Final Lecture: Course Overview

Acknowledgments: Lecture slides are from Computer networks course thought by Jennifer Rexford at Princeton University. When slides are obtained from other sources, a reference will be noted on the bottom of that slide and full reference details on the last slide.
Goals of Today’s Class

• What are the key concepts in networking?
  – Hierarchy, indirection, caching, randomization
  – Soft state, layering, (de)multiplexing, e2e argument

• Why was there no math in this course?
  – Is theory even useful in data networking?
  – Control theory, graph theory, game theory, optimization theory, queuing theory, scheduling theory, …

• What’s going to happen to the Internet?
Key Concepts in Networking
Networking Has Some Key Concepts

• Course was organized around protocols
  – But a small set of concepts recur in many protocols

• Many of these are general CS concepts
  – Hierarchy
  – Indirection
  – Caching
  – Randomization

• Others are somewhat networking-specific
  – Soft state
  – Layering
  – (De)multiplexing
  – End-to-end argument
Hierarchy

• Scalability of large systems
  – Cannot store all information everywhere
  – Cannot centrally coordinate each component

• Hierarchy as a way to manage scale
  – Divide large system into smaller pieces
  – And manage the pieces separately

• Hierarchy as a way to divide control
  – Decentralized management of common infrastructure

• Examples of hierarchy in the Internet
  – Example #1: IP address blocks
  – Example #2: routing protocols
  – Example #3: Domain Name System (DNS)
  – Example #4: super-peers in P2P systems
Hierarchy: IP Address Blocks

- Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN

1.2.3.4  1.2.3.7  1.2.3.156  5.6.7.8  5.6.7.9  5.6.7.212

LAN 1  LAN 2

Forwarding table
Hierarchy: IP Address Blocks

• Separation of control
  – Prefix: assigned to an institution
  – Addresses: assigned by the institution to their nodes

• Who assigns prefixes?
  – Internet Corporation for Assigned Names and Numbers
    • Allocates large address blocks to Regional Internet Registries
  – Regional Internet Registries (RIRs)
    • E.g., ARIN (American Registry for Internet Numbers)
    • Allocates address blocks within their regions
    • Allocated to Internet Service Providers and large institutions
  – Internet Service Providers (ISPs)
    • Allocate address blocks to their customers
    • Who may, in turn, allocate to their customers…
Hierarchy: Routing Protocols

- **AS-level topology**
  - Nodes are Autonomous Systems (ASes)
  - Edges are links and business relationships
  - Hides the detail within each AS’s network
Hierarchy: Routing Protocols

• Interdomain routing ignores details within an AS
  – Routers flood information to learn the topology
  – Routers determine “next hop” to reach other routers…
  – By computing shortest paths based on the link weights
Hierarchy: Routing Protocols

• Scaling challenges within an AS
  – Flooding link-state packets throughout the network
  – Running Dijkstra’s shortest-path algorithm

• Introduce hierarchy through “areas”
Hierarchy: Domain Name System

- Labeled A through M

- A Verisign, Dulles, VA
- C Cogent, Herndon, VA (also Los Angeles)
- D U Maryland College Park, MD
- G US DoD Vienna, VA
- H ARL Aberdeen, MD
- J Verisign, (11 locations)
- K RIPE London (also Amsterdam, Frankfurt)
- I Autonomica, Stockholm (plus 3 other locations)
- B USC-ISI Marina del Rey, CA
- F Internet Software C. Palo Alto, CA (and 17 other locations)
- E NASA Mt View, CA
- m WIDE Tokyo
- L ICANN Los Angeles, CA
Hierarchy: Domain Name System

- **com**
- **edu**
- **org**
- **ac**
- **uk**
- **zw**

**Unnamed root**

**Generic domains**
- **foo**
  - **west**
  - **east**
    - **foo**
    - **my**

**Country domains**
- **ac**
  - **cam**
  - **usr**

**Examples:**
- my.east.foo.edu
- usr.cam.ac.uk
Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
Hierarchy: Super Peers in KaZaA

• Each peer is either a group leader or assigned to a group leader
  – TCP connection between peer and its group leader
  – TCP connections between some pairs of group leaders

• Group leader tracks the content in all its children
Indirection

• Referencing by name
  – Rather than the value itself
  – E.g., manipulating a variable through a pointer

• Benefits of indirection
  – Human convenience
  – Reducing overhead when things change

• Examples of indirection in the Internet
  – Example #1: host names instead of IP addresses
  – Example #2: mobile IP
Indirection: Host Names vs. Addresses

• Host names
  – Mnemonic name appreciated by humans
  – Variable length, alpha-numeric characters
  – Provide little (if any) information about location

• IP addresses
  – Numerical address appreciated by routers
  – Fixed length, binary number
  – Hierarchical, related to host location
  – Examples: 64.236.16.20 and 193.30.227.161
Indirection: Host Names vs. Addresses

- Names are easier to remember
  - www.cnn.com vs. 64.236.16.20

- Addresses can change underneath
  - Move www.cnn.com to 173.15.201.39
  - E.g., renumbering when changing providers

- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site

- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content

- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu
Indirection: Mobile IP

• Seamless transmission to a mobile host
Indirection: Mobile IP

1. Correspondent addresses packets using home address of mobile.
2. Home agent intercepts packets, forwards to foreign agent.
3. Foreign agent receives packets, forwards to mobile.
4. Mobile replies directly to correspondent.

Networks:
- Home network
- Wide area network
- Visited network
Caching

• Duplicating data stored elsewhere
  – To reduce latency for accessing the data
  – To reduce resources consumed

• Caching is often quite effective
  – Speed difference between cache and primary copy
  – Locality of reference, and small set of popular data

• Examples from the Internet
  – Example #1: DNS caching
  – Example #2: Web caching
Caching: DNS Caching

• What is cached?
  – Mapping of names to IP addresses
  – IP addresses for DNS servers (e.g., for .com)
  – DNS queries that failed (e.g., www.cnn.com)

• Why it reduces latency?
  – DNS queries can take a long time (e.g., 1 second)
  – Local DNS server is typically very close to the users

• Why is the cache hit rate is very high?
  – Cached information remains valid for awhile
  – Popular sites (e.g., www.cnn.com) are visited often
  – The cache is shared among a group of users
Caching: Web Caching

• What is cached?
  – Web object, like an HTML file or embedded image

• Where is it cached?
  – Browser cache, proxy cache, main-memory on server

• Why it reduces latency?
  – Avoids fetching across the network (or the disk)
  – Reduces load on the network and the server

• What helps increase the hit rate?
  – Cacheable content (not dynamically generated)
  – Sharing of the cache among multiple users
  – Small amount of very popular content
Randomization

• **Distributed adaptive algorithms**
  – Multiple distributed parties
  – Adapting to network conditions independently

• **Risk of synchronization**
  – Many parties reacting at the same time
  – Leading to bad aggregate behavior

• **Randomization can desynchronize**
  – Example #1: Ethernet back-off mechanism
  – Example #2: Random Early Detection

• **Rather than imposing centralized control**
Randomization: Ways to Share Media

• **Channel partitioning MAC protocols:**
  – Share channel efficiently and fairly at high load
  – Inefficient at low load: delay in channel access, \( \frac{1}{N} \) bandwidth allocated even if only 1 active node!

• **“Taking turns” protocols**
  – Eliminates empty slots without causing collisions
  – Vulnerable to failures (e.g., failed node or lost token)

• **Random access MAC protocols**
  – Efficient at low load: single node can fully utilize channel
  – High load: collision overhead
Randomization: Ethernet Back-off

- Random access: exponential back-off
  - After collision, wait random time before retrying
  - After $m^{th}$, choose $K$ randomly from $\{0, \ldots, 2^m - 1\}$
  - Wait for $K \times 512$ bit times before trying again
Randomization: Dropping Packets Early

• Congestion on a link
  – Eventually the queue becomes full
  – And new packets must be dropped

• Drop-tail queuing leads to bursty loss
  – Many packets encounter a full queue
  – Many TCP senders reduce their sending rates
Randomization: Dropping Packets Early

• Better to give early feedback
  – Get a few connections to slow down
  – … before it is too late

• Random Early Detection (RED)
  – Randomly drop packets when queue (near) full
  – Drop rate increases as function of queue length

![Diagram showing the relationship between probability and average queue length. The probability increases as the average queue length increases.](image-url)
Networking Has Some Key Concepts

- Course was organized around protocols
  - But a small set of concepts recur in many protocols
- Many of these are general CS concepts
  - Hierarchy
  - Indirection
  - Caching
  - Randomization
- Others are somewhat networking-specific
  - Soft state
  - Layering
  - (De)multiplexing
  - End-to-end argument
Soft State

• State: stored in nodes by network protocols
  – Installed by receiver of a set-up message
  – Updated when network conditions change

• Hard state: valid unless told otherwise
  – Removed by receiver of a tear-down message
  – Requires error handling to deal with sender failure

• Soft state: invalid if not told to refresh
  – Removed by receiver via a timeout
  – Periodically refreshed as needed

• Soft state reduces complexity
  – Example #1: DNS caching
  – Example #2: DHCP leases
Soft State: DNS Caching

Application

DNS resolver

DNS query

DNS response

DNS cache

Local DNS server

1

2

3

4

5

6

7

8

9

10

Root server

Top-level domain server

Second-level domain server
Soft State: DNS Caching

- Cache consistency is a hard problem
  - Ensuring the cached copy is not out of date

- Strawman: explicit revocation or updates
  - Keep track of everyone who has cached information
  - If name-to-host mapping changes, update the caches
  - If you fail to reach a cache, keep trying till success

- Soft state solution
  - DNS responses include a “time to live” (TTL) field
  - Cached entry is deleted after TTL expires
Soft State: DHCP Bootstrapping

Dynamic Host Configuration Protocol
Soft State: DHCP Leases

- DHCP “offer message” from the server
  - Configuration parameters (proposed IP address, mask, gateway router, DNS server, ...)
  - Lease time (the time information remains valid)

- Why is a lease time necessary?
  - Client can release address (DHCP RELEASE)
    - E.g., “ipconfig /release” at the DOS prompt
    - E.g., clean shutdown of the computer
  - But, the host might not release the address
    - E.g., the host crashes (blue screen of death!)
    - E.g., buggy client software
  - You don’t want address to be allocated forever
Layering: A Modular Approach

• Sub-divide the problem
  – Each layer relies on services from layer below
  – Each layer exports services to layer above

• Interface between layers defines interaction
  – Hides implementation details
  – Layers can change without disturbing other layers

| Application | Application-to-application channels | Host-to-host connectivity | Link hardware |
Layering: Standing on Shoulders

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>HTTP</td>
</tr>
<tr>
<td></td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>IP</td>
</tr>
<tr>
<td></td>
<td>Ethernet interface</td>
</tr>
<tr>
<td>Router</td>
<td>IP</td>
</tr>
<tr>
<td></td>
<td>Ethernet interface</td>
</tr>
<tr>
<td></td>
<td>SONET interface</td>
</tr>
<tr>
<td>Router</td>
<td>IP</td>
</tr>
<tr>
<td></td>
<td>Ethernet interface</td>
</tr>
<tr>
<td></td>
<td>SONET interface</td>
</tr>
<tr>
<td>Host</td>
<td>HTTP</td>
</tr>
<tr>
<td></td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>IP</td>
</tr>
<tr>
<td></td>
<td>Ethernet interface</td>
</tr>
<tr>
<td></td>
<td>SONET interface</td>
</tr>
</tbody>
</table>

HTTP message
TCP segment
IP packet
Ethernet frame
SONET frame
Layering: Internet Protocol Suite

The waist facilitates interoperability
Layering: Encapsulation of Data

• Different devices switch different things
  – Physical layer: electrical signals (repeaters and hubs)
  – Link layer: frames (bridges and switches)
  – Network layer: packets (routers)

Application gateway

Transport gateway

Router

Bridge, switch

Repeater, hub

Frame header Packet header TCP header User data
Demultiplexing

• Separating multiple streams out of one
  – Recognizing the separate streams
  – Treating the separate streams accordingly

• Examples in the Internet

![Diagram showing frame header, packet header, TCP header, user data, type, port #, protocol]
(De)multiplexing: With a NAT

10.0.0.1

NAT

138.76.29.7

outside

10.0.0.2

inside
Power at the End Host

**End-to-End Principle**
Whenever possible, communications protocol operations should be defined to occur at the end-points of a communications system.

**Programmability**
With programmable end hosts, new network services can be added at any time, by anyone.
Telephone Network

Smart Network

Dumb Terminals
Internet

Dumb Network

Smart Terminals
Why No Math in This Course?

• Hypothesis #1: theory not relevant to the Internet
  – Body of mathematics created for telephone networks
  – Teletraffic modeling, computing call blocking statistics
  – Many of these models don’t work in data networks

• Hypothesis #2: too many kinds of math to cover
  – Queuing theory: statistical multiplexing works
  – Control theory: TCP congestion control works
  – Optimization theory: TCP maximizes aggregate utility
  – Game theory: reasoning about competing ASes
  – We still need to build a much stronger foundation…
What Will Happen to the Internet
No Strict Notions of Identity

- Leads to
  - Spam
  - Spoofing
  - Denial-of-service
  - Route hijacking
Protocols Designed Based on Trust

• That you don’t spoof your addresses
  – MAC spoofing, IP address spoofing, spam, …

• That port numbers correspond to applications
  – Rather than being arbitrary, meaningless numbers

• That you adhere to the protocol
  – Ethernet exponential back-off after a collision
  – TCP additive increase, multiplicative decrease

• That protocol specifications are public
  – So others can build interoperable implementations
Nobody in Charge

• Traffic traverses many Autonomous Systems
  – Who’s fault is it when things go wrong?
  – How do you upgrade functionality?

• Implicit trust in the end host
  – What if some hosts violate congestion control?

• Anyone can add any application
  – Whether or not it is legal, moral, good, etc.

• Nobody knows how big the Internet is
  – No global registry of the topology

• Spans many countries
  – So no government can be in charge
Challenging New Requirements

• Disseminating data
  – Rather than communicating between two machines

• Mobile hosts
  – Rather than machines at a fixed location

• Sometimes-connected hosts
  – Developing regions with intermittent connectivity

• Large number of hosts
  – Billions of small networked devices (e.g., sensors)

• Real-time applications
  – Voice over IP, gaming, and IPTV
The Internet of the Future

• Can we fix what ails the Internet
  – Security
  – Performance
  – Upgradability
  – Managability
  – <your favorite gripe here>

• Without eliminating the good things
  – Ease of adding new hosts
  – Ease of adding new services
  – Ease of adding new link technologies

• An open technical and policy question…
Thank You!