Guide to Design and Implementation of Distributed Applications (DIDA) 2024/2025

Version 1.0

Luís Rodrigues

July 2024

Abstract

This document makes a brief overview of the topics covered in the classes. It aims at being a starting point to help you to follow the course

1 Course Content and Goals

This course addresses theory and practice of distributed systems. It covers a number of techniques that are the key ingredients of many modern distributed systems but that could not be addressed in detail in the BSc course on "Distributed Systems". We cover topics suc as: how to achieve consistency in an asynchronous system, how to reconfigure a system, how to support transactions in a distributed system, and how to build peer-to-peer applications. Major cloud providers and cloud users such as Google, Amazon, Facebook, etc have developed and deployed real systems based on these concepts. In fact, we address a few concrete examples of such systems (including Chubby, Spanner, and Dynamo).

Many former IST students are now working for such companies and contributing to the design of cutting-edge distributed systems. But even if you do not go work for Google or Amazon, you are likely to use the cloud services that are provided by these companies. To use those services wisely you need to understand their limitations. In "DIDA" we aim at providing you the knowledge to differentiate key technical features from marketing hype.

2 Studying Materials

The book "Distributed Systems: Concepts and Design (Fifth Edition): George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair 2012 Pearson Custom Publishing" is very good but a bit outdated. Still, for several topics (for instance, view-synchrony, peer-to-peer systems, background on concurrency control, some early Google tech) it is still an excellent reference.

Unfortunately, because nobody buys books, nobody writes books anymore. As a result, for the more recent topics, there are no good books and you will have to study by reading papers. I have listed a number of relevant papers in the web page of the course. Some papers are more important than others. Later in the text, I will point which papers are the most relevant for the exam.

On the positive side, given that the topics that we cover are really important, it is easy to find study material in the Internet. If you google you can quickly find a myriad of slides, videos, and other learning material covering things like Paxos, Zookeeper, etc.

I have also prepared a collection of "Sample Questions" that illustrate the type of questions you may expect to find in the exam. Make sure you read those questions.

I will be also make available some notes regarding the content of each class in the form of "slides". I may not use these slides in the class and they are very sketchy by design: the goal is not replace the book or the papers by the slides. They merely serve as log of the topics that have beed addressed during the class. I do understand that, in many cases, students use the slides as the only source of knowledge. However, using just the slides, offers a very shallow understandanding of the topics covered in the course.

3 On the Role of "Theory" Classes

My perspective on teaching is that the role of "Theory" classes is *not* to try to expose the students to *every* piece of information that need to known by the end of the semester. First, there is a limit to the amount of information one can absorb in a 90m or 120m slot. Then, learning is a result of theory classes, practical classes, exercises, and *a lot* of self-study.

Instead, in my classes I try to focus on those aspects that I find more challenging or intriguing in the topics covered by the course. I try to explain with detail the parts that I find the hardest and just motivate and make students aware (and hopefully curious) for other aspects that are covered in more detail in the bibliography. So, the bottom line is that you should look at the topics covered in the lectures just as a starting point for the study.

Students must also be aware that the idea that there is a single, optimal, solution for every problem is often illusive. It would be probably easier for me, and for you, if I would arrive at a class a present a given solution without discussing alternatives, their limitations and their drawbacks, and omitting that there are many other solutions out there. However, I believe that is not enough. Of course, it is important to learn a solution that works: you need that background to solve the problems you will be faced in the future. But often, it is equally important to understand why some other alternatives do not work! Sooner or later, someone will try to convince you to use the wrong solution and it will be *your* job to stop them and avoid bad mistakes. I did my best to encourage you to think critically about the problems and their solutions.

4 Summary of Lectures

Lecture 1: Introduction to the Course

- *Summary:* Short Introduction to the course.

Lectures 2 to 4: Consensus, Paxos and Muti-Paxos

- *Clickbait:* We agreed that one of us would decide and then we got into a fight. This is what happened!
- *Summary:* Paxos is one of the most known protocols to solve consensus in a partially synchronous system (in particular, in the case of Paxos, in a system where messages can be delayed but where eventually a leader can be elected and a majority of nodes remains connected to the leader). These lectures described Paxos operation. An optimisation of Paxos, known as multipaxos have also been addressed.
- *Things to remember:* Why the several phases of Paxos are needed. How is safety ensured when there are multiple leaders trying to commit a value concurrently. What is the mechanisms that ensures that, once a value is committed, any future leader will adopt that same value.
- *Why should I care?* It is hard to get a job in the silicon valley if you do not know what Paxos is.
- *Must read:* "Paxos made Simple" is probably the better source to understand Paxos. If you did the project, you are probably familiar with Paxos at this point. Also, there is a lot of material about Paxos in the Internet.
- *Further reading:* It is worth reading at least the introduction of the original "The Part-Time Parliament", at least to understant where the work "Paxos" comes from. The paper "Paxos made live" addresses the challenges of implementing Paxos in a real system. Not required for the exam, but very good to understand the difference between theory and practice.

Lecture 5: Coordination Services

- *Clickbait:* Paxos is very complex. Let's use something equally complex instead!
- *Summary:* Some applications do not require Paxos or use just too many workers (making hard to run Paxos among all the replicas). A coordination service is a service that helps to build fault-tolerant applications without using Paxos directly. The coordination service is usually implemented by a (small) set of server that run some Paxos variant.
- *Things to remember:* Chubby and ZooKeeper are two of the most important coordination services. Chubby provides strong consistency to clients. Unfortunately, the source code is proprietary and not available. ZooKeeper is open source but provides weaker guarantees. People have build layers on topy of ZooKeeper to make it look more like Chubby.
- *Why should I care?* It is important to understand the advantages and limitations of ZooKeeper, given that many other projects use ZooKeeper as a building block.
- *Must read:* The "Chubby" and "ZooKeeper" papers.
- *Further reading:* The links to "Curator" are useful to understand the limitation of ZooKeeper and also to better understand how ZooKeeper can be used.

Lecture 6: Group Communication and View-Synchrony

- *Clickbait:* My life was in a state of confusion: I was living in a group without a view.
- *Summary:* Introduction to group communication. View-synchrony as an abstraction to manage the complexity of dynamic reconfiguration. Services and properties.
- *Things to remember:* How can we define multicast reliability (i.e., deliver a message to "all" group members) in a setting where the number of group members may change dynamically (i.e., "all" is a moving target)? View synchrony provides an answer to this problem.
- *Why should I care?* Many practical system have been built using this abstraction, from stock exchange to traffic control systems.
- *Must read:* "Distributed Systems: Concepts and Design" is probably the best source to understand view-synchrony and what system like ISIS provided.
- *Further reading:* The book chapter named "A History of the Virtual Synchrony Replication Model", written by the inventor of the view-synchrony concept, provides a historical perspective of what view-synchrony was trying to achieve and why it failed to be a commercial success in the long run. You probably can skip this for the exam, but is an interesting reading, as it shows that many things that we discuss in this course, and are now well understood, have been a source of controversy for many years, among the most distinguished scholars!

  "Chain Replication" is an example of a system that supports reconfiguration in a setting similar to that of view-synchrony, by that relies on a coordination service to install the views.

Lecture 7: Reconfiguring Paxos

- *Clickbait:* Stopping is hard when you drive at full speed, even with Paxos! Drive safely.
- *Summary:* The original view-synchronous supported reconfiguration (aka, view-changes) in systems using a perfect failure detector. This class discusses how you can reconfigure a system in face of asynchrony.
- *Things to remember:* Why a special version of Paxos may be needed to stop a state-machine and why deriving such version ("stoppable Paxos") is trickier than it might look at a first look. How to reconfigure a storage service running ABD with the help of a coordination service.
- *Why should I care?* Old machines need to be decommissioned. New machines need to be added to replace old machines. A static configuration is not very useful in practice.
- *Must read:* "Reconfiguring a State Machine" not only describes different techniques that can be used to reconfigure a Paxos deployment but also provides some light about the relation between reconfigurable Paxos and view-synchrony. "Reconfiguring Replicated Atomic Storage: A Tutorial" is excellent but a bit dense: sections 1-5 are a must read.
- *Further reading:* "High Throughput Replication with Integrated Membership Management" is a good example of how you can pick the idea of chain-replication and make it work in an asynchronous system, where Paxos is used to avoid the problems that may occur when different nodes believe to be in different configurations.

  ABD provides atomic registers and does not require consensus. It would be good to be able to reconfigure a storage system without using consensus. The last sections of 'Reconfiguring Replicated Atomic Storage: A Tutorial" 'present a solution that allows to achieve this goal. The protocol is somehow complex, was not discussed in classes, and will not be addressed in the exam.

Lecture 8: RAFT

- *Clickbait:* How many variants of Paxos do we need?
- *Summary:* RAFT is an attempt to make a "better" Paxos. One annoying aspect of the "standard" multi-paxos protocol, is that a new leader can find that there is an accepted value for instance n but no accepted value for instance $n-1$. At least, RAFT fixes that, because it ensures that there are never "holes" in the command history of a node. The side effect is that it suffers from some non-intuitive behaviours: for instance, to propagate a value to a majority of replicas may not be enough to commit that value.
- *Things to remember:* How leaders are elected in RAFT.
- *Why should I care?* RAFT is open source and many projects use RAFT as a building block.
- *Must read:* The RAFT paper.
- *Further reading:* Ongaro's PhD thesis proposes some "optimisations" to the RAFT reconfiguration protocol presented in the paper. Some of these optimisatios were found to be wrong! This illustrates how difficult Paxos reconfiguration is!

Lecture 9: Database Replication

- *Clickbait:* How many operations can you fit in a single Paxos command?
- *Summary:* This class covers some of the basic strategies to replicate a database: primary-backup, state-machine replication, and multi-master replication with totally-ordered certification-
- *Things to remember:* Databases offer strong consistency, so you will need some form of total-order algorithm (such as Paxos) to serialise conflicting transactions. However, you can execute the transaction after defining a total order, or execute the transaction optimistically, and later define the total order. Also, you can broadcast just the transaction identifier (just to establish an order), the transaction code (such that all replicas run the transaction), or the outcome of the transaction (the read and write set). Then, you can combine these ingredients in different ways to implement different algorithms.
- *Why should I care?* Many applications use databases. If you want your application to be fault-tolerant you need to replicate a database.
- *Must read:* "Understanding Replication in Databases and Distributed Systems" provides a good overview of the main techniques that can be used to implement database replication. Note that you need to have some background in database concurrency control to understand some of these algorithms. If you need to refresh your memory, the book "Distributed Systems: Concepts and Design" offers a good introduction to distributed databases.
- *Further reading:* "Comparison of Database Replication Techniques Based on Total Order Broadcast" compares the performance of different algorithms; it is interesting to understand the tradeoffs involved. "AKARA: A Flexible Clustering Protocol for Demanding Transactional Workloads" shows how you can speed things up, by exploring the parallelism supported by the local (non-replicated) database.' "Blotter: Low Latency Transactions for Geo-Replicated Storage" discusses the problems you face when replicas are distant from each other.

Lecture 10: Spanner

- *Clickbait:* Who said you cannot have strong consistency and large-scale?
- *Summary:* Spanner is a large-scale database that offers strict serialisability, the strongest form of consistency for databases. It does this in a large-scale system, where the database is split into multiple shards, and each shard is replicated for fault-tolerance.
- *Things to remember:* Spanner is very interesting for several reasons. It illustrates how several of the concepts adressed in DS and DIDA can be combined to build a powerful system. Spanner uses clock synchronisation, Paxos, Skeen's collective agreement protocol, and a mix of pessimistic and optimistic concurrency control. Just imagine how hard it would be to understand Spanner before having SD and DIDA.
- *Why should I care?* For many years, Spanner has supported a large number of applications at Google. Many system need to offer fault-tolerantce and support transactions; these systems must implement mechanisms similar to those used by Spanner.
- *Must read:* "Spanner: Google's Globally Distributed Database".
- *Further reading:* "Spanner's Concurrency Control" attempts to explain the concurrency control mechanisms of Spanner in a different way. "Megastore" is an earlier system from Google that offers simpler form of transactions (you can see Megastore as a step towards Spanner). "Paxos Replicated State Machines" shows that replicated datastores based on Paxos can have good performance.

Lecture 11: TCC

- *Clickbait:* How consistent can a highly available system be?
- *Summary:* The CAP theorem states that you cannot have consistency and availability in an asynchronous system where the network can suffer transient outages. Spanner offers strong consistency but can block, because it relies on Paxos to keep replicas consistent. Transactional Causal Consistency (TCC) is a consistency model designed for geo-replication. It is the strongest consistency model that can be implemented without blocking. This lecture addresses the challenges in implementing TCC systems.
- *Things to remember:* How can a system propagate updates that are part of a given transaction from one datacenter to another without violating causality, atomicity, and availability?
- *Why should I care?* When an application is down, large amounts of revenue may be lost. High availability ensures that clients always get some service and the the provider always gets some money. This is critical in many areas.
- *Must read:* "Cure: Strong semantics meets high availability and low latency", the first system to implement TCC.
- *Further reading:* "Stronger Semantics for Low-Latency Geo-Replicated Storage" is a system that offers a weaker guarantees than TCC (it does not support read-write transactions, and all transactions need to be static) but has an interesting scheme to allow clients to read the most recent consistent version of the datastore. "Distributed transactional reads: the strong, the quick, the fresh & the impossible" discusses the tradeoffs between availably and freshness in geo-replicated systems. "Wren" attempts to improve Cure. "Transactional Causal Consistency for Serverless Computing" shows that TCC can also be useful to improve "Function-as-a-Service" applications (in a single datacenter).

Lectures 12-14: P2P Systems

- *Clickbait:* You should never do this at home. But then, if you don't, nobody else will.
- *Summary:* Introduction to P2P systems. Unstructured P2P systems: the importance of peer-sampling. Structured P2P systems: distributed hash table. Chord and Pastry. Implementing file systems and publish-subscribe systems on top pf P2P systems.
- *Things to remember:* What is the purpose of a peer-sampling service. How the routing tables of these systems and built, maintained, and used. How can we use consistent hashing in a datacenter.
- *Why should I care?* P2P system are often associated to illegal file sharing. Many cripto-coins are based on P2P systems. However, theere are many other uses of the technology. Systems are becoming larger and larger. P2P protocols are scalable and may be applied in many contexts.
- *Must read:* Chapter 10 of "Distributed Systems: Concepts and Design". The Dynamo papers.
- *I want to know more:* The original Dynamo offers only eventual consistency. Peer-to-peer systems suffer from "churn", i.e., the constant arrival and departure of nodes, that cause recurrent reconfigurations to the ring. As we have discussed in the "reconfiguration" block of this course, reconfiguration is hard. Frequent reconfiguration is even harder. Still, several systems have attempted to offer strong consistency on top of a DHT. In the web page you can find pointers to papers that attempt to do that. There was not time to discuss these attempts in the class, and they will not appear in the exam, but if yu are curious you can take a peek of these problems to understand the challenges and how they can be potentially adressed.

5 Paper Presentations

The course also includes a number of paper presentations. In the past, we have selected a number of papers covering multiple topics. This year, I am

going to try something slightly different: all papers address the same general problem, in this case, how to build a distributed shared log. Students will be encourage to discuss the tradeoffs among these alternatives.

6 Further Questions

If you have further questions, I may have the answers. Do not forget that you may (and you should) contact me to clarify any question you may have.