PEGASE

PAN EUROPEAN GRID ADVANCED SIMULATION AND STATE ESTIMATION

D4.2-part 1

DTS ENGINE

Associated report

Grant Agreement Number: 211407 implemented as Large-scale Integrating Project
Coordinator: TRACTEBEL ENGINEERING S.A.
Project Website: http://www.fp7-pegase.eu

Document Information

Document Name:	D4.2-part 1- DTS Engine
ID:	**DEL_TE_WP4.5_D4.2_part1-DTS_Engine_v2**
WP:	4
Task:	5
Revision:	2
Revision Date:	14/03/2012
Author:	F.X. Bouchez, B. Haut

Diffusion list

STB, SSB

Approvals

	Name	Company	Date	Visa
Author	F.X. Bouchez, B. Haut	TE		
Task Leader	F.X. Bouchez	TE		
WP Leader	B. Haut	TE		

Documents history

Revision	Date	Modification	Author
0	11/01/2012	Creation	F.X. Bouchez
1	18/01/2012	Review	B. Haut
2	14/03/2012	Integration of comments from SSB	F.X. Bouchez

Table of content

1. Introduction

This document is the report associated to the DTS engine prototype developed in WP4. The prototype itself is the main deliverable.

The prototype under consideration has been obtained by re-integrating the algorithms developed and validated in T4.1|T4.2|T4.3 and T4.4 in existing product grade software called FAST [1] to produce a real-time simulation engine able to cope with the target of PEGASE project.

This document also describes the functionnalities which have been developped around the simulation engine in order to make the DTS Prototype a complete solution.

This document is structured as follow:

- Chapter 2 describes the functional requirements of the prototype;
- Chapter 3 provides a general presentation of functional structure;
- Chapter 4 presents the hardware requirements to achieve real-time performances;
- Chapter 5 focuses on telemetry model;
- Chapter 6 presents performances of the algorithm.

2. Objectives

Prototype was developed regarding all the requirements specified by WP1:

- Ability to handle at least 10.000 substations;
- Ability to handle 125.000 state variables;
- Ability to handle detailed topology substation and protection;
- Ability to handle VSC, HVDC and FACTS models;
- Ability to handle user defined models of protection systems;
- Ability to simulate the behaviour of complex generating unit controls;
- Ability to integrate models describing fast numerical controls.

Moreover, the prototype being a real-time simulation engine, the delay with respect to effective real-time should be seamless.

All these objectives are related to the core DTS engine and not to the “surrounding” modules such as the data handling or post-processing tools. These additional modules are provided with the DTS Prototype but were not the primary focus of the work carried out in Task 4.5. Only the performances/accuracy aspects will be considered by the tests carried out by WP6.

It is also to be mentionned that the DTS Prototype is ready to be connected to advanced interface prototype developed in WP2 (see chapter 5).

3. General presentation

3.1. Introduction

This chapter aims at explaining the functional structure of the DTS Prototype.

This prototype has been developed on basis of product grade software named **FAST** and contains the computation engine as well as the instructor and operator consoles.

Pre-processing of data is performed with the help of **EUROSTAG 4.5 LTS,** a special version of product grade software EUROSTAG. This software is needed to design the associated Bus-Branch topology system, create initial load-flow state and is also used to validate results.

The DTS Prototype contains a telemetry model which has been designed to accept connections of one or several external operator consoles through a standard up-to-date communication protocol.

The prototype is thus able to be connected with advanced interfaces developed in WP2 as well with some industrial SCADAs.

3.2. Structure

The functional structure of the DTS Prototype is as follows:

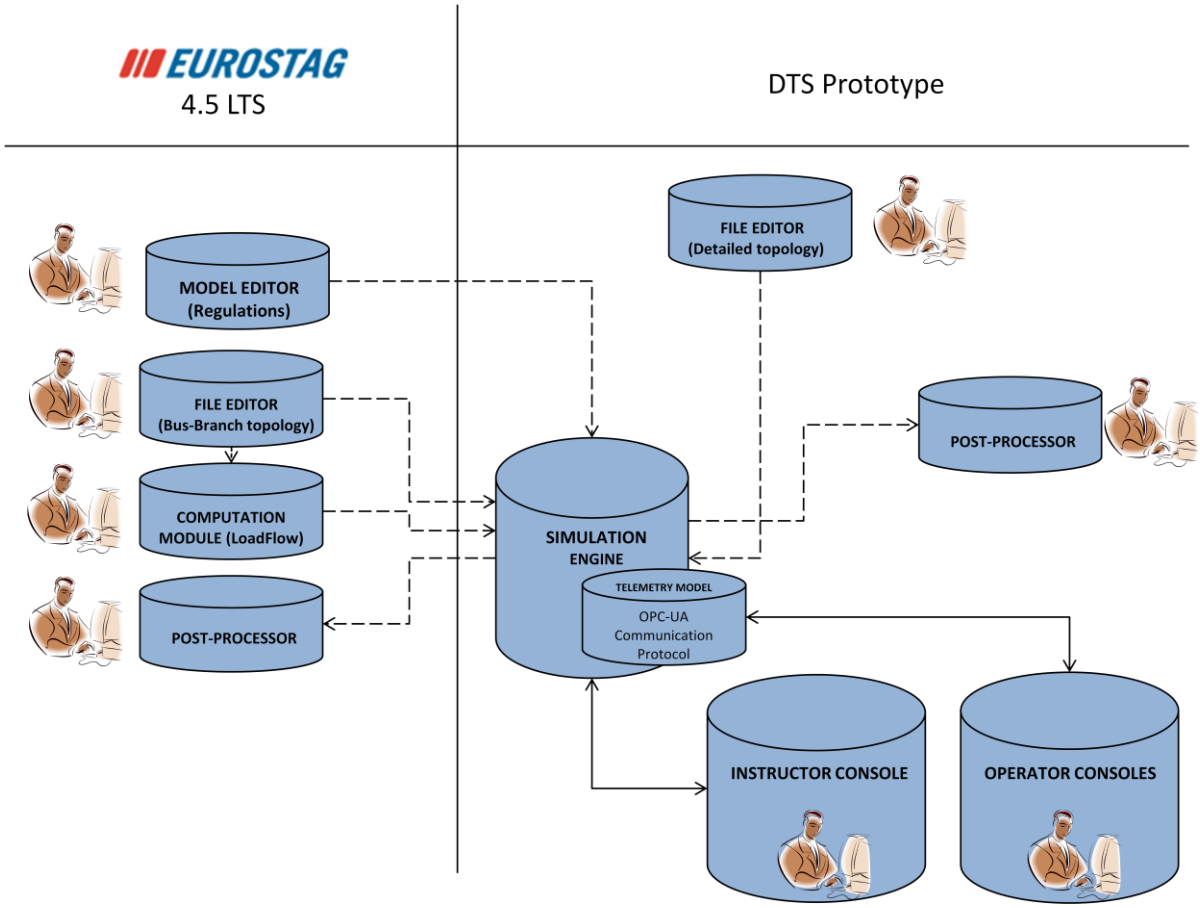

Figure 3-1: Functional structure

On the above picture, all dashed arrows correspond to pre-processing and post-processing operations and continuous ones correspond to interactions performed during simulation. As a result, it can be deduced that EUROSTAG 4.5 LTS is the program used to prepare major part of data and to prepare scenarios. The following sections present each module in details.

3.3. EUROSTAG 4.5 LTS

3.3.1. Data preparation

Three main sets of data are created in the program:

- Bus-Branch static topology: This data set contains all static definitions of equipments (buses, loads, generators, capacitors). Each bus created in this topology corresponds to a substation which will be detailed in DTS Prototype.

  Tool: Eurostag 4.5 LTS File Editor

- Dynamic data: Definition of all dynamics. This data set can be shared between EUROSTAG 4.5 LTS and DTS Prototype.

  Tool: Eurostag 4.5 LTS File Editor

- Regulations: Definition of machine regulations using graphical modelling language. This data set can be shared between EUROSTAG 4.5 LTS and DTS Prototype.

  Tool: Eurostag 4.5 LTS Model Editor

3.3.2. Initial loadflow computation

DTS Prototype starts from an initial converged state produced by a loadflow. This loadfLow is produced by EUROSTAG 4.5 LTS.

Tool: Eurostag 4.5 LTS Computation Module

3.4. DTS Prototype

3.4.1. Data preparation

Although major part of data is prepared in EUROSTAG 4.5 LTS, static data related to detailed topology must be introduced in DTS Prototype in separate data set.

Moreover, user must create an event file containing scenarios parameters and, optionally, events which will be played during simulation.

Tool: DTS Prototype File Editor

3.4.2. Real-time simulation

Real-time simulation is performed in DTS Prototype when all data preparation has been performed; following data sets are thus required to perform simulation:

- Initial loadflow state;
- Dynamic data set;
- Regulations;
- Data related to detailed topology;
- Event file.

3.4.3. Instructor console

DTS Prototype comes with a graphical instructor console which enables the following operations:

- Start | Stop | Pause the simulation;
- Launch interactive events ;
- Follow the simulation.

3.4.4. Operator consoles

DTS Prototype comes with a graphical operator console which enables the operator to visualise and to take actions on the network. Moreover, it is possible to connect one or several external operator consoles to the telemetry model included in DTS Prototype using a standard up-to-date communication protocol (OPC UA).

3.4.5. Post-processing

Before executing real-time simulation in DTS Prototype, it is possible to program the recording of some variables in an exportation file which can be processed in Post-Processor after simulation.

Moreover, in order to compare EUROSTAG and DTS Prototype simulations, it is possible to load exportation file issued from DTS Prototype as well as results files issued from Eurostag.

4. Hardware requirements

As a result of the choice of the algorithm, the DTS Prototype runs on a multi-processor system in a shared memory environment. In order to achieve real-time on PEGASE testcase, following machine characteristics are needed:

- **Processor**: 12 Intel computation cores (>3.4 Ghz) sharing same physical memory
- **Memory** : 6 GB of memory
- **Network**: 1GB ethernet
- **Hard drive**: At least 10 GB of free space
- **OS** : Windows 64 bits

5. Telemetry Model

5.1. Introduction

Telemetry Model is a necessary part of the DTS Prototype. Indeed, it permits the transfer of information between the DTS Prototype engine and visualization interfaces (SCADAs or MMIs). As requirements, we tried to focus on a solution which responds to the following characteristics:

- Use of a standard, secure and up-to-date protocol: this permits to use a large variety of SCADA and MMIs;
- Connection to Telemetry Model through long distance Network: this permits to foreseen a distributed architecture.

Following picture illustrates the solution:

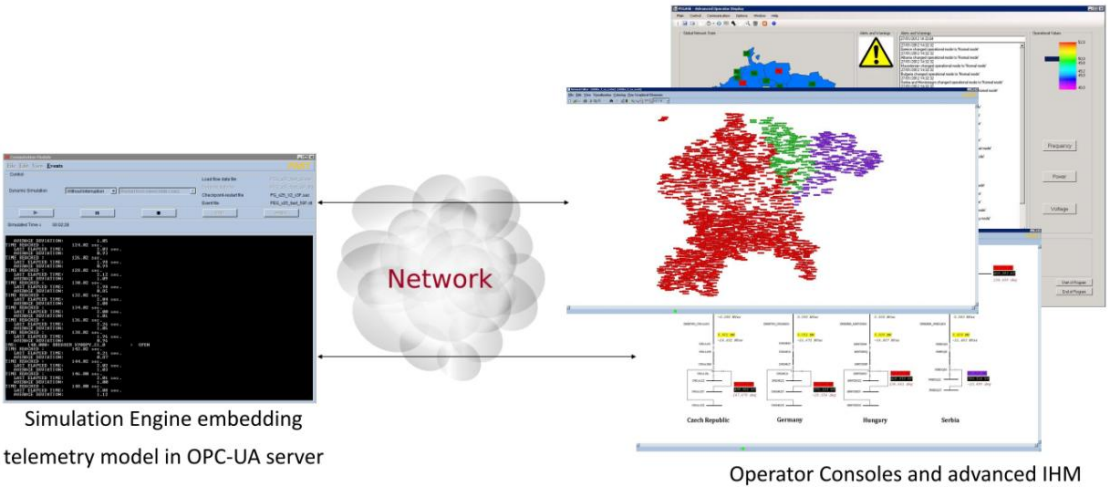

Figure 5-1: Interactions between DTS Prototype engine and operator consoles

5.2. Structure

Core of the telemetry model is the OPC-UA server which presents following characteristics:

- Contains a database, dynamically created and updated by DTS Prototype engine ;
- Contains methods to communicate actions to DTS Prototype engine;
- Is always ready to accept connections from visualization interfaces.

Following picture illustrates the structure of solution:

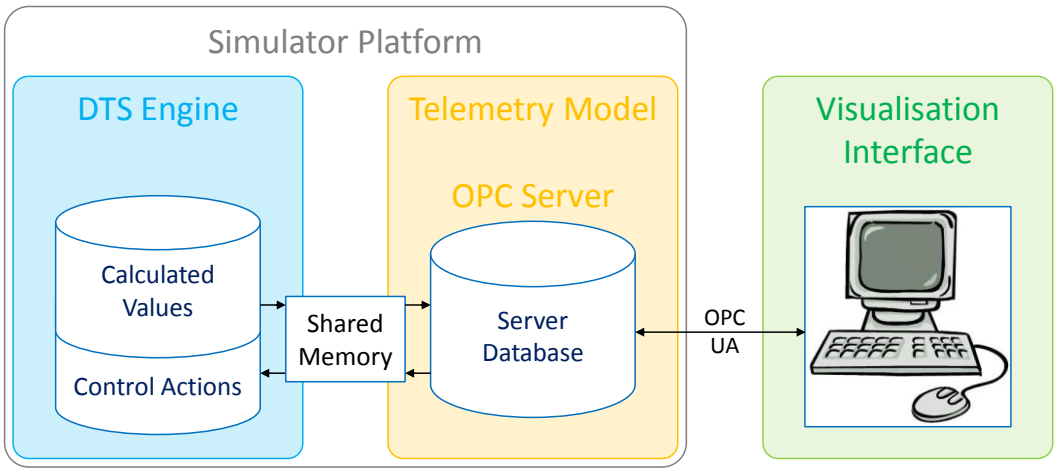

Figure 5-2 Telemetry Model data flow structure

Information circulates both ways:

- From DTS Prototype engine to operator consoles, 2 kinds of information are transferred :
  - Logical states corresponding to breaking device positions, to alarms, etc.
  - Analogical values corresponding to measurable physical values (fluxes, voltages, currents,…) or discrete values (transformer tap position, etc.).
- From visualization interface to DTS Engines, 2 kinds of information are transferred:
  - Logical states corresponding to command actions of breaking devices.
  - Analogical values corresponding to regulation actions (machine set point modification) or command actions (transformer tap position, etc.).

For performance and simplicity reasons, it has been chosen to integrate the whole telemetry model in the simulator platform. This structure permits the exchange of information between DTS Engine and Telemetry Model using shared memory mechanisms.

Each OPC/UA client can access all the observables exported by the DTS Prototype. The following figure illustrates the type of observable exported by the DTS Prototype and the hierarchical structure used.

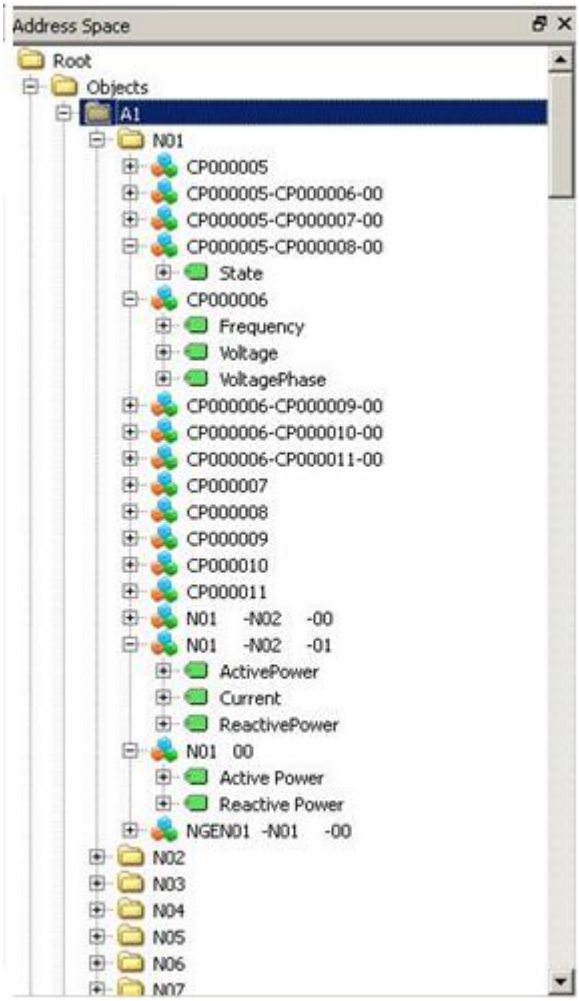

Figure 5-3: Observables access through a standard OPC UA client browser

In this structure:

- A1 is the name of a zone;
- N01 is the name of a substation;
- CP0000005-CP000008-00 is the name of a breaker;
- CP0000006 is the name of a busbar;
- N01 -N02 -01 is the name of a branch;
- N01 00 is the name of a load.

6. Performances

6.1. Introduction

Results obtained in T4.1|T4.2|T4.3 and T4.4 have been mixed to create a real-time simulation engine meeting PEGASE objectives. The details of the algorithms can be found in the deliverable *DEL_WP4.1-2-3-D4.1 Algorithmic requirements for simulation of large network extreme scenarios* and the report *REP_ULg_4.4_R442_Algorithmic and modelling improvements for simplified simulation; validation on industrial cases*.

Performances presented in this section have been measured on the PEGASE testcase containing 125.000 state variables, 15.000 substations, 13000 branches, 4000 machines on a hardware environment corresponding to hardware requirements chapter.

Results are presented on 2 scenarios:

- **Cascade**: Opening of a line leading to a cascade of automata impacting the network topology;
- **Loss of synchronism**: short-circuit of 1 s. near a step-up transformer in order to impose the loss of synchronism of a unit.

On both scenarios, we have set the synchronisation time between the simulation engine and the telemetry model to 2 seconds. This means that the real-time must be crossed at this interval. Simulation stepsize is fixed to 20 milliseconds.

It should be noted that these two testcases are very computational demanding and have been chosen in order to push the prototypes to its limit. These can be considered as "worst testcases". Most of the simulation carried out for training purposes will be less computational demanding.

6.2. Cascade scenario

This case corresponds to the opening of a line leading to a cascade of automata impacting the network topology.

Following graph presents the average deviation between computation time and simulated time in function of the simulation time. Synchronisations are performed every 2 seconds (indicated using green circles). Results are presented for sequential (using 1 computational core) and parallel simulations (using 12 computational cores).

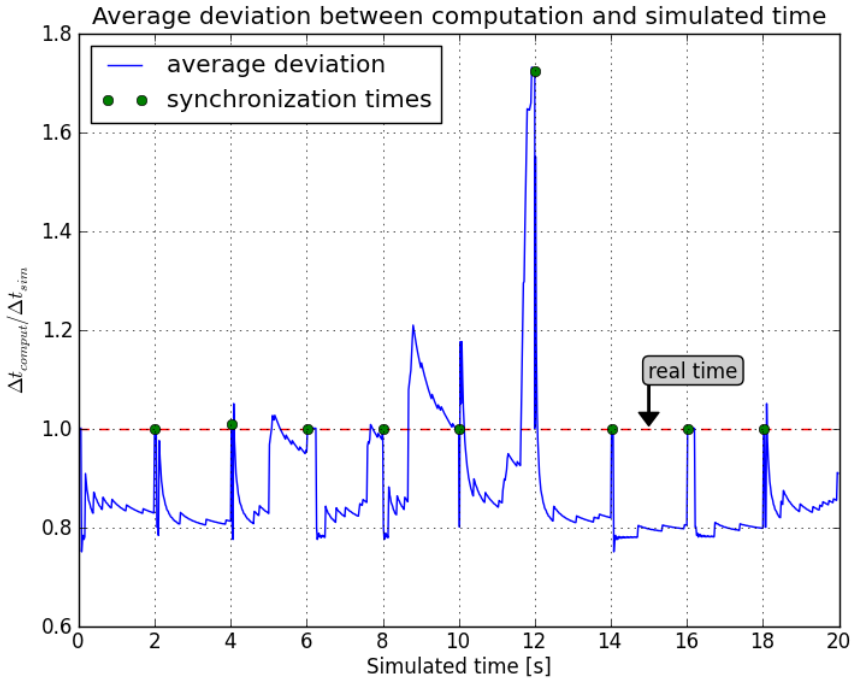

Figure 6-1: average deviation between computation and simulated time for the "cascade" scenario using 12 threads

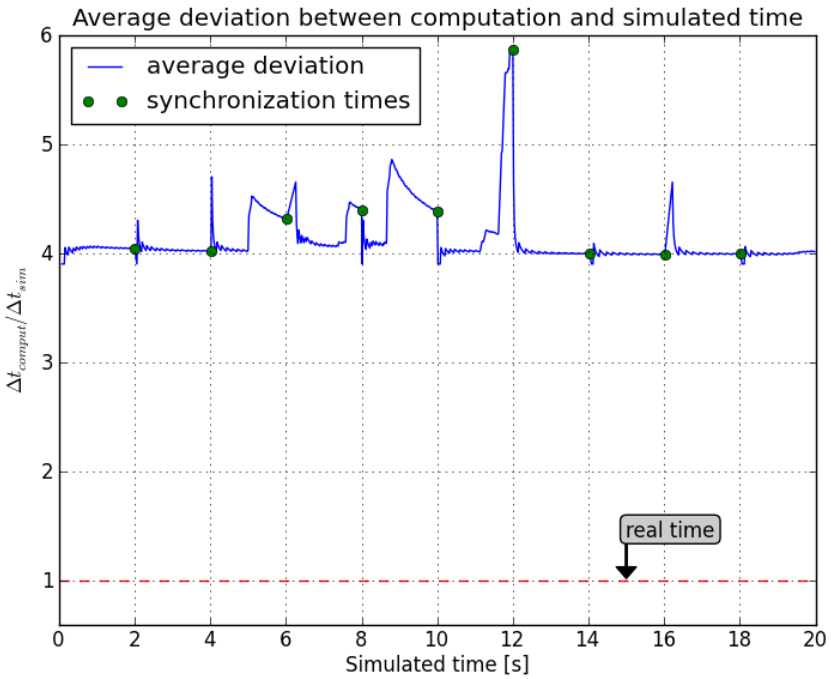

Figure 6-2: average deviation between computation and simulated time for the "cascade" scenario without parallelization

It can be seen that most of the time the computation engine is faster than real-time (blue curve below the red line) and has to wait before the synchronization times (every 2 seconds). In some case (around 9 sec. for example), the engine is temporary slower than real-time (blue curve above red line) but real-time is caught back before the synchronization time: the user won't notice anything.

Between between t=10 s. and t=12 a cascade of 10 automata impacting the network topology at different times happens. This is very computational demanding since after each automaton triggering the network Jacobian must be re-evaluated and re-factorized. Since this cascade has happened just before the synchronization period, the computation engine does not have sufficient time to catch back real-time for the synchronization (green circle above red line). However, starting from the next period (t=12 s. to t=14 s.) up to the end of the simulation the real-time target is again satisfied.

The second figure illustrates the very large importance of a good parallelization. Without this parallelization, real-time is never satisfied and the computation engine is sometimes **six times** slower than real-time.

6.3. Loss of synchronism scenario

This case corresponds to the simulation of a short-circuit of 1 second near a step-up transformer. This (very) large short-circuit period ensures that the close units lose the synchronism. Following graph presents the average deviation between computation time and simulated time in function of the simulation time. Synchronisations are performed every 2 seconds (indicated using green circles).

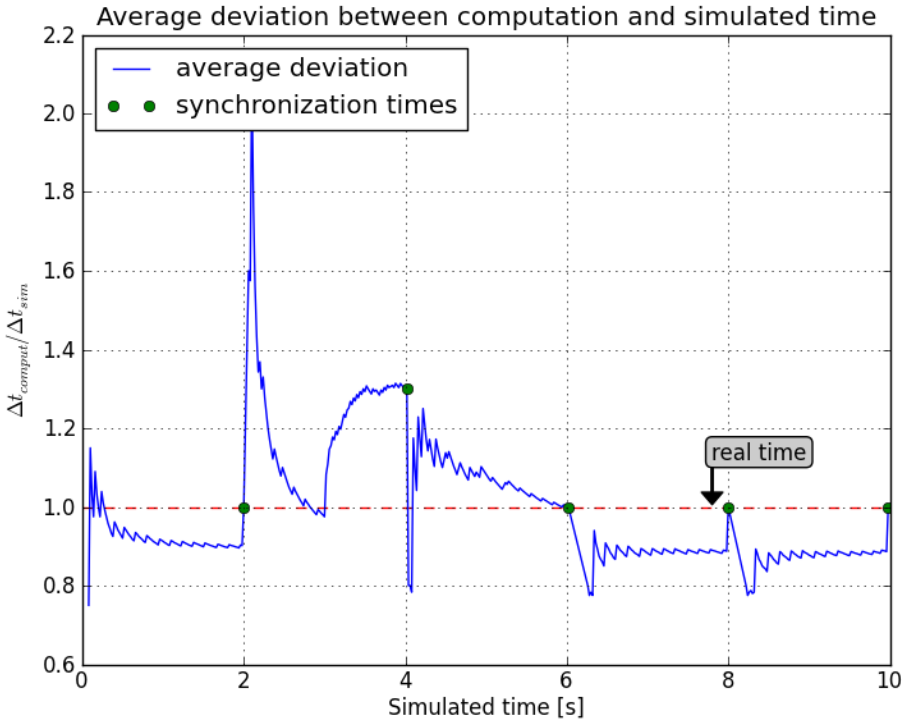

Figure 6-3: average deviation between computation and simulated time for the "loss of synchronism" scenario using 12 threads

The same conclusions can be drawn than previously: most of the time the computation is faster than real and has to wait before synchronization. Even if at one particular time during the simulation the real-time target was not reached (around t=4s.), the real-time is caught back during the next period and held up to the end of the simulation.

7. User Guide

The user is referred to the FAST-DTS and EUROSTAG user manuals.

8. Conclusions

The DTS Prototype is ready and all requirements specified by the WP1 have been satisfied. The prototype is not only able to simulate very large systems but has now a greater flexibility thanks to the introduction of the macro-language. It allows the user to define its own regulation through a block diagram editor. Complex FACTS and HVDC systems can be modelled using this approach.

First internal tests show that the computation engine is able to satisfy the real-time constraint even on very difficult testcases. More tests will be carried out by the WP6.

9. Bibliography

[1] S. Gissinger, P. Chaumes, J.-P. Antoine, A. Bihain, and M. Stubbe, “Advanced dispatcher training simulator,” Computer Applications in Power, IEEE, vol. 13, pp. 25–30, Apr. 2000.