Electronic Systems Design
Synthesis Verification Testing Power Communication

Network synthesis for the design of NES

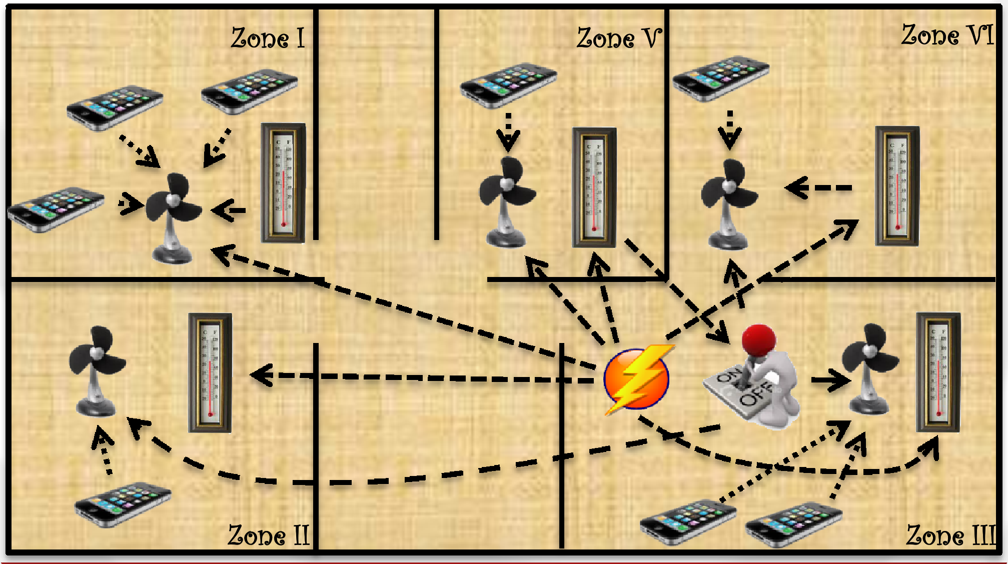
Davide Quaglia, Emad Ebeid

Outline

- Introduction and motivation
- Background and open issues
- Contribution
- Proposed methodology
  – Modeling requirements
  – System view simulation
  – Network synthesis
  – Network view simulation
- Experimental results
- Conclusion

Introduction and motivation

- **Distributed embedded application** as a single system to be designed
- Start from an abstract **Model-Based System** Specification
- Modeling and Analysis of Real-Time and Embedded Systems (**MARTE**) profile for the unified modeling language (**UML**)
- **Refinement steps** and simulations



Background and open issues

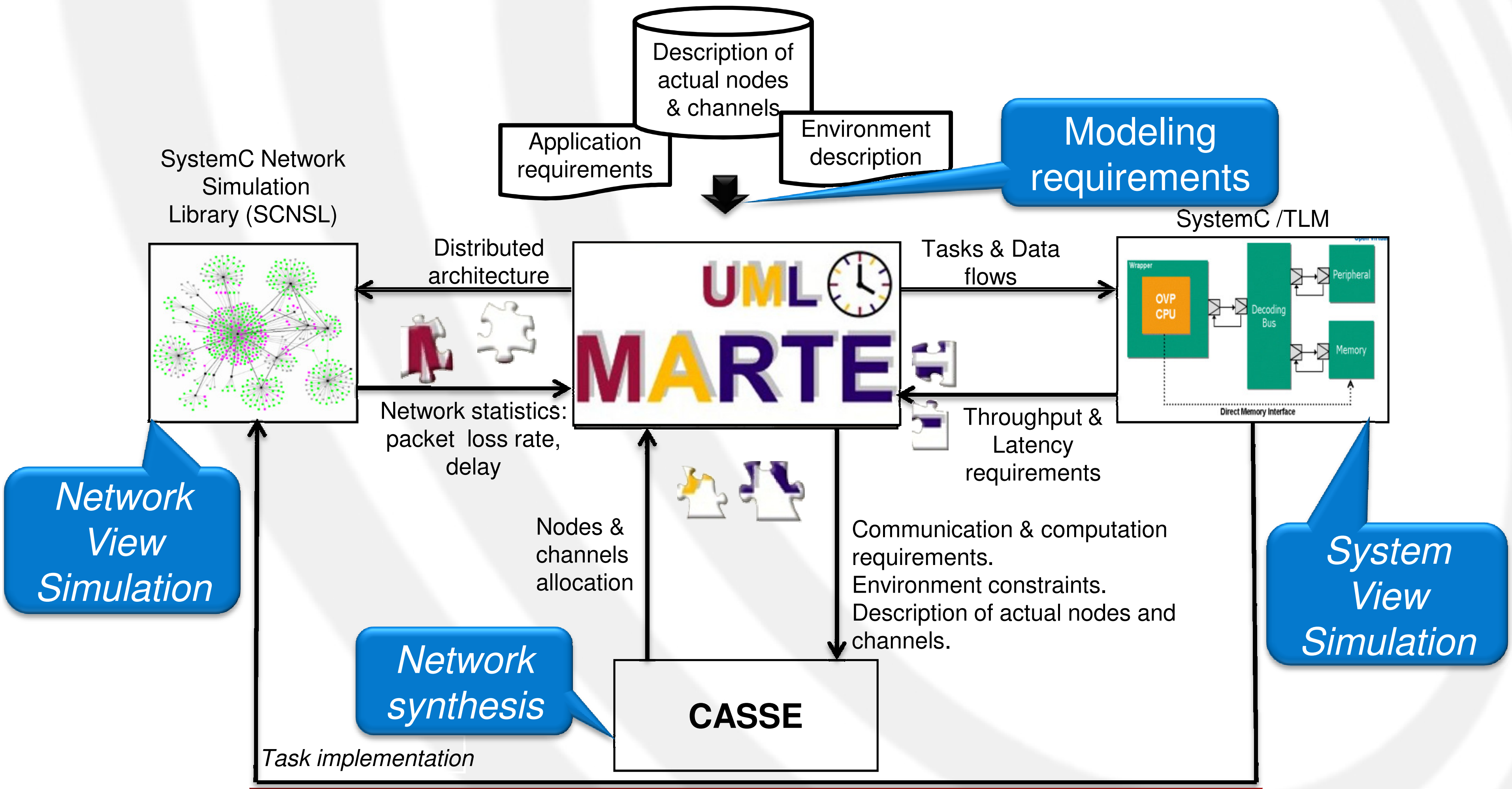
- Design of the network infrastructure starting from a library of nodes and channels (Network synthesis)
  - Communication Aware Specification and Synthesis Environment (CASSE), [FDL 2010]
- Open issue : *We need a standard representation of requirements (from the initial user specification) and solutions*

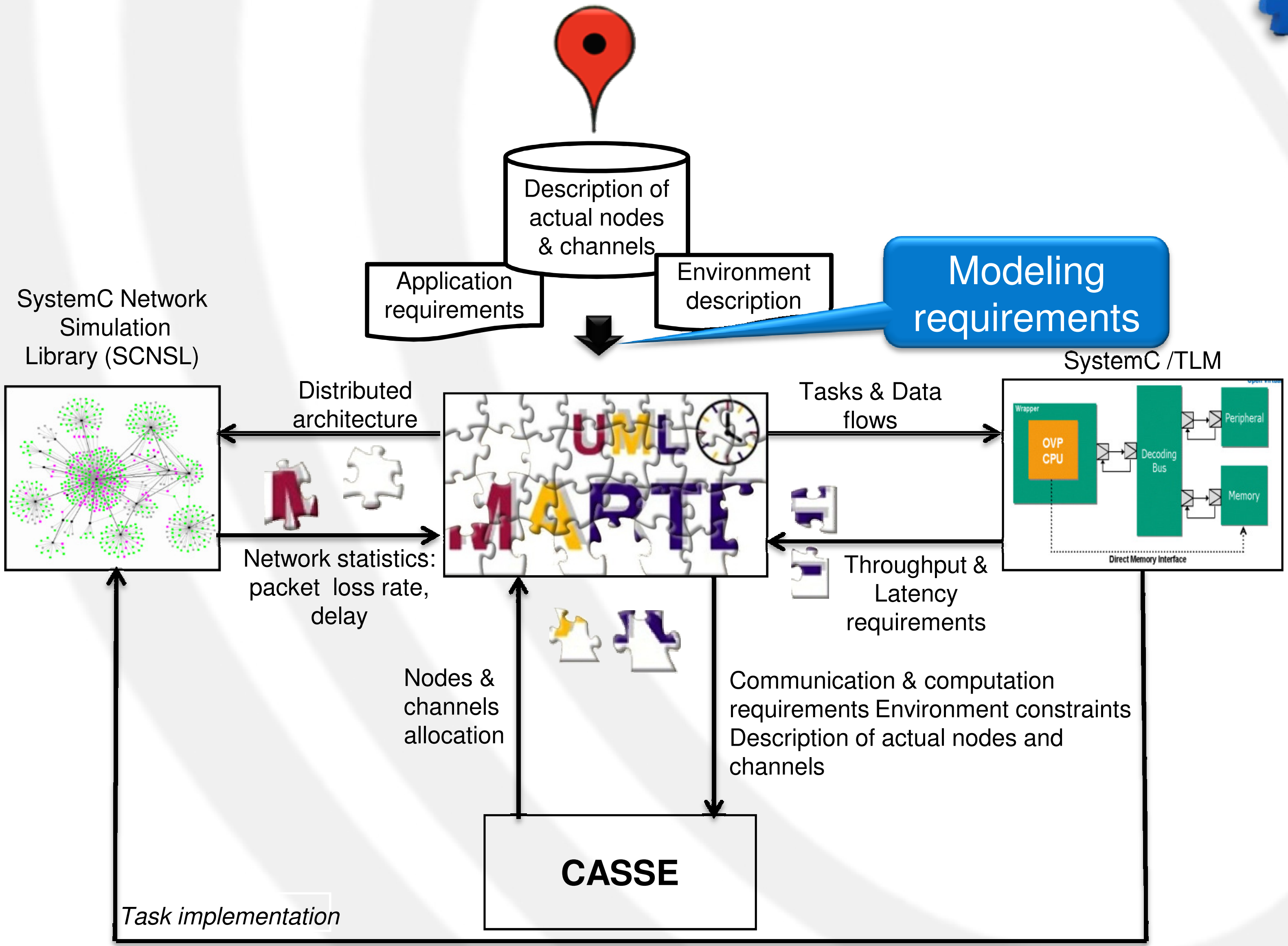
Main idea of this design flow

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

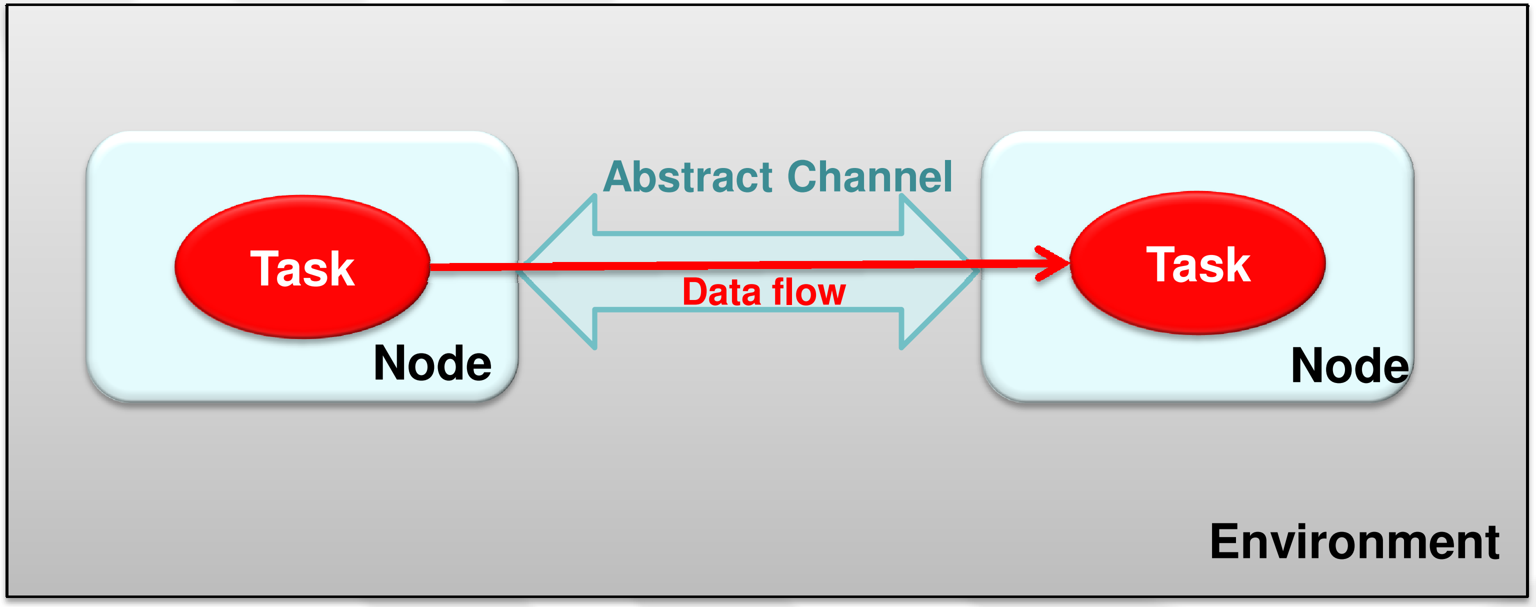
Proposed methodology




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

Modeling requirements

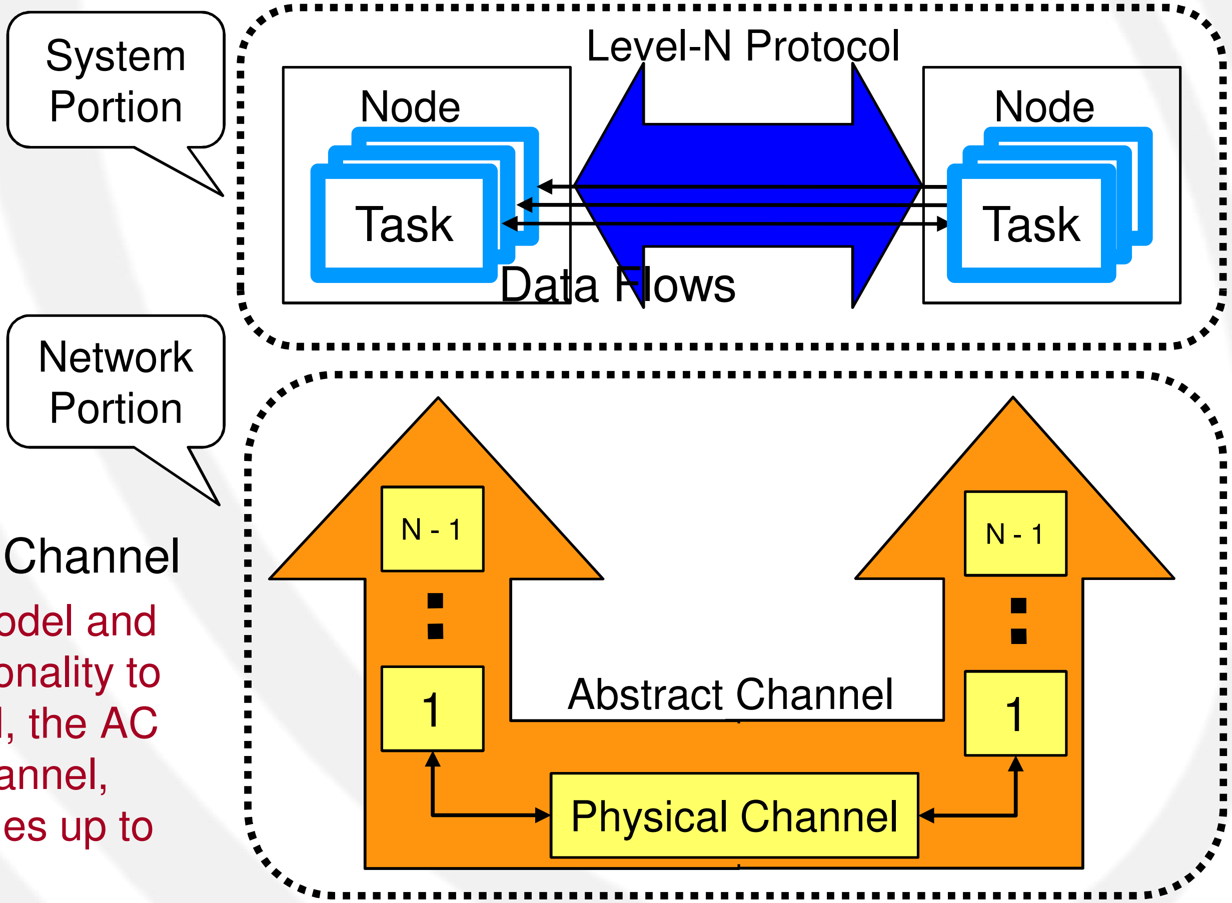
- The main entities to be represented in UML/MARTE are:
  - Tasks, data flows, nodes, channels and the external environment



Entities

- Task: basic function; it can have either digital input or digital output as well as both
- Data flow: unidirectional sequence of digital data between two tasks
- Node: networked embedded system hosting one or more tasks
- Abstract channel: abstraction of a channel hosting one or more data flows; two or more nodes can be connected to the same abstract channel
- Zone: piece of the external environment in which one or more nodes are deployed
- Contiguity: distance between zones

Abstract channel



Definition: Level-N Abstract Channel

Referring to the ISO/OSI model and assuming that the functionality to be designed is at level N, the AC contains the physical channel, and all the protocol entities up to level N - 1

Abstract channel (2)

- A level-N abstract channel with N>2 may contain intermediate systems
  - They are not considered as nodes in the network synthesis
  - But they are taken into account during the design of the abstract channel

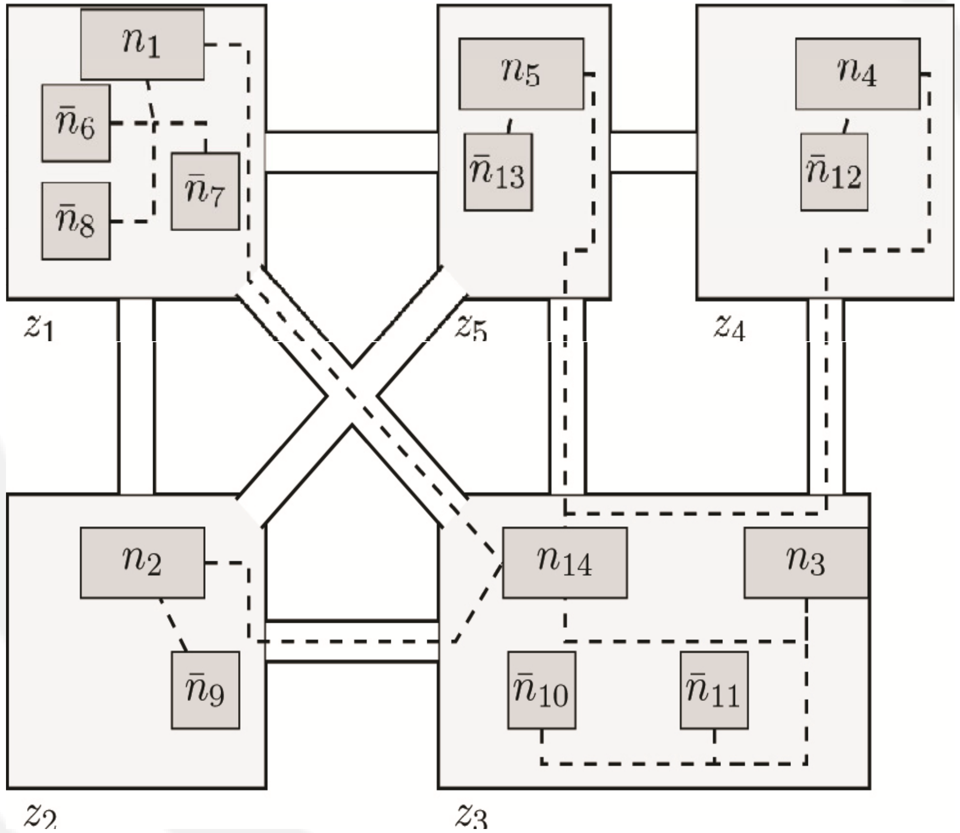
Zones and contiguity

- Many distributed embedded applications interact with the physical environment for data sampling and/or actuation
- Network infrastructure is constrained by the physical environment
  - Lenght of cables
  - Propagation of wireless signals
- The physical environment is modeled as a set of one or more Zones
- The distance between Zones is modeled by using Contiguity entity

Example

Node	*Type*	*Tasks*
n_{1-5}	A	t_4, t_5
n_{6-13}	D	t_3
n_{14}	B	t_1, t_2

Abstract Channel	*Type*	*Nodes*
a_1	Z	n_1, n_{6-8}
a_2	Z	n_2, n_9
a_3	Z	n_3, n_{10}, n_{11}
a_4	Z	n_4, n_{12}
a_5	Z	n_5, n_{13}
a_6	X	n_{1-5}, n_{14}



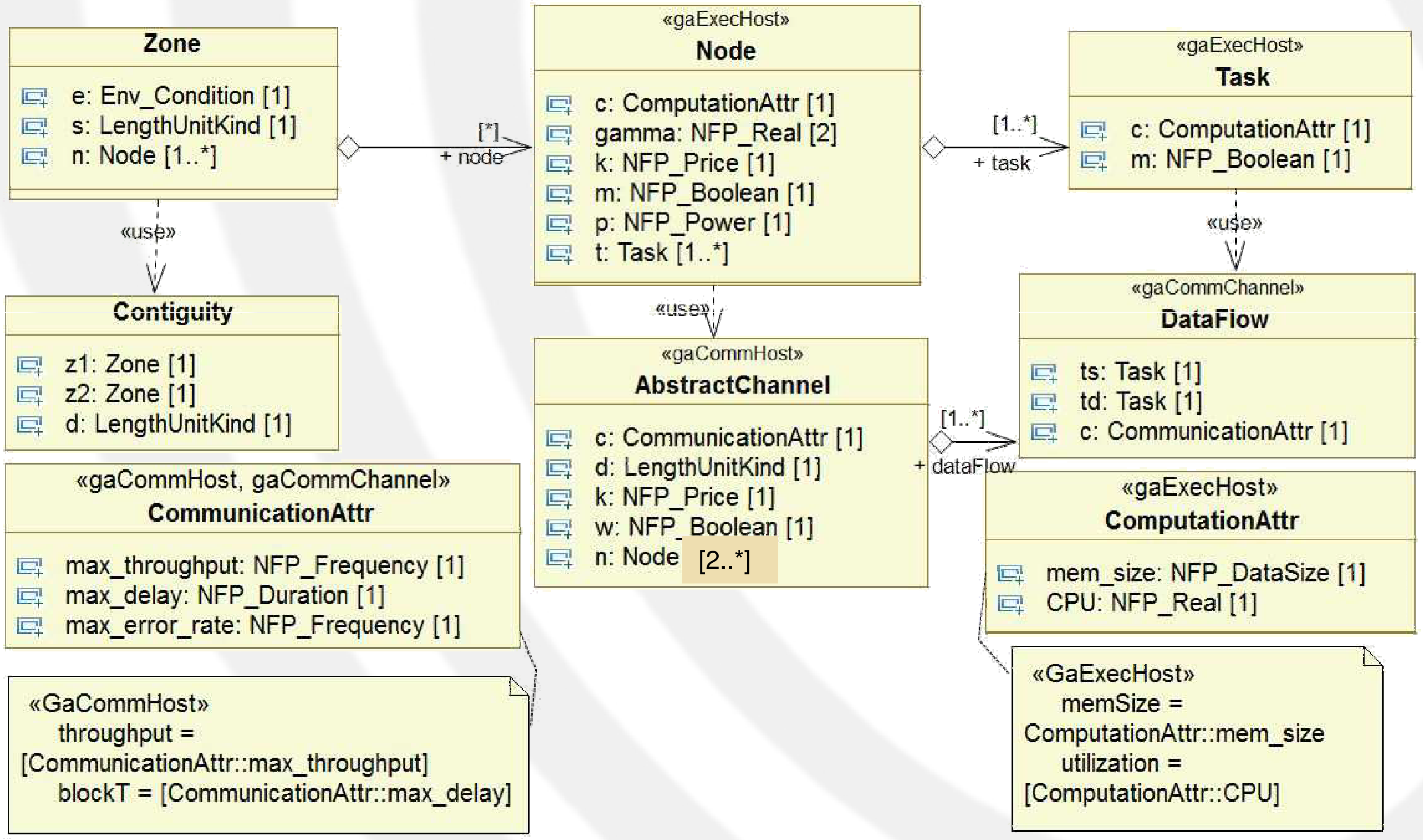
Goal of this flow

- The described flow allows to perform Network Synthesis = define the application in terms of
  - Tasks on nodes
  - Dataflows on abstract channels
  - Nodes on zones

Modeling entities in UML/MARTE

- Entities are modeled as classes inside UML/MARTE
  - Definition of their attributes
- Generic Quantitative Analysis Modeling (GQAM) sub-profile of MARTE profile is used to specify the semantics of some classes and their attributes

Class diagram of the entities



Extension of the entities

- Some entities need to be specialised for modeling actual applications
  - Task, dataflow, node and abstract channel
  - Attributes get values which are constant for all the instances of the specialized class

Extension of the entities (2)

- Task
  - Can be specialised into: sensor, actuator, controller, etc.
- DataFlow
  - Specific dataflows between subclasses of Task
- Node
  - Nodes taken from a datasheet
- Abstract channel
  - Well-known channels and protocols

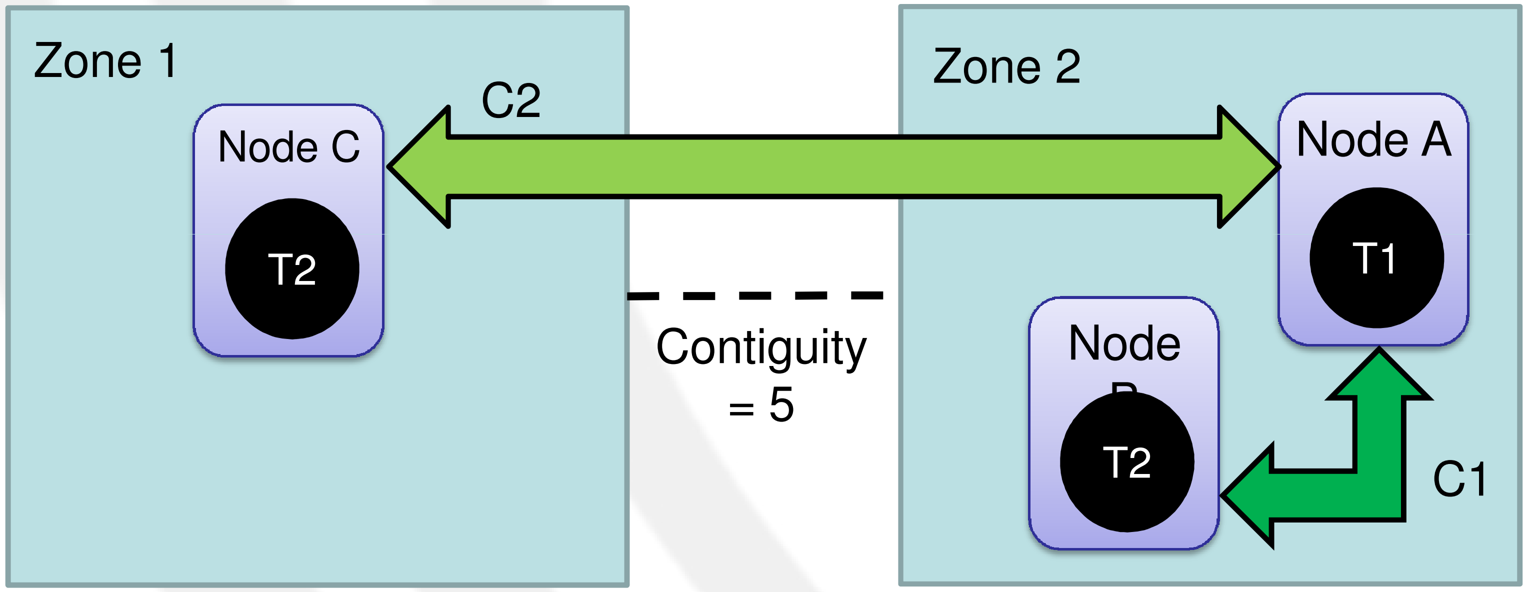
Requirements vs.availability

- Tasks and Nodes as well as Data Flows and Abstract Channels have similar attributes with different meaning
  - **Requirements** for Task and Data Flows
  - **Available resources** for Nodes and Abstract Channels
- Example: throughput
  - Max throughput *generated* by the Data Flow
  - Max throughput *supported* by the Abstract Channel

Distance in Abstract Channel and in Contiguity

- In Abstract Channels *d* attribute represents the communication range provided by it
- In Contiguity *d* attribute represents the distance between the corresponding zones
- When an Abstract Channel crosses two or more zones the communication range must be higher than the distance between zones

Example: Contiguity affects Abstract Channel choice



Even if data flows between T1 and T2 are the same, C1 and C2 cannot be the same since C1 does not cross Zones while C2 must cross Zones with Contiguity=5

Implicit constraints and application-level constraints

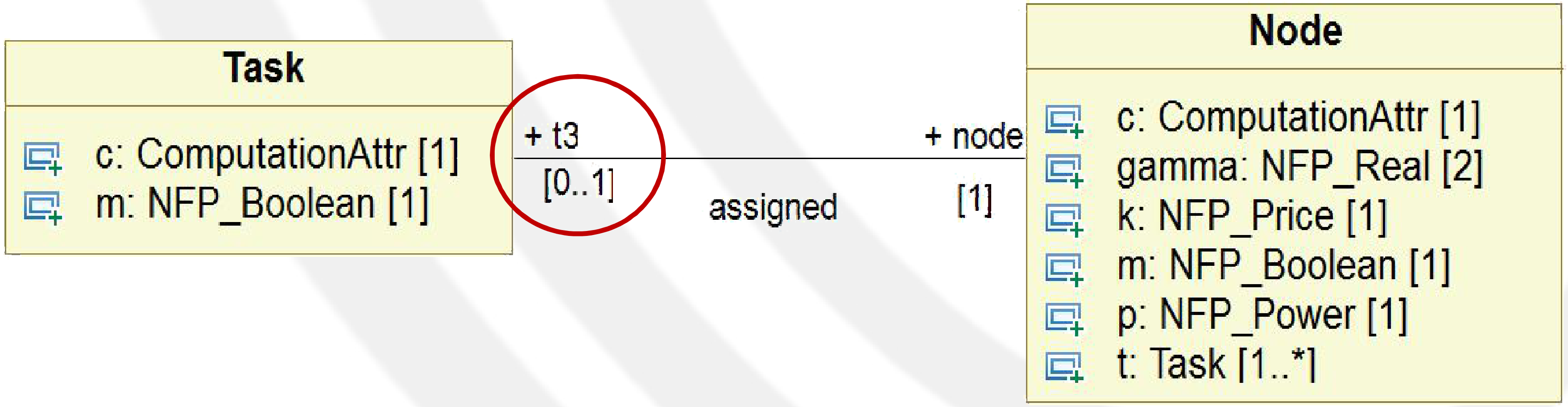
- Implicit constraints are given by the semantics of nodes, tasks, channels, zones
  - They are always present and therefore they don't need to be modeled in UML/MARTE
- Application-level constraints come from the customer
- Both of them will be put into the optimization problem of network synthesis

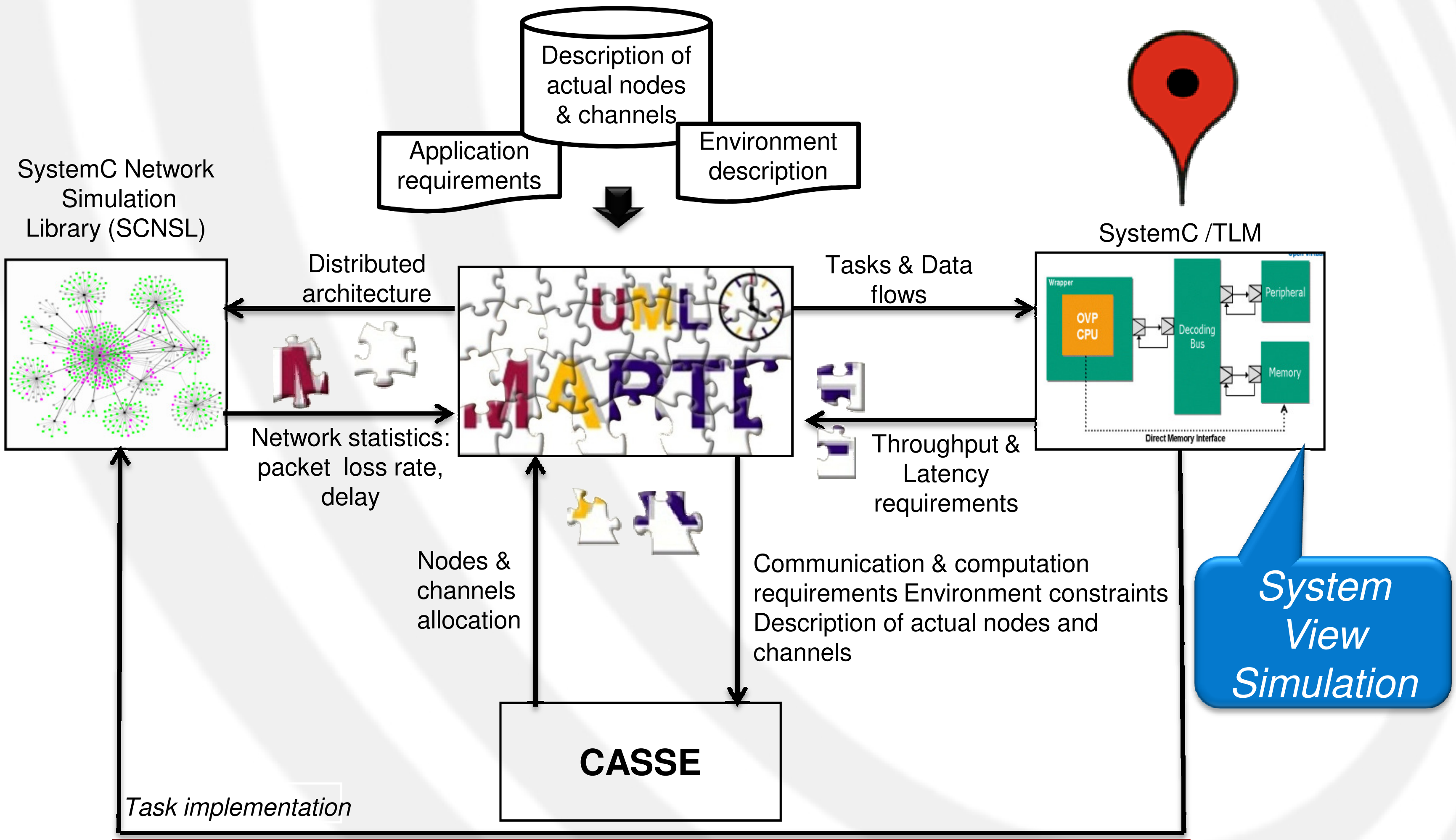
Examples of implicit constraints

- The memory available in a node must be greater than the sum of memory required by the hosted tasks
- The max delay provided by a channel must be lower than the max delay permitted by each of the hosted data flows
- The distance between two zones (attribute of contiguity) must be lower than the distance provided by all the channels crossing such zones

Modeling application-level constraints

- Example of constraint: “maximum one instance of task t3 can be assigned to each node”
- Application constraints are specified by using cardinality on the relationships between objects





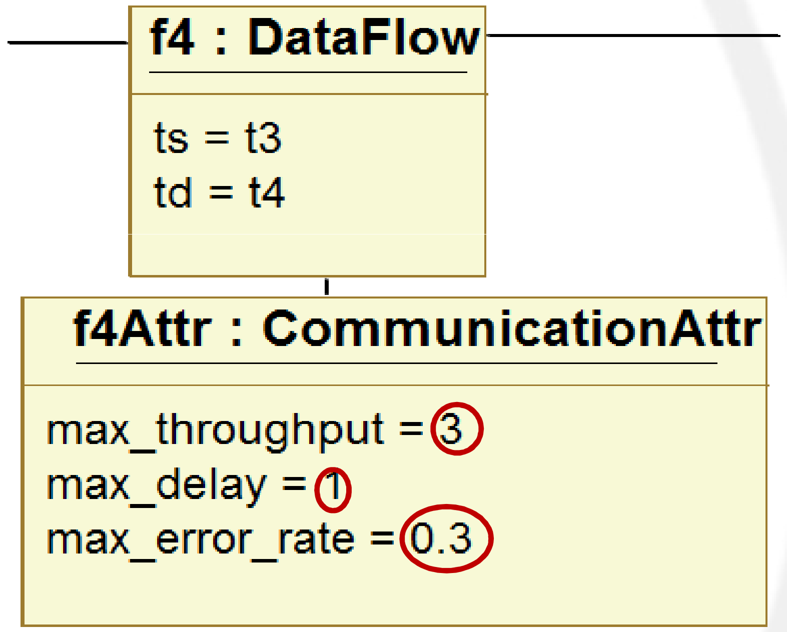
Description of actual nodes & channels
Application requirements
Environment description
SystemC Network Simulation Library (SCNSL)
Distributed architecture
Network statistics: packet loss rate, delay
UML MARTE
Tasks & Data flows
SystemC /TLM
Wrapper
OVP CPU
Decoding Bus
Peripheral
Memory
Direct Memory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

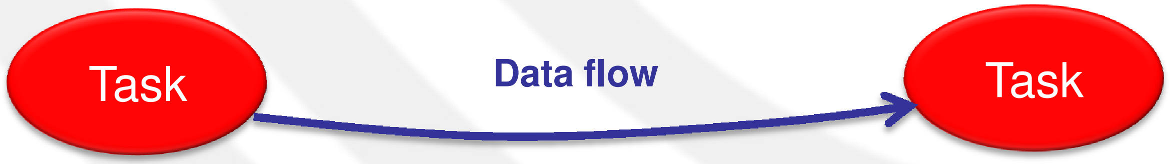
System View simulation

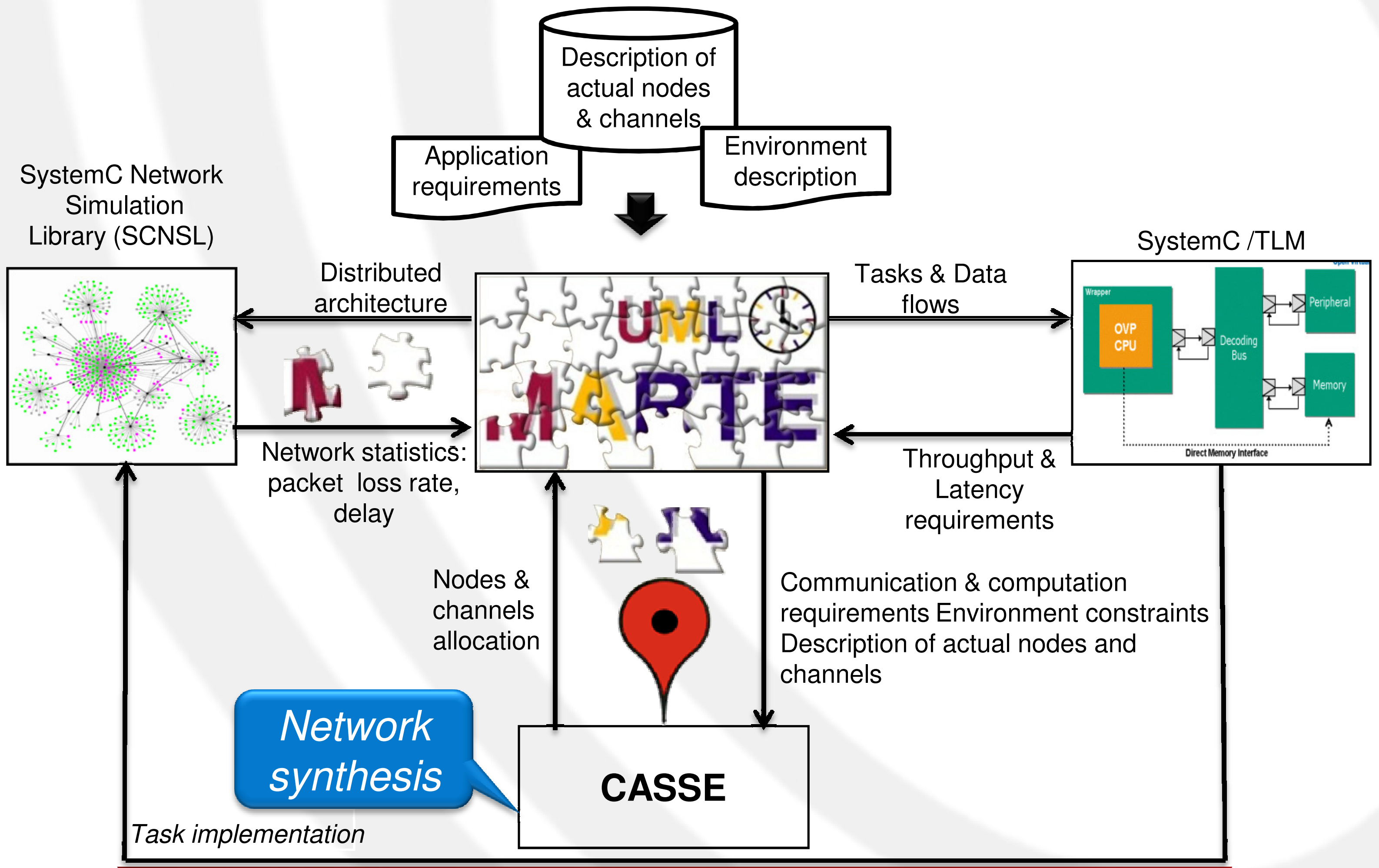
- A sc_module class for Task entity is provided
  - It must be specialized for application-specific tasks
- Task instances are created and connected through TLM sockets representing data flows

System view simulation

- UML/MARTE class diagram is extracted and used to generate SystemC/TLM model
  - Other ways to generate executable models in [Villar 2009] and [Vanderperren 2008]
- Execution of the SystemC model
  - Validate the 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

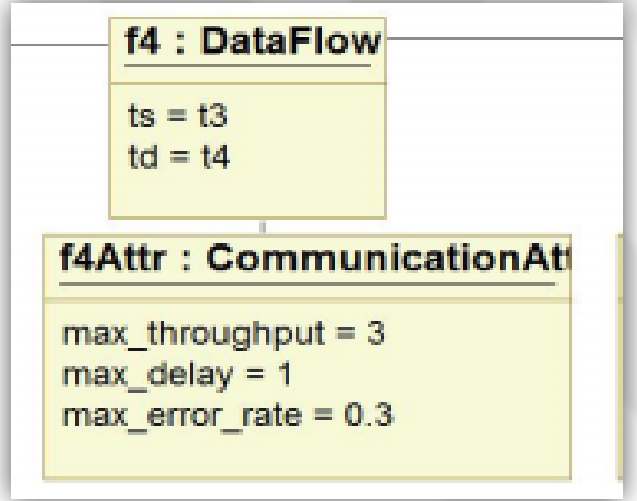





Description of actual nodes & channels
Application requirements
Environment description
SystemC Network Simulation Library (SCNSL)
Distributed architecture
Network statistics: packet loss rate, delay
UML
MARTE
Tasks & Data flows
SystemC /TLM
Wrapper
OVP CPU
Decoding Bus
Peripheral
Memory
Direct Memory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



f4 : DataFlow

ts = t3
td = t4

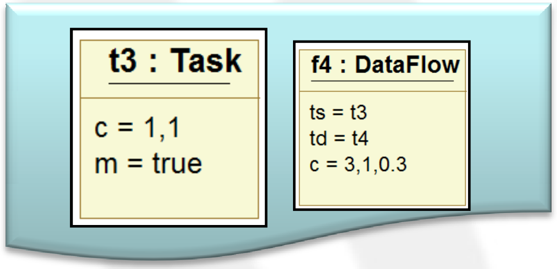
f4Attr : CommunicationAt

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



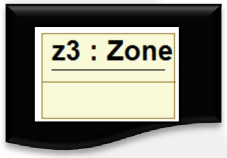
t3 : Task

c = 1,1
m = true

f4 : DataFlow

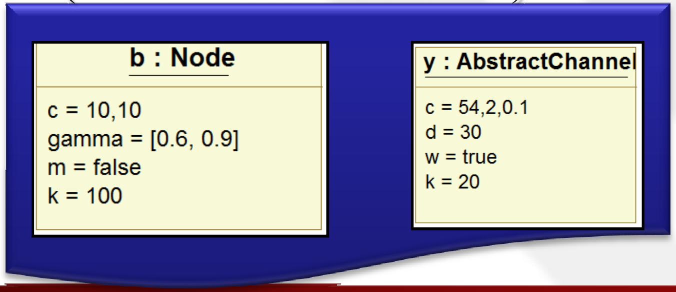
ts = t3
td = t4
c = 3,1,0.3

The building geometry



z3 : Zone

Technological library
(network nodes and channels)



b : Node

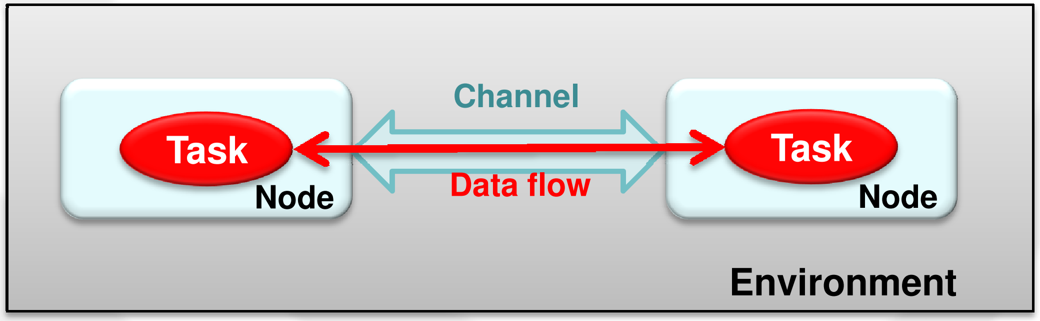
c = 10,10
gamma = [0.6, 0.9]
m = false
k = 100

y : AbstractChannel

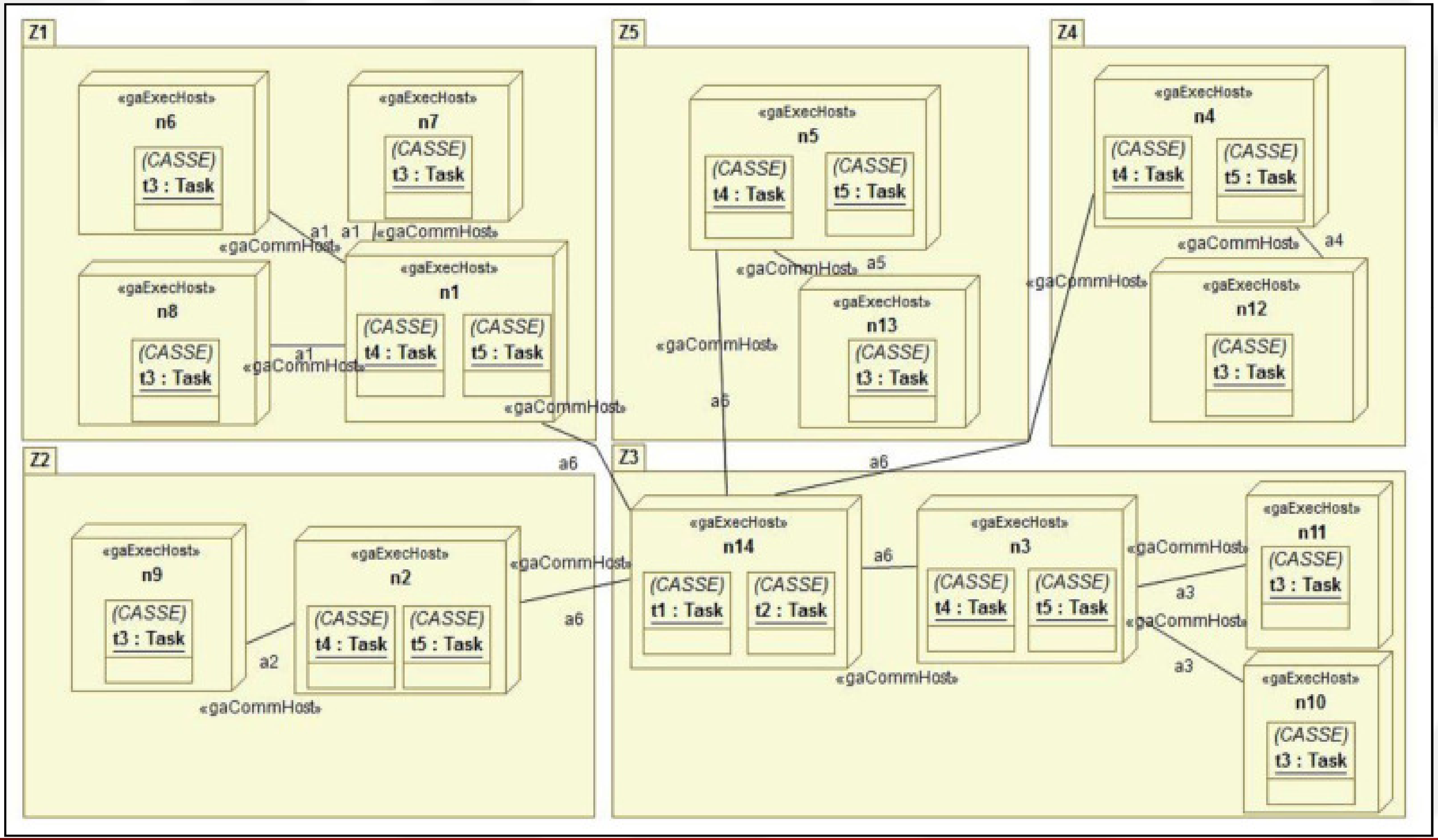
c = 54,2,0.1
d = 30
w = true
k = 20

Network Synthesis

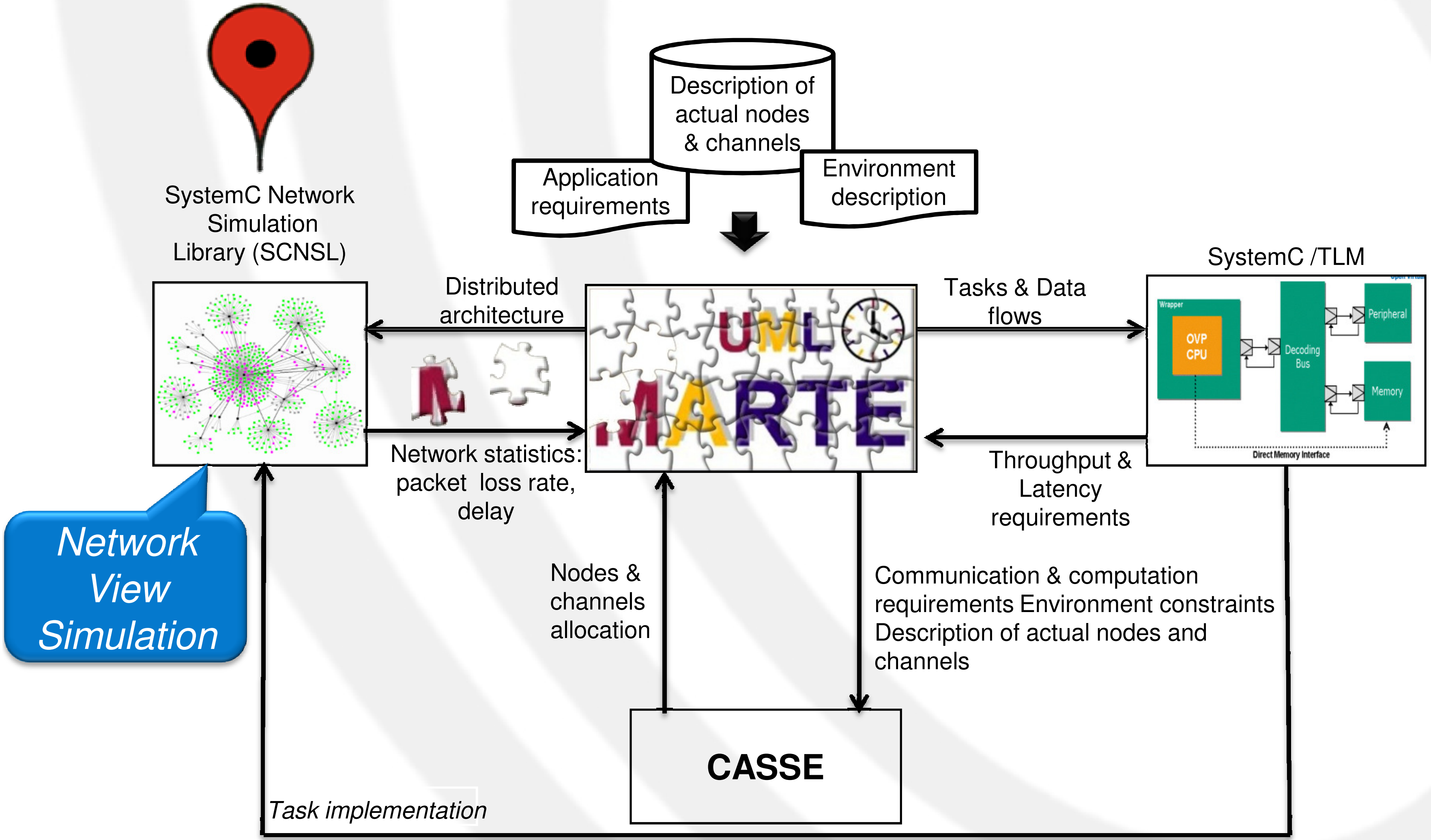
UML deployment diagram:
Assignment of *tasks inside nodes* and *data flows inside channels*



UML deployment diagram



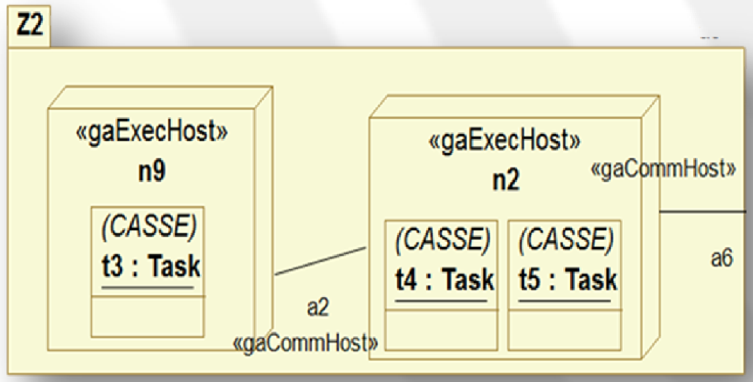
Z1
«gaExecHost» n6 (CASSE) t3 : Task
«gaExecHost» n7 (CASSE) t3 : Task
«gaExecHost» n8 (CASSE) t3 : Task
«gaExecHost» n1 (CASSE) t4 : Task (CASSE) t5 : Task
a1
a1
a1
«gaCommHost»
«gaCommHost»
«gaCommHost»
«gaCommHost»
Z5
«gaExecHost» n5 (CASSE) t4 : Task (CASSE) t5 : Task
«gaExecHost» n13 (CASSE) t3 : Task
«gaCommHost» a5
«gaCommHost»
a6
Z4
«gaExecHost» n4 (CASSE) t4 : Task (CASSE) t5 : Task
«gaExecHost» n12 (CASSE) t3 : Task
«gaCommHost» a4
«gaCommHost»
Z2
«gaExecHost» n9 (CASSE) t3 : Task
«gaExecHost» n2 (CASSE) t4 : Task (CASSE) t5 : Task
a2
«gaCommHost»
«gaCommHost»
a6
a6
Z3
a6
«gaExecHost» n14 (CASSE) t1 : Task (CASSE) t2 : Task
«gaExecHost» n3 (CASSE) t4 : Task (CASSE) t5 : Task
a6
«gaCommHost»
«gaExecHost» n11 (CASSE) t3 : Task
«gaExecHost» n10 (CASSE) t3 : Task
«gaCommHost» a3
«gaCommHost» a3
«gaCommHost»



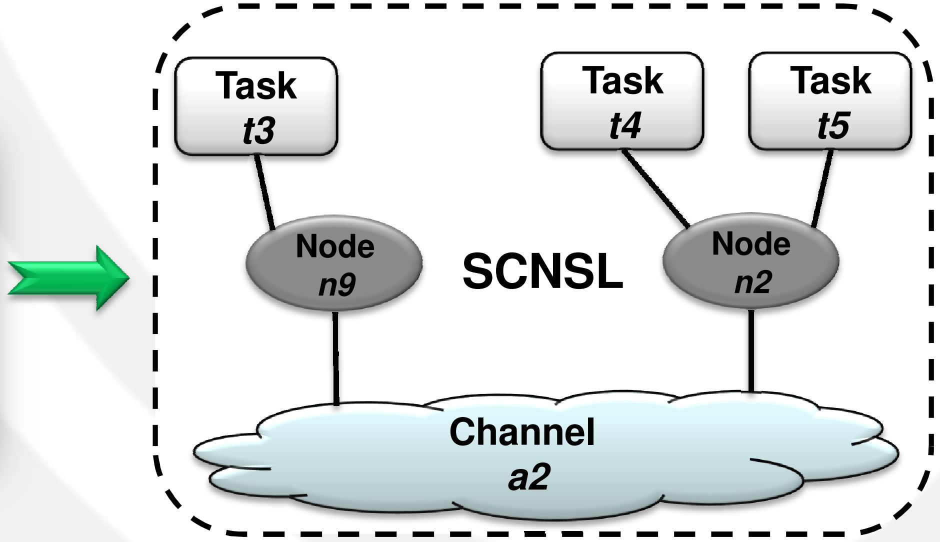
Description of actual nodes & channels
Application requirements
Environment description
SystemC Network Simulation Library (SCNSL)
Distributed architecture
Network statistics: packet loss rate, delay
Tasks & Data flows
SystemC /TLM
Wrapper
OVP CPU
Decoding Bus
Peripheral
Memory
Direct Memory Interface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



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);

Design the network before designing nodes and channels

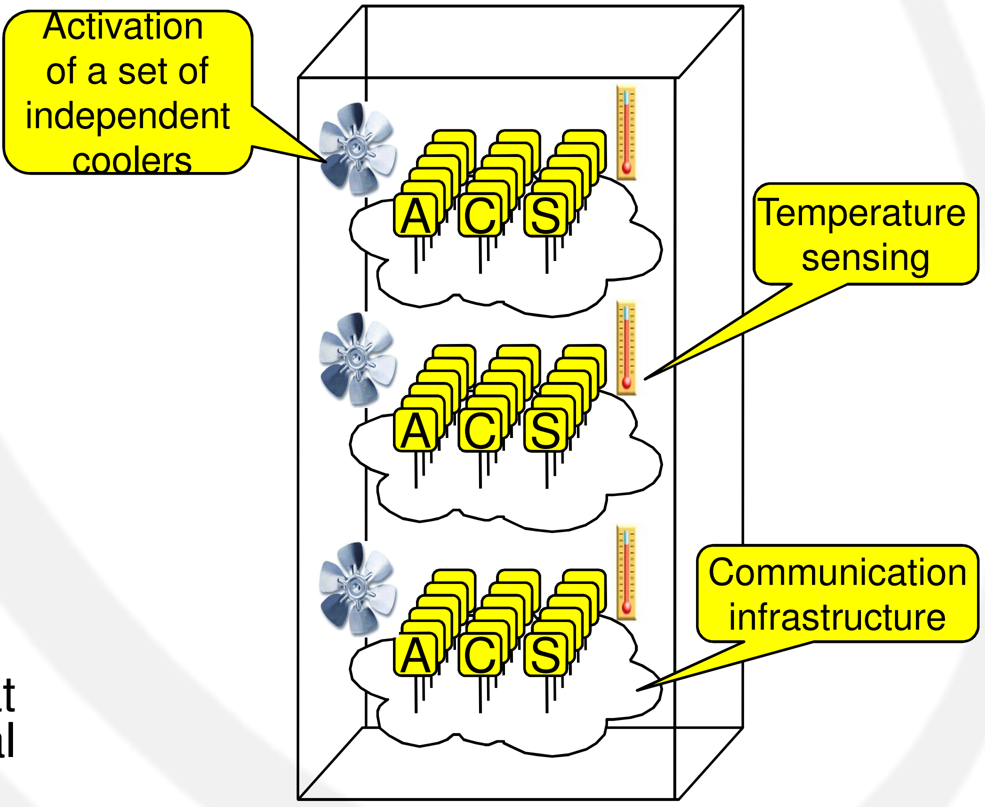
- The shown flow allows to define the application in terms of
  - Tasks on nodes
  - Dataflows on abstract channels
  - Nodes on zones
- The network organisation is designed before nodes and channels

Design the network before designing nodes and channels

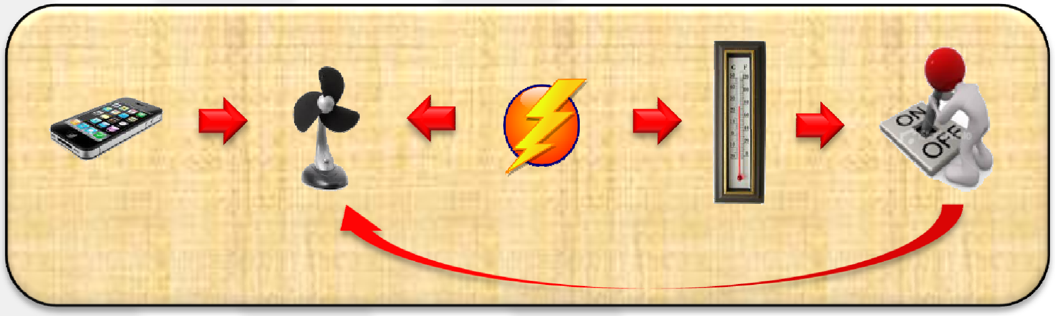
- At the end of this flow nodes and channels can be considered in an actual way
  - Nodes are either taken from the market or designed with traditional techniques for embedded systems
  - Channels are created by assembling
    - Physical channels: wired, wireless
    - Protocols
    - Intermediate systems: switch, router

Case study: temperature control application of a skyscraper

- In each room, there is at least one sensor *S* which keeps track of the local temperature. Collected data are sent to controllers *C*, which send commands to actuators A, e.g., coolers.
- Controllers can be either fixed (e.g., on the wall of each room) or embedded into mobile devices to let people set the preferred temperature of the environment around them.
- A centralized controller is also present to perform some global operations on the temperature control system, e.g., to ensure that room settings comply with the total energy budget.


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

Case study

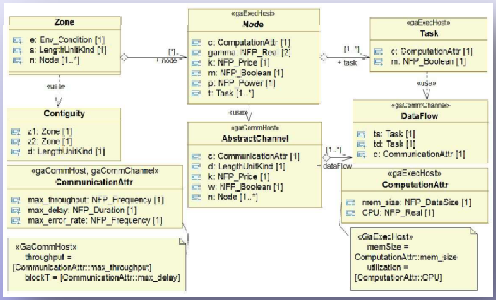


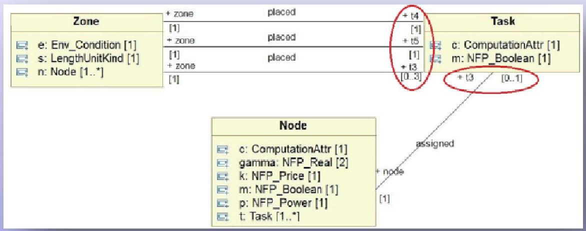
+

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

User requirements

System View modeling

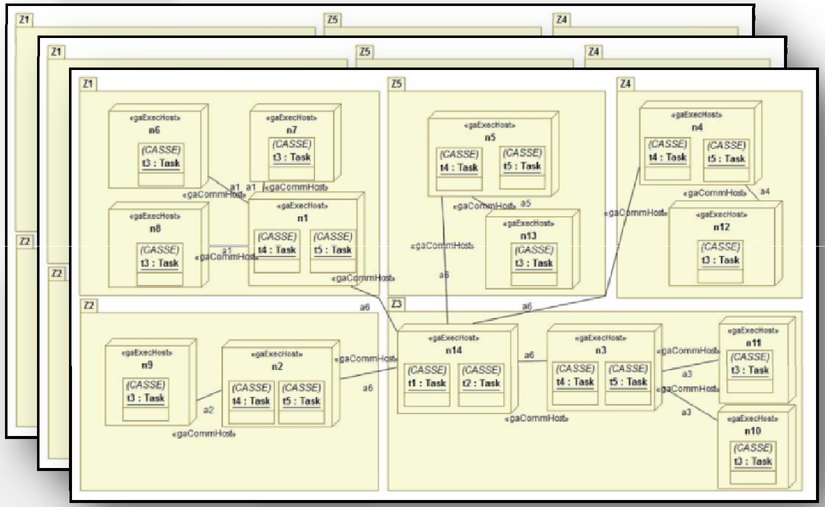


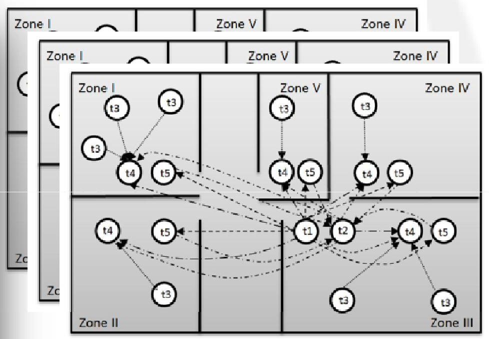


NW synthesis tool

Case study

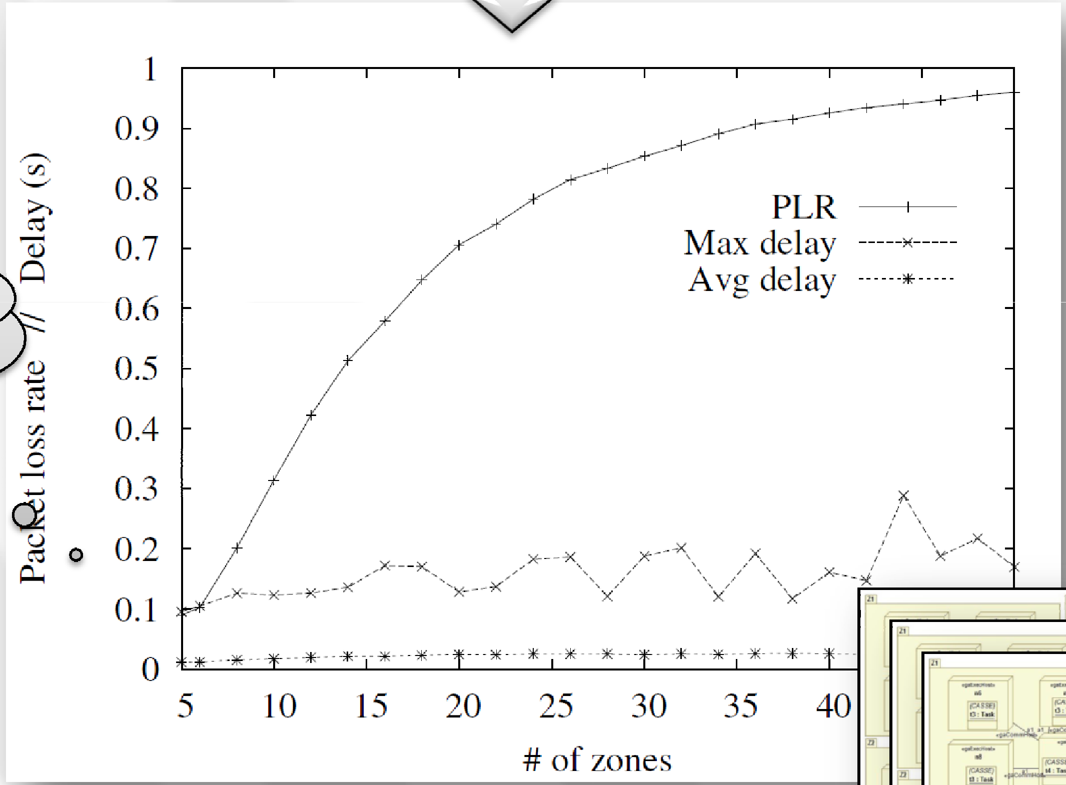
NW view modeling



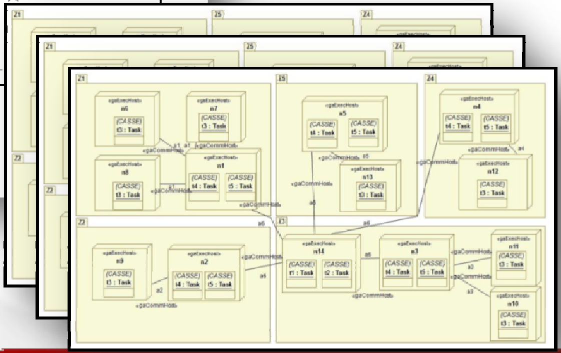


Network simulator (SCNSL)

Case study



NW simulation statistics



Conclusions

- Some UML/MARTE diagrams and stereotypes have been used to represent the building blocks of a distributed embedded application
  - Class diagram, GQAM and deployment diagrams
- SystemC code has been generated for both functional and network-aware simulation

A MARTE-centric flow for network synthesis