CS-438

Decentralized systems engineering

Ford Bryan Alexander

Cursus	Sem.	Type
Computer science	MA1, MA3	Opt.
SC master EPFL	MA1, MA3	Opt.

Language of teaching	English
Credits	6
Session	Winter
Semester	Fall
Exam	During the semester
Workload	180h
Weeks	14
Hours	**6 weekly**
Courses	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 development and testing of their own decentralized system incorporating messaging, encryption, and blockchain concepts.

Content

- Networking Foundations. Addressing, forwarding, routing. Client/server versus peer-to-peer communication. Firewalls, NATs, traversal.
- Gossip: a foundation for decentralized systems. UseNet: technical, security, and social lessons. Randomized rumor-mongering and anti-entropy algorithms.
- Communicating Securely: Basic Cryptographic Tools. Symmetric-key encryption. Hash functions, message authentication. Diffie-Hellman key exchange. Public-key encryption, digital signatures.
- Trust and Reputation. Authorities, trust networks. Sybil attacks and defenses.
- Naming and search. Request flooding. Hierarchical directories, landmark structures. Self-certifying identities. Distributed hash tables.
- Distributed consensus, distributed ledgers (blockchains), and cryptocurrencies.
- Following a Moving Target. Location services, reference points, forwarding. Composite identities.
- 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.
- Pessimistic locking. Disconnected operation, eventual consistency, conflict resolution.
- Distributed Consensus. Paxos. Accountability (PeerReview). Byzantine fault tolerance.
- Mobile Code and Agents. Privacy: trusted computing, fully homomorphic encryption. Decentralized virtual organizations.

Keywords

distributed systems, decentralized systems, security, privacy, anonymity, cryptography, gossip, consensus, swarming, blockchain, cryptocurrency

Learning Prerequisites

Required courses

- COM-208: Computer networks

Recommended courses

- CS-206 Parallelism and concurrency
- COM-301: Network security
- CS-323: Introduction to operating systems

Important concepts to start the course

Students must already be highly competent at programming and debugging in a high-level systems programming language such as Java, C#, or Go. Programming exercises will be in Go, but students already well-versed and experienced in programming with comparable systems languages should be able to pick up Go during the course.
Students should have both solid foundational knowledge of how networks function, and some experience actually writing network programs, e.g., TCP/IP programming using the Sockets API.

Learning Outcomes

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

- Design practical distributed and decentralized systems
- Implement systems via hands-on coding, debugging, and interoperability testing

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, social trust networks, 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.

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 optionally in small teams during the final project-oriented part of the course.

Assessment methods

- Lecture attendance: 10%
- Programming assignment grading (evaluating both function and student documentation): 50%

• Final project grading (accounting for both scope, appropriateness, and follow-through in implementation quality and documentation): 40%

Supervision

Office hours	Yes
Assistants	Yes
Forum	Yes