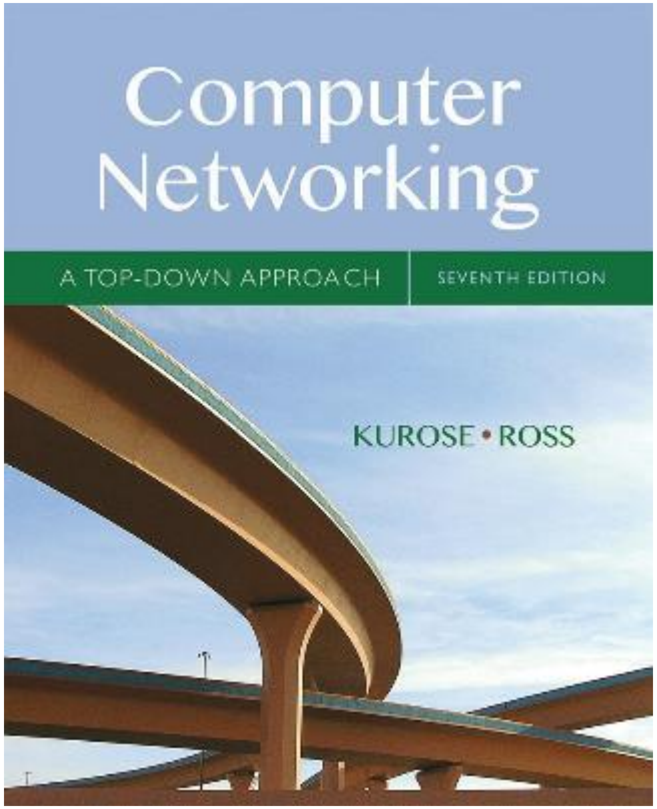
Chapter 6 The Link Layer and LANs



Computer Networking: A Top Down Approach

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

Chapter 6: Link layer and LANs

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

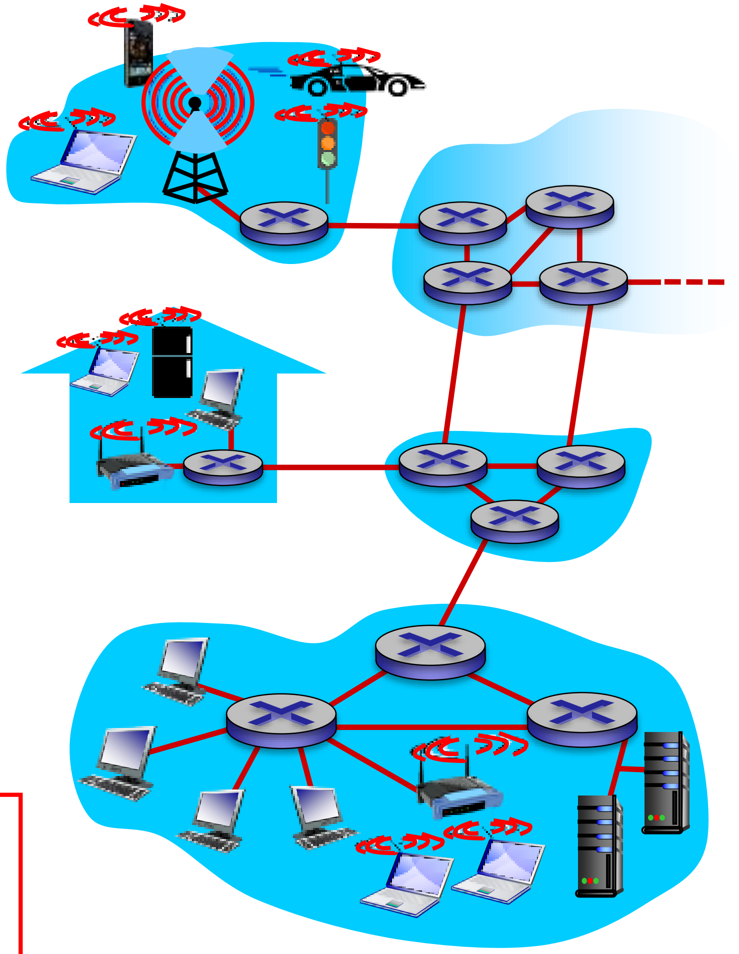
6.6 data center networking

6.7 a day in the life of a web request

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

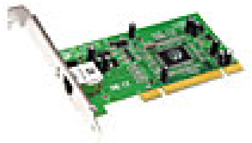
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, destination
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates

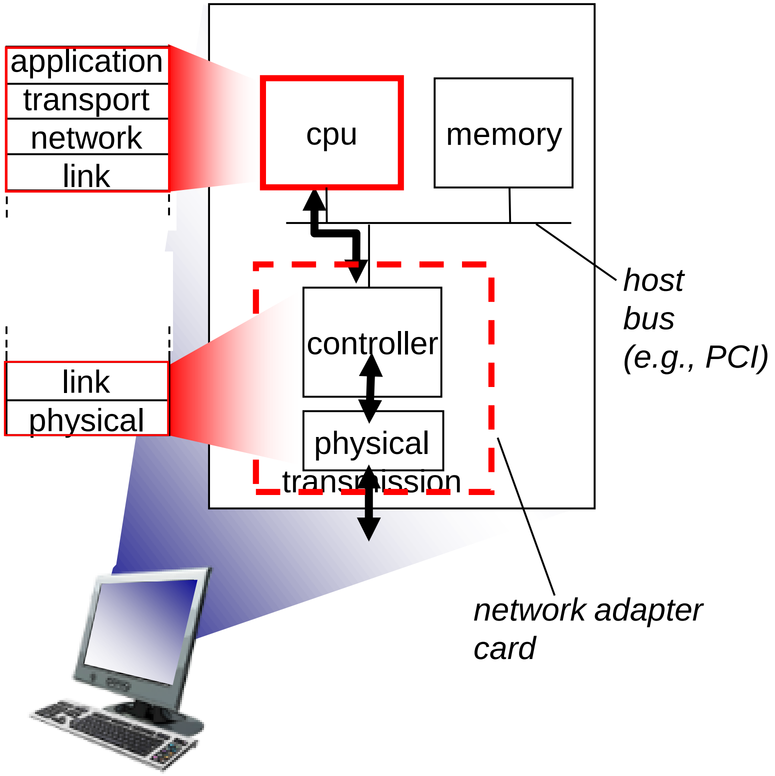
Link layer services (more)

- *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

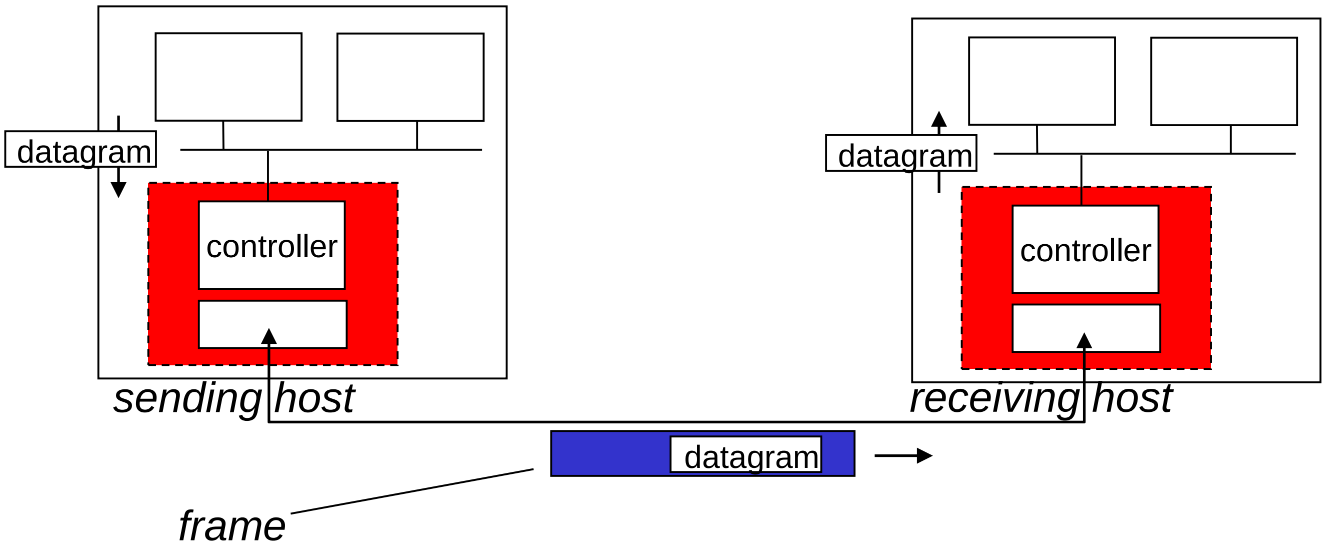
Where is the link layer implemented?

- 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

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

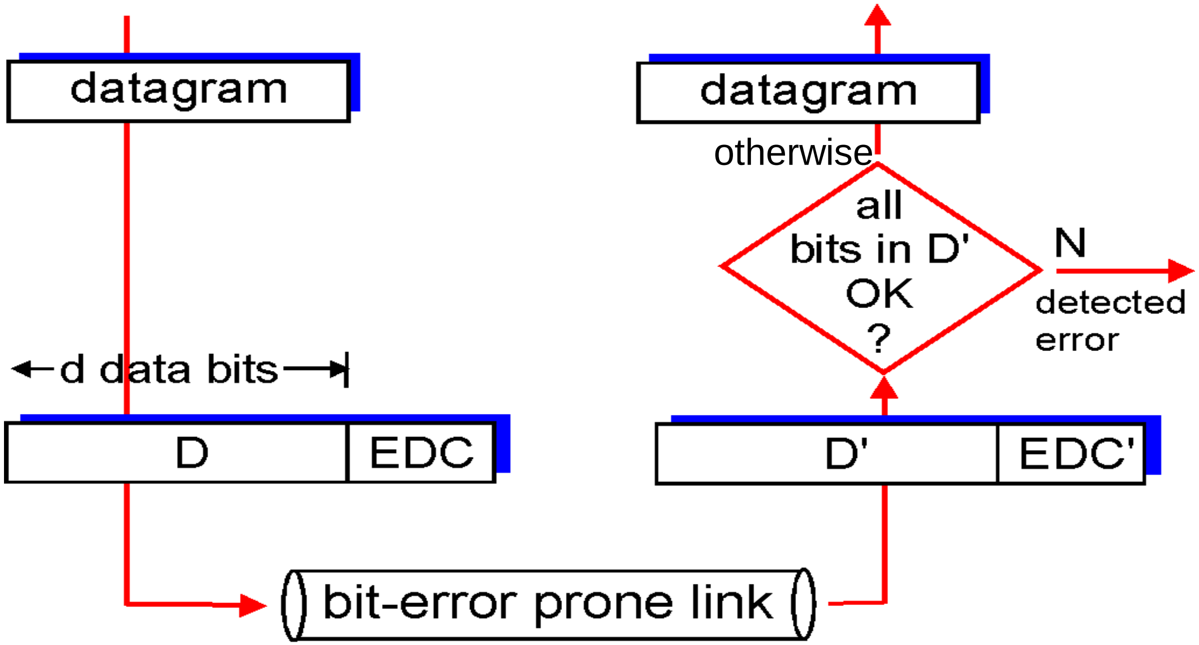
6.6 data center networking

6.7 a day in the life of a web request

Error detection

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

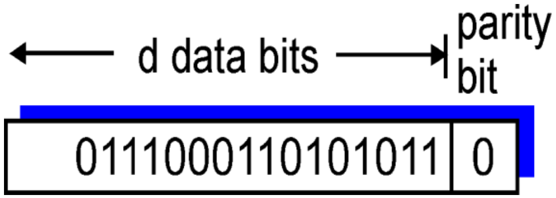
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Parity checking

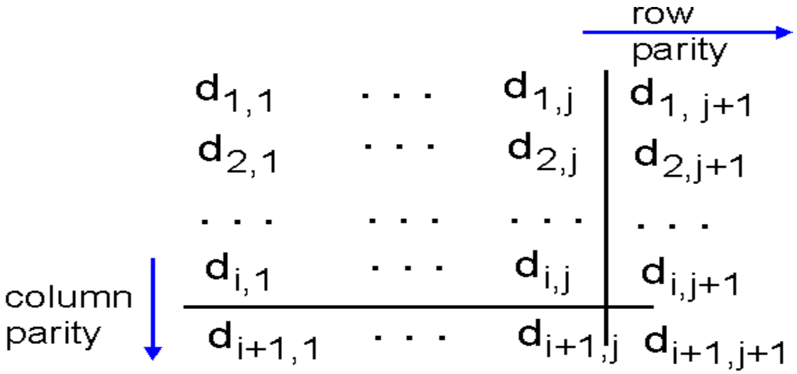
single bit parity:

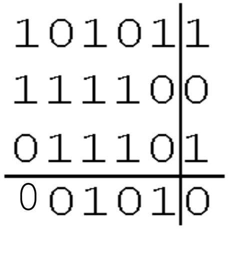
- *d*etect single bit errors



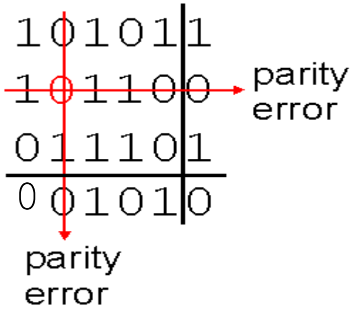
two-dimensional bit parity:

- 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

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

Internet checksum (review)

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

sender:

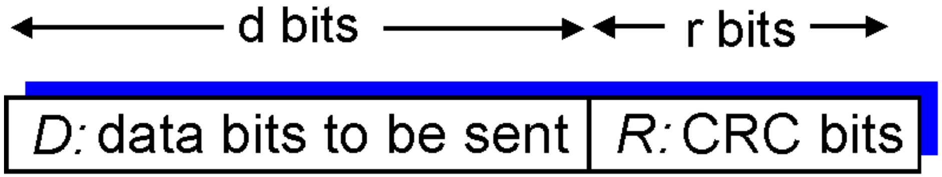
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

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)



bit pattern

$D * 2^r$ XOR R

mathematical formula

CRC example

want:

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

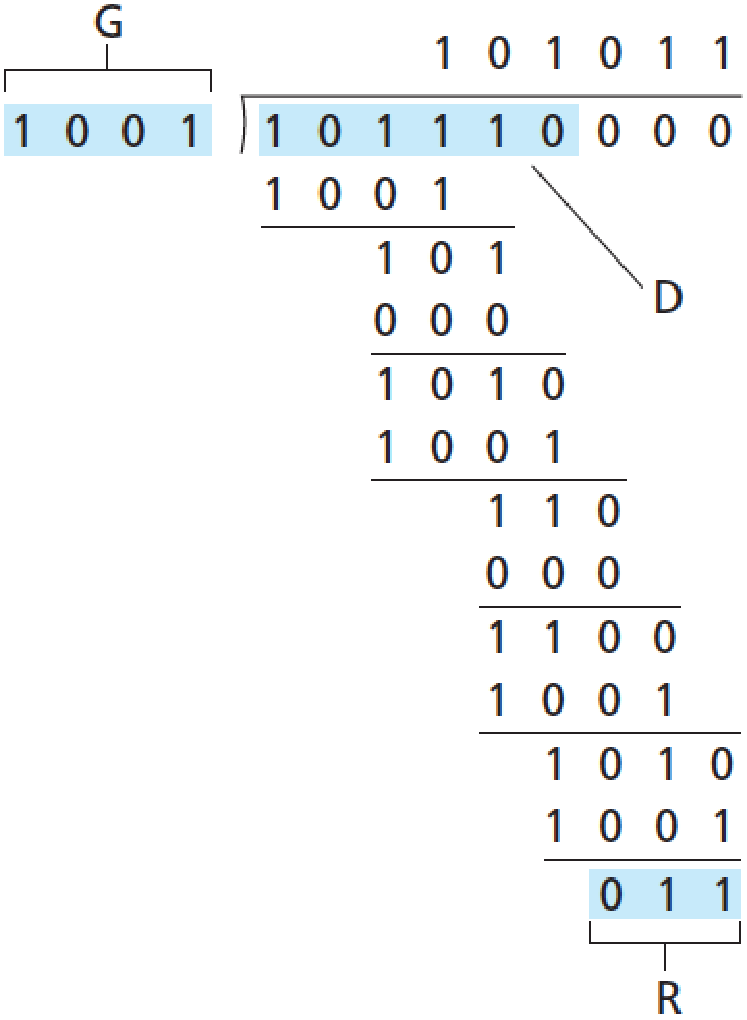
equivalently:

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

equivalently:

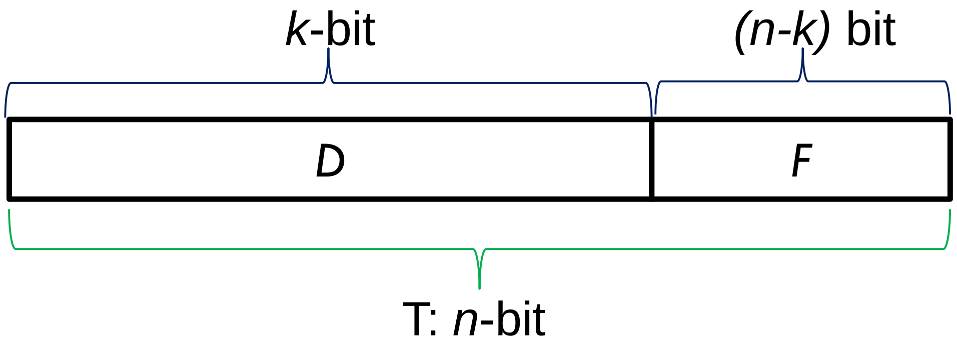
if we divide $D \cdot 2^r$ by G, want remainder R to satisfy:

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



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

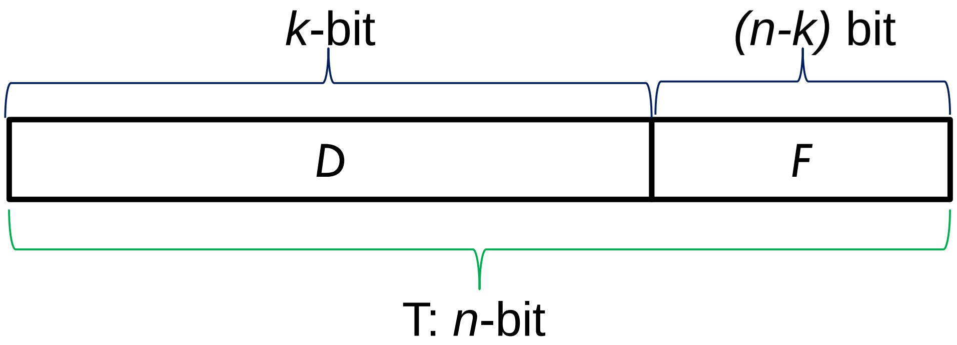
CRC



1. Given
   - Message D = 1010001101 (10 bits)
   - Pattern P = 110101 (6 bits)
   - FCS R = to be calculated (5 bits)
   - Thus, $n=15$, $k=10$, and $(n-k) = 5$
2. The message is multiplied by 2^5, yielding
   - 101000110100000 (c.f. $2^{n-k}D / P = Q + R/P$)
3. This product is divided by P

(see the next page)

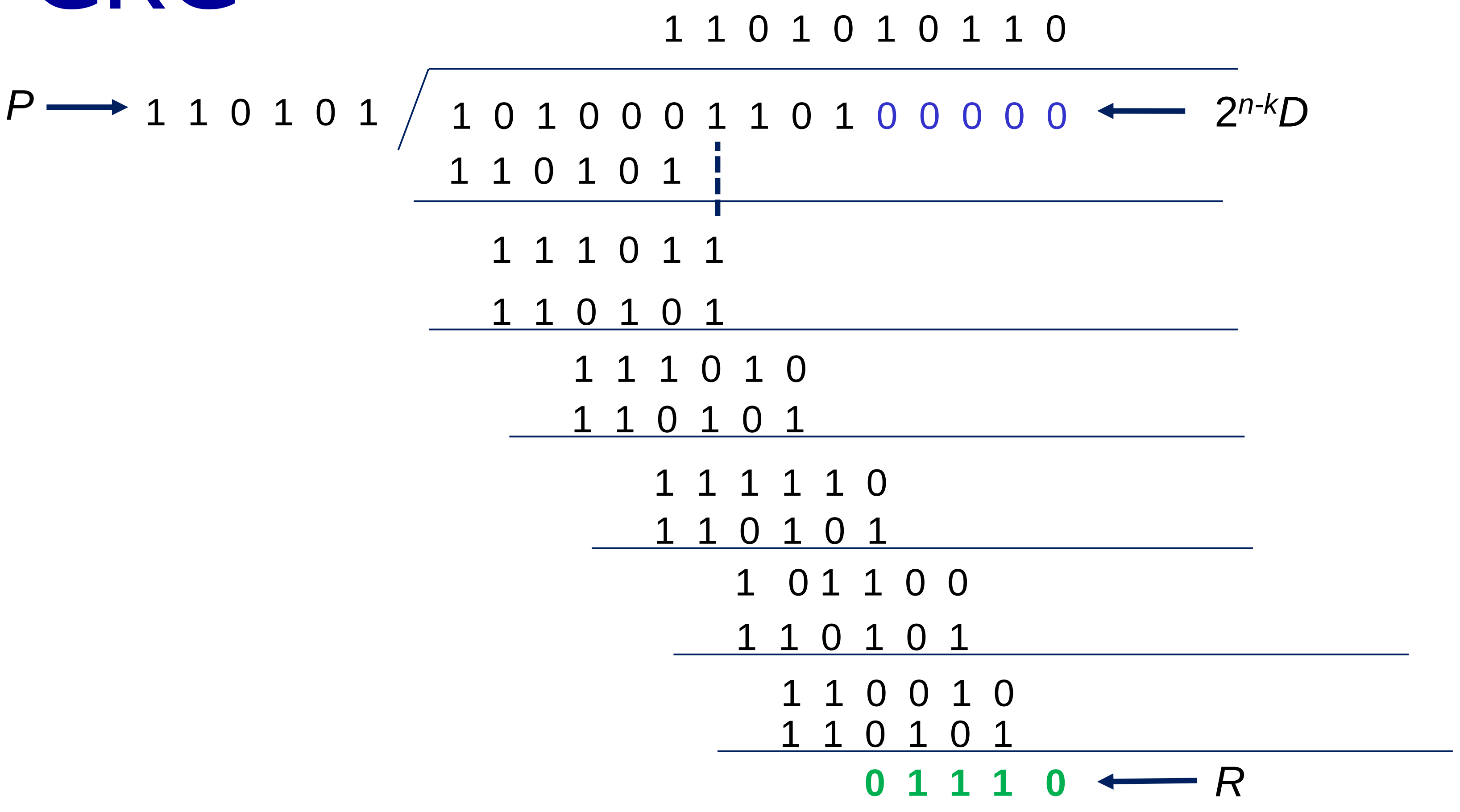
CRC



1. Given
   - Message D = 1010001101 (10 bits)
   - Pattern P = 110101 (6 bits)
   - FCS R = to be calculated (5 bits)
   - Thus, $n=15$, $k=10$, and $(n-k)=5$
2. The message is multiplied by 2^5, yielding
   - 101000110100000 (c.f. $2^{n-k}D / P = Q + R/P$)
3. This product is divided by P

(see the next page)

CRC

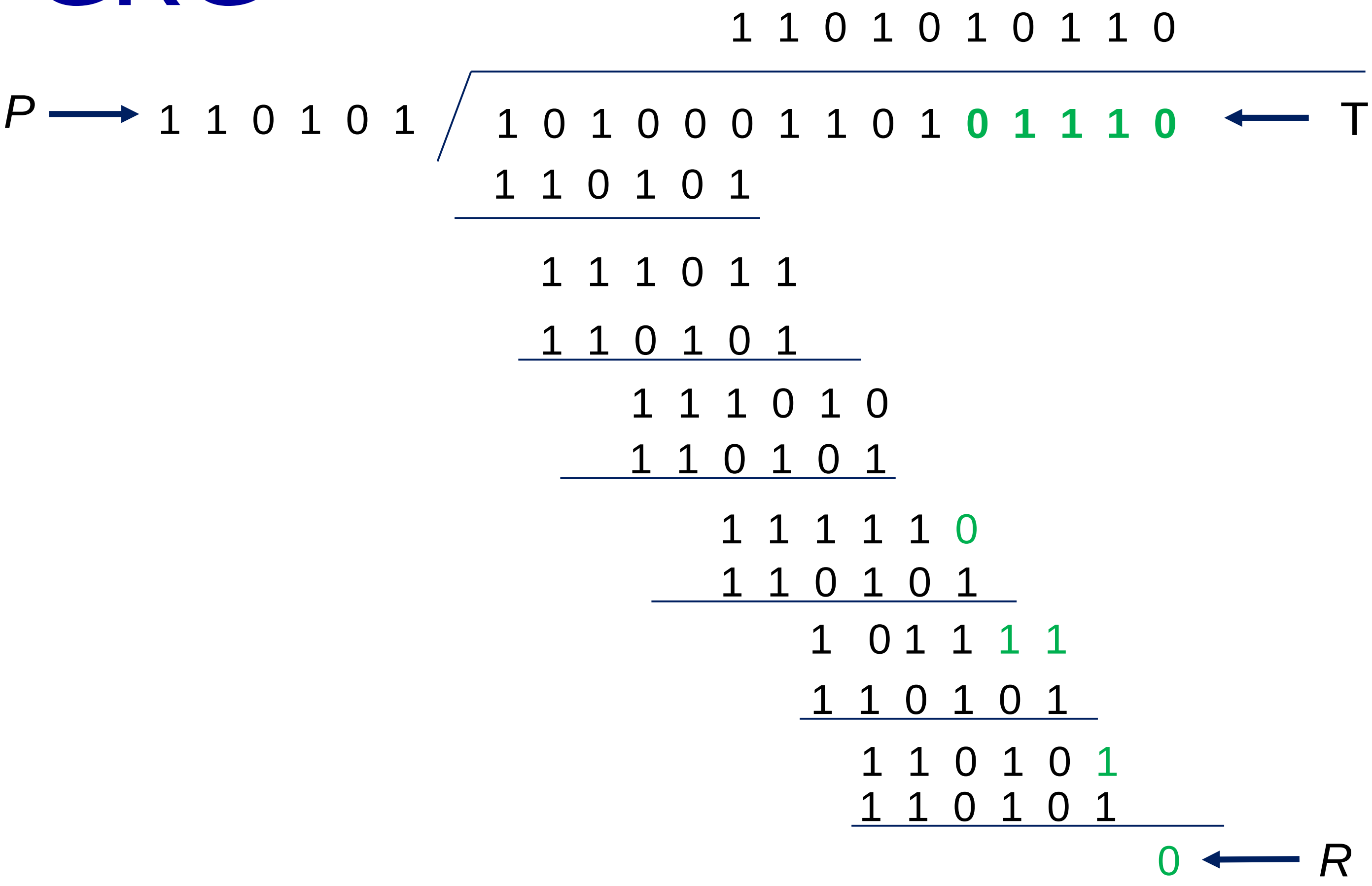


```
                         1 1 0 1 0 1 0 1 1 0
P ⟶ 1 1 0 1 0 1 / 1 0 1 0 0 0 1 1 0 1 0 0 0 0 0  ⟵ 2^{n-k}D
                  1 1 0 1 0 1
                    1 1 1 0 1 1
                    1 1 0 1 0 1
                        1 1 1 0 1 0
                        1 1 0 1 0 1
                            1 1 1 1 1 0
                            1 1 0 1 0 1
                                1  0 1 1 0 0
                                1 1 0 1 0 1
                                  1 1 0 0 1 0
                                  1 1 0 1 0 1
                                      0 1 1 1  0  ⟵ R
```

4. The remainder is added to 2^5D to give T= 1010001101**01110**
5. If there are no errors, the receiver receives *T* intact (i.e., no damage). The received frame is divided by *P*:

   (see the next page)

CRC

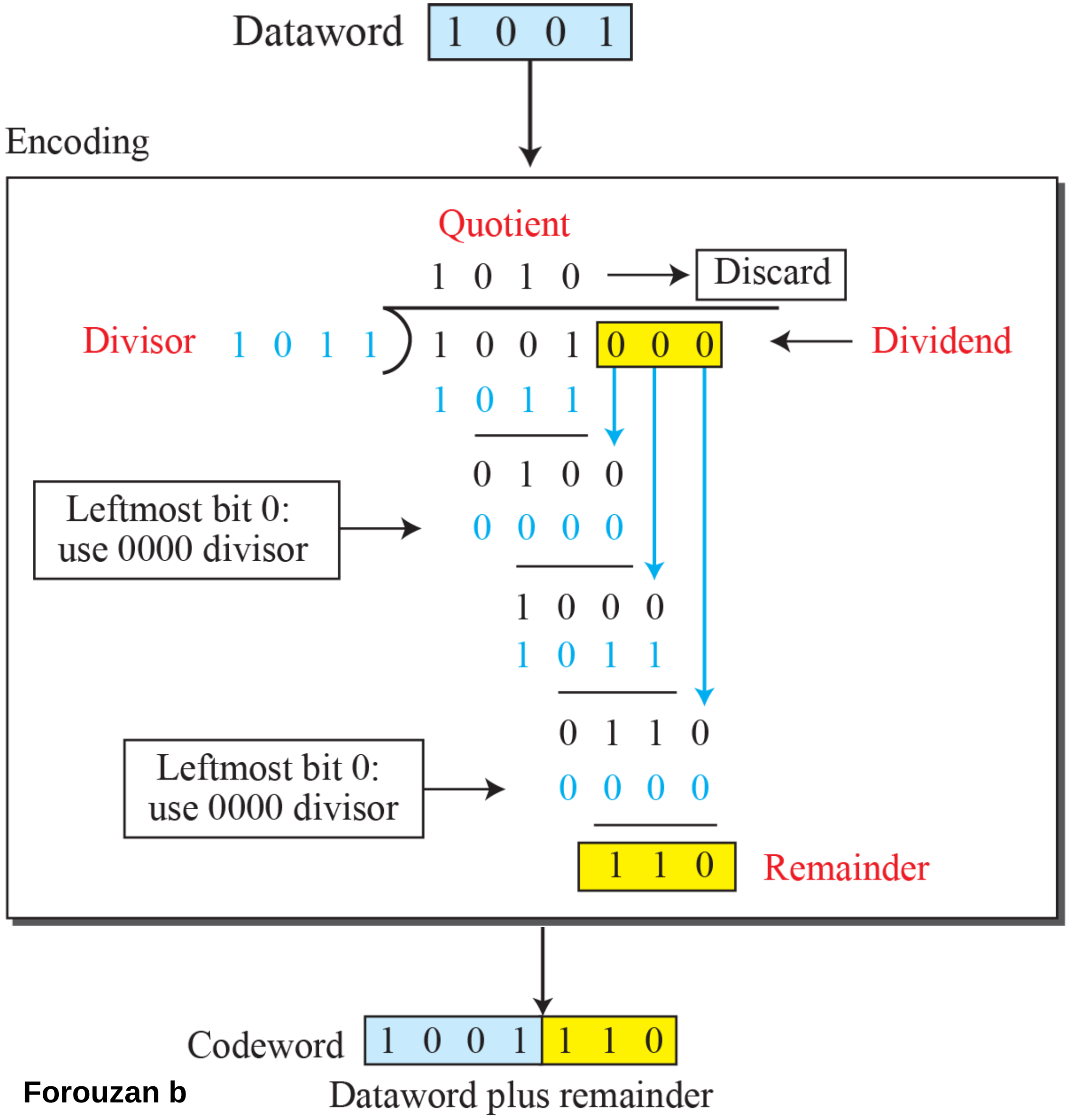


```
                               1 1 0 1 0 1 0 1 1 0
                    ________________________________________
P ──→ 1 1 0 1 0 1 / 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0   ←── T
                    1 1 0 1 0 1
                    ___________
                      1 1 1 0 1 1
                      1 1 0 1 0 1
                      ___________
                          1 1 1 0 1 0
                          1 1 0 1 0 1
                          ___________
                              1 1 1 1 1 0
                              1 1 0 1 0 1
                              ___________
                                  1  0 1 1 1 1
                                  1 1 0 1 0 1
                                  ___________
                                    1 1 0 1 0 1
                                    1 1 0 1 0 1
                                    ___________
                                              0  ←── R
```

- Because there is no remainder, it is assumed that there have been no errors

Figure 10.6: Division in CRC encoder

dividend ÷ divisor = *quotient*



Dataword 1 0 0 1

Encoding

Quotient

1 0 1 0 → Discard

Divisor 1 0 1 1) 1 0 0 1 0 0 0 ← Dividend

1 0 1 1

0 1 0 0

Leftmost bit 0: use 0000 divisor → 0 0 0 0

1 0 0 0

1 0 1 1

0 1 1 0

Leftmost bit 0: use 0000 divisor → 0 0 0 0

1 1 0 Remainder

Codeword 1 0 0 1 1 1 0

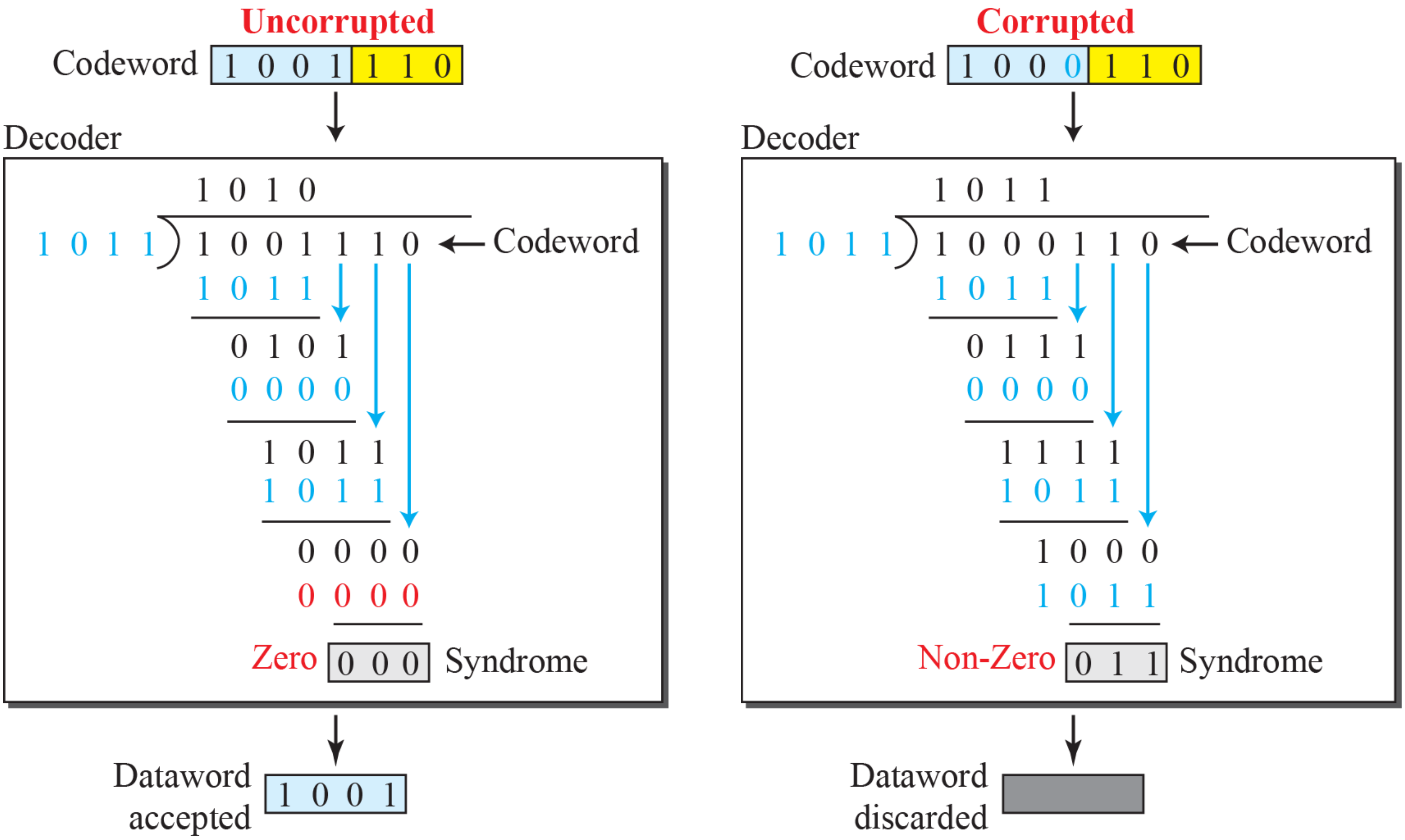
Dataword plus remainder

Forouzan b

Note:

Multiply: AND
Subtract: XOR

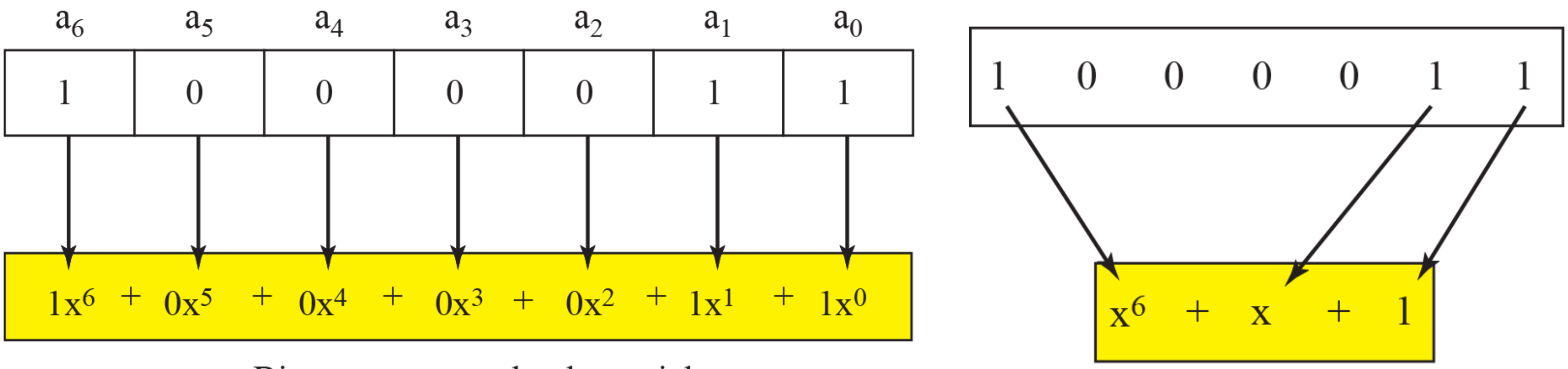
Figure 10.7: Division in the CRC decoder for two cases



Forouzan book: 10.21

Binary vs. polynomials

- A better way to understand ***cyclic codes*** and how they can be analyzed is to represent them as ***polynomials***.
- In practice, all commonly used CRCs employ the Galois field of two elements, GF(2).
  - The two elements are usually called 0 and 1, comfortably matching computer architecture; A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1.
- The power of each term shows the position of the bit;
  - the coefficient shows the value of the bit.

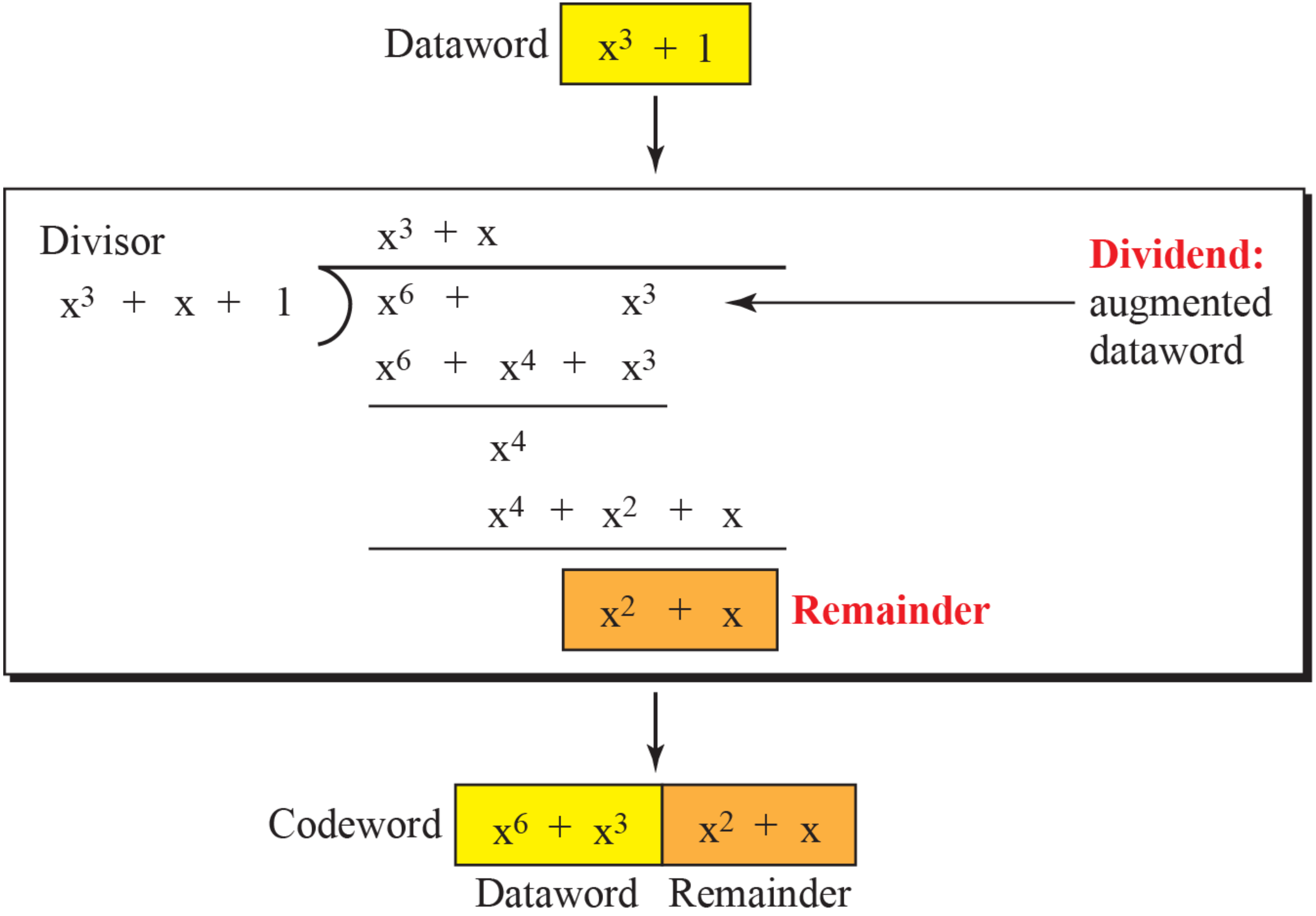


a_6 a_5 a_4 a_3 a_2 a_1 a_0
1 0 0 0 0 1 1
$1x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0$
1 0 0 0 0 1 1
$x^6 + x + 1$

a. Binary pattern and polynomial

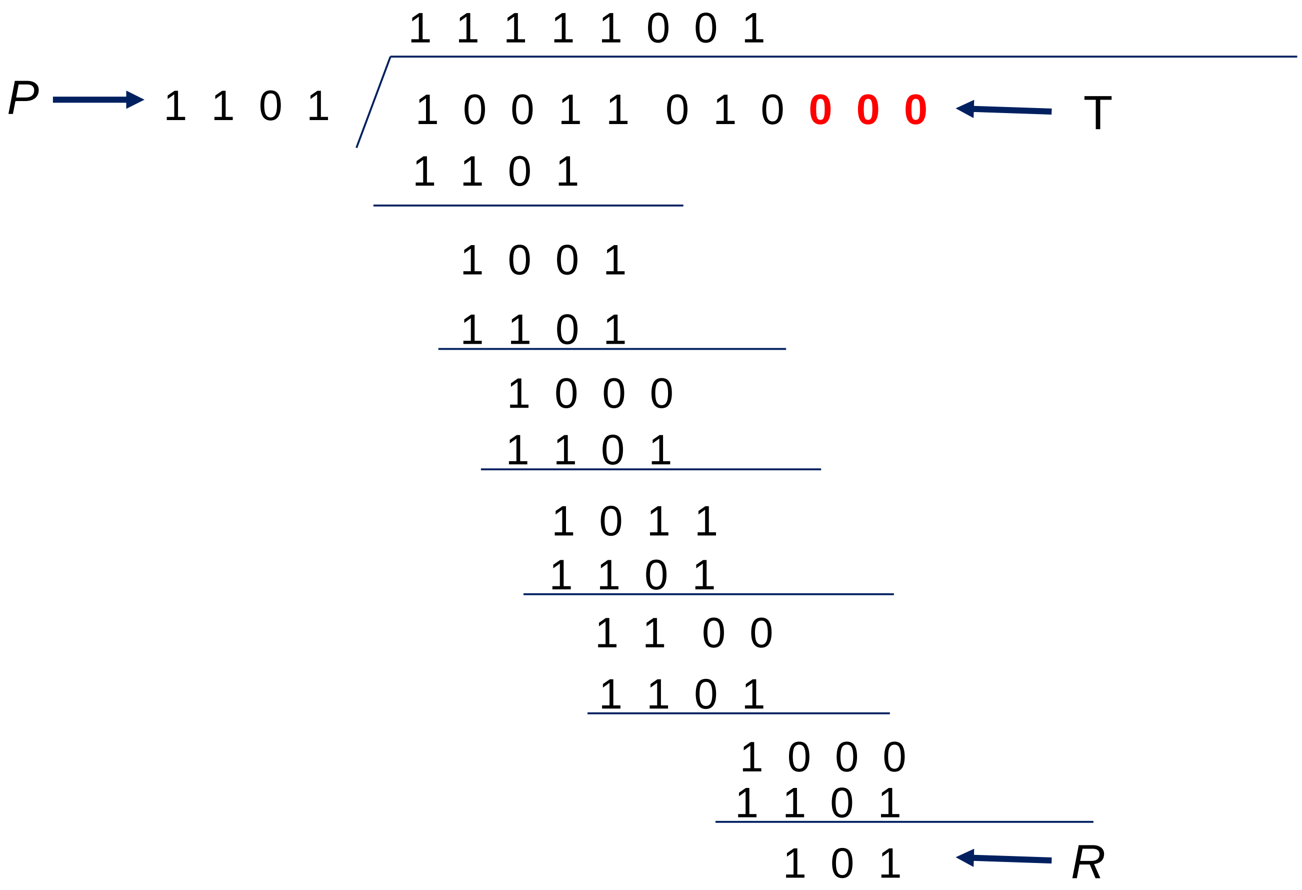
b. Short form

CRC division using polynomials



Example using a polynomials

- Message D= $X^7 + X^4 + X^3 + X^1$, 10011010
- $2^{n\text{-}k}D$ = *10001101000*
- *P* = 1101



1 1 1 1 1 0 0 1

P 1 1 0 1

1 0 0 1 1 0 1 0 0 0 0

T

1 1 0 1

1 0 0 1

1 1 0 1

1 0 0 0

1 1 0 1

1 0 1 1

1 1 0 1

1 1 0 0

1 1 0 1

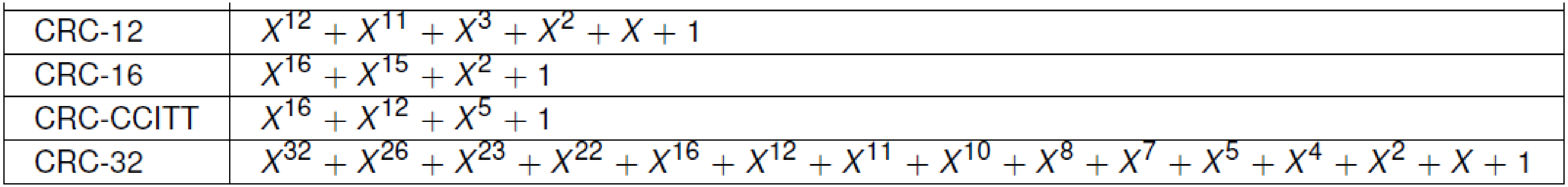
1 0 0 0

1 1 0 1

1 0 1

R

CRC – Some Standard Polynomials



CRC-12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$
CRC-16	$X^{16} + X^{15} + X^2 + 1$
CRC-CCITT	$X^{16} + X^{12} + X^5 + 1$
CRC-32	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

- CRC-12: for transmission of streams of 6-bit characters and generates a 12-bit frame check sequence (FCS)
- CRC-16 and CRC-CCITT: are popular for 8-bit characters and result in a 16-bit FCS; High-Level Data Link Control (**HDLC**)
- CRC-32: is specified as an option in some point-to-point synchronous transmission standards and is used in **IEEE 802 LAN** standards
- An example: http://srecord.sourceforge.net/crc16-ccitt.html#overview

Detection vs. Correction

- In error detection, we are only looking to see if any error has occurred.
  - The answer is a simple yes or no.
  - We are not even interested in the number of corrupted bits.
  - A single-bit error is the same for us as a burst error.
- The **correction** of errors is more difficult than the detection.
- In error correction, we need to know the exact number of bits that are corrupted and, more importantly, their location in the message.

Forward Error Correction (FEC)

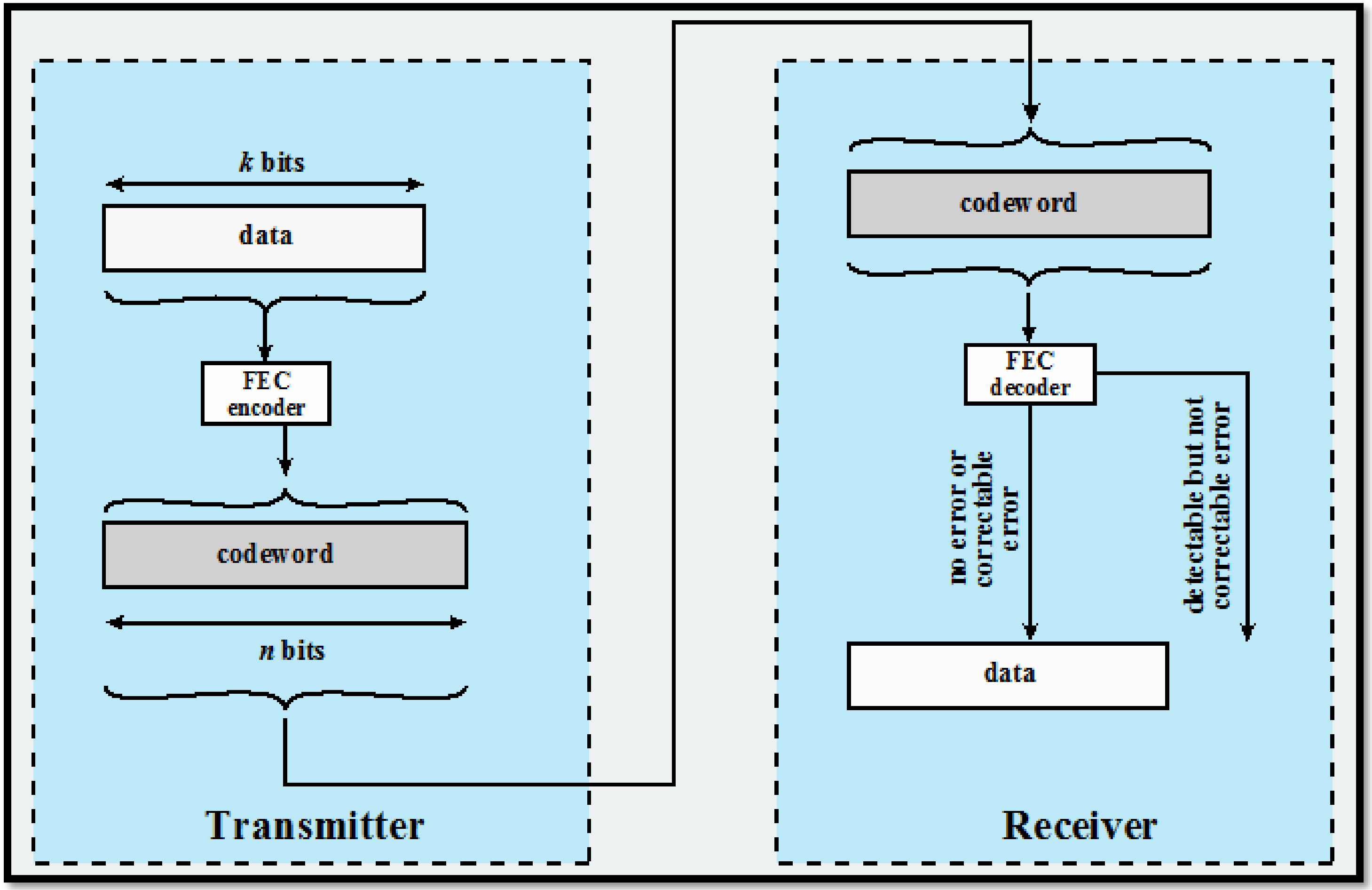
- Correction of detected errors usually requires data blocks to be ***retransmitted***
- Not appropriate for wireless applications:
  - The bit error rate (BER) on a wireless link can be quite high, which would result in a large number of retransmissions
  - Propagation delay is very long compared to the transmission time of a single frame
- **Need to correct errors on basis of bits received**

Codeword

- On the transmission end each *k*-bit block of data is mapped into an *n*-bit block ($n > k$) using a forward error correction (FEC) encoder

FEC Process

- In block coding, we divide our message into blocks, each of k bits, called *datawords*. We add *r* redundant bits to each block to make the length *n* = *k* + *r*. The resulting *n*-bit blocks are called *codewords*.



FEC Process (2)

- During transmission, the signal is subject to impairments, which may produce bit errors in the signal.
- At the receiver, the incoming signal is demodulated to produce a bit string that is similar to the original codeword but may contain errors.

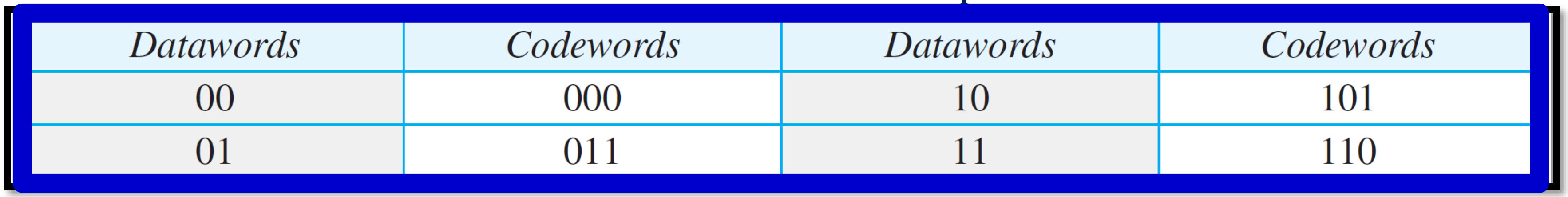
One of four possible outcomes:

1. **No errors:**
   - If there are no bit errors, the input to the FEC decoder is identical to the original codeword, and the decoder produces the original data block as output.
2. **Detectable, correctable errors:**
   - For certain error patterns, it is possible for the decoder to detect and correct those errors.
   - Thus, even though the incoming data block differs from the transmitted codeword, the FEC decoder is able to map this block into the original data block.
3. **Detectable, not correctable errors:**
   - For certain error patterns, the decoder can detect but not correct the errors.
   - In this case, the decoder simply reports an uncorrectable error.
4. **Undetectable errors:**
   - For certain, typically rare, error patterns, the decoder does not detect the error and maps the incoming n-bit data block into a *k*-bit block that differs from the original k-bit block.

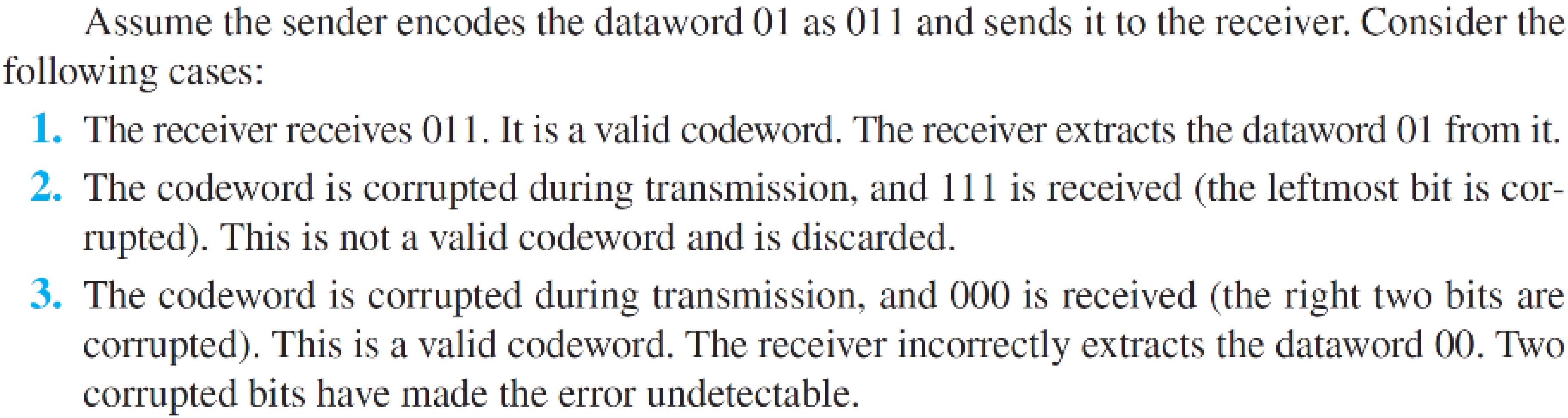
An example

- Let us assume that *k* = 2 and *n* = 3. Table 10.1 shows the list of datawords and codewords.

Table 10.1: A code for error detection in Example 10.1



Datawords	*Codewords*	*Datawords*	*Codewords*
00	000	10	101
01	011	11	110



Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

1. The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.
2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.
3. The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

Block Code Principles

- **Hamming distance**
  - $d(v_1, v_2)$ between two n –bit binary sequences v_1 and v_2 is the number of bits in which v_1 and v_2 disagree
- Redundancy of the code
  - The ratio of redundant bits to data bits $(n-k)/k$
- Code rate
  - The ratio of data bits to total bits k/n
  - Is a measure of how much additional bandwidth is required to carry data at the same data rate as without the code

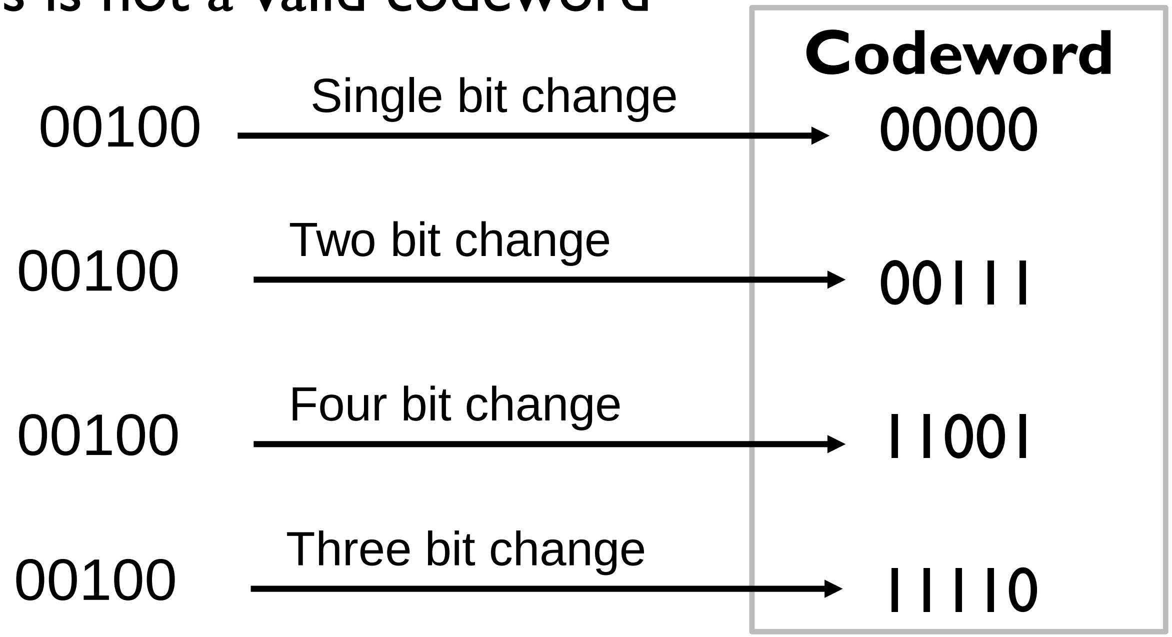
Block Code Principles (2)

- Hamming distance
  - E.g. v_1=011011, v_2=110001, $d(v_1,v_2)$ = 3
- Suppose we wish to transmit blocks of data of length *k* bits.
  - Instead of transmitting each block as *k* bits, we map each *k*-bit sequence into a unique *n*-bit codeword.
  - E.g. For *k* = 2 and *n*=5, we can make the following assignment:

Data Block (dataword)	**Codeword**
00	00000
01	00111
10	11001
11	11110

Block Code Principles (3)

- A codeword block is received with the bit pattern 00100
  - This is not a valid codeword



Block Code Principles (3)

- If an invalid codeword is received, then valid codeword is closest to it (*minimum* distance) is selected
  - This will only work if there is a unique valid codeward at a minimum distance from each invalid codeword
  - e.g., it is not true that for every invalid codeword there is one and only one valid codeword at a minimum distance
  - There are $2^5 = 32$ possible codwords of which 4 are valid, leaving 28 invalid codewords.

Block Code Principles (4)

Invalid codeword	Minimum Distance	Valid Codeword	Invalid Codeword	Minimum Distance	Valid Codeword
00001	1	00000	10000	1	00000
00010	1	00000	10001	1	11001
00011	1	00111	10010	2	**00000 or 11110**
00100	1	00000	10011	2	**00111 or 11001**
00101	1	00111	10100	2	**00111 or 11001**
00110	1	00111	10101	2	**00111 or 11001**
01000	1	00000	10110	1	11110
01001	1	11001	10111	1	00111
01010	2	**00000 or 11110**	11000	1	11001
01011	2	**00111 or 11001**	11010	1	11110
01100	2	**00000 or 11110**	11011	1	11001
01101	2	**00111 or 11001**	11100	1	11110
01110	1	11110	11101	1	11001
01111	1	00111	11111	1	11110

Block Code Principles (5)

- There are eight cases in which an invalid codeword is at a distance 2 from two different valid codewords.
  - Thus, if one such invalid codeword is received, and error in 2 bits could have caused it and the receiver has no way to choose between the two alternatives.
  - An error is detected but cannot be corrected
- This code is capable of correcting all single-bit errors but cannot correct double-bit errors

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

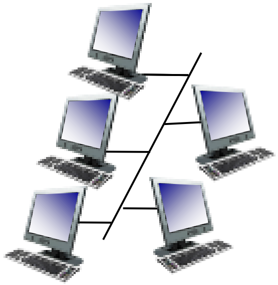
6.6 data center networking

6.7 a day in the life of a web request

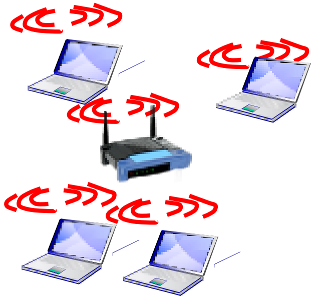
Multiple access links, protocols

two types of "links":

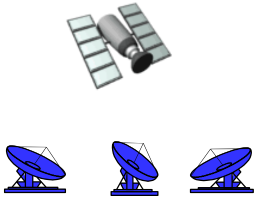
- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - upstream Hybrid fiber-coaxial (HFC)
  - 802.11 wireless LAN



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

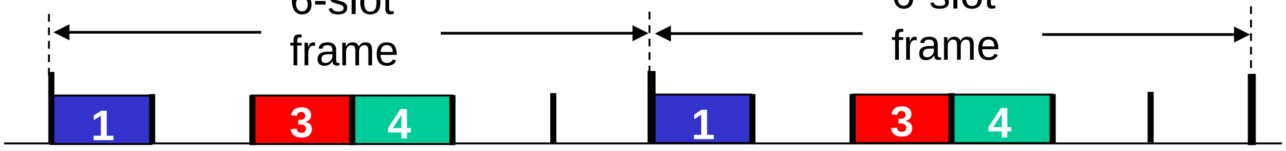
three broad classes:

- *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

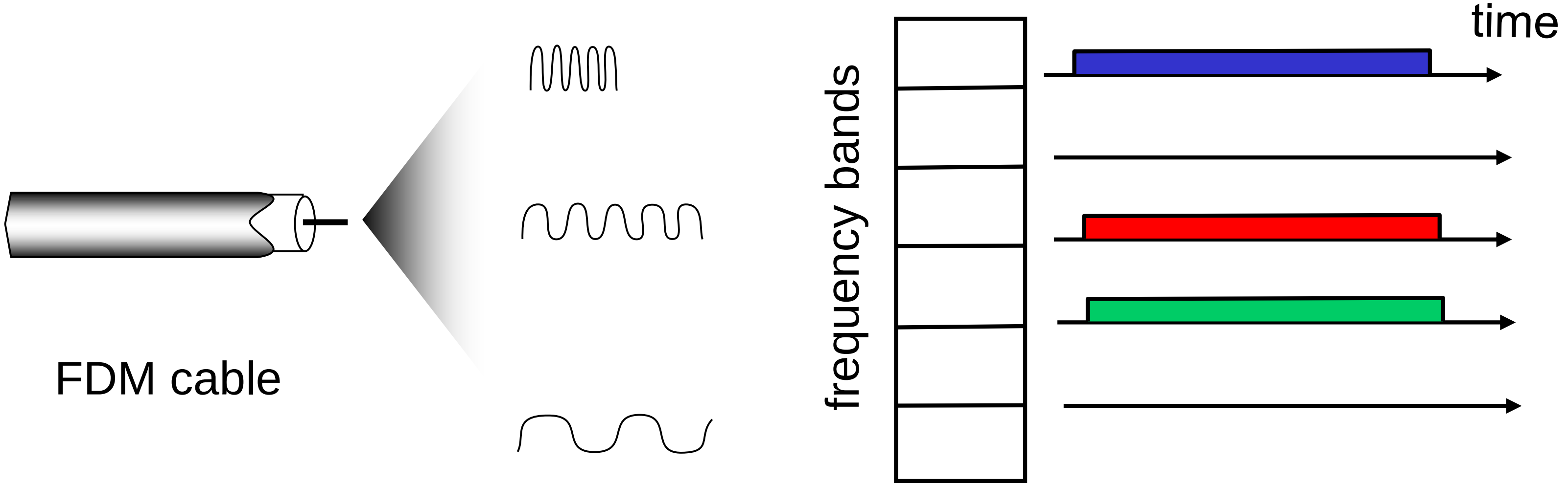
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



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 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

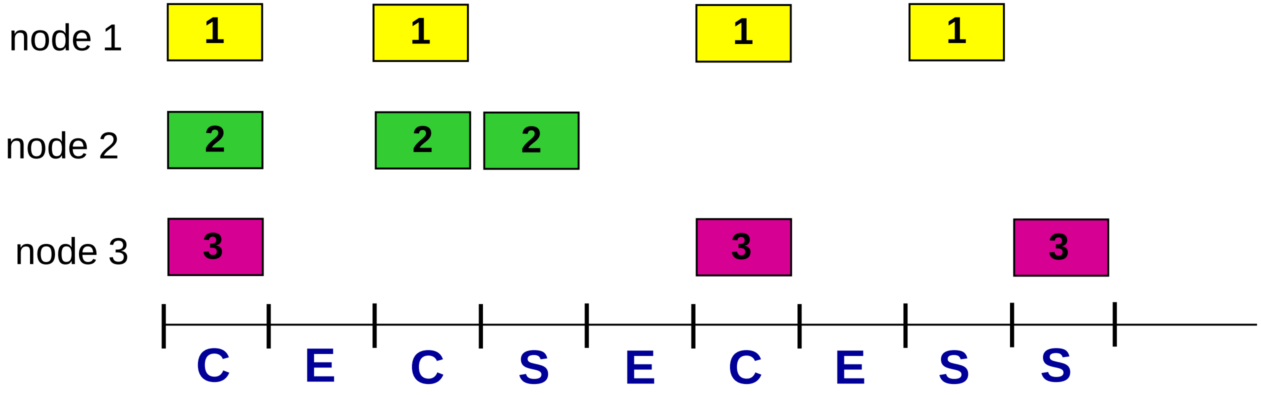
assumptions:

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

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

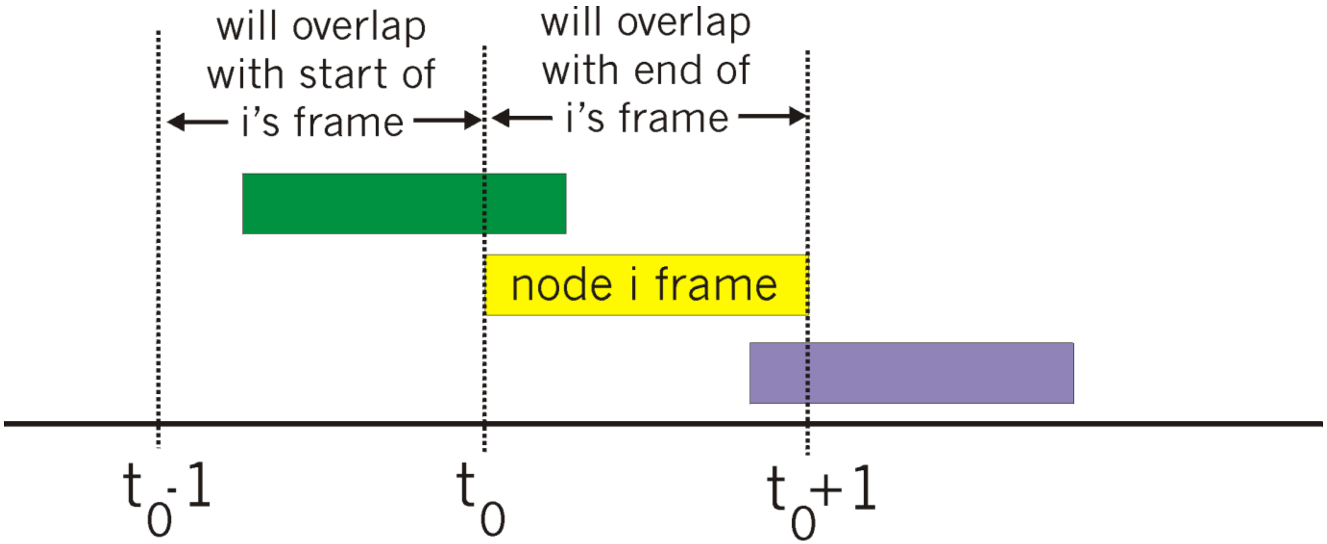
max efficiency = 1/e = .37

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

!

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- 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 $[t_0\text{-}1, t_0]$ ·
P(no other node transmits in $[t_0\text{-}1, t_0]$

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

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

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

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

even *worse* than slotted Aloha!

CSMA (carrier sense multiple access)

CSMA: listen before transmit:

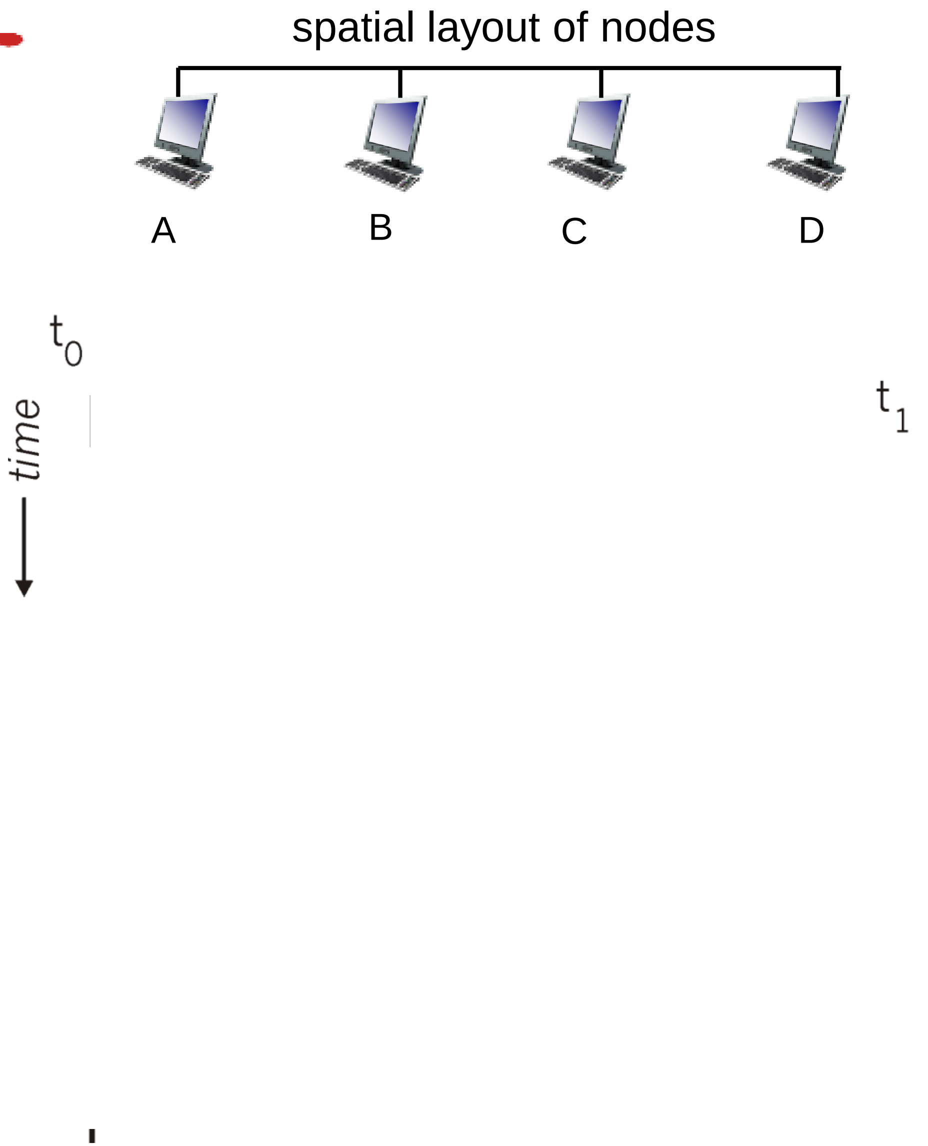
if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

- human analogy: don't interrupt others!

CSMA collisions

- collisions *can* still occur: propagation delay means two nodes may not hear each other' s transmission
- collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability
- Nodes do not perform collision detection;
  - both B and D continue to transmit their frames in their entirety even though a collision has occurred.



CSMA/CD (collision detection)

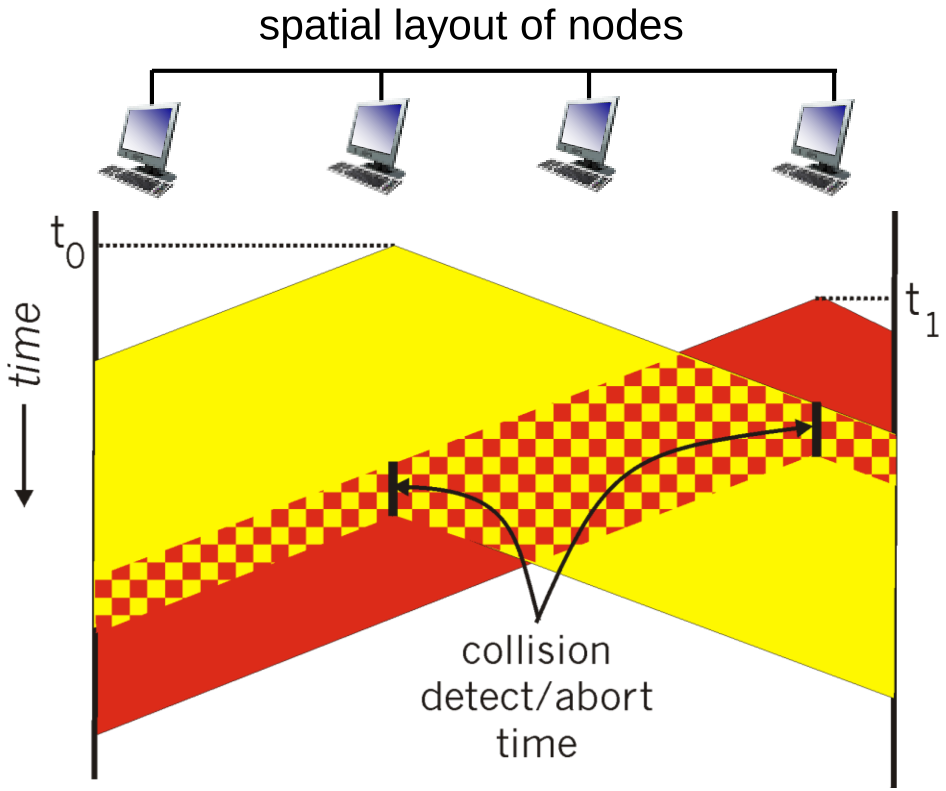
CSMA/CD: carrier sensing, deferral as in CSMA

- 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 $\{0,1,2, \ldots, 2^m-1\}$. NIC waits K·512 bit times, returns to Step 2
   - longer backoff interval with more collisions

Ethernet CSMA/CD algorithm

- Suppose that a node attempts to transmit a frame for the first time and while transmitting it detects a collision.
- The node then chooses K =0 with probability .5 or choose K = 1 with probability 0.5.
  - If the node chooses K = 0, then it immediately begins sensing the channel.
  - If the node chooses K=1, it waits 512 bit times (e.g., 5.12 microseconds for a 100 Mbps Ethernet) before beginning the sense-and-transmit-when-idle cycle.
- after *m*th collision, NIC chooses *K* at random from *{0,1,2, …, 2*^{m}*-1}*. NIC waits K·512 bit times
  - After a second collision, K is chosen with equal probability from {0, 1, 2,3}.
  - After three collisions, K is chose with equal probability from {0,1,2,3,4,5,6,7}
  - After three collisions, K is chose with equal probability from {0,1,2,3,4,5,6,7}
  - After 10 or more collisions, K is chosen with equal probability from {0,1,2,…,1023}.
- Thus, the size of the sets from which K is chosen grows exponentially with the number of collisions; for this reason this algorithm is referred to as binary exponential backoff.

CSMA/CD efficiency

- t_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame (approx. 1.2 msecs for a 10 Mbps Ethernet)

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

- efficiency goes to 1
  - as t_{prop} goes to 0 (if the propagation delay is zero, colliding node abort immediately without wasting channel)
  - as t_{trans} goes to infinity (when a frame grabs the channel, it will hold on to the channel for a long time; thus, the channel will be doing productive work most of the time).
- 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

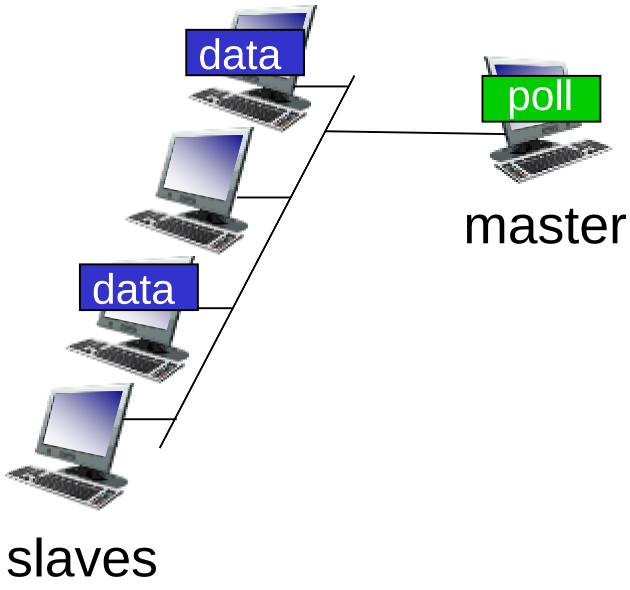
"taking turns" protocols

look for best of both worlds!

"Taking turns" MAC protocols

polling:

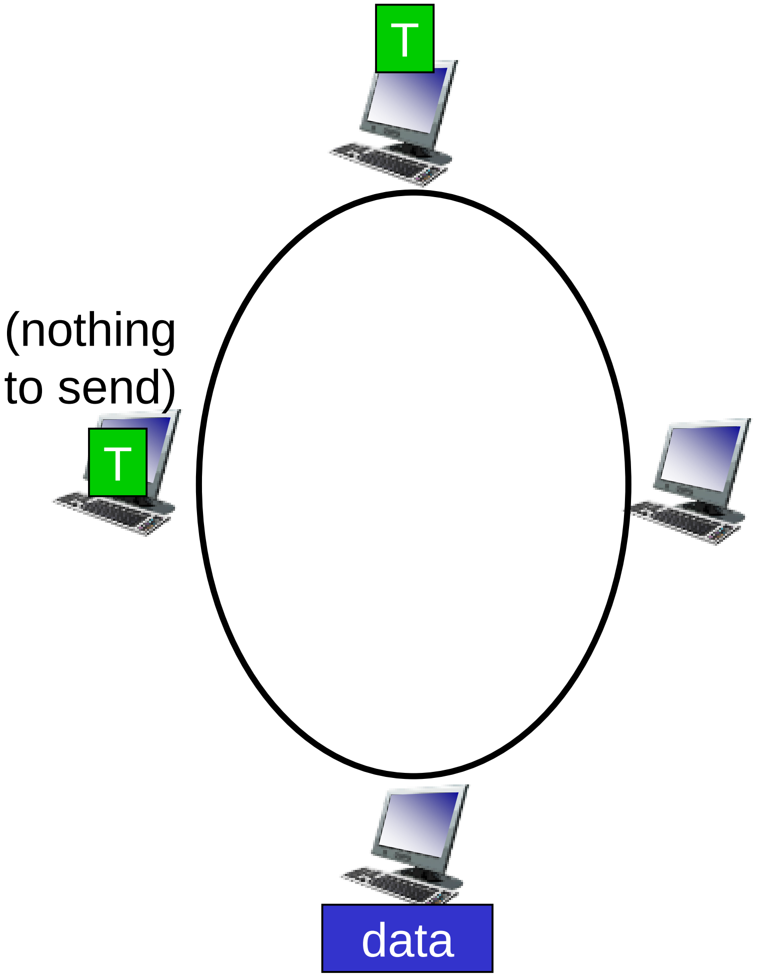
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"Taking turns" MAC protocols

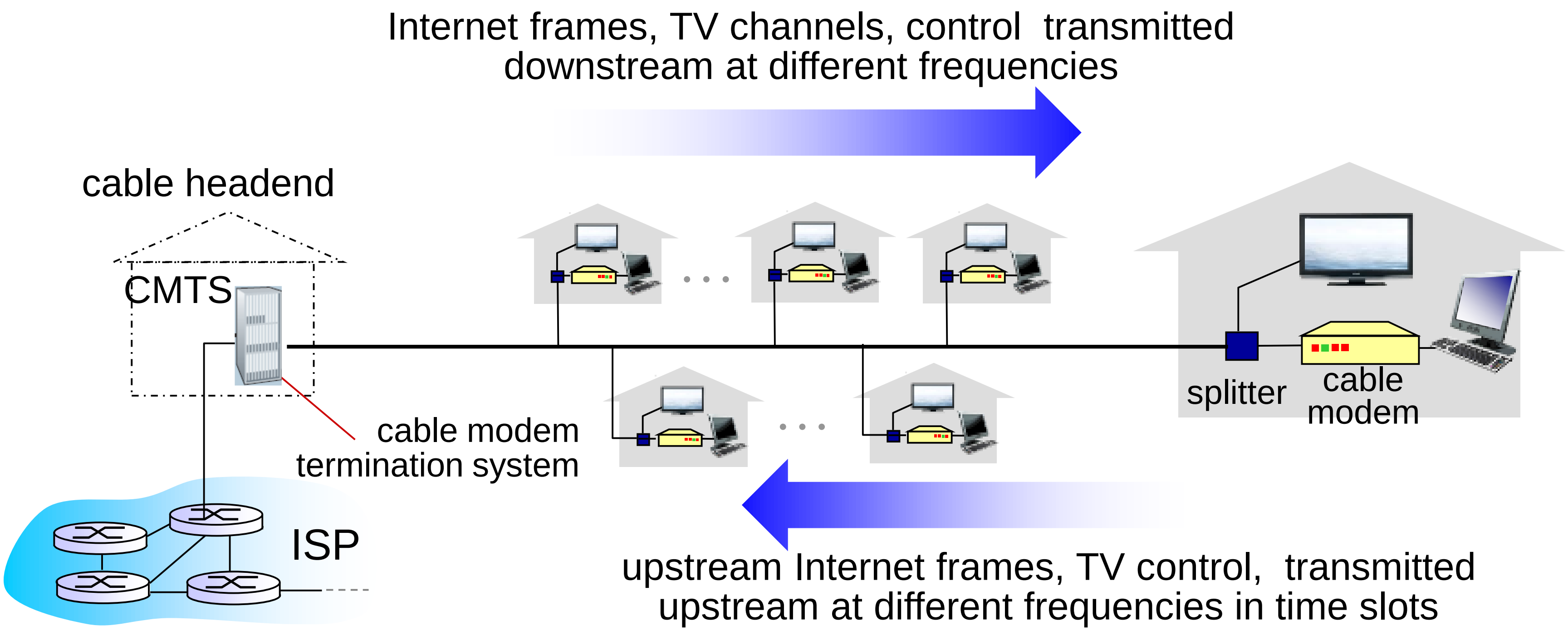
token passing:

- No master node.
- control *token* passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)
- Examples
  - Fiber distributed data interface (FDDI) proto.
  - IEEE 802.5 token ring protocol.



Cable modem termination system (CMTS)

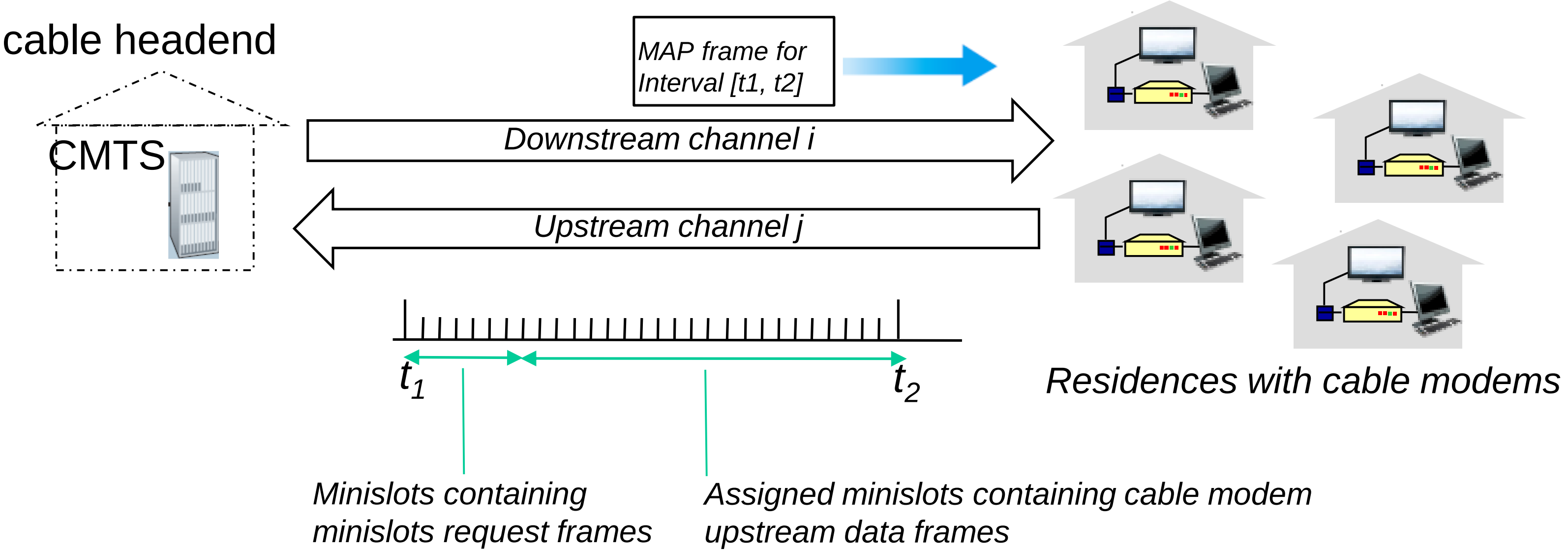
Cable access network


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

- multiple 40Mbps downstream (broadcast) channels
  - Data-Over-Cable Service Interface Specifications (DOCSIS) specifies protocols.
- multiple 30 Mbps upstream channels
  - multiple access: all users contend for certain upstream channel time slots (others assigned)

bandwidth allocation map (MAP)

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
  - single CMTS transmits into channels (each channel 6 MHz wide, with a maxium throughput of approx. 40 Mbps per channel).
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - Each 6.4MHz wide with max. throughput of appx. 30 Mbps)
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

Summary of MAC protocols

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

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

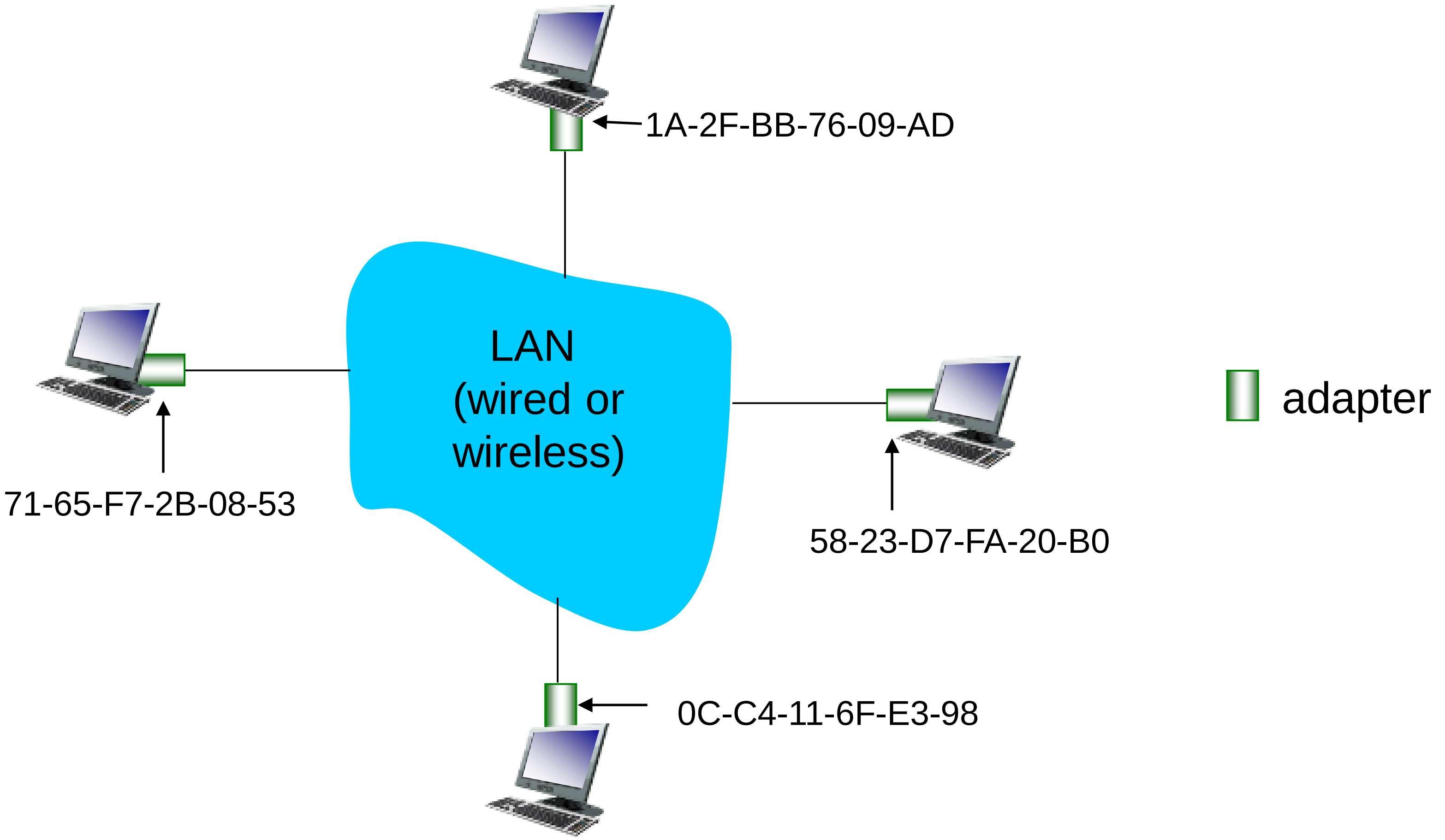
MAC addresses and ARP

- 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

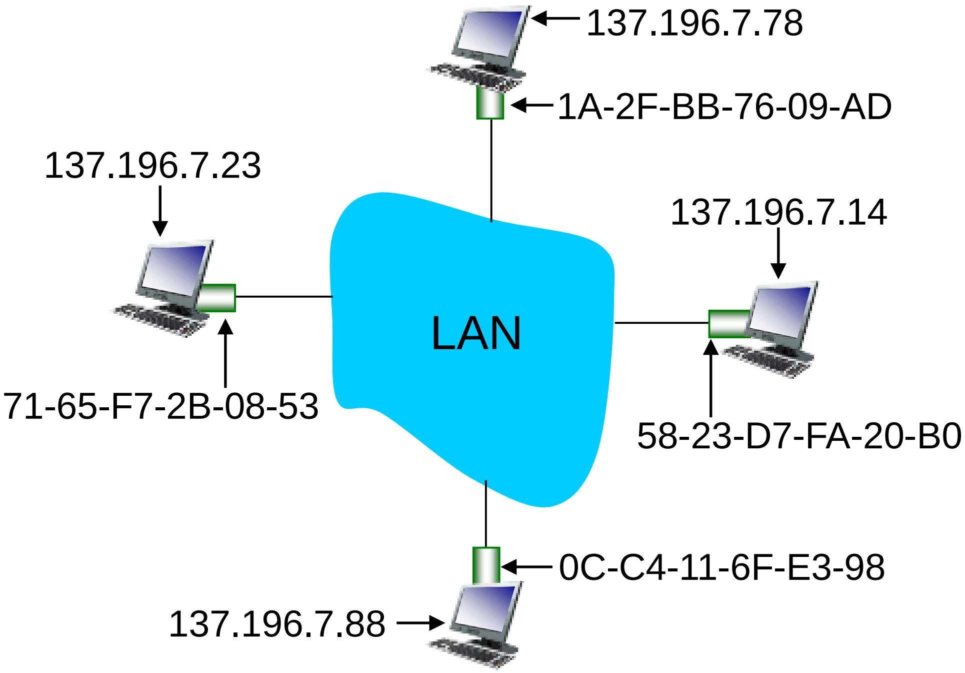


LAN addresses (more)

- 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 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)

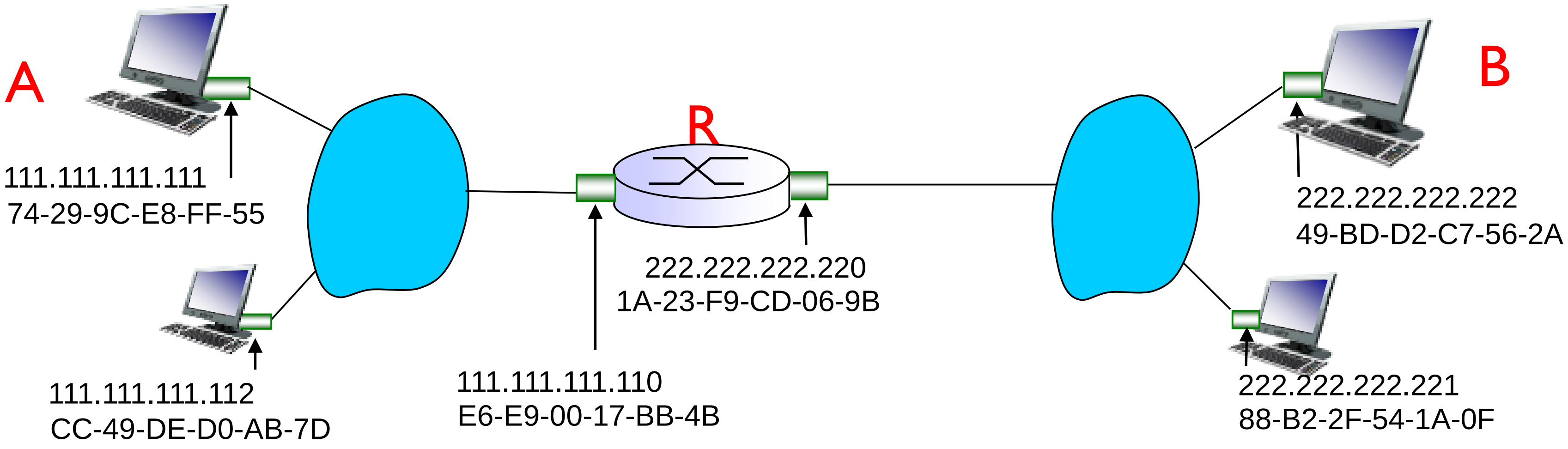
ARP protocol: same LAN

- 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
  - 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

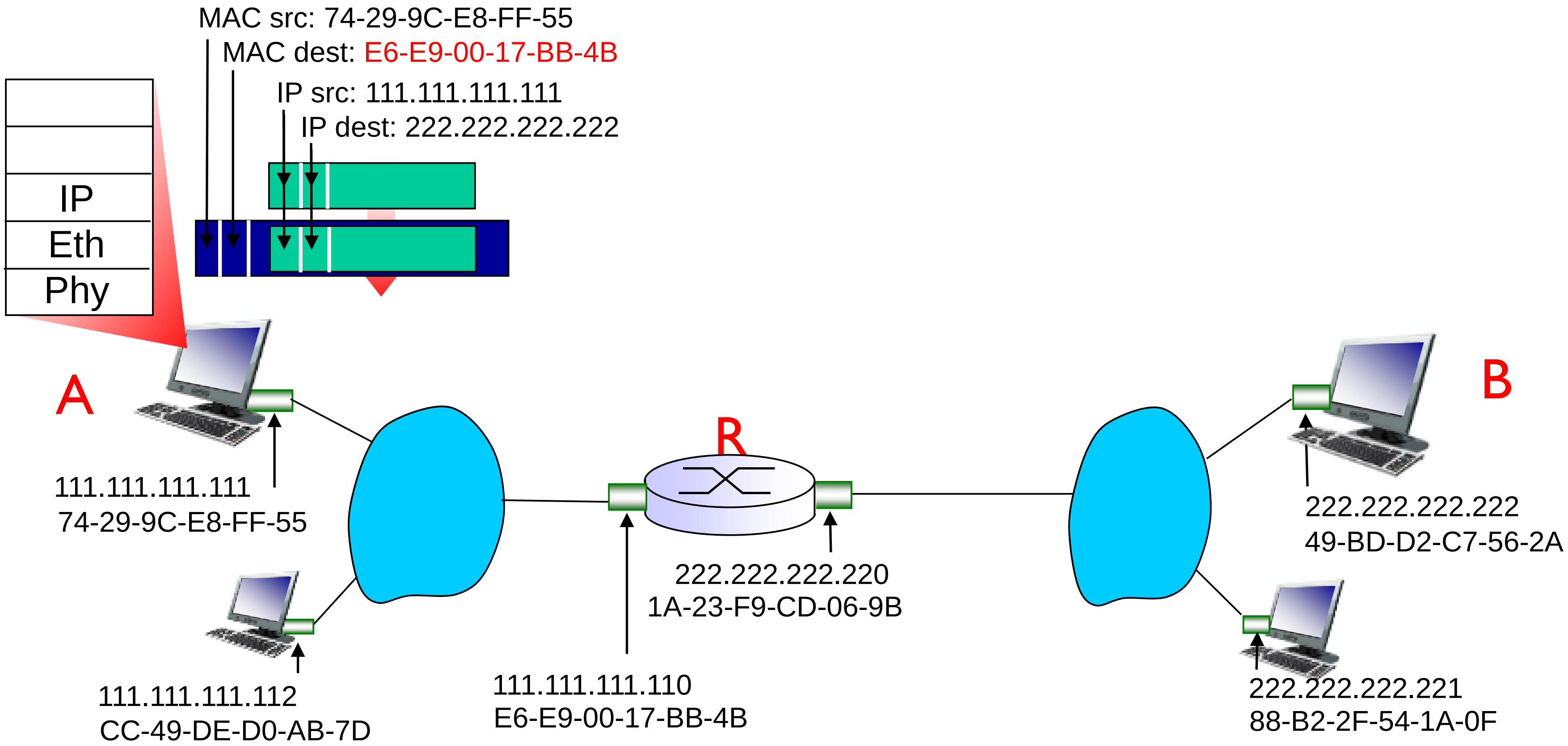
walkthrough: send datagram from A to B via R

- 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?)



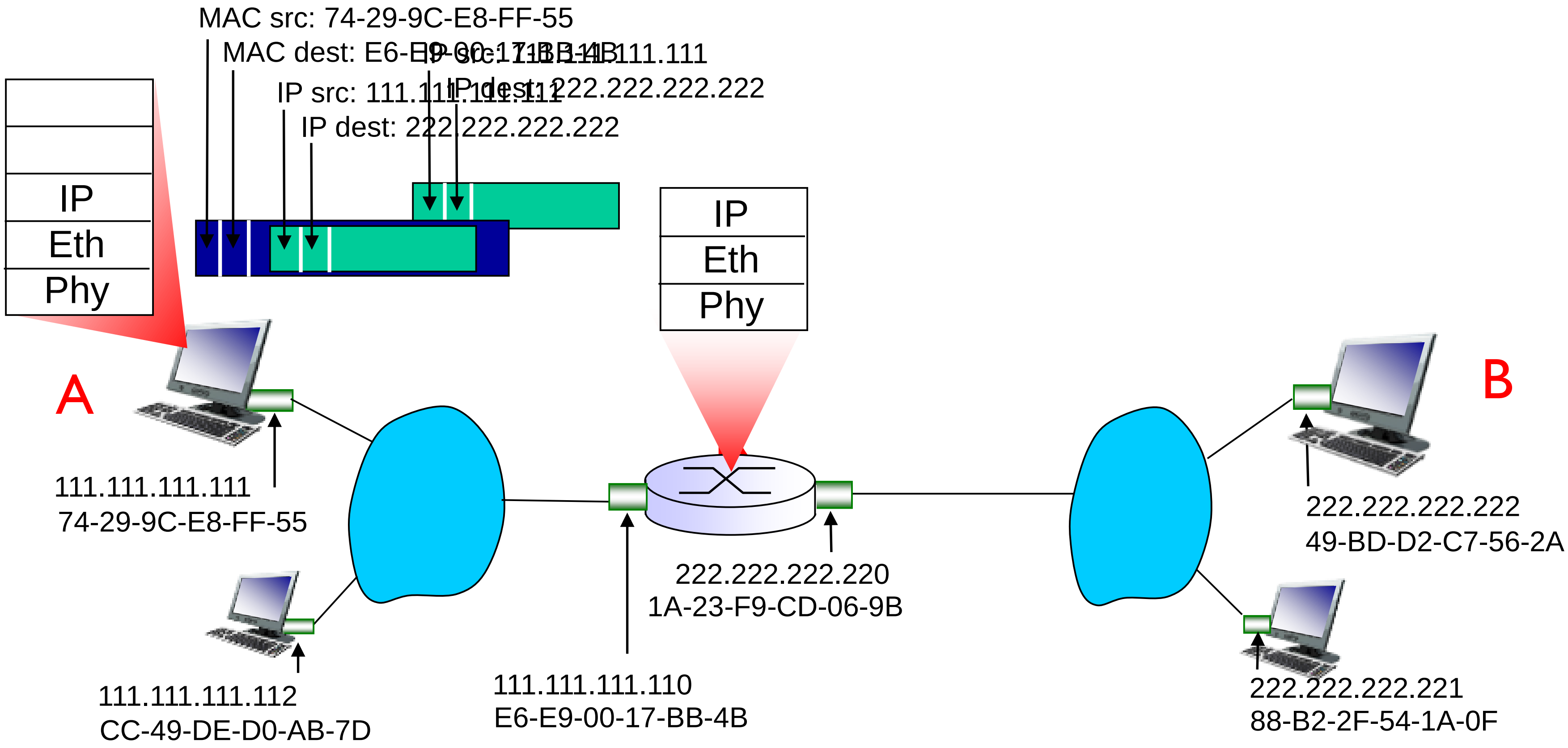
Addressing: routing to another LAN

- 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



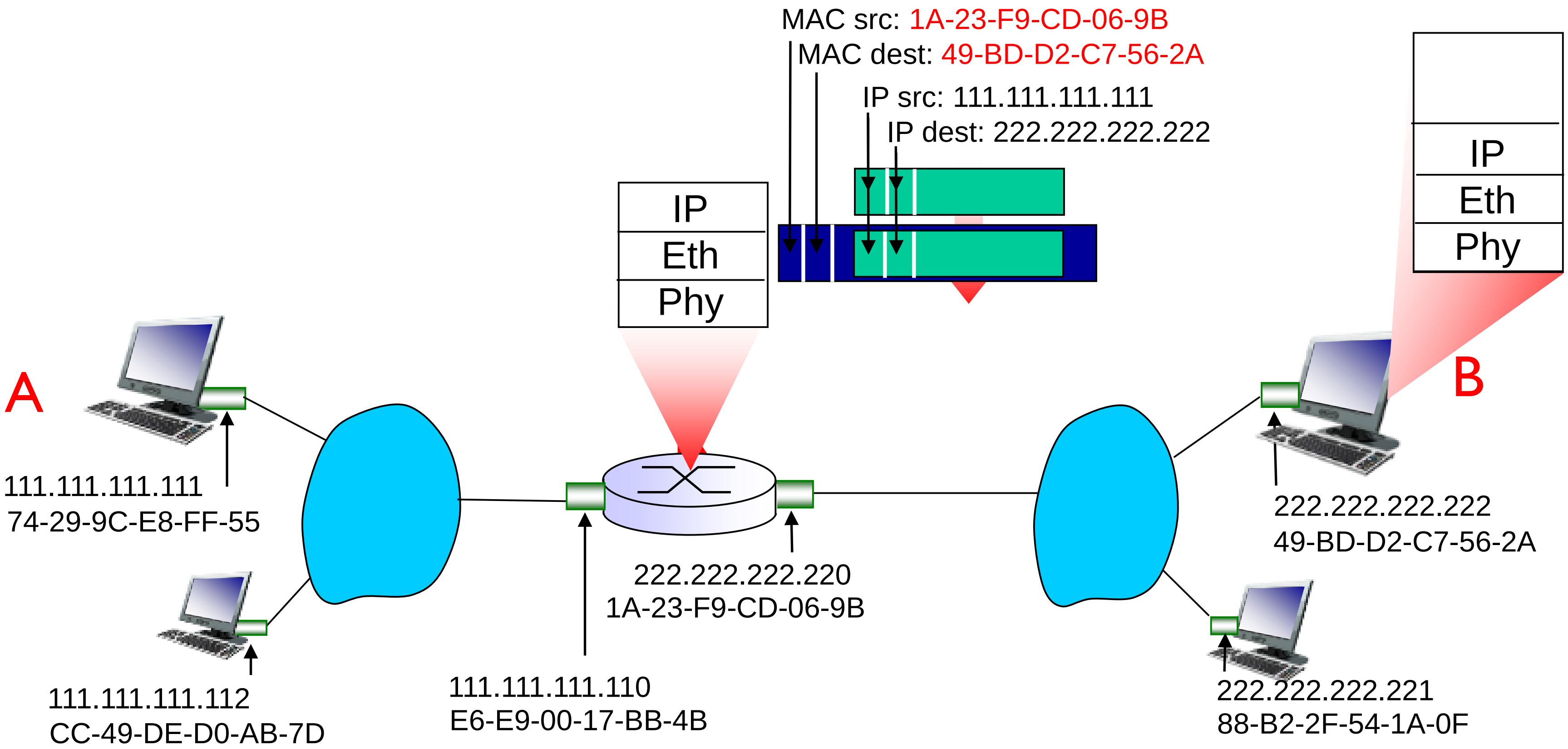
Addressing: routing to another LAN

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



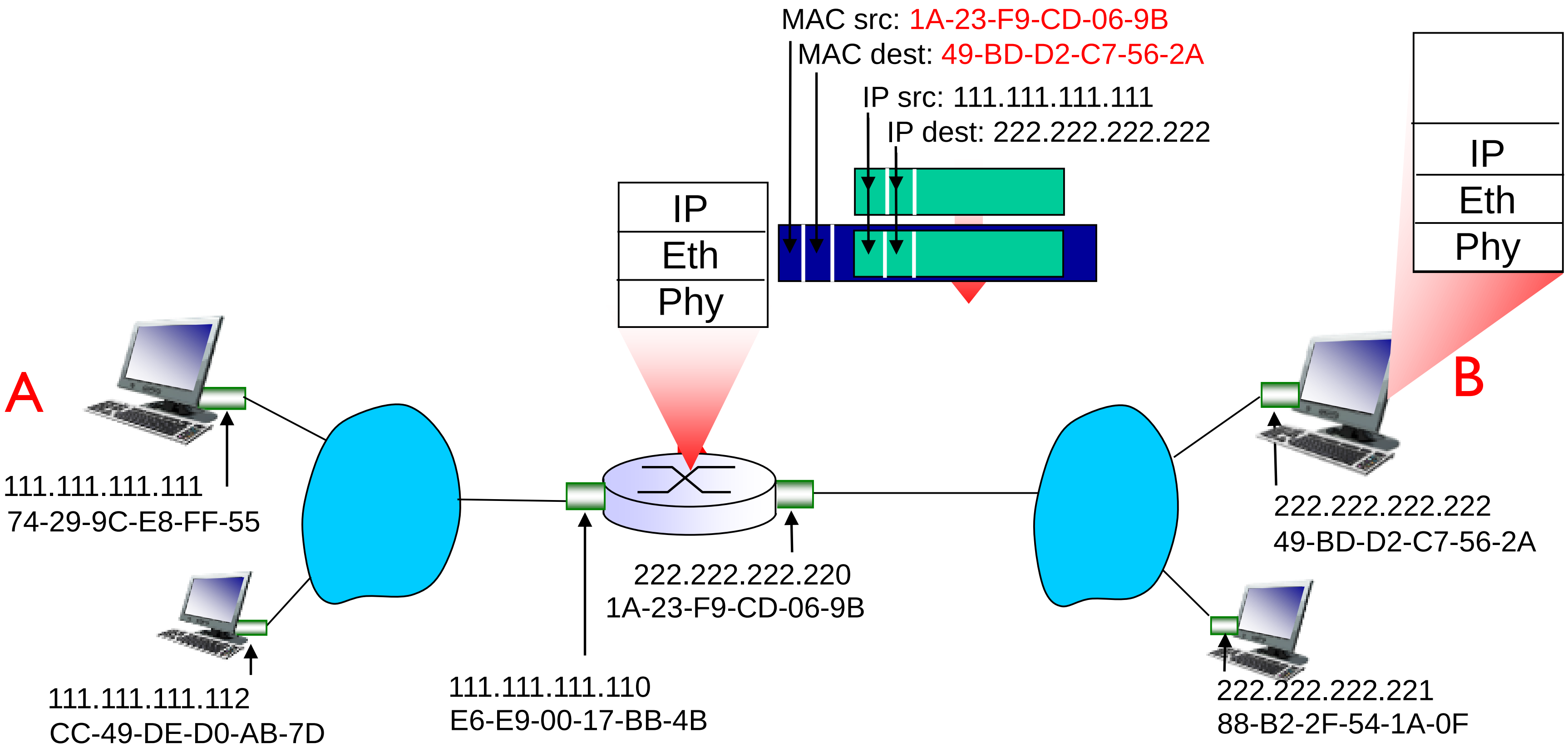
Addressing: routing to another LAN

- 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



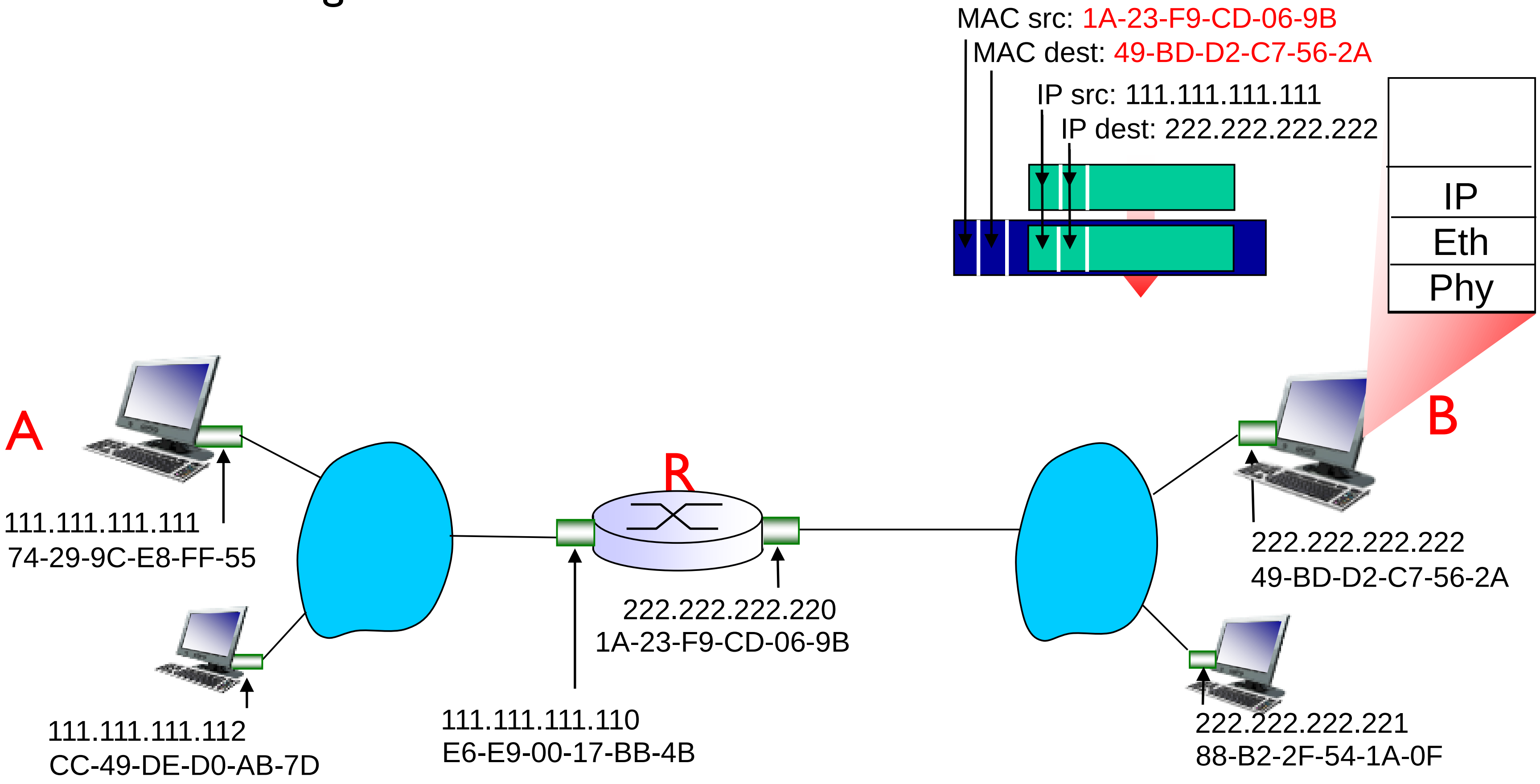
Addressing: routing to another LAN

- 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



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

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

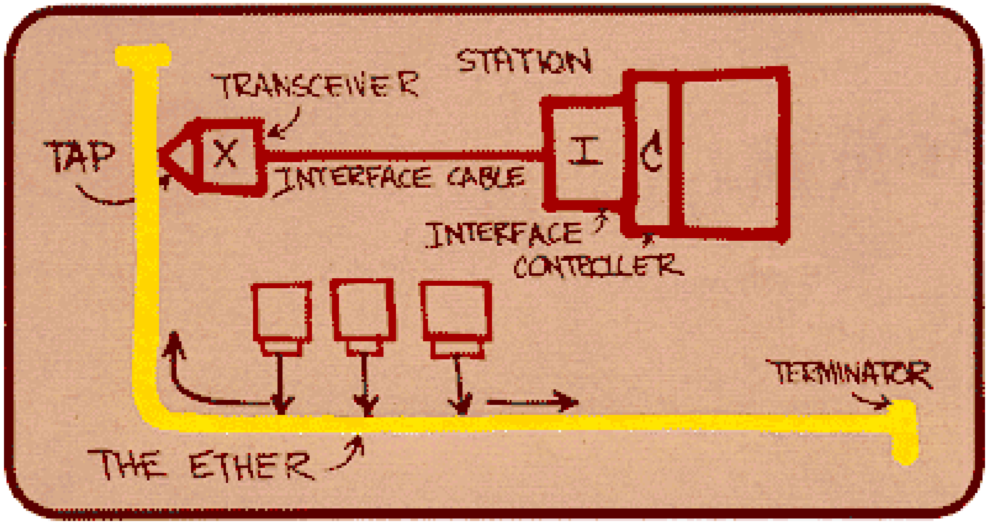
6.6 data center networking

6.7 a day in the life of a web request

Ethernet

"dominant" wired LAN technology:

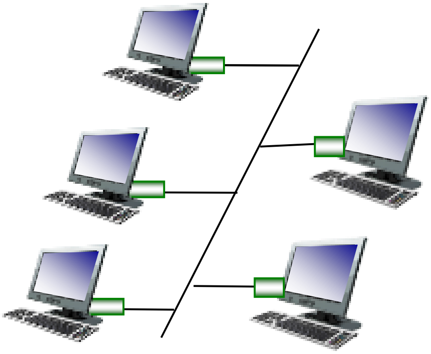
- single chip, multiple speeds (e.g., Broadcom BCM5761)
- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 10 Gbps



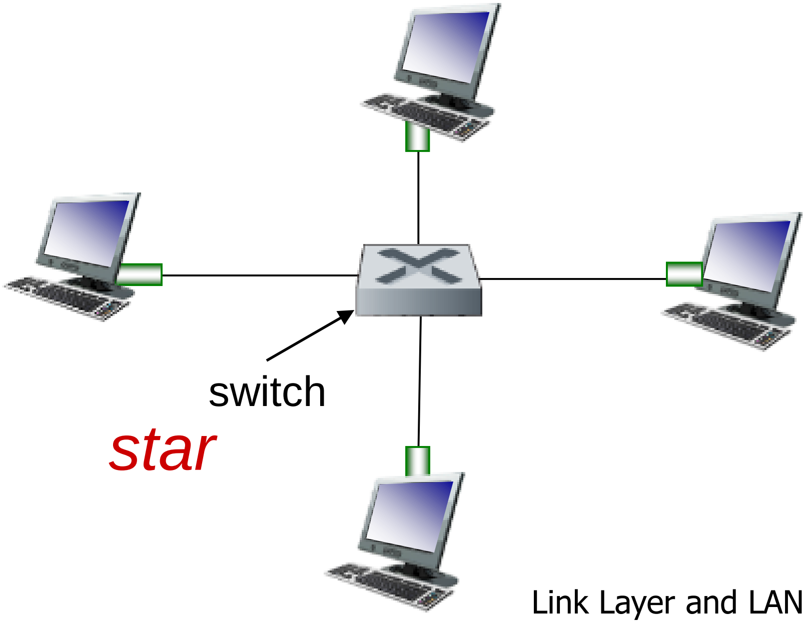
Metcalfe's Ethernet sketch

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)



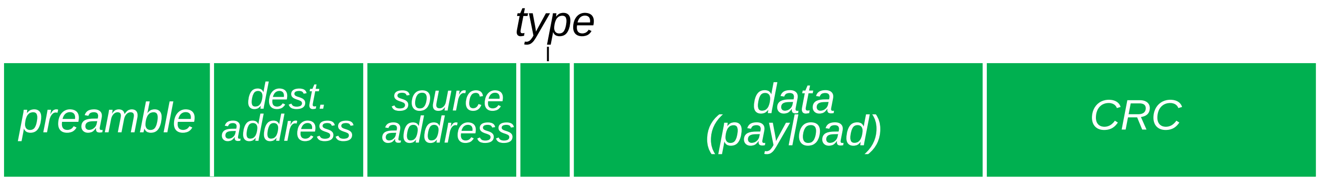
bus: coaxial cable



star

Ethernet frame structure

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

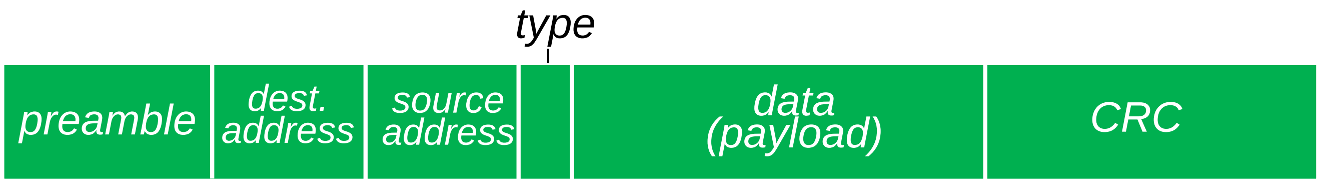


preamble:

- 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

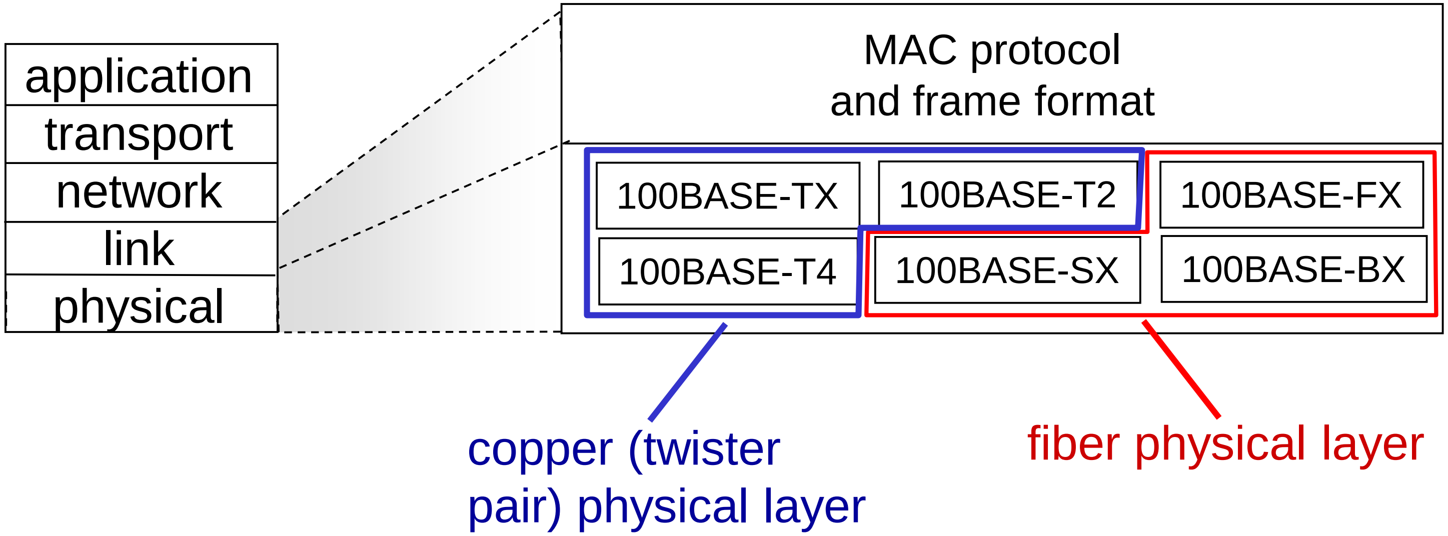


Ethernet: unreliable, connectionless

- *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 with 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, 10 Gbps, 40 Gbps
  - different physical layer media: fiber, cable



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

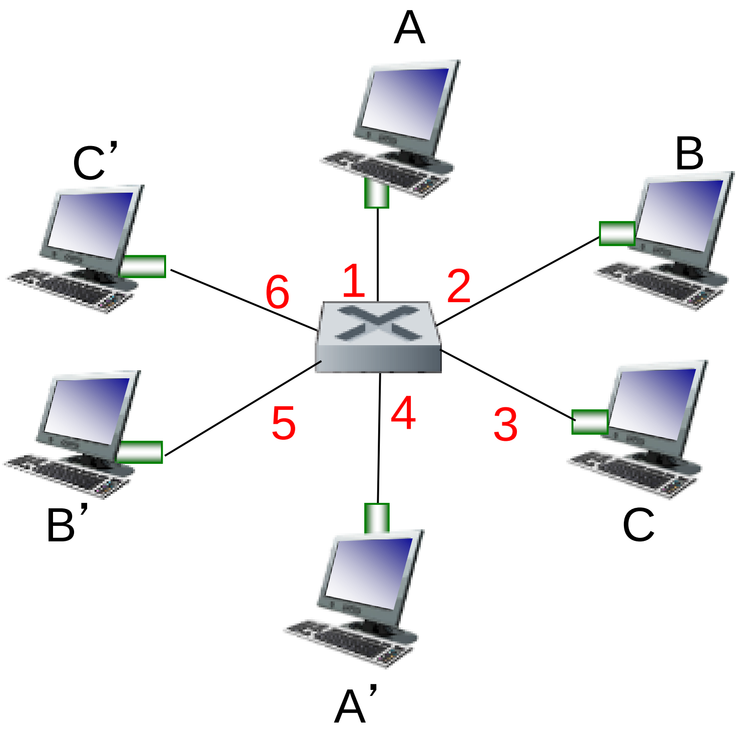
6.7 a day in the life of a web request

Ethernet switch

- link-layer device: takes an *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

Switch: *multiple* simultaneous transmissions

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

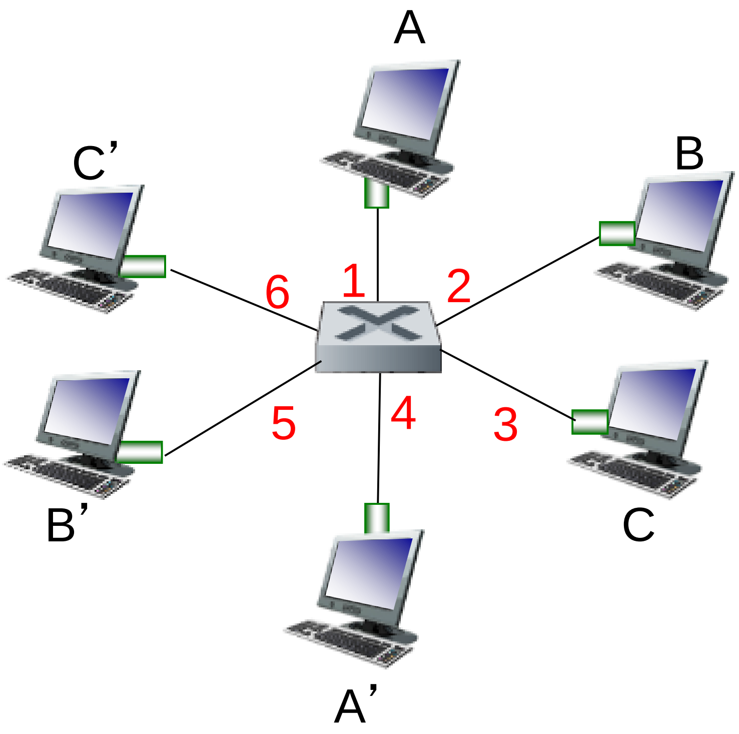


switch with six interfaces (1,2,3,4,5,6)

Switch forwarding table

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

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



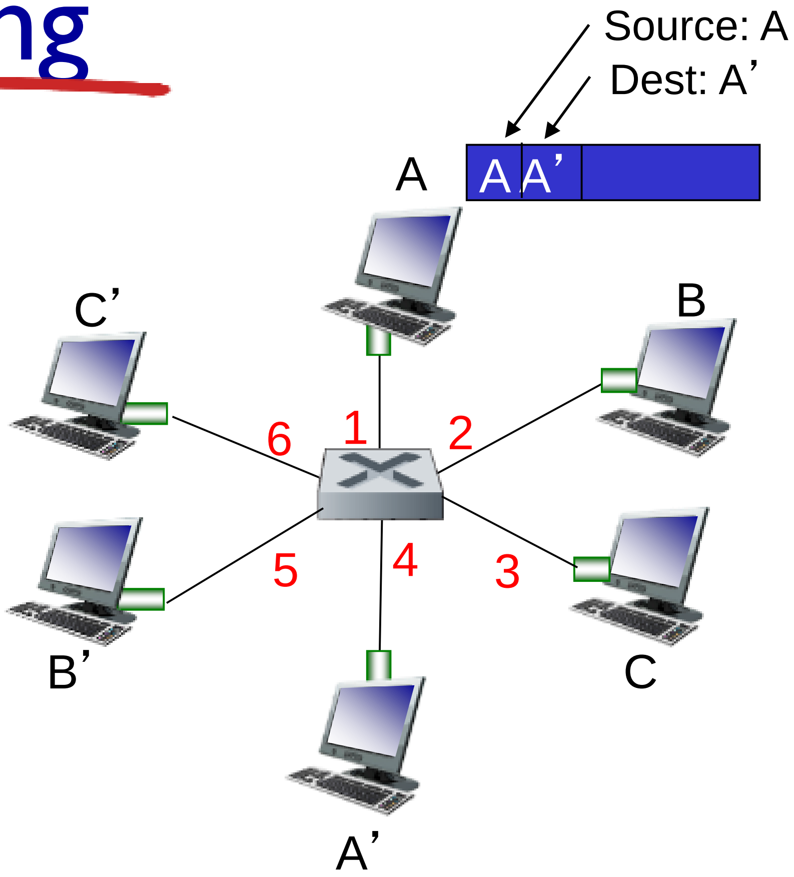
switch with six interfaces (1,2,3,4,5,6)

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

- something like a routing protocol?

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)

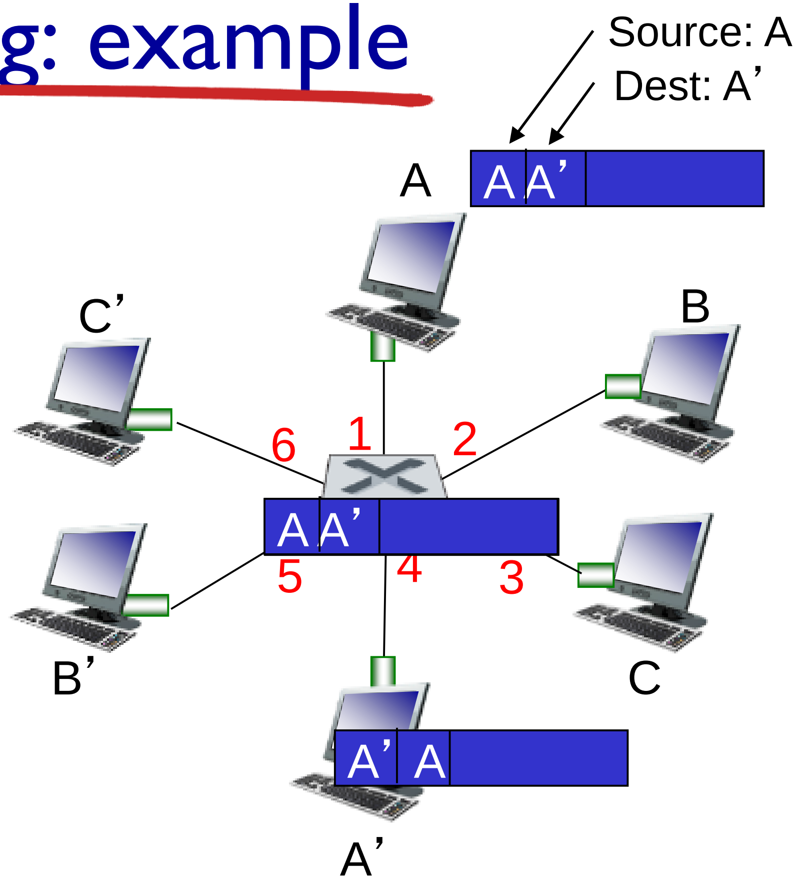
Switch: frame filtering/forwarding

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

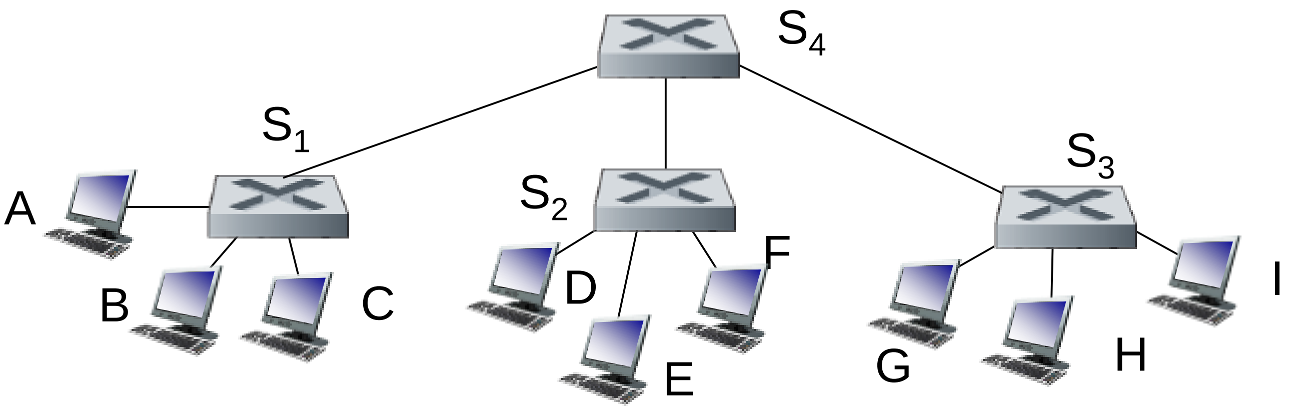


MAC addr	interface	TTL
A *A'*	*1* *4*	*60* *60*

switch table (initially empty)

Interconnecting switches

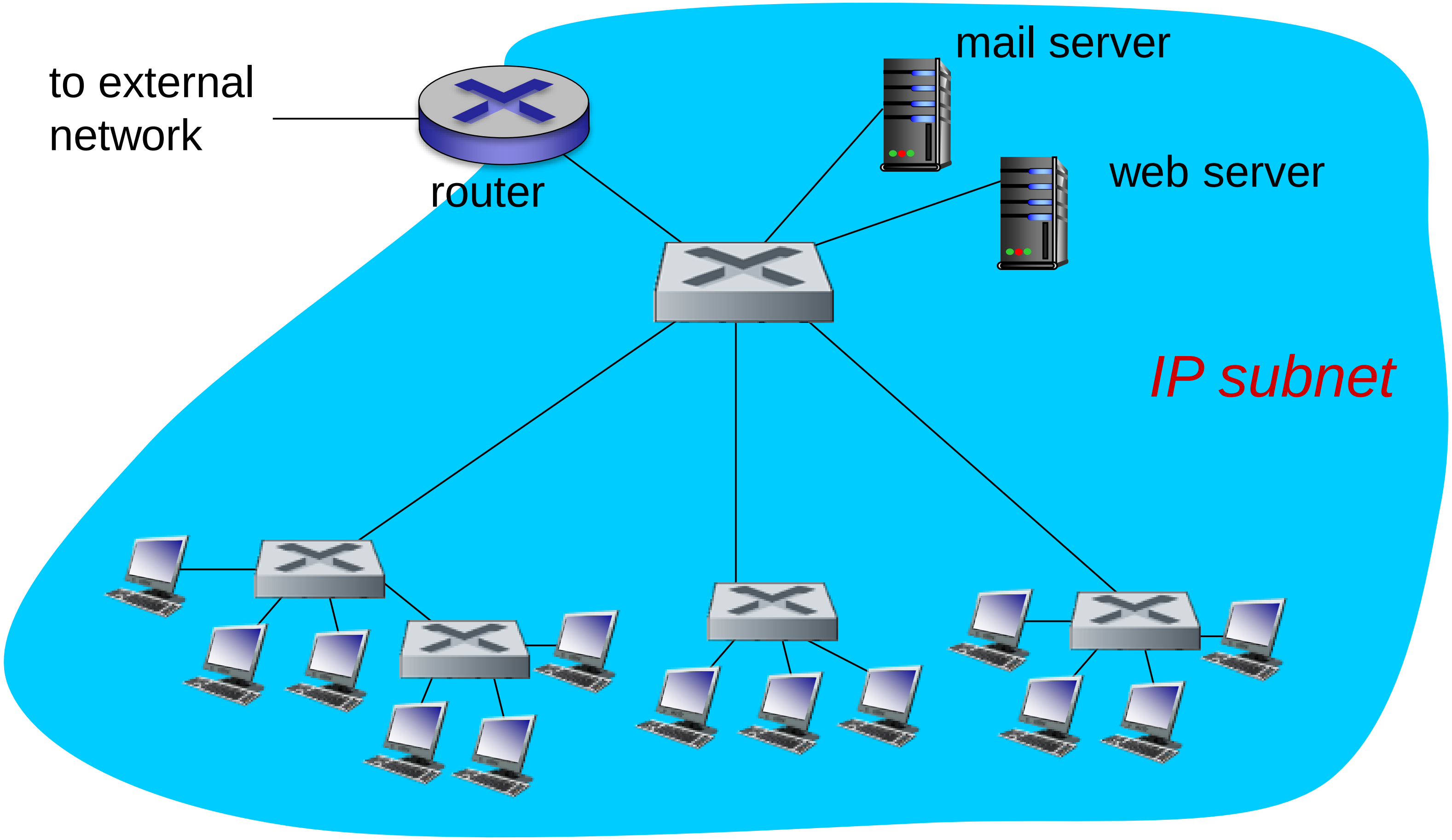
self-learning switches can be connected together:



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



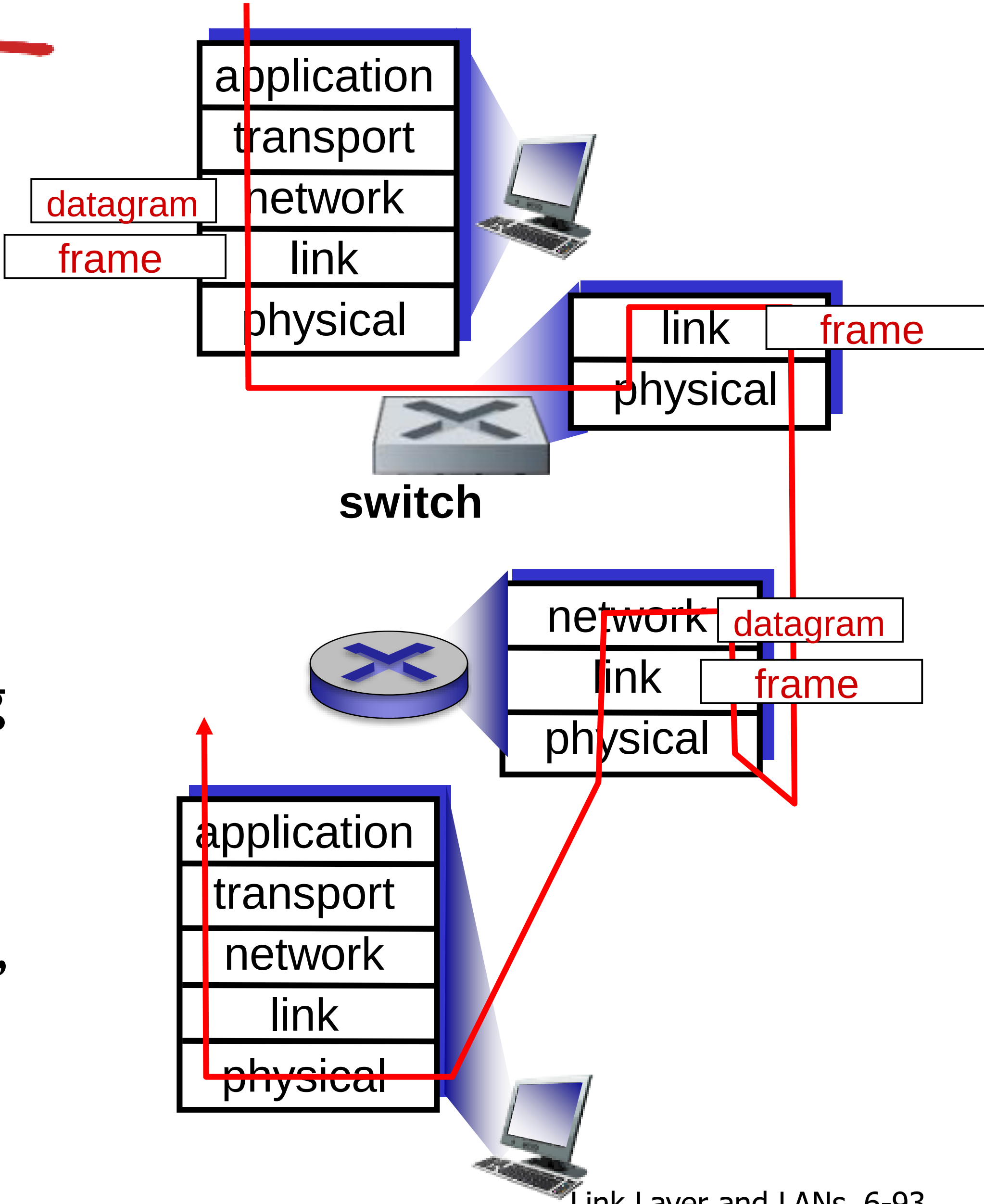
Switches vs. routers

both are store-and-forward:

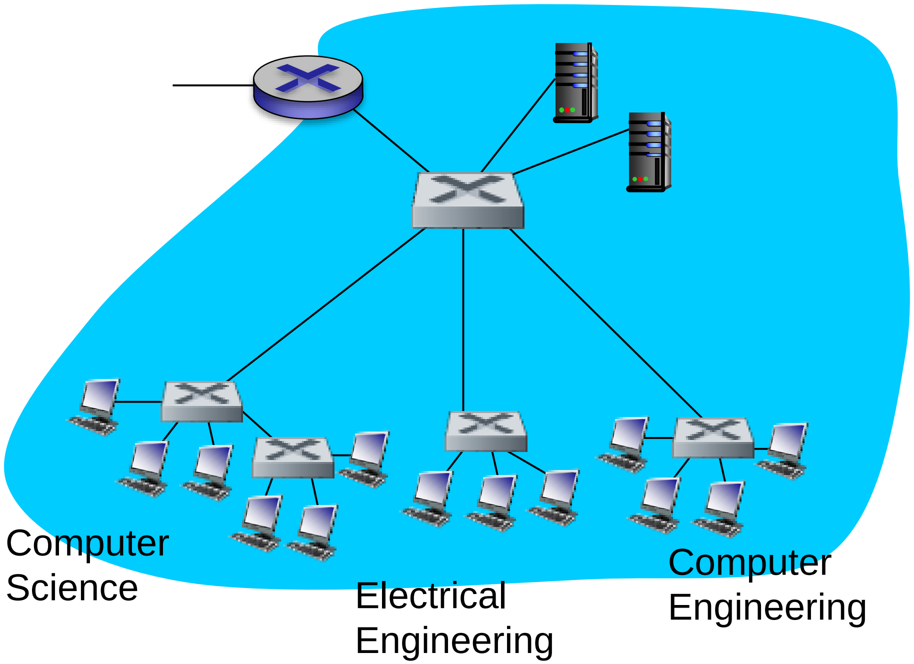
- *routers:* network-layer devices (examine network-layer headers)
- *switches*: link-layer devices (examine link-layer headers)

both have forwarding tables:

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



VLANs: motivation



consider:

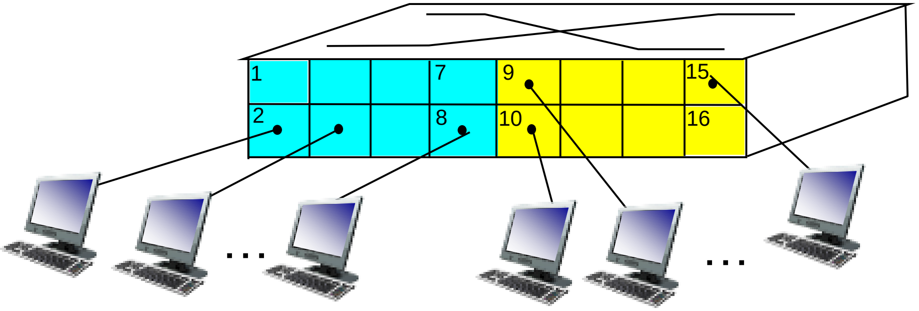
- CS user moves office to EE, but wants connect to CS switch?
- single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - security/privacy, efficiency issues

VLANs

Virtual Local Area Network

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

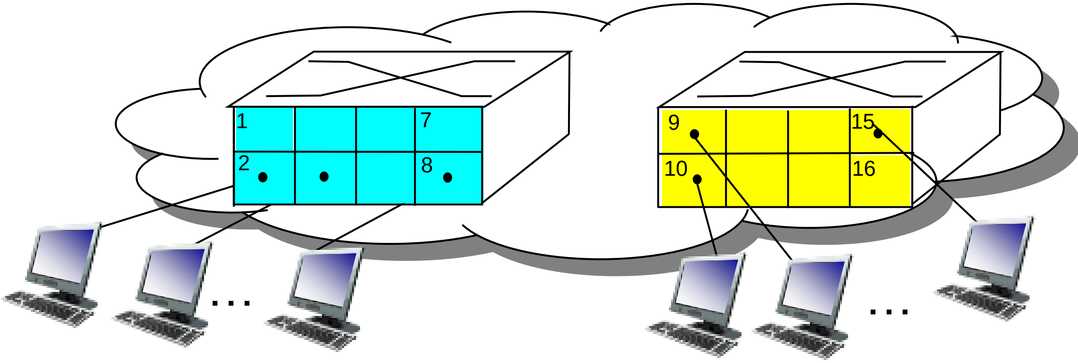
port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch ……



Electrical Engineering
(VLAN ports 1-8)

Computer Science
(VLAN ports 9-15)

… operates as multiple virtual switches

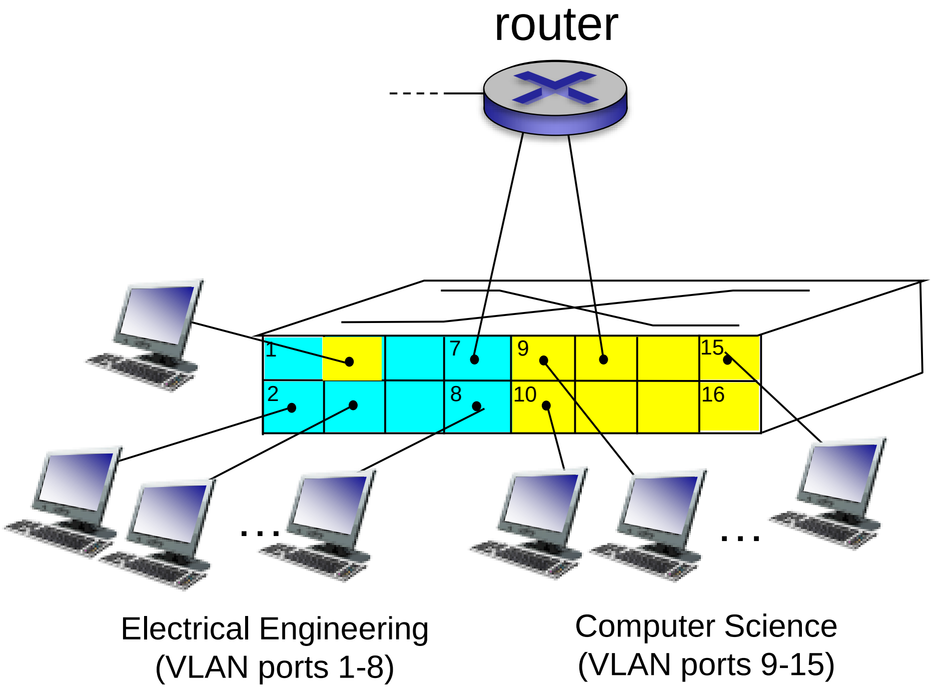


Electrical Engineering
(VLAN ports 1-8)

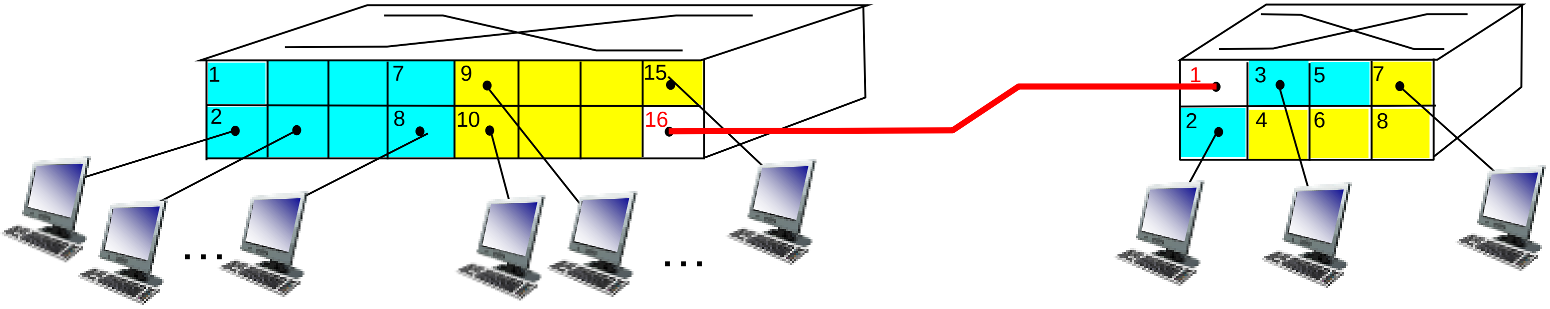
Computer Science
(VLAN ports 9-16)

Port-based VLAN

- *traffic isolation:* frames to/from ports 1-8 can *only* reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port
- dynamic membership: ports can be dynamically assigned among VLANs
- forwarding between VLANS: done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers



VLANS spanning multiple switches



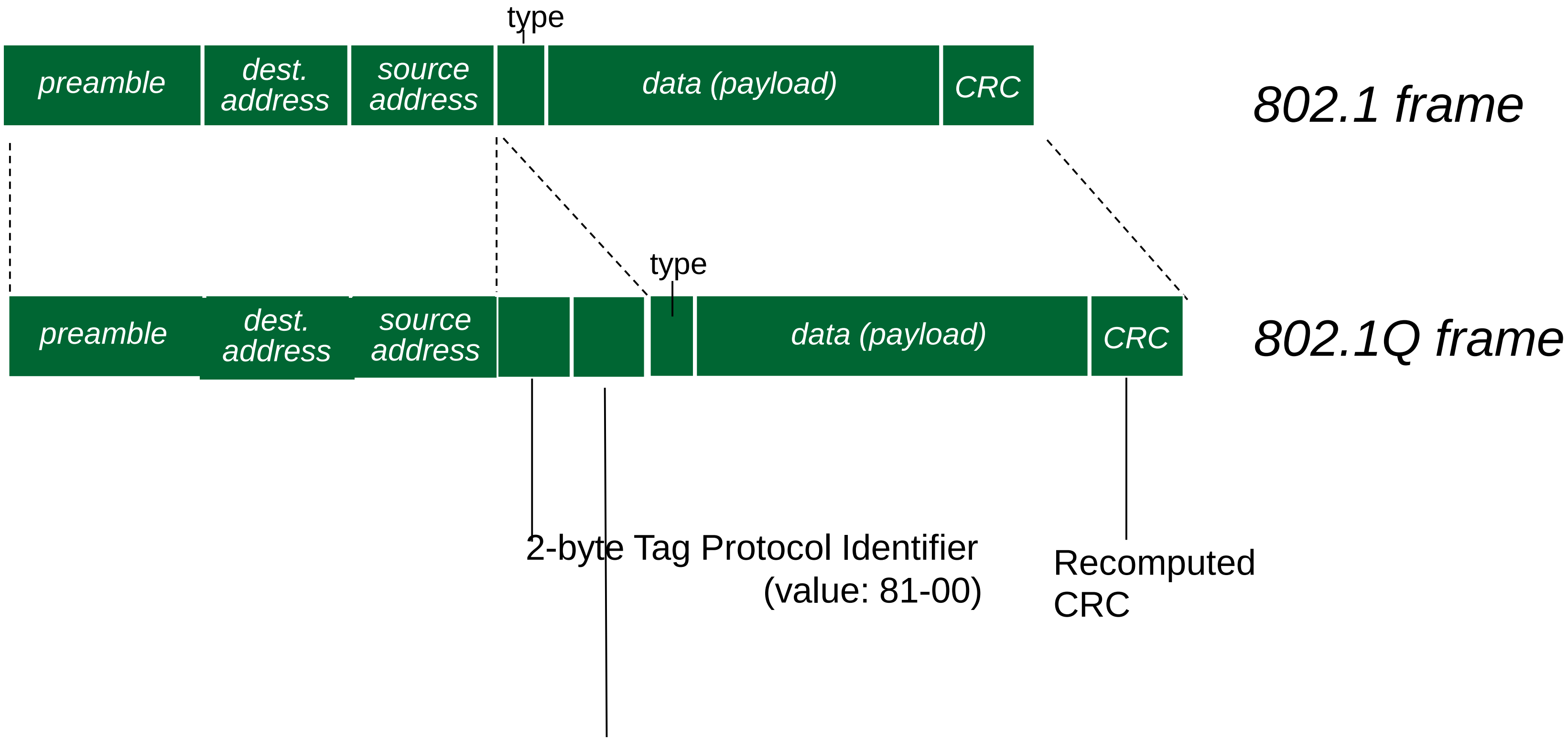
Electrical Engineering
(VLAN ports 1-8)

Computer Science
(VLAN ports 9-15)

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

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

802.1Q VLAN frame format



2-byte Tag Protocol Identifier
(value: 81-00)

Recomputed
CRC

Tag Control Information (12 bit VLAN ID field,
3 bit priority field like IP TOS)

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

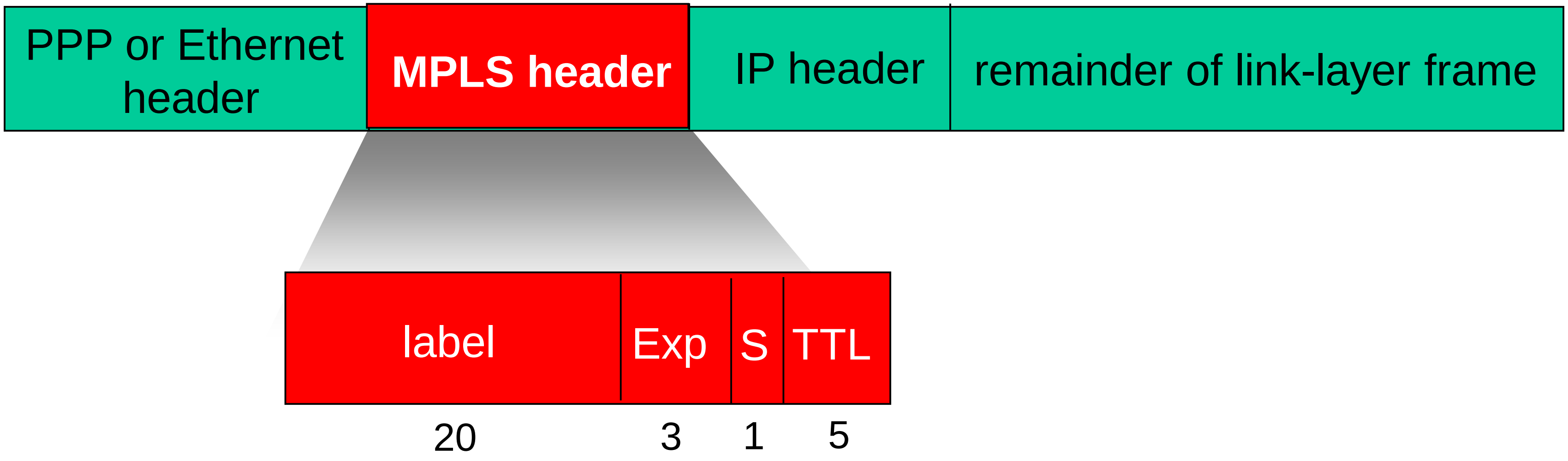
6.5 link virtualization: MPLS

6.6 data center networking

6.7 a day in the life of a web request

Multiprotocol label switching (MPLS)

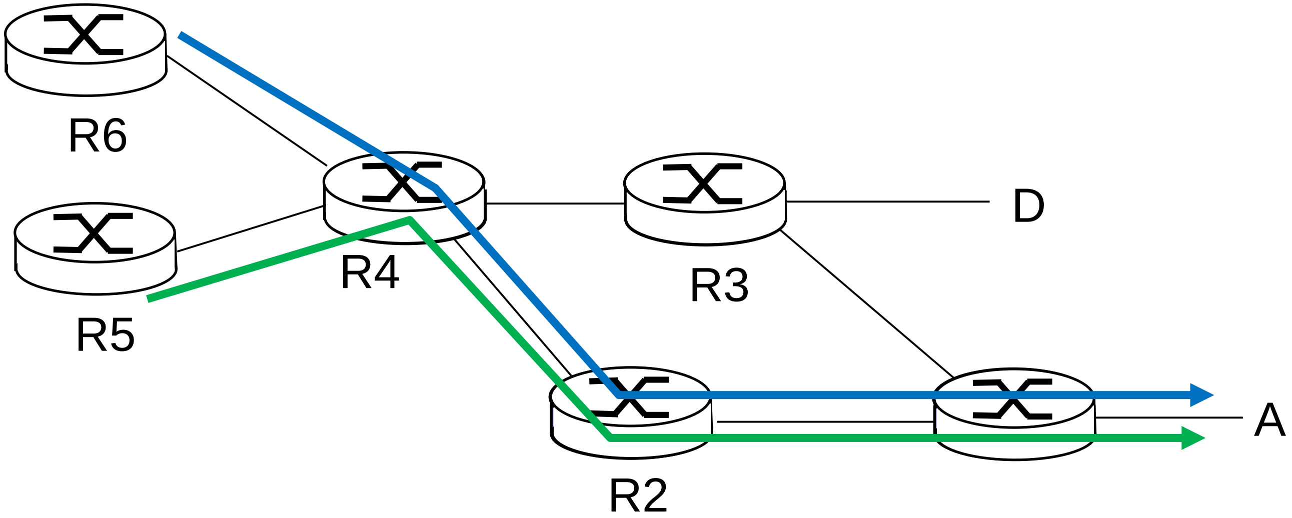
- initial goal: high-speed IP forwarding using fixed length label (instead of IP address)
  - fast lookup using fixed length identifier (rather than shortest prefix matching)
  - borrowing ideas from Virtual Circuit (VC) approach
  - but IP datagram still keeps IP address!



MPLS capable routers

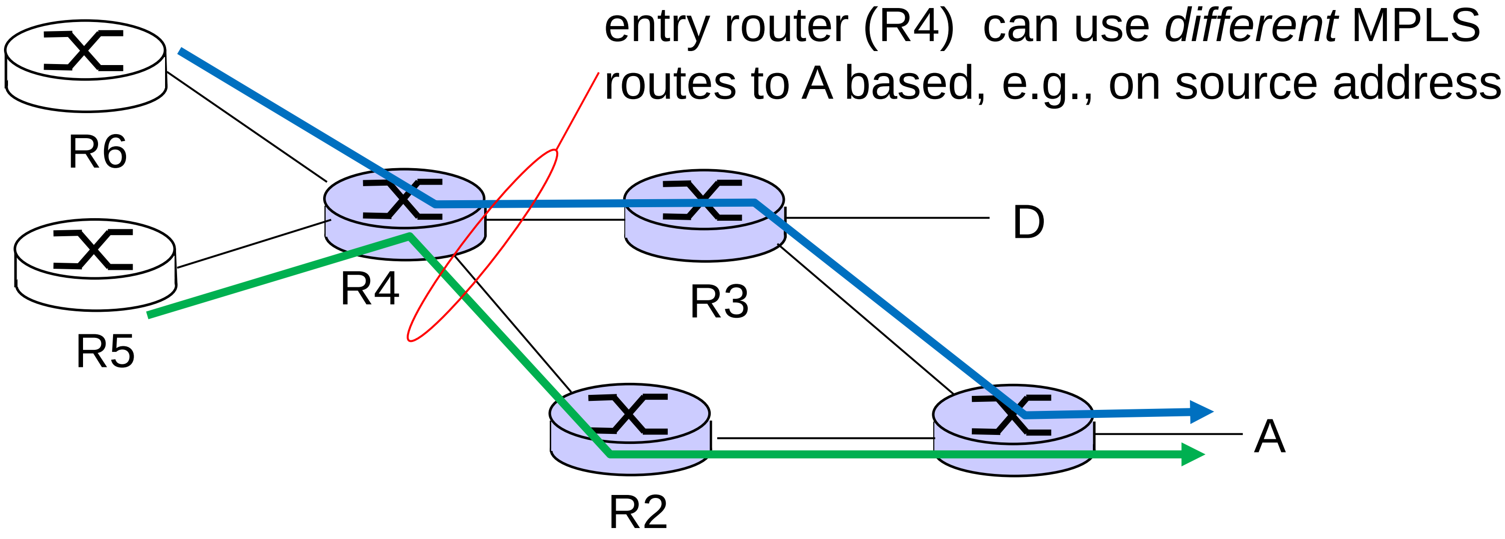
- a.k.a. label-switched router
- forward packets to outgoing interface based only on label value (*don't inspect IP address*)
  - MPLS forwarding table distinct from IP forwarding tables
- *flexibility:* MPLS forwarding decisions can *differ* from those of IP
  - use destination *and* source addresses 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



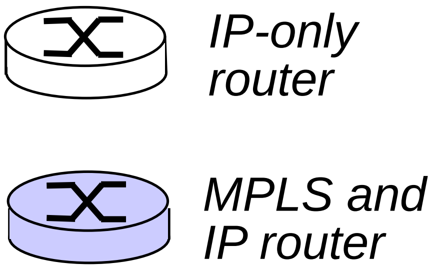
- *IP routing: path to destination determined by destination address alone*

MPLS versus IP paths



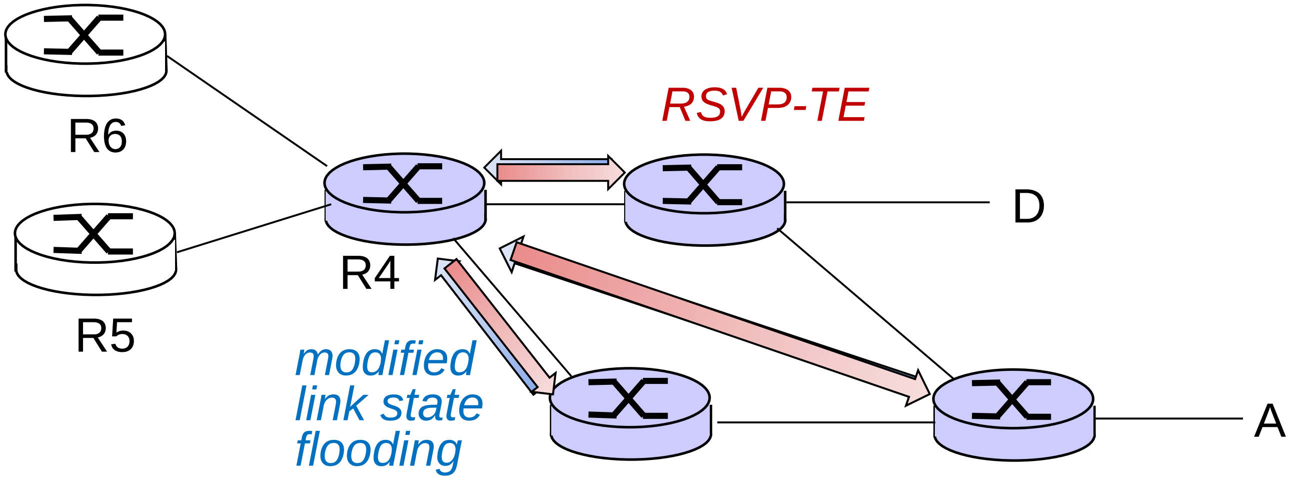
entry router (R4) can use *different* MPLS routes to A based, e.g., on source address

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



MPLS signaling

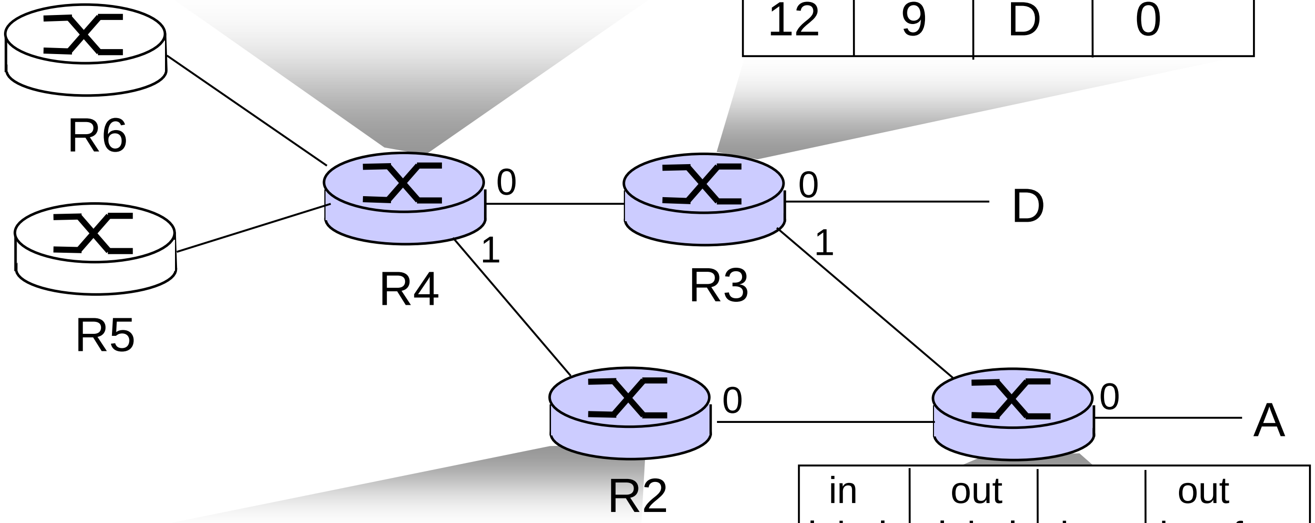
- modify OSPF, IS-IS link-state flooding protocols to carry info used by MPLS routing,
  - e.g., link bandwidth, amount of "reserved" link bandwidth
- *entry MPLS router uses RSVP-TE signaling protocol to set up MPLS forwarding at downstream routers*



MPLS forwarding tables

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

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



in label	out label	dest	out interface
8	6	A	0

in label	out label	dest	out interface
6	-	A	0

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

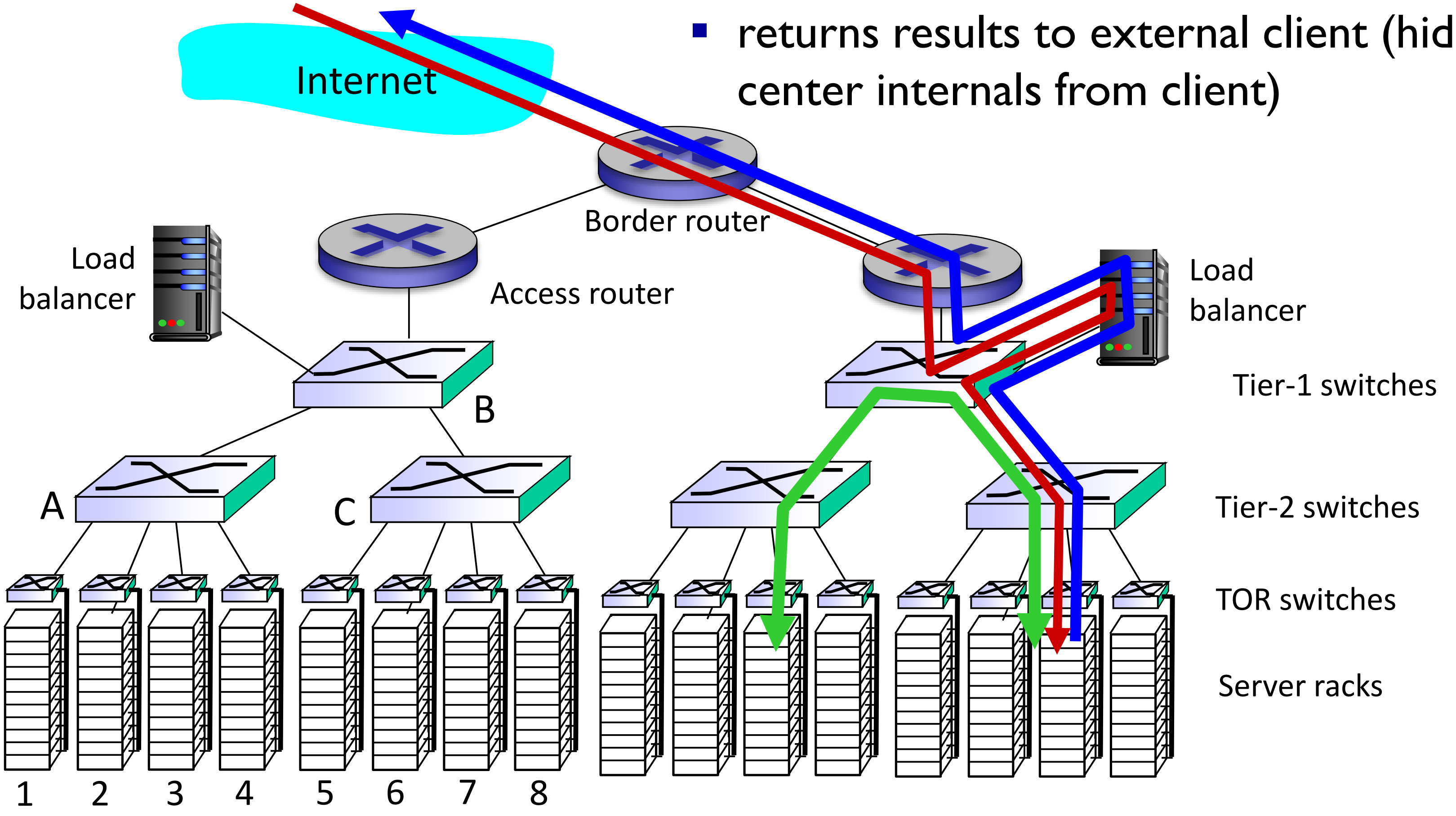
- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
  - e-business (e.g. Amazon)
  - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
  - search engines, data mining (e.g., Google)
- challenges:
  - multiple applications, each serving massive numbers of clients
  - managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Data center networks

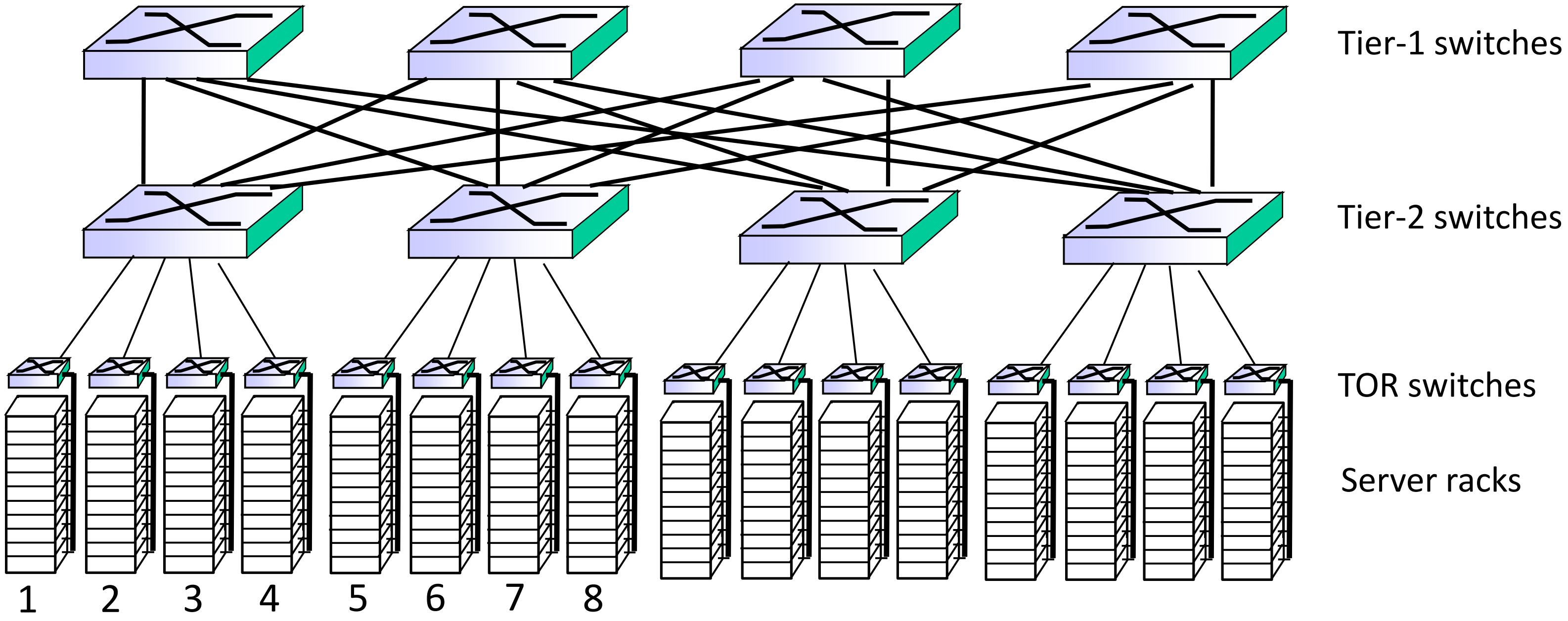
load balancer: application-layer routing

- 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



Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

64 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

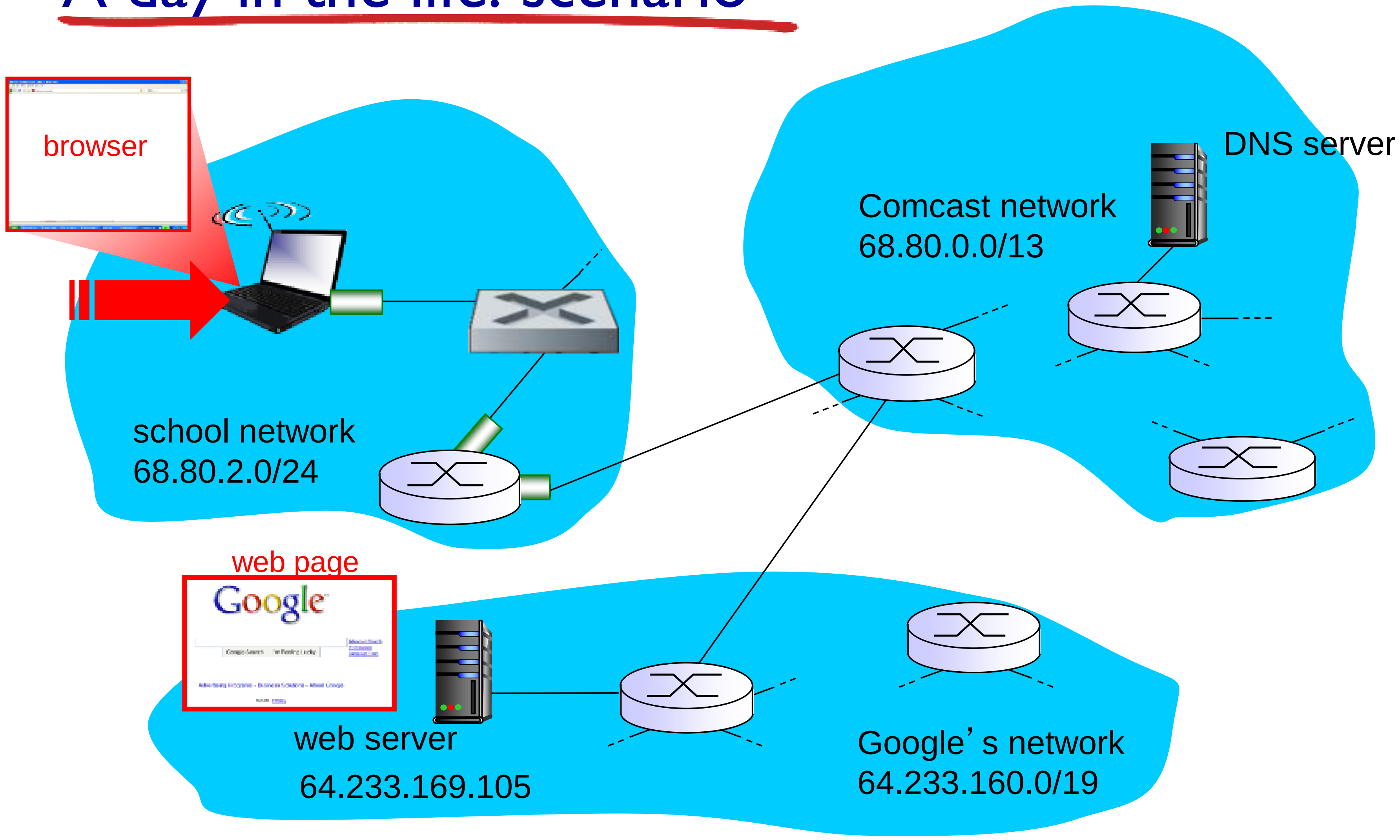
6.6 data center networking

6.7 a day in the life of a web request

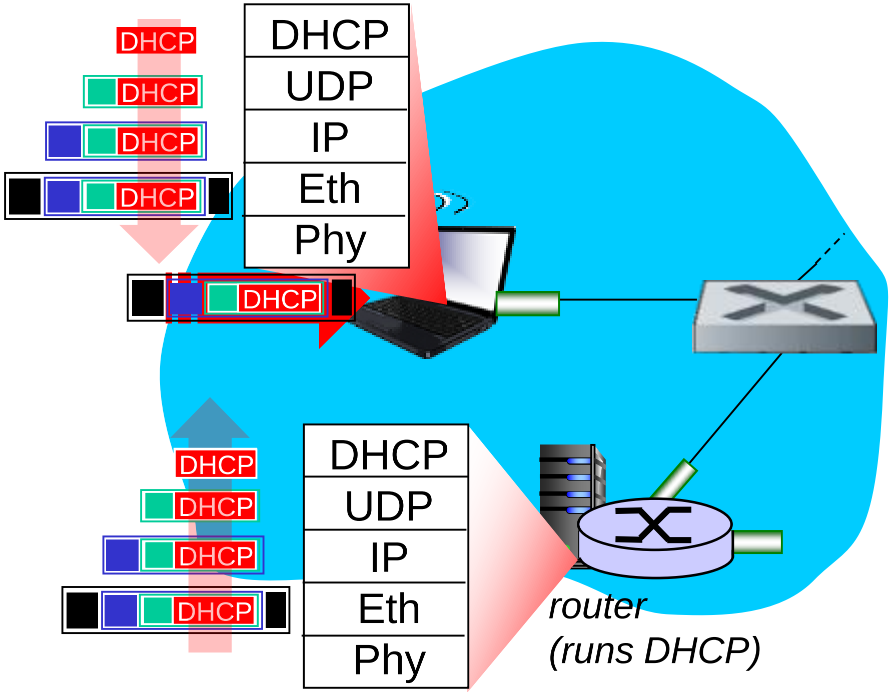
Synthesis: a day in the life of a web request

- 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

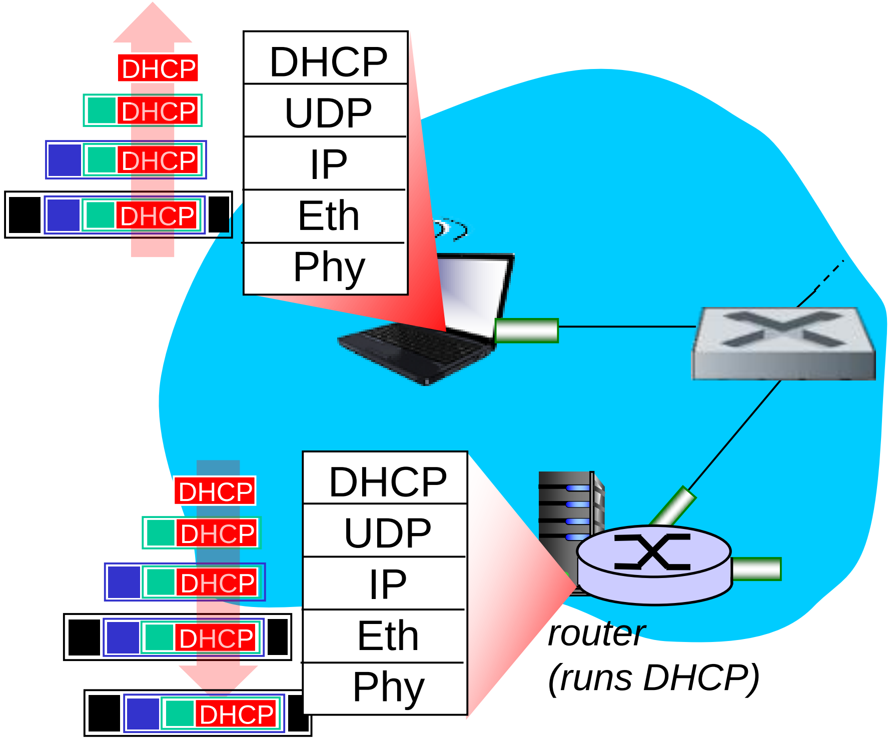


A day in the life… connecting to the Internet



- 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

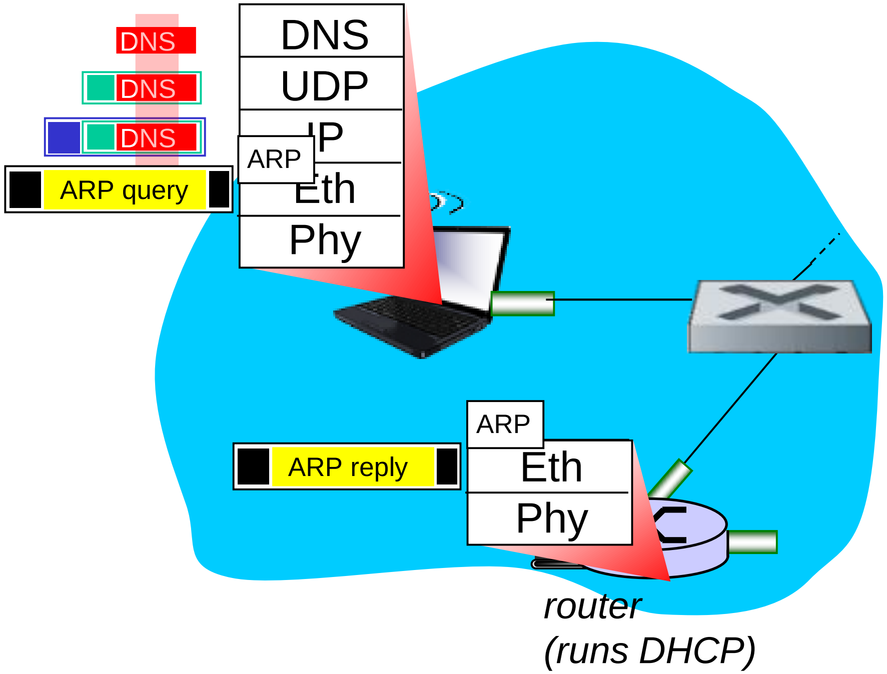
A day in the life… connecting to the Internet



- 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

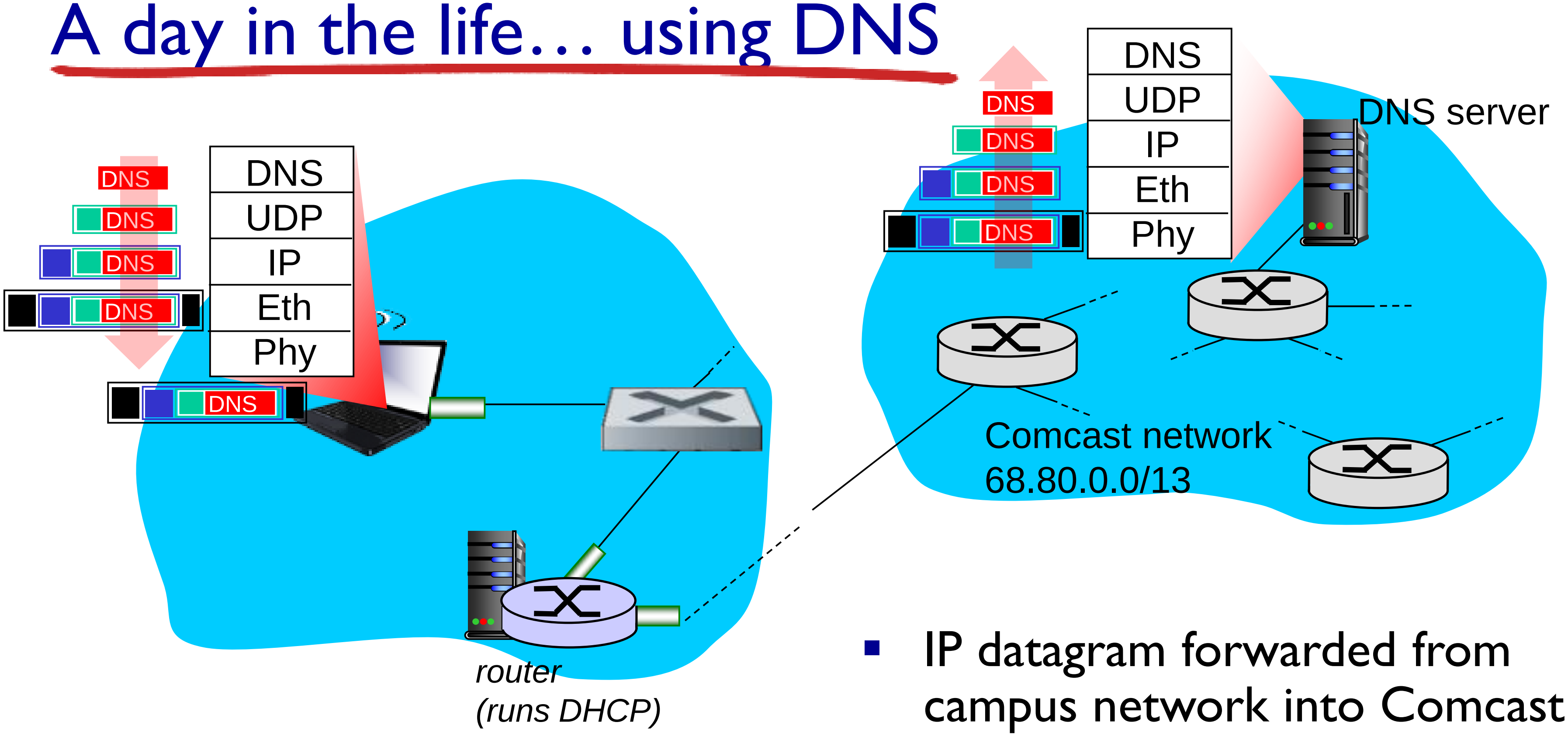
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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

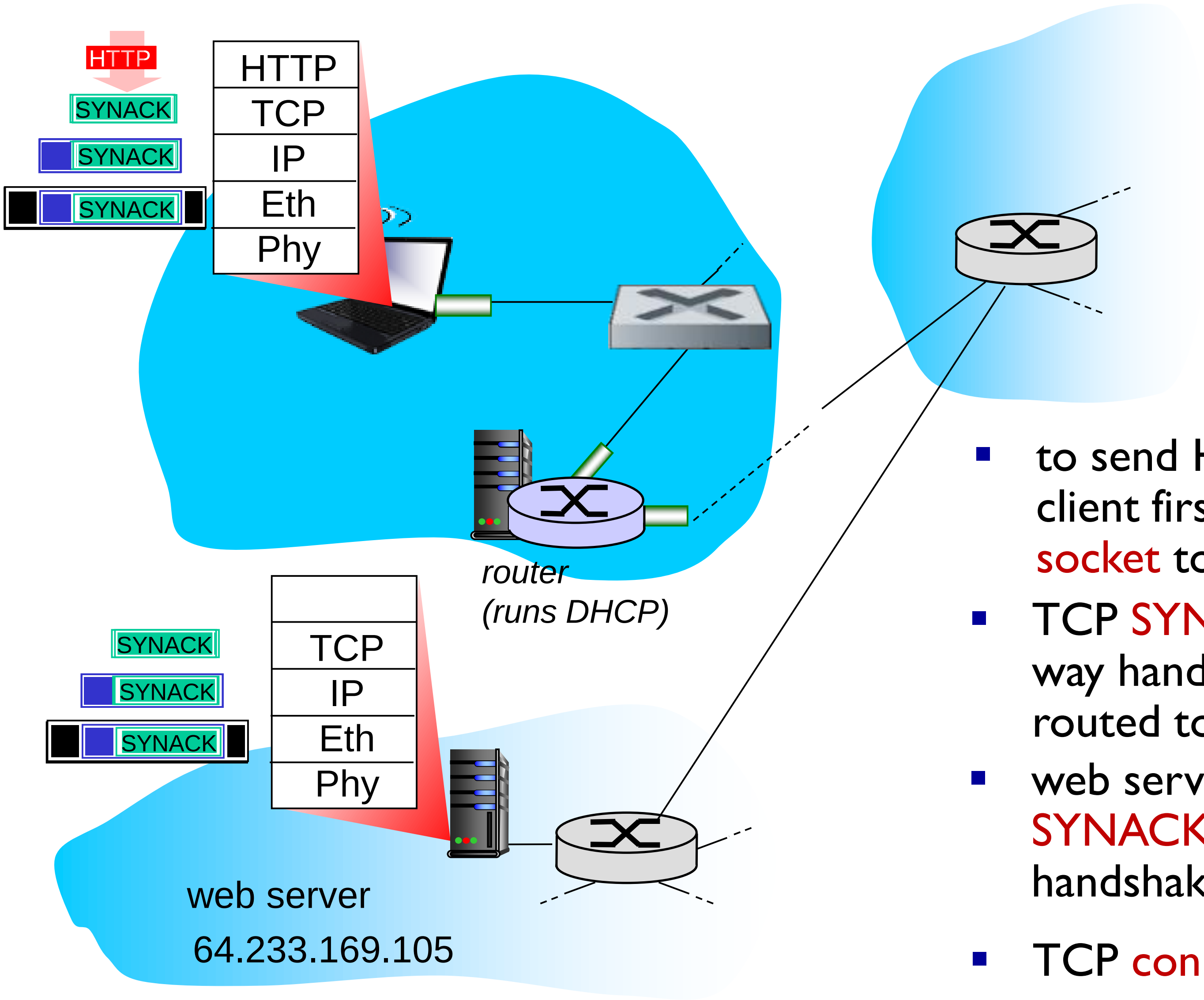
A day in the life… using DNS



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

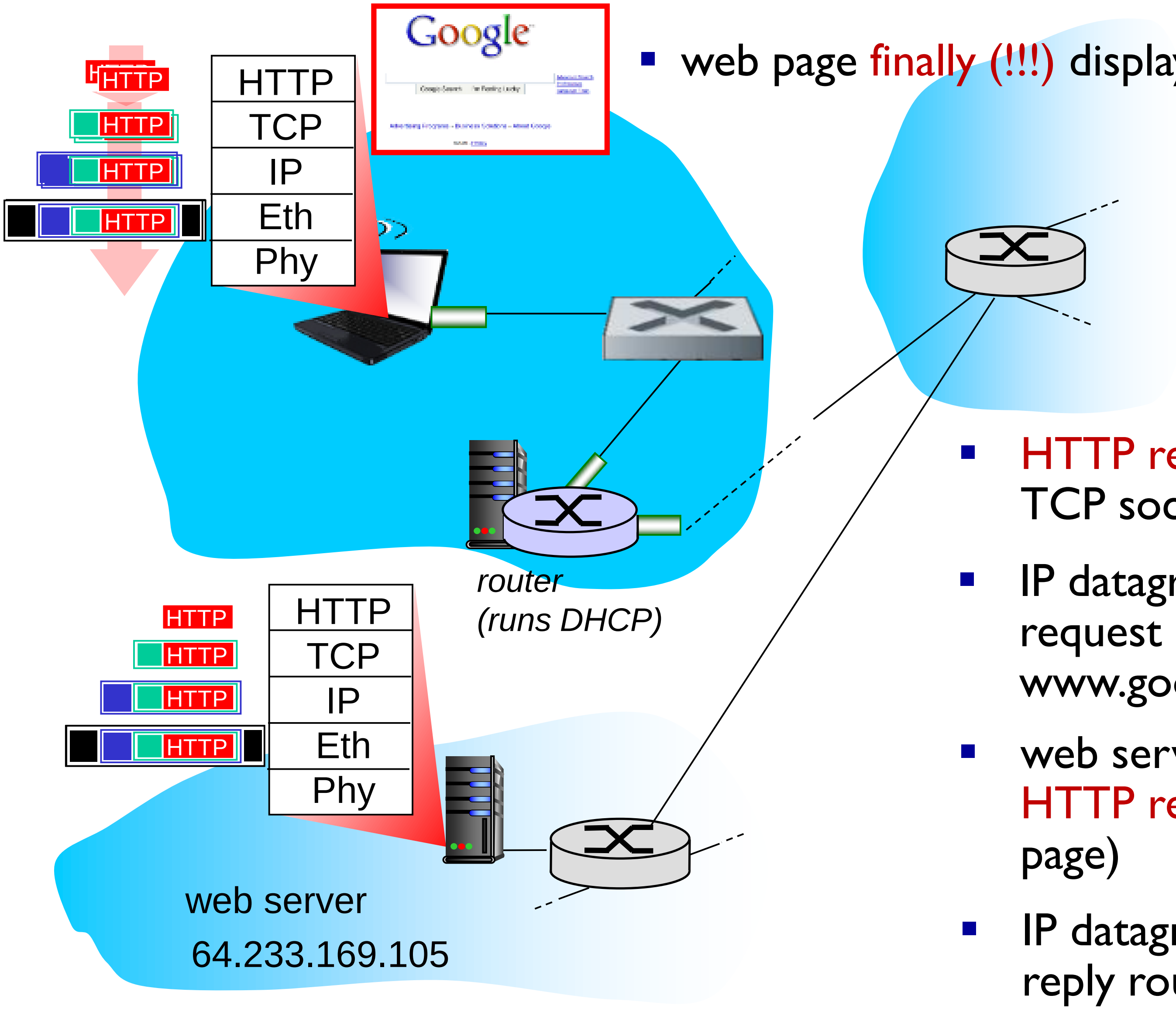
A day in the life…TCP connection carrying HTTP



- 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