Distributed * Systems
CPS 512
Fall 2015

Instructor: Jeff Chase

This offering of CPS 512 will focus on core concepts in distributed systems, using geo-distributed mega-services in the cloud as a motivation and driving example. Well-designed cloud applications are layered above common service platforms that handle the hard problems: tracking groups of participating servers (views), distributing state and functions across a group, coordinating control and ownership of data, managing consensus, and recovering from server and network failures. The course focuses on the design of these service platforms and their abstractions.

Although the course covers the fundamentals, the emphasis is on practical technologies and their limitations. This course includes an important software technology component. We will do some projects using the Scala language and the Akka actor system for scalable "reactive" applications. Programming labs include a key-value application, a leased lock service, atomic transactions, and a consensus service. There will also be a course project of your choosing.

The course divides loosely into three parts. The **first part** covers basic problems and abstractions, focusing on elastic key-value stores as an example. In the **second part**, we dive deeper into foundational distributed systems topics that underlie these systems: quorum replication, logical time and causality, eventual consistency with vector clocks, views and leader election, and consensus.

The **third part** shifts focus to secure Internet-scale systems with multiple identities and federation, showing how cryptosystems are used to manage naming, identity, and authorization. We introduce trust logic as a formalism for building secure networked systems, and use it to represent the Internet security architecture and cloud access control.

The readings for the course are research works: there is no textbook. The papers include some tutorial and survey works, with a handful of full-strength research papers, including systems papers from major cloud services companies (e.g., Amazon and Google).

Here is a general outline of topics and readings for the course. We may make minor changes and adjustments as the semester progresses. Each section corresponds to a course unit and each bullet corresponds roughly to one class.

Coordination and failures

Structure of distributed systems: clients and servers, multi-tier services, geo-distributed mega-services. Messaging, failure models, and the problem of network partitions. State and the data storage tier: "SQL, NoSQL, and NewSQL". The consensus problem: consistency, availability, and the CAP theorem. Consistency models: ACID transactions, BASE and eventual consistency. Introduction to the Scala/Akka actor model.

- **Course intro**.
- **CAP: Consistency vs. availability in networked systems**. ACID transactions, two-phase commit, and the blocking problem. Reading: *CAP Revisited* [3], *Your Coffee Shop Doesn't Use Two-Phase Commit* [9].
- **Leases and failure detection**. Use of leases for cache consistency. Optional reading: *Practical Uses of Synchronized Clocks in Distributed Systems* [12] (Sections 1, 5, 8).

- **Working with Scala/Akka actors** (guest speaker: Dr. Qiang Cao)
- **The Chubby lock service**. Replication, server failure, and primary election. A first look at the Paxos state machine. Reading: Chubby [4].
- **Data consistency**. Overview of strong and weak models. Serializable transactions, single-copy semantics, and eventual consistency. Reading: *Eventual Consistency Today* [1], *Concurrency Control and Recovery* [8] (through 3.1.1; but we won't talk much about buffering as in 2.2.2).

Elastically scalable key-value service

The challenges of scale: partitioning/sharding and replication in the data tier. We use Amazon's Dynamo as an example, although many systems use similar techniques. Riak is an open-source key-value store based on Dynamo. Reading: Dynamo [7]. Optional reading: A Little Riak Book.

- **Consistent hashing**. The problem of churn. Balancing and rebalancing load. Service performance.
- **Data replication and quorum consistency**. Primary/backup, write propagation, server failure and reconciliation. Availability and sloppy quorums. Versioning and conflicts. Optional reading: *Quantifying Eventual Consistency with PBS* [2].
- **Mobile computing** (9/24). Guest speaker: Dr. Landon Cox. Application partitioning, cloud offload, and disconnected operation.

Time, clocks, and causality

How to order concurrent events in a distributed systems with no centralized primary? Logical clocks, version vectors/vector clocks, causal ordering and causal communication. Asynchronous replication, anti-entropy, and update conflicts in Bayou.

- **Concurrency and logical time**. Logical/Lamport clocks, causality, and the origins of state-machine replication. Reading: *Time, Clocks, and the Ordering of Events in a Distributed System* [10].
- **Vector clocks and causal ordering**. Replica reconciliation by anti-entropy exchanges. Update conflicts and reordering. Reading: Bayou [15]. Optional reading: Bayou's flexible update propagation [16].
- More Bayou, causal consistency, causal message ordering, and consistent snapshots.
- **Midterm Exam 10/6. No class 10/13 (Fall break).**
- **Conflict-free replicated data types (CRDTs)**. CALM and "ACID 2.0". Reading: TBD.
- **Strong consistency and time revisited.** Reading: Spanner [6].

Consensus

We spend a few days discussing consensus in theory and in practice. A safe, live consensus algorithm is the cornerstone of reliable distributed systems. But it is impossible to build one! So this is a study of what works in practice.

- **Foundations of consensus**: FLP impossibility result, hierarchical systems and consensus for master election, tolerating failures in practice.
- **Consensus by design**. Reading: RAFT [14]. Optional reading: *How to Build a Highly Available System Using Consensus* [11], *Paxos Made Moderately Complex* [17] (see paxos.systems).
- **Byzantine consensus (BFT)**. *Reading*: *From Viewstamped Replication to Byzantine Fault Tolerance* [13].
- **Midterm exam 11/3**

Trust in networked systems

- Cryptosystem review and introduction to trust logic
- **Logic of authentication**. Identity services. Reading: BAN Logic [5].
- **Secure DNS naming**. DNSSEC and the CA hierarchy in trust logic
- **Authorization** in IaaS clouds and federated systems: Amazon's IAM and SFS. Reading: TBD.
- Secure routing and/or other topics here, e.g., bitcoin blockchain

References

[1] P. Bailis and A. Ghodsi. Eventual consistency today: Limitations, extensions, and beyond. *Commun. ACM*, 56(5):55–63, May 2013. [local pdf].

[2] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica. Quantifying eventual consistency with PBS. *Communications of the ACM*, 57(8):93–102, Aug. 2014. [local pdf].

[3] E. Brewer. CAP twelve years later: How the "rules" have changed. *Computer*, 45(2):23–29, February 2012. [local pdf].

[4] M. Burrows. The Chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 335–350. USENIX Association, May 2006. [local pdf].

[5] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computing Systems (TOCS)*, 8(1):18–36, 1990. [local pdf].

[6] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, et al. Spanner: Google's globally distributed database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):8, 2013. [local pdf].

[7] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *SOSP '07: Proceedings of 21st ACM SIGOPS Symposium on Operating systems Principles*, pages 205–220, New York, NY, USA, 2007. ACM. [local pdf].

[8] M. J. Franklin. Concurrency control and recovery, 1997. [local pdf].

[9] G. Hohpe. Your coffee shop doesn't use two-phase commit. *IEEE Software*, 22(2):64–66, Mar. 2005. [local pdf].

[10] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978. [local pdf].

[11] B. W. Lampson. How to build a highly available system using consensus. In *Distributed Algorithms*, pages 1–17. Springer, 1996. [local pdf].

[12] B. Liskov. Practical uses of synchronized clocks in distributed systems. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '91, pages 1–9, New York, NY, USA, 1991. ACM. [local pdf].

[13] B. Liskov. From viewstamped replication to byzantine fault tolerance. In *Replication*, pages 121–149. Springer, 2010. [local pdf].

[14] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the USENIX Annual Technical Conference*, pages 305–320, June 2014. [local pdf].

[15] K. Petersen, M. Spreitzer, D. Terry, and M. Theimer. Bayou: Replicated database services for world-wide applications. In *Proceedings of the 7th ACM SIGOPS European Workshop: Systems Support for Worldwide Applications*, pages 275–280. ACM, September 1996. [local pdf].

[16] K. Petersen, M. J. Spreitzer, D. B. Terry, M. M. Theimer, and A. J. Demers. Flexible update propagation for weakly consistent replication. In *SOSP '97: Proceedings of the Sixteenth ACM Symposium on Operating systems Principles*, pages 288–301, New York, NY, USA, October 1997. ACM. [local pdf].

[17] R. Van Renesse and D. Altinbuken. Paxos made moderately complex. *ACM Computing Surveys*, 47(3):42:1–42:36, Feb. 2015. [local pdf].