Institute of Computer Science, University of Bern
Distributed Algorithms, Spring 2020

Prof. Christian Cachin
Jovana Mićić

Defining Dependable Systems

0.1 Terminology

Systems. A *specification* describes the ideal behavior of a *system* at its *interfaces*. A system is called *dependable* if it follows the specification as closely as possible, even if faults occur. Typical faults are when some of parts of the system fail or when the environment behaves differently than assumed.

Three causally related terms have been defined to describe dependable systems [ALRL04]:

$$\boxed{\text{Fault}} \longrightarrow \boxed{\text{Error}} \longrightarrow \boxed{\text{Failure}}$$

Fault: (Hypothetic) Cause of an error.

Error: Internal system state that does not correspond to specification, not visible at interfaces.

Failure: Deviation of system from specification at interfaces.

Recursive!

Example 1. In a computer system, fan in power supply congested by dust, airflow massively reduced, fan not effective, power supply overheats, some part burns out, system loses power, computer stops working.

- Fan system: dust = fault, reduced airflow = error, no cooling = failure.
- Power supply system: no cooling = fault, overheating = error, loss of power = failure.
- Computer system: broken power supply = fault, no power on mainboard = error, computer stopping = failure.

Example 2. RAID storage system (bits with ECC, disks with RAID mirroring).

Example 3. Typical software vulnerability, buffer-overflow attack exploited by a worm.

0.2 Attributes of dependable systems

- Availability — readiness for correct service
- Reliability — continuity of correct service
- Safety — absence of catastrophic failures
- Confidentiality — no unauthorized disclosure
- Integrity — no improper states or state changes

0.3 Techniques

Prevention: Formal design, access control, good engineering.

Tolerance:

- Error & fault detection — Failure detectors, intrusion detection systems.
- Recovery — Isolation, rollback, compensation, fail-over, database transactions (ACID).
- Redundancy — Replication, voting (ECC, RAID), diversity.

Removal:

- Formal verification of implementation w.r.t. specification
- Validation of specification w.r.t. real environment
- Fault injection and testing

Forecasting:

- Modeling, prediction
- Evaluation, testing (fault trees, attack graphs)

References and further reading

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, *Basic concepts and taxonomy of dependable and secure computing*, IEEE Transactions on Dependable and Secure Computing **1** (2004), no. 1, 11–33.

[Gra85] J. Gray, *Why do computers stop and what can be done about it?*, Tech. Report 85.7, Tandem Corp., June 1985, Available from `https://www.hpl.hp.com/techreports/tandem/TR-85.7.pdf`.

[Pat02] D. A. Patterson, *An introduction to dependability*, ;login: — The Magazine of the USENIX Association **27** (2002), no. 4, 61–65, Available from `https://www.usenix.org/system/files/login/articles/1818-patterson.pdf`.

[Ros14] S. M. Ross, *Introduction to probability models*, 11th ed., Academic Press, 2014.

[SS98] D. P. Siewiorek and R. S. Swarz, *Reliable computer systems: Design and evaluation*, 3rd ed., A K Peters, 1998.