CSE 534
Fundamentals of Networking
Intro + Undergrad Overview

Vyas Sekar
Stony Brook University

Based in part on content borrowed from:
Aditya Akella, Samir Das, Nick McKeown, Don Porter, Scott Shenker, Ion Stoica
Introduction

• Me = Vyas Sekar

• Best way to reach me – vyas@cs
  – (Email to setup time before coming to my office Rm 2424)

• Tentative Office Hours: Mon 5:30 – 6:30

• TA: TBD

• Course webpage
  http://www.cs.stonybrook.edu/~vyas/teaching/CSE_534/Spring13/

Keep checking for updates!
Class Expectations

• Not a theory class
  (don’t get mislead by “Fundamentals”)

• Expect to write code
  – All *three projects* will involve writing code
  – All *three projects* involve use real n/w tools

• We assume you have UG
  networks/systems background
Goals

• How does the Internet work?
  – Architecture, protocols, systems

• Get familiar with current Internet research
  – Critically read research papers

• Where is it headed?

• Understand solutions in context
  – Goals (e.g., performance, security, scalability)
  – Assumptions (e.g., technology, economics)
Grading

• 25% Midterm

• 25% Final

• 30% Projects: Three projects
  – To be done in groups of 3

• 10% Discussion/participation
  – Expect pop quizzes

• 10% Homeworks/paper summaries
  – To be done alone
Logistics

• How to submit paper reviews
  https://docs.google.com/spreadsheet/viewform?formkey=dGpLaEdOdHA2azJ3OUlKWVv2dEFNRXc6MQ

• Class discussion
  https://piazza.com/stonybrook/spring2013/cse534/home

• Project submissions (TBD)

• All links from course homepage
Waitlist

• Wait for 2-3 classes to see what happens

• Do well on pop quiz 😊
Outline

• Preliminaries

• Undergrad overview

• How to read a paper

• Questions
Key concepts

• Layering

• Packet vs. Circuit switching

• Basics of Layer2

• Basics of routing protocols

• Brief history of the Internet
Network Communication:
Lots of Functions Needed

- Links
- Multiplexing
- Routing
- Addressing/naming (locating peers)
- Reliability
- Flow control
- Fragmentation

How do you implement these functions?
Key: Layering and protocols
At each layer, numerous protocols are available. At the lowest level, where interfaces exchange data, the protocol in use is predetermined. A driver for that protocol is associated with the interface, and all data that comes in on the interface is assumed to follow the protocol (i.e., Ethernet); if it doesn't, errors are reported and no communication takes place.

But once the driver has to hand over data to a higher layer, a choice of protocols ensues. Should data at L3 be handled by IPv4, IPv6, IPX (the Novell NetWare protocol), DECnet, or some other network-layer protocol? And a similar choice must be made going from L3 to L4, where TCP, UDP, ICMP, and other protocols reside.

This chapter deals with the lower three layers and briefly touches on the fourth one. An individual package of transmitted data is commonly called a **frame** on the link layer, L2; a **packet** on the network layer; a **segment** on the transport layer; and a **message** on the application layer.

The layers are often called the **network stack**, because communication travels down the layers until it is physically transmitted across the wire (or wireless bands) and then travels back up. Headers are also added and removed in a LIFO manner.

The Big Picture

Figure 13-2 builds on the TCP/IP model in Figure 13-1. Figure 13-2 shows which chapter covers each interface between adjacent layers. Some of these interfaces involve communication down the stack, whereas others involve communication upward:

- **Going up in the stack (for receiving a message)**
  - This chapter describes how ingress traffic is handed to the right protocol handler. (The meaning of **ptype_base** and **ptype_all** will become clear in the section “Protocol Handler Organization.”)
What is Layering?

• A way to deal with complexity
  – Add multiple levels of abstraction
  – Each level encapsulates some key functionality
  – And exports an interface to other components

• Layering: Modular approach to implementing network functionality by introducing abstractions

• Challenge: how to come up with the “right” abstractions?
Power of Layering

- Solution: Intermediate layer that provides a single abstraction for various network technologies
  - $O(1)$ work to add app/media
  - variation on “add another level of indirection”
Layering vs Not Layering

- Layer N may duplicate layer N-1 functionality
  - E.g., error recovery

- Layers may need same info (timestamp, MTU)

- Strict adherence to layering may hurt performance

- Some layers are not always cleanly separated
  - Inter-layer dependencies in implementations for performance reasons
  - Many cross-layer assumptions, e.g., buffer management

- Layer interfaces are not really standardized.
  - It would be hard to mix and match layers from independent implementations, e.g., windows network apps on unix (w/o compatibility library)
Key concepts

• Layering

• Packet vs. Circuit switching

• Basics of Layer2

• Basics of routing protocols

• Brief history of the Internet
Circuit Switching

- Source first establishes a circuit to destination
  - Switches along the way stores info about connection
    - Possibly allocate resources
    - Different srs-dst’s get different paths

- Source sends the data over the circuit
  - No address required since path is established beforehand

- The connection is explicitly set up and torn down

- Switches use TDM (digital) or FDM (analog) to transmit data from various circuits
Circuit Switching Discussion

• Positives
  – Fast and simple data transfer, once the circuit has been established
  – Predictable performance since the circuit provides isolation from other users
    • E.g. guaranteed max bandwidth

• Negatives
  – How about bursty traffic
    • Circuit will be idle for significant periods of time
    • Also, can’t send more than max rate
  – Circuit set-up/tear down is expensive
  – Also, reconfiguration is slow
    • Fast becoming a non-issue
Packet Switching

• Packet-switching: Benefits
  – Ability to exploit statistical multiplexing
  – More efficient bandwidth usage

• Packet switching: Concerns
  – Needs to buffer and deal with congestion:
  – More complex switches
  – Harder to provide good network services (e.g., delay and bandwidth guarantees)
Key concepts

• Layering

• Packet vs. Circuit switching

• Basics of Layer2

• Basics of routing protocols

• Brief history of the Internet
Ethernet History

• Originally designed for a shared wire (e.g., coax cable)
• Each device listens to all traffic
  – Hardware filters out traffic intended for other hosts
    • I.e., different destination MAC address
  – Can be put in “promiscuous” mode, and record everything (called a network sniffer)
• Sending: Device hardware automatically detects if another device is sending at same time
  – Random back-off and retry
Early competition

• Token-ring network: Devices passed a “token” around
  – Device with the token could send; all others listened
• Send latencies increased proportionally to the number of hosts on the network
  – Even if they weren’t sending anything (still have to pass the token)
• Ethernet has better latency under low contention and better throughput under high
Token ring

Source: http://www.datacottage.com/nch/troperation.htm
**Shared vs Switched**

**Shared Ethernet**: 1 collision domain for multiple nodes. The possibility of collisions. Non-deterministic

**Switched Full Duplex Ethernet**: 1 collision domain per node. Use of switch. No possibility of collisions. Deterministic.

Key concepts

• Layering

• Packet vs. Circuit switching

• Basics of Layer2

• Basics of routing protocols

• Brief history of the Internet
Distance-Vector Routing

• Idea
  – At any time, have cost/next hop of best known path to destination
  – Use cost $\infty$ when no path known

• Initially
  – Only have entries for directly connected nodes

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>$\infty$</td>
<td>–</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>–</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>F</td>
</tr>
</tbody>
</table>

Diagram of network with costs:

- A to B: 4
- A to C: $\infty$
- A to D: $\infty$
- A to E: 2
- A to F: 6
- B to C: 1
- B to D: 3
- C to D: 1
- E to F: 3
- F to C: 1
- F to D: 1
Distance-Vector Update

- **Update**(x,y,z)
  
  \[ d \leftarrow c(x,z) + d(z,y) \]  
  
  \# Cost of path from x to y with first hop z

  if \( d < d(x,y) \)
  
  \# Found better path
  
  return \( d,z \)  
  
  \# Updated cost / next hop

  else
  
  return \( d(x,y), \text{nexthop}(x,y) \)  
  
  \# Existing cost / next hop
Link State Protocol Concept

• Every node gets complete copy of graph
  – Every node “floods” network with data about its outgoing links

• Every node computes routes to every other node
  – Using single-source, shortest-path algorithm

• Process performed whenever needed
  – When connections die / reappear
Sending Link States by Flooding

- **X Wants to Send Information**
  - Sends on all outgoing links

- **When Node B Receives Information from A**
  - Send on all links other than A
Comparison of LS and DV Algorithms

Message complexity
• **LS**: with n nodes, E links, O(nE) messages
• **DV**: exchange between neighbors only O(E)

Speed of Convergence
• **LS**: Complex computation
  – But...can forward before computation
  – may have oscillations
• **DV**: convergence time varies
  – may be routing loops
  – count-to-infinity problem
  – (faster with triggered updates)

Space requirements:
– LS maintains entire topology
– DV maintains only neighbor state
“Flat” routing not suited for the Internet
- Doesn’t scale with network size
  - Storage \( \rightarrow \) Each node cannot be expected to store routes to every destination (or destination network)
  - Convergence times increase
  - Communication \( \rightarrow \) Total message count increases
- Administrative autonomy
  - Each internetwork may want to run its network independently
    - E.g hide topology information from competitors

Solution: Hierarchy via autonomous systems
What is an Autonomous System (AS)?

- A set of routers under a single technical administration
  - Use an *interior gateway protocol (IGP)* and common metrics to route packets within the AS
  - Connect to other ASes using *gateway routers*
  - Use an *exterior gateway protocol (EGP)* to route packets to other AS’s
- IGP: OSPF, RIP (last class)
- Today’s EGP: BGP version 4
- Similar to an “inter-network”
  - Could also be a group of internetworks owned by a single commercial entity
An example

Intra-AS routing algorithm + Inter-AS routing algorithm → Forwarding table
Key concepts

• Layering

• Packet vs. Circuit switching

• Basics of Layer1/Layer2

• Basics of routing protocols

• Brief history of the Internet
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

BBN team that implemented the interface message processor
History of the Internet

- 70’s: started as a research project, 56 kbps, < 100 computers
- 80-83: 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, NSI (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
- Today: backbones run at >100 Gbps, >700 millions computers in >200 countries
ARPANET GEOGRAPHIC MAP, OCTOBER 1980

---

SATELLITE CIRCUIT

- IMP
- TIP
- PLURIBUS IMP
- PLURIBUS TIP
- C30

(Note: This map does not show ARPA's experimental satellite connections. Names shown are IMP names, not necessarily host names.)
Outline

• Preliminaries

• Undergrad overview

• *How to read a paper*

• Questions
Quick survey?

• How many of you have read a research paper?

• How many read the Keshav paper posted?
Three-pass approach

- **Pass1**: The skim
  - Read abstract, intro, conclusions, section titles

- **Pass2**: Grasp technical content
  - Main thrust + supporting evidence

- **Pass3**: Virtually re-implement the paper
  - Challenge/Question every assumption
Five Cs to summarize the paper

• **Category:**
  What type of paper is this? A measurement paper? An analysis of an existing system? A description of a research prototype?

• **Context:**
  Which other papers is it related to? Which theoretical bases were used to analyze the problem?

• **Correctness:**
  Do the assumptions appear to be valid?

• **Contributions:**
  What are the paper’s main contributions?

• **Clarity:**
  Is the paper well written?
Writing your review

• Did you like this paper?

• What problem is this paper solving?
  – 2-3 sentences.

• What are the strengths of this paper?
  – 3-4 sentences.

• What are the main weaknesses in the paper?
  – 3-4 sentences.

• What would you do differently?
  Are there assumptions you disagree with?
  Do you see ideas for future work or improving the solution?
  – 5-6 sentences
Outline

• Preliminaries

• Undergrad overview

• How to read a paper

• Questions/Discussion
Why is networking cool?

• The Internet is on every top-100 inventions list

• Real artifact
  – Can measure/build things
  – Can impact the real world

• Interdisciplinary: Systems, Theory, Economics, Security..

High impact!
  – Many of the most cited papers in CS are in networking
  – We will read many of them!
Why is networking uncool?

• Identity crisis: What is networking?
  – Protocol formats?
  – Boxes in the network?
  – Theory of routing/queueing?

• Is it a “discipline”?
  – What are the open problems in this field?

• Still relatively young → “Problem domain”
What makes networking challenging?

• Asynchronous (speed of light)

• Things fail

• Diversity

• Need consensus across principals/domains

• Growth, unforseen apps
Example of Layering

- Software and hardware for communication between two hosts

- Advantages:
  - Simplifies design and implementation
  - Easy to modify/evolve
Switching in the Telephone Network