CS-438

Decentralized systems engineering

Borsò-Tan Pierluca, Ford Bryan Alexander

Cursus	Sem.	Type
Computer science minor	H	Opt.
Computer science	MA1, MA3	Obl.
Cyber security minor	H	Opt.
Cybersecurity	MA1, MA3	Obl.
SC master EPFL	MA1, MA3	Opt.

Language of teaching	English
Credits	8
Session	Winter
Semester	Fall
Exam	Written
Workload	240h
Weeks	14
Hours	**6 weekly**
Lecture	2 weekly
Exercises	2 weekly
Project	2 weekly
Number of positions	

Summary

A decentralized system is one that works when no single party is in charge or fully trusted. This course teaches decentralized systems principles while guiding students through the engineering of their own decentralized system featuring messaging, file sharing, encryption, and blockchain concepts.

Content

- Addressing, Forwarding, Routing. Peer-to-peer communication.
- Information gossip. UseNet: technical, security, and social lessons. Randomized rumor-mongering and anti-entropy algorithms.
- Trust and Reputation. Authorities, trust networks. Sybil attacks and defenses.
- Naming and search. Request flooding. Hierarchical directories, landmark routing. Self-certifying identities. Distributed hash tables.
- Distributed consensus, distributed ledgers (blockchains), and cryptocurrencies.
- Anonymous Communication. Onion routing, mix networks. Dining cryptographers. Voting, verifiable shuffles, homomorphic encryption. Anonymous disruption.
- Fireproofing Alexandria: Decentralized Storage. Replication. Parity, erasure coding. Renewal. Digital preservation.
- Content Distribution. Opportunistic caching (FreeNet). Content integrity: hash trees, hash file systems. Convergent encryption. Swarming downloads: BitTorrent. Free-riding, incentives.
- Gaining perspective. Spam, malicious content. Review/moderation and reputation systems. Leveraging social networks (Peerspective). Balancing local and global viewpoints.
- Decentralized Collaboration. Network file systems, version management. Consistency.
- Consistency Models. Disconnected operation, eventual consistency, conflict resolution.
- Distributed Consensus. Paxos. Accountability (PeerReview). Byzantine fault tolerance.
- Mobile Code. Smart contract systems. Privacy: trusted computing, fully homomorphic encryption. Decentralized virtual organizations.
- Securing Decentralized Systems: threat modelling, use of threshold cryptosystems.
- Going into production: Quality assurance, chaos engineering and operations.

Learning Prerequisites

Required courses

- CS-202 Computer systems or COM-208 Computer networks

- COM-301 Computer security and privacy

Learning Outcomes

By the end of the course, the student must be able to:

- Implement decentralized systems via hands-on coding, debugging, and testing
- Design effective testing strategies for distributed and decentralized systems
- Design practical distributed and decentralized systems

Transversal skills

- Set objectives and design an action plan to reach those objectives.
- Identify the different roles that are involved in well-functioning teams and assume different roles, including leadership roles.
- Assess progress against the plan, and adapt the plan as appropriate.
- Take account of the social and human dimensions of the engineering profession.
- Continue to work through difficulties or initial failure to find optimal solutions.

Teaching methods

Lectures: The course's lectures will present and discuss challenges, known techniques, and open questions in decentralized system design and implementation. Lectures will often be driven by examination of real decentralized systems with various purposes in widespread use the past or present, such as UseNet, IRC, FreeNet, Tor, BitTorrent, and Bitcoin. Throughout the course we will explore fundamental security and usability challenges such as decentralized identification and authentication, denial-of-service and Sybil attacks, and maintenance of decentralized structures undergoing rapid changes (churn).
Labs: During the semester, students will develop a small but usable peer-to-peer communication application that reflects a few of the important design principles and techniques to be explored in the course, such as gossip, distributed hash tables, consensus algorithms, and cryptocurrencies. The labs will designed so that solutions can initially be tested individually on private, virtual networks running on one machine, then tested collectively by attempting to make different students' solutions interoperate on a real network.

Warning: This course is extremely programming-intensive. Students should be strong and confident in their programming skills in general, and be willing to spend substantial time outside of class debugging difficult distributed concurrency bugs and other challenges. TAs will be available to help at the exercise sessions, but *they will not solve your problems or debug your code for you*.

Expected student activities

Students will be expected to attend lectures to understand the concepts needed for the course, but the main workload will be actual hands-on programming assignments, which the students will perform on their own during the first part of the course and in teams during the final project-oriented part of the course.

Assessment methods

- Programming assignment grading (evaluating function, performance, correctness and implementation quality): 40%
- Team project grading (evaluating for scope, implementation quality, testing, evaluation, documentation and report): 30%
- Written exam: 30%

Resources

Virtual desktop infrastructure (VDI)
No

Moodle Link
- https://go.epfl.ch/CS-438