EECS 489: Computer Networks

Lecture
• Time: MW 9:00-10:30
• Room: 1690 CSE

Discussion section:
• Time: Fri 9-10 (discussion starts this Friday)
• Room: 1690 CSE

Instructor: Z. Morley Mao
Office: 4629 CSE
Office hours: Mon 10:30-11:30, Wed 10:30-11:30, and by appt. (but never right before lecture)
Tel: +1 734 763 5407

GSI: Zhiyun Qian (zhiyun)
Office: 4917 CSE
Office hours: Tu 3:30-4:30PM, Fri 10-11AM, and by appt.

Web Site and Forum

Course Web site:
http://www.eecs.umich.edu/courses/eecs489/w10
• Course grade composition
• Policy on collaboration and cheating
• Grading and late days

Lecture slides will be posted on web site after lecture
Web site and forum are “required readings”

Prerequisites:
• EECS 482
• C

Grading Policy

1 Final Exam: 30%
1 Midterm Exam: 20%
3 Homeworks: 15%
3 Programming Assignments: 30%
Class Participation: 5%

Collaboration:
• All work must be done individually
• Cheating and plagiarizing are not tolerated

Regrade:
• Within 5 working days (except Final Exam, same day)
• Written request
• Whole work regraded

Late days:
• 4 free late days for the whole term
• 4% per 24 hours or fraction thereof thereafter

Required Textbook
Highly recommended Textbook
Source code at
http://www.unpbook.com/unpv13e.tar.gz

UNIX Network Programming
The Unix Networking API
Network Solutions
Richard Stevens
Second Edition
What this Course is NOT about

We do NOT cover:
• Homepage design, CSS, MySQL
• Photoshop, Flash, Silverlight, whatever
• Web site administration
• Web hosting and data center setup and maintenance
• DSL or cable modem setup
• LAN setup and administration
• How to connect to the Internet
• How to become an ISP
• How to run an ISP

What this Course Covers

Network Protocols and Architectures:
• Internet: a network of networks
• Application layer, naming
• Network layer: addressing and routing
• Link layer and wireless
• Transport layer and queue management
• Multimedia networking
• Security in networks

• Focus on the fundamental concepts, not merely the technology
• There will always be new protocols, but how do people design them?
• What are architectural principles?
• Provide high-level overview of computer networks
• Wired, wireless, cellular
• How they are interconnected

What is a Communication Network

Communication network offers one basic service: move information
• bird, fire, messenger, truck, telegraph, telephone, Internet ...

Another example, transportation network: move objects
• horse, train, truck, airplane ...

• a car carrying a trunk load of tapes can also move information . . .
What distinguish different types of networks?
• the services they provide
What distinguish the services?
• Latency
• Bandwidth
• a car carrying a trunk load of tapes has high bandwidth but low latency
• Loss rate
• Number of end systems supported
• Service interface (how to invoke the service?)
• Others
• Reliability, unicast vs. multicast, real-time...

Communication Networks Taxonomy

POTS Network:
* parses number dialed
* sets up a circuit between caller and callee
* sends signal to ring callee’s phone
* a circuit is set up between the two ends

Strengths:
* no end-point intelligence
* excellent voice performance

Weaknesses:
* difficult to add new services
* achieves performance and reliability by over-allocating resources (expensive)

The Internet:
* data parceled into packets
* each packet carries a destination address
* each packet is routed independently
* packets can arrive out of order
* packets may not arrive at all

Strengths:
* intelligence at end points
* decentralized control
* operates over heterogeneous access techs

Weaknesses:
* variable performance, no quality of service
* no trusted infrastructure
What’s the Internet

Internet: “network of networks”
- loosely hierarchical
- public Internet versus private intranet

Network components:
- hosts – communication endpoints: workstations, PDAs, cell phones, toasters, millions of them, running network applications
  - Applications: Web browser, Email clients, Chat clients, etc....
- links – carry bits from one place to another: fiber, copper, satellite, ...
- routers/switches – interconnect links: store and forward packets

History of the Internet

- 68-70’s: started as a research project, 56 kbps, initially 4 nodes (UCLA, UCSB, SRI, Utah) then < 100 computers
- 80-83: TCP/IP, DNS, ARPANET and MILNET split
- 85-86: NSF builds NSFNET as backbone, links 6 Supercomputer centers, 1.5 Mbps, 10,000 computers
- 87-90: link regional networks, NIS (NASA), ESNet (DOE), DARTnet, TWBNet (DARPA), 100,000 computers
- 90-92: NSFNET moves to 45 Mbps, 16 mid-level networks
- 94: NSF backbone dismantled, multiple private backbones;
  Introduction of Commercial Internet
- Today: backbones run at 10 Gbps, close to 200 millions computers in 150 countries

The ARPANet

Paul Baran
- RAND Corp, early 1960s
- Communications networks that would survive a major enemy attack
  ARPANet: Research vehicle for “Resource Sharing Computer Networks”
- 2 September 1969: UCLA first node on the ARPANet
- December 1969: 4 nodes connected by phone lines

ARPANet Evolves into Internet

TCP/IP NSFNet
Deregulation & Commercialization
WWW
ISP
Data Center
Web Hosting Multiple ISPs Internet2 Backbone Internet Exchanges
Application Hosting, Data Center
Parallel Backbones

vBNS Backbone Network Map
Qwest IP Backbone (Late 1999)

Digex Backbone

GTE Internetworking Backbone
**What’s the Internet**

Earlier we said . . . .
- data parcelled into packets
- each packet carries a destination address
- each packet is routed independently
- packets can arrive out of order
- packets may not arrive at all

However, the Internet provides two types of delivery services:
- connectionless (datagram, UDP, e.g., streaming media app, games)
- connection oriented (byte stream, TCP, e.g., web, email)

Connection oriented service provides:
- end-to-end reliability (sender retransmit lost packets)
- in-sequence delivery (receiver buffers incoming packets until it can deliver them in order)

How does a router know which router to forward a packet to?
How does a receiver know the correct ordering of packets?
How does a sender know which packet is lost and must be retransmitted?

**Protocols** — rules ("syntax" and "grammar") governing communication between nodes, e.g., TCP/IP
- *a "node" is one of sender, router, or receiver*

Protocols define the format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt

---

**Internet Protocol Stack**

- **application**: supporting network applications
  - HTTP, SMTP, FTP, etc.
- **transport**: endhost-endhost data transfer
  - TCP, UDP
- **network**: routing of datagrams from source to destination
  - IP, routing protocols
- **link**: data transfer between neighboring network elements
  - Ethernet, WiFi
- **physical**: bits “on the wire”

---

**Some Network Apps (and Their Protocols)**

- E-mail (SMTP)
- Web (HTTP)
- Instant messaging (IRC)
- Remote login (Telnet)
- P2P file sharing (Napster, Gnutella, KaZaa)
- Multi-user network games
- Streaming stored video clips (Adobe’s RTMP)
- Internet telephone (Skype)
- Real-time video conference (RTP)
- Massively parallel computing

---

**Layering in the IP Protocols**

```plaintext
Layering in the IP Protocols

HTTP  Telnet  FTP  DNS  RTP
   |
   V
Transmission Control Protocol (TCP)  User Datagram Protocol (UDP)
   |
   V
Internet Protocol
   |
   V
SONET  Ethernet  ATM
```
Why Layering?
Networks are complex! Many “pieces”:
- applications
- hosts
- routers
- links of various media

Dealing with complex systems:
- explicit structure allows identification of well-defined, specific parts of a large and complex system
- modularization eases maintenance, updating of system
- change of implementation of layer’s service transparent to rest of system
  - note that change of implementation is different from change of service definition!

Creating a Network Application

- Write programs that
  - run on different end systems and
  - communicate over a network.
  - e.g., Web browser software communicates with browser server

- No app software written for devices in network core
  - Network core devices do not function at app layer
  - This design allows for rapid app development

Application Architecture: Client-Server Computing
Email (SMTP) uses the client-server paradigm
server:
- a process that manages access to a resource
- usually has a permanent IP address
- waits for connection
- server farms for scaling
  - how do server farms maintain a single IP address externally?
client:
- a process that needs access to a resource
- initiates connection with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

Process vs. machine
Alternative(s) to client-server?
Sockets

• process sends/receives messages to/from its socket
• socket analogous to door
  - sending process shoves message out of door
  - sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process

Socket API Programming

What is a socket?
Review of sockets (assume familiarity from EECS 482)
(learn how to read man pages!)

Applications need Application Programming Interface (API) to use the network

API: set of function types, data structures, and constants
• allows programmer to learn once, write anywhere
• greatly simplifies job of application programmer

Addressing Socket

Server host may support many simultaneous application processes, each with one or more sockets

Web servers, for example, have different sockets for each connecting client

For a socket to receive messages, it must have an identifier

A host has a unique 32-bit IP address

Q: does the IP address of the host on which the process runs suffice for identifying the process?

Multiplexing/demultiplexing

Demultiplexing at rcv host:
- delivering received segments to correct socket

Multiplexing at send host:
- gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)
**How Demultiplexing Works**

Host receives IP packets
- each packet has source and destination IP addresses
- each packet carries one transport-layer segment
- each segment has source and destination port numbers

Host uses IP addresses & port numbers to direct segment to appropriate socket

**TCP/UDP segment format**

```
<table>
<thead>
<tr>
<th>source port #</th>
<th>dest port #</th>
</tr>
</thead>
<tbody>
<tr>
<td>other header fields</td>
<td></td>
</tr>
<tr>
<td>application data (message)</td>
<td></td>
</tr>
</tbody>
</table>
```

**Socket Addresses**

Somewhere in the socket structure:

- **bind()**
- **connect()**

TCP Server:
- IP address
- Port #
- InADDR_ANY
- Well-known
- client's address
- ephemeral
- server's address
- well-known

TCP Client:
- IP address
- Port #

**Connection-oriented Demux**

Socket identifier includes both the IP address and port numbers associated with the socket on the host

Example port numbers:
- HTTP server: 80
- Mail server: 25
- See /etc/services

TCP socket identified by 4-tuple:
- source IP address
- source port number
- dest IP address
- dest port number

Receiver kernel uses all four values to direct packet to appropriate socket

**Connection-oriented demux**

Client IP: A
- DP: 80
- S-IP: B
- D-IP: C

Server IP: C
- DP: 80
- S-IP: B
- D-IP: C

Say Port 9157
Not so “recent” news

Taiwan earthquake on Dec. 26, 2006
Damaged several undersea cables, disrupting telecommunication services in various parts of Asia.
Is Internet not redundant enough?

Traceroute from Taiwan to China goes through U.S.!

[michigan_1@planetlab1 michigan_1]$ traceroute scut1.6planetlab.edu.cn
traceroute to scut1.6planetlab.edu.cn (219.243.200.23), 30 hops max, 38 byte packets
1  140.112.107.254 (140.112.107.254)  0.459 ms  0.435 ms  0.371 ms
2  140.112.0.21 (140.112.0.21)  0.506 ms  0.630 ms  0.468 ms
3  140.112.0.1 (140.112.0.1)  0.621 ms  0.524 ms  0.478 ms
4  203.160.226.233 (203.160.226.233)  0.953 ms  0.839 ms  0.806 ms
5  tp.j160_1.twgate.net (203.160.227.176)  1.607 ms  1.431 ms  1.357 ms
MPLS Label=389977 CoS=3 TTL=1 S=0
6  ls_c124_1.twgate.net (203.160.228.246) 139.122 ms 143.088 ms 138.635 ms
MPLS Label=187 CoS=3 TTL=1 S=0
7  ge11.my_c76_2.twgate.net (203.160.228.190) 203.585 ms 203.288 ms 203.461 ms
8  12.18.94.65 (12.18.94.65)  203.806 ms  203.779 ms  204.823 ms
9  12.122.82.158 (12.122.82.158)  209.953 ms  210.320 ms  204.227 ms
MPLS Label=31209 CoS=3 TTL=1 S=0
10 ggr2-p3120.n54ny.ip.att.net (12.123.3.109) 203.499 ms 203.746 ms 205.360 ms
11 att-gw.nyc.dragon (192.205.32.58)  205.137 ms  206.690 ms  205.350 ms
12 * 217.239.41.10 (217.239.41.10)  450.638 ms *
13 ***
14 * 202.112.61.17 (202.112.61.17) 532.956 ms *
15 202.112.61.193 (202.112.61.193)  509.868 ms  528.030 ms *
16 202.112.53.181 (202.112.53.181)  529.697 ms  531.163 ms  526.445 ms
17 202.127.216.22 (202.127.216.22)  539.867 ms  562.172 ms
18 * 202.112.53.150 (202.112.53.150) 552.322 ms 552.684 ms
19 ***
20 ***
21 ***