FLEX - Distributed Computing

Simon Härer

Created: 2024-06-10 Mon 07:48

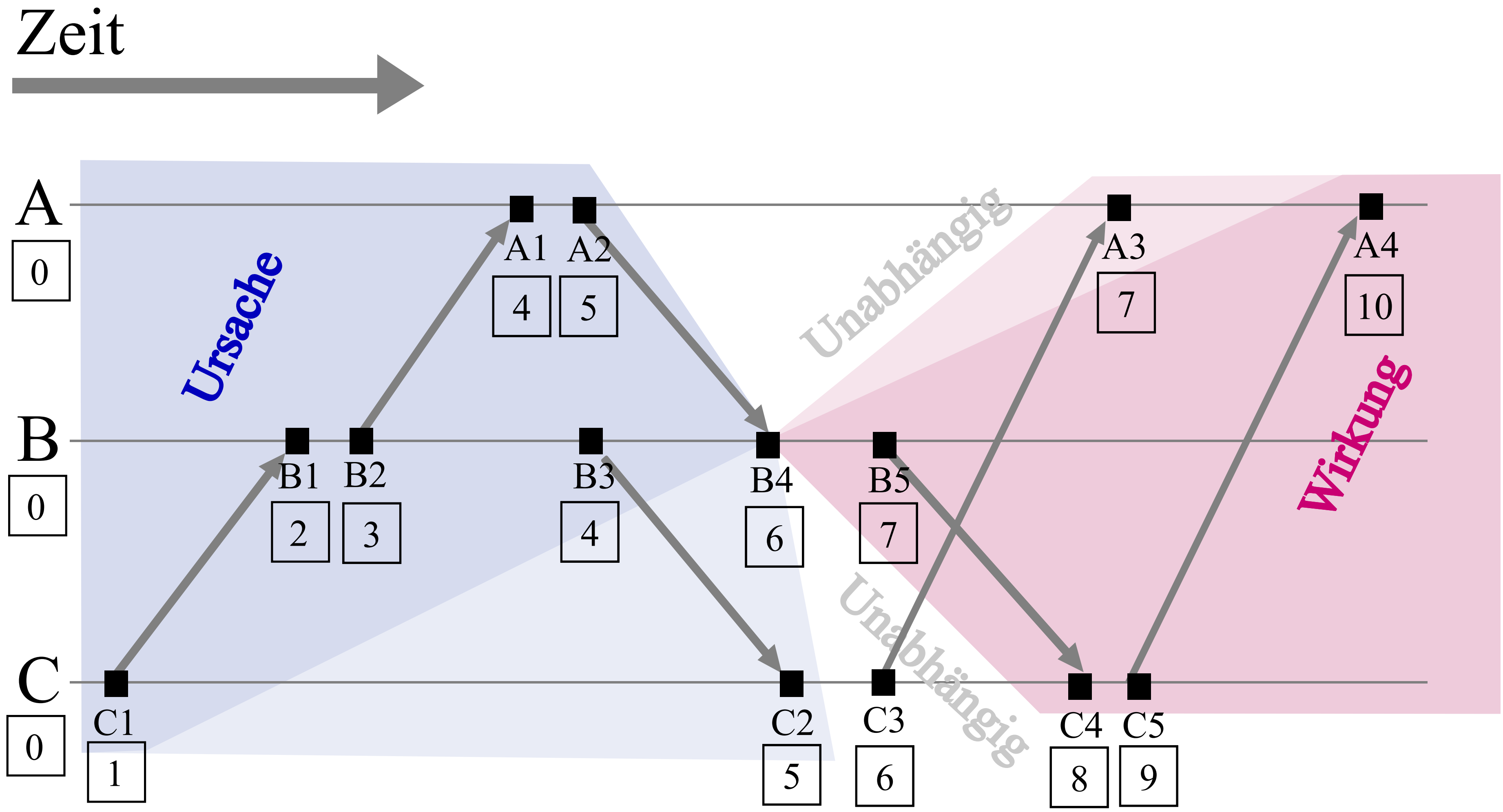
Monolithic Systems

- everything on one machine, in one process
- displayed information is instantly updated
- data storage is synchronously updated

Traditional Stability Approaches

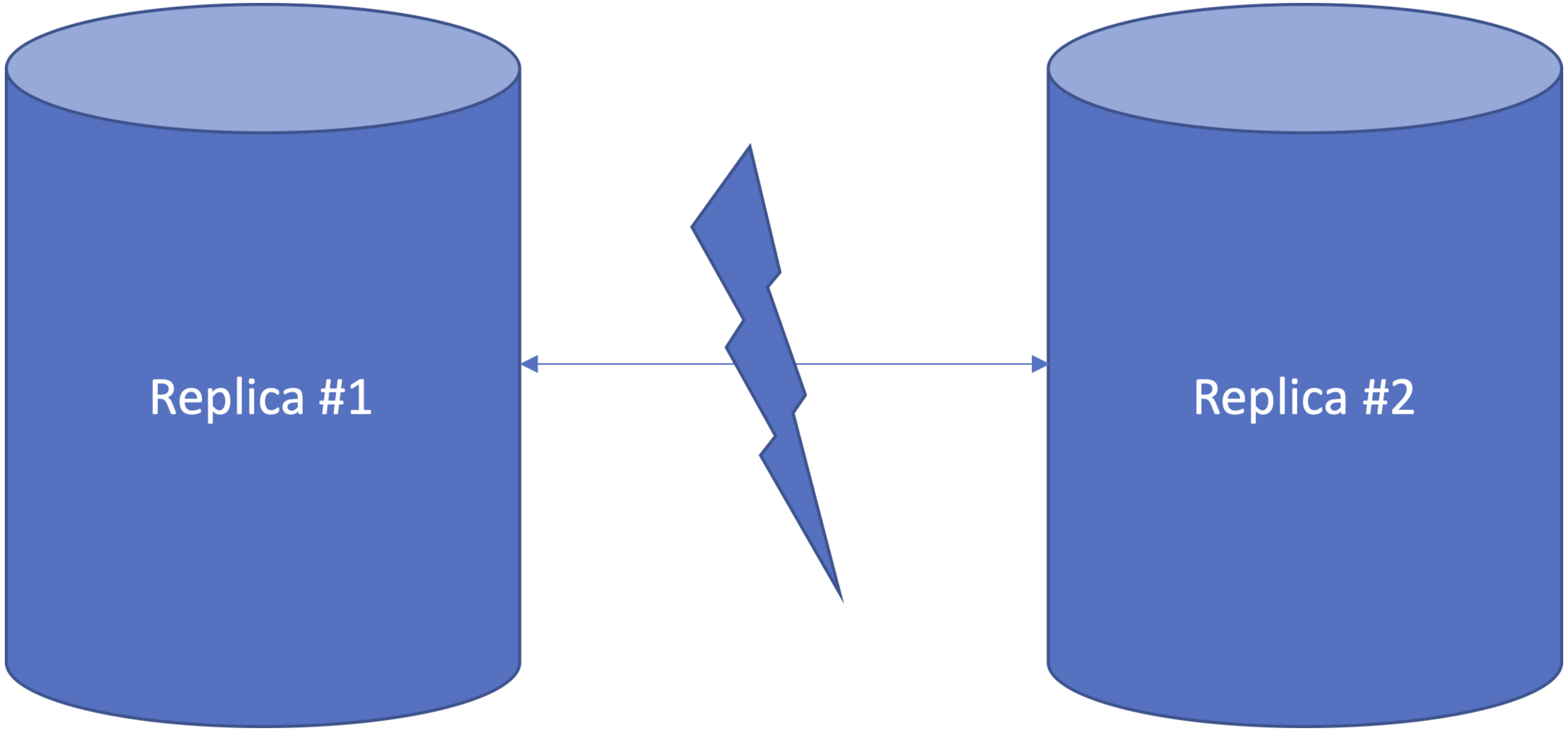
- ”pretend monolith”
- synchronous communication
- keep everything running at all costs
- use database transactions to ensure synchronous updates

Causality and Event Ordering



Lamport-Uhr (Quelle: Wikipedia)

Split-Brain-Problem



Fallacies of distributed computing

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure

Rotem-Gal-Oz, Arnon. (2008). *Fallacies of Distributed Computing Explained*. Dr. Dobb's Journal.

Fallacies of distributed computing

- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

Rotem-Gal-Oz, Arnon. (2008). *Fallacies of Distributed Computing Explained*. Dr. Dobb's Journal.

Exercise: Erlbank - Fallacies of Distributed Computing

Analyze the improved Erlbank application with respect to these fallacies.

Resilience

In case of distributed computing, we always have to think about what happens when a service is unreachable from another service. One strategy is to increase uptime, another is to program with errors in mind.

There are several patterns that can be used to increase resilience in a distributed setup. We will discuss Retry, Fallback, Circuit breaker and Bulkhead.

Nygard, Michael T.. *Release It!: Design and Deploy Production-Ready Software*. (2017).

Retry Pattern

Very simple:

- A service retries in case a request to another service fails
- Helps in case of e.g. temporary package loss
- Doesn't help if the other service is overloaded, maybe makes it even worse

Fallback

- Define fallback logic if a service is not reachable
- E.g. product ratings cannot be fetched? Just don't show ratings for now
- Not always possible, e.g. what if the product information itself cannot be fetched?

Circuit Breaker

The circuit breaker sits between the requesting service and the responding service. It can have 3 states: Open, closed and half-open.

- In closed state, all requests are handled unconditionally
- In open state, all requests are rejected
- In half-open state, some requests are handled, others rejected

Circuit Breaker

In case of multiple consecutive (internal) errors in the responding service, the circuit breaker opens. It then rejects incoming requests. After some time, it half-opens and handles some requests. If these are successful again, it resets to close.

Circuit breakers allow controlled error handling and are especially helpful, when a service is overloaded. They work well combined with e.g. the Fallback pattern.

Bulkhead Pattern

Bulkhead is a resource partitioning between many service that normally would access the same resources, e.g. CPU.

- If one service exhaust its resources, it is not reachable anymore
- Due to bulkheads, other services are not affected, they have their own resources
- Helpful combined with Fallback, so that the application is still working in case of one service being overloaded

No Silver Bullet

Patterns and measures depend on:

- requirements
- environment
- people

Availability

Availability = Uptime / (Uptime + Downtime)

Availability = MTBF / (MTBF + MTTR)

MTBF Maximization Strategies

- automated testing before deployment
- "chaos engineering"
- automatic restart of failed components
- reroute to working replicas
- supervisor hierarchies
- event monitoring
- latency monitoring

MTTR Minimization Strategies

- incident-management action plan
- define roles for incident management
- training on roles and functions
- monitor and alert
- anomaly detection
- document
- review incidents

Fault Isolation Strategies

- small components with loose coupling
- expect other components to fail
- horizontal replication
- ″let it crash″
- supervisor hierarchies