DISTRIBUTED SYSTEMS AND ALGORITHMS

CSCI 4963/6963

8/29/2016

General Information

- Lectures: MR 12pm – 1:50pm, Sage 5510
- Instructor: Stacy Patterson (me) sep@cs.rpi.edu
- Office Hours: M 2pm – 3pm in Lally 301
- Course web site: http://www.cs.rpi.edu/~pattes3/dsa_fall2016

- TA: Erika Mackin (mackie2@rpi.edu)
- TA Office Hours: TBD

Course Objectives

- This is a theory course, despite the name.
- The goal is to for you to learn important theory and algorithms for distributed computing systems.
  - Through theory and practice.
- These algorithms are actually used in data centers and cloud computing systems today.

General Information (continued)

- Course content will be presented in lectures.
  - Related conference and journal papers will be posted on the course web site.
  - I will not post lecture notes on the web site.

- **Optional** supplementary textbook:
  Distributed Systems and Concepts by Coulouris et al.
  - The book may present different variants of algorithms that we cover in class.
  - You are responsible for learning the algorithms taught in lecture.

Pre-Requisites

- CSCI-2300: Intro to Algorithms
  - Analysis of algorithm correctness and performance
  - Writing **correct** proofs of algorithm properties
- CSCI-4210: Operating Systems
  - Multi-threaded programming
  - Network communication (socket programming)
- No linear algebra or PDEs in this course.

Grading

- Quizzes: 55%
- Take-home Final Exam: 15%
- Programming Projects: 30%

- Grades will be posted on LMS
- We will be trying out Gradescope for quiz and exam grading.

Course Letter Grades

	B+: 87 – 89	C+: 77 – 79	D+: 67 – 69	F: 0 – 59
A: 93 – 100	B: 83 – 86	C: 73 – 76	D: 60 – 66	
A-: 90 – 92	B-: 80 – 82	C-: 70 – 72		

- I may lower the cutoff points.
- I may use different curves for 4963 and 6963.

Quizzes

- Quizzes will be:
  - Closed book
  - About 45 minutes each
  - Done independently
  - Announced in the lecture preceding the lecture in which they will be given.
- Quizzes are meant to evaluate your understanding of the algorithms, not test your memorization skills.
- No makeup quizzes will be given without an official excused absence.
- Regrade requests must be made within 7 days of quiz return.

Final Exam

- The final exam will be:
  - Take-home
  - Comprehensive
  - Open notes
  - Due in the last week of classes

- We will talk about the collaboration policy closer to the exam date.

Programming Projects

- There will be 2 programming projects.
- Projects will be done in groups of 2.
  - Exceptions to this must be approved by me in advance.
- Projects will give you the chance to implement distributed algorithms in real-world distributed computing systems – Amazon EC2
- You can use your language of choice (within reason).

- More details in a few weeks.

Special Accommodations

- If you need special accommodations for this class, please let me know at least two weeks before the affected assignment.

Academic Integrity Policy

- No collaboration or outside resources are allowed on quizzes unless I announce otherwise.
- For programming assignments, you may discuss the project with other students, but you (your team) must write your own code.
    - No sharing code or reusing code unless approved by me in advance.
- We will discuss collaboration policy for exams closer to the final exam date.

- Any student who violates these policies will be subject to penalties outlined in the Rensselaer Student Handbook.

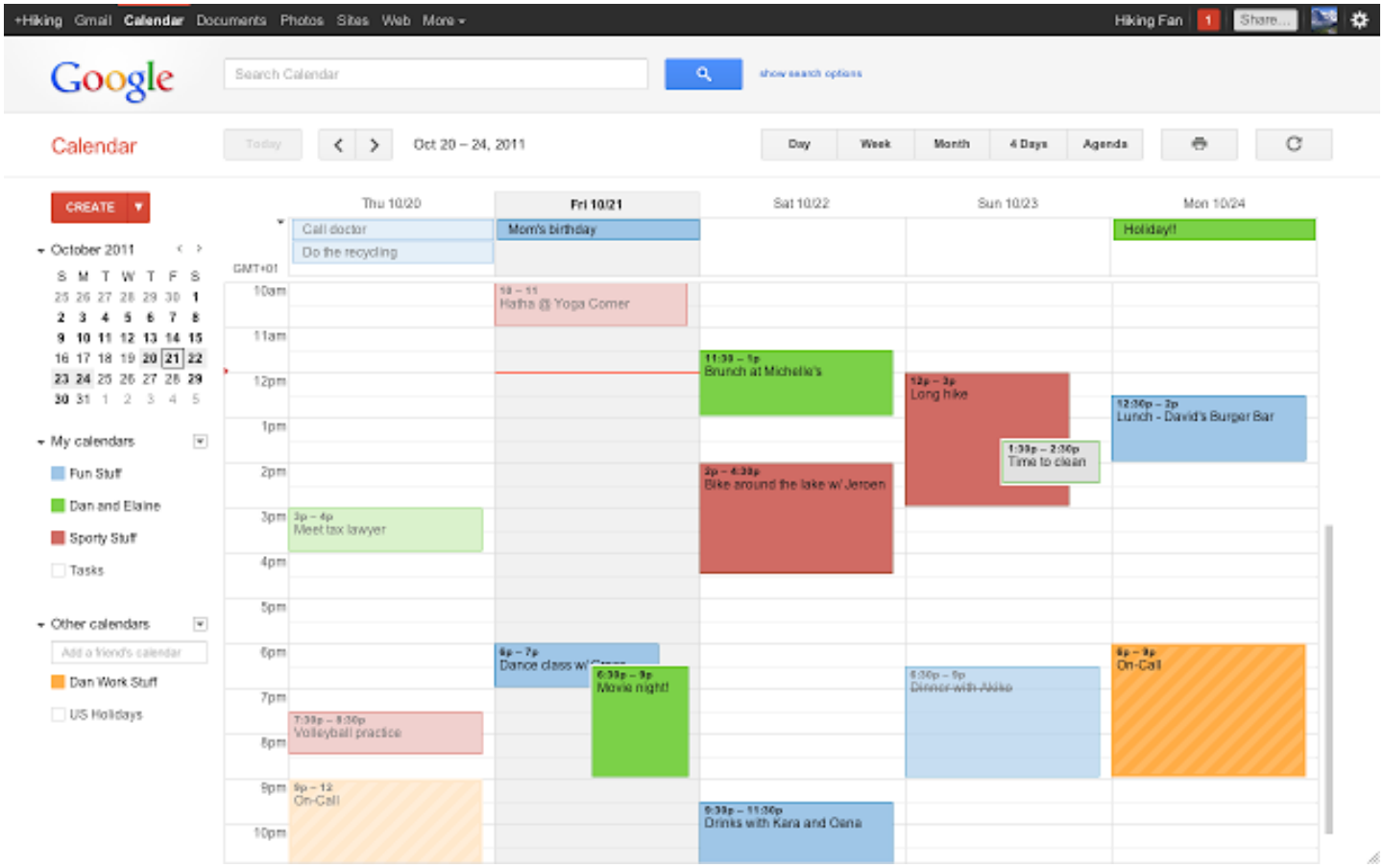
INTRO TO DISTRIBUTED SYSTEMS

What is a distributed system?

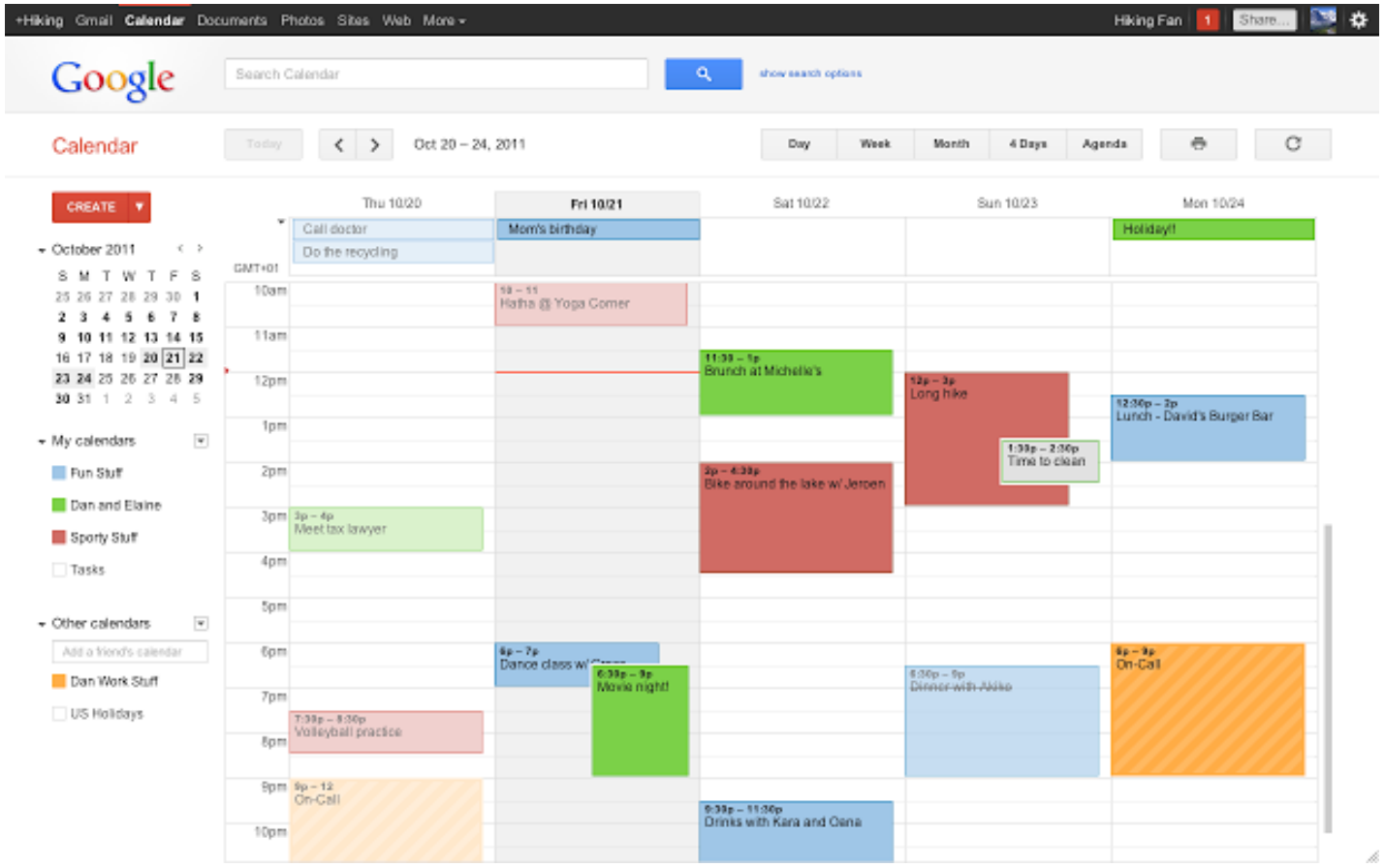
- "A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages."

  Coulouris et al., *Distributed Systems*

- Significant characteristics
  - **Concurrency**: Different operations executed on different computers at the same time
  - **No global clock**: Difficult to synchronize (coordinate) actions on different computers
  - **Independent failures**: computers can crash, the network may fail or slow down, network partitions may arise.
    - The rest of the system keeps running, may not be aware of failures.



- I want the application to behave like
  - it is running on a single computer with infinite resources that never fails,
  - and I am the only one using that application.



- The application is actually
  - running on thousands (more or less) of computers,
  - spread across multiple data centers,
  - with thousands (or more) of simultaneous users.

The Horrible Truth...

Typical first year for a new cluster:

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packetloss)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~12 router reloads (takes out DNS and external vips for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

~1000 individual machine failures

~thousands of hard drive failures

slow disks, bad memory, misconfigured machines, flaky machines, etc.

Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.

- Reliability/availability must come from software!

Slide by Jeff Dean, Google Senior Fellow

Google

What is a distributed system?

“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

Models of Distributed Systems

- What are the entities that are communicating in the distributed system?
  - **An entity is a single process**
  - Other options: objects, services, …
- What communication paradigm do they use?
  - **Entities communicate by sending messages**
  - Other options: shared memory, RPC, publish/subscribe, …
- How are they mapped onto the physical distributed infrastructure?
  - **A process runs on a single physical machine**
  - Other options: mobile code, mobile agents, …

Some Components of a Model

- Interaction characteristics
  - Can messages be lost?
  - Do they arrive in the order in which they were sent?
  - What about message delay?
- Failures
  - Can processes crash?
  - Can they recover?
- Security
  - Do all processes follow the specified algorithm?
  - If not, what kind of “attacks” are allowed?

Two Important Model Variants

- Synchronous System: Known bounds on times for message transmission, processing , bounds on local clock drifts, etc.
  - Can use timeouts

- Asynchronous System: No known bounds on times for message transmission, processing, bounds on local clock drifts, etc.
  - More realistic, practical, but no timeout.

What is a distributed algorithm?

- Steps taken by each process including:
  - Sending and receiving messages.
  - Changing local state.

- We will analyze algorithms in the context of models.
  - An algorithm may work under one model but not another.
  - Some problems may be solvable under one model but not another.

Course Topics

- Clocks and the ordering events in distributed systems
- Distributed mutual exclusion
- Distributed logs
- Global snapshots
- Broadcast algorithms
- Leader Election
- Distributed Agreement

Course Topics (cont.)

- Distributed Commit Protocols
- Concurrency Control
- Replication and Consistency Models
- Consistent Hashing and P2P Networks
- Digital Currencies