C:\cygwin\home\zahorjan\cse461\12au\01_Intro.cp3

CSE 461: Introduction to Computer Communications Networks

Autumn 2012

Module 1

Course Introduction

Ryan Emmett
Melissa Winstanley

John Zahorjan

Today's Agenda

1. Overview of course mechanics

2. The Course
   - What is it about?
   - Why should you want to take it?
   - What will you learn?
   - What will you be able to do?

1. Course Mechanics

- Course Administration
  - Everything you need to know will be on the course web page:

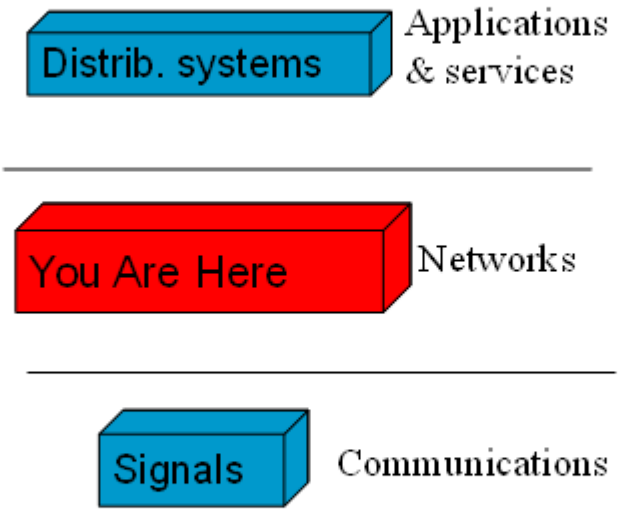
    http://www.cs.washington.edu/461/

  - Most everything (lecture schedule, reading, assignments, section materials, …) is linked off the schedule
- The project
  - Substantial programming effort
  - Teams of 1, 2, or perhaps 3
  - The project and the goals of the course are intimately intertwined

2. The Course

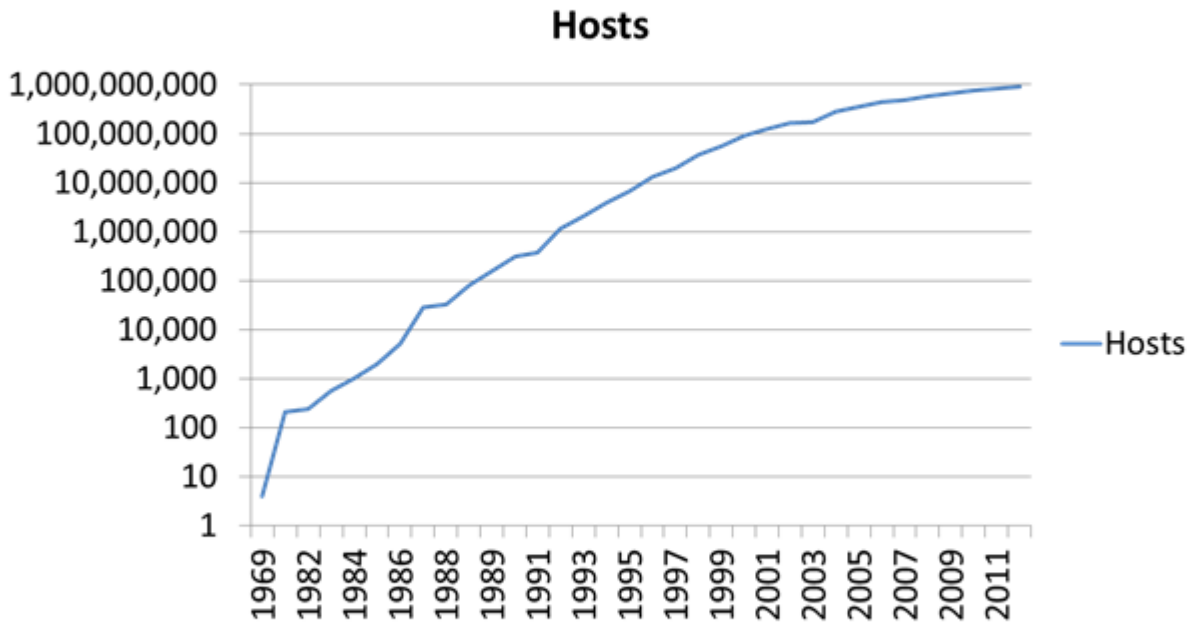
- The Internet has had a huge effect (on everything)
  - Mobile networking is extending that reach of that effect
- It's likely that networking will be a consideration in most everything you will design
- *One* goal of the course is to explain the basic foundation of the Internet
  - The Internet vs. networking

Focus of This Course



Networks are combinations of hardware and software whose goal is to move bits from one system to another.

Learning from the Internet



1. It's still here!
2. How is this possible?

One Part of the Answer

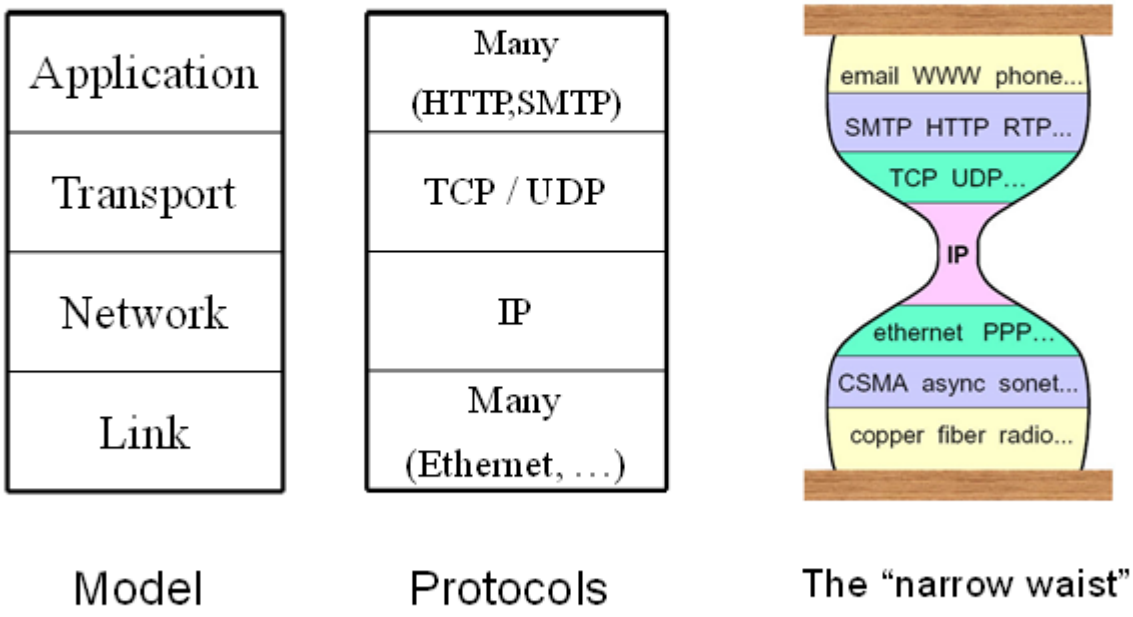
- The "End to End Argument" (Reed, Saltzer, Clark, 1984):

  Functionality should be implemented at a lower layer only if it can be correctly and completely implemented. (Sometimes an incomplete implementation can be useful as a performance optimization.)

- Tends to push functions to the endpoints, which has aided the transparency and extensibility of the Internet.

- The network makes almost no guarantees
  - We'll try to get you data to the destination, eventually. Maybe.
  - No performance guarantee.
  - No reliability guarantee.

The "Narrow Waist" Layered Architecture



The Internet Vs. Internet Applications

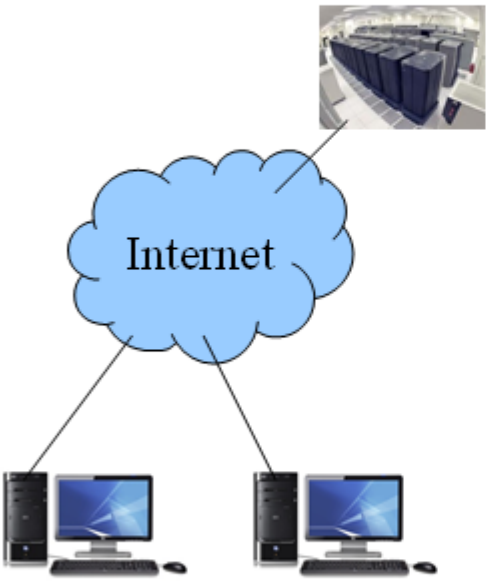
- It turns out that many of the problems the Internet must address come up again when writing Internet applications
  - Example: reliability
- Not too surprisingly, it turns out the techniques used by the Internet are useful in these applications
- So, a second goal of the course is to relate the design decisions made by the Internet to the problems that you're likely to encounter in designing applications

The Project

- The project is intended to help generalize from the specifics of the Internet to other scenarios
  - We'll use techniques used by the Internet to solve the same problems the Internet solves
- In some sense, the project starts where the course ends
  - It starts by using the TCP and UDP protocols to communicate
- The project has multiple parts.
  - I have a plan for working on part C even if part B doesn't work
- Groups of 2, or possibly 3 or 1

Project Plan

Client-Server

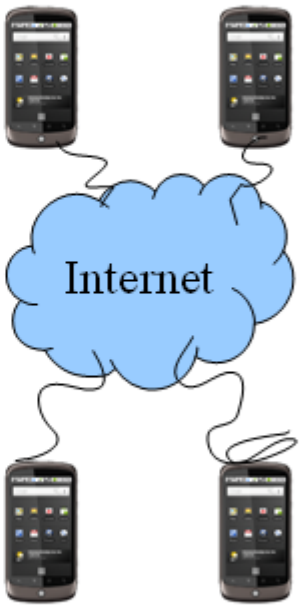


Peer-to-Peer

Project Plan

Client-Server

Peer-to-Peer



Client Platform: Android Phones

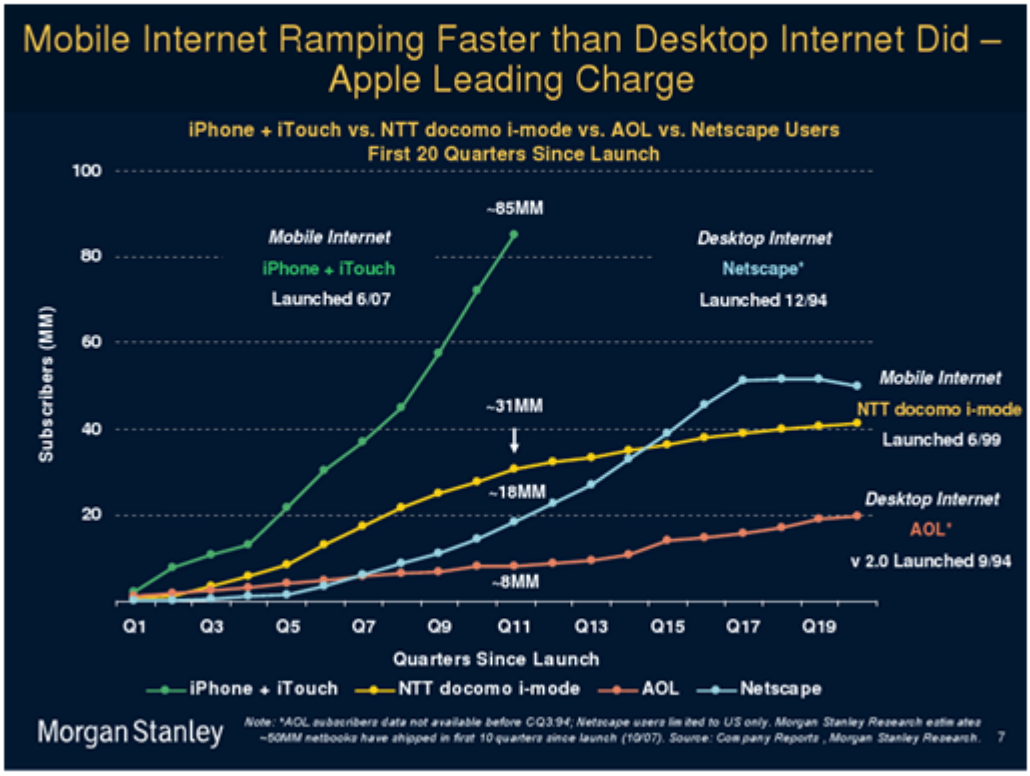
The Emulator

The Phones

Projects On Phones: Pros

- Pros
  - Fun…
  - The core ideas and experiences of the course apply
  - Mobile devices will be/are the "standard platform"

Smartphone Adoption Rate



2011 Data

Worldwide smart phone and client PC shipments
Shipments and growth rates by category, Q4 2011 and full year 2011

Category	Q4 2011 shipments (millions)	Growth Q4'11/Q4'10	Full year 2011 shipments (millions)	Growth 2011/2010
Smart phones	158.5	56.6%	487.7	62.7%
Total client PCs	120.2	16.3%	414.6	14.8%
- Pads	26.5	186.2%	63.2	274.2%
- Netbooks	6.7	-32.4%	29.4	-25.3%
- Notebooks	57.9	7.3%	209.6	7.5%
- Desktops	29.1	-3.6%	112.4	2.3%

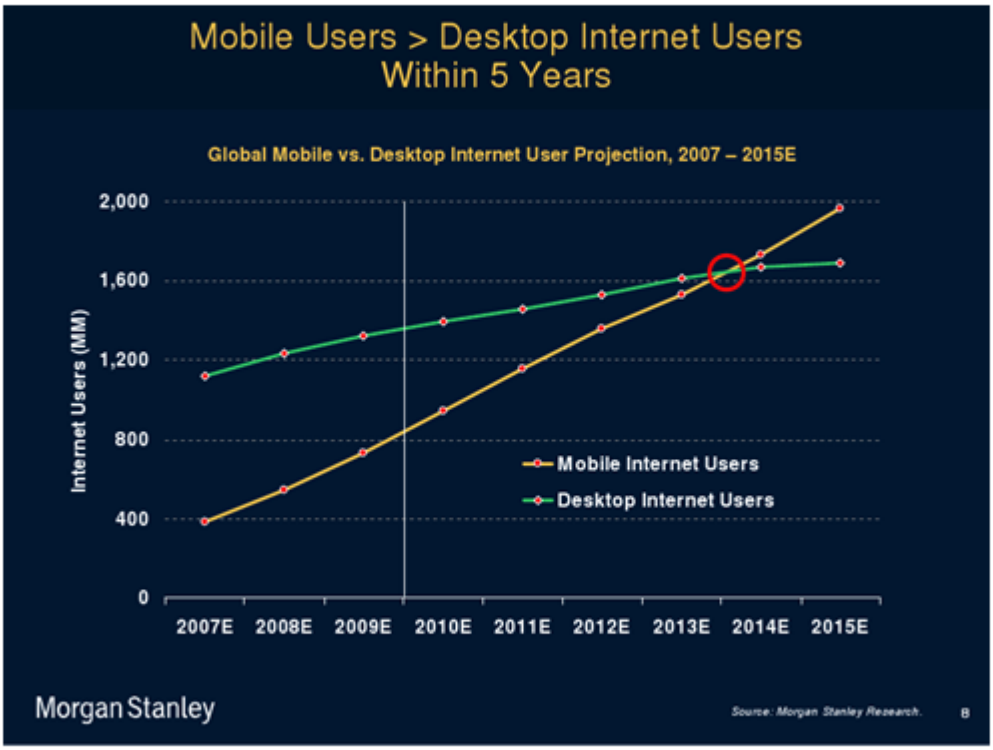
Source: Canalys estimates © Canalys 2012

Worldwide smart phone market
Shipments by platform, full year 2011

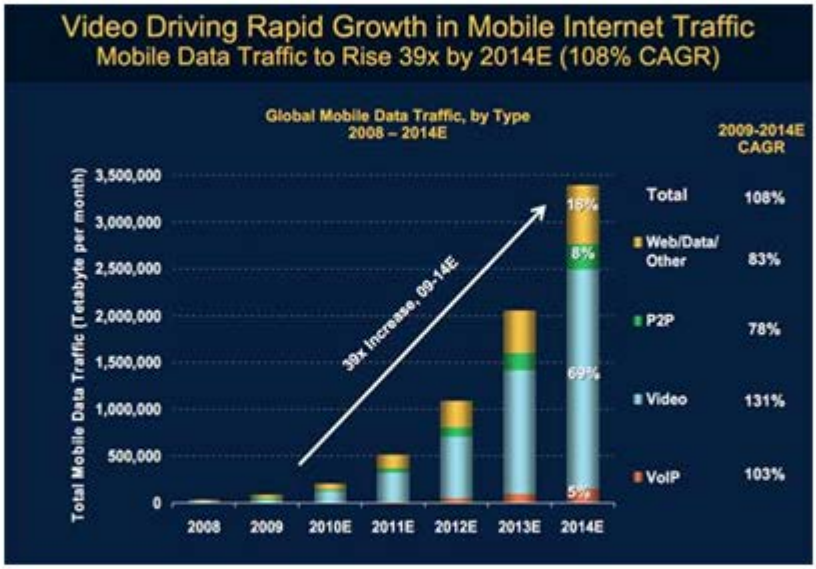
Platform	Full year 2011 shipments	Share (%)	Growth Q4'11/Q4'10
Total	487.7	100.0%	62.7%
Android	237.8	48.8%	244.1%
iOS	93.1	19.1%	96.0%
Symbian	80.1	16.4%	-29.1%
BlackBerry	51.4	10.5%	5.0%
bada	13.2	2.7%	183.1%
Windows Phone	6.8	1.4%	-43.3%
Others	5.4	1.1%	14.4%

Source: Canalys estimates © Canalys 2012

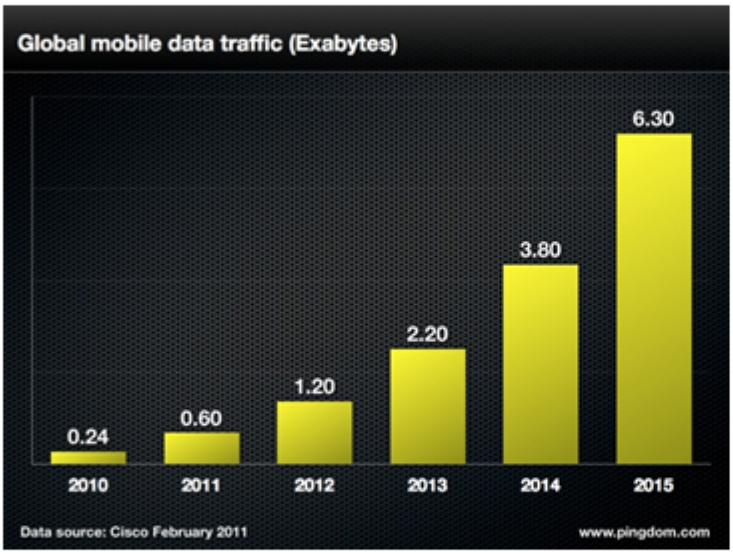
Number of Users



Mobil Traffic: Types



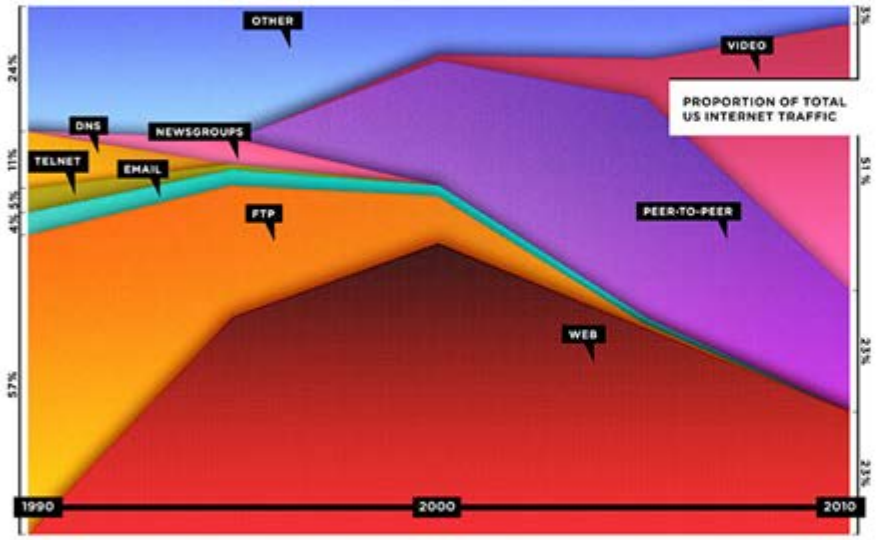
Total Mobile Traffic



Cisco's projection for total Internet traffic in 2014 is 766.8 exabytes

1 exabyte = 1 billion gigabytes

Total Internet Traffic By Type



Cisco's projection for 2014: 91% of all traffic is video

Phone Projects: Cons

- Cons
  - Eclipse
  - Java
  - Phone UI
  - Some somewhat different programming constructs
  - Hard to grade...

- Time is built into the schedule to ease into the new parts
  - Project 1's "programming" is mainly Eclipse + Android setup
  - Project 2's programming includes Android + UI

How Are Phones Different Than PCs?

- Limited resources:
  - Energy, compute, memory, persistent storage, connectivity, screen size
- To keep things simple, we try to ignore the limitations, as best we can, but…
- They affect the basic design of the system (Android)
  - Some things are a bit harder than you'd expect
  - The design of the project software is more intricate than you might expect

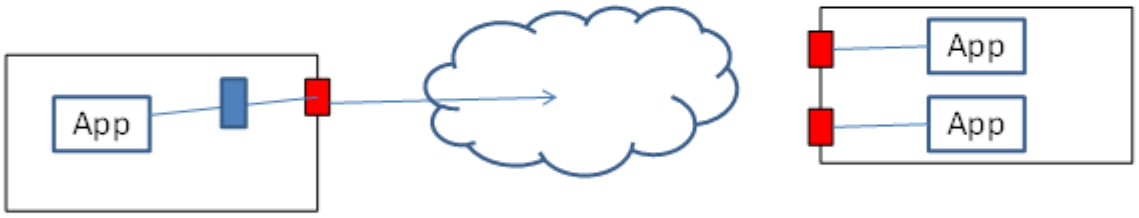
Part II:
Project Background:
Sockets

The Socket API

- The network API is most commonly exposed through a *socket interface*
- A socket is an OS-provided abstraction representing a communication endpoint
  - Process to process "link"
- Sockets commonly support two protocols:
  - UDP: "unreliable, datagram, connectionless"
  - TCP: "reliable, stream, connection-based"

UDP Socket API

- UDP is an unreliable datagram service
  - It sends "packets" (fixed maximum size units)
  - The packet either gets to the destination or it doesn't



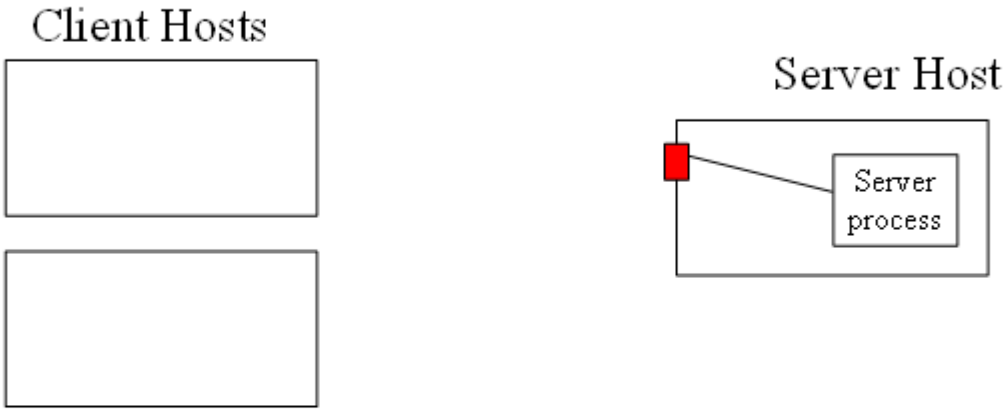
Sender needs to *name* the destination.

IP address names a remote host (sort of). Example: 128.208.1.139
Port number and protocol names a socket on that host.

UDP Sockets in Java

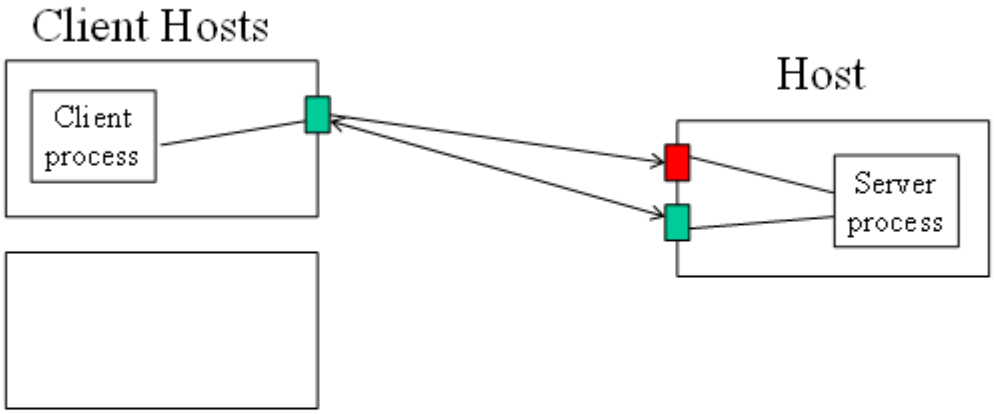
```
//-------------------------------------------------------------
// Sending

DatagramSocket socket = new DatagramSocket(localPort);

DatagramPacket pkt = new DatagramPacket( buf,
                                         buf.length,
                                         dest IP and port );

socket.send(pkt);

//-------------------------------------------------------------
// Receiving

socket.receive(pkt);
```

TCP Socket API – The Typical Case



1.Server process is launched, creates a socket, and waits someone to connect to it.

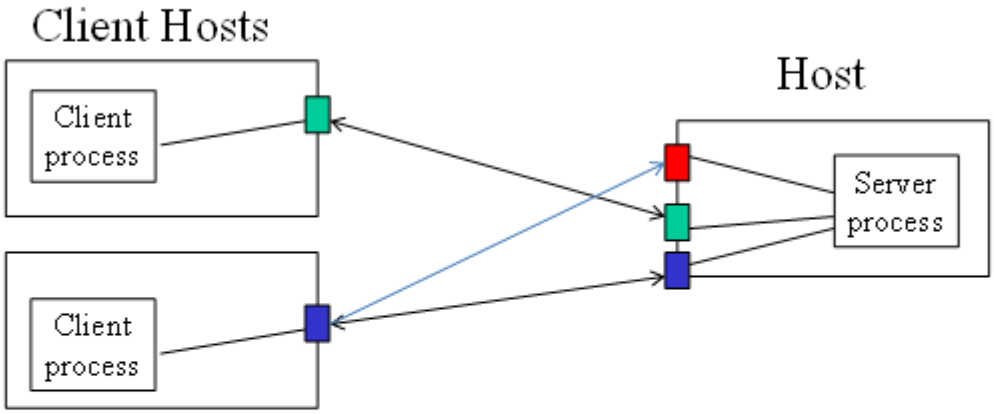
TCP Socket API (2)



1.Server process is launched, creates a socket, and waits someone to connect to it.

2.Client process is launched on some host, creates a socket, and causes it to be contact the server-side socket. This creates a new socket at the server, representing this particular connection. A connection is a stream.

TCP Socket API (3)



1. Server process is launched, creates a socket, and waits someone to connect to it.

2. Client process is launched on some host, creates a socket, and causes it to be contact the server-side socket

3. Another client does the same thing…

Dealing with Errors

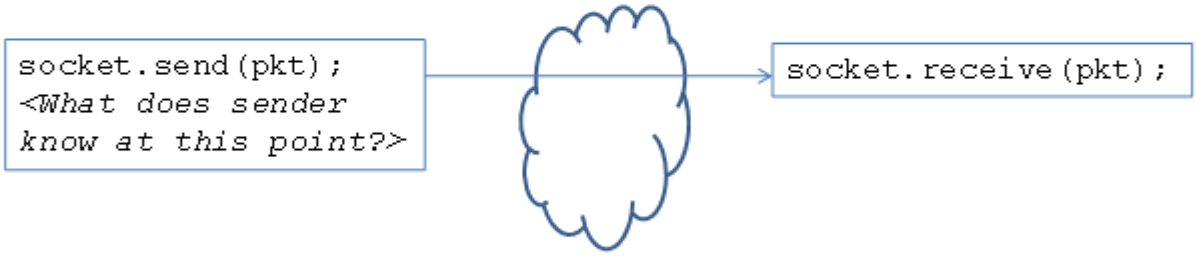
- What we're used to:

```
x = 5;
y = x * x;
```

```
try {
  fis = new FileInputStream("foo.txt");
  while ( fis.read(buf) >= 0 ) {
        …
  }
  fis.close();
} catch (IOException e) {
  …
}
```

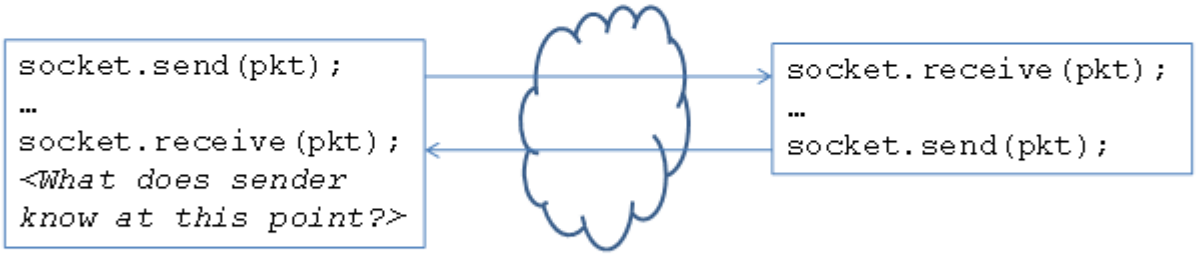
Networks and Errors

- What can fail?
- What happens when it fails?



Mechanism #1: Positive Acknowledgements

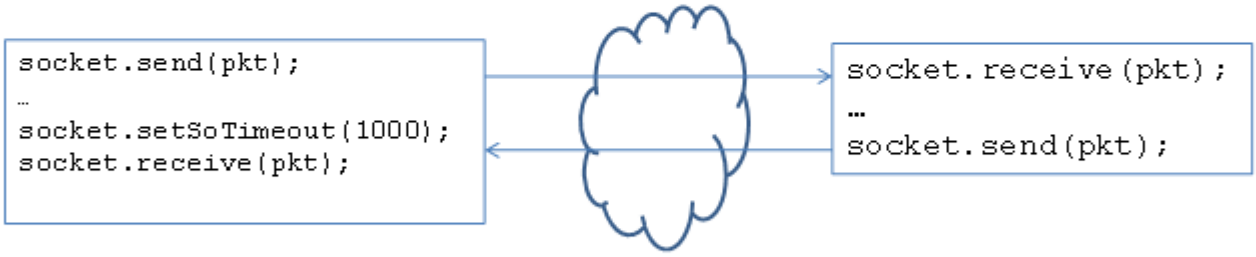
- The sender receives the positive acknowledgement only if the receiver received the original message



- What may fail?
  What happens when it fails?

Mechanism #2: Timeouts

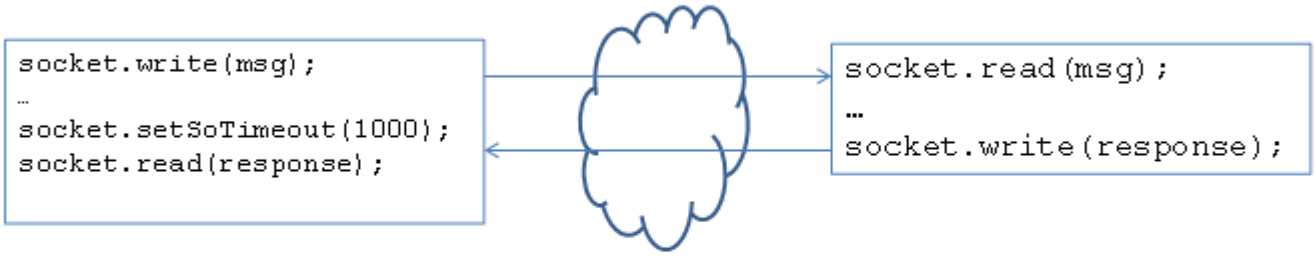
- Sender can't block forever trying to receive acknowledgement
- Eventually must time out, presuming something went wrong



- What may fail?
  What happens when it fails?

Does Using TCP Solve the Reliability Problem?

- (This isn't quite legal Java, for space reasons, but it's close.)



- What does TCP mean by "reliable"?
- What may fail?
  What happens when it fails?

For Project 1

- We will worry about blocking forever on reception
  - Code (sometimes) won't run to completion if you ignore that issue
- We won't worry about anything more
  - We'll know a communication attempt failed, but we won't try to deal with the failure

Part III:
Protocols and Layering

What to Watch For

(Now and later in the course)

- Layering (a familiar idea)
  - In networks, corresponds to a mechanism used when sending data (*encapsulation*)
  - Layering is an opportunity for *interposition*
    - Used extensively in networking
    - Seen later in the course
- Protocols
  - Layer n to layer n communication
  - A protocol is correct for some failure model (and perhaps incorrect for others)

Protocols and Layering

We need abstractions to handle all this system complexity

A protocol is an agreement dictating the form and function of data exchanged between parties to effect communication

Two parts:

- Syntax: format -- where the bits go
- Semantics: meaning -- what the words mean, what to do with them

Examples:

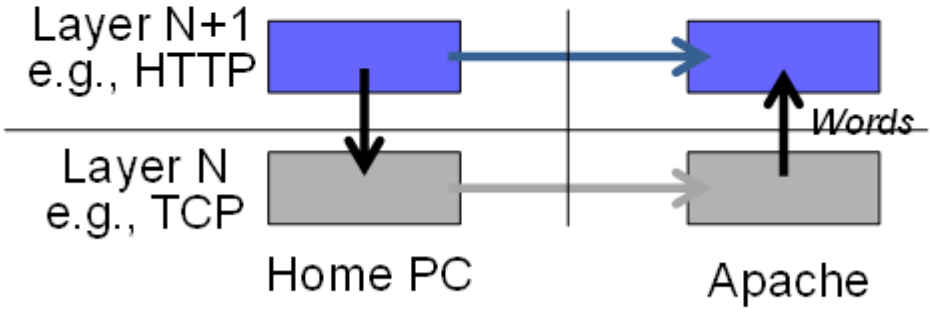
- Ordering food from a drive-through window
- TCP/IP, the Internet protocol
- HTTP, for the Web

Protocol Standards

- Why do we need standards?
- Different functions require different protocols
- Thus there are many, many protocol standards
  - E.g., IP, TCP, UDP, HTTP, DNS, FTP, SMTP, NNTP, ARP, Ethernet/802.3, 802.11, RIP, OSPF, 802.1D, NFS, ICMP, IGMP, DVMRP, IPSEC, PIM-SM, BGP, ...
  - every distributed application requires a protocol...
- Organizations: IETF, IEEE, ITU
- IETF (www.ietf.org) specifies Internet-related protocols
  - RFCs (Requests for Comments)
  - "We reject kings, presidents and voting. We believe in rough consensus and running code." – Dave Clark.

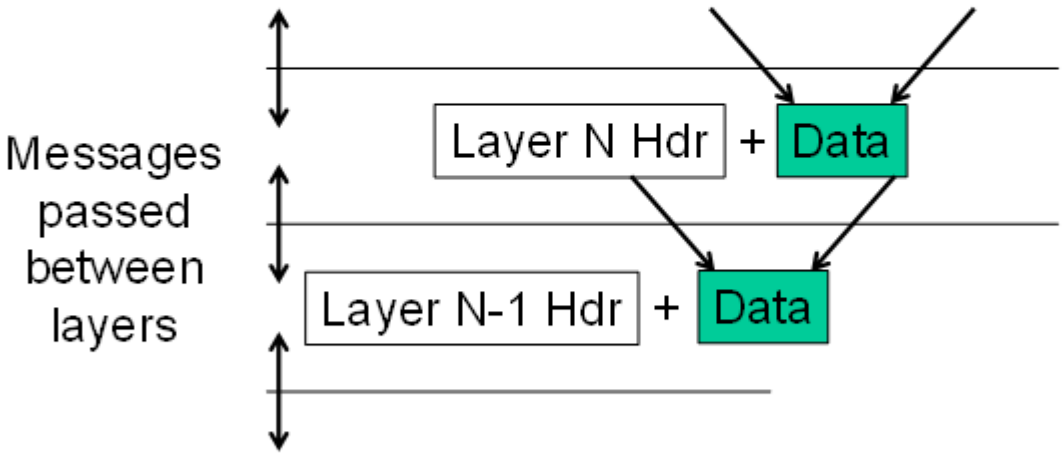
Layering and Protocol Stacks

- Layering is how we combine protocols
  - Higher level protocols build on services provided by lower levels
  - Peer layers communicate with each other
    - Each lower level can be viewed as a communication channel with some characteristics, analogous to a wire between two nodes



Layering Mechanics

Encapsulation and de(en)capsulation

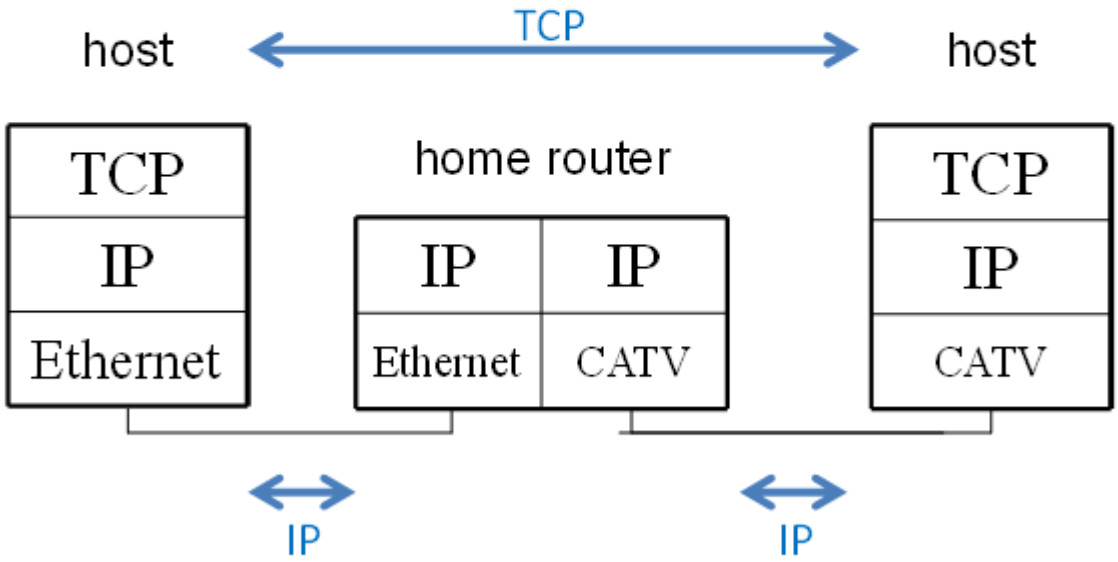


A Packet on the Wire

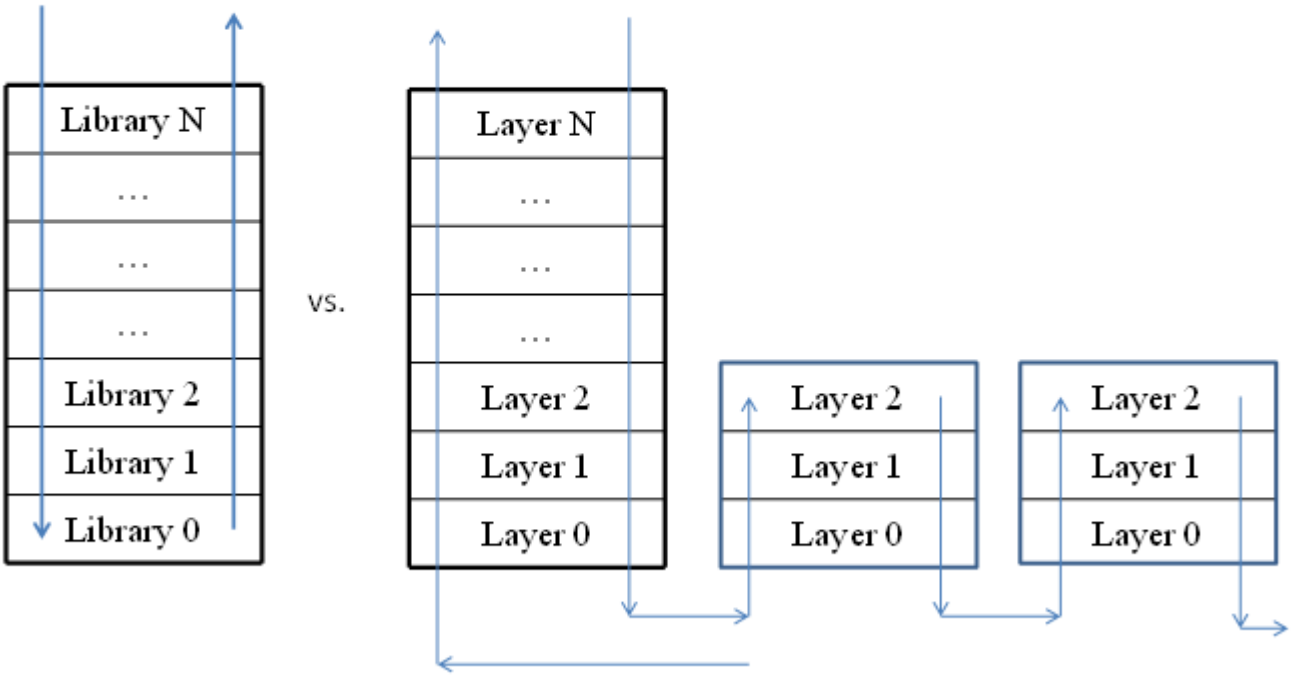
- Starts looking like an onion:

- This isn't entirely accurate
  - Reality can be a bit messier, but idea is accurate.
- But you can see that:
  - layering adds overhead
  - one protocol's header is another protocol's data

Example – Layering at work

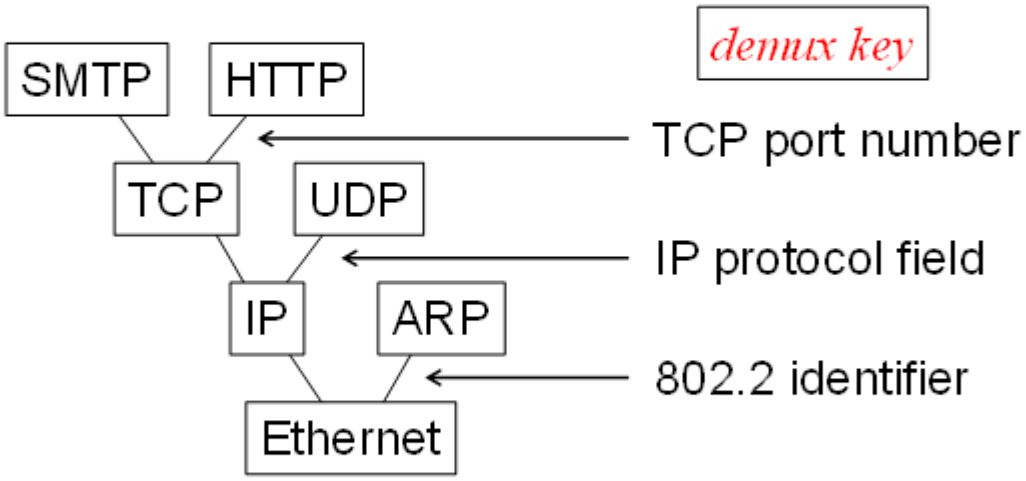


Comparison with SW Layering

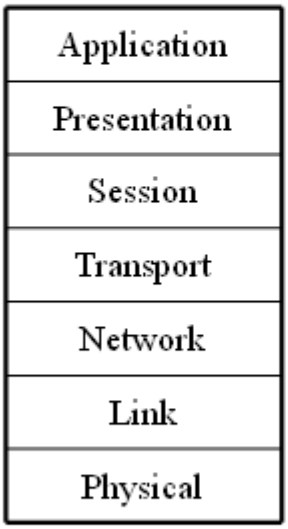


More Layering Mechanics

Multiplexing and demultiplexing in a protocol graph



OSI "Seven Layer" Reference Model



Their functions:

- Up to the application
- Encode/decode messages
- Manage connections
- Reliability, congestion control
- Routing
- Framing, multiple access
- Symbol coding, modulation

Layers and the End-to-End Argument

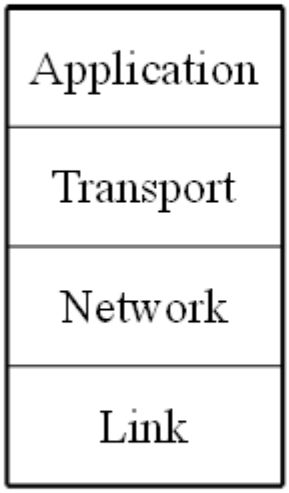
Key Question: What functionality goes in which protocol?

- The "End to End Argument" (Reed, Saltzer, Clark, 1984):

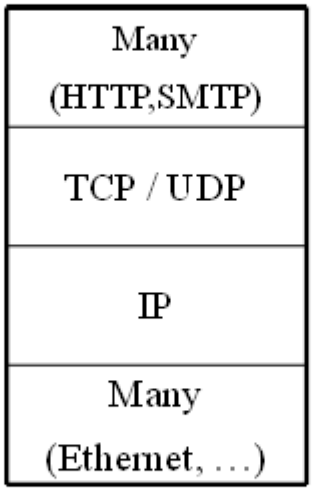
Functionality should be implemented at a lower layer only if it can be correctly and completely implemented. (Sometimes an incomplete implementation can be useful as a performance optimization.)

- Tends to push functions to the endpoints, which has aided the transparency and extensibility of the Internet.

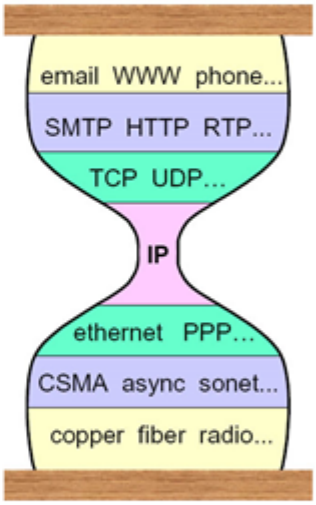
Internet Protocol Framework



Model



Protocols



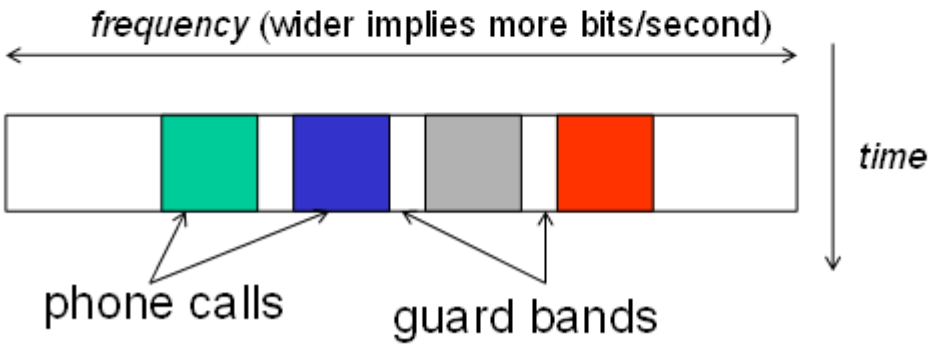
The "narrow waist"

How to share: multiplexing

- How should you multiplex (share) a resource among multiple users?
  - In particular, how do you share network links and switches?
- There are two classes of approaches:
  - Static Partitioning ("Reservations")
    - » Time Division Multiplexing (TDM)
    - » (Space) Frequency Division Multiplexing (FDM)
  - Statistical Multiplexing ("On demand")
    - » Packet Switching

Frequency Division Multiplexing

- Simultaneous transmission in different frequency bands
- "Speaking at different pitches"
  - e.g., take one 3MHz signal and break it into 1000 3KHz signals
    - Analog: Radio, TV, AMPS cell phones (800MHz)
  - also called Wavelength DMA (WDMA) for fiber



Time Division Multiplexing

- "Slice up" the single frequency band among users
- "Speaking at different times"
  - Digital: used extensively inside the telephone network
  - T1 (1.5Mbps) is 24 x 8 bits/125us; also E1 (2Mbps, 32 slots)

Statistical Multiplexing: Packet Switching

The basic idea is very familiar from everyday life (e.g., washrooms on airplanes).

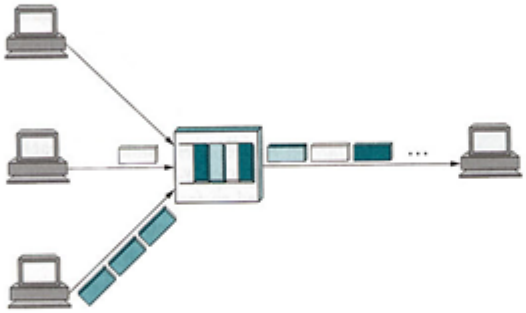


Figure 1.6 A switch multiplexing packets from multiple sources onto one shared link.

Bandwidth is allocated on demand.

Statistical Multiplexing

- Three "details":
  - why is there buffering?
  - why must there be a maximum packet size (even if there were infinite buffering)?
  - under what conditions might a switch run out of buffers?

- Static multiplexing can suffer large overheads when some client can't make use of its allocation.
  - Are there any overheads involved in packet switching?

Statistical vs. Static / Performance Measures

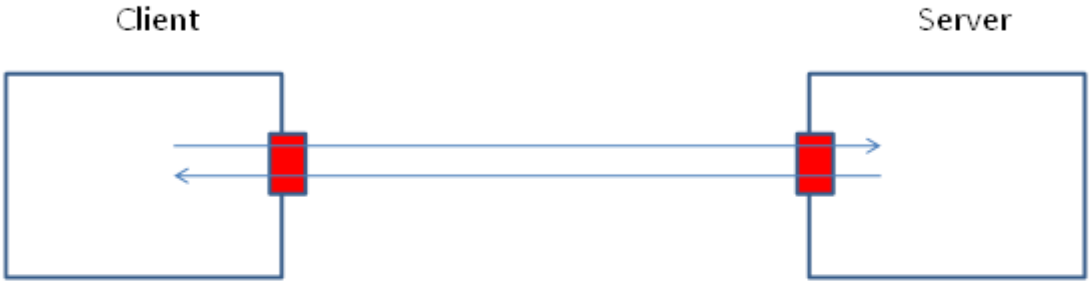
- Which is better?
  - We have to decide what we mean by better.
  - We often do this by talking about types of performance measures, and the kinds of workloads that care about them.
    (This gets us near quality-of-service issues, which are addressed late in the course.)

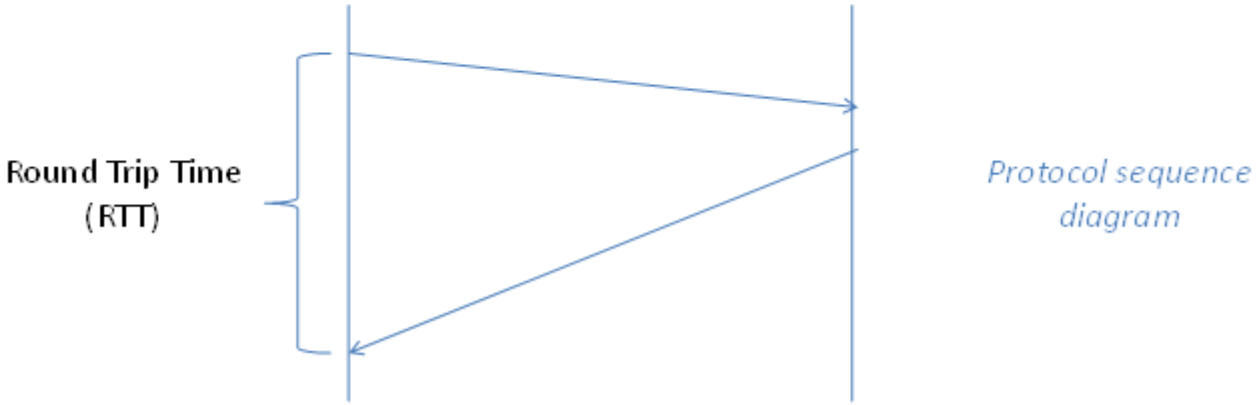
- There are many different performance measures one might be interested in
  - average throughput (goodput)
    - important when you're sending a lot of data (e.g., file transfer)
  - average latency
    - important when you're sending a little data and you want a response (telnet/ssh)
  - variance in throughput and latency (jitter)
    - important to streaming media (audio, video, Skype (VoIP))
    - real time systems (e.g., airplane flight control)
  - minimum guaranteed throughput / maximum guaranteed latency
    - when does the client know that it won't get it's minimum?
    - example use: deciding on an encoding quality for streaming audio/video

Performance: Project 1

- Project 1 is about many things, but one of them is measuring performance
- The `ping` application measures latency
- The `dataxfer` application measures throughput and error rate

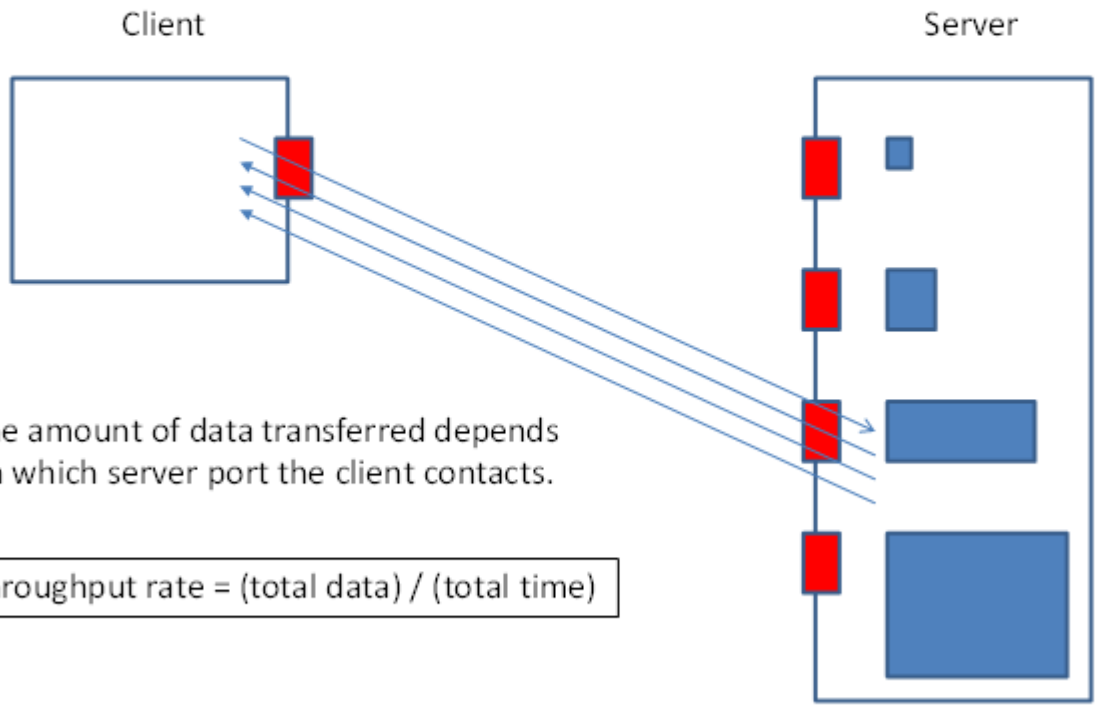
Ping: latency





Protocol sequence diagram

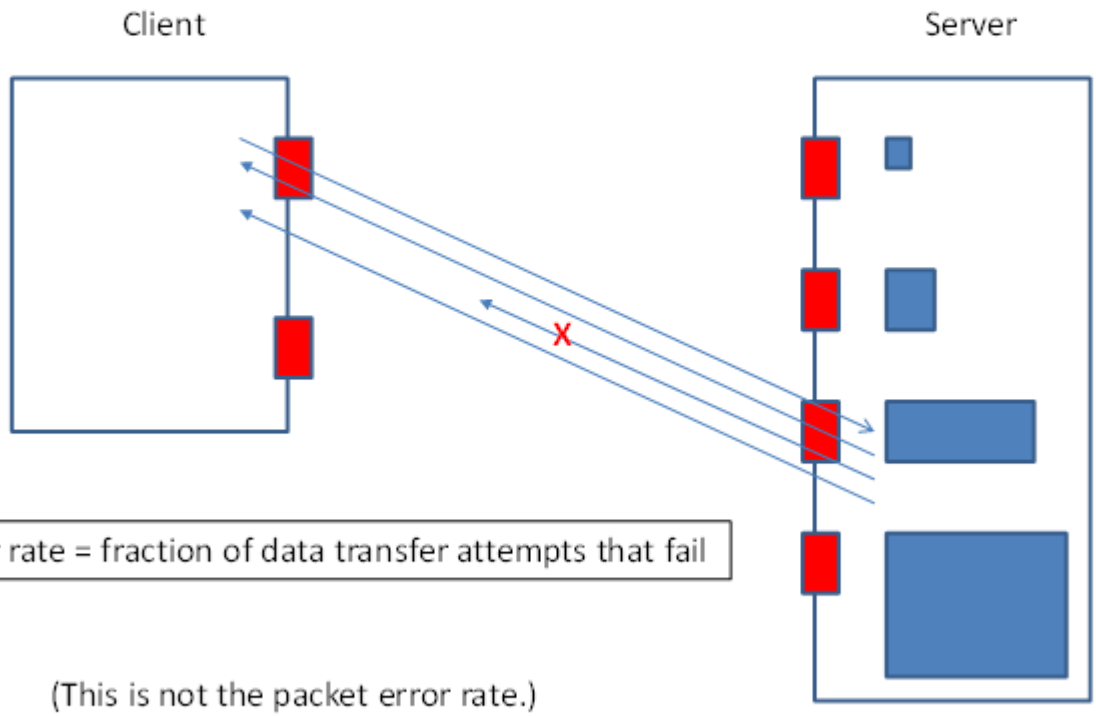
Data Transfer: throughput



The amount of data transferred depends on which server port the client contacts.

Throughput rate = (total data) / (total time)

Data Transfer: error rate



Error rate = fraction of data transfer attempts that fail

(This is not the packet error rate.)