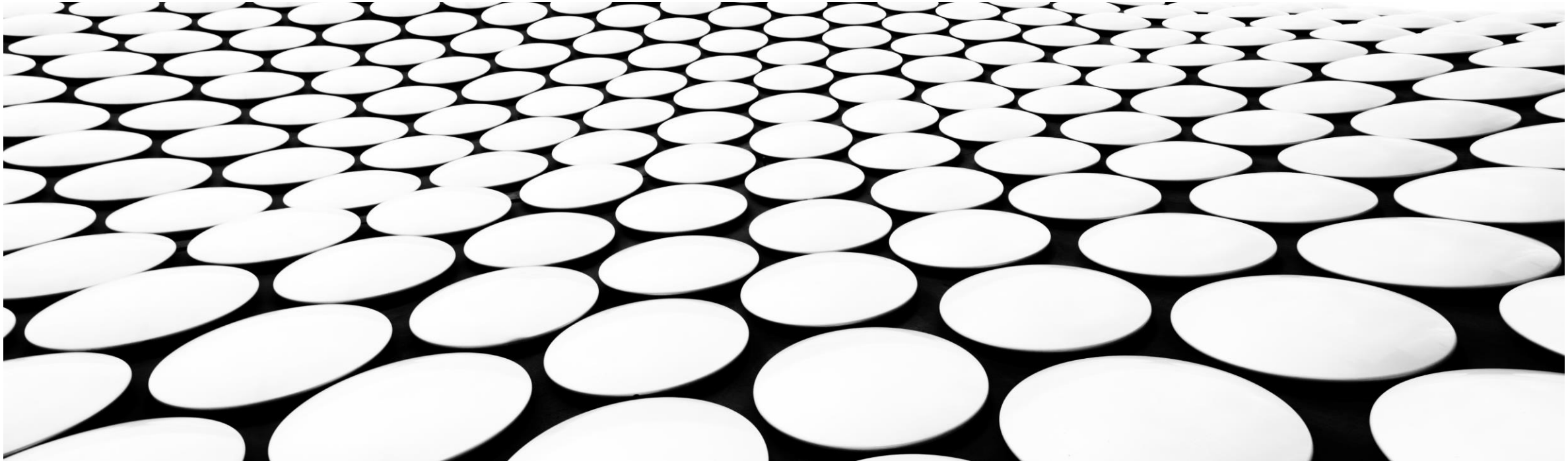
Resiliency in distributed systems (part 2 of 2)

COEN-317: Distributed Systems
Robert Bruce
Department of Computer Science and Engineering
Santa Clara University



Properties of dependable systems

Availability: "The property that a system is ready to be used immediately" [1].

Reliability: "The property that a system can run continuously without failure" [1].

Safety: The extent that a system failure has no catastrophic consequences [1].

Maintainability: "How easily a failed system can be repaired" [1].

[1] pp. 424-425, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Types of faults in distributed systems

Transient fault: "[Faults that] occur once and then disappear" [1]

Intermittent fault: Faults that occur repeatedly then disappear yet are difficult to diagnose [1].

Permanent fault: Faults that persist until a failed component is replaced [1].

[1] p. 427, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Failure models

Crash failure: "When a server prematurely halts but was working correctly until it stopped" [1].

Receive-omission failure: A type of failure in which "the server never got the request in the first place" [1].

Send-omission failure: A type of failure in which a server completes a task "but somehow fails in sending a response" [1].

Timing failure: A type of failure in which a server's response "lies outside a specified real-time interval" [1].

Response failure: A type of failure in which the server provides either an incorrect response [2].

State-transition failure: "A kind of failure [that occurs] when the server reacts unexpectedly to an incoming request" [2].

Arbitrary failures: Synonymous with ***Byzantine failures*****,** refers to the most serious type of failure [2].

[1] p. 428, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.
[2] p. 429, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Achieving fault tolerance through redundancy: three techniques

Information redundancy: "extra bits are added to allow recovery from garbled bits" [1].

Time redundancy: "an action is performed, and then, if need be, it is performed again" [1].

Physical redundancy: "extra equipment or processes are added to make it possible for the system as a whole to tolerate the loss or malfunctioning of some components" [1].

[1] p. 431, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Five classes of RPC failures

Five classes of Remote Procedure Call (RPC) failures in distributed systems [1]:

1. "The client is unable to locate the server."
2. "The request message from the client to the server is lost."
3. "The server crashes after receiving a request."
4. "The reply message from the server to the client is lost."
5. "The client crashes after sending a request."

[1] p. 462, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

Recovering from failure

Two forms of error recovery:

- **Backward recovery**: "bring the system from its present erroneous state back into a previously correct state" [1].
- **Forward recovery**: "an attempt is made to bring the system in a correct new state from which it can continue to execute" [1].

[1] p. 491, *Distributed Systems* (3rd edition) by Maarten van Steen and Andrew S. Tanenbaum.

For further reading.

Blockchain technology could certainly be used to increase resiliency in a distributed system. Here are some references:

- *Mastering Bitcoin: Programming the Open Blockchain (2nd edition)*
  https://www.oreilly.com/library/view/mastering-bitcoin-2nd/9781491954379/
- Source code for *Mastering Bitcoin* book above
  https://github.com/bitcoinbook/bitcoinbook
- *Bitcoin: A Peer-to-Peer Electronic Cash System*
  https://bitcoin.org/bitcoin.pdf
- *Blockchain for Fault-Tolerant Grid Operations*
  https://www.osti.gov/biblio/1846926
- *Fault Tolerance in a distributed system forming a blockchain*
  https://medium.com/game/fault-tolerance-1800d3093657
- *Blockchain: Byzantine Fault Tolerant*
  https://www.blockchain-council.org/blockchain/blockchain-byzantine-fault-tolerant/
- *The Raft Consensus Algorithm* [note: RAFT is an implementation of Paxos but included here as a potential useful source]
  https://raft.github.io/