Course Syllabus

Edit

College of Science/Computer Science Department

CS 249-01, Distributed Computing, Spring 2026

Course and Contact Information

Instructor:	Sriram Rao
Office Location:	DH 282
Telephone:	
Email:	sriram.rao@sjsu.edu
Office Hours:	Thursdays 12-1p and by appointment
Class Days/Time:	Tue/Thu: 1:30-2:45p
Classroom:	Sweeney Hall 347
Prerequisites:	CS 149 (https://catalog.sjsu.edu/content.php?filter%5B27%5D=CS&filter%5B29%5D=249&cur_cat_oid=15&navoid=5382#tt714) and Graduate standing. Allowed Declared Major: Computer Science, Bioinformatics, Data Science. Or instructor consent.

Grader and Labs Information

Grader	Nainoa-Faulkner Jackson
Email	nainoa.faulkner-jackson@sjsu.edu
Lab Hours	To Be Determined
Lab Location	To Be Determined
Office Hours	Additional hours by appointment.
Class Discord Link	**Link (https://discord.gg/Q3S69mxg)**

Course Description

Introduction to application protocols for large scale distributed systems including multiprocessor systems. Lab is based on using protocols to build distributed systems.

Course Format

This is an in-person course.

Course Learning Outcomes (CLO)

Upon successful completion of this course, students will be able to:

1. define the terminology and common ideas of distributed computing.
2. explain fundamental ideas of distributed computing.
3. demonstrate experience with distributed computing principles in modern applications.
4. explain the challenges present in a distributed environment.
5. explain the differences and trade-offs between various solutions for distributed systems problems.

Tentative Schedule

The tentative schedule is listed **here (https://sjsu.instructure.com/courses/1619545/pages/course-schedule)** .

Required Texts/Readings

Textbook

There is **no** required textbook for this class. Instead, there will be reading assignments centered on research papers that discuss in greater depth what we will be discussing in class.

Reading

The reading schedule for this course will be intense. The readings are grouped into major categories, such as coordination, consistency, fault tolerance, etc. Within each category, we will read papers that address that issue in the context of different aspects of distributed systems.

To get the most out of readings, you will be assigned to a group for talking about the papers. Each group has three or four people, and you should discuss each paper sometime before class meets. For each group, one member should writeup a summary of the discussion and post on Canvas **before** class. This counts for 10% of the course grade.

Before you read the first paper assigned to this class, please read: **How to Read a Paper (http://ccr.sigcomm.org/online/files/p83-keshavA.pdf)** .

Operating Systems Background Materials

- Here is a link to **Operating Systems In Three Easy Steps (OSTEP) (http://pages.cs.wisc.edu/~remzi/OSTEP)** book.
- **Lectures (https://www.cse.iitb.ac.in/~mythili/os)** on Operating Systems (with videos)

Other technology requirements / equipment / material

Programming assignments will be a significant part of this course, so access to a computer is required.

Course Requirements and Assignments

I do not grade on a curve. The exams and projects measure what you are expected to have learned. There aren't many opportunities for extra credit apart from potential bonus questions on exams.

Programming Project: We will be doing **individual** programming assignments**. Individual programming assignments are not group projects.** If students get help on assignments, even to resolve a seemingly trivial problem, it must be documented in the code with the name of the person rendering the help and a brief description of the help provided. Extensive help on a project will result in a reduced grade. Failure to document help, or any other forms of cheating will result in a failing grade on the assignment at a minimum and may result in failure of the course. See **http://info.sjsu.edu/static/schedules/integrity.html (http://info.sjsu.edu/static/schedules/integrity.html)** for more information. Even in open source, you cannot copy code from one open source project to another without attribution.

Programming assignments will be distributed via Github. **Here (https://github.com/SJSU-CS-249/dslabs-sp26)** is the Github repo for this course. Please create a Github account (if you don't already have one). We will use Gradescope for assignment submission and grading.

The labs for this class are known to be demanding. While the number of lines of code you write will be small (on the order of a few hundreds), debugging distributed systems is hard. Get started on the labs early.

Late submission:

1. Submissions that are up to 5 days late will be penalized 10% per day.
2. Submissions that are up to two weeks late will be penalized 60% of the score.
3. I **will not** accept submissions that are more than two weeks beyond the submission date. This will be considered as a non-submission.
4. At my discretion, non-submissions will likely result in a grade penalty.
5. There will be a 50% penalty for late submission on the Paper Reading group discussion writeup.

Class Participation: You will be expected to read the papers we are discussing before we discuss them in class. For each class, I will randomly select a group to lead the discussion about the paper being discussed in class. To get credit for discussing a paper, each group will be expected to write a short summary about the paper on Canvas. Short answers such as "Yes" or "I agree" or "I was thinking the very same thing" will not count as well.

Help on Labs: Nainoa Faulkner-Jackson (course grader) will have weekly **in-person** "**lab hours (https://sjsu.instructure.com/courses/1595954)**" in to help you with the labs. Please seek his help whenever needed.

Grading Information

Grades will be calculated based on the individual project grades, the two mid-semester exams, the final, discussion participation. Please see **Grading Policy (https://sjsu.instructure.com/courses/1619545/pages/grading-policy)** for how you will be graded in this course. Briefly, the weighted distribution is

Programming Project	40%
Midterm 1	15%
Midterm 2	15%
Final	20%
Paper discussion/participation	10%

The **University Policy S16-9 (http://www.sjsu.edu/senate/docs/S16-9.pdf)**, Course Syllabi requires the following language to be included in the syllabus:

“Success in this course is based on the expectation that students will spend, for each unit of credit, a minimum of 45 hours over the length of the course (normally three hours per unit per week) for instruction, preparation/studying, or course related activities, including but not limited to internships, labs, and clinical practica. Other course structures will have equivalent workload expectations as described in the syllabus.”

Final Examination or Evaluation

This course will have a cumulative final exam given during **exam week (https://www.sjsu.edu/classes/final-exam-schedule/spring-2025.php)**. The final exam scheduled for this class is on 5/19/26 from 1:00-3:00pm

There will be three in-class exams given in the semester (the last being the final exam).

Classroom Protocol

The schedule listed below is tentative and may change based on student needs.

This is your class. Please ask questions. Please come prepared. Do not engage in activity that may distract other students.

I do not take attendance except for the first two classes. Students not attending either of the first two classes will be dropped to make room for students on the waiting list. Attempting to get marked as present (by have someone else attend in your place or using technological deceptions) will be considered academic dishonesty and at a minimum will result in you getting dropped from the course.

Please turn off phones and close your laptops before start each of class.

Teaching Philosophy

I am passionate about teaching and helping students prepare for whatever is next (i.e., career in the industry, continue with Graduate studies, etc.). This means that I expect students to actively participate in class, work independently, and seek help from the course staff when needed.

Coursework will typically entail experiential learning: you learn by building working systems. Assignments will be distributed via Github. You should familiarize yourself with git and other software development tools.

University Policies

Per University Policy S16-9, university-wide policy information relevant to all courses, such as academic integrity, accommodations, etc. will be available on Office of Graduate and Undergraduate Programs' **Syllabus Information web page (http://www.sjsu.edu/gup/syllabusinfo)** Make sure to review these policies and resources.

Programming Assignments

In this course you will gain practical experience with distributed systems by building a highly available, scalable, fault tolerant, and transactional key-value store. Key-value stores are widely used in cloud computing.

For the projects, we will use the **DSLabs (https://ellismichael.com/dslabs)** framework from University of Washington. See **Programming Assignments (https://sjsu.instructure.com/courses/1619545/pages/programming-assignments)** page for more details.

This programming project comprises of **individual** programming assignments**. Individual programming assignments are not group projects.** The programs you submit must be your work

and your work alone.

Caveats:

1. A requirement is that you **do not** publish solutions on public GitHub repos.
2. The labs are known to be demanding and debugging can be time consuming. So, please get started early for each lab.

Cheating/Academic Dishonesty

I take issues of Academic Dishonesty very seriously. ***Do not cheat***. If we detect cheating in a programming assignment or an exam, you will get a **0** for that lab/exam. Repeat offense will likely lead to a **F** grade in the course.

All exams will be in-class. You are not allowed to use any phones during the exam.

Artificial intelligence (AI) tools like ChatGPT, Google Gemini, and GitHub Copilot are not permitted to be used as a replacement for the writing or problem-solving components of this class. SJSU's subscription to Turnitin has an AI-detection feature, and assignments that have been determined by that application or by other convincing evidence to have been written by AI in substantial fractions will receive an automatic zero. The incident will also be reported to the University as academic misconduct.

Acknowledgements

This course uses **DSLabs framework developed by Ellis Micheal and others (https://ellismichael.com/dslabs)** from the University of Washington. Additionally, this course adapts (and also uses) materials from courses taught by **Tom Anderson, Doug Woos, Ellis Micheal, and Arvind Krishnamurthy (https://courses.cs.washington.edu/courses/cse452)**, **Michael Freedman and Kyle Jameson (https://www.cs.princeton.edu/courses/archive/fall17/cos418/syllabus.html)**, **Lorenzo Alvisi (https://www.cs.cornell.edu/courses/cs5414/2024sp)**, and **Ben Reed (https://www.sjsu.edu/people/ben.reed)**. Thank you!

Copyright

taken against the student or other legal action against an outside entity. Additionally, the sharing of class materials without the specific, express approval of the instructor may be a violation of the University's Student Honor Code and an act of academic dishonesty, which could result in further disciplinary action. This includes, among other things, uploading class materials to websites for the purpose of sharing those materials with other current or future students.

CS 249-01, Distributed Computing, Spring 2026

Course Schedule (Tentative)

Week	Date	Topic/Theme	Reading(s)
		Pre-read	1. How to Read a Paper (http://ccr.sigcomm.org/online/files/p83-kesh 2. DSLabs Overview (https://ellismichael.com/papers/dslabs-eurosys 3. Setup DSLabs (https://github.com/SJSU-CS-249/dslabs-sp26/tre (https://github.com/SJSU-CS-249/dslabs-sp25) on your laptop and c (https://github.com/SJSU-CS-249/dslabs-sp26/tree/main/labs/lab0-pi
1	1/22	Introduction	1. Introduction to Distributed System Design (http://www.hpcs.cs.tsukuba.ac.jp/~tatebe/lecture/h23/dsys/dsd-tuto 2. Why Do Computers Stop - Jim Gray (http://pages.cs.wisc.edu/~remzi/Classes/739/Fall2017/Papers/gray-v 85.pdf)
2	1/27	Remote Procedure Call	1. (g)RPC: ◦ gRPC Documentation: Introduction (https://grpc.io/docs/what- Core Concepts (https://grpc.io/docs/what-is-grpc/core-concept ◦ gRPC Introduction Slides (https://docs.google.com/presentati _4dwBMLyqfwchvS6iXtbclSQPLAXL6gSYOcc/edit#slide=id.g1c2 2. Setup gRPC on your laptop and run HelloWorld (https://grpc.io/docs/languages/java/quickstart)
2	1/29	Replication	1. VMWare vSphere 4.0 (https://www.cs.princeton.edu/courses/archive/fall16/cos418/papers 2. (Optional) State Machine Replication (https://www.cs.cornell.edu/fbs/publications/SMSurvey.pdf)
3	2/3	Time in a Distributed System	1. Lamport Clock (https://amturing.acm.org/p558-lamport.pdf) 2. (Optional) Knowledge and Common Knowledge (https://www.cs.cornell.edu/courses/cs5414/2017fa/papers/p549-halp Section 4)
			1. Distributed Snapshots (http://www.cs.wisc.edu/areas/os/Qual/pap

3	2/5	Causality	2. Vector Clocks (https://www.vs.inf.ethz.ch/publ/papers/VirtTimeGlol Sections 1, 6, and 7) 3.
4	2/10	View changes and reconfiguration	1. View stamped Replication Revisited (https://pmg.csail.mit.edu/p 2. (Optional) Morning Paper blog on View Replication Revisited (https://blog.acolyer.org/2015/03/06/viewstamped-replication-revisite 3.
4	2/12	Distributed Recovery	1. Non-blocking Two phase commit (https://dl.acm.org/doi/pdf/10.1 2. Book chapter on Two phase commit (Read Ch. 7.1-7.4) (https:/ us/research/wp-content/uploads/2016/05/chapter7.pdf)
5	2/17	Caching and Consistency Models	1. Leases (https://dl.acm.org/doi/pdf/10.1145/74851.74870) 2. (Optional) On Interprocess Communication (Read Sec. 4) (https://lamport.azurewebsites.net/pubs/interprocess.pdf)
5	2/19	Shared Log	1. Shared Memory Consistency Models (https://courses.grainger.illinois.edu/CS533/sp2023/reading_list/adv (Classical paper: Read upto Sec. 6) 2. Linearizability (https://cs.brown.edu/~mph/HerlihyW90/p463-herlih 3. CORFU (https://www.usenix.org/system/files/conference/nsdi12/
6	2/24	**Midterm I**	
6	2/26	Consensus	1. Google Tech Talk on Paxos (short) (https://www.youtube.com/wa 2. (Optional) Paxos Made Simple (https://lamport.azurewebsites.net/ 3. (Optional) The Part-time Parliament (https://lamport.azurewebsites.net/pubs/pubs.html#lamport-paxos)
7	3/3	Paxos Contd.	1. View stamped Replication Revisited (https://pmg.csail.mit.edu/p (Read Section 4.2 and Section 4.3) 2. (Optional) CAP Theorem (http://www.julianbrowne.com/article/brev
7	3/5	More Consensus	1. Paxos Made Moderately Complex (overview (https://paxos.systen (https://www3.cs.stonybrook.edu/~liu/cse535/vanRenesse-paxos-12 one once before class. Refer to these as you work on Lab 3. 2.
			1. RAFT USENIX ATC'14 presentation (https://www.youtube.com/w

8	3/10	Even more Consensus	(short) 2. (Optional) RAFT Tutorial (https://www.youtube.com/watch?v=vYp4
8	3/12	RAFT Contd. and Update Propagation	1. RAFT (https://web.stanford.edu/~ouster/cgi-bin/papers/raft-atc14) 2. Chain Replication (http://www.cs.cornell.edu/home/rvr/papers/OSD
9	3/17	Byzantine Fault Tolerance	1. Super cool 8-min video on BFT and Blockchain (https://www.yo v=_fgW2lM6ctM) 2. Practical BFT (https://www.cs.princeton.edu/courses/archive/fall17 3. (Optional) The Byzantine Generals Problem (https://lamport.azurewebsites.net/pubs/pubs.html#byz)
9	3/19	Distributed Hash Tables	1. Chord (https://pdos.csail.mit.edu/papers/ton:chord/paper-ton.pdf) 2. (Optional) Consistent Hashing Blog Post (https://arpitbhayani.m hashing) 3. (Optional) Tail at Scale (https://cseweb.ucsd.edu/classes/sp18/cse dean.pdf)
10	3/24	Eventually Consistent Systems	1. Amazon-Dynamo (http://s3.amazonaws.com/AllThingsDistributed/ sosp2007.pdf) 2. Bayou (http://www.news.cs.nyu.edu/~jinyang/ds-reading/bayou-cor
11	3/26	Zookeeper	1. Zookeeper (https://www.usenix.org/legacy/event/atc10/tech/full_
	3/31	**No Class: Spring break**	
	4/2	**No Class: Spring break**	
11	4/7	Memcache	1. Scaling Memcache at Facebook (https://www.usenix.org/confere sessions/presentation/nishtala)
12	4/9	**Midterm II**	
12	4/14	Scalable Filesystems	1. The Google Filesystem (https://static.googleusercontent.com/media/research.google.com/e sosp2003.pdf) 2. (Optional) The Quantcast Filesystem (http://www.vldb.org/pvldb/

			ovsiannikov.pdf)
13	4/16	Big Table	1. Google's BigTable (https://static.googleusercontent.com/media/research.google.com/e osdi06.pdf) 2. (Optional) Building Large Scale Internet Services (http://pages.cs.wisc.edu/~remzi/Classes/739/Fall2017/Papers/dean-
13	4/21	Spanner	1. Google's Spanner (https://static.googleusercontent.com/media/research.google.com/e osdi2012.pdf)
14	4/23	Spanner Contd and Dynamo DB	1. DynamoDB Tech Talk at USENIX (https://www.usenix.org/conference/atc22/presentation/elhemali) 2. (https://www.usenix.org/conference/atc22/presentation/elhemali) Amazon DynamoDB (https://www.usenix.org/system/files/atc22-el
14	4/28	Firestore	1. Google's Firestore (https://research.google/pubs/firestore-the- database-for-the-application-developer) 2. Short Video explaining CAP Theorem (https://www.youtube.com 3. Associated Papers: CAP Theorem (https://en.wikipedia.org/wiki/CAP_theorem)
15	4/30	Cluster Resource Management: YARN, Borg	1. Apache Hadoop YARN: Yet Another Resource Negotiator (https://www.cse.ust.hk/~weiwa/teaching/Fall15-COMP6611B/reading 2. Borg, Omega, and K8s: Lesson Learned (https://static.googleusercontent.com/media/research.google.com/e
	5/4	Storage support for AI: Cachelib	Cachelib Caching Engine (https://www.usenix.org/conference/osd
	5/6	Object Stores	1. Facebook's Tectonic Filesystem (https://www.usenix.org/system
15	5/8	**No Class**	Papers you may find interesting to read over Summer: 1. (Optional) Hints for Computer Design (https://www.microsoft.co content/uploads/2016/02/acrobat-17.pdf) 2. COPS Talk at SOSP'11 (https://www.youtube.com/watch?v=jh9

			3. **COPS (https://www.cs.princeton.edu/courses/archive/spring22/cos** 4. **Apache Iceberg (https://iceberg.apache.org)** 5. (Optional) **Apache Ozone (https://ozone.apache.org)**
	5/19/26	**Final Exam and Wrapup**	1-3pm