Towards Dynamic Reconfiguration of Distributed Systems in VDM-RT

Claus Ballegård Nielsen

Aarhus School of Engineering

Agenda

Introduction

Dynamic Reconfiguration Extension

Case Study

Future Work

The What?

- Proposal for an VDM-RT extension made to enable dynamic reconfiguration during the runtime execution.
- Dynamic reconfiguration involves
  - System reconfiguration:
    - Changing the configuration of the network topology
  - Object migration:
    - moving an object from one executable unit to another during while still maintaining the internal state and references of the object.

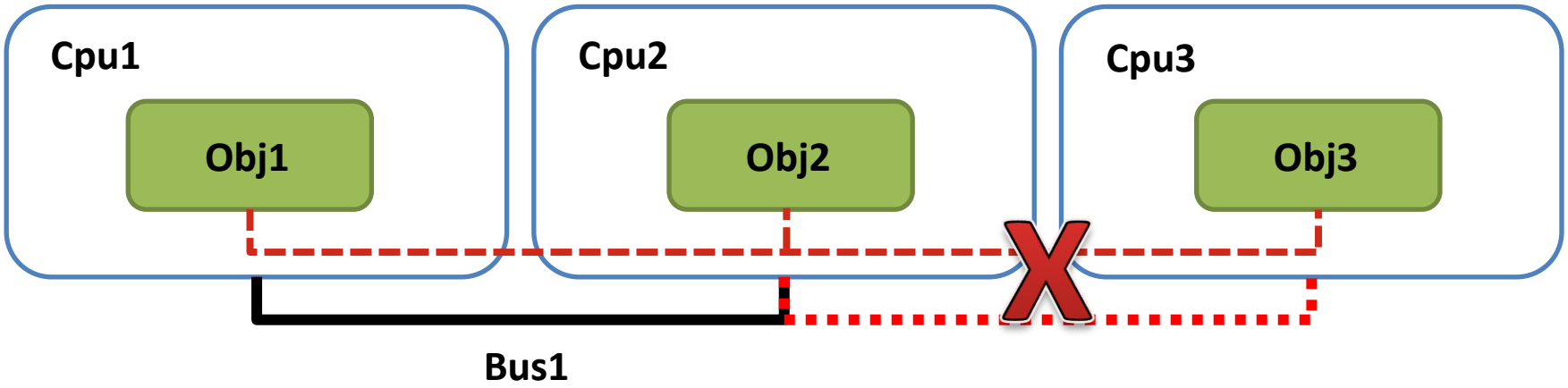
The Why?

- Newer kinds of computing systems have high demands for:
  - Adaptability, scalability and availability
- Because of:
  - Quality of service as a principal requirement
    - Recovery and failover safety measures
  - Ubiquitous behavior
    - Movement, environment change

Deployment in VDM-RT

- VDM-RT allows for the modeling of distributed real-time systems.
- The distributed system architecture is formed on the basis of two predefined classes BUS and CPU.
- The distributed system is defined in the special **system** class**.**
- Deployment is preconfigured and static

Deployment in VDM-RT



Dynamic Reconfiguration Extension

- Introduction of new language constructs to VDM-RT for expressing the dynamic reconfiguration.
- The proposal does not introduce any new classes or language keywords.
- Only new operations to existing predefined classes.
- The new operations are implicitly added to the three classes (**system, BUS, CPU)**

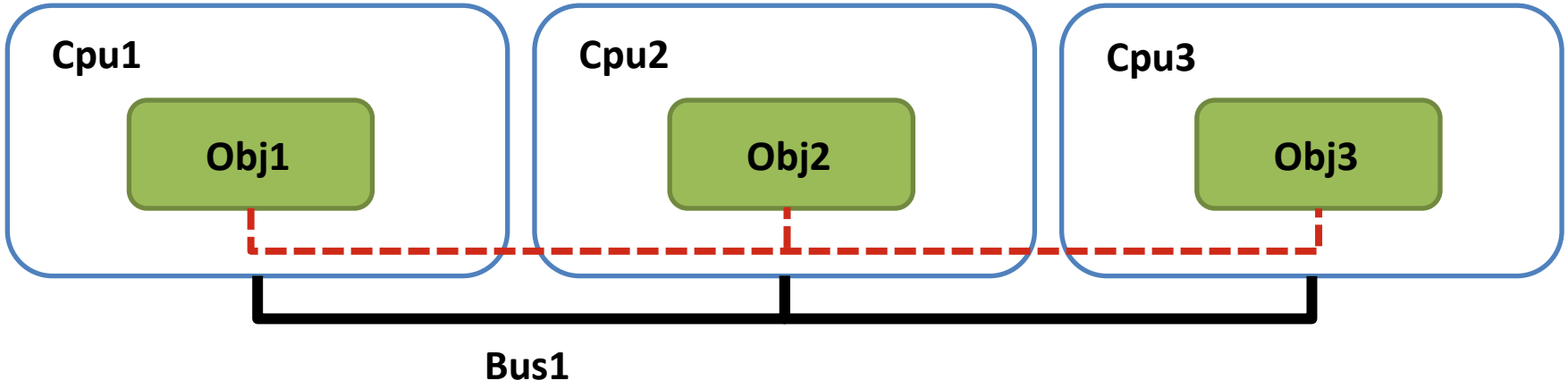
Dynamic Reconfiguration Extension

- 7 new operations
  - 2 for network reconfiguration, 1 for object migration, 4 for system status

Operation	Parameter types	Description
Sys.**ConnectToBus(**obj , bus)	Object * BUS	Connect object with a BUS.
Sys.**DisconnectFromBus**(obj ,bus, idle)	Object * BUS * boolean	Disconnect object from BUS.
Sys.**Migrate**(obj, cpu, idle)	Object * CPU * boolean	Migrate object to a specific cpu
Sys.**ActivityOnBus**(obj, bus)	Object * BUS	Determines the objects current BUS activity
Sys.**IsConnected**(obj, obj)	Object * Object	Are objects connected by a BUS
BusX.I**sConnected**(obj)	Object	Is the object connected to BusX.
CpuX.I**sDeployed**(obj)	Object	Is the object deployed to CpuX

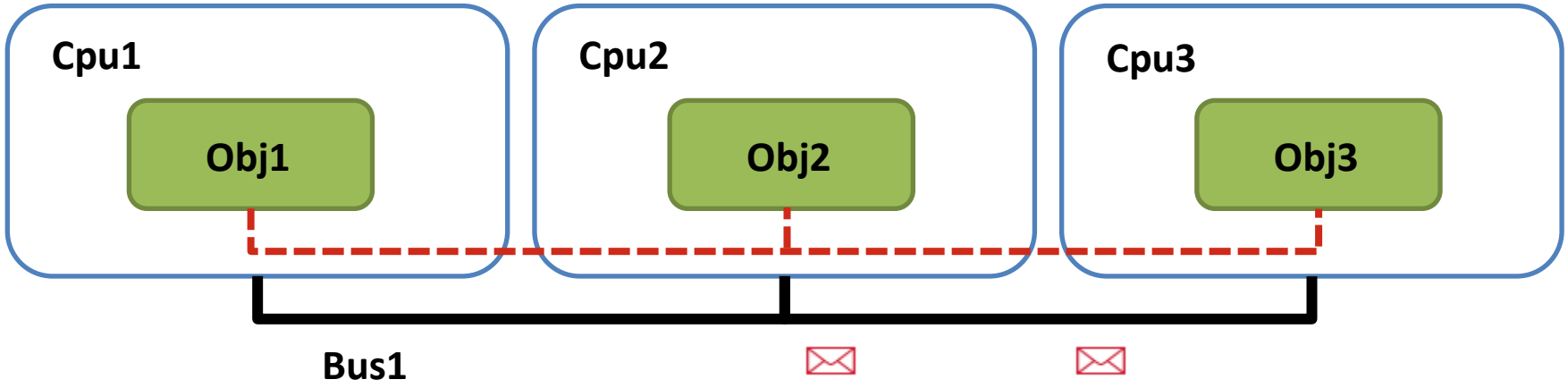
Establishing a new connection

- Sys.**ConnectToBus(**obj3 , bus1)
- Occur as an atomic operation. (uninterruptible, takes no time)
- No event or other type of notification is supplied.
- Only creates a connection from a CPU to a BUS. It does not create object references



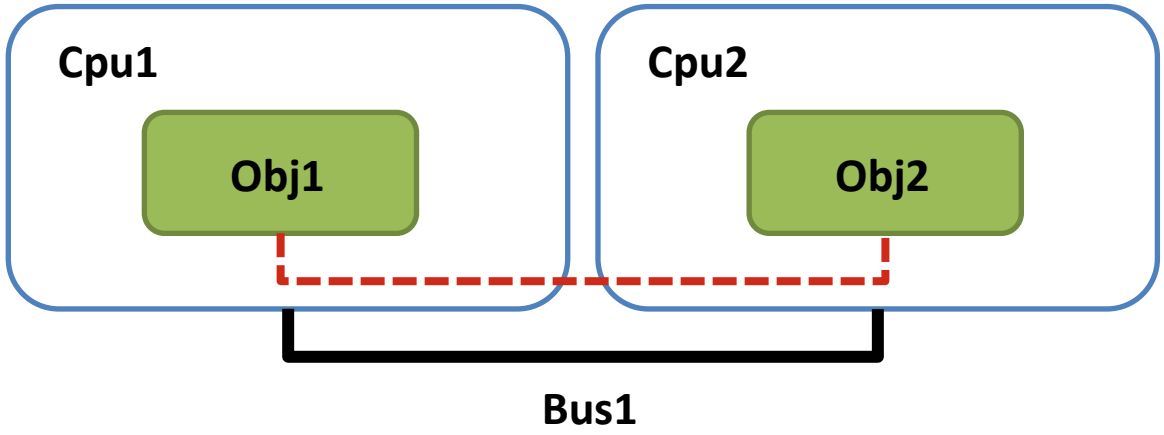
Tearing down a connection

- Atomic operation
- No graceful disconnect or notification
- Current messages will be dropped
- Sys.**DisconnectFromBus**(Obj3 ,bus1, false)
- 3. parameter indicates if the BUS should be idle before the disconnection occurs. Check is before entering atomic transaction

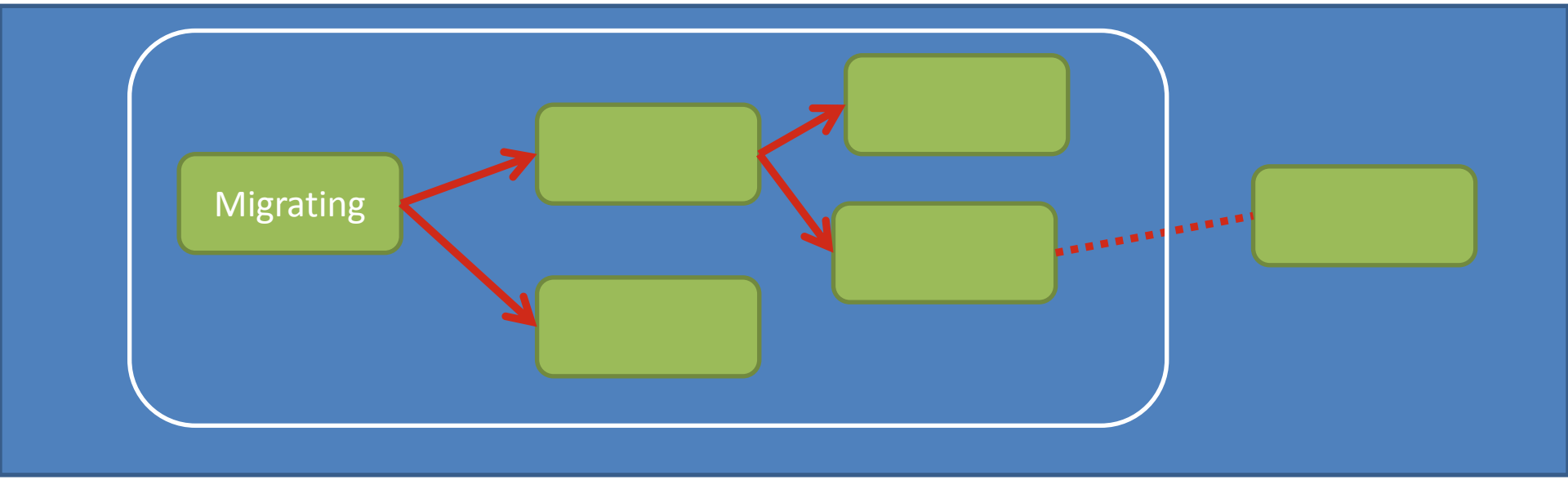


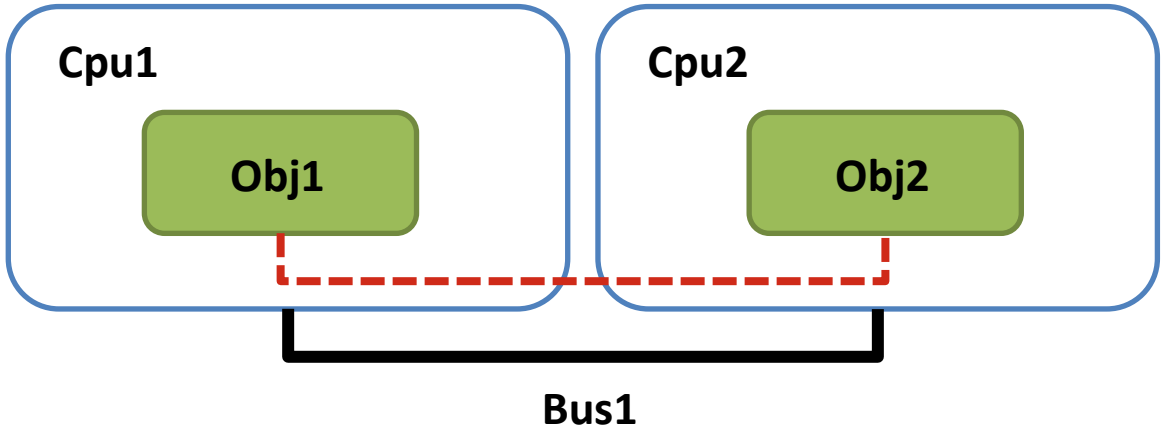
Migrate

- Moves an object to a new CPU in an atomic operation while preserving its identity and state
- Referential integrity is a challenge
  - From the object: All transitive references will be included as part of the object
  - To the object: calls may now occur over a network



Migrate



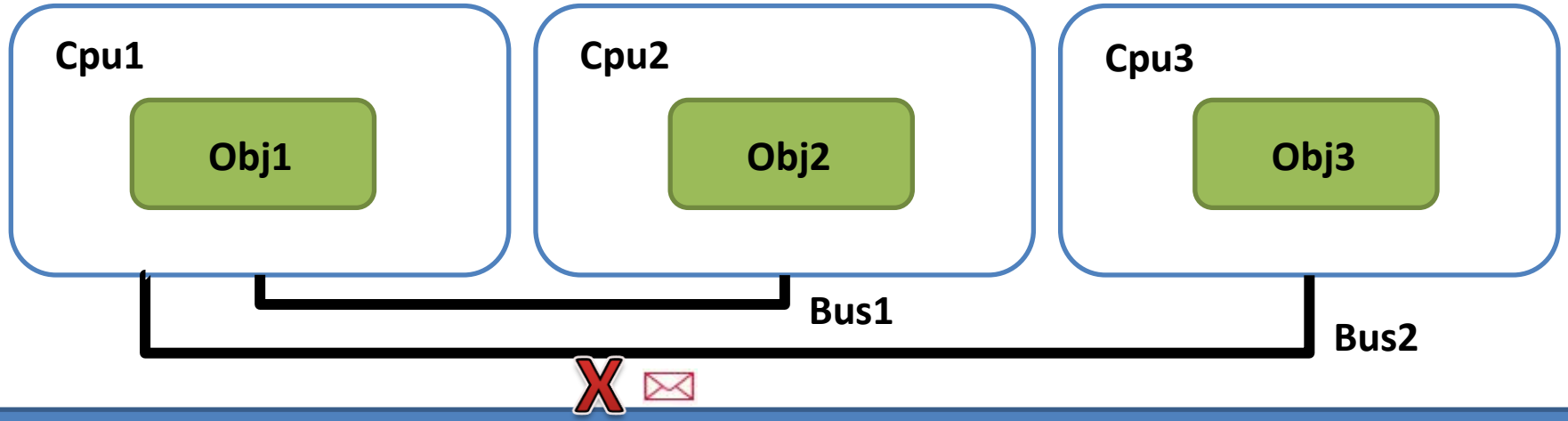


Migrate

- Moves an object to a new CPU in an atomic operation while preserving its identity and state
- Referential integrity is a challenge
  - From the object: All transitive references will be included as part of the object
  - To the object: calls may now occur over a network
- The target CPU must be connected to the same BUS as the source CPU

Migrate

- Sys.Migrate(obj1, cpu2, false);
- Message on the BUS that were send prior to reconfiguration will still be processed after the reconfiguration.
- Unless: the migrating object had communication with remote objects through a second BUS



Determine status

- The introduction of dynamic reconfiguration increases the need for knowledge.

- Sys.**IsConnected**(obj, obj)
- BusX.I**sConnected**(obj)
- CpuX.I**sDeployed**(obj)

Messages on the bus

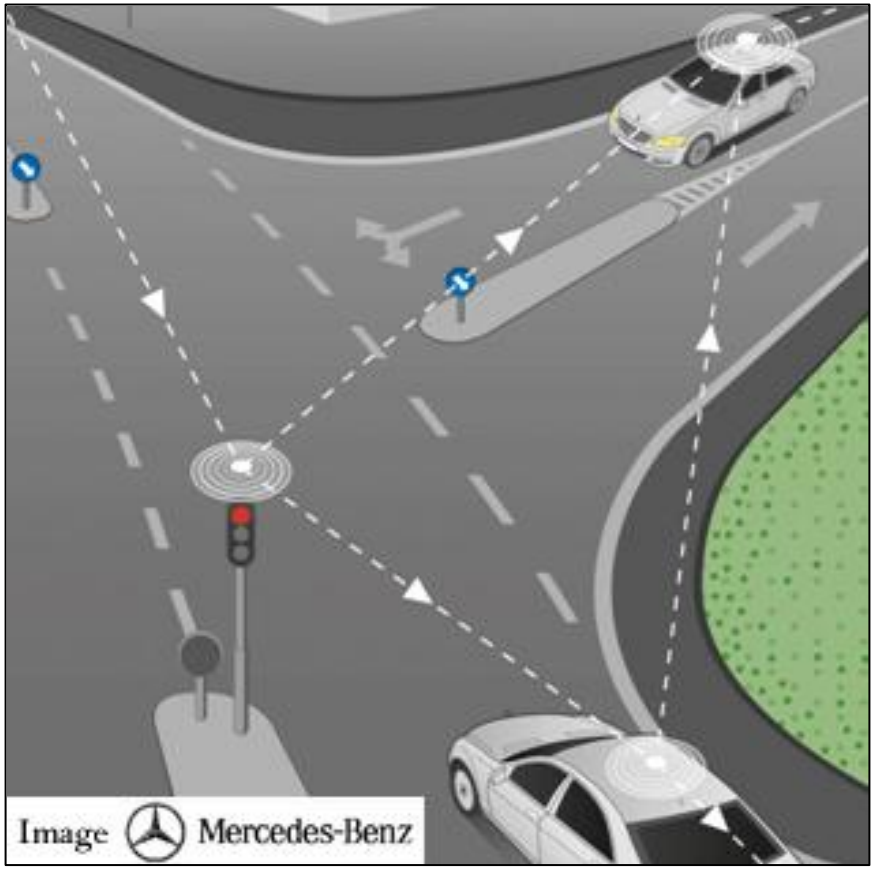
- The ActivityOnBus operation takes an object and a BUS and returns an enumerated type depicting the current activity the given object has on the given BUS.

- Idle:
  - The object has no communication on the bus.
- ProcessingRequest:
  - The object is currently processing an operation invocation by a remote object.
- WaitingForReply:
  - The object is waiting on a reply from an operation invocation on a remote object.
- RequestAndReply:
  - The object is both processing a request and waiting for a reply.

VeMo – a case study

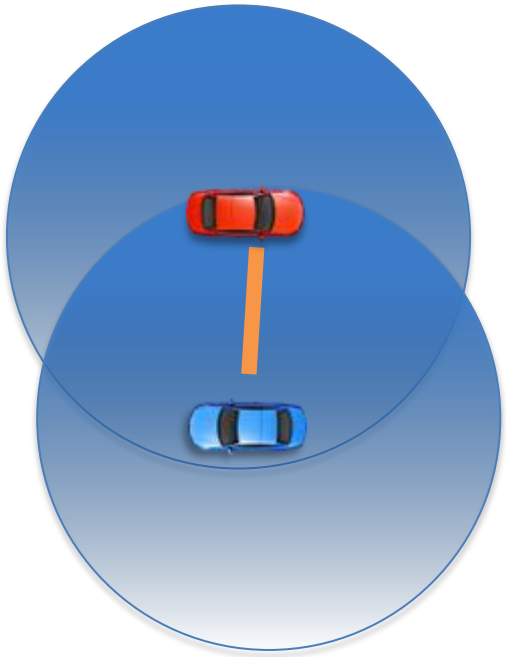
Vehicle Monitoring System

- Improve road safety
- vehicles in near proximity share information through a co-operative wireless network



VeMo – basics

- The model has a controller which calculate when vehicles are in range of each other.
- With static network topology vehicles are constantly connected.



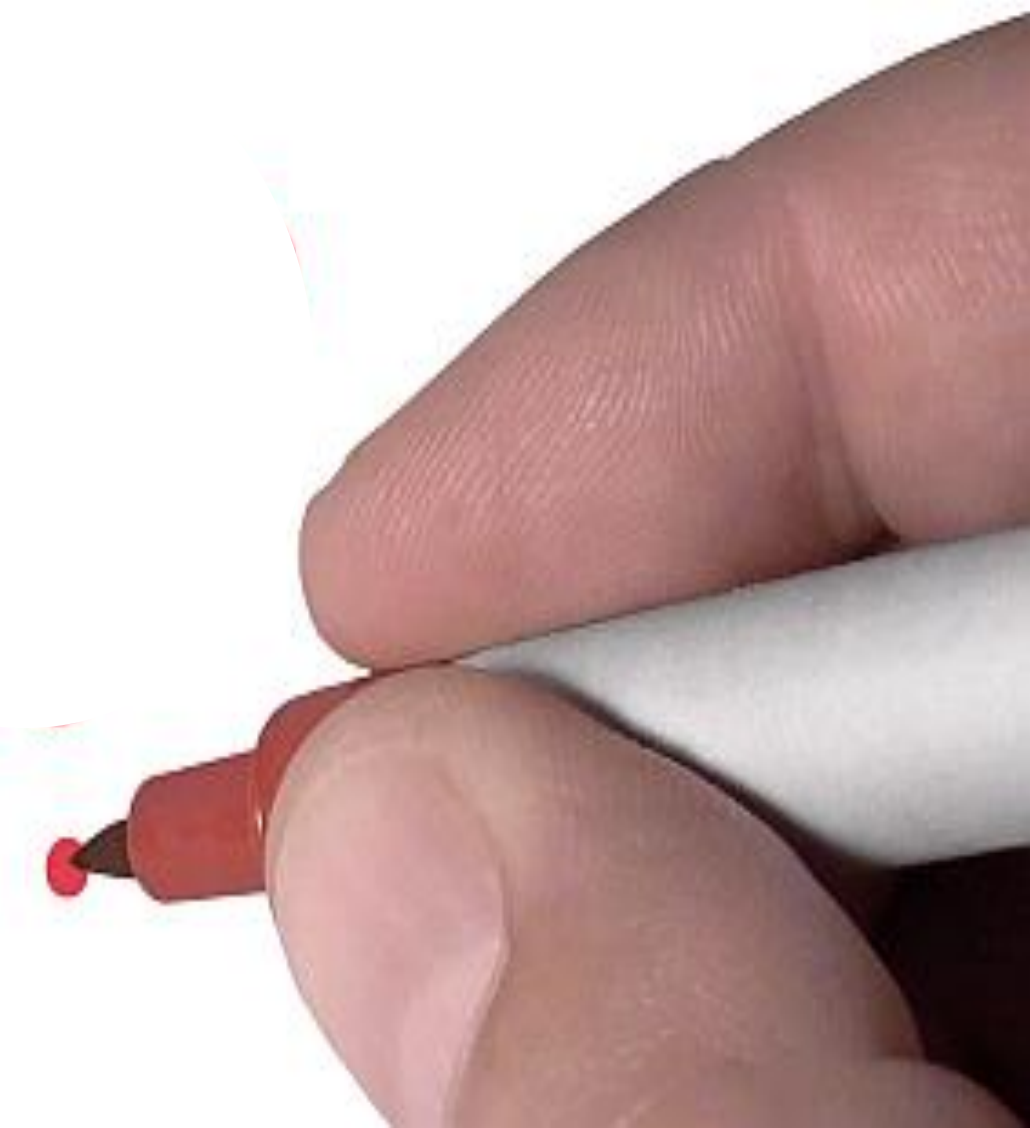
A new challenge arise

- A large number of units with autonomous behavior can create an unpredictable number of network connections.
- Number of BUS connections available is static
  - Add dynamic creation of busses

Future Work

- Tool support for the extension by integration into Overture platform
- Investigation of the necessity for a special construct, such as a policy, for managing the dynamic reconfiguration
- A representation of messages being lost on the BUS needs to be established.
  - return value for synchronous operations.
    - exception handling
    - special type
- Dynamic creation of new BUS class instances

Q & A



clausbn@cs.au.dk