



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS2031 Telecommunications II

Introduction

Stefan Weber
sweber@tcd.ie

Overview

- Housekeeping
- Motivation/HTML use-case
- Overview of Assignments

Housekeeping

- Lectures
 - 12 weeks (– reading week)
 - 2 slots per week

Friday 10:00-11:00 & 11:00-12:00, LB01
- Tutorials
 - 1 slot per week
 - Friday 13:00-14:00, LB01
- Labs / Assignments
 - Mondays 16:00-17:00, ICT Lab1
 - Tuesdays 16:00-17:00, ICT Lab2

Tutorials

- Error Detection and Correction
- Data Compression
- Error and Flow Control
- Point-to-Point Protocol / HDLC
- Medium Access Control
- Spanning Tree
- Internet Protocol
- Routing

Assignments

- Lectures
- Tutorials
- Labs/Assignments
 - Flow Control
 - Protocol Development
- Attendance
 - Use common sense ☺

Assignments

- Assignment 1:

Flow Control & Connection Management

Marking:

50% Implementation

50% Documentation

- Assignment 2:

Routing Protocols

Assignment Timeline

- Preliminary Deadlines:

4th November: Flow Control

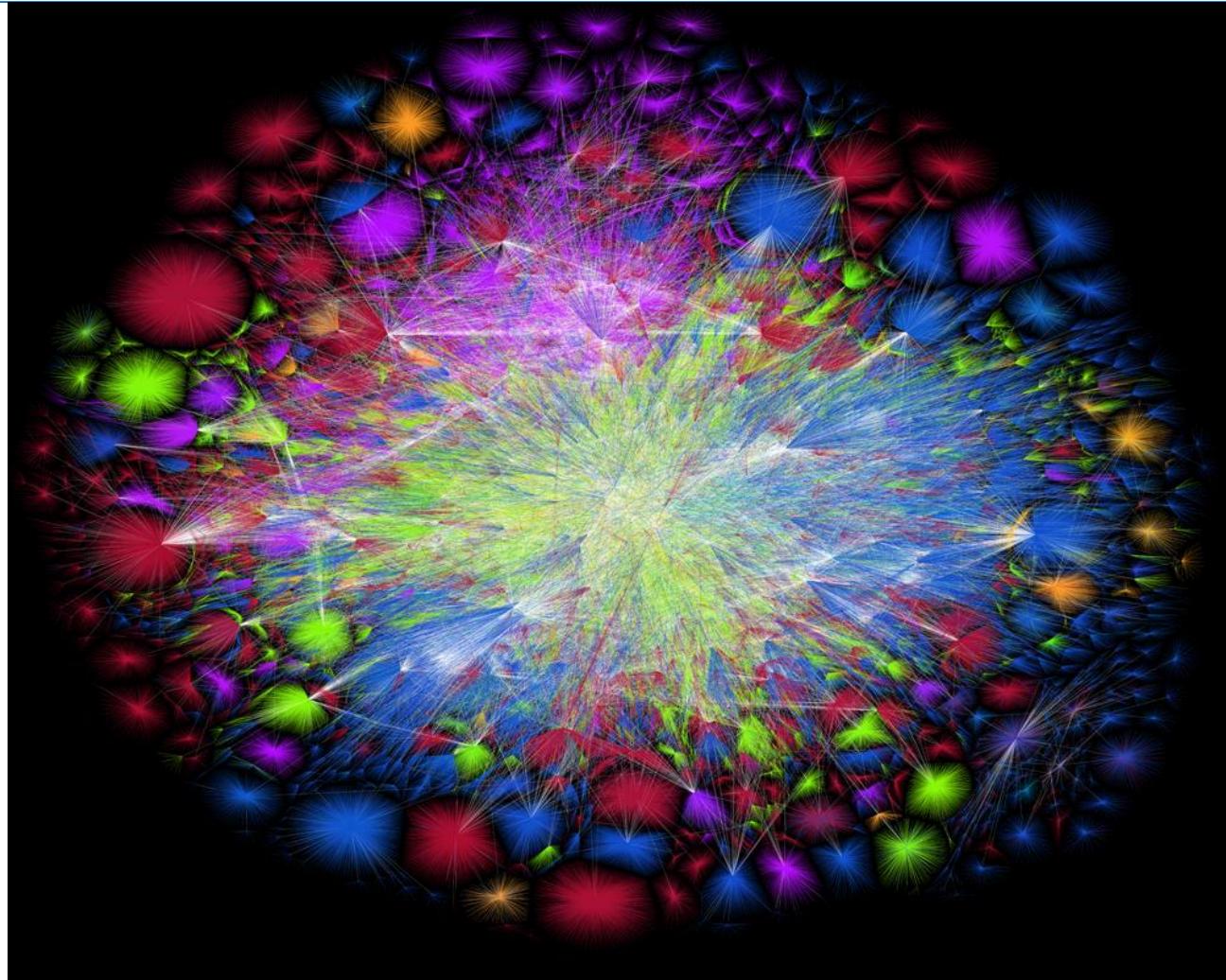
16th December: Routing Protocols

- Submission through Blackboard **mymodule.tcd.ie**
- Deadlines on Blackboard count

Recommended Books

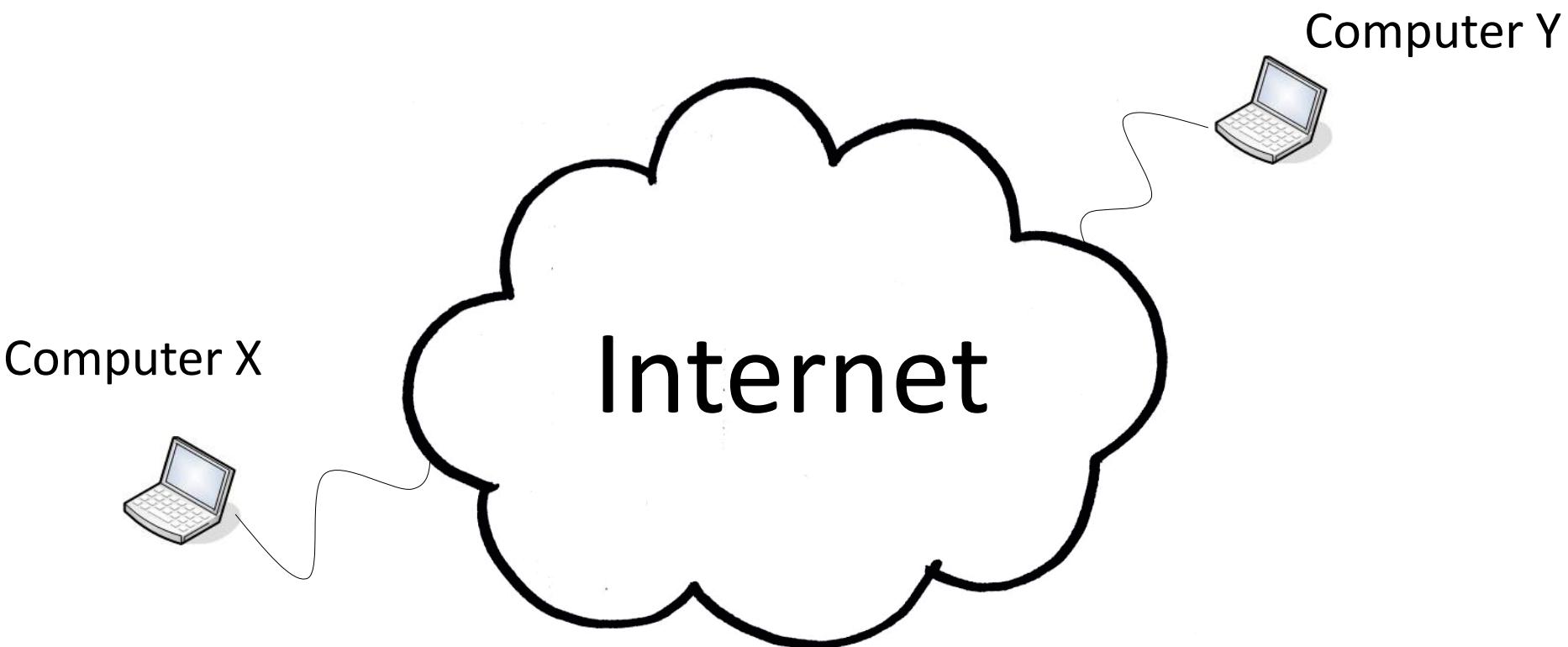
- Andrew Tanenbaum & David Wetherall
 - Title: Computer Networks
 - ISBN-10: 0132126958
- Behrouz A. Forouzan
 - Title: Data Communications and Networking
 - ISBN: 9780071315869
 - Chapter 1, 2, 9-22 – 5th edition

Today's Internet

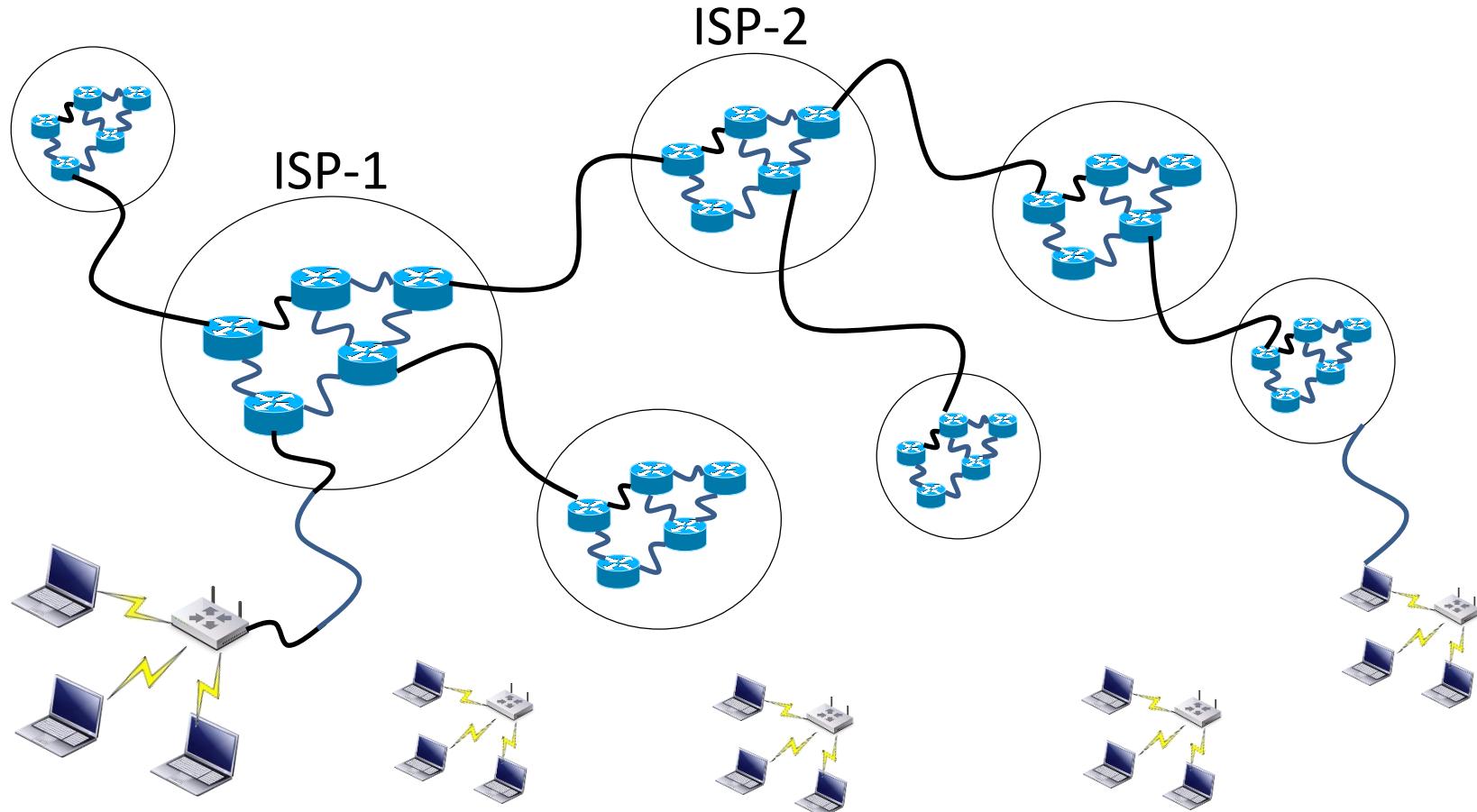


* Figure is courtesy of <http://www.opte.org/>

General View of the Internet



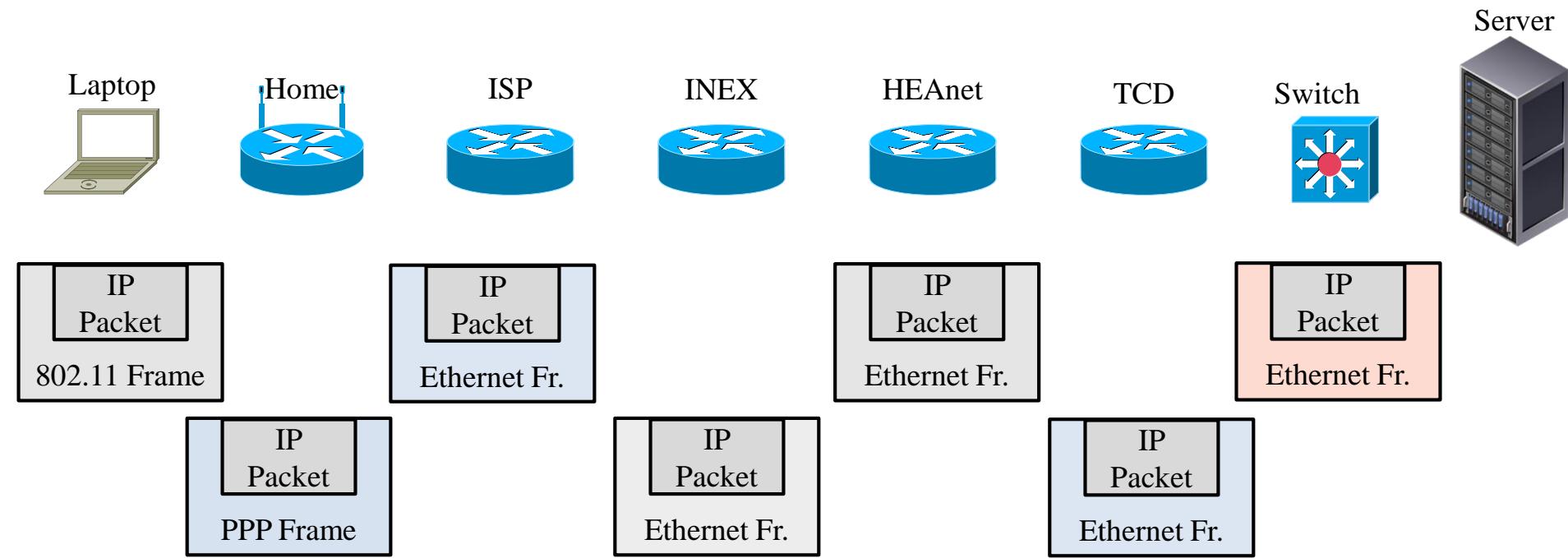
Internet = Network of Networks



Home Network

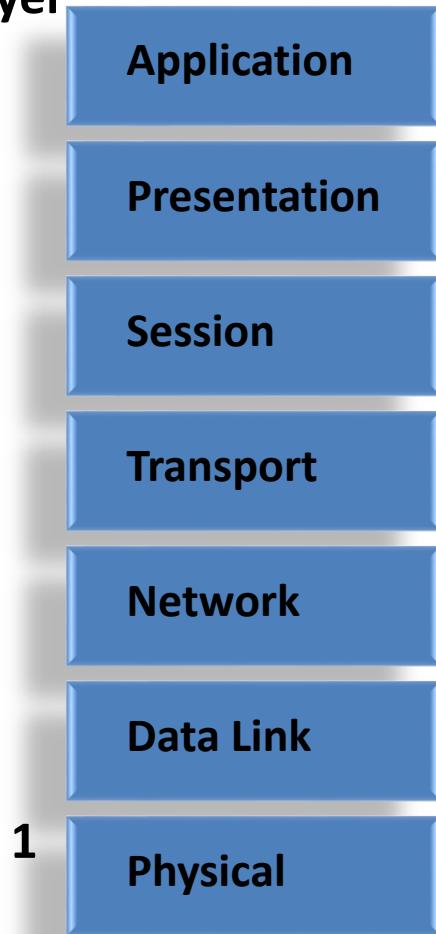
11

Hop-by-Hop Communication



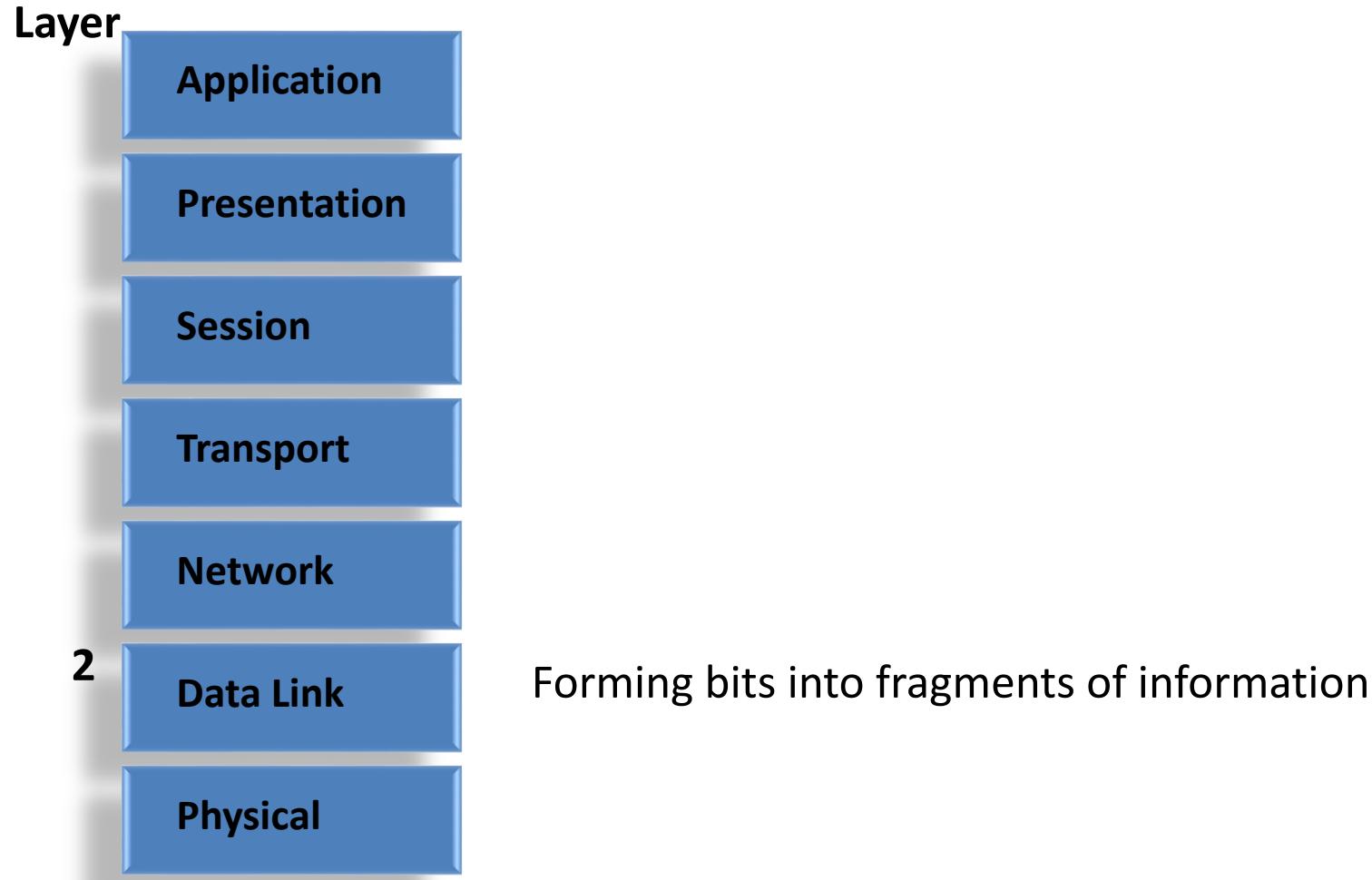
OSI Stack

Layer



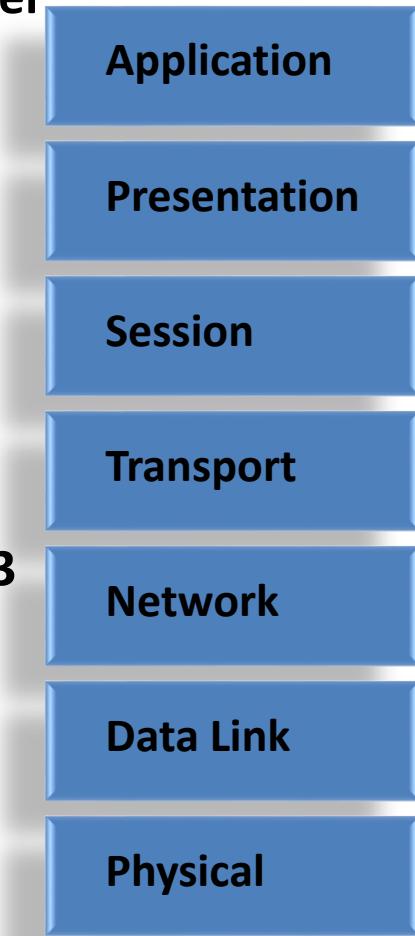
Interpreting signals as bits

OSI Stack



OSI Stack

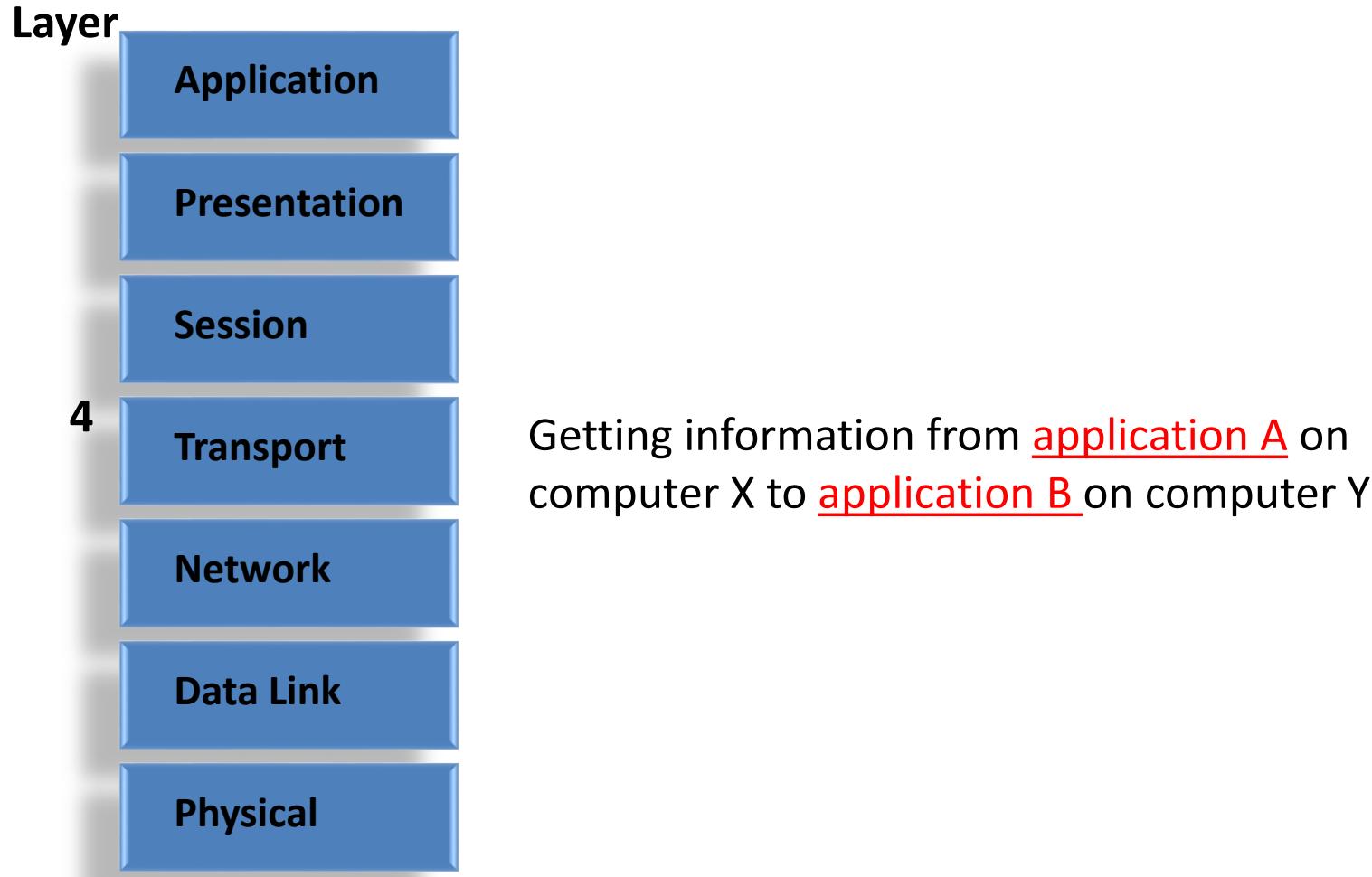
Layer



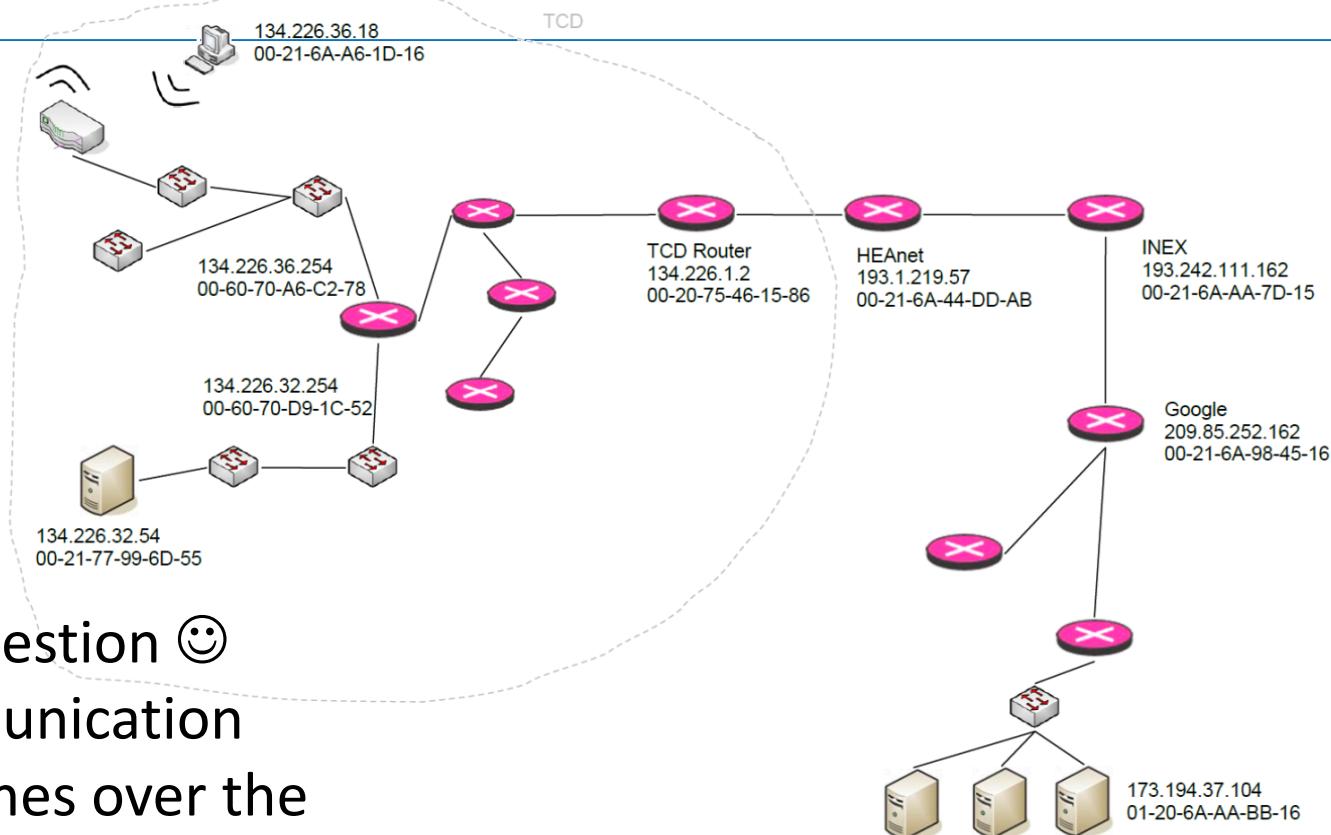
3

Getting packets of information from **network A** to **network B** over network C...Z

OSI Stack



Common Scenario



Recent Interview Question ☺
“Describe the communication
between two machines over the
Internet in as much detail as you
can.”

Dublin-located Games Company,
2014: 67 million players, 7.5mil/hr at peak

HTML Use Case

- Webbrowsers
 - Accepts URL e.g. `http://www.dsg.cs.tcd.ie/index.html`
 - Send out HTTP requests
 - `GET URI`
 - `PUT URI`

`GET index.html HTTP/1.1`

`Host: www.dsg.cs.tcd.ie`

`Connection: keep-alive`

`Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`

`Accept-Encoding: gzip,deflate,sdch`

`Accept-Language: en-US,en;q=0.8`

`Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3`

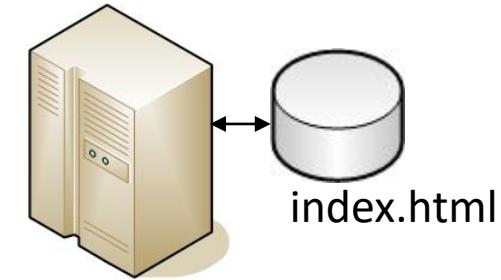
Clients & Servers

URL: <http://www.dsg.cs.tcd.ie/index.html>



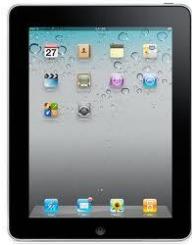
To www.dsg.cs.tcd.ie

```
GET index.html HTTP/1.1  
Host: www.dsg.cs.tcd.ie  
Connection: keep-alive  
  
....
```



www.dsg.cs.tcd.ie

Domain Name Servers (DNS)



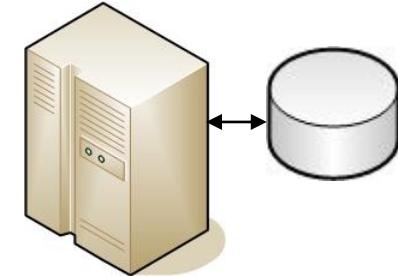
Its address is
134.226.36.14

What is
www.dsg.cs.tcd.ie?

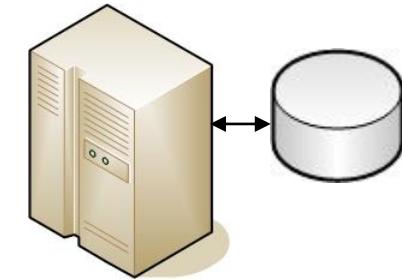


Name Server for dsg.cs.tcd.ie

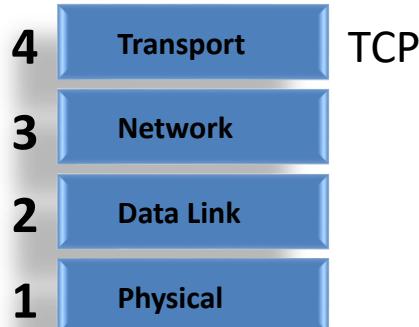
www AA 134.226.36.14



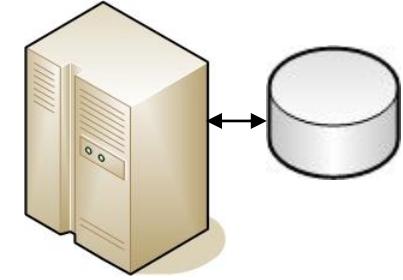
Transport Layer



I need to talk to **134.226.36.14**



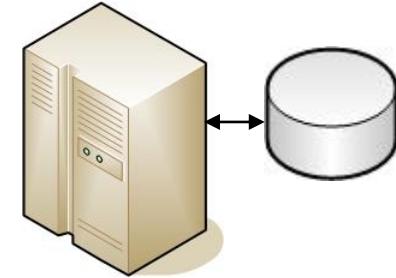
Transmission Control Protocol (TCP)



I need to talk to **134.226.36.14**

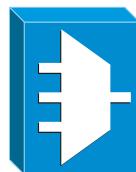
socket.connect (134.226.36.14, 80)

Transmission Control Protocol (TCP)



I need to talk to **134.226.36.14**

`socket.connect (134.226.36.14, 80)`



TCP Socket



Incoming



Outgoing



Incoming

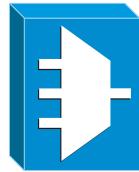
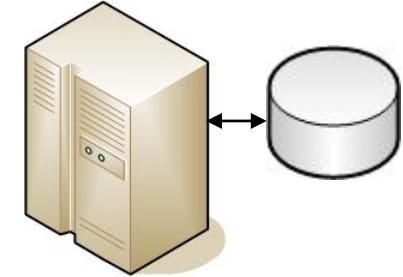


Outgoing

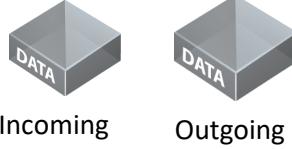
Transmission Control Protocol (TCP)



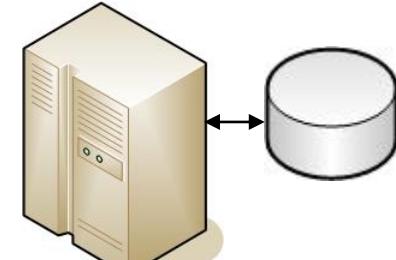
socket.connect(134.226.36.14, 80)



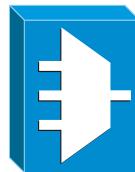
Socket: 54321



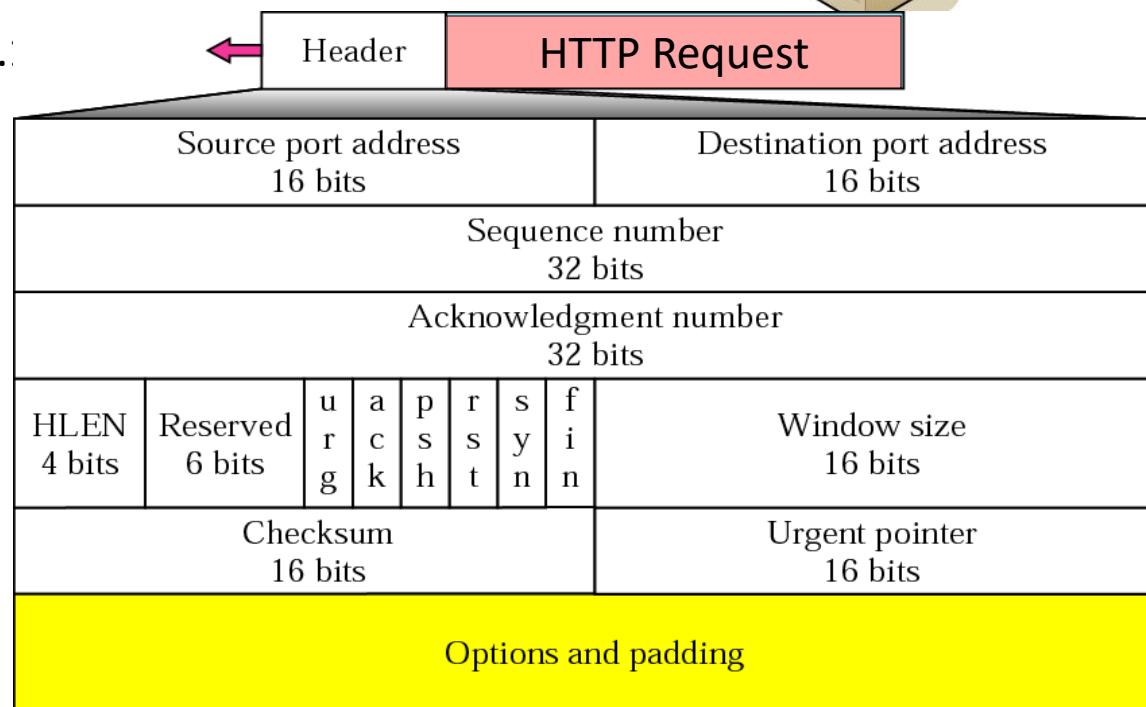
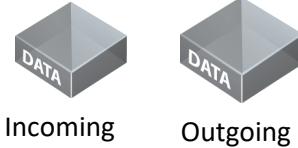
TCP Header & Payload



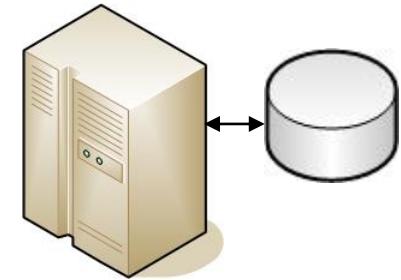
socket.connect(134.226.36.)



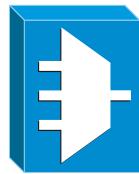
Socket: 54321



Transmission Control Protocol (TCP)



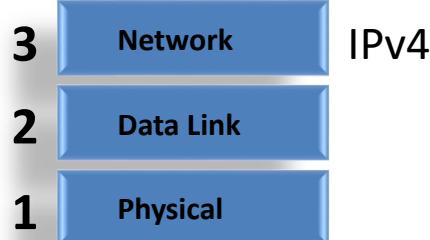
socket.connect(134.226.36.14, 80)



Synchronize State



Socket: 54321



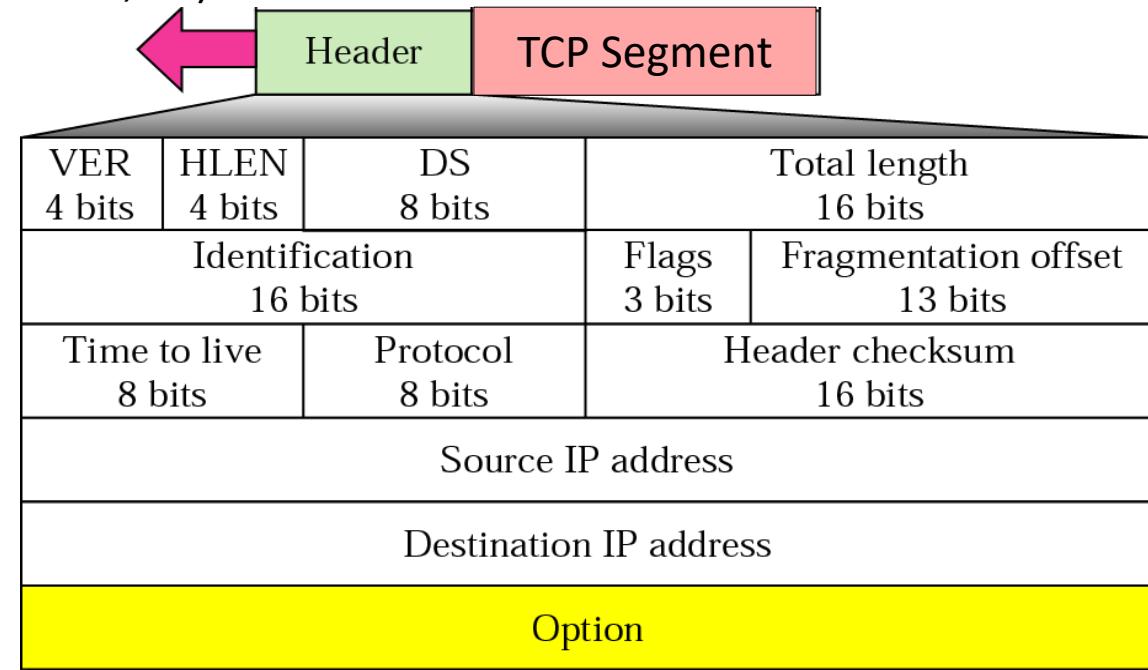
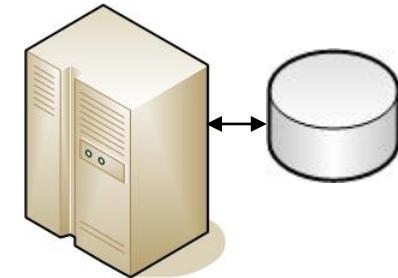
Internet Protocol (IPv4)



socket.connect(134.226.36.14, 80)



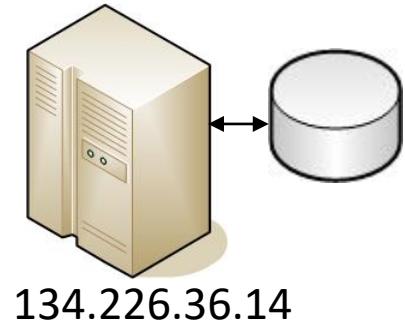
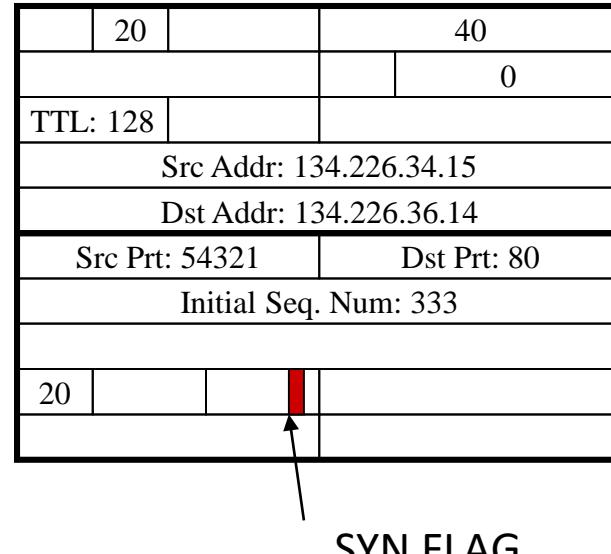
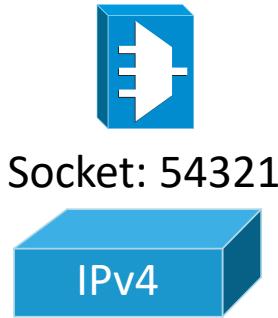
Socket: 54321



Internet Protocol (IPv4)



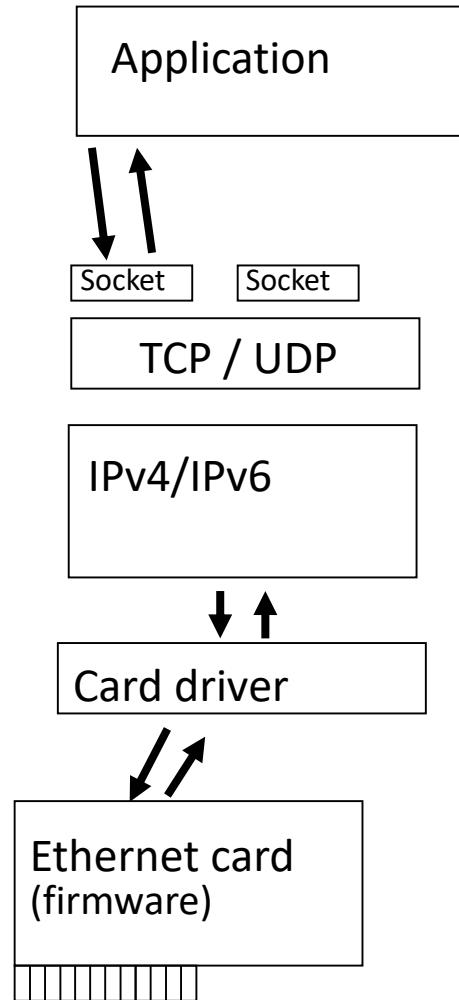
socket.connect(134.226.36.14, 80)



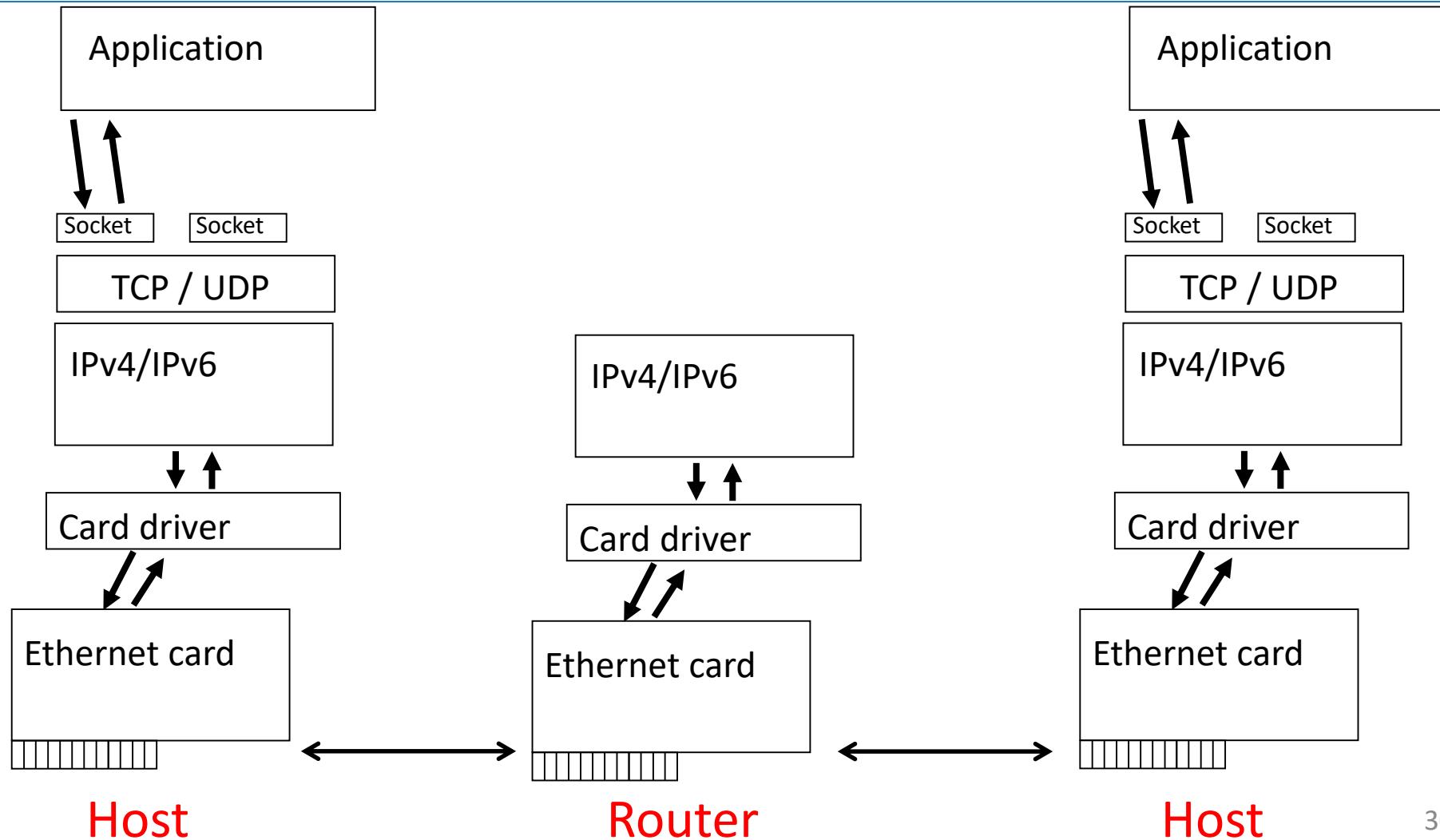
IP Header

TCP Header

Layer 4 to Layer 2



Host to Router to Host



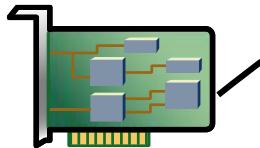
Ethernet between Routers



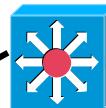
Socket: 54321



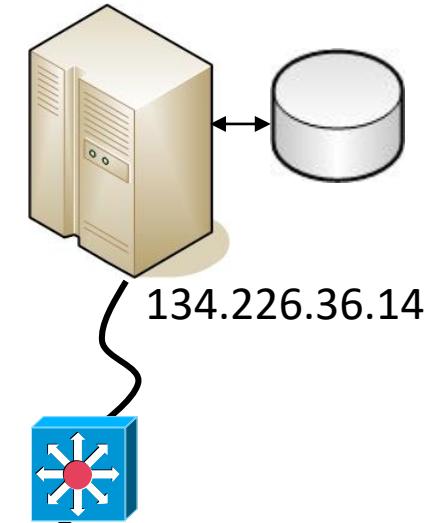
socket.connect(134.226.36.14, 80)



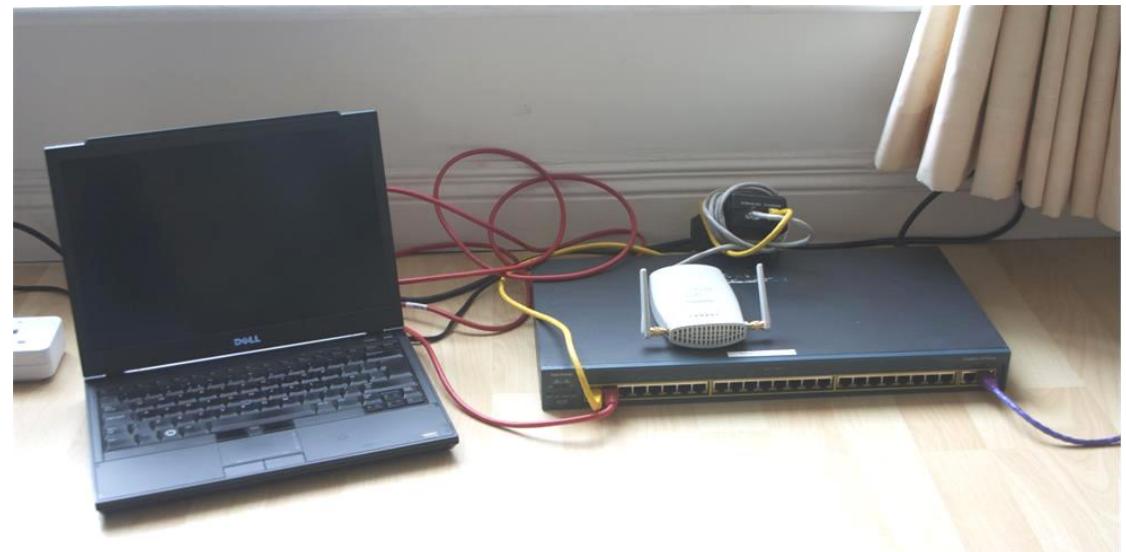
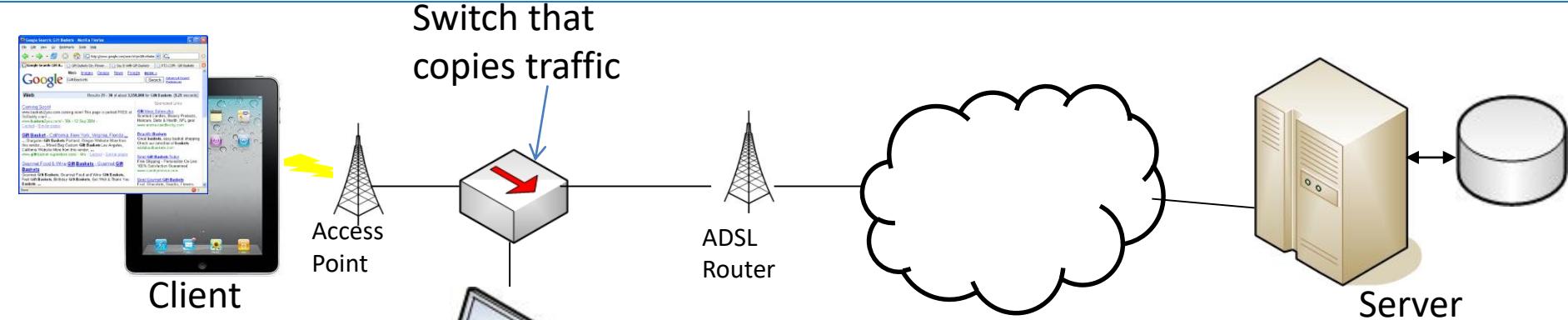
00:DD:CC:BB:FF:CC



00:CC:EE:BC:FE:CD



Capturing Traffic



32

Captured IP Packets

192.168.1.110	78 49536 > 80 [SYN] Seq=2805732566 Win=65535 Len=0 MSS=1460 WS=16
	66 80 > 49536 [SYN, ACK] Seq=2374059617 Ack=2805732567 Win=14600 Len=0
192.168.1.110	60 49536 > 80 [ACK] Seq=2805732567 Ack=2374059618 Win=262144 Len=0
192.168.1.110	326 [TCP segment of a reassembled PDU]
192.168.1.110	236 POST /ep.php HTTP/1.1 (application/x-www-form-urlencoded)
	60 80 > 49536 [ACK] Seq=2374059618 Ack=2805732839 Win=15744 Len=0
	60 80 > 49536 [ACK] Seq=2374059618 Ack=2805733021 Win=16768 Len=0
	427 HTTP/1.1 200 OK (text/html)
192.168.1.110	60 49536 > 80 [ACK] Seq=2805733021 Ack=2374059991 Win=261760 Len=0
192.168.1.110	60 49536 > 80 [FIN, ACK] Seq=2805733021 Ack=2374059991 Win=262144
	60 80 > 49536 [FIN, ACK] Seq=2374059991 Ack=2805733022 Win=16768

POST /ep.php HTTP/1.1

Host: foo.bar.com

User-Agent: foo/1.5 CFNetwork/548.1.4 Darwin/11.0.0

Content-Length: 182

Accept: */*

Accept-Language: en-us

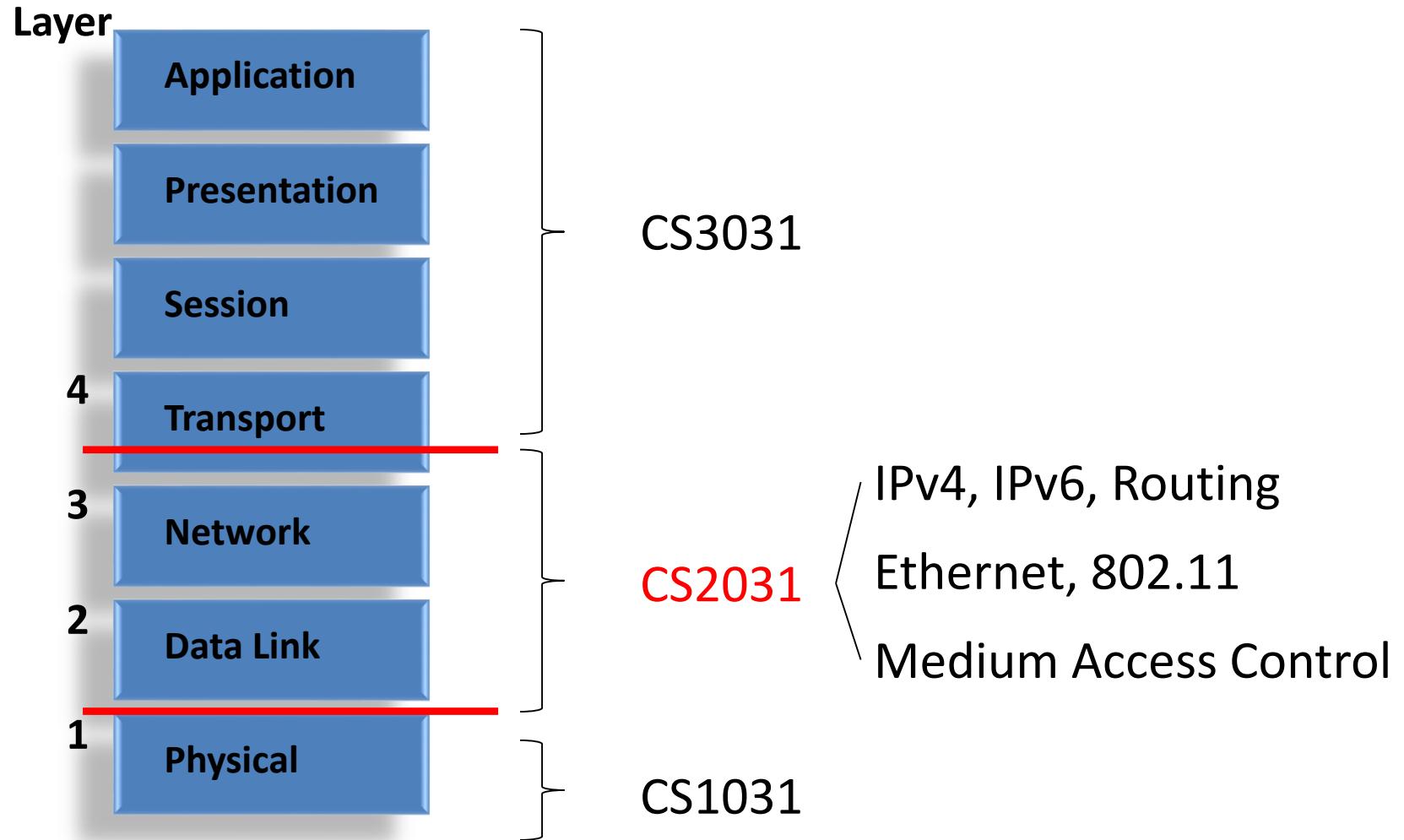
Accept-Encoding: gzip, deflate

Content-Type: application/x-www-form-urlencoded

Connection: keep-alive

udid=09e83bc00abc45f01f4935adf46803635052e677&
rand=985268246&hash=a4ae73033d507f4d1c99781d6e30d592&
action=getStatus¶ms=rating%3Ainvites%3Agifts%3Aposts
%3AblogDate&lang=en&ver=1.5

OSI Stack



Learning Outcome

What you should get out of this:

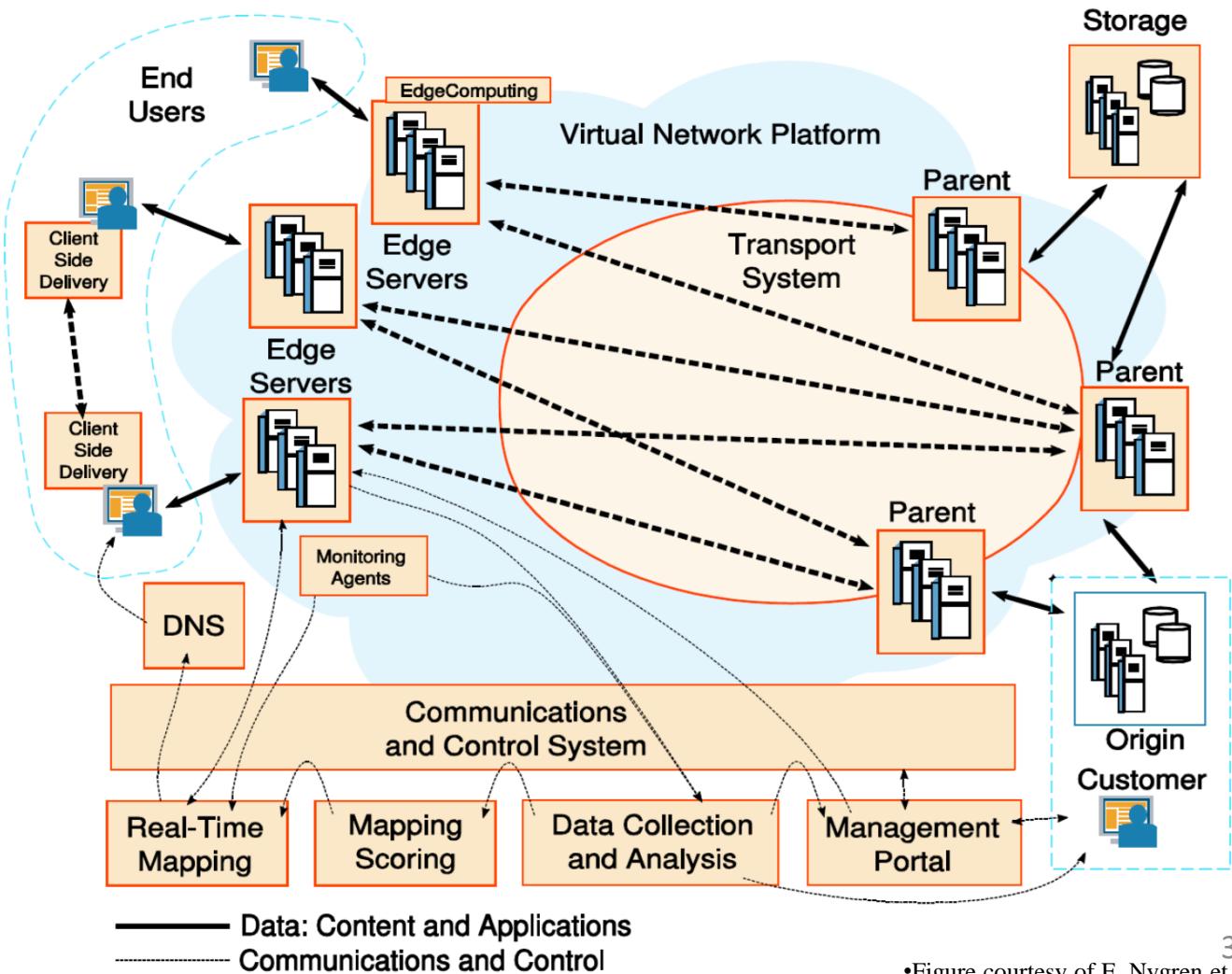
- Understanding of Protocol Design
- Understanding of Routing and IPv4
- Understanding Competition for the Medium
- Being able to use multi-threading

Advanced Topics

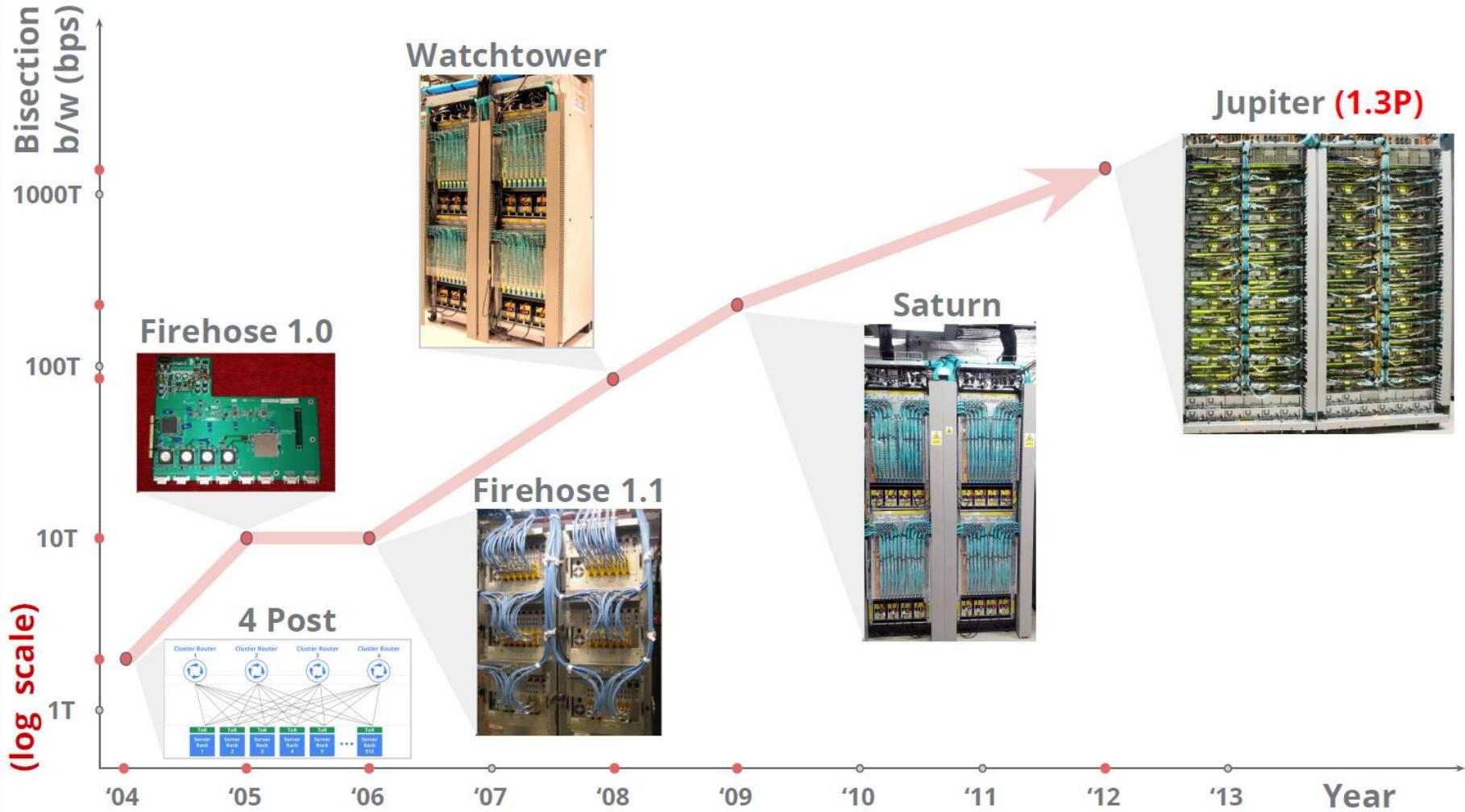
- Data Centre Comms
 - Amazon/Google/etc services
- Software-Defined Networking
- Network-Function Virtualization
- Data-Centric Networking
- Content-Delivery Networks
- Internet-of-Things

Akamai Scenario

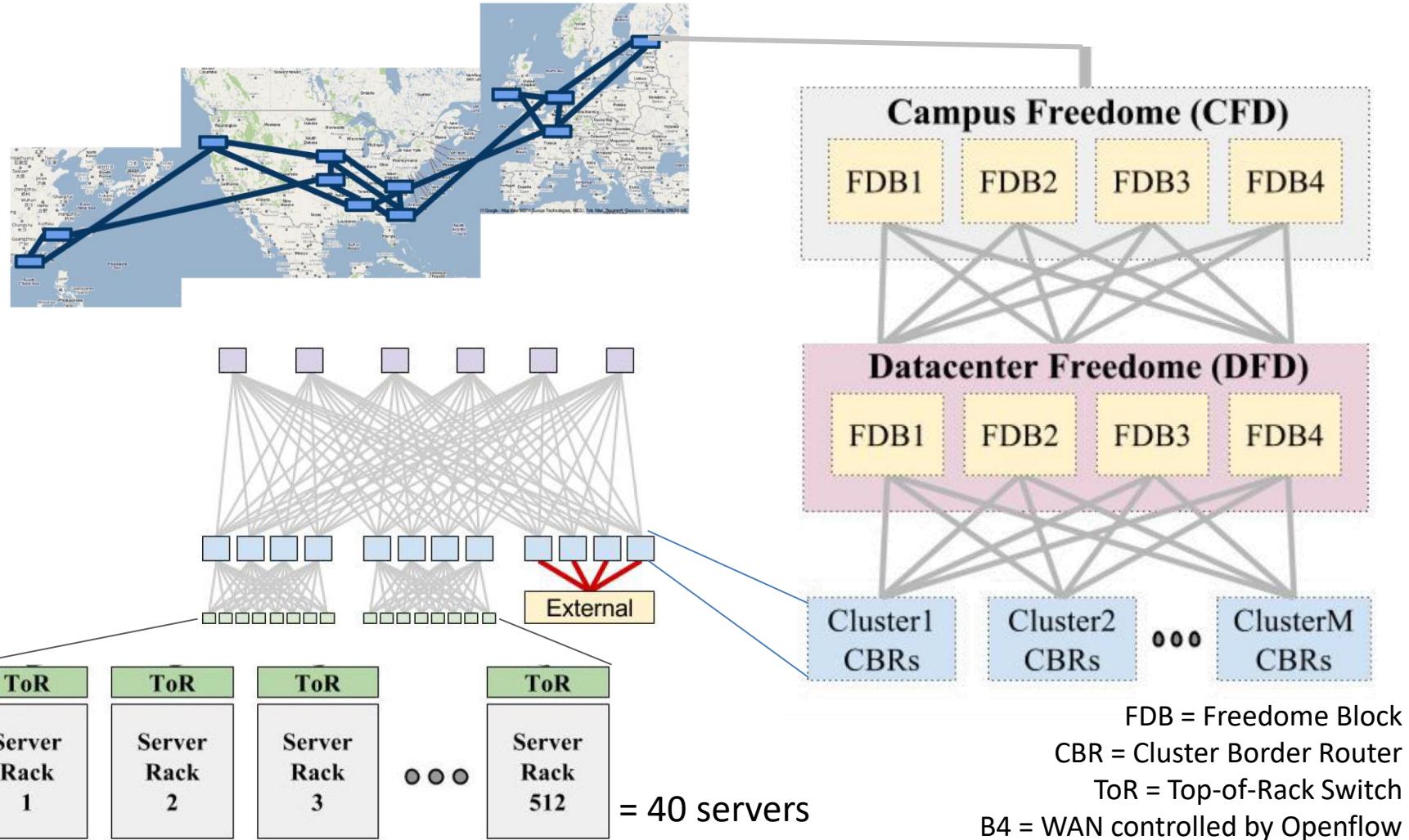
- DNS redirection
- Clusters of servers at points of presence
- Replication



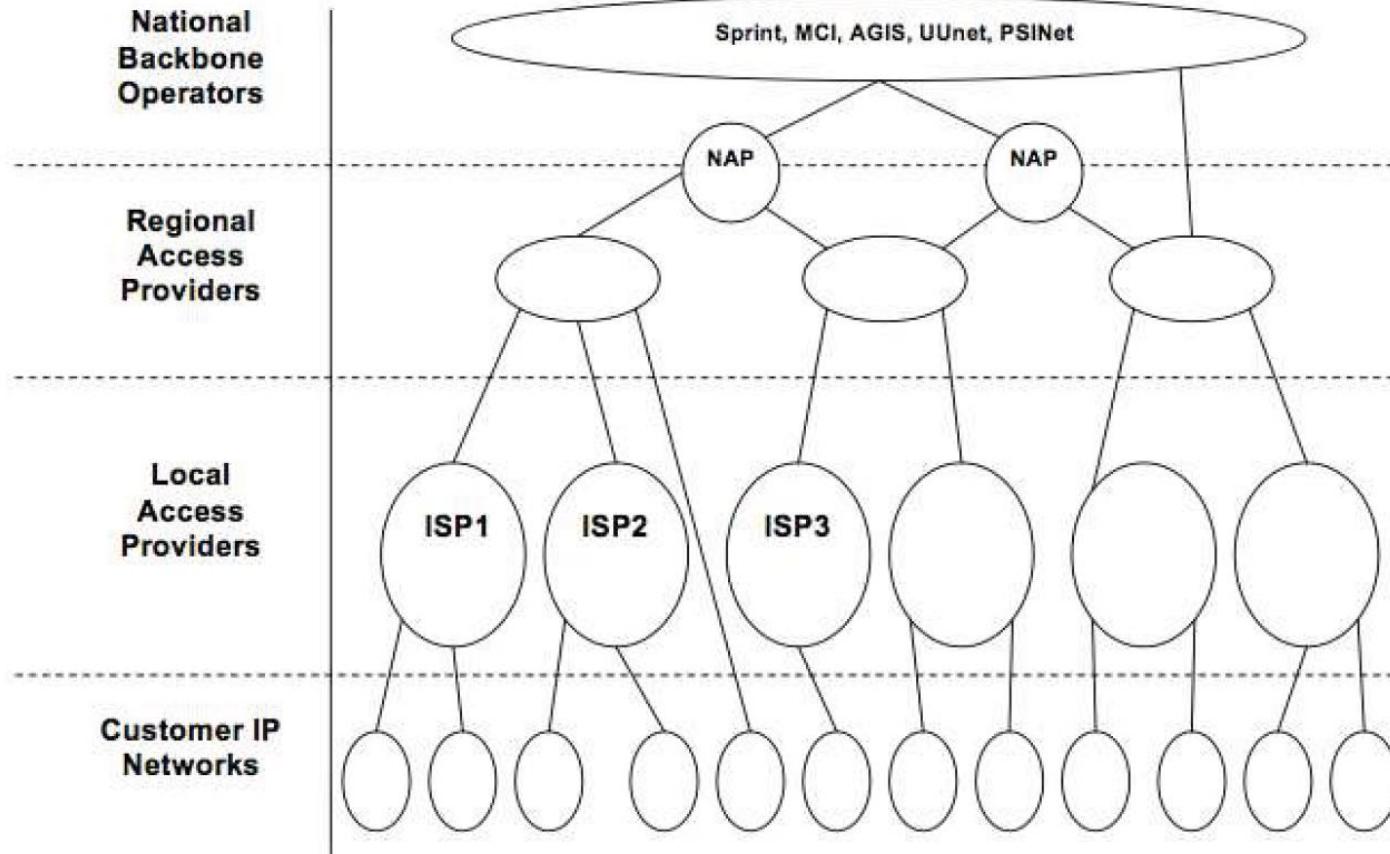
Google's Infrastructure



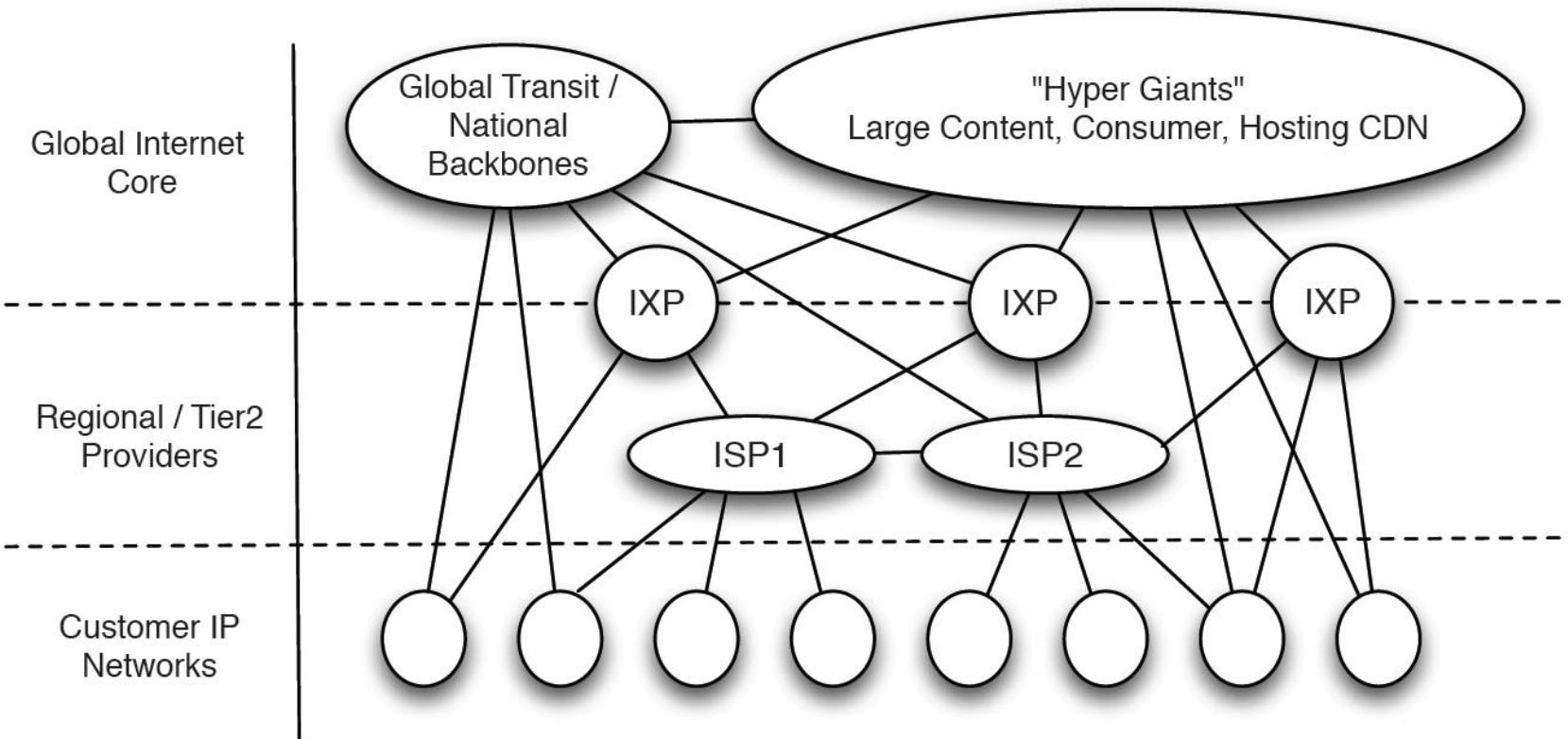
Google's B4 to Jupiter



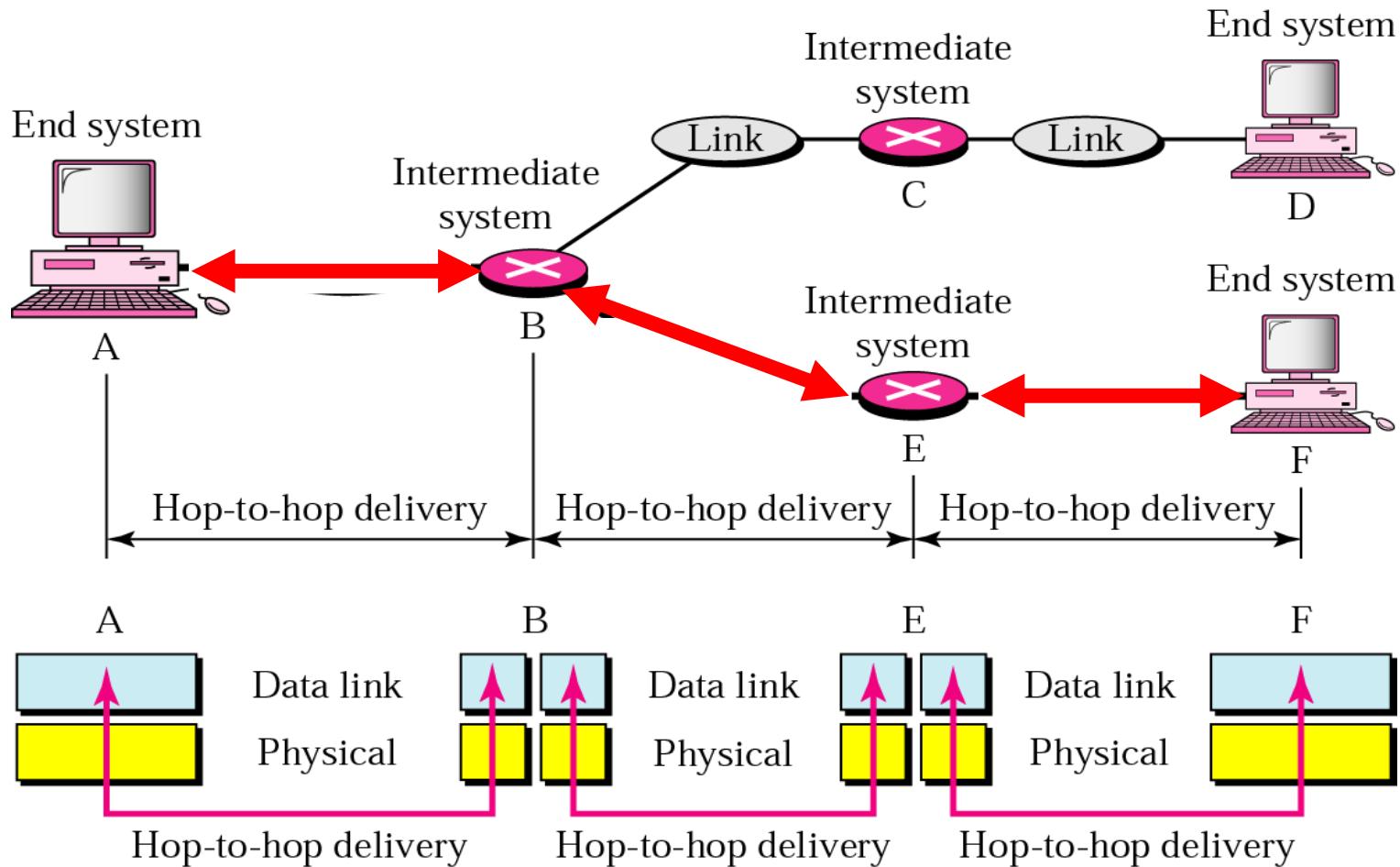
Traditional Logical Internet Topology



Emerging Logical Internet Topology

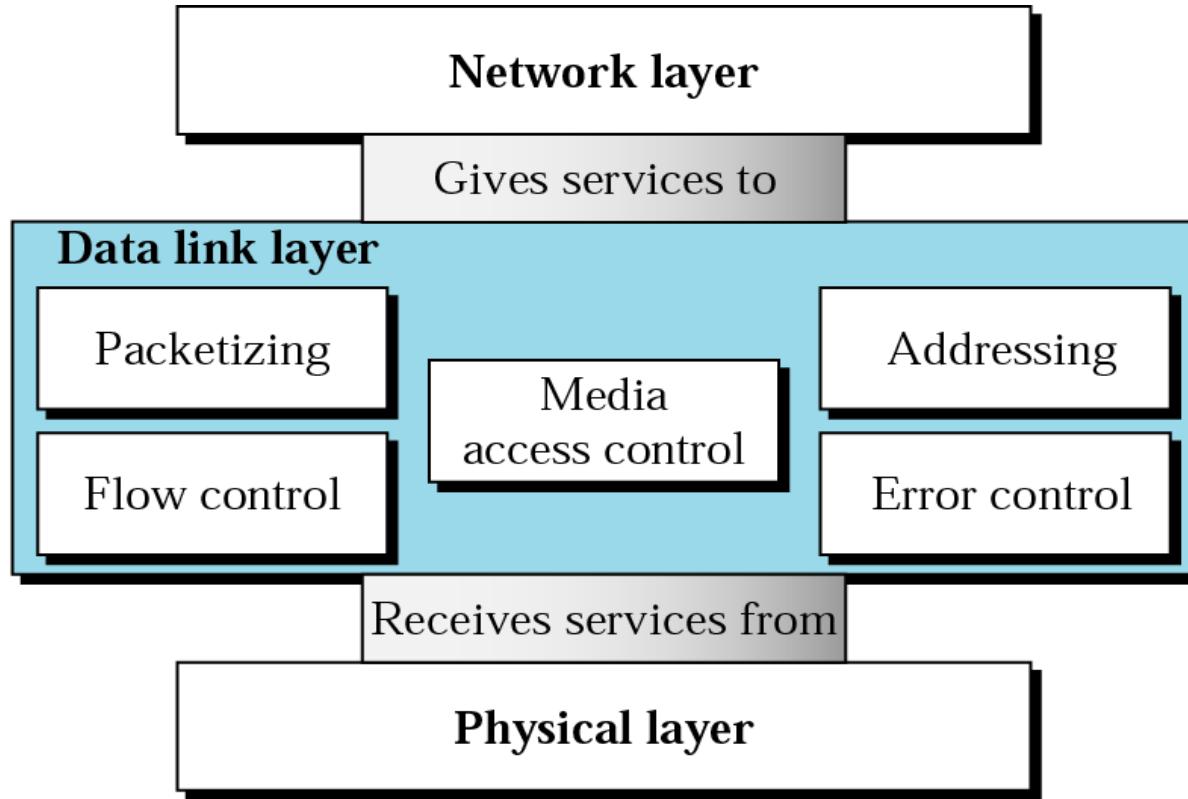


Link Layer



* Figure is courtesy of B. Forouzan 42

Duties of the Link Layer



The link layer is responsible for transmitting frames from one station to the next.

* Figure is courtesy of B. Forouzan 43



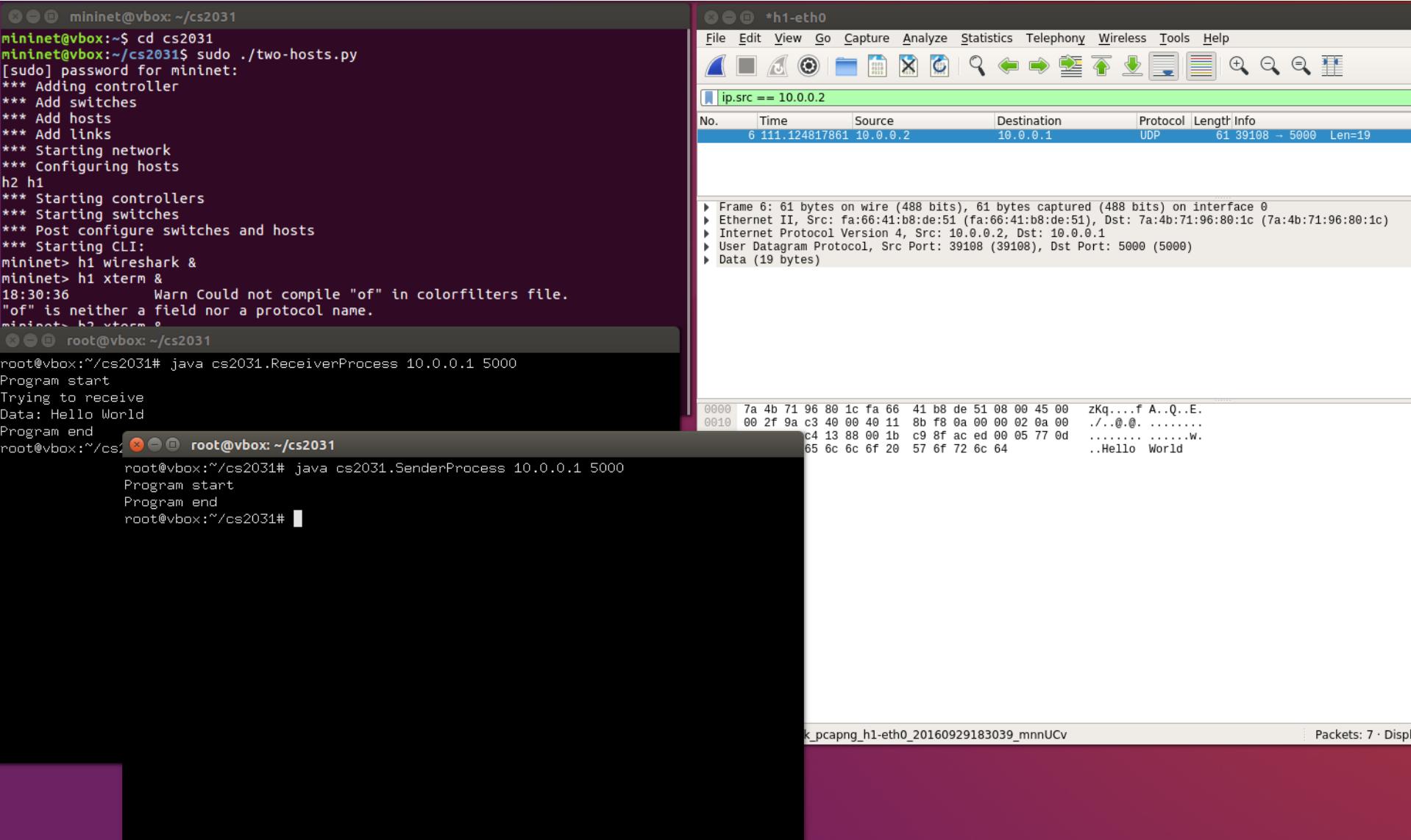
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin



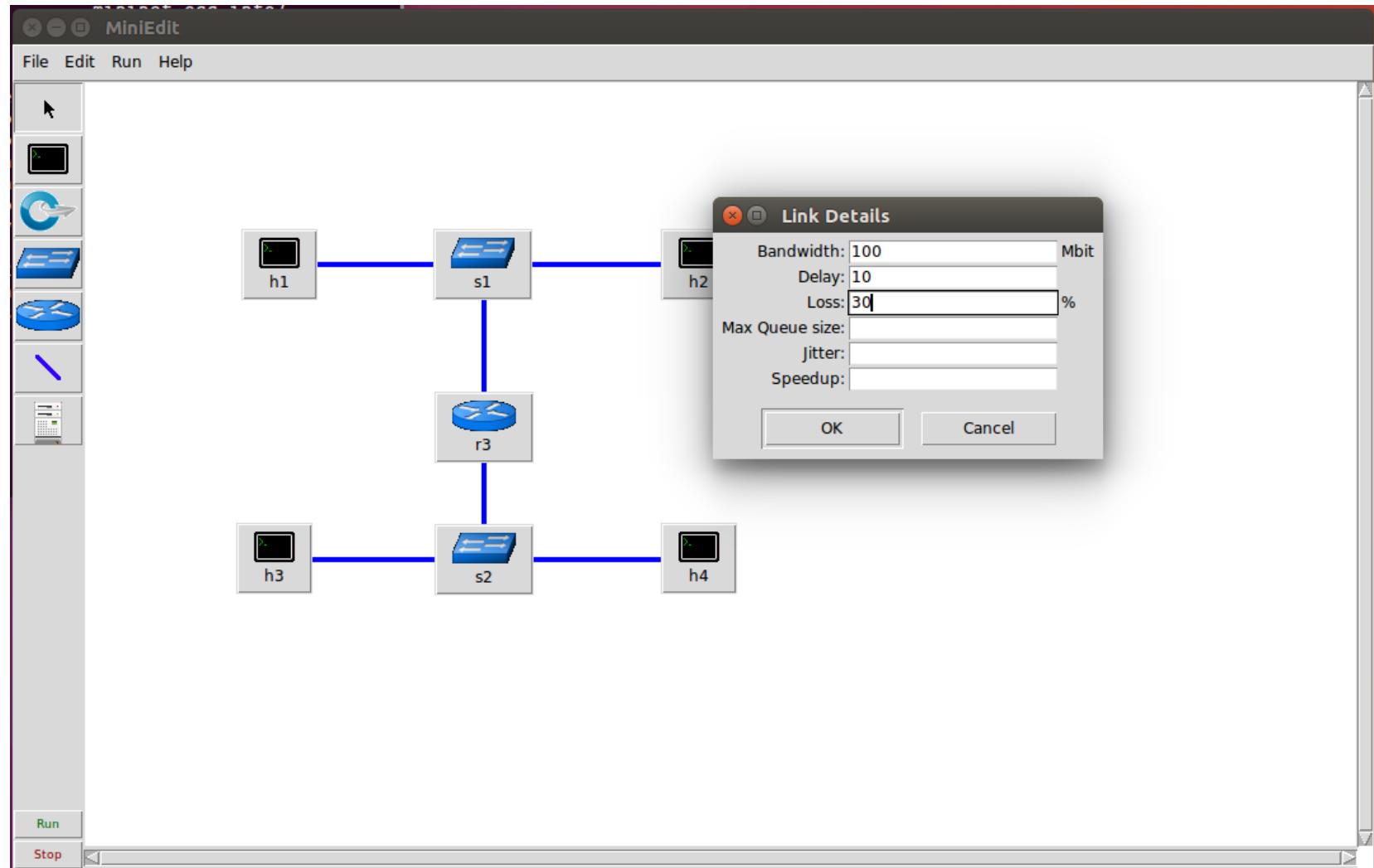
Changes from Last Year

- Suggested use of Mininet
 - Emulator for networks
 - Uses containers in Linux
- Reports should include
 - Capture of traffic (Wireshark/TCPdump) and
 - Explanation of captured traffic

Network Emulation

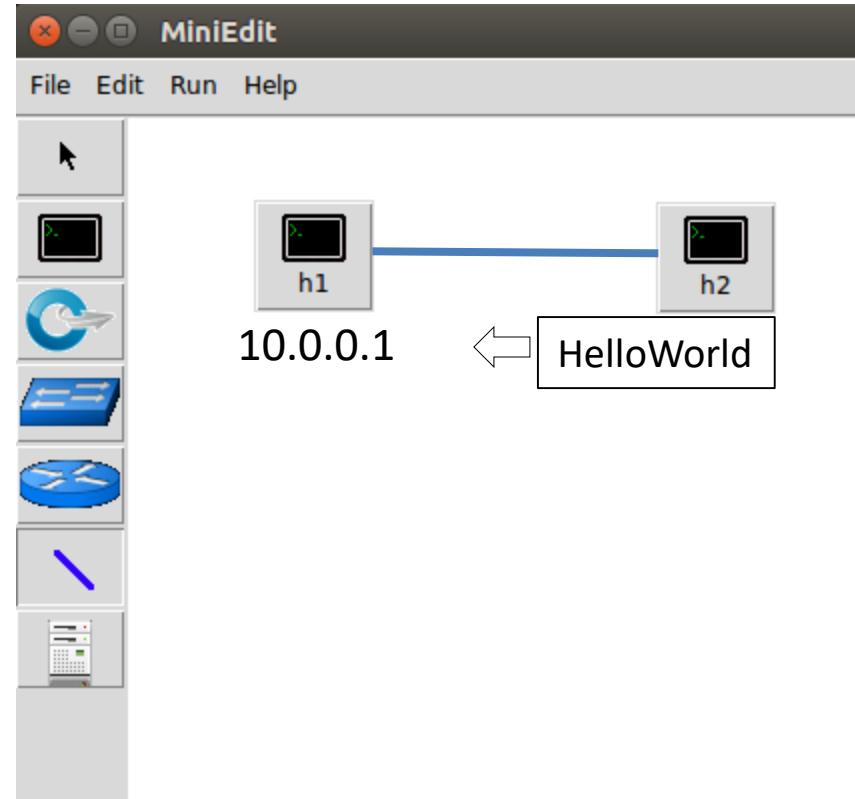


miniedit.py



Simple Example in CS2031.ova

- 2 hosts
- ReceiverProcess 10.0.0.1 5000
- SenderProcess 10.0.0.1 5000



Wireshark Capture

- Payload:
“Hello World”
- Payload Length:
19 Bytes???
- String+Extra Info

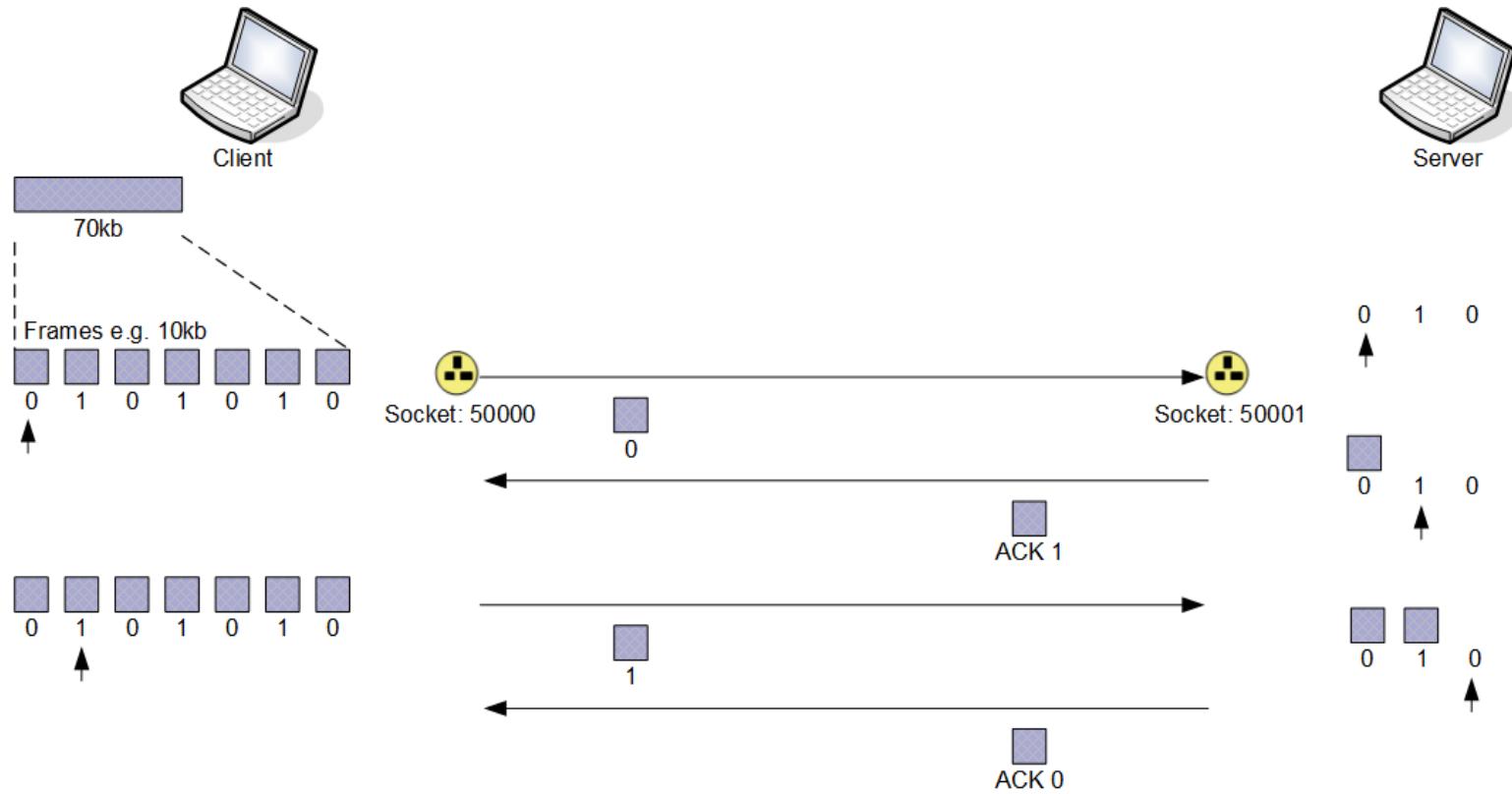
No.	Time	Source	Destination	Protocol	Length	Info
3	0.000067930	10.0.0.2	10.0.0.1	UDP	61	41162
▶ Frame 3: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface 0 ▶ Ethernet II, Src: 4a:9c:3d:6e:8c:4d (4a:9c:3d:6e:8c:4d), Dst: d2:a7:0a:c0:8e:c9 (d2:a7:0a:c0:8e:c9) ▶ Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1 ▼ User Datagram Protocol, Src Port: 41162 (41162), Dst Port: 5000 (5000) Source Port: 41162 Destination Port: 5000 Length: 27 Checksum: 0xc189 [validation disabled] [Stream index: 0] ▼ Data (19 bytes) Data: aced0005770d000b48656c6c6f20576f726c64 [Length: 19]						
0000 d2 a7 0a c0 8e c9 4a 9c 3d 6e 8c 4d 08 00 45 00J.=n.M..E. 0010 00 2f 20 a3 40 00 40 11 06 19 0a 00 00 02 0a 00 ./ .@. 0020 00 01 a0 ca 13 88 00 1b c1 89 ac ed 00 05 77 0dW. 0030 00 0b 48 65 6c 6c 6f 20 57 6f 72 6c 64 ..Hello World						

Wireshark Capture of byte[]

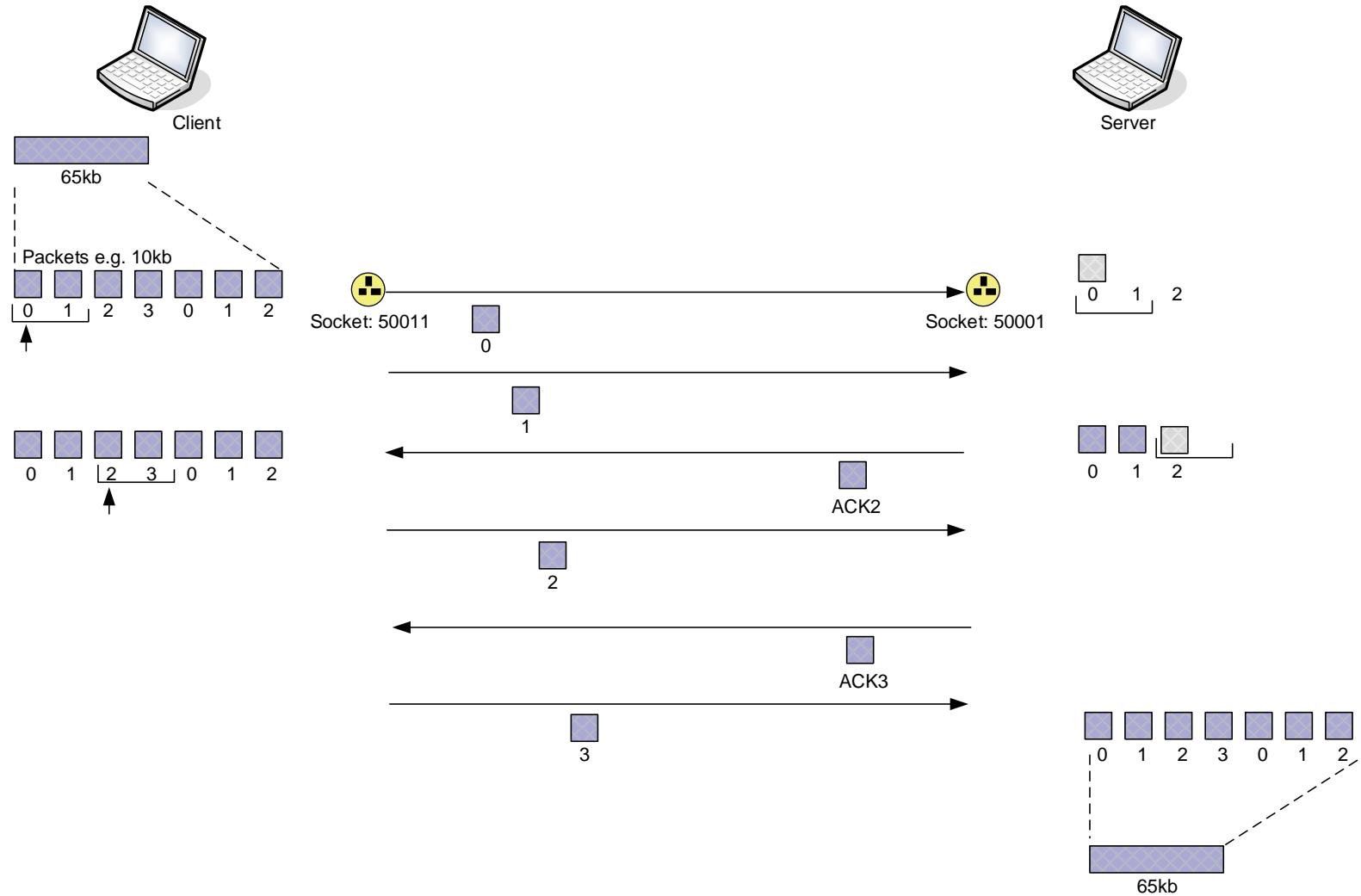
```
▶ Frame 4: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface 0
▶ Ethernet II, Src: 52:0d:eb:05:1e:1d (52:0d:eb:05:1e:1d), Dst: 0a:a9:5e:f9:d4:e0 (0a:a9:5e:f9:d4:e0)
▶ Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1
▼ User Datagram Protocol, Src Port: 33428 (33428), Dst Port: 5000 (5000)
  Source Port: 33428
  Destination Port: 5000
  Length: 18
  ▶ Checksum: 0x4c9f [validation disabled]
    [Stream index: 1]
▼ Data (10 bytes)
  Data: 0102030405060000000000
  [Length: 10]
```

Hex	Dec	ASCII
0000	0a a9 5e f9 d4 e0 52 0d eb 05 1e 1d 08 00 45 00	..^...R.E.
0010	00 26 77 5a 40 00 40 11 af 6a 0a 00 00 02 0a 00	.&wZ@.0. .j.....
0020	00 01 82 94 13 88 00 12 4c 9f 01 02 03 04 05 06 L.....
0030	00 00 00 00

Stop & Wait



Selective Repeat





Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin





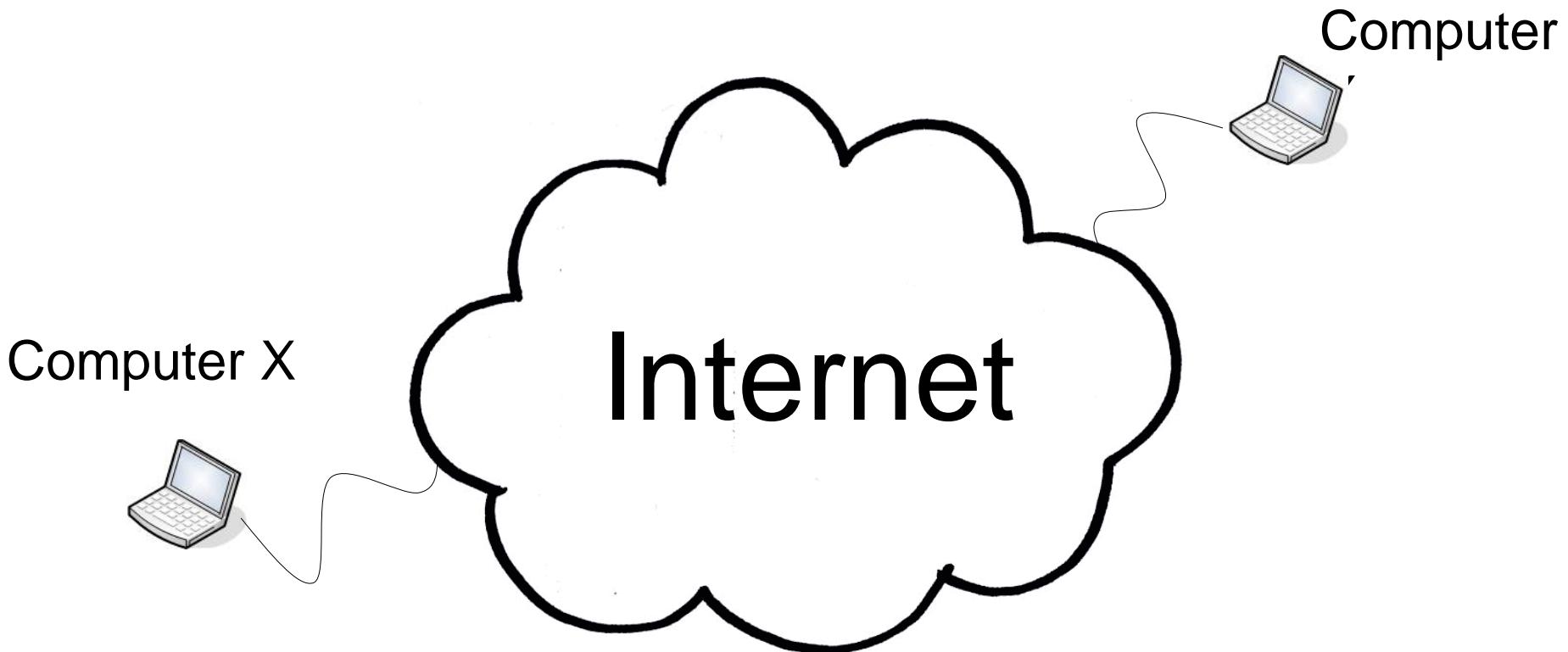
CS2031

Telecommunications II

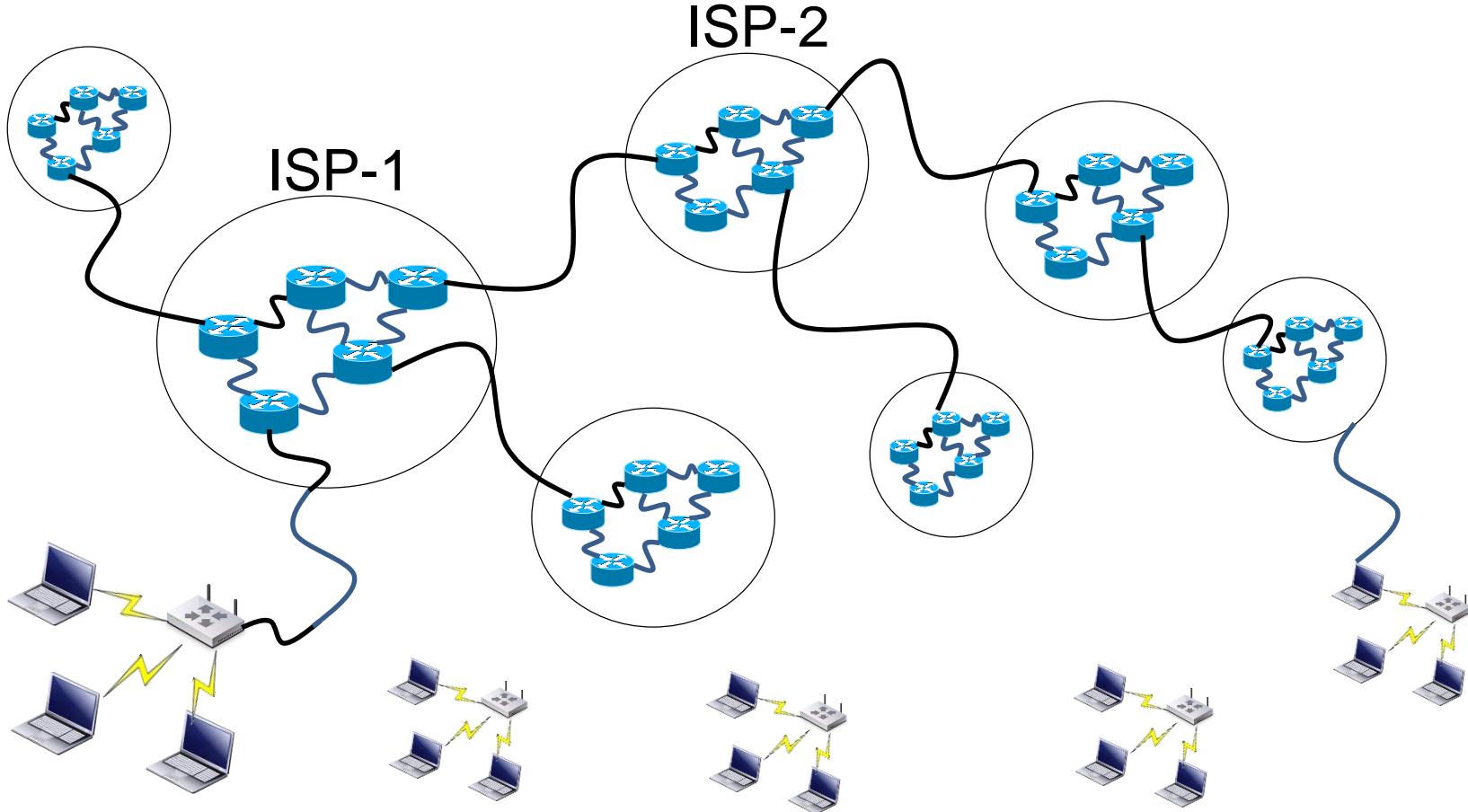
Flow Control



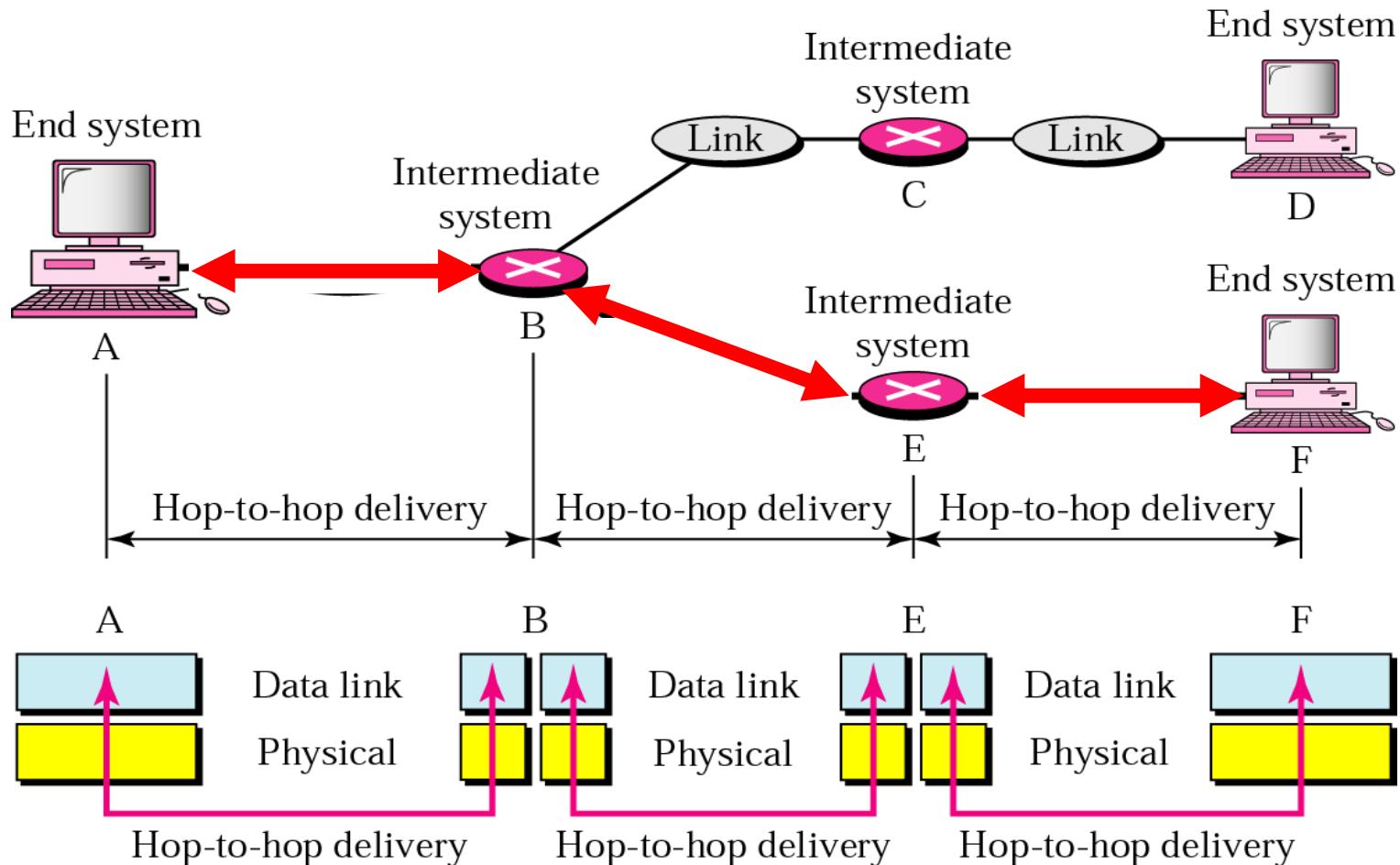
Aim of the Module



Internet = Network of Networks

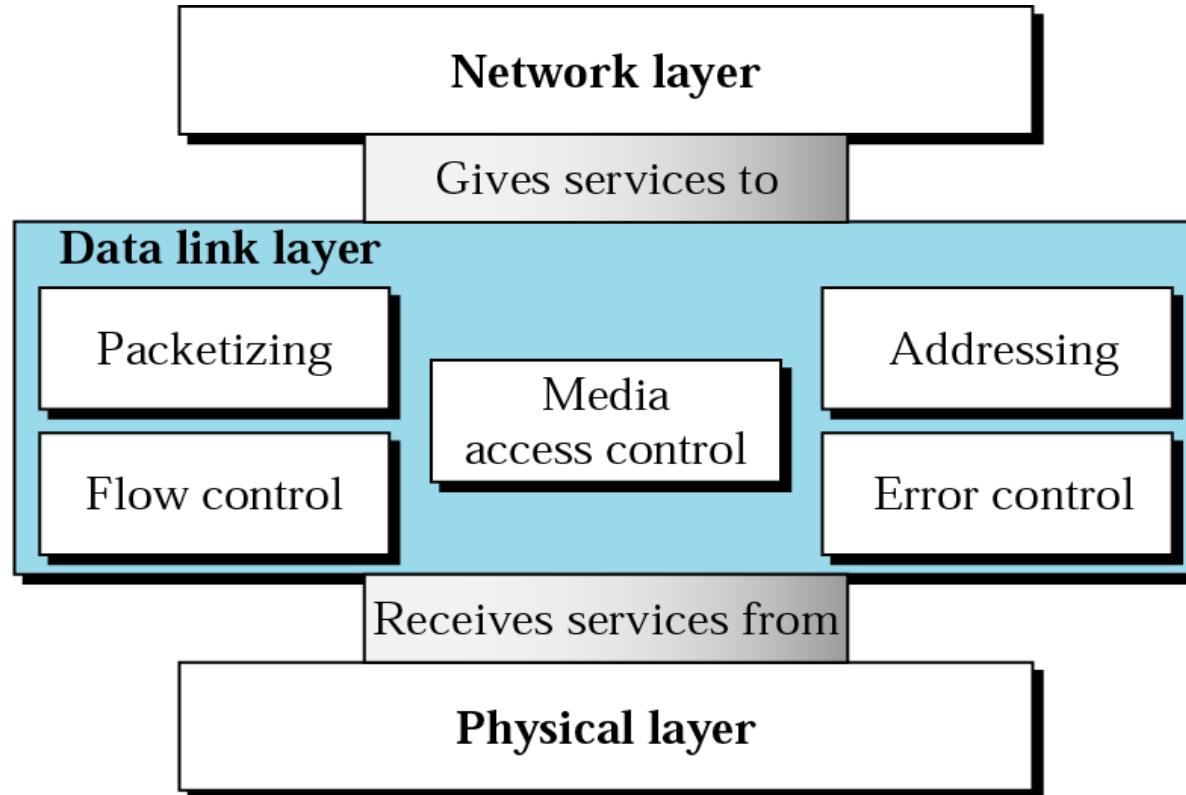


Link Layer



* Figure is courtesy of B. Forouzan

Duties of the Data Link Layer

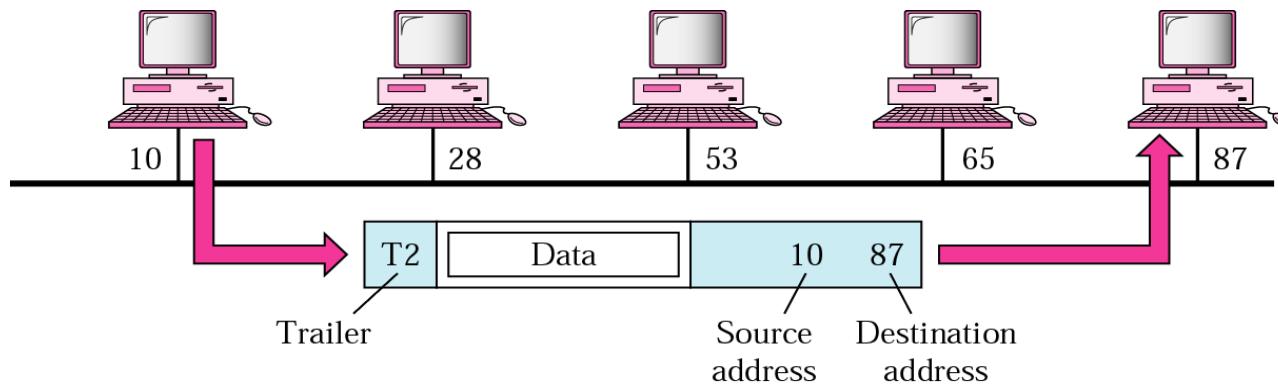


The data link layer is responsible for transmitting frames from one node to the next.

* Figure is courtesy of B. Forouzan

Packetizing & Addressing

- Packetizing: Encapsulating data in frame or cell i.e. adding header and trailer
- Addressing: Determining the address of the next hop (LANs) or the virtual circuit address (WANs)



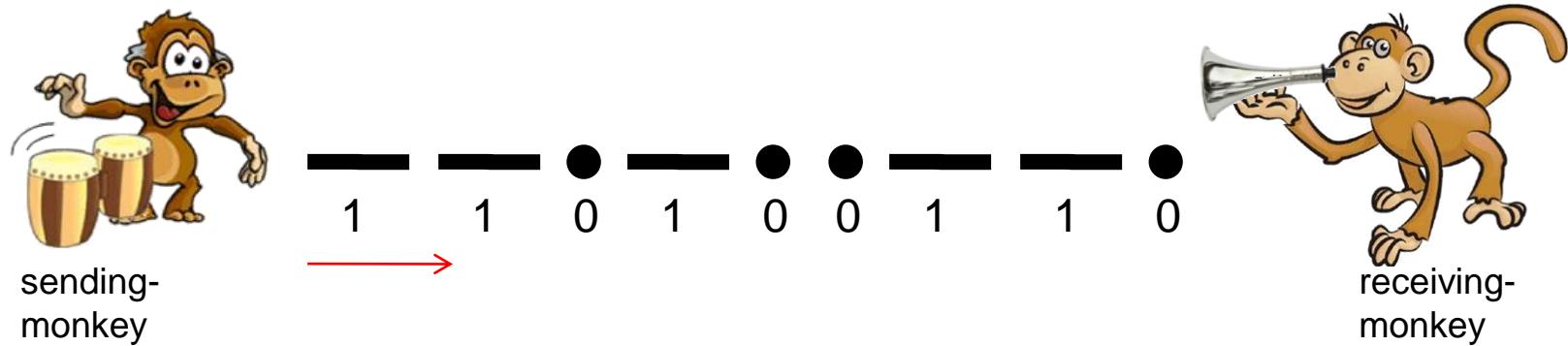
* Figure is courtesy of B. Forouzan

Communication

- What is telecommunication really about?

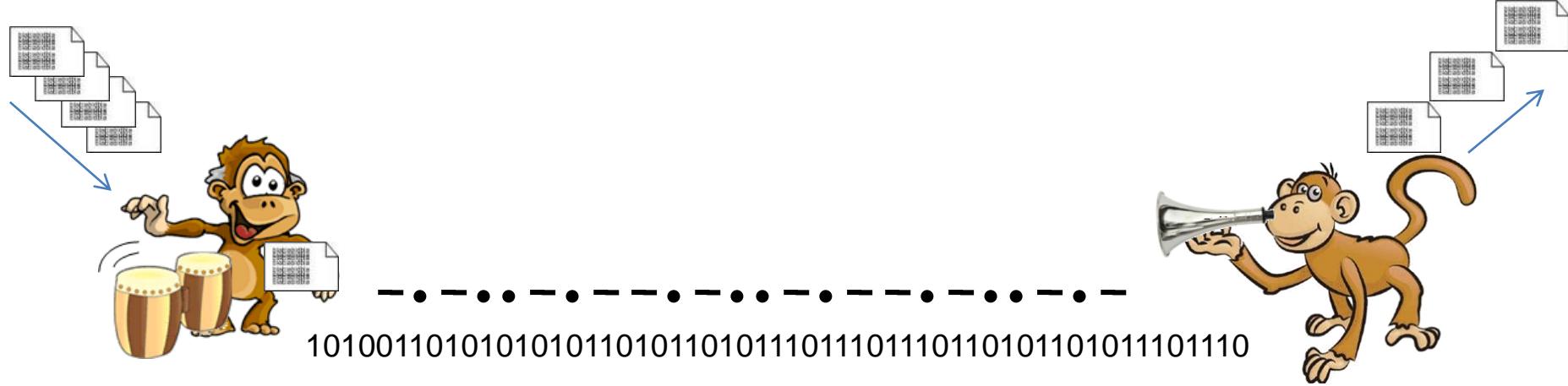
Communication

- What is telecommunication really about?



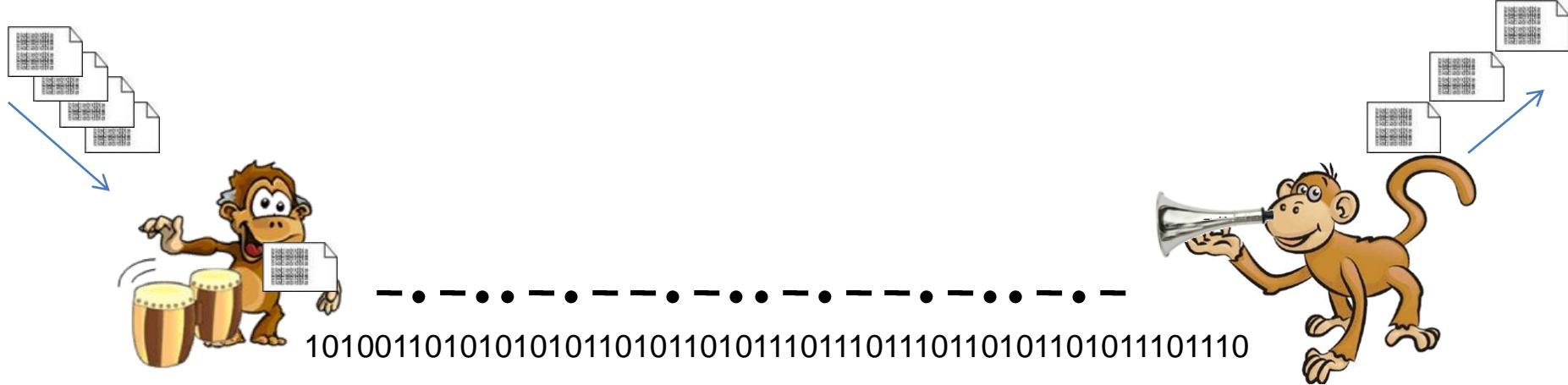
Bongo-Playing Monkeys Doing Morse-code!

Framing



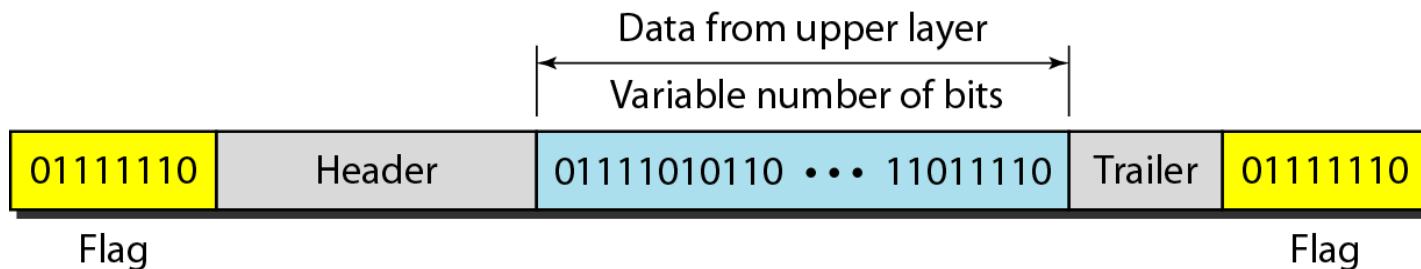
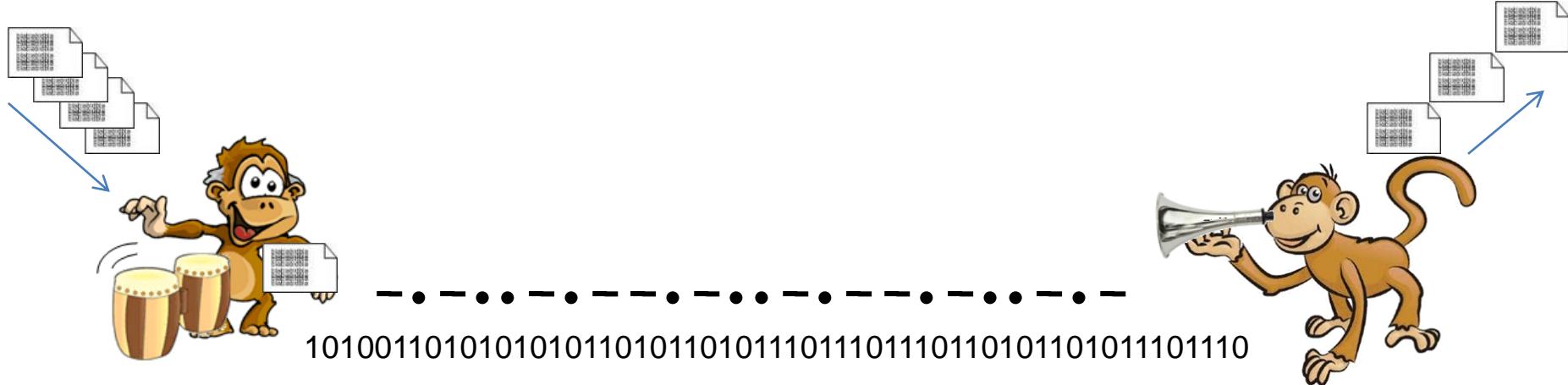
- Spews out signals at a furious rate

Framing



- Spews out signals at a furious rate
- How does receiving-monkey know when a unit of information begins?

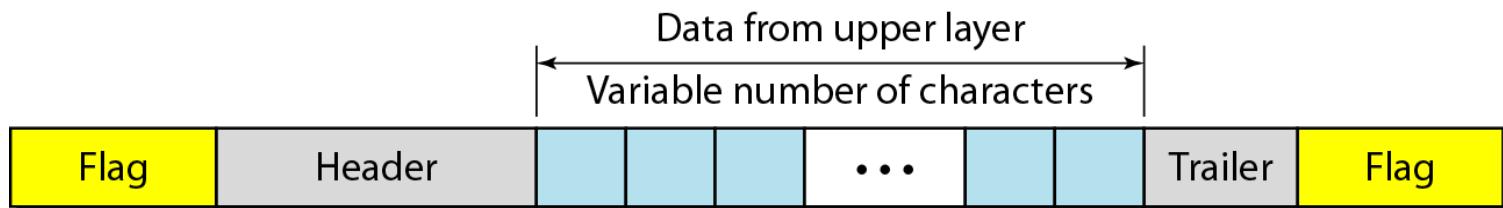
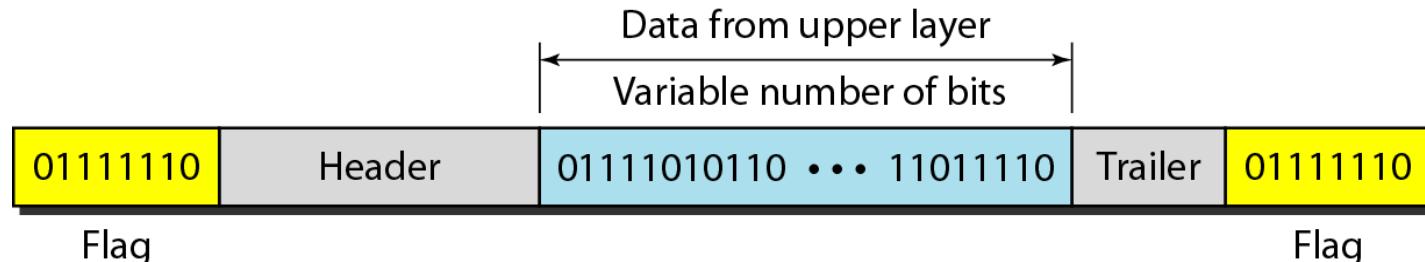
Framing



* Figure is courtesy of B. Forouzan

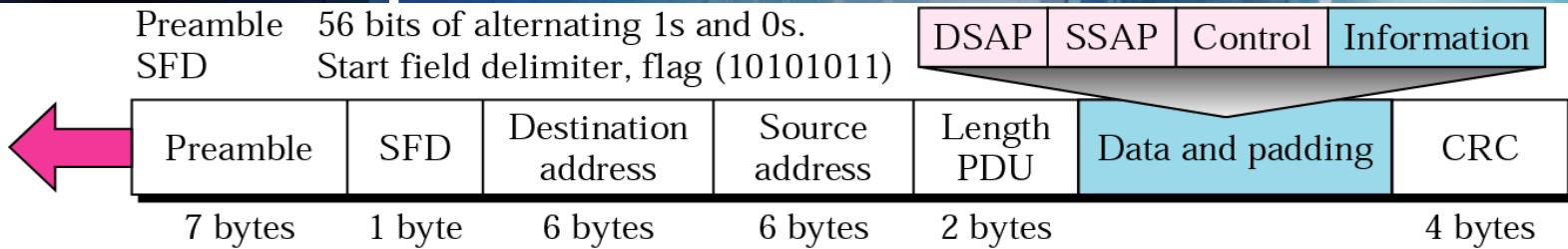
Bit- & Byte-Oriented Protocols

- Two Variations



* Figure is courtesy of B. Forouzan

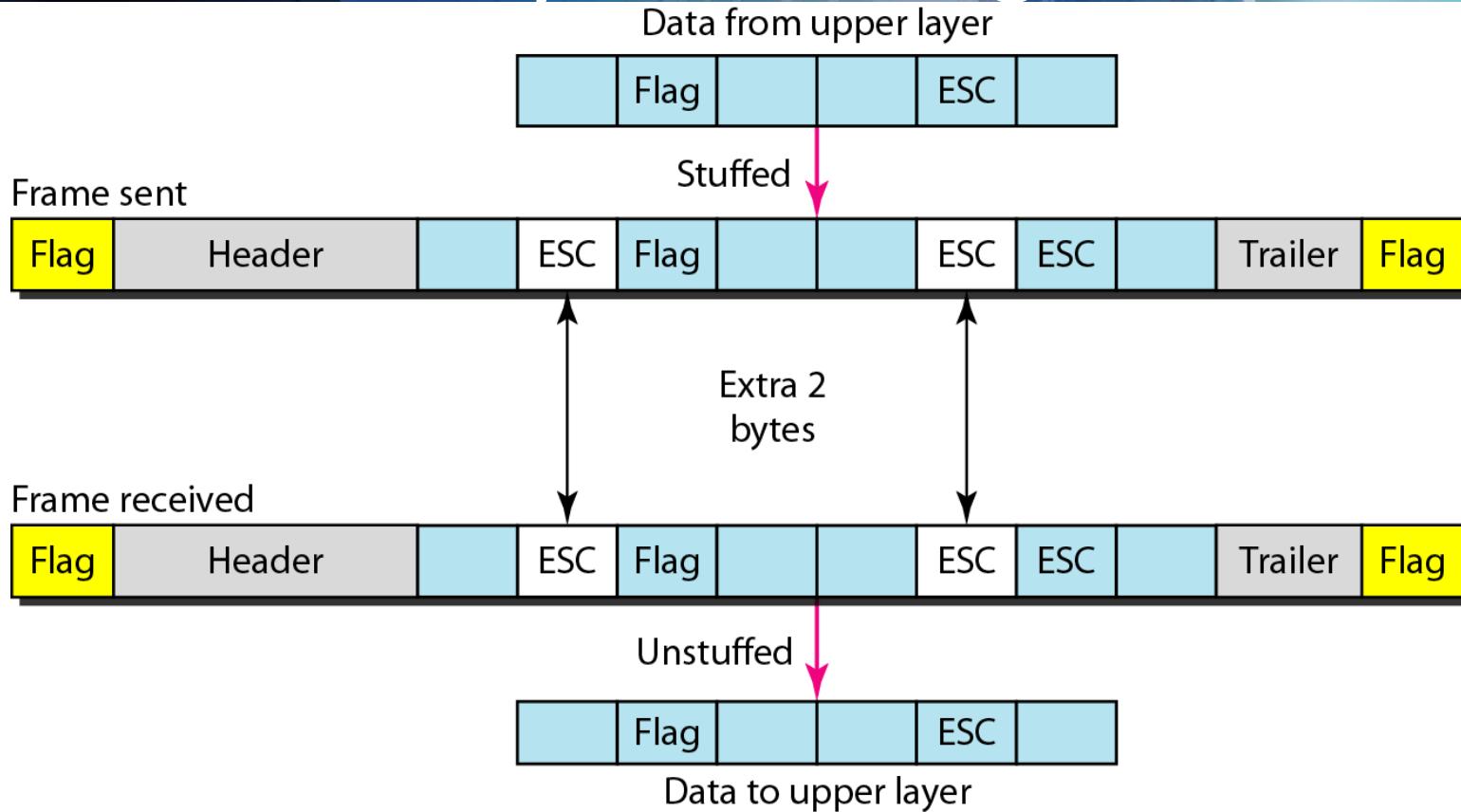
Example: 802.3 MAC Format



- 64-bit frame preamble (10101010) used to synchronize reception
 - 7 bit preamble (10101010) + 1 start flag (10101011)
- Maximum frame length: 1518 bytes
 - ⇒ max 1500 bytes payload
- Minimum frame length: 64 bytes
 - ⇒ min 46 bytes payload

* Figure is courtesy of B. Forouzan

Byte Stuffing

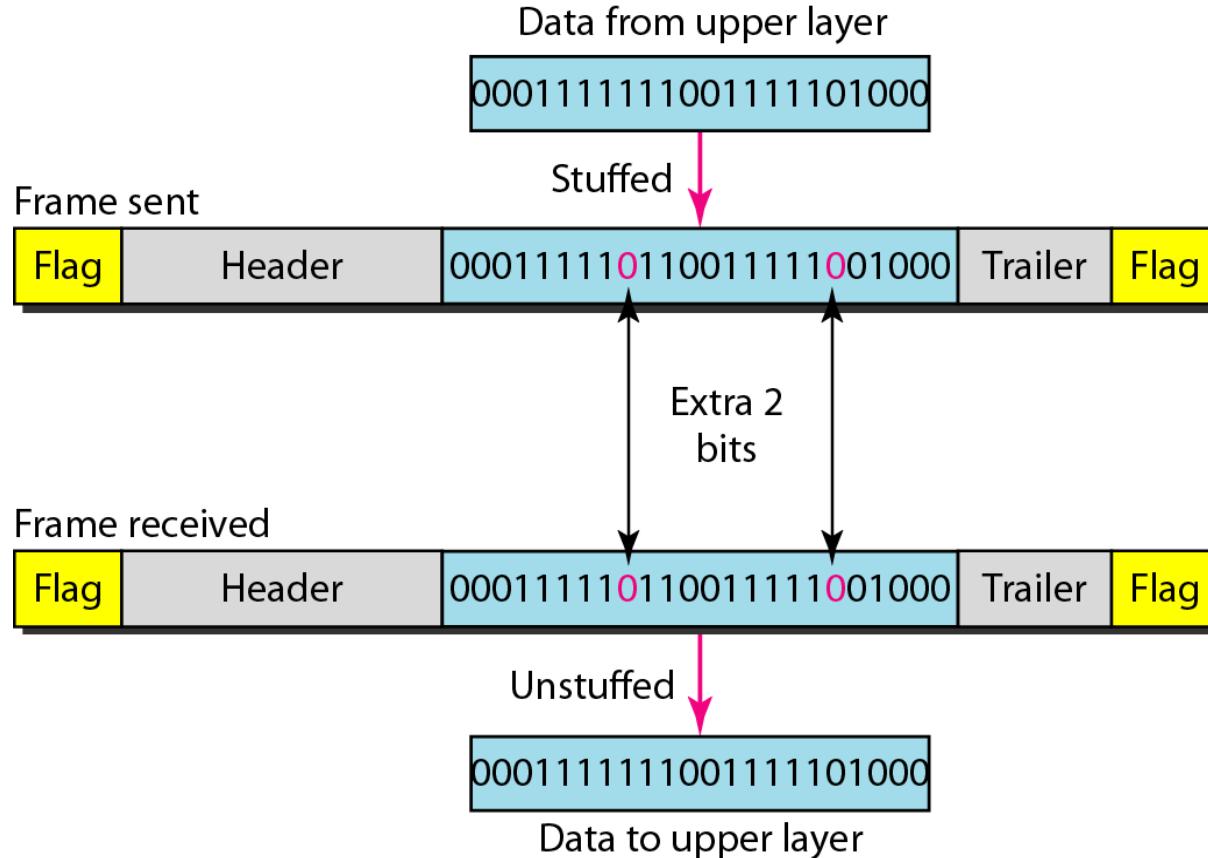


Process of adding 1 extra byte whenever
there is a flag or escape character in the text.

* Figure is courtesy of B. Forouzan



Bit Stuffing

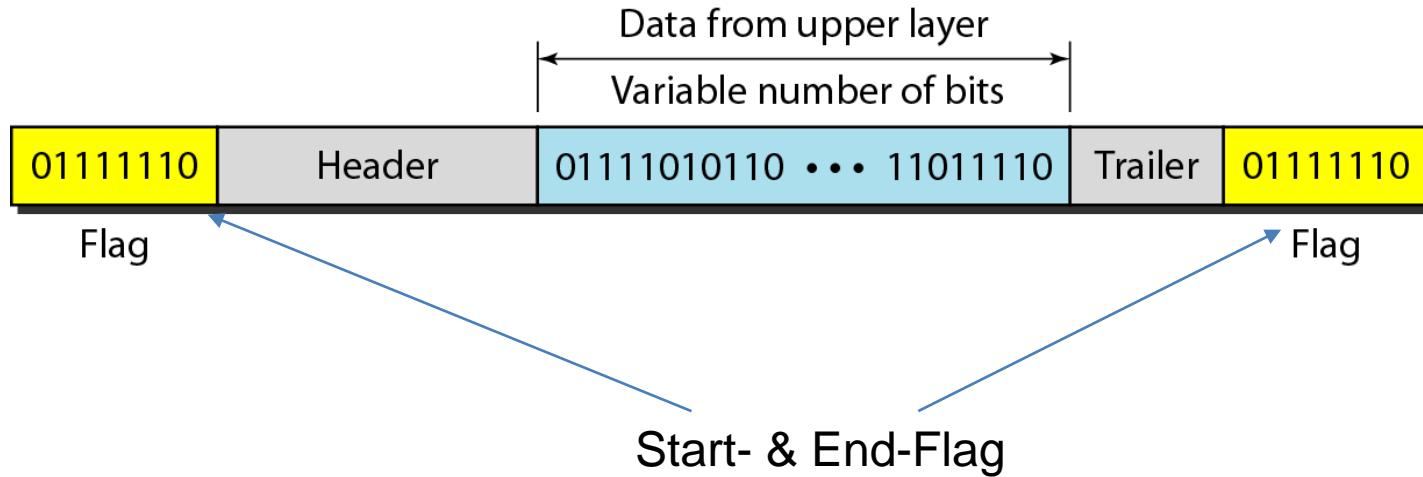


Process of adding an extra 0 whenever five consecutive 1s follow a 0 in the data

* Figure is courtesy of B. Forouzan



Framing



* Figure is courtesy of B. Forouzan



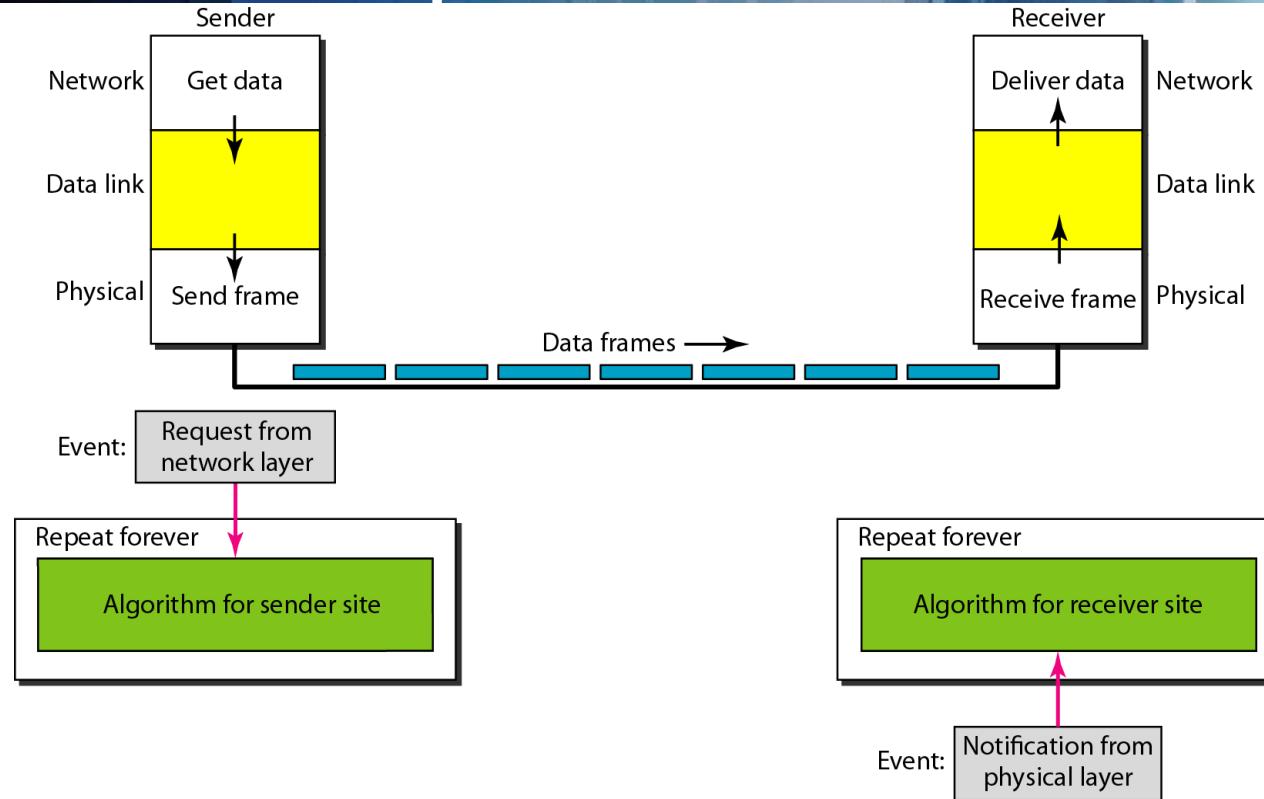
Networking Issues

- Time → Latency
- Amount → Throughput



- Management Information → Overhead
 - May lead to better efficiency
- Overhead vs Payload

Simplest Protocol

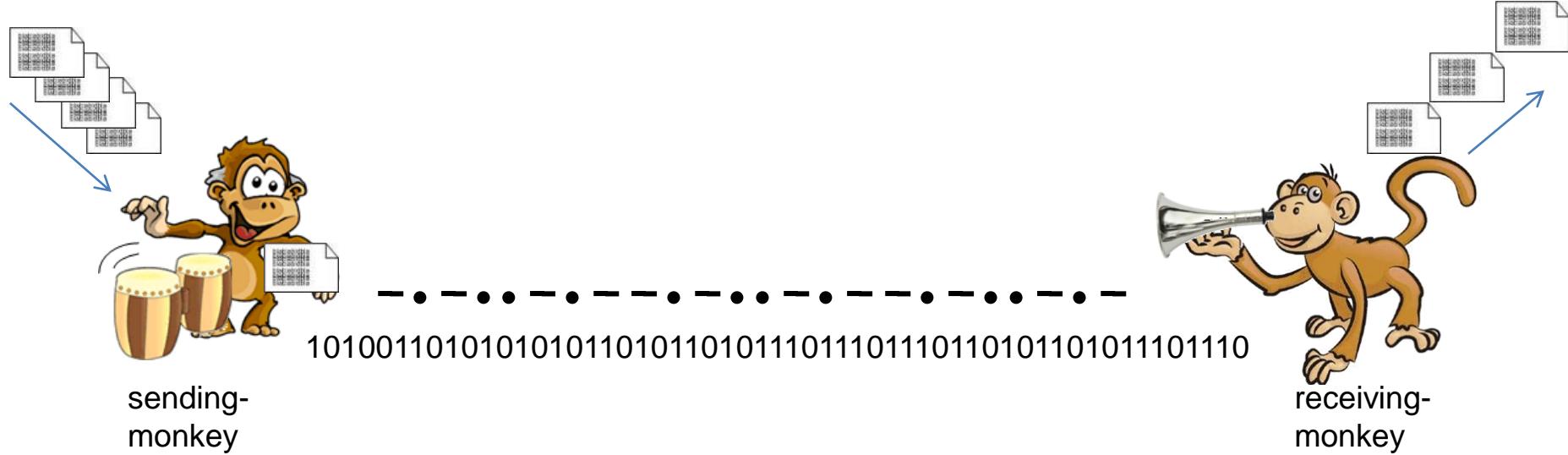


- Hope that the receiver is fast enough!
- No overhead

* Figure is courtesy of B. Forouzan



Flow Control

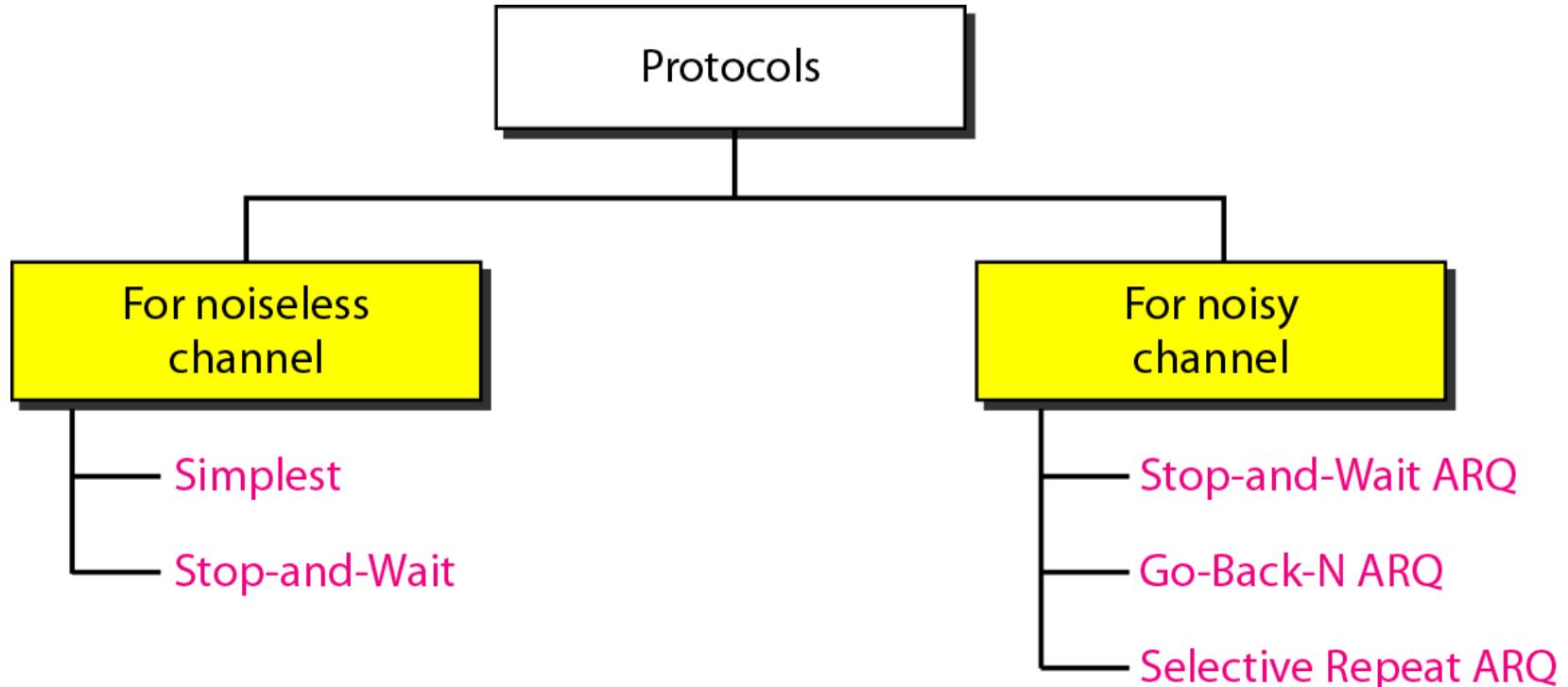


- What happens if sending-monkey can drum faster than receiving-monkey can write?

Flow Control

- Forouzan's Definition: Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.
- “My” Definition: Flow Control refers to the control of the amount of data that a sender can transmit without overflowing the receiver.

Flow Control Protocols

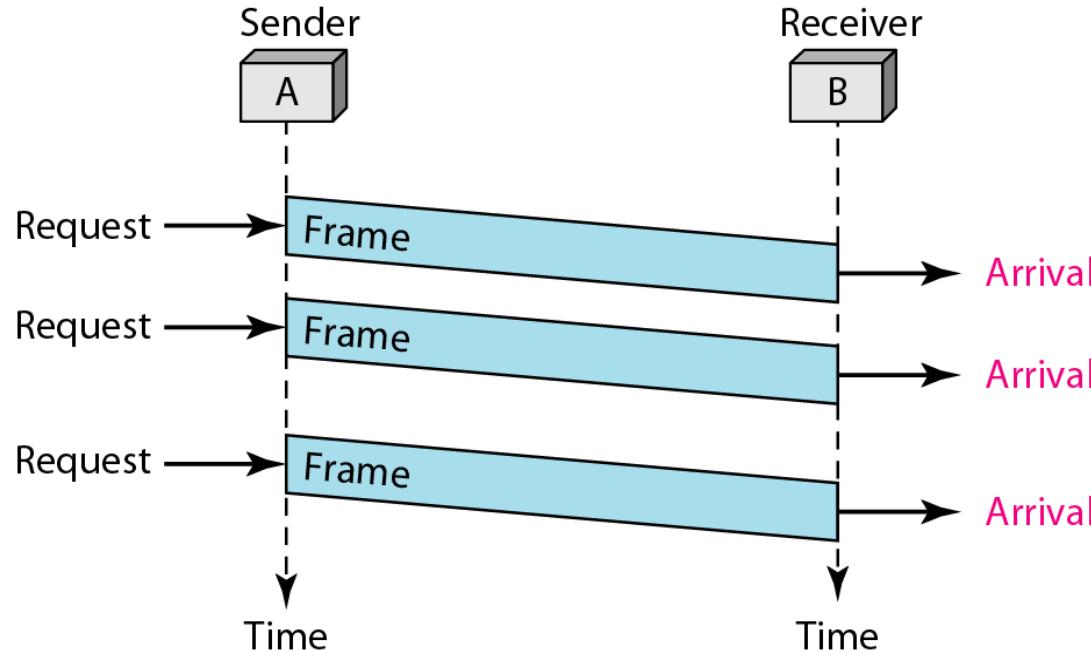


* Figure is courtesy of B. Forouzan



Simplest Protocol: Flow Diagram

- Sender sends frames as fast as data arrives
- Receiver receives all data sent

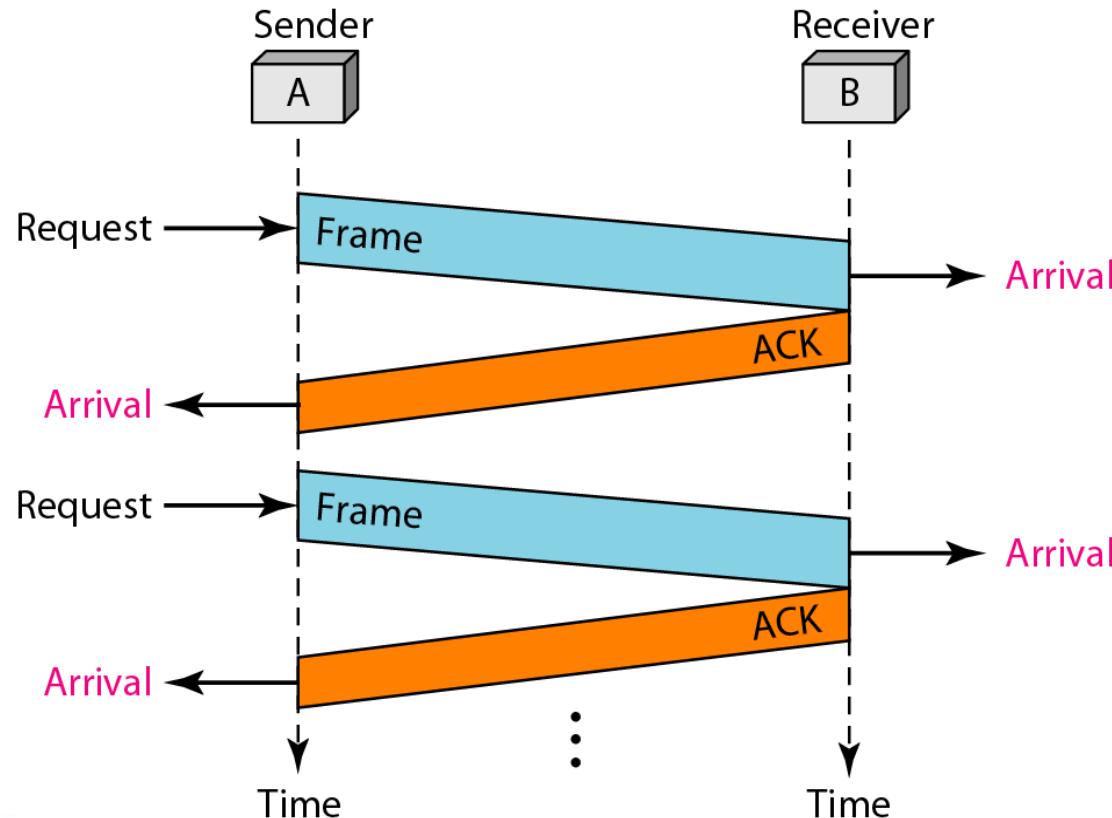


* Figure is courtesy of B. Forouzan



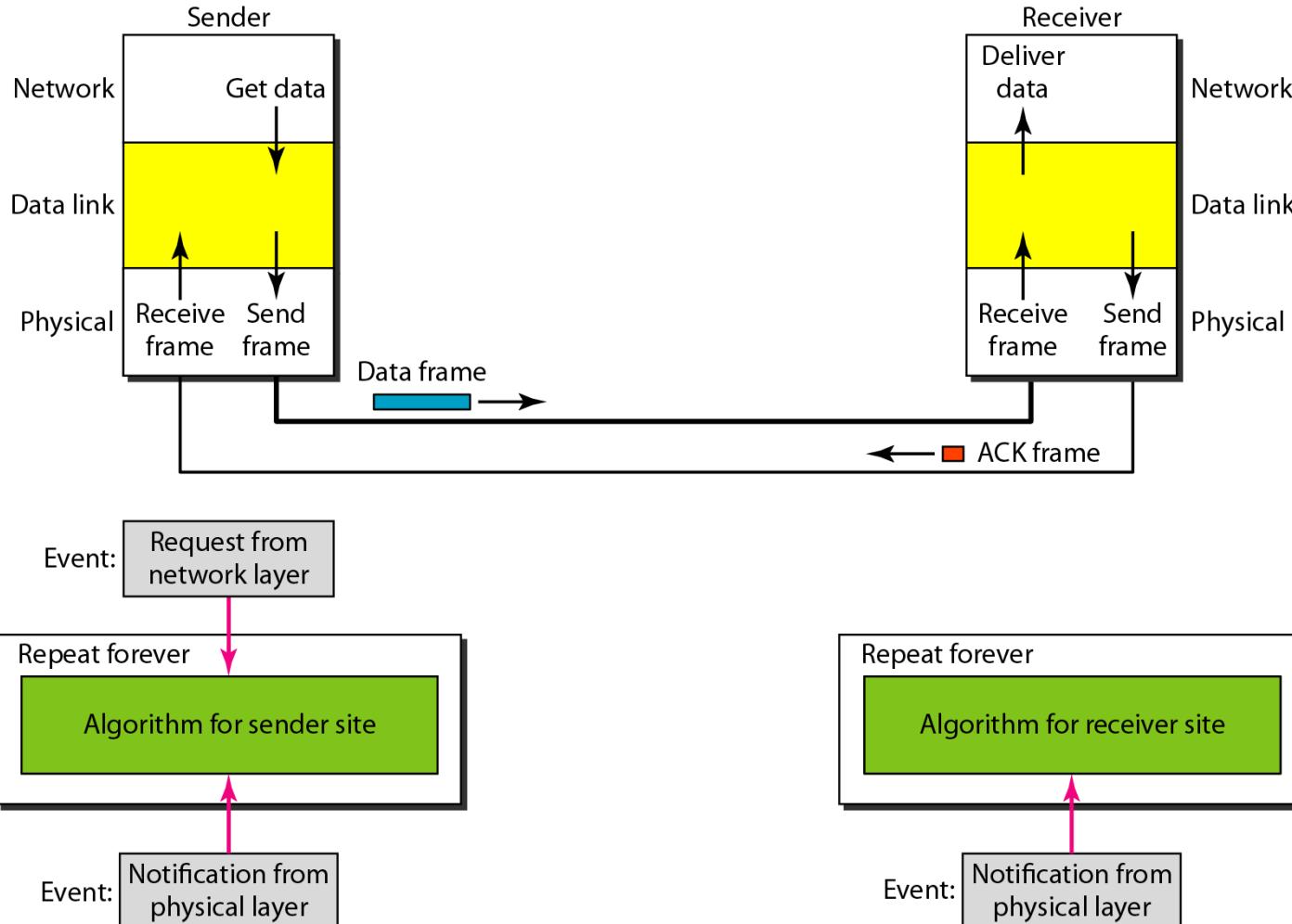
Stop-and-Wait: Flow Diagram

- Sender sends frame and waits for ACK
- Receiver replies to received frame with ACK



* Figure is courtesy of B. Forouzan

Stop-and-Wait Protocol



* Figure is courtesy of B. Forouzan

Error Control

- Frames may get lost or corrupted
 - Incorrect checksum, CRCs, etc
- Error control need to ensure retransmission
- Error Control Protocols:
 - Stop-and-Wait ARQ*
 - Go-back-N ARQ
 - Selective Repeat ARQ

*ARQ = Automatic Repeat Request

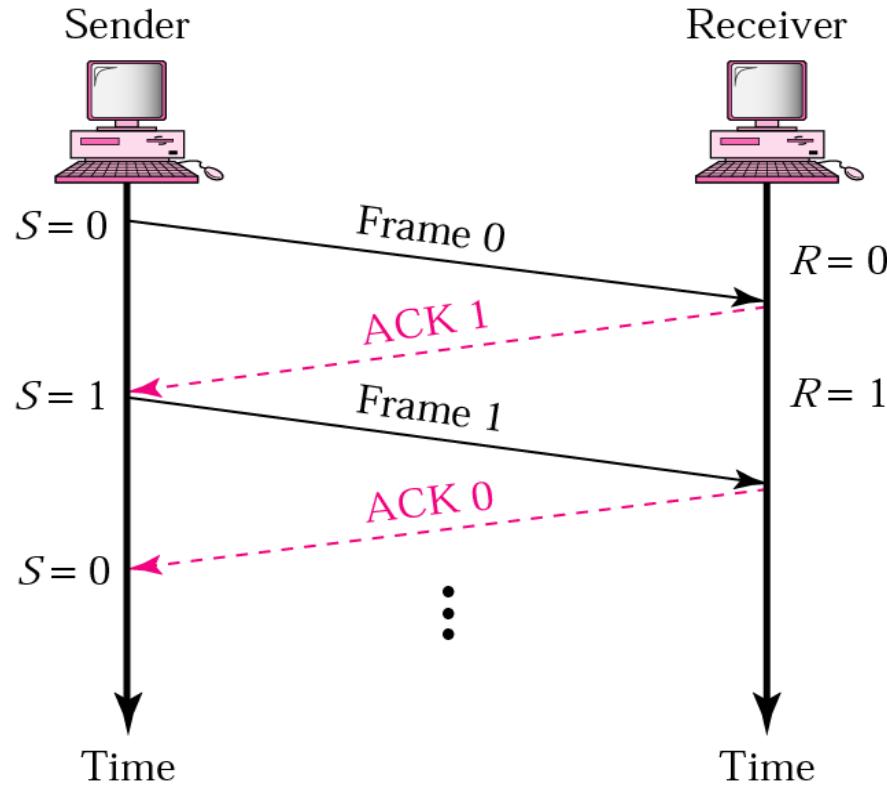


Ingredients for Error Control

- Error detection
- Positive acknowledgement
 - Receiver returns positive ACK for received, error-free frames
- Retransmission after timeout
 - Sender retransmit packet after given time
- Negative acknowledgement and retransmission
 - Receiver returns negative ACK - or NACK - for packets with errors



Stop-and-Wait ARQ

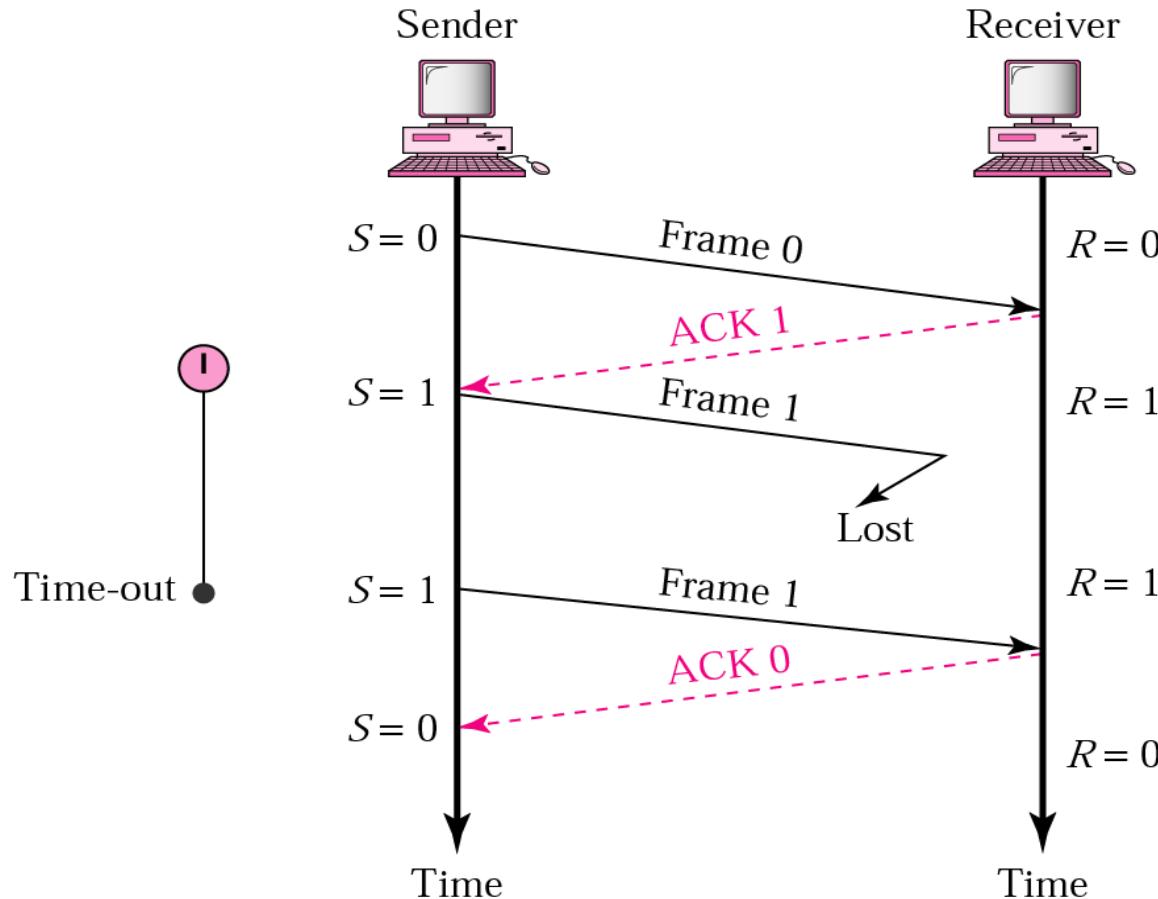


- ACK = received packet, ready to receive packet #
- ARQ = Automatic Repeat Request

* Figure is courtesy of B. Forouzan

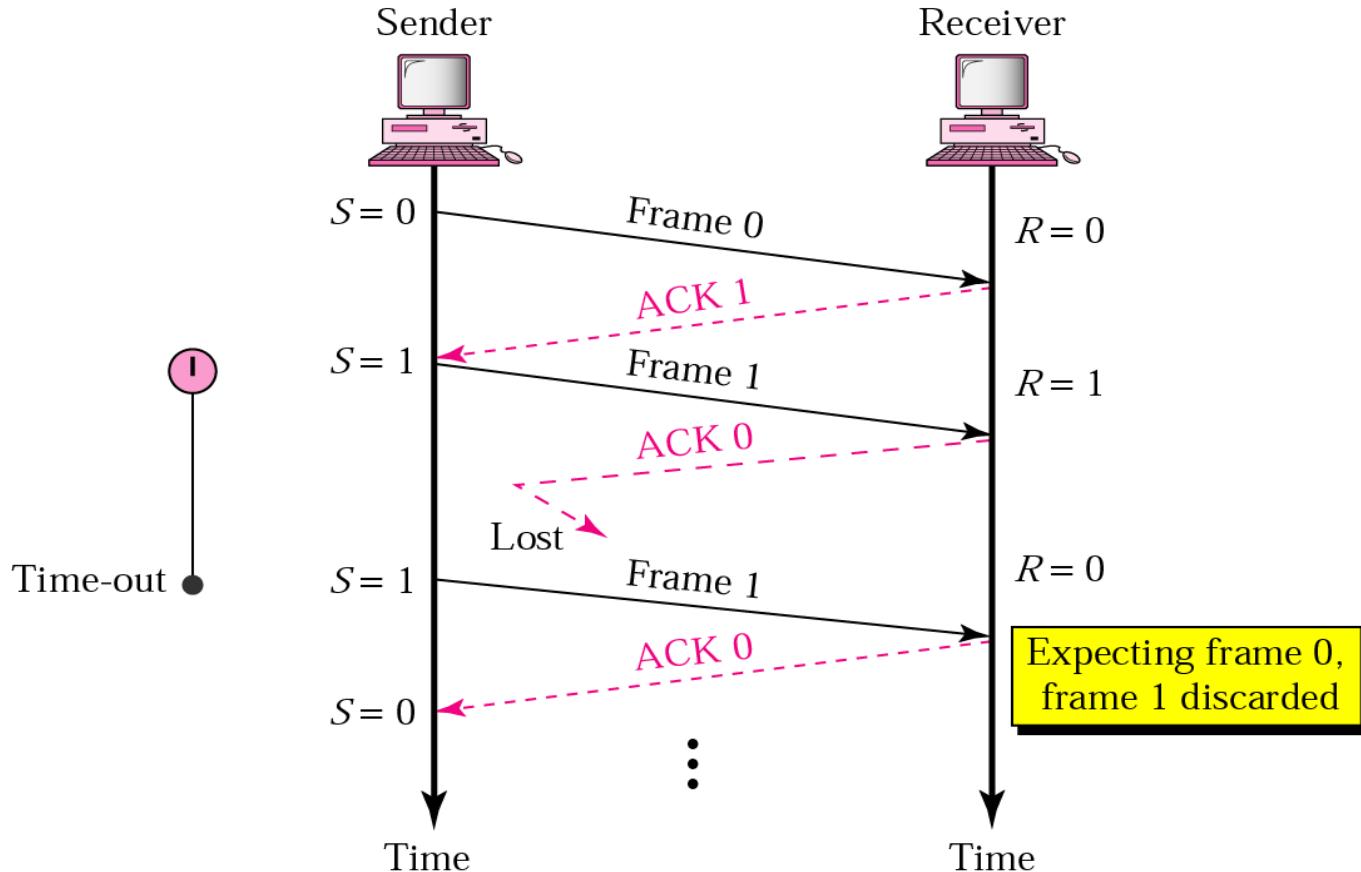
Stop-and-Wait ARQ: Time-Out

- Frame is lost during transmission



* Figure is courtesy of B. Forouzan

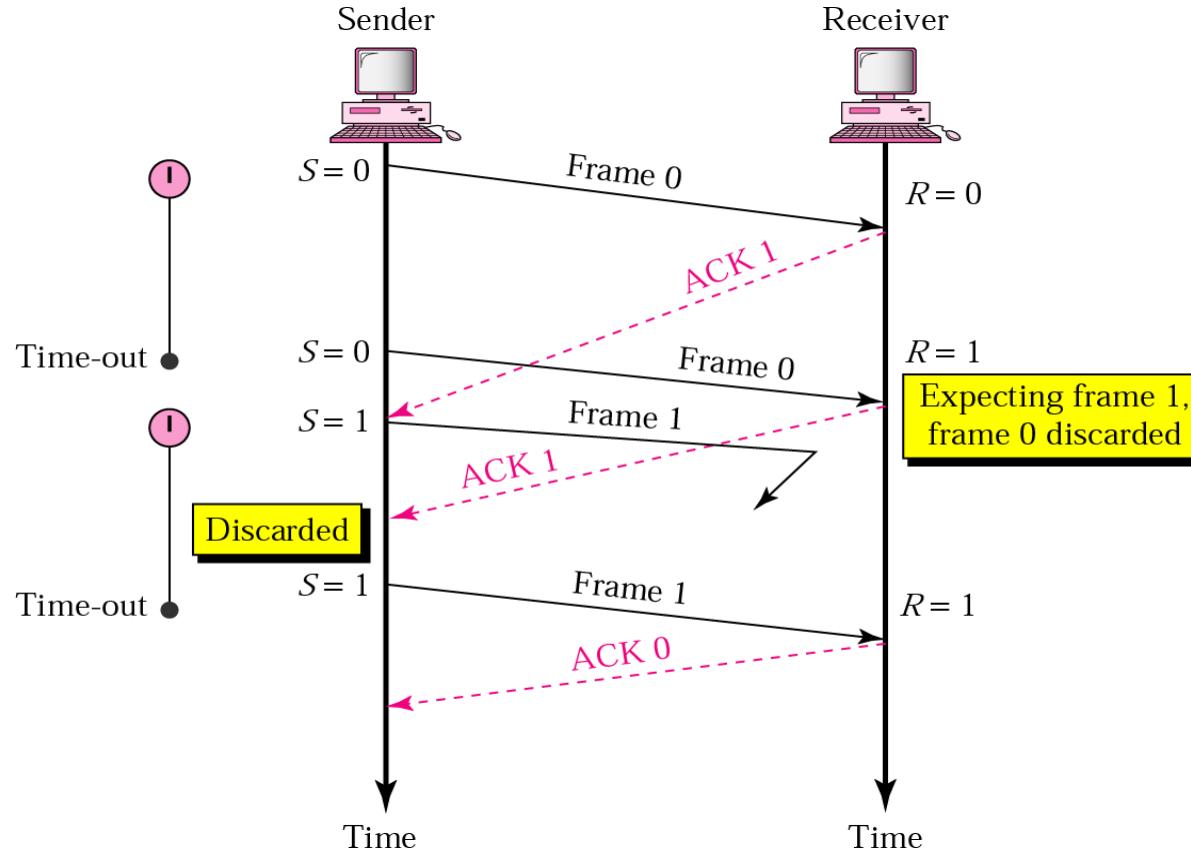
Stop-and-Wait ARQ: Lost-ACKs



- Numbering frames prevents retaining duplicate frames
- Every received frame is acknowledged

* Figure is courtesy of B. Forouzan

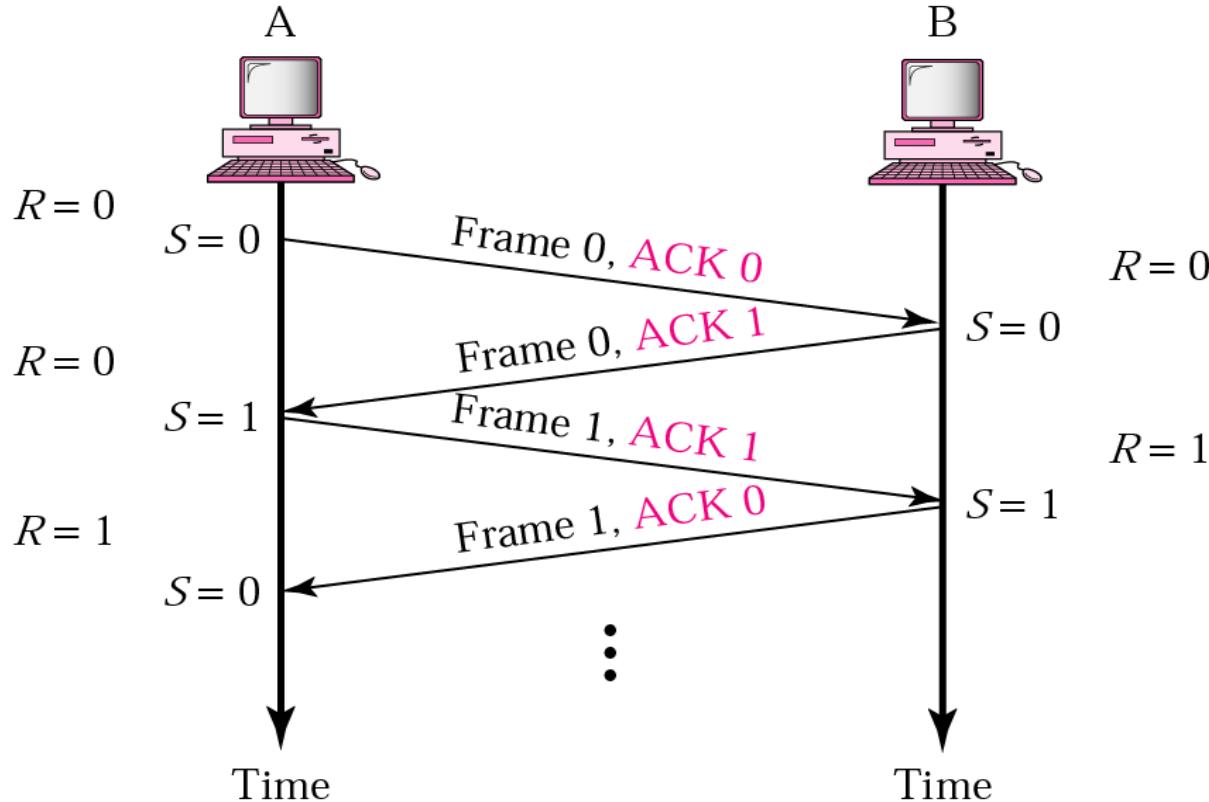
Stop-and-Wait ARQ: Delayed ACK



Numbered acknowledgments are needed if an acknowledgment is delayed and the next frame is lost.

* Figure is courtesy of B. Forouzan

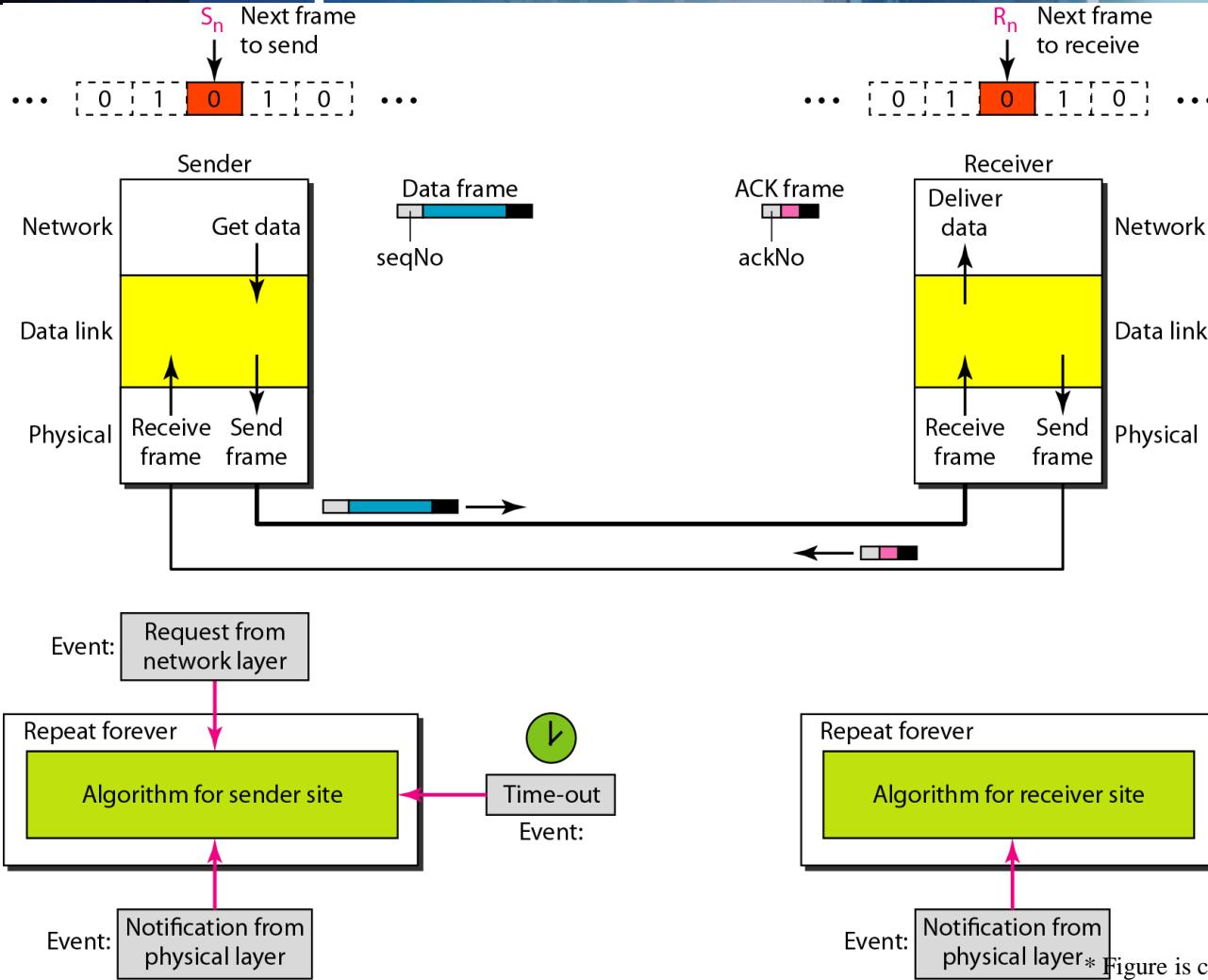
Piggybacking ACKs



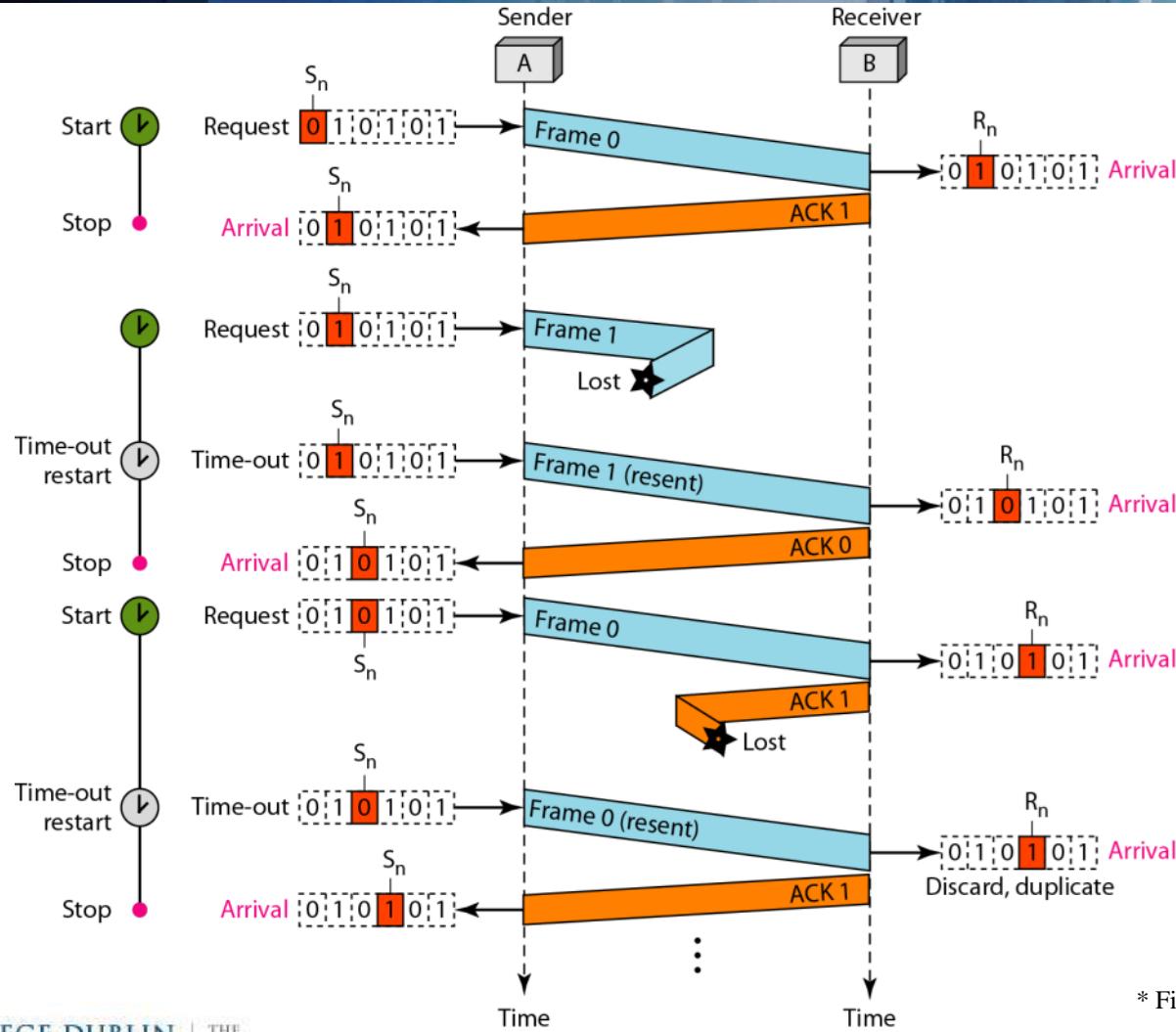
Next data frame send carries
the acknowledgement for the last frame received

* Figure is courtesy of B. Forouzan

Stop-and-Wait ARQ

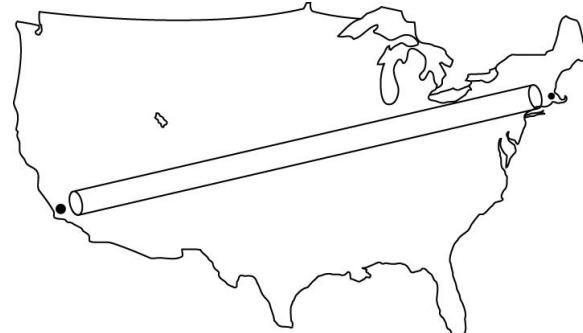


Stop-and-Wait ARQ: Flow Diagram

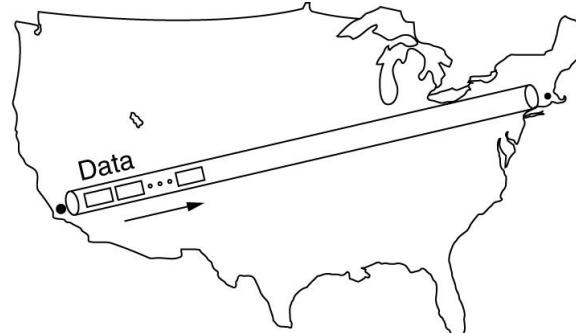


* Figure is courtesy of B. Forouzan

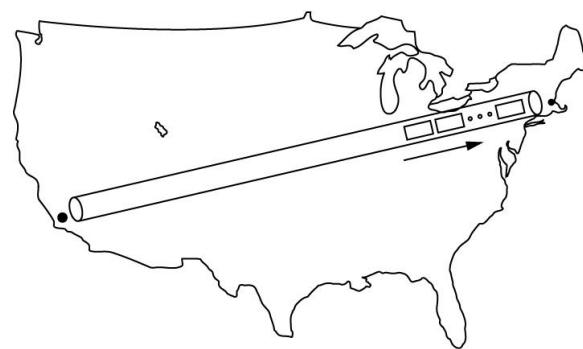
Round Trip Time



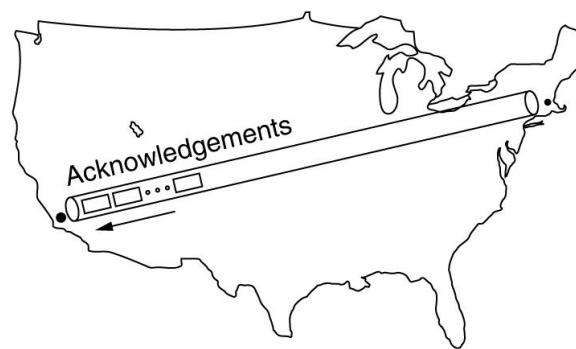
(a)



(b)



(c)



(d)

(a) At $t = 0$ (b) After 500 μ sec (c) After 20 msec (d) after 40 msec

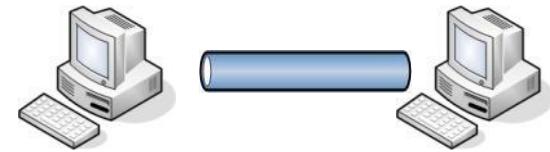
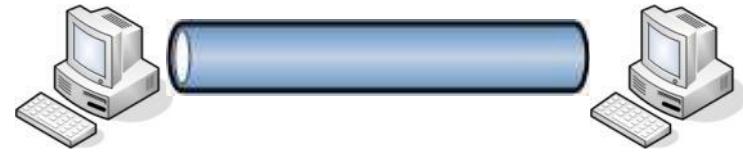
Flow Control

- Definitions
 - Transmission time
 - Time taken to emit all bits onto the medium
 - Proportional to length of frame
 - Propagation time
 - Time for a bit to traverse the link

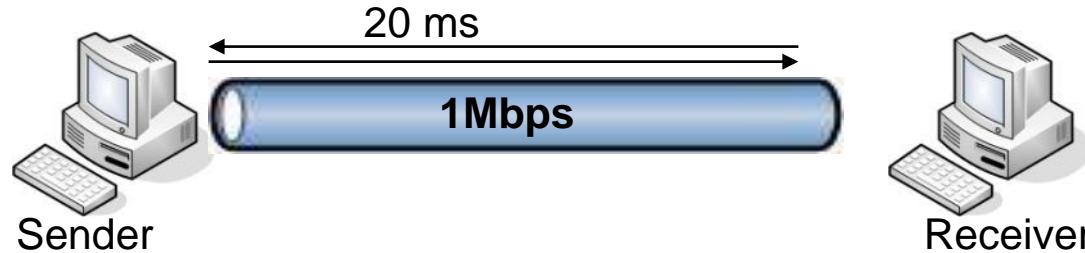


Bandwidth-Delay Product

- Bandwidth:
 - Size of the pipe
 - Determines how much data can be send
- Round-Trip Time (RTT)
 - Determines how long an ACK takes
- High Bandwidth (big pipe)
 - Lots of data can be send
- Depending on RTT
 - Sender may exhaust window quickly
- $\text{Bandwidth} \times \text{RTT}$
 - Gives indication of amount of data that can be send while waiting for ACK



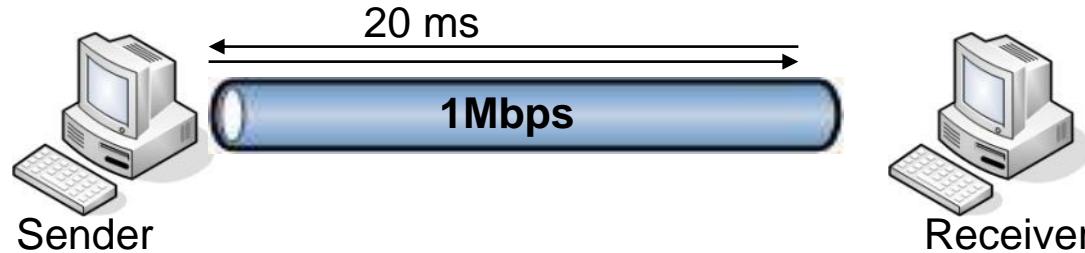
Delay Before Receiving ACK



- Communication link with 1Mb/s
- Round-Trip time: $20 \text{ ms} = 20 * 10^{-3} \text{ s}$
- How much data can you send during the time it takes for 1 bit e.g. an ACK to arrive at the sender:

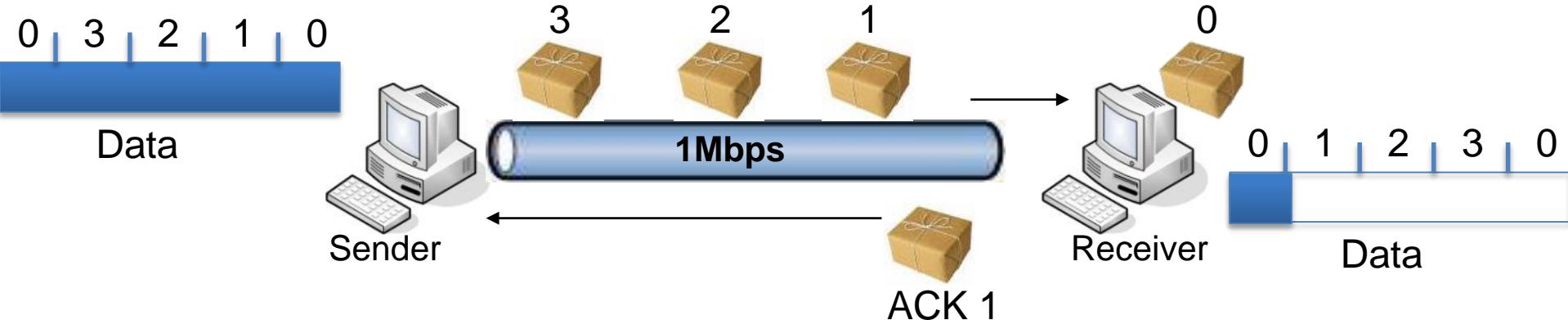
$$20 * 10^{-3} \text{ s} * 1 * 10^6 \text{ b/s} = 20.000 \text{ bits}$$

Delay Before Receiving ACK



- Communication link with 1Mb/s
- Round-Trip time: $20 \text{ ms} = 20 * 10^{-3} \text{ s}$
- How much data can you send during the time it takes for 1 bit e.g. an ACK to arrive at the sender:
$$20 * 10^{-3} \text{ s} * 1 * 10^6 \text{ b/s} = 20.000 \text{ bits}$$
- Frame of 2000 bit \Rightarrow 10% of bandwidth used

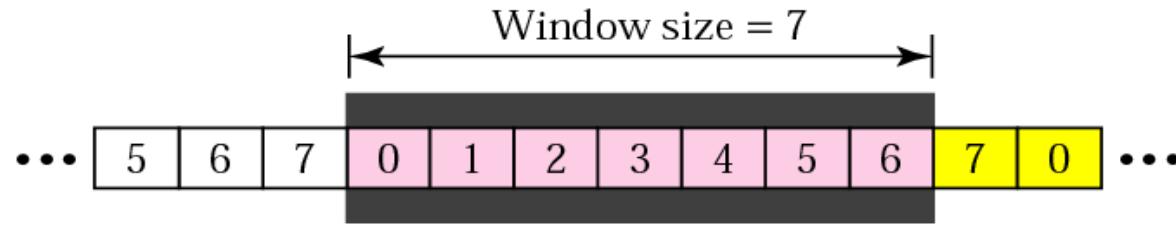
Ideal Solution to Filling the Pipe



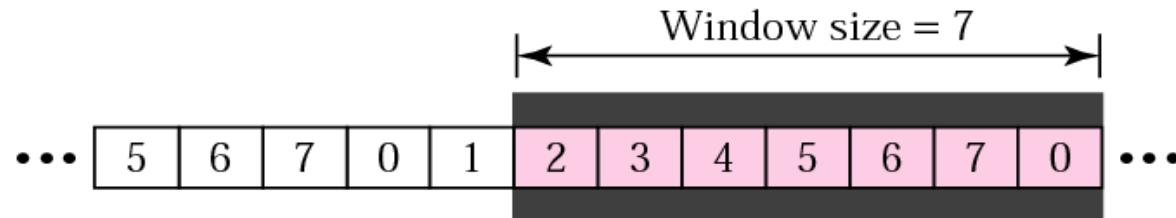
- Allow multiple frames to be in transit
- Receiver has a buffer
- Transmitter can send a number of frames
 - without receiving an ACK
- Each frame is numbered
- ACK includes number of next frame expected

Sliding Window

- m : Size of the sequence number field in bits
- $1 \dots 2^m$: Sequence numbers
- Send window: Box of size $2^m - 1$



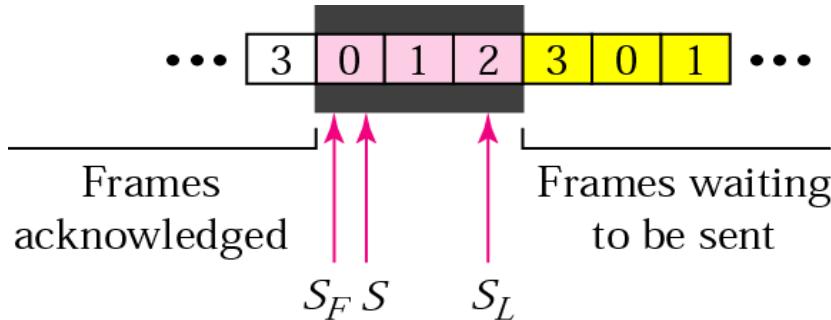
a. Before sliding



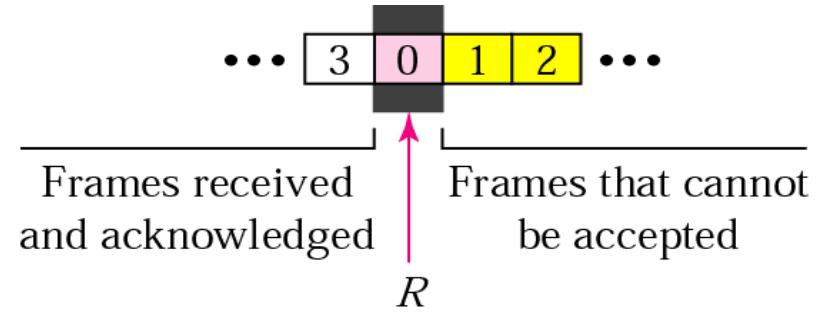
b. After sliding two frames

* Figure is courtesy of B. Forouzan

Go-Back-N ARQ: Control variables



a. Sender window



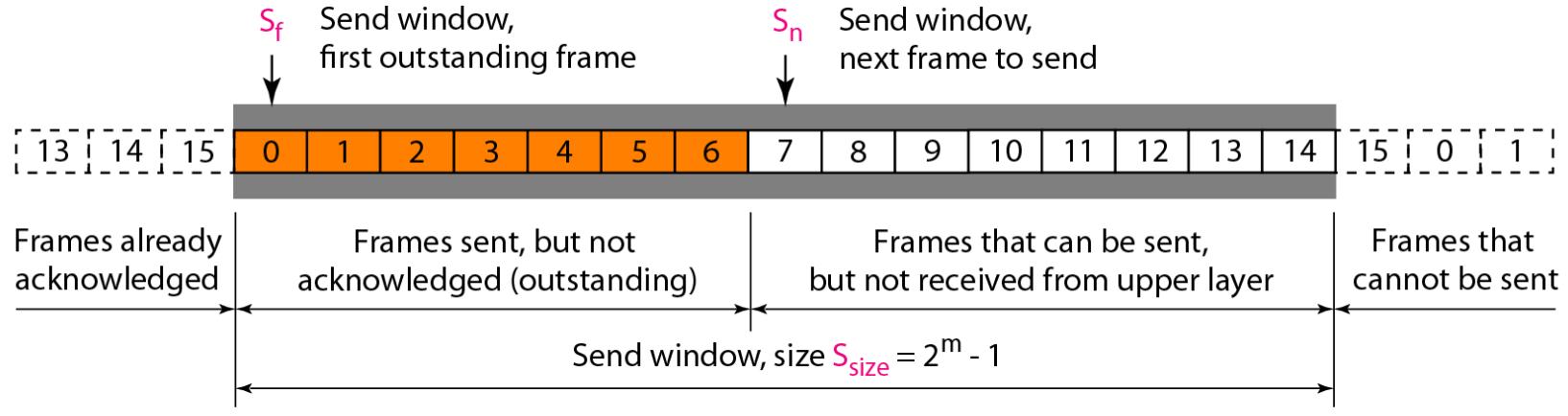
b. Receiver window

- $S = \#$ of recently send frame
- $S_F = \#$ of first send frame of window
- $S_L = \#$ of last send frame of window

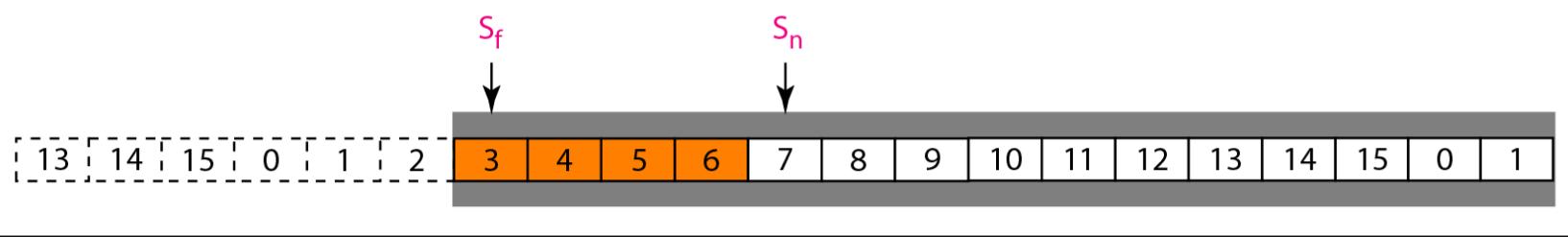
- $R = \#$ of recently received frame

* Figure is courtesy of B. Forouzan

Sliding Window



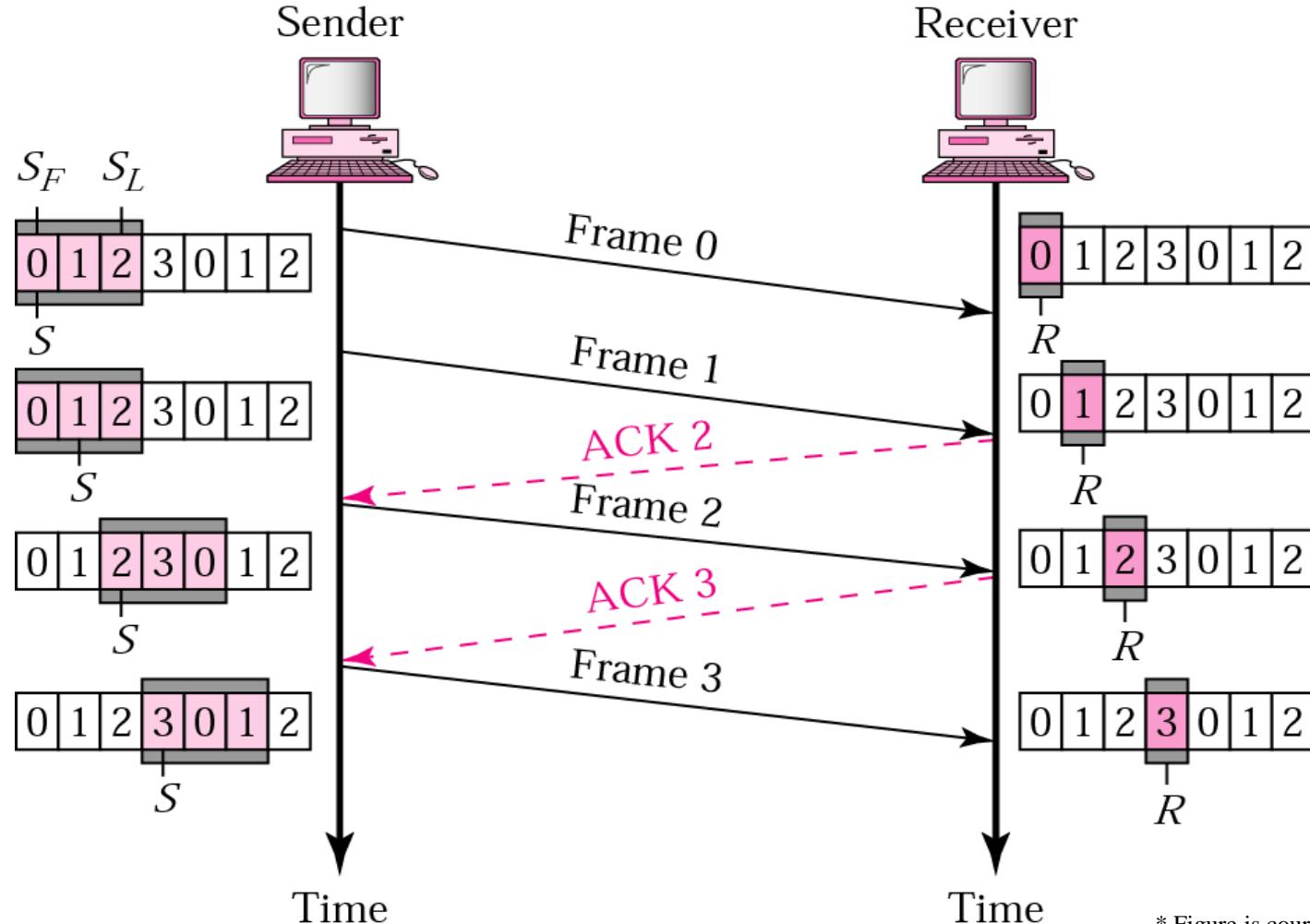
a. Send window before sliding



b. Send window after sliding

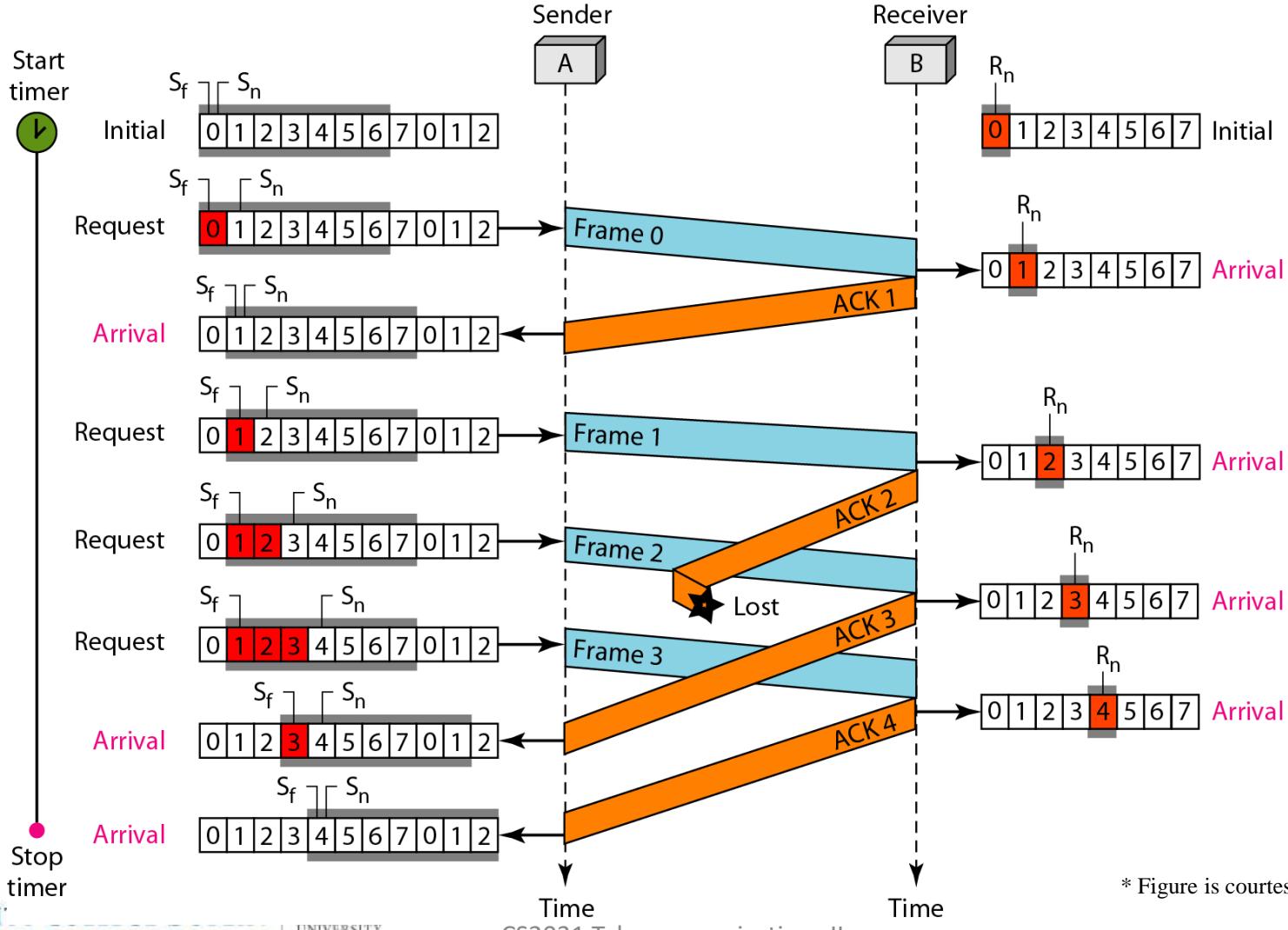
* Figure is courtesy of B. Forouzan

Go-Back-N ARQ



* Figure is courtesy of B. Forouzan

Go-Back-N: Lost ACK

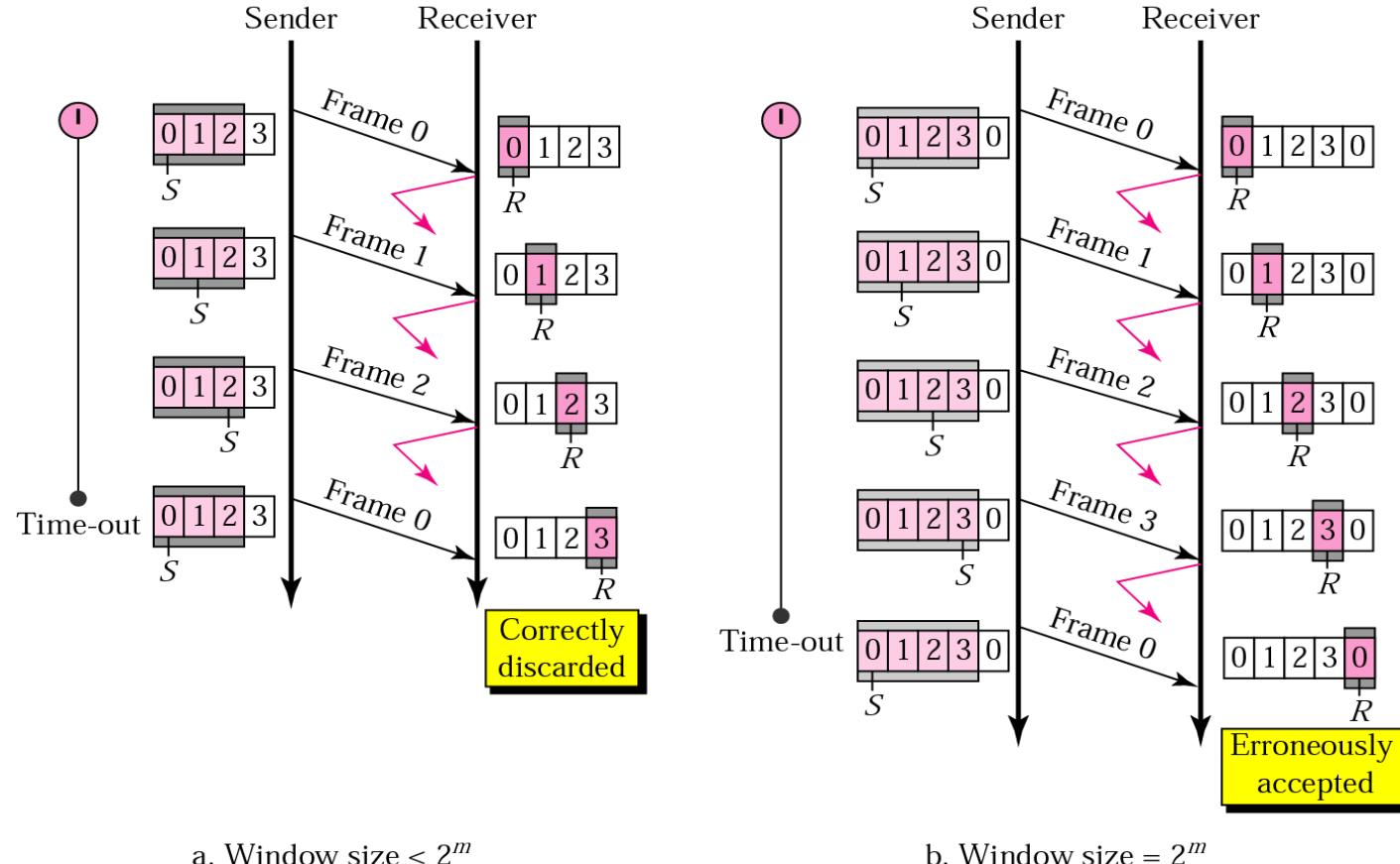


* Figure is courtesy of B. Forouzan

Window Size for Go-Back-N

- Depends on size of max. frame number
 - Frame # needs to be included in every frame
 - e.g. 4 bits – $2^4 = 16$ frame numbers
- Trade-off between window size and frame size

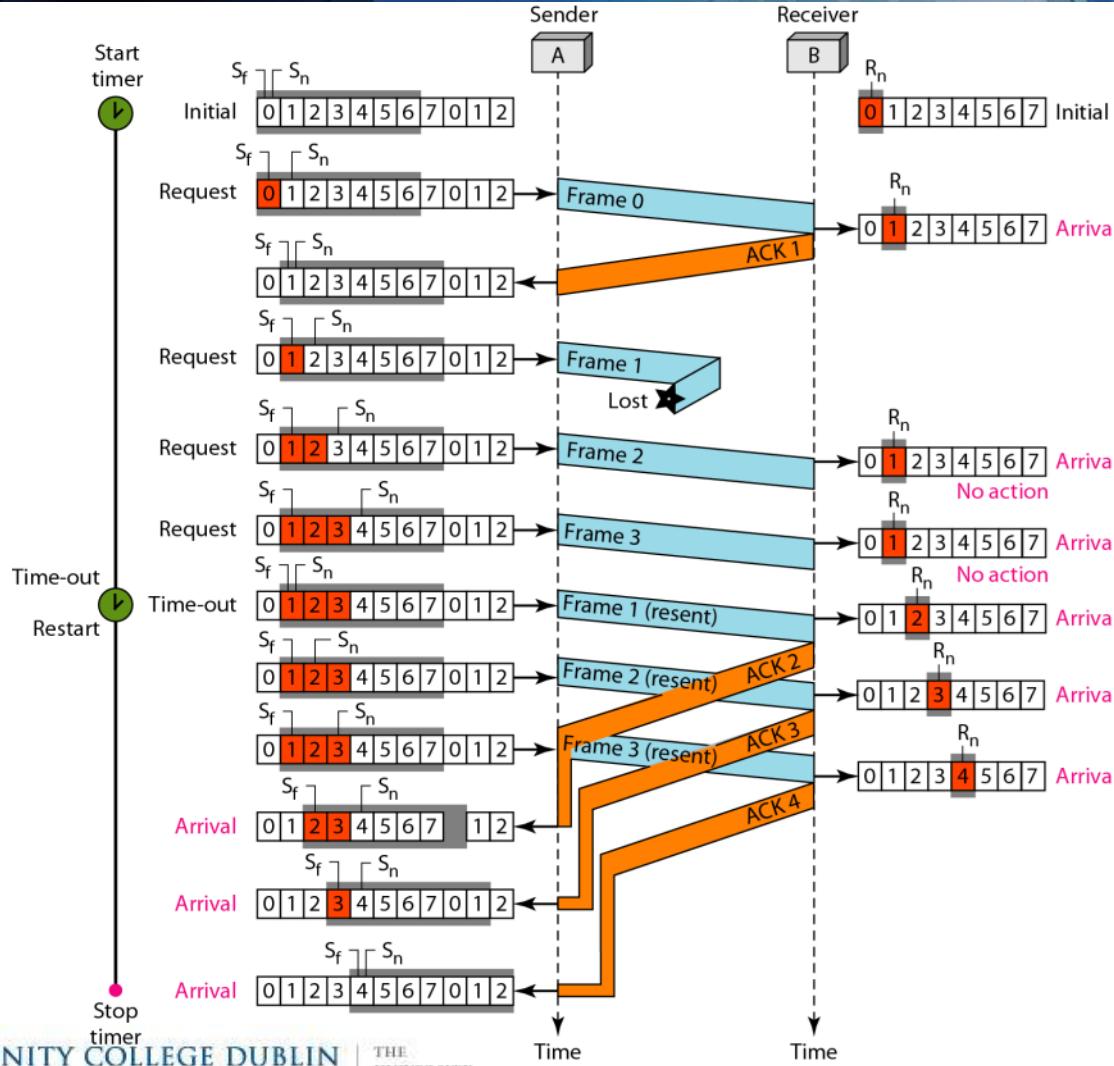
Go-Back-N: Limitation of window size



$m = \# \text{ of bits for index}$
 Size of the sender window must be less than 2^m

* Figure is courtesy of B. Forouzan

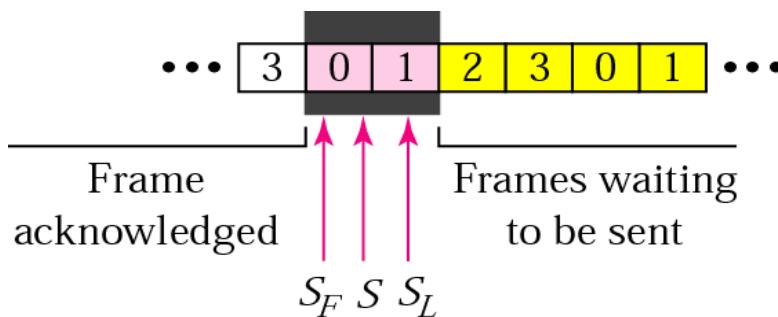
Go-Back-N ARQ: Bad Behaviour



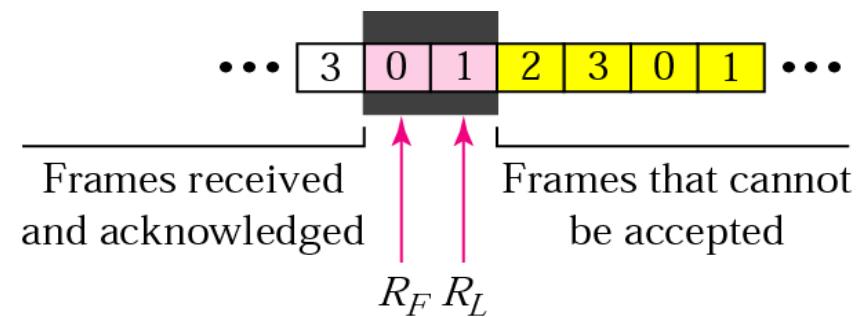
* Figure is courtesy of B. Forouzan

Selective Repeat

- Two Windows:
 - 1 Sender Window – 1 Receiver Window



a. Sender window



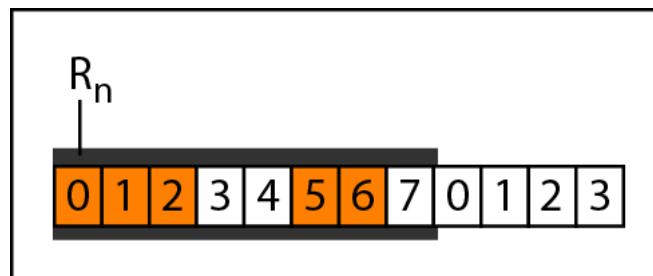
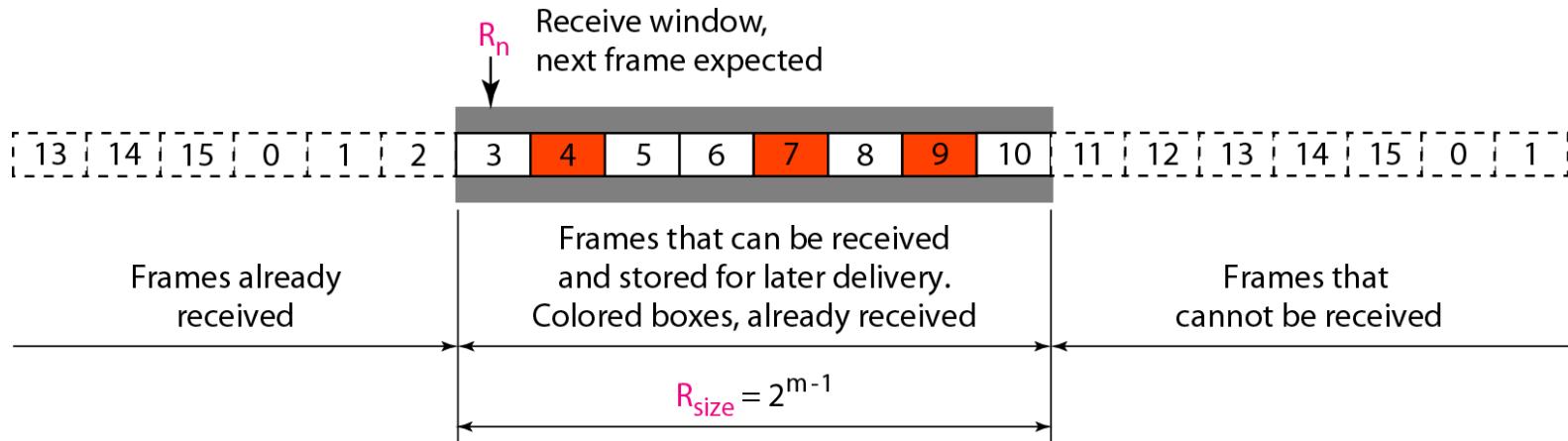
b. Receiver window

* Figure is courtesy of B. Forouzan

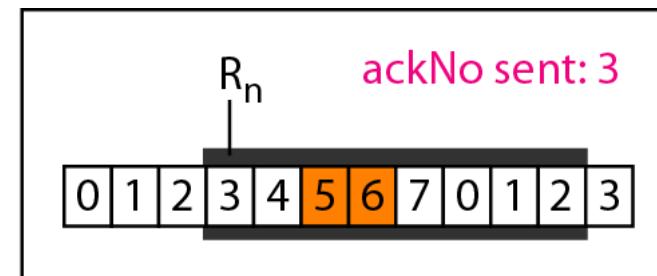


Selective Repeat ARQ

- Window records received frames:



a. Before delivery

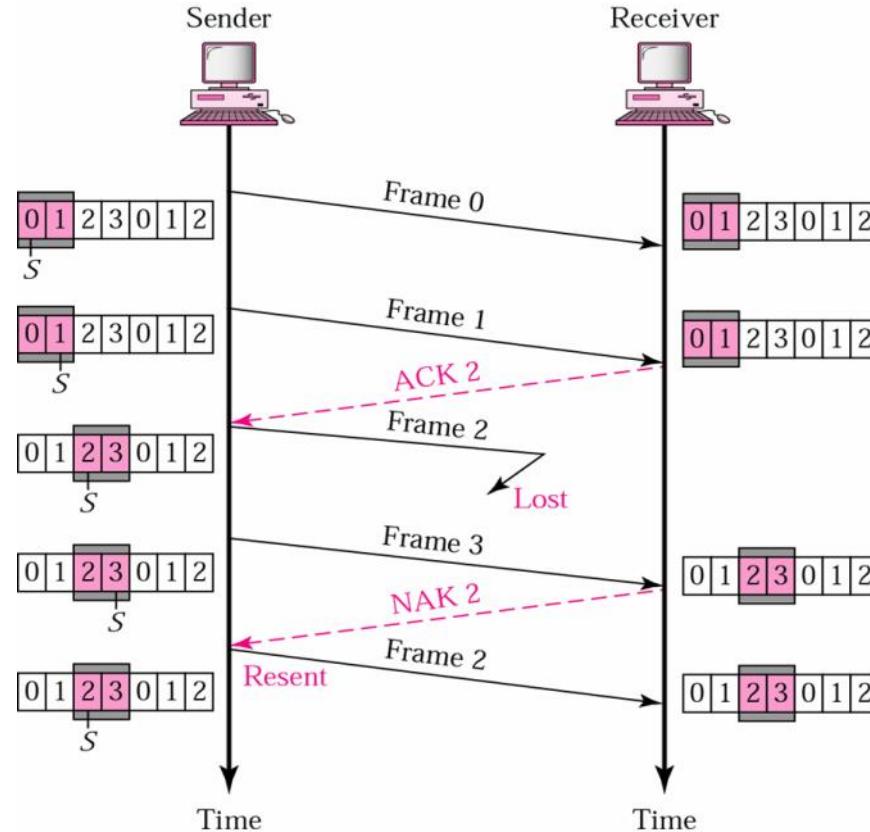


b. After delivery

* Figure is courtesy of B. Forouzan



Selective Repeat ARQ: Lost Frame

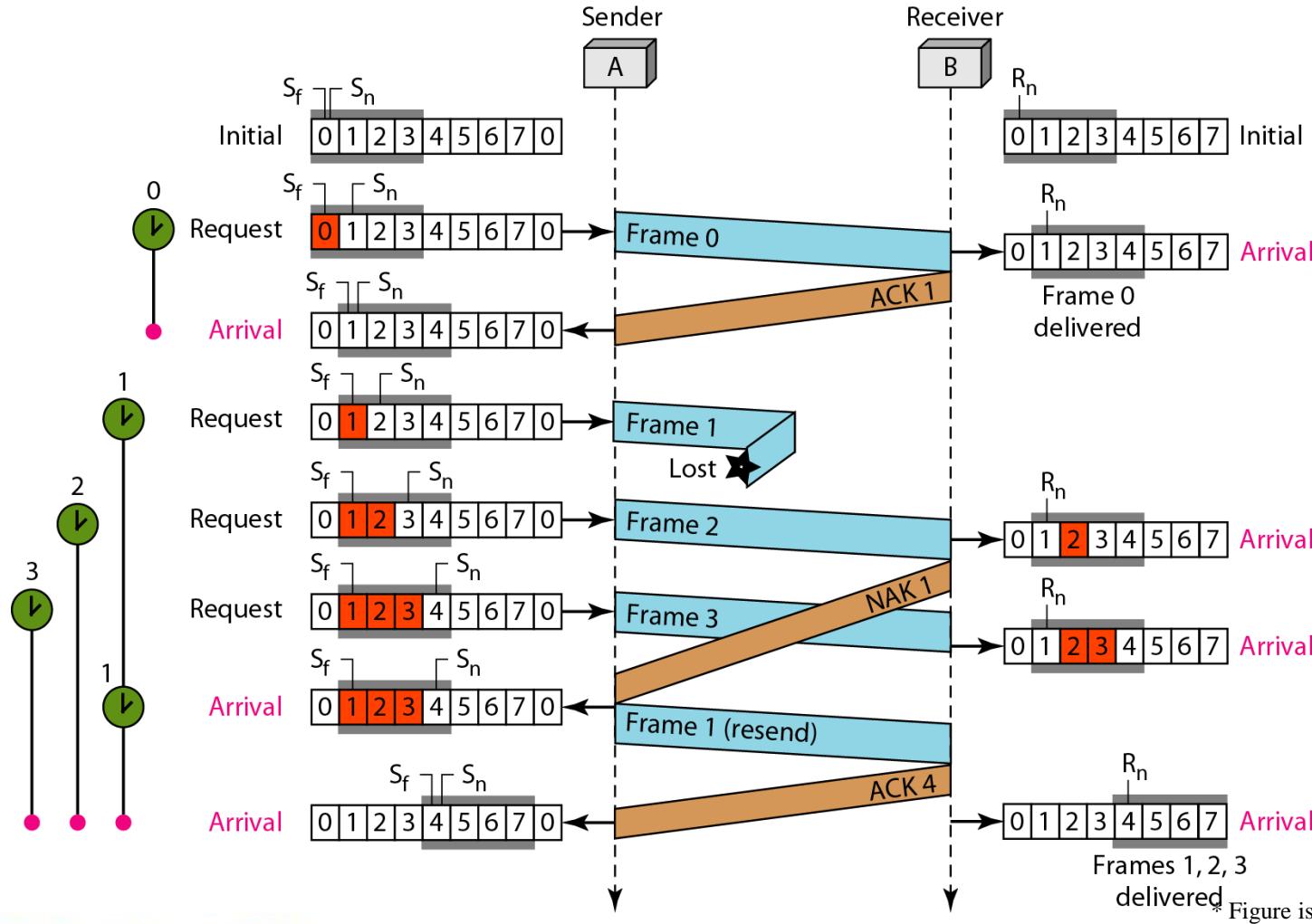


- NAK = Negative Acknowledgement
- Sender still maintains timers for packets in case NAK gets lost

* Figure is courtesy of B. Forouzan

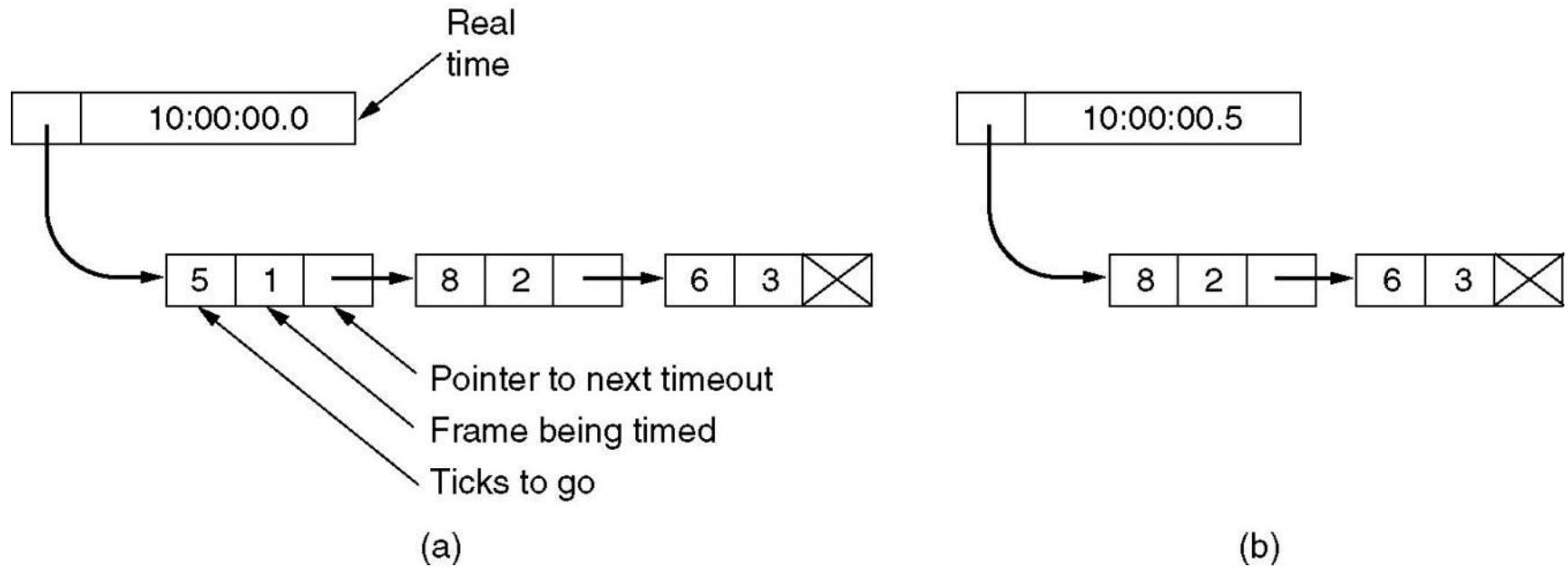


Selective Repeat ARQ



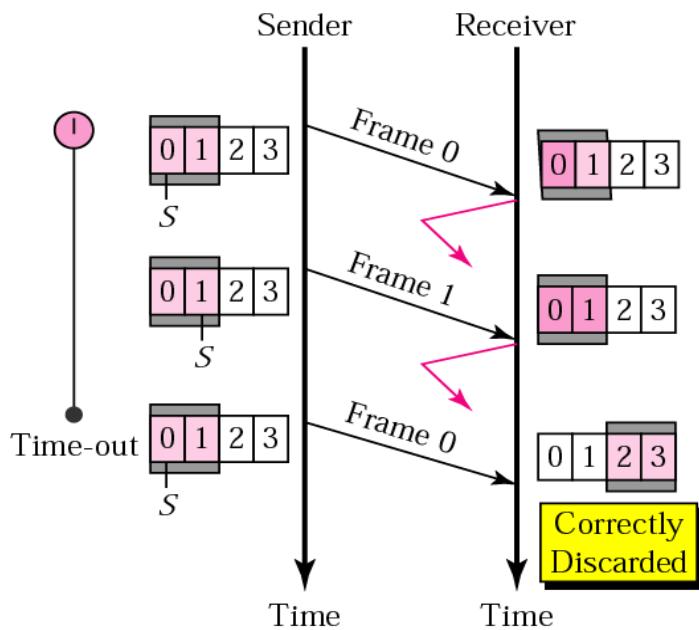
*Figure is courtesy of B. Forouzan

Simulation of Multiple Timers in Software

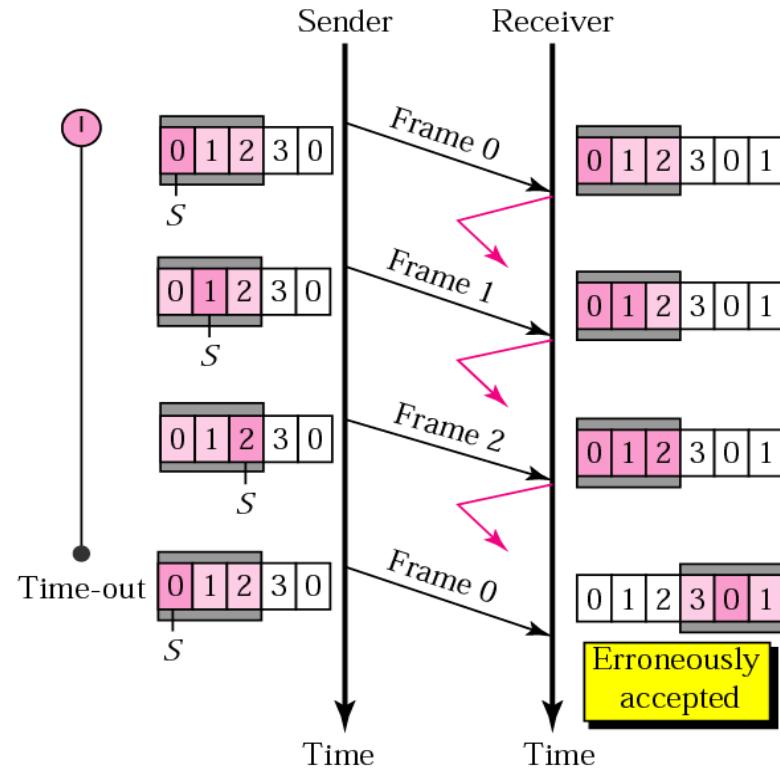


- Maintain linked list of timers
 - Number of frame
 - Offset from current time

Selective Repeat ARQ: Sender Window



a. Window size = 2^{m-1}



b. Window size > 2^{m-1}

Size of the sender and receiver window must be at most one-half of 2^m

* Figure is courtesy of B. Forouzan



Summary: Flow Control

- Flow Control:
 - Stop-and-Wait
 - Sliding Window
- Error Control
 - Stop-and-Wait ARQ
 - Go-back-N ARQ
 - Selective Repeat ARQ

Items from Today

- Bit-Stuffing/Byte-Stuffing
- Flow Control
- Stop-and-Wait
- Sliding Window







CS2031 Telecommunications II

Datagram Sockets



Nodes & Ports



foo.cs.tcd.ie

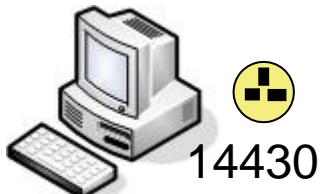
134.226.14.55



bar.cs.tcd.ie

134.226.14.24

Sockets & Ports



foo.cs.tcd.ie

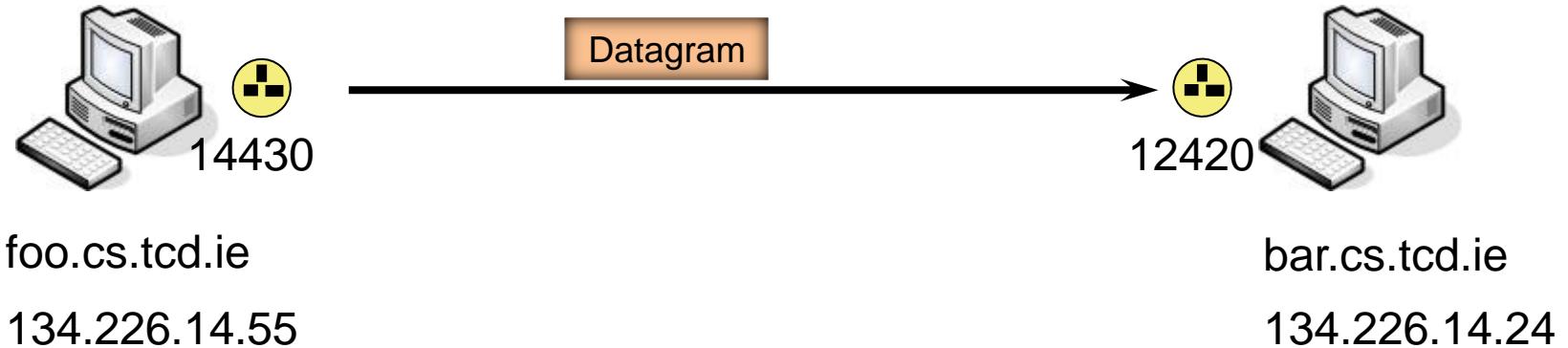
134.226.14.55



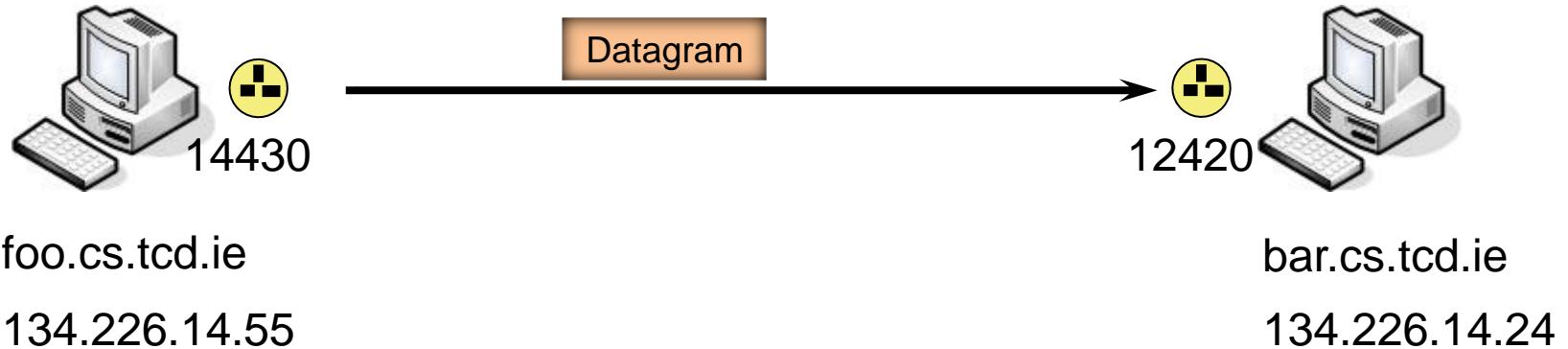
bar.cs.tcd.ie

134.226.14.24

Sockets & Ports



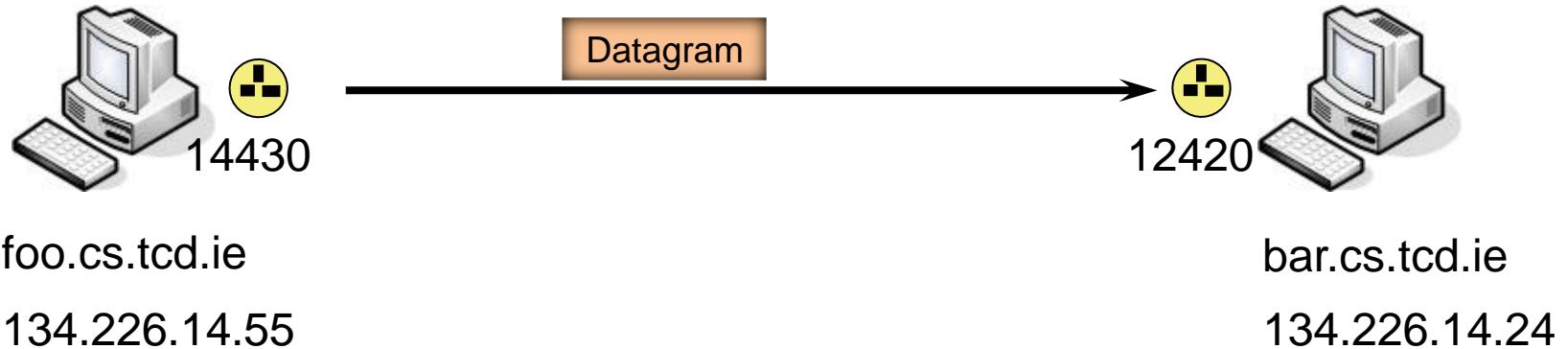
Sockets & Ports



```
socket= new DatagramSocket(14430);
```

```
dstAddress= new InetSocketAddress("bar.cs.tcd.ie", 12420);
packet= new DatagramPacket(data, data.length, dstAddress);
socket.send(packet);
```

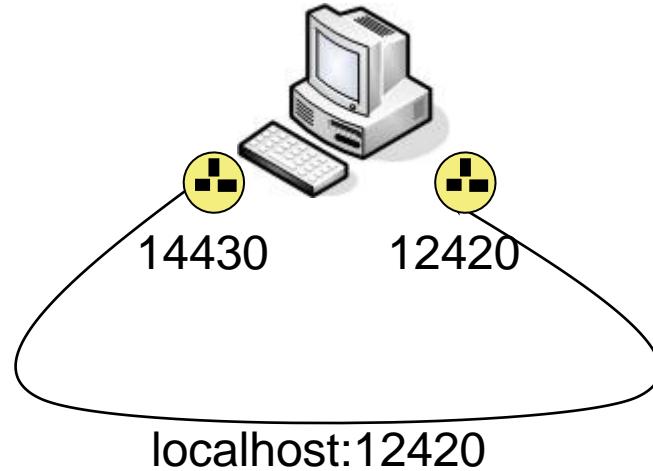
Sockets & Ports



```
socket= new DatagramSocket(12420);
```

```
packet = new DatagramPacket(new byte[SIZE], SIZE);  
socket.receive(packet);
```

Sockets & Ports



```
socket= new DatagramSocket(14430);
```

```
dstAddress= new InetSocketAddress("localhost", 12420);
packet= new DatagramPacket(data, data.length, dstAddress);
socket.send(packet);
```



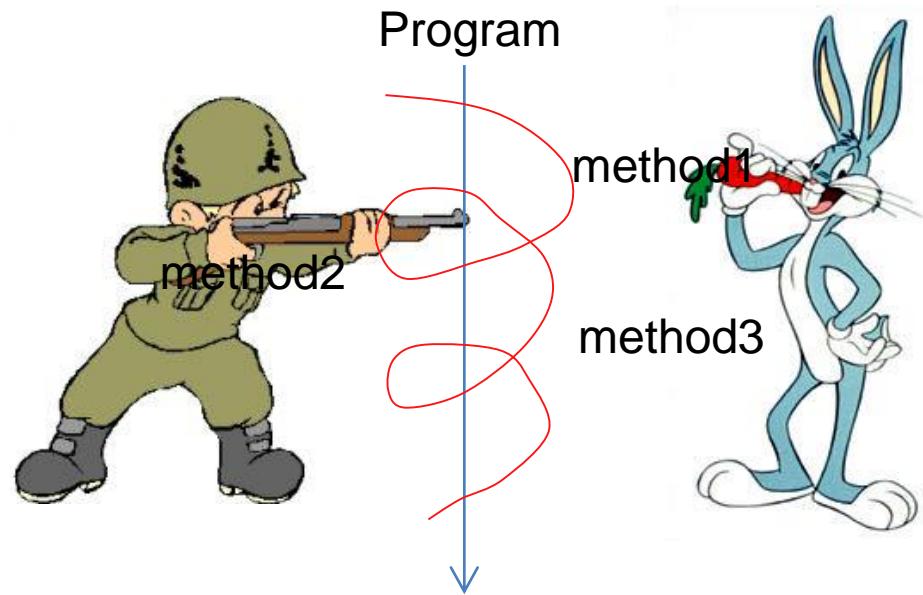
**...and now to something
completely different**



Threads

- Threats of Execution

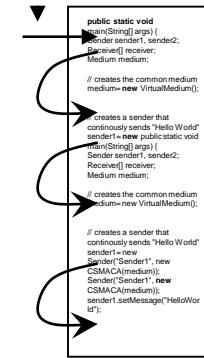
- Lightweight Processes



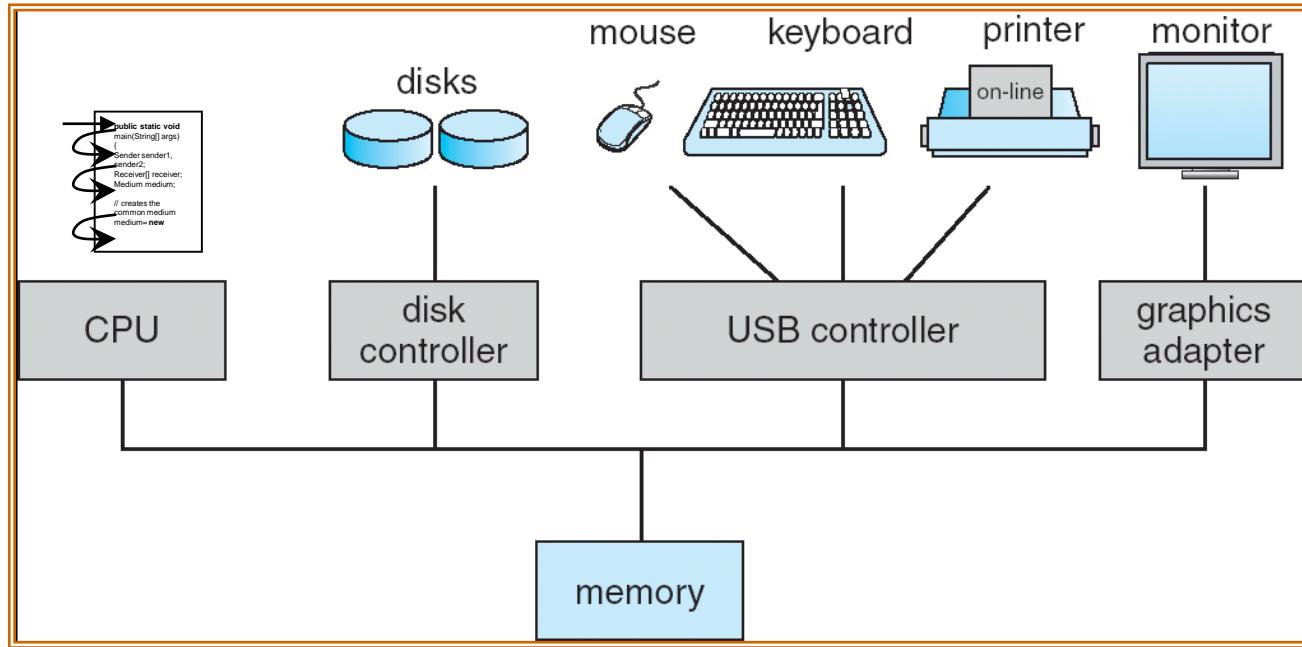
Single-Process System

- One process

Instruction Pointer

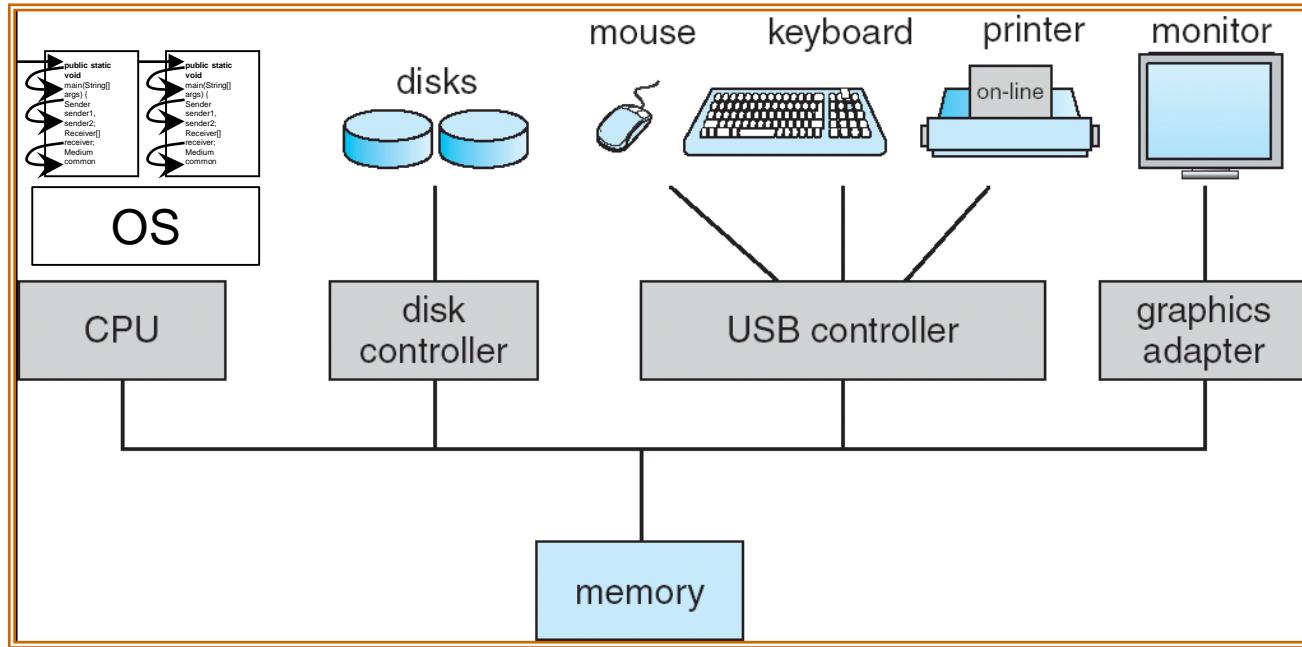


Single Program – Complete Control



* Figure is courtesy of Silberschatz, Galvin, Gagne

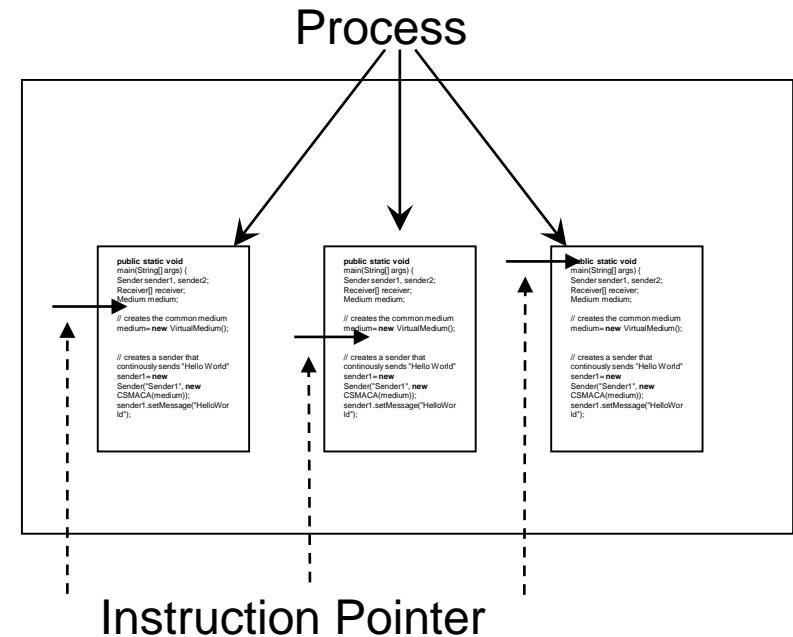
OS & Multiple Programs → Chaos



* Figure is courtesy of Silberschatz, Galvin, Gagne

Processes

- Separate address spaces
- Registers per process
- Problem:
 - Switching between processes



Per-Process Details

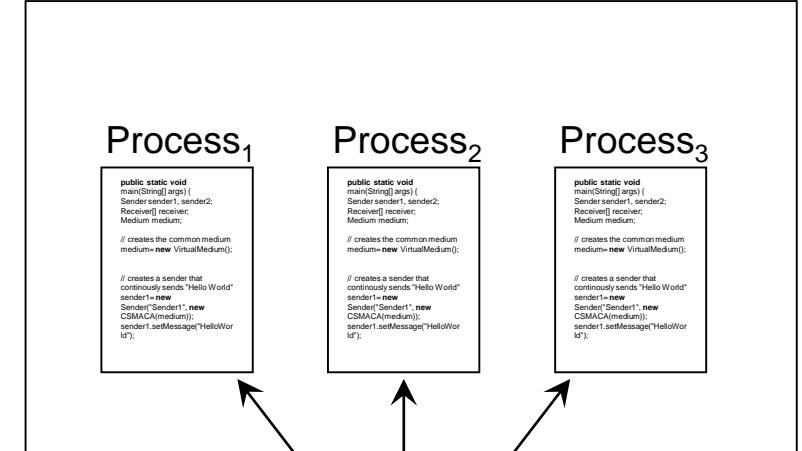
Process management	Memory management	File management
Registers	Pointer to text segment	UMASK mask
Program counter	Pointer to data segment	Root directory
Program status word	Pointer to bss segment	Working directory
Stack pointer	Exit status	File descriptors
Process state	Signal status	Effective uid
Time when process started	Process id	Effective gid
CPU time used	Parent process	System call parameters
Children's CPU time	Process group	Various flag bits
Time of next alarm	Real uid	
Message queue pointers	Effective uid	
Pending signal bits	Real gid	
Process id	Effective gid	
Various flag bits	Bit maps for signals	
	Various flag bits	

* Figure is courtesy of A. Tanenbaum



Process Switching

- Saving of registers
 - Instruction pointer
 - Stack pointers
 - Other registers



- Switching Virtual Memory

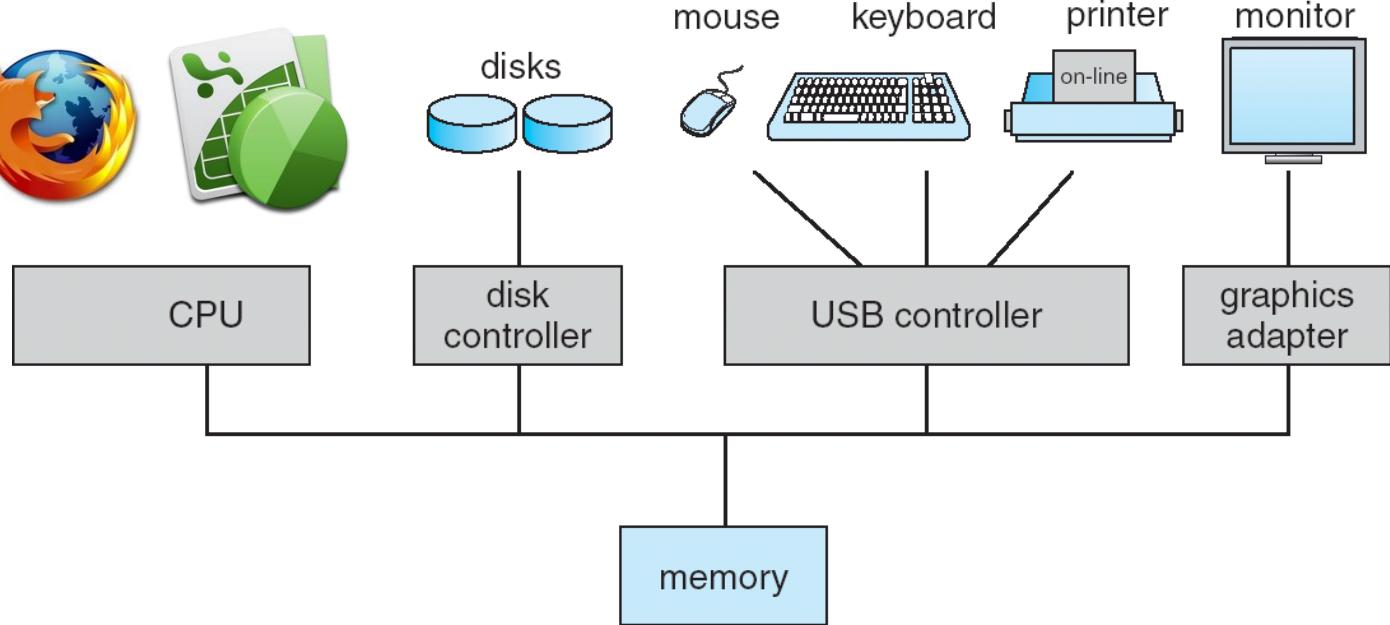


Overhead!

Process management	Memory management	File management
Registers Program counter Program status word Registers Process state This process is shared CPU time used Children's CPU time Time of next alarm Memory queue pointers Pending signals Process id Various flag bits	Pointer to last segment Pointer to current segment Pointer to last segment Signal value Signal value Parent process Current process Read set Write set Effective pid Effective pid Bit maps for signals	UWMS task Working directory Working directory Effect set Effect set System call parameters Various flag bits
Registers Program counter Program status word Registers Process state This process is shared CPU time used Children's CPU time Time when process started CPU time Message queue pointers Pending signals Process id Various flag bits	Pointer to last segment Pointer to data segment Pointer to data segment Ext stat Ext stat Parent process Process id Read set Write set Effective pid Effective pid Bit maps for signals	UWMS task Working directory Working directory Effect set Effect set System call parameters Various flag bits
Registers Program counter Program status word Registers Process state This process is shared CPU time used Children's CPU time Time of next alarm Memory queue pointers Pending signals Process id Various flag bits	Pointer to last segment Pointer to data segment Pointer to data segment Signal status Signal status Parent process Process id Read set Write set Effective pid Effective pid Bit maps for signals	UWMS task Working directory Working directory Effect set Effect set System call parameters Various flag bits



Switching Programs

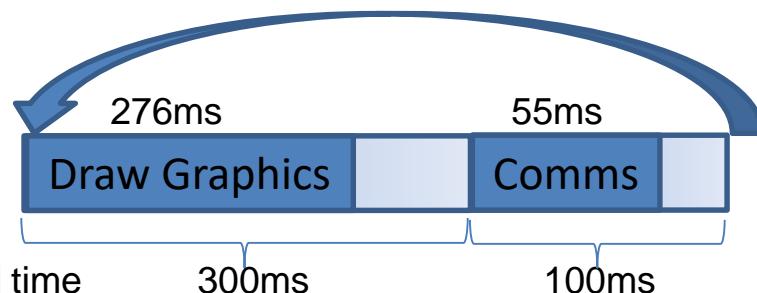
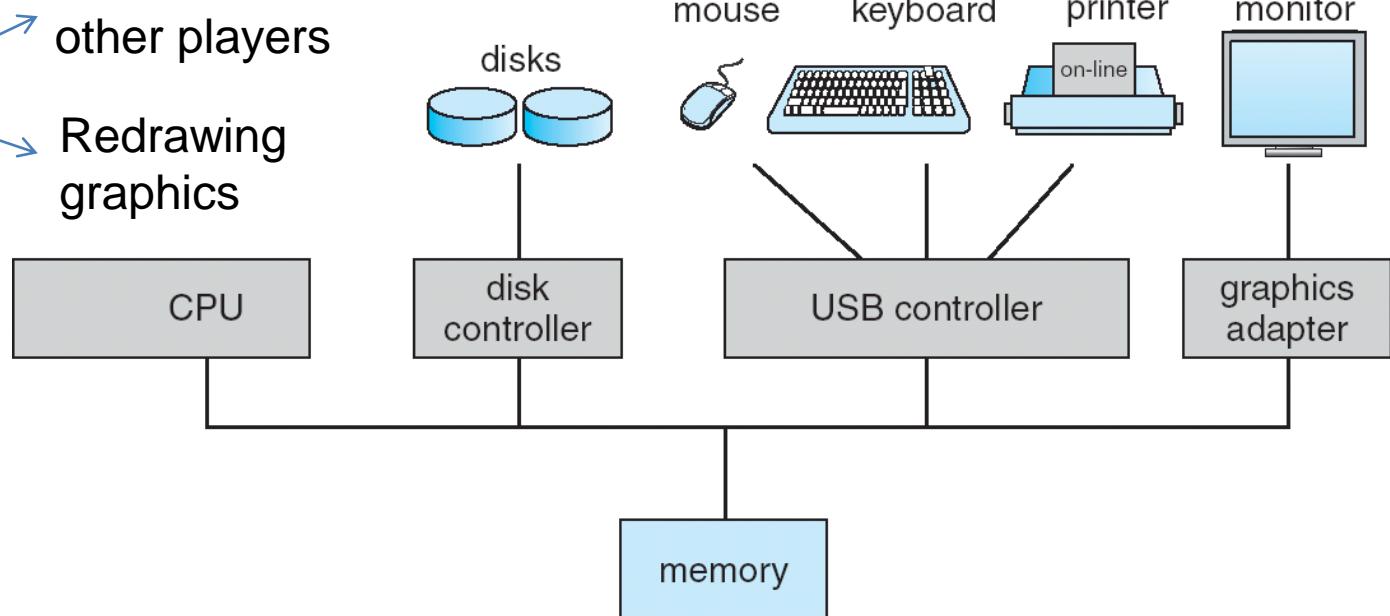


Switching Tasks in a Program



Comms with
other players

Redrawing
graphics



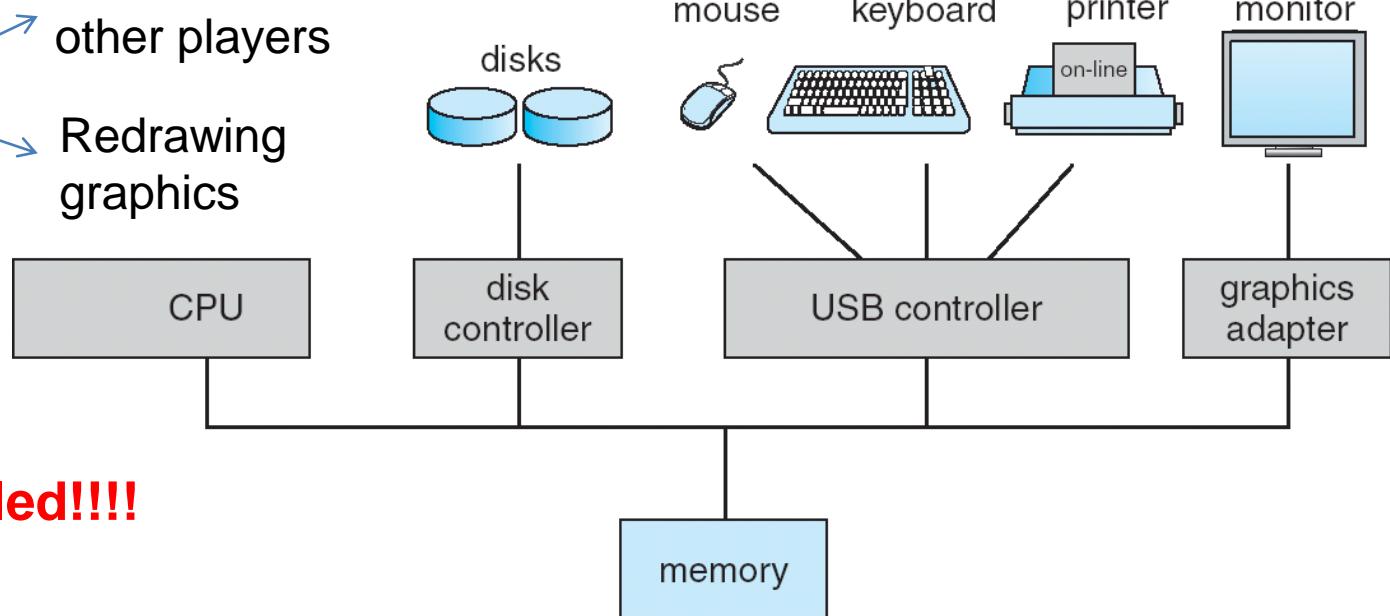
- Every 400ms the graphic will be redrawn
- It is important that the tasks are shorter than the allocated time

Switching Tasks in a Program

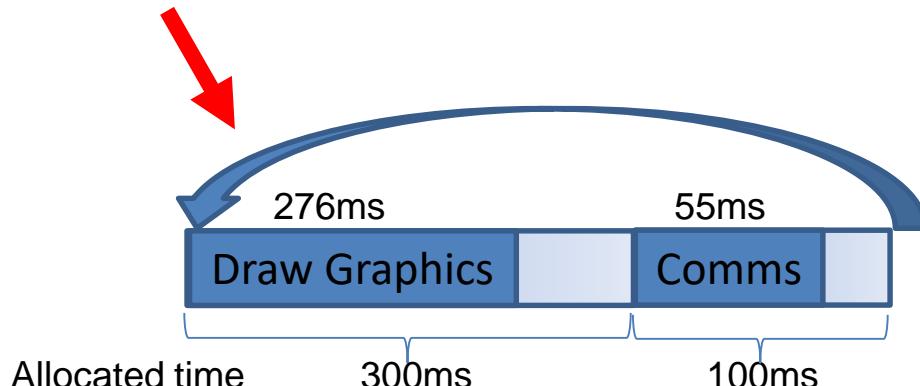


Comms with
other players

Redrawing
graphics



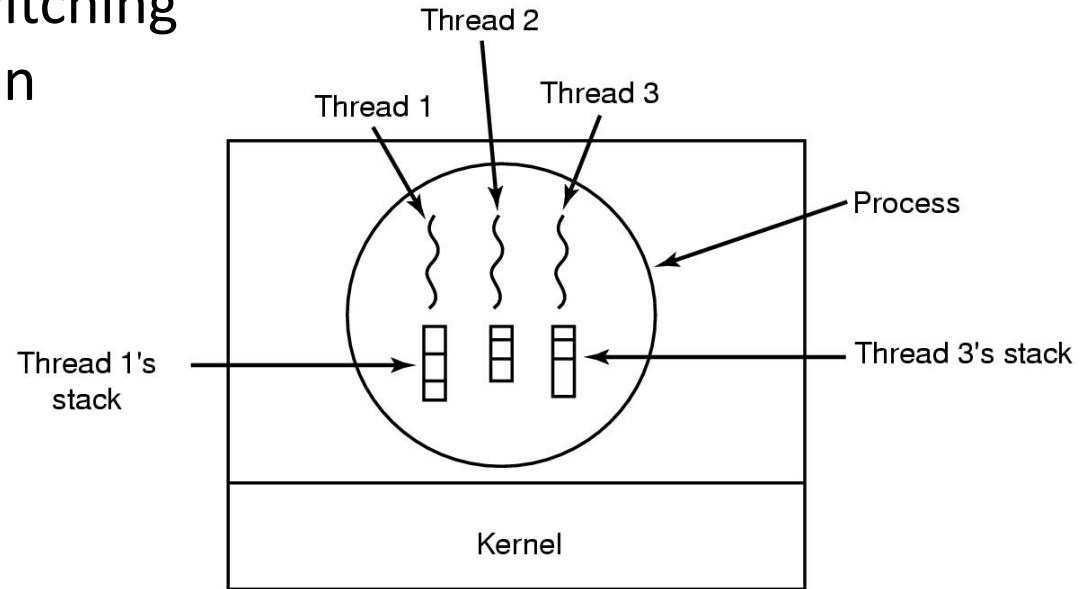
Tightly coupled!!!!



- Every 400ms the graphic will be redrawn
- It is important that the tasks are shorter than the allocated time

Threads

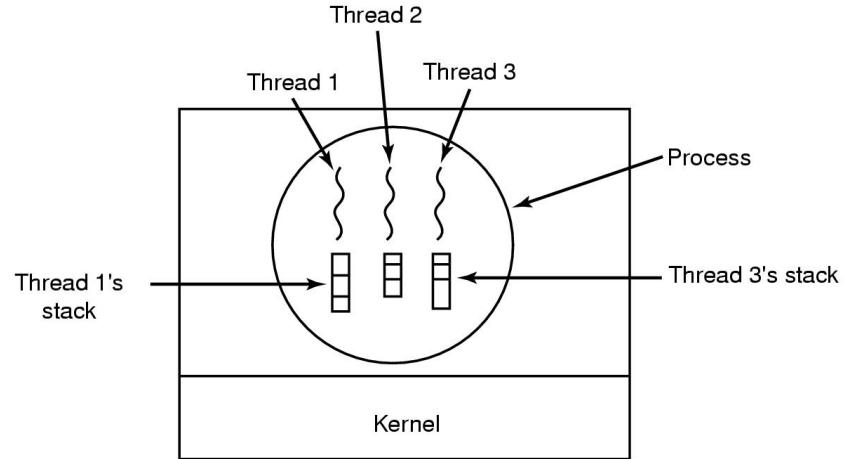
- Lightweight processes
- Share same address space
- Less overhead for switching between threads than between processes



* Figure is courtesy of A. Tanenbaum

Threads

- Lightweight processes
- Share same address space
- Less overhead for switching between threads than between processes



Per process items

Address space
Global variables
Open files
Child processes
Pending alarms
Signals and signal handlers
Accounting information

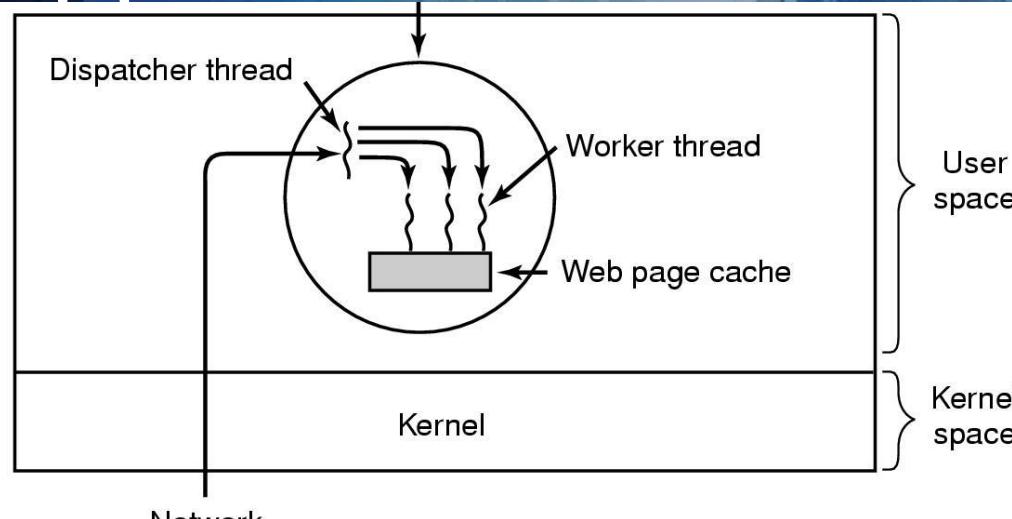
Per thread items

Program counter
Registers
Stack
State

* Figure is courtesy of A. Tanenbaum



Application of Threads



Dispatcher

```
while (TRUE) {
    get_next_request(&buf);
    handoff_work(&buf);
}
```

(a)

Worker

```
while (TRUE) {
    wait_for_work(&buf)
    look_for_page_in_cache(&buf, &page);
    if (page_not_in_cache(&page))
        read_page_from_disk(&buf, &page);
    return_page(&page);
}
```

(b)

* Figure is courtesy of A. Tanenbaum

Java Threads

```
class Thread {  
    public Thread (String name);  
    public Thread (Runnable target)  
    ...  
    public void start ();  
    static void sleep (long millis)  
}  
}
```



Selection of methods of
class “Thread”

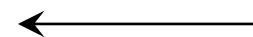
Java Threads

```
class Thread {  
    public Thread (String name);  
    ...  
    public void start ();  
    public void run();  
}
```



Selection of methods of
class “Thread”

```
class XYZ extends Thread {  
    public void run() {  
    }  
}
```



Class that extends
“Thread” needs to
implement the **run**
method

Java Thread – Socket Example I

```
class SocketThread extends Thread {  
    DatagramSocket socket;  
  
    SocketThread (String name, int port) {  
        super (name);  
        socket= new DatagramSocket(port);  
    }  
}
```

```
t1 = new SocketThread ("Socket1", 50000);
```



Java Thread – Socket Example II

```
class SocketThread extends Thread {
```

```
    DatagramSocket socket;
```

```
    SocketThread (String name, int port) {
        super (name);
        socket= new DatagramSocket(port);
    }
```

```
    public void run() {
        while(TRUE) {
            packet= socket.receive();
            System.out.println (name + ":" + packet.getData());
        }
    }
}
```



Creating & Starting Threads I

SocketThread t1, t2, t3;

```
t1 = new SocketThread ("Socket1", 50000);
```

```
t2 = new SocketThread ("Socket2", 50200);
```

```
t3 = new SocketThread ("Socket3", 55000);
```



Creating & Starting Threads II

```
SocketThread t1, t2, t3;
```

```
t1 = new SocketThread ("Socket1", 50000);
```

```
t2 = new SocketThread ("Socket2", 50200);
```

```
t3 = new SocketThread ("Socket3", 55000);
```

```
t1.start();
```

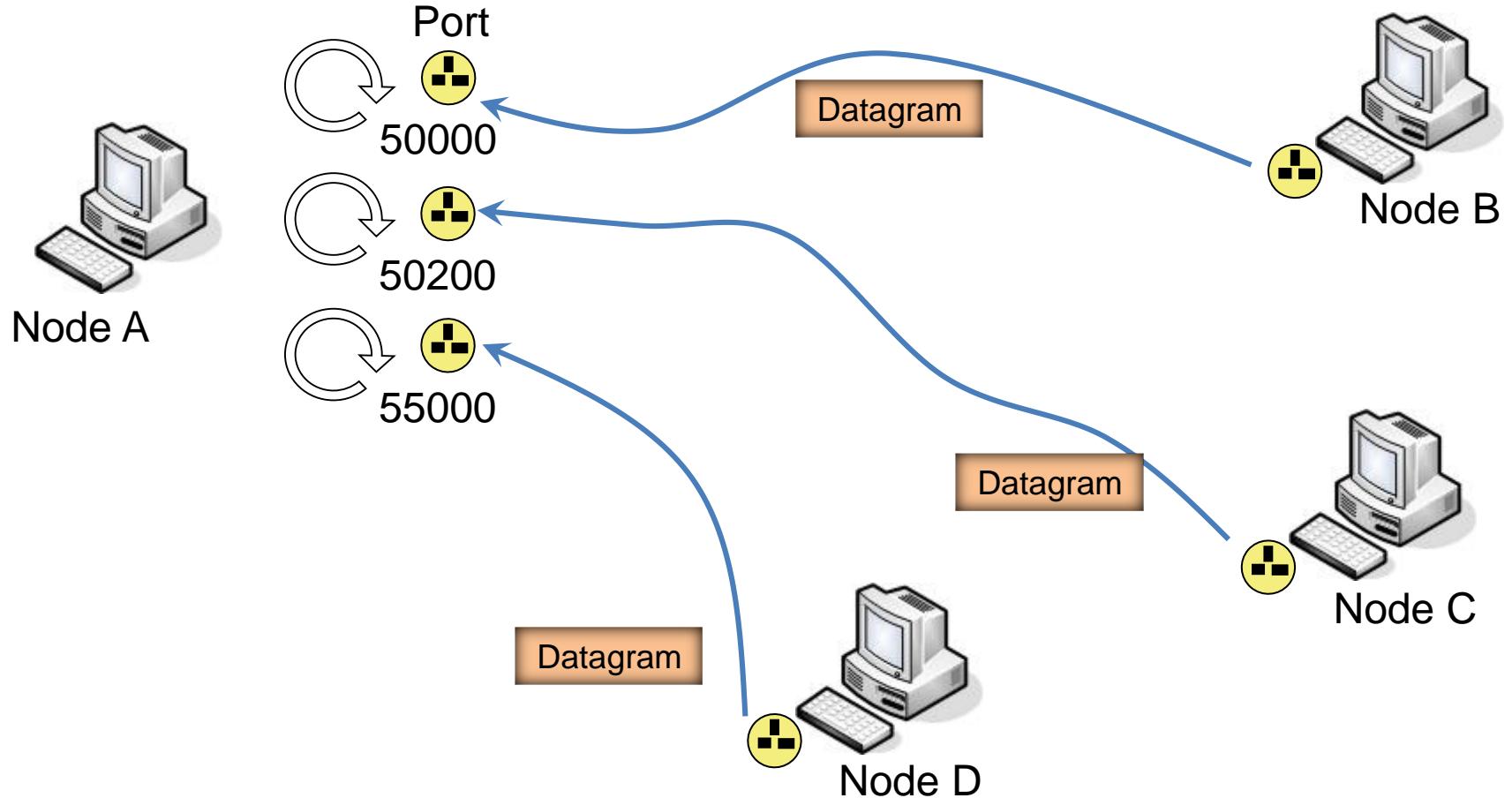
```
t2.start();
```

```
t3.start();
```



Insert thread into list of
running threads and
execute “run” method

Concurrent Communication



Thread Execution Example I

```
class CounterThread extends Thread {  
    long counter;  
  
    CounterThread (String name, long counter) {  
        super (name);  
        this.counter = counter;  
    }  
}
```

```
t1 = new CounterThread ("T1", 10);
```



Thread Execution Example II

```
class CounterThread extends Thread {  
    long counter;  
  
    CounterThread (String name, long counter) {  
        super (name);  
        this.counter = counter;  
    }  
  
    public void run() {  
        while(TRUE) {  
            counter++;  
            System.out.println (name + ":" + counter);  
            Thread.sleep (Math.random() * 5000);  
        }  
    }  
}
```



Thread Execution Example III

```
CounterThread t1, t2, t3;
```

```
t1 = new CounterThread ("T1", 10);
```

```
t2 = new CounterThread ("T2", 10);
```

```
t3 = new CounterThread ("T3", 10);
```

```
t1.start();
```

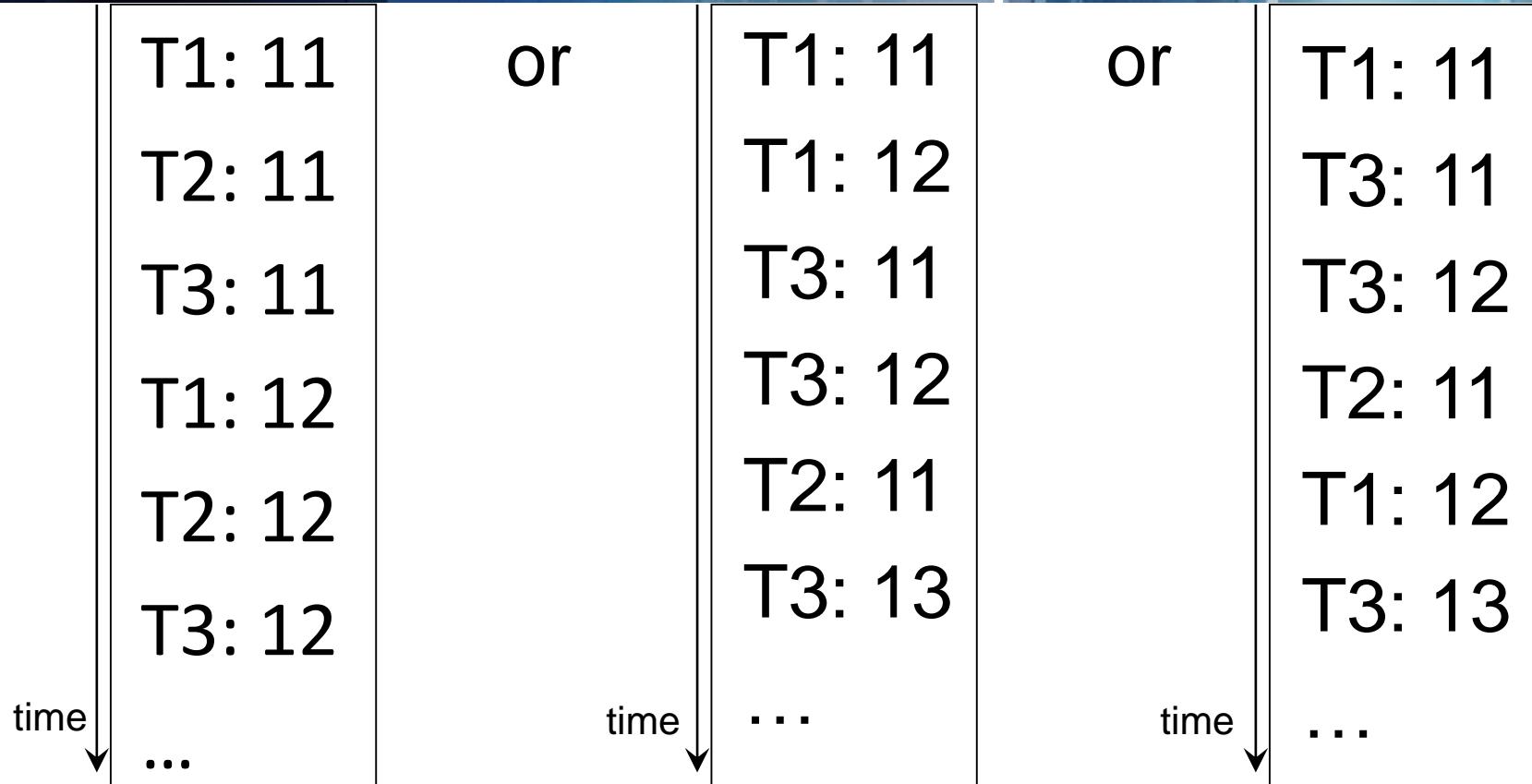
```
t2.start();
```

```
t3.start();
```



Insert thread into list of
running threads and
execute “run” method

Possible Output



Execution is non-deterministic!

Interface: java.lang.Runnable

Java doesn't support Multiple Inheritance:

```
class AccountThread extends Thread, Account {...
```

← **ERROR**

**Java doesn't support
multiple inheritance**



Interface: java.lang.Runnable

Java doesn't support Multiple Inheritance:

```
class AccountThread extends Thread, Account {...} ← ERROR
```

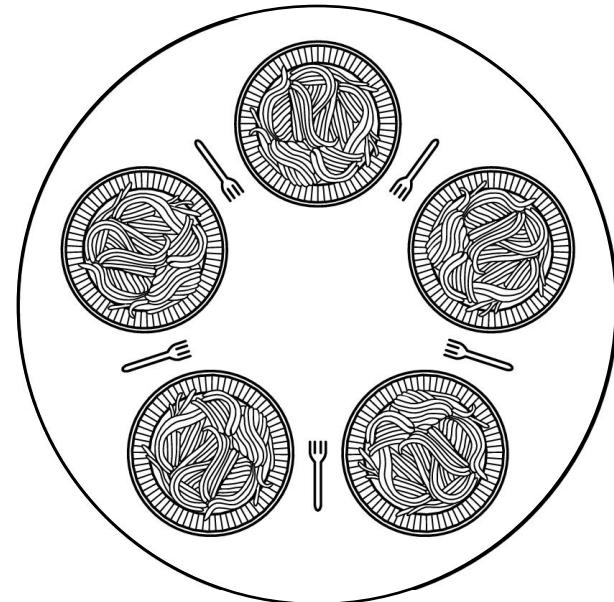
Java doesn't support
multiple inheritance

```
class CounterThread implements Runnable {  
    ...  
    public void run() {  
    }  
}
```

```
new Thread (new CounterThread("T1", 10)).start;
```

Problems with Concurrency

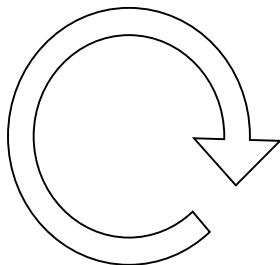
- Concurrent access to global variables, etc
 - Requires synchronization
 - Approaches
 - Monitors
 - Semaphores
 - Barriers
- (see Principles of Concurrent Programming, M. Ben-Ari)



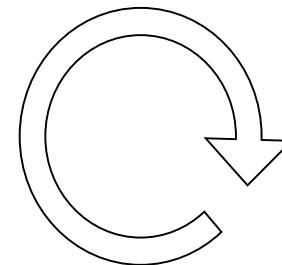
Dining Philosophers

Producer-Consumer Problem

Producer



Consumer



- Producer delivers 1 egg at a time
- Basket can hold exactly 1 egg
- Consumer can only consume an egg if an egg is in the basket

Producer-Consumer in Java I

```
class TestSystem {  
    Basket basket;  
  
    TestSystem() {  
        basket= new Basket(0);  
    }  
}
```

```
class Basket {  
    int content;  
  
    public Basket (int content) {  
        this.content= content;  
    }  
}  
}
```



Producer-Consumer in Java II

```
class TestSystem {  
    ...  
    class Basket {  
        int content;  
        ...  
        public void putEgg () {  
            content++;  
        }  
  
        public void takeEgg() {  
            content--;  
        }  
    }  
}
```



Producer-Consumer in Java III

```
class TestSystem {  
    Basket basket;  
  
    class Producer extends Thread {  
        public void run() {  
            while (true) basket.putEgg();  
        }  
    }  
  
    class Consumer extends Thread {  
        public void run() {  
            while (true) basket.takeEgg();  
        }  
    }  
}
```



Producer-Consumer in Java IV

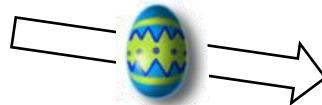
```
class TestSystem {  
  
    public static void main (String[] args) {  
        Producer producer;  
        Consumer consumer;  
  
        producer= new Producer();  
        consumer= new Consumer();  
  
        producer.start();  
        consumer.start();  
    }  
}
```



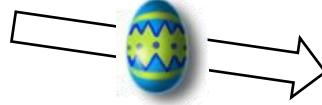
Problem???

Producer

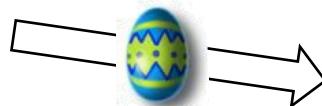
putEgg



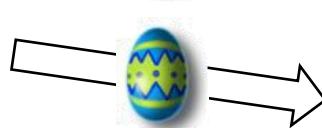
putEgg



putEgg

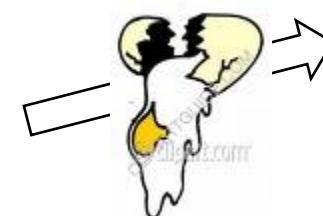


putEgg

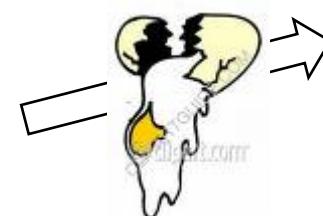


Consumer

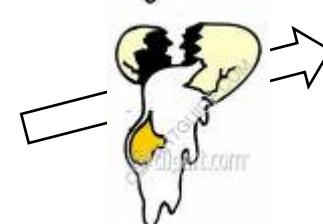
getEgg



getEgg



getEgg



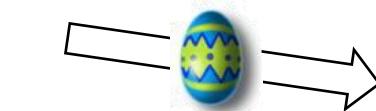
Execution is non-deterministic!



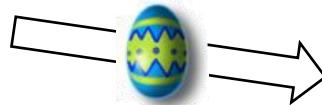
Problem???

Producer

putEgg

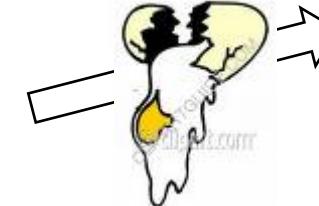
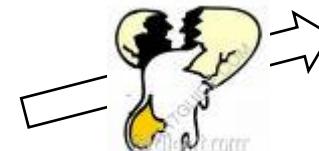
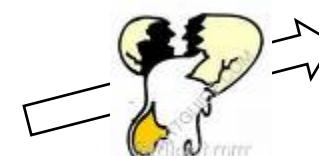


putEgg



Consumer

getEgg



getEgg

getEgg

getEgg

Execution is non-deterministic!

Producer-Consumer in Java V

```
class TestSystem {  
    ...  
    class Basket {  
        int content;  
        ...  
        public synchronized void putEgg () {  
            while (content!=0) wait();  
            content++;  
            notify();  
        }  
    }  
}
```



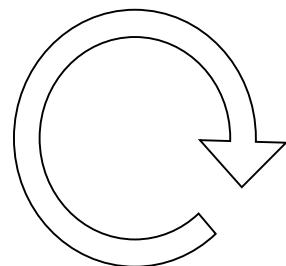
Producer-Consumer in Java VI

```
class TestSystem {  
    ...  
    class Basket {  
        int content;  
        ...  
        public synchronized void takeEgg () {  
            while (content!=1) wait();  
            content--;  
            notify();  
        }  
    }  
}
```

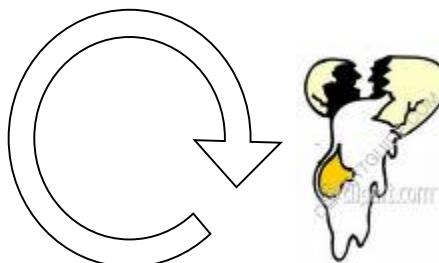


Producer-Consumer Problem

Producer



Consumer

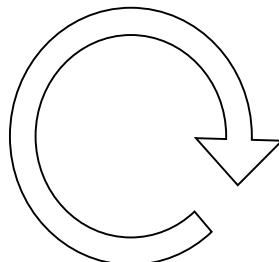


```
public synchronized void putEgg () {  
    while (content!=0) wait();  
    content++;  
    notify();  
}
```

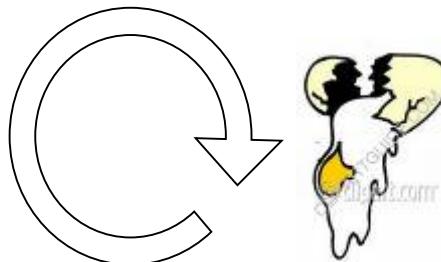
```
public synchronized void takeEgg () {  
    while (content!=1) wait();  
    content--;  
    notify();  
}
```

Producer-Consumer Problem

Producer



Consumer



```
public synchronized void putEgg () {  
    while (content!=0) wait();  
    content++;  
    notify();  
}
```

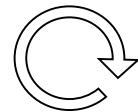
```
public synchronized void takeEgg () {  
    while (content!=1) wait();  
    content--;  
    notify();  
}
```

Monitor in Java: One active thread in method per instance!

Monitors in Java



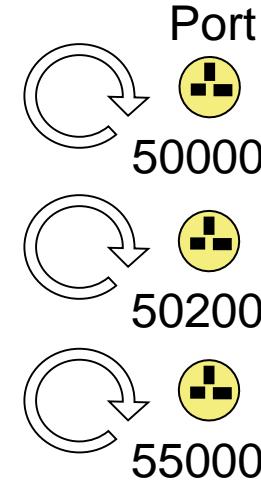
Node A



Thread waiting
for notification



Shared Data



```
synchronized void getSharedData() {   synchronized void changeSharedData() {  
    wait();                         // change data  
    //do something                      notifyAll();  
}                                         }  
}
```

Only one thread can be in a synchronized method of a class at a given time.

Synchronized Methods

```
class SharedData {
```

```
    synchronized void put(Object o) {...}
```

```
    synchronized Object get() {...}
```

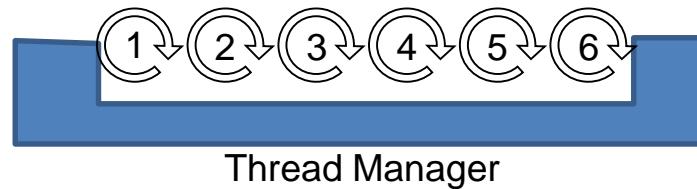
```
    synchronized void printContent() {...}
```

```
}
```



A Word of Warning

- Costs associated with Threads
 - Time for creation
 - Memory allocation
 - Garbage collection
 - etc
- Moderation is the key
- Thread pools
- Deadlocks!



Summary: Threads

- Concurrent Execution
 - Non-deterministic Execution
- Java
 - Inherit from Thread class
 - Implement Runnable interface
- Synchronization
 - `wait()` & `notifyAll()` / `notify()`





CS2031

Telecommunications II

Event-based Programming



“Event-based Programming”

```
public void run() {  
    DatagramPacket packet;  
  
    try {  
        while(true) {  
            packet = new DatagramPacket(new byte[PACKETSIZE], PACKETSIZE);  
            socket.receive(packet);  
            onReceipt(packet);  
        }  
    } catch (Exception e) {e.printStackTrace();}  
}
```



“Event-based Programming”

Listener : Thread



receive packet
call onReceipt()

“Event-based Programming”

abstract Node

Listener : Thread



onReceipt()

void abstract onReceipt()



“Event-based Programming”

Client : Node

```
void onReceipt(packet) {  
    /*handle packet*/  
}
```

abstract Node

Listener : Thread

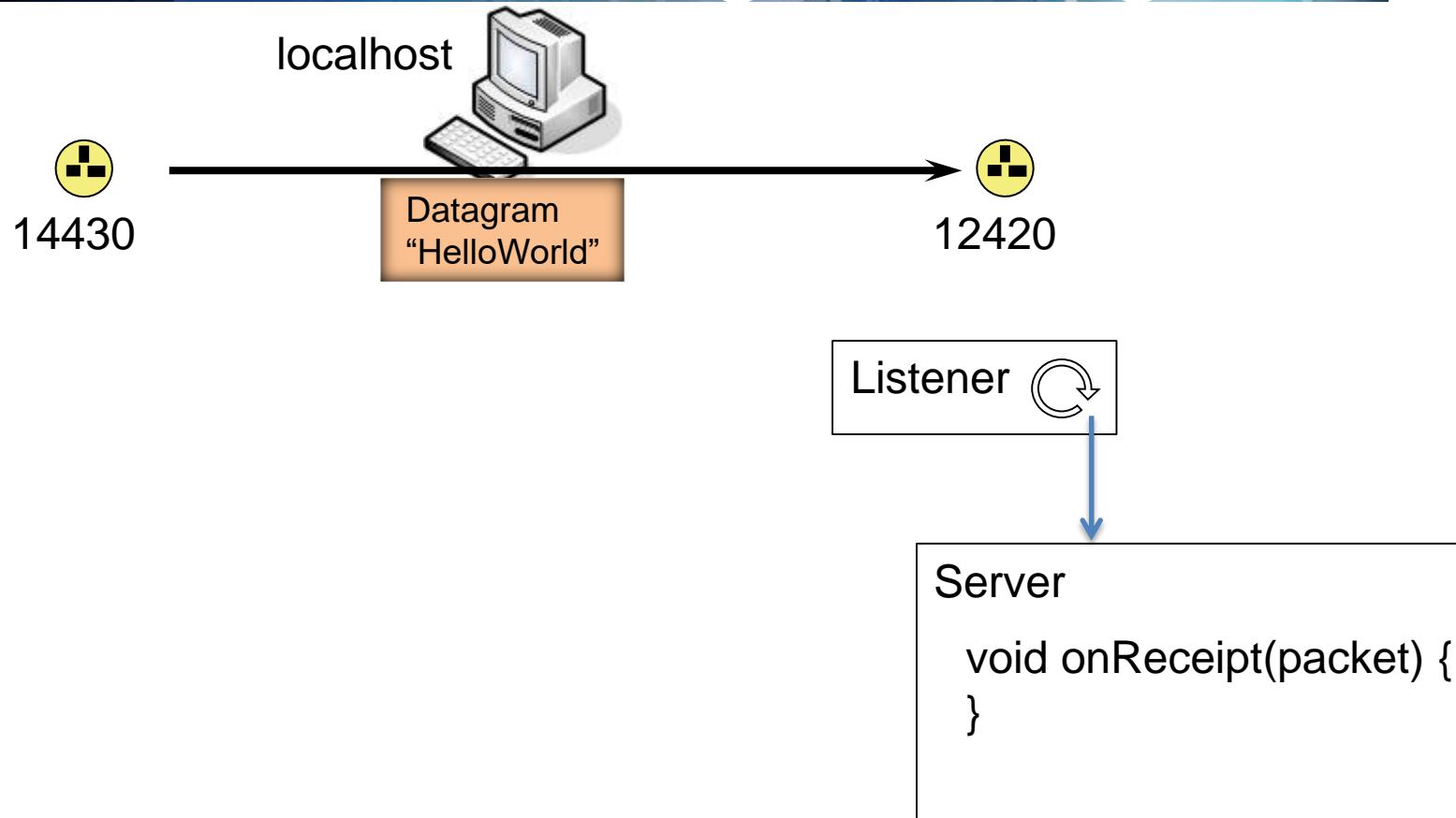


onReceipt()

void abstract onReceipt()



“Event-based Programming”



“Event-based Programming”

```
public void go() {latch.countDown();}  
public void run() {  
    DatagramPacket packet;  
  
    try {  
        latch.await();  
        while(true) {  
            packet = new DatagramPacket(new byte[PACKETSIZE], PACKETSIZE);  
            socket.receive(packet);  
            onReceipt(packet);  
        }  
    } catch (Exception e)  
        {if (!(e instanceof SocketException)) e.printStackTrace();}}
```





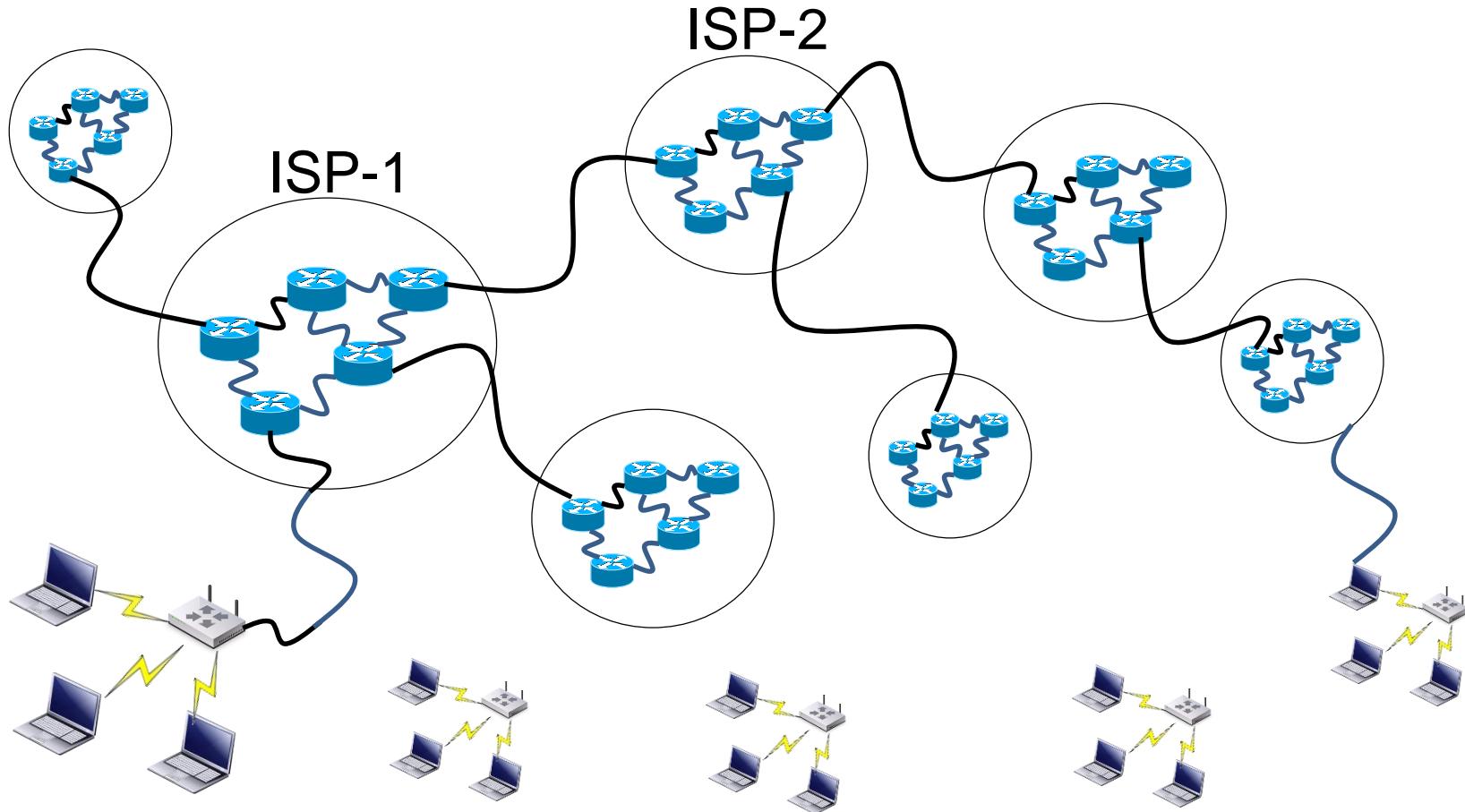
CS2031

Telecommunications II

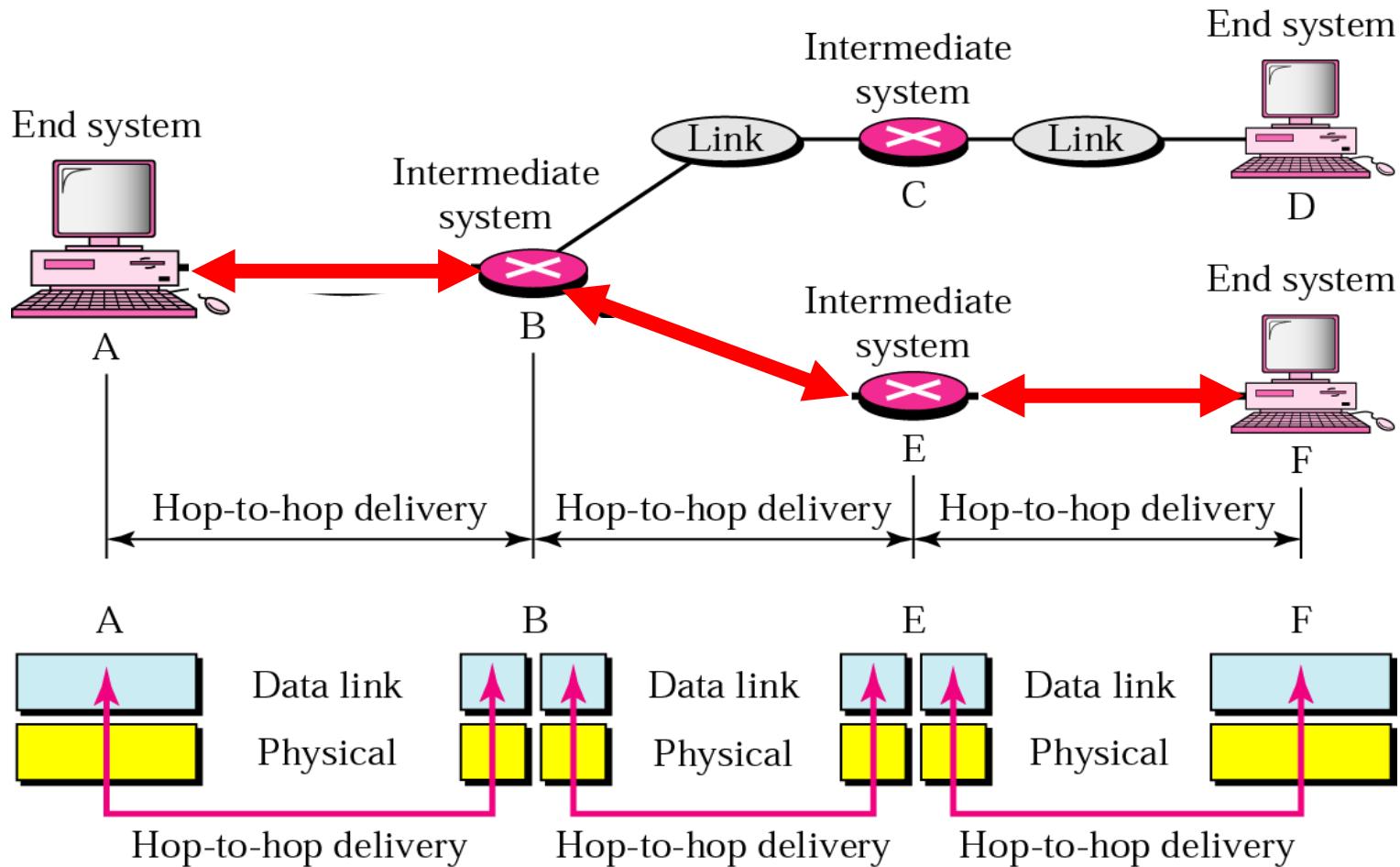
Error Detection and Correction



Internet = Network of Networks

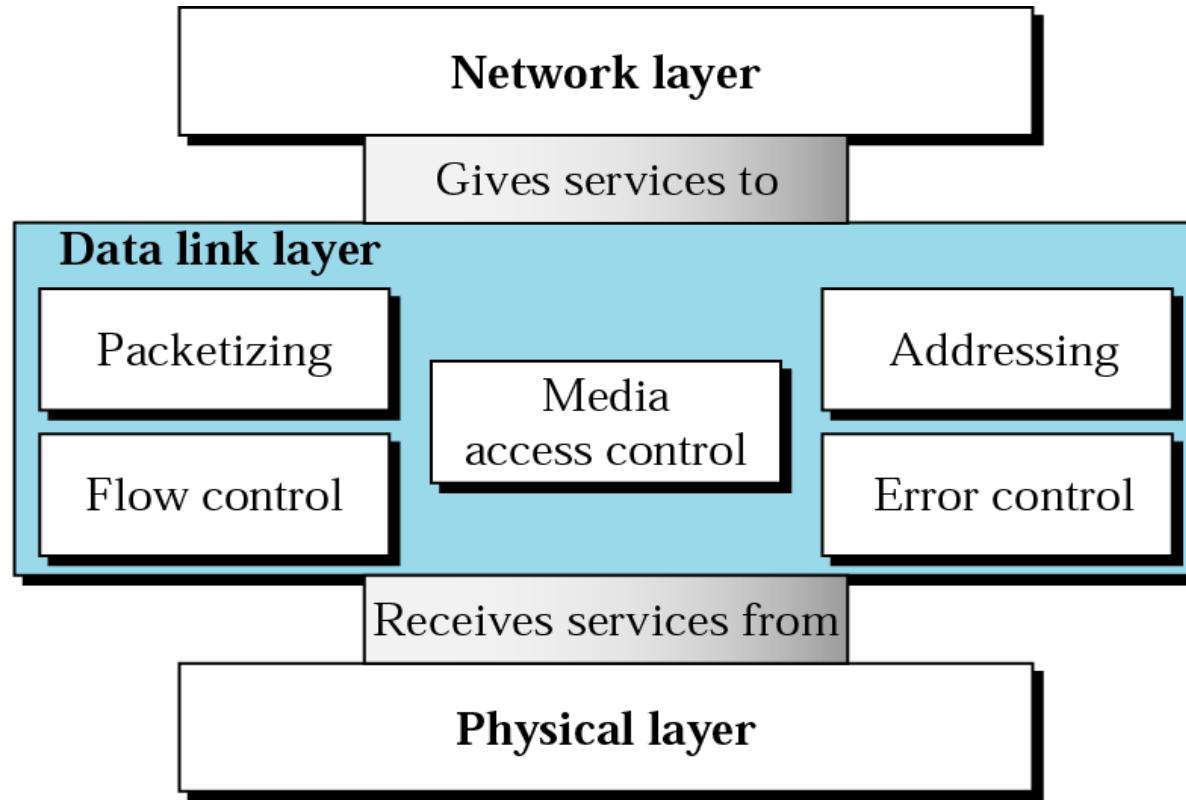


Link Layer



* Figure is courtesy of B. Forouzan

Duties of the Link Layer



The link layer is responsible for transmitting frames from one station to the next.

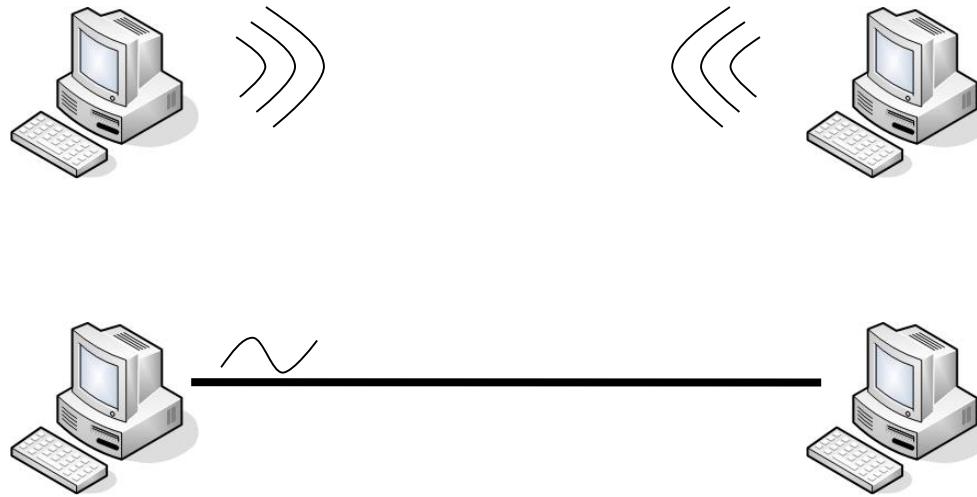
* Figure is courtesy of B. Forouzan

Errors in Transmissions

- Causes for Errors
- Types of Errors
- Detection of Errors
- Correction of Errors

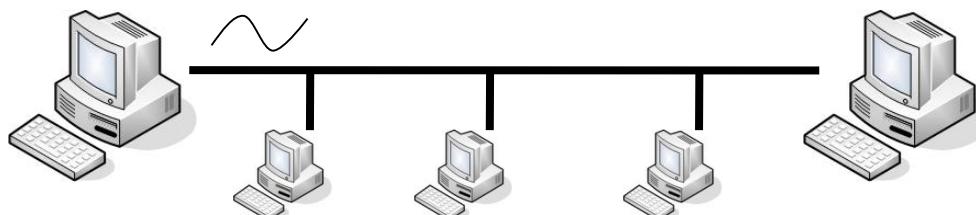
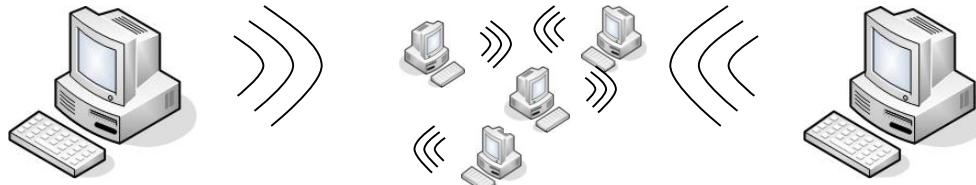


Terminal to Terminal Comms



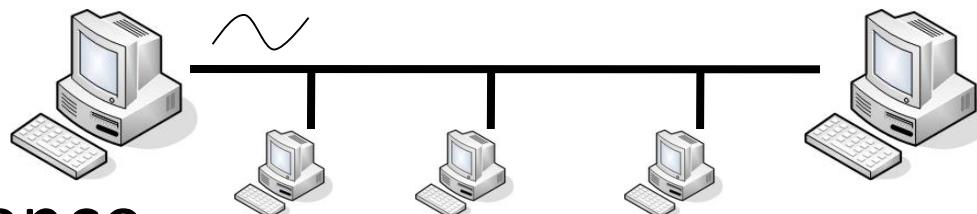
- Either over dedicated or shared medium

Causes for Errors



- **Interference**
 - Collision with communication from other nodes
 - Electrical interference from third parties
 - Thermal interference

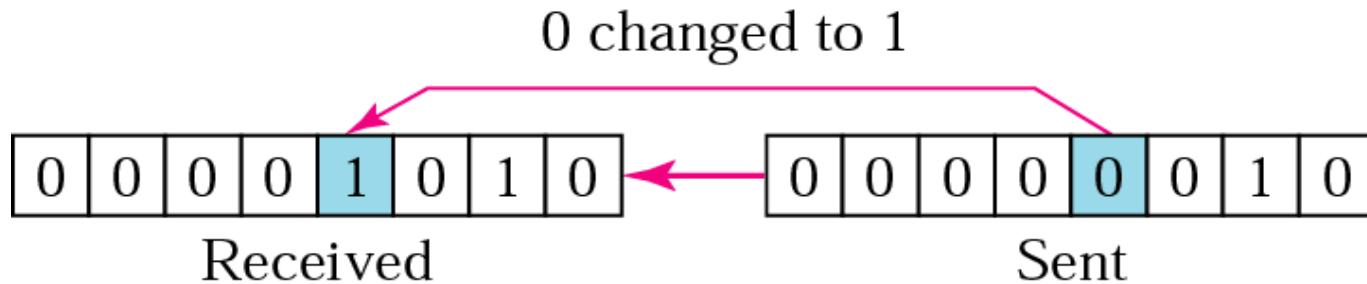
Causes for Errors



- **Interference**
 - Collision with communication from other nodes
 - Electrical interference from third parties
 - Thermal interference

Types of Errors: Single-Bit Error

In a **single-bit error**, only **one bit** in the data unit has changed.

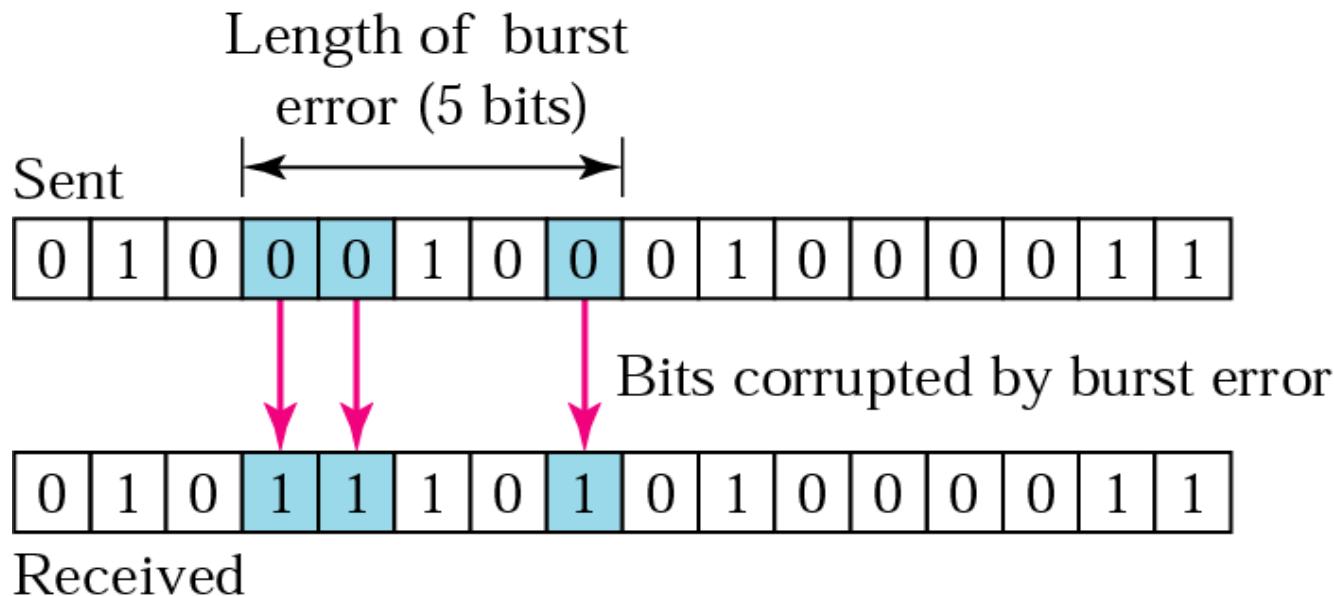


* Figure is courtesy of B. Forouzan



Types of Errors: Burst Error

A **burst error** means that **2 or more bits** in the data unit have changed

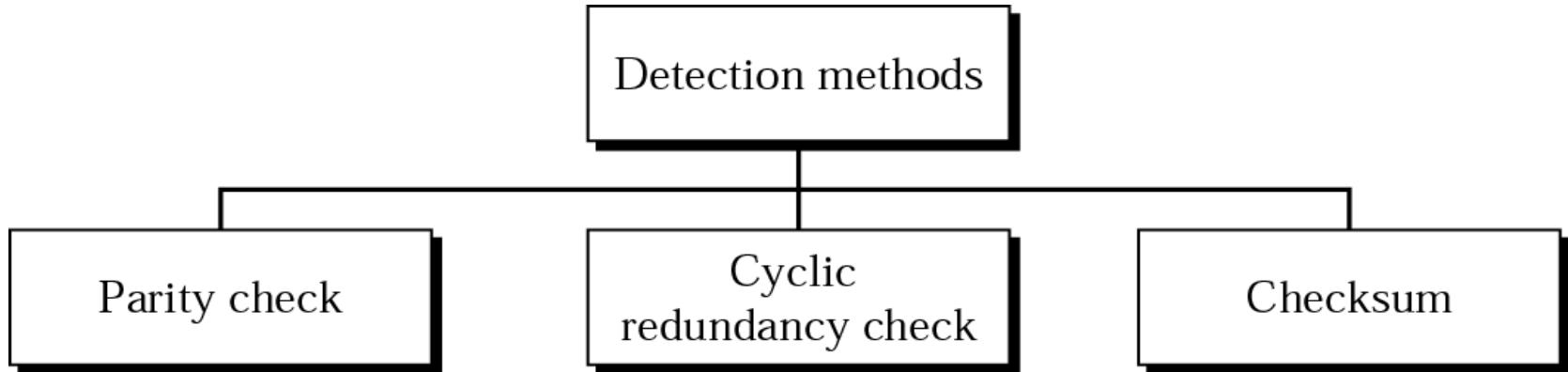


* Figure is courtesy of B. Forouzan



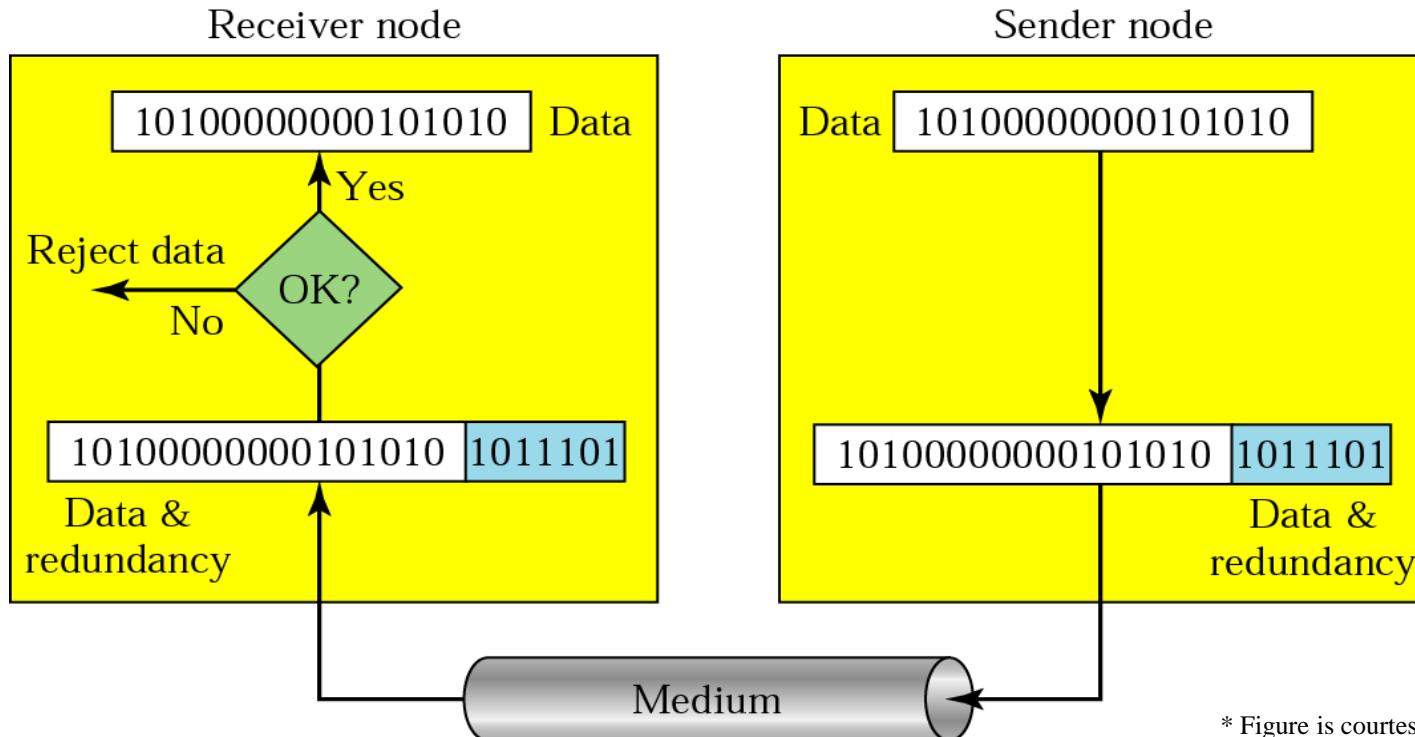
Detection of Errors

- Redundancy



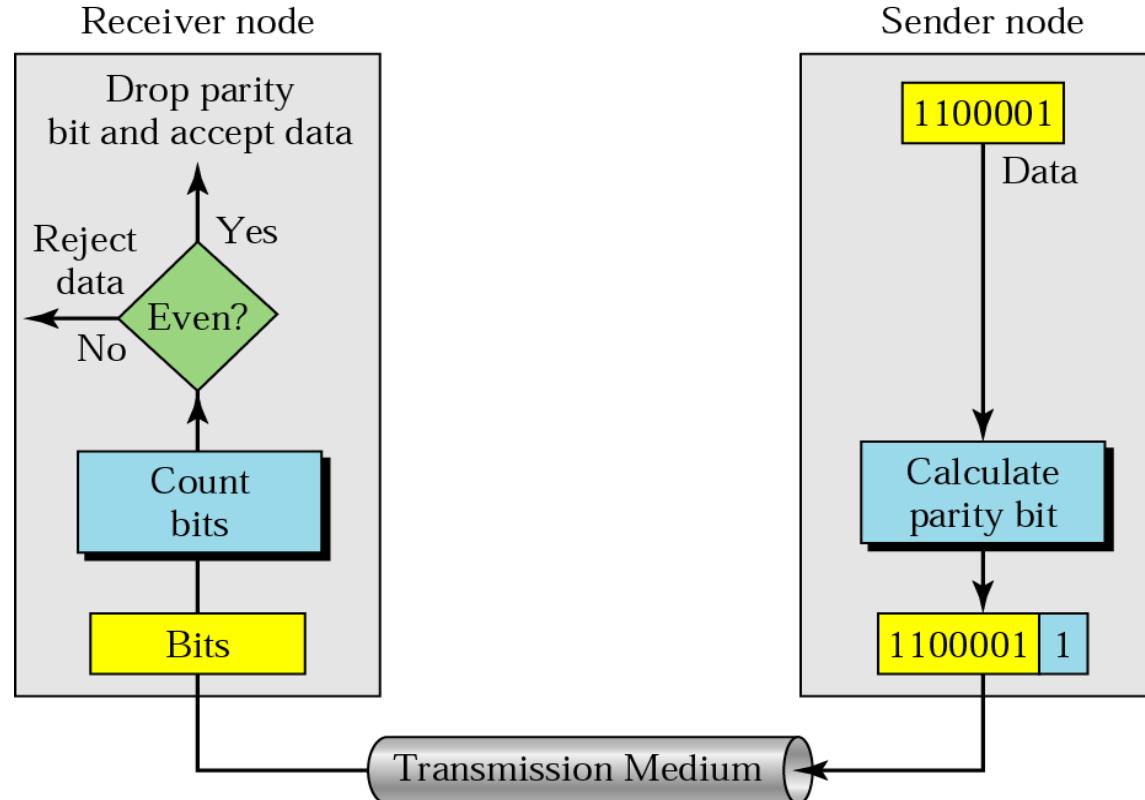
Redundancy

Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination



* Figure is courtesy of B. Forouzan

Even-Parity Concept



A parity bit is added to every data unit so that the total number of 1s is even (or odd for odd-parity).

* Figure is courtesy of B. Forouzan

Even-Parity: Example - Sender

- Assume you want to send the following:

1110111 1101111 1110010 1101100 1100100

- The following bits are actually sent:

11101110 11011110 11100100 11011000 11001001

Even-Parity: Example - Receiver

11101110	11011110	11100100	11011000	11001001
6	6	4	4	4

- The receiver counts the 1s in each character and comes up with even numbers (6, 6, 4, 4, 4). The data are accepted.

11111110	11011110	11101100	11011000	11001001
7	6	5	4	4

- The receiver counts the 1s in each character and comes up with even and odd numbers (7, 6, 5, 4, 4). The receiver knows that the data are corrupted, discards them, and asks for retransmission.

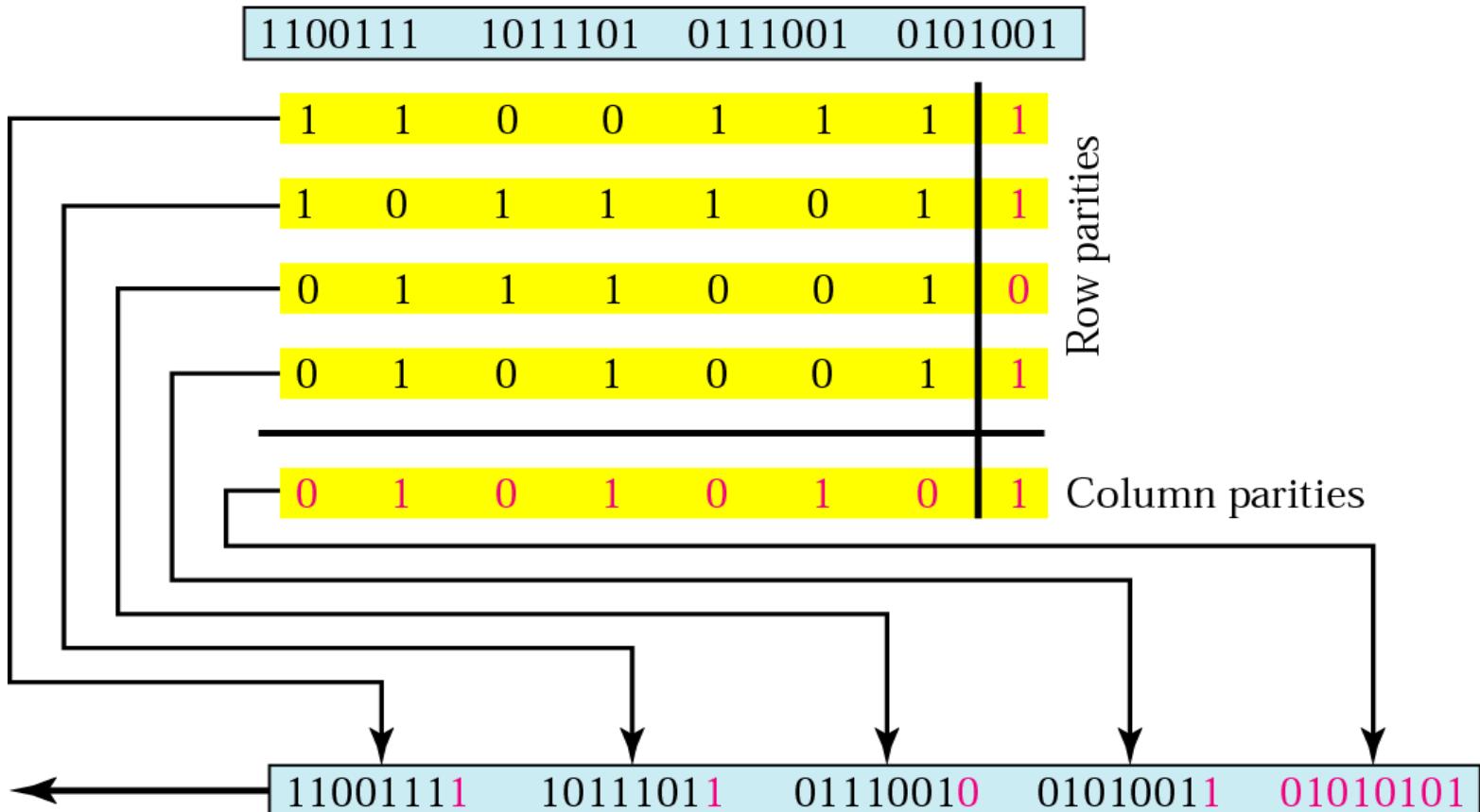
Simple Parity Check

- Can detect all single-bit errors
- Can detect burst errors only if the total number of errors in each data unit is odd



Two-Dimensional Parity Check

In two-dimensional parity check, a block of bits is divided into rows and a redundant row of bits is added to the whole block.



* Figure is courtesy of B. Forouzan

Example: 2D-Parity Check

Suppose the following block is sent:

10101001 00111001 11011101 11100111 10101010

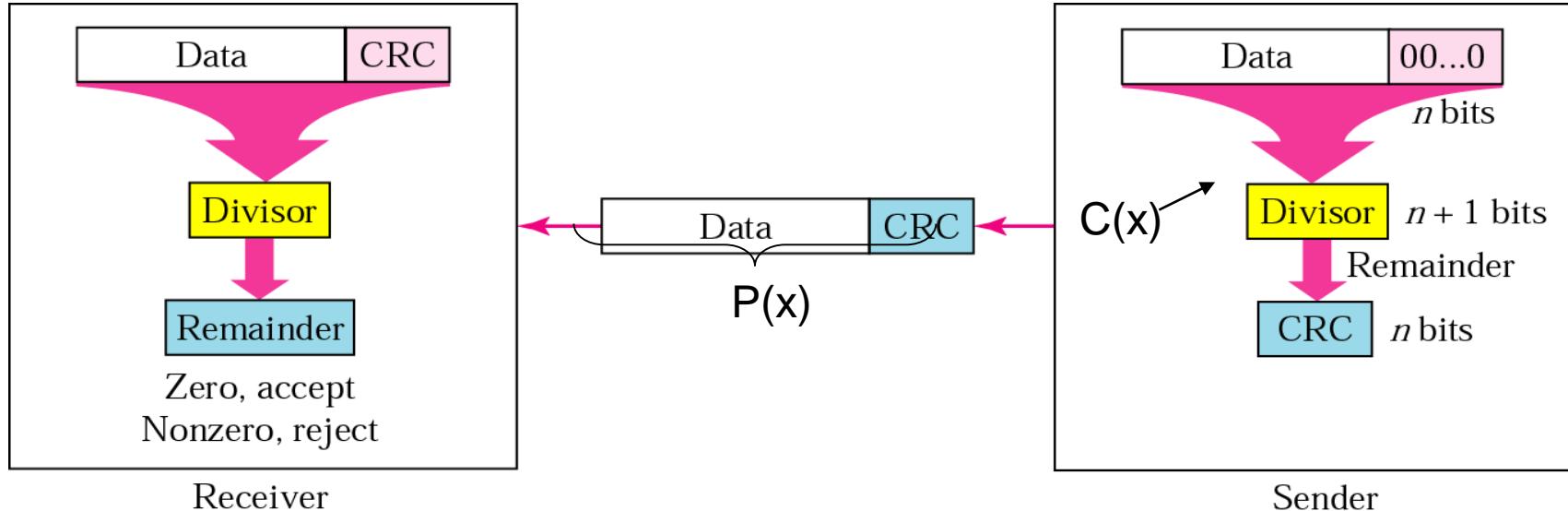
However, it is hit by a burst noise of length 8, and some bits are corrupted.

10100011 10001001 11011101 11100111 10101010

When the receiver checks the parity bits, some of the bits do not follow the even-parity rule and the whole block is discarded.

10100011 10001001 11011101 11100111 10101010

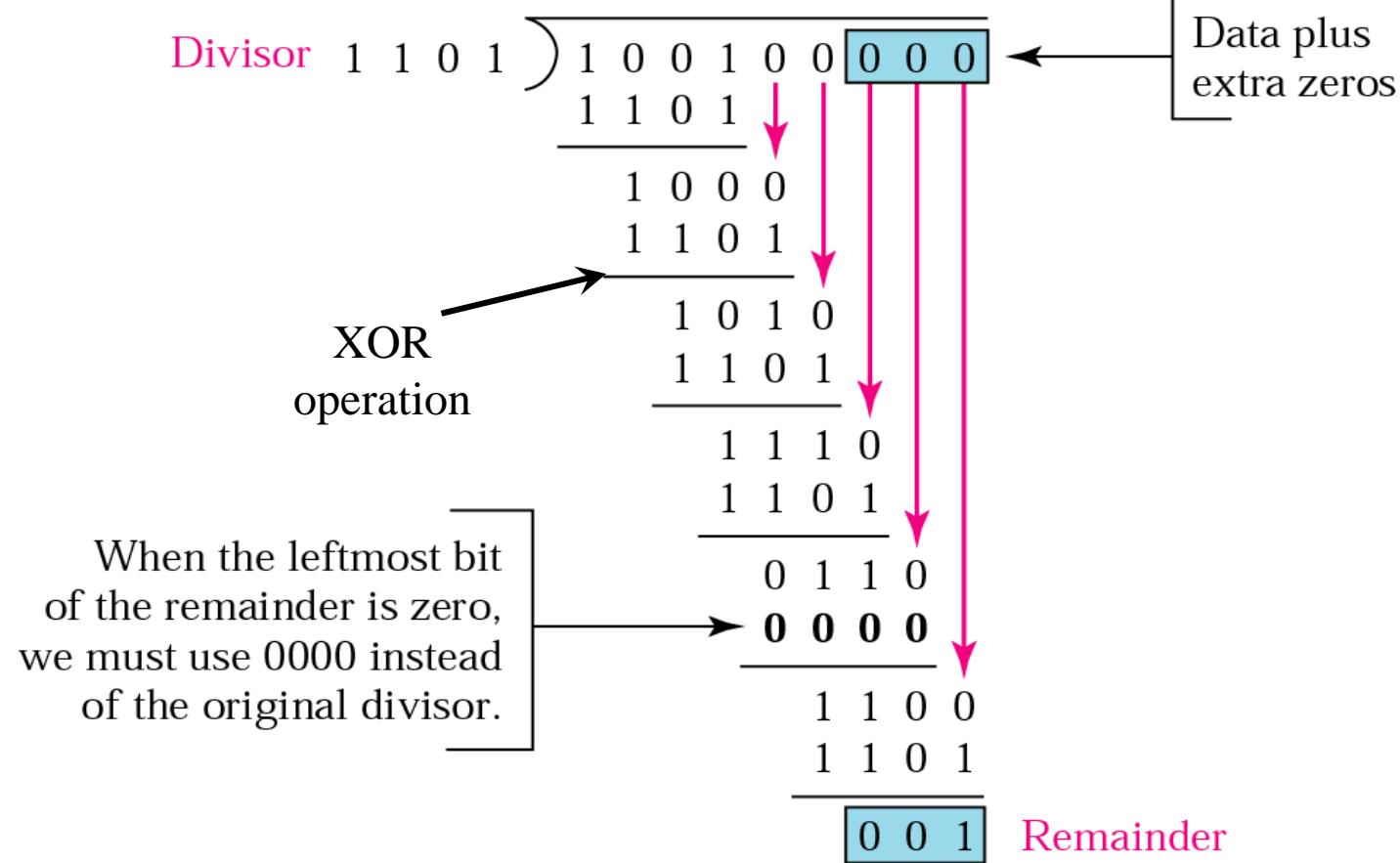
Cyclic Redundancy Check (CRC)



- $P(x)$ divided by $C(x) = 0$
- $(P(x)+\text{remainder})$ divided by $C(x)$ should be $\neq 0$

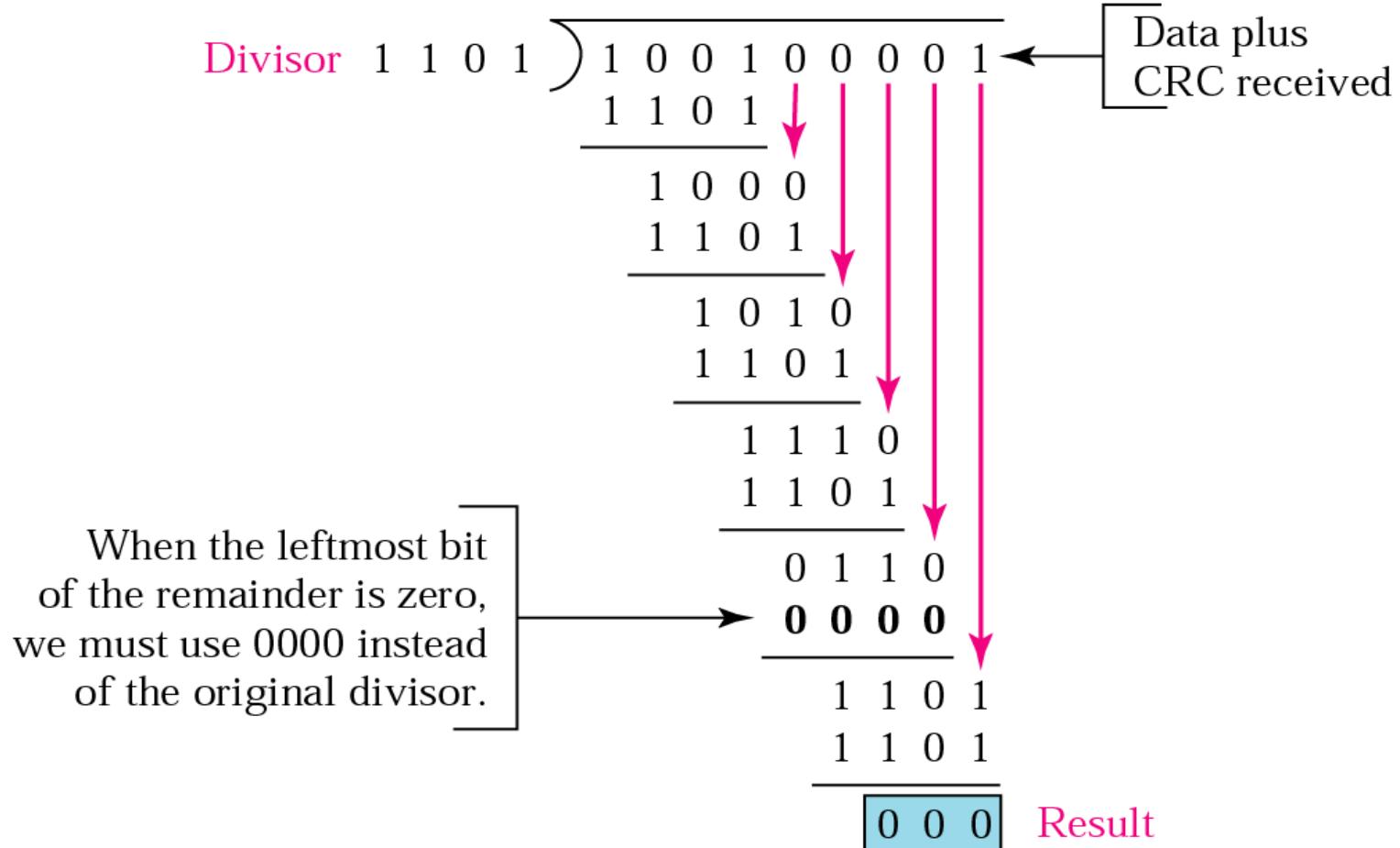
* Figure is courtesy of B. Forouzan

CRC: Sender



* Figure is courtesy of B. Forouzan

CRC: Receiver

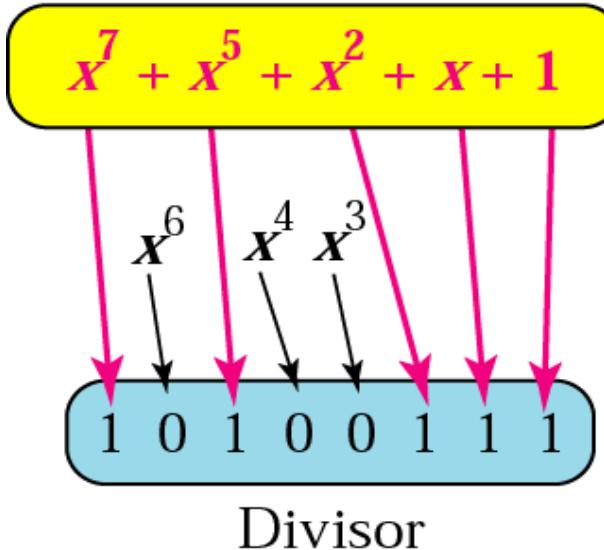


* Figure is courtesy of B. Forouzan



Polynomial Notation

Polynomial



- Rules for selecting divisor:
 - It should not be divisible by x
 - It should be divisible by $x+1$

* Figure is courtesy of B. Forouzan



Polynomials

- We cannot choose x (binary 10) or $x^2 + x$ (binary 110) as polynomial because both are divisible by x .
- However, we can choose $x + 1$ (binary 11) because it is not divisible by x , but is divisible by $x + 1$. We can also choose $x^2 + 1$ (binary 101) because it is divisible by $x + 1$ (binary division).

* Figure is courtesy of B. Forouzan

Standard Polynomials

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

* Figure is courtesy of B. Forouzan

CRC Performance

- Can detect all burst errors that effect an odd number of bits
- Can detect all burst errors of the length less than or equal to the degree of the polynomial
- Can detect with a very high probability burst errors of a length greater than the degree of the polynomial.



CRC-12 Performance

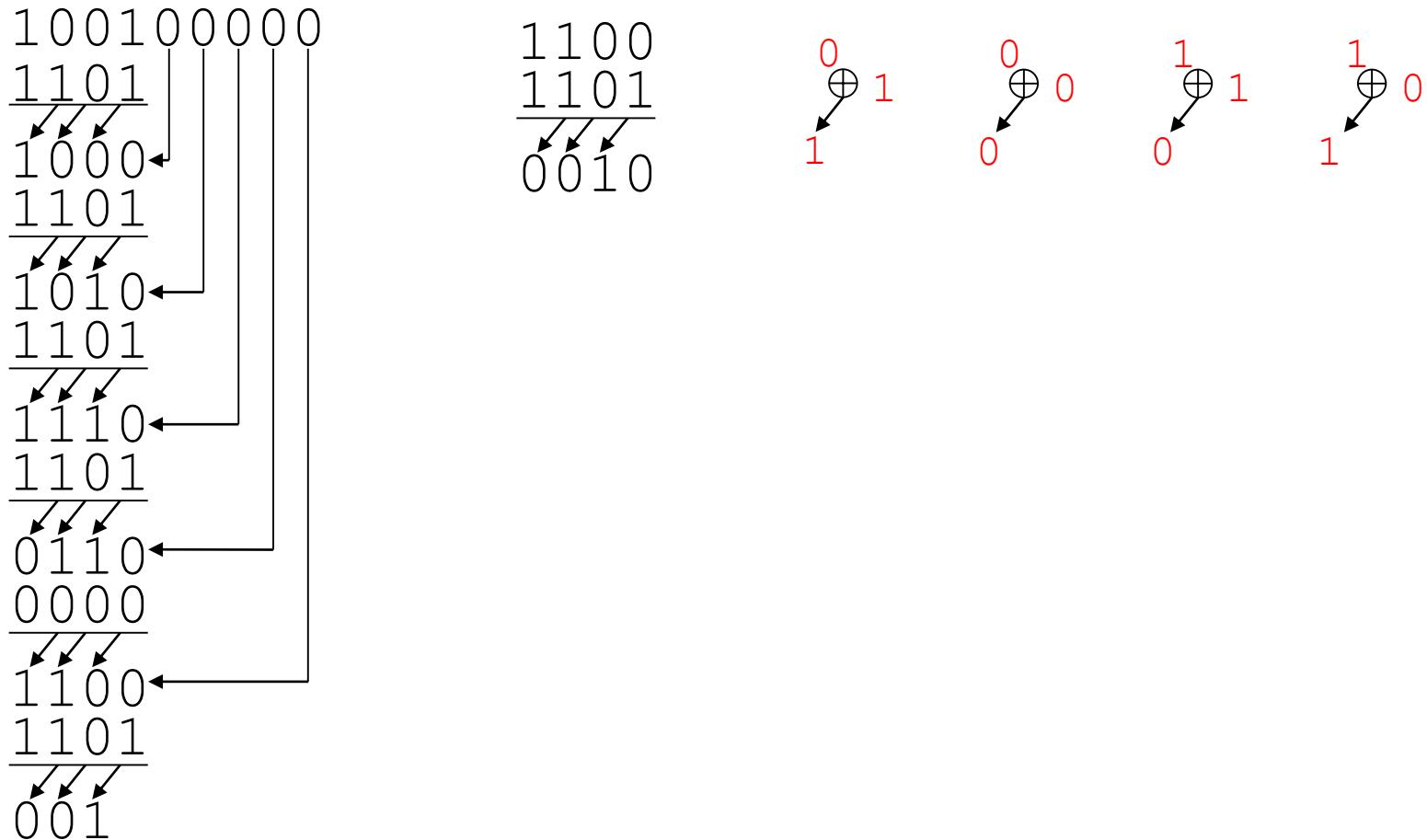
The CRC-12

$$x^{12} + x^{11} + x^3 + x + 1$$

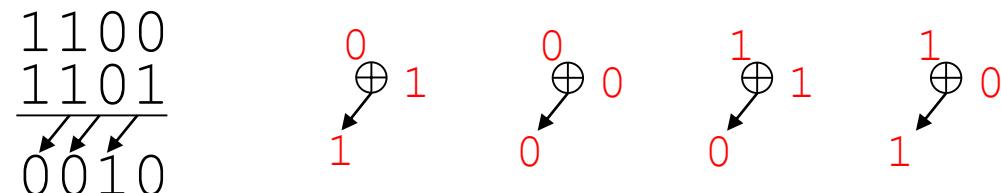
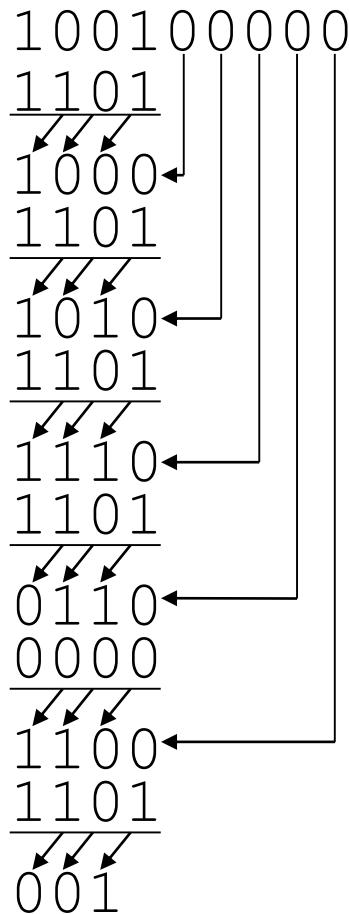
which has a degree of 12, will detect all burst errors affecting an odd number of bits, will detect all burst errors with a length less than or equal to 12, and will detect, 99.97 percent of the time, burst errors with a length of 12 or more.



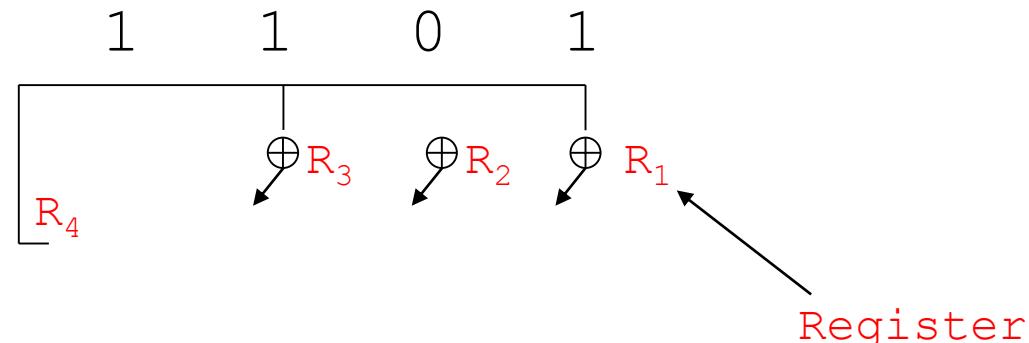
CRC Calculation



CRC Calculation



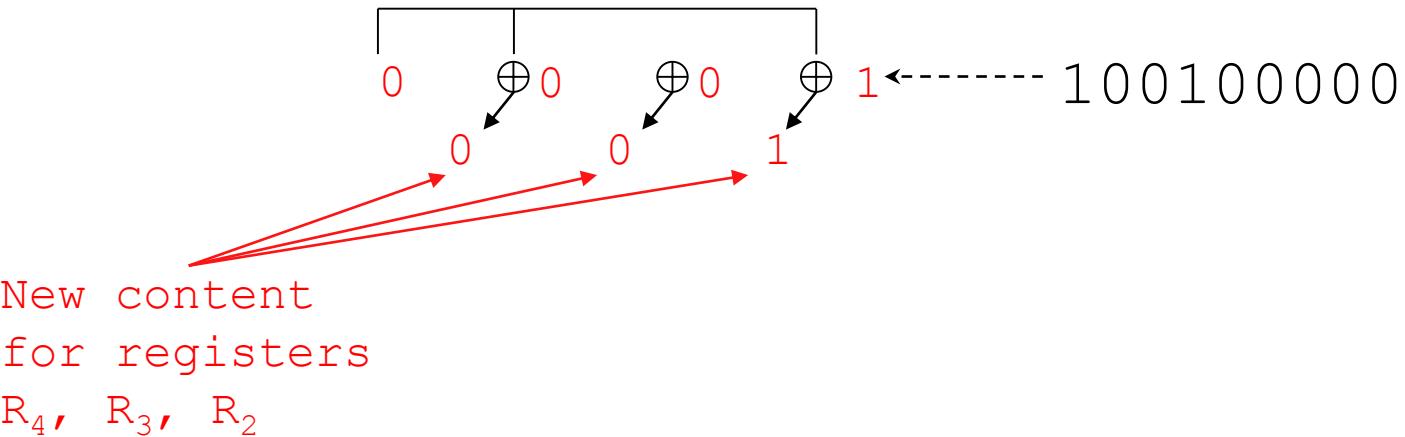
Representation of divisor:



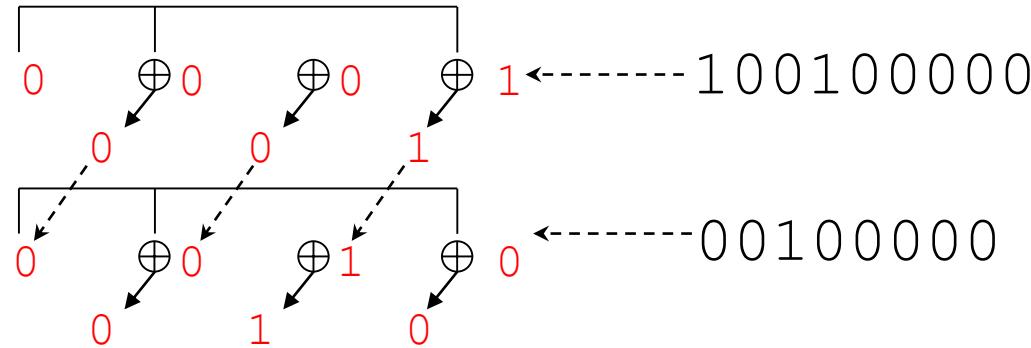
Register



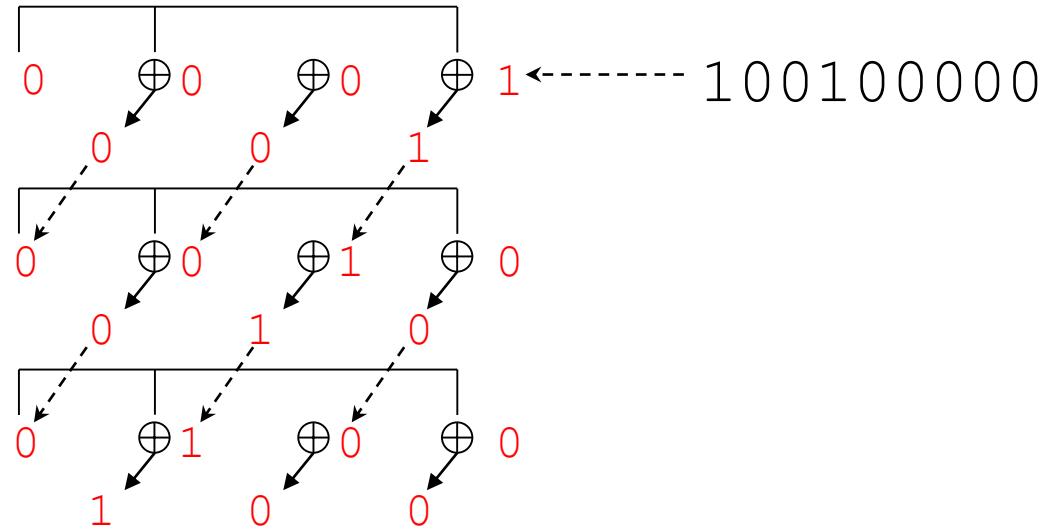
CRC Calculation



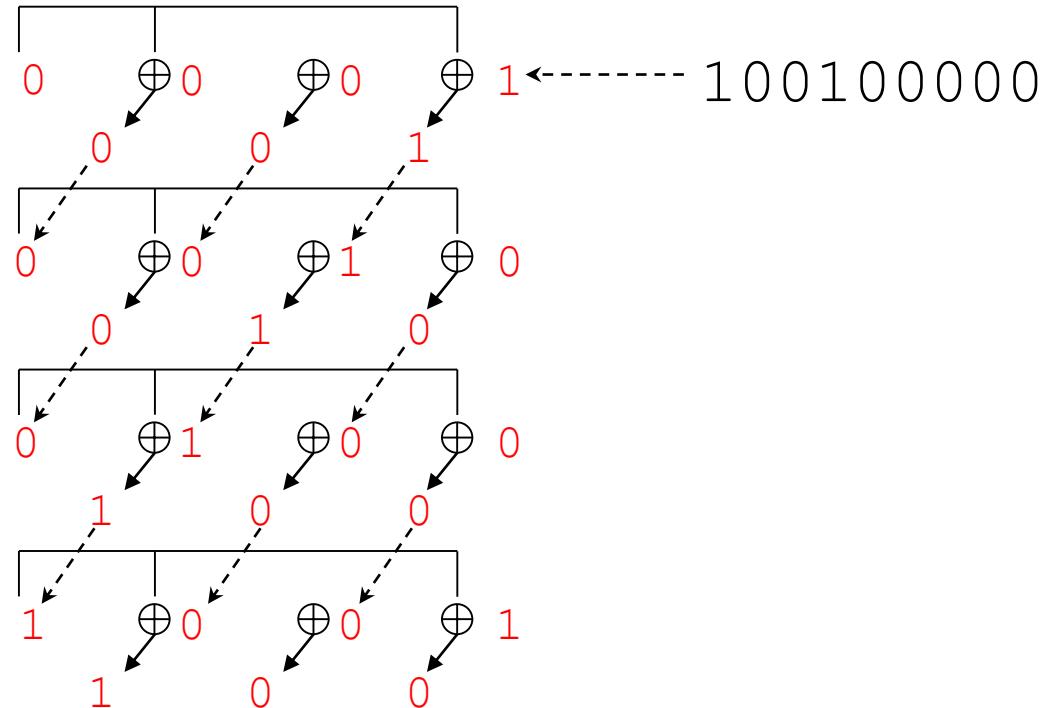
CRC Calculation



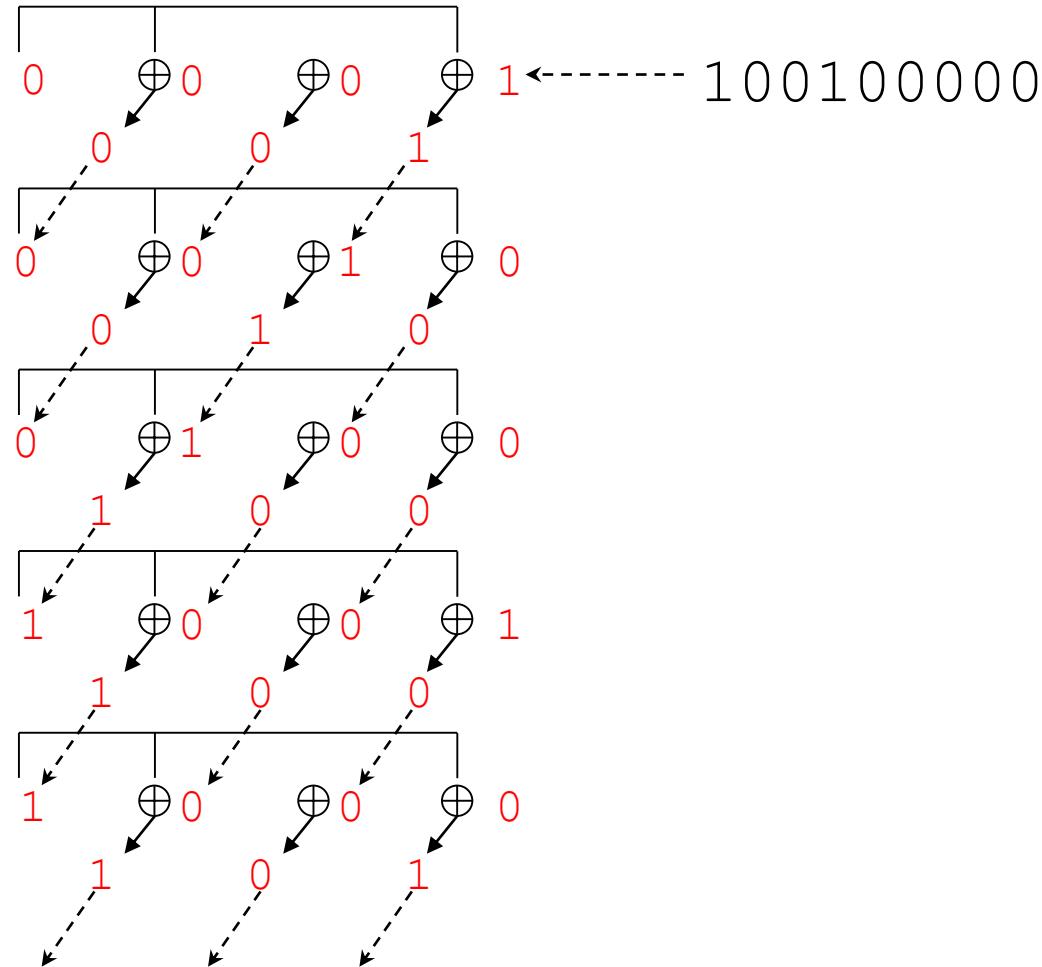
CRC Calculation



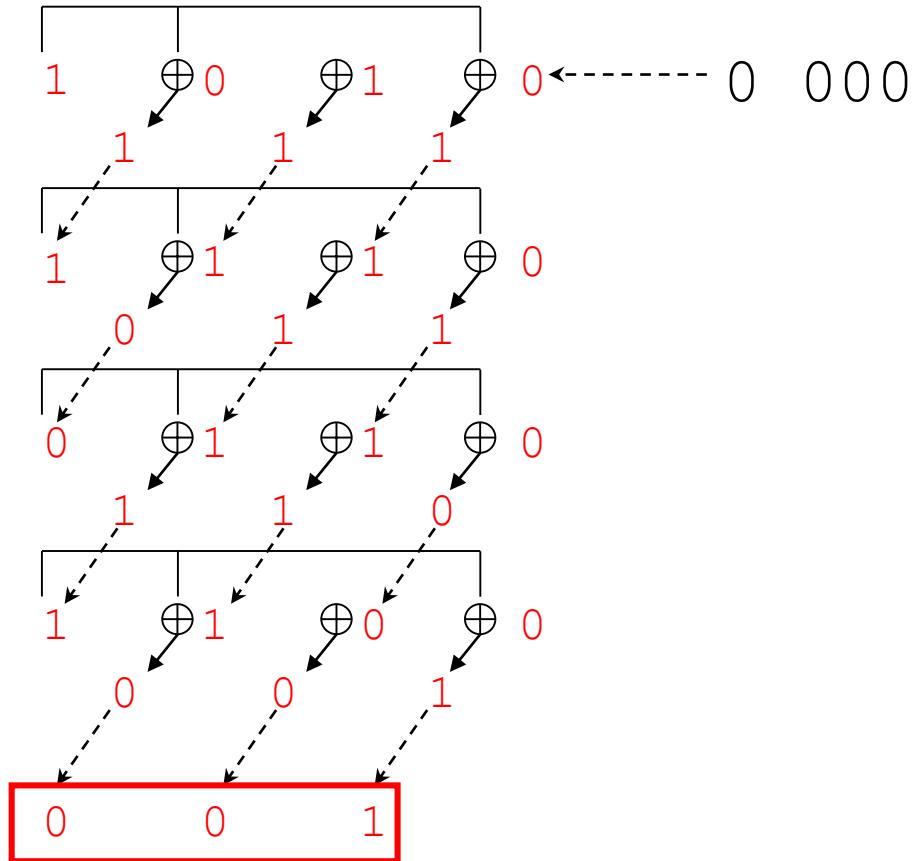
CRC Calculation



CRC Calculation



CRC Calculation

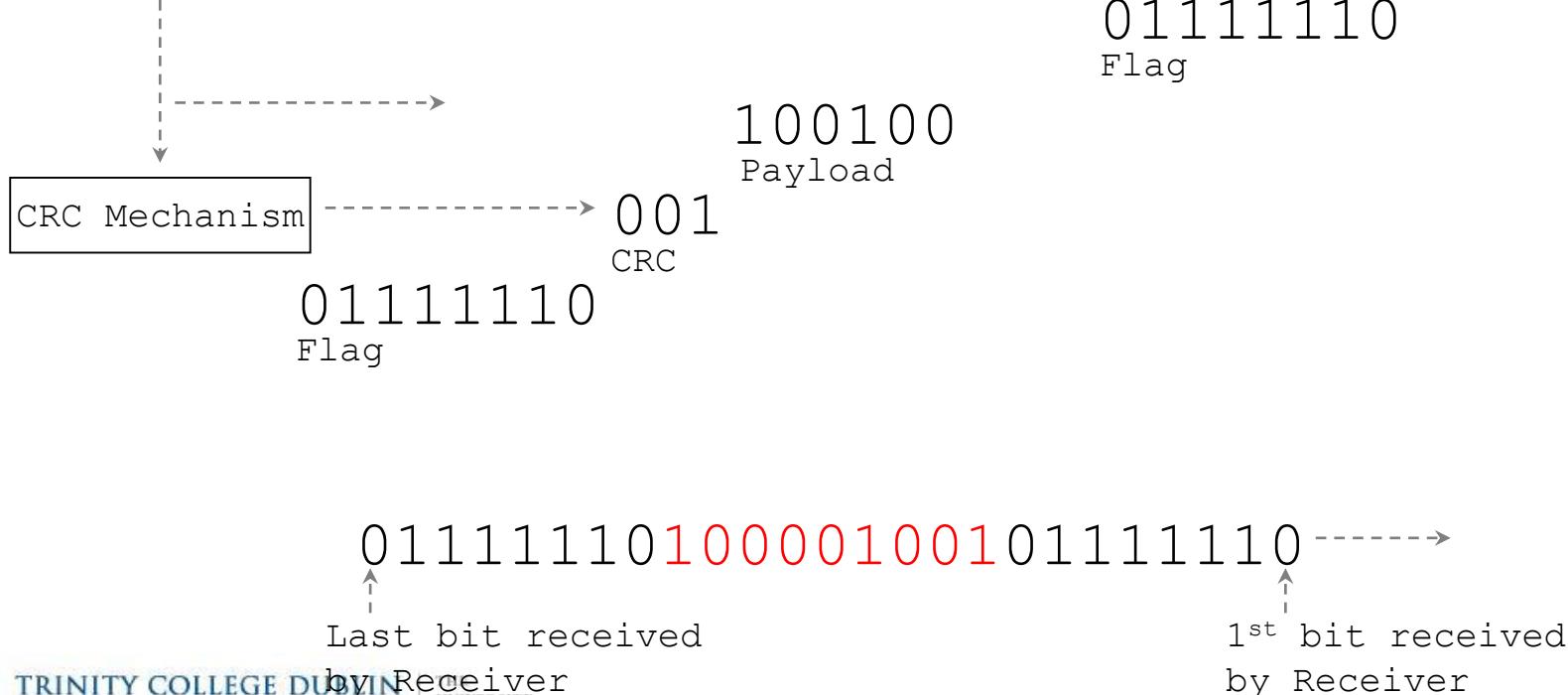


CRC Calculation

Sender

Payload + 3*0s for remainder
100100000

Receiver

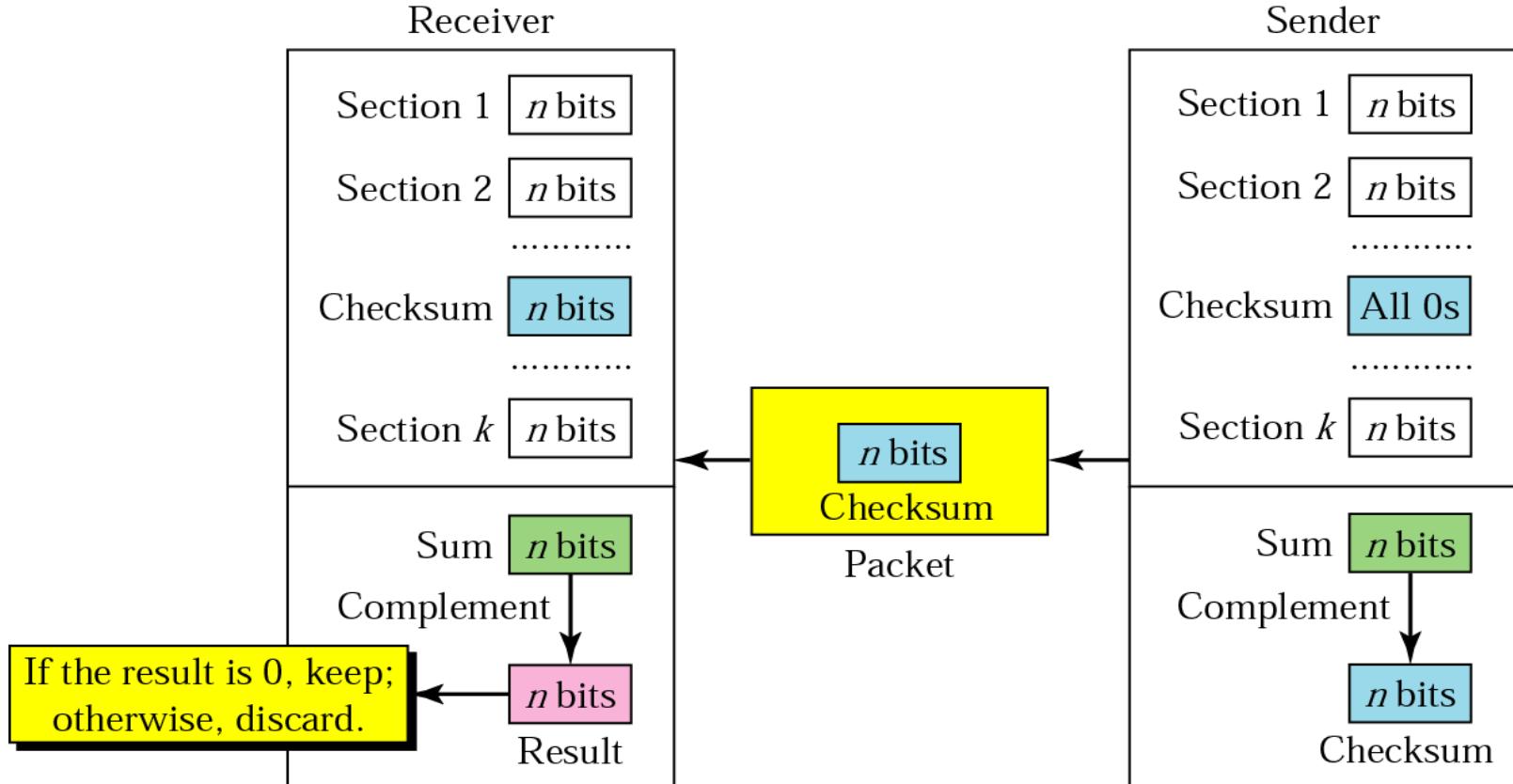


Example from a Linux box

```
wlan0      Link encap:Ethernet HWaddr 00:0b:81:89:56:ca  
          inet addr:192.168.192.12 Bcast:192.168.192.255 Mask:255.255.255.0  
          inet6 addr: fe80::20b:81ff:fe89:56ca/64 Scope:Link  
             UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
             RX packets:292 errors:0 dropped:374 overruns:0 frame:0  
             TX packets:199 errors:0 dropped:2 overruns:0 carrier:0  
             collisions:0 txqueuelen:1000  
             RX bytes:47787 (46.6 KiB) TX bytes:26749 (26.1 KiB)
```



Checksum



* Figure is courtesy of B. Forouzan

Checksum II

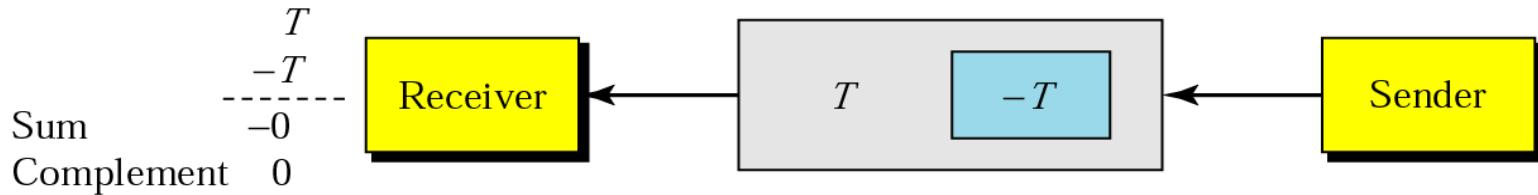
Sender:

The unit is divided into k sections, each of n bits.

All sections are added using one's complement to get the sum.

The sum is complemented and becomes the checksum.

The checksum is sent with the data.



Receiver:

The unit is divided into k sections, each of n bits.

All sections are added using one's complement to get the sum.

The sum is complemented.

If the result is zero, the data are accepted: otherwise, rejected.

* Figure is courtesy of B. Forouzan



Example: Checksum

Sender:

10101001

00111001

Sum 11100010

Checksum **00011101**

The data that is send:

10101001 00111001 **00011101**



Example: Checksum

Sender:

10101001

00111001

Sum

11100010

Checksum

00011101

Receiver:

10101001

00111001

00011101

Sum

11111111

Complement

00000000

The data that is send:

10101001 00111001 **00011101**

Complement: **00000000**

means that the frame is OK.



Example: Checksum

Sender:

10101001

00111001

Sum

11100010

Checksum

00011101

Receiver:

10101111

11111001

00011101

Partial Sum

1 11000101

Carry

1

Sum

11000110

Complement

00111001

The data that is send:

10101001 00111001 **00011101**

Complement: **00111001**

means that the frame is corrupted.

Sample TCP / IP Packet

0000	00 07 e9 7c 22 fc 00 11 93 85 e0 c4 08 00	45 00
0010	00 2c db 26 40 00 3f 0e 77 86 e2 20 37 86 e2	
0020	24 33 01 bd 12 3f 3d fa 0f b6 a8 6f 87 c0 50 18	
0030	bc 40 8a 7c 00 00 85 00 00 00 00 00	

Ethernet Header:

src addr: 00 07 e9 7c 22 fc

dest addr: 00 11 93 85 e0 c4

IP Header:

src addr: 134.226.36.55

dest addr: 134.226.36.51

TCP Header:

src port: 445

dest port: 4671

NetBios Information

... | ".....E.

., .&@.?..w.. 7..

\$3....?=....o...P.

.@. |

IP Header Checksum

The IP header is generally 20 byte and can be divided into units of 2 bytes/16 bits to calculate the checksum



Summary: Detection of Errors

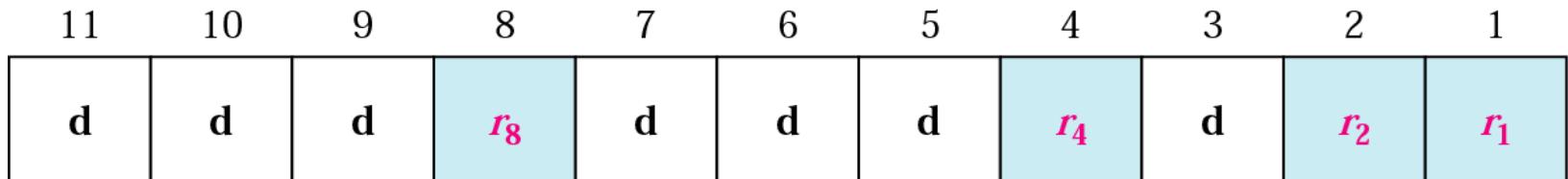
- Parity Check
- Cyclic Redundancy Check (CRC)
- Checksum



Correction of Errors

- Error Correction through Retransmission
 - Parity, CRC, Checksum determine validity
 - If not valid, discard and wait for sender to retransmit
- Forward Error Correction
 - Determine the corrupted bit or bits at the receiver

Hamming Code

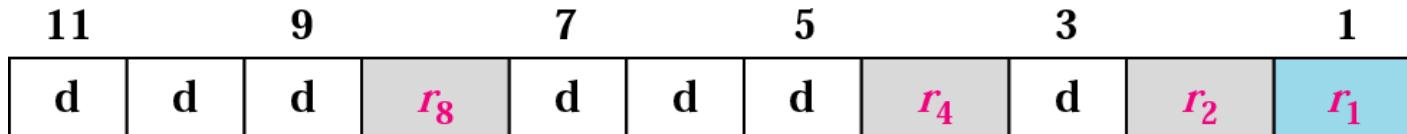


- Redundancy bits distributed throughout data bits
- Individual redundancy bits work as parity bits for specific data bits
 - e.g. r_1 is the parity bit for all odd numbers
 - 3 = binary 001 $\textcolor{red}{1}$
 - 5 = binary 010 $\textcolor{red}{1}$
 - 7= binary 011 $\textcolor{red}{1}$
 - 9= binary 100 $\textcolor{red}{1}$

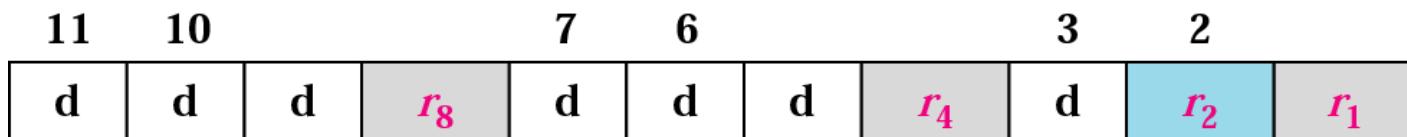
* Figure is courtesy of B. Forouzan

Redundancy Bits Calculation

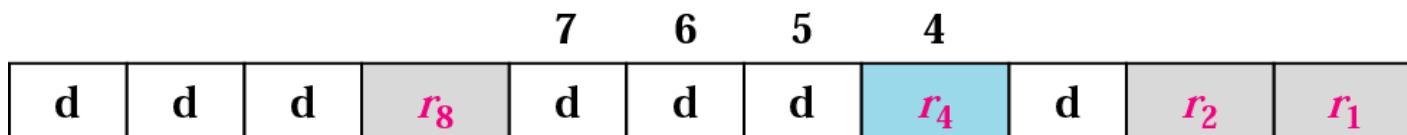
r_1 will take care of these bits.



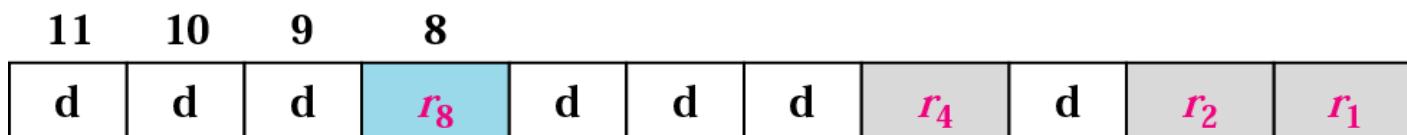
r_2 will take care of these bits.



r_4 will take care of these bits.



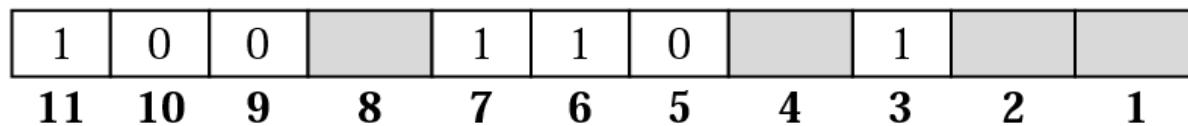
r_8 will take care of these bits.



* Figure is courtesy of B. Forouzan



Redundancy Bit Calculation

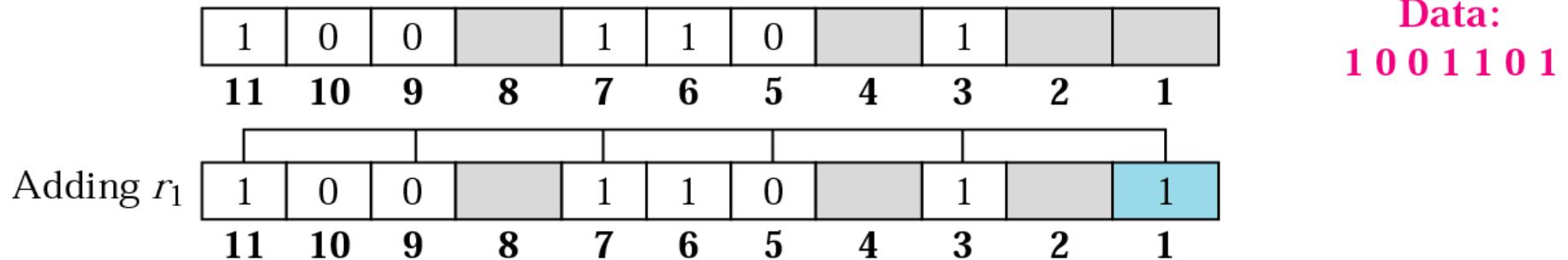


Data:
1 0 0 1 1 0 1

* Figure is courtesy of B. Forouzan



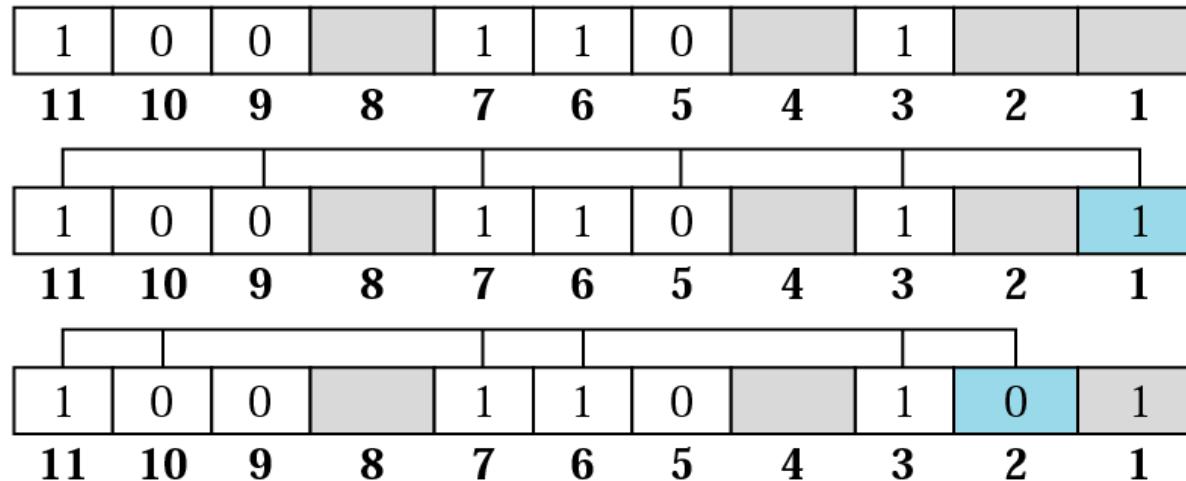
Redundancy Bit Calculation



* Figure is courtesy of B. Forouzan



Redundancy Bit Calculation



* Figure is courtesy of B. Forouzan

Redundancy Bit Calculation

1	0	0		1	1	0		1		
11	10	9	8	7	6	5	4	3	2	1

Data:
1 0 0 1 1 0 1

Adding r_1

1	0	0		1	1	0		1		1
11	10	9	8	7	6	5	4	3	2	1

Adding r_2

1	0	0		1	1	0		1	0	1
11	10	9	8	7	6	5	4	3	2	1

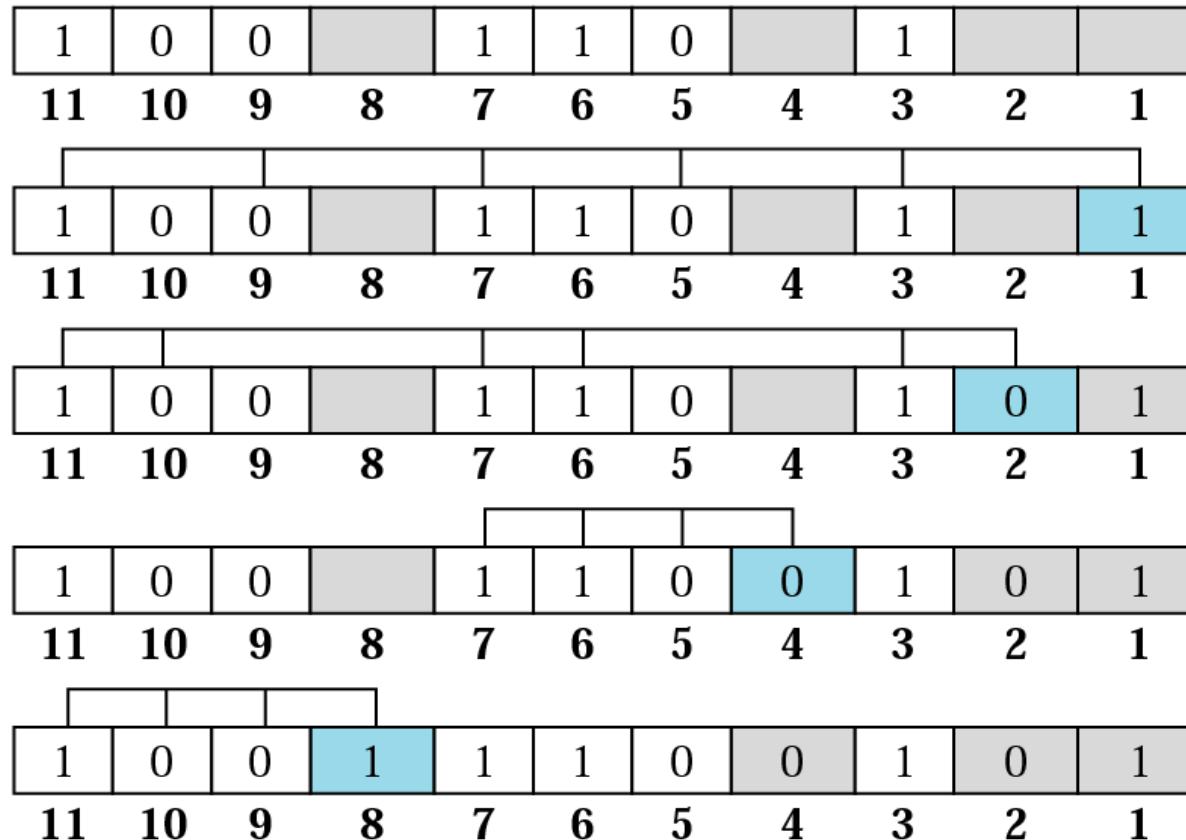
Adding r_4

1	0	0		1	1	0	0	1	0	1
11	10	9	8	7	6	5	4	3	2	1

* Figure is courtesy of B. Forouzan



Redundancy Bit Calculation



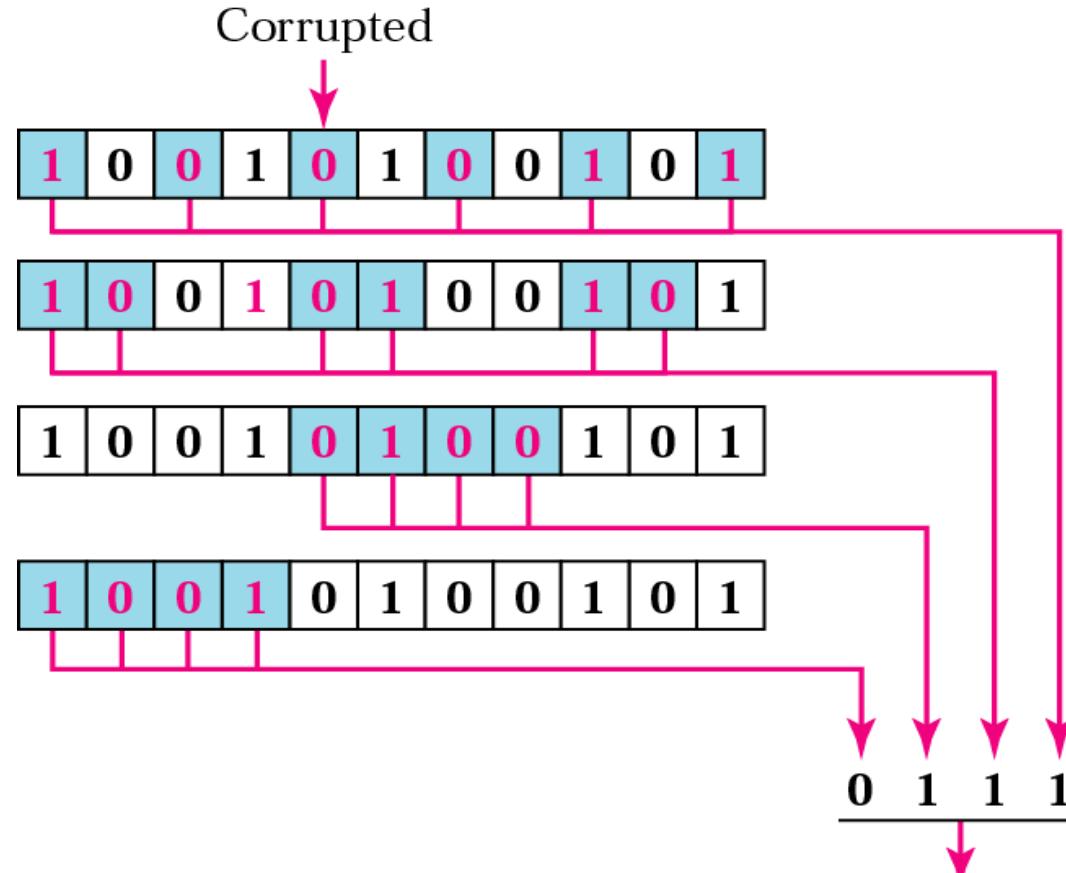
Data:
1 0 0 1 1 0 1

Code:
1 0 0 1 1 1 0 0 1 0 1

* Figure is courtesy of B. Forouzan



Error Detection using Hamming Code



The bit in position 7 is in error. 7

* Figure is courtesy of B. Forouzan

Data and Redundancy Bits

Number of data bits m	Number of redundancy bits r	Total bits $m + r$
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

* Figure is courtesy of B. Forouzan



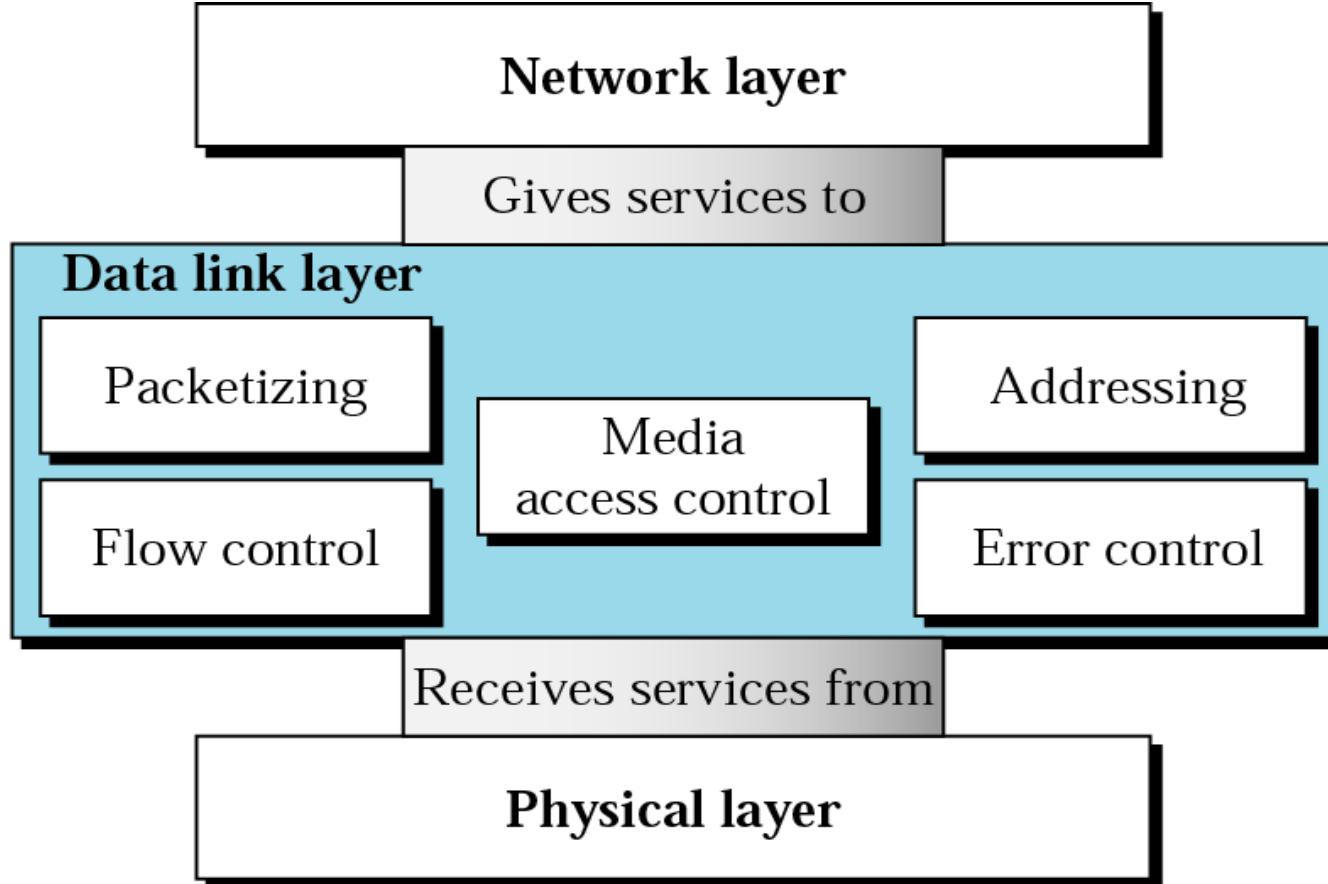
Summary

- Types of Errors
 - Single-Bit & Burst Errors
- Detection of Errors
 - Parity Check / 2D Parity Check
 - CRC
 - Checksum
- Correction of Errors
 - Error Correction by Retransmission
 - Forward Error Correction – Hamming Code





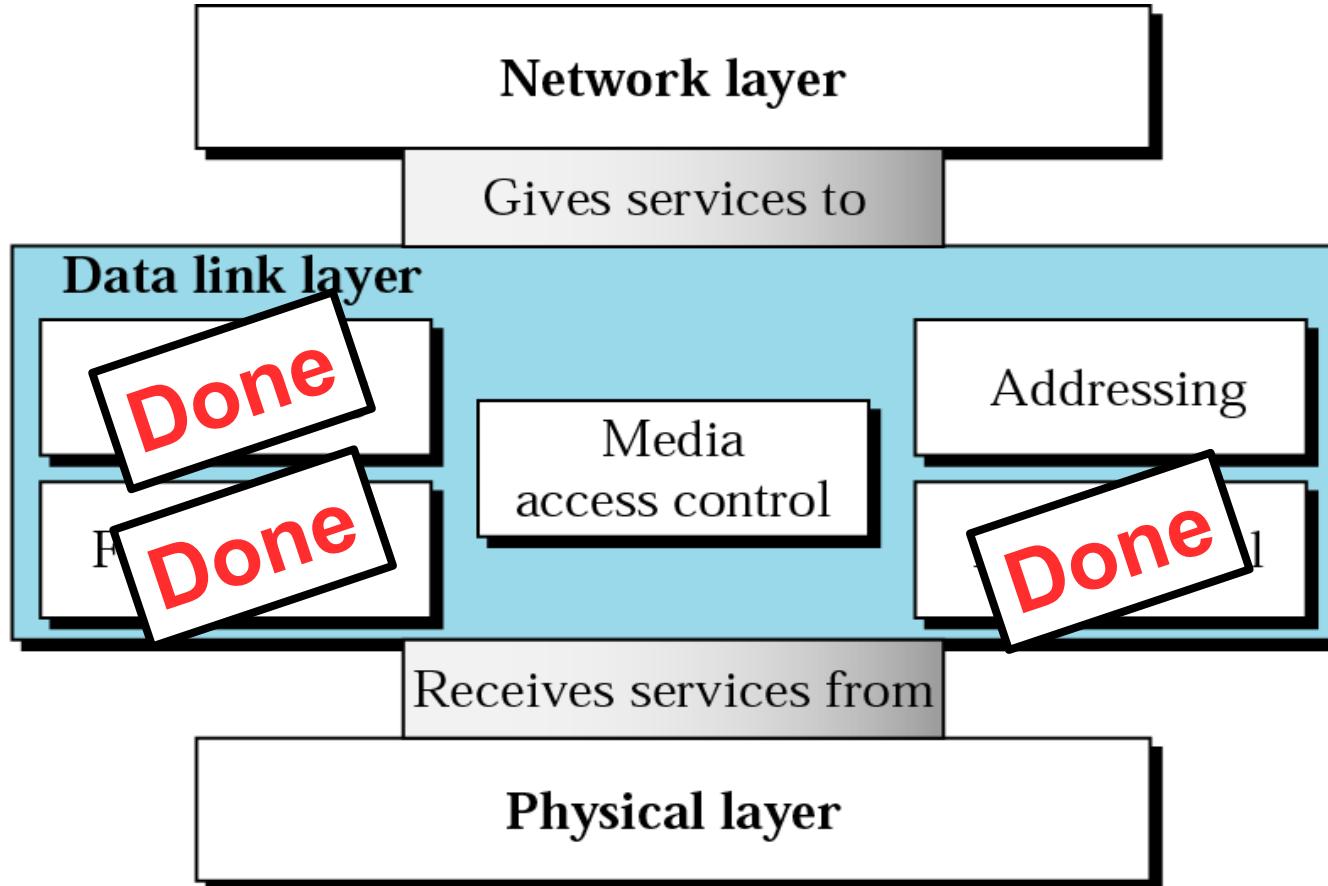
Link Layer



* Figure is courtesy of B. Forouzan



Link Layer

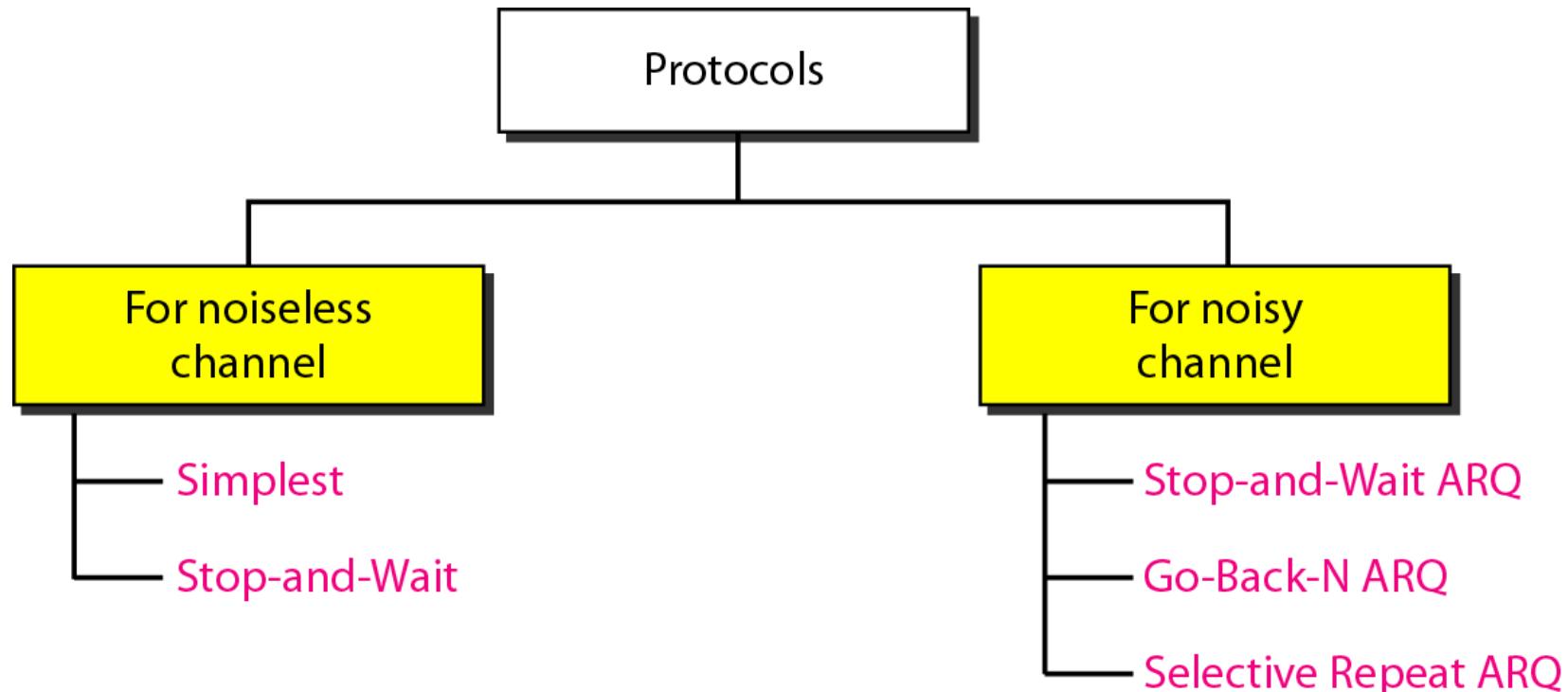


* Figure is courtesy of B. Forouzan



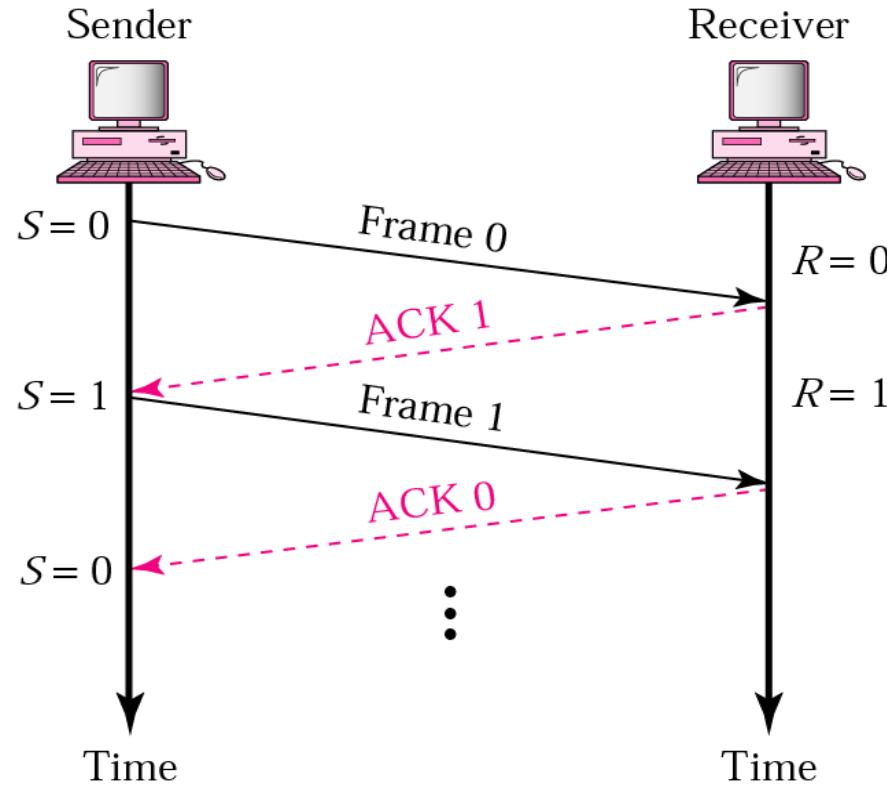
Review: Flow Control

Flow Control: Refers to the control of the amount of data that a sender can transmit **without overflowing the receiver.**



* Figure is courtesy of B. Forouzan

Stop-and-Wait ARQ

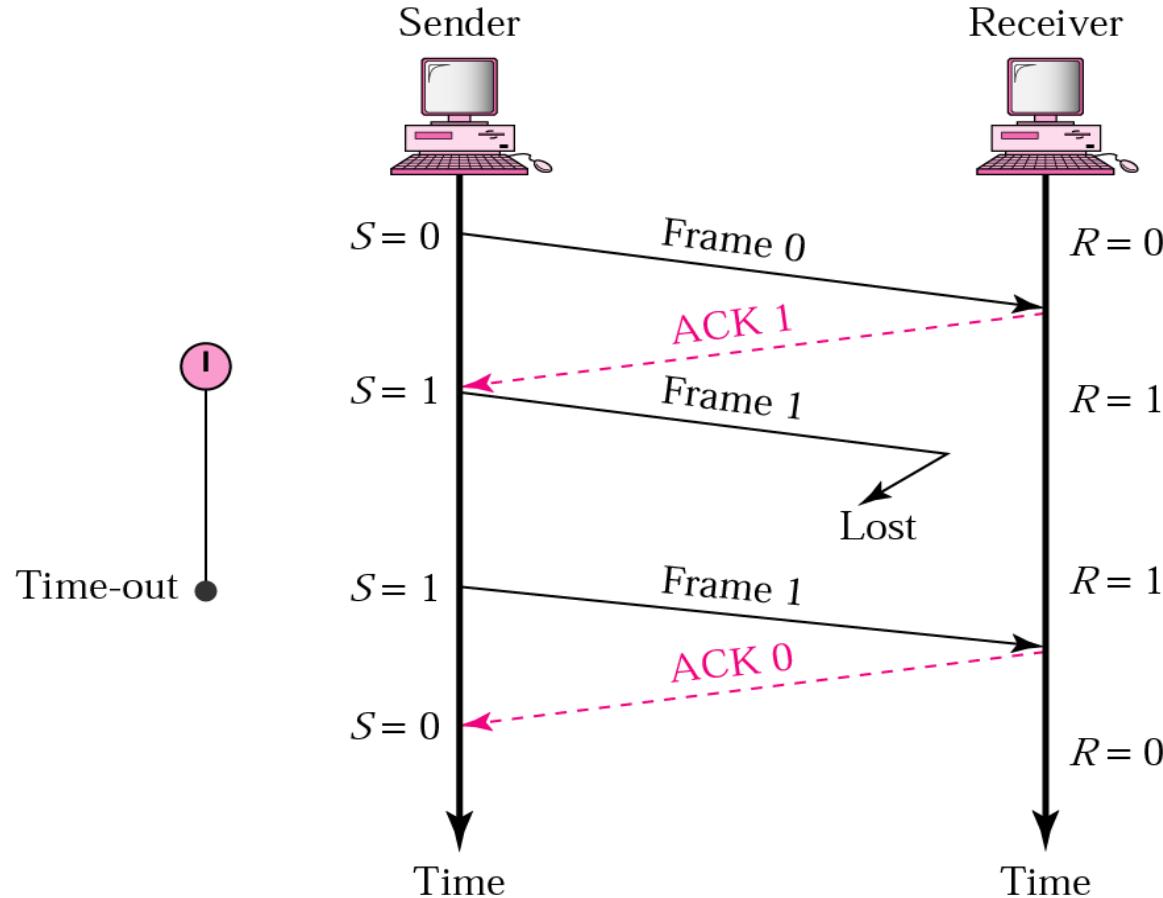


- ACK = received packet, ready to receive packet #

* Figure is courtesy of B. Forouzan

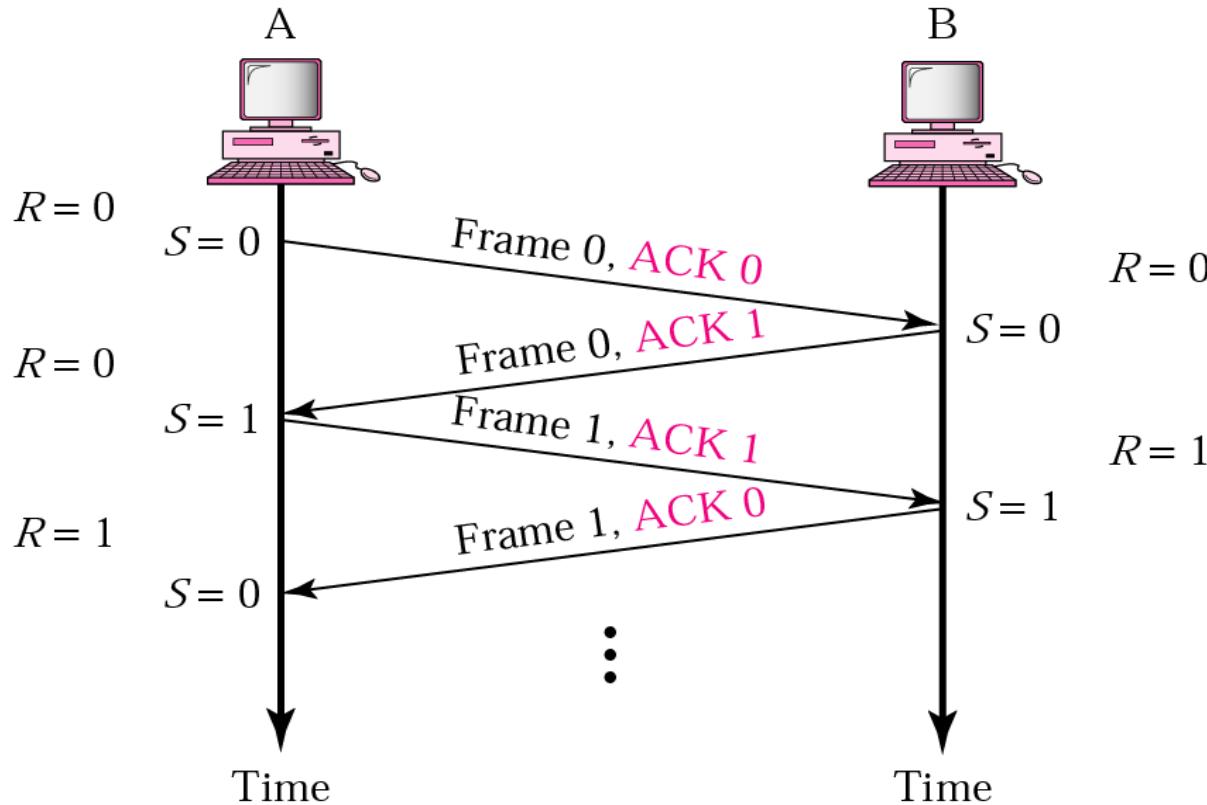
Stop-and-Wait ARQ: Time-Out

- Frame is lost during transmission



* Figure is courtesy of B. Forouzan

Piggybacking ACKs

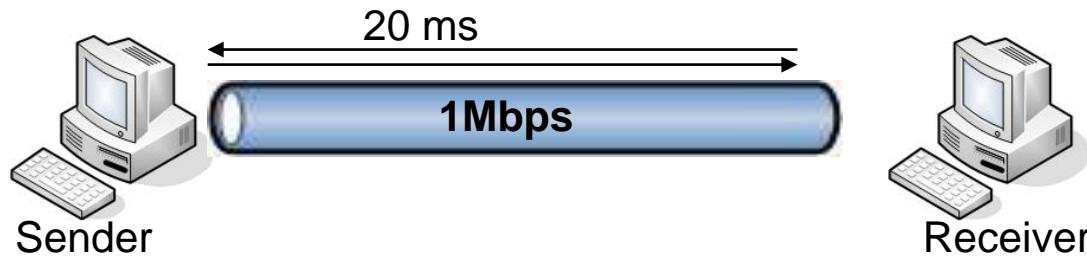


- Next data frame send carries acknowledgement for last frame received

* Figure is courtesy of B. Forouzan

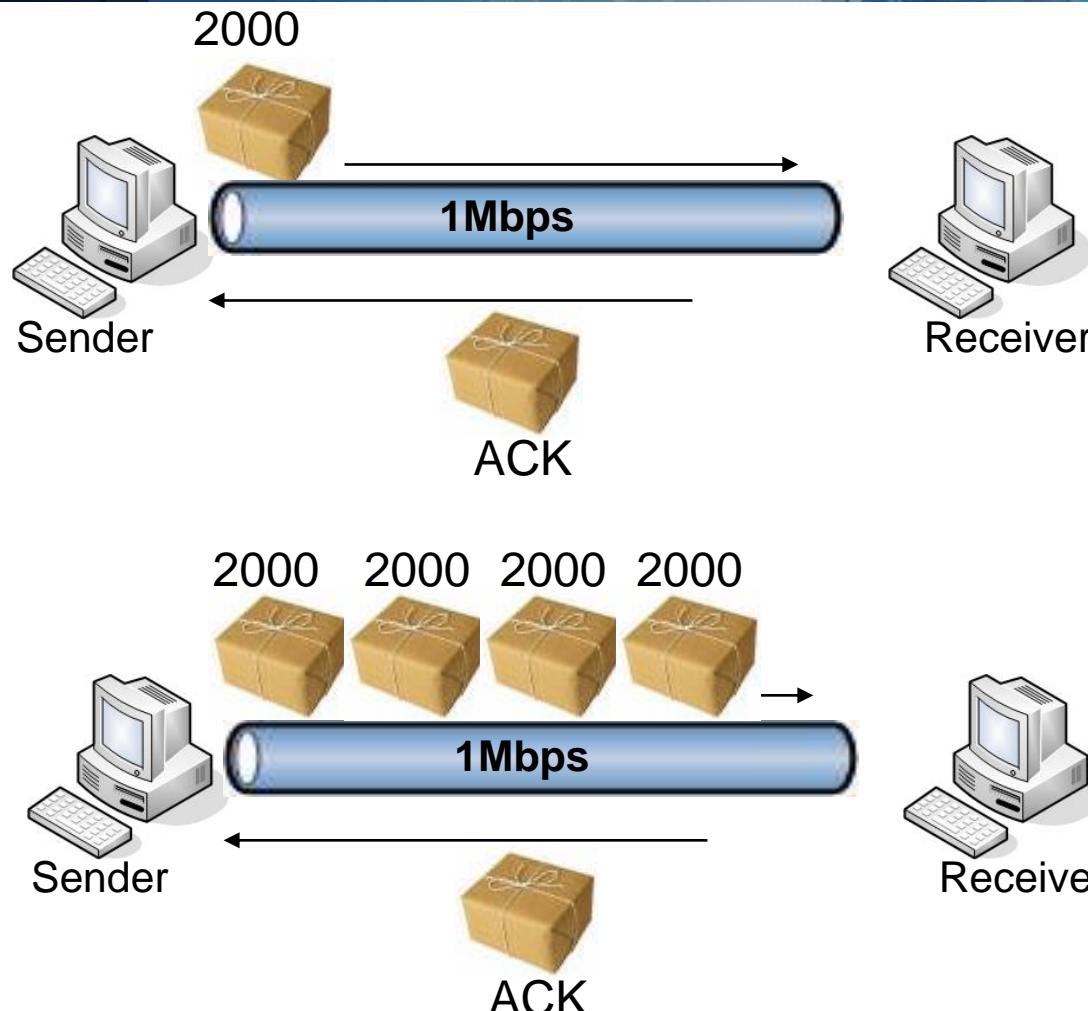
Bandwidth-Delay Product: Example

- Bandwidth × Round-Trip-Time
 - gives indication of amount of data that can be send while waiting for ACK

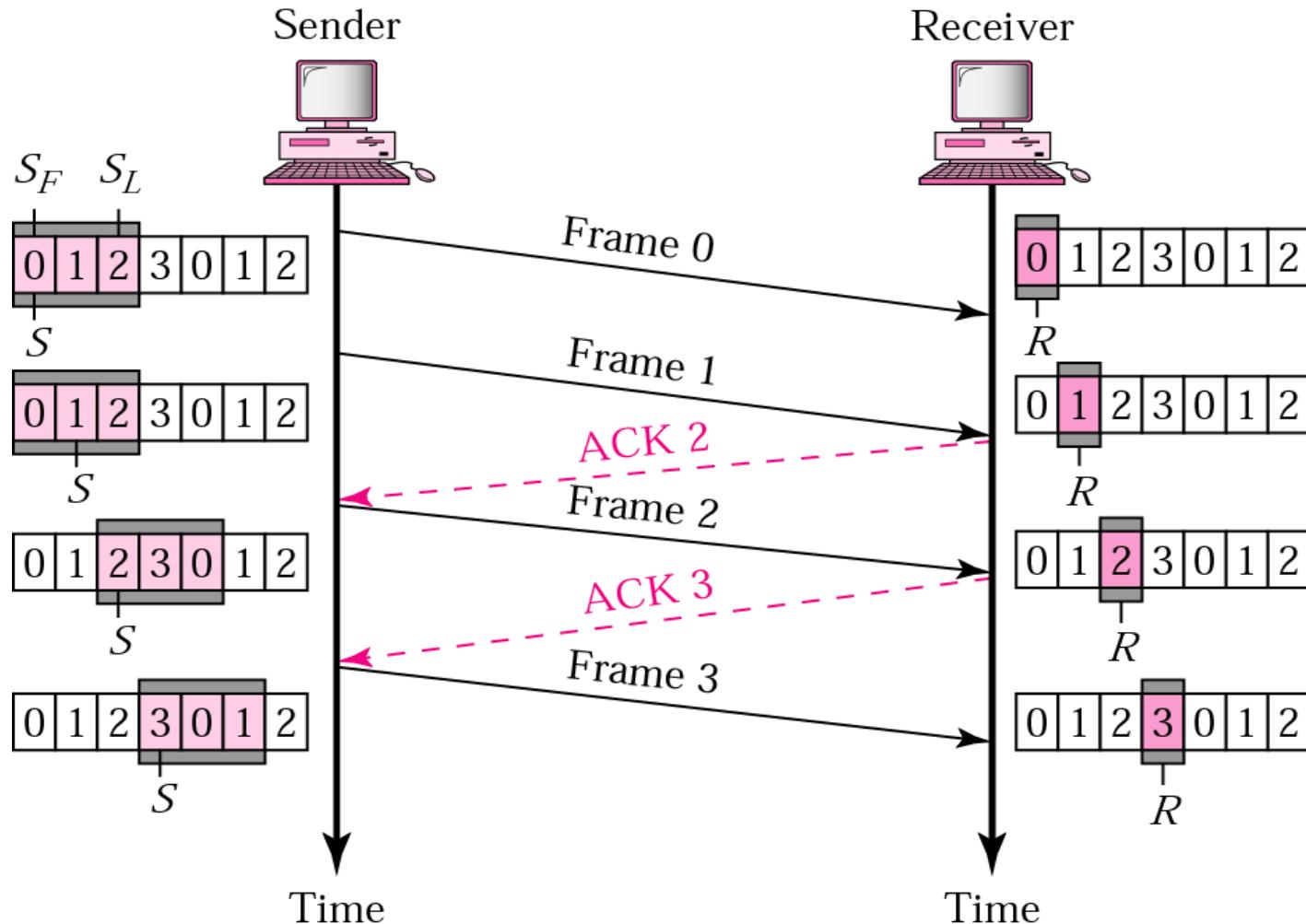


- Communication link with 1Mb/s
- Round-Trip time: $20 \text{ ms} = 20 * 10^{-3} \text{ s}$
- How much data can you send during the time it takes for 1 bit to travel?
 $20 * 10^{-3} \text{ s} * 1 * 10^6 \text{ b/s} = 20.000 \text{ bits}$
- Frame of 2000 bit $\Rightarrow 10\%$ of bandwidth used

Bandwidth-Delay Product: Example

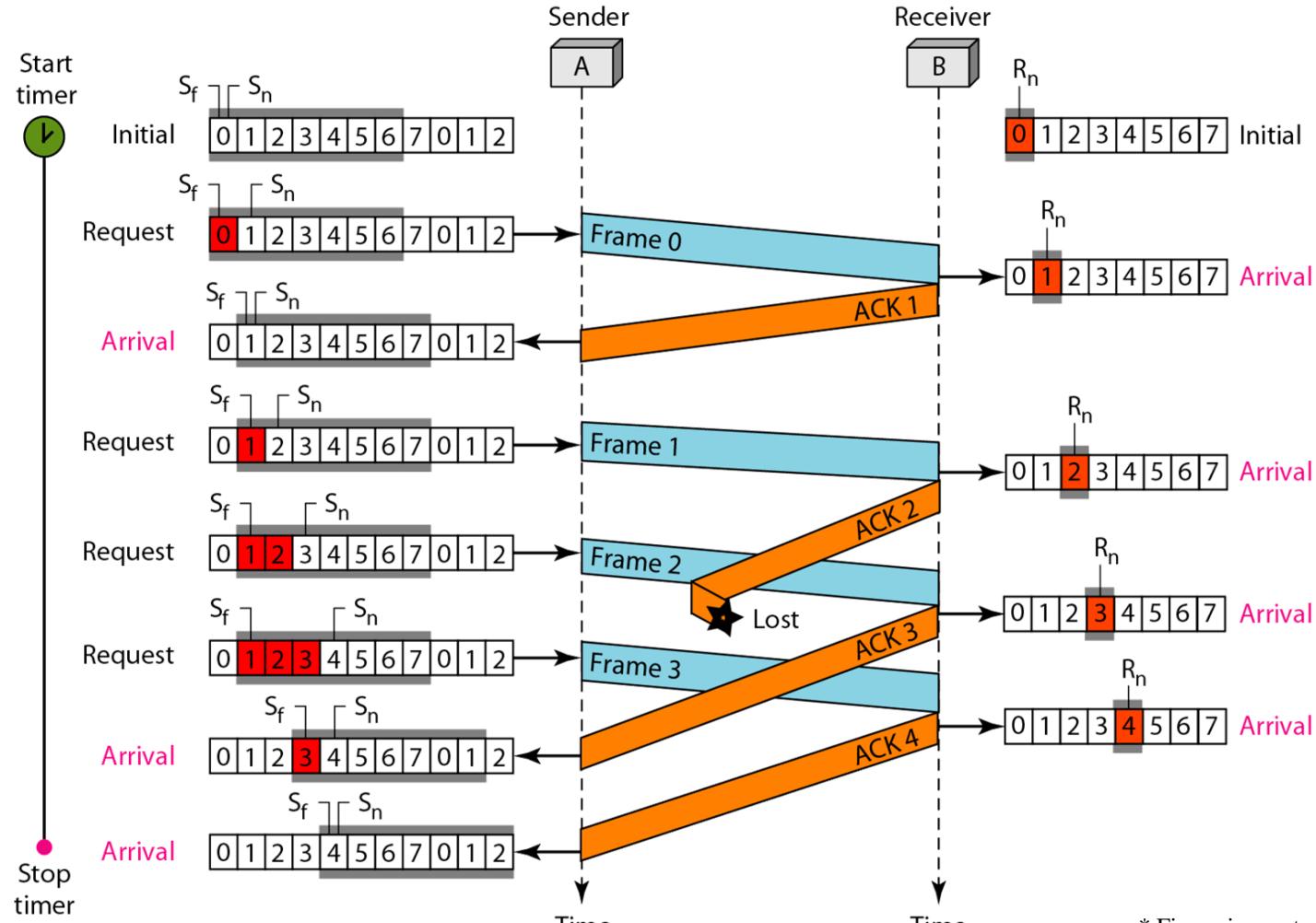


Go-Back-N ARQ



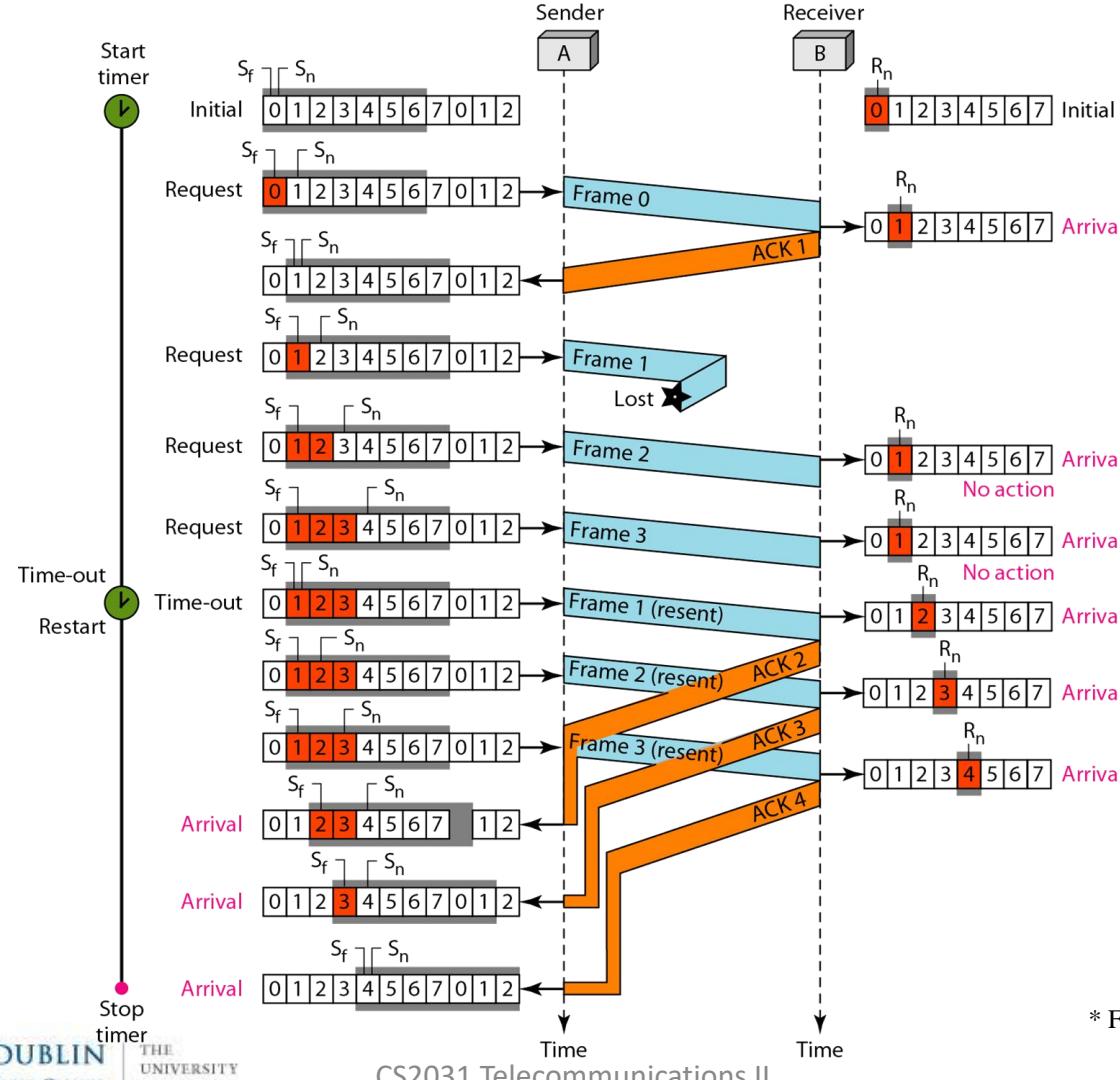
* Figure is courtesy of B. Forouzan

Go-Back-N: Lost ACK



* Figure is courtesy of B. Forouzan

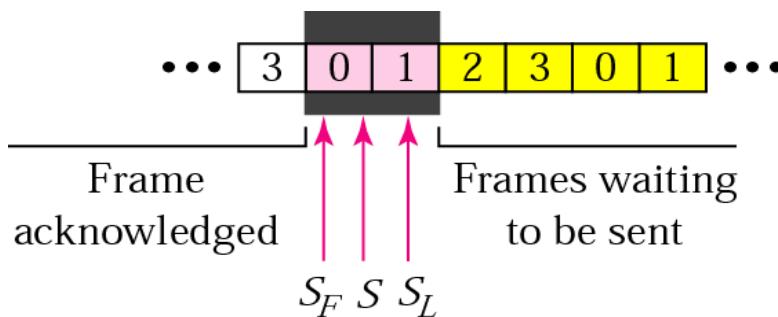
Go-Back-N ARQ: Bad Behaviour



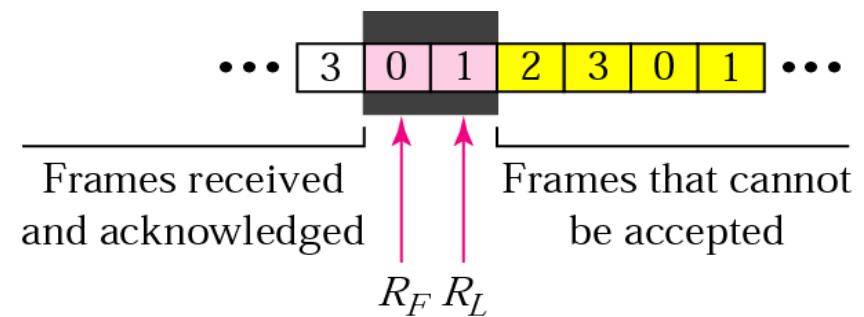
* Figure is courtesy of B. Forouzan

Selective Repeat

- Two Windows:
 - 1 Sender Window – 1 Receiver Window



a. Sender window

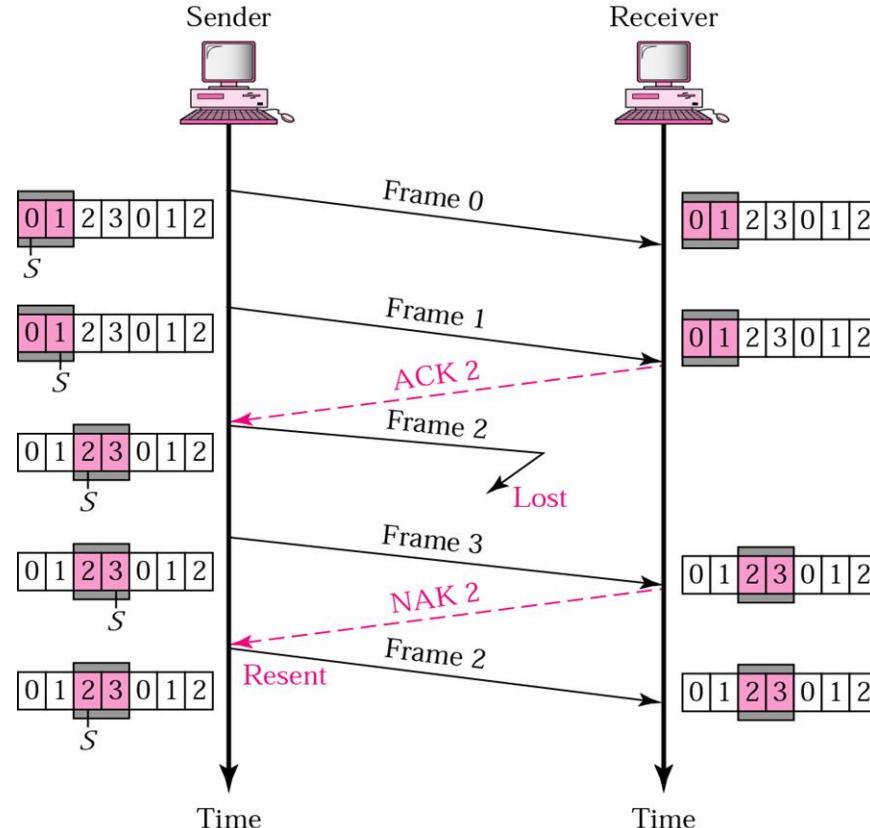


b. Receiver window

* Figure is courtesy of B. Forouzan



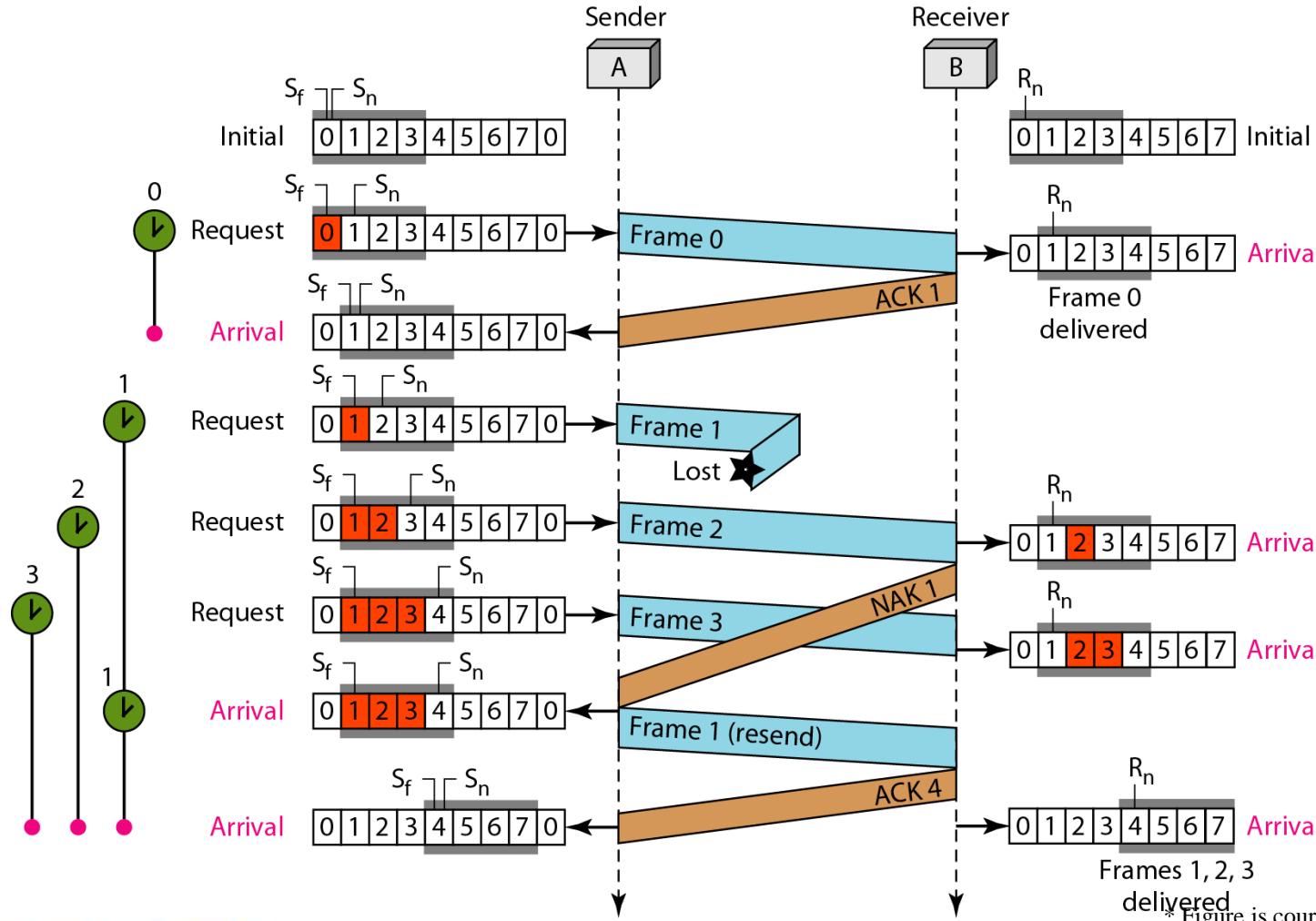
Selective Repeat ARQ: Lost Frame



- NAK = Negative Acknowledgement
- Sender still maintains timers for packets in case NAK gets lost

* Figure is courtesy of B. Forouzan

Selective Repeat ARQ



*Figure is courtesy of B. Forouzan

Sliding Windows

- Allow multiple frames to be in transit
- Receiver has buffer w long
- Transmitter can send up to w frames
 - without ACK
- Each frame is numbered
- ACK includes number of next frame expected



Window Size for Go-Back-N

- Depends on size of max. frame number
 - Frame # needs to be included in every frame
 - e.g. 4 bits – $2^4 = 16$ frame numbers
- Trade-off between window size and frame size



CS2031 Telecommunications II

High-level Data Link Control (HDLC)



HDLC

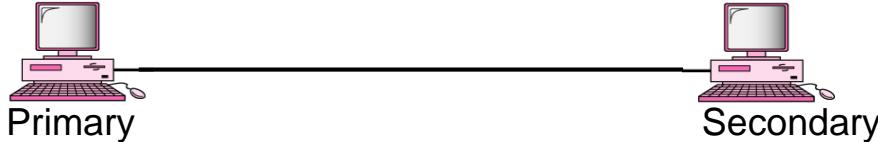
- ISO 33009, ISO 4335, Used initially in X.25
1979, ISO 3309
- It's old – so, why should we care?
 - Implements framing, addressing
 - Implements flow control mechanisms
- Do we have to learn it by heart?
 - **No** – learn the principles – not the frames layout!

HDLC Station Types

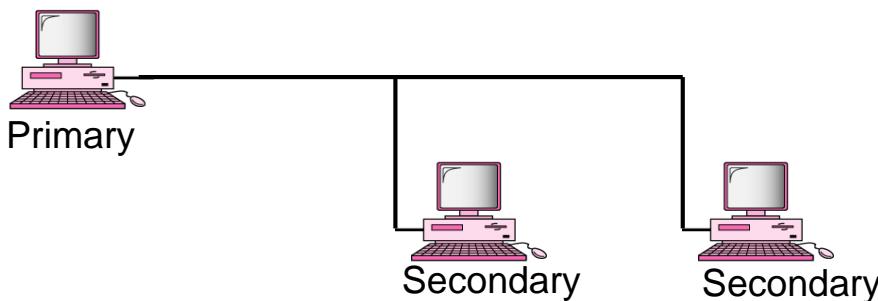
- Primary station
 - Controls operation of link
 - Frames issued are called commands
- Secondary station
 - Under control of primary station
 - Frames issued called responses
- Combined station
 - Combination of primary and secondary station
 - May issue commands and responses



HDLC Station Types

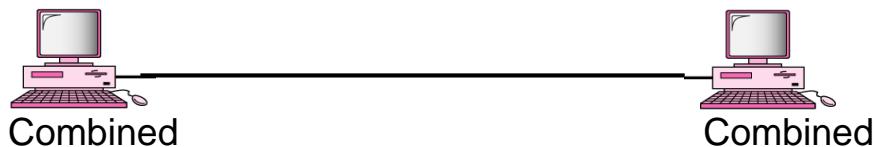


Unbalanced Configuration



Two types of stations:

- Primary station
- Secondary station



Balanced Configuration

One type of stations:
• Combined station



HDLC Modes

- Three modes:
 - Normal Response Mode (NRM)
 - Asynchronous Response Mode (ARM)
 - Asynchronous Balanced Mode (ABM)



Normal Response Mode (NRM)

Primary



Command

Secondary



Response

a. Point-to-point

Primary



Command

Secondary



Response

Secondary



Response

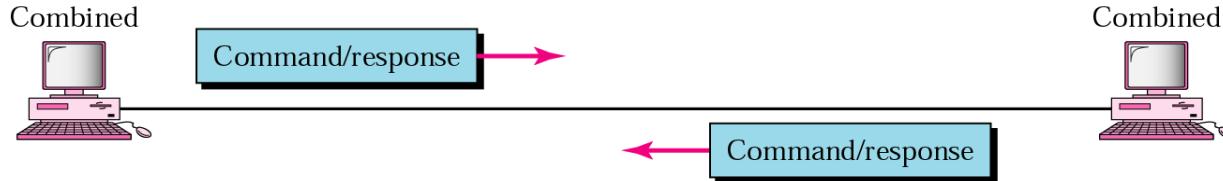
b. Multipoint

- Master/Slave architecture
- Unbalanced configuration
- Primary initiates transfer to secondary
- Secondary may only transmit data in response to command from primary
- Used on multi-drop lines

* Figure is courtesy of B. Forouzan



Asynchronous Balanced Mode (ABM)

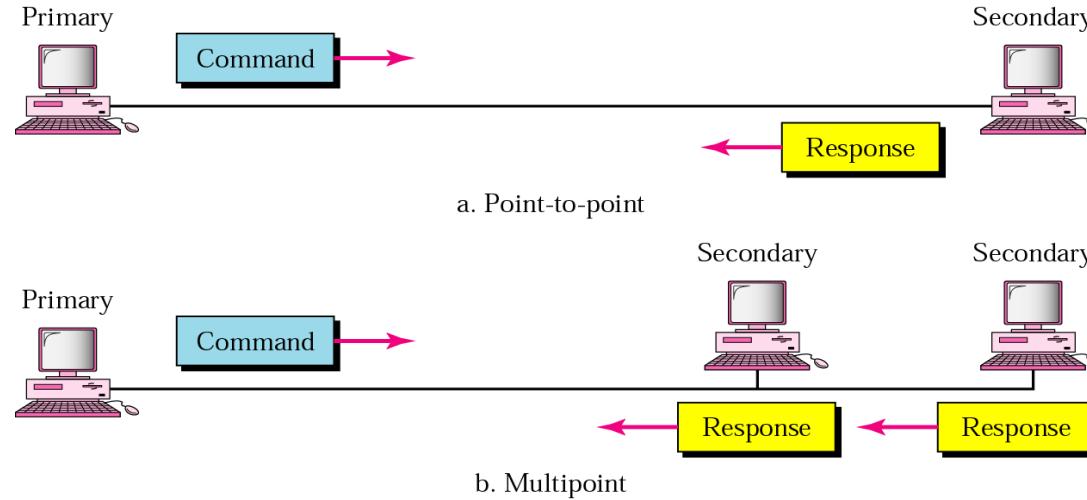


- Balanced configuration
- Either station may initiate transmission without receiving permission
- Most widely used
- No polling overhead

* Figure is courtesy of B. Forouzan



Asynchronous Response Mode (ARM)

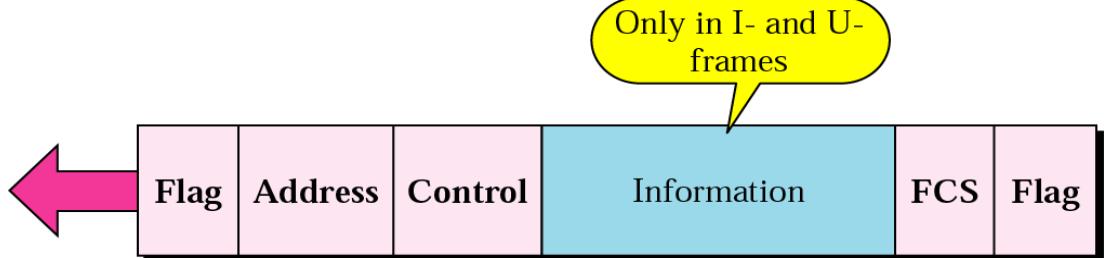


- Unbalanced configuration
- Secondary may initiate transmission without permission from primary
- Primary responsible for line
- Rarely used

* Figure is courtesy of B. Forouzan



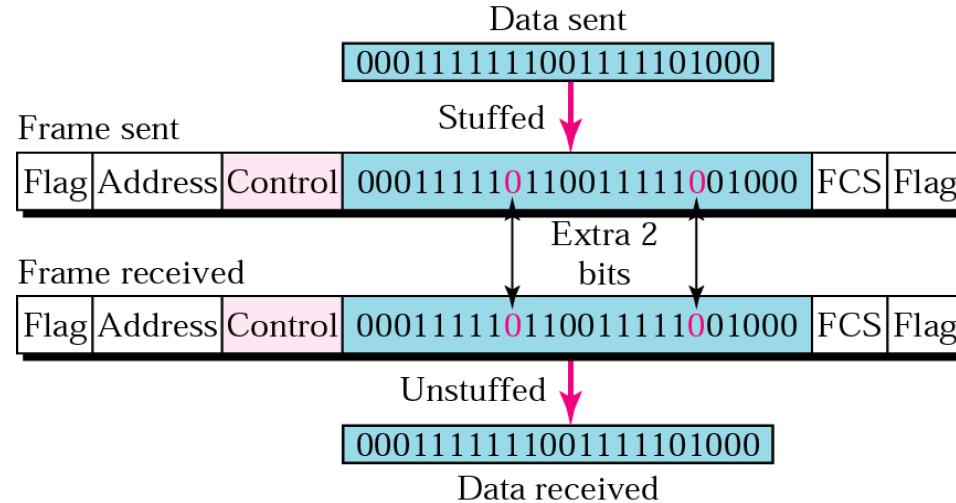
HDLC frame

- Flag= 01111110
 - specifies beginning and end of frame
 - Address
 - specifies secondary station
 - as either sender or receiver
 - Control
 - specifies type of frame and seq.&ack. number
 - Frame Check Sequence (FCS)
 - either 16- or 32-bit CRC
- 
- Only in I- and U- frames

* Figure is courtesy of B. Forouzan



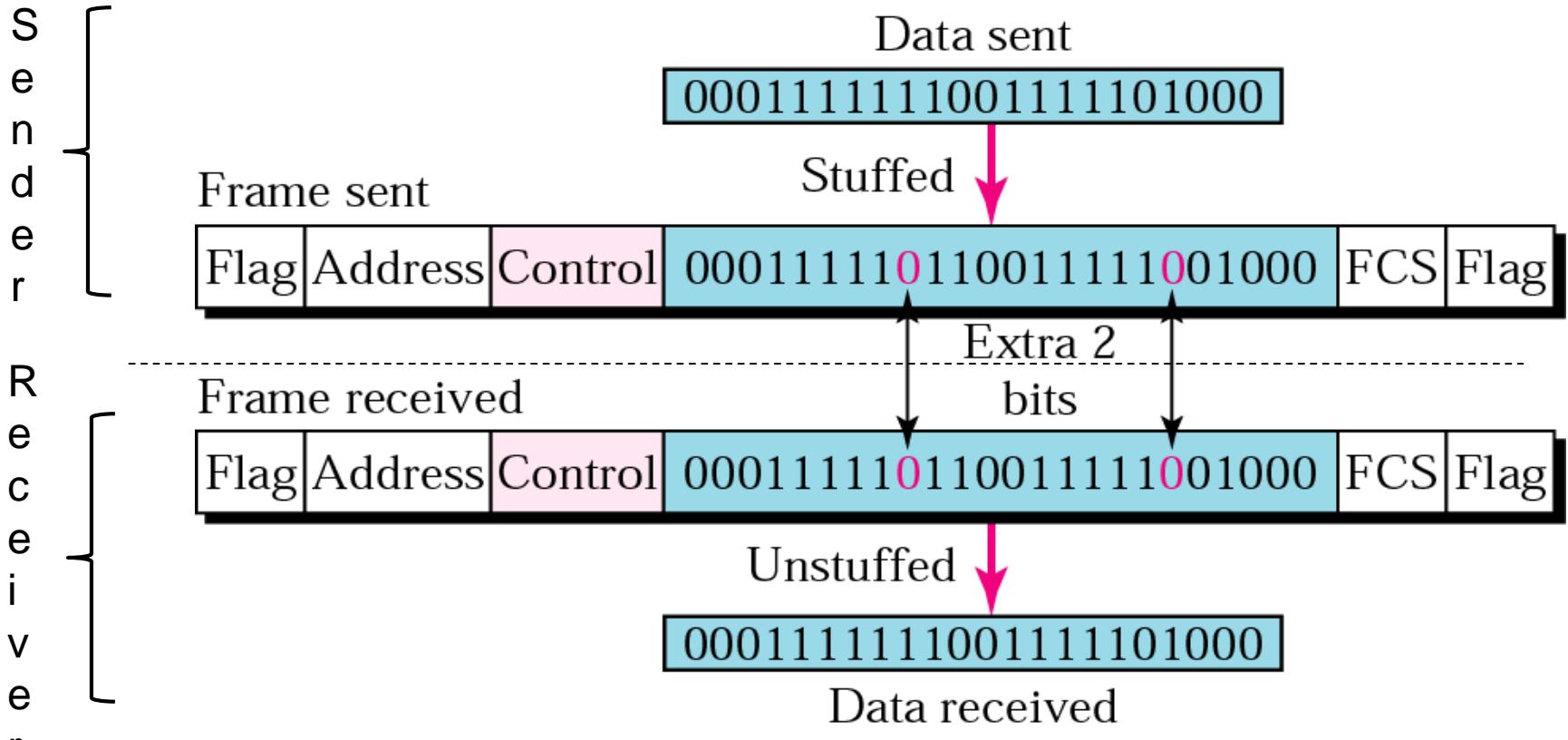
Bit-Stuffing



- Bit stuffing used to avoid confusion with data containing same combination as flag **0111110**
 - 0 inserted after every sequence of **five** 1s
 - If receiver detects five 1s
 - it checks next bit
 - If 0, it is deleted
 - If 1 and seventh bit is 0, accept as flag
 - If sixth and seventh bits 1, sender is indicating abort

* Figure is courtesy of B. Forouzan

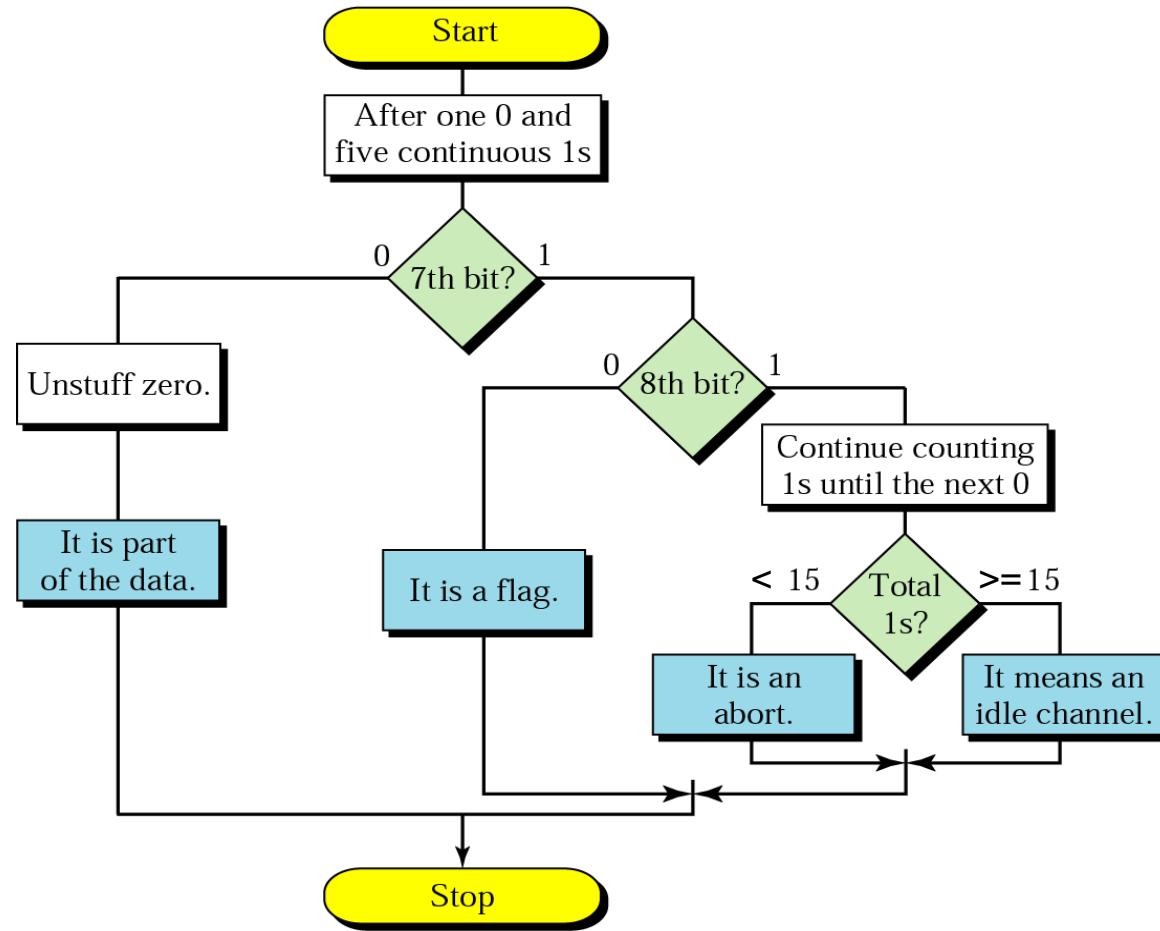
Bit Stuffing



Process of adding 0 whenever there is a flag or escape sequence in the text.

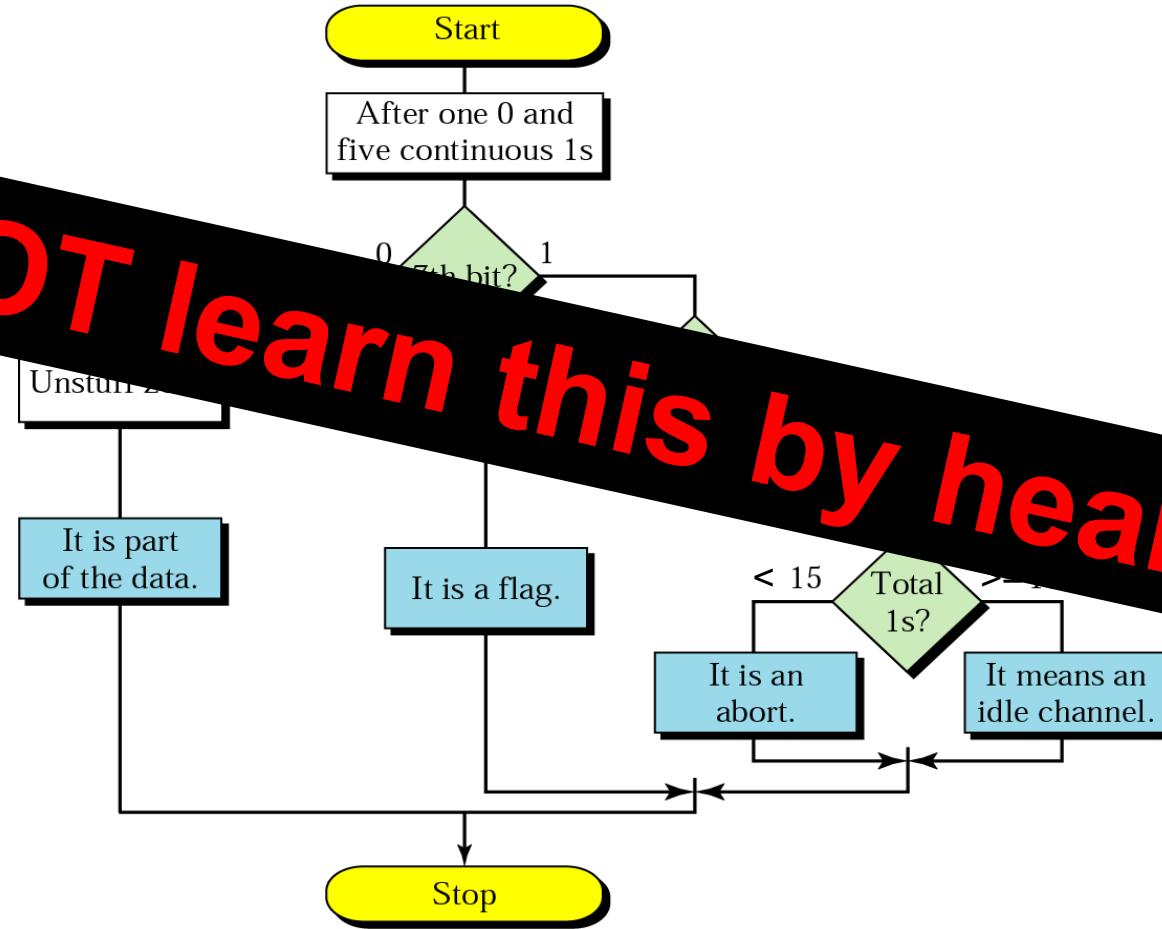
* Figure is courtesy of B. Forouzan

Bit stuffing in HDLC



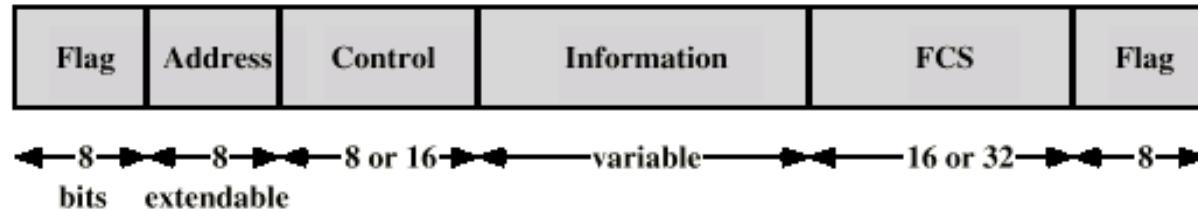
* Figure is courtesy of B. Forouzan

Bit stuffing in HDLC



* Figure is courtesy of B. Forouzan

Address Field

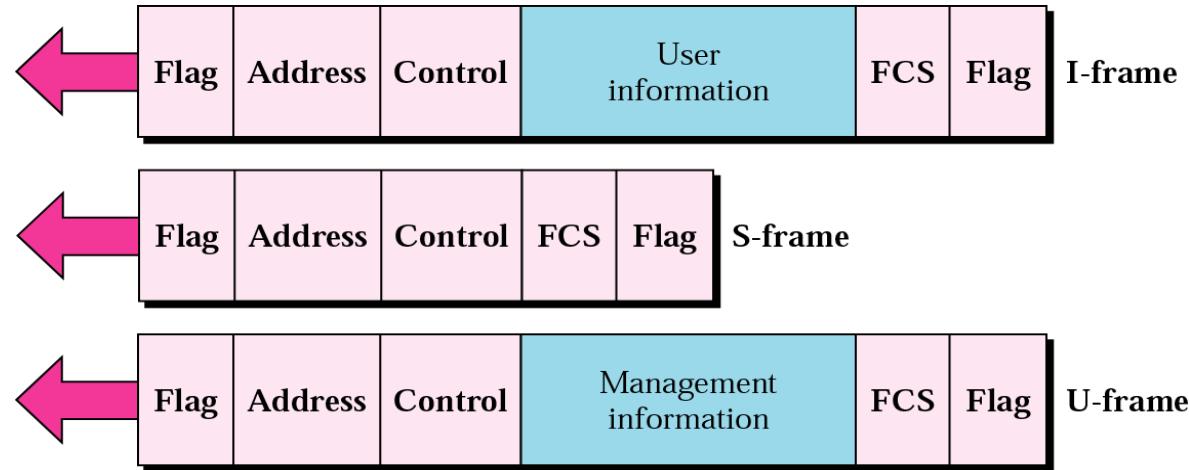


- Usually 8 bits long
- May be extended to multiples of 7 bits
 - LSB of each octet indicates that it is the last octet (1) or not (0)
- All ones (11111111) is broadcast



* Figure is courtesy of W. Stallings

HDLC Frame Types



- **I-Frame: Information Transfer Format**

- Control=

- **S-Frame: Supervisory Format**

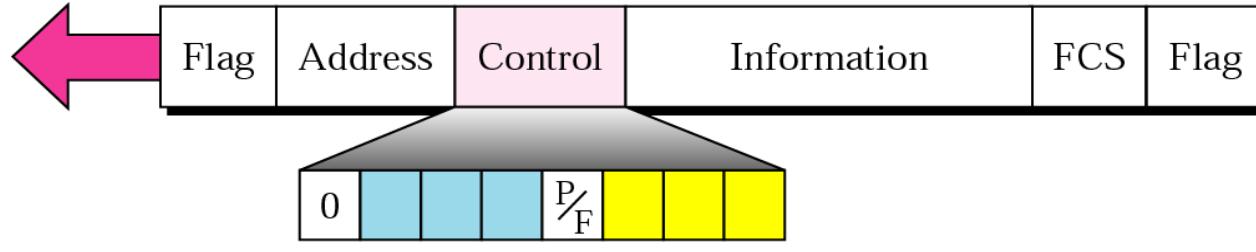
- Control=

- **U-Frame: Unnumbered Format**

- Control=

* Figure is courtesy of B. Forouzan

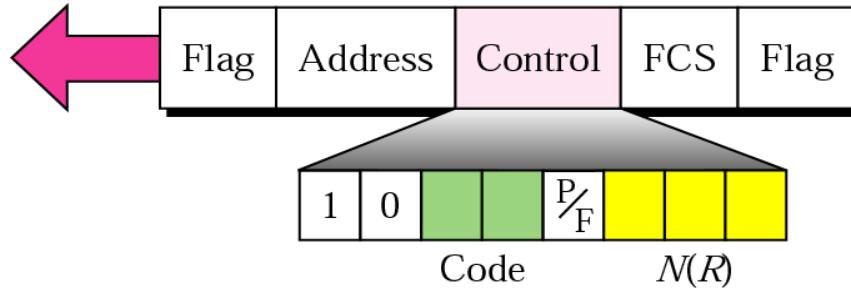
I-Frame



- $N(S)$
 - Sequence Number of Sender
- $N(R)$
 - Sequence Number of Receiver
- P/F
 - Poll/Final bit
 - Set by Primary station as request for information
 - Set by Secondary station to signal response or to signal final frame of a transmission

* Figure is courtesy of B. Forouzan

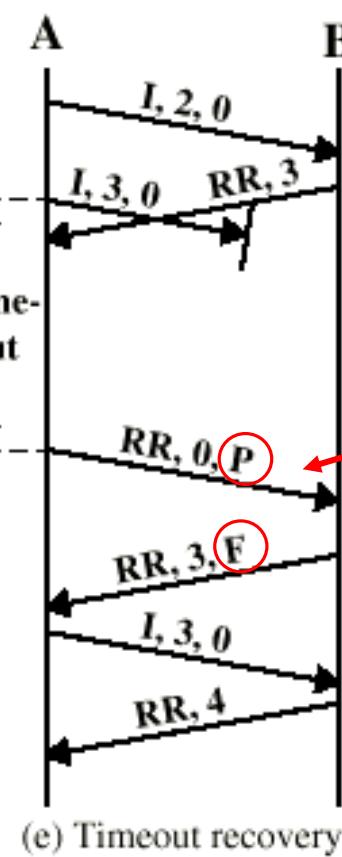
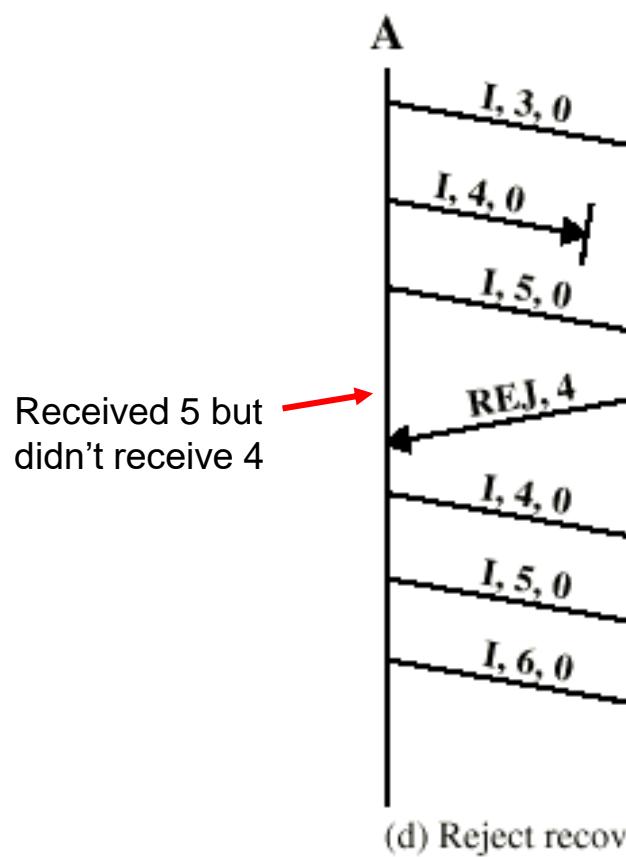
S-Frame Control Field



- Code 00 = Receive Ready (RR)
 - Acknowledge frames & waiting for more
- Code 10 = Receive Not Ready (RNR)
 - Acknowledge frames & busy right now
- Code 01 = Reject (REJ)
 - Go-Back-N NAK
- Code 11 = Selective Reject (SREJ)
 - Selective Repeat NAK

* Figure is courtesy of B. Forouzan

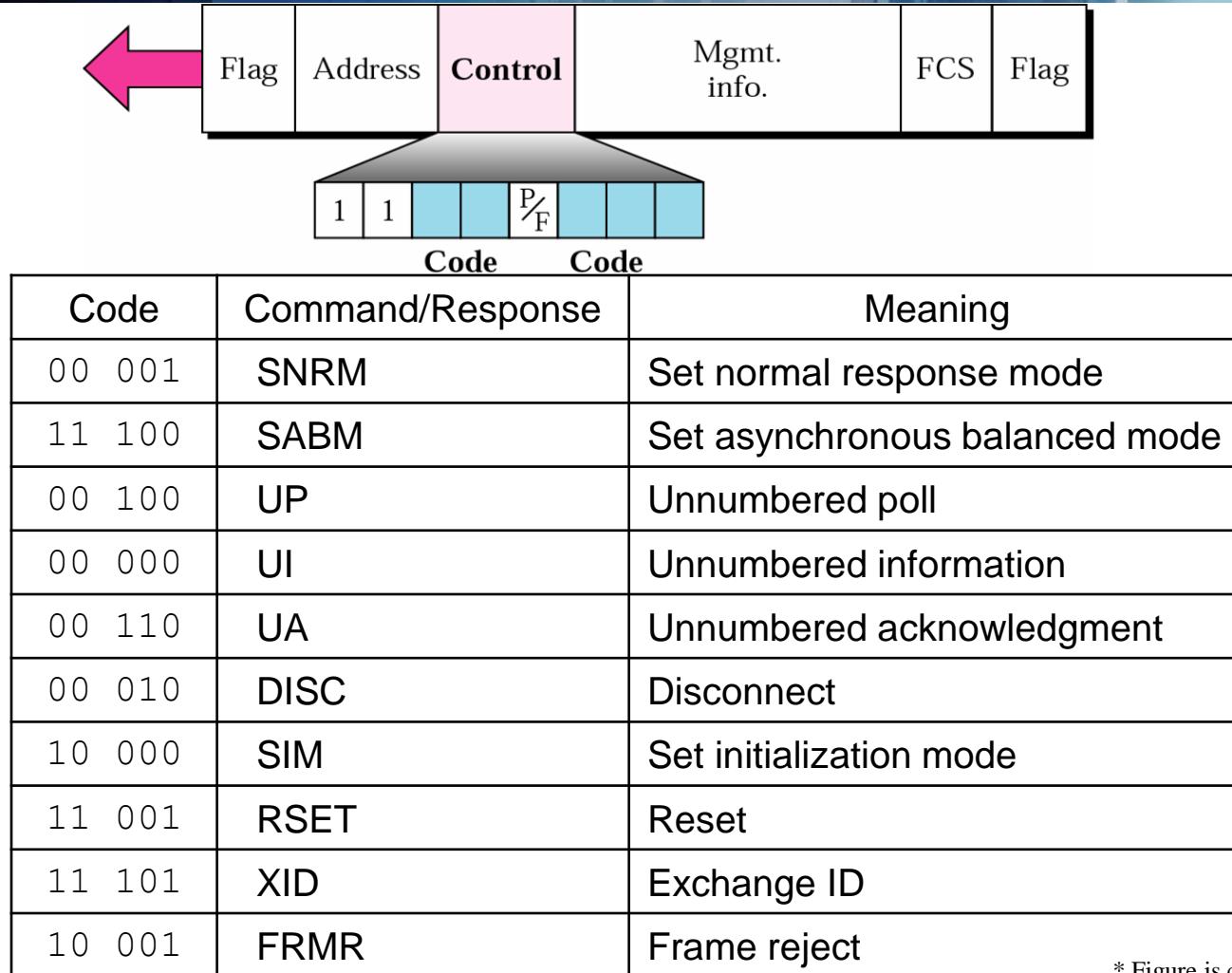
Example



* Figure is courtesy of W. Stallings



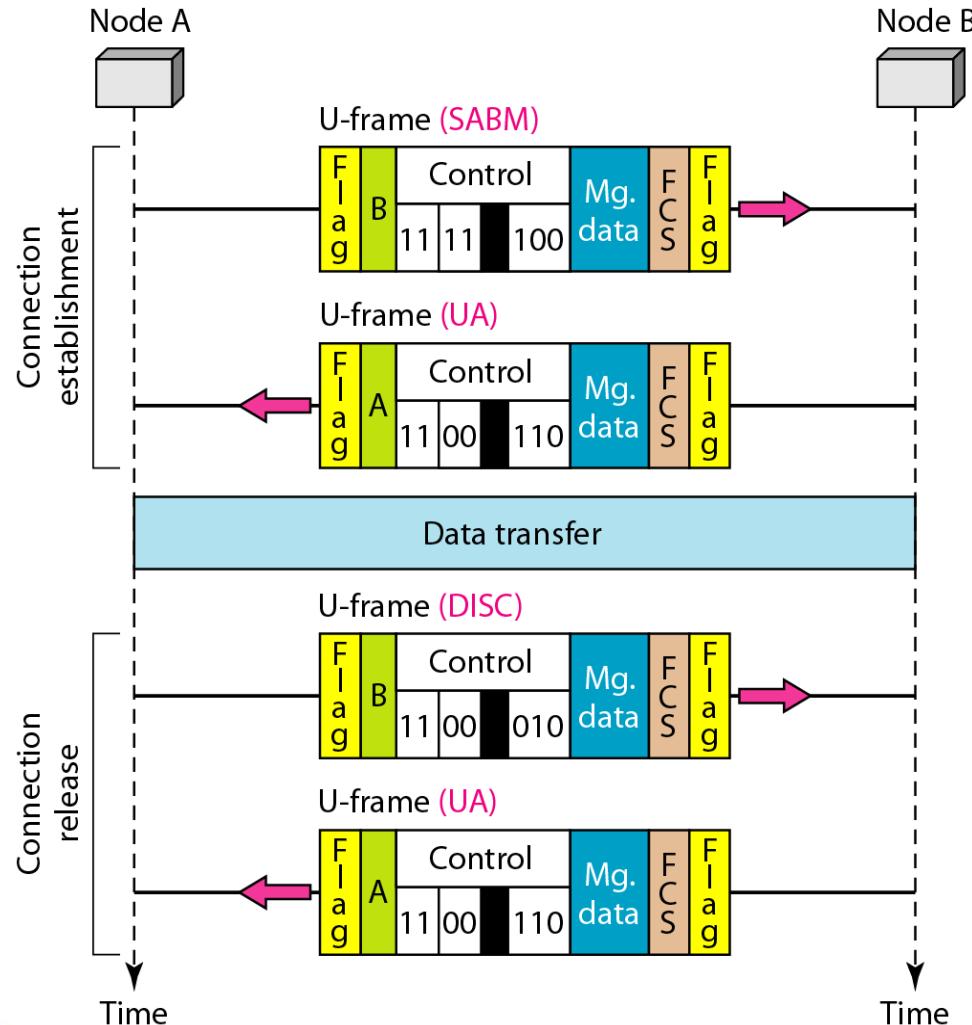
U-Frame Control Field



* Figure is courtesy of B. Forouzan



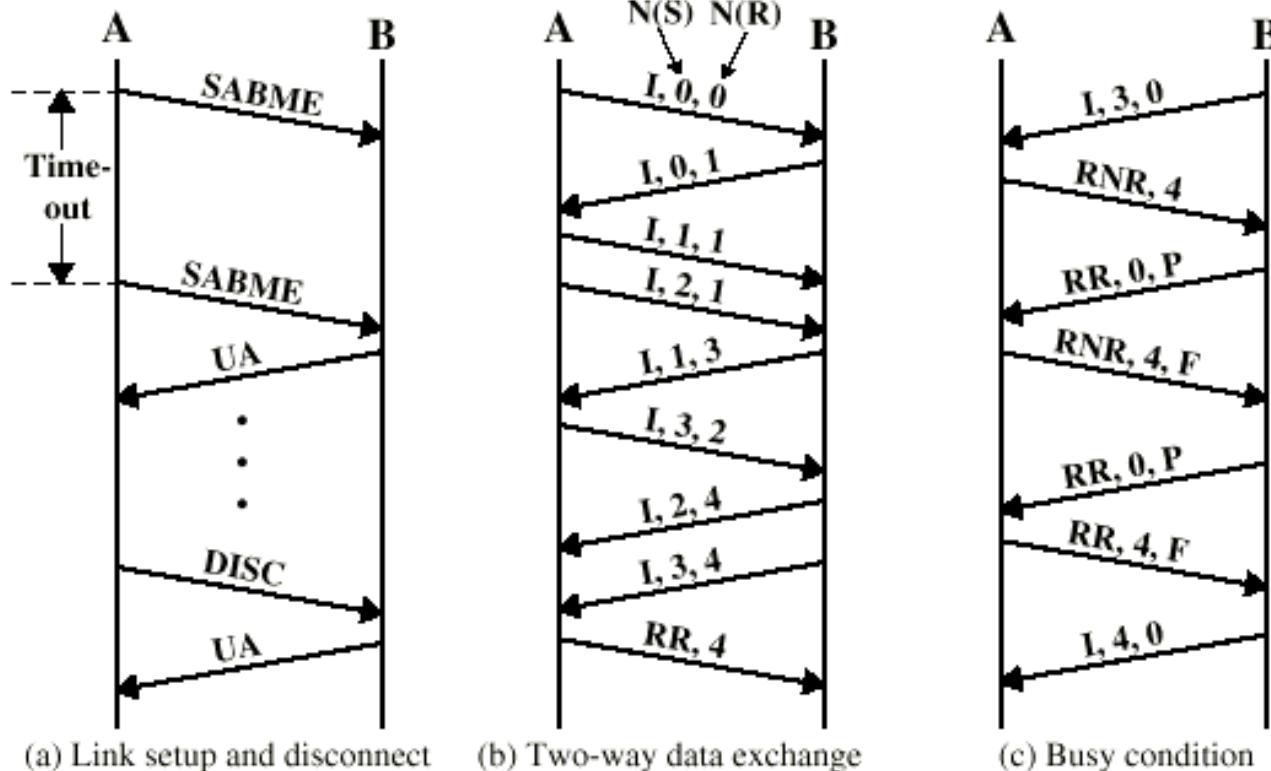
Connection & Disconnection



* Figure is courtesy of B. Forouzan



Examples of Operation

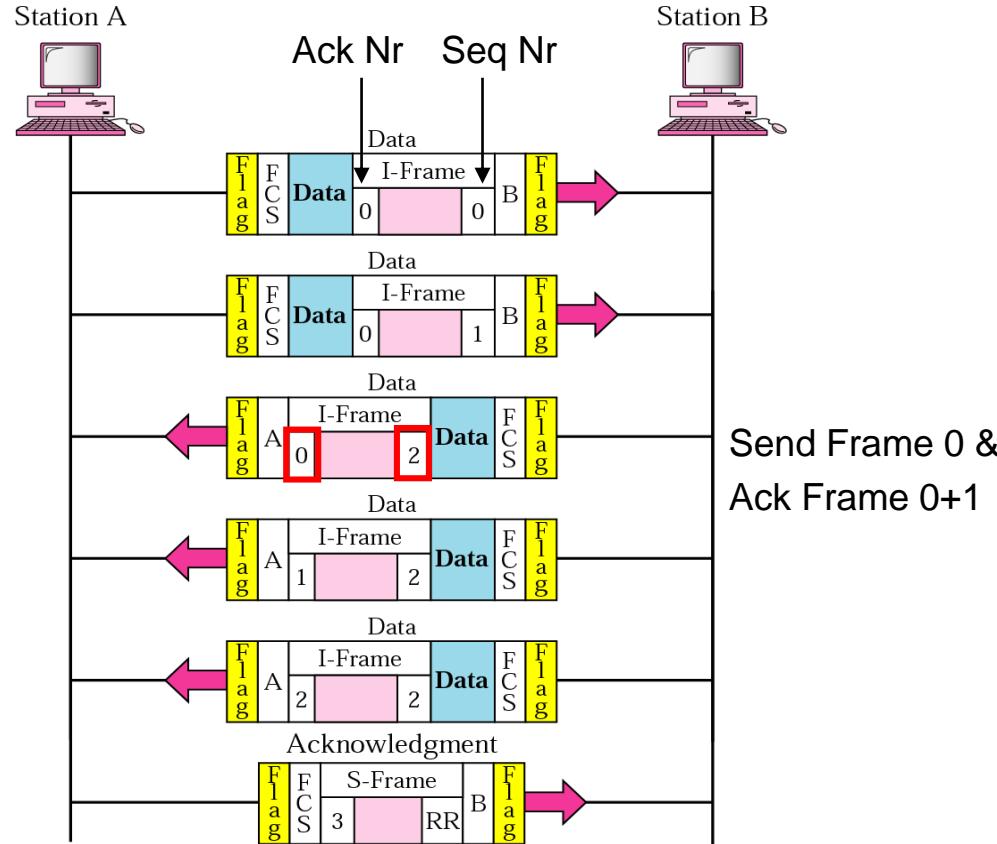


SABME	Set asynchronous balanced mode	RR	Receive Ready
I	Information	RNR	Receive Not Ready
UA	Unnumbered acknowledgment	REJ	Reject
DISC	Disconnect	SREJ	Selective Reject

* Figure is courtesy of W. Stallings

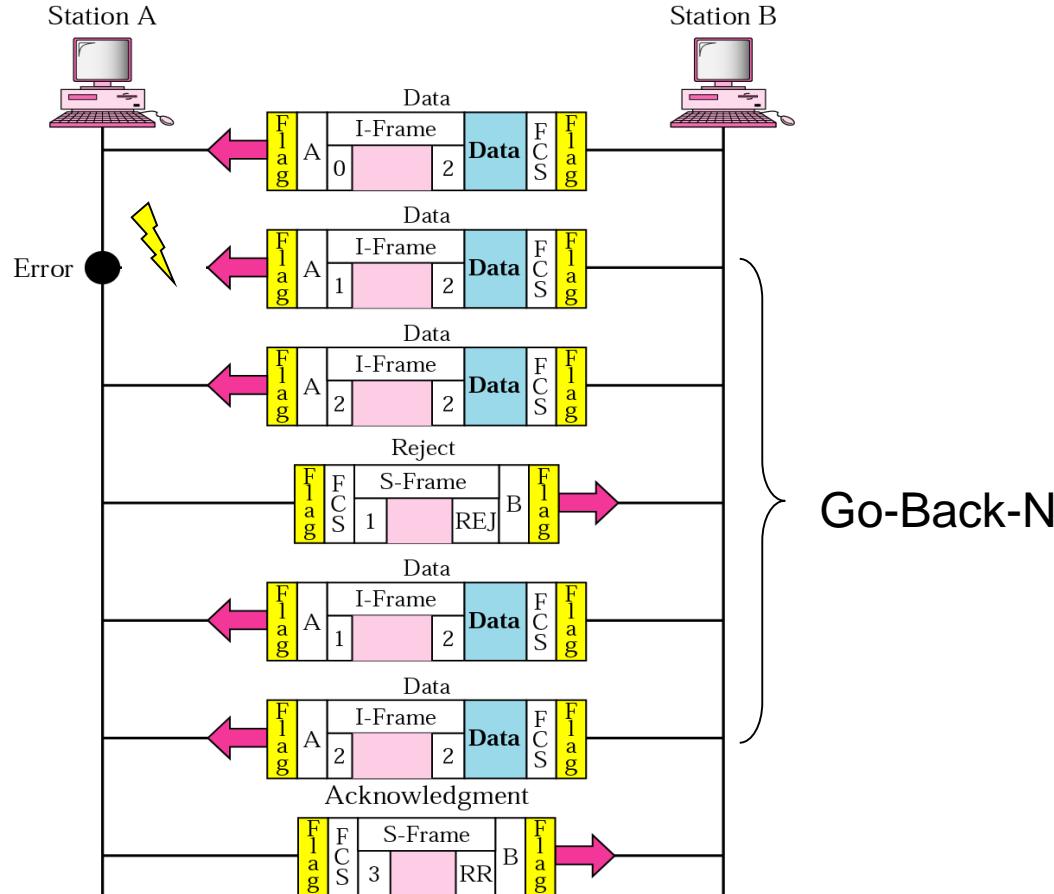


Piggybacking without Error



* Figure is courtesy of B. Forouzan

Piggybacking with Error



* Figure is courtesy of B. Forouzan

Summary: HDLC

- Three station types
 - Primary station
 - Secondary station
 - Combined station
- Operation modes
 - Normal response mode
 - Asynchronous response mode
- Three frame types
 - I-Frame: Information Transfer Format
 - S-Frame: Supervisory Format – Flow Control
 - U-Frame: Unnumbered Format – Connection setup/term./etc
- Bit-Stuffing - to avoid confusion of data and flag

HDLC – Why?

- ‘should give you a feeling for a protocol
- It includes most of the basic mechanisms
 - Framing
 - Addressing
 - Bit-stuffing
 - Flow/Error control
- Once you can run through HDLC in your head, you understand the basics of link layer protocols

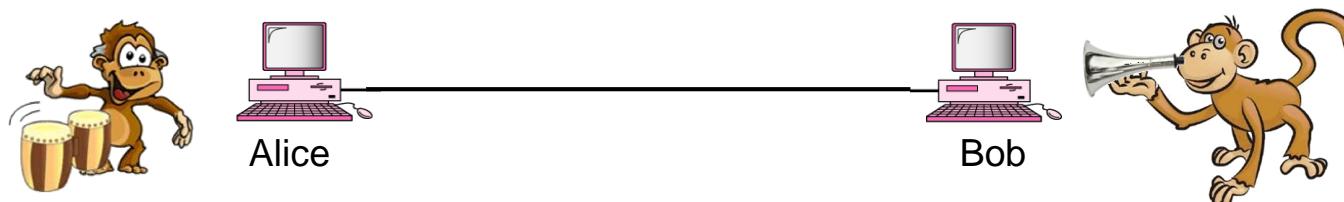
Binary Example

011111100111100011110110101011010111110

011111100111101011000110111011010111110

011111100111100011000010101111010111110

011111100111101011000110111011010111110



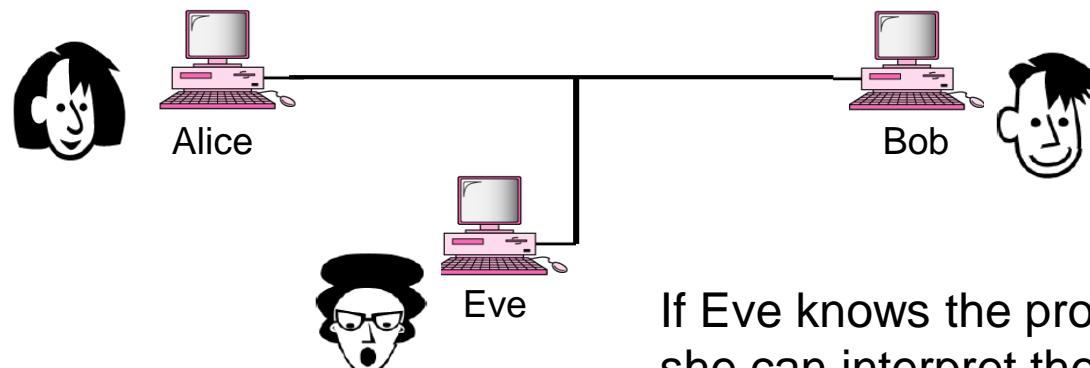
Binary Example

011111100111100011110110101011010111110

011111100111101011000110111011010111110

011111100111100011000010101111010111110

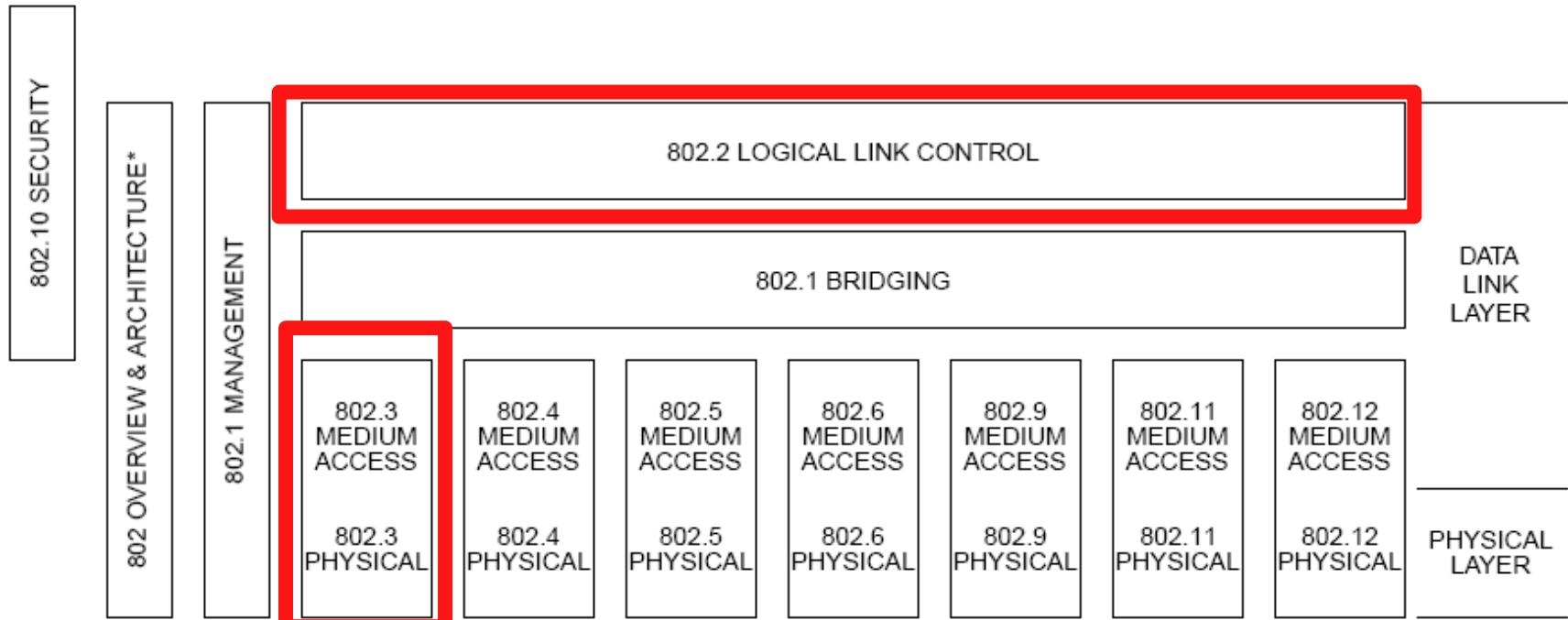
011111100111101011000110111011010111110



If Eve knows the protocol,
she can interpret the information



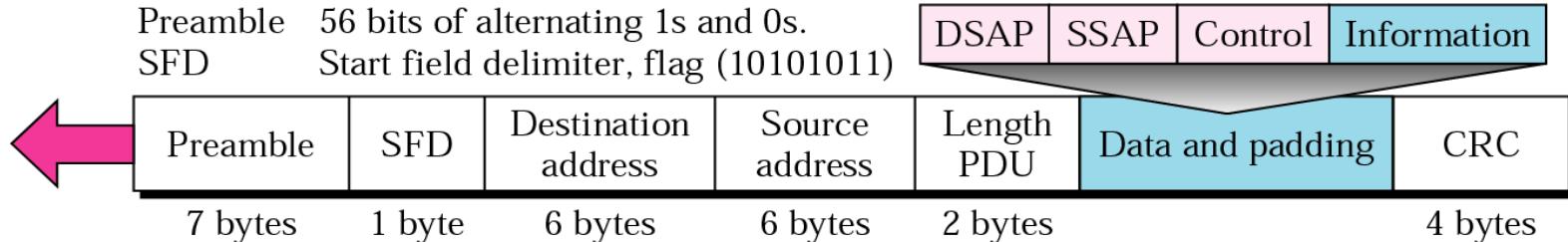
IEEE 802



- 802.3: Ethernet
- 802.11: Wifi
- 802.16: WiMAX
- 802.20: Mobile Broadband Wireless Access (MBWA)
- 802.15.1: Bluetooth
- 802.15.4: ZigBee

* Figure is courtesy of ANSI/IEEE Std 802.11

802.3 MAC Frame



- 64-bit frame preamble (10101010) used to synchronize reception
 - 7 bit preamble (10101010) + 1 start flag (10101011)
- Maximum frame length: 1518 bytes
 - ⇒ max 1500 bytes payload
- Minimum frame length: 64 bytes
 - ⇒ min 46 bytes payload

* Figure is courtesy of B. Forouzan

802.2 LLC Control Fields

	LLC PDU control field bits									
	1	2	3	4	5	6	7	8	9	10–16
Information transfer command/response (I-format PDUs)	0	N(S)						P/F	N(R)	
Supervisory commands/responses (S-format PDUs)	1	0	S	S	X	X	X	X	P/F	N(R)
Unnumbered commands/responses (U-format PDUs)	1	1	M	M	P/F	M	M	M		

N(S) = sender send sequence number (Bit 2=lower-order-bit)
N(R) = sender receive sequence number (Bit 10=lower-order-bit)
S = supervisory function bit
M = modifier function bit
X = reserved and set to zero
P/F = poll bit—command LLC PDUs
 final bit—response LLC PDUs
 (1=poll/initial)

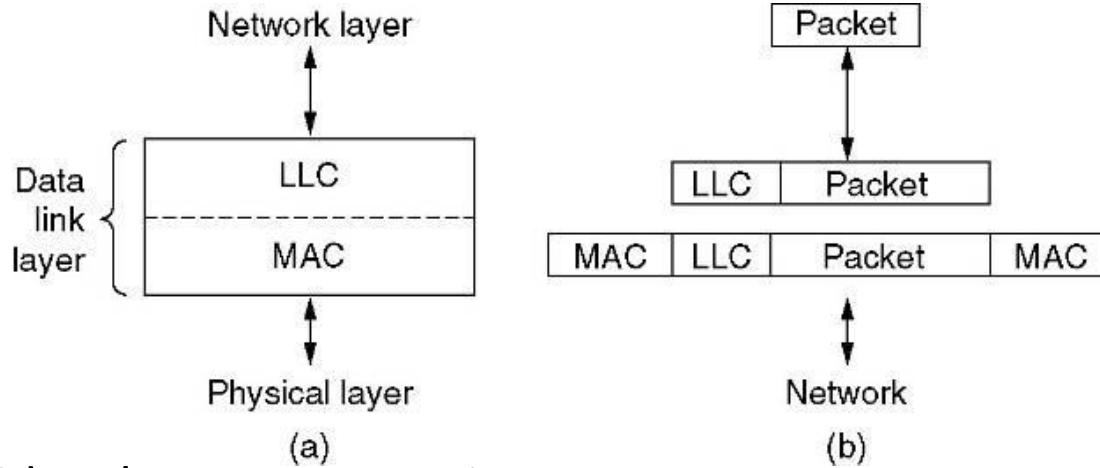
$m = 7$
 $2^m = 128$
 max w-size= 64 frames

Figure 9—LLC PDU control field formats

ANSI/IEEE Std 802.2, 1998 Edition



IEEE 802.2: Logical Link Control (LLC)

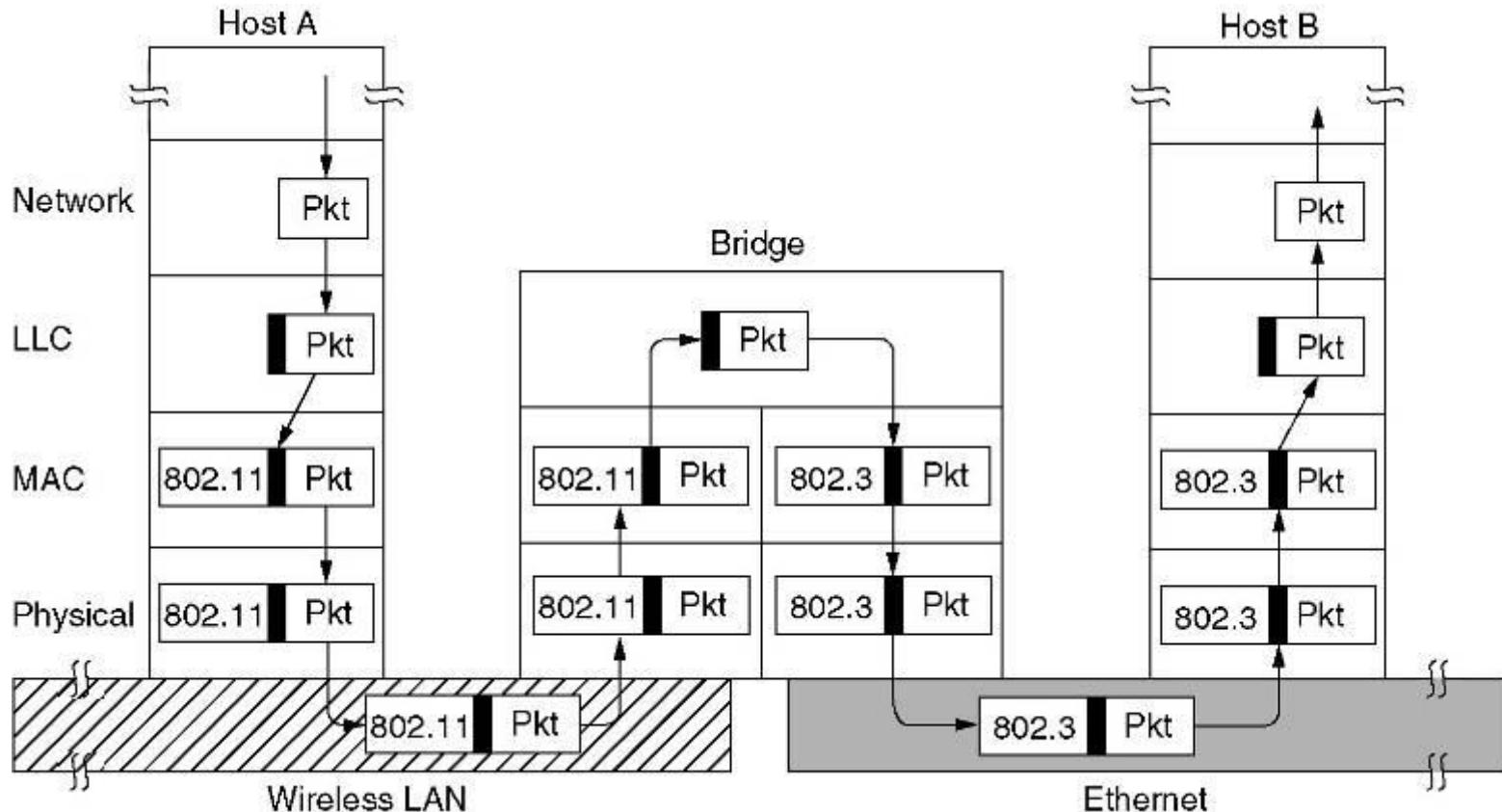


- LLC provides three HDLC services:
 - 1. Unacknowledged connectionless service, recall HDLC has unnumbered frames;
 - 2. Reliable connection-oriented service in the form of HDLC ABM mode;
 - 3. Acknowledged connectionless service, need to add two unnumbered frames to HDLC frame set.
- LLC can provide reliable packet transfer service

* Figure is courtesy of A. Tanenbaum

Bridges from 802.x to 802.y

Operation of a LAN bridge from 802.11 to 802.3.

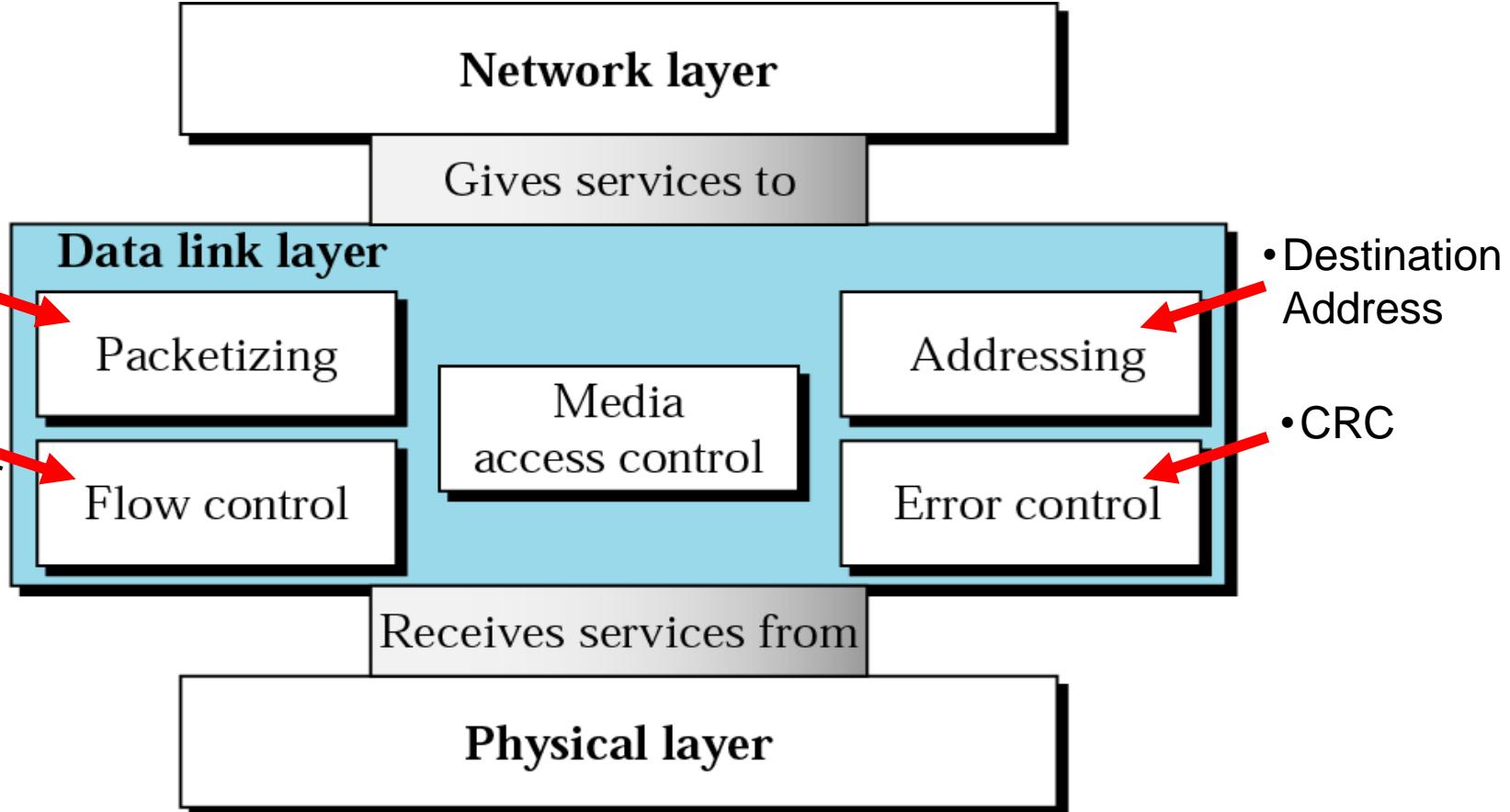


* Figure is courtesy of A. Tanenbaum



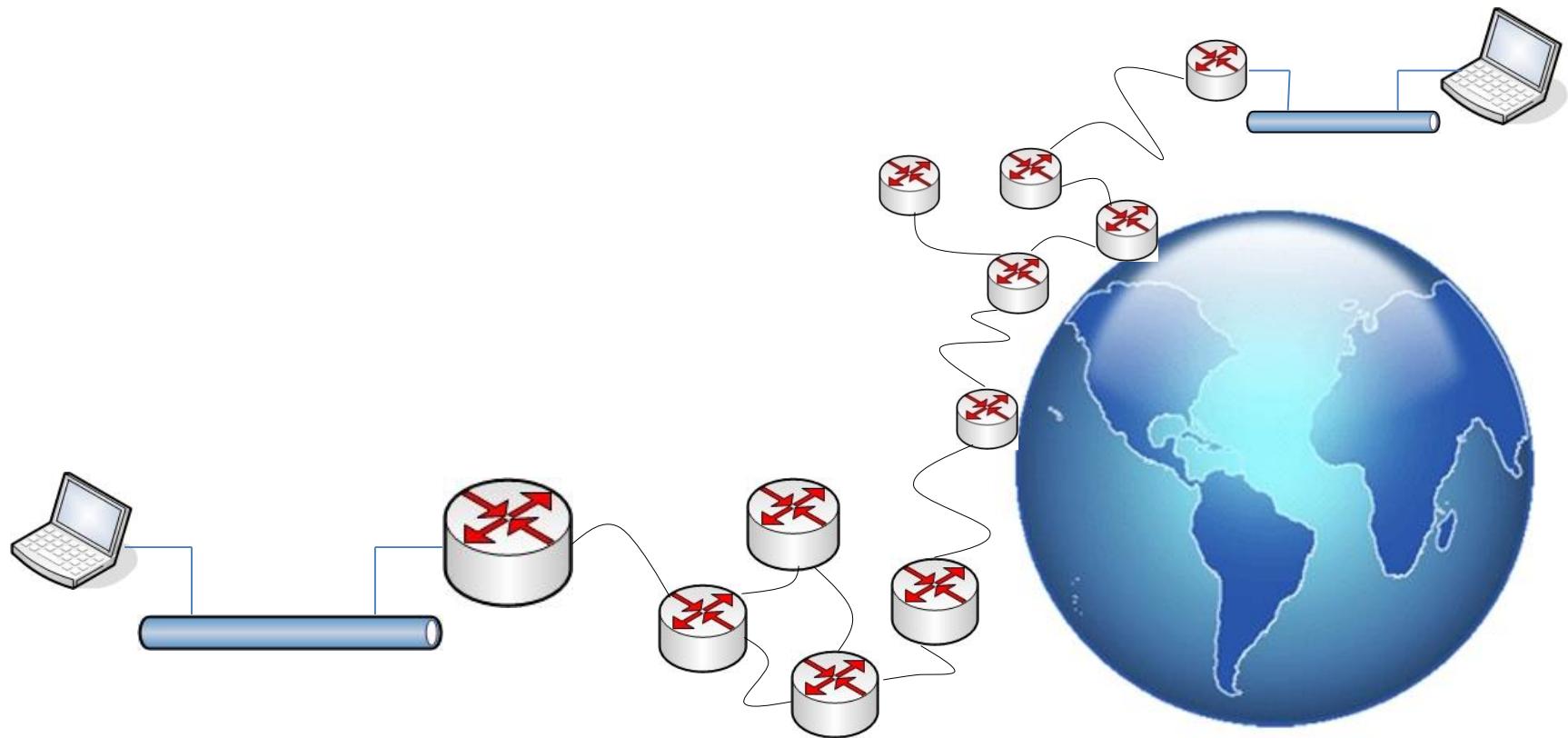
Link Layer

- Flag Bytes
- Frame Format
- Options for Stop&Wait, Selective Repeat, etc



* Figure is courtesy of B. Forouzan

Every Connection involves the Link Layer





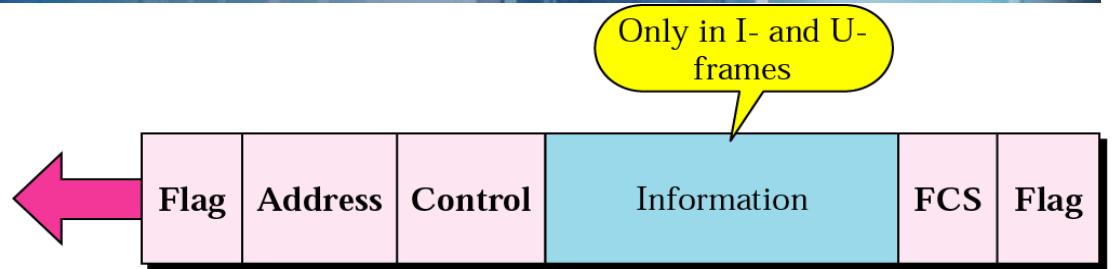
Review: HDLC

- Three Frame Types
 - I-Frame: Information Transfer Format
 - S-Frame: Supervisory Format
 - U-Frame: Unnumbered Format
- Implements Flow Control & Error Control mechanisms
 - Stop-And-Wait
 - Go-Back-N
 - Selective Repeat
- Bit-Stuffing - to avoid confusion of data and flag



HDLC Frame

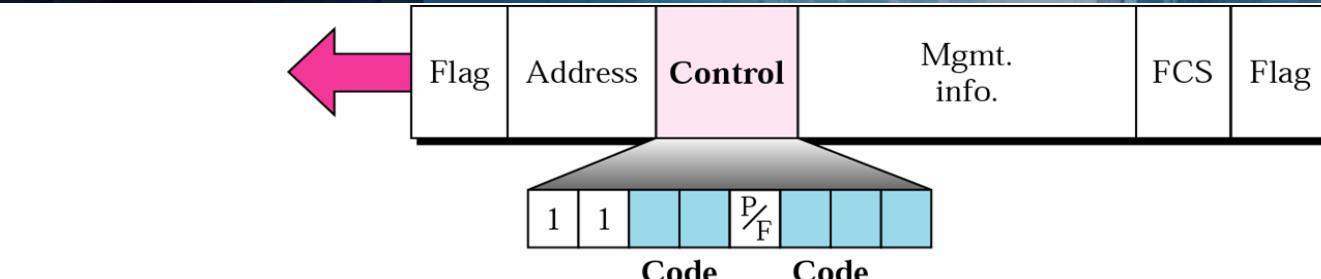
- Flag= 01111110
 - Specifies beginning and end of frame



- Address
 - Specifies secondary station as either sender or receiver
- Control
 - Specifies type of frame and seq.&ack. number
- Frame Check Sequence (FCS)
 - Either 16- or 32-bit CRC

* Figure is courtesy of B. Forouzan

U-Frame Control Field

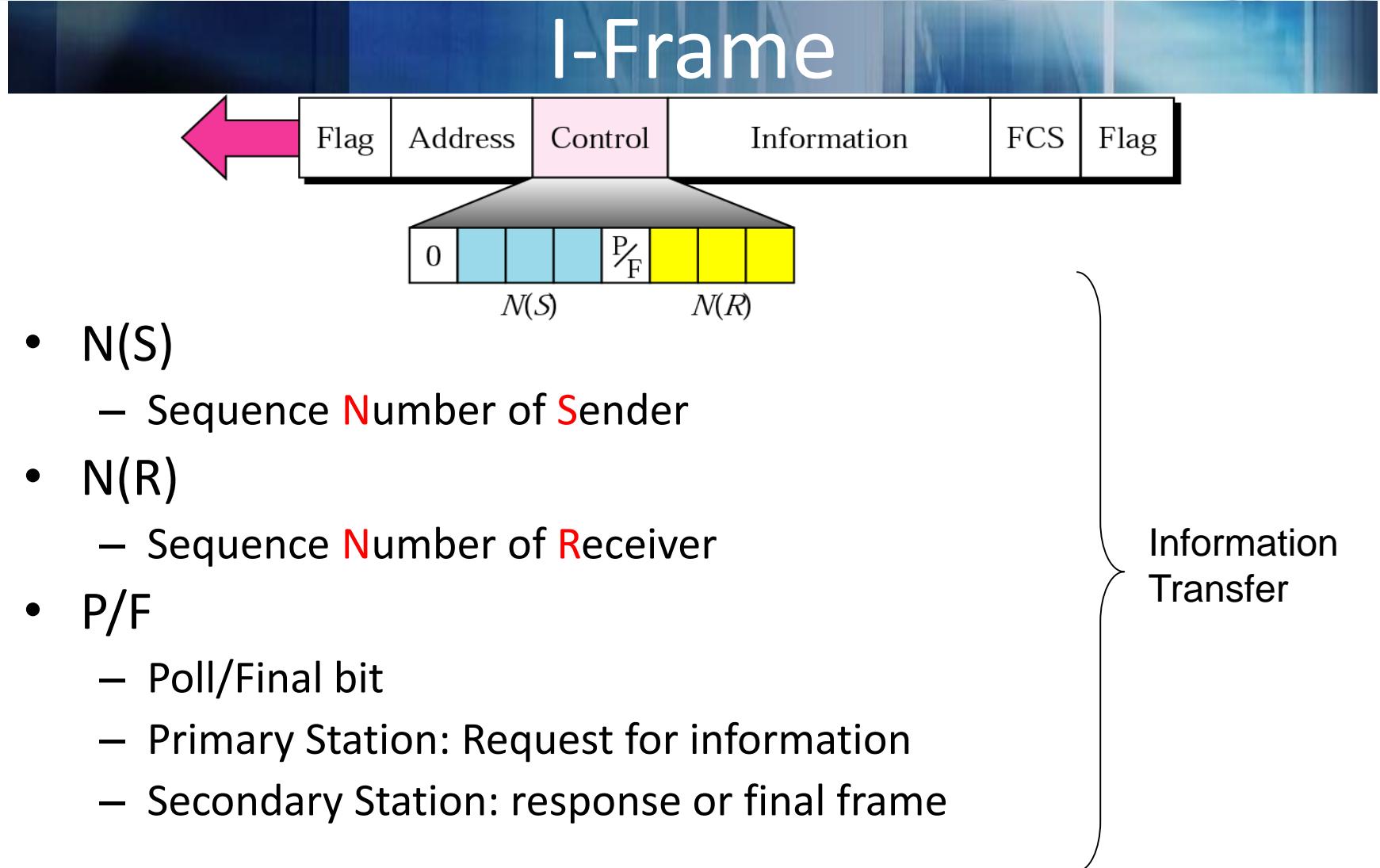


Code	Command/Response	Meaning
00 001	SNRM	Set normal response mode
11 100	SABM	Set asynchronous balanced mode
00 100	UP	Unnumbered poll
00 000	UI	Unnumbered information
00 110	UA	Unnumbered acknowledgment
00 010	DISC	Disconnect
10 000	SIM	Set initialization mode
11 001	RSET	Reset
11 101	XID	Exchange ID
10 001	FRMR	Frame reject

Managing
Connection

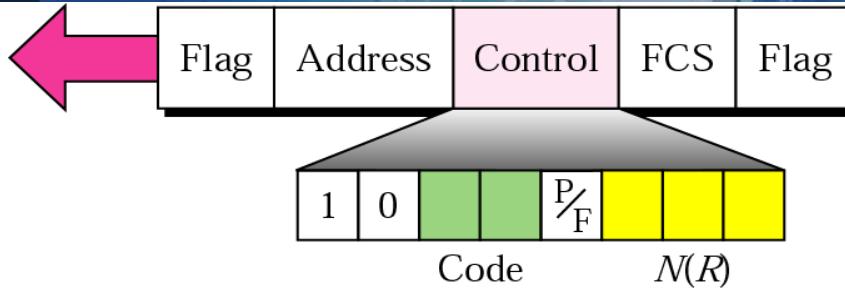
* Figure is courtesy of B. Forouzan





* Figure is courtesy of B. Forouzan

S-Frame Control Field

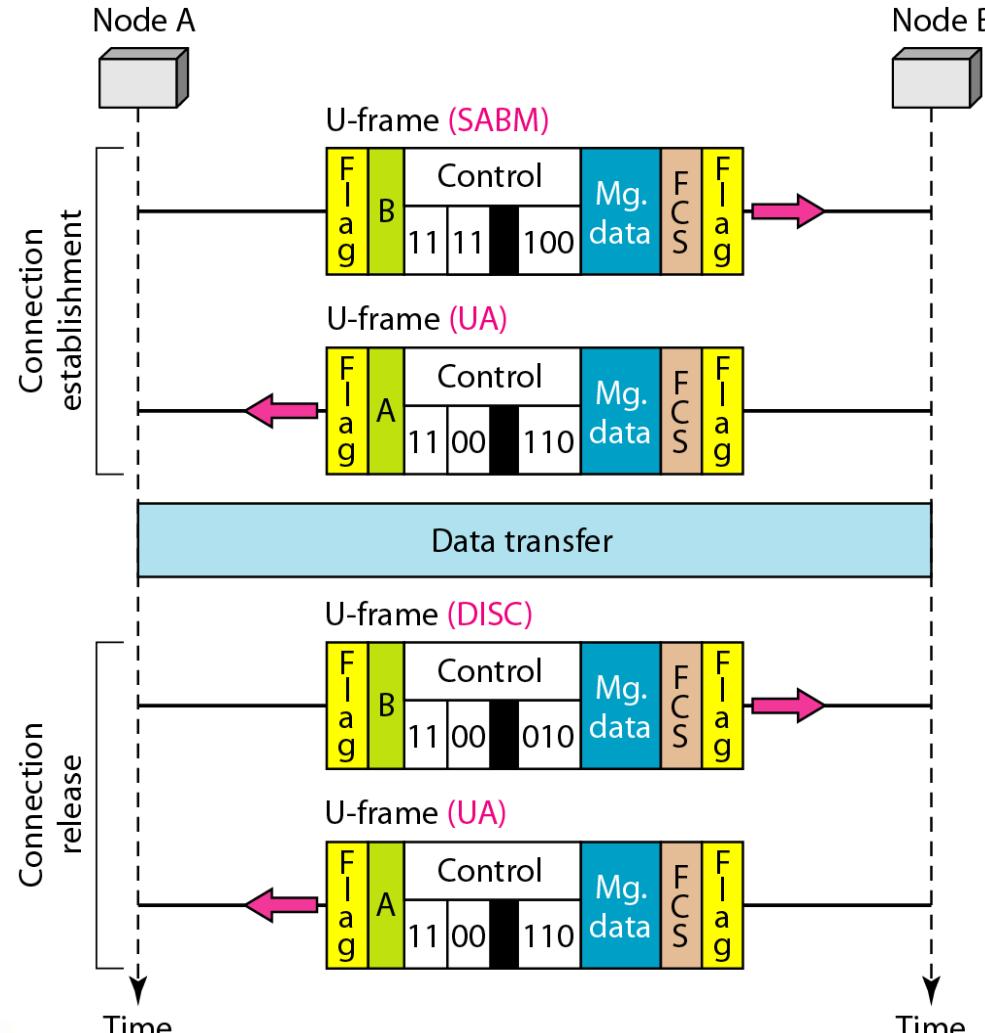


- Code 00 = Receive Ready (RR)
 - Acknowledge frames & waiting for more
- Code 10 = Receive Not Ready (RNR)
 - Acknowledge frames & busy right now
- Code 01 = Reject (REJ)
 - Go-Back-N NAK
- Code 11 = Selective Reject (SREJ)
 - Selective Repeat NAK

Flow&Error
Control

* Figure is courtesy of B. Forouzan

Connection & Disconnection

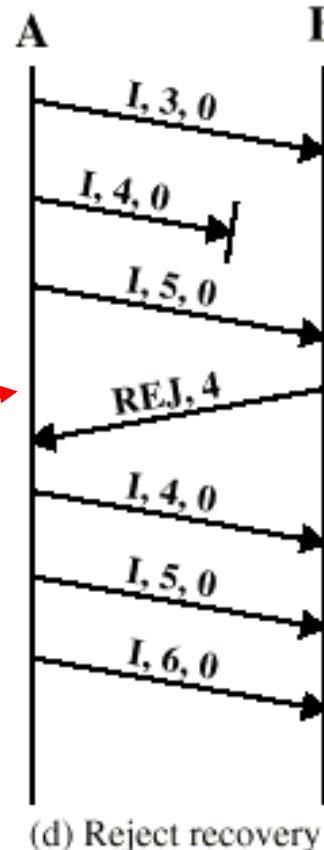


* Figure is courtesy of B. Forouzan

Examples of Operation

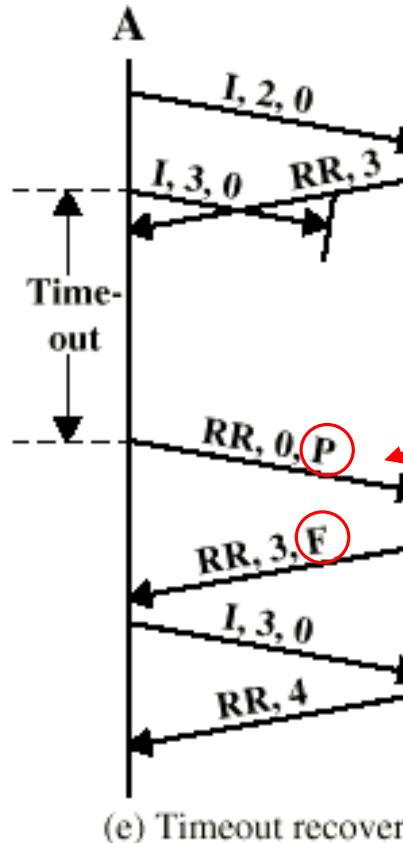
Unexpected Seq#

Received 5 but
didn't receive 4



(d) Reject recovery

Timeout of ACK4

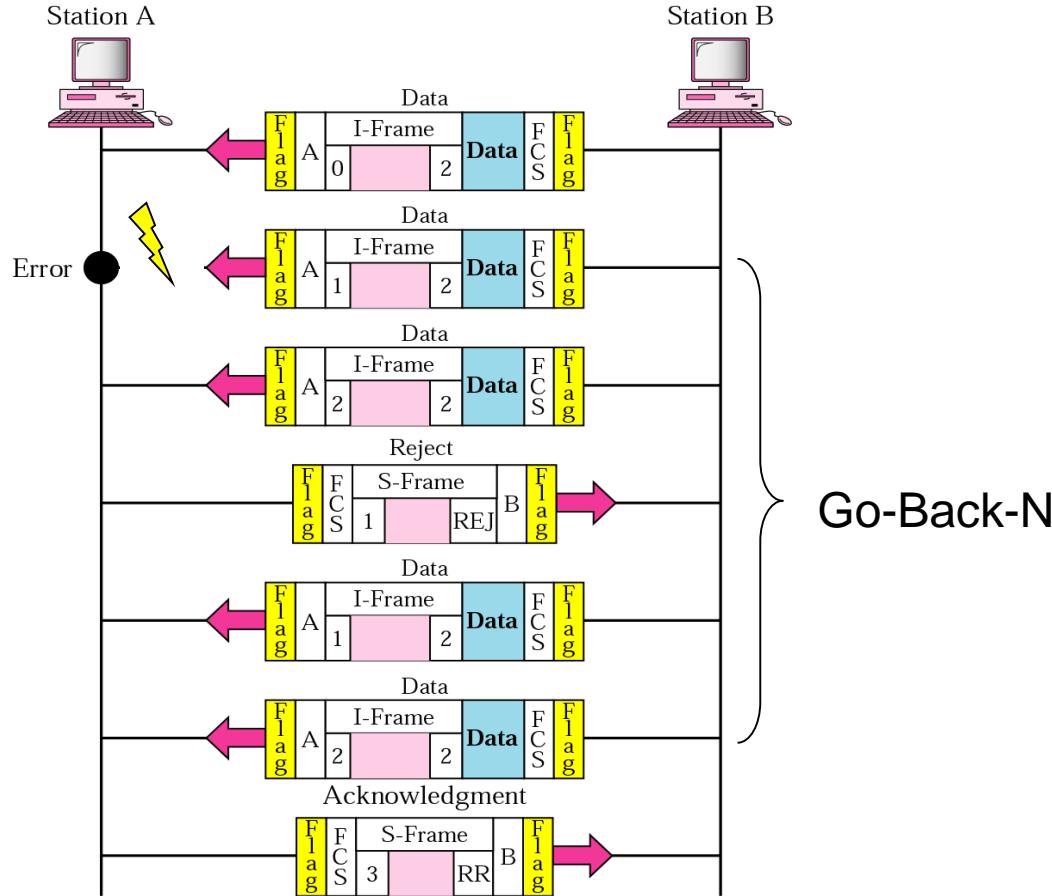


Hello?? Anybody
home??

* Figure is courtesy of W. Stallings



Piggybacking with Error



* Figure is courtesy of B. Forouzan

802.2 LLC Control Fields

	LLC PDU control field bits									
	1	2	3	4	5	6	7	8	9	10–16
Information transfer command/response (I-format PDUs)	0	N(S)						P/F	N(R)	
Supervisory commands/responses (S-format PDUs)	1	0	S	S	X	X	X	X	P/F	N(R)
Unnumbered commands/responses (U-format PDUs)	1	1	M	M	P/F	M	M	M		

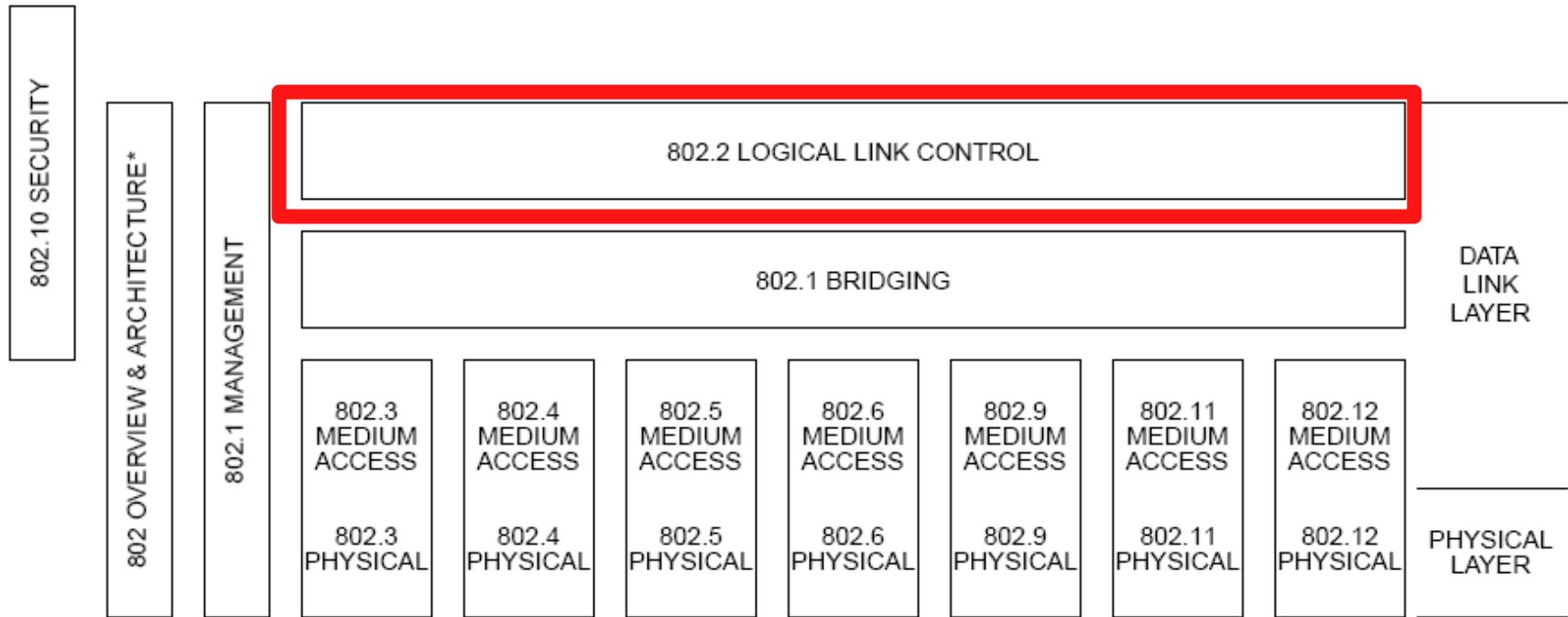
N(S) = sender send sequence number (Bit 2=lower-order-bit)
N(R) = sender receive sequence number (Bit 10=lower-order-bit)
S = supervisory function bit
M = modifier function bit
X = reserved and set to zero
P/F = poll bit—command LLC PDUs
 final bit—response LLC PDUs
 (1=poll/initial)

$m = 7$
 $2^m = 128$
 max w-size= 64 frames

Figure 9—LLC PDU control field formats

ANSI/IEEE Std 802.2, 1998 Edition

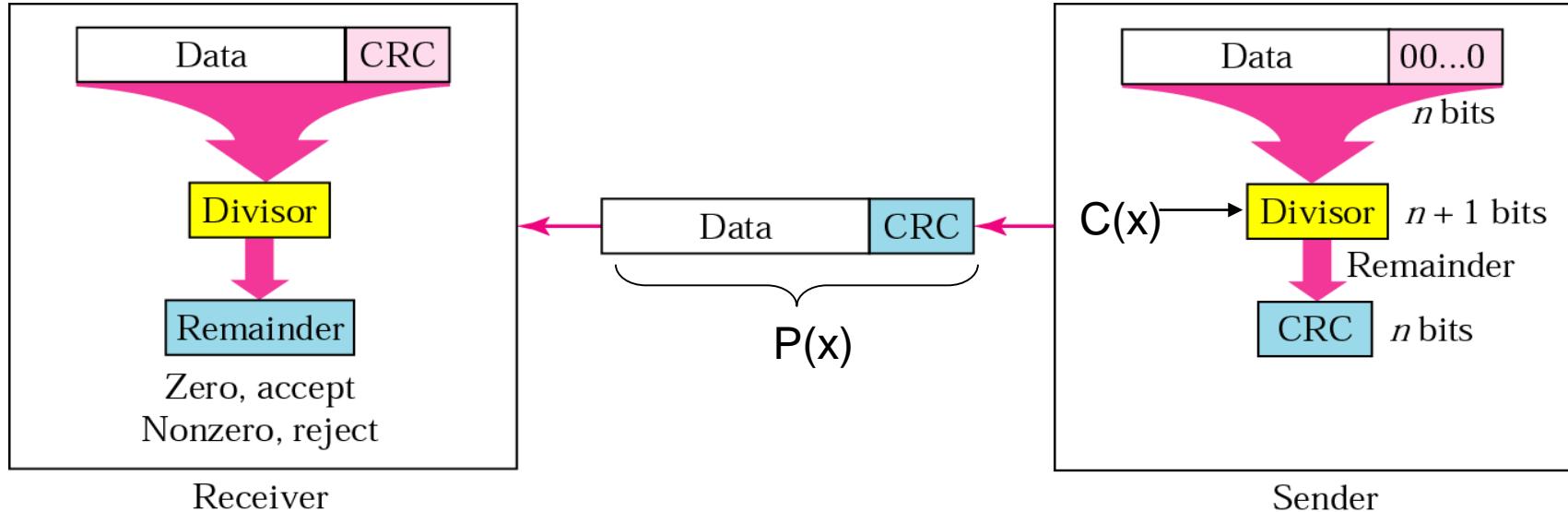
IEEE 802



- 802.3: Ethernet
- 802.11: Wifi
- 802.16: WiMAX
- 802.20: Mobile Broadband Wireless Access (MBWA)
- 802.15.1: Bluetooth
- 802.15.4: ZigBee

* Figure is courtesy of ANSI/IEEE Std 802.11

Cyclic Redundancy Check (CRC)

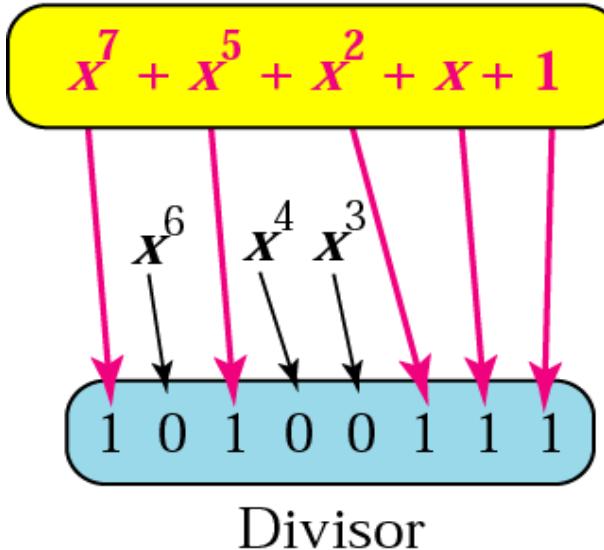


- $P(x)$ divided by $C(x) = 0$
- $(P(x)+\text{remainder})$ divided by $C(x)$ should be $\neq 0$

* Figure is courtesy of B. Forouzan

Polynomial Notation

Polynomial



- Rules for selecting divisor:
 - It should not be divisible by x
 - It should be divisible by $x+1$

* Figure is courtesy of B. Forouzan

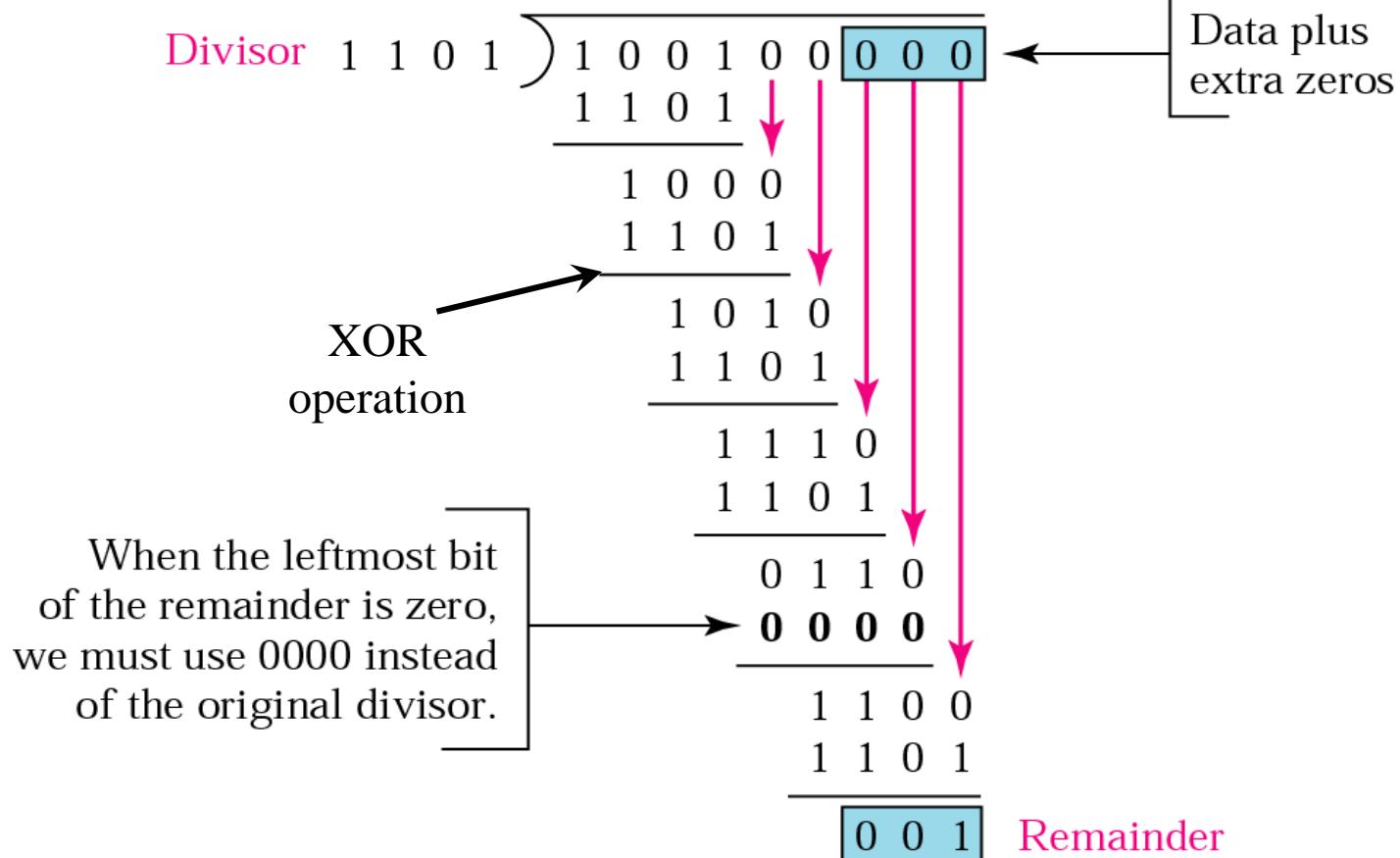
Standard Polynomials

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

* Figure is courtesy of B. Forouzan



CRC: Sender



Data transmitted to receiver: $\underbrace{1 \ 0 \ 0 \ 1 \ 0 \ 0}_{\text{Data}} \underbrace{0 \ 0 \ 1}_{\text{CRC}}$

* Figure is courtesy of B. Forouzan

Division – Decimal&Binary

$$39 / 20 = 1 + 19$$

$$\begin{array}{r} 100111 \quad / \quad 10100 \\ 32 \quad 4 \ 2 \ 1 \qquad 16 \quad 4 \\ \end{array} = 1 + \begin{array}{r} 10011 \\ 16 \quad 2 \ 1 \end{array}$$



CRC Calculation

- CRC Calculation → Polynomial Division
not Binary Division!!!

$$\begin{array}{r} x^3 + 4x^2 + 3x + 12 \quad / \quad x^2 + 3 \quad = \quad x + 4 \\ x^3 \qquad \qquad + 3x \\ \hline 4x^2 \qquad \qquad + 12 \\ \hline 0 \end{array}$$



CRC Calculation

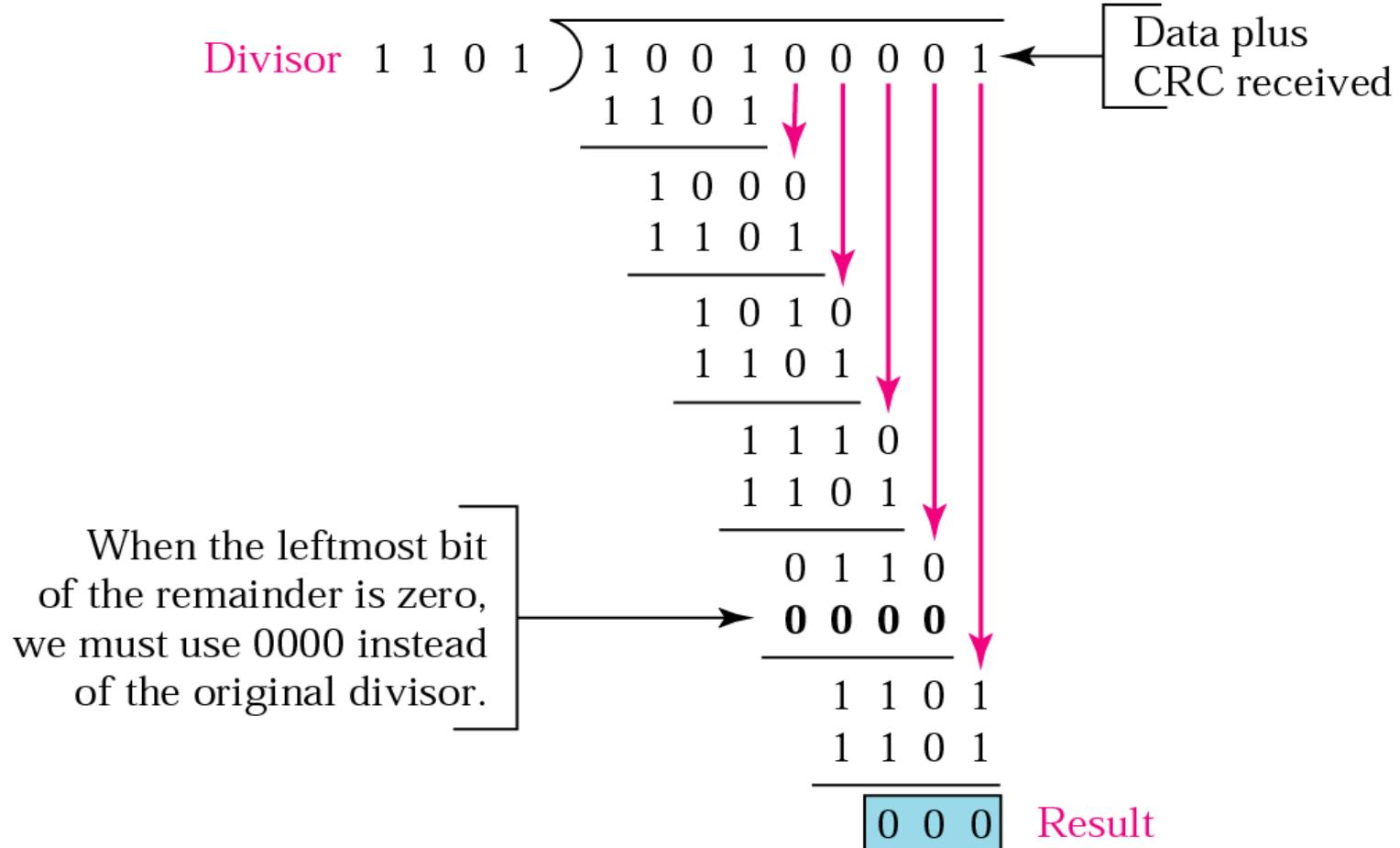
- CRC Calculation → Polynomial Division
not Binary Division!!!
- CRC: Coefficient $r=\{0,1\}$

10001000000000001011

$$x^{20} + x^{15} + x^4 + x + 1 \quad / \quad x^{16} + x^{12} + x^5 + 1$$



CRC: Receiver



* Figure is courtesy of B. Forouzan





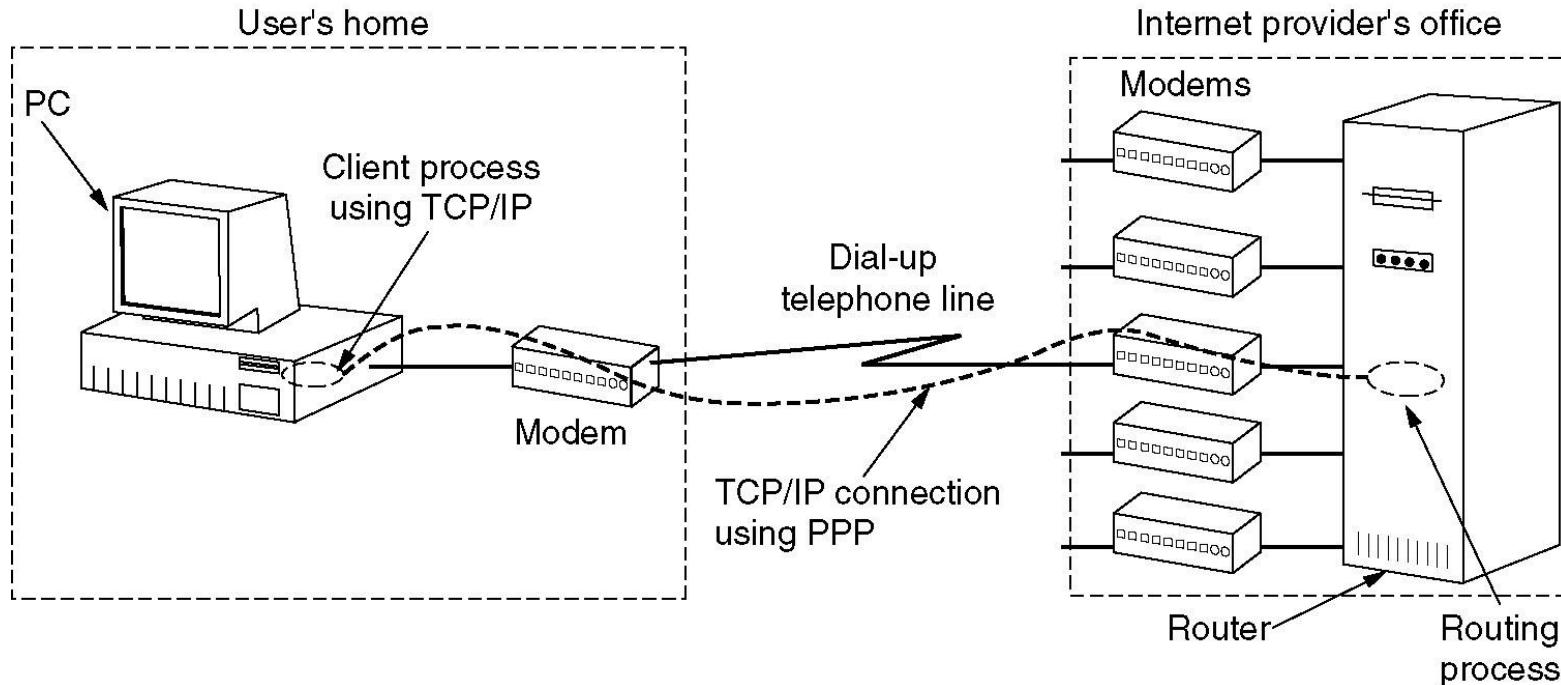
Point-to-Point Protocol (PPP)



Point-to-Point Protocol (PPP)

- Used for any kind of serial point to point connection e.g. dial-up, serial x-wire
- Based on HDLC
- Provides
 - Format negotiation
 - Authentication
 - Connection establishment/termination

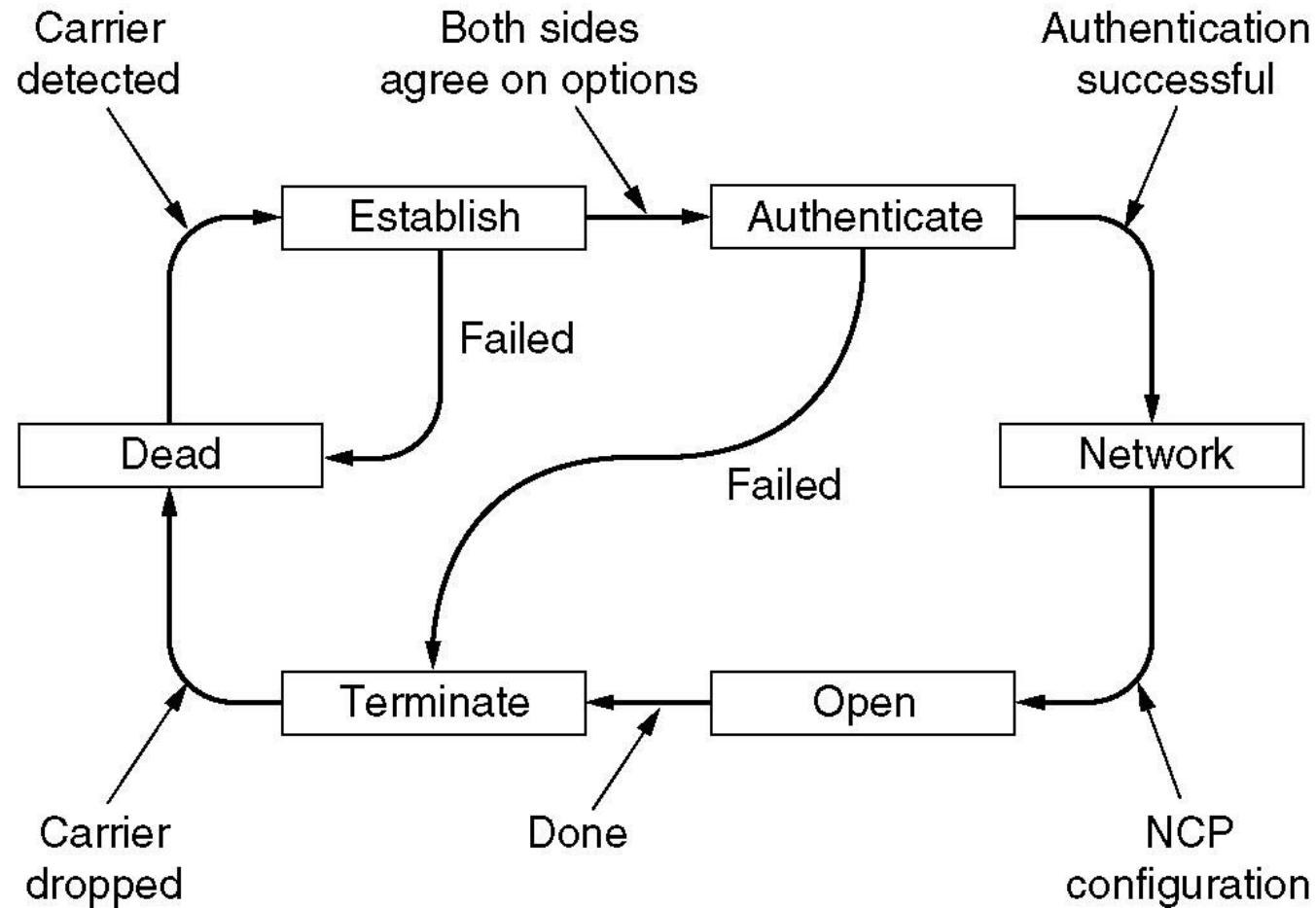
PPP between Home & ISP



* Figure is courtesy of A. Tanenbaum



PPP – Life Cycle



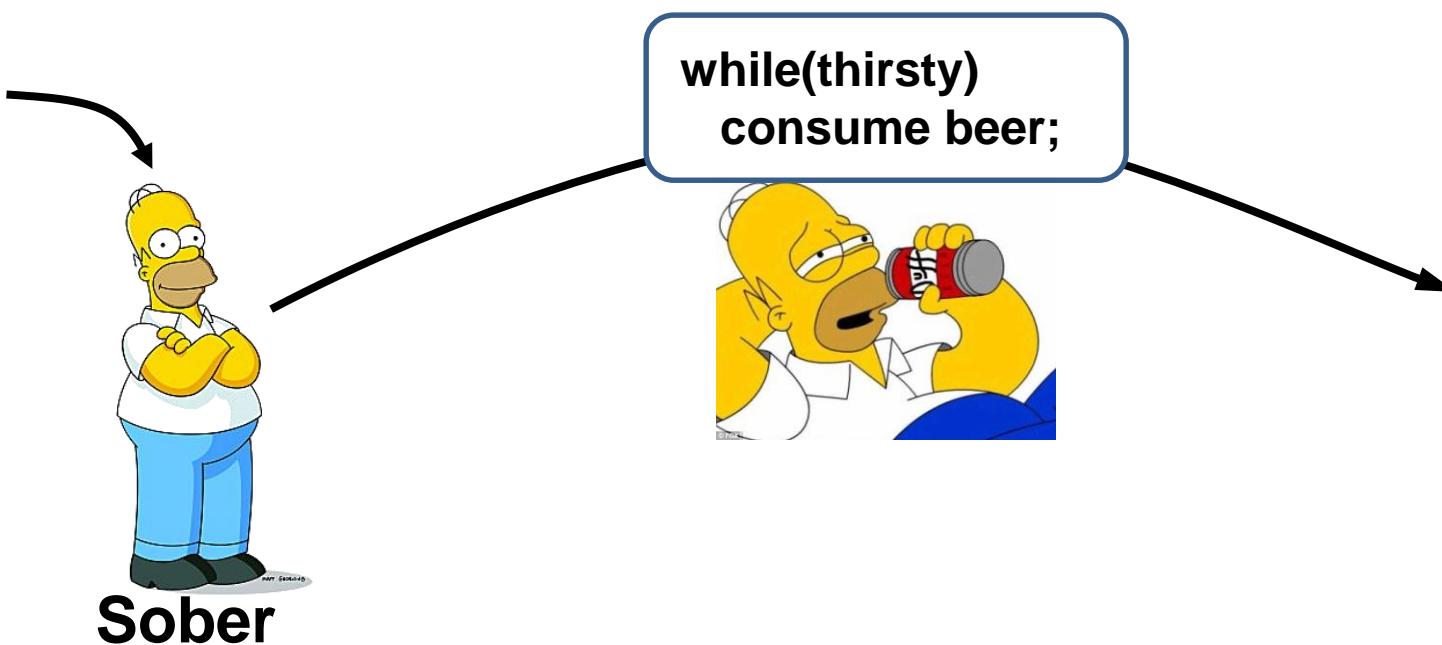
* Figure is courtesy of A. Tanenbaum



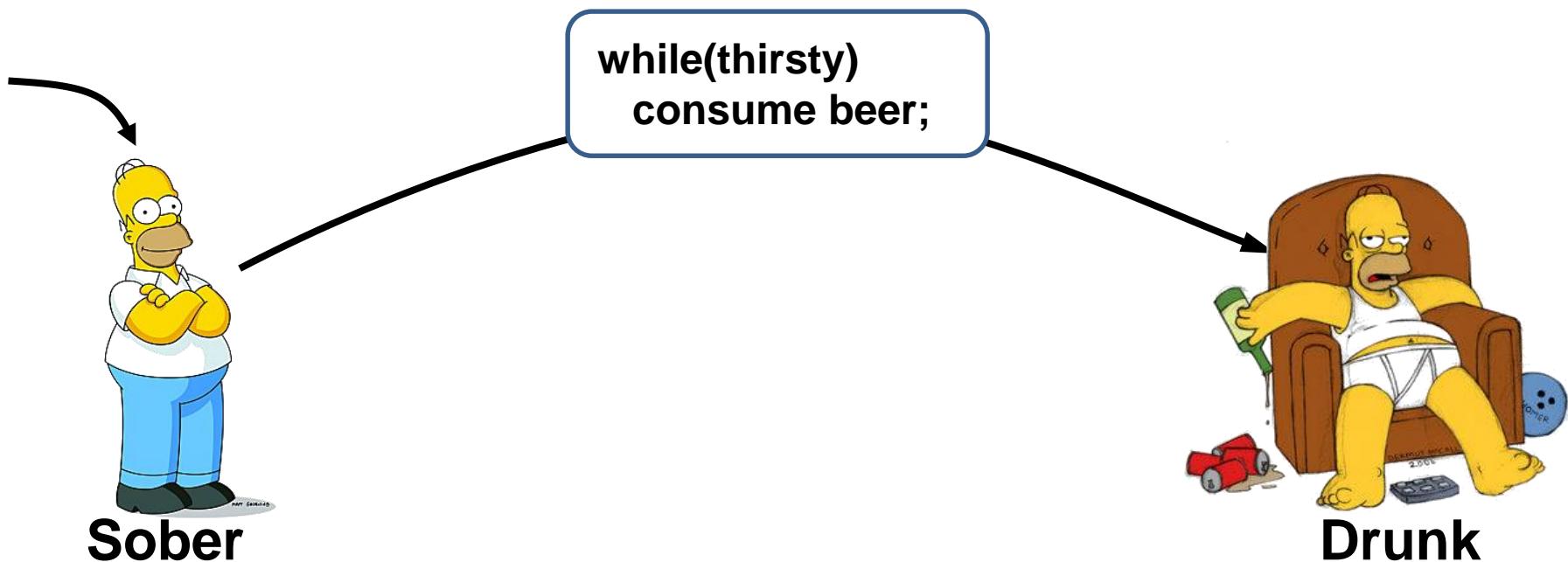
Simpson State Diagram



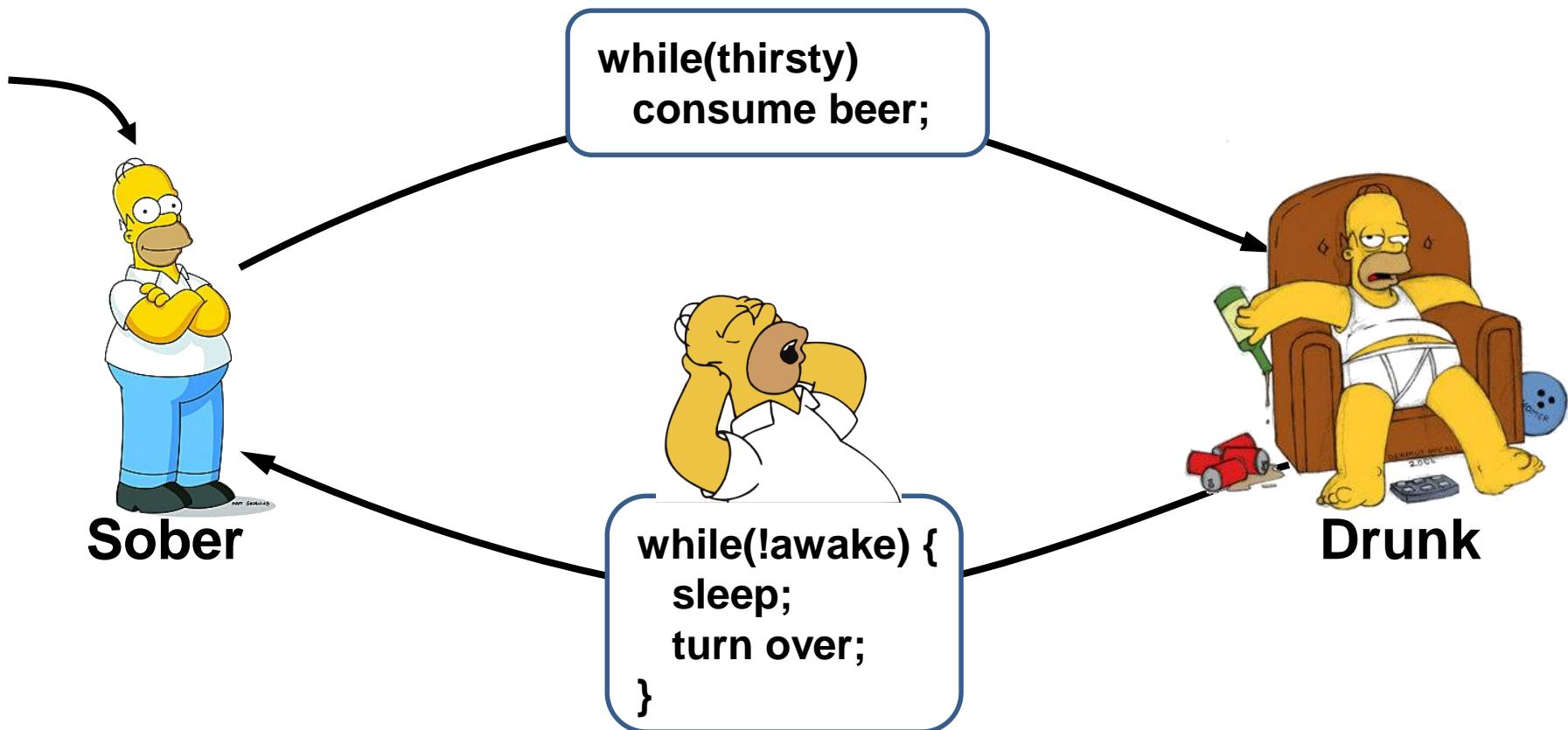
Simpson State Diagram



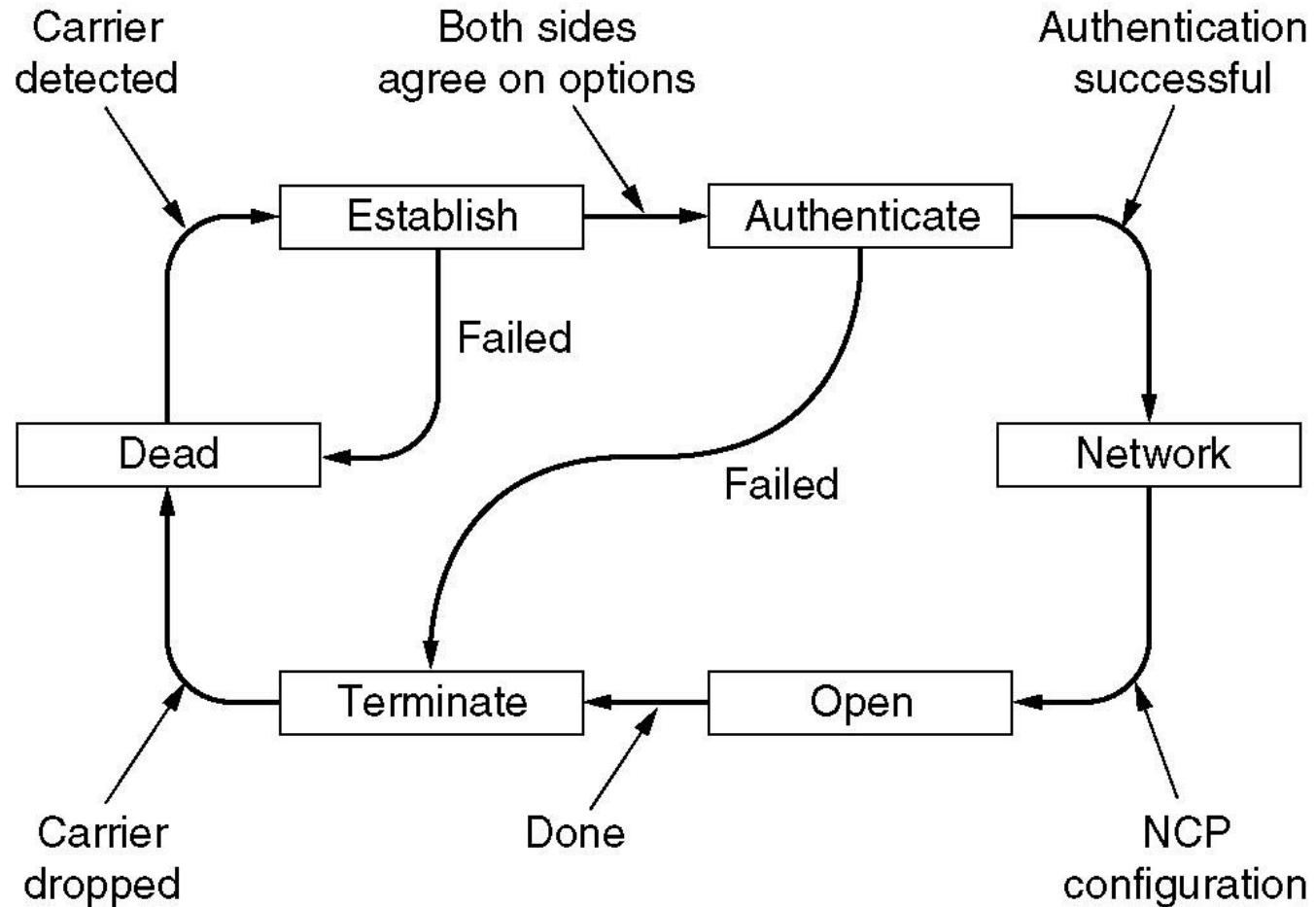
Simpson State Diagram



Simpson State Diagram



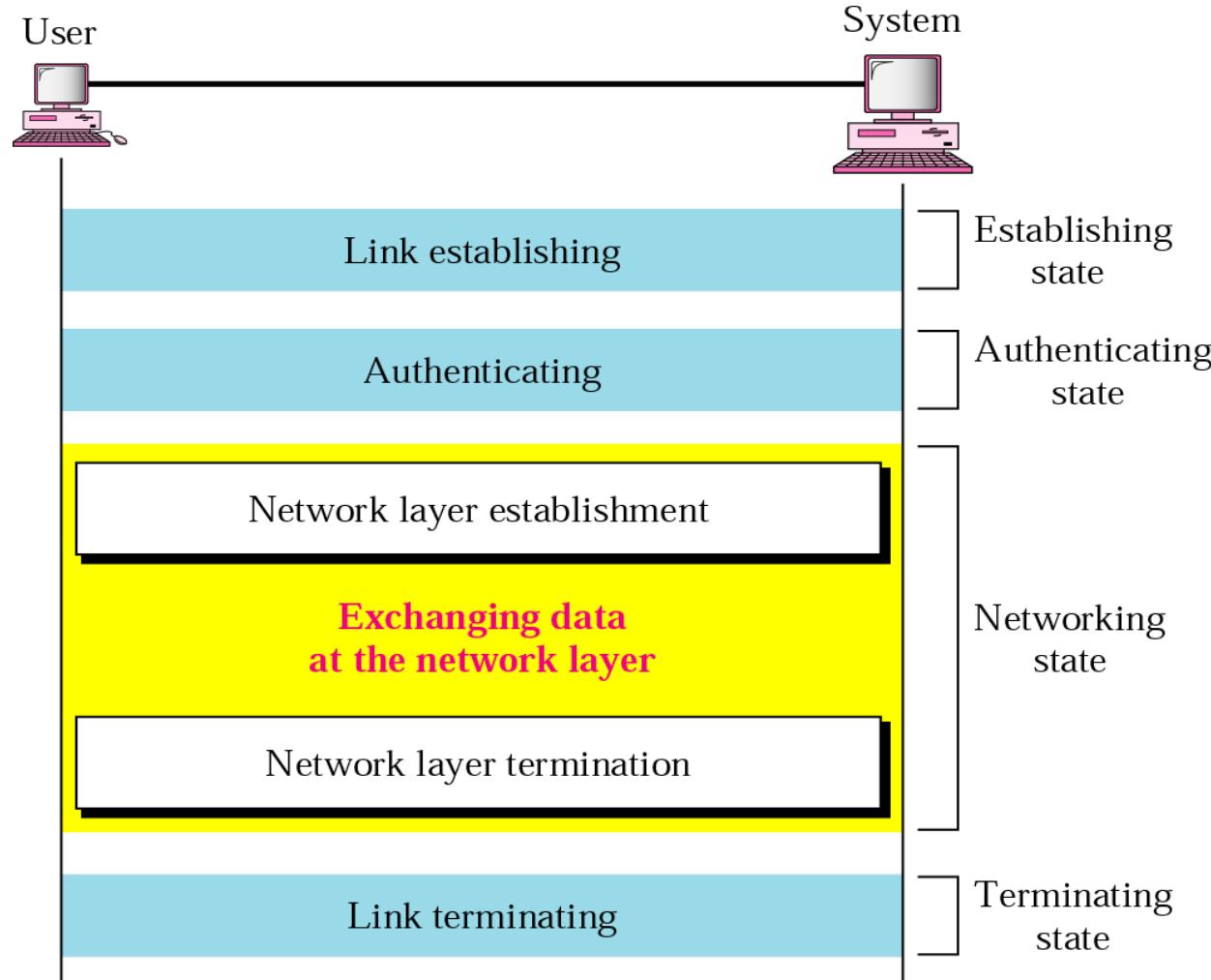
PPP – Life Cycle



* Figure is courtesy of A. Tanenbaum



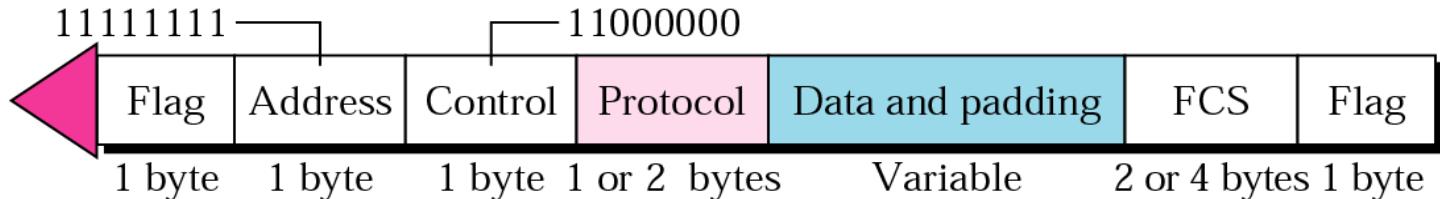
Tasks of PPP



* Figure is courtesy of B. Forouzan

PPP Frame

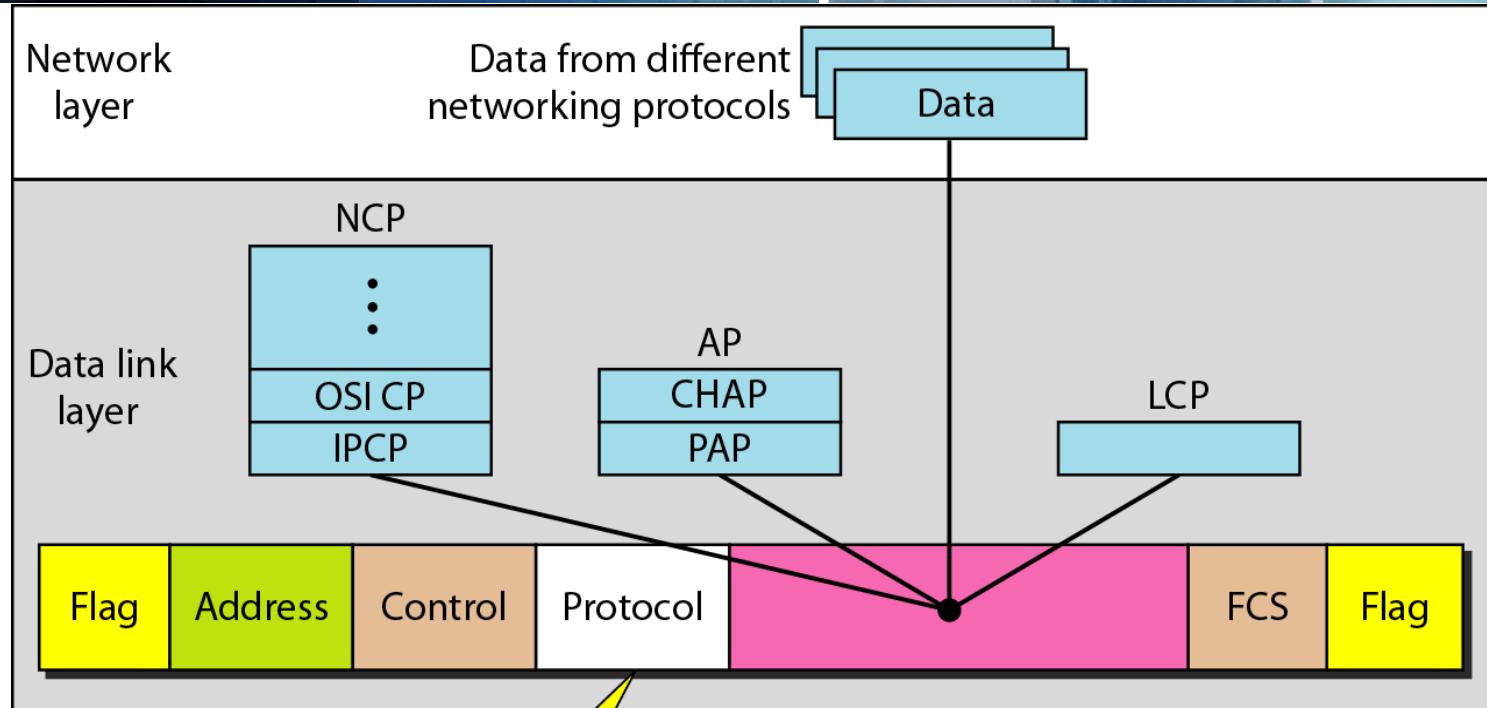
■ Modified HDLC frame:



- Byte-oriented Protocol
 - Flag Byte: 01111110
 - Escape Byte: 01111101
- FCS: 16- or 32-bit CRC
 - $x^{16} + x^{12} + x^5 + 1$
 - 1 0001 0000 0010 0001 → **16 bits remainder** ← **16-degree polynomial**
 - $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

* Figure is courtesy of B. Forouzan

PPP Components



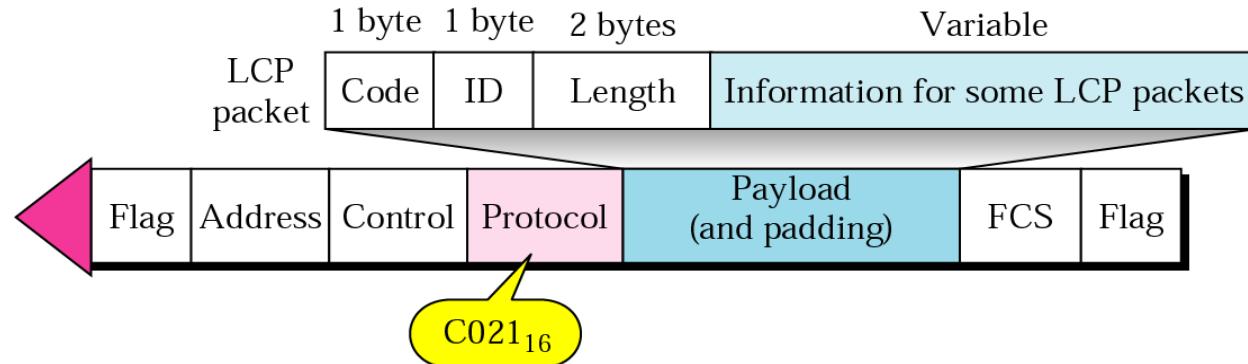
LCP: 0xC021
AP: 0xC023 and 0xC223
NCP: 0x8021 and
Data: 0x0021 and

LCP: Link Control Protocol
AP: Authentication Protocol
NCP: Network Control Protocol

* Figure is courtesy of B. Forouzan



LCP Packet



Code	Packet Type	Description
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Requests to shut down the line
0x06	Terminate-ack	Accepts the shut down request

* Figure is courtesy of B. Forouzan

Common Options

Option	Default
Maximum receive unit	1500
Authentication protocol	None
Protocol field compression	Off
Address and control field compression	Off

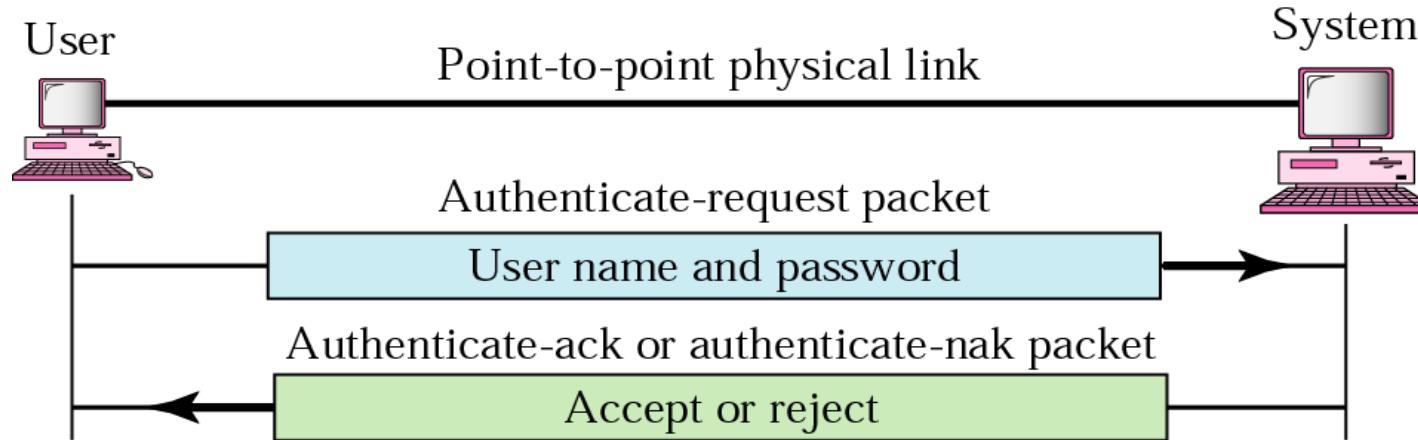


LCP Debug Codes

Code	Packet Type	Description
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet



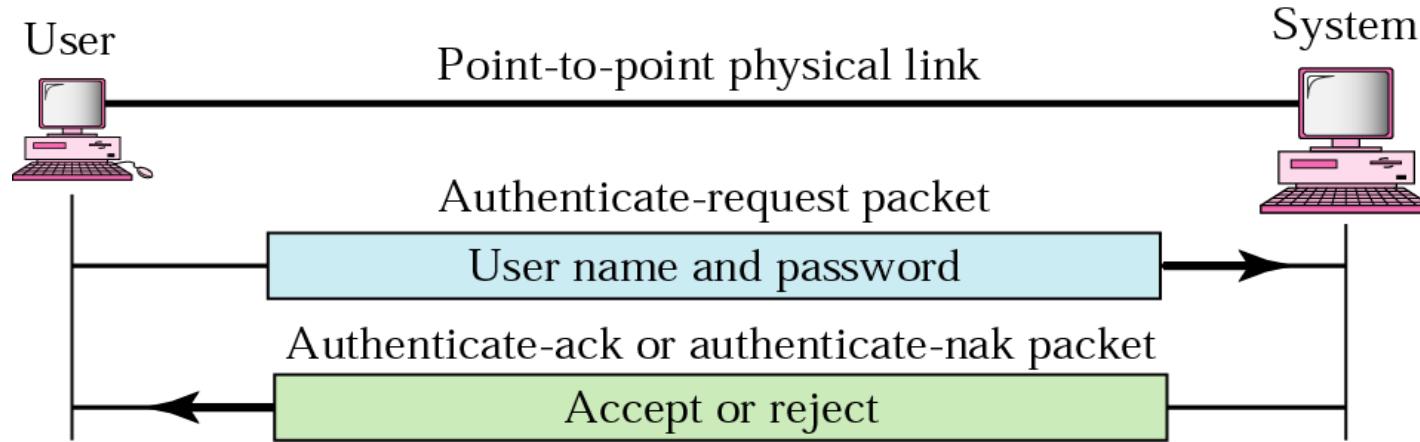
Password Authentication Protocol (PAP)



- 2-Way Handshake
- Password transmitted in clear text

* Figure is courtesy of B. Forouzan

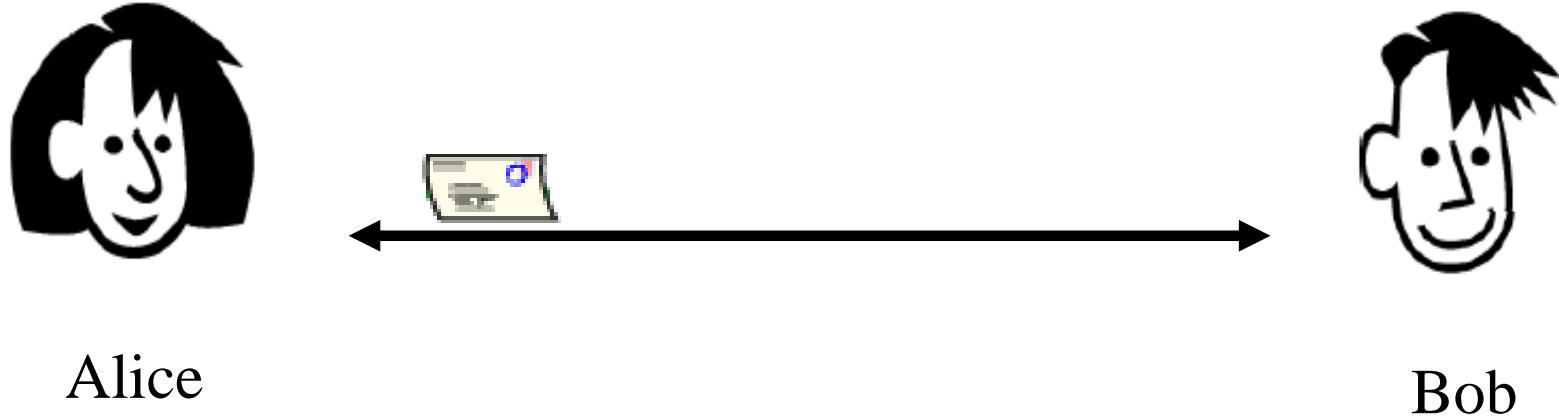
Password Authentication Protocol (PAP)



- 2-Way Handshake
- Password transmitted in clear text

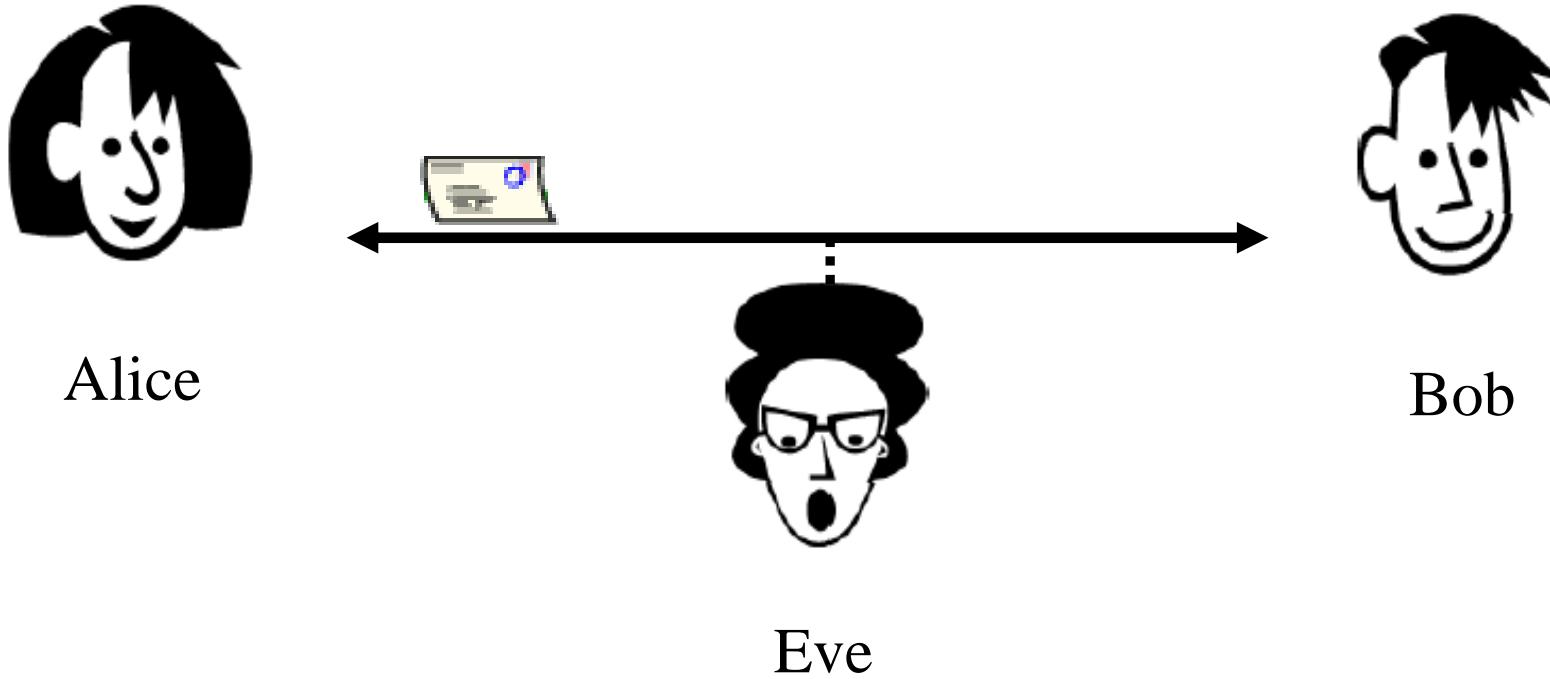
...what were
they thinking???
Never transfer
passwords in
clear text

Communication Scenario



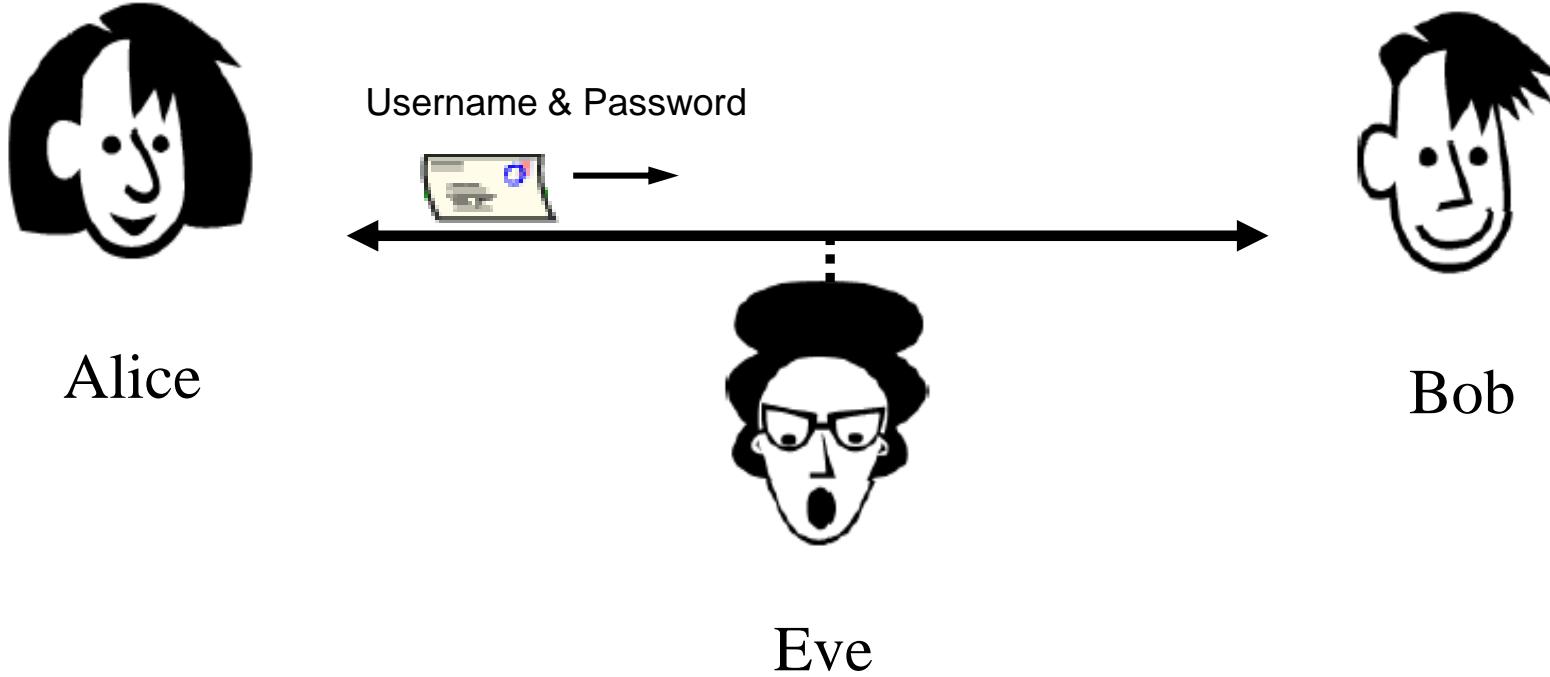
- Alice and Bob want to communicate

Threat: Eavesdropping



- Alice and Bob want to communicate
- Eve is eavesdropping (intercept, delete, add messages)

PAP Transfer

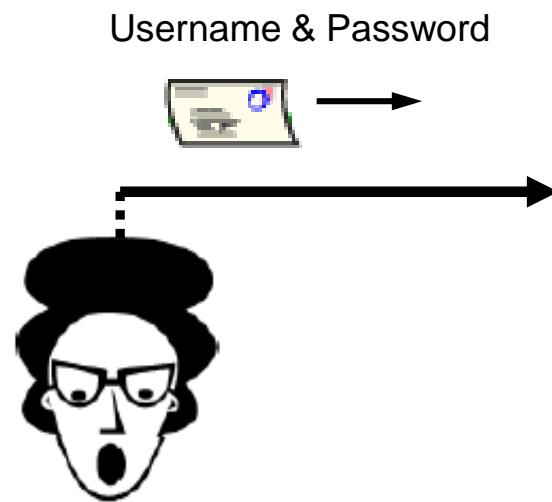


- Eve captures Alice's username & password

Playback Attack



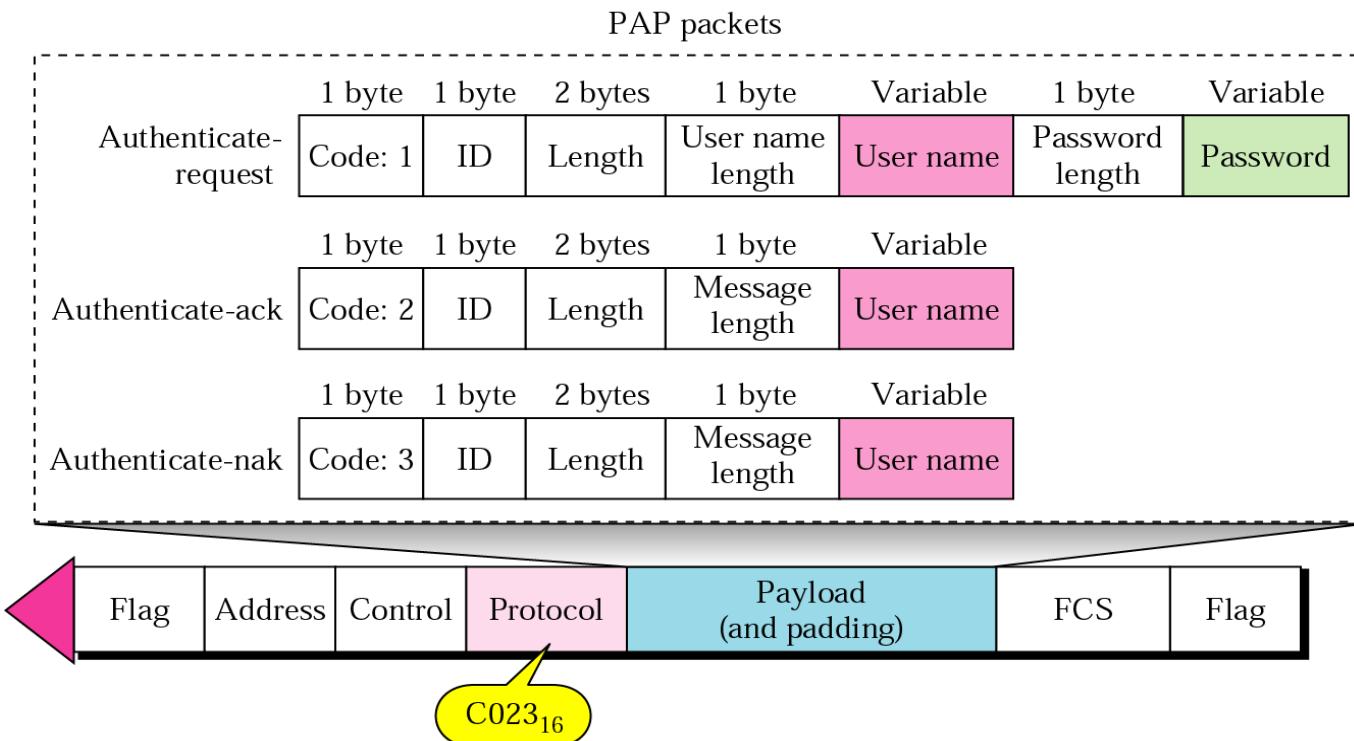
Alice



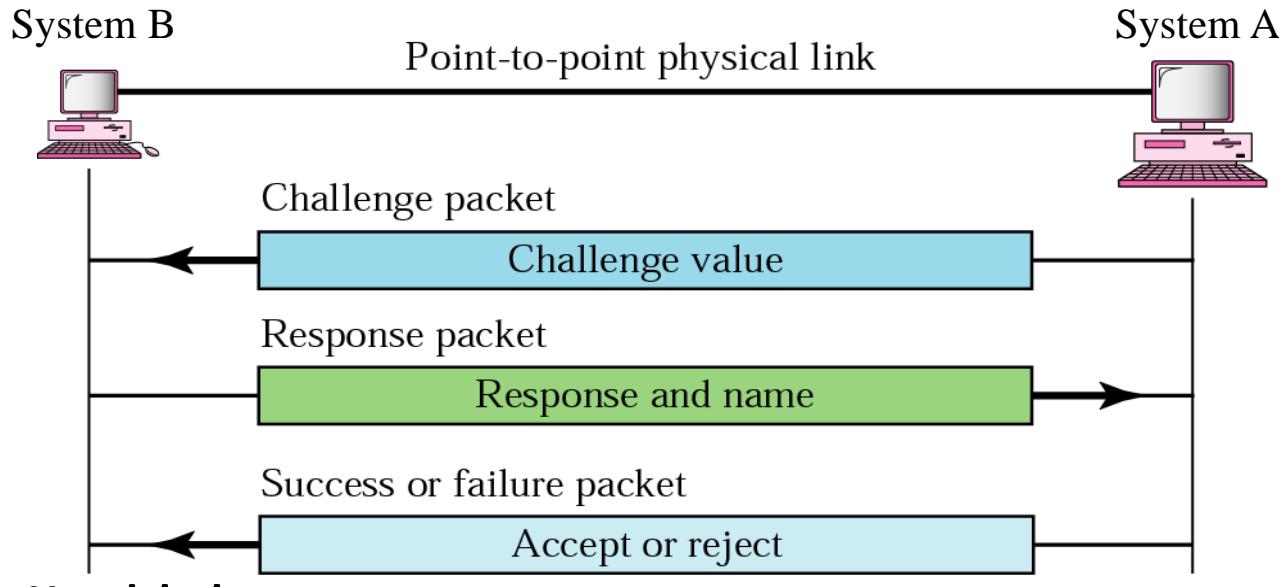
Bob

- Eve pretends to be Alice by using sending out the capture message

PAP Packets



CHAP

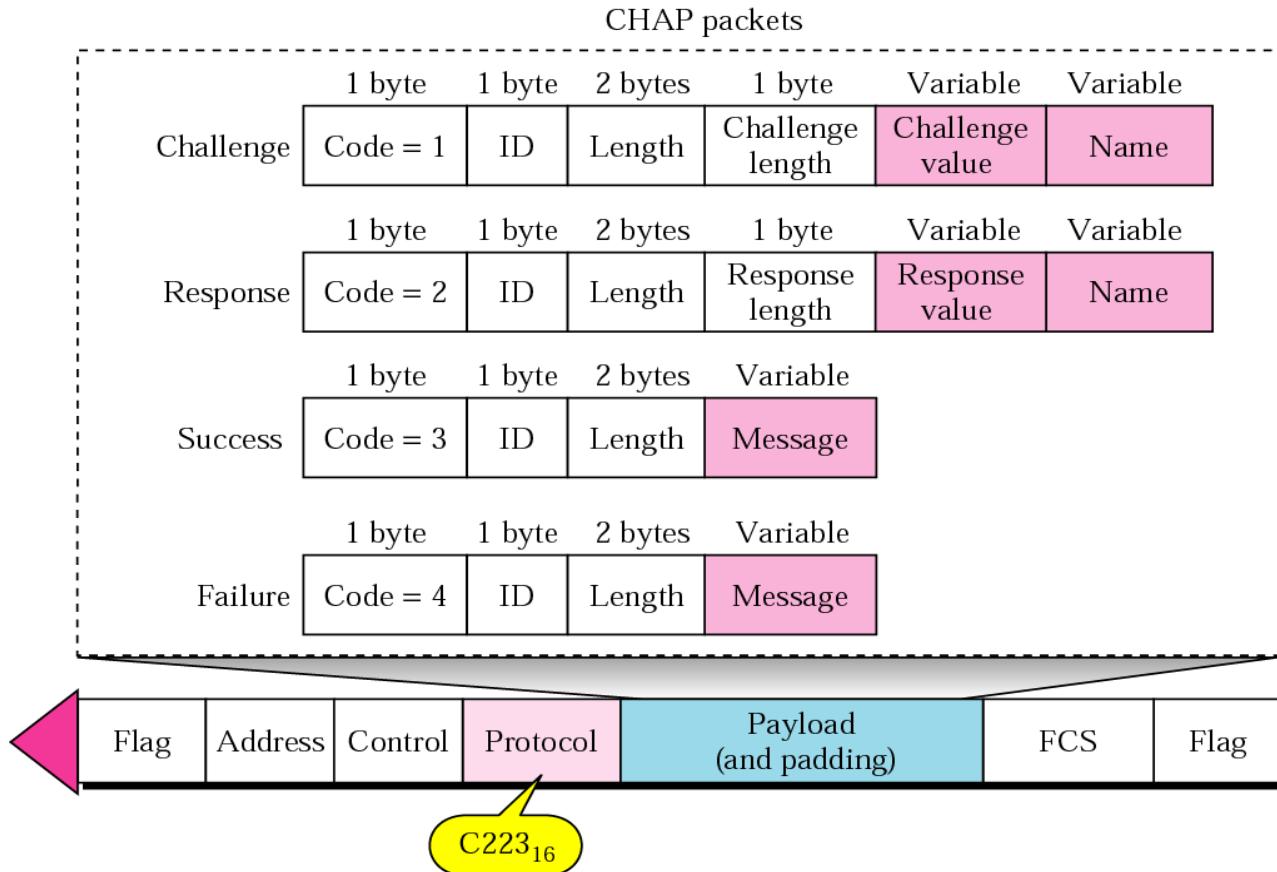


- **3-Way Handshake**

1. A creates challenge \Rightarrow challenge value
2. B processes challenge value with password \Rightarrow response
3. A compares response with own calculation \Rightarrow accepts or rejects response

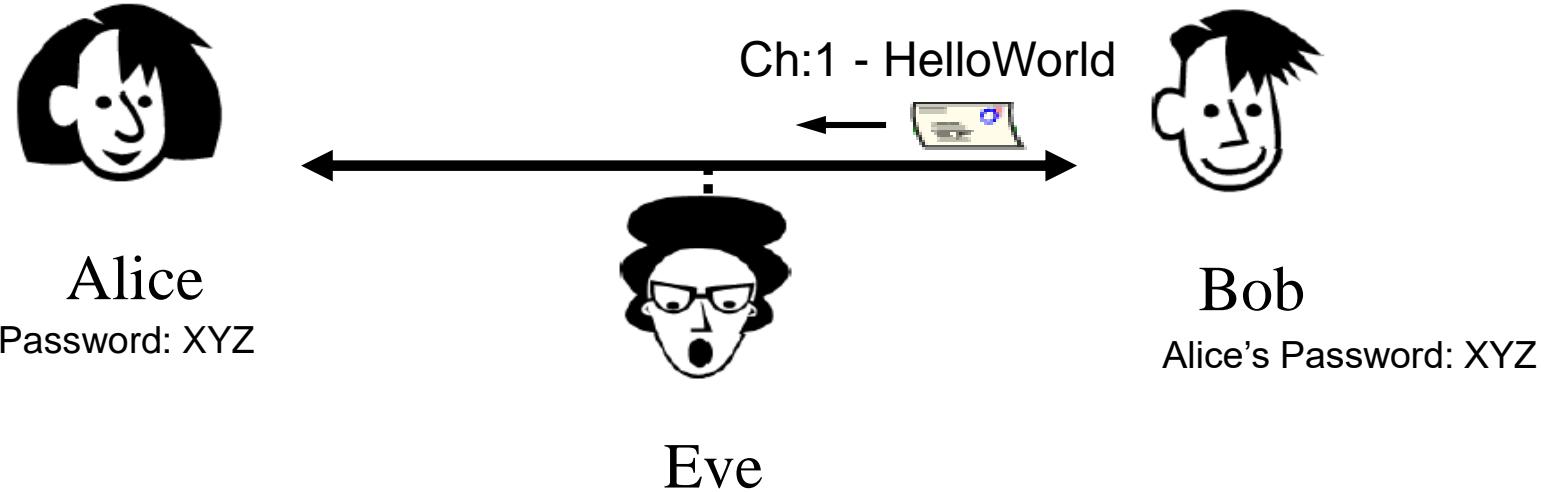
* Figure is courtesy of B. Forouzan

CHAP Packets

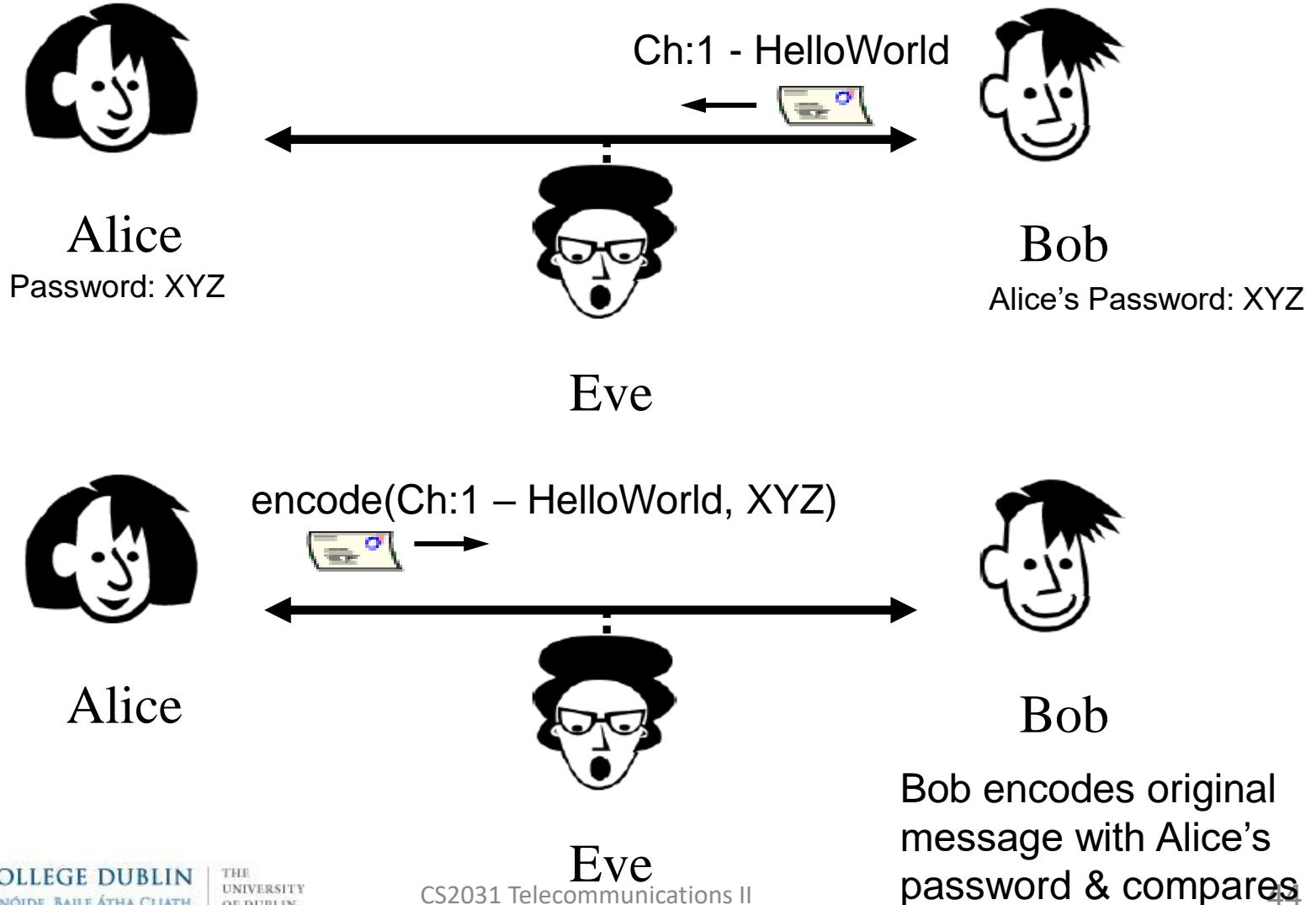


* Figure is courtesy of B. Forouzan

Chap Exchange



Chap Exchange



Chap Replay



Alice

encode(Ch:1 – HelloWorld, XYZ)



Bob

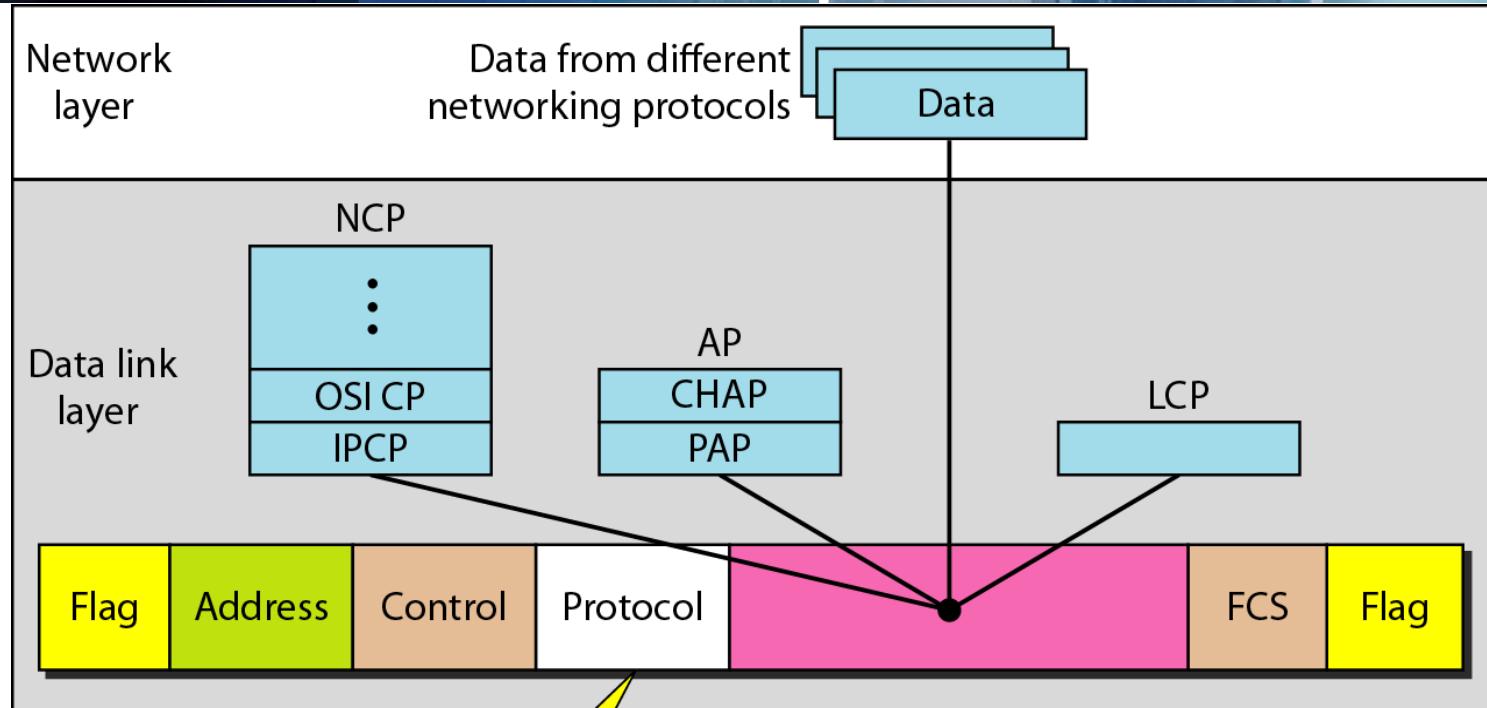


Eve

I've seen this already

- Eve can capture both messages
 - but will not see the password
 - or be able to use replay attack

PPP Components



LCP: Link Control Protocol
AP: Authentication Protocol
NCP: Network Control Protocol

* Figure is courtesy of B. Forouzan

PPP – Encapsulation

flag	addr	ctrl	protocol	data	CRC	flag
7E	FF	03	2	<= 1500	2	7E

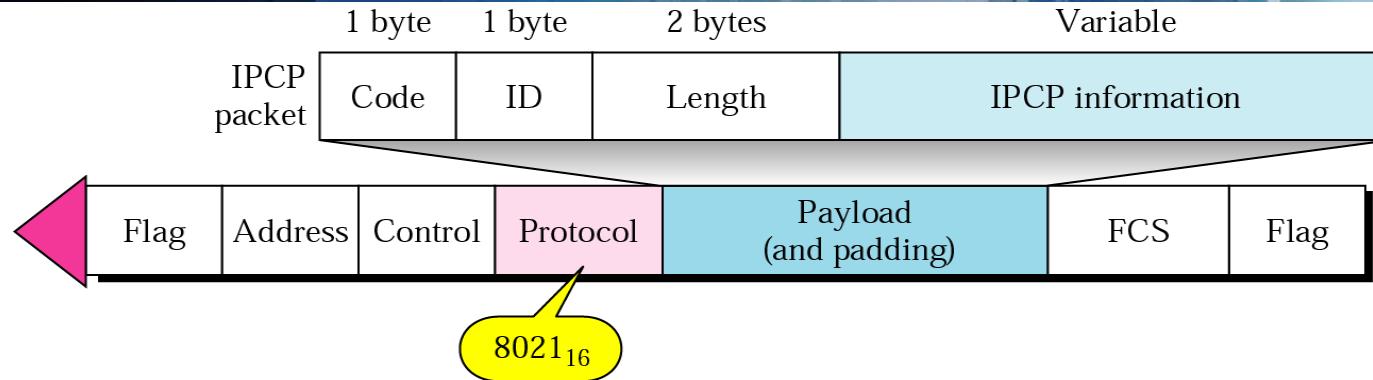


* Figure is courtesy of W. Stallings

Network Control Protocol (NCP)

- PPP negotiates Network Layer information
- Consists of specific protocols for network layer protocols
 - IP Control Protocol (IPCP)
 - IPX Control Protocol (IPXCP)
- May include IP Header compression

IPCP Packet in PPP Packet



Code	IPCP Packet
01	Configure Request
02	Configure ACK
03	Configure NAK
04	Configure Reject
05	Terminate-request
06	Terminate-ack
07	Code-reject

* Figure is courtesy of B. Forouzan

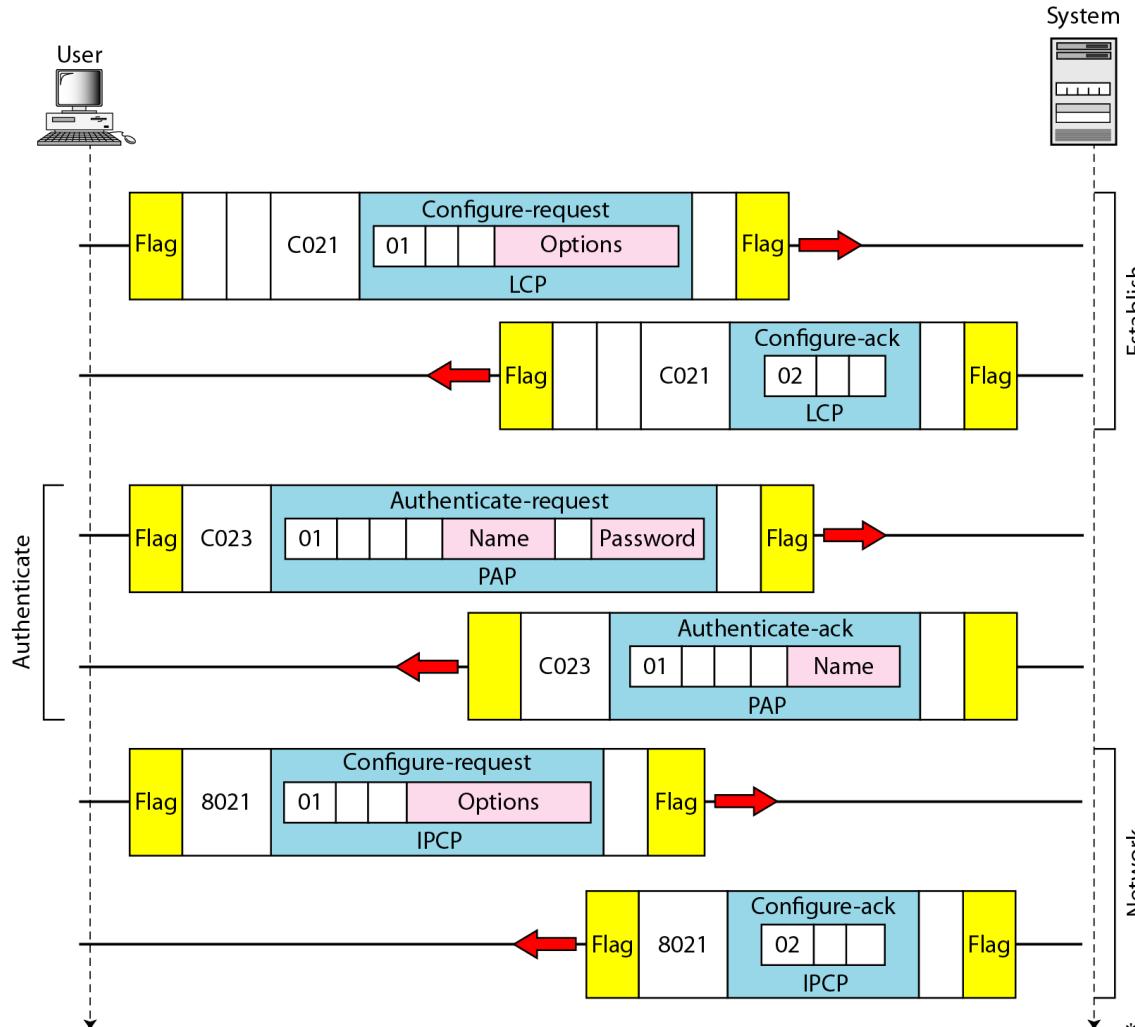
IP Packet Encapsulation



- PPP frame carries IP Packet as payload

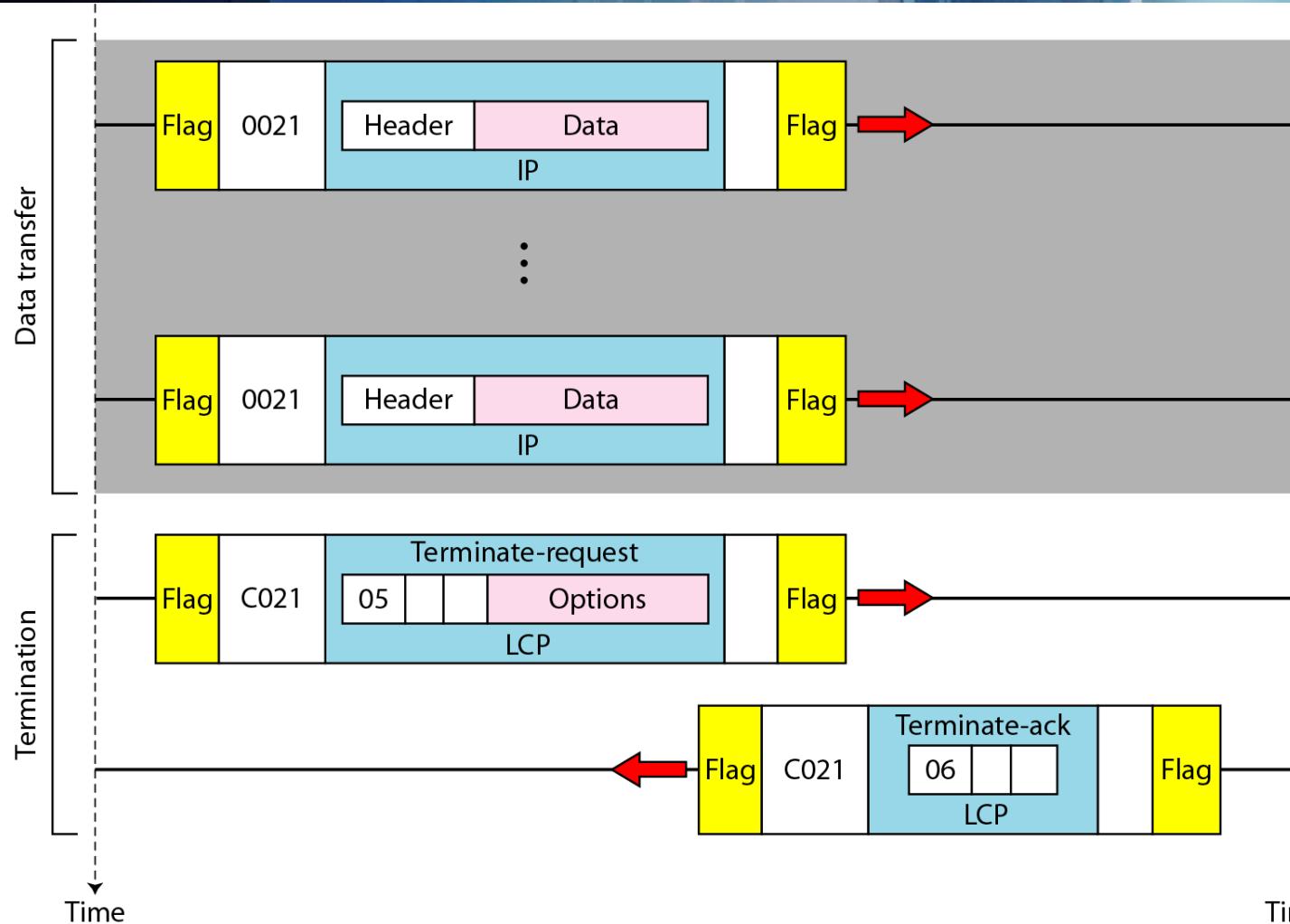


Connection Setup & Authentication



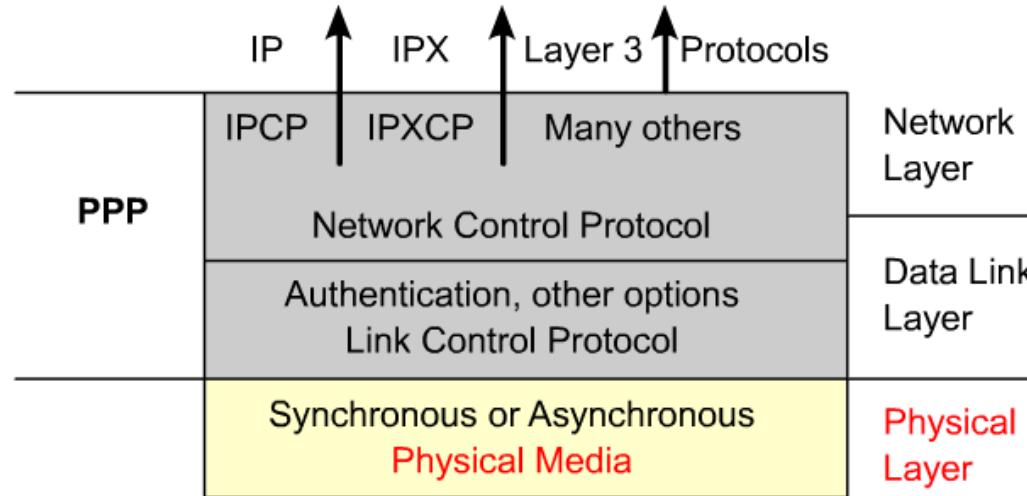
* Figure is courtesy of B. Forouzan

Data Transfer & Termination



* Figure is courtesy of B. Forouzan

PPP in OSI Stack

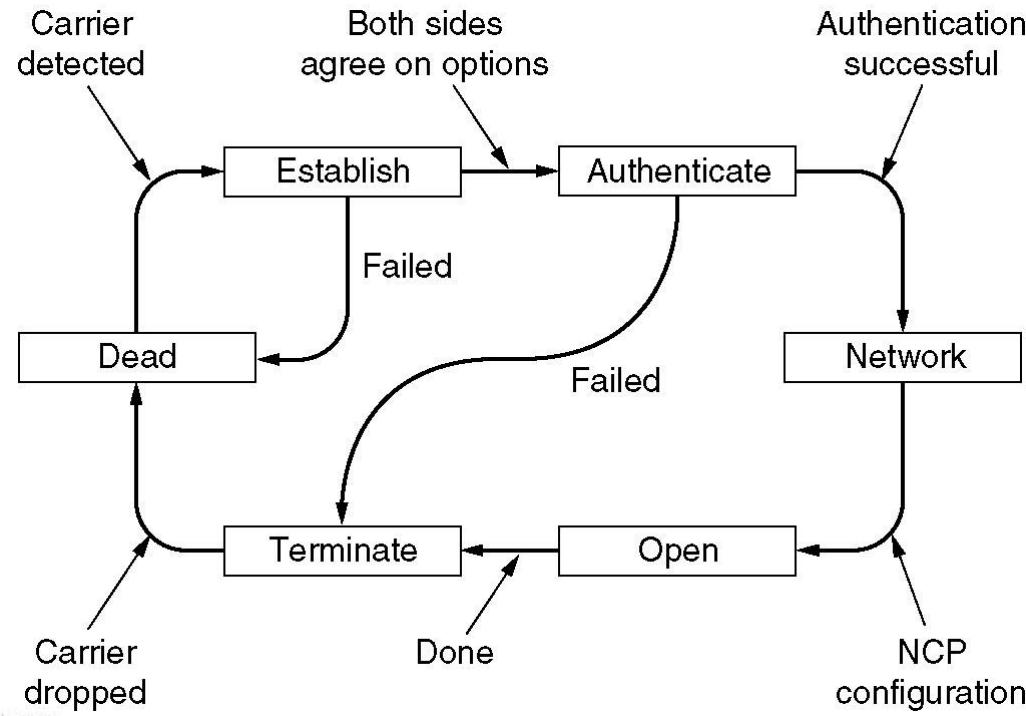


- PPP doesn't implement:
 - error correction/recovery (but error detection)
 - flow control
 - ordered delivery
 - multipoint links
- ⇒ all relegated to higher layers!

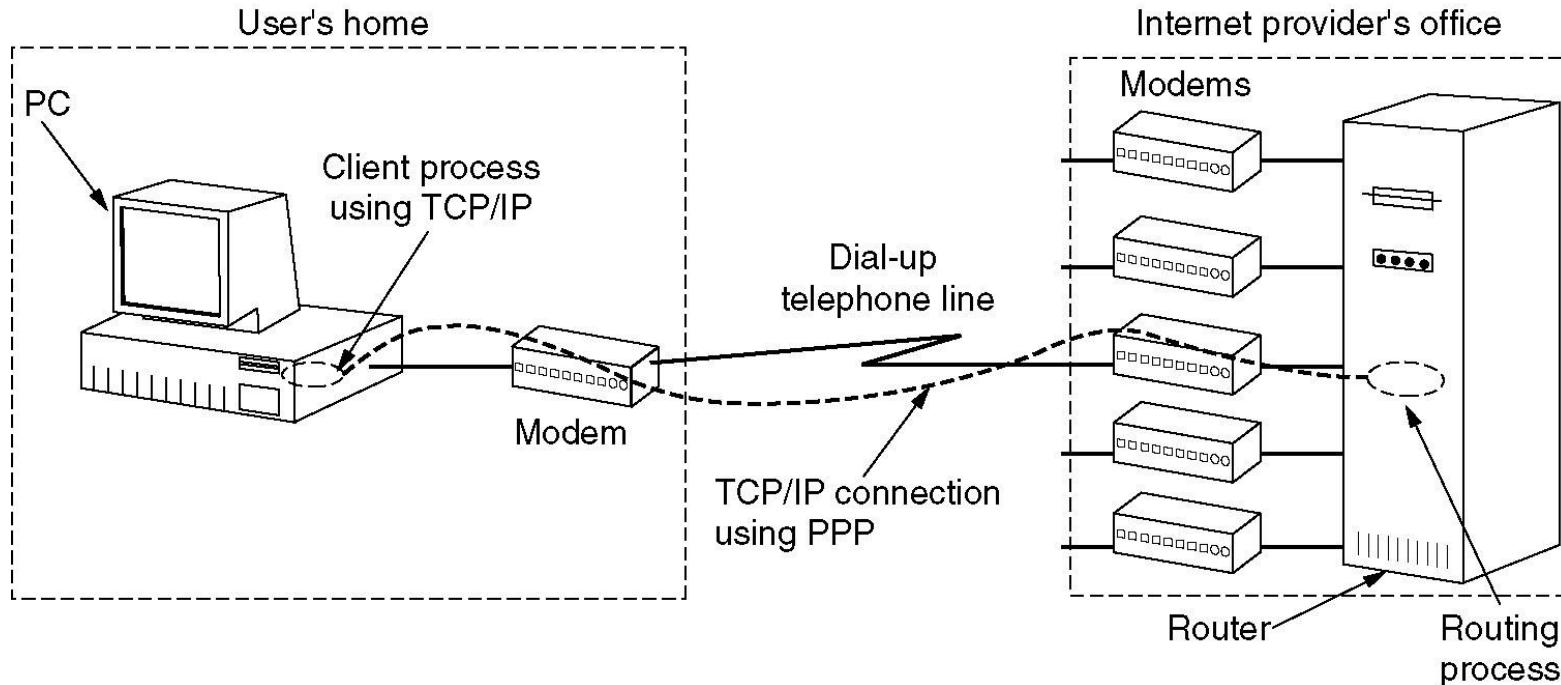
* Figure is courtesy of Cisco Corp.

PPP

- Connection Establishment & Termination
- Format Negotiation
- Authentication



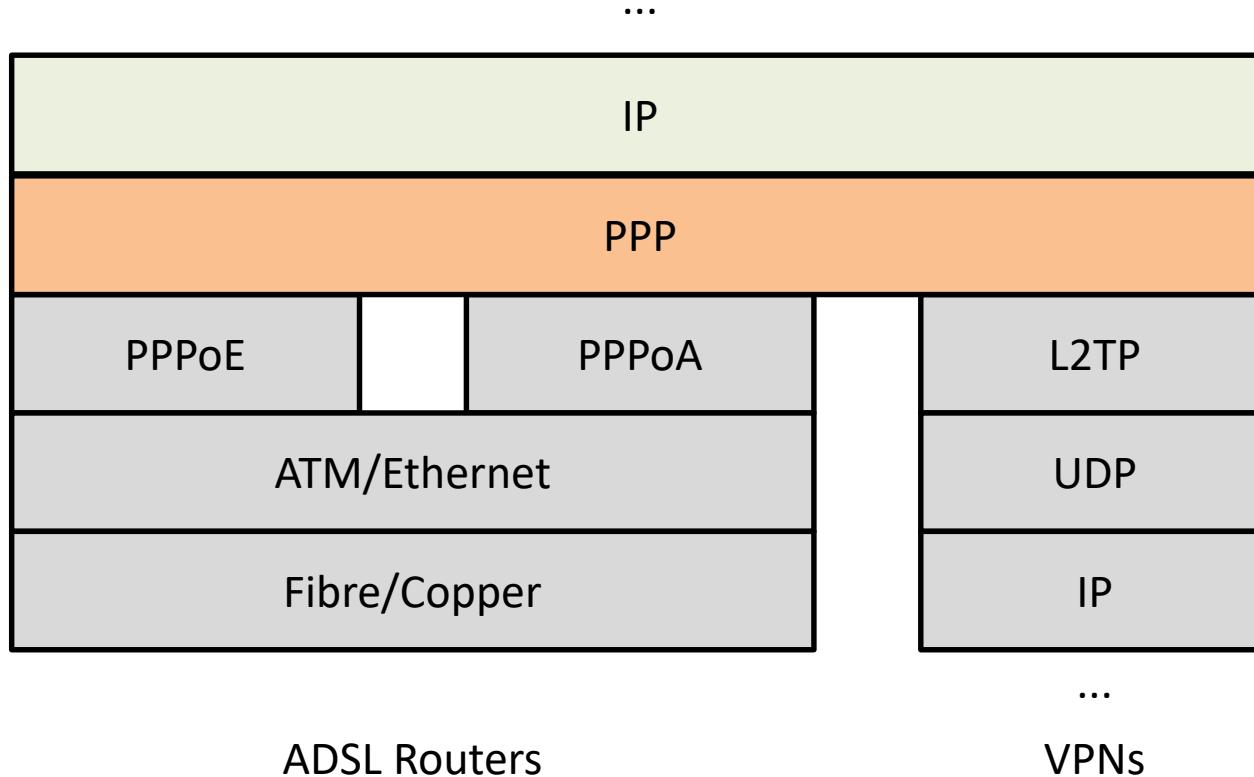
PPP – Why???



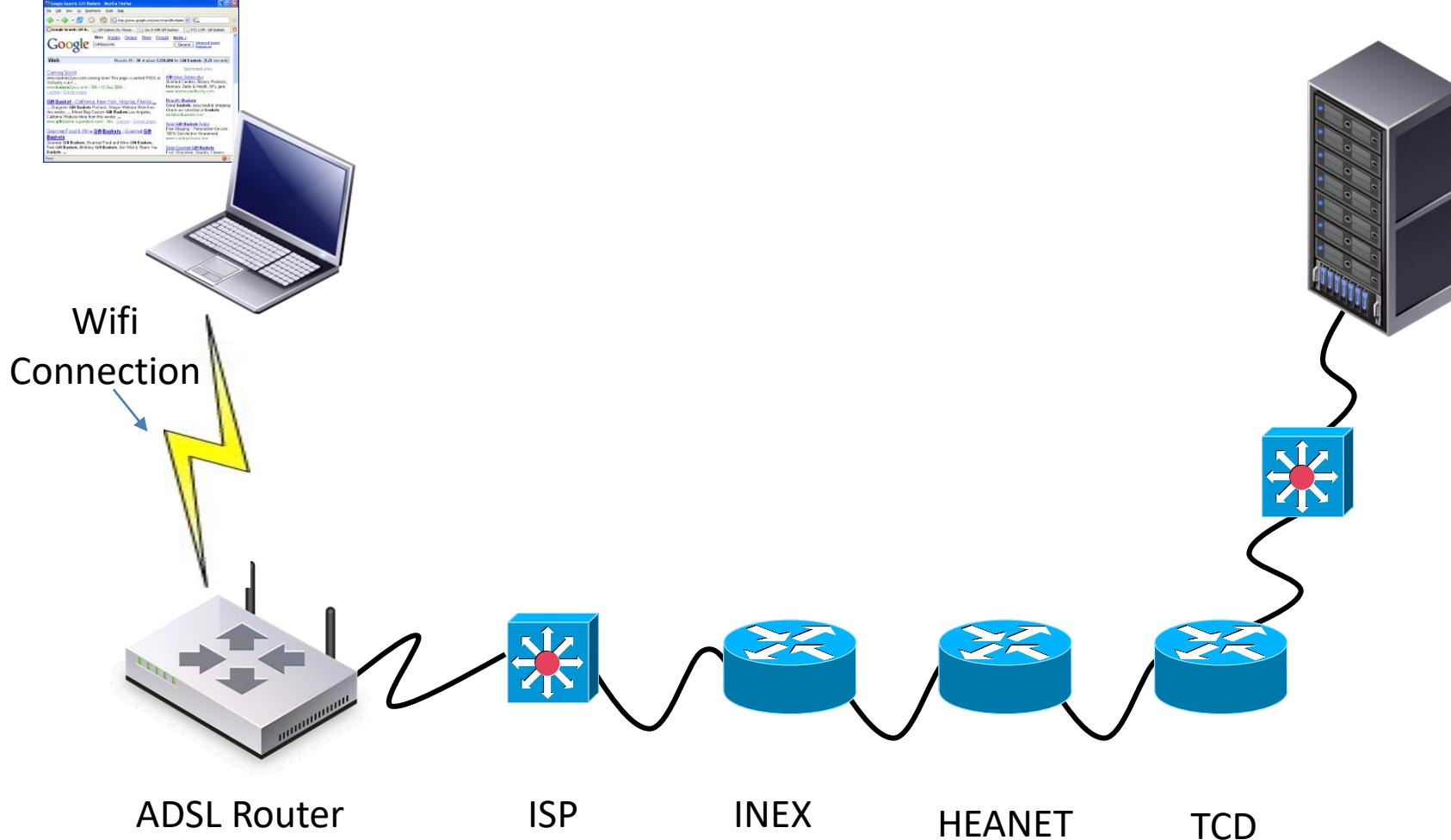
* Figure is courtesy of A. Tanenbaum



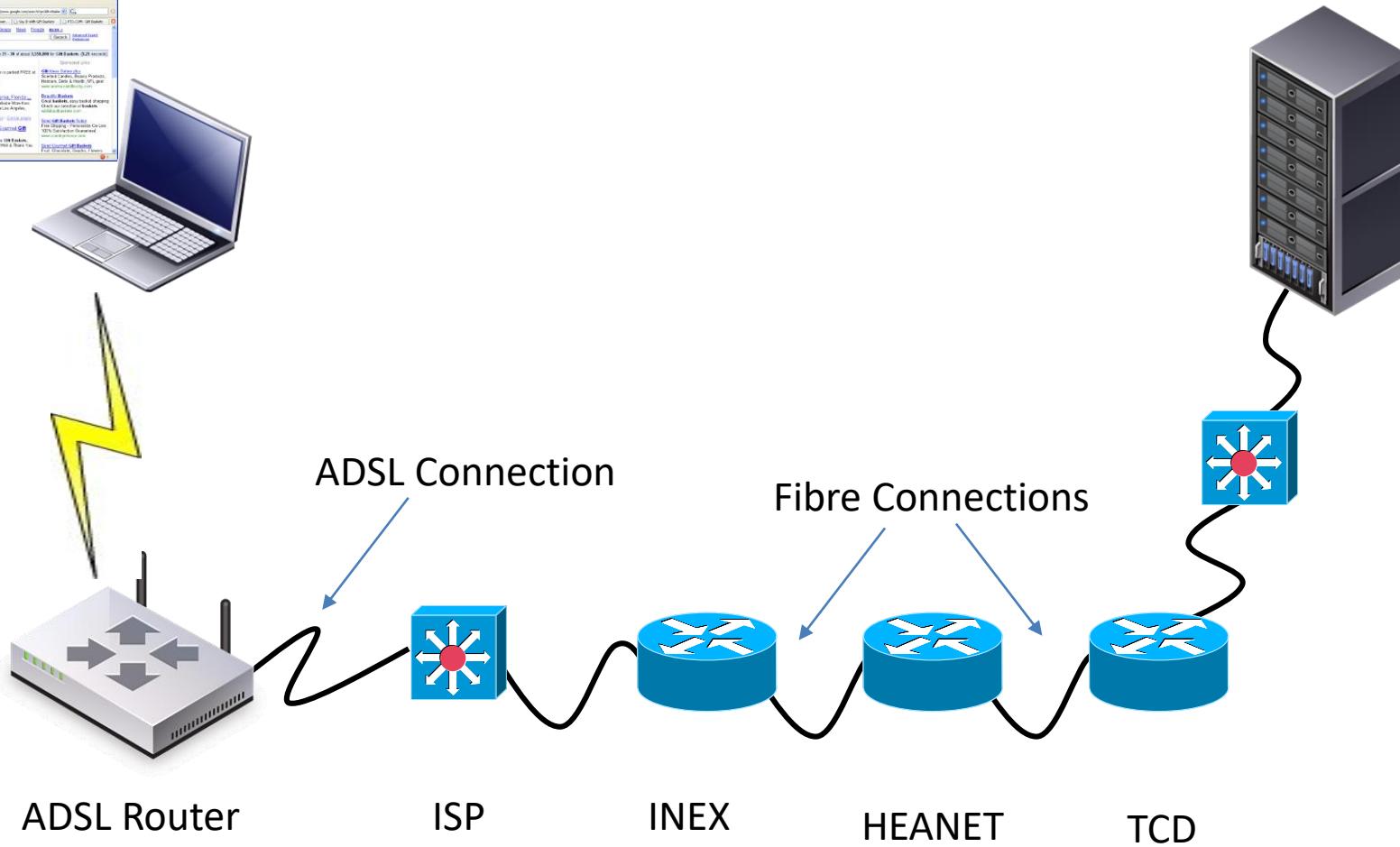
PPP – As Foundation for IP

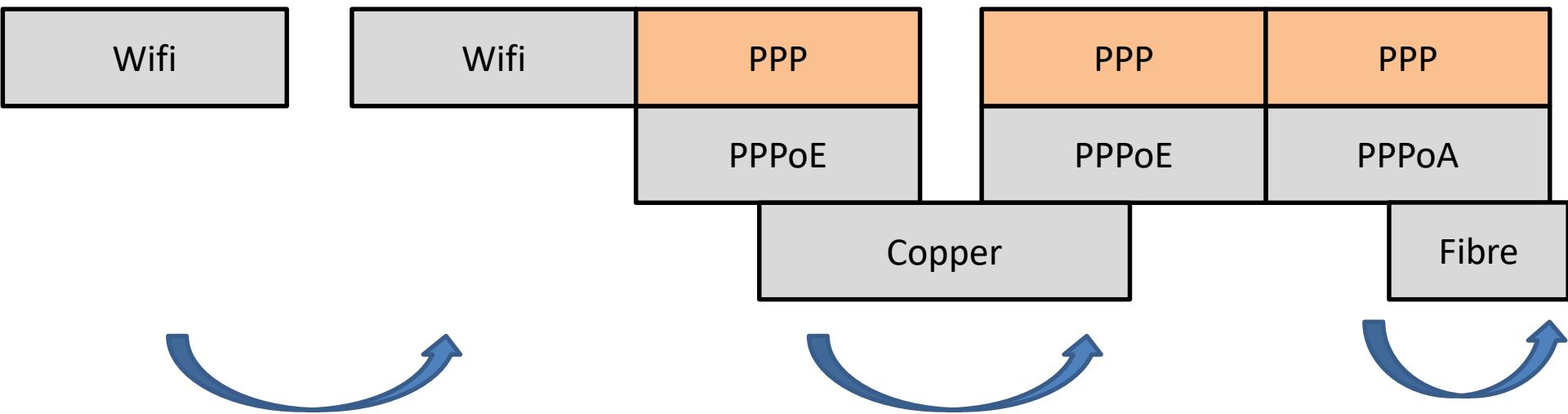


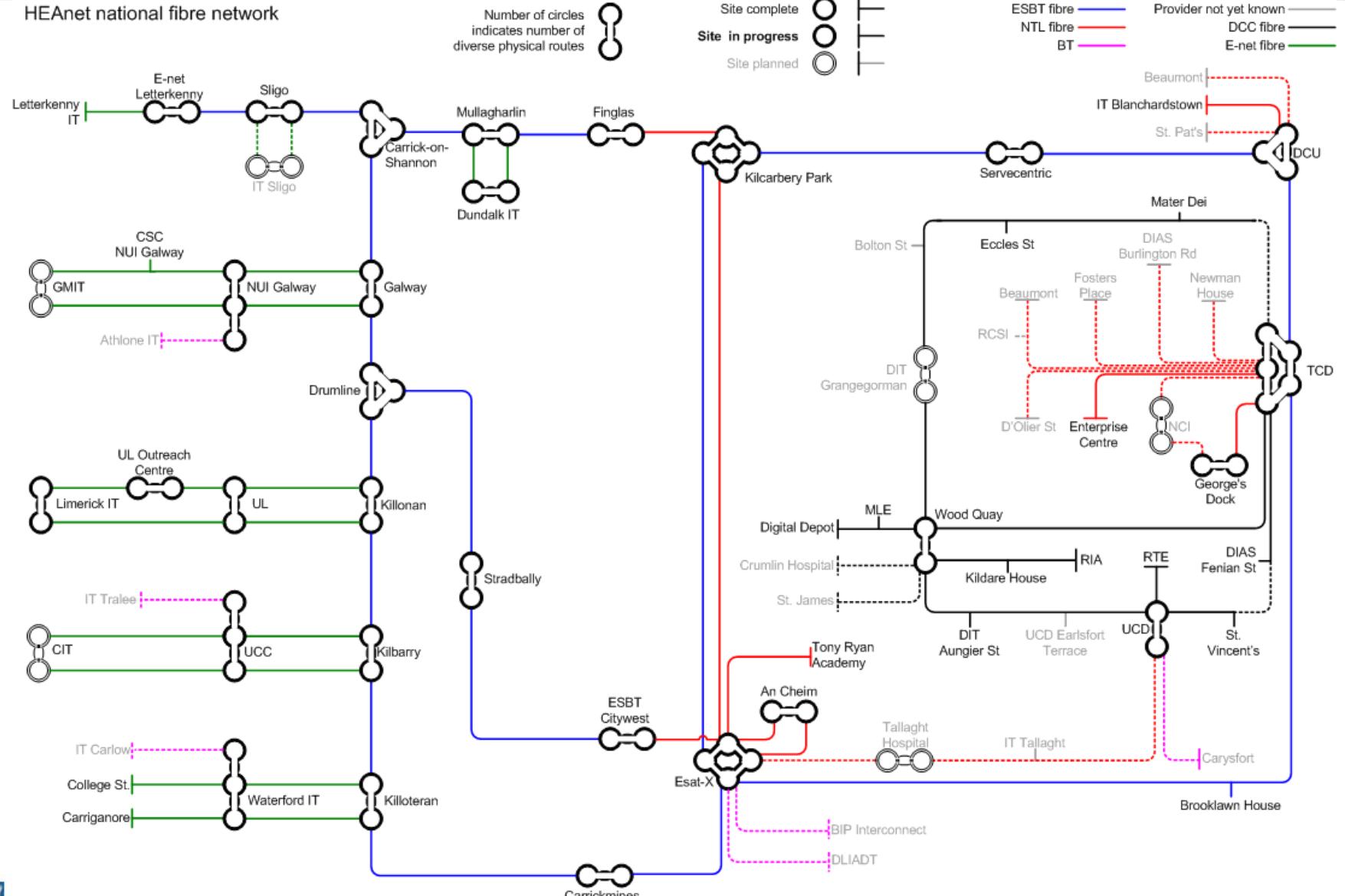
HTML Use Case



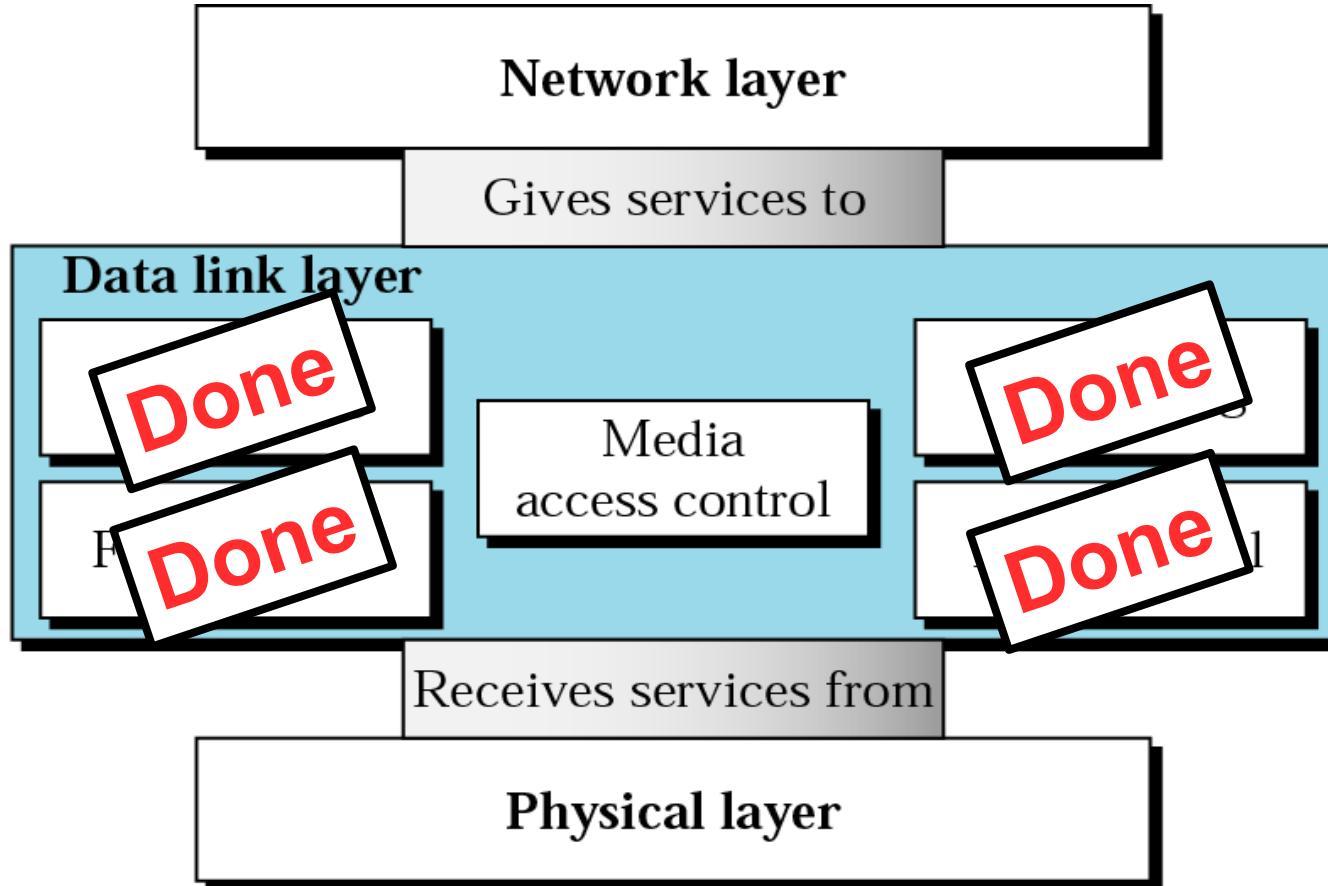
PPP in the HTML Use Case







Link Layer



* Figure is courtesy of B. Forouzan



Assignment Clarifications

- Implement 2 out of 3 approaches; either
 - Stop&Wait and Go-Back-N
 - or
 - Stop&Wait and Selective Repeat
- Receivers are reactive
 - ACKs are never retransmitted





Overview

- Link Layer
 - Error Detection
 - Flow Control
 - HDLC – Frames/Control Bytes
 - PPP – Lifecycle/State Diagram
 - Medium Access Control
 - 802.11 DCF & PCF
 - CDMA & Ethernet
 - Bridges & Switches
- Network Layer



Overview

- Link Layer
 - Error Detection
 - Flow Control
 - HDLC – Frames/Control Bytes
 - PPP – Lifecycle/State Diagram
 - Medium Access Control
 - 802.11 DCF & PCF
 - CDMA & Ethernet
 - Bridges & Switches
- Network Layer





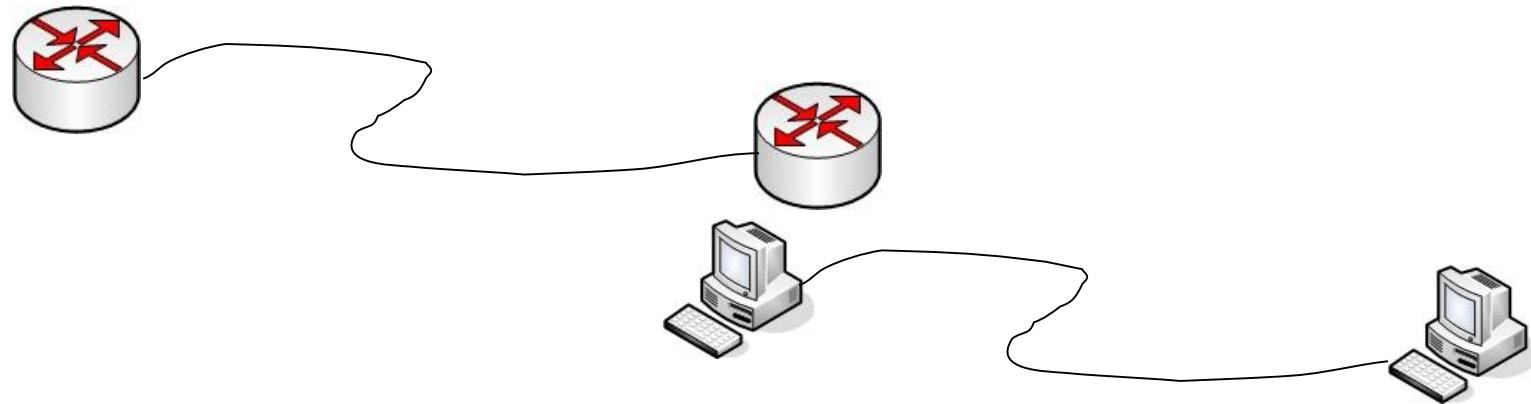
CS2031

Telecommunications II

Medium Access Control



HDLC / PPP



Primary

Command

Secondary

Response

a. Point-to-point

Primary

Command

Secondary

Secondary

Response

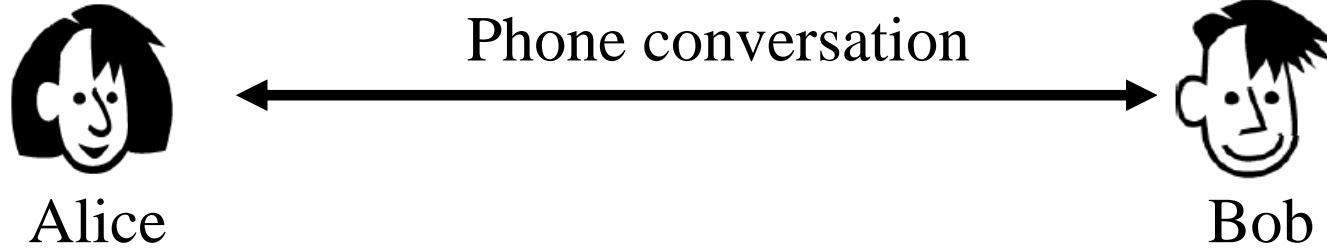
Response

b. Multipoint

* Figure is courtesy of B. Forouzan

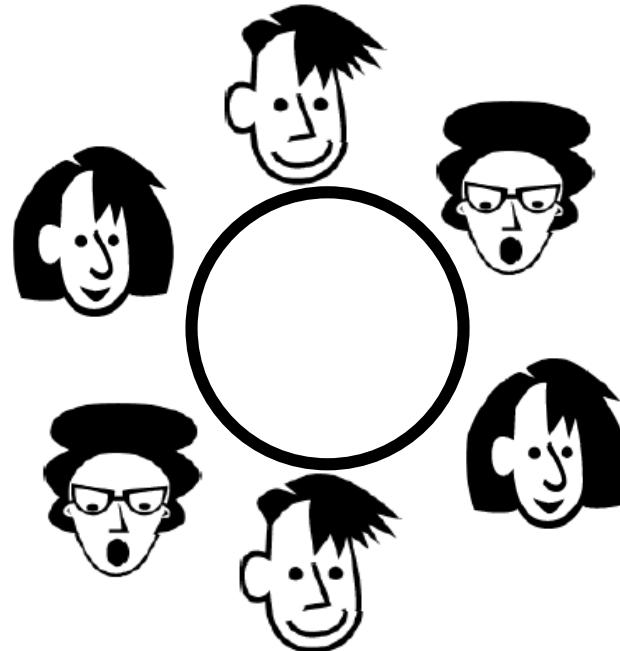


Analogy: Point-to-Point Communication



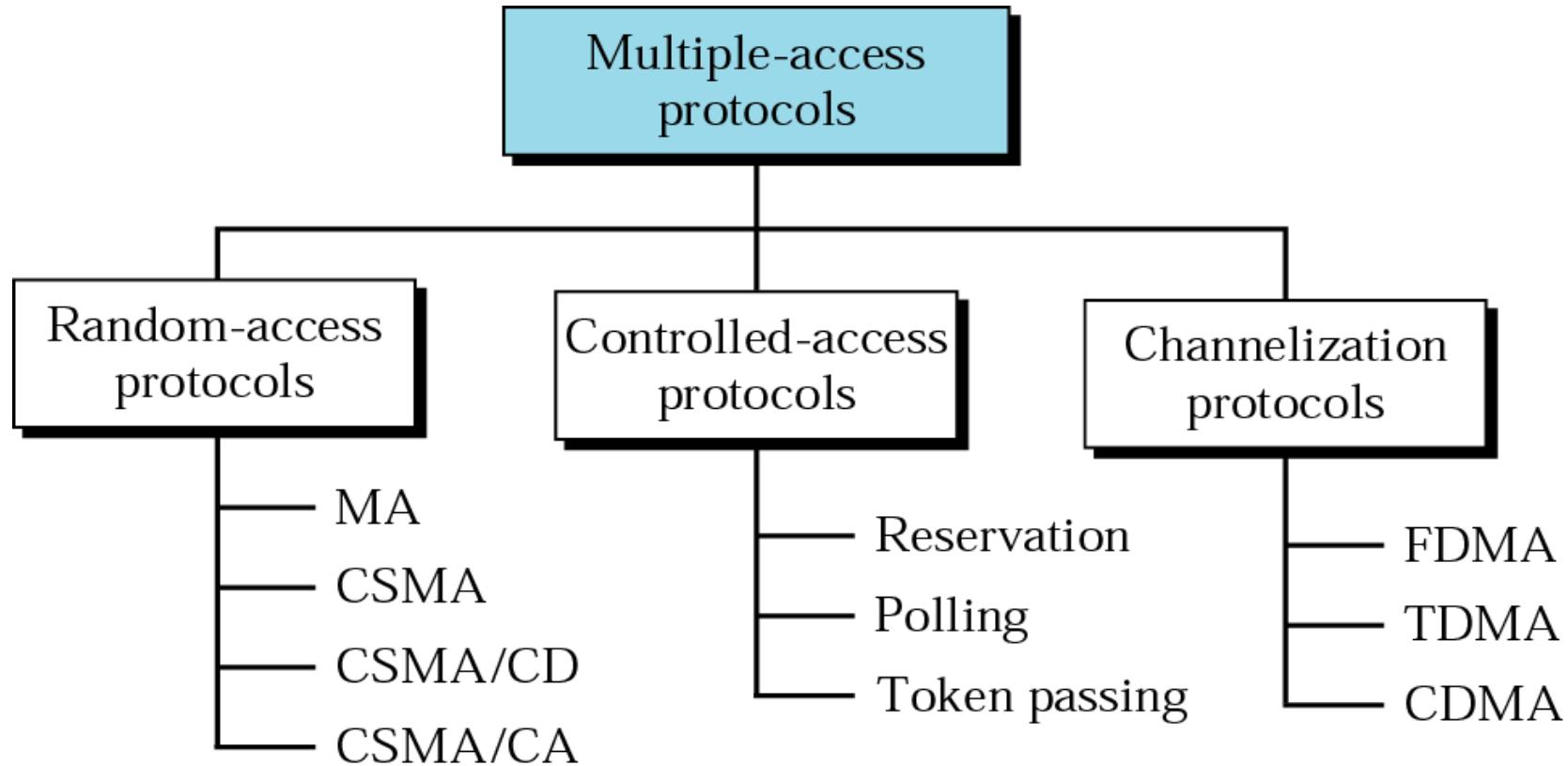
- Synchronization: Simple!

Analogy: Shared Medium



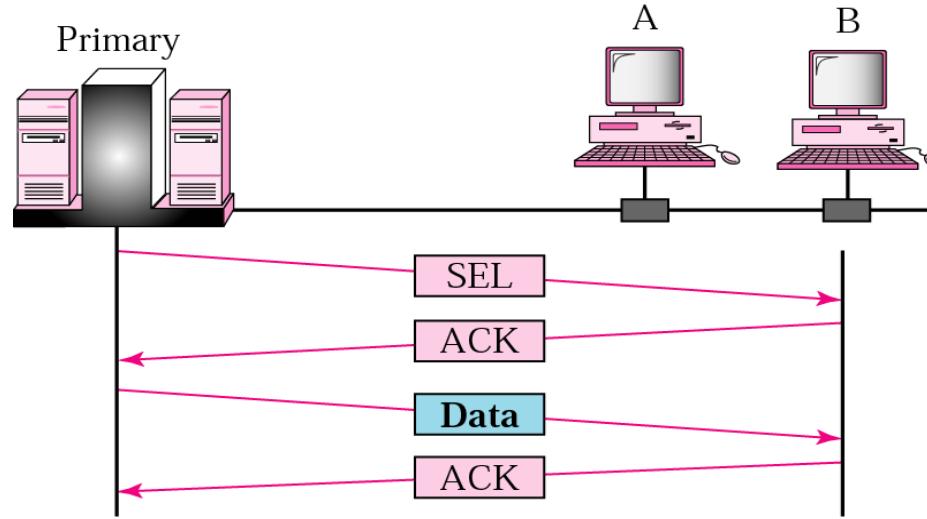
- Synchronisation: More complex

Multiple-Access Protocols



* Figure is courtesy of B. Forouzan

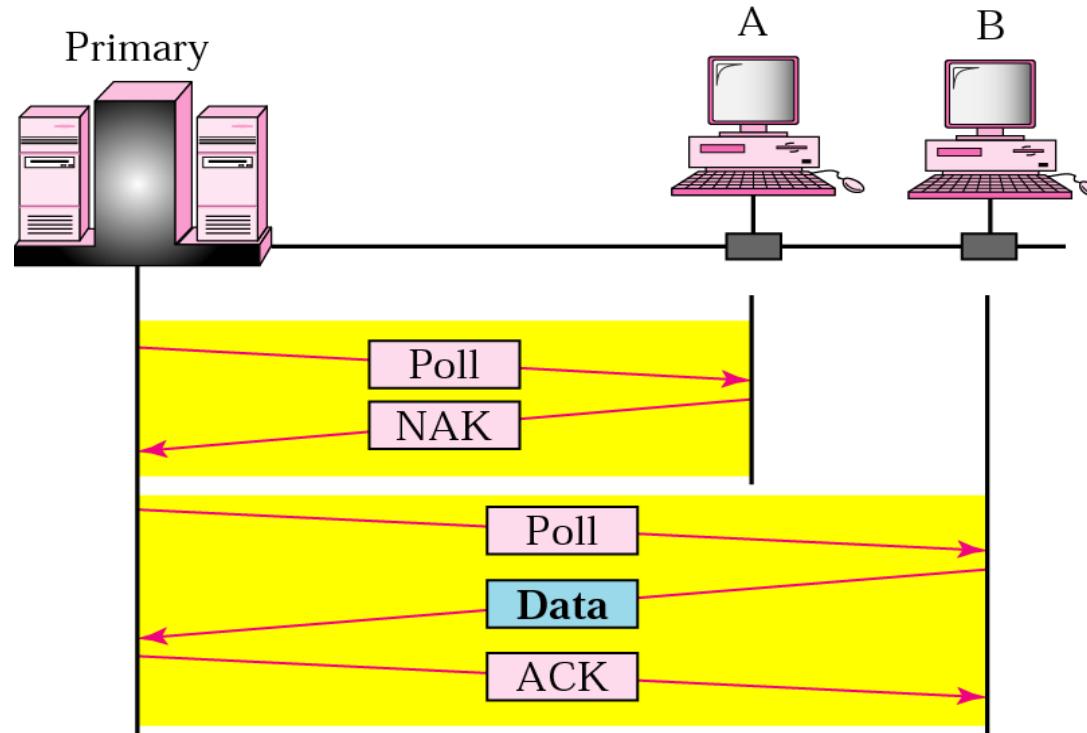
Select / Push



- Primary co-ordinates all communication
- Primary selects station that is destination then transmits data

* Figure is courtesy of B. Forouzan

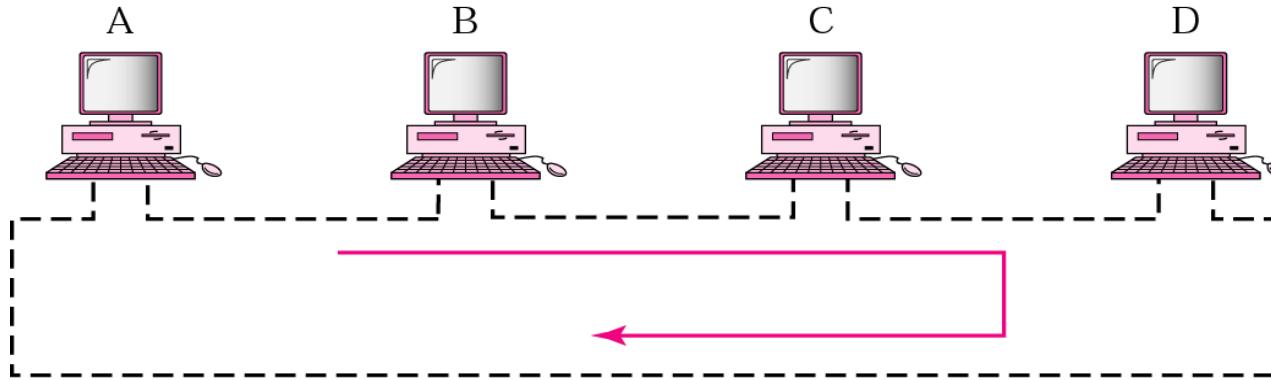
Poll



- Primary contacts stations to determine if they have data to transmit

* Figure is courtesy of B. Forouzan

Token-Passing Network



- Token passes around a network
- Machine with token is allowed to transmit data

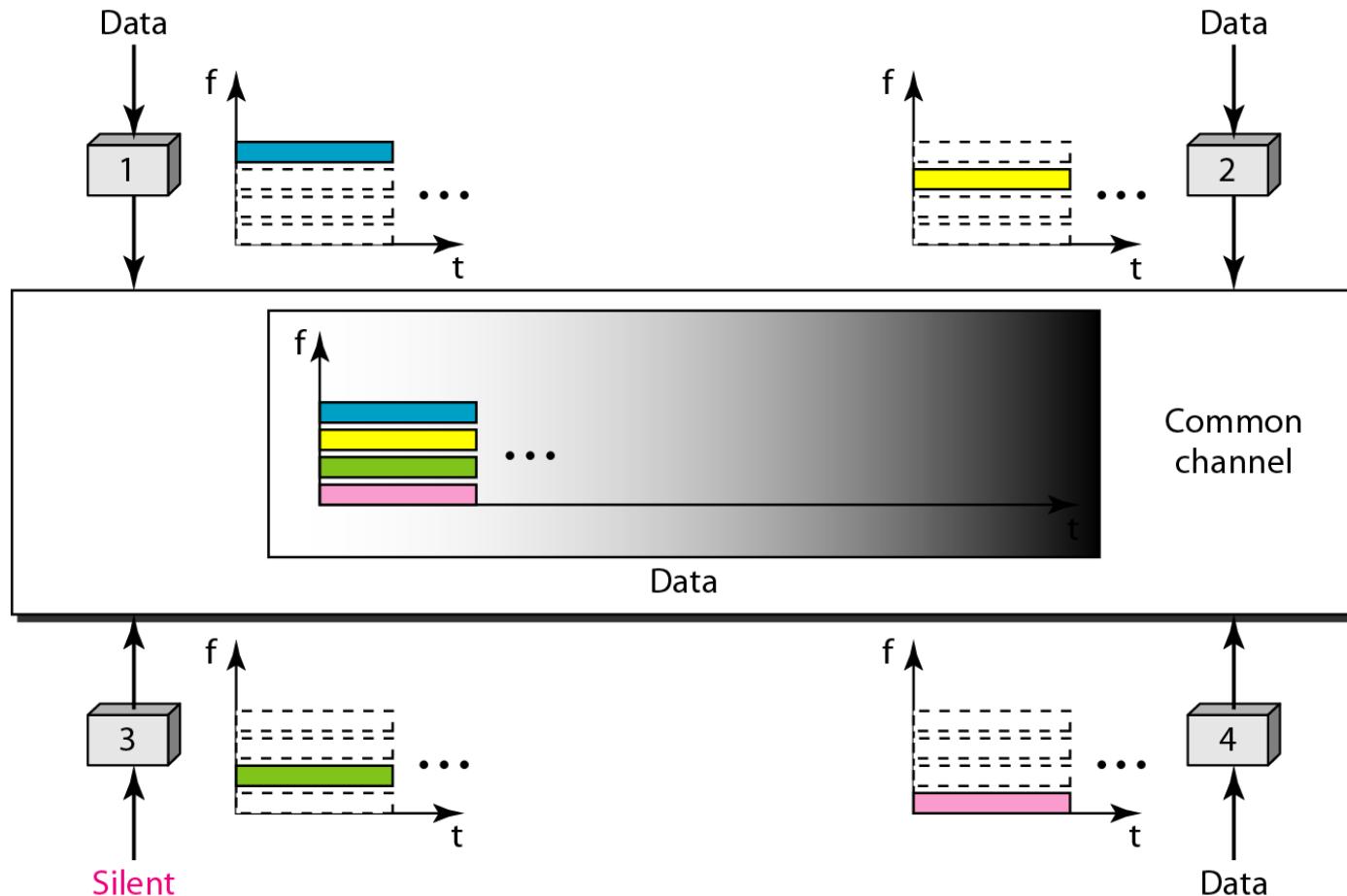
* Figure is courtesy of B. Forouzan



Static Channel Allocation

- Frequency Division Multiplexing (FDM)
 - N users get $1/N$ of the total bandwidth
 - $\ll N$ users \Rightarrow wasted bandwidth
 - $> N$ users \Rightarrow denial of service
 - Bursts cannot be accommodated
- Time Division Multiplexing (TDM)
 - N users get full bandwidth $1/N$ of the time
 - Same arguments apply

Frequency Division Multiple Access (FDMA)



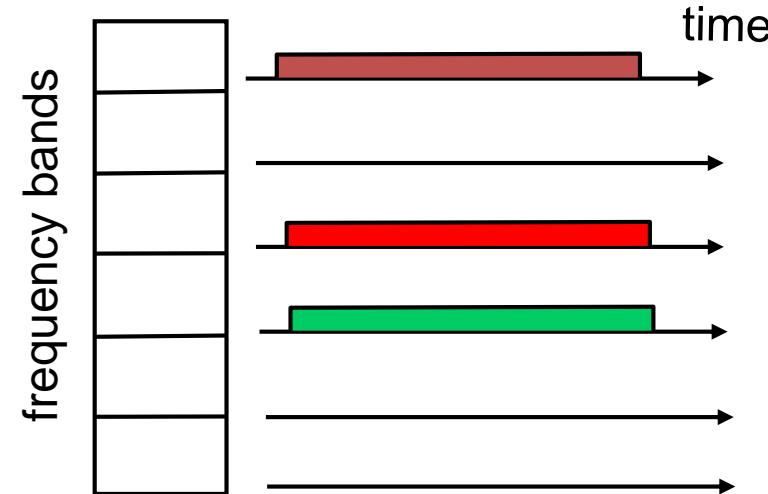
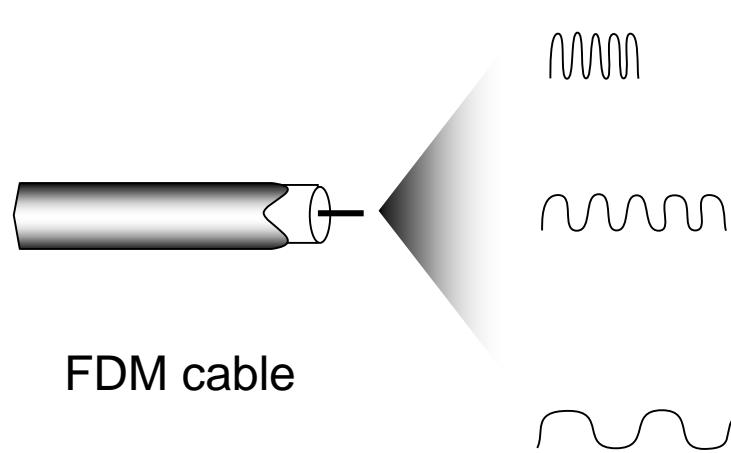
* Figure is courtesy of B. Forouzan



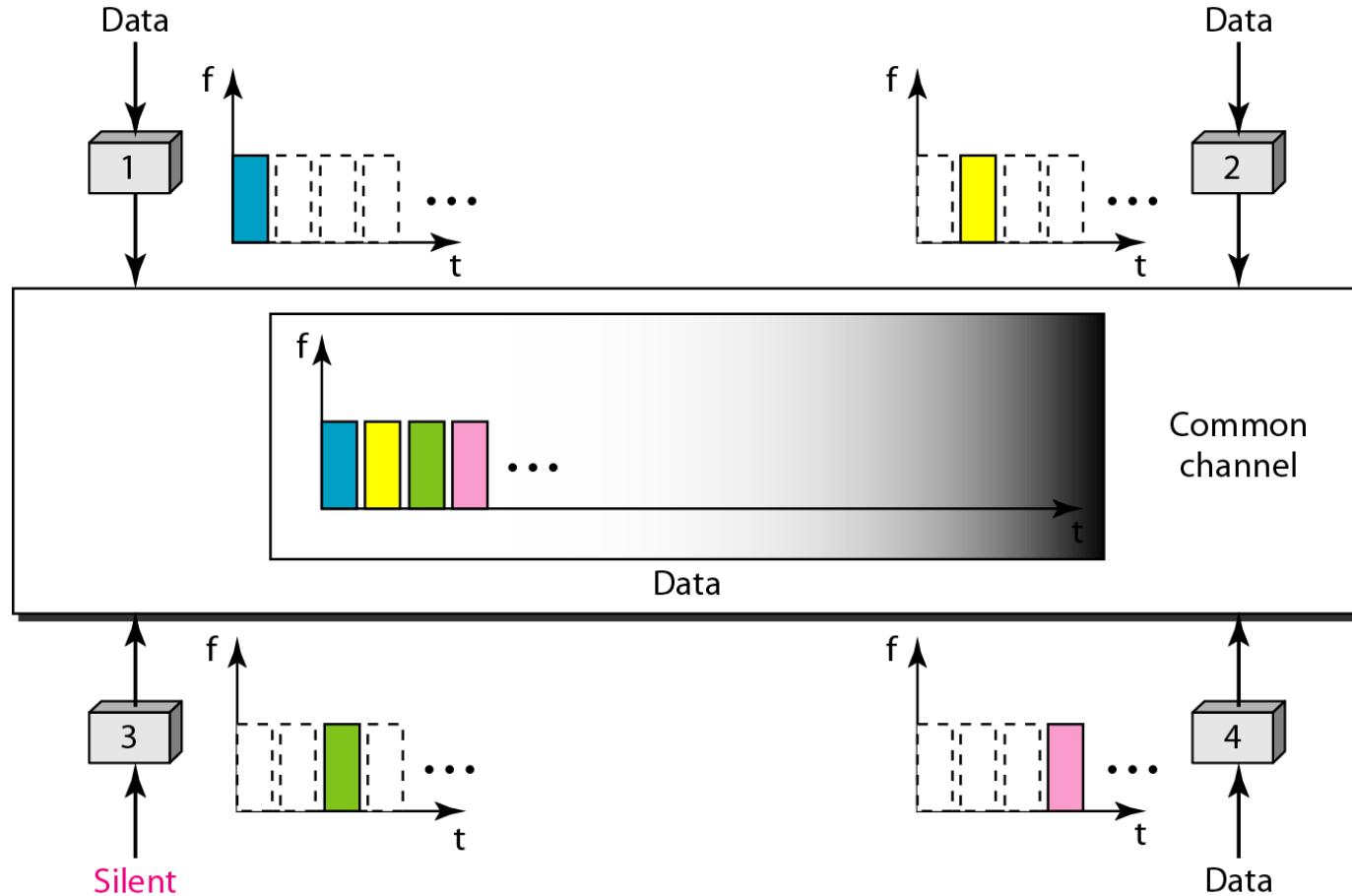
Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Time Division Multiple Access (TDMA)



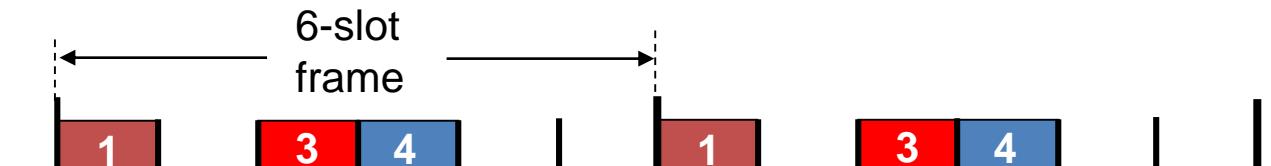
* Figure is courtesy of B. Forouzan



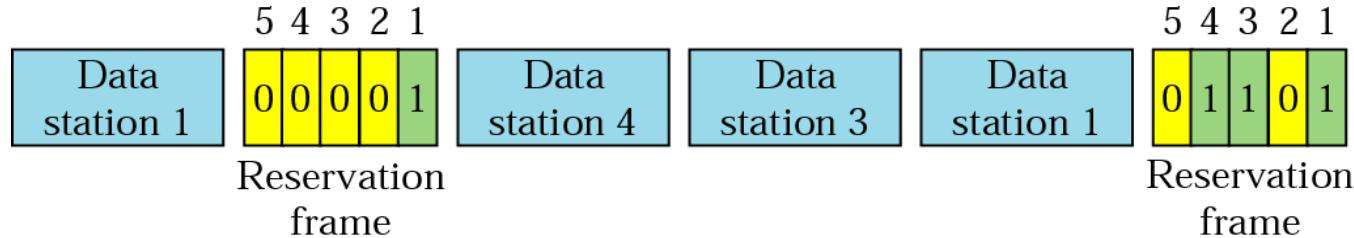
Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

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



Reservation Access Method

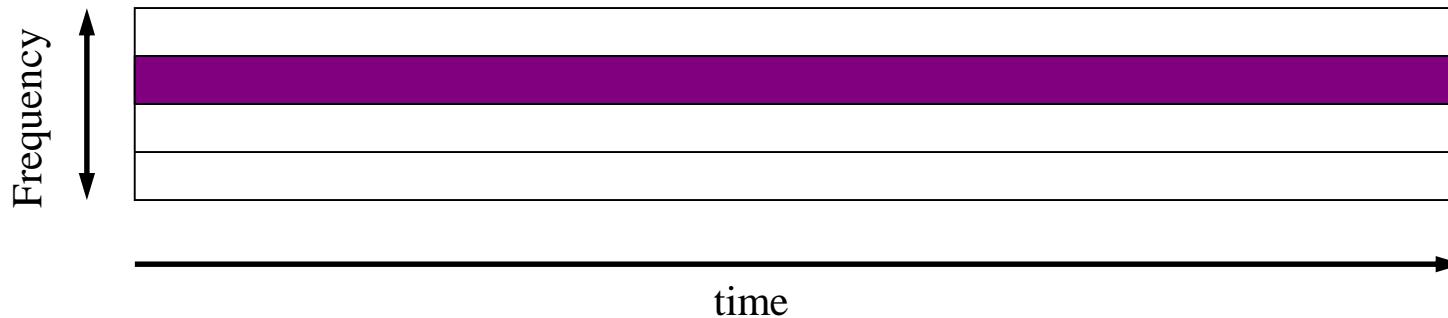


- Station that wants to transmit data
 - transmits 1 during its slot in the reservation frame
- All stations are informed about all planned communication
- Limited number of pre-allocated slots/stations

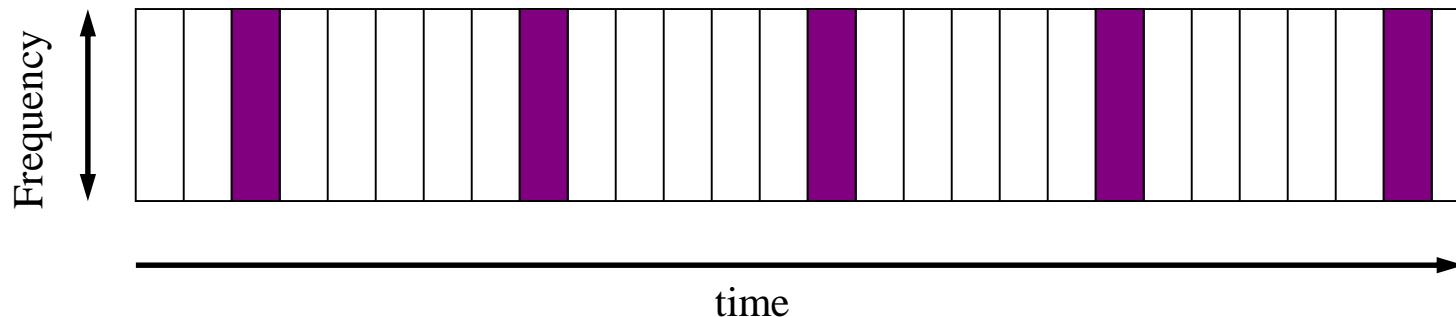
* Figure is courtesy of B. Forouzan

Static Channel Allocation

- Frequency Division Multiplexing (FDM)

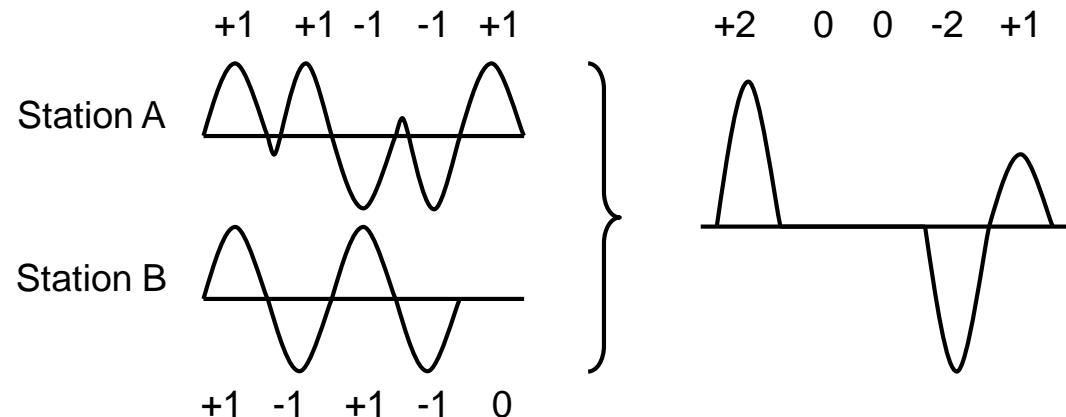
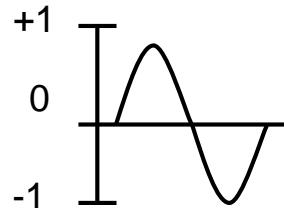


- Time Division Multiplexing (TDM)



Code Division Multiple Access (CDMA)

- Makes use of physical properties of interference
 - If two stations send signals in phase, they will "add up" to give twice the amplitude
 - If the signals are out of phase, they will "subtract" and give a signal that is the difference
- Difficult to implement because control of exact power strength is essential



Chip Sequences

- Every station is identified by an individual chip sequence

+1, +1, +1, +1

A

+1, -1, +1, -1

B

+1, +1, -1, -1

C

+1, -1, -1, +1

D

- Data bits are encoded as either +1, 0, or -1:

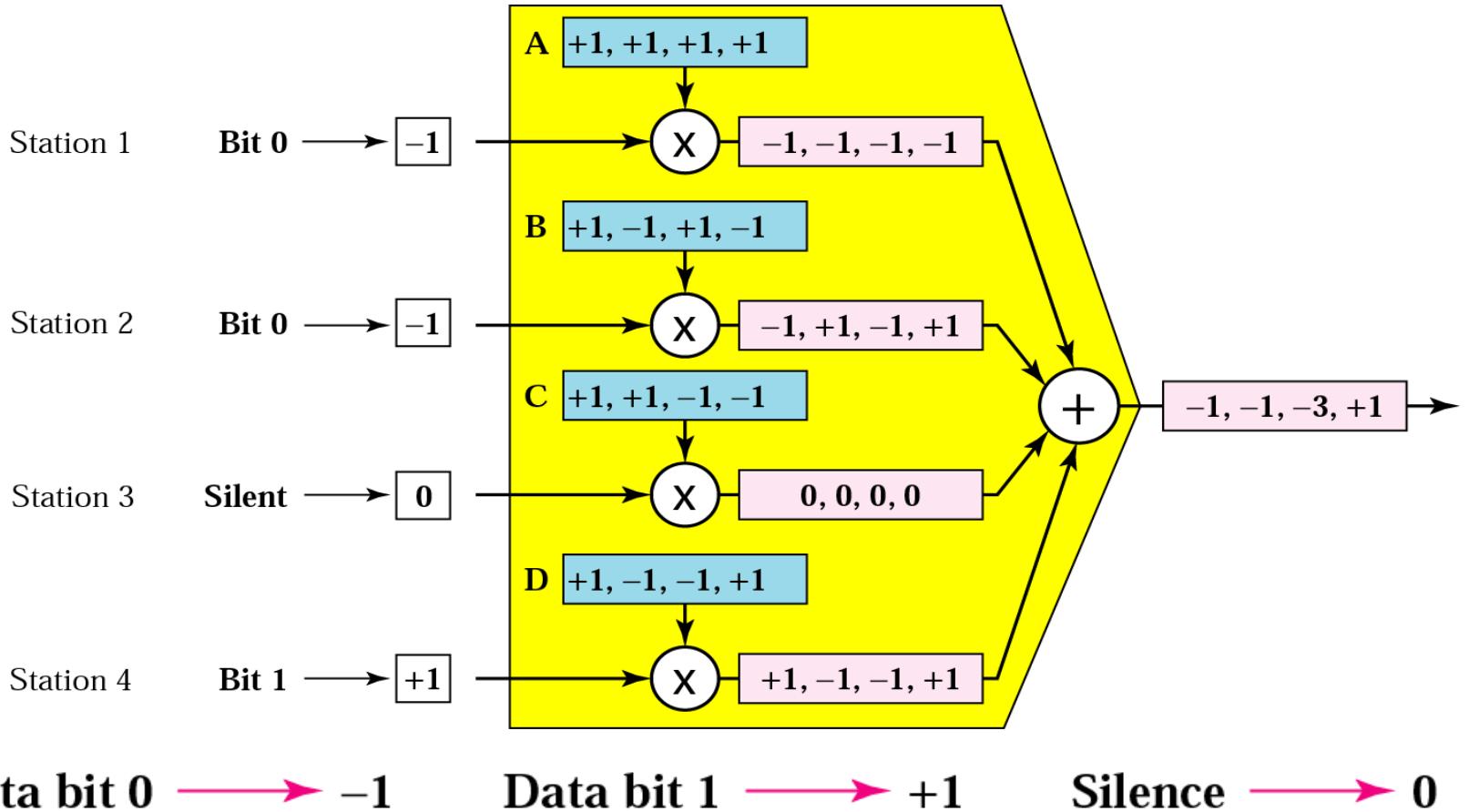
Data bit 0 \longrightarrow -1 Data bit 1 \longrightarrow +1 Silence \longrightarrow 0

Databit \otimes Chip Sequence = Transmission

$$-1 \otimes [+1, +1, -1, -1] \rightarrow [-1, -1, +1, +1]$$

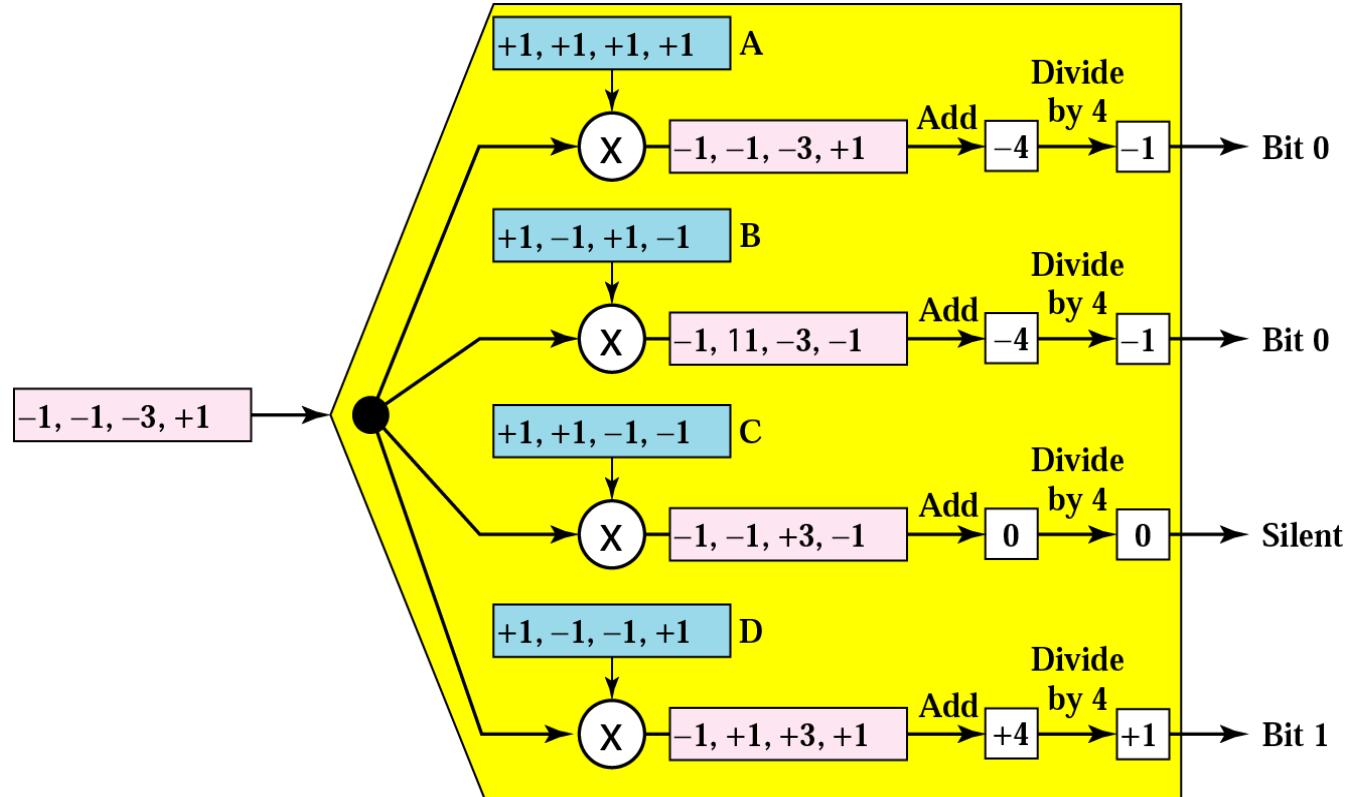
* Figure is courtesy of B. Forouzan

CDMA Multiplexer



* Figure is courtesy of B. Forouzan

CDMA De-Multiplexer



Decoding of received signal

* Figure is courtesy of B. Forouzan

Walsh Tables

$$W_1 = \begin{bmatrix} +1 \end{bmatrix}$$

$$W_{2N} =$$

$$\begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix}$$

$$W_1 = \begin{bmatrix} +1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$$

$$W_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

* Figure is courtesy of B. Forouzan



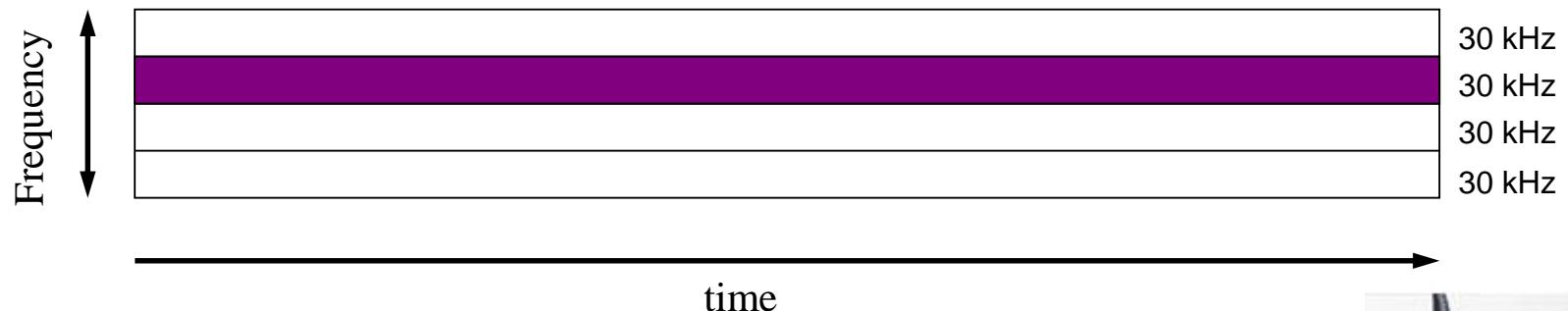
Summary: Synchronous CDMA

- Makes use of physical properties of interference
- All stations use whole bandwidth
- Computational requirements at stations
- Stations hold individual chip sequences

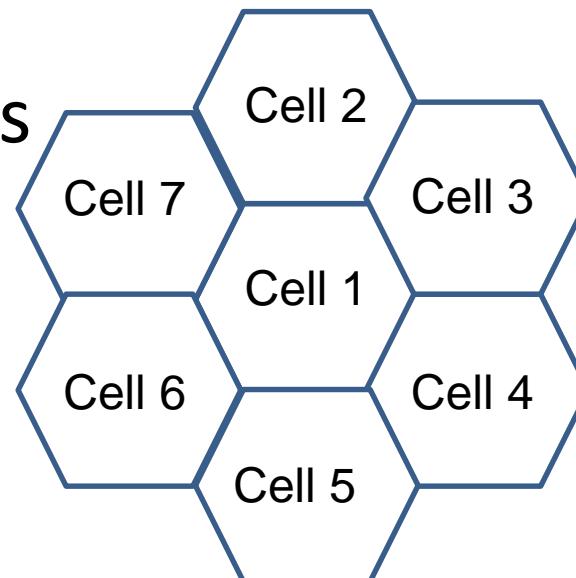


FDMA in AMPS

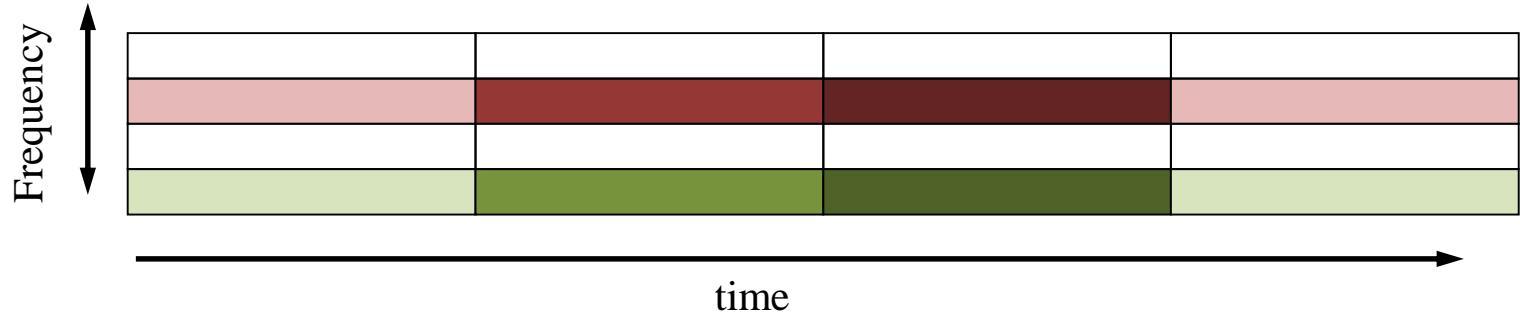
- FDMA



- Non-overlapping channels



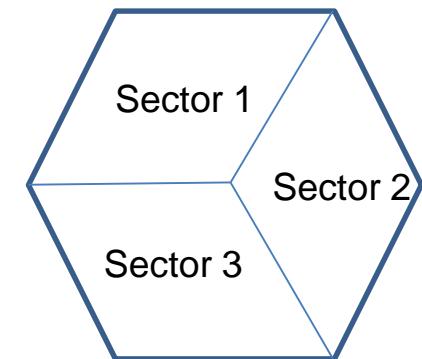
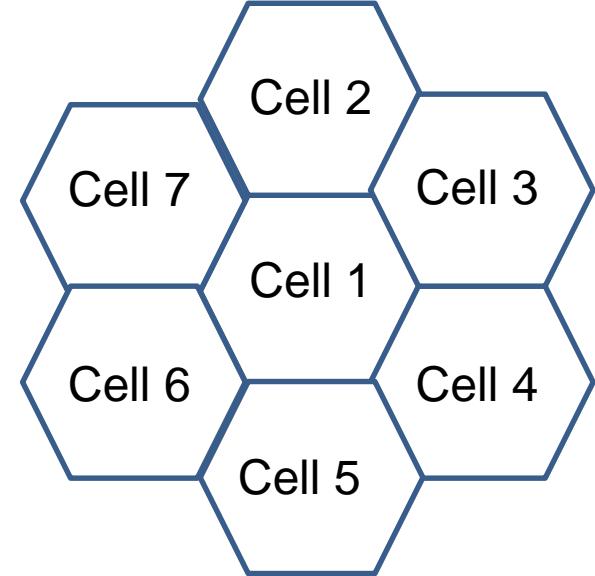
TDMA in GSM



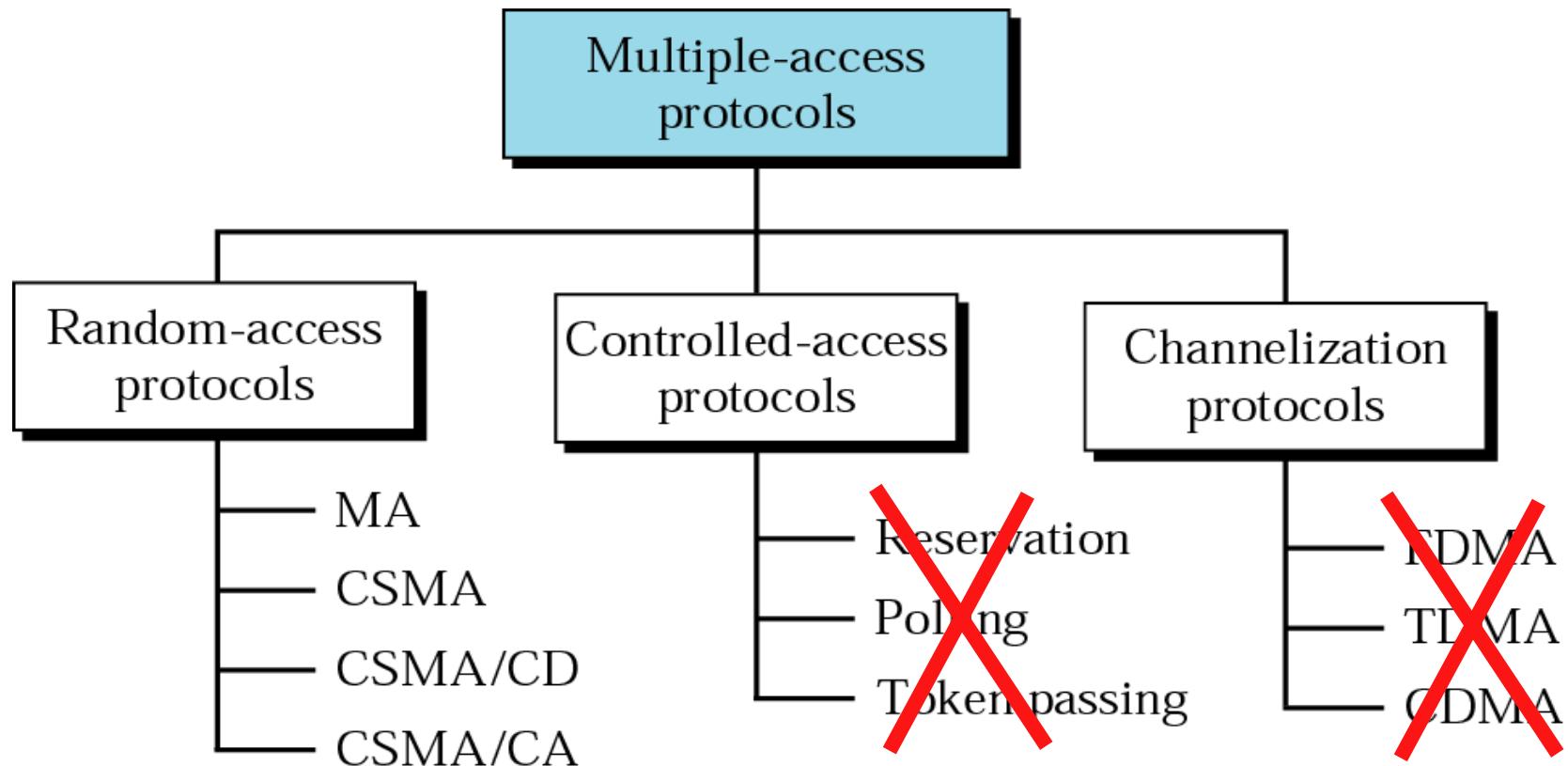
- Mixture of TDMA and FDMA

CDMA in Mobile Phones

- Asynchronous CDMA
- Cells use same frequencies
- Directional antennae used to split cell into sectors
 - 3x capacity



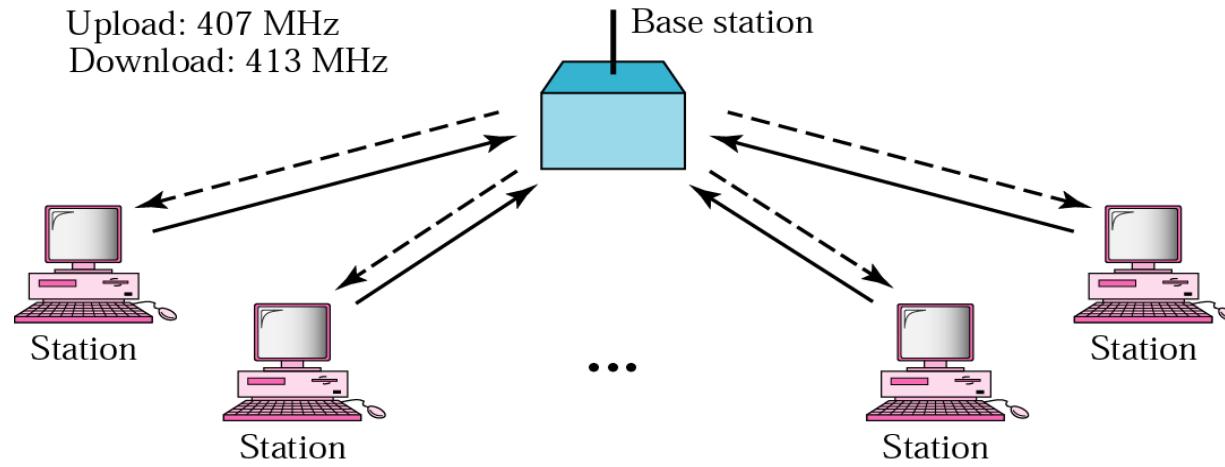
Multiple-Access Protocols



* Figure is courtesy of B. Forouzan



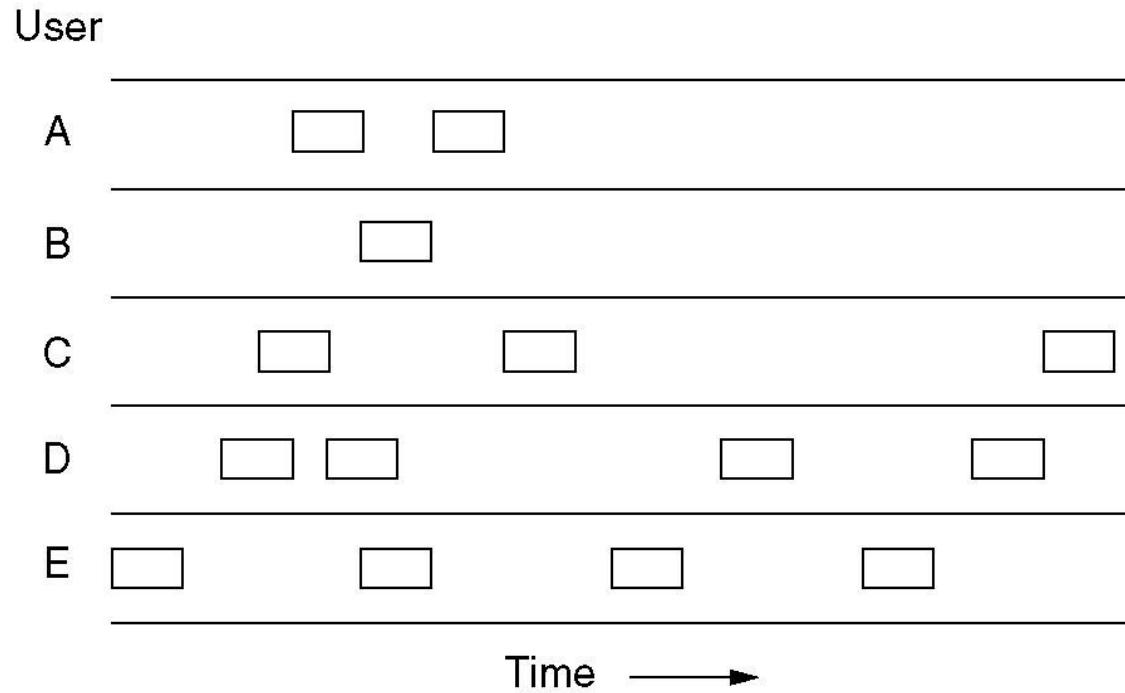
ALOHA Network



* Figure is courtesy of B. Forouzan



Pure ALOHA

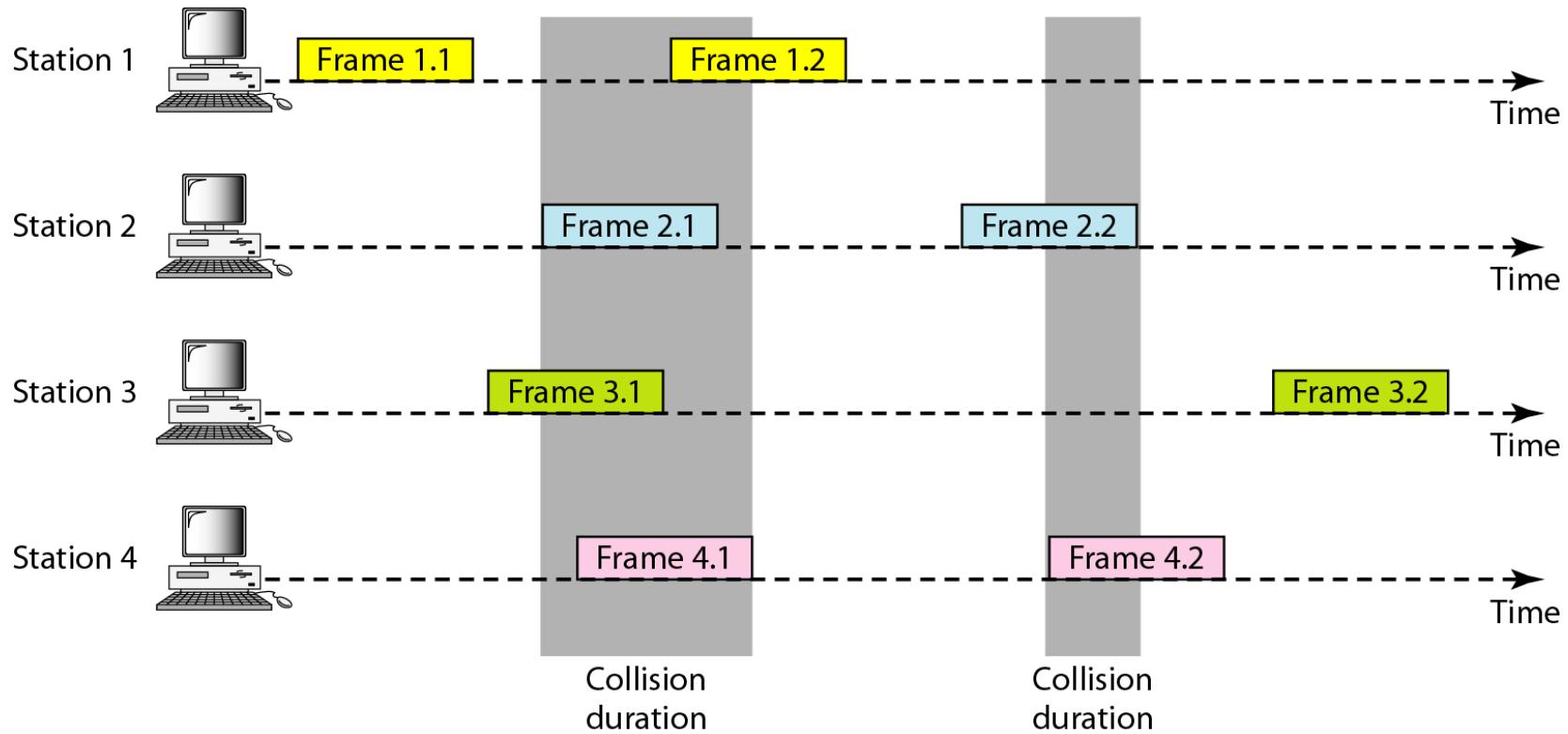


- Assuming all frames of equal length
- Frames are transmitted at completely arbitrary times

* Figure is courtesy of A. Tannenbaum



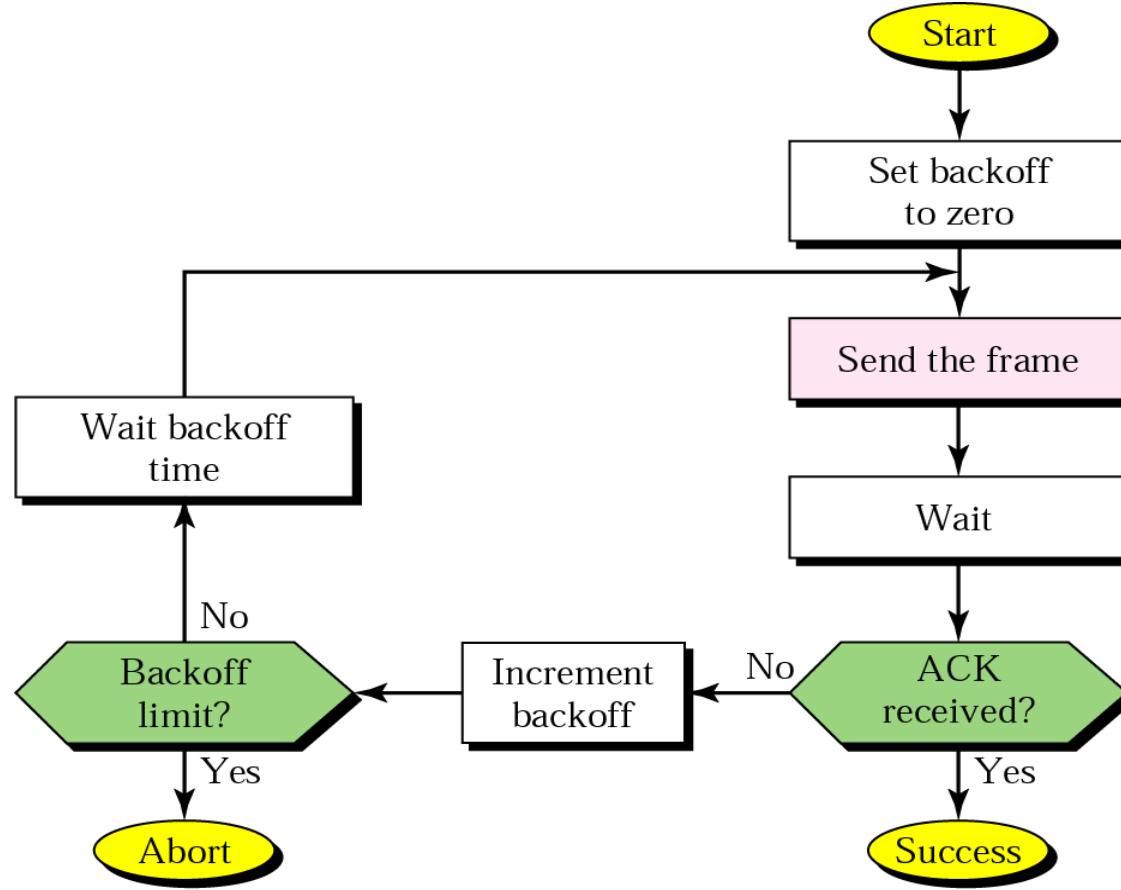
Pure Aloha II



- Collision occurs when frames are transmitted by stations at the same time

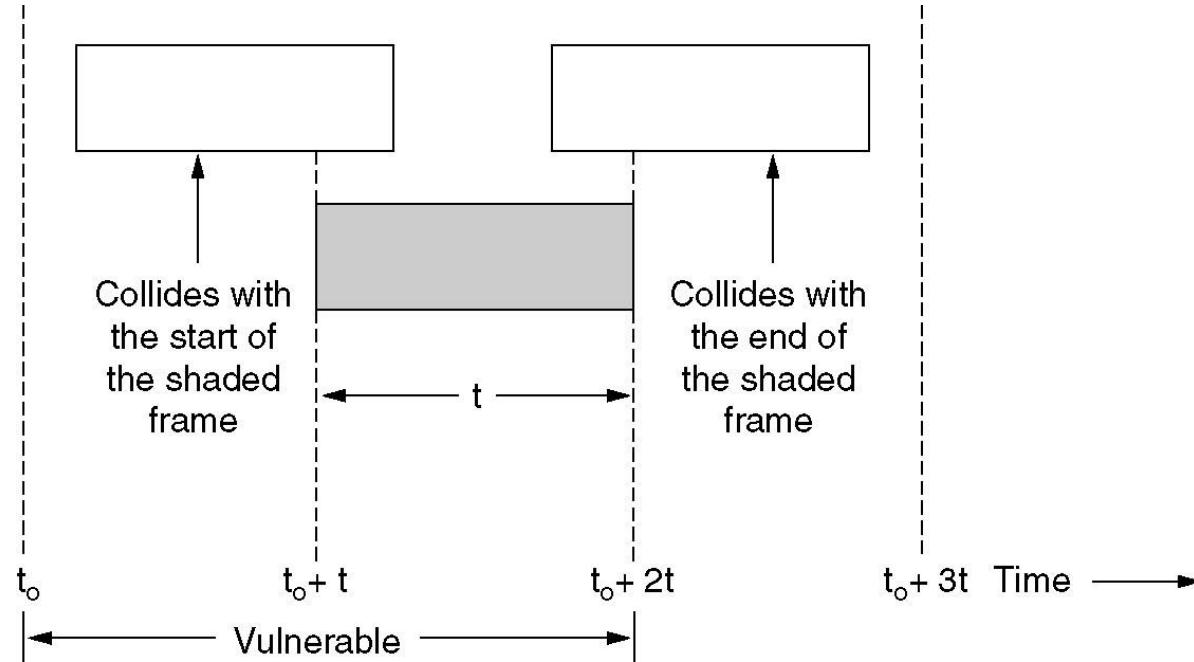
* Figure is courtesy of B. Forouzan

Procedure for ALOHA Protocol



* Figure is courtesy of B. Forouzan

Vulnerable Period for Frame

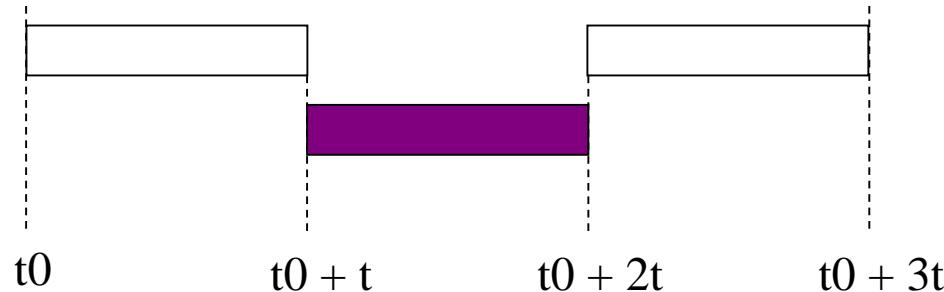


Maximum utilization around 18%

* Figure is courtesy of A. Tannenbaum

Slotted Aloha

Divide time into intervals (timeslots)



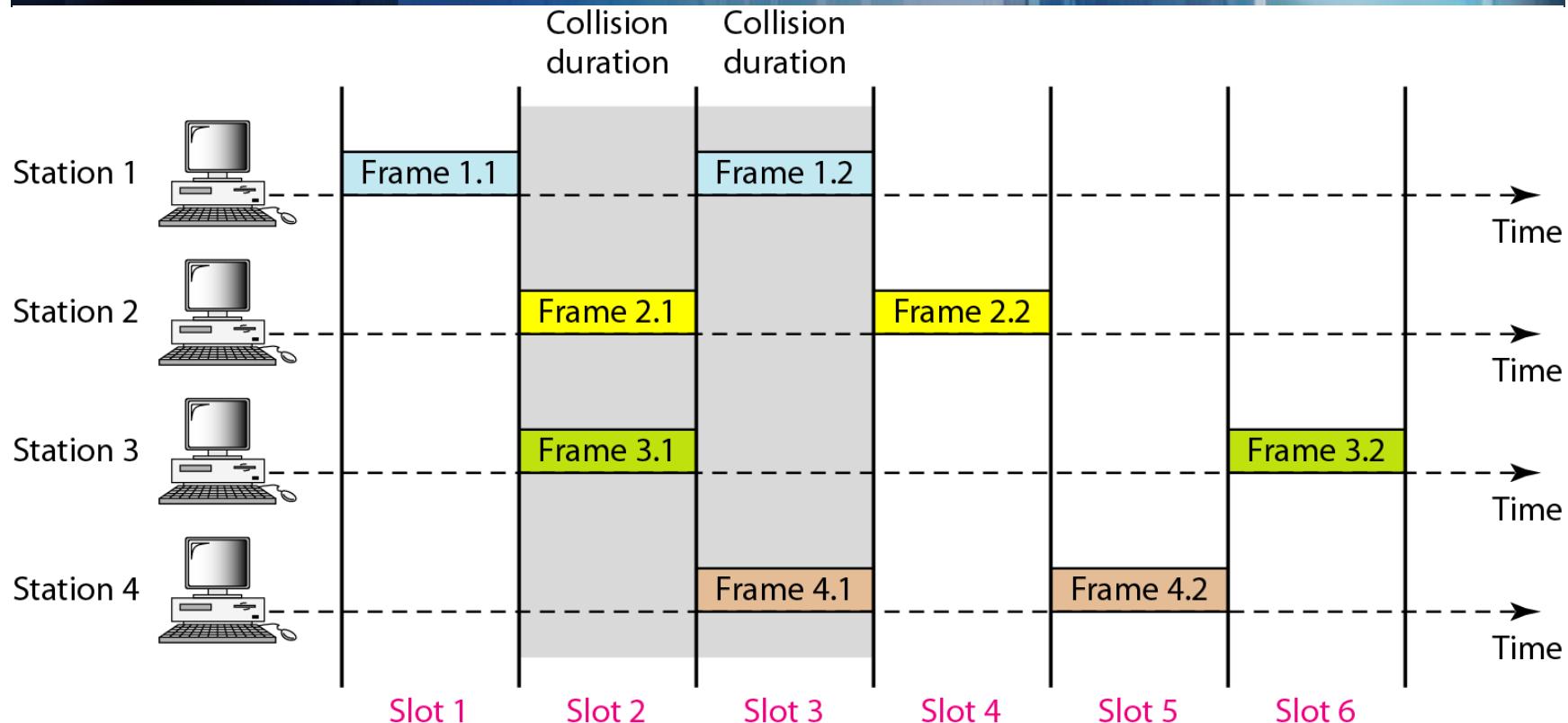
Algorithm:

```
when data ready wait for next timeslot  
transmit  
if (collision)  
    wait and retransmit
```

Maximum utilization
is $2 \times$ Pure Aloha

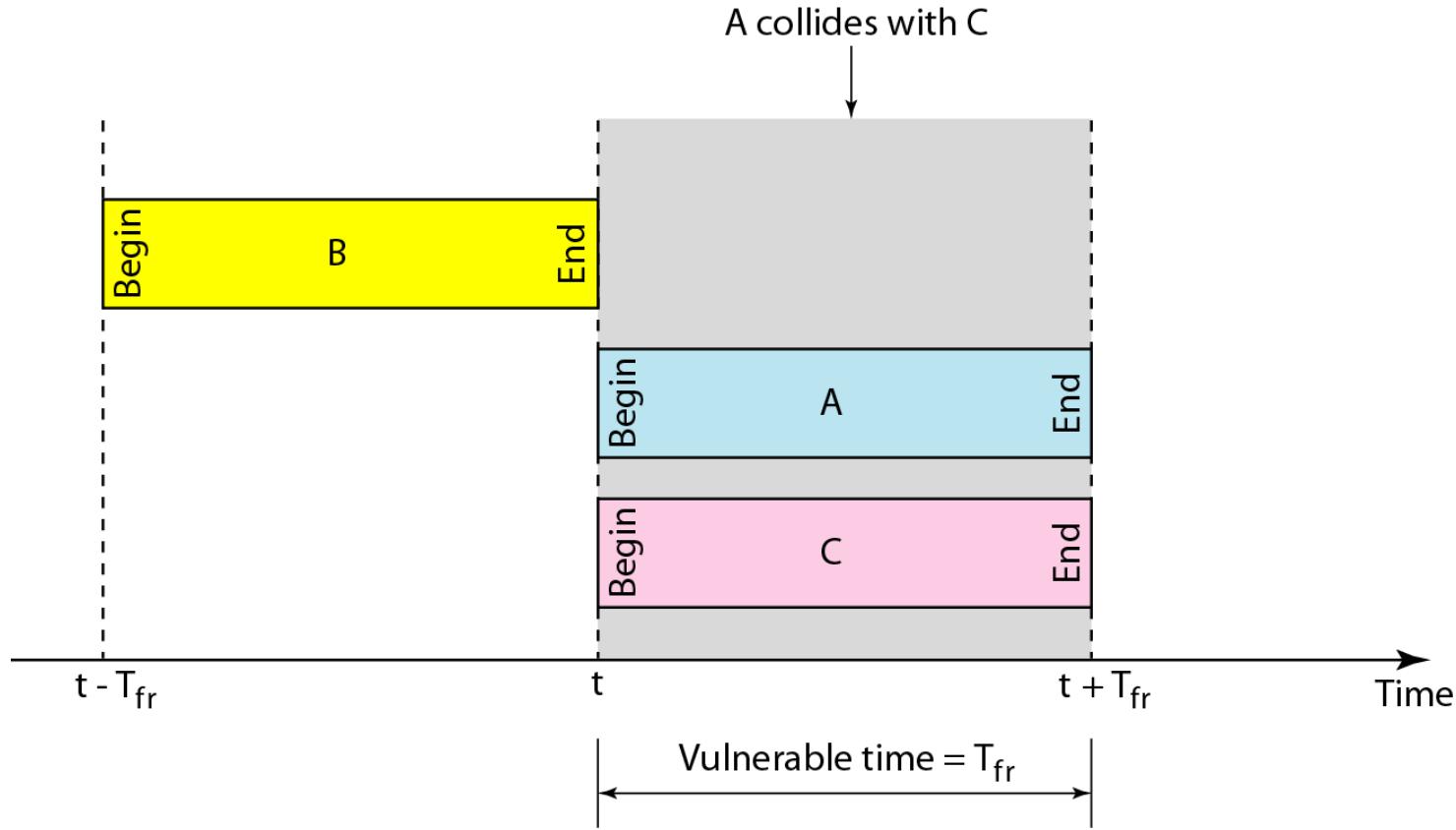


Frames in Slotted Aloha



* Figure is courtesy of B. Forouzan

Advantage of Slotted Aloha



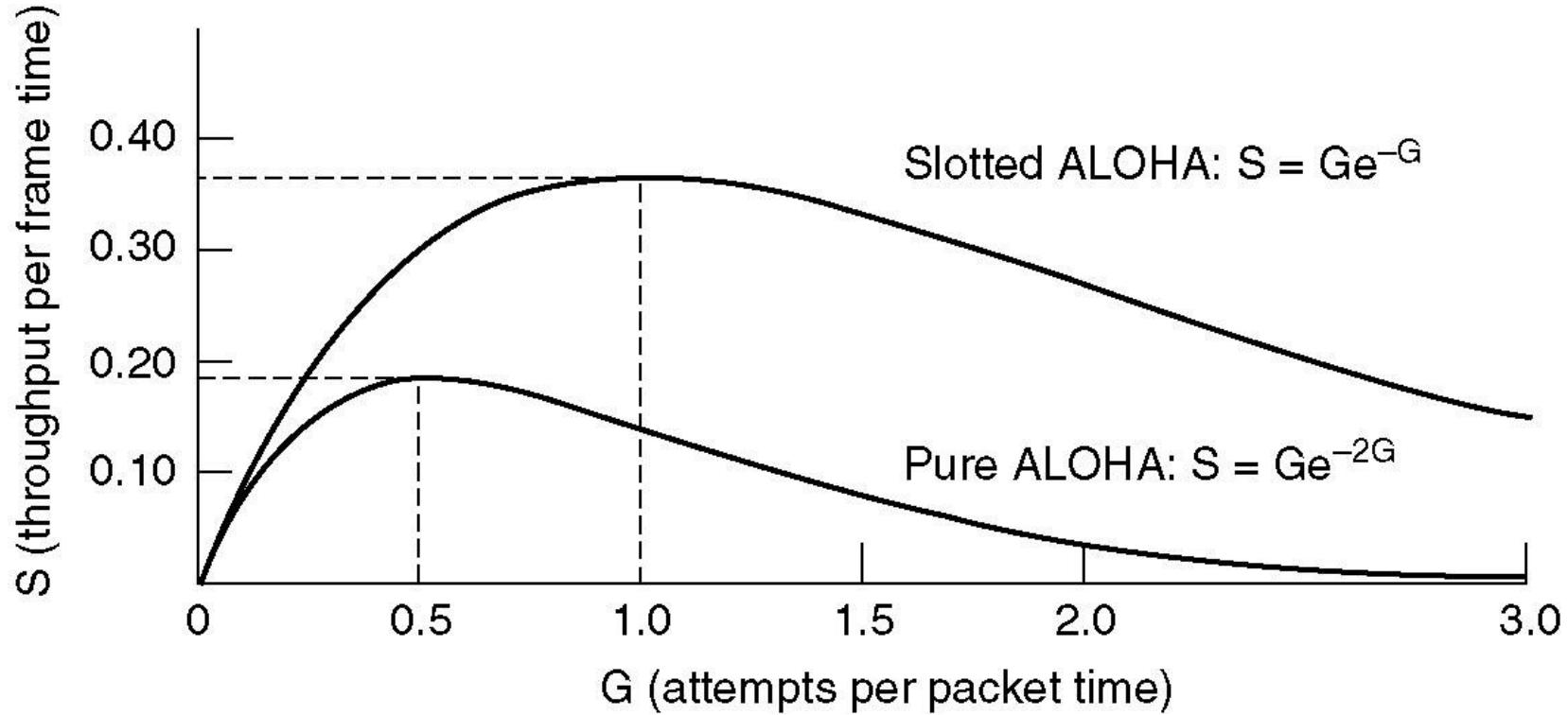
- Collisions can only occur within a slot

* Figure is courtesy of B. Forouzan



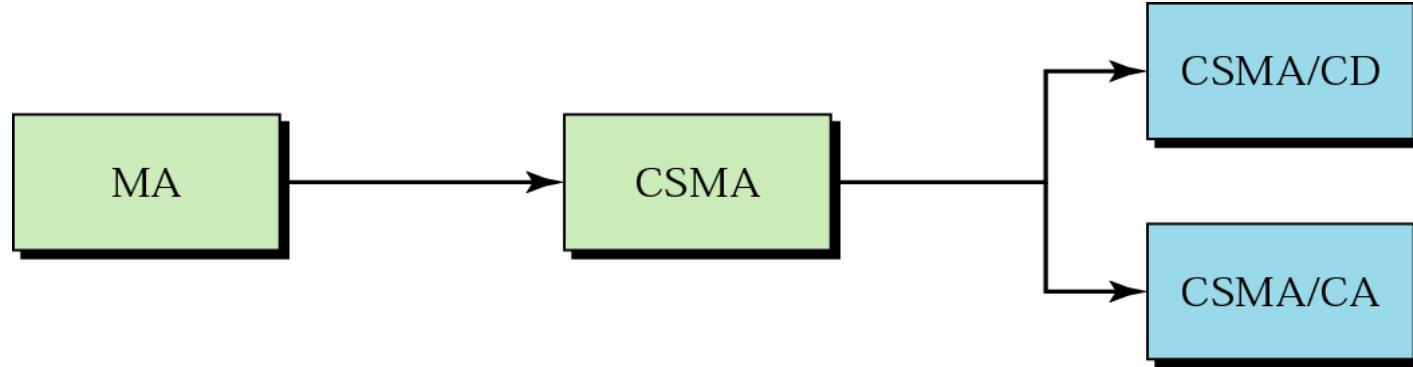
Performance of ALOHA

Throughput versus offered traffic for ALOHA systems.



* Figure is courtesy of A. Tannenbaum

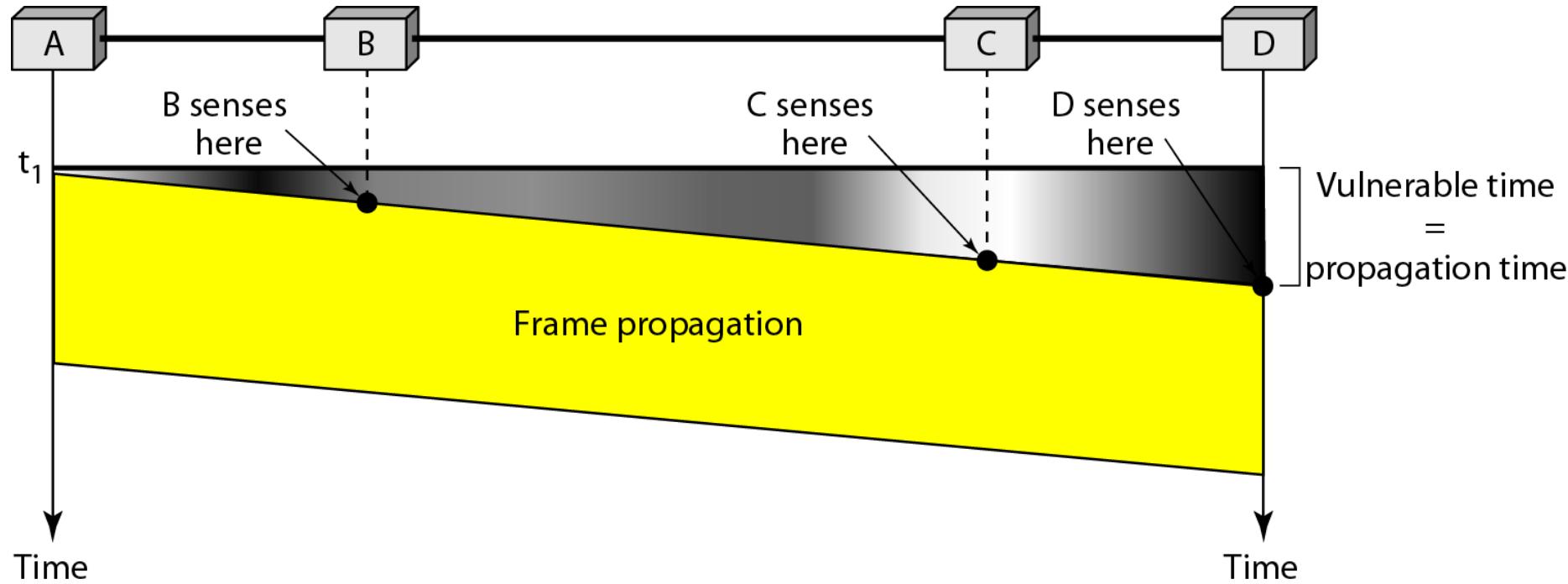
Random-Access Methods



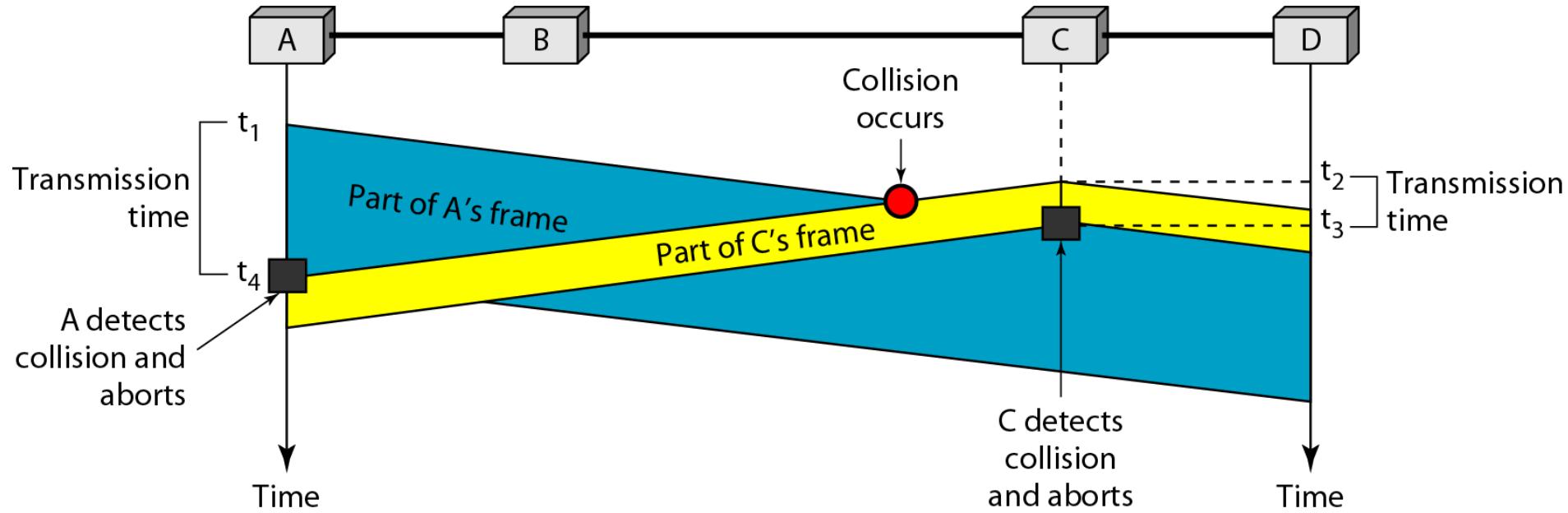
- CS \Rightarrow Carrier Sense
- MA \Rightarrow Multiple Access
- CD \Rightarrow Collision Detection
- CA \Rightarrow Collision Avoidance

* Figure is courtesy of B. Forouzan

CSMA



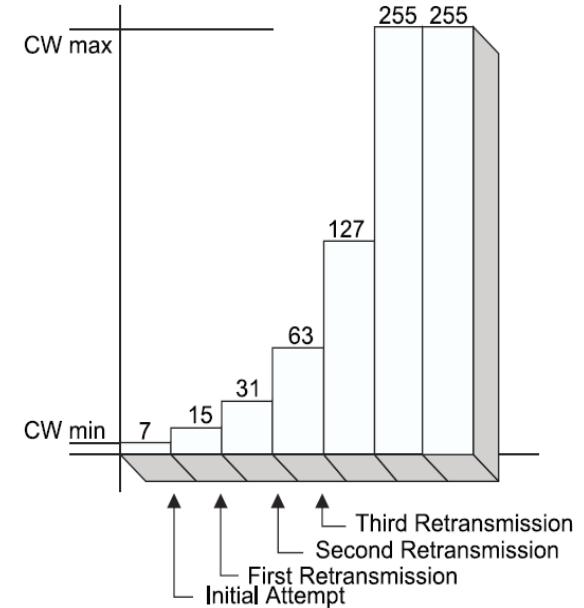
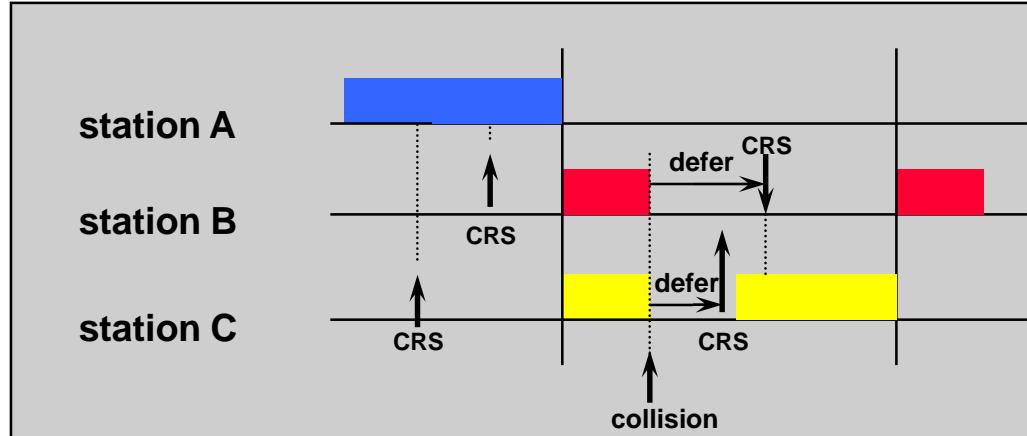
Collision in CSMA/CD



- Both stations will realize that a collision has taken place
- Backoff and attempt to send later

* Figure is courtesy of B. Forouzan

Binary Exponential Backoff



$$\text{Backoff Time} = \text{Random()} \times \text{aSlotTime}$$

where

Random() = Pseudorandom integer drawn from a uniform distribution over the interval $[0, \text{CW}]$, where CW is an integer within the range of values of the PHY characteristics aCWmin and aCWmax , $\text{aCWmin} \leq \text{CW} \leq \text{aCWmax}$. It is important that designers recognize the need for statistical independence among the random number streams among STAs.

aSlotTime = The value of the correspondingly named PHY characteristic.

* Figure is courtesy of ANSI/IEEE Std 802.11 & Avaya Communications Inc

An Alternative Representation

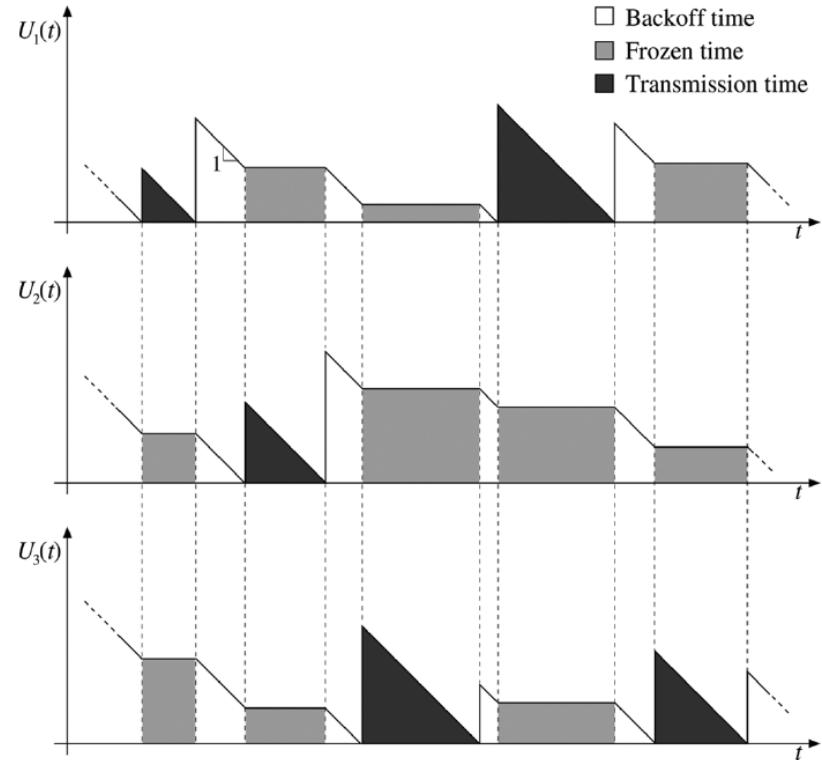
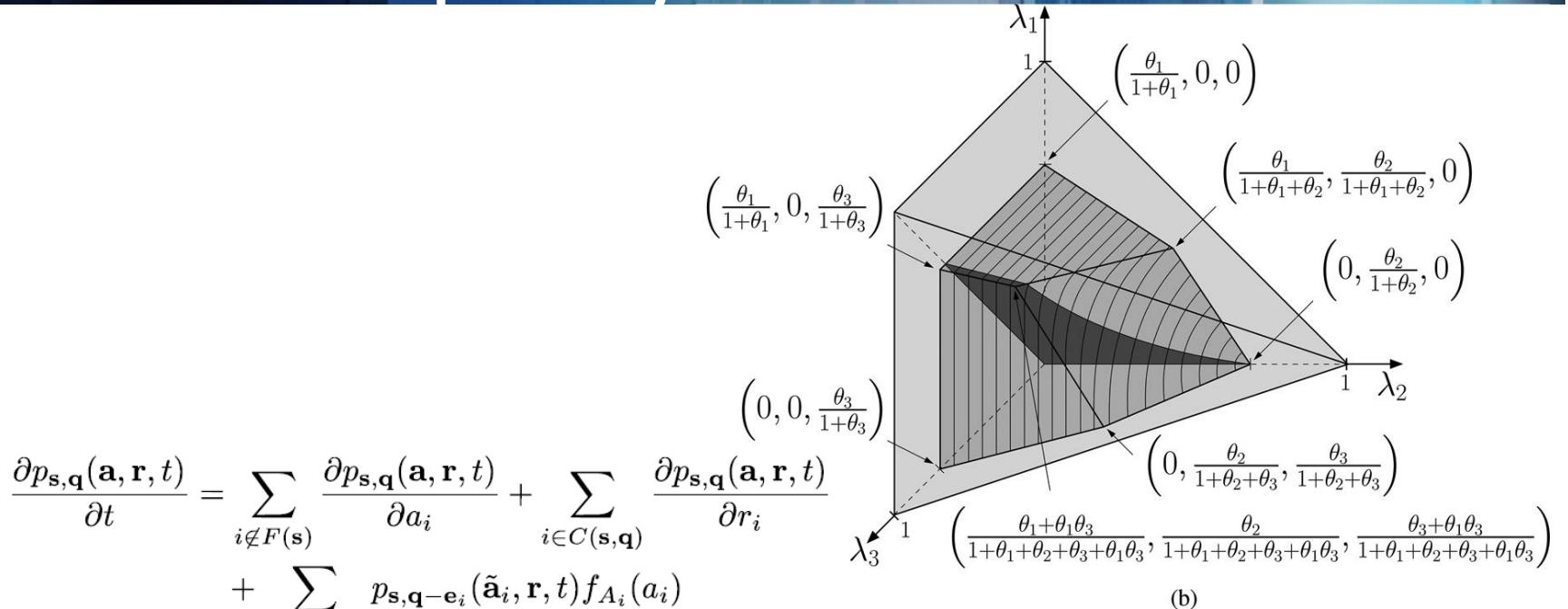


Fig. 1. The operation of three saturated links within carrier-sense range. The graphs show the unfinished work $U_i(t)$ of each transmitter τ_i at time t , which can be either the remaining backoff or the remaining transmission time.

Rafael Laufer and Leonard Kleinrock, The Capacity of Wireless CSMA/CA Networks,
 IEEE/ACM TRANSACTIONS ON NETWORKING, vol. 24, no. 3, pp 1518-1532, JUNE 2016

Capacity of Networks

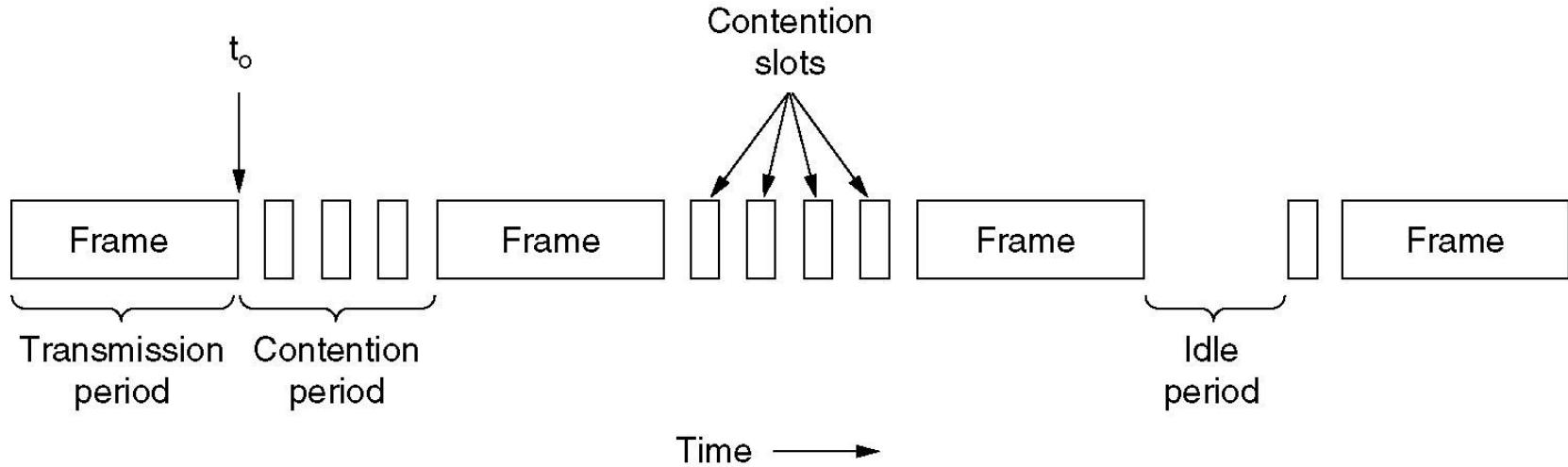


$$\begin{aligned}
 \frac{\partial p_{\mathbf{s}, \mathbf{q}}(\mathbf{a}, \mathbf{r}, t)}{\partial t} = & \sum_{i \notin F(\mathbf{s})} \frac{\partial p_{\mathbf{s}, \mathbf{q}}(\mathbf{a}, \mathbf{r}, t)}{\partial a_i} + \sum_{i \in C(\mathbf{s}, \mathbf{q})} \frac{\partial p_{\mathbf{s}, \mathbf{q}}(\mathbf{a}, \mathbf{r}, t)}{\partial r_i} \\
 & + \sum_{i \in A(\mathbf{s}, \mathbf{q})} p_{\mathbf{s}, \mathbf{q} - \mathbf{e}_i}(\tilde{\mathbf{a}}_i, \mathbf{r}, t) f_{A_i}(a_i) \\
 & + \sum_{i \in T(\mathbf{s})} p_{\mathbf{s} - \mathbf{e}_i, \mathbf{q}}(\mathbf{a}, \tilde{\mathbf{r}}_i, t) f_{T_i}(r_i) \\
 & + \sum_{i \in B(\mathbf{s})} p_{\mathbf{s} + \mathbf{e}_i, \mathbf{q} + \mathbf{e}_i}(\mathbf{a}, \tilde{\mathbf{r}}_i, t) f_{B_i}(r_i). \quad (31)
 \end{aligned}$$

(b)

Rafael Laufer and Leonard Kleinrock, The Capacity of Wireless CSMA/CA Networks, IEEE/ACM TRANSACTIONS ON NETWORKING, vol. 24, no. 3, pp 1518-1532, JUNE 2016

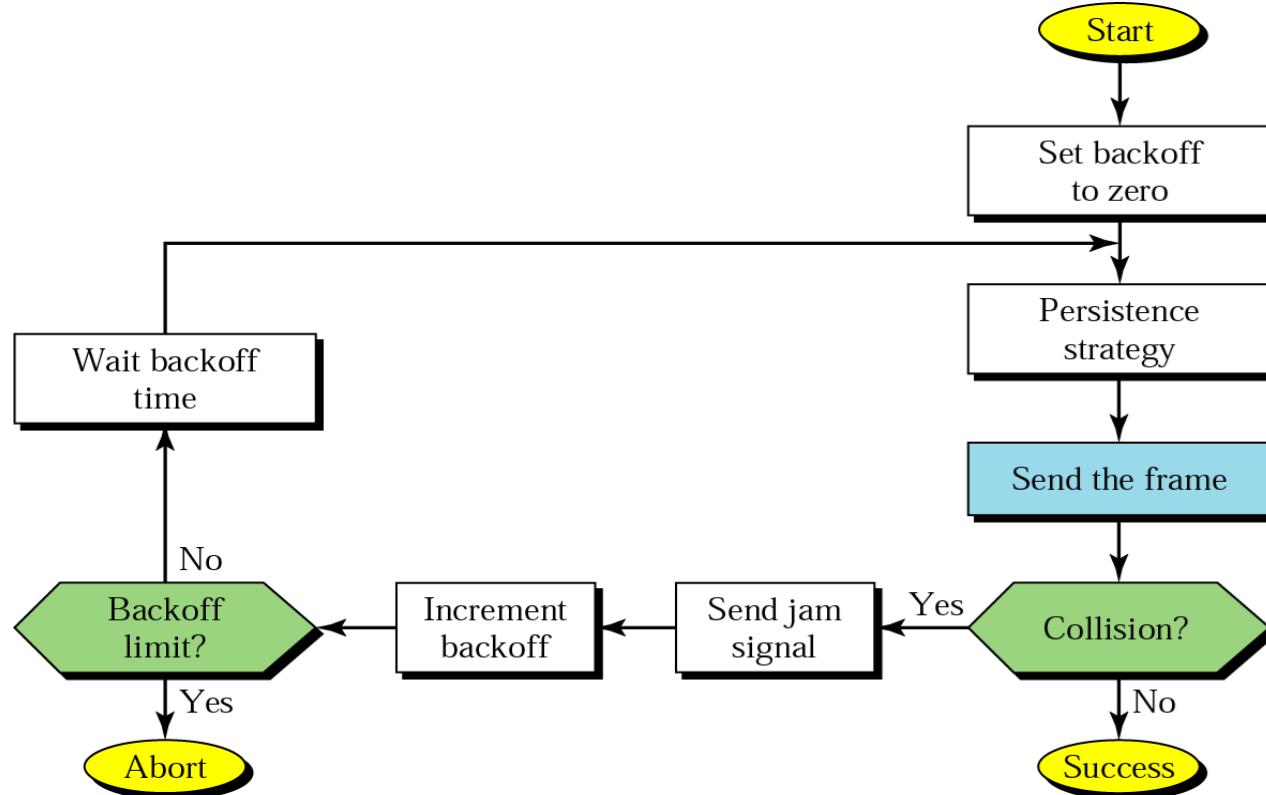
CSMA with Collision Detection



CSMA/CD can be in one of three states:
contention, transmission, or idle.

* Figure is courtesy of A. Tannenbaum

CSMA/CD Procedure

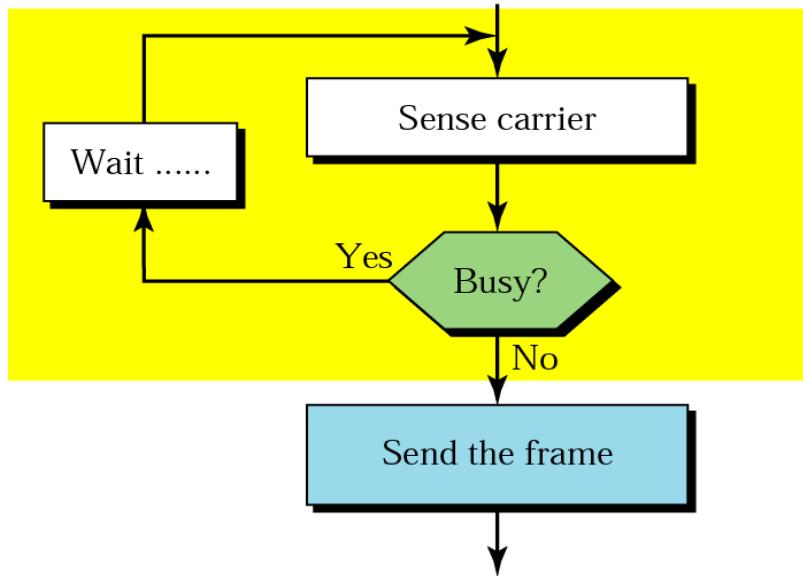


* Figure is courtesy of B. Forouzan

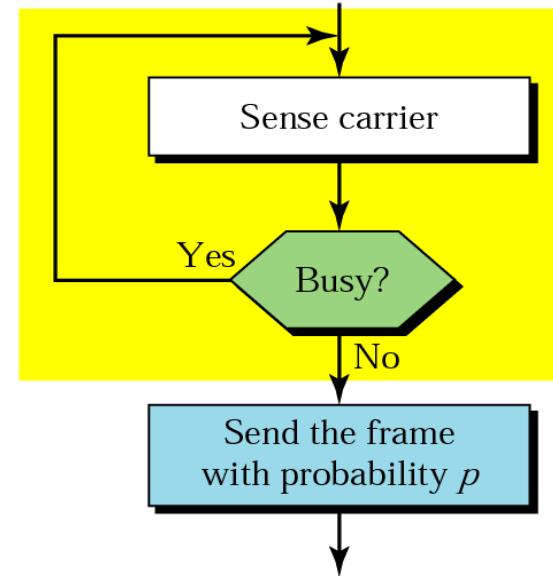


Persistence Strategies

Nonpersistent strategy



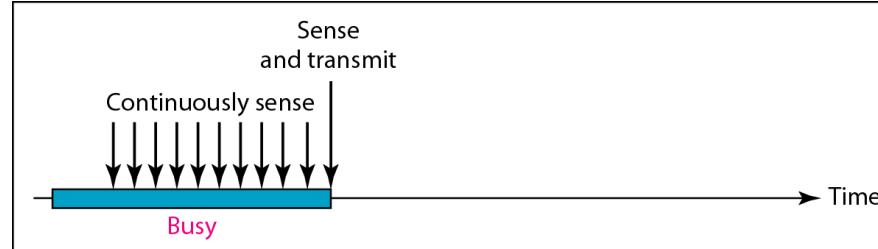
Persistent strategy



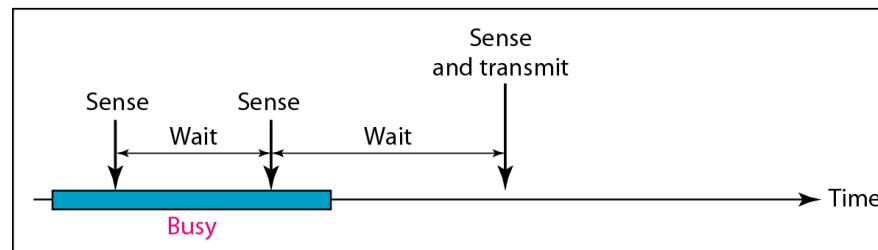
- **1-persistent CSMA**
 - if medium idle send immediately
- **p-persistent CSMA**
 - if medium available station may send depending on probability
 - reduces chance of collision and improves efficiency

* Figure is courtesy of B. Forouzan

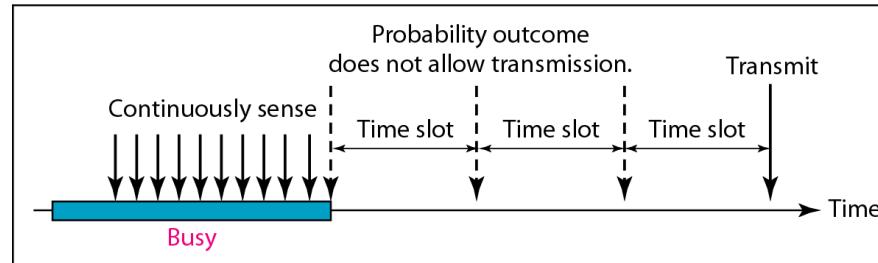
Persistence Strategies II



a. 1-persistent



b. Nonpersistent



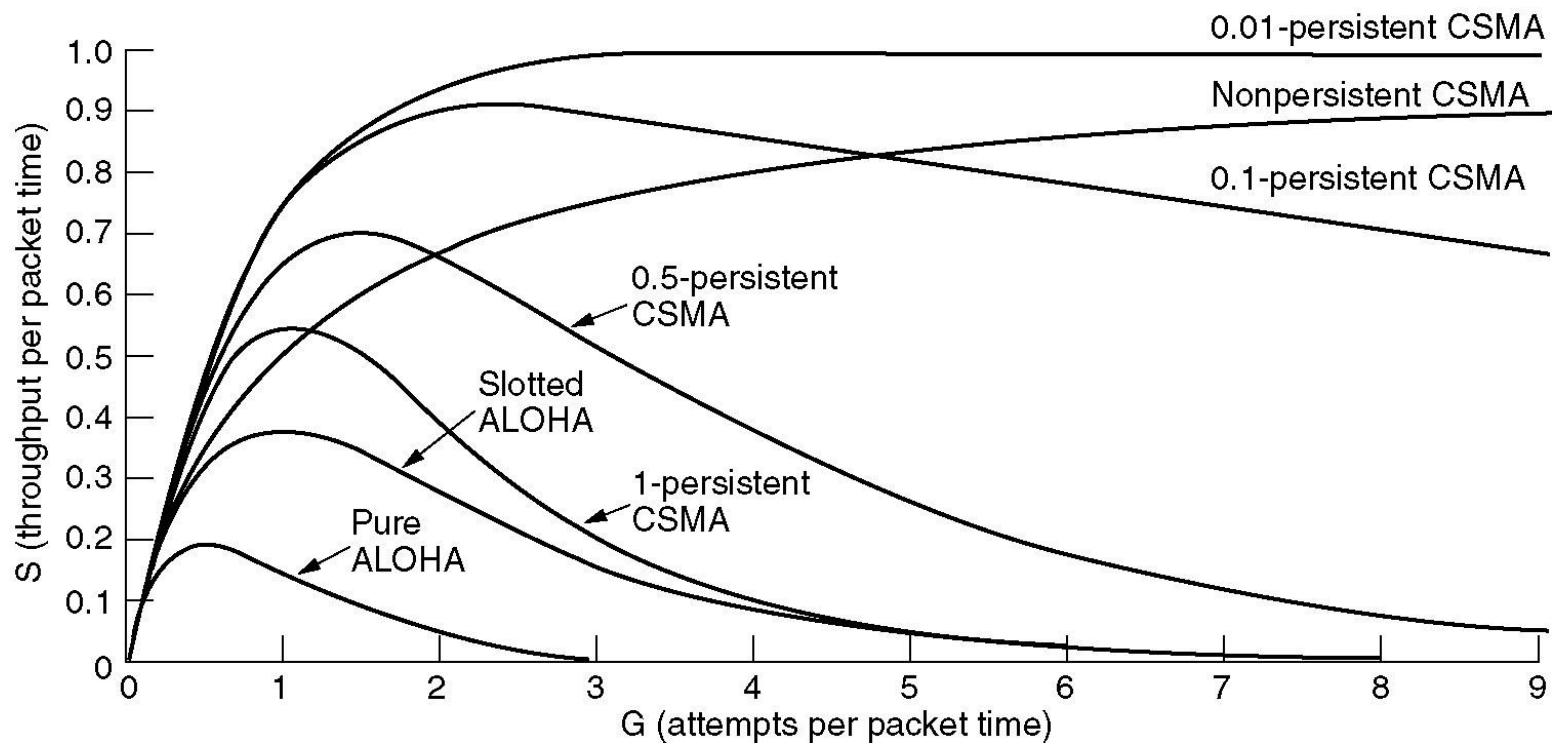
c. p-persistent

* Figure is courtesy of B. Forouzan



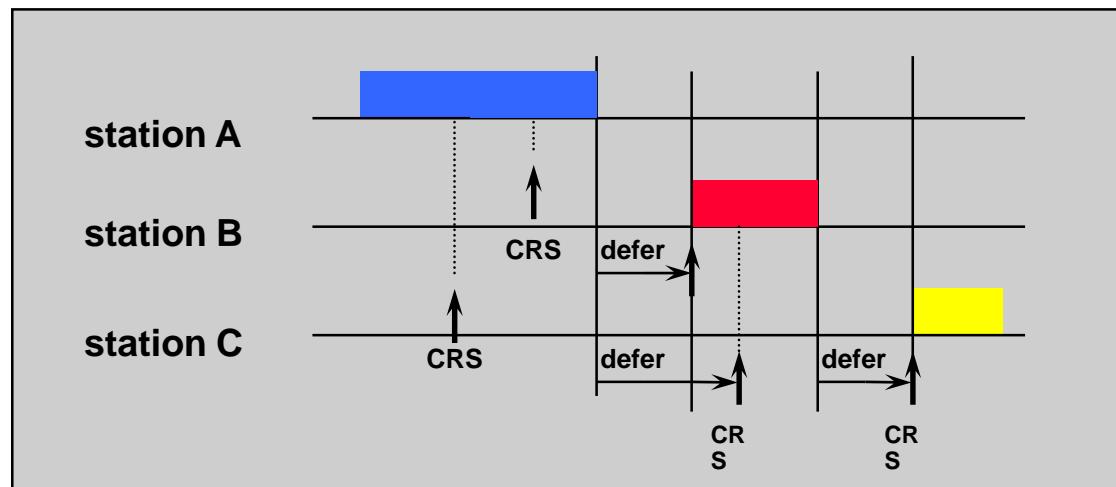
Persistent and Non-Persistent CSMA

Comparison of the channel utilization versus load for various random access protocols



* Figure is courtesy of A. Tanenbaum

CSMA in Wireless Media

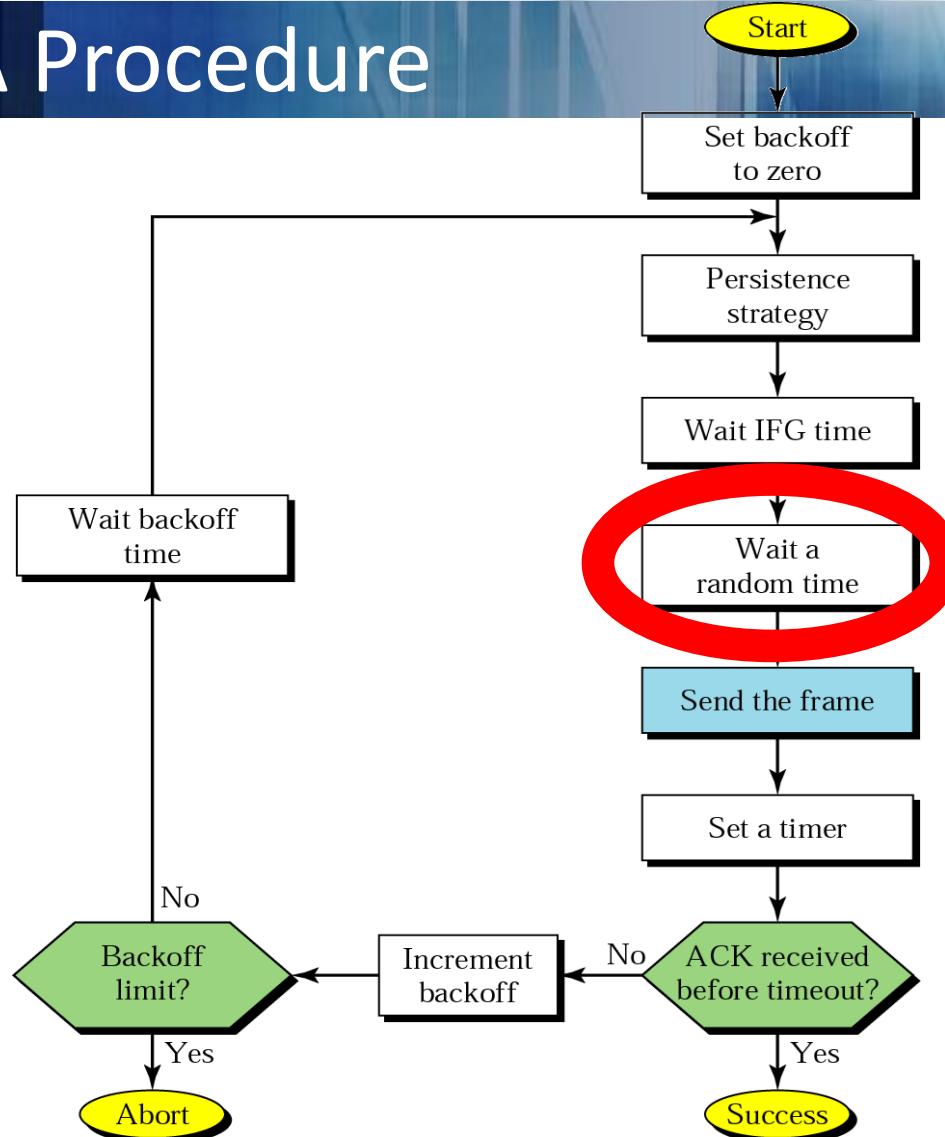


Collision is at the receiver !!!

- Sense carrier to determine if medium is free
- Once free pick a random number
 - then start sending

* Figure is courtesy of Avaya Communications Inc

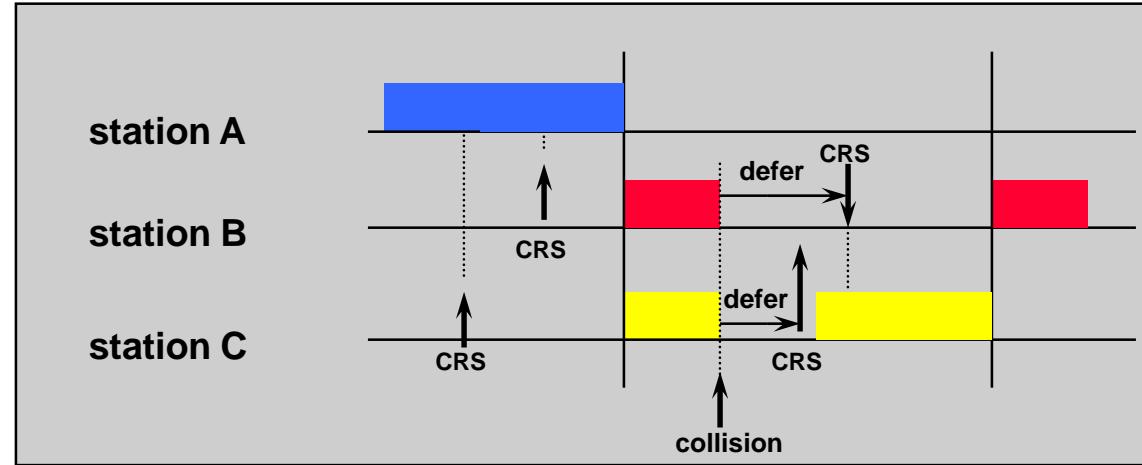
CSMA/CA Procedure



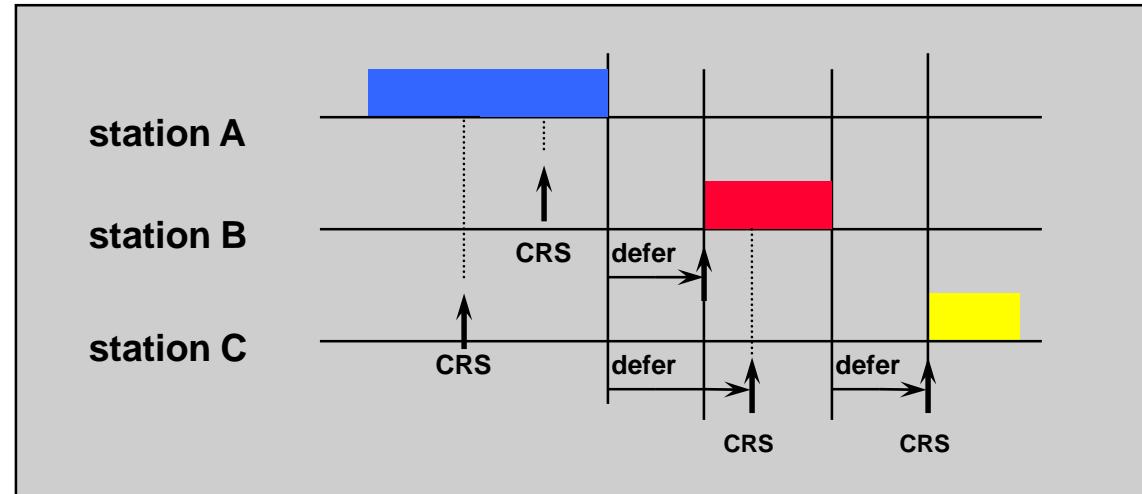
* Figure is courtesy of B. Forouzan

CSMA/CD and CSMA/CA

- CSMA/CD



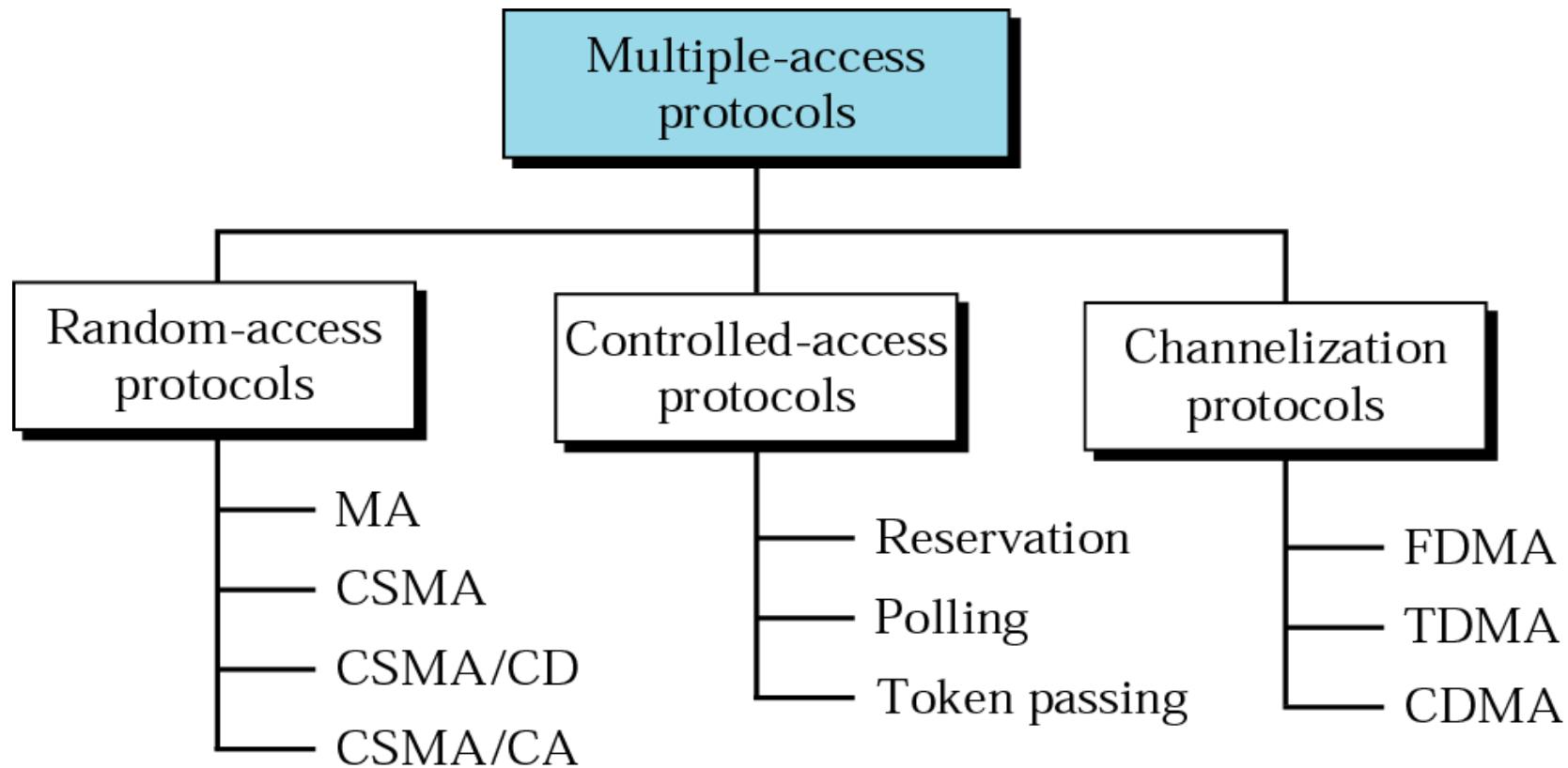
- CSMA/CA



CRS = Carrier Sense

* Figure is courtesy of Avaya Communications Inc

Multiple-Access Protocols



* Figure is courtesy of B. Forouzan

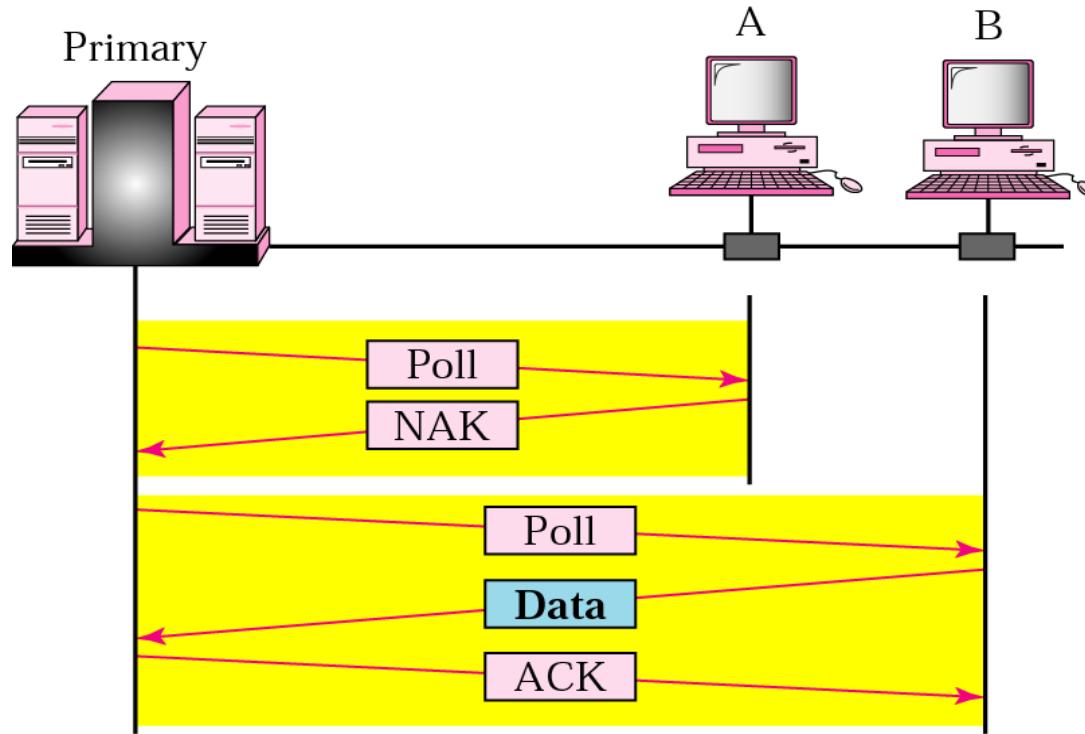


Overview of Today's Topics

- 802.11
- Ethernet
- Discussion of Sample Code
- Discussion of Assignment
- Discussion of Report



Poll



- Primary contacts stations to determine if they have data to transmit

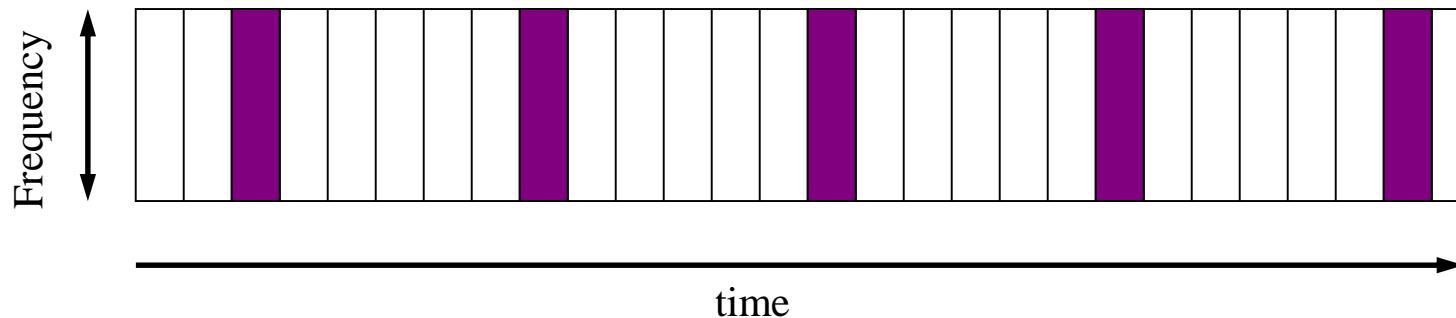
* Figure is courtesy of B. Forouzan

Static Channel Allocation

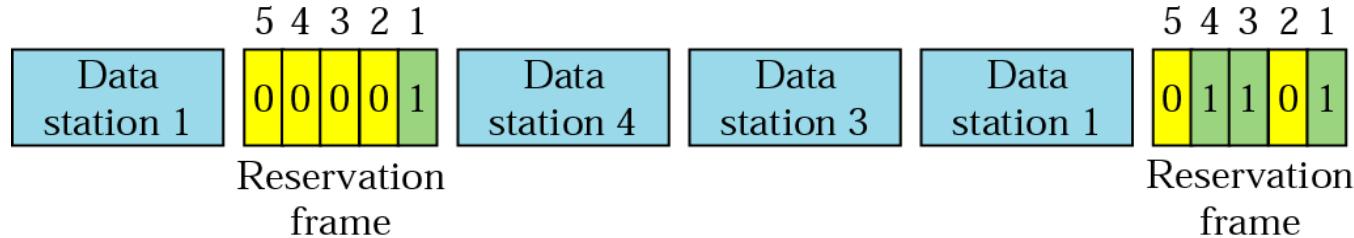
- Frequency Division Multiplexing (FDM)



- Time Division Multiplexing (TDM)



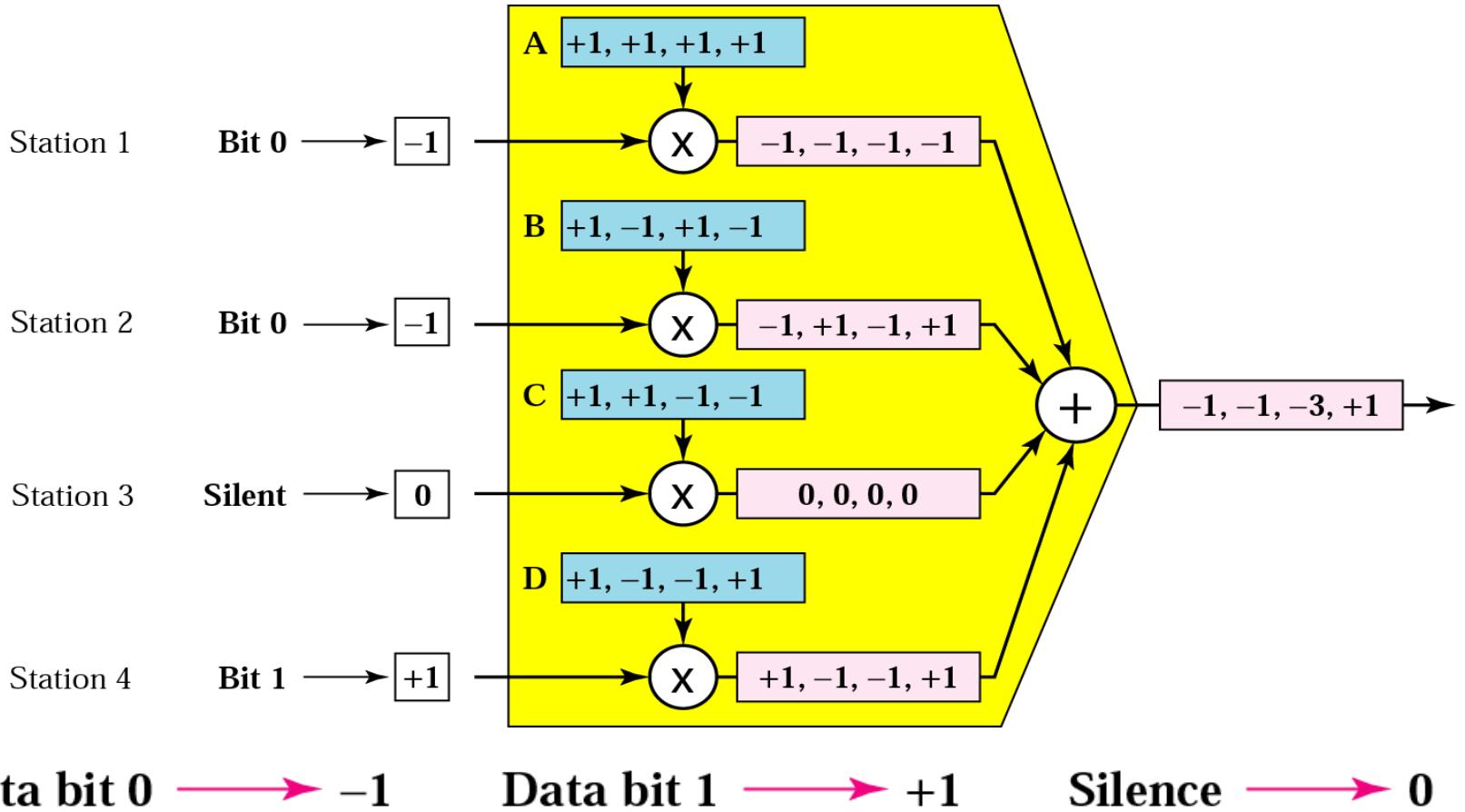
Reservation Access Method



- Station that wants to transmit data
 - transmits 1 during its slot in the reservation frame
- All stations are informed about all planned communication
- Limited number of pre-allocated slots/stations

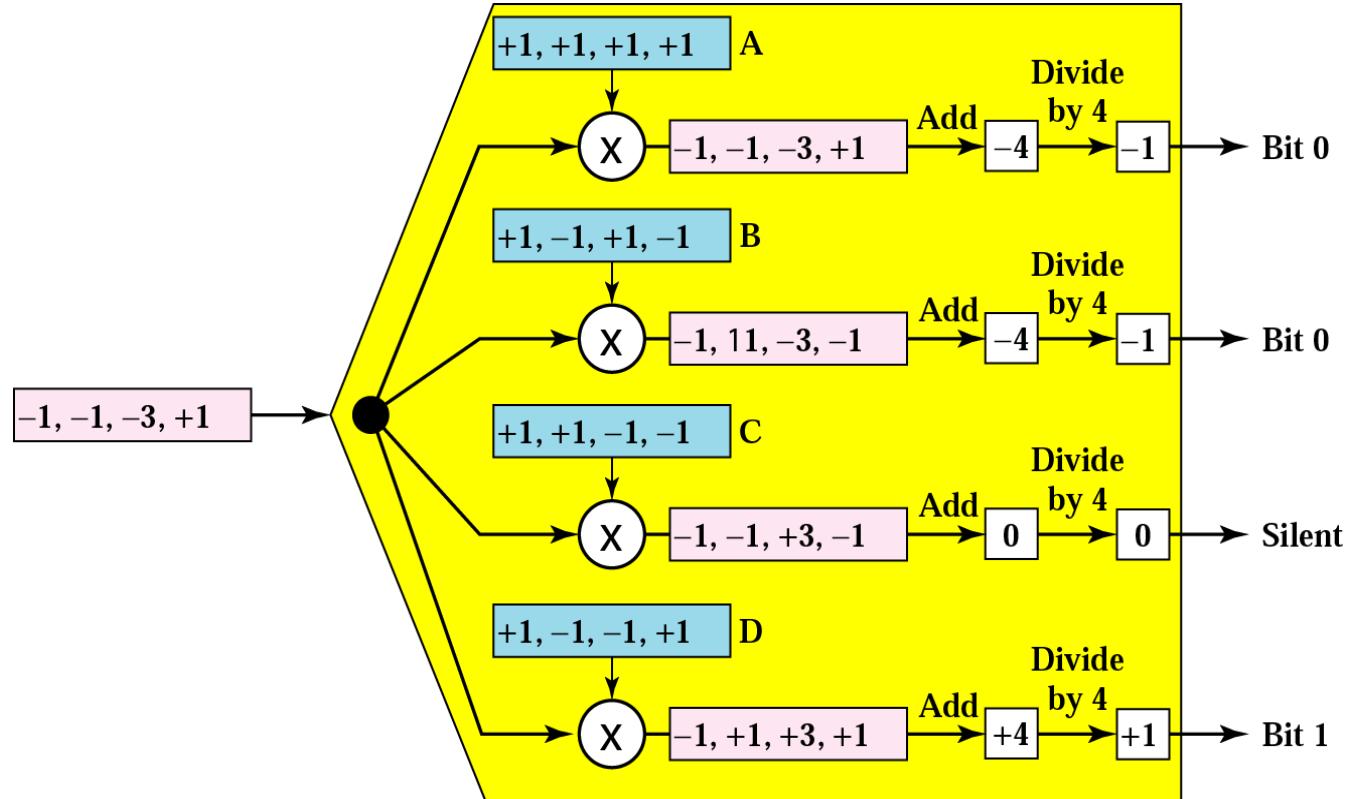
* Figure is courtesy of B. Forouzan

CDMA Multiplexer



* Figure is courtesy of B. Forouzan

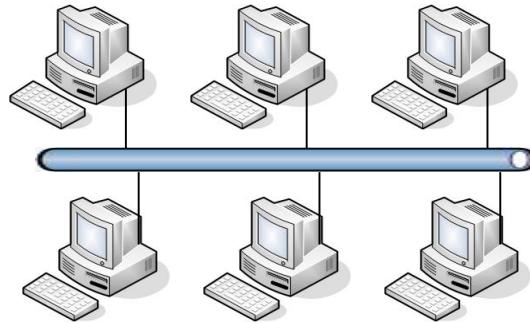
CDMA De-Multiplexer



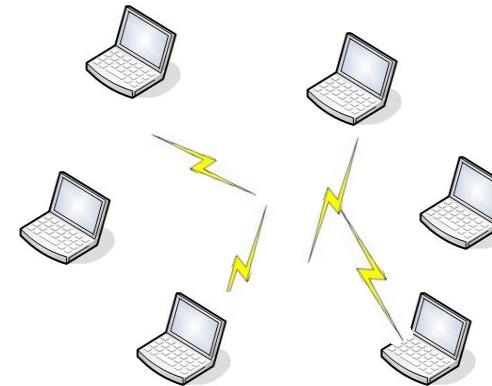
Decoding of received signal

* Figure is courtesy of B. Forouzan

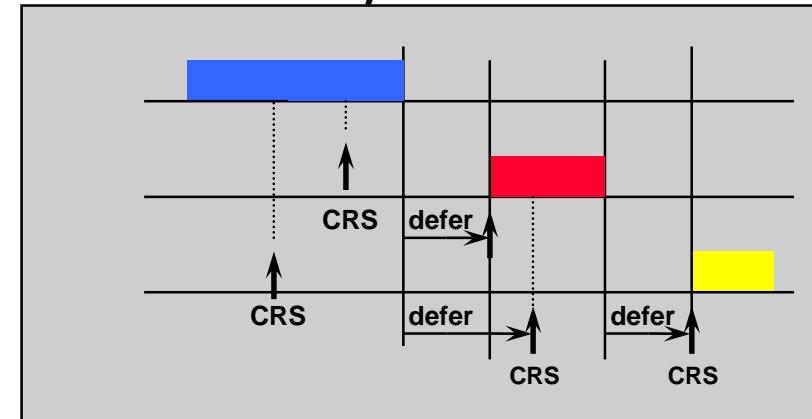
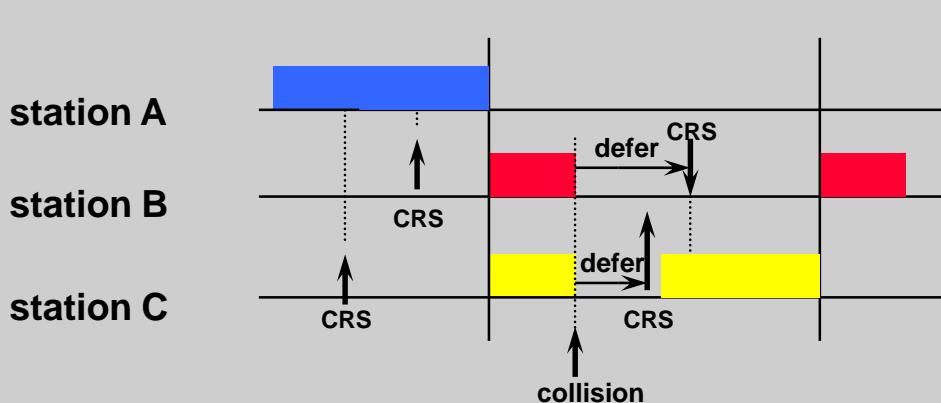
Collision Detection vs Collision Avoidance



CSMA/CD



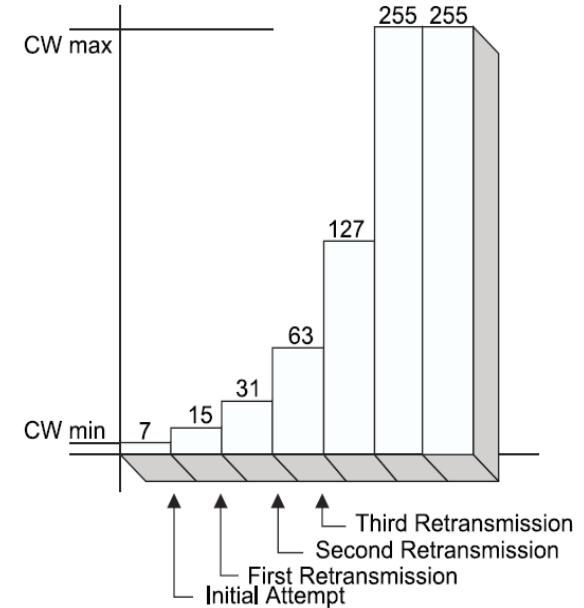
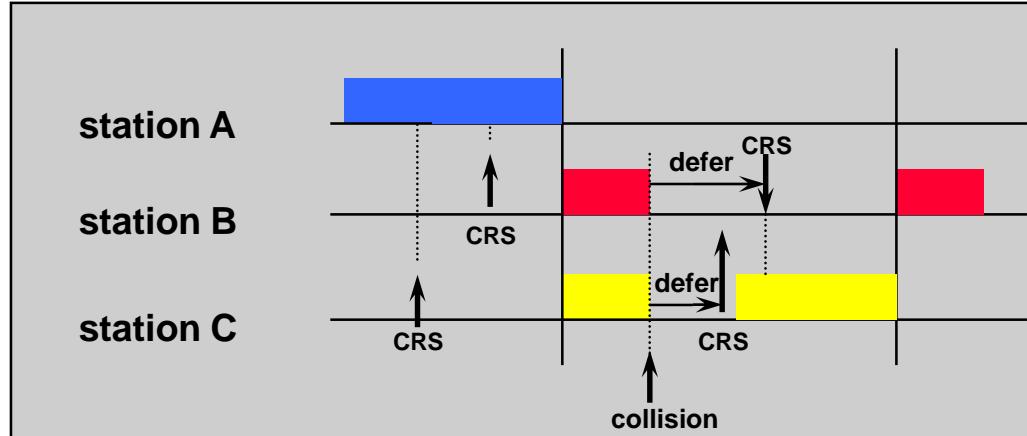
CSMA/CA



CRS = Carrier Sense

* Figure is courtesy of Avaya Communications Inc

Binary Exponential Backoff



$$\text{Backoff Time} = \text{Random()} \times \text{aSlotTime}$$

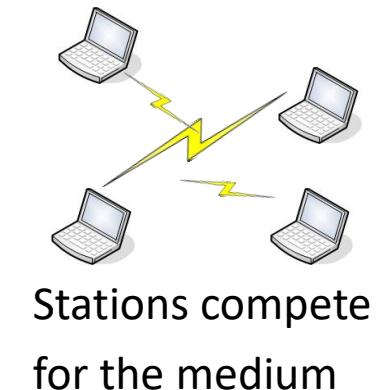
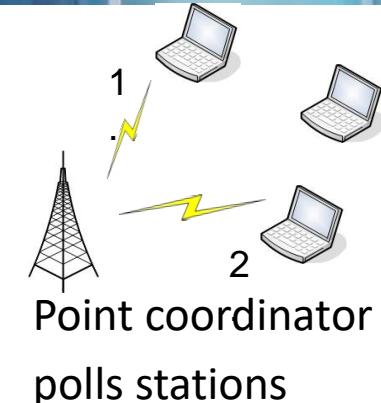
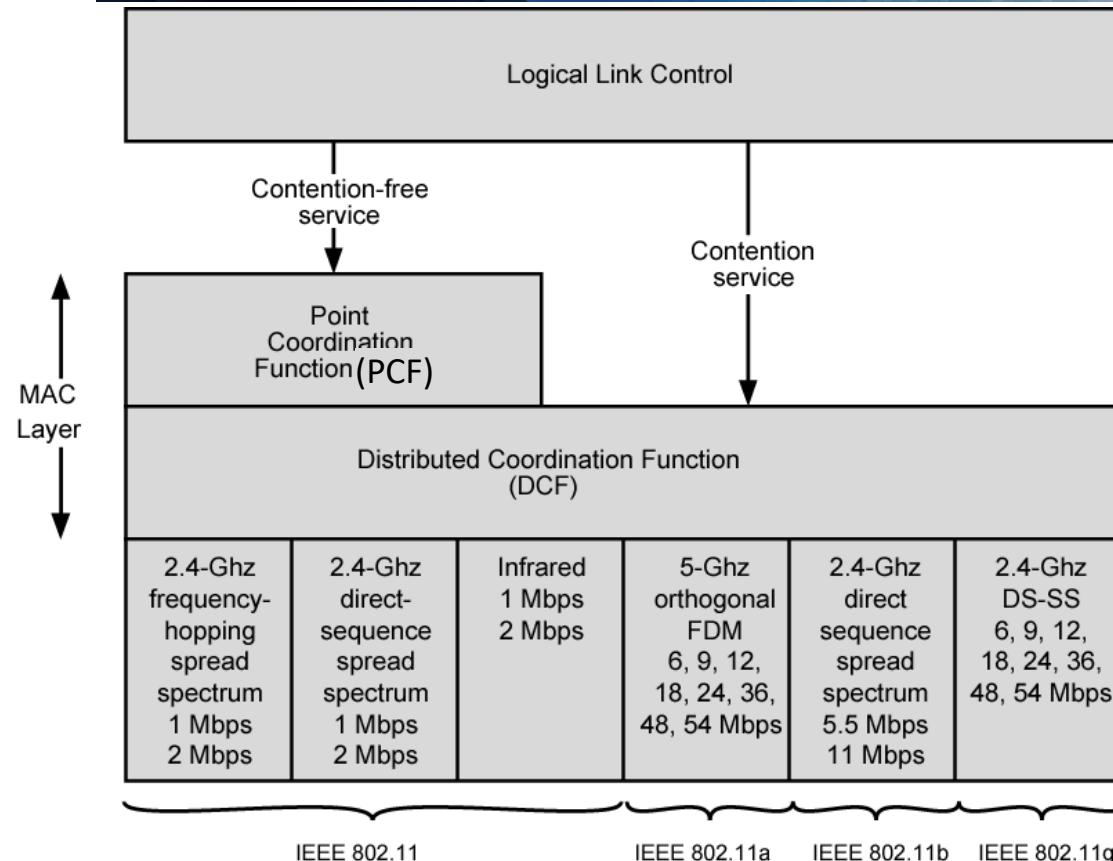
where

Random() = Pseudorandom integer drawn from a uniform distribution over the interval $[0, \text{CW}]$, where CW is an integer within the range of values of the PHY characteristics aCWmin and aCWmax , $\text{aCWmin} \leq \text{CW} \leq \text{aCWmax}$. It is important that designers recognize the need for statistical independence among the random number streams among STAs.

aSlotTime = The value of the correspondingly named PHY characteristic.

* Figure is courtesy of ANSI/IEEE Std 802.11 & Avaya Communications Inc

802.11 DCF & PCF



- PCF – Point Coordination Function – Access Points
- DCF – Distributed Coord. Function – between Stations

* Figure is courtesy of W. Stallings



CS2031

Telecommunications II

802.11

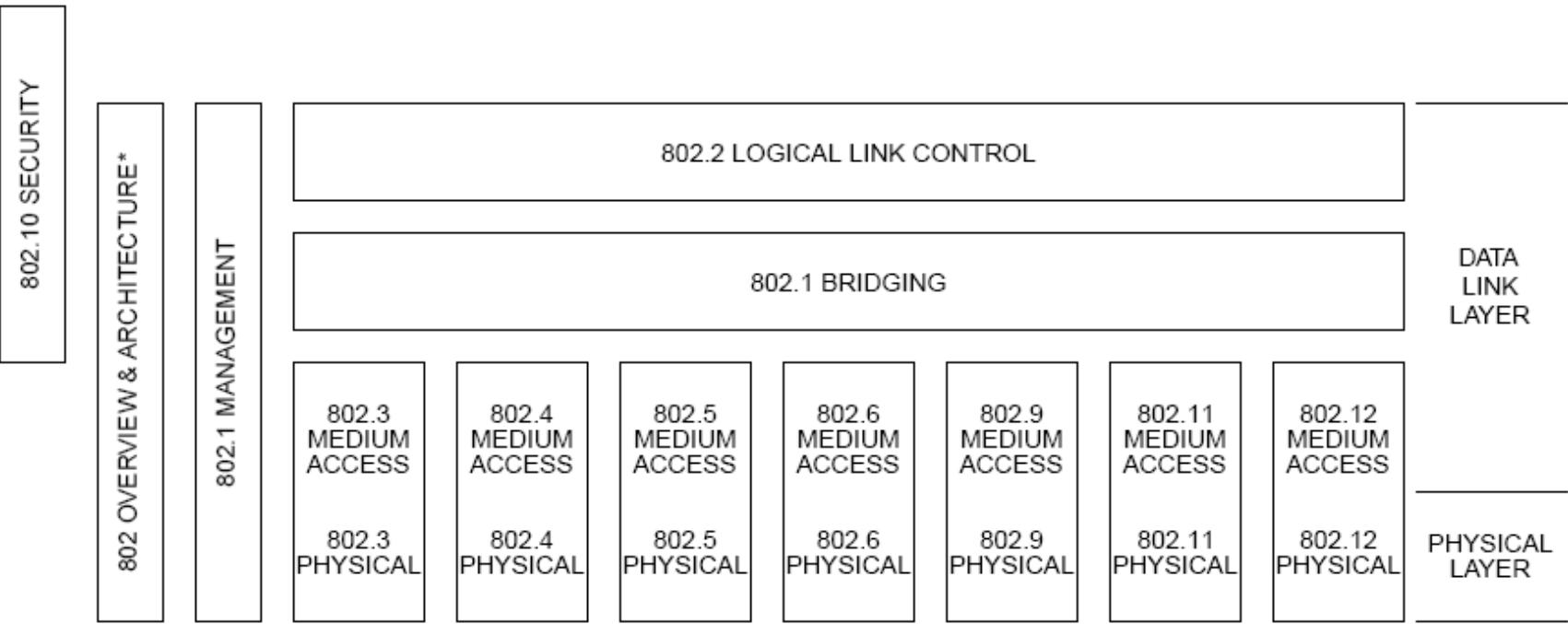


802.11

- DCF \Rightarrow Distributed Coordination Function
 - Stations compete for access to the medium
 - Hidden Station / Expose Station Problem
 - CSMA/CA + RTS/CTS
- PCF \Rightarrow Point Coordination Function
 - Access point polls stations
- IFS \Rightarrow Inter-Frame Space
 - Time between frames



IEEE 802



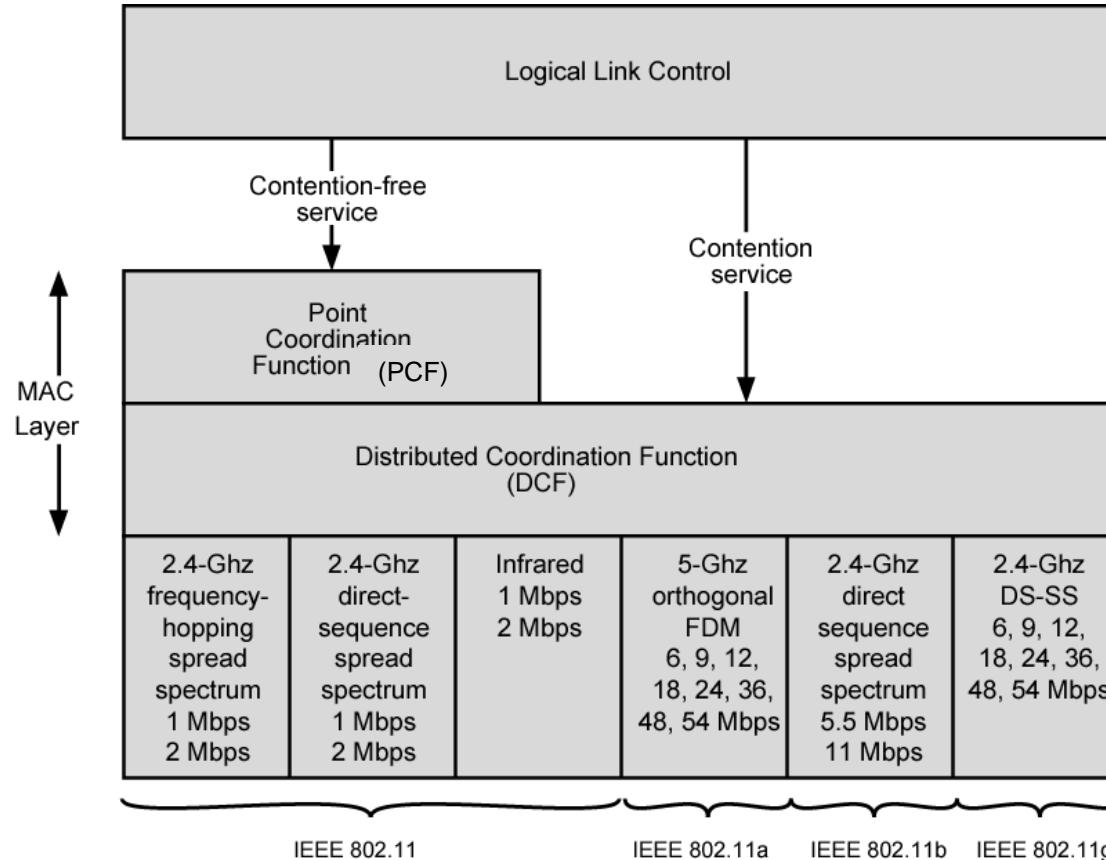
- 802.3: Ethernet
- 802.5: Token Ring
- 802.11: Wifi
- 802.16: WiMAX
- 802.15.1: Bluetooth
- 802.15.4: ZigBee



IEEE 802.11 Standards

- **IEEE 802.11** - The original 1 Mbit/s and 2 Mbit/s, 2.4 GHz RF and IR standard
- **IEEE 802.11a** - 54 Mbit/s, 5 GHz standard
- **IEEE 802.11b** - Enhancements to 802.11 to support 5.5 and 11 Mbit/s
 - IEEE 802.11c - Bridge operation procedures; included in the IEEE 802.1D standard
 - IEEE 802.11d - International (country-to-country) roaming extensions
 - IEEE 802.11e - Enhancements: QoS
 - IEEE 802.11F - Inter-Access Point Protocol
- **IEEE 802.11g** - 54 Mbit/s, 2.4 GHz standard
 - IEEE 802.11h - Spectrum Managed 802.11a (5 GHz)
 - IEEE 802.11i - Enhanced security
 - IEEE 802.11j - Extensions for Japan
 - IEEE 802.11k - Radio resource measurement enhancements
 - IEEE 802.11m - Maintenance of the standard; odds and ends.
- **IEEE 802.11n** - Higher throughput improvements
 - IEEE 802.11p - WAVE - Wireless Access for the Vehicular Environment
 - IEEE 802.11r - Fast roaming
 - **IEEE 802.11s - ESS Mesh Networking**
 - IEEE 802.11t - Wireless Performance Prediction (WPP)
 - IEEE 802.11u - Interworking with non-802 networks
 - IEEE 802.11v - Wireless network management
 - IEEE 802.11w - Protected Management Frames

802.11 Structure

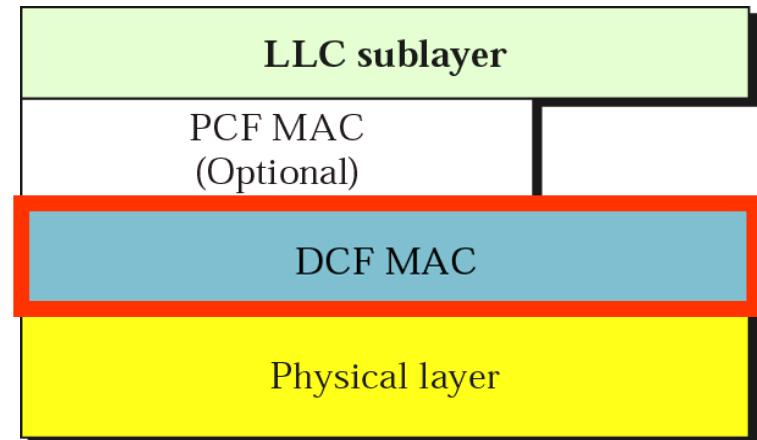


- PCF – Point Coordination Function – Access Points
- DCF – Distributed Coord. Function – between Stations

* Figure is courtesy of W. Stallings

Distributed Coordination Function (DCF)

- Stations compete for access to the medium
- Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)
- No collision detection
 - Not practical on wireless network
 - Transmitting station cannot distinguish incoming weak signals from noise and effects of own transmission

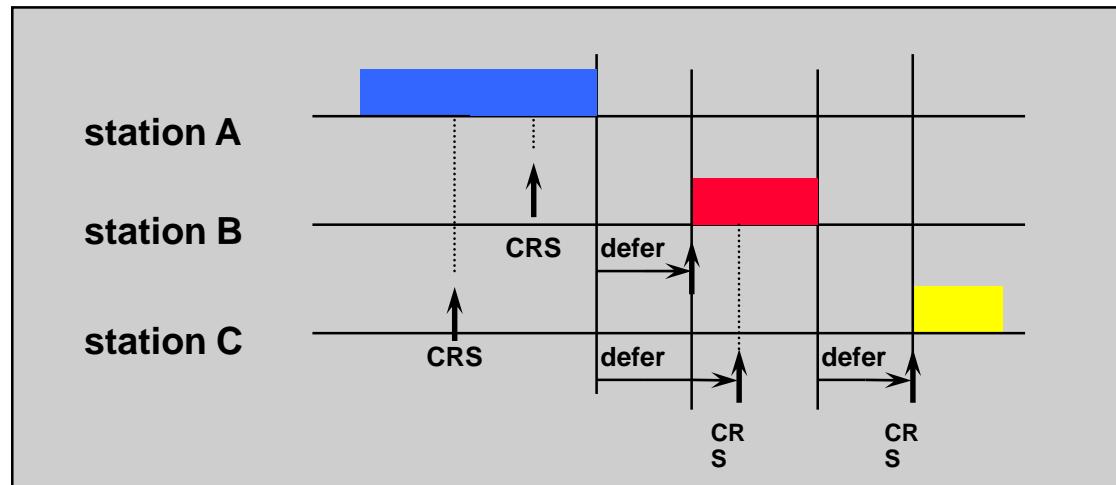


* Figure is courtesy of B. Forouzan

CSMA in Wireless Media



Collision is at the receiver !!!

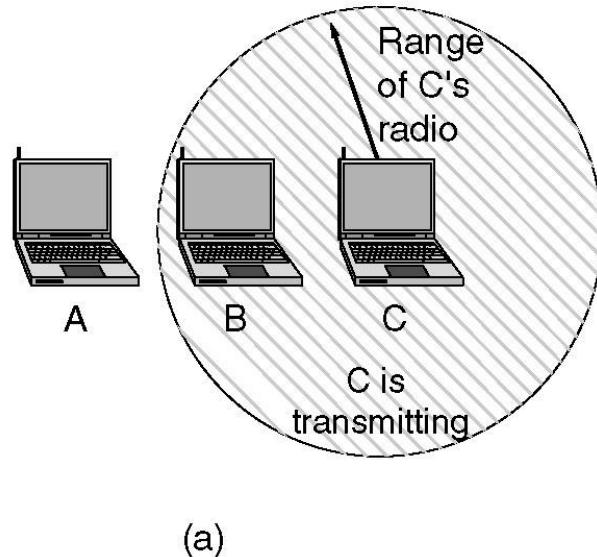


- Sense carrier to determine if medium is free
- Once free pick a random number
 - then start sending

* Figure is courtesy of Avaya Communications Inc

Hidden Station Problem

A wants to send to B
but cannot hear that
B is busy

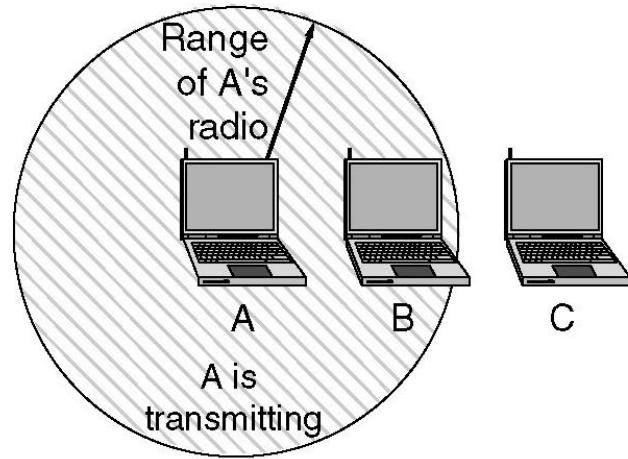


(a) Hidden Station Problem

* Figure is courtesy of A. Tanenbaum

Exposed Station Problem

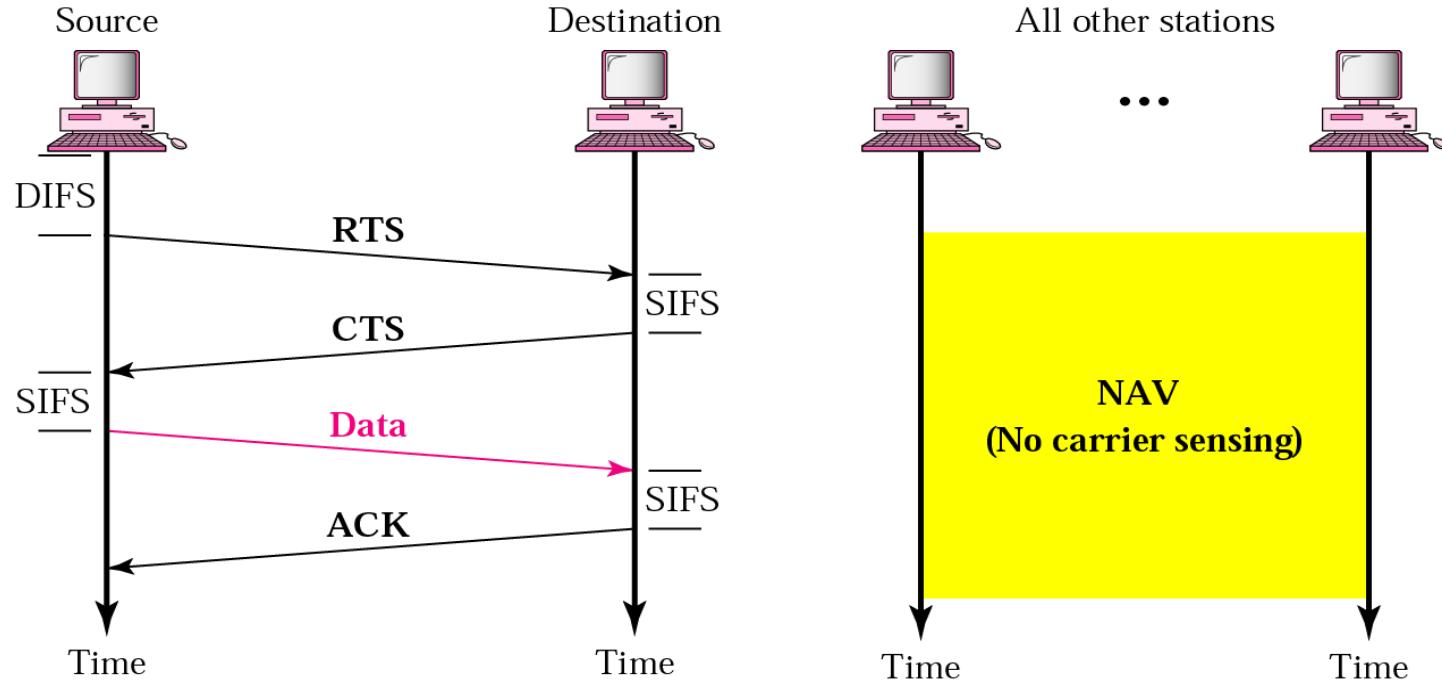
B wants to send to C
but mistakenly thinks
the transmission will fail



(b)

(b) Exposed Station Problem

CSMA/CA and NAV

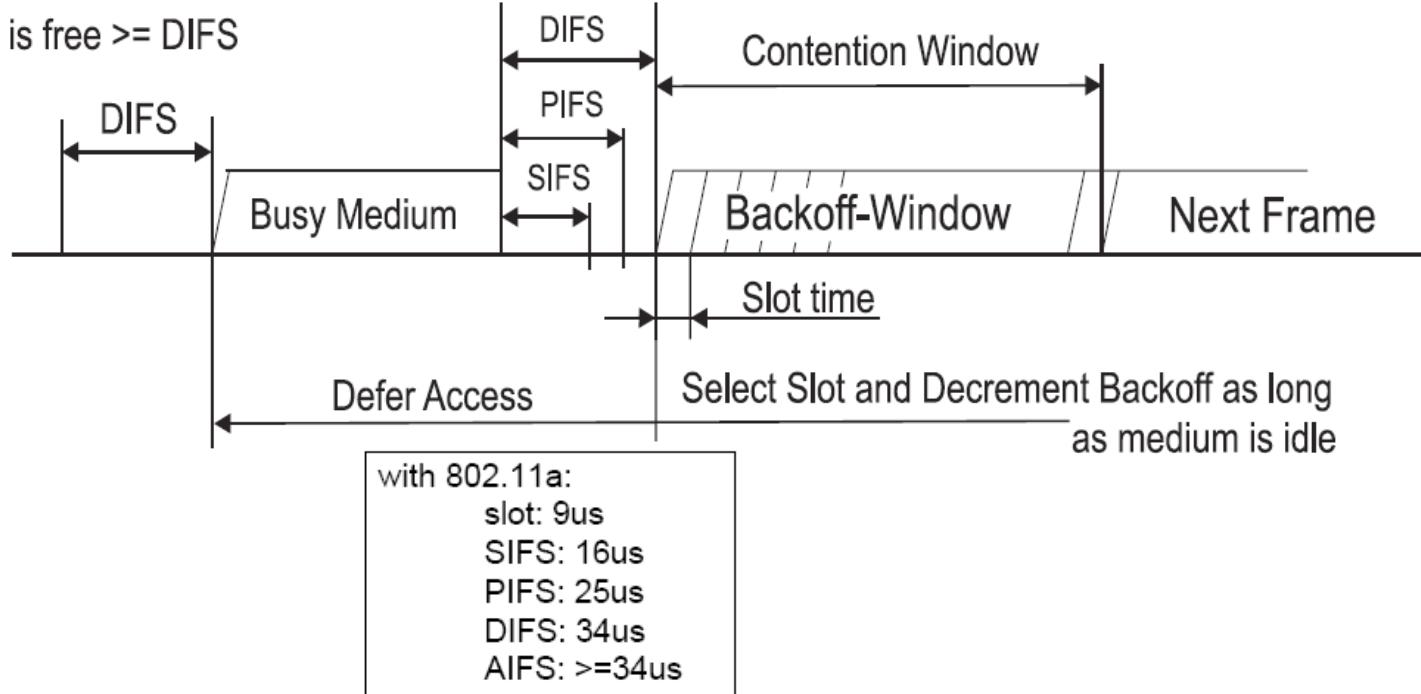


- Ready-To-Send (RTS) announces the intention to send traffic
- Clear-To-Send (CTS) announces that the receiving station is ready
- SIFS is the smallest possible Inter-Frame Space that separates two transmissions
- The Network Allocation Vector (NAV) as part of RTS/CTS announces the length of the subsequent transmission

* Figure is courtesy of B. Forouzan

Inter-Frame Space (IFS)

Immediate access when medium is free \geq DIFS

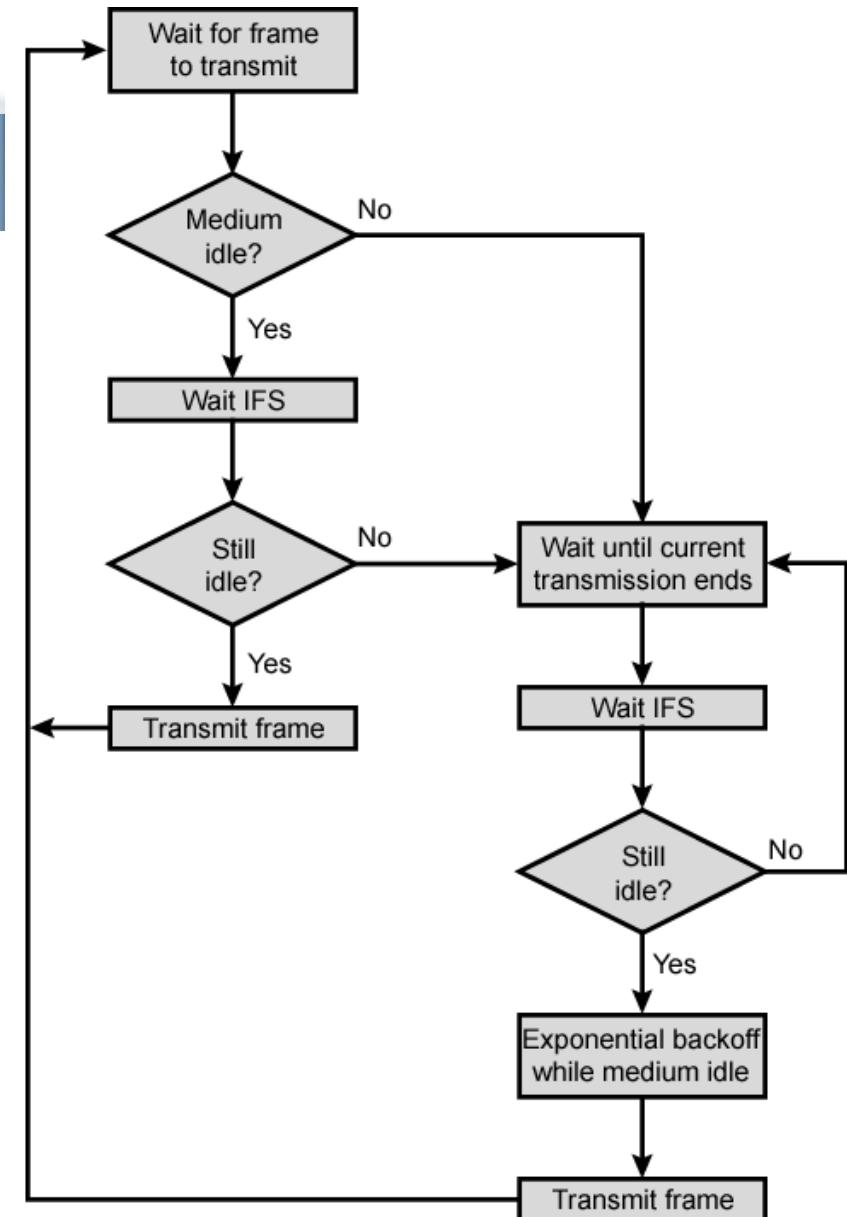


- Short IFS (SIFS) defines the minimum time between frames
- DCF IFS (DIFS) defines the time between the end of one transmission and the beginning of a subsequent transmission

* Figure is courtesy of ANSI/IEEE Std 802.11

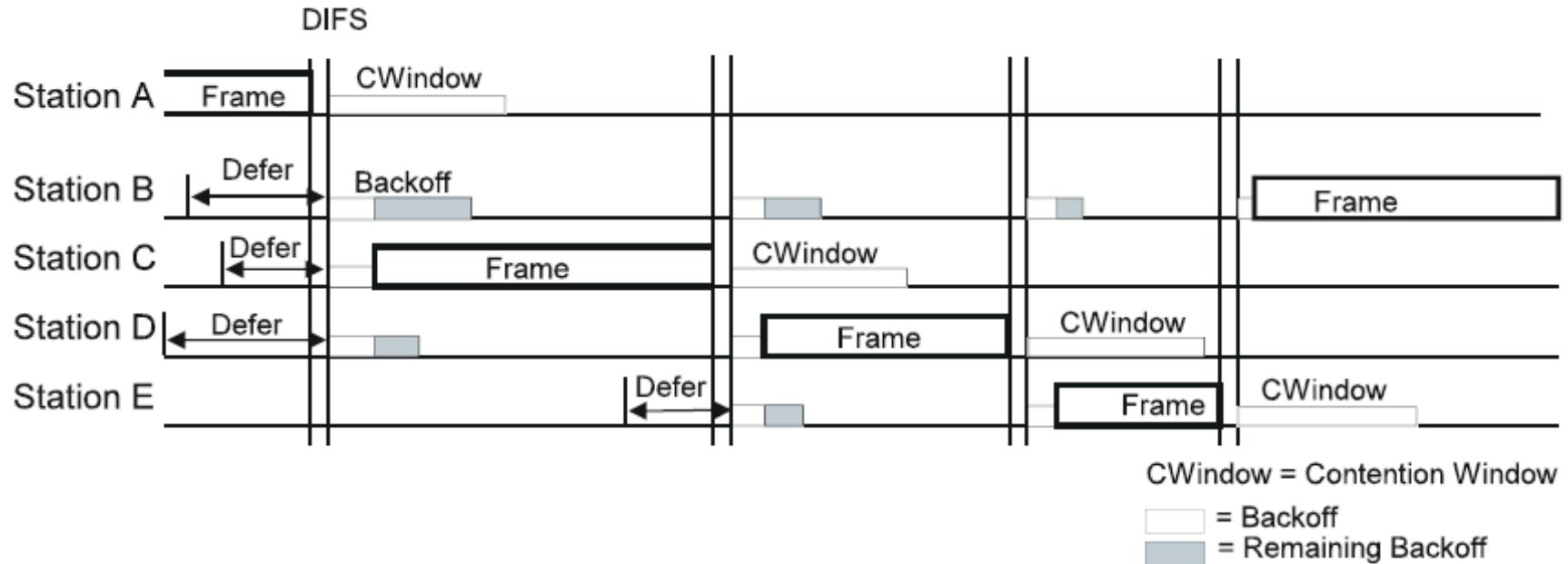
802.11 MAC

- Station with frame senses medium
 - If idle, wait to see if remains idle for one IFS.
 - If so, may transmit immediately
- If busy - either initially or becomes busy during IFS - station defers transmission
 - Continue to monitor until current transmission is over
- Once current transmission over, delay for another IFS
 - If remains idle, back off random time and again sense
 - If medium still idle, station may transmit
 - During backoff time, if becomes busy, backoff timer is halted and resumes when medium becomes idle



* Figure is courtesy of W. Stallings

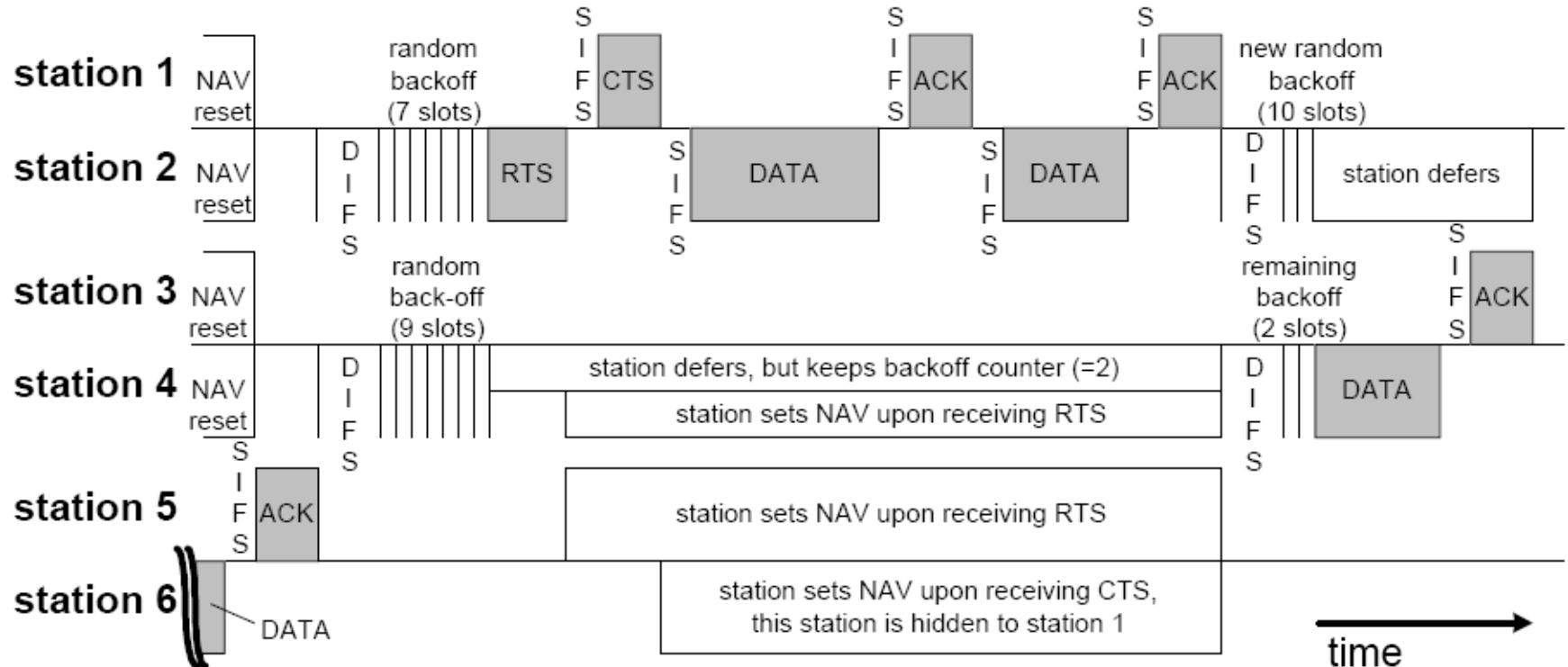
Contention & Backoff



- DIFS defines the minimum time between the end of one transmission and the beginning of a subsequent transmission
- All stations that want to send sense the medium
- Once the sending station is silent all stations start their DIFS timer
- After the DIFS timer every station starts a random exponential backoff

* Figure is courtesy of ANSI/IEEE Std 802.11

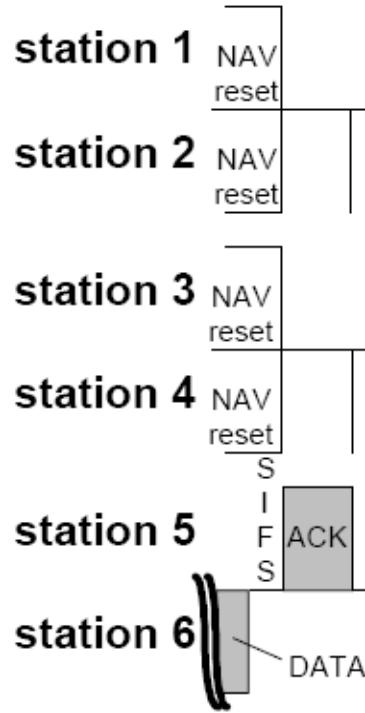
DCF & RTS/CTS



with 802.11a:
 slot: 9us
 SIFS: 16us
 PIFS: 25us
 DIFS: 34us
 AIFS: >=34us

* Figure is courtesy of W. Stallings

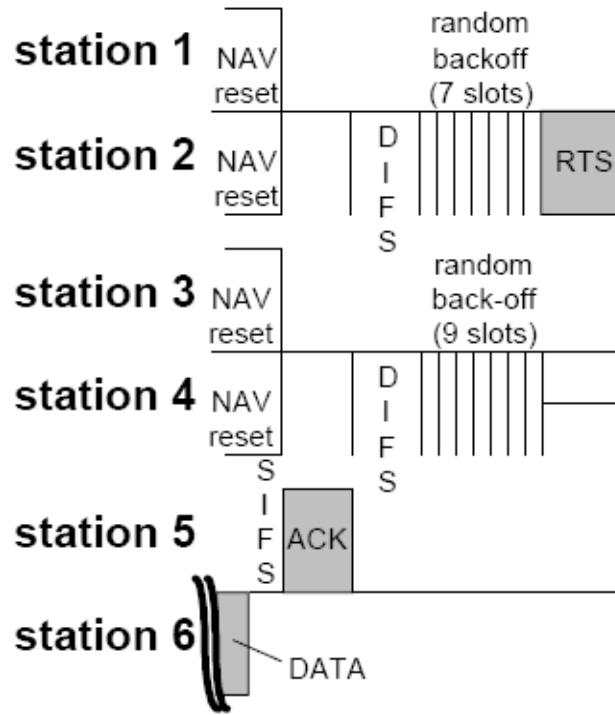
DCF



with 802.11a:
slot: 9us
SIFS: 16us
PIFS: 25us
DIFS: 34us
AIFS: ≥ 34 us



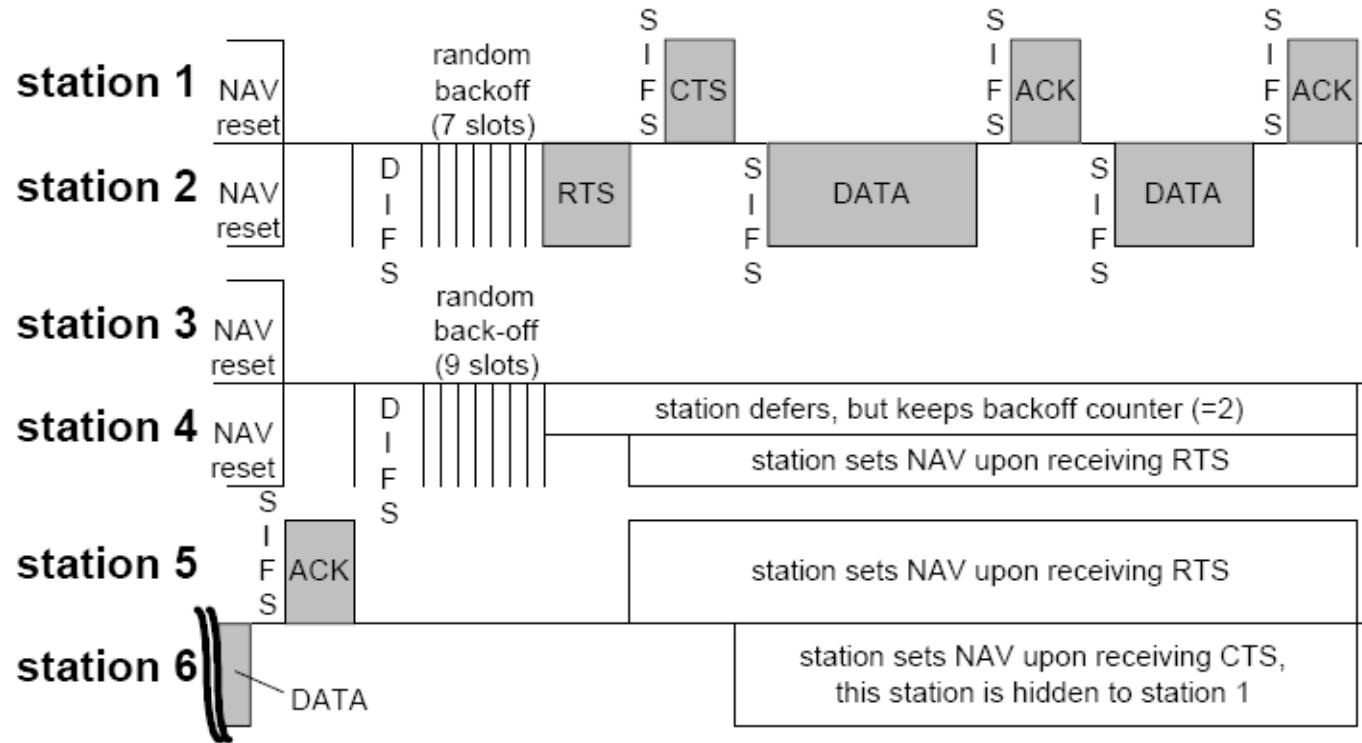
DCF



with 802.11a:
 slot: 9us
 SIFS: 16us
 PIFS: 25us
 DIFS: 34us
 AIFS: ≥ 34 us

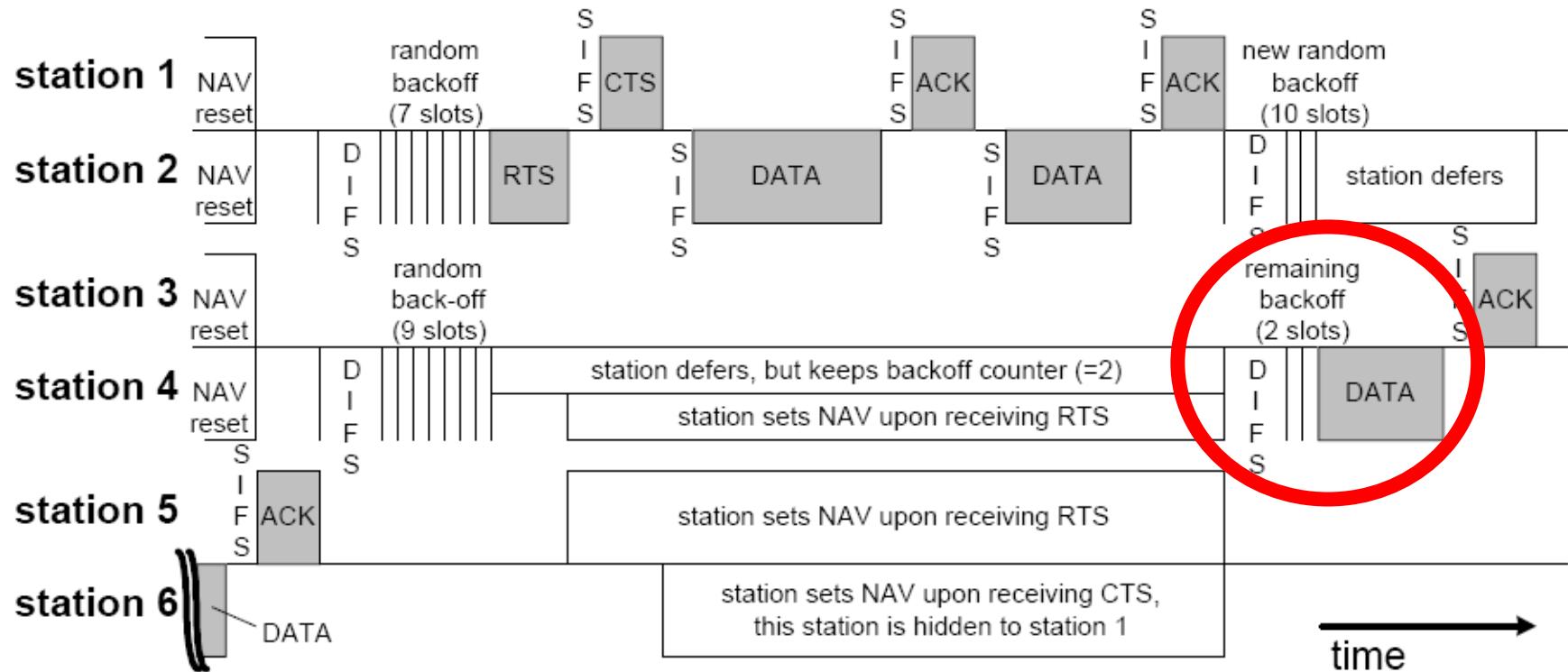


DCF



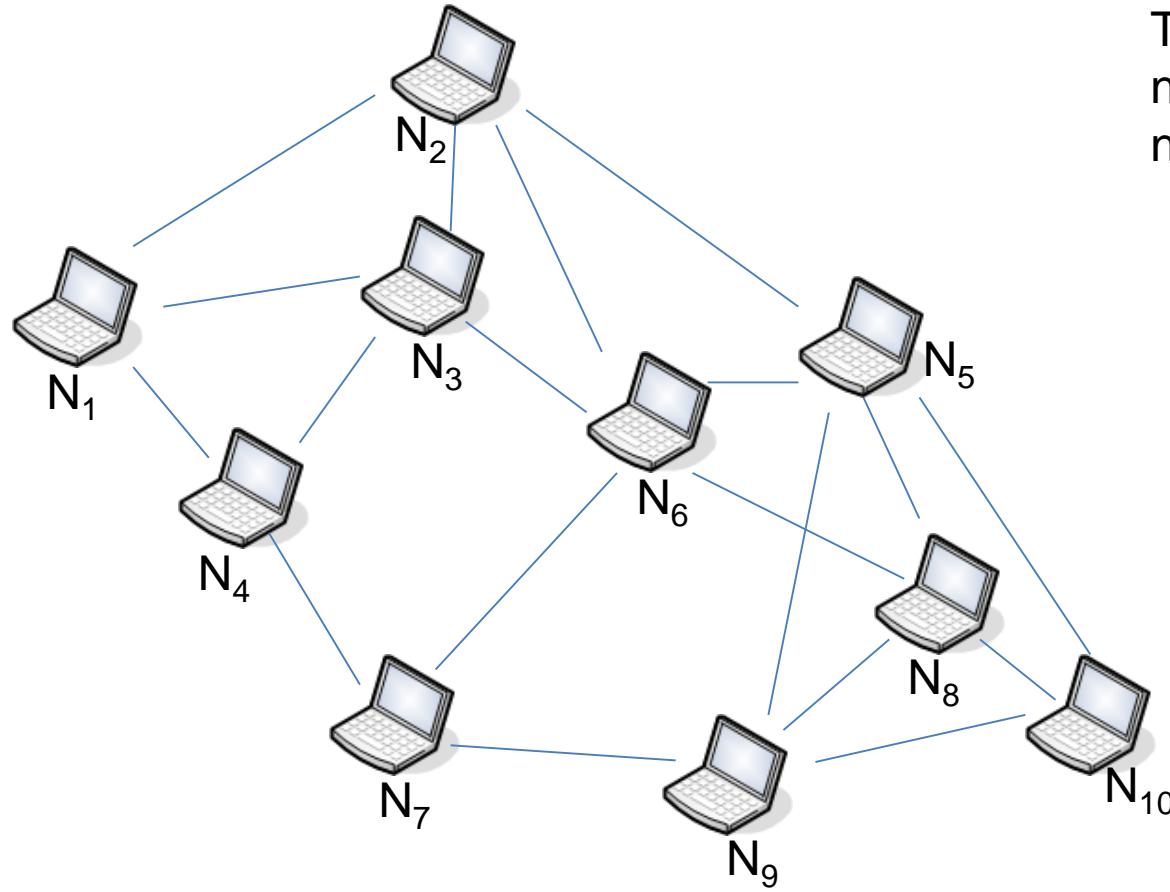
with 802.11a:
 slot: 9us
 SIFS: 16us
 PIFS: 25us
 DIFS: 34us
 AIFS: $\geq 34\text{us}$

DCF



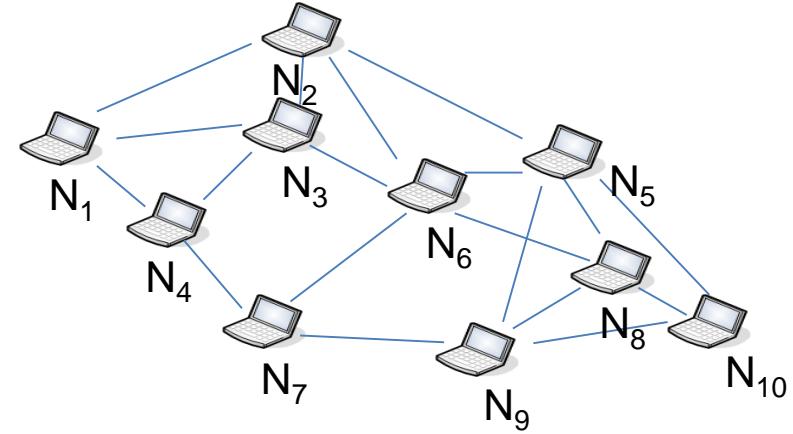
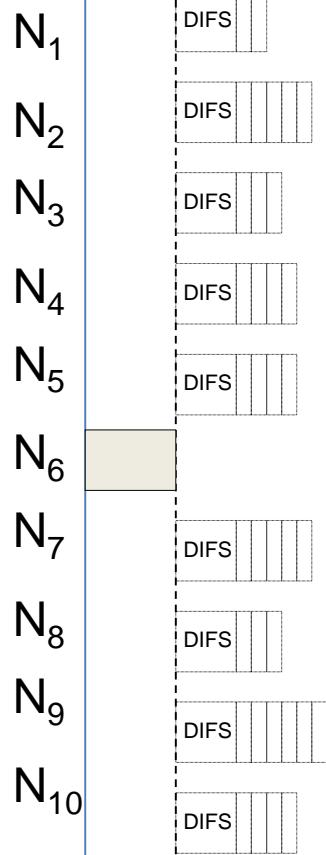
with 802.11a:
 slot: 9us
 SIFS: 16us
 PIFS: 25us
 DIFS: 34us
 AIFS: ≥ 34 us

DCF Example

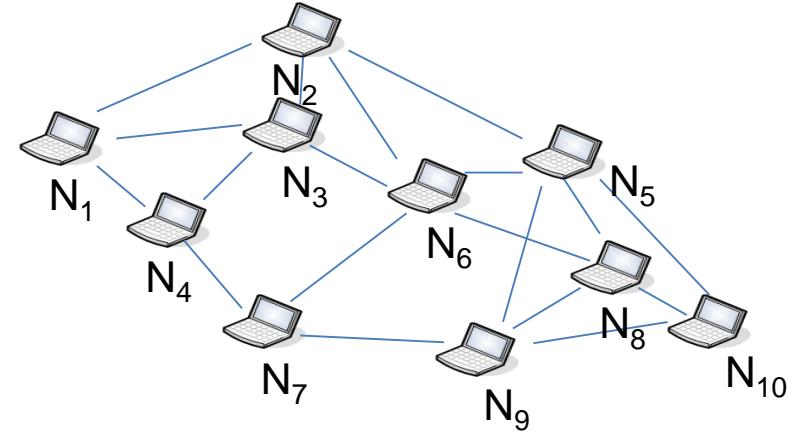
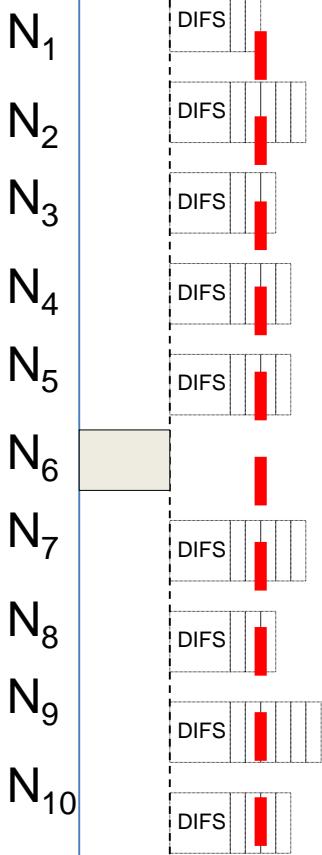


The lines indicate which nodes will receive their neighbours signals

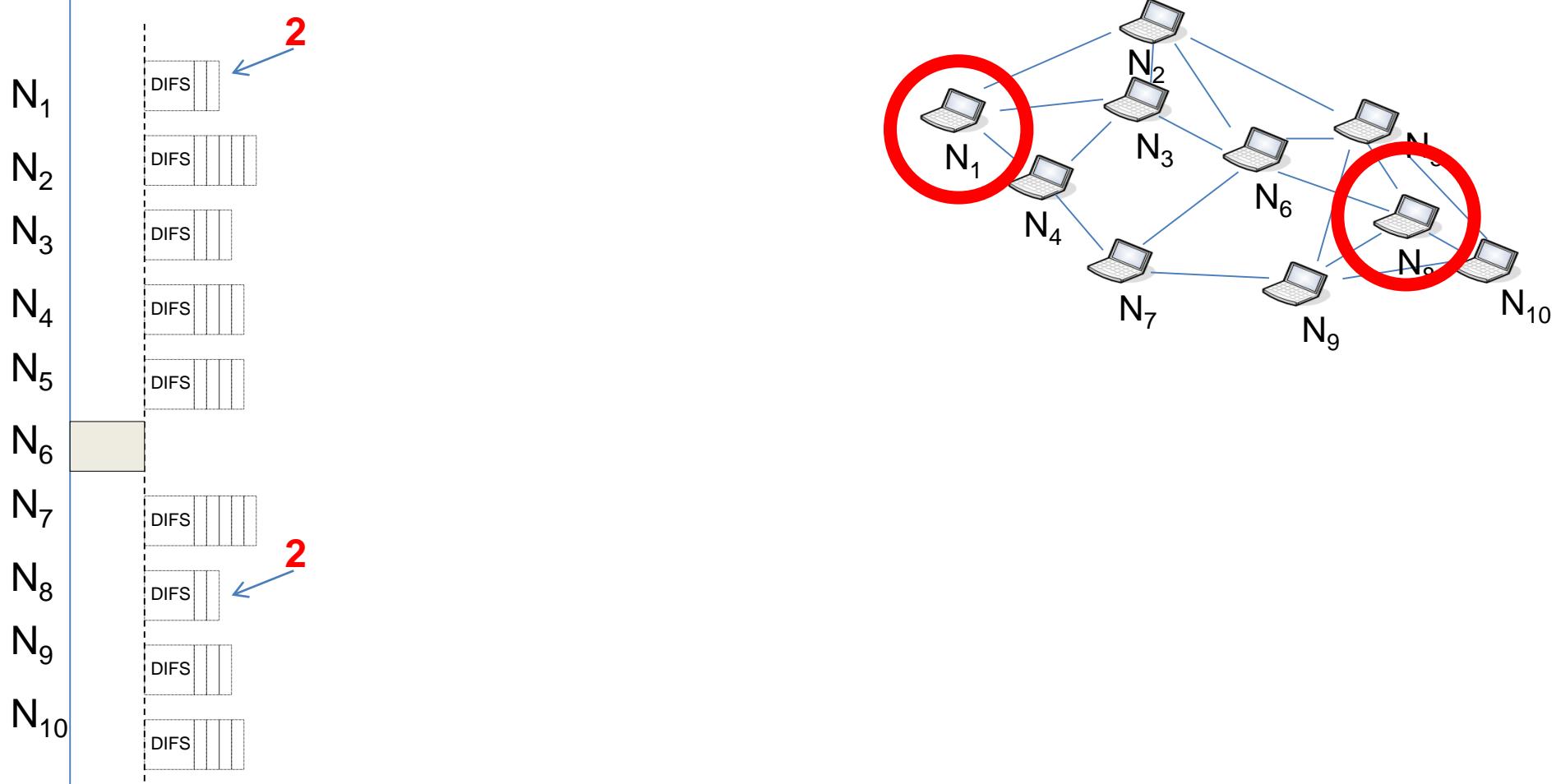
DCF Example



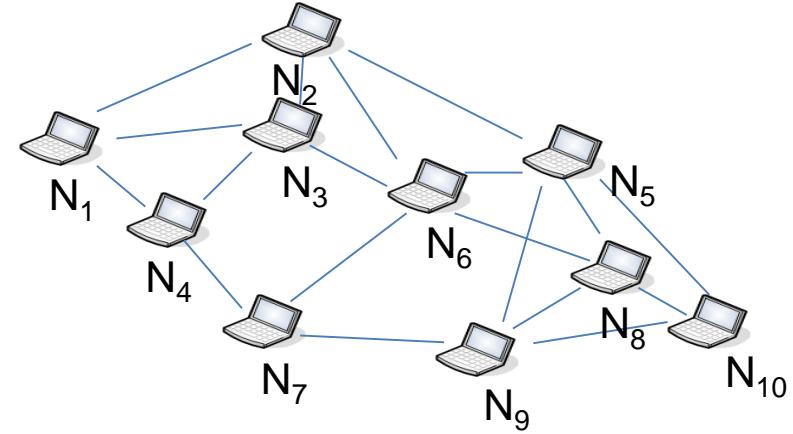
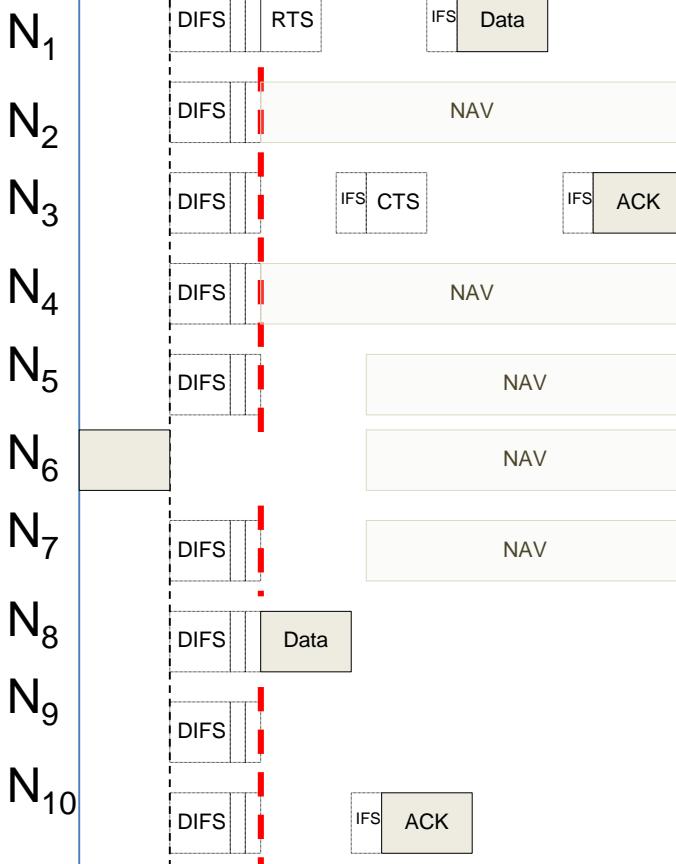
DCF Example



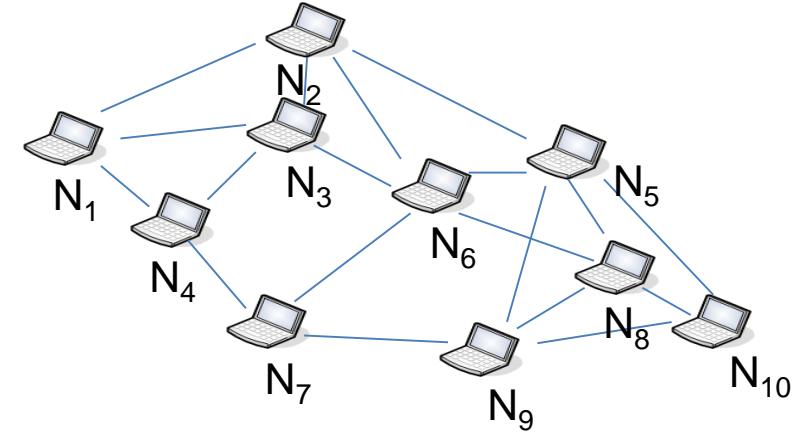
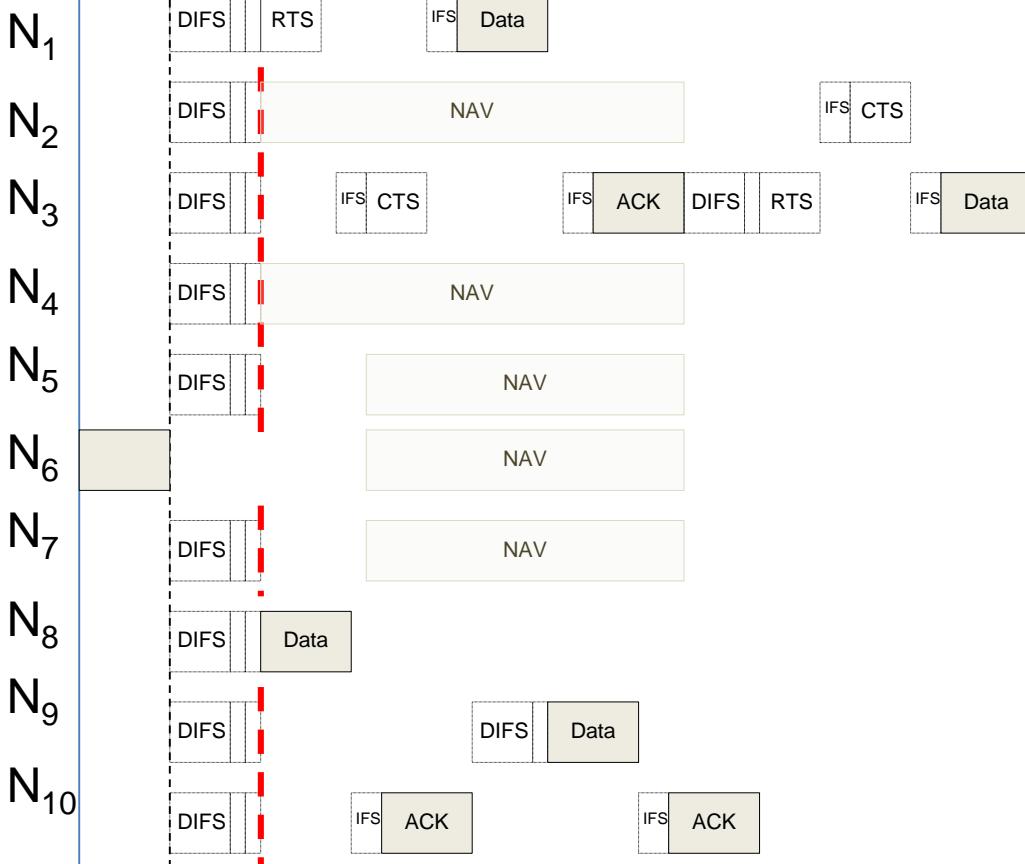
DCF Example



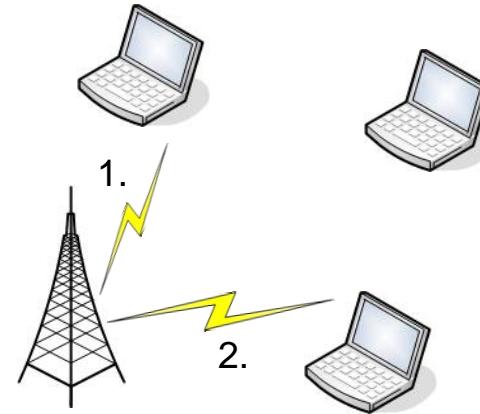
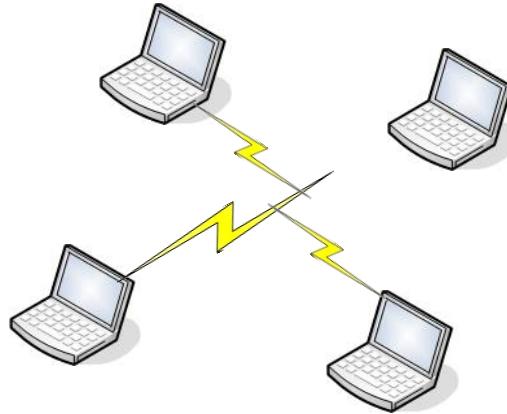
DCF Example



DCF Example



DCF vs. PCF

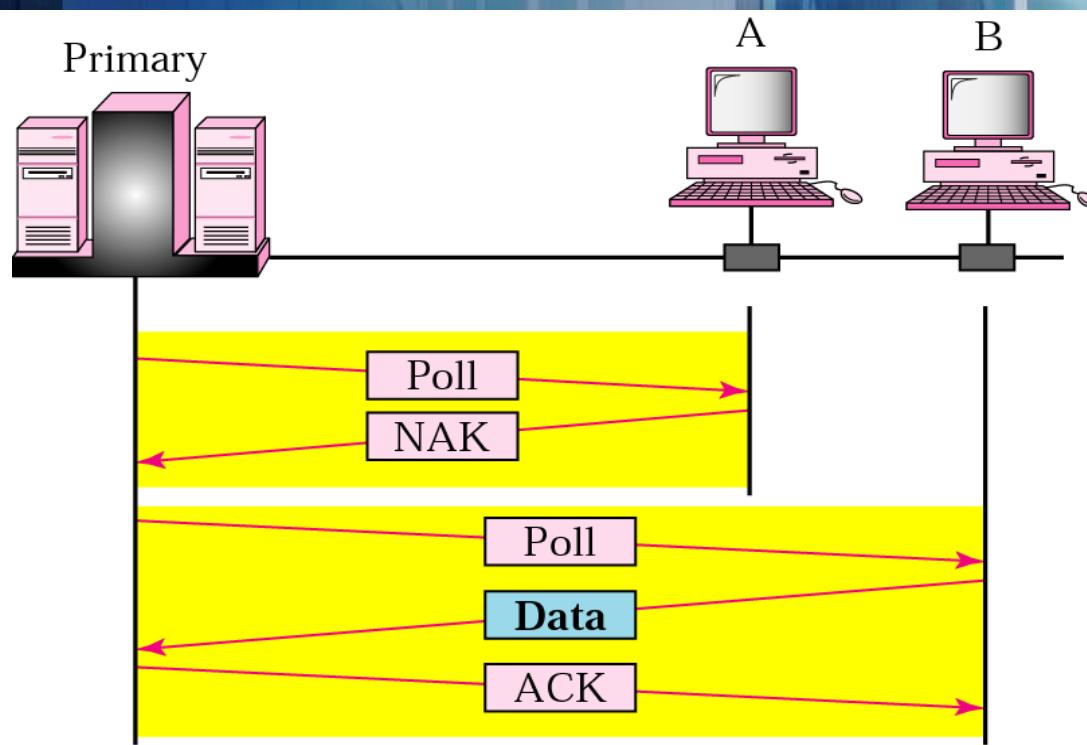


- Stations compete for the medium
- Point coordinator polls stations

Point Coordination Function (PCF)

- Used by access points
- Polling by centralized polling master – or point coordinator
- Uses PIFS
 - PIFS smaller than DIFS
 - Gives coordinator priority over individual stations
- Point coordinator polls in round-robin to stations configured for polling
 - When poll issued, polled station may respond within SIFS
 - If point coordinator receives response, it issues another poll

Poll

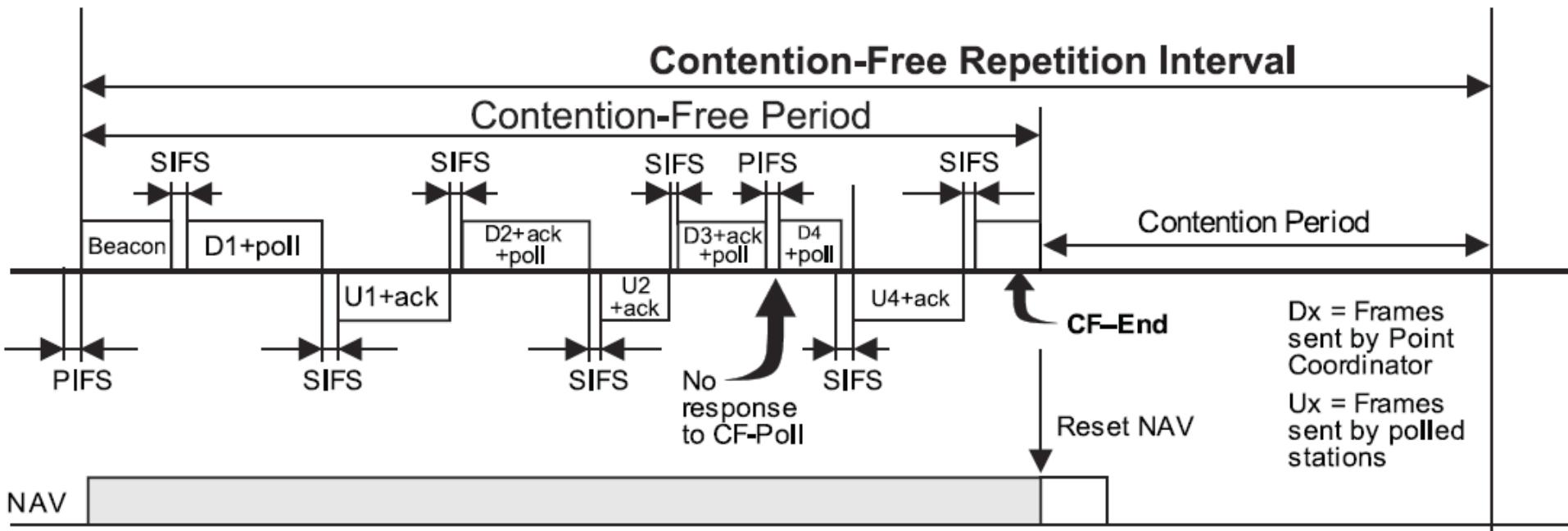


- Primary contacts stations to determine if they have data to transmit

* Figure is courtesy of B. Forouzan

Contention Free Period

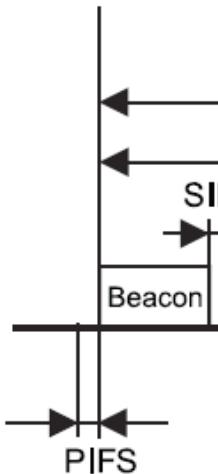
- Time= Contention Period + Contention Free Period
 - Contention Period: All stations compete for the medium
 - Contention Free Period: The AP coordinates communication



* Figure is courtesy of ANSI/IEEE Std 802.11

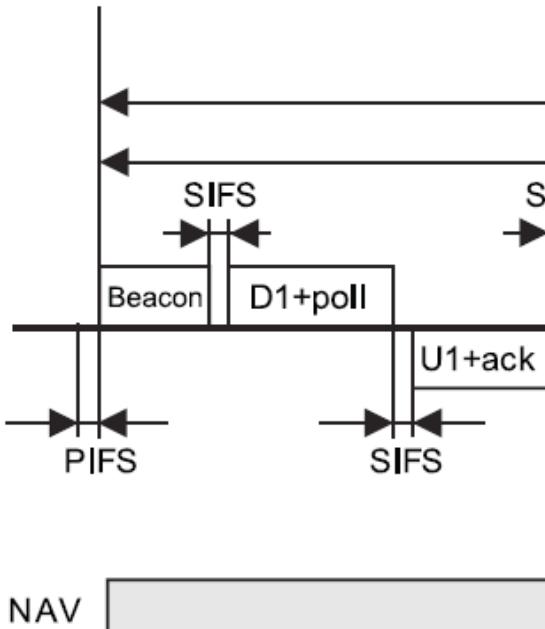
PCF

- Time= Contention Period + Contention Free Period
 - Contention Period: All stations compete for the medium
 - Contention Free Period: The AP coordinates communication



PCF

- Time= Contention Period + Contention Free Period
 - Contention Period: All stations compete for the medium
 - Contention Free Period: The AP coordinates communication



NAV

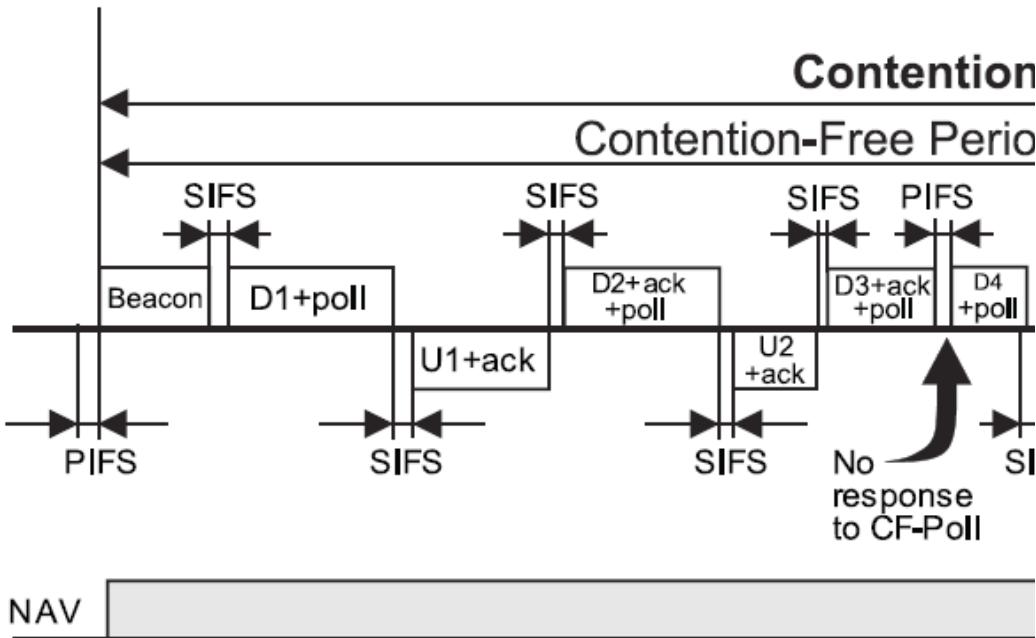


TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

THE
UNIVERSITY
OF DUBLIN

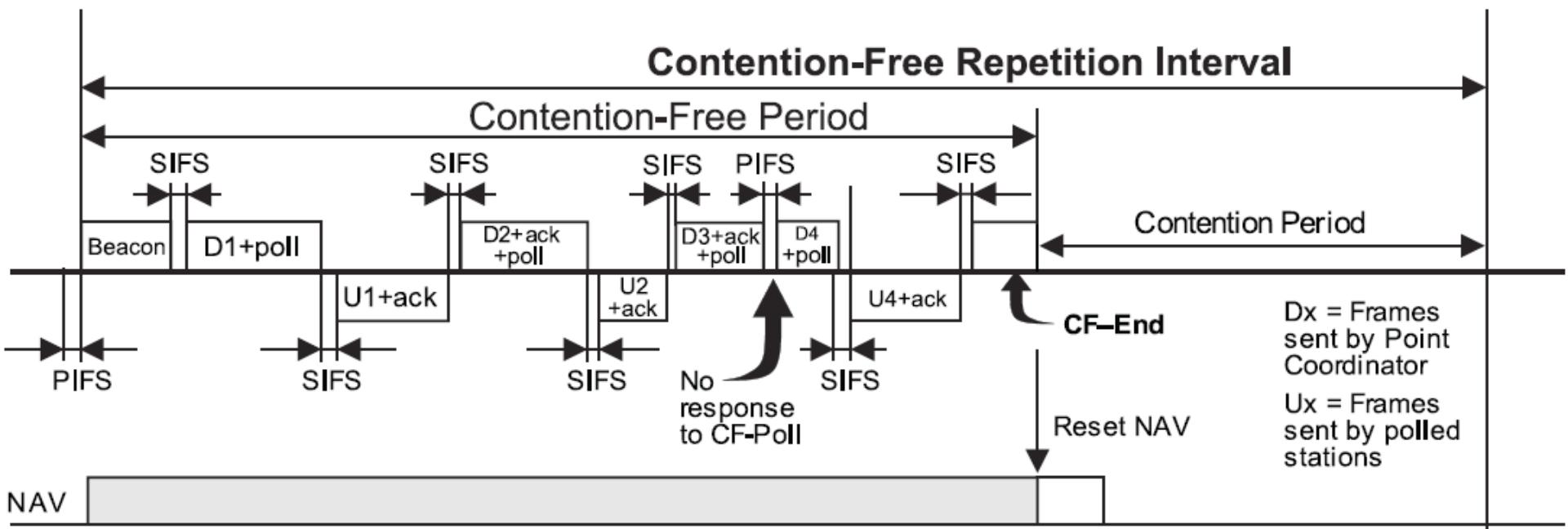
PCF

- Time= Contention Period + Contention Free Period
 - Contention Period: All stations compete for the medium
 - Contention Free Period: The AP coordinates communication

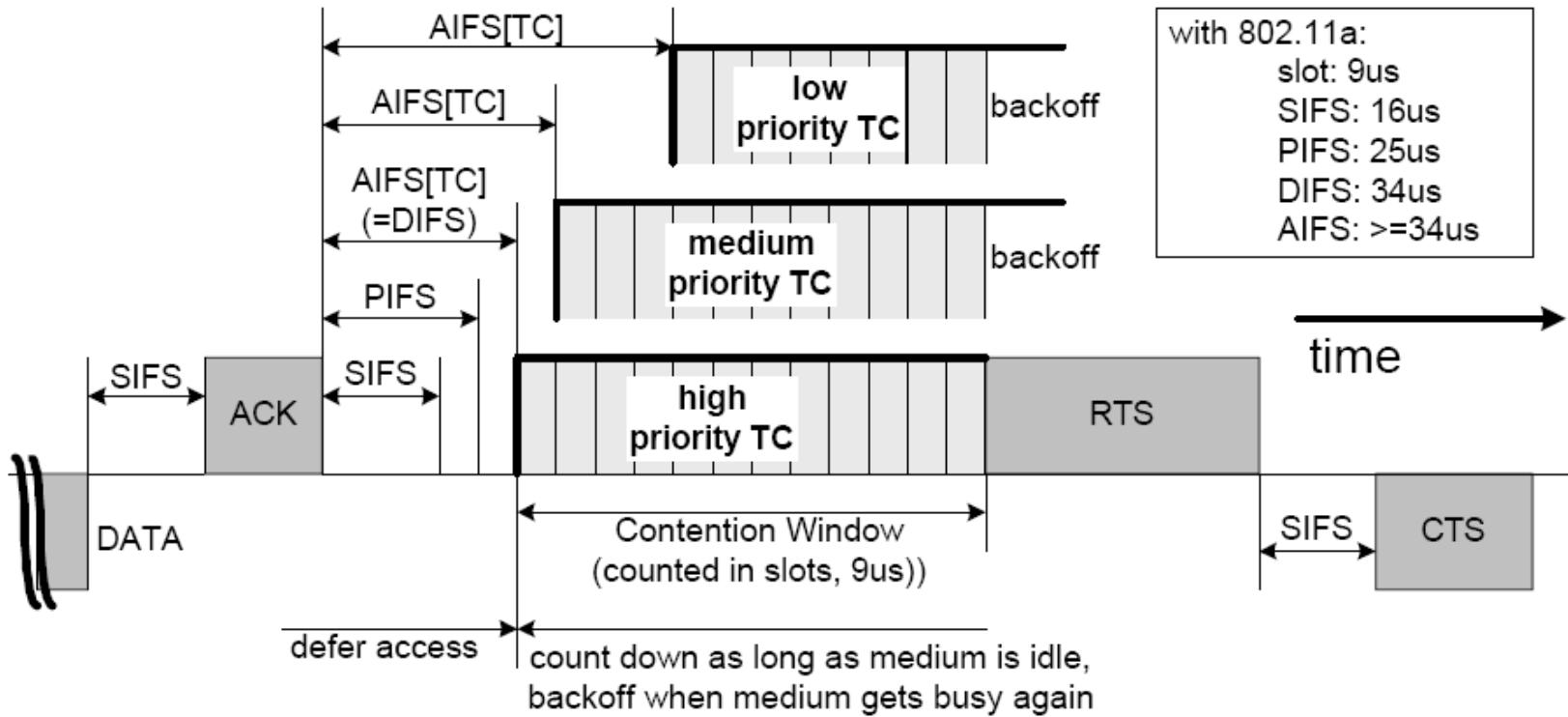


PCF

- Time= Contention Period + Contention Free Period
 - Contention Period: All stations compete for the medium
 - Contention Free Period: The AP coordinates communication

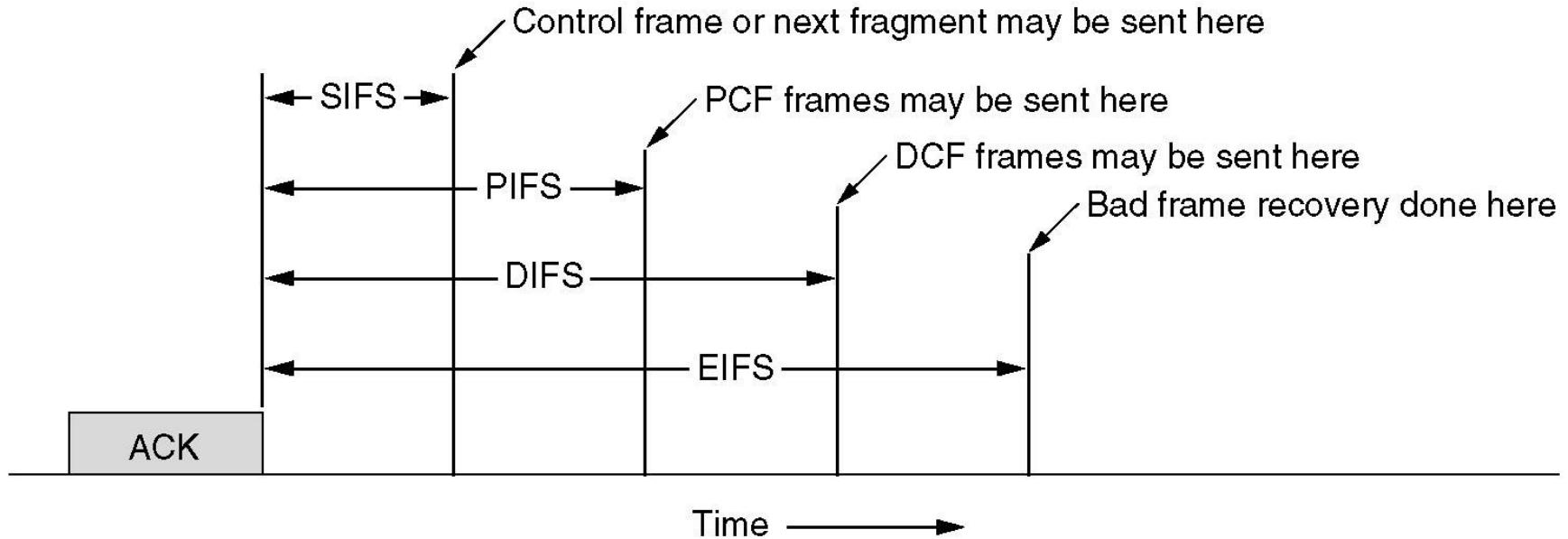


Example of IFSs for 802.11a



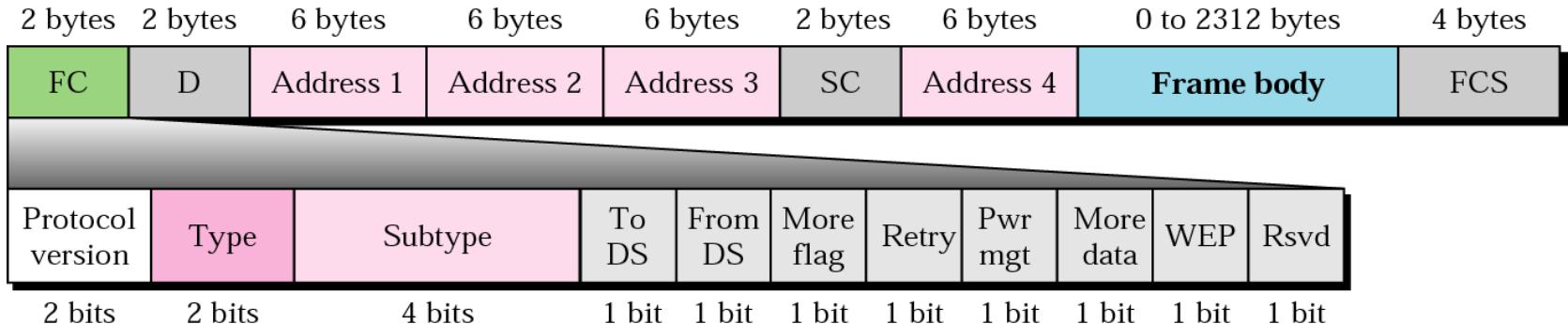
* Figure is courtesy of W. Stallings

IFS in 802.11



- SIFS influences replies
- PIFS gives PCF priority over DCF
- DIFS is the time between two DCF communications

Frame Format



Control Frames

Type:management (00), control (01), or data (10).

Subtype	Meaning
1011	Request to send (RTS)
1100	Clear to send (CTS)
1101	Acknowledgment (ACK)

2 bytes 2 bytes 6 bytes 6 bytes 4 bytes



RTS

2 bytes 2 bytes 6 bytes 4 bytes



CTS or ACK

* Figure is courtesy of B. Forouzan

IEEE 802.11 MAC Frame Format

- Frame format as described in the standard:

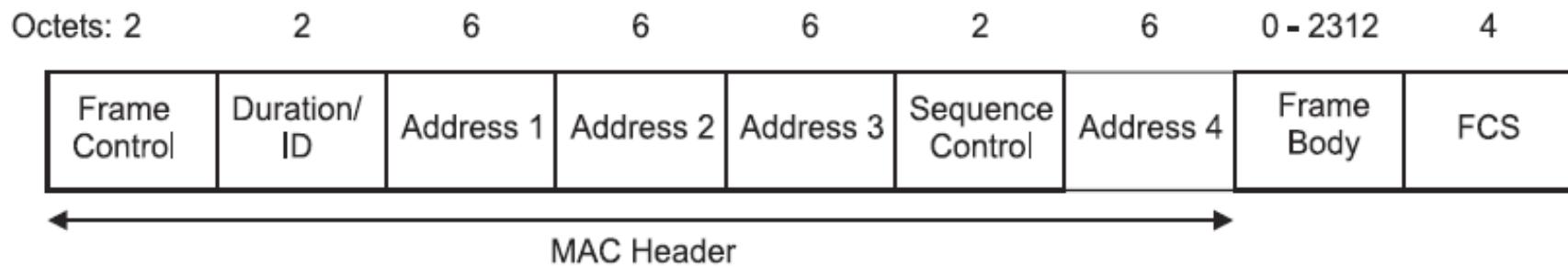


Figure 12—MAC frame format

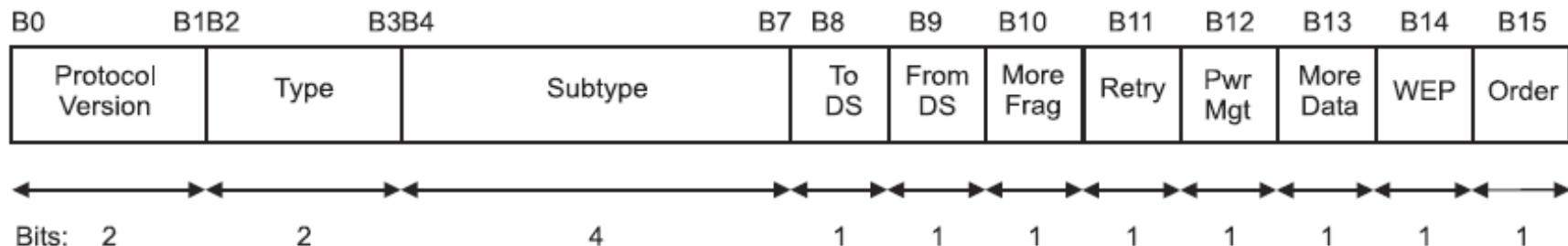


Figure 13—Frame Control field

* Figure is courtesy of ANSI/IEEE Std 802.11

Types of Frames

Type value b3 b2	Type description	Subtype value b7 b6 b5 b4	Subtype description
00	Management	0000	Association request
00	Management	0001	Association response
00	Management	0010	Reassociation request
00	Management	0011	Reassociation response
00	Management	0100	Probe request
00	Management	0101	Probe response
00	Management	0110–0111	Reserved
00	Management	1000	Beacon
00	Management	1001	Announcement traffic indication message (ATIM)
00	Management	1010	Disassociation
00	Management	1011	Authentication
00	Management	1100	Deauthentication
00	Management	1101–1111	Reserved
01	Control	0000–1001	Reserved
01	Control	1010	Power Save (PS)-Poll

Types of Frames

01	Control	1011	Request To Send (RTS)
01	Control	1100	Clear To Send (CTS)
01	Control	1101	Acknowledgment (ACK)
01	Control	1110	Contention-Free (CF)-End
01	Control	1111	CF-End + CF-Ack
10	Data	0000	Data
10	Data	0001	Data + CF-Ack
10	Data	0010	Data + CF-Poll
10	Data	0011	Data + CF-Ack + CF-Poll
10	Data	0100	Null function (no data)
10	Data	0101	CF-Ack (no data)
10	Data	0110	CF-Poll (no data)
10	Data	0111	CF-Ack + CF-Poll (no data)
10	Data	1000–1111	Reserved
11	Reserved	0000–1111	Reserved



Control Frames

- Assist in reliable data delivery
- Power Save-Poll (PS-Poll)
 - Sent by any station to station that includes AP
 - Request AP transmit frame buffered for this station while station in power-saving mode
- Request to Send (RTS)
 - First frame in four-way frame exchange
- Clear to Send (CTS)
 - Second frame in four-way exchange
- Acknowledgment (ACK)
- Contention-Free (CF)-end
 - Announces end of contention-free period part of PCF
- CF-End + CF-Ack:
 - Acknowledges CF-end
 - Ends contention-free period and releases stations from associated restrictions



Data Frames – Data Carrying

- Eight data frame subtypes, in two groups
- First four carry upper-level data from source station to destination station
- **Data**
 - Simplest data frame
 - May be used in contention or contention-free period
- **Data + CF-Ack**
 - Only sent during contention-free period
 - Carries data and acknowledges previously received data
- **Data + CF-Poll**
 - Used by point coordinator to deliver data
 - Also to request station send data frame it may have buffered
- **Data + CF-Ack + CF-Poll**
 - Combines Data + CF-Ack and Data + CF-Poll

Management Frames

- Used to manage communications between stations and APs
- E.g. management of associations
 - Requests, response, reassociation, dissociation, and authentication



Summary: 802.11

- Hidden Station / Expose Station
- DCF \Rightarrow Distributed Coordination Function
 - Stations compete for access to the medium
- CSMA/CA + RTS/CTS
- PCF \Rightarrow Point Coordination Function
 - Access point polls stations
- IFS \Rightarrow Inter-Frame Space
 - Time between frames
- Three types of frames:
 - Control frames
 - Data frames
 - Management frames







CS2031

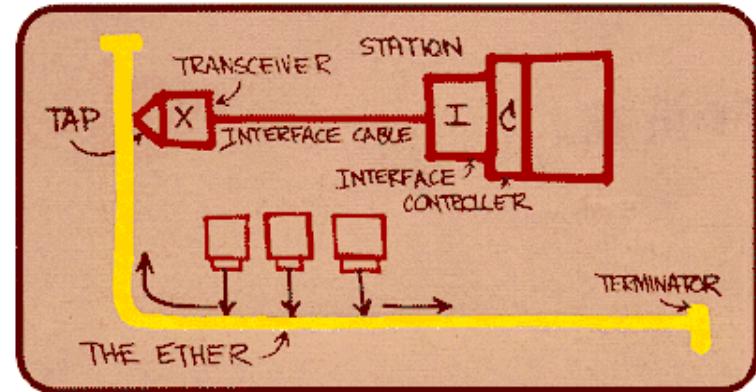
Telecommunications II

Ethernet



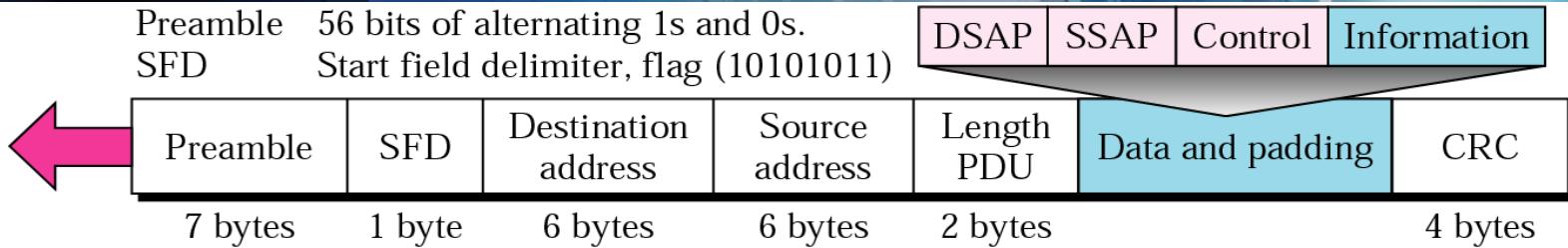
Ethernet

- Developed by Metcalfe 1972/3
- Standards in 1978, 1995, 1998
- Types of Ethernet
 - Original Ethernet
 - Switched Ethernet
 - Fast Ethernet
 - Gigabit Ethernet
- Manchester Encoding
- Medium Access Control
 - CSMA/CD



Metcalfe's Ethernet sketch

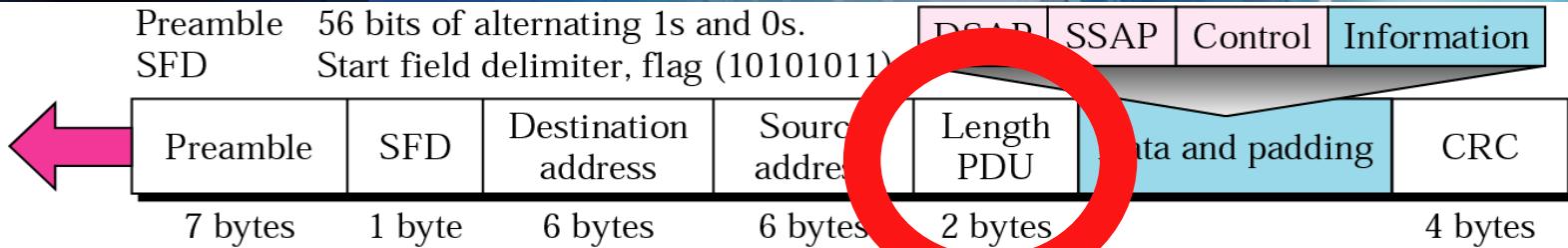
802.3 MAC Format



- 64-bit frame preamble (10101010) used to synchronize reception
 - 7 bit preamble (10101010) + 1 start flag (10101011)
- Maximum frame length: 1536 bytes
 - ⇒ max 1500 bytes payload
- Minimum frame length: 64 bytes
 - ⇒ min 46 bytes payload

* Figure is courtesy of B. Forouzan

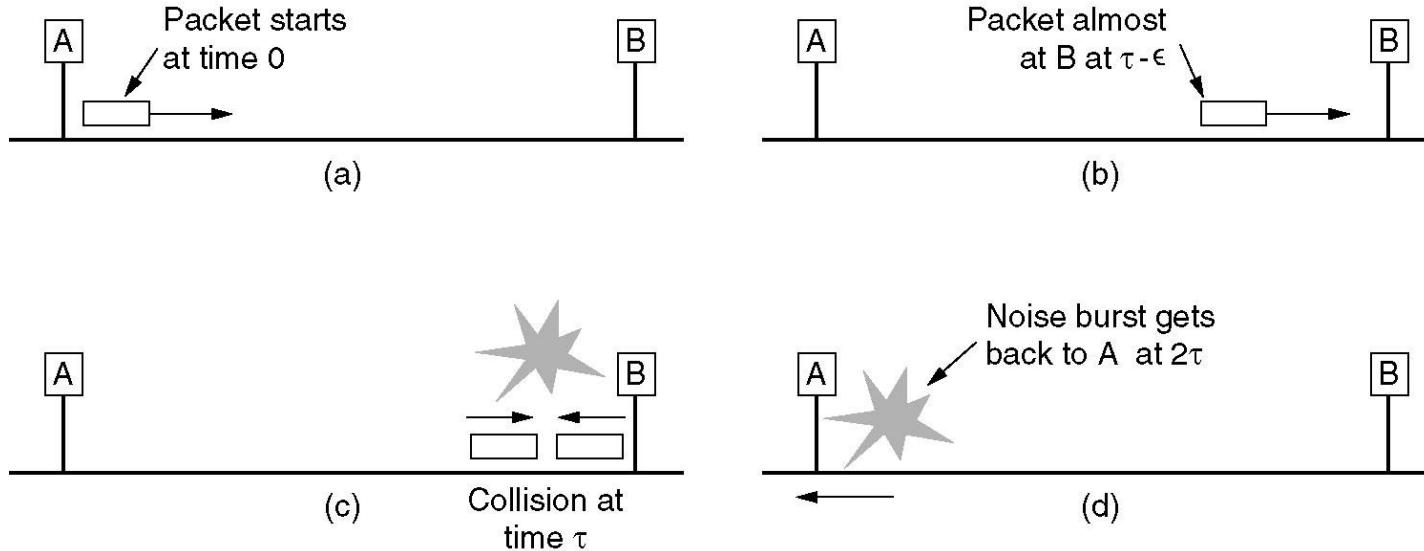
802.3 MAC Format



- 64-bit frame preamble (10101010) used to synchronize reception
 - 7 bit preamble (10101010) + 1 start flag (10101011)
- Maximum frame length: 1536 bytes
 - ⇒ max 1500 bytes payload
- Minimum frame length: 64 bytes
 - ⇒ min 46 bytes payload

Length: Up to 0x600
Type: eg. 0x800 IP
0x806 ARP

Frame Length

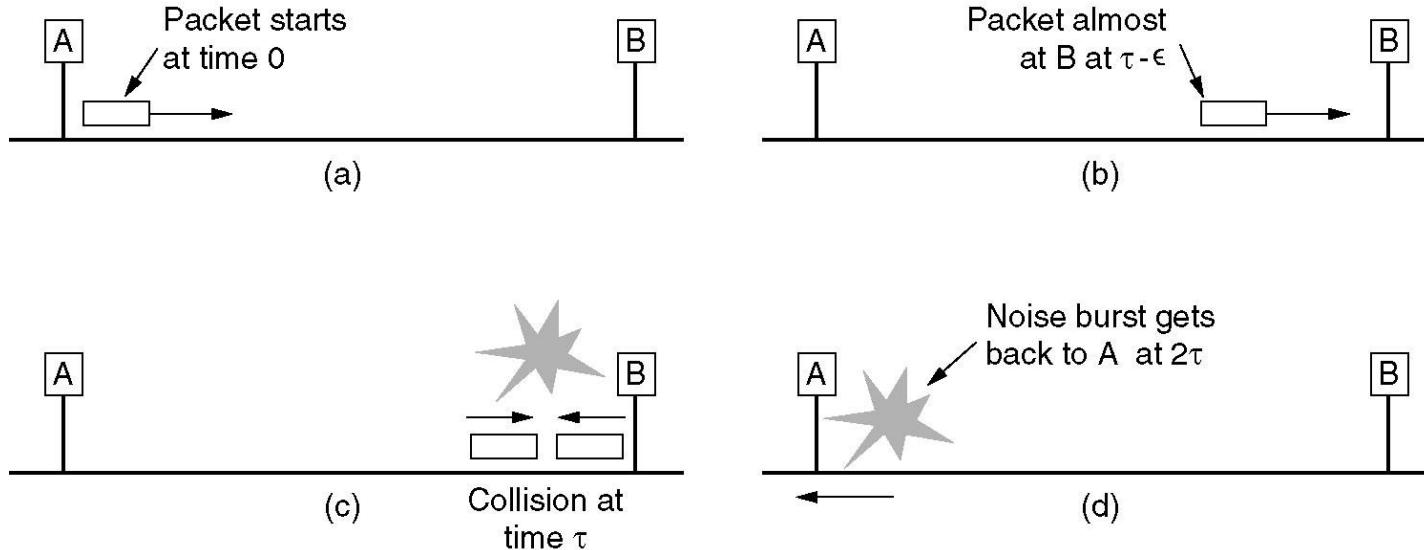


- Sender starts at $t= 0$
 - Packet takes τ time to get to B
 - Shortly before B starts transmitting
 - But discovers collision with A's signal
 - 48-bit Jamming signal takes τ time to get to B
- ⇒ It takes at 2τ to detect a collision

* Figure is courtesy of A. Tanenbaum



Frame Length II



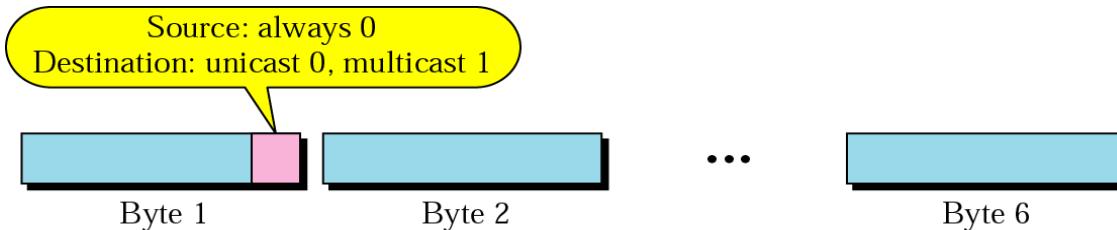
- It takes at 2τ to detect a collision
- Roundtrip time = 100μsec
- $10 \text{ Mbit/s} \Rightarrow 500 \text{ bits}$
~512 bits or 64 bytes

* Figure is courtesy of A. Tanenbaum



Ethernet Addresses

- Types of Addresses:
 - Unicast – delivered to one station
 - 00-10-4B 3Com 3C905-TX PCI
 - 00-A0-C9 Intel (PRO100B and PRO100+)
 - Multicast – delivered to a set of stations
 - 01-80-C2-00-00-00 Spanning tree (for bridges)
 - 03-00-00-00-00-01 NETBIOS
 - Broadcast – delivered to all stations
 - FF-FF-FF-FF-FF-FF



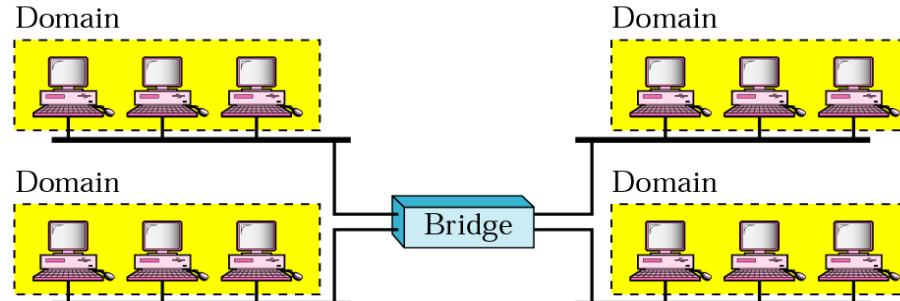
06-01-02-01-2C-4B
vendor-specific

* Figure is courtesy of B. Forouzan

Collision Domains



a. Without bridging

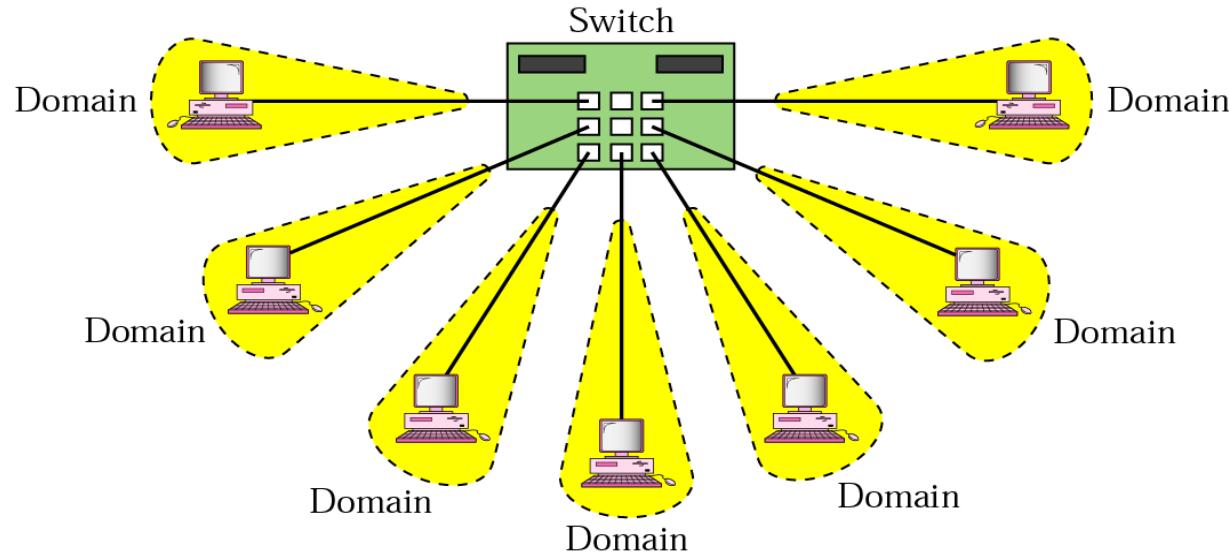


b. With bridging

- Extension of Networks:
 - Repeaters, Hubs - Physical Layer
 - Bridges, Switches - Data Link Layer
 - Routers - Network Layer
- Collision domains:
 - Collision affects all machines in one segment

* Figure is courtesy of B. Forouzan

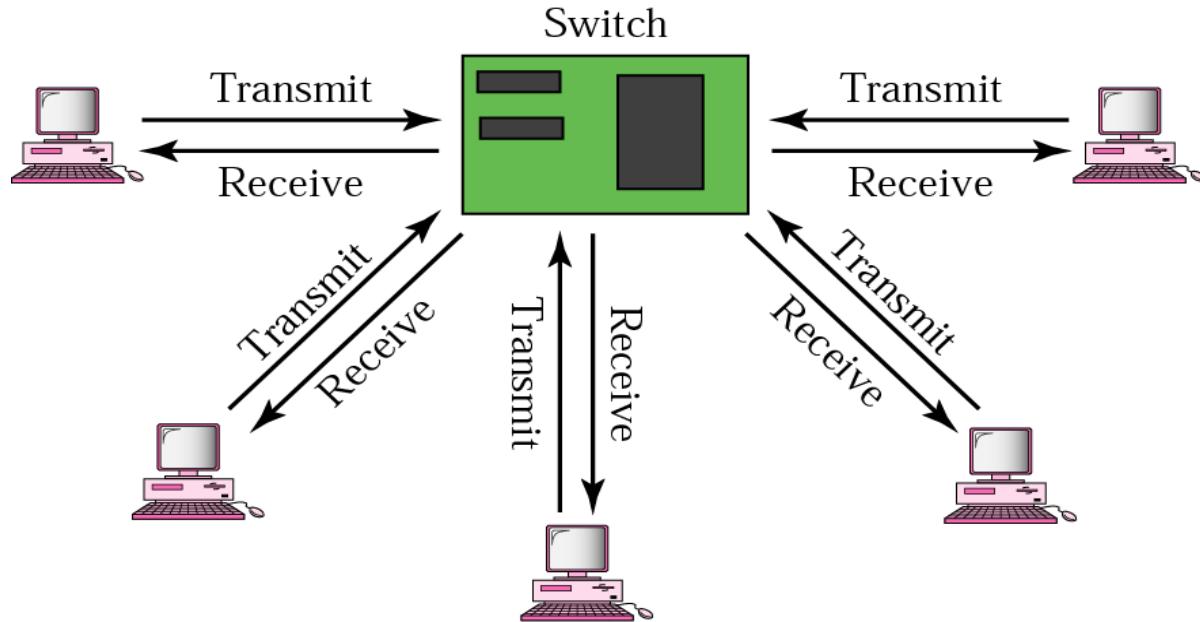
Switched Ethernet



- **Switch delivers packets to individual machines**
 - Without affecting communication with other machines
- **Collisions only occur on individual links**

* Figure is courtesy of B. Forouzan

Full-duplex Switched Ethernet

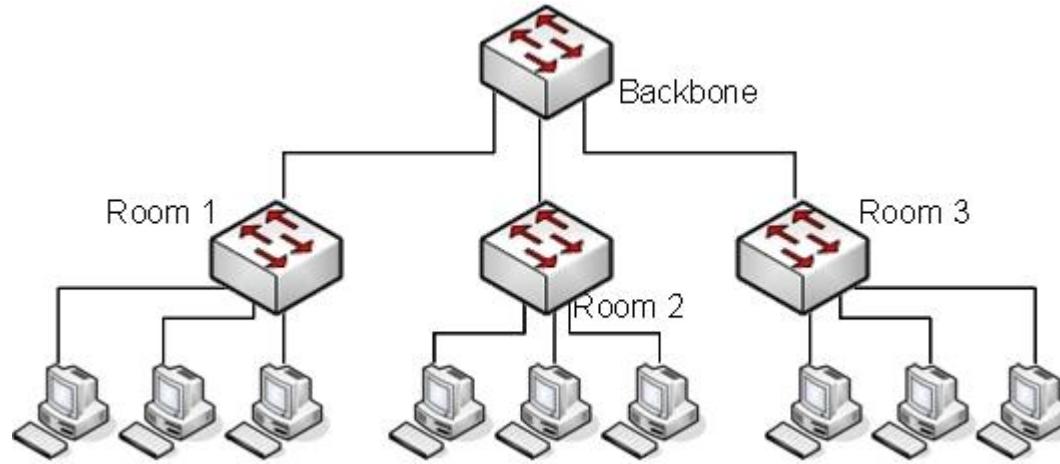


- No collisions
 - One channel to send
 - One channel to transmit

* Figure is courtesy of B. Forouzan

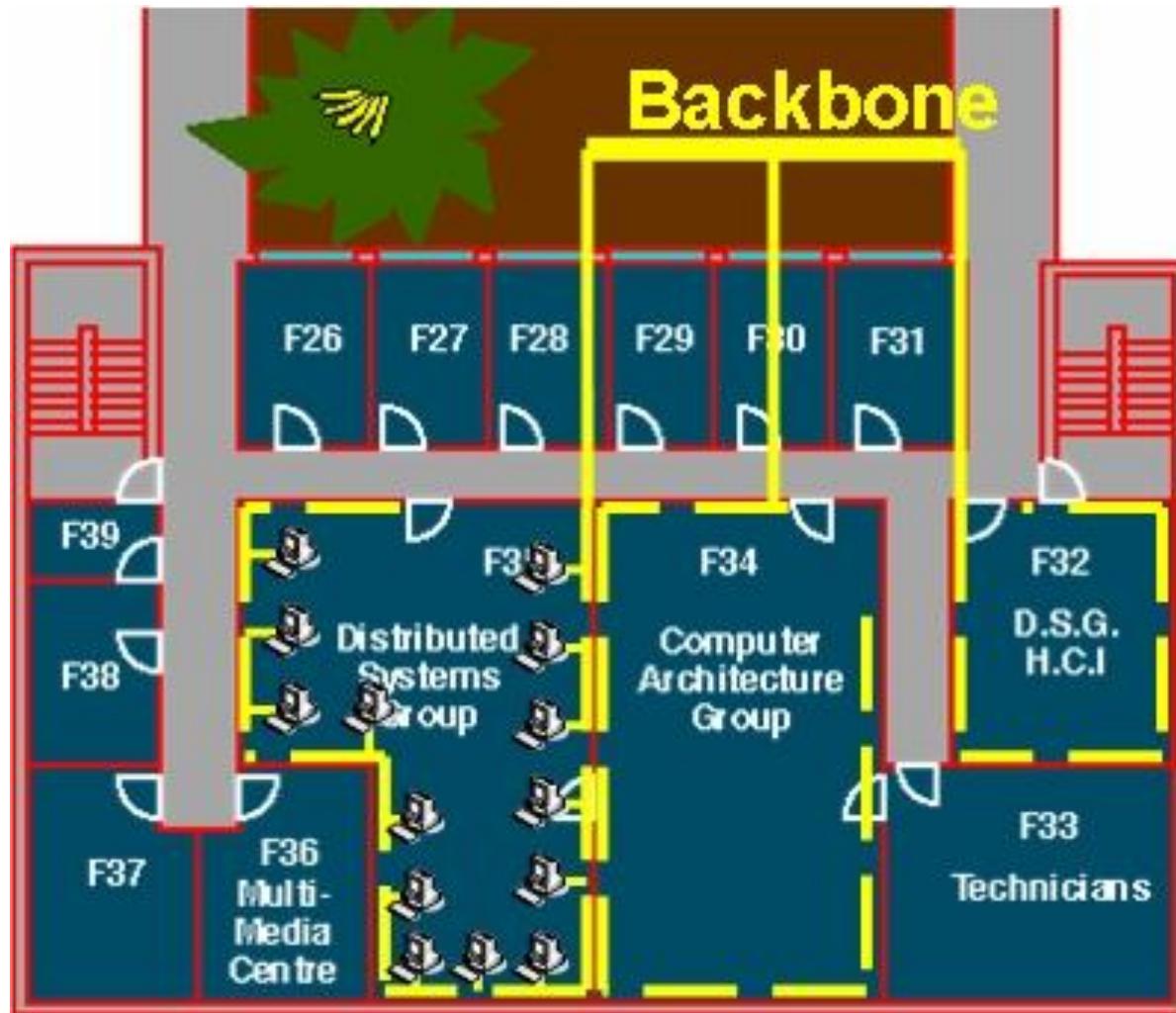


Switched Networks

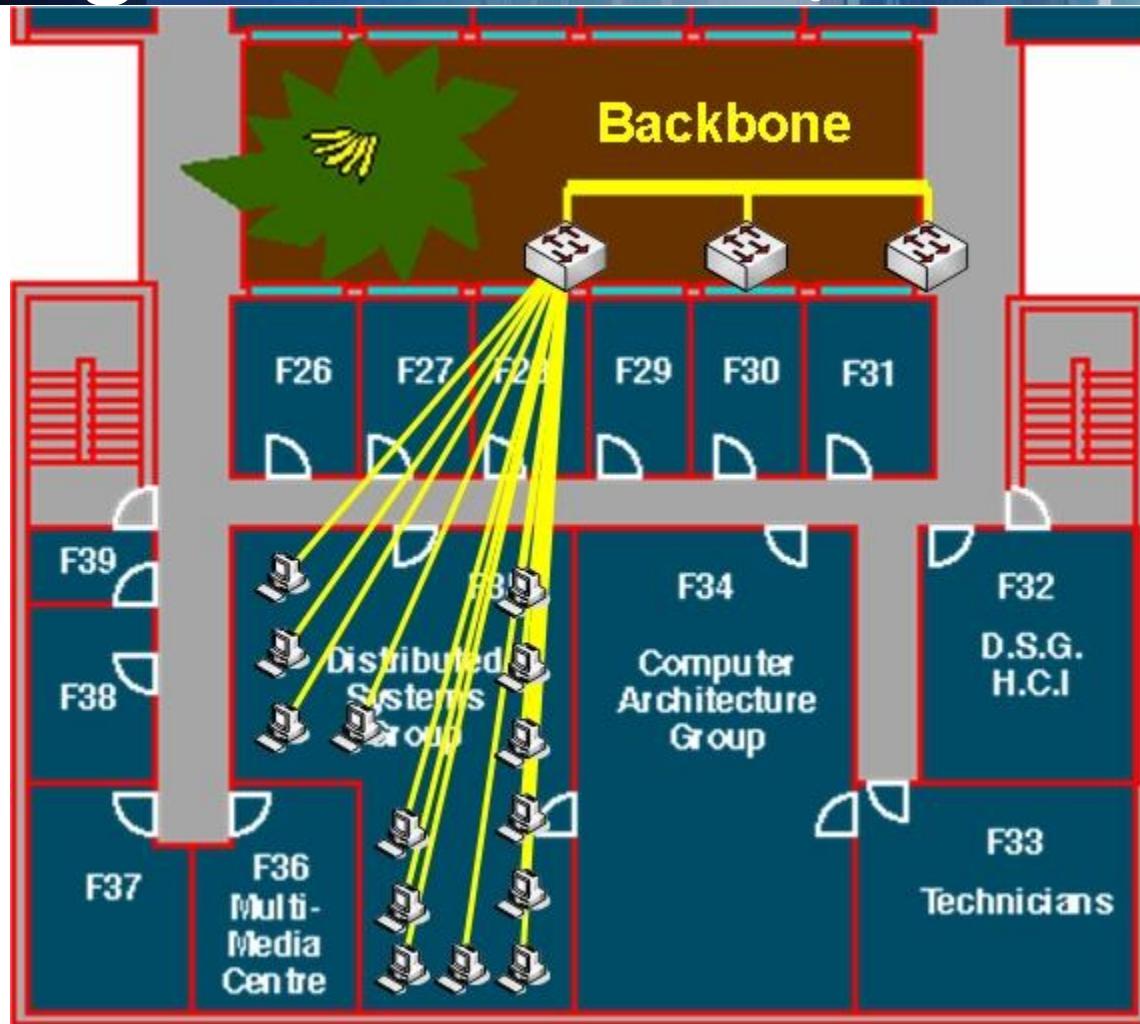


- Hierarchical Organization
- Separation into Segments
- Keep traffic in one segment - if possible

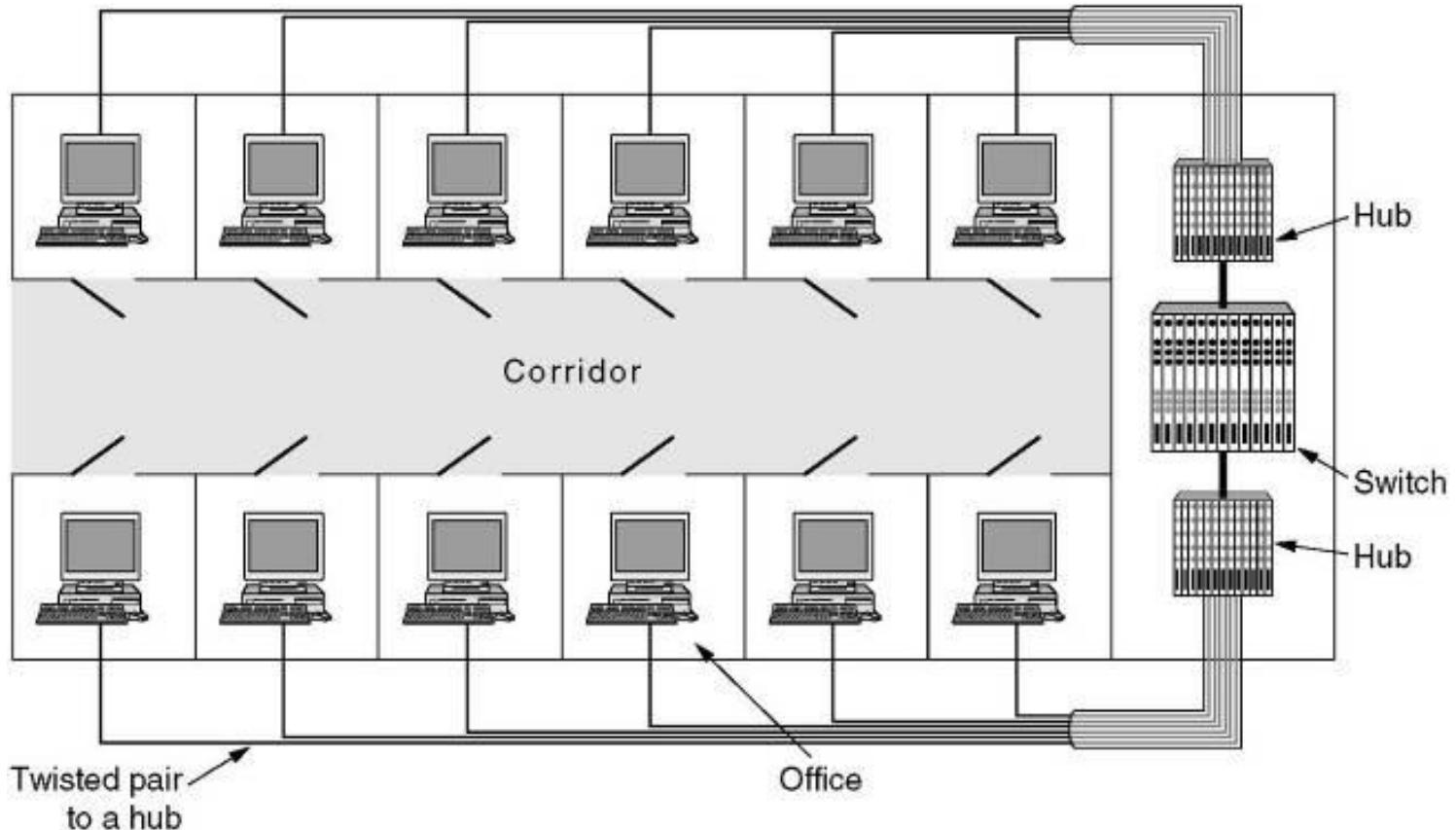
College Network – 10Base2



College Network–10/100BaseT

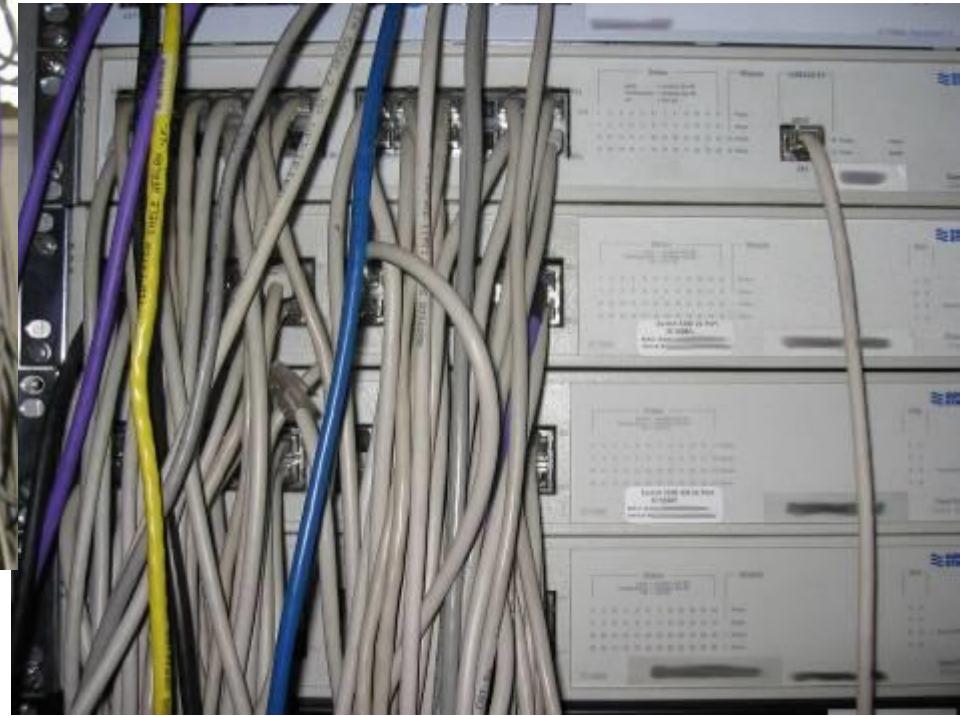
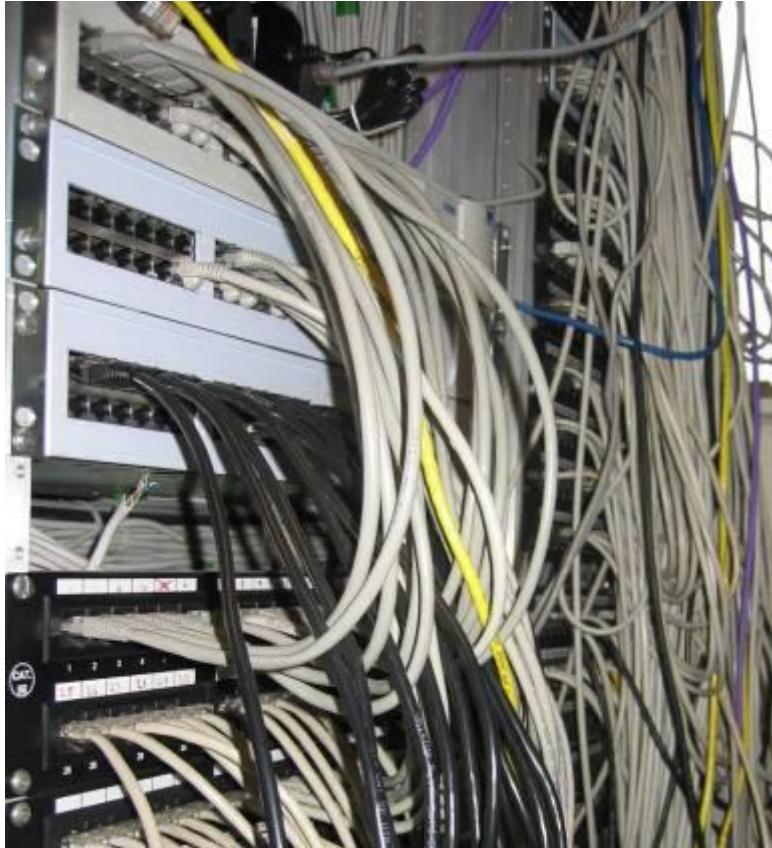


Switched Network

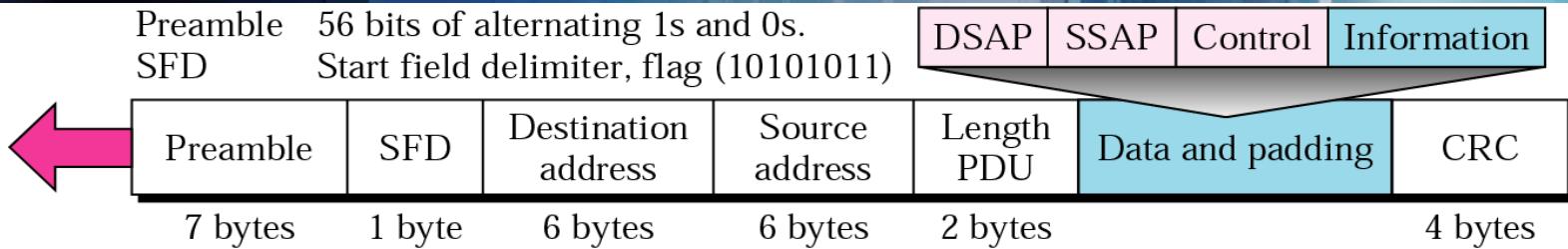




Switches in Comms Rooms



802.3 & 802.2 MAC Format



- 64-bit frame preamble (10101010) used to synchronize reception
 - 7 bit preamble (10101010) + 1 start flag (10101011)
- Maximum frame length: 1536 bytes
 - ⇒ max 1500 bytes payload
- Minimum frame length: 64 bytes
 - ⇒ min 46 bytes payload

* Figure is courtesy of B. Forouzan

Summary: Ethernet

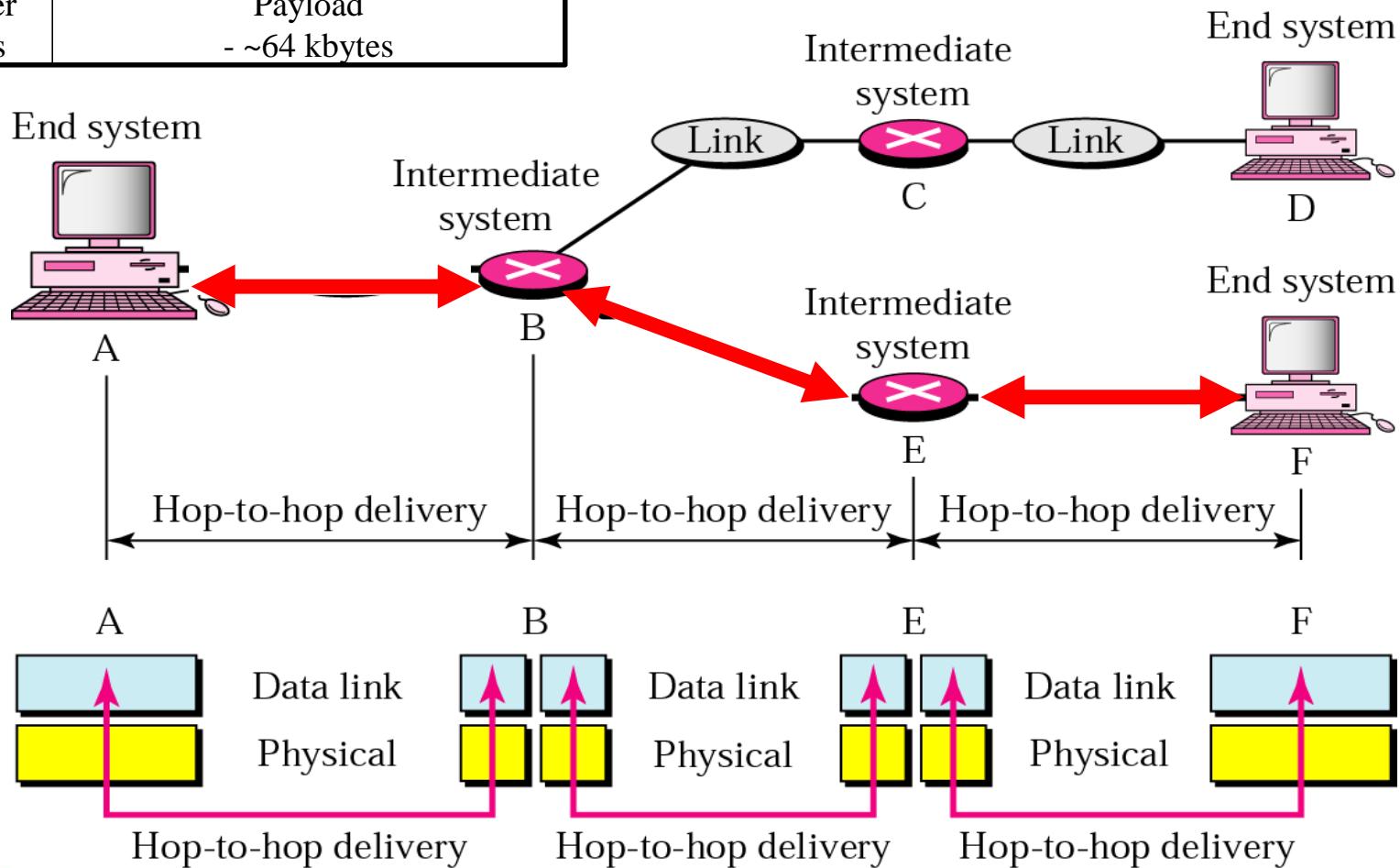
- Ethernet frame
 - Preamble to signal start of frame
 - MTU & minimum frame size
 - Addressing
- CSMA/CD
- Collision Domains
- Switched Networks



Link Layer

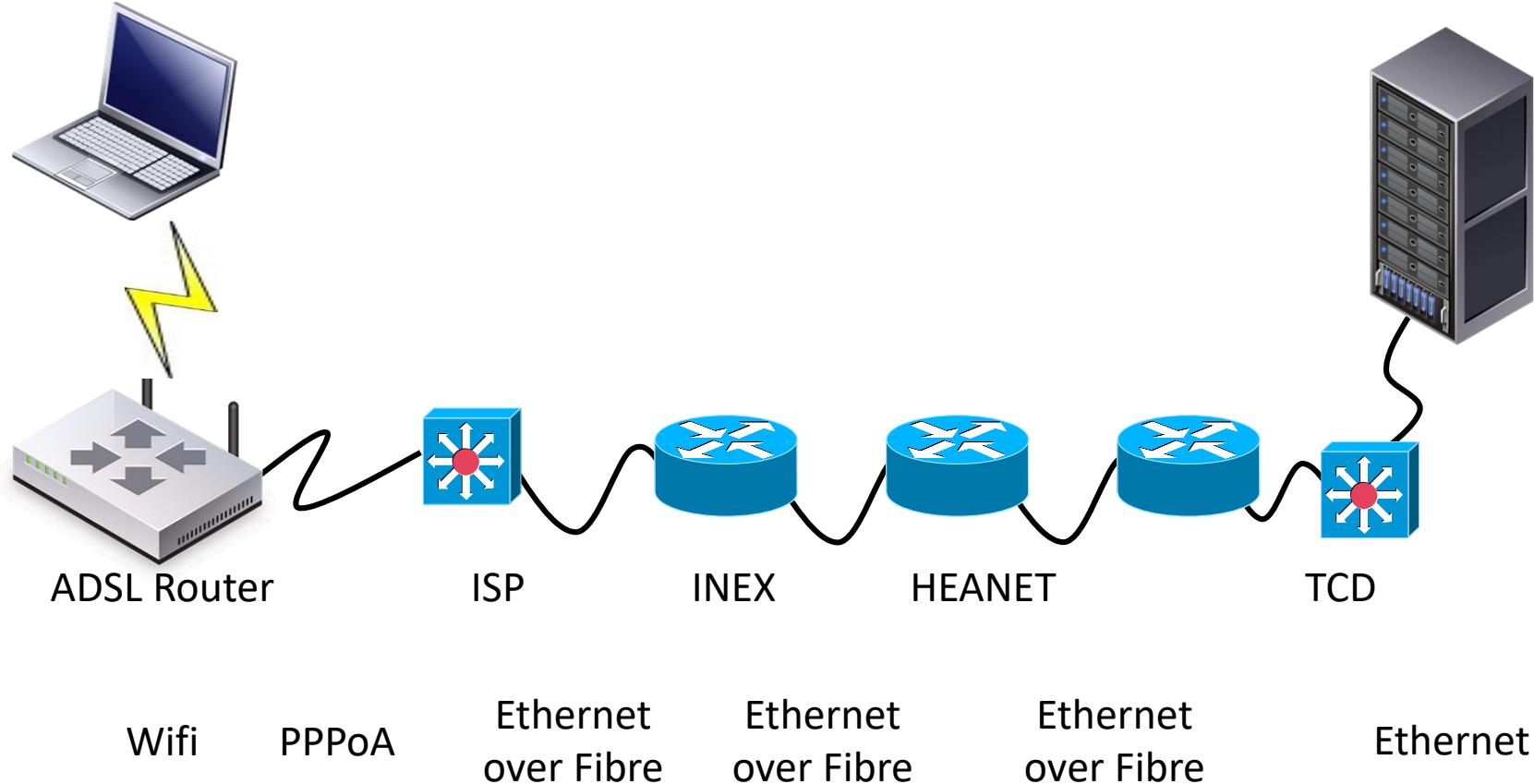
IP Packet

IP Header 20 bytes	Payload ~64 kbytes
-----------------------	-----------------------

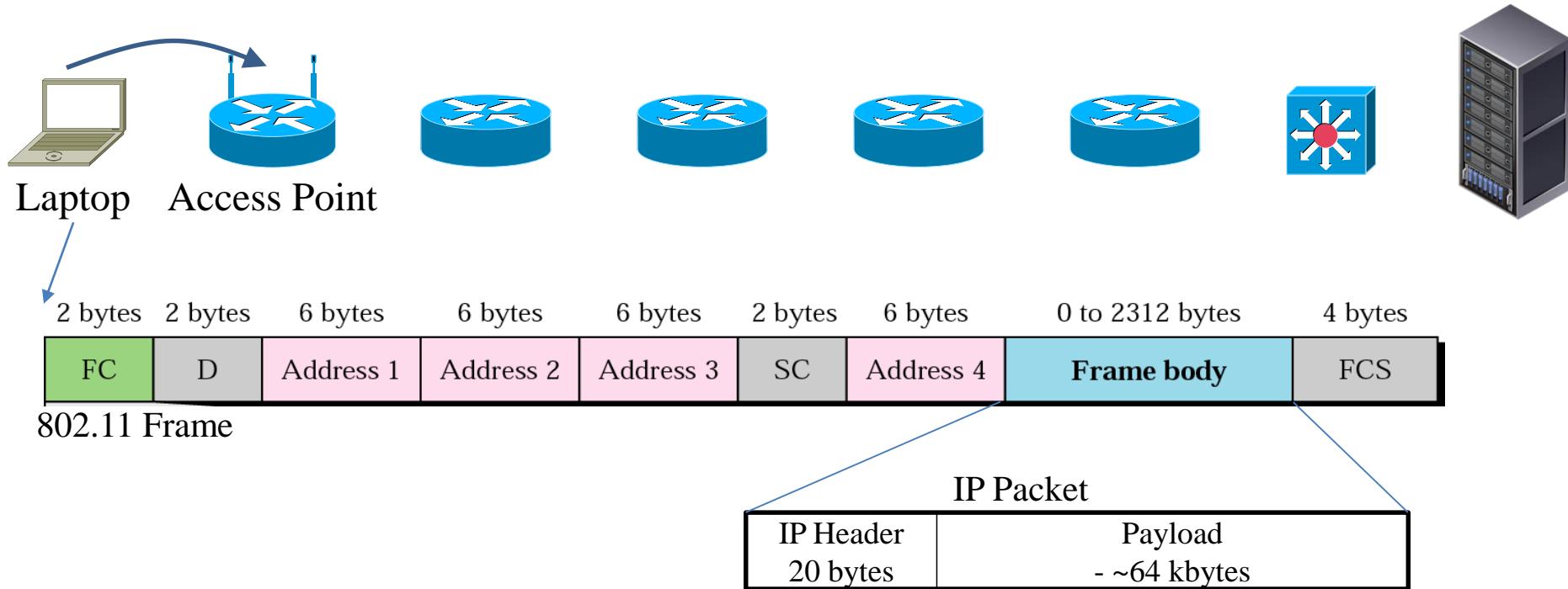


* Figure is courtesy of B. Forouzan

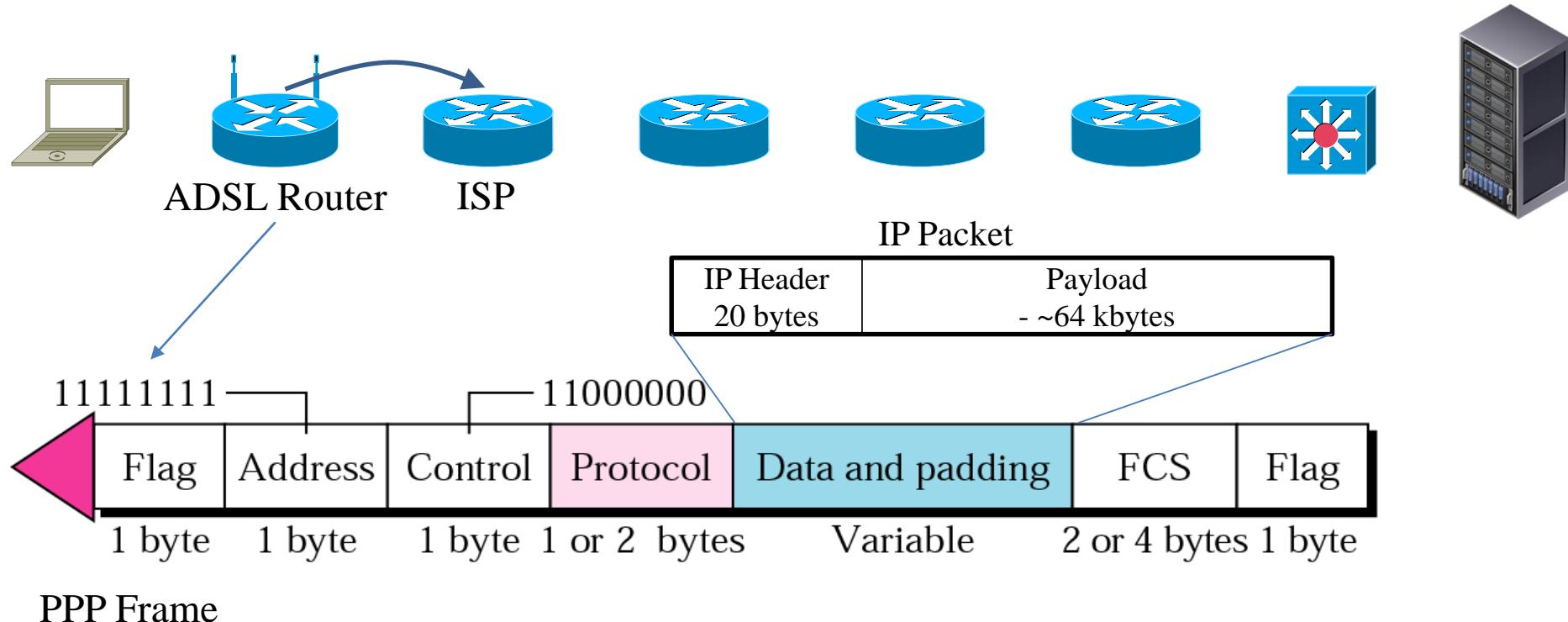
HTML Use Case



Wifi in Home Network

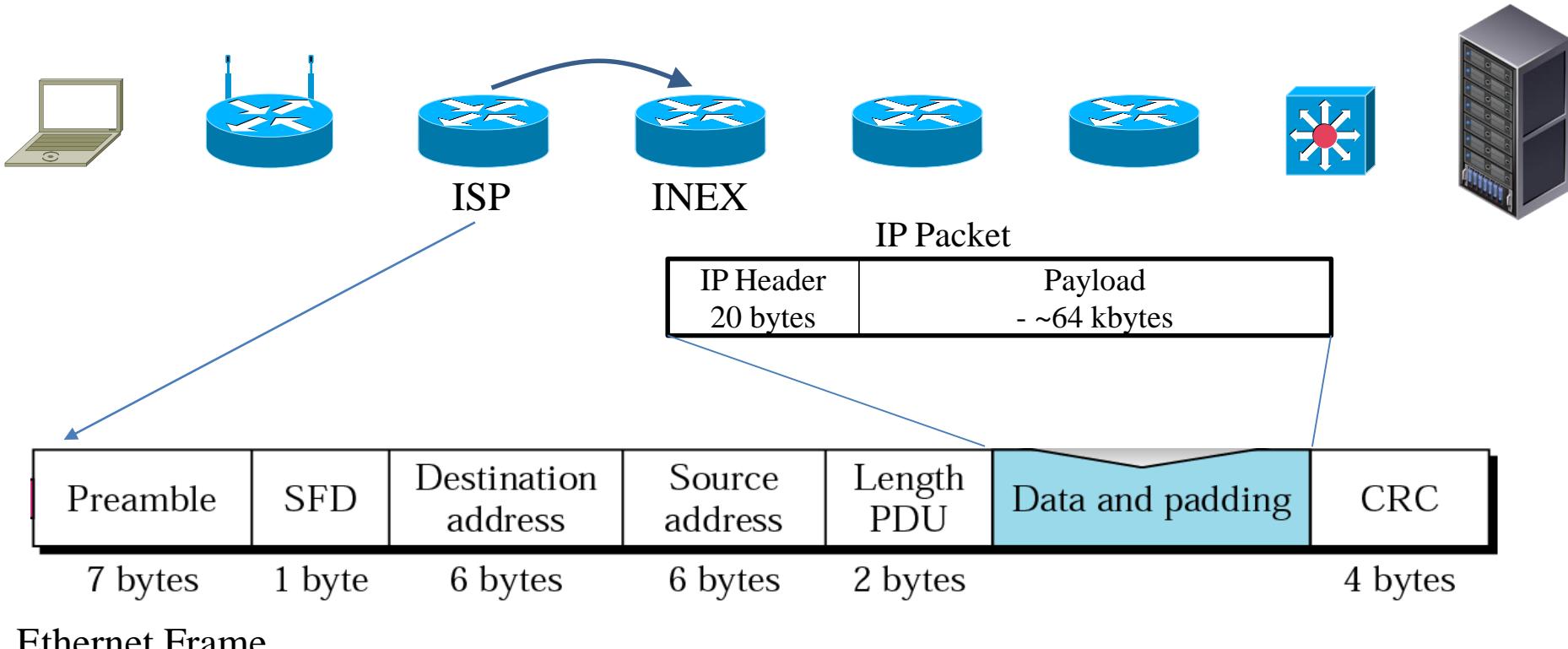


PPP to ISP



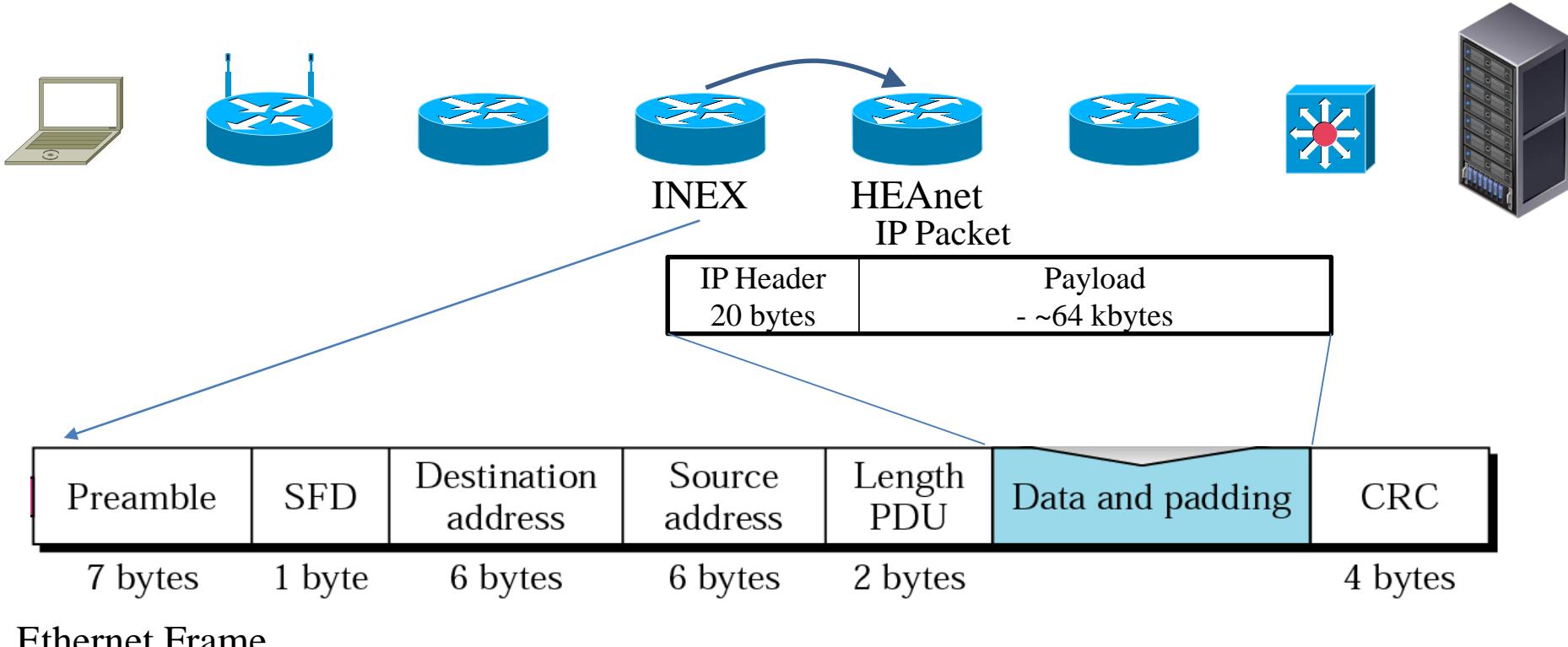
PPP Frame

Ethernet over Fibre

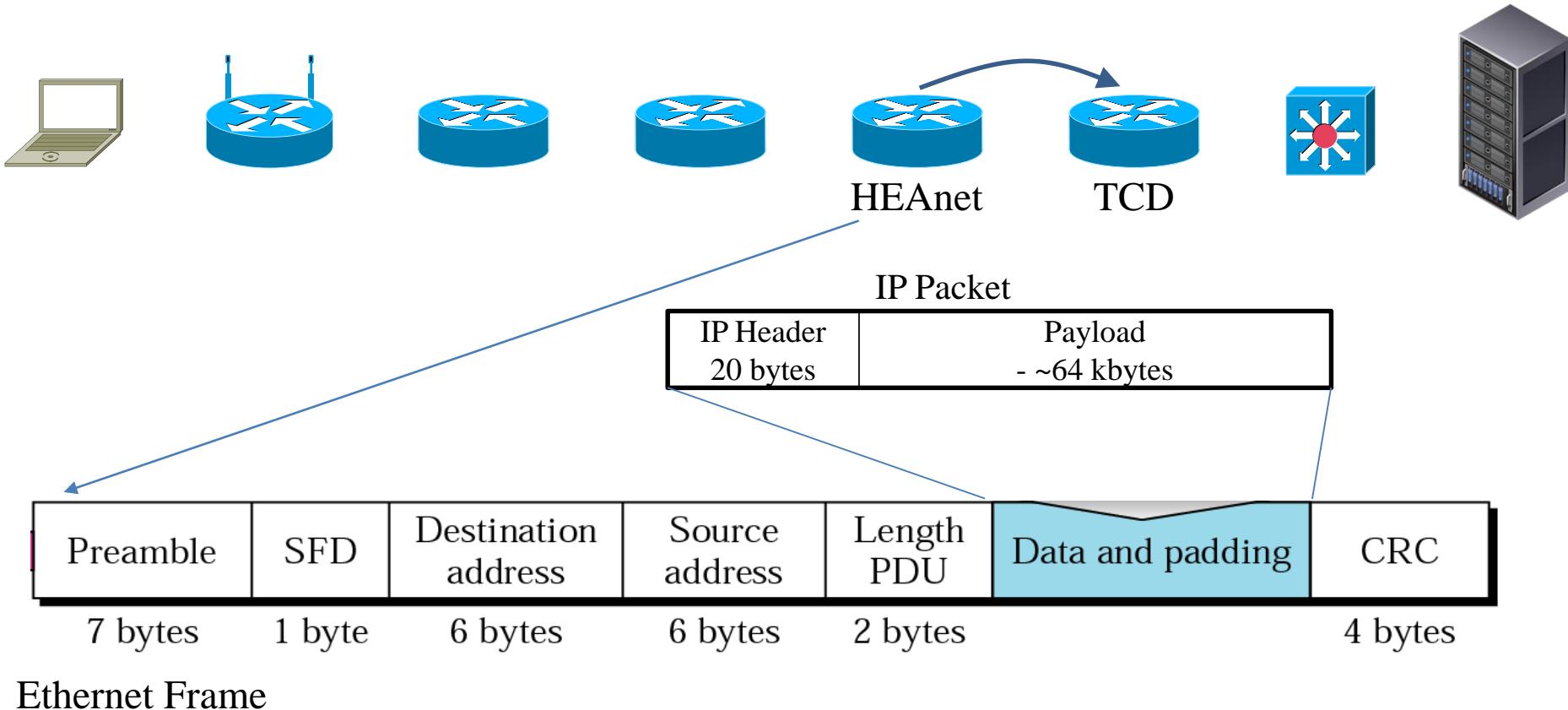


Ethernet Frame

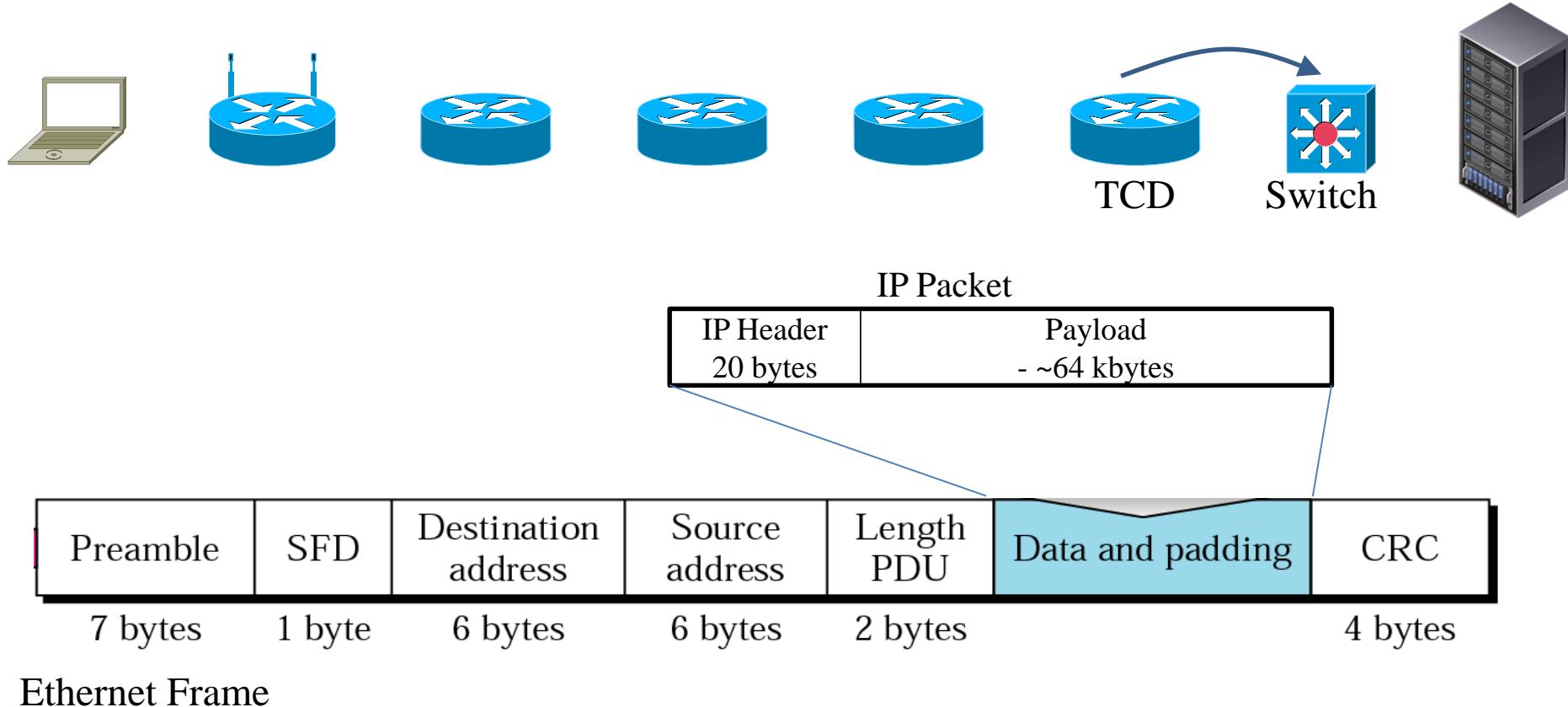
Ethernet over Fibre



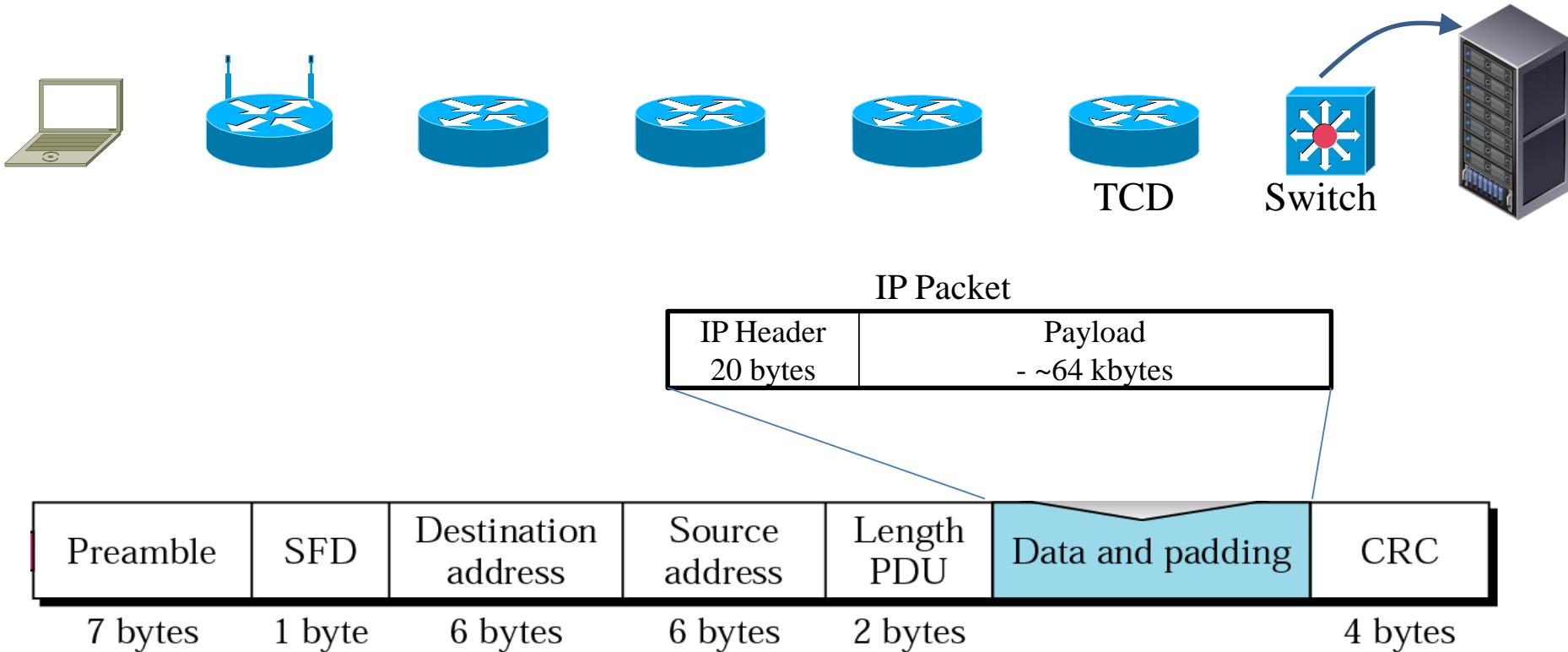
Ethernet over Fibre



Ethernet

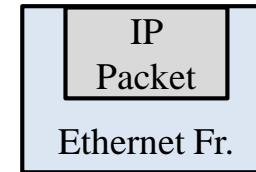
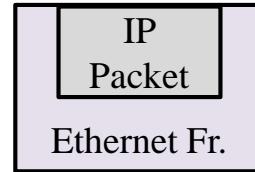
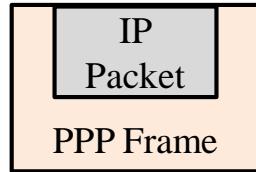
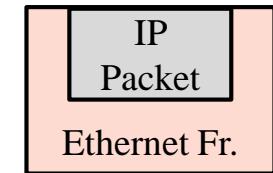
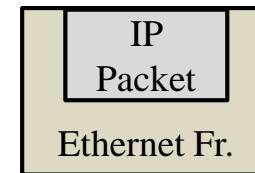
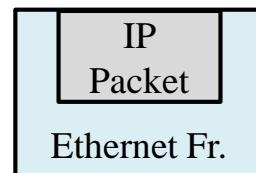
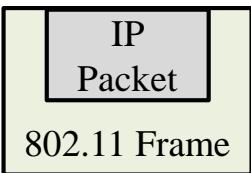
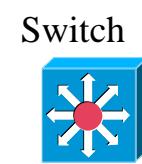
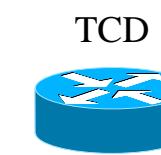


Ethernet

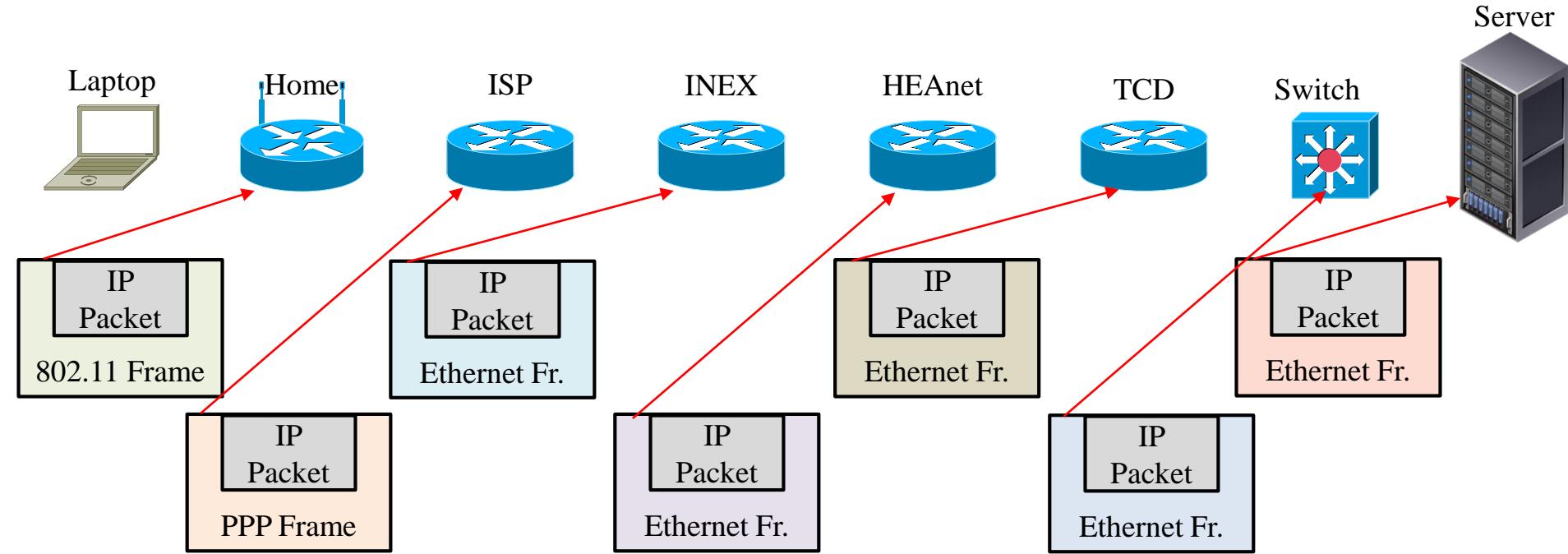


Ethernet Frame

Encapsulation



Addressing Next Hop



→ Addressed To



Assignment Discussion

- Sample Code
- Reports



File Input

```
String fname;  
File file= null;  
FileInputStream fin= null;  
byte[] buffer= null;  
int size;
```

```
fname= terminal.readString("Name of file: ");
```

Read filename

```
file= new File(fname);  
buffer= new byte[(int) file.length()];  
fin= new FileInputStream(file);  
size= fin.read(buffer);
```

Open file

Reserve byte buffer

Initialize input stream

Read file content

Rest of File Input

```
if (size== -1) {  
    fin.close();  
    throw new Exception("Problem with File Access"); }  
  
terminal.println("File size: " + buffer.length);
```

```
fcontent= new FileInfoContent(fname, size);  
terminal.println("Sending packet w/ name & length");  
packet= fcontent.toDatagramPacket();  
packet.setSocketAddress(dstAddress);  
socket.send(packet);  
terminal.println("Packet sent");  
this.wait();
```

```
fin.close();
```

End file access



Byte[] copying

```
DatagramPacket packet= null;  
byte[] payload= null; byte[] header= null; byte[] buffer= null;
```

```
payload= (terminal.readString("String to send: ")).getBytes();
```

```
header= new byte[PacketContent.HEADERLENGTH];
```

```
buffer= new byte[header.length + payload.length];
```

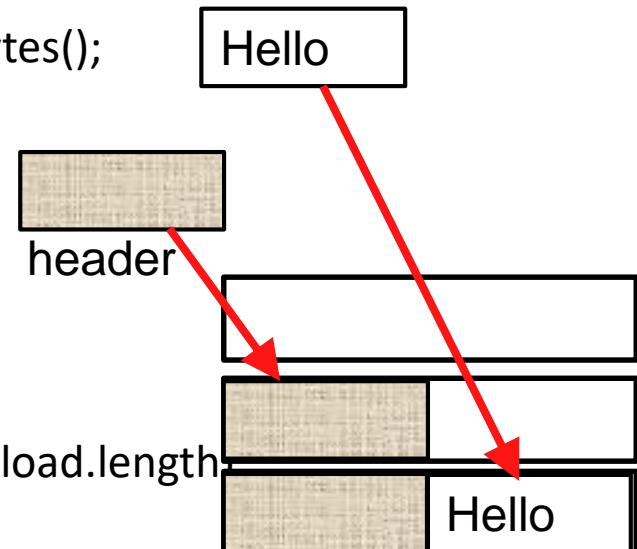
```
System.arraycopy(header, 0, buffer, 0, header.length);
```

```
System.arraycopy(payload, 0, buffer, header.length, payload.length);
```

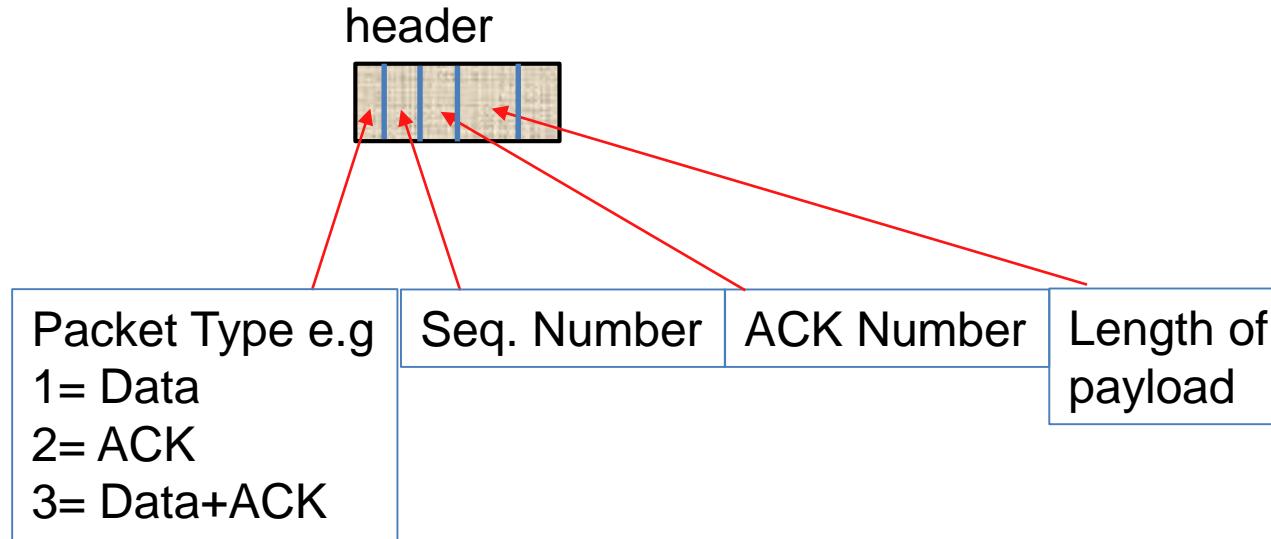
```
packet= new DatagramPacket(buffer, buffer.length, dstAddress);
```

```
socket.send(packet);
```

```
this.wait();
```



Example Header



tcd.lossy.DatagramSocket

```
package tcd.lossy;  
public class DatagramSocket extends java.net.DatagramSocket {  
    ...  
    public void send(DatagramPacket arg0) throws IOException {  
        if ((Math.random()*100) > noise) {  
            super.send(arg0);  
        }  
        else {  
            System.out.println("** Packet dropped");  
        }  
    }  
}
```

“import tcd.lossy.DatagramSocket;” instead of “import java.net.DatagramSocket;”

Reports

- Stop&Wait
 - Explanation of your understanding of the mechanism
 - Explanation of your code
 - Reflection on your implementation
- Go-Back-N/Selective Repeat
 - Explanation of your understanding of the mechanism
 - Explanation of your code
 - Reflection on your implementation
- Reflection on Flow Control/on your understanding of Flow Control





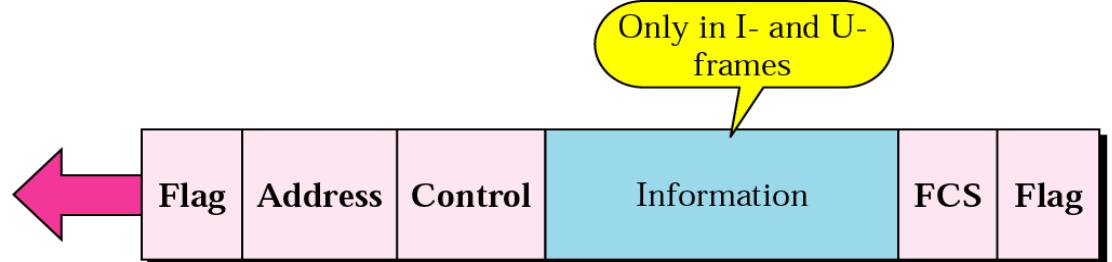
Spare Slides

The following slides were omitted from the lecture to reserve time to discuss the assignment



HDLC Frame

- Flag= 01111110
 - Specifies beginning and end of frame

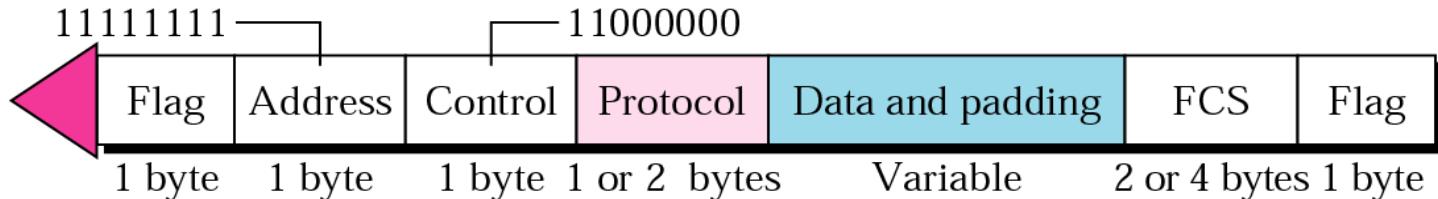


- Address
 - Specifies secondary station as either sender or receiver
- Control
 - Specifies type of frame and seq.&ack. number
- Frame Check Sequence (FCS)
 - Either 16- or 32-bit CRC

* Figure is courtesy of B. Forouzan

PPP Frame

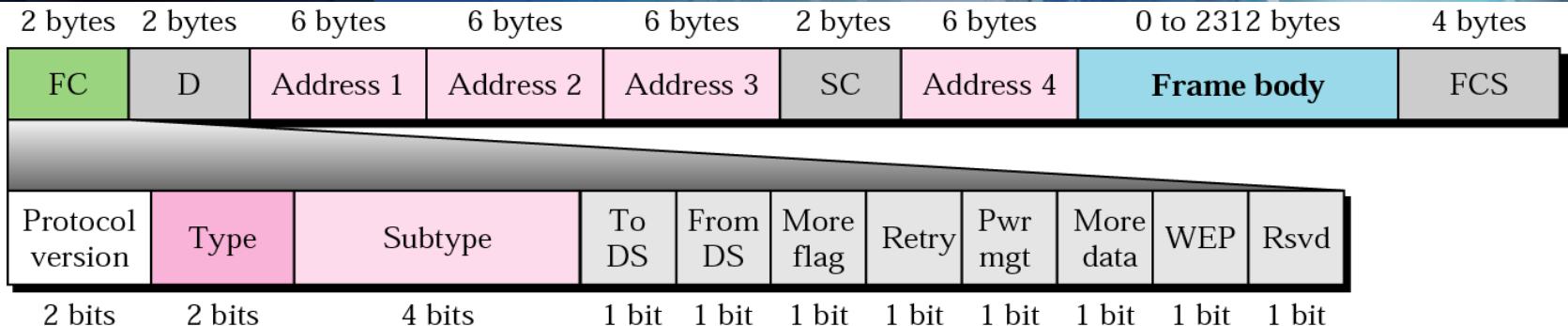
■ Modified HDLC frame:



- Byte-oriented Protocol
 - Flag Byte: 01111110
 - Escape Byte: 01111101
- FCS: 16- or 32-bit CRC
 - $x^{16} + x^{12} + x^5 + 1$
 - 1 0001 0000 0010 0001 → **16 bits remainder** ← **16-degree polynomial**
 - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

* Figure is courtesy of B. Forouzan

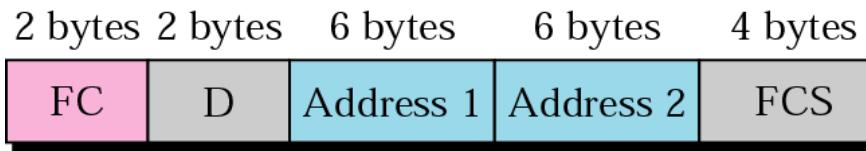
802.11 MAC Frame Format



Control Frames

Type:management (00), control (01), or data (10).

Subtype	Meaning
1011	Request to send (RTS)
1100	Clear to send (CTS)
1101	Acknowledgment (ACK)



RTS



CTS or ACK

* Figure is courtesy of B. Forouzan

802.11 Frames

- DSSS PLCP frame format

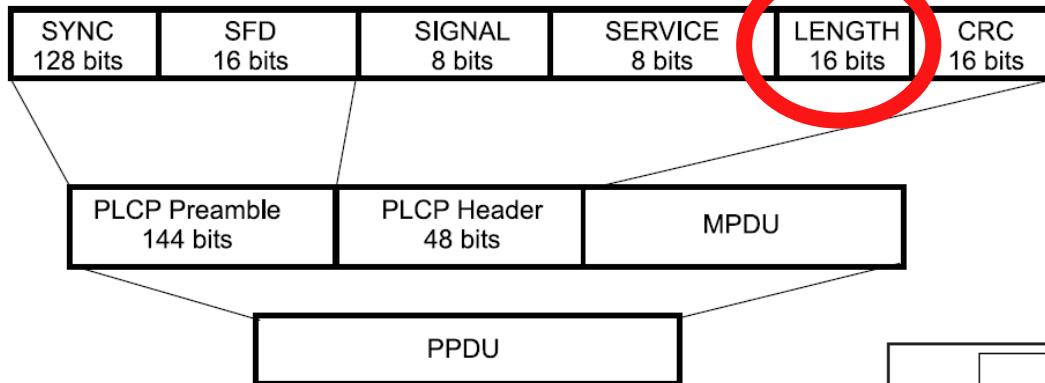


Figure 86—PLCP frame format

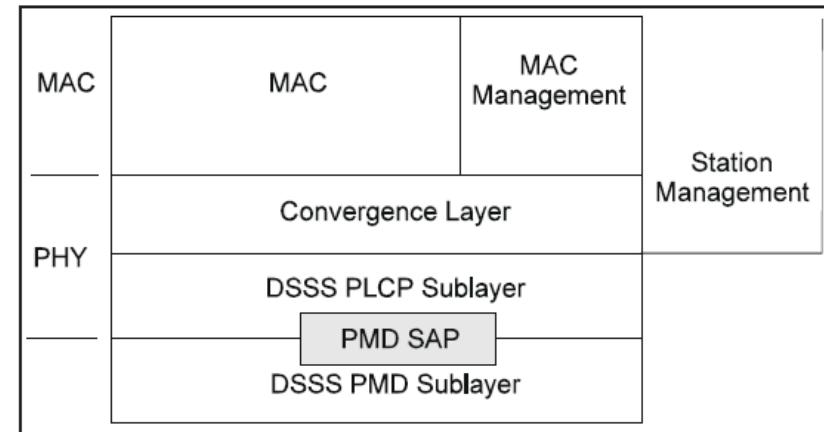
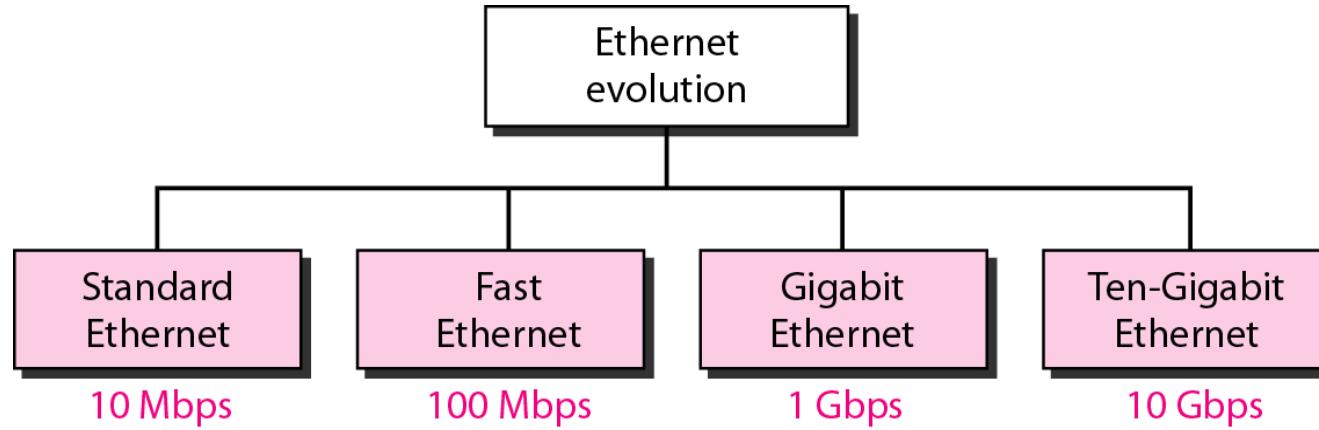


Figure 95—PMD layer reference model

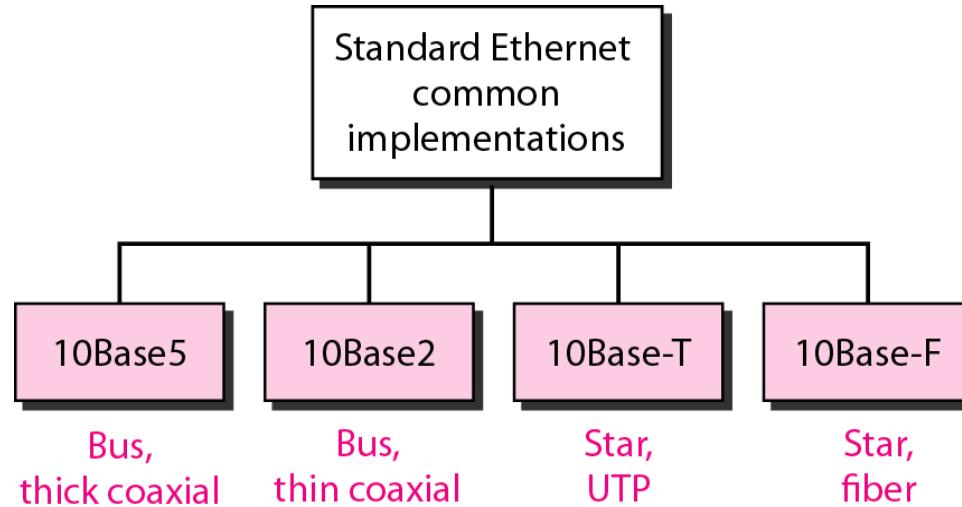
Evolution of Ethernet



- 1972/73 defined for coaxial cable
- Fast Ethernet used mainly unshielded twisted pair (UTP)
- Gigabit Ethernet common in desktops and laptops
- 10GB Ethernet used mainly for backbone

* Figure is courtesy of B. Forouzan

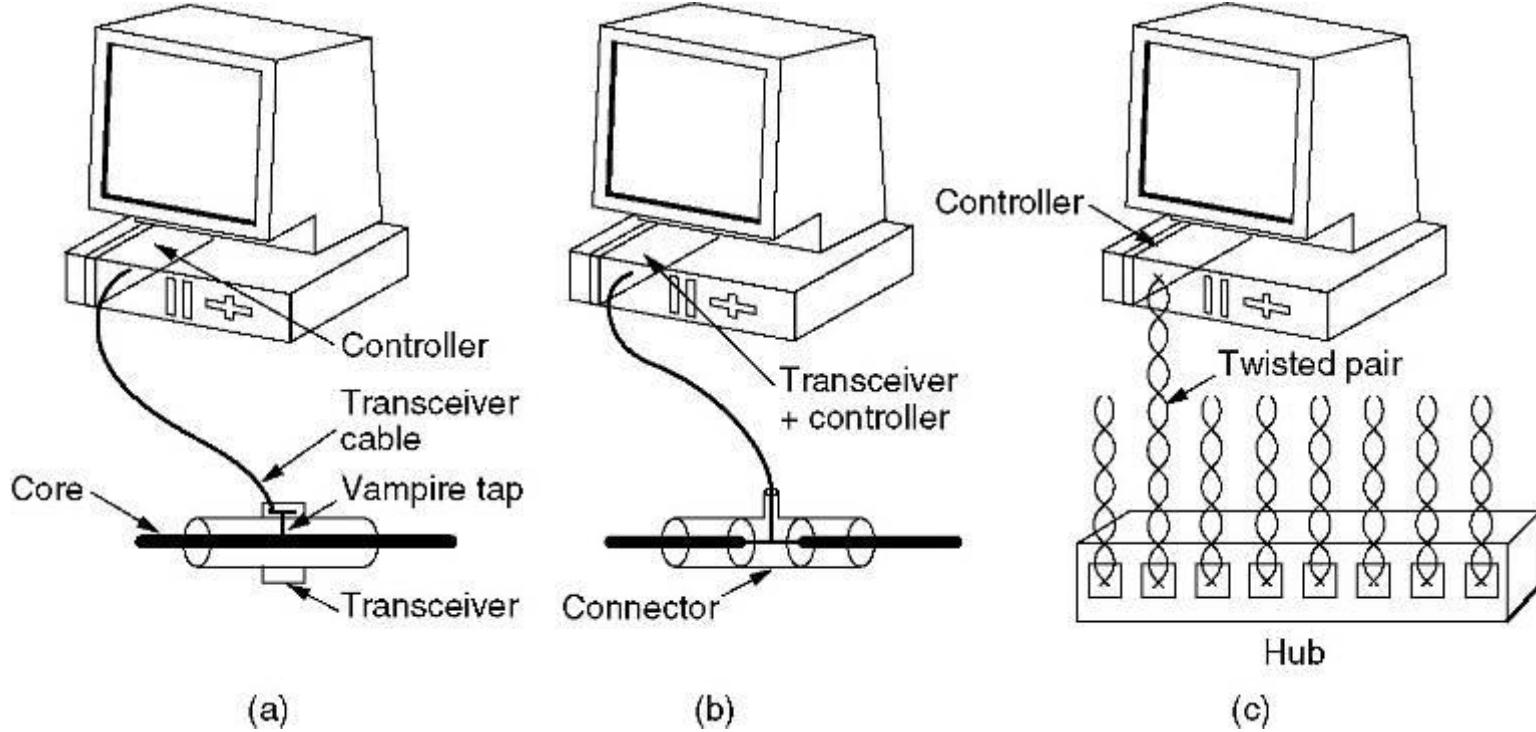
Types of Ethernet



Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

* Figure is courtesy of B. Forouzan

Ethernet Cabling



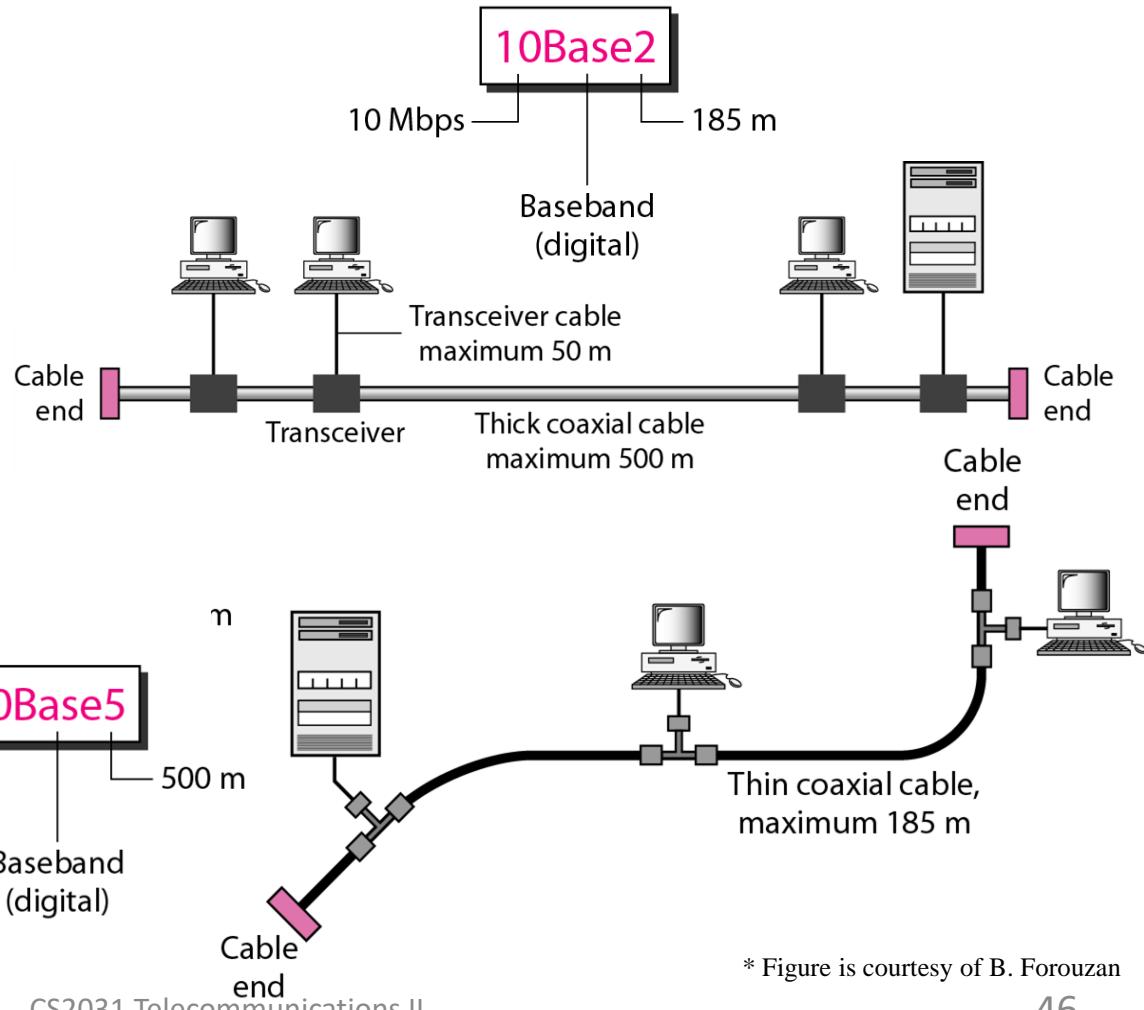
(a) 10Base5, (b) 10Base2, (c) 10Base-T.

* Figure is courtesy of A. Tanenbaum

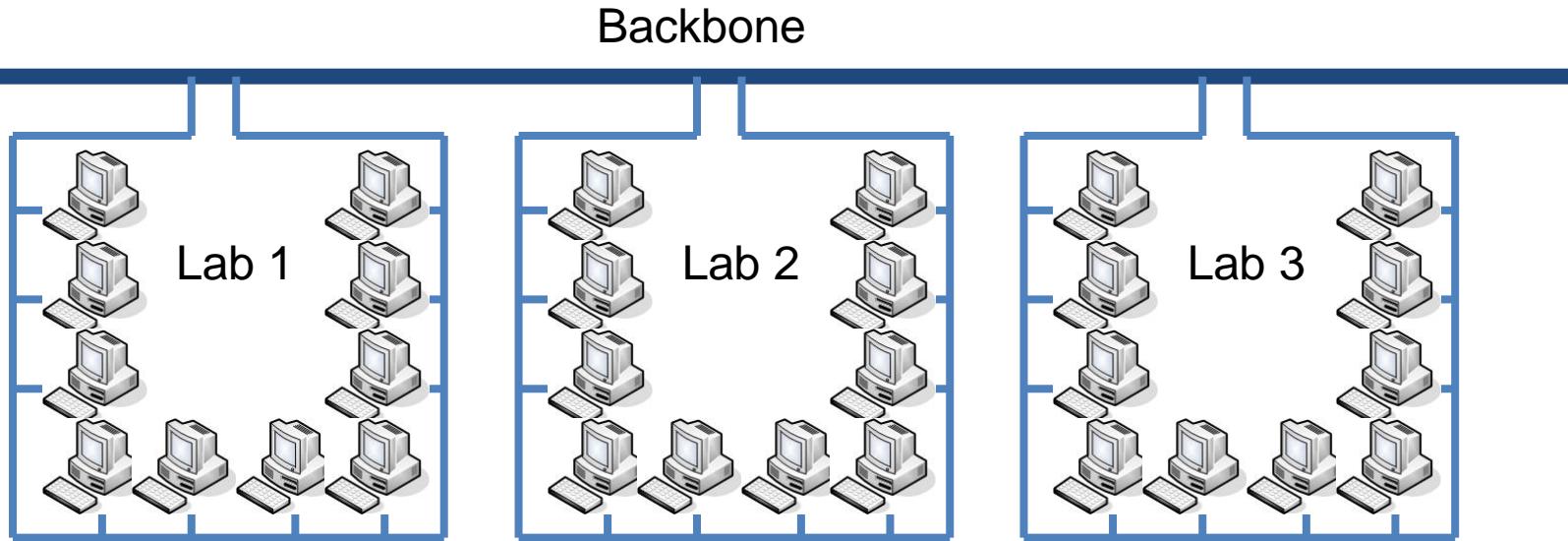


10Base5 & 10Base2

- Signal travels over cable & is picked up by all stations
- Used as backbone technology
- 10Base5: Stations linked into coaxial cable



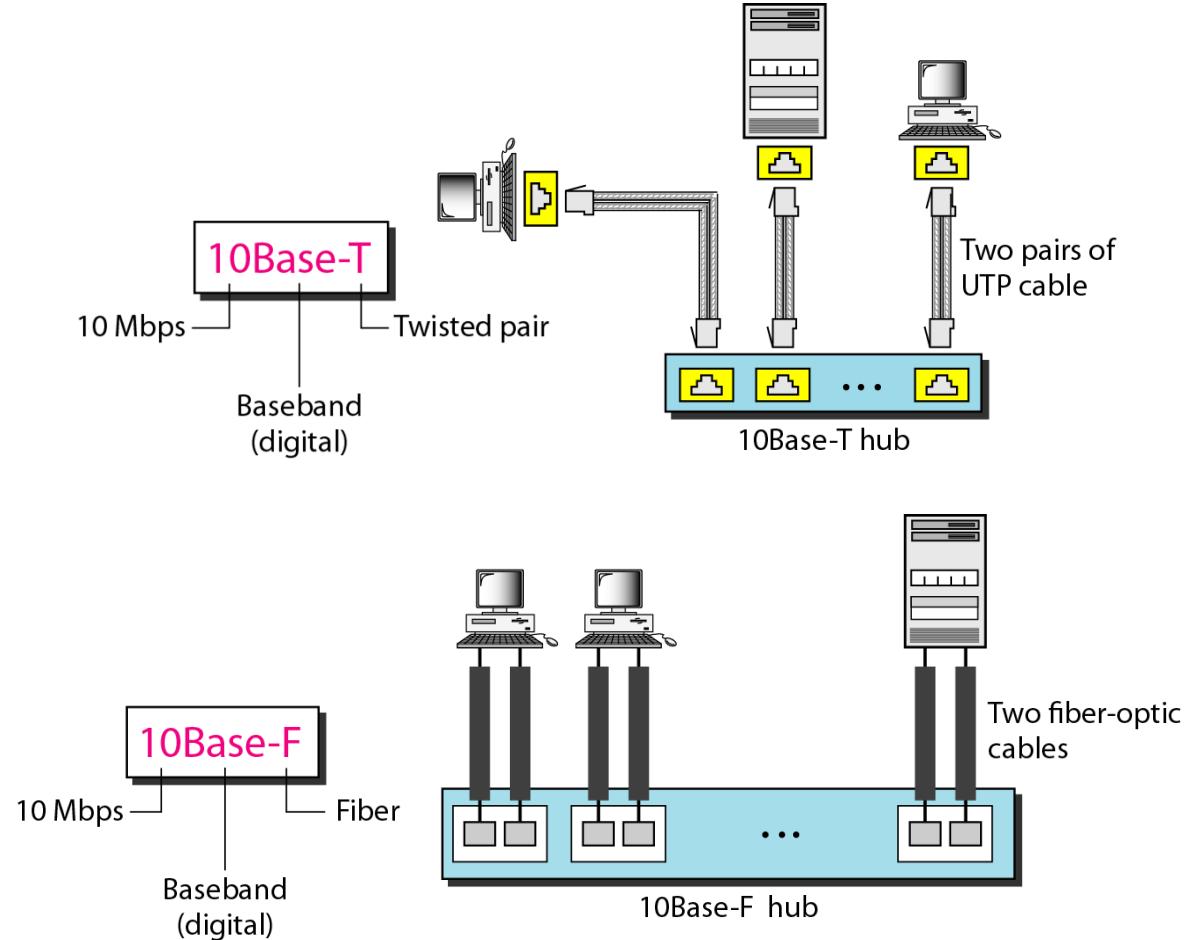
Common Configuration



- Thick coax cable as backbone:
 - Inflexible, large segments, 500m
 - Called Thicknet
- Thin coax cable to desktop:
 - Flexible, short segments, 200m
 - Called Thinnet

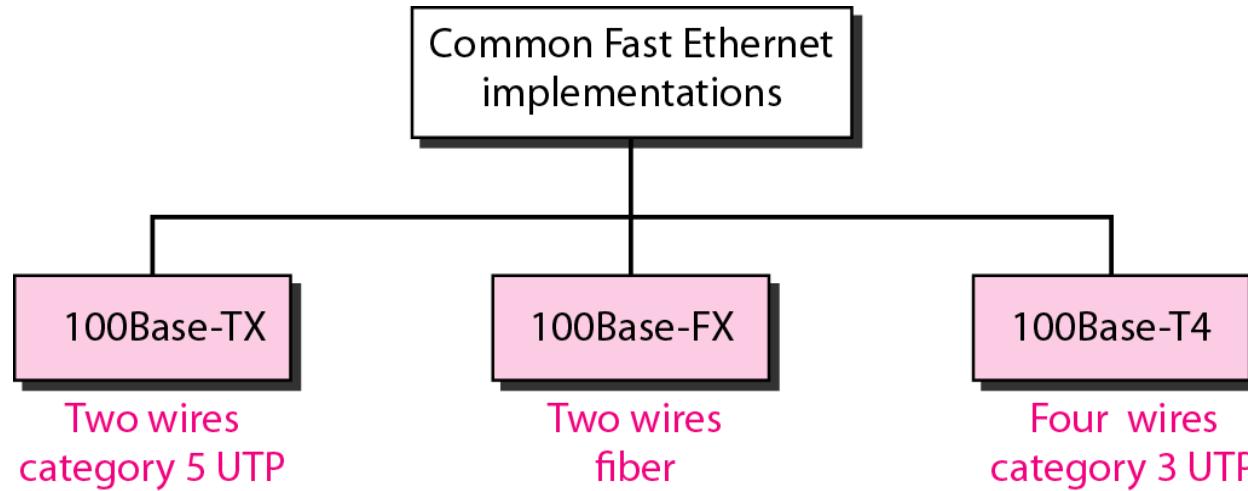
10Base-T & 10Base-F

- Hub replicates traffic to connected stations
- Each station has its own connection to hub
- Every station hears all traffic!



* Figure is courtesy of B. Forouzan

100Base-X

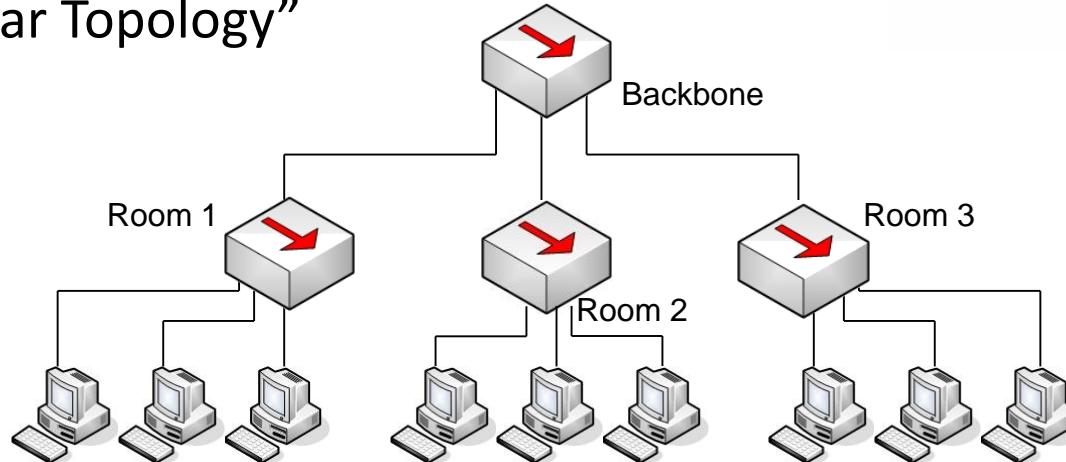
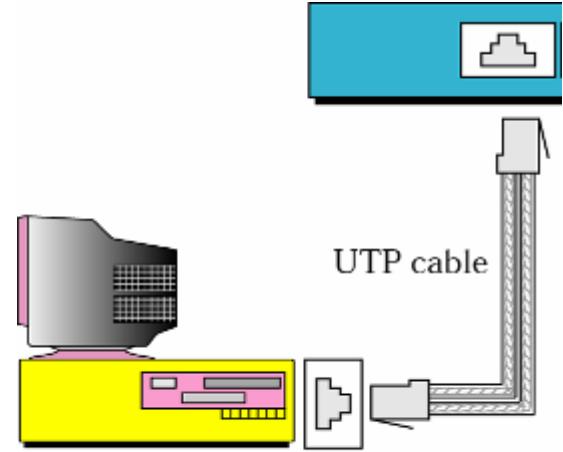


Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

* Figure is courtesy of B. Forouzan

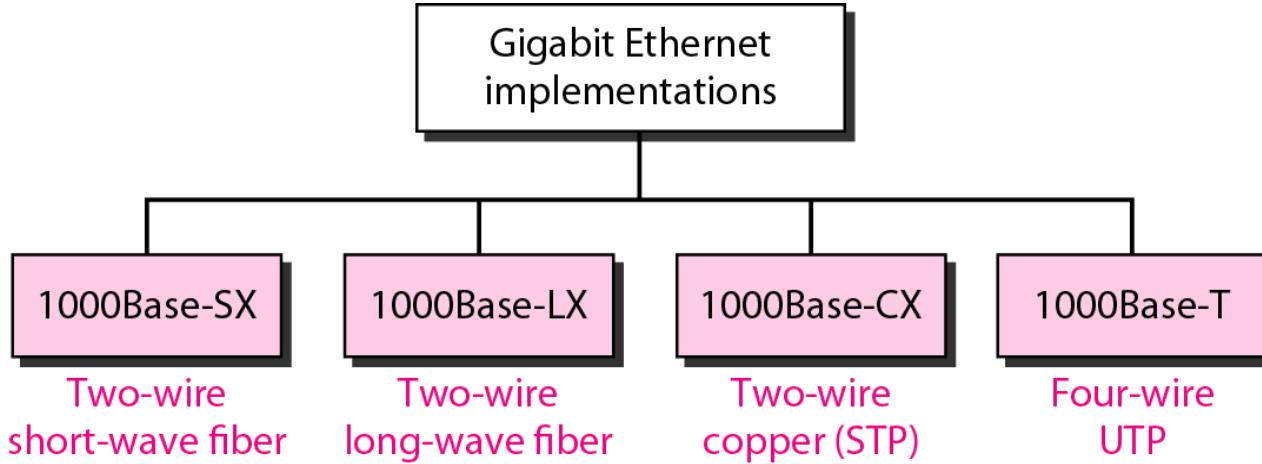
100Base-T

- 10/100 Mbps rate
 - latter called “Fast Ethernet”
- T stands for Twisted Pair
- Hub to which nodes are connected by twisted pair
 - “Star Topology”



* Figure is courtesy of B. Forouzan

Gigabit Ethernet



- Minimum frame length: 512 bytes

* Figure is courtesy of B. Forouzan



Ethernet Standards

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP



802.3ae 10GB-Ethernet

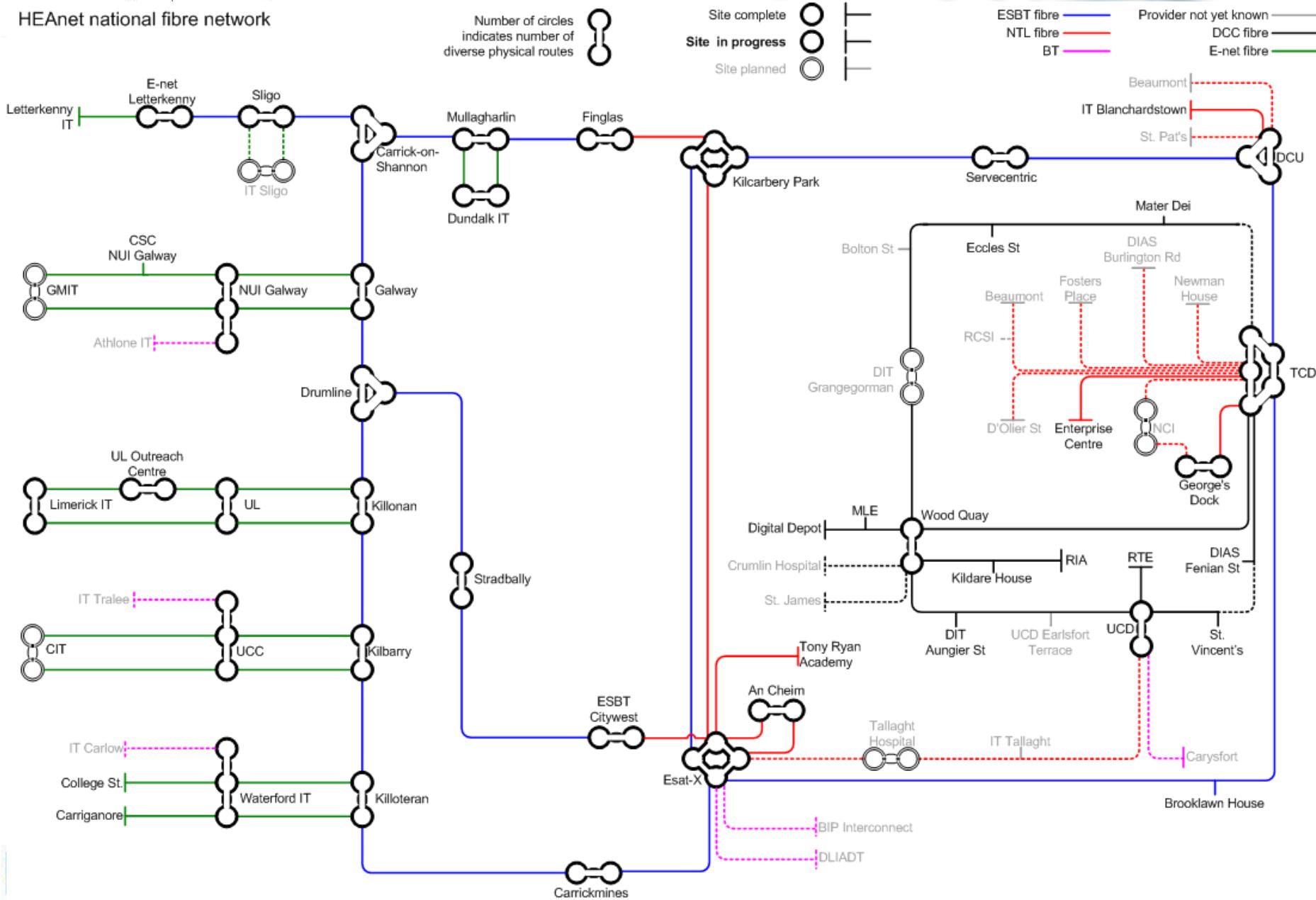
Characteristics	10GBase-S	10GBase-L	10GBase-E
Media	Short-wave 850-nm multimode	Long-wave 1310-nm single mode	Extended 1550-mm single mode
Maximum length	300 m	10 km	40 km

- Backbone technology
- Based on optical fibre

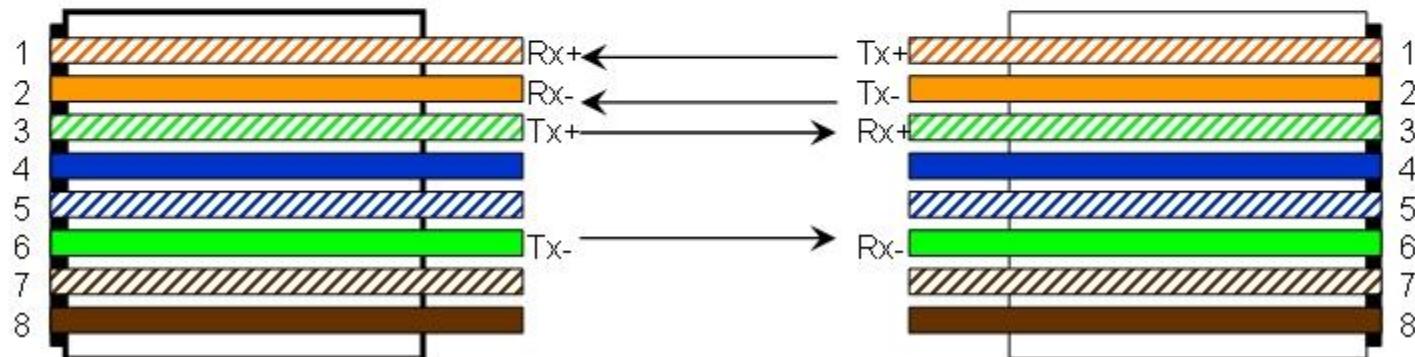
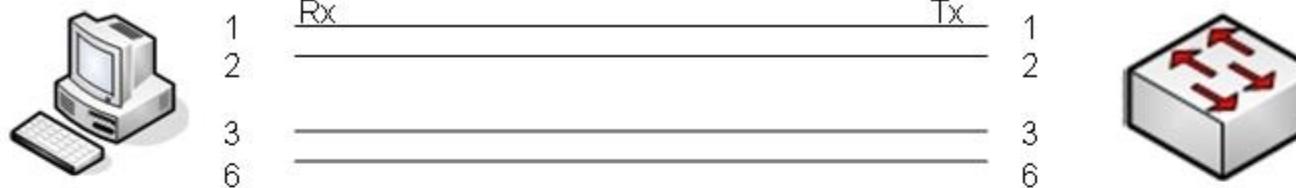


school of Computer Science & Statistics

HEAnet national fibre network

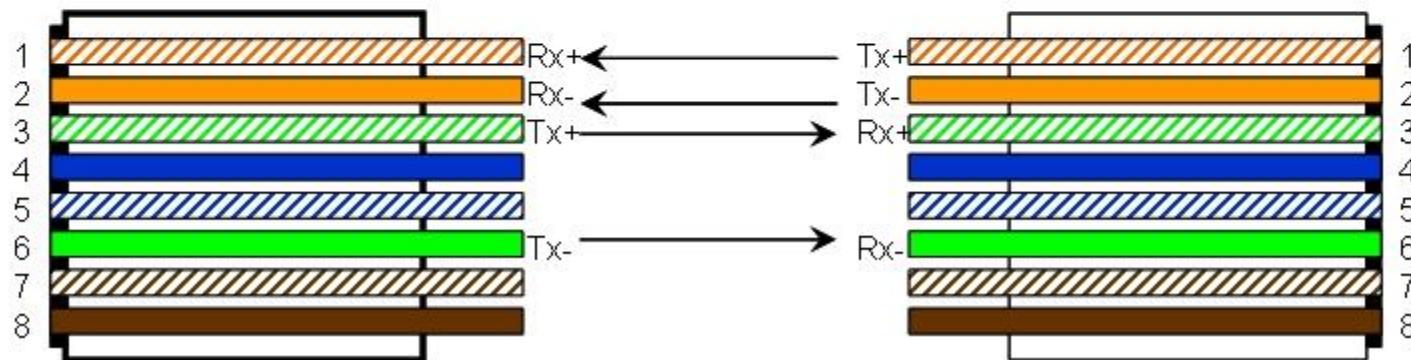
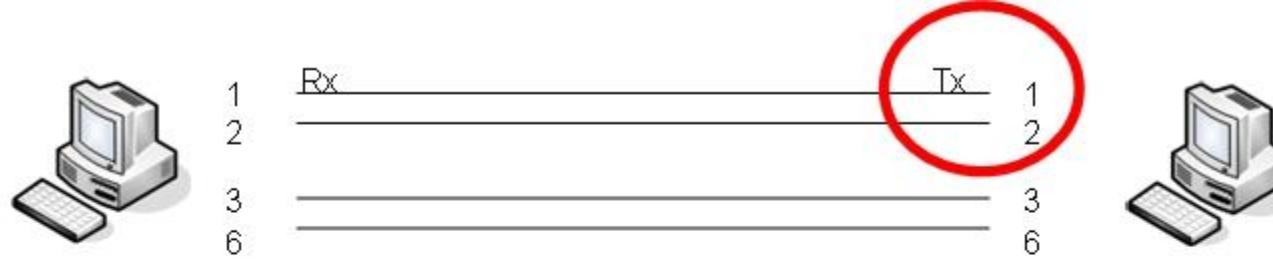


Straight Cabling & RJ45



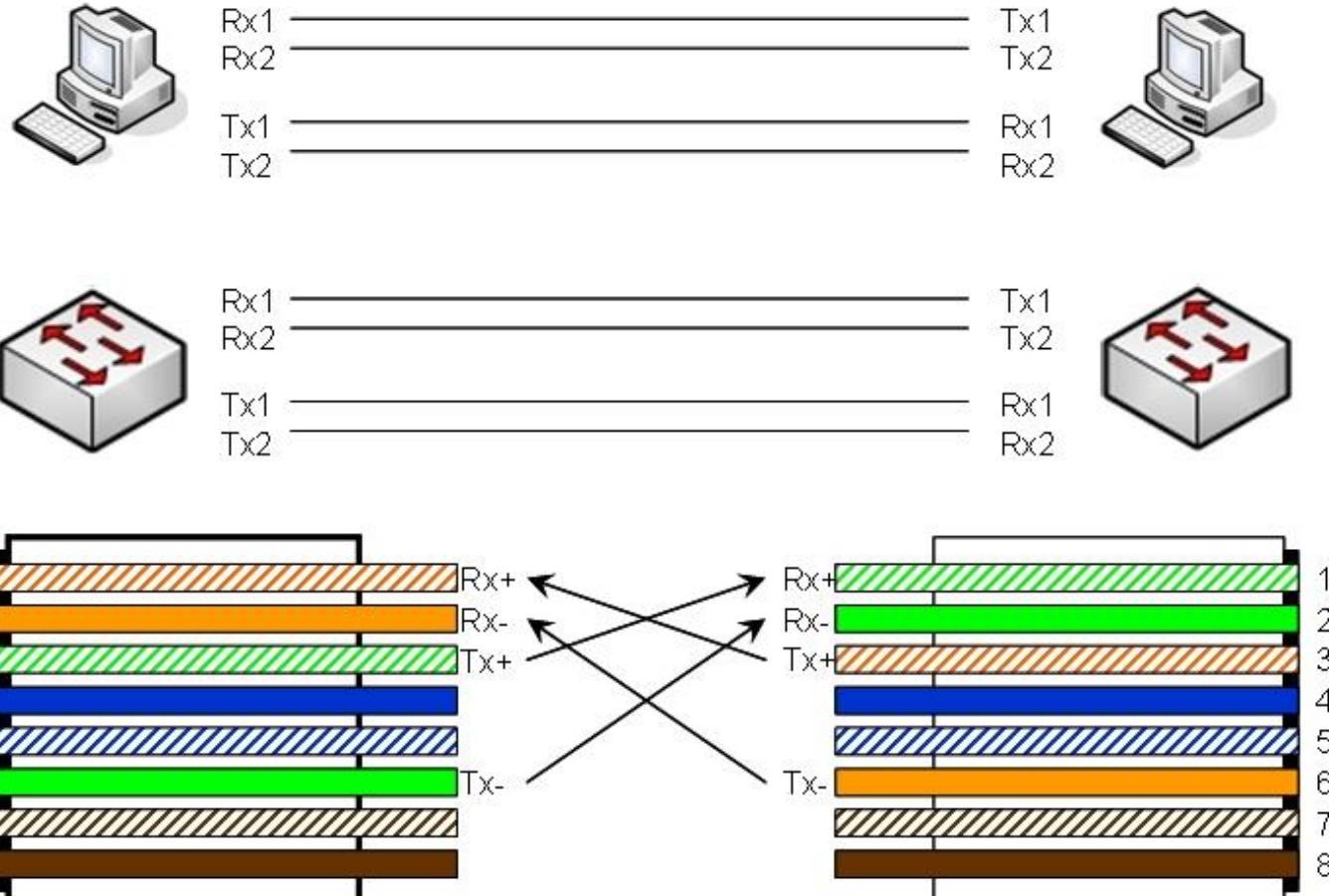
- Switch expects
 - Transmission from station on 3 & 6
 - Transmits on 1 & 2

Straight Cabling II



- Switch expects
 - Transmission from station on 3 & 6
 - Transmits on 1 & 2

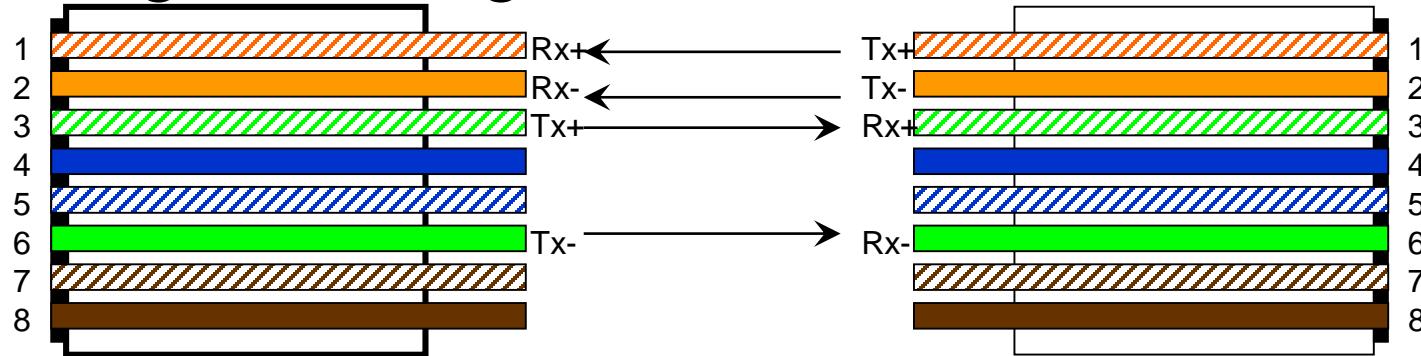
Crossover Cabling



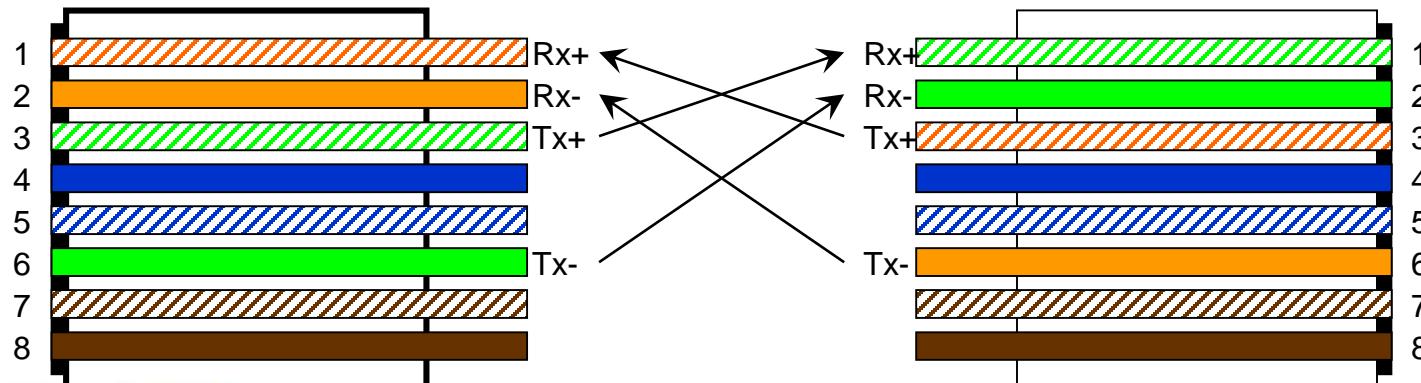
- Direct connection of desktops & infrastructure equipment

RJ45 Cabling

- Straight cabling:



- Crossover cabling:

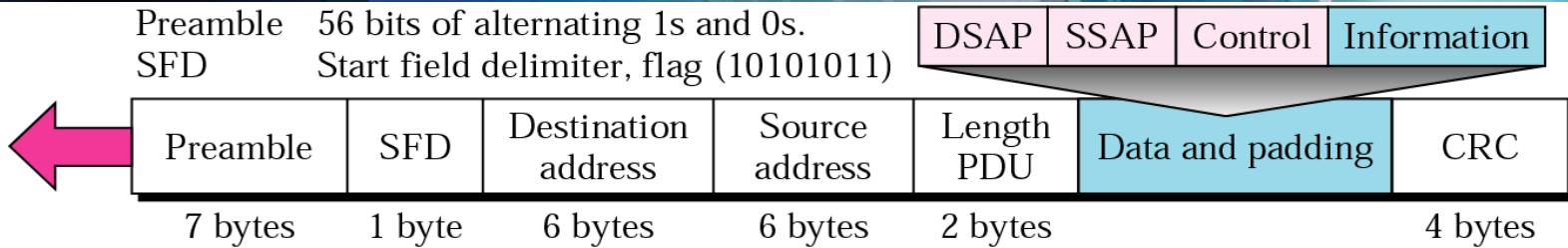


Overview

- Switches / Ethernet Switches
- Network Layer
- Addressing, IPv4 Addresses & Routers
- CIDR
- NAT
- ARP



802.3 MAC Format



- 64-bit frame preamble (10101010) used to synchronize reception
 - 7 bit preamble (10101010) + 1 start flag (10101011)
- Maximum frame length: 1536 bytes
 - ⇒ max 1500 bytes payload
- Minimum frame length: 64 bytes
 - ⇒ min 46 bytes payload

* Figure is courtesy of B. Forouzan

Ethernet Standards

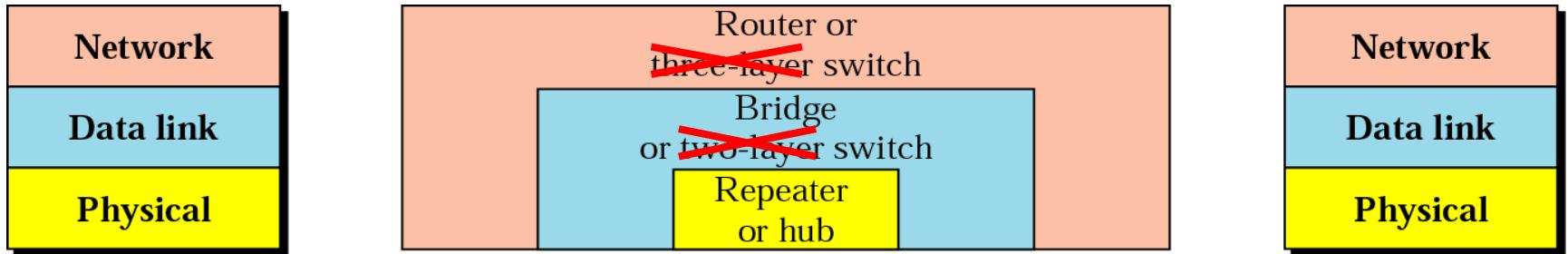
Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP



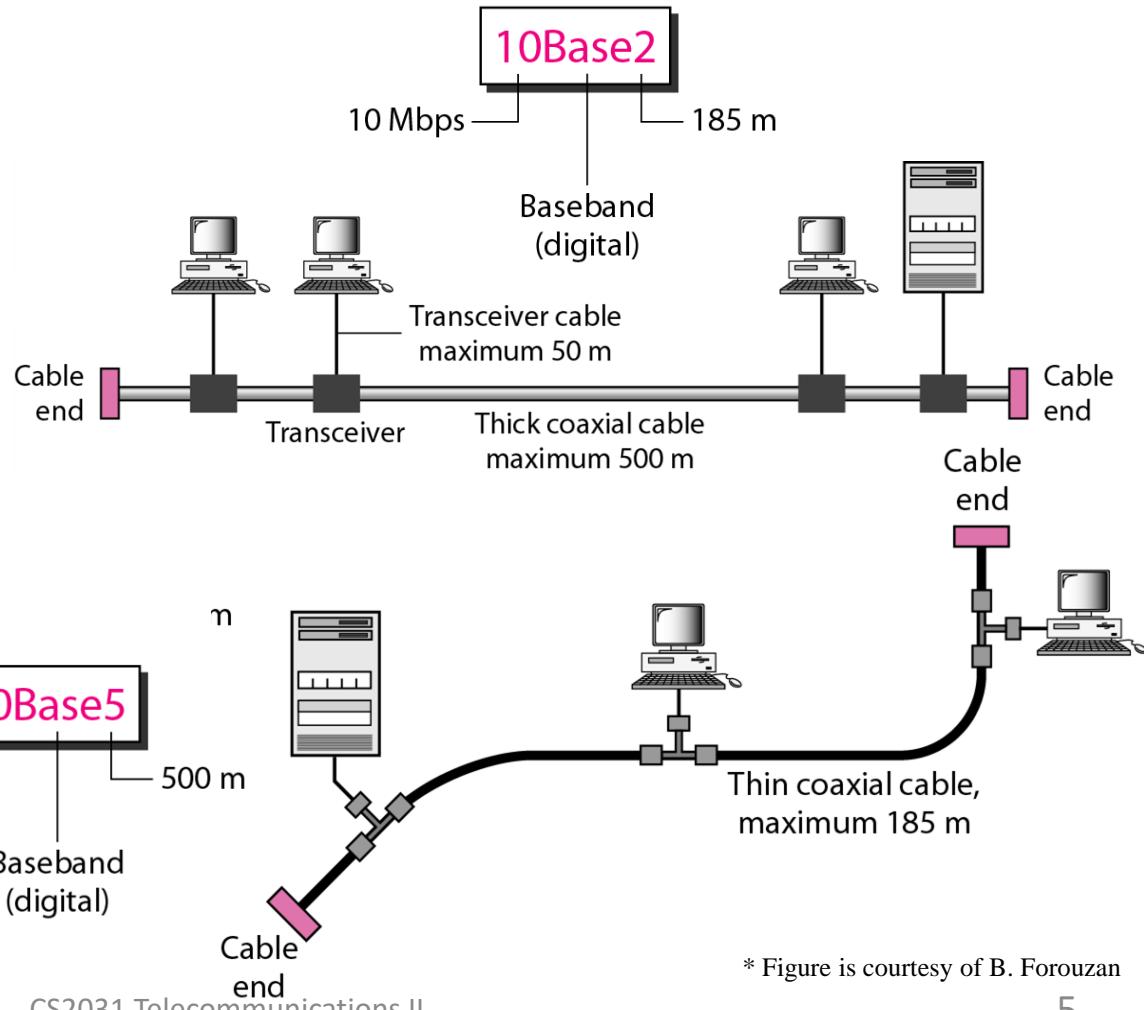
Connecting Devices



- Physical Layer: Repeater or Hub
- Data Link Layer: Bridge or Layer-2 Switch
- Network Layer: Router or Layer-3 Switch

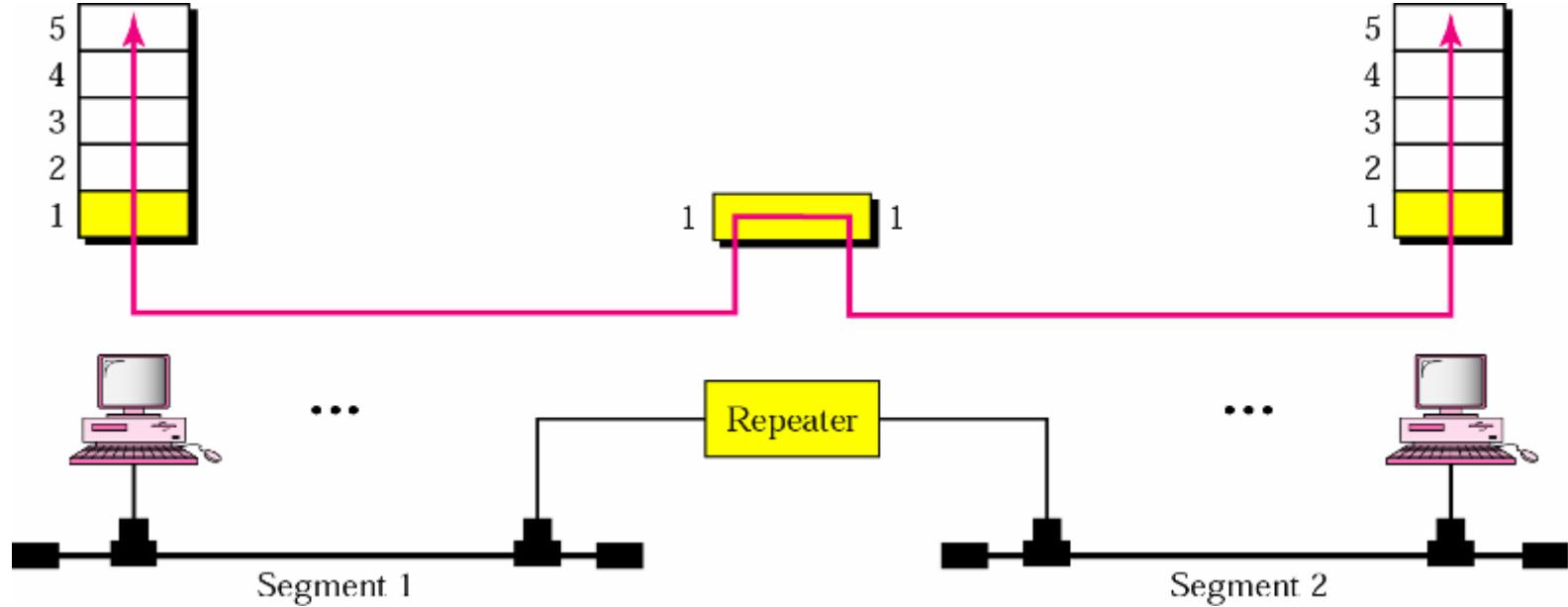
10Base5 & 10Base2

- Signal travels over cable & is picked up by all stations
- Used as backbone technology
- 10Base5: Stations linked into coaxial cable



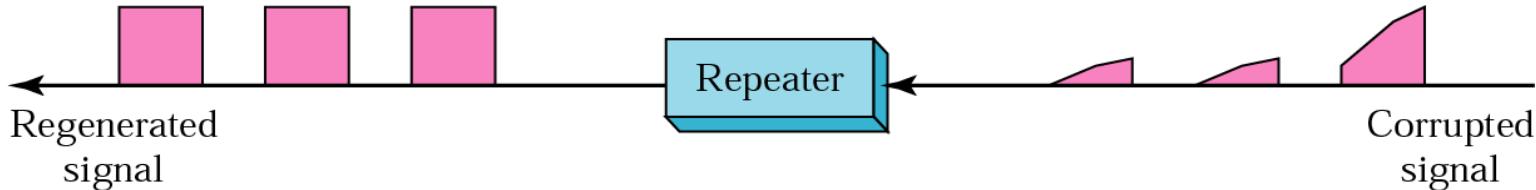
* Figure is courtesy of B. Forouzan

Repeater

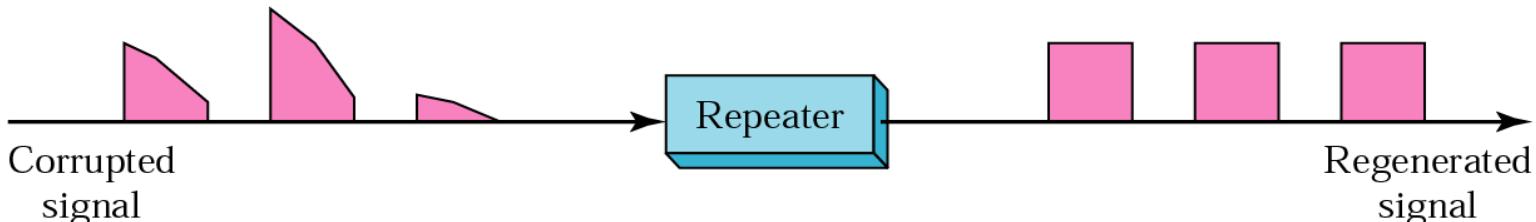


- A repeater connects segments of a LAN
- A repeater forwards every frame; it has no filtering capability

Function of a Repeater



a. Right-to-left transmission.

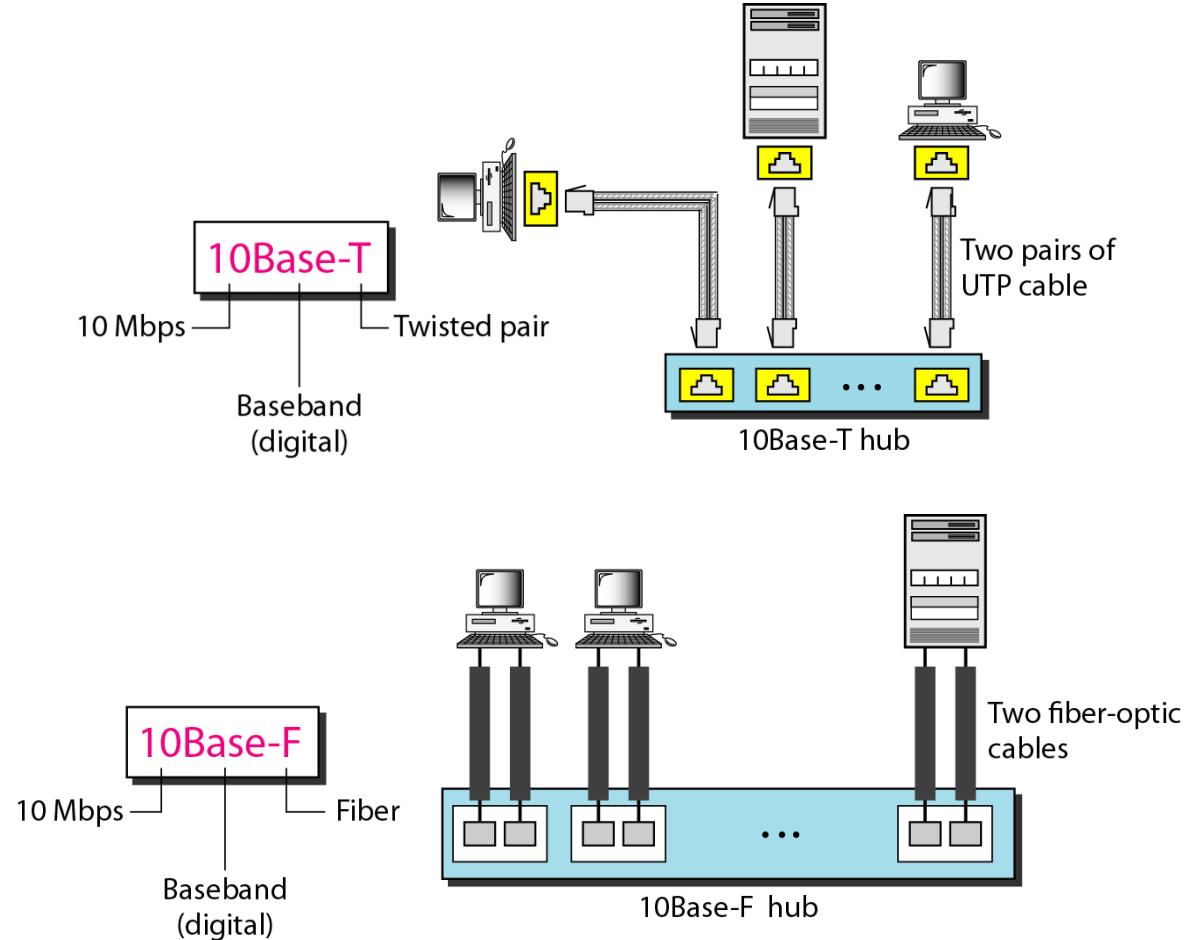


b. Left-to-right transmission.

- A repeater is a regenerator,
not an amplifier!

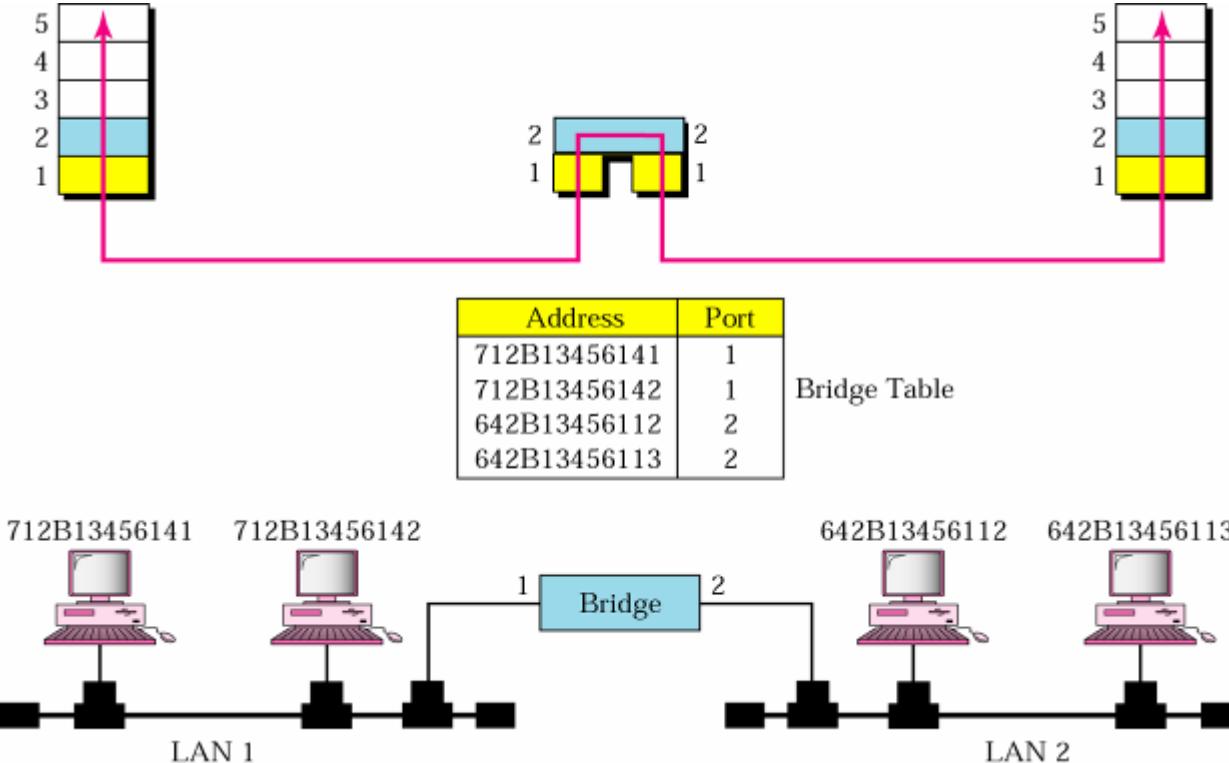
10Base-T & 10Base-F

- Hub replicates traffic to connected stations
- Each station has its own connection to hub
- Every station hears all traffic!



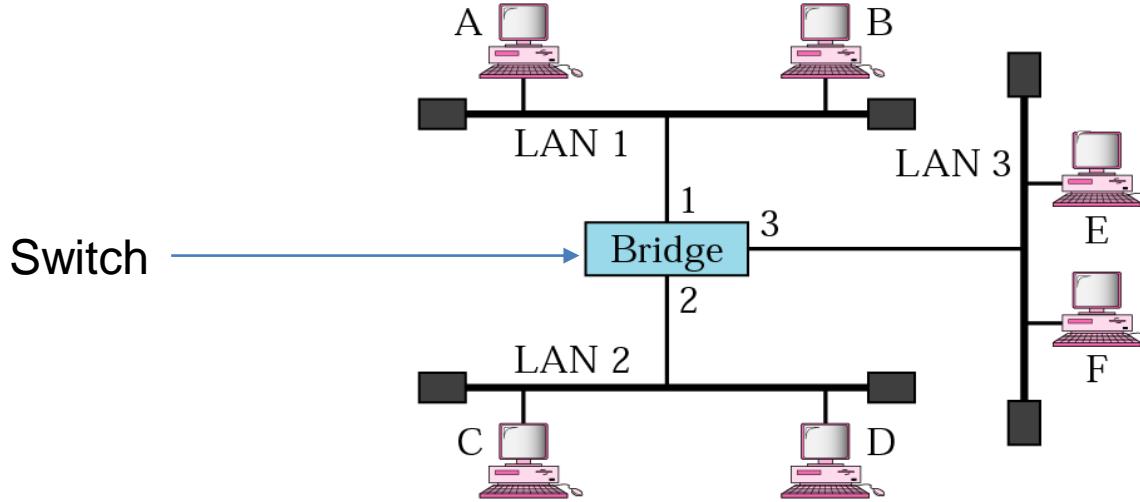
* Figure is courtesy of B. Forouzan

Functions of a Bridge



- Read all frames transmitted on one LAN and accept those address to any station on the other LAN
- Using MAC protocol for second LAN, retransmit each frame
- Do the same the other way round

Learning Bridges



Address	Port

a. Original

Address	Port
A	1

b. After A sends a frame to D

Address	Port
A	1
E	3

c. After E sends a frame to A

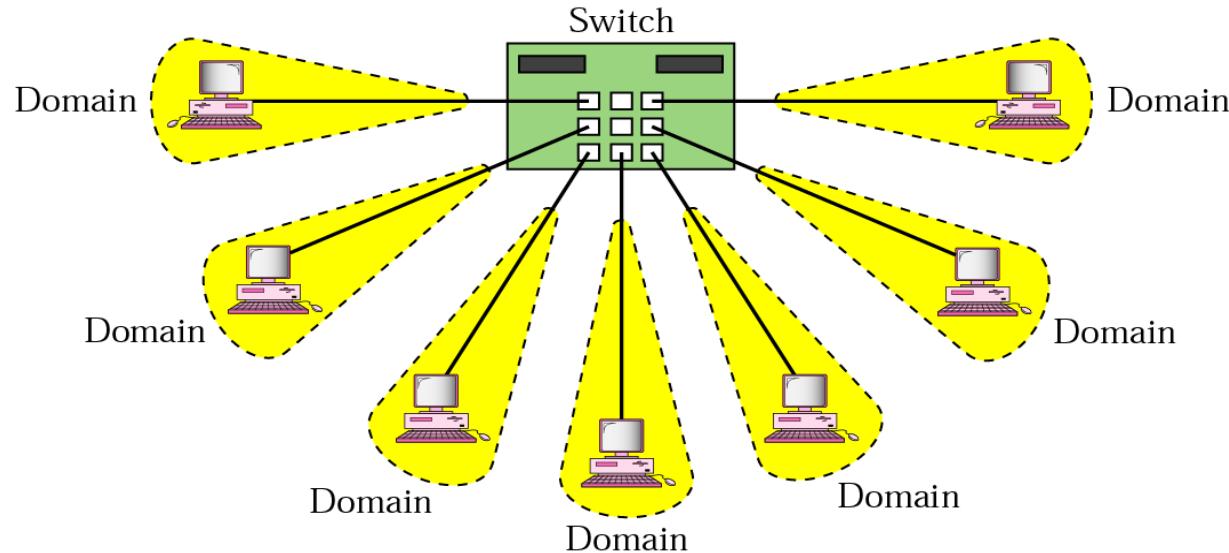
Address	Port
A	1
E	3
B	1

d. After B sends a frame to C

- Initially bridge forwards frames on all segments except incoming port
- Learns addresses from frames that pass through



Switched Ethernet



- **Switch delivers packets to individual machines**
 - Without affecting communication with other machines
- **Collisions only occur on individual links**

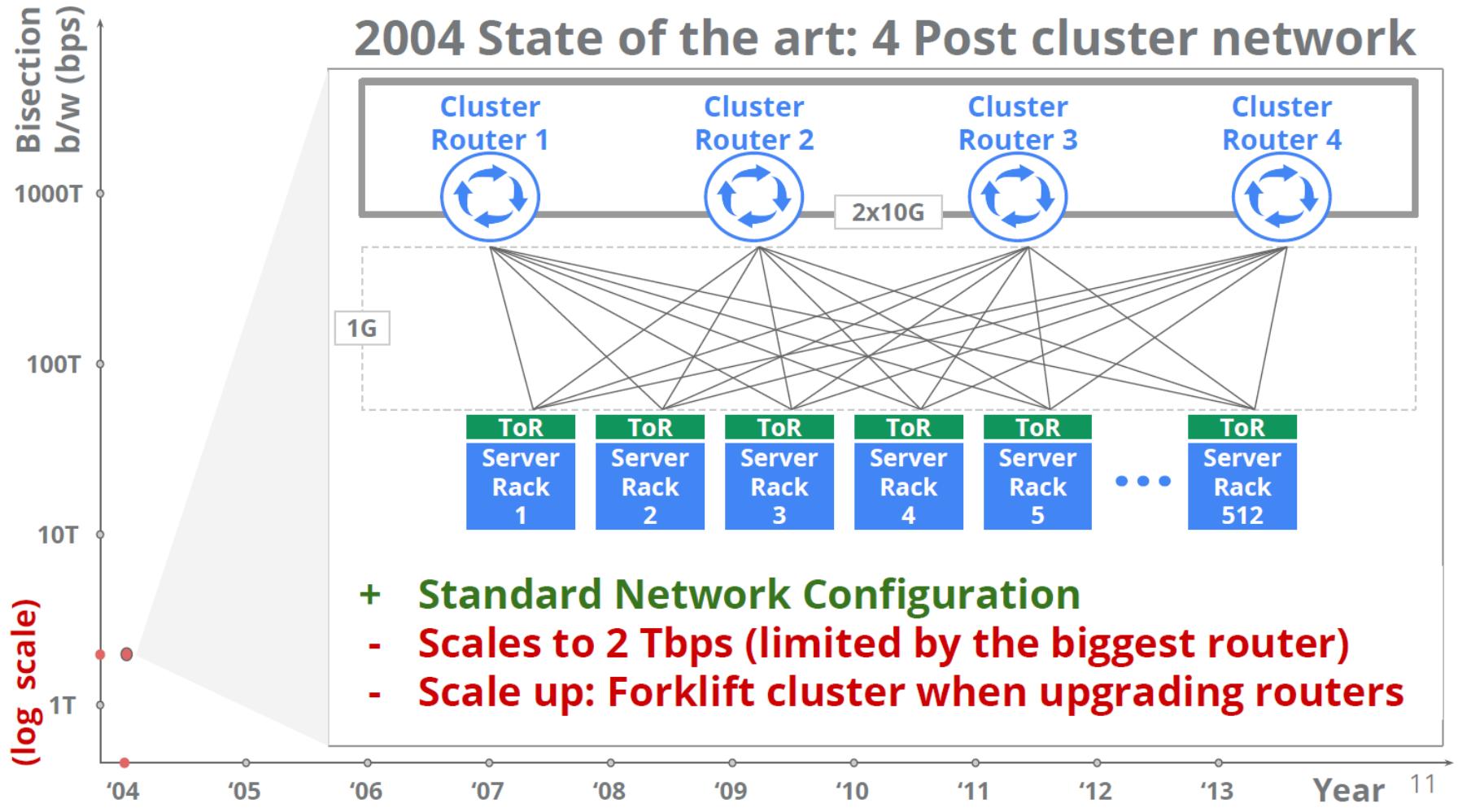
* Figure is courtesy of B. Forouzan

Switch



* Figure is courtesy of Cisco & Reddit

Google's Infrastructure



Types of Layer 2 Switches

- **Store-and-Forward** switch
 - Accepts frame on input line
 - Buffers it briefly,
 - Then routes it to appropriate output line
 - Delay between sender and receiver
 - Boosts integrity of network
- **Cut-Through** switch
 - Takes advantage of destination address appearing at beginning of frame
 - Switch begins repeating frame onto output line as soon as it recognizes destination address
 - Highest possible throughput
 - Risk of propagating bad frames
 - Switch unable to check CRC prior to retransmission



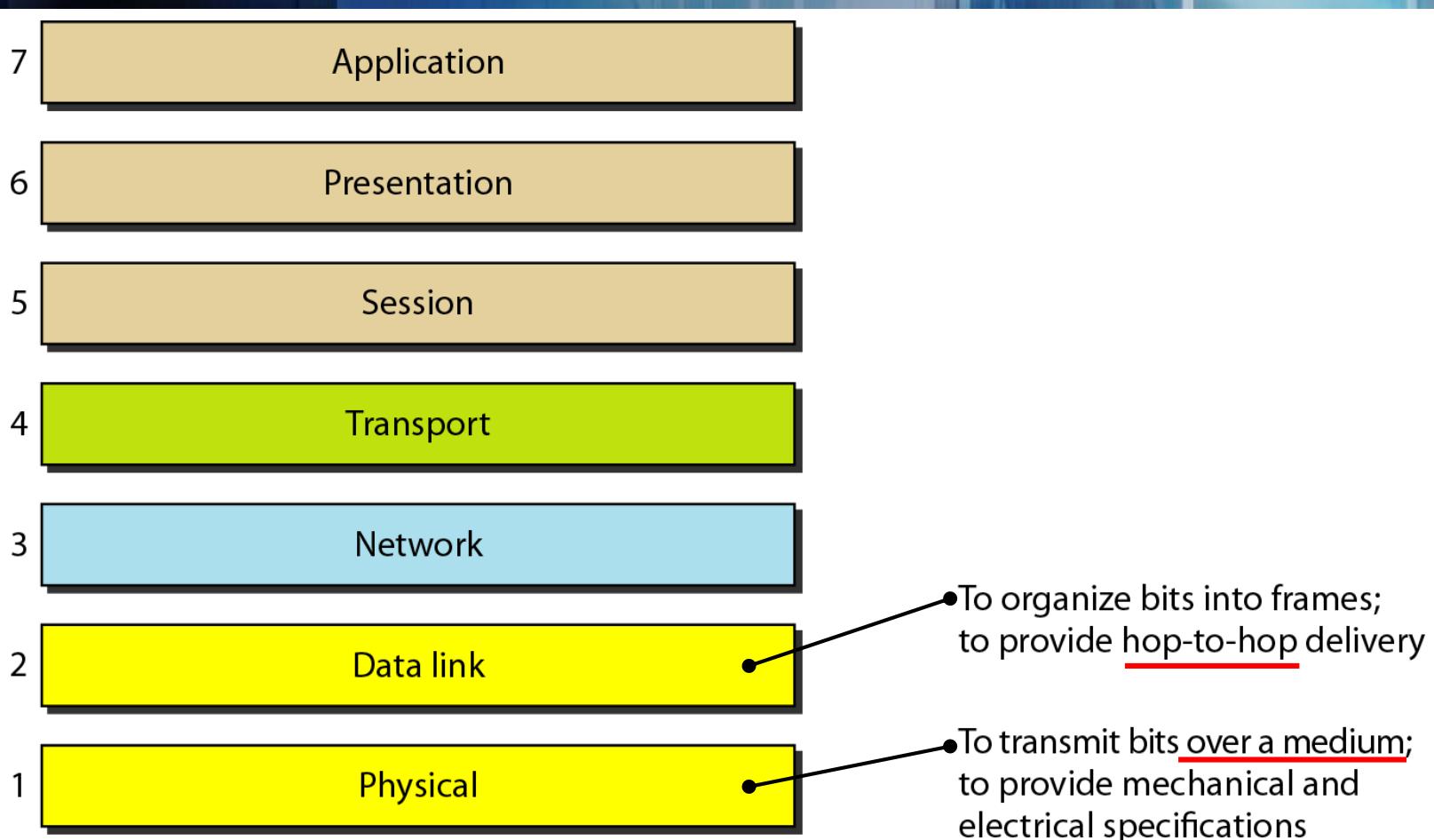
CS2031

Telecommunications II

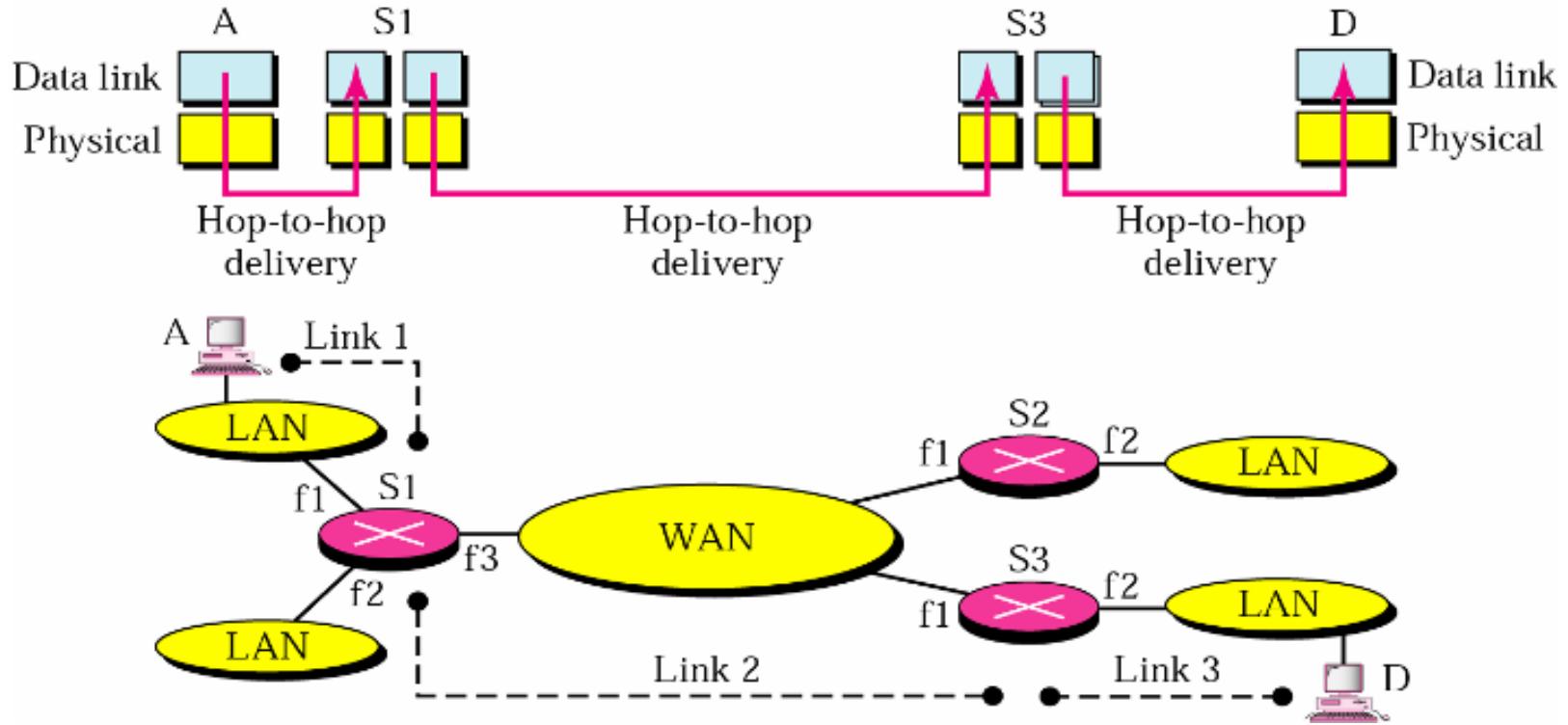
Network Layer



OSI Model



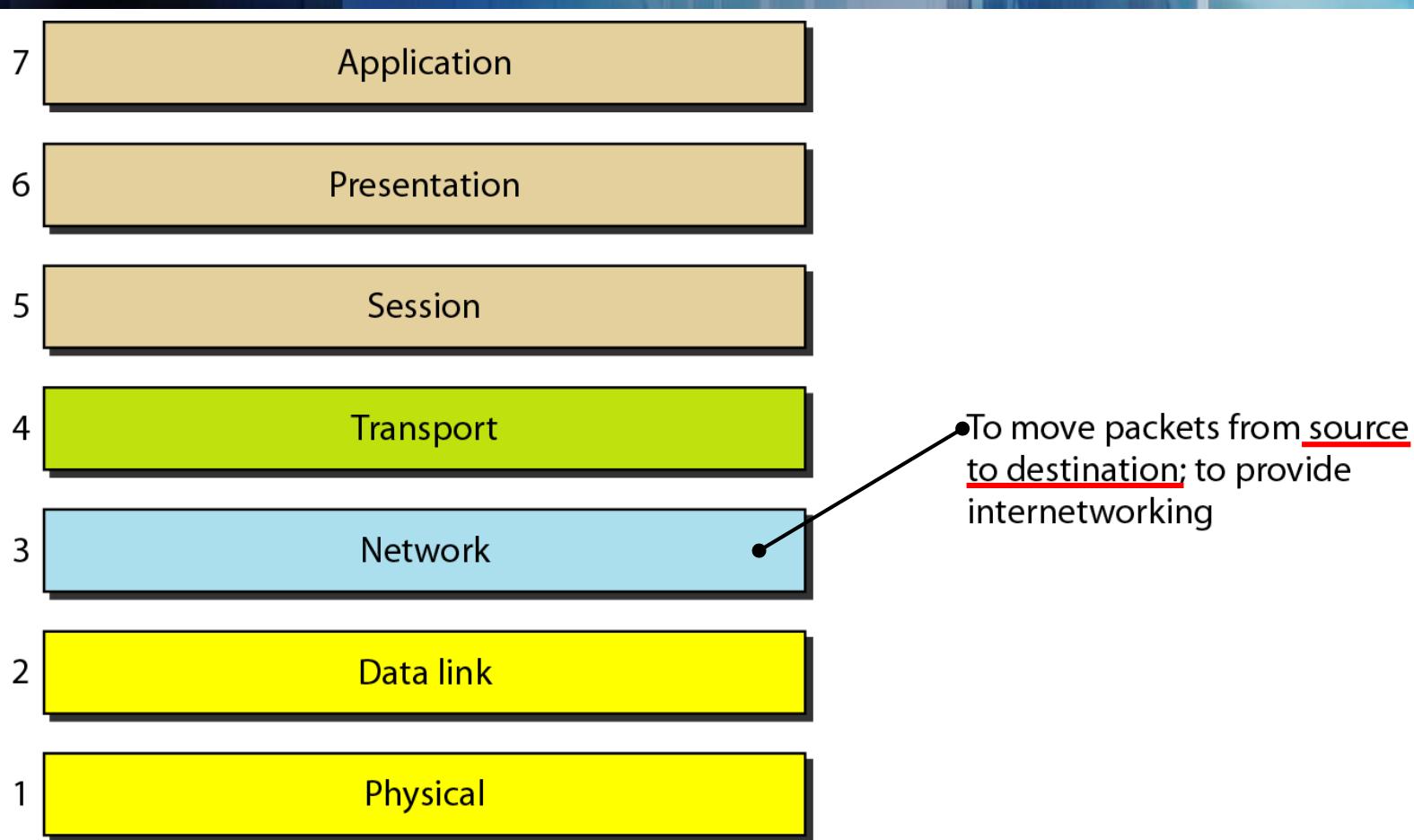
Data Link Layer



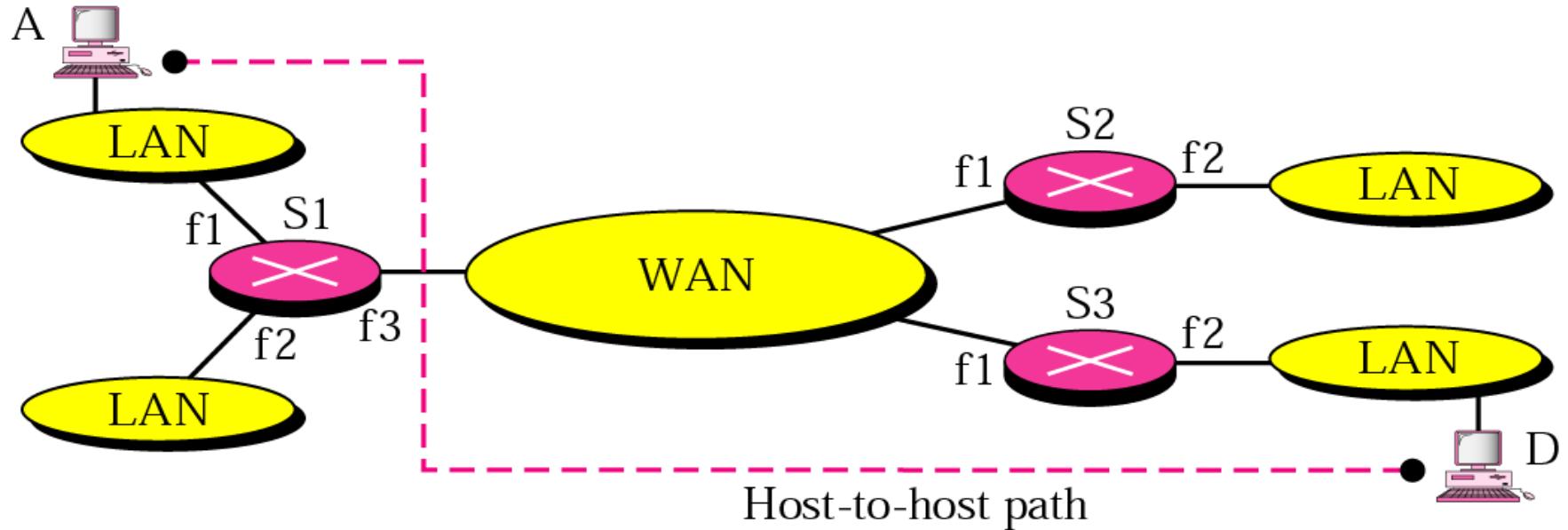
- Hop-to-hop communication

* Figure is courtesy of B. Forouzan

OSI Model

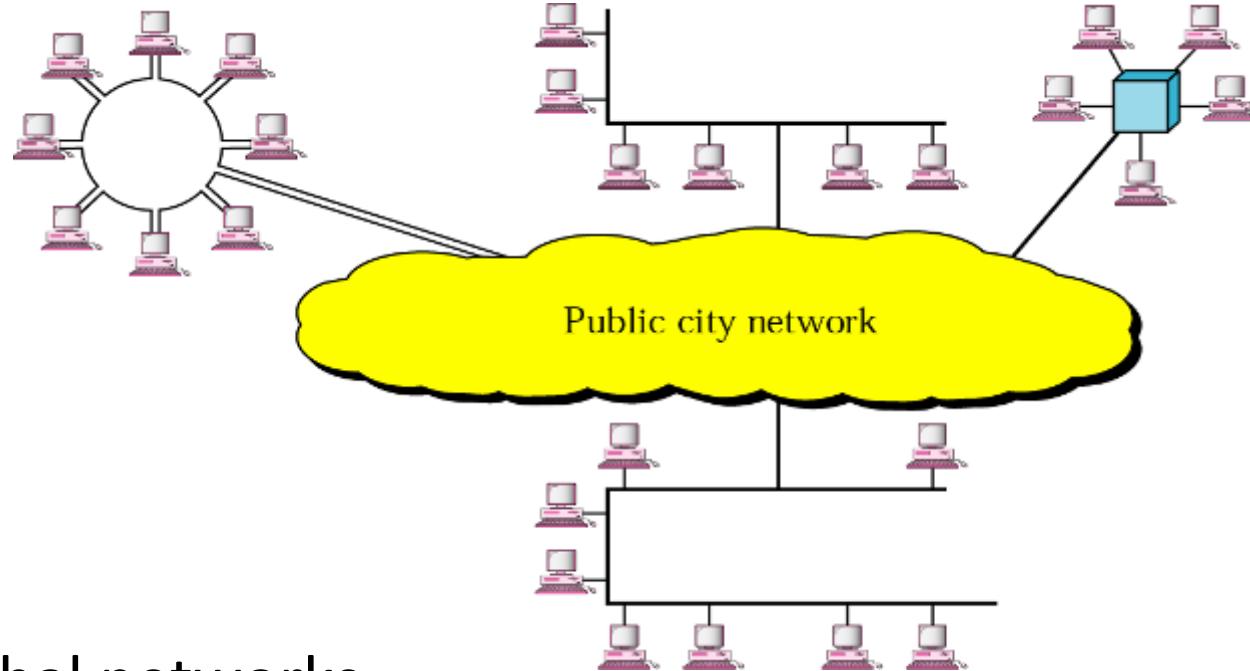


Network Layer



- Responsible for source-to-destination delivery of a packet across multiple network links.

Problems faced by Network Layers



- Global networks
 - Various different architectures
 - Various formats for hardware addresses
 - Various maximum transmission units (MTUs)

Paul Baran, 1964

- Paul Barran, On Distributed Communication Networks, IEEE Transactions on Communication Systems, Volume 12, No. 1, March 1964
- Introduced Concepts of
 - Distributed Networks
 - Routing
(hot-potato-routing)
 - Packet-Switching
(message-block)

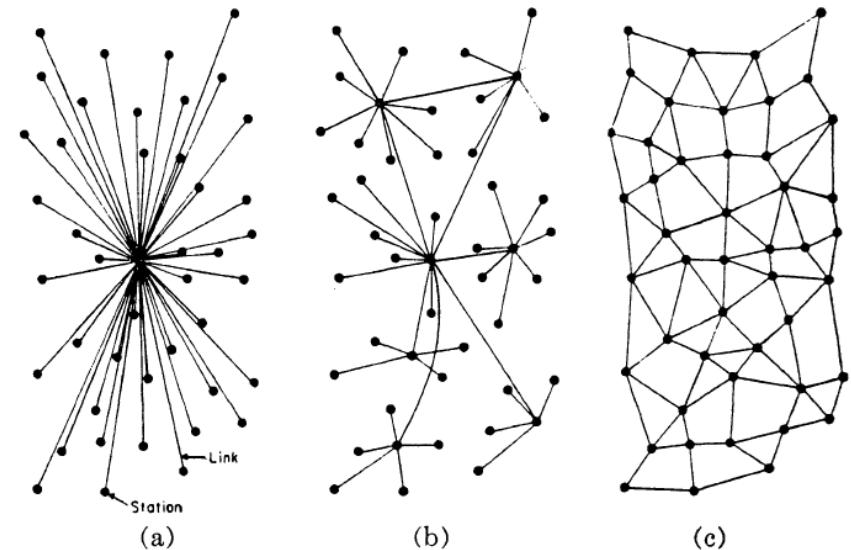
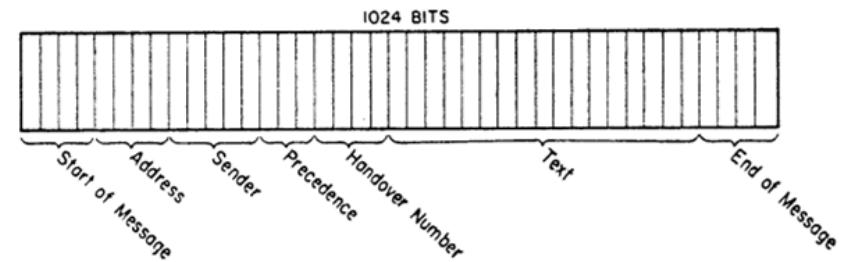


Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.



AT&T's comment back then was that packet-switching will never be useful.

HOST-HOST Communication, 1970

- Paul Barran, [On Distributed Communication Networks](#), IEEE Transactions on Communication Systems, Volume 12, No. 1, March 1964
- Introduced Concepts of
 - Distributed Networks
 - Routing
(hot-potato-routing)
 - Packet-Switching
(message-block)
- Name worth mentioning: Bob Kahn

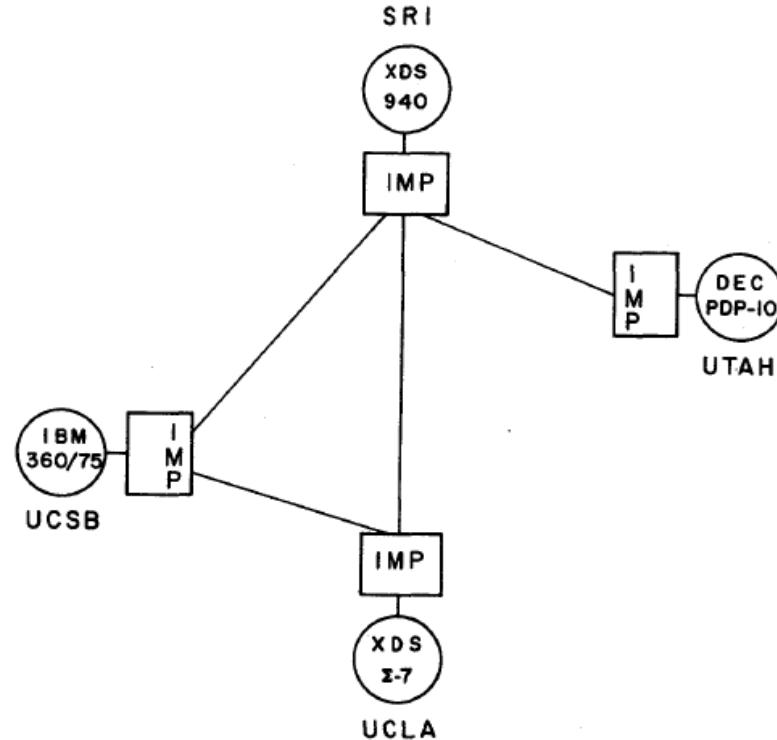
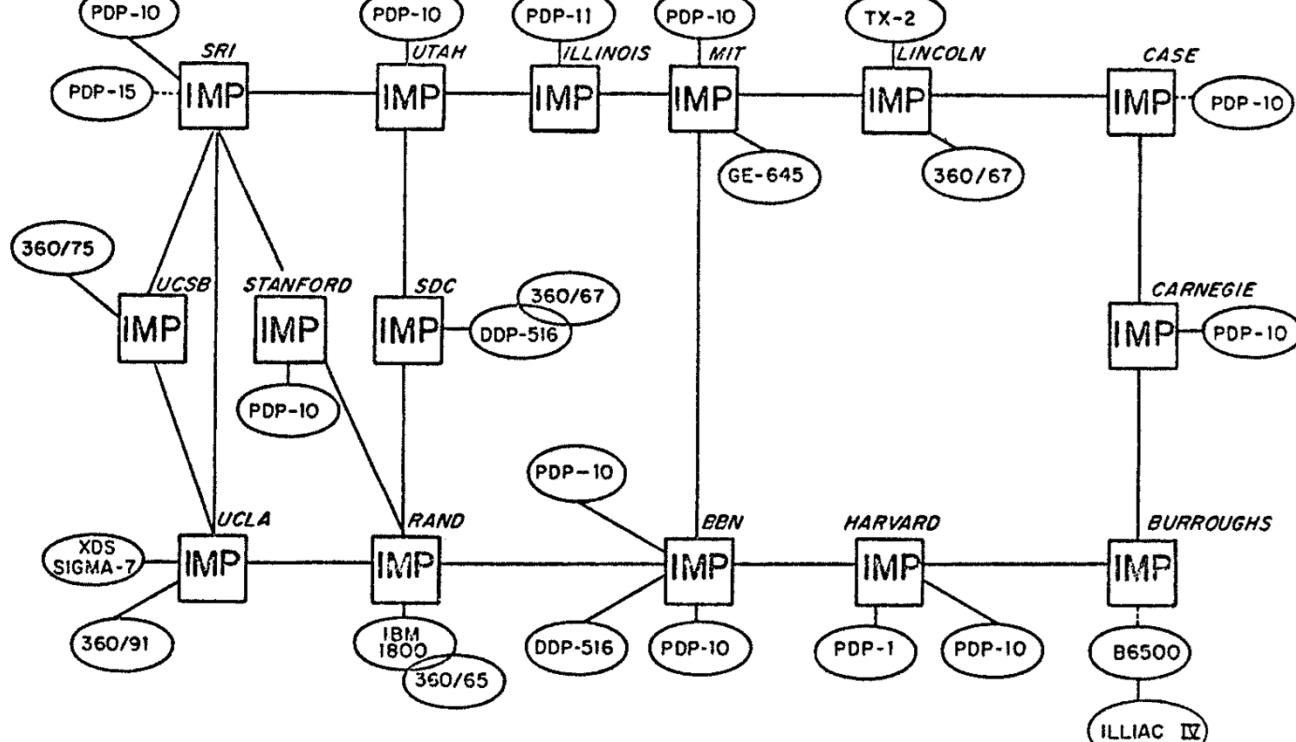


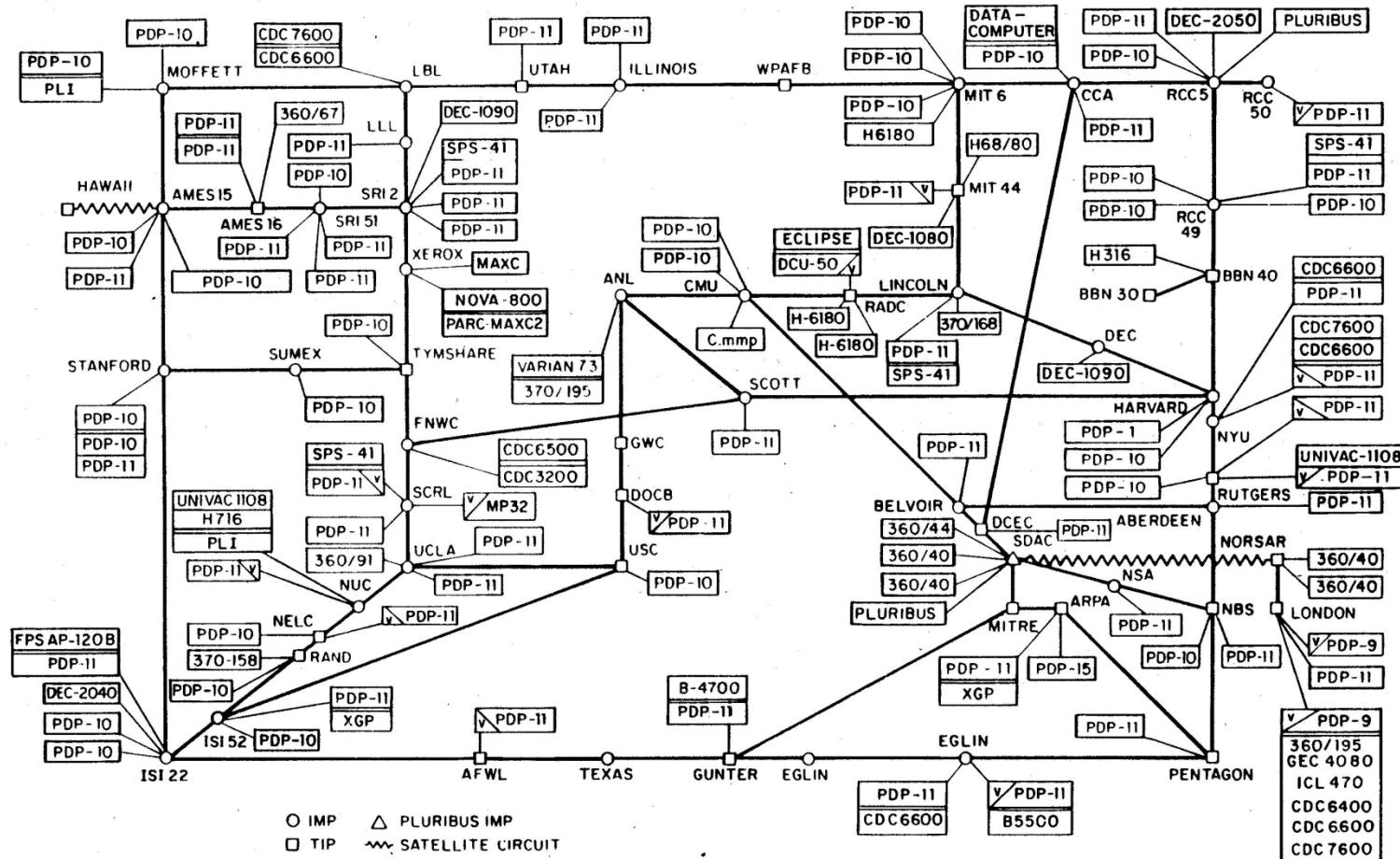
Figure 1—Initial network configuration

* C. Stephen Carr, Stephen D. Crocker, and Vinton G. Cerf, [HOST-HOST Communication Protocol in the ARPA Network](#), Spring Joint Computer Conference, pages 589-597, Atlantic City, NJ, USA, May 1970

ARPANET 1971



ARPANET 1977



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY.)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES



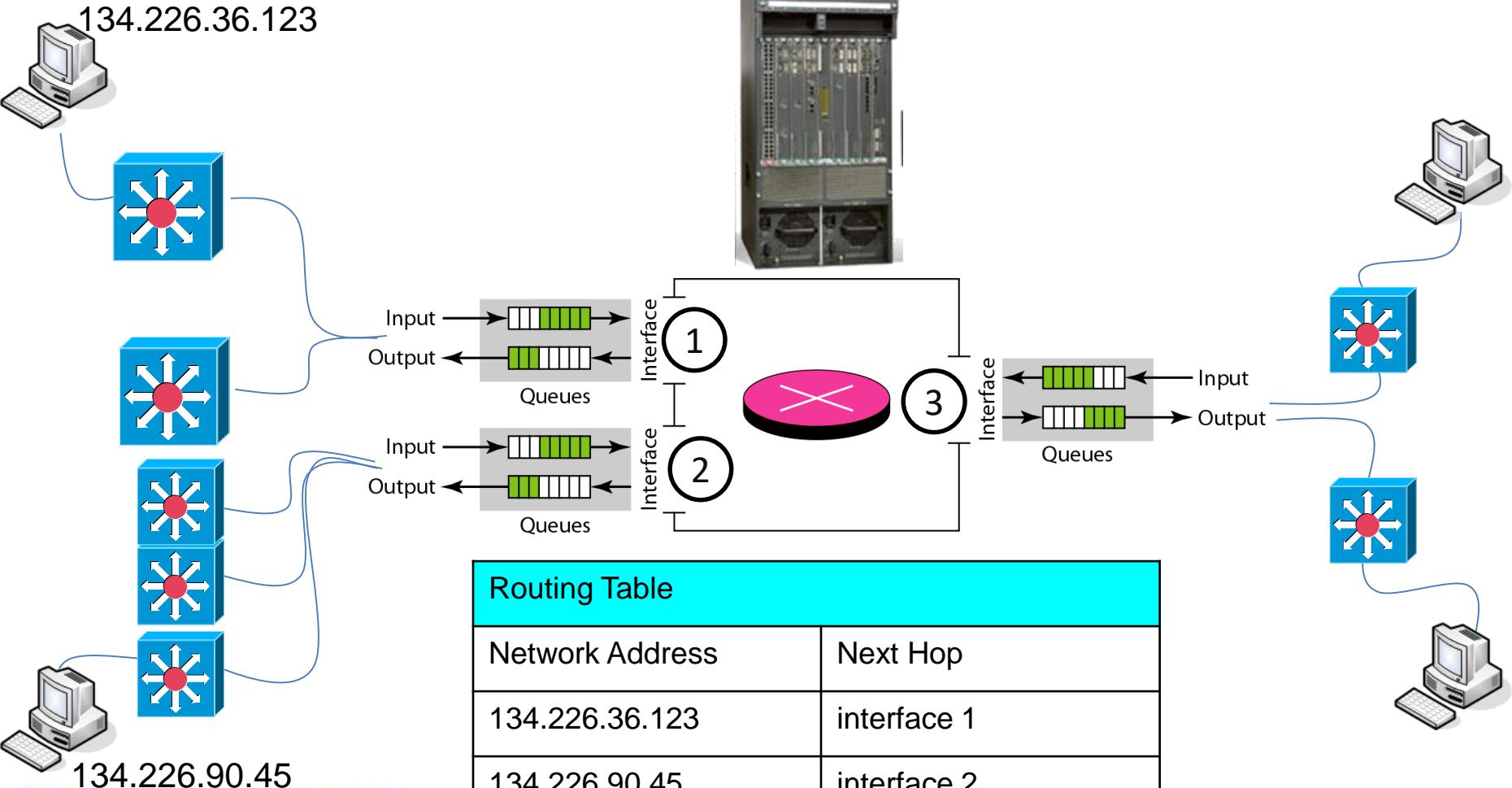
IP Addresses

10000000 00001011 00000011 00011111

128.11.3.31

- 32-bit number
 - 4.294.967.296 addresses
- IP addresses are unique and universal
 - with some exceptions
- Dotted decimal notation:
 - Bytes of binary notation represented as decimal separated by dot

Task of Routers



Routers

- One Main Interest
Forwarding Packets
- Important Aspects
Queue Length
Routing Table

Destination	Gateway	Interface
IP Range ₁	G ₁	IF ₁
IP Range ₂	G ₂	IF ₂



* Figure is courtesy of Cisco

Everything's a Router

Active Routes:

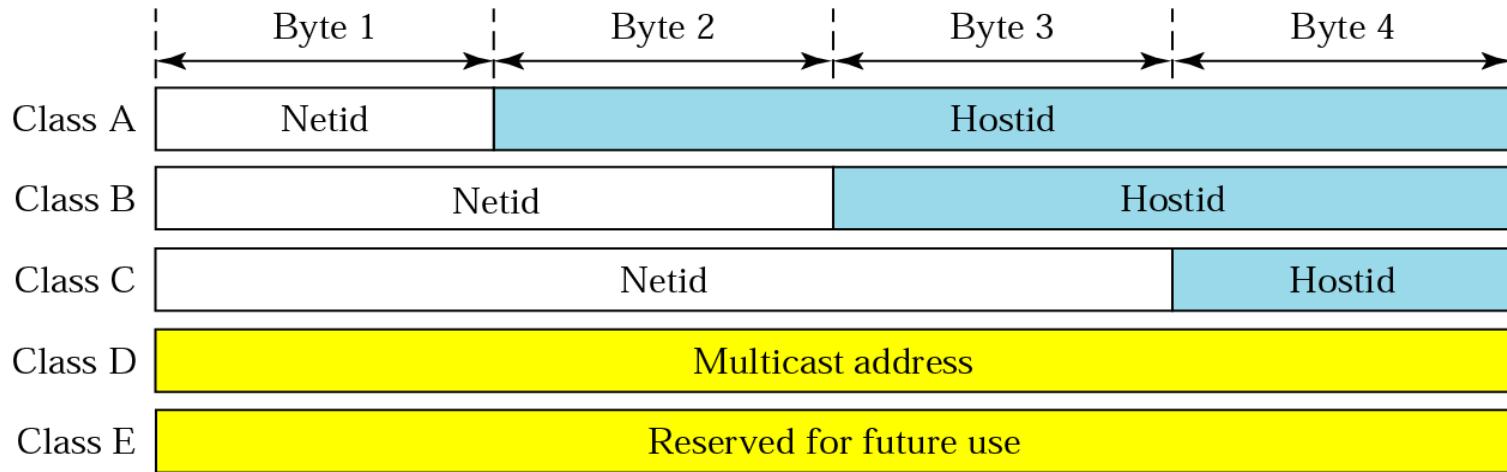
Network	Destination	Netmask	Gateway	Interface	Metric
	0.0.0.0	0.0.0.0	192.168.192.1	192.168.192.37	25
	127.0.0.0	255.0.0.0	On-link	127.0.0.1	306
	127.0.0.1	255.255.255.255	On-link	127.0.0.1	306
127.255.255.255	255.255.255.255		On-link	127.0.0.1	306
	192.168.21.0	255.255.255.0	On-link	192.168.21.1	276
	192.168.21.1	255.255.255.255	On-link	192.168.21.1	276
	192.168.21.255	255.255.255.255	On-link	192.168.21.1	276
	192.168.111.0	255.255.255.0	On-link	192.168.111.1	276
	192.168.111.1	255.255.255.255	On-link	192.168.111.1	276
192.168.111.255	255.255.255.255		On-link	192.168.111.1	276
	192.168.150.0	255.255.255.0	On-link	192.168.150.1	276
	192.168.150.1	255.255.255.255	On-link	192.168.150.1	276
192.168.150.255	255.255.255.255		On-link	192.168.150.1	276
	192.168.192.0	255.255.255.0	On-link	192.168.192.37	281
	192.168.192.37	255.255.255.255	On-link	192.168.192.37	281
192.168.192.255	255.255.255.255		On-link	192.168.192.37	281



4.294.967.296 Possible Nodes???



Network ID and Host ID

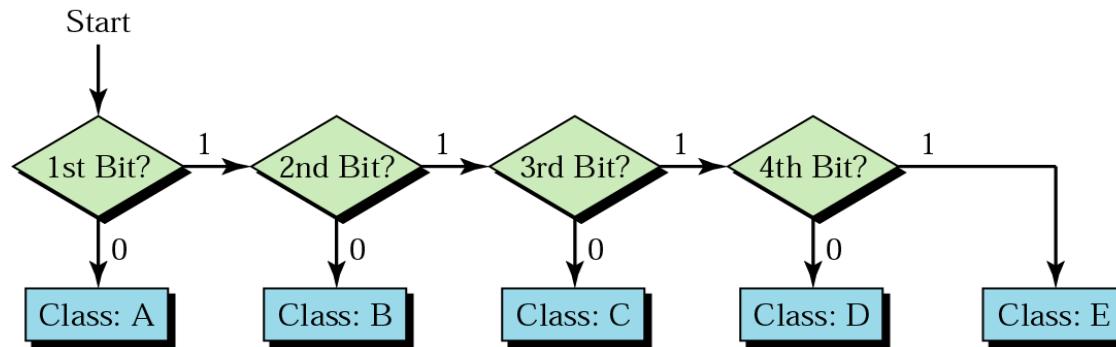


- Network ID: Used to find a particular network
- Host ID: Identifies individual nodes

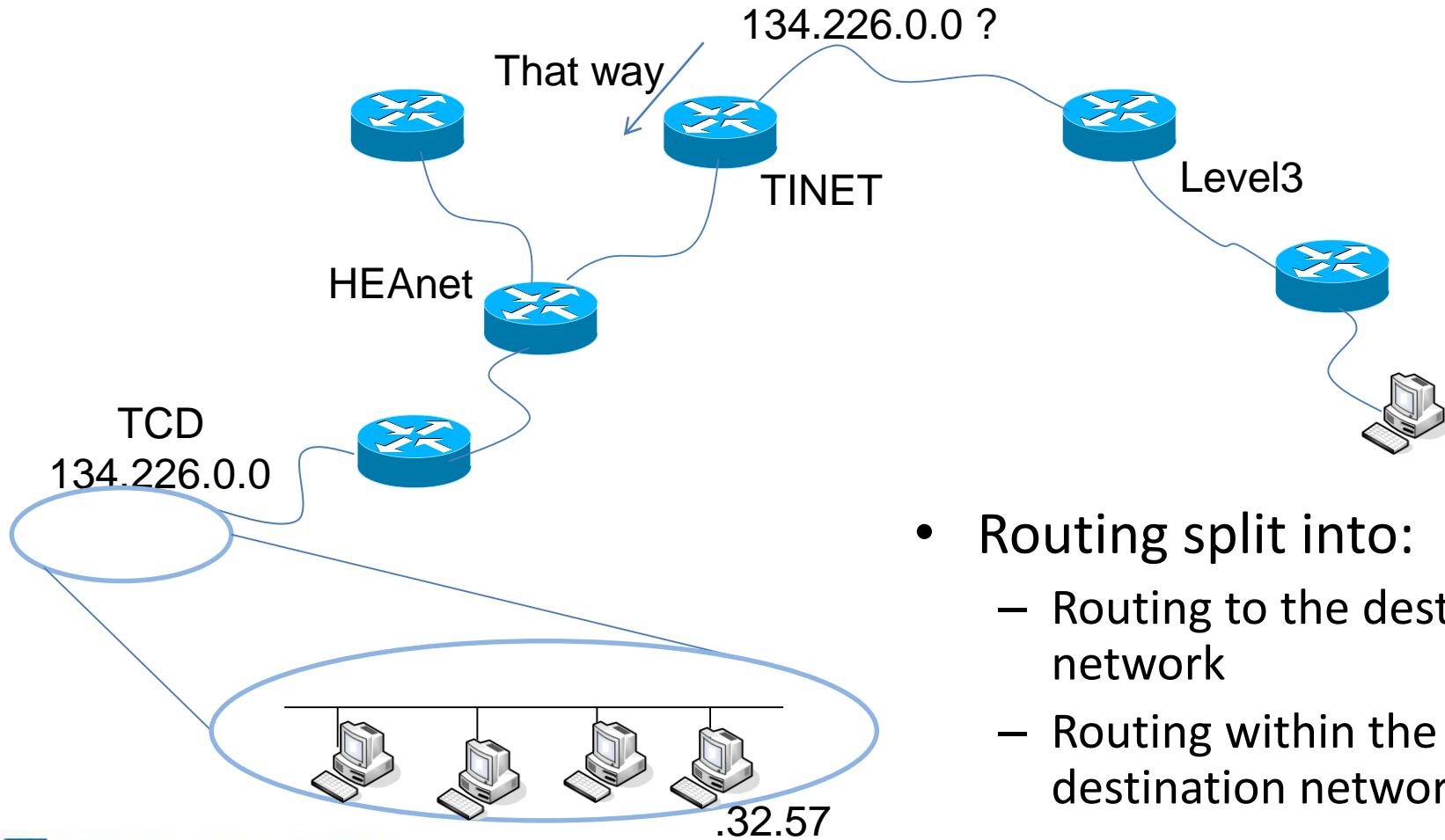
Classes in Binary Notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

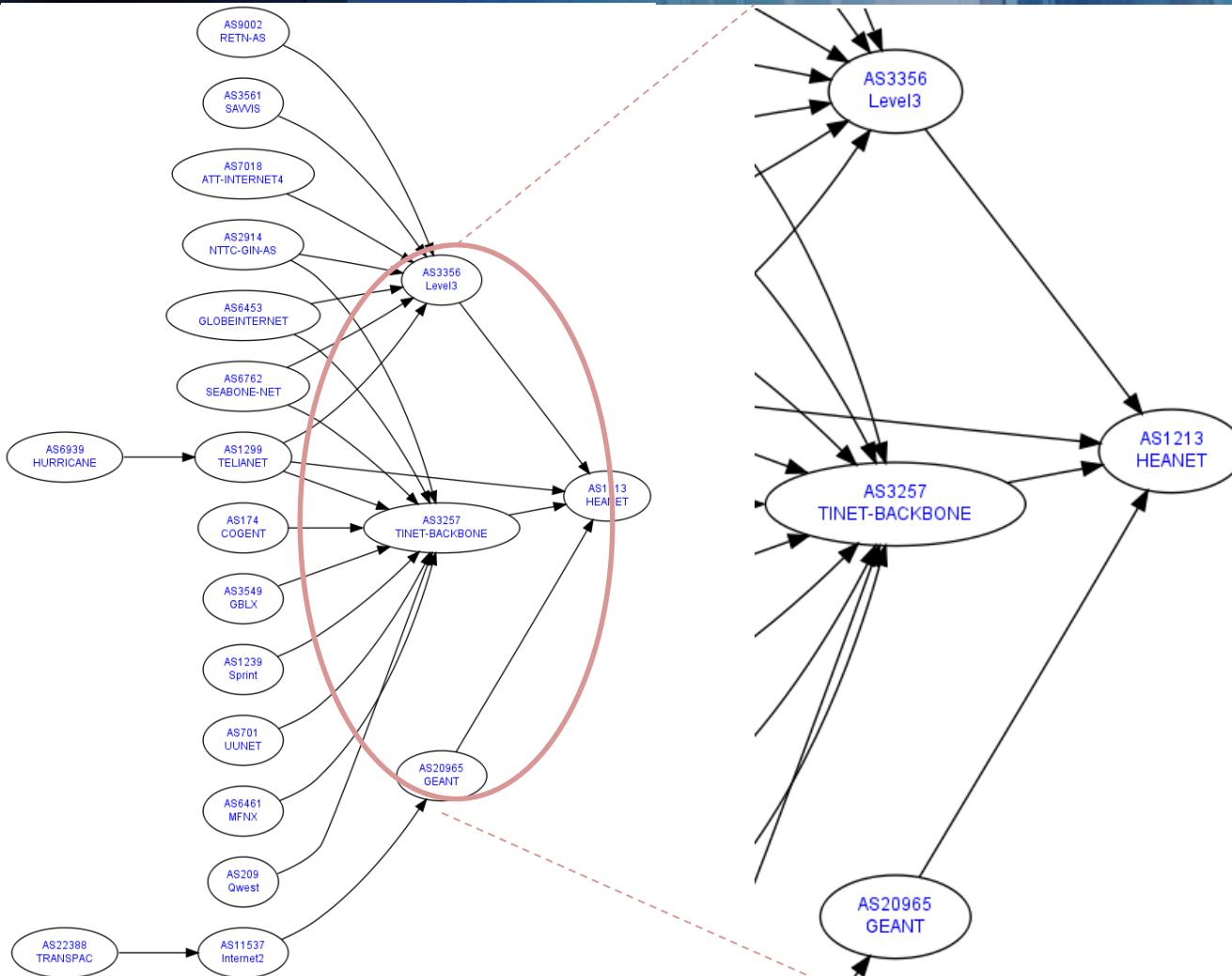
- Decision Process:



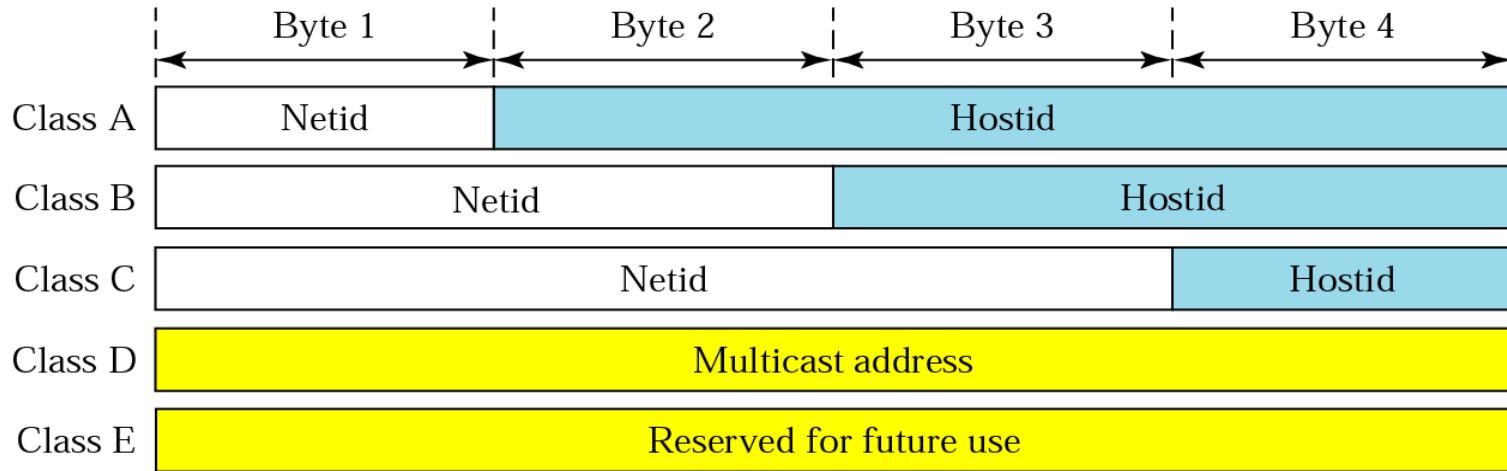
Network IDs and Host IDs



AS1213 - HEANET

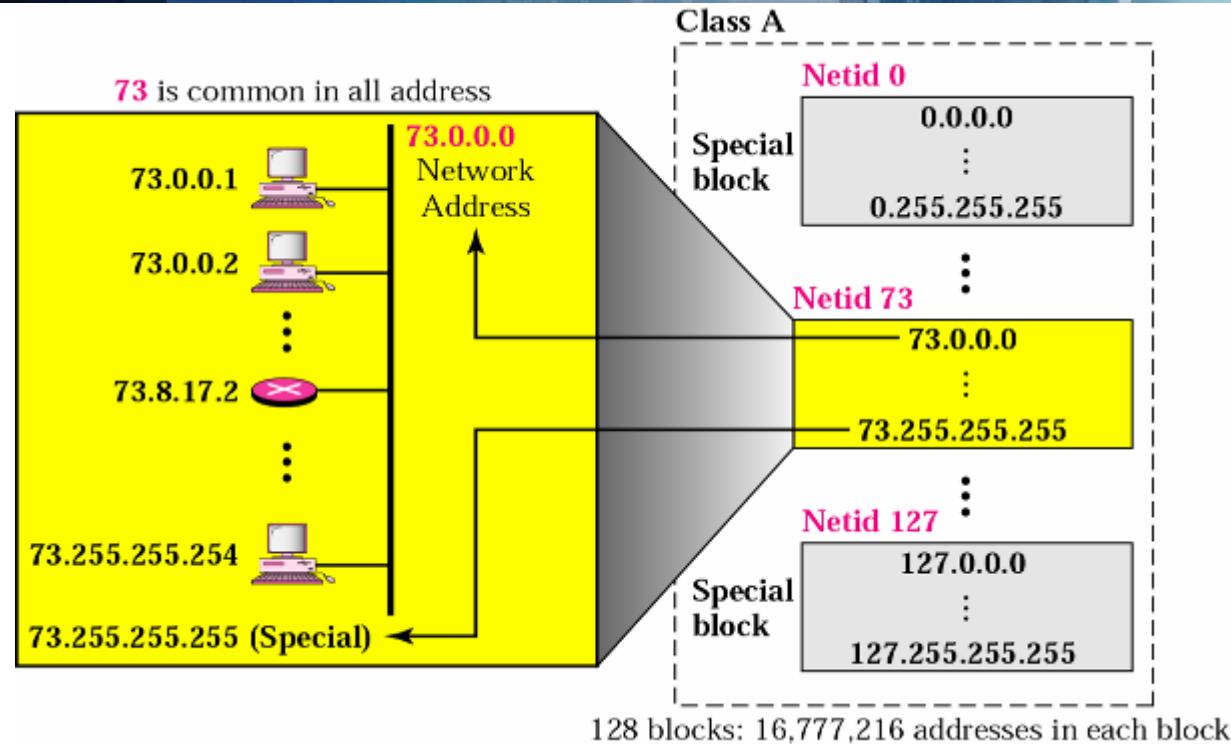


Classful Addresses



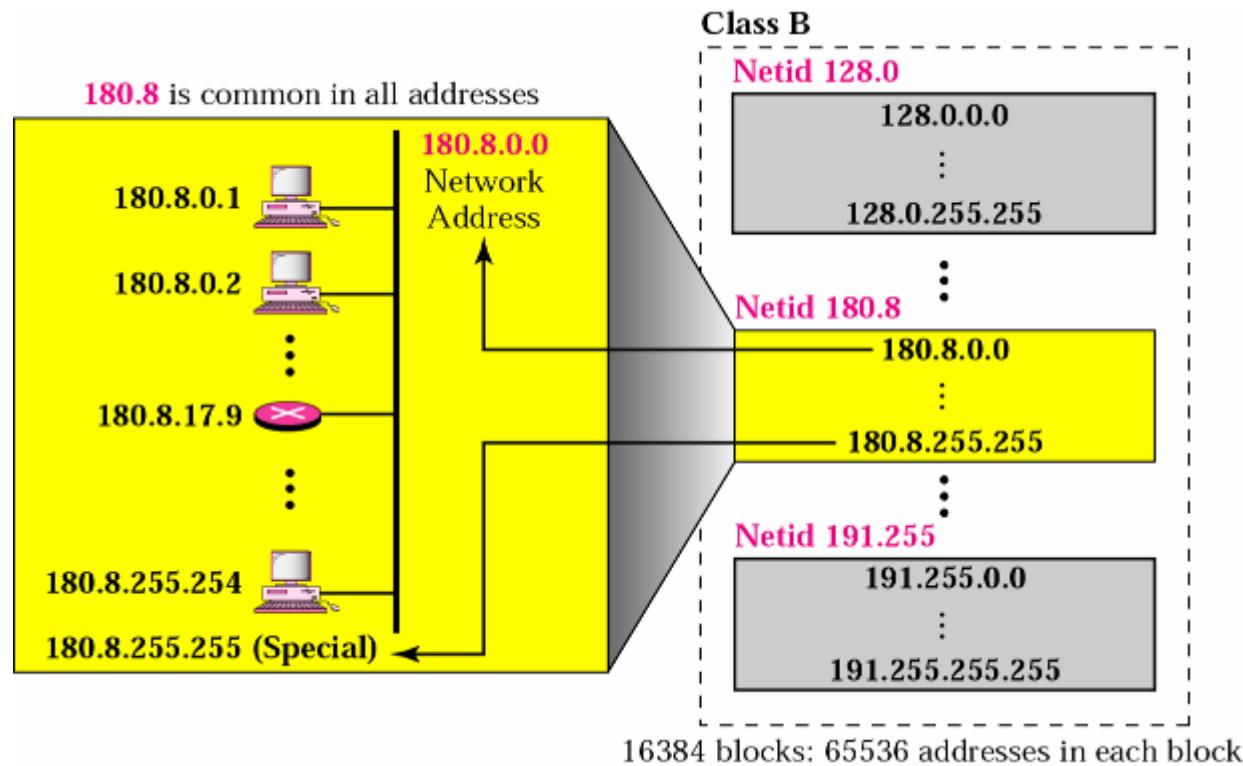
- Class A (international organisations)
 - 126 networks with 16,277,214 hosts each
- Class B (large companies)
 - 16,384 networks with 65,354 hosts each
- Class C (smaller companies)
 - 2,097,152 networks with 254 hosts each

Class A



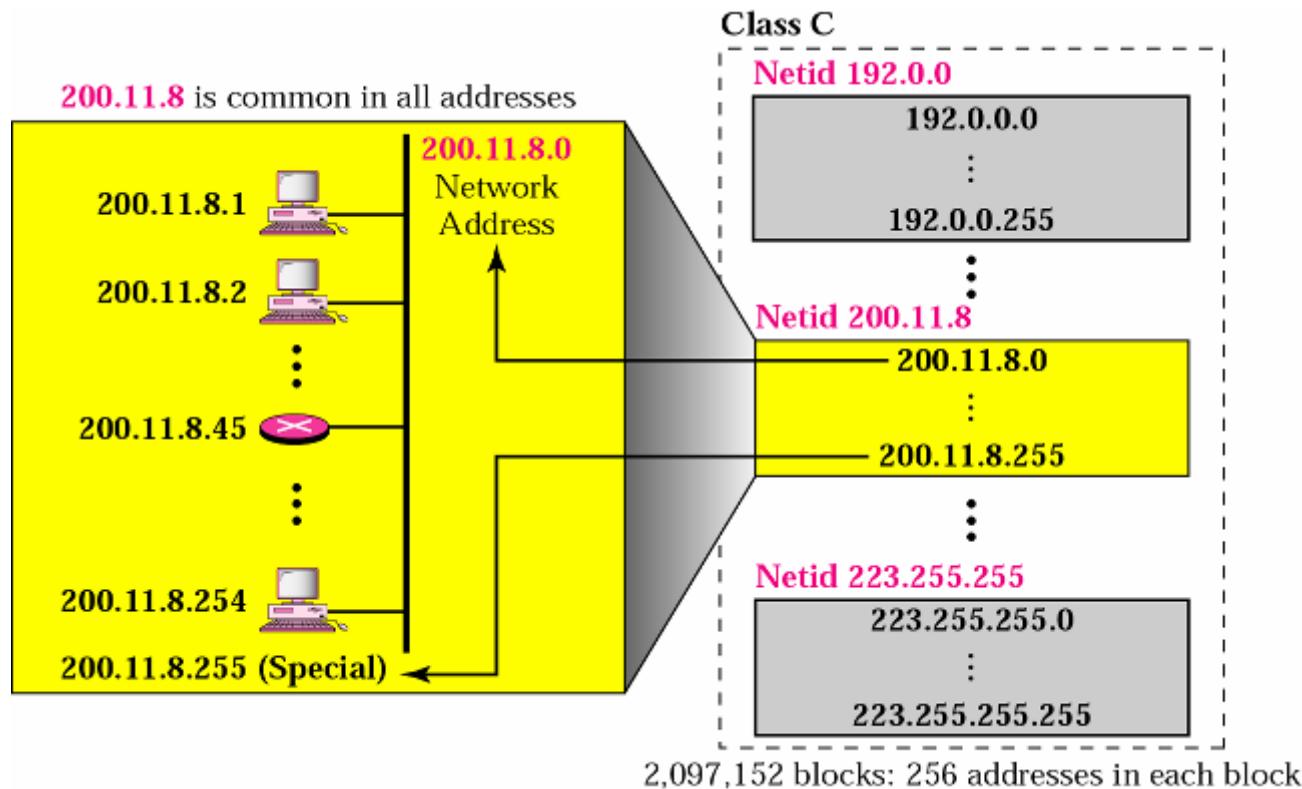
- Limited number of very large networks
- 126 networks with 16,277,214 hosts each

Class B



- Limited number of relatively large address ranges
- 16,384 networks with 65,354 hosts each

Class C



- Large number of small address ranges
- 2,097,152 networks with 254 hosts each

Classes & Private Addresses

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

Private Address Ranges:

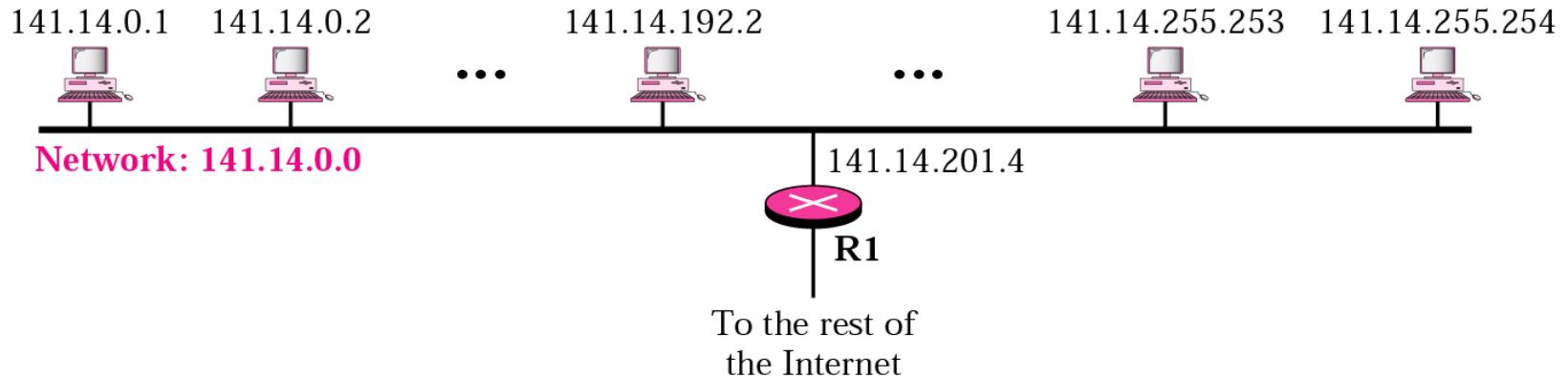
Range	Total
10.0.0.0 to 10.255.255.255	2^{24}
172.16.0.0 to 172.31.255.255	2^{20}
192.168.0.0 to 192.168.255.255	2^{16}

Special Addresses

- Loopback device: 127.x.x.x
 - e.g. 127.0.0.1 = localhost
- This network: 0.0.0.x (all zero's)
 - e.g. 0.0.0.54 = host 54 on this network
- Broadcast: x.x.255.255 (all one's)
 - e.g. 134.226.36.255 = all nodes in this network



2-Level Hierarchy with Classful Addresses



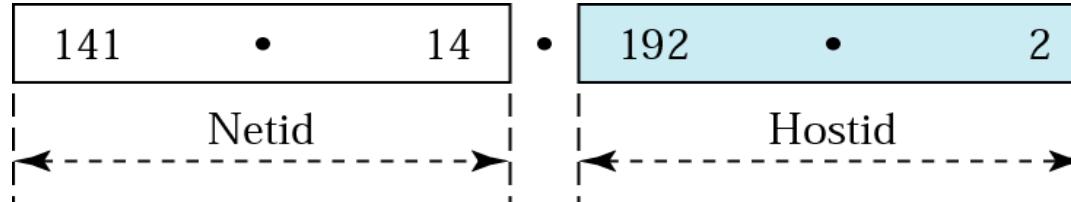
- 1st level: NetworkID within the Internet
- 2nd level: HostID within the network

College Network

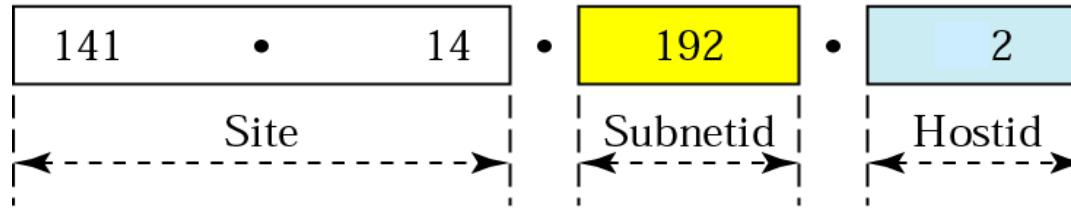
- Flat Network with 1000s of Nodes



Subnetting



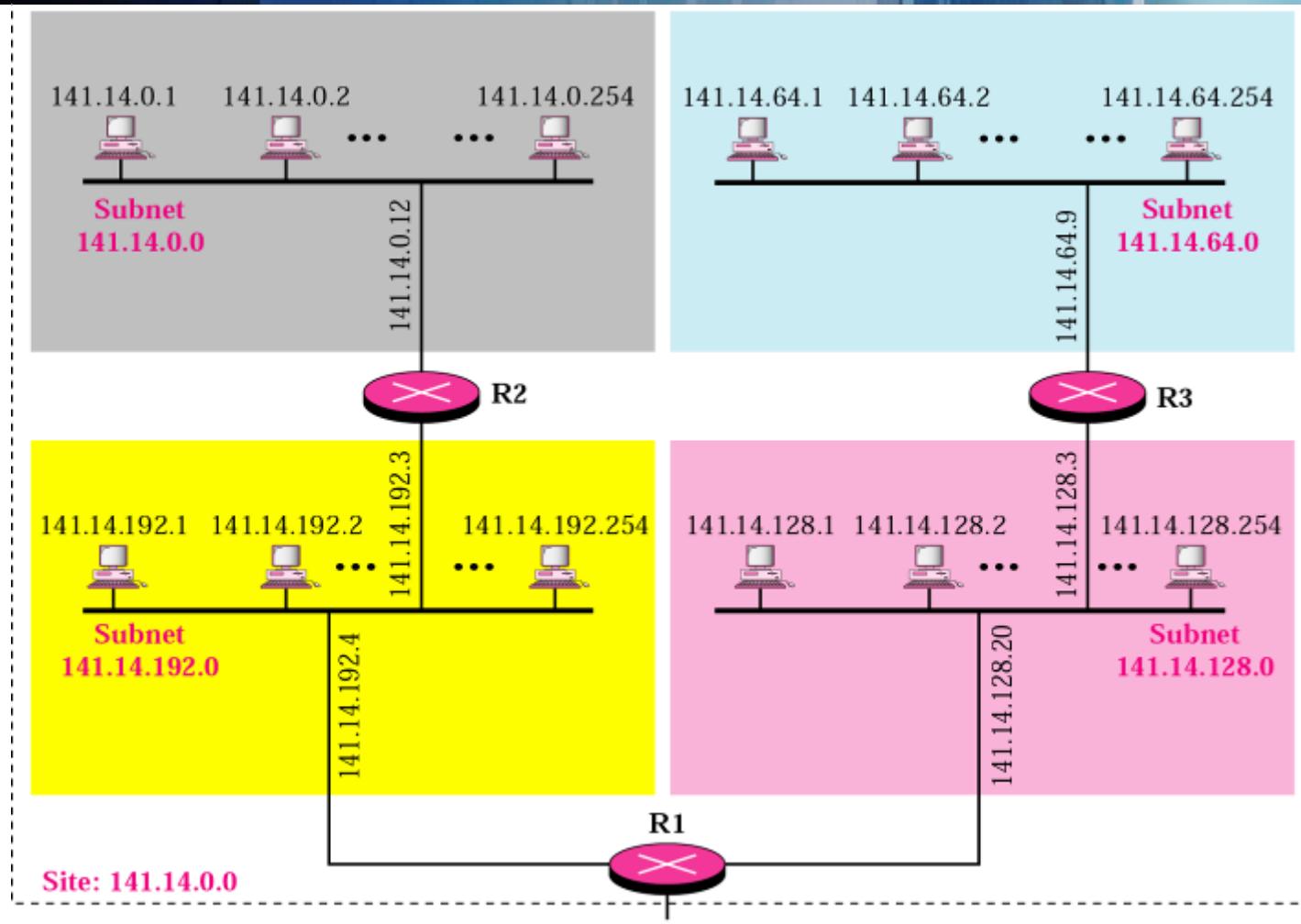
a. Without subnetting



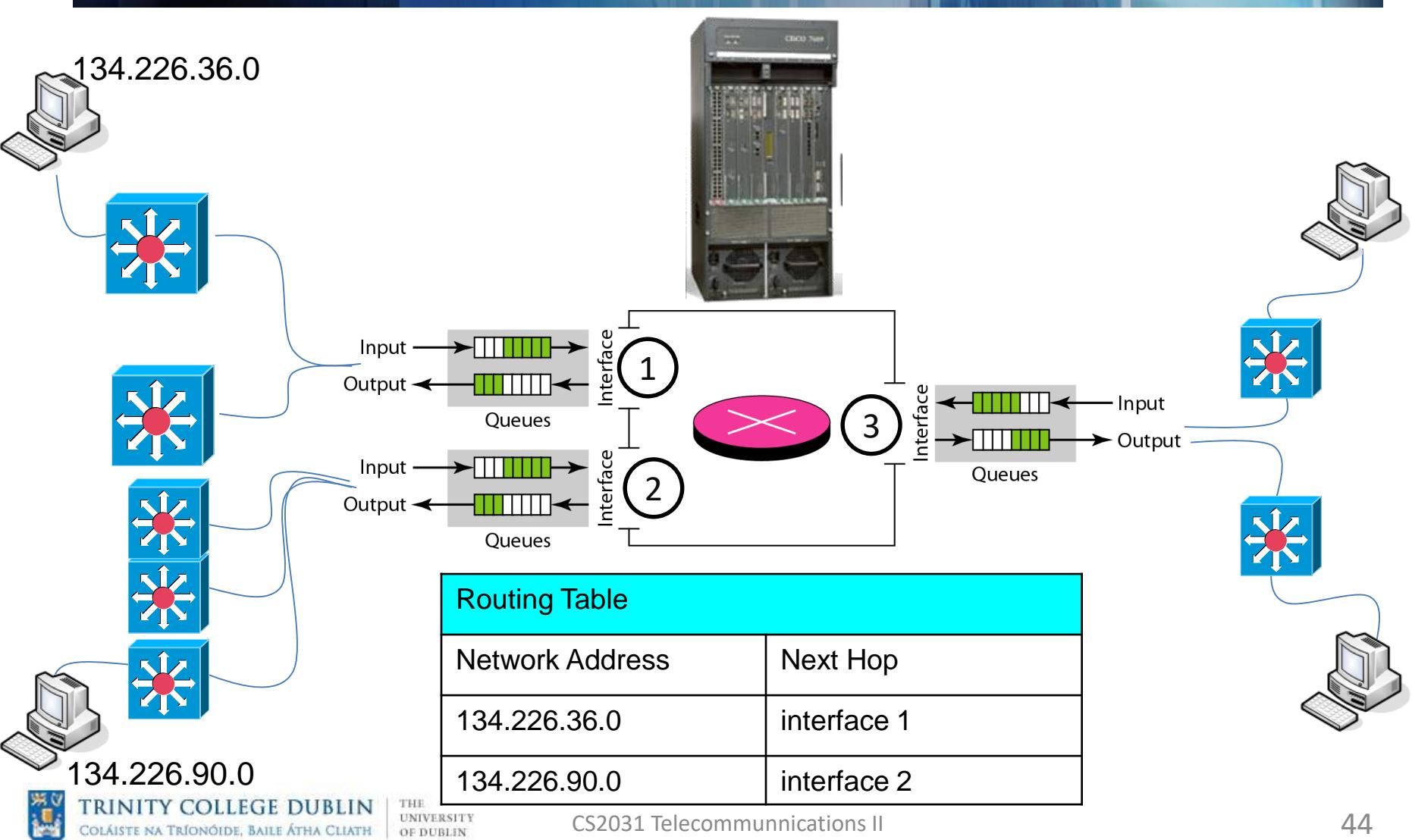
b. With subnetting

- Add another level to address hierarchy: *subnet*
- Splitting a class B network into a number of class C subnets

3-Level Hierarchy through Subnetting

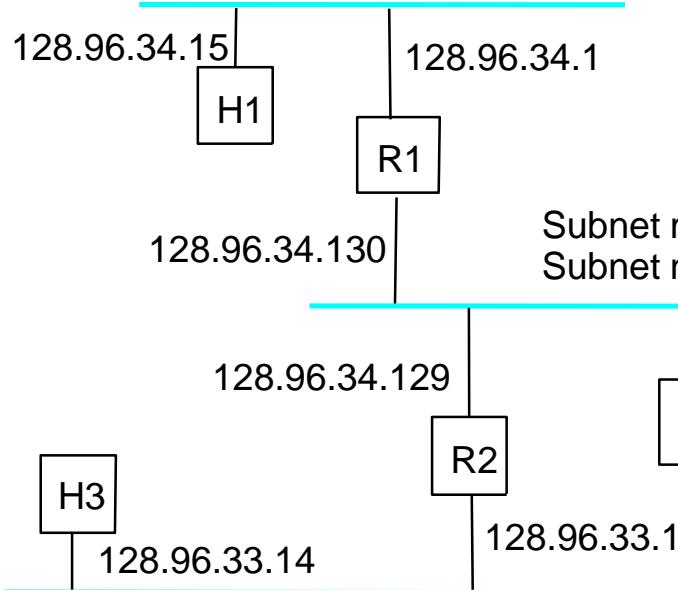


Task of Routers



Subnet Example

Subnet mask: 255.255.255.128
 Subnet number: 128.96.34.0



Subnet mask: 255.255.255.128
 Subnet number: 128.96.34.128

Forwarding table at router R1

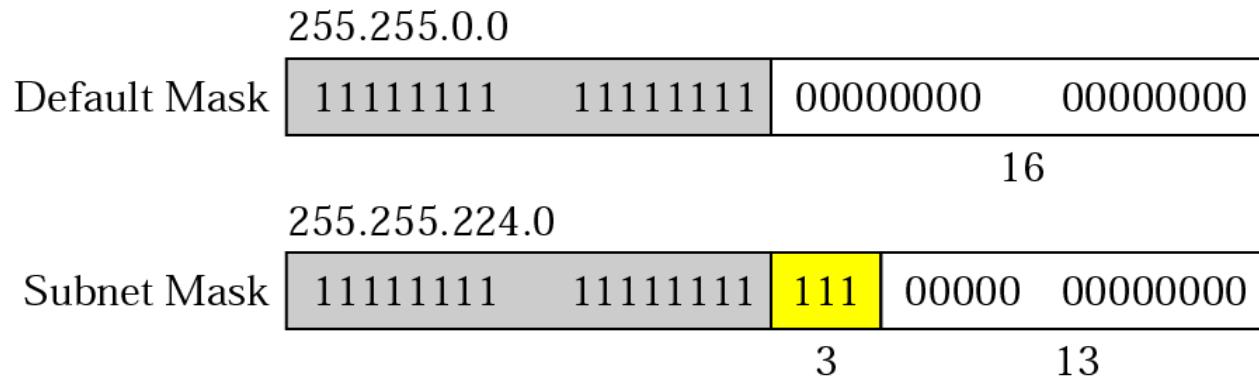
Subnet mask: 255.255.255.0
 Subnet number: 128.96.33.0

Subnet Number	Subnet Mask	Next Hop
128.96.34.0	255.255.255.128	interface 0
128.96.34.128	255.255.255.128	interface 1
128.96.33.0	255.255.255.0	R2

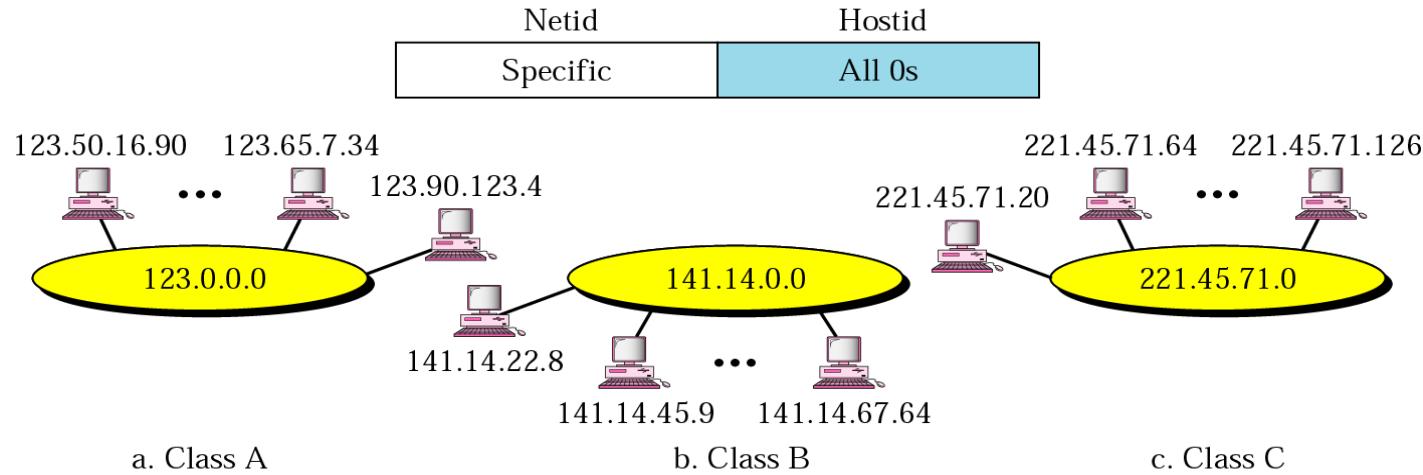
Default Masks

Class	In Binary	Dotted-Decimal	Using Slash
A	11111111 00000000 00000000 00000000	255.0.0.0	/8
B	11111111 11111111 00000000 00000000	255.255.0.0	/16
C	11111111 11111111 11111111 00000000	255.255.255.0	/24

- Subnet Masks



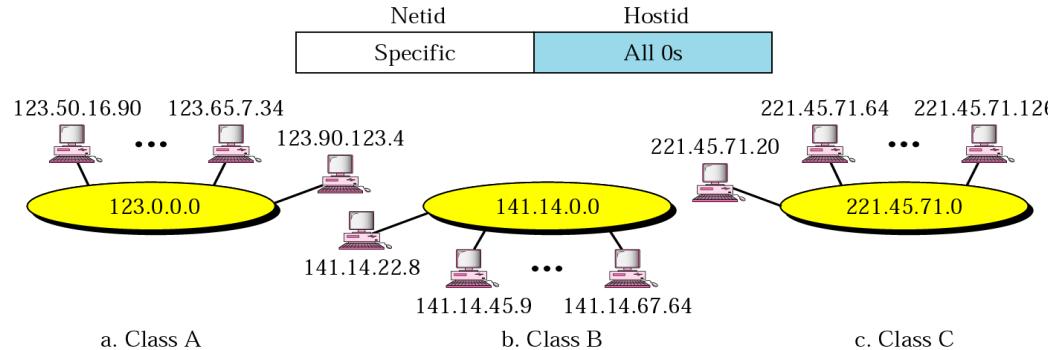
Classful Addresses & NetworkID



- Classful Addresses:

Class	Networks	Addresses
A	126	16,777,214
B	16,382	65,534
C	2,097,152	254

Inefficiency of Classful Addresses



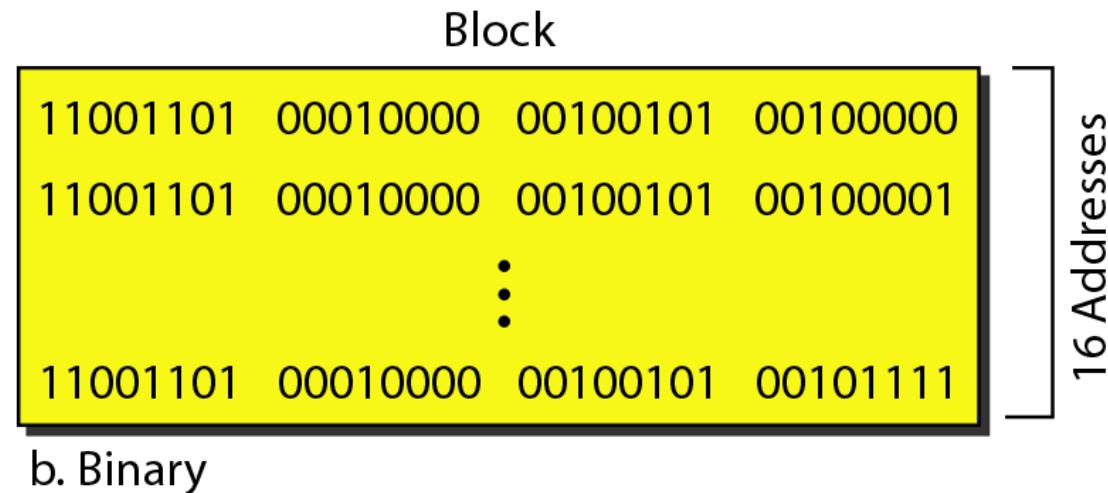
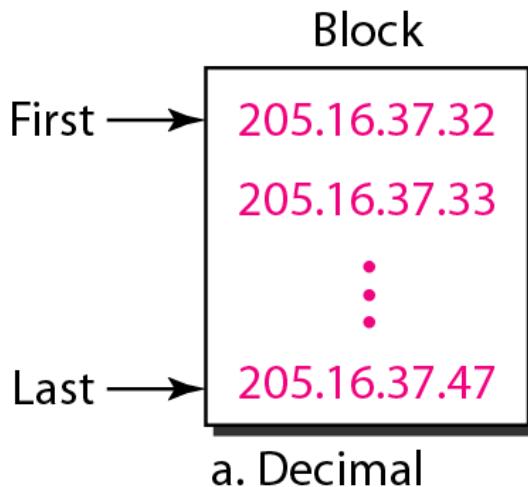
- Classful Addresses:

Class	Networks	Addresses
A	126	16,777,214
B	16,382	65,534
C	2,097,152	254

- Inefficient use of Hierarchical Address Space
 - Class C with 2 hosts ($2/254 = 0.78\%$ efficient)
 - Class B with 256 hosts ($256/65534 = 0.39\%$ efficient)

Classless Inter-Domain Routing(CIDR)

- Allow address space to be divided into blocks of addresses
 - only limited to the power of 2
- Notation as decimal number of the significant bits e.g.
134.226.36.0 /29
- 205.16.37.32/28
 - 32 bits – 28 bits are static - 4 bits are varied



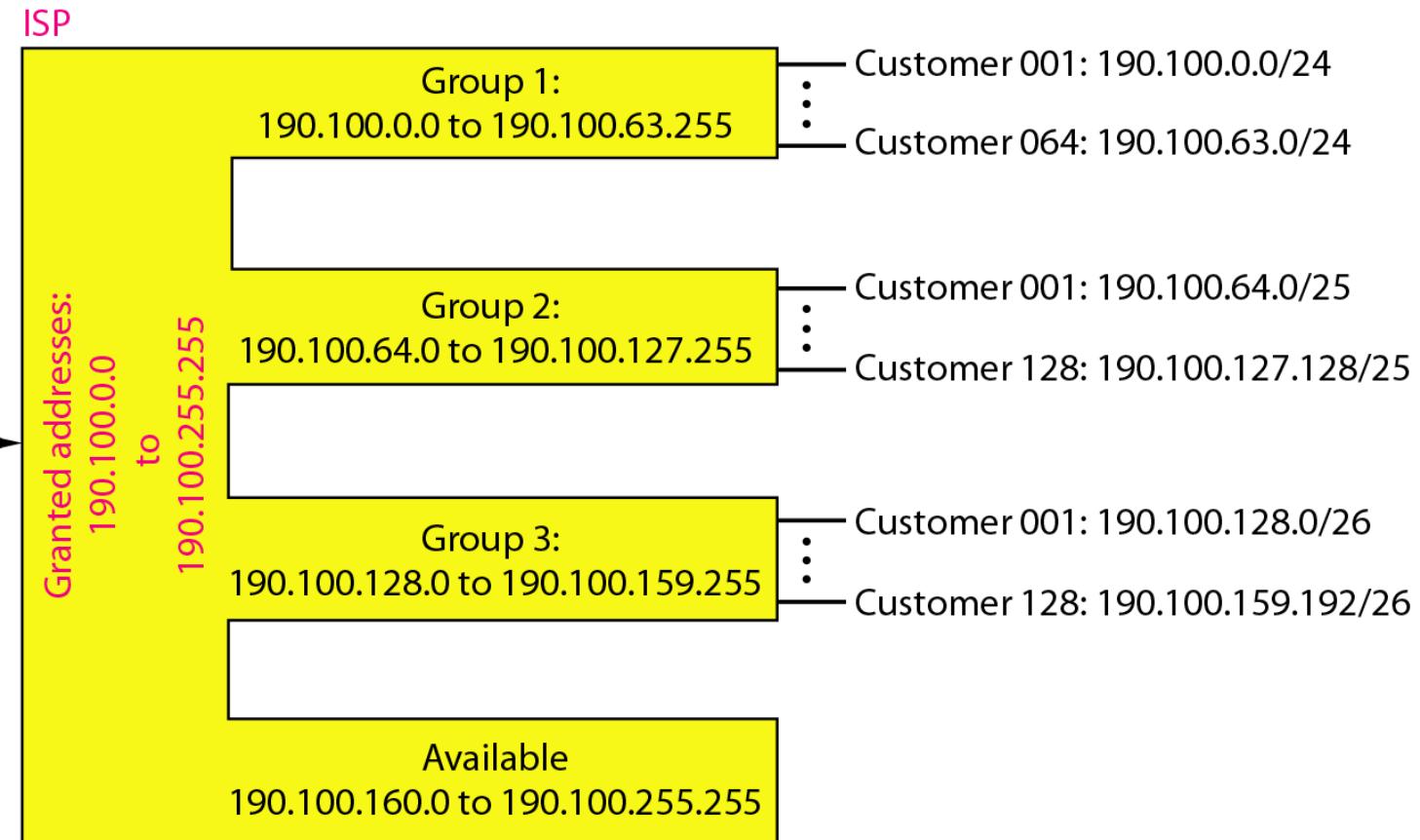
Classless Inter-Domain Routing(CIDR)

	Dotted Decimal	32-bit binary equivalent
Lowest	128.211.168.0	10000000 11010011 10101 000 00000000
Highest	128.211.175.255	10000000 11010011 10101 111 11111111

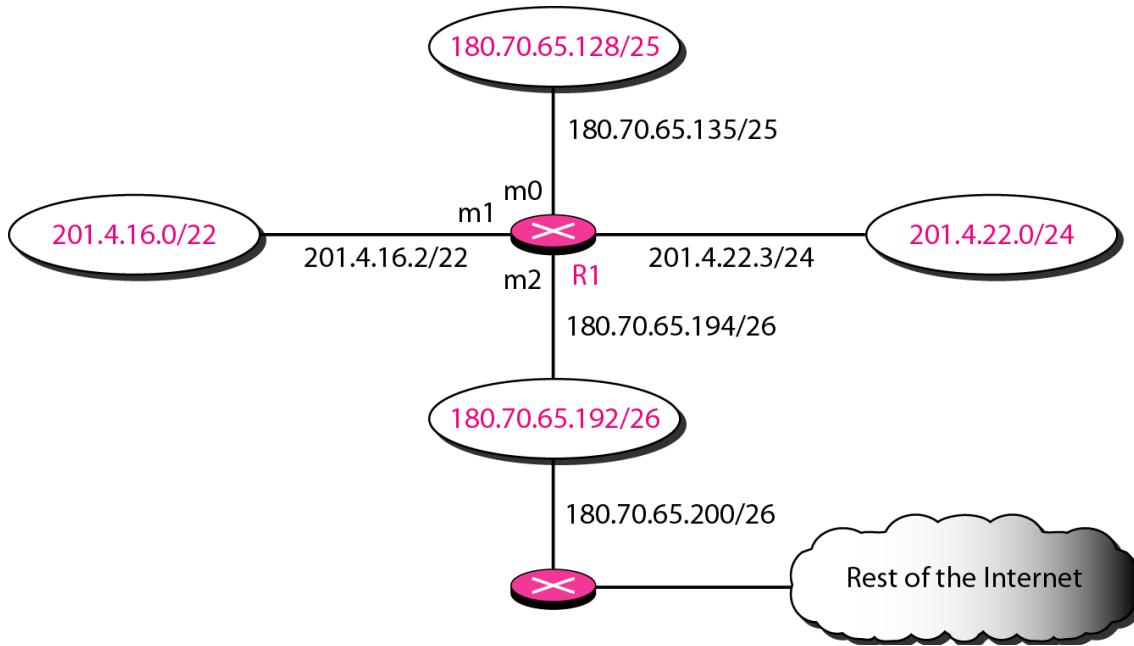
= 128.211.168.0/21

- Aggregation: For example, class C networks can be combined to larger networks
- /21 = 8 class C = 8 * 256 addresses = 2048 addresses

ISPs & Classless Addresses

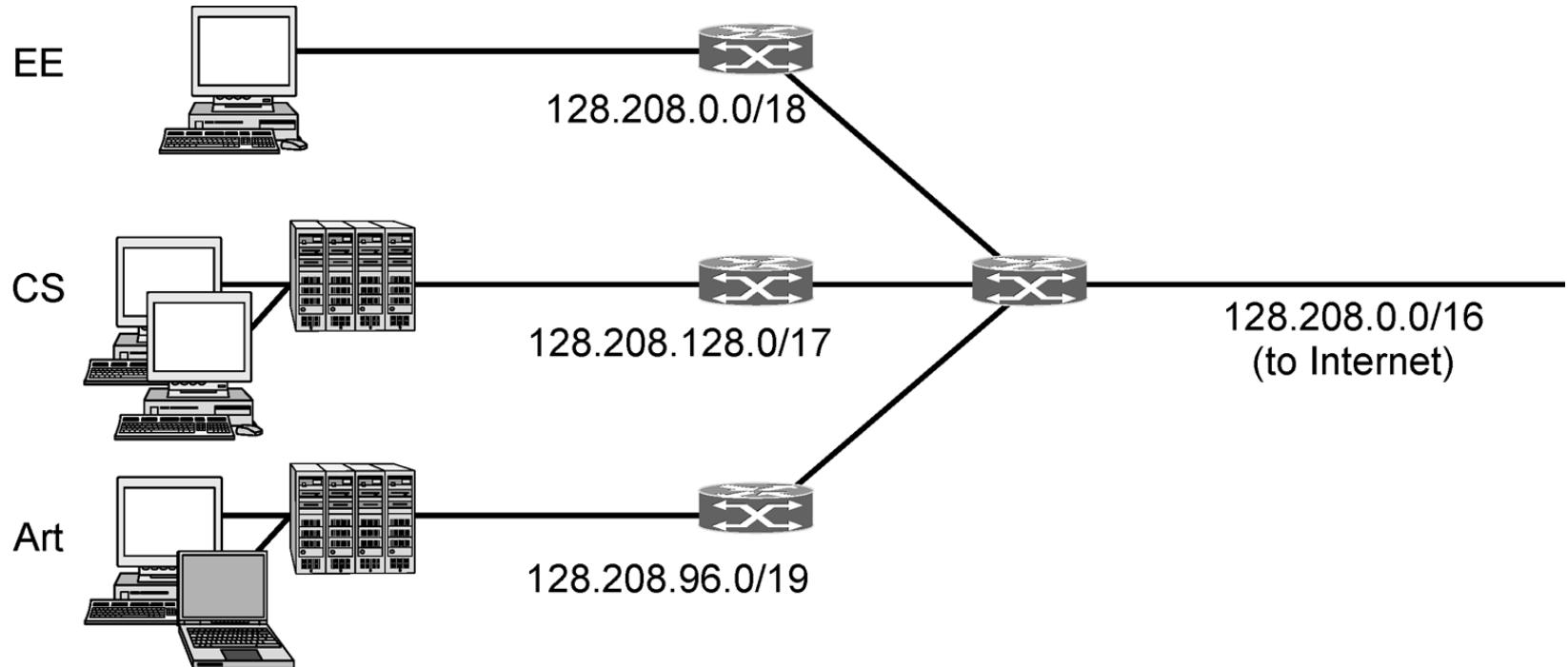


CIDR & Router

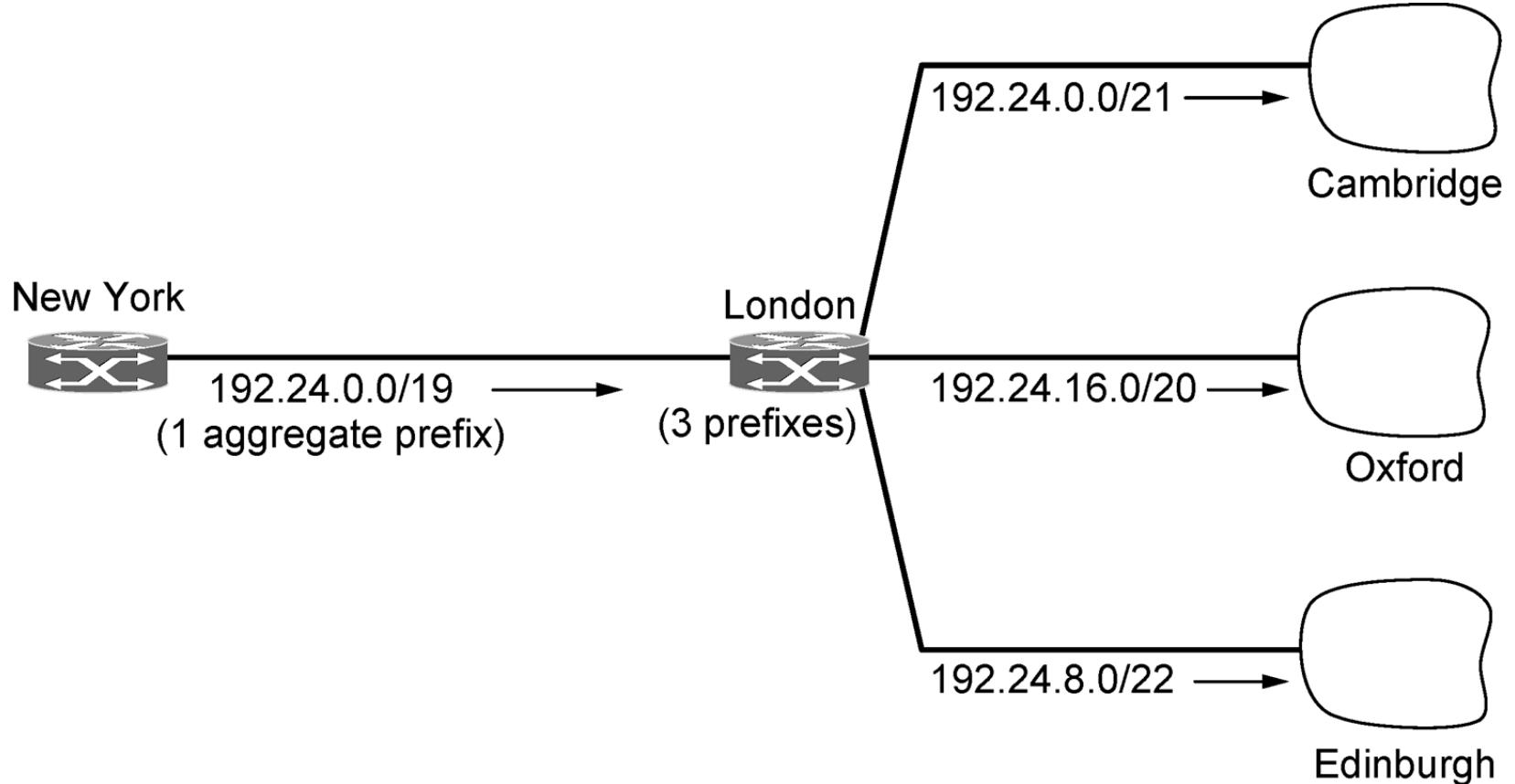


Mask	Network Address	Next Hop	Interface
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	m1
Any	Any	180.70.65.200	m2

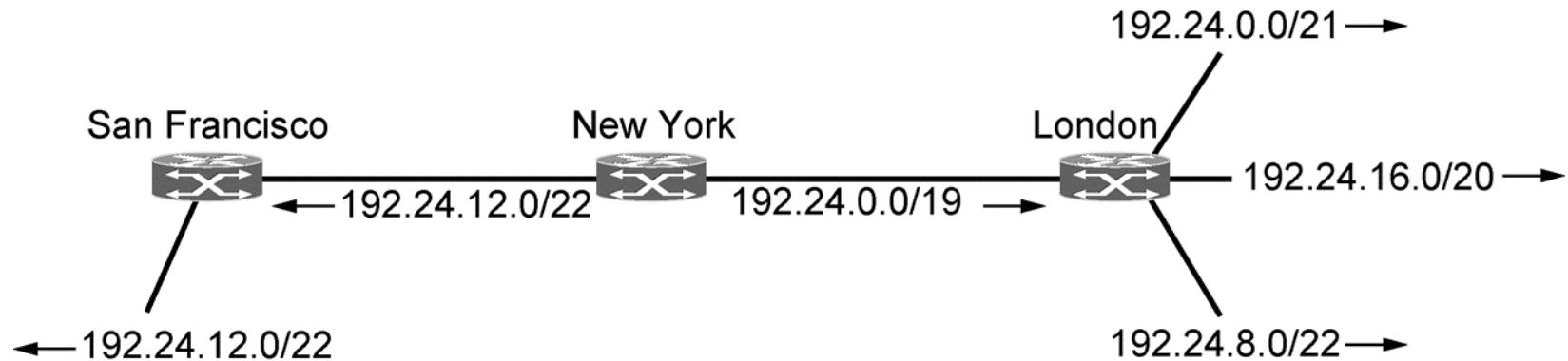
IP Prefix & Subnets



Aggregate Prefix

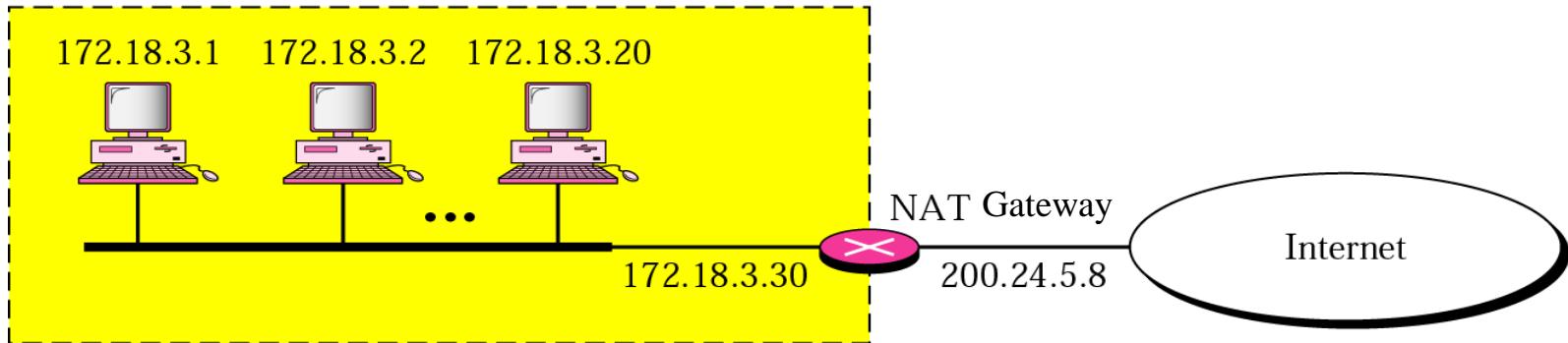


Longest matching Prefix



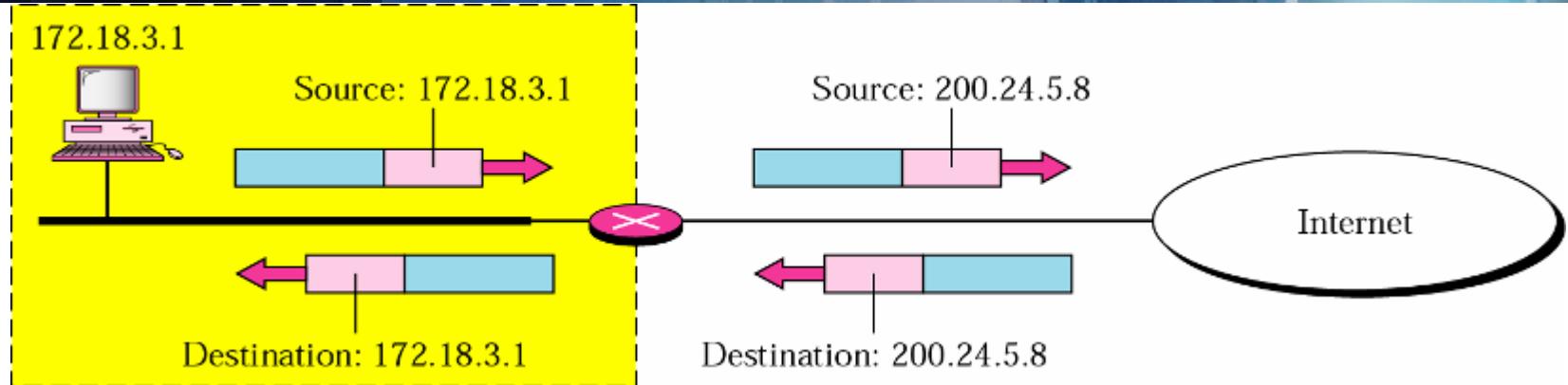
Network Address Translation (NAT)

Site using private addresses



- NAT gateway translates traffic from the local network to the IP address of the gateway
- Involves processing of outgoing & incoming packet e.g. translation between addresses, recalculation of checksums, etc

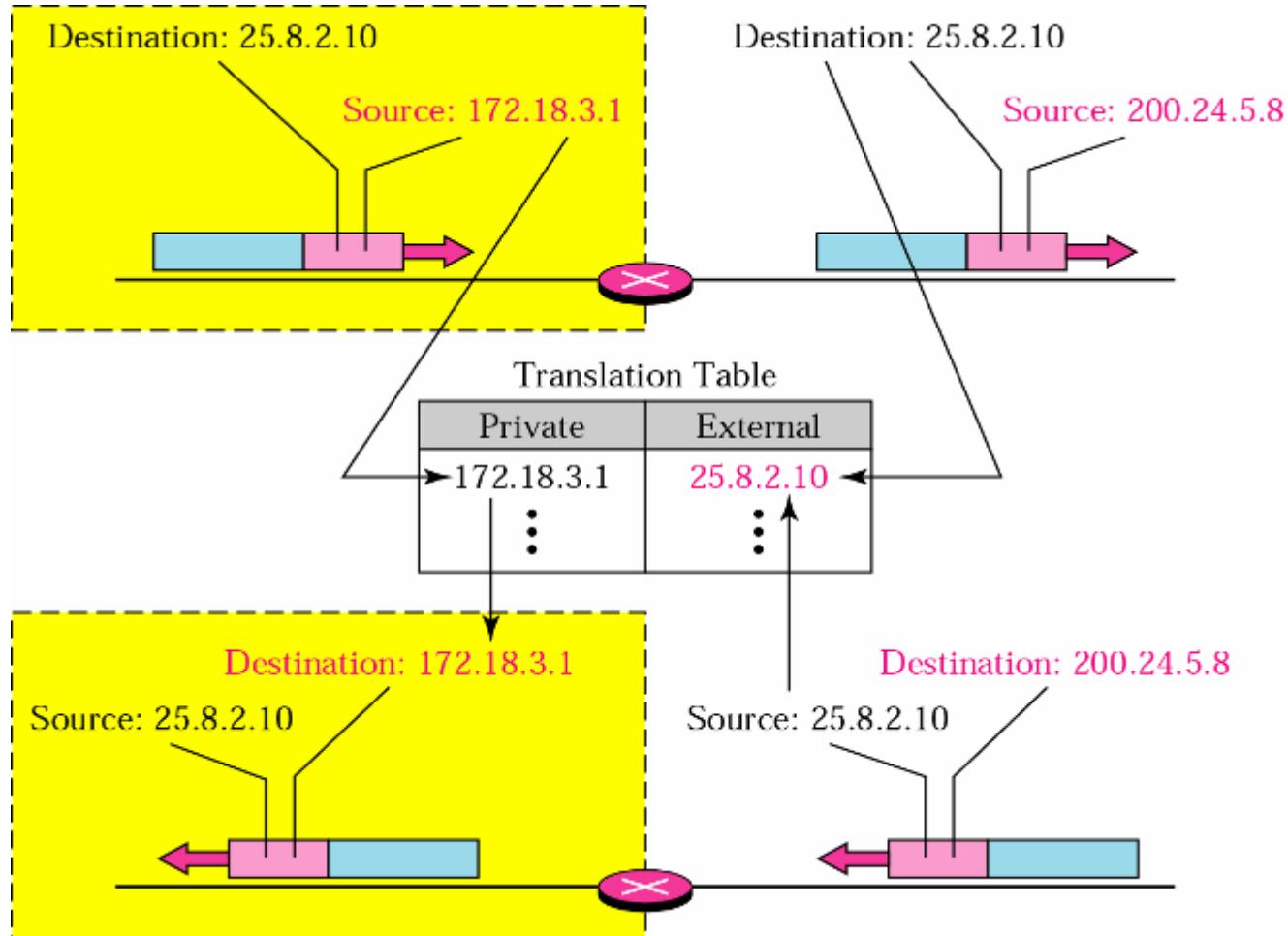
Address Translation



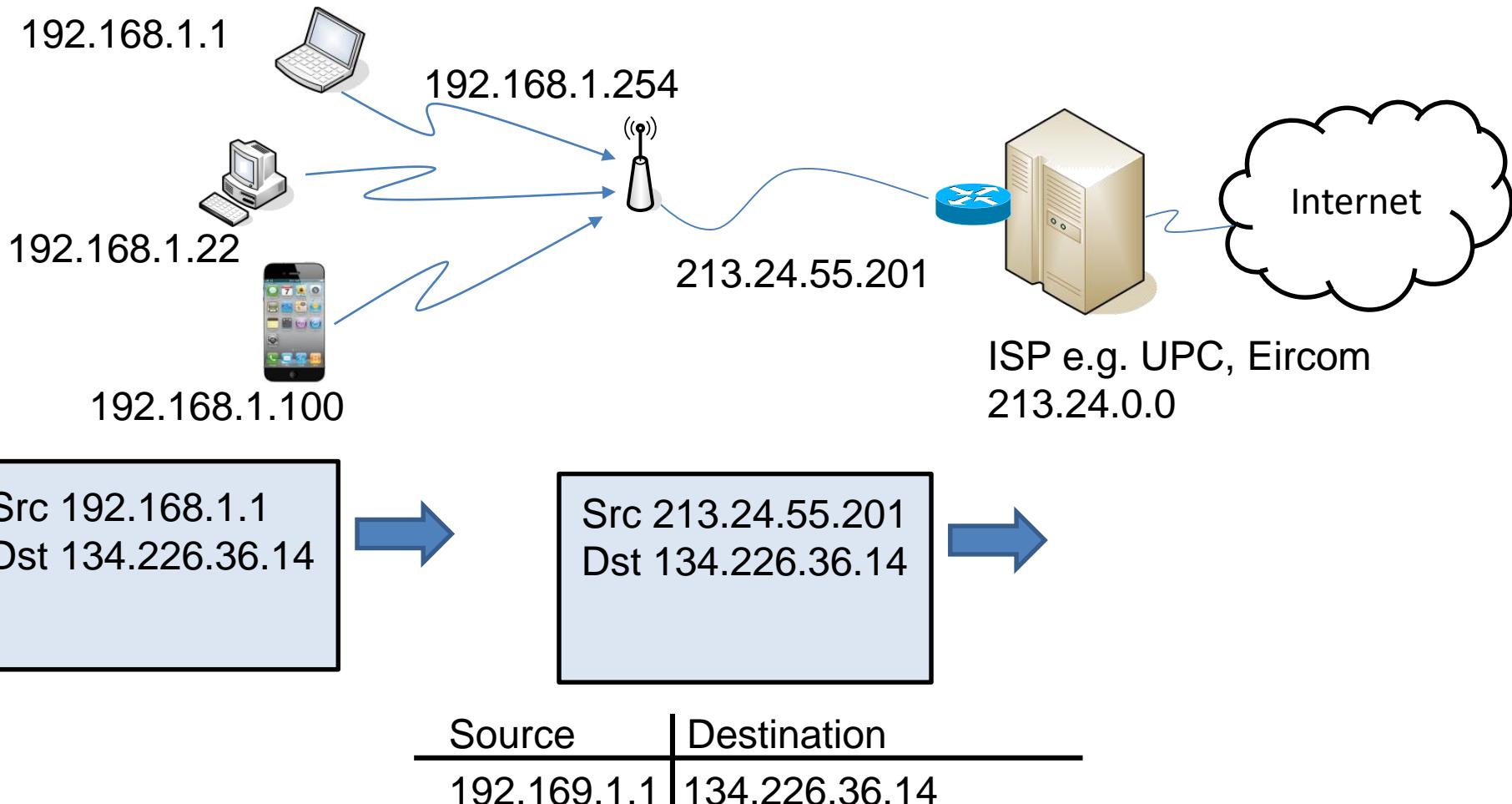
Private Address	Private Port	External Address	External Port	Transport Protocol
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
...

- Gateway maintains table to match incoming and outgoing packets; including IDs for applications

Example for NAT



NAT Example

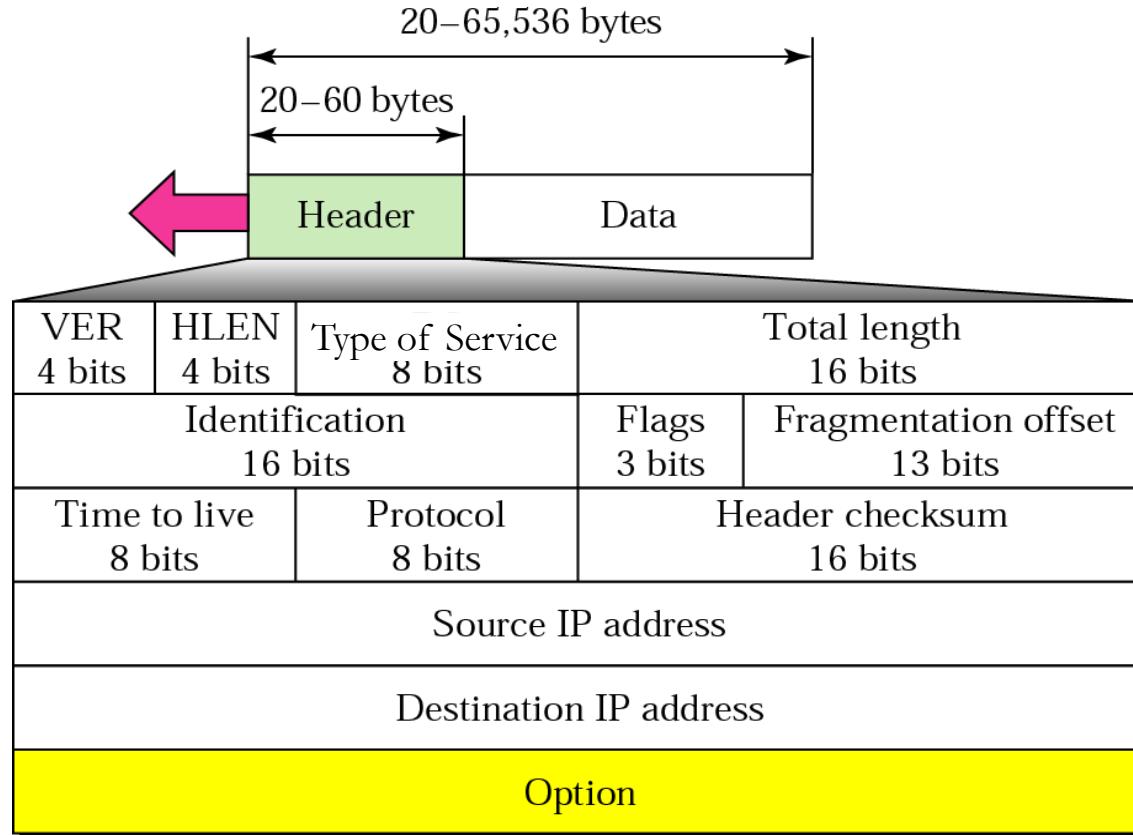


Summary: Addresses

- 32-bit number / Dotted decimal notation
- IP addresses are unique and universal
 - Exception: Private Addresses
- Classful addresses
 - Classes A, B, and C for networks, D for multicast
 - Routing on Network IDs
- Subnetting + Netmasks
 - Dealing with scale in local networks
- Classless Inter-Domain Routing (CIDR)
 - / notation – significant bits of address
- Network address translation (NAT)

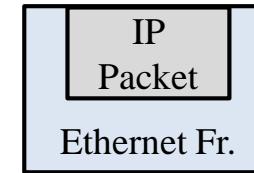
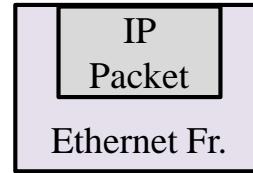
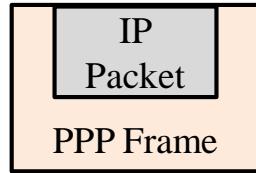
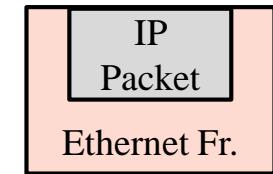
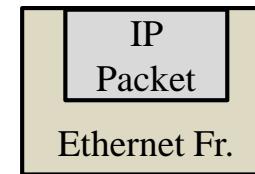
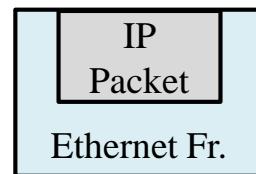
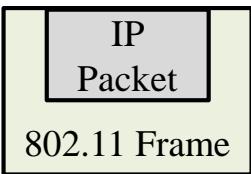
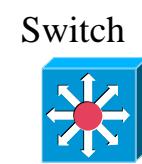
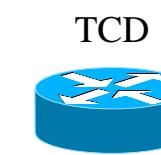


IP Datagram

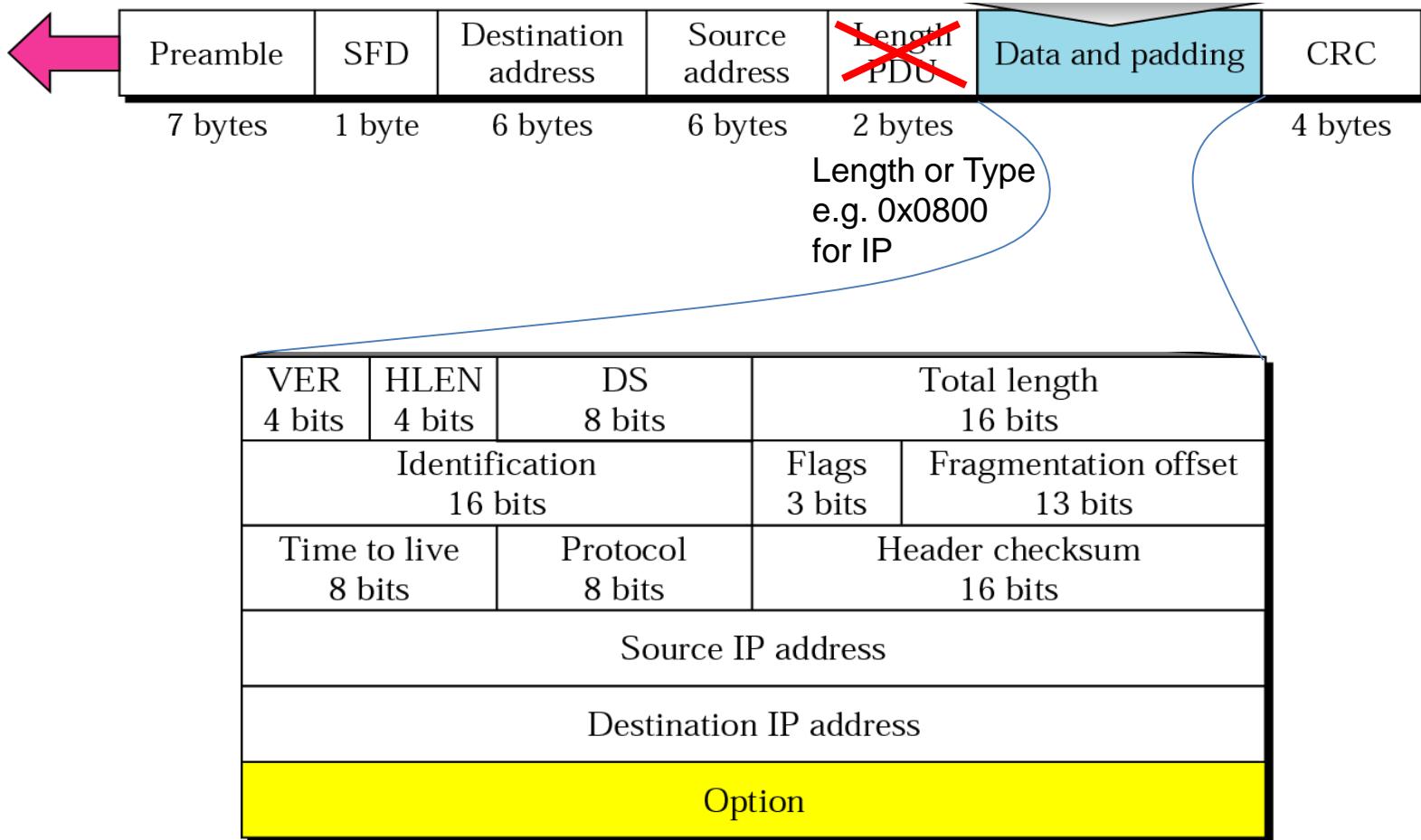


- The total length field defines the total length of the datagram including the header.

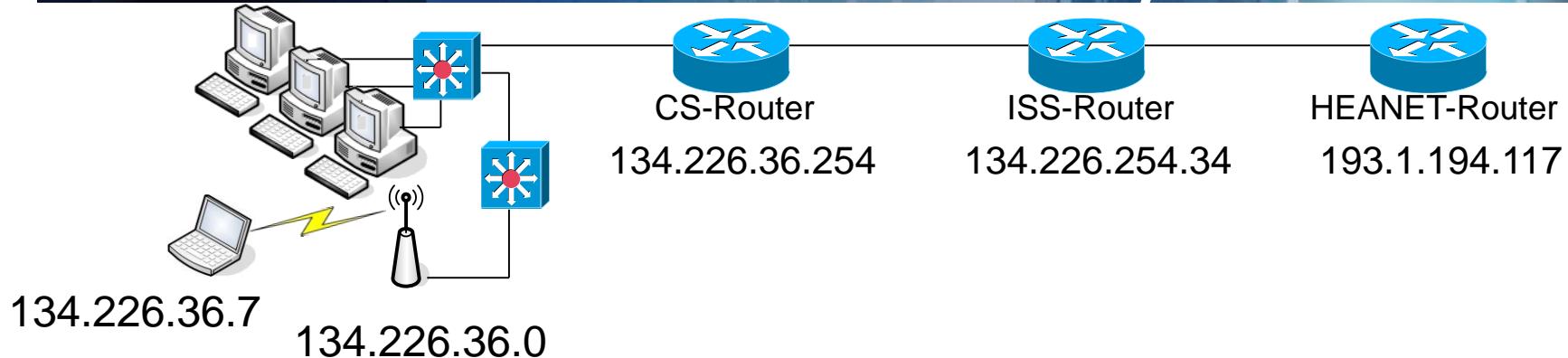
Encapsulation



Ethernet & IP



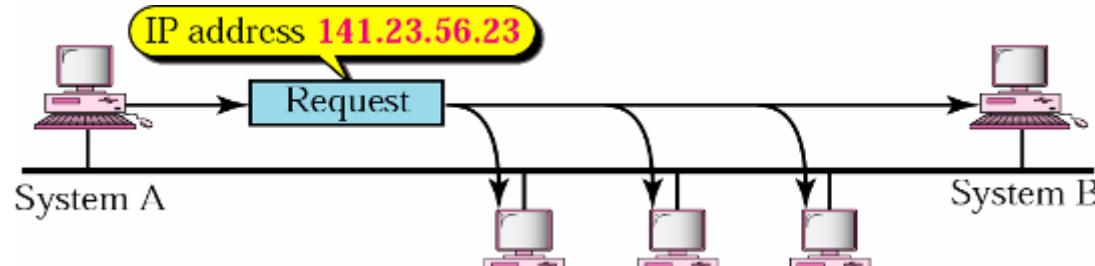
Default Gateway



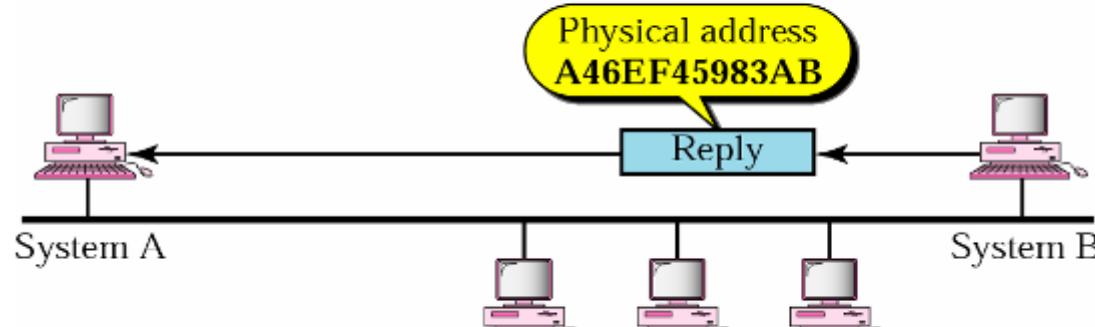
Subnet	Gateway	Netmask	Interface
134.226.36.0	0.0.0.0	255.255.0.0	eth1
0.0.0.0	134.226.36.254	0.0.0.0	eth1

- All nodes within the subnet can communicate directly with each other
- All communication with nodes in other networks passes through the default gateway e.g. router 134.226.36.254

Address Resolution Protocol (ARP)



a. ARP request is broadcast



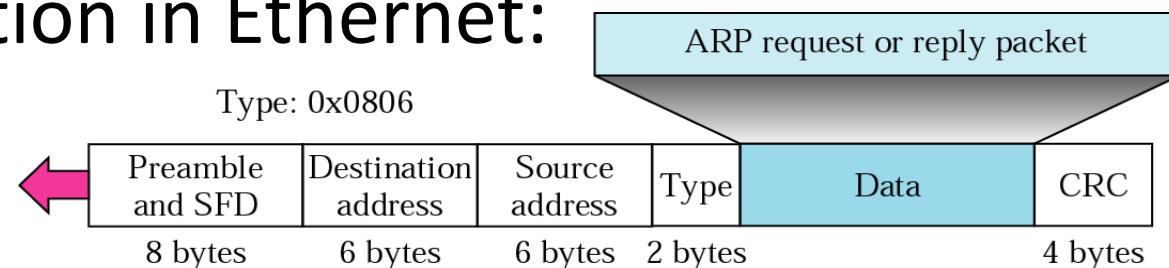
b. ARP reply is unicast

- Association between hardware address and IP address

ARP Packet

Hardware Type		Protocol Type
Hardware length	Protocol length	Operation Request 1, Reply 2
Sender hardware address (For example, 6 bytes for Ethernet)		
Sender protocol address (For example, 4 bytes for IP)		
Target hardware address (For example, 6 bytes for Ethernet) (It is not filled in a request)		
Target protocol address (For example, 4 bytes for IP)		

- Encapsulation in Ethernet:



“Who has”-Packet

```
ARP      Who has 192.168.1.2? Tell 192.168.1.5

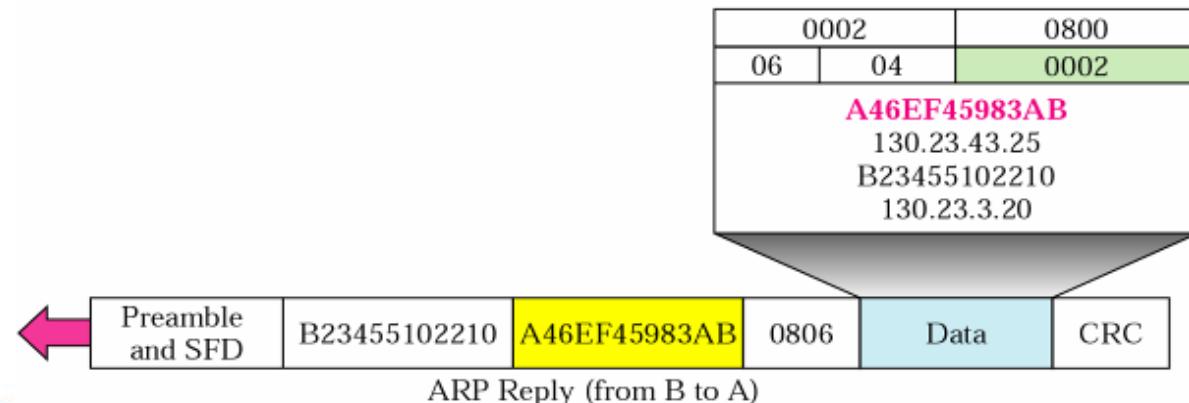
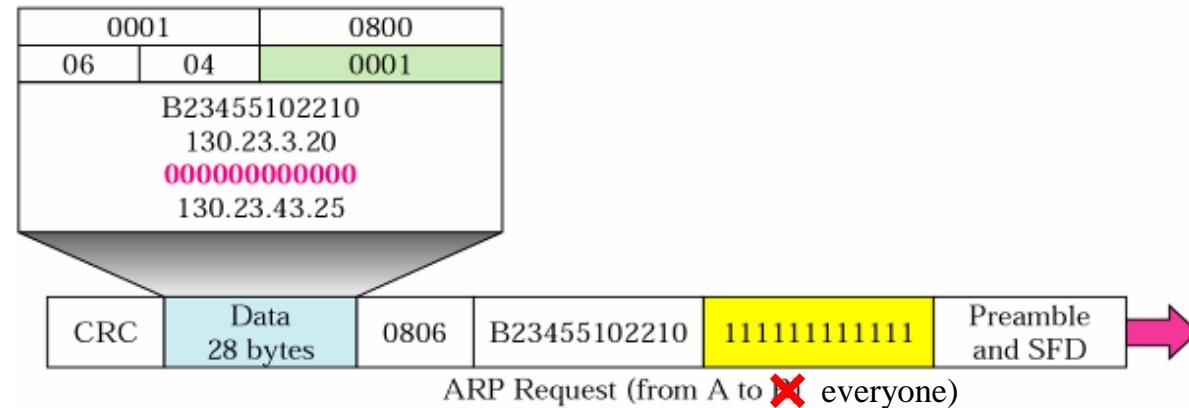
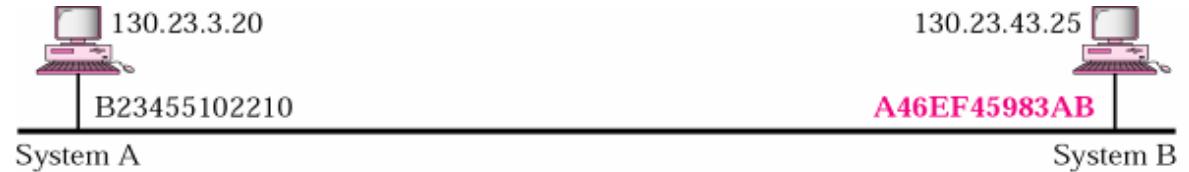
Frame 34 (42 bytes on wire, 42 bytes captured)
Arrival Time: Nov 27, 2005 02:30:26.608147000
[Time delta from previous packet: 59.766200000 seconds]
[Time since reference or first frame: 176.316976000 seconds]
Frame Number: 34
Packet Length: 42 bytes
Capture Length: 42 bytes
[Protocols in frame: eth:arp]
Ethernet II, src: 00:0f:b5:96:19:e5 (00:0f:b5:96:19:e5), dst: ff:ff:ff:ff:ff:ff
Destination: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source: 00:0f:b5:96:19:e5 (00:0f:b5:96:19:e5)
Type: ARP (0x0806)
Address Resolution Protocol (request)
Hardware type: Ethernet (0x0001)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (0x0001)
Sender MAC address: 00:0f:b5:96:19:e5 (00:0f:b5:96:19:e5)
Sender IP address: 192.168.1.5 (192.168.1.5)
Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
Target IP address: 192.168.1.2 (192.168.1.2)

0000  ff ff ff ff ff 00 0f b5 96 19 e5 08 06 00 01
0010  08 00 06 04 00 01 00 0f b5 96 19 e5 c0 a8 01 05
0020  00 00 00 00 00 00 c0 a8 01 02  ....:....:....:
```

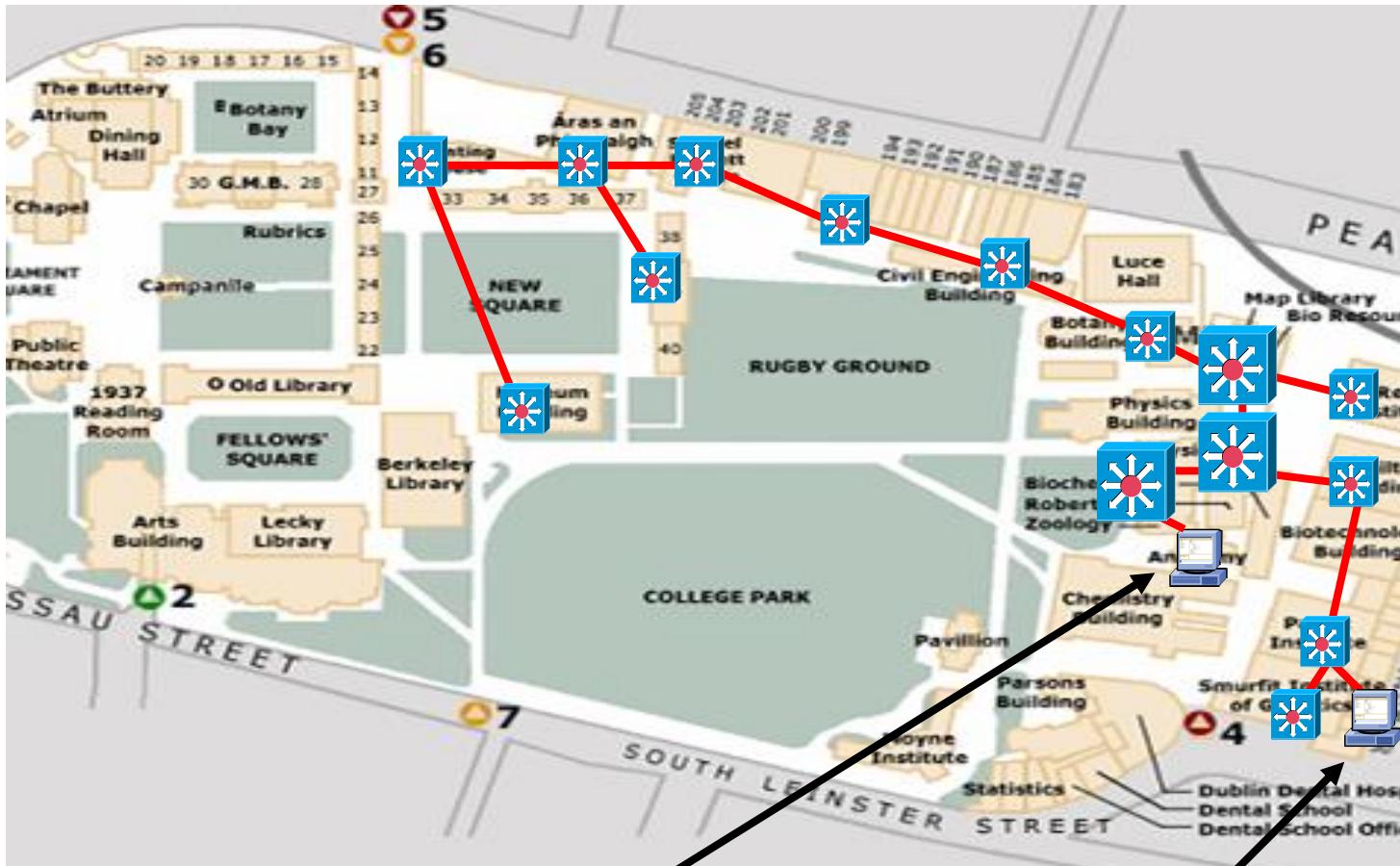
- ARP Request: Who has 192.168.1.2? Tell 192.168.1.5



ARP Example



Trinity Network with Switches



Switch

Address: 134.226.38.55
Subnet: 134.226.0.0

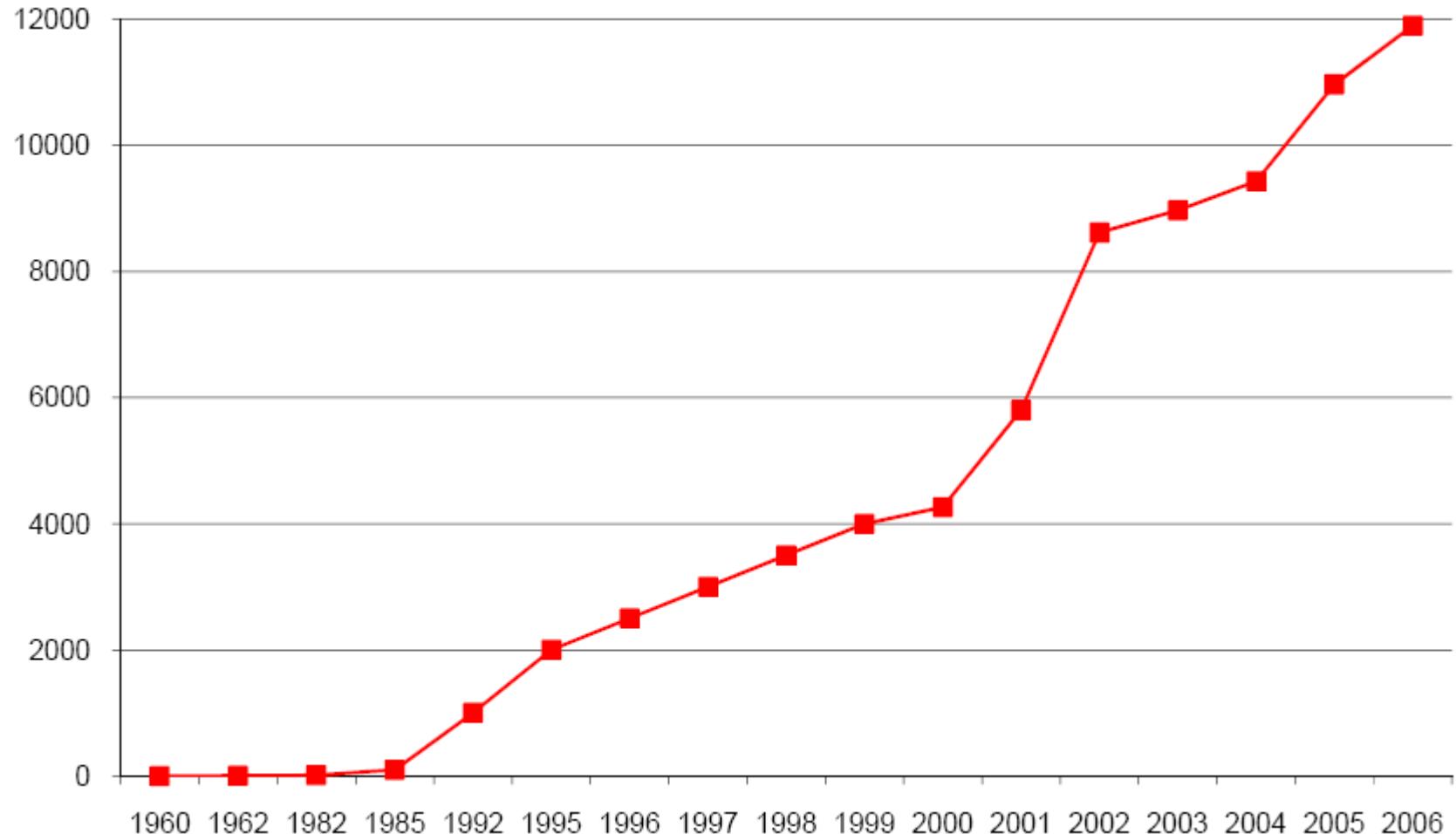
Address: 134.226.32.55
Subnet: 134.226.0.0



TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

THE
UNIVERSITY
OF DUBLIN

Number of Computers in College

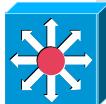
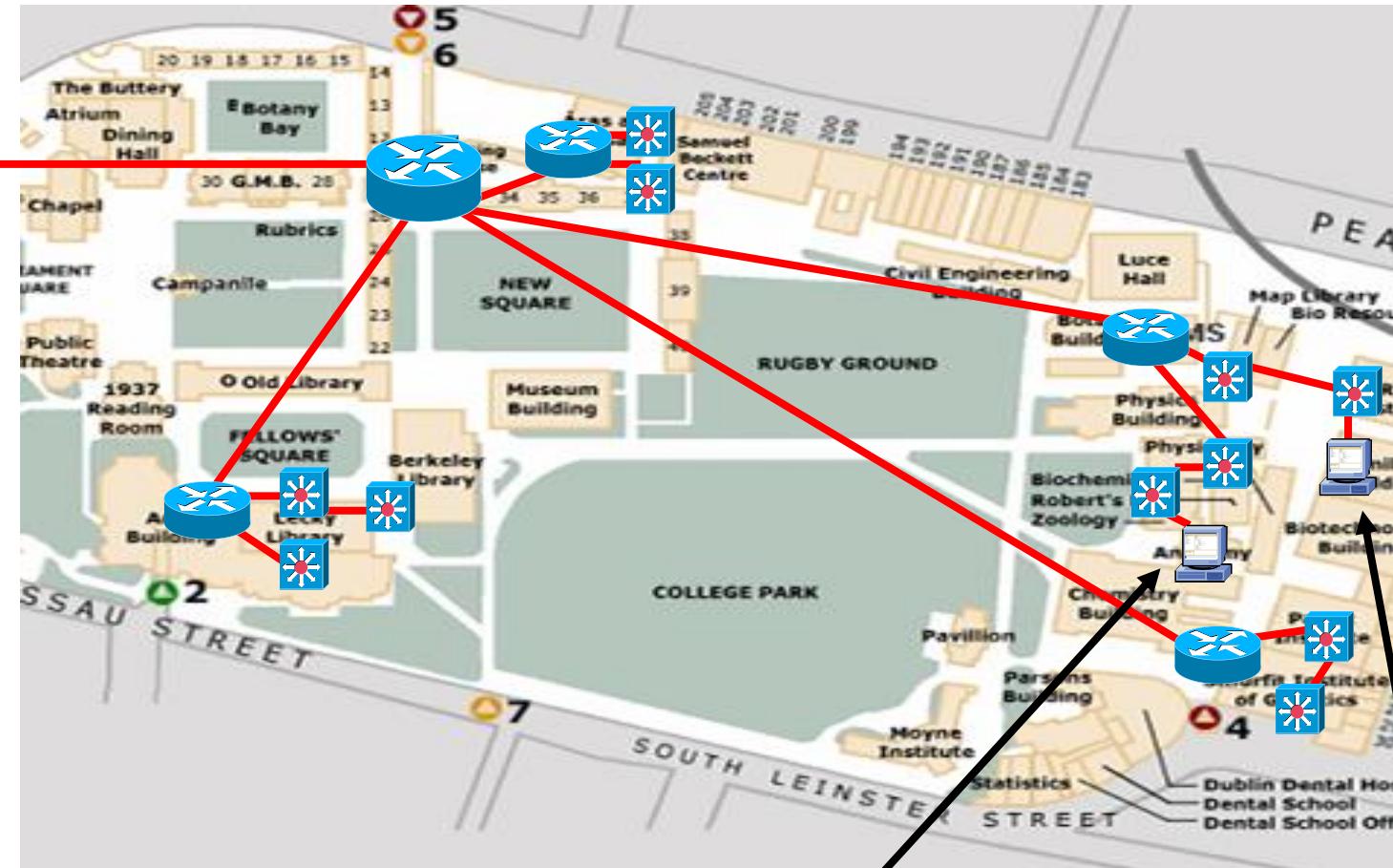


Broadcast-Storms

- Imagine 1000s of machines sending ARP requests



Trinity Network with Subnets



Switch



Router



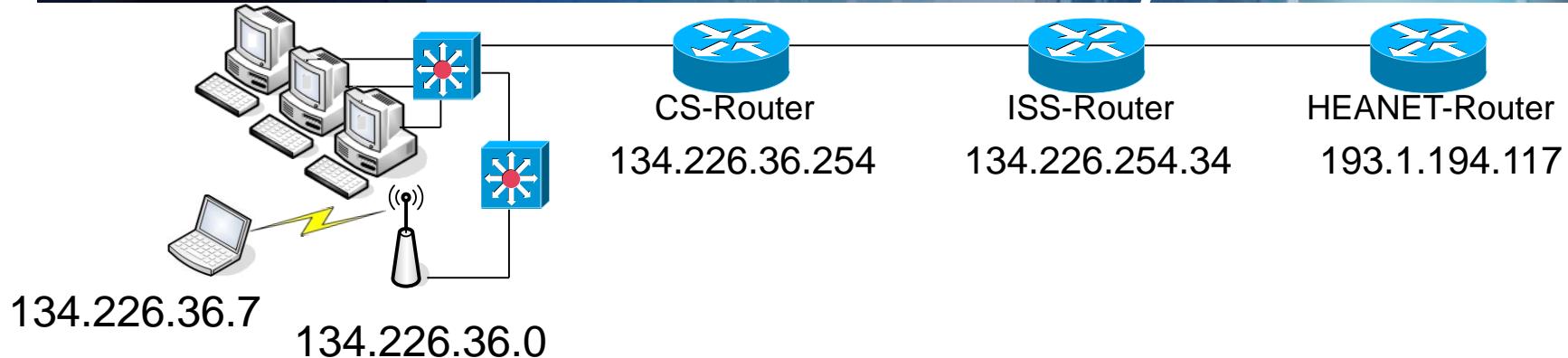
TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

THE
UNIVERSITY
OF DUBLIN

Address: 134.226.38.55
Subnet: 134.226.38.0

Address: 134.226.32.55
Subnet: 134.226.32.0

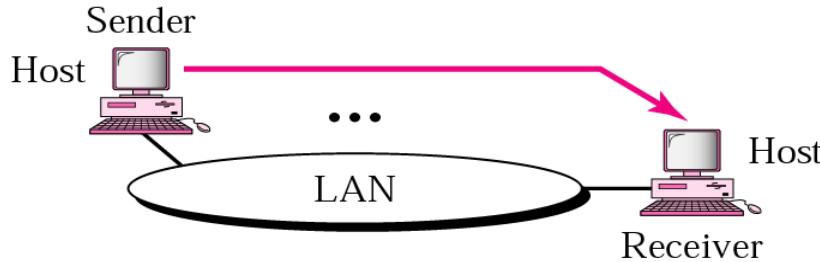
Default Gateway



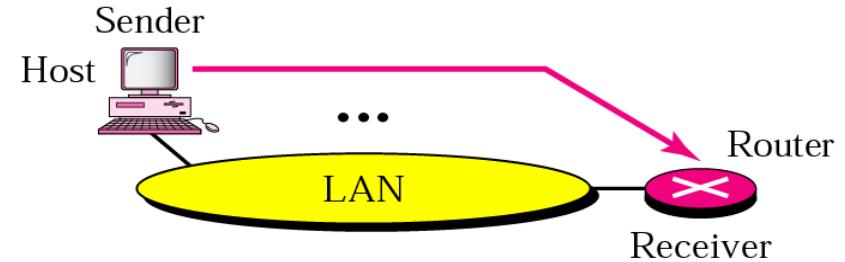
Subnet	Gateway	Netmask	Interface
134.226.36.0	0.0.0.0	255.255.0.0	eth1
0.0.0.0	134.226.36.254	0.0.0.0	eth1

- All nodes within the subnet can communicate directly with each other
- All communication with nodes in other networks passes through the default gateway e.g. router 134.226.36.254

4 Cases for ARP



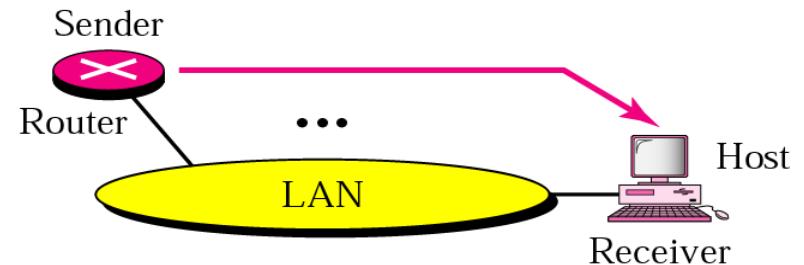
Case 1. A host has a packet to send to another host on the same network.



Case 2. A host wants to send a packet to another host on another network.
It must first be delivered to the appropriate router.



Case 3. A router receives a packet to be sent to a host on another network.
It must first be delivered to the appropriate router.



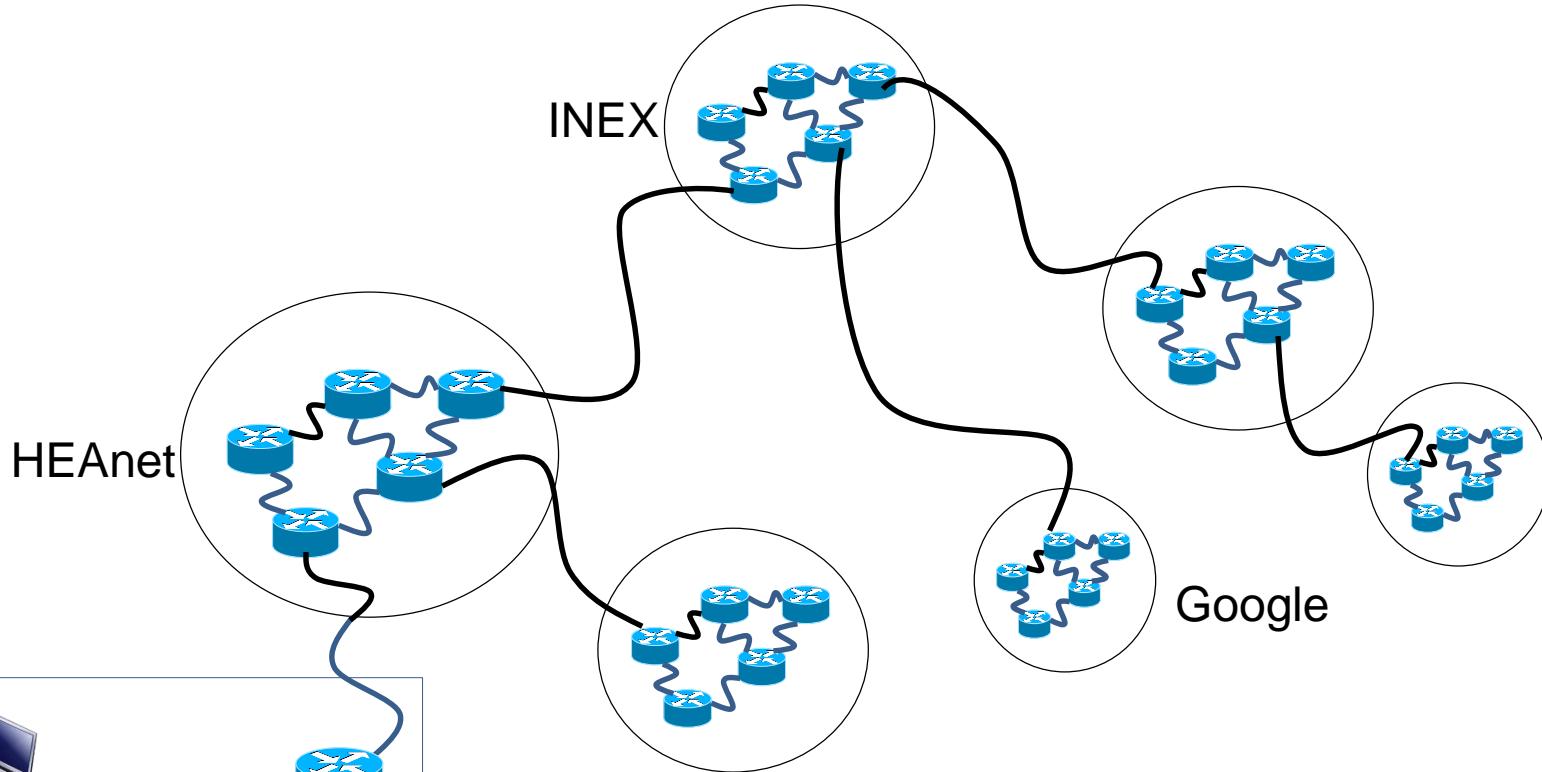
Case 4. A router receives a packet to be sent to a host on the same network.

Summary: ARP

- “Who-has?” packet
- Associating IP addresses with hardware addresses
- Based on broadcast



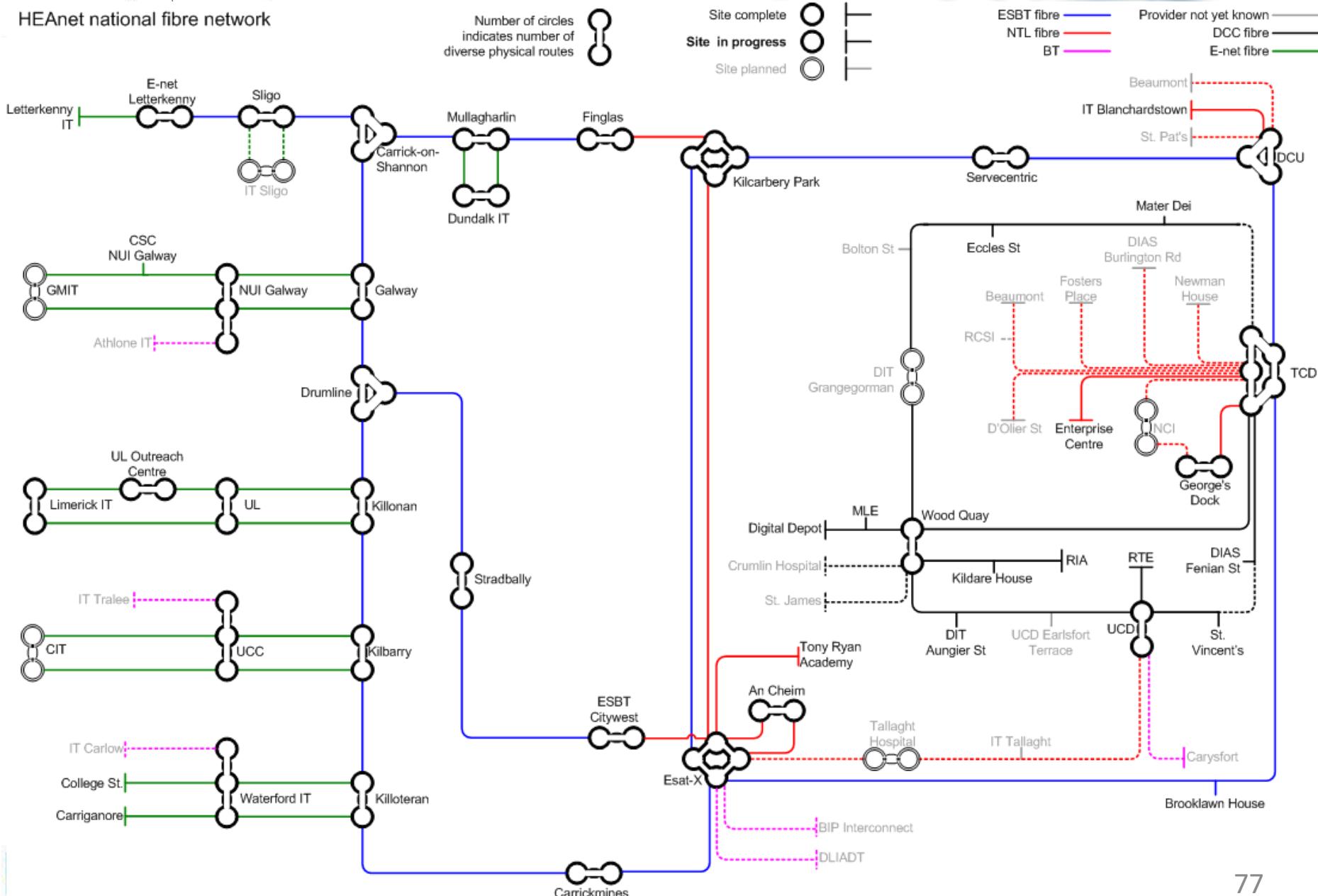
Internet = Network of Networks



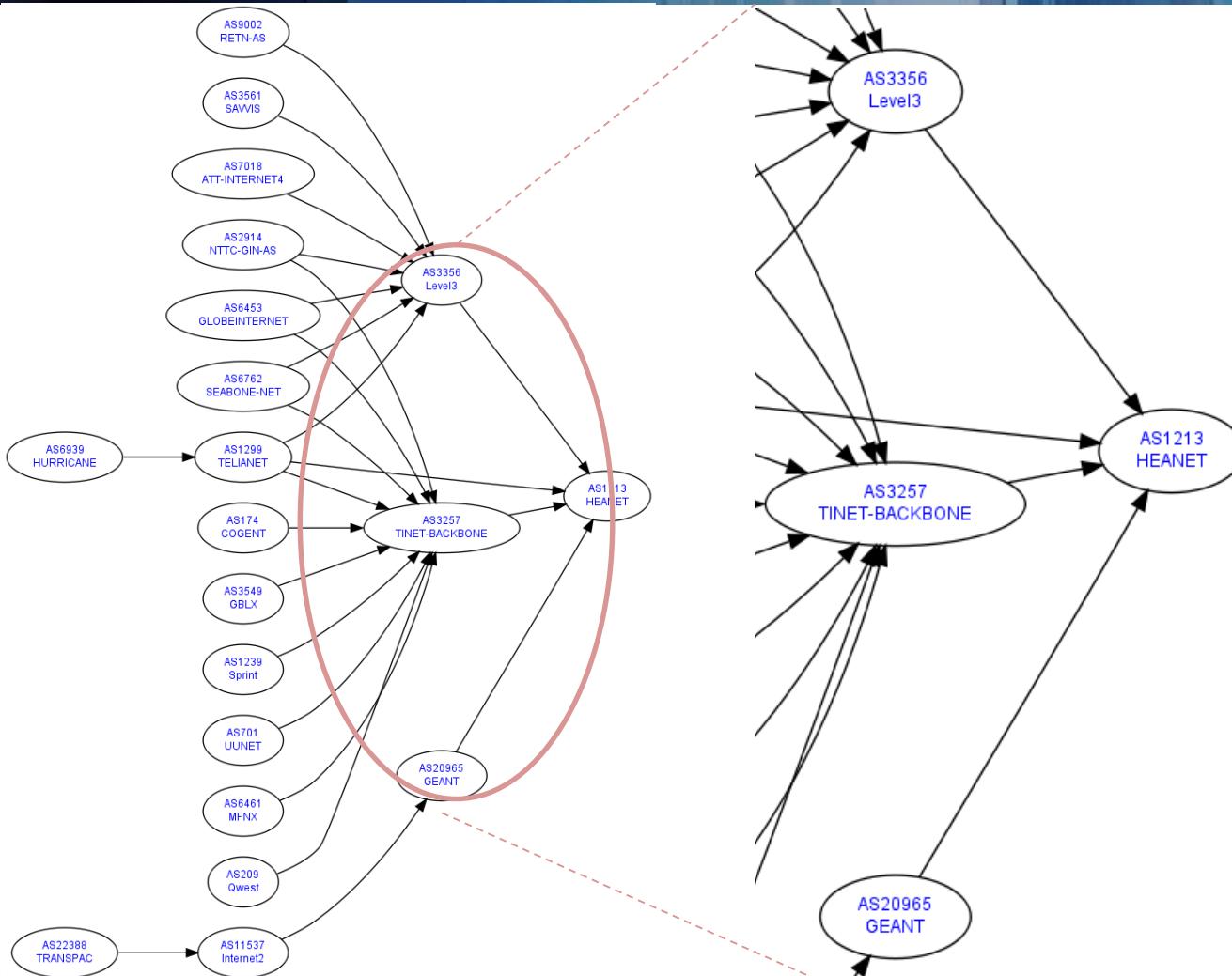
TCD/Edge Network

school of Computer Science & Statistics

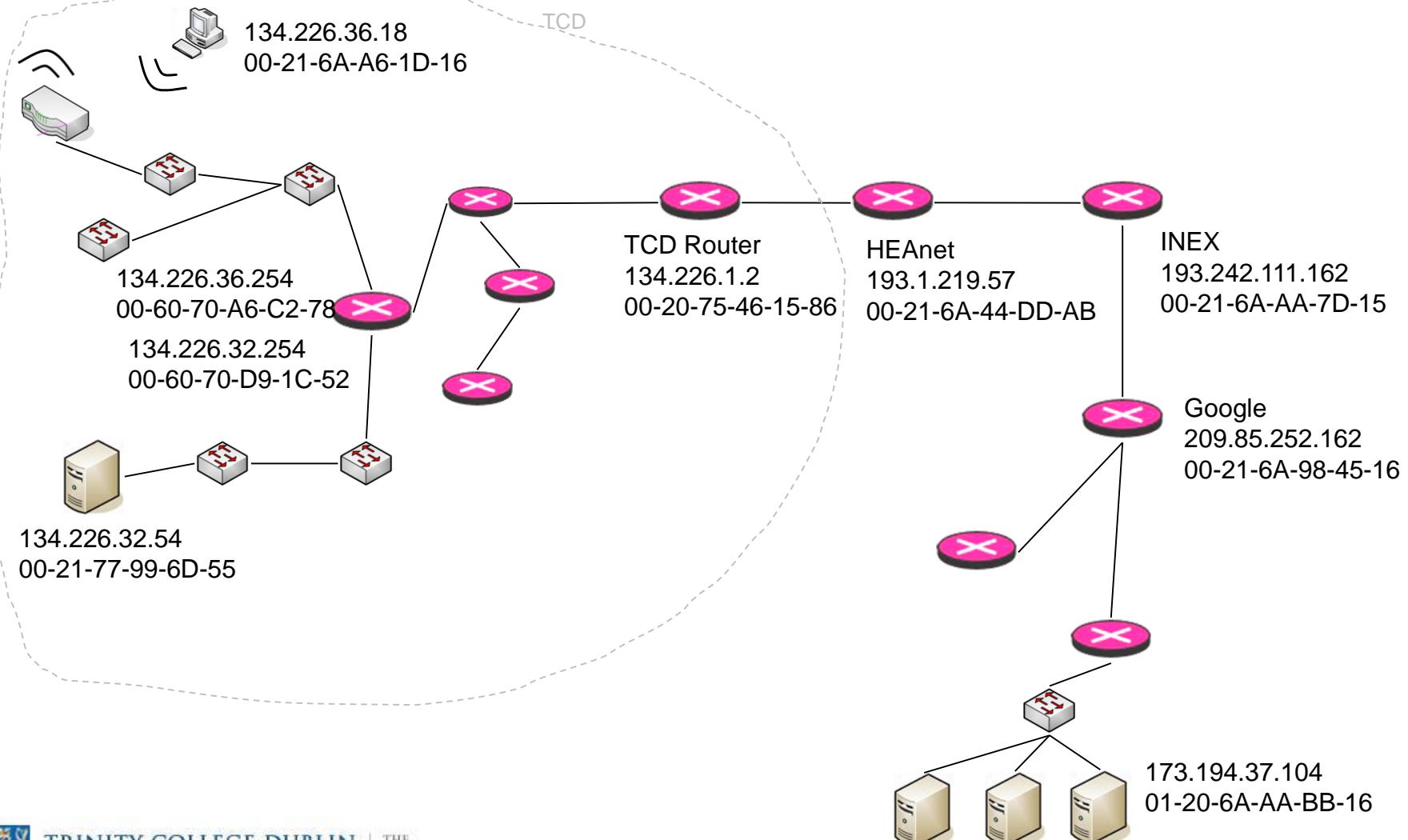
HEAnet national fibre network



AS1213 - HEANET

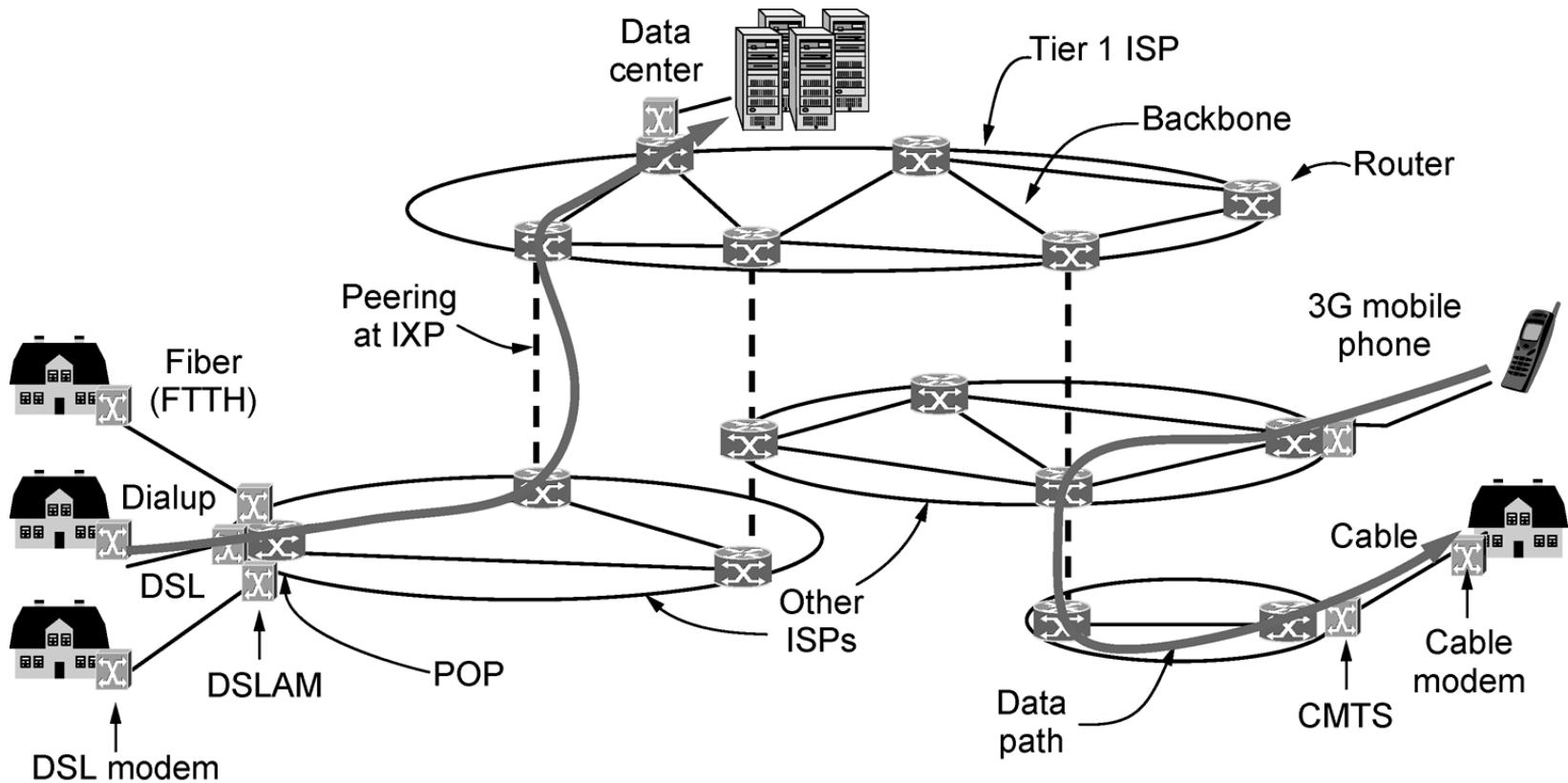


Network Layer Scenario

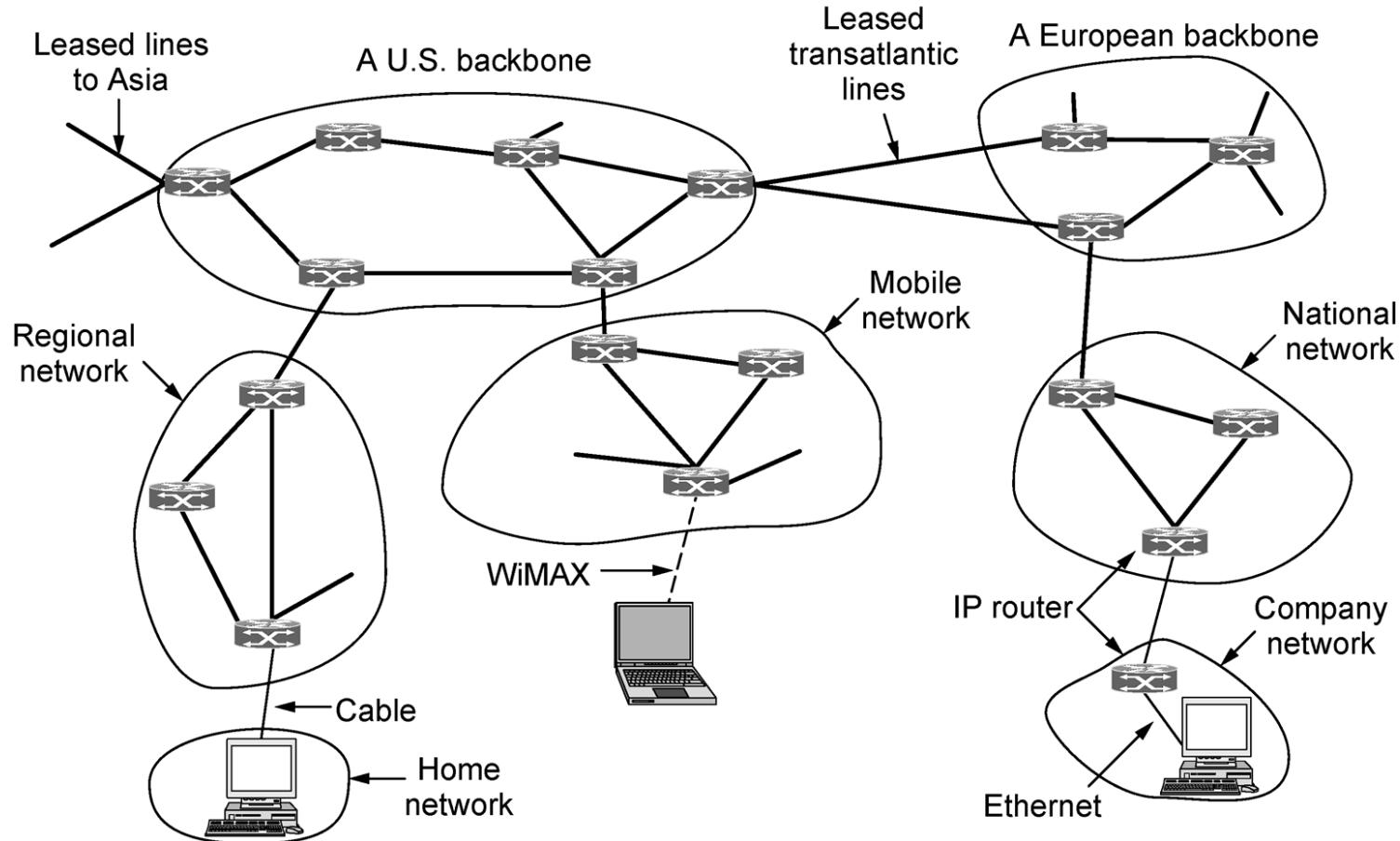




Overview of the Internet Architecture



Collection of Networks



IP Addresses

10000000 00001011 00000011 00011111

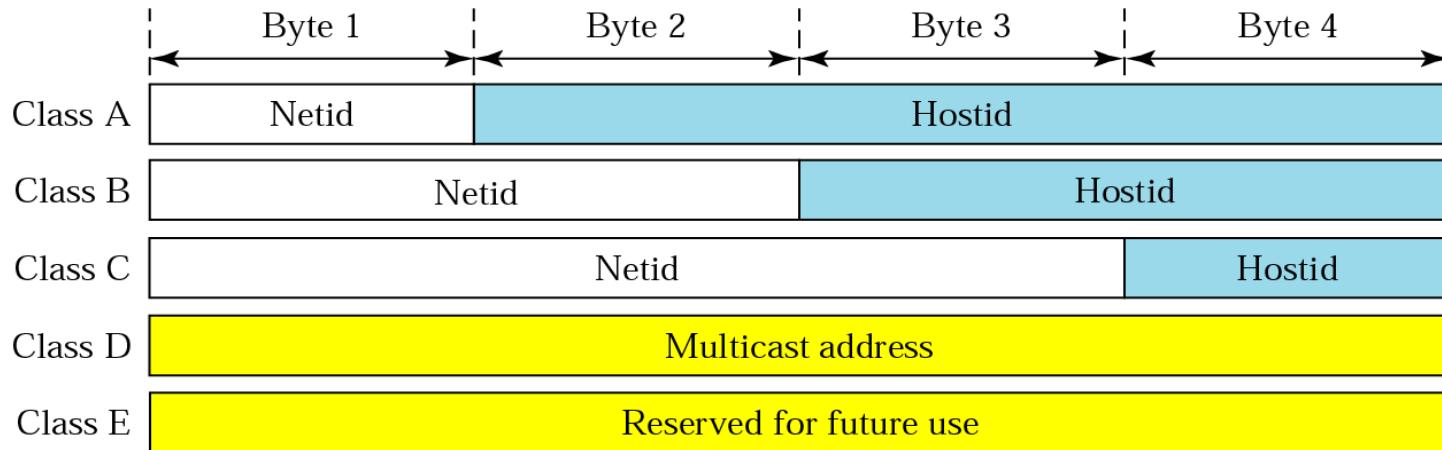
128.11.3.31

- 32-bit number
 - 4.294.967.296 addresses
- IP addresses are unique and universal
 - except private addresses

Range	Total
10.0.0.0 to 10.255.255.255	2^{24}
172.16.0.0 to 172.31.255.255	2^{20}
192.168.0.0 to 192.168.255.255	2^{16}

- Dotted decimal notation:
 - Bytes of binary notation represented as decimal separated by dot

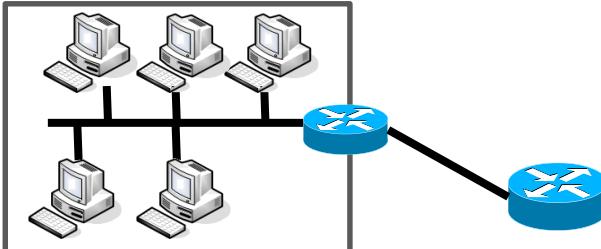
Classful Addresses



- Class A (international organisations)
 - 126 networks with 16,277,214 hosts each
- Class B (large companies)
 - 16,384 networks with 65,354 hosts each
- Class C (smaller companies)
 - 2,097,152 networks with 254 hosts each
- Inefficient use of hierarchical address space
 - Class C with 2 hosts ($2/254 = 0.78\%$ efficient)
 - Class B with 256 hosts ($256/65534 = 0.39\%$ efficient)

Network Layer

Trinity College



134.226.0.0

← Class B



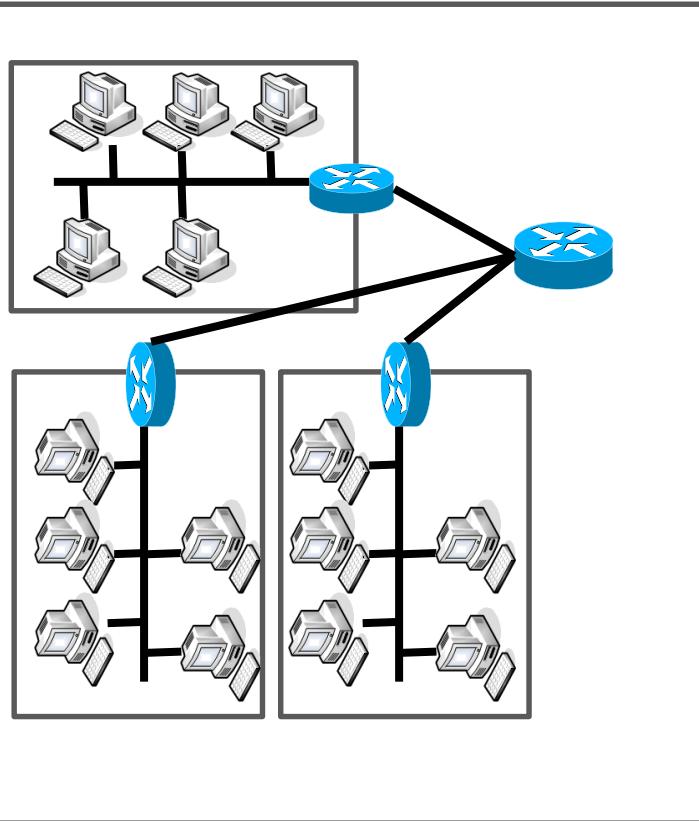
TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

THE
UNIVERSITY
OF DUBLIN

CS2031 Telecommunications II

Network Layer

Trinity College



134.226.0.0

←
Class B



TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

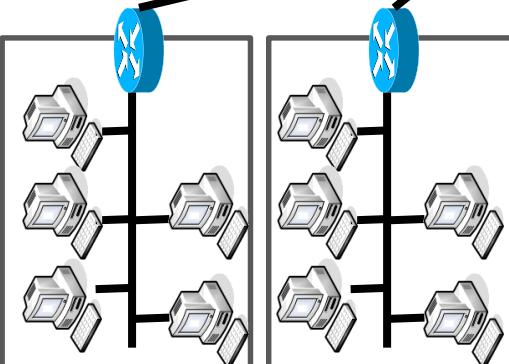
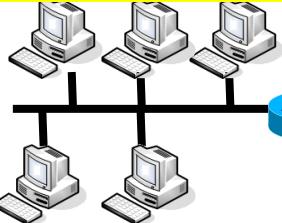
THE
UNIVERSITY
OF DUBLIN

CS2031 Telecommunications II

Network Layer

Trinity College

134.226.36.0



134.226.52.0

134.226.120.0

134.226.0.0

Class B ← Classful Addresses

CS2031 Telecommunications II

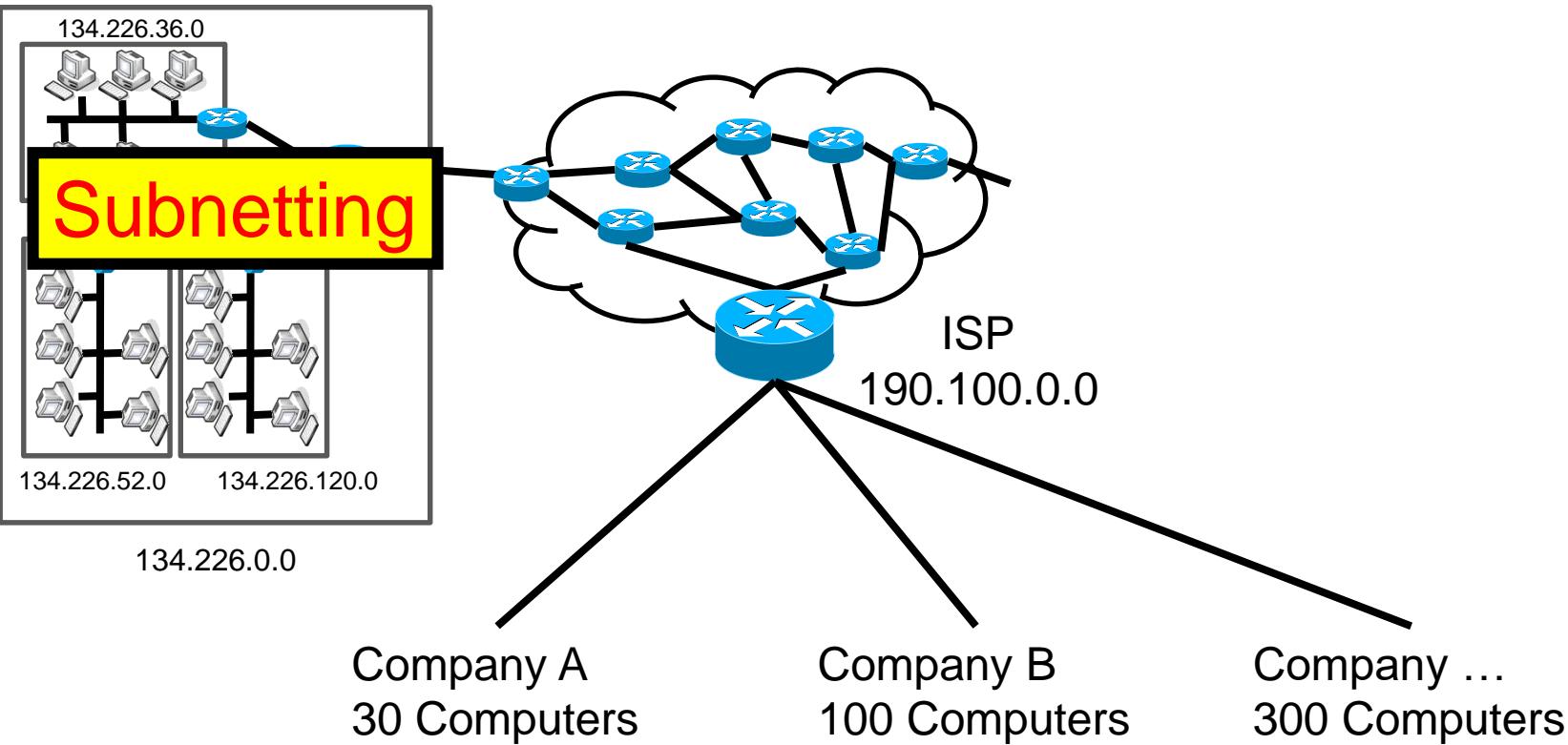


TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

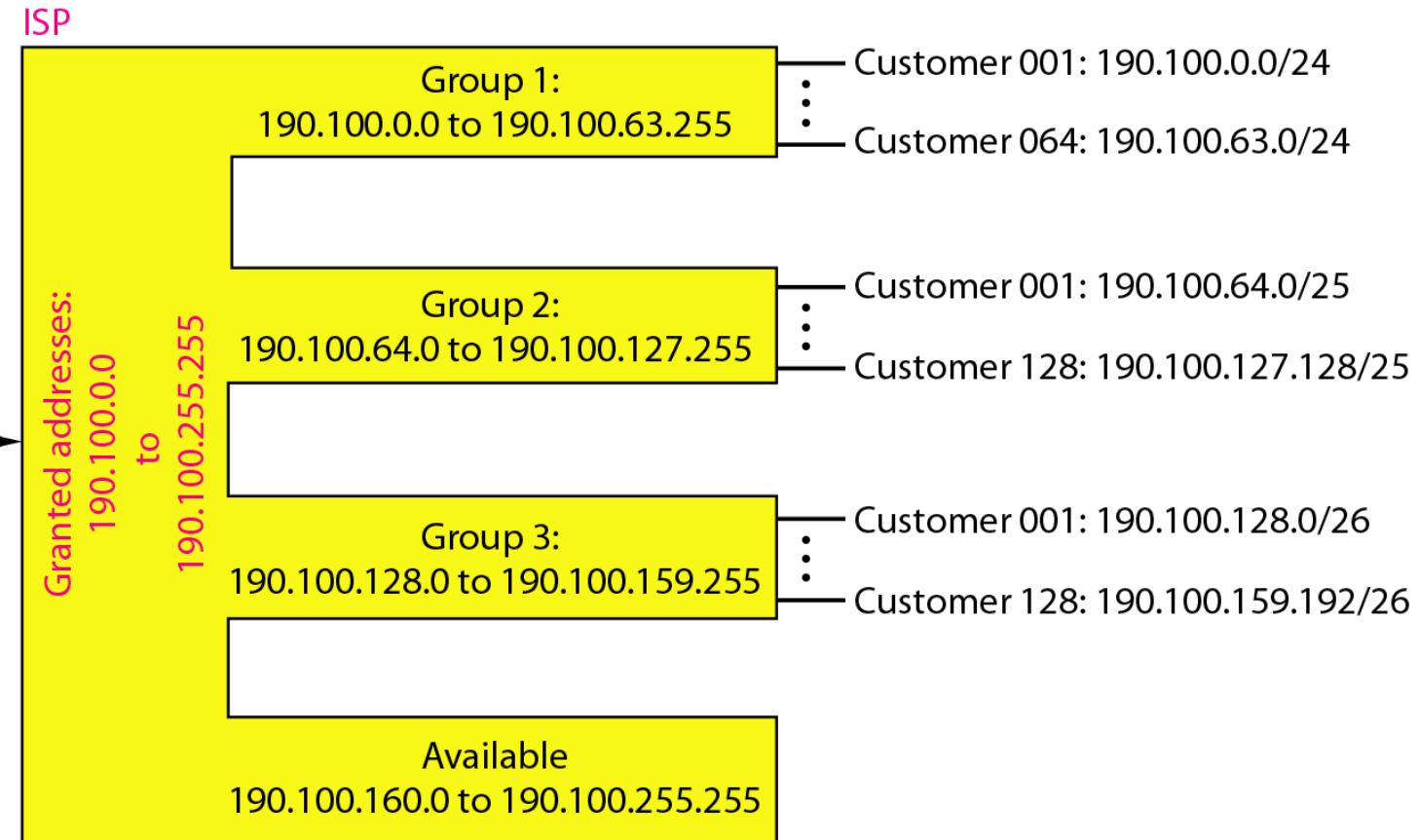
THE
UNIVERSITY
OF DUBLIN

Network Layer

Trinity College



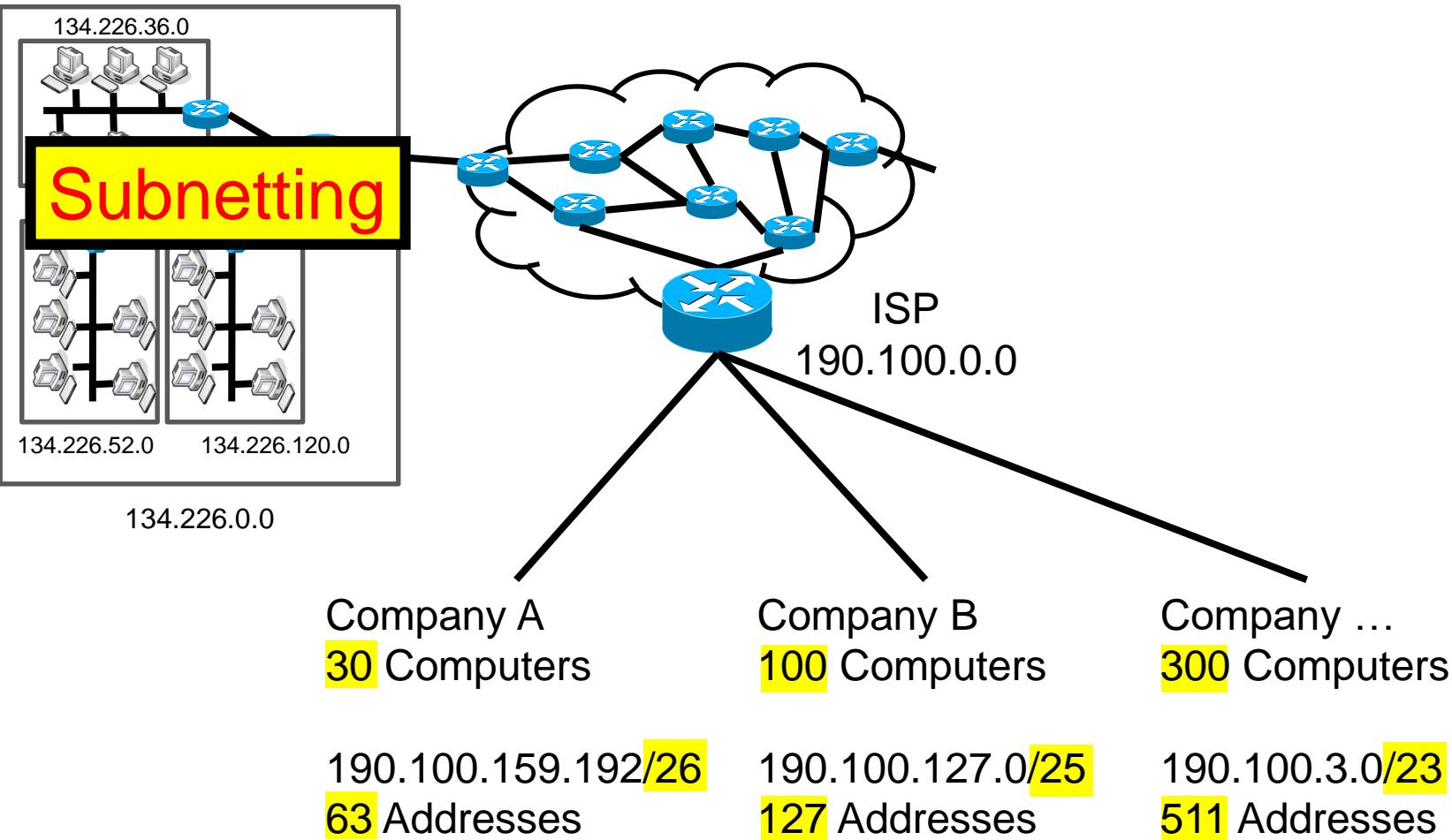
ISPs & Classless Addresses



* Figure is courtesy of B. Forouzan

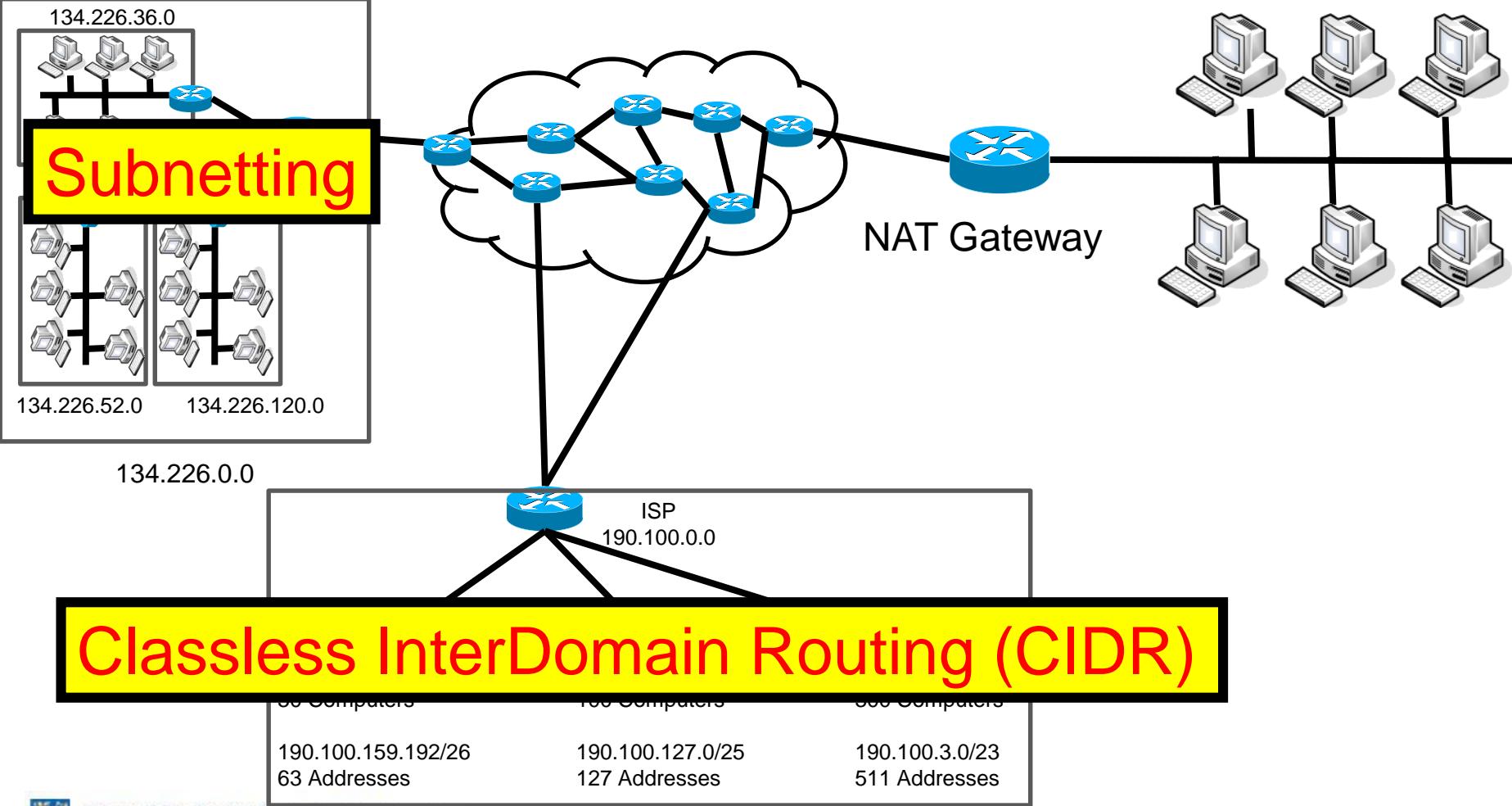
Network Layer

Trinity College



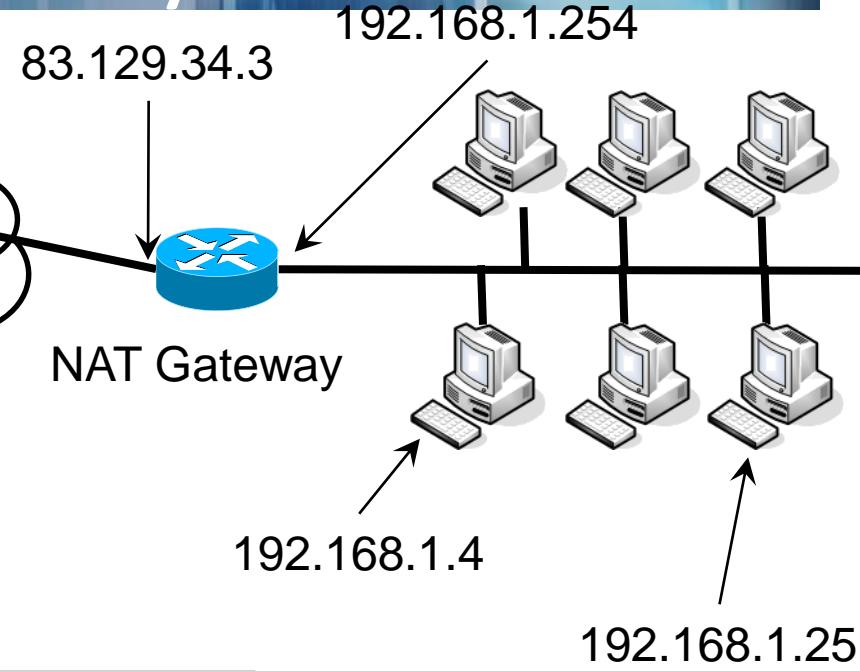
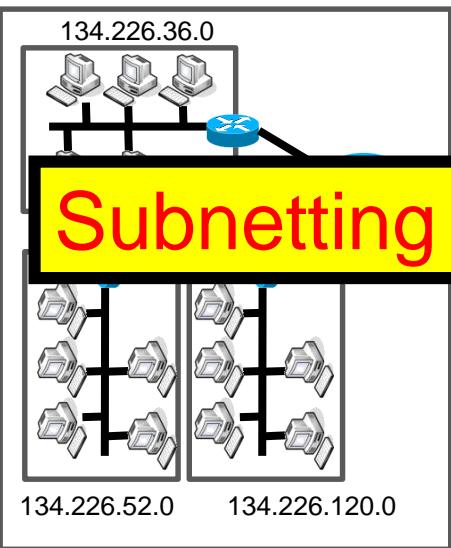
Network Layer

Trinity College



Network Layer

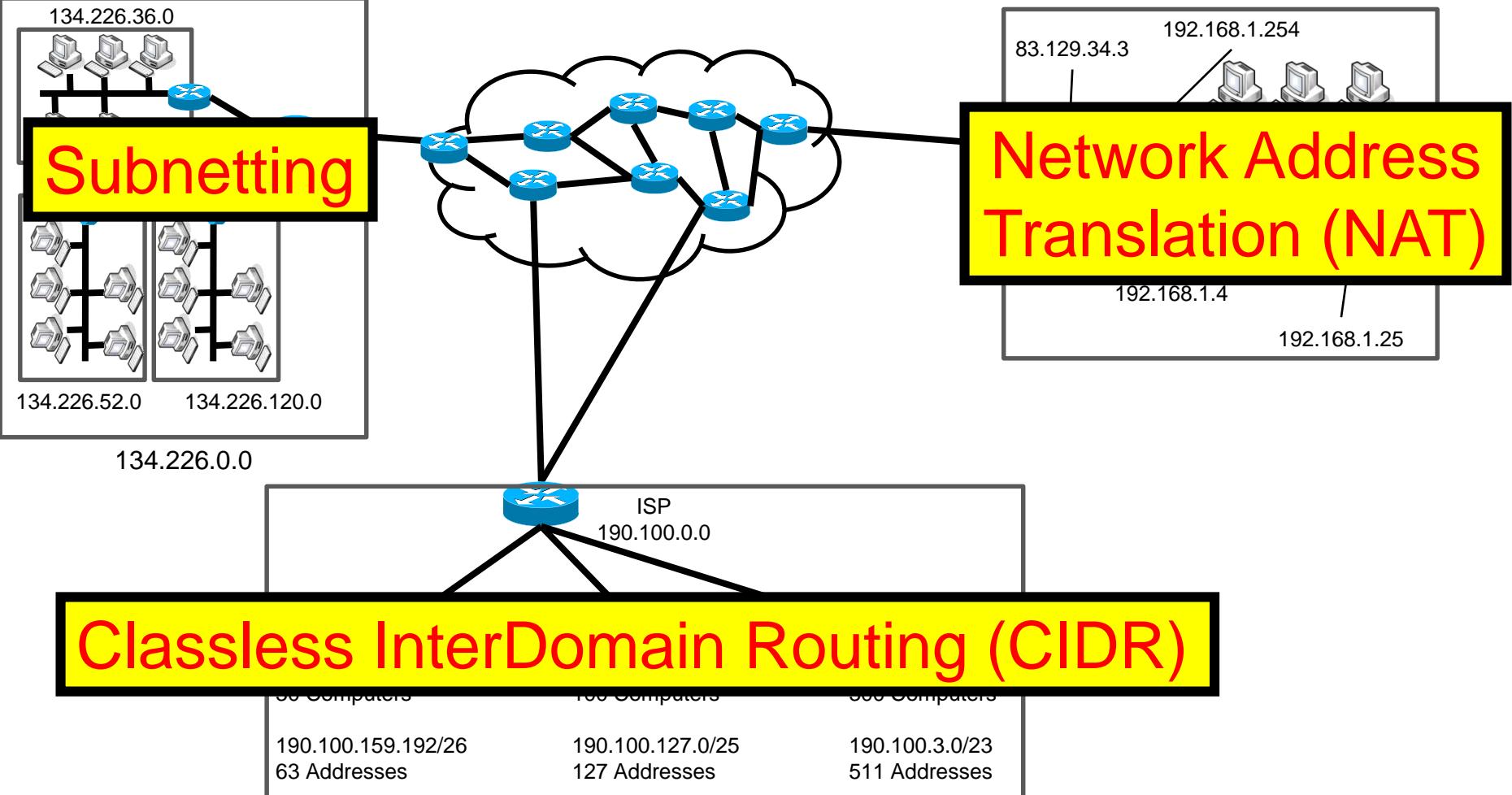
Trinity College



Classless InterDomain Routing (CIDR)

Subnet	IP Range	Number of Addresses
1	190.100.159.192/26	63 Addresses
2	190.100.127.0/25	127 Addresses
3	190.100.3.0/23	511 Addresses

Network Layer



Summary: Network Layer&Addresses

- Dotted-decimal notation
- Classful addresses
 - Classes A, B, and C for networks, D for multicast
- Subnetting
- Classless Inter-Domain Routing (CIDR)
 - / notation = significant bits in subnet mask
- Network address translation (NAT)





Internet Protocol (IP)

Fragmentation



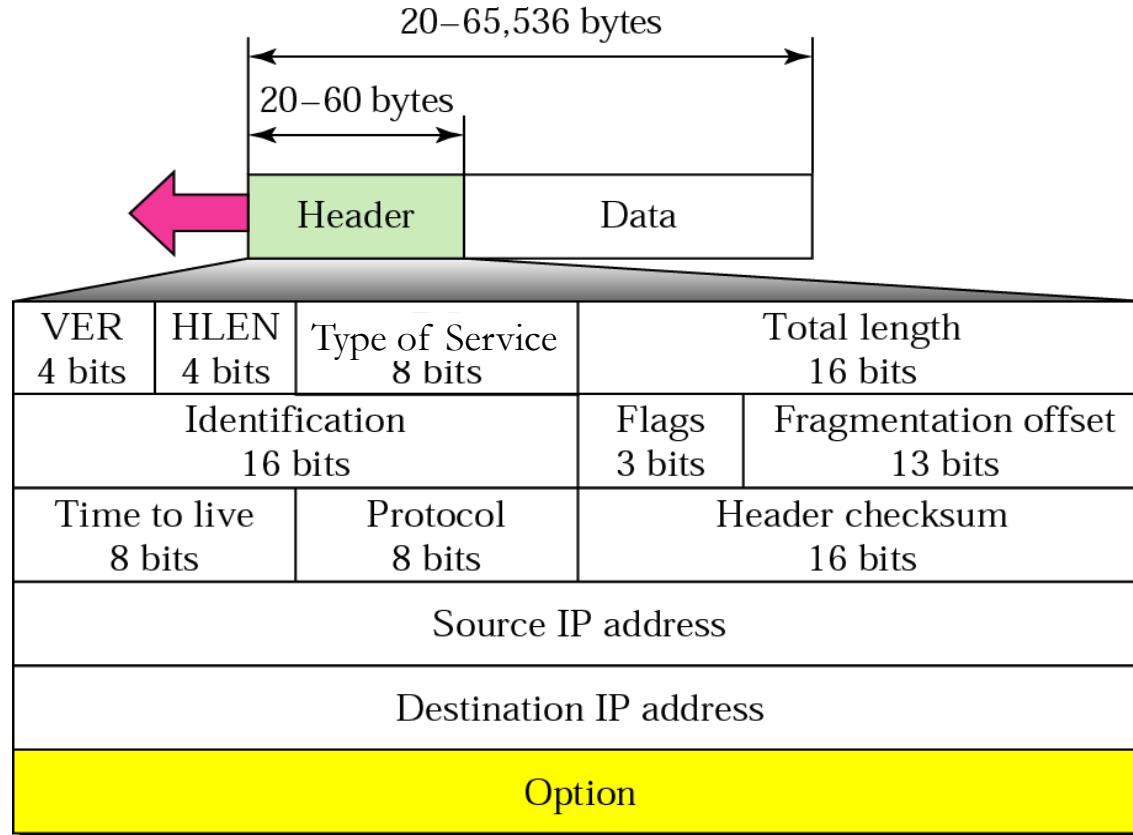
IP Datagram Format

- The unit of IP data delivery is called a datagram
- Datagrams can have different sizes
- Includes header area and data area



- Header area generally 20 bytes
 - but can be more depending on options

IP Datagram



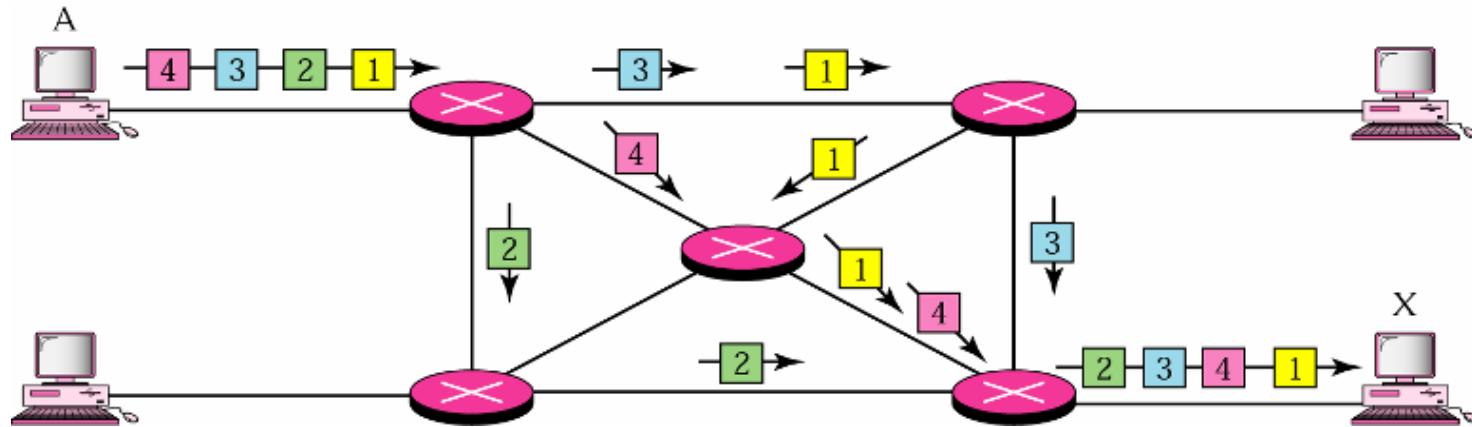
- The total length field defines the total length of the datagram including the header.

IP Datagram Header Fields

Length in bit	Name	Description
4	Version	Version of IP (4, 6)
4	HLEN	Length of the header
8	Service Type	Sender's preference for services e.g. low latency, etc
16	Total Length	Total length of datagram in bytes/octects
16	Identification	Unique identification for each datagram
4	Flags	Control fragmentation of datagrams
12	Fragment Offset	Used in fragmentation of datagrams
8	TTL	Time to live; decremented at each hop it passes
8	Protocol	Higher-layer protocol such as TCP, UDP, etc
16	Header Checksum	Checksum to verify correctness of header information
32	Source address	IP address of the sender
32	Dest. address	IP address of the destination



IP Service Model



- Connection-less Communication
 - No state is kept about individual packets
- Order not guaranteed
- Best-effort delivery (unreliable service)
 - Packets may be lost
 - Packets may be delivered out of order
 - Duplicate copies of a packet may be delivered
 - Packets can be delayed for a long time

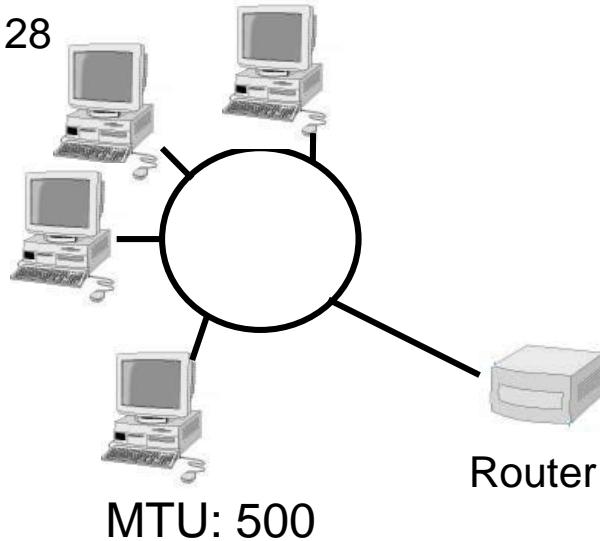
Maximum Transmission Unit (MTU)

- Maximum size of a data unit depends on underlying hardware architecture
- Maximum frame size determines *Maximum Transmission Unit (MTU)*

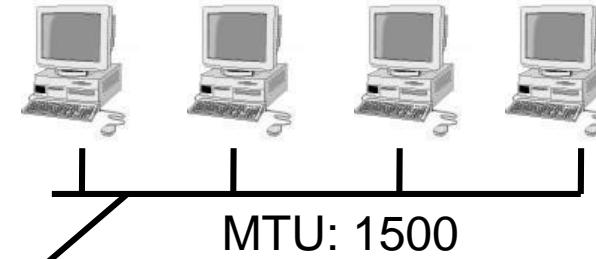
Protocol	MTU
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

IP & Network Architectures

Computer B
172.16.2.128



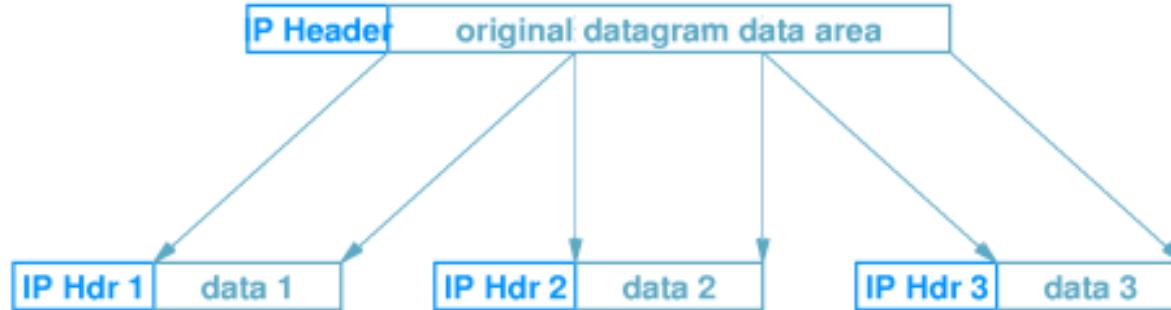
Computer A
172.16.1.63



- IP datagrams may be larger than most MTUs of underlying network architectures
- Maximum frame size limits maximum size of an IP datagram for a network



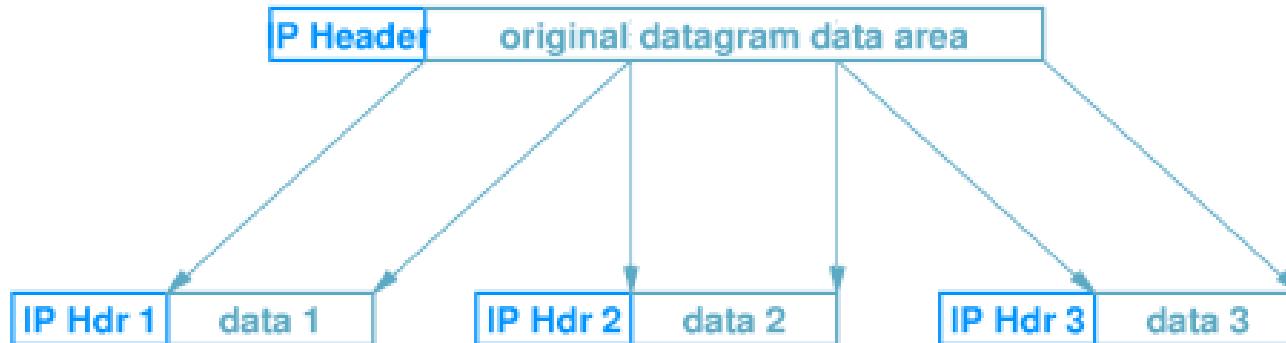
Fragmentation



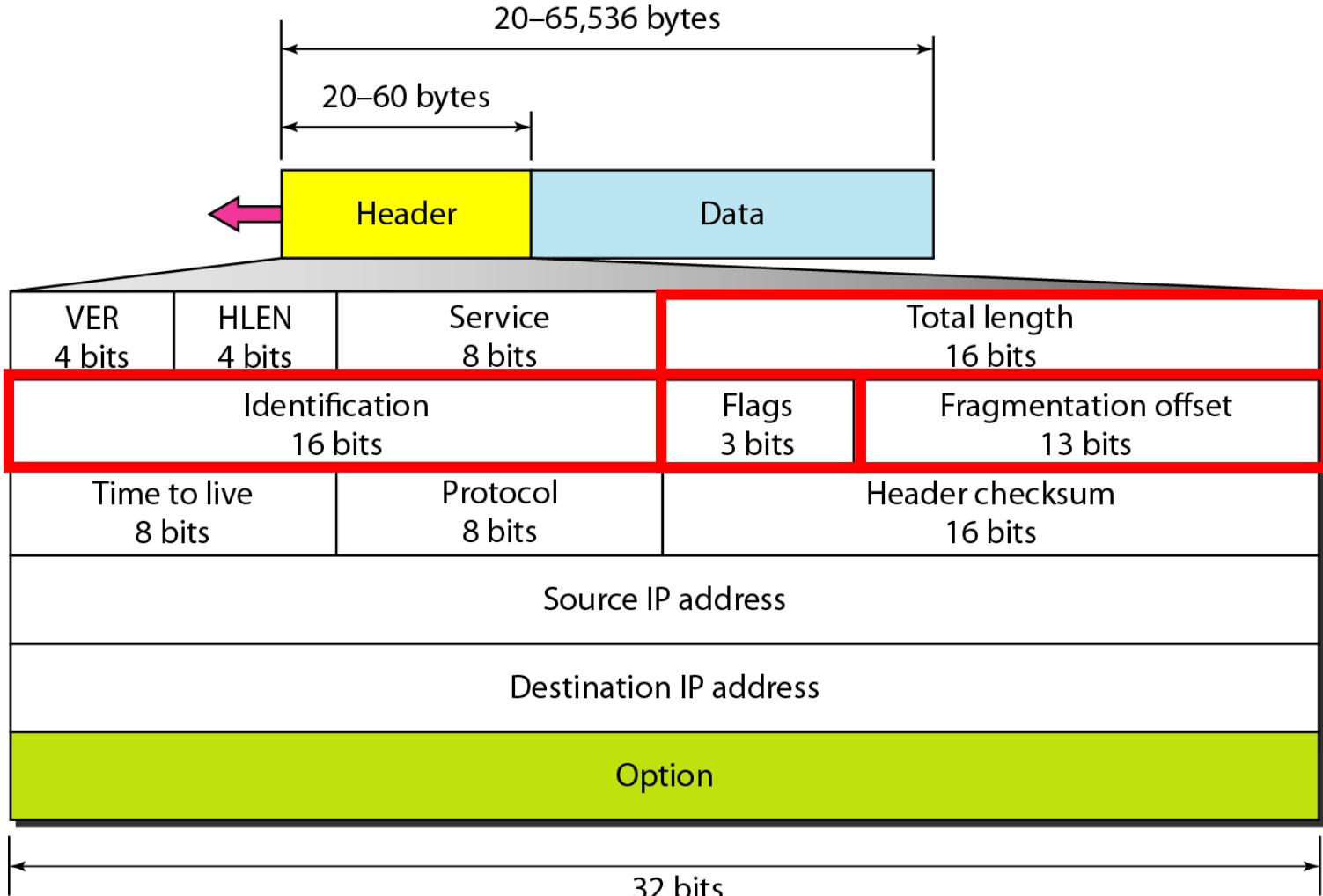
- Possible techniques:
 - Limit datagram size to smallest MTU of any network
 - Adjust datagram size as packet progresses through networks
 - Router detects datagram larger than network MTU and splits into pieces
- IP Strategy
 - Fragment when necessary (Datagram > MTU)
 - Try to avoid fragmentation at source host
 - Re-fragmentation is possible
 - **Delay reassembly until destination host**
 - Do not recover from lost fragments
 - If one fragment is lost all fragments are discarded

Fragmentation (details)

- Each fragment is an independent datagram
 - Includes all header fields
 - Bit in header indicates datagram is a fragment
 - Other fields have information for reconstructing original datagram
 - FRAGMENT OFFSET gives original location of fragment



Fragmentation & IPv4 Header

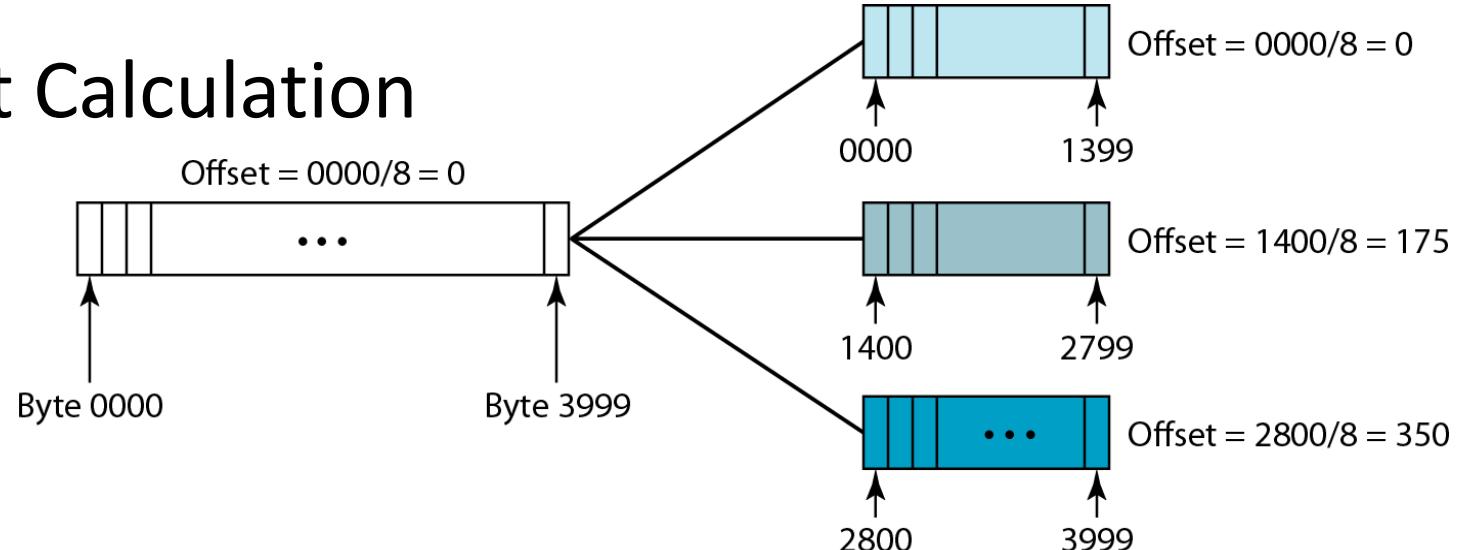


Header Fields

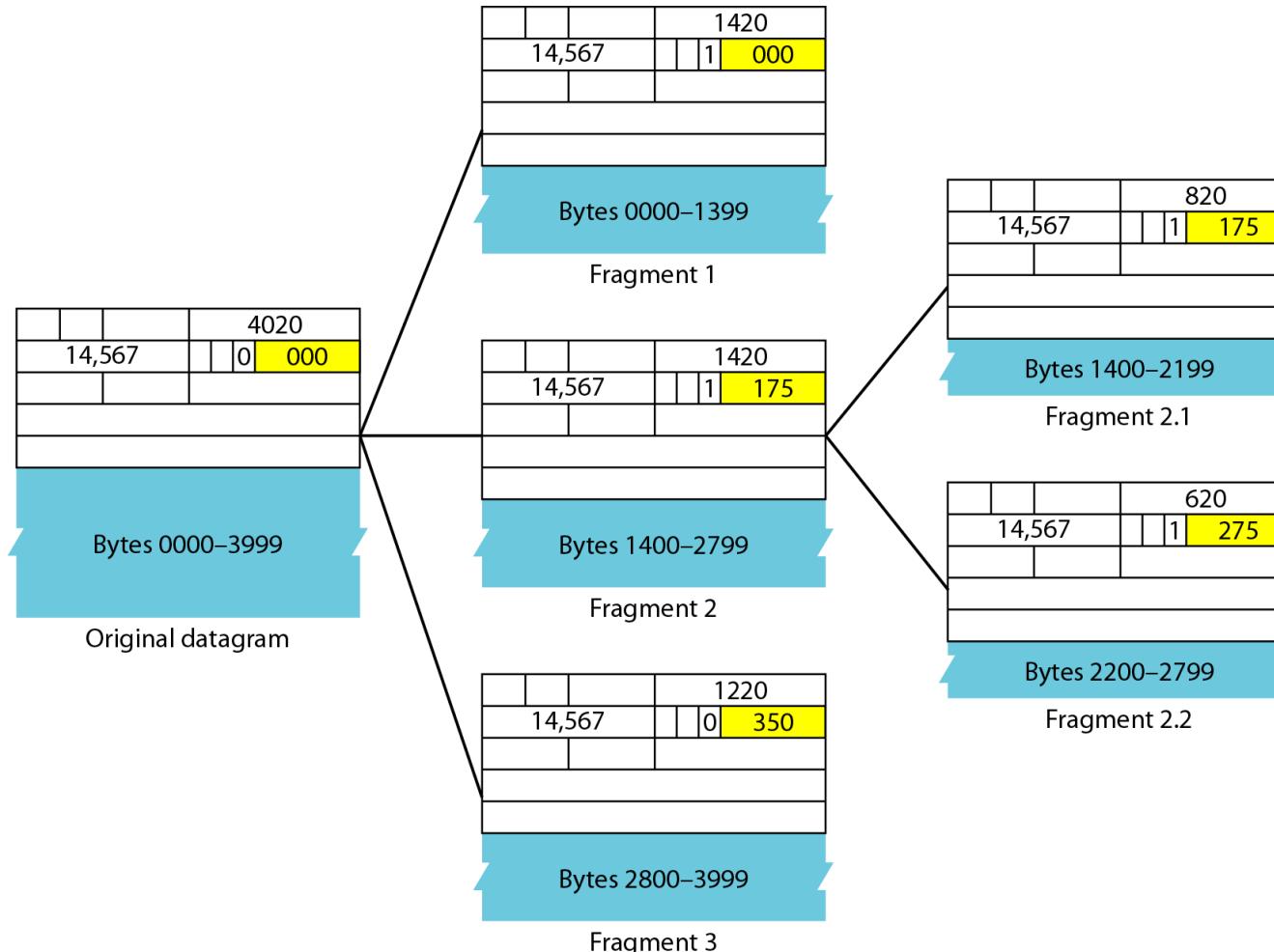
- “Do not fragment”-Request
- More Fragments



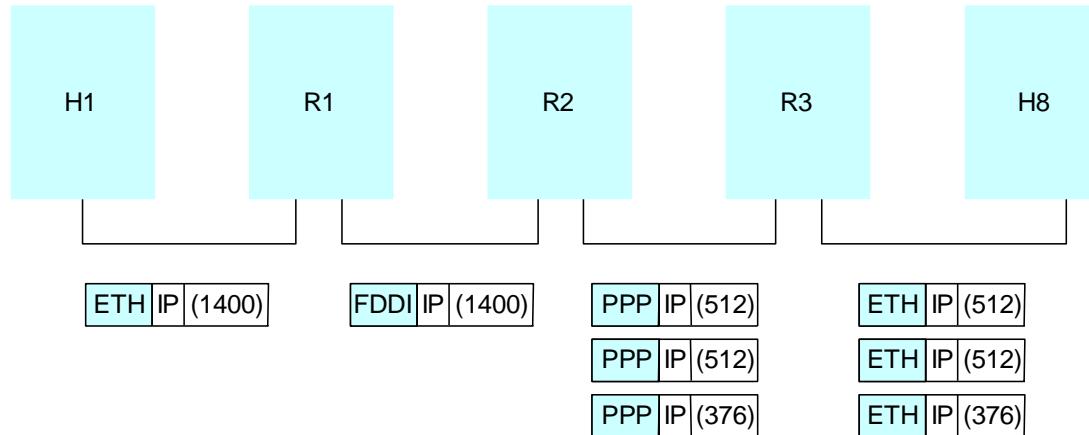
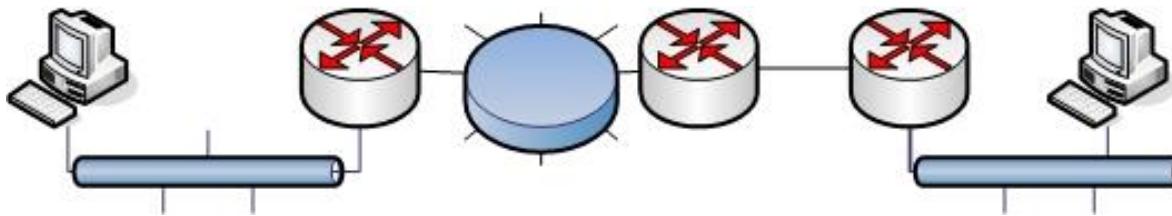
- Offset Calculation



Fragmentation Example I



Fragmentation Example II



Start of header
Ident= x 0 Offset= 0
Rest of header
1400 data bytes

Start of header
Ident= x 1 Offset= 0
Rest of header
512 data bytes

Start of header
Ident= x 1 Offset= 64
Rest of header
512 data bytes

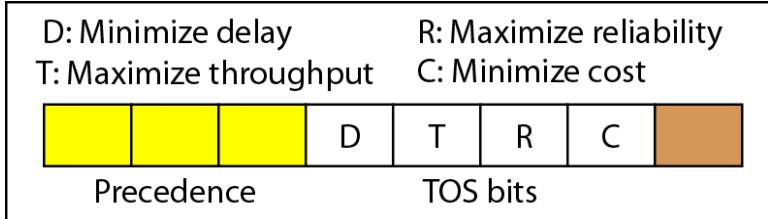
Start of header
Ident= x 0 Offset= 128
Rest of header
376 data bytes



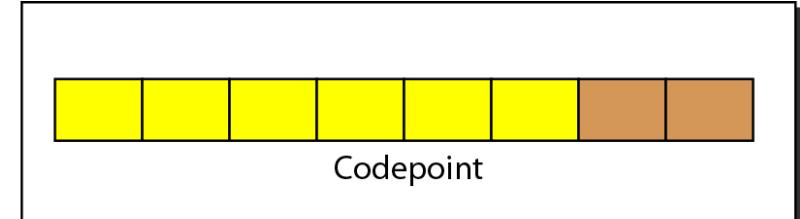
Fragment loss

- A fragment may be lost/dropped in transfer
- What happens to original datagram?
 - Destination drops entire original datagram
- How does destination identify lost fragment?
 - Sets timer with each fragment
 - If timer expires before all fragments arrive, fragment assumed lost
 - Datagram dropped
- Source is assumed to retransmit

Service types



Service type



Differentiated services

<i>TOS Bits</i>	<i>Description</i>
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

Protocols & TOS Bits

<i>Protocol</i>	<i>TOS Bits</i>	<i>Description</i>
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput



Checksum in IPv4 Header

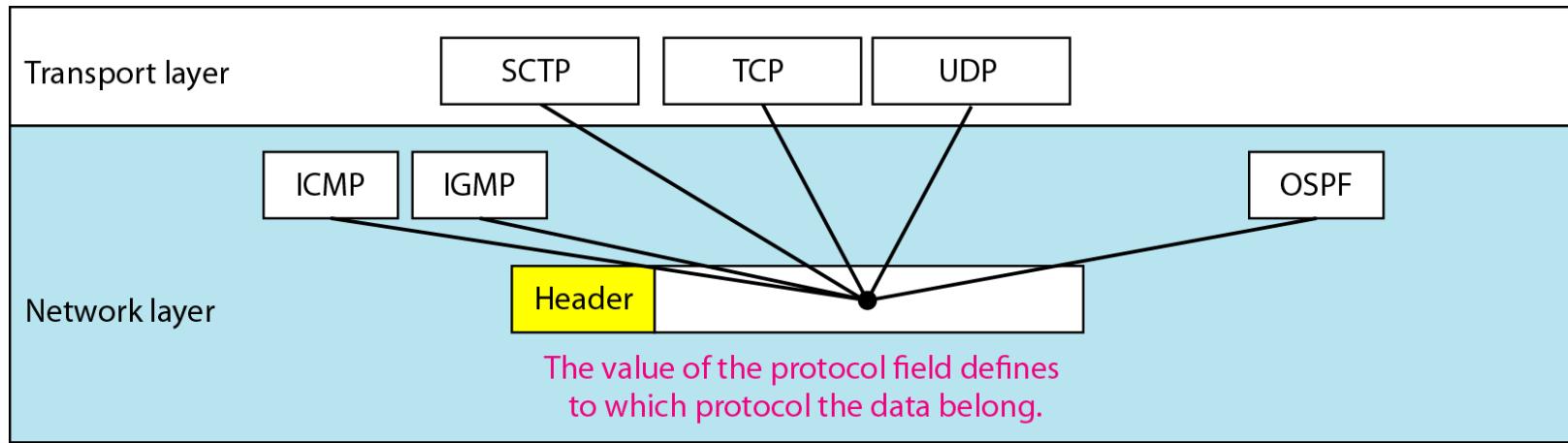
- Calculation omits:
 - Service field
 - Fragment fields and offset
 - Checksum field

4	5	0	28
	1	0	0
4	17	0	
10.12.14.5			
12.6.7.9			
4, 5, and 0	→ 4	5	0 0
28	→ 0	0	1 C
1	→ 0	0	0 1
0 and 0	→ 0	0	0 0
4 and 17	→ 0	4	1 1
0	→ 0	0	0 0
10.12	→ 0	A	0 C
14.5	→ 0	E	0 5
12.6	→ 0	C	0 6
7.9	→ 0	7	0 9
Sum	→ 7	4	4 E
Checksum	→ 8	B	B 1

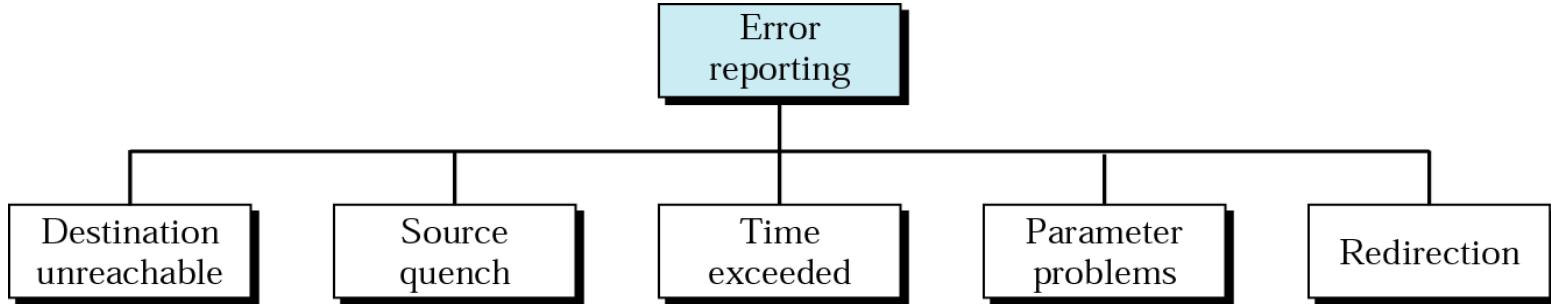
Protocol Field

- Specifies next protocol in stack

Value	Protocol
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

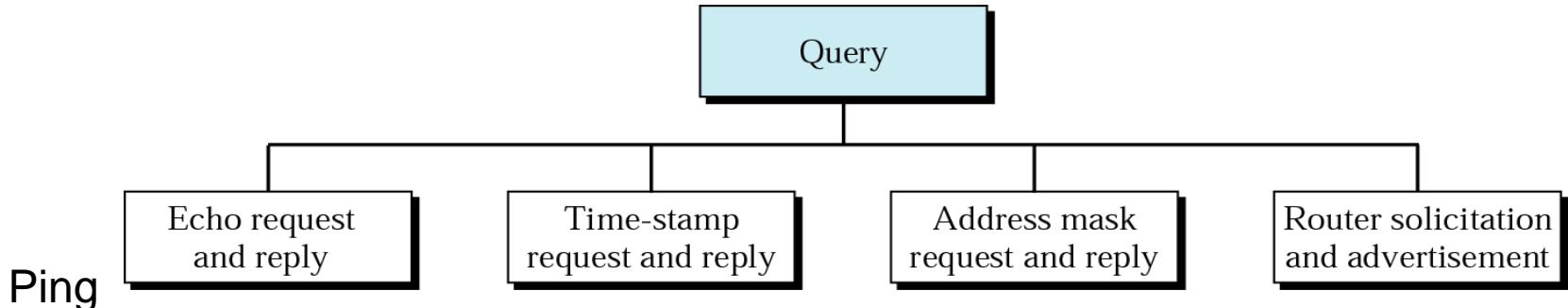


Internet Control Message Protocol



- Destination unreachable
- Source quench
 - Mechanism for destination and intermediate nodes to limit traffic from source
- Time exceeded
 - Send when datagram discarded due to TTL value of 0
- Parameter problems
 - Send when datagram discarded due to parameter ambiguity
- Redirection
 - Send from router to update routing table of source

ICMP Queries



Ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of bytes used to fill out the packet.

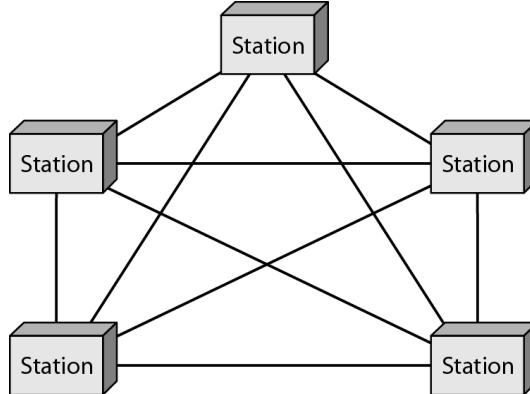
```
$ ping www.tcd.ie
PING dux6.tcd.ie (134.226.1.61): 56 data bytes
64 bytes from 134.226.1.61: icmp_seq=0 ttl=64 time=0.781 ms
64 bytes from 134.226.1.61: icmp_seq=1 ttl=64 time=0.466 ms
64 bytes from 134.226.1.61: icmp_seq=2 ttl=64 time=0.490 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.457/0.548/0.781/0.135 ms
```

Summary: Internet Protocol

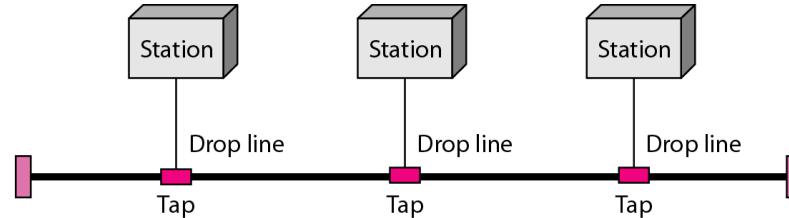
- IP Service Model
 - Connection-less, no order guaranteed
- IP Header
 - 20 bytes + options
- Fragmentation
 - Datagrams split into fragments to fit MTUs
 - Only re-assembled at destination
- Internet Control Message Protocol (ICMP)
 - Error Reporting e.g. source quench
 - Querying e.g. Ping



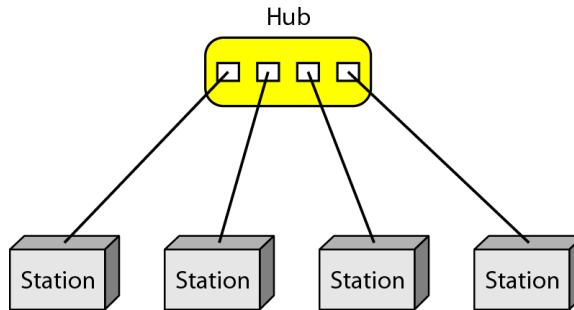
Topologies



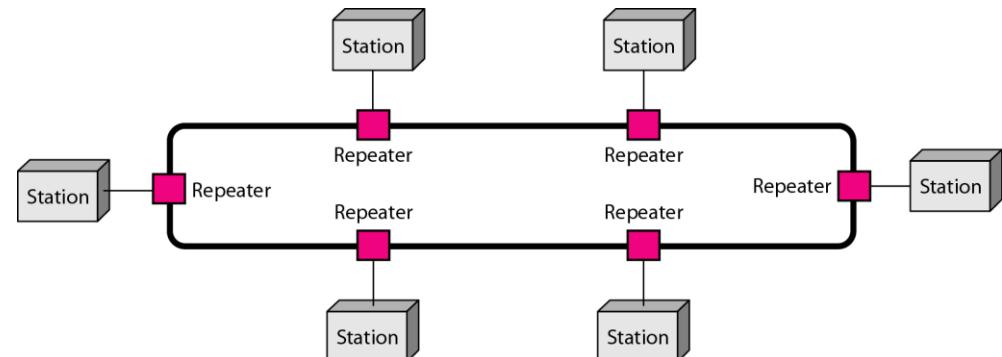
Mesh



Bus



Star

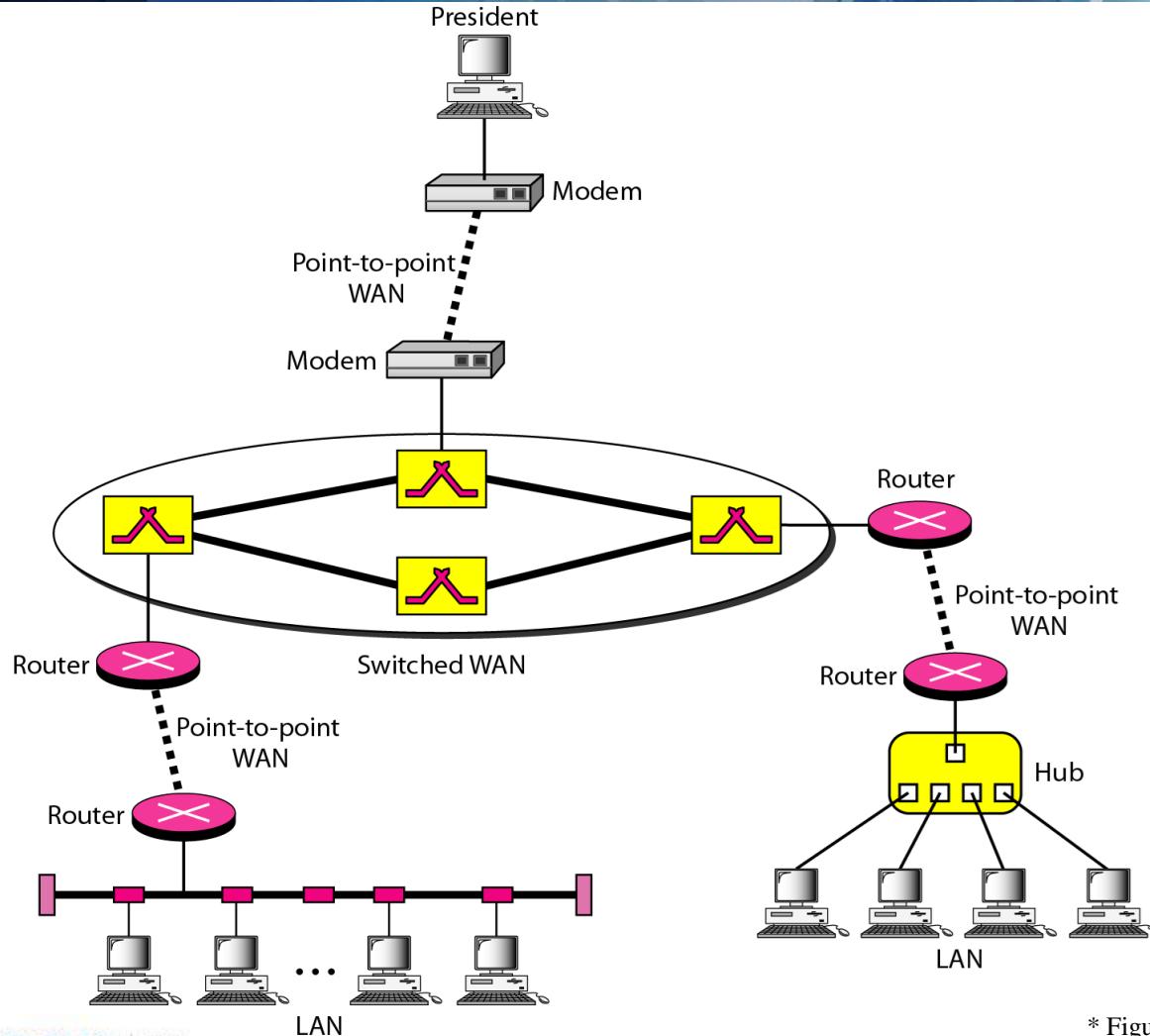


Ring

* Figure is courtesy of B. Forouzan



Combined Networks



* Figure is courtesy of B. Forouzan



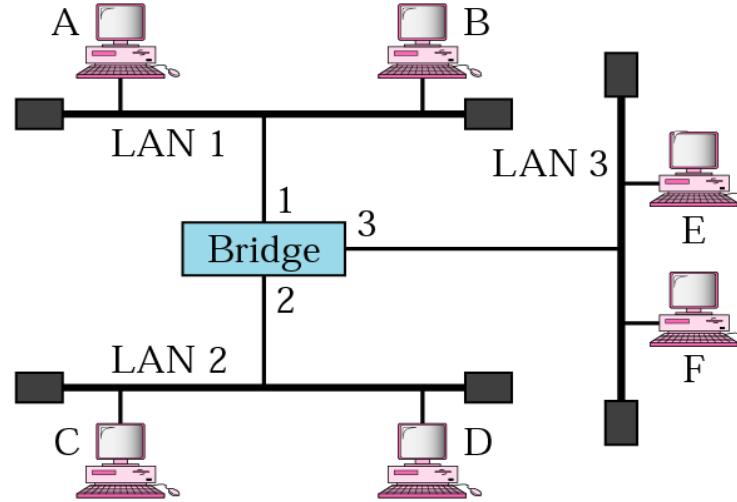
CS2031

Telecommunications II

Learning Bridges



Learning Bridges



Address	Port

a. Original

Address	Port
A	1

b. After A sends a frame to D

Address	Port
A	1
E	3

c. After E sends a frame to A

Address	Port
A	1
E	3
B	1

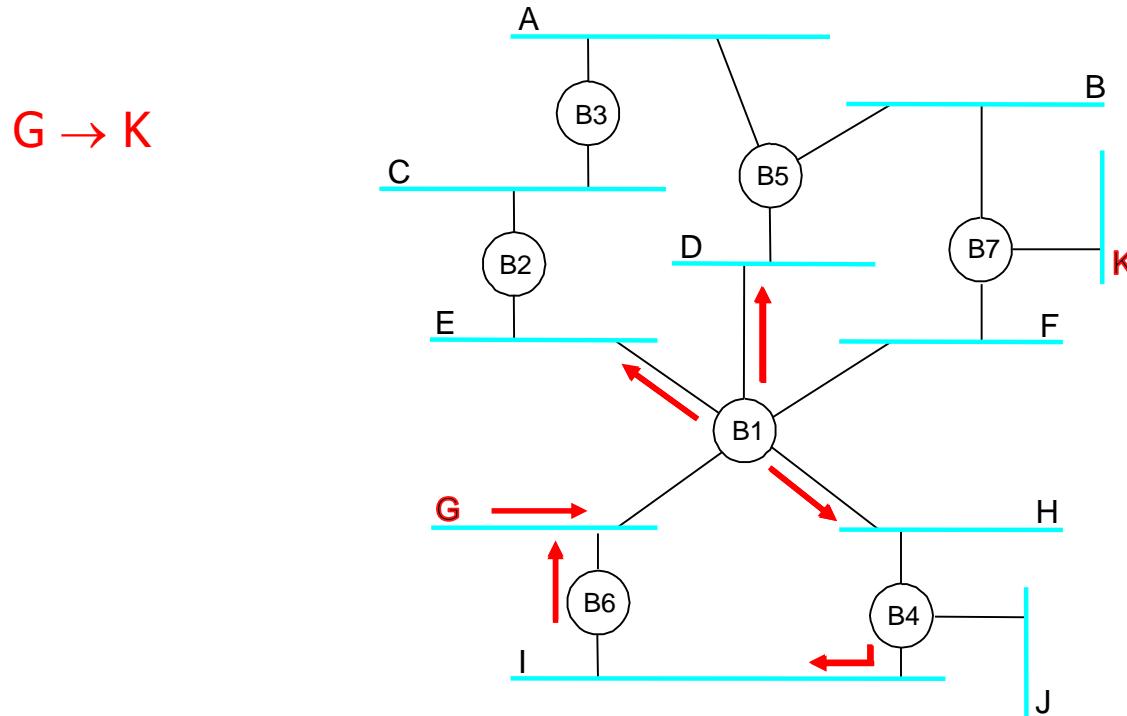
d. After B sends a frame to C

- Initially bridge forwards frames on all segments except incoming port
- Learns addresses from frames that pass through



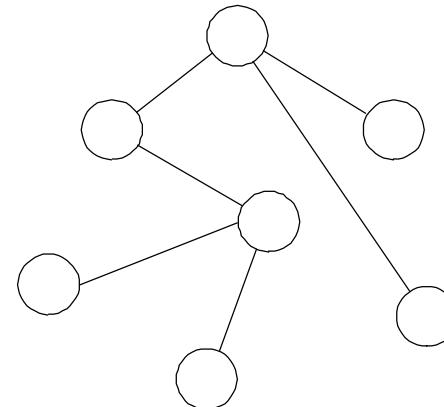
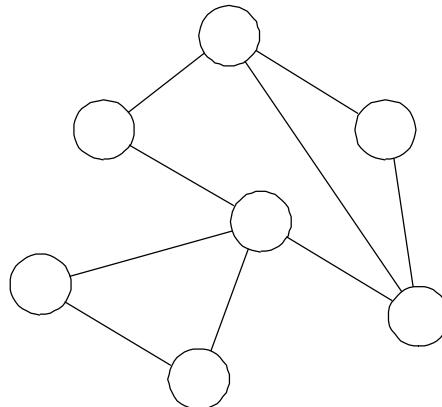
Problem with Learning Bridges

- Problem with learning bridges is loops



Spanning Tree Algorithm

- Bridges run a distributed spanning tree algorithm
 - select which bridges actively forward
 - developed by Radia Perlman
 - now IEEE 802.1 specification
- Spanning tree includes all nodes
- Spanning tree only includes edges on the shortest path from the root to any other given node

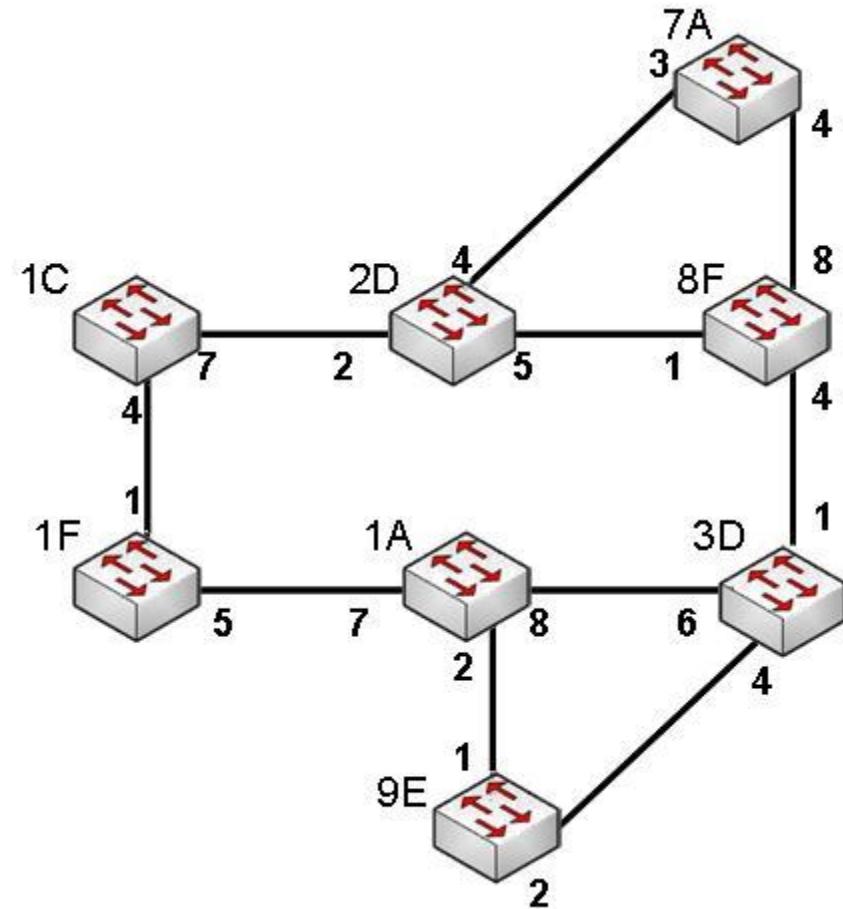


Spanning Tree Algorithm

1. Bridge with smallest ID is selected as root bridge
2. Mark port on each bridge with least-cost to root bridge as root port
3. Select designated bridge for each LAN that has root port with least-cost to root bridge – if two bridges with same cost select bridge with lowest ID
4. Mark root ports and designated ports as forwarding ports; other ports as blocking ports

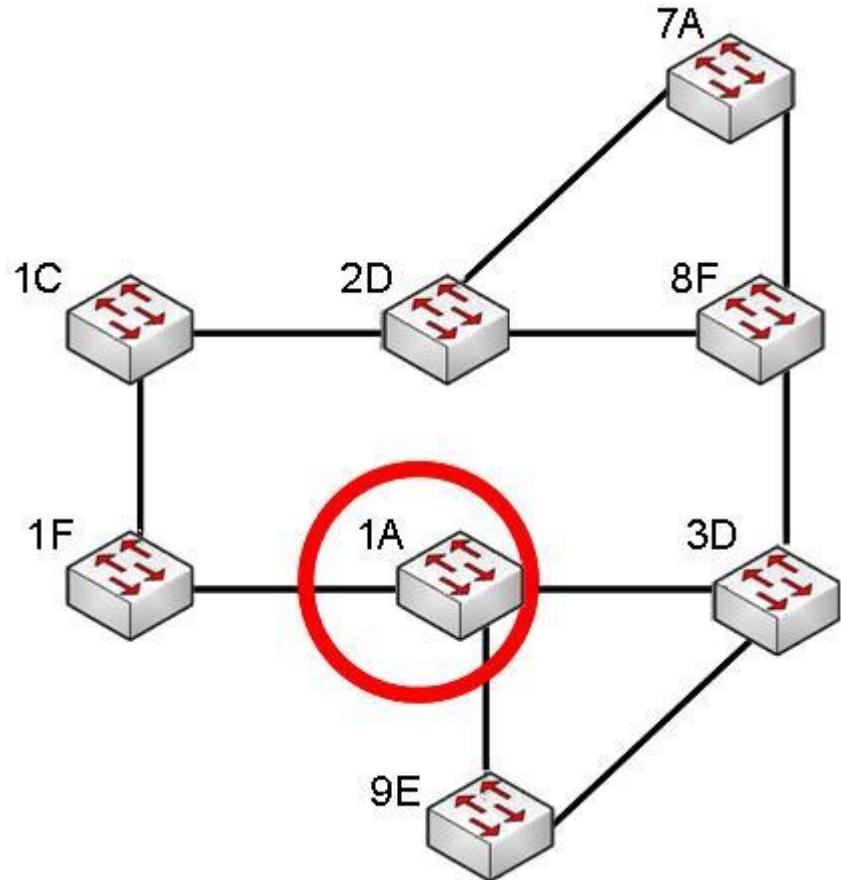
Spanning Tree Algorithm

1. Bridge with smallest ID is selected as root bridge
2. Mark port on each bridge with least-cost to root bridge as root port
3. Select designated bridge for each LAN that has root port with least-cost to root bridge – if two bridges with same cost select bridge with lowest ID
4. Mark root ports and designated ports as forwarding ports; other ports as blocking ports



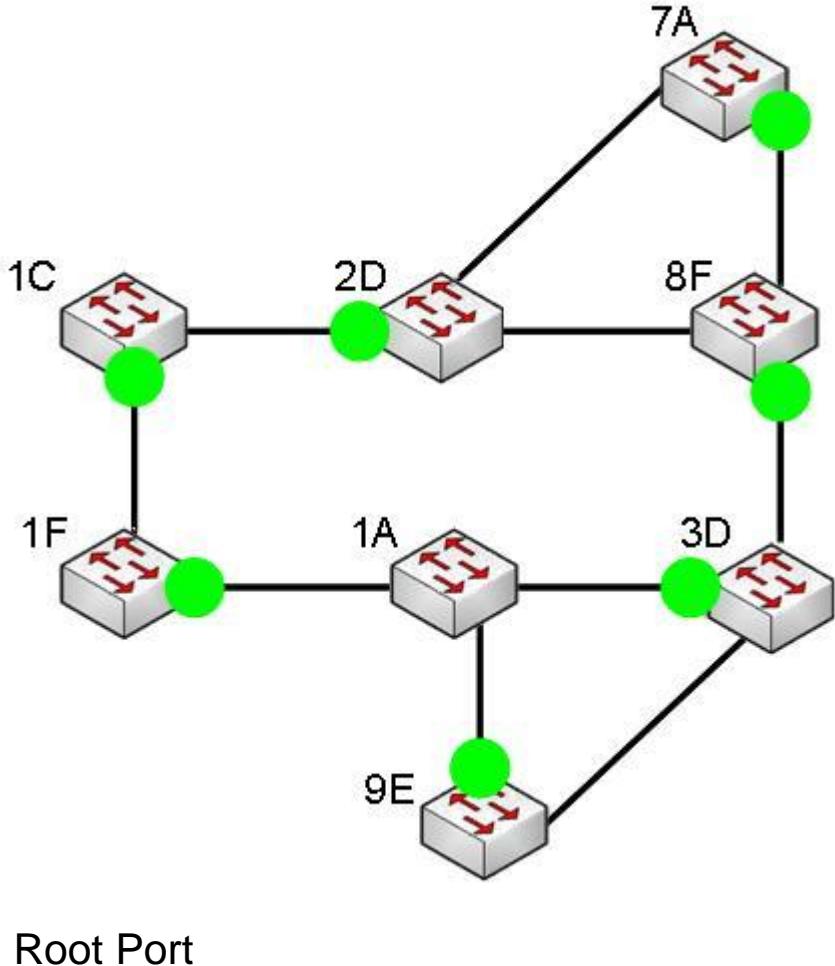
Spanning Tree Algorithm

- Bridge with smallest ID is selected as root bridge



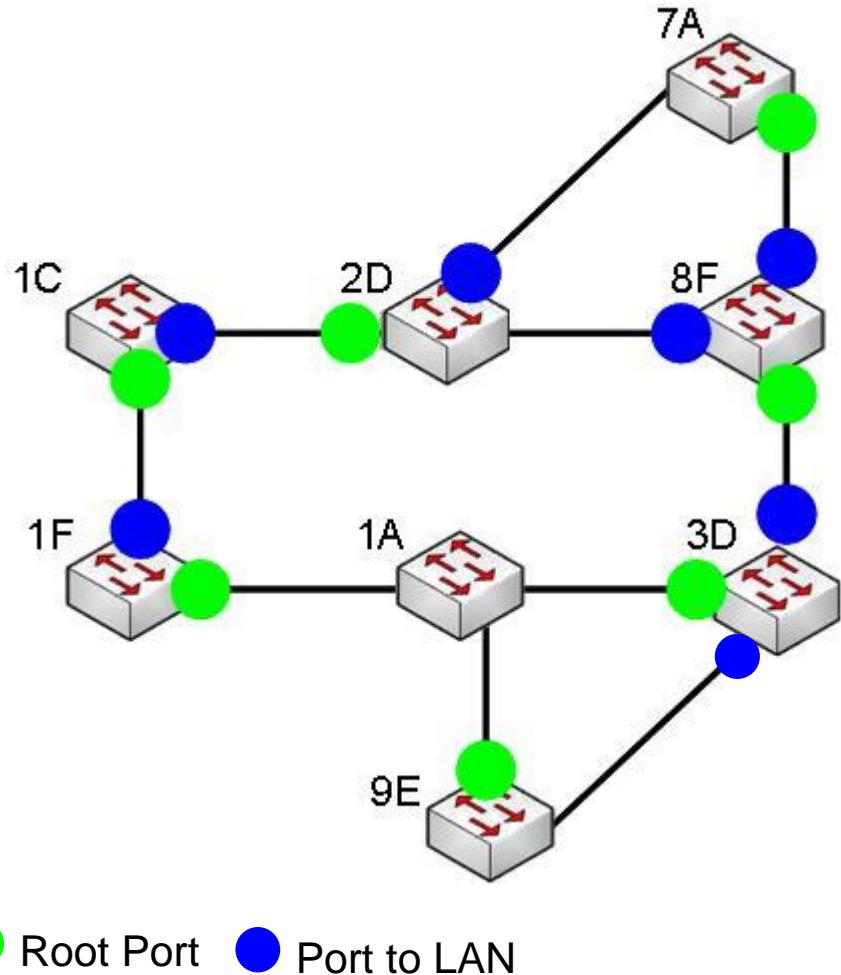
Spanning Tree Algorithm

1. Bridge with smallest ID is selected as root bridge
2. Mark port on each bridge with least-cost to root bridge as root port



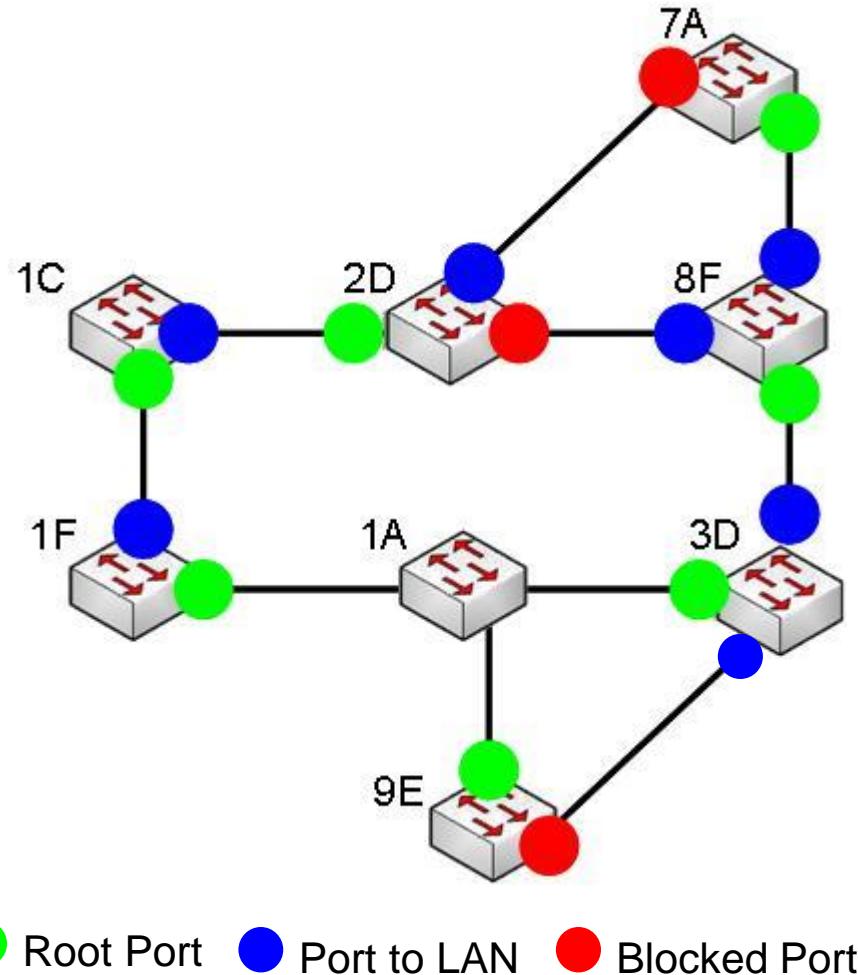
Spanning Tree Algorithm

1. Bridge with smallest ID is selected as root bridge
2. Mark port on each bridge with least-cost to root bridge as root port
3. Select designated bridge for each LAN that has root port with least-cost to root bridge – if two bridges with same cost select bridge with lowest ID



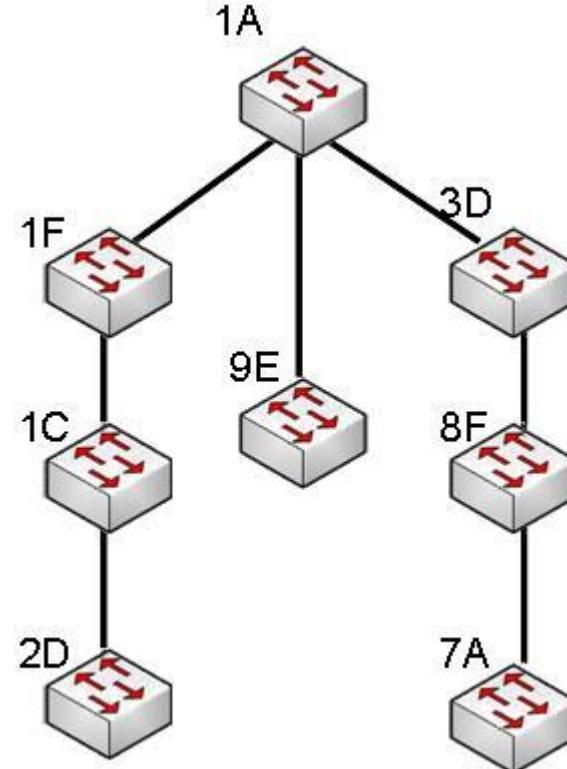
Spanning Tree Algorithm

1. Bridge with smallest ID is selected as root bridge
2. Mark port on each bridge with least-cost to root bridge as root port
3. Select designated bridge for each LAN that has root port with least-cost to root bridge – if two bridges with same cost select bridge with lowest ID
4. Mark root ports and designated ports as forwarding ports; other ports as blocking ports



Spanning Tree Algorithm

1. Bridge with smallest ID is selected as root bridge
2. Mark port on each bridge with least-cost to root bridge as root port
3. Select designated bridge for each LAN that has root port with least-cost to root bridge – if two bridges with same cost select bridge with lowest ID
4. Mark root ports and designated ports as forwarding ports; other ports as blocking ports





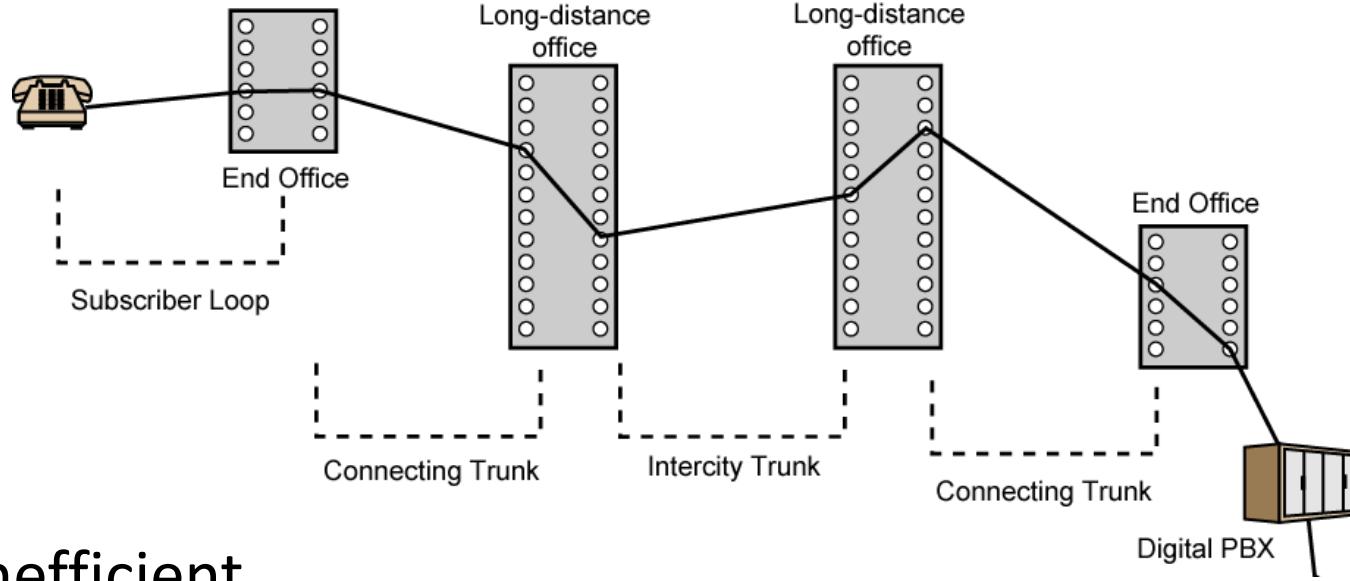
CS2031

Telecommunications II

Circuit Switching & Packet Switching

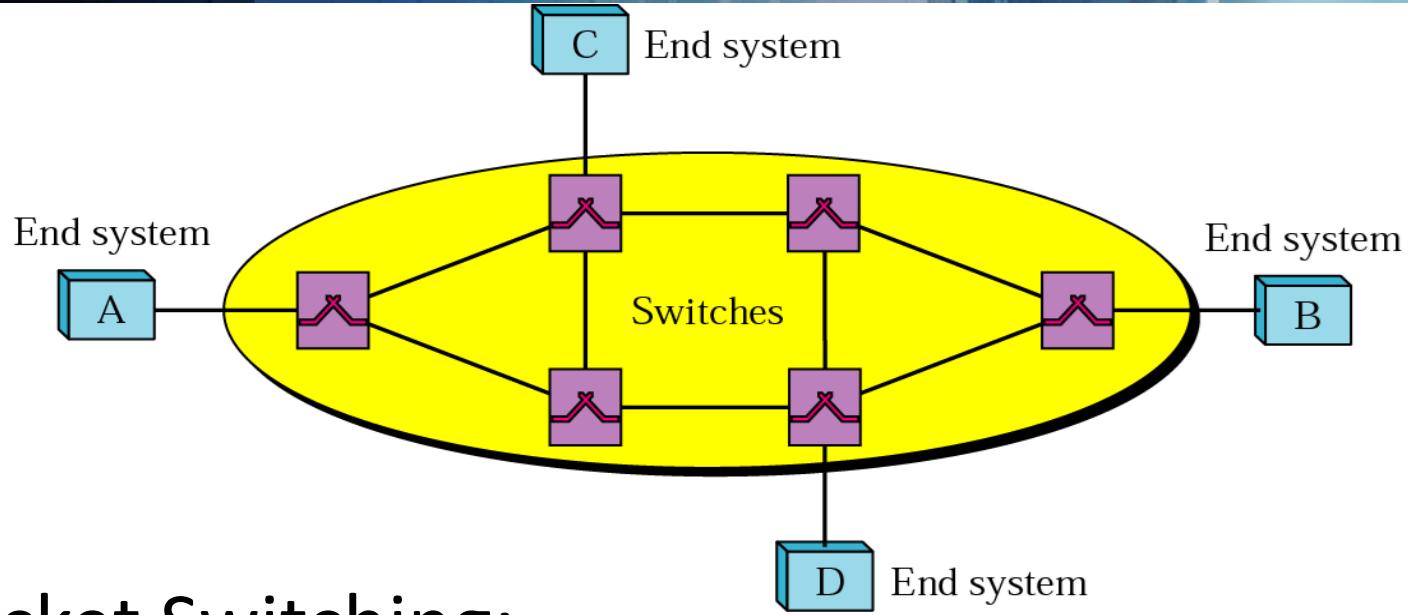


Public Circuit Switched Network



- Inefficient
 - Channel capacity dedicated for duration of connection
 - If no data, capacity wasted
- Set up of connection takes time
- Once connected, transfer is transparent

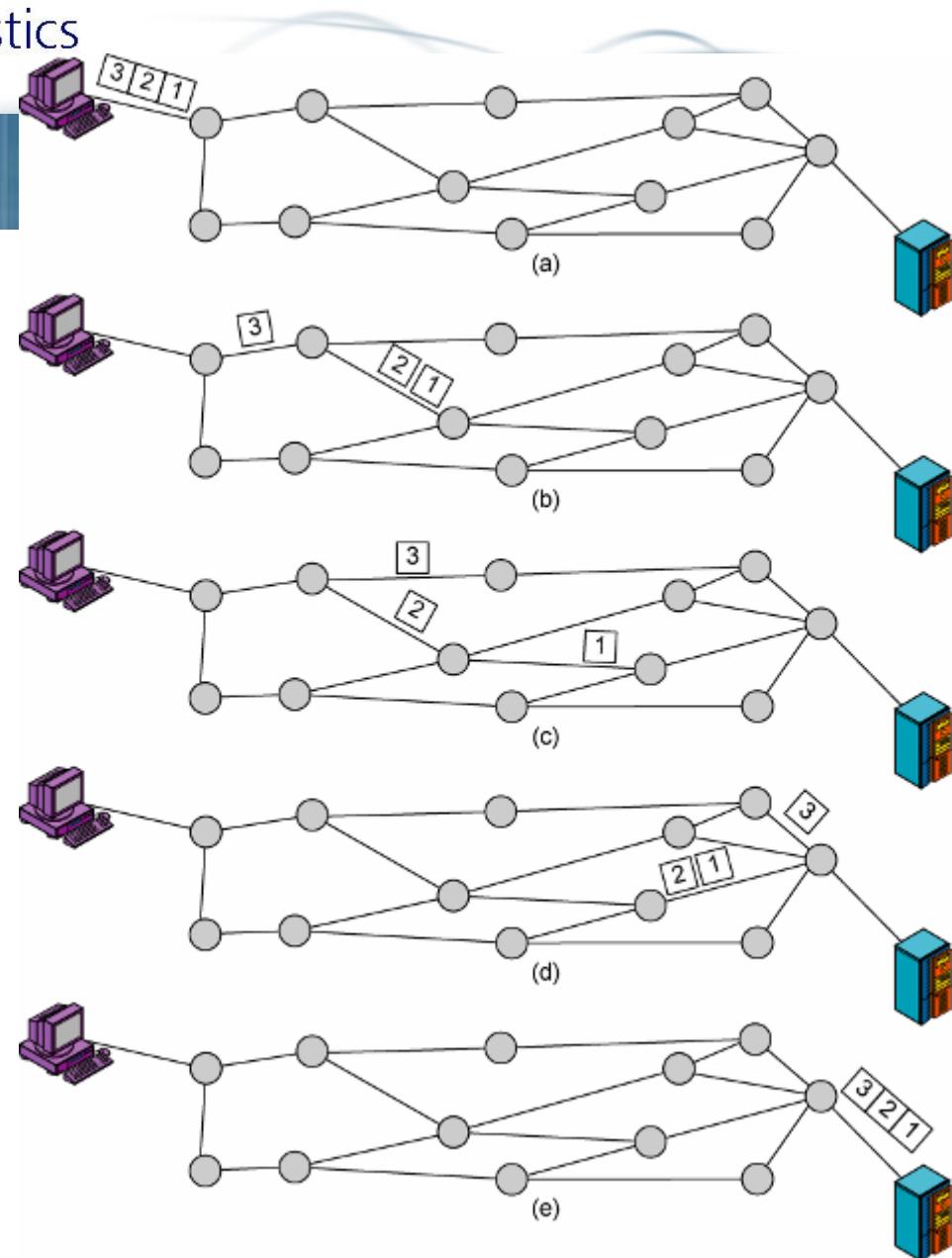
Switched Networks



- **Packet Switching:**
 - Switching decisions are made on individual packets
- **Virtual Circuit Switching:**
 - A circuit is setup explicitly for individual connections

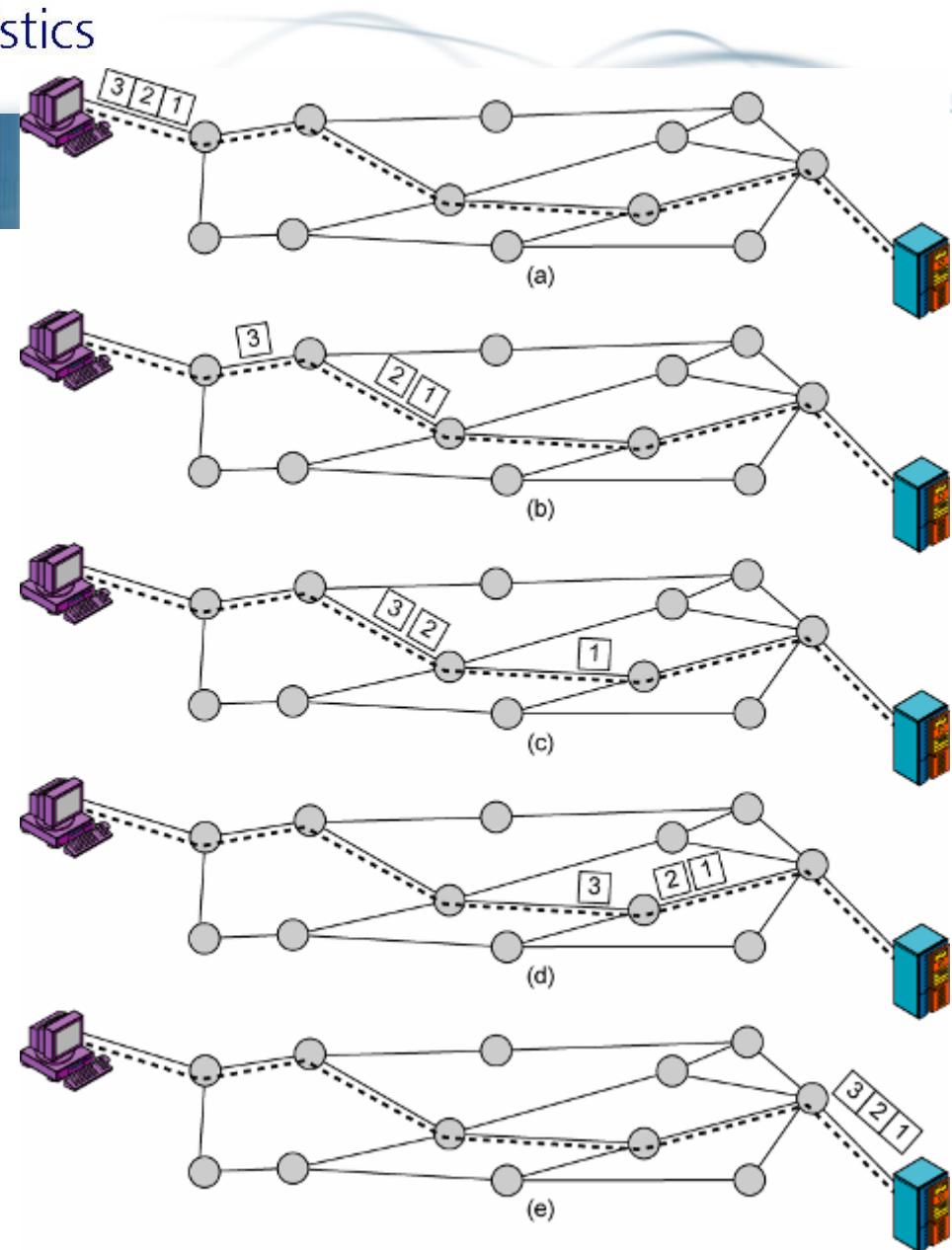
Packet switching

- Frames can be transferred over different paths in the network
- Reliability is generally delegated to higher layers
- Order is not necessarily maintained



Virtual Circuits

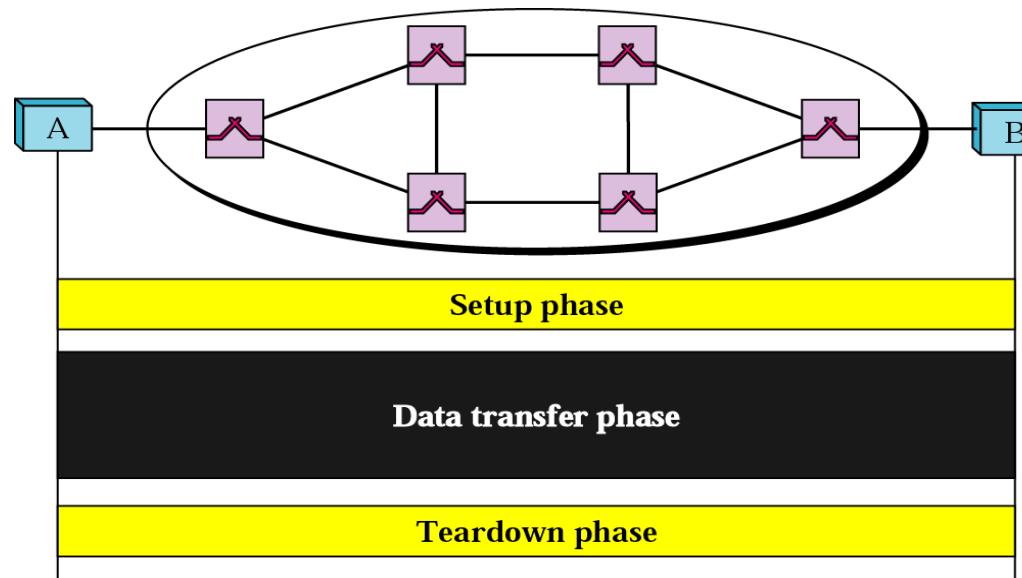
- Connection-oriented communication
- Connection is established before communication
- The network maintains order



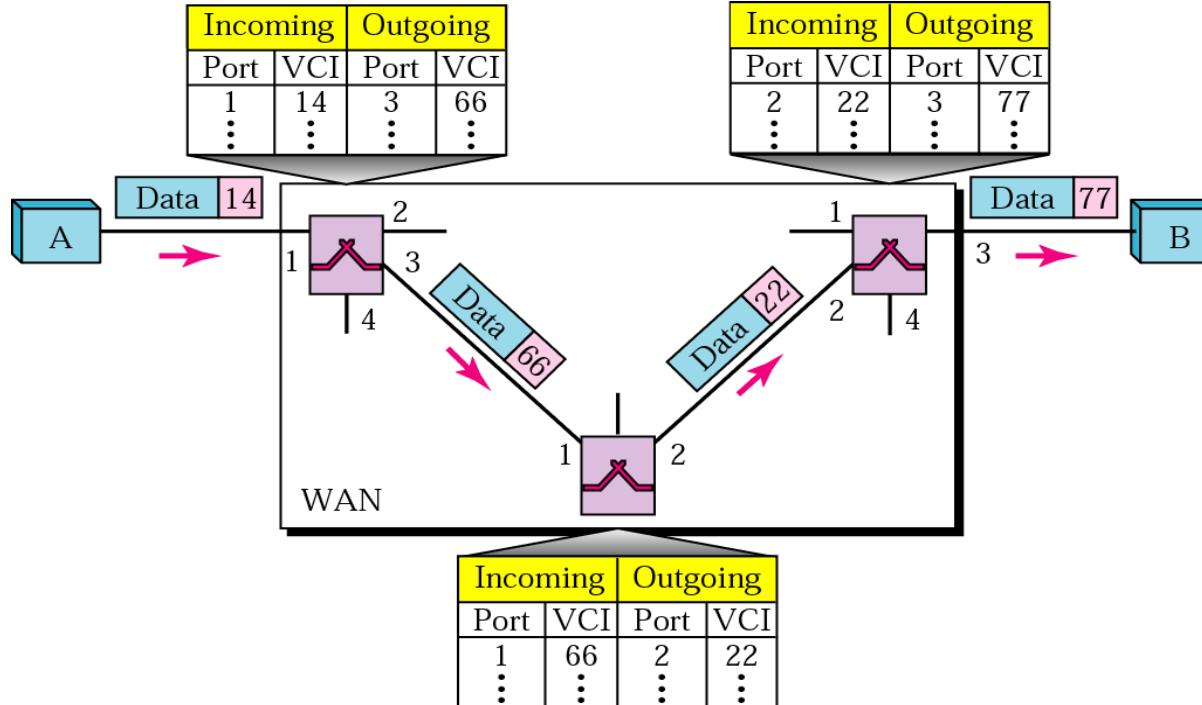
* Figure is courtesy of B. Forouzan

Phases for Virtual Circuit Communication

- Three phases
 - Connection setup
 - Data Transfer
 - Connection termination



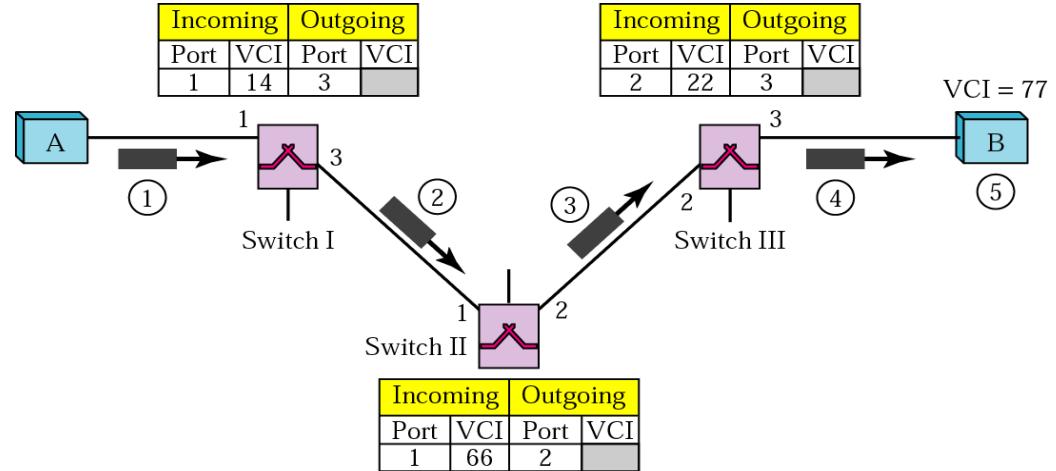
Virtual Circuit Switching



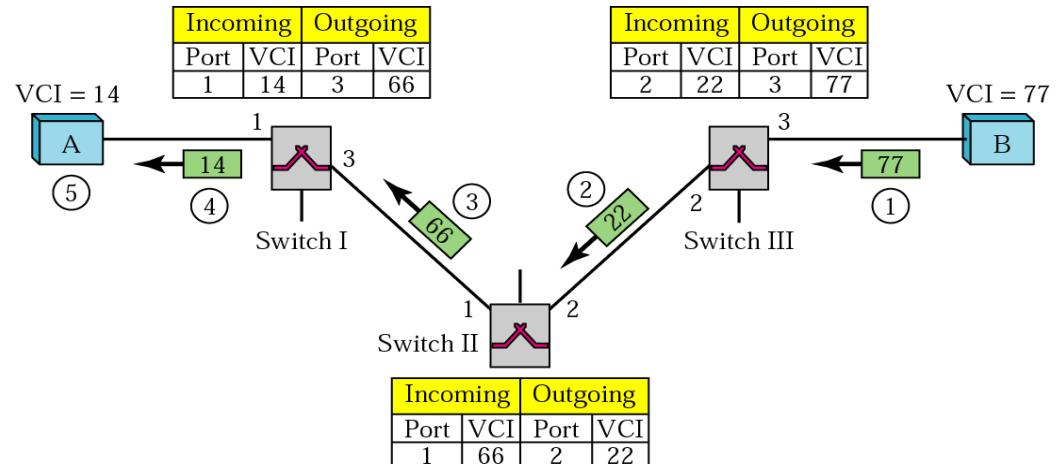
- Every switch maintains a table
 - For duration of communication one entry for incoming and outgoing line
 - Incoming and outgoing line are identified by port number and virtual circuit identifier

Setup Phase

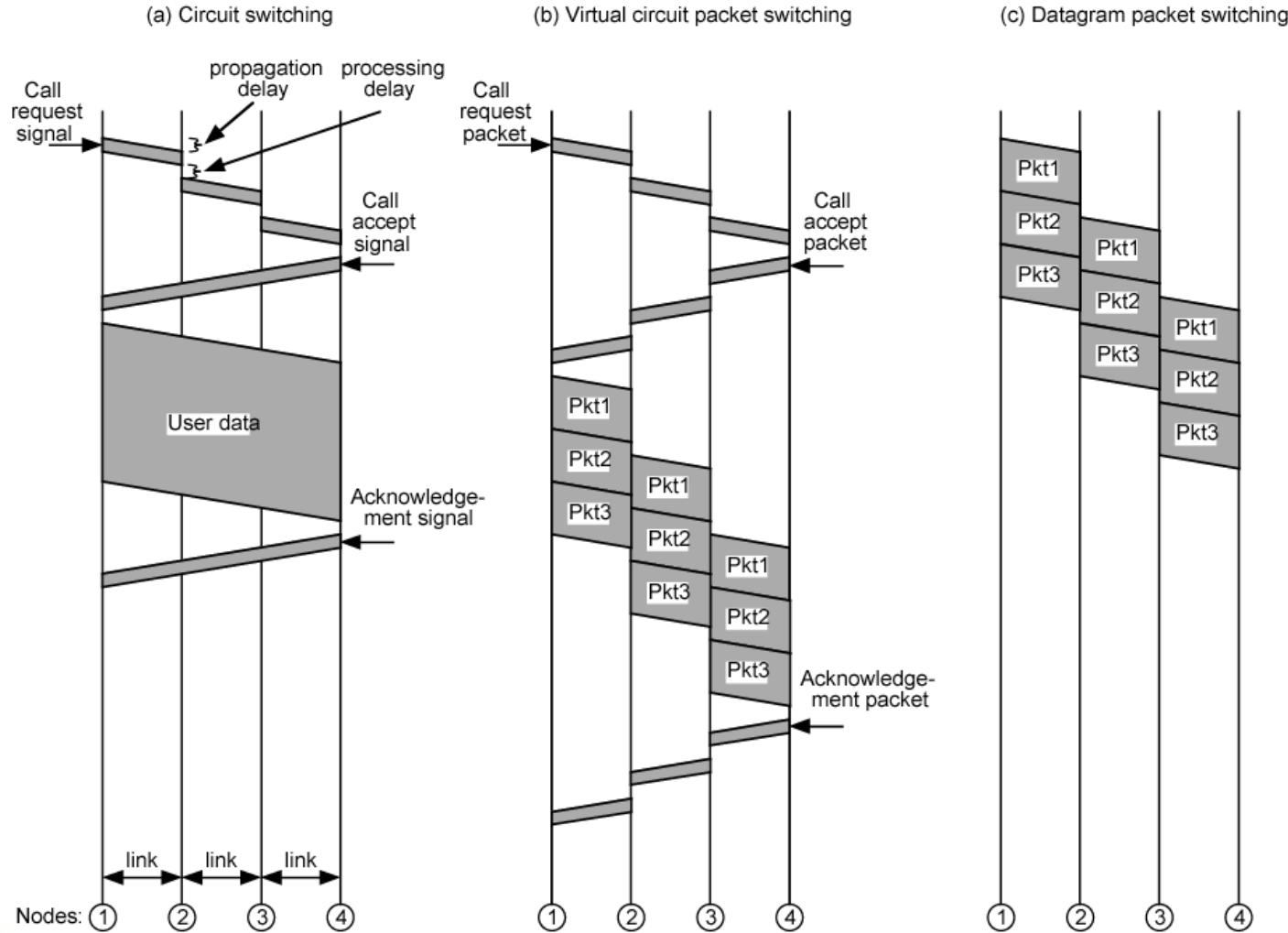
- Setup request



- Setup acknowl.



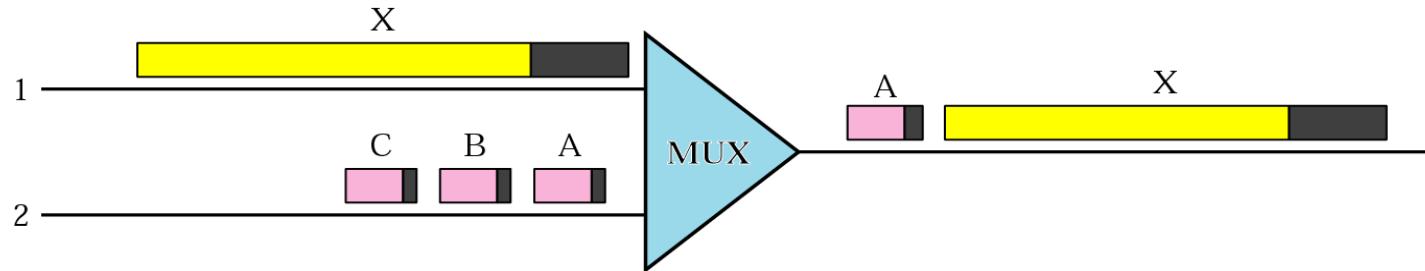
Event Timing



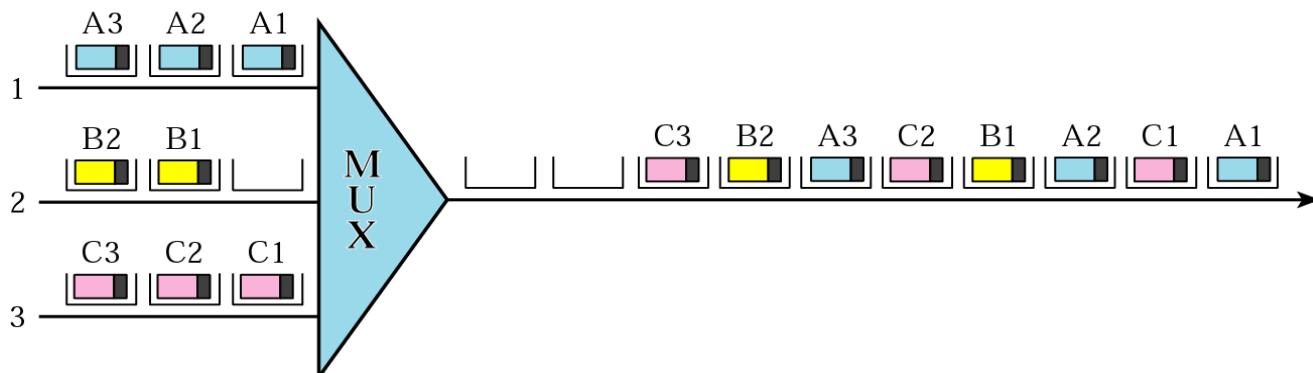
Asynchronous Transfer Mode (ATM)

- Example of virtual circuit switching
 - Cell-Switching
- Similarities between ATM and packet switching
 - Transfer of data in discrete chunks
 - Multiple logical connections over single physical interface
- In ATM flow on each logical connection is in fixed sized packets called cells
- Minimal error and flow control
 - Reduced overhead

Motivation for ATM

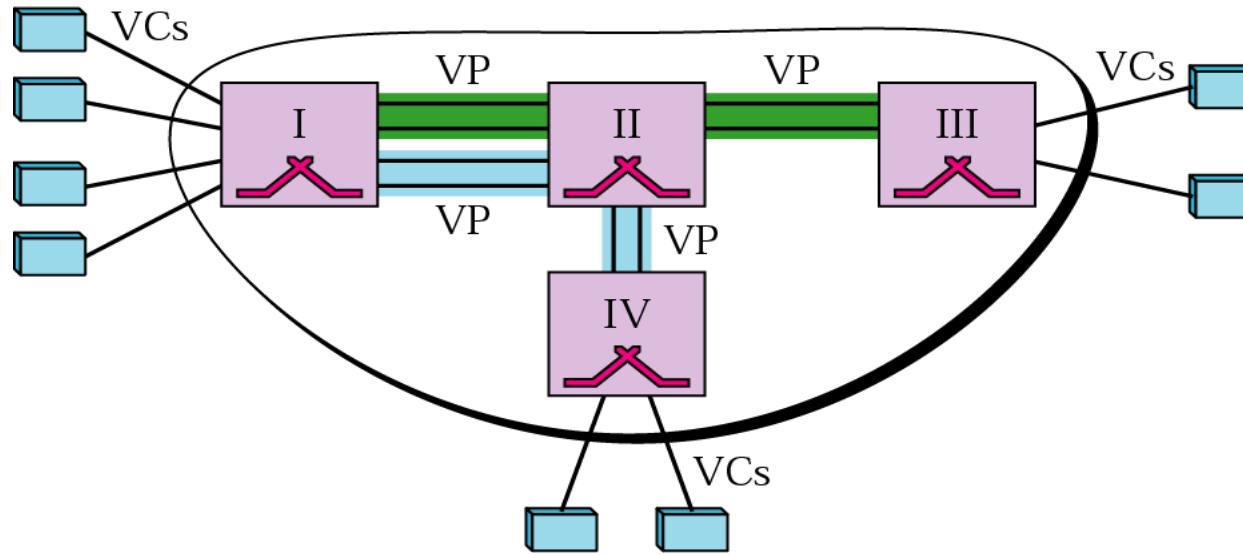


- Frames at a switch may be handled in any order and occupy switch for underspecified time



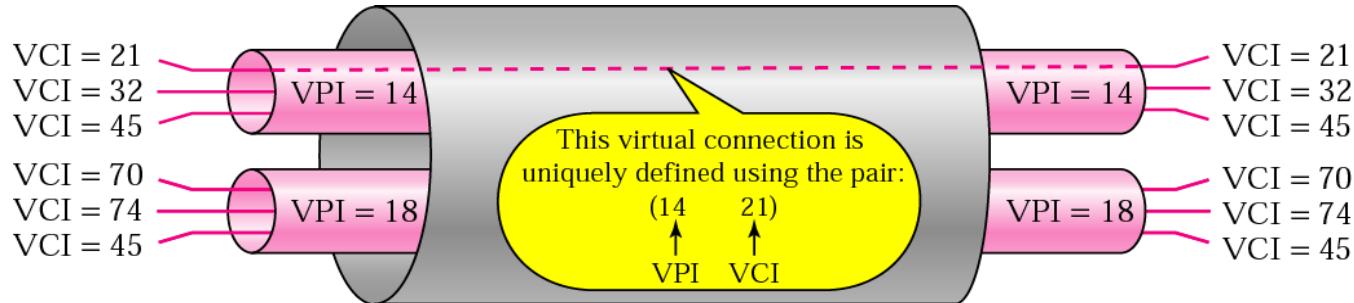
- Small, fixed-size frames allow simple, fast switches

Virtual Circuits / Virtual Paths

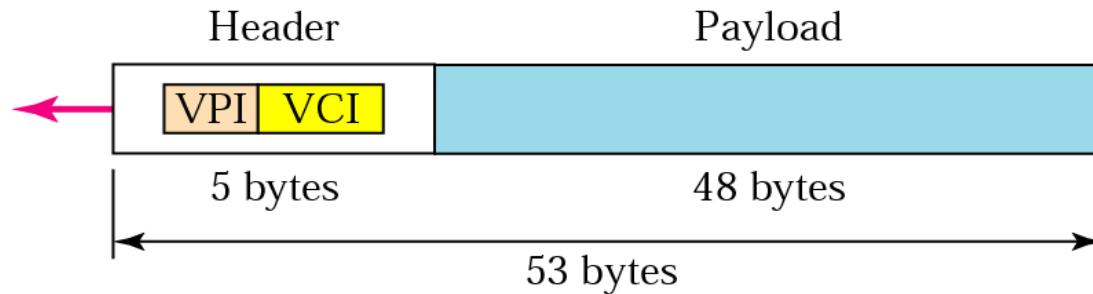


- Virtual circuits are collected into virtual paths

ATM Packet

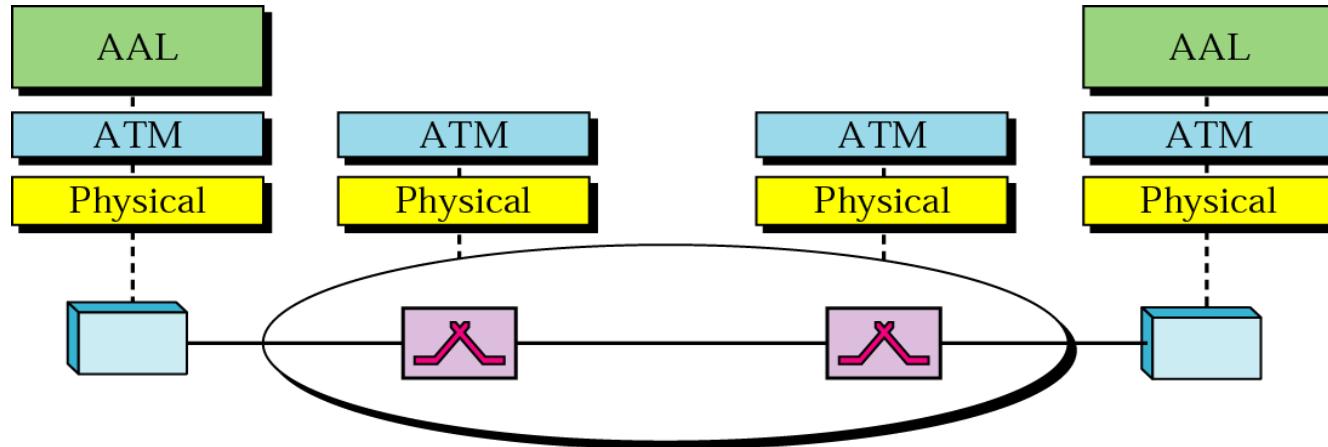


- Connection is specified by combination of Virtual Path ID and Virtual Circuit ID



- Every frame is exactly 53 bytes

Application Adaptation Layer (AAL)



- ATM defined a number of AALs for various purposes (each has its own header format):
 - AAL1: Constant bit rate e.g. multimedia
 - AAL2: Variable-data-rate
 - AAL3/4: Connection-oriented data services
 - Sequencing and Error Control
 - AAL5: Simple and efficient adaptation layer (SEAL)

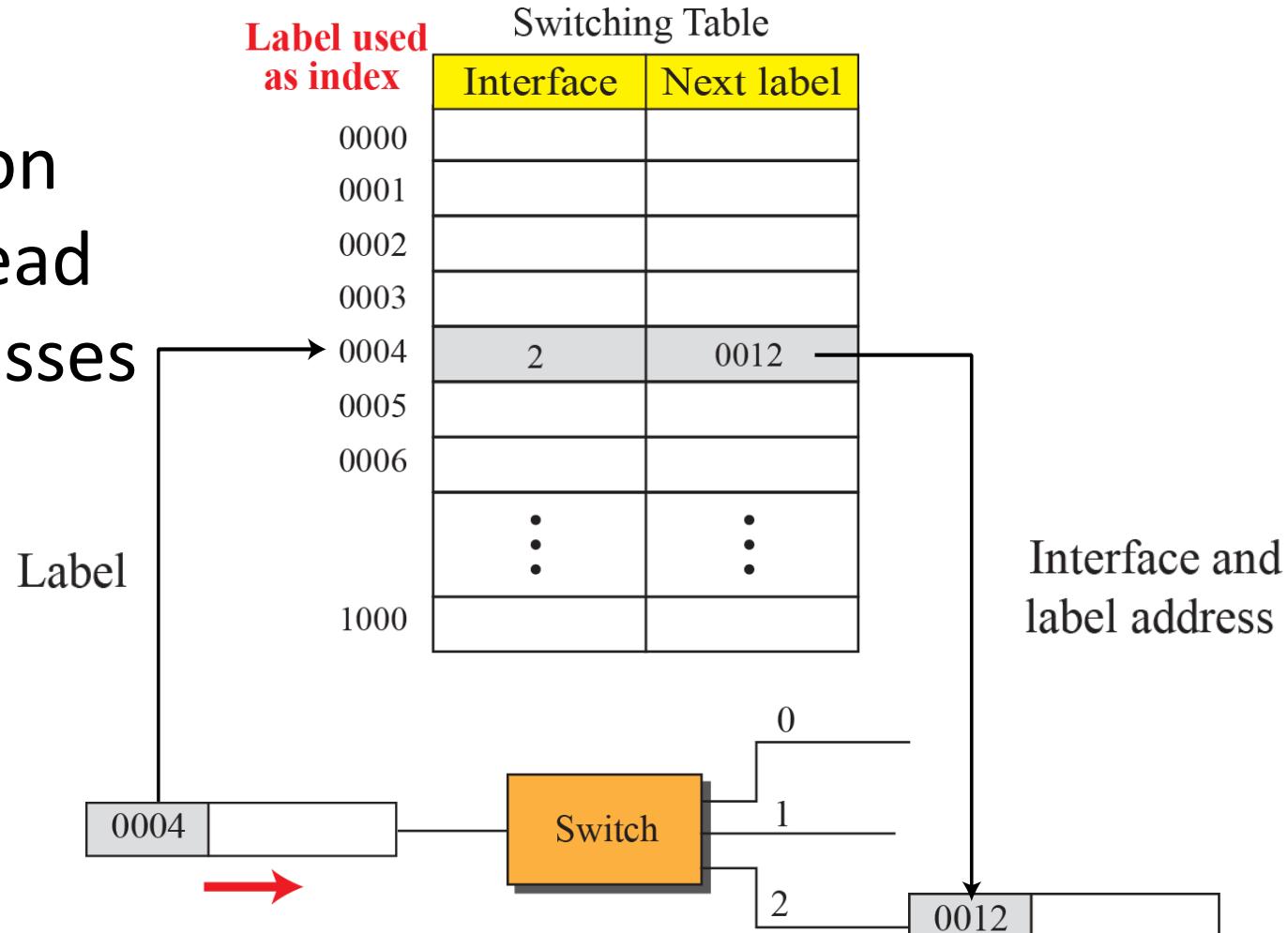
ATM – It Didn't Happen

- From Tanenbaum:

“ATM was going to solve all the world’s networking and telecommunications problems by merging voice, data, cable television, telex, telegraph, carrier pigeons, ...”
- It didn’t happen:
 - Bad Timing
 - Technology
 - Implementation
 - Politics

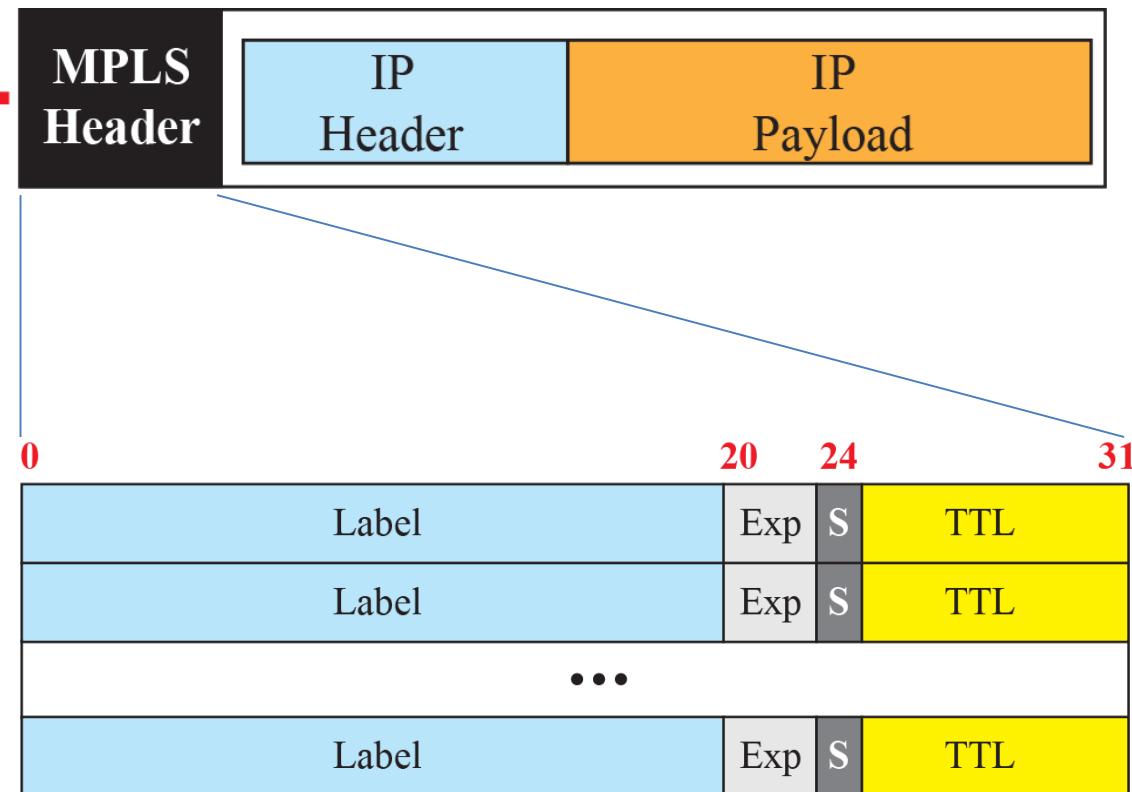
Multiprotocol Label Switching (MPLS)

- Enables switching on labels instead of IP addresses

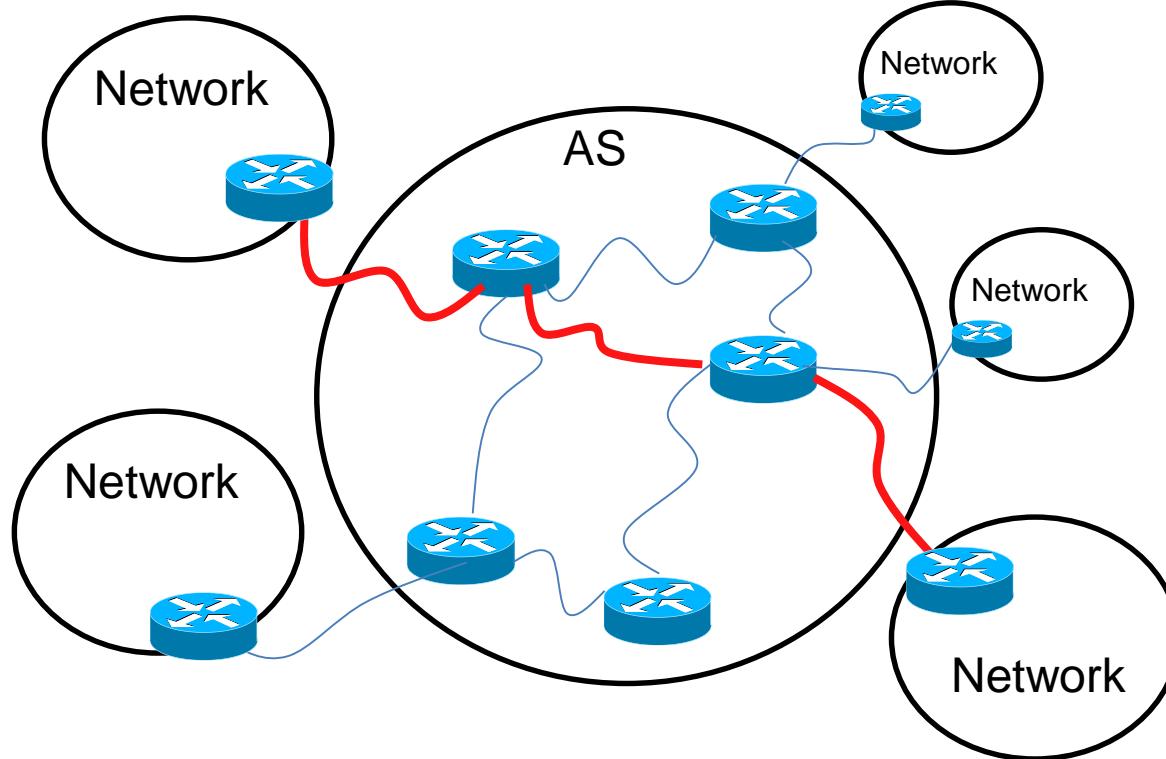


MPLS Header

- MPLS header as stack of labels

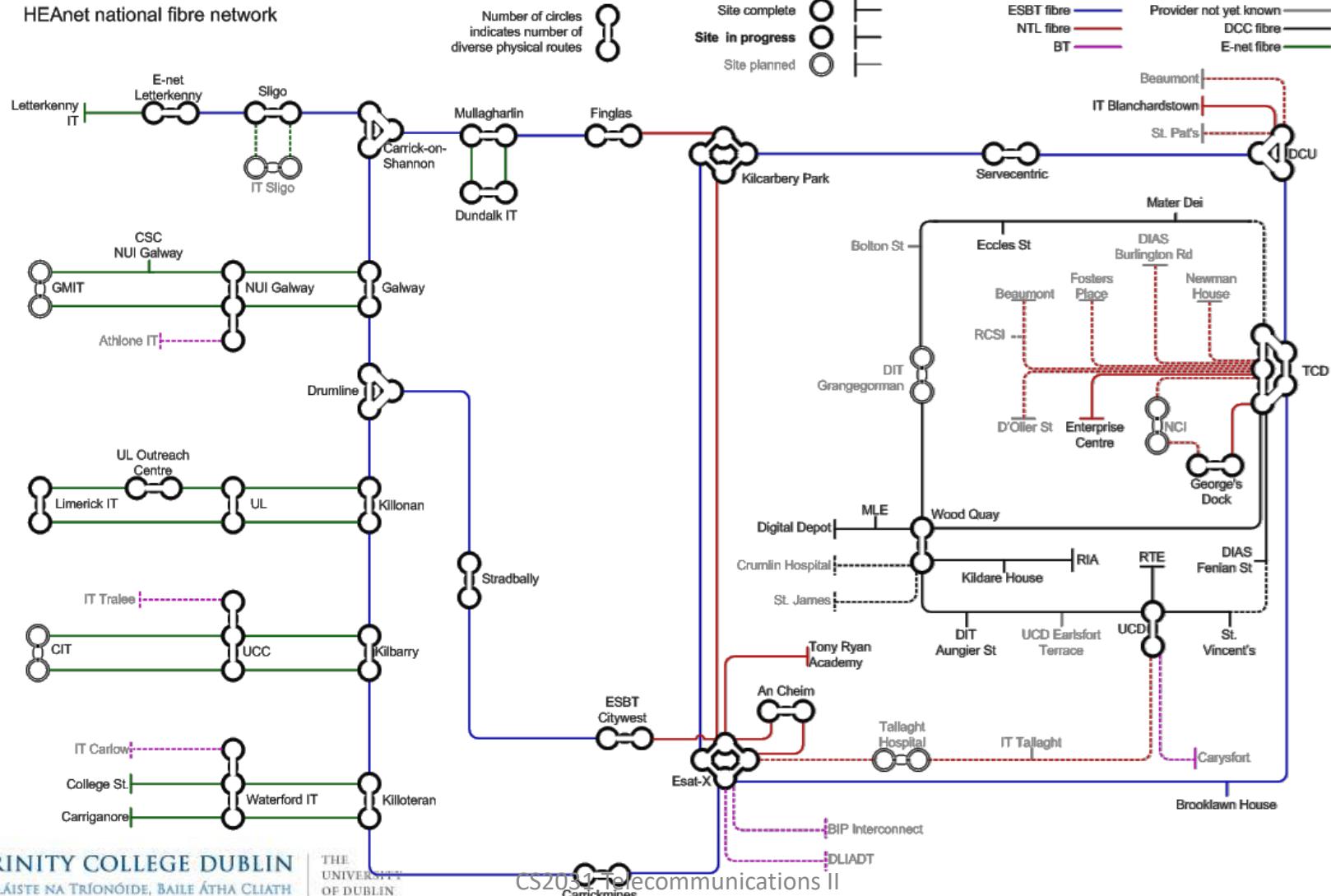


MPLS Use Case



- Creating a virtual network

HEAnet Fibre Network



Summary:Virtual Circuit Switching – ATM

- Virtual Circuit Switching
 - Preplanned route established before any frames sent
 - Call request and call accept frames establish connection (handshake)
 - Each frame contains a virtual circuit identifier instead of destination address
 - No routing decisions required for each frame
 - Clear request to drop circuit
 - Not a dedicated path
- Asynchronous Transfer Mode (ATM)
 - Example for virtual circuit switching
 - Cells consist of 5-byte header and 48-byte payload
 - Circuits identified by virtual circuit ID and virtual path ID
 - Application adaptation layer (AAL) for specific application areas

Tutorial - Question 3

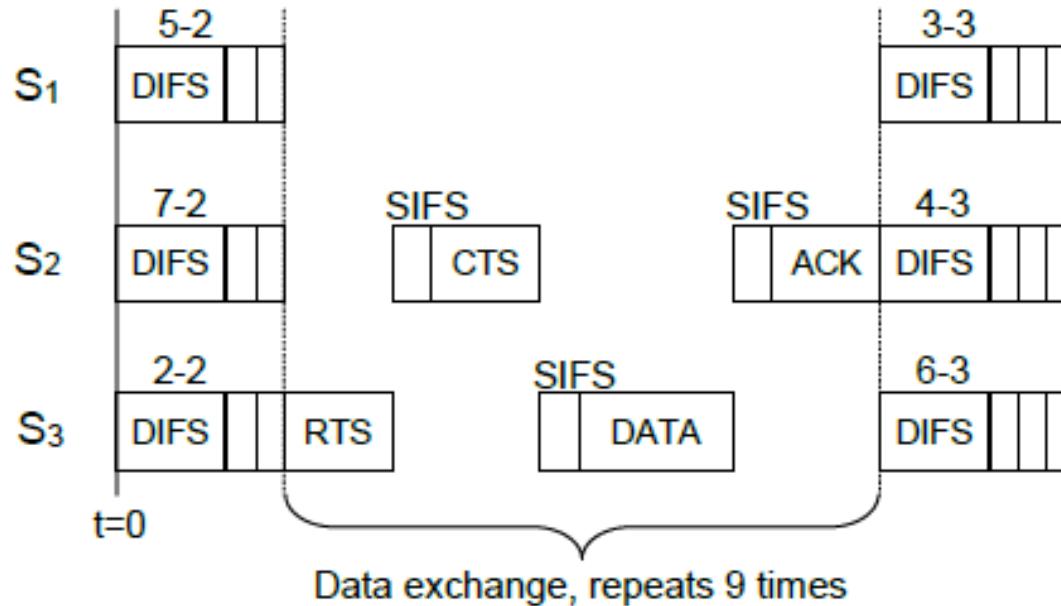
3 stations intent to transmit each 3 data frames

Transmission times for

- data frames and the Beacon: 190us
- RTS and Poll: 180us
- CTS, ACK and CF-End: 132us



DCF - Individual Transmissions



DIFS+ RTS+CTS+ data+ack+ 3*SIFS
= 34us+180us+132us+190us+132us+3*16us
= 716us

DCF – Backoff Procedure

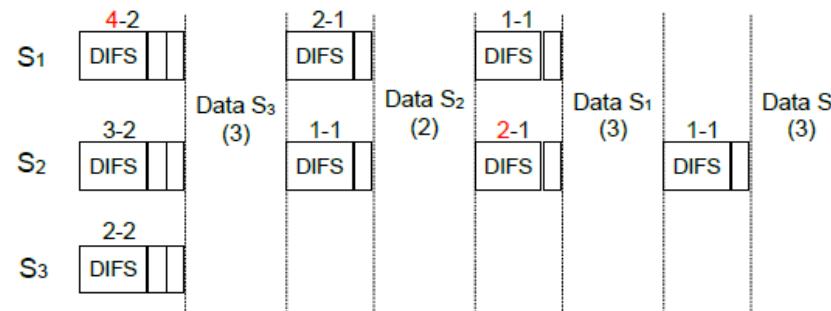
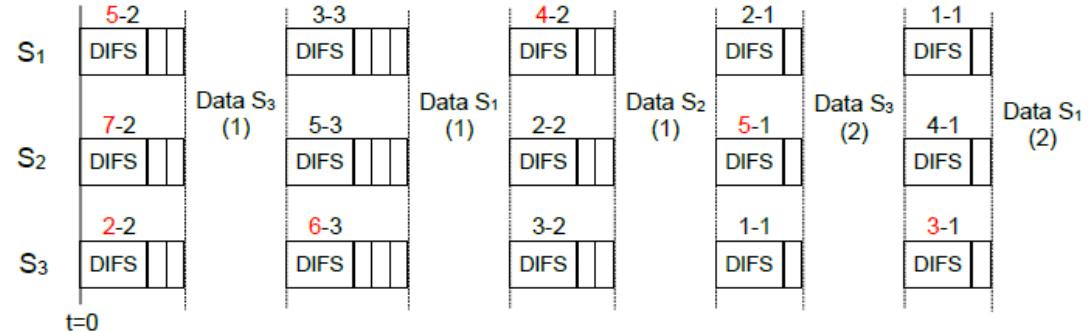
Step 1:

S₁: 5, 4

S₂: 7, 5, 2

S₃: 2, 6,

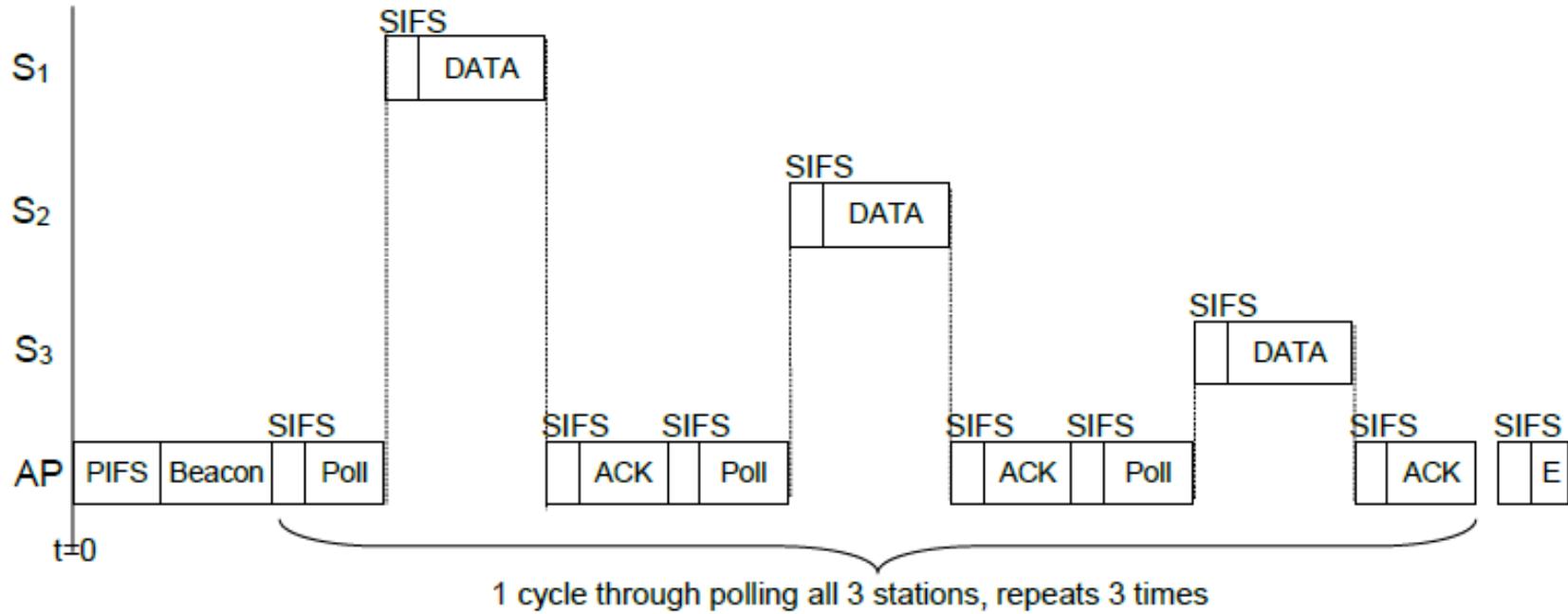
Step 2:



$$\begin{aligned}
 9 * T_x + 14 * \text{slots} &= 9 * 716\text{us} + 14 * 9\text{us} \\
 &= 6,570\text{us} = \sim 0.006\text{ms}
 \end{aligned}$$

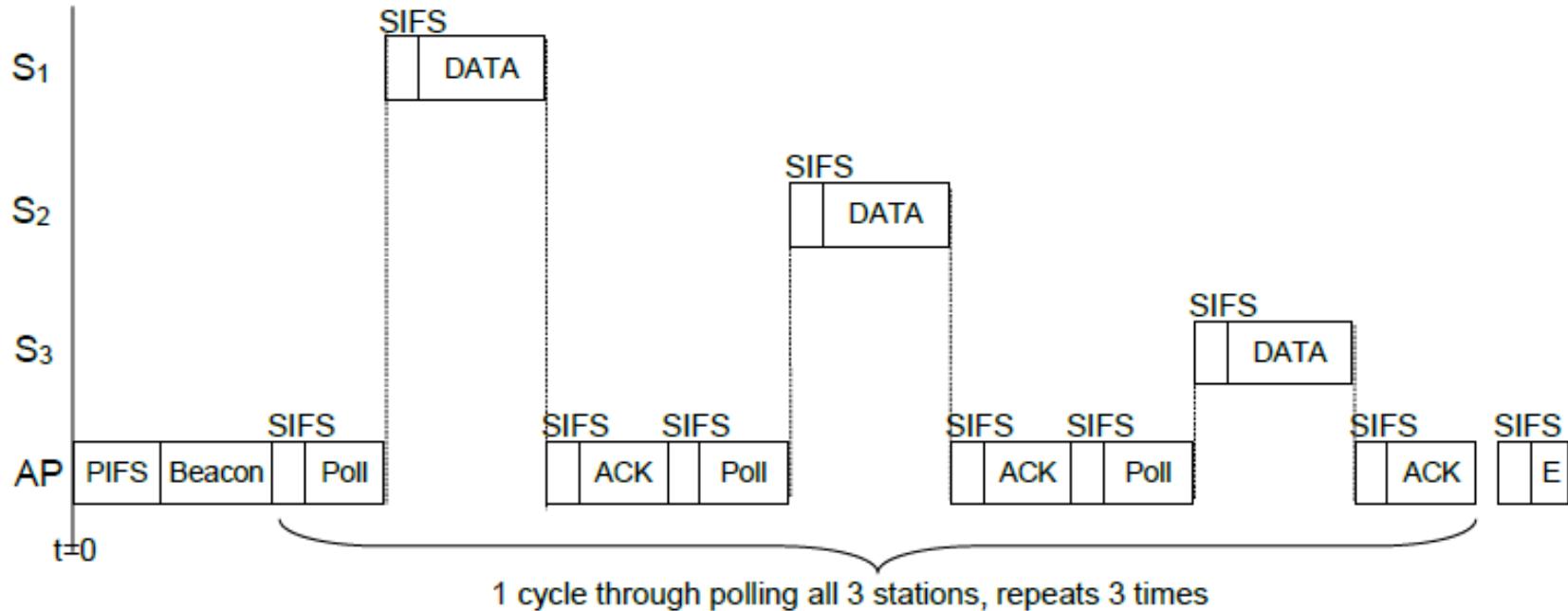


PCF – Round of Polls



$$\begin{aligned} & 9 \cdot \text{SIFS} + 3 \cdot \text{poll} + 3 \cdot \text{data} + 3 \cdot \text{ack} \\ & = 9 \cdot 16\text{us} + 3 \cdot 180\text{us} + 3 \cdot 190\text{us} + 3 \cdot 132\text{us} \\ & = 1,650\text{us} \end{aligned}$$

PCF – Round of Polls



$$\begin{aligned}
 & \text{PIFS} + \text{beacon} + 3^*Dx + \text{SIFS} + \text{CF-End} \\
 & = 25\text{us} + 190\text{us} + 3^*1650\text{us} + 16\text{us} + 132\text{us}
 \end{aligned}$$

$$= 5,313\text{us} = \sim 0.005\text{ms}$$

HOST-HOST Communication, 1970

- Paul Barran, [On Distributed Communication Networks](#), IEEE Transactions on Communication Systems, Volume 12, No. 1, March 1964
- Introduced Concepts of
 - Distributed Networks
 - Routing
(hot-potato-routing)
 - Packet-Switching
(message-block)
- Name worth mentioning: Bob Kahn

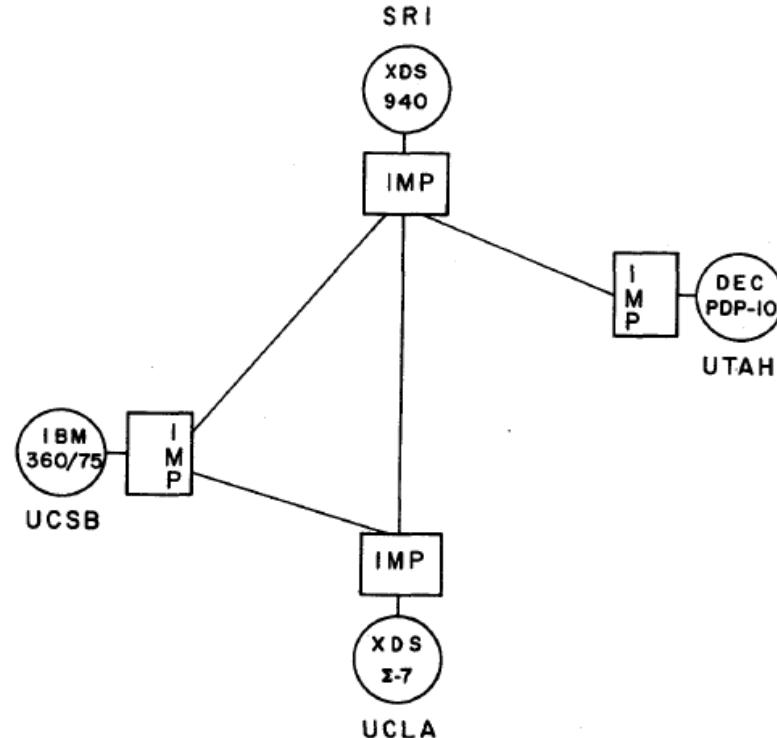
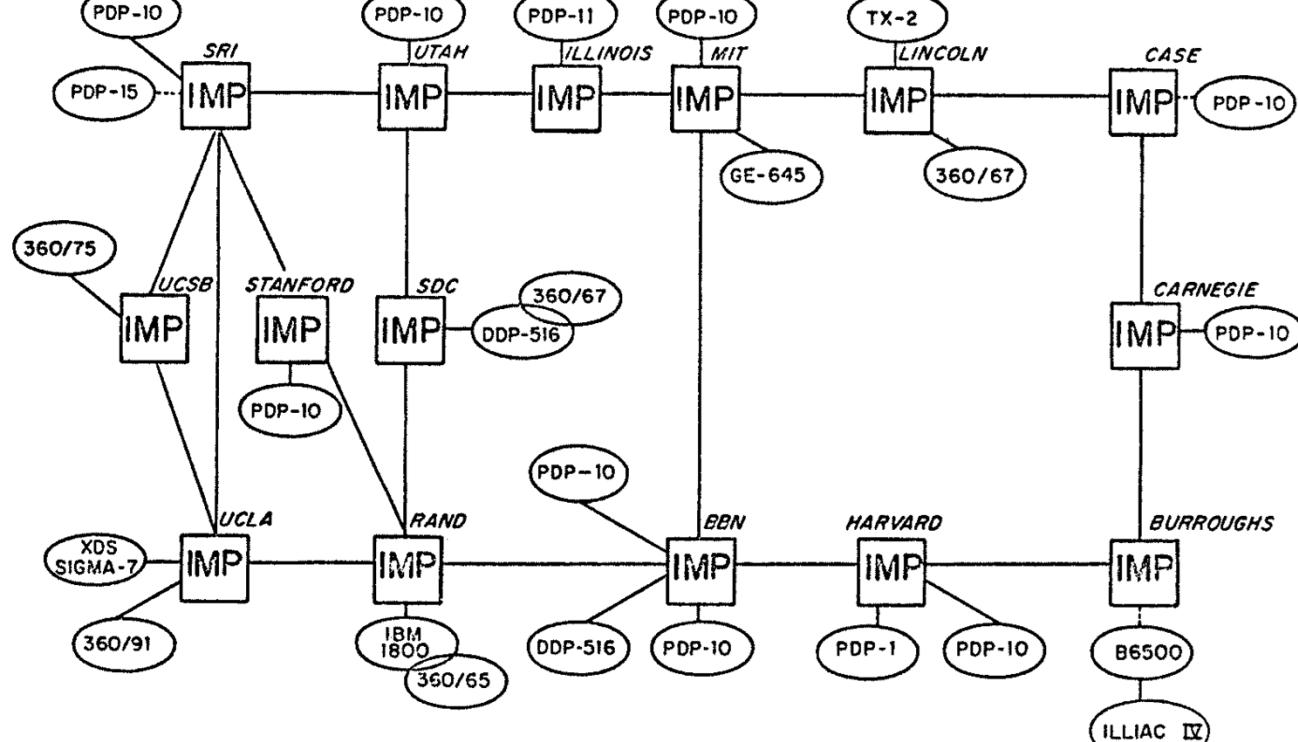


Figure 1—Initial network configuration

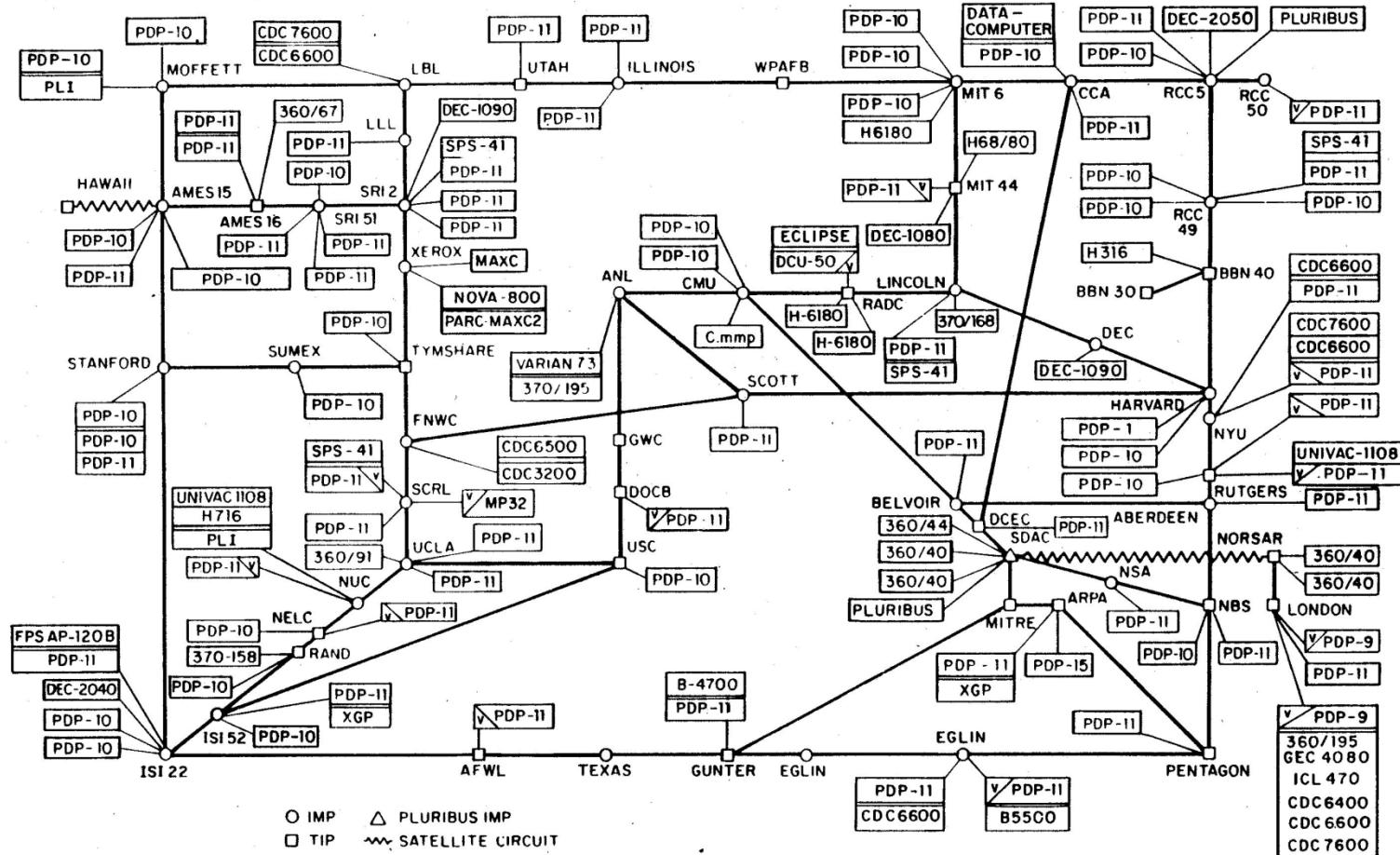
* C. Stephen Carr, Stephen D. Crocker, and Vinton G. Cerf, [HOST-HOST Communication Protocol in the ARPA Network](#), Spring Joint Computer Conference, pages 589-597, Atlantic City, NJ, USA, May 1970

AT&T's comment was that packet-switching will never be useful.

ARPANET 1971



ARPANET 1977



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY.)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES



IP Addresses

10000000 00001011 00000011 00011111

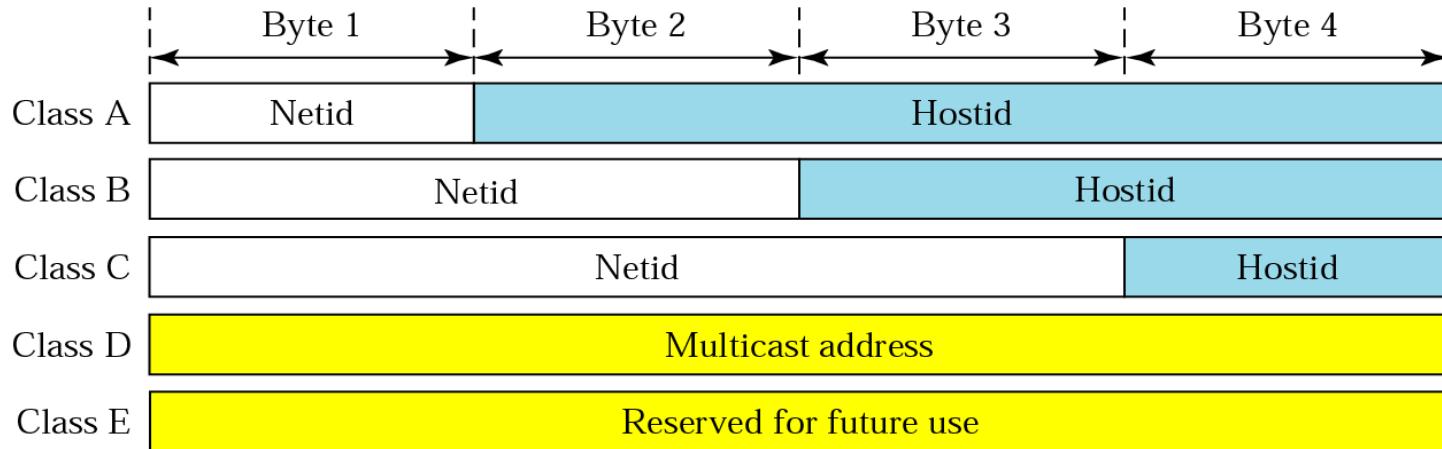
128.11.3.31

- 32-bit number
 - 4.294.967.296 addresses
- IP addresses are unique and universal
 - except private addresses

Range	Total
10.0.0.0 to 10.255.255.255	2^{24}
172.16.0.0 to 172.31.255.255	2^{20}
192.168.0.0 to 192.168.255.255	2^{16}

- Dotted decimal notation:
 - Bytes of binary notation represented as decimal separated by dot

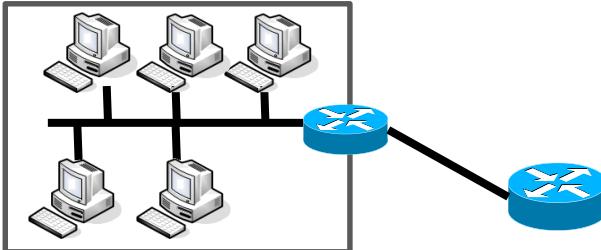
Classful Addresses



- Class A (international organisations)
 - 126 networks with 16,277,214 hosts each
- Class B (large companies)
 - 16,384 networks with 65,354 hosts each
- Class C (smaller companies)
 - 2,097,152 networks with 254 hosts each
- Inefficient use of hierarchical address space
 - Class C with 2 hosts ($2/254 = 0.78\%$ efficient)
 - Class B with 256 hosts ($256/65534 = 0.39\%$ efficient)

Network Layer

Trinity College



134.226.0.0



Class B



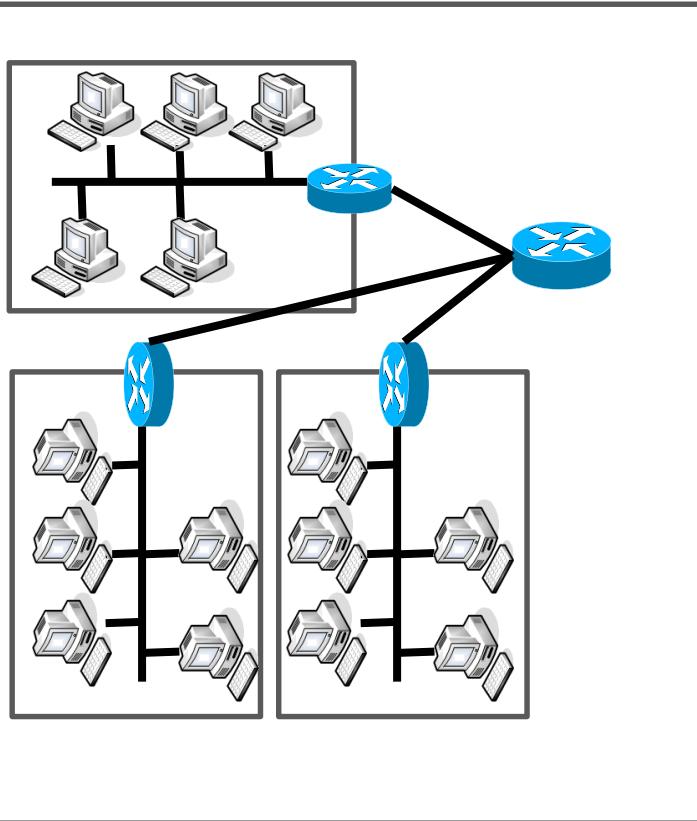
TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

THE
UNIVERSITY
OF DUBLIN

CS2031 Telecommunications II

Network Layer

Trinity College



134.226.0.0

←
Class B



TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

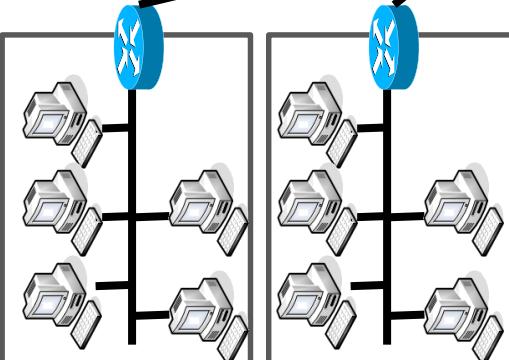
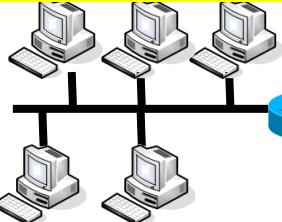
THE
UNIVERSITY
OF DUBLIN

CS2031 Telecommunications II

Network Layer

Trinity College

134.226.36.0



134.226.52.0

134.226.120.0

134.226.0.0

Class B ← Classful Addresses

CS2031 Telecommunications II

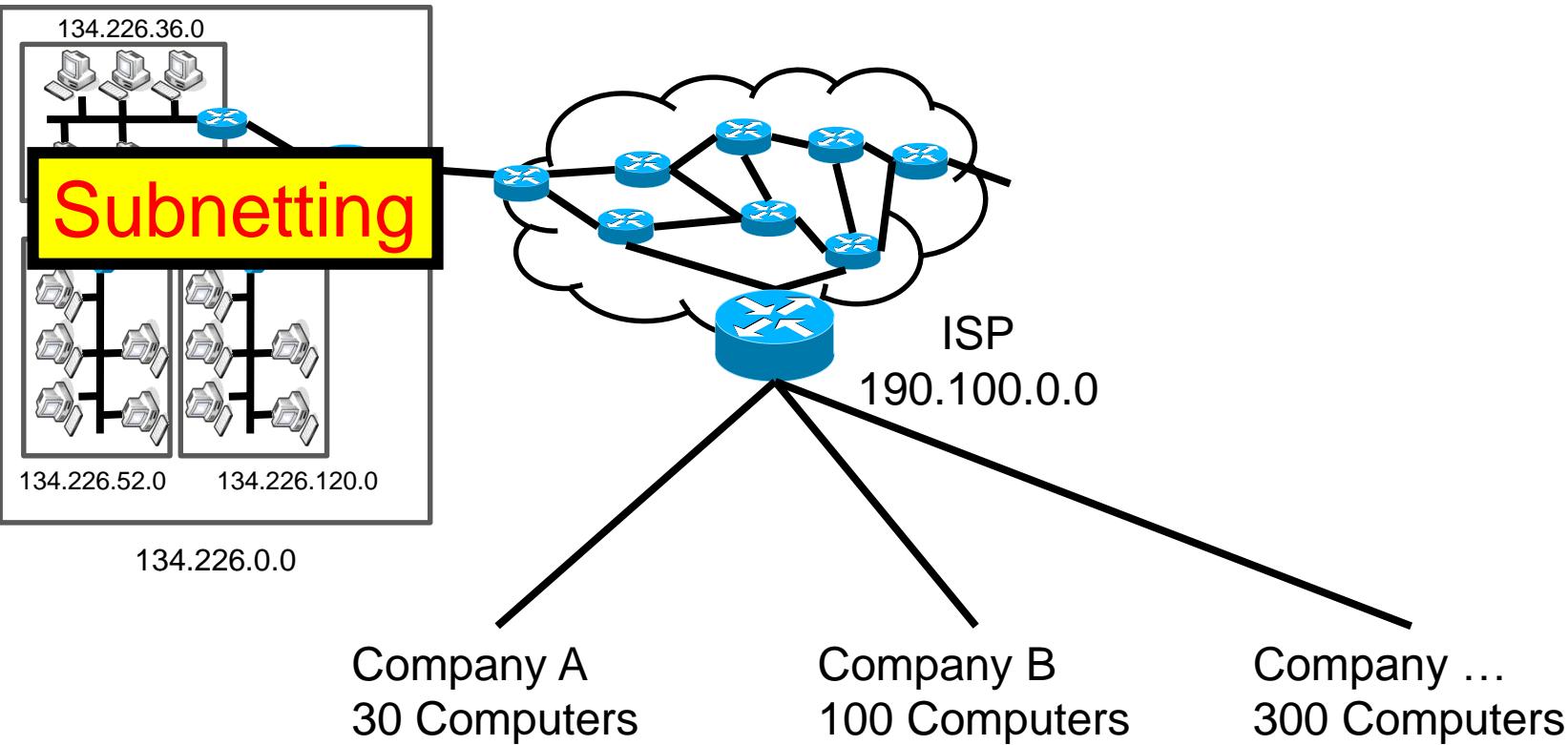


TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

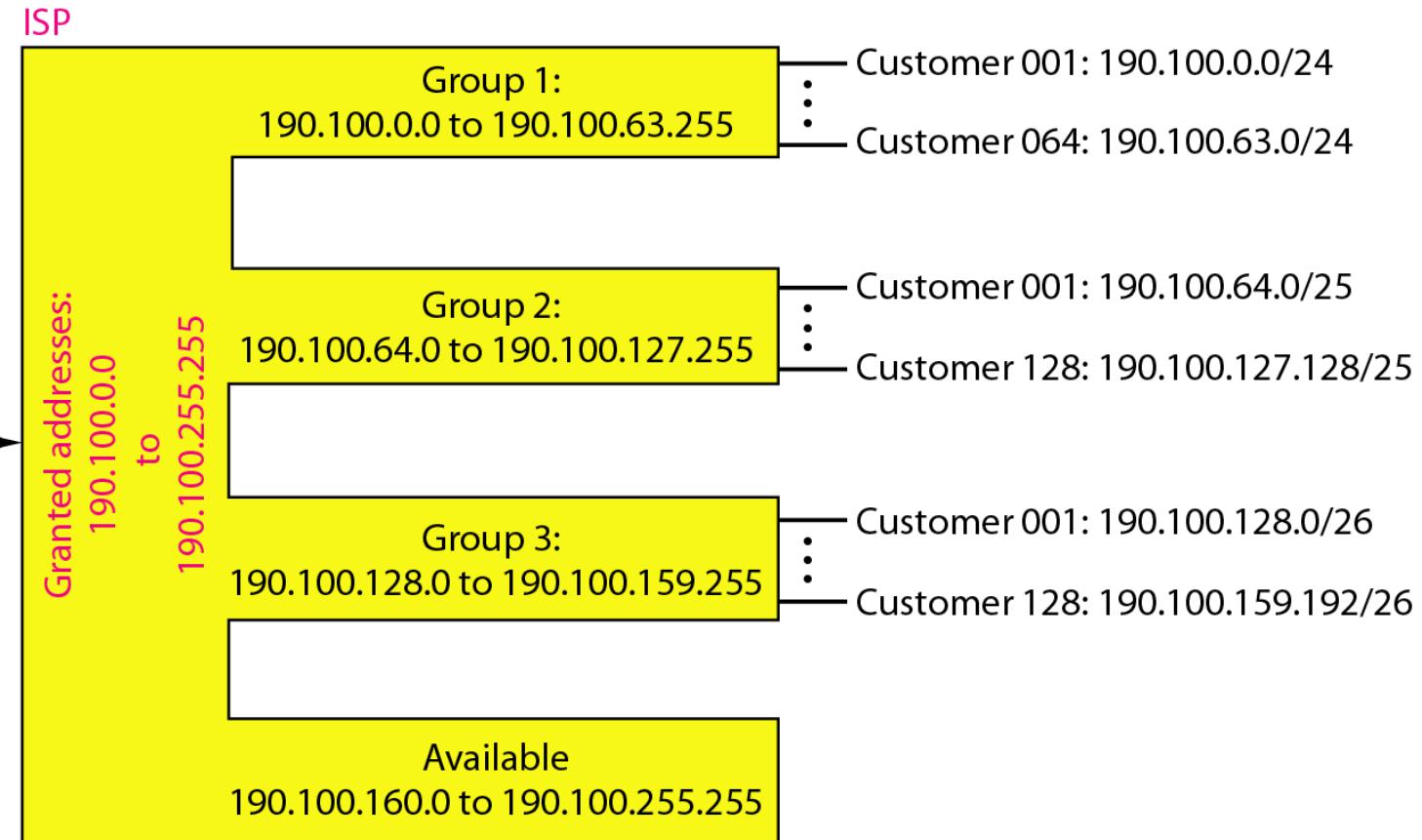
THE
UNIVERSITY
OF DUBLIN

Network Layer

Trinity College



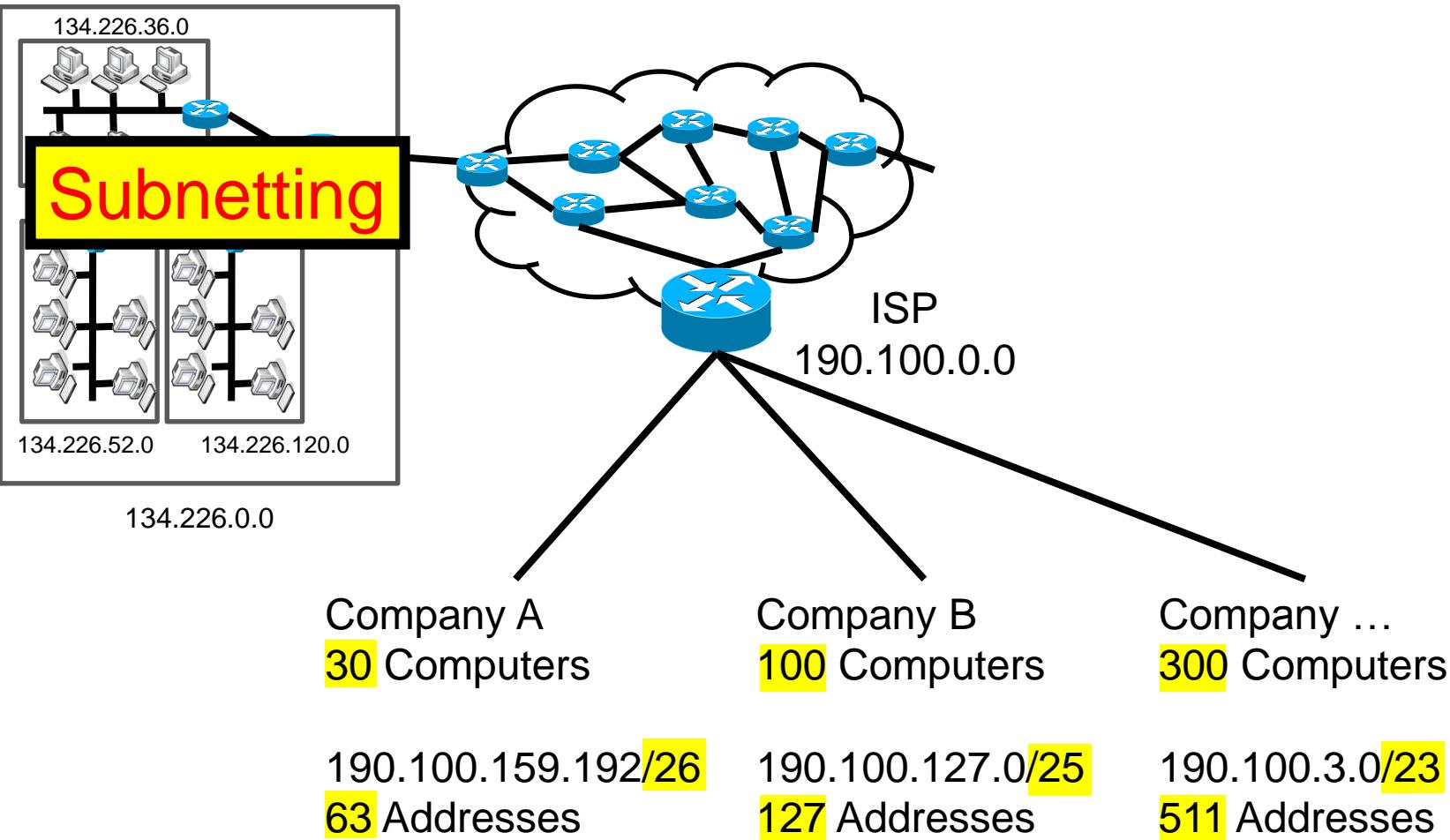
ISPs & Classless Addresses



* Figure is courtesy of B. Forouzan

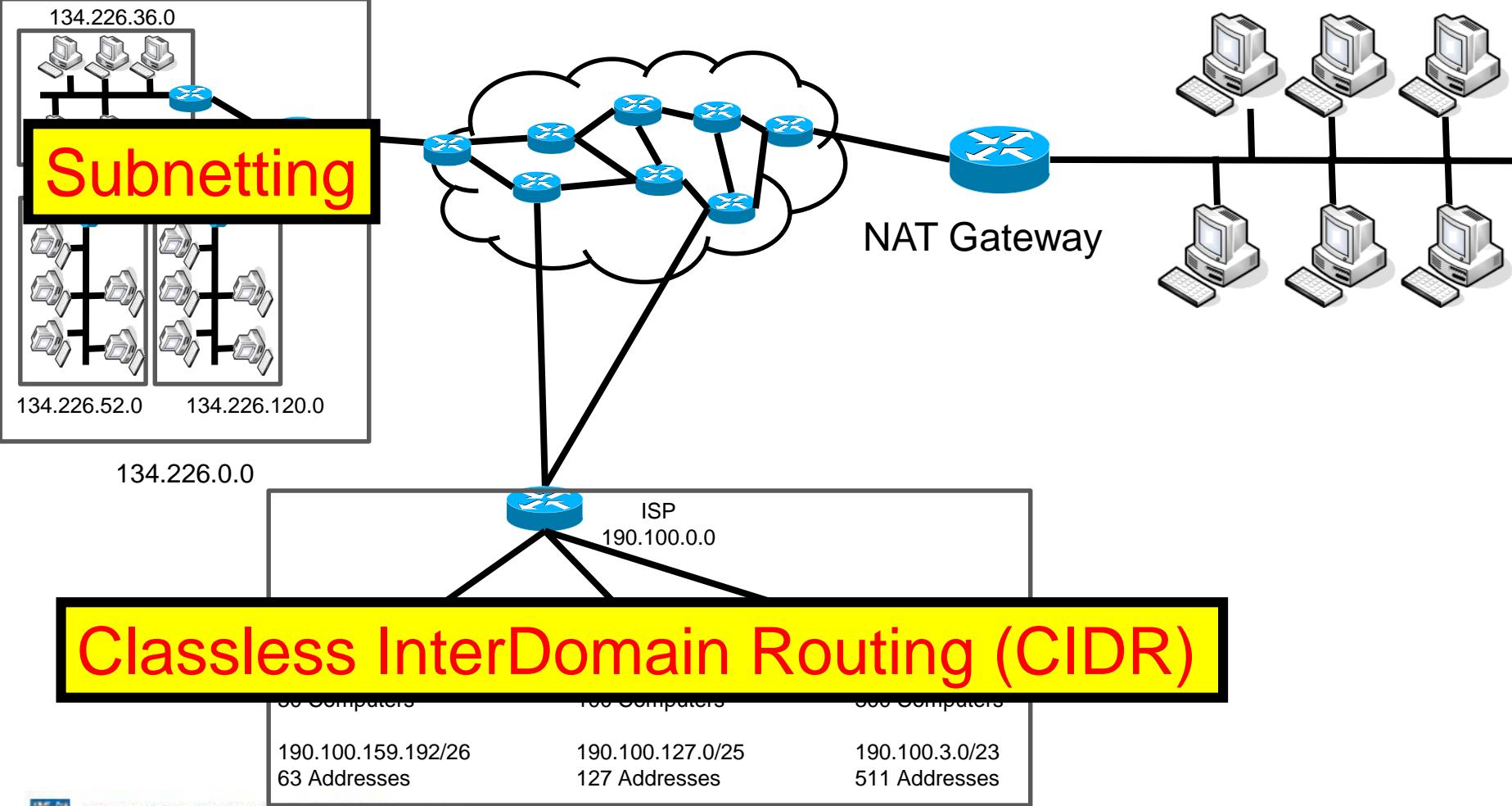
Network Layer

Trinity College



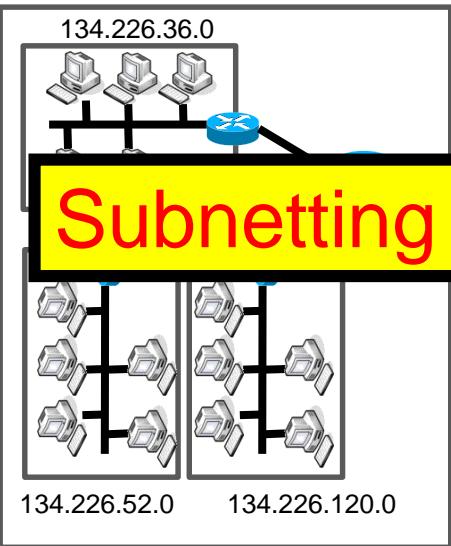
Network Layer

Trinity College



Network Layer

Trinity College

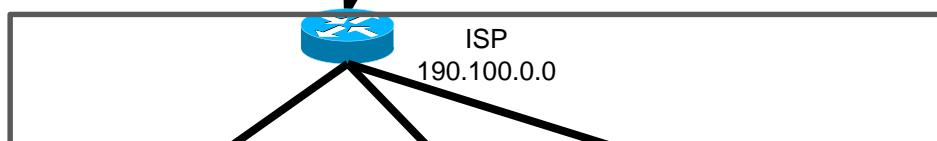


192.168.1.254
83.129.34.3

NAT Gateway

192.168.1.4

192.168.1.25



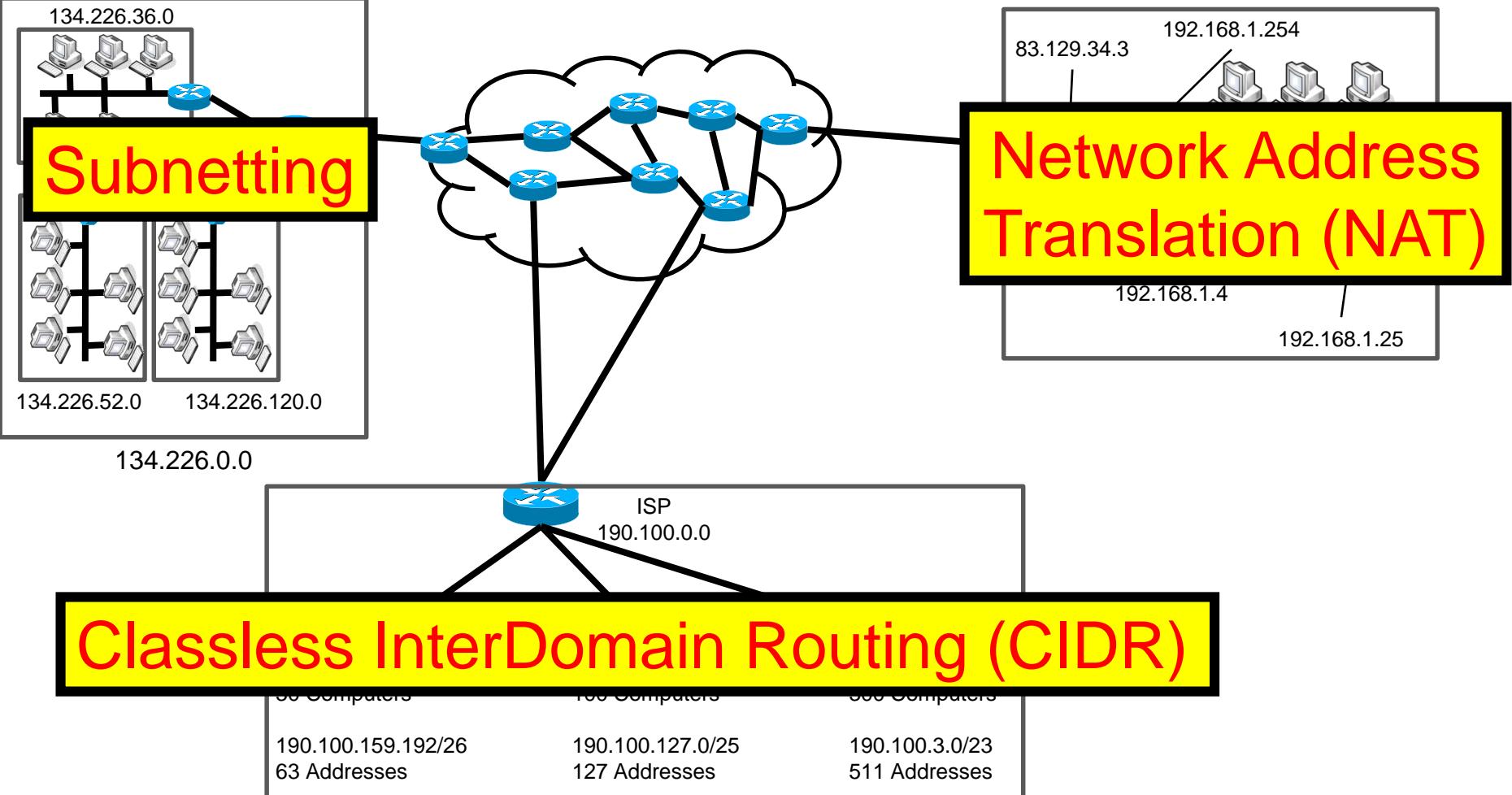
Classless InterDomain Routing (CIDR)

60 Computers
190.100.159.192/26
63 Addresses

100 Computers
190.100.127.0/25
127 Addresses

300 Computers
190.100.3.0/23
511 Addresses

Network Layer





CS2031

Telecommunications II

Routing

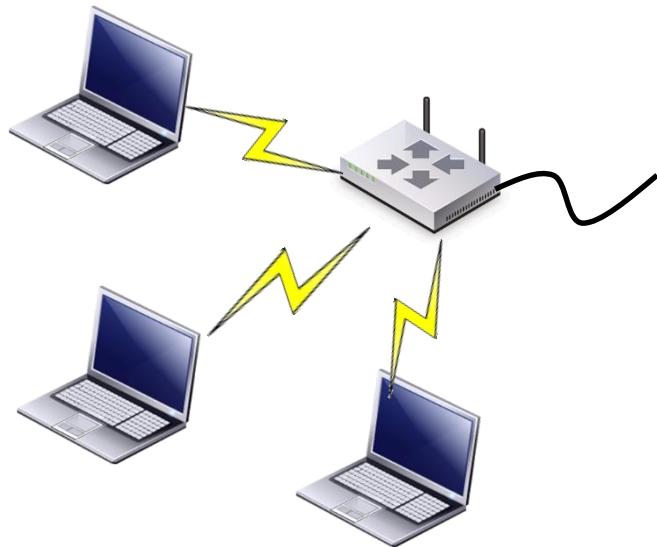


Outcome for Today

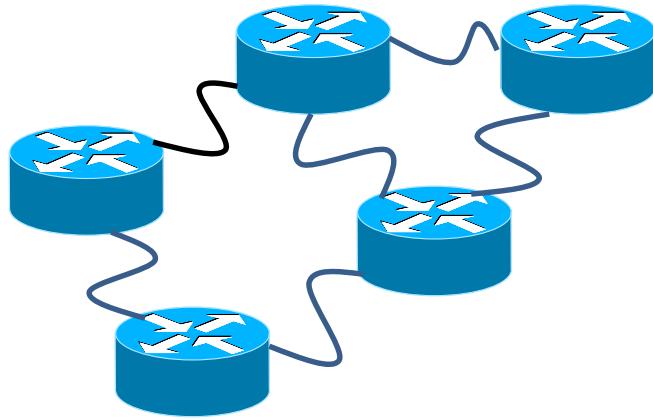
- Distance Vector Routing
- Link State Routing
- Multicast Routing
 - Dense Mode (DM)
 - Sparse Mode (SM)



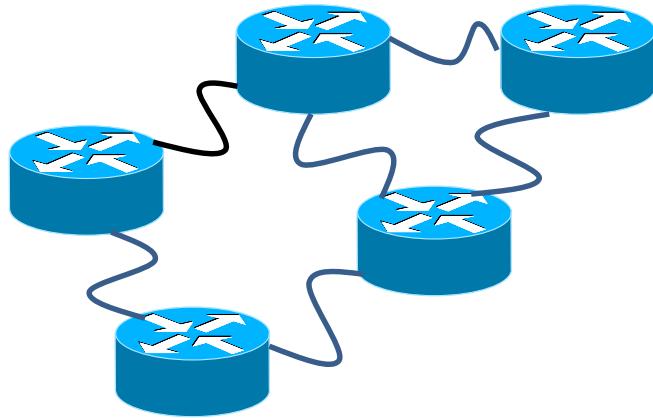
Home Network



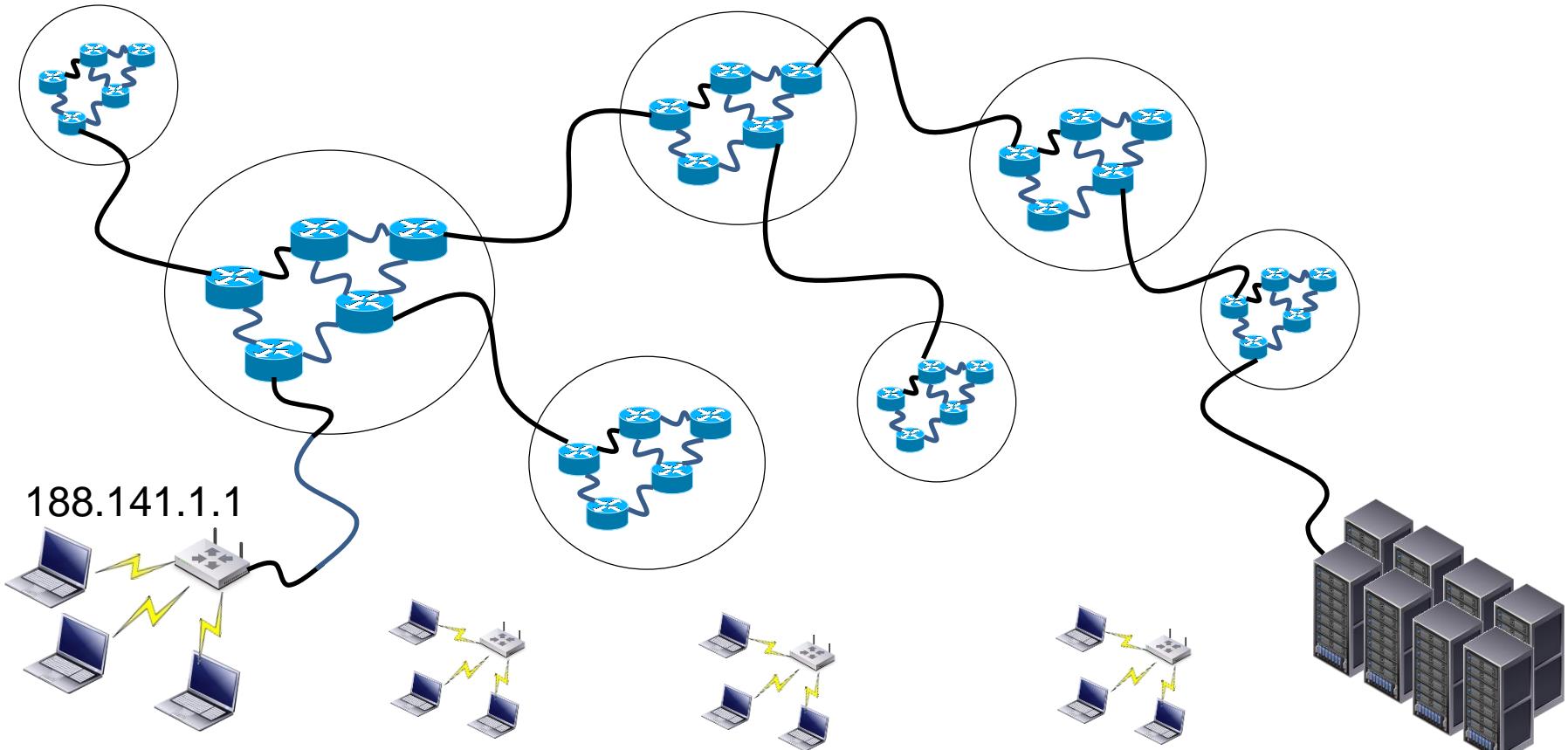
ISPs



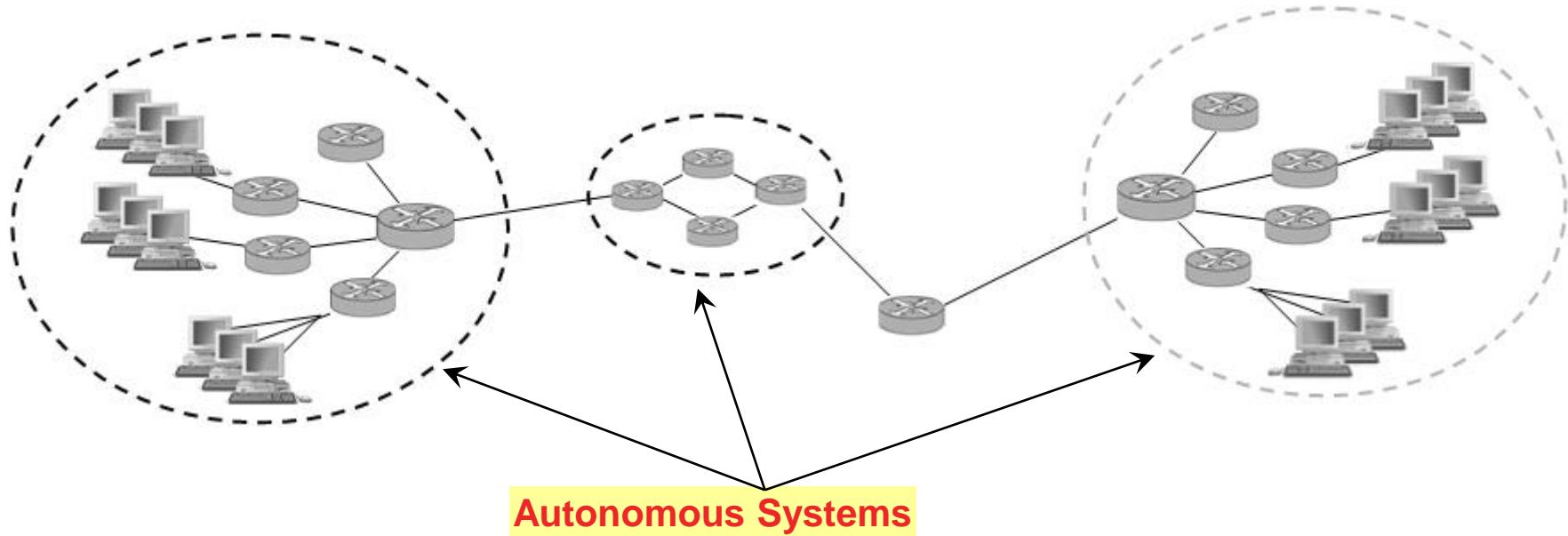
ISPs



Internet = Network of Networks

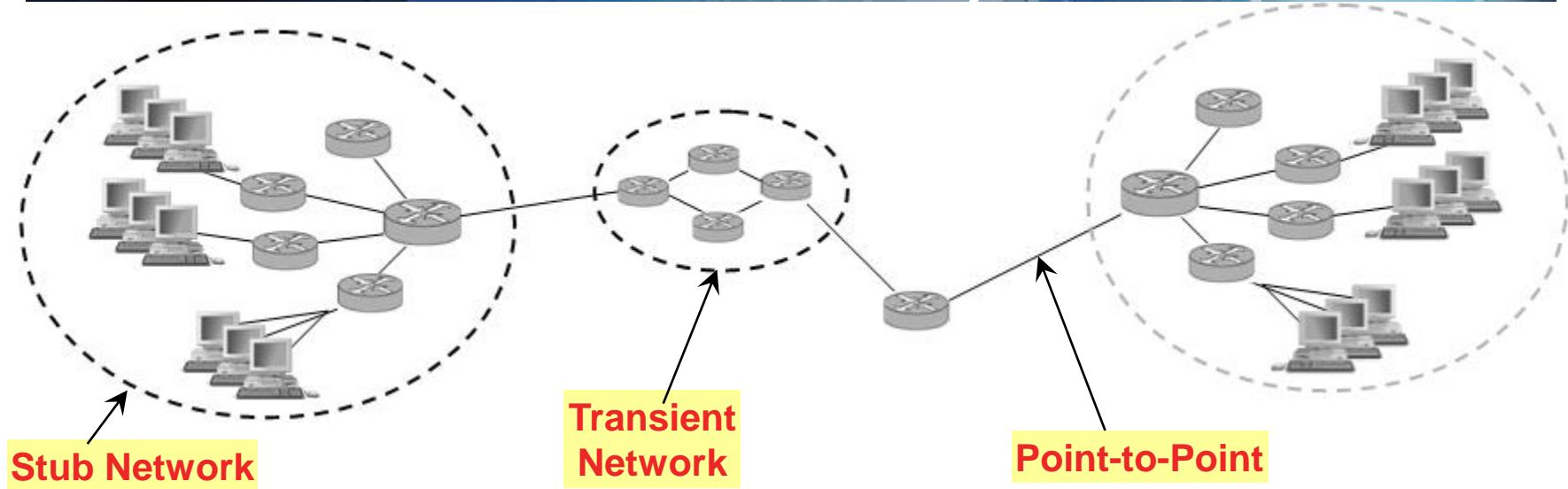


Structure of the Internet



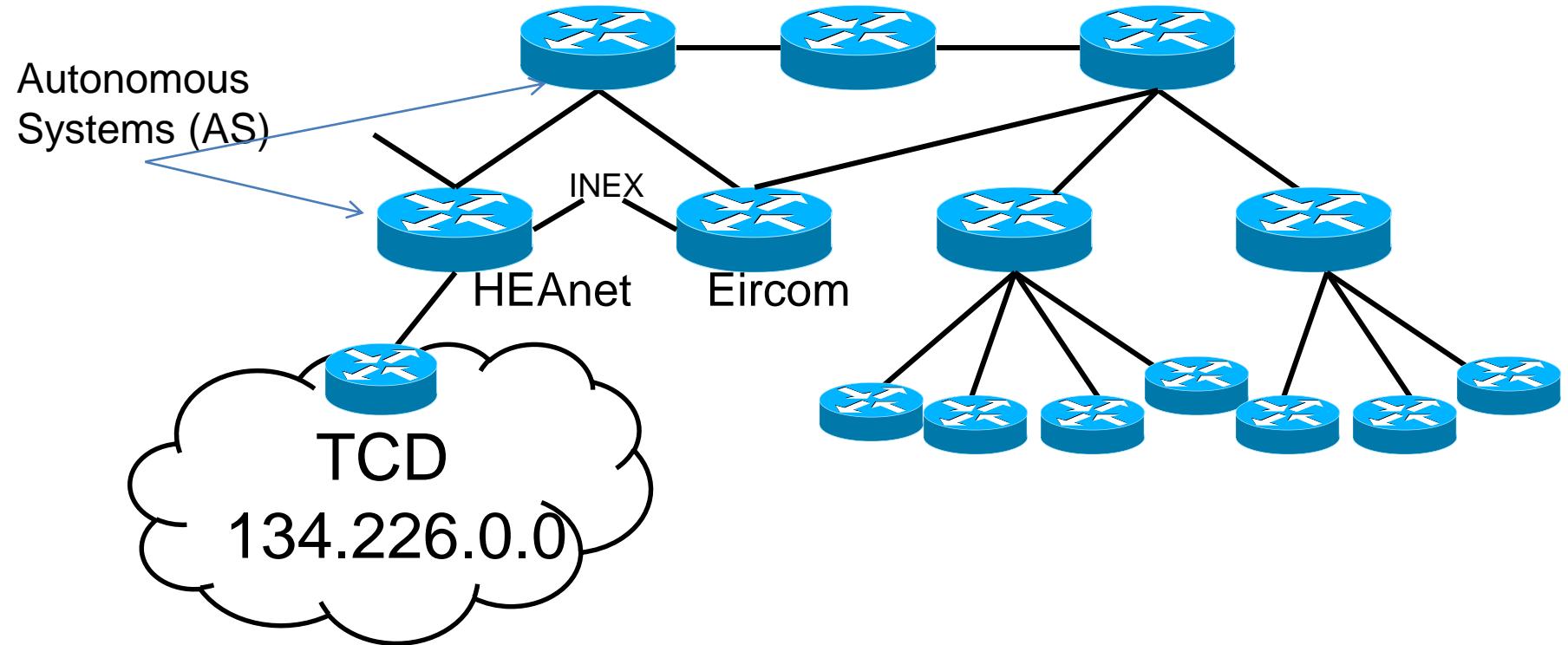
- Autonomous Systems
 - e.g. Companies, ISPs, 3rd-level Institutions

Autonomous Systems

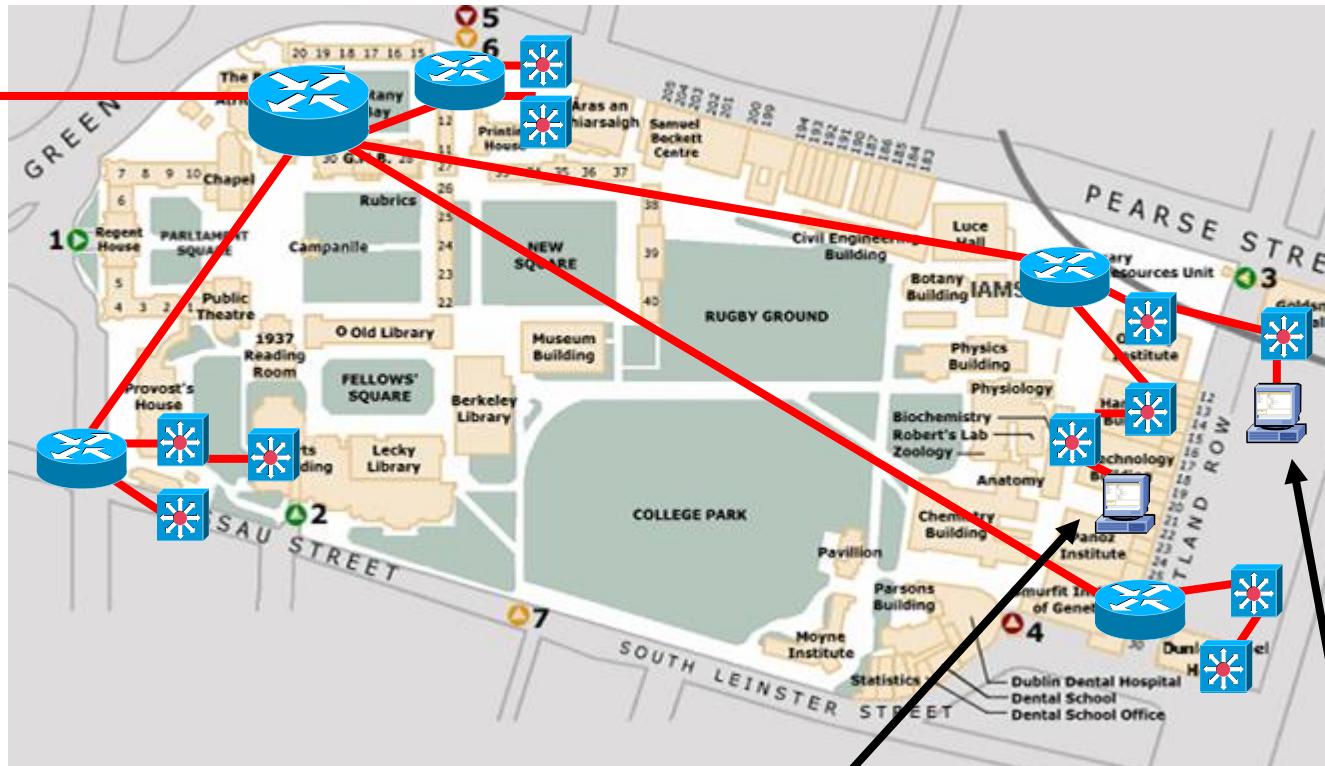


- Stub network
 - Network that does not forward to other network
- Transient network
 - Network that forwards traffic between other networks
- Point-to-point link

Trinity's Connection



Trinity Network with Routers



Switch



Router

**Address: 134.226.38.55
Subnet: 134.226.38.0**

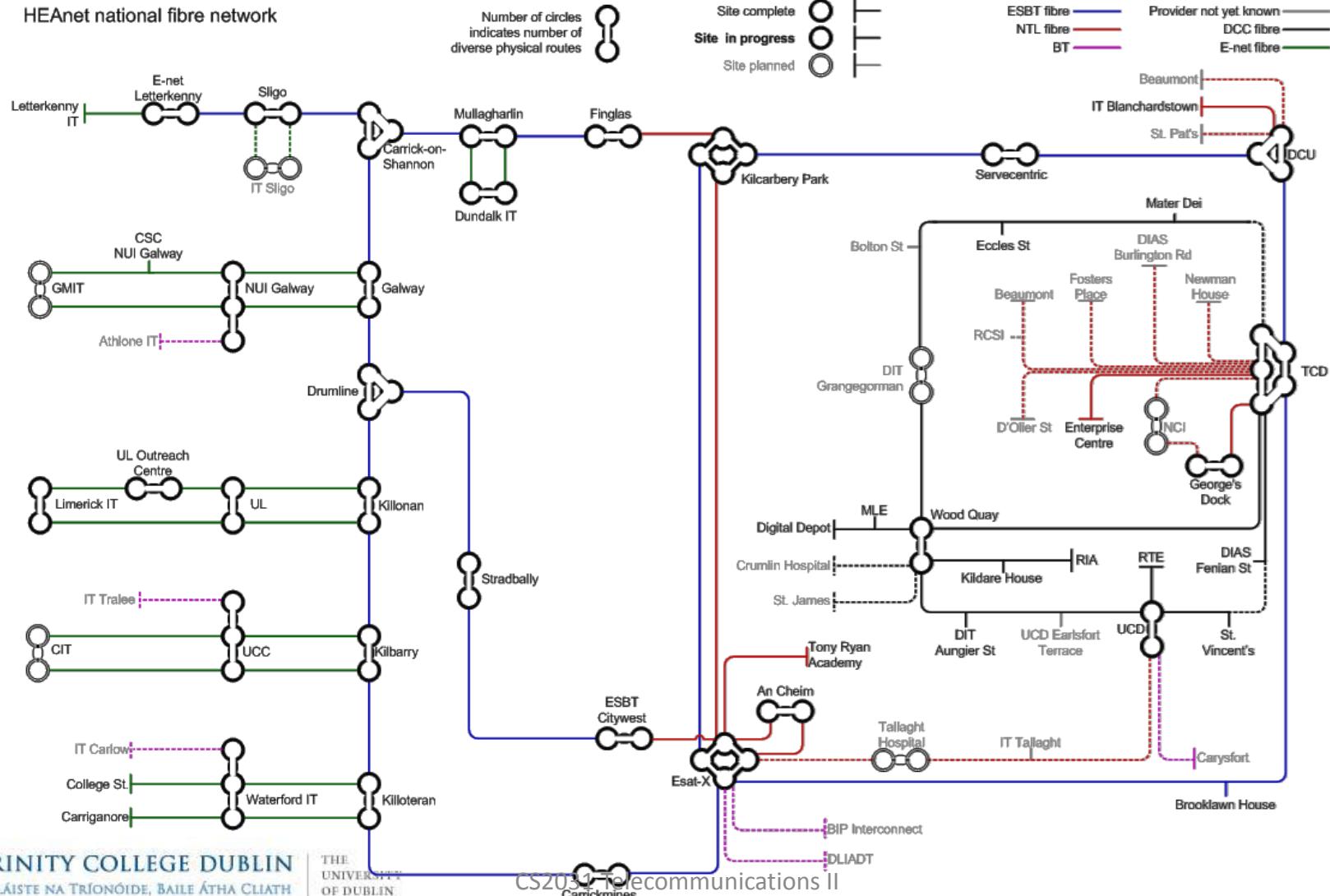
**Address: 134.226.32.55
Subnet: 134.226.32.0**



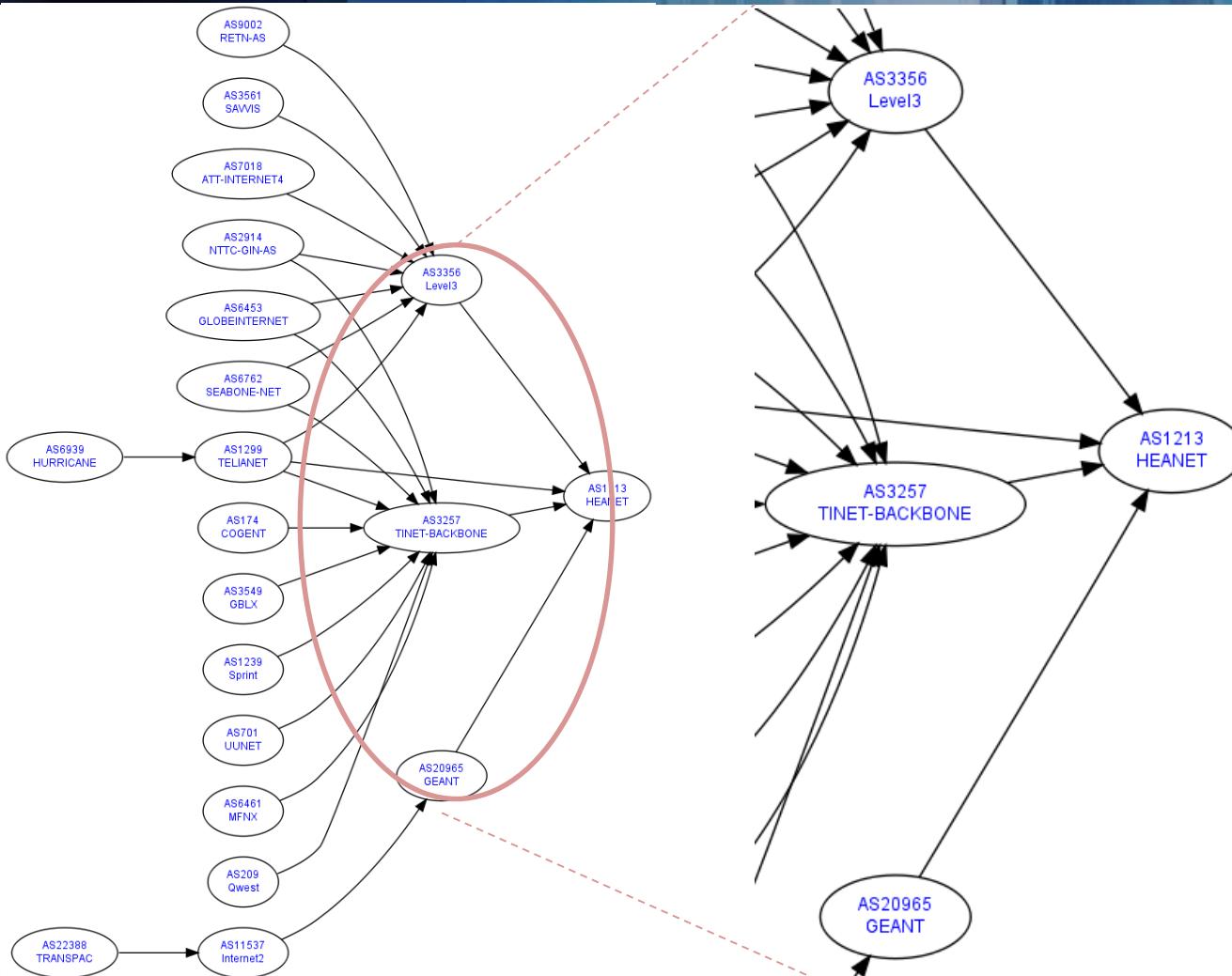
TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE, BAILE ÁTHA CLIATH

THE
UNIVERSITY
OF DUBLIN

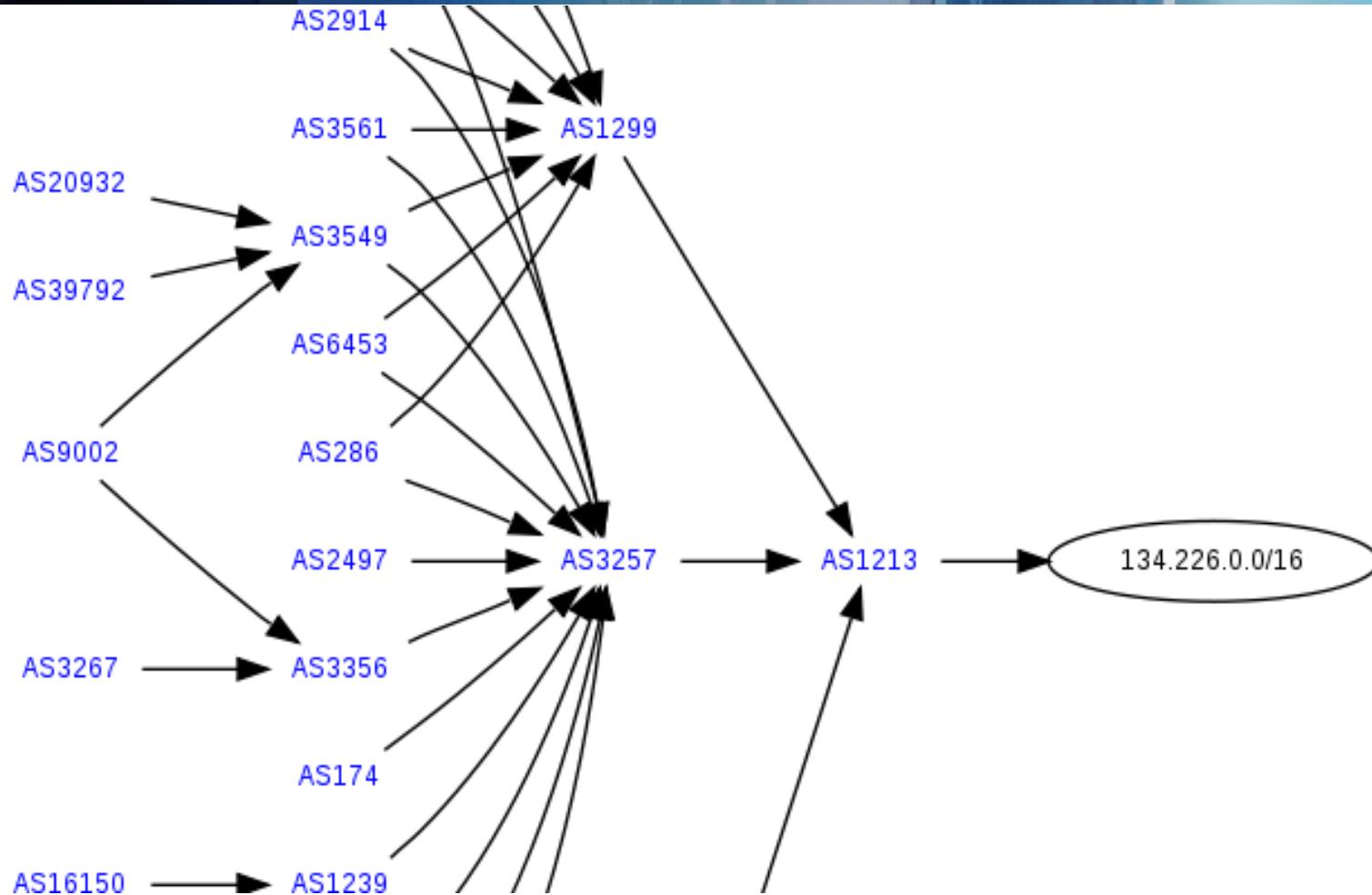
HEAnet Fibre Network



AS1213 - HEANET



AS1213 - HEANET

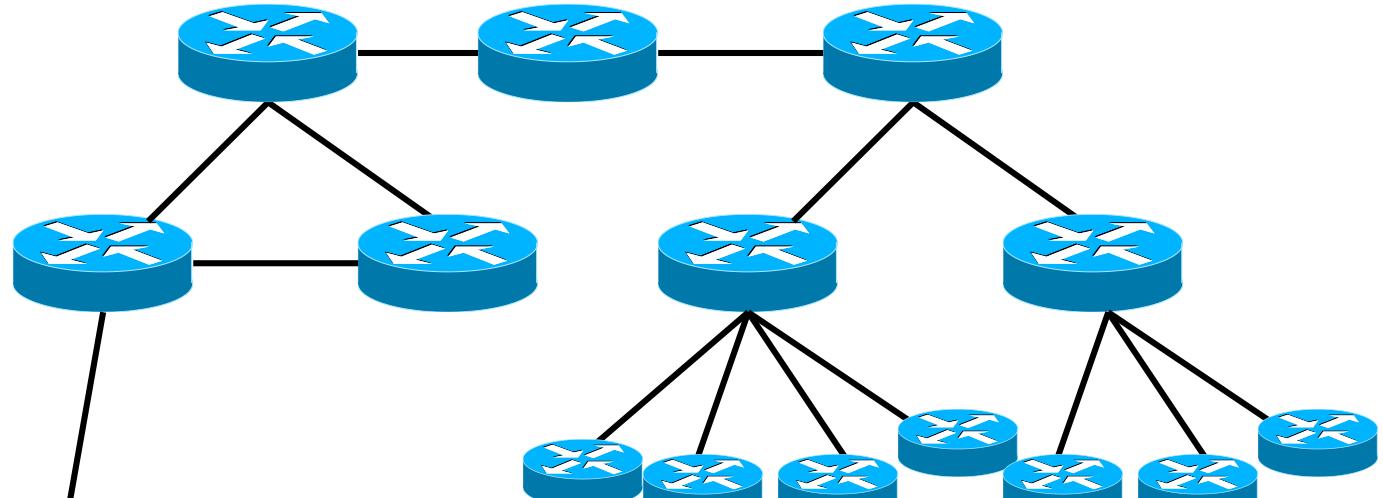


<http://www.robtex.com/route/134.226.0.0-16.html>

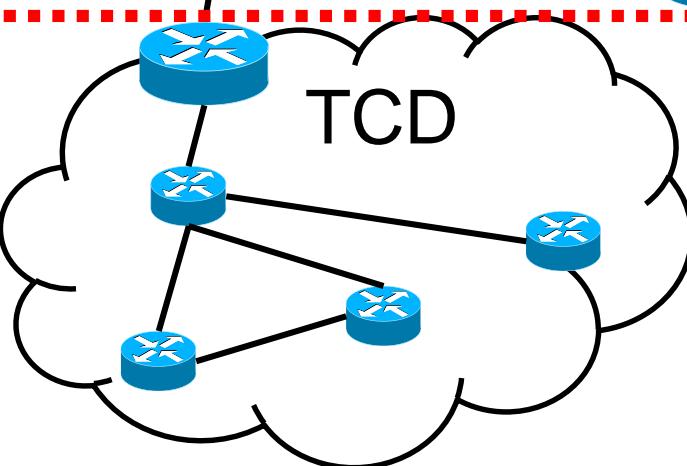


InterAS vs IntraAS Routing

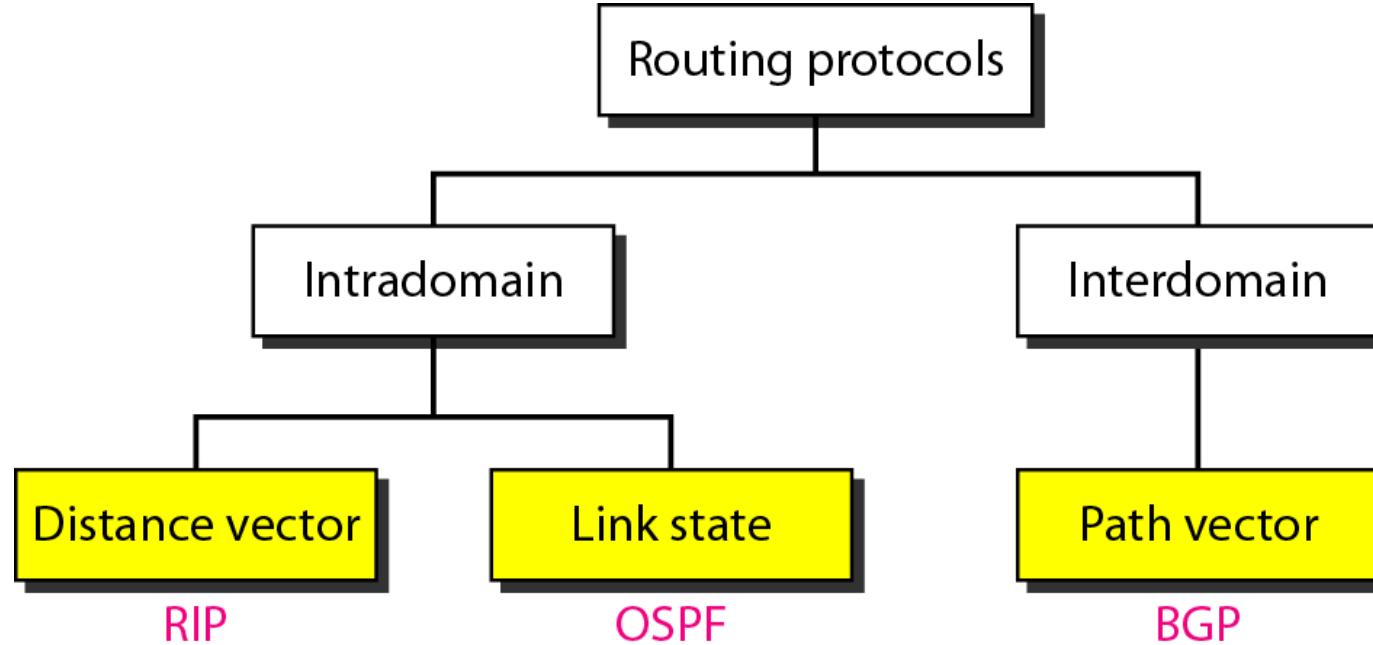
InterAS
Routing



IntraAS
Routing



Routing Protocols



- Interior Routing Protocols
 - Routing within ASs
- Exterior Routing Protocols
 - Routing between ASs

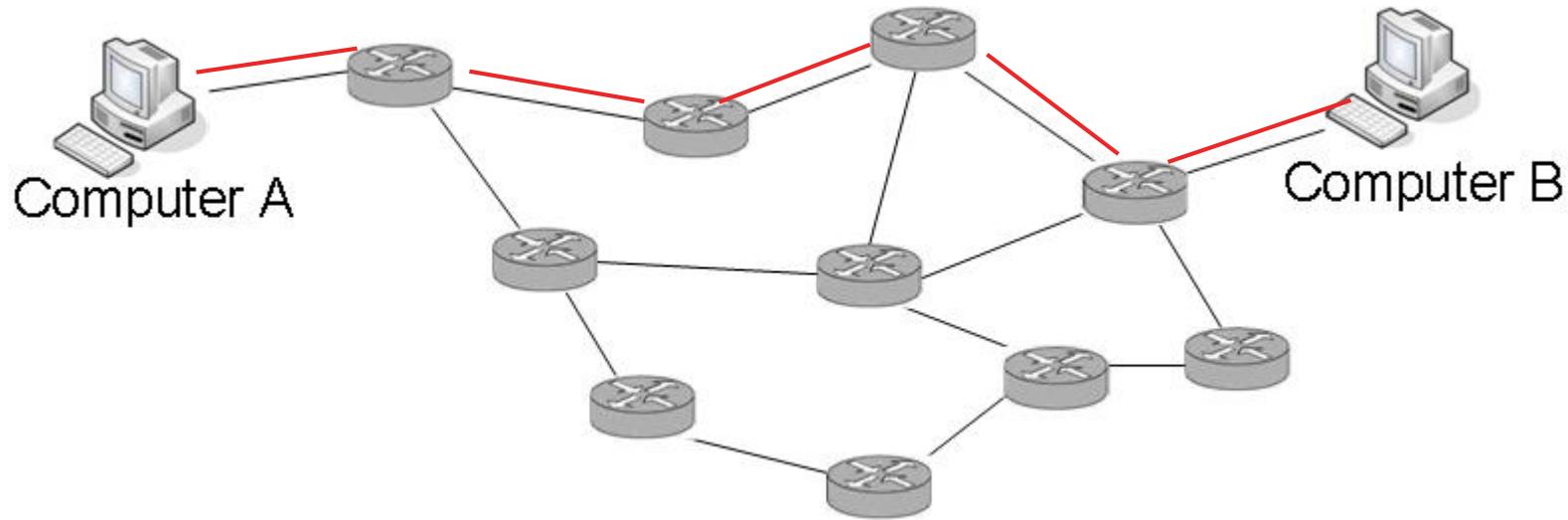
Routers

- One Main Interest
Forwarding Packets
- Important Aspects
Queue Length
Routing Table

Destination	Gateway	Interface
IP Range ₁	G ₁	IF ₁
IP Range ₂	G ₂	IF ₂

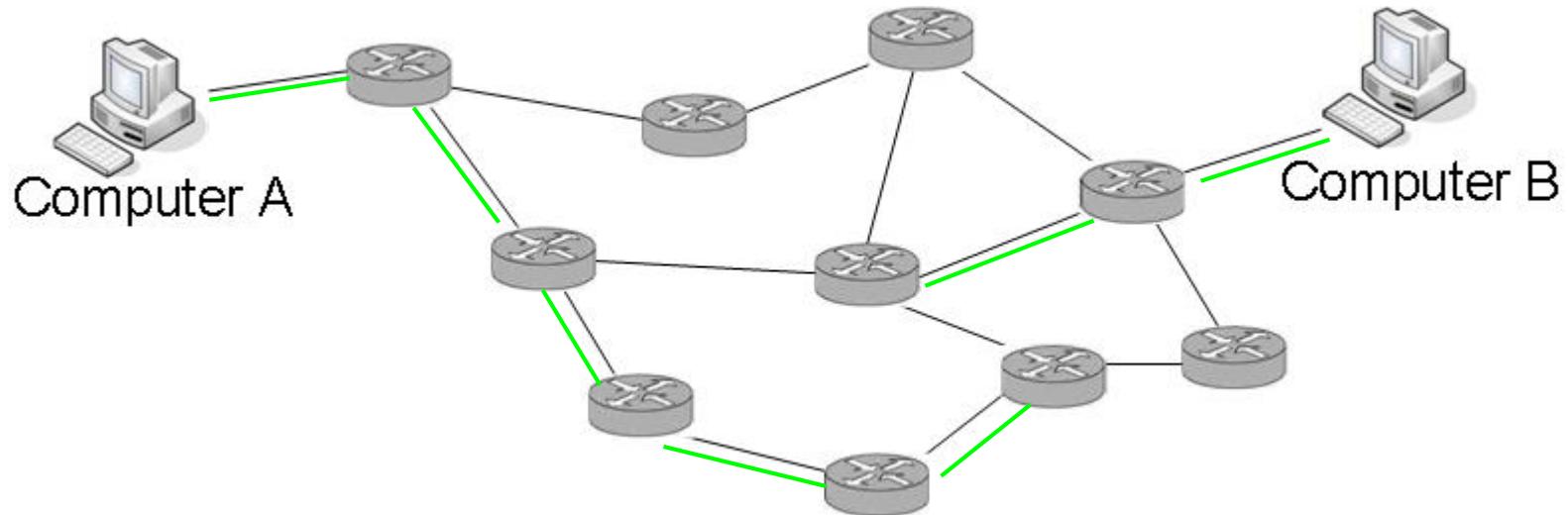


The Scenario

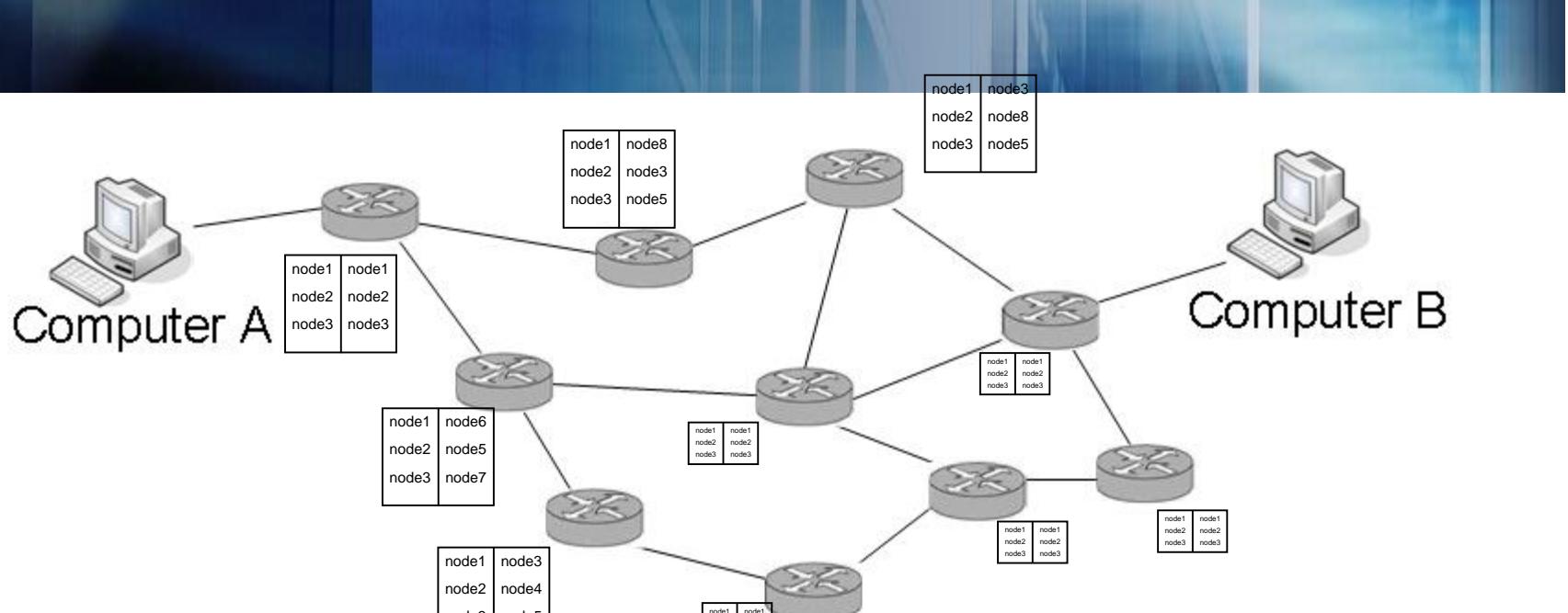


- Best route: Smallest number of hops?

The Scenario



- Best route:
 - Fastest round-trip time?
 - Highest Bandwidth?



- Routing Tables
 - Creating tables
 - Dynamic vs. Static
 - Maintaining tables
 - Periodic vs. Aperiodic

Routing Approaches

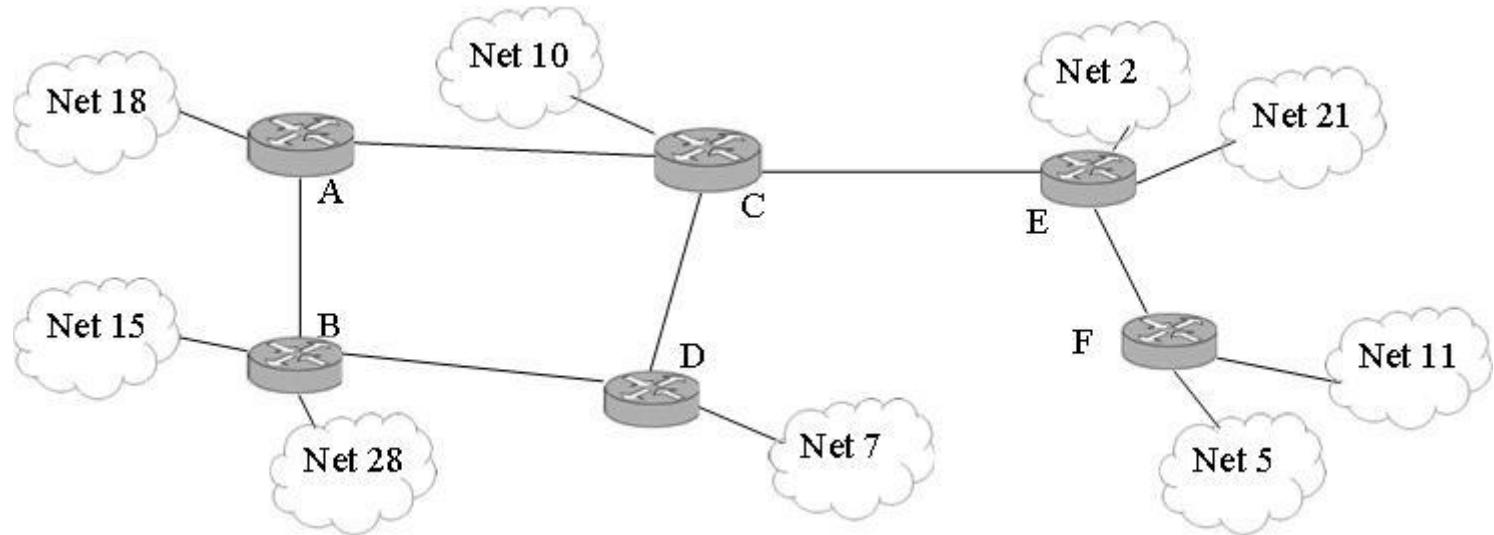
- Distance Vector routing
 - Routes propagate through exchange of routing tables
 - Based on communication with neighbours
- Link State routing
 - Establishing view of complete topology
 - Makes use of Dijkstra's Shortest-Path Algorithm

Distance Vector Routing

- Each node maintains a set of triples
 - (**Destination**, **Cost**, **NextHop**)
- Exchange updates with directly connected neighbors
 - periodically (on the order of several seconds)
 - whenever table changes (called *triggered update*)
- Each update is a list of all pairs in the routing table:
 - (**Destination**, **Cost**)

Destination	Hop Count	Next Router	Other info
163.5.0.0	7	172.6.23.4	
197.5.13.0	5	176.3.6.17	
189.45.0.0	4	200.5.1.6	
115.0.0.0	6	131.4.7.19	

Distance Vector: Example



A	Net18	1	-

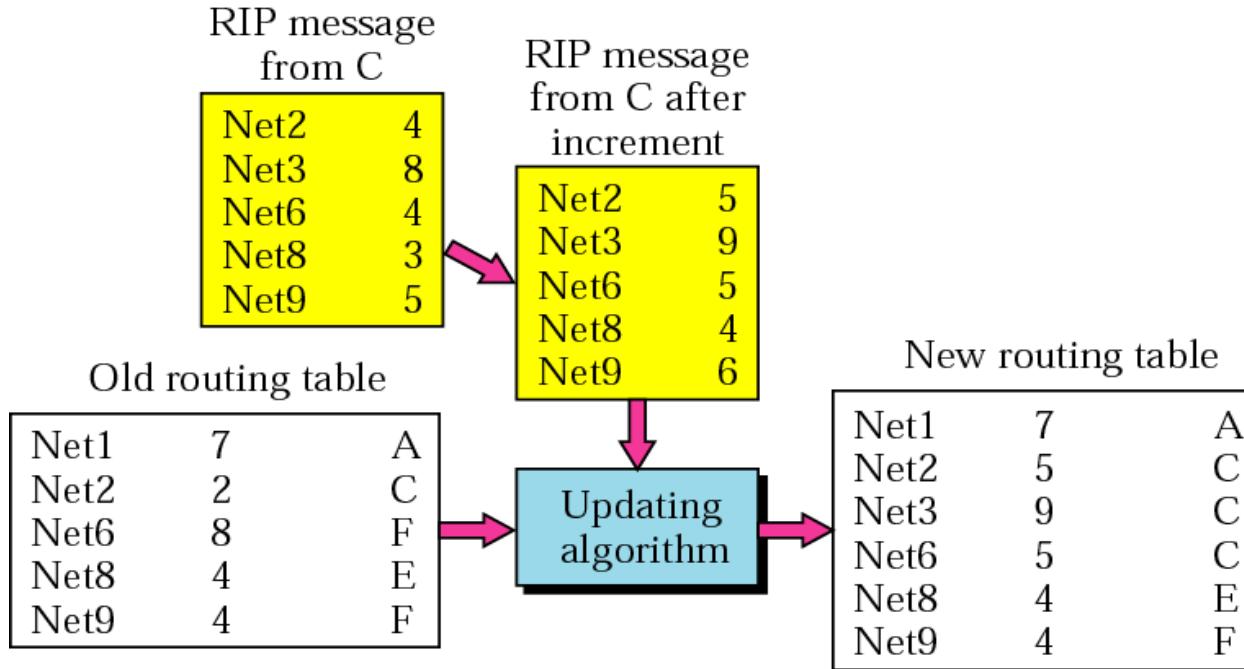
B	Net15	1	-

C	Net10	1	-

D	Net7	1	-

- Each router know the networks that are immediately connected to it

Receiving an Update



Net1: No news, do not change

Net2: Same next hop, replace

Net3: A new router, add

Net6: Different next hop, new hop count smaller, replace

Net8: Different next hop, new hop count the same, do not change

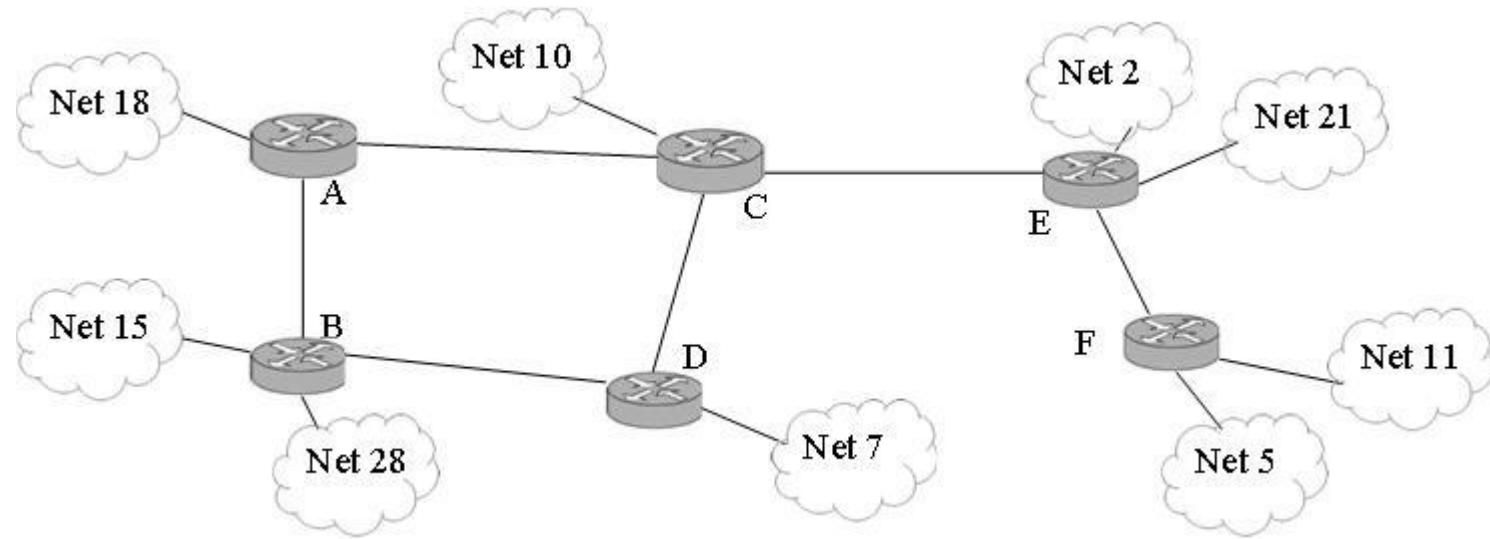
Net9: Different next hop, new hop count larger, do not change

RIP Updating Algorithm

Receive: A response RIP message

- Add one hop to the hop count for each advertised destination.
- Repeat the following steps for each advertised destination:
 - a) If (destination not in the routing table)
Add the advertised information to the table.
 - b) Else
If (next-hop field is the same)
Replace entry in the table with the advertised one.
Else
If (advertised hop count smaller than one in the table)
Replace entry in the routing table.

Distance Vector: Example II



A		
Net18	1	-
Net15	2	B
Net28	2	B
Net10	2	C

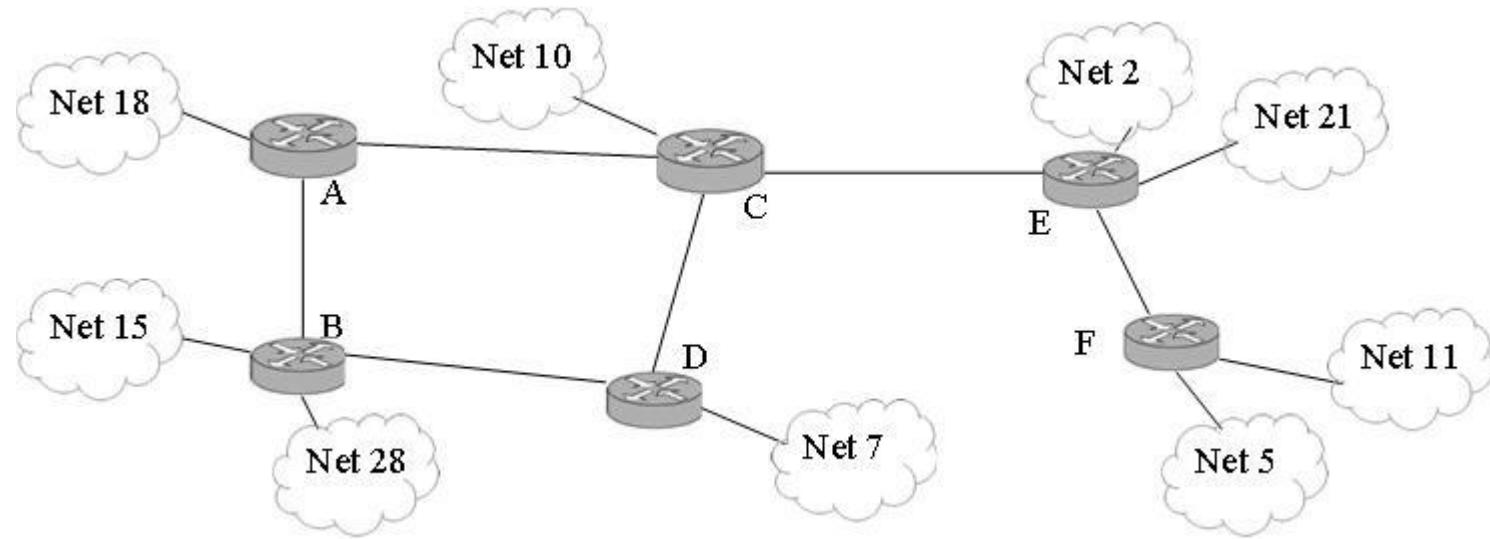
B		
Net15	1	-
Net28	1	-
Net18	2	A
Net7	2	D

C		
Net10	1	-
Net7	2	D
Net2	2	E
Net21	2	E

D		
Net7	1	-
Net15	2	B
Net28	2	B
Net10	2	C

- e.g. Node A incorporates information from Node B

Distance Vector: Example III



A		
Net18	1	-
Net15	2	B
Net28	2	B
Net10	2	C
Net7	3	B

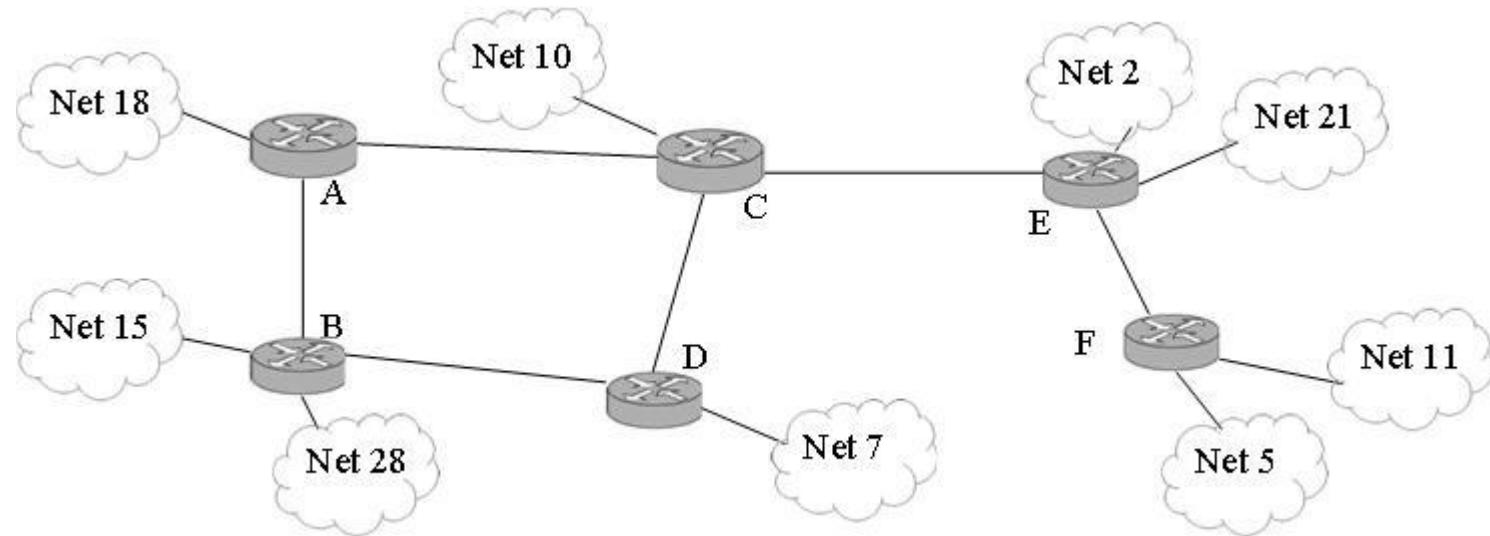
B		
Net15	1	-
Net28	1	-
Net18	2	A
Net7	2	D
Net10	3	D

C		
Net10	1	-
Net7	2	D
Net2	2	E
Net21	2	E
Net11	3	E
Net5	3	E

D		
Net7	1	-
Net15	2	B
Net28	2	B
Net10	2	C
Net2	3	C
Net21	3	C

- e.g. Node A receives information Node B received during the last round

Distance Vector: Example IV



A		
Net18	1	-
Net15	2	B
Net28	2	B
Net10	2	C
Net7	3	B
...
Net11	5	C
Net5	5	C

- Time require to build complete routing tables for all nodes
 - this is known as convergence

Count-to-Infinity Problem



Initially

1	•	•	•	•
1	2	•	•	•
1	2	3	•	•
1	2	3	4	

(a)



Initially

1	2	3	4	
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges
7	8	7	8	After 6 exchanges
⋮	⋮	⋮	⋮	⋮

(b)

- Link between A and B breaks
- B receives update from C advertising route to A

- Routers exchange updates
- After 4 steps the network converges
- Every router knows how to get to router A

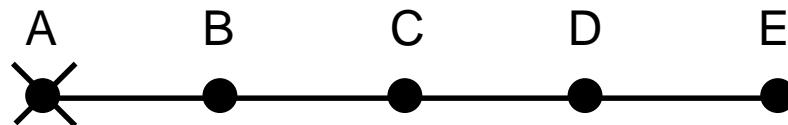


Solutions for Count-to-Infinity

- Possible solutions:
 1. Impose upper bound on maximum distance
 2. *Split horizon*
 - C should not send to node B its new distance to node A, if node B is C's next-hop towards A
 3. *Split horizon with poisoned reverse*
 - C should tell node B that its distance to node A is ∞ , when node B is C's next-hop towards A
- Unfortunately, none of these solutions can deal with arbitrary topology cycles

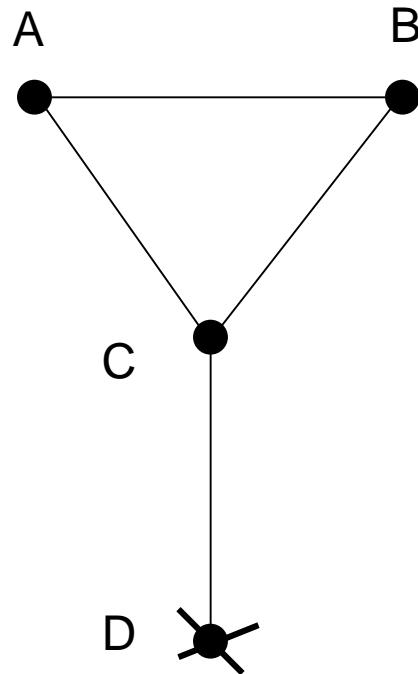
Split Horizon with Poisoned Reverse

Report “split-horizon” routes as **infinity** to break loops on the first routing exchange.



inf.	2	3	4	B learns A is dead
inf.	<u>inf.</u> → 2	3	4	B reports to C that A's metric is inf.
inf.	inf.	3	4	After 1 exchange
inf.	inf.	inf.	4	After 2 exchanges
inf.	inf.	inf.	inf.	After 3 exchanges

Split Horizon Failure

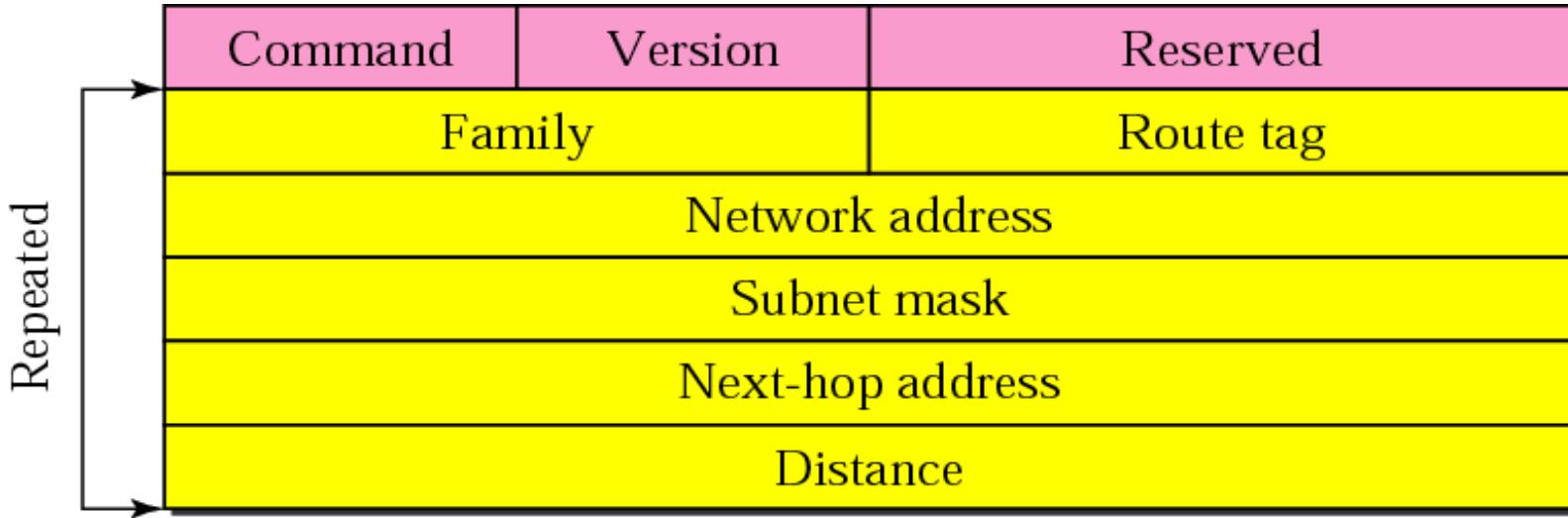


If D goes down, A and B will still count to infinity.

Split-Horizon infinity messages are sent from A->C and B->C, not A<->B

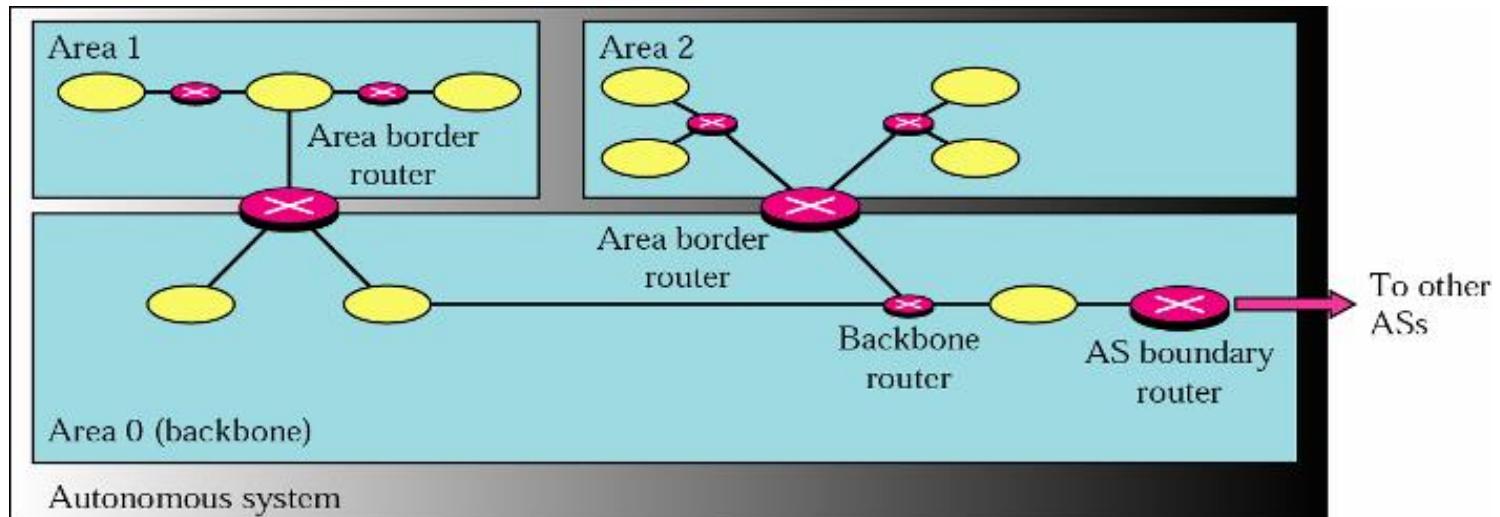


RIPv2 Message Format



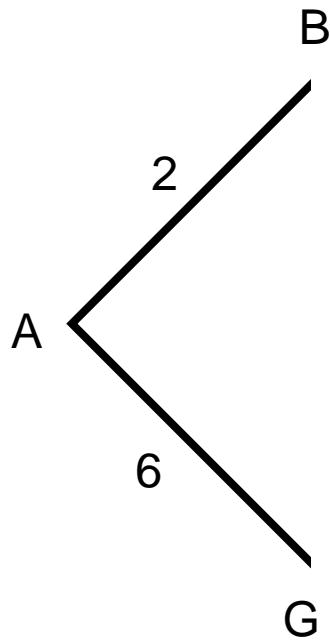
Link State Routing

- Sharing knowledge about neighbourhood
- Sharing with every other router
- Uses Dijkstra's Shortest-Path Algorithm to calculate the routing table
- Open Shortest Path First (OSPF)
 - Divides autonomous system into areas
 - Border routers summarize information for an area



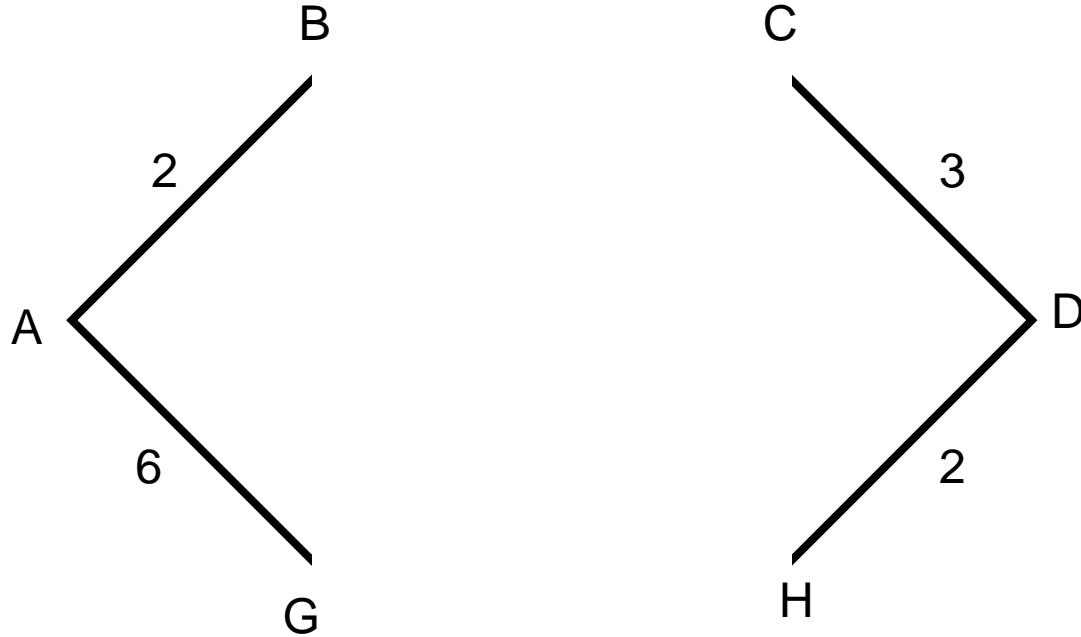
Example: Building a network graph

A discovers 2 neighbours B and G in distance 2 and 6 respectively



Example: Building a network graph

A receives an update from D

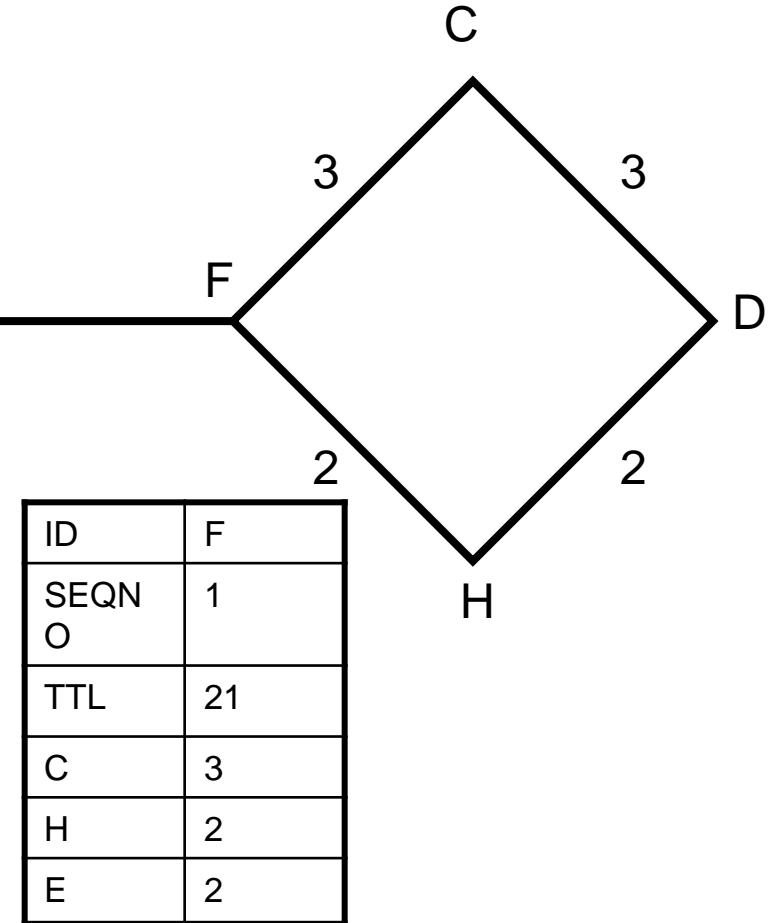
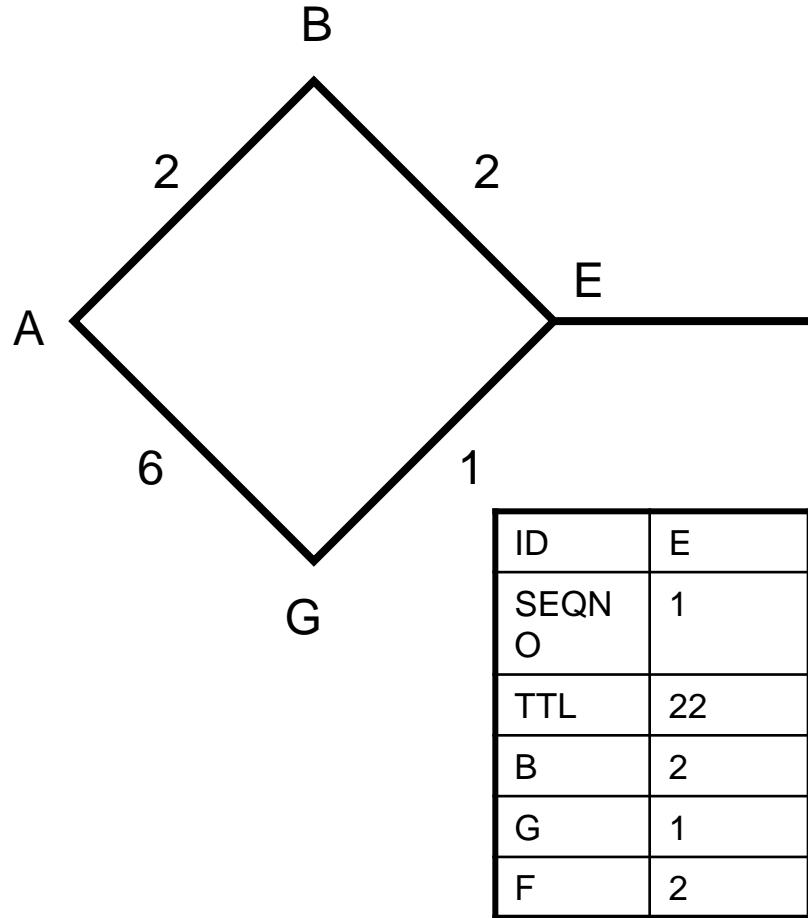


ID	D
SEQN	0
TTL	21
C	3
H	2



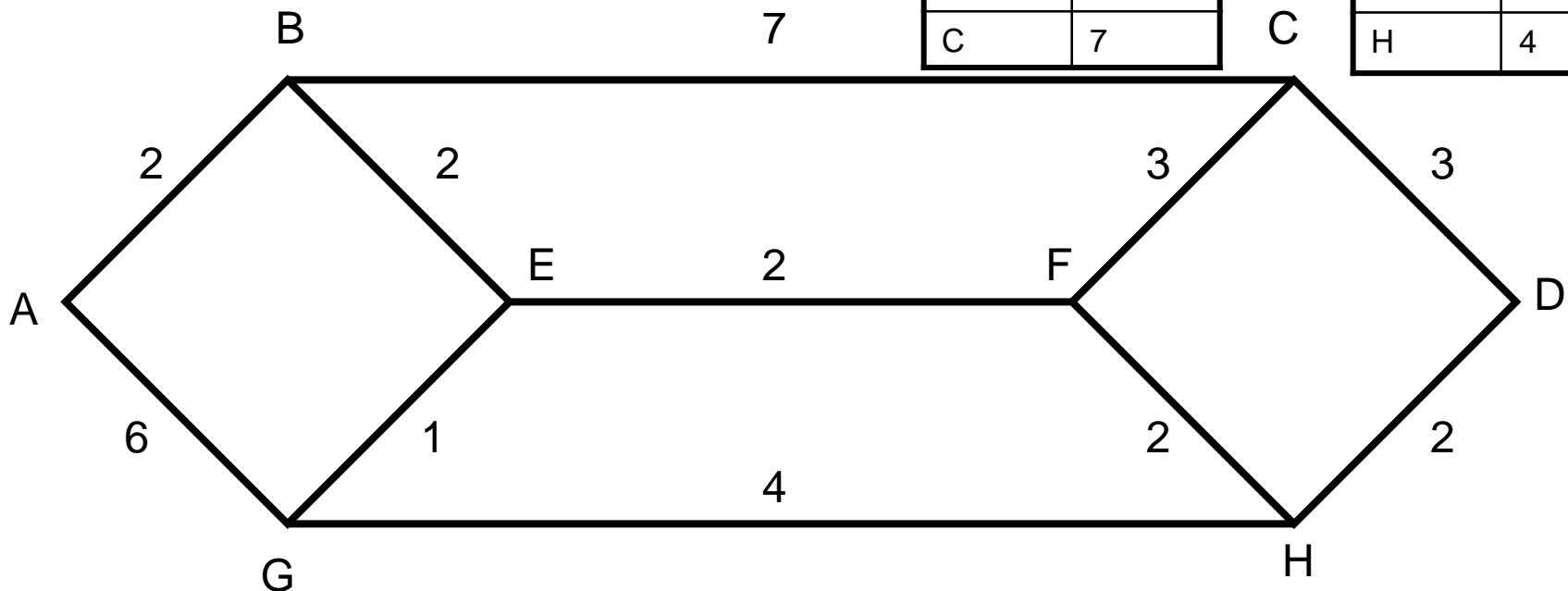
Example: Building a network graph

A receives updates from E and F



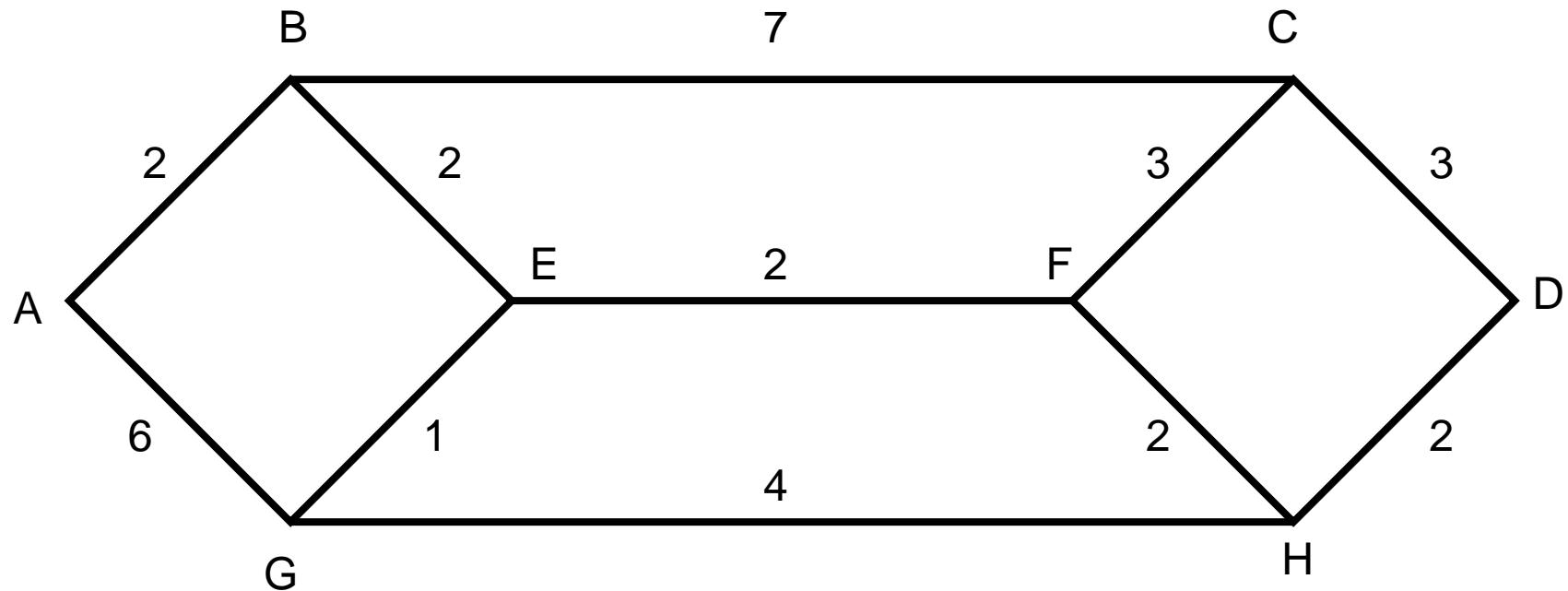
Example: Building a network graph

A receives updates from B and G



Example: Building a network graph

Final graph



Dijkstra's Algorithm

Let

- N denotes set of nodes in the graph
- $l(i, j)$ denotes non-negative cost (weight) for edge (i, j)
- s denotes initial node (source)
- M denotes the set of nodes incorporated so far
- $C(n)$ denotes cost of the path from s to node n

```
M = {s}  
for each n in N - {s}  
    C(n) = l(s, n)  
while (N != M)  
    M = M ∪ {w} such that C(w) is the minimum for  
        all w in (N - M)  
    for each n in (N - M)  
        C(n) = MIN(C(n), C(w) + l(w, n))
```

Dijkstra's Algorithm

1. Start with the local node (router): the root of the tree.
2. Assign a cost of 0 to this node and make it the first permanent node.
3. Examine each neighbour node of the node that was the last permanent node.
4. Assign a cumulative cost to each node and make it tentative.
5. Among the list of tentative nodes
 - a) Find the node with the smallest cumulative cost and make it permanent.
 - b) If a node can be reached from more than one direction
 - I. Select the direction with the shortest cumulative cost.
6. Repeat steps 3 to 5 until every node becomes permanent.

Example: Dijkstra's Algorithm

Routing Table

Tentative Nodes

A	0	-



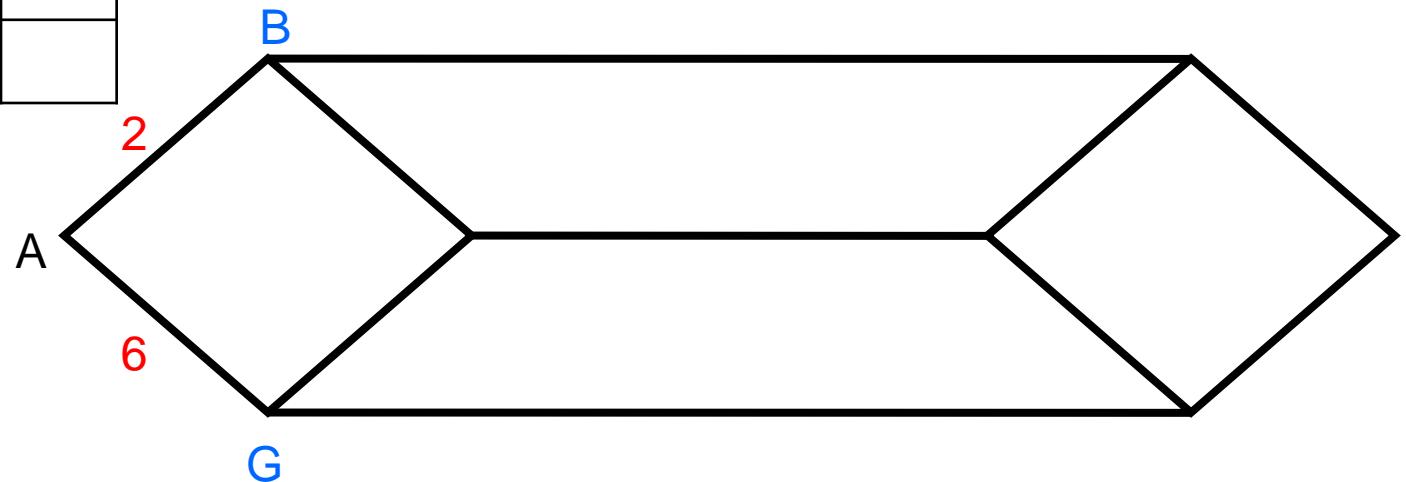
Example: Dijkstra's Algorithm

Routing Table

A	0	-

Tentative Nodes

B	2	A
G	6	A



Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A

Tentative Nodes

G	6	A
---	---	---



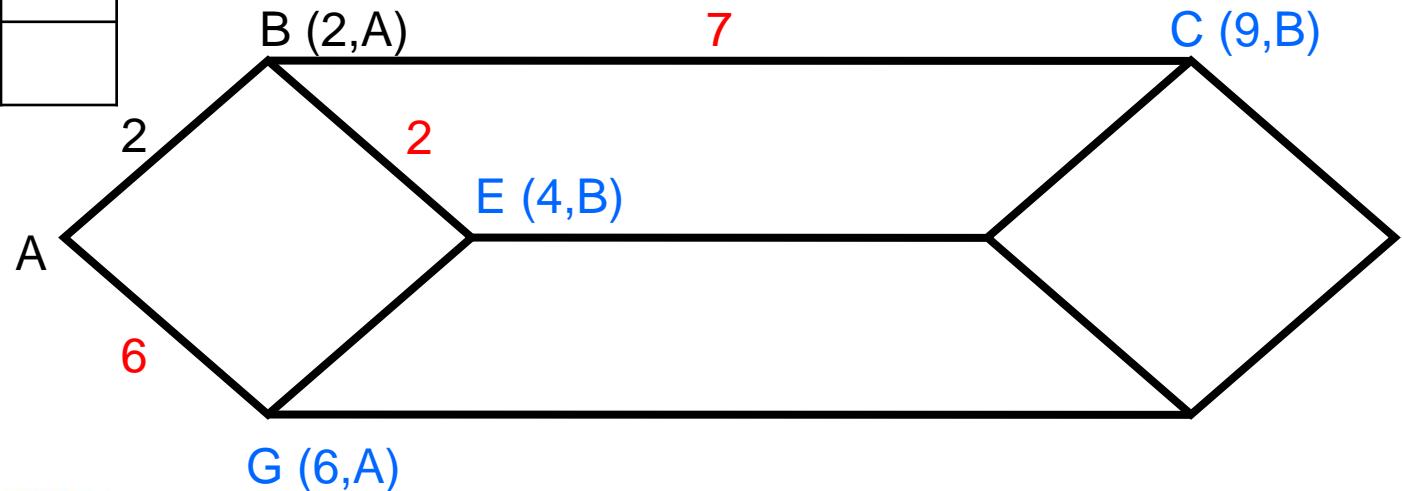
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A

Tentative Nodes

G	6	A
E	4	B
C	9	B



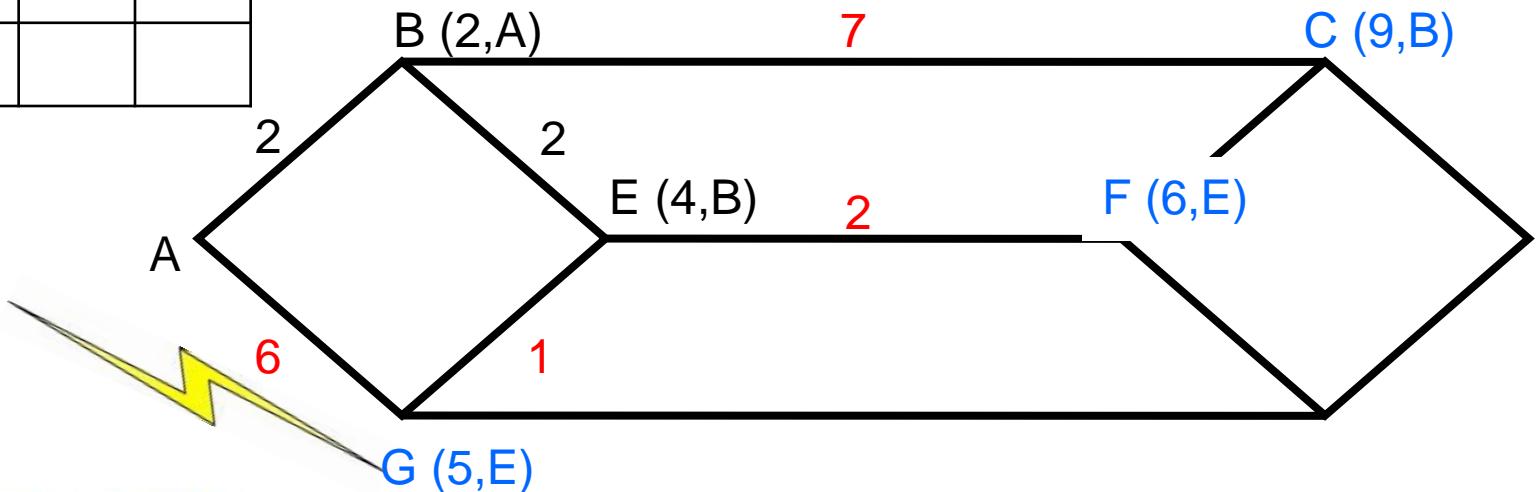
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A
E	4	B

Tentative Nodes

G	0	A
C	9	B
F	6	E
G	5	E



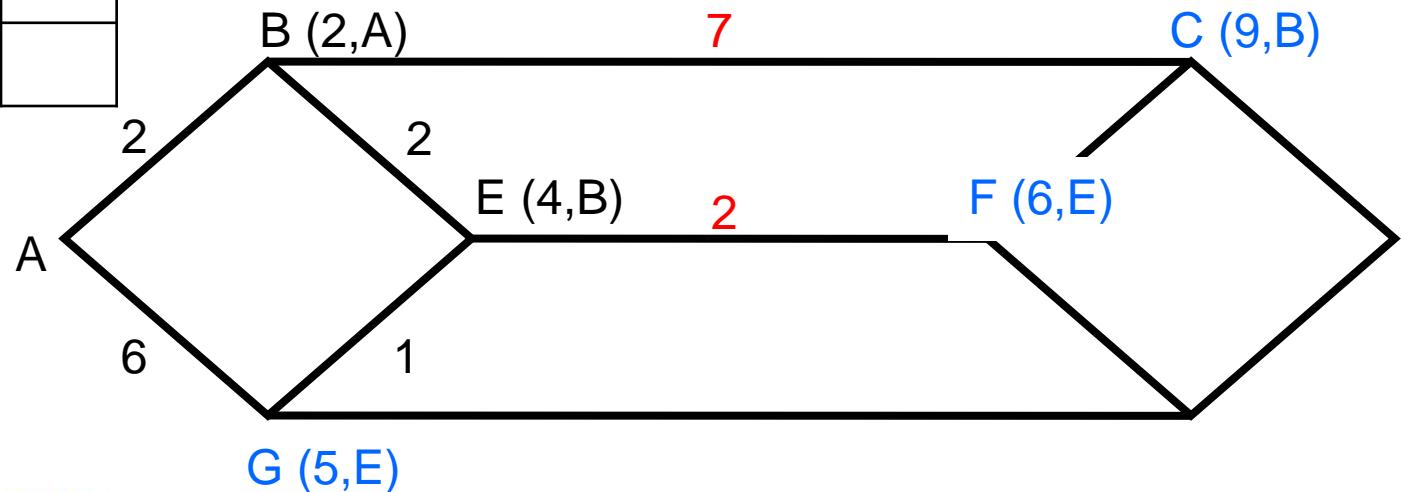
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A
E	4	B
G	5	E

Tentative Nodes

C	9	B
F	6	E



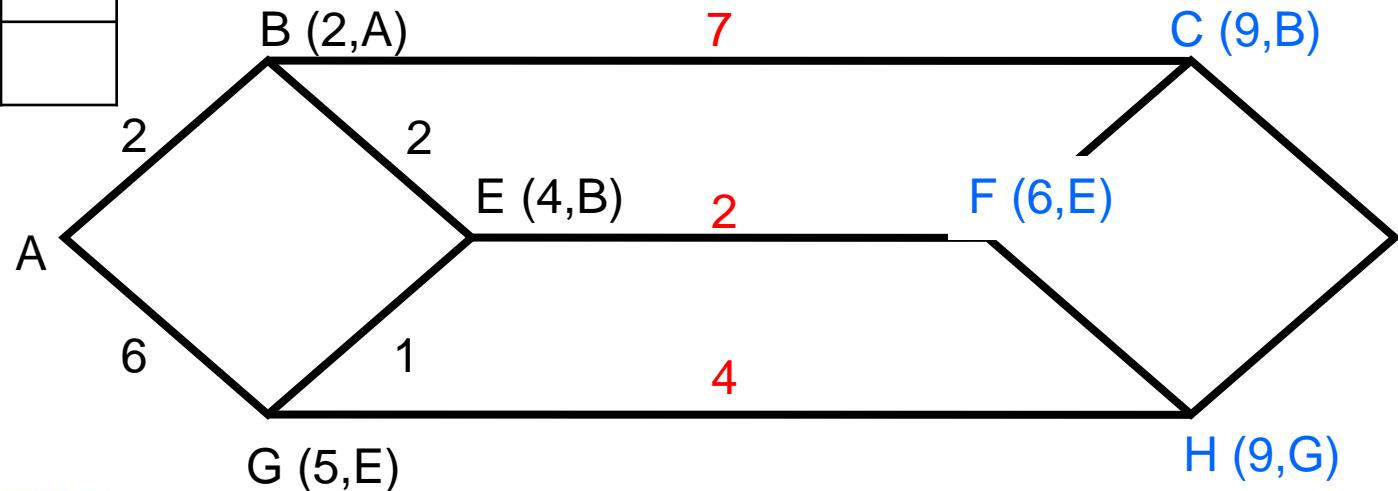
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A
E	4	B
G	5	E

Tentative Nodes

C	9	B
H	9	G
F	6	E



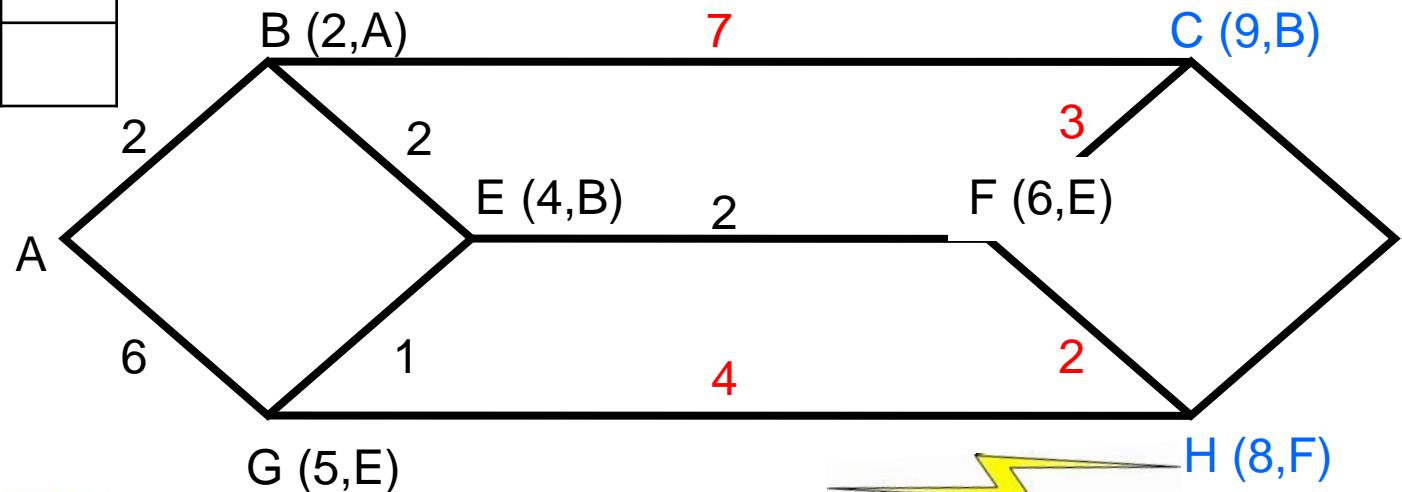
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A
E	4	B
G	5	E
F	6	E

Tentative Nodes

C	9	B
H	0	G
H	8	F



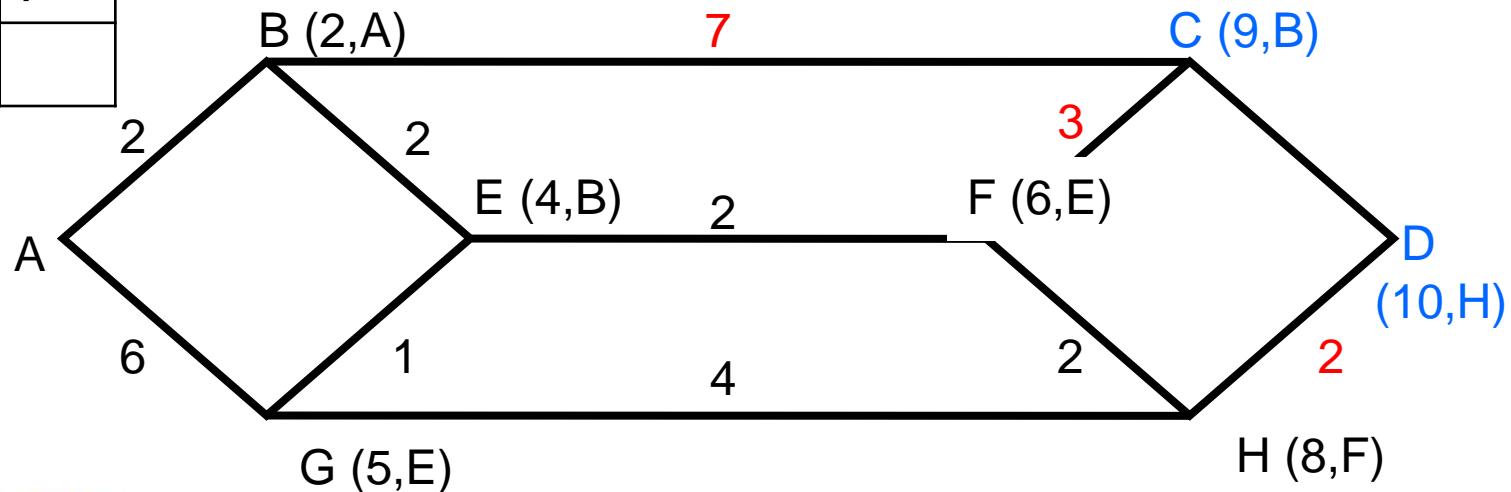
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A
E	4	B
G	5	E
F	6	E
H	8	F

Tentative Nodes

C	9	B
D	10	H



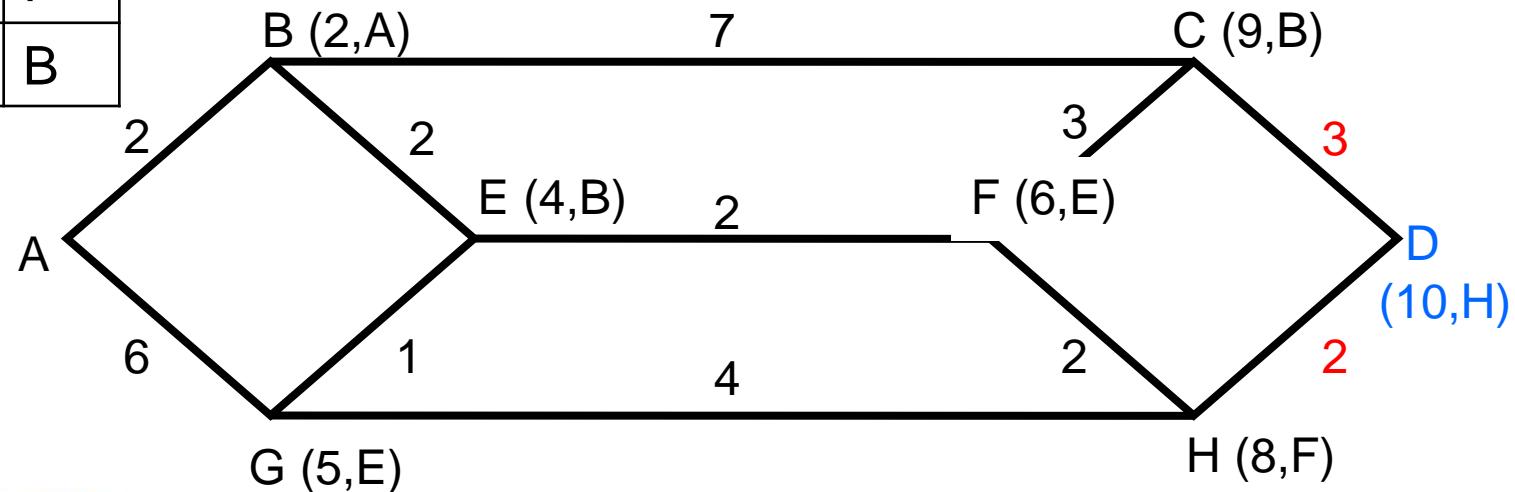
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A
E	4	B
G	5	E
F	6	E
H	8	F
C	9	B

Tentative Nodes

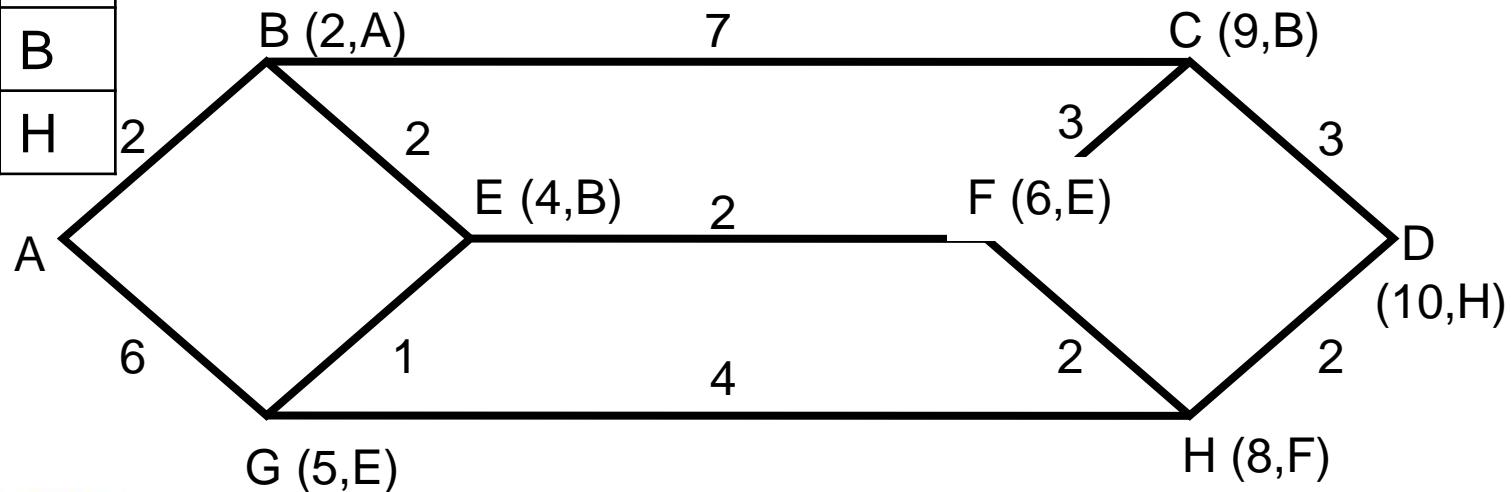
D	12	C
D	10	H



Example: Dijkstra's Algorithm

Routing Table

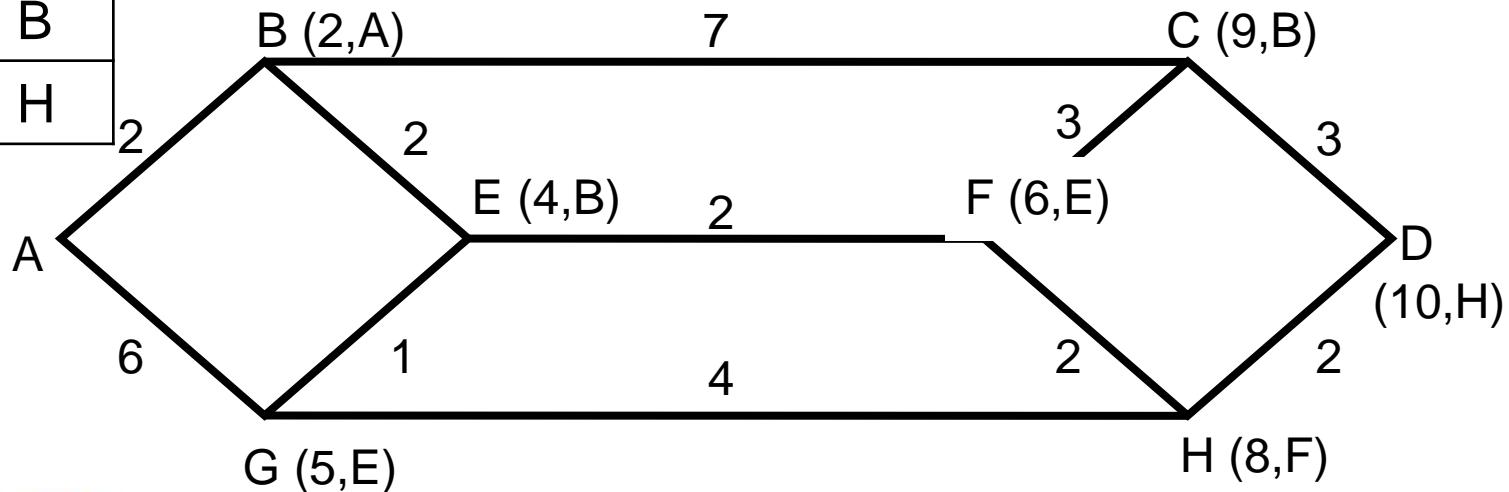
A	0	-
B	2	A
E	4	B
G	5	E
F	6	E
H	8	F
C	9	B
D	10	H



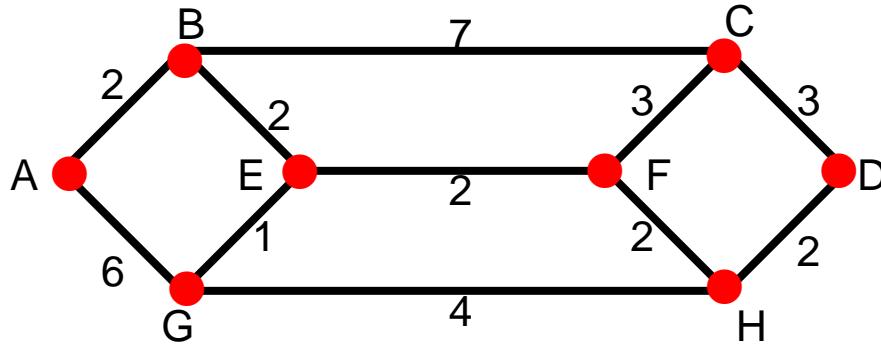
Example: Dijkstra's Algorithm

A	0	-
B	2	A
E	4	B
G	5	E
F	6	E
H	8	F
C	9	B
D	10	H

- Shortest connection from A to any other node
- Greedy algorithm! Follows minima!



Informal Description of Shortest Path Algo.



origin = starting point (e.g. point A)

destination = point that has a distance from the origin associated with itself (e.g. C, distance 9)

neighbour = destination that has a direct connection to another destination (e.g. B's neighbours A, E and C)

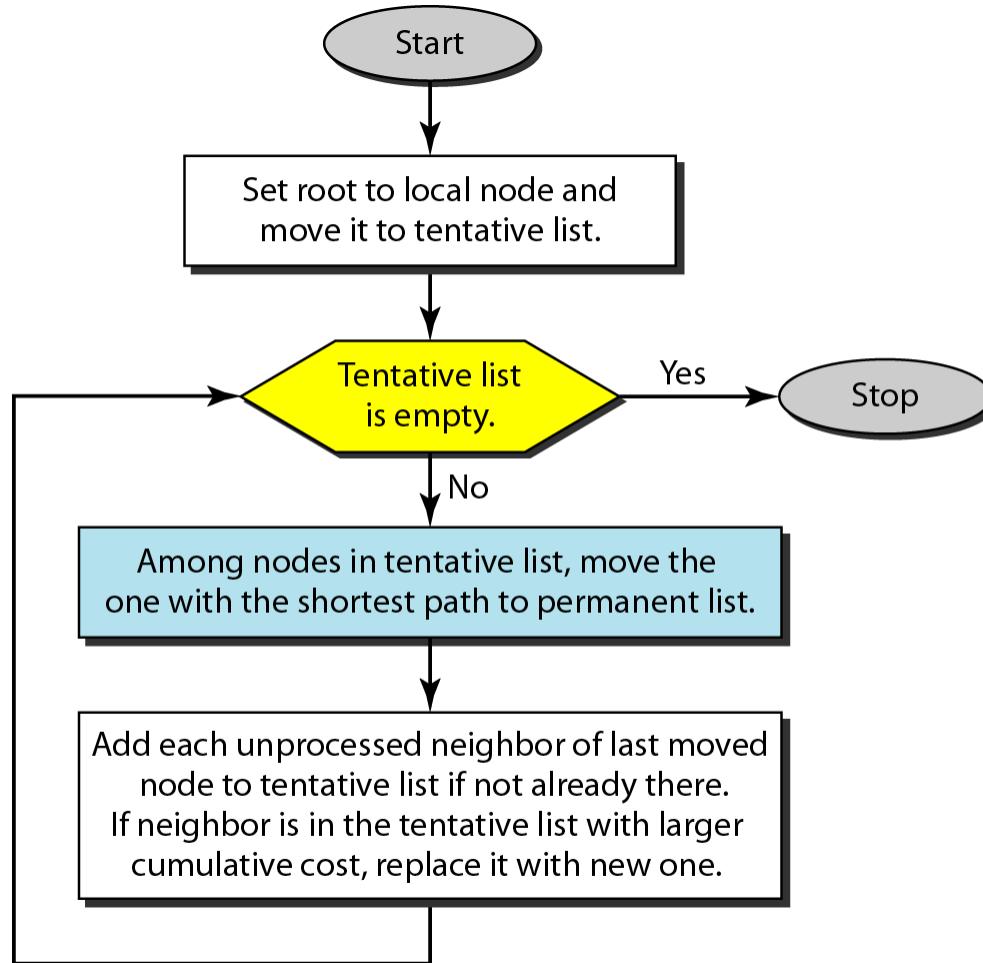
table = table of destinations with a known distance from the origin

bag = bag of destinations to be resolved

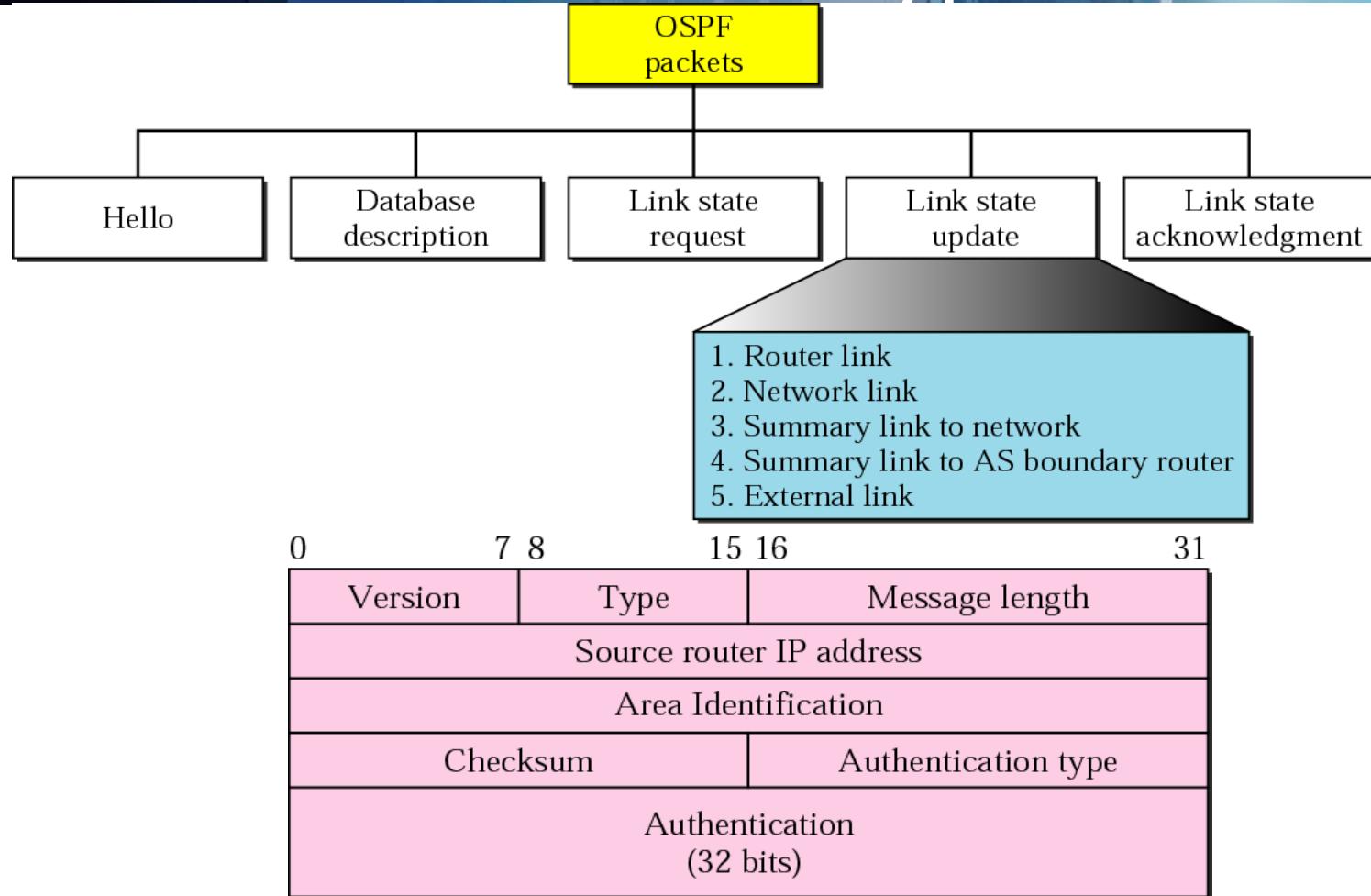
Informal Description of Shortest Path Algo.

- Table and bag are empty at the beginning
- Put the origin as a destination with the distance 0 into the bag
- As long as there are destinations in the bag
 - Take the destination with the smallest distance out of the bag and put it into the table (if the distance is smaller than an already existing entry for the same destination)
 - Put all the neighbours of that destination with their distance to the origin (=distance to the destination + distance from the destination to the neighbour) into the bag

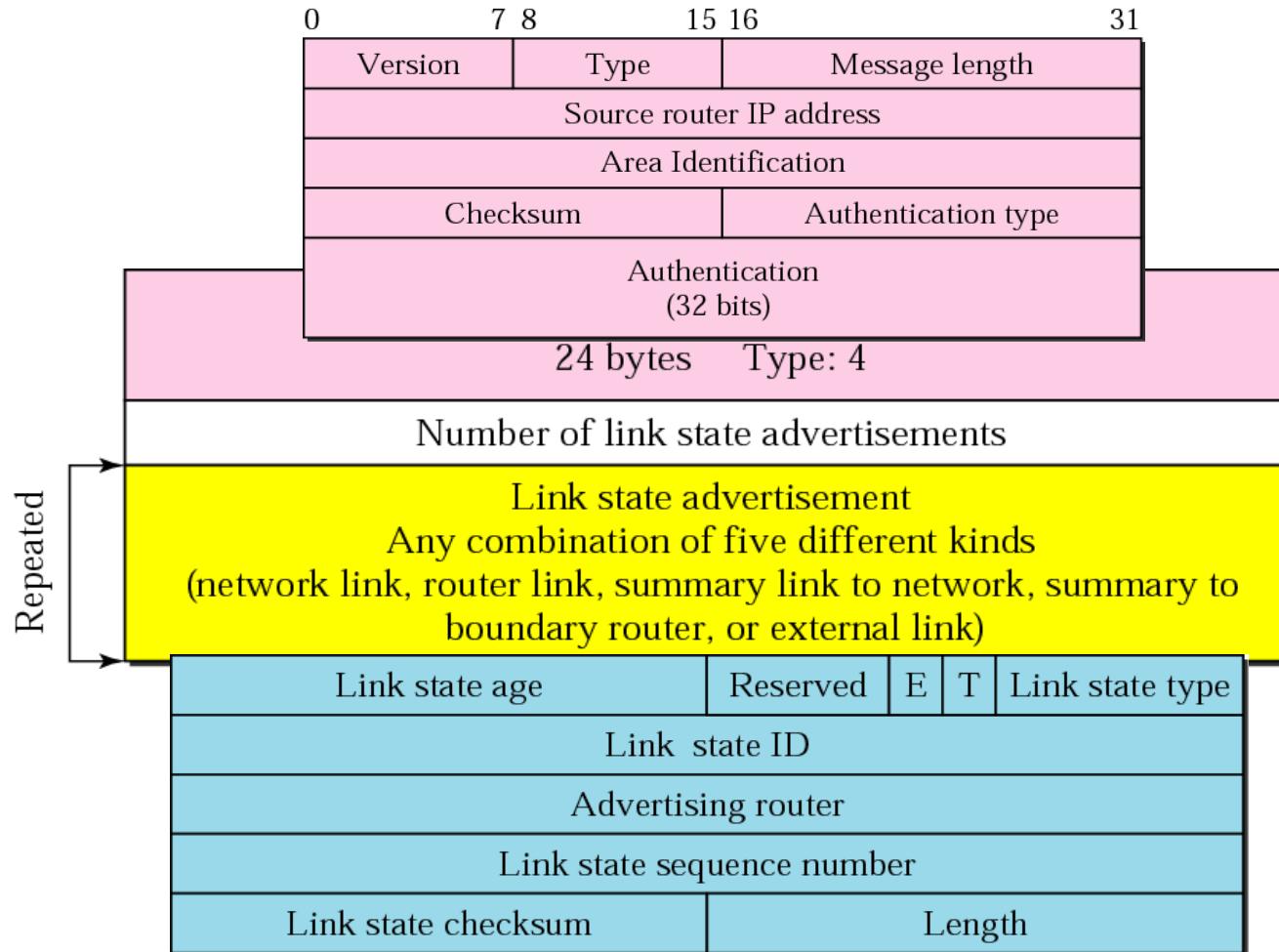




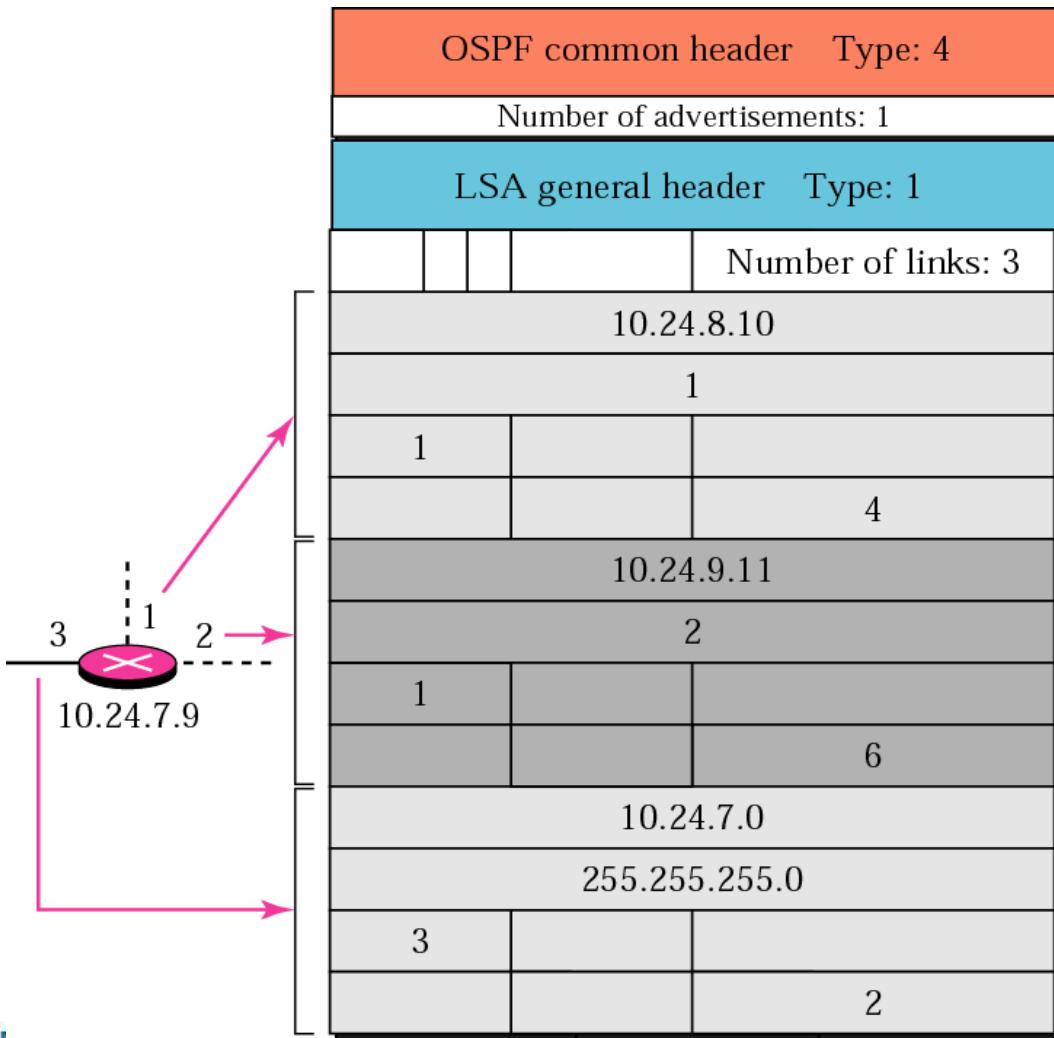
OSPF Packet Types



OSPF Link State Advertisement



OSPF LSA Example

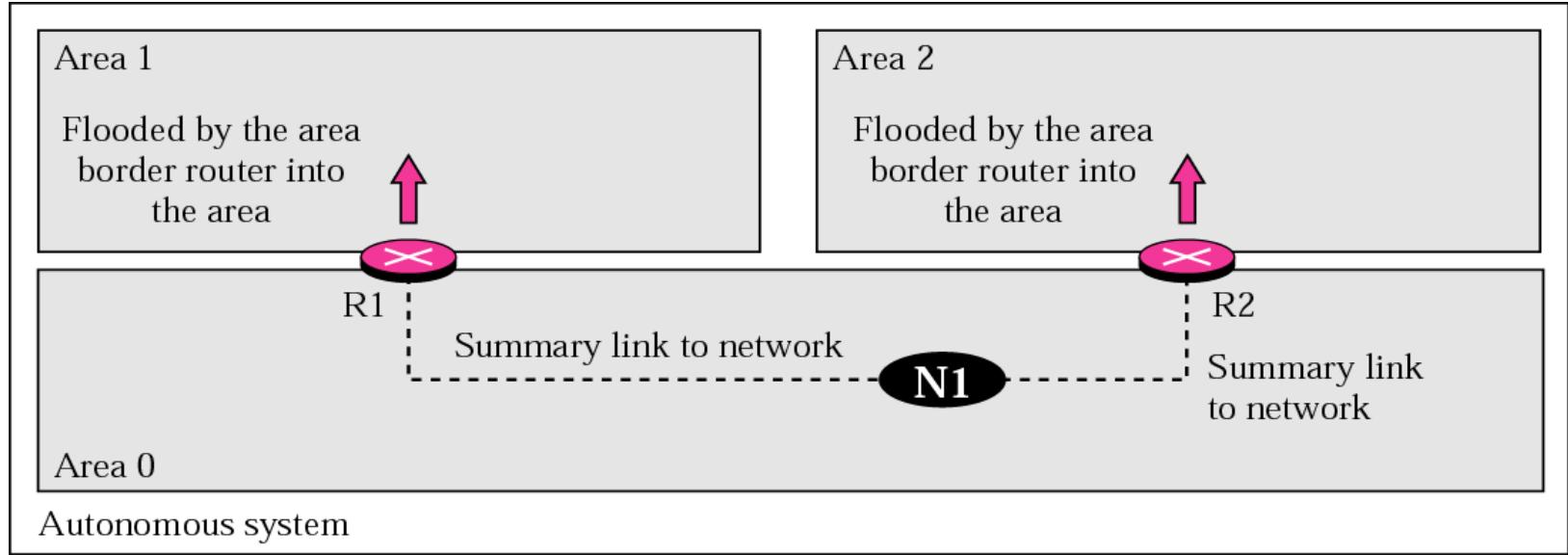


<i>Link Type</i>	<i>Link Identification</i>
Type 1: Point-to-point	Address of neighbor router
Type 2: Transient	Address of designated router
Type 3: Stub	Network address
Type 4: Virtual	Address of neighbor router

<i>Link Type</i>	<i>Link Data</i>
Type 1: Point-to-point	Interface number
Type 2: Transient	Router address
Type 3: Stub	Network mask
Type 4: Virtual	Router address

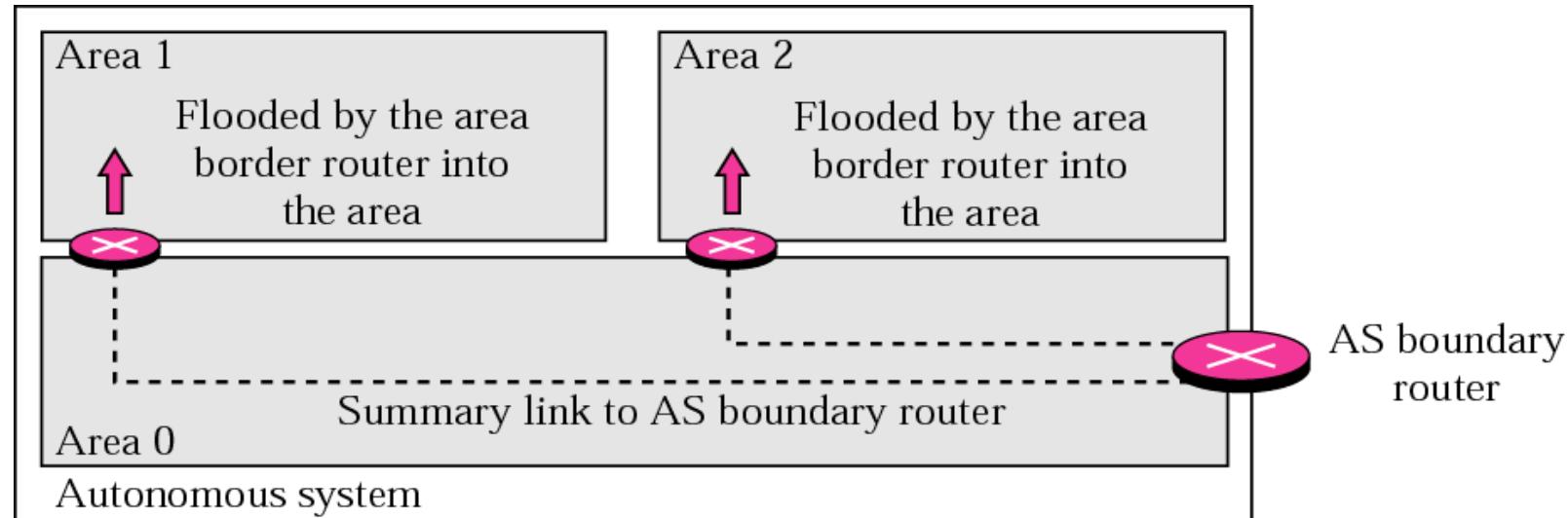
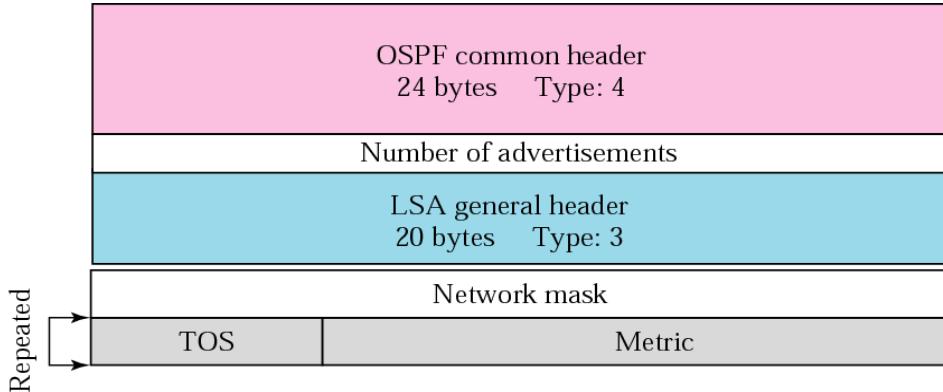
One router link advertisement

Areas in OSPF



- Limits flooding of advertisements to areas

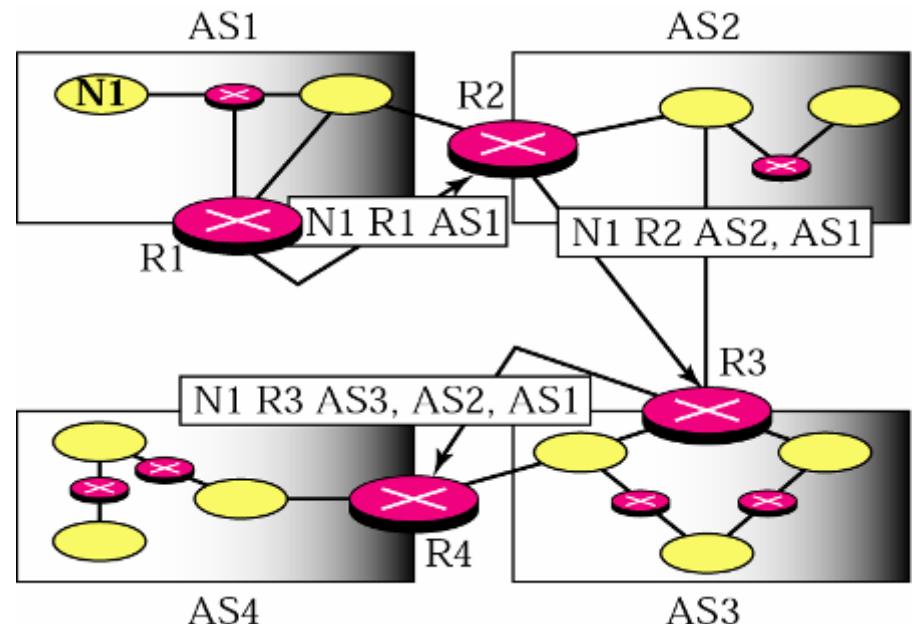
OSPF Summary Link



Border Gateway Protocol (BGP)

- Uses Path Vector Routing
- Advertisements include complete path to destination
- Router that forwards advertisement adds itself to the list
- Path can be checked for loops
- Policies are applied when incorporating new routes

Network	Next Router	Path
N01	R01	AS14, AS23, AS67
N02	R05	AS22, AS67, AS05, AS89
N03	R06	AS67, AS89, AS09, AS34
N04	R12	AS62, AS02, AS09



Tables at Autonomous Systems

Dest. Path

A1	AS1
A2	AS1
A3	AS1
A4	AS1
A5	AS1

A1 Table

AS 1

Dest. Path

C1	AS3
C2	AS3
C3	AS3

C1 Table

AS 3

Dest. Path

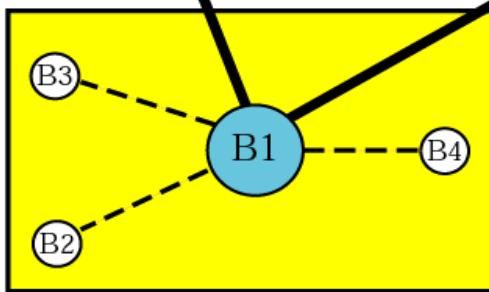
D1	AS4
D2	AS4
D3	AS4
D4	AS4

D1 Table

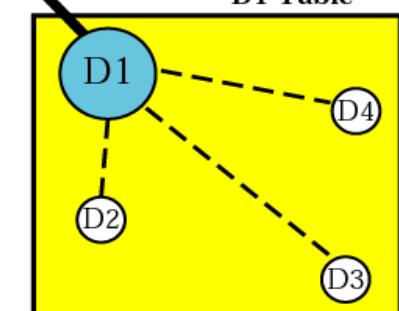
Dest. Path

B1	AS2
B2	AS2
B3	AS2
B4	AS2

B1 Table



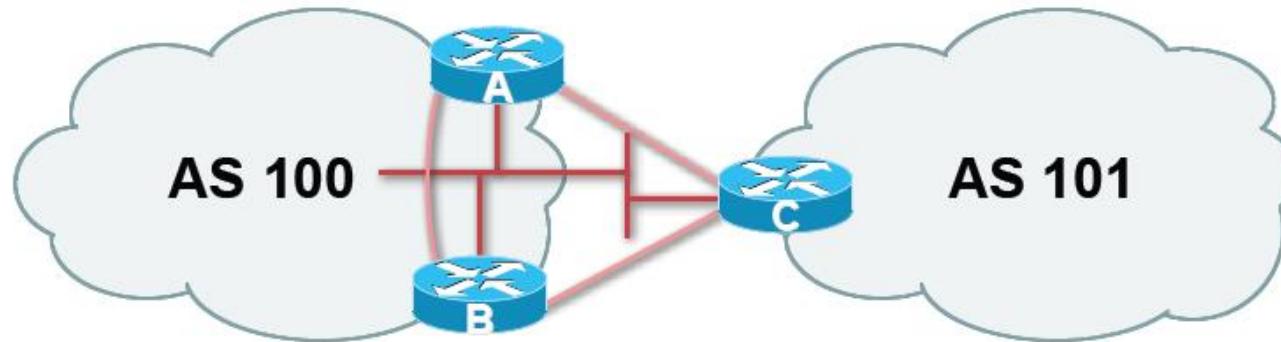
AS 2



AS 4



BGP Example



Router A in AS100

```
interface ethernet 5/0
 ip address 102.102.10.2 255.255.255.240
!
router bgp 100
 network 100.100.8.0 mask 255.255.252.0
 neighbor 102.102.10.1 remote-as 101
 neighbor 102.102.10.1 prefix-list RouterC in
 neighbor 102.102.10.1 prefix-list RouterC out
!
```

ip address of Router C
ethernet interface

ip address on
ethernet interface

Router C in AS101

```
interface ethernet 1/0/0
 ip address 102.102.10.1 255.255.255.240
!
router bgp 101
 network 100.100.8.0 mask 255.255.252.0
 neighbor 102.102.10.2 remote-as 100
 neighbor 102.102.10.2 prefix-list RouterA in
 neighbor 102.102.10.2 prefix-list RouterA out
!
```

ip address on
ethernet interface

Inbound and
outbound filters

ip address of Router A
ethernet interface

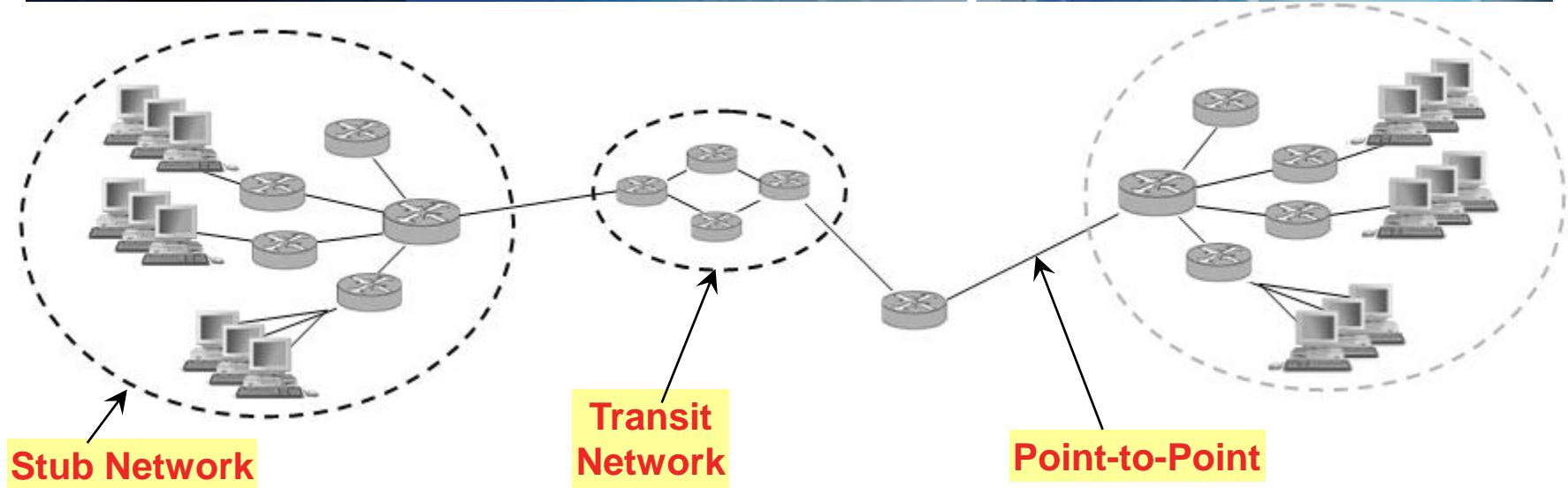
Inbound and
outbound filters



BGP-4: Border Gateway Protocol

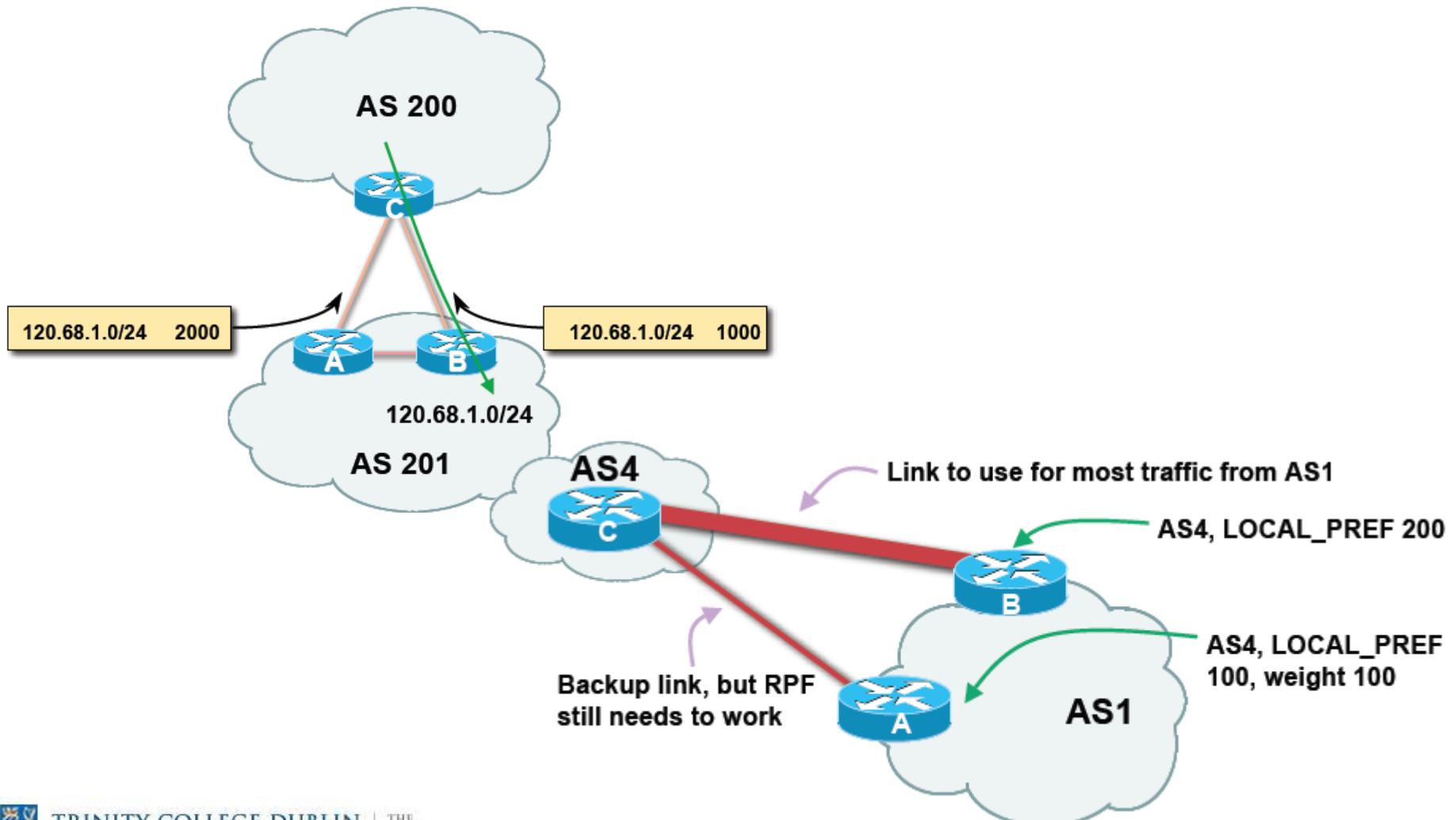
- AS Types
 - Stub AS: has a single connection to one other AS
 - Carries local traffic only
 - Multihomed AS: has connections to more than one AS
 - Refuses to carry transit traffic
 - Transit AS: has connections to more than one AS
 - Carries both transit and local traffic
- Each AS has:
 - One or more border routers
 - One BGP *speaker* that advertises:
 - Local networks
 - Other reachable networks (transit AS only)
 - Gives *path* information

Autonomous Systems

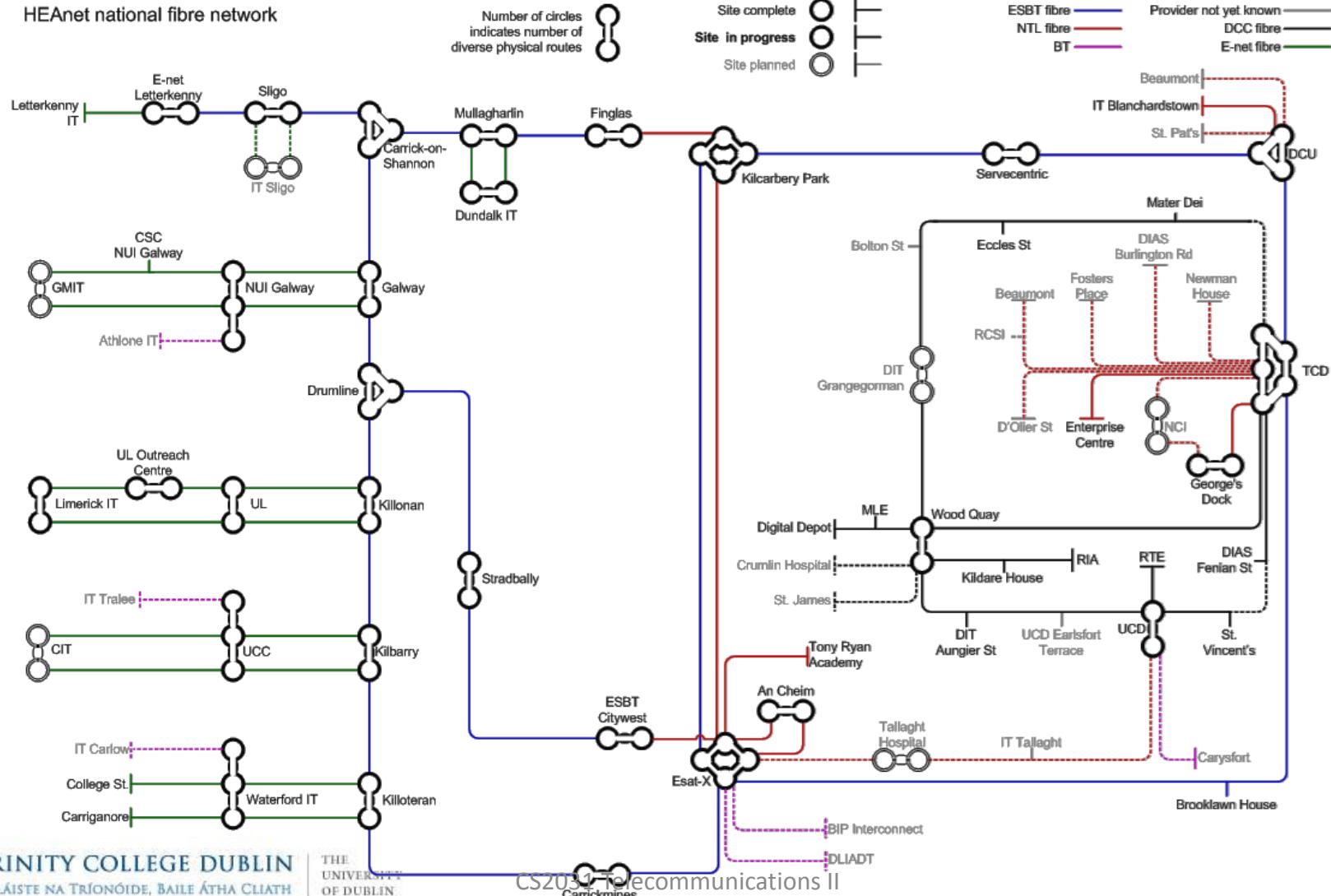


- Stub network
 - Network that does not forward to other network
- Transit network
 - Network that forwards traffic between other networks
- Point-to-point link

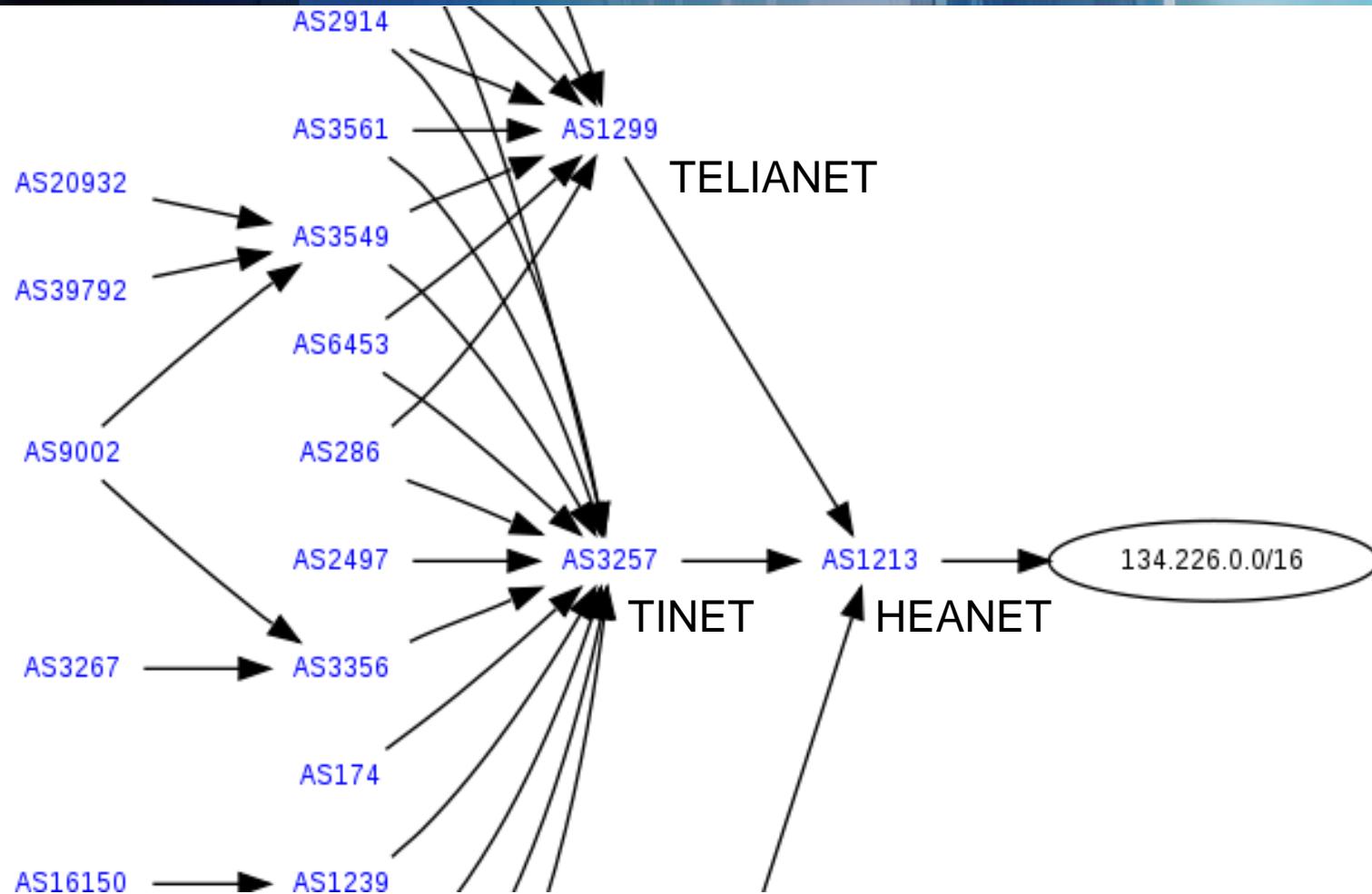
Multiple Exits & Backup Links



HEAnet Fibre Network



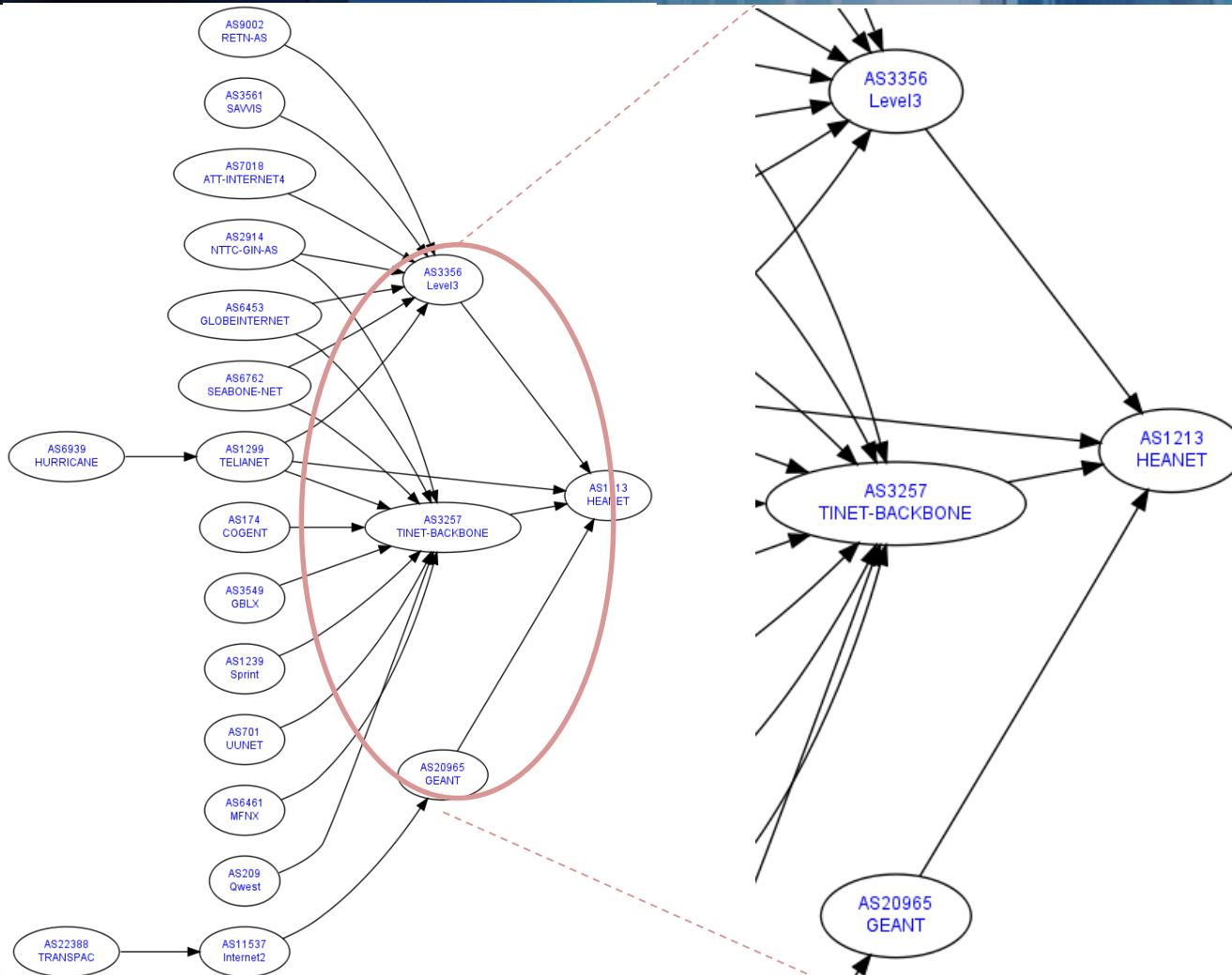
AS1213 - HEANET



<http://www.robtex.com/route/134.226.0.0-16.html>

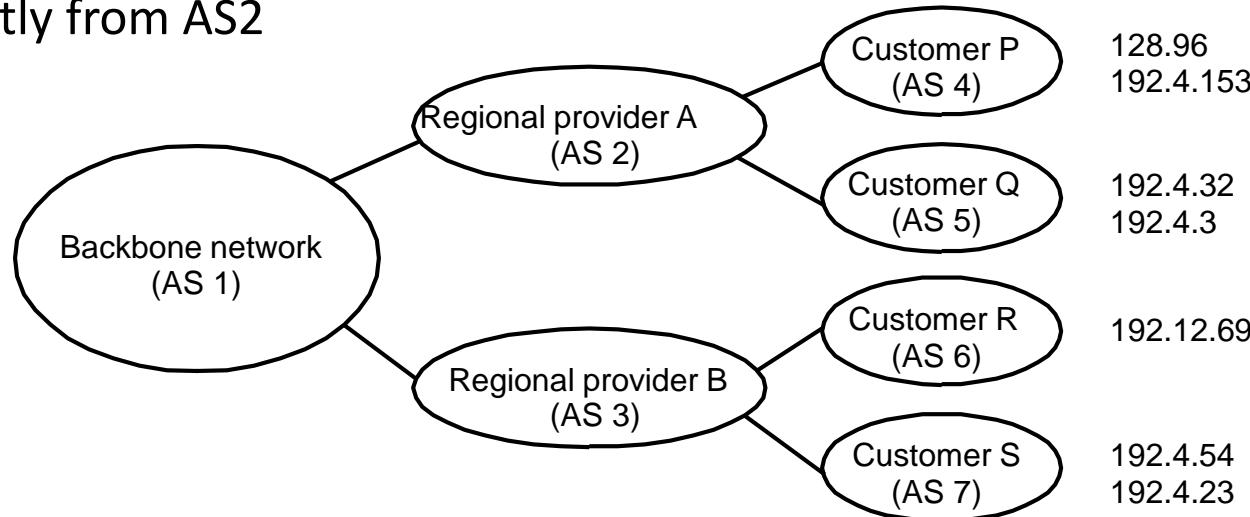


AS1213 - HEANET



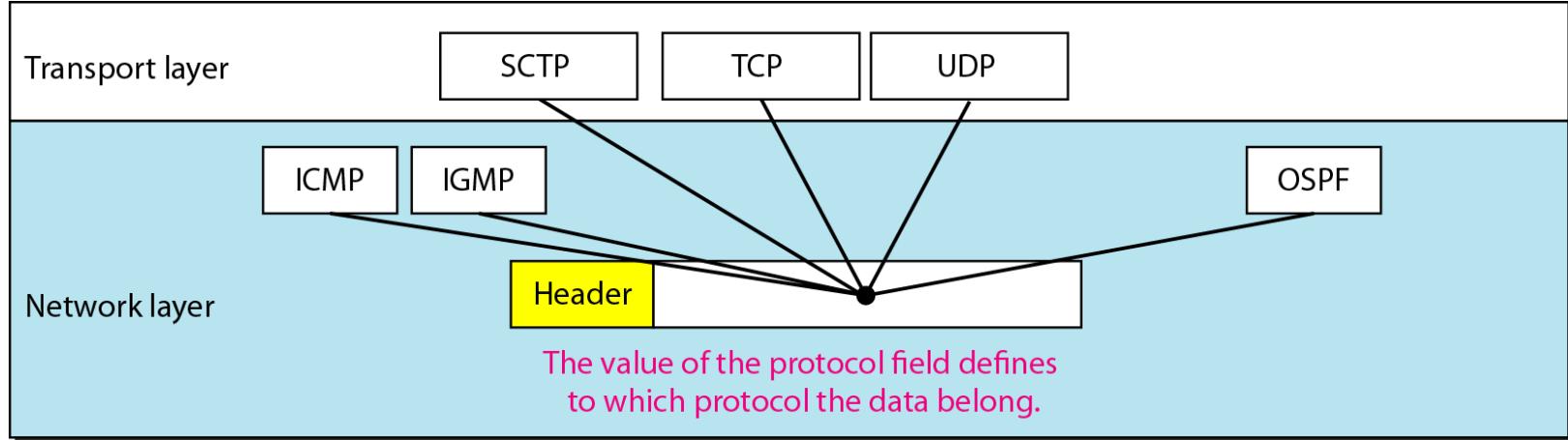
BGP Example

- Speaker for AS2 advertises reachability to P and Q
 - network 128.96, 192.4.153, 192.4.32, and 192.4.3, can be reached directly from AS2



- Speaker for backbone advertises
 - networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS1, AS2).
- Speaker can cancel previously advertised paths

Routing Protocols in the Stack



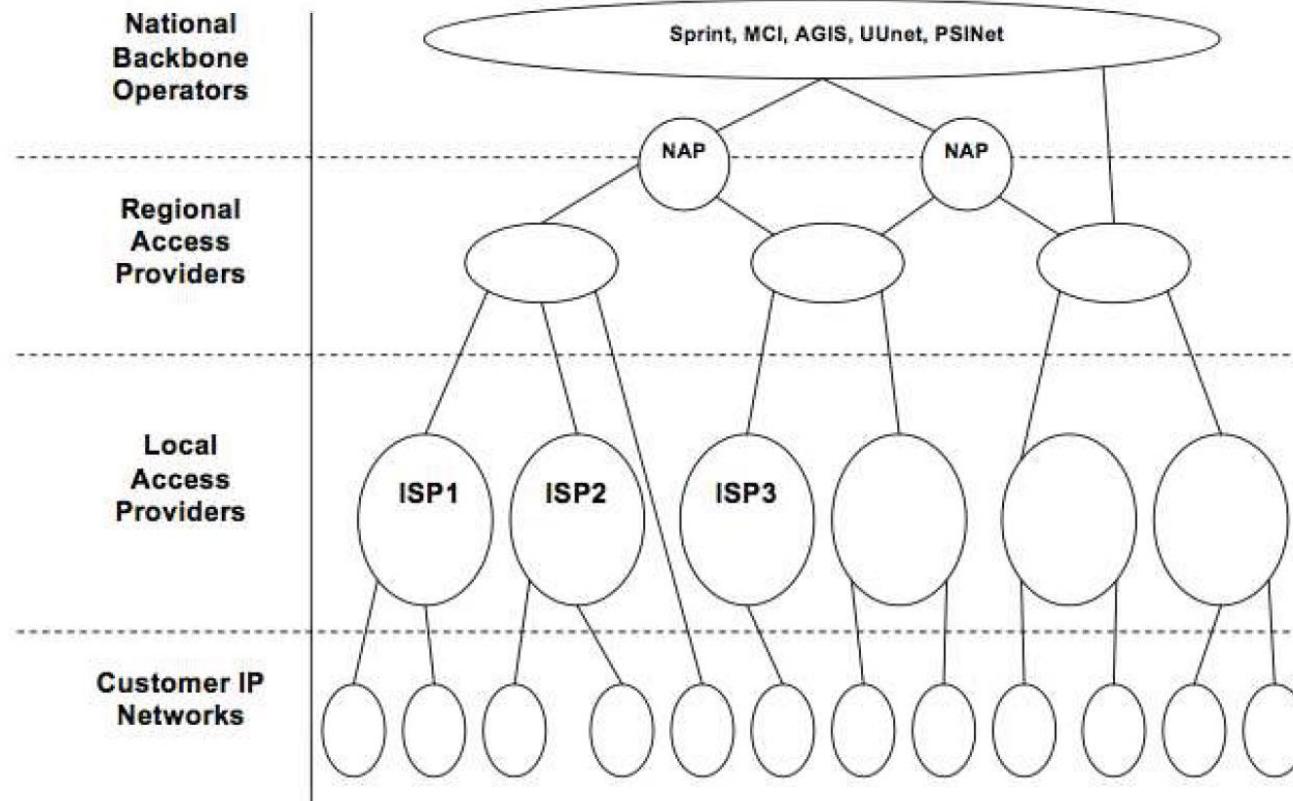
Protocol	Underlying Protocol	Protocol ID or Port
OSPF	IP	89
RIPv2	UDP	520
BGP	TCP	179

Summary: Routing

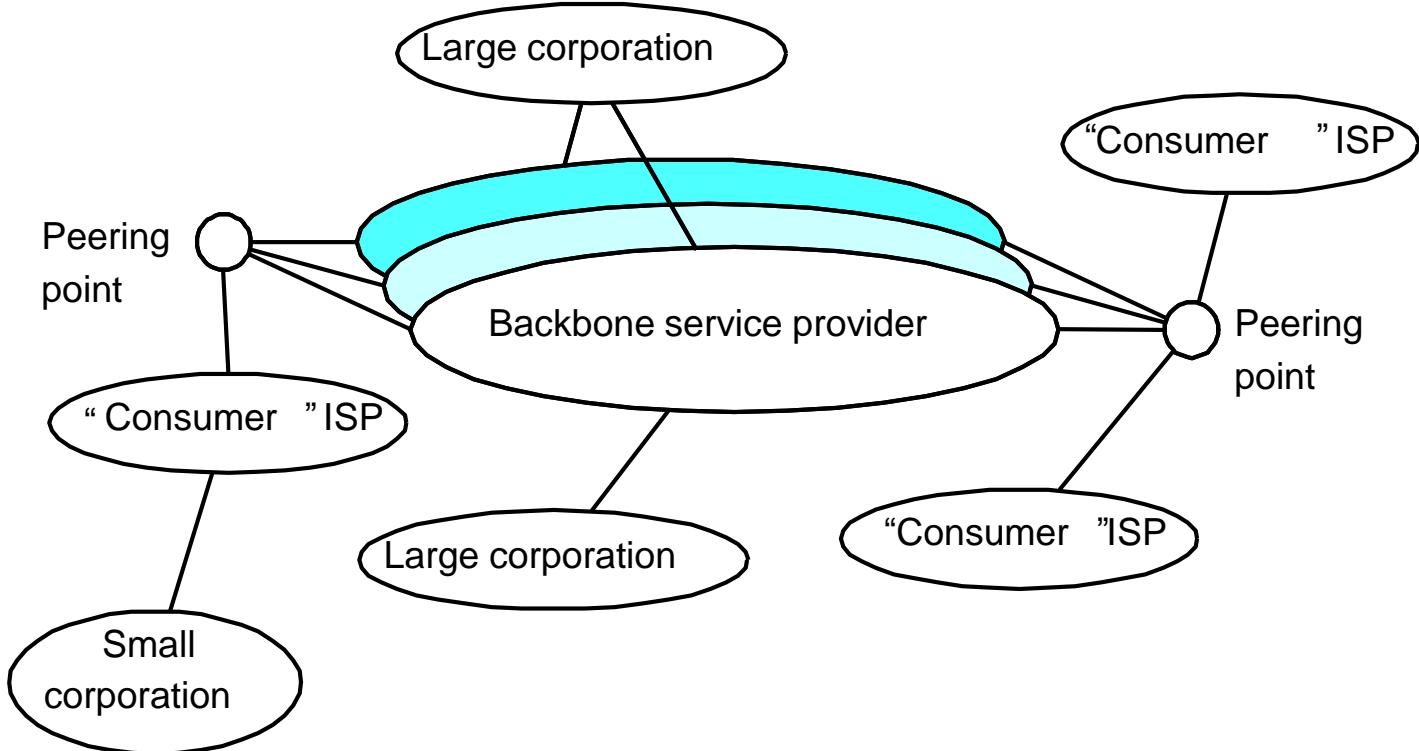
- Autonomous Systems
 - Stub network
 - Transient network
 - Point-to-point link
- Distance Vector routing
 - Share complete information with neighbours
 - Count-to-Infinity problem
 - Example: Routing Information Protocol (RIP)
- Link State routing
 - Share information about neighbours with everyone
 - Dijkstra's Shortest-Path Algorithm
 - Example: Open Shortest Path First (OSPF)



Traditional Logical Internet Topology

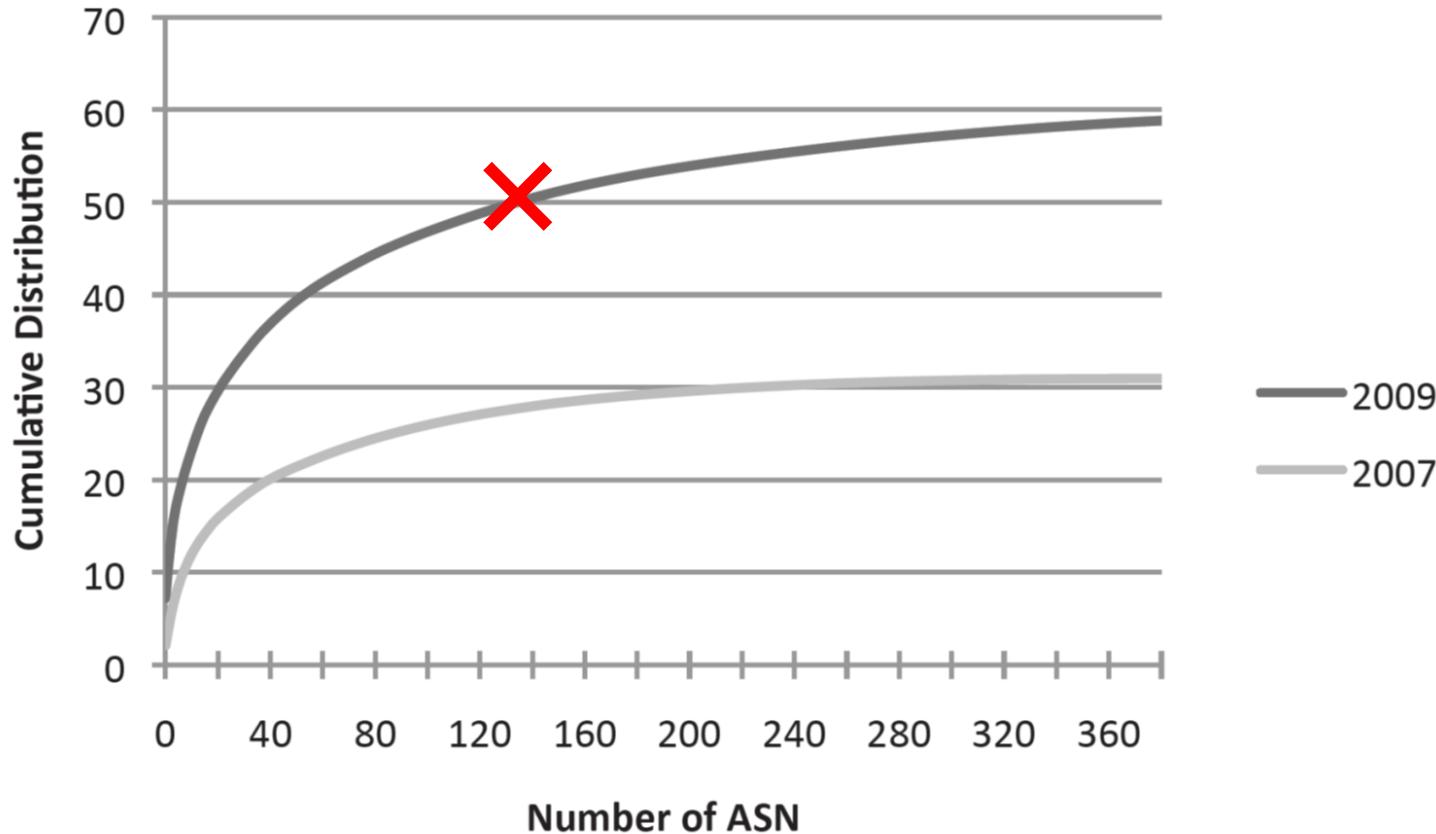


Structure of the Internet



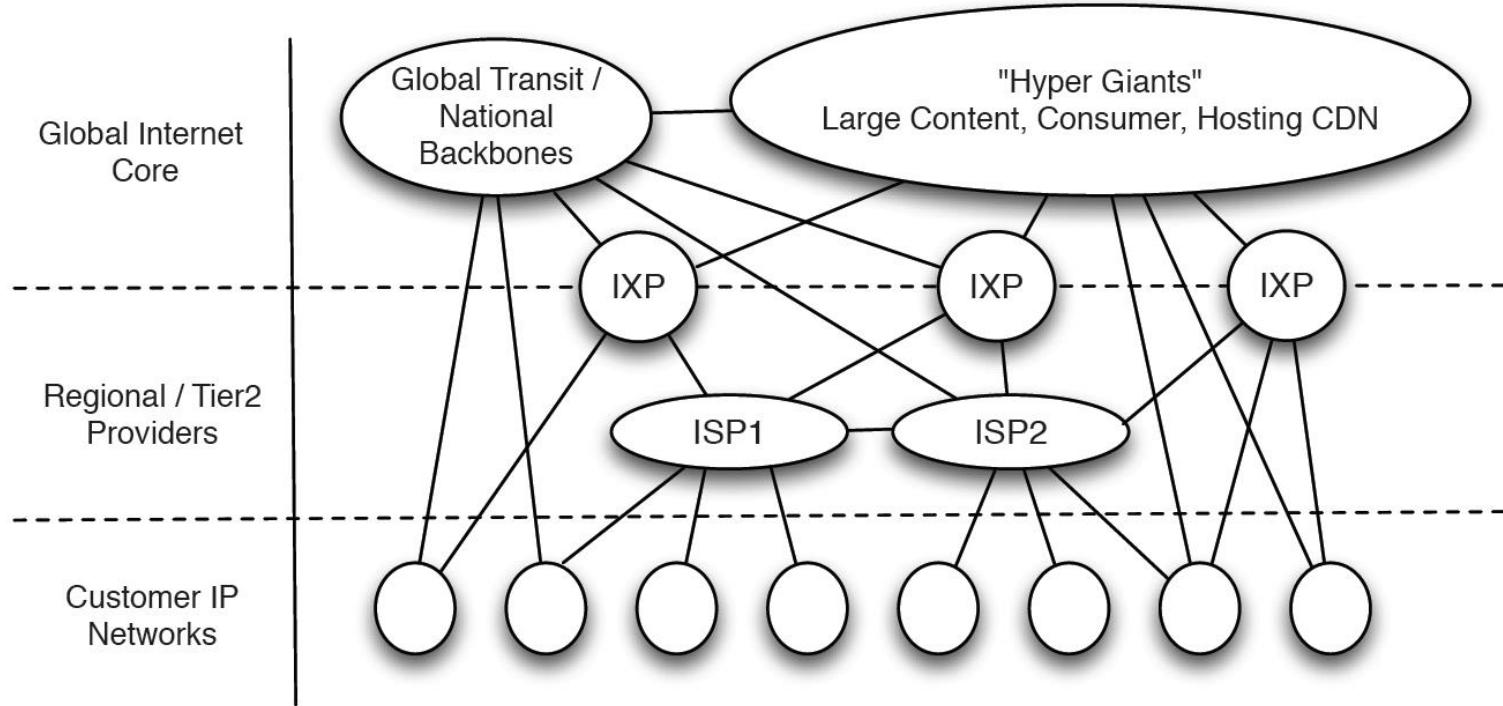
- Tier 1 provider eg. Verizon (WorldCom/MCI), Global Crossing
- Peering points – Exchange points between ISPs

Cumulative Inter-Domain Traffic

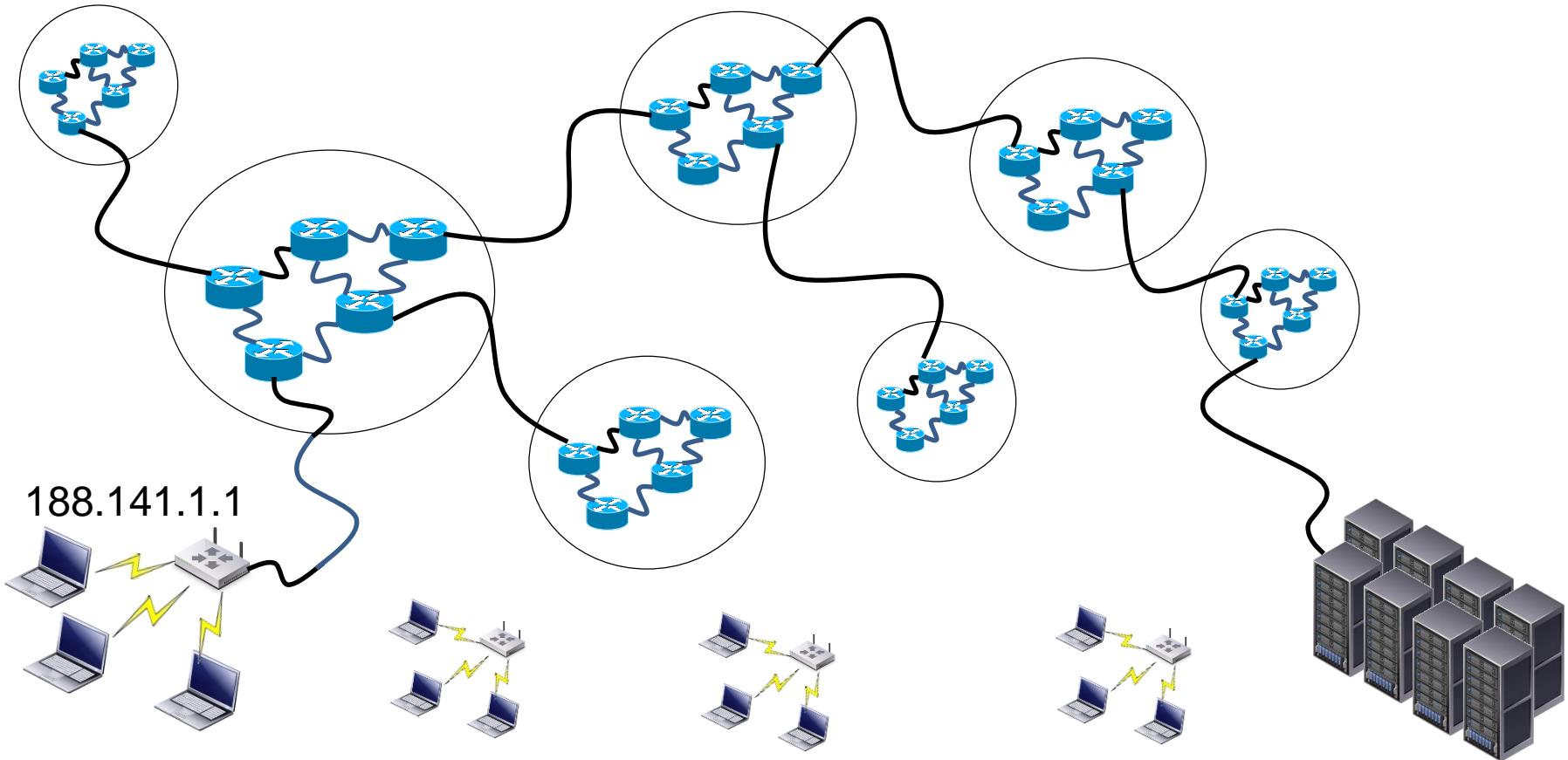


150 ASN in 2009 – According to C. Labovitz now 30

Emerging Logical Internet Topology



Internet = Network of Networks





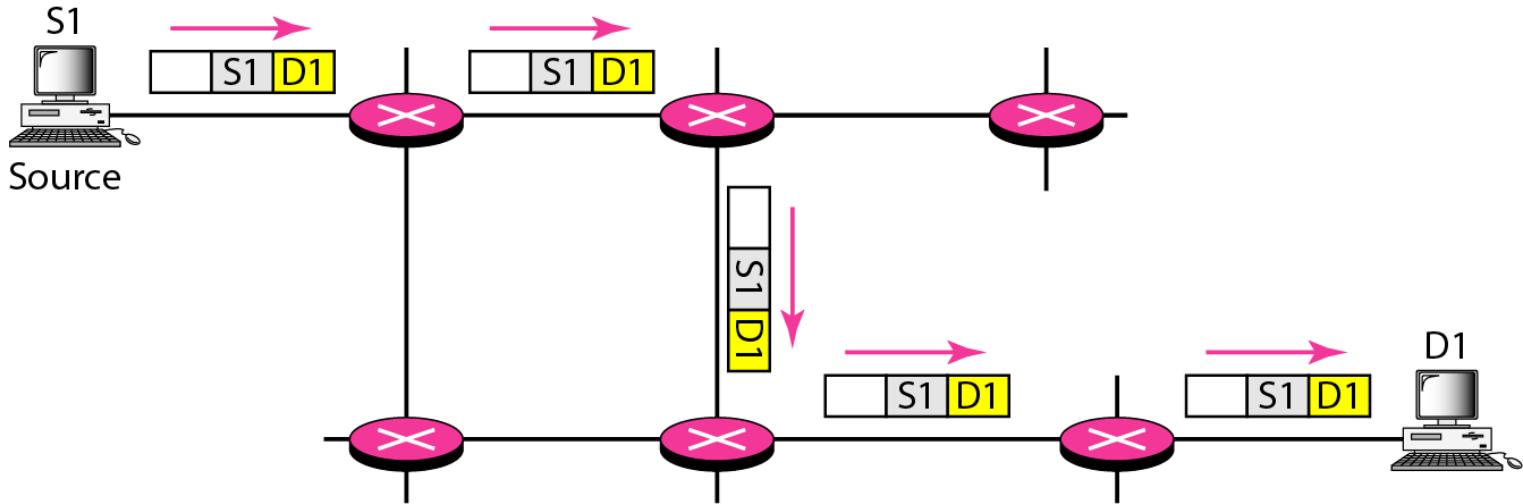


CS2031 Telecommunications II

Multicast Routing

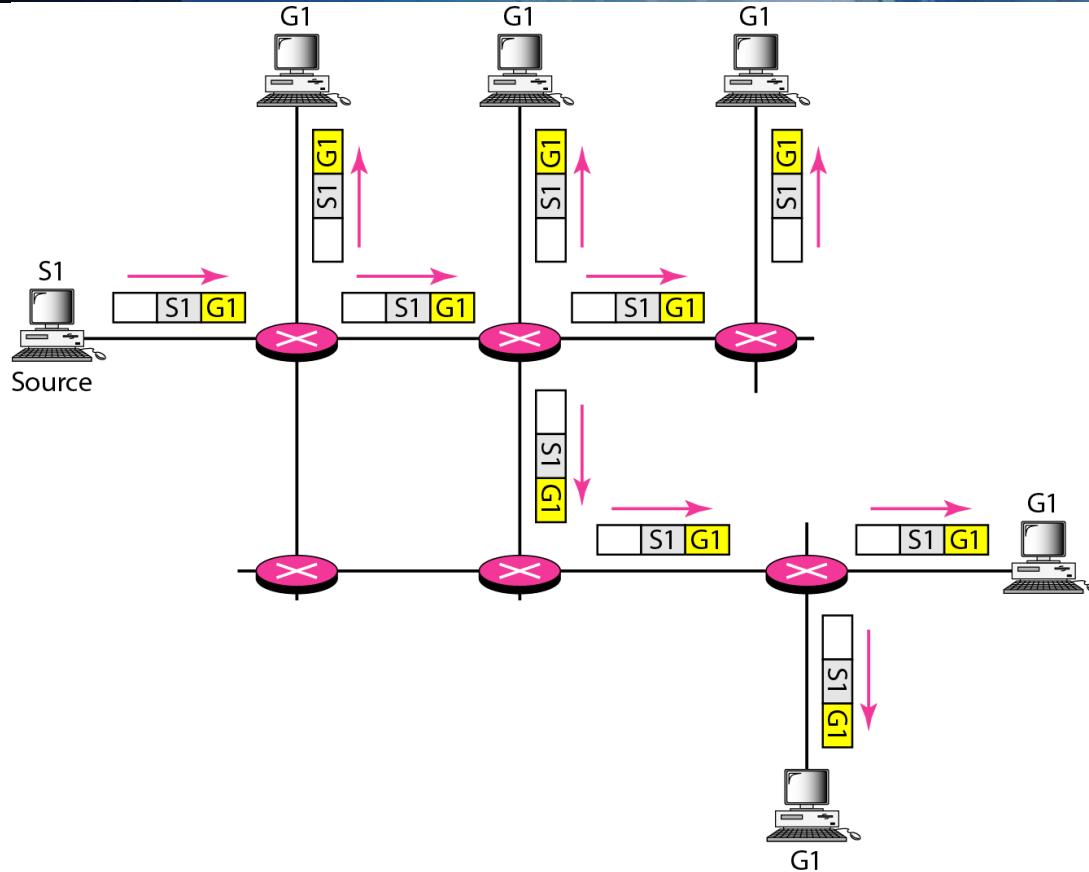


Routing & Unicast



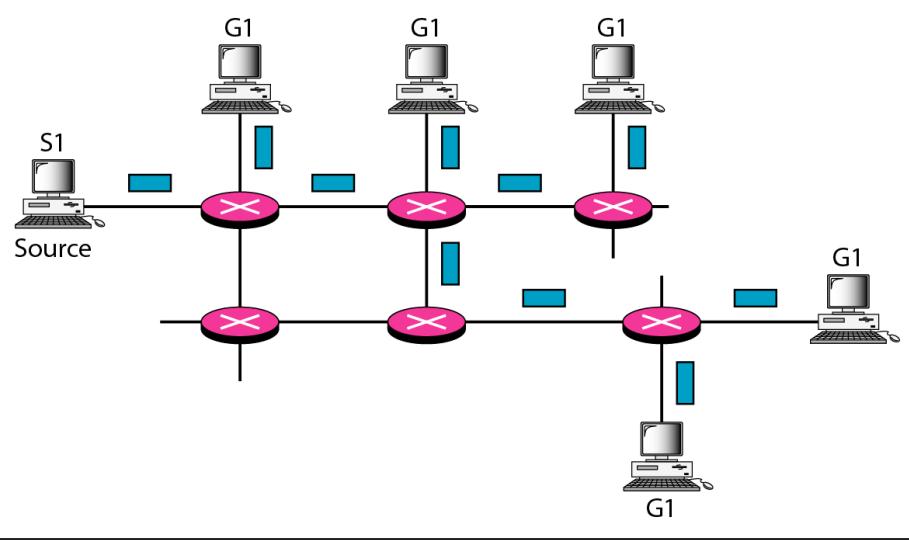
- Routers guide traffic towards destination

Multicast

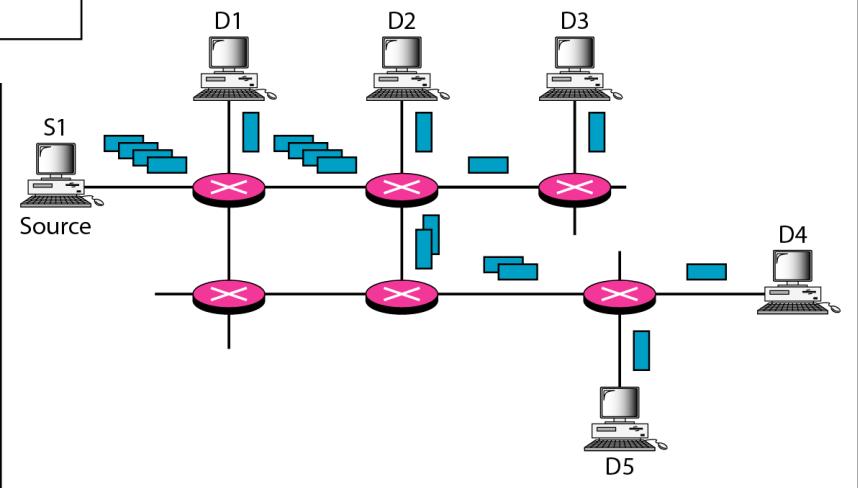


- G1= multicast address e.g. 230.0.0.1

Multicast vs Multiple Unicasts



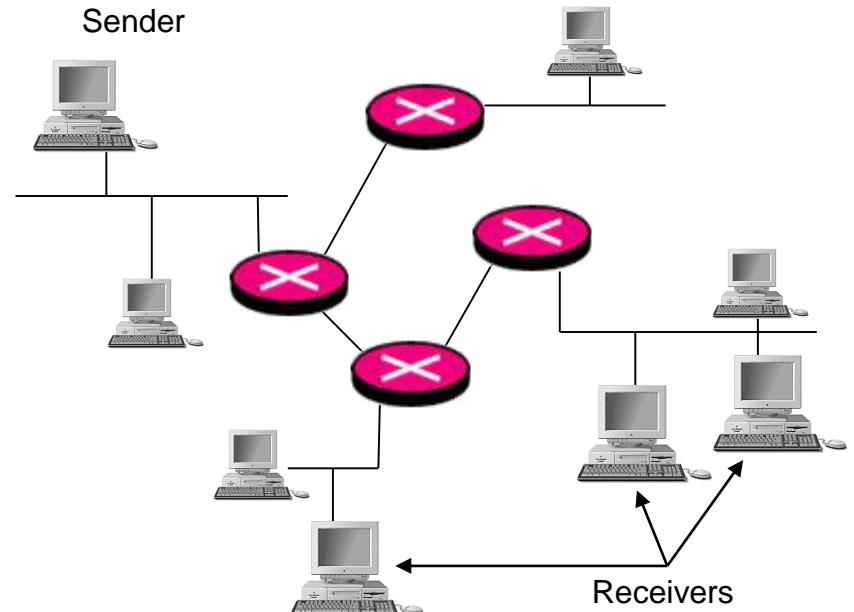
a. Multicasting



b. Multiple unicasting

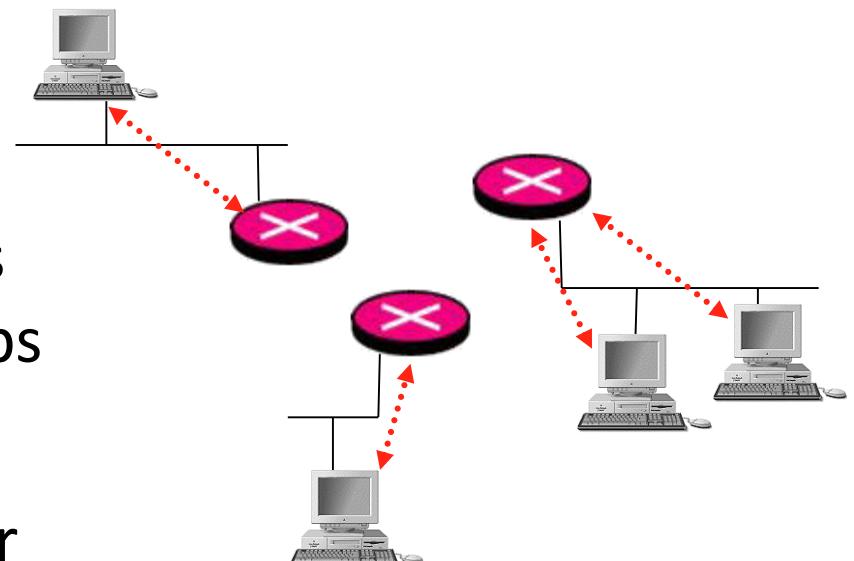
Multicast Overview

- Multicast requires group management
- Receivers join&leave multicast groups
- Multicast Addresses:
224.0.0.0 – 239.255.255.255
or 224.0.0.0/4



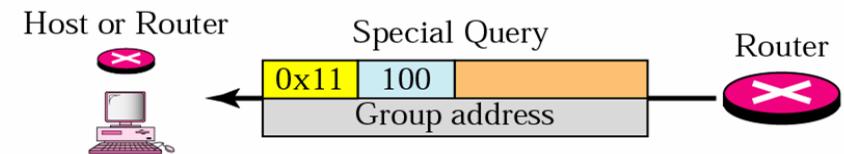
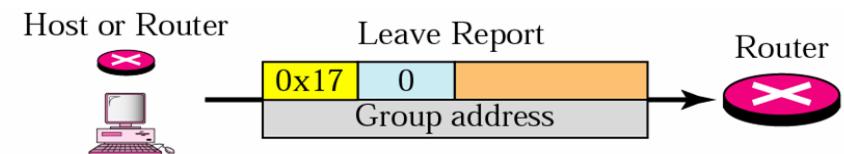
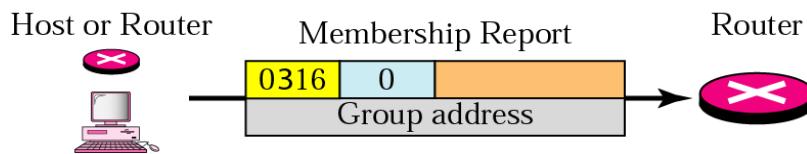
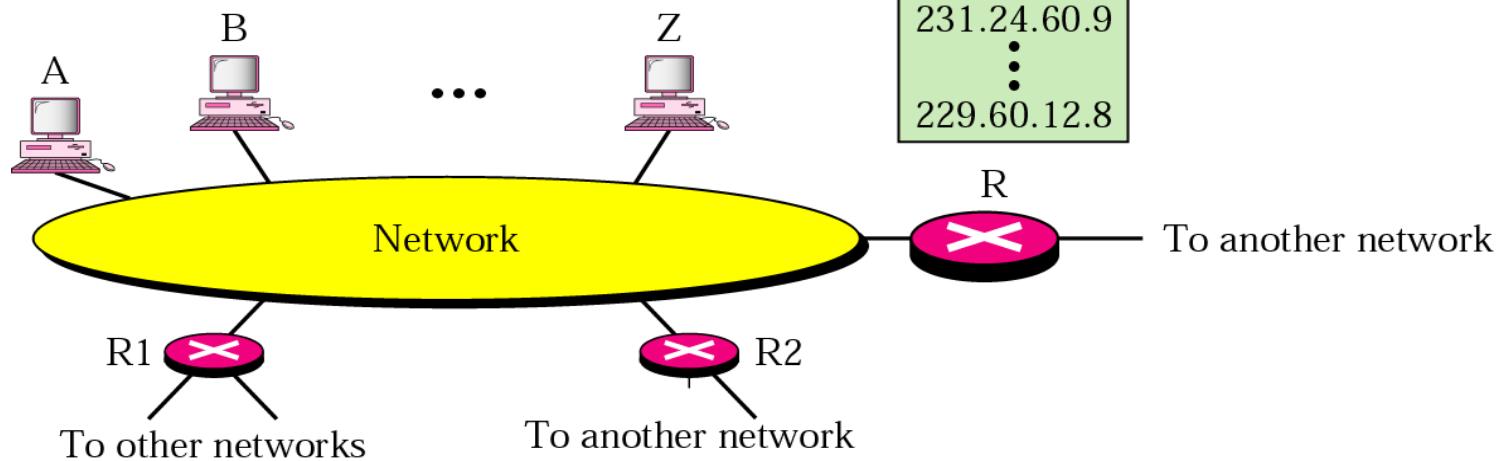
Internet Group Management Protocol (IGMP)

- Defines communication between hosts and router
- Specifies messages for hosts for joining and leaving groups
- Specifies query messages for routers



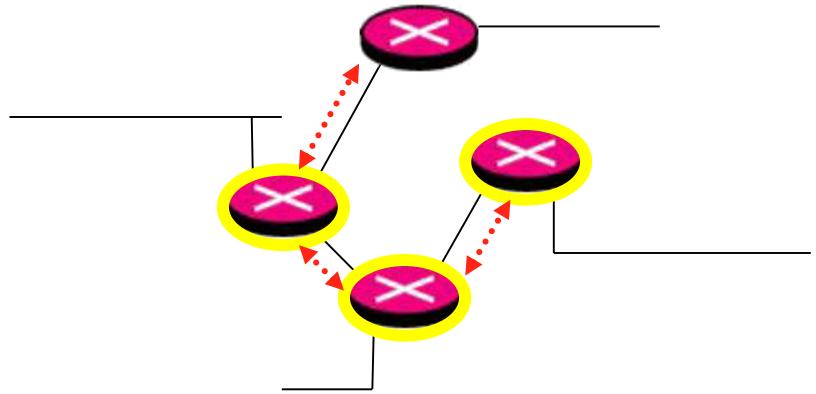
IGMP Operation

List of groups
having loyal members

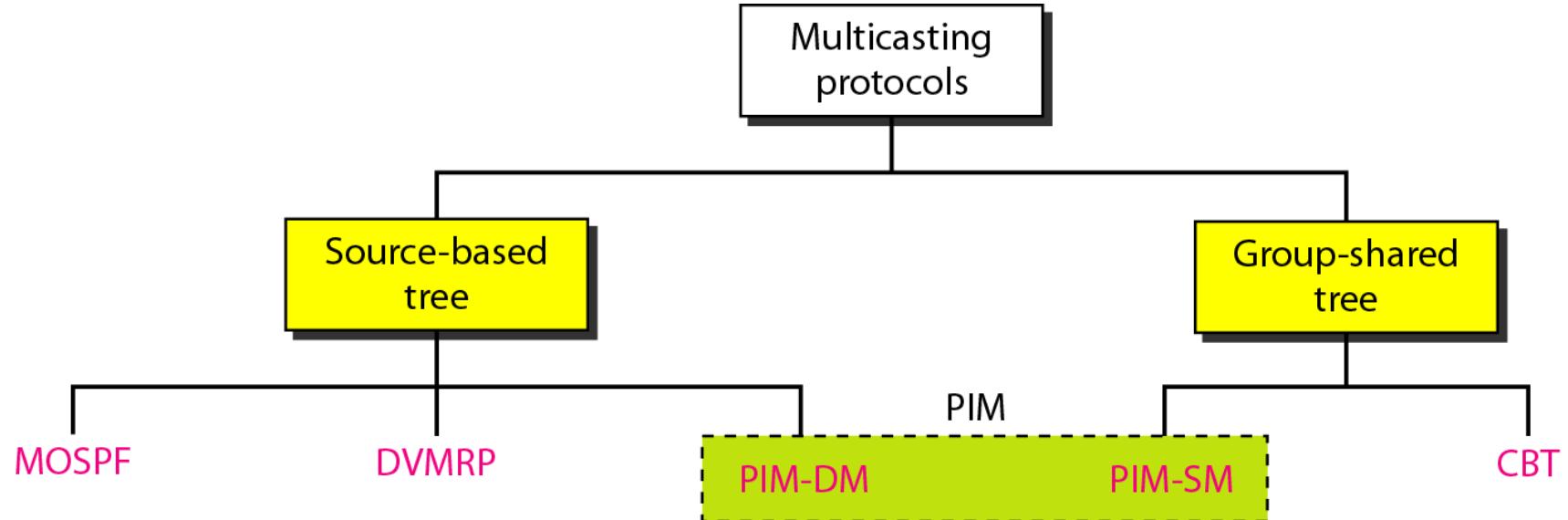


Network-Layer Multicast Protocols

- Distance Vector Multicast Routing Protocol (DVMRP)
- Multicast Open Shortest Path First protocol (MOSPF)
- Protocol Independent Multicast (PIM)



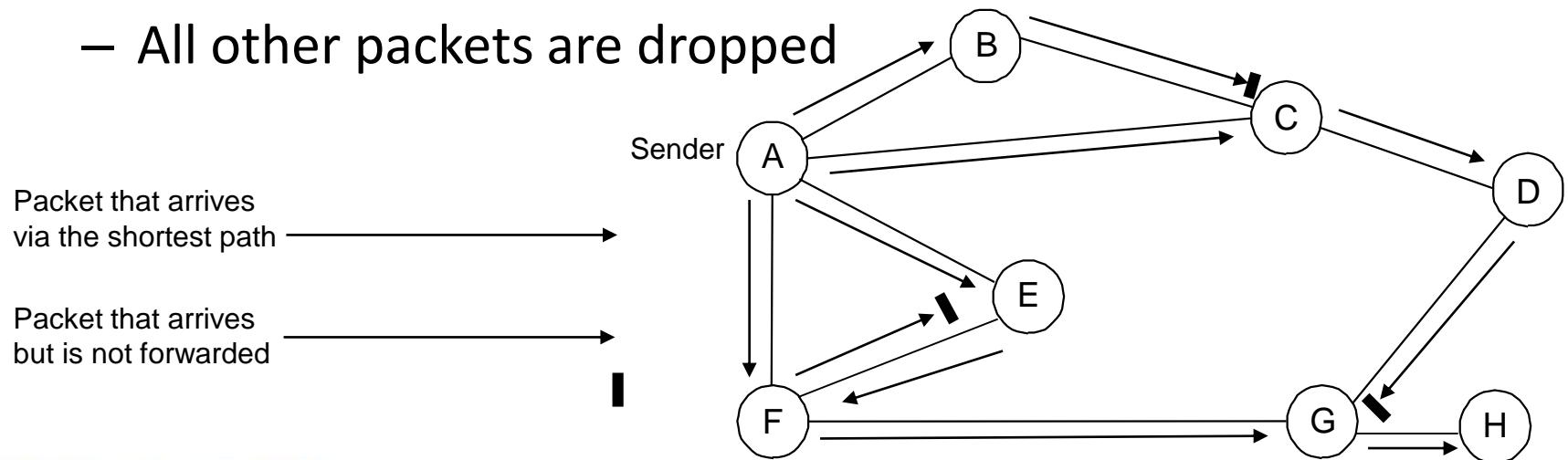
Multicast Routing Protocols



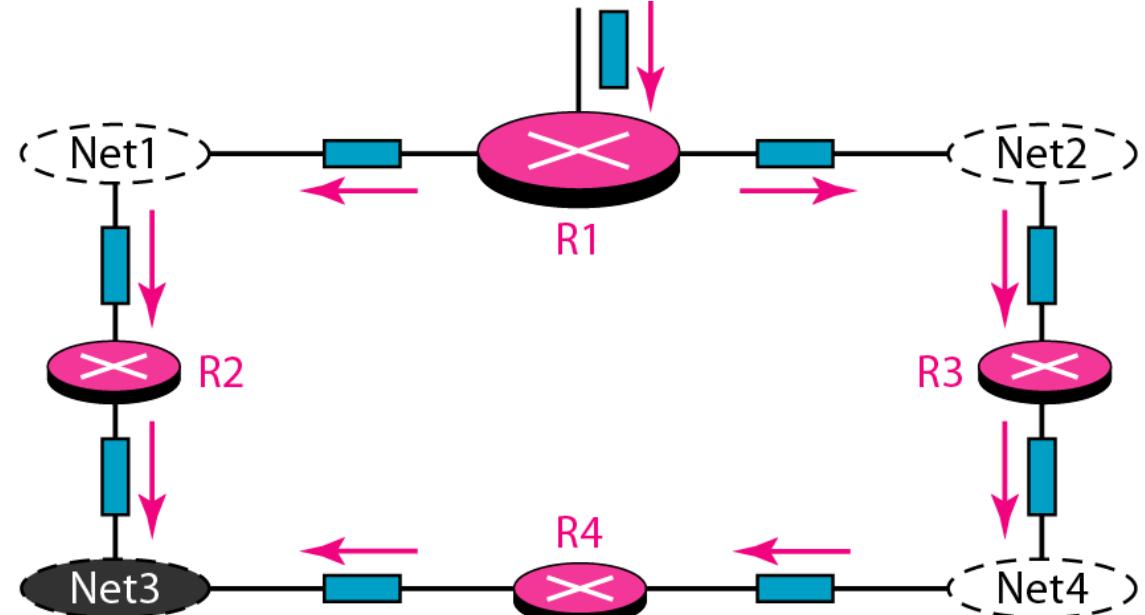
- Intra-AS
 - MOSPF
 - DVMRP
 - PIM
 - Sparse mode
 - Dense mode
- Inter-AS
 - MBGP + MSDP
 - BGMP + MASC

Reverse-Path Forwarding (RPF)

- Reverse-path forwarding simulates spanning tree routing without keeping state in the router
 - Each router knows shortest path to destination
 - Packets from A arriving on next hop to A are presumed to have followed shortest route from A, so they are forwarded on all other links
 - All other packets are dropped



Problem with RPF

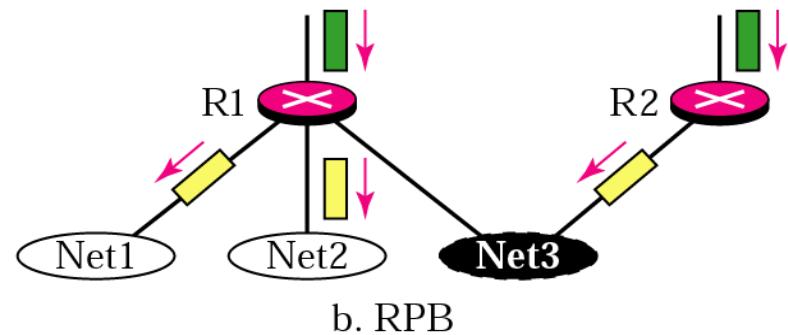
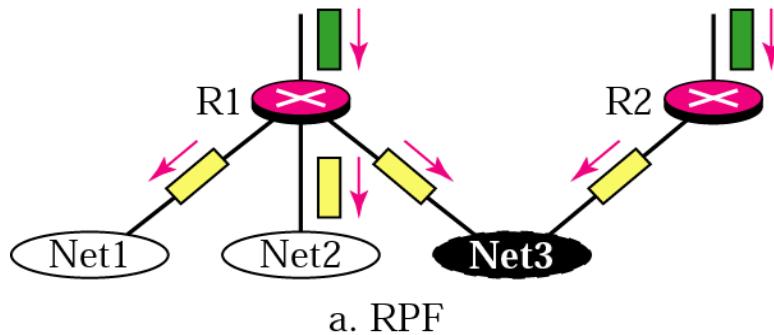


Net3 receives two copies of the packet

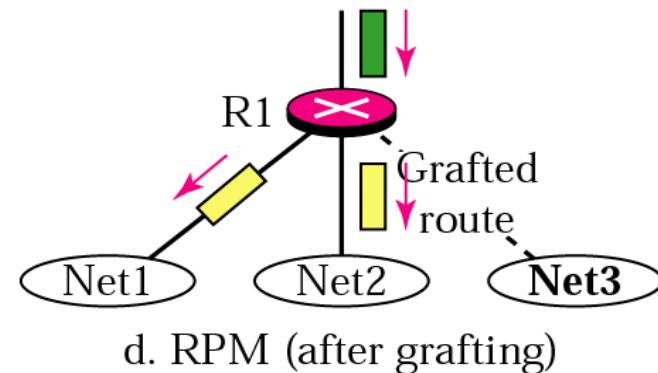
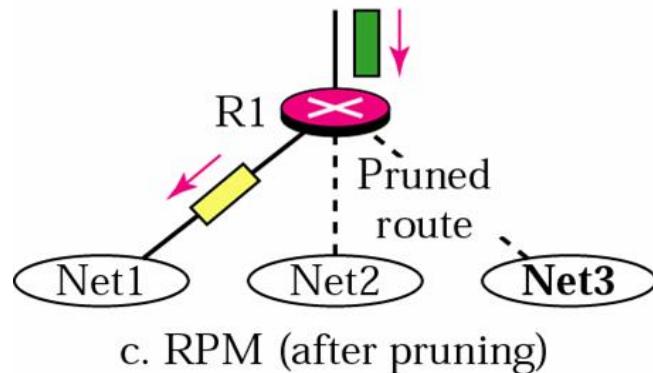
Reverse Path Broadcast/Multicast

- Reverse Path Broadcast

R1 is the parent of Net1 and Net2.
R2 is the parent of Net3.



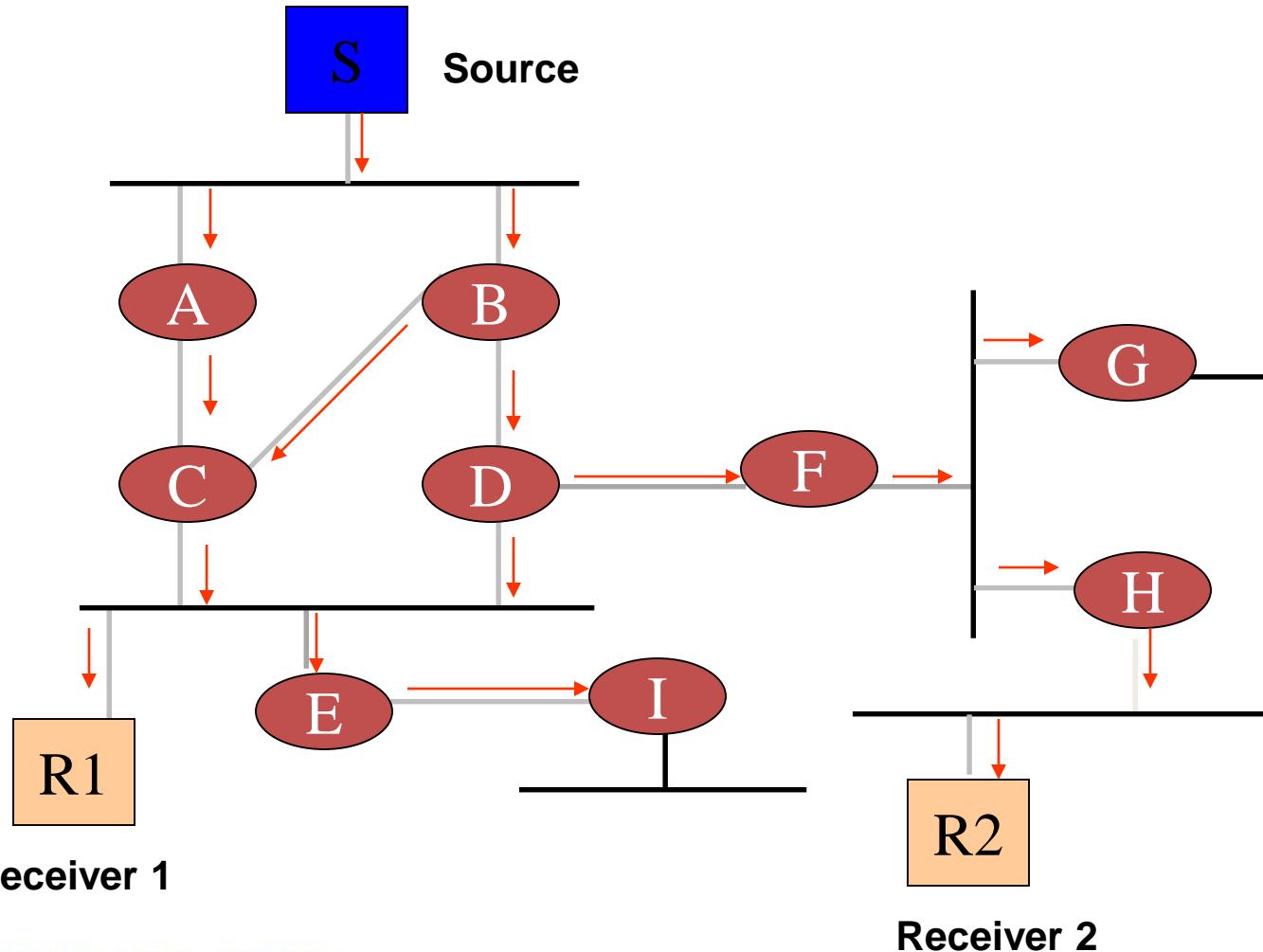
- Reverse Path Multicast



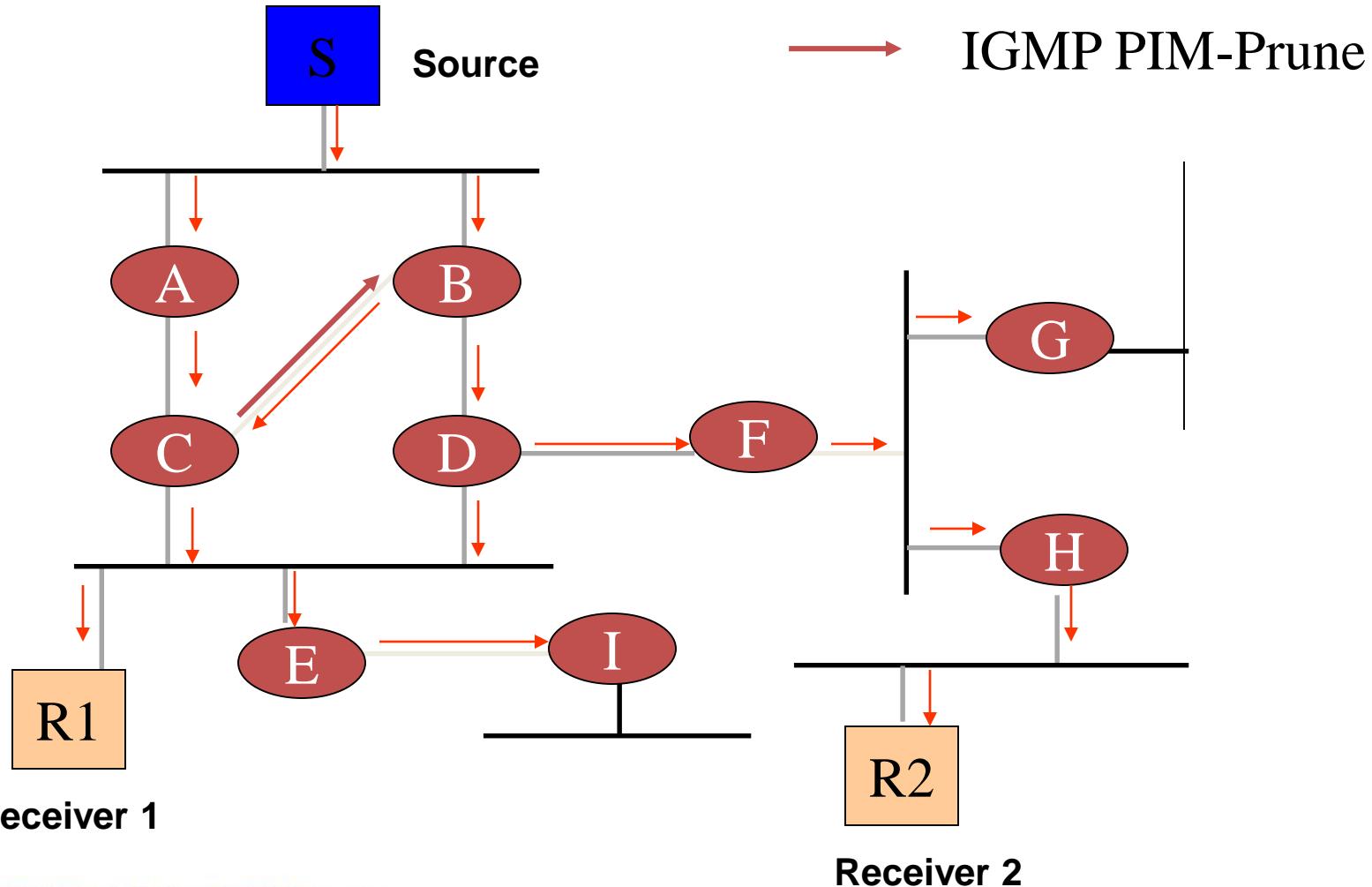
PIM – Dense Mode (DM)

- When it is likely that many routers are involved in multicast routing
- Source tree created on demand based on RPF rule
- If the source goes inactive, the tree is torn down
- Branches that don't want data are pruned
- Grafts are used to join existing source tree

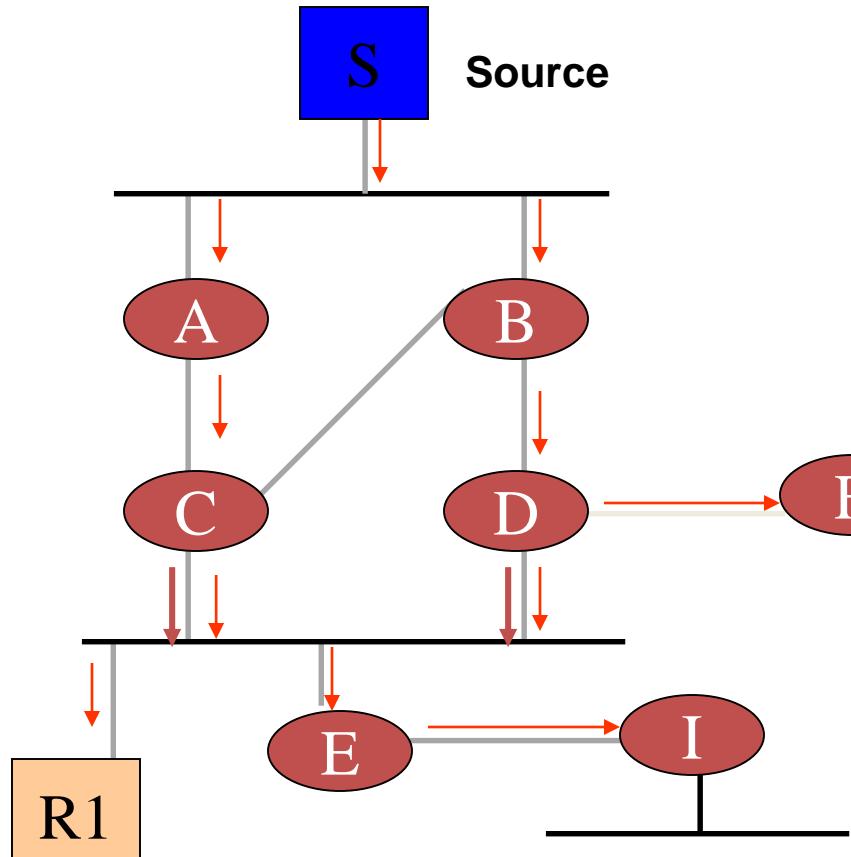
PIM-DM - Initial flood of data



PIM-DM - Prune non-RPF P2P link



PIM-DM

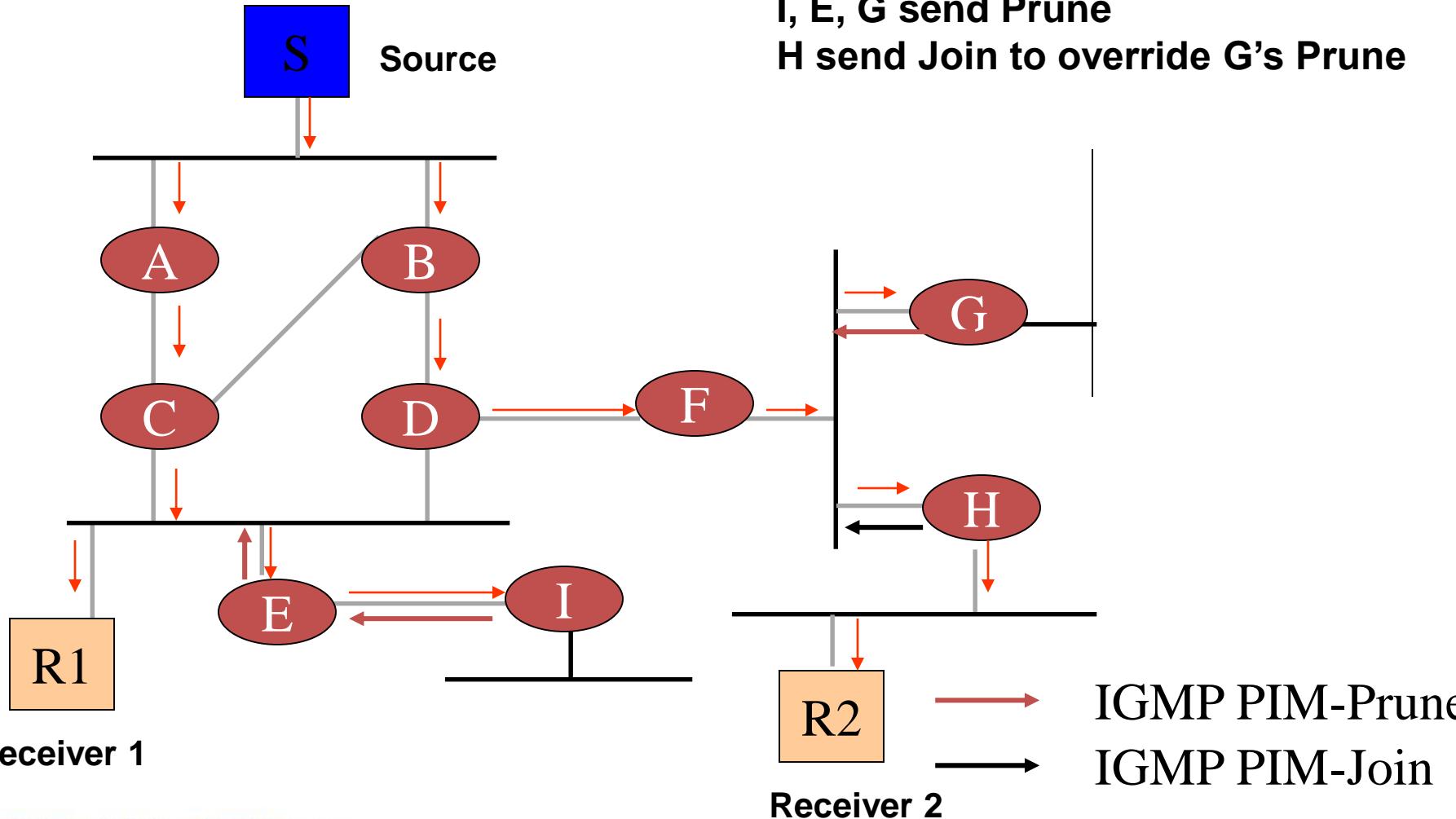


C and D Assert to Determine Forwarder for the LAN, C Wins

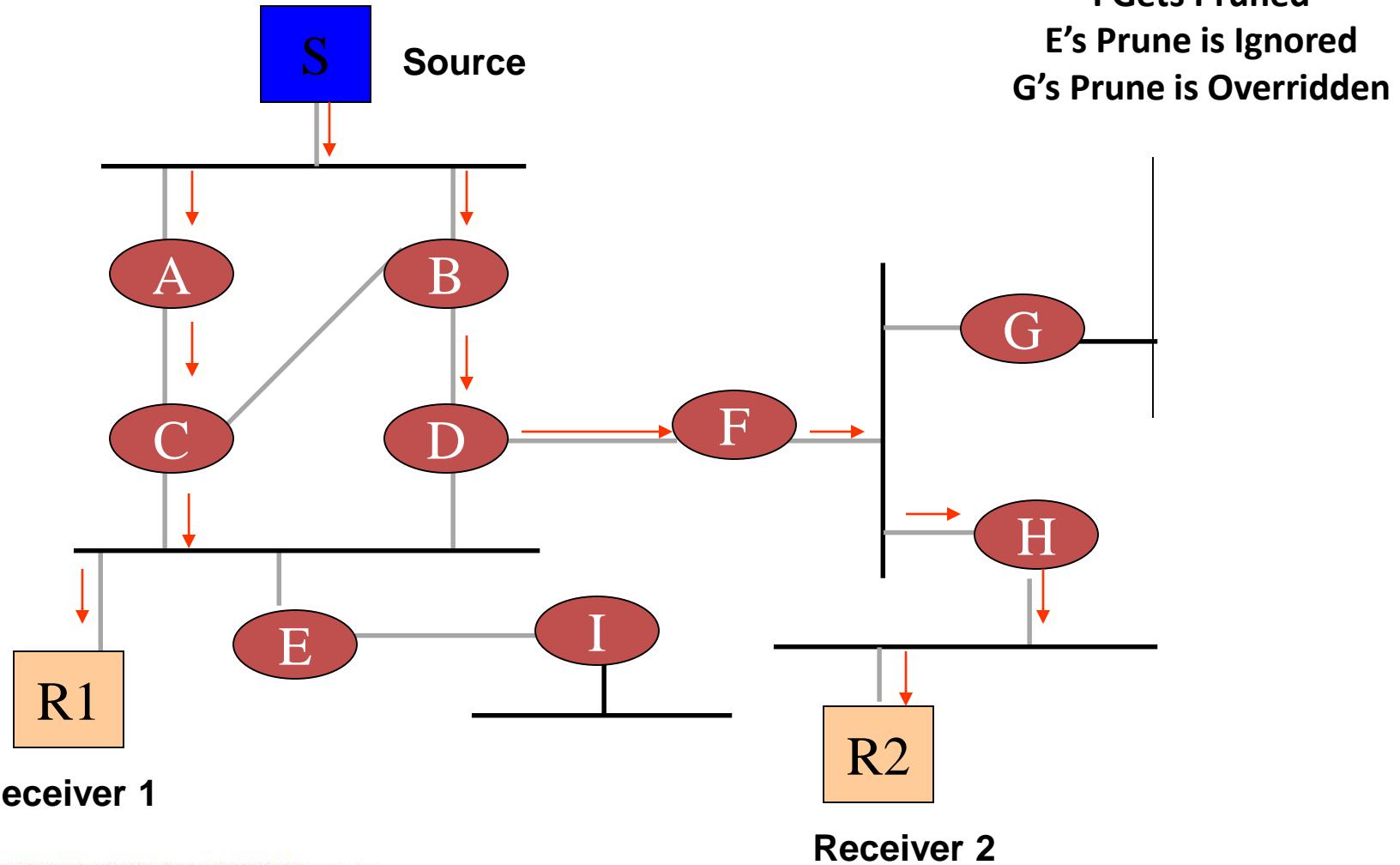
IGMP PIM-Assert
with its own IP address



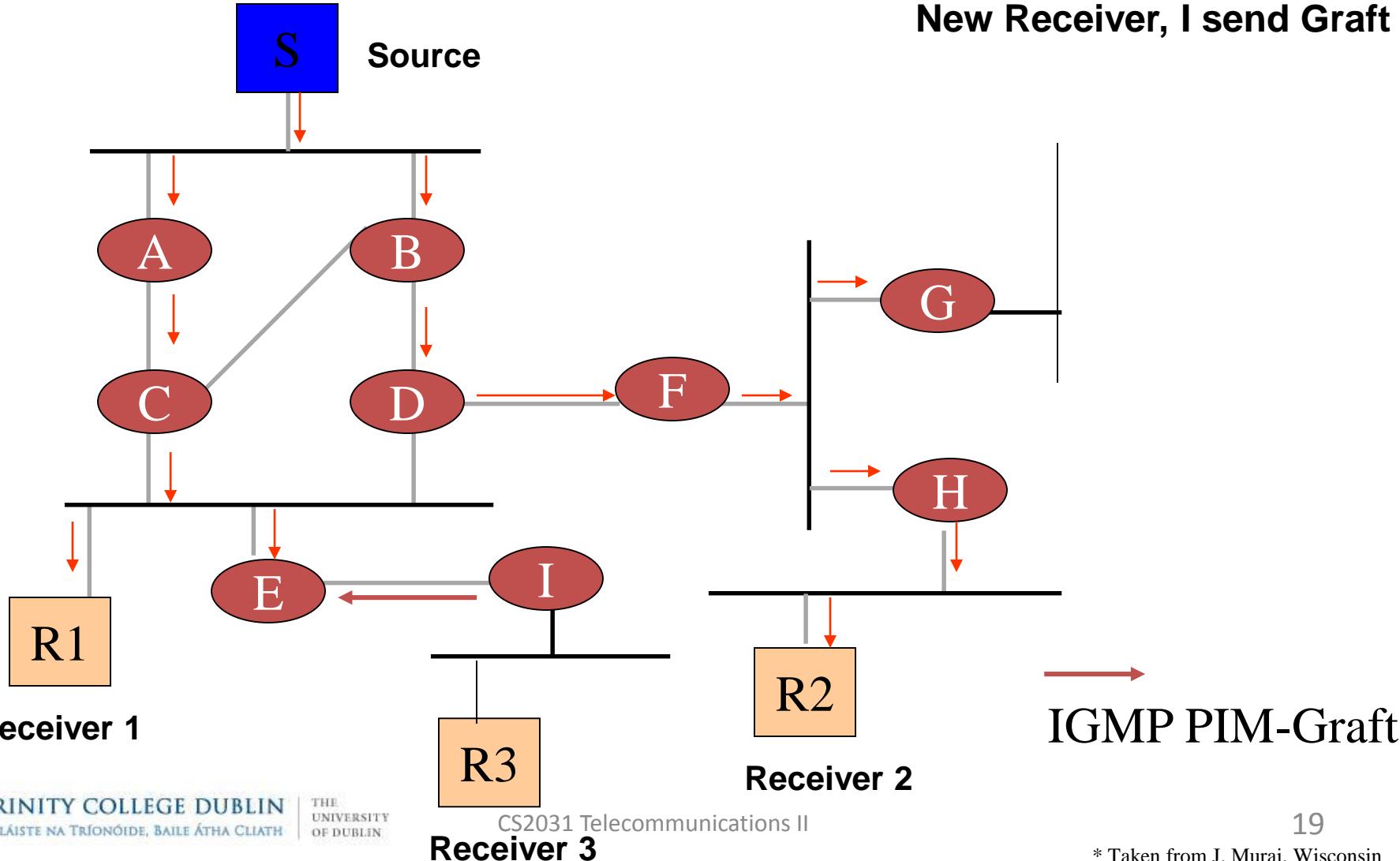
PIM-DM



PIM-DM

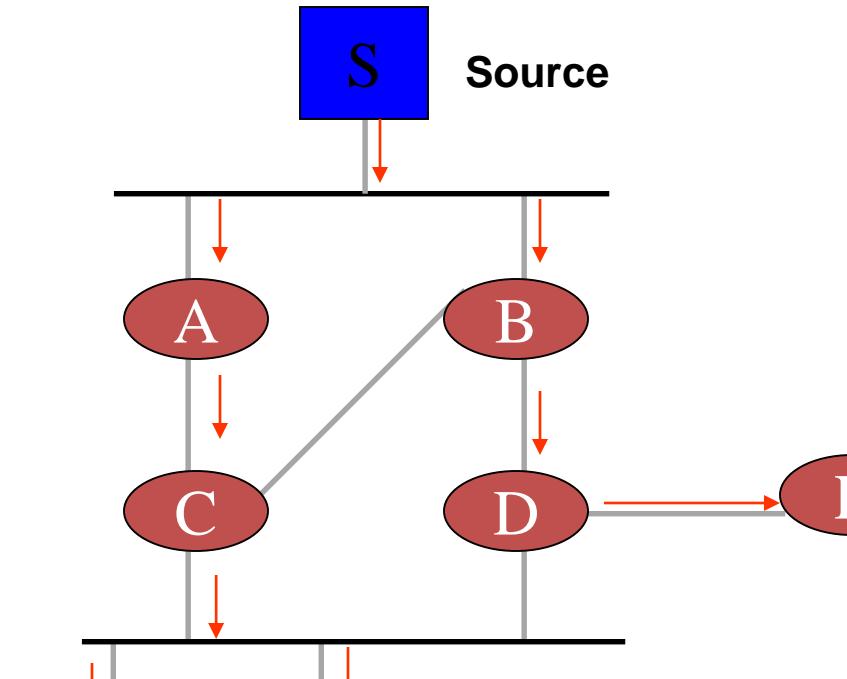


PIM-DM



PIM-DM

New branch



Receiver 1

Receiver 2

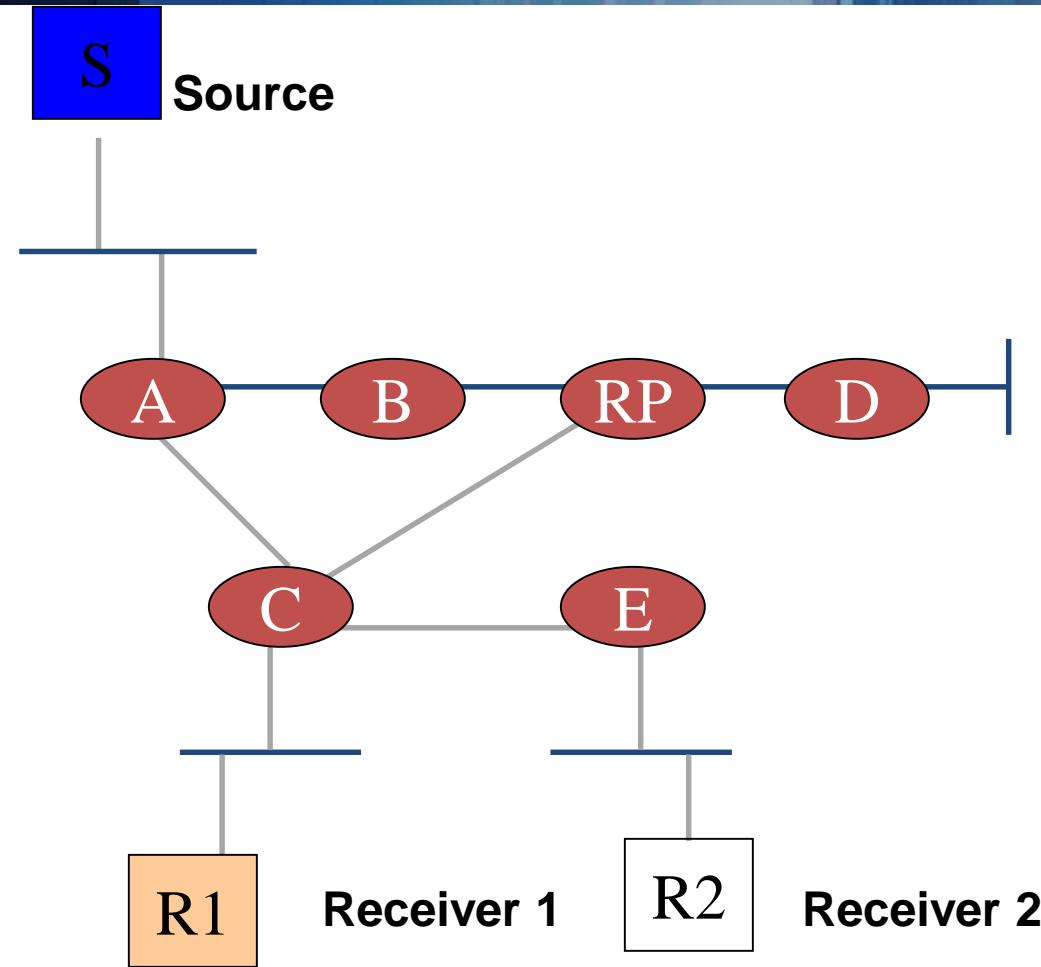


IGMP PIM-Graft

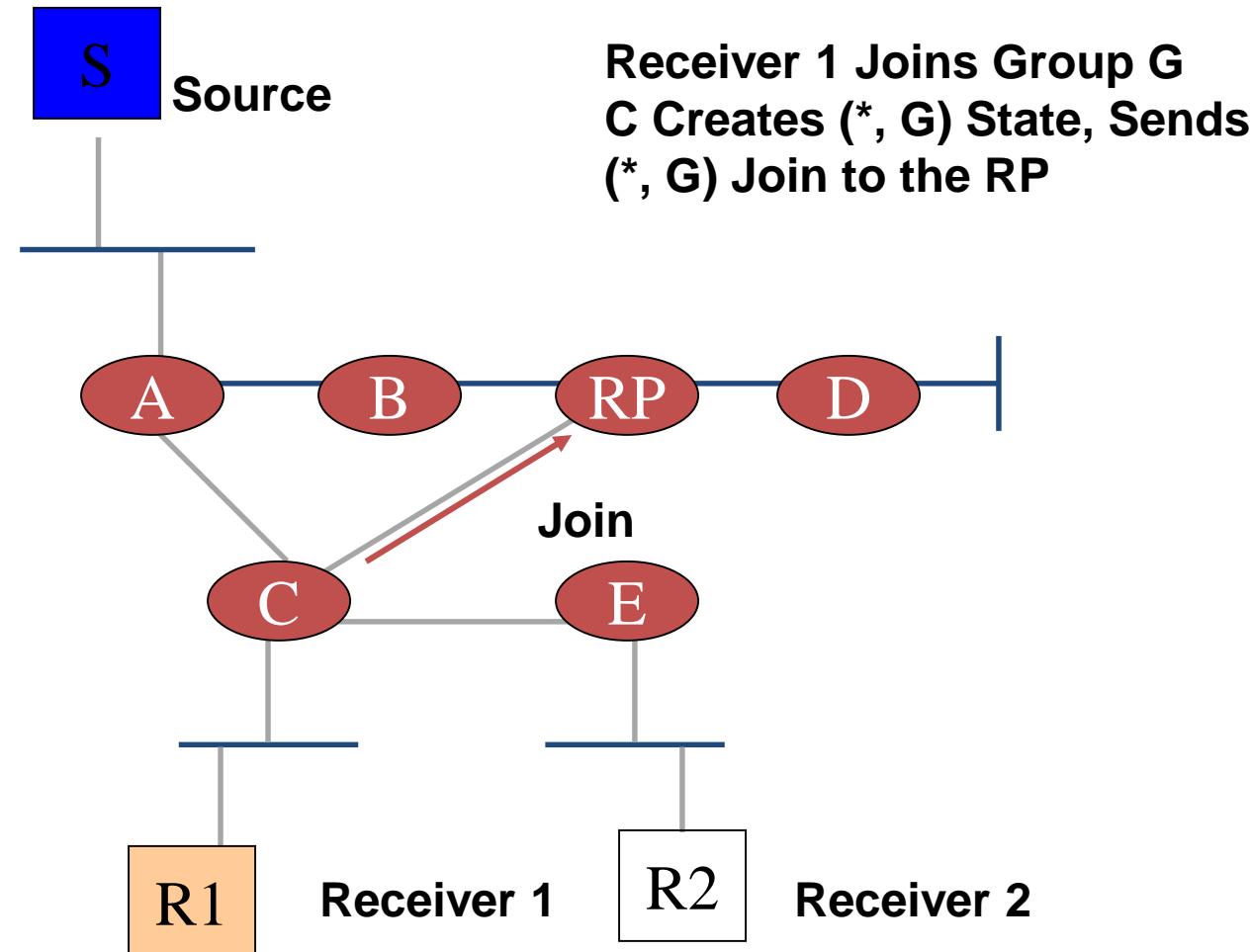
PIM – Sparse Mode (SM)

- When it is likely that many routers are involved in multicast routing
- One Rendez-Vous Point (RP) per group
- Explicit Join Model
 - Receivers send Join towards the RP
 - Sender Register with RP
 - Last hop routers can join source tree if the data rate warrants by sending joins to the source
- Dedicated “All-PIM-Routers” (224.0.0.13, ff02::d) multicast group

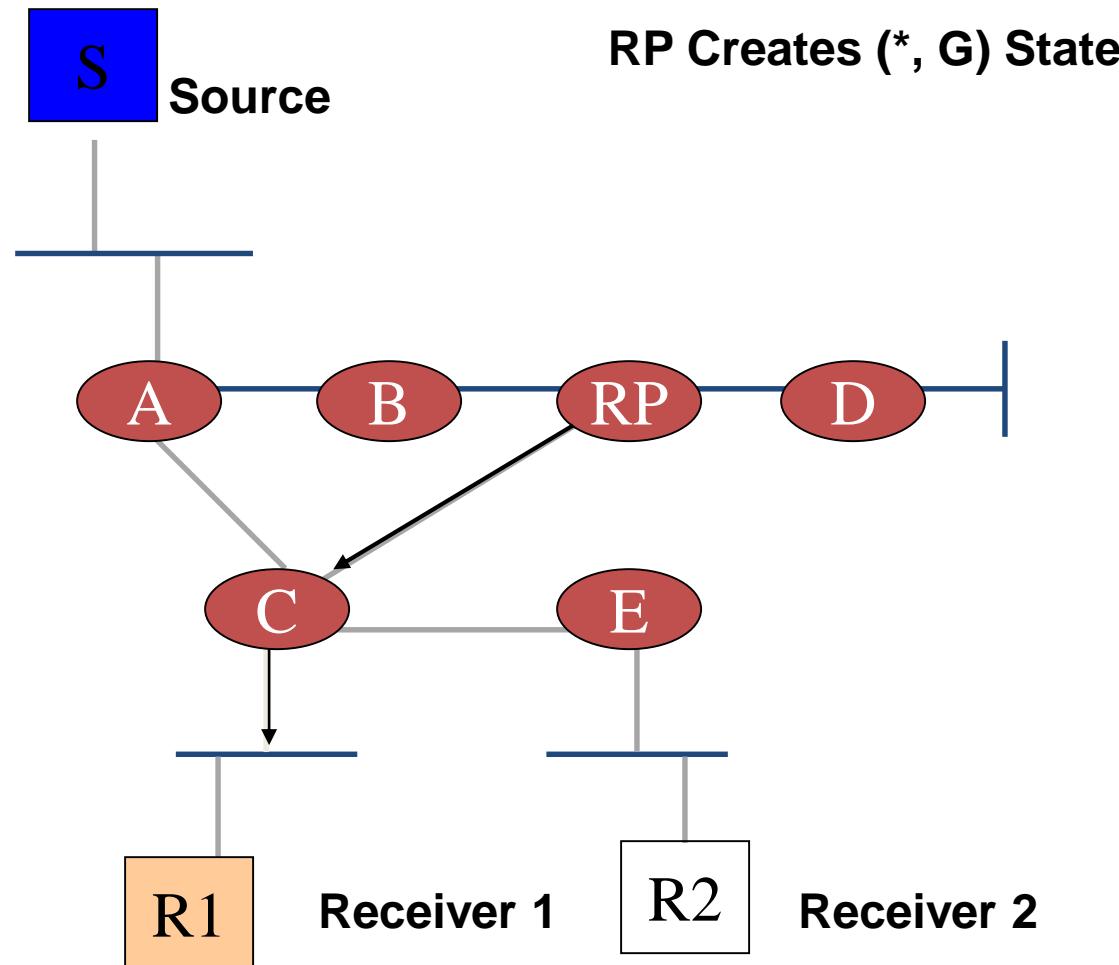
PIM-SM



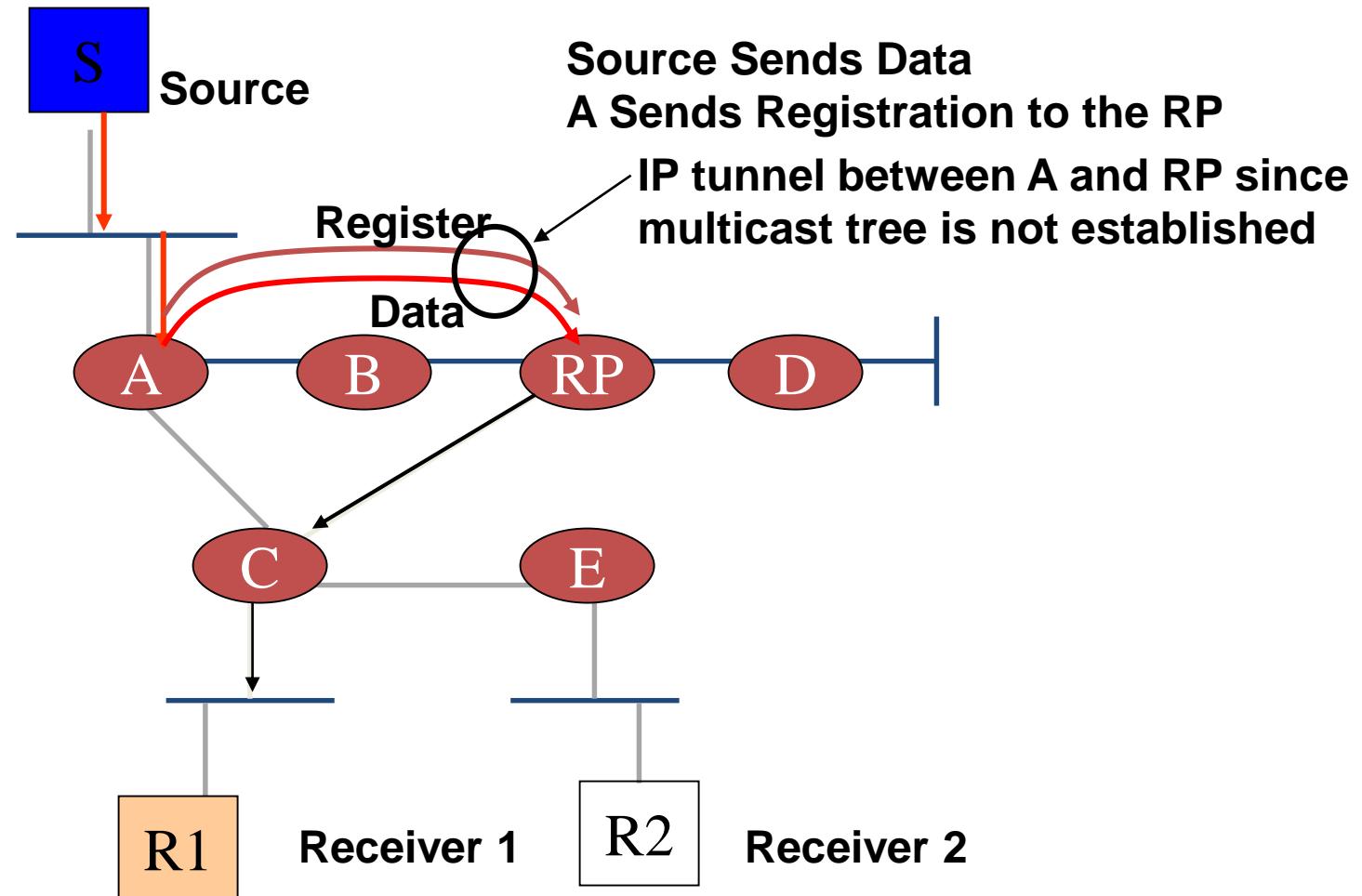
PIM-SM



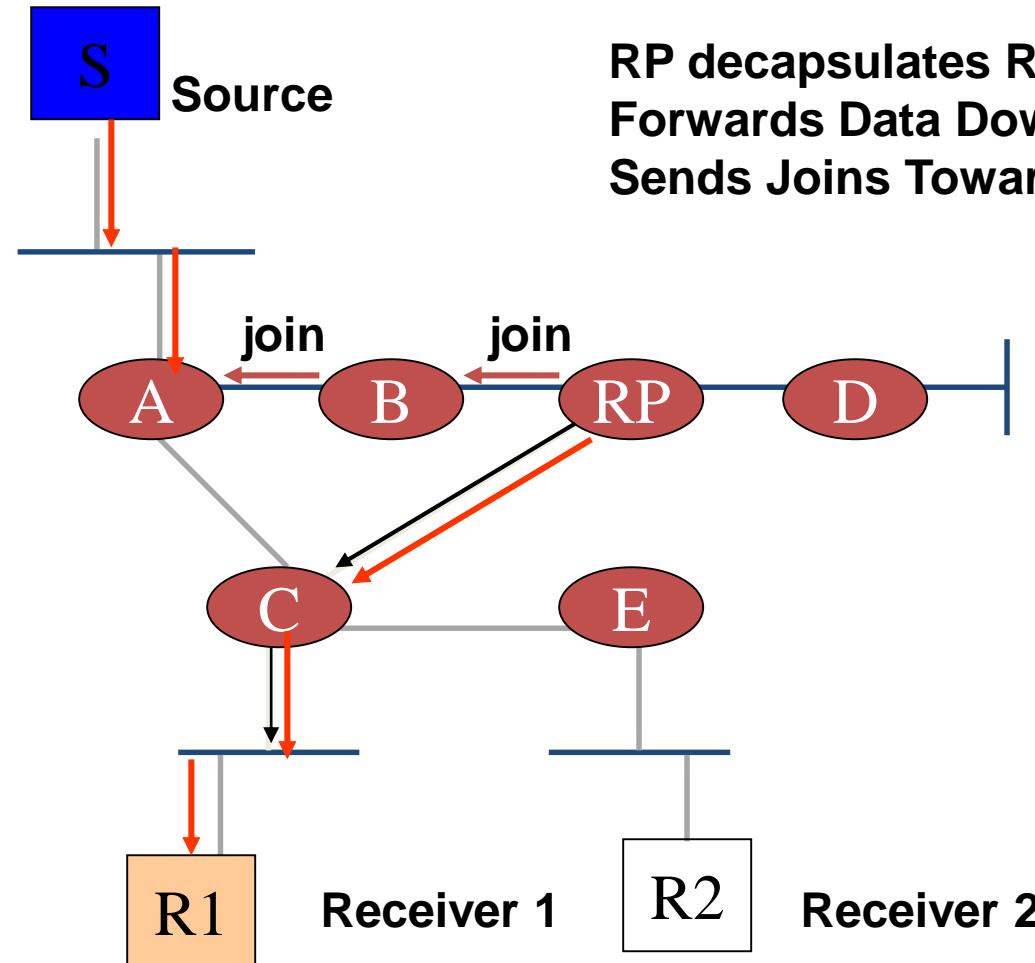
PIM-SM



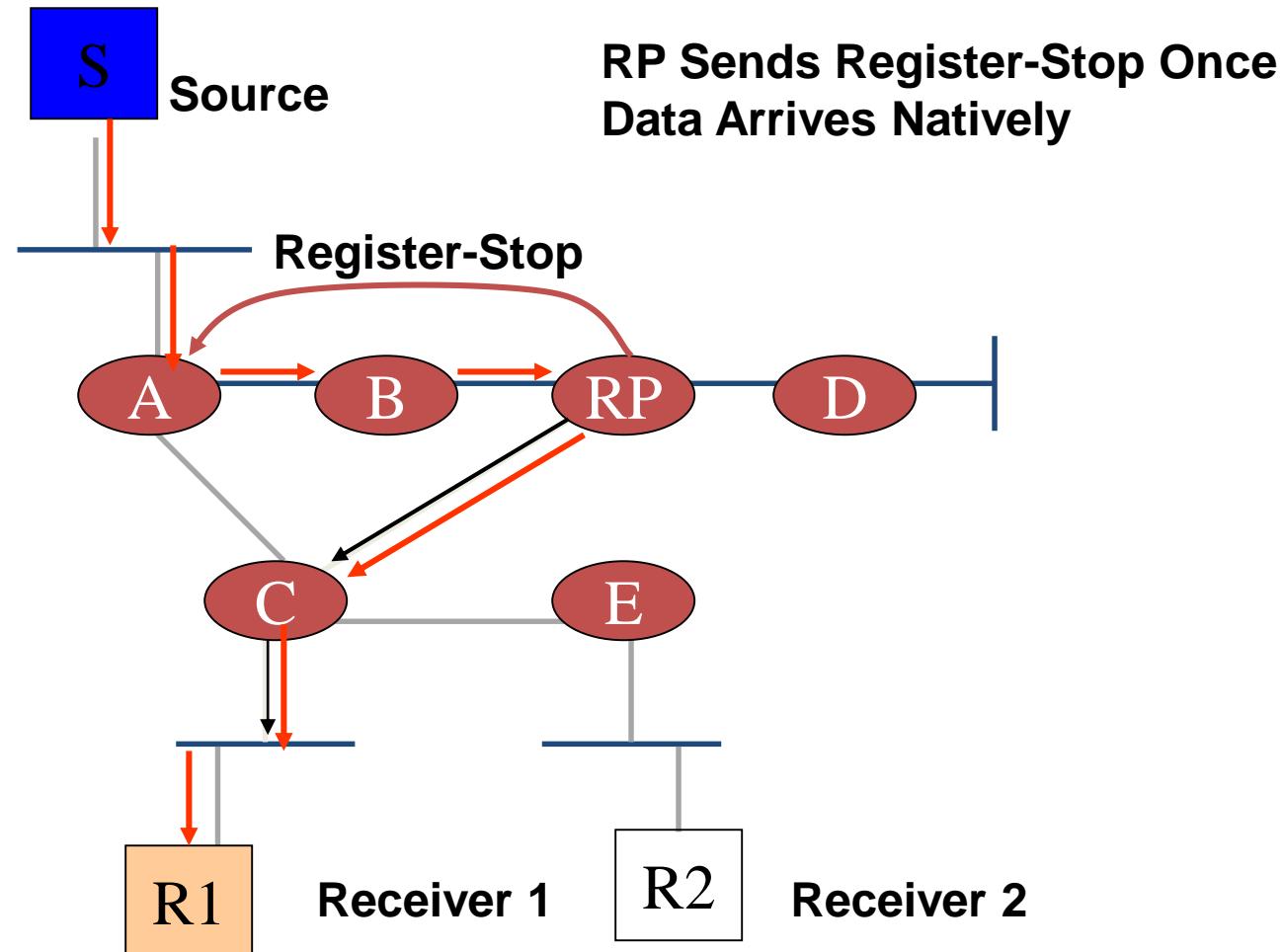
PIM-SM



PIM-SM

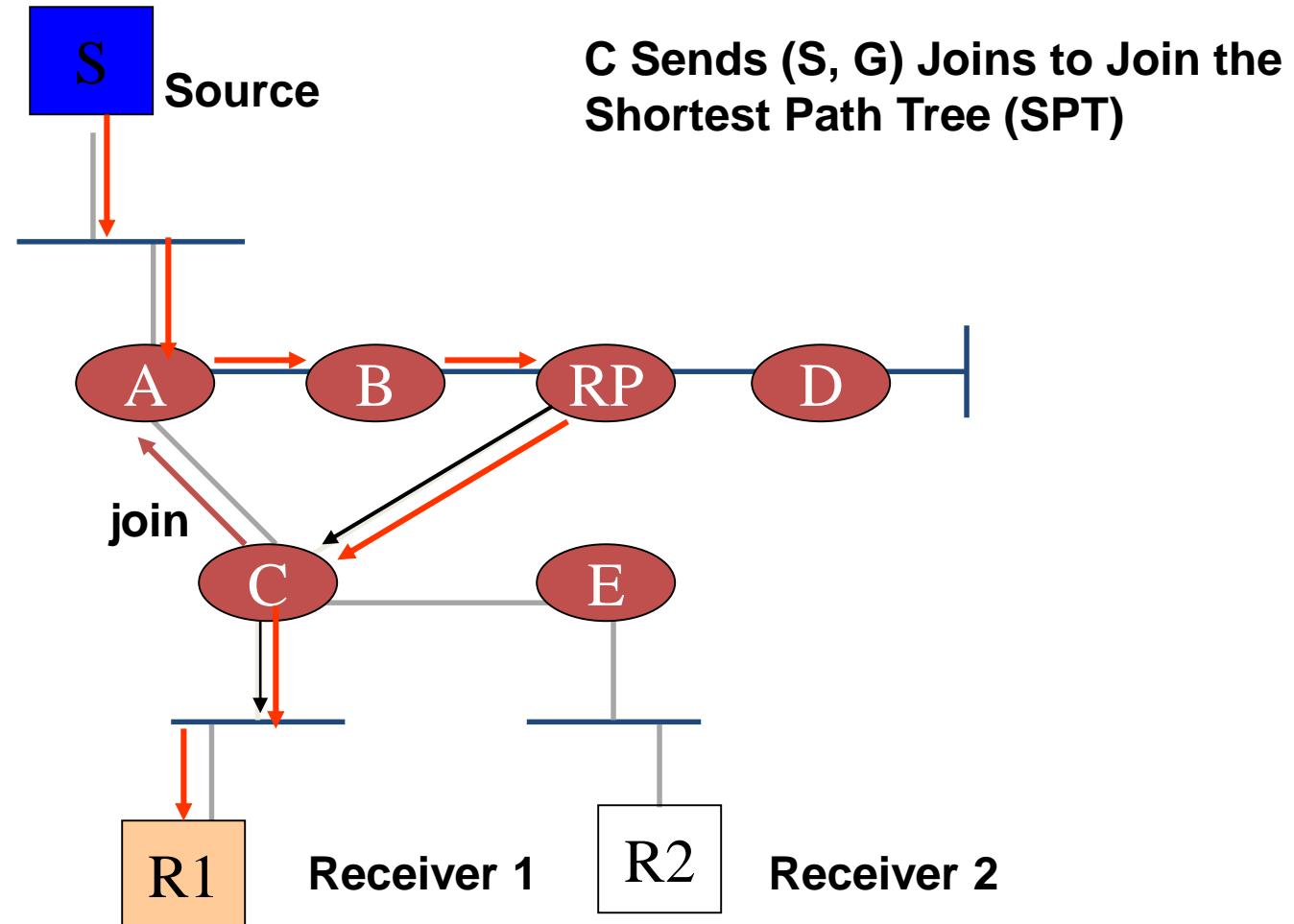


PIM-SM

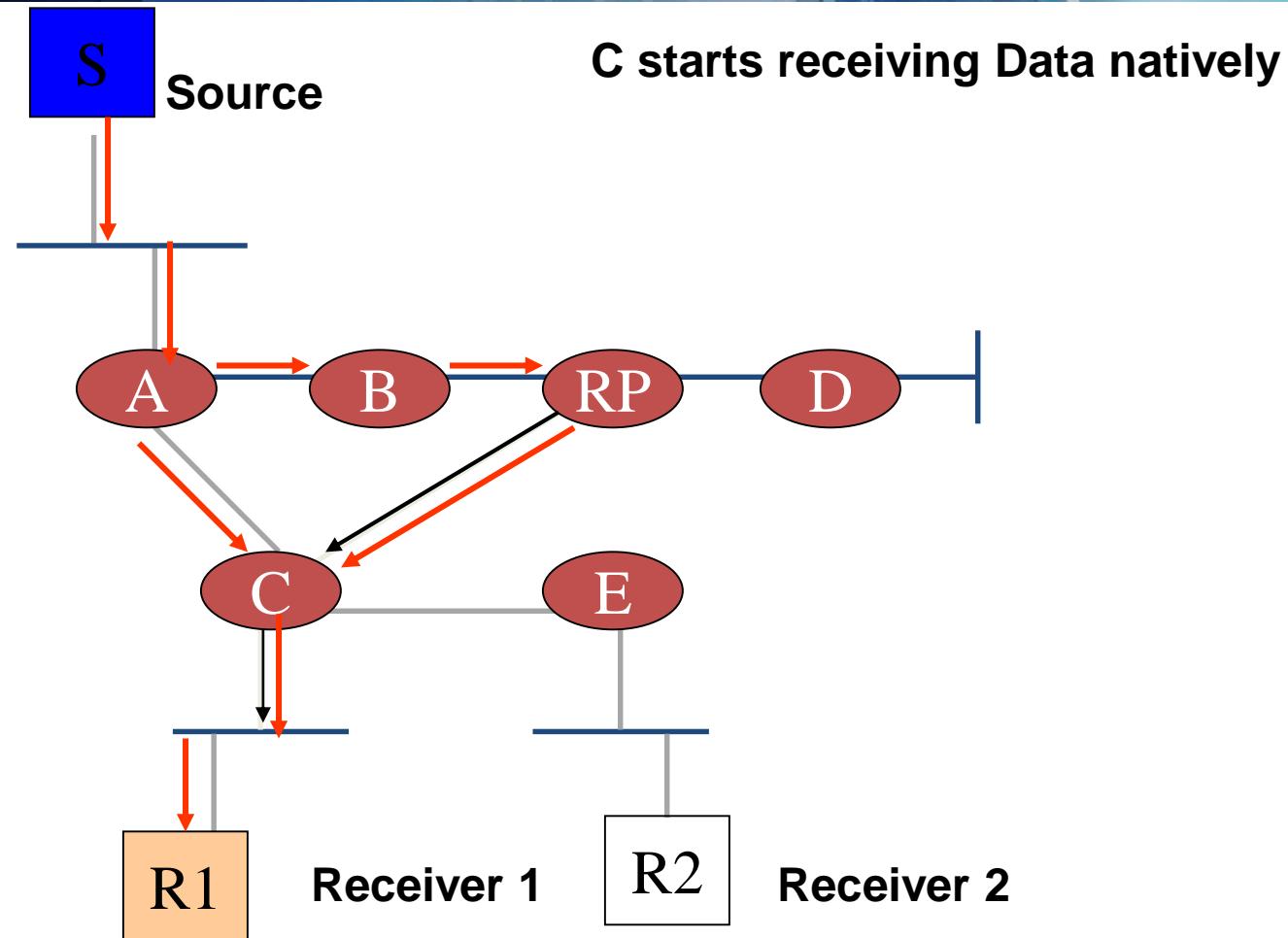


SPT Switchover

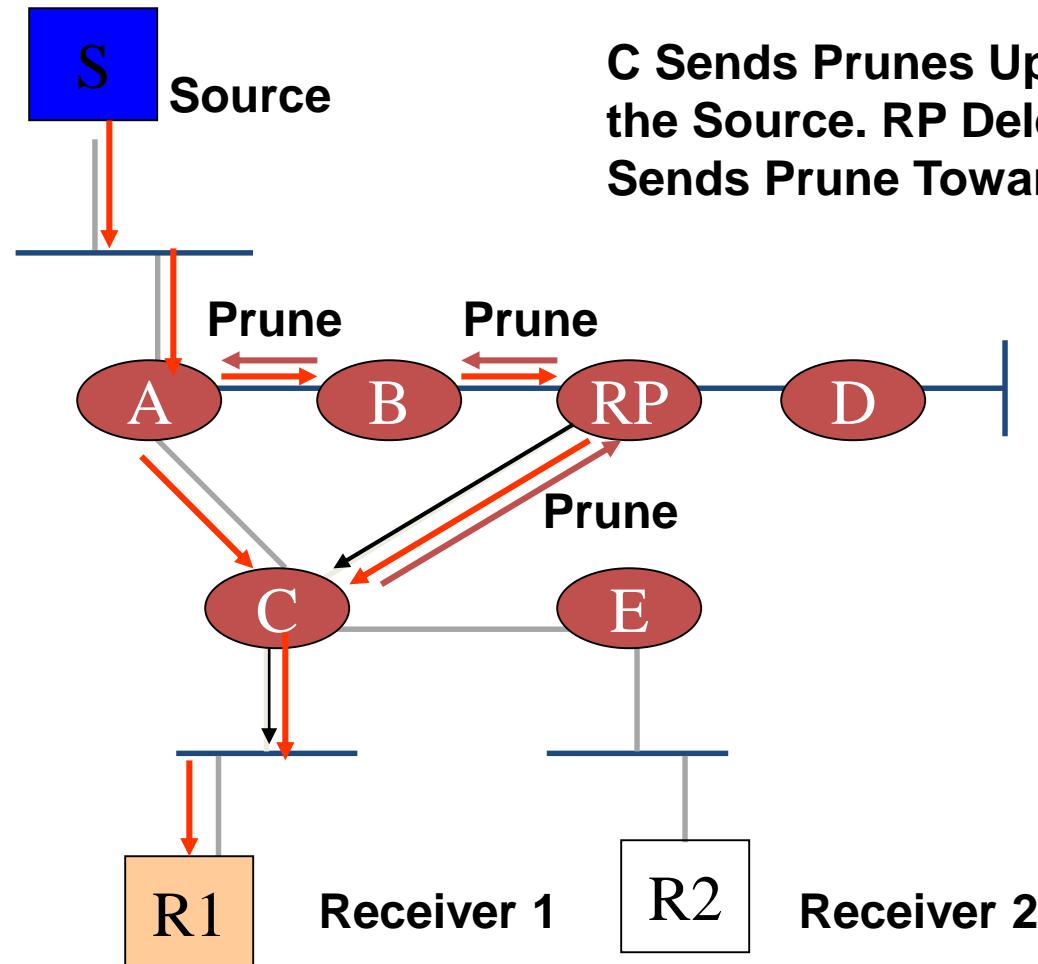
PIM-SM



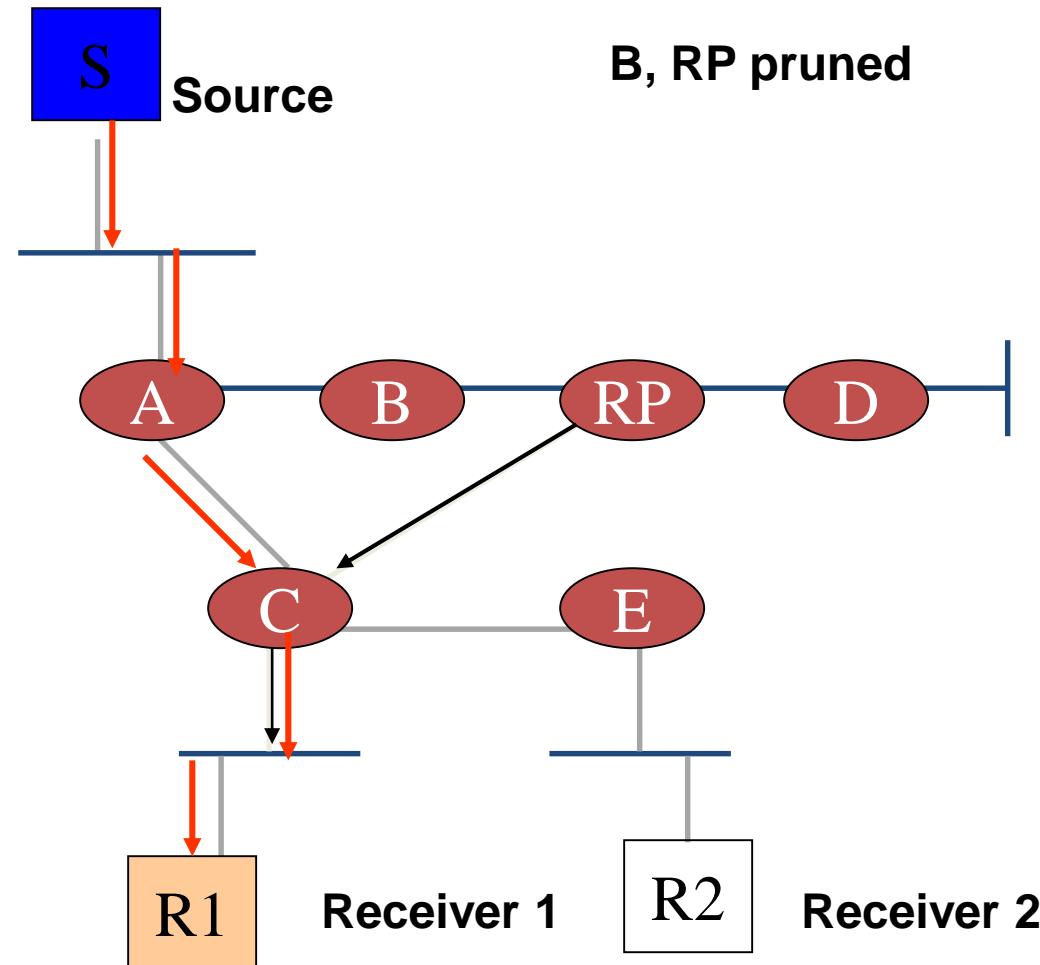
PIM-SM



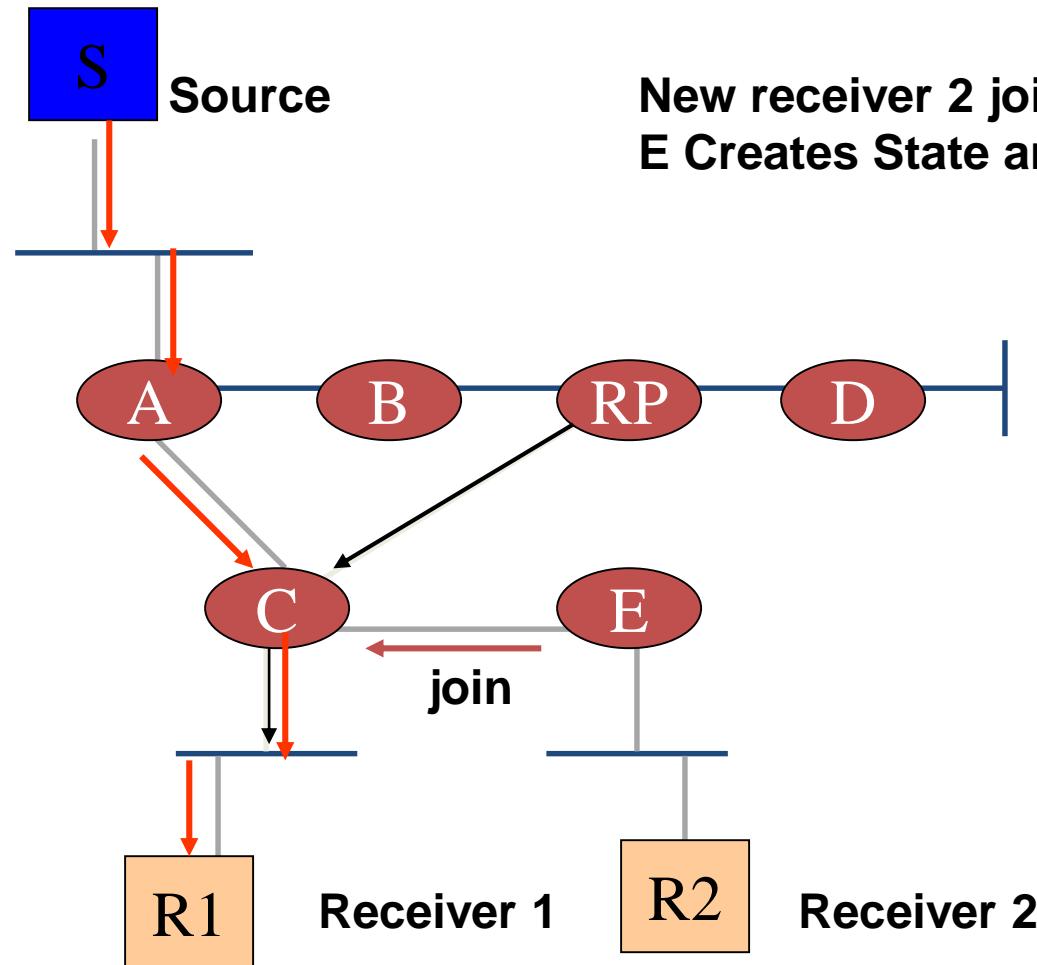
PIM-SM



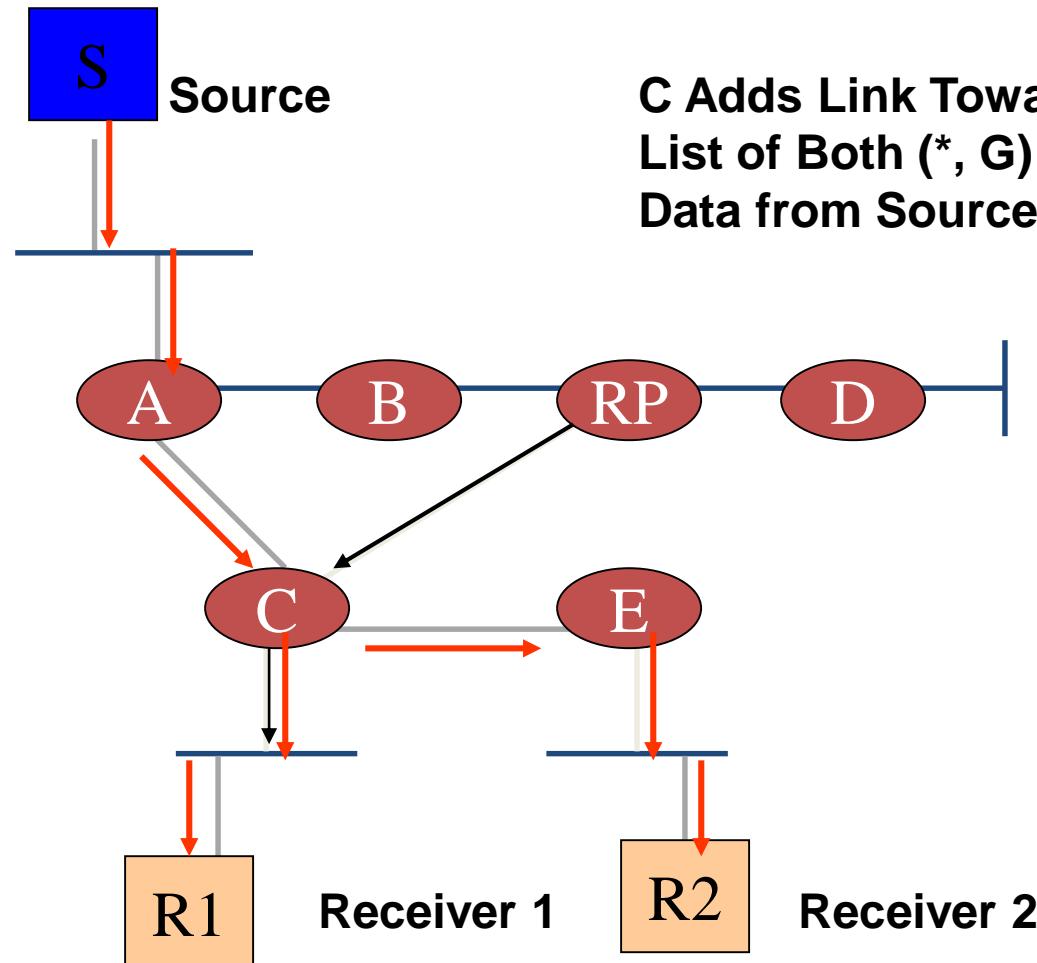
PIM-SM



PIM-SM



PIM-SM



Summary: Multicast Routing

- Internet Group Management Protocol (IGMP)
 - Join&leave messages from hosts to routers
- Most protocols based on source trees
 - Reverse-Path Forwarding/Broadcast
 - Prune – remove subtree from tree
 - Graft – join subtree to tree
- Protocol Independent Multicast (PIM)
 - Dense Mode (DM)
 - Sparse Mode (SM)

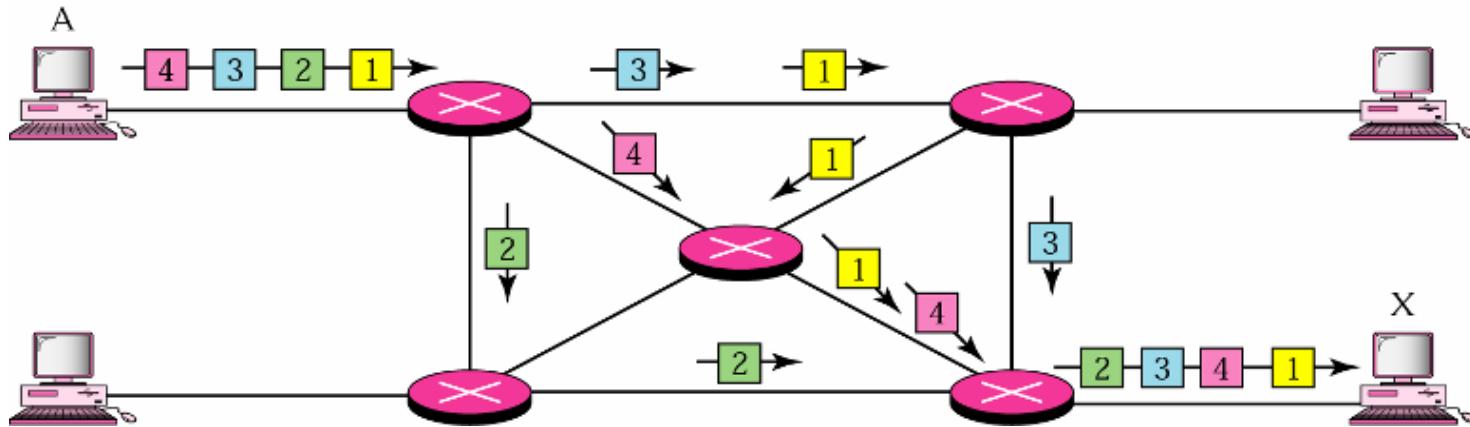


Overview

- Link Layer
- Network Layer
 - Addressing
 - Address Resolution (ARP)
 - Fragmentation
 - Intra-AS Routing
 - Distance Vector
 - Link State
 - Multicast Routing
 - IPv6
- Transport Layer
 - UDP
 - DNS

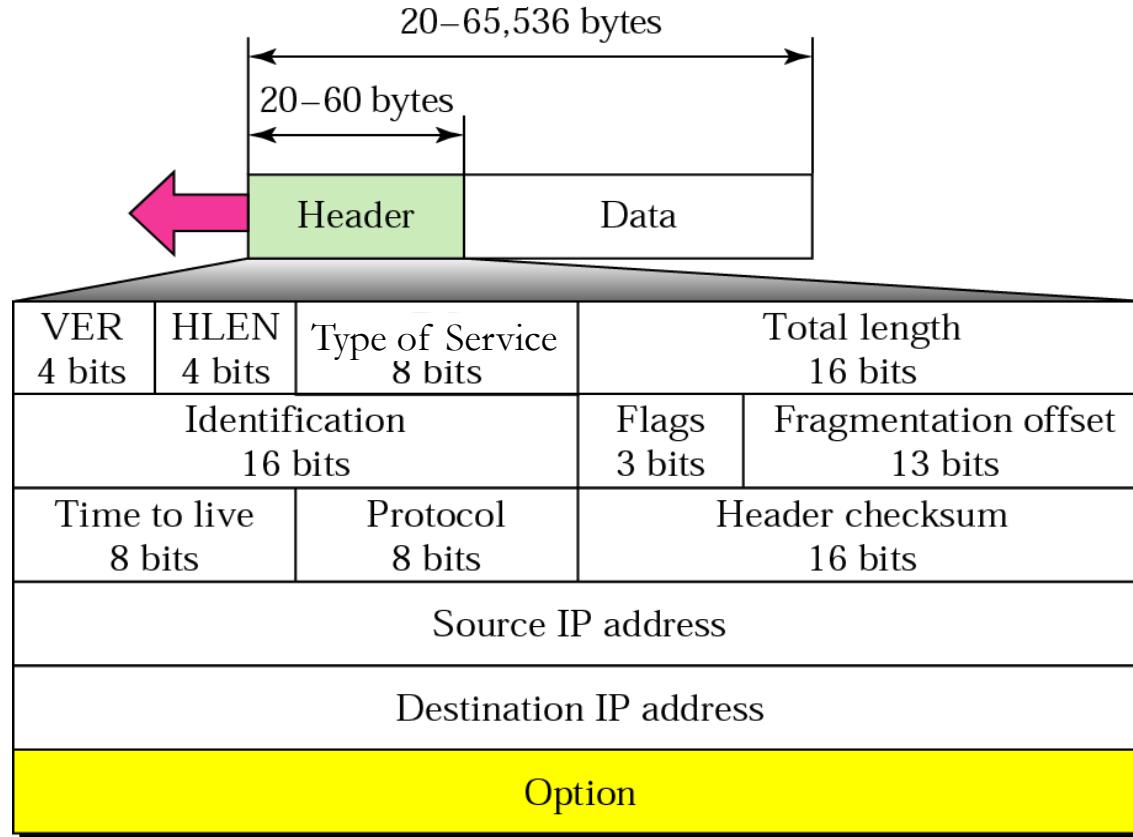


IP Service Model



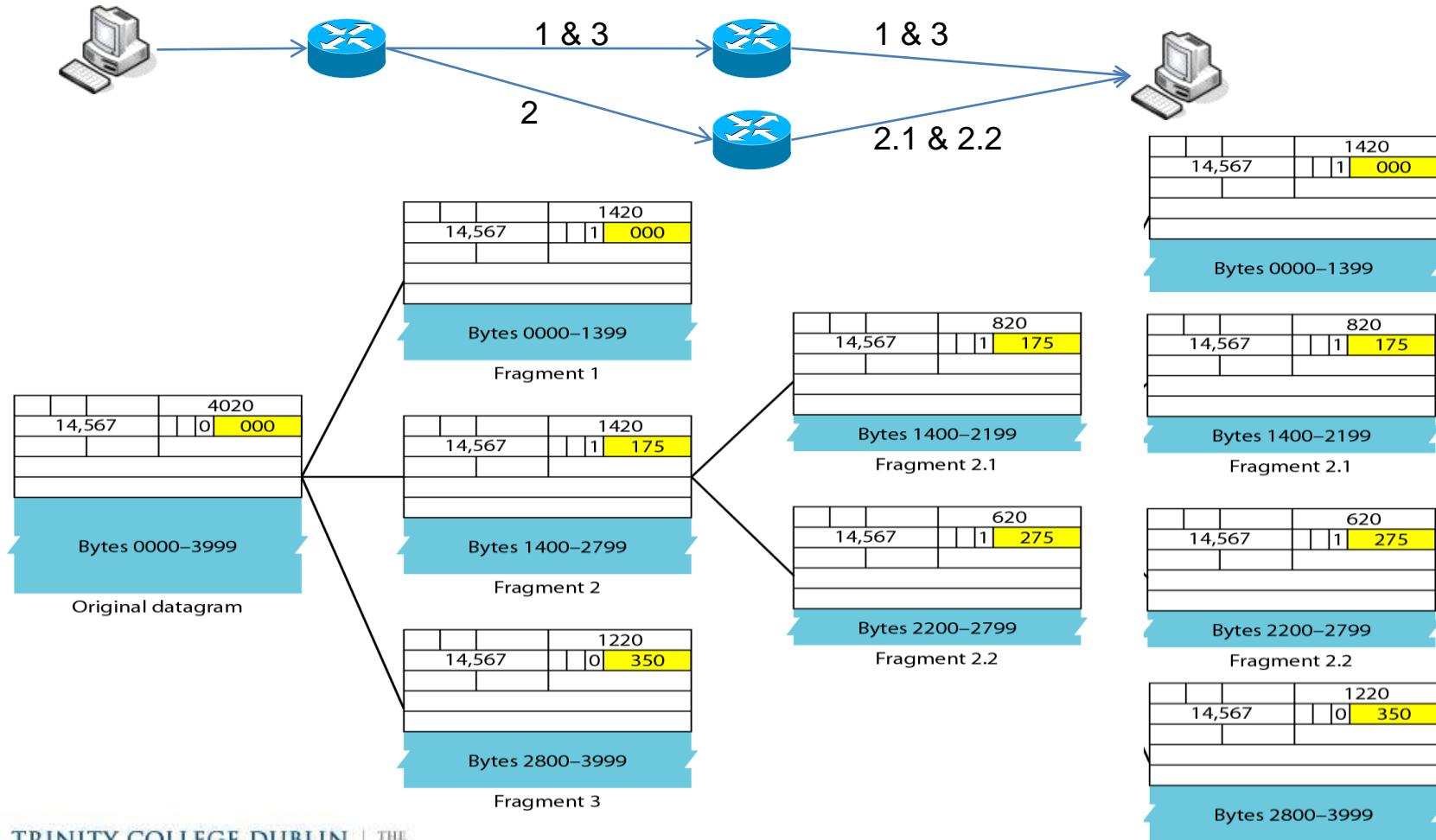
- Connection-less Communication
 - No state is kept about individual packets
- Order not guaranteed
- Best-effort delivery (unreliable service)
 - Packets may be lost
 - Packets may be delivered out of order
 - Duplicate copies of a packet may be delivered
 - Packets can be delayed for a long time

IP Datagram

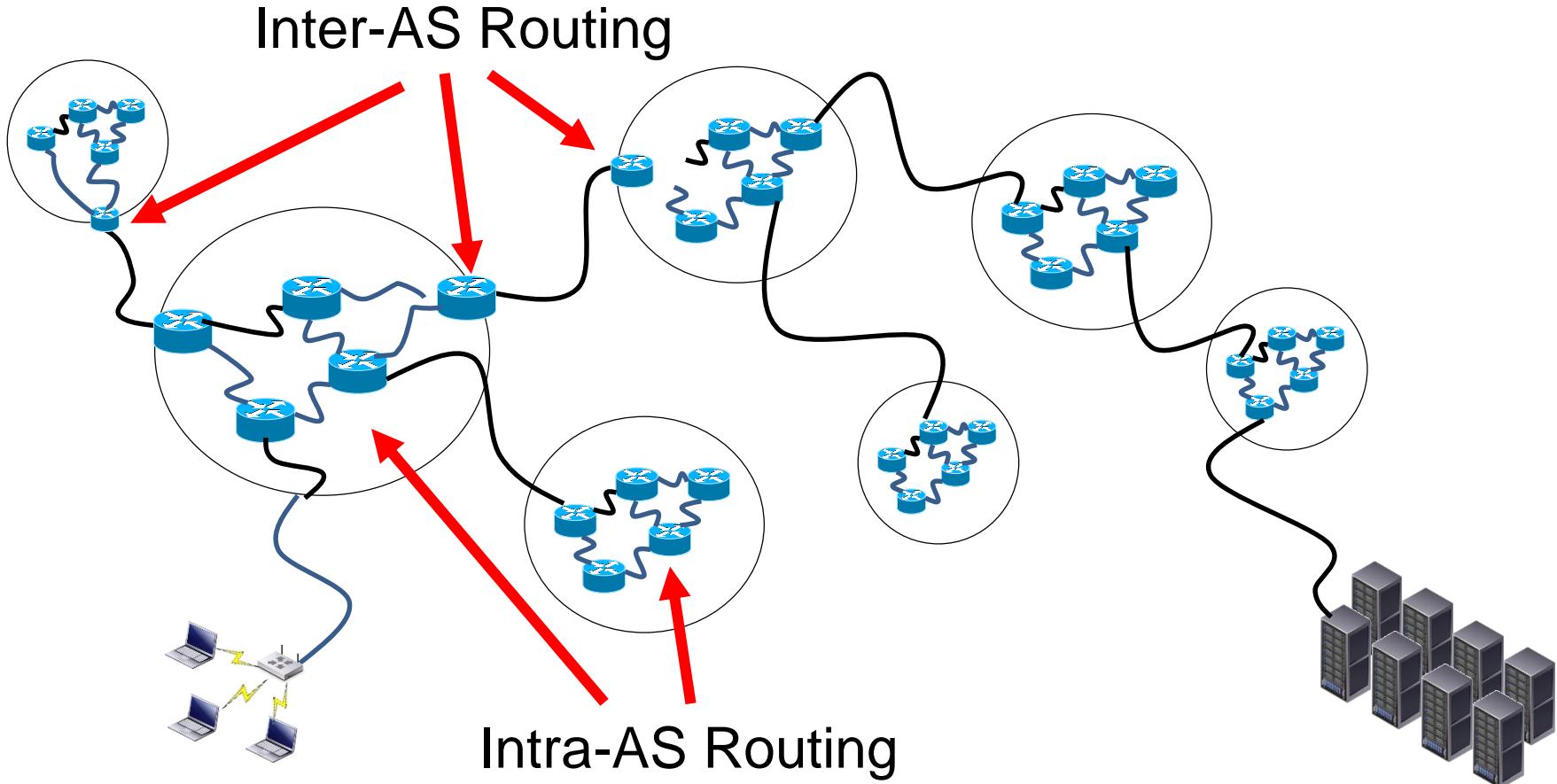


- The total length field defines the total length of the datagram including the header.

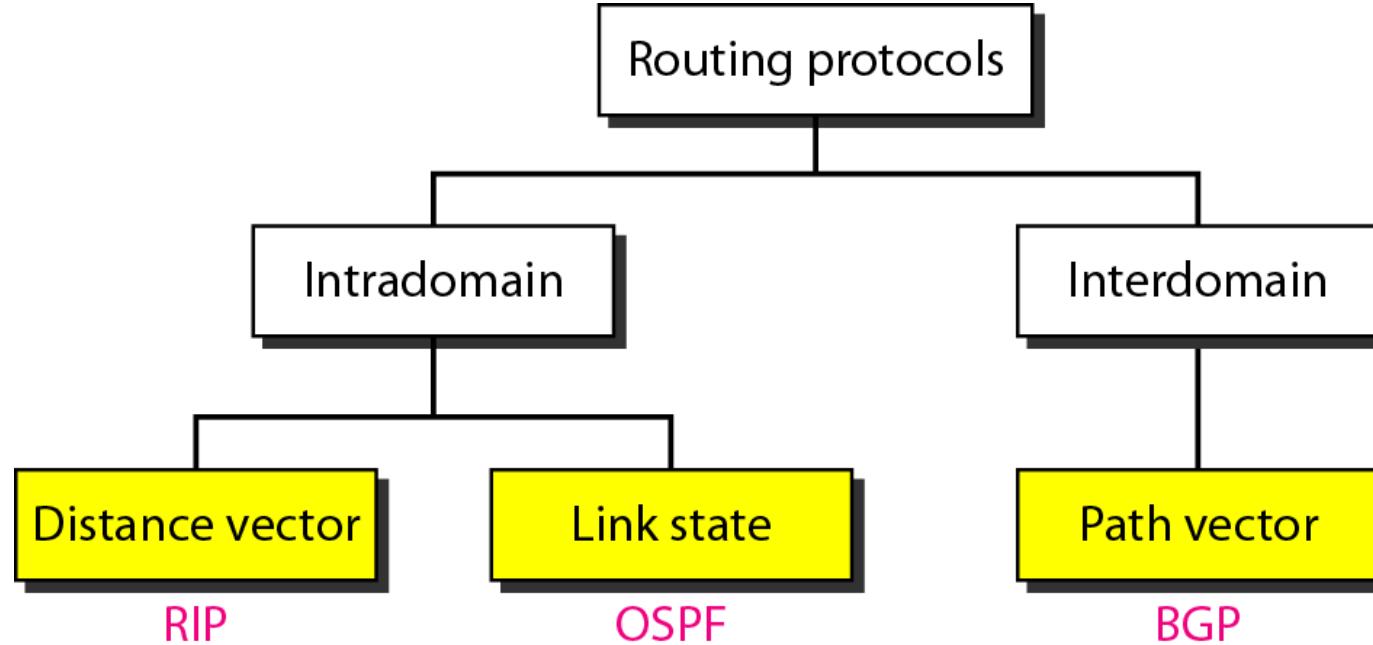
Fragmentation Example



Internet = Network of Networks

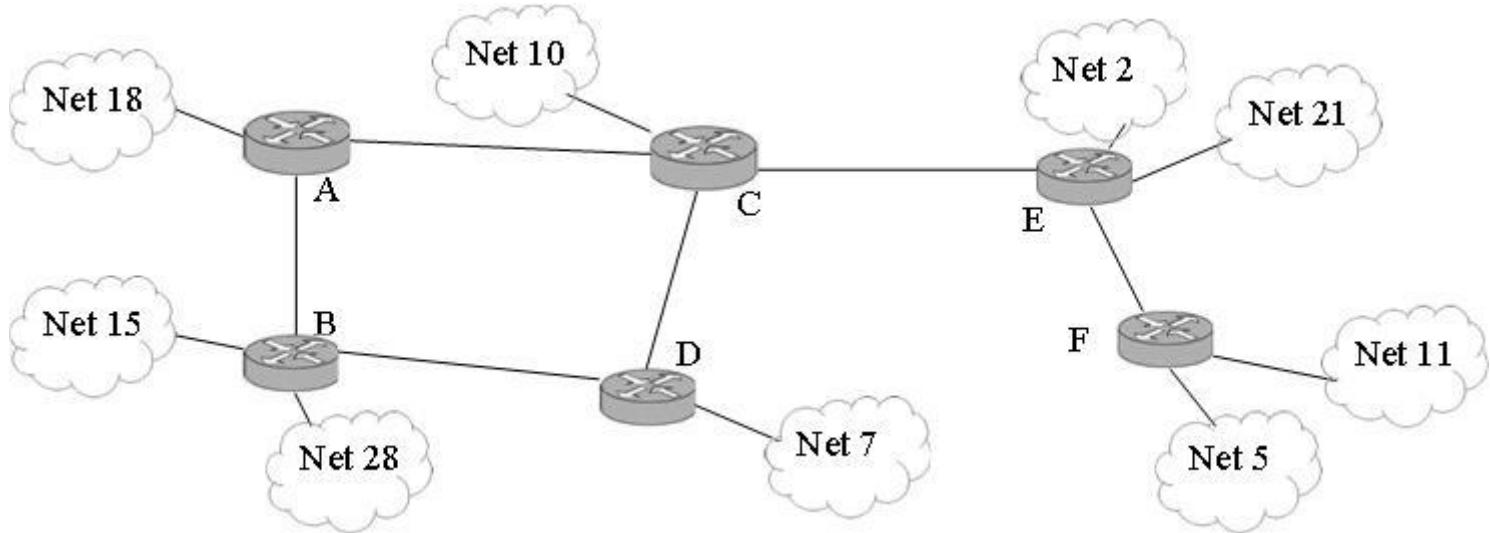


Routing Protocols



- Distance Vector routing
 - Routes propagate through exchange of routing tables
- Link State routing
 - Establish view of topology & run algorithm e.g. Dijkstra's Shortest-Path

Distance Vector Routing



A	Net18	1	-

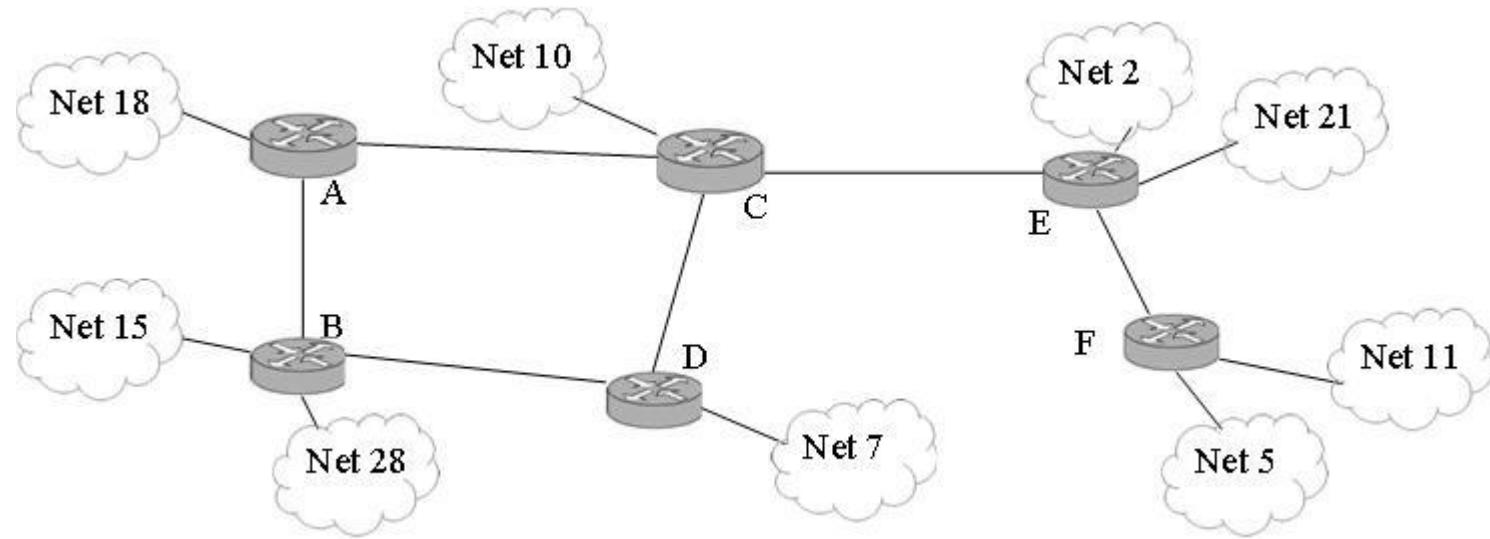
B	Net15	1	-

C	Net10	1	-

D	Net7	1	-

- Each router know the networks that are immediately connected to it

Distance Vector Routing



A		
Net18	1	-
Net15	2	B
Net28	2	B
Net10	2	C

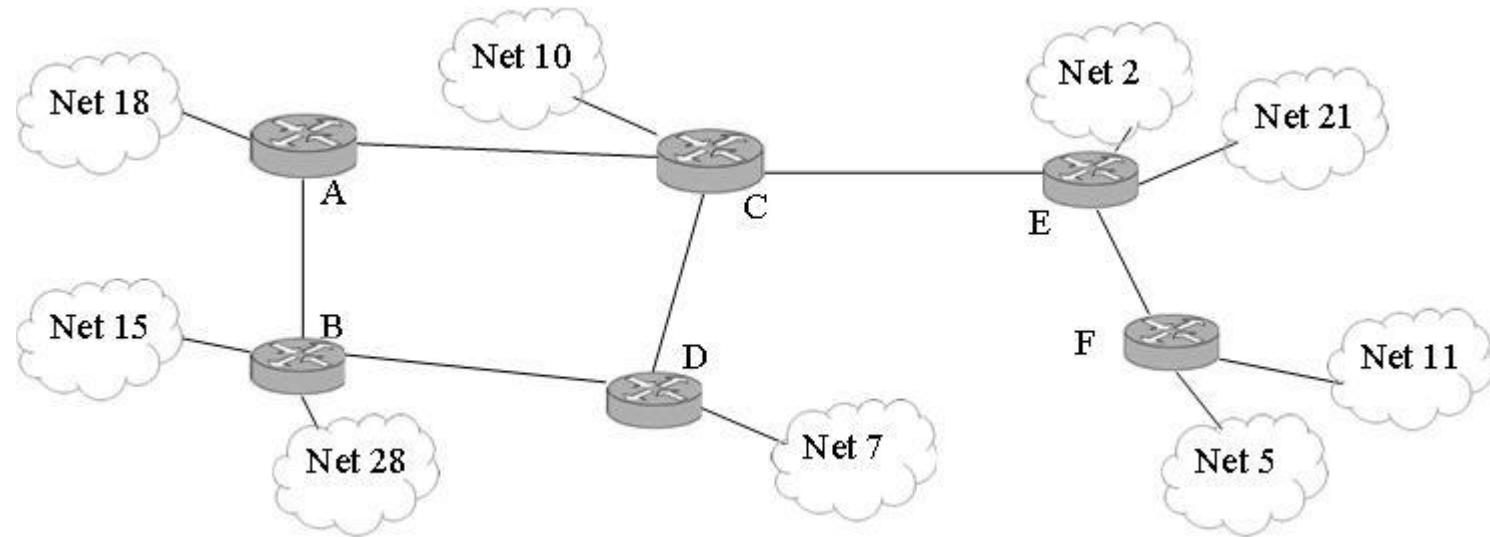
B		
Net15	1	-
Net28	1	-
Net18	2	A
Net10	2	C

C		
Net10	1	-
Net7	2	D
Net2	2	E
Net21	2	E

D		
Net7	1	-
Net15	2	B
Net28	2	B
Net10	2	C

- Learning routes from neighbours
 - e.g. Node A incorporates information from Node B

Distance Vector: Example IV

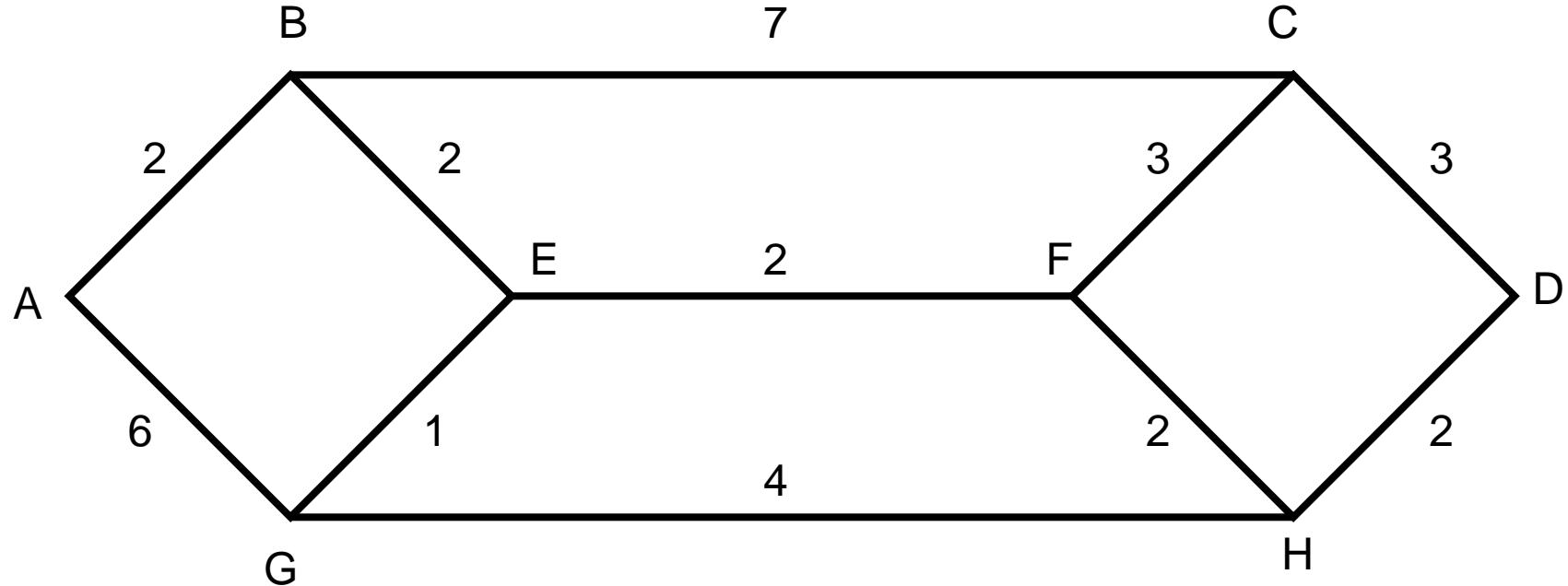


A		
Net18	1	-
Net15	2	B
Net28	2	B
Net10	2	C
Net7	3	B
...
Net11	5	C
Net5	5	C

- Time require to build complete routing tables for all nodes
 - this is known as convergence

Link State Routing

- View of complete topology
 - through broadcasts



Dijkstra's Algorithm for Router A

Routing Table

Tentative Nodes

A	0	-



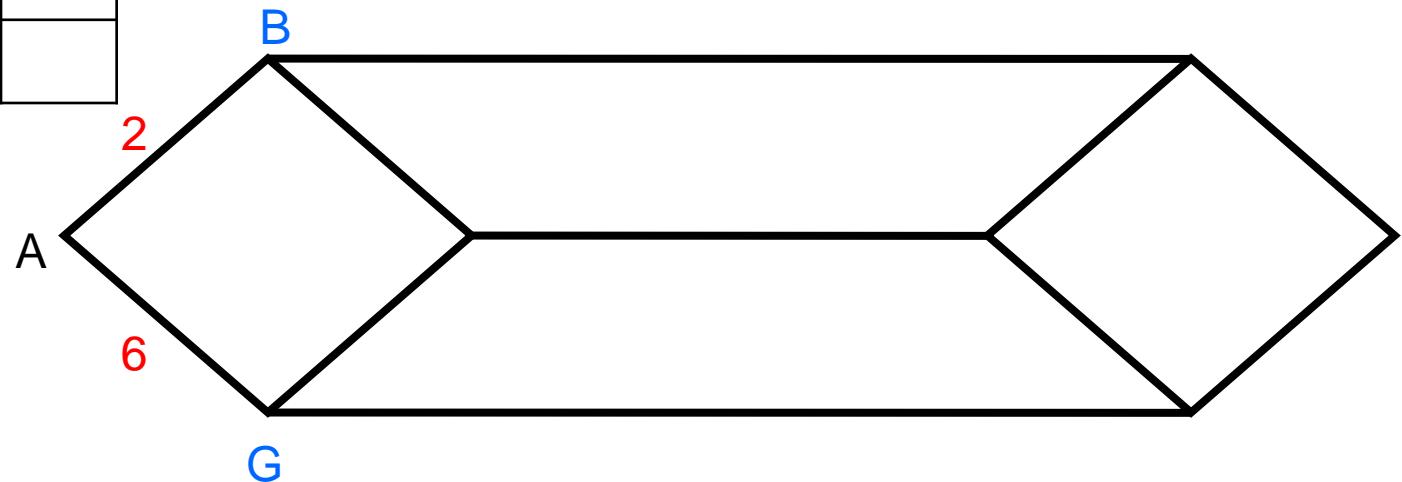
Dijkstra's Algorithm for Router A

Routing Table

A	0	-

Tentative Nodes

B	2	A
G	6	A



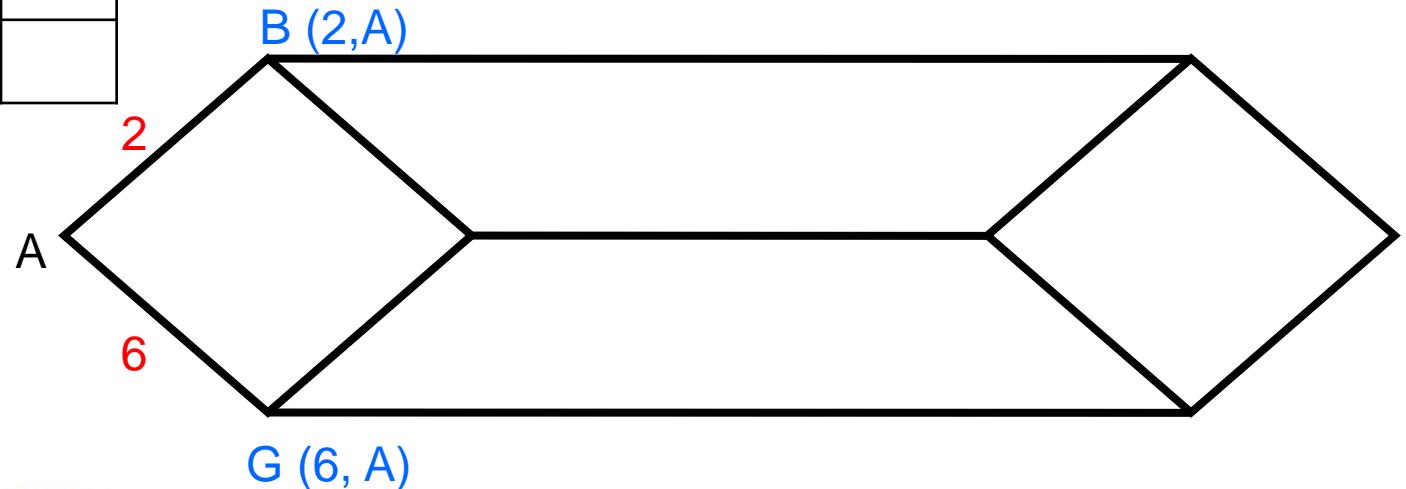
Dijkstra's Algorithm for Router A

Routing Table

A	0	-
B	2	A

Tentative Nodes

G	6	A
---	---	---



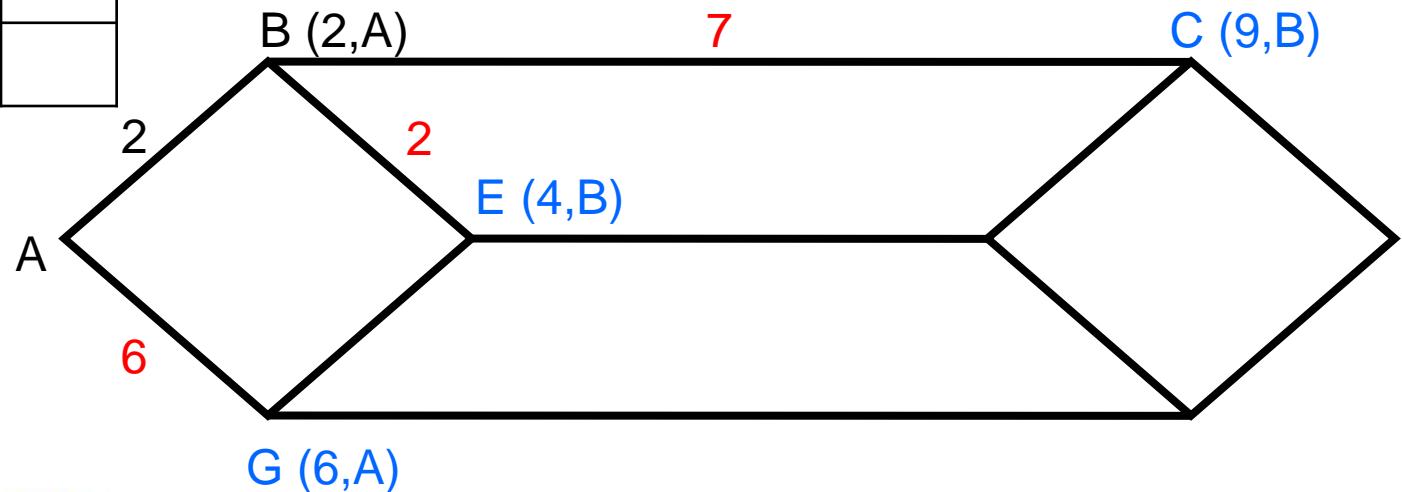
Example: Dijkstra's Algorithm

Routing Table

A	0	-
B	2	A

Tentative Nodes

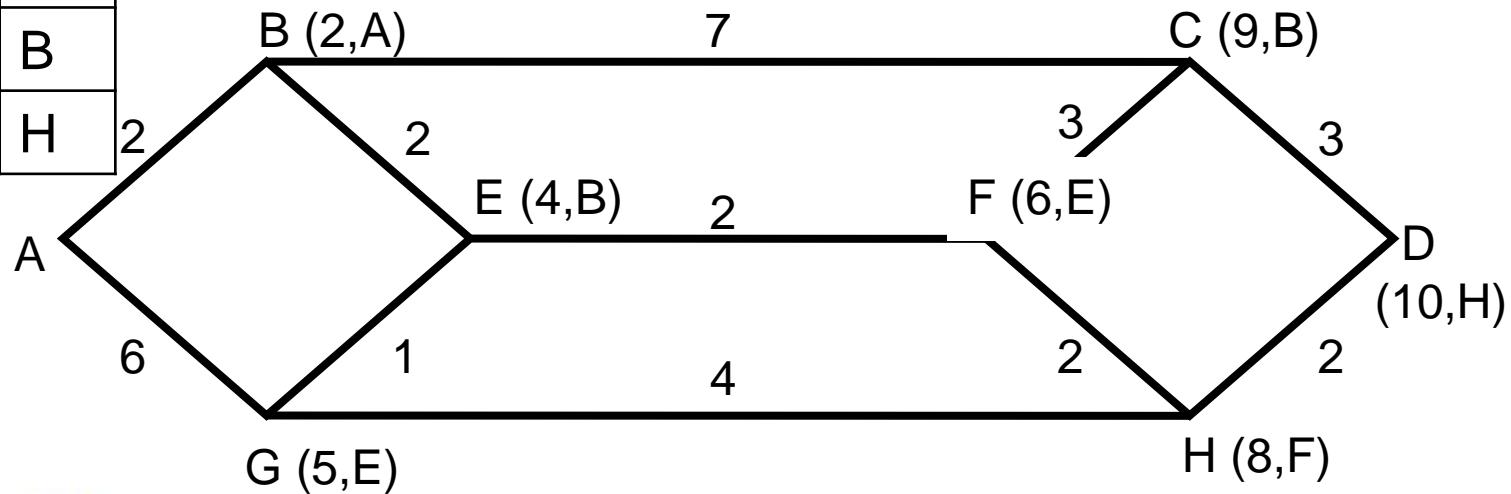
G	6	A
E	4	B
C	9	B



Example: Dijkstra's Algorithm

Routing Table

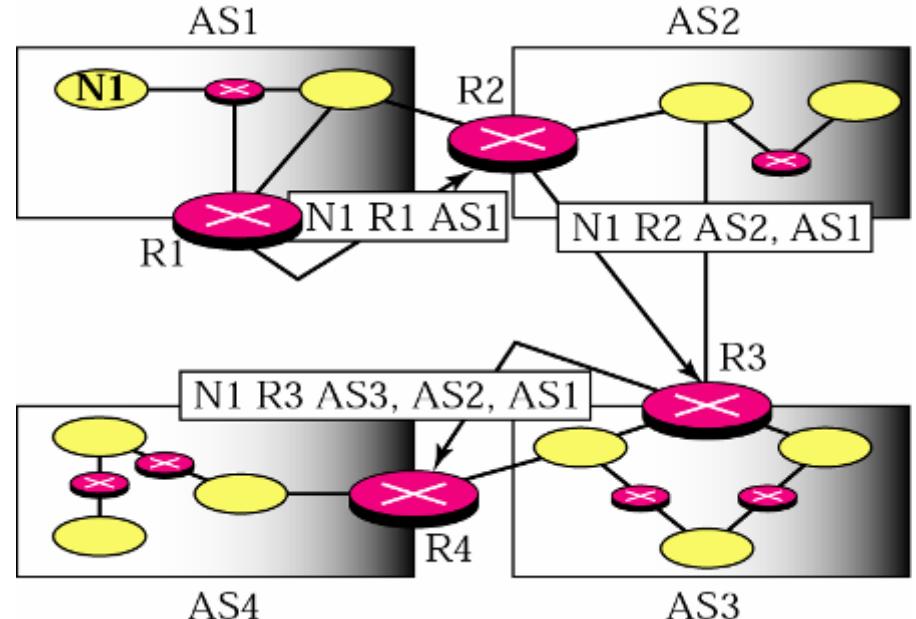
A	0	-
B	2	A
E	4	B
G	5	E
F	6	E
H	8	F
C	9	B
D	10	H



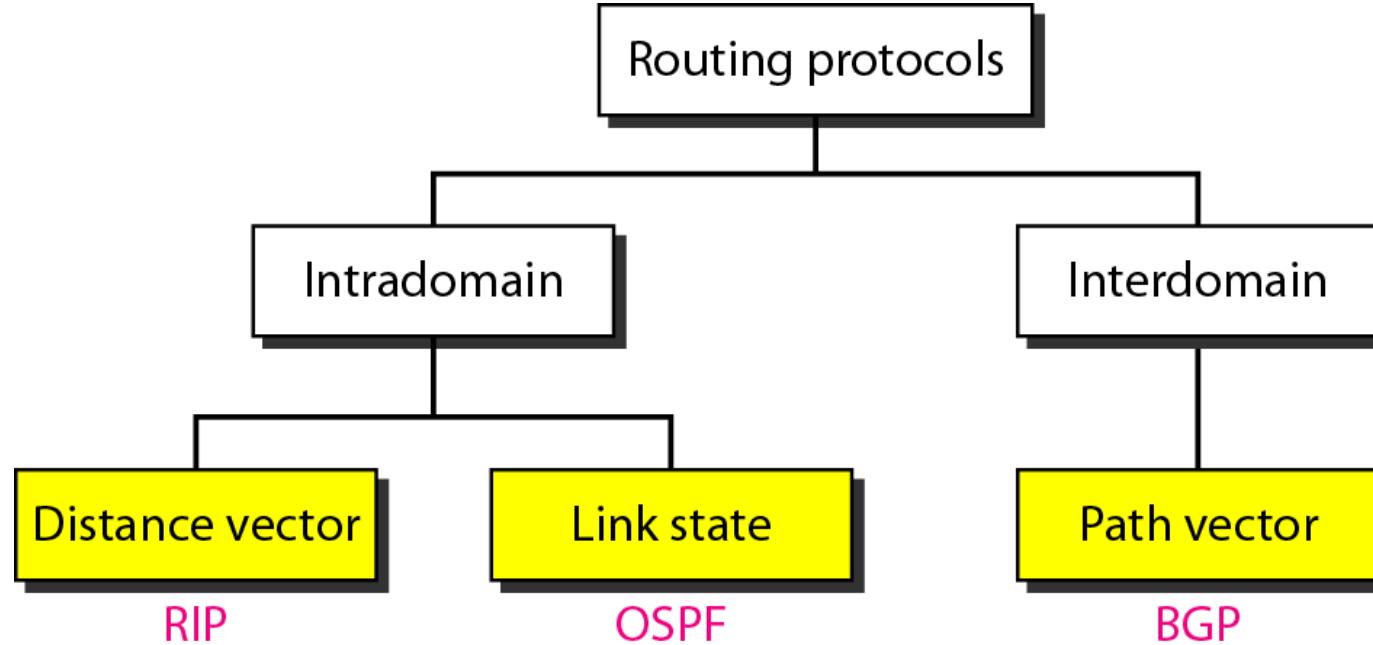
Border Gateway Protocol (BGP)

- Uses Path Vector Routing
- Advertisements include complete path to destination
- Router that forwards advertisement adds itself to the list
- Path can be checked for loops
- Policies are applied when incorporating new routes

Network	Next Router	Path
N01	R01	AS14, AS23, AS67
N02	R05	AS22, AS67, AS05, AS89
N03	R06	AS67, AS89, AS09, AS34
N04	R12	AS62, AS02, AS09



Routing Protocols

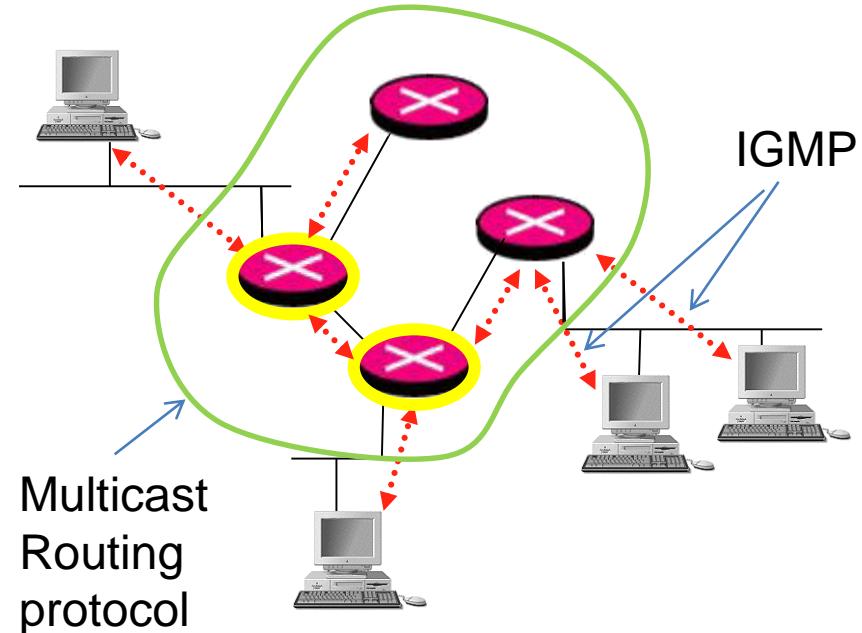


- Distance Vector routing
 - Routes propagate through exchange of routing tables
- Link State routing
 - Establish view of topology & run algorithm e.g. Dijkstra's Shortest-Path



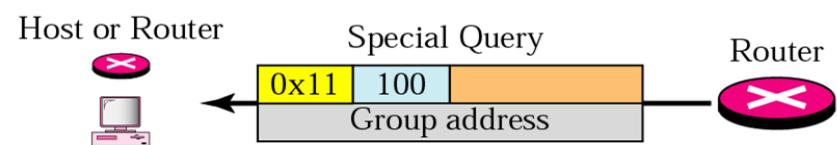
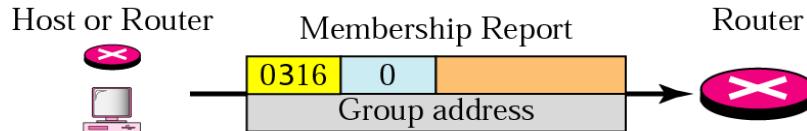
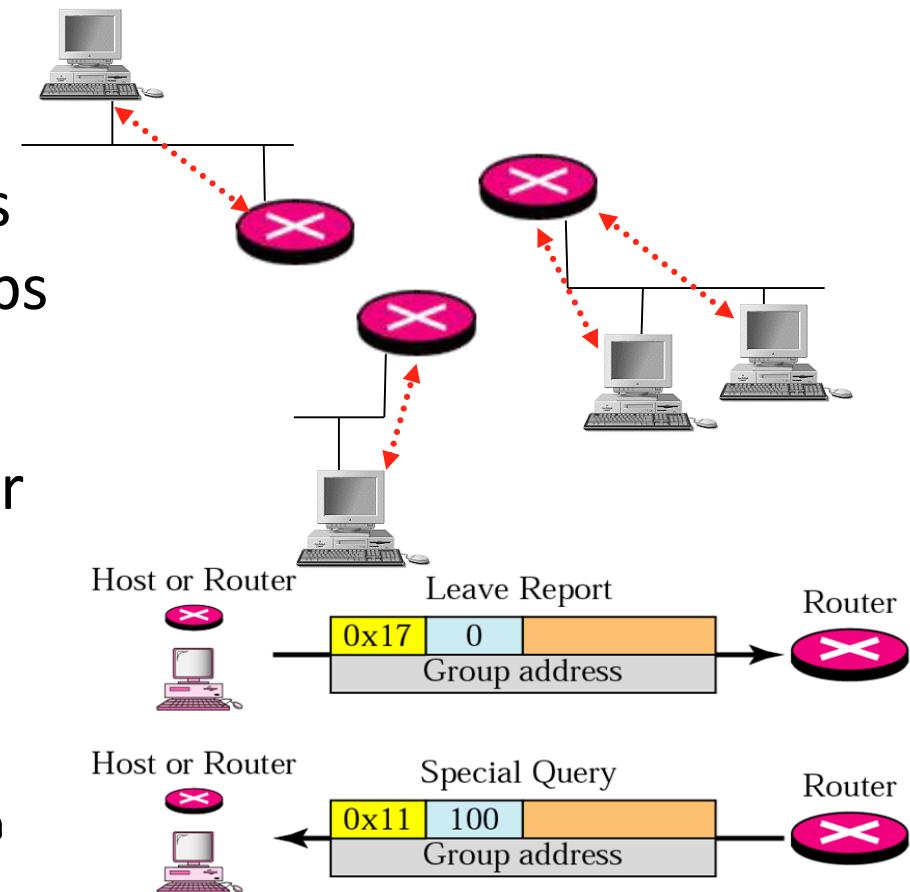
Multicast Overview

- 2-Layer Approach:
 - Endpoints and Routers:
Internet Group Management Protocol (IGMP) between
 - Join & Leave –
 - Routers:
Multicast Routing protocols
- Multicast Addresses:
224.0.0.0 – 239.255.255.255
or 224.0.0.0/4

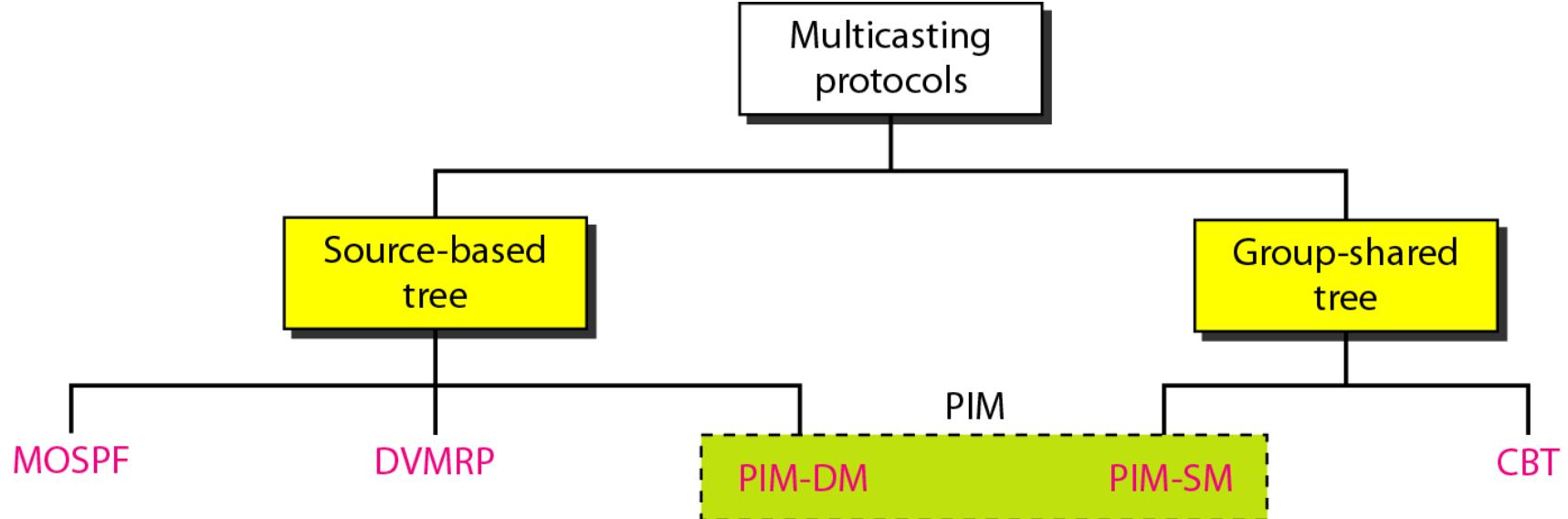


Internet Group Management Protocol (IGMP)

- Defines communication between hosts and router
- Specifies messages for hosts for joining and leaving groups
- Specifies query messages for routers



Multicast Routing Protocols



- Intra-AS
 - MOSPF
 - DVMRP
 - PIM
 - Sparse mode
 - Dense mode
- Inter-AS
 - MBGP + MSDP
 - BGMP + MASC

Summary: Multicast Routing

- Internet Group Management Protocol (IGMP)
 - Join&leave messages from hosts to routers
- Protocol Independent Multicast (PIM)
 - Dense Mode (DM)
 - Start from broadcast and get routers not interested to opt out
 - Sparse Mode (SM)
 - Start from rendezvous point and grow tree from there



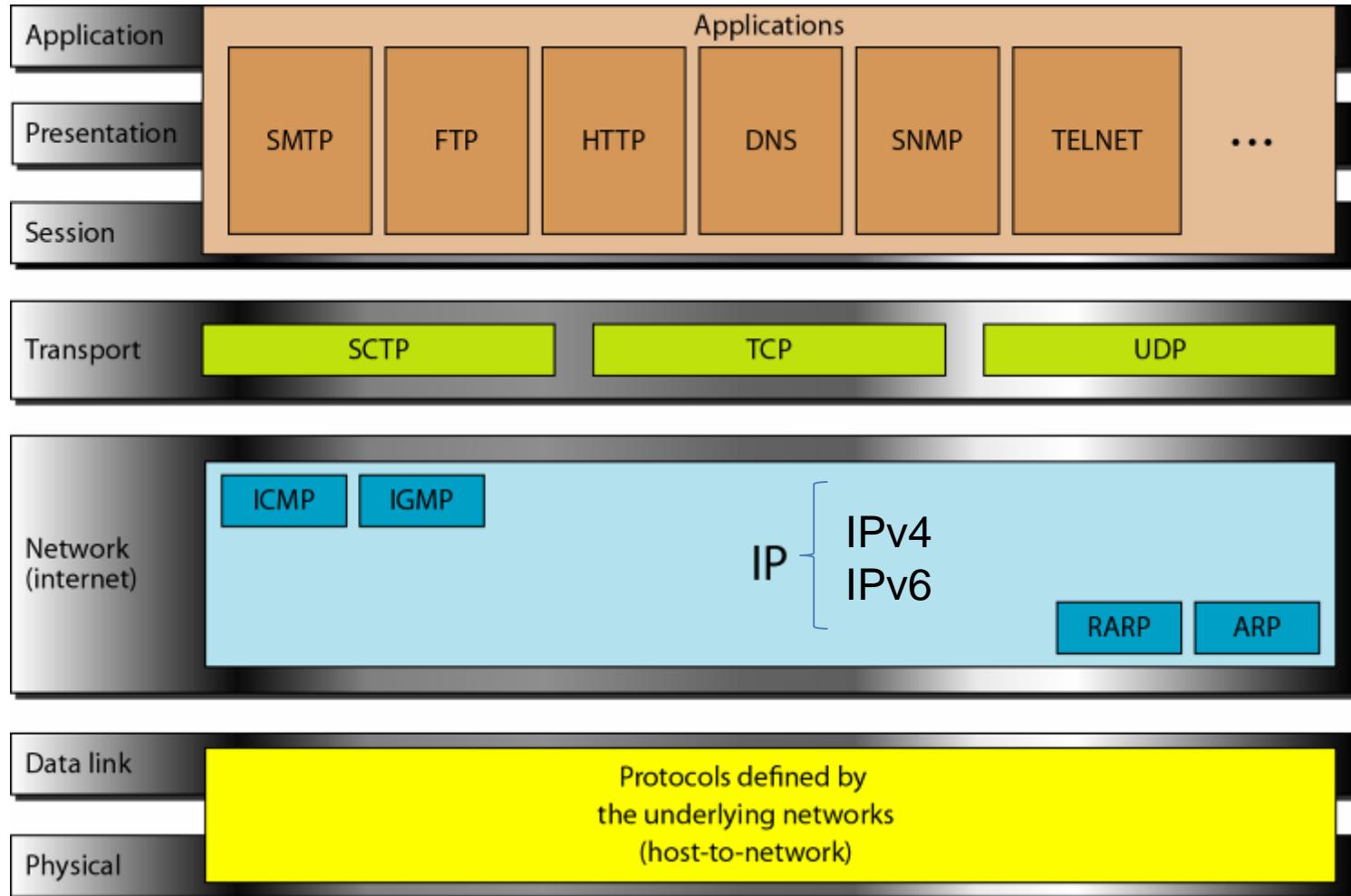
CS2031

Telecommunications II

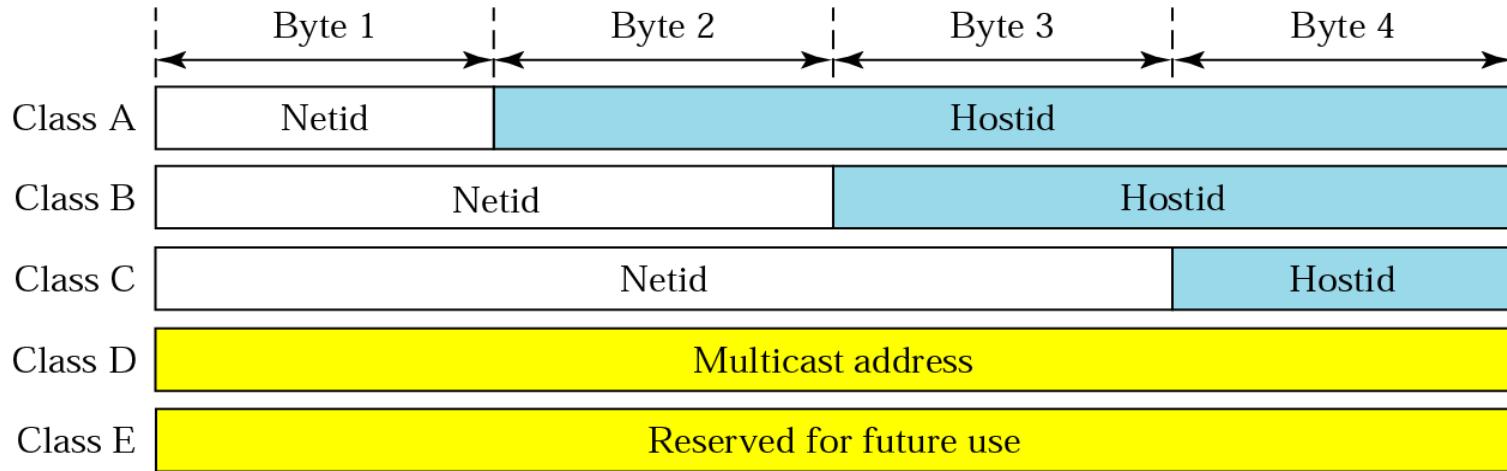
IPv6



Protocols in the OSI Model

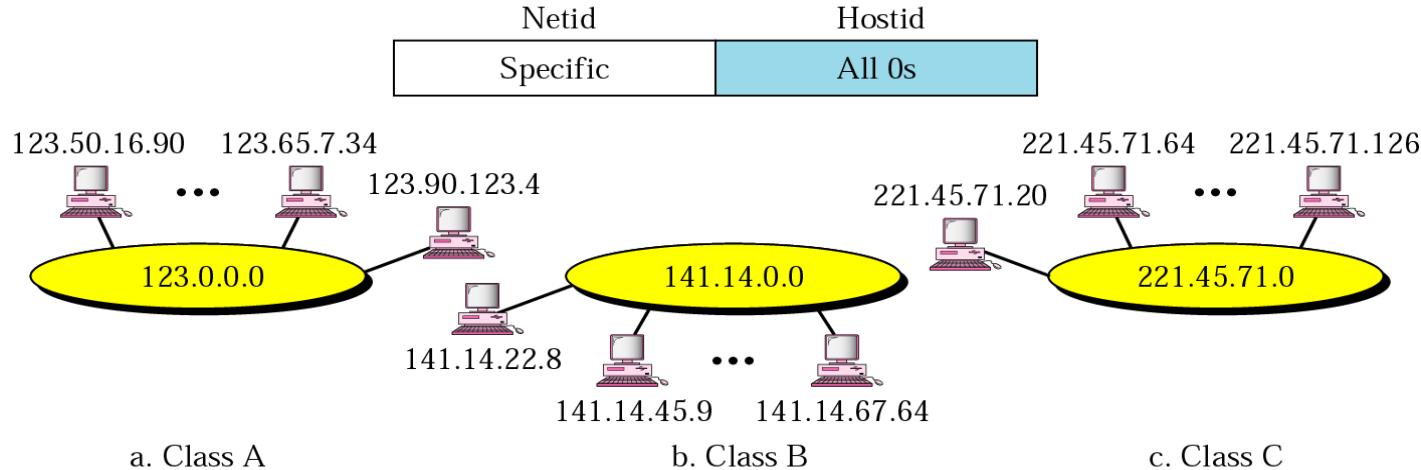


Classful Addresses



- Class A (international organisations)
 - 126 networks with 16,277,214 hosts each
- Class B (large companies)
 - 16,384 networks with 65,354 hosts each
- Class C (smaller companies)
 - 2,097,152 networks with 254 hosts each

Inefficiency of Classful Addresses



- Classful Addresses:

Class	Networks	Addresses
A	126	16,777,214
B	16,382	65,534
C	2,097,152	254

- Inefficient use of Hierarchical Address Space
 - Class C with 2 hosts ($2/254 = 0.78\%$ efficient)
 - Class B with 256 hosts ($256/65534 = 0.39\%$ efficient)

Why IPv6? (Current Business Reasons)

2002 ☺

- Demand from particular regions
 - Asia, EU
 - technical, geo-political, and business reasons
 - demand is now
- Demand for particular services
 - cellular wireless (especially 3GPP standards)
 - Internet gaming (e.g., Sony Playstation 2)
 - use is \geq 1.5 years away (but testbeds needed now)
- Potential move to IPv6 by Microsoft?
 - IPv6 included in Windows XP, but not enabled by default
 - to be enabled by default in next major release of Windows
 - use is \geq 1.5 years away



Why IPv6? (Theoretical Reasons)

- Only compelling reason: **more IP addresses!**
 - For billions of new users (Japan, China, India,...)
 - For billions of new devices (mobile phones, cars, appliances,...)
 - For always-on access (cable, xDSL, ethernet-to-the-home,...)
 - For applications that are difficult, expensive, or impossible to operate through NATs (IP telephony, peer-to-peer gaming, home servers,...)
 - To phase out NATs to improve the robustness, security, performance, and manageability of the Internet



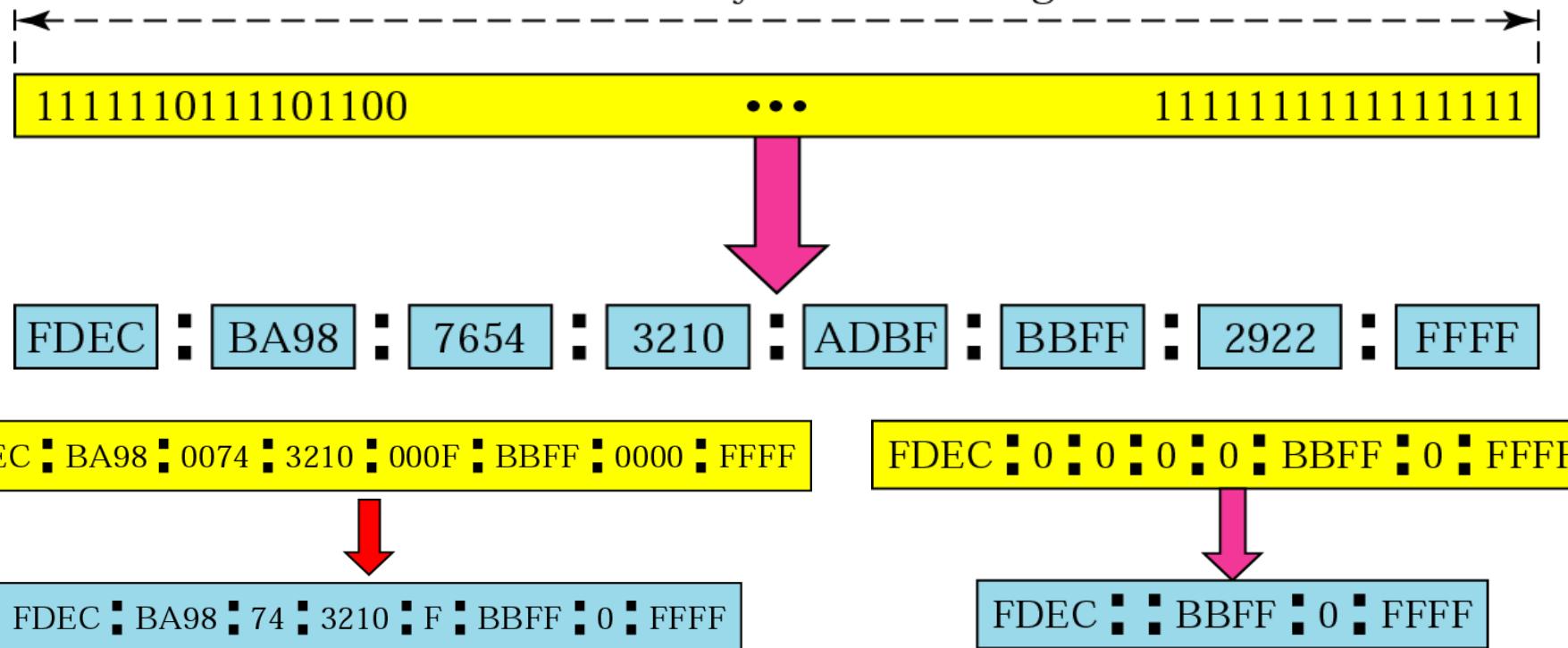
How Was The Address Size Chosen?

- Proposal: Fixed-length, 64-bit addresses
 - Minimizes growth of per-packet header overhead
 - Efficient for software processing
- Proposal: Variable-length, up to 160 bits
 - Compatible with OSI NSAP addressing plans
 - Big enough for auto-configuration using IEEE 802 addresses
 - Could start with addresses shorter than 64 bits & grow later
- Settled on fixed-length, 128-bit addresses
(340,282,366,920,938,463,463,374,607,431,768,211,456 in all!)
 - ~1500 addresses/sqft of the earths surface

IPv6 Address

- Standard representation is set of eight 16-bit values separated by colons

128 bits 5 16 bytes 5 32 hex digits



Address Abbreviation

- Sequences of zeros can be replaced with series of colons

Original

FDEC :: 0074 :: 0000 :: 0000 :: 0000 :: BOFF :: 0000 :: FFF0



Abbreviated

FDEC :: 74 :: 0 :: 0 :: 0 :: BOFF :: 0 :: FFF0

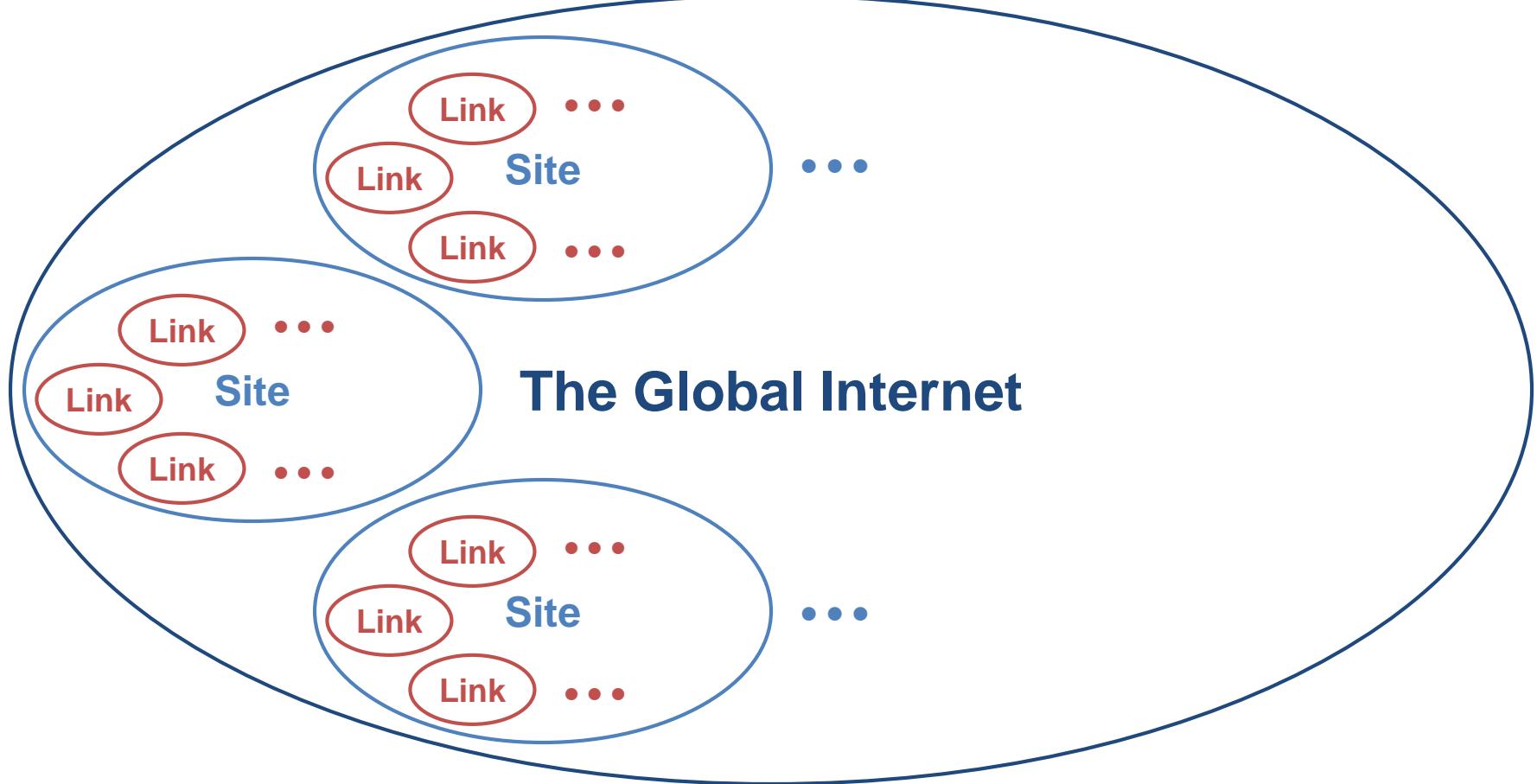


More abbreviated

FDEC :: 74 :: :: BOFF :: 0 :: FFF0

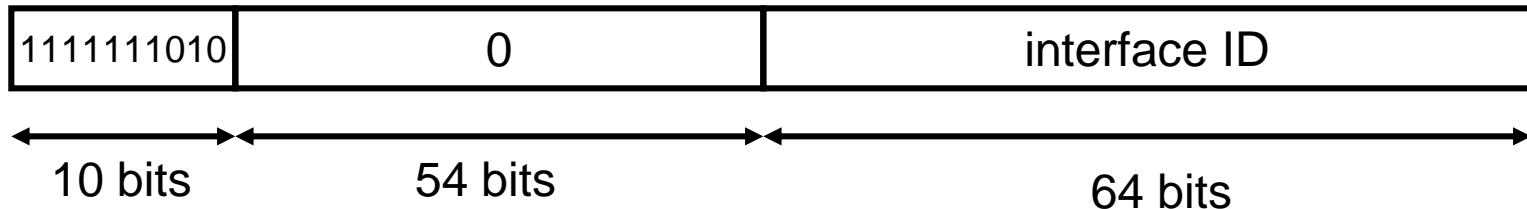


Address Zones and Scopes

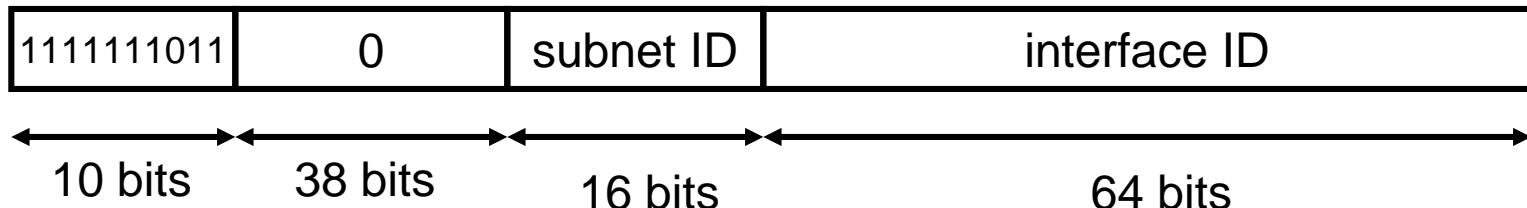


Non-Global Unicast Addresses

- Link-local unicast addresses are meaningful only in a single link zone, and may be re-used on other links

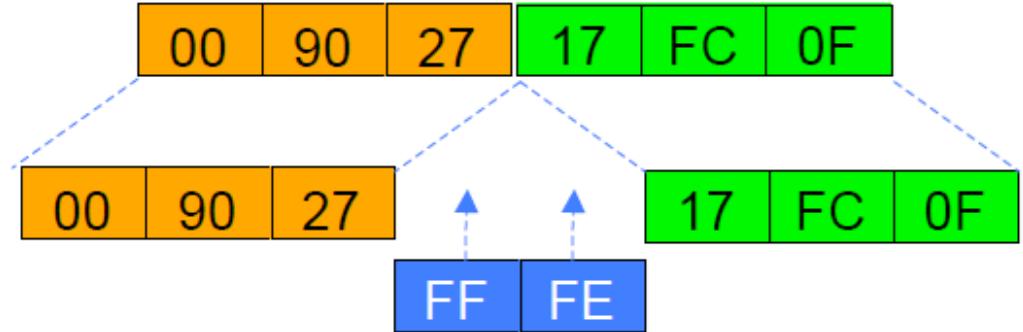


- Site-local unicast addresses are meaningful only in a single site zone, and may be re-used in other sites



64-bit EUI Address

Ethernet MAC address
(48 bits)

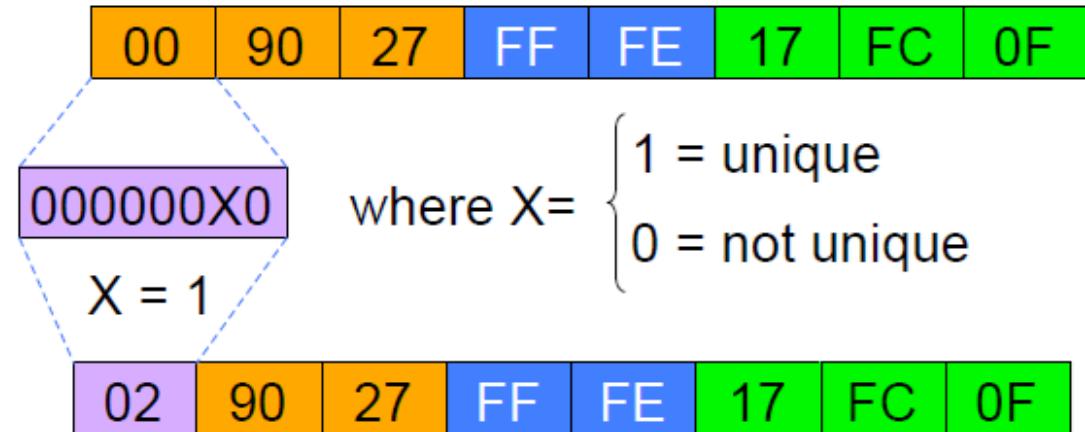


Extended Unique Identifier (EUI)

64 bits version

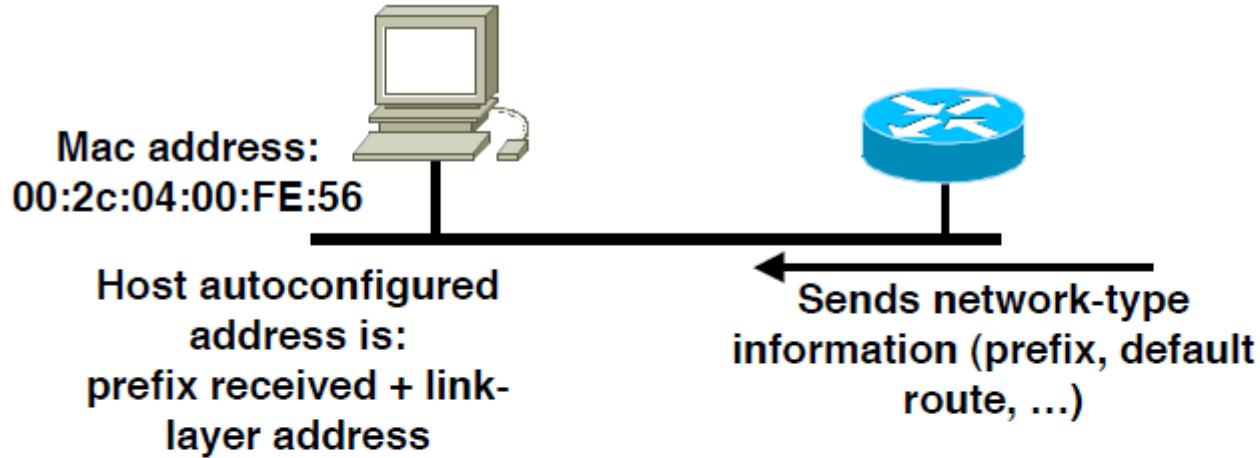
Uniqueness of the MAC

Eui-64 address



- EUI-64 address is formed by inserting FFFE and OR'ing a bit identifying the uniqueness of the MAC address

IPv6 Autoconfiguration



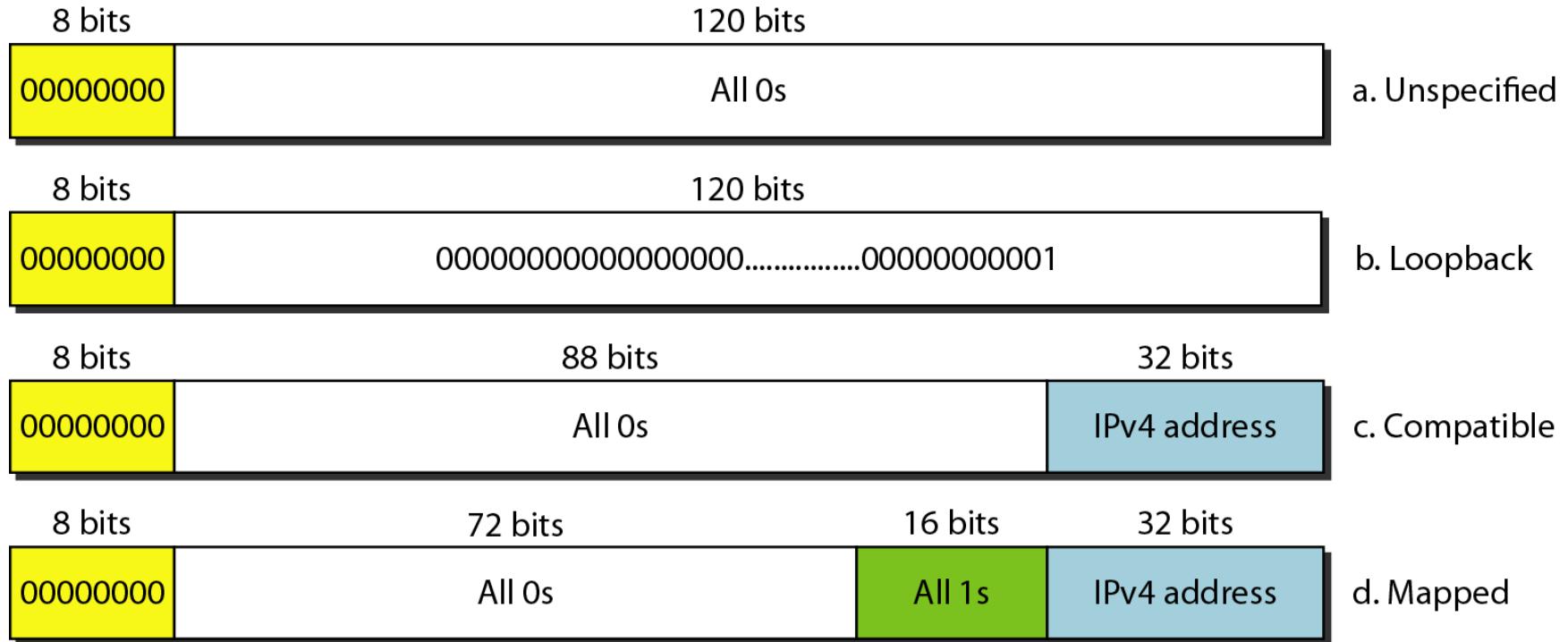
- Client sends router solicitation (RS) messages
- Router responds with router advertisement (RA)
This includes prefix and default route
- Client configures its IPv6 address by concatenating prefix received with its EUI-64 address

Multicast Addresses



- Low-order flag indicates permanent / transient group; three other flags reserved
- Scope field:
 - 1 - interface-local (for multicast loopback)
 - 2 - link-local (same as unicast link-local)
 - 3 - subnet-local
 - 4 - admin-local
 - 5 - site-local (same as unicast site-local)
 - 8 - organization-local
 - B - community-local
 - E - global (same as unicast global)
 - (all other values reserved)

Reserved Addresses



Different Types of Addresses

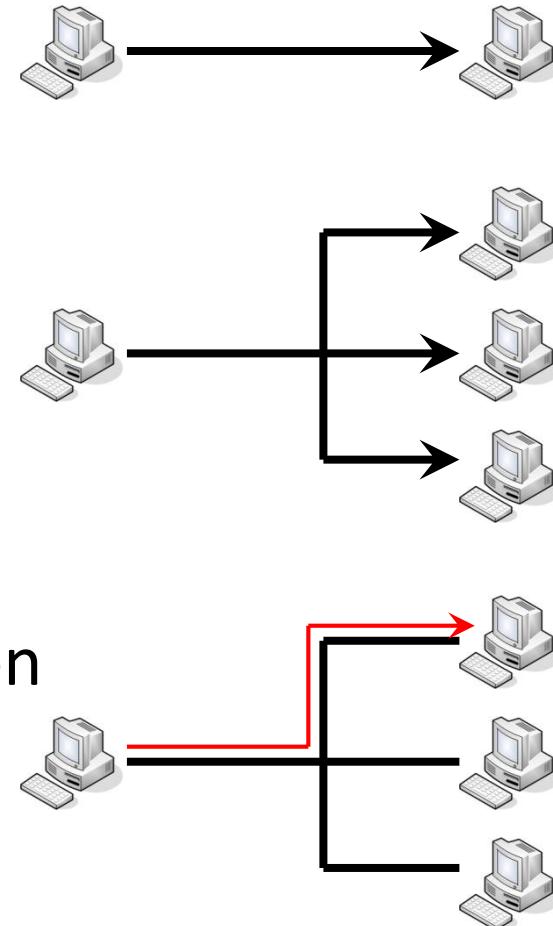
Type	Binary	Hex
Unspecified	000...0	::/128
Loopback	000...1	::1/128
Global Unicast Address	0010	2000::/3
Link Local Unicast Address	1111 1110 10	FE80::/10
Unique Local Unicast Address	1111 1100 1111 1101	FC00::/7
Multicast Address	1111 1111	FF00::/8

Many Addresses

- An interface on an IPv6 node can, and usually will, have many addresses
 - Link-Local
 - Site-Local
 - Auto-configured 6to4
 - Solicited-Node Multicast
 - All-Nodes Multicast
 - Global anonymous
 - Global published

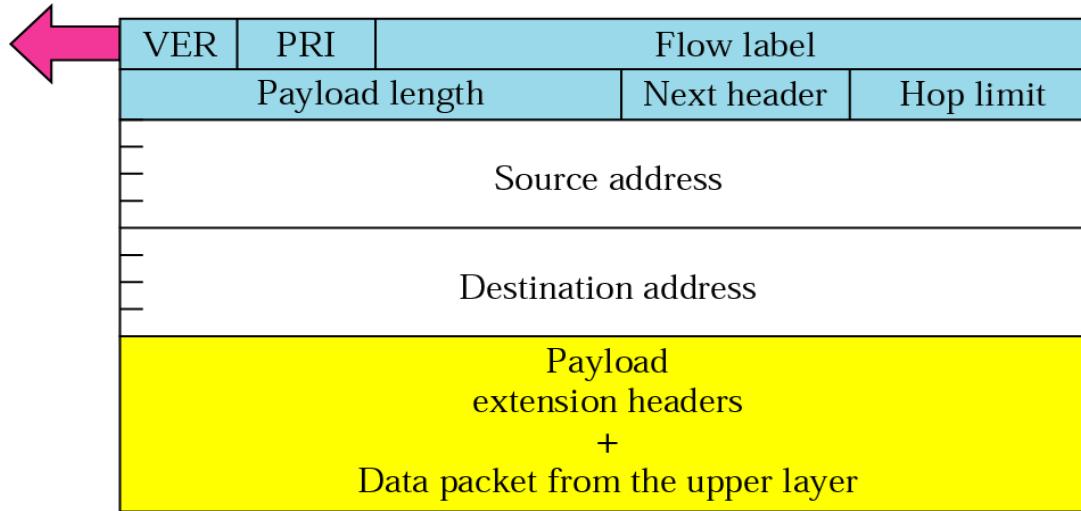
Communication Types

- **Unicast**
 - One-to-one communication
- **Multicast**
 - One-to-many communication
- **Anycast**
 - One-to-nearest communication
 - Delivered to any one interface



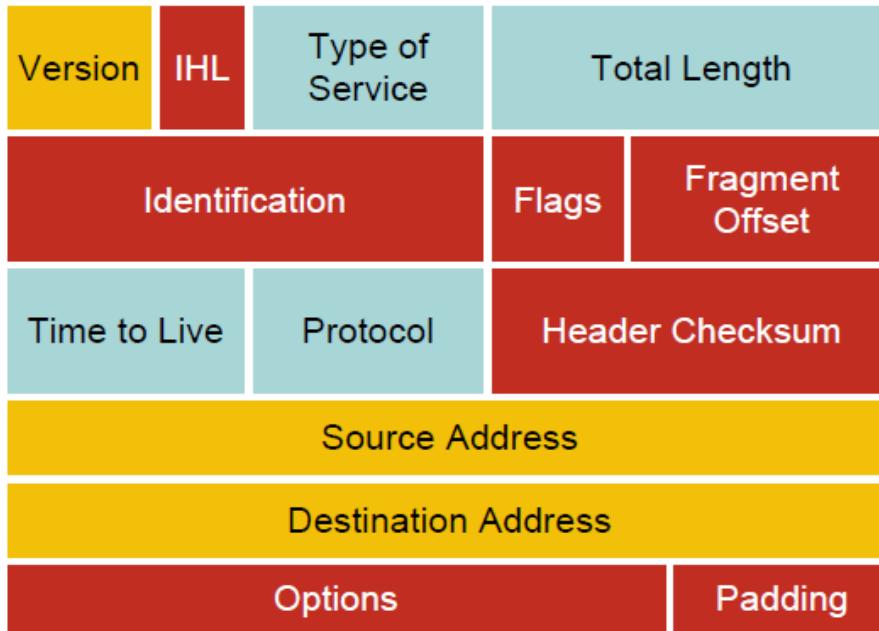
IPv6 Header

- Fixed length of all fields, header length irrelevant
- Remove Header Checksum – other layers are responsible
- No hop-by-hop fragmentation – fragment offset irrelevant
 - MTU discovery before sending or **minimum MTU=1280**
- Extension headers – next header type
- Basic Principle: Routers along the way should do minimal processing



Header Comparison

IPv4 Header



IPv6 Header

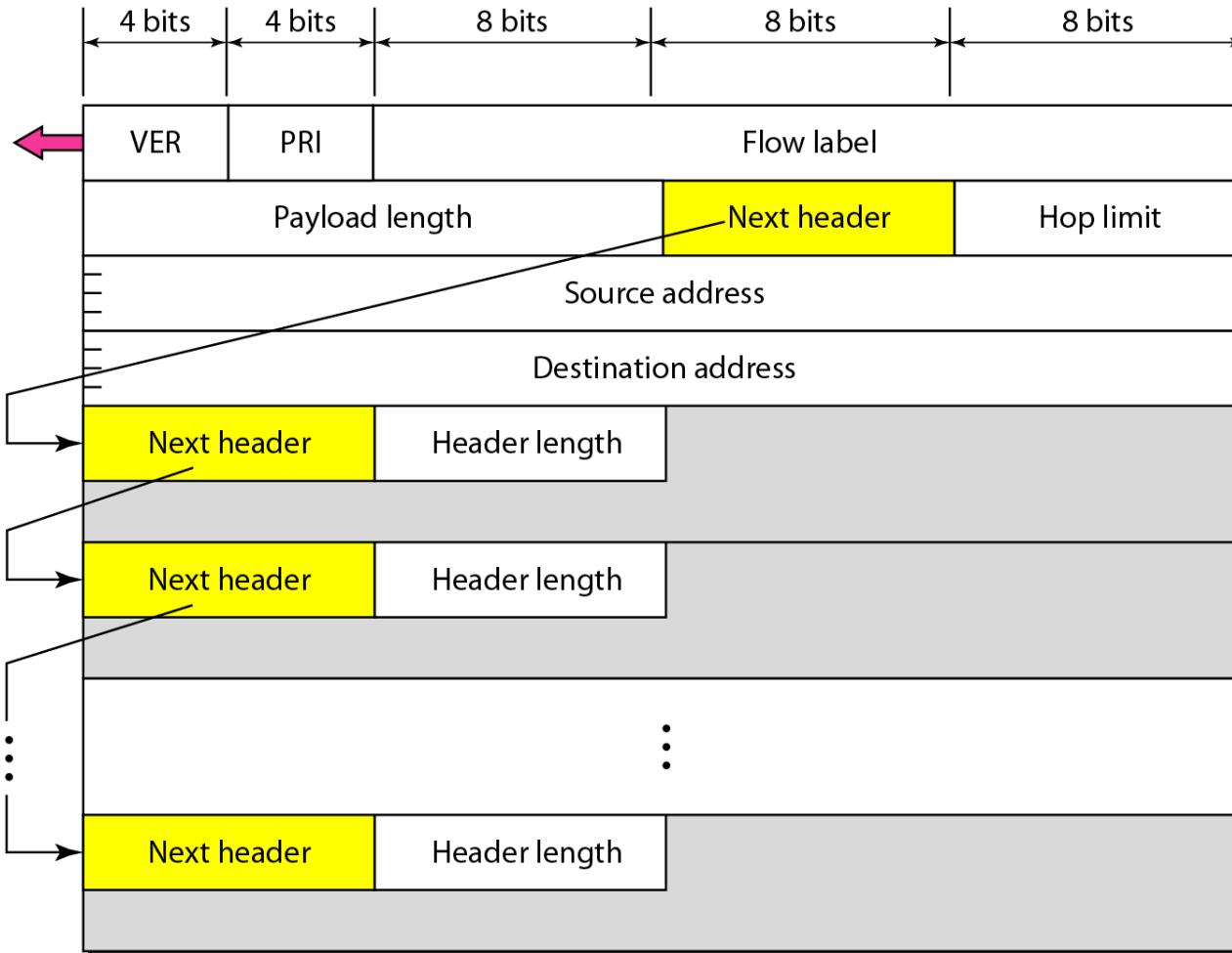


Legend

- Field's name kept from IPv4 to IPv6
- Fields not kept in IPv6
- Name and position changed in IPv6
- New field in IPv6



Extension Headers



Extension Headers

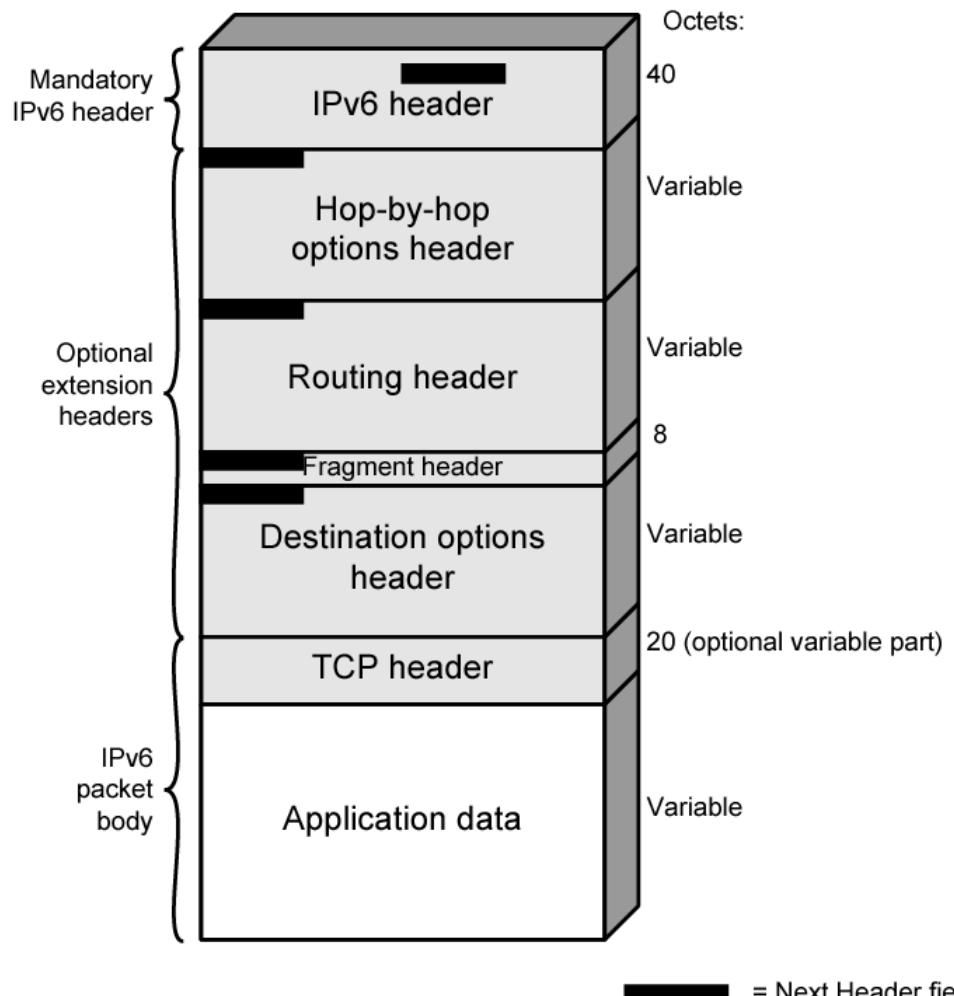
- Hop-by-Hop Options
 - Require processing at each router
- Routing
 - Similar to v4 source routing
- Fragment
- Authentication
- Encapsulating security payload
- Destination options
 - For destination node

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents



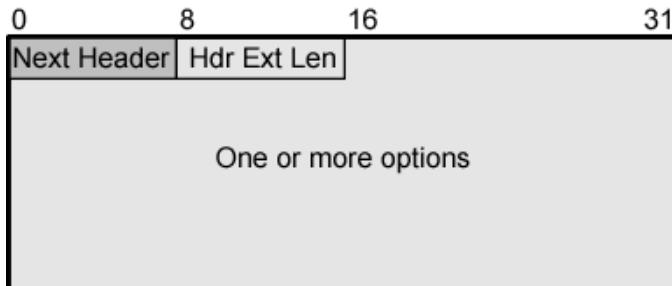
IPv6 Structure

- Every additional header is identified by “next header” field
 - including TCP and UDP header

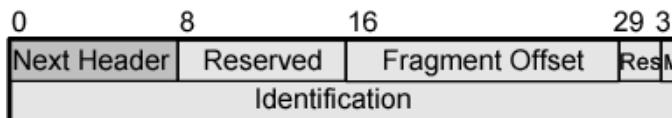


= Next Header field

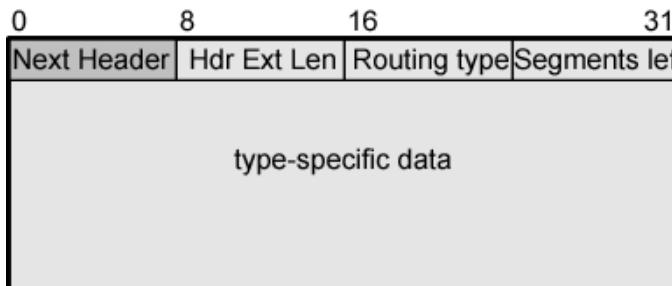
IPv6 Extension Headers



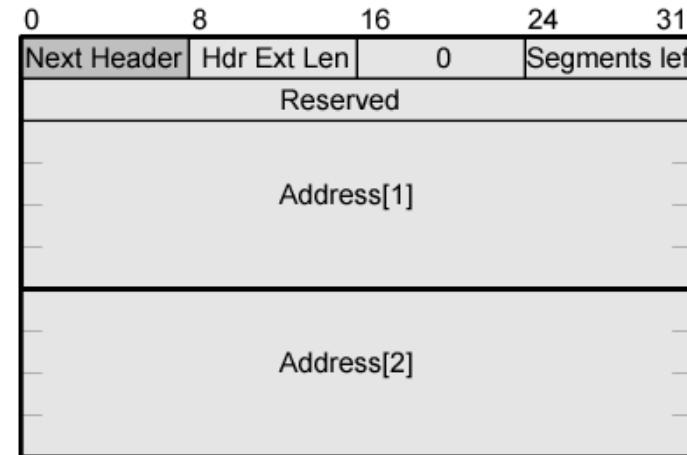
(a) Hop-by-hop options header;
destination options header



(b) Fragment header



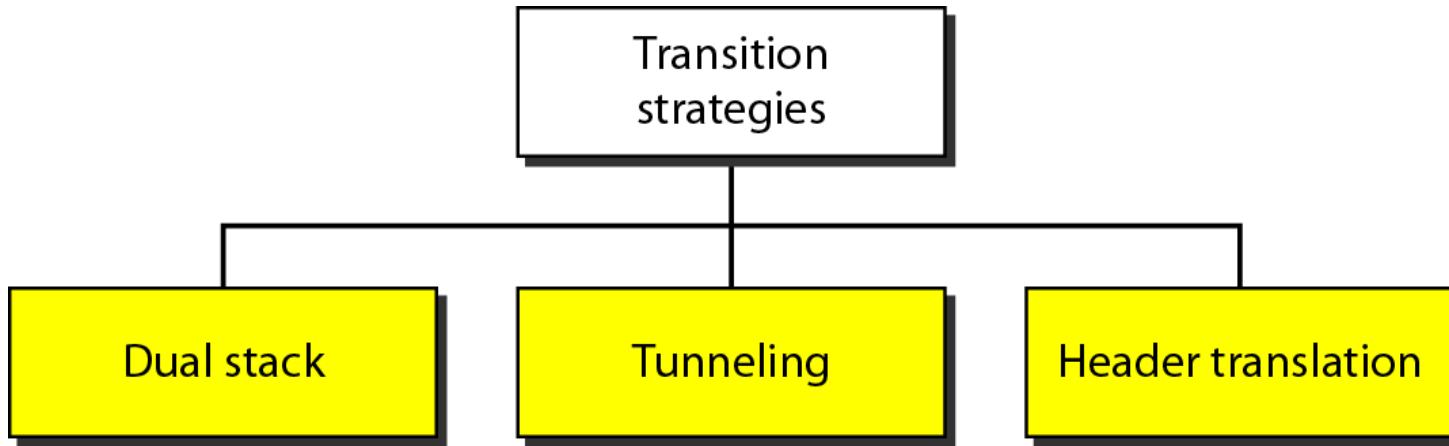
(c) Generic routing header



(d) Type 0 routing header

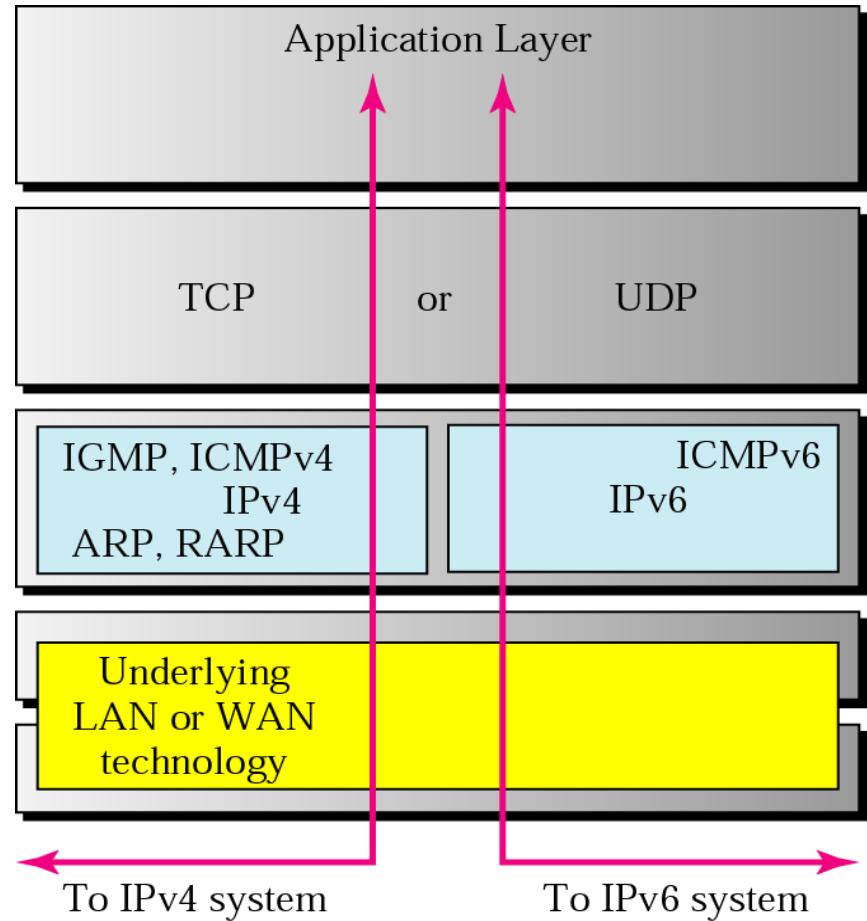


Transition from IPv4 to IPv6



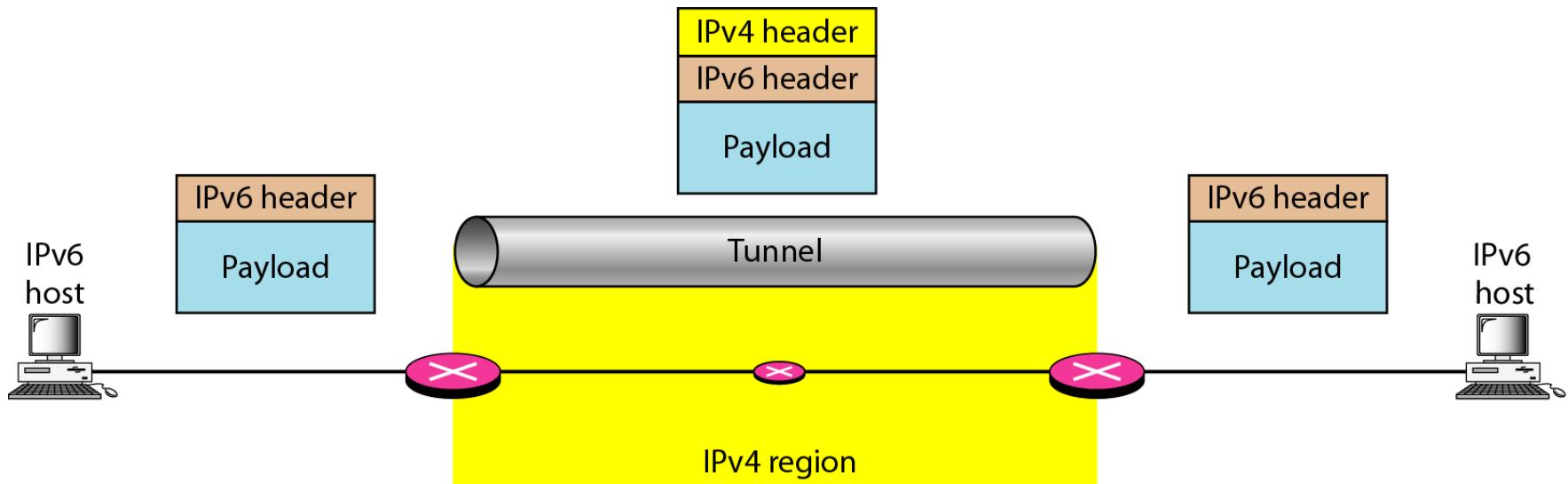
IPv4-IPv6 Transition

- Dual-Stack
 - Stack implements support for both IPv4 and IPv6
 - Allow IPv4 and IPv6 to co-exist in the same devices and networks



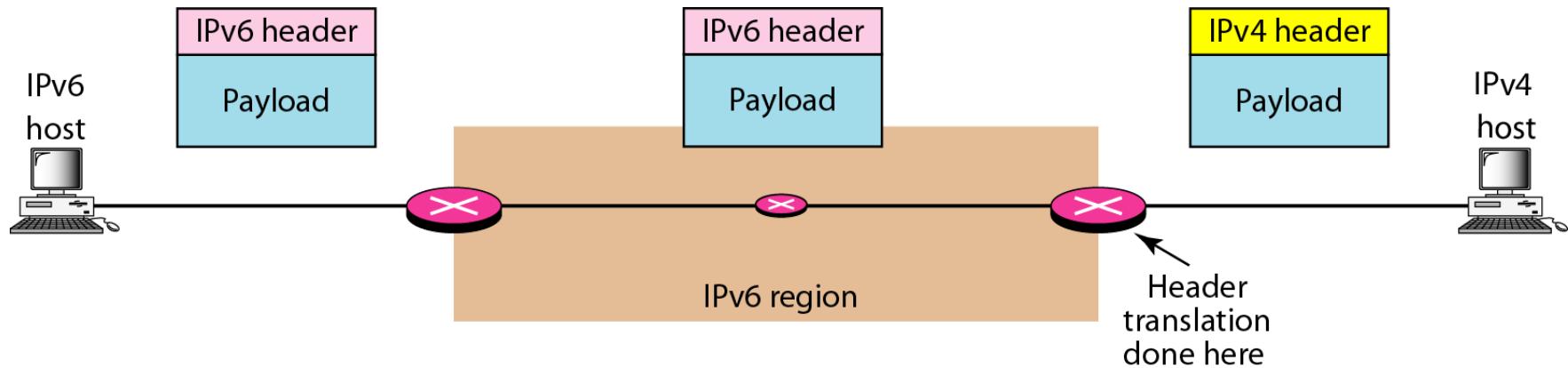
Tunneling

- Encapsulate IPv6 in IPv4 traffic
- Avoid order dependencies when upgrading hosts, routers, or regions



Header Translation

- Headers are translated into other format at “gateway”
- Allow IPv6-only devices to communicate with IPv4-only devices



Summary: IPv6

- Longer addresses: 128 bit
- Simpler, fixed-sized header
- Types of communication
 - Unicast
 - Multicast
 - Anycast
- Extension headers
 - Hop-by-Hop Options, Routing, Fragmentation, Authentication
- Techniques for transition
 - Dual-Stack
 - Tunneling
 - Translation







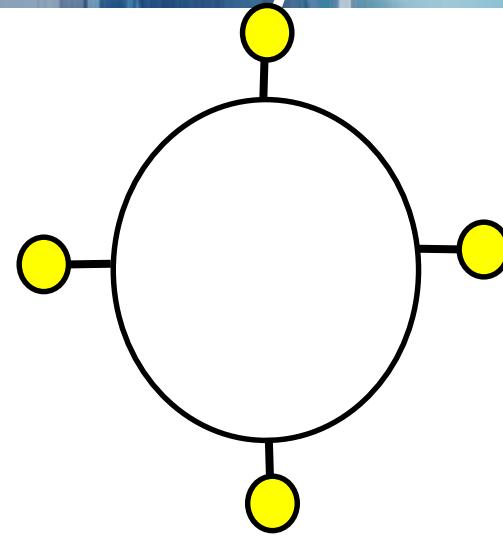
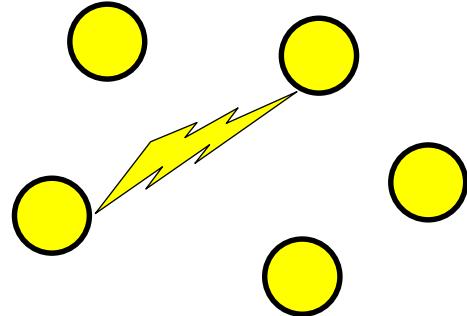
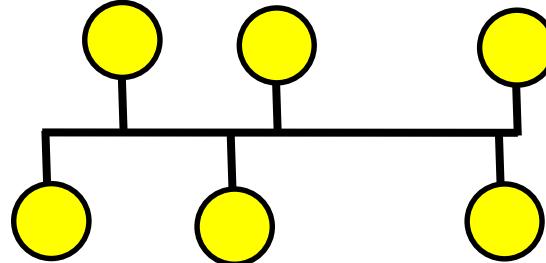
CS2031

Telecommunications II

UDP



Naming at the Link Layer

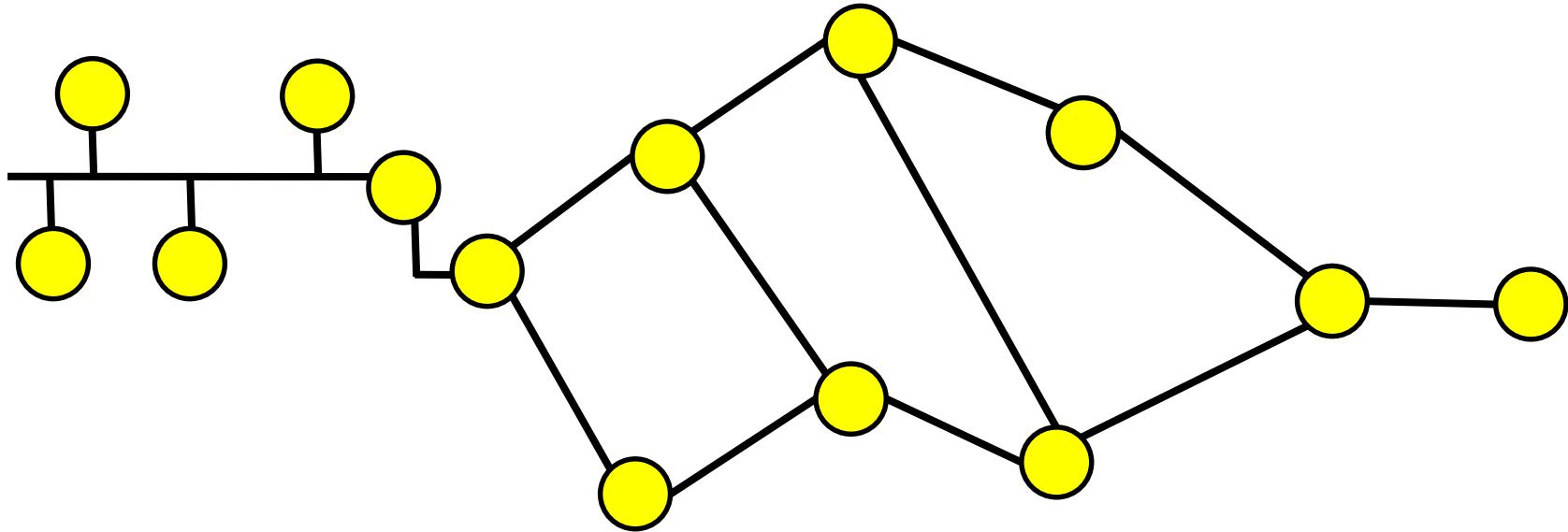


One direct link between
two terminals

○ = Terminals or Stations

Data units = Frames

Naming at the Network Layer

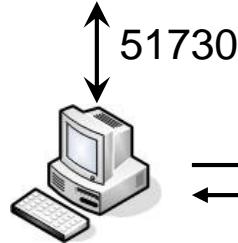


○ = Nodes of a graph

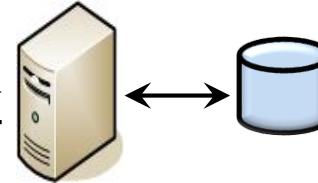
Any number of edges
between two nodes

Data units = Packets

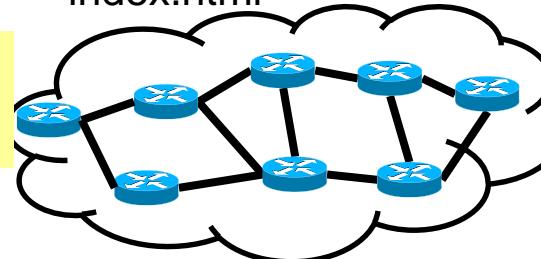
Naming at the Transport Layer



80

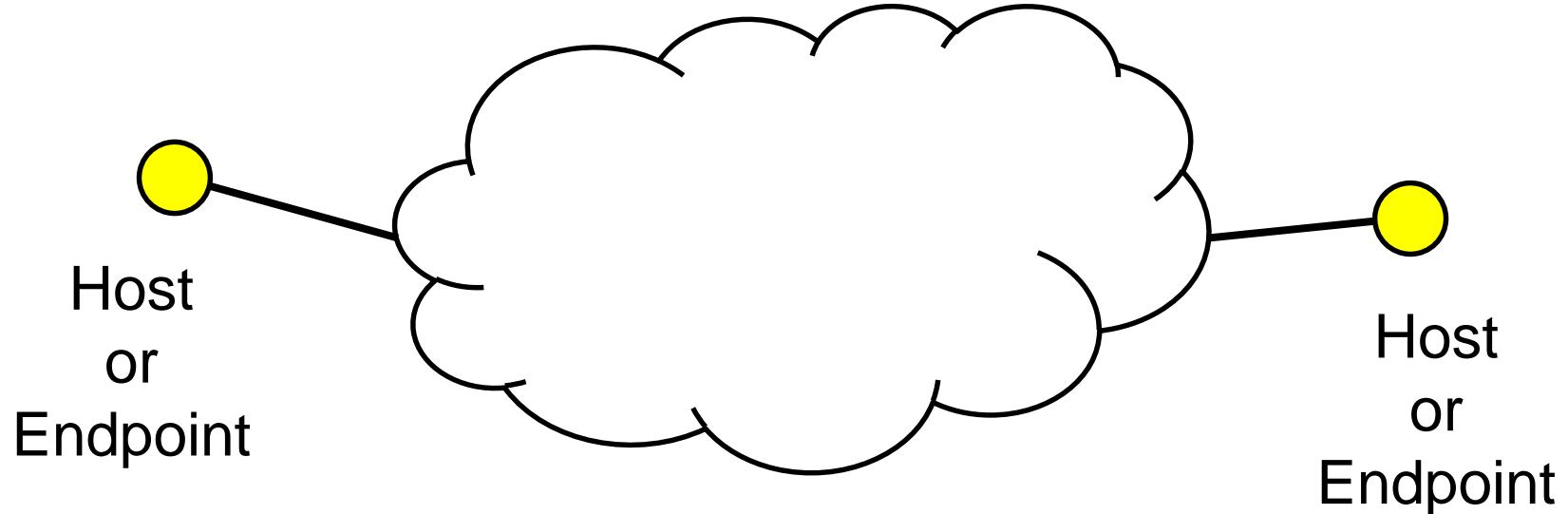


Computer hosting
the webbrowser



Computer hosting
the webserver

Naming at the Transport Layer

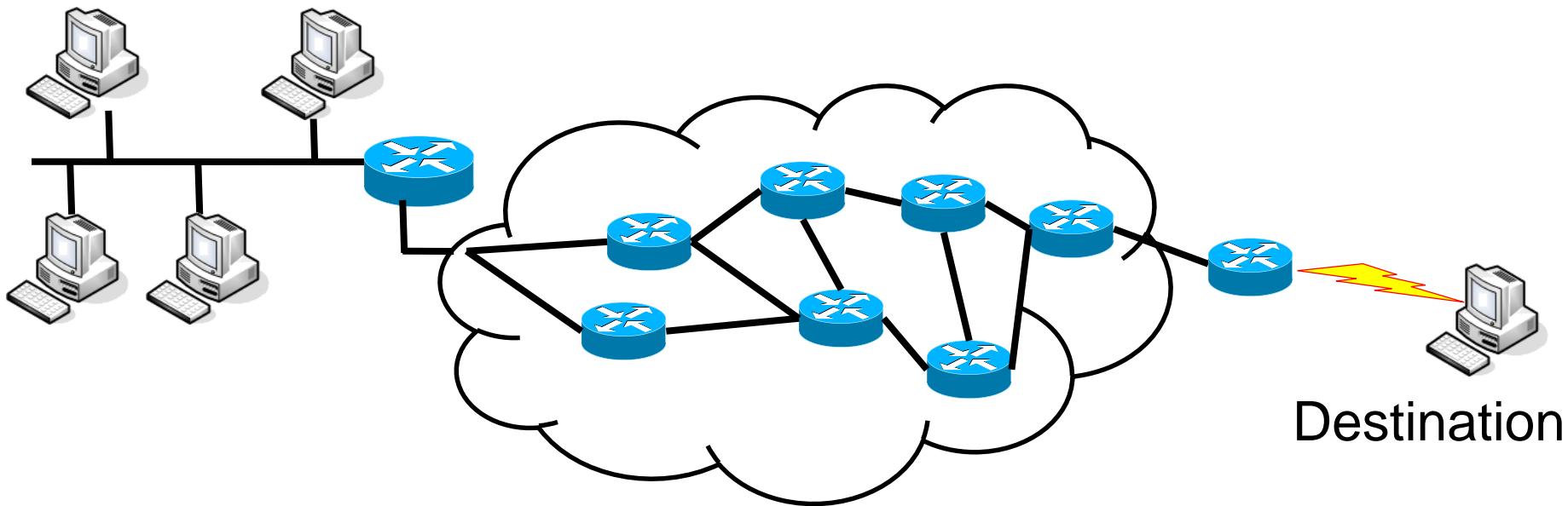


 = Host

Data units = Segments or Datagrams

Task of the Network Layer

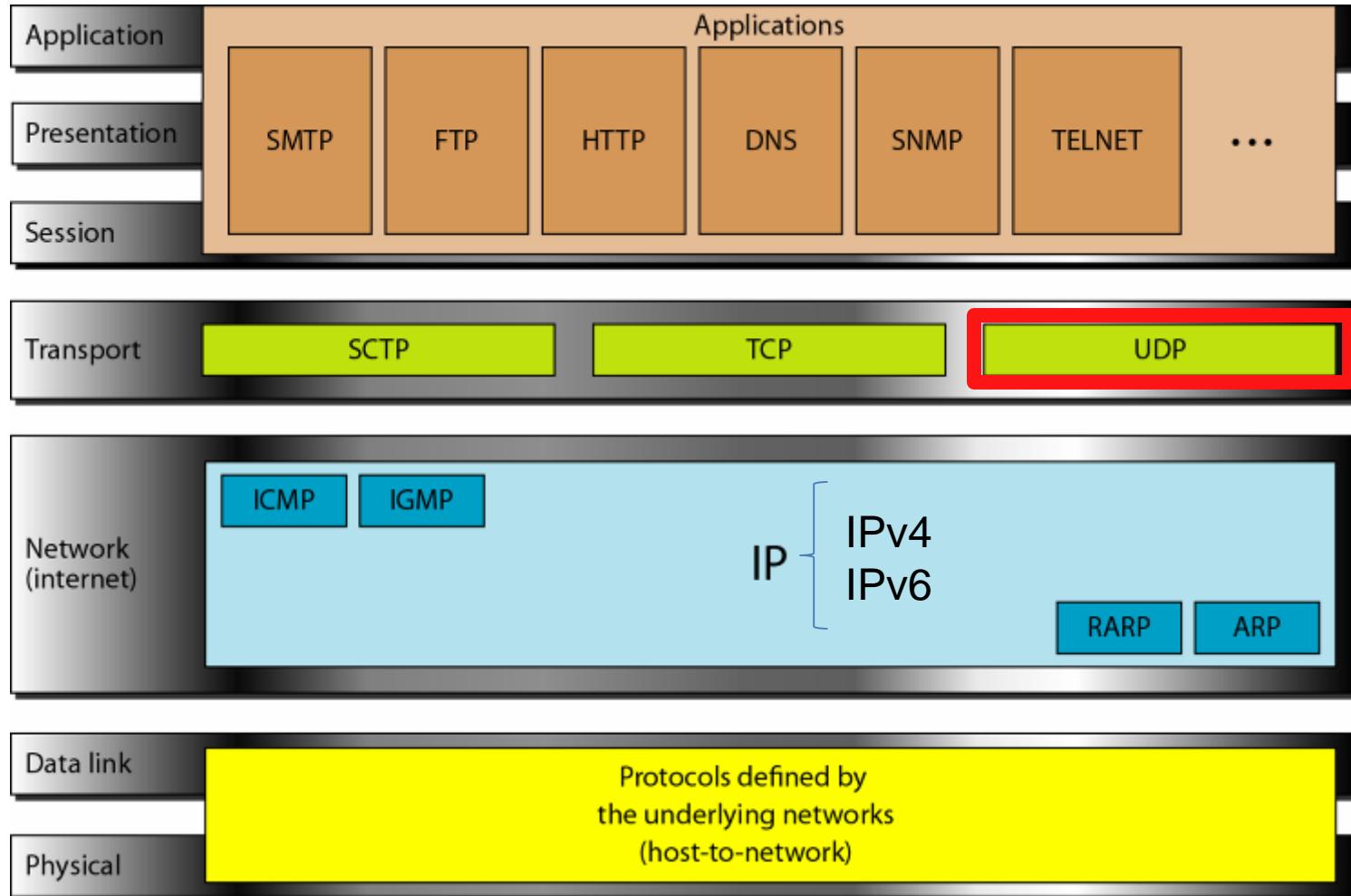
Source



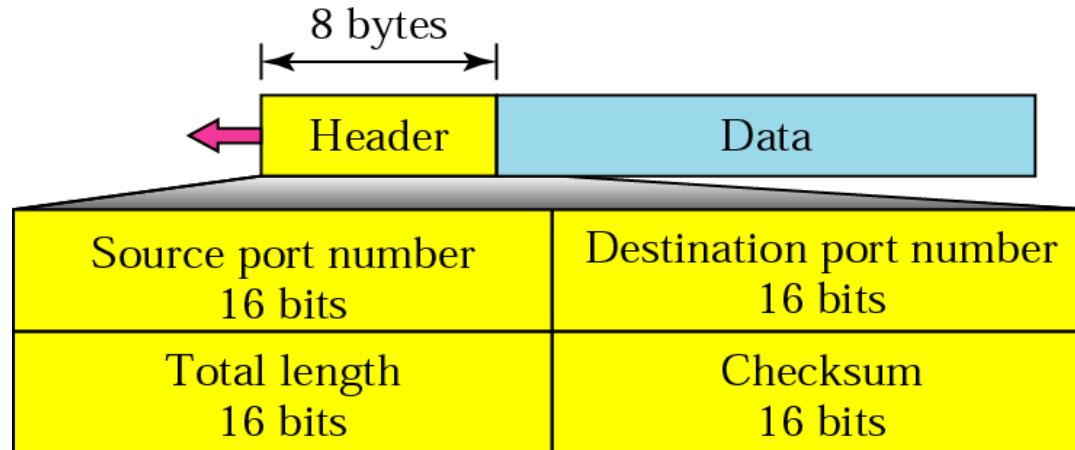
Destination



Protocols in the OSI Model

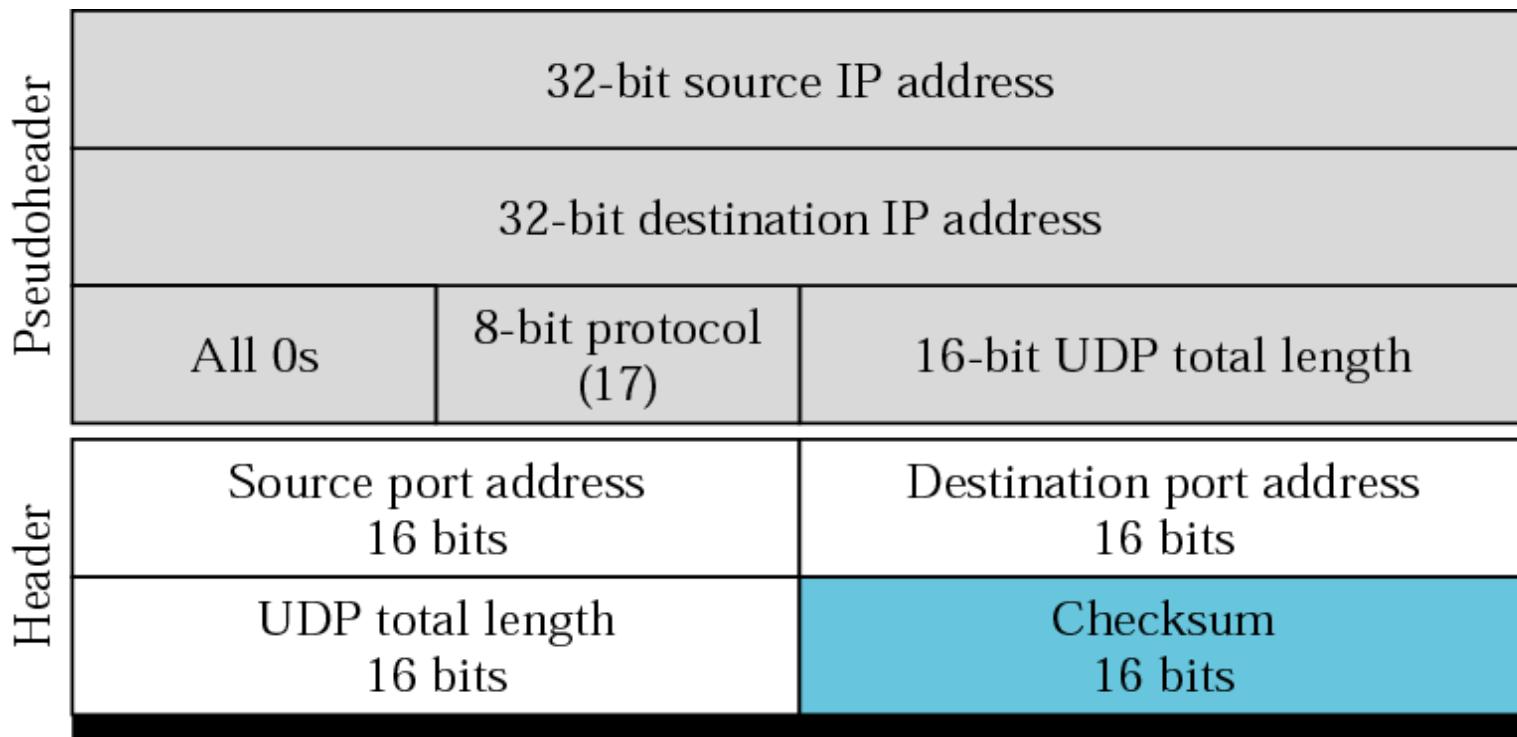


User Datagram Protocol (UDP)



- UDP is a connectionless, unreliable protocol
 - No flow and error control
 - Port numbers are used to multiplex data
- Calculation of checksum & its inclusion in datagram are optional.
- Convenient transport-layer protocol for applications that provide their own flow and error control
 - Also used by multimedia applications.

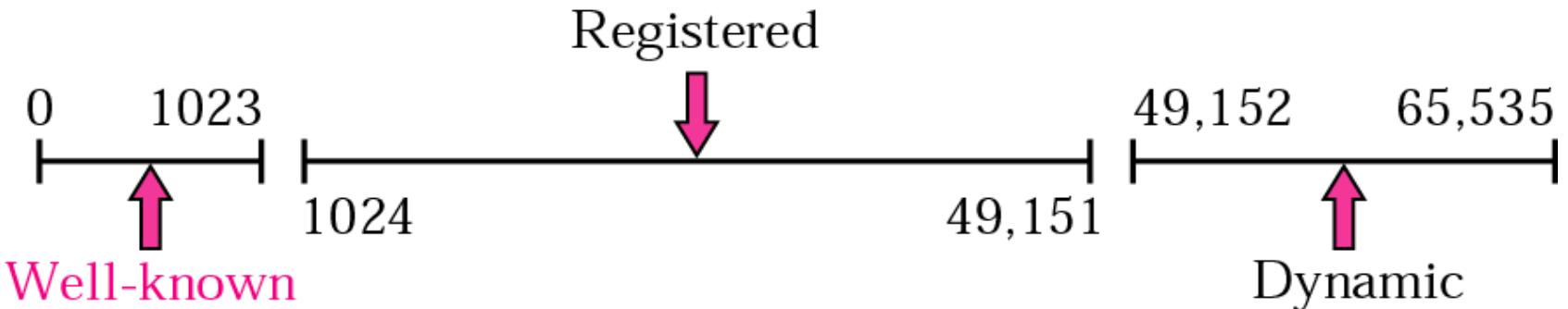
Pseudo-Header for Checksum



Data

(Padding must be added to make the data a multiple of 16 bits)

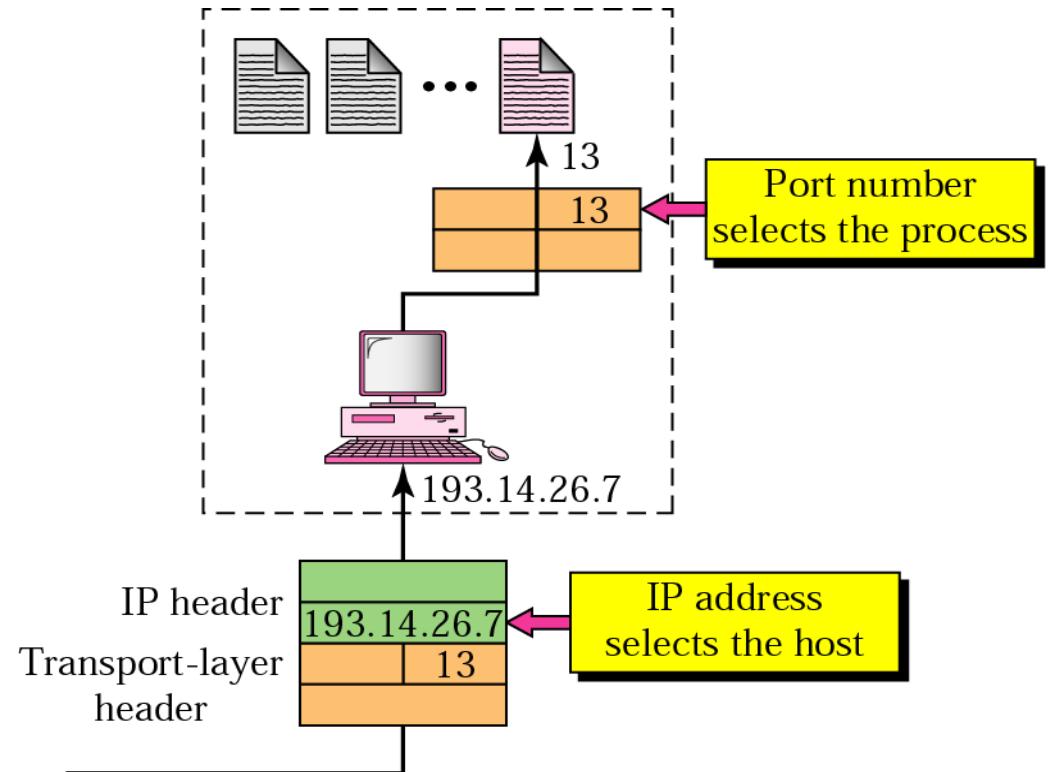
Well-Known Port Numbers



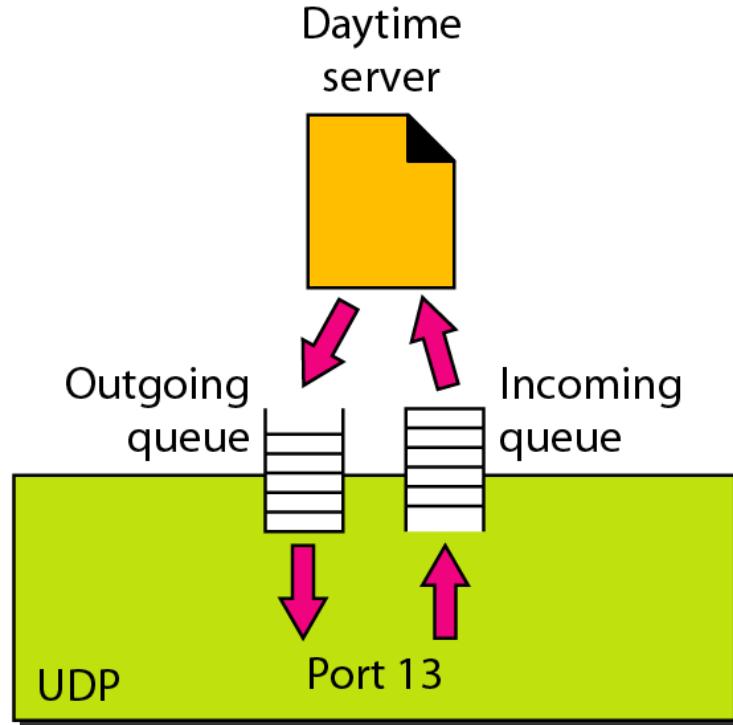
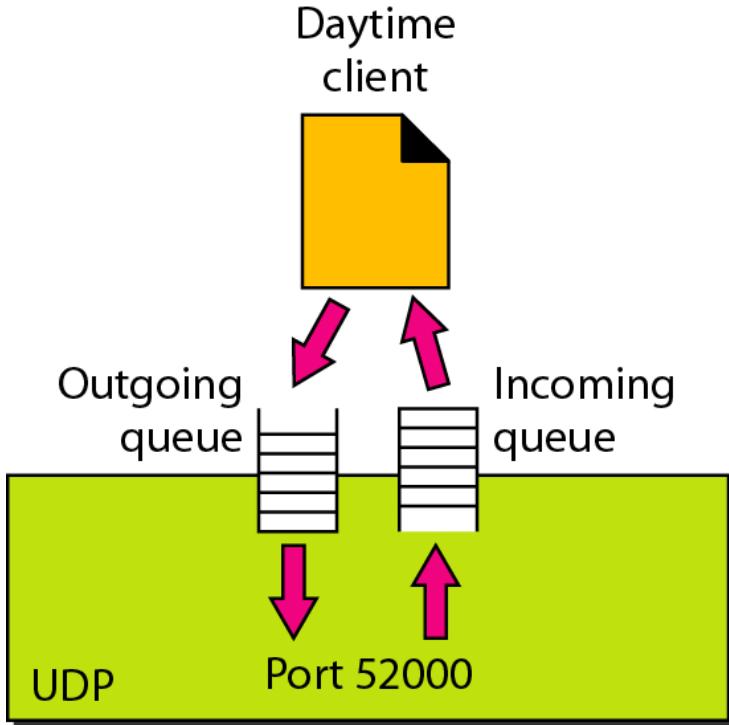
- Assigned by Internet Assigned Numbers Authority (IANA)
- 3 Categories of Ports:
 - Well-known Ports: 0 – 1023 (restricted access)
 - Registered Ports: 1024 – 49151
 - Dynamic/Private Ports: 49152 – 65535

IP Addresses & Port Numbers

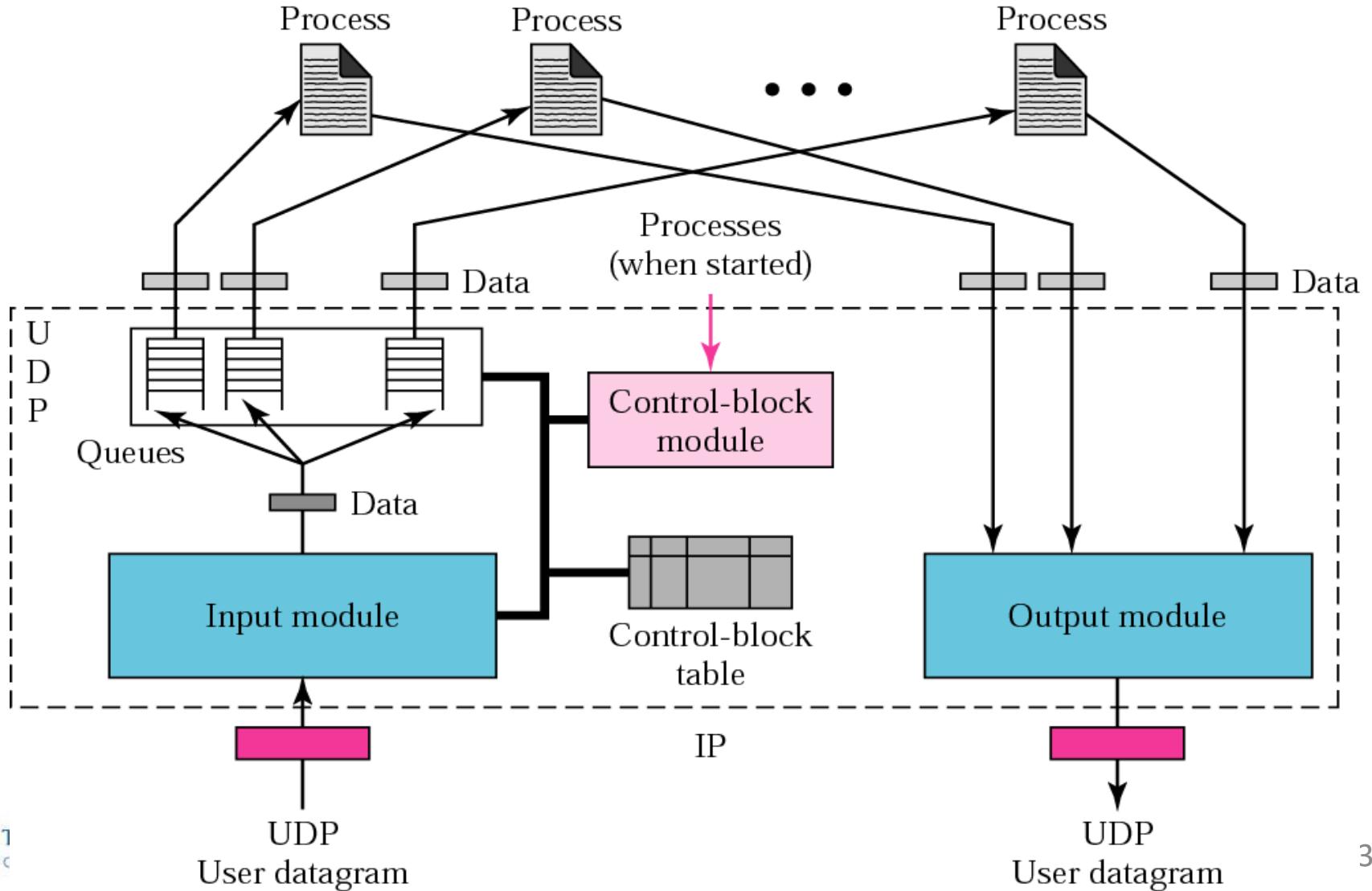
- IP Addresses determine the host
- Port Numbers determine the application



Queuing in UDP



Processes and UDP Queues



UDP Queue Example I

<i>State</i>	<i>Process ID</i>	<i>Port Number</i>	<i>Queue Number</i>
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
FREE			
IN-USE	4,652	52,012	38
FREE			

- Socket is opened on port 52014

<i>State</i>	<i>Process ID</i>	<i>Port Number</i>	<i>Queue Number</i>
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
IN-USE	4,978	52,014	
IN-USE	4,652	52,012	38
FREE			

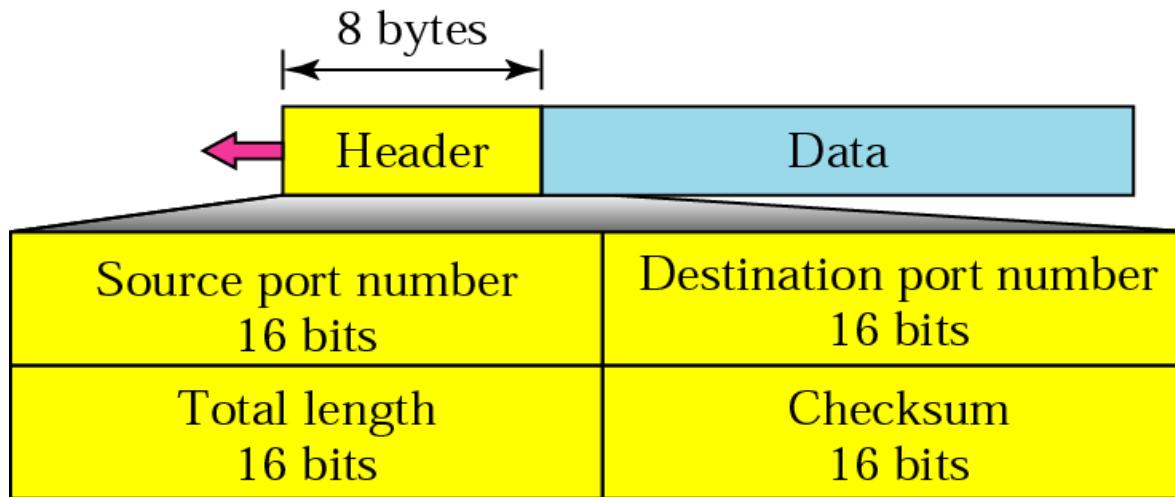
UDP Queue Example II

<i>State</i>	<i>Process ID</i>	<i>Port Number</i>	<i>Queue Number</i>
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	43
IN-USE	4,978	52,014	
IN-USE	4,652	52,012	38
FREE			

- Packet for Port 52011 arrives
 - Queue is created and packet is queued
- Packet for Port 53255 arrives
 - Packet is dropped

User Datagram Protocol (UDP)

- Connectionless
- Unreliable
 - No flow or error control
- Small Header:





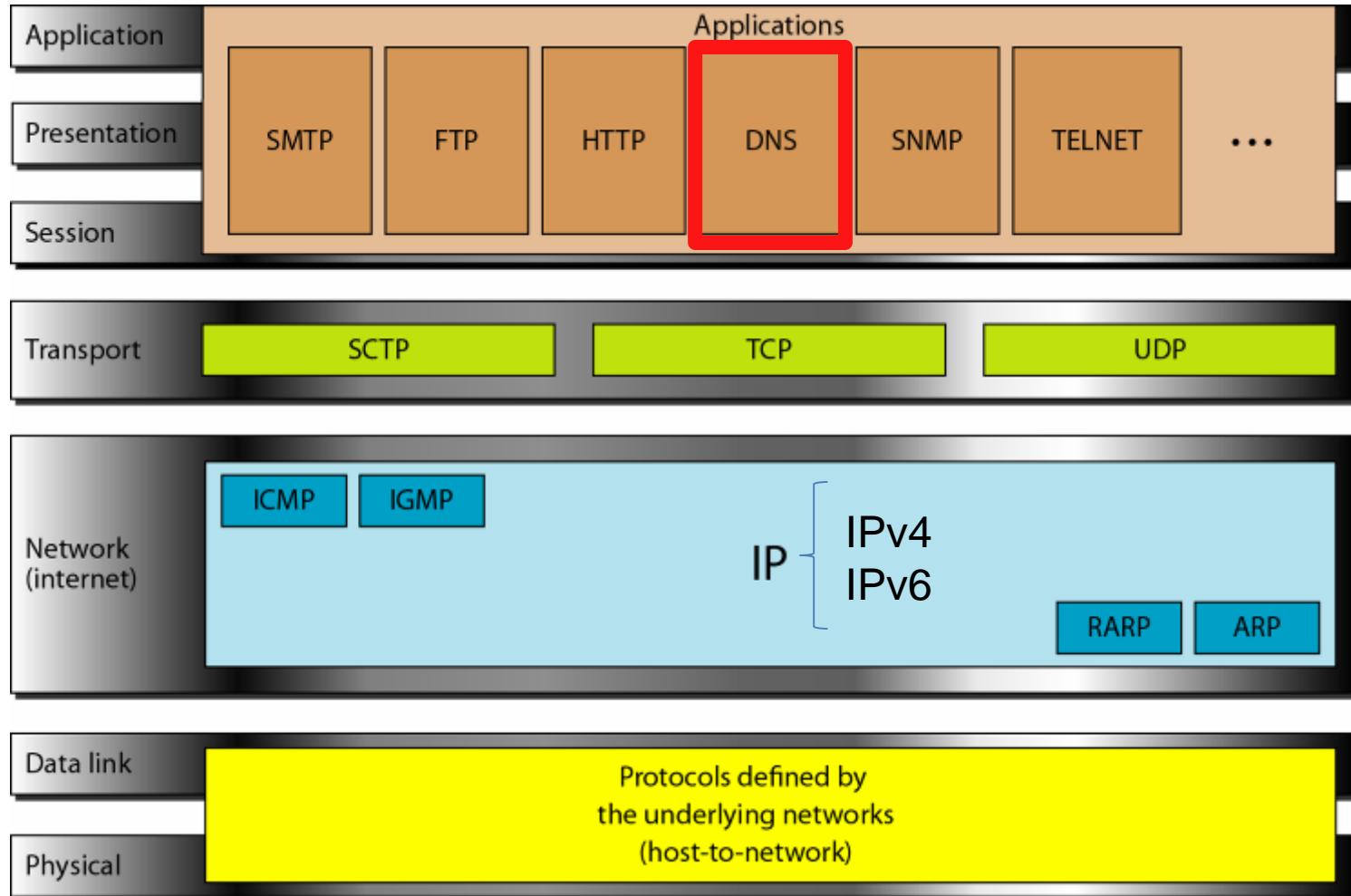
CS2031

Telecommunications II

DNS



Protocols in the OSI Model



URLs to Names to Addresses

URL

DNS

IP Address

`http://www.wiki.com/index.html`

`www.wiki.com`

66.96.149.1

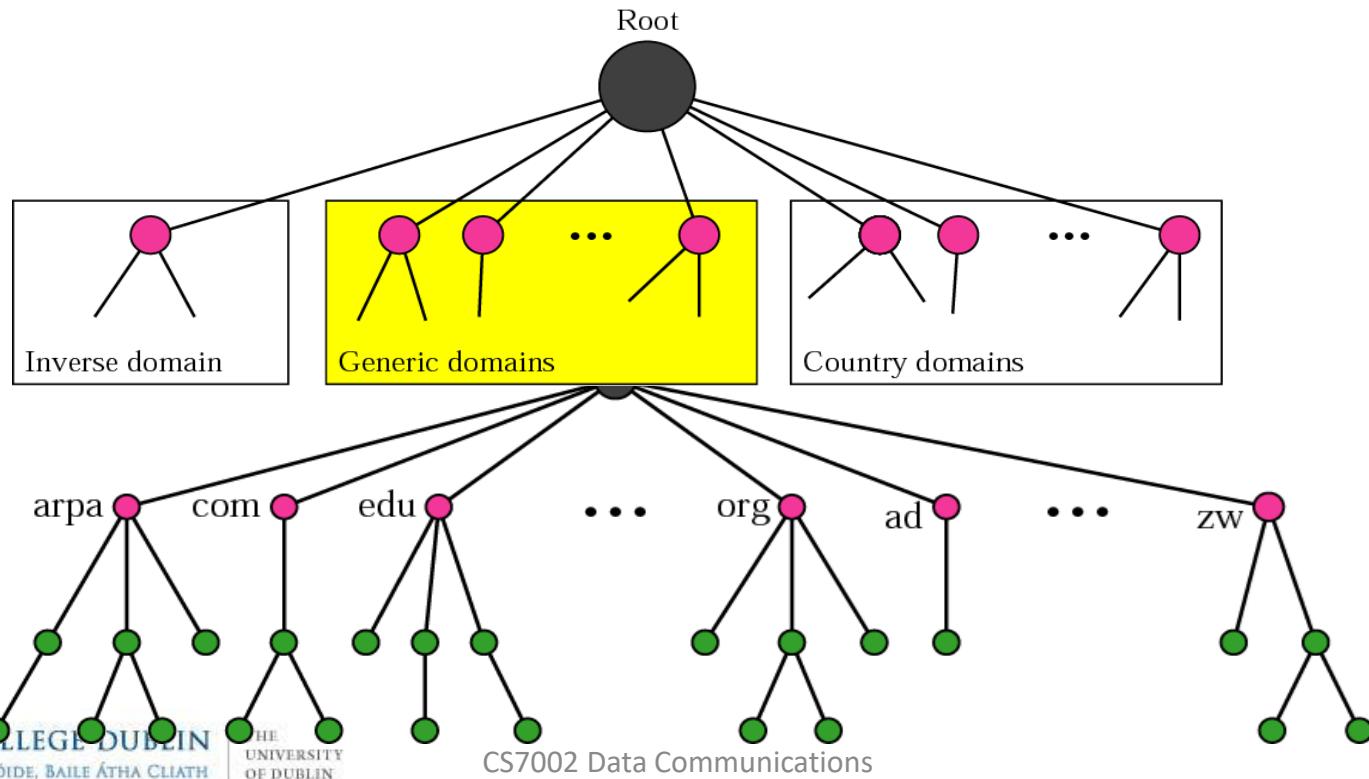
*URL = Uniform Resource Locator



Domain Name Space

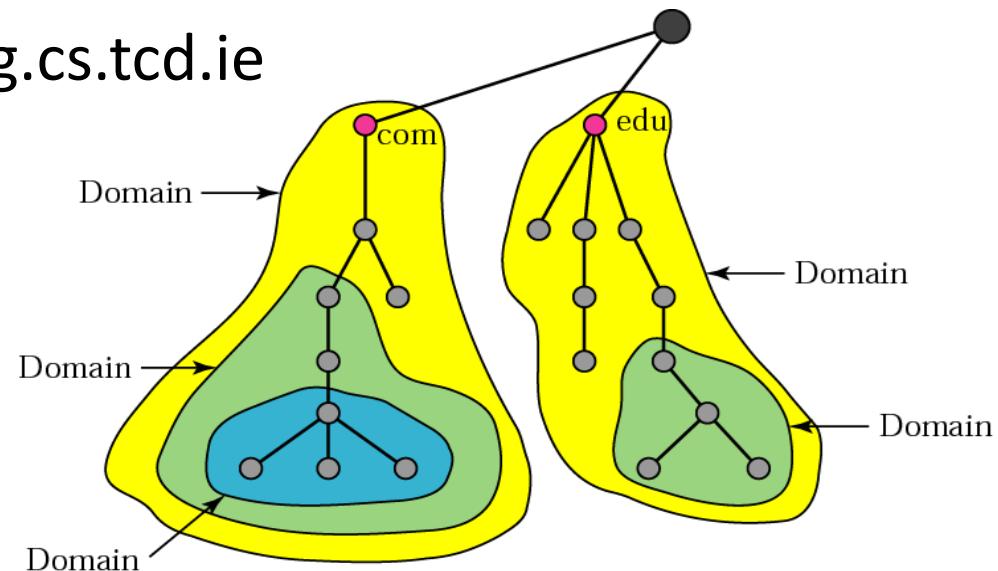
- Association between names and IP addresses

www.dsg.scss.tcd.ie - 134.226.36.14



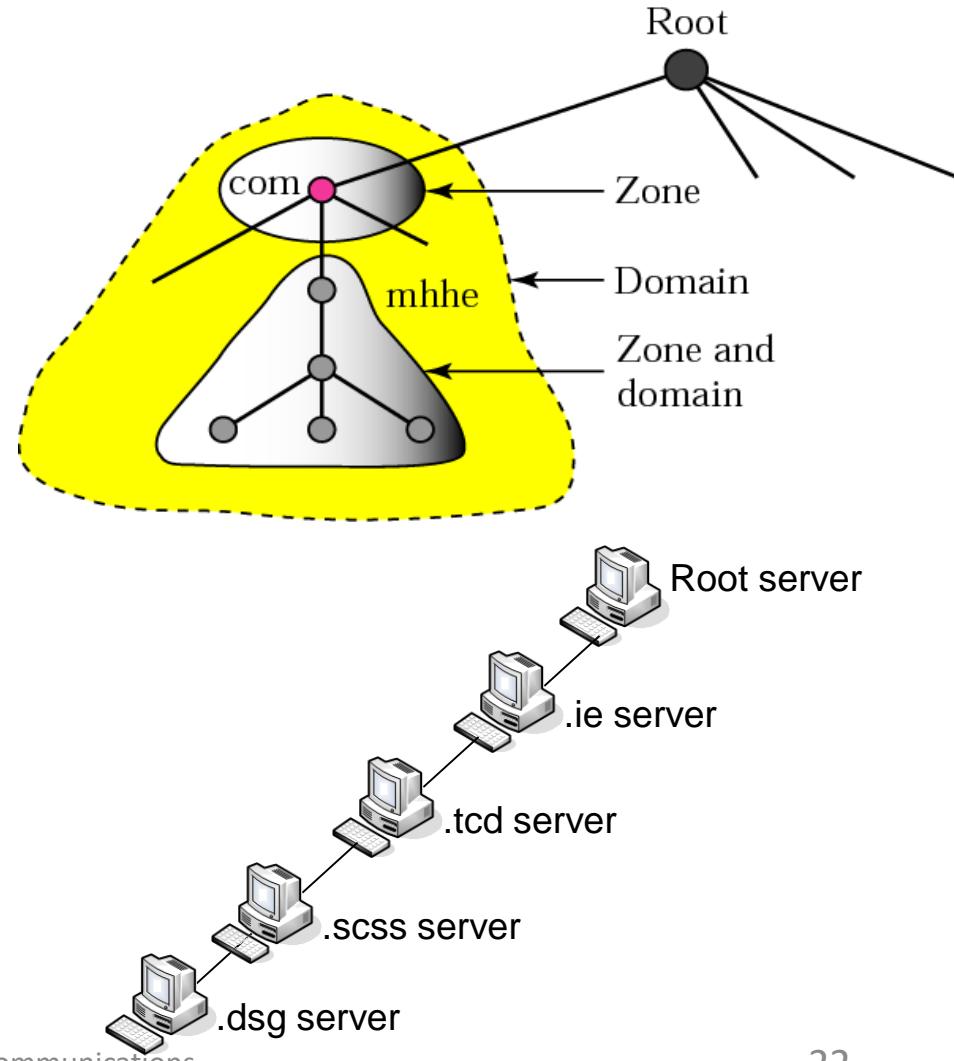
Domain Name Space

- Association between names and IP addresses
`www.dsg.scss.tcd.ie` - 134.226.36.14
- Each domain may contain a number of sub-domains e.g.
tcd.ie contains cs.tcd.ie, mee.tcd.ie
cs.tcd.ie contains dsg.cs.tcd.ie



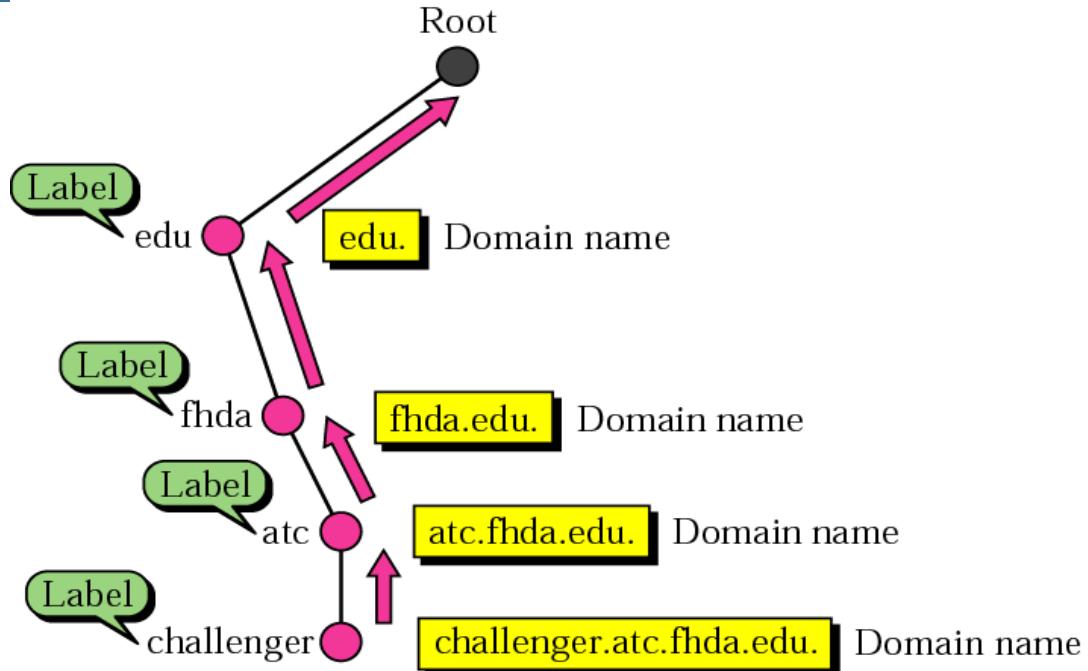
Hierarchy of Name Servers

- Every zone has a DNS server
- DNS server maintain lists of
 - Nodes in the zone
 - References to servers of zones underneath it



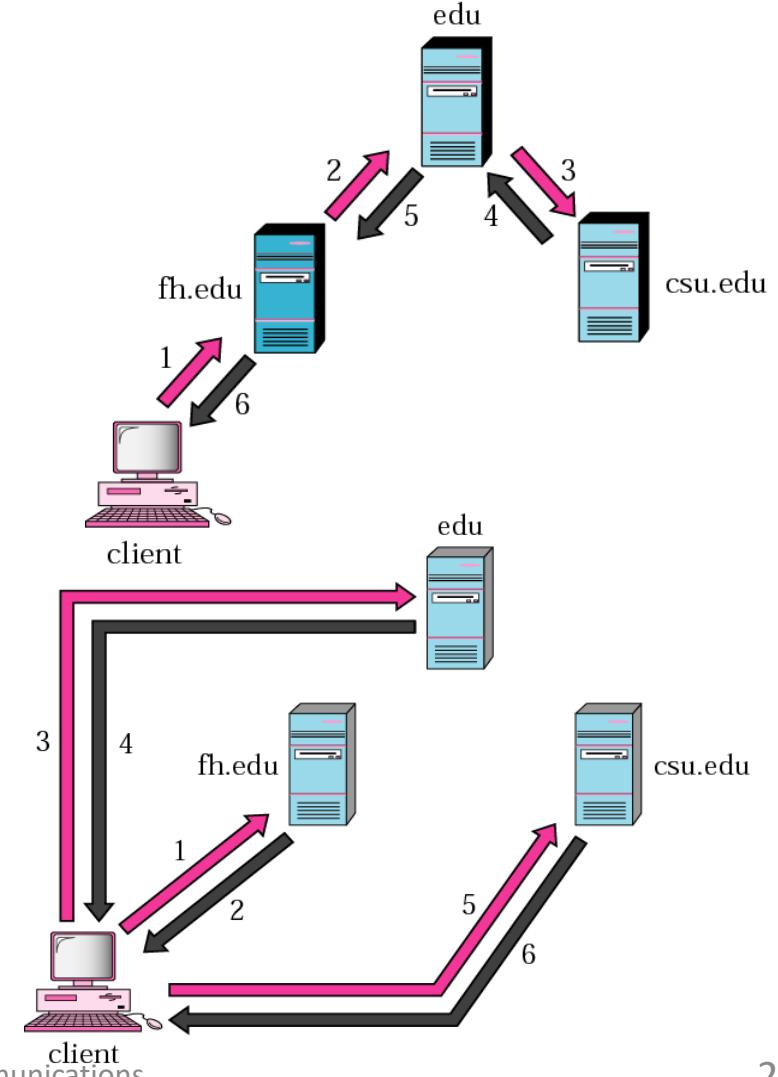
Domain Names and Labels

- A domain name consist of a number of labels



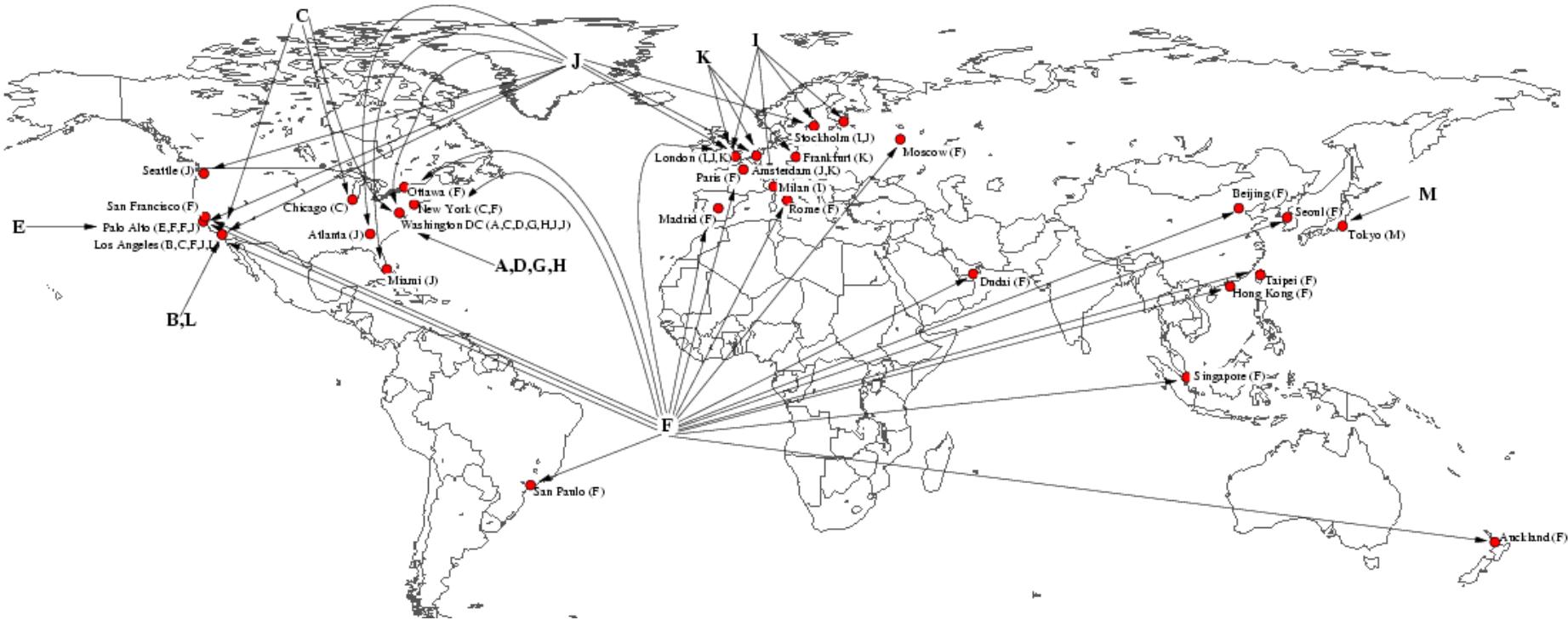
Name Resolution

- Recursive resolution



- Iterative resolution

DNS Root Servers - Anycast



root.zone file

e.dns.id.	172800	IN	A	103.19.177.177
e.dns.id.	172800	IN	AAAA	2001:df5:4000:4:0:0:0:4
ie.	172800	IN	NS	a.ns.ie.
ie.	172800	IN	NS	b.ns.ie.
ie.	172800	IN	NS	c.ns.ie.
ie.	172800	IN	NS	d.ns.ie.
ie.	172800	IN	NS	e.ns.ie.
ie.	172800	IN	NS	f.ns.ie.
ie.	172800	IN	NS	g.ns.ie.
ie.	172800	IN	NS	h.ns.ie.
IE.	86400	IN	DS	25105 8 2 3883D06014FA40518A53C70442C3601A271C0F96
IE.	86400	IN	RRSIG	DS 8 1 86400 20151206050000 20151126040000 62530 .
hHchxthV1+mIjN7sPVl27PSK040Jkegzc2Pib9+4q6bXYU3L6JPS4oXhsBqhSEA/WP7MvWemL0hSiETvuo3b8CAoMr0oTQnspt				
ie.	86400	IN	NSEC	ifm. NS DS RRSIG NSEC
ie.	86400	IN	RRSIG	NSEC 8 1 86400 20151206050000 20151126040000 62530 .
QXF5lQuk4H1casAa0GTKv2Mueizyb8p06x3RU2BtQBG609nhU9dPHIN8AA6NTQaUTleBBwAaAF3aUh37Q6r2K6+x8gsj46nxsl				
a.ns.ie.	172800	IN	A	77.72.72.44
a.ns.ie.	172800	IN	AAAA	2a01:4b0:0:0:0:0:0:3
b.ns.ie.	172800	IN	A	77.72.72.34
b.ns.ie.	172800	IN	AAAA	2a01:4b0:0:0:0:0:0:2
c.ns.ie.	172800	IN	A	194.146.106.98
c.ns.ie.	172800	IN	AAAA	2001:67c:1010:25:0:0:0:53
d.ns.ie.	172800	IN	A	77.72.229.245
d.ns.ie.	172800	IN	AAAA	2a01:3f0:0:309:0:0:0:53
e.ns.ie.	172800	IN	A	199.19.2.1
e.ns.ie.	172800	IN	AAAA	2001:500:93:0:0:0:0:1
f.ns.ie.	172800	IN	A	199.19.3.1



.ie Servers

Name Servers

Host Name	IP Address(es)
e.ns.ie	199.19.2.1 2001:500:93:0:0:0:0:1
b.ns.ie	77.72.72.34 2a01:4b0:0:0:0:0:2
g.ns.ie	192.111.39.100 2001:7c8:2:a:0:0:64
c.ns.ie	194.146.106.98 2001:67c:1010:25:0:0:53
d.ns.ie	77.72.229.245 2a01:3f0:0:309:0:0:53
f.ns.ie	199.19.3.1 2001:500:95:0:0:0:1
a.ns.ie	77.72.72.44 2a01:4b0:0:0:0:0:3
h.ns.ie	192.93.0.4 2001:660:3005:1:0:0:1:2



ipconfig /all

wireless LAN adapter WiFi:

```
Connection-specific DNS Suffix . : scss.tcd.ie
Description . . . . . : Intel(R) Dual Band wireless-AC 7260
Physical Address . . . . . : 28-B2-BD-A0-C0-A3
DHCP Enabled . . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
IPv6 Address . . . . . : 2001:770:10:203:c018:a372:1daa:424f(Prefe
rred)
Temporary IPv6 Address . . . . . : 2001:770:10:203:f0c0:4f35:6fd5:90d4(Prefe
rred)
Link-local IPv6 Address . . . . . : fe80::c018:a372:1daa:424f%3(Preferred)
IPv4 Address . . . . . : 134.226.62.20(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained . . . . . : 26 November 2015 11:11:18
Lease Expires . . . . . : 26 November 2015 14:26:16
Default Gateway . . . . . : fe80::c664:13ff:fe42:7a42%3
                           134.226.62.254
DHCP Server . . . . . : 134.226.32.58
DHCPv6 IAID . . . . . : 52998845
DHCPv6 client DUID . . . . . : 00-01-00-01-1B-DD-F3-17-28-B2-BD-A0-C0-A3
DNS Servers . . . . . : 134.226.32.57
                           134.226.56.13
```

nslookup

```
C:\Users\sweber>nslookup
Default Server: challenger.cs.tcd.ie
Address: 134.226.32.57

> set type=NS
> tcd.ie
Server: challenger.cs.tcd.ie
Address: 134.226.32.57

Non-authoritative answer:
tcd.ie    nameserver = int-ns1.tcd.ie
tcd.ie    nameserver = int-ns2.tcd.ie

int-ns1.tcd.ie  internet address = 134.226.251.108
int-ns2.tcd.ie  internet address = 134.226.251.109
> scss.tcd.ie
Server: challenger.cs.tcd.ie
Address: 134.226.32.57

scss.tcd.ie      nameserver = ns2.scss.tcd.ie
scss.tcd.ie      nameserver = ns.scss.tcd.ie
ns.scss.tcd.ie  internet address = 134.226.32.58
ns.scss.tcd.ie  AAAA IPv6 address = 2001:770:10:200:e8e0:c8ff:fec5:6b63
ns2.scss.tcd.ie internet address = 134.226.56.13
ns2.scss.tcd.ie AAAA IPv6 address = 2001:770:10:200:a0dd:c1ff:fe89:ed50
```

SOA

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name

```

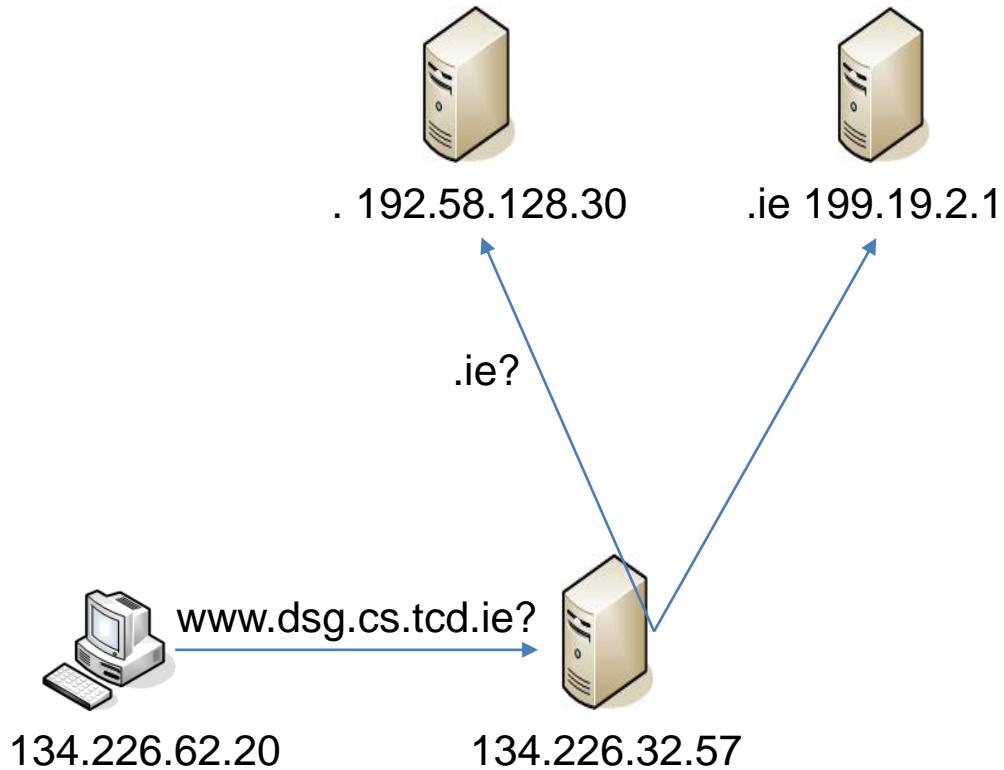
@           IN      SOA      ns1.dsg.cs.tcd.ie. dsgadmin.cs.tcd.ie. (
                           2003100801          ; Serial  yearmonthdayversion
                           7200                ; Refresh 2 hours
                           1800                ; Retry  1/2 hour
                           86400               ; Expire 1 day
                           10800              ; Minimum 3 hours
                         MX      10       relay.cs.tcd.ie.
                         IN      NS       ns1.dsg.cs.tcd.ie.
                         IN      NS       ns2.dsg.cs.tcd.ie.

dsg.cs.tcd.ie   IN      A        134.226.36.0
                           MX      10       relay.cs.tcd.ie.

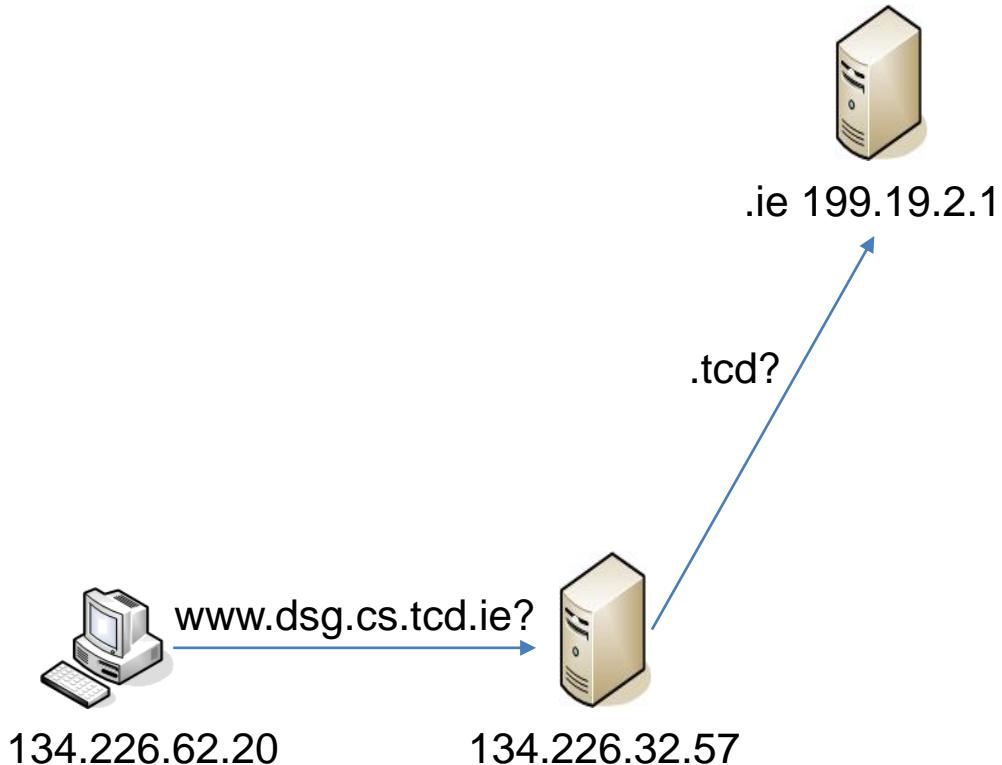
computerA IN      A        134.226.36.1
dilbert        IN      CNAME    computerA
dogbert        IN      A        134.226.36.2
                           IN      A        134.226.36.3

```

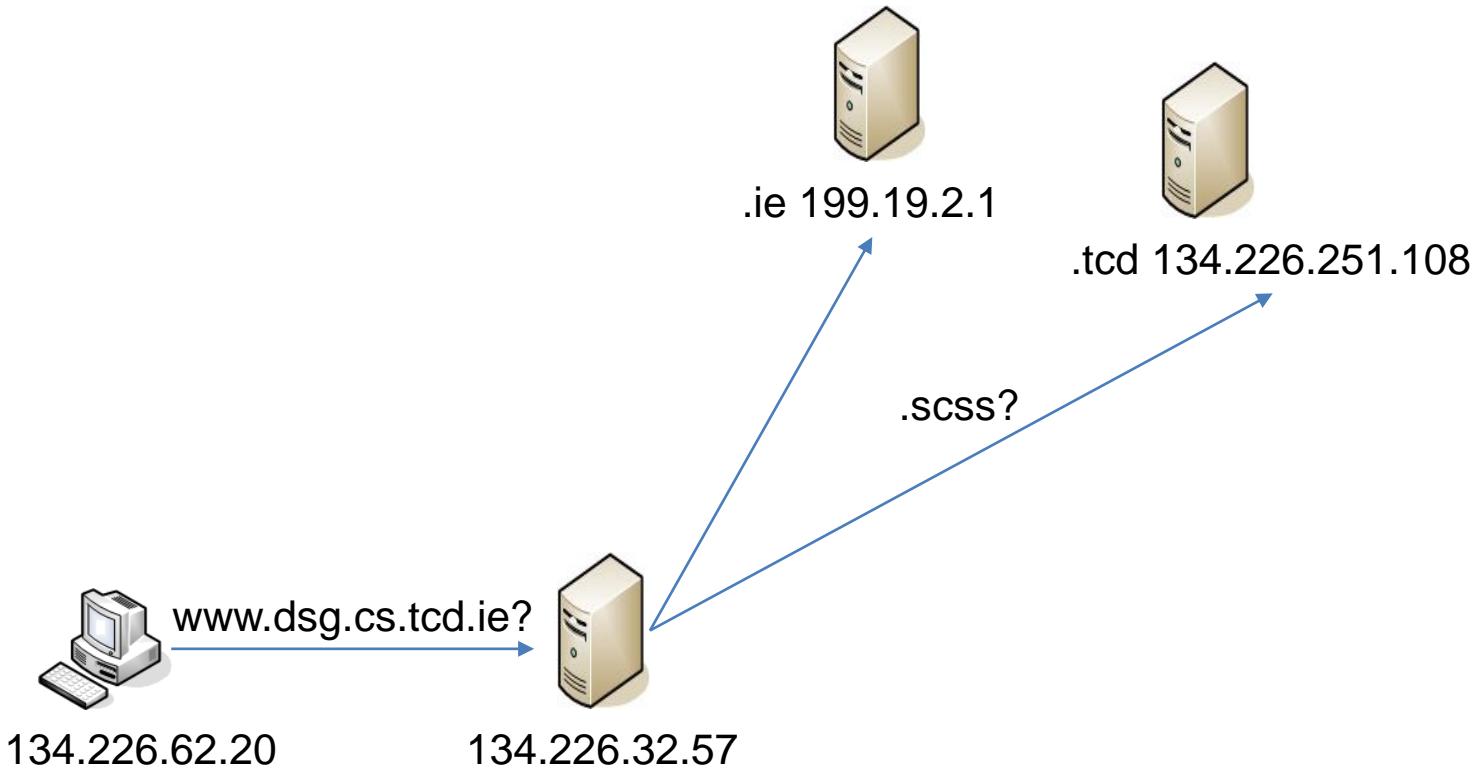
Lookup of www.dsg.scss.tcd.ie.



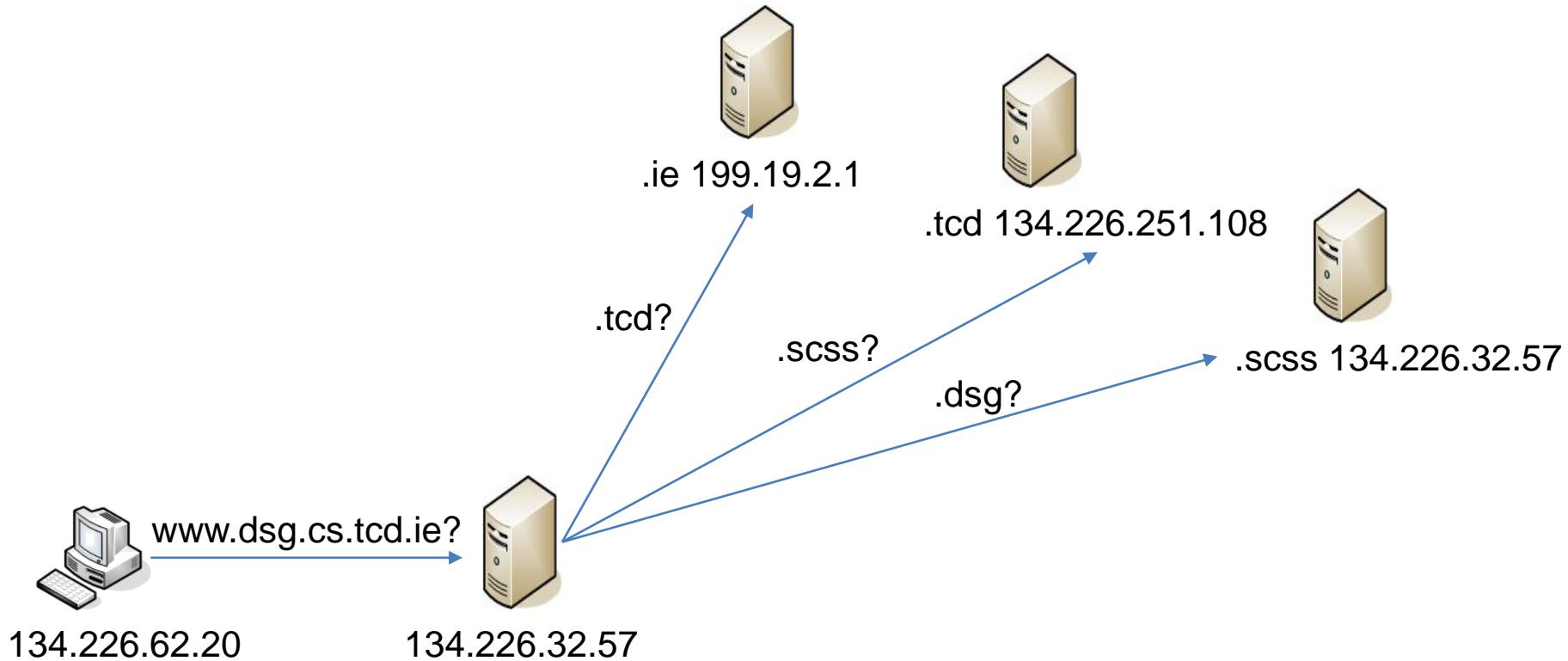
Lookup of www.dsg.scss.tcd.ie.



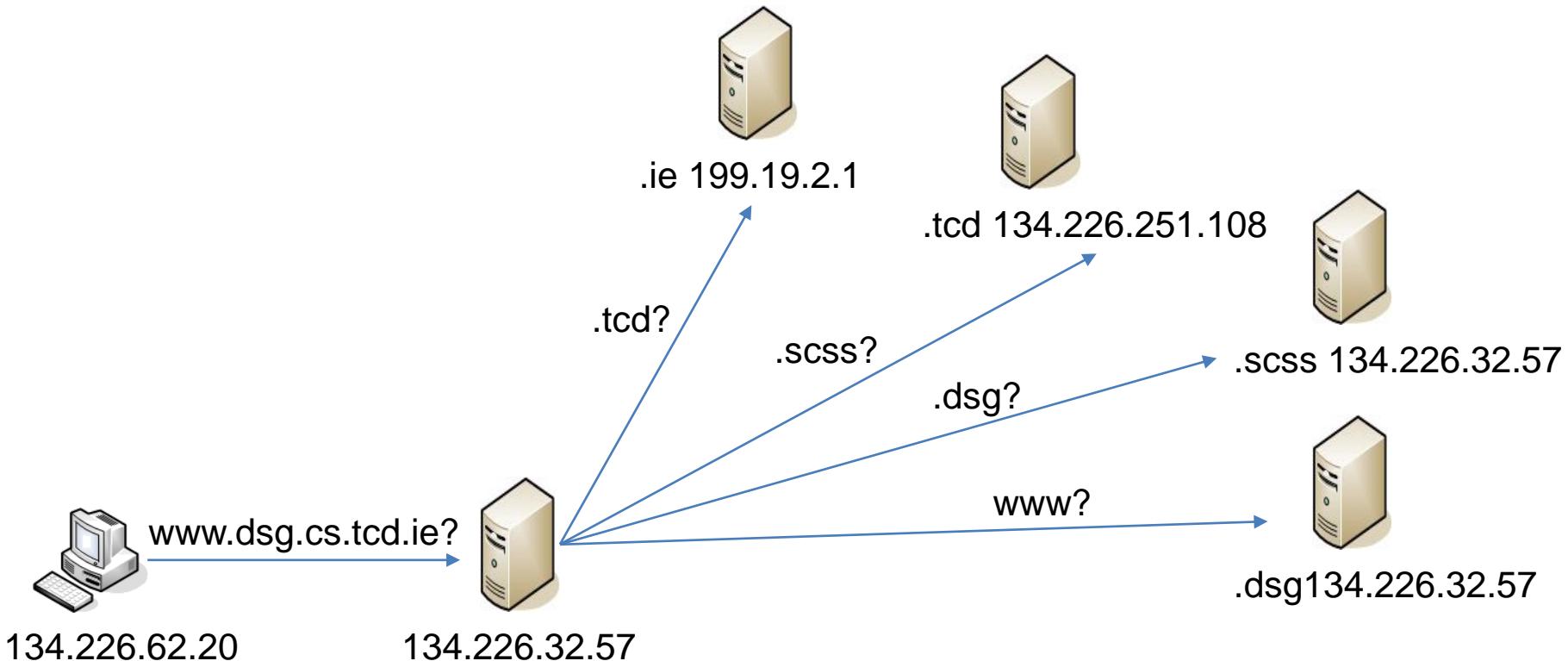
Lookup of www.dsg.scss.tcd.ie.



Lookup of www.dsg.scss.tcd.ie.

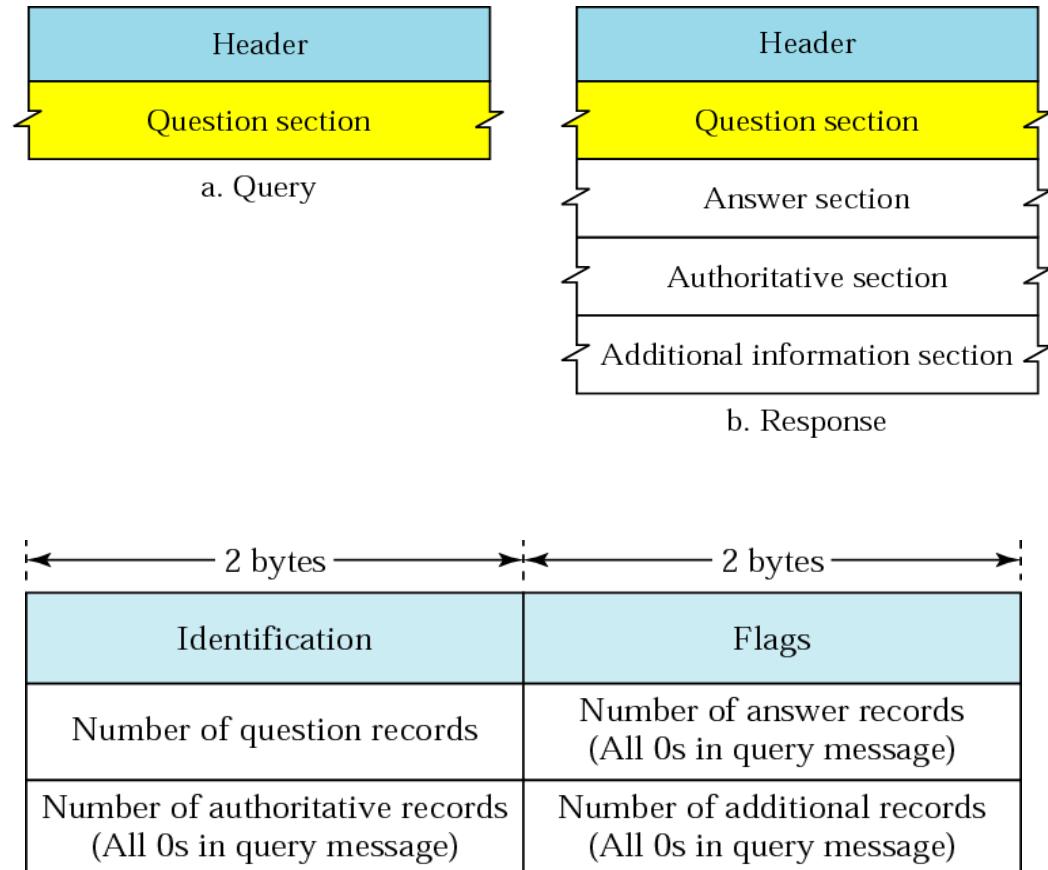


Lookup of www.dsg.scss.tcd.ie.



Query and Response Messages

- Two types of replies:
 - Authoritative answers
 - Cached or unauthoritative answers
- 6-byte header

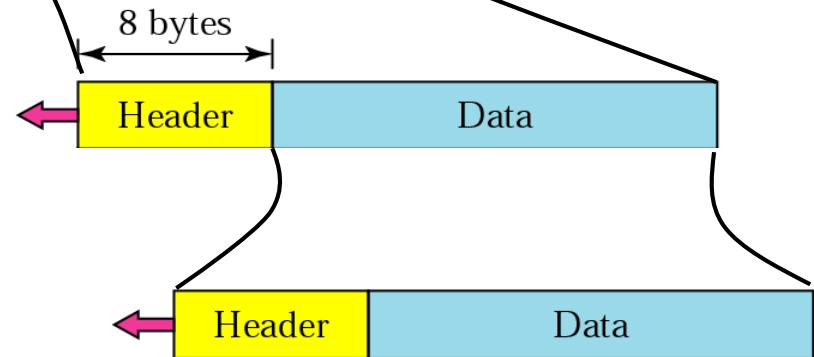


DNS Request

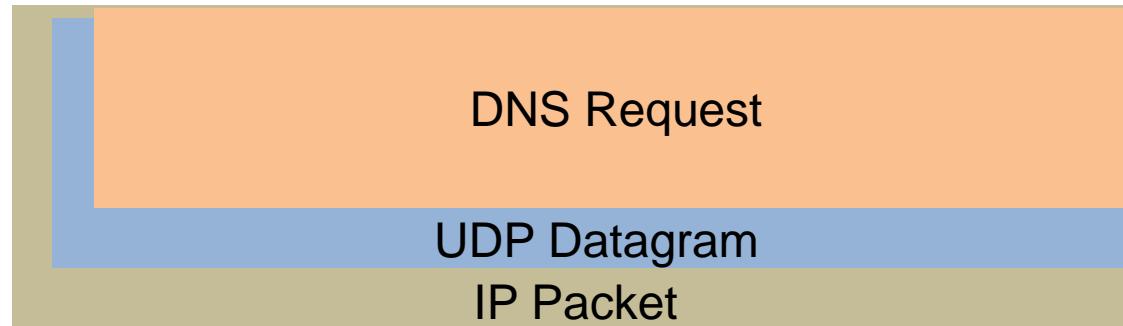
IP Packet



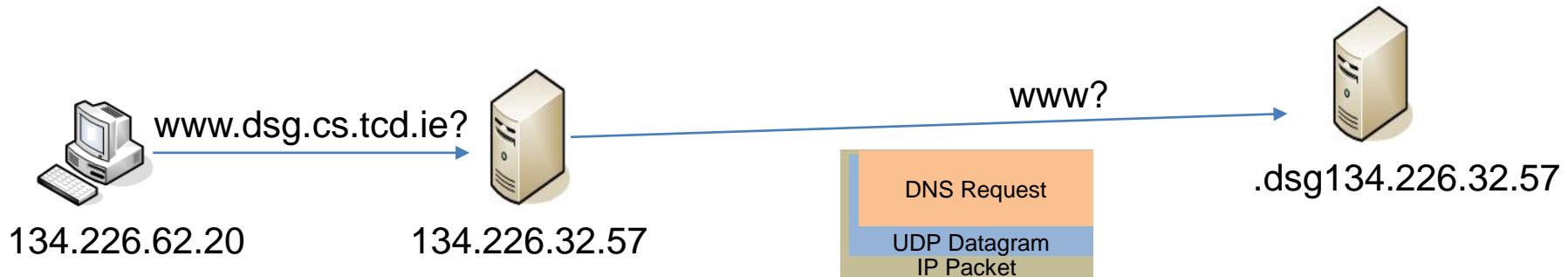
UDP Datagram



DNS Request



Lookup of www.dsg.scss.tcd.ie.



SOA

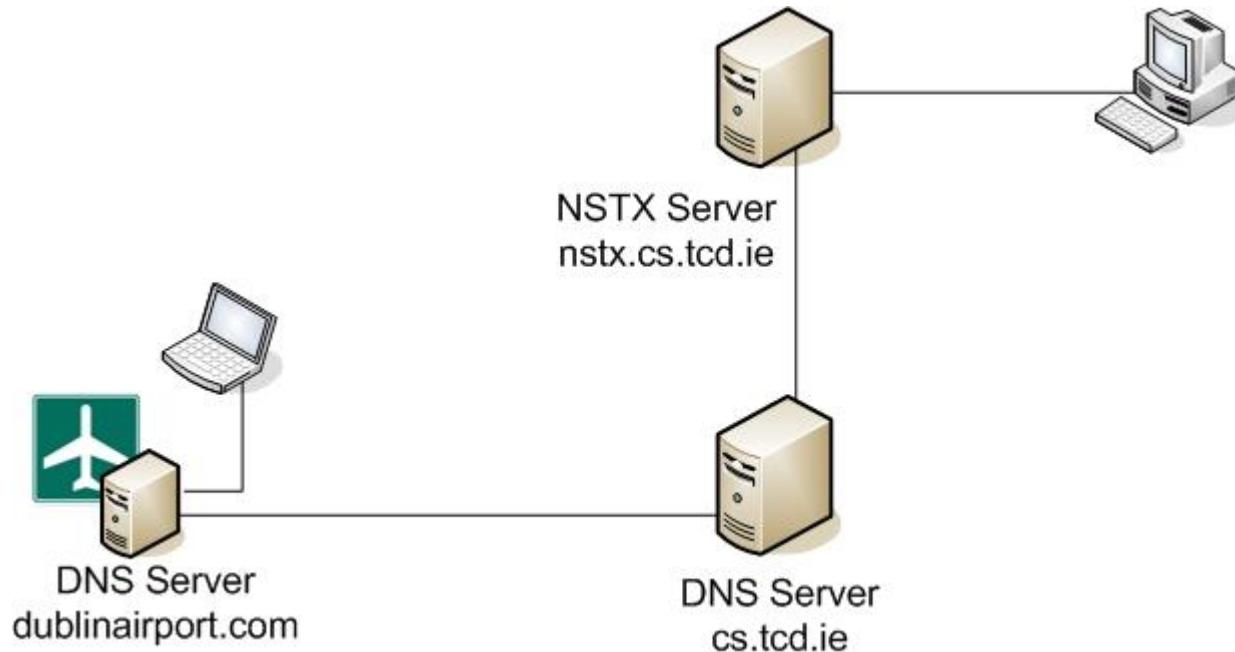
Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name

\$ORIGIN .cs.tcd.ie.

@ IN NS nstx.cs.tcd.ie.
nstx IN A 134.226.32.44

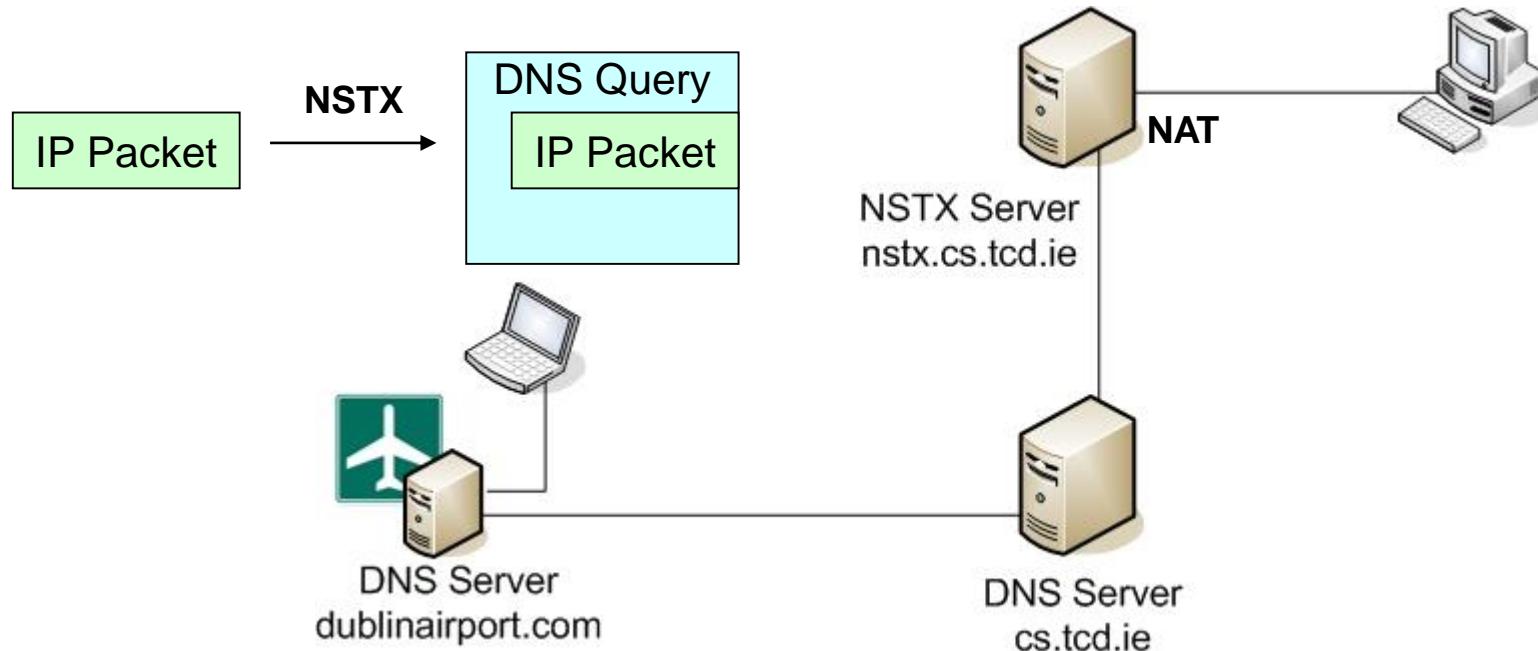


IP-over-DNS II



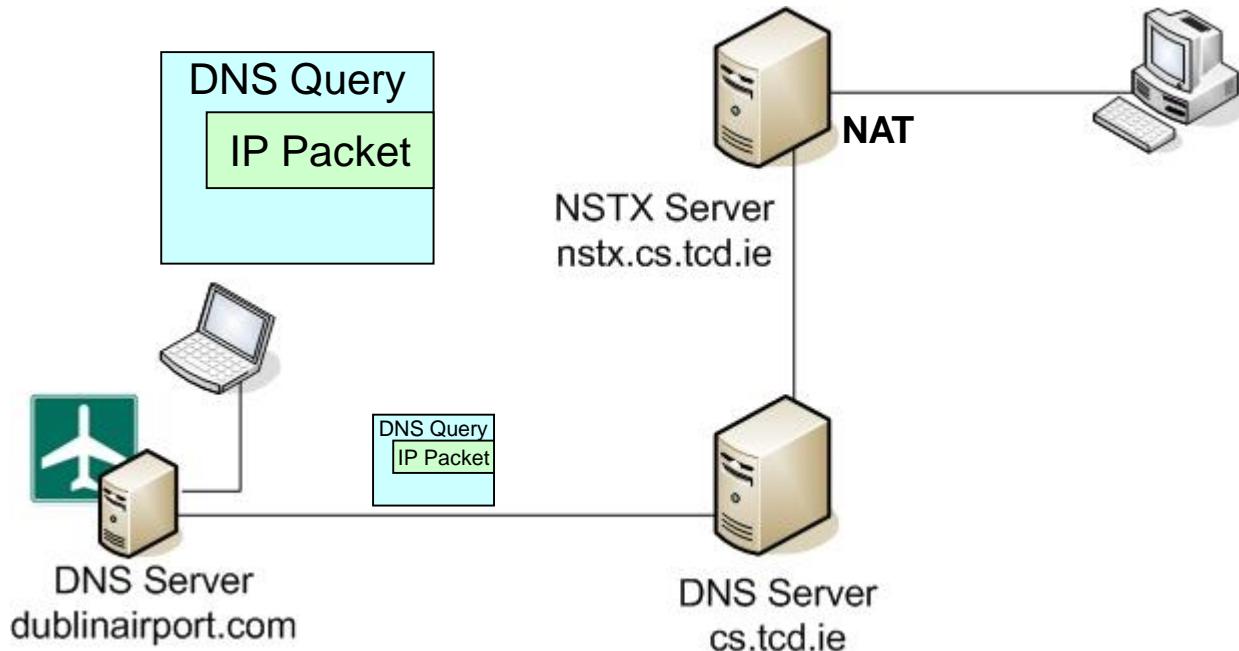
- Laptop with limited access at airport
- Encapsulate IP packet into DNS query
 - Example: NSTX, Iodine

IP-over-DNS II



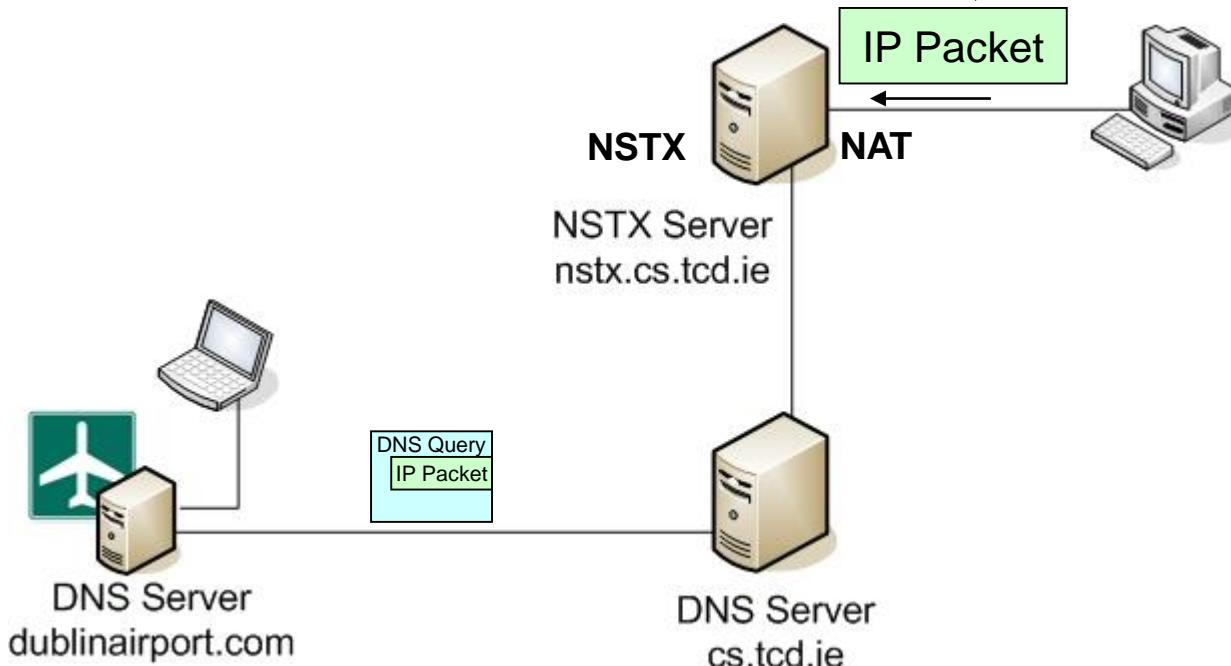
- Laptop with limited access at airport
- Encapsulate IP packet into DNS query
 - Example: NSTX, Iodine

IP-over-DNS III



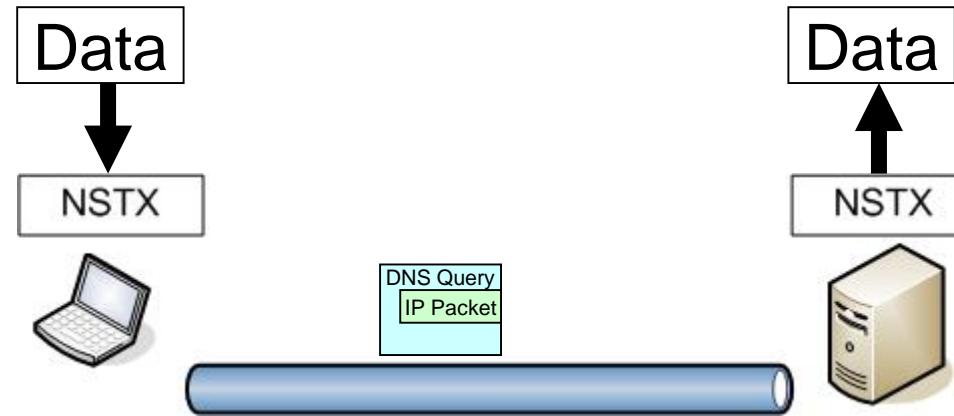
- Laptop with limited access at airport
- Encapsulate IP packet into DNS query
 - Example: NSTX, Iodine

IP-over-DNS IV



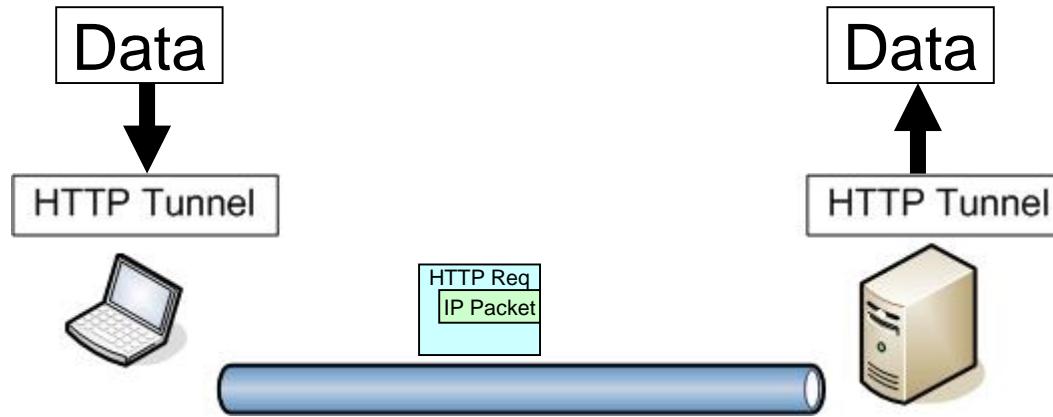
- Laptop with limited access at airport
- Encapsulate IP packet into DNS query
 - Example: NSTX, Iodine

Tunnelling I



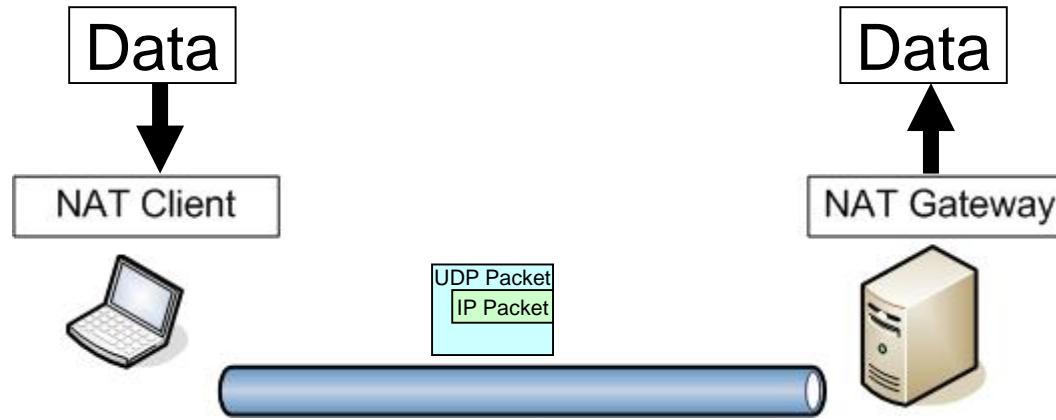
- Machine with access to Internet
- Machine with restricted access
- Both run programs that can pack and unpack data to be tunneled

Tunnelling II



- Same process: Tunnel runs on both machines to pack and unpack data
- HTTP Request can transverse proxies etc

Tunnelling III



- Same process: Client and gateway should know how to pack and unpack data
- Gateway could be any machine in College with access to machines outside



CS2031

Telecommunications II

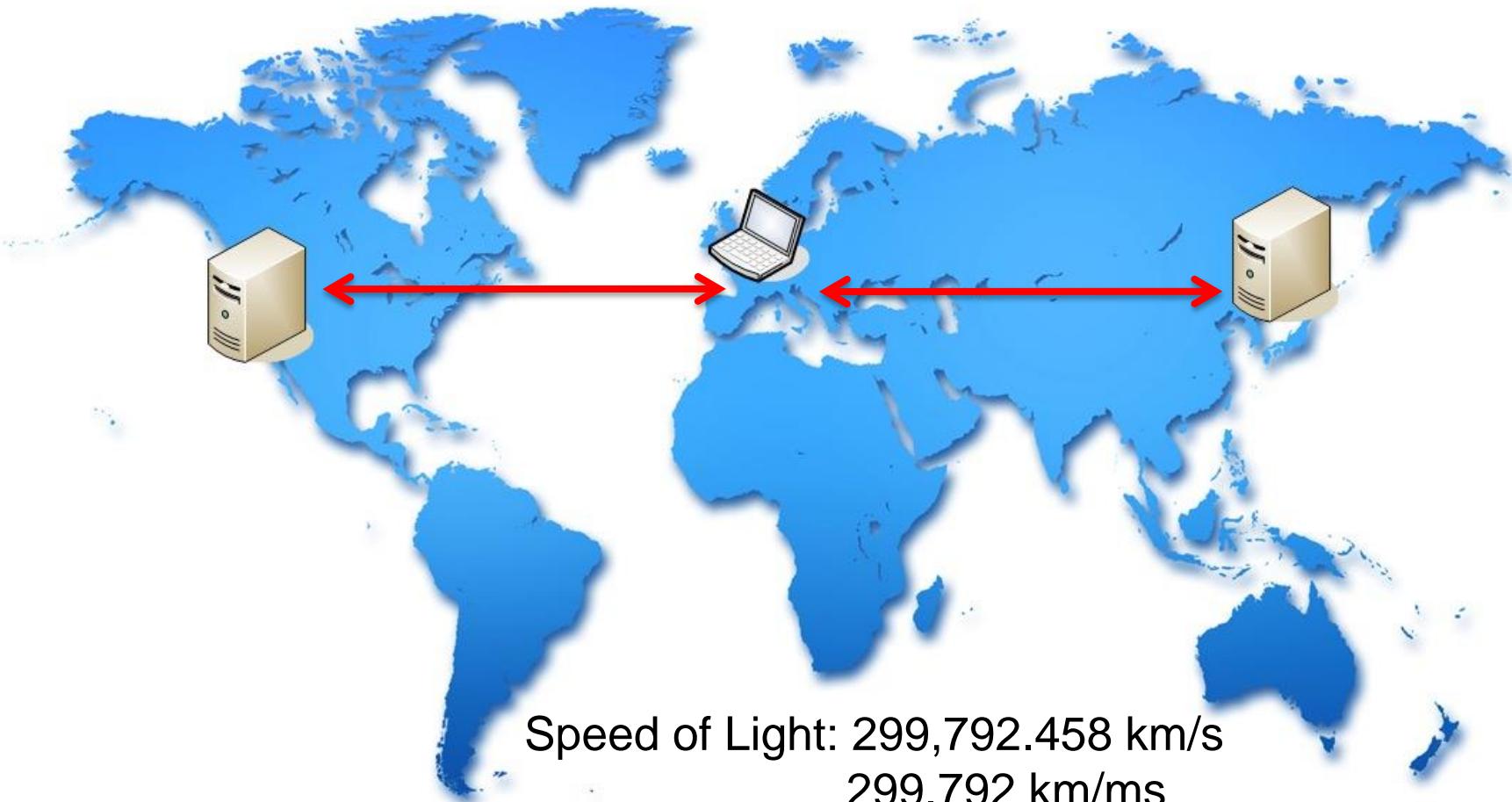
CDN



CDN - Motivation



Fixed Latency



Speed of Light: 299,792.458 km/s
299.792 km/ms
~300 km/ms

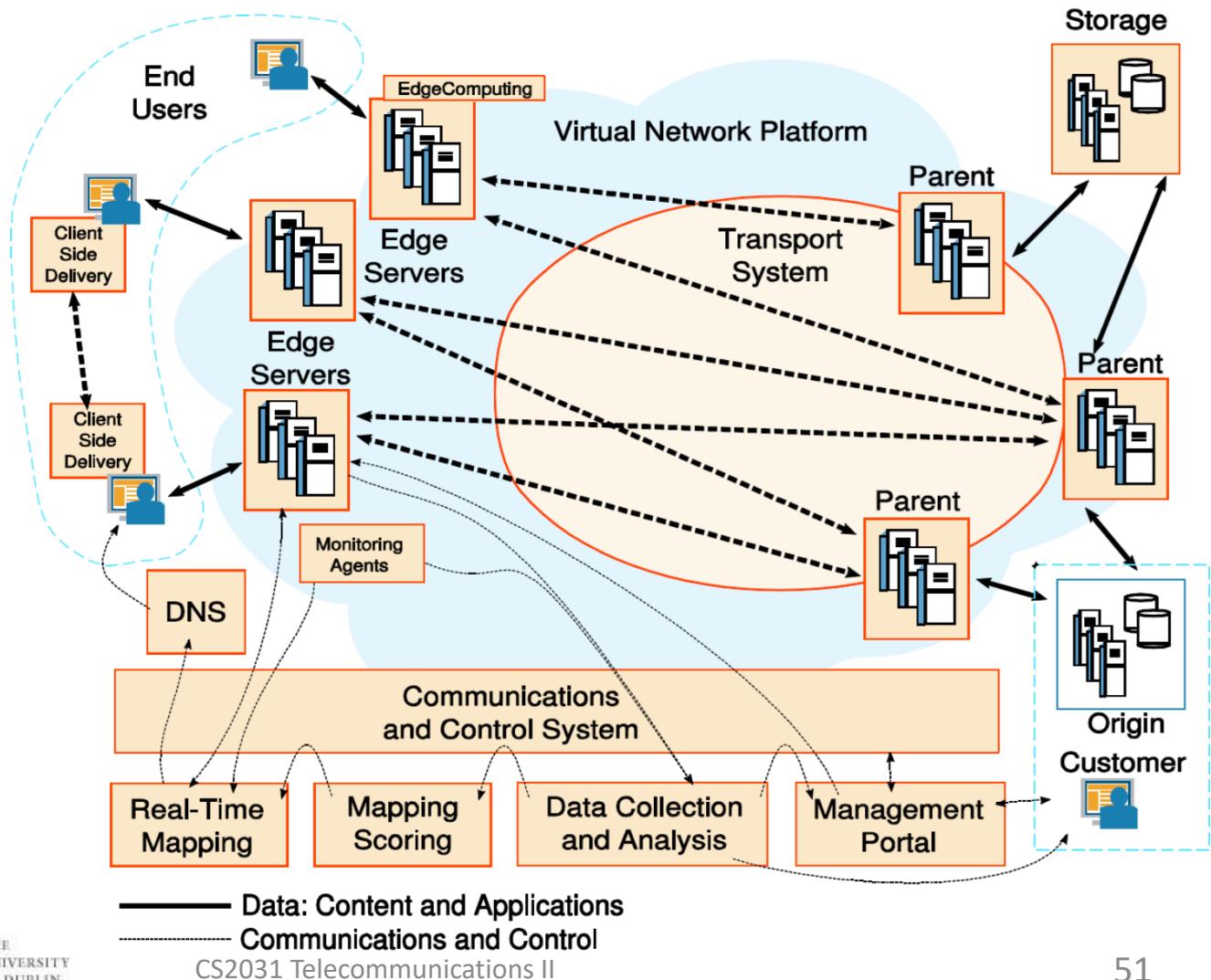
Content-Distribution Networks



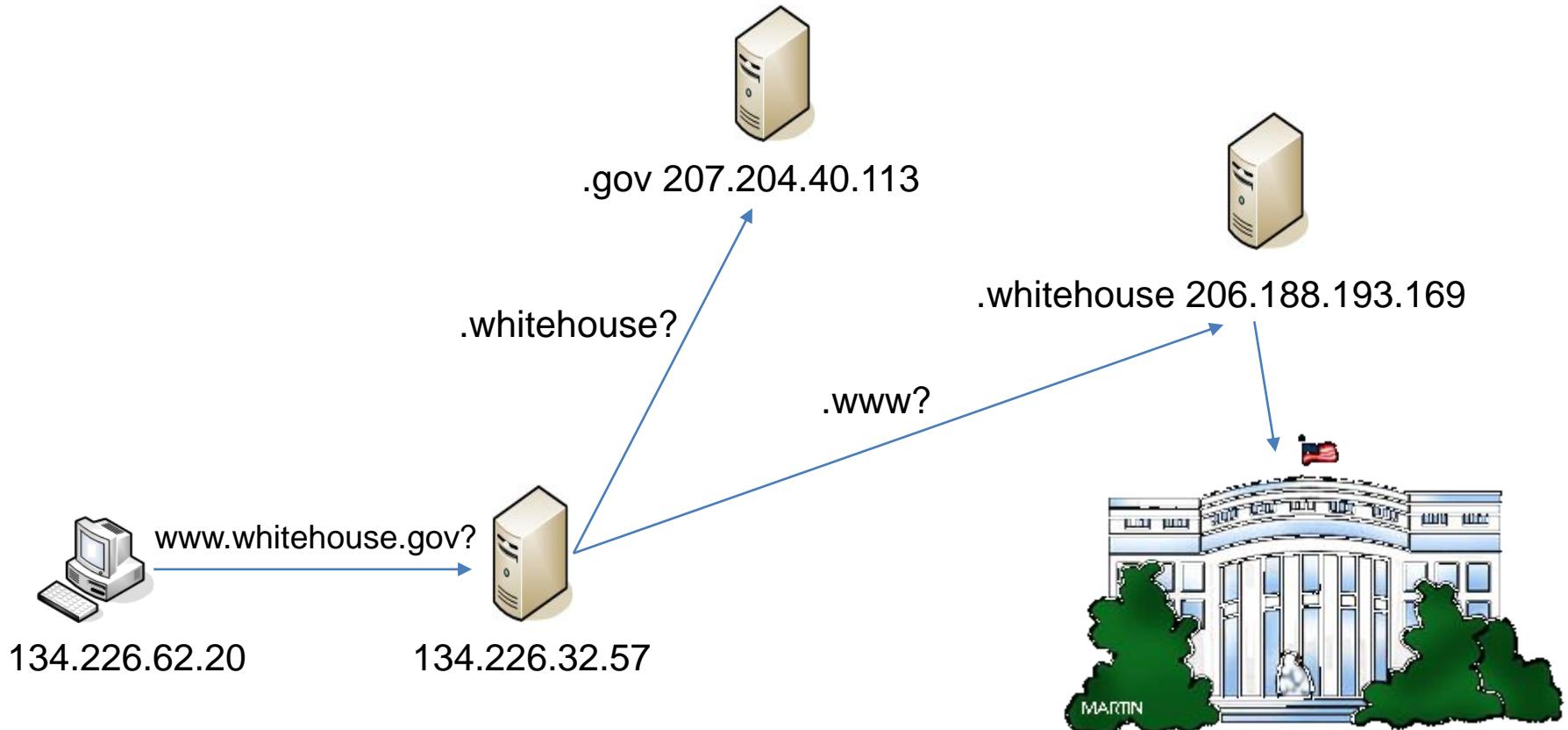
e.g. Akamai, Limelight, Amazon, Netflix, etc

Akamai Scenario

- DNS redirection
- Clusters of servers at points of presence
- Replication

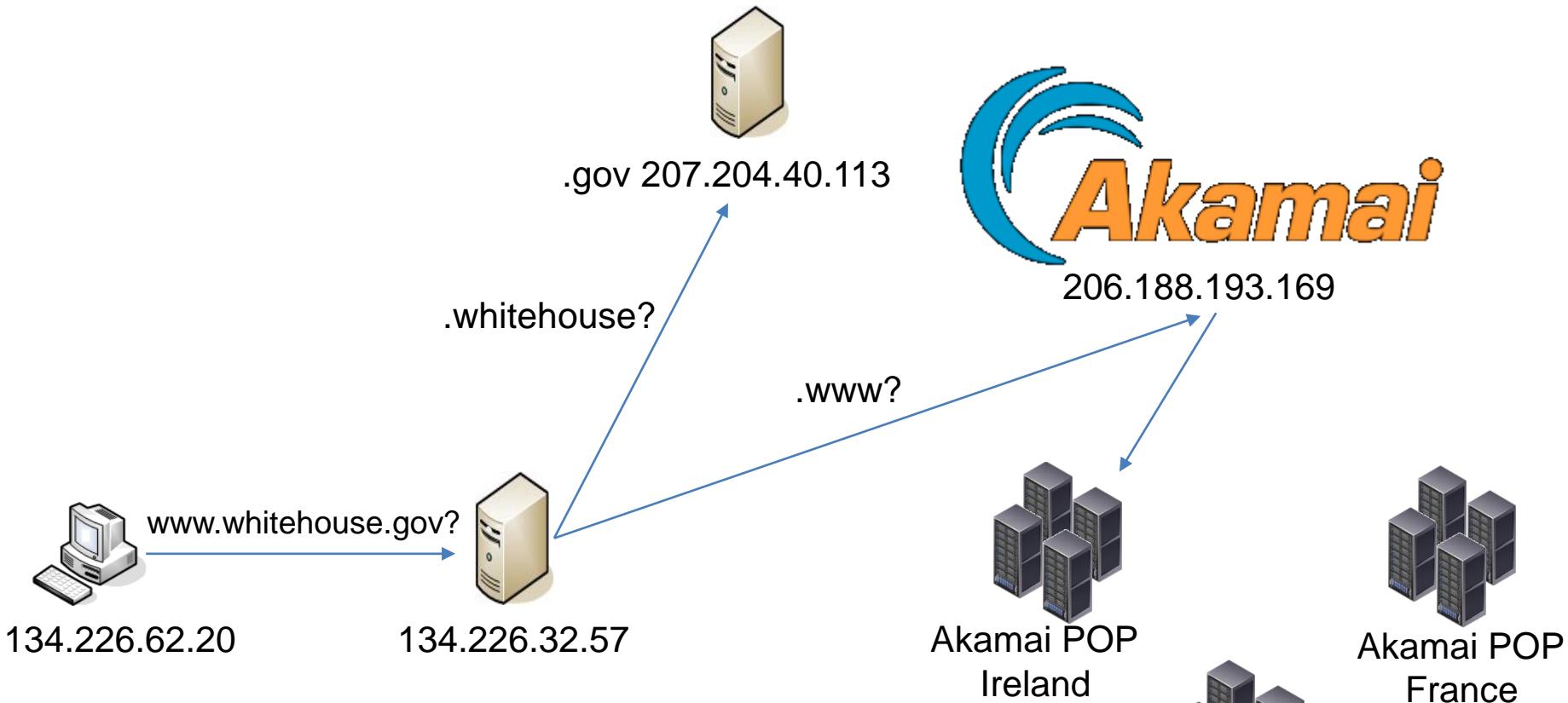


The Usual Case



- Asking for `www.whitehouse.gov`

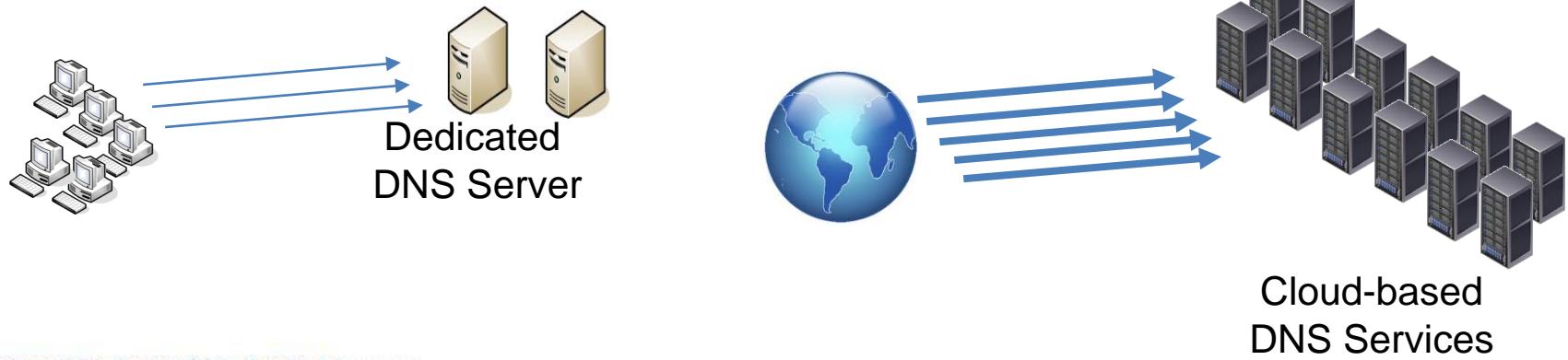
Redirection of DNS



- Asking for `www.whitehouse.gov`

Mirai Bot & The Attack on Dyn

- Mirai Bot Source published (Sept 30th)
- Attack on Dyn (Oct 21st)
- From one manufacturer:
“username: root” and “password: xc3511” hardcoded into the device firmware of a number of devices



URLs to Names to Addresses

URL

DNS

IP Address

`http://www.wiki.com/index.html`

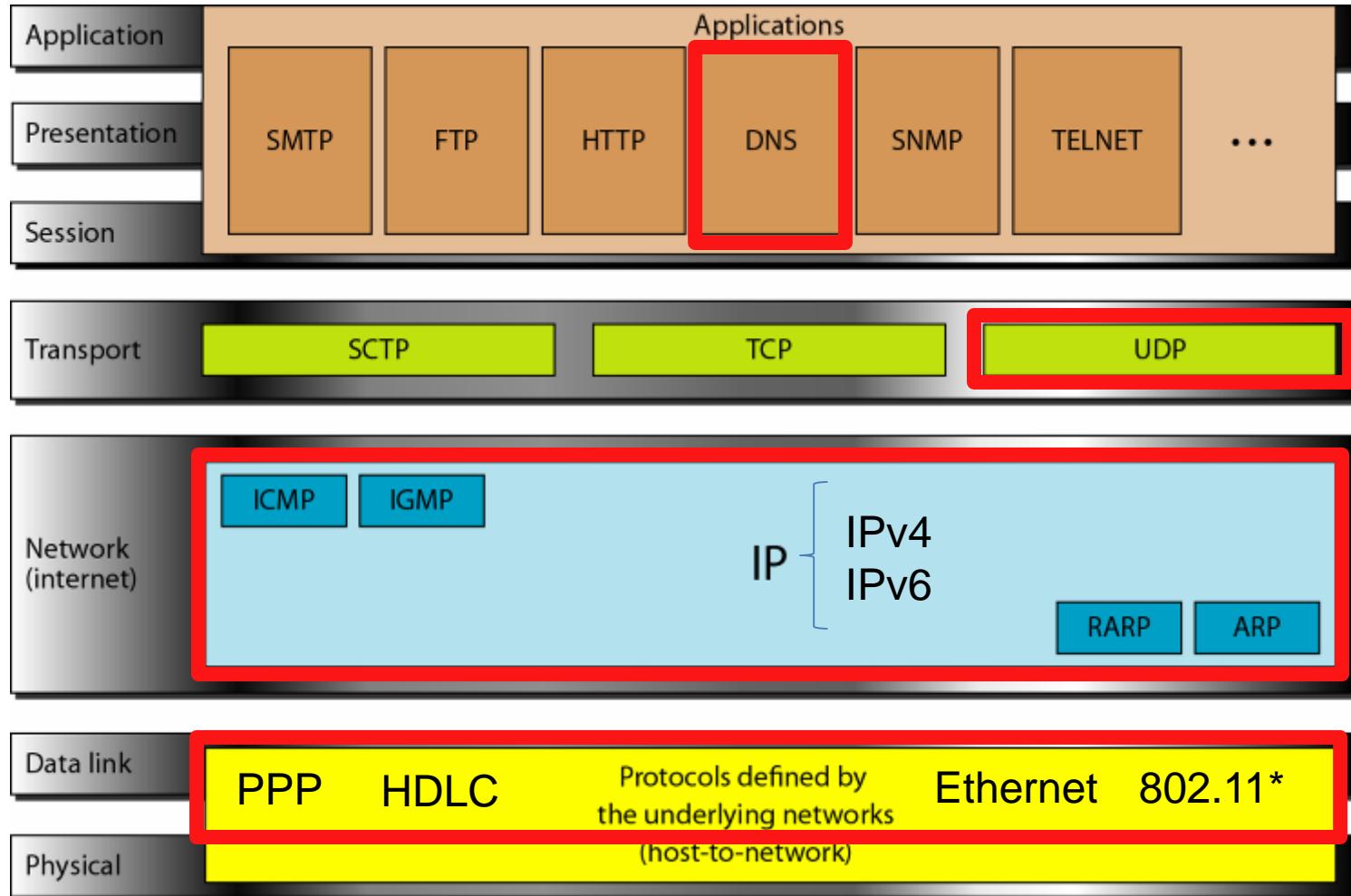
`www.wiki.com`

66.96.149.1

*URL = Uniform Resource Locator



Protocols in the OSI Model



CS Predictions

- "I think there is a world market for maybe five computers."

Thomas Watson, President of IBM, 1943

- "There is no reason anyone would want a computer in their home."

Ken Olsen, Founder of Digital Equipment Corporation, 1977

- "I predict the Internet in 1996 will catastrophically collapse."

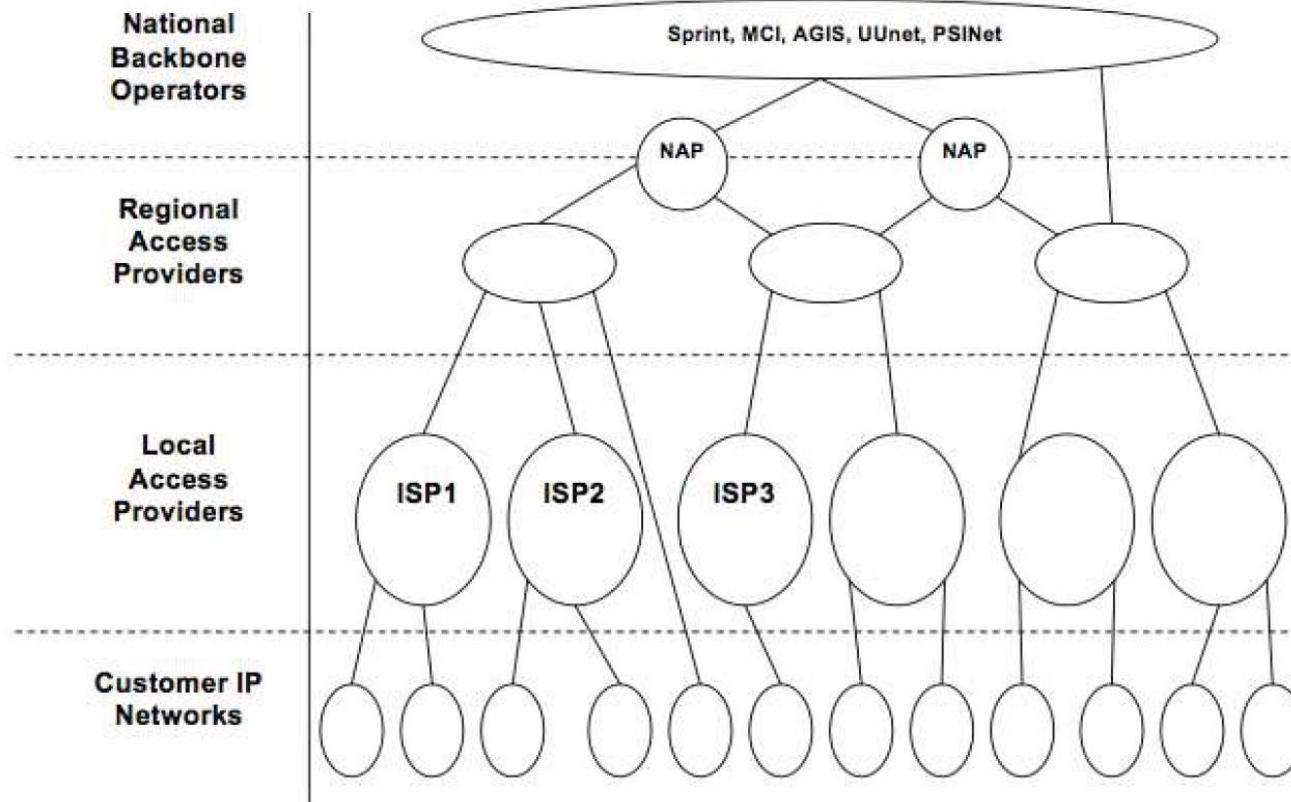
Robert Metcalfe, 1995

- "IPv6 is dead."

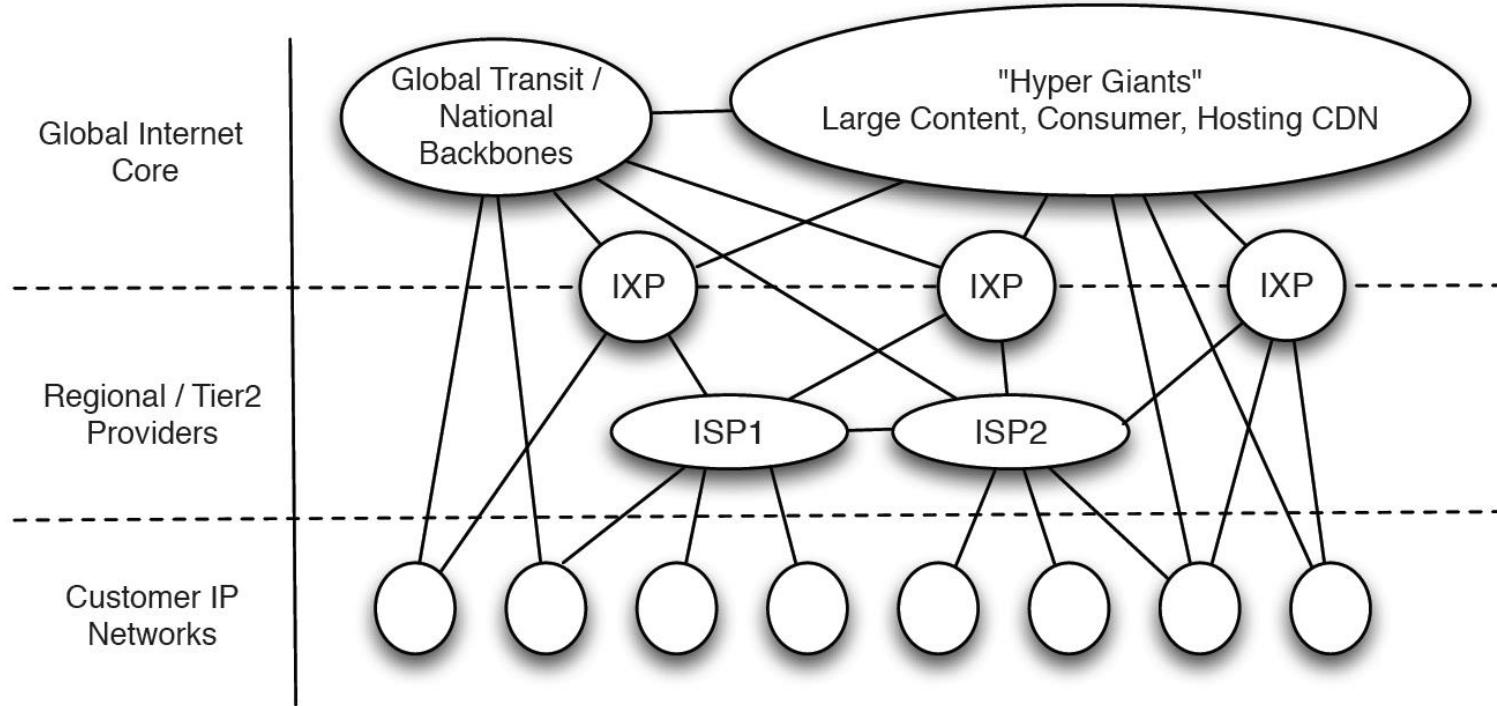
David Cheriton, 1999



Traditional Logical Internet Topology

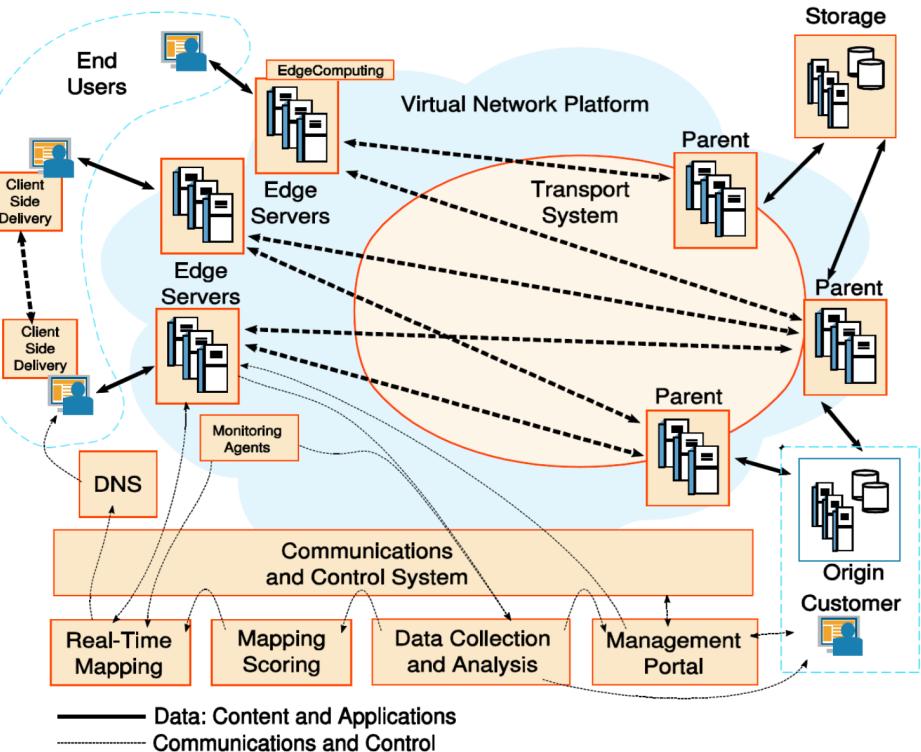
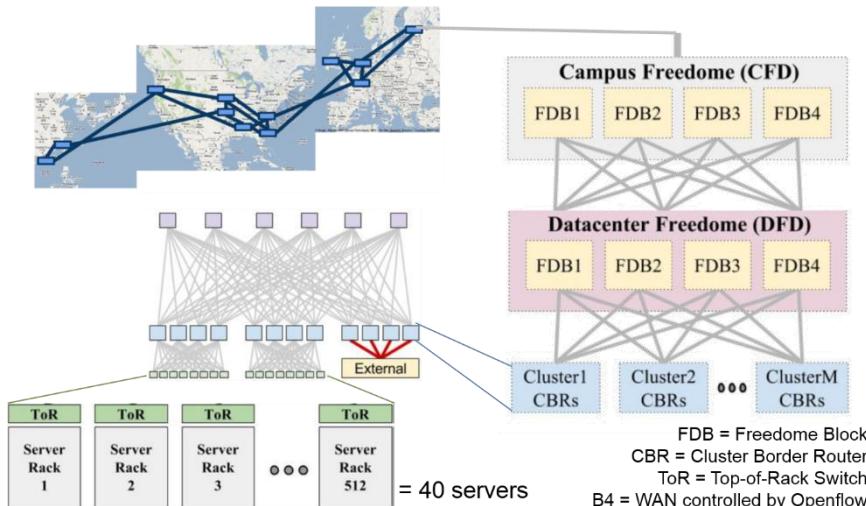


Emerging Logical Internet Topology



- According to a statement by Craig Labovitz in 2014:
 - 30 Entities created 50% of the traffic in the US at peak time

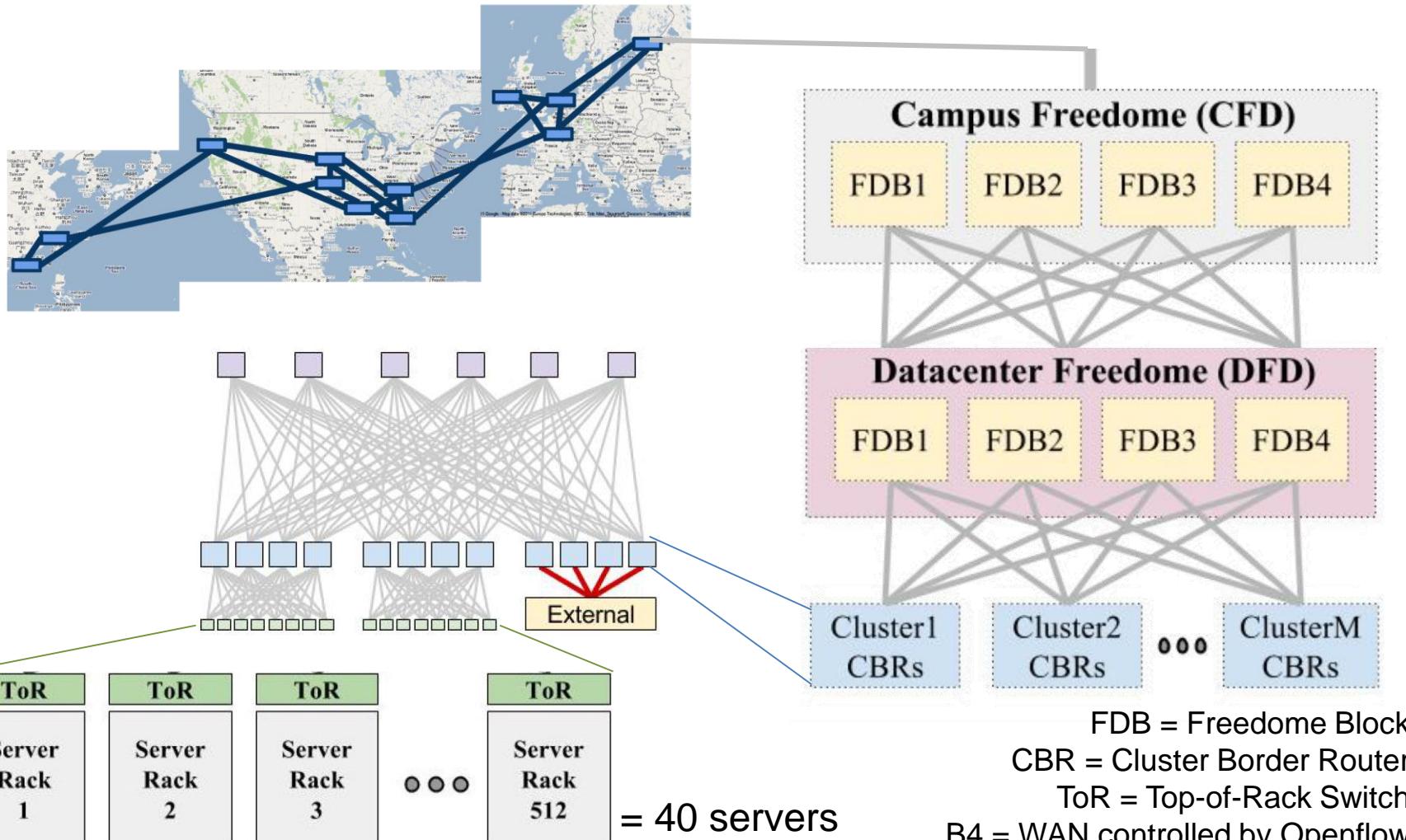
Hyper-Giants



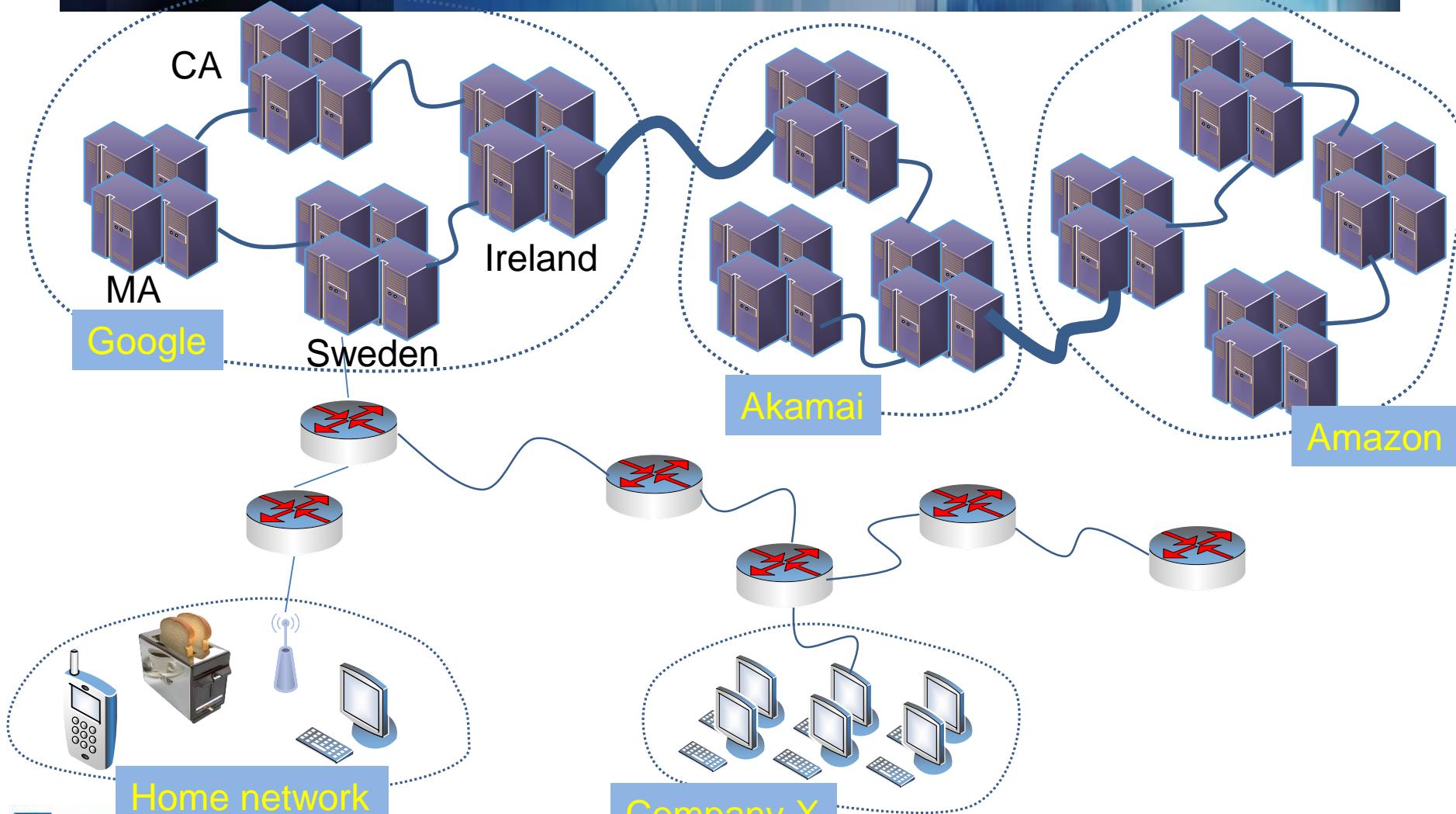
- Google

- Akamai

Google's B4 to Jupiter

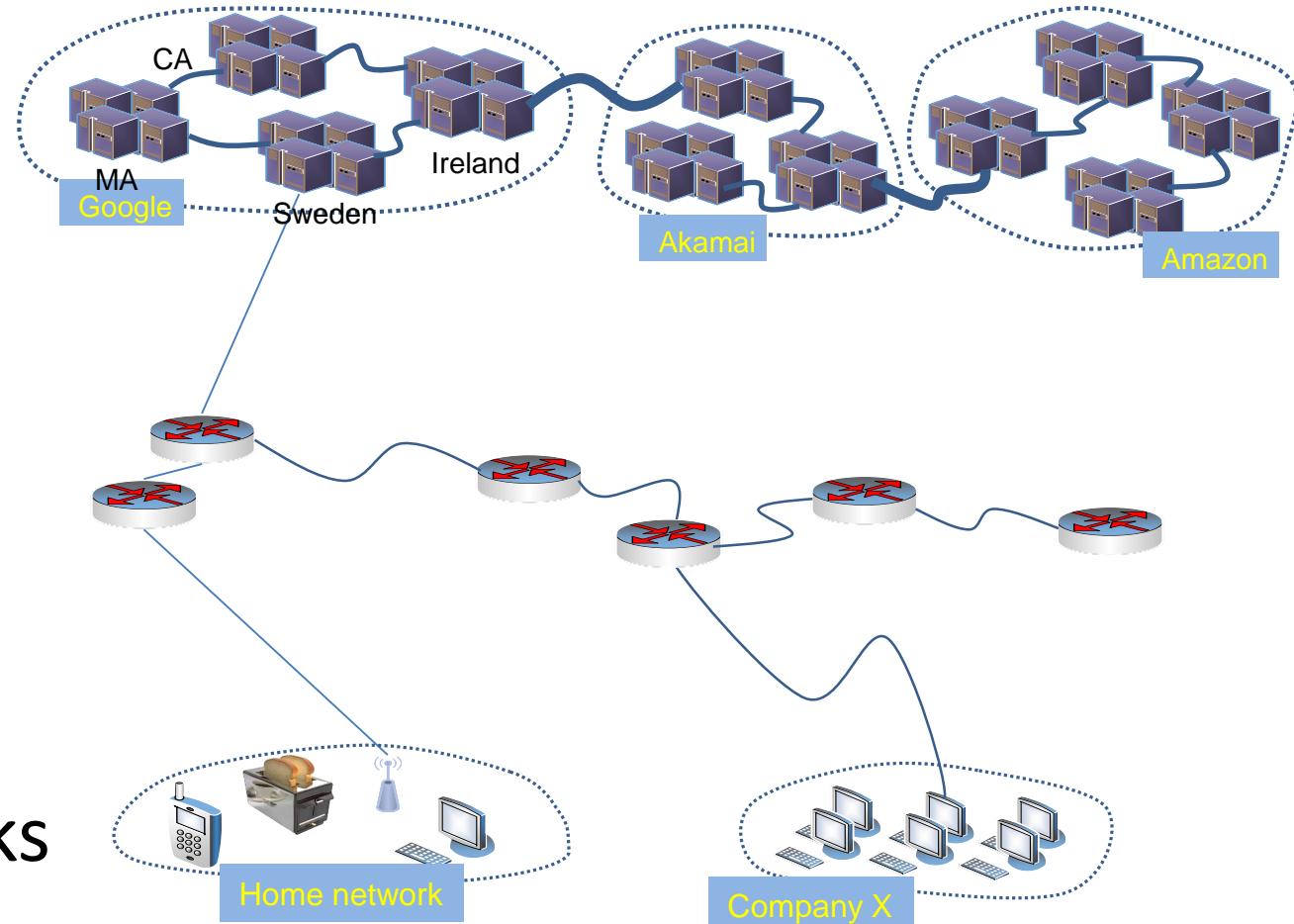


Datacentres at the Core?



My view of “future”* networking

- Sets of Datacentres



- Traditional Internet

- Edge networks

*or current?



Overview

- Link Layer
- Network Layer
 - Addressing
 - Address Resolution (ARP)
 - Fragmentation
 - Intra-AS Routing
 - Distance Vector
 - Link State
 - Multicast Routing
 - IPv6
- Transport Layer
 - UDP
 - DNS
- ATM/MPLS
- Software-Defined Networking / Openflow
- CLOS / Fat-tree



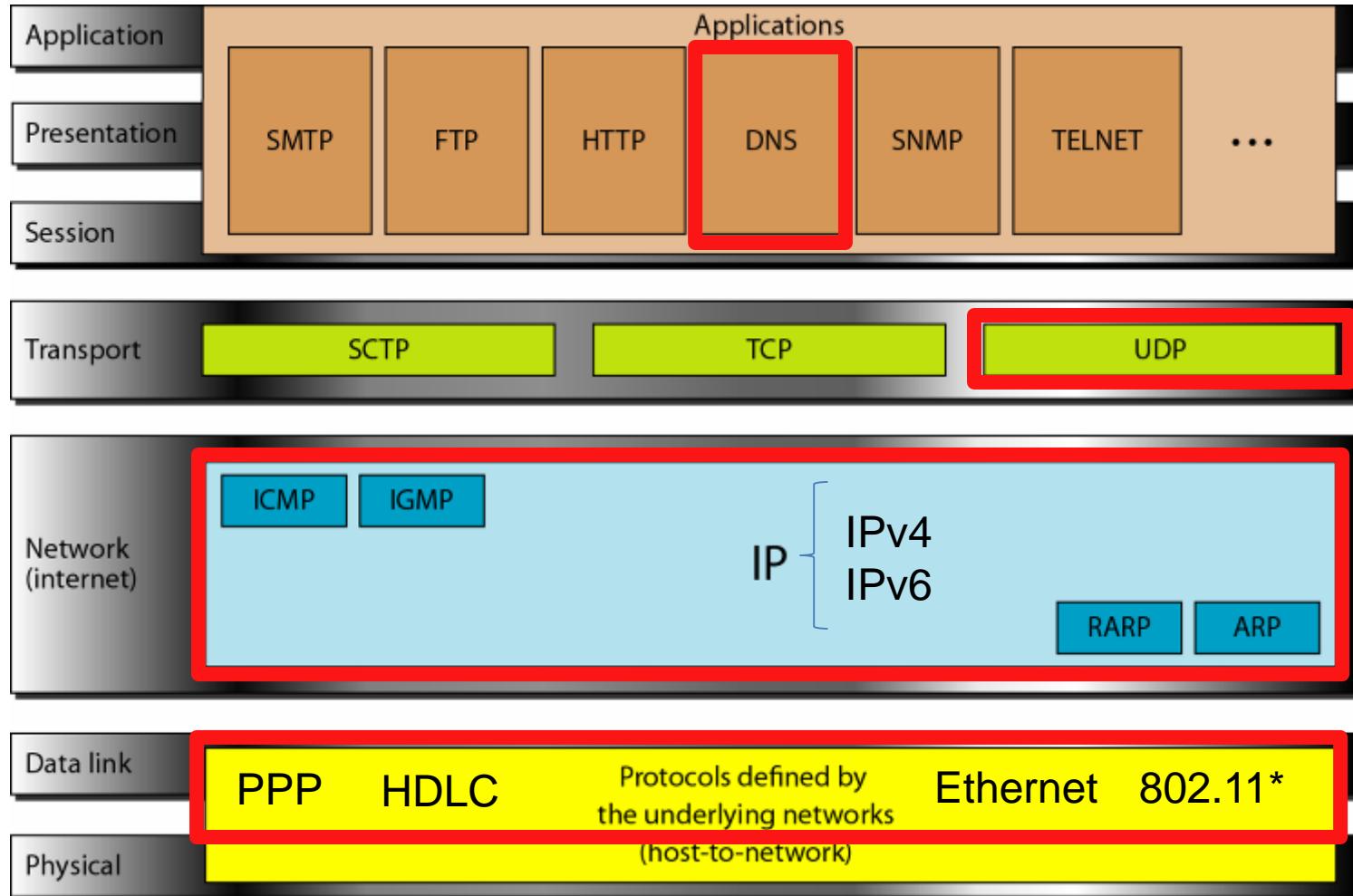


Overview

- Link Layer
- Network Layer
 - Addressing
 - Address Resolution (ARP)
 - Fragmentation
 - Intra-AS Routing
 - Distance Vector
 - Link State
 - Multicast Routing
 - IPv6
- Transport Layer
 - UDP
 - DNS



Protocols in the OSI Model



URLs to Names to Addresses

URL

DNS

IP Address

`http://www.wiki.com/index.html`

`www.wiki.com`

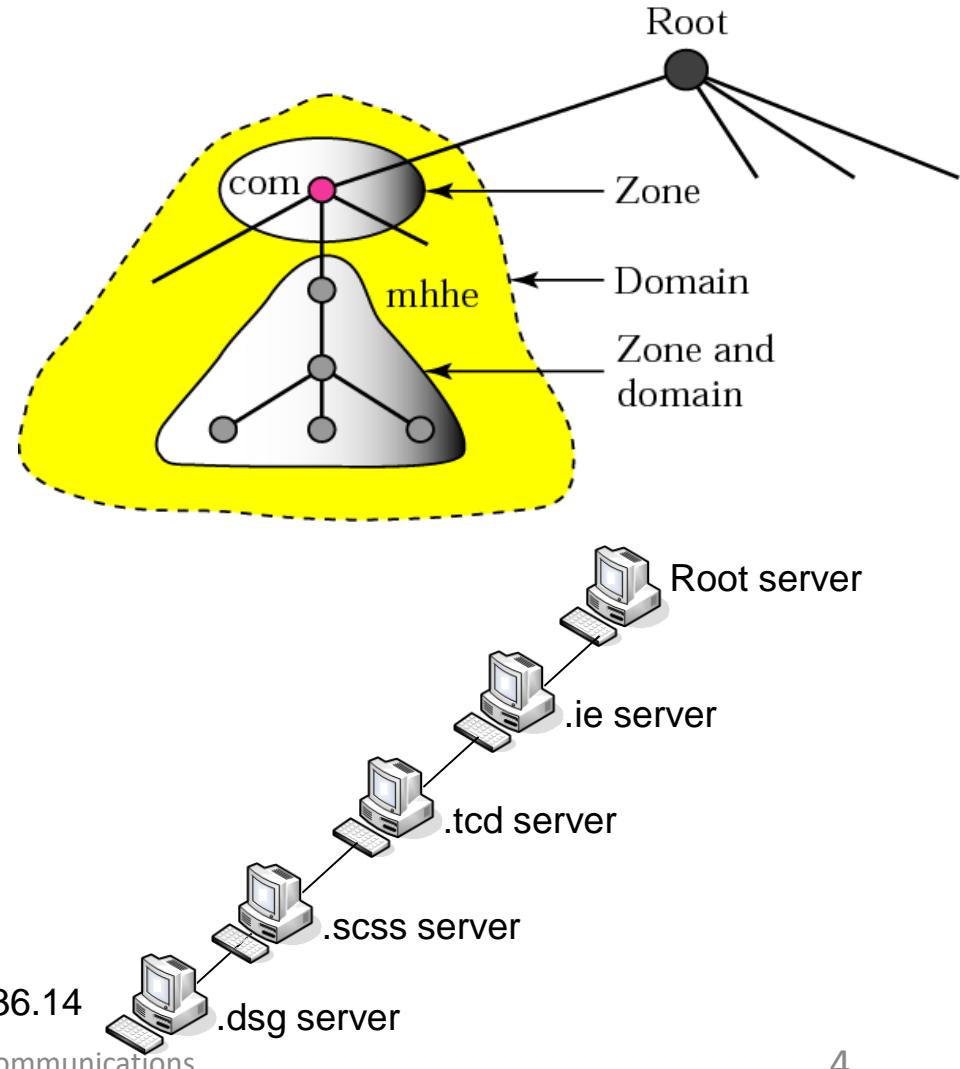
66.96.149.1

*URL = Uniform Resource Locator

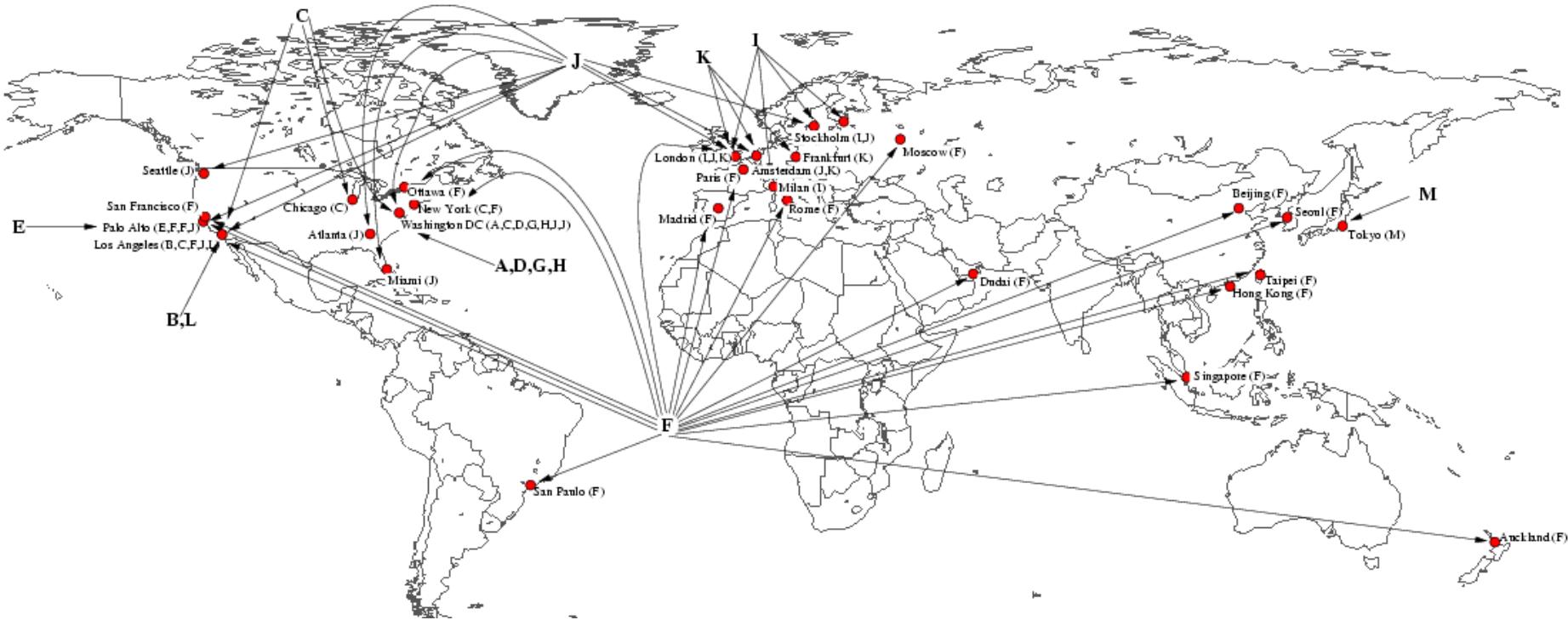


Hierarchy of Name Servers

- Every zone has a DNS server
- DNS server maintain lists of
 - Nodes in the zone
 - References to servers of zones underneath it

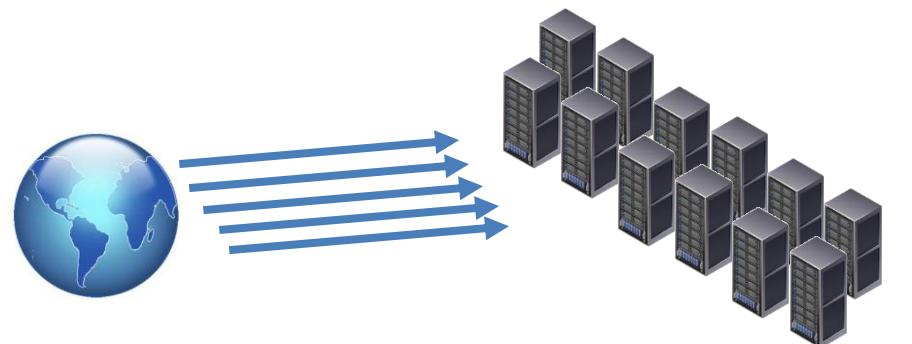
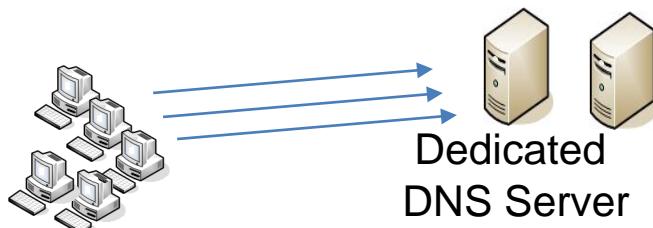


DNS Root Servers - Anycast



Mirai Bot & The Attack on Dyn

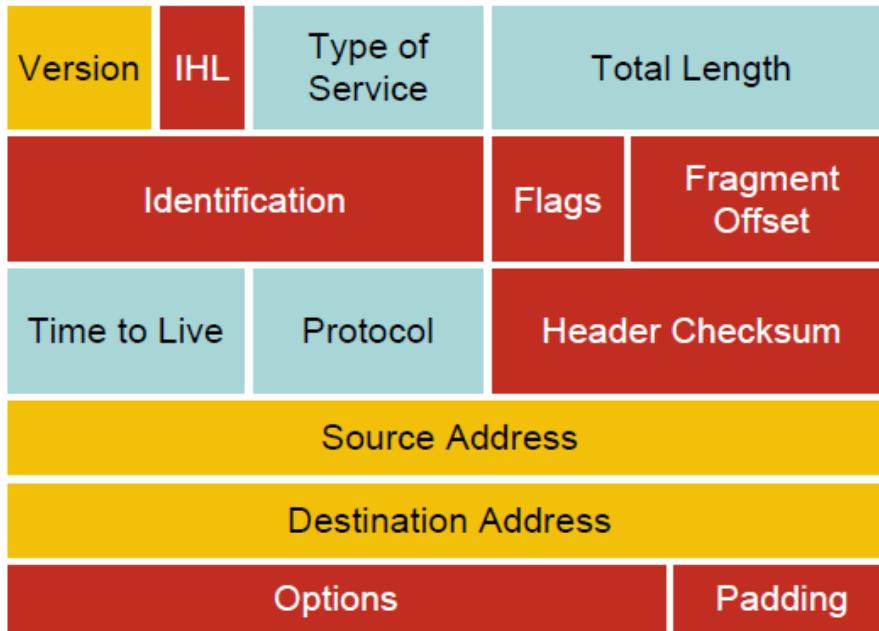
- Good example for scalable DNS services
- ... and the vulnerability of these services



Cloud/Datacentre-based
DNS Services

Header Comparison

IPv4 Header



IPv6 Header

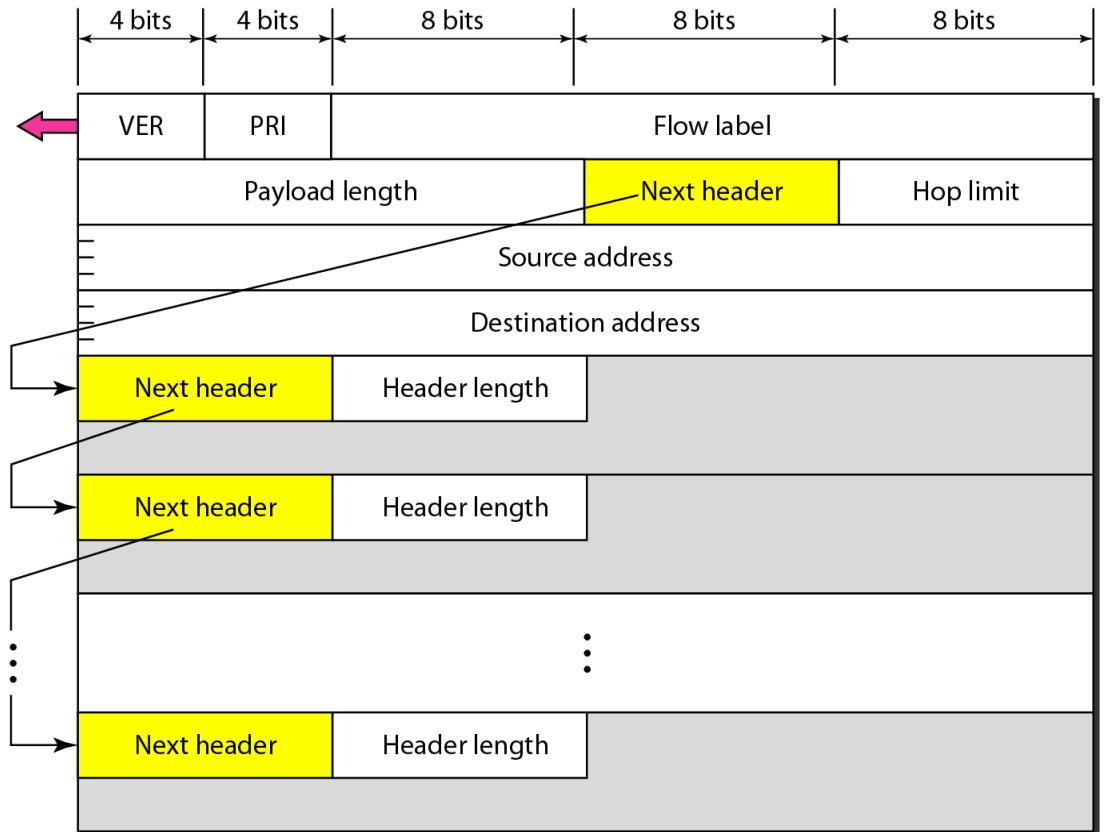


Legend

- Field's name kept from IPv4 to IPv6
- Fields not kept in IPv6
- Name and position changed in IPv6
- New field in IPv6

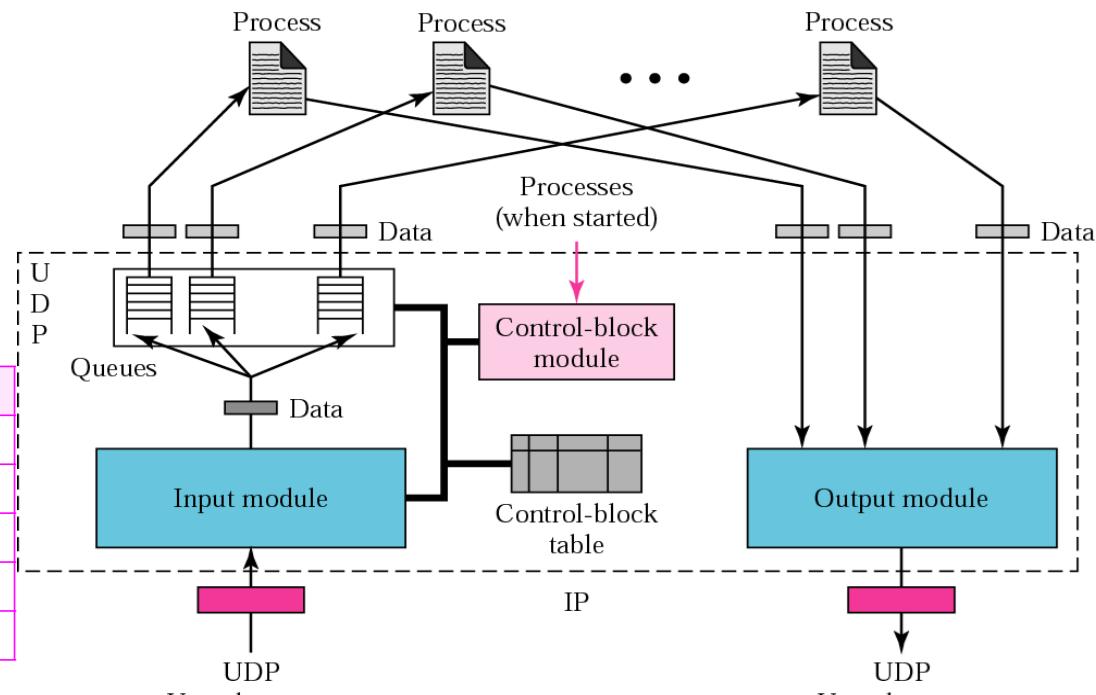
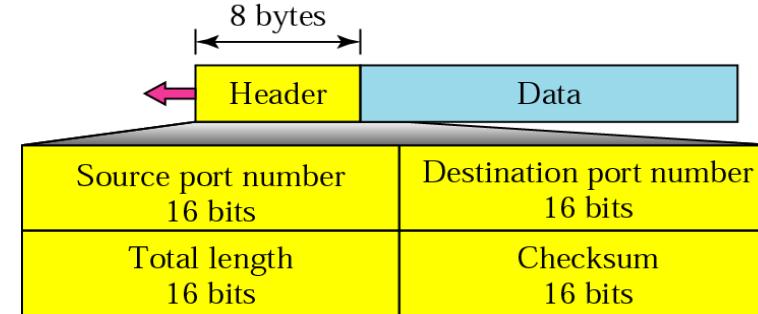


Extension Headers



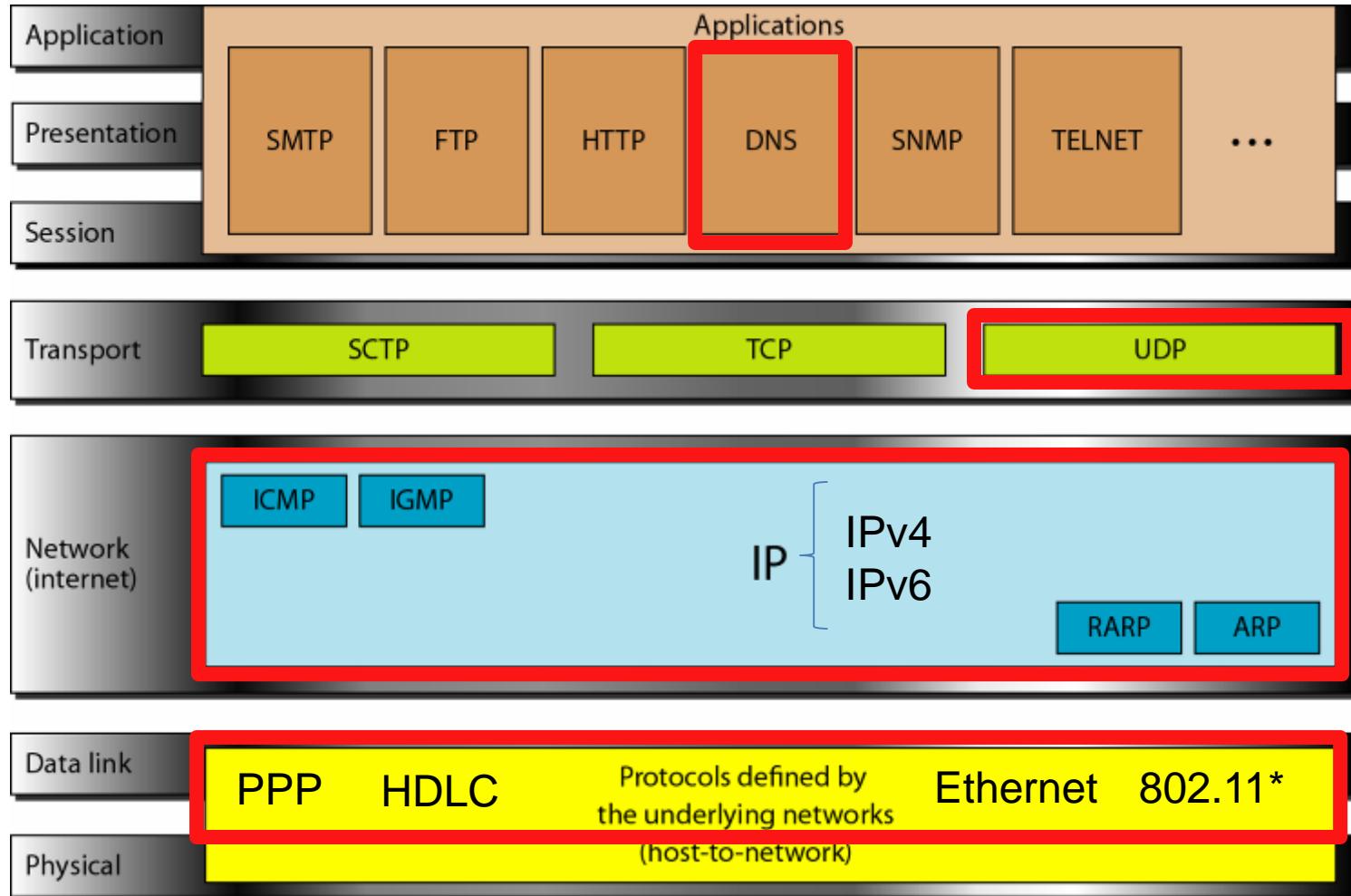
UDP – Portnumber id's Queues

- Connectionless, unreliable protocol
 - No flow and error control
 - Port numbers are used to multiplex data

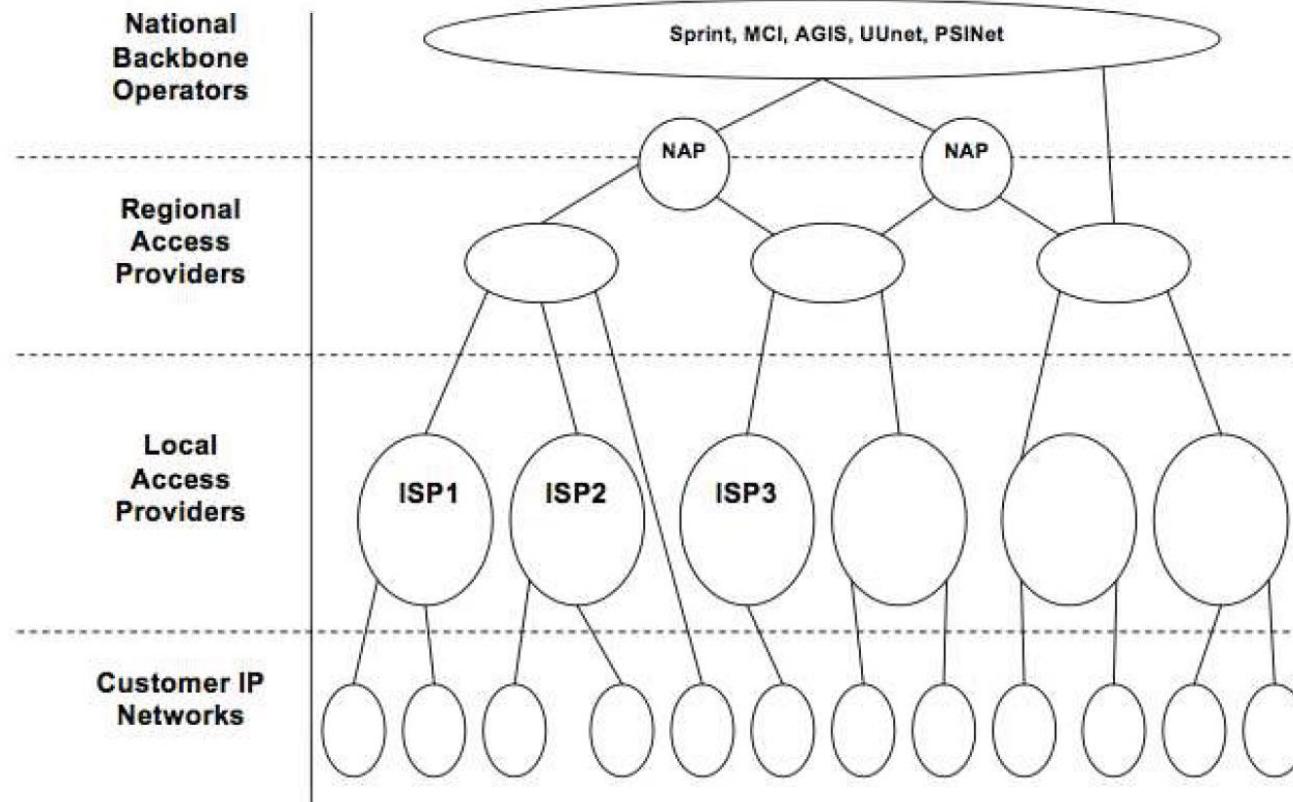


State	Process ID	Port Number	Queue Number
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
FREE			
IN-USE	4,652	52,012	38
FREE			

Protocols in the OSI Model



Traditional Logical Internet Topology



CS Predictions

- "I think there is a world market for maybe five computers."

Thomas Watson, President of IBM, 1943

- "There is no reason anyone would want a computer in their home."

Ken Olsen, Founder of Digital Equipment Corporation, 1977

- "I predict the Internet in 1996 will catastrophically collapse."

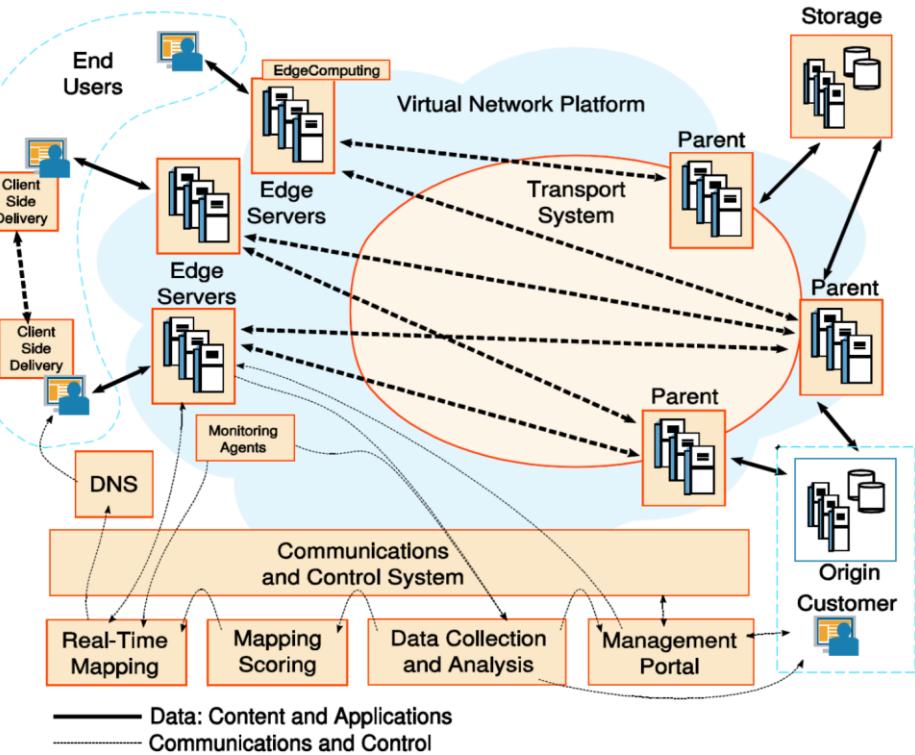
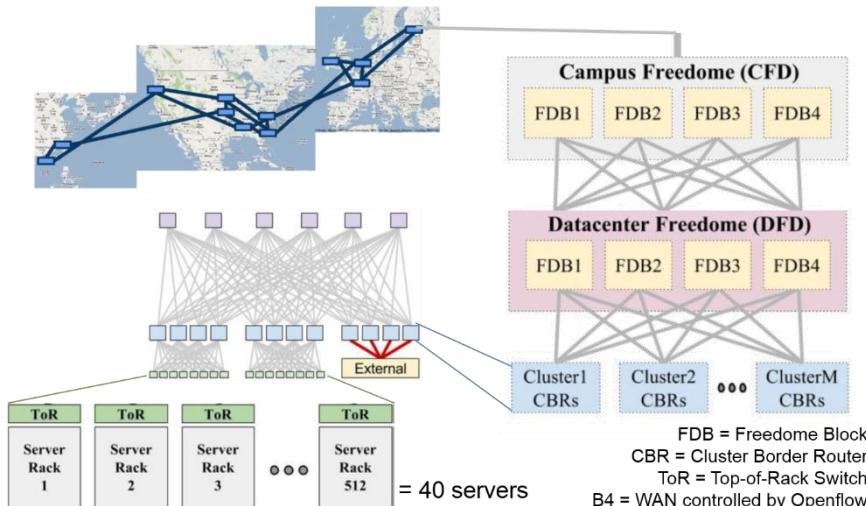
Robert Metcalfe, 1995

- "IPv6 is dead."

David Cheriton, 1999



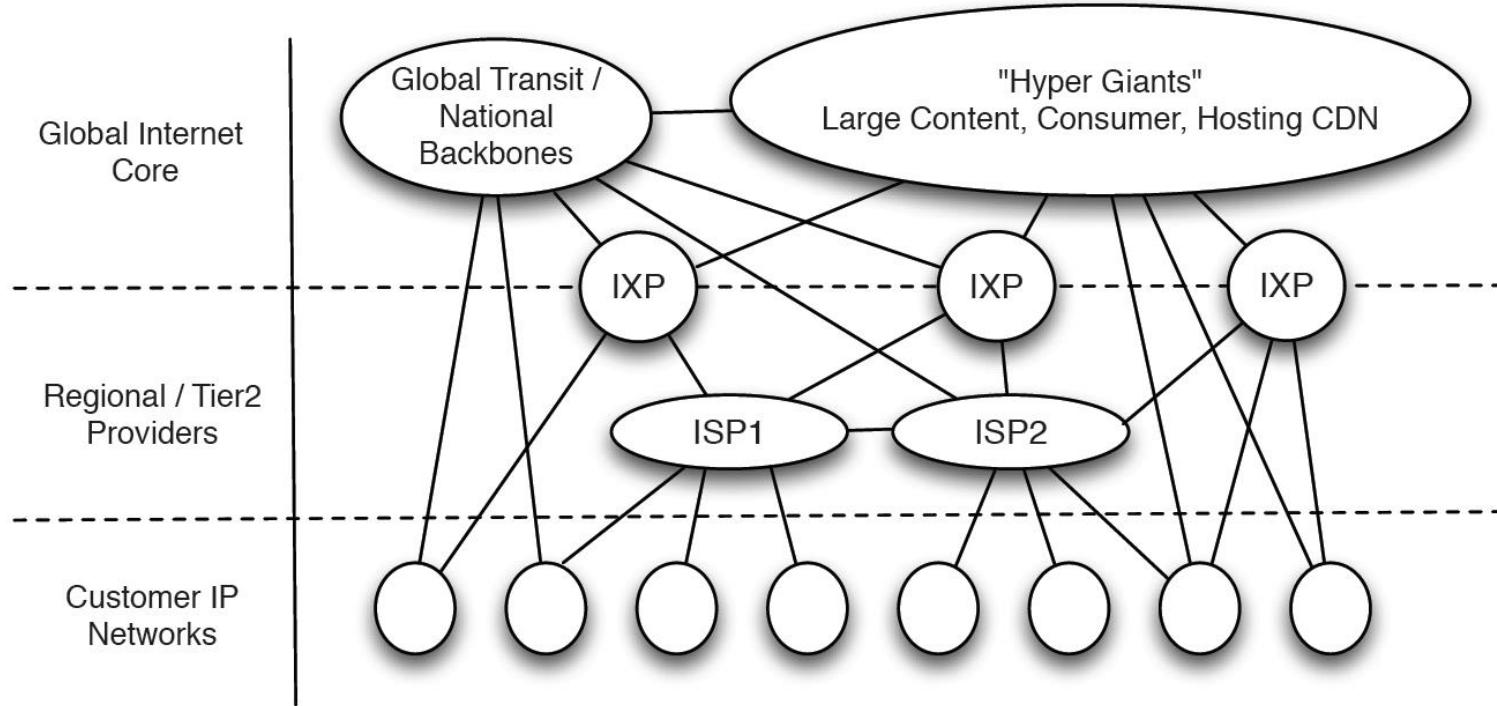
Hyper-Giants



- Google

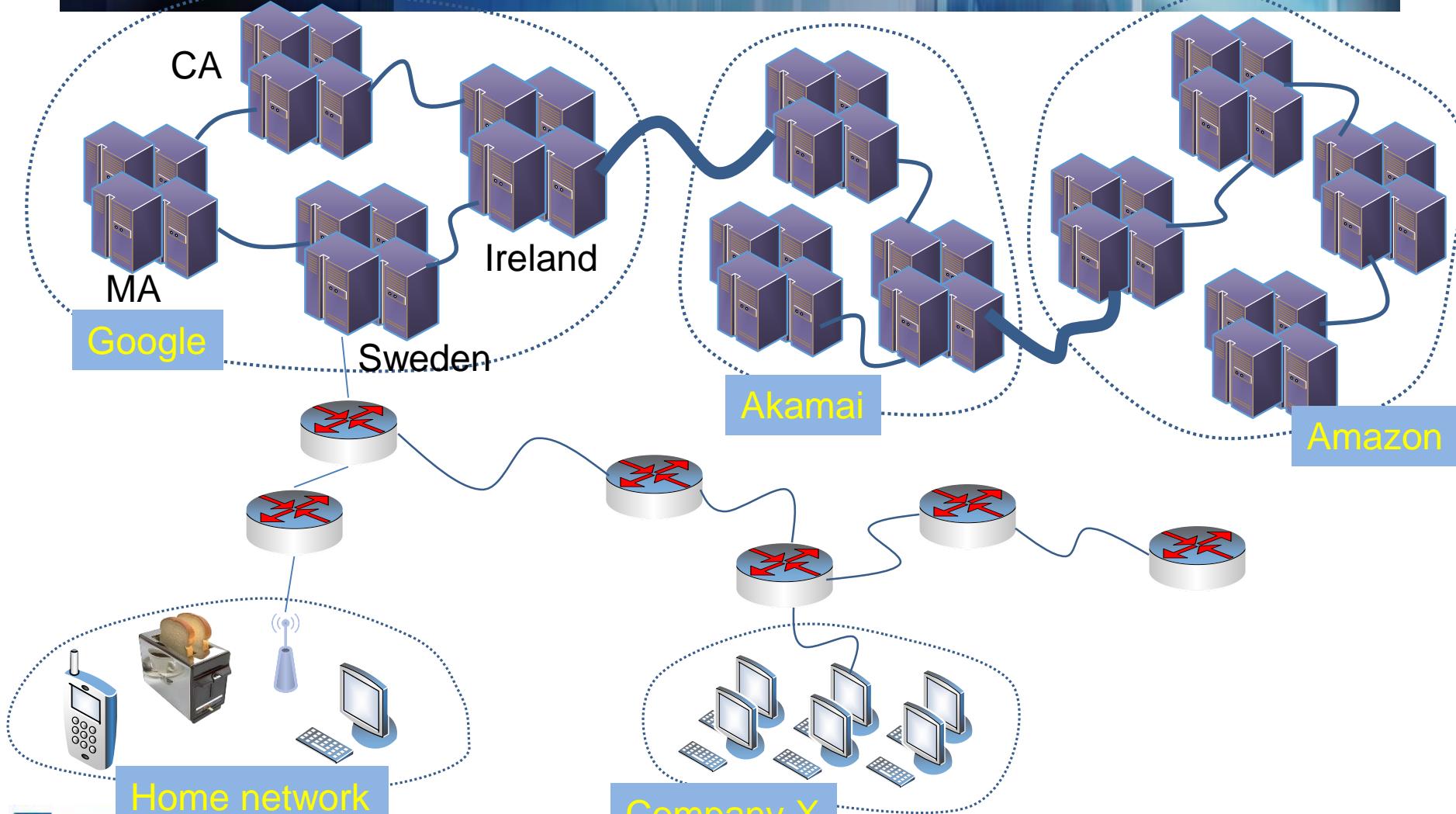
- Akamai

Emerging Logical Internet Topology



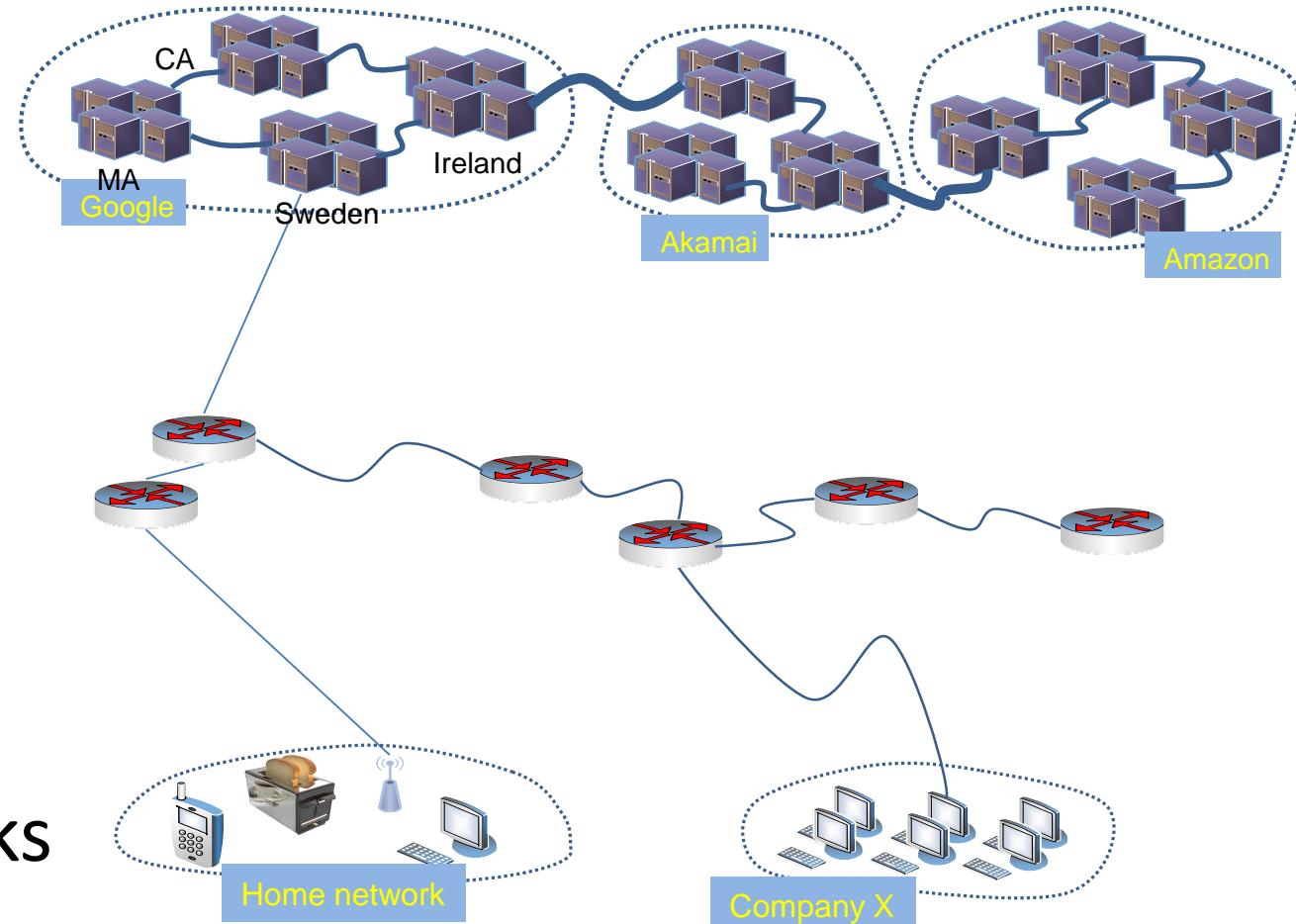
- According to a statement by Craig Labovitz in 2014:
 - 30 Entities created 50% of the traffic in the US at peak time

Datacentres at the Core?



My view of “future”* networking

- Sets of Datacentres



- Traditional Internet

- Edge networks

*or current?



Overview

- Link Layer
- Network Layer
 - Addressing
 - Address Resolution (ARP)
 - Fragmentation
 - Intra-AS Routing
 - Distance Vector
 - Link State
 - Multicast Routing
 - IPv6
- Transport Layer
 - UDP
 - DNS
- Software-Defined Networking / Openflow
- CLOS / Fat-tree
- ATM/MPLS



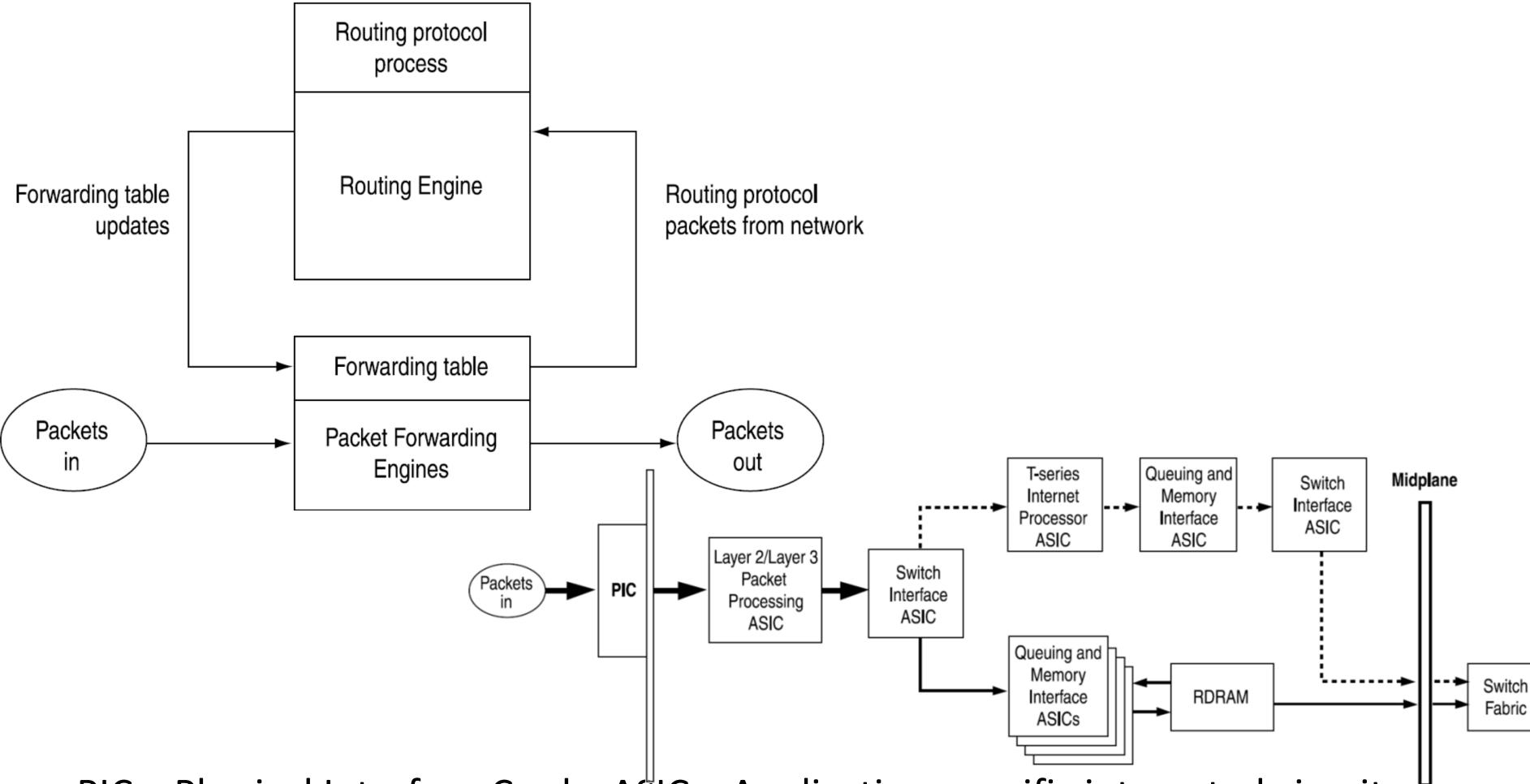


CS2031 Telecommunications II

SDN & Openflow



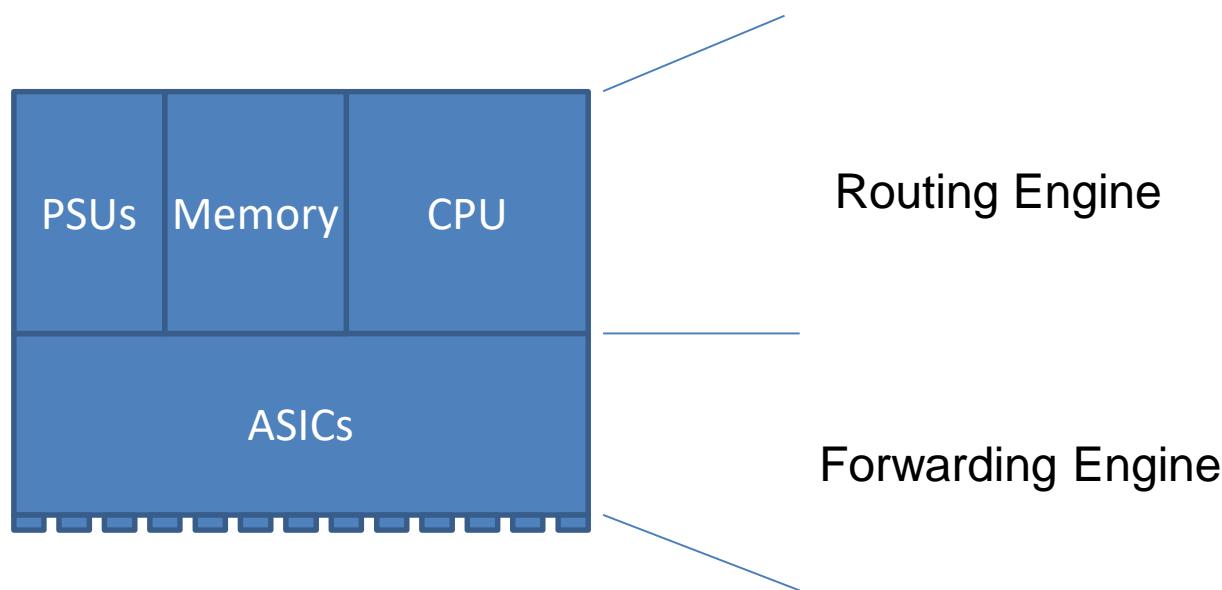
Switch/Router Internals



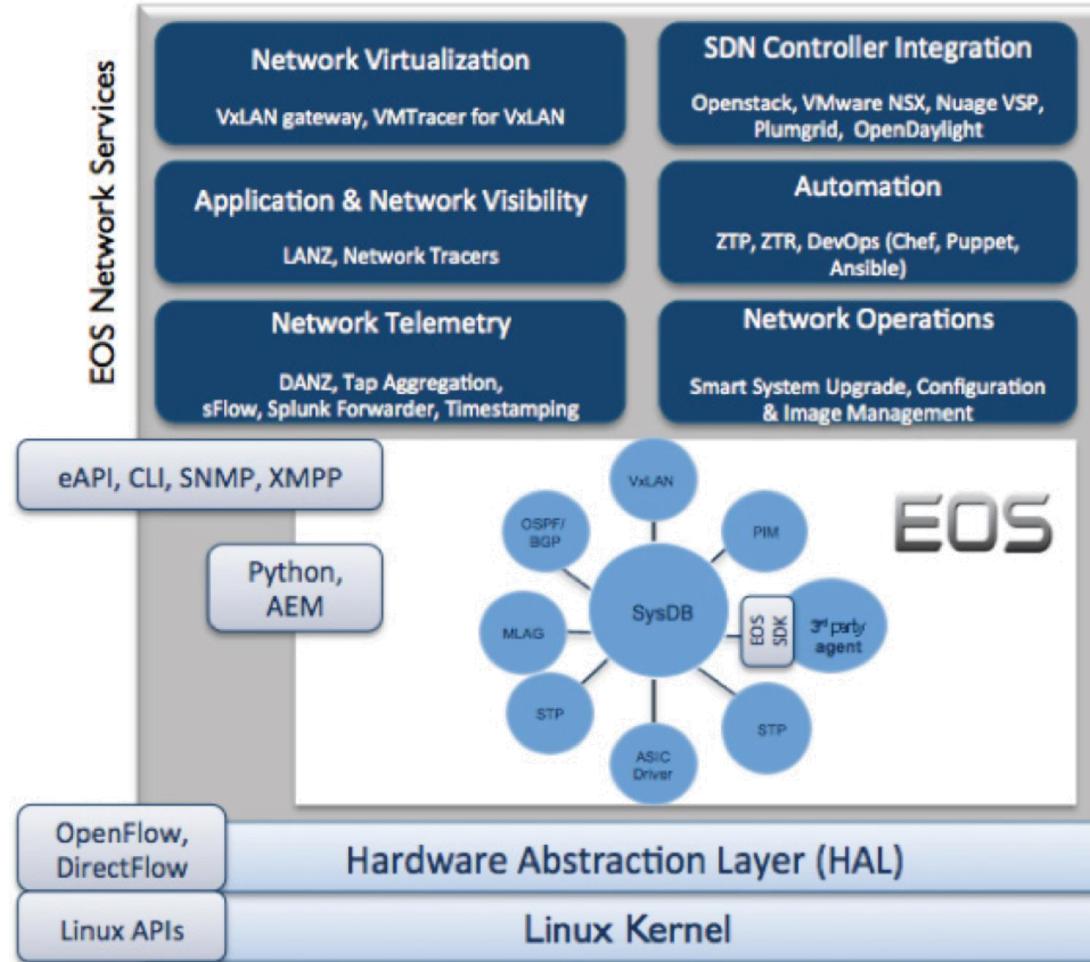
PIC = Physical Interface Card

ASIC = Application-specific integrated circuits

Switches/Routers



Example Switches/Router OS



Original Openflow Paper

- Openflow switches
- Controller w/ secure connection
- Configurable flow tables

OpenFlow: Enabling Innovation in Campus Networks

March 14, 2008

Nick McKeown
Stanford University

Guru Parulkar
Stanford University

Scott Shenker
University of California,
Berkeley

Tom Anderson
University of Washington

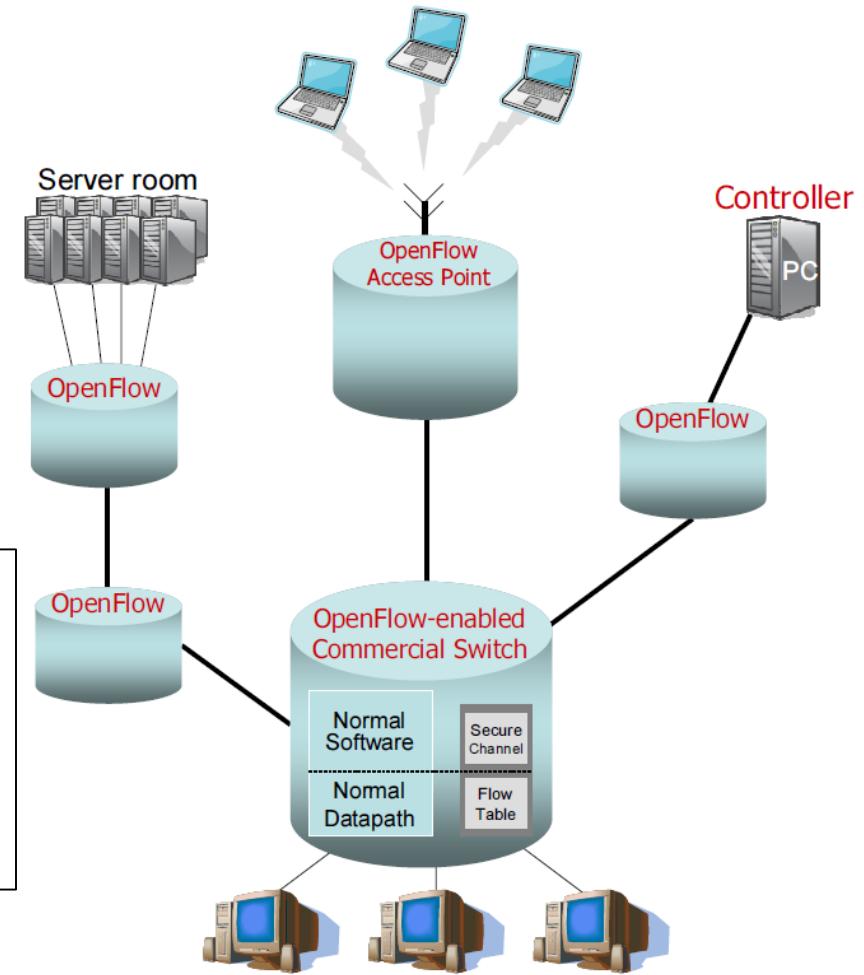
Larry Peterson
Princeton University

Jonathan Turner
Washington University in
St. Louis

Hari Balakrishnan
MIT

Jennifer Rexford
Princeton University

Header of 2008 Paper



Openflow Switch

Software Layer

OpenFlow Client

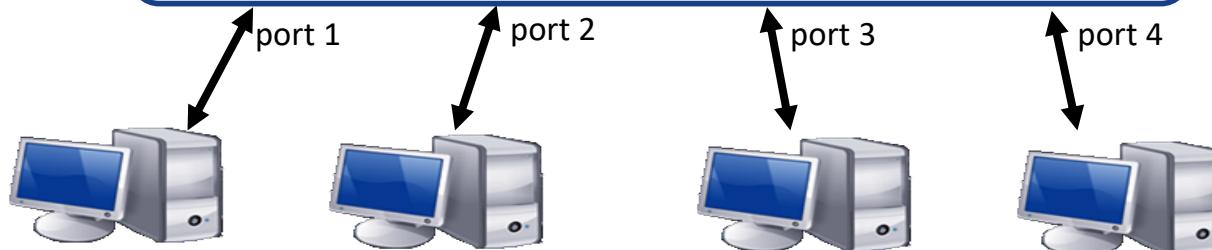
Hardware Layer

Flow Table						
MAC src	MAC dst	IP Src	IP Dst	TCP sport	TCP dport	Action
*	*	*	5.6.7.8	*	*	port 1

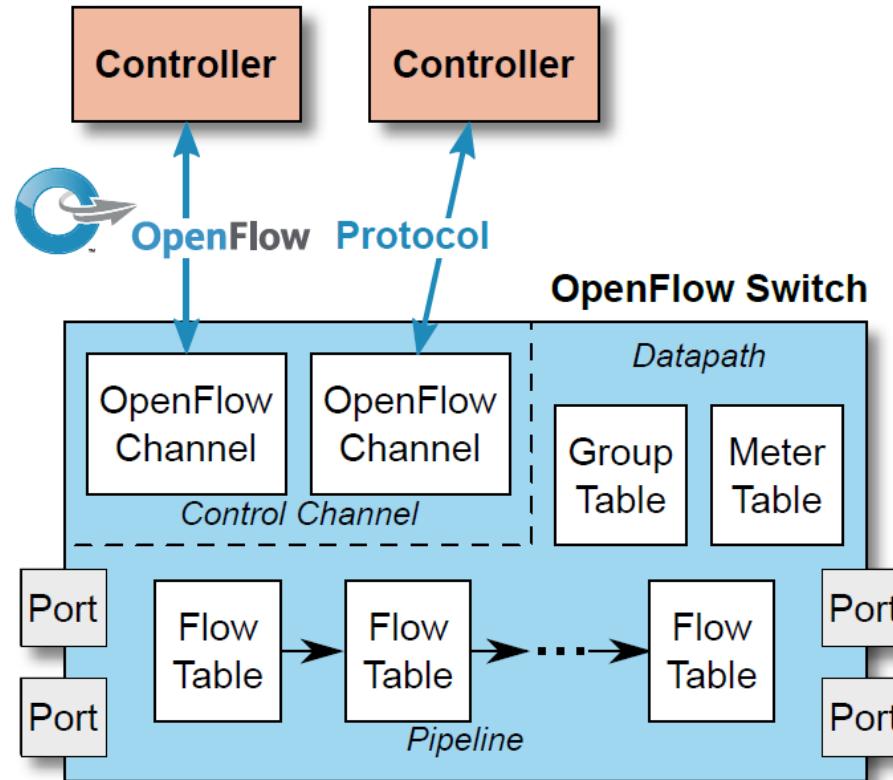


SSH conn.

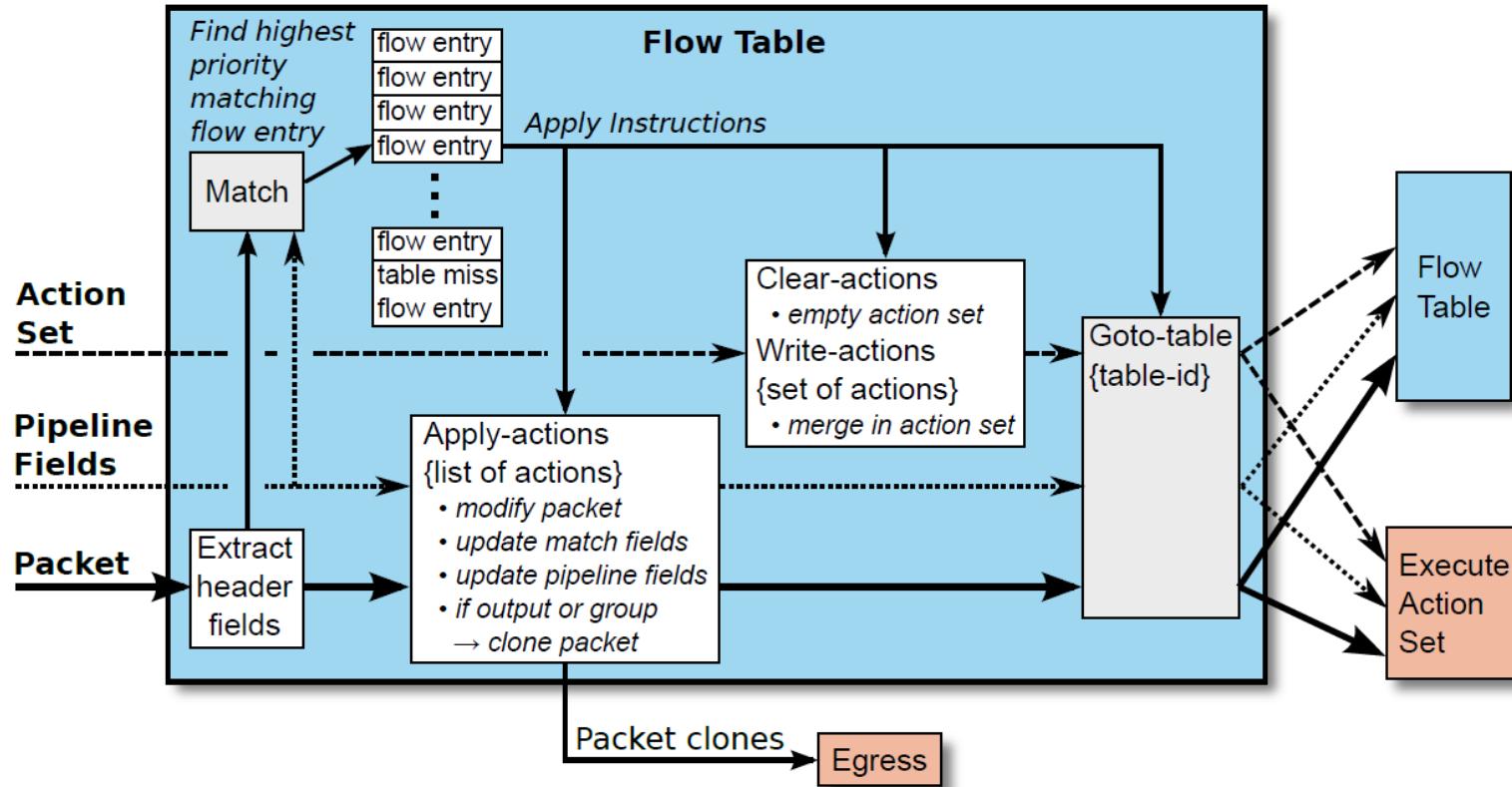
Controller



Components of Openflow Switches



Flowtable Processing



Counters in Flowtables

Counter	Bits	
Per Flow Table		
Reference Count (active entries)	32	<i>Required</i>
Packet Lookups	64	<i>Optional</i>
Packet Matches	64	<i>Optional</i>
Per Flow Entry		
Received Packets	64	<i>Optional</i>
Received Bytes	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>
Per Port		
Received Packets	64	<i>Required</i>
Transmitted Packets	64	<i>Required</i>
Received Bytes	64	<i>Optional</i>
Transmitted Bytes	64	<i>Optional</i>
Receive Drops	64	<i>Optional</i>
Transmit Drops	64	<i>Optional</i>
Receive Errors	64	<i>Optional</i>
Transmit Errors	64	<i>Optional</i>
Receive Frame Alignment Errors	64	<i>Optional</i>
Receive Overrun Errors	64	<i>Optional</i>
Receive CRC Errors	64	<i>Optional</i>
Collisions	64	<i>Optional</i>
Duration (seconds)	32	<i>Required</i>
Duration (nanoseconds)	32	<i>Optional</i>
Per Queue		





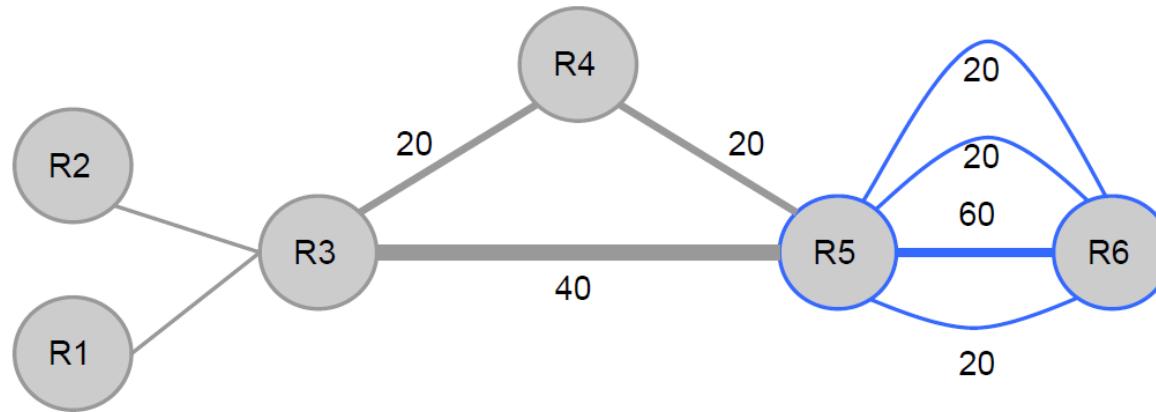
Why????

As in: Why Openflow? ☺



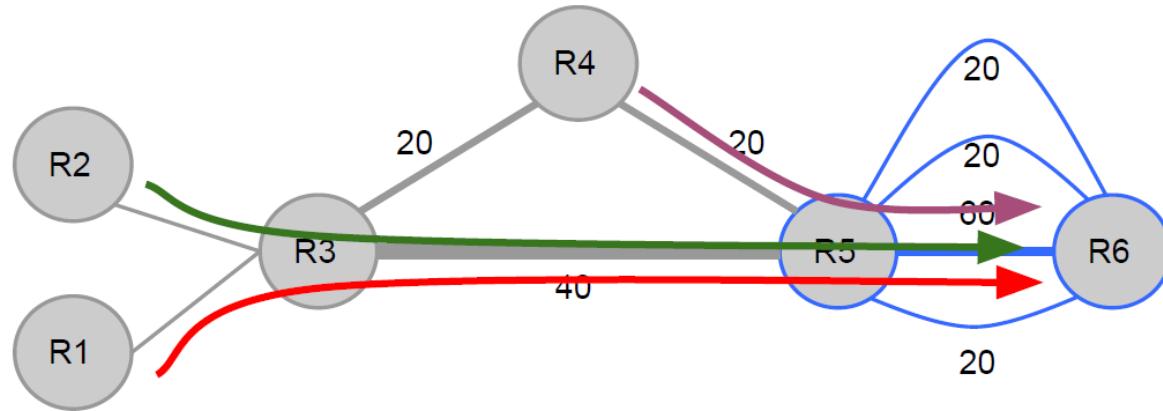
Traditional Net. Example

- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



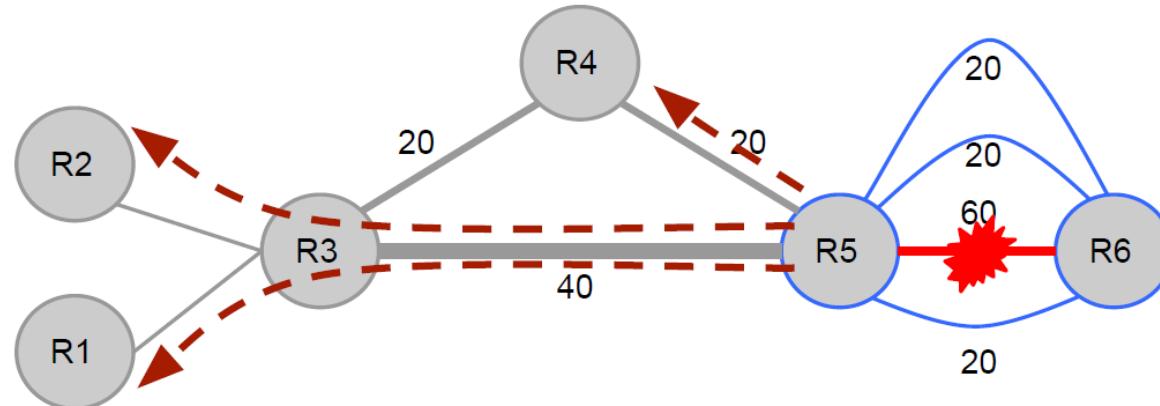
Traditional Net. Example

- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



Traditional Net. Example

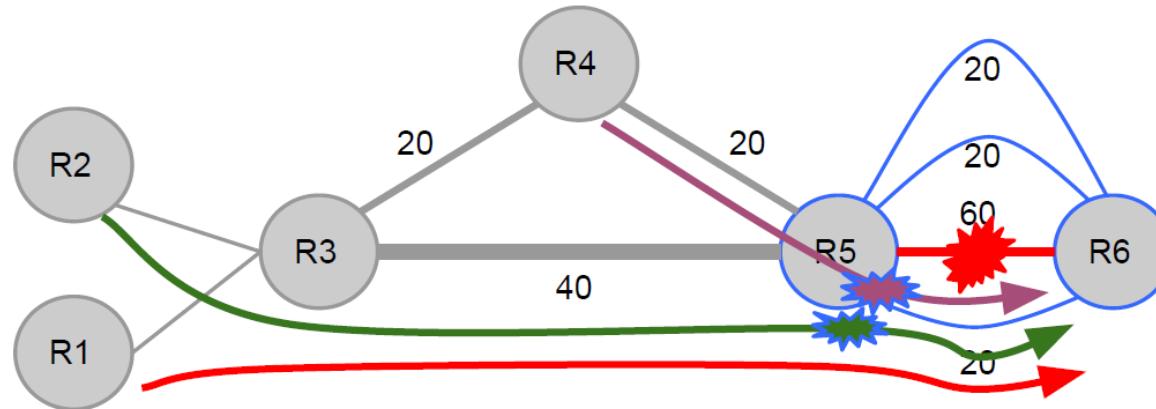
- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



- R5-R6 link fails
 - R1, R2, R4 autonomously try for next best path

Traditional Net. Example

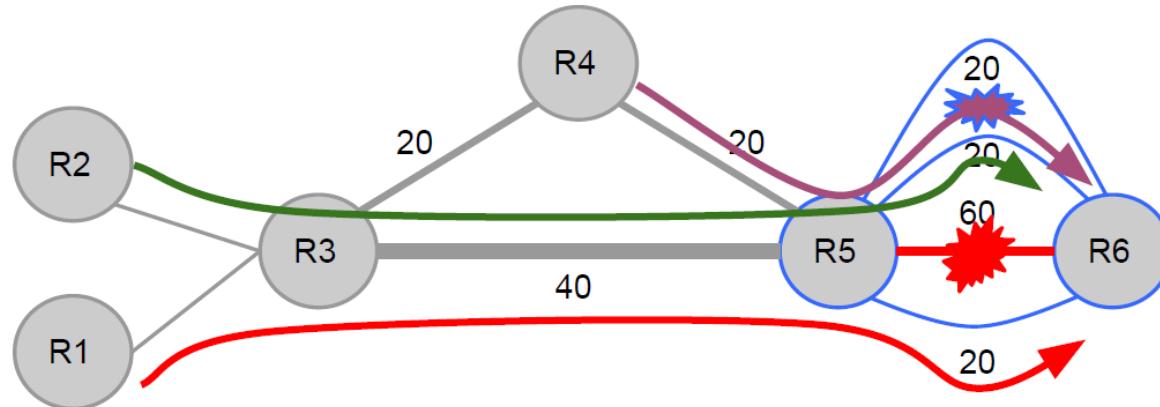
- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



- R5-R6 link fails
 - R1, R2, R4 *autonomously* try for next best path
 - R1 wins, R2, R4 retry for next best path

Traditional Net. Example

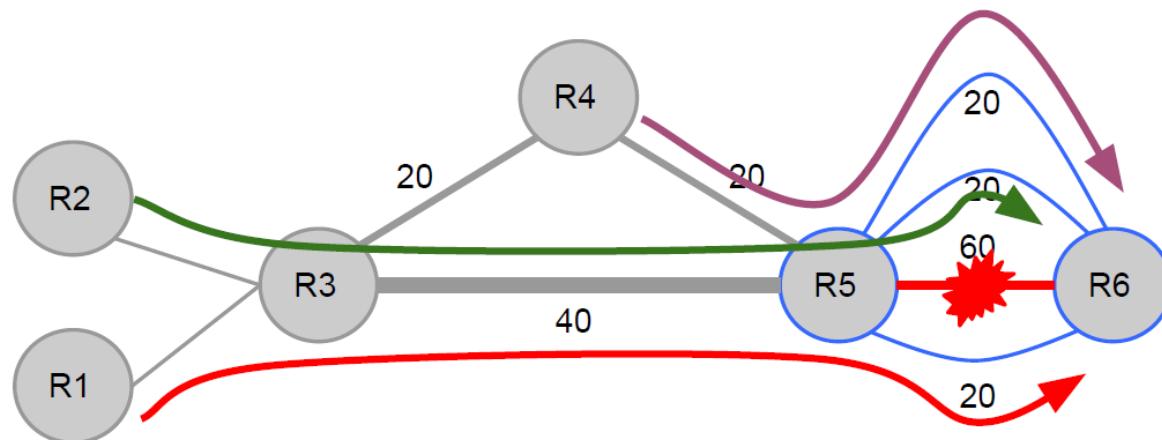
- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



- R5-R6 link fails
 - R1, R2, R4 *autonomously* try for next best path
 - R1 wins, R2, R4 retry for next best path
 - R2 wins this round, R4 retries again

Traditional Net. Example

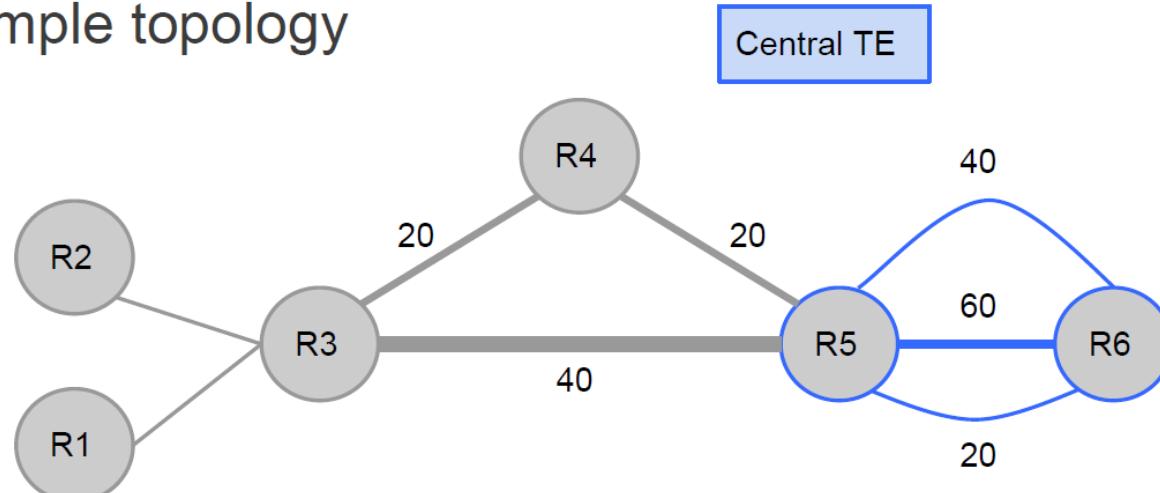
- Flows: R1->R6: 20; R2->R6: 20; R4->R6: 20



- R5-R6 link fails
 - R1, R2, R4 *autonomously* try for next best path
 - R1 wins, R2, R4 retry for next best path
 - R2 wins this round, R4 retries again
 - R4 finally gets third best path

Topology with Central TE

- Simple topology



- Flows:
 - R1->R6: 20; R2->R6: 20; R4->R6: 20

TE = Traffic Engineering

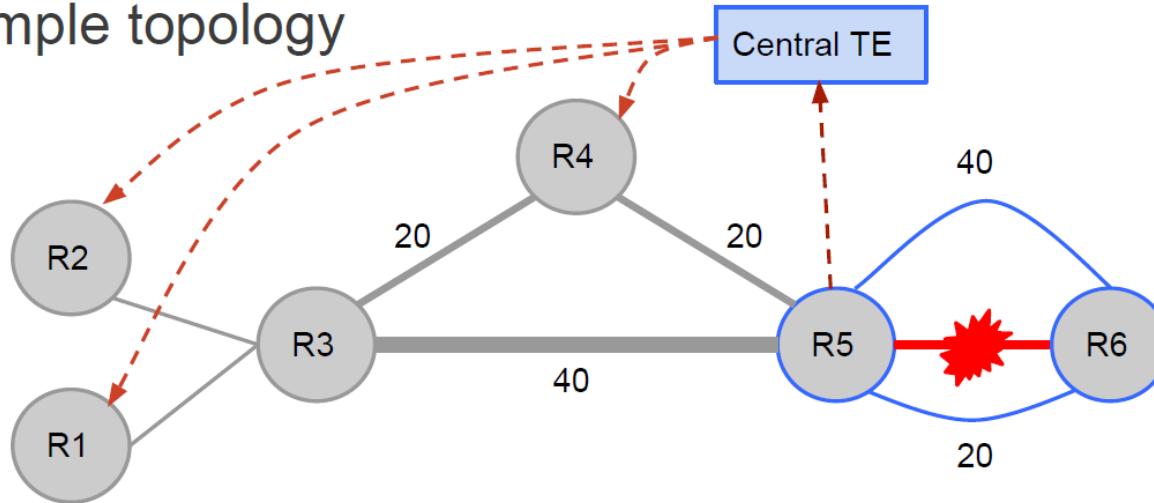
34

• Urs Hoelzle, Open Network Summit 2012



Topology with Central TE

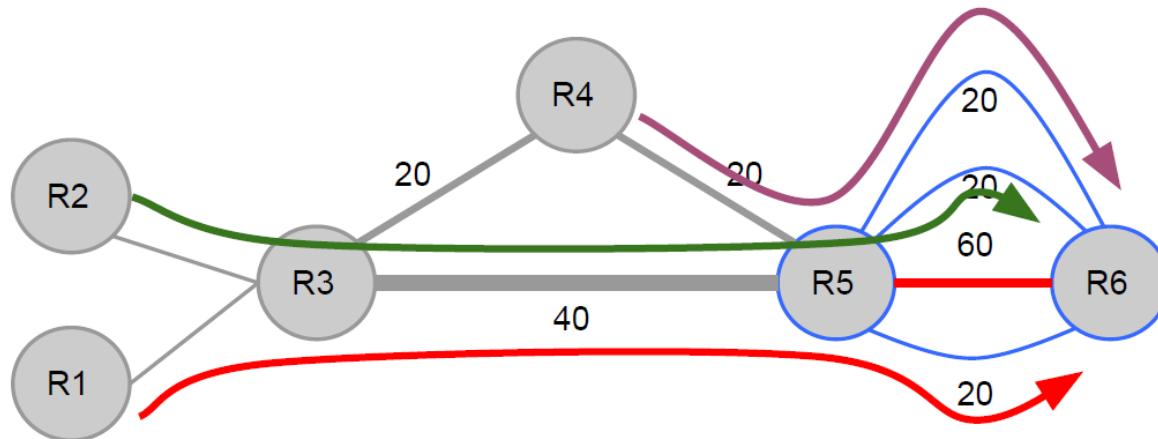
- Simple topology



- Flows:
 - R1->R6: 20; R2->R6: 20; R4->R6: 20
- R5-R6 fails
 - R5 informs TE, which programs routers in one shot

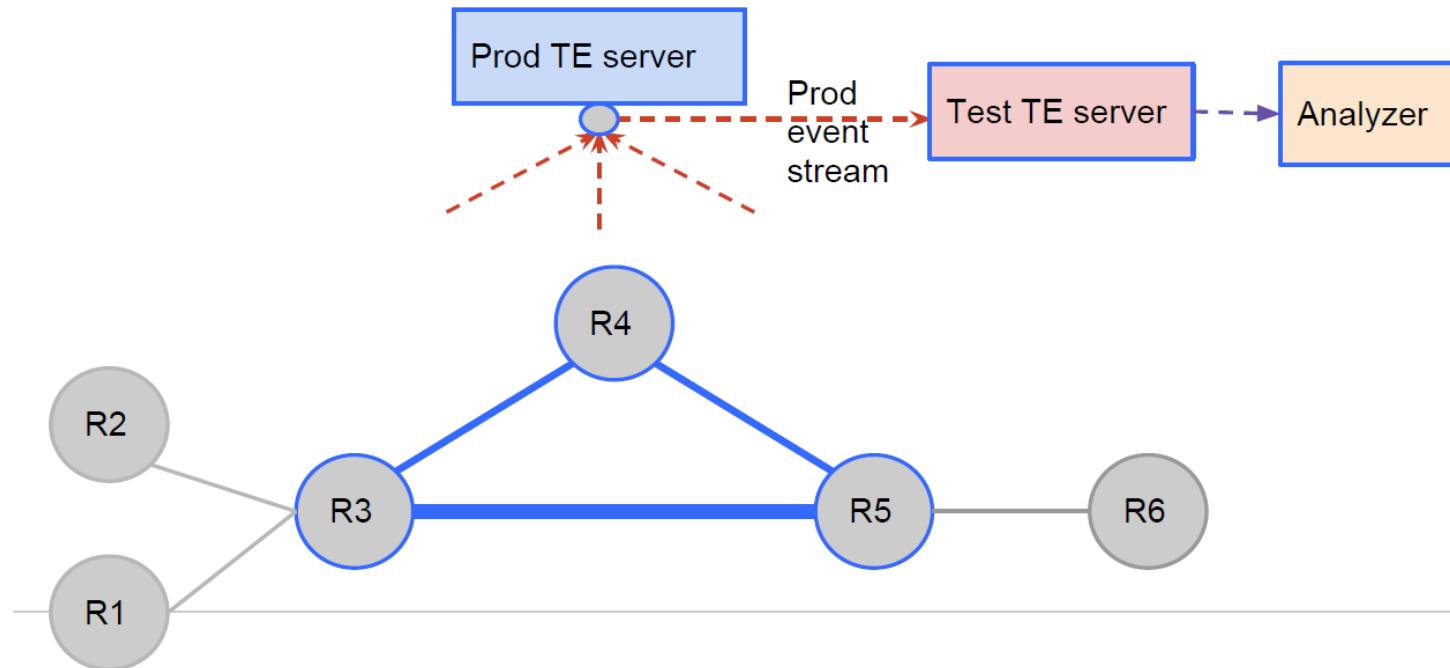
Topology with Central TE

- Simple topology

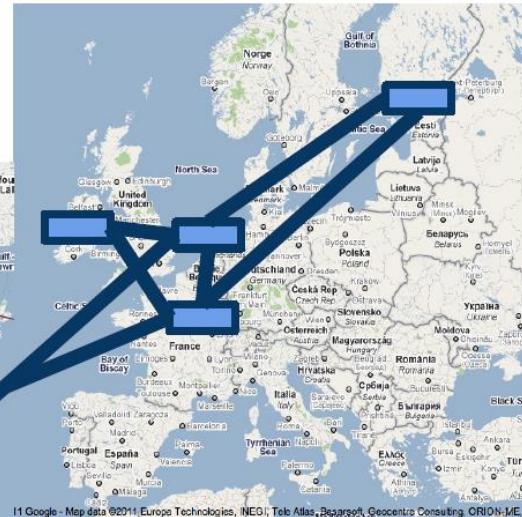
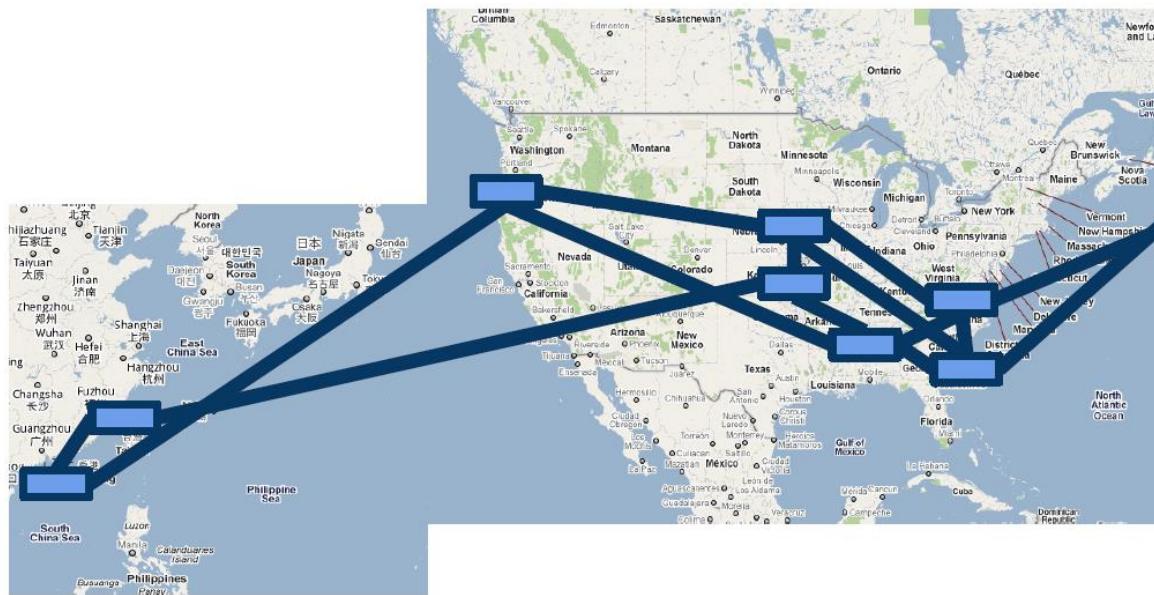


- Flows:
 - R1->R6: 20; R2->R6: 20; R4->R6: 20
- R5-R6 link fails
 - R5 informs TE, which programs routers in one shot
 - Leads to faster realization of target optimum

Traffic Engineering with Analyzer

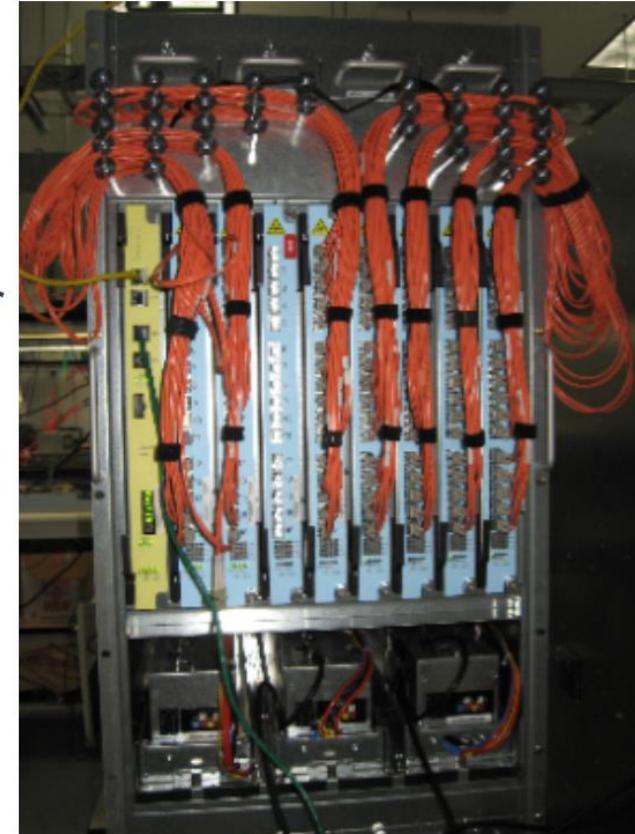


Google's WAN

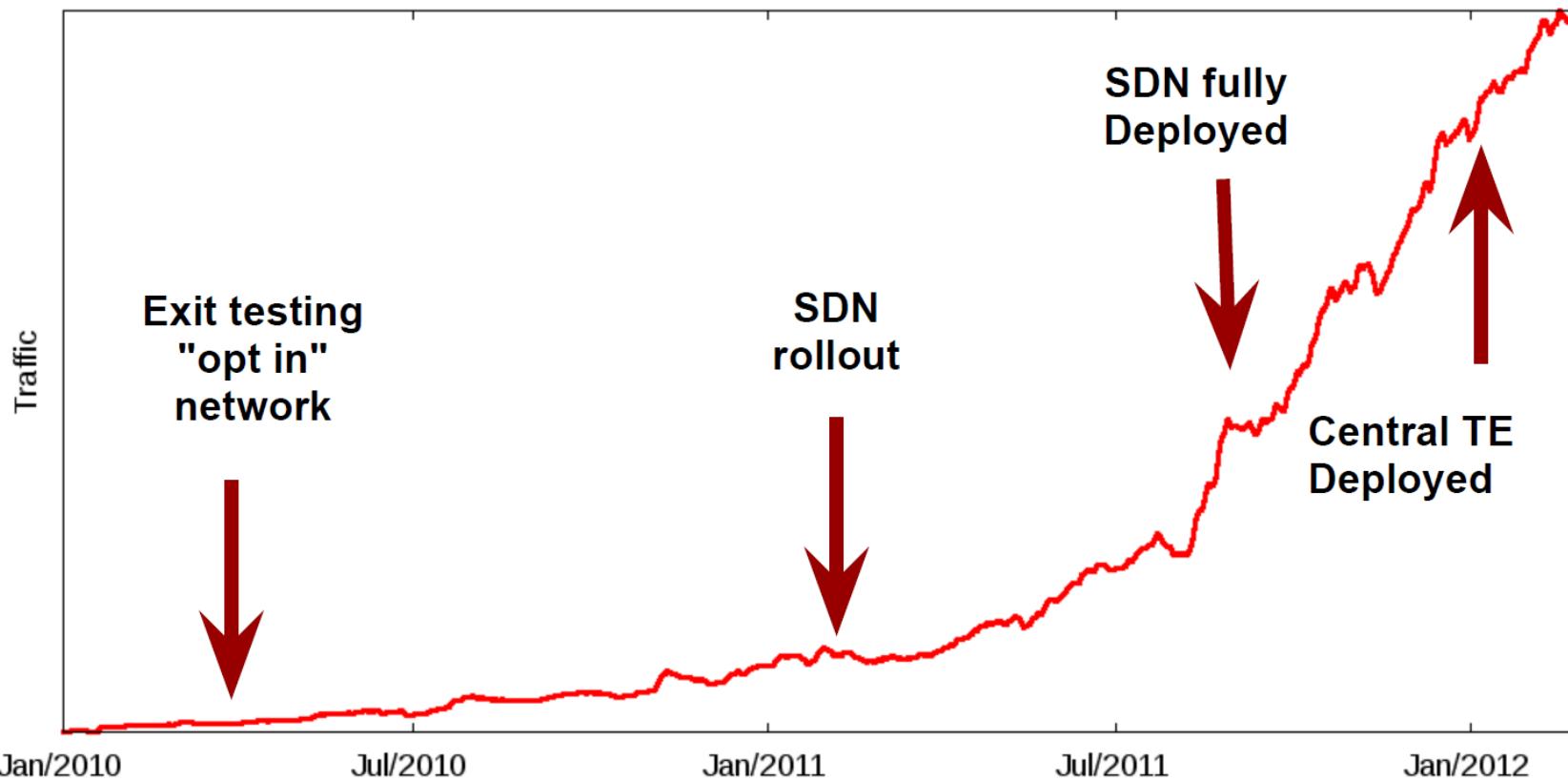


Google's Hardware

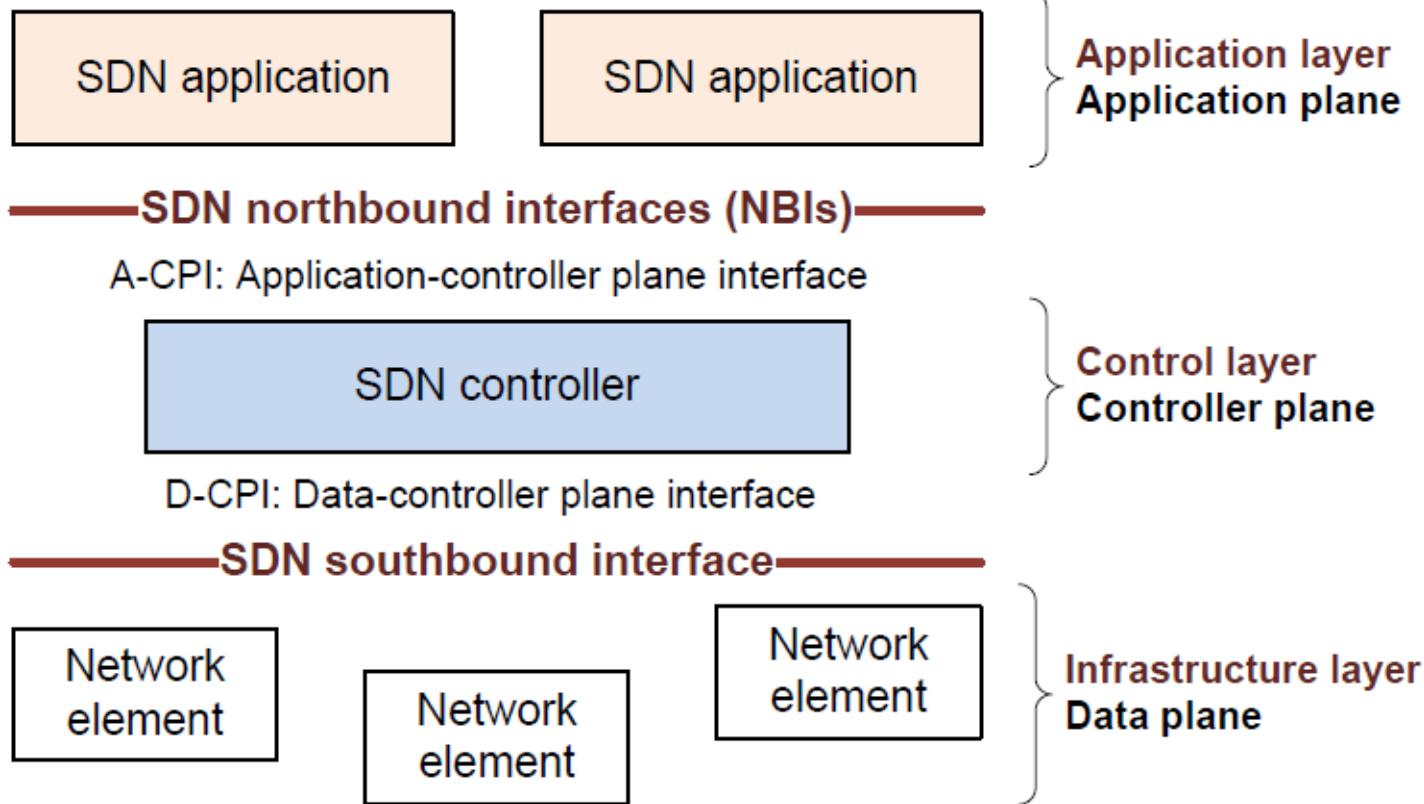
- Built from merchant silicon
 - 100s of ports of nonblocking 10GE
- OpenFlow support
- Open source routing stacks for BGP, ISIS
- Does not have all features
 - No support for AppleTalk...
- Multiple chassis per site
 - Fault tolerance
 - Scale to multiple Tbps



Timeline of Deployment



SDN Architecture



SDN Overview

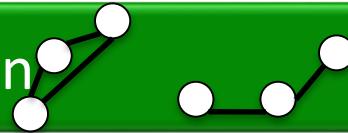
Specifies behavior

Control Program

Abstract Network Model

Compiles to topology

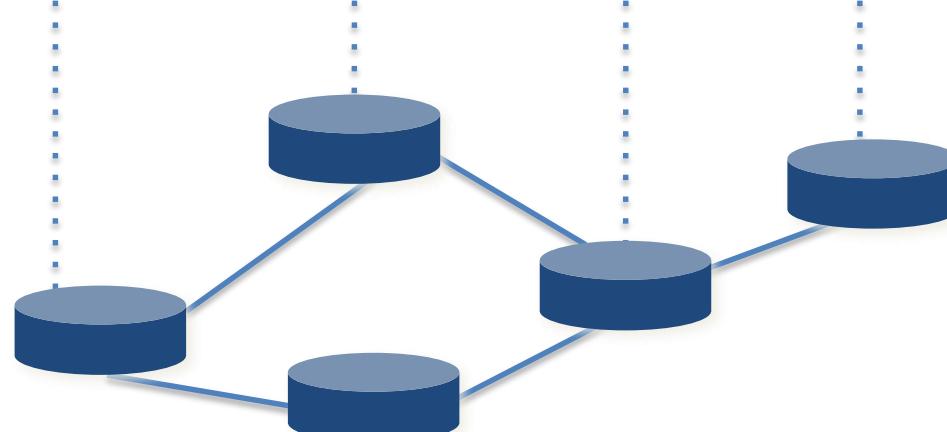
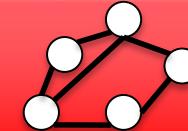
Network Virtualization

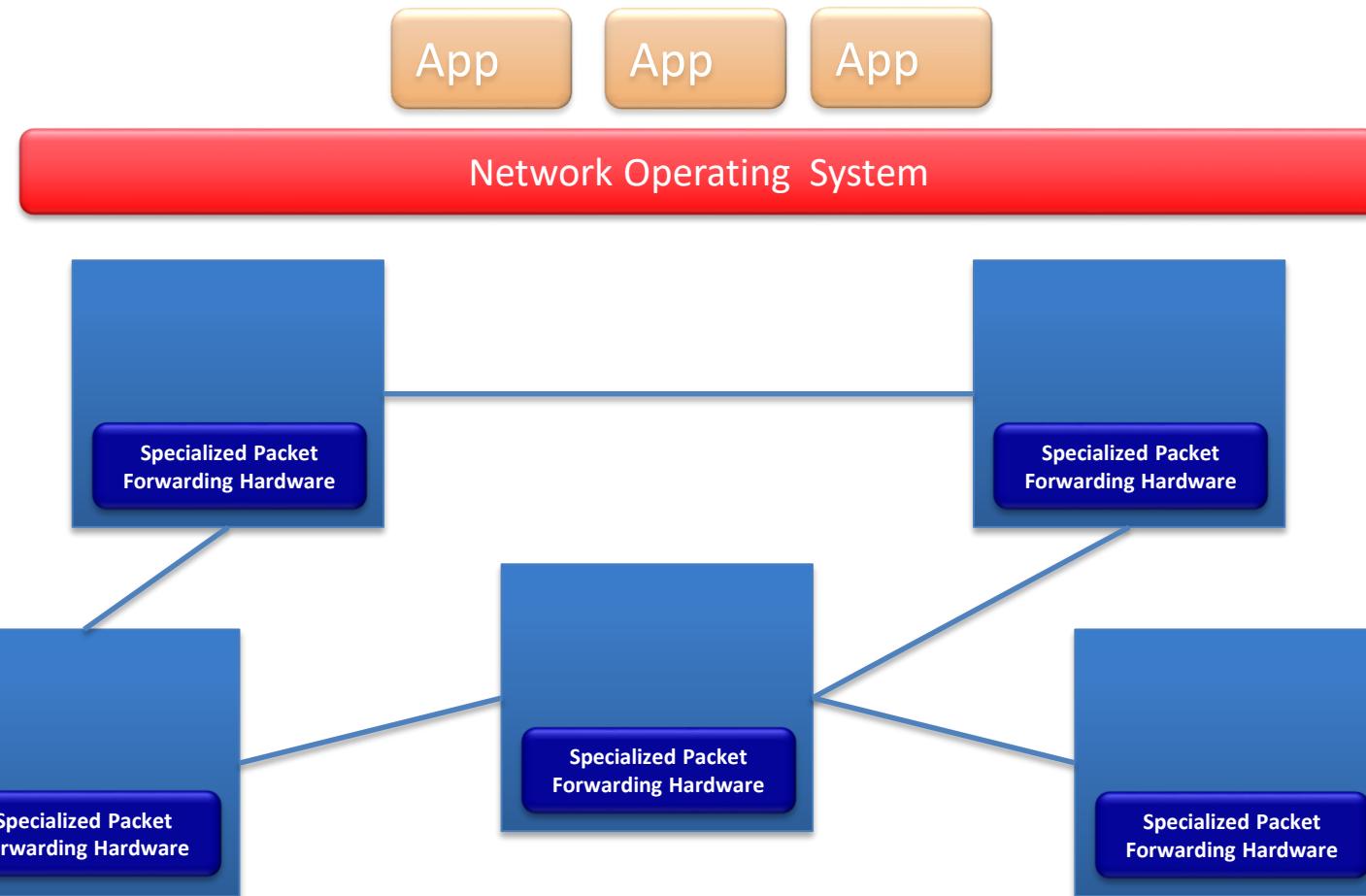


Global Network View

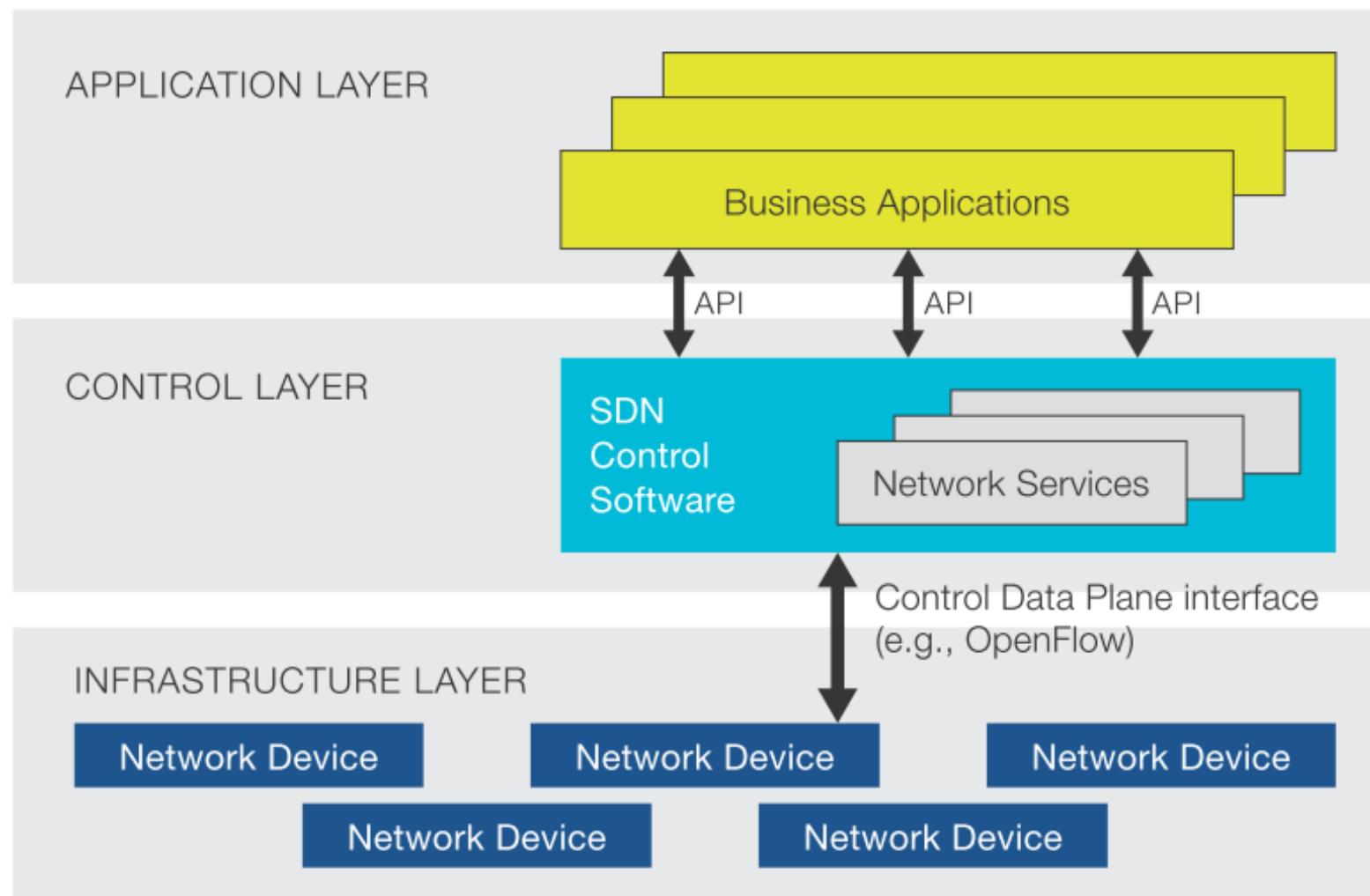
Transmits to switches

Network OS

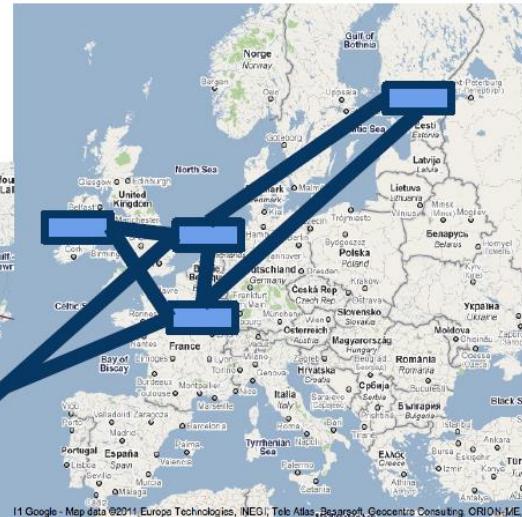
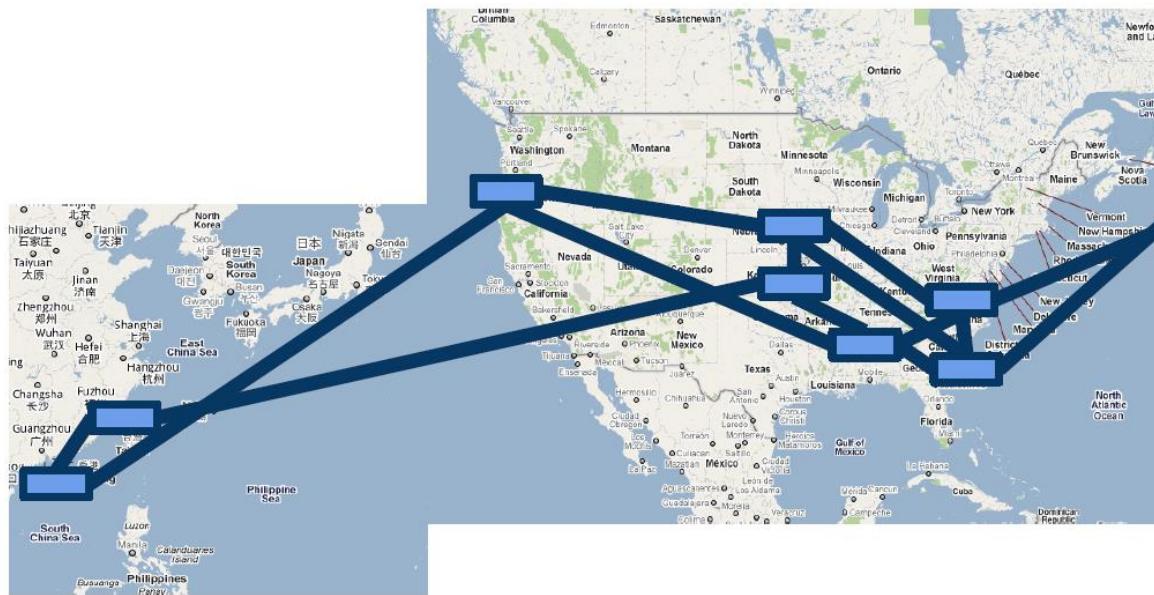




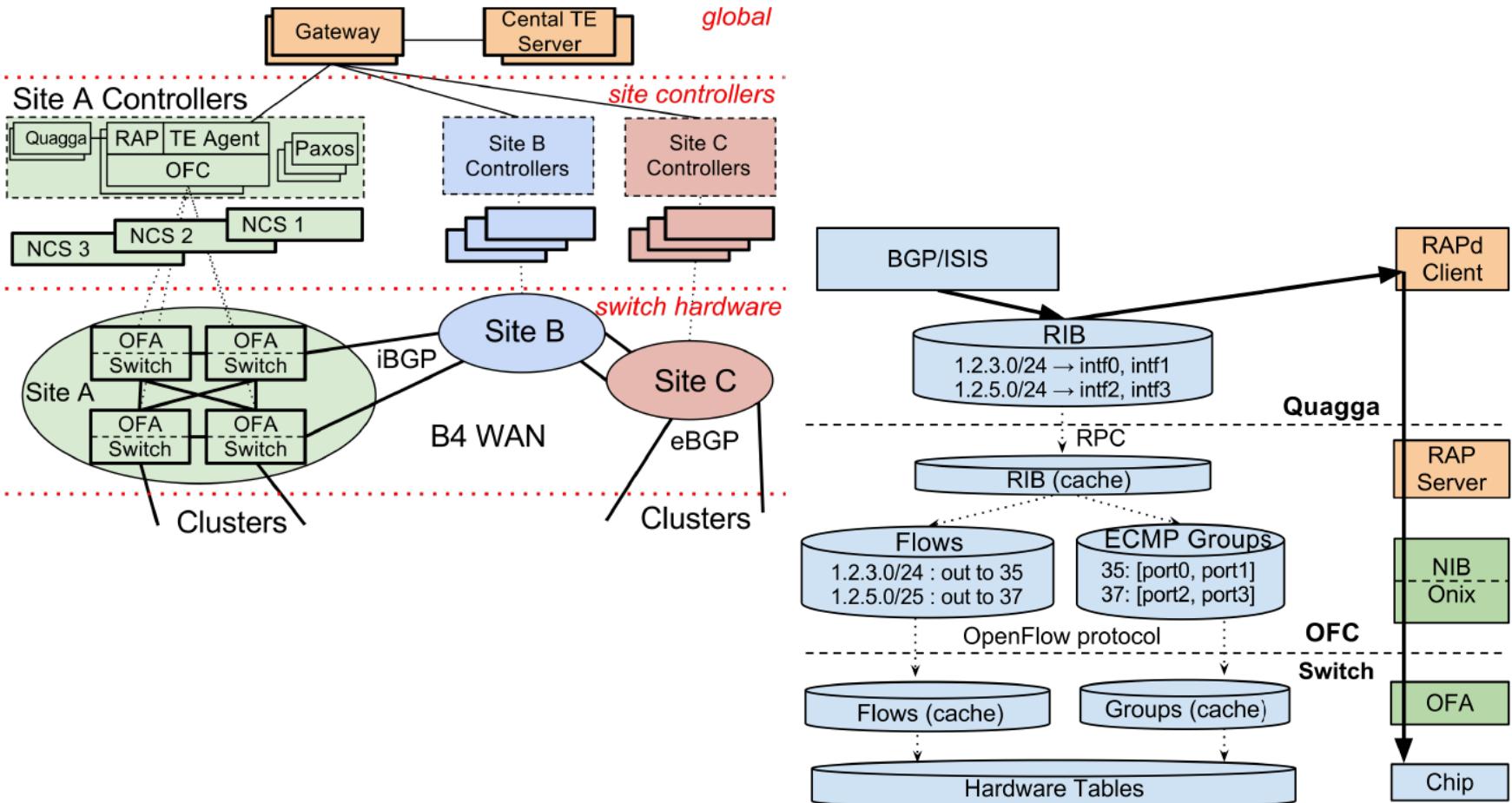
Openflow SDN Architecture



Google's WAN



Openflow in B4





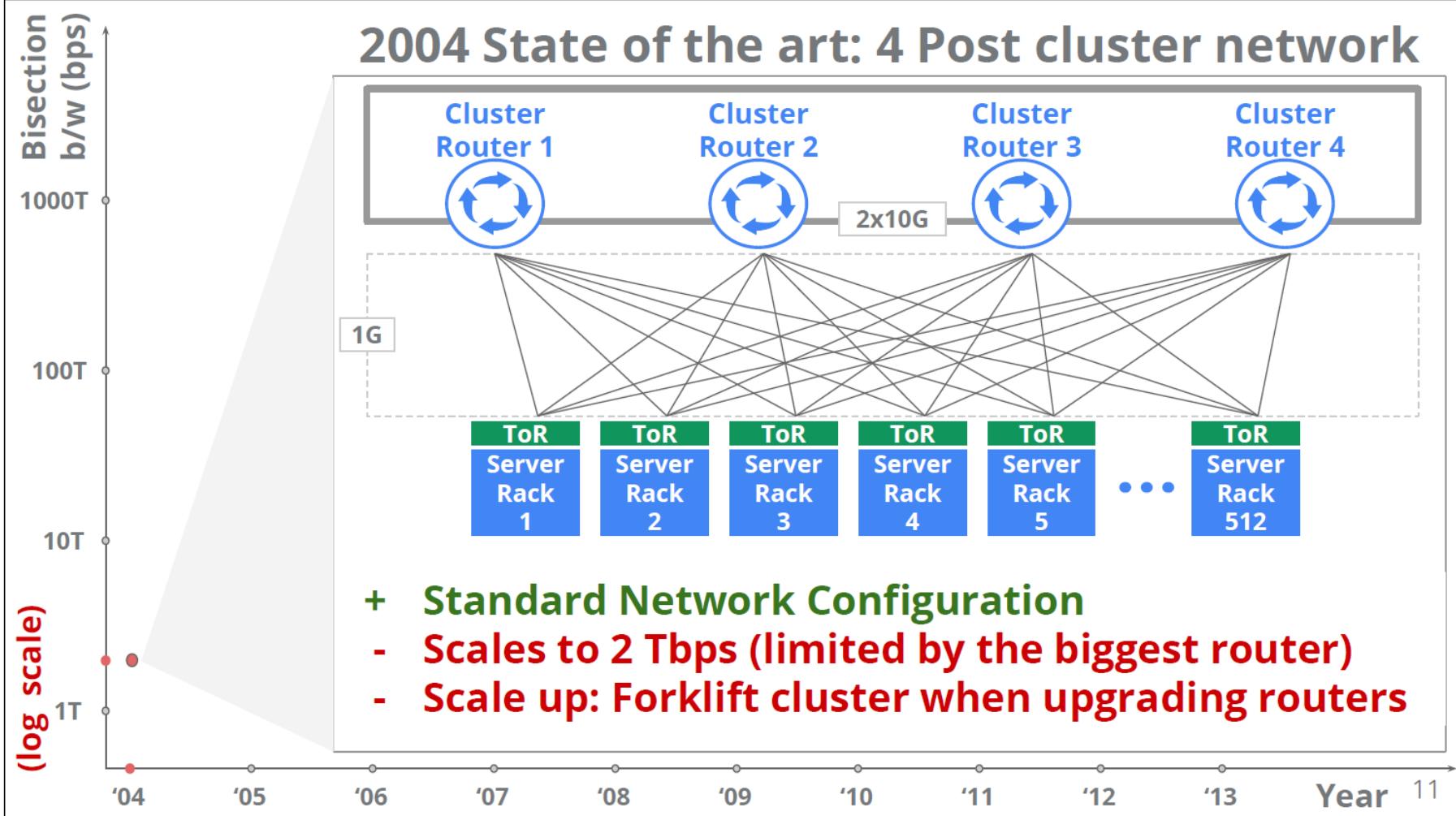
CS2031

Telecommunications II

Clos / Google's Infrastructure



Google's Infrastructure

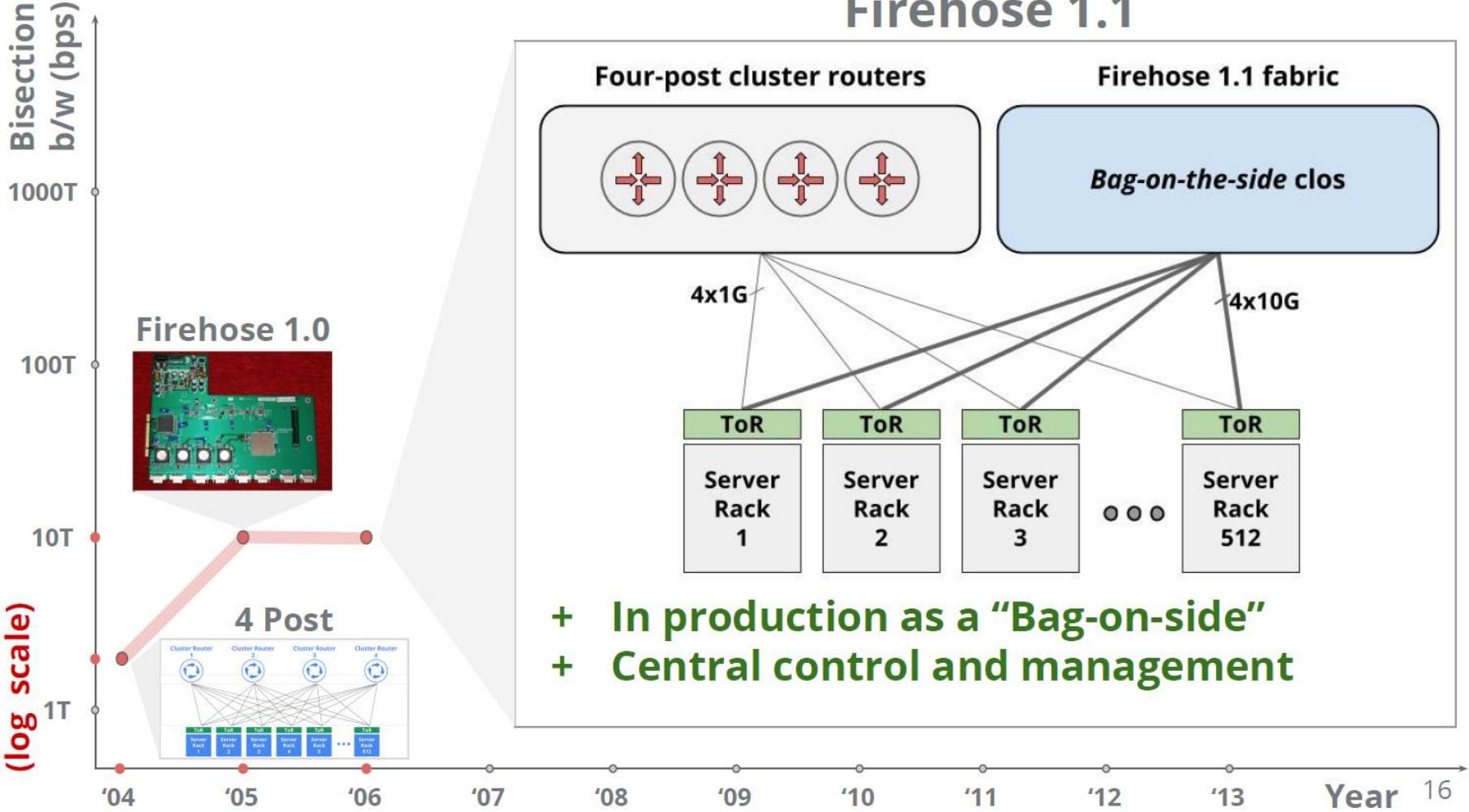


The Obligatory Datacentre Pic

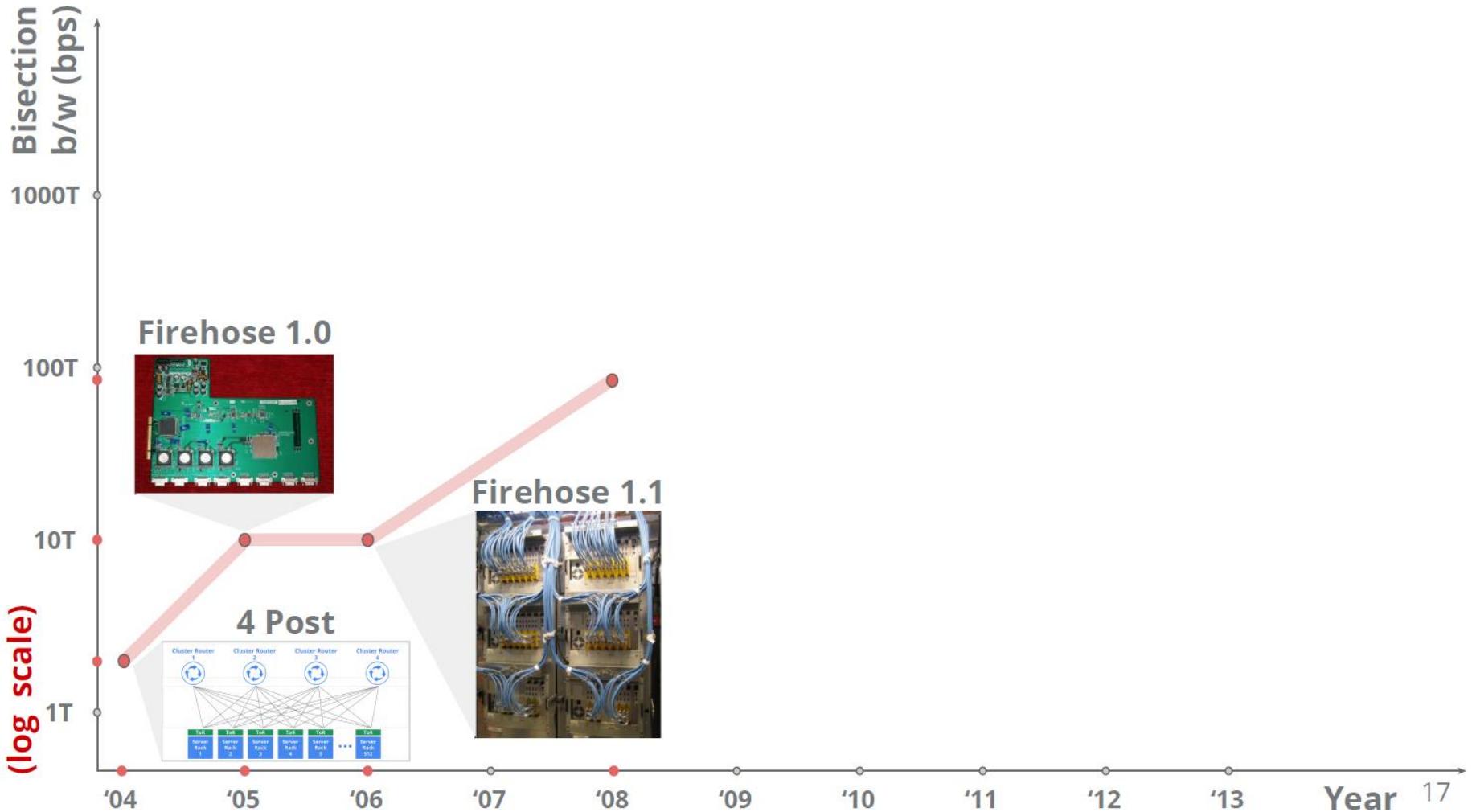


Google's Infrastructure

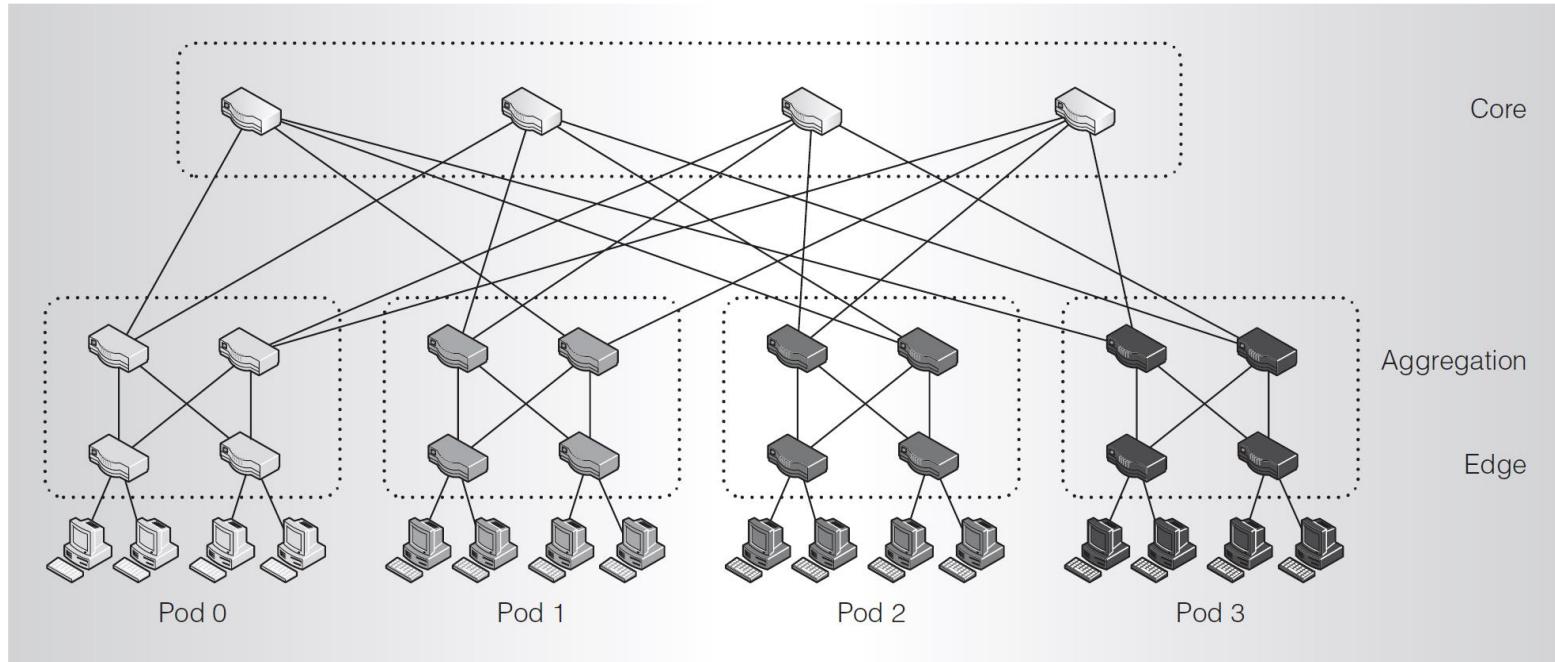
Firehose 1.1



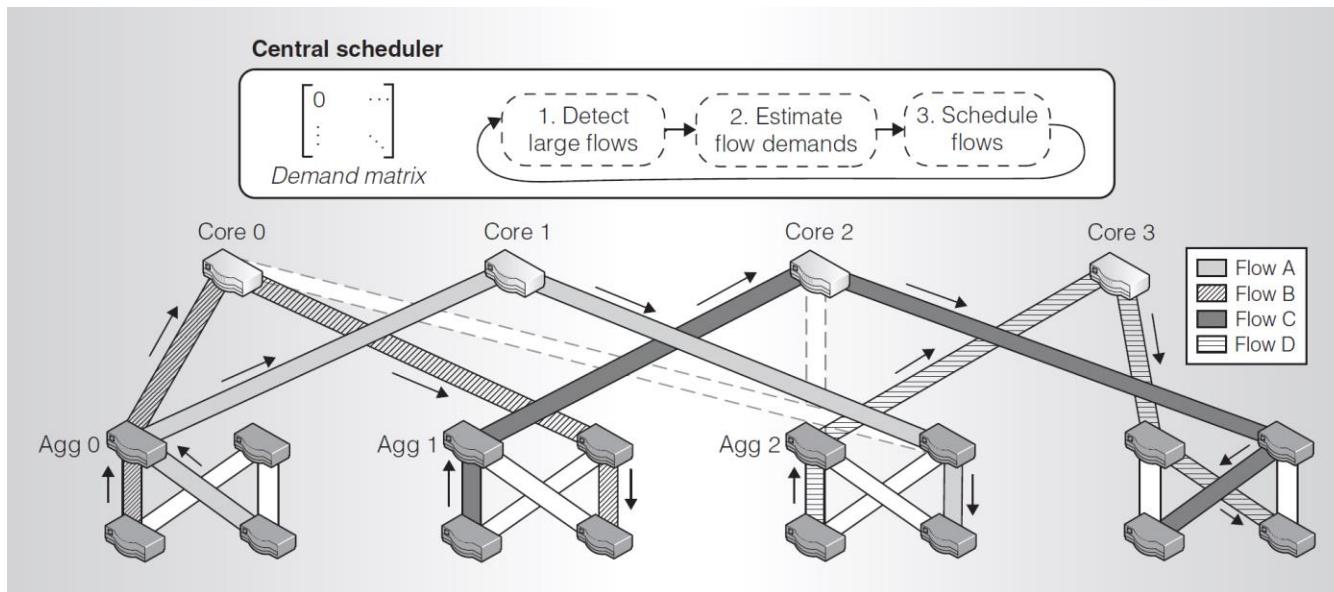
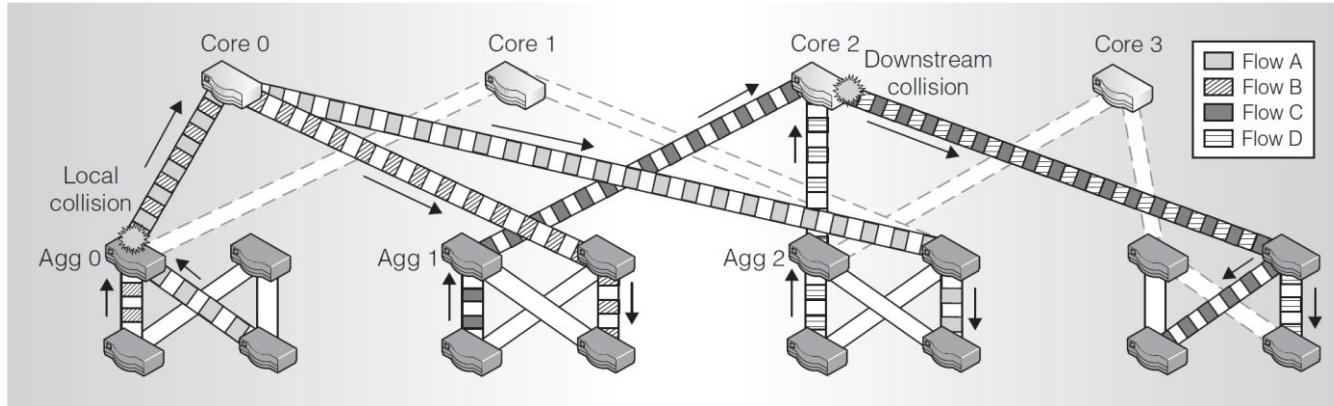
Google's Infrastructure



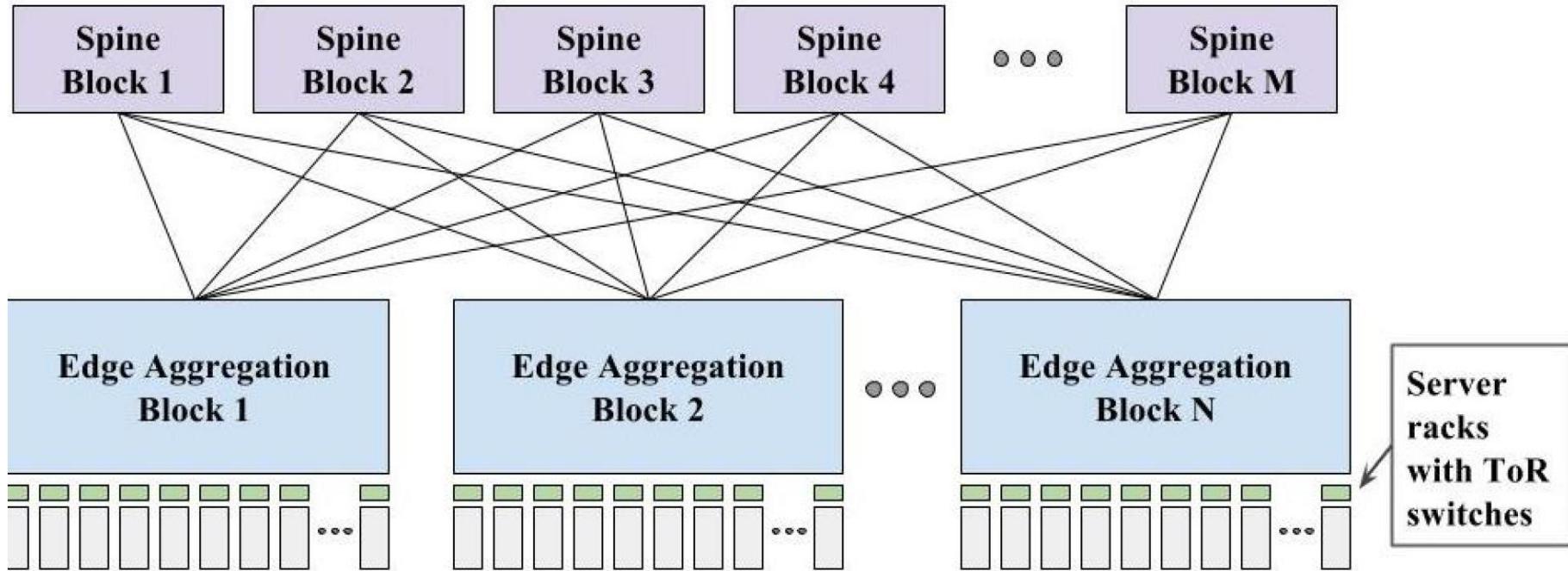
Fat-tree Structure



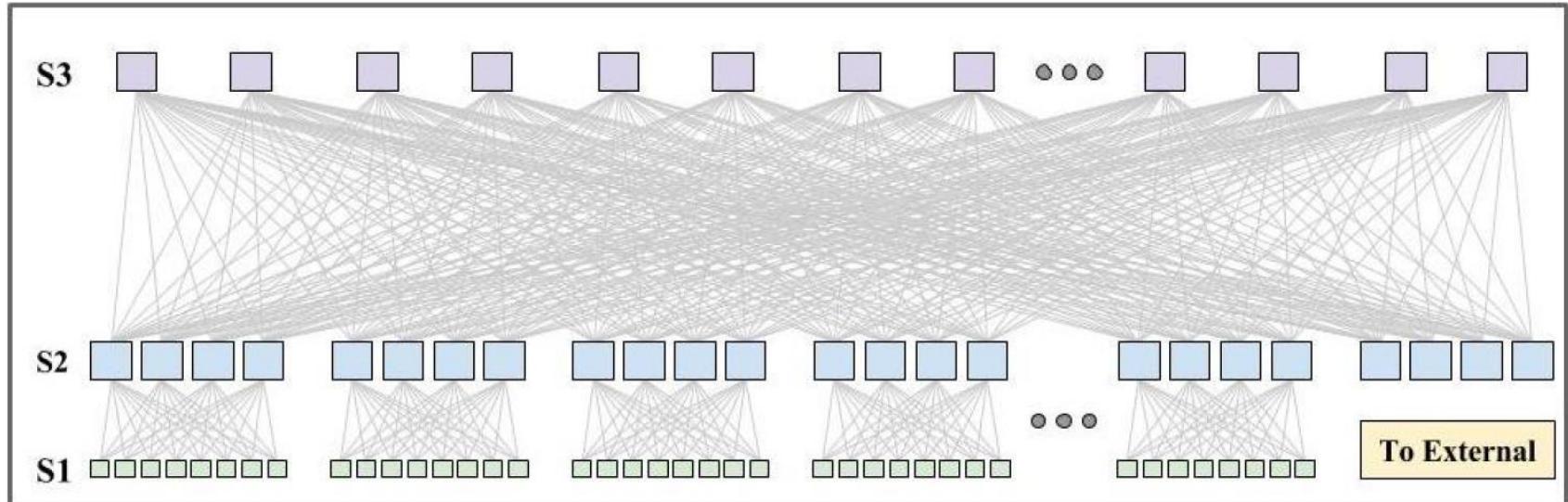
Equal-cost multipath (ECMP) forwarding collisions



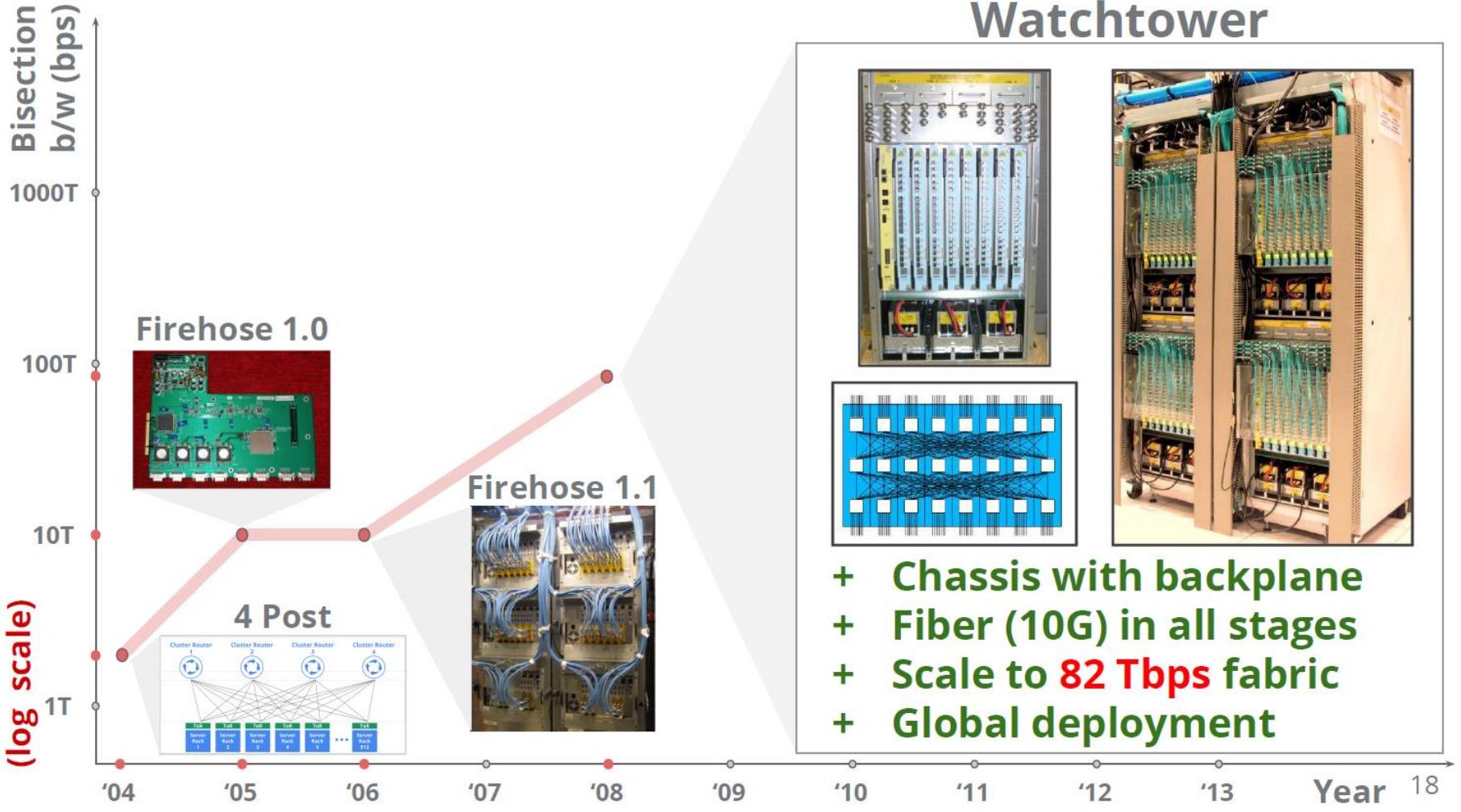
CLOS Architectures



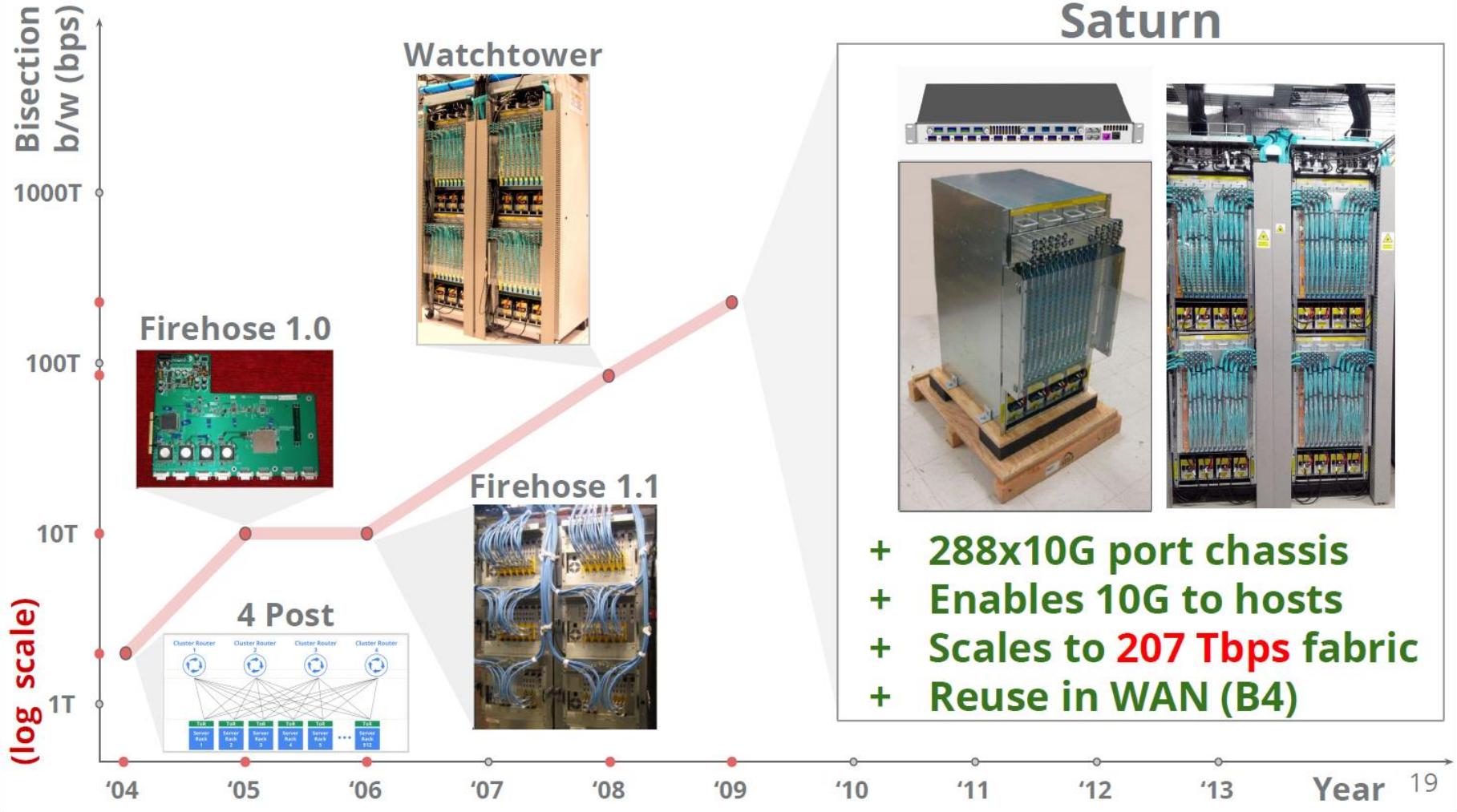
CLOS Architectures



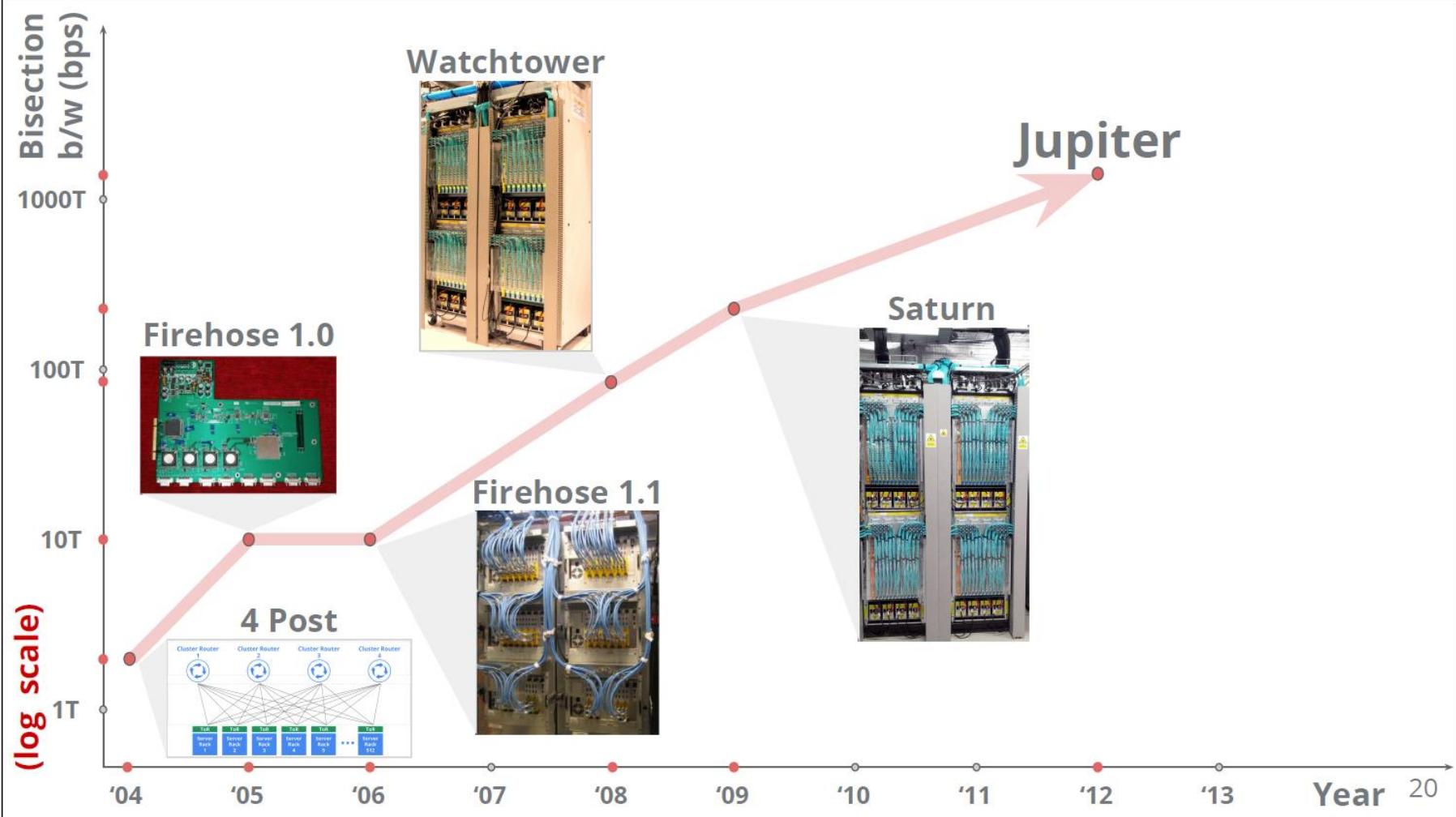
Google's Infrastructure



Google's Infrastructure

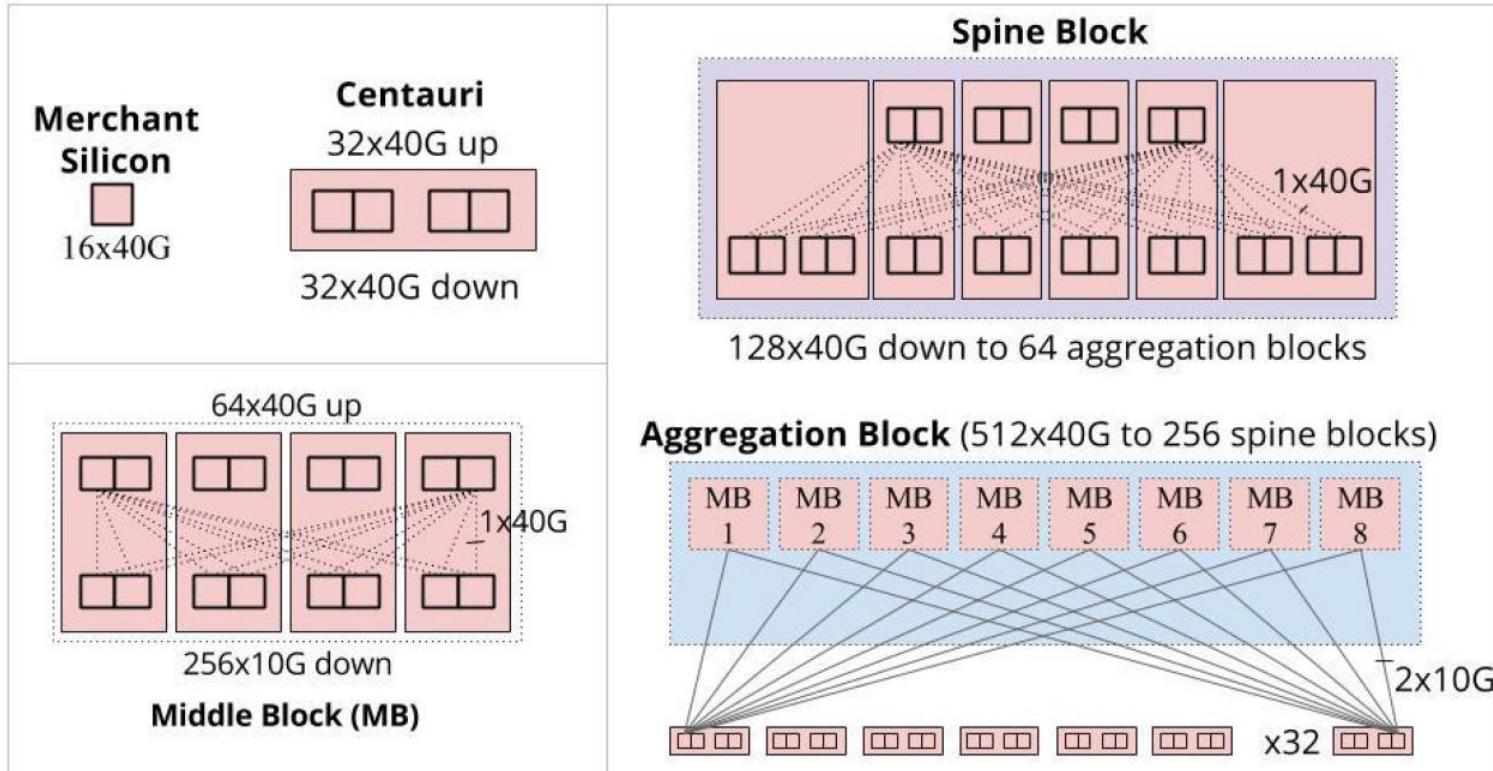


Google's Infrastructure



Google's Infrastructure

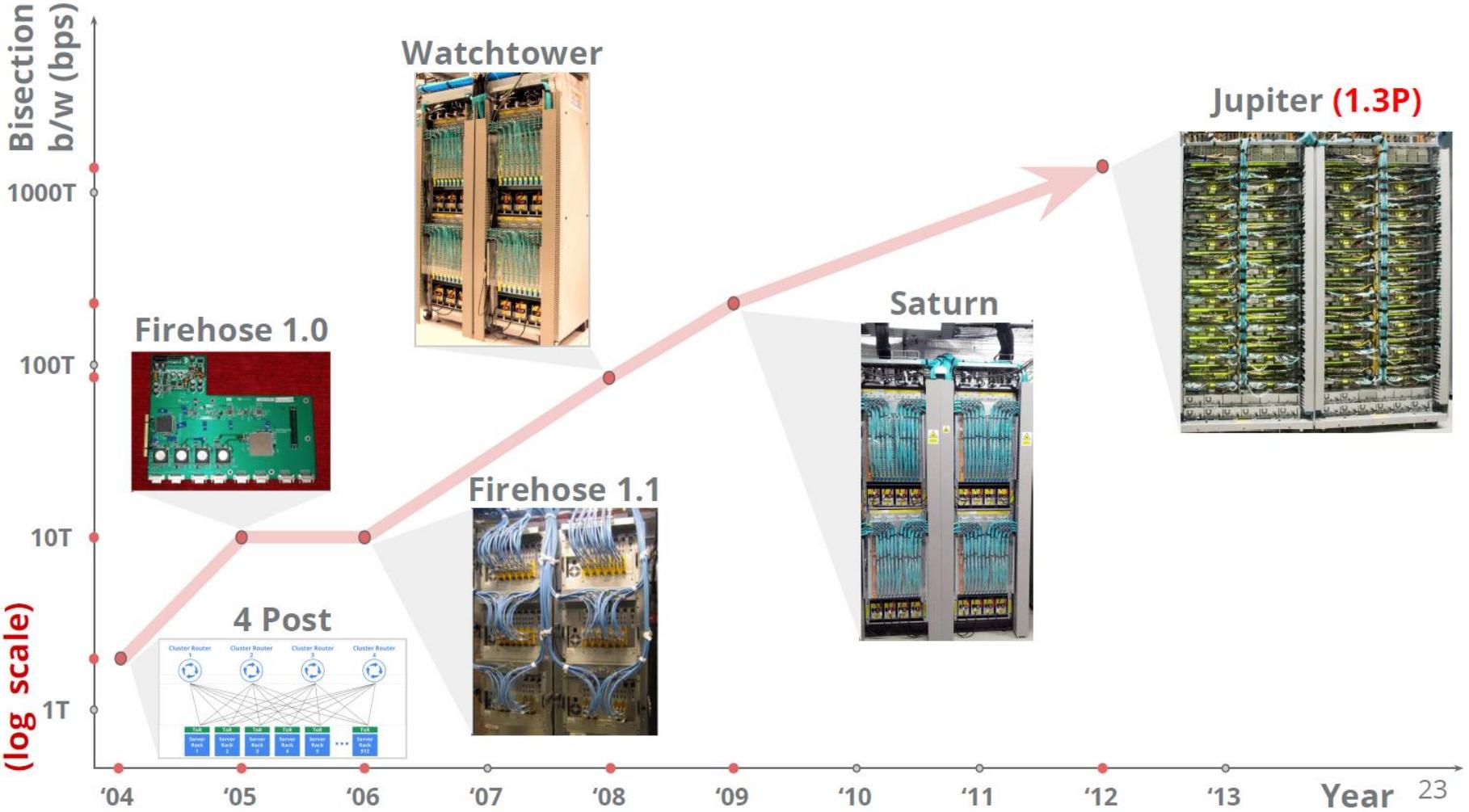
Jupiter topology



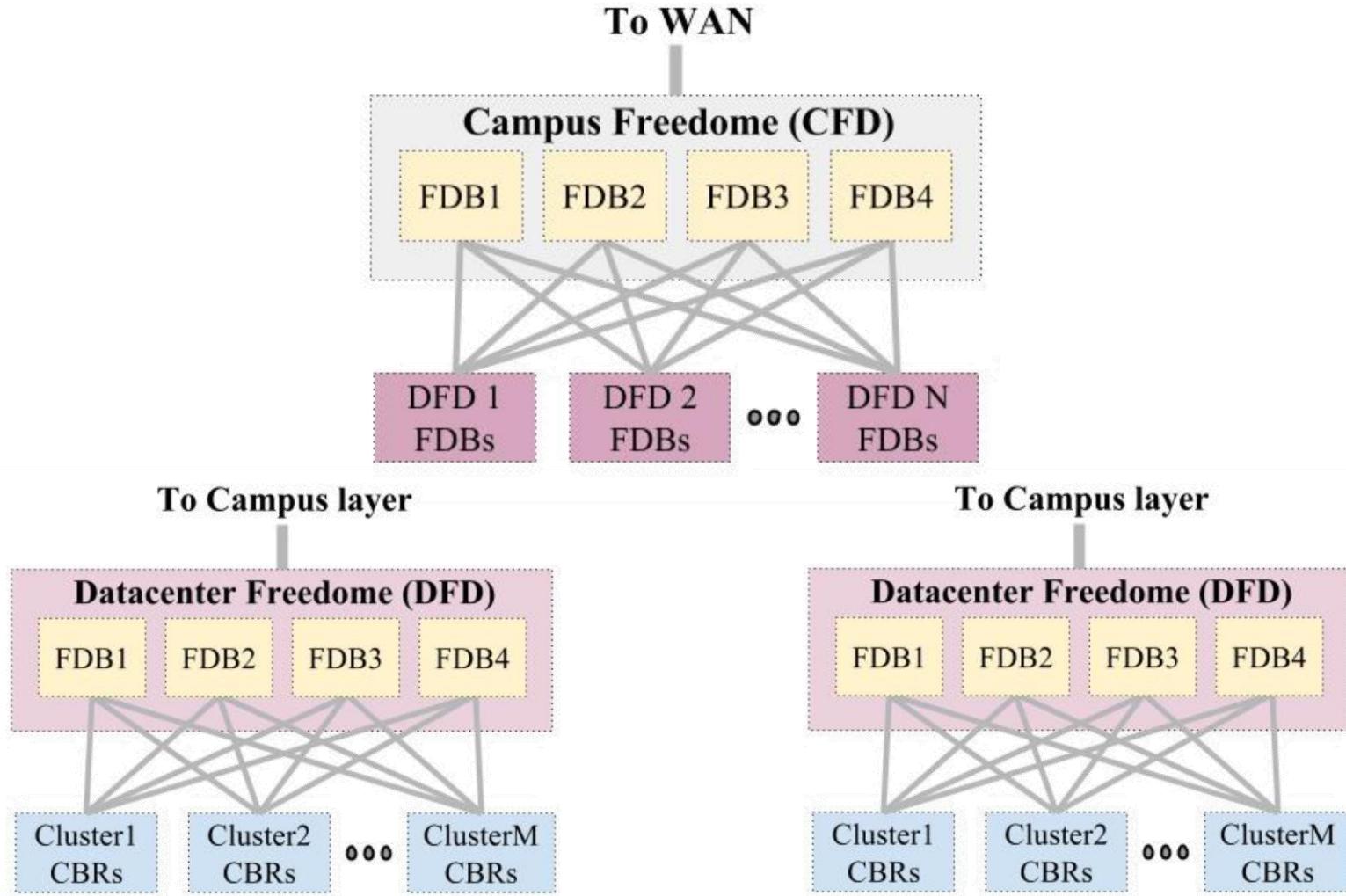
+ Scales out building wide 1.3 Pbps



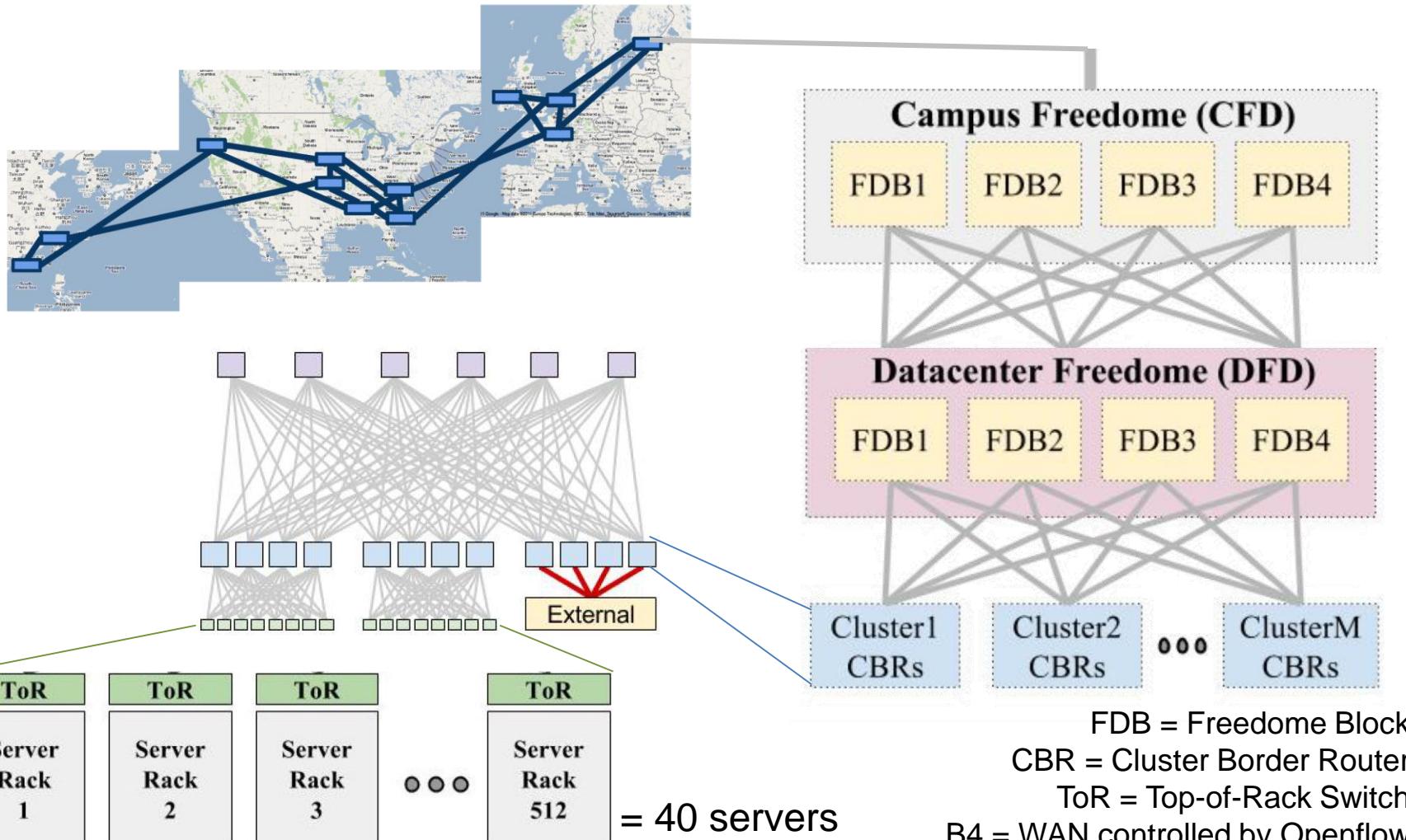
Google's Infrastructure



Google's Infrastructure

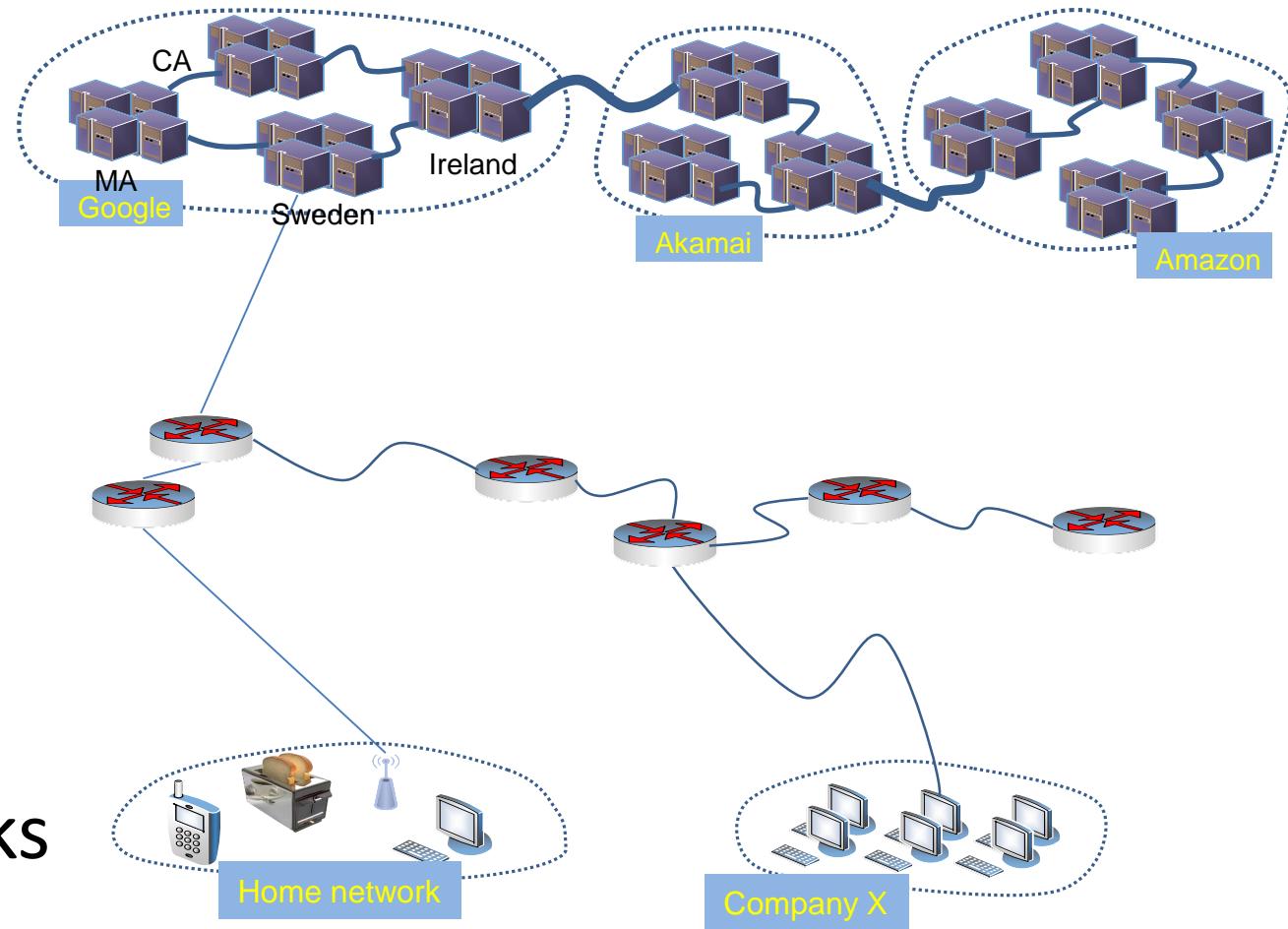


Google's B4 to Jupiter



My view of “future”* networking

- Sets of Datacentres
- Traditional Internet
- Edge networks



*or current?



9.12. & 16.12.
No Lectures & Tutorials



Akamai & IPv6

- Akamai will offer IPv6 services to its entire customer base in April - a long-awaited move that will be a major boon to the adoption rate of the next-generation internet protocol.

Carrying between 20% and 30% of the internet's web traffic on any given day, Akamai is the world's largest content delivery network (CDN). Akamai's engineering team has been working for two years to upgrade its 95,000 servers in 71 countries connected by 1,900 networks to support IPv6.

- Mar 27, 2012
<http://www.techworld.com/news/apps/ipv6-akamai-launch-new-internet-protocol-service-in-april-3347188/>

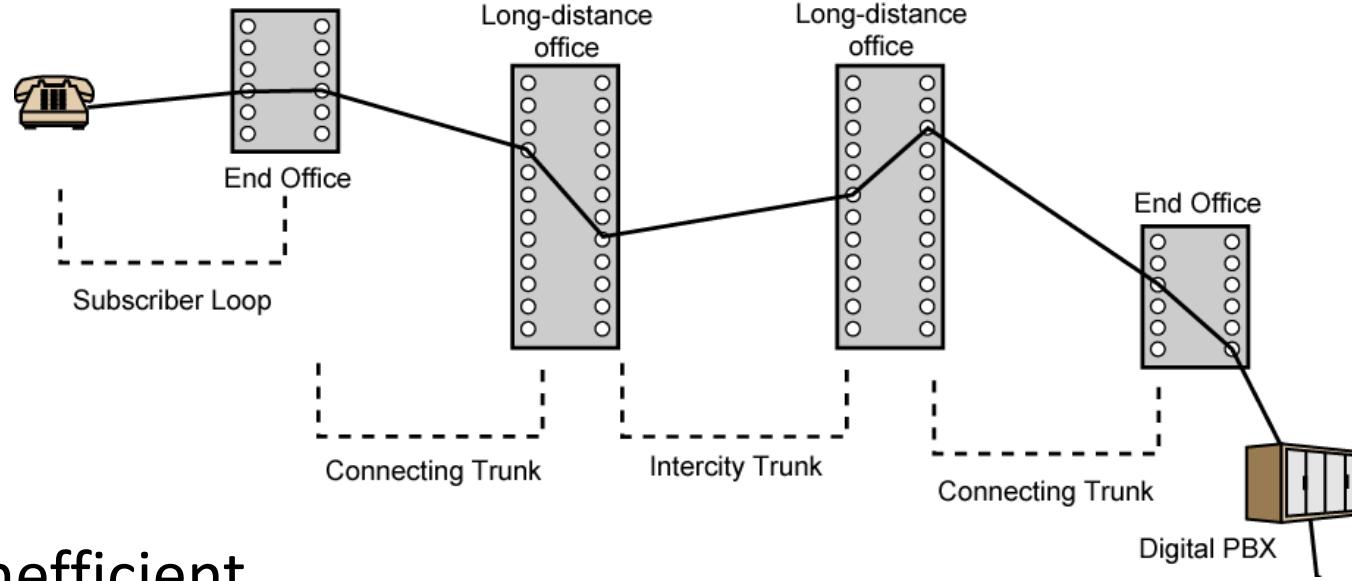


CS2031 Telecommunications II

Circuit Switching & Packet Switching

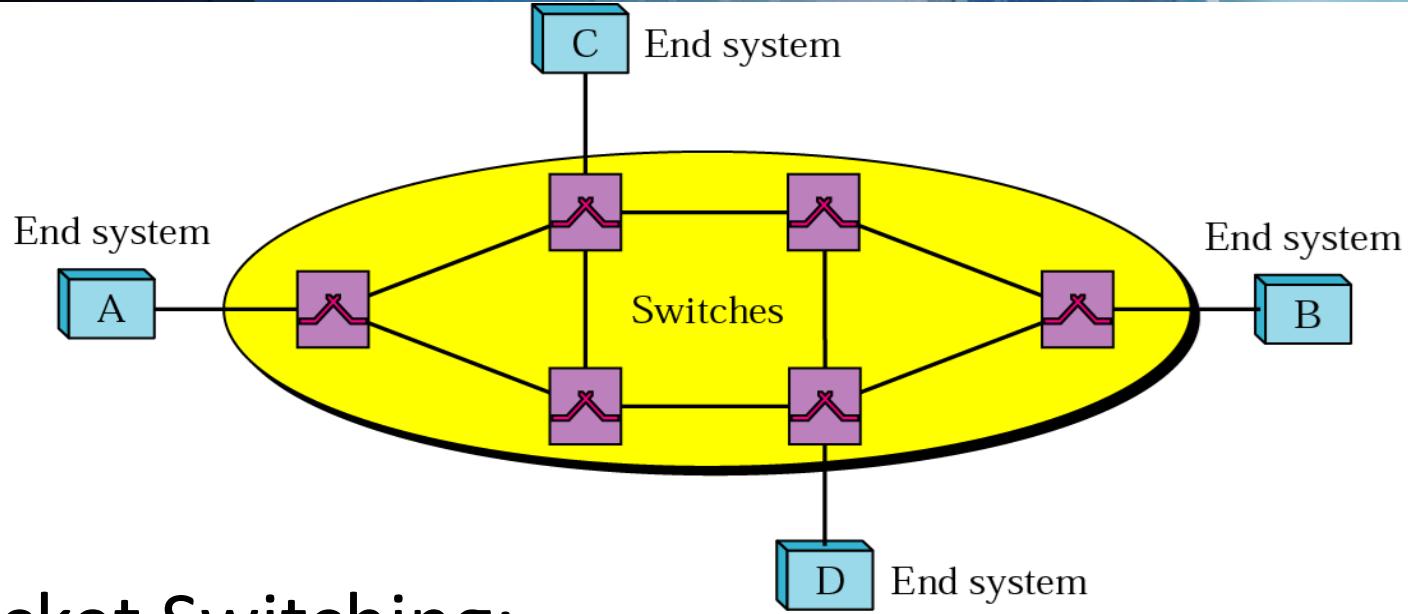


Public Circuit Switched Network



- Inefficient
 - Channel capacity dedicated for duration of connection
 - If no data, capacity wasted
- Set up of connection takes time
- Once connected, transfer is transparent

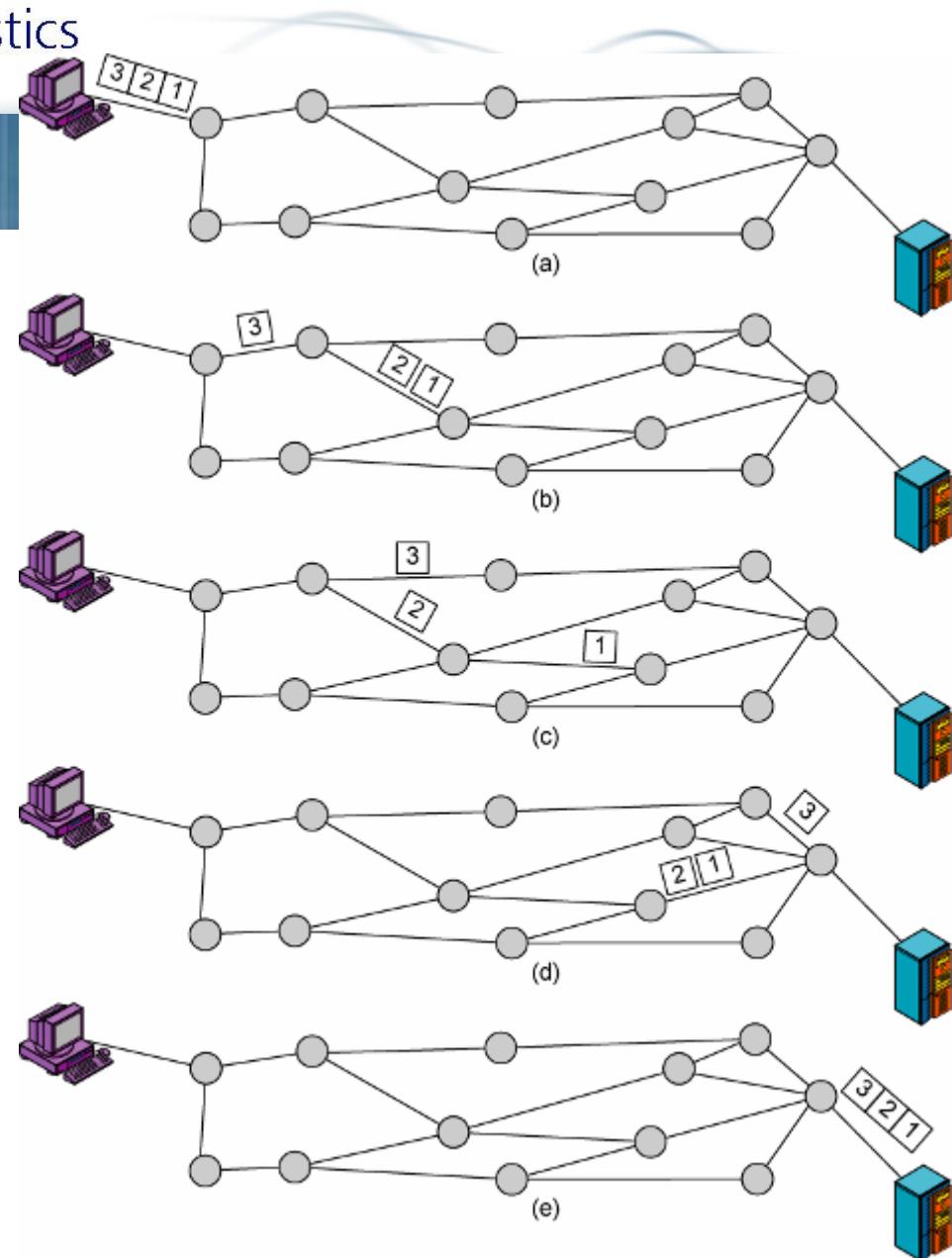
Switched Networks



- **Packet Switching:**
 - Switching decisions are made on individual packets
- **Virtual Circuit Switching:**
 - A circuit is setup explicitly for individual connections

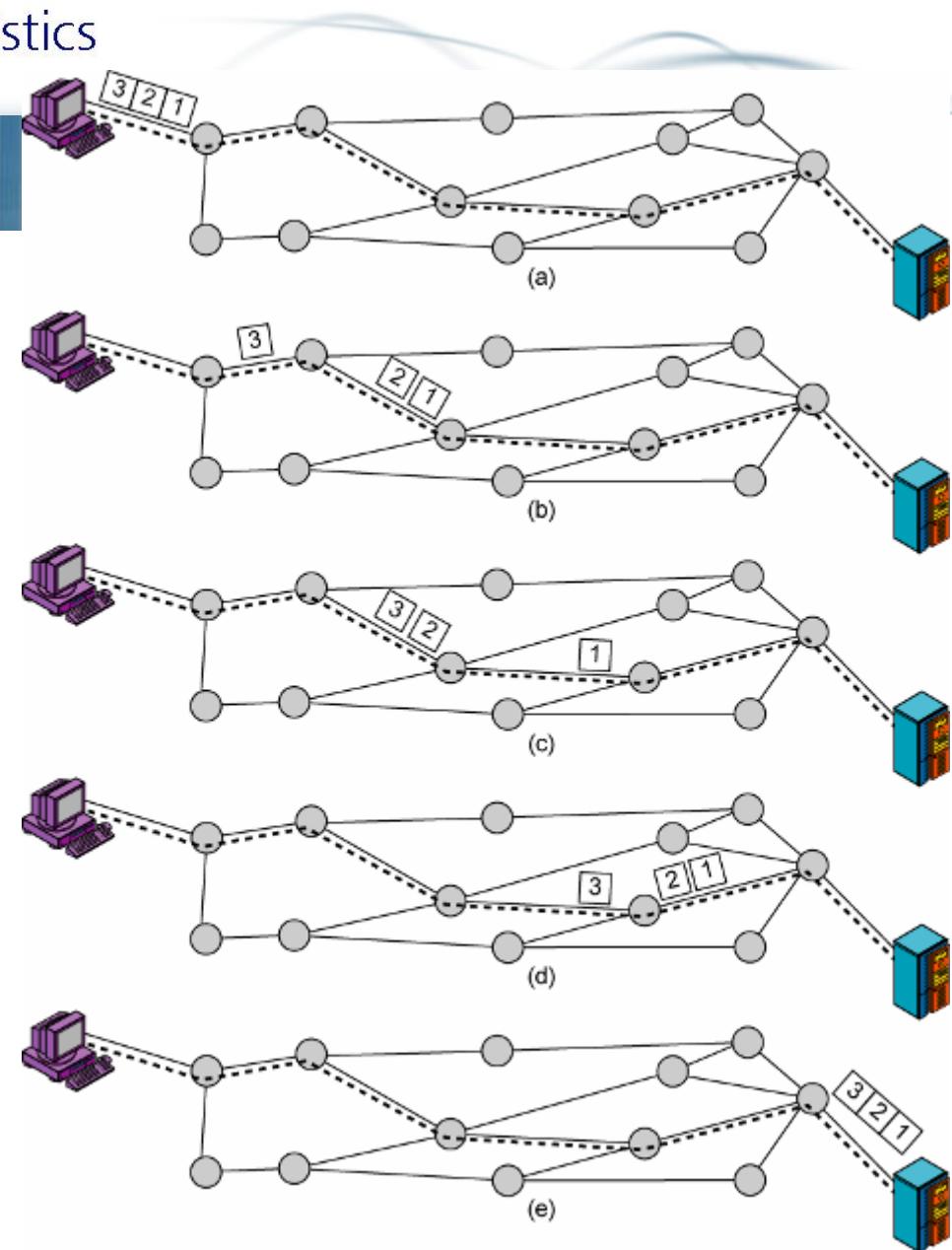
Packet switching

- Frames can be transferred over different paths in the network
- Reliability is generally delegated to higher layers
- Order is not necessarily maintained



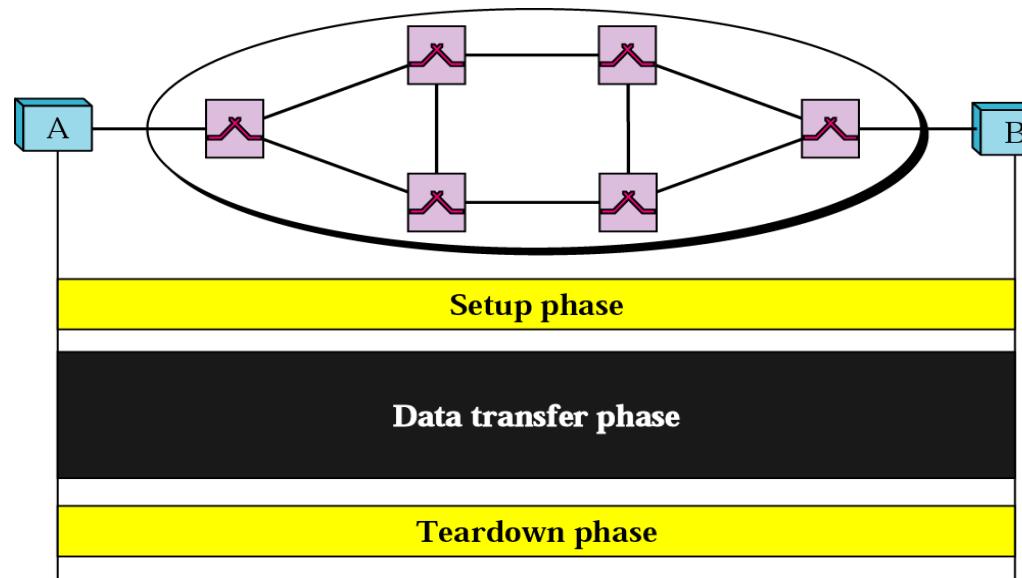
Virtual Circuits

- Connection-oriented communication
- Connection is established before communication
- The network maintains order

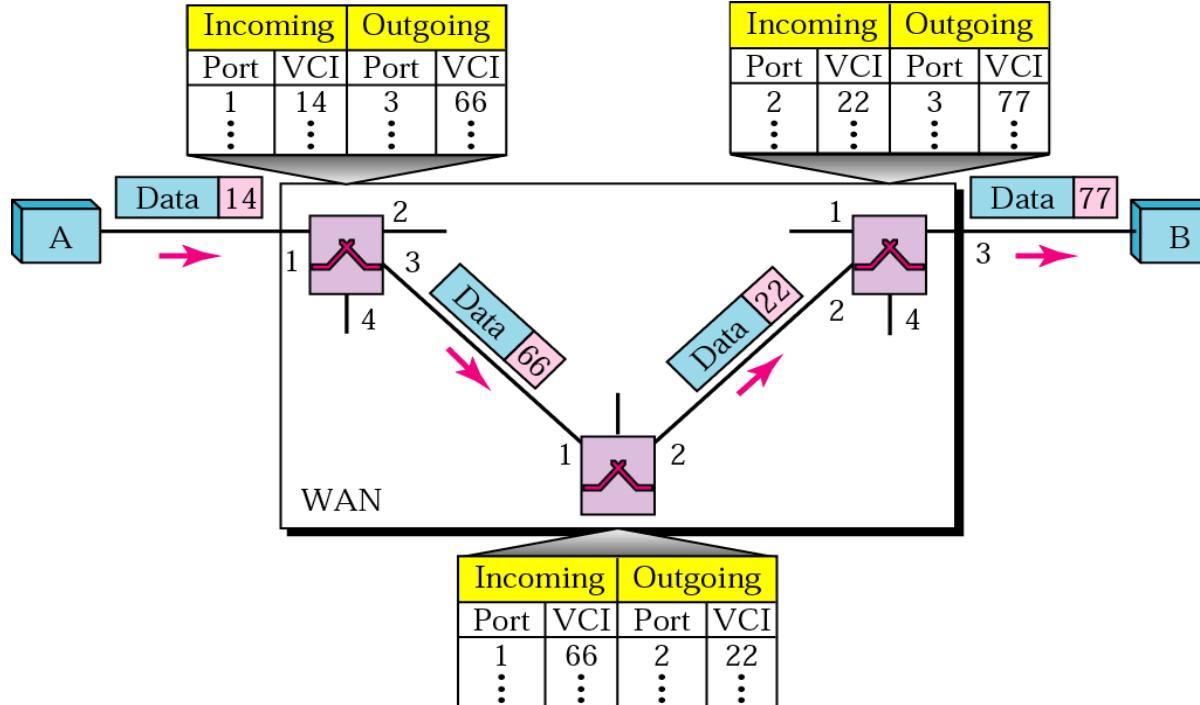


Phases for Virtual Circuit Communication

- Three phases
 - Connection setup
 - Data Transfer
 - Connection termination



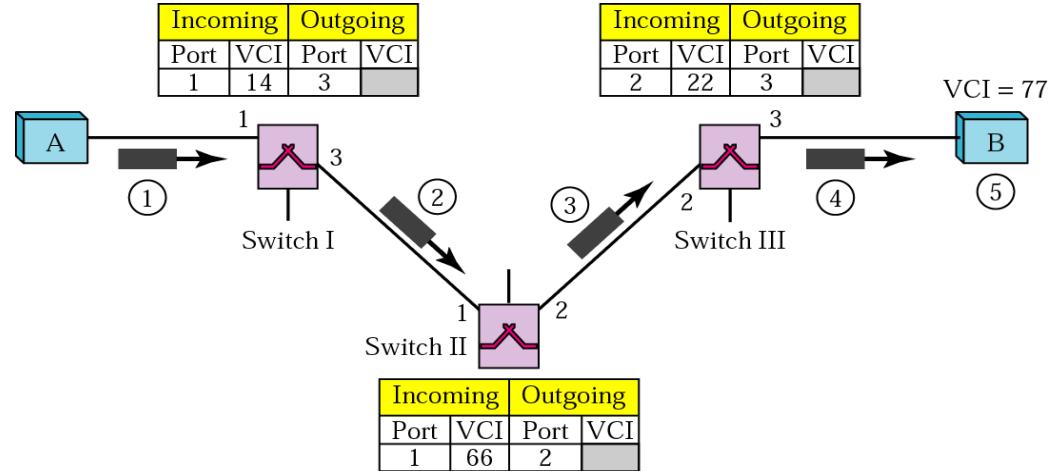
Virtual Circuit Switching



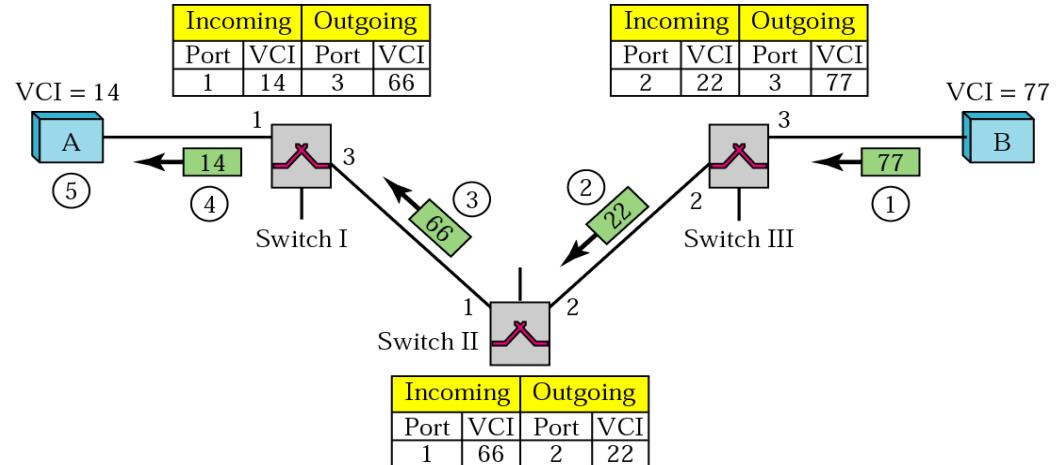
- Every switch maintains a table
 - For duration of communication one entry for incoming and outgoing line
 - Incoming and outgoing line are identified by port number and virtual circuit identifier

Setup Phase

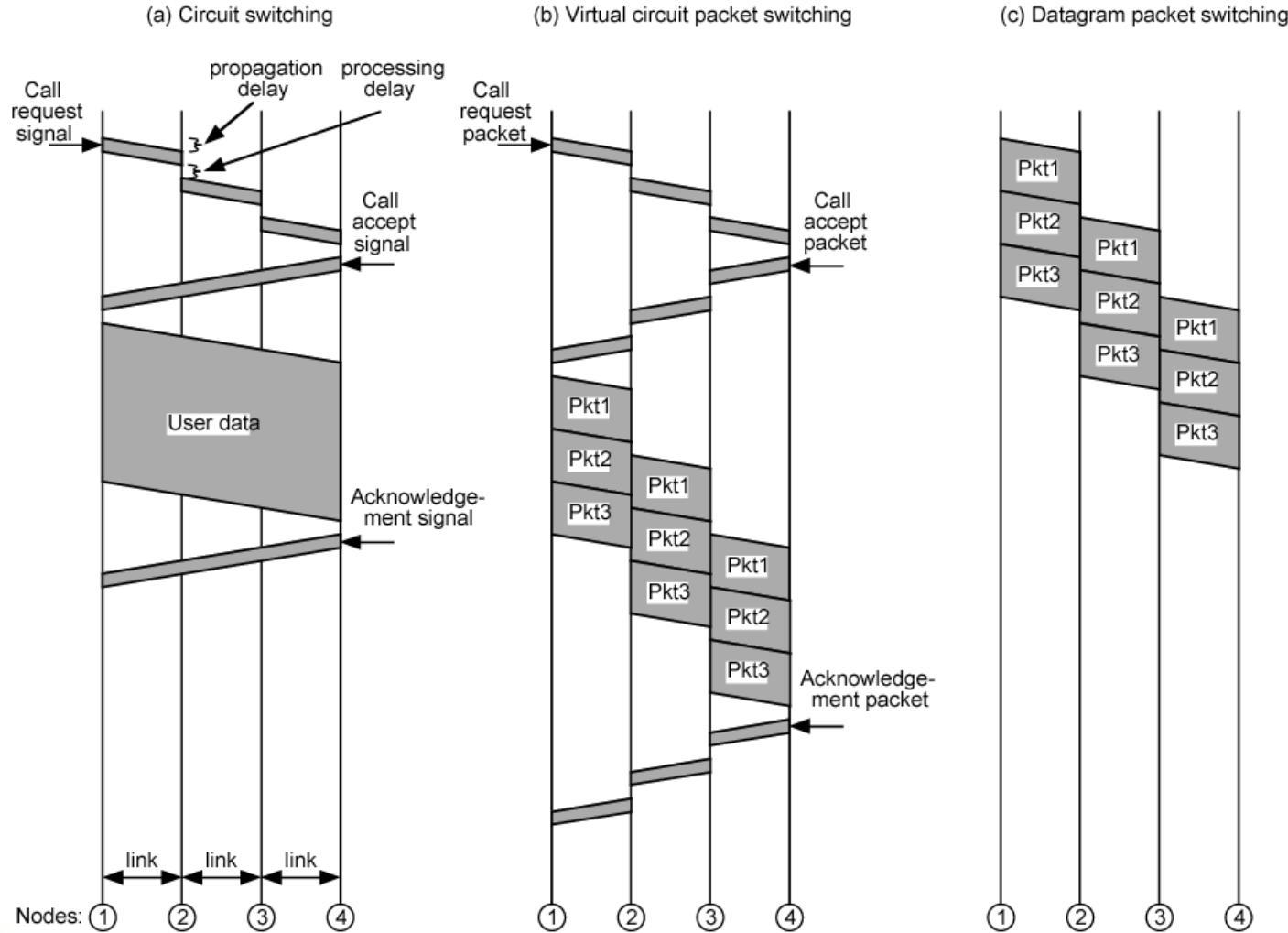
- Setup request



- Setup acknowl.



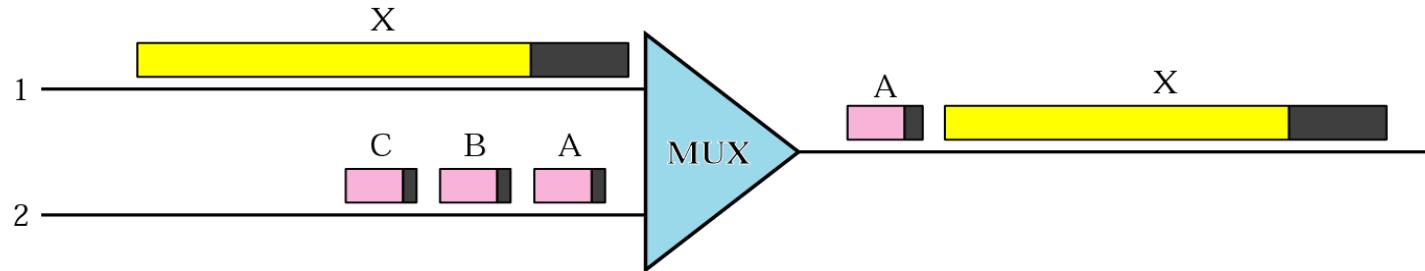
Event Timing



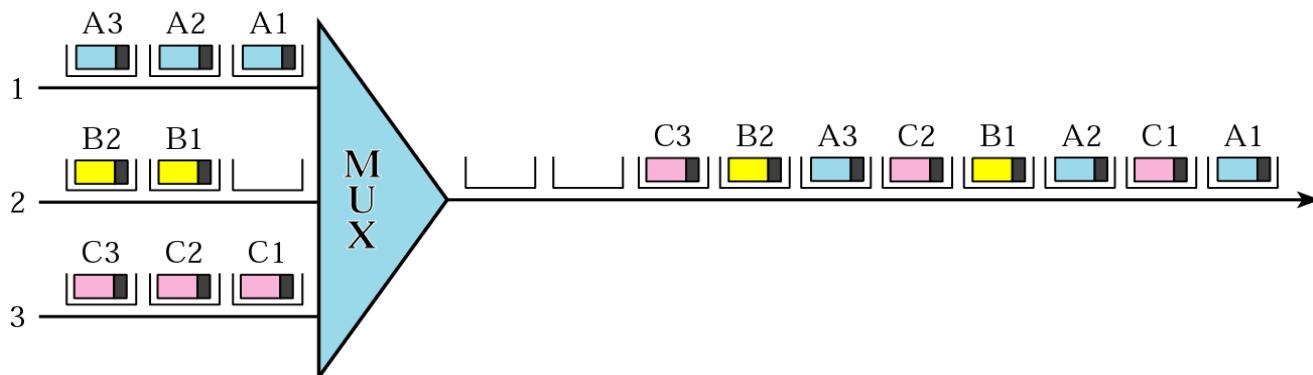
Asynchronous Transfer Mode (ATM)

- Example of virtual circuit switching
 - Cell-Switching
- Similarities between ATM and packet switching
 - Transfer of data in discrete chunks
 - Multiple logical connections over single physical interface
- In ATM flow on each logical connection is in fixed sized packets called cells
- Minimal error and flow control
 - Reduced overhead

Motivation for ATM

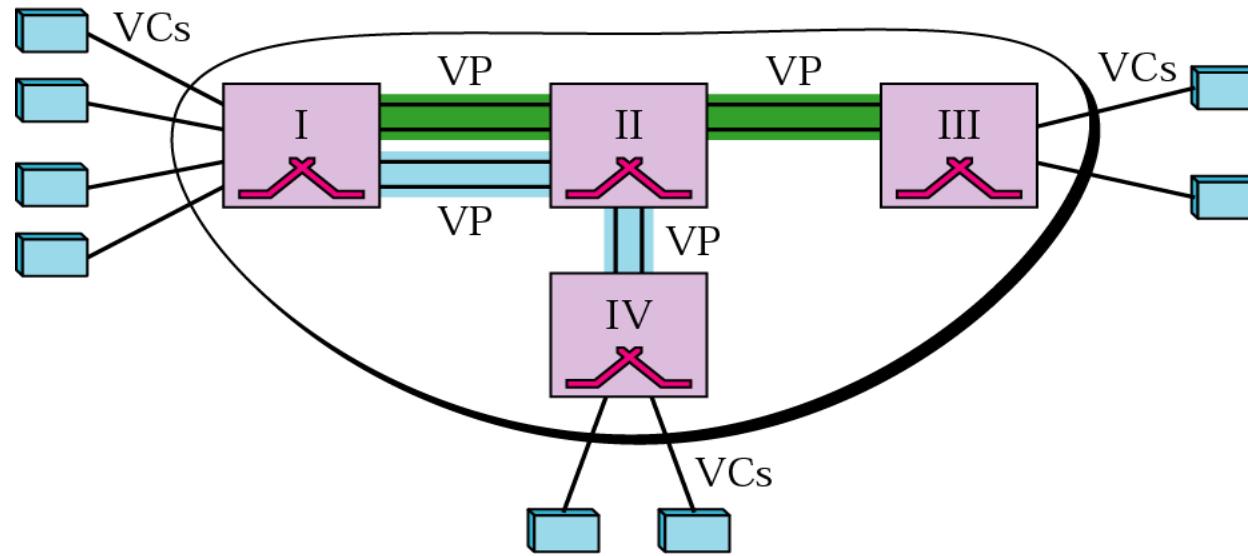


- Frames at a switch may be handled in any order and occupy switch for underspecified time



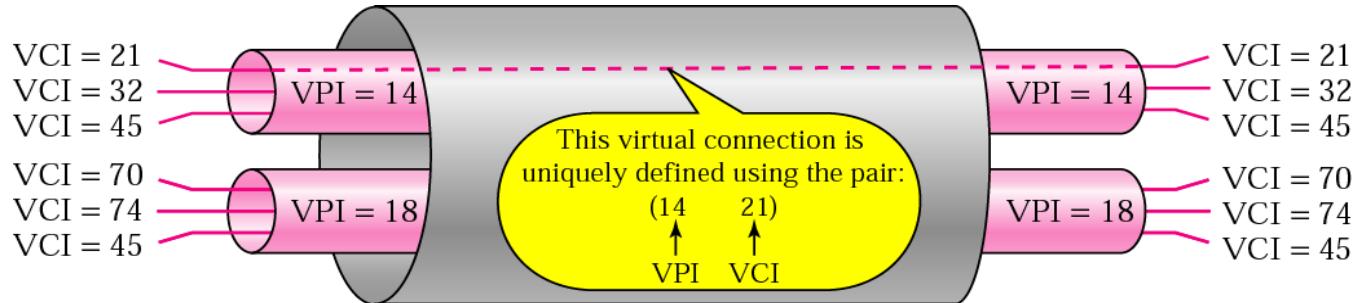
- Small, fixed-size frames allow simple, fast switches

Virtual Circuits / Virtual Paths

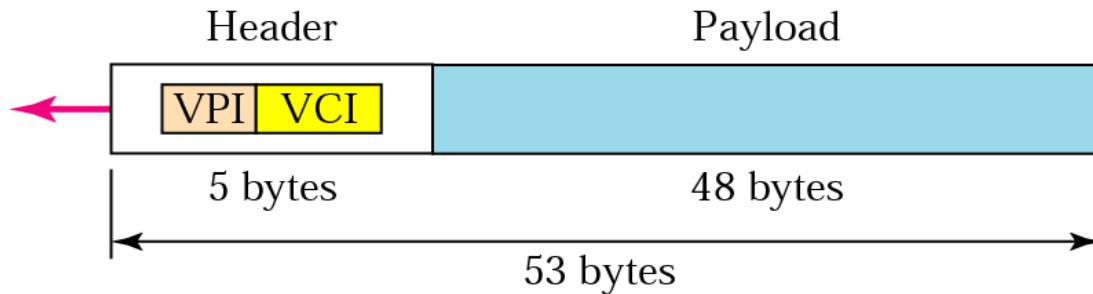


- Virtual circuits are collected into virtual paths

ATM Packet

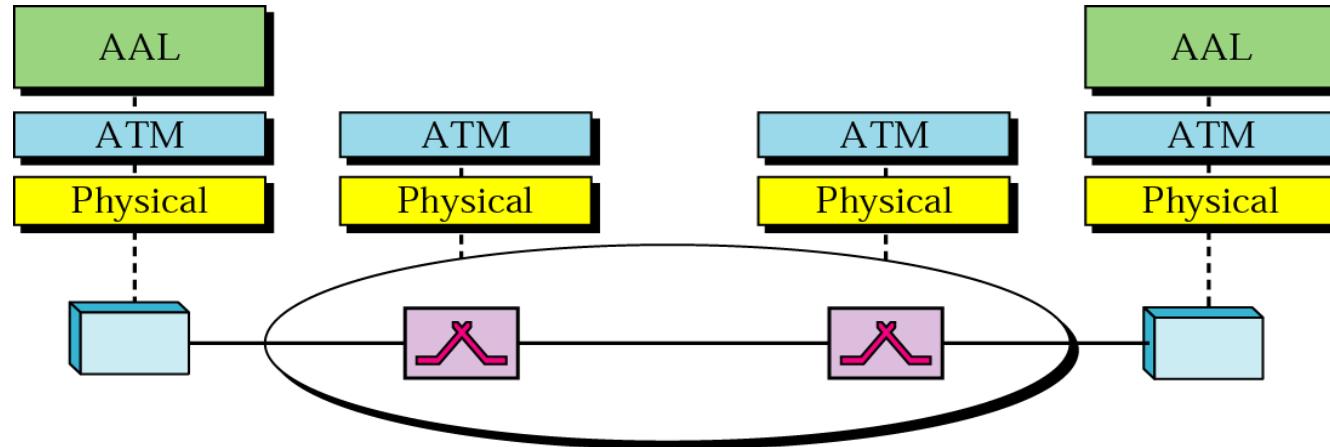


- Connection is specified by combination of Virtual Path ID and Virtual Circuit ID



- Every frame is exactly 53 bytes

Application Adaptation Layer (AAL)



- ATM defined a number of AALs for various purposes (each has its own header format):
 - AAL1: Constant bit rate e.g. multimedia
 - AAL2: Variable-data-rate
 - AAL3/4: Connection-oriented data services
 - Sequencing and Error Control
 - AAL5: Simple and efficient adaptation layer (SEAL)

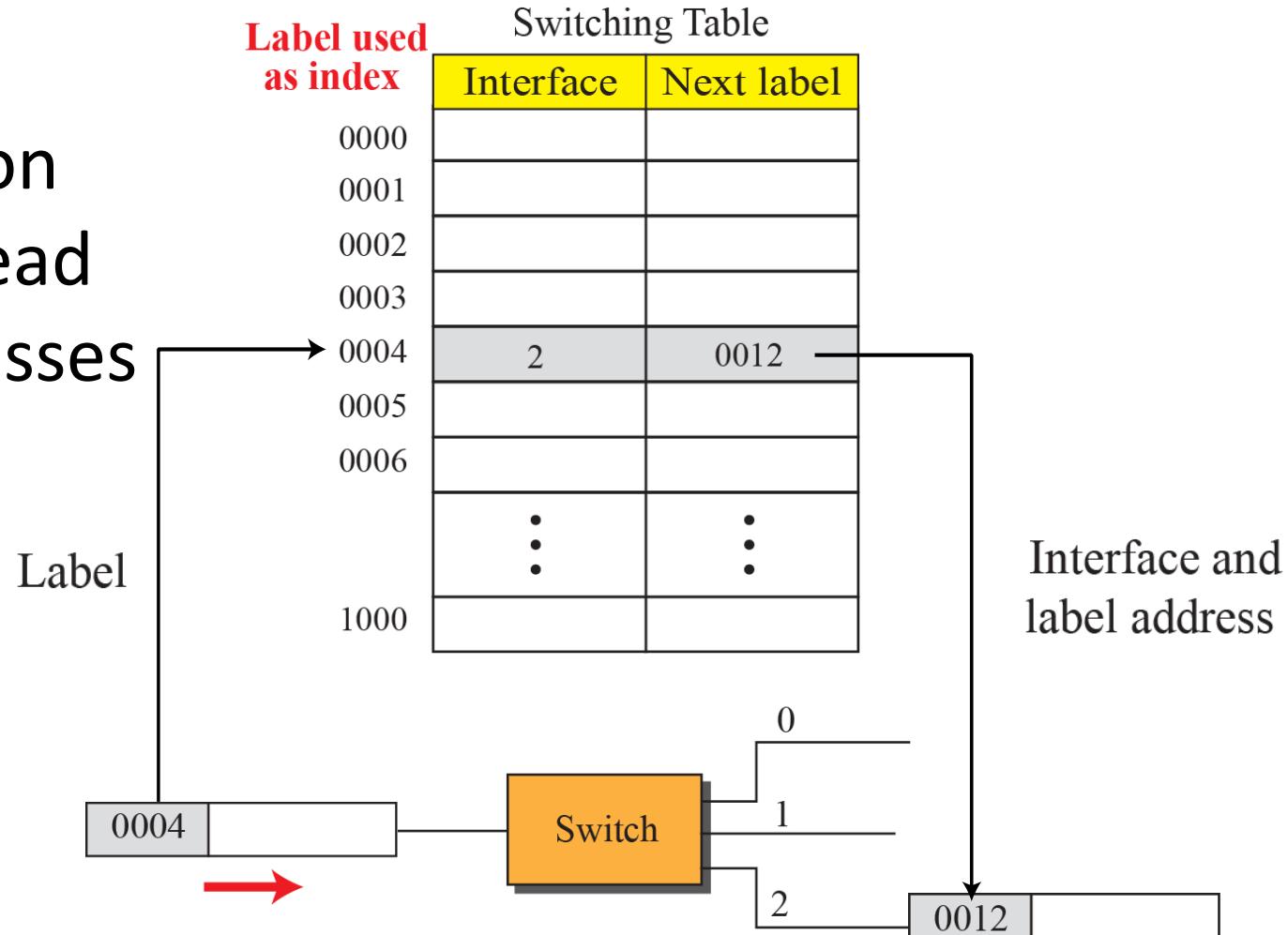
ATM – It Didn't Happen

- From Tanenbaum:

“ATM was going to solve all the world’s networking and telecommunications problems by merging voice, data, cable television, telex, telegraph, carrier pigeons, ...”
- It didn’t happen:
 - Bad Timing
 - Technology
 - Implementation
 - Politics

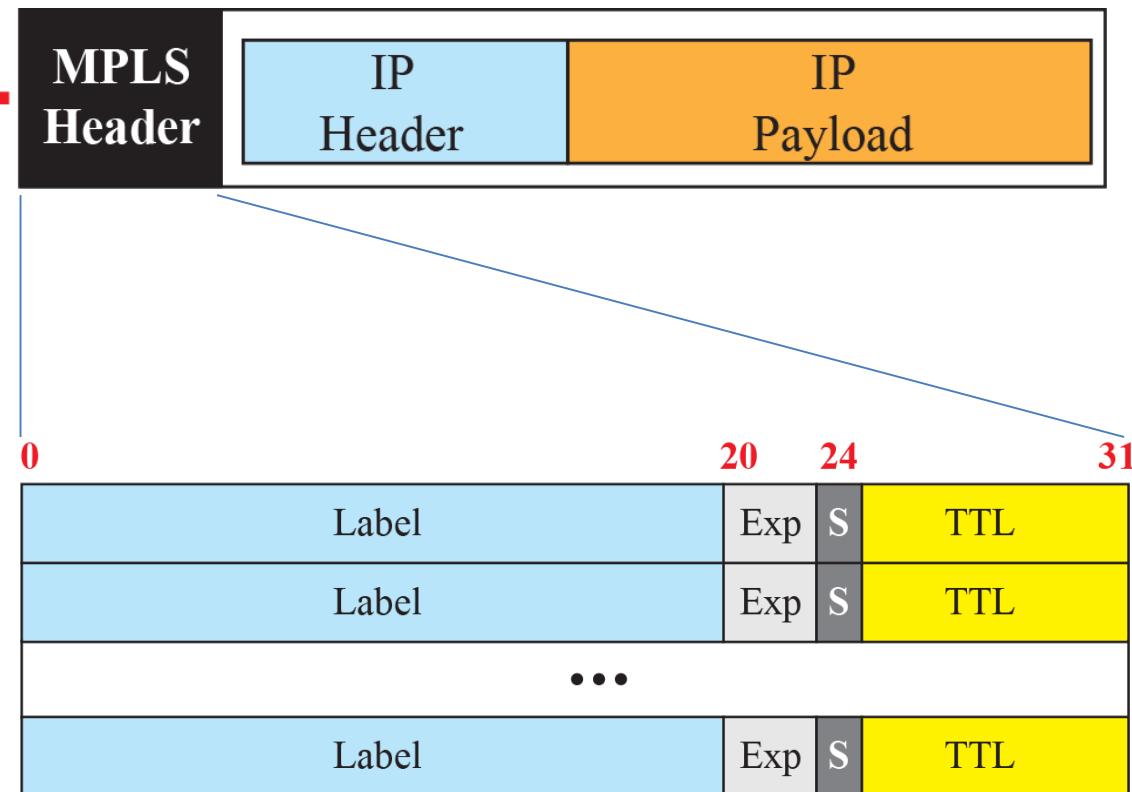
Multiprotocol Label Switching (MPLS)

- Enables switching on labels instead of IP addresses

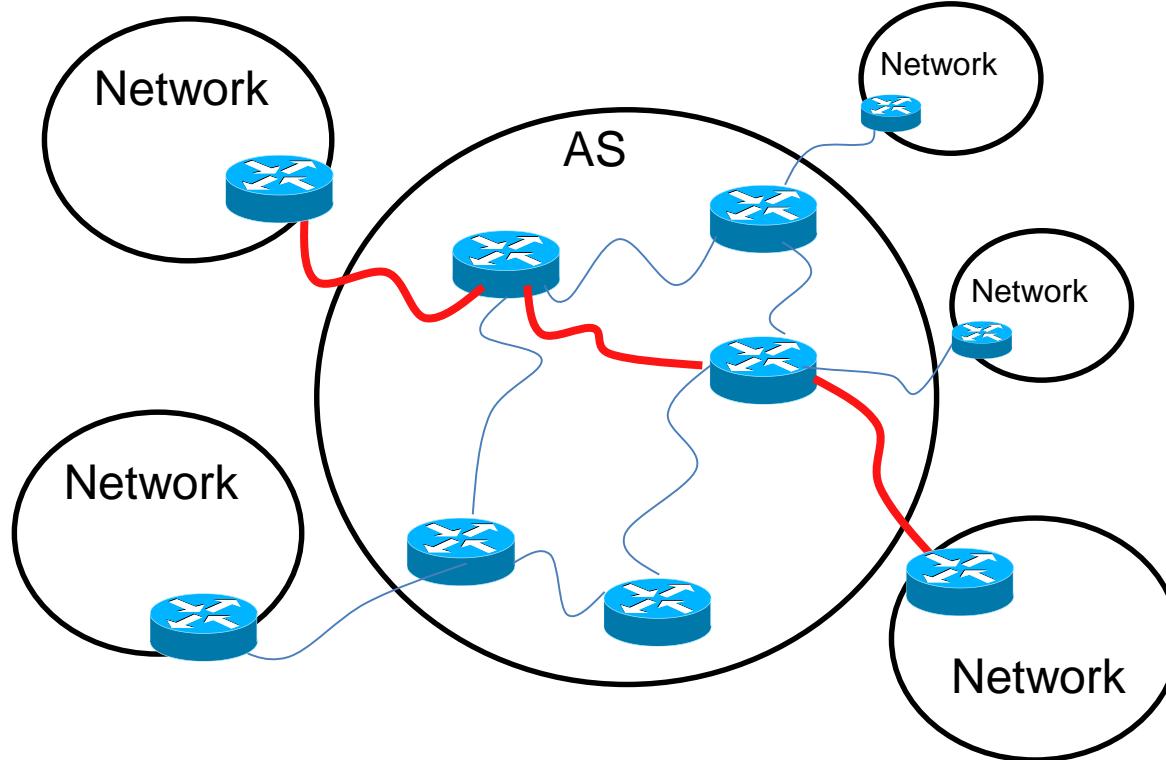


MPLS Header

- MPLS header as stack of labels

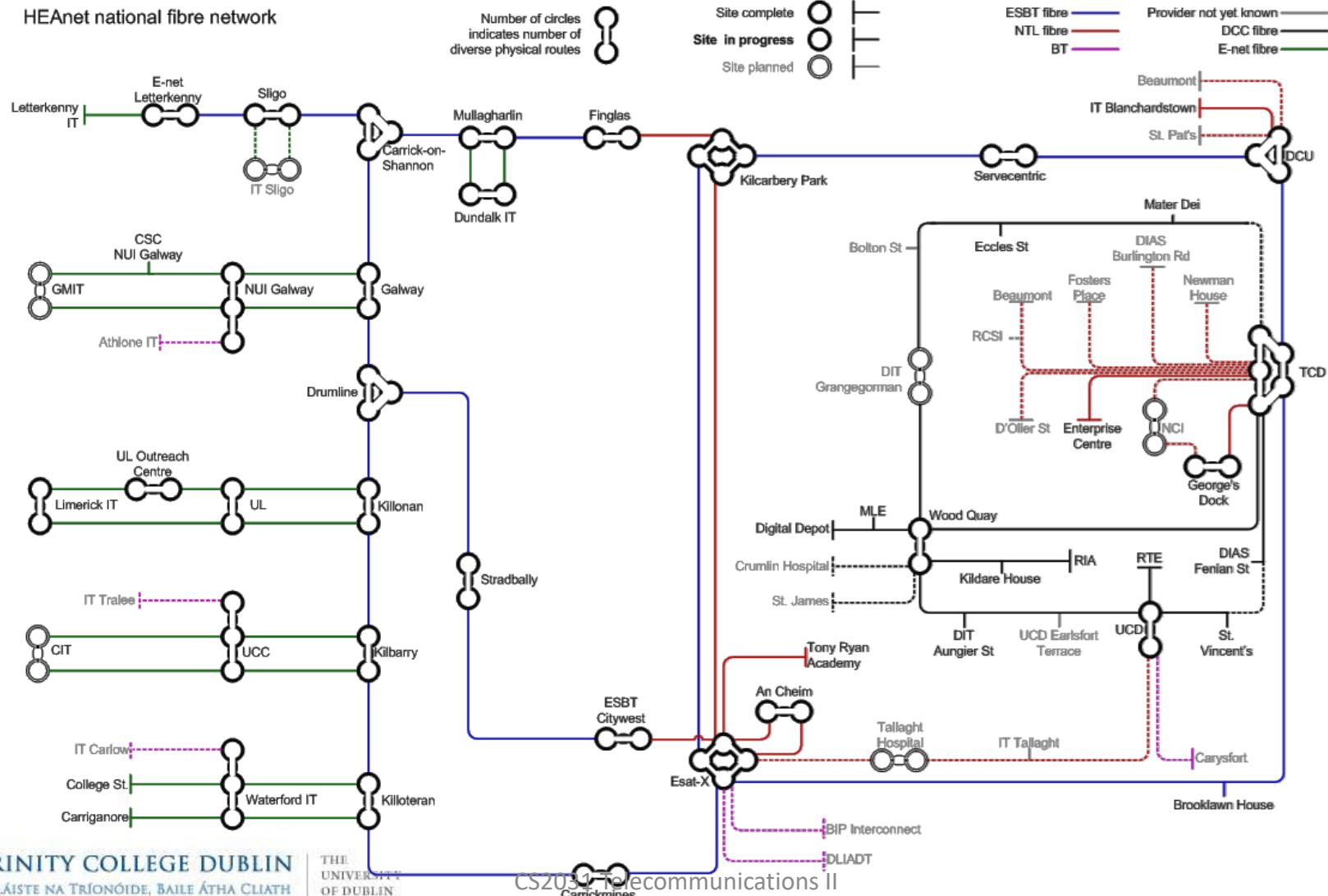


MPLS Use Case



- Creating a virtual network

HEAnet Fibre Network



Summary: Virtual Circuit Switching – ATM

- Virtual Circuit Switching
 - Preplanned route established before any frames sent
 - Call request and call accept frames establish connection (handshake)
 - Each frame contains a virtual circuit identifier instead of destination address
 - No routing decisions required for each frame
 - Clear request to drop circuit
 - Not a dedicated path
- Asynchronous Transfer Mode (ATM)
 - Example for virtual circuit switching
 - Cells consist of 5-byte header and 48-byte payload
 - Circuits identified by virtual circuit ID and virtual path ID
 - Application adaptation layer (AAL) for specific application areas