Università degli Studi di Verona

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Specialistica in Informatica

Tesi di Laurea Specialistica

CASSE

Communication-Aware Specification and Synthesis Environment

Candidato:
Giovanni Lovato
Matricola VR077231

Relatore:
Davide Quaglia

Correlatore:
Francesco Stefanni

Anno Accademico 2009-2010

Revision July 16, 2010
Typesetting by LaTeX 2_ε
Printed in July, 2010

Ai miei genitori,
che dandomi motivazione e sostegno
mi hanno permesso di arrivare fino a qui.

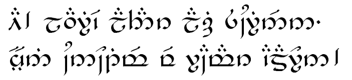
To my parents,
the ones who have given me the support
and the motivation needed to get this far.

Contents

List of Figures

List of Tables

Prefazione

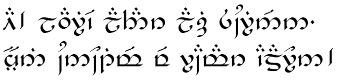


Ah! Simili ad oro cadono le foglie
al vento, lunghi innumerevoli anni
come le ali degli alberi!

J. R. R. Tolkien
Il Signore degli Anelli

Il computer-aided design è stato tradizionalmente utilizzato per calcolatori e sistemi embedded, ma non per l'infrastruttura di comunicazione tra di loro. Questa tesi contribuisce a colmare questa lacuna, proponendo di utilizzare un linguaggio matematico per modellare un'applicazione distribuita in termini di attività (*tasks*), nodi (*nodes*) e le interazioni con l'ambiente. Le attività sono descritte in termini di requisiti di calcolo e di comunicazione, anche in relazione con lo stato dell'arte dei linguaggi per la specifica di sistemi. Entità e relazioni sono introdotte per associare le attività, i flussi di dati e le caratteristiche ambientali, i nodi della rete, i canali tra loro e i protocolli di comunicazione. Gli attributi e i vincoli risultanti possono essere utilizzati durante un approfondimento in fase di *design-space exploration* per sintetizzare automaticamente un'adeguata infrastruttura di comunicazione. L'approccio può essere applicato a importanti applicazioni, ad esempio, quelle basate su reti di sensori e reti peer-to-peer. Casi di studio relativi al calcolo distribuito e building automation sono riportati al fine di dimostrare le potenzialità del modello.

Preface



> Ah! Like gold fall the leaves in the wind, long years numberless as the wings of trees!
>
> J. R. R. Tolkien
> *The Lord of the Rings*

Computer-aided design has been traditionally applied to computers and embedded systems but not to the communication infrastructure among them. This thesis contributes to fill this gap by proposing the use of 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 automatically synthesize 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. Case studies related to distributed computing and building automation are also reported to demonstrate the potentiality of the model.

Chapter 1

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 1.1, reporting the temperature control application of a skyscraper. In each room, there is at least one sensor (in Figure denoted by S) which keeps track of the local temperature variations. Collected data are sent to controllers (denoted by C), which send commands to actuators (denoted by 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;
- 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;

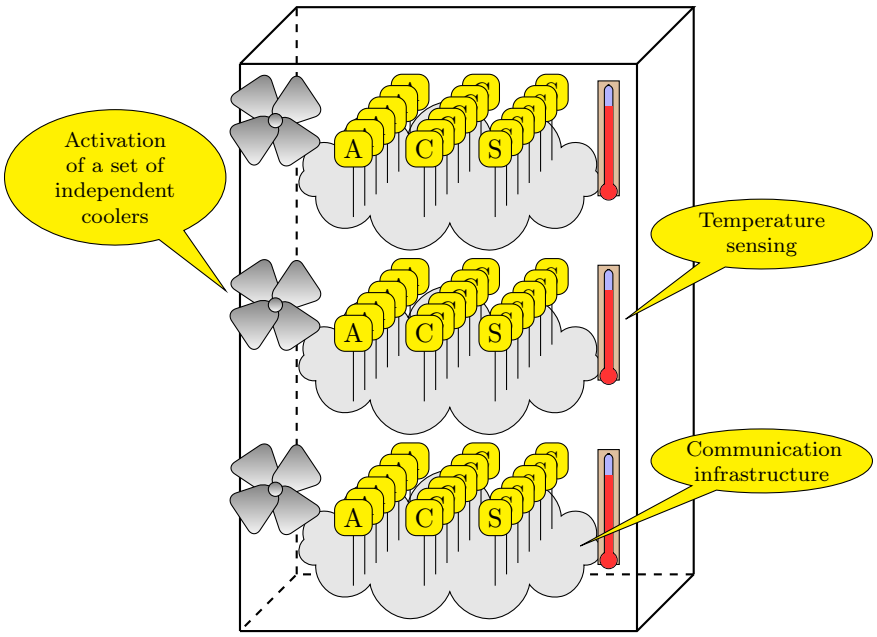


Figure 1.1: Example scenario of an application for building automation.

- communication protocols can be either reliable or un-reliable, best effort or with priority;
- characteristics of the environment in which sensors, controllers and actuators are placed affect the communications between them, the sensing performance and the control policy.

Instead of traditional point-to-point applications, 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; questions 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 (with the corresponding communication requirements) to network nodes, which channel types and protocols should be used among them to satisfy such requirements?
- How the ambiental features (e.g., temperature, humidity, altitude) may affect 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 (see Chapter 2):

1. focused on hardware/software design and not considering the network as part of the design space;
2. focused on network-related problems, but without a general design methodology considering the whole application.

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

The main contributions of this thesis are:

- a formal model that allows the *specification* of the communication aspects of the applications;
- a classification of *synthesis* methodologies which considers the communication infrastructure as a dimension of the design space;
- case studies to show the potentiality of the approach.

The proposed work is named Communication-Aware Specification and Synthesis Environment (CASSE) and is a first step towards the *specification* of a network environment through a formal model and the subsequent *synthesis* of the communication infrastructure (i.e., the automatic choice of channel type, protocols

and intermediate systems to support a distributed application). A concurrent paper [2] where the modeling language of CASSE is proposed was accepted for presentation at ECSI Forum on specification and Design Languages (FDL) 2010.

The rest of the thesis is organized as follows. Related work is reported in Chapter 2. The global design flow is introduced in Chapter 3. The proposed specification of a formal network model is reported in Chapter 4. A first discussion of the synthesis methodology is provided in Chapter 5. Case studies are described in Chapter 6. Conclusions and future works are reported in Chapter 7.

Chapter 2

Related work

The synthesis of distributed systems has been addressed by many research works, in different fields, such as Wireless Sensor Network (WSN). A virtual architecture has been proposed in order to simplify the synthesis of algorithms for WSN [3]. Some network information, like the topology and high-level functionality, are used to configure the virtual architecture. However CASSE is mainly focused on the application part of the system rather than on communication aspects. Platform-Based Design (PBD) has been adopted to project WSNs for industrial control [4]. 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 [5]. 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 Network-on-Chip (NoC) offers an example of system-network integrated approach which is close to the one proposed in this thesis. NoCs are embedded systems and, thus, they are designed with the traditional specification-refinement-synthesis flow; nevertheless they have also a communication infrastructure which is 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 [6]; 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 [7]. Routing protocols are a design issue, too. In NoCs routing can be deterministic or adaptive according to the current state of the network and to appropriate Quality of Service (QoS) policies [7]. Regarding

the latter, best effort and guaranteed throughput have been proposed [8]; they are very similar to those defined for TCP/IP [9]. 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 [7, 10]. A Mixed Integer Linear Programming (MILP) formulation of the problem has been proposed [11]. 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 (which is proportional to power consumption and communication delay), and minimization of the maximum link bandwidth (which minimizes the most-congested link-queuing time and maximizes the throughput).

Synthesis of communication protocols is another research topic related to the focus of CASSE. Automatic tools have been adopted to derive the actual implementation of protocols specified through Finite State Machine (FSM) [12, 13], Petri Nets [14], trace models [15], and languages like LOTOS [16].

All these works are focused on particular problems and do not provide any general model or a global design flow. This thesis tries to fill such a lack.

Chapter 3

Design flow of distributed systems

In the traditional system design flow a platform-independent model of the system is created starting from application requirements, both functional and non-functional. This is a pure specification model of concurrent resources (e.g., tasks) and their interaction via a communication medium (e.g., channels). Behavior in this specification is usually expressed through languages like Unified Modeling Language (UML) and C/C++ or through the use of tools like Matlab/Simulink/Stateflow (please, refer to Chapter 2 for a survey).

This specification, together with a description of the target platform, is the subject of a design-space exploration which maps tasks onto hardware and software components for the target platform. The result is platform-dependent description of the system in which hardware blocks are described by Hardware Description Language (HDL) languages like SystemC and VHSIC Hardware Description Language (VHDL) while software blocks are implemented in C/C++ and compiled for the target CPU.

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: Figure 3.1 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.

The new design path is quite symmetric with respect to the traditional path since it applies the same concepts to the communication aspects of the system. Starting from the platform-independent model, a formal *network specification*

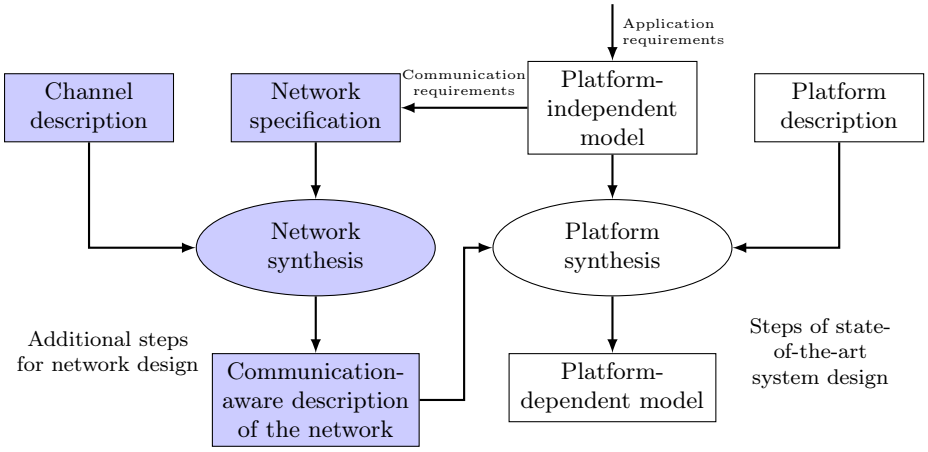


Figure 3.1: The proposed design flow for distributed systems: new steps for network design are added to the state-of-the-art system design flow.

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 them, e.g., their throughput and permitted latency. The output of this phase will be a parametric model with valued and un-valued parameters and constraints among them.

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 parametric model obtained in the previous phase. Actual channels and protocols are chosen according to their affinity to the optimal solutions. This last phase is *network synthesis*. The final result is a *communication-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 which may provide additional intermediate nodes to improve communication performance.

The communication-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 and the presence of new tasks to handle network protocols. 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 3.1.

3.1 High-level system specification languages

The elements of the formal network model depicted in Figure 3.1 can be extracted from a high-level description of the application created by well-known

languages as those reported in this Section.

Modeling and Analysis of Real-Time and Embedded systems (MARTE) [17] is a profile of UML [18] designed to allow an easy specification of real-time and embedded systems. It provides some sub-profiles, like Non-Functional Requirements (NFR), which allows to describe the "fitness" of the system behavior (e.g., performance, memory usage, power consumption). The Software Resource Modeling (SRM) and the Hardware Resource Modeling (HRM) profiles are derived from NFR, and they address the modeling of resources.

The System Modeling Language (SysML) [19] is an extension of UML to provide a general-purpose modeling language for systems engineering applications.

Mathworks has developed Simulink [20] and Stateflow [21] to model and simulate dynamic and embedded systems. Simulink models an application as the inter-connection of dynamic blocks (e.g., a filter) or digital blocks. Stateflow describes applications as finite-state machines. The tools can also be combined to represent hybrid automata.

The Ptolemy Project [22] is born to model concurrent real-time and embedded systems. One of its main advantages is the support for heterogeneous mixtures of computation models. Ptolemy supports simulation by using the actor-oriented design; actors are software components executed concurrently and able to communicate by sending messages throw interconnected ports. Ptolemy also supports communication modeling through Khan process networks [23].

SystemC [24], initially born as hardware description language, has been extended with the Transaction Level Modeling (TLM) [25], to describe hardware/software systems. SystemC and TLM allow to describe tasks as nested components with event-driven or clock-driven processes. Communications between tasks can be described by using standard protocols and payloads which simplify the specification of their requirements.

SpecC [26] is an extension of C language to be used as system-level design language, like SystemC/TLM. The SpecC methodology is a top-down design flow, with four well-defined levels of abstraction. It allows different ways to describe the target control (sequential, FSM, parallel and behavioral). One key concept of SpecC is the clear separation of the communication and computation model which can be useful to specify computation and communication aspects of tasks.

Metropolis [27] is a framework based on the idea of meta-model to support various communication and computation semantics in a uniform way. This approach implements the abstract semantics of process networks and uses the concepts advocated by the PBD methodology, i.e., functionality and architecture across models of computation and abstraction levels, and the mapping relationships between them.

Chapter 4

Network specification

This Chapter defines entities and relationships which constitute the network specification step in the design flow of CASSE.

Figure 4.1 on the following page 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. According to the concepts described in Chapter 3 both partitions will be described with entities and relationships in this Chapter.

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* unifies the physical channel of the first example and the transport channel of the second one. Abstract channels are defined as follows.

Definition 4.1. Referring to the ISO/OSI model and assuming that the functionality to be designed is at level N, the abstract channel contains the physical channel, and all the protocol entities up to level $N-1$.

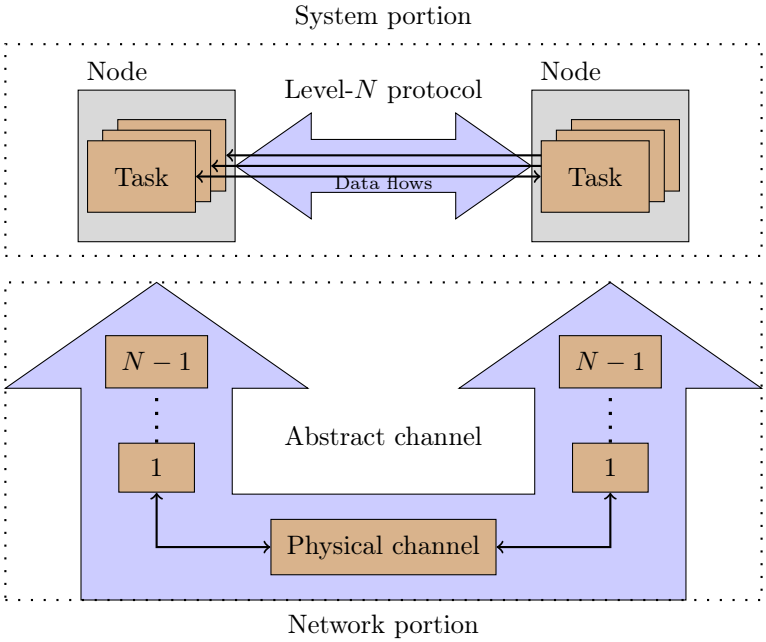


Figure 4.1: System/network partitioning for network modeling and synthesis.

The abstract channel connects network nodes whose tasks are implemented using level-N protocols. The abstract channel is the subject of network synthesis while network nodes and tasks fall in the domain of traditional system design. According to the ISO/OSI reference model, a pair of level-N entities may communicate either directly or through a chain of intermediate systems working at lower level. In the same way, a level-N abstract channel may include additional intermediate systems which shall be considered both in the evaluation of its cost and during network synthesis.

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 Chapter, entities will be described in details together with relationships between them.

4.1 Tasks

A task represents a basic functionality of the application; it takes some data as input and provides some output. From the point of view of network synthesis the focus is not on the description of the functionality in itself and on its hardware/software implementation but rather on its computational and mobility

requirements which affect the assignment of tasks to network nodes.

A task t is defined as follows,

$$\begin{aligned} t &= [c, m] \in \mathscr{T}, \text{where} \\ c &\in \mathbb{R}^n. \\ m &\in \mathbb{B}. \end{aligned} \tag{4.1}$$

$\mathscr{T}$ is the multiset of tasks (the choice to use a multiset is due to the fact that more instances of the same task could exist). 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.

4.2 Data flows

A data flow represents communication between two tasks; output from the source task is delivered as input for the destination task. 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,

$$\begin{aligned} f &= [t_s, t_d, c] \in \mathscr{F}, \text{where} \\ t_s &\in \mathscr{T}, \\ t_d &\in \mathscr{T}, \\ c &\in \mathbb{R}^n. \end{aligned} \tag{4.2}$$

$\mathscr{F}$ is the multiset of data flows (the choice to use a multiset is due to the fact that more instances of the same data flow could exist). Source and destination tasks are represented by t_s and t_d, respectively. The vector named $f.c$ contains the communication requirements, e.g., data throughput, maximum permitted delay, maximum permitted error rate. Throughput is the amount of transmitted information in the time unit; the maximum permitted delay is the maximum time to deliver data to destination and it can be important for real-time applications; the maximum permitted error rate is the maximum number of errors tolerated by the destination. For instance, in a file transfer application no er-

rors are permitted while in multimedia applications this requirement could be relaxed.

4.3 Nodes

A node can be seen as a container of tasks. At the end of the application design flow, nodes will become hardware entities with CPU and network interfaces and tasks will be implemented either as hardware components or as software processes. 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, a node n is a tuple defined as follows,

$$\begin{aligned} & n = [T, c, p, e, m, \pi, \kappa] \in \mathscr{N}, \text{where} \\ & T \subseteq \mathscr{T}, \\ & c \in \mathbb{R}^n, \\ & e \in \mathbb{R}^m, \\ & m \in \mathbb{B}, \\ & p \in \mathbb{R}, \\ & \pi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \\ & \kappa \in \mathbb{R}. \end{aligned} \tag{4.3}$$

$\mathscr{N}$ is the multiset of nodes (the choice to use a multiset is due to the fact that more instances of the same node could exist). The multiset named $n.T$ contains tasks assigned 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 attribute $n.e$ describes the environmental conditions supported by the node, e.g., maximum temperature, maximum humidity, altitude. The component named $n.m$ is a boolean attribute indicating if the node can be mobile. The power budget (i.e., the battery) of the node is denoted by $n.p$; it could be useful for mobile nodes, otherwise it can be 0 and an external power source could be expressed in $n.e$. The function π is used to calculate the node's power consumption based on assigned tasks and node's capabilities. The economic cost of the node is denoted by $n.\kappa$ and could be estimated by referring to an actual platform.

4.4 Abstract channels

An abstract channel interconnects two or more nodes as described in Definition 4.1 on page 11.

Formally, an abstract channel a is a tuple defined as follows,

$$\begin{aligned} a &= [N, c, d, w, \kappa] \in \mathscr{A}, \text{where} \\ N &\subseteq \mathscr{N}, \\ c &\in \mathbb{R}^n, \\ d &\in \mathbb{R}, \\ w &\in \mathbb{B}, \\ \kappa &\in \mathbb{R}. \end{aligned} \tag{4.4}$$

$\mathscr{A}$ is the multiset of abstract channels (the choice to use a multiset is due to the fact that more instances of the same abstract channel could exist). The multiset $a.N$ is the multiset of nodes that communicate between them by using the given abstract channel. The vector $a.c$ is a vector of capabilities, i.e., it represents the communication resources of the given abstract channel, e.g., the maximum transmission throughput, the transmission delay, 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 abstract channel while the latter represents the communication requirements needed by the data flow and the involved tasks. 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 range of a wireless channel, or the maximum number of hops which can be performed by a packet. 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 abstract channel is wireless. If at least one node binded with the abstract channel is mobile, then the abstract channel shall be wireless. The economic cost of the abstract channel is denoted by $a.\kappa$ and could be estimated by referring to an actual platform.

4.5 Zones

When designing a network infrastructure, the relationship with the environment is important. For instance, the placement of wireless access points in a building should take into account its subdivision into floors and rooms and the presence of obstacles. In another 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. In this way, the designer can relate the data in a given zone with the presence of nodes in that zone (e.g., at least one node per zone is required even if more redundant architecture may be designed).

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

$$\begin{aligned} & z = [n, e] \in \mathscr{Z} \text{ where:} \\ & n \subseteq \mathscr{N} \\ & e \in \mathbb{R}^m \end{aligned} \tag{4.5}$$

$\mathscr{Z}$ is the set of zones. The attribute named $z.N$ is the multiset of nodes placed in the given zone. The environmental informations (e.g., the value of temperature, humidity, altitude, spatial features) are described by the attribute named $z.e$. The designer has the responsibility to complete the semantic value of $z.e$ according to design goals and he should consider the relationships of this attribute with $n.e$ respectively; see Section 4.8 on the facing page for further informations.

4.6 Contiguities

Zones are related by the notion of contiguity and its main purpose is to describe the distances between them. Contiguities between zones set constraints on the reachability of the corresponding nodes. In this way, the creation of network topologies can be controlled during network synthesis.

A contiguity c is defined as follows,

$$\begin{aligned} c = [z_1, z_2, d] \in \mathscr{C}, \text{where} \\ z_1 \in \mathscr{Z}, \\ z_2 \in \mathscr{Z}, \\ d \in \mathbb{R}. \end{aligned} \tag{4.6}$$

$\mathscr{C}$ is the set of contiguities. Attributes z_1 and z_2 represent 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.

If two nodes are placed in non-contiguous zones, then they are unable to communicate directly but they shall use some intermediate systems. Therefore, the notion of contiguity is very useful to make explicit intermediate systems

during the step of network synthesis.

4.7 Graph representation

Entities described in this Chapter can be correlated using three graphs,

$$\begin{aligned} FG &= (\mathscr{T}, \mathscr{F}), \\ CG &= (\mathscr{N}, \mathscr{A}, E), \\ ZG &= (\mathscr{Z}, \mathscr{C}). \end{aligned} \tag{4.7}$$

The Flow Graph (FG) is a directed graph in which tasks are the vertices and data flows are the edges. The Channel Graph (CG) is a directed bipartite graph, in which nodes and abstract channels are the vertices 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 zones are the vertices and contiguities are the edges.

4.8 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 entities' attributes.

For instance, to state that tasks associated to a node shall not require more resources than the ones available on that node, a formal constraint can be expressed as follows,

$$\sum_{t \in n.T} t.c \leqslant n.c. \tag{4.8}$$

Similarly, a constraint on power consumption can be expressed as follows,

$$\sum_{t \in n.T} n.\pi(t.c, n.c) \leqslant n.p. \tag{4.9}$$

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 expressed

as follows,

$$\begin{aligned} \forall n \in \mathscr{N} \bigvee_{t \in n.T} t.m \Rightarrow n.m, \\ \forall a \in \mathscr{A} \bigvee_{n \in a.N} n.m \Rightarrow a.w, \end{aligned} \tag{4.10}$$

where the operator $\bigvee$ is intended as the logical OR.

The attribute $n.e$, which expresses the environmental conditions supported by the node, can be easily related to $z.e$, which expresses the environmental features of a zone. Formally, a node can be placed into a zone only if the following equation is satisfied,

$$z.e \leqslant \min_{n \in z.N} n.e \tag{4.11}$$

where min returns the vector of the minimum coefficients of all $n.e$.

Expressing constraints on minimum abstract channel range to connect nodes placed in two different zones can take advantage of the graph representation described on Section 4.7 on the previous page,

$$a.d \geqslant d(z_1, z_2), \tag{4.12}$$

where

$$d \colon \mathscr{Z} \times \mathscr{Z} \to \mathbb{R} \tag{4.13}$$

is a function which determines the cost of the shortest path connecting two zones on the graph $ZG = (\mathscr{Z}, \mathscr{C})$, while considering the attribute $c.d$ as the cost of each edge $c \in \mathscr{C}$.

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

Chapter 5

Network synthesis

In the previous Chapter all the network-related entities and relationships have been formalized. What 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 discussed in this Chapter.

5.1 Problem formulation

All the variables expressed in CASSE as entity attributes can be either free or bounded. Thus, the network synthesis process can be defined as follows.

Definition 5.1. 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 application tasks onto network nodes, their spatial displacement among zones and connections with abstract channels, assigning optimal values to free attributes.

The optimization metric can be very complex, taking into account many parameters, e.g., power and economic costs, number of nodes or computation time. 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.

Although CASSE is been designed for the maximum flexibility, letting the designer to choose the preferred variable definitions, sequence of steps and optimization heuristics, a general methodology classification and associated syn-

thesis processes are provided to demonstrate the potentialities of the proposed environment.

Within the design process of a distributed application and its best fitting network, one can look forward between the following three main design aspects.

Platform oriented. Definition of technological specifications of network hardware (nodes, channels) required to satisfy application requirements and environment conditions.

Environment oriented. Definition of the zone characteristics (spatial and environmental features, contiguities and distances) to reflect an optimal environment for the application.

Application oriented. Definition of an optimal task assignment to nodes, node distribution between zones and node assignment to abstract channels to get the best application performance.

5.2 Platform Oriented Problems

A Platform Oriented Problem (POP) will be typically provided with a formal description in CASSE where elements of $\mathscr{T}$, $\mathscr{F}$, $\mathscr{Z}$ and $\mathscr{C}$ are completely defined and elements of $\mathscr{N}$ and $\mathscr{A}$ must have at least $n.T$ and $a.N$ sets respectively populated while having all the other attributes potentially undefined; see Table 5.1 on page 24 for reference.

That is, the application is given as is and each zone has a number of nodes placed in and its spatial and environmental features are well defined; each node has tasks assigned to and belongs to an existing abstract channel; both nodes and abstract channels are undefined on their other attributes such as computation and communication capabilities, letting the synthesis process to define them for an optimal configuration.

5.2.1 Synthesis process for POPs

The formal model defined by CASSE permits a straightforward process to synthesize nodes' and abstract channels' specifications. Given sets $\mathscr{T}$, $\mathscr{F}$, $\mathscr{N}$, $\mathscr{A}$, $\mathscr{Z}$, $\mathscr{C}$, for each zone $z \in \mathscr{Z}$ and for each node $n \in z.N$ it's easy to calculate the minimum values for $n.c$, $n.e$, $n.p$ and the value of $n.m$,

$$n.c = \sum_{t \in n.T} t.c \tag{5.1}$$

$$n.e = z.e \tag{5.2}$$

$$n.p = \sum_{t \in n.T} n.\pi(t.c, n.c) \tag{5.3}$$

$$n.m = \bigvee_{t \in n.T} t.m \tag{5.4}$$

Similar equations could be used to calculate abstract channels' $a.c$ and $a.w$ attributes,

$$a.c = \sum_{f \in F_a} f.c \tag{5.5}$$

$$a.w = \bigvee_{n \in a.N} n.m \tag{5.6}$$

where, in Equation 5.5,

$$\begin{aligned} F_a = \{f \in \mathscr{F} \mid & \exists n_1 \in a.N \, \exists n_2 \in a.N, n_1 \neq n_2, \\ & \exists t_1 \in n_1.t \, \exists t_2 \in n_2.t, t_1 = f.t_s \wedge t_2 = f.t_d\} \end{aligned} \tag{5.7}$$

is the multiset of data flows between tasks that are assigned to different nodes, each one belonging to the abstract channel.

To calculate abstract channels' $a.d$ attribute it should be considered that abstract channels could contain nodes placed in different zones. So, it's convenient to use graph representations as formalized in Section 4.7 of Chapter 4 on page 17; that is,

$$a.d = \max_{z_1, z_2 \in Z_a} d(z_1, z_2) \tag{5.8}$$

where

$$Z_a = \{z \in \mathscr{Z} \mid \exists n \in a.N, n \in z.N\} \tag{5.9}$$

is the set of zones where is placed at least one node belonging to the abstract channel, and d is the same function defined in 4.13 on page 18.

5.3 Environment Oriented Problems

An Environment Oriented Problem (EOP) will be typically provided with a formal description in CASSE where elements of $\mathscr{T}$, $\mathscr{F}$, $\mathscr{N}$ and $\mathscr{A}$ are completely defined, elements of $\mathscr{Z}$ must have at least $z.N$ set populated while all other attributes can be potentially undefined; finally, $\mathscr{C} = \emptyset$ and should be populated. See Table 5.1 on page 24 for reference.

That is, the application is given as is and each zone has a number of nodes placed in; each node has tasks assigned to and belongs to an existing abstract channel; both nodes and abstract channels are completely defined on their attributes such as computation and communication capabilities; zones and con-

tiguities between them are instead undefined on their attributes, letting the synthesis process to model an optimal topology in accordance to the application requirements and devices capabilities.

5.3.1 Synthesis process for EOPs

The first step is to define the maximum value of $z.e$,

$$z.e = \min_{n \in z.N} n.e \tag{5.10}$$

where zone's environmental characteristic $z.e$ (e.g., temperature, humidity) should be acceptable for the most restricted node in its environmental features.

To populate $\mathscr{C}$, it can be used a definition like the following,

$$\begin{aligned} \mathscr{C} = \{c \mid & \forall z_1 \in \mathscr{Z}\, \forall z_2 \in \mathscr{Z}, z_1 \neq z_2, \exists a \in \mathscr{A}\, \exists n_1 \in \mathscr{N}\, \exists n_2 \in \mathscr{N}, \\ & (n_1 \in z_1.n \wedge n_2 \in z_2.n) \rightarrow (c.z_1 = z_1 \wedge c.z_2 = z_2 \wedge c.d = a.d)\}, \end{aligned} \tag{5.11}$$

that is, if there are nodes placed in different zones and they are connected by the same abstract channel, those zones must be contiguous and the maximum allowed distance between them is the maximum range of the abstract channel. It's worth noting that to avoid redundant and/or misleading informations, $\mathscr{C}$ should be purged of all contiguities which connects the same zones, keeping the one with minimum $c.d$; formally,

$$\begin{aligned} \mathscr{C} = \mathscr{C} \setminus \{c \in \mathscr{C} \mid & \exists c_0 \in \mathscr{C}, (c_0.z_1 = c.z_1 \vee c_0.z_1 = c.z_2) \\ & \wedge (c_0.z_2 = c.z_1 \vee c_0.z_2 = c.z_2) \wedge c_0.d \leqslant c.d\}. \end{aligned} \tag{5.12}$$

5.4 Application Oriented Problems

An Application Oriented Problem (AOP) will be typically provided with a formal description in CASSE where elements of all $\mathscr{T}$, $\mathscr{F}$, $\mathscr{N}$, $\mathscr{A}$, $\mathscr{Z}$ and $\mathscr{C}$ are at most defined, with the exception of sets $n.T$, $a.N$ and $z.N$; see Table 5.1 on page 24 for reference.

That is, the application is given as is, nodes and abstract channels are completely defined on their characteristics and also a well defined zone topology is provided. The synthesis process must take care of task assignment to nodes and node distribution between zone and abstract channels, in order to optimize application performance and satisfy its requirements.

5.4.1 Synthesis process for AOPs

The optimal solution to the problem can be found analytically but this option is usually unfeasible due to the high number of possible solutions. Furthermore, it is known that similar problems are NP-hard even in specific fields as NoC [7].

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 solution space.

Table 5.1: Problem classification and respective set definitions.

Problem class	Set	Defined attributes	Undefined attributes
Platform Oriented Problem	$\mathscr{T}$	$t.c, t.m$	—
	$\mathscr{F}$	$f.t_s, f.t_d, f.c$	—
	$\mathscr{N}$	$n.T, n.\pi, n.\kappa$	$n.c, n.e, n.m, n.p$
	$\mathscr{A}$	$a.N, a.\kappa$	$a.c, a.d, a.w$
	$\mathscr{Z}$	$z.N, z.e$	—
	$\mathscr{C}$	$c.z_1, c.z_2, c.d$	—
Environment Oriented Problem	$\mathscr{T}$	$t.c, t.m$	—
	$\mathscr{F}$	$f.t_s, f.t_d, f.c$	—
	$\mathscr{N}$	$n.T, n.c, n.e, n.m, n.p, n.\pi, n.\kappa$	—
	$\mathscr{A}$	$a.N, a.c, a.d, a.w, a.\kappa$	—
	$\mathscr{Z}$	$z.N$	$z.e$
	$\mathscr{C}$	—	$c.z_1, c.z_2, c.d$
Application Oriented Problem	$\mathscr{T}$	$t.c, t.m$	—
	$\mathscr{F}$	$f.t_s, f.t_d, f.c$	—
	$\mathscr{N}$	$n.c, n.e, n.m, n.p, n.\pi, n.\kappa$	$n.T$
	$\mathscr{A}$	$a.c, a.d, a.w, a.\kappa$	$a.N$
	$\mathscr{Z}$	$z.e$	$z.N$
	$\mathscr{C}$	$c.z_1, c.z_2, c.d$	—

Chapter 6

Case studies

In this Chapter three case studies are described to show the effectiveness of CASSE in situations where optimization goals differ.

The first case study attempts to find an optimal platform (nodes and channels) specification for a WSN (i.e., distributed temperature control) where the goal is to lower the economic cost of the network; the third case study analyzes the application of CASSE to design a feasible environment specification of a data center facility; the third case study concerns with an application for distributed computing (i.e., multiplication of matrices) and its goal is optimizing application performance (i.e., computation time).

In all cases, the inputs are a multiset of tasks $\mathscr{T}$, a multiset of data flows $\mathscr{F}$, a multiset of nodes $\mathscr{N}$, a multiset of abstract channels $\mathscr{A}$, a set of zones $\mathscr{Z}$ and a set of contiguities $\mathscr{C}$. Each case may provide some constraints and target sets to populate (e.g., for each zone z in $\mathscr{Z}$, each multiset of nodes within the zone $z.N$).

Table 6.1 summarizes some aspects of these case studies.

Table 6.1: Case studies summary.

Problem	Class	Output
Temperature control	POP	$n.c, n.e, n.m, n.p, a.c, a.d, a.w$
Data center plan	EOP	$z.e, \mathscr{C}$
Matrix multiplication	AOP	$n.T, a.N, z.N$

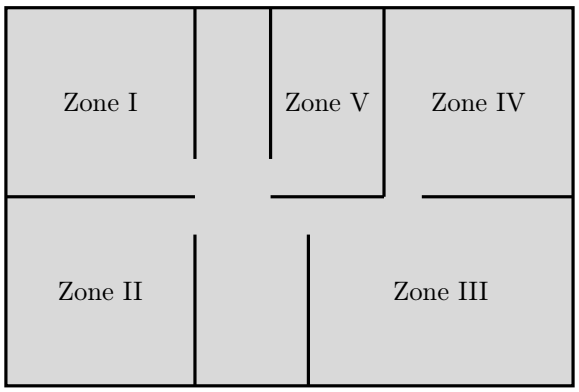


Figure 6.1: Layout of the floor considered in the POP case study.

6.1 Distributed temperature control

The proposed specification model has been applied to a case study consisting in the temperature control application described at the beginning of Chapter 1 and depicted in Figure 1.1 on page 2. 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 as shown in Figure 6.1.

In this case study the topology of the network is well defined and the goal is to synthesize nodes and abstract channels specifications; therefore, this could be considered a Platform Oriented Problem.

6.1.1 Network specification

Task and data flows

We assume as starting point that a multiset 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; they are:

t_1 system initialization;

t_2 centralized temperature control;

t_3 user temperature control;

t_4 air-conditioner actuation;

t_5 temperature sensing.

Tasks exchange information according to the multiset of data flows $\mathscr{F} = \{f_1, f_2, f_3, f_4, f_5\}$ as shown in Figure 6.2 on the facing page.

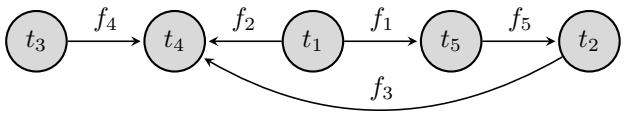


Figure 6.2: Tasks and data flows of the given application in the POP case study.

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:

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

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:

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

Nodes and abstract channels

The multiset of available nodes is $\mathscr{N} = \{n_1^5, n_2^8, n_3\}$, where

$$n_1 = [\{t_4, t_5\}, c, e, m, p, \pi, 2.5],$$
$$n_2 = [\{t_3\}, c, e, m, p, \pi, 6],$$
$$n_3 = [\{t_1, t_2\}, c, e, m, p, \pi, 5]$$

where

$$n_1.\pi \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$$
$$(t.c, n.c) \mapsto (n.c + t.c) \cdot [0.8, 1.1]^T$$
$$n_2.\pi \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$$
$$(t.c, n.c) \mapsto (n.c + t.c) \cdot [1.5, 1.2]^T$$
$$n_3.\pi \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$$
$$(t.c, n.c) \mapsto (n.c + t.c) \cdot [0.6, 0.9]^T$$

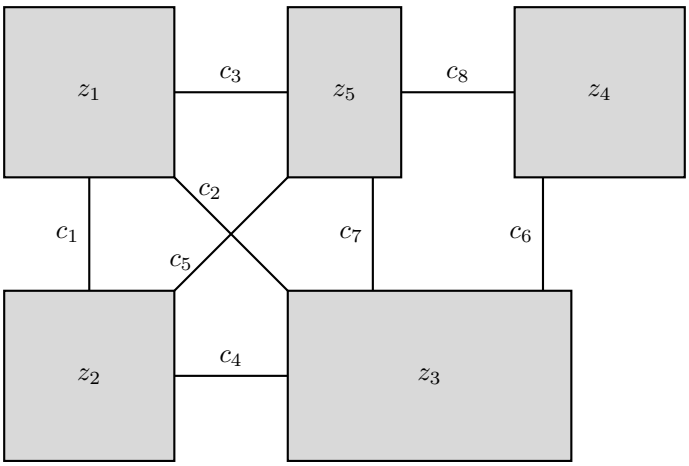


Figure 6.3: Zone graph of given scenario for the POP case study.

and the multiset of available abstract channels is $\mathscr{A} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$, where

$$
\begin{aligned}
a_1 &= [\{n_1, n_2^3\}, c, d, w, 10], \\
a_2 &= [\{n_1, n_2\}, c, d, w, 10], \\
a_3 &= [\{n_1, n_2^2\}, c, d, w, 10], \\
a_4 &= [\{n_1, n_2\}, c, d, w, 10], \\
a_5 &= [\{n_1, n_2\}, c, d, w, 10], \\
a_6 &= [\{n_1^5, n_3\}, c, d, w, 1].
\end{aligned}
$$

Zones and contiguities

The floor in which the application will be deployed has been partitioned into a set of *zones* $\mathscr{Z} = \{z_1, z_2, z_3, z_4, z_5\}$, where

$$
\begin{aligned}
z_1 &= [\{n_1, n_2^3\}, [24, 30]], & z_2 &= [\{n_1, n_2\}, [24, 30]], \\
z_3 &= [\{n_1, n_2^2, n_3\}, [24, 30]], & z_4 &= [\{n_1, n_2\}, [24, 30]], \\
z_5 &= [\{n_1, n_2\}, [24, 30]],
\end{aligned}
$$

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 record the distance between zones as follows:

$$
\begin{aligned}
c_1 &= [z_1, z_2, 3] & c_2 &= [z_1, z_3, 6] \\
c_3 &= [z_1, z_5, 3] & c_4 &= [z_2, z_3, 5] \\
c_5 &= [z_2, z_5, 5] & c_6 &= [z_3, z_4, 3]
\end{aligned}
$$

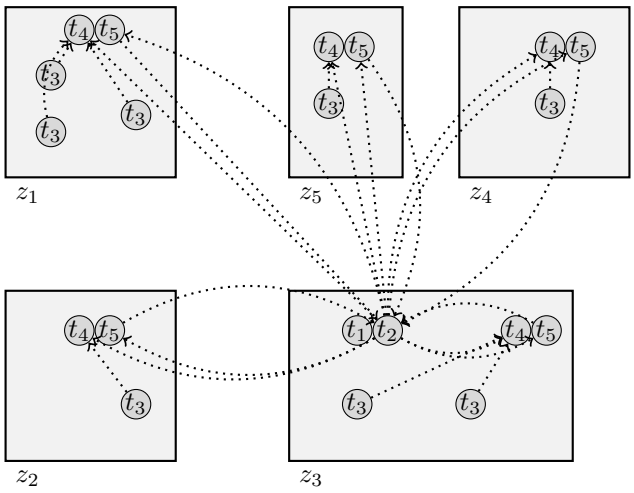


Figure 6.4: Assignment of tasks and data flows to zones in the POP case study.

$$c_7 = [z_3, z_5, 3] \qquad\qquad c_8 = [z_4, z_5, 3]$$

The zone graph is depicted in Figure 6.3 on the preceding page; Figure 6.4 shows the resulting task distribution with the corresponding data flows; Figure 6.5 on the next page shows the resulting assignment of nodes to zones.

6.1.2 Network synthesis

The process proposed for POP in Section 5.2 on page 20 can be used to show how CASSE can help on the synthesis of nodes and channels specifications. The steps to find out the requested values $n.c, n.e, n.m, n.p, a.c, a.d, a.s, a.w$ for each node and abstract channel will be:

1. for each node $n \in \mathscr{N}$, calculate $n.c, n.e, n.m, n.p$ attributes;
2. for each abstract channel $a \in \mathscr{A}$, calculate $a.c, a.d, a.s, a.w$ attributes.

Node synthesis

To calculate $n.c$, we can use Equation 5.1 on page 20 for each node $n \in \mathscr{N}$

$$\begin{aligned} n_1.c &= \sum_{t \in n_1.T} t.c \\ &= t_4.c + t_5.c \\ &= [1,2] + [1,2] = [2,4], \\ n_2.c &= \sum_{t \in n_2.T} t.c \end{aligned}$$

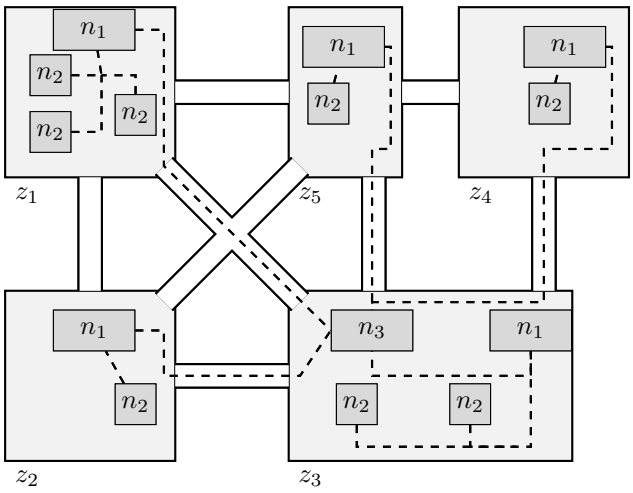


Figure 6.5: Assignment of nodes and abstract channels to zones in the POP case study.

$$
\begin{aligned}
&= t_3.c \\
&= [1,1], \\
n_3.c = &\sum_{t \in n_3.T} t.c \\
&= t_1.c + t_2.c \\
&= [6,4] + [3,5] = [9,9];
\end{aligned}
$$

for $n.p$, we can use Equation 5.3 on page 21

$$
\begin{aligned}
n_1.p = &\sum_{t \in n_1.T} n_1.\pi(t.c, n_1.c) \\
&= n_1.\pi(t_4.c, n_1.c) + n_1.\pi(t_5.c, n_1.c) \\
&= ([2,4] + [1,2]) \cdot [0.8, 1.1]^T + ([2,4] + [1,2]) \cdot [0.8, 1.1]^T = 18, \\
n_2.p = &\sum_{t \in n_2.T} n_2.\pi(t.c, n_2.c) \\
&= n_2.\pi(t_3.c, n_2.c) \\
&= ([1,1] + [1,1]) \cdot [1.5, 1.2]^T = 5.4, \\
n_3.p = &\sum_{t \in n_3.T} n_3.\pi(t.c, n_3.c) \\
&= n_3.\pi(t_1.c, n_3.c) + n_3.\pi(t_2.c, n_3.c) \\
&= ([9,9] + [6,4]) \cdot [0.6, 0.9]^T + ([9,9] + [3,5]) \cdot [0.6, 0.9]^T = 40.5;
\end{aligned}
$$

for $n.e$, we can use Equation 5.2 on page 20

$$\begin{aligned} n_1.e &= \bar{z}.e = [24, 30], \\ n_2.e &= \bar{z}.e = [24, 30], \\ n_3.e &= \bar{z}.e = [24, 30], \end{aligned}$$

where $\bar{z}.e = \min_{z \in \mathscr{Z}} z.e$ since various instances of each node could be located in different zones; for $n.m$, we can use Equation 5.4 on page 21

$$\begin{aligned} n_1.m &= \bigvee_{t \in n_1.T} t.m \\ &= t_4.m \vee t_5.m = \bot, \\ n_2.m &= \bigvee_{t \in n_2.T} t.m \\ &= t_3.m = \top, \\ n_3.m &= \bigvee_{t \in n_3.T} t.m \\ &= t_1.m \vee t_2.m = \bot. \end{aligned}$$

Abstract channel synthesis

To calculate $a.c$, we can use Equation 5.5 on page 21 for each abstract channel $a \in \mathscr{A}$

$$\begin{aligned} a_1.c &= \sum_{f \in F_{a_1}} f.c \\ &= f_4.c + f_4.c + f_4.c \\ &= [3, 1, 0.3] + [3, 1, 0.3] + [3, 1, 0.3] = [9, 3, 0.9], \\ a_2.c &= \sum_{f \in F_{a_2}} f.c \\ &= f_4.c \\ &= [3, 1, 0.3], \\ a_3.c &= \sum_{f \in F_{a_3}} f.c \\ &= f_4.c + f_4.c \\ &= [3, 1, 0.3] + [3, 1, 0.3] = [6, 2, 0.6], \\ a_4.c &= \sum_{f \in F_{a_4}} f.c \\ &= f_4.c \end{aligned}$$

$$
\begin{aligned}
&= [3, 1, 0.3], \\
a_5.c &= \sum_{f \in F_{a_5}} f.c \\
&= f_4.c \\
&= [3, 1, 0.3], \\
a_6.c &= \sum_{f \in F_{a_6}} f.c \\
&= 5 \cdot f_1.c + 5 \cdot f_2.c + 5 \cdot f_3.c + 5 \cdot f_5.c \\
&= 5 \cdot [1, 3, 0.1] + 5 \cdot [1, 3, 0.1] + 5 \cdot [5, 2, 0.1] + 5 \cdot [5, 2, 0.1] = [60, 50, 2],
\end{aligned}
$$

where F_{a_i} is the multiset of data flows defined in Equation 5.7 on page 21; for $a.w$ we can use Equation 5.6 on page 21

$$
\begin{aligned}
a_1.w &= \bigvee_{n \in a_1.N} n.m \\
&= n_1.m \vee n_2.m \vee n_2.m \vee n_2.m = \top, \\
a_2.w &= \bigvee_{n \in a_2.N} n.m \\
&= n_1.m \vee n_2.m = \top, \\
a_3.w &= \bigvee_{n \in a_3.N} n.m \\
&= n_1.m \vee n_2.m \vee n_2.m = \top, \\
a_4.w &= \bigvee_{n \in a_4.N} n.m \\
&= n_1.m \vee n_2.m = \top, \\
a_5.w &= \bigvee_{n \in a_5.N} n.m \\
&= n_1.m \vee n_2.m = \top, \\
a_6.w &= \bigvee_{n \in a_6.N} n.m \\
&= n_1.m \vee n_1.m \vee n_1.m \vee n_1.m \vee n_1.m \vee n_3.m = \bot;
\end{aligned}
$$

for $a.d$ we can use Equation 5.8 on page 21

$$
\begin{aligned}
a_1.d = a_2.d = a_3.d = a_4.d = a_5.d \\
= \max_{z_1, z_2 \in \bigcup_{i=1}^{5} Z_{a_i}} d(z_1, z_2) = 0, \\
a_6.d = \max_{z_1, z_2 \in Z_{a_6}} d(z_1, z_2) = 6,
\end{aligned}
$$

Table 6.2: Result of the synthesis process for the POP case study.

Node	#	$n.c$	$n.e$	$n.m$	$n.p$
n_1	5	$[2, 4]$	$[24, 30]$	$\bot$	18
n_2	8	$[1, 1]$	$[24, 30]$	$\top$	5.4
n_3	1	$[9, 9]$	$[24, 30]$	$\bot$	40.5

A.C.	#	$a.c$	$a.d$	$a.w$
a_1	1	$[9, 3, 0.9]$	0	$\top$
a_2	1	$[3, 1, 0.3]$	0	$\top$
a_3	1	$[6, 2, 0.6]$	0	$\top$
a_4	1	$[3, 1, 0.3]$	0	$\top$
a_5	1	$[3, 1, 0.3]$	0	$\top$
a_6	1	$[60, 50, 2]$	6	$\bot$

where Z_{a_i} is the set of zones defined in Equation 5.9 on page 21.

6.1.3 Result of the synthesis process

This case study showed how CASSE is able to provide an efficient specification model and consequently give a synthesis process to characterize the target node and abstract channels. Table 6.2 shows all the results of the synthesis process.

Finally, a couple of remarks about costs and power consumption. The total cost of the synthesized solution is

$$\begin{aligned} K &= \sum_{n \in \mathcal{N}} n.\kappa + \sum_{a \in A} a.\kappa \\ &= 5 \cdot n_1.\kappa + 8 \cdot n_2.\kappa + n_3.\kappa \\ &+ a_1.\kappa + a_2.\kappa + a_3.\kappa + a_4.\kappa + a_5.\kappa + a_6.\kappa \\ &= 116.5 \end{aligned}$$

and the power consumption

$$\begin{aligned} P &= \sum_{n \in \mathcal{N}} \sum_{t \in n.T} n.\pi(t.c, n.c) \\ &= 5 \cdot (n_1.\pi(t_4.c, n_1.c) + n_1.\pi(t_5.c, n_1.c) \\ &+ 8 \cdot n_2.\pi(t_3.c, n_2.c) + \\ &+ n_3.\pi(t_1.c, n_3.c) + n_3.\pi(t_2.c, n_3.c) \\ &= 173.7 \end{aligned}$$

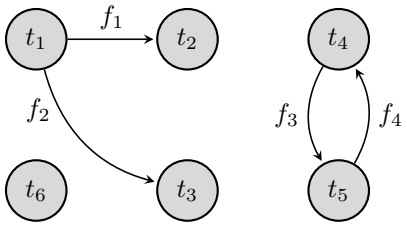


Figure 6.6: Task and flow graphs involved in the EOP case study.

6.2 Data center

This case study express the capacity of CASSE to come in help when the need of the designer is to model the environment specifications to make zones suitable to host nodes with specific environmental requirements. In this case study, the goal is to synthesize such environment specifications in a data center facility, where the large number of devices (i.e., computers, routers, switches, backup systems, racks...) requires a specific temperature, humidity and power supply.

6.2.1 Network specification

Since there are many different applications running in the data center, and most of them are independent from each other, the *flow graph* may have various independent sub-graphs (see Figure 6.6);

$$\mathscr{T} = \{t_1, t_2, t_3, t_4, t_5, t_6\} \quad \text{and} \quad \mathscr{F} = \{f_1, f_2, f_3\}.$$

Tasks' computation capabilities $t.c$ are defined in $\mathbb{R}$,

$$\begin{aligned} t_1 &= [5, \bot], & t_2 &= [8, \bot], \\ t_3 &= [2, \bot], & t_4 &= [6, \bot], \\ t_5 &= [9, \bot], & t_6 &= [2, \bot]; \end{aligned}$$

data flows' communication requirements $f.c$ are defined in $\mathbb{R}$ too,

$$\begin{aligned} f_1 &= [t_1, t_2, 5], & f_2 &= [t_1, t_3, 5], \\ f_3 &= [t_4, t_5, 1], & f_4 &= [t_5, t_4, 9]. \end{aligned}$$

The multiset of nodes is

$$\mathscr{N} = \{n_1, n_2, n_3, n_4, n_5, n_6\},$$

where $n.c \in \mathbb{R}$ and $n.e \in \mathbb{R}^3$,

$$
\begin{aligned}
n_1 &= [\{t_1\}, 6, [30, 10, 220], \bot, 0, \pi, 5], \\
n_2 &= [\{t_2\}, 10, [45, 20, 220], \bot, 0, \pi, 3.5], \\
n_3 &= [\{t_3\}, 5, [20, 15, 220], \bot, 0, \pi, 4], \\
n_4 &= [\{t_4\}, 6, [30, 8, 220], \bot, 0, \pi, 2.6], \\
n_5 &= [\{t_5\}, 9, [50, 12, 220], \bot, 0, \pi, 5], \\
n_6 &= [\{t_6\}, 4, [10, 0, 220], \bot, 0, \pi, 2]
\end{aligned}
$$

where

$$
\begin{aligned}
n_1.\pi &\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\
&\quad (t.c, n.c) \mapsto (n.c + t.c) \cdot 2 \\
n_2.\pi &\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\
&\quad (t.c, n.c) \mapsto (n.c + t.c) \cdot 1.5 \\
n_3.\pi &\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\
&\quad (t.c, n.c) \mapsto n.c + t.c \\
n_4.\pi &\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\
&\quad (t.c, n.c) \mapsto (n.c + t.c) \cdot 1.2 \\
n_5.\pi &\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\
&\quad (t.c, n.c) \mapsto n.c + t.c \\
n_6.\pi &\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\
&\quad (t.c, n.c) \mapsto n.c + t.c
\end{aligned}
$$

Abstract channels are defined in $\mathscr{A} = \{a_1, a_2\}$ as

$$
\begin{aligned}
a_1 &= [\{n_1, n_2^8\}, 40, 100, \bot, 2], \\
a_2 &= [\{n_4^3, n_5^3\}, 30, 22, \bot, 6].
\end{aligned}
$$

Finally, zones are defined in $\mathscr{Z} = \{z_1, z_2, z_3\}$ as

$$
z_1 = [\{n_1, n_2^8\}, e], \qquad z_2 = [\{n_4^3, n_5^3\}, e],
$$
$$
z_3 = [\{n_6^4\}, e].
$$

and $\mathscr{C} = \emptyset$.

The resulting scenario is depicted in Figure 6.7 on the following page.

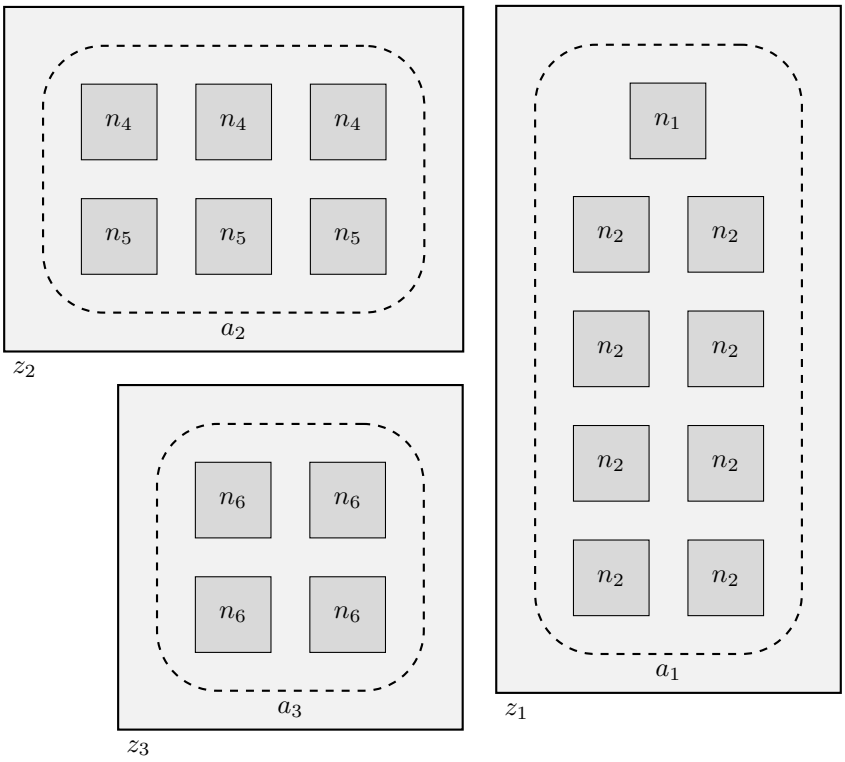


Figure 6.7: Scenario for the EOP case study.

Table 6.3: Result of the synthesis process for the EOP case study.

Zone	$z.e$
z_1	$[20, 10, 220]$
z_2	$[30, 8220]$
z_3	$[10, 0, 220]$

6.2.2 Network synthesis

This is clearly an EOP so the synthesis process proposed in Section 5.3 on page 21 can be followed:

1. for each zone $z \in \mathscr{Z}$ calculate $z.e$;
2. populate contiguities set $\mathscr{C}$.

Zone synthesis

To calculate $z.e$ it's possible to use Equation 4.11 on page 18

$$z_1.e = \min_{n \in z_1.N} n.e = [20, 10, 220],$$
$$z_2.e = \min_{n \in z_2.N} n.e = [30, 8, 220],$$
$$z_3.e = \min_{n \in z_3.N} n.e = [10, 0, 220].$$

Contiguities synthesis

Since none of the abstract channels have nodes in different zones, contiguities set $\mathscr{C}$ can be empty; formally

$$\forall a \in \mathscr{A} \forall z_1, z_2 \in \mathscr{Z}\ z_1 \neq z_2\ \nexists n_1, n_2 \in a.N \mid n_1 \in z_1.N \wedge n_2 \in z_2.N.$$

6.2.3 Result of the synthesis process

This case study showed how CASSE is able to provide an efficient specification model to help the synthesis of environmental features satisfying given requirements. Table 6.3 summarizes the result of the synthesis process.

Cost and power consumption remarks are not necessary since the synthesis process doesn't affect them.

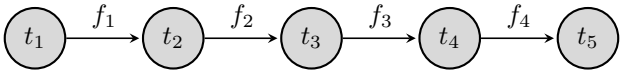


Figure 6.8: Flow graph of the given application for the AOP case study.

6.3 Matrix multiplication

The formal model provided with CASSE is able to describe efficiently applications for distributed computation, such as an algorithm for matrix multiplication. This is clearly an Application Oriented Problem, since the goal is the synthesis of an optimal combination of tasks and nodes to allow the application the chance to maximize parallelism.

6.3.1 Network specification

The application for matrix multiplication is given in the form of a *flow graph* $FG = (\mathscr{T}, \mathscr{F})$ (see Figure 6.8), where

$$\mathscr{T} = \{t_1, t_2, t_3, t_4, t_5\} \quad \text{and} \quad \mathscr{F} = \{f_1, f_2, f_3, f_4\}.$$

CPU and memory usage for each task are defined in $\mathbb{R}^2$ as follows:

$$t_1 = [[2,4], \perp], \qquad t_2 = [[1,3], \perp],$$
$$t_3 = [[4,1], \perp], \qquad t_4 = [[3,2], \perp],$$
$$t_5 = [[2,3], \perp].$$

On the other hand, data flows have communication requirements defined in $\mathbb{R}^3$:

$$f_1 = [t_1, t_2, [5,3,0.2]], \qquad f_2 = [t_2, t_3, [3,3,0.2]],$$
$$f_3 = [t_3, t_4, [1,3,0.2]], \qquad f_4 = [t_4, t_5, [1,3,0.2]].$$

Though not necessary, a description of each task is provided for clarity:

t_1 splits input matrices in vectors;

t_2 splits input vectors in single values;

t_3 multiplies input values;

t_4 sums up input values;

t_5 builds the output matrix.

Given matrices $X_{m\times n}$ and $Y_{n\times p}$ for a time-optimal computation there will be $m \cdot p$ instances of t_2 and t_4, and $m \cdot n \cdot p$ instances of t_3 (see Example 6.3.1 on the next page).

The multiset of available nodes is $\mathscr{N} = \{n_1, n_2\}$, where:

$$n_1 = [T, [4, 7], [50, 60], \perp, 0, \pi, 12],$$
$$n_2 = [T, [4, 1], [50, 60], \perp, 0, \pi, 4]$$

where

$$\begin{aligned} n_1.\pi\colon \mathbb{R}^2 \times \mathbb{R}^2 &\to \mathbb{R} \\ (t.c, n.c) &\mapsto (n.c + t.c) \cdot [0.4, 0.3]^T \\ n_2.\pi\colon \mathbb{R}^2 \times \mathbb{R}^2 &\to \mathbb{R} \\ (t.c, n.c) &\mapsto (n.c + t.c) \cdot [0.6, 0.5]^T \end{aligned}$$

and the multiset of available abstract channels is $\mathscr{A} = \{a_1\}$, where

$$a_1 = [N, [100, 3, 0.2], 100, \perp, 40].$$

The set of zones is $\mathscr{Z} = \{z_1\}$, where $z_1 = \{N, [26, 60]\}$; the set of contiguities $\mathscr{C}$ is empty.

6.3.2 Network synthesis

The process proposed for AOPs in Section 5.4 on page 22 can be used to show the effectiveness of CASSE. The steps to find out the requested output values $n.T, a.N, z.N$ will be:

1. for each task $t \in \mathscr{T}$, instantiate as many occurrences as needed;
2. for each task instance, assign it to an existing node with compatible attributes or instantiate a new one;
3. for each instantiated node, assign it to an existing zone with compatible attributes;
4. for each instantiated node, assign it to an existing abstract channel whose attributes satisfy communication requirements of node's tasks or instantiate a new one.

Since the number of task and nodes is humanly approachable, no particular heuristic is used and the choices are made at a rough guess.

Task instantiation

First, we must decide how many tasks we should instantiate and to which nodes we could assign them to. Still considering matrices $X_{m\times n}$ and $Y_{n\times p}$ as input, there will be a single instance of both t_1 and t_5, $m \cdot p$ instances of t_2 and t_4, and $m \cdot n \cdot p$ instances of t_3.

Node assignment

Since the goal is optimizing the time spent by the computation, we assign each instance of t_3 to $m \cdot n \cdot p$ instances of n_2 (n_2 is cheaper, $n_2.k < n_1.k$, and its capacity fits the requirements of t_3, $t_3.c \leq n_2.c$); each instance of t_2 and t_4 to $m \cdot p$ instances of n_1 ($t_2.c + t_4.c \leq n_1.c$) and instances of t_1 and t_5 to another instance of n_1 ($t_1.c + t_5.c \leq n_1.c$).

All nodes will be placed within zone z_1, since $\forall n \in \mathscr{N}\ n.e \geqslant z_1.e$ and there are no other zones; the resulting mesh could be similar to the one depicted in Figure 6.10 on the next page, related to Example 6.3.1.

Abstract Channel placement

Once tasks are assigned to nodes and nodes are placed within zones, data flows show which nodes have the necessity to communicate with others. Since the available abstract channel a_1 fits the communication requirements of all data flows ($\sum_{i=1}^{|\mathscr{F}|} f_i.c \leq a_1.c$), we can instantiate just once a_1 and fill $a_1.N$ with all the nodes.

6.3.3 Result of the synthesis process

In the following example is shown the result of the synthesis process described in this case study for two specific matrices given as input to the application.

Example 6.3.1 (Matrix multiplication)**.** Given matrices

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} y_{1,1} & y_{1,2} \\ y_{2,1} & y_{2,2} \\ y_{3,1} & y_{3,2} \end{bmatrix}$$

the flow graph with multiple task instances is shown in Figure 6.9 on the next page and the final configuration is depicted in Figure 6.10 on the facing page.

The goal of this case study was the minimization of computation time. The solution given in Table 6.4 on page 42 maximizes parallelism, reducing communication delays to the minimum.

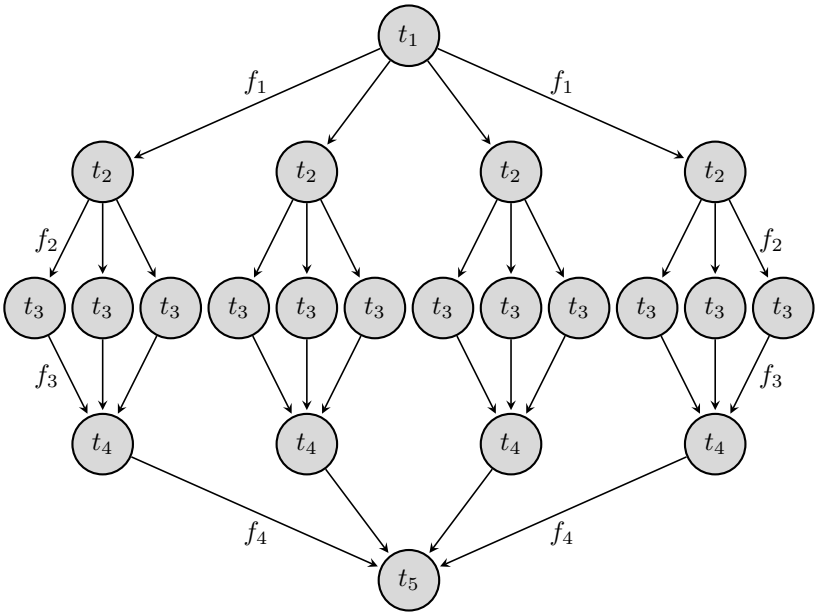


Figure 6.9: Flow graph with multiple task instances for the AOP case study.

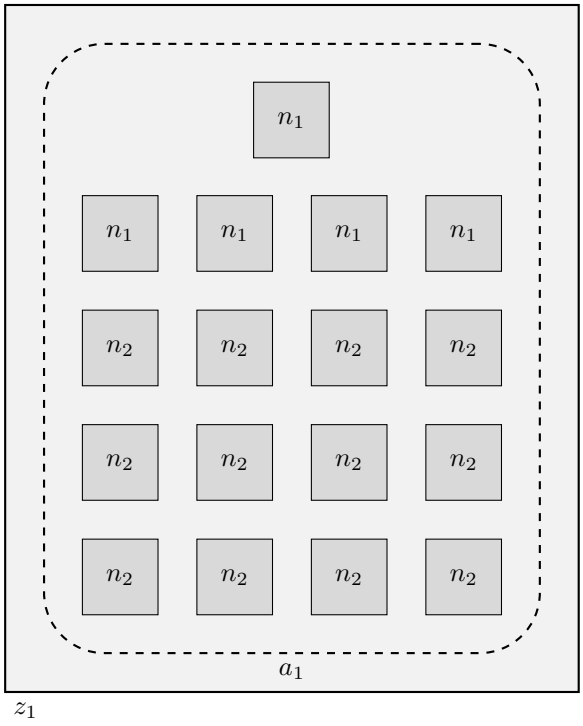


Figure 6.10: Nodes and abstract channel topology for the AOP case study.

Table 6.4: Result of the synthesis process for the AOP case study.

Node	#	$n.T$	A.C.	#	$a.N$	Zone	$z.N$
n_1	1	t_1, t_5	a_1	1	n_1^5, n_2^{12}	z_1	n_1^5, n_2^{12}
n_1	4	t_2, t_4					
n_2	12	t_3					

Finally, a couple of remarks about costs and power consumption. The total cost of the synthesized solution is

$$\begin{aligned} K &= \sum_{n \in \mathcal{N}} n.\kappa + \sum_{a \in A} a.\kappa \\ &= 5 \cdot n_1.\kappa + 12 \cdot n_2.\kappa + a_1.\kappa \\ &= 148 \end{aligned}$$

and the total power consumption

$$\begin{aligned} P &= \sum_{n \in \mathcal{N}} \sum_{t \in n.T} n.\pi(t.c, n.c) \\ &= n_1.\pi(t_1.c, n_1.c) + n_1.\pi(t_5.c, n_1.c) \\ &+ 4 \cdot (n_1.\pi(t_2.c, n_1.c) + n_1.\pi(t_4.c, n_1.c)) \\ &+ 12 \cdot n_2.\pi(t_3.c, n_2.c) \\ &= 120.3 \end{aligned}$$

Chapter 7

Conclusions and future works

An extension of the traditional design flow has been proposed for distributed applications based on networked embedded systems. Additional design steps have been introduced to model the application from a communication perspective and provide the synthesis of the communication infrastructure.

The former is the main focus of CASSE, i.e., the creation of a network specification environment to model tasks, data flows, nodes, channels and topology. Entities and relationships have been introduced through a mathematical approach which simplifies the specification of requirements and constraints.

A classification of synthesis methodologies has been proposed to explain three different approaches to the network synthesis process, depending on design goals, i.e., a platform oriented methodology, an environment oriented methodology and an application oriented methodology.

Finally, case studies have been described to show the effectiveness of the model applied to three different network application, one for each methodology.

This work is a first contribution to a global strategy for network synthesis and thus some research topics are still open and will be addressed to future works:

- extension of the model with *time* and *probability* concepts;
- fast and efficient heuristics to perform a design-space exploration of network solutions;
- the role of simulation in the network modeling and design-space exploration.

Acronyms

CASSE Communication-Aware Specification and Synthesis Environment . . . 3

ECSI Electronic Chips and Systems design Initiative

FDL Forum on specification and Design Languages 4

WSN Wireless Sensor Network 5

PBD Platform-Based Design 5

NoC Network-on-Chip 5

QoS Quality of Service 5

MILP Mixed Integer Linear Programming 6

UML Unified Modeling Language 7

HDL Hardware Description Language 7

VHSIC Very High Speed Integrated Circuits

VHDL VHSIC Hardware Description Language 7

MARTE Modeling and Analysis of Real-Time and Embedded systems 9

NFR Non-Functional Requirements 9

SRM Software Resource Modeling 9

HRM Hardware Resource Modeling 9

SysML System Modeling Language 9

TLM Transaction Level Modeling 9

FSM Finite State Machine 6

POP Platform Oriented Problem 20

EOP Environment Oriented Problem 21

AOP Application Oriented Problem 22

Bibliography

[1] Nicola Bombieri, Franco Fummi, and Davide Quaglia. System/network design space exploration based on tlm for networked embedded systems. *ACM Transactions on Embedded Computer Systems*, to appear 2010.

[2] Franco Fummi, Giovanni Lovato, Davide Quaglia, and Francesco Stefanni. Modeling of communication infrastructure for design-space exploration. In *Forum on specification, verification and Design Languages (FDL 2010)*, Southampton, UK, September 2010. ECSI.

[3] Amol Bakshi and Viktor K. Prasanna. Algorithm design and synthesis for wireless sensor networks. In *International Conference on Parallel Processing (ICPP 2004)*, pages 15–18 (1). IEEE Society, August 2004.

[4] 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)*, pages 1–6 (1), Munich, March 2006. IEEE Society.

[5] Luca Mottola, Animesh Pathak, Amol Bakshi, Viktor K. Prasanna, and Gian Pietro Picco. Enabling scope-based interactions in sensor network macroprogramming. In *Internatonal Conference on Mobile Adhoc and Sensor Systems*, pages 1–9. IEEE, October 2008.

[6] Tobias Bjerregaard and Shankar Mahadevan. *A Survey of Research and Practices of Network-on-Chip.*

[7] A. Agarwal, C. Iskander, H.T. Multisystems, and R. Shankar. Survey of Network-on-Chip (NoC) architectures & contributions. In *Journal of Engineering, Computing and Architecture*, 2009.

[8] E. Rijpkema, K.G.W. 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.

[9] 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*, 25(1):100–108, January 2002.

[10] 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.

[11] R. Chae-Eun, J. Han-You, and Ha 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.

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

[13] Ahmed Khoumsi, Gregor von Bochmann, and Rachida 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)*, pages 417–433, Deventer, The Netherlands, The Netherlands, 1999. Kluwer, B.V.

[14] Hirozumi Yamaguchi, Kozo Okano, Teruo Higashino, and Kenichi Taniguchi. Protocol synthesis from time petri net based service specification. *Parallel and Distributed Systems, International Conference on*, 0:236, 1997.

[15] Robert L. Probert and Kassem Saleh. Synthesis of communication protocols: Survey and assessment. *IEEE Trans. Comput.*, 40(4):468–476, 1991.

[16] Peter Van Eijk and Jeroen Schot. An exercise in protocol synthesis. In *Formal Description Techniques IV. North-Holland*, pages 117–131, 1991.

[17] Object Management Group. MARTE. `http://www.omgmarte.org/`.

[18] Object Management Group. Unified Modeling Language. `http://www.uml.org/`.

[19] Object Management Group. SysML. `http://www.sysml.org/`.

[20] Mathworks. Simulink. `http://www.mathworks.com/products/simulink/`.

[21] Mathworks. Stateflow. `http://www.mathworks.com/products/stateflow/`.

[22] Center for Hybrid and Embedded Software System. Ptolemy. `http://ptolemy.eecs.berkeley.edu/index.htm`.

[23] G. Kahn. The semantics of a simple language for parallel programming. *Info. Proc.*, pages 471–475, 74 (4), 1974.

[24] Open SystemC Initiative. IEEE Std 1666 — 2005 IEEE Standard SystemC Language Reference Manual. *IEEE Std 1666-2005*, pages 1–423, 2006.

[25] Transaction Level Modeling Working Group. OSCI TLM 2.0. `http://www.systemc.org`, 2006.

[26] Center for Embedded and Computer Systems. SpecC. `http://cecs.uci.edu/~specc`.

[27] Center for Electronic Systems Design. Metropolis. `http://embedded.eecs.berkeley.edu/metropolis/index.html`.

Index