Introduction to Networking

CAN201 – Lecture 10

Lecturer: Dr. Wenjun Fan

Lecture 10 – Link Layer (2)

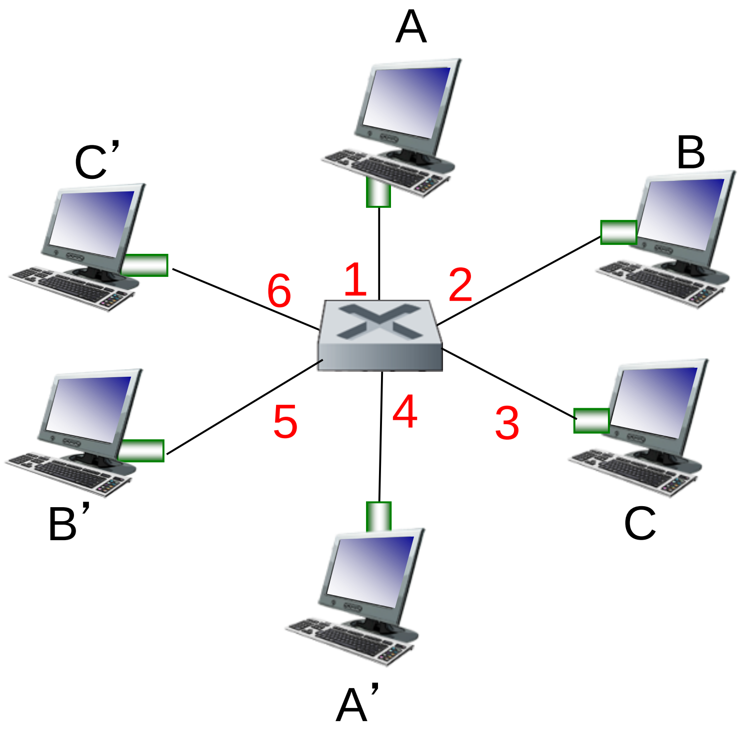
- **Roadmap**
  1. Switches
  2. VLANs
  3. Data center networking

Ethernet switch

- **Link-layer device (layer-2 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/link
- **Transparent**
  - Hosts are unaware of presence of switches
- **Plug-and-play, self-learning**
  - Switches do not need to be configured

Switch: multiple simultaneous transmissions

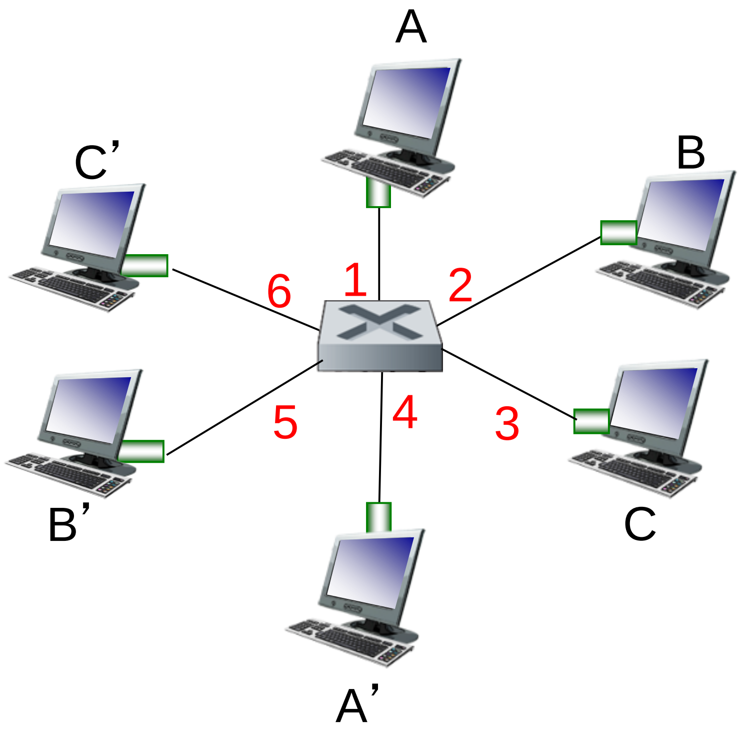
- **Hosts have dedicated, direct link to switch**
- **Switches buffer packets**
- **Ethernet protocol used on *each* incoming link, but no collisions; full duplex**
  - Each link is its own collision domain
- ***Switching, e.g.,:* A-to-A' and B-to-B' can transmit simultaneously, without collisions**



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

Switch (forwarding) table

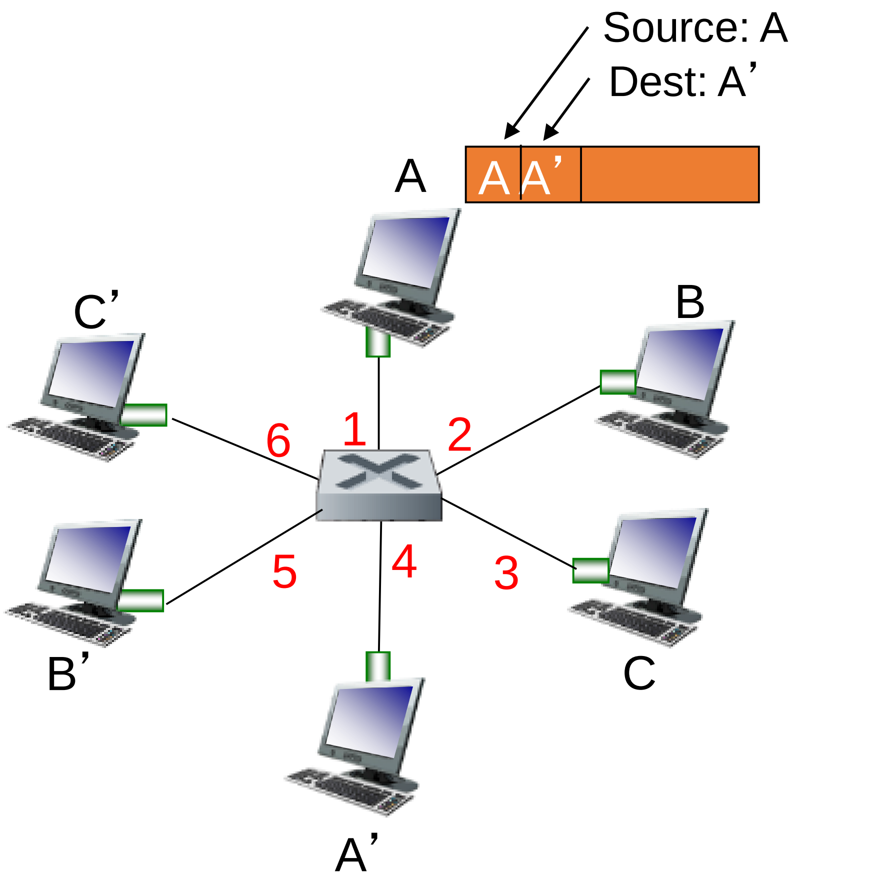
- ***Q:*** **how does switch know A' reachable via interface 4, B' reachable via interface 5?**
- *A:* each switch has a switch table, which contains entries, and each entry has:
  - (1) A MAC address, (2) the switch interface toward that MAC address, (3) the time stamp that the entry was placed.
  - 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)

Self-learning

- **Switch *learns* which hosts can be reached through which interfaces**
  - When frame received, switch “learns” location of sender: incoming LAN segment/link
  - Records MAC/interface (sender/location) pair in switch table
  - Entries get removed if no frames with that MAC received after timeout (which can be configured)

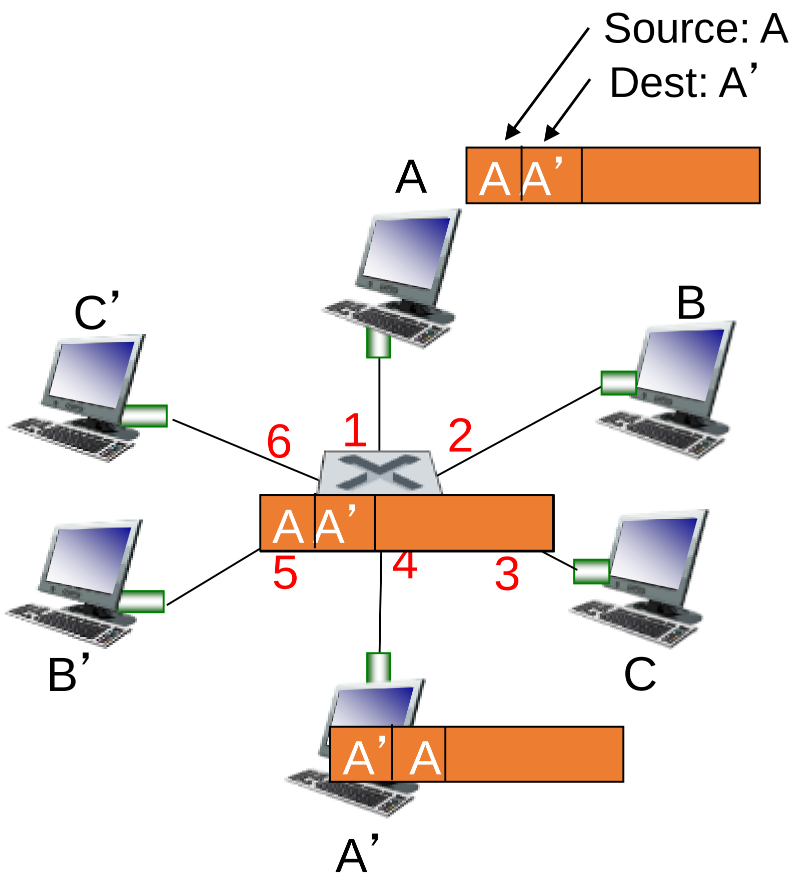


MAC addr	interface	Time
A	*1*	*9:10*

Switch table (initially empty)

Self-learning

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



MAC addr	interface	Time
A	*1*	*9:10*
A'	*4*	*9:15*

switch table (initially empty)

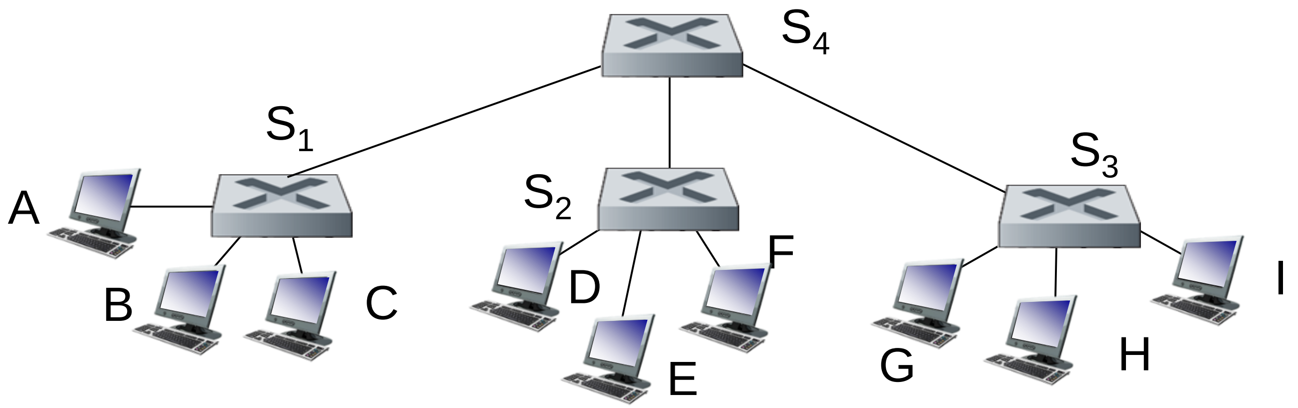
Switch: frame filtering/forwarding algorithm

When frame received at switch:

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

Interconnecting switches

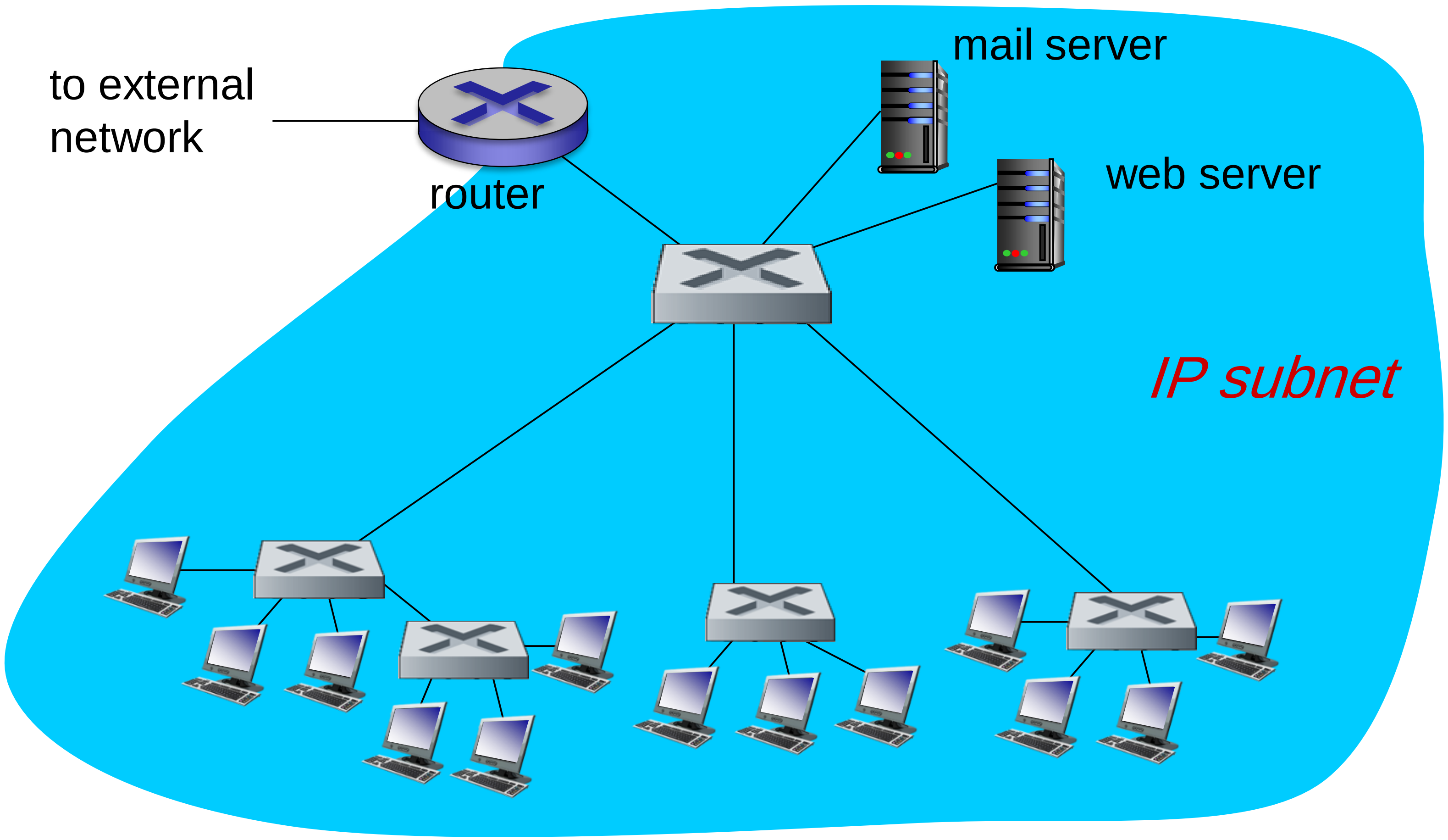
- Self-learning switches can be connected together:



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



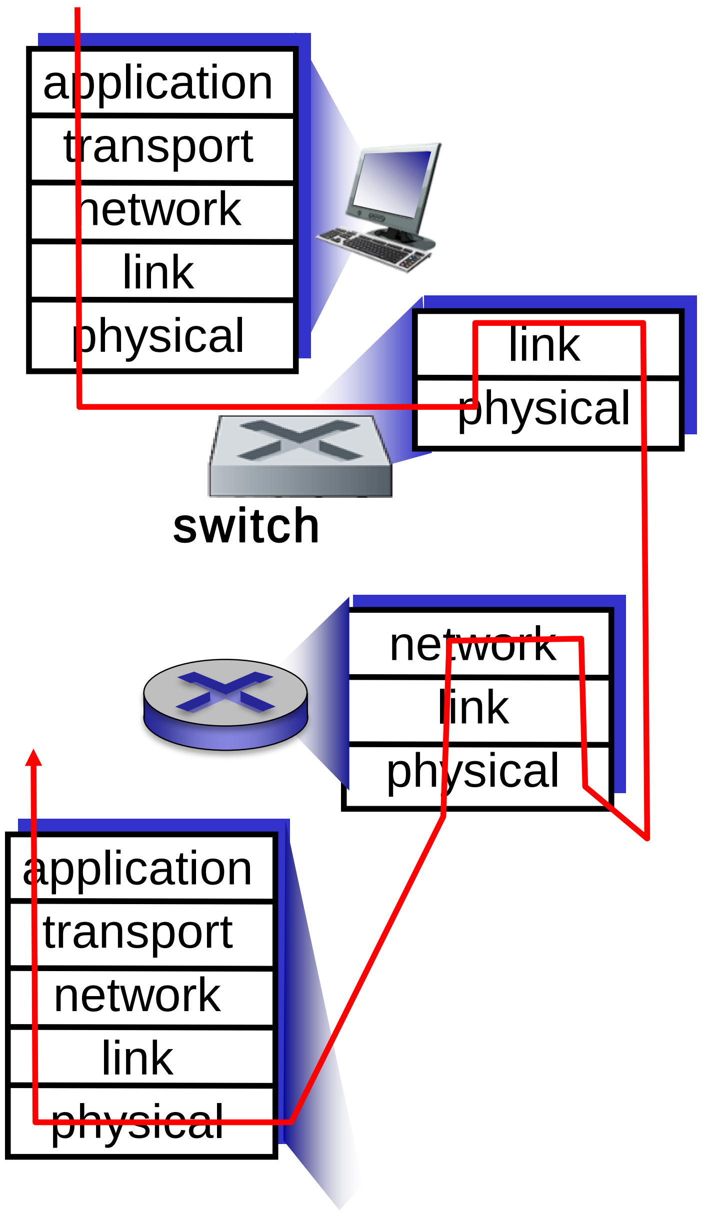
Switches vs. routers

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:

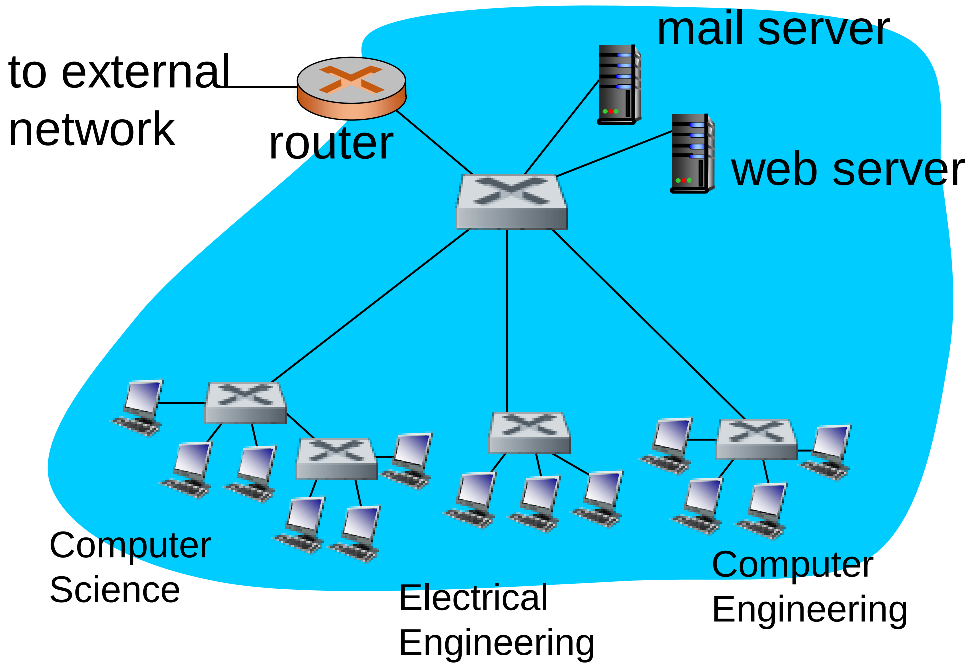
- ***Routers:*** compute tables using routing algorithms, IP addresses
- ***Switches:*** learn forwarding table using flooding, learning, MAC addresses



Lecture 10 – Link Layer (2)

- **Roadmap**
  1. Switches
  2. VLANs
  3. Data center networking

Virtual Local Area Network - Motivation



Consider drawbacks:

- **Lack of traffic isolation (single broadcast domain):**
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN.
  - security/privacy, efficiency issues.
  - This can be solved by replacing the center switch in the figure with a router. We'll see another solution via using a switch

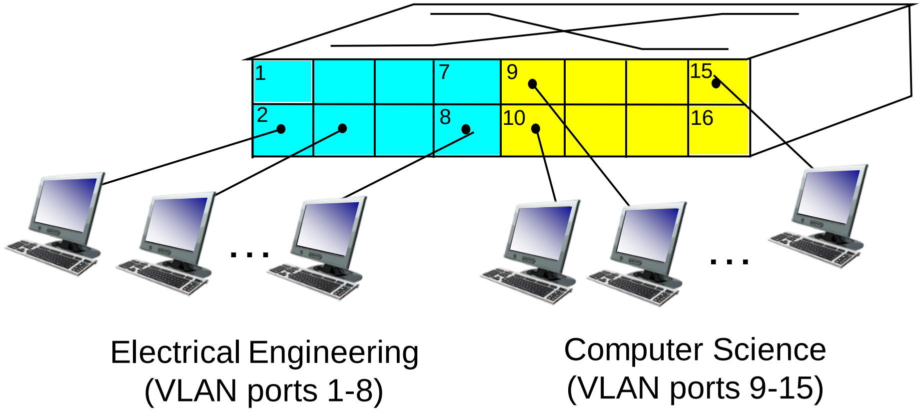
- **Difficult for managing users:**
  - CS user moves office to EE (in a different building), but wants connect to CS switch?

Virtual Local Area Network - VLAN

- **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 ……

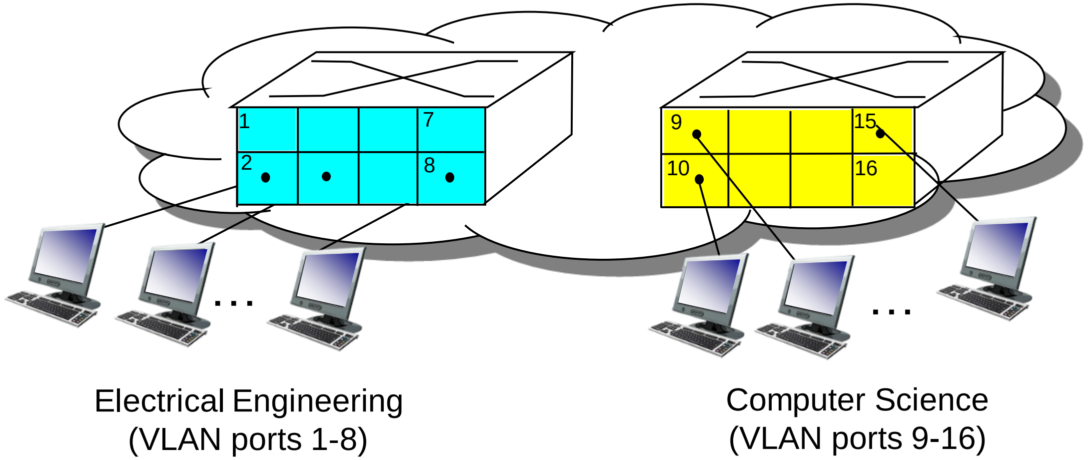


Virtual Local Area Network - VLAN

- **Virtual Local Area Network:**

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

… operates as multiple virtual switches

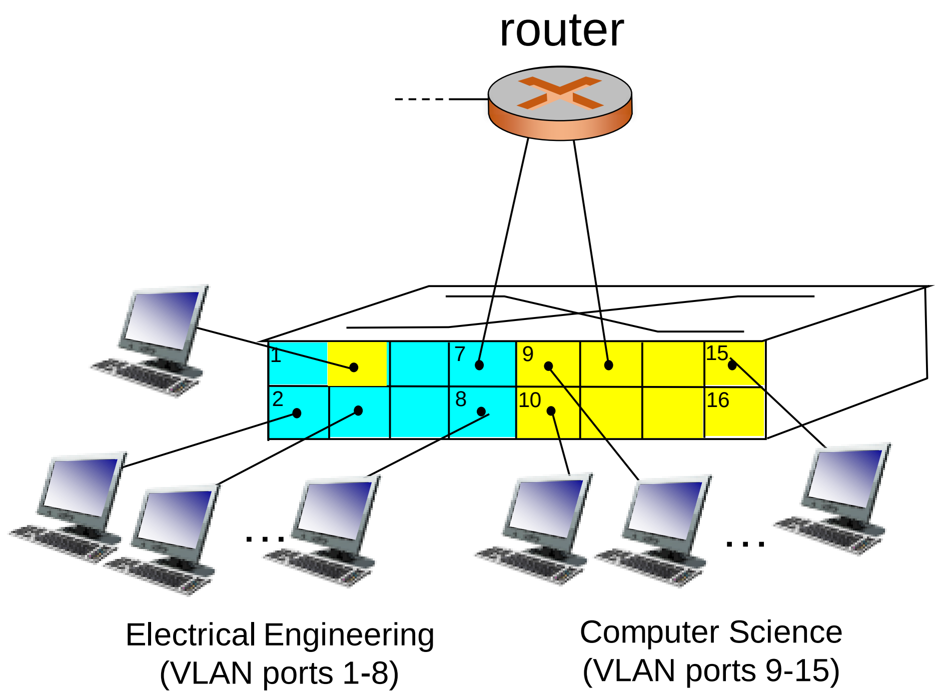


Port-based VLAN - Properties

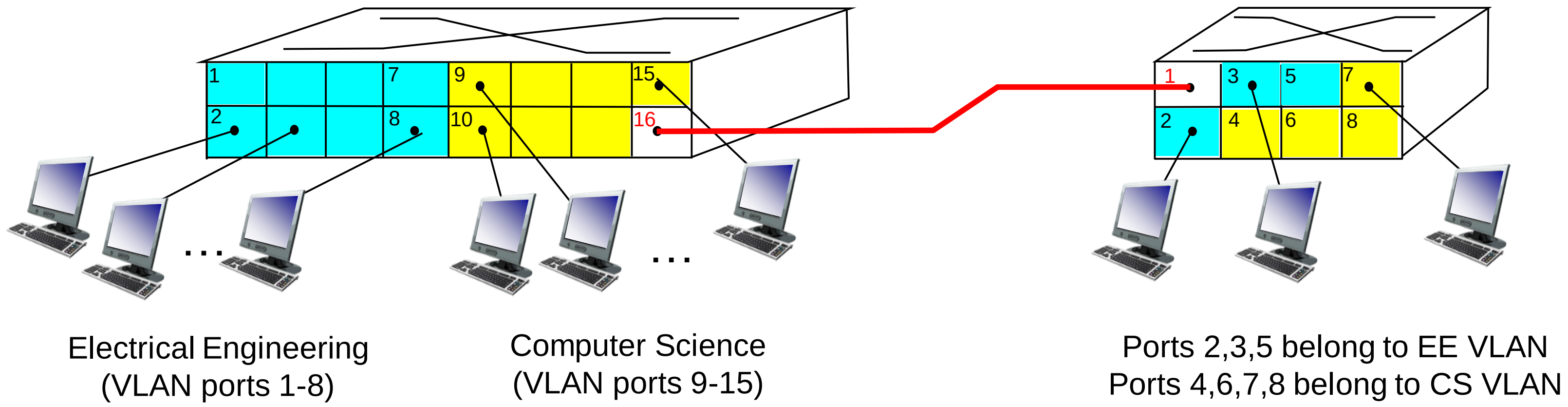
- **traffic isolation:** frames to/from ports 1-8 can only reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port

- **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 VLAN switches plus routers (so the external router shown in the figure is not needed).

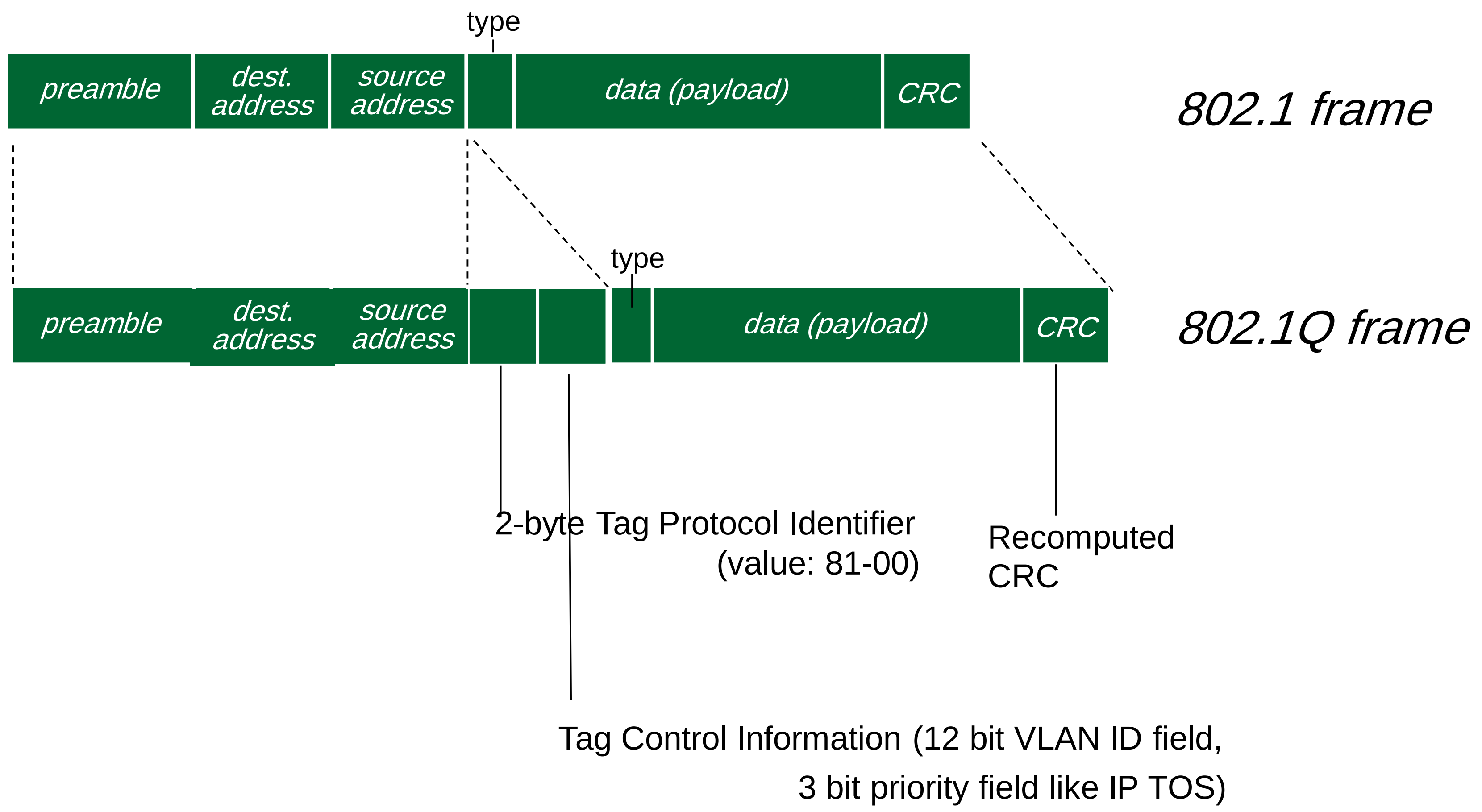


VLANS spanning multiple switches



- ***Trunk port:*** **carries frames between VLANS defined over multiple physical switches**
  - Frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - How does a switch know that a frame arriving on a trunk port belongs to a particular VLAN?
    - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

802.1Q VLAN frame format



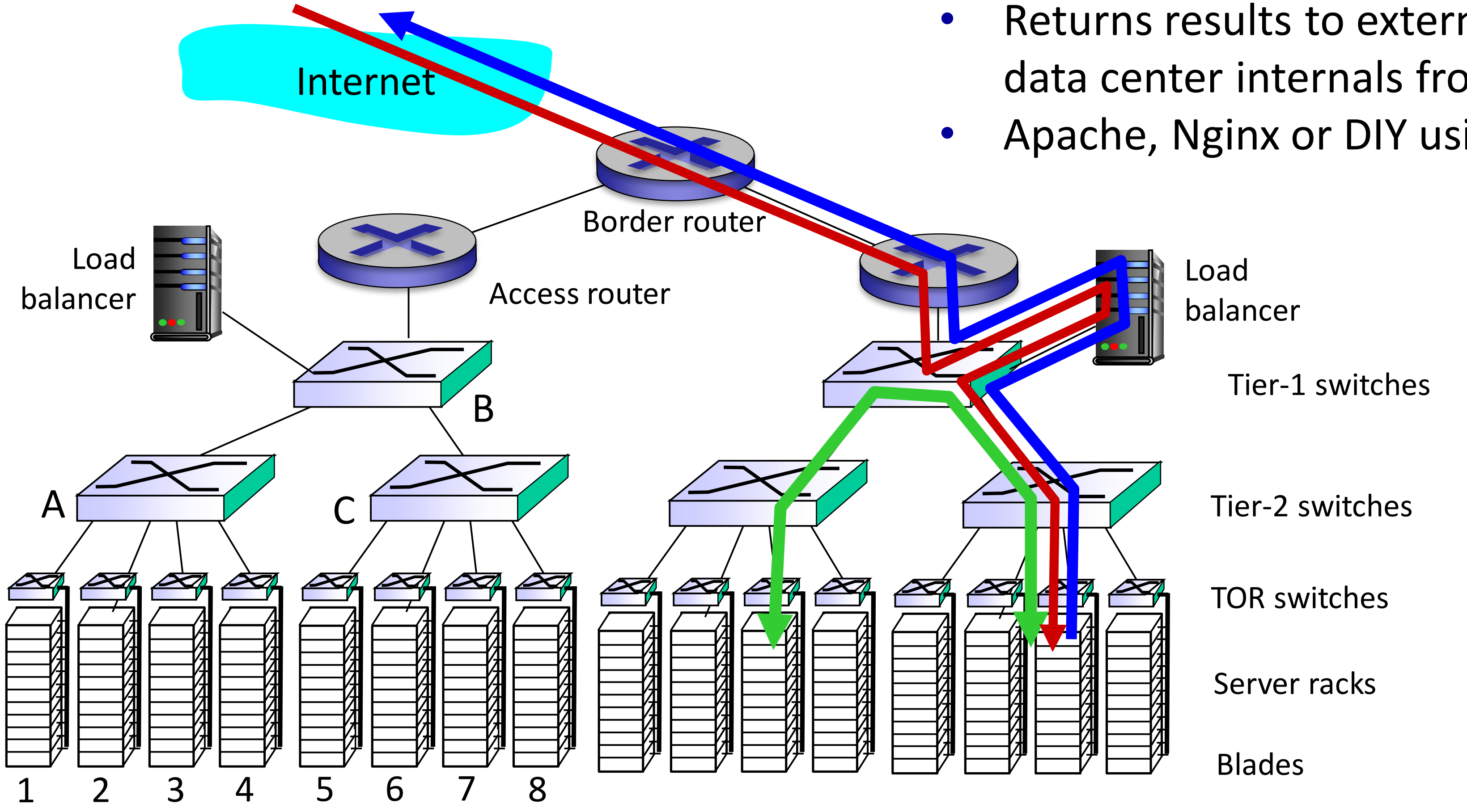
Lecture 10 - Link Layer (2)

- **Roadmap**
  1. Switches
  2. VLANs
  3. Data center networking

Data center networks

- **Internet companies house thousands of hosts, closely coupled, supporting distinct cloud applications:**
  - Search engines, data mining (google, baidu)
  - E-Business (Alibaba, Amazon)
  - SNS (Tencent, Facebook)
  - Content-servers (Youtube, Apple, Microsoft)
- **Challenges:**
  - Multiple applications, each serving massive numbers of clients
  - Managing/balancing load, avoiding processing, networking, data bottlenecks

Data center networks

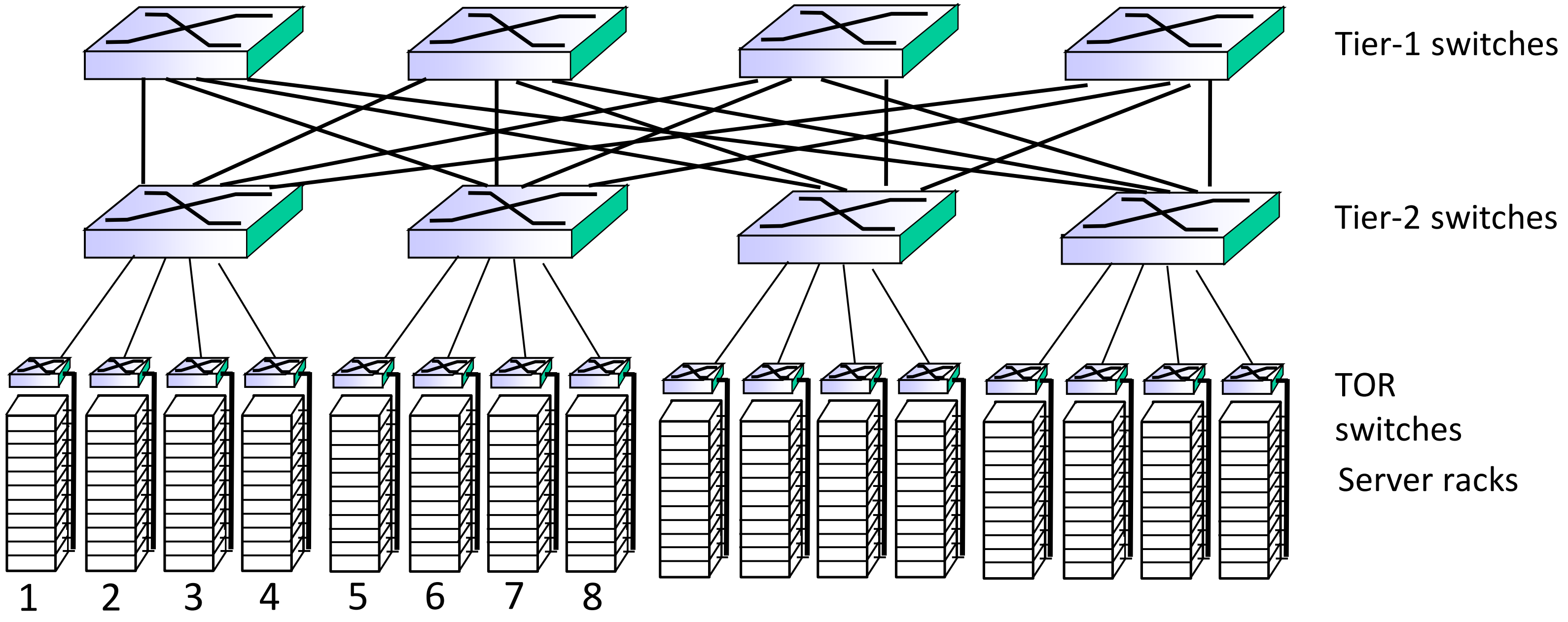


Load balancer: application-layer routing

- Receives external client requests
- Directs workload within data center
- Returns results to external client (hiding data center internals from client)
- Apache, Nginx or DIY using Python/nodejs…

Data center networks

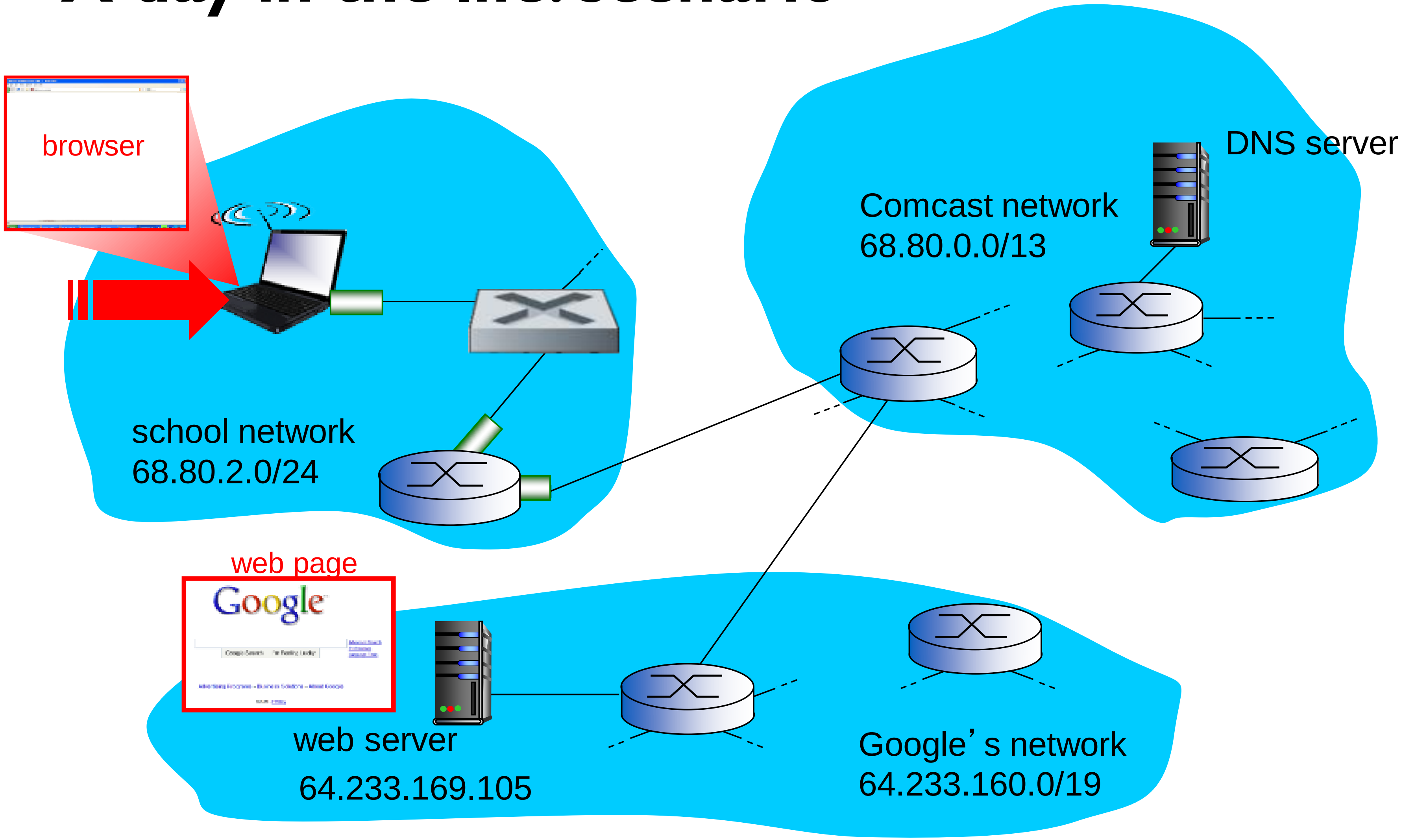
- **Rich interconnection among switches, racks:**
  - Increased throughput between racks (multiple routing paths possible)
  - Increased reliability via redundancy



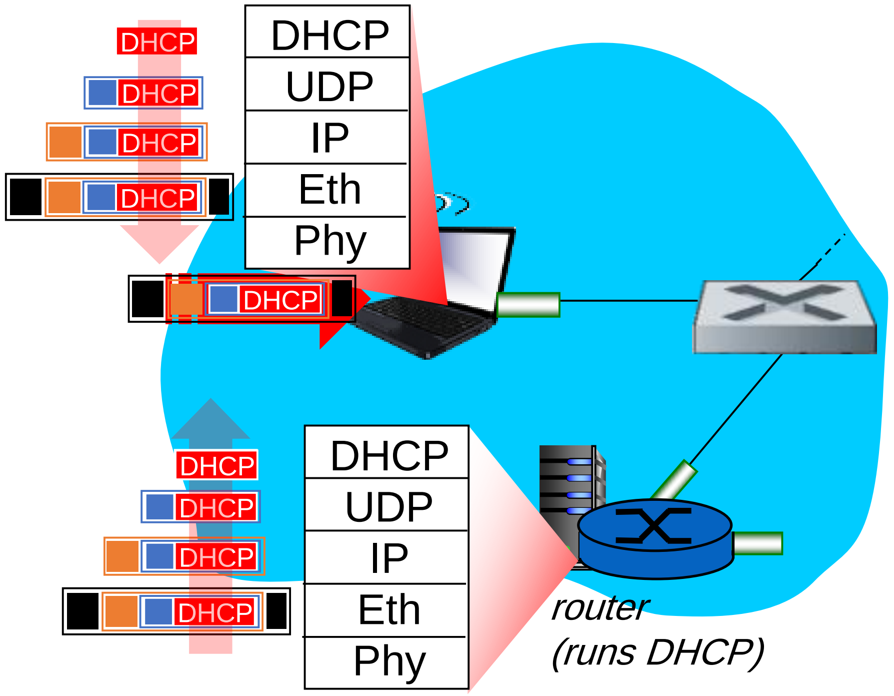
Synthesis: a day in the life of a web request

- **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

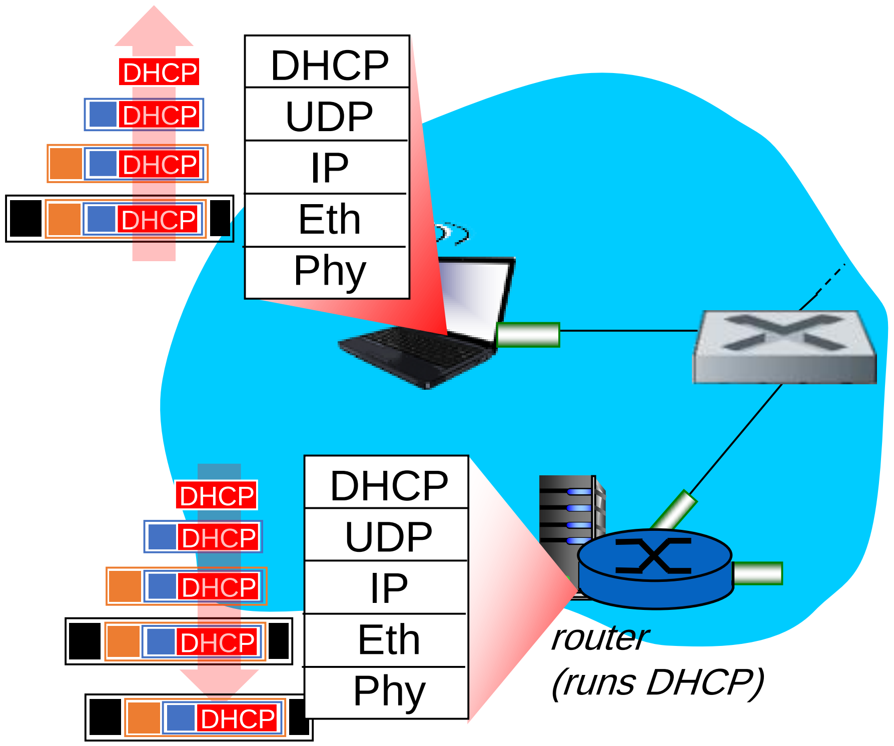


A day in the life… connecting to the Internet



- 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** (dest: broadcast IP), encapsulated in **802.3** Ethernet
- Ethernet frame **broadcast** (dest: FFFFFFFFFFFF) on LAN (switch uses self-learning for adding forwarding entry), received at router running **DHCP** server
- Ethernet **demuxed** to IP demuxed, UDP demuxed to DHCP

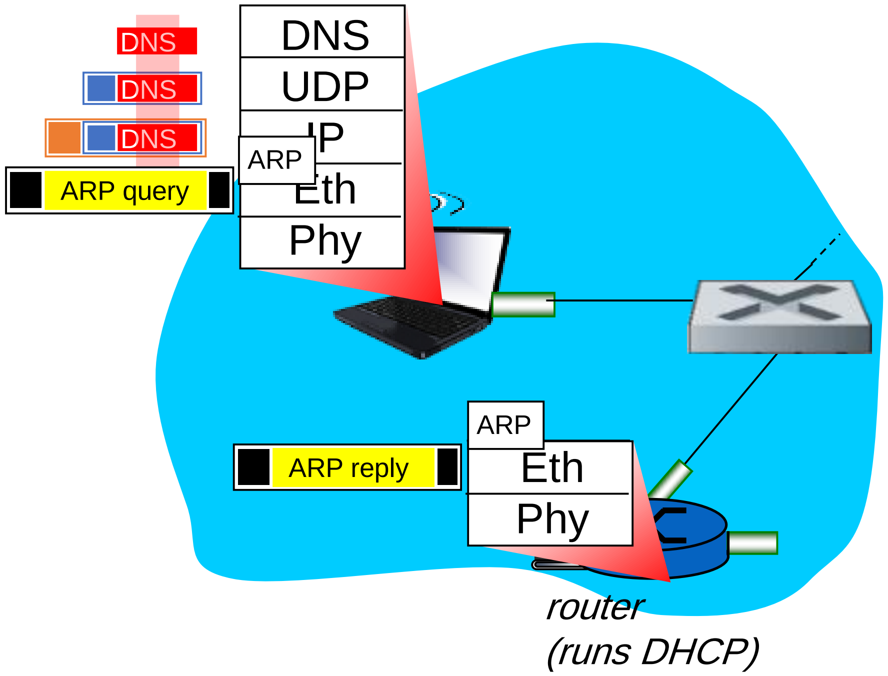
A day in the life… connecting to the Internet



- 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

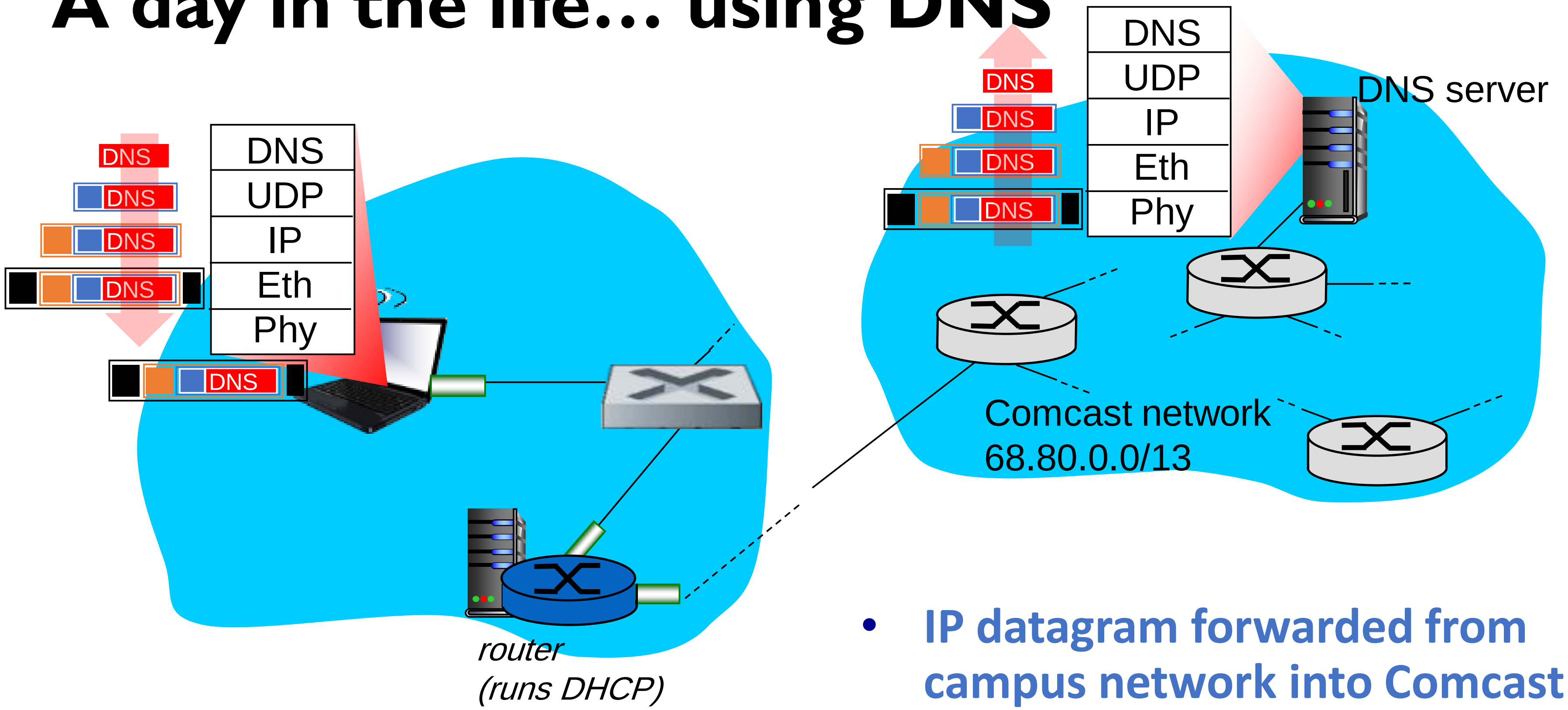
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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

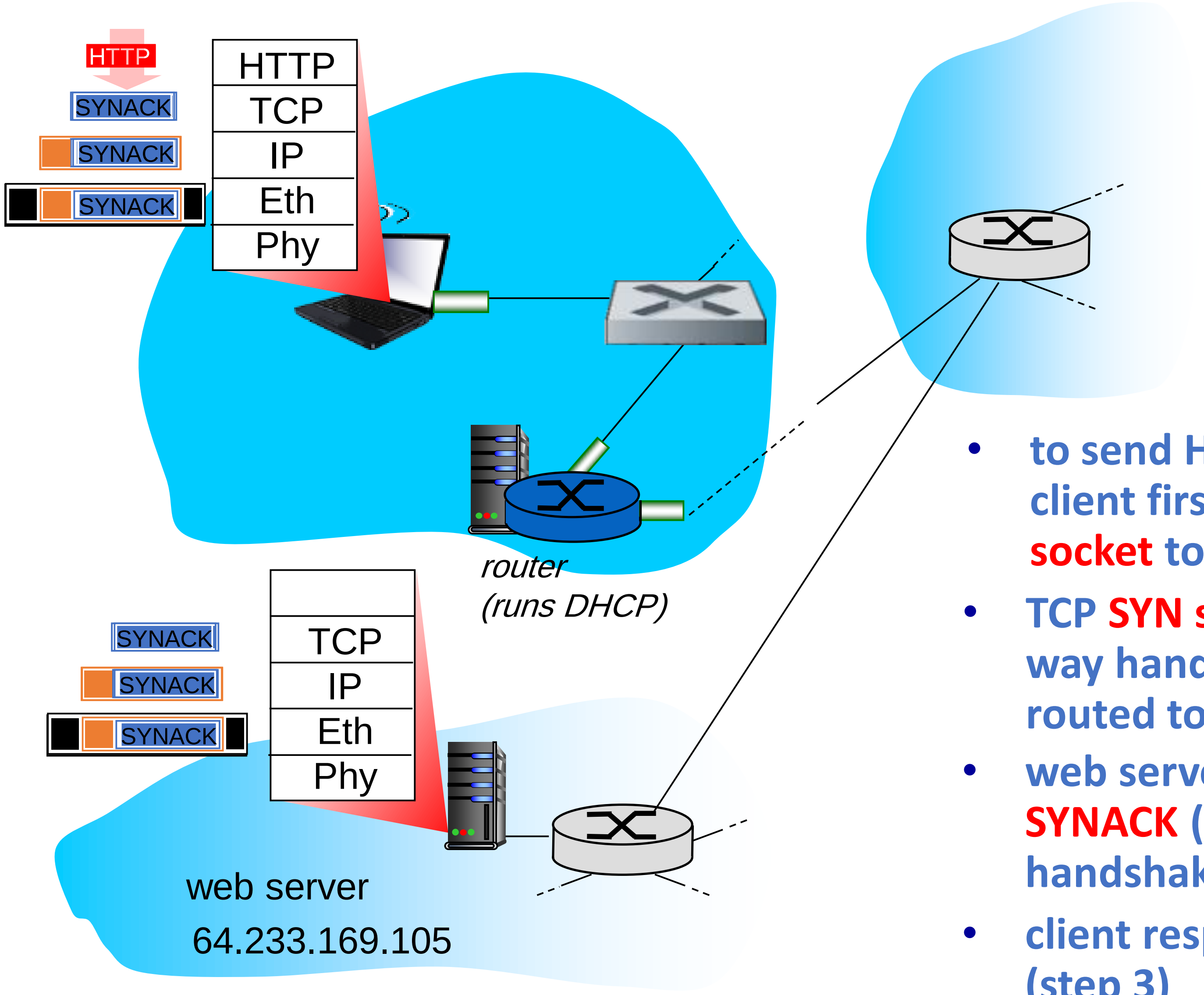
A day in the life… using DNS



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

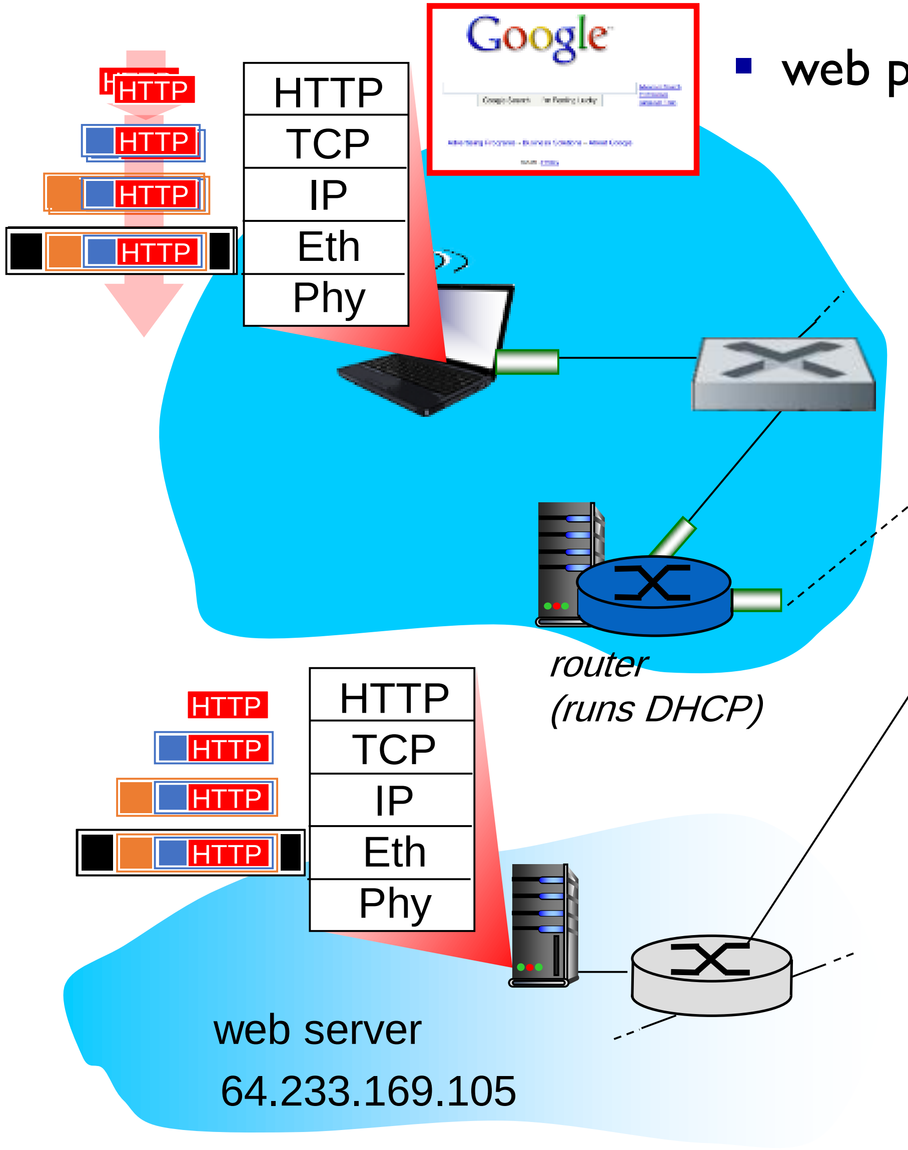
- IP datagram forwarded from campus network into Comcast network, routed (tables created by **RIP, OSPF, IS-IS** and/or **BGP** routing protocols) to DNS server
- demuxed to DNS server
- DNS server replies to client with IP address of www.google.com

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)
- client responds with **TCP ACK** (step 3)
- 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**

Lecture 10 – Network Security (1)

- **Roadmap**
  1. What is network security
  2. Principles of cryptography

What is network security?

confidentiality: only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

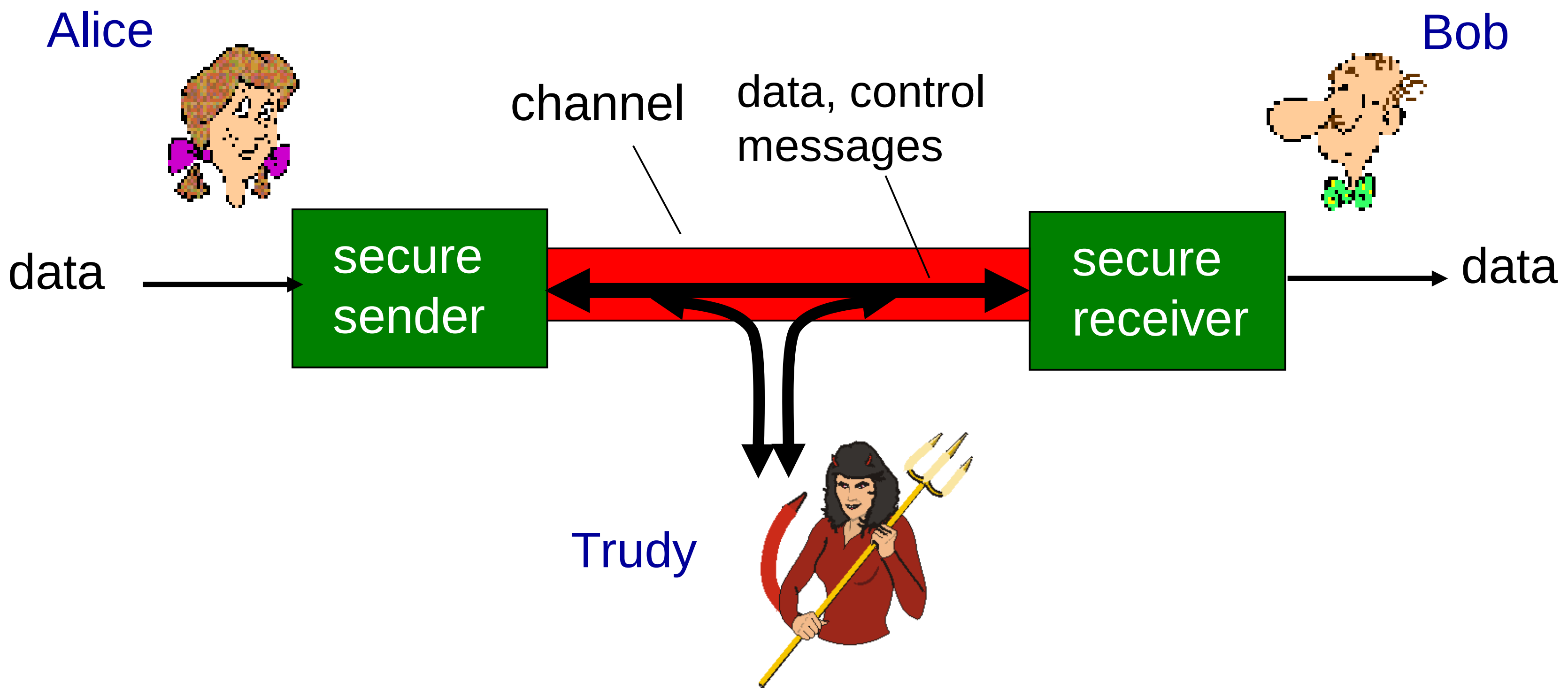
authentication: sender, receiver want to confirm identity of each other

message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

access and availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

- … well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

There are bad guys (and girls) out there!

Q: **What can a "bad guy" do?**

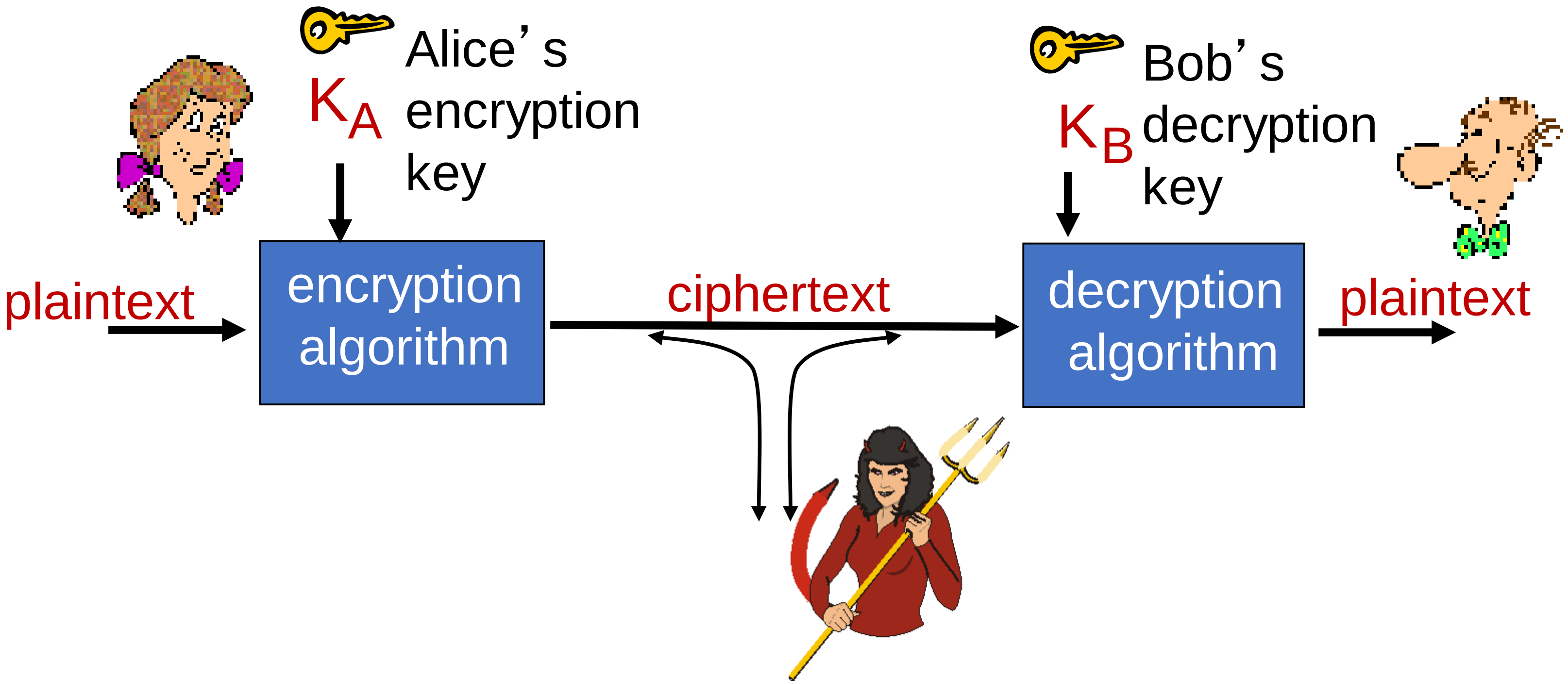
A: **A lot!**

- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

Lecture 10 – Network Security (1)

- **Roadmap**
  1. What is network security
  2. Principles of cryptography

The language of cryptography



m **plaintext message**

$K_A(m)$ **ciphertext, encrypted with key K_A**

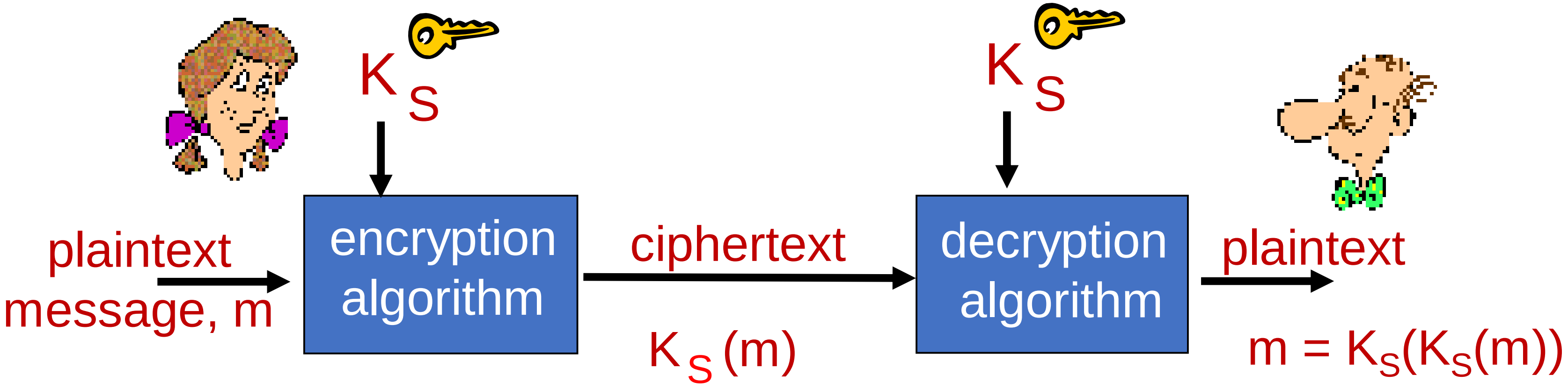
$m = K_B(K_A(m))$

Breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis

- known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can **choose** the plaintext message and obtain its corresponding ciphertext form

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

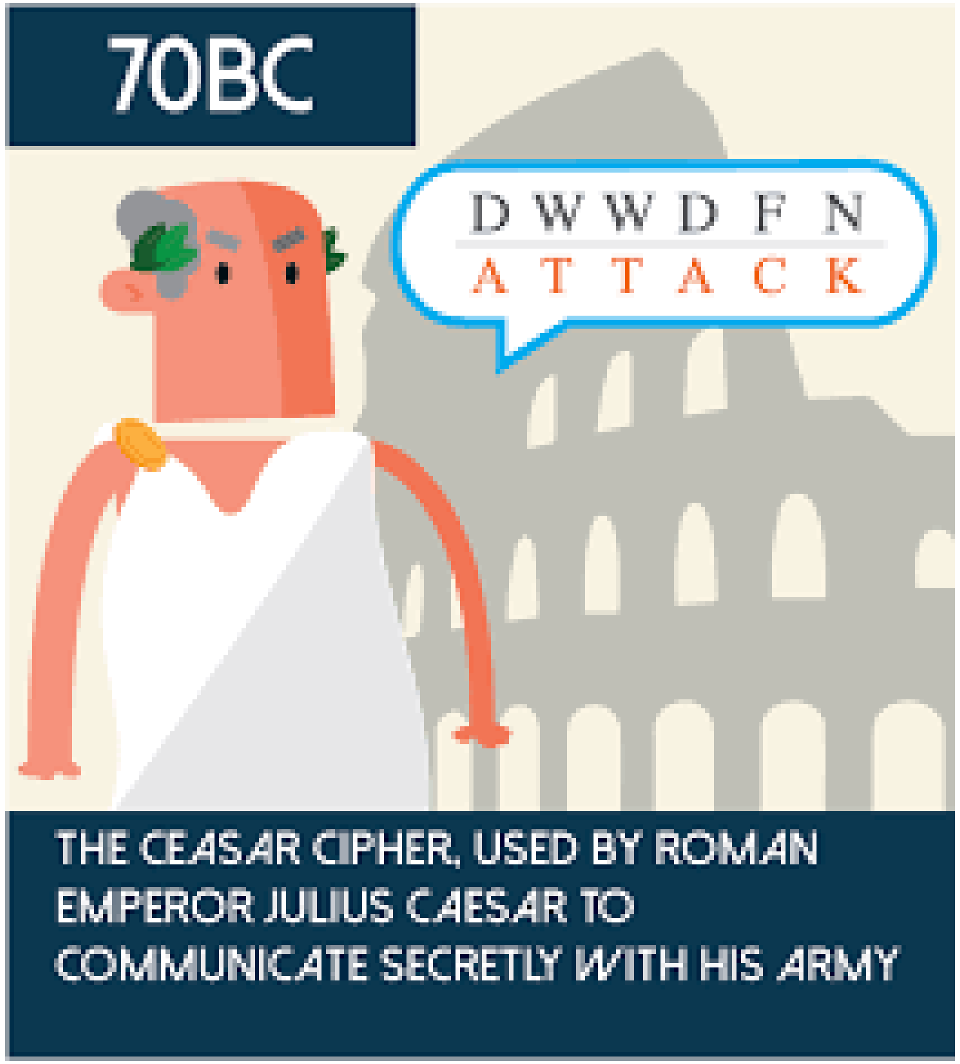
- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

<u>Q:</u> how do Bob and Alice agree on key value?

Caesar Cipher

For English text, the Caesar cipher would work by taking each letter in the plaintext message and substituting the letter that is *k* letters later (allowing wraparound; that is, having the letter *z* followed by the letter *a*) in the alphabet.

For example if k=3, then the letter *a* in plaintext becomes *d* in ciphertext; *b* in plaintext becomes *e* in ciphertext, and so on.



Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

e.g.:

```
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Encryption key: mapping from set of 26 letters to set of 26 letters

A more sophisticated encryption approach

- n substitution (**polyalphabetic**) ciphers, $M_1, M_2, \ldots, M_n$
- cycling pattern:
  - e.g., n=4: M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

Encryption key: n substitution ciphers, and cyclic pattern (i.e., key need not be just n-bit pattern)

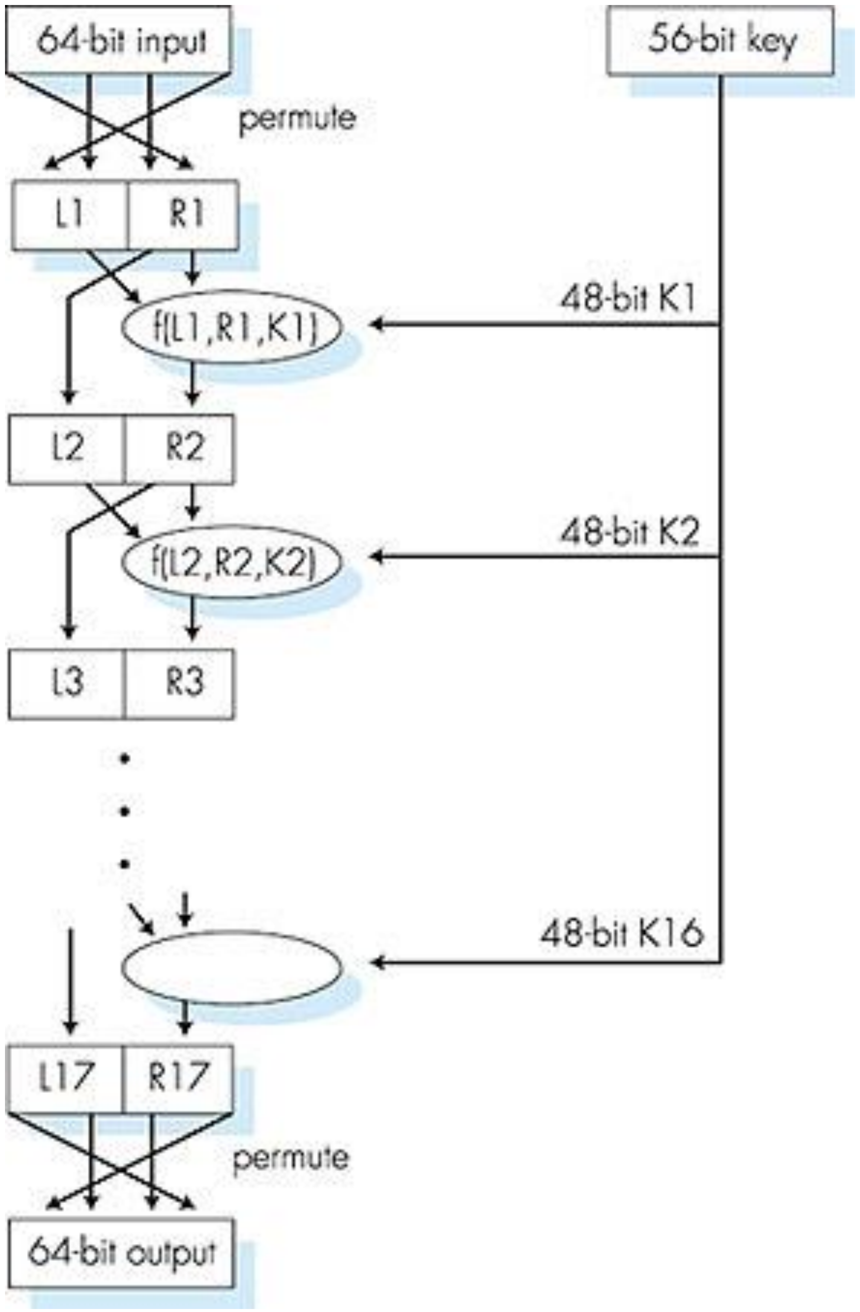
Symmetric key crypto: DES

DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

final permutation



Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- block cipher with cipher block chaining
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

AES: Advanced Encryption Standard

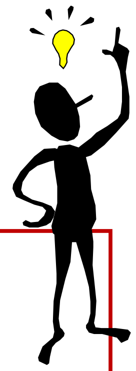
- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- a machine could brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Public Key Cryptography

symmetric key crypto

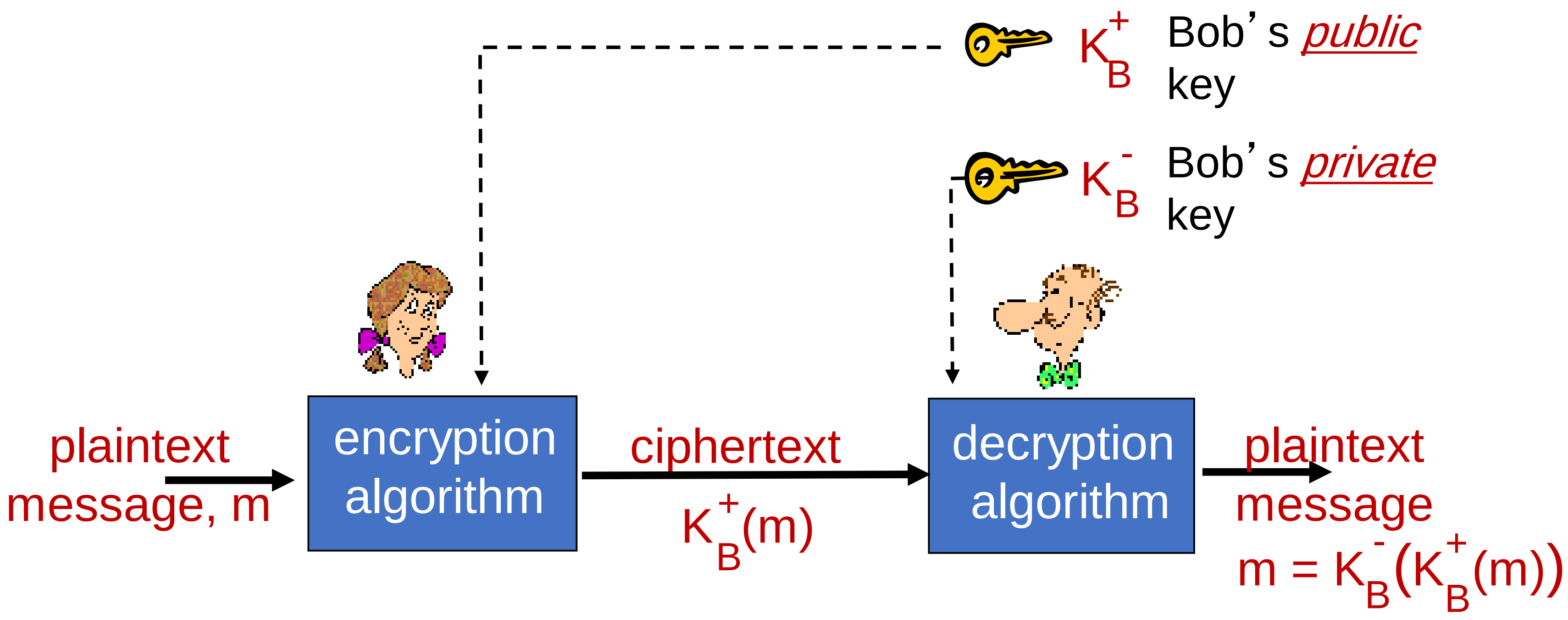
- requires sender, receiver know shared secret key
- **Q**: how to agree on key in first place (particularly if never "met")?

public key crypto



- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public key cryptography



Public key encryption algorithms

requirements:

1 need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

2 given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adleman algorithm

Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:

  [(a mod n) + (b mod n)] mod n = (a + b) mod n
  [(a mod n) - (b mod n)] mod n = (a - b) mod n
  [(a mod n) * (b mod n)] mod n = (a * b) mod n

- thus

  $(a \bmod n)^d \bmod n = a^d \bmod n$

- example: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number

example:

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

1. choose two large prime numbers *p, q.* (e.g., 1024 bits each)

2. compute $n = pq$, $z = (p-1)(q-1)$

3. choose e (with $e<n$) that has no common factors with z (e, z are "relatively prime").

4. choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).

5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above

1. to encrypt message m $(<n)$, compute
$$c = m^e \bmod n$$

2. to decrypt received bit pattern, c, compute
$$m = c^d \bmod n$$

magic happens! $$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$$

RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

e=5 (so *e, z* relatively prime).

d=29 (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.

encrypt: $\underbrace{\text{bit pattern}}_{00001000}$ $\quad \underbrace{m}_{12}$ $\quad \underbrace{m^e}_{24832}$ $\quad \underbrace{c = m^e \bmod n}_{17}$

decrypt: $\underbrace{c}_{17}$ $\quad \underbrace{c^d}_{481968572106750915091411825223071697}$ $\quad \underbrace{m = c^d \bmod n}_{12}$

Why does RSA work?

- must show that $c^d \bmod n = m$
  where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where $n = pq$ and $z = (p-1)(q-1)$
- thus,
  $c^d \bmod n = (m^e \bmod n)^d \bmod n$
  $= m^{ed} \bmod n$
  $= m^{(ed \bmod z)} \bmod n$
  $= m^1 \bmod n$
  $= m$

RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\substack{\text{use public key first,}\\ \text{followed by}\\ \text{private key}}} = m = \underbrace{K_B^+(K_B^-(m))}_{\substack{\text{use private key}\\ \text{first, followed by}\\ \text{public key}}}$$

result is the same!

Why $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$?

follows directly from modular arithmetic:

$$
\begin{aligned}
(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\
&= m^{de} \bmod n \\
&= (m^d \bmod n)^e \bmod n
\end{aligned}
$$

Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_S

- Bob and Alice use RSA to exchange a symmetric key K_S
- once both have K_S, they use symmetric key cryptography

Thanks.

- **Addresses**
  - Email: Wenjun.Fan@xjtlu.edu.cn
  - Office: EE 214
- **Office hours**
  - Monday: 12:00 – 13:00
  - Tuesday: 12:00 – 13:00