Design flow for Networked Embedded Systems

Emad Ebeid
Post-doc @ ENG Department
Aarhus University
Denmark

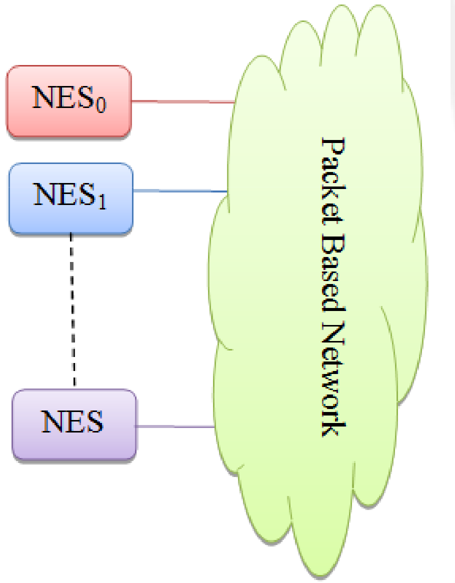
Davide Quaglia
Assistant Professor @ CS depart
University of Verona, Italy
Davide.Quaglia@univr.it

Outline

- Introduction and motivation
- Background
- Proposed methodology
- Modeling requirements
- System view simulation
- Network synthesis
- Network view simulation
- Case study
- Conclusion

Introduction

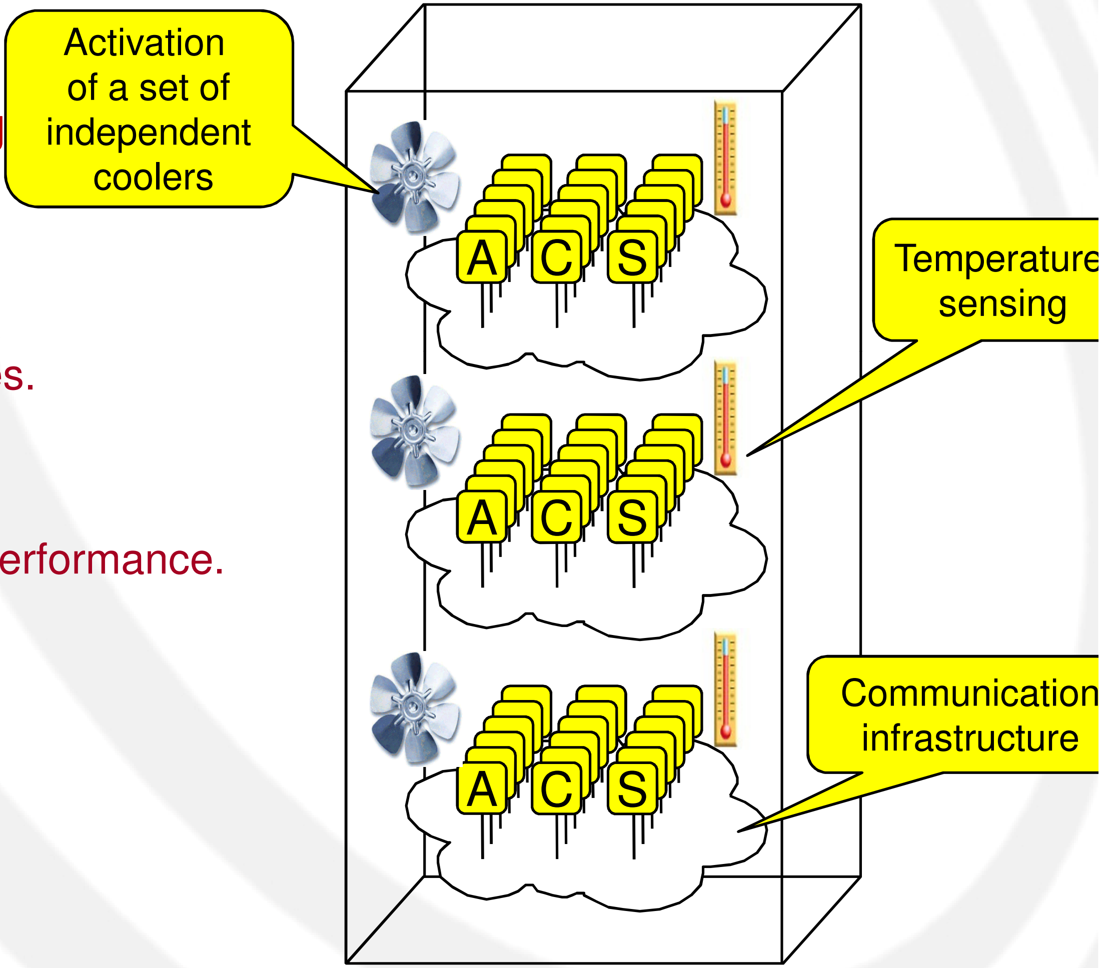
- **Networked Embedded Systems (NES)** are an important class of devices
  - Network functionalities are at the core of design objectives
  - Network requirements come together with traditional requirements
- **Distributed Embedded Systems** are group of NES which are connected together using network interfaces, standardized protocols and channels
  - Example: Temperature control of a building



Introduction

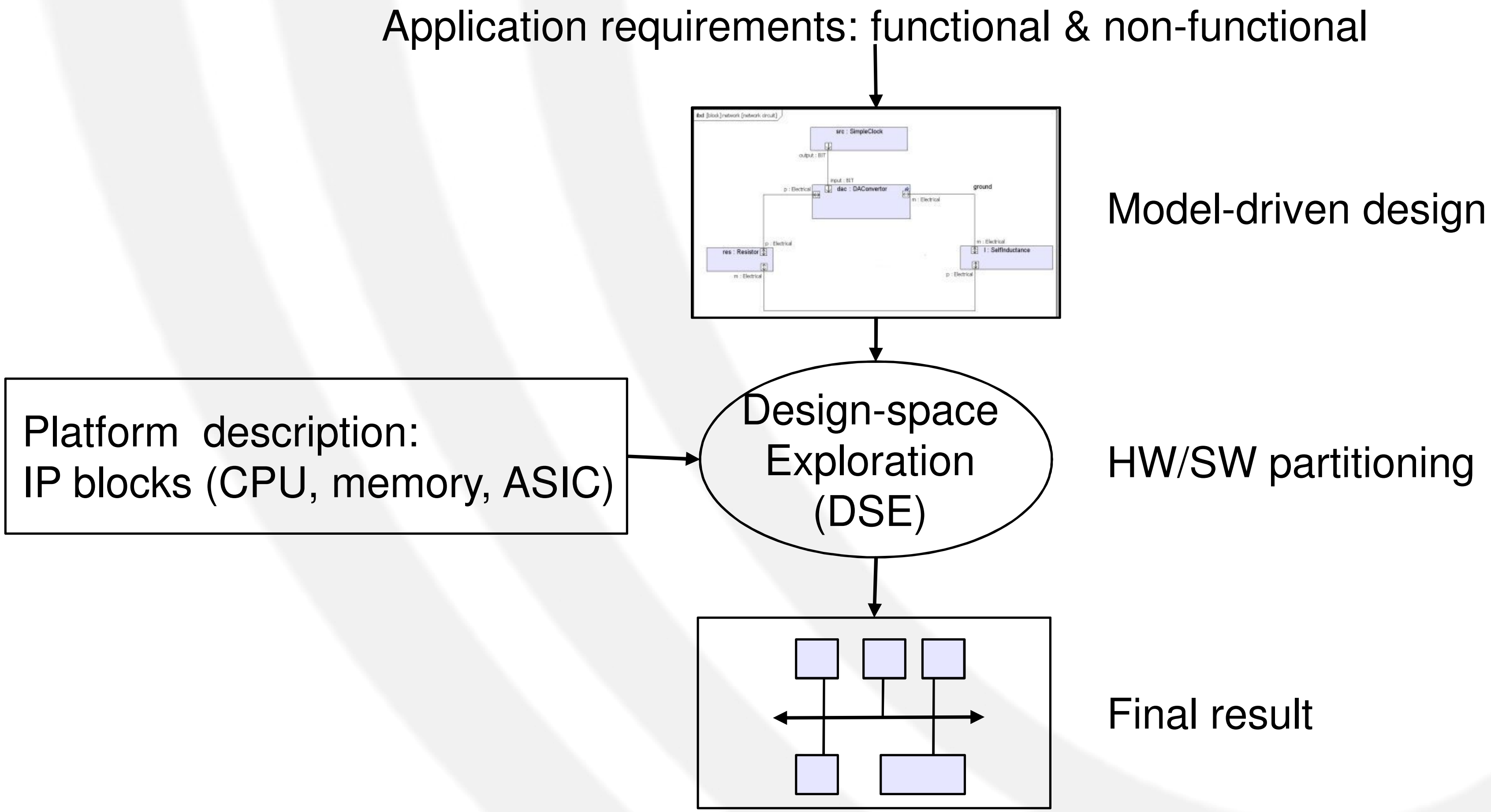
Temperature control of a building

- Scenario:
  - Hundreds of concurrent tasks.
  - Heterogeneous tasks.
  - Devices with different capabilities.
  - Wireless and wired channels.
  - Many communication protocols.
  - Nodes position affects system performance.
- Questions:
  - How many nodes?
  - How to assign tasks to nodes?
  - Which network protocols?
  - Which intermediate systems?


Activation of a set of independent coolers
Temperature sensing
Communication infrastructure
A
C
S
A
C
S
A
C
S

Introduction

Traditional design flow for embedded systems:



Application requirements: functional & non-functional

Model-driven design

Platform description:
IP blocks (CPU, memory, ASIC)

Design-space Exploration (DSE)

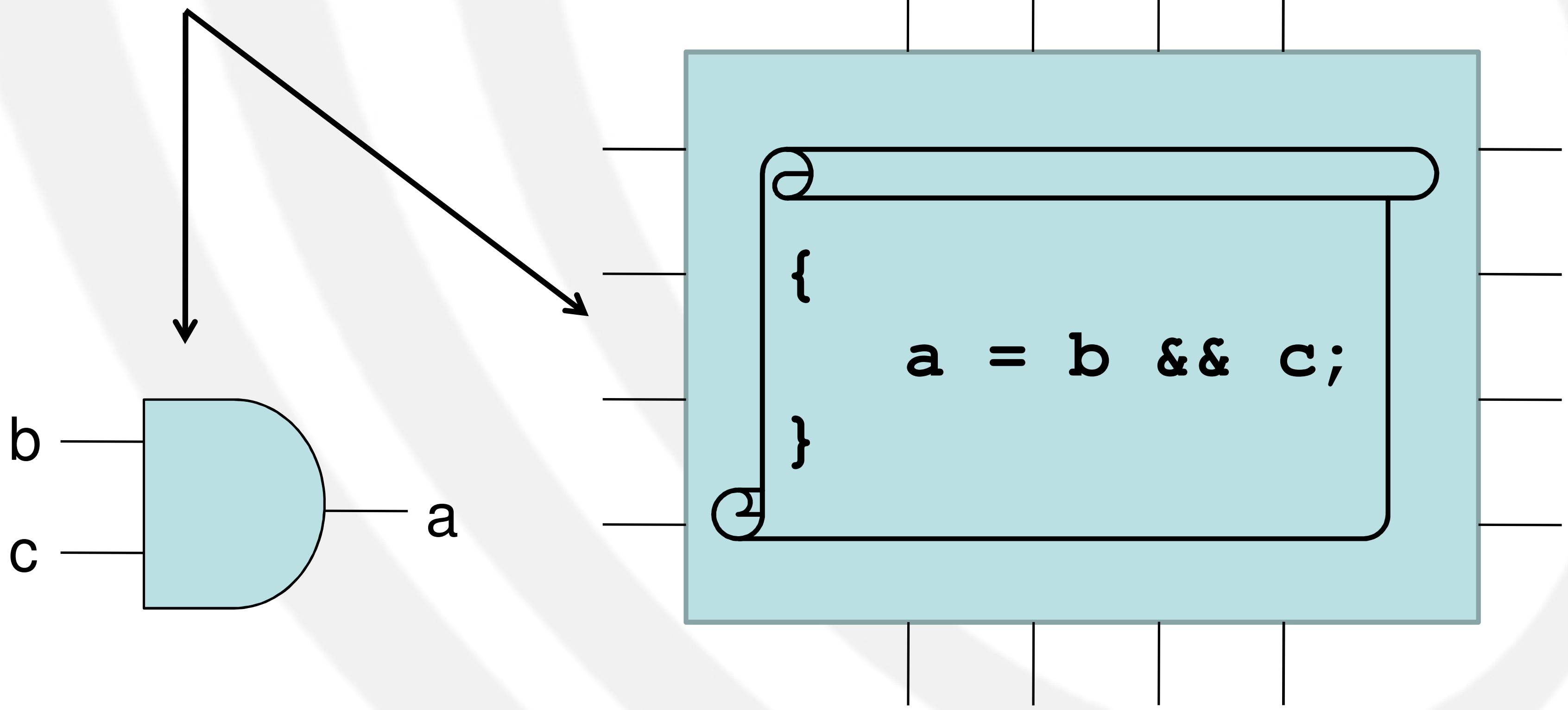
HW/SW partitioning

Final result

Introduction

HW/SW partitioning:

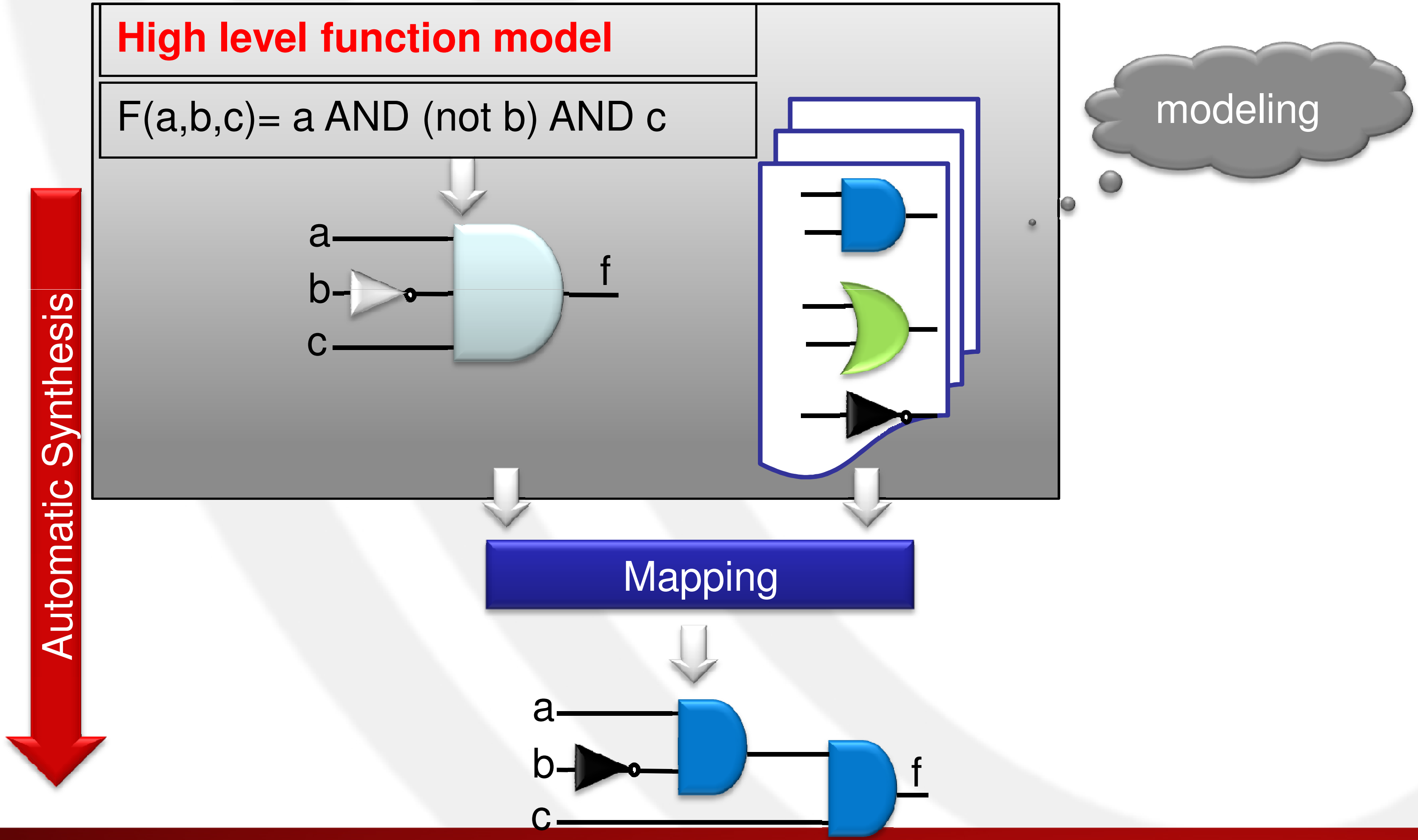
a = b AND c



```
{
    a = b && c;
}
```

Introduction

Hardware design:



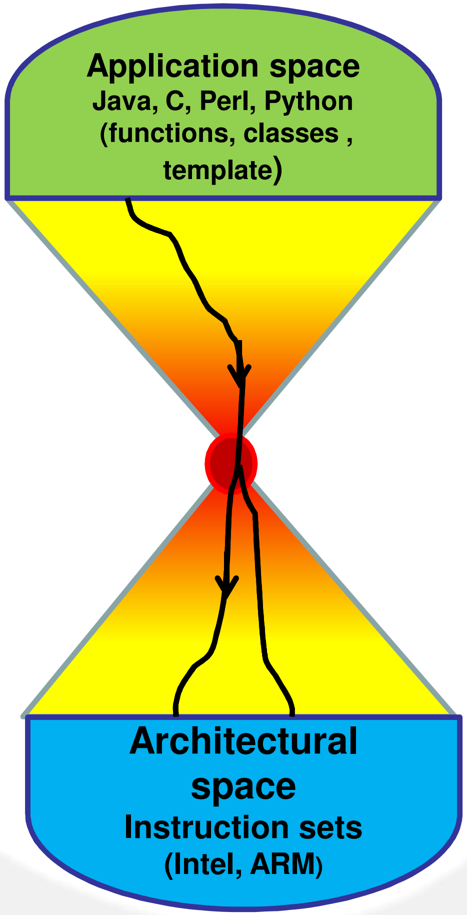
High level function model

F(a,b,c)= a AND (not b) AND c

Introduction

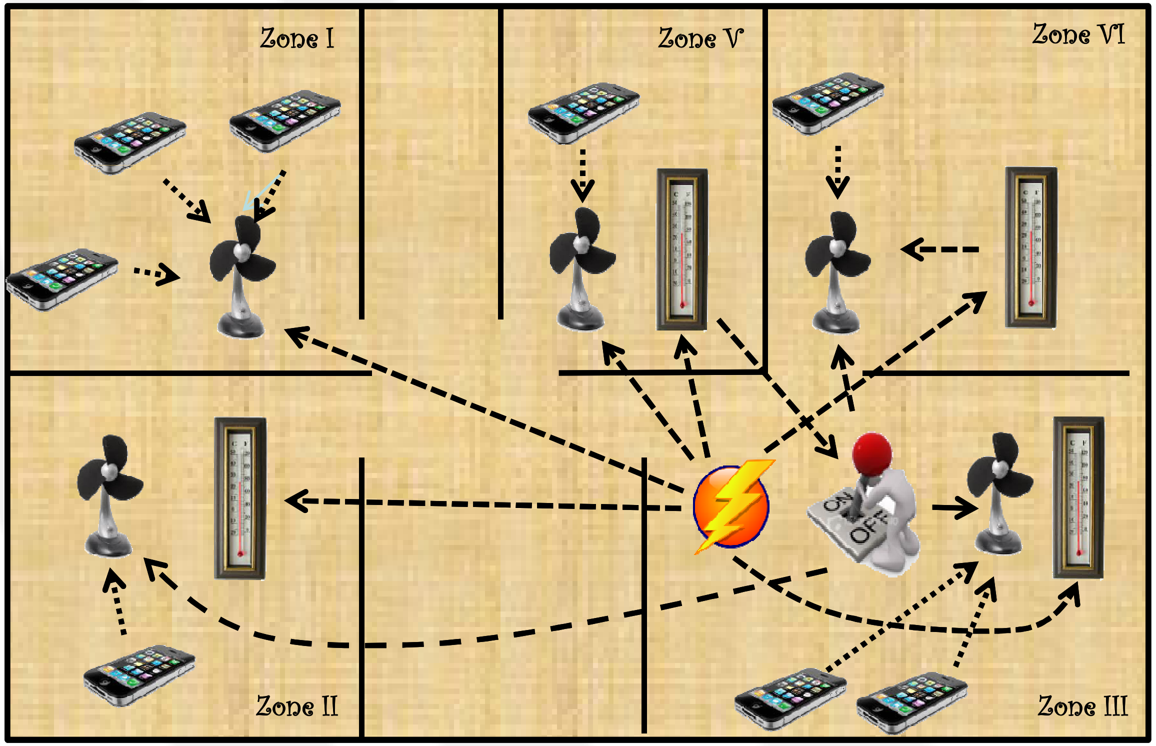
Software development

- Functionality is described with different languages and an automatic process is used to generate assembly code for different target CPU's
- **Modeling** of the functionality: High level languages
- **Automatic synthesis**: Compilers



Introduction

- **Distributed embedded system** as a single system to be designed

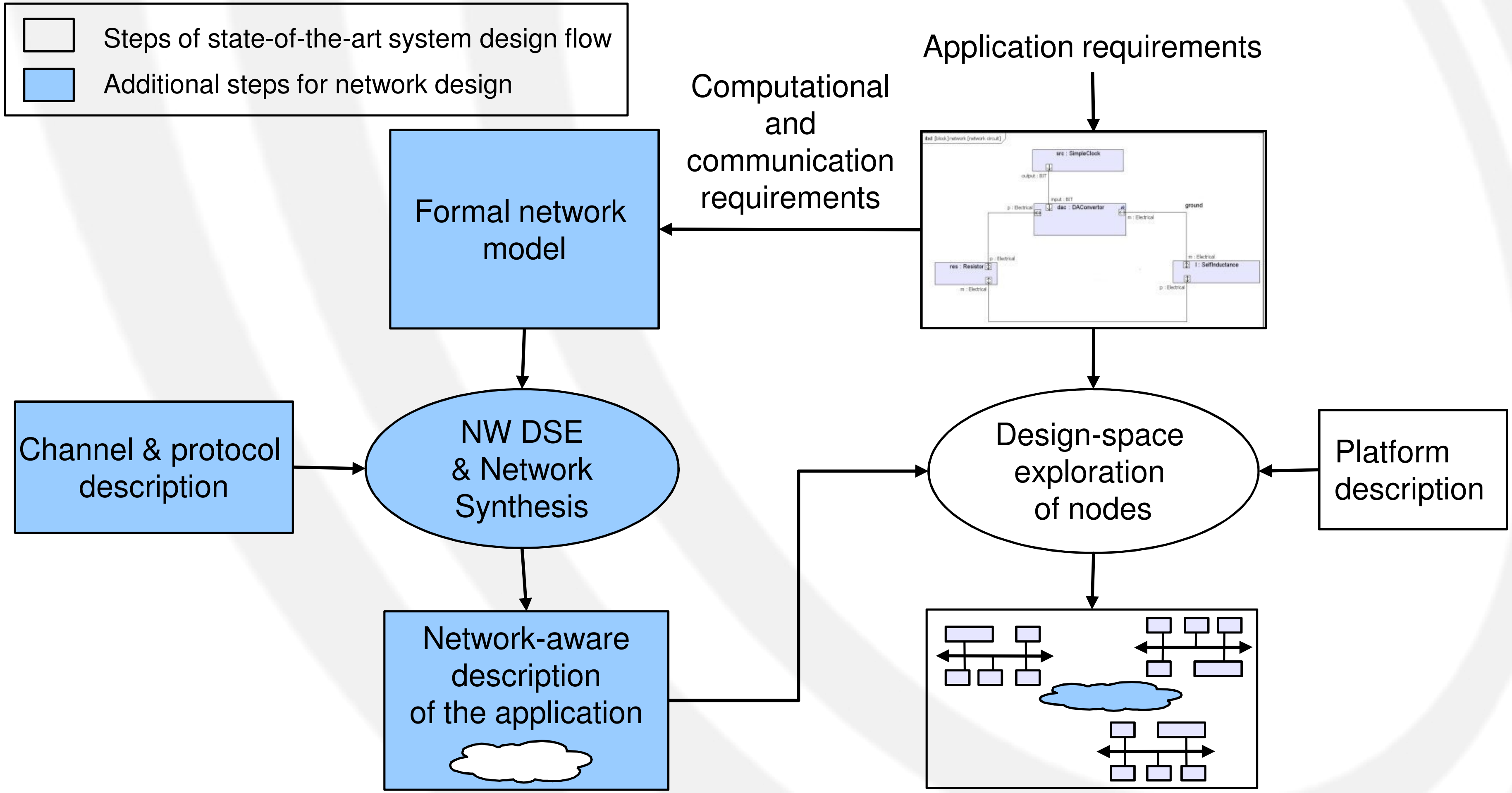


Introduction

New design flow for NES

□ Steps of state-of-the-art system design flow

🟦 Additional steps for network design



Introduction

- Start from an abstract Model-Driven System Specification
- Modeling and Analysis of Real-Time and Embedded Systems (MARTE) profile for the unified modeling language (UML)
- Refinement steps and simulations
- Standard representation of requirement and solutions

UML
MARTE

Background

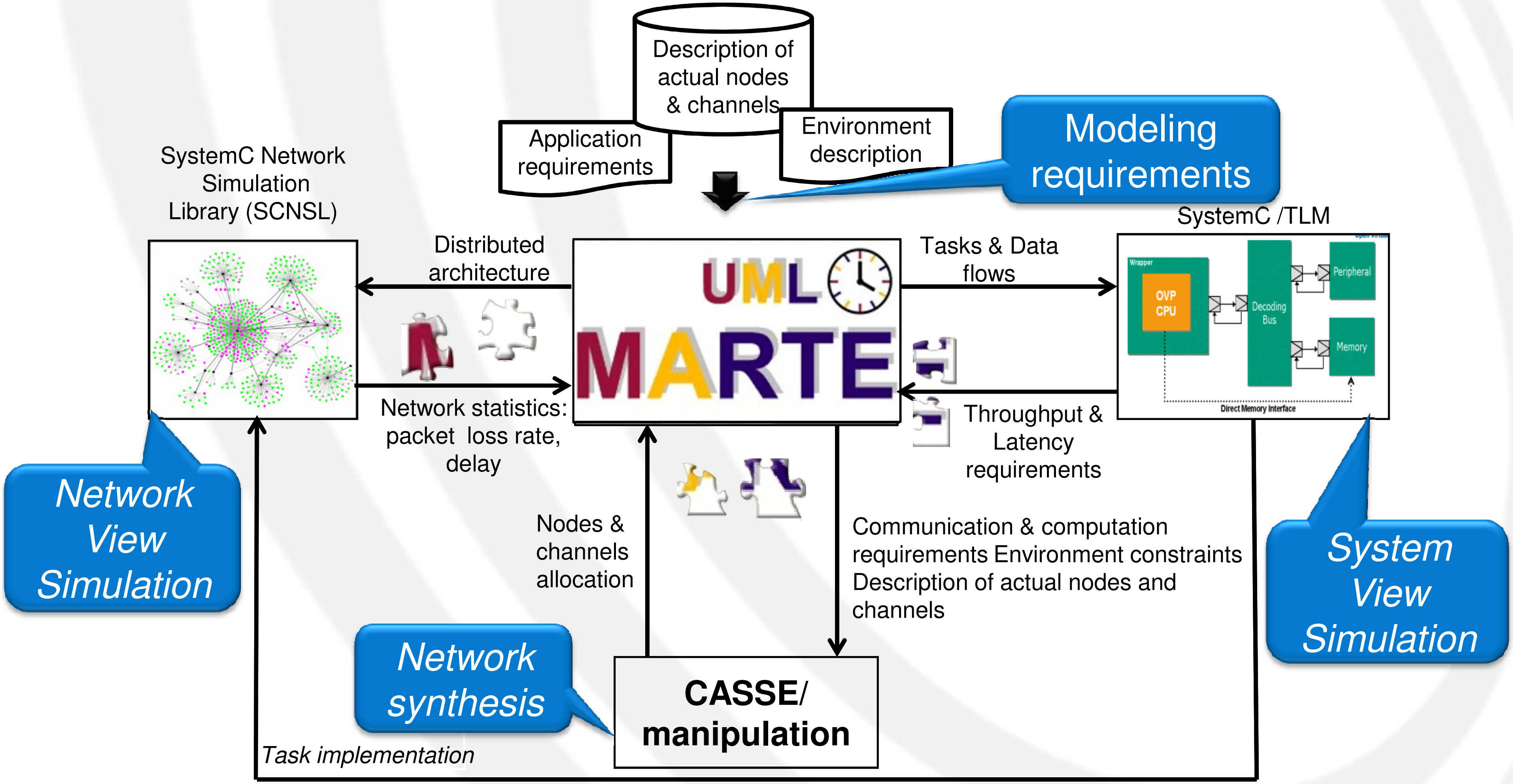
- Design of the network infrastructure starting from a library of nodes and channels (Network synthesis)
  - Communication Aware Specification and Synthesis Environment (CASSE), [FDL 2010]
  - COmmunication Synthesis Infrastructure framework (COSI), [IEEE TASE '12]
- **Open issue :** Both approaches do not rely on a standard representation of requirements (from the initial user specification) and solutions

Key idea

Design methodology for networked embedded systems which combines UML/MARTE, network synthesis, and simulation

UML/MARTE not only at the starting point but also at the center of design flow as repository of refined version of the system up to the final solution

Proposed methodology



Description of actual nodes & channels

Application requirements

Environment description

Modeling requirements

SystemC Network Simulation Library (SCNSL)

SystemC /TLM

Distributed architecture

Tasks & Data flows

Network statistics: packet loss rate, delay

Throughput & Latency requirements

Nodes & channels allocation

Communication & computation requirements Environment constraints Description of actual nodes and channels

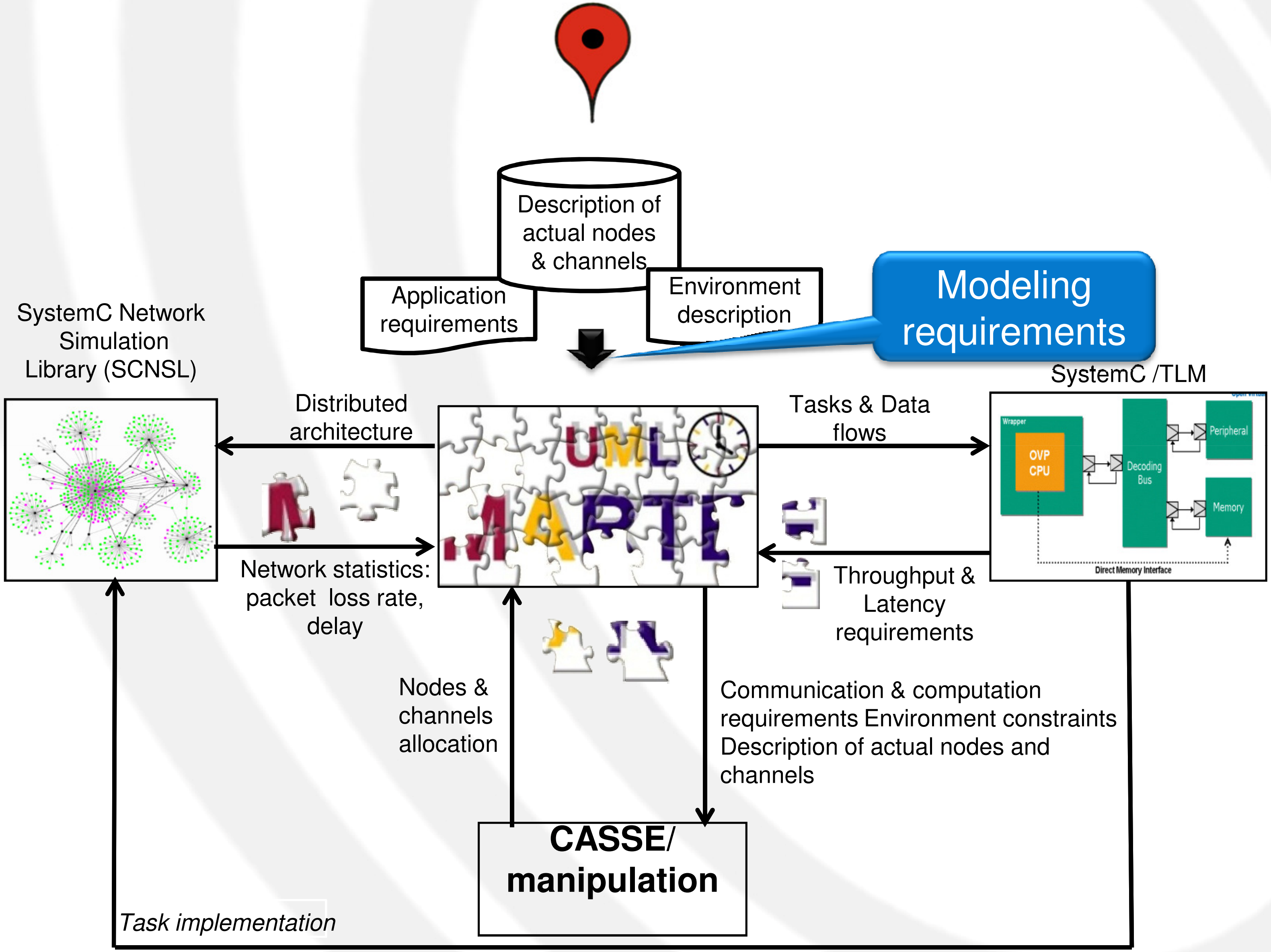
CASSE/ manipulation

Network View Simulation

System View Simulation

Network synthesis

Task implementation



Description of actual nodes & channels

Application requirements

Environment description

Modeling requirements

SystemC Network Simulation Library (SCNSL)

Distributed architecture

Network statistics: packet loss rate, delay

Tasks & Data flows

SystemC /TLM

Throughput & Latency requirements

Nodes & channels allocation

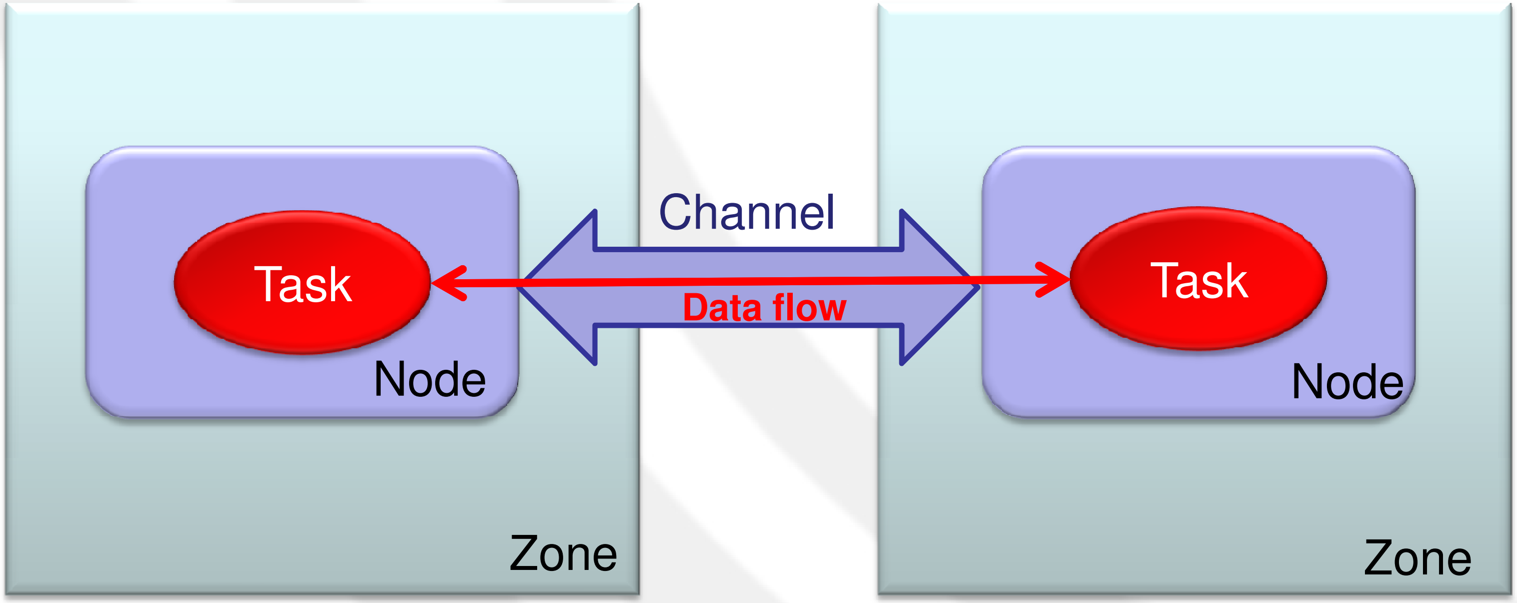
Communication & computation requirements Environment constraints Description of actual nodes and channels

CASSE/ manipulation

Task implementation

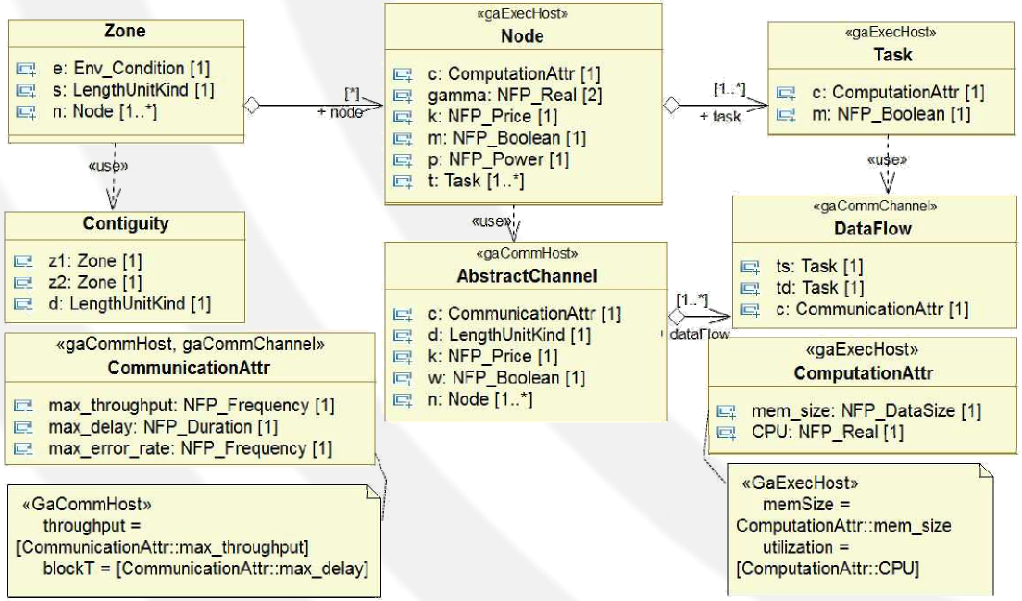
Modeling requirements

- The main aspects to be represented in UML/MARTE are:
  - Tasks, data flows, nodes, channels and the external environment



Modeling requirements

- Generic Quantitative Analysis Modeling (GQAM) sub-profile of MARTE profile are used to specify the semantics of some classes and their attributes
- This is the first time that GQAM is used to model the network



Zone
e: Env_Condition [1]
s: LengthUnitKind [1]
n: Node [1..*]

[*]
+ node

«use»

Contiguity
z1: Zone [1]
z2: Zone [1]
d: LengthUnitKind [1]

«gaCommHost, gaCommChannel»
CommunicationAttr
max_throughput: NFP_Frequency [1]
max_delay: NFP_Duration [1]
max_error_rate: NFP_Frequency [1]

«GaCommHost»
throughput =
[CommunicationAttr::max_throughput]
blockT = [CommunicationAttr::max_delay]

«gaExecHost»
Node
c: ComputationAttr [1]
gamma: NFP_Real [2]
k: NFP_Price [1]
m: NFP_Boolean [1]
p: NFP_Power [1]
t: Task [1..*]

[1..*]
+ task

«use»

«gaCommHost»
AbstractChannel
c: CommunicationAttr [1]
d: LengthUnitKind [1]
k: NFP_Price [1]
w: NFP_Boolean [1]
n: Node [1..*]

[1..*]
dataFlow

«gaExecHost»
Task
c: ComputationAttr [1]
m: NFP_Boolean [1]

«use»

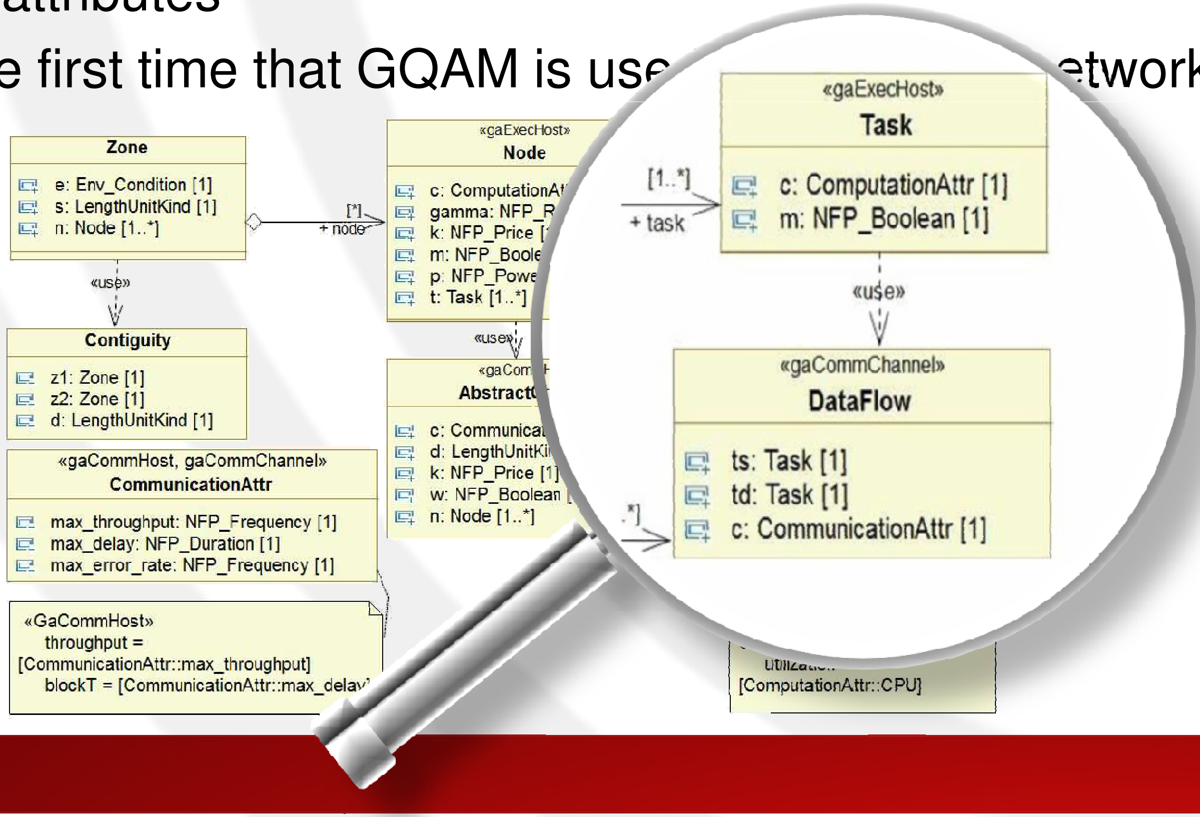
«gaCommChannel»
DataFlow
ts: Task [1]
td: Task [1]
c: CommunicationAttr [1]

«gaExecHost»
ComputationAttr
mem_size: NFP_DataSize [1]
CPU: NFP_Real [1]

«GaExecHost»
memSize =
ComputationAttr::mem_size
utilization =
[ComputationAttr::CPU]

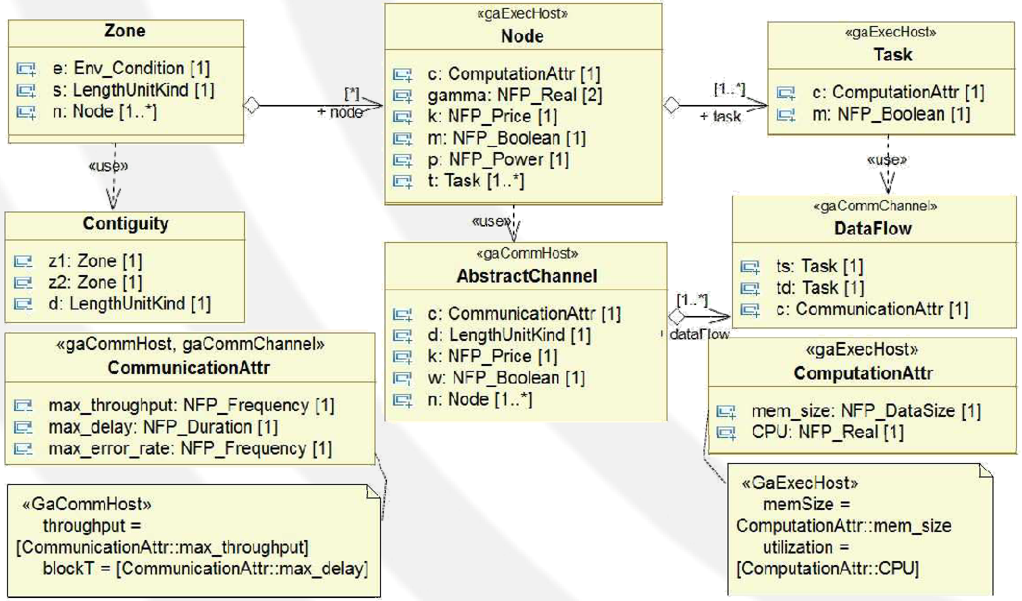
Modeling requirements

- Generic Quantitative Analysis Modeling (GQAM) sub-profile of MARTE profile are used to specify the semantics of some classes and their attributes
- This is the first time that GQAM is use etwork



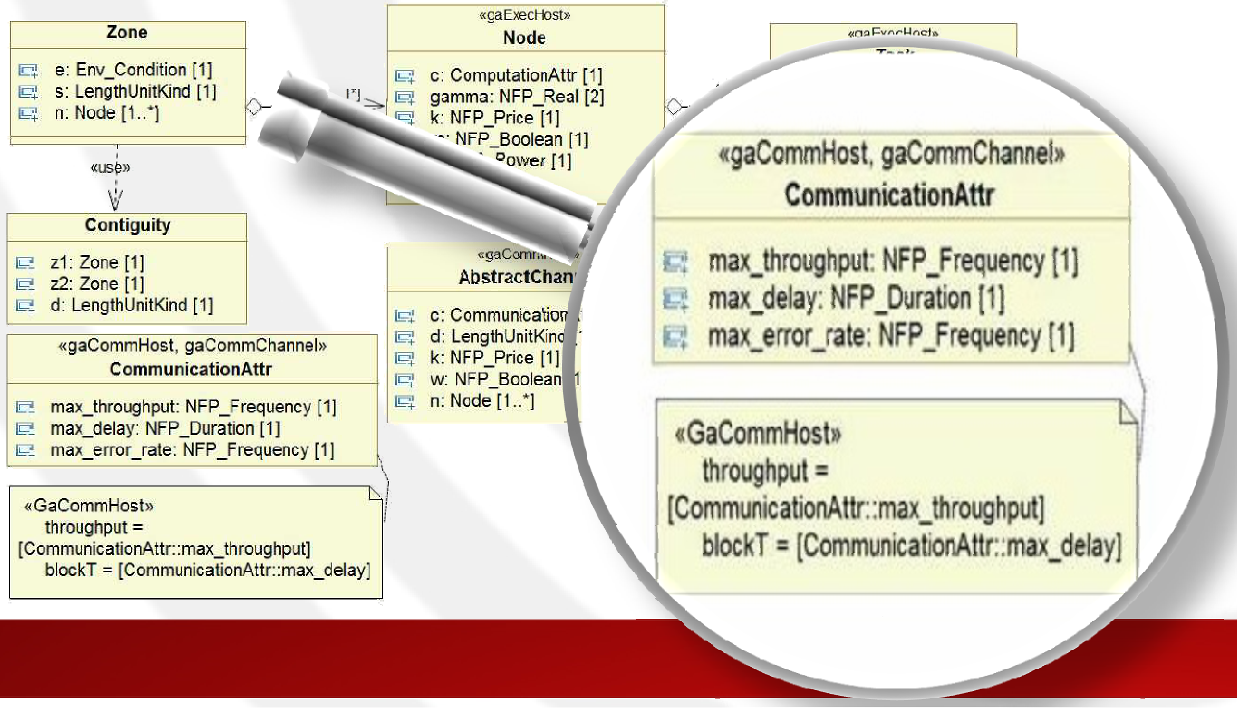
Modeling requirements

- Generic Quantitative Analysis Modeling (GQAM) sub-profile of MARTE profile are used to specify the semantics of some classes and their attributes
- This is the first time that GQAM is used to model the network



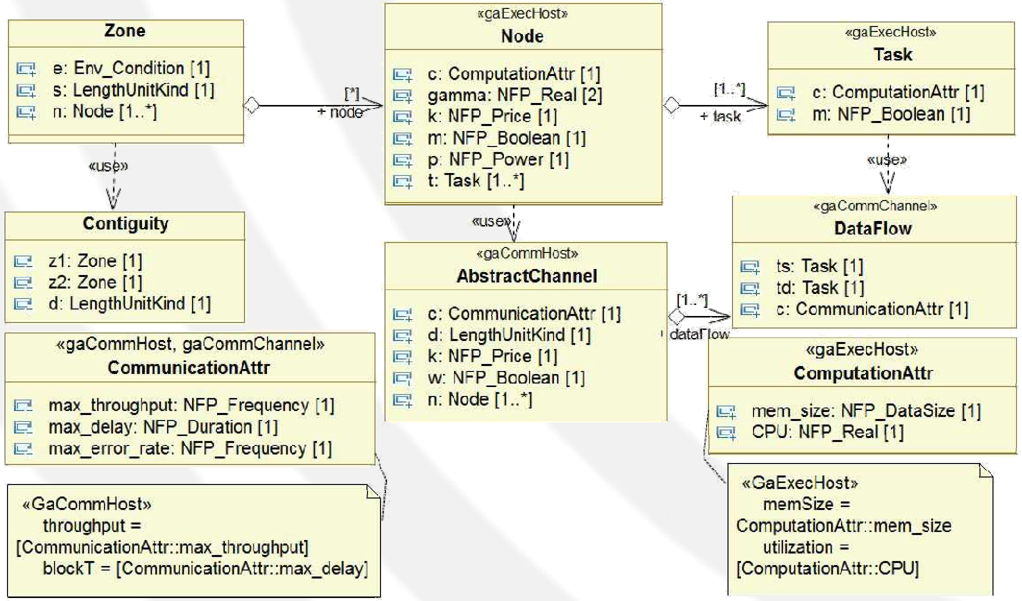
Modeling requirements

- Generic Quantitative Analysis Modeling (GQAM) sub-profile of MARTE profile are used to specify the semantics of some classes and their attributes
- This is the first time that GQAM is used to model the network



Modeling requirements

- Generic Quantitative Analysis Modeling (GQAM) sub-profile of MARTE profile are used to specify the semantics of some classes and their attributes
- This is the first time that GQAM is used to model the network



Zone
e: Env_Condition [1]
s: LengthUnitKind [1]
n: Node [1..*]

[*]
+ node

«gaExecHost»
Node
c: ComputationAttr [1]
gamma: NFP_Real [2]
k: NFP_Price [1]
m: NFP_Boolean [1]
p: NFP_Power [1]
t: Task [1..*]

[1..*]
+ task

«gaExecHost»
Task
c: ComputationAttr [1]
m: NFP_Boolean [1]

«use»

Contiguity
z1: Zone [1]
z2: Zone [1]
d: LengthUnitKind [1]

«use»

«gaCommHost»
AbstractChannel
c: CommunicationAttr [1]
d: LengthUnitKind [1]
k: NFP_Price [1]
w: NFP_Boolean [1]
n: Node [1..*]

[1..*]
dataFlow

«use»

«gaCommChannel»
DataFlow
ts: Task [1]
td: Task [1]
c: CommunicationAttr [1]

«gaCommHost, gaCommChannel»
CommunicationAttr
max_throughput: NFP_Frequency [1]
max_delay: NFP_Duration [1]
max_error_rate: NFP_Frequency [1]

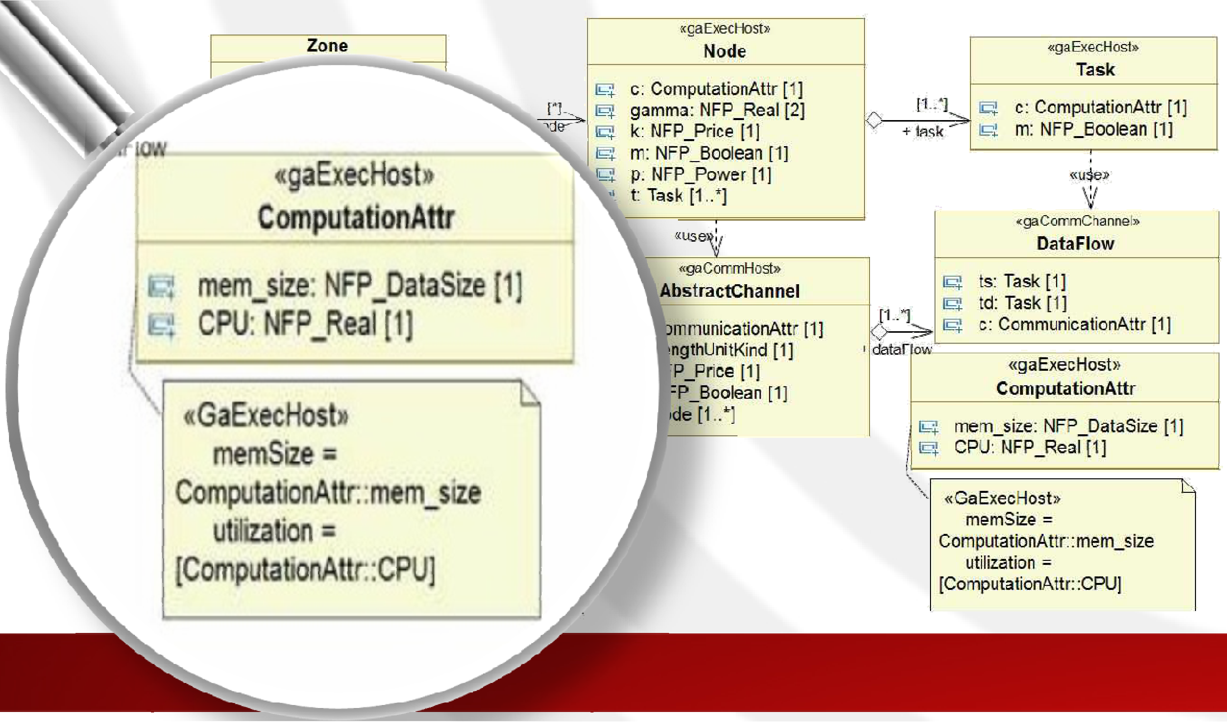
«gaExecHost»
ComputationAttr
mem_size: NFP_DataSize [1]
CPU: NFP_Real [1]

«GaCommHost»
throughput =
[CommunicationAttr::max_throughput]
blockT = [CommunicationAttr::max_delay]

«GaExecHost»
memSize =
ComputationAttr::mem_size
utilization =
[ComputationAttr::CPU]

Modeling requirements

- Generic Quantitative Analysis Modeling (GQAM) sub-profile of MARTE profile are used to specify the semantics of some classes and their attributes
- This is the first time that GQAM is used to model the network



Zone

«gaExecHost»
Node
c: ComputationAttr [1]
gamma: NFP_Real [2]
k: NFP_Price [1]
m: NFP_Boolean [1]
p: NFP_Power [1]
t: Task [1..*]

[*]
ode

[1..*]
+ task

«gaExecHost»
Task
c: ComputationAttr [1]
m: NFP_Boolean [1]

«use»

«gaCommChannel»
DataFlow
ts: Task [1]
td: Task [1]
c: CommunicationAttr [1]

«use»

«gaCommHost»
AbstractChannel
ommunicationAttr [1]
ngthUnitKind [1]
P_Price [1]
P_Boolean [1]
de [1..*]

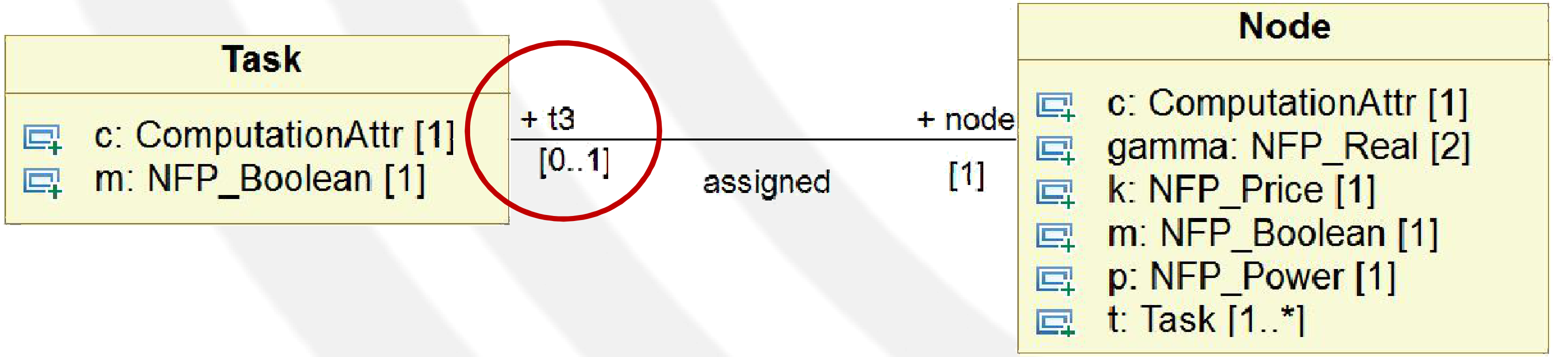
[1..*]
dataFlow

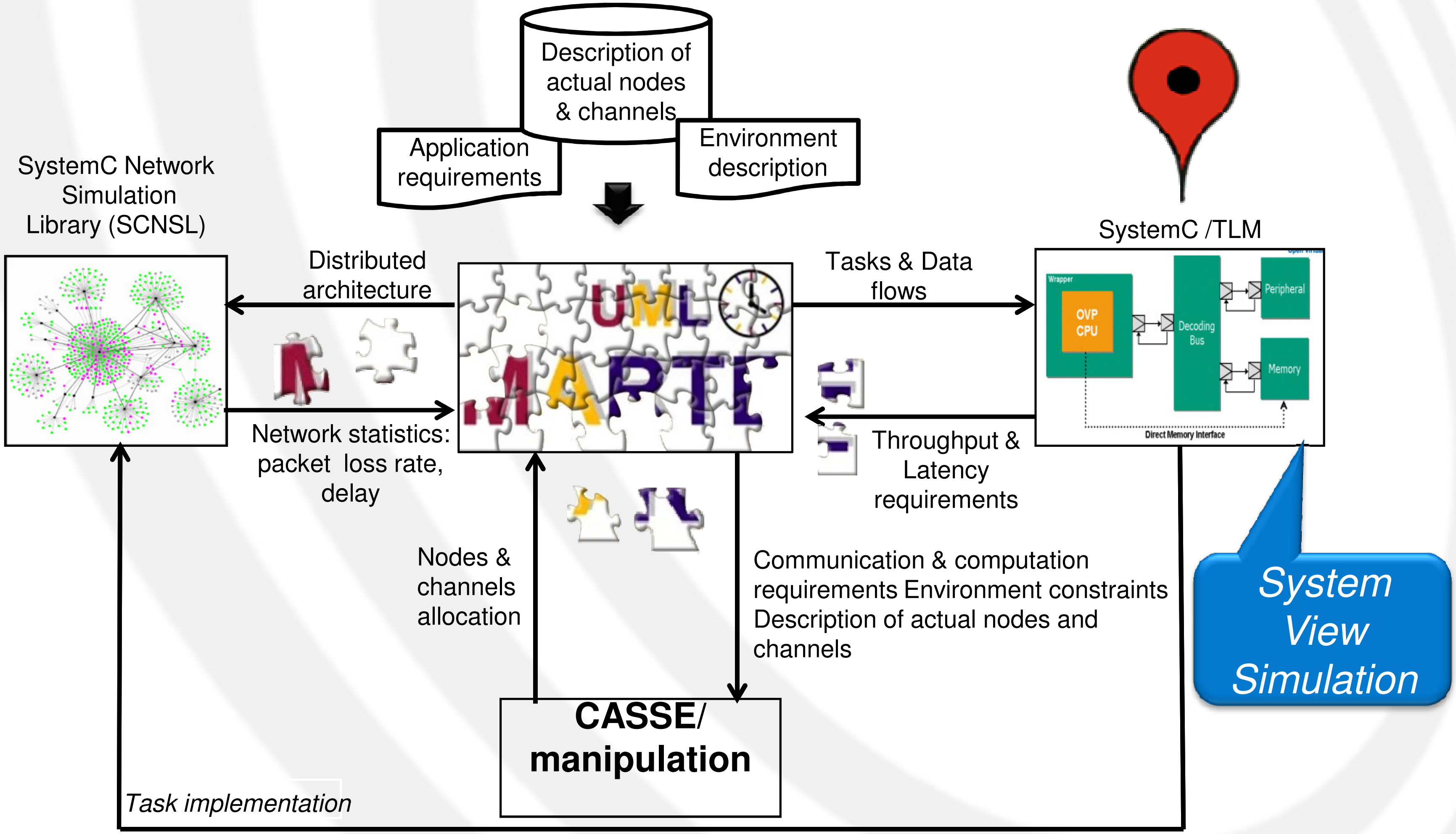
«gaExecHost»
ComputationAttr
mem_size: NFP_DataSize [1]
CPU: NFP_Real [1]

«GaExecHost»
memSize =
ComputationAttr::mem_size
utilization =
[ComputationAttr::CPU]

Modeling requirements

- Modeling of constraint:
  - Application constraints are specified by using cardinality on the relationships between classes
- Example of constraint: *“maximum one instance of t3 can be assigned to a single node”*





Description of actual nodes & channels

Application requirements

Environment description

SystemC Network Simulation Library (SCNSL)

Distributed architecture

Network statistics: packet loss rate, delay

SystemC /TLM

Tasks & Data flows

Throughput & Latency requirements

Nodes & channels allocation

Communication & computation requirements Environment constraints Description of actual nodes and channels

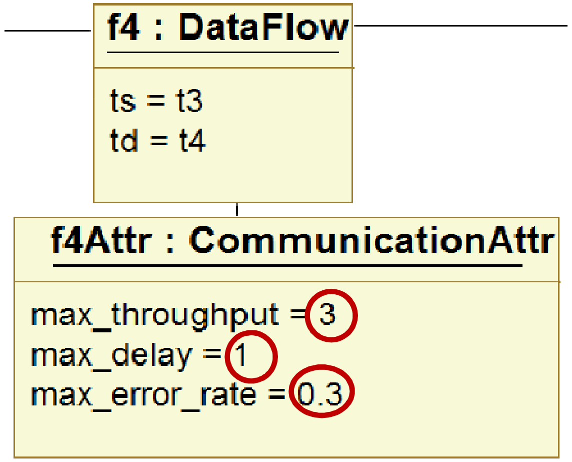
CASSE/ manipulation

Task implementation

System View Simulation

System view simulation

- UML/MARTE class diagram is extracted and used to generate SystemC/TLM model
  - Transformations are straight forward also (Villar,2009 and Vanderperren,2008)
- Execution of the SystemC model
  - Validate of functional behavior of the application
  - Fine-tune implementation details such as the content of exchanged messages and their sending rates
- Back annotation of throughput, latency and max error rate inside UML/MARTE model

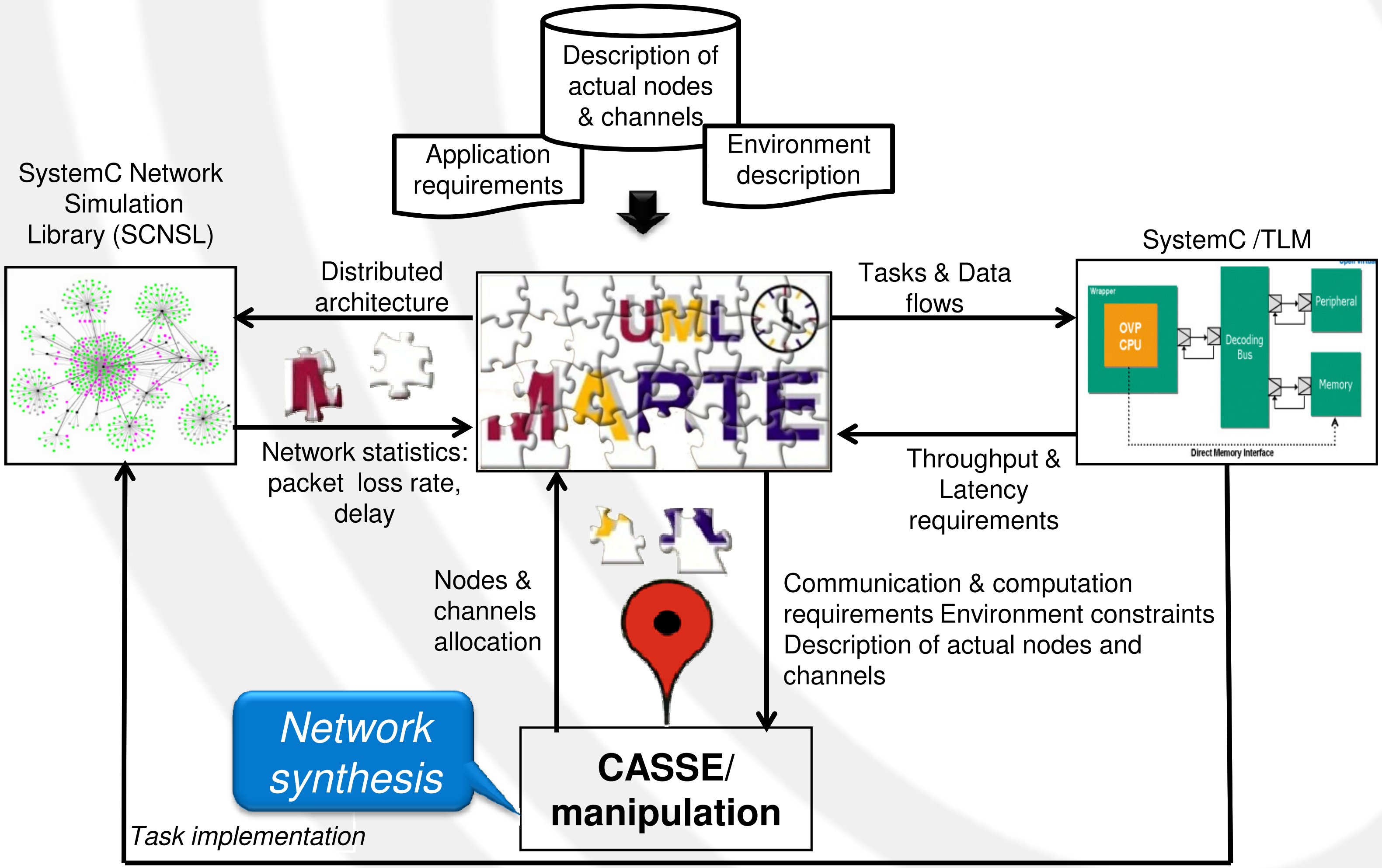


f4 : DataFlow

ts = t3
td = t4

f4Attr : CommunicationAttr

max_throughput = 3
max_delay = 1
max_error_rate = 0.3



Description of actual nodes & channels

Application requirements

Environment description

SystemC Network Simulation Library (SCNSL)

Distributed architecture

Network statistics: packet loss rate, delay

Tasks & Data flows

SystemC /TLM

Throughput & Latency requirements

Nodes & channels allocation

Communication & computation requirements Environment constraints Description of actual nodes and channels

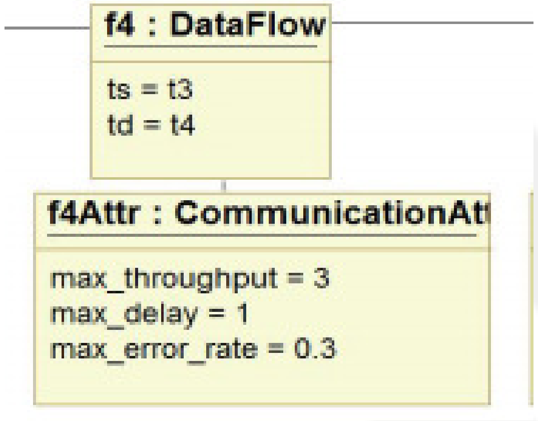
Network synthesis

CASSE/ manipulation

Task implementation

Network synthesis

- All the information about user constraints, communication requirements and actual channels and nodes are extracted from the UML/MARTE model and translated into Network synthesis mathematical representation
- CASSE provides a mathematical notation to specify the network dimension of a distributed embedded system, preparing the way for network synthesis



f4 : DataFlow

ts = t3
td = t4

f4Attr : CommunicationAtt

max_throughput = 3
max_delay = 1
max_error_rate = 0.3

Dataflow(f4) = [t3, t4, [3, 1, 0.3]].

Network synthesis cont’d

Set of tasks & data flows

t3 : Task

c = 1,1
m = true

f4 : DataFlow

ts = t3
td = t4
c = 3,1,0.3

The building geometry

z3 : Zone

Technological library
(network nodes and channels)

b : Node

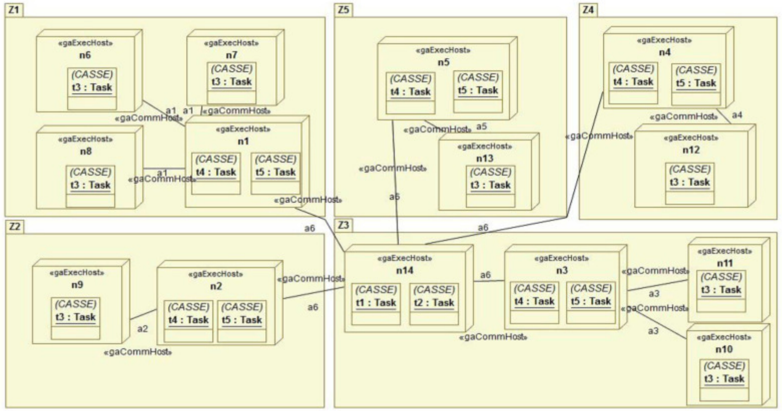
c = 10,10
gamma = [0.6, 0.9]
m = false
k = 100

y : AbstractChannel

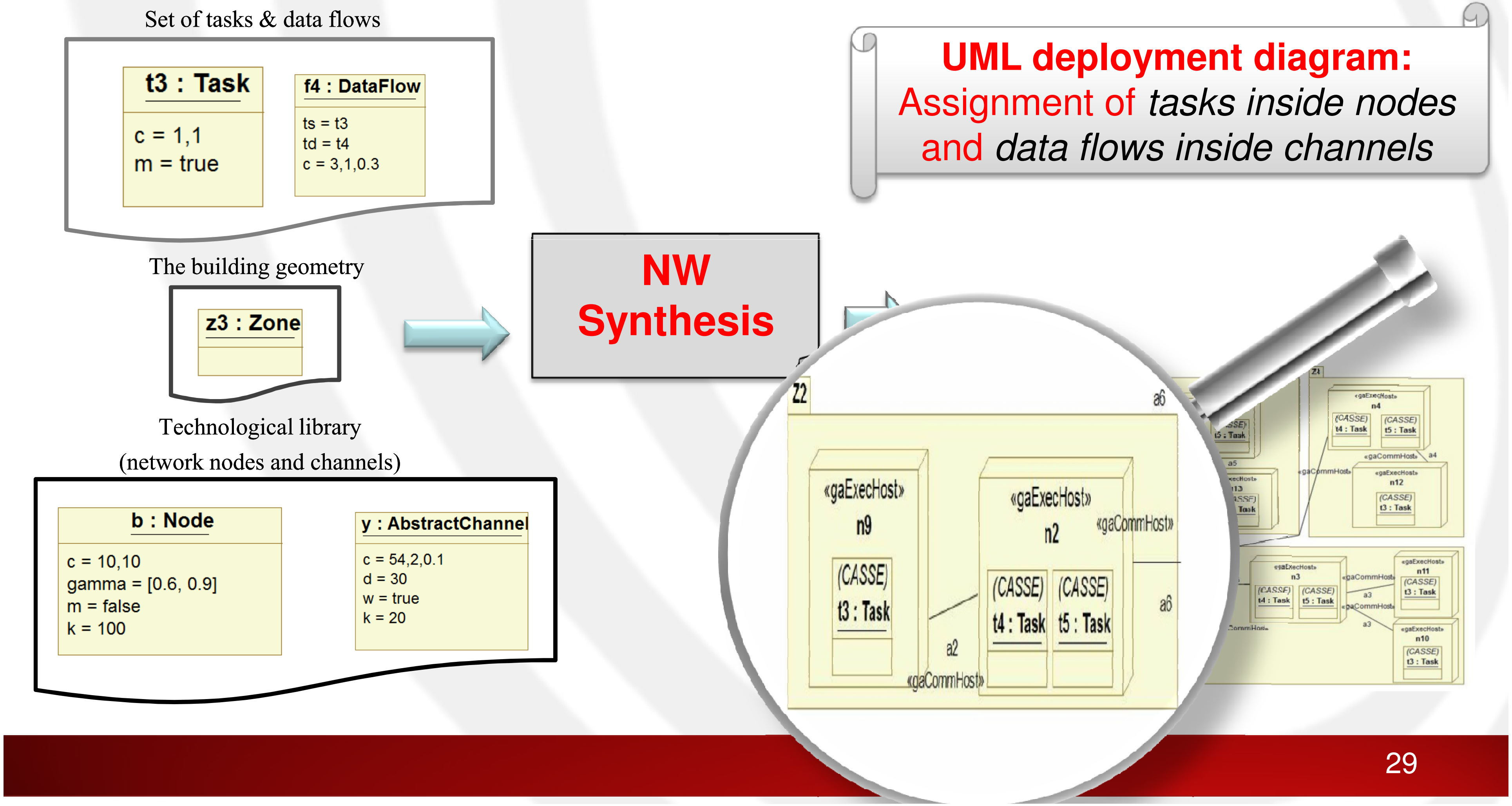
c = 54,2,0.1
d = 30
w = true
k = 20

NW Synthesis

UML deployment diagram:
Assignment of *tasks inside nodes* and *data flows inside channels*



Network synthesis cont'd



Set of tasks & data flows

t3 : Task
c = 1,1
m = true

f4 : DataFlow
ts = t3
td = t4
c = 3,1,0.3

The building geometry

z3 : Zone

Technological library
(network nodes and channels)

b : Node
c = 10,10
gamma = [0.6, 0.9]
m = false
k = 100

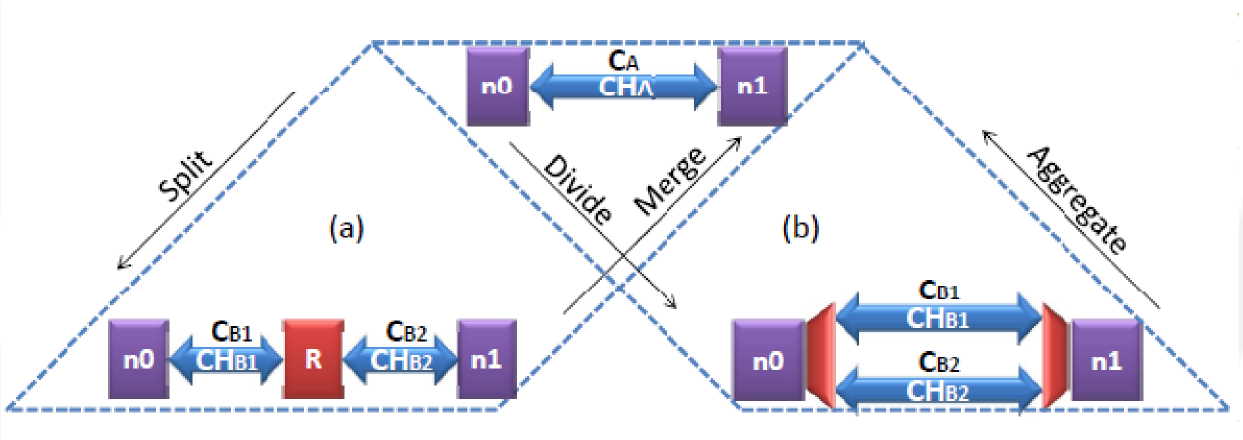
y : AbstractChannel
c = 54,2,0.1
d = 30
w = true
k = 20

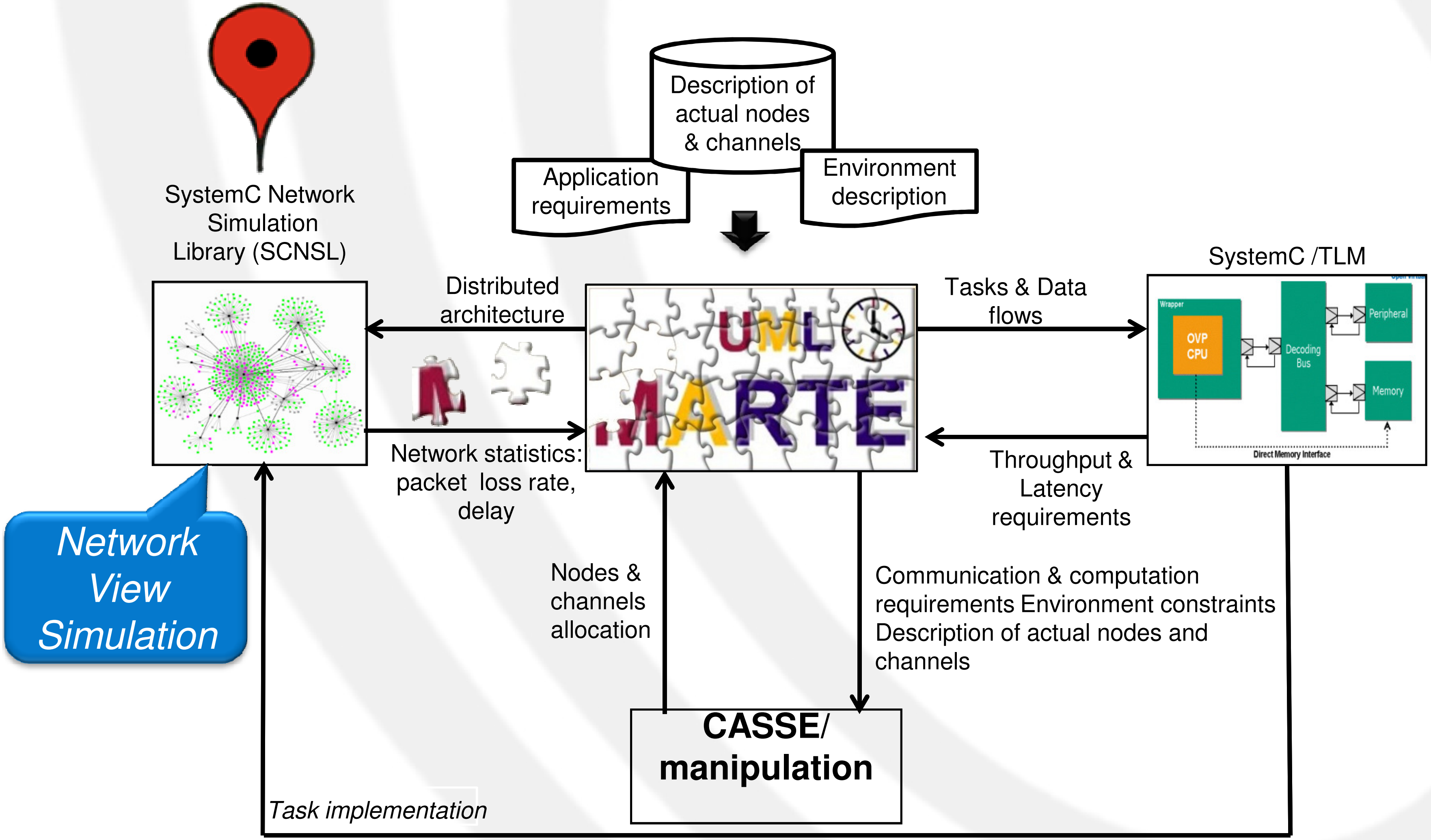
NW Synthesis

UML deployment diagram:
Assignment of *tasks inside nodes* and *data flows inside channels*

Manipulation

- This step aims at obtaining several NW alternatives which are equivalent from the network perspective
- Examples of manipulation rules
  - – Divide
  - – Split
  - – Merge
  - – Aggregate





SystemC Network Simulation Library (SCNSL)

Description of actual nodes & channels

Application requirements

Environment description

SystemC /TLM

Distributed architecture

Tasks & Data flows

Network statistics: packet loss rate, delay

Throughput & Latency requirements

Network View Simulation

Nodes & channels allocation

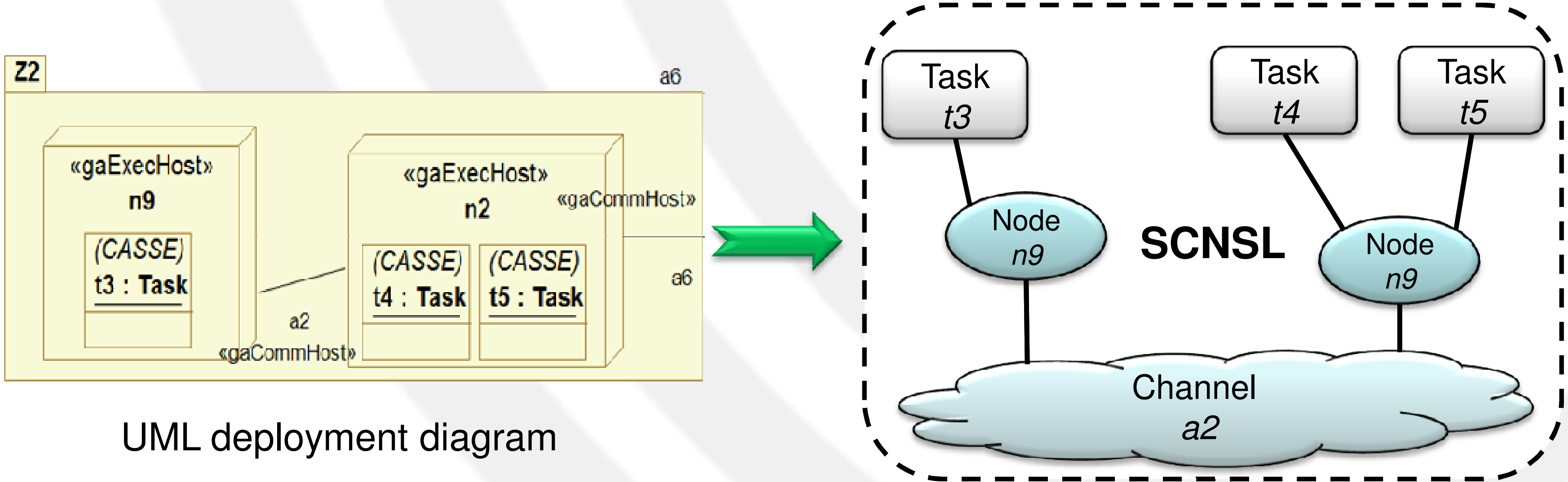
Communication & computation requirements Environment constraints Description of actual nodes and channels

CASSE/ manipulation

Task implementation

Network view simulation

- SCNSL is an extension of SystemC to allow modeling packet-based networks
  - It allows the easy and complete modeling of distributed applications of networked embedded systems such as wireless sensor networks, routers, and distributed plant controllers



UML deployment diagram

Network view simulation

- Correspondence between UML/MARTE and SCNSL elements

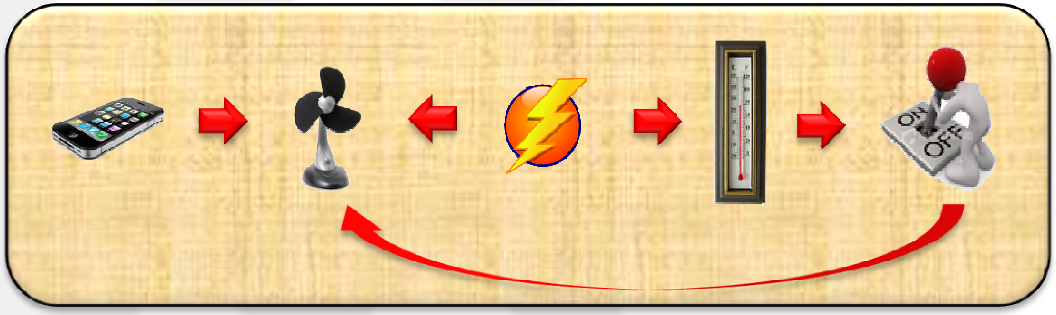
UML/MARTE	SCNSL
Node (n1)	n1 = scnsl->createNode();

Channel (ch)
bound to node (n1)
	CoreChannelSetup t ccs;
	ch = scnsl->createChannel(ccs);
	BindSetup base t bsb1;
	scnsl->bind(n1,ch,bsb1);

Data flow between task (t1)
and task (t2)
	CoreCommunicatorSetup t ccoms;
	mac1 = scnsl ->createCommunicator(ccoms);
	scnsl->bind(& t1,& t2,ch,bsb1,mac1);

Zones (z1…zn)	Environment_if_t object

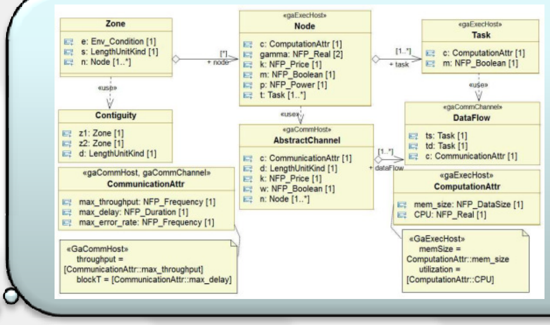
Case study

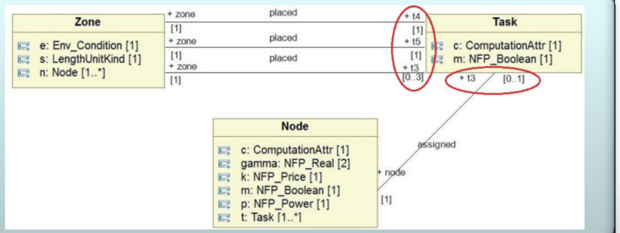


User requirement

+

- one instance of actuator should be placed in each zone
- max……



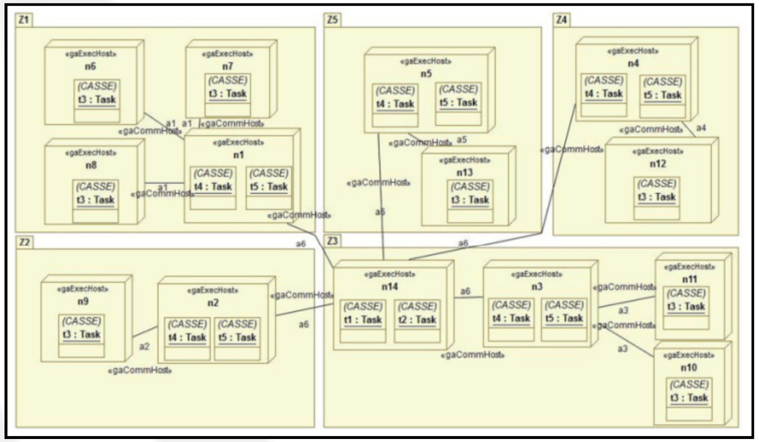


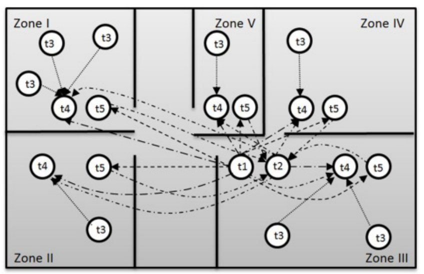
System View modeling

NW synthesis tool

Case study cont'd

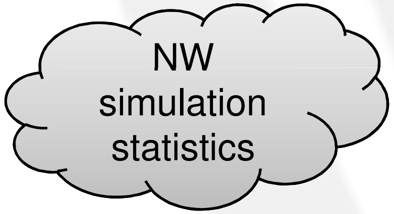
NW view modeling



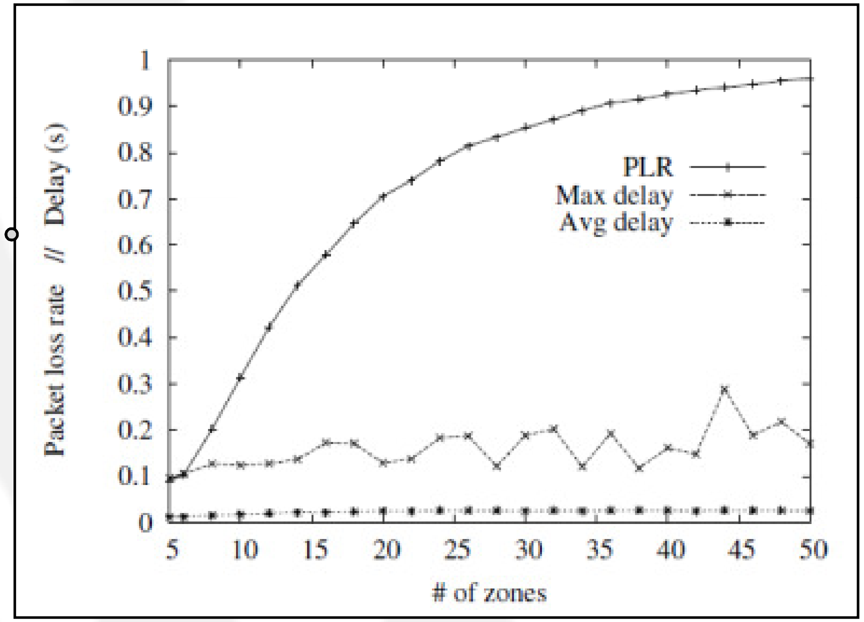


Network simulator (SCNSL)

Case study cont’d

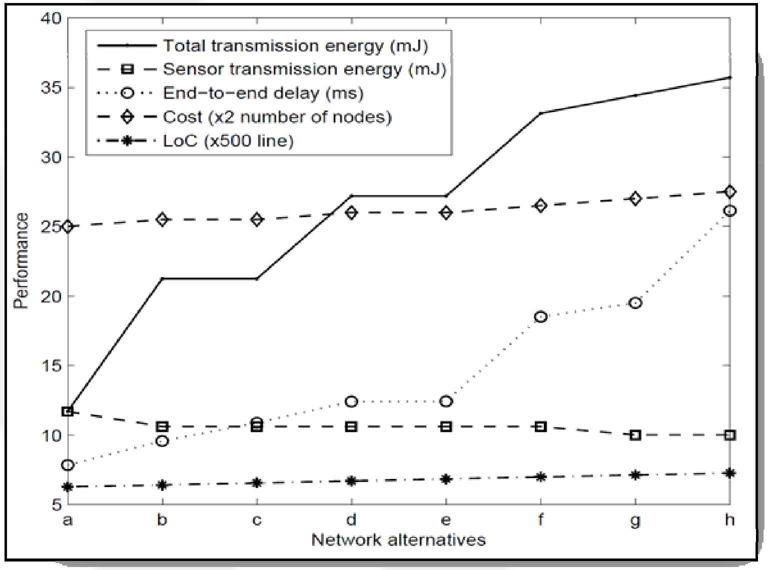


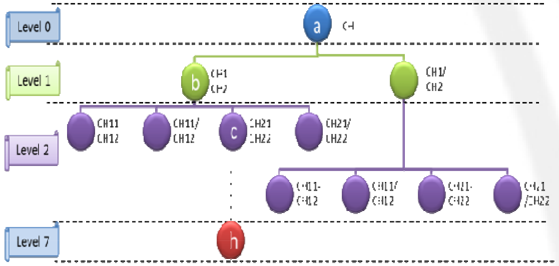
NW simulation statistics



Case study cont’d

NW simulation statistics





NW manipulation

Summary

- User requirements and constraints has been modeled by using UML/MARTE profile and simulated by SystemC/TLM at system view level
- Simulation results has been used to refine the user model
- Network synthesis tools have been used to solve the application problem
- Network solutions have been modeled and simulated by using SCNSL
- Network statistics have been used for the final refinement of application model
- **M**anipulation and **A**utomatic design-space exploration

Conclusions

- Some UML/MARTE diagrams and stereotypes have been used as a first time to represent the building blocks of a distributed embedded application
  - Elements from the MARTE specification have been applied to the context of distributed embedded applications
- Some gaps in MARTE standard have been identified concerning the representation of constraints and attributes related to error rate information
- SystemC code has been generated for both functional and network-aware simulation

A UML-centric design flow for networked embedded systems has been created