Distributed Systems and Security: An Introduction

Brad Karp
UCL Computer Science

CS GZ03 / M030
2\textsuperscript{rd} October 2017
Today’s Lecture

• Administrivia
• Overview of Distributed Systems
  – What are they?
  – Why build them?
  – Why are they hard to build well?
• Operating Systems Background
• Questionnaire
Prerequisites

• Undergraduates: must have taken UCL CS 3035, Networked Systems, or equivalent experience (3rd-year undergrad networking class, covering Internet protocols and architecture in depth)

• Graduates: must be concurrently enrolled in UCL CS GZ01, Networked Systems, or equivalent prior experience (3rd-year undergrad networking class, covering Internet protocols and architecture in depth)

• All: substantial programming experience; C or C++ preferred, Java acceptable if willing to commit to catch up on C on own time
Course Staff and Office Hours

- Instructor:
  - Brad Karp, MPEB 6.20, Mon 6 – 7 PM, ext. 30406

- Teaching Assistant:
  - Nikola Gvozdiev, MPEB 7th floor lab, Wed 6 – 7 PM, ext. 33670

- Office hours begin today

- Time reserved for answering your questions

- Outside office hours, email to schedule appointment
Meeting Times and Locations

- (Most) Mondays 11 AM – 12:30 PM, ULU Student Central 3D
- (Most) Wednesdays 9:30 – 11 AM, 16 Taviton St. 433
- (A Few) Fridays 5 – 6:30 PM, Roberts 422
- Lecture will usually run 90 minutes
- Occasionally lecture will be followed by a 30-minute discussion of an additional topic (e.g., Q&A on a coursework); on these dates, full two hours!
- No lecture 9th, 30th October; 1st, 22nd, 29th November
- Reading week: 6th – 10th November, no lecture!
Class Communication

- Class web page
  - http://www.cs.ucl.ac.uk/staff/B.Karp/gz03/f2017/
  - Detailed calendar, coursework, class policies
  - Your responsibility: check page daily!

- M030/GZ03 Piazza Page
  - https://piazza.com/ucl.ac.uk/fall2017/computersciencem030gz03
  - Important announcements from class staff (also forwarded to you by email)
  - Postings from class staff and students
  - Subscribe using enrollment key
  - You must subscribe (class policy)
  - Your responsibility: check email daily!
Using Piazza

• Please post questions on class material on Piazza, rather than emailing course staff.
• Whole class benefits from seeing your question and its answer.
• Students are encouraged to answer one another’s questions!
• When discussing something private (e.g., your marks, or details of your specific solution to a coursework), mark your post as private, so only class staff see it!
Programming Coursework: GitHub for Submission

• New this year: we are providing private GitHub repositories for GZ03/M030 students
• Obtain code we give you by cloning our repo
• Use your private GZ03/M030 GitHub repo for source code control (revision history, backup, etc.)
• Submit your coursework through your private GZ03/M030 GitHub repo (i.e., not via Moodle!)
Readings, Lectures, and Lecture Notes

• Readings must be read before lecture; lectures assume you have done so
• Lecture notes will be posted to the class web site just after lecture
• Class calendar shows all reading assignments day by day...
Readings

- No textbook
- Classic and recent research papers on real, built distributed and secure systems
- Available on class web page; print these yourselves
- All readings examinable
- Research papers are dense and complex; they are often challenging
  - Be prepared to read and re-read the papers
  - Come to lecture with questions, and/or use office hours
Grading

• Final grade components:
  – One programming coursework: 15%
  – One problem set coursework: 15%
  – Final exam: 70%
Late Work Policy

• N.B. that M030/GZ03 policy differs from that for other CS classes!
• For every day late or fraction thereof, including weekend days, 10% of marks deducted
• Each student receives budget of 3 late days for entire term
  – Each late day “cancels” one day of lateness
  – Goal: give you flexibility, e.g., in case you can’t find a bug, or encounter unexpected other snag
  – You declare how many late days to use when turning in a coursework late; cannot declare or change later!
  – Must use whole late days—cannot use fractional ones!
Late Days (cont’d)

• If submission more than 2 days late after taking late days into account, zero marks
• Programming courseworks turned in online; may be submitted 24/7
• Problem set courseworks turned in on paper in lecture; can be submitted M – F only
  – Weekend days after deadline still count as elapsed days
Late Days (cont’d)

• If submission more than 2 days late after taking late days into account, zero marks

Late days give you flexibility. No other extensions given on coursework, unless for unforeseeable, severely extenuating circumstances!

Paper in lecture; can be submitted M – F only
– Weekend days after deadline still count as elapsed days
Academic Honesty

• All courseworks must be completed individually
• May discuss understanding of problem statement, general sketch of approach
• May not discuss details of solution
• May not show your solution to others (this year or in future years)
• May not look at others’ solutions (this year or from past years)
• May neither post questions on nor troll for related questions and answers on public Internet Q&A forums (e.g., StackExchange, etc.)
Academic Honesty (cont’d)

- We use code comparison software
  - Compares parse trees; immune to obfuscation
  - Produces color-coded all-student-pairs code comparisons
- Don’t copy code—you **will** be caught!
- Penalty for copying: automatic zero marks, referral for disciplinary action by UCL (usually leads to exclusion from all exams at UCL)
Today’s Lecture

• Administrivia

• Overview of Distributed Systems
  – What are they?
  – Why build them?
  – Why are they hard to build well?

• Operating Systems Background

• Questionnaire
What Is a Distributed System?

- Multiple computers (“machines,” “hosts,” “boxes,” &c.)
  - Each with CPU, memory, disk, network interface
  - Interconnected by LAN or WAN (e.g., Internet)
- Application runs across this dispersed collection of networked hardware
- But user sees single, unified system
What Is a Distributed System? (Alternate Take)

“A distributed system is a system in which I can’t do my work because some computer that I’ve never even heard of has failed.”

– Leslie Lamport, Microsoft Research (ex DEC), 2013 Turing Award winner
Start Simple: Centralized System

- Suppose you run Gmail
- Workload:
  - Inbound email arrives; store on disk
  - Users retrieve, delete their email
- You run Gmail on one server with disk
Start Simple: Centralized System

• Suppose you run Gmail
• Workload:
  - Inbound email arrives; store on disk
  - Users retrieve, delete their email

What are shortcomings of this design?

• You run Gmail on one server with disk
Why Distribute?
For Availability

• Suppose Gmail server goes down, or network between client and it goes down
• No incoming mail delivered, no users can read their inboxes
• Fix: replicate the data on several servers
  – Increased chance some server will be reachable
  – Consistency? One server down when delete message, then comes back up; message returns in inbox
  – Latency? Replicas should be far apart, so they fail independently
  – Partition resilience? e.g., airline seat database splits, one seat remains, bought twice, once in each half!
Why Distribute?
For Scalable Capacity

• What if Gmail a huge success?
• Workload exceeds capacity of one server
• Fix: spread users across several servers
  – Best case: linear scaling—if $U$ users per box, $N$ boxes support $NU$ users
  – Bottlenecks? If each user’s inbox on one server, how to route inbound mail to right server?
  – Scaling? How close to linear?
  – Load balance? Some users get more mail than others!
Performance Can Be Subtle

• Goal: predictable performance under high load
• 2 employees run a Starbucks
  – Employee 1: takes orders from customers, calls them out to Employee 2
  – Employee 2:
    • writes down drink orders (5 seconds per order)
    • makes drinks (10 seconds per order)
• What is throughput under increasing load?
Starbucks Throughput

- Peak system performance: 4 drinks / min
- What happens when load > 4 orders / min?
- What happens to efficiency as load increases?
Starbucks Throughput

What would preferable curve be?
What design achieves that goal?
Why Are Distributed Systems Hard to Design?

- **Failure: of hosts, of network**
  - Remember Lamport’s lament

- **Heterogeneity**
  - Hosts may have different data representations

- **Need consistency (many specific definitions)**
  - Users expect familiar “centralized” behavior

- **Need concurrency for performance**
  - Avoid waiting synchronously, leaving resources idle
  - Overlap requests concurrently whenever possible
Security

• Before Internet:
  – Encryption and authentication using cryptography
  – Between parties known to each other (e.g., diplomatic wire)

• Today:
  – Entire Internet of potential attackers
  – Legitimate correspondents often have no prior relationship
    – Online shopping: how do you know you gave credit card number to amazon.com? How does amazon.com know you are authorized credit card user?
    – Software download: backdoor in your new browser?
    – Software vulnerabilities: remote infection by worms!
  – Crypto not enough alone to solve these problems!