Lecture 12
Reliability (Part 2)

<lecturer, date>

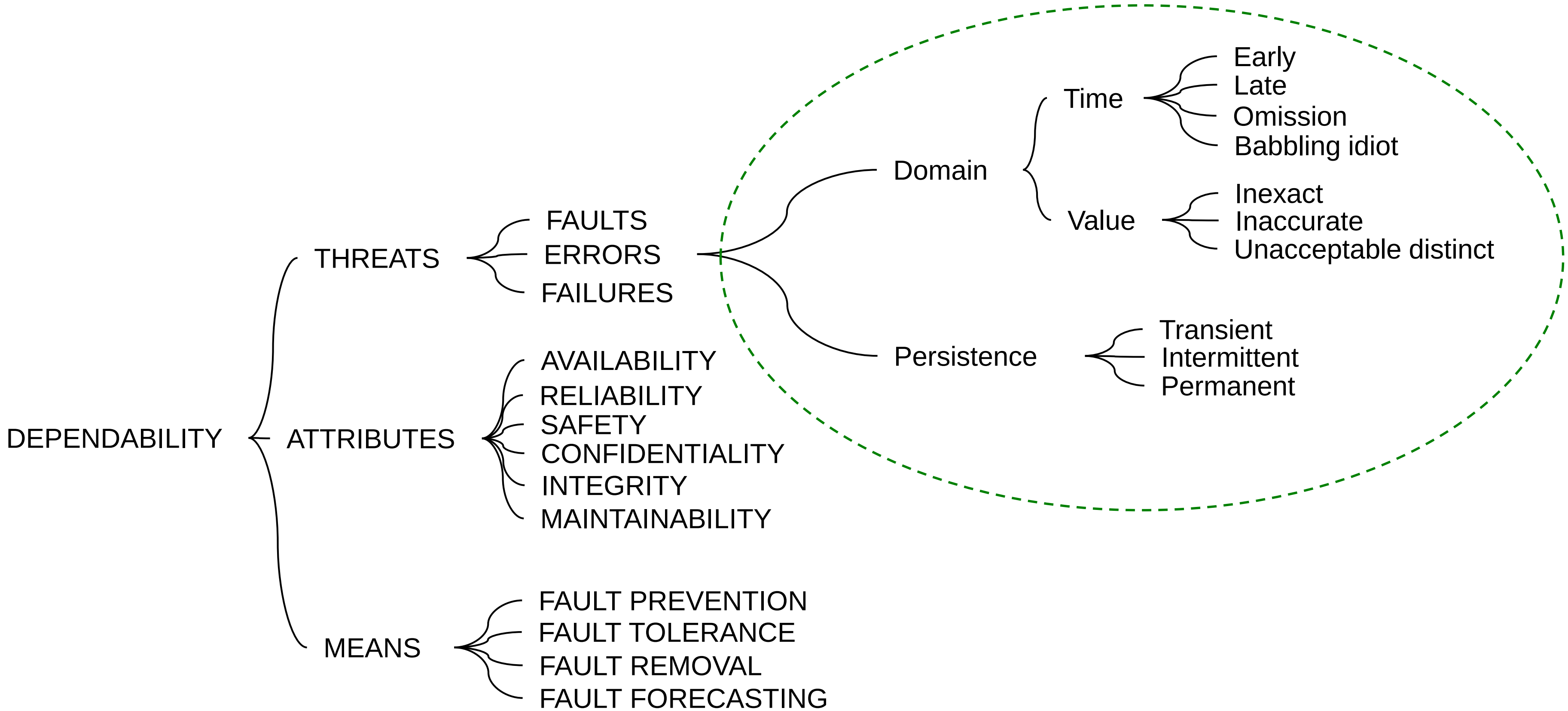
Outline

- Faults, Errors and Failures
  - Types of Errors
- Preventing Failures
  - Fault tolerance
  - Verification and Validation
  - Safety Analysis
- Fault Tolerance using Redundancy
  - Time redundancy
  - Space redundancy
  - N-version programming

Faults, Errors and Failures

- Fault
  - a defect in the system that can lead to failure
  - ex: radiation hits a memory cell in an altimeter component
- Error
  - manifestation of a fault
  - e.g., the hit changes the contents of the memory cell (i.e., resulting in wrong altitude value) and remains latent until the memory is read
- Failure
  - system malfunction
  - e.g., the altitude value is used by the autopilot, and the plane crashes

Types of Errors



Outline

- Faults, Errors and Failures
  - Types of Errors
- Preventing Failures
  - Fault tolerance
  - Verification and Validation
  - Safety Analysis
- Fault Tolerance using Redundancy
  - Time redundancy
  - Space redundancy
  - N-version programming

Preventing Failures

- Fault tolerance
  - to cope with the effects of faults
  - typically by redundancy

Focus of this lecture

- Verification and validation
  - to identify and eliminate faults

- Safety analysis
  - to focus on the most important faults

Outline

- Faults, Errors and Failures
  - Types of Errors
- Preventing Failures
  - Fault tolerance
  - Verification and Validation
  - Safety Analysis
- Fault Tolerance using Redundancy
  - Time redundancy
  - Space redundancy
  - N-version programming

Fault Tolerance using Redundancy

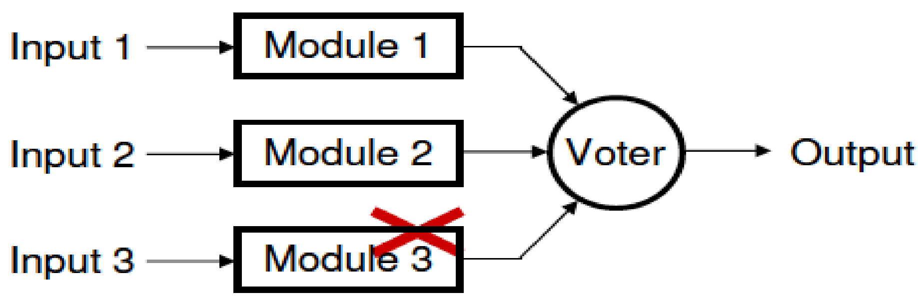
Two commonly used redundancy approaches:

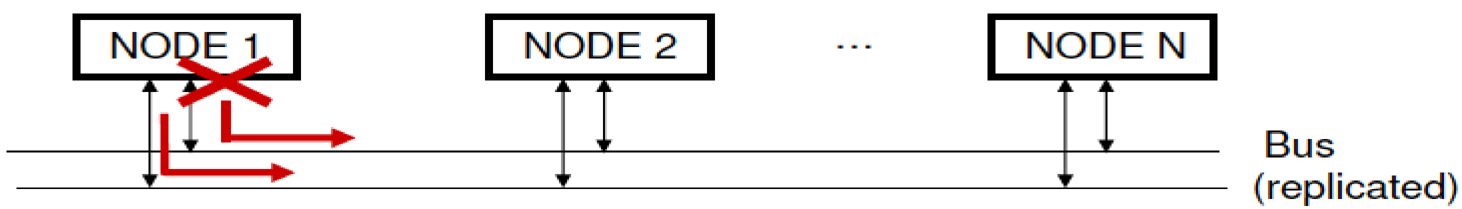
1. Spatial redundancy – replication
   ☺ Instant error masking, simplicity
   ☹ Weight, space, cost of hardware

2. Temporal redundancy – re-execution
   ☺ No extra weight, space, hardware
   ☹ Cost of processing power, delays, complexity

Spatial Redundancy

Mainly to tolerate *permanent faults*





Fault Tolerance using Redundancy

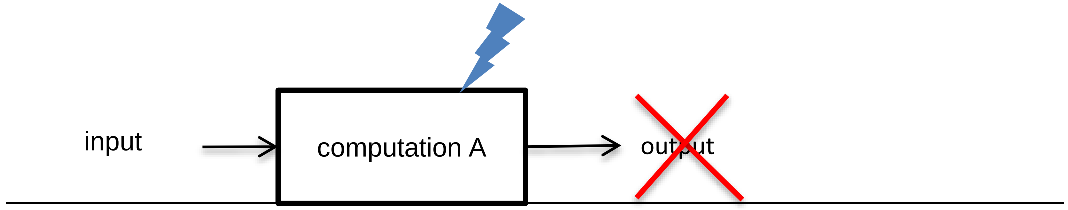
Two commonly used redundancy approaches:

1. Spatial redundancy – replication
   - ☺ Instant error masking, simplicity
   - ☹ Weight, space, cost of hardware

2. Temporal redundancy – re-execution
   - ☺ No extra weight, space, hardware
   - ☹ Cost of processing power, delays, complexity

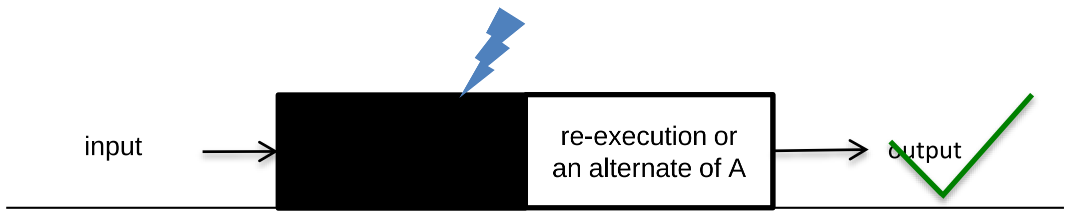
Temporal Redundancy

Mainly to tolerate *transient faults*



Temporal Redundancy

Mainly to tolerate *transient faults*



If A is a critical task then re-execute!

- ignore otherwise

Outline

- Faults, Errors and Failures
  - Types of Errors
- Preventing Failures
  - Fault tolerance
  - Verification and Validation
  - Safety Analysis
- Fault Tolerance using Redundancy
  - Time redundancy
  - Space redundancy
  - N-version programming

N version Programming

Multiple versions implementing the same functionality are developed from the same specifications

- Based on the conjecture that independently developed programs will greatly reduce the probability of the same types of faults occurring in two or more versions
- Central to the concept of fault tolerance using redundancy
- To improve reliability of software when implementing redundancy
  - Spatial redundancy: no two versions will give identical erroneous outputs
  - Temporal redundancy: the re-executed version will not generate the same erroneous output
- Interesting discussion in the research community on whether the above conjecture hold (references 4 to 7)

Challenges for Mobile Applications

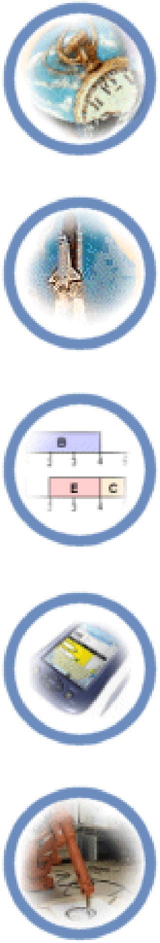
- Redundancy implies increased Size, Weight and Power (SWaP) constraints
  - More hardware implying bigger and heavier systems
  - Increased power requirements means reduced lifetime or alternately larger batteries
- Wireless communication is typically more unreliable and less secure than wired ones
  - external disturbances
  - easier to eavesdrop
- Higher complexity of the software due to mobile nature of the network
  - Need to cater to dynamically changing scenarios
- N-version programming increases the required software development efforts and hence the cost
  - Need more programmers to develop different versions

References

1) Basic concepts and taxonomy of dependable and secure computing, Avizienis, A. ; Laprie, J.-C. ; Randell, B. ; Landwehr, C., IEEE Transactions on Dependable and Secure Computing, 2004
2) An Experimental Evaluation Of The Assumption Of Independence In Multi-Version Programming, J. C. Knight , N. G. Leveson, IEEE Transactions on Software Engineering, 1986
3) N. G. Leveson, “High-pressure steam engines and computer software,” in Proceedings of the 14th International Conference on Software Engineering, 1992
4) Algirdas A. Avizienis, The methodology of N version programming, http://www.cse.cuhk.edu.hk/~lyu/book/sft/pdf/chap2.pdf
5) J.C Knight and N.G. Leveson, An experimental evaluation of the assumption of independence in multiversion programming, IEEE Transactions on Software Engineering, http://sunnyday.mit.edu/papers/nver-tse.pdf
6) A A. Avizienis, M R Lyu, and W Schutz, In Search of Effective Diversity: A Six-Language Study of Fault-Tolerant Control Software, http://ftp.cs.ucla.edu/tech-report/198_-reports/870060.pdf

References

7) J.C Knight and N.G. Leveson, A reply to the criticisms of the Knight & Leveson experiment, ACM SIGSOFT Software Engineering Notes, http://sunnyday.mit.edu/critics.pdf



Lab 12
Reliability (Part 2)

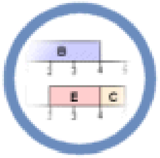
<lecturer, date>

Description

Take the unreliable version of the water tank controller and the mobile application that you developed in Lab 11.

- Implement temporal and spatial redundancy to tolerate these random data losses
- Draw a graph that plots the expected temperature and pressure of the water tank controller vs. the actual temperature and pressure for *30* simulations for each combination
- Write a report that details your conclusions and reflections. Also discuss which type of redundancy technique is more suitable for your scenarios.





Seminar 12
Reliability (Part 2)

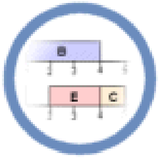
<lecturer, date>

Description

Discuss your lab in the class:

- Which type of redundancy is more suitable for mobile applications?
- Does redundancy affect security? Discuss this with respect to what you have learned in the previous lectures.
- How can your design reliable networks for such applications?
  - Perform a short literature survey and discuss the findings





Mini-project 12
Reliability (Part 2)

<lecturer, date>

Description

Summarize the following articles, while also including your reflections about the same:

1) Algirdas A. Avizienis, The methodology of N version programming, http://www.cse.cuhk.edu.hk/~lyu/book/sft/pdf/chap2.pdf
2) J.C Knight and N.G. Leveson, An experimental evaluation of the assumption of independence in multiversion programming, IEEE Transactions on Software Engineering, http://sunnyday.mit.edu/papers/nver-tse.pdf
3) A A. Avizienis, M R Lyu, and W Schutz, In Search of Effective Diversity: A Six-Language Study of Fault-Tolerant Control Software, http://ftp.cs.ucla.edu/tech-report/198_-reports/870060.pdf
4) J.C Knight and N.G. Leveson, A reply to the criticisms of the Knight & Leveson experiment, ACM SIGSOFT Software Engineering Notes, http://sunnyday.mit.edu/critics.pdf