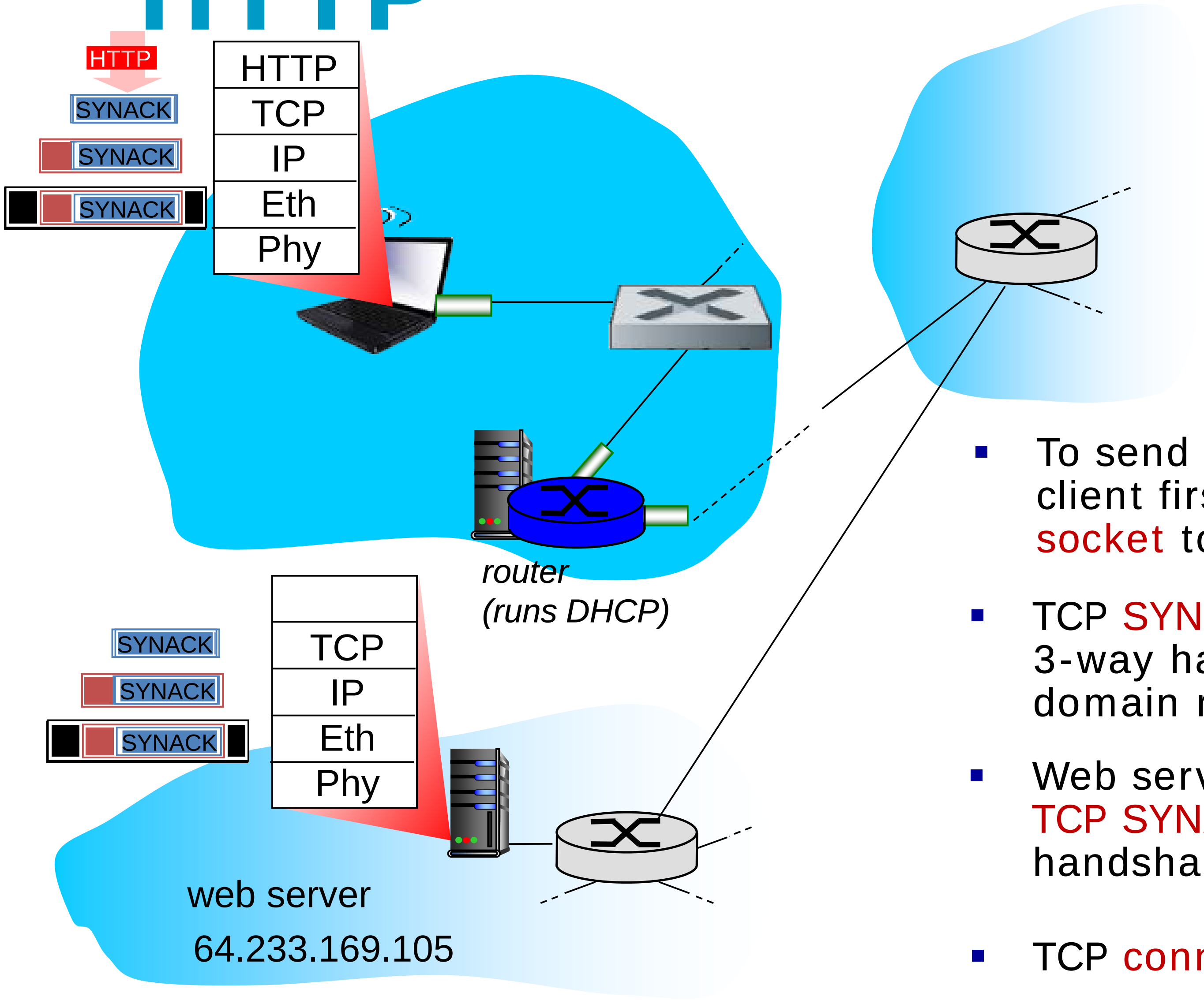


- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

- IP datagram forwarded from campus network into Vodafone network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server

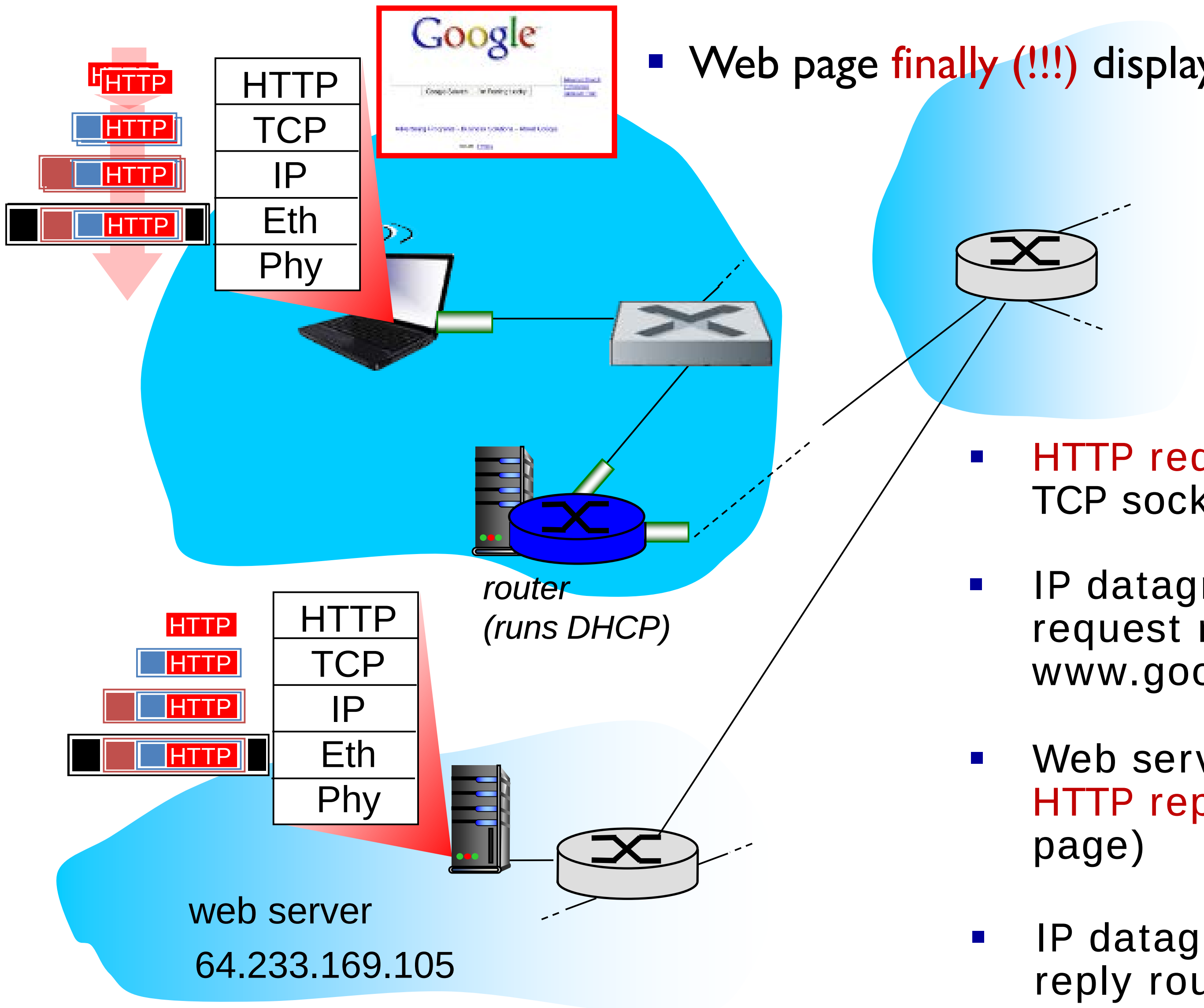
- Demuxed to DNS server
- DNS server replies to client with IP address of www.google.com

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!

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

Summary

Described the link layer and its purpose

Described the services the link layer provides

Introduced some ways of detecting and correcting errors: Parity bits, Internet checksums and Cyclic Redundancy Checks

Channel partitioning, by time, frequency or code

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

Learnt how LANs work, including: MAC Addresses, Address Resolution Protocol, Ethernet protocols and Switches

Learnt how switches are self-learning

Introduced data centres and their unique challenges

Synthesized our knowledge on how a web request is served

ENGINEERING