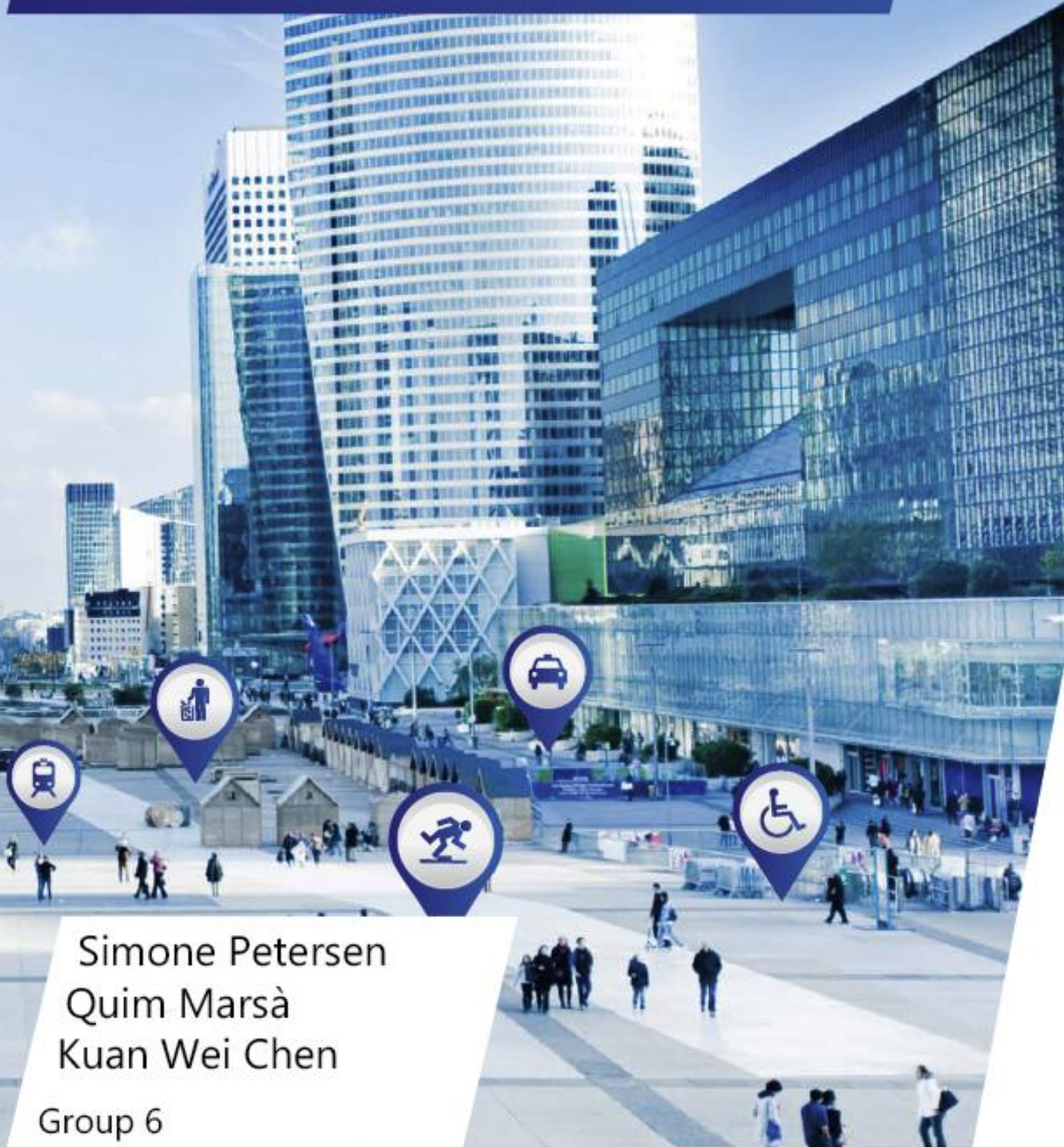




TESEUS

SMART CITIES



Simone Petersen
Quim Marsà
Kuan Wei Chen

Group 6



Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú

UNIVERSITAT POLITÈCNICA DE CATALUNYA

2014-2015 EPS



**Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú**

UNIVERSITAT POLITÈCNICA DE CATALUNYA

EPS - PROJECT

TITLE: Teseus project – group 6

STUDENTS:

EPS	Quim Marsa
EPS	Simone Petersen
EPS	Kuan Wei

EPSEVG	Lorena Arriaga
EPSEVG	Joan Josep

SUPERVISORS:

Eva Tordera – University supervisor
Sergi Sanchez – University supervisor
Josep Farré – Company supervisor

DATE:

12th of June 2015

TITLE:

Teseus project

FAMILY NAME: Petersen

FIRST NAME: Simone Folkvard

HOME UNIVERSITY: Technical University of Denmark

SPECIALITY: Global business engineering

FAMILY NAME: Marsà Bou

FIRST NAME: Quim

HOME UNIVERSITY: Universitat de politecnica de Cataluna - EPSEVG

SPECIALITY: Electronic engineering

FAMILY NAME: Chen

FIRST NAME: Kuan Wei

HOME UNIVERSITY: National Taiwan University of Science and Technology

SPECIALITY: Electronic and computer engineering

FAMILY NAME: Arriaga

FIRST NAME: Lorena

HOME UNIVERSITY: Universitat de politecnica de Cataluna - EPSEVG

SPECIALITY: Computer engineer

FAMILY NAME: Aleixendri

FIRST NAME: Joan Josep

HOME UNIVERSITY: Universitat de politecnica de Cataluna - EPSEVG

SPECIALITY: Computer engineer

Abstract

This project, TESEUS, is being carried out within the umbrella of the ongoing Urban Node study and focuses on the development of a computational system capable of handling a huge amount of data received from urban nodes in a smart city.

There are already existing computer systems with computational power capable of handling the amount of data a smart city requires. Although these computers may provide all the power required to maintain and manage a smart city, they can always be made more efficient. The lack of efficiency comes from the inability to perform in a scalable way. A computer is not always required to work with its full power. The way supercomputers are designed is rather inefficient when managing lesser amounts of data than expected. The Teseus project objective is the possibility of building a low cost, scalable data server with focus on efficiency and the facilities a scalable system offers. In order to create a new computational system, research and possibilities for both hardware and software that offers scalability and its management have been carried out and explored.

Given the extent of the development, as well as the complexity of the project that involves combining existing technology hardware with available software solutions, leaves further programming and research for future groups.

In addition, this report includes a Business Model Canvas and Eco Design considerations regarding the future plans and environmental aspects for Teseus. The methodology used as a project management tool in order to organize the project and description of the Urban Node projects relation to Teseus, is also included.

Keywords—data transfer and acquisition, ARM architecture, computational system efficiency, scalability, smart cities, Urban node, grid, supercomputer, single-board computers and Beowulf clusters.

Table of content

Abstract	1
List of figures	4
List of tables.....	5
1. Introduction	6
1.1 Previous projects	6
1.2 Teseus and Neàpolis	7
1.3 Objectives	7
1.4 Team	8
1.5 Limitations	8
2. Smart cities.....	9
2.1 Urban Node project.....	10
2.2 State of the art.....	11
3. Hardware.....	13
3.1 Advanced 'RISC' Machine (ARM) vs. Intel x86 architecture (CISC).....	13
3.2 Energy efficiency.....	14
3.3 Performance	15
3.4 Single Board Computer (SBC)	17
3.5 Switch vs. router	22
4. Software	25
4.1 Configuration of switch	25
4.2 Virtualization	26
4.3 Cloud.....	27
4.4 Grids & clusters.....	27
4.5 Distributed operating system (DOS).....	30
4.6 Sub conclusion	31
4.7 Security considerations	31
4.8 Basic informatics methodologies explained	33
5. Implementation.....	34
5.1 Linux.....	34
5.2 Beowulf cluster	34
5.3 Acquired physical components	35
5.4 Configuration and installation	36

1. Creating connections.....	36
2. Installation of Olympus	38
3. Set and employ controller.....	39
4. Install virtual machine (VM) on RPi2s with x86 cores.....	41
5. Install the Beowulf controller on teseus0	41
6. Setting Beowulf on each RPi2	42
5.5 Optional and final Implementations.....	44
5.6 Problematic	45
Prototype.....	45
Discussion	47
Further work.....	47
Conclusion	48
Acknowledgements	49
Project Management.....	50
Scrum.....	50
Sprints.....	51
Pro's & Con's.....	52
Bibliography.....	53
Appendix A - Rating	55
Appendix B – Urban Node	56
Appendix C – Creating VMs	57
Appendix D – SSH keys	58
Appendix E – commands	59
Appendix F - Business model canvas	60
BMC	61
Appendix G – Eco design	63
Ecoaudit tool – Teseus.....	63
Appendix H	69

List of figures

Figure 1 - Middleware associates	10
Figure 2 - Urban Node (See pictures of the prototype in Appendix B)	10
Figure 3 - Teseus grid	11
Figure 4 - Cortex-A7 versus Cortex-A9	16
Figure 5 - Cortex-A17 Power efficiency	16
Figure 6 - Cortex-A7	17
Figure 7 - Comparison power consumption	19
Figure 8 - Raspberry Pi 2	21
Figure 9 - OSI model	22
Figure 10 - Internal router	22
Figure 11 - Combination between router and switch	23
Figure 12 - Difference between a router and a switch.....	24
Figure 13 - Cisco switch of Teseus	24
Figure 14 - Features of Teseus switch	25
Figure 15 - Configuration of switch	25
Figure 16 - CNA example - link.....	26
Figure 17 - standard x86	33
Figure 18 - Left: virtualization - Right: Emulation.....	33
Figure 19 - Emulation and virtualization venture	34
Figure 20 - Left: Teseus components - Right: rack holder at Neàpolis.....	35
Figure 21 - Raspbian OS download.....	36
Figure 22 - Writing Raspbian OS to RPi2 memory SD.....	36
Figure 23 - Left: Add connection - Right: Add storage pool	39
Figure 24 - Left: Create virtual machine- Right: Choose architecture	40
Figure 25 - Created virtual machine	40
Figure 26 - Teseus0 control node	42

Figure 27 - Nodes in the Teseus grid (RPi2s)	42
Figure 28 - teseus1 hosts.....	43
Figure 29 - Emulation software on top of grid	44
Figure 30 - Emulation software on top of ARM hardware	44
Figure 31- Beowulf cluster of RPi2s.....	44

List of tables

Table 1 ARM vs. x86 architecture on p. 12.....	13
Table 2 - Available processors of the Cortex A-series	14
Table 3 - Comparison of SBCs – See Appendix A for all boards + calculations.....	18

1. Introduction

The paradigm of the computer hardware infrastructure must be changed. Nowadays, large arrays of x86 servers compose all of the world data centers. This kind of computer farms spends an enormous amount of energy and needs a powerful cooling system. In the last years, and within the EPS program, a new kind of urban furniture called urban node has been developed. A few urban nodes combined provide us with one powerful grid computer that must be managed. This grid is could be an ARM-based system. Today, we can construct a new kind of grid computers with this architecture: the ARM. Most world computer labs, such as the Barcelona Supercomputing Center operate this way. We want to make a study with low cost single-board computers and existing open source software on this type of architecture.

The Teseus project is a collaboration between national and international engineering students of the EPS (European Project Semester) of the EPSEVG (Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú) and Neàpolis with supervision of professors from EPSEVG.

The purpose of the project is to create a more efficient, cheaper and a scalable single computational system consisting of single-board computers in urban nodes. The project is named Teseus and it is an attempt to configure Teseus to function as the city brain and process city data more efficiently in smart cities.

1.1 Previous projects

For two years students of the EPS organization has worked on the Urban Node project and smart cities development. This project wants to embrace all aspects of a smart city. This is a very broad subject and therefore, several EPS groups have made projects focusing on different aspects that are required to designing it.

The project was initially given in 2013 by the municipality of Vilanova y la Geltrú and focused on data acquisition, intelligent structure and electronics for the EPS students, and focused on design, energy and different features for the IDPS (International Design Project Semester) students. In 2014 a new group continued the 2013 project with a different scope in terms of construction, manufacturing and business. In addition, a part of the 2014 project was dedicated to analysing the previous project¹.

Teseus is a part of the Urban Node project. The Urban Node project created at Neàpolis consists of four main parts Teseus, Hermes, Zeus and Medusa. Hermes is the improvement of mobile communications in cities, Zeus; the energy generation, Medusa; distribution of sensors around cities for big data collection and the Teseus project; distributed computing in urban nodes forming a grid network.

¹ <http://www.epsevg.upc.edu/eps-idps-projects/download/5/Article.pdf> accessed 16.05.15

1.2 Teseus and Neàpolis

Neàpolis is local company located in Vilanova I la Geltrú, Catalunya, Spain. It is an agency of innovation of technology, design and entrepreneurship. Since 2006 they have been supporting local innovation and competitiveness as well as cooperating with companies nationally and internationally. They offer many educational programs with everything required to carry them out. Another interest that characterizes Neàpolis is that they focus on aiding and backing up the entrepreneurship of small-sized and medium-sized companies to whom they also connect with an application orientated investigation and financial subsidizations². Neàpolis accepted the project of designing a Smart City and has been the creator of the Urban Node Project, hereunder Teseus for Vilanova I la Geltrú; they became interested and started working on it for a few years back.

The Teseus project focus on computational architecture development and we were given all necessary tools, a conditioned room and aid from scholar and from our supervisor and Technical Director of Neàpolis; Josep Farré. Neàpolis is situated close to the university and has close connections with the UPC department EPSEVG.

1.3 Objectives

The objective of the Teseus project is to evaluate the possibility of building a low cost, modular and scalable data server, with existing technology. The system must place special emphasis on energy consumption.

Requirements:

- Must use ARM credit card sized computers
- Must use a Linux distribution, such as Debian, Ubuntu or similar
- Must evaluate the possibility of making x86 servers with an ARM-based virtualisation system
- It is desirable to run one usual data application on a virtual server to evaluate the system performance

The objectives for Teseus was provided by Neàpolis and aim of the Teseus project is to create a distributed computer system consisting of SBC (single board computer) forming a grid but appearing as a one united system.

The goals of the project include research and investigation of possible SBC solutions, considering energy, computational power and economic efficiency. In order to connect the SBCs to function as a single system the best software solution to form a grid, must be found and tested. In addition, the system must be able to

² <http://www.neapolis.cat/index.php?id=43> accessed 16.04.15

manage a large amount of data very fast and therefore a consideration of the use of either switches or routers to connect the SBC are made. The system will be tested on a usual commercial application and integrated on a different platform.

1.4 Team

EPS students

Simone

Global Business Engineer student from the Technical University of Denmark – 20 ECTS

Quim

Electrical & Electronic Engineer student from EPSVEG, UPC, Spain – 20 ECTS

Kuan Wei

Electrical Engineer student from National Taiwan University of Science and Technology
– 20 ECTS

EPSEVG students

Joan Josep

Computer Science Engineer student from EPSVEG, UPC, Spain – 5 ECTS point

Lorena

Computer Science Engineer student from EPSVEG, UPC, Spain – 5 ECTS point

Supervisors

Neàpolis

Josep Farré – Technical Director of Neàpolis, Vilanova I la Geltrú

EPSEVG

Eva Tordera – Doctor Degree in Physics - Computer Architecture Department

Sergi Sanchez– Doctor Degree in Engineering - Computer Architecture Department

1.5 Limitations

To match the duration of the EPS project as well as the number of people assigned to the Teseus group, there are limitations to what can be achieved in this period:

- Due to the fact that none of the fulltime members in the Teseus team studies computer science, much time has been spent on learning how to understand software configurations and the progress in the software development has a lot more potential.
- The project will not focus on economic and marketing aspects.

2. Smart cities

A city in China called Shenzhen had a population of 58.000 people in 1980 and currently the population has grown to over 10 million people³. In addition experts predict that the global population will increase from 7 billion to 9 billion people in the next 35 years⁴.

If the size and population density of the cities increases, more electricity will be required overtime. Another important thing to consider is the fact that resources to produce energy are limited unless new ways of producing and using energy are made more efficient and sustainable,

When considering smart cities, it is cities that use technology to improve living standards and processes and thereby reduce resource consumption and also costs. A smart city with all its information should be able to respond and predict local and global challenges related to data that can be measured with urban nodes.

The most common data collected in smart cities relates to energy, transport, health care, weather, waste pollution and water activities or consumption⁵.

There are several ways to define Smart City and we think the following definition is the one that adjusts best: *“the use of sensors, communications, computational ability and control in some form to enhance the overall functionality of the electric power delivery system. A dumb system becomes smart by sensing, communicating, applying intelligence, exercising control and through feedback, continually adjusting.”*⁶

In order to enhance the defined *overall functionality of the electric power delivery system* a great computational network is basic to communicate the electric supply and demand and create a better city. By better we mean a more efficient and sustainable system that will be feasible however little or high the population growth will be. Hence a flexible and scalable computing system will be required to deal with any future population behaviour.

Unlike conventional network architectures, where there are minor changes in quantity and capabilities of the devices that get connected every day, Smart Cities are prone to have a more dynamic behaviour in terms of services and applications. There may be services that become unusable due to several reasons such as battery depletion, security quarantines, node unavailability or damage. On the other hand, new services may become available by attaching new equipment that may or may not have been previously taken into account for communications and data transfers. This poses obvious challenges, especially

³ <http://www.popline.org/node/340708> accessed 16.05.15

⁴ <http://www.donellameadows.org/archives/population-growth-the-last-and-the-next-35-years/> accessed 16.05.15

⁵ http://en.wikipedia.org/wiki/Smart_city accessed 20.04.15

⁶ <http://www.hindawi.com/> accessed 20.04.15

considering that a Smart City is likely to have millions of sensors and other devices to make it as efficient and sustainable as possible.

Therefore in order to organize a more efficient, smart, sustainable and reliable grid system in a city, it is essential to rely on and work with information technologies as well as virtual social networks. The entire electric grid must be associated closely with an effective middleware. The current computers may be capable to handle a huge amount of data but they are not as efficient as they would be if their architecture were scalable.

The way the middleware associates the different hardware and applications appear as shown in figure 1⁷:

Teseus takes part in the design of a Smart City by implementing a scalable computer that is capable to operate without problem regardless of receiving large or small amounts of data, as well as the possibility of new applications or services add up to the grid or terminate (either accidentally or on purpose). Scalability is the key to achieve efficiency and sustainably for future requests.

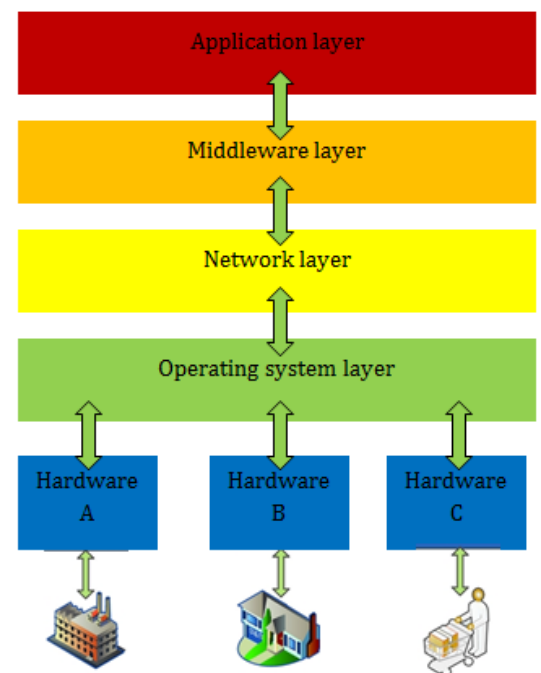


Figure 1 - Middleware associates

2.1 Urban Node project

The Urban Node project was created as a part of the transformation of Vilanova I la Geltrú (VNG) into a Smart City. The Teseus project is a part of the Urban Node project from Neàpolis.

The Urban Nodes of Neàpolis will contain a microcomputer, microcontroller, sensors and other services for the public. The microcomputer will be a part of the Teseus grid system. The Urban Node is already smart due to the fact that it produces more energy than it spends by producing electric energy via solar cells, collects information about the city environment and provides light, which can replace old street lamps⁸. The combination of using sensors, microcomputers and microcontrollers makes it possible to enhance

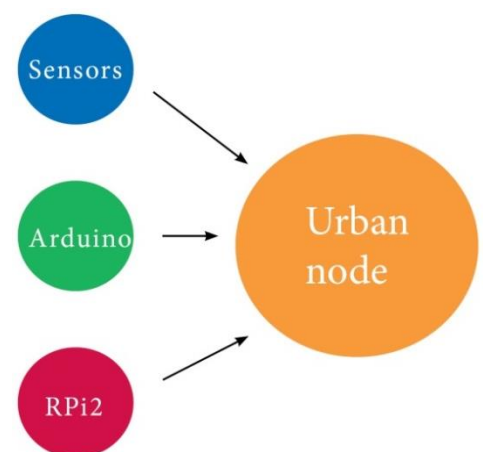


Figure 2 - Urban Node (See pictures of the prototype in Appendix B)

⁷ <http://www.hindawi.com/> accessed 20.04.15

infrastructure and the city services, hence adding intelligence to the city environment is a step towards developing a Smart City. Whereas the Urban Node project focuses on the complete urban node, Teseus focus only on the microcomputer properties and functions as the 'Brain Unit' in the Urban Node project. In the Urban Node project it is necessary for the 'Brain Unit' to process data from sensors and cameras and interactive screen; therefore it needs to be powerful enough to handle standard computer operations. Teseus is the part where everything is connected physically and virtually by compiling microcomputers in a grid to function as a single computer.

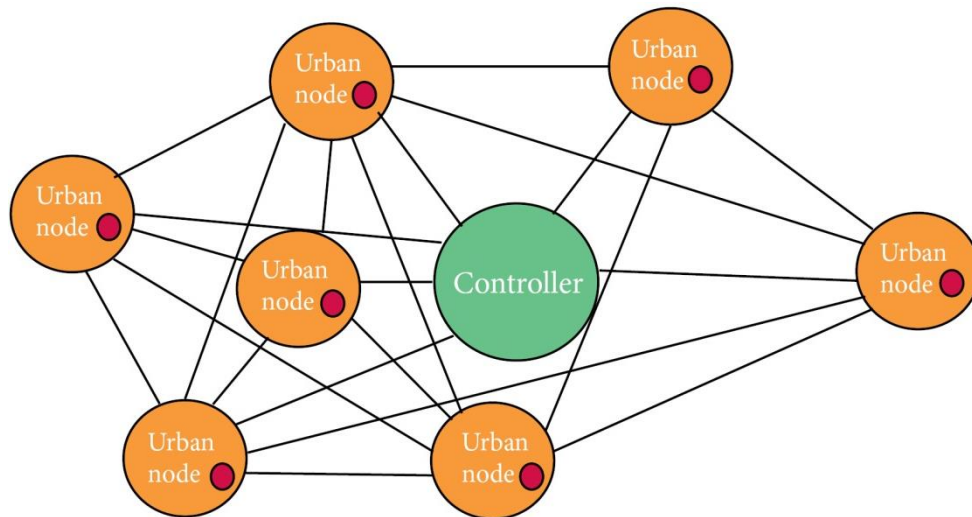


Figure 3 - Teseus grid

2.2 State of the art

2.2.1 Smart cities

The new scalable supercomputers of ARM architecture single-board computers, appearing as one system should be prepared for the amount of data that will need to be handled and managed. Open innovation of open sensor networks, living labs and fibre to the home platforms are work in progress and are some of topics that are being developed on, which will be the type of data the supercomputers will need to manage⁸. The ultimate Smart City is hard to define, but many larger cities are working towards becoming smarter; Barcelona has established smart projects and implemented sensors to inform gardeners of the level of water level required for plants in some areas. In addition, Amsterdam has Smart City initiatives of 79 projects that have the intention to reduce traffic, improve public safety and save energy. In Stockholm the Green IT program entails minimising heating cost by having energy efficient buildings, traffic monitoring

⁸ <http://www.epsevg.upc.edu/eps-idps-projects/download/5/Article.pdf> accessed 26.05.15

⁹ <http://cordis.europa.eu/docs/projects/cnect/6/270896/080/deliverables/001-D11StateoftheArtOpenInnovation.pdf> accessed 16.04.15

and developing e-services to minimise paper usage. The initiatives have the overall goal of reducing the environmental impact through IT functions¹⁰.

2.2.2 Supercomputers

Super computers come in many sizes depending on the amount of data that it needs to manage. More and more technical universities create their own supercomputer to handle the information and research done by professors, scientists and students. The supercomputer Mare nostrum found in Barcelona facilitates large amounts of data, but operates with x86 architecture.

Pleiades, NASA's own state-of-the-art supercomputer consists of 11,280 nodes of different Intel processors and can handle 5.33 Pflop/s in peak cluster. It is a distributed memory cluster connected with Infinityband® in a dual plane hypercube technology. The storage consists of 15 PB¹¹ of RAID disk configured over several cluster-wide Lustre file systems and SUSE Linux OS (operating system). The supercomputer has been made powerful enough to conduct modelling and simulation for NASA missions.¹² The comparison gives an image of what is necessary and unnecessary for a distributed system of a smart grid city data amount. The things to consider when building an enterprise supercomputer are also comparable with Teseus that needs to perform high availability and high performance computing.¹³

2.2.3 Grid systems

The Worldwide LHC (Large Hadron Collider) Computing Grid (WLCG) is one of, if not the largest, distributed computing infrastructure in the world. CERN did not have the computing resources to crunch all the data on site, so a grid computing was created to distribute the burden of running more than two million jobs per day¹⁴. This means the installation of optical fiber networks is a reality in many network areas around the world, which is a requirement for the data transmission in between Urban Nodes¹⁵.

2.2.4 Single-board computers (SBC)

More and more SBC are developed and constant innovation is made in this area in order to make them as small and efficient as possible. The focus on each board lies in different areas depending on the need of performance, speed and the machine complexity. Many SBC and cell phones uses ARM technology since it is one of the most developed companies for SBC and computing architecture developing in general.

¹⁰ http://en.wikipedia.org/wiki/Smart_city accessed 31.05.15

¹¹ <http://da.wikipedia.org/wiki/Petabyte> accessed 17.04.15

¹² <http://www.nas.nasa.gov/hecc/resources/pleiades.html> accessed 17.04.15

¹³ http://www.mellanox.com/pdf/whitepapers/Building_an_Enterprise_Supercomputer_final.pdf accessed 17.04.15

¹⁴ <http://home.web.cern.ch/about/computing> accessed 31.05.15

¹⁵ <http://www.thefoa.org/tech/ref/OSP/nets.html> accessed 31.05.15

3. Hardware

Since the Urban Node project, where a Raspberry Pi was proposed to be the microcomputer of the node, new SBCs have been developed. Therefore research must be made to find the best SBC that fulfil the requirements of Teseus. The hardware part of a computer is just as important as the software. The design of the hardware is essential when it comes to energy efficiency, computational power and power consumption that can save time and money. The following sections are an investigation of single-board computers on the market to find a match for the desired qualities described in the objectives.

3.1 Advanced 'RISC' Machine (ARM) vs. Intel x86 architecture (CISC)

Inside every physical computer you can find the Central Processing Unit (CPU) which functions as the main think-tank. The job of the CPU is to perform a series of instructions to control the hardware installed in the computer or device. Computers are complex technology and millions of executions of instructions are made to make it perform as we expect. Therefore, the speed and the power efficiency are considered when deciding the right CPU for devices. The power efficiency will affect battery life and the speed will affect the user experience and thereby the processes¹⁶. The following table is a comparison of the two most used computer architectures ARM and x86. The main difference between the two and the feature that poses the large difference in power consumption is the type of instruction set. X86 operates with a complex instruction set on direct memory, while ARM use a reduced instruction set that only use a few instructions for saving and loading data from the memory¹⁷. Hence the difference in the price and power consumption, however one standard x86 computer of Intel is equivalent to 5-6 single-board computers¹⁸.

	ARM	X86
Year of creation	1980	1980-1990
Producer	ARM Holdings	Intel
Computing at	Register	Memory
Instruction set	RISC	CISC
Average power consumption	3W	45W (i7)
Hardware	Simple	Complex
Compiler	Complex	Simple
Price	35-199 USD ¹⁹	1400USD ²⁰

Table 1 ARM vs. x86 architecture on p. 12

¹⁶ <http://www.androidauthority.com/arm-vs-x86-key-differences-explained-568718/> accessed 21.04.15

¹⁷ <http://stackoverflow.com/questions/14794460/how-does-the-arm-architecture-differ-from-x86> accessed 21.04. 15

¹⁸ Josep Farré – Technical Director of Neàpolis 23.04.15

¹⁹ <http://iqjar.com/jar/an-overview-and-comparison-of-todays-single-board-micro-computers/> accessed 22.04.15

²⁰ <http://computerpricetrends.com/14083-intel-xeon-x5570-nehalem-2-93ghz-4-x-256kb-l2-cache-8mb-l3-cache-lga> accessed 22.04.15 (The price of the processors used in the state-of-art supercomputer from NASA)

On the basis of these factors ARM is the chance of succeeding in building a low cost and scalable data server.

3.2 Energy efficiency

3.2.1 ARM families

For High-Performance Applications Processing, all current electronic boards available nowadays use Cortex-A Series for Mobile and Enterprise Markets²¹ and neither Cortex-R²² nor Cortex-M²³ families of ARM provide any option that supports virtualization. Therefore the A-series will be investigated further. It will be explained in the following sections why virtualization is an indispensable requirement.

The following chart divides the ARM Cortex-A products available into categories of specialty. Cortex A5, A7 and A53 focus on efficiency in contrast to high performance which will have higher power consumption and this is not what we are searching for.

High efficiency is required over high computational power because the goal is to implement a powerful scalable computer with the combination of several efficient computers that are connected together. The result is cheaper and more efficient than a larger computer that has a similar computational power.

The table below gives an overview of the different processors in the Cortex-A family and their characteristics.

Table 2 - Available processors of the Cortex A-series

	Virtualization	Architecture	Efficiency	Performance	Bits
Cortex-A72	Yes	v8-A compatible with v7-A	Good/high	Highest(v8-A)	64/32
Cortex-A57	Yes	v8-A compatible with v7-A	Slightly better than 15	Higher than 53 and so much better than 15 (x2 in full load)	64/32
Cortex-A53	Yes	v8-A compatible with v7-A	More than A7	High. Better than A7 Lower power than A9	64/32
Cortex-A17	Yes	v7-A		Can be scaled to higher f, 60% better than A9 with same f.	32
Cortex-A15	Yes	v7-A		High. X2 A9	32
Cortex-A9	No/ not mentioned	v7-A	High	50% higher than A8 and a bit higher than A5	32
Cortex-A8	No/ not mentioned	v7-A			32

²¹ <http://arm.com/products/processors/cortex-a/index.php> accessed 20.03.15

²² <http://arm.com/products/processors/cortex-r/index.php> accessed 20.03.15

²³ <http://arm.com/products/processors/cortex-m/index.php> accessed 20. 03.15

Cortex-A7 ²⁴	Yes	v7-A	Very high, much more than A9	20% higher than A5 Similar to A9	32
Cortex-A5	No/ not mentioned	v7-A	Most efficient ARMv7-A.50% higher than A8	Lowest power ARMv7-A 30% lower than A9	32

The processor with the highest efficiency is Cortex A53. Cortex A5 has the highest efficiency in the v7 (version 7) architecture, but the lowest computational power and Cortex A7 is less efficient, but has a higher computational power.

Best choices for efficiency are Cortex A5 and A7 for v7 architecture and Cortex A72 and A53 for v8 architecture, but currently only single-board computers are installed with processors of v7.

3.2.2 The need for virtualization

It is essential that the board chosen is capable of virtualization and if possible it has to have virtualization instructions, so as to facilitate further work elaboration of the OS. Several options have been discarded for not meeting the desired requirements in terms of efficiency and virtualization. This includes Cortex A5, A8 and A9, despite Cortex A5 being the most efficient among those using v7 architecture. More details of virtualization can be found in the software section.

3.3 Performance

When it comes to computational power of the ARM family, Cortex-A7 is a good option since it has a 20 % higher computational power than the Cortex-A5, which is the most efficient in the v7 architecture and similar to Cortex-A9, but with the difference that is much more efficient. However, Cortex-A15 has a much higher performance than Cortex-A7 in the v7 architecture.

Cortex-A7 has a similar performance to Cortex-A9. As shown in figure 4 the efficiency of Cortex-A7 is much higher and their performance resembles.

²⁴ http://en.wikipedia.org/wiki/ARM_Cortex-A7 accessed 20.03.15

Cortex-A7 Power Efficiency Relative to Cortex-A9

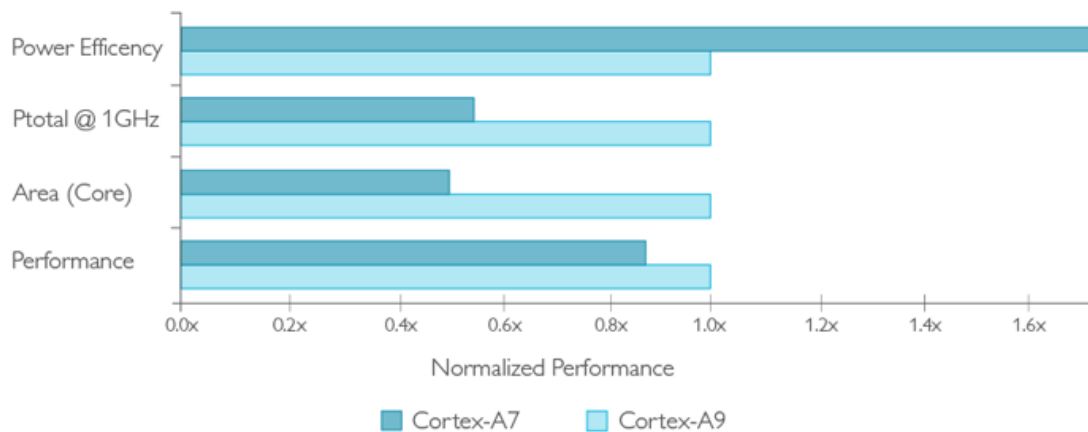


Figure 4 - Cortex-A7 versus Cortex-A9²⁵

All the processors of v8 architecture have a higher performance and they support virtualization, but as previously mentioned none of the SBCs is v8 architecture.

Cortex-A17 Performance Relative to Cortex-A9

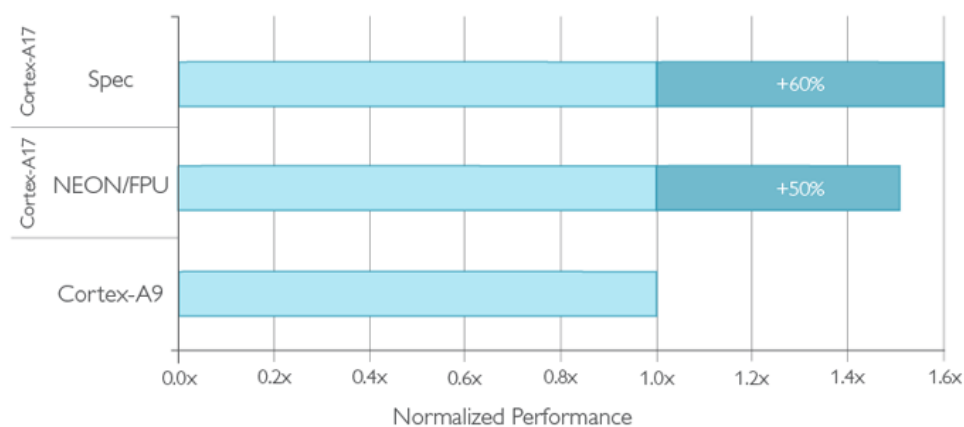


Figure 5 - Cortex-A17 Power efficiency²⁶

Since the performance of Cortex-A9 is similar to Cortex-A7, it is interesting knowing that Cortex-15 and Cortex-17 are superior to the performance of Cortex-A7's. However, overall Cortex-A7 is still a better choice, since our main objective is efficiency.

²⁵ <http://www.arm.com/products/processors/cortex-a/cortex-a7.php> accessed 15.04.15

²⁶ <http://www.arm.com/products/processors/cortex-a/cortex-a17-processor.php> accessed 15.04.15

3.4 Single Board Computer (SBC)

As mentioned in the comparison between ARM and x86, the best option of achieving low power consumption is by using single-board computers of ARM architecture. Thus SBC's containing ARM processors have been researched and rated in order to assemble a unique computer with the lowest energy consumption possible, without sacrificing too many features.

3.4.1 Rating

An elaboration of the parameters considered can be seen below.

- **Price:** Is an important parameter to the project due to the fact that a low cost and high performance board is needed. For example, even the smallest supercomputer may cost more than 1000 euros, but with an ARM SBC it could be possible to create our own unique computer system and thereby saving a great deal of money.²⁷
- **SoC:** System on a Chip integrates all components of a computer or other electronic systems into a single chip. Different companies use different SoC to make it suitable for their software and hardware products and therefore it is expected that the SoC aims for the best and therefore it has a medium weighted score.
- **CPU:** Works as the brain of the board and is very essential thus the high weight. More details will follow.
- **GPU:** The Graphic Processing Unit is not so important to the Teseus project, since there is no direct need to use the boards neither to solve large problems, calculate big questions nor to create many visual images.

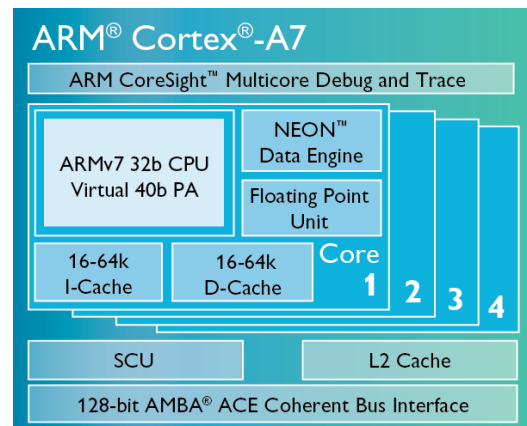


Figure 6 - Cortex-A7²⁸

²⁷ http://www.softlayer.com/info/high-performance-computing?utm_source=google&utm_medium=cpc&utm_content=Supercomputer_-_Supercomputer&utm_campaign=PPC-GLOBAL-HPC&utm_term=%2BSupercomputer&matchtype=b accessed 15.4.15

²⁸ <http://www.arm.com/Cortex-A7-chip-diagram-LG.png> accessed 16.4.15

- **RAM:** Random Access Memory is the place where the CPU directly exchanges data and therefore it is an important parameter. The size of RAM is of great importance, since the more capacity to store data, the more elasticity to solve more tasks at same time.
- **Connectivity:** States how the boards can connect. Ethernet is standard, but state of the art supercomputers use InfiniBand in high-performance computing because it has very high throughput and very low latency.²⁹
- **USB:** *Universal Serial Bus*. All the boards have 2-4 ports and therefore the significance of this parameter is of low importance and the ports will not be much of use in this project.
- **GPIO:** *General Purpose Input / Output* is not important to our project³⁰. It may matter for other projects to have output control or receive information from other devices, but Ethernet creates the fastest connection to the other boards.
- **Power consumption** is quite important in our project and is taken into consideration in one of the following sections.
- **Virtualization** is the most important parameter in this project. There is a need for running multiple processing on all boards to create a unique computer that functions as one. See details of virtualization in the software part of the report.

Value criteria	Weight	RASPBERRY PI 2	score	Banana Pro	score
Price	5	\$35	9	\$68	5
SOC, system on chip	4	Broadcom BCM2835	8	Allwinner A20*	7
Processor	9	A 900MHz quad-core-A7	8	ARM® Cortex™-A7 Dual-Core	8
GPU	3	Mali400MP2	6	Mali400MP2 Complies with C	6
RAM	7	1GB LPDDR2 SDRAM (2x memory)	7	1GB DDR3	6
Connectivity	3	Ethernet port	5	Ethernet port	5
USB	2	4 USB ports	10	2 USB 2.0 host, 1 USB 2.0	5
GPIO ports	3	40 GPIO pins	9	40 pins	9
Power consumption	8	5V, 130mA~350mA	8	5V @ 2A	4
Virtualization	10	Yes	10	Yes	10
Weighted sum	54		44,2		34,4

Table 3 - Comparison of SBCs – See Appendix A for all boards + calculations

Table 3 demonstrates some of the features the electronic boards³¹ contain, and each has been given a weighted score to rate the importance of the properties. Every board has obtained a total score based on

²⁹ <http://en.wikipedia.org/wiki/InfiniBand> accessed 16.4.15

³⁰ <https://www.raspberrypi.org/documentation/usage/gpio/> accessed 16.4.15

³¹ RASPBERRY PI 2: <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/> accessed 15.04.15

Banana Pro: <http://www.lemaker.org/>

their properties. The weighted score varies from 1-10, 1 being the lowest and 10 the highest score obtainable. The processor and the ability to virtualize weigh heavy as well as the price, hence these three parameters are the most important to consider. The SBC that received highest score is the Raspberry Pi 2.

3.4.2 Power consumption

An initial market research has been performed based on the year of production, since technology is in rapid development, and updates of products often results in better products. This led us to 5 different boards; Odroid C1, Banana Pi and Banana Pro, Raspberry Pi 2, Beaglebone Black and Cubieboard 4.

The Odroid C1 board uses a Cortex-A5, which does not support virtualization that is a requirement for our project. Therefore it was ruled out despite its interesting properties. The remaining 4 board's power consumption is displayed below.

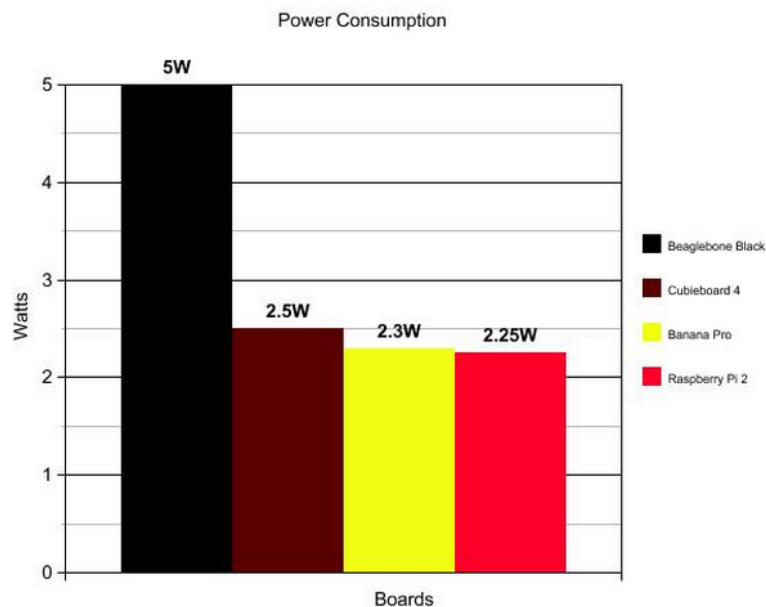


Figure 7 - Comparison power consumption

As shown in figure 7 we can see RPi2 and Banana Pro have approximately the same energy consumption, but since RPi2 had the lowest price and its high score we decided that RPi2 was the most convenient electronic board to purchase. It was also checked that its architecture met the requirements commented in the previous points.

The processor in Banana Pro uses the same ARM family as RPi2, but RPi2 uses a quad core while Banana pro only uses a dual core. When comparing the Cubieboard 4 with the selected RPi2 the processor is also

Cubieboard 4: <http://cubieboard.org/model/cb4/>

Beagle bone black: <http://beagleboard.org/BLACK>

ODROID-C1: http://www.hardkernel.com/main/products/prdt_info.php

similar as it uses Cortex 15 and can work with Cortex A7 implementing and ARM application called ARM big.LITTLE³². Since RPi2 is simpler and consumes less energy it is still chosen as the best option.

3.4.3 Computational Power

The average power consumption of an ARM processor is around 3W and the SBC of this project, the RPi2 card consumes 2.25W. Six of them equal a standard x86 computer in computational power. A standard x86 consumes 45W and six RPi2 consumes 13.5W, which is almost three times less consumption of energy. The RPi2 grid is scalable so that you can connect or disconnect as many as necessary. When the amount of data we desire to handle is lower, the less computational power is required, that means less RPi2 will work which is extra energy one can also save.

For a system that requires all the computational power 24 hours a day, the calculation of the amount of watts used per month is the following:

Standard x86 computer

$$0.045 \frac{kW}{h} \cdot 24h \cdot 365days = \mathbf{394.20 kWh per year}$$

6 RPi2 (computational power equivalent to one x86³³)

$$0.0135 \frac{kW}{h} \cdot 24h \cdot 365days = 34992000 Ws = \mathbf{118.26 kWh per year}$$

The prime objective of saving energy is clearly a success. Below, we now see how much money we save per month if the cost per KWh is 0.12€.

$$394.20 kWh \cdot 0,12€/kWh = 47.31 €$$

$$118.26 kWh \cdot 0,12€/kWh = 14.19 €$$

We can save 33.12€ every month and we see it is a remarkable amount of money saved. The success in this second objective is a direct consequence of the first one. In addition, it is crucial to remember that the best of this computation configuration is that it is scalable and it is unlikely that all SBC in an array are necessary twenty-four hours a day. While an x86 processor would have to consume almost all its power even to run the simplest application, the RPi2 cluster can adjust its computational requirements every time new data needs to be handled. Therefore the consumption would be even lower than what is calculated above. On

³² <http://www.arm.com/products/processors/technologies/biglittleprocessing.php> accessed 15.04.15

³³ Josep Farré – Technical Director of Neàpolis 05.06.15

the other hand, if new large demands of extra power were to happen, more SBC can be added to supply such demand. Yet, this is the beginning of the project and so far only five SBC are connected among them. This is meant to be forwarded to higher scales at big data centres, labs, universities and basic home users. The more data we wish to handle, the more sense it makes to acquire an array with thousands of SBC interconnected.

Regarding sustainability it is remarkable to mention that the more the price per kWh decreases the more companies will be willing to adopt the Teseus project. That will result in a major reduction in the use of energy. This will not only benefit the communities, the home users and the computer centres, but overall also planet earth.

3.4.4 Sub conclusion



Figure 8 - Raspberry Pi 2

34

The SBC with the highest score is RPi2. It has the lowest power consumption, is the fastest at the lowest price and allows virtualization.

The RPi2 has the lowest power consumption amongst the four boards that was considered, and it uses Cortex-A7, which meets the requirements of virtualization and decent efficiency performance. Therefore it is assumed that RPi2 is the best electronic board available with a low price as requirement. The board also supports virtualization to make it possible to run multiple processes at the same time and therefore an implementation of a grid can be made. The way this grid is implemented is further research.

The board choice was based on a compromise of what was available and what met the necessary requirements. The wish for high performance was there, but it also comes with high power consumption. However, by compiling several boards gives us more processing power that can be scaled up or down depending on usage. This way the energy consumption will be lower than any similar computer set up system available.

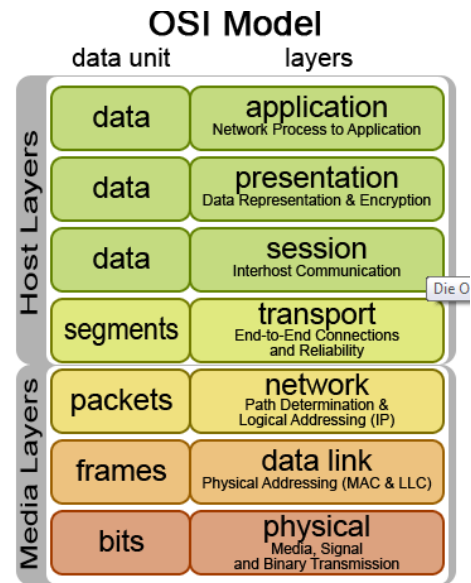
³⁴ <https://www.raspberrypi.org/raspberry-pi-2-on-sale/> accessed 20.4.15

3.5 Switch vs. router

3.5.1 Router

A router is a networking device that forwards data packets between computer networks. When a data packet comes, the router reads the address information of the packet to determine the destination. Hereafter, the router takes the information to look through the routing table and directs the packet to the next network. The address table is located in the network layer in OSI Model, as we can see in figure 9³⁵.

Figure 9 - OSI model



The router will take certain step to get the information it needs to send it the right way. When a packet arrives at the input port of the router physical header is removed to forward it to the link layer. Then, the link layer removes the next header and searches the address to see if the packet matches the same network of the router or another network. In the cases where the address matches the router network, the packet is forwarded to the switch. When the packet needs to be sent to another network, the router removes the header and forwards the packet to the network layer, and searches for the correct network and in the routing table to know where to forward the packet. Figure 10³⁶ shows the internal router and the ports.

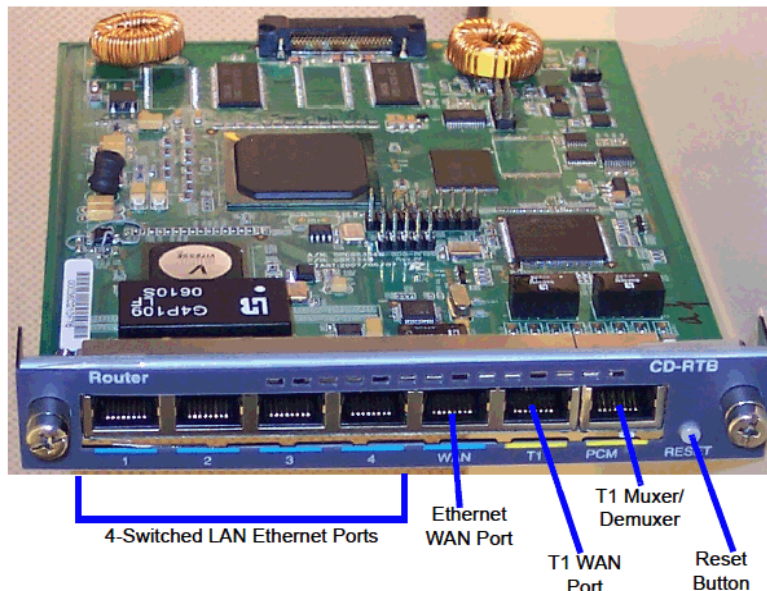


Figure 10 - Internal router

³⁵ <http://af.wikipedia.org/wiki/OSI-model> accessed 20.4.15

³⁶ <http://www.univergesv8100.com/nec-sv8100-applications/nec-sv8100-series-inrouter/> accessed 20.4.15

A router forwards packets between networks by calculating the best path and reading IP addresses, which happens in the network layer. To conclude a router uses WAN, Wide Area Network, which means that it connects LAN networks.

3.5.2 Switch

A switch is a computer networking device that connects devices together locally on a local area networks, LAN, by using packet switching to receive, process and forward data to the destination device. Switch processes packets in the Data Link layer, see Figure 9.

It uses hardware addresses called MAC addresses, which is an identifier to each device, which is unlike a router that makes use of IP addresses. Every device has one unique MAC address.

The function is the same as the router, but when a packet arrives at the data link layer, the switch recognizes that the packet is for the local network and forwards it to the switch. Here the switch will search for the data link header to know which MAC (destination) to send the packet to.

An example of a network connection through a router and switch is shown on Figure 3. Here is the internet connected with a Router ADSL, and this is connected with two switches to two different networks.

Therefore, if a packet is sent to Aula6 it will arrive first at the router, which identifies which network it is located at and afterwards sends it to the Switch_Aula. The switch will look for the MAC address of Aula6 and forward it to this computer device.

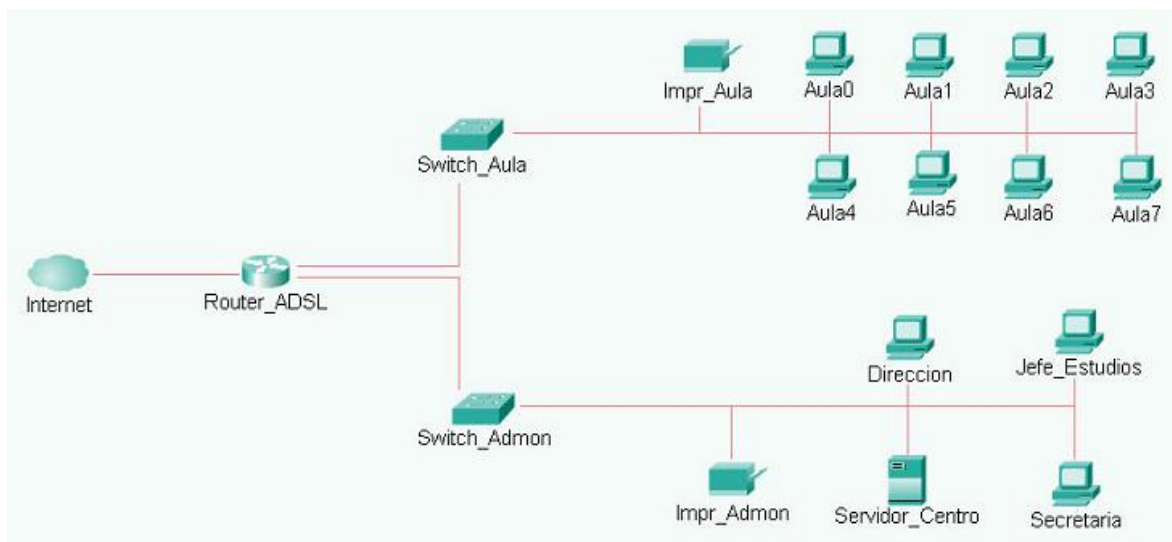


Figure 11 - Combination between router and switch

3.5.3 Sub conclusion

The main difference between a router and switch is that router forwards networks IP addresses and switch forwards MAC addresses, so routers operate on layer three and switches operate on layer two in the OSI Model. Another difference is that switches are for LAN and routers for WAN. Furthermore, routers transmit packets and switches transport frames, more differences are shown below in figure 12.

	Router	Switch
Layer	Network layer (layer 3 in OSI Model)	Data Link Layer (layer 2 in OSI Model)
Table	Store IP address in routing table and maintain addresses as its own	Store MAC addresses in lookup table and maintain addresses as its own
NAT	Can perform NAT	Cannot perform NAT
Allows WAN	LAN and WAN	LAN
Speed	1-10 Mbps (wireless) 100 Mbps (wired)	10/100 Mbps, 1 Gbps
Transmission unit	Packet	Frame

Figure 12 - Difference between a router and a switch

3.5.4 Our case

For our project we decided to connect the different boards to a switch, since one of the goals is to connect different devices in the same network and connect the switch which allows connection to the internet with any router. This means that we can control our own network with a switch configuration as needed.

In the Teseus project an implementation of a cisco catalyst 2160 series switch will be made. Figure 14 shows the switch features:

However, the switch is a three layer switch, which means it is able to operate at layer three with IP addressing instead of layer two MAC addressing that is used by the most common switches.



Figure 13 - Cisco switch of Teseus

In the case of the Teseus case it means that the difference between a router and switch must be updated because the L3 (third layer) switch is very similar to the router.

The difference is the inter-network communication. To set a connection to the internet you need a router because the switch is still focused on LAN communication and does not support WAN.

Figure 14 - Features of Teseus switch

4. Software

In order to accomplish our objective that is scalable and an existing technology, we are looking into three possible server systems to manage Teseus, with emphasis on energy consumption and that it is compatible with a Linux distribution.

4.1 Configuration of switch

The switch can be configured via a HTTP (port 80 connection). As the switch has an IP address, it is possible to establish a connection to the switch's port 80 and configure it via a browser. This is the easier way to configure the switch and also the option chosen for the project.

Console (Serial connection): The oldest way to connect the switch is to plug a cable (serial) between the switch and a host. The host opens a direct serial connection to the switch and configures it with console commands. This is the most secure way to configure a switch, but if you have a big network to manage it is the most inefficient way to configure it. An example can be seen below in figure 20:

```
Switch>enable
Switch#configure terminal
Switch(config)#interface vlan1
Switch(config-if)#ip address 172.16.10.2 255.255.255.0
Switch(config-if)#exit
Switch(config)#ip default-gateway 172.16.10.1
```

Figure 15 - Configuration of switch

Features and Capabilities

Easy to use and upgrade, these fixed configuration Fast Ethernet access switches offer superior Layer 2 threat defense capabilities and basic Layer 3 static routing with 16 routes. They also offer:

- 2x1GE uplink
- 24, and 48 Fast Ethernet port configurations
- Advanced QoS, rate-limiting, Access Control Lists (ACLs), IPv6 management, and multicast services
- IEEE 802.3af PoE with up to 15.4W per port for up to 48 ports

Simple to Configure, Operate, and Manage

- A wide range of software features to make business operations easy, highly secure, and sustainable, and help deliver a borderless networking experience
- Cisco Smart Install to simplify deployment
- Cisco Auto SmartPorts to automate device configuration

Better Investment Protection and Savings

- Cisco EnergyWise technology to promote energy savings
- Lower total ownership costs for enterprise, midmarket, and branch office networks with competitive connectivity options, security, and QoS capabilities
- Limited lifetime hardware warranty

CNA (Cisco Network Assistant): CNA is a program owned by cisco that adds a third way to do the configuration jobs. It is installed on a “network management node” (the host where the network manager works for example) and it creates a typology network map on your computer. With the typology done it is able to set a connection (telnet for configuration ftp (file transfer protocol) for download/upload the firmware) to manage the network easily. It is considered not worth the time (and money) for our “small” project, although it is the perfect choice for big networks with lots of nodes and devices. Example;

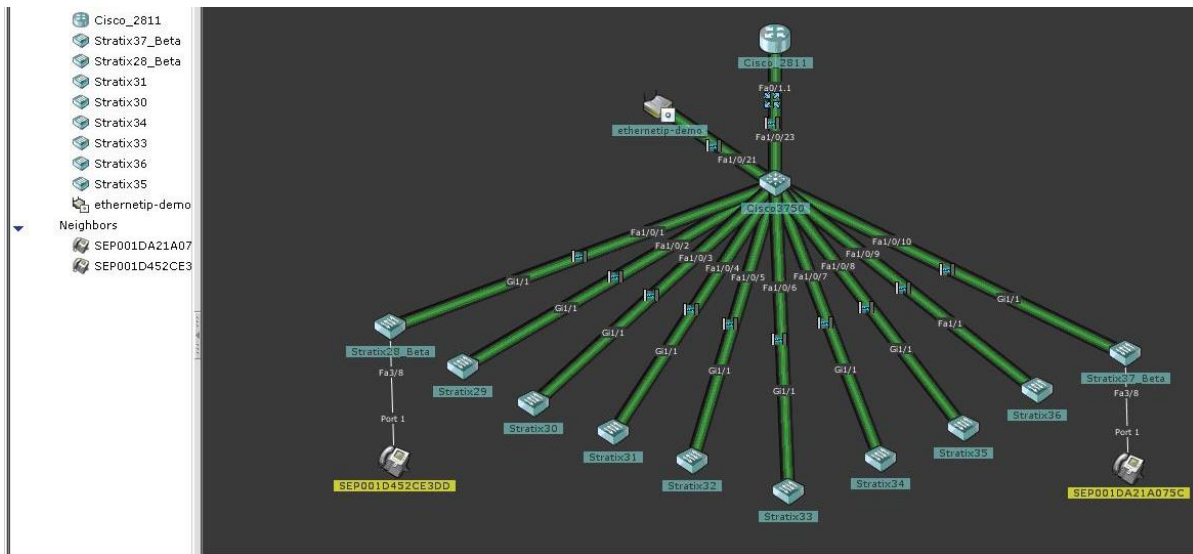


Figure 16 - CNA example - link

4.2 Virtualization

The single board computers in this project are chosen based on the virtualization ability on ARM boards available.

Virtualization allows distribution of running processes on the connected hardware server via a hypervisor. Virtualization allows the running of multiple OS and applications on a single server. The CPU, storage and networking hardware are gathered and delivered to the virtual machines via the hypervisor software. The hypervisor will make the connection between applications and OS with hardware resources such as the processors, memory and disk³⁷. Each running application will dynamically receive the resources it needs to achieve the highest performance. Therefore the server system will be running at its full capacity and for standard situation less computers are needed and this will save energy consumption. Virtual machines can be transferred to other servers while running and it is known as live migration³⁸.

³⁷ <http://www.smallbusinesscomputing.com/testdrive/article.php/3819231/What-is-Virtualization-and-Why-Should-You-Care.htm> accessed 21.4.15

³⁸ <http://www.vmware.com/virtualization/virtualization-basics/what-is-virtualization> accessed 17.4.15

4.3 Cloud

Cloud computing is an informatics system based on utility and consumption of different computer resources. It allows management of your files and use applications with no installation needed on any computer.

There are three layers in cloud computing, the infrastructure, platform and application. Companies used them differently depending on what they offer. The infrastructure layer is where the cloud hosting lies. Cloud computing can access computer power when you need it. An example is a company who owns a website of a cloud host (external server); depending on how many are visiting the website you can scale the service up or down. Cloud computing is in contrast to how it was before when you were paying for a server service when using it or not. The billing of a cloud service is easy and similar to paying for gas or electricity by taking the electrical power from the power grid. Cloud computing has three advantages; it is scalable, instant and saves you time and money and is there to use whenever it is needed.³⁹

In addition, cloud computing have different benefits compared to having your own server:

- Free applications
- Security
- No big storage needed
- Faster (it depends on your Internet connection)
- Access to all your upload information
- No need to download for using

This would benefit the involved parties of Teseus. However, with cloud you can only share folders, but it is a requirement that you can share computing power as well, therefore cloud computing is not the optimal solution for Teseus.

4.4 Grids & clusters

4.4.1 Comparison of computer systems

The difference between grid and cluster is that a cluster is a group of computers connected by LAN and a grid is broader, wider scale and can be geographically distributed. A cluster will typically exist with the same OS and a cloud and grid will consist of very different hardware configurations⁴⁰.

A grid is composed of many clusters, which is typically one set of nodes in one location. Essentially, the grid connects the clusters since they do not really trust one another.

³⁹ <https://www.youtube.com/watch?v=QJncFirhPg> 29.03.15

⁴⁰ <http://stackoverflow.com/questions/9723040/what-is-the-difference-between-cloud-grid-and-cluster> 18.03.15

In order to connect our single board computers of ARM architecture to function as one computer, it is considered whether cloud, grid or distributed system will be the best solution for our SBC cluster.

4.4.2 Cluster types

HPC also known as *High Performance Computing* clusters are used to increase performance by splitting tasks across many different nodes in a cluster. Beowulf is one of the most well-known configurations of Linux clusters; it behaves similarly to a single computer and is very simple. *High availability* (HA) clusters has the purpose of improving the accessibility to the application if hardware failure, and the best solution Linux can offer is Heartbeat.

Load balancing, another type of cluster, distributes the workload across multiple resources⁴¹. One of the best examples is the Linux Virtual Server (LVS) which is a scalable server built on top of a cluster, therefore it is very similar to the Beowulf but with additional load balancer functions.

4.4.3 Smart grid technology

In the world today we see an increase in the use of energy from electronic devices. These include computers, televisions and other electronic devices, but the power grid is busy and peaks in activity in certain hours of the day when people are active, and it is barely used during the night. Grid technology is used to distributed and harvest energy in order to manage the workload being made.

Computers are made with a certain amount of resource to pull from but is rarely used to the fullest, and in most organizations there are great amount of underutilized computer resources⁴². Grid computing creates an opportunity to exploit these resources and thereby increasing the efficiency of energy usage.

4.4.4 Grid computing

The general idea is to take advantage of computer processing power that is not currently being used and utilize their resources and processing power when they are not busy to handle other processes. Grid computing means to have a number of computers that does not need to be on the same location or the same sort of computer and connect/link them together by using grid software. This way you can submit jobs to the grid and the software will match the job to the available computers in the grid. When the computers are not busy they will process the job.

The software makes the grid because without the software it is only a network of computers or cloud computing. Due to the change of computer clusters the distribution of tasks will be ongoing even if one should drop out the task is reassigned to another computer cluster. The process will have been saved from

⁴¹ http://en.wikipedia.org/wiki/Load_balancing_%28computing%29 07.04.15

⁴² Bart, Jacob, Michael Brown, Kentaro Fukui and Nihar Trivedi, "Introduction to Grid Computing" page 8-9.

the previous computer so nothing is wasted and thereby it keeps evolving and changing over time. New computers can be added to the system and make the grid even bigger.

4.4.5 Software solutions

In this project different software options of controlling or creating a grid have been studied in order to determine, which software is compatible with a single computer system consisting of many boards.

- MOSIX

MOSIX is a management distribution program⁴³. It has a numerous frequently asked questions.

The network needs to be secure. This makes it ideal for example a company distributing their own power resources. However, the only CPU architecture that MOSIX supports is the x86 and supports TCP/IP networks, but the Teseus project requires ARM architecture⁴⁴.

- Condor – HTCondor (-G)

Condor can manage a cluster of dedicated compute nodes of for example Beowulf. HTCondor supports high throughput that comes from long running processes that does not require user interaction⁴⁵. The jobs can be scheduled to other machines of the same architecture regardless of the underlying file system that is available to it⁴⁶. High throughput is in contrast to high performance, which is about getting as much as you can right now (a platform is a combination of an OS and a processing architecture).⁴⁷

- Globus Toolkit

The toolkit is open source and will let people enable the “Grid” by sharing computing power, databases and additional tools online across all kinds of networks and geographic distances as well as international securely and all this without sacrificing the local autonomy. The toolkit provides services to build commercial grid products and include software services and libraries for discovery, management, resource monitoring as well as security and file management.⁴⁸ The kit can be used independently or together to allow development of applications and allows users to access the remote resources as if it was on their own computer⁴⁹. The GT would have been chosen because it is possible to create your own grid system and it is used in CERN, but does not work ARM so therefore it was not chosen.

⁴³ http://www.mosix.cs.huji.ac.il/txt_about.html 29.03.15

⁴⁴ http://www.mosix.cs.huji.ac.il/faq/output/faq_q006.html 29.03.15

⁴⁵ <http://research.cs.wisc.edu/htcondor/overview/> 29.03.15

⁴⁶ <http://www.umiacs.umd.edu/labs/parallel/condorintro.htm> 29.03.15

⁴⁷ http://research.cs.wisc.edu/htcondor/tutorials/videos/2014/Intro_To_Using_HTCondor.html 29.03.15

⁴⁸ <http://toolkit.globus.org/toolkit/about.html> 29.03.15

⁴⁹ <http://searchenterpriselinux.techtarget.com/tip/Linux-clusters-vs-grids> 07.04.15

All of the above mentioned software options are open source software. However, distributed operating system and available software solutions should also be considered.

4.5 Distributed operating system (DOS)

4.5.1 Grid vs. distributed computing

The main difference between a grid and a distributed computing is the way resources are managed.

Distributed computing uses a resource manager that is centralized and the nodes work together as a single unified system. Grid computing has the structure where all clusters use their own resource manager and the system does not act a single unit⁵⁰.

4.5.2 DOS Solutions

- Bell lab's Plan 9

Supports ARM, x86-64, PowerPC, SPARC architectures. The latest release (fourth release) is dated from 2002. If an application running on Plan 9 requests for free space, the OS runs a petition to "file servers" (a machine that offers resources in the form of files) and the user is unaware of whether the application is running at local or remote resources, therefore it is 100% transparent⁵¹. Even if you run an app on Plan 9 you can't even tell where it is running, the machines can work as "CPU servers" and can run applications from other machines from the network. The "file servers" and "CPU servers" can be any machine with Plan 9 and free space in the network and you can configure them to become specific servers such as mail server, process control server (the one who knows where the process runs in the network).



- Inferno52

Plan 9 based operating system. It supports ARM and other architectures such as x86 and SPARC as well. It has a very interesting feature; **Inferno**



can run as an application in top of an existing operating system or as a stand-alone operating system. It is based in the same idea as Plan 9, the resources are distributed among the network and the user can run applications that use remote resources and doesn't even know. In comparison to Plan 9, Inferno is mostly

⁵⁰ <http://www.maxi-pedia.com/grid+computing+distributed+computing> accessed 07.04.15

⁵¹ <http://www.slideshare.net/anantn/unix-plan9-from-bell-labs> accessed 07.04.15

⁵² <http://www.vitanuova.com/inferno/> accessed 07.04.15

based on security (because the distributed system features were inherited from Plan 9). A good option if we have an operating system already installed and we only want to test it out.

- QNX53

The commercial UNIX-like operating system intended for handling real-time applications and primary oriented to embedded systems. It is not a real distributed operating system itself but it has lots of



interesting features so we decided to put it in this list too. First of all, when we talk about QNX and its features the first thought we have to have in mind is the “Transparent distributed processing” concept; that means the apps running on QNX make use of both local and remote resources for their needs in fact the app does not even know if the disk the app is accessing is local or remote. However, commercial distributions are not free.

4.6 Sub conclusion

The DOS is the best option in terms of efficiency and transparency, and in addition it is the most complex and difficult option and has the disadvantages of being unstable and the software is not yet fully released. It is still an experimental project in various labs and therefore something for the future and not an option for the Teseus project now.

The grid has the advantage of operating like the DOS as one system. If you need more power you can just add more nodes to grid. However, the disadvantage is the unanswered question of whether it can work with the required ARM architecture and also if failure happens the whole system breaks.

The cloud system has the advantage of being well established and has several free open source software available. The disadvantage is every computer works alone and you don’t share computer resources the same way as the grid. If you need more power you will need to install more servers, which is less flexible and more expensive than adding nodes to a grid.

The best solution if a cluster system is chosen to be Beowulf cluster for testing, when the DOS is more developed, testing it will be our next goal.

4.7 Security considerations

One of the major challenges for engineers today is securing personal information. Due to the three main characteristics high bandwidth connection, extensive computational power and massive storage capacity,

⁵³ <http://www.qnx.com/> accessed 31.05.15

clusters have become more widespread and powerful and have become desirable targets to attackers⁵⁴. Any computer or system in the world can be hacked, even though the more security, the longer it takes to retrieve the wanted information and they usually leave a trail it is happening. In order to have a secure system you want to consider browser and network security⁵⁵.

4.7.1 User

Depending on what type of information is needed the level of security can vary. If a person wishes or allows their computer device(s) to be a part of a grid, they need to install software on their computer that gives the permission to share information. With Globus the user decides what information, documents, and folders they wish to share with others.

4.7.2 Operation System

The OS itself has security considerations, which means that not everyone is allowed to write, read and edit. The OS has a protection system that gives authentication to the user that wish to run particular programs by having usernames, keys that is connected with a password or signature⁵⁶.

Depending on the OS the security system differs, but they have security mechanisms that protect against malicious attacks by as an example encrypting file systems and restricting the privileged levels⁵⁷.

4.7.3 Grid Security Infrastructure

The GSI – formerly known as Globus Security Infrastructure is the communication between software in a grid computing environment and it is secure, secret tamper-proof and authenticatable due to the use of asymmetric encryption. The use of digital signature technology allows resources to lock their data only to those who are supposed to have access to it. However, often a service will retrieve data from a source independent of the user, but GSI will create a new key that needs to be signed by the user and only then is it possible for the service to act on behalf of the user⁵⁸.

When managing the RPi2's you will need to install Secure Shell (SSH), which is a cryptographic network protocol for shell sessions⁵⁹ on remote machines in a secure way and provide a known key for it.

In summary the grid system has is secure enough for the average user and the user decides, as with any other application they use, which information they wish to share or not share and this means Teseus is sustainable when it comes to protection of personal information.

⁵⁴ http://www.researchgate.net/publication/228857478_Cluster_security_as_a_unique_problem_with_emergent_properties_Issues_and_techniques accessed 31.05.15

⁵⁵ http://en.wikipedia.org/wiki/Internet_security accessed 31.05.15

⁵⁶ http://www.tutorialspoint.com/operating_system/os_security.htm accessed 29.05.15

⁵⁷ <http://www.qnx.com/products/neutrino-rtos/neutrino-rtos.html#technology-security> accessed 31.05.15

⁵⁸ http://en.wikipedia.org/wiki/Grid_Security_Infrastructure accessed 31.05.15

⁵⁹ http://en.wikipedia.org/wiki/Secure_Shell accessed 31.05.15

4.8 Basic informatics methodologies explained

In order to explain the basic of the final implementation it is essential to comment and justify the informatics methodologies that will be applied so as to achieve the final computational architecture that is desired.

4.8.1 X86

Figure 17 shows a common computational system where common x86 hardware can be found and installed on it there is a common operating system such as Apple, Windows or Linux that can run ordinary applications such as Internet Explorer, AutoCAD, Microsoft Word and so on.



Figure 17 - standard x86

4.8.2 Emulation

If one desires to run application for an operating system that is not installed in the hardware that we have available, there is an option called emulation that may be useful. To emulate is to provide a translator between two operating systems that uses different computational languages so as to run application on this new operating system. Below there is an example of how to emulate an x86 operating systems from an ARM operating system

4.8.3 Virtualization

Sometimes it is very useful and convenient to split a hardware (also named physical machine) into one, several or many virtual machines. Virtualization is necessary for some informatics distributions and grid connections.

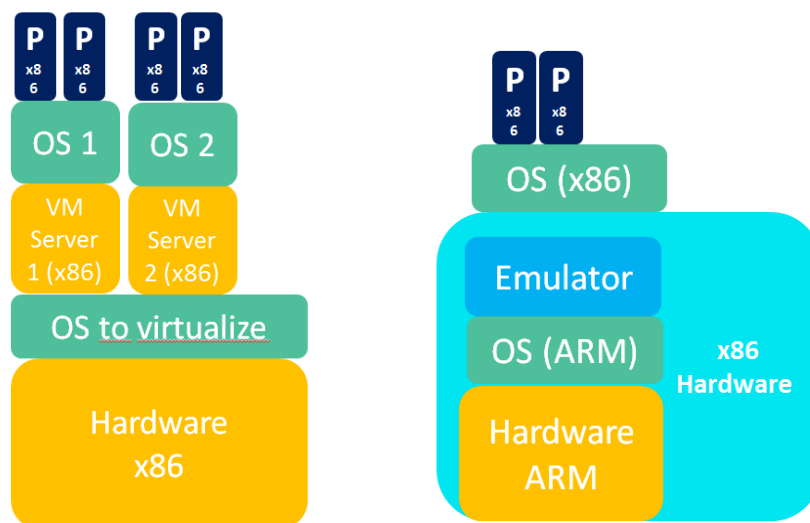


Figure 18 - Left: virtualization - Right: Emulation

4.8.4 Emulation and virtualization joint

In some occasions and more precisely in this project, it is necessary to have a virtual machine with a different computational architecture that is not the one that the physical machine provides. So as to achieve that, first it is required to virtualize and later using an emulator. It is also necessary to install a program called libvirt so that the emulation and the virtualization at the same time are possible.

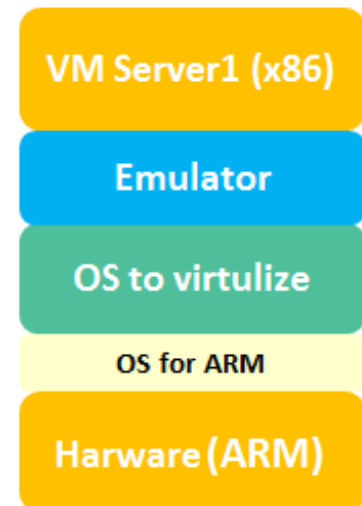


Figure 19 - Emulation and virtualization venture

5. Implementation

5.1 Linux

Teseus use Linux programs and versions of Ubuntu and Debian that has risen to prominence for use of server operating system (OS)⁶⁰. The programs used in Teseus are all of Linux distributions and therefore all commands are Linux, as it was a requirement from the product owner. A list of the most used commands can be found in Appendix E.

For Olympus, the physical machine of the controller, we use Ubuntu 14.04.2 as desktop OS. This OS is heavier than the Ubuntu server OS that is used for our nodes and the virtual controller in the grid, because it uses more programs and has a graphical interface.

Teseus0, the virtual controller of the grid, use Ubuntu server distribution, which is more likely to run dedicated programs.

5.2 Beowulf cluster

In order to create a single computational system, which means shared processing of tasks, a Beowulf computer cluster is configured to build a product that can be tested and analyzed. A Beowulf cluster is a way of sharing processes in a local area network and the result is high performance and efficient parallel computing⁶¹.

It uses open MPI (Message Passing Interface) libraries and is a project that combines technologies and resources from several other projects and used for parallel and distributed computing⁶². This is necessary to

⁶⁰ <http://en.wikipedia.org/wiki/Linux> accessed 31.05.15

⁶¹ http://en.wikipedia.org/wiki/Beowulf_cluster accessed 31.05.15

⁶² <http://www.open-mpi.org/faq/?category=general> accessed 31.05.15

reach the goal of having Teseus function as the city brain as described in the objectives.

In order to obtain grid software of our Raspberry Pi 2 that can emulate an x86 processor and can virtualize the following steps were followed⁶³. The implementation is divided into the controller's configuration and the software installation and configuration for the single-board computers (SBC).

1. Creating connections
2. Configure physical controller (Olympus)
3. Set and employ the controller – teseus0
4. Install virtual machine (VM) on RPi2s with x86 cores
5. Install the Beowulf controller on teseus0
6. Setting Beowulf on each RPi2

The work performed for each step is elaborated after the acquired physical components section.

5.3 Acquired physical components

The list below states the necessary physical components to set up the initial Teseus.

- Large rack
- SBC rack holder
- Switch
- Ethernet cables RJ45 (5 for RPi2, 1 controller, 1 Internet = 7 ps.)
- SD cards (5 ps.) (micro SD memory)
- RPi2 SBCs (5 ps.)
- PC
- Power source for 5 RPi2's



Figure 20 - Left: Teseus components - Right: rack holder at Neàpolis

⁶³ <https://www.digitalocean.com/community/tutorials/how-to-create-a-beowulf-cluster-using-ubuntu-12-04-vps-instances> accessed 31.05.15

5.4 Configuration and installation

1. Creating connections

1.1 Modification of Raspbian

The Teseus team is starting from scratch, which means after receiving the RPi2 boards the initial implementation can begin. First an installation of the Raspbian OS image must be made. After downloading it from the Raspberry Pi official website, a modification of it was made to add services, such as open SSH (Secure Shell) and the QEMU (Quick emulator). The board itself is searching for boots, but with a memory card with the Raspbian installed it is an OS. We use an image of the original OS to insure we have the OS intact should the computer crash.

Unlike with the controller, in order to configure and operate each RPi2 a specific OS needs to be downloaded and installed. This operating system is called Raspbian and can be found at raspberrypi.org. This site provides step by step set-up instructions.

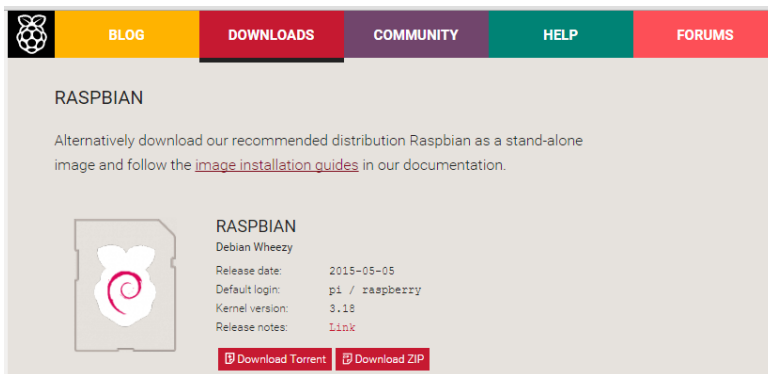


Figure 21 - Raspbian OS download⁶⁴

1.2 Installation of initial operating system

The Raspbian OS image was downloaded for Linux and by using Win32 Disk Imager the OS was successfully written to the memory cards that was inserted into the RPi2 boards, as shown in the image below.

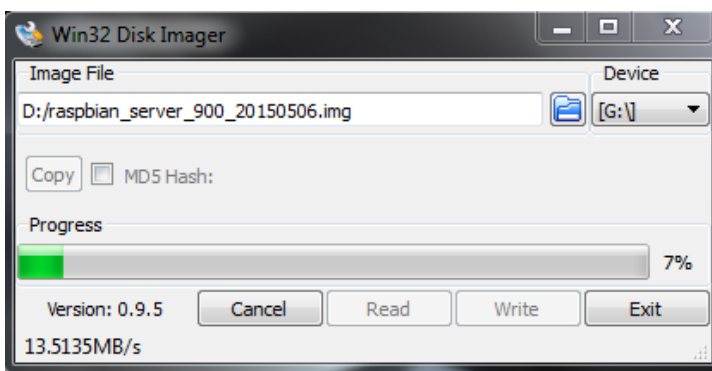


Figure 22 - Writing Raspbian OS to RPi2 memory SD

⁶⁴ <https://www.raspberrypi.org/downloads/> accessed 31.05.15

1.3 IP addresses and hostnames

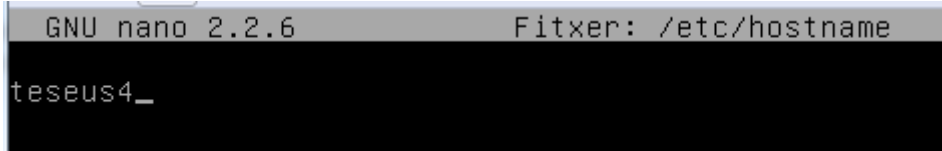
Once the Raspbian is installed in each RPi2, the configuration can be carried out. The configuration includes the hostname of each of them, (teseus1, teseus2, teseus3...) the IP and the DNS (Domain Name Server). The configurations finish with a reboot and the installation of all libraries and updates.

The IP address, hostnames and hosts must be configured and was provided by Neàpolis since that is where the network Teseus will work on.

- | | |
|------------------------------------|-------------------|
| 1. Working node - IP: 10.20.50.51, | hostname: teseus1 |
| 2. Working node - IP: 10.20.50.52, | hostname: teseus2 |
| 3. Working node - IP: 10.20.50.53, | hostname: teseus3 |
| 4. Working node - IP: 10.20.50.54, | hostname: teseus4 |
| 5. Working node - IP: 10.20.50.55, | hostname: teseus5 |

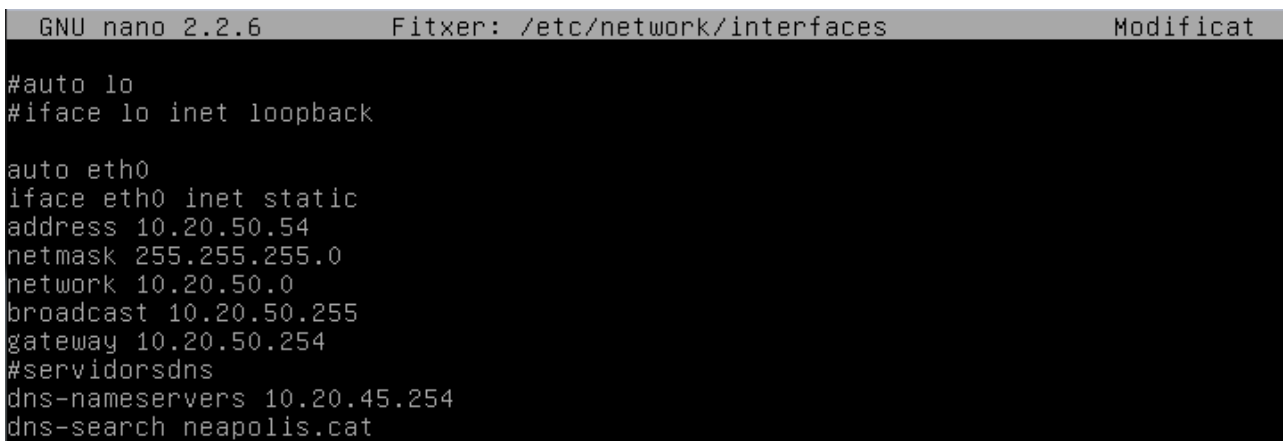
The images below demonstrate the editing of the network interfaces of teseus4 (the fourth RPi2 board). Via the PC that was lent to the team of Teseus each RPi2 were given a new hostname and IP (Internet Protocol) address in order to identify the network interface and the location address. As super user (su/root) it is possible to edit the IP address and its network location with:

```
nano /etc/hostname
```



```
GNU nano 2.2.6      Fitxer: /etc/hostname
teseus4_
```

```
nano /etc/network/interfaces
```



```
GNU nano 2.2.6      Fitxer: /etc/network/interfaces      Modificat
#auto lo
#iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.20.50.54
netmask 255.255.255.0
network 10.20.50.0
broadcast 10.20.50.255
gateway 10.20.50.254
#servidorsdns
dns-nameservers 10.20.45.254
dns-search neapolis.cat
```

```
nano /etc/hosts
```

```

GNU nano 2.2.6          Fitxer: /etc/hosts
127.0.0.1                localhost
::1                      localhost ip6-localhost ip6-loopback
fe00::0                  ip6-localnet
ff00::0                  ip6-mcastprefix
ff02::1                  ip6-allnodes
ff02::2                  ip6-allrouters

127.0.1.1                teseus4_

```

1.4 SSH connection

The image below demonstrates an SSH connection from a virtual machine to teseus1 by addressing the user pi at(@) its IP address 10.20.45.30, which was the initial address of teseus1, and directly to the already known port number 22. The establishment of connections is the first step.

```

root@hacker:~# pi@10.20.45.30 -p 22
-bash: pi@10.20.45.30: no se encontró la orden
root@hacker:~# ssh pi@10.20.45.30 -p 22
pi@10.20.45.30's password:
Linux teseus1 3.18.7-v7+ #755 SMP PREEMPT Thu Feb 12 17:20:48 GMT 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon May  4 16:13:59 2015 from clients-xsf-94.upc.es
pi@teseus1 ~ $

```

2. Installation of Olympus

2.1 Olympus as Virtual Machine Manager (VMM)

Initially Olympus was configured as the master of the whole system providing an IP and DNS. It was necessary to install an OS to run the controller; it will operate with Linux distribution. Hereafter, KVM (Kernel Virtual Machine) emulator and QEMU as virtualization were installed into the controller. In order to configure them libvirt library is required to make them work together and will it be necessary for the SBCs too.

```
apt-get install qemu libvirt-bin
```

The command below enables the libvirt runs as a daemon, which means it runs in the background to ensure that the programs run without default.

```
etc/init.d/libvirt-bin start
```

3. Set and employ controller

After the configuration of a super user for the system and the installation of QEMU and KVM, 'teseus0' was created as a virtual machine the controller, as well as the configurations of its IP and hostname. The following steps and images demonstrate the creation of a virtual machine via VMM:

- Create connection – localhost
- Create the storage pools for disc, the OS images, storage capacity and allocate 8 GB via qcow2
- Create the virtual machine – Name, ISO file from the pools to choose the OS image, define memory, summary of options: internet connection and architecture, type of VM

The first thing to do is to install QEMU. It is necessary for doing the virtual machine, enabling the administration of the virtual machine via SSH and enables KVM and QEMU to work together. Once installed the libvirt has to be executed as demon.

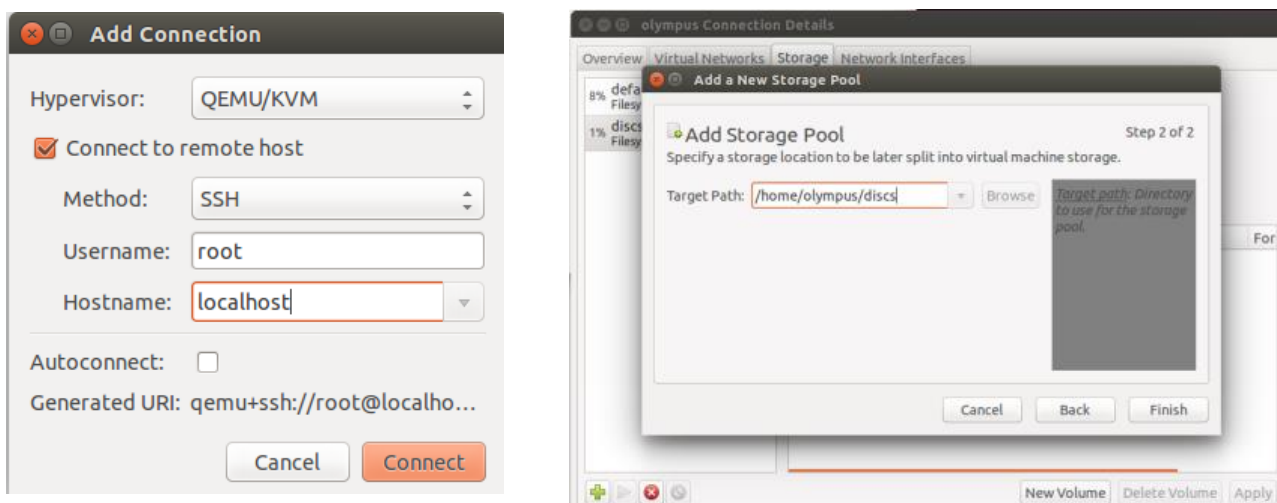


Figure 23 - Left: Add connection - Right: Add storage pool

From Olympus the VMM was turned on. The RPi2 is connected by SSH; the pools were created where the disks and the images of the OS are going to be placed as demonstrated in the two figures above.

From the website the image of the operating system that would be used to do the virtualization was downloaded. Now with the controller, it is Ubuntu Server; once downloaded it was introduced in one of the pools, after that the virtual machine is set.

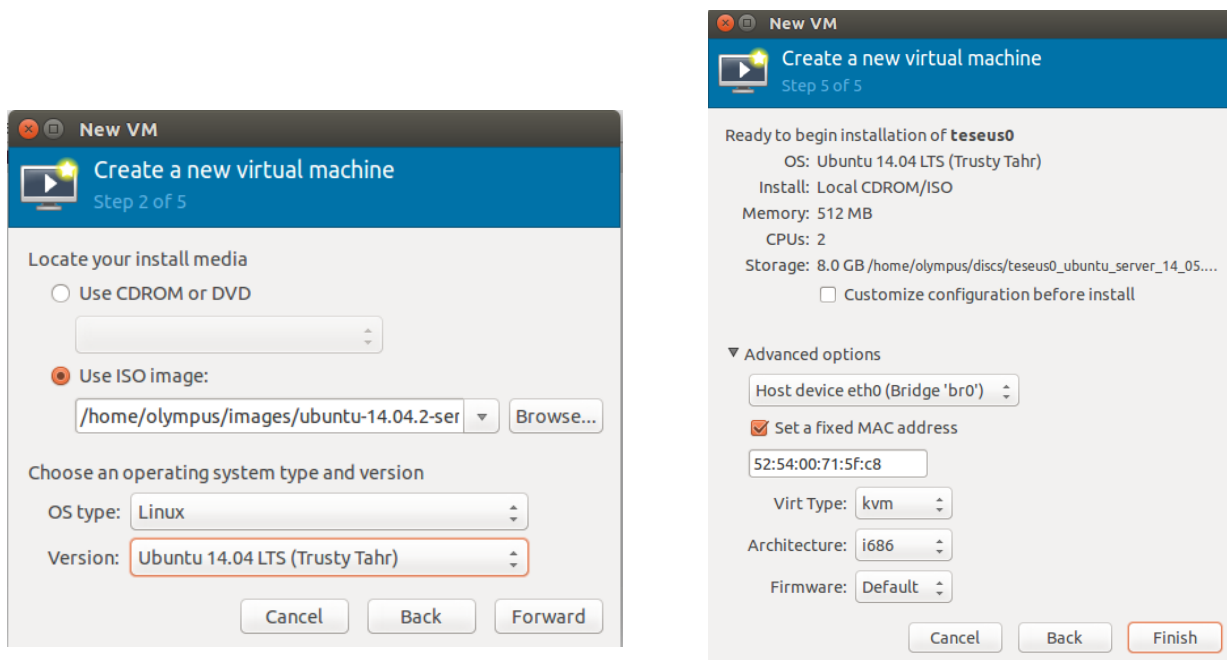


Figure 24 - Left: Create virtual machine- Right: Choose architecture

The first virtual machine was configured with the name `teseus0`. Next the image of the pool, Ubuntu Server, was selected in order to configure its desired RAM. The ram for all the SBCs is in this case 512MB. In this case two CPUs were chosen for the convenient performance of the machine. Subsequently the disk for the new machine was created.

Once finished it was needed to go to advance option and configure the network by default in bridge. The most important thing was to declare the architecture to i686 (X86). This is how the virtual machine for the controller was created.

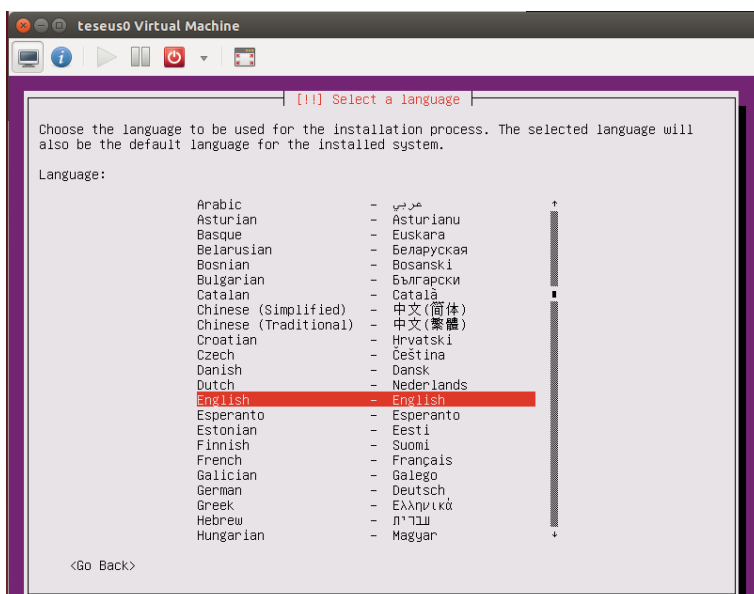


Figure 25 - Created virtual machine

The final image displays the new virtual machine.

The creation process of the virtual machine is the same for all the VM created in connection with Teseus. The specifications of the exact IP numbers an example of it will not be given in this report for confidential reasons. To view all steps see appendix.

4. Install virtual machine (VM) on RPi2s with x86 cores

Once more, the same it was carried out with the controller, QEMU and KVM will be necessary to be installed on each RPi2 in order to be able to run an x86 processor, proceed with the grid connection and be able to virtualize. The process of creating the virtual machines is just repeating the steps from above. However, in our case it OS is Debian8 and unlike the controller, the RAM for all the SBCs is 256MB. The number of CPUs is equal to teseus0, two CPUs.

The most important thing again is to declare the architecture to i686 (X86). This is how the virtual machines for each physical machine (the RPi2 hardware) were created.

At this point it was possible to proceed to install the Debian OS. The following step was to install Beowulf that, as mentioned previously in the report, is not a system itself, but a way to compile computers in a cluster.

5. Install the Beowulf controller on teseus0

The following step was to install and configure (IP, DNS and shared folders) the system Ubuntu Server. Later it was necessary to install and configure the Beowulf controller and to do so, it was required to create a common user named Beowulf.

```
sudo adduser beowulf --id 900
```

Our node cluster communication system is called Message Passing Interface (MPI) that makes it possible to share work and status information and enables parallel processing to communicate easily. Therefore, it was essential to download the following software: open MPI (mpich2), SSH and all its dependencies.

```
apt-get install ssh mpich2
```

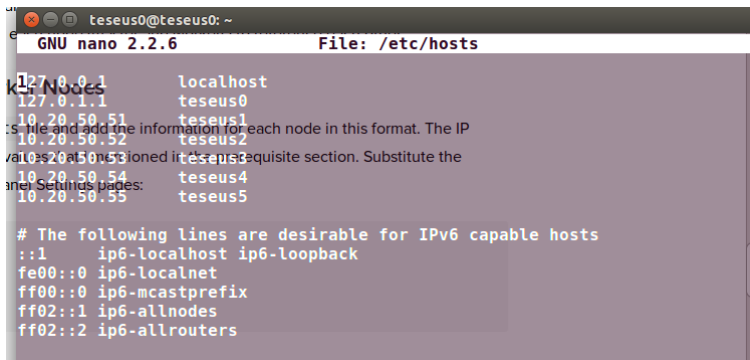
Moreover, two SSH keys were created and we tried access to one of the RPi2 (it is explained how to configure the RPi2 in the following headland) checking one of them to check up the security.

```
ssh-keygen
```

```
scp beowulf@teseus0:~/.ssh/id_dsa.pub ~/.ssh/authorized_keys_
```

Besides that, a new file was saved inside the same folder of the same user Beowulf. The saved file had to be named "host". Inside this file now is possible to add the correct IP or hostname of all the machines

belonging to the cluster. Furthermore a new folder named “bin” was saved into the home folder; and right there is where all the programs and applications that are desired to be run in the cluster are going to be placed.



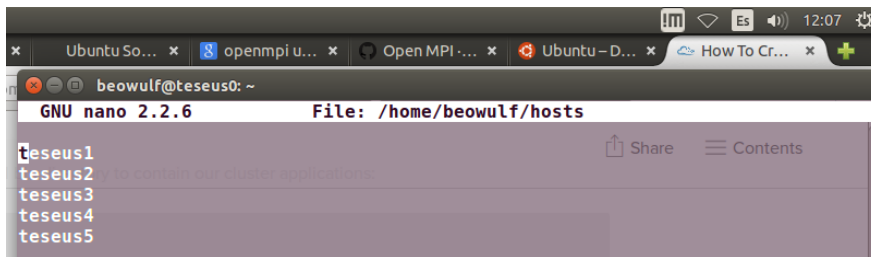
```

GNU nano 2.2.6 File: /etc/hosts
127.0.0.1 localhost
127.0.1.1 teseus0
10.20.50.51 teseus1
10.20.50.52 teseus2
10.20.50.53 teseus3
10.20.50.54 teseus4
10.20.50.55 teseus5

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

```

Figure 26 - Teseus0 control node



```

GNU nano 2.2.6 File: /home/beowulf/hosts
teseus1
teseus2
teseus3
teseus4
teseus5

```

Figure 27 - Nodes in the Teseus grid (RPi2s)

Figure 27 is the file used to distribute the task between the nodes. The threads can be distributed to these. Only the controller knows this.

This is how the controller of the cluster is configured. Once configured all the SBCs, it will be possible to run any program that works with x86. All that is required is to use the correct commands and log in as Beowulf user.

6. Setting Beowulf on each RPi2

The Beowulf system uses the SSH configuration to connect the VMs, and the open Pi (opich2) to make them work as one. This indicates that the Beowulf system will only be capable to run programs with parallelization of MPI.

The procedure to create our Beowulf system is as follows: The user for the cluster was created as: Beowulf. Later it was necessary to download both SSH and open.mpi programs with their own dependencies such as libraries. Once installed the certificate SSH of the controller was accepted. To achieve that, the controller's

public key was copied inside the systems authorized case. To conclude the configuration the keys were accepted.

The process above is repeated four more times for teseus2, 3, 4 and 5 as well as:

- Copy the keys to the nodes.
- Installation of MPI

```
sudo adduser beowulf --id 900
```

```
ssh-keygen
```

See screenshots of programming in Appendix D.

All the nodes have to be aware of the other nodes connected in the grid; therefore in addition it is necessary to copy all keys into each board so they can encrypt the messages sent interchangeably and thereby ensuring that the connection is secure. This way they know who the message comes from and therefore they will create a connection – see command 4 – authorization of keys.

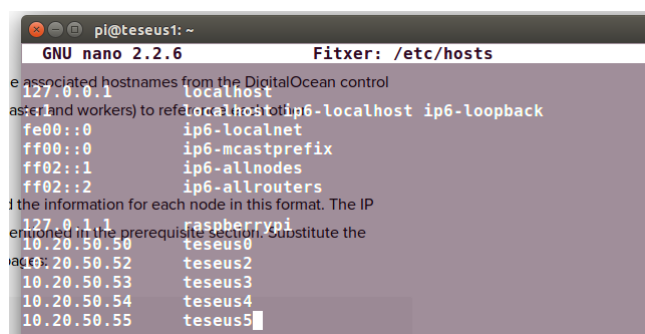


Figure 28 - teseus1 hosts

The image above shows that teseus1 knows the IP of the rest of the cluster. This is also what the DNS does, but should the DNS shut down, the boards can still run.

With the command below an execution of the CPI with 20 cores and distributes threads in the host declared in the host file, which means that the all the cores in the RPi2s is running and that one large task can be distributed and processes by all the nodes in parallel.

```
mpiexec -f hosts ifec eth0 -n 20 /home/beowulf/bin/cpi
```

5.5 Optional and final Implementations

Figure 29 and 30 are the initial architectures of Teseus that were desirable designs before a final implementation clarifies what are possible options with the technology available. Figure 29 illustrates the solution with a grid software installed on top of all the RPi2s and the emulation for X86 and virtualization on top to achieve our goal of running 86 software applications on ARM.

Our second solution is shown in figure 30 and it was similar to the first solution but the emulator was installed and configured on top of each RPi2 and on top of the grid software there would also be installed a virtualization system.

To summarize the possible implementation was closer to the solution on figure 30 and is illustrated in figure 31. On each RPi2 there was installed with the Raspbian OS. The RPi2 has an ARM processor, so both KVM and QEMU were necessary in order to run an x86 application. QEMU provides de virtualization of the RPi2's physical machine.

They are all virtual machines and it is desired that

they can run x86 application, hence the KVM emulator; it emulates an ARM architecture to an x86 architecture. On top of each virtual Beowulf system, the grid software, was installed and configured to make it all work as one.

The entire grid is controlled by teseus0, also a virtual machine. Teseus0 was virtualized using the same procedure as teseus1, 2, 3, 4 and 5. The basic physical machine of teseus0 has a network monitor

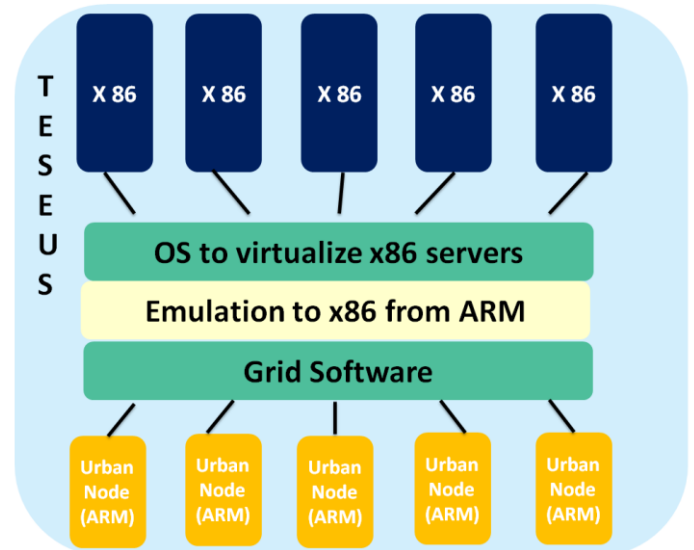


Figure 29 - Emulation software on top of grid

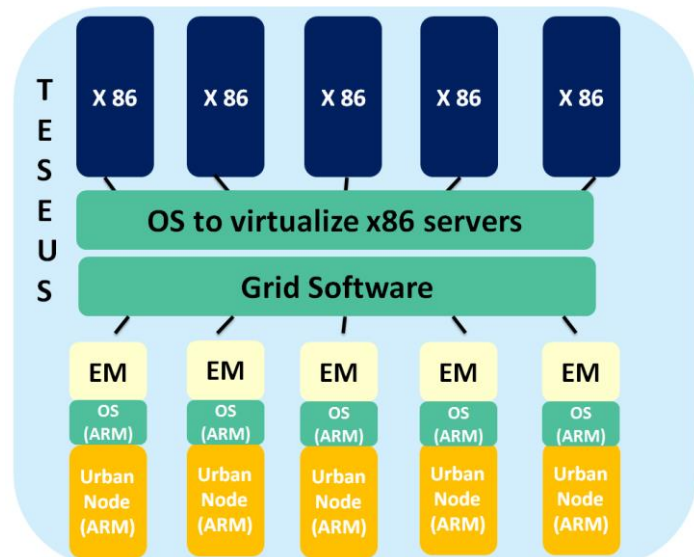


Figure 30 - Emulation software on top of ARM hardware

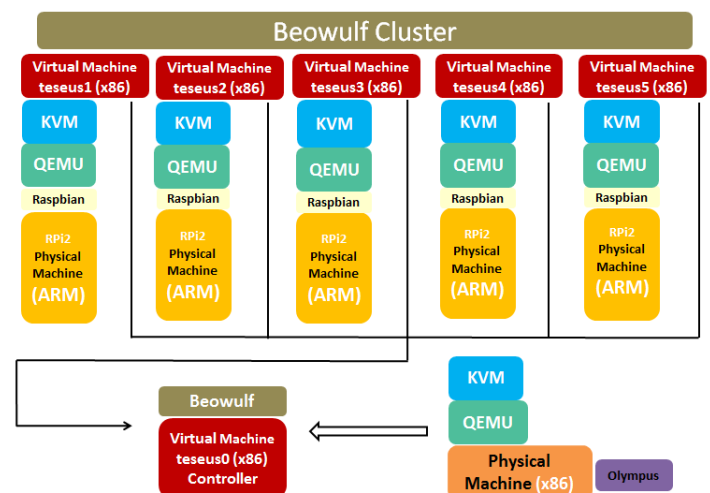
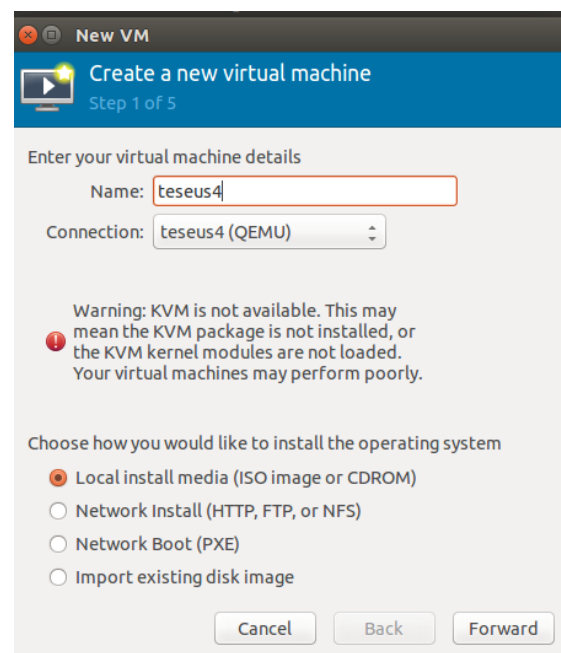
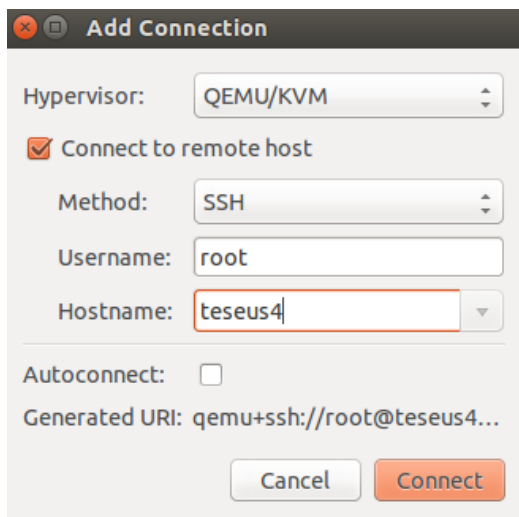


Figure 31- Beowulf cluster of RPi2s

named Olympus for security reasons.

5.6 Problematic

Several attempts and different solutions have been tested but in step 3 it is not yet possible to install a virtual machine on the RPi2s in order to run x86 application on the ARM hardware. The Raspberry Pi has an interruption manager incorporated in the software, but in the RPi2 this is a part of the hardware. The problem can be solved, but it requires the time to rebuild the kernel to isolate one or three cores in order to run QEMU, KVM is installed but not supported. Another solution is waiting for new kernel update that enables virtualization extensions out of the box⁶⁵. In the fourth step it is possible to install the necessary libraries and programs to handle the Beowulf cluster but when installing the same features on the RPi2s we experience crashes once again.



Prototype

The rack is the physical place where the cluster of RPi2 will be placed. The company currently owns five RPi2 but the rack can place up to eight of them and its structure is resistant enough to uphold several more racks on top of it. For the Teseus project this rack is meant to be placed in a locker that is the reason to be for the holes. All measures were decided taking into account the RPi2 and locker dimensions. In the locker there are also the screen and the controller of the grid system.

Design & functionality

The rack is made of polymethyl methacrylate (PMMA) and can contain eight computers. The computer card is screwed into a transparent PMMA board whose measures are 9,5cm high and 12cm long. This board act as a plate that can easily be placed but it has the particularity that it is slightly more complicated to be

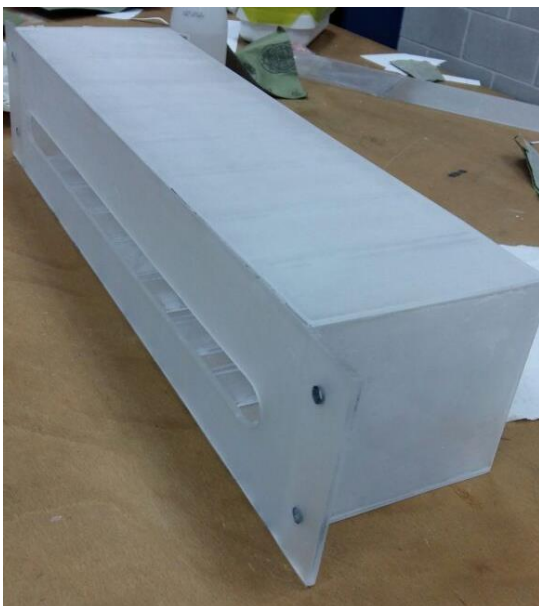
⁶⁵ <http://blog.flexvdi.com/2015/03/17/enabling-kvm-virtualization-on-the-raspberry-pi-2/>

removed as the guides of the base aren't exactly alienated. This is for security and reasons and convenience, this way the rack can be placed in all angles.

The front side is a plate 43cm long and 10,5cm high with four holes to mount the rack to the larger rack that is in a locker, it also has an open space meant for the Ethernet that connects the RPi2s with the used switch/router. The backside of the rack is open in order to make the rack scalable by inserting or removing plates as needed, and to provide a certain amount of air circulation, should the computers heat up a little. The open back provides space for the power source cable that is connected from the south side when attached. By having a vertical placement of the boards it is possible to place more boards than if it had a horizontal placement.

The guides can be used in two different ways. The transparent plates where the RPi2s are screwed are placed in the base guides and one can decide whether aliening the upper side of the plate with the upper guides to obtain more friction when removing the computer boards or to place it next to them to obtain less friction and therefore easier removal.

The choice of material is based on price, density, transparency, cooperation and on the possibility to recycle. The whole structure was polished in order to hide printing marks and obvious dust. The plates where the computers are screwed are transparent since they are not exposed to dust in the same way. The pieces have been assembled with super glue, and by melting PMMA with acetone especially in the front side where the joints are to make the structure more durable.



Discussion

The initial configuration of Teseus has been made. However, when trying to install an x86 program on the ARM boards many obstacles appeared, such as: The Raspberry Pi 2 board is too new when considering installing the necessary OS. Most programs only run on x86 machines, but in the future a great deal more will be made for ARM programs and that is good news for Teseus.

The concept of Teseus might be ahead of its time, but not by much, so with a few adjustments in the hardware or the software compatible with RPi2, Teseus will be a powerful grid. Due to the fact that most of the network infrastructure in Vilanova is fibre optics, makes the city a good place to start using the RPi2s as the microcomputer in the Urban Nodes. It is highly likely to build a low cost, scalable data server with existing technology, but the configuration of creating a grid system and pooling the computer resources as well as having task distributed to several nodes at the same time, takes more time to develop than first anticipated.

Further work

The future of Teseus will involve more research of how to install a DOS such as Inferno or Plan 9, but since current DOS' are still experimental, the Globus toolkit for creating a grid of nodes will be tested first. In order to reach the goal, a great deal of research and programming is yet to be done. Currently, Teseus consist of 5 RPi2s, so in order to ensure that large tasks can be divided and processes on multiple nodes with success, more single-board computers must be added. Teseus is created, but not limited, to serve Smart Cities and can be a success if it is further developed, which it is expected to happen in the following semesters.

Conclusion

Teseus is an innovative project that will serve and improve the way energy is used today. In Smart Cities there is a need for smart energy consumption, and the Teseus project is a step in the right direction. This semester the team had the objective of finding the possibility of building a low cost, scalable computer with existing technology. First, it was necessary to know what the future need for energy will be, and smart cities use technology for almost everything. Teseus would first of all be installed in the Urban Nodes, and process the data obtained by it and therefore Teseus have to handle this.

The second step, was to find the best suited single-board computer to function as the brain of the Urban node, and it was found that the Raspberry Pi 2 due to its high ranking in computational power, energy consumption and price was the best solution amongst the many available. The SBCs have much lower power consumption than the standard x86 computers, and by using a compilation of smaller, but several cheaper boards, makes it possible to control the energy consumed.

In order to determine the best possible software server solution to connect the Urban Nodes, three different data distribution systems were considered. The grid system was chosen as the opportunity for the Teseus team to obtain the best result with the current available software solutions. Therefore, the team decided to implement a Beowulf cluster that allows sharing the processing of tasks among computer nodes. The team was successful in the configuration of the switch and RPi2 boards by creating a virtual control node with an x86 computer and an Ubuntu operating system. In addition, the team established a network of the five nodes, teseus1-5, accessible from the control node Olympus with Beowulf, the super user. However, despite many attempts the installation and configuration of installing the libraries and programs that allows the parallel computing in a Beowulf cluster, was not possible. The team came to conclusion that the possibility of creating a low cost, modular and scalable data server, with existing technology is more likely in the near future, certainly when the project is further developed on.

Acknowledgements

Apart from us, the EPS students, more people have participated in the creation of this project and involve both professors and other students. We students are very thankful to the supervisors and professors that have been inspiring and we have learned so much, and we are forever grateful. They taught us how to organize the project with the scum methodology and gave us guidelines especially at the beginning. We received classes and guidance in programming, and we would not be here without them. We have had several meetings and they were always willing to give constructive criticism and feedback to guide us in the right direction.

In addition we are grateful of Neàpolis for providing an interesting project, lending us their facilities and for Josep Farré, who always were willing to help us with any difficult obstacles that we might have faced during the course of this project. Due to our inexperience in the informatics major, without him probably it would have been impossible to develop the project the way we did it.

We are the most thankful for the help and encouragement from the computer science students, Joan Josep and Lorena that were always there for any inconvenient or doubt that came up. We truly appreciate their patience, professionalism and kindness, especially during the last sprint of the project.

Our acknowledgment goes to the EPS organization for providing the opportunity to work on real projects with a company and bringing people together from around the world. We would also like to thank Santi from the EPSEVG workshop, who provided us most of the material and tools necessary to assemble the rack holder.

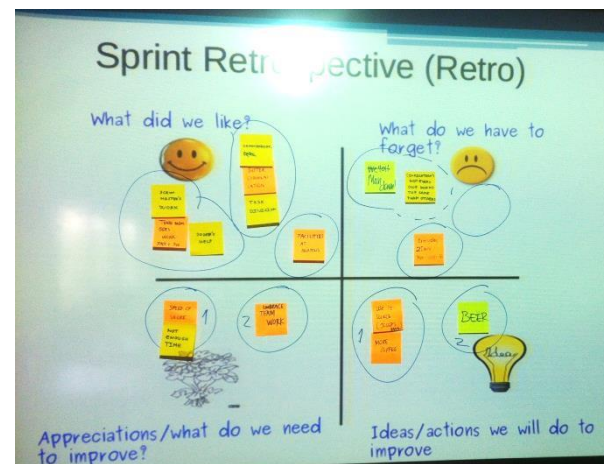
Project Management

Scrum

Throughout the project we have used the Scrum methodology. This means that the entire project has been broken down into smaller stories in a product backlog by our company supervisor. The Epics of the product backlog has user stories where users (us, students) describe aims and goals, for example “as students we want to...” and after the product backlog is then broken down into tasks. These user stories are made in a sprint meeting by all the people involved in the project and here the team will estimate the time needed for each task.

Sprint meetings

Typically, a sprint will last 2-4 weeks, and in our case 1.5-2.5 weeks and in the end of the each sprint a sprint review is held, each containing a *demo* and a *retro*. The *demo* displays the work done, and needs to be presented as to sell the idea to the buyer or to convince them to continue the project. In the *retro* the team members will each present their thoughts on a post-it of; what worked, what we need to forget, what we wish to improve and what actions are needed to solve our problems.



Daily scrum meetings

During the sprints there will be *daily scrum meetings* of 15 minutes where each member of the team explains;

- *What have you done until today?*
- *What will you do today?*
- *Have they had any problems so far?*

It is agreed by all members of the team how frequent the daily meetings will be held, because sometimes a weekly meeting also suffice.

Trello

We have used Trello as a project management platform that visualize our Scrum work and it has worked as a great tool to give an overview of the tasks that needs to be completed. In case a team member has not been able to participate in a meeting, it is possible for them to see what tasks they have been assigned and

is responsible for on Trello. It also displays how far you are in the process and you can upload anything related to the specific task, such as documents and as a communicative tool for each member and create a checklist to follow.

Scrum Master

During each sprint there needs to be a *Scrum Master* that divides tasks and ensures that each person responsible for a task reaches the deadline, and that the work delivered is good enough. If a problem occurs for any team member on their daily scrum updates, it is the job of the scrum master to help find a solution to solve the problem. It is also the task of the Scrum Master to manage Trello, divide task to others → such as, who needs to find a room for the team meetings, take notes during meetings, provide supplies for meetings, give updates to missing members and pass on information to supervisors or other involved parties. Furthermore, the Scrum Master leads the meetings by initiating and making sure everyone has a chance to express their opinion, as well as making sure everyone sticks to the agenda of the meeting. Normally, these are only tasks of the Scrum Master's and he or she is not assigned other tasks, but in our case with only 2-3 members working full time on the project it has been necessary for the Scrum Master to work on individual tasks.

Sprints

In each sprint we set different goals in the group for what each individual as a team should achieve in the time span we had. It was quite a task to determine how much time was necessary for the given tasks, since we were a team of three EPS students, two part time students and it was a challenge to distribute task with such an abstract project. This was also due to the fact that none of the fulltime students has their subject in computer science.

Sprint 0 – Research and document research of ARM families that allows hardware virtualization, ARM electronic boards, computational power, and the scrum master has the project manager role.

Sprint 1 – Making project agreement, finish research and documentation of computational power, energy efficiency and consumption. This sprint also included research of switch vs. router, differences between cloud, grid and DOS, and software available. Go to Neàpolis and find place for the server rack and receive components from supervisor.

Sprint 2 – Finish documentation of software options available, as well as switch/router connection possibilities. A week went by preparing for the midterm report and presentation and this included coordinating the project.

Sprint 3 – Making eco design reports for other classes, decide software solution, discuss the feedback from midterm, configuring chosen switch and installing OS on RPi2s.

Sprint 4 – Creating of rack holder, study installation and configuration of switch and RPi2 boards, as well as starting installation.

Sprint 5 – The fifth sprint was initially not planned, but as a team we decided to continue using scrum and Trello as project management tool, and because we needed more time due to programming complications. In the final sprint we created the grid controller, users for our system and virtual machines on our RPi2. We finished writing the report and with thanks to our methodology we reached our deadlines.

Pro's & Con's

Pro's

It has been a great tool for everyone and Trello is a great platform to control the project from. The sprint meetings have been great because we have all the dates for each sprint, so we know exactly how long we have for each task. The sprints are also frequent enough to keep the team on track. The sprint reviews are a great opportunity for the members of the team to express their opinion and give feedback of the methodology and of other team members. The structure of Scrum ensures that deadlines are reached if the time has been estimated correctly.

Con's and solutions

There are not many disadvantages by Scrum itself, but it takes some time to get introduced properly and to solve this, we have at each sprint reviewed the rules of Scrum. For us doing is also educating, so after doing retros, demos and meetings a few time we got the hang of it. It has been a challenge to get everyone to check Trello and making sure they are up to date, but by using WhatsApp as a communicative tool it has been easier to get responses from team members and reminding them to check Trello.

One thing is to know your plan of approach, another is to understand and collaborate with the people in your group, especially when working in and international team, where cultural differences can have an impact on your teamwork and end result. Therefore, each team member has learned from each other and got an understanding of the culture they come from, which is a great tool when acting in project management.

Bibliography

Books

Bart, Jacob, Michael Brown, Kentaro Fukui and Nihar Trivedi, *"Introduction to Grid Computing"*. Redbooks. [Www.ibm.com/redbooks](http://www.ibm.com/redbooks). December 2005. (International Technical Support Organization)

Websites

Front page: <https://www.pinterest.com/pin/55450639138716233/> accessed 17.04.15

A

<http://www.androidauthority.com/arm-vs-x86-key-differences-explained-568718/> accessed 21.04.15

<http://www.arm.com> accessed 15.04.15

B

<http://www.bananapi.org/p/product.html> accessed 17.04.15

<http://beagleboard.org/BLACK> accessed 15.04.15

<http://blog.flexvdi.com/2015/03/17/enabling-kvm-virtualization-on-the-raspberry-pi-2/>

C

<http://computerpricetrends.com/14083-intel-xeon-x5570-nehalem-2-93ghz-4-x-256kb-l2-cache-8mb-l3-cache-lga> accessed 22.04.15 <http://cordis.europa.eu/docs/projects/cnect/6/270896/080/deliverables/001-D11StateoftheArtOpenInnovation.pdf> accessed 16.04.15

<http://cubieboard.org/model/cb4/> accessed 15.04.15

D

<http://da.wikipedia.org/wiki/Petabyte> accessed 20.04.15

<https://www.digitalocean.com/community/tutorials/how-to-create-a-beowulf-cluster-using-ubuntu-12-04-vps-instances> accessed 31.05.15

<http://www.donellameadows.org/archives/population-growth-the-last-and-the-next-35-years/> accessed 16.05.15

E

<http://www.epsevg.upc.edu/eps-idps-projects/download/5/Article.pdf> accessed 16.05.15

H

http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141578608433&tab_idx=2 accessed 20.04.15

www.hindawi.com accessed 16.04.15

<http://home.web.cern.ch/about/computing> accessed 31.05.15

<http://iqjar.com/jar/an-overview-and-comparison-of-todays-single-board-micro-computers/> accessed 22.04.15

L

<http://www.lemaker.org/> accessed 15.04.15

M

http://www.mellanox.com/pdf/whitepapers/Building_an_Enterprise_Supercomputer_final.pdf accessed 17.04.15

http://www.mosix.cs.huji.ac.il/txt_about.html 29.03.15

<http://www.maxi-pedia.com/grid+computing+distributed+computing> 07.04.15

http://www.mosix.cs.huji.ac.il/faq/output/faq_q006.html 29.03.15

N

<http://www.neapolis.cat/> accessed 16.04.15

<http://www.neapolis.cat/index.php?id=43> accessed 16.04.15

<http://www.nas.nasa.gov/hecc/resources/pleiades.html> accessed 16.04.15

P

<http://www.popline.org/node/340708> accessed 16.05.15

Q

<http://www.qnx.com/> accessed 31.5.15

<http://www.qnx.com/products/neutrino-rtos/neutrino-rtos.html#technology-security> accessed 31.05.15

R

<https://www.raspberrypi.org/products/raspberry-pi-2-model-b> accessed 20.04.15

<http://research.cs.wisc.edu/htcondor/overview/> accessed 29.03.15

http://research.cs.wisc.edu/htcondor/tutorials/videos/2014/Intro_To_Using_HTCondor.html acc. 29.03.15

http://www.researchgate.net/publication/228857478_Cluster_security_as_a_unique_problem_with_emergent_properties_Issues_and_techniques accessed 31.5.15

S

<http://searchenterpriselinux.techtarget.com/tip/Linux-clusters-vs-grids> accessed 07.04.15

<http://stackoverflow.com/questions/9723040/what-is-the-difference-between-cloud-grid-and-cluster> 18.03.15

<http://stackoverflow.com/questions/14794460/how-does-the-arm-architecture-differ-from-x86> - 21.04.15

<http://www.slideshare.net/anantn/unix-plan9-from-bell-labs> accessed 07.04.15

<http://www.smallbusinesscomputing.com/testdrive/article.php/3819231/What-is-Virtualization-and-Why-Should-You-Care.hTM> accessed 17.04.15

http://www.softlayer.com/info/high-performance-computing?utm_source=google&utm_medium=cpc&utm_content=Supercomputer_-_Supercomputer&utm_campaign=PPC-GLOBAL-HPC&utm_term=%2BSupercomputer&matchtype=b

accessed 15.4.15

T

<http://www.thefoa.org/tech/ref/OSP/nets.html> accessed 31.05.15

<http://toolkit.globus.org/toolkit/about.html> accessed 29.05.15

http://www.tutorialspoint.com/operating_system/os_security.htm accessed 29.05.15

U

<http://www.umiacs.umd.edu/labs/parallel/condorintro.htm> 29.03.15

<http://www.univergesv8100.com/nec-sv8100-applications/nec-sv8100-series-inrouter/> accessed 20.4.15

V

<http://www.vitanuova.com/inferno/> accessed 07.04.15

<http://www.vmware.com/virtualization/virtualization-basics/what-is-virtualization> accessed 17.04.15

W

http://en.wikipedia.org/wiki/ARM_Cortex-A7 accessed 20.03.15

http://en.wikipedia.org/wiki/Beowulf_cluster accessed 31.05.15

http://en.wikipedia.org/wiki/Internet_security accessed 31.5.15

http://en.wikipedia.org/wiki/Grid_Security_Infrastructure accessed 31.05.15

<http://en.wikipedia.org/wiki/Linux> accessed 31.05.15

http://en.wikipedia.org/wiki/Load_balancing_%28computing%29 accessed 07.04.15

http://en.wikipedia.org/wiki/Secure_Shell accessed 31.05.15

http://en.wikipedia.org/wiki/Smart_city accessed 20.04.15

Y

<https://www.youtube.com/watch?v=QJncFirhjPg> accessed 29.03.15

<https://www.youtube.com/watch?v=59MiSspGYul> accessed 29.03.15

Appendix A - Rating

Value criteria	Weight	Cubieboard 4	score	Beaglebone black	score
Price	5	\$100	1	\$35	5
SOC,system on chip	4	Allwinner UltraOcta A80	3	AM335x 1GHz ARM	4
Processor	9	Octa-Core A15/A7 big.	7	ARM® Cortex-A8	7
GPU	3		2	AM335x 1GHz	2
RAM	7	2GB DDR3 RAM	7	512MB DDR3	5
Connectivity	3	Ethernet port	3	Ethernet port	3
USB	2	1 x USB HOST		4 x USB HOST	2
GPIO ports	3	10 pins	3	40 pins	3
Power consumption	8	DC5V@2.5A	4	210-460mA @ +5V	4
Virtualization	10	yes	10	No(Cortex-A8)	0
Weighted sum	54		40		35

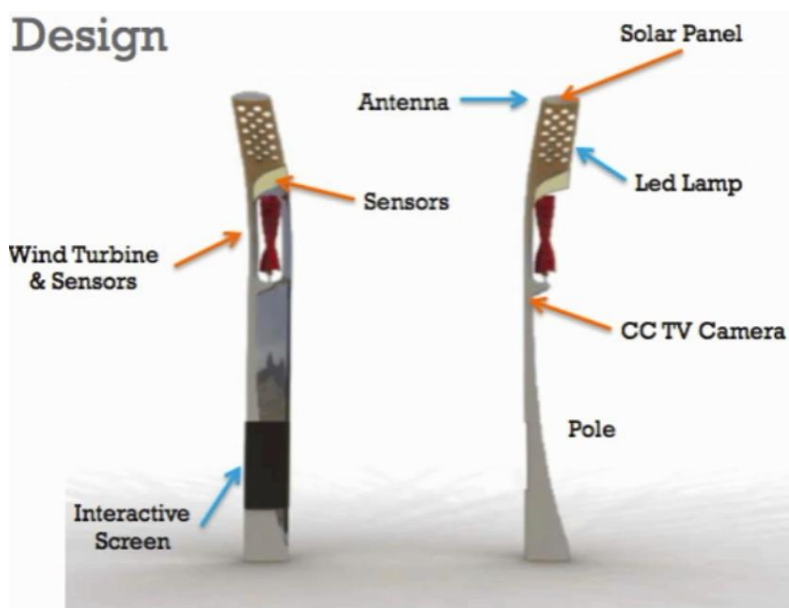
Value criteria	Weight	Banana Pi	score	ODROID-C1	score
Price	5		\$35		
SOC,system on chip	4	Allwinner A20*	3	Amlogic S805	4
Processor	9	ARM Cortex-A7 dual-core, 1GHz	7	Quad Core Cortex™-A5 process	7
GPU	3	Mali400MP2 GPU	2	Dual Core Mali™-450 GPU	3
RAM	7	1GB DDR3 DRAM	6	1GByte DDR3 32bit RAM (512M	7
Connectivity	3	Gigabit Ethernet	2	Gigabit Ethernet	3
USB	2	2* USB 2.0 ports, USB OTG x 1 , 1* micro USB for power supply	2	2.0 Host x 4, USB OTG x 1	2
GPIO ports	3	26-pin	2	40 GPIO pins	3
Power consumption	8	5V,200mA~400mA	6	5V, under 500mA	6
Virtualization	10	yes	10	No	0
Weighted sum	54		75		35

Score of Raspberry Pi 2 = $5 \times 0.9 + 4 \times 0.8 + 9 \times 0.8 + 3 \times 0.6 + 7 \times 0.7 + 3 \times 0.5 + 2 \times 1 + 3 \times 0.9 + 8 \times 0.8 + 10 \times 1 = 44.5 + 3.2 + 7.2 + 1.8 + 4.9 + 1.5 + 2 + 2.7 + 6.4 + 10 = 44.2$ points

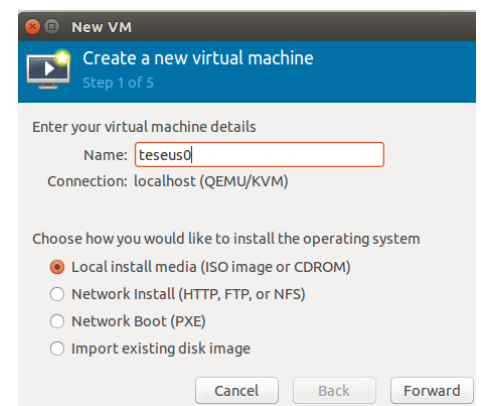
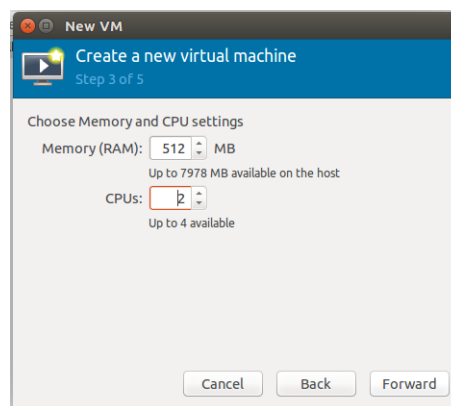
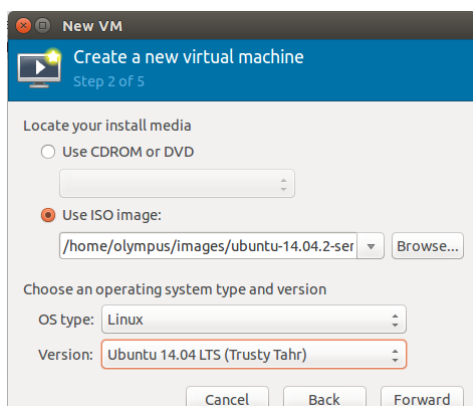
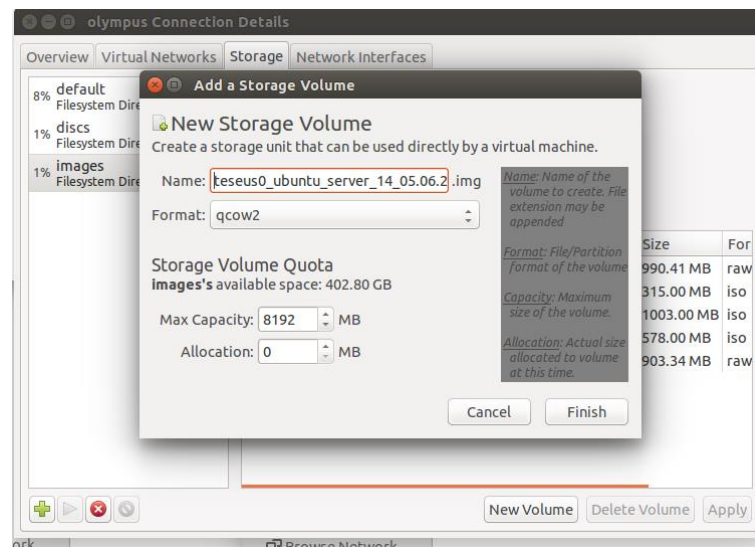
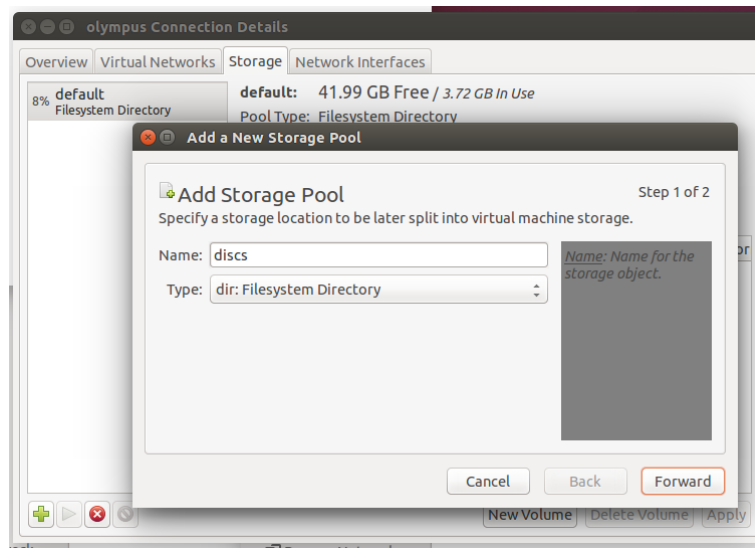
Score of Banana Pro = $5 \times 0.5 + 4 \times 0.7 + 9 \times 0.8 + 3 \times 0.6 + 7 \times 0.6 + 3 \times 0.5 + 2 \times 0.5 + 3 \times 0.9 + 8 \times 0.4 + 10 \times 1 = 2.5 + 2.8 + 7.2 + 1.8 + 4.2 + 1.5 + 1 + 2.7 + 3.2 + 10 = 34.4$ points

Appendix B – Urban Node

The pictures below are taken from the Urban Node video of the project in the spring of 2014 and demonstrate the components of the Urban Node and a typical scenario of placement in Vilanova I la Geltrú.



Appendix C – Creating VMs



Appendix D – SSH keys

```

beowulf@teseus1: ~
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 19 16:32:05 2015 from olympus.neapolis.cat
pi@teseus1 ~$ sudo adduser beowulf --uid 900
S'està afegint l'usuari «beowulf»...
S'està afegint el grup nou beowulf (900)...
S'està afegint el nou usuari beowulf (900) amb grup beowulf...
S'està creant el directori personal «/home/beowulf»...
S'estan copiant els fitxers des de «/etc/skel»...
Introduïu la nova contrasenya d'UNIX:
Torneu a escriure la nova contrasenya d'UNIX:
passwd: s'ha actualitzat la contrasenya satisfactòriament
S'està canviant la informació d'usuari per a beowulf
Introduïu el nou valor, o premeu INTRO per al predeterminat
Nom complet []: beowulf
Número d'espai []:
Telèfon de la feina []:
Telèfon de casa []:
Altres []:
És aquesta informació correcta? [S/n] s
pi@teseus1 ~$ su - cluster
No hi ha entrada de contrasenya per a l'usuari «cluster».
pi@teseus1 ~$ su - beowulf
Contrasenya:
Generating public/private rsa key pair.
Enter file in which to save the key (/home/beowulf/.ssh/id_rsa):
Created directory '/home/beowulf/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/beowulf/.ssh/id_rsa.
Your public key has been saved in /home/beowulf/.ssh/id_rsa.pub.
The key fingerprint is:
be:39:d6:a5:15:82:32:cb:ce:02:7e:21:37:c7:cf:65 beowulf@teseus1
The key's randormart image is:
+---[ RSA 2048 ]-----
      o . . .
      o +S . .
      o + = . E o
      . + * o . +
      . o o =oo
      . . .o.
      +-----+
First, change users to the new cluster user. Supply the password you set during creation:

su - cluster

Now, we can generate RSA keys with the following command:

ssh-keygen

```

```

pi@teseus1: ~
ssh-copy-id localhost
pi@teseus1 ~$ exit
logout
pi@teseus1 ~$ apt-get install mpich2
E: No es pot resoldre el fitxer de blocat /var/lib/dpkg/lock - open (13: S'ha denegat el permís)
E: No es pot blocar el directori d'administració (/var/lib/dpkg/), sou root?
pi@teseus1 ~$ sudo apt-get install mpich2
S'està llegint la llista de paquets... Fet 0%
S'està construint l'arbre de dependències

```


Appendix E – commands

Directories: /..

- **bin** (stores the binary codes)
- **etc** has the configuration of the programs, also the ones that are not in bin (you can configure all from here,
- **root** is the system user, mail and other programs can create their own users. The users created by root are stored here.
- **sshd** (user created by a program to stop hackers, not human created)
- **media** (stores the interfaces of your CD-ROMs or SD cards etc.)
- **proc** (the programs running – don't go here)
- **sys** (where the kernel system is stored, RAM, firmware(drivers))
- **boot** stores the grub (the one that asks you what OS you wish to use)

Commands:

ls - list the files in the directory

ls -l - gives more information of the files, such as the date and time of last update etc.

cat - (catalog) gets the file and translate it from bit to ASCII code

nano - is a small editor that allows you to edit the file (write the filename after)

echo - (echoes what you write, so stupid but useful → example echo ghgghskgh > asd) inserts into new file

^C - quit

touch – updates the date (or can create a file→ example: touch mom will create a file called mom, and this also ensures that the file does not already exist with that name)

stdout - will show you the screen (standard output) →

rm – remove

cp – copy

exit – exits

Appendix F - Business model canvas

Problem description

Today most computer users only use a part of their personal computers capacity when doing their daily tasks and the energy consumption is high for the amount of work actually being carried out. One way to take advantage of the full computer capacity is to add the computer to a grid network. Another way is to make future computers more scalable and cheaper by changing the architecture of the CPU.

The aim of the Teseus project is to lower the power consumption of computer systems. Instead of having one powerful machine it is possible to compile small machines and make them perform as one, which will increase both increase the performance of the computer and lower the energy bill. Teseus will positively affect companies, the individual user and create a foundation for sustainable energy solutions around the world.

Ideal final result

The ideal final result is to create a supercomputer or a distributed grid network consisting of single board computers⁶⁶ compiled and programmed to function as one large computer with low power consumption. The system is not limited to, but considered to be implemented in smart cities where there are placed urban nodes around the city to collect smart data related to sectors such as health, transport and sustainable energy. In each urban nodes there will be placed a single board computer to treat the data from the sensor in the nodes. Furthermore, the single board computers will be connected to other urban nodes via fibre optics that will secure fast internet connection. The network will be able to manage it-self and the processes running in order to secure high speed transmission and a distribution of applications in order to avoid overload on the system and maintain fast internet.

The supercomputer should be able to manage and run programs and applications available for all operating systems and manage the data transmission necessary. However, if this is not possible the Teseus will function as any other system already in development by other companies.

Area of innovation and competitive advantage

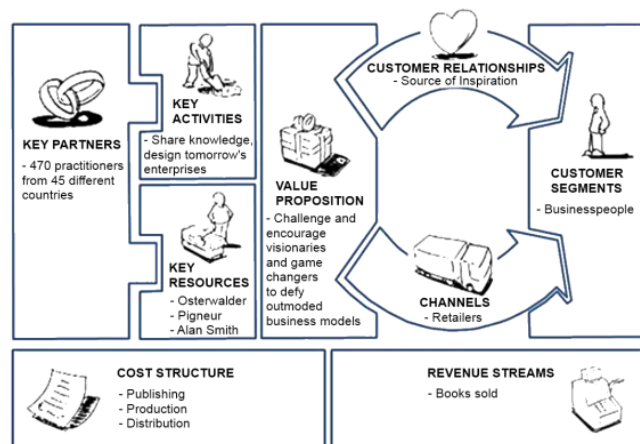
The way Teseus differentiate itself from other types of supercomputers is partly the hardware components, as well as the setup of the software. The CPU, RAM and memory are some of the architecture of the hardware that differs from the standard of computer systems. The development of single board computers of ARM architecture⁶⁷ makes it possible to lower the power consumption of your computer hardware. When considering the development in the human population and the predicted need for energy resources as well as the increase in use of information required in smart cities, a need for a more sustainable and smarter computer system has arisen. Teseus will run with a distributed operating system, which gives the opportunity to exploit the full capacity of all the computers (nodes) in the grid. In a distributed operating system it is possible to divide an application in process between several computers, which is not possible with for example simple cloud computing. Teseus combines new hardware architecture with developing operating system to create a more efficient, less power consuming and more transparent computer network than seen before.

⁶⁶ <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>

⁶⁷ http://en.wikipedia.org/wiki/ARM_architecture

BMC

The Business Model Canvas⁶⁸ is used to create a proposed business plan for Teseus and will display the first draft of the best customers segments, what added value we provide in comparison with other recent computing systems and what key partners and resources we need in order to achieve an inexpensive, efficient product.



1 Customer Segments

Teseus is scalable and can suit any type of private or public company that wish to supply cities with a highly interconnected network. The typical user company, regardless of size, is related to urban node data in sectors such as pollution, sustainable energy, recycling and transport.

Another stakeholder is the individual user that will install the grid software on their personal computer to add it as an urban node to the grid.

Possible customers

1. Public Administrations. All the cities who want to know what happens in their area.
2. Telecommunication infrastructure companies. In the next future all the companies who goes to 5th generation mobile networks needs something similar to expand their infrastructures. The communications business is changing now. We will not pay for "data" and "voice". We will only pay for data and then we will choose how we wish to speak with somebody. Probably, a new kind of company will grow: the computing power enterprise.
3. Computing power companies. Today it does not exist, but possibly in few years they will appear. These types of companies will sell computing power to its subscribers. You will have a little computer in your house, but connected to a "city computer grid". If you need more power, for example when you are doing a cad, you can get this power from the network, by paying for it.
4. Data selling. The data we will capture will be very important. The infrastructure owner can sell the data (temperature, parking occupation, gardens humidity, power consumption, and ...) to companies who wants it. This data must be processed by Teseus.

A **SWOT** is applied here to provide an overview of the current status of Teseus and used as tool to see where improvements are needed.

Strengths: the Teseus idea has substance and is innovative for its time and focuses on efficiency which is an aspect that anybody, either company or ordinary user, is interested. It

⁶⁸ <https://nbsbookclub.wordpress.com/2013/02/14/business-model-generation-a-handbook-for-visionaries-game-changers-and-challengers/>

is also inexpensive to develop due to the price of the SBCs in comparison to the standard solution.

Weaknesses: the configuration of compiling and configuration of hardware components has not yet been tested.

Opportunities: the system precisely as Teseus does not exist.

Threats: current supercomputers are optimized all around the world by companies and organization with favourable means, the distributed operating system is not fully released.

2 Value proposition

Teseus can provide value to all customer segments by providing the service of lower energy consumption, high speed, reliable and secure internet connection servers. The product will have a favourable price due to low maintenance, inexpensive setup and product assembling.

Research has shown the hardware properties it-self can process task much faster with a lower consumption of energy than current computer system solutions. The system is suited for all kinds of users that are interested in low efficiency computers, ranging from common domestic users and small or medium companies to high computational centres such as the supercomputer MareNostrum in Barcelona.

3 Key partners, resources & activities

The most important suppliers of our resources and therefore a key partner are the provider of the single board computers (Raspberry Pi 2 (RPi2), the additional hard discs on memory cards and the switch provider. Other key resources is people and this means the software programmers that will be handling the maintenance of the Teseus as well as the sales people that will make sure the product is known by relevant customers. The major part of the customers at Neàpolis is the same size, and therefore no client is considered more important than others.

The key activities lie in the management of the RPi2 clusters. The grid controller needs a manual supervisor. When Teseus has a distributed operating system installed, the system will need little if any maintenance. This means the key activity will be ordering, assembling and installing RPi2's to the Teseus cluster at Neapolis.

4 Channels & Revenue streams

Teseus is a server cluster rack that will be placed at Neàpolis in Vilanova I la Geltrú at first and thereafter distributed further online where it also will be a distributed service to the customers of Neàpolis the same way.

The new server system of RPi2 boards will reduce the current energy consumption of Neàpolis. Instead of each customer of Neàpolis' already established network, will pay the same or maybe even less. Therefore,

the revenue stream will come from the customers who wish to buy the cloud server service that Neàpolis provides.

5 Cost structure

The cost per each Raspberry Pi 2 board is 35USD. Other expenses lies in sustaining the system, which means that when you need more server capacity you need to add more RPi2 boards as well as the chargers (included in the price), a switch for additional 10 boards and Ethernet cables to connect each board.

Appendix G – Eco design

Ecoaudit tool – Teseus

The Ecoaudit tool will be applied on Teseus as tool to investigate the life cycle of the product and it is also a helpful tool of researching for an even more sustainable solution. The Teseus project focus lies on the configuration of multiple small computers components combined to create one large single computer.

The following images are an example of the prototype of Teseus to display the components involved in a small scale. The table shows the individual break down of materials included in the prototype. The materials included are assumption to give an idea of use. Teseus in full scale is extremely complex and also for future research and not taken into consideration.

Materials

Component	Material	Recycled content* (%)	Part mass (kg)	Qty.	Total mass (kg)	CO2 footprint (kg)	%
Raspberry Pi 2	Printed circuit board assembly	Virgin (0%)	0,03	5	0,15	1,5	1,5
Cable	Cable	Virgin (0%)	0,01	7	0,07	0,48	0,5
Controller	Desktop computer (without screen)	Virgin (0%)	3,5	1	3,5	84	86,2
Keyboard	Keyboard	Virgin (0%)	0,3	1	0,3	6,6	6,8
Mouse	Optical mouse, with cable	Virgin (0%)	0,05	1	0,05	2,1	2,2
Rack material	Polymethyl methacrylate (Acrylic, PMMA)	Virgin (0%)	0,4	1	0,4	2,7	2,8
Memory	Hard disk drive	Virgin (0%)	0,005	1	0,005	0,13	0,1
Total				17	4,5	97	100

Table 4 – detailed breakdown of individual life phases

- single board computers (printed circuit boards)⁶⁹ – Raspberry Pi 2

⁶⁹ <http://www2.epa.gov/sites/production/files/2014-05/documents/handout-10-circuitboards.pdf> accessed 15.5.2015

- cables/wires - Ethernet⁷⁰
- controller – desktop computer
- keyboard for the controller machine
- mouse with cable
- memory – micro USB cards
- rack holding structure model

Manufacture

The manufacturing of all the necessary components are not possible to locate in the CES Edupack program, so no exact numbers for Teseus is specified. However, the difference between our solution and a normal supercomputer⁷¹ is the manufacturing of the CPU, so that is taken into consideration.

For the Raspberry Pi 2, the price is 35USD⁷² in comparison to the x86⁷³ microprocessor installed in one example of a supercomputer, which has a price around 140USD⁷⁴. The instruction sets of the computers are complex for x86 which makes the manufacturing process longer and more energy consuming and the RPi2⁷⁵ has a reduced which makes the process shorter.

Transport

It is assumed that all the hardware components are produced in Taiwan and transported by containerships to Barcelona, with a distance of 6000km.

⁷⁰ http://en.wikipedia.org/wiki/Category_5_cable accessed 15.5.2015

⁷¹ <http://simple.wikipedia.org/wiki/Supercomputer>, <http://www.anandtech.com/show/6421/inside-the-titan-supercomputer-299k-amd-x86-cores-and-186k-nvidia-gpu-cores> accessed 15.5.2015

⁷² <https://www.raspberrypi.org/raspberry-pi-2-on-sale/> accessed 15.5.2015

⁷³ http://en.wikipedia.org/wiki/Opteron#Opteron_.2832_nm_SOI.29 accessed 15.5.2015

⁷⁴ <http://www.amd.com/en-us/products/pricing/opteron> accessed 15.5.2015

⁷⁵ <https://www.youtube.com/watch?v=Tza6Hl8wSJ0> accessed 15.5.2015

Breakdown by transport stage

Stage name	Transport type	Distance (km)	Energy (MJ)	%
Total transport of material	Sea freight	6e+03	4,3	100,0
Total		6e+03	4,3	100

Breakdown by components

Component	Mass (kg)	Energy (MJ)	%
Raspberry Pi 2	0,15	0,14	3,4
Cable	0,07	0,067	1,6
Controller	3,5	3,4	78,2
Keyboard	0,3	0,29	6,7
Mouse	0,05	0,048	1,1
Rack material	0,4	0,38	8,9
Memory	0,005	0,0048	0,1
Total	4,5	4,3	100

Table 5 – Breakdown by transport stage and by components

Depending on how large a computer is needed, more memory, RPi2's and cables are needed, since only one controller for the whole system is needed. The additional energy (MJ) if you want a supercomputer of 100 cards each with a 16GB memory, you will have to add $0.14 \cdot 20 = 2.8$ (MJ) and so on.

Static mode**Use**

Energy input and output type	Electric to thermal
Use location	Europe
Power rating (W)	15
Usage (hours per day)	24
Usage (days per year)	3,7e+02
Product life (years)	2

Table 6 – Use of Teseus – static mode

The two bar charts display a summary of the energy and CO2 footprint for Teseus over a two year period. The system will be running 24 hours a day 365 days a year, and have a guaranty of 2 two years.

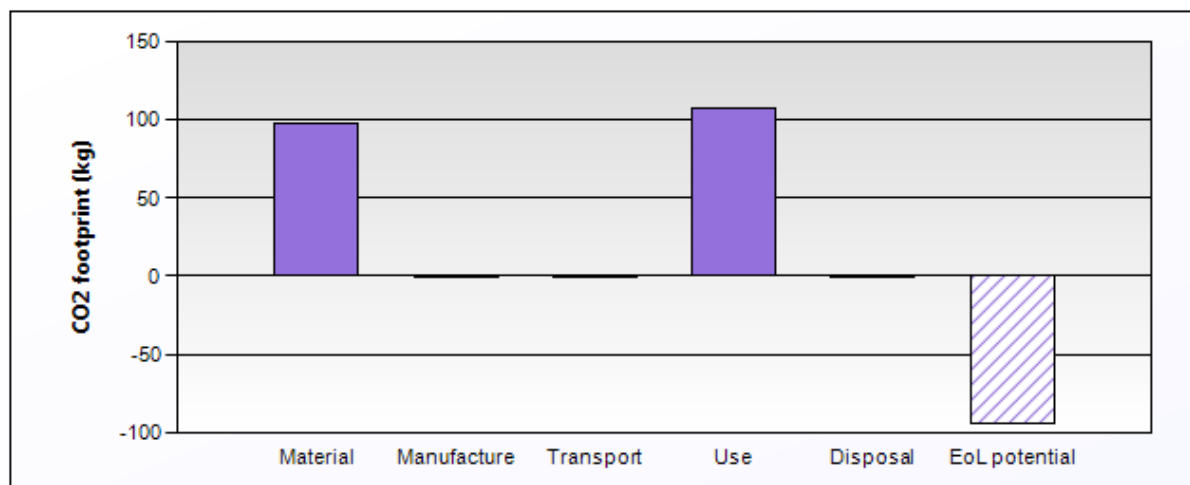


Table 7 - CO2 footprint

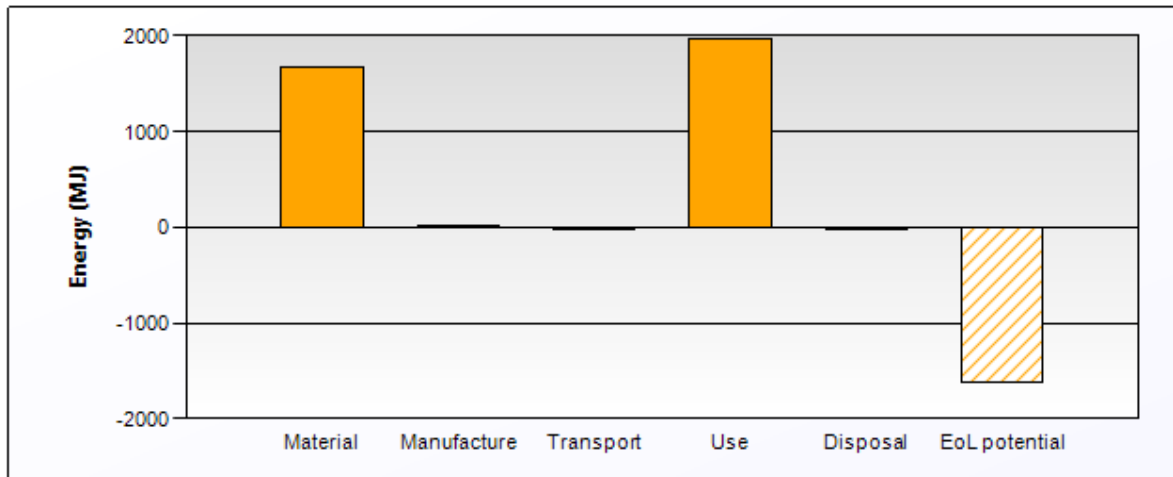


Table 8 – Energy used in each phase (MJ)

Phase	Energy (MJ)	Energy (%)	CO2 (kg)	CO2 (%)
Material	1,67e+03	45,9	97,4	47,5
Manufacture	7,39	0,2	0,555	0,3
Transport	4,3	0,1	0,305	0,1
Use	1,96e+03	53,8	107	52,1
Disposal	1,14	0,0	0,0797	0,0
Total (for first life)	3,64e+03	100	205	100
End of life potential	-1,62e+03		-93,7	

Teseus .prd

NOTE: Differences of less than 20% are not usually significant.

Page 1 / 3

Table 9 - Energy & CO2 summary

	Energy (MJ/year)
Equivalent annual environmental burden (averaged over 2 year product life):	1,82e+03

Table 10 – Energy analysis

In order to display the sustainable and economic advantage Teseus holds it is compared to an example of typical x86 computer as seen in the chart below.

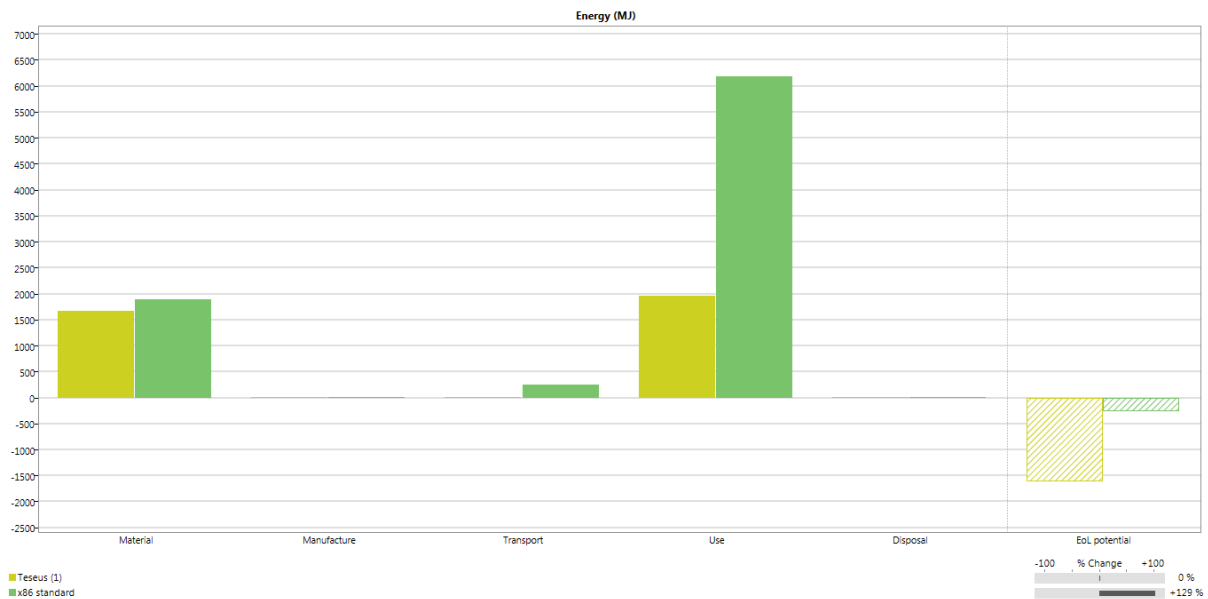


Table 11 – Summary chart

Teseus = yellow

x86 computer = green

The major difference between a standard supercomputer and Teseus is the use of energy. Teseus is designed for the purpose of decreasing the energy consumption of computer processes. The x86 computer has one computer that spends $45W^{76}$ in comparison to the Teseus that uses $3W \cdot 5$ for the same tasks, and it contains the same exact same components except the material used for the rack. The material used as frame or cover for the x86 is of ABS plastic, which takes more energy to produce than the PMMA used for the Teseus rack. The x86 is transported the same distance, but by plane with approximately the same weight which results in higher energy consumption.

Improvements

In conclusion, the heavy weighing object of our prototype, when it comes to use of materials and CO2 footprint is the **controller** that in this case is a desktop computer of x86 architecture. Teseus will in the future have one of the small single board computers as a controller, which means with the desktop gone the total mass will be decreased significantly.

The **cables** used as interconnection between the boards are Ethernet cables. State of the art of interconnection used in supercomputers today is InfiniBand⁷⁷, which will be a more sustainable solution for

⁷⁶ Josep Farré – Technical Director of Neàpolis.

⁷⁷ <http://en.wikipedia.org/wiki/InfiniBand> accessed 15.5.2015

Teseus due to the fact that more and more data needs to be transported and the faster connection you have, less energy will be used.

The **single board computers** used are Raspberry Pi 2s. They are the state of the art for the price, performance, lowest energy consumption and meet the requirements of virtualization and being of ARM (Advanced RISC Machine) available. However, the market is high in growth and new technologies are constantly developed, therefore the lifespan of one board is set to two years.

When designing a supercomputer, the design of the entire **rack material** should be taken into account. The rack of Teseus is made out of transparent PMMA plastic which is more eco-friendly than using ABS plastic. However, the most important thing is to keep the system stable and in best case scenario the case should be able to handle the heat generated from the computers, even if the cooling system breaks down, so no data is lost. Should there be a fire, one must assume that the room of the supercomputer is protected from fire- and water accidents, but the material of the rack should be airy and serve as protection from dust.

Appendix H

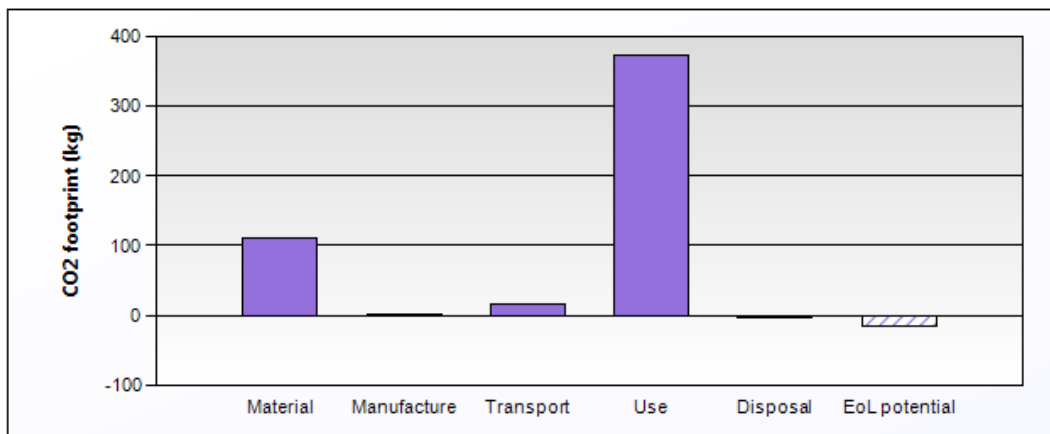
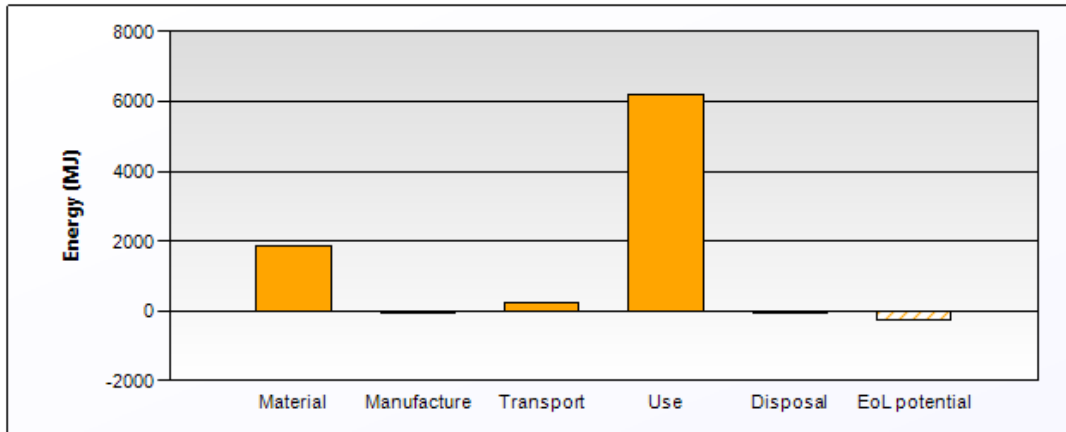


Eco Audit Report

Product Name x86 standard

Product Life (years) 2

Energy and CO2 Footprint Summary:



[CO2 Details...](#)

Phase	Energy (MJ)	Energy (%)	CO2 (kg)	CO2 (%)
Material	1,88e+03	22,6	110	22,0
Manufacture	4,14	0,0	0,31	0,1
Transport	246	3,0	16,4	3,3
Use	6,19e+03	74,4	372	74,6
Disposal	2,25	0,0	0,157	0,0
Total (for first life)	8,32e+03	100	499	100
End of life potential	-270		-15,2	

Eco Audit Report

Energy Analysis

[Energy and CO2 Summary](#)

	Energy (MJ/year)
Equivalent annual environmental burden (averaged over 2 year product life):	4,16e+03

Detailed breakdown of individual life phases

Material:

[Energy and CO2 Summary](#)

Component	Material	Recycled content* (%)	Part mass (kg)	Qty.	Total mass (kg)	Energy (MJ)	%
x86	Printed circuit board assembly	Virgin (0%)	0,5	1	0,5	65	3,4
Ethernet connection	Cable	Virgin (0%)	0,01	3	0,03	2,7	0,1
Controller	Desktop computer (without screen)	Virgin (0%)	3,5	1	3,5	1,4e+03	77,0
Keyboard	Keyboard	Virgin (0%)	0,3	1	0,3	1,1e+02	5,9
Mouse	Optical mouse, with cable	Virgin (0%)	0,2	1	0,2	1,5e+02	7,9
Memory	Hard disk drive	Virgin (0%)	0,2	1	0,2	88	4,7
Computer cover	Acrylonitrile butadiene styrene (ABS)	Virgin (0%)	0,2	1	0,2	19	1,0
Total				9	4,9	1,9e+03	100

Manufacture:

[Energy and CO2 Summary](#)

Component	Process	Amount processed	Energy (MJ)	%
Computer cover	Polymer molding	0,2 kg	4,1	100,0
Total			4,1	100

Transport:

[Energy and CO2 Summary](#)

Breakdown by transport stage

Stage name	Transport type	Distance (km)	Energy (MJ)	%
Total transport of material	Air freight - long haul	6e+03	2,5e+02	100,0
Total		6e+03	2,5e+02	100

Breakdown by components

Component	Mass (kg)	Energy (MJ)	%
x86	0,5	25	10,1
Ethernet connection	0,03	1,5	0,6
Controller	3,5	1,7e+02	71,0
Keyboard	0,3	15	6,1
Mouse	0,2	10	4,1
Memory	0,2	10	4,1
Computer cover	0,2	10	4,1
Total	4,9	2,5e+02	100

Use:**Static mode**

Energy input and output type	Electric to thermal
Use location	World
Power rating (W)	45
Usage (hours per day)	24
Usage (days per year)	3,7e+02
Product life (years)	2

Relative contribution of static and mobile modes

Mode	Energy (MJ)	%
Static	6,2e+03	100,0
Mobile	0	
Total	6,2e+03	100