Ευχαριστίες

Ύστερα από μία πορεία πέντε ετών στο Τμήμα Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, ολοκληρώνω τις προπτυχιακές μου σπουδές με την εκπόνηση της παρούσας διπλωματικής εργασίας

Θα ήθελα αρχικά να ευχαριστήσω θερμά τον κ. Κοράκη Αθανάσιο, καθηγητή του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων, για την απεριόριστη υποστήριξη που μου προσέφερε όλο αυτό το διάστημα, καθώς και για τις χρήσιμες συμβουλές και υποδείξεις του κατά την εκπόνηση της διπλωματικής μου εργασίας.

Ευχαριστώ επίσης τον επιβλέποντα της εργασίας μου αυτής, Καθηγητή του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων, κ. Λέανδρο Τασιούλα και τον συνεπιβλέποντα Λέκτορα του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων κ. Αντώνιο Αργυρίου για την πολύτιμη καθοδήγησή τους.

Από καρδιάς θα ήθελα να ευχαριστήσω τον κ. Ιωάννη Καζδαρίδη και τον κ. Χαρίλαο Νιαβή, μεταπτυχιακούς φοιτητές του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων, τον κ. Δημήτριο Γιάτσιο, υποψήφιο Διδάκτορα του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων και γενικότερα όλα τα παιδιά του NITLab για την πολύτιμη συνδρομή τους στο να φέρω εις πέρας την διπλωματική μου εργασία.

Τέλος, ευχαριστώ θερμά από καρδιάς την οικογένεια μου για την αμέριστη συμπαράσταση που μου παρείχε όλα αυτά τα χρόνια για την επιτυχημένη ολοκλήρωση των σπουδών μου.

Στην οικογένεια μου

CONTENTS

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη Διπλωματική Εργασία αναφέρεται στη σχεδίαση και ανάπτυξη μεθόδων συνεργατικότητας για τη διαχείριση ερευνητικών πειραματικών διατάξεων. Το project περιγράφεται σε τέσσερα (4) μέρη.

Το πρώτο μέρος της Διπλωματικής αποτελεί μία εισαγωγή στη γενική ιδέα του τι είναι ένα testbed. Στη συνέχεια γίνεται μια αναφορά στο τι είναι το Federation ανάμεσα στα testbeds και ποιά είναι η χρησιμότητά του. Έπειτα ακολουθεί μια σύντομη περιγραφή του NITOS testbed στο οποίο έγινε το σημαντικότερο κομμάτι της δουλειάς μου. Επιπλέον υπάρχει μια μικρή εισαγωγή στο Control and Managment Framework (OMF).

Στο δεύτερο μέρος παρουσιάζω το NITOS CMC Service, ένα OMF Service που δίνει τη δυνατότητα να ανοίγουμε να κλείνουμε ή να κάνουμε reset έναν κόμβο. Παράλληλα περιγράφω τη χρήση των Chassis Manager Cards (CM cards).

Στο τρίτο μέρος παρουσιάζω το measure Service, ένα OMF Service που σχεδιάσαμε και υλοποιήσαμε ώστε να παίρνουμε περιβαλλοντικές μετρήσεις από τους αισθητήρες που έχουμε βάλει στις CM κάρτες. Το Service μας δίνει ακόμα τη δυνατότητα να γνωρίζουμε την κατάσταση του κόμβου (on/off) αλλά και την κατανάλωση ενέργειάς του.

Τέλος, στο τέταρτο μέρος περιγράφω το SFA γενικά και το Scheduler API που έχουμε υλοποιήσει. Το SFA είναι το μεγαλύτερο βήμα στον τομέα του federation μεταξύ των testbeds μέχρι σήμερα. Το Scheduler API που υλοποιήσαμε αποτελεί το μέσον το οποίο θα χρησιμοποιούν τα testbed που επιθυμούν να είναι συμβατά με το SFA προκειμένου να μπορεί να γίνει εξερεύνηση και δέσμευση των resources τους.

ABSTRACT

The present Final Project Dissertation - Thesis deals with the design and development of federation methods on networking testbeds. The project is analyzed in four (4) parts.

The first part of the project consists of an introduction about the main idea of a testbed. After that, there is a chapter that outlines the federation generally and its necessity. Also I describe the NITOS testbed where I have done the most important part of my work. Additional there is a sort introduction in Control and Management Framework (OMF).

In the second part, I present the NITOS CMC Service, an OMF Service that is used to set on, off or resets the nodes of the testbed. Also I describe the usage of Chassis Manager Cards (CM cards).

At the third part, I present the measure Service, another OMF Service that is used to take environmental measurements from the sensors attached on the CM cards. The service is also used to take the node's state (on/off) and to measure its power consumption.

Finally, in the fourth part I describe generally the SFA and the Scheduler API we have implemented. SFA is the biggest step in federation between testbeds until now. The Scheduler API that we developed is the way that the testbed that want to be SFA compatible, will use for browsing and reservation of their resources.

1. Introduction

1.1 The meaning of a testbed

The last few years, the term of testbed is mentioned more and more. On the other hand, many are the people who refer to testbeds as the future form of the Internet.

But exactly what is a testbed?
A testbed is a platform for experimentation of large development projects. It consists of software, hardware and networking components that allow the experimenter to design, execute and measure an experiment. Testbeds allow for rigorous, transparent and replicable testing of scientific theories, computational tools and also new technologies. The term is used across many disciplines to describe a development environment that is shielded from the hazards of testing in a live or production environment. It is a method of testing a particular module (function, class, or library) in an isolated fashion. Also, a testbed is used as a proof of concept or when a new module is tested apart from the program or the system it will be added to later.

A testbed has many types of persons who engaged with it.
The first one is its **owner**. The owner of a testbed may be a single person, a corporation or the State. The owner ensures the availability of the resources (nodes, hardware, cables, the building etc.) that the testbed needs for a normal operation.
In the other hand, there are the **developers**. A developer cares about the appropriate usage of the testbed's components. Also, develops new functionalities and tools that experimenters may need, or according to the increasing of technology, in order to make the testbed more and more powerful.
Finally, there are the testbed **users**. A user usually is an experimenter that wants to use the testbed to simulate the true environment conditions under which, his product will operate. In this way, he will confirm its right or wrong operation. A user may be a student that works on a laboratory or a researcher.

Thanks to the state-of-the-art Slice Federation Architecture (SFA), testbeds are now able to federate with PlanetLab-based testbeds, OMF-based testbeds and with some SFA back-end programming testbeds based upon other technologies.

Summarizing therefore the main reasons for federation are:

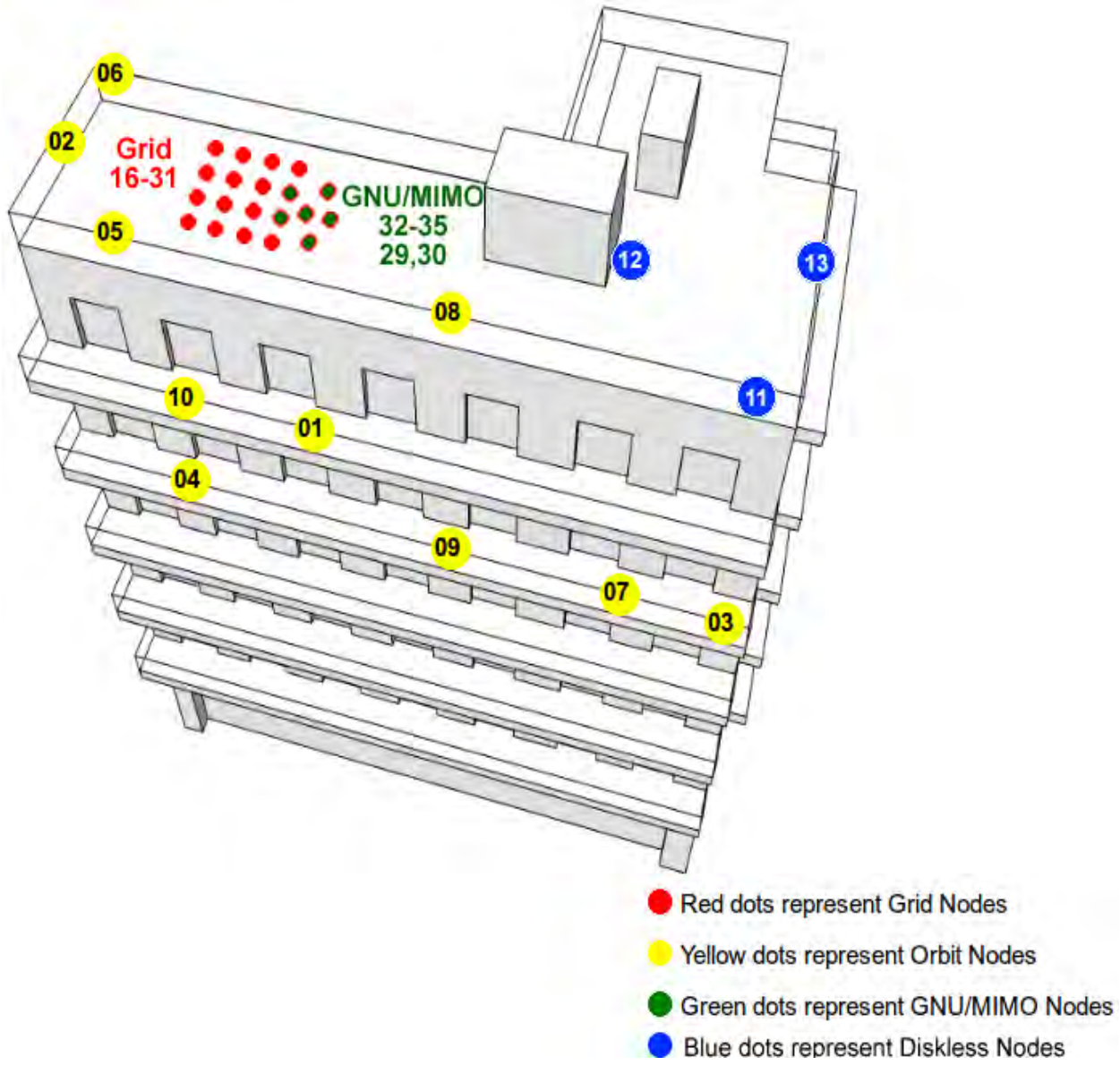
- **Develop and promote a testbed**
  The testbed will open to the thousands of registered users who are already part of the global network, and gain access to the expertise of the community.

- **Offer new facilities to the users**
  Through federation the users will gain access to new testbeds. For example, PlanetLab Europe's geographically distributed facility gives access to over 1000 machines located all over the world, while the NITOS wireless testbed's rich features offer extensive resources to users, including mobile nodes, cameras, sensors and software-radio boards. In addition, the ETOMIC measurement infrastructure offers high-precision measurements - synchronized active network measurements on the order of tens of nanoseconds.

- **Join a community of testbed developers**
  When a testbed is connected, anyone who uses it, is also connected to a research community that includes some of the world's leading experts in computer networking, publishing in the world's top conferences and journals. Also, he will have the opportunity to participate in forums and discussions that will shape the future of networking testbeds.

1.3 The NITOS Wireless Testbed

The most important part of my work based on NITOS testbed. CERTH has developed a wireless testbed called Network Implementation Testbed using Open Source platforms (NITOS). NITOS is a testbed offered by NITLab and consists of 50 wireless nodes based on open source software. The testbed is outdoor and uses a wireless layout of nodes which can be used for measurements and experiments in a real time environment. That gives the opportunity to observe the results of an experiment out of the "secured" environment of a simulation program and to take conclusions in real dangers or problems that the final product maybe deal with.



```
▼<service name="power">
   <info>Get a power consumption measurement</info>
  ▼<args>
   ▼<arg isRequired="true" value="hrn" name="hrn">
      <info>hrn of the resource</info>
    </arg>
   ▼<arg isRequired="true" value="domain" name="domain">
      <info>domain for request.</info>
    </arg>
   </args>
  </service>
▼<service name="state">
   <info>Get the current state of a node</info>
  ▼<args>
   ▼<arg isRequired="true" value="hrn" name="hrn">
      <info>hrn of the resource</info>
    </arg>
   ▼<arg isRequired="true" value="domain" name="domain">
      <info>domain for request.</info>
    </arg>
   </args>
  </service>
▼<service name="temperature">
   <info>Get a temperature measurement</info>
  ▼<args>
   ▼<arg isRequired="true" value="hrn" name="hrn">
      <info>hrn of the resource</info>
    </arg>
   ▼<arg isRequired="true" value="domain" name="domain">
      <info>domain for request.</info>
    </arg>
   </args>
  </service>
 </serviceGroup>
</services>
```

3.3 How to call the Service

There are three different ways to call and execute a measure service. The first one is to be utilized by a testbed user directly by invoking it in a browser or using the curl command. For example a user can send the following HTTP request

http://nitlab.inf.uth.gr:5054/measure/temperature?hrn=omf.nitos.node019

and the result will be like the figure bellow:

```
▼<Measurement>
   temp
   <value>31.4</value>
 </Measurement>
```

Certainly, all the actions above will be executing according to pre-agreed policies. Moreover, it is a secured API. That means that only users that we have authorized are able to use it. The authorization is carried out by providing a password key to the user that we want to use our API.

The communication between Scheduler API and Generic SFA Wrapper effects according to the XMLRPC protocol.

Scheduler API is an XMLRPC server that listens to a specific port and serves all the possible requests from the Generic SFA Wrapper. On the other hand, except from the Generic SFA Wrapper, any XMLRPC client could make calls to this API and get responses, if it is allowed to.

4.4 How SFA scheduler works

The Generic SFA Wrapper will query the API about the availability of resources and will make requests for resource allocation on slices, according to the requests that an experiment done using the plugin of MySlice. Then, the API receives all the above requests. Then it speaks with the database by sending appropriate SQL queries.

However, it has some sophisticated components like policies and a conflict resolution algorithm. For example, When two users want a resource at the same time then, the scheduler should check who are these users and if they have equal rights. If so, then a conflict resolution algorithm should give the resource to only one user. Until now, as a first approach we should implement just a FIFO reservation scheme. On the other hand, if a user wants a specific topology and some of these nodes are already reserved, the API reserves the other available nodes from the list.

4.5 The methods of the Scheduler API

There are four (4) different types of methods

- Get methods – Give the oportunity to know all the available information of the testbed
- Add methods – Add new information to the testbed
- Delete methods – Delete useless informations from the testbed
- Update methods – Update testbed's current information

Analytically, the methods we have implemented are:

GET methods:

- getNodes
- getChannels
- getTestbedInfo
- getReservedNodes
- getReservedChannels
- getSlices
- getUsers

ADD methods:

- reserveNodes
- reserveChannels
- addUser
- addUserToSlice
- addUserKey
- addSlice
- addNodeaddChannel

DELETE methods:

- deleteKey
- deleteNode
- deleteUser
- deleteUserFromSlice
- deleteSlice
- deleteChannel
- releaseNodes
- releaseChannels

UPDATE methods:

- updateNode
- updateChannel
- updateUser
- updateSlice
- updateReservedNodes
- updateReservedChannels

4.6 Scheduler API standardization

The Scheduler API that I described, we tried to make it as generic as we can. As NITOS is the first wireless testbed that used the meaning of the scheduler and the meaning of the slice, we had the experience required to make a generic API and we achieved it. The Scheduler API will be standardized as the API that will be used from all the testbeds that will be SFA compatible.

4.7 Results of the Scheduler API until now

Generally, the meaning of SFA is very new. NITOS is the first OMF-SFA compatible testbed.

Right now the Scheduler API is running in our server and anyone who has an authorized XMLRPC client can use it to observe the available resources of NITOS, to make a reservation and run an experiment.

The first results of its usage will be presented at the forthcoming review of the OpenLab project in 15th of October.

5. REFERENCES

[1] NITOS testbed:
http://www.nitlab.inf.uth.gr/NITlab/index.php/testbed

[2] OMF – A Control and Management Framework for Networking Testbed
http://www.omf.mytestbed.net/projects/omf/wiki/Introduction

[3] The Generic SFA Wrapper
http://www.sfawrap.info/deploying/docs

[4] "Slice (-based) Federation Architecture - SFA Tutorial v0.1" Panayotis (Panos) Antoniadis Jordan Augé, Marco Bicudo, Timur Friedman (UPMC) Thierry Parmentelat (INRIA), Tony Mack (Princeton)

[5] NITLab Chassis Manager Cards
http://www.nitlab.inf.uth.gr/NITlab/index.php/testbed/hardware/cm-card

[6] "Integrating sensor measurements through CM cards as an OMF service", in the proceedings of TridentCom 2012, Thessaloniki, Greece, June 2012 V. Maglogiannis, D. Giatsios, G. Kazdaridis, T. Korakis, I. Koutsopoulos and L. Tassiulas.
http://nitlab.inf.uth.gr/NITlab/papers/Integrating%20sensor%20measurements%20through%20CM%20cards%20as%20an%20OMF%20service.pdf

[7] OMF : A Control and Management Framework for Networking Testbeds, T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar. ACM SIGOPS Operating Systems Review, Vol. 43, pp. 5459, 2010.