Back To Module Home

Distributed Systems

0% completed

Introduction to Distributed Systems

Mark Module as Completed

Difficulties Designing Distributed Systems

Let's see what makes distributed systems hard to design.

We'll cover the following

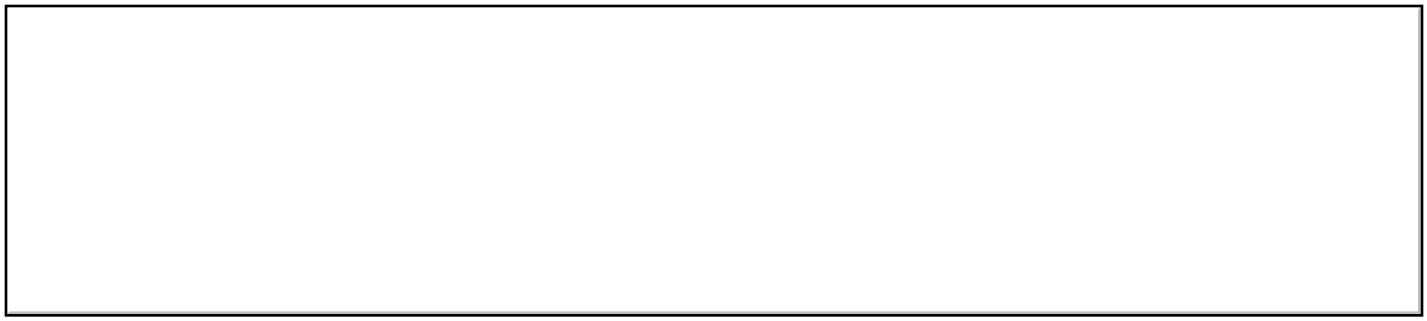
Why distributed systems are hard to design#

In general, distributed systems are hard to design, build, and reason about. This increases the risk of error.

It's worth questioning this: why are distributed systems so hard to design? The answer to this question will help us eliminate our blind spots, and provide guidance on some aspects we should pay attention to.

Properties that make distributed systems challenging#

The following illustration shows the main properties that make distributed systems challenging to reason about.

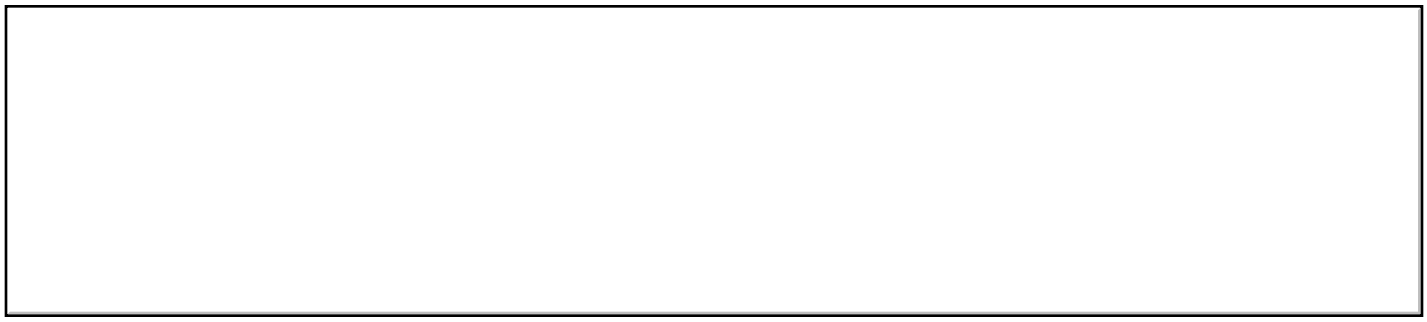


Main properties of distributed systems

Let's look at each property.

Network asynchrony#

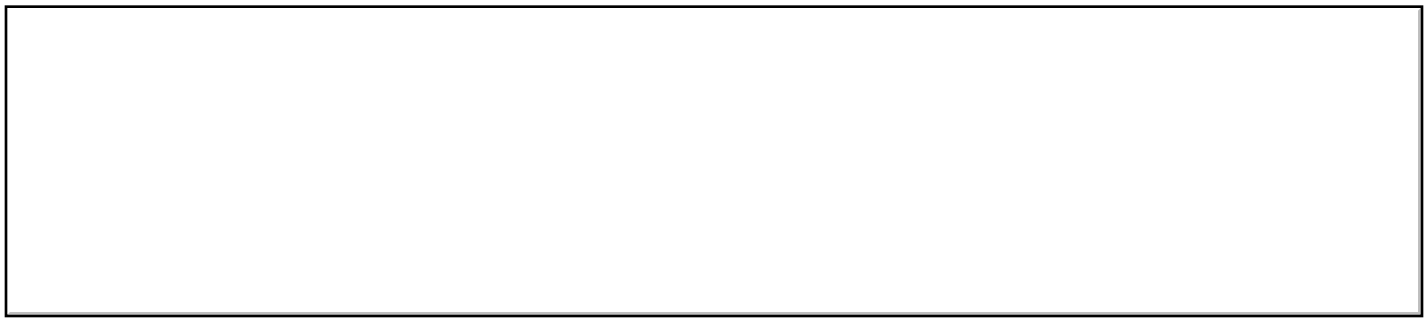
Network asynchrony is a property of communication networks that cannot provide strong guarantees around delivering events, e.g., a maximum amount of time a message requires for delivery. This can create a lot of counter-intuitive behaviors that are not present in non-distributed systems. This contrasts to memory operations that provide much stricter guarantees. For instance, messages might take extremely long to deliver in a distributed system. They may even deliver out of order—or not at all.



Clocks with different times

Partial failures#

Partial failures are the cases where only some components of a distributed system fail. This behavior can contrast with certain kinds of applications a single server deploys. These applications work under the assumption that either everything is working fine, or there has been a server crash. It introduces significant complexity when it requires atomicity across components in a distributed system. Thus, we must ensure that we either apply an operation to all the nodes of a system, or to none of them.



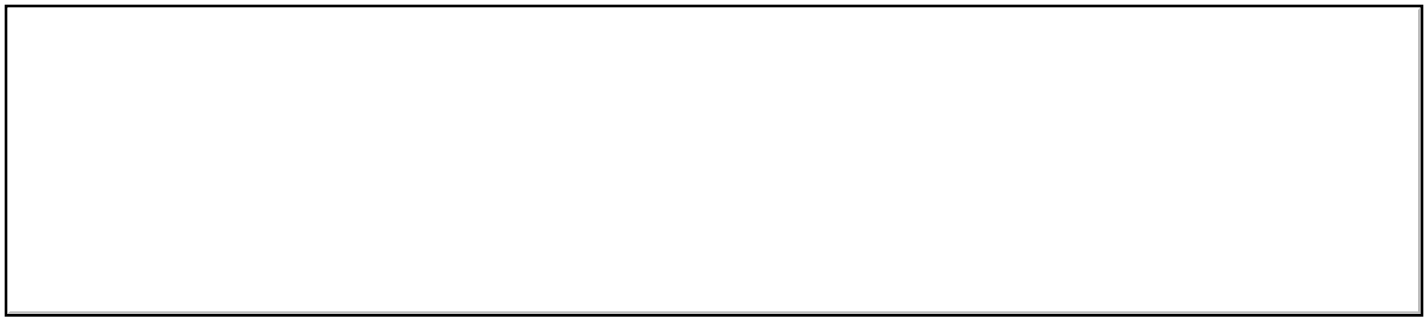
Failures of two nodes in a distributed system of six nodes

> The chapter about achieving atomicity analyses this problem.

Concurrency#

Concurrency is the execution of multiple computations at the same time, and potentially on the same piece of data. These computations interleave with each other. This introduces additional complexity since these computations can interfere with each other and create unexpected behaviors. This is, again, in contrast to simplistic

applications with no concurrency, where the program runs in the order the sequence of commands in the source code defined.



Two processes writing to the same resource concurrently

> The chapter that talks about isolation explains the various types of problematic behaviors that arise from concurrency.

Network asynchrony, partial failures, and concurrency are the major contributors to complexity in the field of distributed systems. So, we should keep them in mind when we build distributed systems in real life. Doing so would help us anticipate edge cases and handle them appropriately.

Fallacies of Distributed Computing

Next

Measures of Correctness in Distribute…

Mark as Completed

Report an Issue