Project No. 608930

FP7-SMARTCITIES-2013

OPtimising Hybrid Energy grids

for smart citieS

WP3: Monitoring and System Analysis

Deliverable D3.1.3

Recommendations for ICT M2M Infrastructure for the evolution towards Cooperative Hybrid Energy Grids integrated in Smart Cities Operation Centre

Apostolos Papageorgiou[1], Christer Åhlund[2], Karan Mitra[2], Saguna Saguna[2]

[1]NEC Europe Ltd., [2]LTU

Version:	1.0
Due Date:	31 Oct 2016
Submission Date:	31 Oct 2016
Lead Coordinator:	NEC

Copyright

Consisting of

Coordinator	WIP – Renewable Energies (WIP)	Germany
Participants	Hochschule Ulm (HSU)	Germany
	Stadtwerke Ulm GmbH (SWU)	Germany
	Technische Universität Wien (TUW-EEG)	Austria
	Austrian Institute of Technology (AIT)	Austria
	Deutsches Zentrum für Luft- und Raumfahrt EV (DLR)	Germany
	NEC Europe Ltd. (NEC)	UK
	Lulea Tekniska Universitet (LTU)	Sweden
	Skelleftea Kraft (SKR)	Sweden

Deliverable Description

Abstract: This deliverable (D3.1.3) provides ICT-related recommendations for future energy monitoring and control systems, as well as their integration into more widely scoped Smart City systems and Smart City Control Centers. In line with the investigations around “horizontal M2M architectures” and “M2M data filtering”, which were in the core of Deliverables D3.1.1 and D3.1.2, respectively, the recommendations of D3.1.3 go in three main directions. **Firstly,** *we provide an analysis of “further” ICT/M2M or Smart City-related data sources that could be used for energy control optimization.* **Secondly,** *we discuss business and technical aspects and provide recommendations for the integration of systems like OrPHEuS into Smart City Control Centers.* **Thirdly,** *we describe in detail a recommended solution for supporting actuation and low-latency interactions in data streaming systems such as the OrPHEuS monitoring system.*

Key Words: ICT, recommendations, smart cities, Smart City Control Center, stream processing, low latency, smart grid, energy control, monitoring, sensor communication

Document History

Version	Date	Authors	Description
0.1	2016-03-15	NEC	Draft outline
0.2	2016-04-11	NEC	Added Section 2 (Smart City data sources) and more
0.3	2016-05-17	NEC	Added Section 4 (Recommended Stream Processing Ext.)
0.4	2016-06-24	LTU	Added contents in Section 3 (SCCC Recommendations)
0.5	2016-06-27	LTU	Extended contents in Section 3 (SCCC Recommendations)
0.6	2016-06-29	NEC	Updated intro, abstract, and more; added placeholders
0.7	2016-09-20	LTU	Updates in Section 3 (SCCC Recommendations)
0.8	2016-09-20	LTU	Updates in Section 3 (SCCC Recommendations)
0.9	2016-10-04	NEC	Revision, contents merging, tables and lists, etc.
1.0	2016-10-28	NEC	Clean final version with submission dates
1.1	2016-10-31	WIP	Review of final version

Dissemination Level

Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the Consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Executive Summary

The OrPHEuS project elaborates a Hybrid Energy Network Control System for Smart Cities implementing novel cooperative local grid and inter-grid control strategies for the optimal interactions between multiple energy grids by enabling simultaneous optimization for individual response requirements, energy efficiencies and energy savings as well as coupled operational, economic and social impacts. Starting from existing system setups in two cities, enhanced operational scenarios are demonstrated for today's market setup, as well as for future market visions.

This Deliverable provides recommendations related to the ICT infrastructure towards a more complete and more efficient energy control system, which can also be integrated with other modern Smart City ICT systems. To this end, it provides an analysis of ICT/M2M or Smart City-related data sources that could be used in addition to currently captured energy grid data in order to further enhance energy control optimization. Further, based on data from the demo site of Skelleftea, this work goes into detailed recommendations about the data can be pre-aggregated by a Smart City Control Center (SCCC) before it is fed into the energy control system. This can be seen as a domain-specific way of applying data filtering mechanisms such as those that were contributed in previous phases of the project. Finally, solution for supporting actuation and low-latency interactions in data streaming systems is described in detail, along with explanations why this capability is recommended for future versions of ICT systems that support energy control.

Administrative Overview

Task Description

Task 3.1 is investigating "*approaches for monitoring, managing, filtering, and disseminating data with M2M (Machine-to-Machine) systems*". The third phase of Task 3.1 (described in the current Deliverable) focuses on ICT-related recommendations for M2M systems that serve -among others- energy networks and energy optimization systems. These recommendations go beyond the traditional scope of energy-network ICT systems, discussing their role and their potential enhancements in the context of more widely integrated Smart City systems.

Relation to the Scientific and Technological Objectives

This task is related to STO2, addressing the targets of:

a) "*Integration of existing independent energy grid ICT systems as subsystems for future Smart City Operation Centers for the energy domain*", by focusing mainly on the:
b) "*Utilization, development of extensions and customization of Machine-to-Machine (M2M) infrastructure*"

With regard to a), further progress has been achieved by explicitly providing recommendations the integration of systems like OrPHEuS into the context of Smart City systems. This covered both business and technical aspects.

With regard to b), an analysis of various data sources has been provided, which have typically not been used for the optimization of energy control in the past. The potential of their use has been explained. Further, a concrete technical solution for achieving low-latency interactions and meaningful actuation has been described. This solution is recommended for M2M data stream processing systems such as the OrPHEuS monitoring system or any similar ICT system that supports energy control.

The current Deliverable is directly linked with the following Performance Indicator:

No.	Objective/expected result	Indicator name	STO	Deliverable	MS	Expected Progress		
						Year 1	Year 2	Year 3
21	Recommendations for ICT M2M Infrastructure	Future Cases	STO2	D3.1.3	MS3			Due: M30 Draft: M27

Relations to activities in the Project

This work builds on insights of previous WP3 results in order to provide ICT recommendations. The analysis of "additional (Smart-City-related) data sources" and the discussion of their potential use for energy control have been based on intensive cooperation with energy control experts of WP5. Finally, some aspects of the Smart City Control Center integration were partly inspired by WP5 works on visualization tools and WP6 experiences from the demo sites.

Terminologies

Abbreviations[1]

CPU	Central Processing Unit
DB	Database
GW	Gateway
ICT	Information and Communications Technology
IFC	Industry Foundation Classes
M2M	Machine-to-Machine
MB	Megabyte
MS	Milestone
NW	Network
SCCC	Smart City Control Centre
SOC	State Of Charge
SPF	Stream Processing Framework
STO	Scientific & Technological Objective
WP	Work Package

[1] Not including the modules of the SPF extensions in Section 4, which are used only locally and are rather module names than abbreviations.

Table of Contents

1 Introduction ... 8
2 Recommendations for involving additional Smart City data sources in energy control ... 9
2.1 Nature and scope of additional Smart City data sources ... 9
2.2 List and evaluation of potential Smart City data sources ... 9
3 Recommendations for the Smart City business models ... 12
3.1 A Smart City control center ... 12
3.2 Data management ... 12
4 Recommendations for actuation-readiness and low-latency interactions ... 20
4.1 Low-latency requirements in energy control ICT systems ... 20
4.2 Stream Processing and its limitations ... 22
4.3 Design of a solution for achieving low-latency interactions in Stream Processing systems ... 24
4.3.1 Architecture of the recommended system extensions ... 24
4.3.2 Theory of Operation ... 26
5 Conclusion ... 28
6 Bibliography ... 29
7 Disclaimer ... 30
8 Contacts ... 31
Appendix A. Guidelines for Prototypical Implementation of the Recommended Stream Processing Framework Extensions based on Apache Storm ... 32

1 Introduction

Although the OrPHEuS ICT/M2M infrastructure can serve the purpose of collecting and managing the data that are required for optimizing the control of hybrid grids, the OrPHEuS project has shed light on various aspects of the ICT system that could be enhanced or extended in order to:

- Perform more efficient energy control by including additional Smart City-related data sources and feeding into the energy control software.
- Be smoothly integrated with Smart City Control Centers, and benefit from the extensive data management capabilities of the latter.
- Address situations in which fast actuation or other ultra-low latency interactions are required, e.g., in Smart Grid power supply synchronization.

Through different phases of the project, we have identified and/or designed additional data management capabilities, as well as platform extensions, which could address the aforementioned limitations of most energy control ICT systems. Sections 2, 3, 4 contain our technical recommendations towards addressing the three listed limitations, respectively.

2 Recommendations for involving additional Smart City data sources in energy control

This section discusses the potential of assisting and enhancing energy control strategies by using additional Smart City information, i.e., information that cannot be provided by the ICT infrastructure of energy networks, but might be made available through the integration of the smart energy systems into the broader context of Smart City systems.

2.1 Nature and scope of additional Smart City data sources

Smart City platforms are expected to connect (or even completely unify) the underlying ICT M2M infrastructure of different Smart City vertical industries, such as e-health, smart homes, logistics, safety, smart buildings and construction, and more. The extent and the purpose of this “horizontal” M2M platform design have been discussed in more detail in Deliverable D3.1.1. Some of the main advantages are related to reduced start off costs, hardware and software reusability, increased interoperability, unified control, and more.

Furthermore, a secondary advantage is that vertical applications (e.g., smart energy applications) can get easier and more intuitive access to data sources which are traditionally used mainly by other verticals and their integration has not been worth the costs or the effort (please refer also to the “horizontal” vs “vertical” platform discussion and the related figures of Deliverable D3.1.1).

With smart energy systems becoming part of Smart City platforms, it is a worthwhile goal to consider which data sources of other verticals the smart energy system could potentially make use of, in order to enhance its performance, e.g., by enhancing its predictions about energy consumption.

The identification of Smart City data sources with such potential has been performed through workshops with the participation and discussions between Smart City experts and Energy Control experts, from Work Packages WP3 and WP5, respectively. The potential data sources and the reasoning about their possible usage are listed in the following subsection. Integration or tests with such data sources goes beyond the scope of the OrPHEuS project. Further, taking such additional data into account could be sometimes impeded by users’ privacy considerations, but this is also a different research topic (which is out of scope for this initial analysis).

2.2 List and evaluation of potential Smart City data sources

The following table summarizes the results of the aforementioned workshops and analyses. The first column lists the potential additional Smart City data sources, while the second column describes the idea behind integrating such information into the lifecycle of energy control systems, and the third column provides comments about the feasibility of the implementation of the idea as well as further comments and examples related to the OrPHEuS project. General expectations of the approach (i.e., independently of the OrPHEuS demo sites and scenarios) are also listed.

Smart City data source	Reasoning of usage	Feasibility and expectations
City construction information	Areas with intense public works planned for the future might show a big change in demand (for the construction period, e.g., if a building is closed, or even for the demand pattern afterwards).	This is a promising source but implementation might be tricky if the planning horizon of the energy control system concerns time periods for which it is impossible to get detailed construction information on time.
IFC model instances of buildings, esp. building thermal data	Buildings thermal data might help to know energy losses etc., so that the accuracy of building modelling in simulation environments can be enhanced.	Communities (esp. for the Skelleftea site in Sweden) would be willing to provide relevant information if people of the project go to cooperate with them. Further, the age of building blocks, the type of constructions, or even more fine granular information, from the IFC instances could be used there.
Crowd detection (e.g., based on patterns or event schedule)	Especially for sports or concert facilities, the demand on days for which “big crowds” are predicted might deviate significantly from their regular pattern. Knowledge of such events could enhance demand prediction.	There is already a system called CityFlow (in place in Skelleftea almost two years now) which performs WiFi-based crowd detection and mobility pattern analysis. Using the data available from this Smart City system as input to control strategies could help.
Customers feedback	Platforms that interconnect energy consumers (esp. of households) could enable individual users to provide their consumption plans (implicitly, e.g., by indicating absences, or explicitly, e.g., by stating that they will need high amounts of energy on specific days).	A possible implementation could involve applications with which big customers controlled by the same substation could communicate and coordinate peak load “handling”. It could also be examined if –in addition to the big customers– some families or neighbors that are equipped with an accompanying (e.g., mobile) application can contribute to the respective planning. However, households are more likely to see their measurements as private data and therefore not participate.
Household temperatures	Currently used temperature information comes only from weather forecasts or external	This is not really monitored at the moment and it does not look like a very feasible solution

	sensors. Detailed knowledge about the temperatures of households could enhance planning, especially by using pre-heating in order to achieve peak shaving, whenever a peak demand is predicted for the near future.	for energy providers. However, if the information exists for other Smart City applications, then they could become an important element of demand prediction and analysis, esp. if it is combined with appropriate machine "learning" methods. One problem can be that the households that are being monitored are normally not even aware that they are monitored, because it is done at the substation level.
Users' temperature settings (desired ranges)	A platform that connects energy consumers to the supplier could enable a business model in which user-specified desired temperature ranges (for a household) are flexibly agreed. Further, the knowledge about them can help the supplier shave peak demands and still satisfy user requirements	Everyone has a different "level of demand" and it could be useful to capture it in more detail. Although this sounds specific to the energy vertical industry, it is not likely to be enabled unless other Smart City applications are involved, as explained also in the previous point.
Profiles of use of domestic hot water per household	Being able to predict hot water usage improves heat demand prediction.	The time resolution is normally enough to catch the peaks, because it is feasible to "capture" if someone is taking a shower, for example (per minute basis). It is however, again, difficult to find a feasible and non-costly way to convince customers to "export" such data.
Monitoring of prosumers' electricity and thermal storage (State of Charge - SOC)	Assuming the existence of energy storage (batteries or thermal storage) at the customer premises, centralized control can manage charging and discharging more efficiently if it is provided with enough monitoring information such as SOC, total capacity, power rates etc.	In OrPHEuS, this would be more relevant for the Ulm scenario, because of the requirement for active prosumers. However, even in Ulm there is no electricity storage, but if there were, information about the State-Of-Charge would be extremely useful for energy control. Information about the distributed thermal storage in Skelleftea would be something different, but also useful.

3 Recommendations for the Smart City business models

The Orpheus project contributes to a Smart City business model by providing information and control parameters of how to optimize heterogeneous energy grids for smart cities, being an important factor for an overall city control strategy. Based on this aspect, we are considering a Smart City Control Centre (SCCC) combining and filtering data and information from different sources, to optimize the control of a city. Stakeholders can then define business models using information from one subsystem to be used in another. Data that is made available can also be used to create new services combining information from different sources, creating open data initiatives.

3.1 A Smart City control center

A Smart City Control Centre (SCCC) combines and filters data and information from heterogeneous sources, such as smart buildings automation systems to optimize the control of a city. In the Orpheus project, we focus on the smart hybrid energy grids in the context of a smart city. Here the SCCC samples operational parameters and related information, to make the intelligent decisions regarding the efficient operation of smart energy grids. Stakeholders (such as the city council), other than the smart energy grid operator can utilize this information for other city services that maximizes the control of a city. Similarly, the information originating from other systems (for example, weather monitoring systems) within a city connected via the SCCC can be used for improving the operation of energy systems. The distribution of information or simply "data" between smart city services enables new business opportunities for stakeholders.

3.2 Data management

The proposed M2M gateway architecture (mentioned in the subsequent sub-sections) provides functionalities that allow data gathering and filtering for the optimization of smart energy grids. The M2M gateway uses metrics and algorithms for data filtering based on time and value. The sampled and the pre-processed data then provided to the SCCC to be shared across several sub-systems. The communication between the substations/components in the energy grids and the SCCC is adaptive and adjustable according to current requirements.

In the case of high heat and electricity demands, and varying consumption-loads, updates (between the systems) need to be sent more frequently, than during lower demand conditions and stable consumption patterns. The digitalization of future cities also includes monitoring of societal infrastructures causing a challenge with data heterogeneity that can be managed by data transformation and filter to provide homogeneous and interchangeable datasets. The M2M gateway solution, therefore, provides new business opportunities in the vision of an SCCC combining subsystem and sharing data between them.

We now present an example pertaining to the consumption side of the district-heating grid in the Skellefteå scenario. Figure 1(a) shows the control loop of the district-heating grid. In this grid, a number of sensors are placed at the substation as shown in Figure 1 (b). These sensors produce data that is continuously (at 1-minute time interval) transmitted to the SCCC. It is important to note here that the algorithms used for heat load prediction do not require per-minute values but instead need data at higher time intervals, such as, at every fifteen to thirty minutes, or hourly values ranging from

1 to 48 hours. Therefore, necessitating data aggregation and pre-processing at the SCCC to enable input data in the correct time intervals for the algorithms. Figure 2 shows a 30-minute snapshot of per-minute values originating from sensors at one of the substations in Skellefteå. Further, meteorological data (cf. Figure 3) also collected from the local weather stations, as well as the weather sensors installed outside the substations buildings. All this collected data is pre-processed within the SCCC.

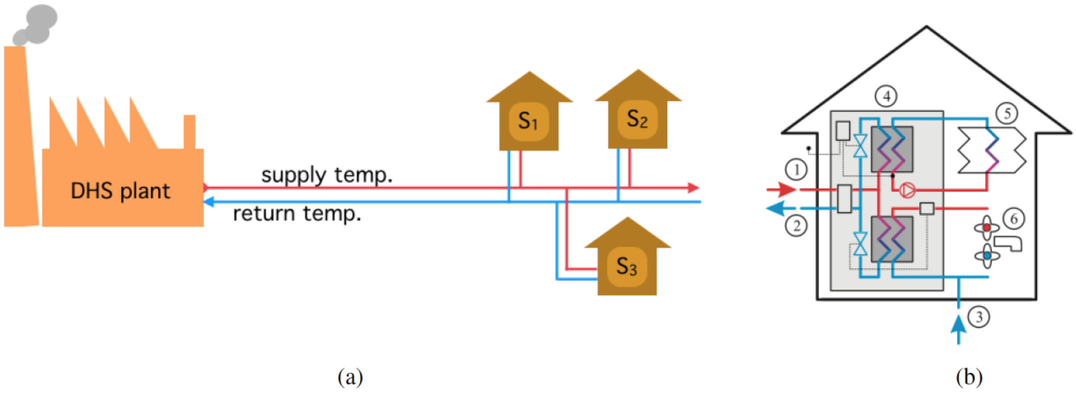


Figure 1. (a) A schematic diagram showing a district heating network and its plant. Here S1, S2, and S3 depict the substations. (b) Schematic illustration of a building with a district heating substation. Indicated in the figure are the: (1) primary water supply; (2) primary return water; (3) tap water supply; district heating substation including heat exchangers, (4) electronic energy meter and control system with related sensors; (5) heating system; and (6) tap water.

Meteorological Factors
- Outdoor Temperature (T_{out})
- Humidity
- Solar radiation
- Wind

Figure 2. Meteorological data parameters which can be used for heat demand prediction.

In a heat-demand prediction system developed by LTU ([5] [6] and [8]), within a district heating grid, such pre-processing of data is done within the data aggregator and data-preprocessor blocks that lies in the SCCC as shown in Figure 4. The data aggregation involves merging sources of data from each substation with its locally collected weather information. Also, the necessary time information is merged with the corresponding data instances. The data-preprocessing block at the SCCC converts the original sampling interval to a forecast target range. Our recent work in [6] studied varying forecast ranges, for example, 15 minutes to hourly ranges. An essential part of the data pre-processing block is data transformation that outputs the predictor's input variables and their corresponding target variables. The transformation is based on the intended forecast horizon. To be more precise, a transformation required for a 24-hour horizon is different from that required for a 6-hour horizon. The process of data aggregation and transformation is complex and may not be scalable if performed at SCCC. This is due to the fact that there may be several hundreds of substations present in a smart city. For example, there are approximately, 7000 substations in Skellefteå demonstration site.

To mitigate this challenge, we propose that this data is aggregated and pre-processed at the M2M gateways for low latency computation. The pre-processed data can then be offloaded to the SCCC for prediction tasks. This is critical for running a fast and efficient control loop in a district-heating grid. This will shorten the control loop in a district heating grid and enable faster decisions. This also directly impacts the production process, as it will help in reducing carbon emissions and the elimination of fossil fuels during peak demands.

Datum	Tid	Effekt (W)	Flöde (l/s)	Framl. (°C)	Retur (°C)	Utetemp (°C)
Date	Time	Value	Value	Value	Value	Value
23/06/14	00:00:58	41 400,00	0,22	69,81	33,39	10,31
23/06/14	00:01:58	59 000,00	0,27	69,83	31,91	10,29
23/06/14	00:02:58	41 600,00	0,31	69,84	23,89	10,26
23/06/14	00:03:58	37 000,00	0,25	69,84	30,94	10,20
23/06/14	00:04:58	36 900,00	0,23	69,86	32,09	10,17
23/06/14	00:05:58	37 100,00	0,23	69,87	30,98	10,13
23/06/14	00:06:58	33 500,00	0,23	69,87	30,99	10,10
23/06/14	00:07:58	31 200,00	0,22	69,88	33,39	10,09
23/06/14	00:08:58	32 100,00	0,21	69,88	34,13	10,08
23/06/14	00:09:58	31 800,00	0,21	69,87	33,04	10,07
23/06/14	00:10:58	31 400,00	0,21	69,87	33,49	10,09
23/06/14	00:11:58	32 700,00	0,21	69,86	33,95	10,08
23/06/14	00:12:58	31 000,00	0,22	69,86	34,11	10,07
23/06/14	00:13:58	31 100,00	0,21	69,85	34,23	10,06
23/06/14	00:14:58	31 200,00	0,21	69,81	34,16	10,04
23/06/14	00:15:58	31 200,00	0,21	69,80	34,05	10,03
23/06/14	00:16:58	32 600,00	0,21	69,79	34,04	10,00
23/06/14	00:17:58	40 700,00	0,22	69,77	34,06	9,98
23/06/14	00:18:58	47 600,00	0,26	69,77	28,16	9,97
23/06/14	00:19:58	43 200,00	0,28	69,76	27,12	9,96
23/06/14	00:20:58	38 000,00	0,25	69,76	28,89	9,96
23/06/14	00:21:58	36 600,00	0,24	69,77	31,17	9,95
23/06/14	00:22:58	34 200,00	0,23	69,73	31,49	9,93
23/06/14	00:23:58	31 400,00	0,22	69,74	32,82	9,94
23/06/14	00:24:58	32 900,00	0,21	69,72	33,79	9,92
23/06/14	00:25:58	34 700,00	0,22	69,71	32,31	9,90
23/06/14	00:26:58	31 300,00	0,22	69,73	32,21	9,88
23/06/14	00:27:58	30 900,00	0,21	69,72	34,03	9,83
23/06/14	00:28:58	32 500,00	0,21	69,70	34,18	9,82
23/06/14	00:29:58	35 300,00	0,22	69,68	34,02	9,80
23/06/14	00:30:58	51 500,00	0,23	69,68	32,61	9,77

Figure 3. Substation data showing, energy consumption, flow rate, incoming water temperature, return temperature and outside temperature. These values are collected at the substation by Schneider and transmitted to servers.

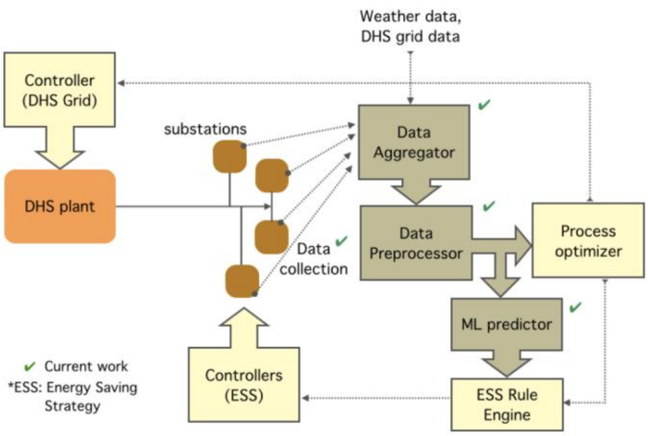


Figure 4. A high-level architecture showing the control loop in a district heating grid where timely processing of substation data helps in the prediction of heat demand and production. The M2M system lies within the data pre-processor block.

Similarly, at the production side of the district-heating grid, there are hundreds of sensors and data from them is collected at per minute time interval. For example, in the Orpheus project there are about 80 parameters provided by Skellefteå Kraft for simulations and results analysis from one of the main plant sites in Skellefteå city. This forms only a small part of all the functioning and operation of the Skellefteå demonstration site. There is further data that is collected from the different sites located both in and around Skellefteå. Skellefteå Kraft also collects data from its number of hydro and wind stations. If we take into account the 80-parameter values for one-month data there are 44,640 records (approximately 31.6 MB). We show some details and snapshots of this data in Figure 5 - Figure 19.

Acc cistern H-byn (MW)
Oil boiler 12 MW "Dalen"
El. boiler 24 MW "Dalen"
Oil boiler 25 MW "Lasarettet"
Oil boiler 12 MW "Kuggstången"
Oil boiler 12 MW "Skruven" ERROR
Oil boiler 12 MW "H21"
Oil boiler 12 MW "H22"

Figure 5. List of units in the district heating grid from where 'heat to grid' related parameters are collected, these include oil boilers, electric boilers at different locations and the accumulator at the main site in Skellefteå.

	Acc cistern H-byn (MW)	Oil boiler 12 MW "Dalen"	El. boiler 24 MW "Dalen"	Oil boiler 25 MW "Lasarettet"	Oil boiler 12 MW "Kuggstången"	Oil boiler 12 MW "Skruven"	Oil boiler 12 MW "H21"	Oil boiler 12 MW "H22"
2014/08/01 00:00	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:01	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:02	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:03	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:04	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:05	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:06	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:07	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:08	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:09	8,094406128	0	0	0,017152824	0	0	0	0
2014/08/01 00:10	8,094406128	0	0	0,017152824	0	0	0	0

Figure 6. Dataset showing 'heat to grid' per-minute values collected from the different boiler units and accumulator listed in Figure 5.

	CFB 92 MW CHP H2 (MW)	FB H1 25 MW (MW)	Oil boiler 12 MW H10 (MW)	Oil boiler 12 MW "Dalen"	El. boiler 24 MW "Dalen"	Oil boiler 25 MW "Lasarettet"	Oil boiler 12 MW "Kuggstången"	Oil boiler 12 MW "Skruven"	Oil boiler 12 MW "H21"	Oil boiler 12 MW "H22"
2014/08/01 00:00	-0,070948094	7,980118275	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:01	-0,070948094	8,035727452	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:02	-0,070948094	8,124016814	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:03	-0,070948094	8,203892708	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:04	-0,070948094	8,129849566	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:05	-0,070948094	8,141269283	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:06	-0,070948094	8,073123692	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:07	-0,070948094	8,080656943	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:08	-0,070948094	8,13891581	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:09	-0,070948094	8,005936818	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00
2014/08/01 00:10	-0,070948094	8,032308274	0,018309429	0	0	0,017152824	0	0,00	0,00	0,00

Figure 7. Dataset showing 'heat production' per minute values collected from different boiler units and accumulator listed in Figure 5.

Heat load to grid (MW)
Heat load In/Out (MW)
Temp. to grid (Deg. C)
Temp. from grid (Deg. C)
Outside temp (Deg. C)
Massflow, distr.pump (kg/S)
dP distr. Pump (Bar)
Energy in acc (MWh)
Acc 47,65 meter
Acc 45,3 meter
Acc 42,05 meter
Acc 40,6 meter
Acc 38,25 meter
Acc 35,9 meter
Acc 33,55 meter
Acc 31,2 meter
Acc 28,85 meter
Acc 26,5 meter
Acc 24,15 meter
Acc 21,8 meter
Acc 19,45 meter
Acc 17,1 meter
Acc 14,75 meter
Acc 12,4 meter
Acc 10,05 meter
Acc 7,7 meter
Acc 5,35 meter
Acc 3 meter
Acc 0,65 meter

Figure 8. List of parameters relating to the heat load from the main demonstration site at Hedensbyn to the grid including the different temperature at different levels in the accumulator.

	Heat load to grid (MW)	**Heat load In/Out (MW)**	Temp. to grid (Deg. C)	Temp. from grid (Deg. C)	**Outside temp (Deg. C)**	Massflow, distr.pump (kg/S)	dP distr. Pump (Bar)
2014/08/01 00:00	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	71,79445988	3,285932064
2014/08/01 00:01	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	68,75762177	3,285932064
2014/08/01 00:02	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	68,04548726	3,285932064
2014/08/01 00:03	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	67,17738647	3,285932064
2014/08/01 00:04	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	68,91380787	3,285932064
2014/08/01 00:05	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	66,27063369	3,285932064
2014/08/01 00:06	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	69,13448966	3,285932064
2014/08/01 00:07	8,094406128	1,589896321	74,29050446	47,00945663	15,90478897	65,73079681	3,285932064
2014/08/01 00:08	8,094406128	1,589896321	73,80373991	47,00945663	15,90478897	68,09280118	3,285932064
2014/08/01 00:09	8,094406128	1,589896321	73,64967346	47,00945663	15,90478897	66,62761688	3,285932064
2014/08/01 00:10	8,094406128	1,589896321	73,64967346	47,00945663	15,90478897	66,62761688	3,285932064

Figure 9. Dataset per minute values relating to parameters listed out in Figure 8.

Power load CHP 34 MW tubine, gros (MW)
Power load pellet 5 MW turbine, gros (MW)

Figure 10. 'Power to grid' data parameters recorded at Skellefteå demonstration site.

	Power load CHP 34 MW tubine, gros (MW)	Power load pellet 5 MW turbine, gros (MW)
2014/08/01 00:00	-0,030515715	0,076290131
2014/08/01 00:01	-0,030515715	0,076290131
2014/08/01 00:02	-0,030515715	0,076290131
2014/08/01 00:03	-0,030515715	0,076290131
2014/08/01 00:04	-0,030515715	0,076290131
2014/08/01 00:05	-0,030515715	0,076290131
2014/08/01 00:06	-0,030515715	0,076290131
2014/08/01 00:07	-0,030515715	0,076290131
2014/08/01 00:08	-0,030515715	0,076290131
2014/08/01 00:09	-0,030515715	0,076290131
2014/08/01 00:10	-0,030515715	0,076290131

Figure 11. 'Power to grid' are also collected at per minute time interval.

Tot thermal load (MW)
Heat load to acc/grid (MW)
Power load CHP 34 MW tubine, gros (MW)
Steam pressure CFB (Bar)
Steam temp. CFB (Deg C)
Steam massflow tot. CFB (kg/S)
Fuel flow 1(2) CFB (kg/S)
Fuel flow 2(2) CFB (kg/S)
Fluegas CFB O2 (%)
Fluegas CFB CO (ppm)
Fluegas CFB NOx (ppm)
Fluegas CFB SO2 (ppm)

Figure 12. CFB/gas emission, fuel flow and steam related parameters H2 site.

	Tot thermal load (MW)	Heat load to acc/grid (MW)	Power load CHP,gros (MW)	Steam pressure CFB (Bar)	Steam temp. CFB (Deg C)	Steam massflow tot. CFB (kg/S)	Fuel flow 1(2) CFB (kg/S)	Fuel flow 2(2) CFB (kg/S)	Fluegas CFB O2 (%)	Fluegas CFB CO (ppm)	Fluegas CFB NO
2014/08/01 00:00	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:01	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:02	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:03	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:04	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:05	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:06	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:07	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:08	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:09	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	
2014/08/01 00:10	0	-0,070948094	-0,030515715	0,274641424	300,3661804	0,02243845	0,010527922	0,042111687	15	-5,231260777	

Figure 13. Snapshot of per minute data values for CFB/gas emission, fuel flow and steam related parameters H2 site.

Heat load to acc/grid (MW)
Boiler pressure (Bar)
Fuel flow (kg/S)
Fluegas O2 (%)
Fluegas CO (ppm)
Fluegas NOx (ppm)
Fluegas SO2 (ppm)

Figure 14. BFB/gas emission, fuel flow parameters H1 site.

	Heat load to acc/grid (MW)	Boiler pressure (Bar)	Fuel flow (kg/S)	Fluegas O2 (%)	Fluegas CO (ppm)	Fluegas NOx (ppm)	Fluegas SO2 (ppm)
2014/08/01 00:00	7,980118275	9,086359024	0,702865746	3,854491151	178,5148885	84,16326505	0,762892842
2014/08/01 00:01	8,035727452	9,003978503	0,826515967	4,720015501	215,7003281	81,58502847	0,762892842
2014/08/01 00:02	8,124016814	9,017950682	0,835056684	4,714289159	170,7826205	87,06843472	0,762892842
2014/08/01 00:03	8,203892708	9,086359024	0,767265193	3,92663896	159,9023438	90,73256359	0,762892842
2014/08/01 00:04	8,129849566	9,049630177	0,779908472	3,715185987	265,2846941	87,52402452	0,762892842
2014/08/01 00:05	8,141269283	9,00335598	0,828242783	3,929566864	277,1192909	83,46743114	0,762892842
2014/08/01 00:06	8,073123692	9,046493201	0,750880347	3,405104546	241,9273569	80,66363353	0,762892842
2014/08/01 00:07	8,080656943	8,617905169	1,183875459	4,369429599	224,8875883	80,12504205	0,762892842
2014/08/01 00:08	8,13891581	9,012999125	0,821060066	2,785250038	208,789531	80,35897092	0,762892842
2014/08/01 00:09	8,005936818	9,076581793	0,759990708	2,910911199	1057,424794	70,22574698	2,847421297
2014/08/01 00:10	8,032308274	8,876410484	0,974312083	4,388169267	714,9658636	63,29372113	3,509307146

Figure 15. Snapshot of per minute data values for BFB/gas emission, fuel flow parameters H1 site.

Pellet production (Ton/h)
Power load pellet 5 MW turbine
Moisture Raw material (%)
Moisture Dryed material (%)
Moisture Pellet (%)

Figure 16. Biofuel pellet production related parameters.

	Power load turbine (MW)	**Pellet production (Ton/h)**	Moisture Raw material (%)	Moisture dryed material (%)	Moisture Pellet (%)
2014/08/01 00:00	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:01	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:02	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:03	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:04	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:05	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:06	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:07	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:08	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:09	0,08	**0,00**	30,00	25,00	7,05
2014/08/01 00:10	0,08	**0,00**	30,00	25,00	7,05

Figure 17. Snapshot of per minute data values collected for Biofuel pellet production process.

Temp inlet turbine (deg. C)
Massflow inlet turbine (kg/s)
Pressure inlet turbine (bar)
Non of the requested available
Non of the requested available
Massflow steam to feedwater cistern(kg/s)
Temp steam to feedwater cistern(deg. C)
Pressure in feedwater cistern (bar)
Pressure in crossover to LP turbine (bar)
Pressure inlet LP turbine (bar)
Temp steam to pellet plant, H5 (deg. C)
Massflow steam to pellet plant, H5 (kg/s)
Pressure steam to pellet plant, H5 (bar)
Non of the requested available
Temp inlet Heat condenser 2 (deg.C)
Pressure inlet Heat condenser 2 (bar Abs)
Temp inlet Heat condenser 1 (deg.C)
Pressure inlet Heat condenser 1 (bar Abs)
Non of the requested available
Non of the requested available
Non of the requested available
Non of the requested available
Non of the requested available
Temp outlet LP preheater (deg.C)
Pressure Feedwater cistern (bar)
Temp feedwater inlet LP preheater (deg.C)
Pressure outlet feedwaterpump 1 (bar)
Pressure outlet feedwaterpump 2 (bar)
Temp feedwater outlet LP preheater (deg.C)
Non of the requested available
Temp feedwater outlet HP preheater (deg.C)
Pressure feedwater to boiler (bar)
Massflow feedwater (kg/s)
Pressure condensate outlet H5 (bar)
Temp condensate outlet H5 (deg.C)
Massflow condensate outlet H5 kg/s)
Non of the requested available
Sheet CFB 92 MW CHP "H2"
Sheet Biofuel pellet production "H5"
Temp dh inlet Heat condenser 1 (deg.C)
Massflow dh H2 (kg/s) condenser 1+2 in serial
Temp dh outlet Heat condenser 1 (deg.C)
Temp dh outlet Heat condenser 2 (deg.C)

Figure 18. List of parameters relating to bioenergy combine at Skellefteå demonstration site.

	Temp inlet turbine (deg. C)	Massflow inlet turbine (kg/s)	Pressure inlet turbine (bar)	Massflow steam to feedwater cistern(kg/s)	Temp steam to feedwater cistern(deg. C)
2014/08/01 00:00	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:01	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:02	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:03	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:04	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:05	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:06	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:07	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:08	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:09	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057
2014/08/01 00:10	24,90082359	0,02243845	0,988709152	-0,001830943	23,34452057

Figure 19. A snapshot of some of the parameters from those listed in figure 18 also collected at per minute time interval.

This leads to large amount of data collection, transmission and processing at the plant side with hundreds of sensors. These data points are collected and stored in a central repository where the aggregation and pre-processing is done before running any further analysis. A system to filter out data and its pre-processing is essential since for example in the Orpheus project many of the simulations required 15-minute time interval values instead of per minute values. In some cases, 30-minute time interval values were needed. For many of Skellefteå Kraft's control operations these values can vary and also the point of processing could be located in different locations from where the data is sourced. The data is sourced from the various locations including for example, main plant site, the heat exchangers, the many different electric and oil boilers in different parts of the city and around the city. Many of the data is required for near real-time analytics and control operations where time delay can lead to delay in crucial decisions and losses. Thus, it is also suitable for the production side scenario, similar to the consumption side scenario that M2M gateways are placed at different locations, which can pre-process and aggregate data as and when required by the different systems that use this data for analysis to help stakeholders make efficient decisions.

The M2M gateways and other data filtering/pre-processing are not only relevant for smart grids alone but also for other smart city scenarios as well. The growth of M2M devices has led to the deployment of large number of sensors in smart cities, like Skellefteå, which are already being used for a variety of applications, and these applications are envisioned to grow manifold in the near future. These sensors produce large amounts of data, which leads to the problem of its processing, transmission and storage. Also, the data is in most cases never used in the form it is produced, thus M2M gateways are essential and relevant across application domains for filtering data and pre-processing it in the required ways by different applications. Further, the gateways could perform appropriate data transformations, e.g., transform to a standard output format (e.g. IEC61850), if the devices don’t use it already.

4 Recommendations for actuation-readiness and low-latency interactions

This section deals with an issue related to the placement of data streaming tasks inside the M2M platform. The issue appeared mainly because of the necessity to flexibly deploy data filtering functions on different places in the network and not only statically on GWs (cf. Deliverable D3.1.2), but it becomes even more interesting when the deployable tasks have actuation functions (i.e., they control certain machines, sensors, or devices) with low-latency requirements.

In the previous phase of our work on M2M platform enhancements we have focused on data filtering (normally “close to the data sources”), in order to save bandwidth, energy, and storage costs. After having developed data filtering solutions, the next question that naturally arises is “where exactly the data filtering handlers should be running”. Despite the assumption that data filtering takes place on a GW that directly communicates with the data collection/generation devices, the data filtering GW can actually be anywhere on the communication path from the edge (sensors and on-sites GWs) to the Cloud (data centers). It might often make sense to perform the data filtering even at the backend, e.g., if we only care about data storage.

Further, although we have not actively dealt with this scenario in the OrPHEuS demo sites, the ICT M2M platform should consider actuation Use Cases and provide design recommendations that can be followed in order to fulfill the requirements of such Use Cases.

The mechanism that is described in this chapter is a solution for flexibly deploying (aka placing and migrating) data stream processing tasks (such as those that we expect to find in a system like the OrPHEuS M2M platform) in a way that low-latency requirements of actuating tasks can be fulfilled. Before describing the solution, which is designed generically but also includes a prototype implementation based on the Apache Storm [3] stream processing engine, we briefly explain examples of low-latency requirements in energy control ICT systems (4.1) and the respective limitations (4.2) of Stream Processing systems (such as Apache Storm) when it comes to covering such requirements.

4.1 Low-latency requirements in energy control ICT systems

This section discusses actuation scenarios and relevant low-latency requirements based on an example processing topology.

Imagine a system such as the one depicted in Figure 20. The lower part of the figure (“Stream Processing Topology”) shows three data streaming tasks, which need to be executed, each of them operating constantly upon continuous input data items and making use of the output of the previous task. The tasks can be parallelized, thus each of the tasks can be instantiated many times (maybe even hundreds or thousands, depending on the extent of the system) and deployed on different compute nodes. The upper part of the figure (“Infrastructure and topology-external interactions”) shows the mentioned compute nodes (which are here either at the edge or in the Cloud), the interactions of the task with some external entities such as actuators or databases, and an important metric, which will be explained in the following.

In this example, meter-reading tasks (Task 1 in Figure 20) retrieve information from power sources and they forward them to Grid-synchronizing tasks (Task 2), which in turn send the detected failures to report-generating tasks (Task 3). Note that this is a very simplified topology (or just part of the system), since when having a grid-synchronizing task, the device needs different information than "standard" meters provide (e.g., detailed information of Phase- Angles and Amplitude) and not the exact amount of energy produced. In an inverter-based power plant all the measurements for a grid-synchronizing task are also relevant for the operation. Now looking into the topology-external part of this simplified topology, the Grid synchronizer must instantly adjust the appropriate power sources upon certain events (outages, partial discharges, lack of synchronization), while it also stores measured data into either (or both) of two databases (one at the network edge and one in the Cloud) depending on the system status.

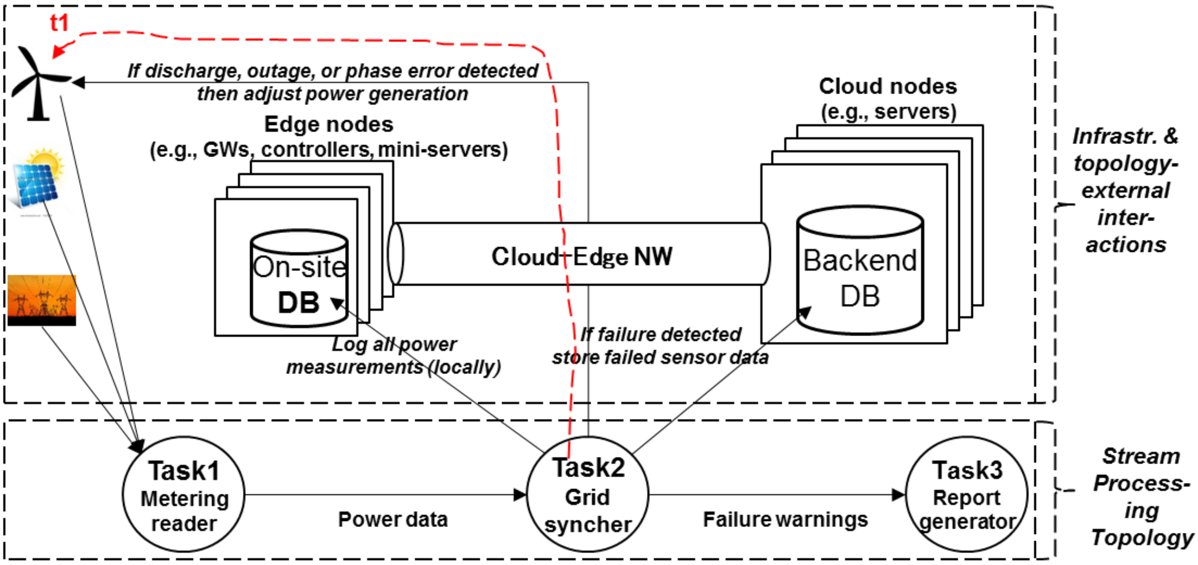


Figure 20. Example stream processing topology with low-latency requirements

Figure 20 also illustrates the critical actuation-related metric for this system: t_1 is the time from the moment a critical situation has been identified until the responding actuator(s) have been activated. As explained in [1] and [7] , this is necessary in cases of energy distribution automation or Smart Grid teleprotection, and it must be performed with ultra-low latency (often <10 milliseconds), otherwise huge costs can occur. For instance, [1] explains that if partial discharge is being monitored in a high voltage transformer, delayed detection of the situation could cause instability of the grid or even failure of the transformer if no necessary actions take place immediately. Similarly, [7] explains that specifically for wide area control and protection applications, the data are required to be transmitted to a control center, and control commands are required to be issued and implemented within a few milliseconds to prevent cascading outages in real-time.

Thus, section 4.2 explains why state-of-the-art stream processing systems cannot cope very well with such scenarios and section 4.3 describes our recommended solution, which we have also implemented prototypically as an extension of Apache Storm.

4.2 Stream Processing and its limitations

Stream Processing Frameworks (SPF) are software solutions that run on parallel networked systems in order to facilitate and regulate the execution of applications that consist of multiple data-intensive processing steps. These steps usually use the output of previous steps while they provide input to the next ones, so that the steps run sequentially (or according to a graph). The OrPHEuS data filtering handlers of Deliverable D3.1.2, as well as the steps of the topology of the previous section, are good examples of such processing steps.

SPFs have been developed and become widely used because of Big Data, i.e., data that is going into a system with very high incoming ratios or volumes and needs to be analyzed in various ways (or steps) "on the fly" before it is stored (or even without being stored). The systems on which SPFs run are traditionally server clusters, but they can be any set of networked devices, i.e., the devices that comprise the cluster might be heterogeneous and physically distributed. The traditional scenario of server clusters stems from the fact that most Big Data streams were coming from Web analytics applications, while the latter scenario of running SPFs on heterogeneous and/or geo-distributed clusters is now motivated by the huge data streams that can be produced and need to be dynamically analyzed in the Internet of Things.

Different SPFs, e.g., Apache Storm, S4, Spark, or Samza use different terminologies and slightly different architectures, but from a high-level perspective most of them operate as shown in Figure 21. Developers provide the sequence of the steps that needs to be executed, along with the implementation of each step and some required settings (e.g., desired number of instances for each step, desired number of used machines/devices etc.) to the SPF. Adhering to the terminology of Apache Storm, the sequence (or graph) of steps will be called "topology", the individual steps will be called "components", and the devices will be called "nodes". When the SPF has received the necessary input and the respective commands, it generates one or more instances of each component (called "tasks") and deploys them on nodes according to its internal logic, the settings, and/or the system state.

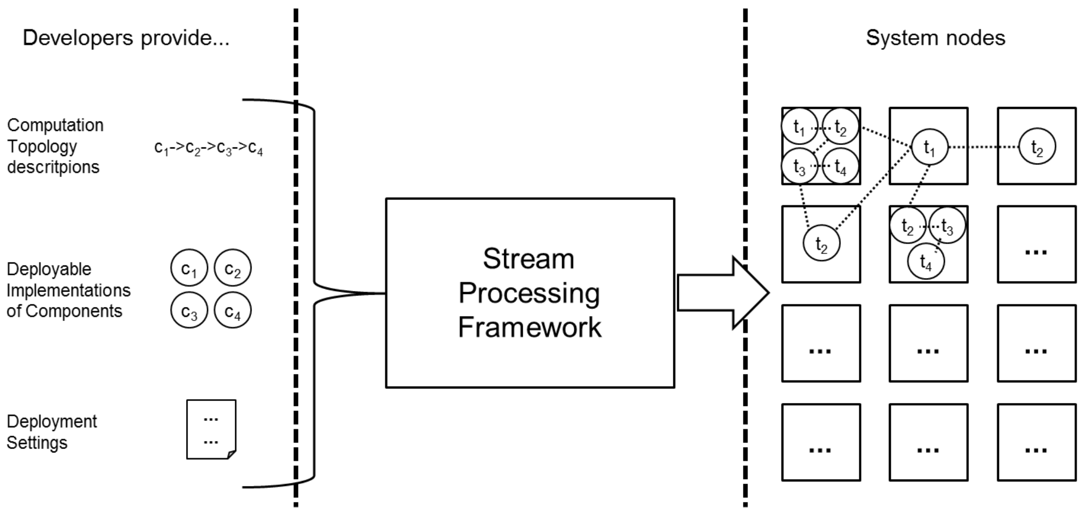


Figure 21. Basic SPF functionality in simple scenario with a server cluster

Because the requirements of heterogeneous and geo-distributed systems are different than those of server clusters, researchers and SPF developers have contributed systems and methods for extending SPFs in ways that best serve the "heterogeneous" scenario. More concretely, they have specified additional inputs (such as descriptions of network link or node capabilities), additional SPF modules (such as attached system monitors or sophisticated schedulers), and different server cluster types, as well as the algorithms that exploit these add-ons. Figure 22 visualizes these additions. For example, [1] and [4] describe extensions of Apache Storm for taking CPU and network load between the servers into account in order to rebalance the allocation of tasks to nodes, while [9] is a similar solution but based on design-time knowledge about nodes, as well as performance tests.

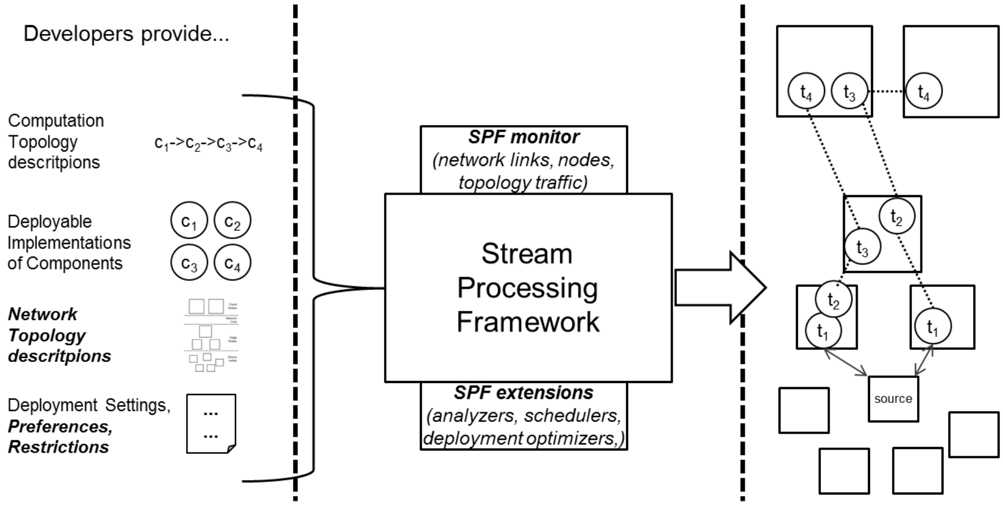


Figure 22. SPF functionality depicting extensions for supporting heterogeneous nodes (IoT scenario)

However, when the internal logic of the components or the usage of the system (not only in terms of load and performance but also with regard to functional aspects) require an application-specific deployment of the tasks, none of the mentioned solutions enables the SPF to deploy the tasks where it actually should deploy them. Think of the following examples for the components of a topology:

- Components activate actuators or raise alarms with unknown frequencies and varying latency requirements. For example, you might prefer to have a fault detection task which analyzes inputs from factory equipment running in the Cloud if you just want to store analysis results, but you might prefer to have it running on an on-premises gateway if it urgently switches off a machine upon fault detection. This is because in the latter case the reaction can be faster and these milliseconds for switching off the machine might be critical. However, the most critical actuations are currently handled locally and many of them might have to continue being done this way.
- Components write data into databases that might be accessed in different areas or by different kinds of users. For example, you might have a sensor-reading and analysis task that sometimes writes data into a local DB that is accessed directly by user smartphones and sometimes writes data into a remote server DB that is used for presentation on a website (cf. also example topology of Figure 20). Depending on which of the two happens more frequently, you might prefer to have the task running either on-premises or at the backend.
- Components have time- or mission-critical interactions with other modules, which are potentially not part of the topology. For example, a video analysis task might be preferably run close to the surveillance camera if it urgently increases the camera resolution upon detection of suspicious situations or it can also be run in the Cloud otherwise.

4.3 Design of a solution for achieving low-latency interactions in Stream Processing systems

In order to address the limitations described in section 4.2 and optimally address the deployment of topologies that involve low-latency requirements (such as the example topology of section 4.1), we have designed an SPF extension that takes such topology-external interactions into account. In the following, we describe the high-level architecture of our SPF extensions (4.3.1) and the way of operation of the extended system (4.3.2). Appendix A. provides some technical details about the prototypical implementation of the recommended solution as extension of Apache Storm.

4.3.1 Architecture of the recommended system extensions

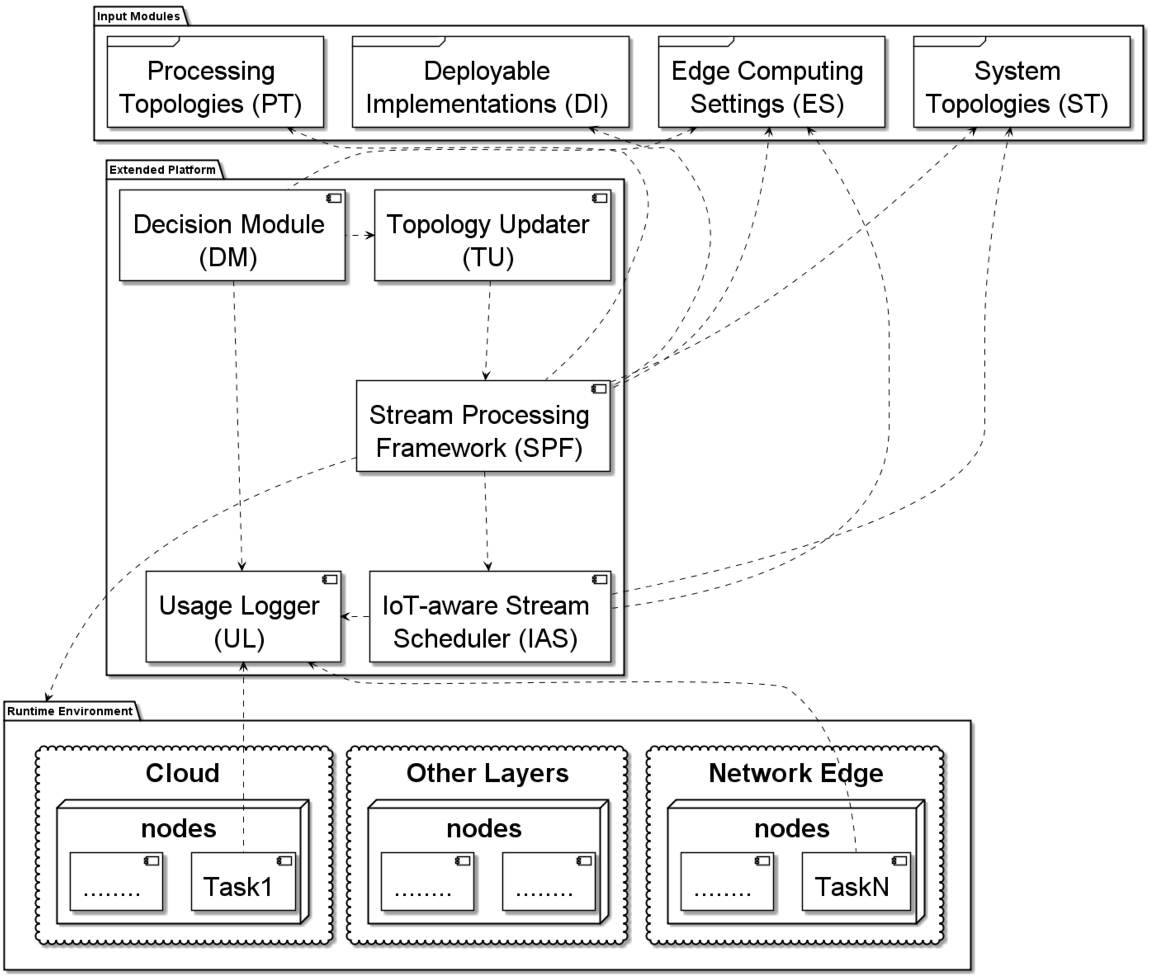


Figure 23. Architecture of the recommended Stream Processing Framework extensions

The suggested SPF system extension is shown in Figure 23 and it has the following main components:

- **Processing Topologies (PT)** are formal descriptions of computation steps (called topology components) and their relationships. The components may use data streams that come from other components as input and may produce other data streams as outputs, so that the PT corresponds with a computation graph. The logic of the components can be encapsulated, instantiated multiple times, parallelized, and executed on a distributed system. Therefore, the developer must also provide for each topology component a Deployable Implementation.
- **Deployable Implementations (DI)** are packaged code modules that implement the logic of the components. Running instances of this code are then called Tasks. DI also exists in this form in the state of the art.
- **Edge Computing Settings (ES) and the System Topologies (ST)**: In addition to standard SPF-required settings such as the desired number of instances for each component and the desired number of hardware nodes to be used for the execution, the ES contains information about interactions of components with topology-external entities such as actuators or databases, as well as further information about these entities and about computational characteristics of the component. Further, the ST of the proposed system contains information about the layer, the location, the domain, and other features of the nodes, in addition to their capabilities. Further, it contains similar information for system participants other than the nodes, e.g., for databases or actuators.
- The **Stream Processing Framework (SPF)** has the functionality described in the introduction, i.e., it uses the input modules to prepare, deploy, and execute topologies (i.e., their tasks) on a given distributed system.
- The **Usage Logger (UL)** receives information about usage aspects of running tasks that are not captured in the PT. This information is received by the UL in the form of single messages (or events) from a specific list of possible events and summarizes them in a system-wide usage report, which is retrieved regularly (e.g., periodically) by the Decision Module.
- The **Decision Module (DM)** uses this report together with ES information in order to identify potentials for enhancing system performance by triggering a redeployment of a topology. As soon as such a potential has been identified, the DM triggers the Topology Updater.
- The **Topology Updater (TU)** issues commands provided by the SPF in order to kill and then re-deploy the related topology (or topologies). After killing the topology and before actually re-deploying it, the SPF contacts the scheduler.
- The **IoT-aware Stream Scheduler (IAS)** responds to the SPF with a deployment that best satisfies the given requirements based on the characteristics and the usage of the components.
- The **Runtime Environment** is the distributed system which, in the proposed setting, consists of Cloud nodes, edge nodes, and nodes of other layers. The layering of the nodes is determined by the ST. The only interface that running tasks have to the platform is this of the UL.

4.3.2 Theory of Operation

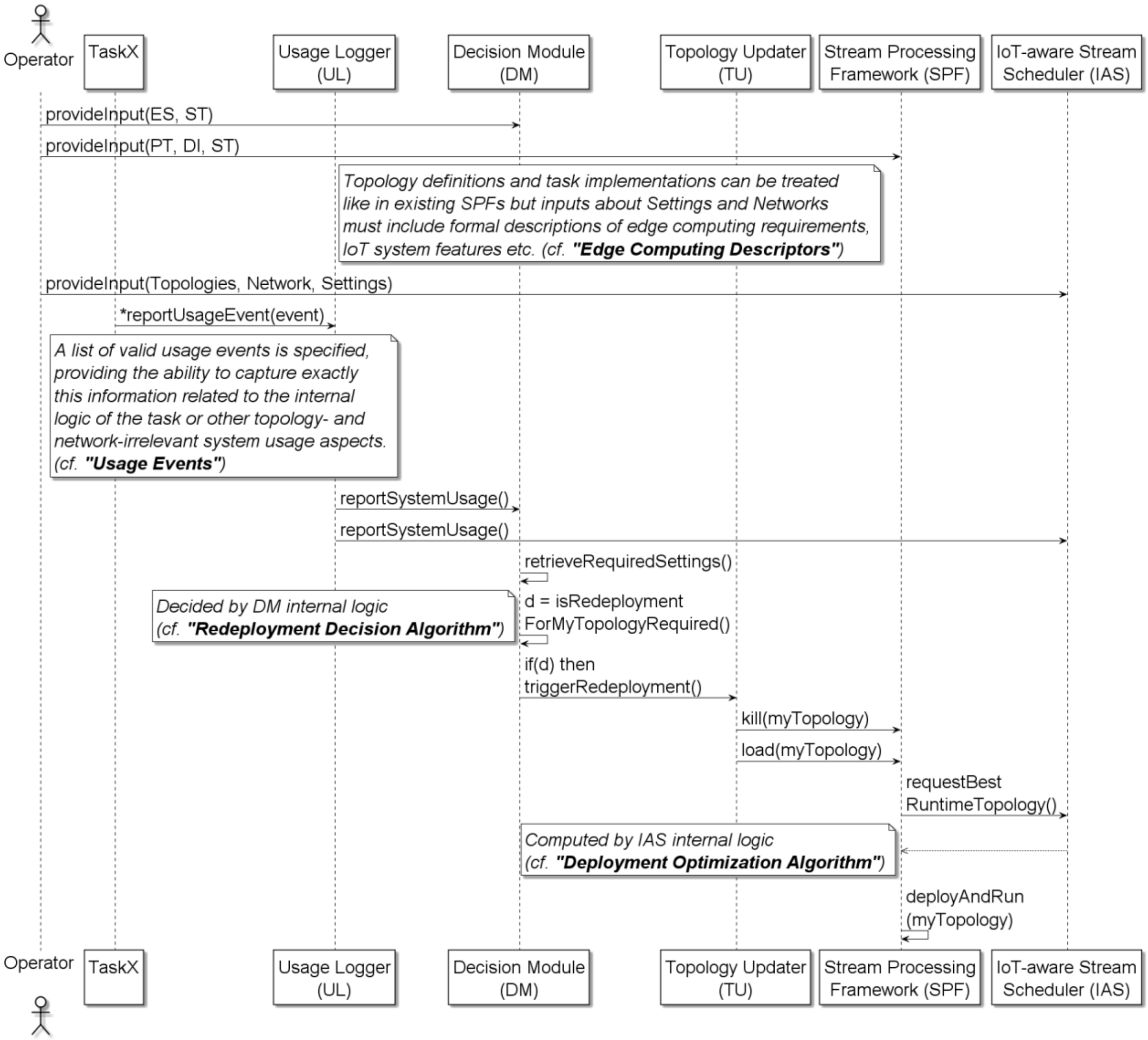


Figure 24. Sequence diagram of the suggested topology execution and re-deployment process

The main interactions that occur at runtime as part of the topology adaptation and execution process are shown in the sequence diagram of Figure 24, while the core enabling structures and algorithms (indicated with bold font in Figure 24) that have a part in it are specified in the following:

- **Edge Computing Descriptors**: These can be included partly in the ES and partly in the ST. In addition to the information that can be retrieved from the topology description, there are three main things (categories of characteristics) that shall determine if a task is relevant to network edge computing (and shall be executed at the edge) or not. These are:
  - The interfaces of the task with the environment, i.e., control of actuators, direct provision of intermediate results to users, event- or alarm-raising.
  - The characteristics of the databases with which the task interacts.
  - The task computation characteristics, namely its CPU- and data-intensity and security restrictions.

- **Usage Events**: Three types of events related to task usage that can be reported by a task back to the platform are specified:
  - Number and types of "recent" interactions with an actuator (note that "recent" is defined by a configurable time window).
  - Number and types of "recent" database transactions.
  - Probability of topology termination based on the most "recent" task executions (this metric actually shows the ratio with which incoming stream items do not lead to any outgoing stream and it can be important when deciding where to execute the tasks).
- **Redeployment Decision Algorithm**: The redeployment decision algorithm is run periodically in order to determine if the Task Allocation Algorithm should be executed in order to re-deploy the topology. The generic steps implemented by the algorithm are:
  - Compare the current deployment with an "optimal deployment".
  - Find the tasks that have different placements in these two deployments.
  - Evaluate the requirements that are violated by the placement of these tasks and decide if the total violation is big enough to justify a re-deployment. For example, n categories of violations can be defined, $\{t_1, \ldots, t_n\}$ are thresholds that indicate up to how many violations of each category can be tolerated before the Redeployment Decision Algorithm decides that re-deployment should be considered, and x is the amount of thresholds that need to be exceeded so that re-deployment is triggered. Example categories of violations are: "task that controls critical actuator of area X is executed in area Y", "task with high CPU intensity and no edge computing requirements is run at the edge", etc., and they can be detected based on the current deployment and the Edge Computing Descriptors.
  - Note that much more complex versions (with many different optimization functions) are possible, but they are out of scope of our recommendations.
- **Task Allocation Algorithm**: This algorithm computes and enforces the exact deployment, i.e., the allocation of tasks to nodes that best satisfies the requirements of the Edge Computing Descriptors, given the current Usage Events, according to a customizable logic. The specifics of this logic are also out of scope but, in fact, very similar principles might be followed as in the Redeployment Decision Algorithm, e.g., avoidance of requirements violations.

5 Conclusion

All in all, this Task has:

- Provided an analysis of “further” ICT/M2M or Smart City-related data sources that could be used for energy control optimization.
- Discussed business and technical aspects and provided recommendations for the integration of systems like OrPHEuS into Smart City Control Centers, focusing on data management and pre-aggregation.
- Described in detail a recommended technical solution for supporting actuation and low-latency interactions in data streaming systems such as the OrPHEuS monitoring system.

Major parts of the listed achievements have been published in the following high-profile publications:

- Apostolos Papageorgiou, Bin Cheng, Ehsan Poormohammady. *Edge-Computing-aware Deployment of Stream Processing Tasks based on Topology-external Information: Model, Algorithms, and a Storm-based Prototype*. 5th IEEE International Congress on Big Data (BigData ‘16), pages 259-266. IEEE, July 2016. DOI: 10.1109/BigDataCongress.2016.40.

6 Bibliography

[1] I. Al-Anbagi, M. Erol-Kantarci, and H. T. Mouftah. “A Low Latency Data Transmission Scheme for Smart Grid Condition Monitoring Applications”. In IEEE Electrical Power and Energy Conference, EPEC ’12, pages 20–25, Oct 2012.

[2] L. Aniello, R. Baldoni, and L. Querzoni. “Adaptive Online Scheduling in Storm”. 7th ACM International Conference on Distributed Event-Based Systems, pages 207-218, ACM, 2013.

[3] Apache Software Foundation. “Apache Storm project”. http://storm.apache.org/ (last visited in March 2016).

[4] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli. “Distributed QoS-aware Scheduling in Storm”. 9th ACM International Conference on Distributed Event-Based Systems, pages 344-347, ACM, 2015.

[5] Idowu, S., Saguna, S., Åhlund, C., & Schelén, O. (2014, November). Forecasting heat load for smart district heating systems: A machine learning approach. In Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on (pp. 554-559). IEEE.

[6] Idowu, S., Saguna, S., Åhlund, C., & Schelén, O. Applied Machine Learning: Forecasting Heat Load in District Heating System. Accepted with minor revisions Energy and Buildings Journal, Elsevier, 2016.

[7] M. Kuzlu, M. Pipattanasomporn, and S. Rahman. “Communication network requirements for major smart grid applications in HAN, NAN and WAN”. Computer Networks, 67:74–88, 2014.

[8] Nanda, R., Saguna, S., Mitra, K., & Åhlund, C. (2016, April). BayesForSG: a bayesian model for forecasting thermal load in smart grids. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (pp. 2135-2141). ACM.

[9] M. Rychly, P. Skoda, and P. Smrz. “Scheduling decisions in stream processing on heterogeneous clusters”. 8th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), pages 614-619, IEEE, 2014.

7 Disclaimer

The OrPHEuS project is co-funded by the European Commission under the 7th Framework Programme "Smart Cities" 2013.

The sole responsibility for the content of this publication lies with the authors. It does not necessarily reflect the opinion of the European Commission.

The European Commission is not responsible for any use that may be made of the information contained therein.

8 Contacts

Project coordinator

Ingrid Weiss & Silvia Caneva @ WIP – Renewable Energies

Sylvensteinstrasse 2, Munich, Germany

Email: Ingrid.weiss@wip-munich.de / Telephone: 0049 (0) 720 12 742

Email: silvia.caneva@wip-munich.de / Telephone: 0049 (0) 720 12 733

Task 3.1 Leader

Apostolos Papageorgiou @ NEC Europe Ltd.

Kurfürstenanlage 36, Heidelberg, Germany

Email: apostolos.papageorgiou@neclab.eu / Telephone: +49 (0)6221 4342 233

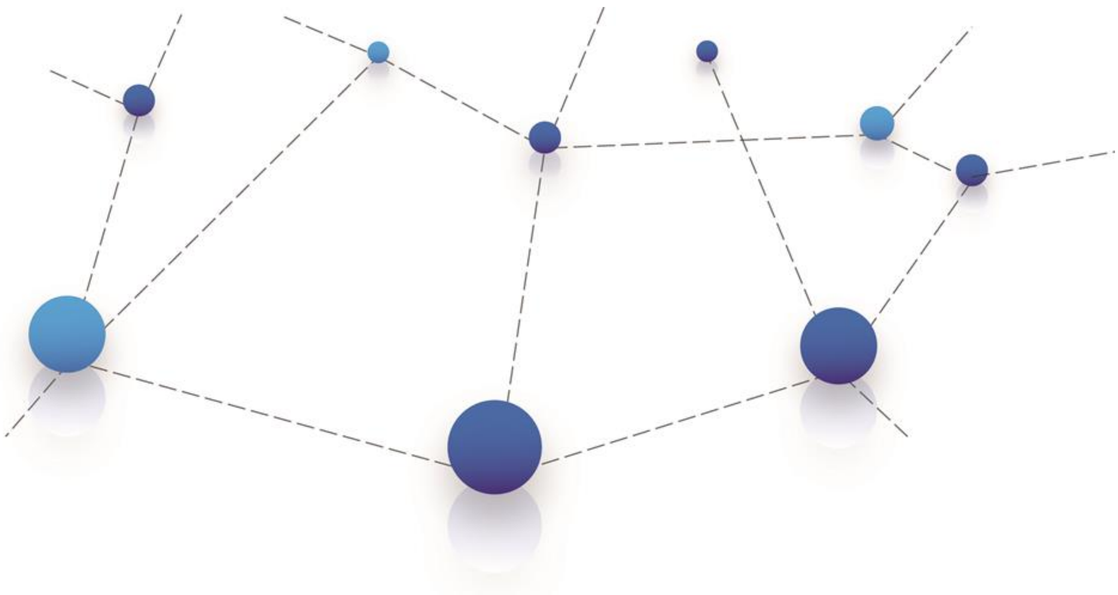
WP3 Leader

David Stakic @ Ulm University of Applied Science

Institute for Energy and Drive Technologies

Eberhard Finckh Str. 11, D-89075 Ulm, Germany

Email: stakic@hs-ulm.de / Telephone: +49 (0) 731 50 283



Appendix A. Guidelines for Prototypical Implementation of the Recommended Stream Processing Framework Extensions based on Apache Storm

This Appendix assumes technical background knowledge with regard to Apache Storm and it explains in Apache Storm terms how the main parts of the prototype have been implemented. Most importantly, it includes configuration files used in the prototype, thus supporting a better understanding of the invention.

The best way to implement the extensions is probably not by using Apache Storm. However, Apache Storm can provide a solid core around which it is possible to implement the main parts. This meant adding modules to the Apache Storm library, developing custom schedulers, using additional scripts, introducing various configuration files and system metrics, and more. More concretely:

- The PT and the DI can be realized with the standard Apache Storm mechanisms, i.e., using the Apache Storm library and implementing classes that extend the BaseRichBolt and BaseRichSpout classes, as well as using the TopologyBuilder to specify the deployment settings and the relationships of the components.
- The ES and the ST are partly realized by specifying JSON configurations for all spouts and bolts according to a template and providing a JSON reader that transforms them into Map objects which can be provided to the Config and TopologyBuilder objects of Storm. The rest of the realization of ES and ST relies on an extended version of storm.yaml (a node-scope configuration used by Storm). One configuration template, as well as a version of an extended storm.yaml, can be seen in Figure 25 and Figure 26, respectively.
- Getting down to the platform, Apache Storm itself is of course our SPF, while the UL has been implemented and provided as a jar file which must be added to the Apache Storm library (note that it is also declared inside the storm.yaml configuration of Figure 26). The DM is shipped as part of the UL, though more complex implementations might need to separate them. Finally, the IAS is a pluggable Apache Storm scheduler, i.e., another jar that has been developed by extending Storm's IScheduler and placed in the "lib" folder of Storm, while the TU is an OS-specific script which executes commands of the "storm" tool in order to kill and deploy topologies. The Task Allocation Algorithm is part of the internal logic of the IAS, while the Usage Events are implemented by exploiting the Storm Metrics concept.
- To create the target Runtime Environment, the Apache Storm platform has been installed onto various networked heterogeneous devices with different capabilities, while the layering of the devices and the involved entities has to be specified in the previously introduced configuration files (cf. Figure 25 and Figure 26; remember that each node has its own storm.yaml).

```
{
  "edgeInteractions": {
    "edgeExecutionRequired": "no",
    "controlledActuators": [
      {
        "name": "actuator1",
        "type": "switch",
        "area": "area1",
        "geo": "coordinates",
        "latencyRequirement": "low/medium/high"
      },
      {
        "name": "actuatorX",
        "type": "switch",
        "area": "areaX",
        "geo": "coordinates",
        "latencyRequirement": "low/medium/high"
      }
    ],
    "provisionedIntermediateResults": [
      {
        "result": "res1",
        "access": "pubSub",
        "mainUsersArea": "area1",
        "mainUsersLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      },
      {
        "result": "resX",
        "access": "pubSub/pull",
        "mainUsersArea": "areaX",
        "mainUsersLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      }
    ],
    "potentiallyRaisedEvents": [
      {
        "event": "event1",
        "propagationArea": "area1",
        "propagationLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      },
      {
        "event": "eventX",
        "propagationArea": "areaX",
        "propagationLayer": "edge/core/cloud",
        "latencyRequirement": "low/medium/high"
      }
    ]
  },
  "editedDatabases": [
    {
      "name": "db1",
      "type": "rdbms",
      "area": "area1",
      "dbLayer": "edge/core/cloud",
      "mainUsersLayer": "edge/core/cloud",
      "latencyRequirement": "low/medium/high"
    },
    {
      "name": "dbX",
      "type": "rdbms/nosql",
      "area": "areaX",
      "dbLayer": "edge/core/cloud",
      "mainUsersLayer": "edge/core/cloud",
      "latencyRequirement": "low/medium/high"
    }
  ],
  "computationCharacteristics": {
    "cpuIntensity": "low/medium/high",
    "expectedDataIOIntensity": "low/medium/high",
    "forbiddenAreas": ["forbiddenArea1", "forbiddenArea2", "forbiddenAreaX"],
    "forbiddenLayers": ["forbiddenLayer1", "forbiddenLayer2", "forbiddenLayerX"],
    "forbiddenDomains": ["forbiddenDomain1", "forbiddenDomain2", "forbiddenDomainX"],
    "specificDevice": "targetDeviceName"
  }
}
```


Figure 25. Component configuration template realizing parts of the Edge Computing Settings (ES) and System Topologies (ST) descriptors

```
storm.zookeeper.servers:
     - "192.168.56.1"

nimbus.host: "192.168.56.1"

topology.metrics.consumer.register:
   - class: "storm.geelytics.GeelyticsUsageLogger"
     parallelism.hint: 1

storm.scheduler:
"storm.schedulers.geelytics.GeelyticsTestScheduler"

supervisor.scheduler.meta:
    nodeId: "node1"
    nodeArea: "area1"
    nodeDomain: "domain1"
    nodeLayer: "edge/core/cloud"
    nodeCpu: "XXX"
    nodeMemory: "XXX"
```


Figure 26. Node-wide Apache Storm configuration for realizing parts of the Edge Computing Settings (ES) and System Topologies (ST) descriptors