A Toolchain for UML-based Modeling and Simulation of Networked Embedded Systems

Emad Ebeid

PhD Student

Department of Computer Science

University of Verona

Italy

Davide Quaglia

Assistant Professor

Department of Computer Science

University of Verona

Italy

Outline

- Introduction
- UML/Deployment diagrams
- UML/ NW Profile
- Toolchain
  - UML2HIF
  - HIF2UML
  - NSM
  - HIF2SystemC
- Experimental results

Introduction (1)

```
#include <tlm.h>
#include <exception>

#include <scnsl.hh>
#include "MyTask_t.hh"
#include "MyTask_tr.hh"
#include <fstream>
#include <iostream>

using namespace Scnsl::Setup;
using namespace Scnsl::Builtin
using Scnsl::Tracing::Traceabl

int sc_main( int argc, char *
{
 try {



        unsigned int NODESNUMBER_ROW=0;
        if ( argc == 2 )
        {
            std::stringstream ss;
            ss << argv[ 1 ];
            ss >> NODESNUMBER_ROW;

        }

        // Singleton.
        Scnsl::Setup::Scnsl_t * scnsl = Scnsl::Setup::Scnsl_t::get_instance();
```

LoC> 1500

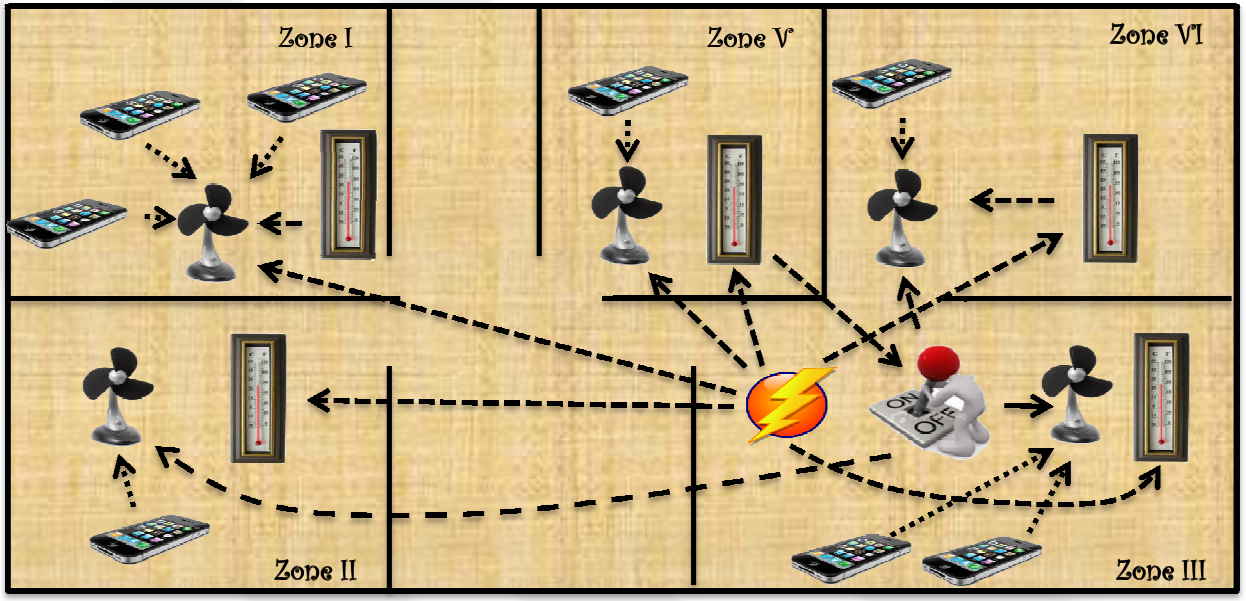
```
////////////////////////////////////////////////////////////////////////////////
// Creating the protocol 802.15.4:
////////////////////////////////////////////////////////////////////////////////
        CoreCommunicatorSetup_t ccoms;
        ccoms.extensionId = "core";
                                                    MAC_802_15_4;
                                                    ccoms );

        Scnsl::Core::Communicator_if_t * mac1;
        ccoms.name = "Mac1";
        ccoms.node = n1;
        mac1 = scnsl->createCommunicator( ccoms );
////////////////////////////////////////////////////////////////////////////////
//Tracing
////////////////////////////////////////////////////////////////////////////////
        // Adding tracing features:
        CoreTracingSetup_t cts;
        cts.extensionId = "core";
        // - Setting the formatter:
        cts.formatterExtensionId = "core";
        cts.formatterName = "basic";
```

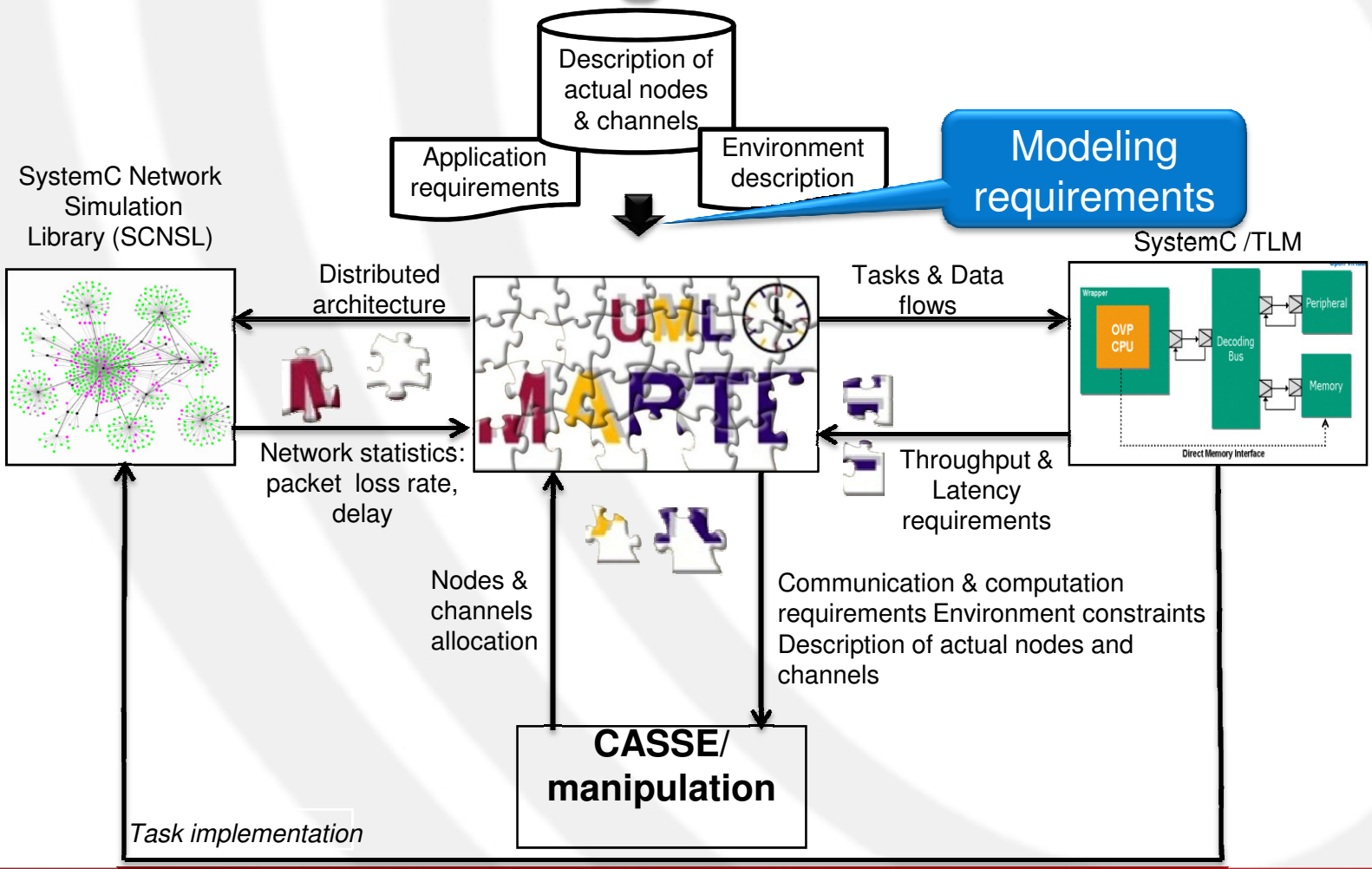
Introduction (2)

Solution:

- **Distributed embedded application** as a single system to be designed
- Start from an abstract Model-Based System Specification



Design flow


Description of actual nodes & channels
Application requirements
Environment description
Modeling requirements
SystemC Network Simulation Library (SCNSL)
Distributed architecture
Tasks & Data flows
SystemC /TLM
UML
MARTE
Wrapper
OVP CPU
Decoding Bus
Peripheral
Memory
Direct Memory Interface
Network statistics: packet loss rate, delay
Throughput & Latency requirements
Nodes & channels allocation
Communication & computation requirements Environment constraints Description of actual nodes and channels
CASSE/ manipulation
Task implementation

Framework requirements

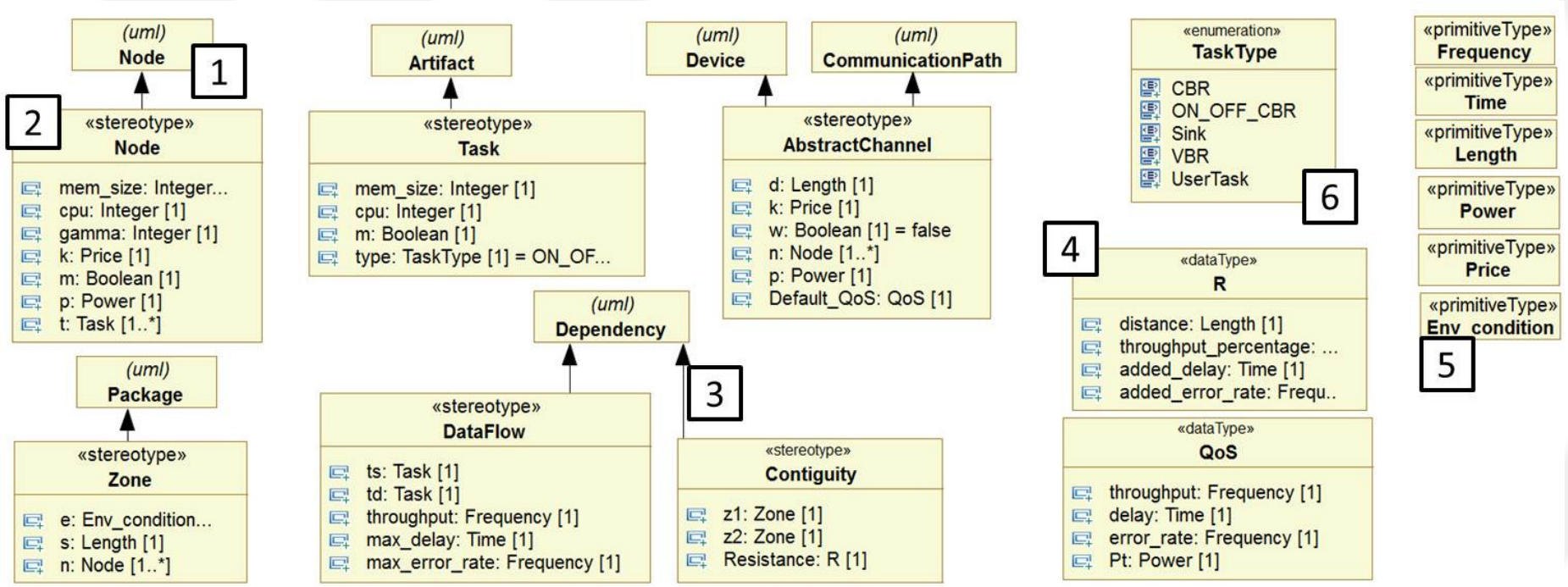
- Intermediate format to represent UML elements
- A set of tools to manipulate and generate executable/simulation code from this format
- Framework for network simulation (e.g. SCNSL)

Modeling requirements

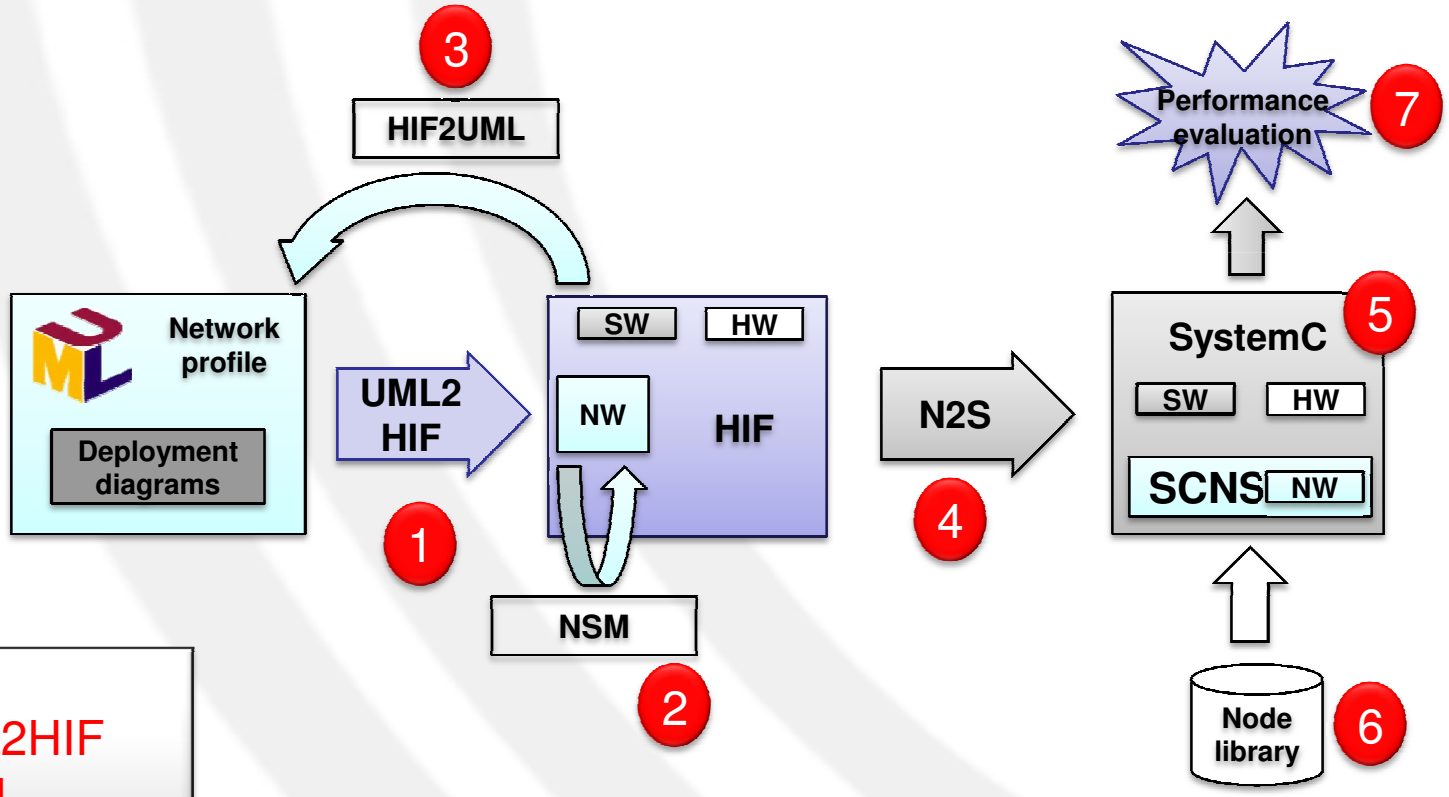
- Graphical modeling language which is closed to user requirements
- New UML profile to add networked embedded systems semantics to this graphical language

UML Network Profile

- Structure of NW profile:



Tools Overview



Tools

1. UML2HIF
2. NSM
3. HIF2UML
4. N2S

HIFSuite

- *HIFSuite* is a set of tools and a library that provide support for modeling and manipulation of **HIF** descriptions
- The Heterogeneous Intermediate Format (HIF) language can be used to build and manipulate descriptions of blocks interconnected together and featuring a finite state machine behavior
- HIF language is currently used to describe HW and SW blocks but it can be used also for the network

UML2HIF

- **UML2HIF** tool is HIF front-end tool for parsing UML diagrams and generating the corresponding HIF description.
- Up to now, the tool generates the complete description of the network from UML/Deployment diagram with NW profile.
- The tool supports wired/wireless communication and the concept of zone

N2S

- **N2S:** (Network to Simulation), is a tool which translates the corresponding HIF description of the NW into a description for the simulation platform
- Currently, the tool generates the SCNSL description from HIF
- In the future, it can support NS-3, OMNET++, etc.

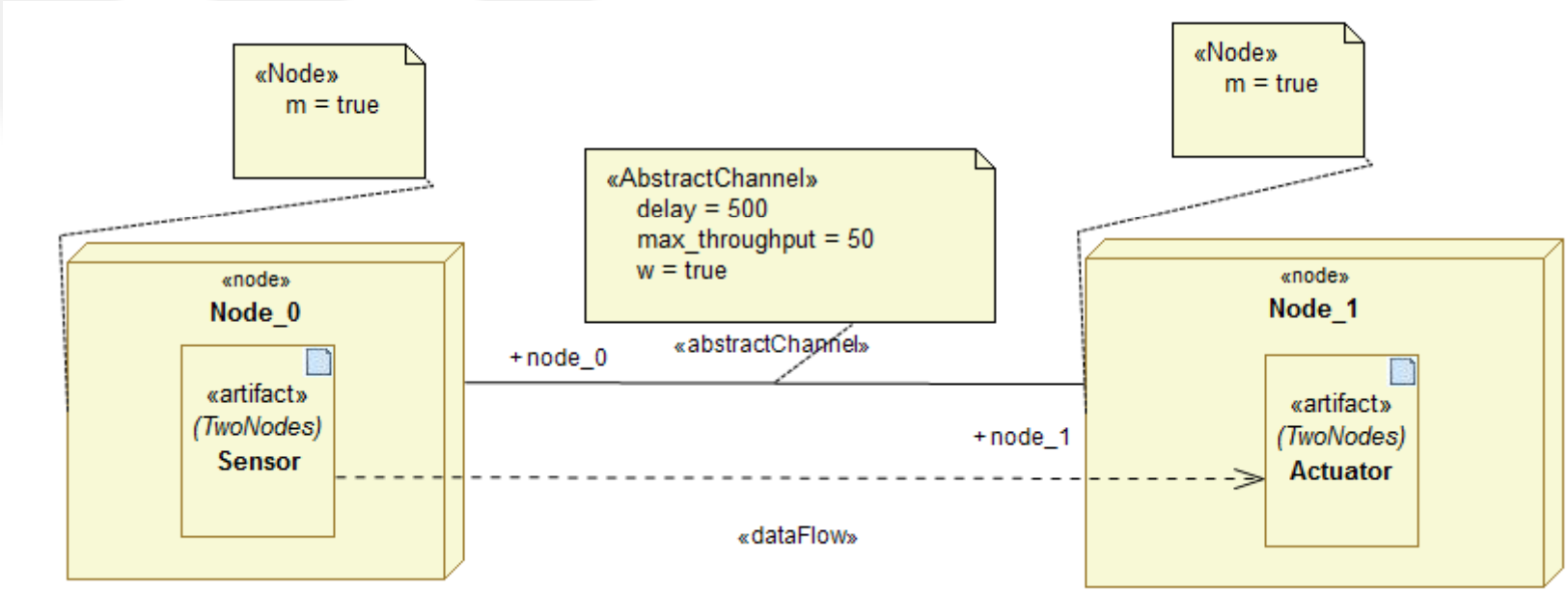
NSM

- **NSM:** (Network Scenario Manipulation), is a tool which manipulate the corresponding HIF description of the NW to generate different network design alternatives
- Currently, the tool supports divide and split channels mechanism

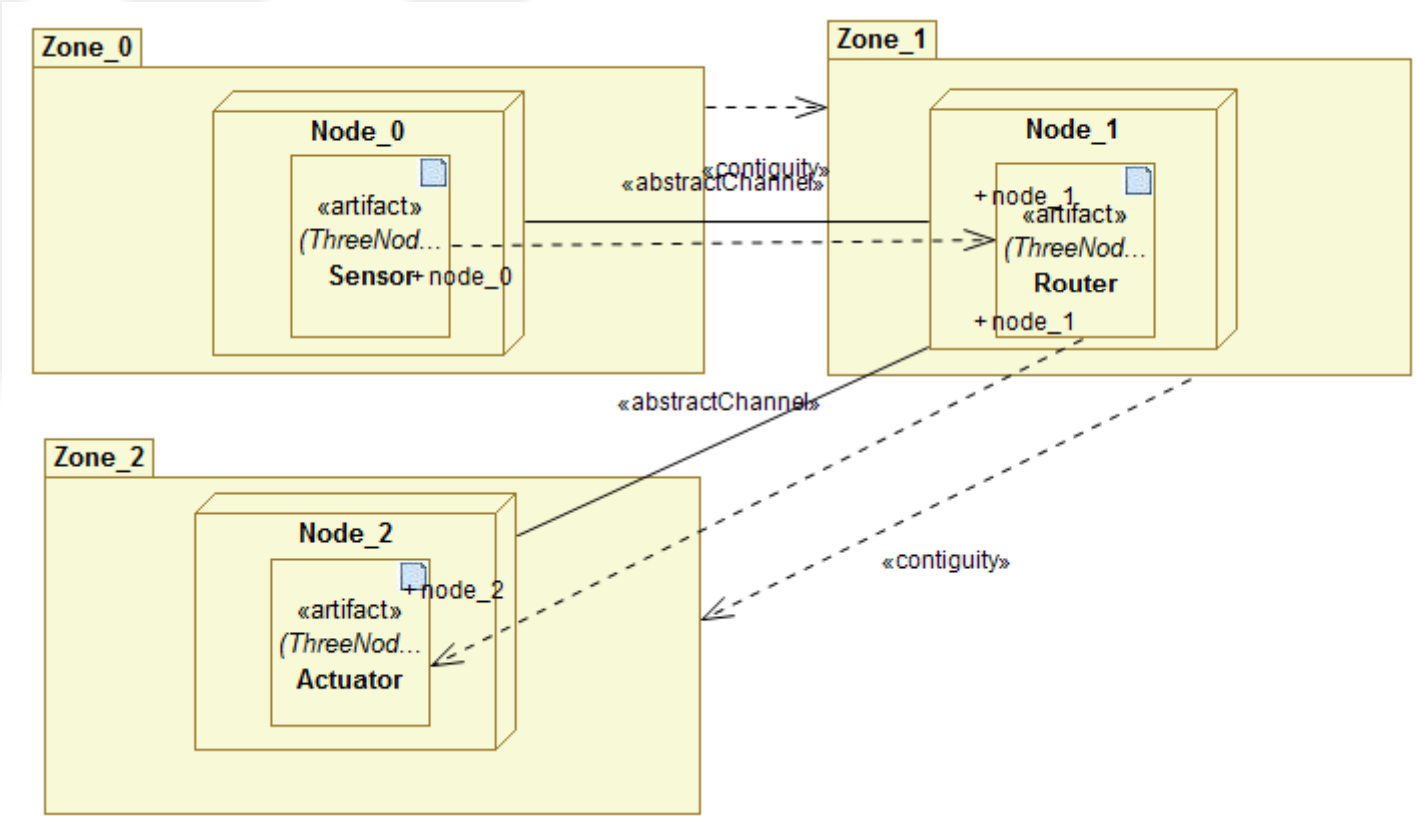
HIF2UML

- **HIF2UML:** it is a tool which redraw the HIF description back into UML level
- The main reason of it is
  - to back annotate UML diagrams
  - to generate UML diagrams from new network alternatives generated through manipulation
- Currently, the tool redraw the HIF descriptions of the network in terms of task, node, channel, data flow and zones in UML deployment diagrams
- In the future, the tool will support profile annotation

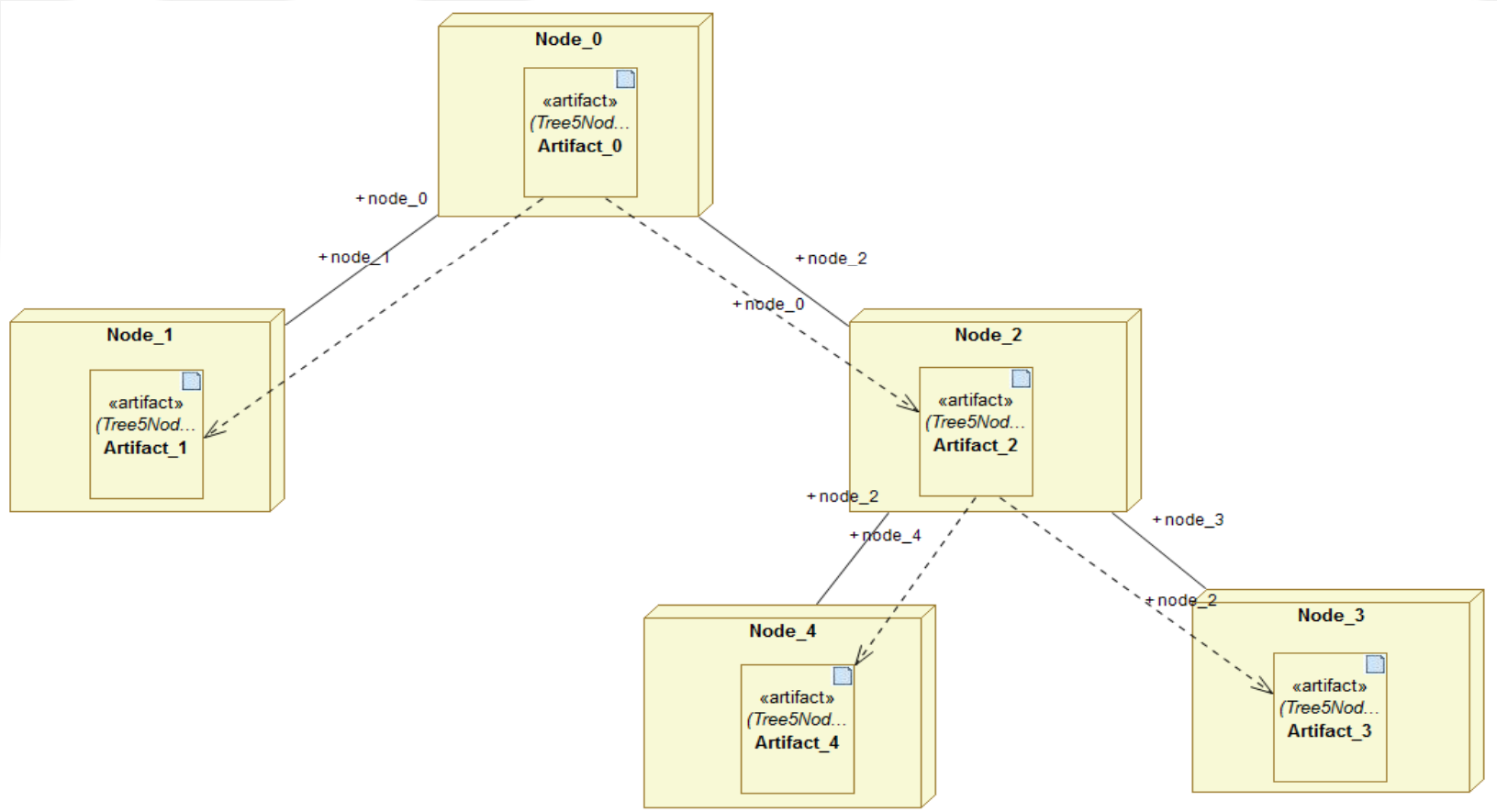
Case : Two Nodes



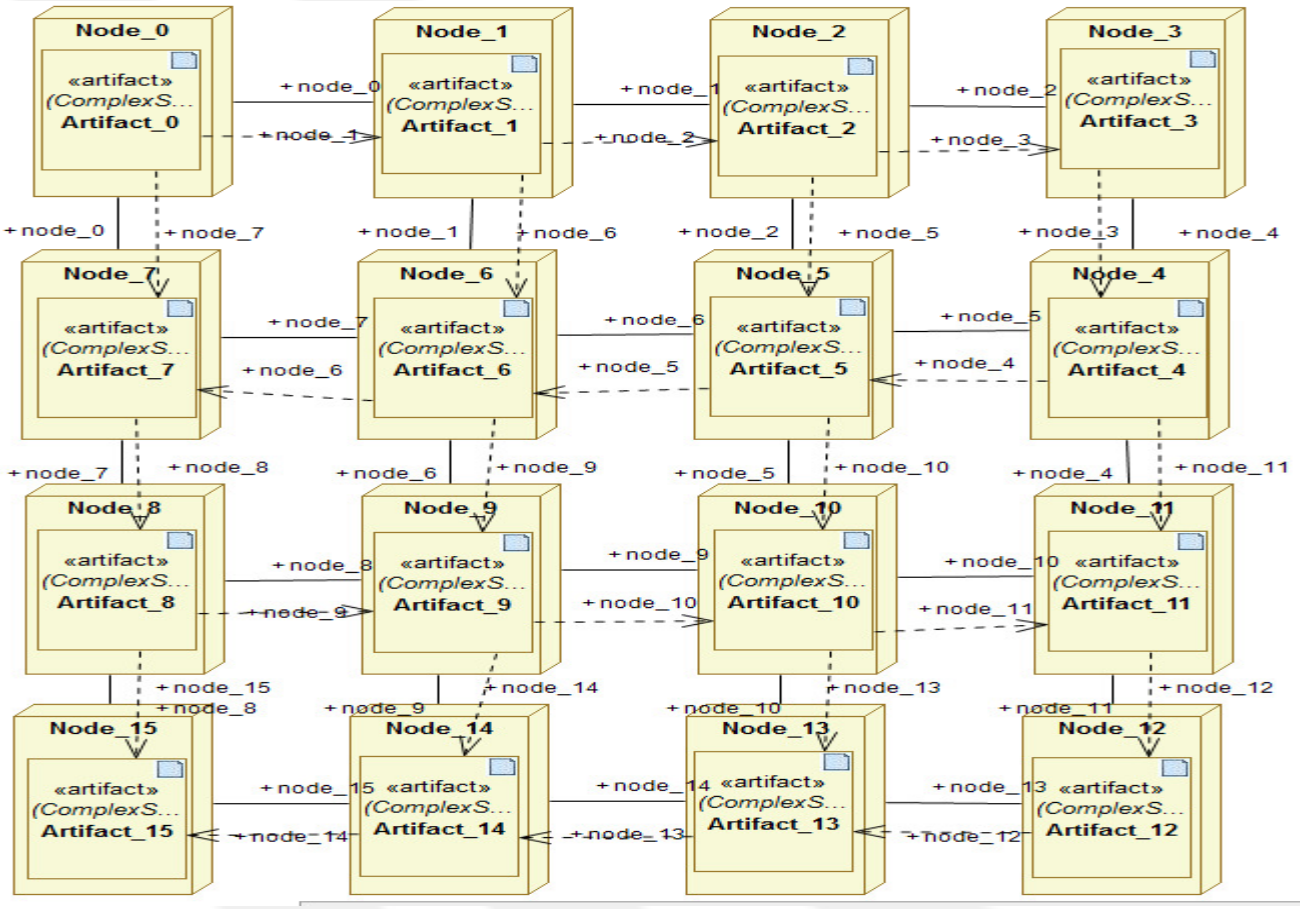
Case : Three Nodes With Zones



Case : Tree topology with 5 Nodes

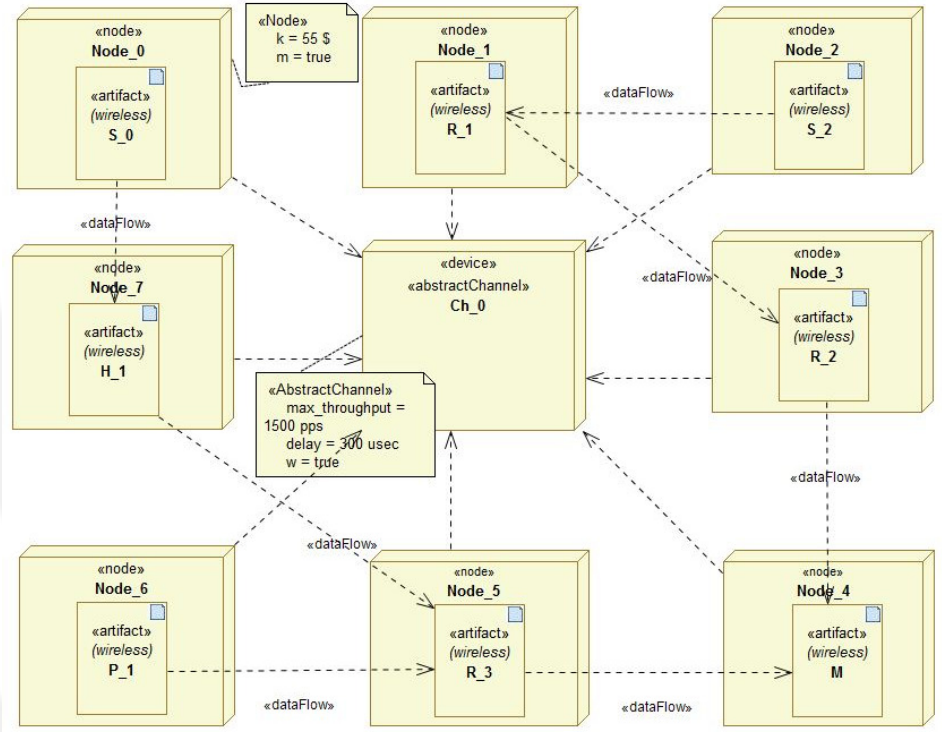


Case : Complex Scenario



Test case

- The building automation application



Test case

```
#include <tlm.h>
#include <exception>
#include <scnsl.hh>
#include "MyTask_t.hh"
using namespace Scnsl::Core;
int sc_main( int argc, char * argv[] )
{
// Node creation
    Node_t * Node_0 = scnsl->createNode();
    Node_t * Node_1 = scnsl->createNode();
    ...
// Task creation
    MyTask_t Controller( "Controller", Node_0 );
    MyTask_t Sensor_1( "Sensor_1", Node_1 );
    ...
// Channel creation
    CoreChannelSetup_t ccs;
    ccs.extensionId = "core";
    ccs.channel_type = CoreChannelSetup_t::UNIDIRECTIONAL;
    ccs.name = "Unidirectional";
    ccs.delay = sc_core::sc_time(400, sc_core::SC_MS );
    ccs.nodes_number = 2;
    Channel_if_t * ch0 = scnsl->createChannel(ccs);
    ...
// Bind setup
    scnsl->bind( Node_0, ch0, bsb0);
    scnsl->bind( Controller, Sensor_1, ch0, bsb0, Mac_0 );
    ...
```