

TDDDD07 – Real-time Systems

Lecture 7 (and part of Lecture 8): Dependability and Fault tolerance

Simin Nadjm-Tehrani

Real-time Systems Laboratory

Department of Computer and information Science

Reading Material

- Avizienis et al. 2004
 - Try to understand the possible fault dimensions (Figures 4, 5), and Fault tolerance approaches (Figure 16) by reading related texts
- Ch. 2 and Ch. 13 of B&W

From predictable to unpredictable?

- <https://www.youtube.com/watch?v=ooJIznO-RII>
- <https://www.youtube.com/watch?v=9by4PiQbSOw>

Dependability and Real-time

- If a system is to produce results within time constraints, it needs to produce results at all!
- Dependable computer systems *justify* and *measure* how well systems meet their requirements *in presence of faults*

Predictability and faults

- How do things go wrong and why?
- What can we do about it?



Dependability topics



Lectures 7 - 9 cover theory and practical examples

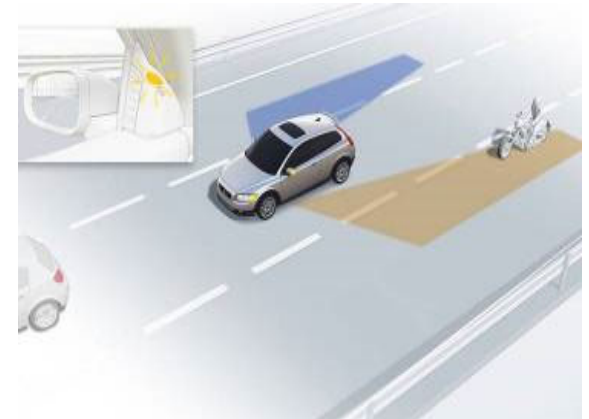
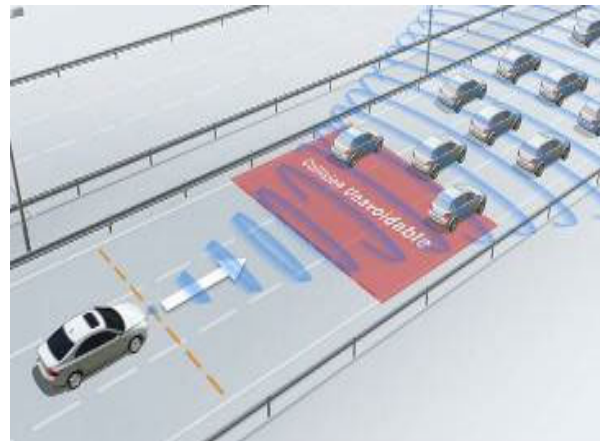
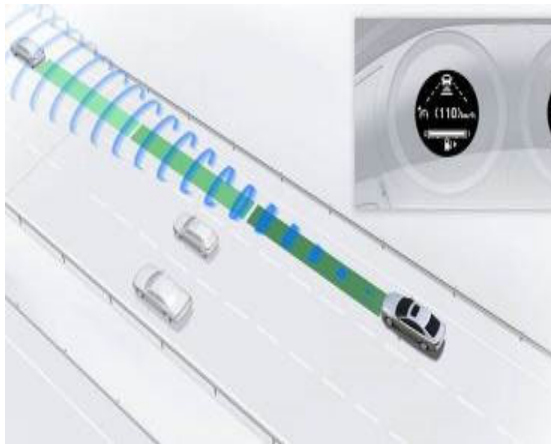
- Basic notions of dependability and redundancy in fault-tolerant systems
- Fault tolerance:
 - Relating faults/redundancy to distributed systems from lectures 4-6
 - Relating timing and fault tolerance

Lecture 8: Adds industrial perspective

Lecture 9: Fault prevention and design aspects

When things go wrong ...

Trends: software in cars



Source: Volvo Cars

From decision support to autonomy

| | | | |
|------|---|------|---|
| 1984 | ABS Anti-lock Braking System | 2004 | Blind Spot Information system (BLIS) |
| 1998 | Dynamic Stability and Traction Control (DSTC) | 2006 | Active Bi-Xenon lights |
| 2002 | Roll Stability Control (RSC) | 2006 | Adaptive Cruise Control (ACC) |
| 2003 | Intelligent Driver Information System (IDIS) | 2006 | Collision warning system with brake support |

October 24, 2013

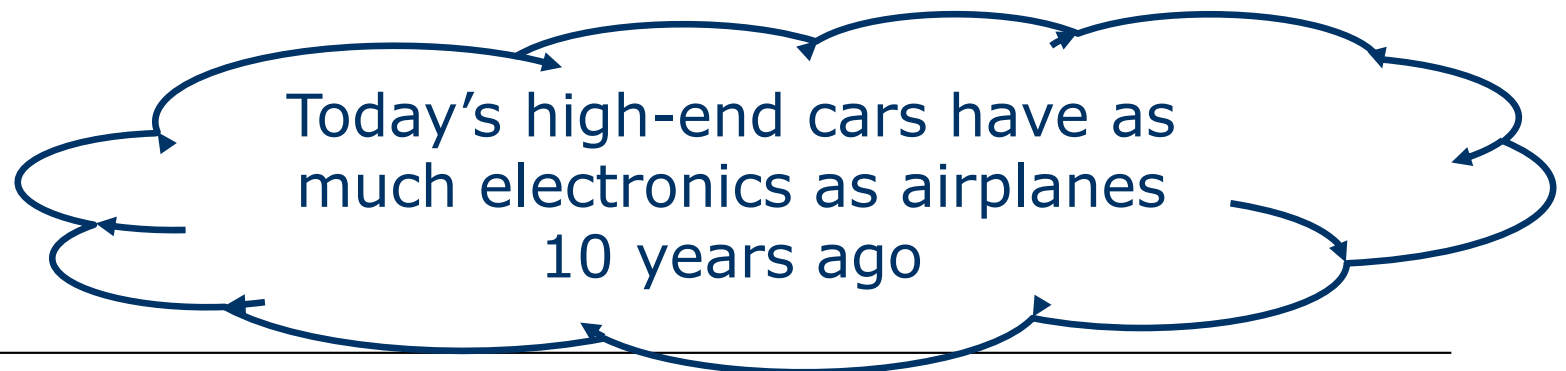
- Toyota in trouble ...

<https://www.edn.com/toyotas-killer-firmware-bad-design-and-its-consequences/>

- Phil Koopman's detailed analysis

<https://users.ece.cmu.edu/~koopman/toyota/>

<https://www.youtube.com/watch?v=DKHa7rxkvK8>



February 2, 2016

- 32 year old Kaushal Gandhi driving a Skoda Octavia on the M40 motorway, found that his cruise control was stuck at over 110 mph. His conversation in the emergency call for 8.5 minutes recorded the chain of events before a fatal accident where the car crashed into a parking lorry.

<https://www.theguardian.com/business/2016/nov/24/skoda-driver-decapitated-in-stuck-cruise-control-mystery>

February 20, 2016

- Volvo recalls 59,000 cars over software fault
- The glitch can shut down the engine and electrical system while the car is in motion
- Stefan Elfström told AP:
 - they would then both restart immediately

Experiments with vision systems

- With AI in today's autonomous driving functions we are getting the bad with the good
- <https://arstechnica.com/cars/2020/01/how-a-300-projector-can-fool-teslas-autopilot/>

June 2022

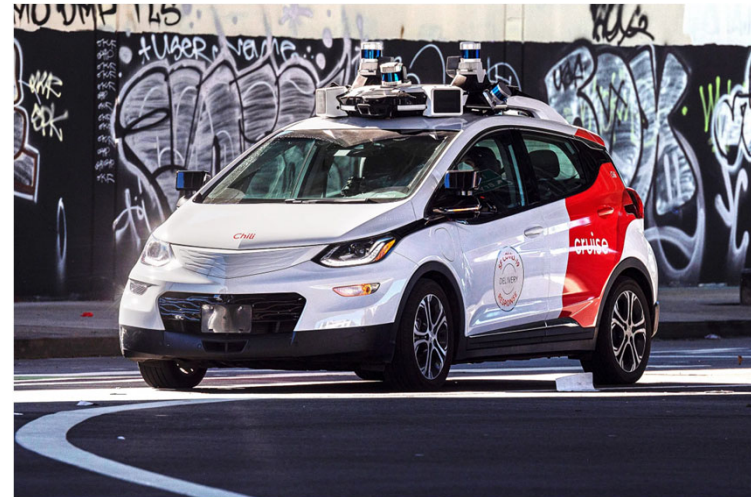
- “General Motors subsidiary recalled software deployed on 80 vehicles after two people were injured in a June crash involving a Cruise car operating autonomously in San Francisco.”

Wired 2022-09-01

<https://www.wired.com/story/gms-cruise-recalls-self-driving-software-involved-in-june-crash/>

October 24, 2023

- GM's Cruise loses its Self-Driving License in San Francisco



Source: David Paul, Morris/Bloomberg/Getty images

<https://www.wired.com/story/cruise-robotaxi-self-driving-permit-revoked-california/>

Early space and avionics

- During 1955, 18 air carrier accidents in the USA (when only 20% of the public was willing to fly!)
- Today's complexity many times higher

Airbus 380

- Integrated modular avionics (IMA), with safety-critical digital components, e.g.
 - Power-by-wire: complementing the hydraulic powered flight control surfaces
 - Cabin pressure control (implemented with a TTP operated bus)



Basic concepts

What is dependability?

Property of a **computing system** which allows reliance to be *justifiably* placed on the service it delivers.

[Avizienis et al. 2004]

The ability to avoid service failures that are *more frequent or more severe* than is acceptable

(sv. Pålitliga datorsystem)

Attributes of dependability

IFIP WG 10.4 definitions:

- **Safety:** absence of harm to people and environment
- **Availability:** the readiness for correct service
- **Integrity:** absence of improper system alterations
- **Reliability:** continuity of correct service
- **Maintainability:** ability to undergo modifications and repairs

Reliability

[sv. Tillförlitlighet]

- Means that the system (functionally) behaves as specified, and does it *continually* over measured intervals of time
- Measured through relating to probability of failure, e.g. 10^{-9}
- Another way of putting it: MTTF
 - In commercial flight systems - One failure in 10^9 flight hours

Safety and Security interplay

- Today we have software embedded in many networked systems that control critical infrastructures
- Water, electricity, transport are indeed safety-critical
- But also subject to severe security threats...

Safety – Security conflict

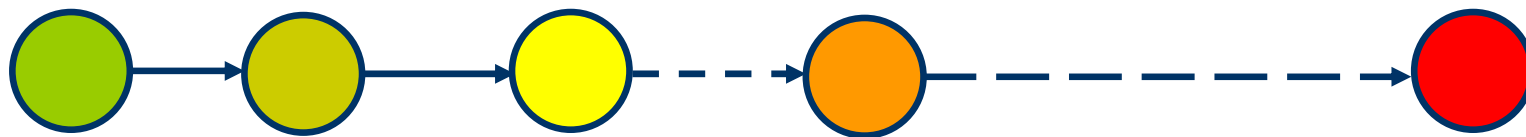
“... after an update of the SAP based maintenance software at DuPont without a subsequent safety review, an alarm notifying on a due date for a hose change disappeared. As a result, a hose used to transfer phosgene from a cylinder to a process wore out and catastrophically failed spraying a worker in the face resulting in his death.”

[Krotofil et al. 2019]

https://doi.org/10.1007/978-3-030-12330-7_1

Faults, errors, and failures

- **Fault:** a defect within the system or a situation that can lead to failure
- **Error:** manifestation (symptom) of the fault - an unexpected behaviour
- **Failure:** system not performing its intended function



Examples

- Year 2000 bug
- Bit flips in hardware due to cosmic radiation in space
- Loose wire
- Aircraft retracting its landing gear while on ground

Effects in time:

Transient/ Intermittent / Permanent

Cognitive gap in AI systems

- Lunar effects on radar in 1960 caused a detection of a large contingent of Soviet missiles heading towards USA

[Brain Cantwell-Smith 2019]

- Today, the technical and ethical considerations of leaving safety-critical decisions to autonomous AI systems are even more crucial


Dependability techniques

Four approaches [IFIP 10.4]

1. Fault prevention
2. Fault removal
3. Fault tolerance
4. Fault forecasting



Next
lecture



Let's look at an
example!

Google's 100 min outage

September 2, 2009:

A small fraction of Gmail's servers were taken offline to perform routine upgrades.

“We had slightly underestimated the load which some recent changes (ironically, some designed to improve service availability) placed on the request routers.”



Fault tolerance

Fault tolerance

- Means that a system provides a degraded (but acceptable) function
 - Even in presence of faults
 - During a period defined by certain **fault model** (i.e. assumptions)
- Foreseen or unforeseen?
 - Fault model describes the foreseen faults

Fault models

- Leading to **Node** failures
 - Crash
 - Omission
 - Timing
 - Byzantine
- Distributed systems can also have **Channel** failures
 - Crash (and potential partitions)
 - Message loss
 - Message delay
 - Erroneous/arbitrary messages

Run-time error management

- Detection: By program or its environment
- Mitigation:
 - Fault containment by architectural choices
 - Fault tolerance using redundancy
 - in software (redundancy in space or time)
 - in hardware
 - in data

Redundancy

From D. Lardner: Edinburgh Review, year 1824:

”The most certain and effectual check upon errors which arise in the process of computation is to cause the same computations to be made by separate and independent computers; and this check is rendered still more decisive if their computations are carried out by different methods.”*

** people who compute*

Static redundancy

Used all the time (whether an error has appeared or not), just in case...

- SW: N-version programming
- HW: Voting systems
- Data: Parity bits, checksums

Dynamic redundancy

Used when error appears and specifically aids the treatment

- SW:
 - Space: Exceptions, Rollback recovery
 - Time: Re-computing a result
- HW: Switching to back-up module
- Data: Self-correcting codes

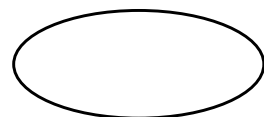
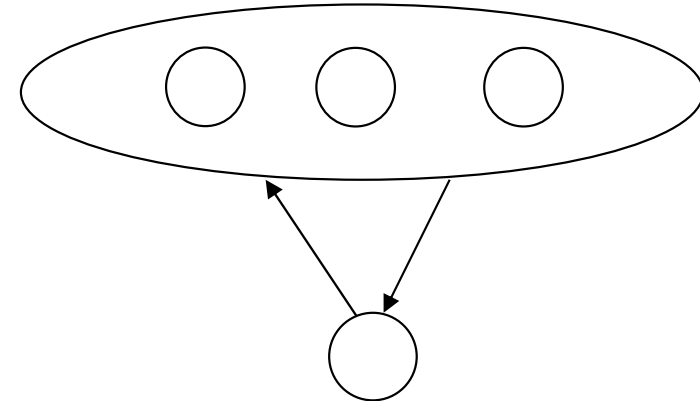
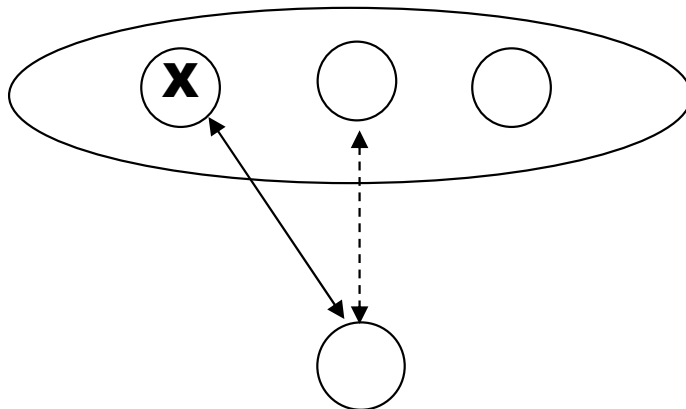
App crashes

- Study of 1800 Android apps showed that 19% of crashes were caused by not handling exceptions

<http://dl.acm.org/citation.cfm?id=2597089>

Server replication models

- Passive replication
- Active replication



Denotes a replica group

Dependability topics



Lectures 7 - 9 cover theory and practical examples

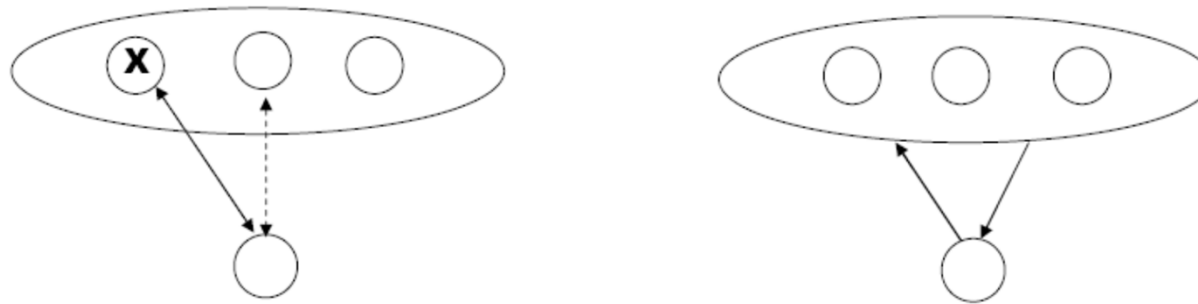
- Basic notions of dependability and redundancy in fault-tolerant systems
- Fault tolerance:
 - Relating faults/redundancy to distributed systems from lectures 4-6
 - Relating timing and fault tolerance

Lecture 8: Adds industrial perspective

Lecture 9: Fault prevention and design aspects

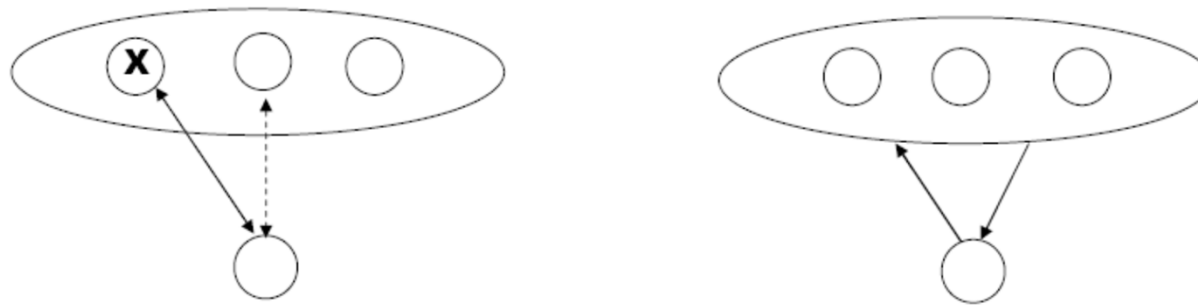
Redundancy and agreement

Underlying mechanisms



- Active replication
 - Replicas must be deterministic in their computations (same output for same sequence of inputs)
 - Relies on group membership protocol: who is up, who is down? Only servers that are up need to have same state

Underlying mechanisms



- Passive replication
 - Primary – backup: brings the secondary server up to date when the primary fails
 - After each “write”, or periodically?

For high availability

- In both cases, servers need to
 - Respect (some) message ordering
- Implicit **agreement** among replicas

Recall: Interaction models

... in distributed systems from lecture 5

- Sharing state information at two servers needs a notion of time or event ordering
- The two possible models
 - Synchronous: related process/clock rates at different nodes, and bounded message delay
 - Asynchronous: related events and their partial order

The consensus problem



Assume that we have a reliable broadcast

- Processes p_1, \dots, p_n take part in a decision
- Each p_i *proposes* (and broadcasts) a value v_i
 - e.g. application state info
- All non-faulty processes *decide* on a common value v that is equal to one of the proposed values

Desired property

- No two non-faulty processes decide differently
(Agreement)

Algorithms for consensus have to be proven to have this property



Basic impossibility result

There is no deterministic algorithm solving the consensus problem in an asynchronous distributed system with a single *crash* fault

[Fischer, Lynch and Paterson 1985]



On the other hand...

Real world problems

- Nodes need to agree on a decision but some nodes are faulty
- Faulty nodes can act in an arbitrary way (can be malicious)

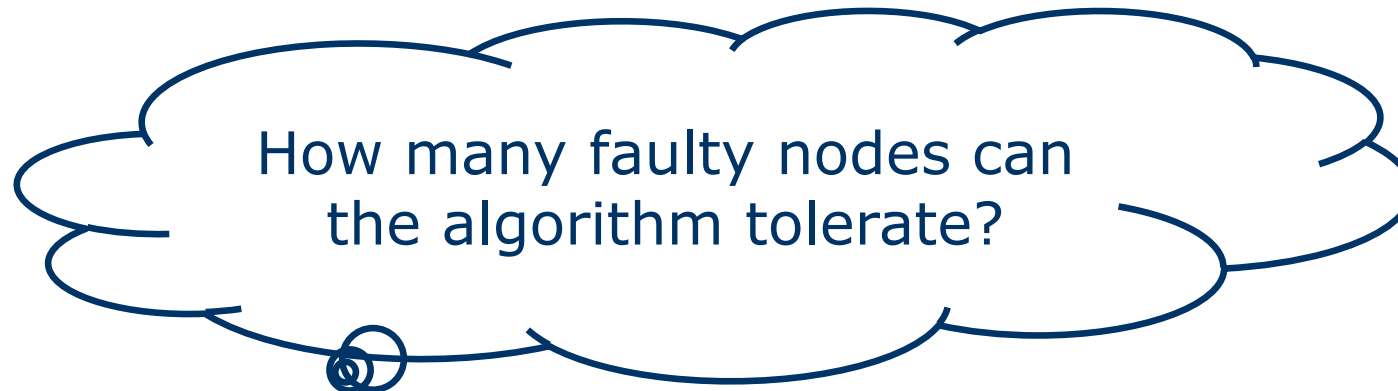


Byzantine Agreement Problem

- Agreement in presence of a harder fault model (Byzantine/arbitrary) is solvable!



- But given the presence of synchrony
[Pease, Shostak and Lamport 1980]

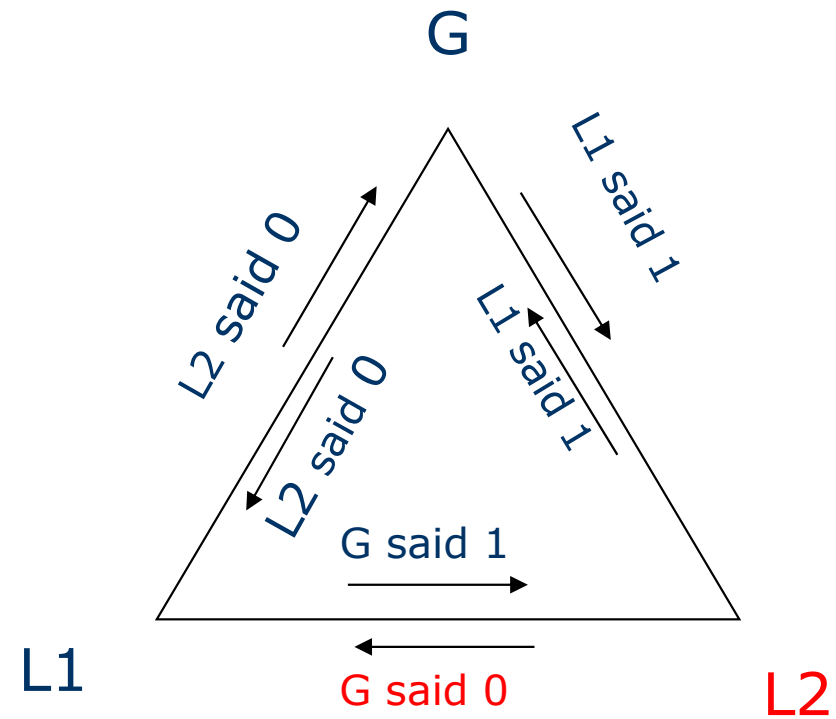
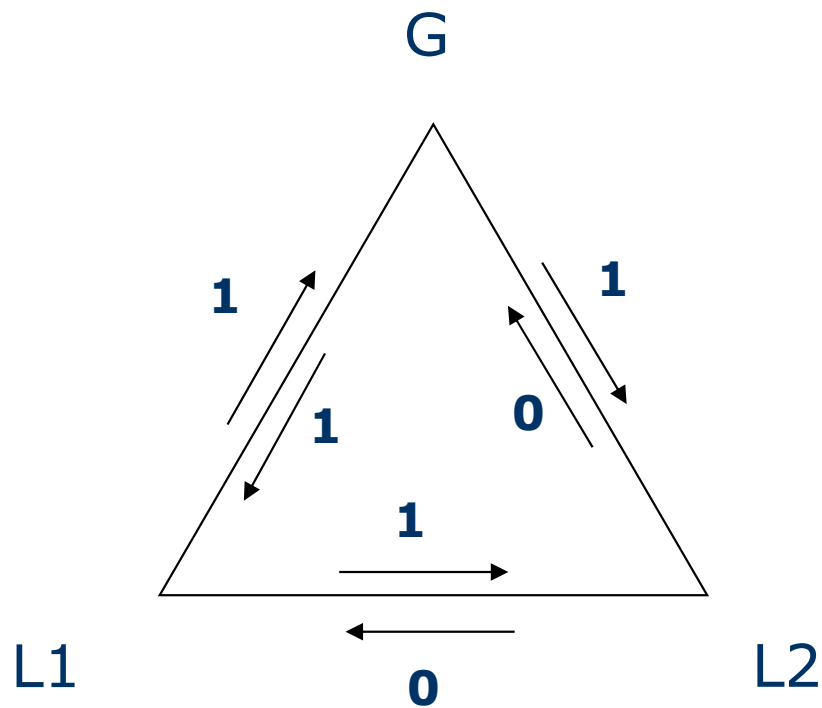


We know since 1980...

- Theorem: There is an upper bound f for the number of byzantine node failures compared to the size of the network N , $N \geq 3f+1$
- Given a $f+1$ round algorithm for solving consensus in a synchronous network
- Here:
 - We just demonstrate that $3f$ nodes would not be enough for agreement!

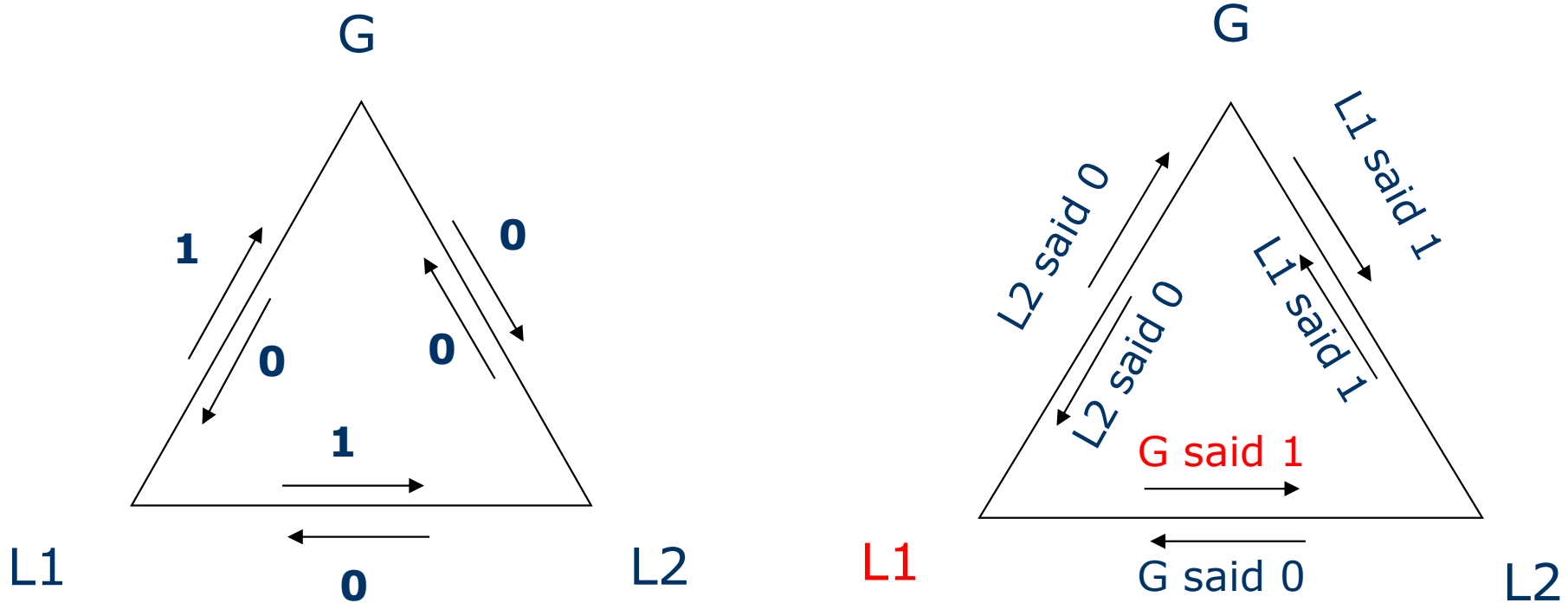
Scenario 1

- G and L1 are correct, L2 is faulty



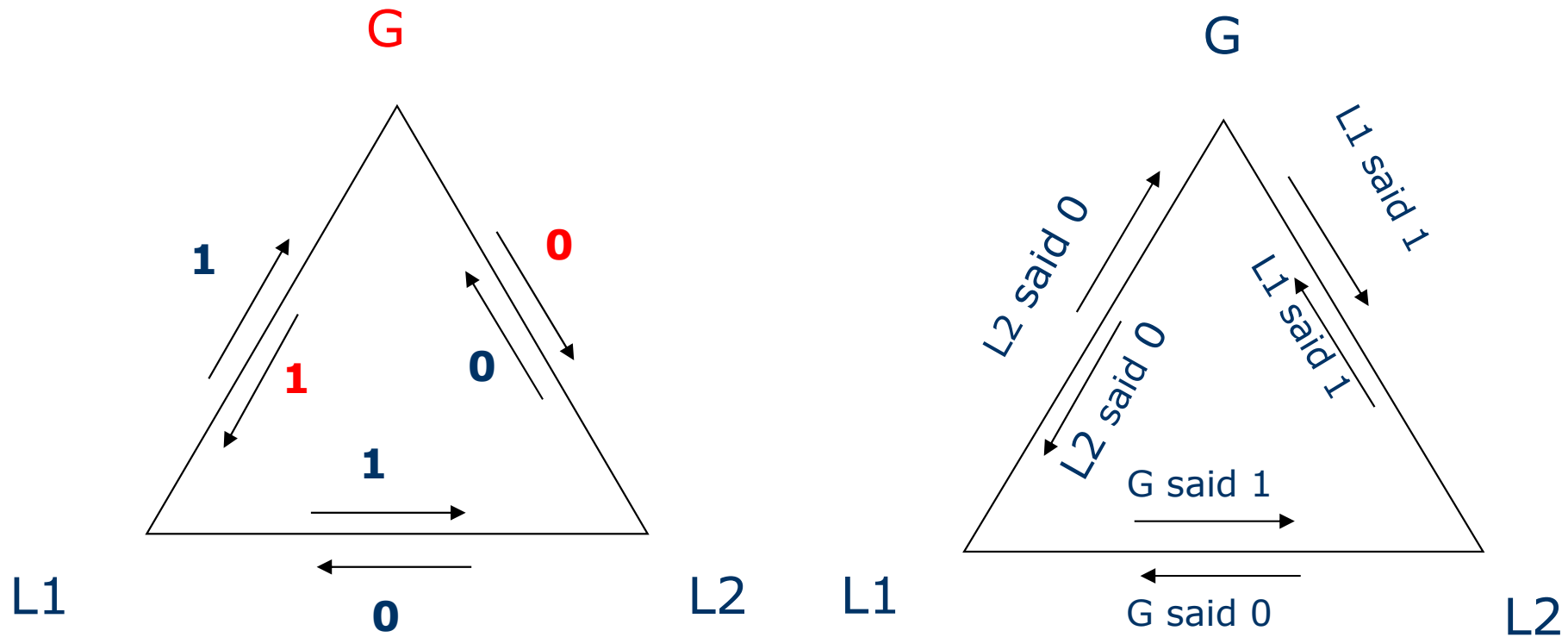
Scenario 2

- G and L2 are correct, L1 is faulty



Scenario 3

- The general is faulty!



2-round algorithm

... does not work with $f=1$, $N=3$!

- Seen from L_1 , scenario 1 and 3 are identical, so if L_1 decides 1 in scenario 1 it will decide 1 in scenario 3
- Similarly for L_2 , if it decides 0 in scenario 2 it also decides 0 in scenario 3
- L_1 and L_2 do not agree in scenario 3!

Summary

- Dependable systems need to *justify* why services can be relied upon
- To tolerate faults, we normally deploy replication
- Replication needs (some kind of) agreement on state
- To prove correctness of agreement protocols in distributed systems we require explicit assumptions on faults (fault model), and (some) synchrony
- A fault model also helps to apply the “right” remedy

Dependability topics



Lectures 7 - 9 cover theory and practical examples

- Basic notions of dependability and redundancy in fault-tolerant systems
- Fault tolerance:
 - Relating faults/redundancy to distributed systems from lectures 4-6
 - Relating timing and fault tolerance

Lecture 8: Adds industrial perspective

Lecture 9: Fault prevention and design aspects

Timing and fault tolerance

Timing and fault tolerance

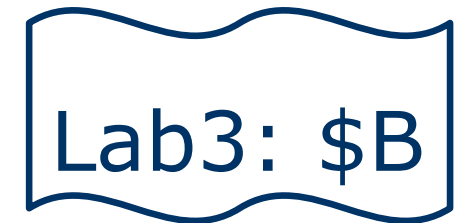
- We saw that support for bounded delays helps fault tolerance
 - We know that consensus is solvable under the synchronous model
- How does fault tolerance affect timing?

Recall: the TTP bus

- Nodes have synchronised clocks
- The communication infrastructure (collection of CCs and the TTP bus communication) guarantees a bounded time for a new data appearing on the communication interface of a (receiving) node (its CNI)
- That's why membership protocol could be implemented: Agreeing on who has crashed!

Exercise: other fault models

- What are the faults that you can think of in a system connected by CAN or TTP?
 - Node related faults
 - Channel related faults
- How does CAN and TTP respectively provide help to detect the errors?



Timing and fault tolerance

- We saw that support for bounded delays helps fault tolerance
 - We know that consensus is solvable under the synchronous model
- How does fault tolerance affect timing?
 - Implementing fault tolerance mechanisms (in turn) affects application timeliness

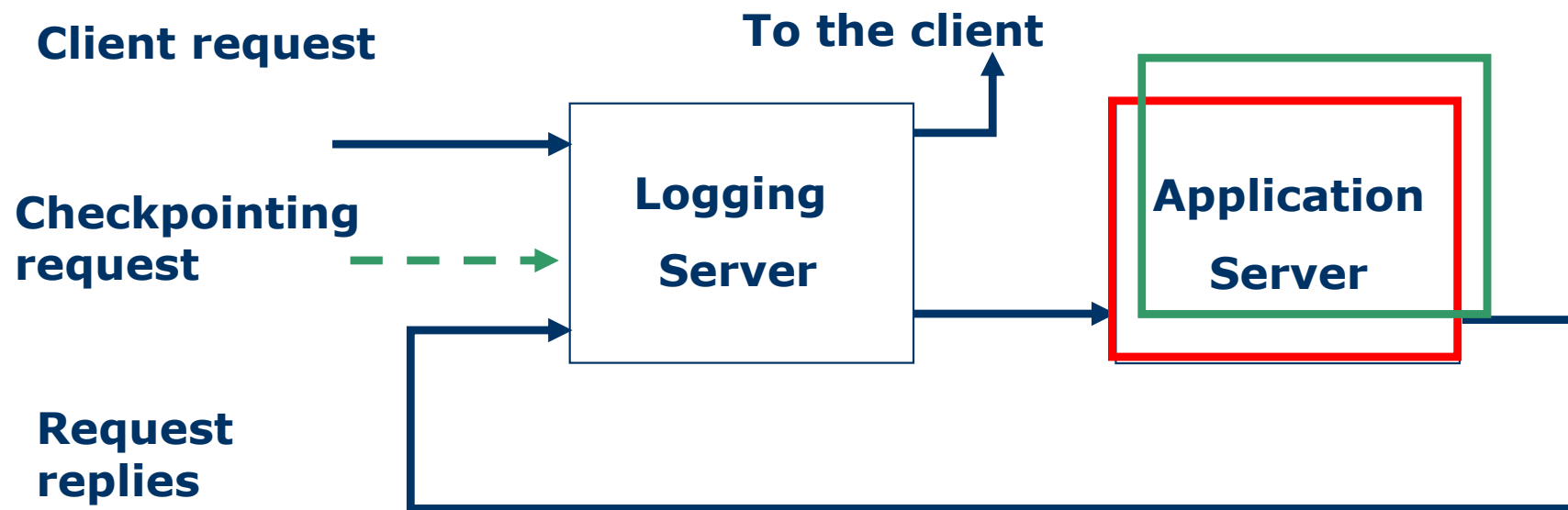
Timing overhead

- Does fault tolerance cost time? Why?
- Two examples:
 - Redundancy in time: Fault-tolerant scheduling
 - Redundancy in space: Replicated server, checkpointing

Fault-tolerant scheduling

- Assume we are running RMS
- How can we deal with *transient* faults that lead to a *process* crash or immature termination?
- We can reschedule the process before its deadline!
 - Need to allow more than the (non-faulty) C_i for each process during analysis

Passive replication: checkpointing



- Optimal checkpointing interval for achieving highest availability?

Replication management *is* complicated

- Google, Dec 2023:

<https://arstechnica.com/gadgets/2023/12/google-calls-drive-data-loss-fixed-locks-forum-threads-saying-otherwise/>

Timing and fault prevention

Prevention has also an impact

- Note that fault prevention has also an impact on timing at run-time!
 - Access control to stop unauthorised access/alteration
 - Authentication
 - Encryption, and so on...
- For some power control systems the response times are in the order of ms!



Questions?

<http://www.ida.liu.se/~TDDD07/>