Today's Lecture

• Administrative matters
• Course Overview
  – topics covered
  – design & prototyping projects
• Introduction to networking
Class Structure

• Friday 3:45-6:30pm
• Lecture format
  – Slides, Board, ...
  – Interactive
• Two 80 min sessions
  – with a 10 min break in between
Contact Information

• **Instructor:**
  – Prof. D. Raychaudhuri
    • Email: ray@winlab.rutgers.edu
    • Office Hours: by appt, WINLAB Tech Center or Core 501

• **Project TA: Sumit Maheswari**
  – Email: sumitm@winlab.rutgers.edu
  – Office hours: by appt
Class Resources

- Web page: http://www.winlab.rutgers.edu/comnet2
- Mailing list: comnet2@winlab.rutgers.edu
- Sign up for mailing list at: http://lists.winlab.rutgers.edu/listinfo/comnet2
Course Readings

• Textbook (required, to be used for ~60% material)
  – Peterson & Davie, "Computer Networks: A Systems Approach", Morgan Kaufman, 4th or 5th editions

• Handouts posted to course website

• Research papers in networking
  – to be distributed either online or in class
  – collection of classical and topical research
    • ~10 papers and standards documents
    • required reading to supplement text book overview
Course Grading

• Class participation & homework: 5%
  – Brief in-class presentations
  – Assigned homework from textbook
• Midterm (25%) and Final (35%)
  – Open book, 1 page of notes permitted; includes both descriptive and numerical problems
• Design & Prototyping Assignments: 35%
  – network architecture paper 10%
  – protocol project & report 25%
• No makeup exams, no extra credit work
Student Commitments

- Keep up with your reading
  - read applicable text book chapter and distributed papers/RFC’s before and after each class
- Sharpen your programming skills
  - study C/C++ & Unix programming as needed and work on simple programming exercises early in the semester
- Work independently (except group projects)
  - no “collaboration” of any sort
- Turn in assignments on time
- Make sure assignments are gradable
  - follow project and program submission rules
Prerequisites

• Curricular prerequisites
  – Computer Networks I or equivalent
  – General communications and computer architecture/OS background

• Skills
  – C/C++ programming
    • significant programming project
  – use of design and analysis tools
Course Topics

- Introduction
- Network Principles
- Shared Media/MAC
- Packet switching
- IP Basics
- IP Advanced
- Mobility Protocols
-- mid-term

- Software defined networks
- Network security
- Transport layer
- Higher-layer protocols
- Hardware issues
- Case studies and research topics
  - Content networks
  - Cloud computing services
  - future Internet architecture
Projects

- Network architecture paper
  - top-down design
  - requirements
  - specifications
  - system analysis
  - report

- Warm-up Projects
  - C/C++ programming exercises
  - Unix sockets, etc.
  - simple link protocols

- Network prototyping project
  - new routing protocol
  - Click software platform
  - student teams will write competing protocol specs
  - meetings to specify “standard”
  - group demo & inter-op demo
What is the problem?

Applications

The Global Network

Technology

Robustness

Scale
Application Considerations

• Variety of current and future applications to be supported by the network
Application Considerations

• Application input to network
  – traffic data rate
  – traffic pattern (bursty or constant bit rate)
  – traffic target (multipoint or single destination, mobile or fixed)

• Network service delivered to application
  – delay sensitivity
  – loss sensitivity
  – Mobility, location, multicast, etc.
Multimedia and Video Applications

The classic Internet App: voice, video, data; streaming BW, latency, ..
Large # devices (~100B+), low power, bursty data, ..
Application Considerations (Data Centers, Cloud)

High traffic volume, latency sensitive, …
Reliable File Transfer

- Loss sensitive
- Not delay sensitive relative to round trip times
- Point-to-point or multipoint
- Bursty
Remote Login

- Loss sensitive
- Delay sensitive
  - subject to interactive constraints
  - can tolerate up to several hundreds of milliseconds
- Bursty
- Point to point
Network Audio

• Relatively low bandwidth
  – Digitized samples, packetized
• Delay variance sensitive
• Loss tolerant
• Possibly multipoint, long duration sessions
  – natural limit to number of simultaneous senders
Network Video

• High bandwidth
• Compressed video, bursty
• Loss tolerance function of compression
• Delay tolerance a function of interactivity
• Possibly multipoint
• Larger number of simultaneous sources
Web

- Transactional traffic
  - short requests, possibly large responses
- Loss tolerant
- Delay sensitive
  - human interactivity
- Point-to-point (multipoint is asynchronous)
What is....

The Global Network

- Structure
- Metrics
- Failure modes
- Functions
Network Structure

- National/Global Networks, Backbones
- Regional Networks, ISP
- Local/Access Networks
- Nodes, Hosts, CPE
- Servers, Data Centers
- Routers, Switches
- Links, LAN
Network Topologies

- Ring
- Mesh
- Star
- Fully Connected
- Line
- Tree
- Bus
Internet Topology
Internet Topology Visualization
Network Metrics

• Bandwidth
  – transmission capacity

• Delay
  – queueing delay
  – propagation delay (limited by $c$)

• Delay-Bandwidth product
  – important for control algorithms
Bandwidth versus Latency

- Relative importance
  - 1-byte: 1ms vs 100ms dominates 1Mbps vs 100Mbps
  - 25MB: 1Mbps vs 100Mbps dominates 1ms vs 100ms

- Infinite bandwidth
  - RTT dominates
    - Throughput = TransferSize / TransferTime
    - TransferTime = RTT + 1/Bandwidth x TransferSize
  - 1-MB file to 1-Gbps link as 1-KB packet to 1-Mbps link
Delay x Bandwidth Product

- Amount of data “in flight” or “in the pipe”
- Example: 100ms $\times$ 45Mbps = 560KB
Chapter 1, Figure 1.20
Network Failures

- Packet loss
  - queue overflows
  - line noise
- Node or link failures
- Routing transients or failures
Statistical Multiplexing Gain

1 Mbps link; users require 0.1 Mbps when transmitting; users active only 10% of the time.

- Circuit switching: can support 10 users
- Packet switching: with 35 users, probability that $\geq$10 are transmitting at the same time = 0.0004.
Back in the old days..
Then came TDM..
Logical network view
Packet switching (Internet)
Packet Switching

Interleave packets from different sources

- Efficient: resources used on demand
  - statistical multiplexing

- General
  - multiple types of applications

- Accommodates bursty traffic
Characteristics of Packet Switching

- Store and forward
  - packets are self contained units
  - can use alternate paths - reordering

- Contention
  - congestion
  - delay
Protocols

• On top of a packet switched network, need
• Set of rules governing communication between network elements (applications, hosts, routers)
• Protocols define:
  – format and order of messages
  – actions taken on receipt of a message
Protocols (contd.)

• Building blocks of a network architecture
• Each protocol object has two different interfaces
  – *service interface*: operations on this protocol
  – *peer-to-peer interface*: messages exchanged with peer
• Term “protocol” is overloaded
  – specification of peer-to-peer interface
  – module that implements this interface
Layering: technique to simplify complex systems
Layering

TCP/IP - model

- Application
- Transport
- Internet
- Network interface
Layering Characteristics

- Each layer relies on services from layer below and exports services to layer above
- Interface defines interaction
- Hides implementation - layers can change without disturbing other layers (black box)
Packet Headers

Packet Headers can contain:
- addresses, flow ID, pkt type, service type, error checks, QoS, …
ISO Architecture

End host

- Application
  - Presentation
  - Session
  - Transport
  - Network
  - Data link
  - Physical

One or more nodes within the network

End host

- Application
  - Presentation
  - Session
  - Transport
  - Network
  - Data link
  - Physical
Internet Architecture

- Defined by Internet Engineering Task Force (IETF)
- Hourglass Design
- Application vs Application Protocol (FTP, HTTP)
Internet Reference Model

- IP is the “narrow waist” of the Internet
- Supports many different links below and apps above.
- Examples of common protocols in each layer
Layering General Issues

- Reliability
- Flow control
- Fragmentation
- Multiplexing
- Connection setup (handshaking)
- Addressing/naming (locating peers)
Example: Transport layer

- First end-to-end layer
- End-to-end state
- May provide reliability, flow and congestion control
Example: Network Layer

- Point-to-point communication
- Network and host addressing
- Routing
Encapsulation

Wire

HTTP

TCP

IP

802.11

TCP

HTTP

802.11

TCP

HTTP

802.11

TCP

HTTP

802.11

TCP

HTTP

802.11

TCP

HTTP

802.11

TCP

HTTP

802.11

TCP

HTTP

802.11

TCP

HTTP
Demultiplexing

- Incoming messages must be passed to the protocol it uses.
- Done with demultiplexing keys in the headers
Inter-Process Communication

• Turn host-to-host connectivity into process-to-process communication.
• Fill gap between what applications expect and what the underlying technology provides.
Network-Application Interface

- Defines how apps use the network
  - Lets apps talk to each other via hosts;
  - hide the details of the network
  - Sockets let apps attach to the local network at different ports
IPC Abstractions

• Request/Reply
  – distributed file systems
  – digital libraries (web)

• Stream-Based
  – video: sequence of frames
    • 1/4 NTSC = 352x240 pixels
    • (352 x 240 x 24)/8=247.5KB
    • 30 fps = 7500KBps = 60Mbps
  – video applications
    • on-demand video
    • video conferencing
Interfaces

Chapter 1, Figure 1.10
Interfaces (contd.)

```
send(IP, message)  deliver(TCP, message)
```
Interfaces (contd.)

Chapter 1, Figure 17
Protocol Machinery

- Protocol Graph
  - most peer-to-peer communication is indirect
  - peer-to-peer is direct only at hardware level
Machinery (cont)

- Multiplexing and Demultiplexing (demux key)
- Encapsulation (header/body)

Chapter 1, Figure 1.12
Socket Basics

Client
- Create Socket
  - socket
- Bind socket to port
  - bind
- Send the connect request to server and waiting server to respond

Server
- Create Socket
  - socket
- Bind socket to port
  - bind
- Client traffic is lining up here and wait to be accepted
- Client accepted will be ready for send and receive data between server
- Process inside the bracket will happen when request of client is accepted.

Connect
- Send and receive data
  - recvfrom/sendto
- close

Listen
- Accept
- Send and receive data
  - sendto/recvfrom
- close
## Socket API

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCKET</td>
<td>Create a new communication endpoint</td>
</tr>
<tr>
<td>BIND</td>
<td>Associate a local address with a socket</td>
</tr>
<tr>
<td>LISTEN</td>
<td>Announce willingness to accept connections; give queue size</td>
</tr>
<tr>
<td>ACCEPT</td>
<td>Passively establish an incoming connection</td>
</tr>
<tr>
<td>CONNECT</td>
<td>Actively attempt to establish a connection</td>
</tr>
<tr>
<td>SEND</td>
<td>Send some data over the connection</td>
</tr>
<tr>
<td>RECEIVE</td>
<td>Receive some data from the connection</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Release the connection</td>
</tr>
</tbody>
</table>
## Client & Server Program Outline

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>socket() //make socket</td>
<td>socket() //make a socket</td>
</tr>
<tr>
<td>getaddrinfo() //server and port name</td>
<td>getaddrinfo() //for port on this host</td>
</tr>
<tr>
<td>//www.example.com:80 bind() //associate port with socket</td>
<td></td>
</tr>
<tr>
<td>connect() //connect to server [block]</td>
<td>listen() //prepare to accept connections</td>
</tr>
<tr>
<td>...</td>
<td>accept() //wait for a connection [block]</td>
</tr>
<tr>
<td>send() //send request</td>
<td>... recv() //wait for request</td>
</tr>
<tr>
<td>recv() //await reply [block]</td>
<td>...</td>
</tr>
<tr>
<td>... //do something with data!</td>
<td>send() //send the reply</td>
</tr>
<tr>
<td>close() //done, disconnect</td>
<td>close() //eventually disconnect</td>
</tr>
</tbody>
</table>
Using Sockets

Client (host 1)
1: socket
5: connect*
7: send
8: receive*
10: close

Server (host 2)
connect
request
reply
disconnect

Time
1: socket
2: bind
3: listen
4: accept*
5: connect*
6: receive*
7: send
8: receive*
9: send
10: close

*=call blocks
Network Architecture

• Goal is to design a complete network solution that meets service requirements and cost constraints

• Design space includes
  – Application platform & software
  – Network topology
  – Core technologies
  – Protocols
  – Traffic engineering
  – Cost estimation
Connecting low power sensors and IoT devices to the Internet…
Concept Example 2: **Cloud-Assisted Vehicular**

Cloud Assisted Driving and Self-Driving will soon become feasible due to advances in wireless network (“5G”) and edge cloud technology.
Designing a Network

• Identify basic service requirements
  – transport service(s)
  – bit-rates to be supported
  – network API
  – # of users
  – terminal type (fixed, portable, etc.)

• Outline network topology
  – access network type (wired/wireless, span, etc.)
  – core network if any (node locations, span, etc.)
Requirements (contd.)

• List additional service and network features
  – QoS, video/audio, etc.
  – special routing (mcast, broadcast,..)
  – mobility
  – availability
  – reliability
  – security/authentication

• Rough system capacity (Mbps) and cost estimates ($/MB or $/user/mo)
# Requirements Analysis

- **Summary table listing key requirements**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport services</td>
<td>CBR, VBR-rt,..</td>
</tr>
<tr>
<td>Bit rate</td>
<td>0.1-10 Mbps</td>
</tr>
<tr>
<td># of users</td>
<td>~1000’s per access network</td>
</tr>
<tr>
<td>Terminal type</td>
<td>portable/mobile, fixed wireless</td>
</tr>
<tr>
<td>Topology</td>
<td>hierarchical, access/core</td>
</tr>
<tr>
<td>QoS features</td>
<td>selectable BW, stream support</td>
</tr>
<tr>
<td>Availability</td>
<td>99.9%</td>
</tr>
<tr>
<td>Reliability</td>
<td>99.99%</td>
</tr>
<tr>
<td>Security features</td>
<td>mobile authentication, on-air encryption</td>
</tr>
<tr>
<td>Cost</td>
<td>$0.1/MB or $50/mo/user</td>
</tr>
</tbody>
</table>
Network Components

- Key hardware components of a network
  - NIC ~10, 100, 155, 622, 1000 Mbps
  - shared media channels (Ethernet, HFC, wireless, satellite, ..) ~Mbps
  - point-to-point links (DSL, CAT-5, microwave, fiber,..)
  - switches (Ethernet, ATM, MPLS/IP) ~ Gbps -Tbps
  - routers (IP) ~Mbps - Gbps
Network Components

• Key software components of a network
  – CPE/Terminal OS & drivers
  – Application interface – “socket” spec
  – Transport layer protocol
  – Network layer protocol (at client)
  – Network layer protocol (at network elements)
  – Network management system
  – Any additional directories or network services
High-Level Design

- Select network topology based on geographic, capacity, reliability, etc.
- Partition into access network, core network, etc. as required
- Assign network hardware components to each subnetwork based on service and QoS requirements
- Define service API and protocol stacks
- Analyze network performance & cost and iterate until requirements are met
High Level Design

Technology choice
(e.g. MPLS optical)

Mbps needed?

Technology choice
(e.g. IP router)

Technology choice
(e.g. Ethernet SW)

Technology choice
(e.g. 802.11n)

Users
(#, density, mobility)

Access Net

Physical Span?

bps
Pkt size
Burst statistics
Stream parameters

bps/sq-m for wireless access
Today's Homework

• Peterson & Davie, Chap 1 (4th ed)
  - 1.3
  - 1.15
  - 1.17
  - 1.23
  - 1.28