Overview and Introduction to Networking

COS 316 Lecture 1

Amit Levy

Course Overview

- Instructors: Amit Levy (me!) & Alan Kaplan
- Staff: Danny Chen, Ashwini Raina, Mary Hogan, David Liu, Will Sweeny
- Lectures: M/W 1:30-2:20, Precepts: Thursdays
  - Guest lectures from: Prof. Rexford, Prof. Freedman, Prof. Jamieson, Prof. Lloyd
- Goals:
  1. Learn common principles of system design
  2. A taste of systems fields: networking, distributed systems, security, operating systems.
  3. Learn practical skills: Go, git, socket programming, concurrent programming...
  4. Build a meaningfully large system: a web framework

What is a System?

What are examples of systems?

What are examples of systems?

- Operating system kernel
- The Internet
- Database
- Distributed file system
- Web framework
- Game engine
- Web browser
- Sensor network

What makes something a *System*?

What makes something a *System*?

- Provides an interface to underlying resources
- Mediates access to shared resources
- Isolates applications
- Abstracts complexity
- Abstracts differences in implementation

Why is this challanging?

Why is this challanging?

- Correctness
- Performance
- Security
- How general should an interface be?
- How portable should an interface be?

Introduction to Networking

The simplest network

Figure 1: A two node network

A Multi-Hop Network

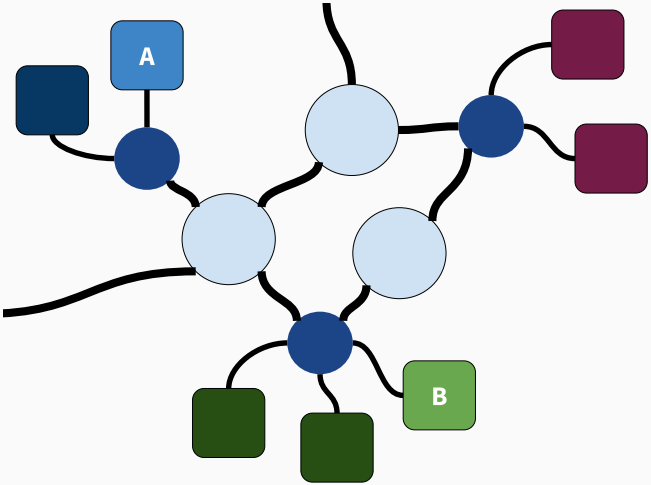

Figure 2: How can we get data from A to B?

Option 1: Circuit Switched Networking

Figure 3: Bell Systems Switchboard [2]

Option 2: Packet Switched Networking

- Data between nodes broken into "packets"
- Routes must only be consistent on a per-packet basis
- Packets include a header with the destination address (e.g. an IP address)
- Internal nodes (routers, switches) forward packets to other nodes closer to the destination.

This is what the Internet uses

How can we get data from A to B *reliably*?

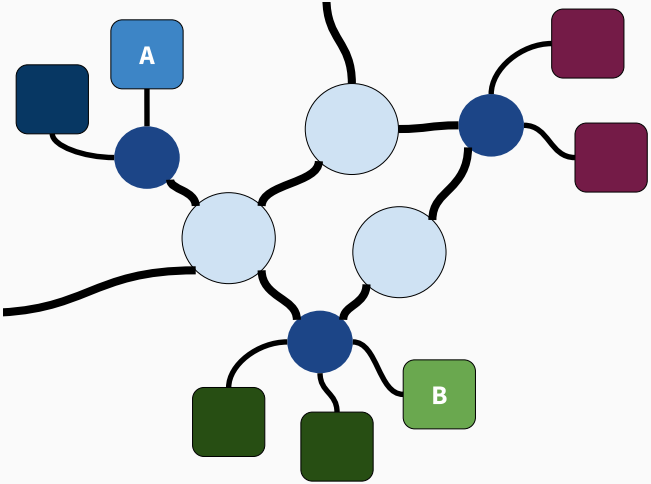

Figure 4: How can we get data from A to B *reliably*?

Option 1: In-network reliability

- Each link-level connection is reliable
- Pros:
  - Simple abstraction to applications (end-to-end protocols)
  - Resiliant to changes in network topology
- Cons:
  - High packet overhead
  - Higher latency
  - Doesn't solve the end-to-end problem
    - Often still need to check validity of multi-packet transfer (e.g. a whole file)

Option 2: End-to-end reliability

- Each link may be unreliable
- Higher-layer protocols use acknowledgement packets, timeouts, checksums...
- Pros:
  - More general: only *some* applications *want* reliable transport
  - More performant when packet loss is rare
- Cons:
  - Pushes complexity up the stack
  - May take longer to notice and retransmit packets
  - Harder to handle changes in network topology

This is what the Internet uses

Narrow Waist: The Internet's Layered Model

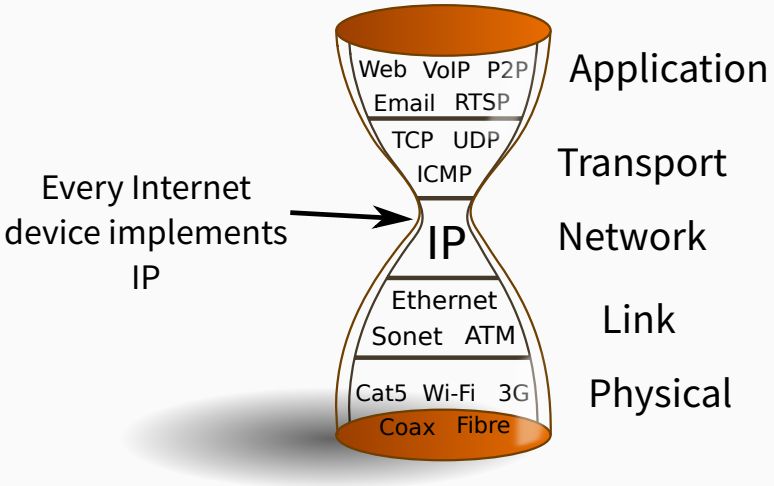

Figure 5: IP is the "narrow waist" of the Inernet [1]

Why is a “narrow waist” useful?

Why is a “narrow waist” useful?

- Portability: Interconnecting varying networks
  - Ethernet, WiFi, DSL, Cable, 4G, etc. . .
  - Hides variation from applications
- Generality: many applications can use the same Internet
  - The Web, file sharing, video streaming, VOIP, e-mail, etc. . .
  - Internal nodes don’t need to change for new application

Summary

This time: The Internet

- The Internet is a graph: a network of networks
- Routing through packet switching
- Reliability end-to-end
- Narrow waist makes one system general for many applications, portable across many networks.

Next time: Syllabus & Distributed systems & Databases

References

References

[1] *A representation of the Internet Hourglass.* Wikimedia Commons.

[2] *Photograph of Women Working at a Bell System international Telephone Switchboard.* The U.S. National Archives.