Fault Tolerance and Exception Handling in Large-Scale MASs

Panel: Dependability and QoS in Large-Scale MASs

Alexander Romanovsky

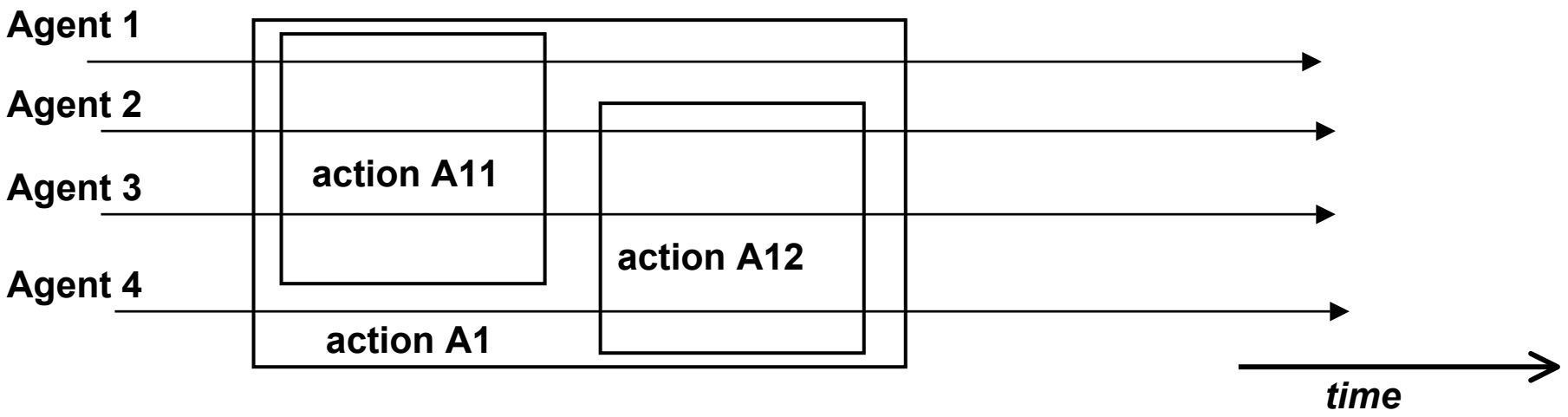
School of Computing Science
University of Newcastle upon Tyne
UK
alexander.romanovsky@ncl.ac.uk

Fault Tolerance

- Fault tolerance: error detection and error recovery
- Types of faults in large-scale MASs: agents' mistakes, environmental faults, mismatches, online upgrades/changes of the environment or other agents, application developers' mistakes, users' mistakes, malicious faults and all types of errors propagated from the underlying levels (OS, middleware, hardware) when they fail to deliver the required services
- This this requires much more than software tolerance of hardware faults (ACID transactions, replications, atomic broadcasts, etc.)
- We need software fault tolerance at the application level that is the level of the large-scale MASs in our case
- Forward error recovery (no rollback)

Fault Tolerance by Exception Handling

- Exception handling as a means
- Separation of the normal and abnormal behaviour: separation of the code and of the flows of control
- Recursive structuring of complex systems. Reduce complexity and add fault tolerance: units of information encapsulation/hiding (i.e. units of system development or system execution) are units of error confinement as well as error recovery
- Multilevel exception handling to limit the scope of recovery
- For a system with cooperating agents we might need atomic actions:



Fault Tolerance by Exception Handling

- Choice of structuring and exception handling techniques depends on many things such as the design paradigm, application types, computational models, types of faults, etc.

- The challenge is to develop novel exception handling techniques suitable for large-scale MASs

- *Definitely not Java or RMI Java exception handling.*
- *Definitely not conventional OO exception handling.*

Specific Characteristics and their Effects

- Agents are *autonomous*. They cannot report exceptions to a higher level (e.g. to a client) but should incorporate all fault tolerance measures
- Agents are *adaptive*, fault tolerance is a form of adaptivity, adaptivity features can be used for providing fault tolerance
- Agents are *interactive*. Very often they have to perform error detection and error recovery in cooperation with a number of other agents (in particular, when errors are not confined in one of them). We need a concept of scope or ***exception context***

Specific Characteristics and their Effects

- Agents are *mobile*. Very special exception handling techniques suitable for mobile systems: agents can leave the location and move to another location, the execution environment and the resources available can change on the fly
- *Asynchronous* communication in large-scale MASs. Decoupling producers and consumers, anonymous communication. Examples: event-based systems, Linda. What if the agent producing a erroneous data moves? Exceptions cannot be treated as the normal events or tuples

Possible Solutions - 1

- In spite of asynchronous communication in large-scale MASs and agents' mobility: *all exceptions have to be handled* (synchronously or asynchronously but without infinite delays).
  - Chase the producer of the event or the tuple causing the exception
  - Create a local handler agent bt only when an agent signals an exception
- Inform several agents about exceptions
- Choose dynamically whom to inform
- Scopes (i.e. exception handling contexts) should include all the agents to be involved in handling, for example, all agents (possibly) contaminated by the error:
  - Defined as all agents in a particular location
  - Dynamically defined by mutual agreements among a number of cooperating agents
  - Use knowledge-management to define it (M.Klein's work)

Possible Solutions - 2

- Agents should always contain *additional information* - called redundant information in fault tolerance - (e.g some sort of specification, list of services they can provide, a description of the behaviour they expect from the environment and other agents, etc.).
- Sometimes this info can be located in some sort of registry or it can be attached to the agents
- Agents should speak the same language - we need ontologies
  - During normal behaviour: it is used for choosing with whom to cooperate, who's service to request, etc.
  - And at the same time this information should be used for error detection and error recovery

• • •

⌘

Coordinated Atomic (CA) Actions

- CA actions is a unified scheme for coordinating complex activities and supporting cooperate error recovery between multiple interacting components:
  - cooperative and competitive concurrency
  - fault tolerance by integrating and extending conversations and transactions
  - conversations enhanced with exception handling are used to control cooperative concurrency and to implement coordinated error recovery
  - transactions are used to maintain the consistency of shared resources in the presence of failures and competitive concurrency
- Nesting. Multiple outcomes. A number of implementations and considerable experience