Modeling of Communication Infrastructure for Design-Space Exploration

F. Fummi, G. Lovato, D. Quaglia, F. Stefanni
Dep. of Computer Science — University of Verona, Italy
[franco.fummi|davide.quaglia|francesco.stefanni]@univr.it
giovanni.lovato@aldu.net

***Abstract*—Computer-aided design has been traditionally applied to computers and embedded systems but not to the communication infrastructure among them. The paper contributes to fill this gap by proposing to use a mathematical language to model a distributed application in terms of tasks, hosting nodes, and interactions with the environment. Tasks are described in terms of computation and communication requirements also in relationship with state-of-the-art languages for system specification. Entities and relationships are introduced to relate tasks, data flows and environmental data to network nodes, channels among them and communication protocols. The resulting attributes and constraints can be used during a further design-space exploration to synthesize automatically a suitable communication infrastructure. The approach can be applied to significant applications, e.g., those based on wireless sensor networks and peer-to-peer networks. An example related to building automation is also reported to demonstrate the potentiality of the framework.**

***Index Terms*—design-space exploration, distributed systems, networked embedded systems**

I. Introduction

Today's distributed applications are becoming more and more complex, involving many different tasks spread over hundreds or thousands of heterogeneous network nodes connected through different types of channels and protocols. In this context, computer-aided design should be fruitfully applied not only to each node, as currently done in the context of electronic systems design, but also to the communication infrastructure among them.

For instance, let's consider the scenario depicted in Figure 1, reporting the temperature control application of a skyscraper. In each room, there is at least one sensor (in Figure denoted with *S*) which keeps track of the local temperature. Collected data are sent to controllers (denoted with *C*), which send commands to actuators (denoted with *A*), e.g., coolers. Controllers can be either fixed (e.g., on the wall of each room) or embedded into mobile applications 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.

To properly design this application, many aspects should be taken into account, e.g.:

- different concurrent tasks are involved, e.g., temperature sensing, and cooler activation;

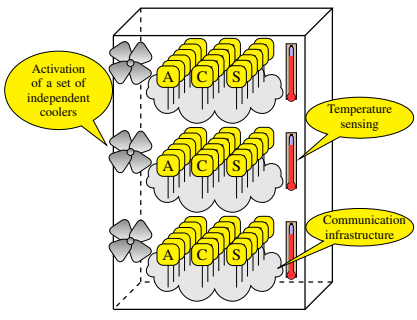

Fig. 1. Example of application for building automation.

- many instances of each task are present, e.g., a mobile controller for each user;
- tasks are hosted by physical devices with different capabilities, e.g., mobile vs. fixed embedded systems;
- physical channels can be either wireless or wired;
- communication protocols can be either reliable or un-reliable, best effort or with priority;
- the position in which sensors, controllers and actuator are placed affects the communications among them, the sensing performance and the control policy.

Instead of traditional point-to-point applications, e.g., email transfer, the design goal for this kind of applications is the good behavior of the *global distributed application*, e.g., satisfying the largest number of users' preferences while minimizing power consumption.

Distributed applications pose new questions to designers, traditionally mainly interested in the specification of each single network node. Some possible questins are:

- How many network nodes are required?
- How many network nodes are supported at most?
- Which is the best assignment between tasks and network nodes by taking into account tasks' requirements and nodes' capabilities?
- Given a partition of the environment into zones which is the best task-zone assignment?
- Given an assignment of tasks to network nodes, which channel types and protocols should be used among them to satisfy such requirements?
- Is the introduction of additional intermediate systems (e.g., routers, range extenders) useful to improve communication performance?

The answers to these questions lead to the *complete definition of the communication infrastructure*. Similarly to the definition of components in electronic system design, we refer this process with the name *network synthesis*; it consists in the following steps:

1) specification of the communication requirements of the application to be designed;
2) progressive refinement of the specification through design-space exploration;
3) mapping of the solution onto actual objects, i.e., physical channels, protocols, intermediate systems.

Related work on the design of networked systems can be divided into two categories (Section II):

1) focused on HW/SW design and not considering the network as part of the design space;
2) focused on network-related problems, but without a general design methodology taking into account the whole application.

This lack of network modeling may lead to non-optimal solutions in the system design, since recent work demonstrated that HW/SW design and network design are correlated [1].

The main contributions of this paper are:

- a formalization that allows to represent the communication aspects of the applications;
- an extended design methodology which considers the communication infrastructure as a dimension of the design space;

- a case study to show the potentiality of the approach.

The paper gives the mathematical basics of the network synthesis and optimization problem, but no concrete algorithm for their solution. This issue will be addressed in future works.

The proposed work is a first step towards the *computer-aided synthesis of the communication infrastructure*, i.e., the automatic choice of channel type, protocols and intermediate systems to support a distributed application. Synthesis is currently done for hardware components but, as far as we know, this is the first attempt to apply the same approach to the communication infrastructure.

The rest of the paper is organized as follows. Related work is reported in Section II. The global design flow is introduced in Section III. The proposed formulation of the problem is reported in Section IV. A first discussion of the synthesis methodology is provided in Section V. A case study is described in Section VI. Conclusions and future work are reported in Section VII.

II. Related Work

The synthesis of distributed systems has been addressed by many research works, in different fields.A virtual architecture has been proposed in order to simplify the synthesis of algorithms for WSN [2]. Some network information, like the topology and high-level functionality, are used to configure the virtual architecture. However this work is mainly focused on the application part of the system rather than on communication aspects. Platform-Based Design (PBD) has been adopted to project WSN for industrial control [3]. As usual in PBD, the application is designed at high level and then mapped onto a set of possible actual candidates for the nodes. However, no guideline is provided about the selection of the appropriate network architecture and communication protocol. Scope-based techniques have been proposed in macroprogramming to specify complex interactions between heterogeneous nodes of a WSN [4]. However, the number of nodes and the network topology are an input of the technique not a result as in our approach.

The design of Networks-on-Chips (NoC) offers an example of system-network integrated approach which is close to the one proposed in this paper, since they have a simplified version of a packet-switched network. The internal NoC topology can be either irregular or have regular form, e.g., mesh, tour, ring, and butterfly [5]; its design presents problems similar to the one of traditional packet-based networks. Some researches have suggested that the mesh topology is most efficient in terms of latency, power consumption and implementation simplicity [6]. Routing protocols are a design issue, too. In NoC's routing can be deterministic or adaptive according to the current state of the network and to appropriate quality-of-service (QoS) policies [6]. Regarding the latter, best effort and guaranteed throughput have been proposed [7]; they are very similar to those defined for TCP/IP [8]. The problem of the optimal mapping of tasks onto NoC's cores is known to be NP-hard. In some works, heuristics based on graph-decomposition techniques have been used [6], [9]. A Mixed Integer Linear Programming (MILP) formulation of the problem has been proposed [10]. It assumes a regular 2D mesh topology, and shortest-path static routing. This methodology allows two different optimization criteria, i.e., minimization of the average hop distance and minimization of the maximum link bandwidth.

Synthesis of communication protocols is another research topic related to the focus of this work. Automatic tools have been adopted to derive the actual implementation of protocols specified through finite state machines [11], [12], Petri Nets [13], trace models [14], and languages like LOTOS [15]. All these works are focused on particular problems, and do not provide any general model or a global design flow. This paper tries to fill such a lack.

III. Design flow of distributed systems

Figure 2 shows the design flow of a distributed system; the right part of the Figure depicts the state-of-the-art design flow of embedded systems while the left part of the Figure introduces the proposed additional steps for the design of the network.

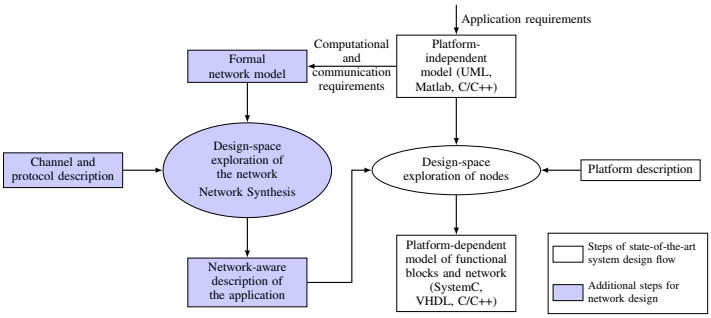

Fig. 2. The proposed design flow for distributed systems: new steps for network design are added to the state-of-the-art system design flow.

In the traditional system design flow a platform-independent model of the system is created starting from application requirements, both functional and non-functional. Behavior in this specification is usually expressed through languages like UML, MARTE and C/C++ or through the use of tools like Matlab/Simulink/Stateflow.

This specification, together with a description of the target platform, is the subject of a design-space exploration which maps tasks onto HW and SW components for the target platform.

This flow is perfect for single embedded systems but in case of distributed applications made of many embedded systems it lacks a specific path devoted to the design of the communication infrastructure between them. For this reason, new steps are proposed as depicted in the left side of Figure 2.

Starting from the platform-independent model, a *formal network model* is derived. It contains computational cost of each task, e.g., in terms of CPU and memory usage, and the requirements of communication flows between tasks, e.g., their throughput and permitted latency.

This formal model of the network, together with a description of actual channels and protocols, is the subject of design-space exploration aiming at searching the optimal solutions of the constraints expressed in the mathematical model. Actual channels and protocols are chosen according to their affinity to the optimal solutions. This last phase is named *network synthesis*. The final result is a *network-aware description of the application* with the mapping of application tasks onto network nodes, their spatial displacement, the type of channels and protocols among them, and the network topology.

The network-aware description of the application contains important information for the design of each node of the network, i.e., the list of tasks assigned to it. For this reason, this description is used as input in the traditional design-space exploration of each node as reported in the right part of Figure 2.

IV. The formal network model

This Section defines the entities and relationships which constitute the formal network model depicted in Figure 2. The structure of this model is motivated by its goal which is network synthesis defined as follows.

Definition 1. *Network synthesis is a design process which starts from a high-level specification of a distributed system and finds an actual description of its communication infrastructure in terms of mapping of application tasks onto network nodes, their spatial displacement, the type of channels and protocols among them, and the network topology.*

Figure 3 reports a system under design with a partitioning between the *system portion* and the *network portion*; the former is the subject of traditional system design while the latter is the subject of network synthesis. Both partitions will be described with entities and relationships in this Section.

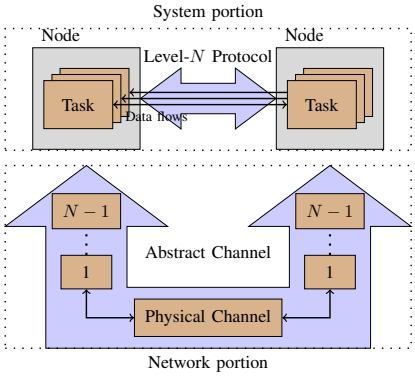

Fig. 3. System/network partitioning for network modeling and synthesis.

The system portion consists of *network nodes* which contains *tasks*, i.e., sequences of operations accomplished to implement the overall behavior of the distributed system. *Data flows* are exchanged between tasks. A communication protocol is established between nodes to convey the data flows of the hosted tasks. The network portion consists of the physical channel and all the protocol entities which permit the communication between the nodes. To better describe the network portion the entity named *abstract channel* will be introduced. To understand this entity, let us consider two examples. In the first example, the design goal is a complete wireless device; therefore, the system portion includes CPU, memory, and all the transmission components up to the antenna while the network portion could be represented by the radio channel. In the second example, the design goal is a a temperature control application; therefore, the system portion is the control algorithm while the network portion could be a reliable byte-oriented data flow as provided by TCP/IP. The concept of Abstract Channel (AC) unifies the physical channel of the first example and the transport channel of the second one. AC is defined as follows.

Definition 2. *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 $N-1$.*

The AC connects network nodes whose tasks are implemented using level-N protocols.

In this work, we assume that all the tasks of the application under design belong to the same ISO/OSI level, i.e., all the Abstract Channels contain the physical channel and all the protocol levels up to $N-1$.

In the rest of the Section, entities will be described in details together with relationships between them. Most of them are described as multiset, since more instances of the same type could exist.

A. Tasks

A task represents a basic functionality of the whole application. From the point of view of network synthesis the focus is not on the description of the functionality in itself but rather on its computational and mobility requirements which affect the assignment of tasks to network nodes.

A task t is defined as follows:

$$\begin{aligned} &t = [c, m] \in \mathscr{T} \text{ where:} \\ &c \in \mathbb{R}^n \quad m \in \mathbb{B} \end{aligned} \tag{1}$$

$\mathscr{T}$ is a multiset of tasks. The vector named $t.c$ represents the resource requirements to perform task's activity. For instance, in the early stage of the design flow, when detailed requirements are not available, it can be described by a single abstract number ($t.c \in \mathbb{R}$). As more details are available, it could be described by many more components, e.g., the memory usage and the computation time ($t.c \in \mathbb{R}^2$). Choosing the appropriate components of $t.c$ is a designer's responsibility. The attribute named $t.m$ is a boolean attribute representing the possible mobility requirement of the task.

B. Data Flows

A data flow (DF) represents communication between two tasks. For network synthesis, the focus is on the communication requirements between tasks which affects the choice of channels and protocols between nodes hosting the involved tasks.

A data flow f is defined as follows:

$$\begin{aligned} &f = [t_s, t_d, c] \in \mathscr{F} \text{ where:} \\ &t_s \in \mathscr{T} \quad t_d \in \mathscr{T} \\ &c = [\text{throughput, max\_delay, max\_error\_rate}] \in \mathbb{R}^3 \end{aligned} \tag{2}$$

$\mathscr{F}$ is a multiset of data flows. Source and destination tasks are represented by t_s and t_d, respectively. The vector named $f.c$ contains the communication requirements: throughput is the amount of transmitted information in the time unit; the maximum permitted delay is the maximum time to deliver data to destination, the maximum permitted error rate is the maximum number of errors tolerated by the destination. As instance, in a file transfer application no errors are permitted while in multimedia applications this requirement could be relaxed.

C. Nodes

A node can be seen as a container of tasks. From the point of view of network synthesis, the focus is on the resources made available by the node to host a number of tasks.

Formally, the node n is a tuple defined as follows:

$$\begin{aligned} &n = [t, c, p, k, \gamma, m] \in \mathscr{N} \text{ where:} \\ &t \subseteq \mathscr{T} \quad c \in \mathbb{R}^n \quad p \in \mathbb{R} \quad k \in \mathbb{R} \quad \gamma \in \mathbb{R}^n \quad m \in \mathbb{B} \end{aligned} \tag{3}$$

$\mathscr{N}$ is a multiset of nodes. The multiset named $n.t$ contains the tasks associated to the node n. The vector named $n.c$ represents node's capabilities, i.e., the resources available on the node and its components have the same type as those of $t.c$. The power budget of the node is denoted by $n.p$. The economic cost of the node is denoted by $n.k$ and it could be estimated by referring to an actual platform.

The vector of coefficients named $n.\gamma$ is used to calculate the contribution to node's power consumption of each task assigned to the node; a scalar product connects node's coefficients $n.\gamma$ to task's resource requirements $t.c$ as follows:

$$\begin{aligned} P\colon \mathbb{R}^n \times \mathbb{R}^n &\to \mathbb{R} \\ (t.c, n.\gamma) &\mapsto n.\gamma \cdot t.c^T \end{aligned} \tag{4}$$

The component named $n.m$ is a boolean attribute indicating if the node can be mobile.

D. Abstract Channels

An AC interconnects two or more nodes as described at the beginning of Section IV.

Formally, the AC a is a tuple characterized as follows:

$$\begin{aligned} &a = [n, c, p, k, d, w] \in \mathscr{A} \text{ where:} \\ &n \subseteq \mathscr{N} \quad p \in \mathbb{R} \quad k \in \mathbb{R} \quad d \in \mathbb{R} \quad w \in \mathbb{B} \\ &c = [\text{max\_throughput, delay, error\_rate}] \in \mathbb{R}^3 \end{aligned} \tag{5}$$

$\mathscr{A}$ is a multiset of abstract channels. The multiset $a.n$ is the multiset of nodes that communicate between them by using the given AC. The vector $a.c$ is a vector of capabilities, i.e., it represents the communication resources of the given AC, i.e., the maximum transmission throughput, the transmission delay, and the error rate. It is worth noting that $a.c$ has the components of the same type of $f.c$, but the former represents the communication resources provided by the AC while the latter represents the communication requirements needed by the data flow and the involved tasks.

The power consumption of an a is denoted by $a.p$. The economic cost is denoted by $a.k$ which can be estimated by referring to an actual platform. Since in the proposed model an AC may includes intermediate systems, their economic cost shall be considered in the evaluation of $a.k$.

The attribute named $a.d$ is the maximum distance between nodes which are connected by the given abstract channel. The notion of distance is quite generic at this point and it could represent the length of a wire. The designer has the responsibility to complete the semantic value of this attribute according to design goals.

The field named $a.w$ is a boolean attribute indicating if the AC is wireless. If at least one node binded with the AC is mobile, then the AC shall be wireless.

E. Zones

When designing a network infrastructure, the relationship with the environment is important. For example, regarding a wireless sensor network for environmental monitoring, the placement of the sensor nodes depends on the spatial behavior of the data to be monitored.

The proposed approach to capture this relationship in the formal model is based on both the partitioning of the space into *zones* which contain nodes and the notion of contiguity between zones. Each zone is characterized by an environmental attribute, e.g., a temperature value in case of a monitoring application. The contiguity between zones poses constraints on the reachability of the corresponding nodes. In this way, the creation of network topologies can be controlled during network synthesis.

Formally, the zone z is a tuple characterized as follows:

$$\begin{aligned} &z = [n, s, e] \in \mathscr{Z} \text{ where:} \\ &n \subseteq \mathscr{N} \quad s \in \mathbb{R}^n \quad e \in \mathbb{R}^m \end{aligned} \tag{6}$$

The attribute named $z.n$ is the multiset of nodes placed in the given zone. The spatial features of the zone (e.g., its extension) are represented by the attribute named $z.s$. The environmental information (e.g., the value of temperature) is described by the attribute named $z.e$. The designer has the responsibility to complete the semantic value of $z.s$ and $z.e$ according to design goals.

The zones are related by the notion of contiguity defined as follows:

Definition 3. *Two zones are contiguous if nodes belonging to them can communicate each other.*

The contiguity is characterized as follows:

$$\begin{aligned} &c = [z_1, z_2, d] \in \mathscr{C} \text{ where:} \\ &z_1 \in \mathscr{Z} \quad z_2 \in \mathscr{Z} \quad d \in \mathbb{R} \end{aligned} \tag{7}$$

The components z_1 and z_2 represents the zones involved in the relationship. The attribute named $c.d$ is the distance between the zones. It must be of the same type as $a.d$ since it represents the minimum value of $a.d$ that an abstract channel shall have to connect two nodes placed in the corresponding zones. This implies that two nodes placed in the same zone are always able to communicate.

F. Graph representation

The entities described above can be connected by using three graphs as follows:

$$FG = (\mathscr{T}, \mathscr{F}) \quad CG = (\mathscr{N}, \mathscr{A}, E) \quad ZG = (\mathscr{Z}, \mathscr{C}) \tag{8}$$

The Flow Graph (FG) is the directed graph in which the tasks are the vertices, and the data flows are the edges.

The Channel Graph (CG) is the directed bipartite graph, in which the vertices are the nodes and the abstract channels while E is the set of edges defined as follows:

$$E = \{[n, a] \mid a \in \mathscr{A} \wedge n \in a.n\}.$$

The representation as bipartite graph is useful to represent channels shared by more than two nodes.

The Zone Graph (ZG) is a non-directed graph in which the vertices are the zones, and the edges are the contiguity relationships between them.

G. Relationships between entities

The proposed entities can be used by the designer to specify the application requirements which can be expressed through formal relationships between their attributes.

For instance, to state that the tasks associated to a node shall not require more resources than the ones available on that node, and that they shall respect the total available power limit, the formal constraints can be written as follows.

$$\sum_{t \in n.t} t.c \leqslant n.c \qquad \sum_{t \in n.t} P(t, n) \leqslant n.p \tag{9}$$

Where P is taken from Equation 4.

Also spatial relationships can be easily expressed. For instance, in a home monitoring application let us assume that the designer created several zones for each room and used the attribute $z.s$ to record the room identifier; the following Equation specifies that there must be at least one node for each room.

$$Z_i = \{z \mid z.s = i\} \qquad \forall i \bigcup_{z \in Z_i} z.n \neq \emptyset \tag{10}$$

The constraint that a node shall be mobile if it contains at least one mobile task, and that the attached abstract channels shall be wireless, can be modeled as follows:

$$\forall n \in \mathscr{N}(\bigvee_{t \in n.t} t.m) \Rightarrow n.m \quad \forall a \in \mathscr{A}(\bigvee_{n \in a.n} n.m) \Rightarrow a.w \tag{11}$$

In Equation 11 the operator $\bigvee$ is intended as the logical or.

From a mathematical point of view, the ability to express relationships and formulae between entities is useful to describe constraints and optimization metrics.

V. Network synthesis

In the previous Section all the network-related variables have been formalized. What it is left to be provided is:

- a formulation of the synthesis process;
- a methodology to find the solution of the synthesis problem.

These issues are briefly introduced in the following Subsections, and they will be addressed in future works.

A. Problem formulation

All the variables expressed in the proposed framework can be either free or bounded. Thus, the network synthesis problem can be defined as follows:

Definition 4. *Given a formal description of a network, a set of constraints and an optimization metric, the network synthesis is a process whose objective is to find the optimal solution, i.e. to find optimal values for the free variables.*

The optimization metric can be very complex, taking into account many parameters, e.g., power, and economic cost. Actually, the parameters have not the same importance and the designer has the responsibility to sort them according to the priority of the design goal. For instance, in a low-budget system the economic cost could be the most important optimization parameter followed by power consumption.

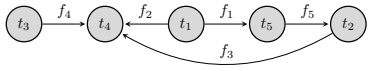

Fig. 4. Tasks and data flows for the given application.

B. How to find solutions

The optimal solution of the problem can be found analytically but this option is usually unfeasible due to the high number of parameters to be set and the search range of their values. Furthermore, it is known that similar problems are NP-hard even in specific fields as NoC [6].

For this reason, there is a lot of literature focused on alternative strategies, which can be divided into two classes:

- algorithms which can theoretically find the optimal solution provided that a long amount of time is spent;
- algorithms which use heuristics to find good solutions in a short amount of time without any guarantee to get the optimal solution.

In the first class there are techniques like the Simulated Annealing while in the second class there are techniques like graph decomposition. Also simulation plays an important role in such design-space exploration since it allows to refine the analytical model thus reducing the search space.

A good starting point for future works could be the extension of the heuristics proposed for NoC's.

VI. Case study

A. Introduction

The proposed modeling methodology has been applied to a case study consisting in the temperature control application described at the beginning of Section I and depicted in Figure 1. Temperature is controlled inside a building composed of floors and rooms. For simplicity's sake, in this example we consider a single floor with five rooms. Furthermore, to simplify the description of problem data, attribute values are reported without dimension type (e.g., bit, second, meter).

The objective of this case study is just to clarify the proposed formal model, and not to provide an example of a complete workflow.

B. Tasks, data flows and zones

We assume as starting point that a set of *tasks* $\mathscr{T} = \{t_1, t_2, t_3, t_4, t_5\}$ can be extracted from a platform-independent description of the application. Task t_1 is the system initialization, t_2 is the centralized temperature control, t_3 is the user temperature control, t_4 is the air-conditioner actuation, and task t_5 is the temperature sensing.

Tasks exchange information according to the set of *data flows* $\mathscr{F} = \{f_1, f_2, f_3, f_4, f_5\}$ as shown in Figure 4.

Each task has an attribute $c \in \mathbb{R}^2$ which represents CPU and memory usage, and a mobility attribute $m \in \mathbb{B}$; their values are:

$$t_1 = [[6,4],0] \qquad t_2 = [[3,5],0] \qquad t_3 = [[1,1],1]$$
$$t_4 = [[1,2],0] \qquad t_5 = [[1,2],0]$$

Similarly, each data flow has an attribute $c \in \mathbb{R}^3$ which represents communication requirements (throughput, maximum allowed delay, maximum allowed error rate); the attribute values are:

$$f_1 = [t_1, t_5, [1,3,0.1]] \qquad f_2 = [t_1, t_4, [1,3,0.1]]$$
$$f_3 = [t_2, t_4, [5,2,0.1]] \qquad f_4 = [t_3, t_4, [3,1,0.3]]$$
$$f_5 = [t_5, t_2, [5,2,0.1]]$$

A technological library for network nodes and channels is also available as input and it is reported in Table I. All the types of node are listed with the corresponding capability c, coefficient vector γ (see Section IV-C), and price k. All the types of channels are listed with the corresponding capability c, distance d, wireless attribute w (see Section IV-D), and price k.

	Name	c	γ	d	w	k
Node	A	$[2,4]$	$[0.8,1.1]$	—	1	25
	B	$[10,10]$	$[0.6,0.9]$	—	0	100
	C	$[2,4]$	$[1.2,0.8]$	—	1	80
	D	$[1,3]$	$[1.5,1.2]$	—	1	15
A.C.	X	$[100,0.2,0]$	—	100	0	5
	Y	$[54,2,0.1]$	—	30	1	20
	Z	$[12,1,0.3]$	—	10	1	10

TABLE I
Technological library for nodes and abstract channels.

The floor in which the application will be deployed has been partitioned into a set of *zones* $\mathscr{Z} = \{z_1, z_2, z_3, z_4, z_5\}$ with the corresponding set of contiguity relationships $\mathscr{C} = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$ which records the distance between zones as follows:

$$c_1 = [z_1, z_2, 3] \quad c_2 = [z_1, z_3, 6] \quad c_3 = [z_1, z_5, 3] \quad c_4 = [z_2, z_3, 5]$$
$$c_5 = [z_2, z_5, 5] \quad c_6 = [z_3, z_4, 3] \quad c_7 = [z_3, z_5, 3] \quad c_8 = [z_4, z_5, 3]$$

Finally, some constraints are given:

1) one instance of t_4 should be placed in each zone;
2) one instance of t_5 should be placed in each zone;
3) minimum five instances of t_3 must be present in the environment;
4) maximum three instances of t_3 can be present in the same zone, simultaneously;
5) maximum one instance of t_3 can be assigned to a single node;
6) an instance of t_3 is able to communicate only with other tasks in the same zone.

The goal of the design problem is to minimize the total cost of the deployed application.

Considering tasks, data flows, zones and the number of deployed nodes $|\mathscr{N}|$ and abstract channels $|\mathscr{A}|$, then the total number of possible solutions would be proportional to $|\mathscr{N}|^{|\mathscr{T}|} \cdot |\mathscr{A}|^{|\mathscr{F}|} \cdot |\mathscr{Z}|^{|\mathscr{N}|}$.

This expression takes into account all the possible assignments of tasks to nodes, flows to channels, and nodes to zones, respectively. The number of solutions could be very large but constraints could considerably decrease it. In the rest of the Section some simple heuristics will be used to find the solution while efficient search techniques will be investigated in future work.

C. Task assignment

Considering the nature of the given constraints the first phase of design-space exploration could be the assignment of tasks to zones.

An instance of t_4 and an instance of t_5 are placed in each zone. Instances of t_1 and t_2 are not subjects to position constraints, so that they can be placed in any zone. We decide to choose one of the most connected zones, i.e., z_3 and z_5. We can randomly choose between z_3 and z_5 (or even place one task per zone) or we can split the solutions' space and proceed with two or more possible solutions; for the sake of simplicity we place t_1 and t_2 in z_3. Instances of t_3 are mobile (for clarity's sake in this example mobile task and node labels are denoted with a bar) so we can randomly place them among all the five zones.

D. Node assignment

Once tasks are placed, they should be assigned to nodes. Node assignment could be done by grouping tasks in the same zone into a single node (provided that node's capacity can support them). Exceptions to this method arise when a task should be mobile and others don't, or the cost of a couple of nodes is lower then the cost a single node. The process of choosing which kind of node should be picked up from the technological library could follow the rule of the *best fitting capacity* which leads to the following configuration:

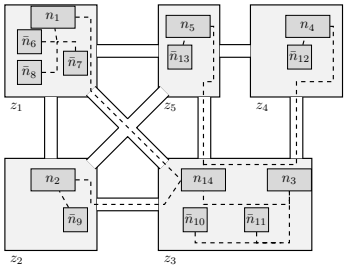

Fig. 5. Assignment of nodes and abstract channels to zones.

Node	Type	Tasks
$n_{1\text{–}5}$	A	t_4, t_5
$n_{6\text{–}13}$	D	t_3
n_{14}	B	t_1, t_2

A.C.	Type	Nodes
a_1	Z	n_1, n_6, n_7, n_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	$n_{14}, n_1, n_2, n_3, n_4, n_5$

TABLE II
A FEASIBLE SOLUTION FOR THE DESIGN PROBLEM.

$$
\begin{aligned}
z_1 &= \{n_1[t_4, t_5], \bar{n}_6[\bar{t}_3], \bar{n}_7[\bar{t}_3], \bar{n}_8[\bar{t}_3]\} \\
z_2 &= \{n_2[t_4, t_5], \bar{n}_9[\bar{t}_3]\} \\
z_3 &= \{n_3[t_4, t_5], n_{14}[t_1, t_2], \bar{n}_{10}[\bar{t}_3], \bar{n}_{11}[\bar{t}_3]\} \\
z_4 &= \{n_4[t_4, t_5], \bar{n}_{12}[\bar{t}_3]\} \\
z_5 &= \{n_5[t_4, t_5], \bar{n}_{13}[\bar{t}_3]\}
\end{aligned}
$$

where nodes $n_1, \ldots, n_5$ are implemented by type-A nodes, nodes $n_6, \ldots, n_{13}$ by type-D nodes and node n_{14} by type-B nodes.

Table II summarizes tasks and nodes assignment and the choice of node types.

E. Assignment of abstract channels

Given the previous assignment of tasks to nodes and considering the data flows between tasks, six channels are needed to connect nodes as follows:

$$
\begin{aligned}
a_1 &= \{n_1, \bar{n}_6, \bar{n}_7, \bar{n}_8\} & a_2 &= \{n_2, \bar{n}_9\} \\
a_3 &= \{n_3, \bar{n}_{10}, \bar{n}_{11}\} & a_4 &= \{n_4, \bar{n}_{12}\} \\
a_5 &= \{n_5, \bar{n}_{13}\} & a_6 &= \{n_{14}, n_1, n_2, n_3, n_4, n_5\}
\end{aligned}
$$

To choose the type of channels from the technological library, the following aspects must be taken into account:

- channel capabilities (in terms of maximum throughput, delay and error rate) must satisfy the requirements of data flows;
- channel distance must satisfy the contiguity between zones;
- node mobility, i.e., mobile nodes must be connected through wireless channels.

According to these issues, channels $a_1, \ldots, a_5$ could be instances of Z type which satisfies requirements concerning throughput, delay, error rate, and mobility. For instance, the maximum throughput in a_1 is 9 when the three instances of task t_3 send data to node n_1; this value is lower than the maximum allowed throughput of the abstract channel Z (i.e., 12).

The abstract channel a_6 can be an instance of X type which satisfies requirements on throughput, delay, and error rate.

The chosen abstract channels and the connected nodes are summarized in Table II. This leads to the complete network specification, and the economic cost associated to the found solution is:

$$K = 5 \cdot A.k + 8 \cdot D.k + B.k + X.k + 5 \cdot Z.k = 400 \qquad (12)$$

VII. CONCLUSIONS AND FUTURE WORK

A extension of the traditional design flow has been proposed for distributed applications based on networked embedded systems. Additional design phases have been introduced to model the application from a communication perspective and to provide the synthesis of the communication infrastructure. The former point is the focus of this work, i.e., the creation of a formal framework to model tasks, network nodes, data flows, channels, and interactions with the environment. Entities and relationships have been introduced through a mathematical approach which simplifies the specification of requirements and constraints. Each element has been described with examples of real-world network applications and, finally, the framework has been applied to a case study. This work is a first contribution to a global strategy for network synthesis and thus some research topics are still open:

- Extraction of task's requirements from the platform-independent model.
- Presence of abstract channels and tasks at different ISO/OSI levels.
- Fast and efficient methodologies to perform design-space exploration of network solutions, e,g, through simulation.

REFERENCES

[1] N. Bombieri, F. Fummi, D. Quaglia, "System/network design space exploration based on tlm for networked embedded systems," *ACM Transactions on Embedded Computer Systems*, to appear 2010.

[2] A. Bakshi and V. K. Prasanna, "Algorithm design and synthesis for wireless sensor networks," in *International Conference on Parallel Processing (ICPP 2004)*. IEEE Society, Aug. 2004, pp. 15–18 (1).

[3] A. Bonivento, L. P. Carloni, and A. Sangiovanni-Vincentelli, "Platform-based design of wireless sensor networks for industrial applications," in *Proc. of Design, Automation and Test in Europe (DATE'06)*. Munich: IEEE Society, Mar. 2006, pp. 1–6 (1).

[4] L. Mottola, A. Pathak, A. Bakshi, V. K. Prasanna, and G. P. Picco, "Enabling scope-based interactions in sensor network macroprogramming," in *Internatonal Conference on Mobile Adhoc and Sensor Systems*. IEEE, Oct. 2008, pp. 1–9.

[5] T. Bjerregaard and S. Mahadevan, *A Survey of Research and Practices of Network-on-Chip*.

[6] A. Agarwal, C. Iskander, H. Multisystems, and R. Shankar, "Survey of network on chip (noc) architectures & contributions," in *Journal of Engineering, Computing and Architecture*, 2009.

[7] E. Rijpkema, K. Goossens, A. Radulescu, and J. Dielissen, "Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip," in *IEEE Computers and Digital Techniques*, 2003.

[8] R. Hunt, "A review of quality of service mechanisms in IP-based networks – integrated and differentiated services, multi-layer switching, MPLS and traffic engineering," *Computer Communications*, vol. 25, no. 1, pp. 100–108, January 2002.

[9] U. Y. Ogras and R. Marculescu, "Energy- and performance-driven noc communication architecture synthesis using a decomposition approach," in *IEEE Conference and Exhibition on Design, Automation and Test in Europe*, 2005.

[10] R. Chae-Eun, J. Han-You, and H. Soonhoi, "Many-to-many core-switch mapping in 2-d mesh noc architectures," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 2004.

[11] Y. Zhang, K. Takahashi, N. Shiratori, and S. Noguchi, "An interactive protocol synthesis algorithm using a global state transition graph," *IEEE Transactions on Software Engineering*, vol. 14, pp. 394–404, 1988.

[12] A. Khoumsi, G. v. Bochmann, and R. Dssouli, "Protocol synthesis for real-time applications," in *FORTE XII / PSTV XIX '99: Proceedings of the IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX)*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V., 1999, pp. 417–433.

[13] H. Yamaguchi, K. Okano, T. Higashino, and K. Taniguchi, "Protocol synthesis from time petri net based service specification," *Parallel and Distributed Systems, International Conference on*, vol. 0, p. 236, 1997.

[14] R. L. Probert and K. Saleh, "Synthesis of communication protocols: Survey and assessment," *IEEE Trans. Comput.*, vol. 40, no. 4, pp. 468–476, 1991.

[15] P. V. Eijk and J. Schot, "An exercise in protocol synthesis," in *Formal Description Techniques IV. North-Holland*, 1991, pp. 117–131.