Syllabus for SE 14-424: Distributed Systems
Department of Software Engineering
Achi Racov School of Engineering
Kinneret College on the Sea of Galilee

Instructor: Michael J. May

Semester 2 of 5786

1 Course Details

The course meets **12:00pm–3:00pm** on Sundays in Room 6201. The laboratory session for the course is **2:00pm–4:00pm** on Thursdays in Room 6201. The course has **3** hours of lecture and **2** hours of laboratory.

2 Overview

This course is a followup to previous courses on network communication and systems programming. It introduces the design and implementation of *distributed systems*, systems where data, computation, and resources are distributed across a network. We will study several internet based applications which illustrate good distributed systems design.

The course is divided into three units, each of which will have an accompanying team development assignment. The course units are as follows:

1. Network distributed communication and coordination
2. Distributed data storage
3. Distributed consensus and failure management

Each unit will include a discussion of the cybersecurity impacts and issues surrounding the relevant algorithms and systems.

The theoretical material for the course is drawn from Van Steen and Tanenbaum [12] supplemented with material from other sources. Each software system that we will use has its own documentation that will be very helpful when using it with assignments.

Course Prerequisites Course prerequisites: SE 14-331: Introduction to Computer Networks, SE 14-322: Software Engineering for Software Intensive Systems, and SE 14-317: Operating Systems.

2.1 Learning Outcomes

By the end of the course, students will be able to:

1. Define a distributed system and give examples of several different distributed systems paradigms.

2. Design and implement application level communication protocols using TCP or UDP.
3. Use an open source message queue system (*e.g.* RabbitMQ) for communication between computers.
4. Use an open source distributed database system for distributed storage and retrieval of records.
5. Use an open source framework for consensus and failure recovery (*e.g.* Raft) for a distributed set of servers.
6. Explain cybersecurity issues related to network communication and coordination, distributed data storage, and distributed consensus and failure management.

3 Lecture Schedule

The course lectures are structured in the following way. The relevant chapters of Van Steen and Tanenbaum (VT) are in the indicated column. Material not covered well in the books may be supplemented from papers or other sources as shown in the O column.

Unit	#	Subject	VT	O
Into	1	Intro to Distributed Systems Client Server and Peer to Peer	1 2.4	
Communication & Coordination	2	Communication: Layering RPC, Sockets, MOM	4.2 4.2–4.3	[3] [2]
	3	Advanced RMI, Apache Kafka	5.3.6	[8, 4]
	4	Multicast Epidemic Algorithms Part 1: C&C cyber security analysis	4.4 4.4 9.1, 9.3	
Distributed Data Storage	5	Naming Intro Flat Naming: Chord, HLS	6.1 6.2	 [13, 1]
	6	Assignment 1 Demos HLS Implementation		
	7	Structured Naming Naming: Resolution and Attribute-based Mutual Exclusion	6.3 6.3–6.4 5.3	
	8	Distributed databases and consistency Part 2: Storage cyber security analysis	7	[11][10]
Consensus and Failure	9	Synchronization: Physical Clocks GPS, NTP	5.1 5.1	
	10	Logical Clocks Totally Ordered Multicast	5.2 5.2	[7][5]
	11	Vector Clocks Data Oriented Consistency	5.2 7.2	[6]
	12	Fault Tolerance and Resistance Raft and reliable group communication Part 3: Consensus cyber security analysis	8.1–8.2 8.2, 8.4	 [9]
Outro	13	Project presentations and course summary		

Since this is an advanced course, students **are expected to come to class having read the material listed above in the lecture schedule**. Students who do not come prepared will find themselves at a

significant disadvantage.

4 Programming Projects

There will be three cumulative assignments during the course of the semester that together will form a single project. They will involve a significant amount of programming and testing. All will involve a large amount of network programming and independent learning.

Each project can be done in groups of 3–4 students. More details of the projects will be distributed during the course of the semester.

5 Laboratory Work

Since the course does not have an assigned Targil period, the Laboratory period will function as both a review forum and a chance to apply the concepts learned in class. Most laboratory periods will feature a programming task, some of which will be spread over several periods. Students may ask questions during the session and the instructor will answer all questions and issues posed.

6 Attendance

Students are responsible for all material presented in class, recitation, and laboratory sessions, all assigned readings, and all material provided for additional reading out of class. Students who miss a lecture or targil can look at the course syllabus and web page to see which material was missed.

Attendance is required. Students who miss 20% or more of course sessions may be expelled from the course.

6.1 Decorum

Students who attend lecture are expected to give their full attention to the material. Reading newspapers, talking on cellular phones, text messaging, or other distracting behavior will not be tolerated.

Students must arrive to lectures **on time**. After ten minutes into class, the door will be locked and no student will be allowed entry. The door will be opened at the next break in the lecture (approximately every 50 minutes). Students who need to leave during lecture for some urgent matter must leave quietly and may return at the next break.

As per college policy, the instructor reserves the right to expel from the classroom any student who is disturbing the lecture or others.

During online lecture and recitation session, any student who consistently disturbs the session may be expelled and barred from attending future sessions.

7 Submissions

7.1 How to Submit Work

To ensure timely submission of projects and work, students must submit work via private GitHub repositories managed by the instructor. Materials sent via email or via any other method risk being ignored or ungraded without consideration of their merits.

When a team of students submits via GitHub, each student on the team must make at least one substantial code commit. If a student's name appears on a submission, but the student doesn't contribute meaningfully to the code on GitHub, the student **will not** receive a grade for the assignment.

7.2 Late Submission Policy

Students are expected to be on time with their project submissions and assignments. Each assignment must be turned in by the date it is due.

Each student will be given 36 slip hours to use for late submissions. The slip hours can be used on a single assignment or divided up among several. Slip hours are rounded up per assignment (*i.e.* 70 minutes late = 2 slip hours, 3 minutes late = 1 slip hour). Once the slip hours are finished, a student's submissions will no longer be accepted.

36 hours after the due date of any assignment, no submissions will be accepted.

Students who are called up to Miluim duty will have their assignment deadlines extended in accordance with college policy.

8 Cheating

Cheating of any sort will not be tolerated. Student collaboration is encouraged, but within limits as set forth in the college's rules on academic integrity. Any students caught cheating will be immediately referred to the department head and the Dean and may receive a failing grade for the course.

Cheating includes:

- Copying information, content, or verbatim text from other students, internet sites, books (other than the ones listed in the bibliography), other unaffiliated individuals to answer questions, solve problems, or aid in programming projects.
- Copying or submitting source code, documentation, or other programming aids **without attribution** from other students, **web sites**, online repositories, text books, open source programs, or other unaffiliated individuals.
- Project teams which submit work which is identical or substantially identical to work submitted by other project teams, whether current or from previous years.
- Other forms of academic misconduct as described on the site: `https://catalog.upenn.edu/pennbook/code-of-academic-integrity/` or as reasonably assessed by the instructor, program head, or dean.

If you have any questions about what constitutes cheating in the above rules, contact the instructor as early as possible.

9 Quizzes

There will be 10 in-class quizzes to assess preparation for the session. The quiz will be short and timed (15 minutes). After the time is up, no more answers will be submittable.

When calculating the quiz score, the lowest 2 grades on quizzes will be dropped.

Students who due to Miluim service complete less than 4 quizzes will be given an alternative assignment instead of the quiz grade.

10 Exams

There is no final exam.

11 Grading

Final grades will be calculated by combining grades from programming projects and a final exam. The grades are weighted as follows:

70%	Programming Projects (required)
30%	Weekly quizzes (required)

The instructor will not address questions about specific individual grades during the lecture or targil sessions. Students may contact the instructor *in person* during office hours or after the lecture/targil sessions at the instructor's convenience.

12 Books

The following book is used for the class:

- Van Steen and Tanenbaum. *Distributed Systems: Principles and Paradigms*. [12]

For Apache Kafka, the following two books are used:

- Narkhede, Shapira, and Palino. *Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale*. [8]
- Kumar and Singh. *Building Data Streaming Applications with Apache Kafka: Design, develop and streamline applications using Apache Kafka, Storm, Heron and Spark*. [4]

The library has copies of the books listed, but students are encouraged to purchase the books as needed.

A bibliography of the books and articles used in the course of the semester is shown below.

13 Contact Information

Instructor:	Michael J. May
Email:	`mjmay@mx.kinneret.ac.il`
Web page:	`https://mjmay-kinneret.github.io/`

References

[1] Gerco Christiaan Ballintijn. *Locating Objects in a Wide-area System*. PhD thesis, Vrije Universiteit, Amsterdam, Oct 2003. https://www.cs.vu.nl/~ast/Theses/ballintijn-thesis.pdf.

[2] Robert Blumen and Doug Fawley. Episode 421: Doug fawley on grpc. IEEE SE-Radio Podcast, 11 Aug 2020. https://www.se-radio.net/2020/08/episode-421-doug-fawley-on-grpc/.

[3] Robert Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile agents for mobile computing. Technical Report PCS-TR96-285, Dept. of Computer Science, Dartmouth College, May 1996. http://www.cs.dartmouth.edu/reports/abstracts/TR96-285/.

[4] Manish Kumar and Chanchal Singh. *Building Data Streaming Applications with Apache Kafka: Design, develop and streamline applications using Apache Kafka, Storm, Heron and Spark*. Packt Publishing, Aug 2017.

[5] Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978.

[6] Friedemann Mattern. Virtual time and global states of distributed systems. In *Proceedings of the Workshop on Parallel and Distributed Algorithms*, volume 2, pages 215–226, 10 1988.

[7] Jeff Myerson and Leslie Lamport. Episode 203: Leslie Lamport on distributed systems. IEEE SE-Radio Podcast, 29 Apr 2014. https://www.se-radio.net/2014/04/episode-203-leslie-lamport-on-distributed-systems/.

[8] Neha Narkhede, Gwen Shapira, and Todd Palino. *Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale*. O'Reilly Media, 1st edition, Sep 2017. https://www.confluent.io/wp-content/uploads/confluent-kafka-definitive-guide-complete.pdf.

[9] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA, June 2014. USENIX Association.

[10] M. Tamer Özsu and Patrick Valduriez. *Principles of distributed database systems*. Springer, New York, 3rd edition edition, 2010.

[11] Jocelyn O. Padallan. *Distributed Database Architecture*. Arcler Press, 2020.

[12] Maarten Van Steen and Andrew S. Tanenbaum. *Distributed Systems: Principles and Paradigms*. CreateSpace Independent Publishing Platform, 4.03 edition, 2023. https://www.distributed-systems.net/index.php/books/ds4/.

[13] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, New York, NY, USA, 2001. ACM.