Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica
Systems, Robotics and Vision Research Group

Dynamic Reconfiguration in Dependable Distributed Embedded Systems

Julián Proenza Arenas

On the concept of DR

- Use of DR is accorded to provide benefits in areas such as QoS or Dependability
- Whereas the role of DR in the area of QoS seems quite clear, it is not so clear in the context of Dependable Systs
- Dependable Systs are usually built using Fault Tolerance mechanisms. These often involve some kind of reconfiguration, e.g.
  - passivate some faults by disconnecting the affected nodes,
  - substitute a faulty module by a spare,
  - add some members to a group of nodes in order to preserve the amount of redundancy in replicated schemes, etc.

A definition for DR?

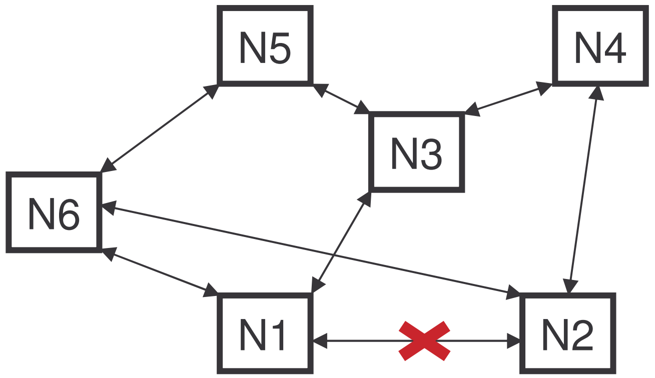
- A clear **definition** of DR in the context of Dependable Systems will be very useful in order to differentiate it from *normal* fault tolerance

- Also a **taxonomy** will help, since DR is a general concept being used to identify a wide variety of mechanisms

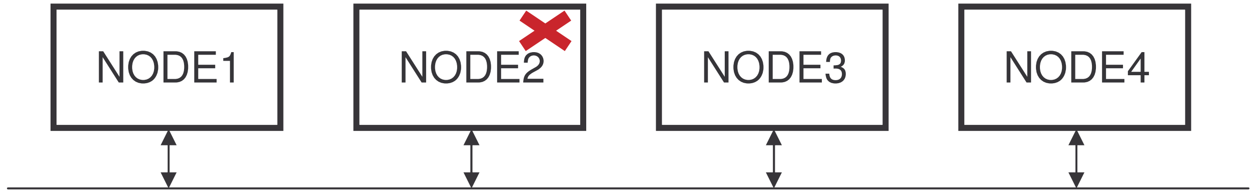
- Regarding the definition of DR, it seems that some specific and differentiating attributes should be included.
  - DR is commonly associated to an efficient use of the already available resources without adding specific redundancy
  - DR is supposed to imply a (high) flexibility-adaptability in the system

Use of already available resources

- This idea of using the already available resources relates DR to some low-cost FT approaches such as the use of Unintentional Redundancy, which is usually available in all kind of systems (particularly in distributed ones)
  - Some interconnection topologies have several paths to connect each node with the rest of the system



  - In Distributed Systems, the presence of multiple nodes opens room to the reallocation of faulty node’s tasks



UR and DR limitations

- But obviously the use of Unintentional Redundancy has a limited capacity for fault tolerance since…
  - Depending on the system, the available redundancy can be not enough to tolerate some faults. Single points of failure may exist
  - In many cases the reconfiguration causes a graceful degradation (the system is no longer fully functional)
- Therefore DR (using UR) is suitable to improve the general dependability of systems but is probably not so well suited to reach the high levels of dependability that are pursued by fault-tolerant architectures
  - Moreover, DR usually takes time, potentially causing the system to be temporarily down. This is not suitable for some applications

Addition of resources for DR

- In fact some resources have to be added to a system for it to be able to perform DR, e.g. middleware
- As will be seen later, some hardware additions would provide an advantageous support for proper DR
- Too many additions would deviate from the initial target of achieving low cost by efficiently using the available resources
- A trade-off has to be found between cost and functionality

DR means flexibility…

- As indicated before, DR is supposed to imply a (high) flexibility in the system

- But, flexibility also means complexity
  - The system has to be able to react in front of various situations and to adapt to the ever changing reality
  - This is a new functionality that is usually implemented in the form of middleware
  - The communication channel also has to transport an increased number of messages

…complexity and overhead

- We have to face the fact that the computational overhead caused by middleware and the communication overhead caused by the increased number of messages are not acceptable for many distributed embedded systems

- FURTHERMORE, more complexity means less reliability…

Facets of unreliability in DR

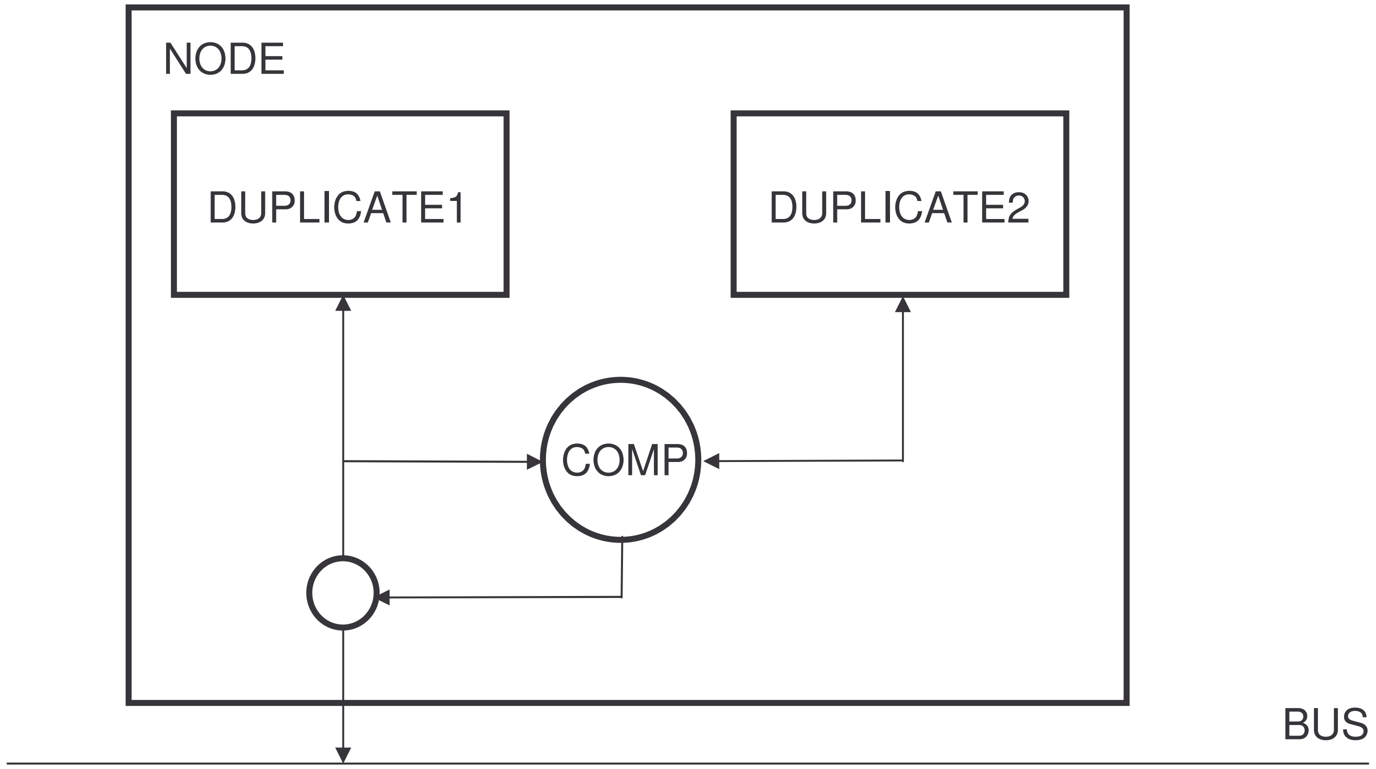
- Increased number of scenarios to be taken into account
  - Caused by faults
  - Caused by the multiple possible configurations of the system
- More difficult systemwide integration of fault tolerance
  - Prevent improper interactions among non-faulty subsystems
  - Prevent improper interference among concurrently active recovery (reconfiguration) algorithms
- More difficult qualitative evaluation
  - E.g. model checking suffers an increased state space size

Suggested solution

- Use techniques to reduce the no. of potential scenarios
- These techniques are introduced in the form of hardware additions that prevent specific error situations as close as possible to the faults that generate them
  - Since errors are not propagated to upper layers of the architecture, the middleware does not have to deal with them
- Some examples:
  - Restrict the failure semantics of the nodes: if arbitrary failures are not possible, the rest of the nodes do not have to deal with them
  - Have communication protocols providing consistency services: the communication channel does not cause additional errors

Node failure semantics restriction

- Using duplication with comparison in the design of the nodes’ circuitry: crash failure semantics

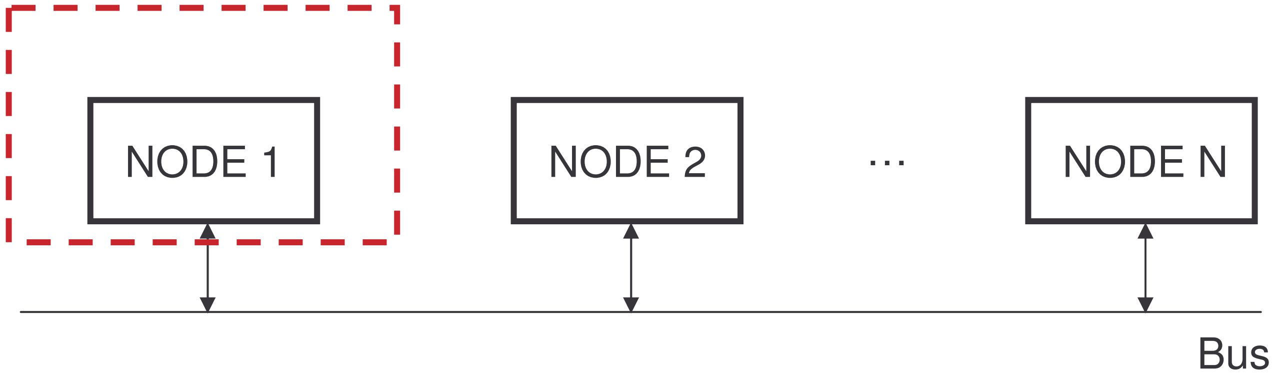


Consistent communication services

- For instance: Atomic Broadcast
- Properties (informal):
  - **Consistency:** Any message is either received by all nodes or by none
  - **Total Order:** All nodes receive all messages in the same order
- Some protocols claim that they provide this service: e.g. Controller Area Network (CAN)

Error containment boundaries

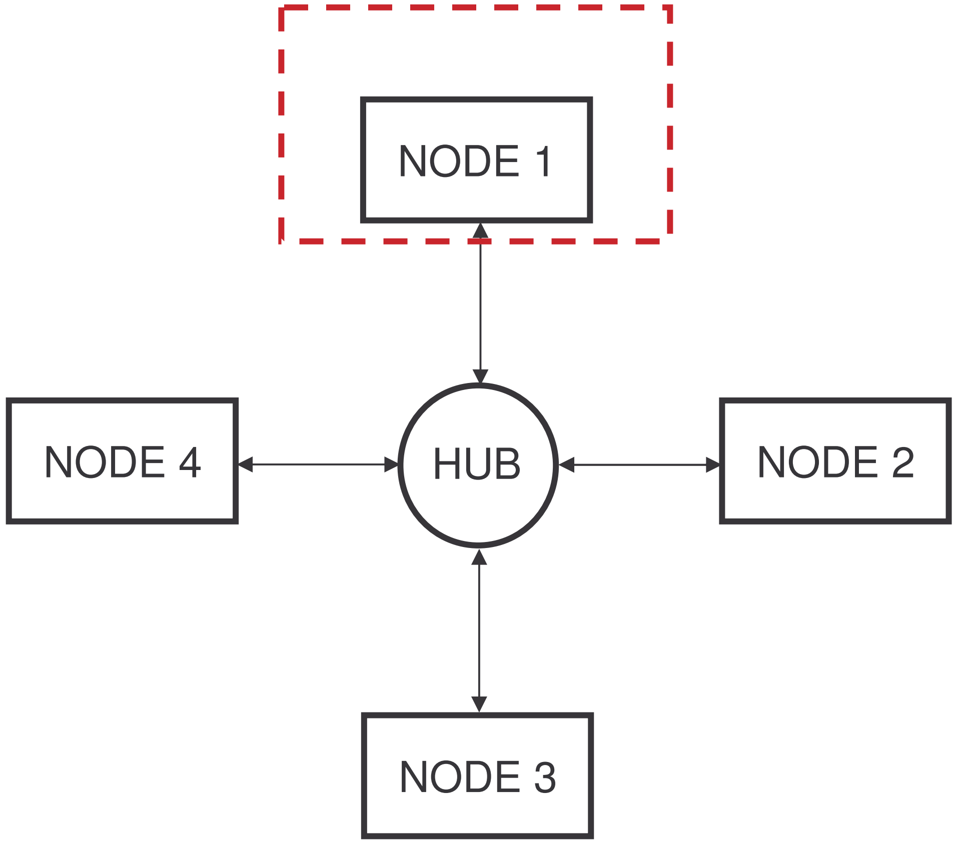
- By combining the previous techniques, error containment boundaries can be defined



- Also the rest of the nodes have to be ready to cope with the omissions caused by crashed nodes

Cheaper containment boundaries

- By using star topologies instead of buses



Contributions of our research group

- We have designed and implemented this kind of mechanisms specially tailored to the CAN protocol
- Nodes with restricted failure semantics (duplication with comparison)
- A modification to the CAN protocol that truly provides A.B. : MajorCAN protocol
- Active hubs for star topologies: CANcentrate and ReCANcentrate

Conclusions (1)

- A clear **definition** of DR in the context of Dependable Systems will be very useful in order to differentiate it from *normal* fault tolerance
- DR seems to imply an efficient use of already available resources (without additional redundancy) and a high flexibility in the system operation
- Only using already available resources has a limited fault tolerance potential
- A trade-off has to be found between cost and additional functionality for reconfigurability (in middleware)

Conclusions (2)

- Added flexibility causes additional overhead, complexity and therefore unreliability
- We suggest: to use techniques to reduce the no. of potential scenarios that the middleware has to deal with
- These techniques are introduced in the form of hardware additions that prevent specific error situations as close as possible to the faults that generate them
  - Restrict the failure semantics of the nodes
  - Have protocols with consistent communication services
  - Define error containment boundaries (cheaper with stars)
- Our group has developed some of these for CAN

Universitat de les Illes Balears
Departament de Ciències Matemàtiques i Informàtica
Systems, Robotics and Vision Research Group

Dynamic Reconfiguration in Dependable Distributed Embedded Systems

Julián Proenza Arenas

julian.proenza@uib.es