2 Dependability

2.1 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.

2.2 Attributes

- 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

2.3 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)

Commercial fault-tolerant systems employ a combination of the above techniques, but focus on sophisticated designs for fault-tolerance through hardware redundancy, combined with isolation and recovery methods for software [BS04]. Examples include the HP Integrity NonStop servers (formerly built by Tandem Corp.) [BBV$^+$05] and IBM's System z mainframe servers and z/OS operating system (successors of S/390 servers and OS/390) [Hof97].

2.4 Measures

[This section corrects mistakes in the originally distributed version.]

The *reliability function* $R(t)$ of a system denotes the probability that the system runs correctly from time 0 until time t. The reliability function is monotonically decreasing with time, starting from $R(0) = 1$.

We assume that the failure of any elementary system considered here occurs with exponential (or Poisson) distribution,

$$R(t) \;=\; \Pr[\,\text{System works correctly from time 0 until } t\,] \;=\; e^{-\lambda t},$$

where λ denotes the *failure rate*. When considering multiple systems, we assume they fail independently of each other.

Serial combination. Consider a system *Ser* that consists of a "serial" combination (without redundancy) of the components $C_1, \ldots, C_n$ that have failure rates $\lambda_1, \ldots, \lambda_n$, respectively. The system functions only if all components function properly. The reliability of *Ser* is the product of the component reliabilities,

$$R_{\mathit{Ser}}(t) \;=\; \prod_i R_{C_i}(t) \;=\; \prod_i e^{-\lambda_i t}.$$

Note that the failure probability of *Ser* is also modeled by an exponential distribution with failure rate

$$\lambda_{\mathit{Ser}} \;=\; \sum_i \lambda_i.$$

Parallel combination. Consider a system *Par* in which components $C_1, \ldots, C_n$ are redundant and used "in parallel," such that *Par* fails only if all components have failed. The component failure rates are again $\lambda_1, \ldots, \lambda_n$. Then reliability function of *Par* is

$$R_{\mathit{Par}}(t) \;=\; 1 - \prod_i \big(1 - R_{C_i}(t)\big) \;=\; 1 - \prod_i \big(1 - e^{-\lambda_i t}\big).$$

Note that the failure probability of *Par* is *not* an exponential distribution.

k-out-of-n combination. More generally, suppose a system *Thresh* consists of n components $C_1, \ldots, C_n$ that have equal reliability function $R(t)$, of which at least k are needed for *Thresh* to function (k represents a "threshold"). It holds

$$R_{\mathit{Thresh}}(t) \;=\; \sum_{i=k}^{n} \binom{n}{i} R(t)^i \big(1 - R(t)^{n-i}\big).$$

MTTF and MTTR. As described so far, our model allowed a system to fail only once and it would remain inoperable forever afterwards. If we include *repair actions* as well, a system may now continuously transition between the operational and the failed state. More precisely, such a system either works as intended (A) or it is being repaired (B). The transition from (A) to (B) corresponds to a failure and the transition from (B) to (A) models a repair.

The *mean time to failure (MTTF)* is computed as[1] $\mathit{MTTF} = \int_0^\infty R(t)dt$. Assuming an exponential failure distribution with rate λ, the MTTF has a closed form,

$$\mathit{MTTF} \;=\; \frac{1}{\lambda}.$$

Analogously to the probabilistic occurrence of failures, one may assume that repairs occur with random delays according to a given *error function* $E(t)$, which denotes the probability that the system remains in a failed state from time 0 until time t. If repairs occur with exponential distribution just like failures, then

$$E(t) \;=\; e^{-\mu t}$$

for a *repair rate* μ. Analogously to failures, the *mean time to repair (MTTR)* is defined by $\mathit{MTTR} = \int_0^\infty E(t)dt$, and for an exponential distribution of repair actions, we get

$$\mathit{MTTR} \;=\; \frac{1}{\mu}.$$

Availability. One uses the *availability* as an overall measure for how likely a system is to perform its service corretly. Formally, it is defined as the probability that the system is operational at any instant and can be expressed as

$$\mathit{Availability} \;=\; \frac{\mathit{MTTF}}{\mathit{MTTF} + \mathit{MTTR}}.$$

Note that if the MTTF is already large, system availability can be further increased by reducing the MTTR — this is important for large-scale online services, where components that may fail are optimized to recover fast.

High availability is usually measured in "nines," expressed as a percentage of uptime in a year and counting the number of nines in the resulting expression. For example, 99.999% or "five nines" of availability denotes up to 5.26 minutes of downtime per year.

Problems of MTTF & MTTR. How can the notions of MTTF and MTTR in the hundred-thousands of hours be interpreted for a system that runs only for a fraction of this time [Pat02]?

There are 8760h per year. Does a system with MTTF of 500'000h on average run for 57 years without failures, even though its manufacturer specifies a system lifetime of 5 years? No. MTTF and MTTR are only statistical measures that are relevant in a large population of samples, i.e., if you have 100 systems expect a faulty one every 0.57 years. Note the fundamental assumption that failures are independent.

For introductions to reliability theory, see the textbooks of Siewiorek and Swarz [SS98] or Ross [Ros09, Chapter 9].

2.5 Redundancy in storage systems — RAID

- RAID (Redundant Arrays of Inexpensive Disks) [CLG+94];
- RAID-5;
- MDS erasure codes, RAID-6 [Pla05], EVENODD [BBBM95];
- Generalized RAID-6 codes [PBVZ11].

[1] Consult the literature [SS98, Ros09] for a complete explanation of this statement.

References

[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.

[BBBM95] M. Blaum, J. Brady, J. Bruck, and J. Menon, *EVENODD: An optimal scheme for tolerating double disk failures in RAID architectures*, IEEE Transactions on Computers **44** (1995), no. 2, 192–202.

[BBV+05] D. Bernick, B. Bruckert, P. D. Vigna, D. Garcia, R. Jardine, J. Klecka, and J. Smullen, *NonStop® Advanced Architecture*, Proc. International Conference on Dependable Systems and Networks (DSN-DCCS), June 2005, pp. 12–21.

[BS04] W. Bartlett and L. Spainhower, *Commercial fault tolerance: A tale of two systems*, IEEE Transactions on Dependable and Secure Computing **1** (2004), no. 1, 87–96.

[CLG+94] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, *RAID: High-performance, reliable secondary storage*, ACM Computing Surveys **26** (1994), 145–185.

[Gra85] J. Gray, *Why do computers stop and what can be done about it?*, Tech. Report 85.7, Tandem Corp., June 1985, Available from `http://research.microsoft.com/~gray/`.

[Hof97] G. F. Hoffnagle, *Preface to the special issue on S/390 Parallel Sysplex Cluster*, IBM Systems Journal **36** (1997), no. 2, 170–371, On-line at `http://www.research.ibm.com/journal/sj36-2.html`.

[Pat02] D. A. Patterson, *An introduction to dependability*, ;login: — The Magazine of the USENIX Association **27** (2002), no. 4, 61–65.

[PBVZ11] J. S. Plank, A. L. Buchsbaum, and B. T. Vander Zanden, *Minimum density RAID-6 codes*, ACM Transactions on Storage **6** (2011), 16:1–16:22.

[Pla05] J. S. Plank, *Erasure codes for storage applications*, Tutorial, presented at the Usenix Conference on File and Storage Technologies (FAST), 2005.

[Ros09] S. M. Ross, *Introduction to probability models*, 10th ed., Academic Press, 2009.

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