Computer Networks

Wenzhong Li
Nanjing University
Fall 2014

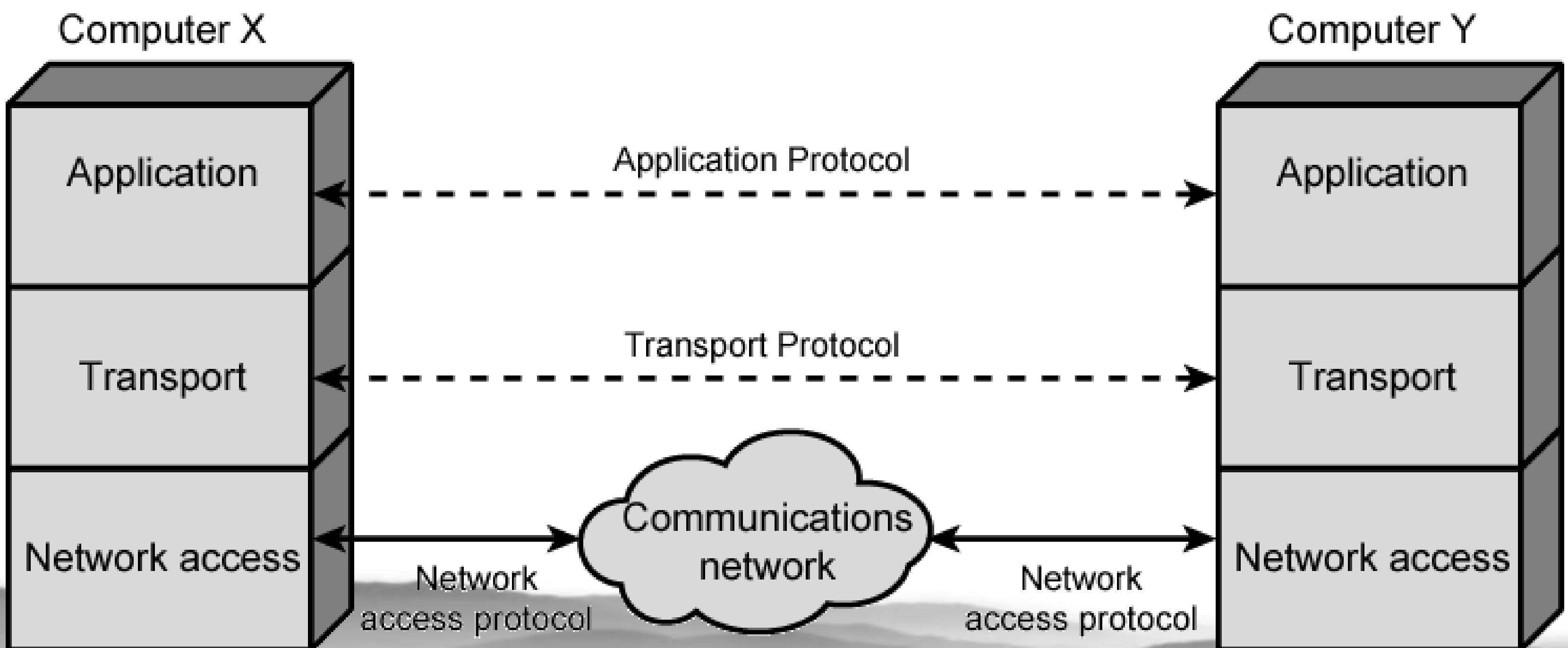
Chapter 1. Introduction of Networking (2)

- Brief Introduction of Internet
- Internet History
- Typical Network Applications
- Protocol Layers and Service Model
- Network Programming
- Network Performance
- Network Security

Protocol Layers and Service Model

Idea of Protocol Layers and Service Model

- A layered structure for File Transfer application
- Protocol – handle the communication issues between peer entities



Many Things to Handle

- Encapsulation
- Segmentation and reassembly

- Connection control
- Ordered delivery

- Flow control
- Error control

- Addressing
- Multiplexing

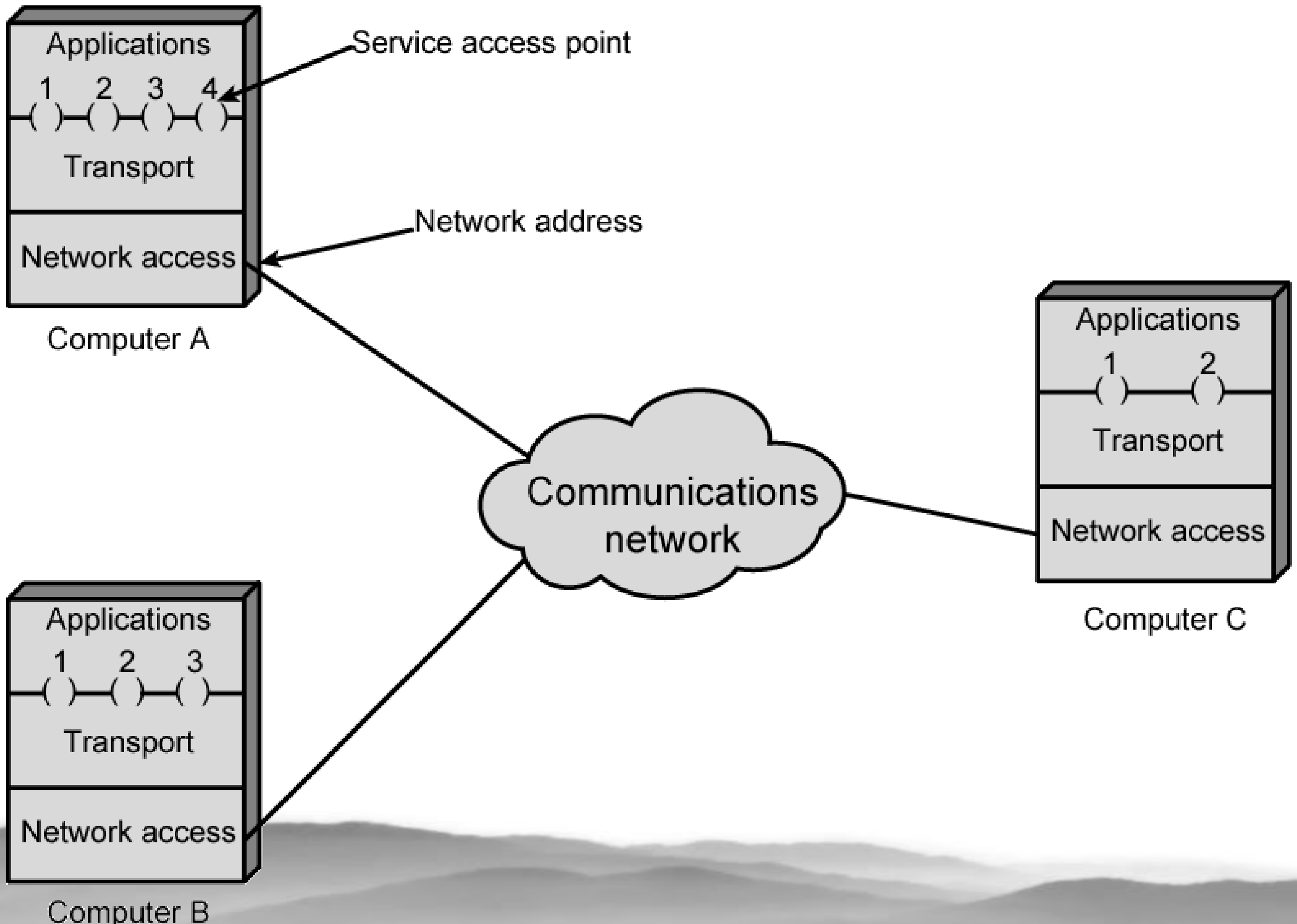
- Transmission services (QOS)

Addressing Requirements

- At least 2 levels of addressing required
  - Each computer needs unique network address
  - Each application on a (multi-tasking) host needs a specific address within the host
  - i.e. the service access point (SAP)

- e.g. the FTP port 21 on TCP/IP stacks on computer 202.106.182.120

Addressing Requirements



Standard Protocol Architectures

- Two standards:
  - OSI Reference model
    - Never lived up to early promises
  - TCP/IP protocol suite
    - Most widely used

- Others
  - IBM Systems Network Architecture (SNA)
  - DECNet, Netware

ISO-OSI

- Open Systems Interconnection (OSI)
- Developed by the International Organization for Standardization (ISO)
- Seven layers structure

- A theoretical system delivered too late
- TCP/IP is the de facto standard now

OSI – The Model

- A layer model, and flow structure
- Each layer performs a subset of the required communication functions
- Each layer relies on the next lower layer to perform more primitive functions
- Each layer provides services to the next higher layer
- Changes in one layer should not require changes in other layers

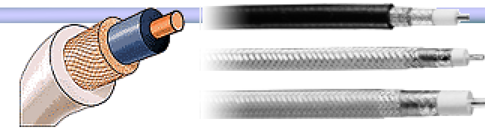
OSI Layers

Example：Alice invite Bob to lunch

Chinese example	OSI Layer	Description
“请客吃饭”	Application	Provides access to the OSI environment for users and al provides distributed information services.
语言表述	Presentation	Provides independence to the application processes from differences in data representation (syntax).
听说同步	Session	Provides the control structure for communication between applications; establishes, manages, and terminates connections (sessions) between cooperating applications.
摘机拨号	Transport	Provides reliable, transparent transfer of data between end points; provides end-to-end error recovery and flow control.
PBX中转	Network	Provides upper layers with independence from the data transmission and switching technologies used to connect systems; responsible for establishing, maintaining, and terminating connections.
信号传输	Data Link	Provides for the reliable transfer of information across the physical link; sends blocks (frames) with the necessary synchronization, error control, and flow control.
插口、双绞线	Physical	Concerned with transmission of unstructured bit stream over physical medium; deals with the mechanical, electrical, functional, and procedural characteristics to access the physical medium.

Physical Layer

- Transfers bits across link



- Specification of the physical aspects of a comm link
  - Mechanical: cable, plugs, pins...
  - Electrical/optical: modulation, signal strength, voltage levels, bit times, …
  - Functional/procedural: activate, maintain, deactivate physical links…
- Physical interface between devices
  - Ethernet, DSL, cable modem, telephone modems, …
  - Twisted-pair cable, coaxial cable, optical fiber, radio, infrared, …

Data Link Layer

- Groups bits into frames
- Activation, maintenance, & deactivation of data link connections

- Transfers frames across direct connections
- Medium access control for local area networks

- Detection of bit errors; Retransmission of frames
- End-to-end flow control

- Higher layers may assume error free transmission

Network Layer

- Transfers packets across multiple links / multiple networks
- Addressing must scale to large networks
- Nodes jointly execute routing algorithm to determine paths across the network
- Forwarding transfers packet across a node
- Congestion control to deal with traffic surges
- Connection setup, maintenance, and teardown when connection-based

Transport Layer

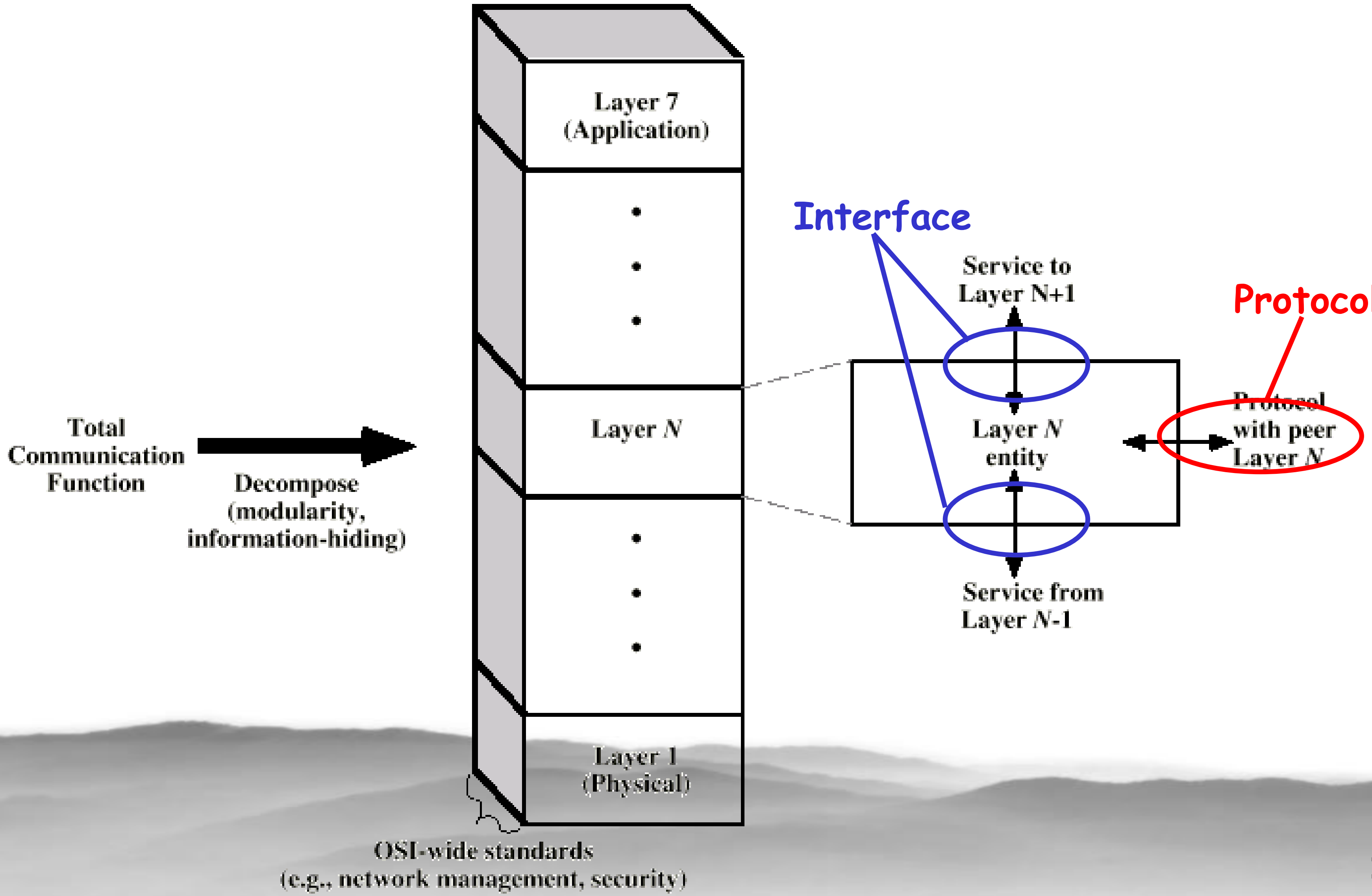
- Exchange of data between end systems
  - Transfers data end-to-end from process in one host to process in another host
- Reliable stream transfer or quick-and-simple single-block transfer
  - Error free
  - In sequence
  - No losses
  - No duplicates
- Connection setup, maintenance, and release

Upper Layers

- Session
  - Control of dialogues between applications
  - Dialogue discipline
  - Grouping data
  - Checkpoint recovery
- Presentation
  - Machine-independent representation of data
  - Data formats and coding
  - Data compression & encryption
- Application
  - Means for applications to access OSI environment

Incorporated into
Application Layer Now

OSI as Framework for Standardization

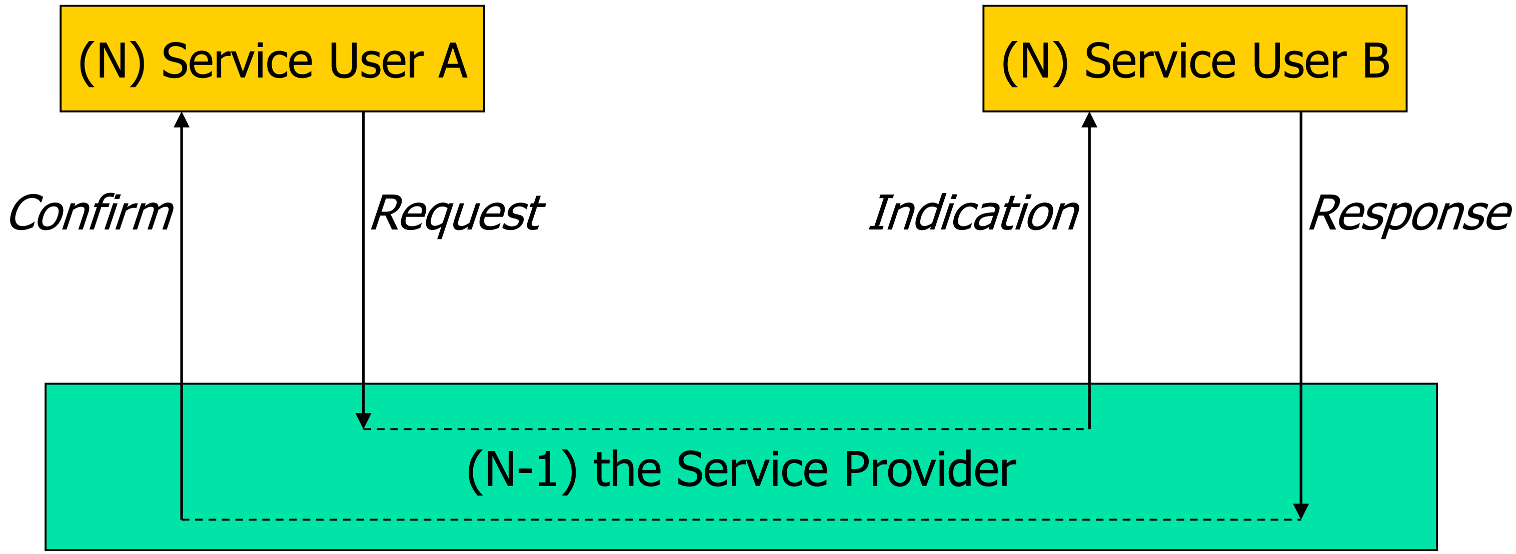


Service Primitives and Parameters

REQUEST	•Issued by a user (upper layer) to invoke some service •Parameters fully specify the requested service
INDICATION	•Issued by a service provider (lower layer) either to: •Indicate that a procedure has been invoked by peer service user, or •Notify the service user of a provider-initiated action
RESPONSE	•Issued by peer user to acknowledge or complete previously invoked procedure
CONFIRM	•Issued by service provider to acknowledge or complete previously invoked procedure

Service Primitives

connect.request → connect.indication →
connect.response → connect.confirm

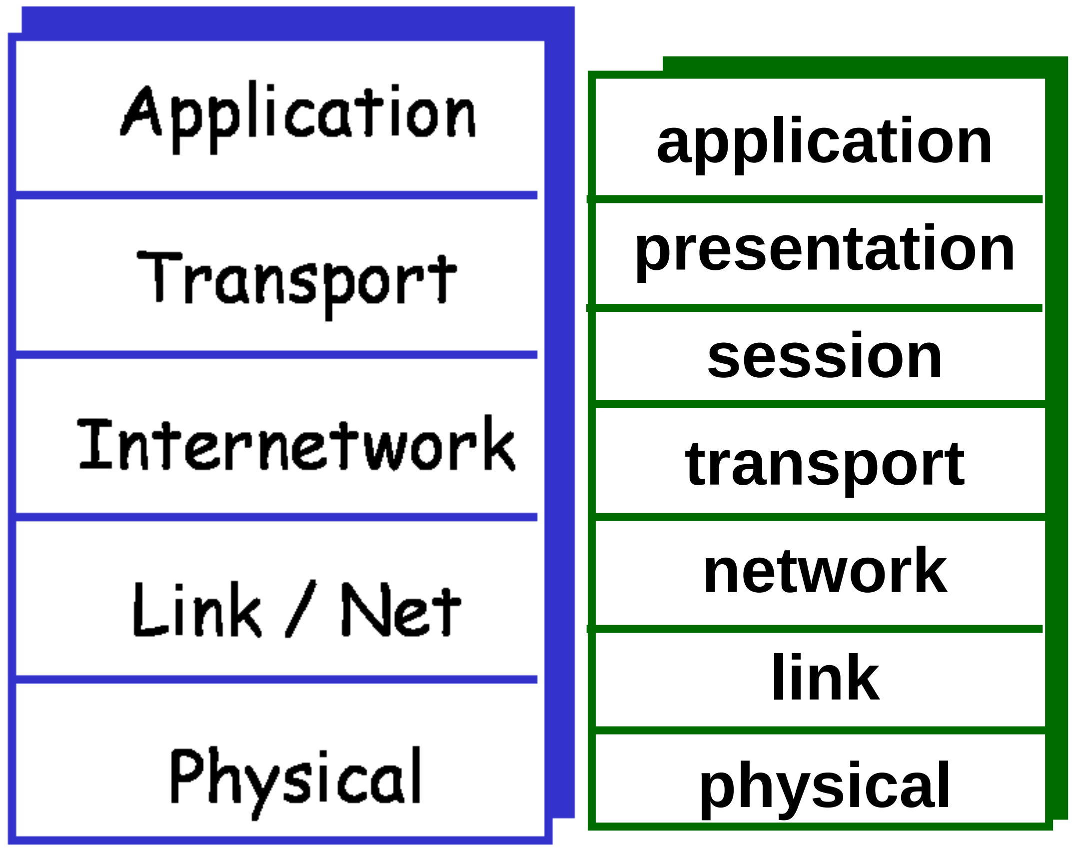


TCP/IP Protocol Architecture

Used by the global Internet

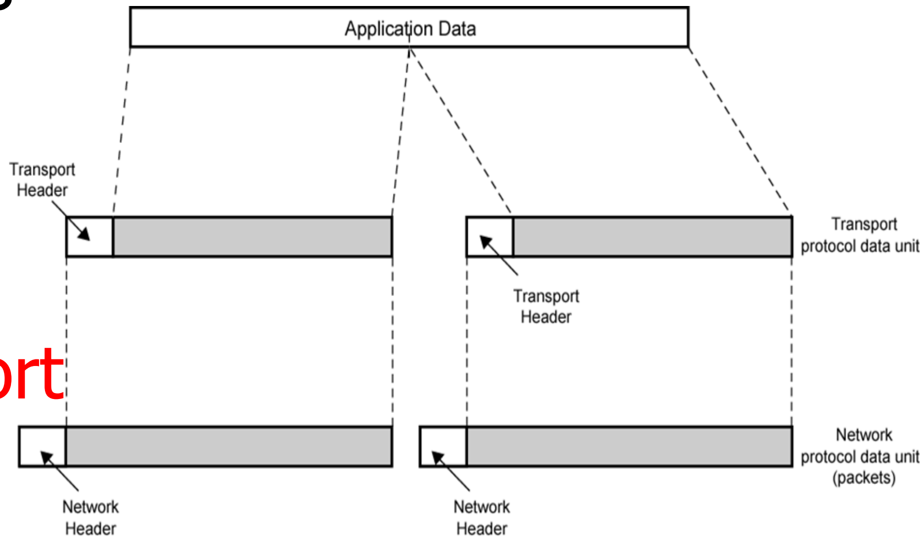
- Application: supporting network applications
  - FTP, SMTP, HTTP
- Transport: process-process data transfer
  - TCP, UDP
- Internetwork: routing of datagrams across net of nets
  - IP, routing protocols
- Link: data transfer between neighboring routers / hosts
  - PPP, Ethernet
- Physical: bits "on the wire"

TCP/IP protocol stack vs. OSI

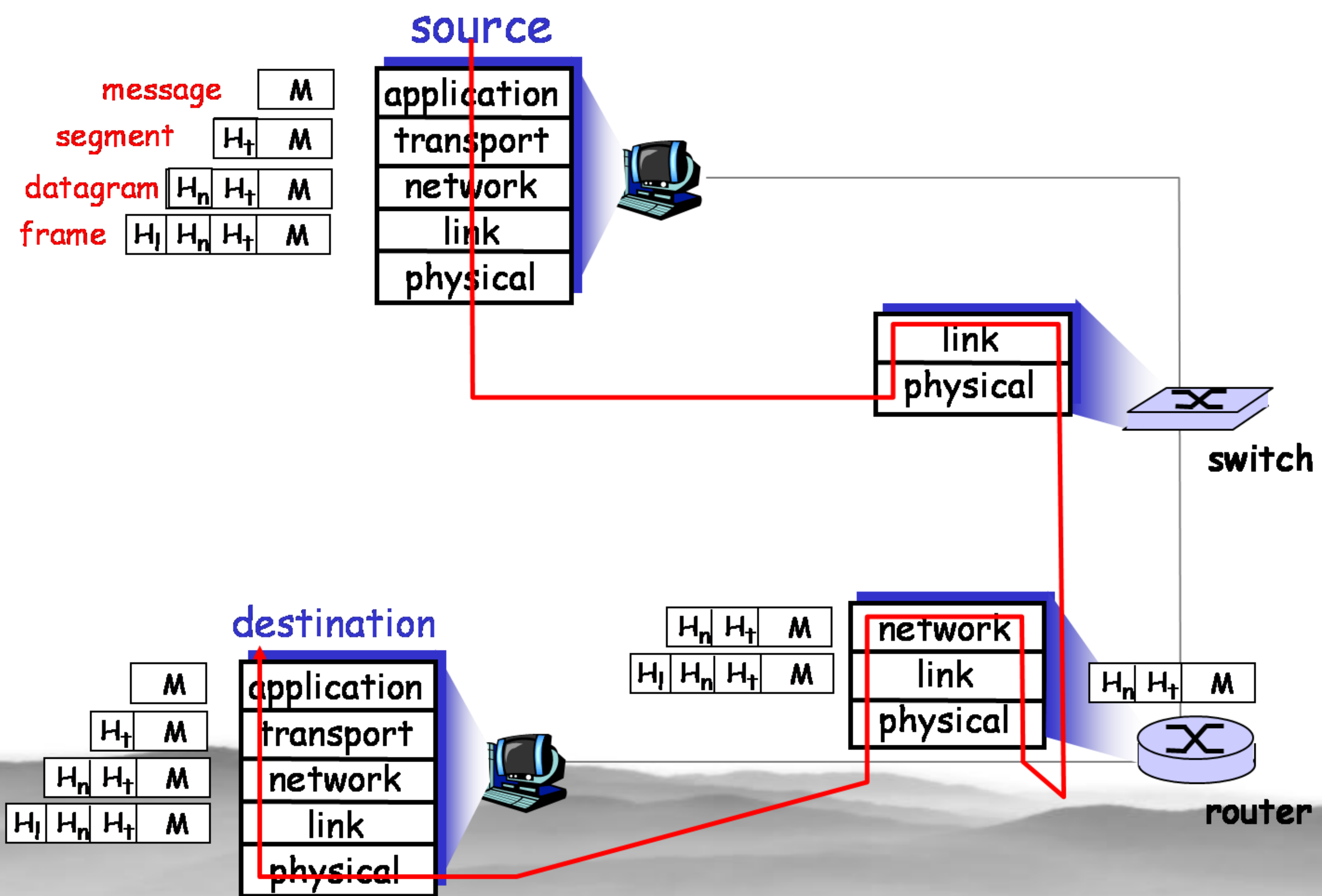


Protocol Data Units

- At each layer, Control info is added to user data to ease communication, e.g.
- Transport layer segments application data
- Each segment has a transport header added
  - Destination port
  - Sequence number
  - Error detection code
- This gives a transport protocol data unit (PDU)

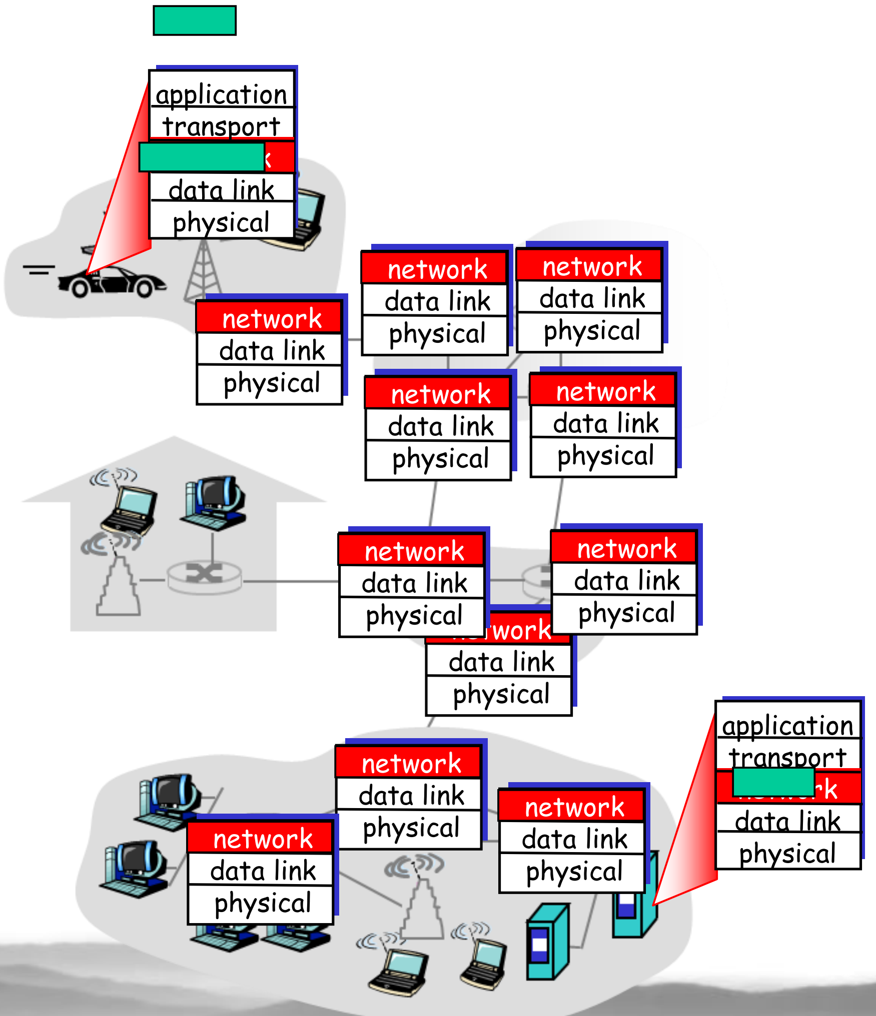


Encapsulation

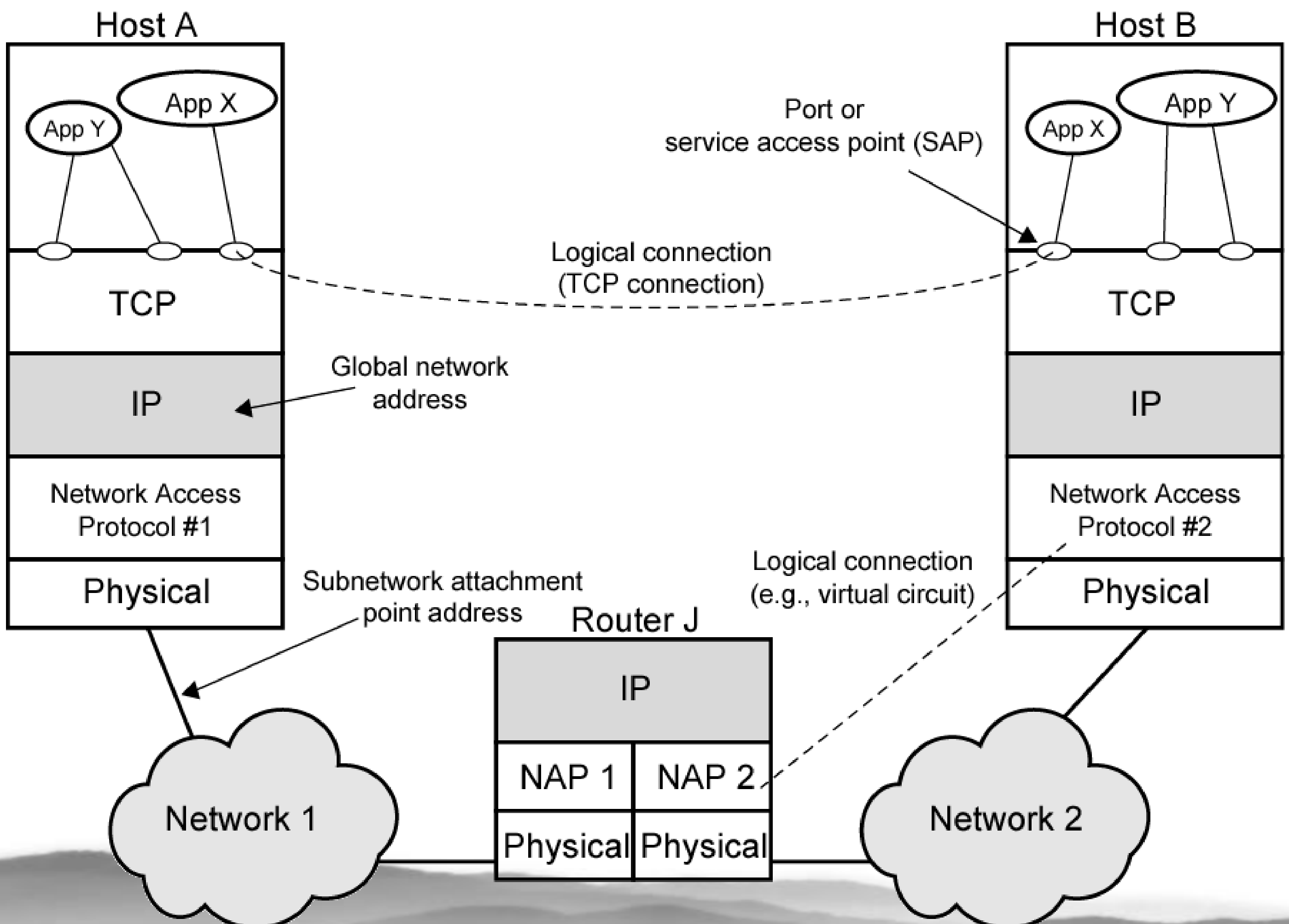


The IP Layer in Detail

- Sender encapsulates segments into datagrams
- Receiver delivers segments to transport layer
- IP layer entity resides on each host and router
- Router examines header fields in all IP datagrams passing through it



Look At This


Host A
App X
App Y
TCP
IP
Network Access
Protocol #1
Physical
Port or
service access point (SAP)
Logical connection
(TCP connection)
Global network
address
Host B
App Y
App X
TCP
IP
Network Access
Protocol #2
Physical
Subnetwork attachment
point address
Router J
Logical connection
(e.g., virtual circuit)
IP
NAP 1
NAP 2
Physical
Physical
Network 1
Network 2

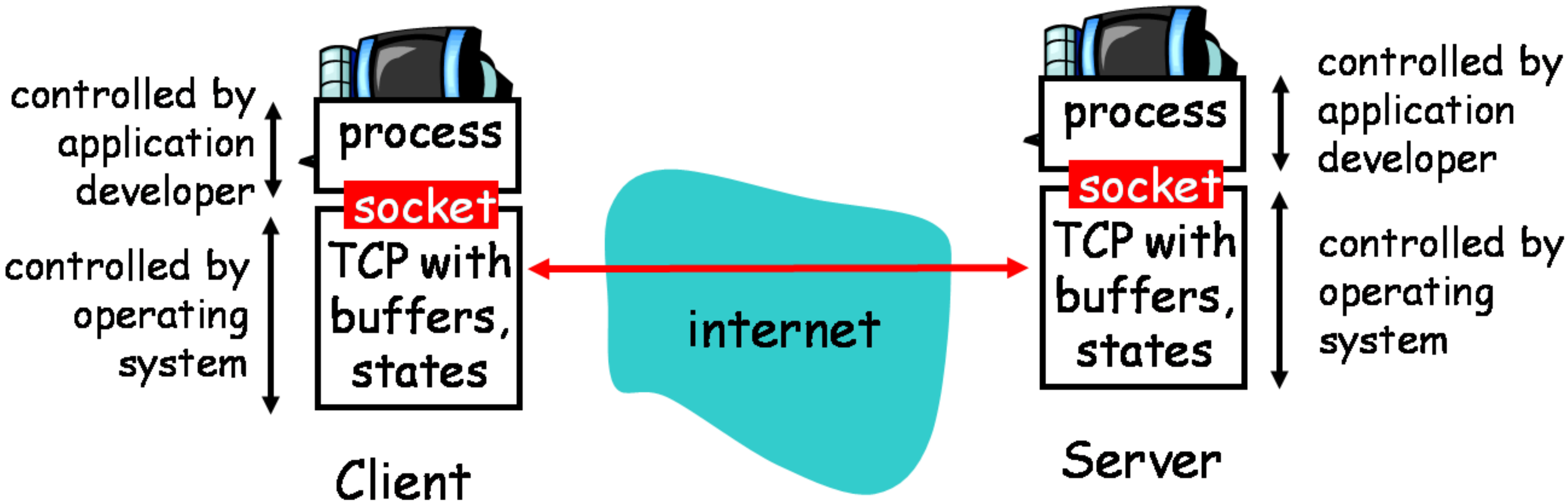
Network Programming

Network Programming

- Socket programming
  - Build client/server application that communicate using sockets
  - A socket is a pair of <IP addresses, port>
- Socket API
  - Introduced in BSD4.1 UNIX, 1981
  - Explicitly created, used, and released by applications
  - Implementing client/server paradigm
- 2 types of transport service via socket API
  - Unreliable datagram, i.e. UDP
  - Reliable, byte stream-oriented, i.e. TCP

Socket Programming via TCP

- TCP Services
  - Reliable transfer of bytes (octets) from one process to another



Socket Programming with TCP

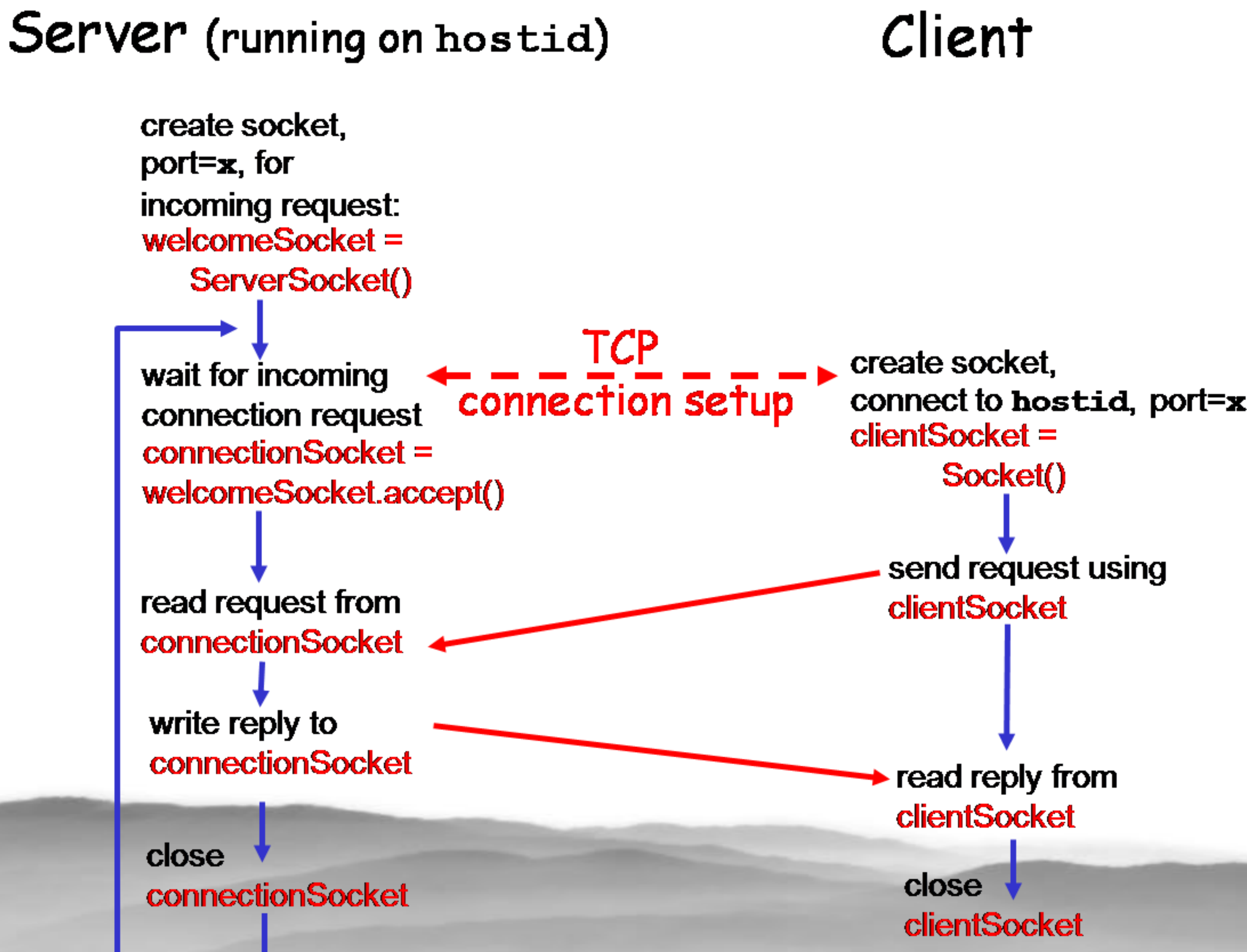
Client must contact server

- Create a client-local TCP socket
- Specify $\langle IP_s, Port_s\rangle$ of server process
- Receive server reply, a connection is created on $\langle IP_c, Port_c; IP_s, Port'_s\rangle$

Server is running first

- Server have created a socket that is listening
- Accept the client contact, create a new socket to communicate with client
- Connection created on $\langle IP_c, Port_c; IP_s, Port'_s\rangle$
- Using $\langle IP_c, Port_c\rangle$ to distinguish different clients

Client/Server Socket Interaction



Example: Java Client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception
    {
        String sentence;
        String modifiedSentence;

        BufferedReader inFromUser =
          new BufferedReader(new InputStreamReader(System.in));

        Socket clientSocket = new Socket("hostname", 6789);

        DataOutputStream outToServer =
          new DataOutputStream(clientSocket.getOutputStream());
```

Create input stream

Create client socket, connect to server

Create output stream attached to socket

Example: Java Client (TCP)

Create input stream attached to socket

```
BufferedReader inFromServer =
  new BufferedReader(new
  InputStreamReader(clientSocket.getInputStream()));

 sentence = inFromUser.readLine();
```

Send line to server

```
 outToServer.writeBytes(sentence + '\n');
```

Read line from server

```
 modifiedSentence = inFromServer.readLine();

 System.out.println("FROM SERVER: " + modifiedSentence);

 clientSocket.close();

}
}
```

Example: Java Server (TCP)

```
import java.io.*;
import java.net.*;

class TCPServer {

  public static void main(String argv[]) throws Exception
   {
    String clientSentence;
    String capitalizedSentence;

    ServerSocket welcomeSocket = new ServerSocket(6789);

    while(true) {

        Socket connectionSocket = welcomeSocket.accept();

       BufferedReader inFromClient =
         new BufferedReader(new
         InputStreamReader(connectionSocket.getInputStream()));
```

Create welcoming socket at port 6789

Wait, on welcoming socket for contact by client

Create input stream, attached to socket

Example: Java Server (TCP)

Create output stream, attached to socket

```
DataOutputStream  outToClient =
  new DataOutputStream(connectionSocket.getOutputStream());
```

Read in line from socket

```
clientSentence = inFromClient.readLine();

capitalizedSentence = clientSentence.toUpperCase() + '\n';
```

Write out line to socket

```
outToClient.writeBytes(capitalizedSentence);
        }
      }
    }
```

End of while loop, loop back and wait for another client connection

Socket Programming with UDP

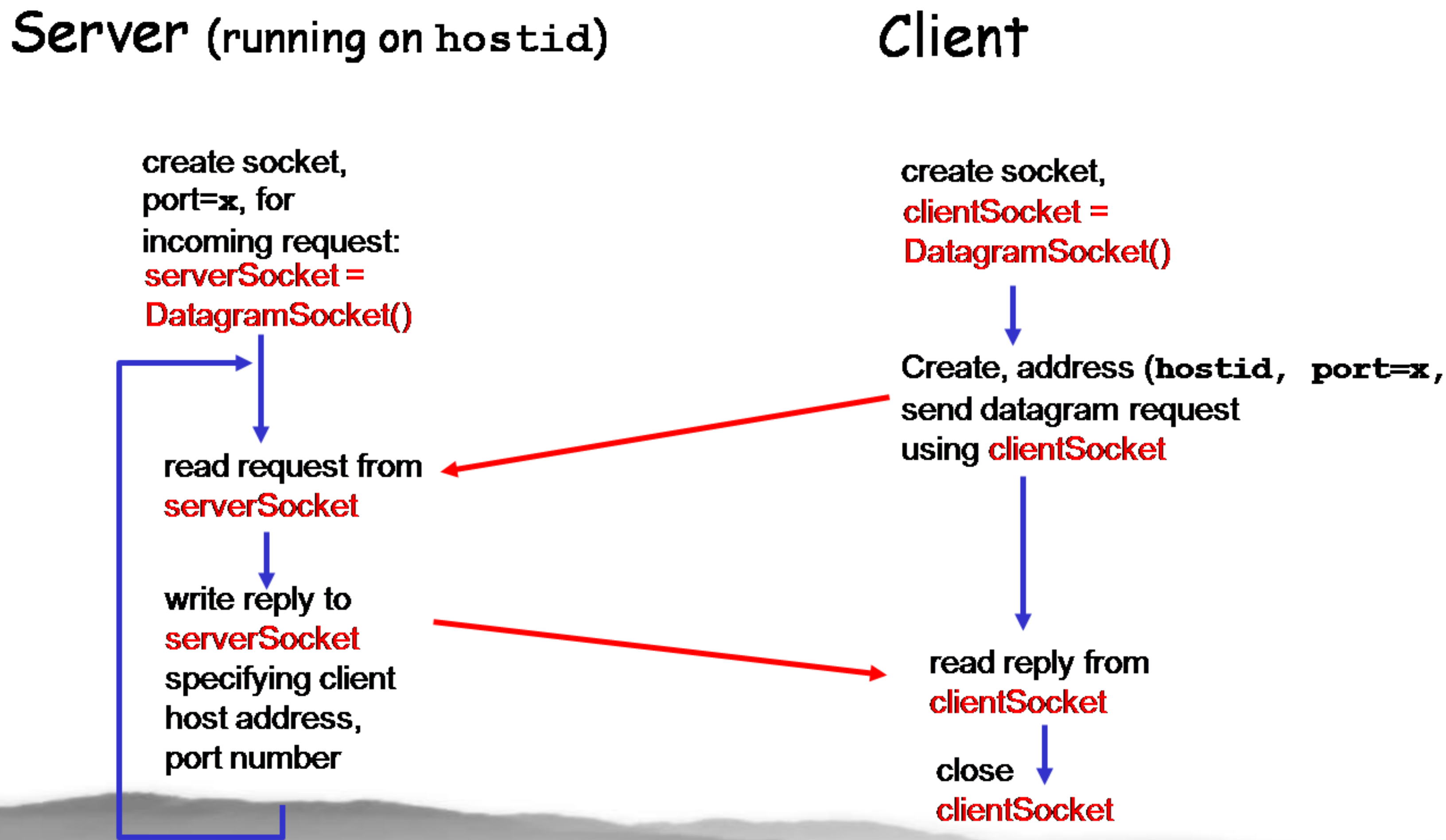
UDP: no "connection" between client and server

- A socket pair $<IP_c, Port_c; IP_s, Port_s>$ is also used, by each datagram

- Sender explicitly attaches IP address and port of destination to each packet
- Receiver must extract IP address, port of sender from received packet

- Transmitted data may be received out of order, or lost

Client/Server Socket Interaction



Example: Java Client (UDP)

Create input stream

Create client socket

Translate hostname to IP address using DNS

```
import java.io.*;
import java.net.*;

class UDPClient {
   public static void main(String args[]) throws Exception
   {

     BufferedReader inFromUser =
       new BufferedReader(new InputStreamReader(System.in));

     DatagramSocket clientSocket = new DatagramSocket();

     InetAddress IPAddress = InetAddress.getByName("hostname");

     byte[] sendData = new byte[1024];
     byte[] receiveData = new byte[1024];

     String sentence = inFromUser.readLine();

     sendData = sentence.getBytes();
```

Example: Java Client (UDP)

Create datagram with data-to-send, length, IP addr, port

```
DatagramPacket sendPacket =
  new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
```

Send datagram to server

```
clientSocket.send(sendPacket);

DatagramPacket receivePacket =
  new DatagramPacket(receiveData, receiveData.length);
```

Read datagram from server

```
clientSocket.receive(receivePacket);

String modifiedSentence =
   new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);
clientSocket.close();
}

}
```

Example: Java Server (UDP)

Create datagram socket at port 9876

Create space for received datagram

Receive datagram

```
import java.io.*;
import java.net.*;

class UDPServer {
  public static void main(String args[]) throws Exception
   {

     DatagramSocket serverSocket = new DatagramSocket(9876);

     byte[] receiveData = new byte[1024];
     byte[] sendData  = new byte[1024];

     while(true)
      {

        DatagramPacket receivePacket =
          new DatagramPacket(receiveData, receiveData.length);
        serverSocket.receive(receivePacket);
```

Example: Java Server (UDP)

```
        String sentence = new String(receivePacket.getData());

        InetAddress IPAddress = receivePacket.getAddress();

        int port = receivePacket.getPort();

                String capitalizedSentence = sentence.toUpperCase();

        sendData = capitalizedSentence.getBytes();

        DatagramPacket sendPacket =
           new DatagramPacket(sendData, sendData.length, IPAddress,
                              port);

        serverSocket.send(sendPacket);
      }
    }
}
```

Get IP addr
port #, of
sender

Create datagram
to send to client

Write out
datagram
to socket

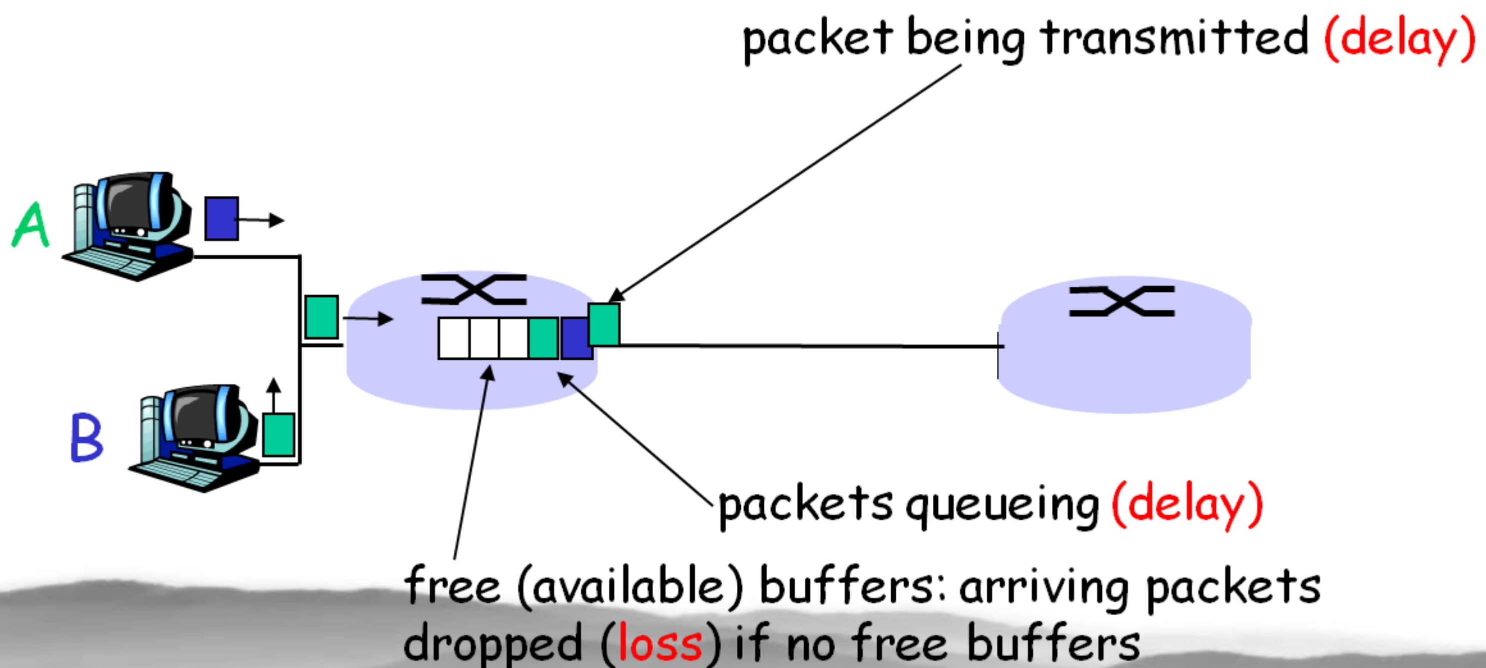
End of while loop,
loop back and wait for
another datagram

Network Performance

Delay

Packets queue in switch buffers

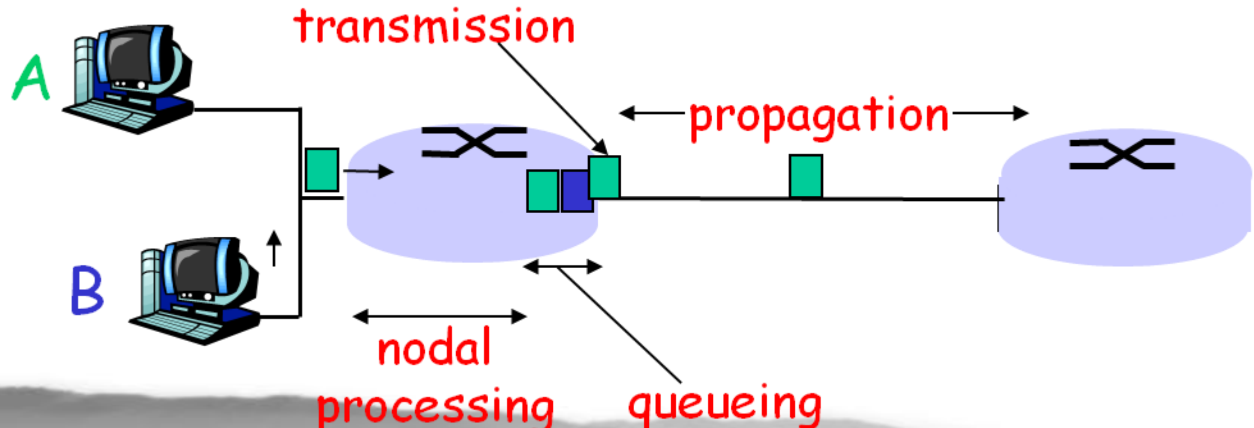
- Packet arrival rate exceeds output link capacity
- Packet queues, wait for its turn



Four Sources of Packet Delay

1. Transmission
- R=link bandwidth (bps)
- L=packet length (bits)
- Time to send bits into link = L/R

2. Propagation
- d = length of physical link
- s = propagation speed in medium (~2×10^8 m/sec)
- Propagation delay = d/s



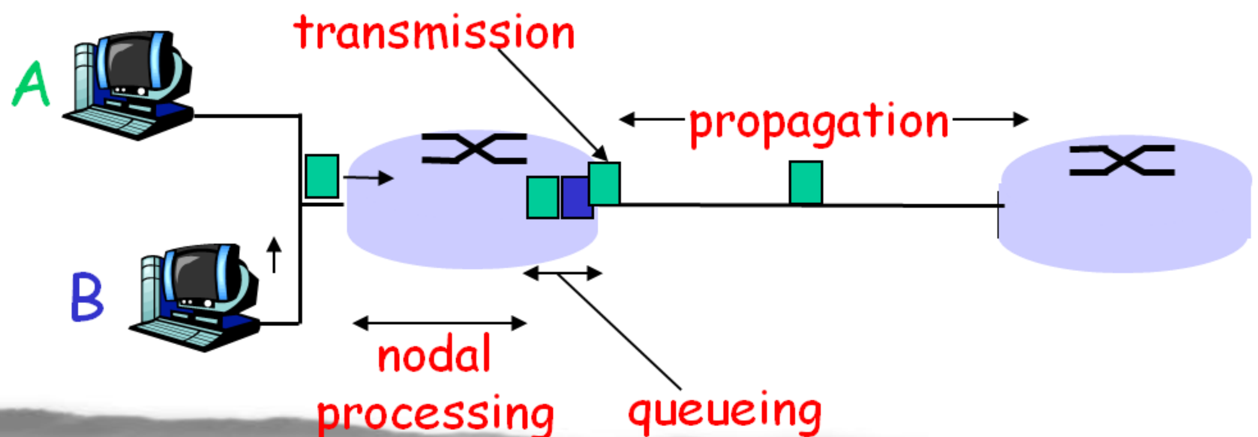
Four Sources of Packet Delay

3. Nodal processing

- Check bit errors
- Determine output link

4. Queuing

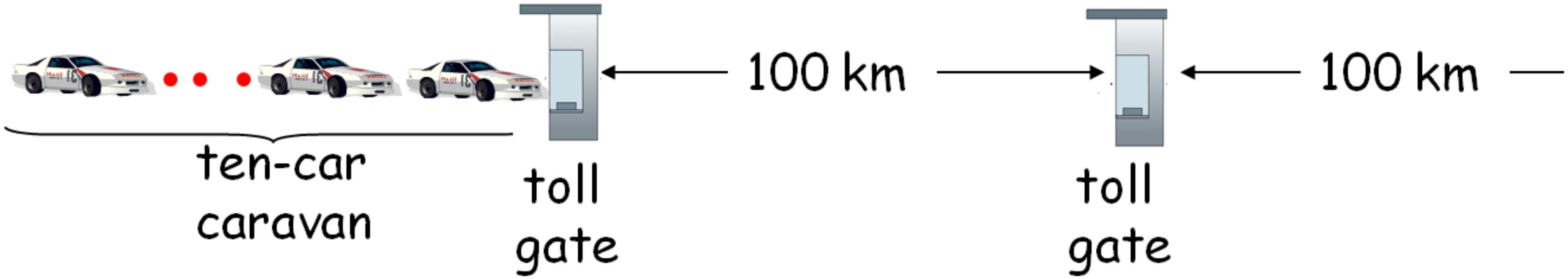
- Time waiting at output link for transmission
- Depending on congestion level of router



Magnitude of Different Delay

- Transmission delay
  - Significant for low-speed links, now typically a few microseconds or less
- Propagation delay
  - A few micro-seconds to hundreds of milliseconds
- Nodal processing delay
  - Typically a few microseconds or less
- Queuing delay
  - Depends on congestion, maybe seconds

Caravan Analogy



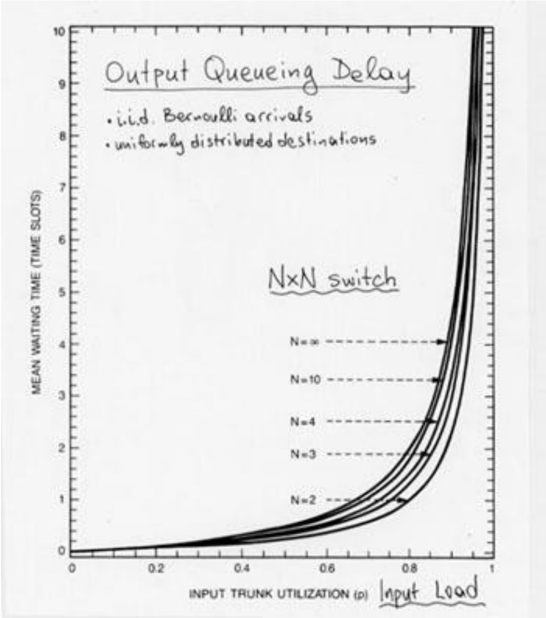
- Cars "propagate" at 100 km/hr
- Toll gate takes 12 sec to service car (nodal+trans)
- Car: packet; Caravan: packet flow
- Q: How long until caravan is lined up before 2nd toll gate?
- Time to "push" entire caravan through toll gate = 12$\times$10 = 120 sec
- Time for last car to propagate from 1st to 2nd toll gate: 100km/(100km/hr)= 1 hr
- Answer: 62 minutes
- Q: what about a single car?

Queuing Delay

- R=link bandwidth (bps)
- L=packet length (bits)
- α=average packet arrival rate

流量强度

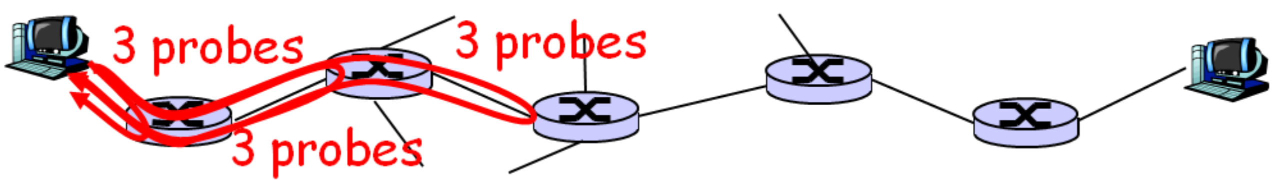
Traffic intensity $\rho = L\times\alpha/R$



- Intensity ρ ~ 0 : average queuing delay small
- Intensity $\rho \rightarrow 1$: delays become large, and huge
- Intensity $\rho \geq 1$: average delay infinite

"Real" Internet Delays and Routes

- traceroute
  - www.traceroute.org
  - Provides delay measurement from source to router along end-to-end Internet path towards destination
  - Each intermediate router will return packets to sender
  - Sender records time interval between transmission and reply



“Real” Internet Delays and Routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

Three delay measurements from gaia.cs.umass.edu to cs-gw.cs.umass.edu

```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

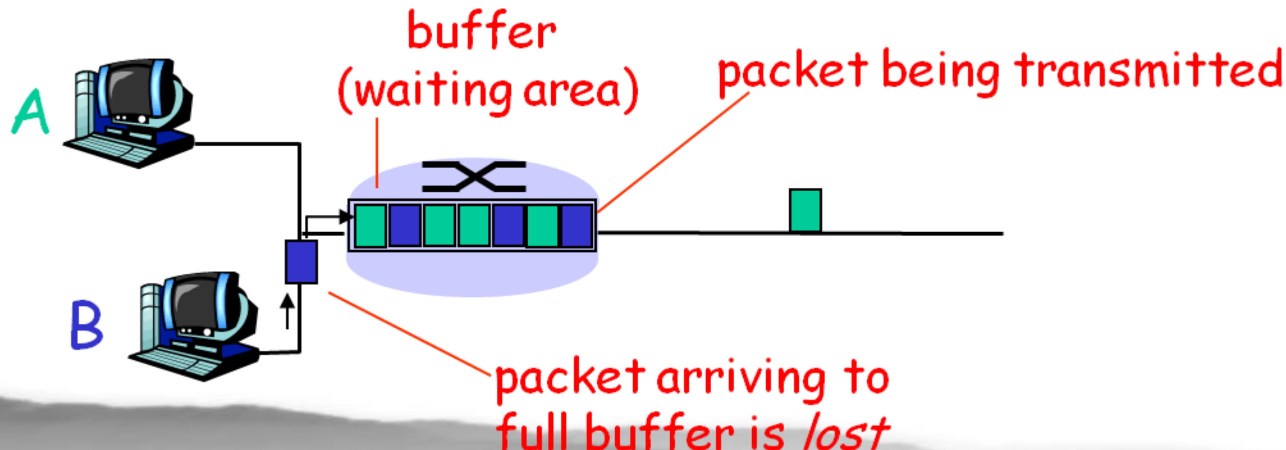
trans-oceanic link

* means no response (probe lost, router not replying)

Packet Loss

- Link in buffer of a router has finite capacity
- Packet arriving to full queue dropped (i.e. lost)

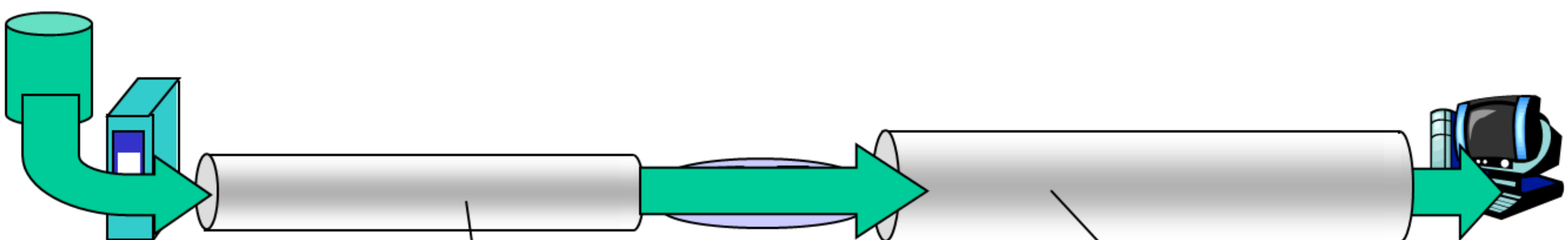
- Lost packet may be retransmitted by previous node, by source end system, or not at all



Throughput

- Throughput
  - Rate (bits/unit per time) at which bits transferred between sender/receiver

  - Instantaneous: rate at given point in time
  - Average: rate over a period of time

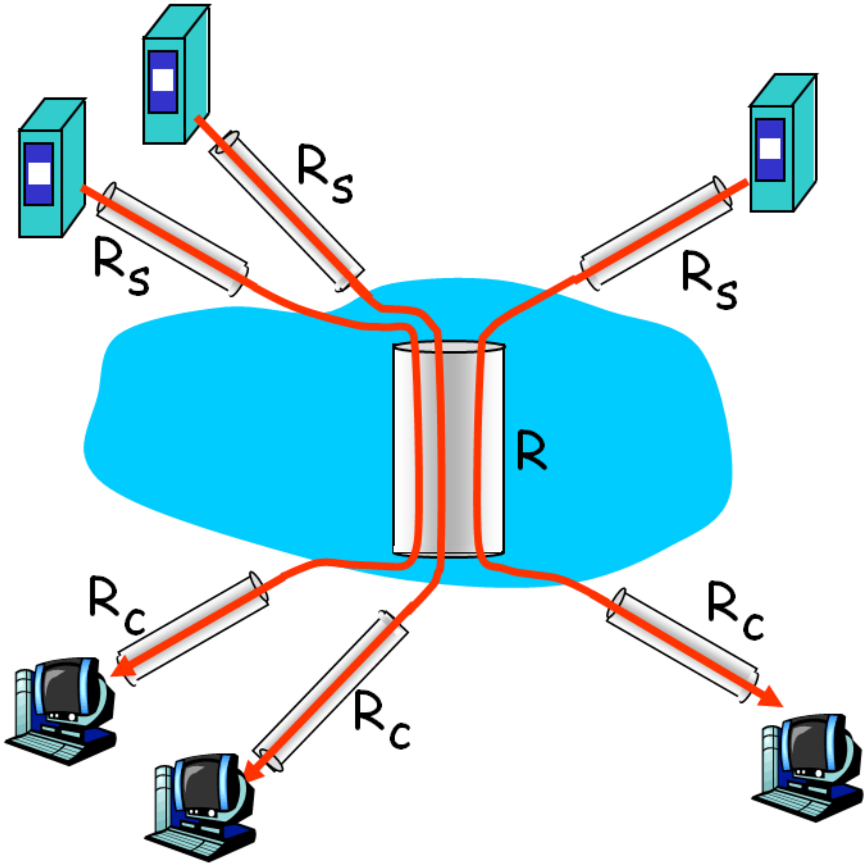


Throughput – Multiplexing

- Per-connection end-to-end throughput:

  $$\min(R_c, R_s, R/10)$$

- In practice: R_s (or R) is often the bottleneck



10 connections (fairly) share backbone bottleneck link R bits/sec

Network Security

Networks under Attack: Security

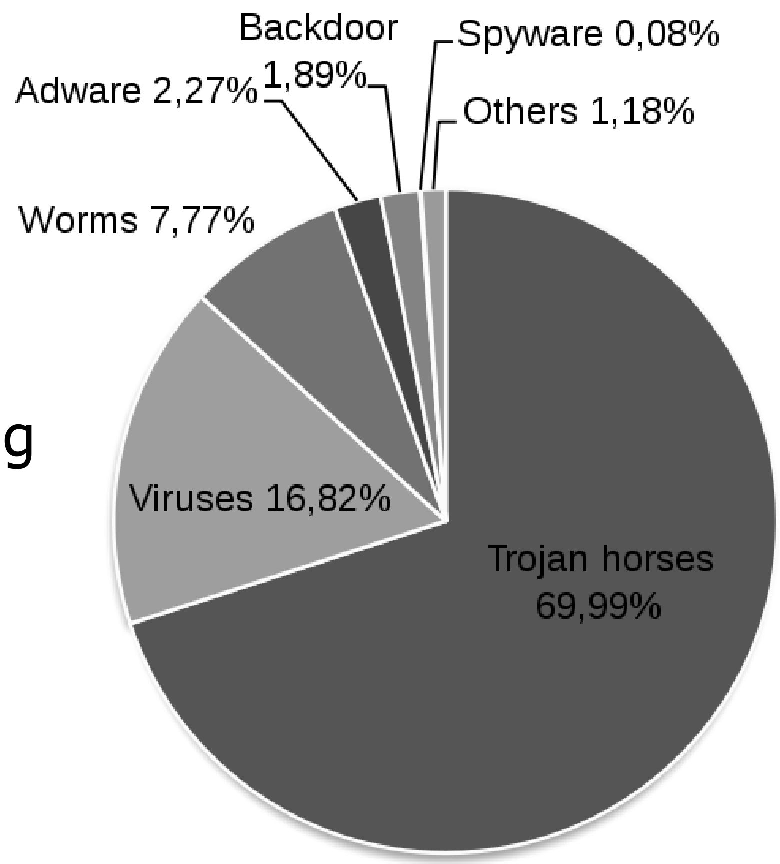
- Attacks on Internet infrastructure
  - Infecting/attacking hosts: malware, spyware, worms, unauthorized access
  - Packet sniffing, replay, masquerade
  - Denial of service: deny access to resources (servers, link bandwidth)
- Internet not originally designed with security in mind
  - Original vision: "a group of mutually trusting users attached to a transparent network"
  - Internet protocol designers playing "catch-up"
  - Security considerations in all layers!

Different Types of Malware

- Virus
  - Infection by receiving and running (unwarily) executables
  - Self-replicating: propagate itself to other executables
- Worm
  - Actively transmitting itself over a network to infect other hosts
- Trojan horses
  - Disguised as something innocuous or desirable, tempting the user to run it

Different Types of Malware

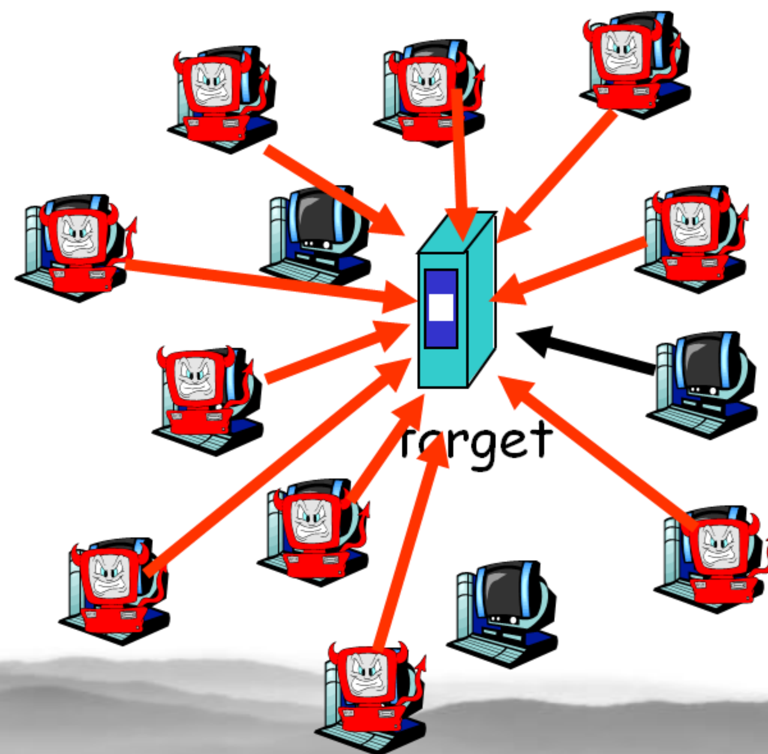
- Backdoor
  - Providing a method of bypassing normal authentication procedures
- Adware
  - Playing, displaying, or downloading advertisements to the user host
- Spyware
  - Infecting in the same way as Trojan horses
  - Recording keystrokes, web sites visited, uploading info to collection site



Denial of Service (DOS)

- Attackers make resources (server, bandwidth) unavailable by overwhelming resource with bogus traffic
- e.g. multiple coordinated sources swamp server with TCP SYN message

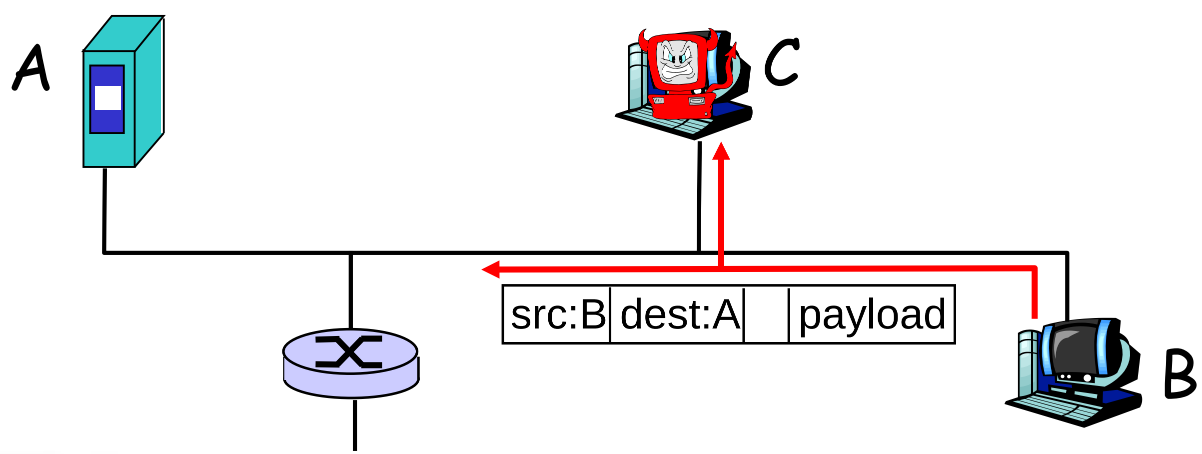
1. Select target
2. Break into hosts around the network using malware
3. Send packets toward target from compromised hosts



Packet Sniffing

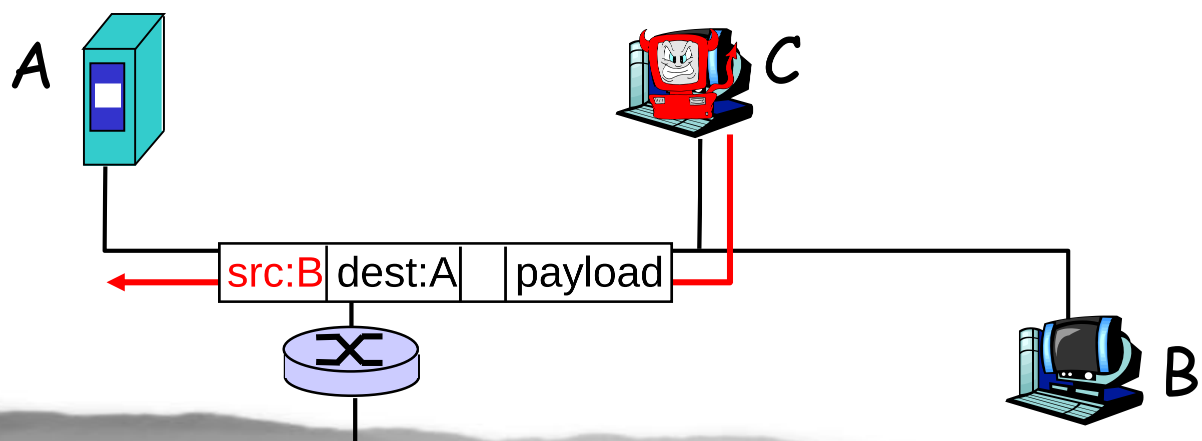
- Broadcast media (e.g. Ethernet or WiFi)
- Promiscuous NIC reads/records all packets passing by
  - Can read all unencrypted data (e.g. passwords)

- e.g. C sniffs B's packets



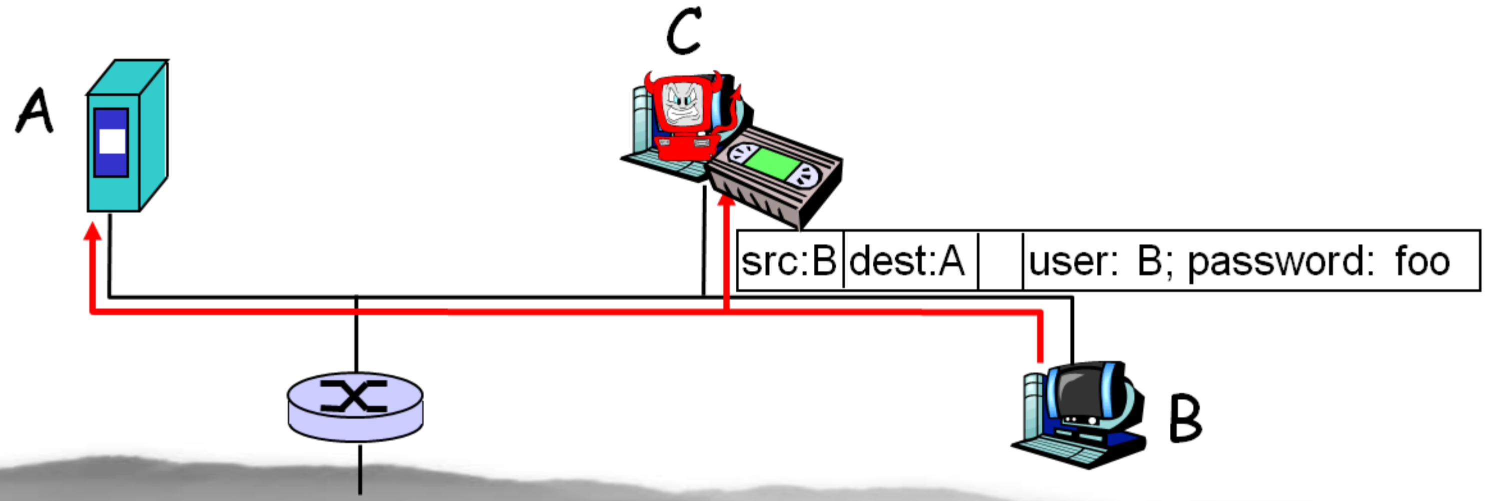
IP Spoofing

- Generate raw IP packets directly, putting any value into IP source address field
- Receiver can't tell if source is spoofed
- e.g. *C* pretends to be *B*



Masquerade

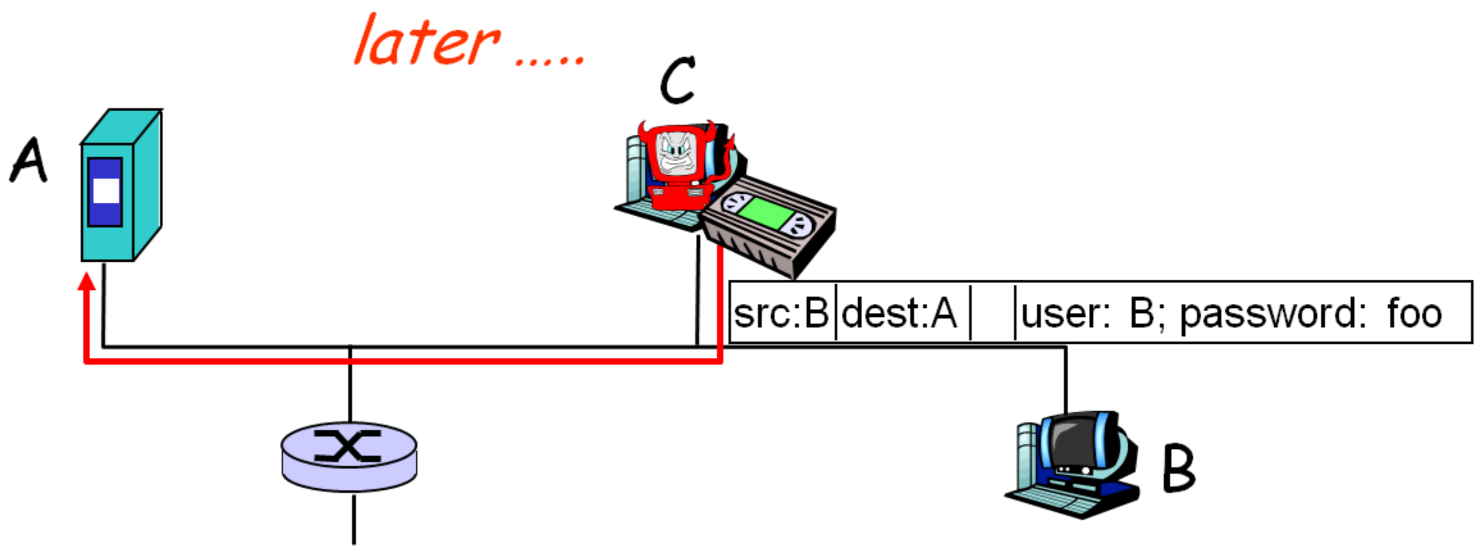
- IP spoofing: send packet with false source address
- Record-and-playback: sniff sensitive info (e.g., password), and use later



Masquerade

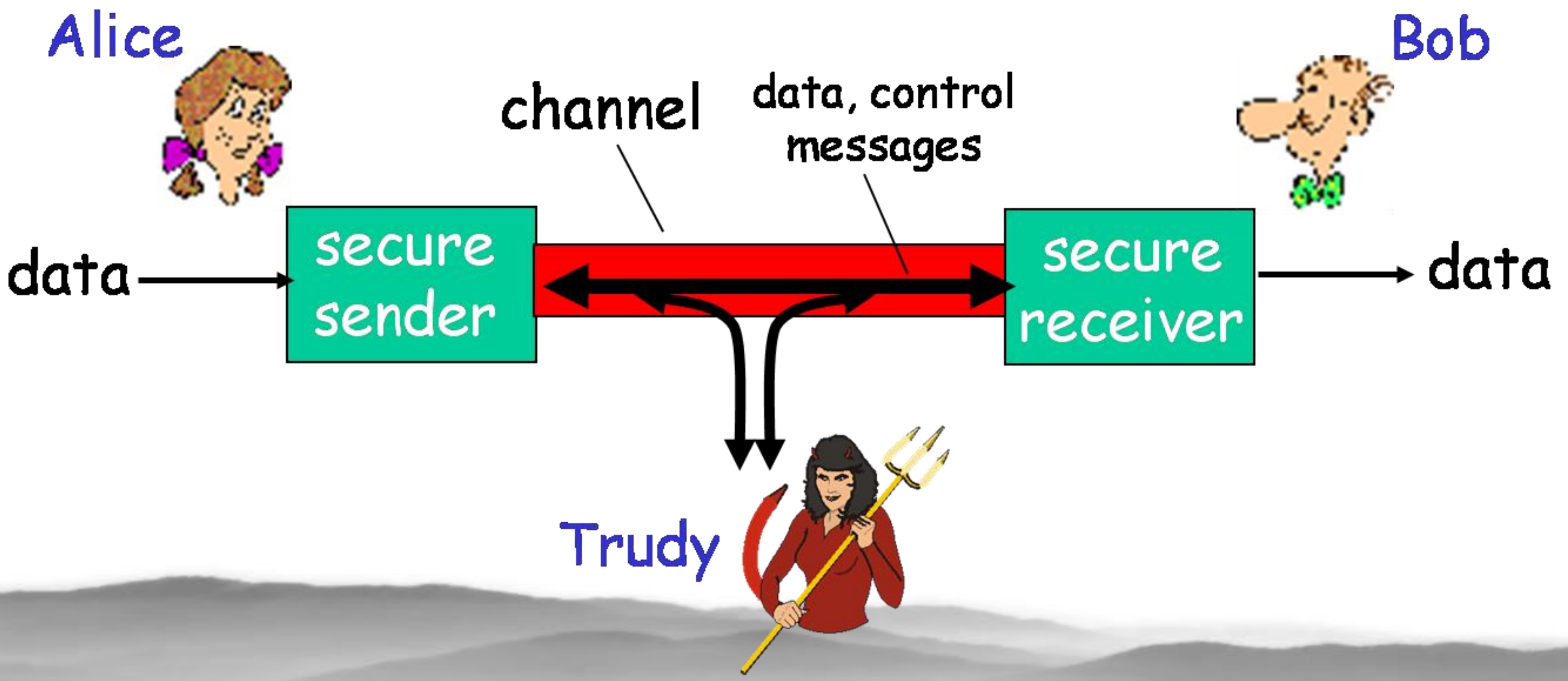
Later

- The server cannot tell who is actually B



Common Scenario of Network Security

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



Scenario vs. Reality

Real-life Bobs and Alices

- Web browser/server for electronic transactions (e.g., on-line purchases)
- On-line banking client/server

- DNS servers exchanging DNS queries/answers
- Routers exchanging routing table updates

- Many others …

What Trudy Might Do

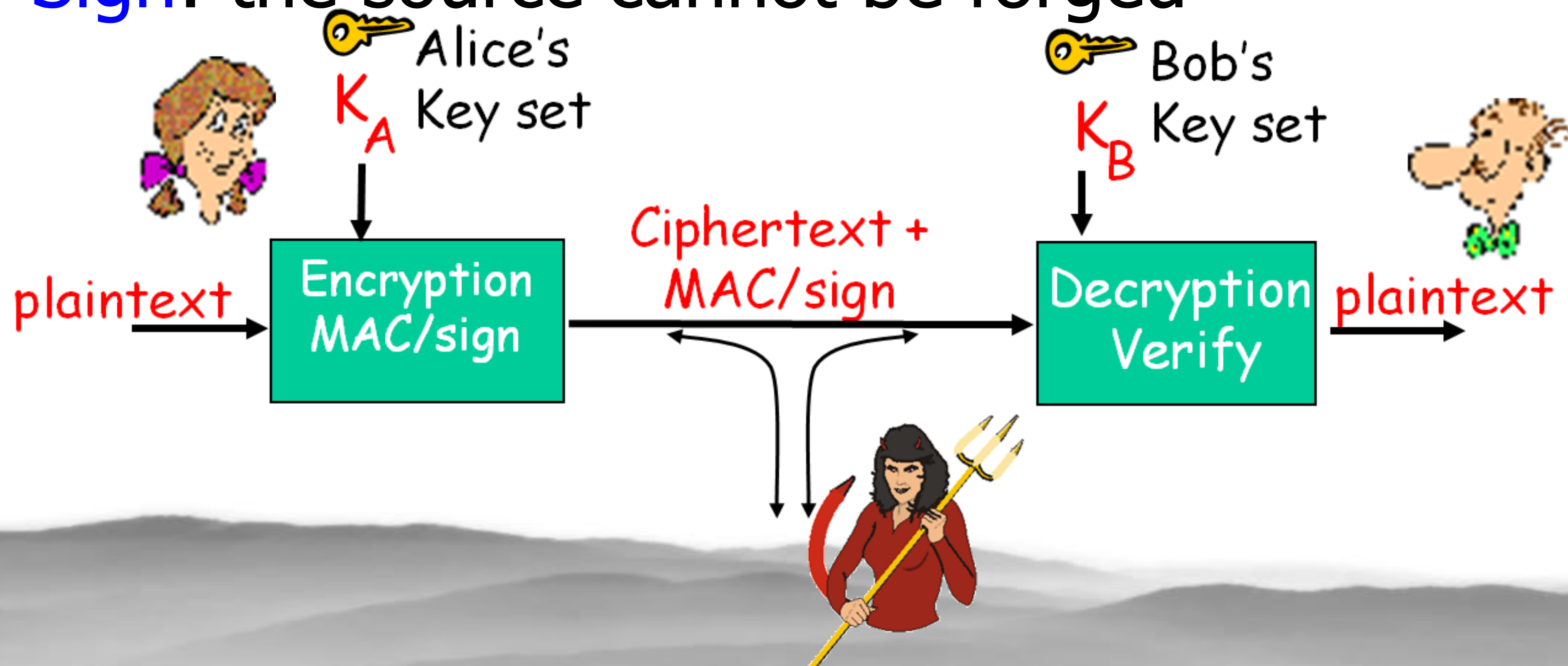
- Eavesdrop（窃听）: intercept messages
- Actively insert messages into connection

- Impersonation （冒充）: can fake (spoof) source address in packet (or any field in packet)
- Hijacking（劫持）: "take over" ongoing connection by removing sender or receiver, inserting himself in place

- Denial of service: prevent service from being used by others (e.g., by overloading resources)

How to Handle This

- Encryption: the message cannot be understood
- Message Authentication Code (MAC): the message cannot be altered
- Sign: the source cannot be forged



Summary

- 协议层次及模型
  - OSI七层模型
  - TCP/IP协议栈五层模型
- 网络编程：TCP，UDP socket
- 网络时延、丢包、吞吐量概念
  - 四种时延：处理、排队、传输、传播
- 网络安全基本概念

Homework

- 书第2章习题：2.2, 2.4, 2.6