Distributed Systems
CPSC 416
Fall 2018

Course: September 6 - November 30, 2018

Sep 6, 2018 Lecture (first class!)
Course staff

- Ivan Beschastnikh, instructor
- TAs
  - Anny Gakhokidze (u)
  - Vaastav Anand (g)
  - Adam Geller (g)
Logistics

- 2016: 77 students (open-ended project)
- 2017: 117 students (assignment hell)
- 2018W: 160 students (assignments + projects)
- 2018F: ~70 students (mix of above)
  - 3 full TAs
  - 2 assignments, 2 projects. 3/4 require group work. One (group) open-ended project
Logistics

• Everything on the website, updated continuously:
  http://www.cs.ubc.ca/~bestchai/teaching/cs416_2018w1/

• Use Piazza for all course-related communication

• 4 hrs office hours/week
Course overview via the website

- Learning goals
- Go programming language (start learning!)
- Schedule (a work in progress)
  - Assignment 1 due Sep 18 (12 days from now)
- Exam (‘just’ a final)
- Advice for doing well
  - learn Go (a must to pass the course)
  - don’t hack, engineer
  - choose team, wisely
  - reach out on Pizza/email for help.
- Collaboration guidelines
Learning goals

• Understand key principles in designing and implementing distributed systems

• Reason about problems that involve distributed components

• Become familiar with important techniques for solving problems that arise in distributed contexts

• Build distributed system prototypes using the Go programming language
Learning goals

• Understand key principles in designing and implementing distributed systems

• Reason about problems that involve distributed components

• Become familiar with important techniques for solving problems that arise in distributed contexts

• Build distributed system prototypes using the Go programming language (the key to all the above)
Some workload comments from last year’s course

• The workload for this course is easily double that of any other course I had this term.

• Ivan has very high expectations of his students.

• I love and hate the fact that this class was a "sink or swim" approach to learning
Assignment 1: Failure detector lib

• What’s a failure detector?

• Why is this a distributed systems topic? And, why do we need a failure detector?

• Isn’t there a library I can use for this already?

• Deeper: why doesn’t Go/OS/switch/network/universe provide a service for this already?
Assignment 1: Failure detector lib

- Topology, message types (hbeat/ack), transport (UDP)
Assignment 1: Failure detector lib

• Two protocols/APIs: client to fdlib and fdlib to fdlib
Assignment 1: Failure detector lib

- Two fdlib capabilities: responding & monitoring
Assignment 1 note

• Last last year’s 416 TA rant:

  TEST YOUR CODE ON THE UGRAD MACHINES!!!!!!!!!!!!!!!!!!!!

  YOU WILL GET ZERO IF IT DOESN'T RUN OR COMPILE. WE HAVE NO SYMPATHY FOR THESE TYPES OF ERRORS.

  … you’ve been warned
What are some examples of distributed systems?

What makes a system distributed?

Why not a distributed application?
Distributed system examples

• YouTube

  • Videos are **replicated** (multiple machines host the same video)

  • **Scalable** wrt. client requests for videos (internally **elastic** — can throw more machines at the service to have it scale out further)
Distributed system examples

• DropBox (or google drive)

  • Replicated content across personal devices

  • Supports disconnected operation (can work while disconnected, and synchronize when re-connected)

  • Maintaining data consistent across devices

• Supports sharing; access control policies (security!)
Distributed system examples

- NASDAQ

- **Transactions** (e.g., ACID semantics from databases). Many DBMS concepts apply to distributed systems!

- Strong *consistency* and *security* guarantees (otherwise people would not trust it with money)
Some D.S. challenges

- Synchronizing multiple machines (protocol complexity)
- Performance (how do you define/measure it?)
- Maintaining consistency: strong models (linearizable) to weak models (eventual) of consistency
- Failures: machine failures (range: failure stop to byzantine); network failures (just a few: disconnections/loss/corruption/delay/partitioning)
- Security (how to prevent malicious control of a single host in a system escalating into control of the entire system?)
For Monday

• Install Go on your personal machine

• Work through Tour of Go! and other tutorials.

• **Practice Go!**