Computer Network Fundamentals
Spring 2008

Week 1
Andreas Terzis
terzis@cs.jhu.edu
Outline

• Logistics
• A quick tour of networking
• Internet Architecture
  – Layering
  – Encapsulation
  – E2E Principle
Course Description

• Learn about the computer network principles and protocols
  – Understand design space and tradeoffs
  – Compare different protocols at the same level
    • Protocol examples: Ethernet, WiFi, IP, TCP, HTTP
  – Examples from alternative architectures
• Learn how to write code for networked applications
Text Books

- **Required**: *Computer Networks: A Systems Approach (4th ed.)* by Peterson, Davie

- **Recommended**: *UNIX Network Programming* by Richard Stevens
Course Times

- Lectures: TuTh 12:00-13:15 Hodson 210
- Office Hours: TuTh 11-12 WP 417 and by appt
- TA Section: Every second week
- TA Office Hours: TBA
Grading Policies

- Grading
  - Homework 5%, Projects 35%
  - Midterm 25%, Final 30%
  - Class Participation 5%
  - Different curves for 344/444

- NO LATE HOMEWORK OR PROJECTS ACCEPTED

- Project groups DO NOT change
Course Schedule

- W1 Introduction, Internet Architecture
- W2 Link Layer: Ethernet, PPP, IP
- W3 MAC
- W4 IP: Addressing, ARP/RARP, ICMP and examples
- W5,6 Intra-domain routing
- W7 Review, **Midterm**
Course Schedule (2)

- W8 Inter-domain routing, IP Mobility
- W9 Spring Break
- W10 Transport Layer
- W11,12 Congestion Control
- W13 QoS
- W14 Apps
- Final TBA
Course Projects

• Two implementation projects
  – Learn how to write networked applications (using the TCP/IP API)
  – Teams of 2-3
  – New this year
    • Mobile, social-network applications
      – Nokia N800 platform
Logistics

• Course website
  – http://hinrg.cs.jhu.edu/cs344s08/
  – Check every day
    • Everything goes there

• Course email list
  – cs344@hinrg.cs.jhu.edu
  – Subscribe through the website
Short Preview

• Very high-level overview of networking
  – Show how the World Wide Web works
  – Present some of the key ideas behind the Internet architecture
  • Large Scale
  • Dealing with failures
  • Co-operation/Competition
World Wide Web

- We use it every day
- Millions of websites
- Webpages contain objects
  - HTML, Text, images, etc
  - Base HTML file contains reference to other objects
  - Each object addressable by a URL

http://www.google.com/index.html
Questions

• How to express what we are looking for
• How to find what we are looking for
• How to transmit bits
• How to protect against errors
• How to direct bits towards the right destination
• How to share resources
Solution

• Divide and Conquer
  – Break the problem in smaller parts
  – Solve each part separately
  – Put the pieces back together

• Layering

• Protocol Stack
Physical Layer

• Send bits between directly connected machines
• Physical Media
  – Twisted Pair
  – Coaxial Cable
  – Fiber Optics
  – Radio Link
• Bandwidth
• Dedicated vs. shared link
Data Link Layer

• Data Link Layer functions
  – Framing
  – Error Detection & Correction
  – Link sharing

• Examples
  – Dialup, ADSL
  – Cable Modems
  – Wireless (WiFi)
Internet: Network of Networks
ISP Topology
Network Layer

• Networking layer functions
  – Transport packets from sending to receiving hosts
  – *Packet Switching vs. Circuit Switching*
Forwarding vs. Routing

- **Forwarding**
  - How to send packet to the next hop towards the destination

- **Routing**
  - How to compute the next hops

- **Routing algorithms**
  - *Distance Vector (RIP)*
  - *Link State (OSPF)*
Transport Layer

- Packets get lost
  - Physical errors
  - Equipment failures
  - Congestion
- Transport layer functions
  - Deliver packets reliably
  - Share resources among all network users
Reliable Transfer

- How to recover from losses
  - Introduce ACKs
  - Timers
  - Stop-and-Wait
    - Sender sends packet set timer
    - Receiver sends ACK
    - If no ACK received when timer expires sender retry

Sender

Receiver

Timer Expires

A:1

1

2

2
Reliable Transfer (2)

- **Stop-and-Wait** has low Utilization
- **Pipelining**
  - Senders allows multiple yet-to-be-acked packets (window)
- **Two variants**
  - **Go-Back-N**
  - **Selective repeat**
Congestion Control

- Congestion
  - Too many sources sending too much data too fast for network to handle
- Results
  - Lost packets
  - Long delays
- Solutions
  - End-to-End
  - Network assisted
Application Layer-HTTP

- So far: data reliably transferred end-to-end
- HTTP: hypertext transfer protocol
- client/server model
  - client: browser that requests, receives, “displays” Web objects
  - server: Web server sends objects in response to requests
Request/Response Timeline

We enter URL www.google.com/index.html (contains text, reference to 1 jpeg image)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.google.com

1b. HTTP server at host www.google.com waiting for TCP connection. “accepts” connection, notifying client

2. HTTP client sends HTTP request message (containing URL). Message indicates that client wants index.html

3. HTTP server receives request message, forms response message containing requested object, and sends response
5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 1 referenced jpeg object

6. Steps 1-5 repeated for jpeg object
HTTP Request Message

HTTP request message:
- ASCII (human-readable format)

```
GET /index.html HTTP/1.1
Host: www.google.com
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

(extra carriage return, line feed)

Carriage return, line feed indicates end of message
HTTP Response Message

- **status line** (protocol status code status phrase)
  - `HTTP/1.1 200 OK`
  - `Connection close`
  - `Date: Tue, 04 Mar 2003 07:28:29 GMT`
  - `Server: GWS/2.0`
  - `Content-Length: 2824`
  - `Content-Type: text/html`

- **header lines**

- **data, e.g., requested HTML file**

  `data data data data data data data data ...

CS344/Spring08 29`
Internet Architecture
The Problem

- Many different applications and underlying transmission technologies
- Do we re-implement every application for every technology?
- Obviously not, but how does the Internet architecture avoid this?
What is Architecture?

• The assignment of tasks and knowledge
  – Who does what, and where is the state kept

• Architecture is not the implementation itself
  – Architecture is how to “organize” implementations
    • What interfaces are supported
    • Where functionality is implemented

• Architecture is the \textit{modular design} of the network
Internet Architecture

- Routers do routing, and almost nothing else
  - No application-specific functions
- Hosts do all application-specific processing
- This allowed wide variety of applications to flourish on Internet
Software Modularity

Break system into modules:

• Well-defined interfaces gives flexibility
  – Can change implementation of modules
  – Can extend functionality of system by adding new modules

• Interfaces hide information
  – Allows for flexibility
  – But can hurt performance
Network Modularity

Like software modularity, but with a twist:
• Implementation distributed across routers and hosts
• Must decide both:
  – How to break system into modules
  – Where modules are implemented
• Lecture will address these questions in turn
Layering

- Layering is a particular form of modularization
- The system is broken into a vertical hierarchy of logically distinct entities (layers)
- The service provided by one layer is based solely on the service provided by layer below
- Rigid structure: easy reuse, performance suffers
ISO OSI Reference Model for Layers

- Application
- (Presentation)
- (Session)
- Transport
- Network
- Datalink
- Physical
Layering Solves Problems

- Application layer doesn’t know about anything below the presentation layer, etc.
- Information about network is hidden from higher layers
- This ensures that we only need to implement an application once!
OSI Model Concepts

- **Service**: *what* a layer does
- **Service interface**: *how* to access the service
  - Interface for layer above
- **Peer interface (protocol)**: *how* peers communicate
  - A set of rules and formats that govern the communication between two network boxes
  - Protocol does not govern the implementation on a single machine, but how the layer is implemented between machines
Next Five (Seven) Slides

• Will summarize each layer

• A good time for a nap....
Physical Layer (1)

- **Service**: move information between two systems connected by a physical link
- **Interface**: specifies how to send a bit
- **Protocol**: coding scheme used to represent a bit, voltage levels, duration of a bit
- **Examples**: coaxial cable, optical fiber links; transmitters, receivers
Datalink Layer (2)

- **Service:**
  - framing (attach frame separators)
  - send data frames between peers
  - others:
    - arbitrate the access to common physical media
    - per-hop reliable transmission
    - per-hop flow control
- **Interface:** send a data unit (packet) to a machine connected to the same physical media
- **Protocol:** layer addresses, implement Medium Access Control (MAC) (e.g., CSMA/CD)...
Network Layer (3)

- **Service:**
  - deliver a packet to specified network destination
  - perform segmentation/reassemble
  - others:
    - packet scheduling
    - buffer management

- **Interface:** send a packet to a specified destination

- **Protocol:** define global unique addresses; construct routing tables
Transport Layer (4)

- **Service:**
  - demultiplexing
  - optional: error-free and flow-controlled delivery

- **Interface:** send message to specific destination process

- **Protocol:** implements reliability and flow control

- **Examples:** TCP and UDP
Application Layer (7)

- **Service**: any service provided to the end user
- **Interface**: depends on the application
- **Protocol**: depends on the application
- **Examples**: FTP, Telnet, WWW browser
Who Does What?

- Five layers
  - Lower three layers are implemented everywhere
  - Next two layers are implemented only at hosts
Logical Communication

- Layers interacts with corresponding layer on peer

![Logical Communication Diagram]
Physical Communication

- Communication goes down to physical network, then to peer, then up to relevant layer
Encapsulation

- A layer can use **only** the service provided by the layer immediate below it
- Each layer may change and add a header to data packet
Example: Postal System

Standard process (historical):
• Write letter
• Drop an addressed letter off in your local mailbox
• Postal service delivers to address
• Addressee reads letter (and perhaps responds)
Postal Service as Layered System

Layers:
- Letter writing/reading
- Delivery

Information Hiding:
- Network need not know letter contents
- Customer need not know how the postal network works

Encapsulation:
- Envelope
Multiple Instantiations

- Can have several instantiations for each layer
  - many applications
  - many network technologies
  - transport can be reliable (TCP) or not (UDP)
- Applications dictate transport
  - In general, higher layers can dictate lower layer
- But this is a disaster!
  - applications that can only run certain networks
Multiple Instantiations of Layers
A universal Internet layer:

- Internet has only IP at the Internet layer
- Many options for modules above IP
- Many options for modules below IP
Hourglass

e-mail WWW phone ...
SMTP HTTP RTP ...
TCP UDP ...
IP
ethernet PPP ...
CSMA async sonet ...
copper fiber radio ...

CS344/Spring08
Implications of Hourglass

- A single Internet layer module:
- Allows all networks to interoperate
  - all networks technologies that support IP can exchange packets
  - Allows all applications to function on all networks
  - all applications that can run on IP can use any network
- Simultaneous developments above and below IP
Placing Functionality

- The most influential paper about placing functionality is “End-to-End Arguments in System Design” by Saltzer, Reed, and Clark.
- The “Sacred Text” of the Internet:
  - endless disputes about what it means
  - everyone cites it as supporting their position
Example: Reliable File Transfer

- Solution 1: make each step reliable, and then concatenate them
- Solution 2: end-to-end check and retry
Example (cont’d)

- Solution 1 not complete
  - What happens if any network element misbehaves?
  - The receiver has to do the check anyway!

- Solution 2 is complete
  - Full functionality can be entirely implemented at application layer with no need for reliability from lower layers

- Is there any need to implement reliability at lower layers?
Conclusion

- Implementing this functionality in the network:
  - Doesn’t reduce host implementation complexity
  - Does increase network complexity
  - Probably imposes delay and overhead on all applications, even if they don’t need functionality
  - However, implementing in network can enhance performance in some cases
    - very lossy link
Conservative Interpretation

- “Don’t implement a function at the lower levels of the system unless it can be completely implemented at this level” (Peterson and Davie)

- Unless you can relieve the burden from hosts, then don’t bother
Extended Version of E2E Argument

- Don’t put application semantics in network
  - Leads to loss of flexibility
  - Cannot change old applications easily
  - Cannot introduce new applications easily

- Normal E2E argument: performance issue
  - Introducing more functionality imposes more overhead
  - Subtle issue, many tough calls (e.g., multicast)

- Extended version:
  - Short-term performance vs long-term flexibility
Summary

- Layering is a good way to organize networks
- Unified Internet layer decouples apps from networks
- E2E argument encourages us to keep IP simple
- Commercial realities threaten to undo all of this...