# CAN201: Introduction to Networking

## Lecture 10 - The Link Layer 2 & Network Security 1

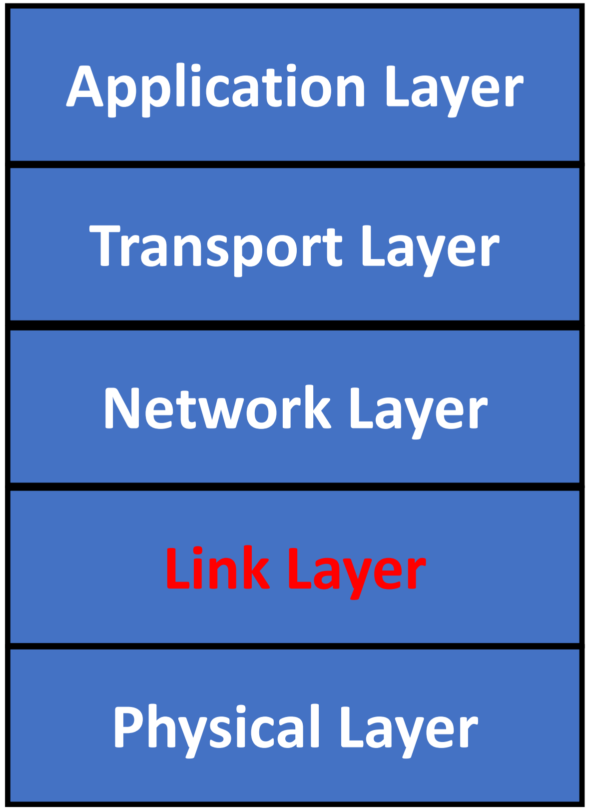
Lecturer: Dr. Gordon Boateng

# Important Information

- **Contact:**
  - Email: Gordon.Boateng@xjtlu.edu.cn
  - Office No.: SC554A

- **Office Hours (Strictly via appointment)**
  - Tuesday: 14:00-15:00
  - Wednesday: 14:00-15:00

# Recap: Top-Down Approach



# Link layer, LANs: roadmap

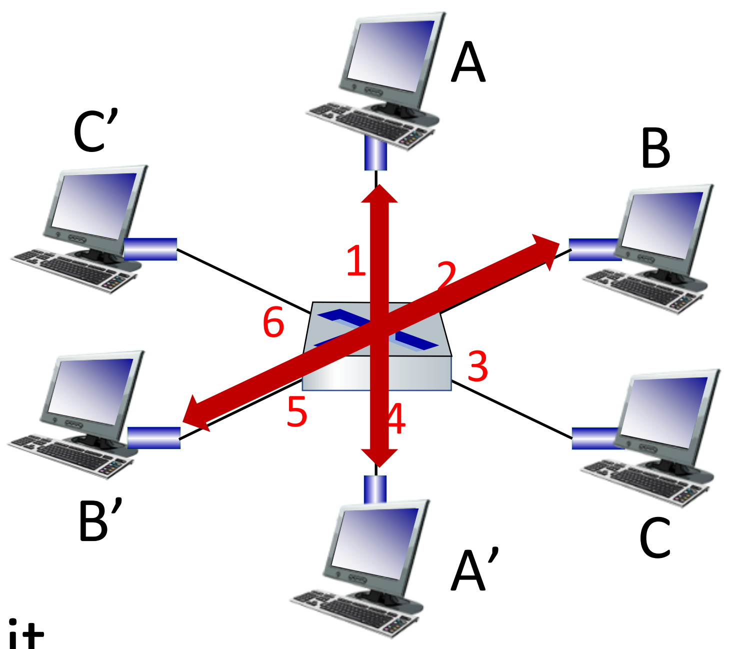
- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- data center networking

# Ethernet switch

- Switch is a 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 *unaware* of presence of switches
- plug-and-play, self-learning
  - switches do not need to be configured

# Switch: multiple simultaneous transmissions

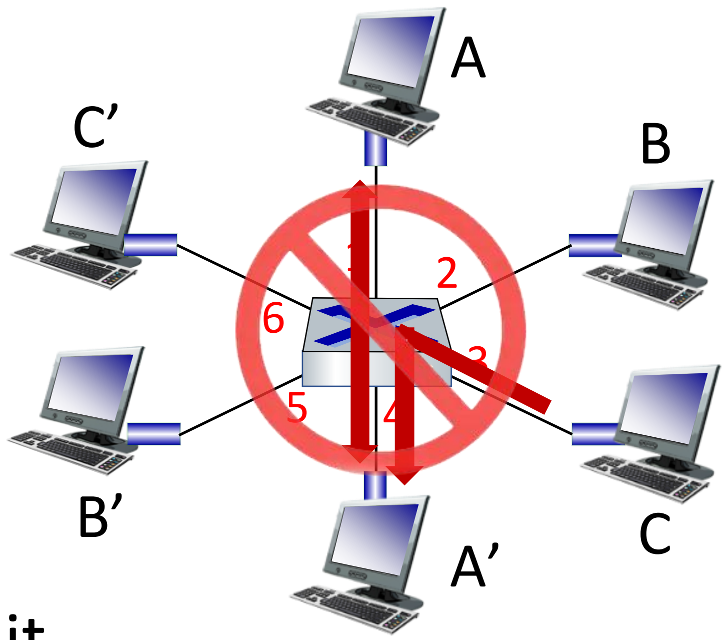
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A' and B-to-B' can transmit simultaneously, without collisions




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

# Switch: multiple simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
  - no collisions; full duplex
  - each link is its own collision domain
- switching: A-to-A' and B-to-B' can transmit simultaneously, without collisions
  - but A-to-A' and C to A' can *not* happen simultaneously



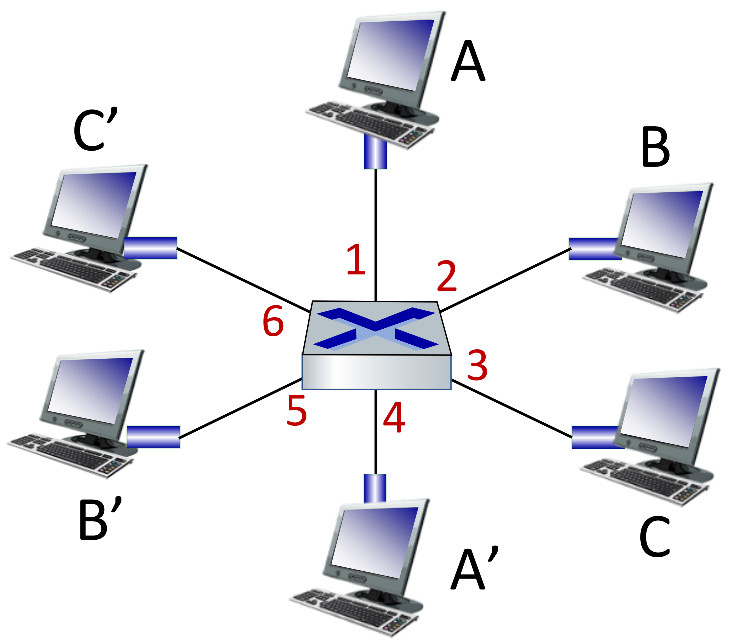

switch with six interfaces (1,2,3,4,5,6)

# 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!

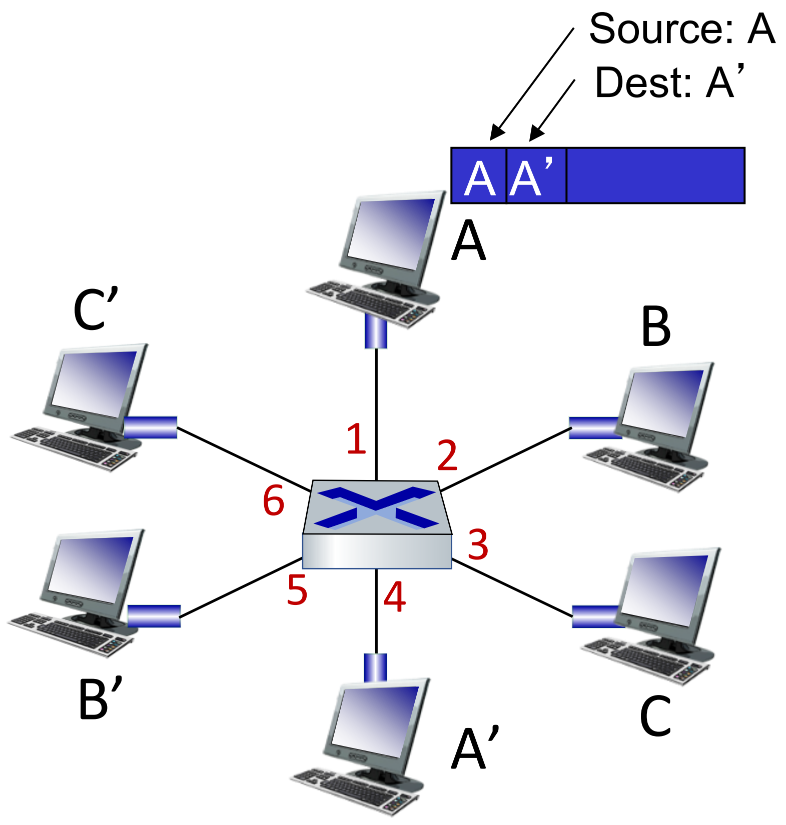



*Q:* how are entries created, maintained in switch table?

- something like a routing protocol?

# Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table

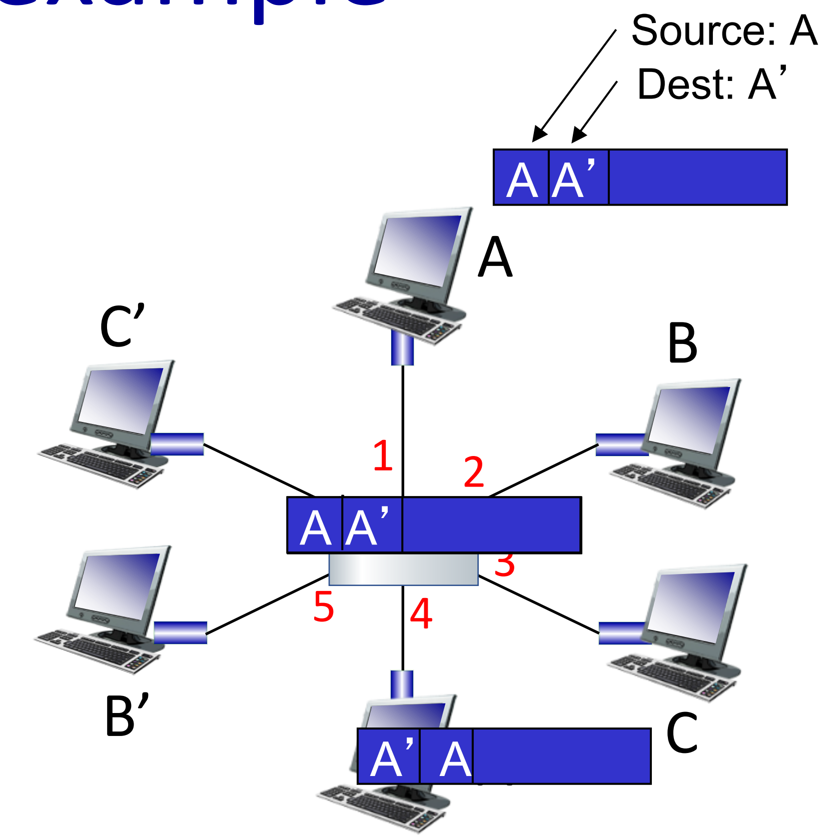



| MAC addr | interface | TTL |
|---|---|---|
| *A* | *1* | *60* |
| | | |

*Switch table (initially empty)*

# Self-learning, forwarding: example

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




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

*switch table (initially empty)*

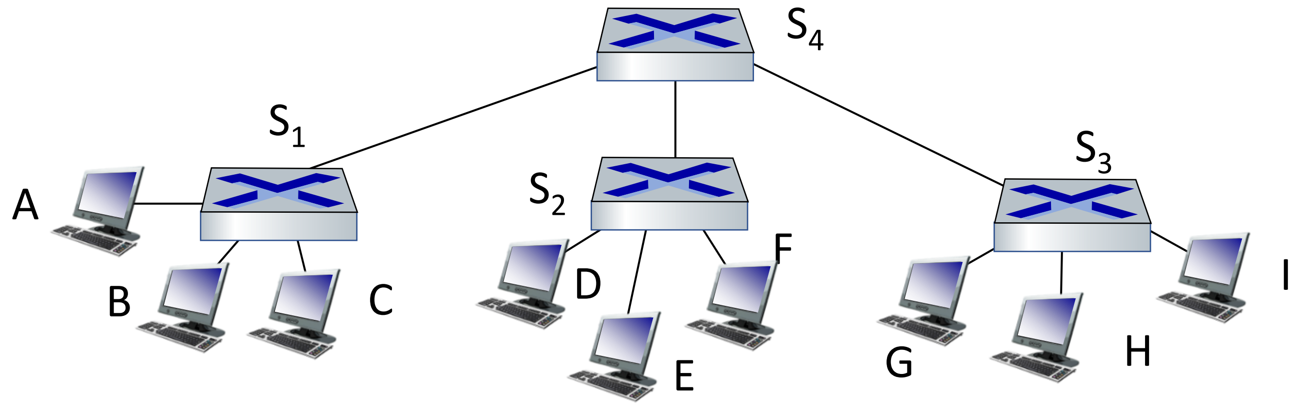
# Switch: frame filtering/forwarding

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 */

# Interconnecting switches

self-learning switches can be connected together:

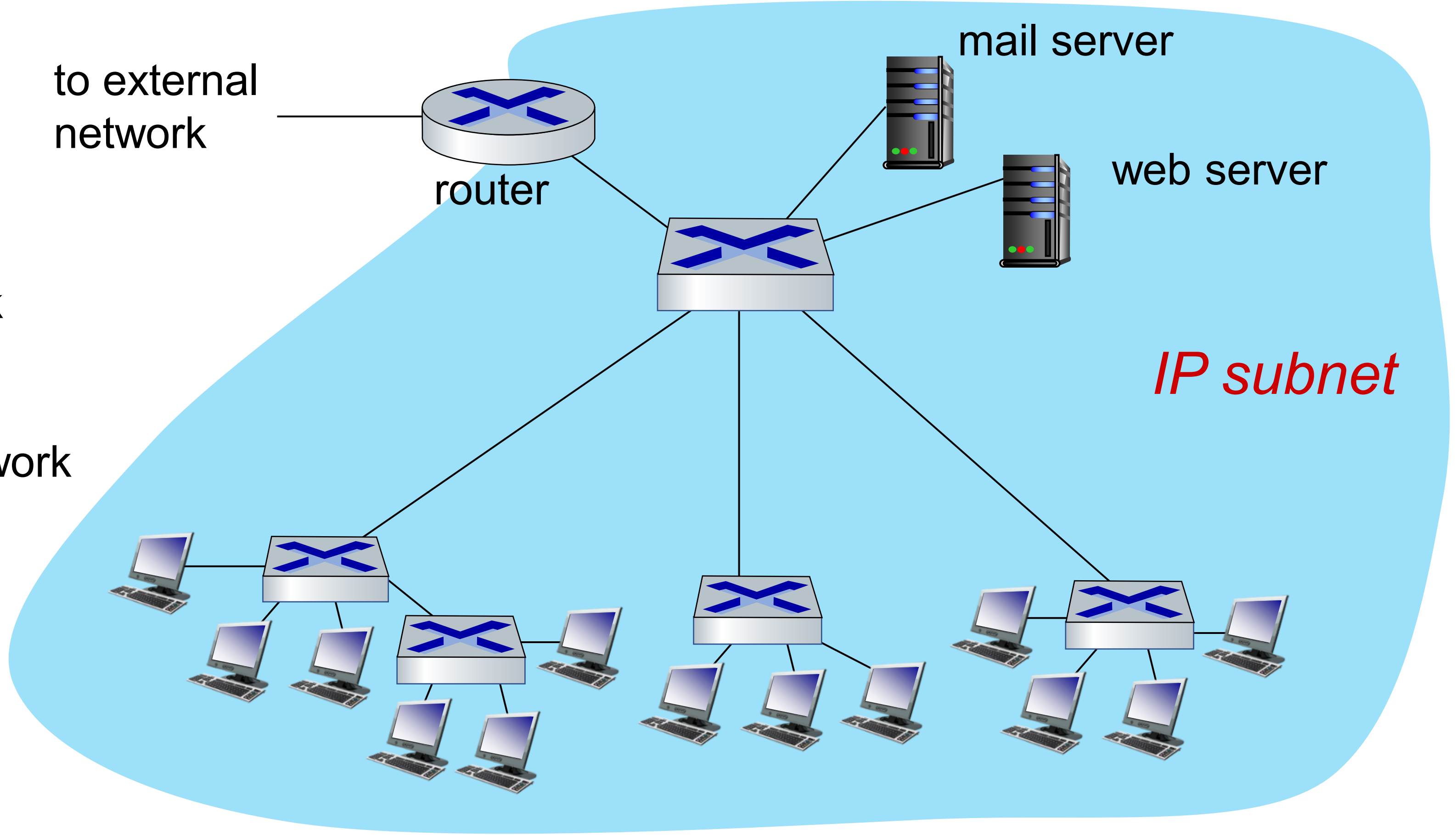



*Q:* sending from A to G - how does $S_1$ know to forward frame destined to G via $S_4$ and $S_3$?

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

**Problem:** Broadcast storms may occur

# Small institutional network




Switch for local network forwarding,

Router for external network forwarding

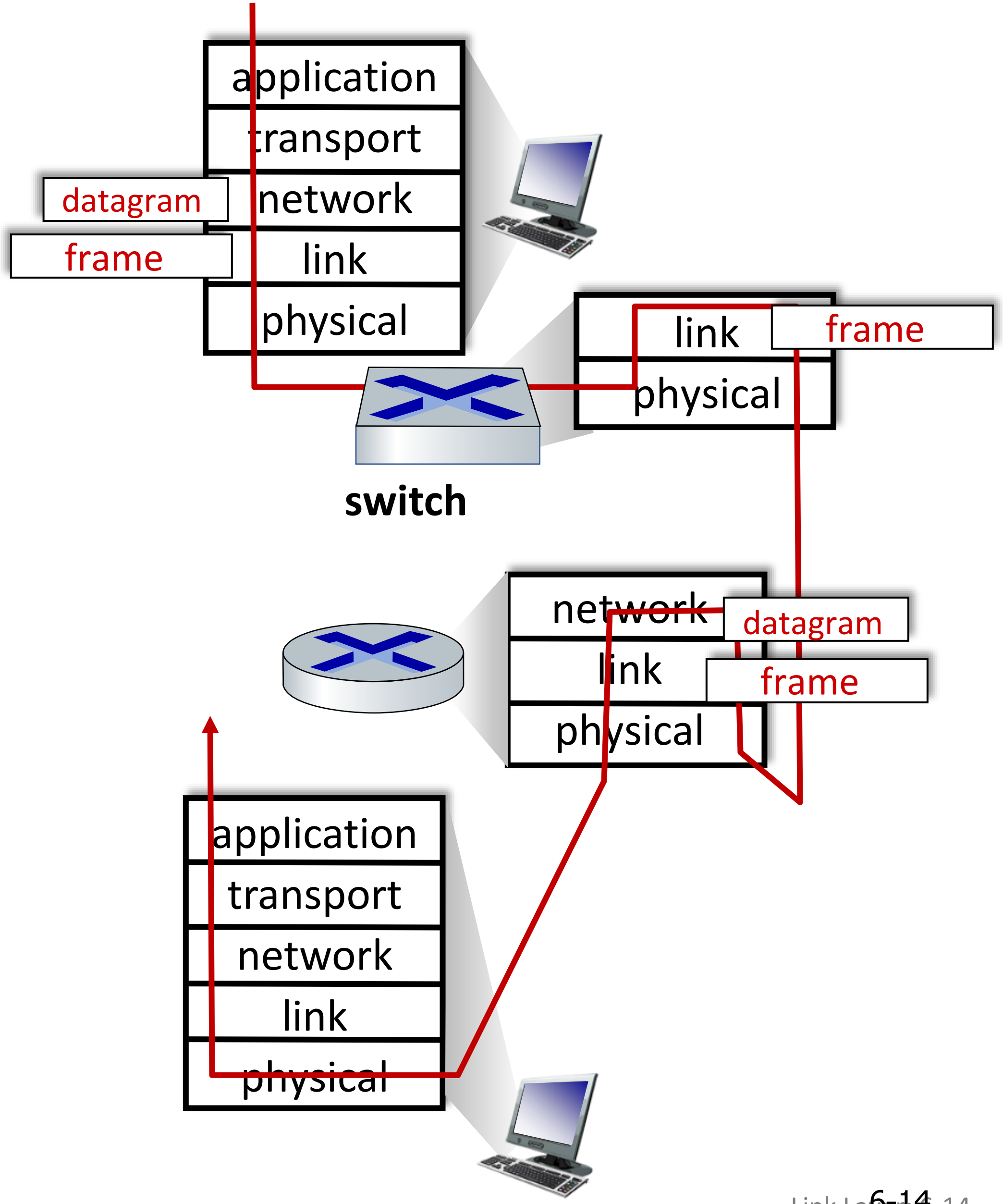
# Switches vs. routers

both are store-and-forward:

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

both have forwarding tables:

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

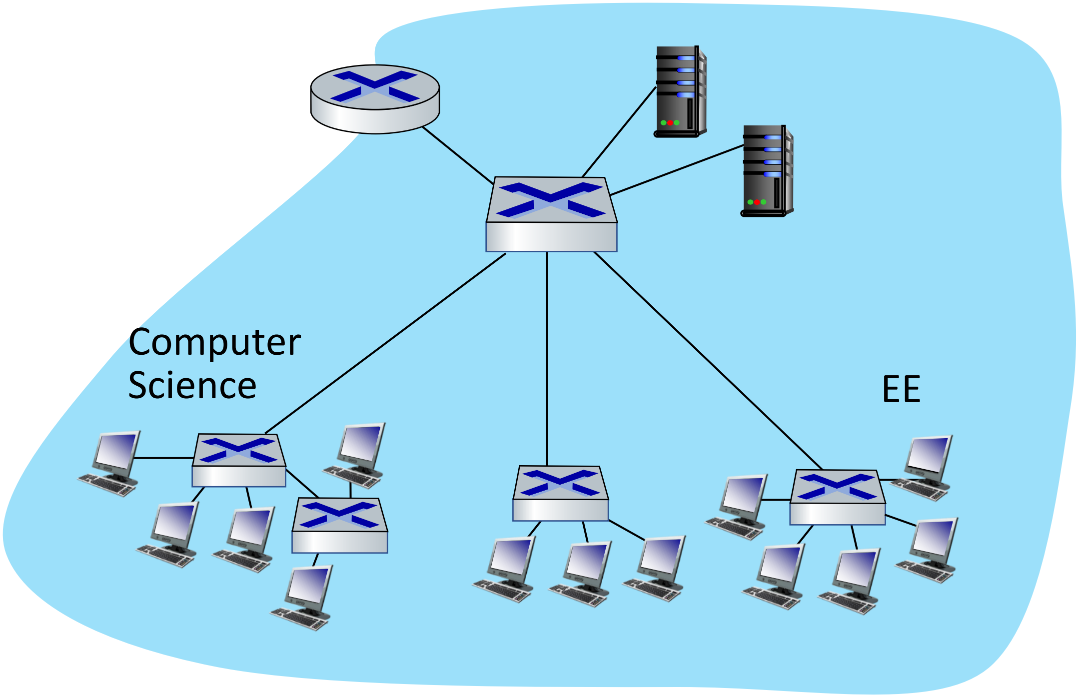


# Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- data center networking

# Virtual LANs (VLANs): motivation

*Q:* what happens as LAN sizes scale, users change point of attachment?

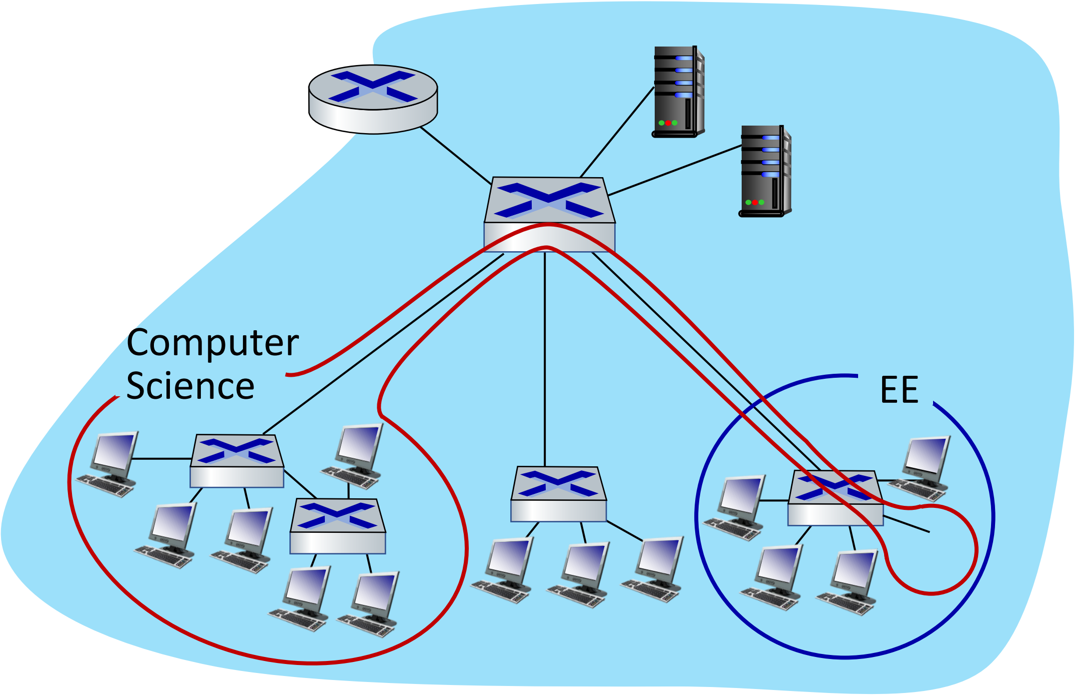



single broadcast domain:

- *scaling:* all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- efficiency, security, privacy issues

# Virtual LANs (VLANs): motivation

*Q:* what happens as LAN sizes scale, users change point of attachment?




## single broadcast domain:

- *scaling:* all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- security, privacy, efficiency issues

## administrative issues:

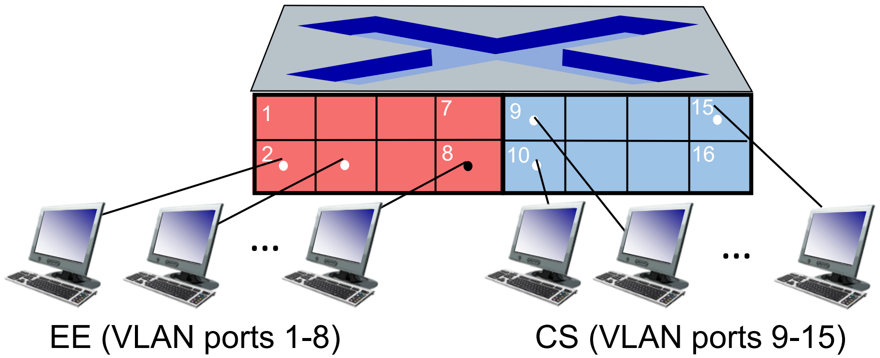
- CS user moves office to EE - *physically* attached to EE switch, but wants to remain *logically* attached to CS switch

# Port-based VLANs

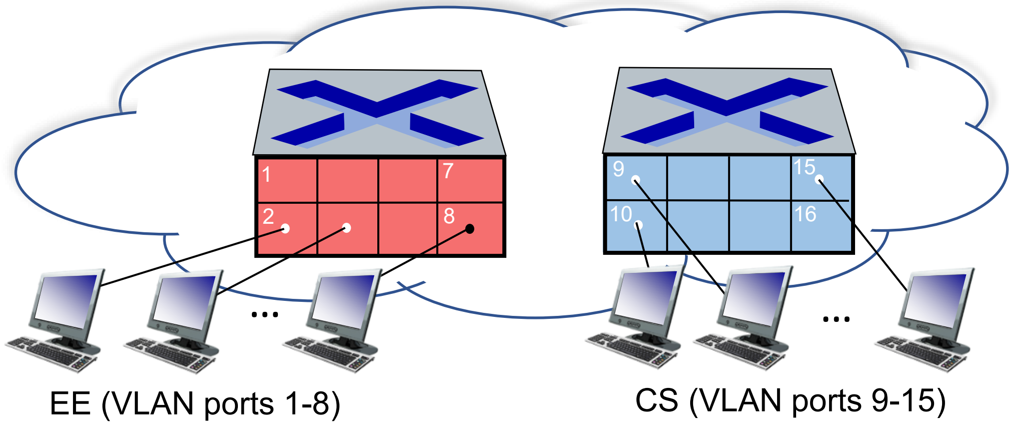
## Virtual Local Area Network (VLAN)

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 ……

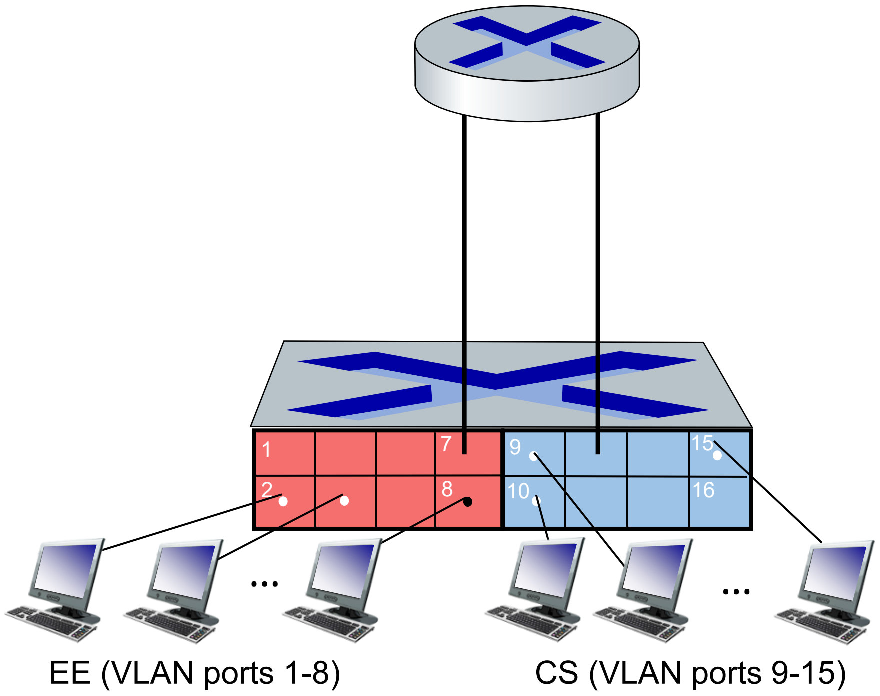



… operates as multiple virtual switches

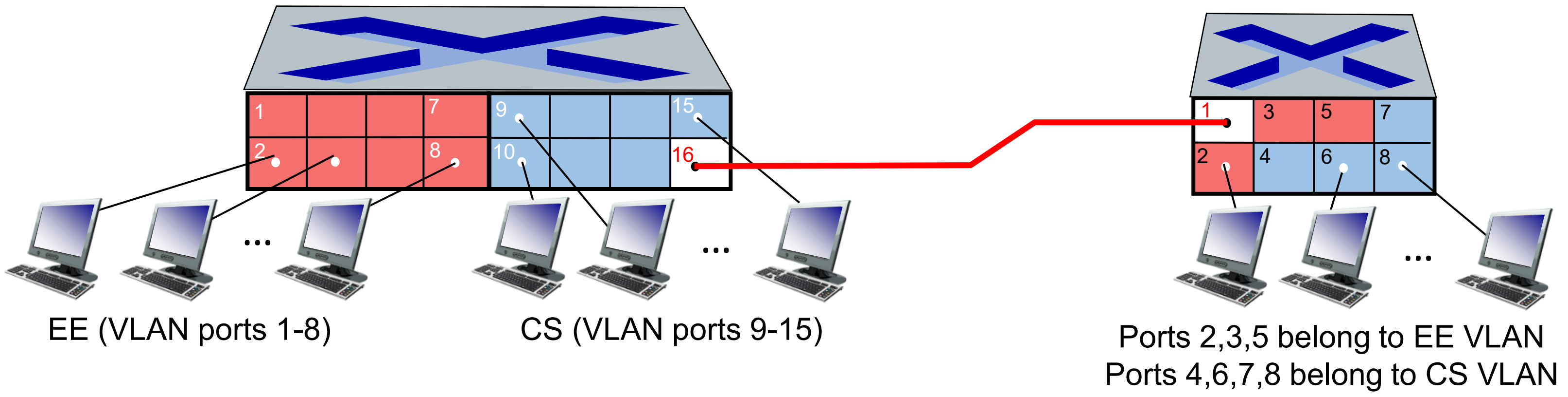


# Port-based VLANs

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



# VLANs spanning multiple switches




trunk port: carries frames between VLANS defined over multiple physical switches

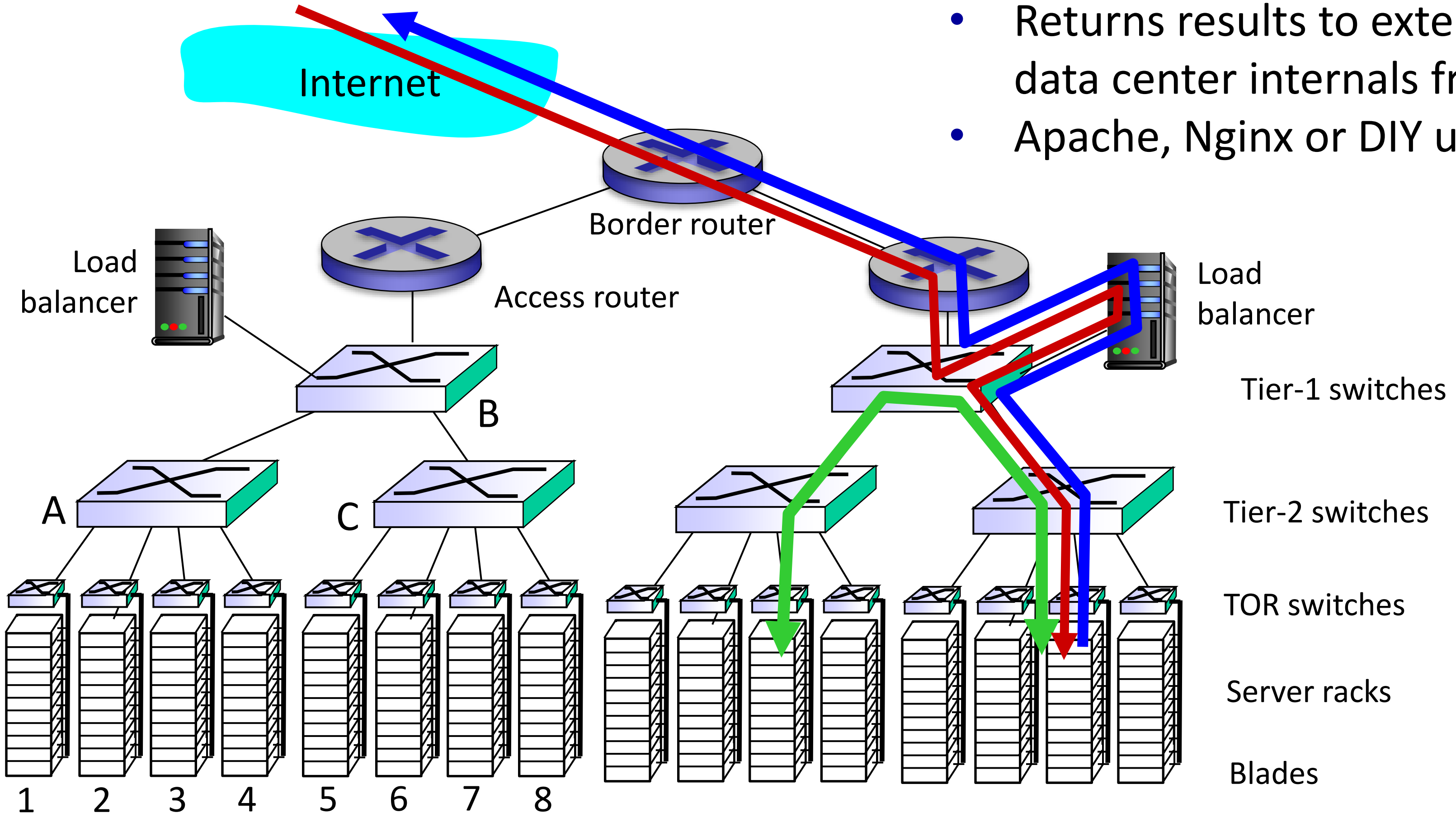
# Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- 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

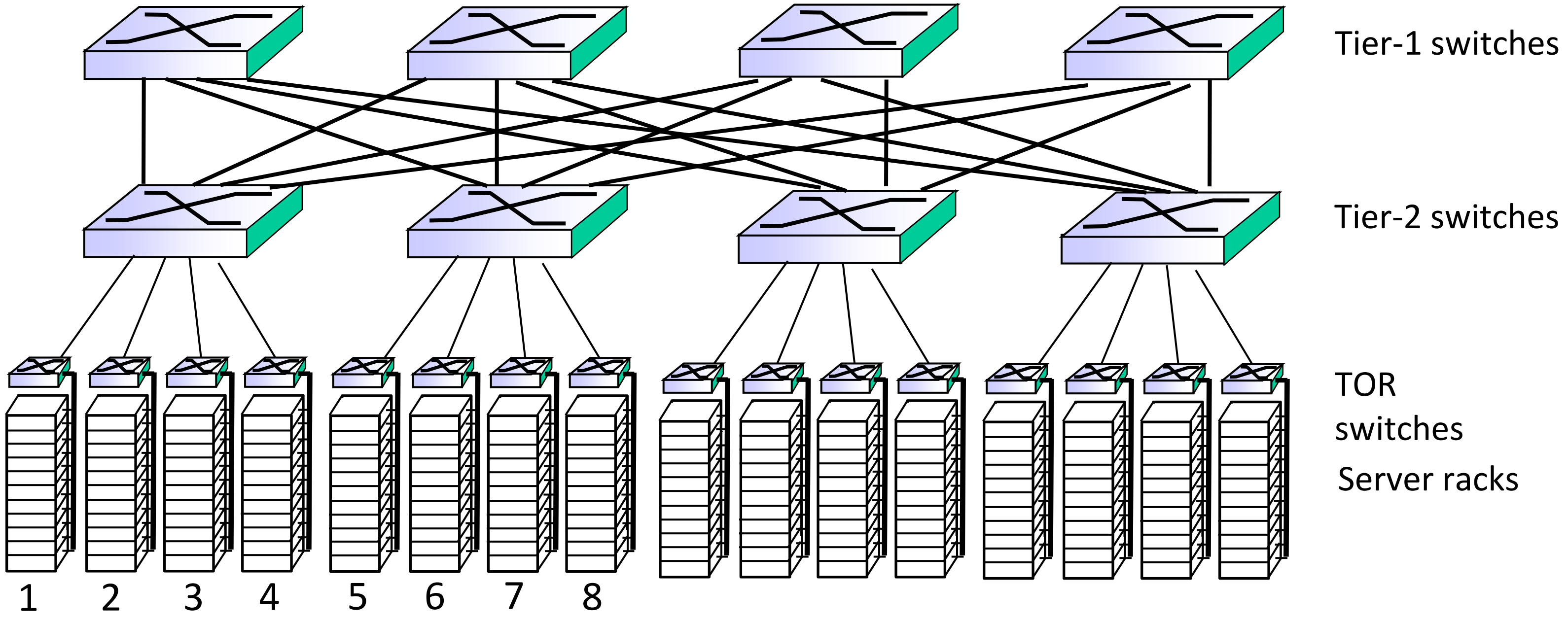



## Load balancer: application-layer routing

- 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:* putting-it-all-together

- **journey down protocol stack complete!**
  - application, transport, network, link

***Scenario:***

A student attaches a laptop to the campus network, requests/receives www.google.com

# Network Security 1

- What is network security
- Principles of cryptography

# What is network security?

***Confidentiality*:** only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

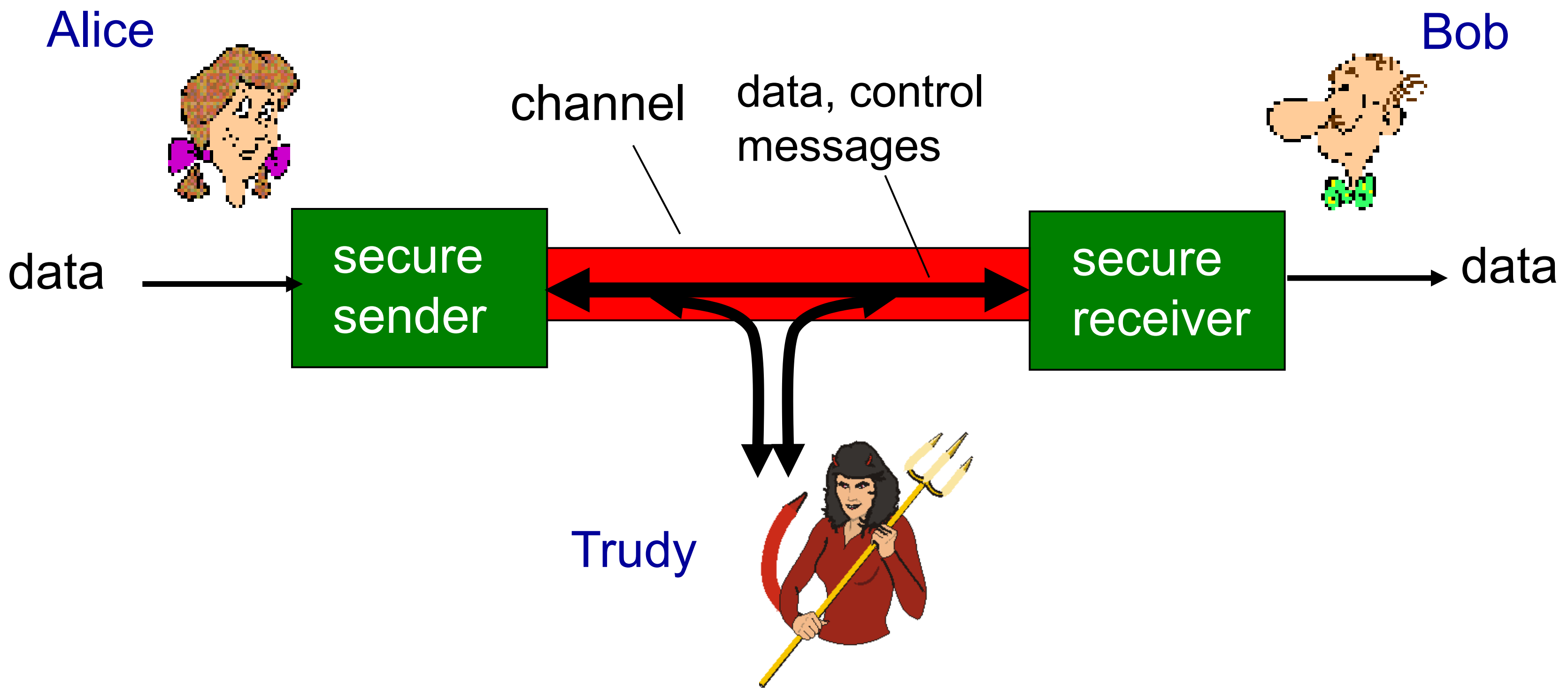
***Integrity:*** sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

***Authentication:*** sender, receiver want to confirm identity of each other

***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 want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



# Who might Bob, Alice be?

- 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 out there!

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

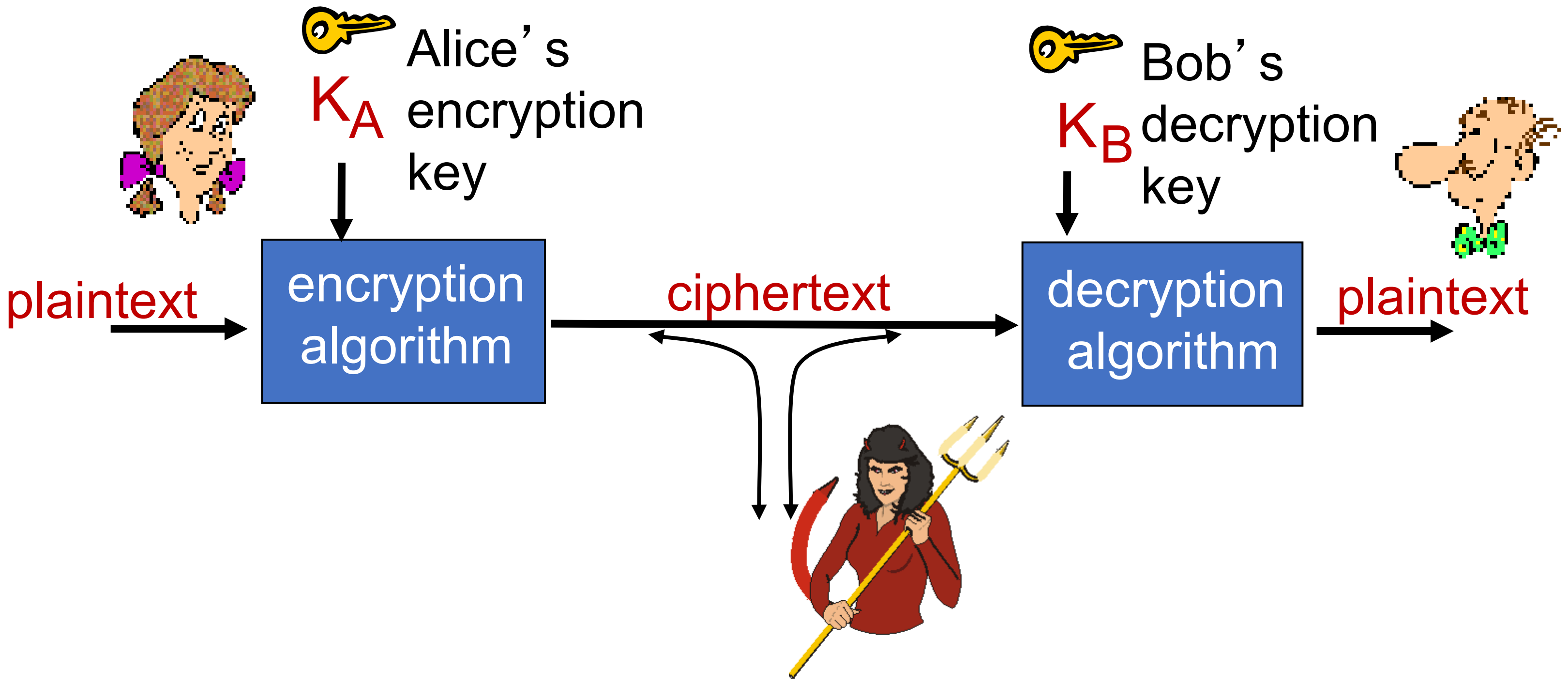
***A:*** **A lot!**

- *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)

# Network Security 1

- What is network security
- Principles of cryptography

# The language of cryptography




**m plaintext message**

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

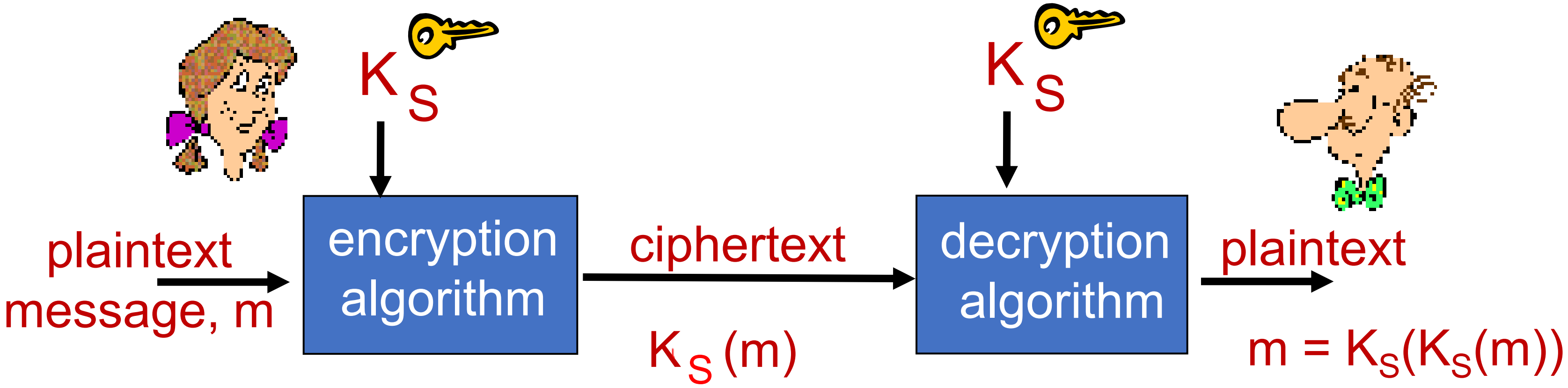
**m = $K_B$($K_A$(m))**

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

# Symmetric key cryptography




Symmetric key crypto: Bob and Alice share same (symmetric) key: K

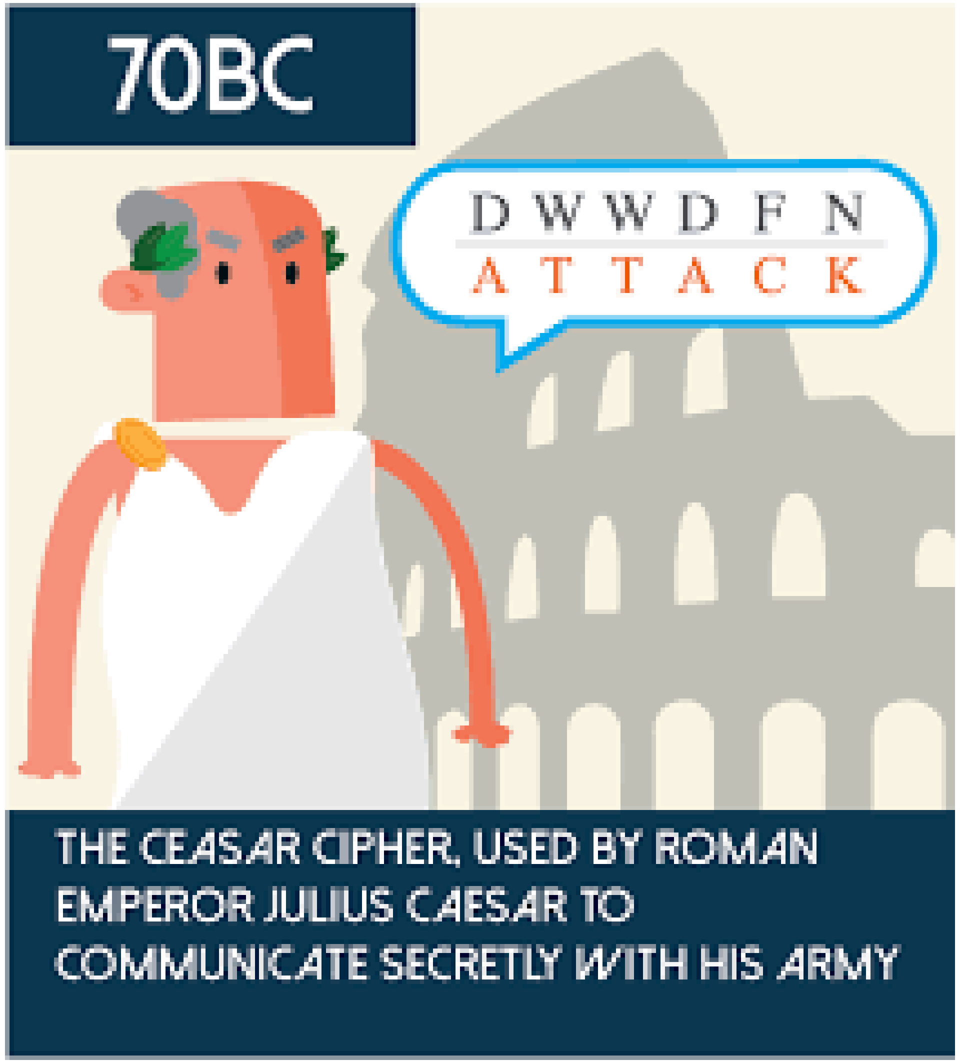
- e.g., key is known substitution pattern in mono alphabetic substitution cipher
- Fast, bur requires secure key sharing

*Q:* how do Bob and Alice agree on key value? (Key distribution problem)

# 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: 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
  - Weak by modern standards
- **making DES more secure:**
  - 3DES: encrypt 3 times with 3 different keys

# AES: Advanced Encryption Standard

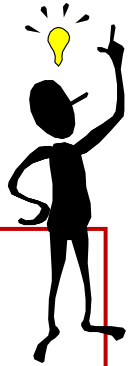
- symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- a machine could brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

# Public Key Cryptography

*symmetric key crypto*

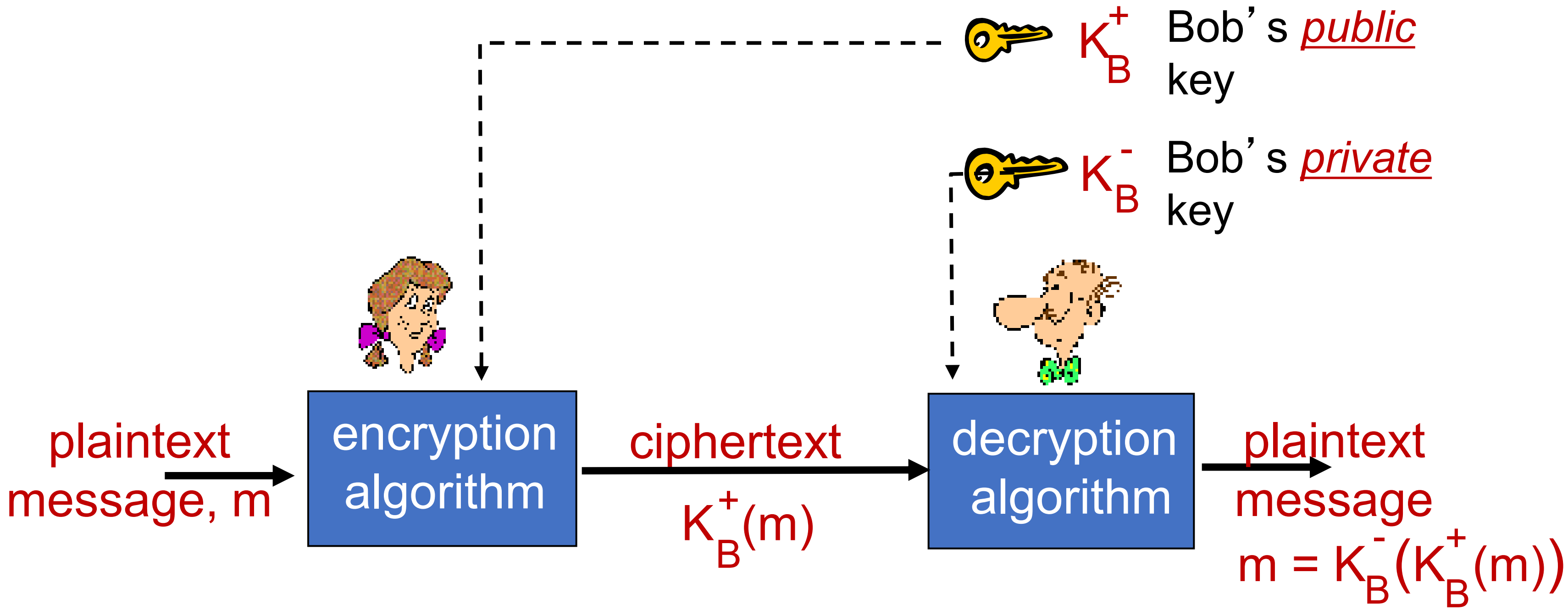
- requires sender, receiver know shared secret key
- **Q**: how to agree on key in first place (particularly if never "met")?

*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:

  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $x^d \bmod n = 14^2 \bmod 10 = 6$

# 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* mod $z = 1$ ).

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

# 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: | bit pattern | m | $m^e$ | $c = m^e \bmod n$ |
|---|---|---|---|---|
| | 00001100 | 12 | 24832 | 17 |

| decrypt: | c | $c^d$ | $m = c^d \bmod n$ |
|---|---|---|---|
| | 17 | 481968572106750915091411825223071697 | 12 |

# RSA: another important property

The following property will be *very* useful later:

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

*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 (RSA is slow – only used to establish a temporary symmetric key
- Solution:
- **Hybrid encryption:** use public key crypto to establish a secure connection, then establish a second key – a 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