UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA

University of Verona

A.A 2016/2017

Laboratory of Networked Embedded Systems

Lesson 4

Toolchain for Network Synthesis

Enrico Fraccaroli

May 14, 2018

Contents

Chapter 1

Introduction

1.1 Network Synthesis

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

1.2 Communication Aware Specification

1.2.1 Tasks

A task represents a basic functionality of the whole 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 itself and its HW/SW implementation but rather on its computational and mobility requirements.

A task $t \in \mathcal{T}$ is a triple $t = [s, m, z]$ whose three attributes (*i.e.*, components) are as follows

$s \in \mathbb{R}_{\geq}$	represents the task size, *i.e.*, the overall resource requirements in order for task t to perform its activity;
$m \in \mathbb{B}$	specifies whether the task should be necessarily placed on a mobile node;
$z \in \mathcal{Z}$	specifies to which zone the task belongs.

1.2.2 Data flows

A data-flow (DF) represents the communication between two tasks; output from the source task is delivered as input to the destination task. Network synthesis focuses only on the communication requirements which affect the choice of channels and protocols between the nodes hosting the involved tasks. A data-flow $d = [ts, td, s, d, e] \in \mathcal{D}$ is characterized by the attributes

- $ts, td \in \mathcal{T}$ the source and destination tasks;
- $s \in \mathbb{R}_{\geq}$ represents the data-flow size (bit-rate);
- $d \in \mathbb{R}_{\geq}$ indicates the maximum accepted delay;
- $e \in \mathbb{R}_{\geq}$ specifies the maximum acceptable error rate.

1.2.3 Nodes

A node can be seen as a container of tasks. At the end of the synthesis flow, nodes will become HW entities with CPUs and network interfaces and tasks will be implemented either as HW components or as SW 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. A node $n = [s, k, e, te, m] \in \mathcal{N}$ is a tuple whose attributes are as follows

- $s \in \mathbb{R}_{\geq}$ represents the node's size, *i.e.*, the available computational resources;
- $k \in \mathbb{R}_{\geq}$ denotes the node's economic cost;
- $e \in \mathbb{R}_{\geq}$ is the energy consumption intrinsic to the node, without considering the tasks contained in it and their contribution to the total consumption;
- $te \in \mathbb{R}_{\geq}$ determines the energy consumption of the tasks assigned to the node over a TU (each task t mapped into the node n consumes an amount of energy equal to $t.s$ times $n.te$);
- $m \in \mathbb{B}$ identifies if the node is mobile or static.

1.2.4 Abstract Channels

An abstract channel (AC) can be seen as a container of data-flows. It is an ideal connection between two or more nodes. An abstract channel $ac = [e, de, k, w, s, dl, er] \in \mathcal{A}$ is a tuple characterized as follows

$e \in \mathbb{R}_{\geq}$	is the energy consumption of the channel;
$de \in \mathbb{R}_{\geq}$	is the energy require to send a bit through the channel over a TU (each data-flow d deployed inside the channel c consumes an amount of energy equal to $d.s$ times $c.de$);
$k \in \mathbb{R}_{\geq}$	specifies its economic cost;
$w \in \mathbb{B}$	tells whether the channel is wireless;
$s \in \mathbb{R}_{\geq}$	the size (capacity) of the channel;
$dl \in \mathbb{R}_{\geq}$	represents the transmission delay of the channel;
$er \in \mathbb{R}_{\geq}$	specifies its error rate.

A data-flow can be assigned to a static AC only when each task participating to the data-flow is mapped into a static node. To make this constraint more explicit, we say that a data-flow *involves* a node n whenever some but not all of its tasks are mapped into n. We then require: *a data-flow involving a mobile node can not be assigned to a static AC.* It is worth noting that the abstract channel has the same three attributes of a data-flow, but the former represents the communication resources provided by the channel while the latter represents the communication requirements needed by the data-flow and the involved tasks.

1.2.5 Zones

The physical devices represented by the nodes are ultimately deployed in 3D space. Our model takes care of some abstract topological issues by partitioning the space into a finite set of zones Z related by contiguities. Zones and contiguities allow an environment-aware design of network topologies. We start by describing zones since these are often determined and characterized by an environmental attribute (*e.g.*, a temperature value). A zone $z = [p] \in \mathcal{Z}$ is a tuple characterized as follows

$p = (x, y, z) \in \mathbb{R}^3_{\geq}$	denotes the zone position inside 3D space.

1.2.6 Contiguities

Contiguities describe the relation between zones. Two nodes placed in the same zone are always able to communicate with the default quality of service of the involved abstract channel. This quality however might drop because of distance, interference, or obstacles, in case the nodes are deployed into two different zone. A contiguity $cnt = [z_1, z_2, ac, c] \in \mathcal{C}$ is a tuple whose attributes are characterized as follows

- $z_1, z_2 \in \mathcal{Z}$ are the source and destination zones; $ac \in \mathcal{A}]$ is the channel to which the contiguity applies;
- $c \in \mathbb{R}_{\geq}$ represents the environmental effects due to the border between two zones (*i.e.*, z_1 and z_2) on the quality communications of a given channel (*i.e.*, ac), by means of a index of conductivity;
- $dc \in \mathbb{R}_{\geq}$ represents the cost required to deploy the given cable channel between the given pair of zones. Thus, such value is of interest only for non wireless channels.

Given two zones $z_1, z_2 \in \mathcal{Z}$ and a channel $ac \in \mathcal{A}$, the hash function $cont(z_1, z_2, ac)$ allows to efficiently retrieve the corresponding contiguity. The proposed entities can be put in relation by using graphs. Moreover, they can be used by the designer to specify the application requirements which can be expressed through formal relations between their attributes.

Chapter 2

Methodology

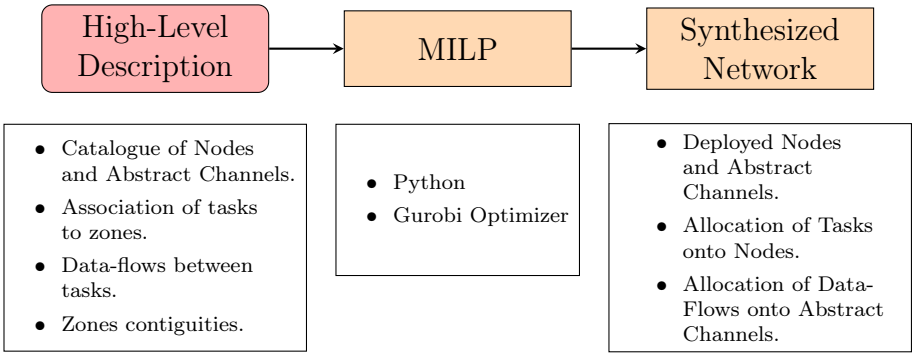


Figure 2.1: Network synthesis flow.

The methodology starts from a **high-level description** of the distributed embedded system, which is **implementation-independent**. The final result is a **synthesized network infrastructure** which can be used for the generation of both a simulation model and the actual deployment.

2.1 High-level description

The high level description comprises the catalogs and the input instance. The catalogs contains:

- **Node Catalog**
  Contains the possible types of nodes.
- **Abstract Channel Catalog**
  Contains the possible types of channels.

An **Input Instance** which reports all the details concerning the problem.

- Set of Tasks inside Zones.
- Set of Data-Flows.
- Set of Contiguities between Zones.

2.2 Synthesized network

A feasible solution to the problem consists of:

- How many nodes of each type should be placed in each zone.
- Which of the instantiated nodes will host the tasks.
- How many abstract channels of each type should be deployed.
- Which of the activated abstract channels will host the data-flows.

Chapter 3

Mixed-integer linear programming

Integer Linear Program (ILP)

1. An optimization model is an **Integer Linear Program**.
2. If all of its variables are discrete, the model is a **pure integer linear program**.
3. Otherwise, the mode is a **mixed-integer linear program**.

The problem which are most commonly solved are of the form:

- **Objective**
  minimize $c^T x$ (linear cost function)
- **Constraints**
  $Ax \leq b$ (linear constraints)
  $x \geq 0$ (bound constraints)
  some or all xj must take integer values (integrality constraints).

3.1 Variables

Continuous Variables

- $N_{n,z}$ How many nodes of type n are deployed inside zone z.
  The upperbound on the number of active nodes is: $\overline{N_{n,z}}$
- C_c Hown many channels of type c are activated.
  The upperbound on the number of active channels is: $\overline{C_c}$

Binary Variables Let us see some examples

Variable	Description
$x_{n,z,p}$	The *p-th* node of type n has been allocated in zone z.
$y_{n,z,p}$	The *p-th* channel of type c has been deployed.
$w_{t,n,p}$	Task t is placed inside the *p-th* node of type n.
$h_{d,c,p}$	Data-flow d is placed inside the *p-th* channel of type c.
ρ_{t_1,t_2}	Tasks t_1 and t_2 are mapped into different nodes.
m_d	One of the tasks of data-flow d has a mobility requirements.

3.2 Constraints

Let us see and example of constraint. The number of active nodes ($N_{n,z}$) is equal to the sum of all the actually instantiated nodes ($x_{n,z,p}$), expressed in constraint C.1.

$$N_{n,z} = \sum_{p=1}^{\overline{N}_{n,z}} x_{n,z,p} \qquad \forall n \in \mathbb{N}_N,\ \forall z \in \mathbb{N}_Z \tag{C.1}$$

Another example concerns the data-flows. Their placement depends on whether the tasks which they connect reside in the different nodes or not. The former case is formalized in Constraint C.2, where data-flows have tasks which reside in different zones, thus their placement inside a channel is necessary.

$$\sum_{c}^{\alpha_c(d)} \sum_{p=1}^{\overline{C}_c} h_{d,c,p} = 1 \qquad \forall d \in \mathbb{N}_D,\ d.ts.z \neq d.td.z \tag{C.2}$$

On the latter case, where data-flow is not necessarily assigned to a channel, its placement depends on variable ρ and ensure by Constraint C.3.

$$\sum_{c}^{\alpha_c(d)} \sum_{p=1}^{\overline{C}_c} h_{d,c,p} = \rho_{d.ts,d.td} \qquad \forall d \in \mathbb{N}_D,\ d.ts.z = d.td.z \tag{C.3}$$

3.3 Objectives

Our goal is to minimize or maximize an objective function value.
For what concerns the network synthesis problem, one could choose to Minimize the Economic Cost of the synthesized network. Such objective can be specified as follows:

$$min\Big[\sum_{n\in N}\sum_{z\in Z}\sum_{p\in\overline{N_{n,z}}}(x_{n,z,p} * n_{price}) + \sum_{c\in C}\sum_{p\in\overline{C_c}}(y_{c,p} * c_{price})\Big]$$

We've basically summed the costs of all the actually instantiated nodes to the sum of all the actually instantiated channels.

Chapter 4

Setup and first execution

4.1 Install Gurobi

Now we will se the steps necessary to get a free version of Gurobi-Optimizer and how to set up the experiments:

1. First, you have to register at:
   - http://www.gurobi.com/registration/general-reg
   - Select as Account Type: **Academic**.
   - At the end of the registration process you will receive a mail.
2. Open the received mail:
   - Inside the mail you will found a link which will allow you to set a password for your Gurobi account.
3. Download gurobi-optimizer at:
   - http://www.gurobi.com/downloads/gurobi-optimizer
   - Select the platform that you are using.
   - This will download a file inside your Donwloads directory.
4. Move the downloaded compressed file inside your home directory
5. Untar the compressed file (change X.X.X with the identifier of your downloaded version).

```
tar xvzf gurobiX.X.X_linux64.tar.gz
```

6. Rename the uncompressed directory:

```
mv gurobiX.X.X ${HOME}/Gurobi
```

7. Create a script:

```
gedit ${HOME}/set-gurobi-env.sh
```

8. Place the following commands inside the script:

```
export GUROBI_HOME="${HOME}/Gurobi/linux64"
export PATH="${PATH}:${GUROBI_HOME}/bin"
export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:${GUROBI_HOME}/lib"
export GRB_LICENSE_FILE="${GUROBI_HOME}/gurobi.lic"
```

9. Make the script executable:

```
chmod +x ${HOME}/set-gurobi-env.sh
```

10. Execute the script:

```
source ${HOME}/set-gurobi-env.sh
```

11. Get a free academic license at:

```
https://user.gurobi.com/download/licenses/free-academic
```

12. Copy the command at the end of the of the page, the one which has the following form:

```
grbgetkey xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

13. Paste the command inside your bash.

14. When prompted set the destination folder to your `GUROBI_HOME`. So if you've followed the instruction just type:

```
${HOME}/Gurobi/linux64
```

15. Move inside `GUROBI_HOME`:

```
cd $GUROBI_HOME
```

16. Install Gurobi inside a directory of your own:

```
python setup.py install --prefix=~/GurobiLib
```

17. Lets define a new environment variable which points to Gurobi library (replace X.X with your version):

```
export GUROBI_LIB=${HOME}/GurobiLib/lib/pythonX.X/site-packages/gurobipy
```

18. Let python know where the library is:

```
export PYTHONPATH=${PYTHONPATH}:${GUROBI_LIB}
```

19. You can place the previous two exports inside your `set-gurobienv.sh` script.

4.2 Run the network synthesizer

In order to use the synthesizer you need to:

1. download the source code for the today's laboratory;
2. decompress the file `source_code_lesson_4.tar.gz`;
3. move inside the directory which contains the source code:

```
cd source_code_lesson_4
```

4. type the following command to see the arguments of the synthesizer:

```
python Synthesizer.py
```

5. Execute the synthesizer on one of the examples:

```
python Synthesizer.py EXAMPLE_NAME/input \
                      EXAMPLE_NAME/nodes \
                      EXAMPLE_NAME/channels \
                      1
```

For the 2016/2017 version of the synthesizer, you can find an example directly inside `case_study_1`. For the 2017/2018 version of the synthesizer, you can find an example inside `examples/test_case_1`.

4.3 Write a high-level description

The synthesizer needs three files:

1. input: Contains the high-level description of the scenario.
2. nodes: Contains the library of nodes.
3. channels: Contains the library of nodes.

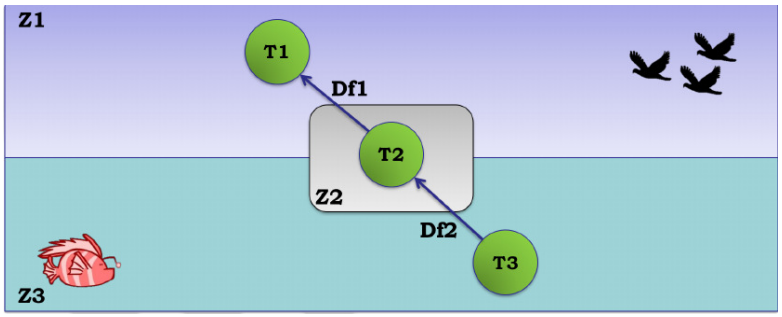
Each input file underlies the following notation and rules:

- A row which starts with a '#' or ';' is a comment and therefore ignored.
- **Do not leave empty lines**.

Chapter 5

Exercises

5.1 Exercise 1



Model the scenario above using the presented formalism, and choose contiguities, nodes, and channels properly.

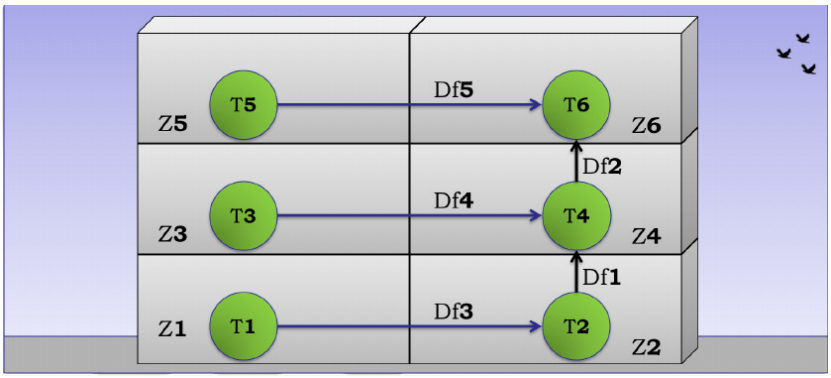
Tasks

Identifier	Size	Mobile
T1	10	1
T2	25	1
T3	5	1

Data-Flows

Data-Flow	Source	Destination	Bandwidth	Max Delay	Max Energy
Df1	T2	T1	7	5	15
Df2	T3	T2	25	15	3

5.2 Exercise 2



Model the scenario above using the presented formalism. Choose properly the **nodes** and **channels**. Set the **contiguities** so that only **wired** channels can be used between **Z2** and **Z4** as well as between **Z4** and **Z6**.

Tasks

Identifier	Size	Mobile
T2, T4, T6	25	0
T1, T3, T5	5	1

Data-Flows

Data-Flow	Bandwidth	Max Delay	Max Energy
Df1, Df2	30	10	10
Df3, Df4, Df5	10	5	5

5.3 Exercise 3

In the second exercise your are required to write an objective function which:

- minimizes the energy consumption of nodes and channels, but not of tasks and data-flows inside them;
- which divides by 2 the energy consumption of non-mobiles nodes.

5.4 Exercise 4

For the last exercise you are required to:

- modify the `preprocess` phase implemented in `NetworkInstance.py` which you can find inside `networklib`, so that mobile nodes could host non-mobile tasks;
- modify the constraints in order to prevent mobile and non-mobile tasks to be placed inside the same mobile node.

That's all folks