Adaptive Distributed Real-Time Systems: A Parallel Approach

Joel Gonçalves
pro12009@fe.up.pt
Glossary

ARTS  Adaptive Real-Time System

ADRTS  Adaptive Distributed Real-Time System

CC  “Comissão Científica”, translated to: Scientific Committee

CdA  “Comissão de Acompanhamento”, translated to: monitoring committee

CFS  Cyber-Physical System

DM  Deadline Monotonic

DP  Dynamic Priority

EDF  Early Deadline First

fps  Frames per second

FJP  Fixed Job Priority

FTP  Fixed Task Priority

IoT  Internet of Things

IT  Information Technology

PhD  Doctor of Philosophy

QoS  Quality of Service

RTS  Real-Time System

SOA  Service Oriented Architecture
5. Plan.................................................................................................................................................. 29

5.1 Schedule ......................................................................................................................................... 29

5.2 Risk Analysis ..................................................................................................................................... 32

6. Conclusions......................................................................................................................................... 33

References............................................................................................................................................ 34

A. Appendix. Plan Gantt Chart ........................................................................................................... 37

B. Appendix. List of conferences ......................................................................................................... 38
1. Introduction

The purpose of this document is to propose a PhD thesis research plan for the Research Planning course, of the Informatics Engineering Doctoral program from University of Porto. During this semester, we were challenged to define a thesis plan for our own PhD thesis. This implies we must start by identifying a relevant scientific problem, analysing how the related scientific community is tackling the problem, devising our own hypothesis/methodology, and contextualizing our contribution in the current state of the art. Upon completing those tasks, we can proceed our thesis plan by highlighting the motivations to pursue such problem, exemplify the usefulness in real-world applications, and conclude with a plan for executing the PhD thesis.

There is a relation between this PhD thesis and Artemis Arrowhead project that should be contextualized. Arrowhead is a European project with over 70 public and private institutions from multiple countries. The aim is to address cooperative automation, and enable technology development based on the concept of the Internet of Things (IoT) and Service Oriented Architecture (SOA). This thesis will be financed by this project, and will contribute with concrete implementations to the Arrowhead prototypes. Nevertheless, it should be highlighted that the challenges of this thesis are independent from the project, in the sense that the problem is abstract enough to be identified in other application domains. For practical reasons, it is convenient to demonstrate our approach by instantiate with the project's specific values.

A final remark about the scope of this document, concerns with the fact that we aim to use the feedback on this document for analysing and improve it in order to submit to the Monitoring Committee (CdA).

1.1 Motivation

Concepts such as IoT and Cyber-Physical Systems (CPS) rely on a set of sensor and actuators deployed over real-world objects. These objects are then represented virtually enabling an interaction at logical level for coordinating operations. Eventually, these operations are reflected in the real-world through the actuators which actually change the real-world objects’ state to what was decided in a logical level. This simple conceptual control loop sets the basis of most of the current hot topics in research, as presented in Fig. 1.

Smart cities should be able to coordinate multiple subsystems such has traffic, communication, security, and urban planning. The cooperation between these subsystems, specifically having an accurate state of each one and share that information with the other systems provides an excellent opportunity not only for economic reasons but also for governance and quality of life of the citizens. Smart buildings aim to achieve higher energy efficiency levels and a new set of services for the residents' comfort by monitoring and coordinating the devices within the house. The smart grid concept aims to replace the current electrical grid infrastructure with a new concept of free market where each element can store, spend, buy and sell electricity. Again the monitoring of each house consumption and storage, along with the energy market opportunities can provide not only enough energy to the house...
but also take profit from the situation. Other applications such as factories, electronic markets, traffic, and surveillance systems also require coping with high amount of data in order to have an accurate picture of the real-world state and take actions in their own system or interact with other system to accomplish some objective.

Intuitively, we understand the usefulness of each system individually but it is in the cooperation between them where the maximum value can be obtained. For example, in case of an industrial accident it would be convenient if the traffic around the location could be eased to security and emergency vehicles, while normal vehicles would be forward to alternative routes in order to minimize the impact in the traffic system; security forces would be automatically warned and evacuation plans to the near locations could dynamically be devised based on the current situation. Another example of coordination occurs if the energy profile of each house be analysed, and based on that profile turn on or off devices in the house to save energy; predict energy requirements can help devise a strategy for buying energy in the electronic market; based on the weather, predict the energy generated and eventually sell it to other elements in the grid.

Since all these systems may assume a role of legacy system or even if they are newly developed, it is desirable for engineering and economic reasons to be developed separately and integrated over a platform. For this reason, the most promising paradigm for developing such systems is SOA. In SOA, each system publishes a set of services that are available to other systems which can therefore rely on them to coordinate actions. SOAs can provide the framework for the systems interoperability, however there are many applications where non-functional requirements are as relevant as functional requirements.
Real-time requirements assume a key role in several applications in this domain. Many applications rely on a specific time window to perform control loops or coordinate actions with other systems in order to obtain not only the desired result but also under useful time. These applications usually rely on data received from the sensors, perform some logical operations and then they eventually communicate with the actuators. These flows can be triggered periodically or by events. The former, time-driven triggers, are common for applications that need to obtain the current state of some set of sensors which usually does not require many network bandwidth or computation resources, while the later event-driven triggers are more suitable for demanding tasks where it only makes sense to perform some actions when specific events are observed.

A scenario that motivates us is a distributed system where multiple services are provided by each subsystem using well defined interfaces. The use of these services is enabled by a framework platform which abstracts the communications functional and non-functional requirements. We will address the specific case of real-time applications, whether they are time-driven or event-driven. Depending on this characteristic, the system’s workload is handled using a constant component for the time-driven applications and a dynamic component for the event-driven unpredictability. For visualization purposes, the number of traffic events during night can be considerably lower when compared with a rush hour, or even an emergency situation. As with any real-time system, we are interested in bound this demand and the respective response time the system might throughput, however there is often a cost dimension associated to the use of computational resources.

Nowadays, large information technology (IT) infrastructure is around the concept of cloud computing, where high demand systems are deployed over the cloud platform which is responsible to provide the needed computational resources. This relieves application owners to buy, configure and maintain large IT infrastructure. The business model associated to cloud computing is utility computing where computational resources are charged as utilities the clients use, so the more they use the more they have to pay. In terms of the previous described scenario, this means that a naive approach for deploying the system in the platform would require to bound the system demand (fixed and dynamic) and reserve the resources need. Assuming feasibility analysis was made, the system will probably work but the resource usage would be inefficient since it is dimensioned to the maximum demand possible the system might have. A solution to this problem is by design the system to be adaptive, thus adjusting the resource usage accordingly to the system’s demand while ensuring its time requirements.

Adaptive Real-Time Systems (ARTS) are already disseminated in the scientific community usually associated to online updates, and Quality of Service (QoS) based applications. These two applications are illustrative of the challenges and advantages of using adaptive systems: we need to take into account the time cost associated with the transition period, and we need to be assured we have the resources needed for its operation. Hence, this characteristics can be used to our problem in order to change the system configuration according to demand, by selecting the configuration with the minimum resource consumption that ensures the time requirements are met.
We propose a novel approach for Adaptive Distributed Real-Time Systems (ADRTS) which achieves adaptability following a parallel computation strategy. In one hand we have the adaptive nature of the system trying to minimize the resource usage, while in the other hand we have service parallelization for coping with the demand. This approach aims for systems where demand can be both time and event driven; some services available in the system can be instantiated multiple times in several nodes of the platform; and the demand variability over time forces the system to balance demand coping with resource usage.

1.2 Research Question
In this thesis we want to pursue the following research question:

**RQ1. How to define a service oriented real-time system’s optimal configuration according to the system’s workload dynamics, such that time requirements are met and the global resource utilization is minimized?**

Of course, the answer to this question depends on the system ability to adapt to the workload. This means, there is for each possible workload there is an optimal answer for that optimization problem, but since workload changes over time so should the system configuration in order to remain optimal. For this reason, this thesis aims to propose a methodology for generating a configuration function such that for every demand, the system assigns tasks to nodes in such a way that time requirements are met and global processor utilization is minimized.

1.3 Hypothesis
In this thesis we will test the following hypothesis:

**H1. By creating multiple service instances and reconfiguring the data distribution structures, there is a function such that maps workload to system configuration subject to fulfils all non-functional time requirements of the application.**

Our strategy for the system adaptability is to create multiple instances of the same task and distribute the load over the task instances, as it is done in parallel programming. This way, we can use few resources to perform the computations under low workload, and as the workload increases so does tasks’ instances.

An example of a SOA application deployed in a multiprocessor platform composed by nine processor sites is presented in Fig. 2. We have two applications deployed: RED and BLUE applications. The RED is composed by the service set of services \{A,B,C\} and application BLUE by services \{D,B,E,F\}. Each service can have one or more service instances, and these instances can be shared by the applications, e.g. service instance B’.

On the left side of the figure we have a scenario of low system workload. All services have only one instance, in processor P5 the B’ service instance is shared by both applications, and service instances E’F’ from BLUE are hosted in P6 for minimize resource usage.

On the right side of the picture we present a high workload scenario, where due to a high increment of the system’s workload, the previous system configuration did not hold the non-functional requirements either on time guarantees or resource minimization. The system
adapts by finding the optimal configuration for the current workload and assigns the service instances to the processors. This is an optimization problem where the possible solutions may be achieved by creating multiple instances for the same service, thus giving the system adaptation behaviour. In the specific scenario this implies that: (i) RED configuration changes to cope with 3 instances of service B, (ii) BLUE still shared a service instance with RED but due to the high workload services E and F need a single processor to cope with the activations.

![Diagram of low and high workload configurations](image)

**Fig. 2.** Example of using a parallelization approach for a SOA based application over a ADRTS.

### 1.4 Expected Contribution
The most relevant contribution this thesis aims is to provide a methodology for converting a set of SOA applications with real-time requirements, to an ADRTS where some of the services can be parallelized. Other important expected contributions are:

- Provide a holistic schedulability analysis to a specific platform, where the nodes and network are yet to be defined.
- Provide a robust system’s configuration function which ensures the minimum resource utilization according to the system’s workload.
- Develop a framework which implements the system’s configuration function and enables the implementation of the system.

### 1.5 Document Structure
We hereby outline the structure of this document. In section 2, we provide an overview on the real-time concepts useful for this thesis. We highly recommend the reader to read them if real-
time concepts are not clear, and also for the sake of understanding further notation. It starts by providing the reader with the fundamental concepts of RTS like the definition and the most important test made for assessing the validity of RTSs. Then we advance to the holistic analysis which is more appropriated for distributed systems.

In order to highlight the novelty of this thesis, in section 3 we present related works. The section starts by describing the systematic review approach process taken. Then, we explain how traditional time analysis does not consider all our requirements, thus it is not suitable for our needs. After that, we cover the diversity of techniques for implementing service oriented adaptive applications. By the end of this section, we compare with a work that establishes very similar scenario and assumptions, and provides an excellent input for our approach.

After creating the contrast between the objectives of this thesis and the related work, in section 4 we explain the methodology for our approach. This section aims to provide a generic iterative step procedure to assess the validity of our hypothesis.

For the purpose of presenting our thesis research plan, in section 5 we propose an expectable schedule in a format of project. The first subsection focuses on explaining each phase of the project and how they interact with the thesis document. Then, a risk analysis is reported to provide insights of the risks identified and possible solutions for such situations.

Finally, in section 6 we recap this document and draw some conclusions about what we expect to contribute with this thesis.
2. Real-Time Systems Background

When a system has real-time requirements, this means that the system design should take into account time instants as a relevant element in the development process. Examples of such requirements can be a video running at 25 fps, where for each second elapsed our application must be able to processes and deliver 25 frames. Therefore on a real-time system the correctness of the system itself does not depend only on its accuracy but also on its bounded response time.

Depending on the application, RTS are configured in such a way that ensures the time requirements are met. This issue is relevant because as developers implement each individual task of the system, they eventually have to communicate and share computational resources between them. Thus, there are two important concepts associated [1):

- **Feasibility Analysis.** This analysis raises a generic decision problem which aims to assess if there is any configuration (on the most traditional way the scheduler policy) such that the system will always meet its time requirements. In practice is similar to check when we receive a workload if it is possible to dispatch it on time, with the current working process or any other. A positive answer to this test means there is at least one way to configure the system such that it will give real-time guarantees. If the answer is negative then it is impossible to always meet the time requirements of the workload.
- **Schedulability Analysis.** This analysis is a specific case of the feasibility analysis, meaning that given a scheduling policy check if the system will meet the time-requirements or not. In practice this means that if the developers propose a specific configuration for the system to work, and it passes in this test then they know that the solution proposed will work. If the test is negative, then it means that the current solution will not work and there might be a solution such that the system can meet the time requirements. This may have varying precision: (i) **exact,** if correctly identifies all schedulable and unschedulable tasks of a system; (ii) **sufficient,** if correctly identifies all unschedulable tasks, and eventually identify false unschedulable tasks.

Due to traditional reasons, most RTS research was made on the a single processor with multiple tasks that need to share the processor time according to a scheduling policy in order to ensure the deadlines of those tasks. Nowadays, most research is done using more complex scenarios such as distributed systems and adaptive systems. Despite this complexity, the fundamental unicore scheduling modelling and strategies are the basis of these more complex systems.

2.1 Scheduling and Resource Management

Scheduling and Resources are tightly coupled. In its broad sense, scheduling means assigning resources to tasks, and the other way around is also valid. In this section we present the basic concepts for modelling and understanding RTS.

2.1.1 Computation Model

When analysing a real-time system we consider all software activities as a set of $n$ real-time tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$, where each **task** is an abstraction of a set of instructions that must be
executed by processing units. Tasks can assume two possible states: (i) wait state, when the
task is stopped, and waiting for the next release (Not the same as blocking); (ii) compete state,
when the task is active, and is competing to run in the processor. It is important to highlight
that tasks might be considered cyclical in the sense that after completing their execution they
spawn after a period of time or acyclic when they are triggered by an external event. The most
basic structure for holding a task is the following:

$$\tau_i = (C_i, D_i)$$ (1)

Where $C_i$ is the worst-case execution time (WCET), and $D_i$ as the relative
deadline. More parameters are added according to the type of task.

Each task instance is designated by job, in the format $\tau_{i,j} (j = 1, 2, ... )$. A job is an
instance of a task’s compete state, placed between two wait states. There are three relevant
real-time concepts associated to jobs: (i) release time, the instant of time when a specific job is
created (change from wait to compete); (ii) completion time, The instant of time when a
specific job goes to the wait state (change from compete to wait), note that this does not
mean that the job complete its purpose; (iii) execution time, The amount of time a job takes
from its release time to completion time (interval since the beginning to the end).

A job can be modelled as:

$$\tau_{i,j} = (c_{i,j}, r_{i,j}, d_{i,j}, s_{i,j}, f_{i,j}, R_{i,j}, R_i)$$ (2)

Where $c_{i,j}$ is the computation time, $r_{i,j}$ is the release time, $d_{i,j}$ the deadline, $s_{i,j}$ as
the start time, $f_{i,j}$ the finishing time, $R_{i,j}$ as the response time ($R_{i,j} = f_{i,j} - s_{i,j}$), and $R_i$
denotes the maximum response time for that task ($R_i = \max_j R_{i,j}$).

For the sake of clarity, deadlines may be presented in one of the following formats: (i)
relative deadline, the fixed amount of time, where after a release the job successful
completion is considered useful and is usually associate with tasks; and (ii) absolute deadline,
the specific instant in time, given by the job’s release time plus its relative deadline, where a
job’s successful completion is considered useful and is usually associated with jobs.

Two final general concepts urge to present. The scheduling window is the time interval
between the job’s release time and the absolute deadline. In practice this is the time window
the job actually has to perform its computations, even if the job might be in a waiting state
during most of the time (e.g. higher priority jobs are competing). Another extremely
important concept for this thesis is the Response Time which is the amount of time between
release time and completion time. This means the actual time taken by the job to finish its
execution, which in the case of RTS we force to be within the scheduling window. This
response time concept is then extended in DRTS to end-to-end response time.

2.1.2 Real-Time System Classification
The deadline concept is an important concept for characterizing a RTS. In terms of the
deadlines fulfilment, if respecting all deadlines is a hard constraint, then we classify our system
as a **Hard RTS**[2], [3]. If it is tolerable for some deadlines to be missed, then we have a **Soft RTS**.

Concerning the relation between deadlines and the arrival of jobs, we have three possible classifications for the RTSs according to its deadlines: (i) **implicit deadlines**, if all tasks’ deadlines are equal to the respective minimum arrival period; (ii) **constrained deadlines**, if exist one task with the deadline shorter than the minimum arrival period, but all respect the condition to be less or equal to the minimum arrival period; (iii) **arbitrary deadlines**, if the previous restrictions do not hold.

### 2.1.3 Task Periodicity

Tasks can be classified according to their periodicity, meaning that each task to be executed in a RTS can be either: periodic, sporadic or aperiodic. Along with the scheduler policy, this configures the necessary inputs for assessing the feasibility of a system.

**Periodic Tasks**

In section 2.1.1 we defined that tasks are assumed to be cyclical, meaning they are spawn periodically in the real-time system as described in [4]. We now review real-time systems composed only by periodic tasks to be executed in a single processing unit environment.

A generic model for describing periodic tasks $i$ ($i=1...N$) in $k$ jobs ($k=1,2,...$) is:

\[
\begin{align*}
    r_{i,k} &= \Phi_i + (k - 1)T_i \\
    d_{i,k} &= r_{i,k} + D_i
\end{align*}
\]  

(3)

In the equation 1, $r_{i,k}$ is the release time of job $k$, $\Phi_i$ is task $i$’s first job release time (meaning $r_{i,1}$), $T_i$ is the task period, and $D_i$ is the relative time deadline.

With such equation system we can indeed describe any periodic task, however to effectively execute all tasks we must define a schedule policy to manage the execution order. A more common notation to model a periodic task is:

\[
\tau_i = (C_i, D_i, T_i)
\]

(4)

Where $C_i$ is the worst-case execution time (WCET), and $D_i$ as the relative deadline, and $T_i$ is the task period. Despite the usefulness of periodic tasks, many tasks have an event-driven nature which is better modelled by aperiodic tasks.

**Aperiodic Tasks**

Many control tasks are modelled as periodic tasks, such as data acquisition, sensory data processing, actuator controls, and so forth. However some systems require that non-periodic tasks are activated by external factors. These types of tasks are referred as aperiodic tasks since they do not have a known period. An example of such system would be a car brake system like ABS where the system needs to free the wheels but it is only activated when the driver decides to press the brake pedal.
There are two main scenarios for systems with aperiodic tasks. The first one is when the aperiodic tasks are always less important than the periodic ones. The strategy in this scenario is to minimize the response time of the aperiodic tasks while respecting the periodic tasks.

Another scenario is when aperiodic tasks require hard-real time deadlines and as consequence we must guarantee at offline time the system schedule feasibility. Here, we must model aperiodic tasks by adding an additional variable referred as the minimum interarrival time which defines the minimum period possible for an aperiodic task to arrive to the system.

A very important concept associated to aperiodic task systems is the **busy period**. Since we can face multiple jobs from the same task at a given time, it is possible to calculate to worst possible schedulable situation where the maximum (possible) number of jobs is on the system at a given time. The busy period corresponds to the sequential time slots from the worst job arrival scenario to the instance the processor is free again. In practice we claim that if the system is schedulable in that situation, which is the worst we admit, and then we are confident the scheduling algorithm will hold for all less stressed situations.

This type of task is usually associated with soft real-time systems since the arrival of tasks is not bounded, there might not be guarantee of that the desired deadline might be met due to processor utilization restriction. Another approach is to use Sporadic Tasks for restricting the arrivals.

**Sporadic Tasks**
The idea of sporadic task systems is not only to define an upper bound where we can guarantee the system can cope, but also define a lower bound where job releases are controlled. In a sporadic task system, a task is defined by:

$$\tau_i = (e_i, d_i, p_i)$$

(5)

Where $e_i$ is the upper bound on the amount of time to execute the computation (WCET), $d_i$ is the relative deadline, and $p_i$ is the minimum time between two consecutive releases. In essence, sporadic tasks are a generalization of periodic tasks since the later have the restriction of $p_i = T_i$. The main focus of this thesis is around the sporadic tasks systems.

**2.1.4 Scheduling Algorithms**
Scheduling algorithms are one of the most important features in RTS since it scheduling plays the decisive role of assigning resources to tasks, where processor time is usually the most predominant resource. Due to the heterogeneity of platform where RTS run, there are two main classes of scheduling algorithms: (i) **partitioned schedulers**, when the algorithm assumes a maximum of one processor per cluster, in other words as if all processors available were considered as individual processors for assign the tasks; (ii) **global schedulers**, aim for a common queue of jobs to be dispatched according to the available processors.

In this thesis we will focus on priority based scheduling algorithms, but it should be mentioned other policies such as timeline scheduling, and reservation. Priority based scheduling algorithms may assume one of the following categories: (i) **Fixed Task Priority**
(FTP), in this class all jobs within a task have equal priority; **Fixed Job Priority** (FJP), in this class exists jobs within a task with different priorities; **Dynamic Priorities** (DP), priority may change from release time to completion time.

For this thesis we also consider the use of three relevant priority based scheduling algorithms. The **Deadline Monotonic** (DM)[5] is a FTP scheduling algorithm where the deadlines are set in design-time accordingly to the shortest relative deadlines, thus the shortest the deadline higher the priority. Still in the FTP class we also have the **Rate Monotonic** (RM)[4] which is similar to DM except that in this case the reference value to assign the priority is the period, thus the shorter the period (high frequency) the higher the priority. This algorithm is known by setting the lower processor utilization bound possible, meaning that there is no feasible scheduling algorithm that has less processor utilization.

Another important algorithm is the **Early Deadline First** (EDF)[6] which is considered a DP scheduler. This algorithm is also famous due to establishing the maximum utilization bound, meaning that if a RTS is not schedulable by EDF, then it is not feasible since it breaks the utilization bound of 100%.

### 2.1.5 Real-Time System Analysis

There are important concepts that provide useful tools to analyze RTS before presenting the analysis. We will emphasize this background around FTP scheduling, in particular the case of constrained or arbitrary deadlines since it is the most probable way we will use in our investigation.

The utilization concept denotes the relation between the task execution and the lower time bound between instances. In practice this represents the percentage of time a job would require to successfully complete between two releases of the same job. The utilization of a processor is given by:

\[
u_{sum}(\tau) \equiv \sum_{\tau_i \in \tau} u_i
\]

The task with higher utilization:

\[
u_{max}(\tau) \equiv \max_{\tau_i \in \tau} u_i
\]

The generalized density of a task establishes the relation between the task execution and, the minimum between its relative deadline and its minimum arrival period.

\[
\lambda_{sum}(\tau) \equiv \sum_{\tau_i \in \tau} \lambda_i
\]

The highest dense task is given by:

\[
\lambda_{max}(\tau) \equiv \max_{\tau_i \in \tau} \lambda_i
\]

As for the constrained density of a task, it establishes the relation between the task execution and the relative deadline and it is given by:
In this case the highest dense task is given by:

\[ \delta_{\text{max}}(\tau) \equiv \max_{t_i \in \tau} \delta_{t_i} \]  

The load of a system expresses the demand bounds over the time, in other words the system’s workload, and it is given by:

\[ \text{load}(\tau) \equiv \max_{t>0} \left( \frac{\sum_{t_i \in \tau} DBF(t_i, t)}{t} \right) \]  

Where the DBF function is defined as:

\[ DBF(t_i, t) = \left( 0, \left( \left| \frac{t - d_i}{p_i} \right| + 1 \right) e_i \right) \]

**Partitioned Scheduling Feasibility for Fixed Task Priority**

For a platform of \( m \) partitioned processors, under local DM we can assess the schedulability test for arbitrary sporadic RTS:

\[ m \geq \frac{\text{load}(\tau) + u_{\text{sum}}(\tau) - \delta_{\text{max}}(\tau)}{1 - \delta_{\text{max}}(\tau)} + \frac{u_{\text{sum}}(\tau) - u_{\text{max}}(\tau)}{1 - u_{\text{max}}(\tau)} \]  

Under the same conditions but for constrained-deadline systems we have the following schedulability test:

\[ \text{load}(\tau) + u_{\text{sum}}(\tau) \leq m - (m - 1)\delta_{\text{max}}(\tau) \]  

**Global Scheduling Feasibility for Fixed Task Priority**

The following feasibility test holds for a platform of \( m \) processors unit-capacity processors, if for every \( \tau_k \), there exists \( \lambda \in \{ \lambda_k \} \cup \{ u_i | u_i \geq \lambda, \tau_i \in \tau \} \) such that either equation 16 or equation 17 is true.

\[ \sum_{i \neq k} \min(\beta_k(i), 1 - \lambda_k) < m (1 - \lambda_k) \]

\[ \sum_{i \neq k} \min(\beta_k(i), 1 - \lambda_k) = m (1 - \lambda_k) \text{ and } \exists i \neq k: 0 < \beta_k(i) \leq 1 - \lambda_k \]

Where:

\[ \beta_k^i(i) \equiv \begin{cases} u_i \left( 1 + \max \left( 0, \frac{p_i - e_i}{d_k} \right) \right) & \text{if } u_i \leq \lambda \\ u_i \left( 1 + \max \left( 0, \frac{d_i + p_i - e_i - \lambda d_i}{u_i} \right) \right) & \text{if } u_i > \lambda \end{cases} \]
2.2 Holistic Analysis Theory
A holistic analysis for a distributed system is a schedulability analysis which takes into account the communication inherent to the distributed nature of distributed systems. Traditional unicore or multicores with shared memory are usually model as a set of tasks which must be scheduled by \( m \) processors. However due to the communication specificities, and network protocol, the previous assumptions do not hold. So, we need not only to care about the local scheduling, but this scheduling must take into account the worst execution possible from the network part.

In a distributed scenario, the response time of a task is extended from local perspective to a distributed perspective called end-to-end delay or end-to-end response time. This comprises the time the initial task that starts the activity until the last task finishes the activity, assuming that the activity is decomposed by several tasks that send messages between each other. Using the example from Fig. 2, that would be the time from the service A receives a message and processes, then send to a service instance B for processing and send to service C which after a final process completes the activity execution; all the time from arrival to A to the completion from C is the end-to-end delay.

2.2.1 Communication Technical Specifications
The technical implementations of communication are abstracted and the following steps are considered:

1. Generation Delay. The time taken from the arrival of the task to the generation of the message.
2. Queuing Delay. The time taken from having access to the transmission queue and insert the message.
3. Transmission Delay. The time taken for transmitting a message from the source node queue, removes from the transmission queue, and reaches the destination node receiver queue.
4. Delivery Delay. The time taken from the receiver task to access the message from the receiver buffer.

This generic specification highlights the technical considerations to be taken into account when modelling the communication. For instance, while step 1, 2, and 4 can be seen as an access to a local shared resource (covered in section 2.1.5); step 3 can also be a shared resource but accessed virtually all task in the system (hosted in all nodes) but is governed by a particular network protocol.

2.2.2 Network Protocol
For the purpose of this document we will focus on a priority based shared bus scheme as our target scenario. An example of such network is CAN. The basic concept is of a global pre-emptive priority queue which delivers data accordingly to the messages’ priorities.

Similar to tasks, each message sent by the tasks also has a single fixed priority. However, due to the network characteristics, messages are decomposed in smaller parts called packets. A message is fully transmitted after all packets arrive to the destination. After arriving
all packets, the message is reconstructed and only then can be delivered to the receiver task. Note that message transmission is pre-emptible but packet transmission is not.

Using arbitrary deadlines, we can consider the following case of busy period:

\[ w_{m,q,k} = \left( qC_m + k + \sum_{\forall n \in hp(m) \setminus outgoing} \left[ \frac{w_{m,q} + f_n}{T_n} \right] C_n \right) \rho \]  \hfill (19)

Where \( q \) is the job invocation under the busy period, \( C_m \) is the worst case transmission time of message \( m \), \( k \) is the worst-case response time of the \( k \)th packet of message \( m \), the \( hp(m) \) is the high priority messages, \( outgoing \) are the set of messages actually transmitted through the bus, the sum represents the interference from previously advents such as previous packets, the jitter, and \( \rho \) stands for worst case transmission time of a packet.

### 2.2.3 Holistic Approach

A holistic approach in this context is an analysis which takes into account both processor and communication schedulability. We can observe both processors and network as processing units which interact between each other.

A characteristic of this analysis is its inheritance since tasks send messages to trigger other tasks and so on. Thus the analysis needs to consider not only the current task and the associated interference from higher priorities within the processor, but it also needs to take into account the possible precedent tasks’ time. The following busy period is given by:

\[ w_i(q) = (q + 1)C_i + \tau_i(w_i(q)) + \sum_{j \in hp(i)} \left\{ \begin{array}{ll} v_j(w_i(q)) & \text{iff is a deliver} \\ [f_j + w_i(q)]/T_j & \text{otherwise} \end{array} \right. \] \hfill (20)

The first two elements of the right side of the equation take into account the computation time of the \( q \) job elements, and the scheduler tick. The last part establishes a difference between deliver and sender, where the major difference is that the deliver needs to consider the packet handling time.

The interaction between processors and network is described in section 2.2.1. The implications of this interaction are reflected on the jobs jitter. The jitter is the difference between the task arrival and the task release, which in our context is: the time between the arrival of the messages that trigger the job, the time to compose the message and finally the time for the job to become active from the scheduling perspective.

\[ J_d(m) = r_{s(m)} + a_m + r_{deliver} + T_{tick} \] \hfill (21)

Where the times from the right side of the equation are the jitter from the sender task, message \( m \) transmission time, the deliver packet handler, and the scheduler tick.

In essence, the holistic analysis merges local scheduling, and communication scheduling by inheritance where the receiver task inherits jitter from the previous one, and by sending a message to the next task will make the new task inherit the jitter. The jitter concept is the main common denominator in the interaction. Moreover, both computations at local scheduling and network scheduling are domain specific and may only take into account the “received” jitter.
3. Related Work in Distributed Adaptive Real-Time Systems

The related work from this document was gathered using a systematic review process. The process is composed by an initial set of the objectives and search strategy, after that it continues over results refinement in order to select only the publications that matter to our thesis.

This work can range from RTS communities to distributed and parallel computing communities. Our major concern is with real-time issues, thus we decided to focus our research mainly through the view of real-time community, with particular interest for the ARTS with specific interest for distributed approaches.

Our strategy for selecting publications to be reviewed consisted in considering all articles from distinguishes publication sites within the time window of 2007 to 2013. Then evaluate those articles appropriateness using quick evaluations which depend on the theme suggested by the title. In the third step the previously admitted articles are the filtered by interest to the work and priority, so some few are discarded and are organized on tiers which describe how important the article is related to our work. For a 4th step, based on the experience from reading, we propose taxonomy and classify each paper accordingly to the taxonomy. Based on the taxonomy we can then have an overview of the area and understand relation between different approaches. The results are compiled and explained in this section, for more detail on the systematic review please consult publication sites check appendix B.

Concerning references, for the purpose of our background (section 2) and for complementary information, we also considered references from papers outside the systematic review, due to the obvious time and publication site limitations above described. The references were obtained from: (i) the book “Handbook of Real-Time and Embedded Systems”[7] which was used in the initial steps of our investigation; (ii) direct references from the articles captured in the systematic review.

3.1 Time Analysis

As covered in section 2.1, the traditional real-time approaches focus on a single resource, usually a single processor. Uniprocessor scenarios have lost many research to multiprocessor platform due to the real-time community anticipating that in a near feature low-cost multiprocessor platforms will be the mainstream platforms for both “normal” and embedded systems. Since in our work we focus on distributed systems, unicom and most multiprocessor models do not capture our requirements.

More related with our approach are the holistic (covered in section 2.2) and the compositional [8], [9], [10], [11] approaches for analysing the system schedulability. In their own way, they extend the analysis to distributed system by taking into account precedent relations between tasks and the network behaviour. Despite its usefulness for analysing distributed systems, they are not suitable for our purpose because these analyses assume the task system is fixed.

Using an adaptive approach, when the system changes from one configuration to another, there are multiple changes that are applied to the task system. In our specific distributed approach, the number of tasks and jobs is dynamic and the task assignment to the
nodes is also dynamic. For this reason, both Compositional and Holistic traditional approaches can only model a single configuration at a time not the multi-configuration space.

3.2 Service Oriented Adaptive Applications

Many SOA applications already have an adaptive and distributed nature, due to the nature of the concept of sharing services with other entities and take advantage of the available resources (composition). In this subsection we cover several related works to understand the state-of-the-art on SOA applied to real-time applications and how we relate these works with our proposed approach.

3.2.1 SOA Frameworks

There are multiple SOA Frameworks for addressing service oriented computing in real-time environments. Example of such frameworks are presented in [12], [13], [14], [15], and [16]. These frameworks can be divided in two major categories: reservation approaches and integration approaches.

Service reservation in [13], [14], [12], are frameworks that use a simple concept in real-time which is resource reservation. Meaning that applications will have a resource set reserved available only to them, meaning that conceptually the application is executing alone in a set of resources. This approach is very common in mixed-criticality RTS, as showed in [12]. Due to be reservation intensive, at both local and network level, the system resource usage fairly inefficient. For reserving a set of resources, the WCET is usually taken into consideration so the resource reservation will be pessimistic. In our approach we intent to share the resource, meaning that no reservation will be use.

Concerning real-time SOA architectures, there is also several approaches of extending the standard SOA infrastructure to with real-time implementations for using specifications design to real-time. Examples of such approaches are [15] and [16]. These approaches do not have a holistic perspective of the system, meaning that they neglect the network component in the sense that it is assumed to have some guarantees (similar to reservation approaches). Our approach addresses takes network into account, thus we also aim for hard real-time systems.

The major purpose of these frameworks is to ease the development of SOA applications and easily enable real-time guarantees. None of the previous frameworks addressed our problem of adapting the system configuration to the current workload and resource consumption.

3.2.2 Task Level Adaptation

Task level adaptation addresses adaptation techniques for application accomplish their requirements during their execution by changing the task set associated to the services. Perhaps the most common application of adaptation techniques is related to QoS applications. The concept is that depending on the resources currently available, the services change their quality and consequently their resource usage in order to deliver the best quality possible solution. Some works consider only this QoS adaptation at task level, like [17], [18]. Adaptation
at task level is certainly an important technique, however in a context of distributed systems not taking into account the network reduces its applicability to hard-real time.

Task migration is another relevant technique used for adapt the system accordingly to the workload. This is particularly used in the grid computing community, as showed by [19], [20], [21]. The main idea is to migrate, during run-time, a set of tasks from a node to another in order to balance the workload. However, this approach usually has a relatively high cost associated while we approach the load balance problem by switching mode within the nodes. Furthermore, the task migration decisions are based on probabilistic predictions of the future workload, therefore it is not an appropriate approach for our hard real-time approach.

Another application of adaptive distributed systems is in on-line updates, to change the task system in run-time for swap a subset of tasks by another one. From a RTS perspective, this procedure is associated to mode change protocols, where the there is a defined policy to change from one configuration to another; each mode change is analysed for its feasibility in order to proceed with the request execution. Examples of such applications in the real-time update can be found in [22], [23].

Some works like [24], [25] specifically address energy consumption by adapting the task system. This is made by turning off the processor, changing the frequency or voltage; despite the approach this implies that the WCET are “dynamic” and therefore the system needs to adapt. A good generic perspective of this area can be found in [26]. In our approach we intent to optimize resource usage, where this generic statement is on purpose due to the diversity of resources available. Energy can be saved, for instance, if we minimize processor utilization; nevertheless it is not the main purpose of this thesis.

The diversity of techniques is high, and usually due to real-time constraints implies low level concerns to implement such systems. Compared with the real-time SOA frameworks, this approach allows specific tune to specific applications, in the sense that we trade efficiency of this approach (e.g. adequate for embedded approaches) by ease of developing services which are managed by the framework. This custom design approaches make the reuse of such system not likely usable for other systems, thus usually only the techniques reported by these works may be useful by the community.

Most task level approaches for adapting the RTS do not take into account the network effect on real-time calculus. Plus, most do not fulfill requirements for hard RTS due to their probabilistic approaches or non-deterministic techniques. Our approach does use task level adaptation, since we aim to: multiply services (along with tasks associated), and change mode in run-time to the best configuration. Note that we do not consider task migration, all modes of the system are determined in design time and we assume that all the needed resources (e.g. binary code) for executing a task are assign at deploy time; this is possible since we know in design time which tasks may be executed in each specific node.

3.2.3 Flow and Network Level Adaptation
A very active part of real-time SOA based systems community provides adaptive behaviour to the RTSs by either change network configuration or change the application workflow during run-time.
In [14], [27], [28], the authors not only consider the network influence on the system but also provide concrete solutions to address such problems. Obviously, ensuring time guarantees at network level also needs to be followed by time guarantees at task level. We are interested in addressing the network properly but using priority based approaches, not by reserving bandwidth.

Probably the most tackled problem in SOA, not only in the real-time niche, is service composability problems [29]. The basic principle is to adjust the application graph flow in run-time according the available service instances. These works [15], [30], [31], [32], address this problem from a soft real-time perspective since they are not time bound. Plus, in order to cope with run-time responsiveness, the algorithms deliver sub-optimal solutions for the composition problems.

These works and our proposed solution share a common concern of modelling the network as an important element of the system which must be taken into account for accomplish real-time requirements. However, the works that address hard RTS use reservation, while the priority based are focused in soft real-time (e.g. multimedia).

3.2.4 Hybrid
Most works in service oriented application with adaptive behaviour have a hybrid classification since they usually combine task, network, or flow changes to adapt the system. The typical application domain are QoS distributed application where exist service instances with different quality throughputs. Hence, the different QoS service instances provide different task sets when the quality changes, and the existing QoS service instances also represent a composability problem where the composition algorithm must adapt the application dataflow accordingly to the current available service instances. Relevant works in this area are available in [18], [28], [32], [33], [34], [35], [36].

Our proposal can also be classified as an hybrid approach due to the mode configuration and service parallelization associated to task level techniques, and load balancing to the multiple service instances can be associated to flow level techniques. However, all of these approaches do not address a fundamental challenge of our approach: service instances are not fixed, since our approach considers the possibility to create multiple service instances to cope with the workload. As consequence, in the previous presented work the system developers need not only to develop the services himself but also assign the service instances. Naturally, after setting the number of instances for each service the system is restricted to those. In practical example, it may happen that a QoS application may have to reduce the quality of all users to cope with the demand, instead of the system generate a new service instance so the application can continue to deliver the maximum quality. This vision and requirements associated to such approach distinguish our work from the previously presented.

3.3 Distributed Adaptive Real-Time System
This subsection is dedicated to this specific paper [37]. Negrean et al address a scenario of distributed mode change taking into account the network. More specifically, the paper aims to set a mathematical framework to support the analysis of such system.
This work is relevant due to the fact that the scenario and requirements of this work being so similar to the ones we intent to use, that the most part of the analysis can be reused to our own approach. The characteristics of the scenario are:

- **Multi-mode distributed RTS.** This term is equivalent to our task level adaptation, in a distributed RTS. Each node can operate in several different modes. The processing units are shared, hence no reservation.

- **Mode Change:**
  - **With periodicity.** Mode change protocols with periodicity allow a subset of tasks (*unchanged tasks*) to continue their execution during mode change.
  - **Asynchronous.** New mode tasks do not need to wait until former mode tasks finish.

- **Task activation.** Tasks can be either activated periodically (time-driven) or can be activated based on input received by other tasks (event-driven).

Using such scenario, the authors main concern is to model such system in order to develop an robust analysis that ensure the system time requirements even during adaptation. As input, the authors consider the task system, an application flow graph for the dependencies, and the workload function with time as dependable variable.

Furthermore, the authors also consider two important requirements: (i) the system mode change is triggered by an initial node that propagates the mode change requests according to an oriented noncyclic graph; (ii) adaptation requests may arrive while other mode change is undergoing. The first item ensures a realistic scenario where based on the application flow, each node involved is triggered to change after the previous node completed. As for the second item, this implies that the system is not alternating between modes: stable-adaptation-stable. Instead, after an adaptation we also consider the possibility of a new adaptation request to override the precedent.

Despite being a related work, there are fundamental differences between our work the authors’ work. An obvious difference is that the author’s problem is a decision problem, since the aims for assessing the schedulability of a task set, system configuration set, and the applications’ flow. Ours is an optimization problem where given a task set, the applications’ flow, and workload function we seek for an adaptation connected graph such that maps each workload instance to the correspondent optimal system configuration subject to systems’ configuration transitions being timely safe. Thus, while the authors work focus on analysing a set of system configuration for schedulability purposes, we focus on find the most appropriate configurations such that mode changes safely transit between optimal configurations that minimize resource utilization.
4. Methodology
A methodology is similar to an algorithm for specifying the tasks in an ordered way in order to accomplish the thesis objectives. This methodology covers from the early specification to the validation of the hypothesis.

We can divide the methodology in four distinct phases. First we face a problem formulation which requires understanding the domain and creating the proper structures to fit it into the existing literature tools. Then, we enter the core RTS calculus where we need to bind the system time response. After mapping our original problem to a RTS structure, we need to interpret that structure and covert it to an appropriate input for the optimization criteria we are aiming to optimize. Finally, we develop a system implementation to validate our approach.

The methodology is not the same as the thesis plan, however it is fully reusable for most key tasks presented in section 5.

4.1 Modelling and Specification
The first challenge we face in this section is to model our platform. Currently we have no defined multiprocessor platform and the network infrastructure, both elements compose the platform and are decisive to a proper modelling. The possible combination of execution platform and network infrastructure is too large and a very generic model would be not only very hard to obtain but also very pessimistic for the real-time calculations.

In this phase, it is also important to map the system from a SOA perspective, which is the perspective of developers, to a real-time task system. In its general form this problem implies that we need to carefully choose an appropriate task resolution (e.g. one for service), along with all important real-time system task variables such as their periodicity, deadlines, WCET, etc... From the specific perspective of our hypothesis, we need to specify which services can be parallelized (for creating multiple instances), and which specific set of instructions are executed to restructure from one dataflow to a new dataflow (for calculating the cost of adapting load policies).

We also select the optimization criteria associated to this process. During the thesis we focus on the global processor utilization but as later explained in section 4.5 this procedure is robust enough for being applied to multiple situations apart from process utilization minimization.

4.2 Assumptions and Constraints
There is a significant reason why assumptions and constraints have own section separated from the Modelling and Specification. The main reason is the authors’ intuition that this is a very flexible part of the process, and it is very likely that we may tight or relax some of the concepts here described during this thesis.

Please note that changes in either assumptions or constraints have a tremendous impact on the real-time calculus framework. They also provide excellent tracks to pursuit in future works.
Assumptions in our context are situations or phenomena that we formulate as a sentence, and we take for granted or to always be true, even if we recognize that in real world situations that might not be the case. We assume the following:

A1. **All tasks produce the correct result.** We assume the task produce the correct result, so we focus on the time problems.

A2. **There are no errors or faults at either hardware or software.** We take for granted that the hardware will always behave correctly and the software has no bugs.

A3. **Messages sent by the network are reliable.** We assume both network signal/velocity is constant and messages are not lost during transmission.

A4. **Workload admission is ensured externally.** We assume the arrival of new workload always respects the temporal and space restrictions defined in design-time, if for some reason the workload do not respect then it is handled before entering the system.

A5. **Closed Platform.** Both platform’s resources and systems’ applications are static and defined at design-time.

A6. **Processing speed constant.** Each processor has a fixed velocity defined at design-time.

A7. **Task execution constant.** Tasks will always have the same execution time which is equal to the respective WCET.

A8. **Real-Time Task System knowledge.** We assume we know all parameters relevant to the real-time task system and they are correct.

Constraints in this context are restriction we impose to our system in order to bound or make the scenarios more realistic. The constraints are:

C1. **Deadline fulfilment.** The time deadlines must be met.

C2. **Deadlines are fixed.** Each real-time task has a relative deadline associated that never changes.

C3. **Fixed task assignments.** Some tasks may be restricted to be executed in a specific processor.

C4. **Task parallelization.** Only tasks signalled for parallelization can be effectively parallelized, meaning that are the only ones with multiple tasks replicated.

C5. **Job execution.** Only one job can be running at a given processor.

After establishing the appropriate assumptions and constraints from both rational point of view and from the project’s stakeholders, we can proceed to analyse the system feasibility as a RTS.

### 4.3 Holistic Schedulability Analysis

**Objective**

The objective of this step is to obtain a set $F$ containing all feasible configurations the system can assume, defined by:

$$F = \{c \mid c \in C, \exists w \in W, \text{feasible}(c, w) = 1\}$$

Where $C$ is a set containing all possible configurations (feasible or not), $W$ is a set containing all possible workload inputs for the system, and the binary function $\text{feasible}(c, w)$
returns 1 if exists a non-empty workload set where configuration $c$ is handle within deadlines or returns 0 otherwise. The $\text{feasible}(c, w)$ function is based on holistic scheduling analysis.

**Description**
If we recover the example of Fig. 2, and focus on the left configuration we can intuitively understand that the workload set supported by the left configuration is certainly smaller than the one in the right configuration. There is a workload set which can be handled by both configurations, but since our objective is to minimize processor utilization we can compare the solutions under the equal workload sets and sort the configurations according to the resources used.

Due to this feasibility bounded by a workload set, assessing the feasibility of a configuration can be a very challenging problem that is outside of traditional holistic scheduling analysis. The traditional approach does the feasibility analysis considering the WCET which is directly related to the most demanding workload from the entire workload set. However, in our case we are analysing the feasibility of a configuration within a subset of $W$, since our system aims for setting the optimal configuration for each individual workload.

### 4.4 Adaptation Graph

**Objective**
The objective in this step is to obtain an adaptation graph $G = (S, A)$, where the vertices belong to the set $S$ and the edges to the set $A$. The edge set is defined as such:

$$A = \{(c_1, c_2) \mid c_1 \in F, c_2 \in F, c_1 \neq c_2, \exists w \in W, \text{adaptation}(c_1, c_2, w) = a, a \in \mathbb{N}_0^+ \}$$

Were $F$ is the feasible set defined in section 4.3, $W$ is the set of all possible workloads as defined in section 4.3, the $\text{adaptation}(c_1, c_2, w)$ functions tests if it is feasible to support an adaptation time unit cost of $a$ and still meet the deadlines or return $-1$ if it is not feasible for any workload subset. The set $S$ is defined by the vertex cover of $A$.

**Description**
This step is perhaps the most critical step in this methodology since the output we obtain is an adaptation graph, a structure that bounds the system’s feasible adaptation capabilities.

Since we aim for ARTS, there is no guarantee that one feasible configuration can be transformed through an adaptive process into another feasible configuration. Therefore in this step we must check the adaptation between each configuration is feasible or not.

The function $\text{adaptation}(c_1, c_2, w)$ is very dependent on the application context because it must consider the WCET of the instruction set that changes the configuration from $c_1$ to $c_2$, which of course depends on the situation. The cost associated is a natural number, in time units, and we consider also the possibility to be zero although since this way we can model several existent soft ADRTS systems, and this is also sometimes convenient in simulation.
4.5 Transformation and Optimization

Objective
After obtaining the adaptation graph \( G \) this methodology continues by applying a transformation to the graph such that the new graph \( \hat{G} \) serve as working graph for the optimization algorithm.

Description
In the beginning of this methodology we model our SOA system as an ADRTS, then we computed the adaptation boundaries of a system which needs real-time guarantees, but now we need to transform the adaptation graph into the appropriate model for our problem. For this thesis, we aim to minimize the global process utilization therefore our new graph needs to convert the adaptation graph vertices (configuration) into global process utilization.

There is still an interesting characteristic in this graph that concerns the workload set, where each vertex and edge is subject to a subset of workload. This means that each configuration and possible adaptations are only feasible within a specific workload subset. Therefore, when performing the transformation and execution of the optimization algorithm these “workload overlays” must be taken into account.

It should be noticed that \( \hat{G} \) may not have the same topology as \( G \) according to the problem. For instance connected and strong connected graphs are probably the most interesting, although depending on the problem non connected graphs may be relevant.

4.6 Verification and Validation

Objective
The final step is to implement our approach to solve an specific problem, verify if our implementation is complaint with the models, and obtain evidence that validates the hypothesis.

Description
The most promising approach for implementing these type of systems is simulation due to its simplicity and fast development. However, in more advance stages of the work we intent to create a prototype and report the implementation techniques used to develop the system.

Apart from the obvious correctness of the implementation, we will focus our verification and validation by collecting system traces. These traces (similar to logs) are important because they describe each state of the system, so we can use those traces to verify if the system is behaving as we defined.

For validating our approach it has to be done in two steps. First we need to check if deadlines are respected using the system traces to prove the time correctness of our approach. Second we need to show evidence that our solution is indeed the optimal by comparing the configuration assumed at every instance with other possible feasible solutions and it must at every comparison be better or equal to the second best alternative.
5. Plan

The plan proposed in this document is design to test the hypothesis defined in this document, and anticipate probable investigation directions we may take in order to apply the work done into the project associated with this thesis. The time windows defined is around two years, meaning that it starts from September 2013 to September 2015.

The rationale behind this plan is to divide this thesis in three distinct phases, and produce small technical reports to document the progress. The three phases are considered as major landmarks for the thesis plan, meaning that if one of them is deferred then it is very likely that we will need to reconsider the next phase objectives. Concerning the technical reports, the strategy is to pressure to document the progress in a formal way such that it can be easily adapted for papers or for the thesis itself.

5.1 Schedule

We present in this section a plan and a schedule for organizing and control our progress. The Gantt map of the plan is available in A. Appendix. Plan Gantt Chart. In the rest of the section we describe each major work unit in detail.

Research

The research task is associated to the research phase previously defined. The main goal here is to cover the area of ARTS and Holistic Scheduling Analysis ensuring we have a deep knowledge on both modelling and methodologies for design ADRTSs using Holistic Scheduling Analysis.

Both reports aim to allow an exhaustive overview of the concepts, and possibly convert to survey paper. Therefore both will be generic, centred on a proposed taxonomy, not focused on any application, and with a perspective of future trends in the area.

The results of this reports are the trigger to start writing the thesis, specifically the background chapter, and part of the related work.

Hard RTS

At this stage of the thesis, we assume there is a clear defined target application which is decided by the project managers. Knowing this application, we may decide to model it or just some part based on our intuition of how challenging might be. Of course, the application main purpose is to contextualize the theoretical work we produce where we provide evidence by recreating the scenario where the application is needed.

Along with the application, the platform and network are also assumed to be decided around this time. Thus, the logical step to do is review our survey report from the previous task and evaluate if the real-time calculus available can be reused to our concrete scenario. This can range from a simpler reutilization of other authors formulas to propose our own formulas to model the real-time scheduling. Network protocols can easily imply to propose new formulation due to its diversity, however depending on the required computation model it is not discarded the possibility to propose scheduling analysis. With this content we can write a report for describing the system modelling.

Another fundamental tool for this work is simulation due to the flexibility and abstraction from actual implementations. We are particularly interested in evaluate if there
are simulators we can use to model our system since half of our work is planned to be around simulation. If none meet the requirements, then we might extend some existing or even create our own. From this work, it is essential for the creditability of the investigation to describe both simulator used, and the modelling of the system using the simulator for the purpose of verification and validation. This also implies to write a report about this matter. With the system modelling and a simulator tool for conducting the investigation we can proceed to implement our approach. Please note that during this implementation, it is likely that the system model and the simulator may be revised if after new findings compromise these tools.

The core theoretical work of this thesis is done in this phase. The general idea is to implement the methodology described in section 4, verify and validate with the simulator. Along this work there is also a need to theoretically prove some properties of each step, which rise from intuition and practice. For example some open questions we expect are:

- Does our approach only work if the adaptation graph is strongly connected?
- If the adaptation graph is connected, what implications have on the maximum possible workload difference between two time units?
- If the graph is not connected and we have to choose a sub-graph, how to ensure the optimal solution is granted?

We expect to perform this work based on small technical reports on how to solve this and other questions that appear. Despite this approach, there is a need to publish our work in the scientific community therefore we expect to publish a paper about our findings.

The sum of technical reports and the paper provide the input for writing a chapter in the thesis document. This chapter is about this work, but more focused in the theoretical aspects than in the concrete implementation. Nevertheless the concrete implementation can be used to provide evidence.

While this work is around hard real-time guarantees it is expectable to evolve for a soft real-time due to the expectable restriction application domain of the former one.

**Soft RTS**

Currently this is the most unpredictable part of the plan, since it depends on both project demand and results from the previous task. For example, if there is a political decision to follow a large scale complex event processor then our approach needs to be more efficient in computing the configurations in run-time; if we intent to pursuit on a hard-real time but investigate deeper the properties of the adaptation graph or even the feasibility analysis and try to explore these properties to find less intensive computations.

For this document, we will assume the soft RTS path, since currently it is the most widely acceptable due to its strategic interest and to the wide application domain of soft RTSs. Since our methodology is performed under the assumptions A5 and A8 which imply that our system is closed and we know all possible system states in design time, this is not the case of soft RTS which can be: (i) open in the sense that the platform may change; (ii) knowledge about the task system may only be known at run-time; (iii) both previous situations combined. Depending on the results of previous hard RTS task, we might use some properties to
successfully have the optimal solution or more probably we will have to trade the optimal solution for a good solution in order to be more efficient (heuristic based approach). This is also required to have a paper on this matter. After completing this work package, we can write a chapter of the thesis on this.

Parallel to this working package, there is a report on implementing these types of systems. Due to the low-level programming and technical workarounds to implement several abstract concepts used in modelling and simulation, it is essential for the success of the next task to learn from already implemented systems.

Prototype Implementation
Since this thesis is in a context of a PhD in Informatics Engineering, it is desirable to not only simulate and mathematically model our systems, but also as engineer to provide evidence by implementing a prototype.

Once again this part is very dependent on what is decided at the time. Two main course of action here: (i) we can implement the “original” hard RTS simulation scenario or (ii) soft RTS simulation scenario. This decision will certainly depend on the time left, so despite a one year time window if a delay in previous tasks may require choosing the faster scenario to implement; otherwise we implement the most useful for the project.

A very specific characteristic of the prototype task is that they are divided in two generations working packages in order to match the project demand. In the first generation, it is expectable that the prototype meets the desired requirements with a very small scenario, possibly with only one application running on the platform. The second generation prototype is considered, and it will be based on the conclusions of the first generation and the project dynamic demand.

An important deliverable in this phase is a technical report on the implementation techniques used to adapt from the conceptual model. This report along with the last generation working package provides the material to write another chapter of the thesis focused in the implementation of ADRTS.

Thesis
As described in the previous sections, the thesis document is developed along the thesis timeline. The main approach is to use most of the work from the technical reports for the thesis.

Some exceptions due to the chapters’ nature are left for the end. The related work chapter is delayed until the start of the implementation in order to consider more recent works. Also, the introduction and conclusion are left to the end. Due to this project management perspective, where tasks and working packages are oriented to deliverables, we opt to give larger time period to the introduction and conclusion chapters in order to represent the time we needed for making the final arrangements, reviews, and corrections to the thesis.
5.2 Risk Analysis

Every PhD thesis has its own risks, and this is not different. We consider that it is important to highlight the most relevant risks of this thesis and provide alternative paths in order to manage the risk associated.

The first major risk is related to real-time calculus. This particular risk is much related to the network and platform select for the project need, in the sense that if the setup decided is already covered in the literature that the risk is low. However, if the setup proposed does not have the proper covered, we must evaluate the situation and take on of the following paths: (i) if platform is not very different from other existing, and may be easily adapted the formulas then we propose the needed real-time calculus for this problem; (ii) if the scenario require deep changes in the real-time calculus, which is out of the scope of this thesis, then we can choose to follow a known scenario from the literature.

The second major challenge is the simulation tool. Although not expected to be too difficult to control the risk, there may be the case where no accepted simulation tool simulating our scenario is available. In that case we are forced to implement our own simulator or extend an existing one.

The third major challenge is the methodology simulation. This risk derives from the two previous ones and the inherently nature of unpredictability of how the results from our approach. This risk is high, but it is a critical task to be done and it is this unpredictability of the results and the properties needed to be successful that establish the core of this thesis.

The fourth major challenge is the soft RTS approach. Since this task can go in multiple paths depending on if we want to pursue a more efficient and hard-RTS approach, or a more soft-RTS based on heuristics. We can consider this part as an extension to the original work, so although important it is not critical to be accomplished from the thesis perspective. Thus, as long as the time spent in this task does not compromise the implementation task, we consider the risk controlled.

The fifth major challenge is the implementation. This is probably the most difficult risk management task. Since in this area the implementation rely on low-level implementation (kernel level, or even some parts of assembly), and some existing implementation are too tailored for their application, the actual implementation can be very tricky. The main risk control assurance is the research group where the author is inserted which has relevant experience in this area.

The sixth major challenge is journal publications. One of the most important risks associated to this work is a journal publication. Albeit not mandatory, it is very desirable for a PhD work. In this community the most relevant publication sites are conferences and symposiums, so it does not share a journal publish culture. Nevertheless, there are good journal in much related areas such as SOA and parallel systems which may provide opportunities for a journal paper publication.
6. Conclusions

Nowadays, the challenges associated to cyber physical based systems face numerous challenges in order to become widely accepted and deployed. Concepts such as smart grids, smart cities and smart homes have some applications that require very specific non-functional requirements. These applications need to be developed assuming a SOA infrastructure, high workload due to messages arriving to the system, and cope with workload characterized by long periods of relatively periodic behaviour, but with sporadic burst of workload. Building a hard RTS that meets the requirements but at the same time consumes only enough resources accordingly to the workload can be a very significant challenge.

In this thesis our aim is to solve the problem of ensuring hard real-time guarantees and minimal resource utilization in system where the workload is generated by periodic and event activities. Our main hypothesis states the existence of a function that maps workload into system configuration subject to minimal resource consumption, under an adaptive distributed approach where the services can be instantiated multiple times and the application load balanced by a set of service instances. Thus, if an algorithm can be devised to methodologically obtain such function, then we can build a framework for hosting service oriented applications where the developers would have guarantee of real-time and minimal resource consumption during design time.

The thesis contributes to the ADRTS community by providing the optimal system configuration set accordingly to the workload. This contribution highlights the need to automatically obtain our RTS’s adaptation graph, where the system can safely transit between configurations. Another contribution is to demonstrate the potential that such structure has, by extracting the optimal state transitions according to some function we intent to optimize. Furthermore, we explore our possible systems’ configuration by parallelizing services which is a very convenient approach for high demand systems.

Concerning the thesis research plan we adjusted the objectives for a 2 year research. We focus on a first phase of consolidate knowledge by research and develop detailed surveys about the related works. On a second phase we will develop our methodology and test using simulation tools. After obtaining evidence that helps validate our hypothesis, we may tune either the scenario or the real-time constraints for continuing the research. On the fourth phase, based on our achievements, we develop a prototype framework that implements the methodology proposed in this thesis. The last phase is used for finishing the thesis document.
References


## B. Appendix. List of conferences

<table>
<thead>
<tr>
<th>Sources</th>
<th>Name</th>
<th>Type</th>
<th>Justification</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISORC</td>
<td>ISORC Object/Component/Service-Oriented Real-Time Distributed Computing</td>
<td>Conference</td>
<td>It is specific for SOA in Real-Time systems</td>
<td></td>
</tr>
<tr>
<td>ETFA</td>
<td>Emerging Technologies and Factory Automation</td>
<td>Symposium</td>
<td>Industrial systems are a common domain for applying SOA and some application require real-time</td>
<td></td>
</tr>
<tr>
<td>ISORCW</td>
<td>ISORCW Object/Component/Service-Oriented Real-Time Distributed Computing Workshop</td>
<td>Workshop</td>
<td>It is specific for SOA in Real-Time systems, probably a good entry level.</td>
<td></td>
</tr>
<tr>
<td>SEUS</td>
<td>SEUS Software Technologies for Future Embedded and Ubiquitous Systems Workshop</td>
<td>Workshop</td>
<td>Our work is tightly related to embedded systems, IoT.</td>
<td></td>
</tr>
<tr>
<td>Industrial Informatics</td>
<td>Industrial Informatics Transactions on Industrial Informatics</td>
<td>Journal</td>
<td>Industrial systems are a common domain for applying SOA and some application require real-time</td>
<td></td>
</tr>
<tr>
<td>Industrial Informatics</td>
<td>Industrial Informatics International Conference on Industrial Informatics</td>
<td>Conference</td>
<td>Industrial systems are a common domain for applying SOA and some application require real-time</td>
<td></td>
</tr>
<tr>
<td>SOSE</td>
<td>Service-Oriented Systems Engineering Conference</td>
<td>Conference</td>
<td>Conference dedicated to service-oriented systems</td>
<td></td>
</tr>
<tr>
<td>ECRTS</td>
<td>Euromicro Conference on Real-Time Systems</td>
<td>Conference</td>
<td>The real-time conference. Contains many works about Distributed RTS, Adaptive RTS.</td>
<td></td>
</tr>
<tr>
<td>RTSS</td>
<td>Real-Time Systems Symposium</td>
<td>Symposium</td>
<td>The real-time symposium</td>
<td></td>
</tr>
<tr>
<td>RTS</td>
<td>International Journal of Time-Critical Computing Systems Journal</td>
<td>Journal</td>
<td>The major journal on real-time systems.</td>
<td>Despite not being present in this document due to the publisher's lack of proper citation extraction of the website, we did not find any relevant paper for this review which wasn't already included in the ECRTS.</td>
</tr>
<tr>
<td>JPDS</td>
<td>Journal of Parallel and Distributed Systems Journal</td>
<td>Distributed and parallel matches SOA and split/join model</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>