CS434/534: Topics in Network Systems

Introduction: Course Topics and Planning, Network Systems Bigger Picture

Sept. 7, 2021

Y. Richard Yang
Computer Science Department, Yale University
Email: yry@cs.yale.edu

http://zoo.cs.yale.edu/classes/cs434/
Outline

- Course topics
- Course planning
- Bigger picture
Scope: Networking Systems as Network Systems

- Networking systems: Create and maintain networking services such as ensuring connectivity, scheduling networking resources

- Networking systems include some of the most influential inventions
  - Internet
    - A research experiment that escaped from the lab
    - ... to become the global communications infrastructure
  - Cellular networking systems (e.g., 5G)
    - An enabling capability for our modern society
  - Data center/cloud networking systems
    - The foundation infrastructure of many technology services
  - Constant innovation
    - IoT networking, space networking ...
Networked systems: higher level computer system services built on top of networking services

Networked systems include some of the most important, pervasive computer systems, e.g.,

- Web, IM, multimedia
- Large-scale data analytics (e.g., Hadoop, Spark, ...)
- Data store, pub/sub system (e.g., Raft, Kafka)
- ...
Networking Systems and Networked Systems

- Depend on networking systems capabilities
- Work around networking systems “limitations”
- Designed to support target networked systems

One goal of this class is to integrate both networking systems and networked systems
Network Systems Topics We Plan to Cover

- T1: High-perform Network Server Design, Impl. Analysis
- T2: High-performance, Reliable Transport Design & Implement.
- T3: Software-Defined Networking
- T4: Microservices Architecture
- T5: Cloud Data Center (CDC) Networking Control Plane
- T6: Cloud Apps & Apps-Driven Network Systems
- T7: Mobile, Edge Cloud Network Systems
Common Design Goals

- Scale to massive scale
- Reduce latency/tail latency
- Drive down costs
- Support large-scale resource sharing across users (multi-tenants), apps and servers
- Integrate next-generation hardware, e.g., accelerator, NIC, programmable switch, RDMA fabric, ...
- Improve software development productivity
Fundamental Challenge Achieving Goals: Complexity

- **Complexity** arises from design strategies intended to create
  - **Scalability**
    - to handle the size and complexity of a system as a whole
  - **Efficiency**
    - to handle resource scarcity
  - **Reliability**
    - to handle component failures
  - **Modularity**
    - to allow reuse of components
  - **Evolvability**
    - to allow reuse of components in time
“Developers who have worked at the small scale might be asking themselves why we need to bother when we could just use some kind of out-of-the-box solution. For small-scale applications, this can be a great idea. We save time and money up front and get a working and serviceable application. The problem comes at larger scales—there are no off-the-shelf kits that will allow you to build something like Amazon... There’s a good reason why the largest applications on the Internet are all bespoke creations: no other approach can create massively scalable applications within a reasonable budget.”

http://www.evontech.com/symbian/55.html
Cloud Advantages

- Single administrative domain, no need to be compatible with outside world when inside itself
- Tiny round trip times (microseconds)
- Control both networks and end hosts
- Economy of scale—huge buyers that can drive market
Course Topic 1: High Performance Network Server Design, Implementation, Analysis

- Representational State Transfer (REST), Fielding Thesis
- RFC7230 (HTTP/1.1)
- Basic socket network programming
- Thread (thread, mutex, condition var) based network server design
- select, epoll, finite-state machine based network server design
  - “scheduler”/eventloop design, mixing io/tasks, handler pipeline
- Design patterns, staged, event-driven architecture
- Netty and nginx source-code analysis (ch.4 vol II, The architecture of open source applications)
- Server bottleneck analysis using operational analysis
- Programming project 1: High-performance HTTP server Design & Analysis
Course Topic 2: High-Performance, Reliable Network Transport Design & Impl

- RFC793 (TCP)
- RFC5681, RFC5682 (TCP congestion control)
- RFC8312 (Cubic), BBR, MP-TCP
- RFC5246 (TLS1.2), RFC8446 (TLS1.3)
- RFC7540 (HTTP/2)
- RFC900 (QUIC)
- HTTP/3 draft

- Programming project 2: Socket API and TCP protocol design and implementation
Course Topic 3: Software-Defined Networking (SDN) Architecture

- OpenFlow SDN (OF1.3)
- SDN programming language: Maple, Frenetic, Magellan, Trident
- Programmable ASIC: RMT, P4
- Network operating systems: Onix/Orion’21 (Google), SONIC (Microsoft), FBOSS (FB)
Course Topic 4: Microservice Architecture

- Microservices basic concepts (containers, pods, deployment, service)
- Foundation of realizing containers: namespace, veth, ip, iptables
- Microservice architecture control plane:
  - Cluster control: etcd, k8s internals scheduler
  - Networking control (flannel; calico; weave; istio)
Course Topic 5: Cloud Data Center Networking Systems Design & Analysis (Part 1; Control)

- Classical cloud DC designs: Basic Clos, Facebook topology, Microsoft VL2, Google Juniper, DC evolving
- DC WAN control: Google B4'13/B4 and after'18, Microsoft SWAN
- Peering control: BGP background, Espresso, edge fabric

- Guest lecture: Google network infrastructure architecture
Course Topic 5: Cloud Data Center Networking Systems Design and Analysis (Part 2/Analysis)

- Data path analysis (HSA, VeriFlow, APKeep)
- Control-path analysis (minesweeper)
- Data center traffic analysis in Microsoft, Facebook

- Guest lecture: Alibaba’s programming and verification framework (Lyra, Aquila)
  - Ennan Zhai (Alibaba)
Course Topic 6: Cloud Applications & Application-Driven Networking Systems

- Data analytics (DA) programming model
  - MapReduce, Noria, Spark
  - Spark perf measurements (DA analysis)

- DC cluster compute scheduling
  - Delayed scheduling, YARN, Mesos, Borg, DRF

- Low latency, high-tput DC transport scheduling: DCTCP, RDMA, DCQCN; TIMELY; HPCC’20; On-Ramp’21

- Coflow scheduling: Coflow scheduling: Sincronia’18

- RPC scheduling: gRPC, eRPC, nanoPU
Course Topic 7: Mobile, Edge Networking

- Edge connection using MANET: RFC7481 (OLSRv2)
- Edge connection using 5G: 3GPP TS23.501, TS23.502, OpenAPI

- Guest lecture: Amazon Wavelength (Deliver ultra-low latency applications for 5G devices)
  - Saravanan Shanmugam (Amazon AWS)
Questions?
Outline

- Course topics
- Course planning
Course Planning

- Planning using “how we learn [Dr. William Glasser]” guideline
  - 10% of what we read
  - 20% of what we hear
  - 50% of what we read and hear
  - 70% of what we discuss
  - 80% of what we experience personally
  - 95% of what we teach others

- Three core components
  - understanding of main real protocols, main real network systems
  - discussing in class and with top experts
  - learning by implementing core network systems

Course Planning: Classroom Time

- Understand the design and implementation of state-of-the-art, foundational network systems, in particular, their complexity (where complexities come from, ...)

- Content based on real protocols, source code of real, large systems, papers reporting design & implementation of real, large systems

- Mixed lectures and discussions
Course Planning: Assignments

- Two types of assignments
  - Programming: Manageable design and implementation of real network systems, compare your design with real systems
  - Paper/system reading: Students need to answer questions and write reviews of some systems and papers
Course Planning: Course Projects

- **Project topics**
  - Undergraduate independent work
  - Graduate student research projects
  - Can overlap with other projects, with permission
  - Must involve writing some software
  - If possible, develop a system that can be truly deployed

- **Can work alone or in small teams**

- **Project topics suggestions:** my ongoing projects, potential projects when we cover a topic
Project Timeline

- Initial proposal + midterm progress report + presentation + final report [6-8 pages]

- We provide help in logistics such as
  - Devices
  - Cloud (amazon/Google) service accounts
  - Interaction with top experts of the fields
Grading

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming assignments</td>
<td>45%</td>
</tr>
<tr>
<td>Project</td>
<td>45%</td>
</tr>
<tr>
<td>Written assignments</td>
<td>15%</td>
</tr>
<tr>
<td>Class participation</td>
<td>10%</td>
</tr>
</tbody>
</table>

- More important is what you realize/learn than the grades
Questions?
Outline

- Course topics
- Course planning
- Big picture
  - Physical infrastructure
The Internet (in Cool Picture)

https://www.kaspersky.com/blog/amazing-internet-maps/10441/
The Internet (w/ Structure)

Residential access network
- Cable, Fiber, DSL, Wireless, Cellular

Campus access network, e.g.,
- Ethernet
- Wireless

- Many network systems operate on a network of networks
- Each individually administrated network is called an Autonomous System (AS)
Typical Campus/Enterprise Network (w/ More Details)
Types of Boxes in Campus/Enterprise Networks

- Modern networks contain diverse types of equipment beyond simple routing/forwarding

Source: [Sherry, et. al SIGCOMM’12]

Small: <=1k hosts; Medium: 1k-10k; Large: 10k-100k; Very Large: >= 100k
Data Center Networks (in Cool Pictures)
Data Centers (in Pictures)
Data Center Network (w/ a Structure)

https://engineering.fb.com/2019/03/14/data-center-engineering/f16-minipack/
Example: K-ary fat tree using k-port switches

- three-layer topology (edge, aggregation and core)
- k pods w/ each pod consists of \((k/2)^2\) servers & 2 layers of \(k/2\) k-port switches
  - each edge switch connects to \(k/2\) servers & \(k/2\) aggr. switches
  - each aggr. switch connects to \(k/2\) edge & \(k/2\) core switches

Offline questions to think: (1) How large a network is built using k-port switches? What is nice about the design?

Outline

- Course topics
- Course planning
- Big picture
  - Physical infrastructure
  - The layering architecture for network systems
Layered Architecture

A technique to organize a network system into a succession of logically distinct entities, such that the service provided by one entity is solely based on the service provided by the previous (lower level) entity.
Why Layering?

Introducing an intermediate layer provides a common abstraction for network technologies.
An Example: No Layering

Application

Transmission Media

Telnet

FTP

HTTP

Ethernet

Fiber optic

Wireless
Layering Basic Concepts

- Service - says what a layer does
- Interface - says how to access the service
- Protocol - specifies how the service is implemented
An Example of Layering
An Example of Layering
Basic Concept: Network Protocol

- Defines the **format** and the **order** of messages exchanged between two or more communicating **peers**, as well as the **actions** taken on the transmission and/or receipt of a message or other events.
Network Systems == Protocols?

SNMP  WAP  PPP  IPX  RAFT
LLDP  FTP  UDP  ICMP  MAC
OSPF  RTP  BGP  HTTP  HIP
EIGRP  RED  ARP  IMAP  IGMP
RIP  IP  NET  TCP  ECN
SMTP  RTSP  MPLS  COAP  CIDR
NNTP  SNMP  TLS  BFD  STUN
QUIC  DNS  SACK  TLS  STUN
POP  VLAN  SSH  VTP  DHCP
VTP  LISP  RTSP  TFTP  ISIS
Example Protocol: Simple Mail Transfer Protocol (SMTP)

- **Messages from a client to a mail server**
  - HELO
  - MAIL FROM: <address>
  - RCPT TO: <address>
  - DATA
    `<This is the text end with a line with a single .>`
  - QUIT

- **Messages from a mail server to a client**
  - status code
    - The first digit of the response broadly indicates the success, failure, or progress of the previous command.
      - 1xx - Informative message
      - 2xx - Command ok
      - 3xx - Command ok so far, send the rest of it.
      - 4xx - Command was correct, but couldn't be performed for some reason.
      - 5xx - Command unimplemented, incorrect, or a serious program error occurred.
  - content

Command: `telnet netra.cs.yale.edu smtp`
Wireshark capture: port smtp
Next Class

- End-to-end argument example
- HTTP/1.1 design

- Read, questions to think about
  - Read (reference also see the Reading document]
    - End-to-end arguments in system design [optional]
    - History of HTTP [optional]
    - REST [recommended]
  - Q: What is some limitation of layered NS?
Backup Slides
Cloud Computing DC

- August 25, 2006: Amazon announced EC2 => Birth of Cloud DCN in reality (Prior theoretical concepts of computing as a utility)

- Prediction by 2020:
  - 11/12th of workload in clouds, 1/12th in traditional data center
  - 92% Data center IP traffic in clouds, 8% in traditional data center
  - 5X growth in data center storage
    - 7/8th in cloud, 1/8th in traditional data centers
    - 2/3rd in public clouds, 1/3rd in private clouds

Ref: Cisco, “Cisco Global Cloud Index: Forecast and Methodology, 2015-2020,” 2016,
Google DCN Traffic Growth

Typical Application Patterns: Web Search

Partition/Aggregate App Structure

• Strict deadlines

"Computers are useless. They can only give you answers."

[Picasso]

"Everything you can imagine is real."
"Bad artists copy. Good artists steal."
"It is your work in life that is the ultimate seduction."
"The chief enemy of creativity is good sense."
"Inspiration does exist, but it must find you working."
"I'd like to live as a poor man with lots of money."
"Art is a lie that makes us realize the truth."
"Computers are useless. They can only give you answers."

worker nodes