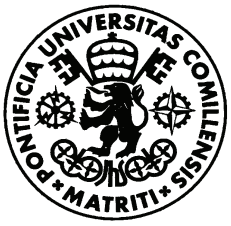

UNIVERSIDAD PONTIFICIA COMILLAS

ESCUELA TECNICA SUPERIOR DE INGENIERIA (ICAI)

INGENIERO ELECTROMECANICO

FINAL DEGREE PROJECT

REAL-TIME SCHEDULING IN SMART BUILDINGS

AUTHOR: Guillermo Gomez Limia

DIRECTOR: Carlo Fischione (KTH Stockholm)

MADRID, June 2014

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN ACCESO *ABIERTO* (*RESTRINGIDO*) DE DOCUMENTACIÓN

1º. Declaración de la autoría y acreditación de la misma.

El autor D. _____ , como _____ de la UNIVERSIDAD PONTIFICIA COMILLAS (COMILLAS), **DECLARA**

que es el titular de los derechos de propiedad intelectual, objeto de la presente cesión, en relación con la obra_____

_____

_[1], que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual como titular único o cotitular de la obra.

En caso de ser cotitular, el autor (firmante) declara asimismo que cuenta con el consentimiento de los restantes titulares para hacer la presente cesión. En caso de previa cesión a terceros de derechos de explotación de la obra, el autor declara que tiene la oportuna autorización de dichos titulares de derechos a los fines de esta cesión o bien que retiene la facultad de ceder estos derechos en la forma prevista en la presente cesión y así lo acredita.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad y hacer posible su utilización de *forma libre y gratuita* (*con las limitaciones que más adelante se detallan*) por todos los usuarios del repositorio y del portal e-ciencia, el autor **CEDE** a la Universidad Pontificia Comillas de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra (a) del apartado siguiente.

3º. Condiciones de la cesión.

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia, el repositorio institucional podrá:

[1] Especificar si es una tesis doctoral, proyecto fin de carrera, proyecto fin de Máster o cualquier otro trabajo que deba ser objeto de evaluación académica

(a) Transformarla para adaptarla a cualquier tecnología susceptible de incorporarla a internet; realizar adaptaciones para hacer posible la utilización de la obra en formatos electrónicos, así como incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato. .

(c) Comunicarla y ponerla a disposición del público a través de un archivo abierto institucional, accesible de modo libre y gratuito a través de internet.[2]

(d) Distribuir copias electrónicas de la obra a los usuarios en un soporte digital. [3]

4º. Derechos del autor.

El autor, en tanto que titular de una obra que cede con carácter no exclusivo a la Universidad por medio de su registro en el Repositorio Institucional tiene derecho a:

a) A que la Universidad identifique claramente su nombre como el autor o propietario de los derechos del documento.

b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

c) Solicitar la retirada de la obra del repositorio por causa justificada. A tal fin deberá ponerse en contacto con el vicerrector/a de investigación (curiarte@rec.upcomillas.es).

d) Autorizar expresamente a COMILLAS para, en su caso, realizar los trámites necesarios para la obtención del ISBN.

[2] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría redactado en los siguientes términos:

(c) Comunicarla y ponerla a disposición del público a través de un archivo institucional, accesible de modo restringido, en los términos previstos en el Reglamento del Repositorio Institucional

[3] En el supuesto de que el autor opte por el acceso restringido, este apartado quedaría eliminado.

d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

a) Deberes del repositorio Institucional:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.

- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusiva del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.

- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.

b) Derechos que se reserva el Repositorio institucional respecto de las obras en él registradas:

- retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a …….. de ……... de …….

ACEPTA

Fdo……

Autorizada la entrega del proyecto del alumno:

Guillermo Gomez Limia

EL DIRECTOR DEL PROYECTO

Carlo Fischione (KTH Stockholm)

Fdo.: ..... Fecha: ..... / ..... / .....

V B DEL COORDINADOR DE PROYECTOS

Fernando de Cuadra Garcia

Fdo.: ..... Fecha: ..... / ..... / .....

Abstract

Scheduling of the information packets sent by sensors to a control unit within a smart building is of great importance to guarantee a correct performance of the control system.
In this report the applications which could benefit from a wireless sensor control network inside a smart building are first identified and then qualitatively classified. The real-time scheduling theory is presented and a simulation comparing different scheduling algorithms is implemented. These scheduling algorithms are compared in order to observe and quantify their performance when a loss of information due to constraints in the communication process happen.
Finally, a scheduling algorithm prioritizing a combination of several attributes of the information packets sent from each sensor is proposed and implemented allowing the user to discriminate the information losses.

Objectives

The aim of this project is to analyze and explore how the real-time scheduling process, coordinating the information packets sent by the sensors inside a smart building, is carried out and to simulate this process.
More specifically three main themes are individually and more extensively treated:

- Control applications inside the smart buildings: In this section several applications that can benefit from a wireless sensor network are identified. The most significant attributes for individual sensors are presented: Period of control, deadlines in latency, packet size and several communication protocols are as well described.

- Real-time scheduling: The theory on real-time scheduling is presented attending to its definition, its basic parameters and the most common scheduling algorithms are exposed.

- Simulation: Four different scheduling algorithms were implemented in MATLAB to analyze and understand how a scheduling buffer works and see their effect on the packet loss throughout the scheduling process.

Method and Results

Firstly, the applications inside the building were identified and classified attending to the attributes considered. This classification can be seen in Table 1 and some specific examples can be observed as well in Table 2.

Table 1. Smart building applications that can benefit from a wireless sensor network

Category	Period	Deadline	Packet size	Protocol
Safety	Intensive	Hard	S/M	IEEE 802.15.4
Security	Intensive	Hard	M/L	ZigBee
Comfort	Moderate	Firm	M/L	ZigBee
Metering	Relaxed	Soft	S/M	WirelessHart
Monitoring	Sporadic	Soft	S/M	ISA SP-100

Table 2. Examples of specific sensors for each application

Category	Examples
Safety	Shower accident sensor, Co2 detector for garage, Smoke detector.
Security	Glass break sensor, Presence sensor, Alarm sensor Automatic lock for doors and windows sensor.
Comfort	Window shades, HVAC, Central heating, Door opening, Irrigation controller.
Metering	Light sensor, Temperature sensor, Underground humidity sensor (backyard).
Monitoring	Electrical properties (Power flow, voltage,etc.), Water quality conditions of the supply to the building.

Secondly, the real-time scheduling section provides the necessary theory to understand how this type of systems work providing the basic knowledge to be able to implement the simulation. Finally, the simulation is implemented in MATLAB by using the list of sensors obtained from the first section and also by using the theory and the scheduling algorithms treated in the second section.
The following four different scheduling algorithms were implemented:

- FIFO (First In First Out): The information packets are not ordered in the buffer in any specific manner. With this algorithm the bandwidth of the buffer is changed to observe the effect on the packet loss. If the bandwidth is lowered, to 220kb/s as in Figure 1, the amount of unsent information in the buffer builds up second after second leading to a failure of the system.

- RM (Rate Monotonic): The information packets are sorted out with decreasing period of control times. Two criteria to define when packet losses occur are implemented:

  1. Deadline loss: This packet loss is associated with the deadline expiry time from which a packet ends being of use to the control unit and is then eliminated.
  2. Bandwidth+Buffer loss: By implementing this loss, a limitation on the amounts of information that can be stored in the buffer (after the bandwidth has been sent) and be forwarded to the following time step is achieved.

- EDF (Earliest Deadline First): With this algorithm it is possible to prioritize the packets with diminishing deadline expiry time. In this case it is possible to observe that the deadline loss disappears as was expected when using this algorithm.

- HPF (High Priority First): The HPF algorithm allows the user to manually set a parameter to decide the order in which the prioritization will be made. To do this, the following algorithm was implemented:

$$P = \alpha * K_1 * \left(\frac{ps}{K_2} + pr\right) + (1 - \alpha) * (dl), \alpha \in [0, 1] \quad (1)$$

where the parameter α can be manually adjusted to decide in which percentage the scheduling priority function P depends on the packet size ps and the period of control pr and in which proportion it depends in the absolute deadline dl. $K_1 = 1.125$ and $K_2 = 5kb/s$ are two constants designed to keep a proper dimensioning of the addend.
By changing the value of α the user can set the weight on the part of the equation desired. For instance if $\alpha = 1$ the individual packet loss is reduced as we prefer to let out the bigger and less frequent packets as in Figure 2.

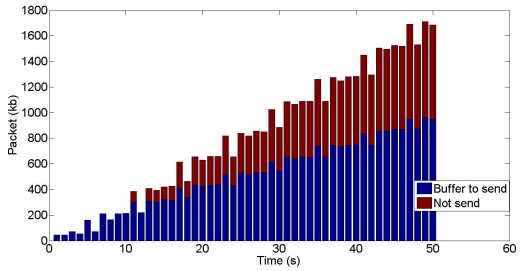

Figure 1. FIFO as a result of lowering the bandwidth is unable to send all the packets at each time step.

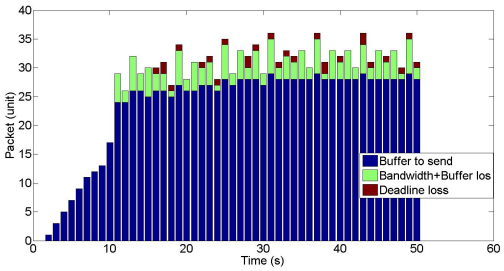

Figure 2. HPF scheduling with $\alpha = 1$ minimized packet loss since the bigger packets are the ones preferred to lose.

Conclusions

Successful scheduling of all the information coming from the sensors deployed inside a smart building is essential to be able to perform and control in a proper way all the applications installed in order to coordinate security or safety functions and to make the stay inside the building more comfortable.
This project classifies different applications in the smart building that can benefit from a wireless sensor network control and shows that by using different scheduling algorithms the information packets sent by these sensors can be prioritized by a chosen specific attribute. This is important in case limitations in the scheduling buffer's size or in the bandwidth of the communications channels could imply loss of information.
In further studies the implementation of scheduling algorithms which can be controlled by an external user could be further developed attending to many other characteristics of the packets sent by different sensors. Nevertheless, a very interesting field of study would consist in testing if these control parameters could be controlled dynamically and changed each time step of the simulation based on continuous feedback to reduce the overall packet loss.

Resumen

La planificación de los paquetes de información enviados por los sensores a una unidad de control en el interior de un edificio inteligente tiene una gran importancia para poder garantizar una correcta actuación del sistema de control.
En este artículo las aplicaciones dentro de un edificio inteligente las cuales podrían beneficiarse del uso de una red sensorial inalámbrica de control son primeramente identificadas y posteriormente son cualitativamente clasificadas. La teoría de planificación en tiempo real es presentada y una simulación comparando distintos algoritmos de planificación es implementada. Estos algoritmos de planificación son comparados entre sí para observar y cuantificar su actuación cuando se produce una perdida de información debida a restricciones en el proceso de comunicación.
Finalmente, un algoritmo de planificación, el cuál prioriza una combinación de varios atributos de los paquetes de información enviados desde cada sensor, es propuesto e implementado permitiendo al usuario discriminar la pérdida de información.

Objetivos

El objetivo de este proyecto es analizar y explorar como un proceso de planificación en tiempo real, coordinando los paquetes de información enviados por los sensores en un edificio inteligente, es llevado a cabo y simular el mismo.
Específicamente tres temas son tratados individualmente y de forma más extensiva:

- Aplicaciones de control en el edificio inteligente: En esta sección varias aplicaciones que pueden verse beneficiadas por una red sensorial inalámbrica son identificadas. Los atributos mas significativos para cada sensor son presentados: Periodo de control, fecha de expiración, tamaño del paquete y varios protocolos de comunicación son asimismo descritos.
- Planificación en tiempo real: La teoría acerca de la planificación en tiempo real es presentada haciendo hincapié en su definición, los parámetros básicos y en los algoritmos de planificación.
- Simulación: Cuatro algoritmos de planificación diferentes fueron implementados en MATLAB para analizar y comprender el funcionamiento de un buffer de planificación y ver su efecto en la pérdida de paquetes a lo largo del proceso de planificación.

Método y resultados

En primer lugar, las aplicaciones del interior de un edificio fueron identificadas y clasificadas atendiendo a los atributos considerados. Esta clasificación puede verse en el Cuadro 1 y algunos ejemplos específicos pueden verse en el Cuadro 2.

Cuadro 1. Aplicaciones del edificio inteligente que puede ser automatizadas con una red sensorial inalámbrica

Category	**Period**	**Deadline**	**Packet size**	**Protocol**
Safety	Intensive	Hard	S/M	IEEE 802.15.4
Security	Intensive	Hard	M/L	ZigBee
Comfort	Moderate	Firm	M/L	ZigBee
Metering	Relaxed	Soft	S/M	WirelessHart
Monitoring	Sporadic	Soft	S/M	ISA SP-100

Cuadro 2. Ejemplos de sensores específicos para cada aplicación

Category	**Examples**
Safety	Shower accident sensor, Co2 detector for garage, Smoke detector.
Security	Glass break sensor, Presence sensor, Alarm sensor Automatic lock for doors and windows sensor.
Comfort	Window shades, HVAC, Central heating, Door opening, Irrigation controller.
Metering	Light sensor, Temperature sensor, Underground humidity sensor (backyard).
Monitoring	Electrical properties (Power flow, voltage,etc.), Water quality conditions of the supply to the building.

En segundo lugar, la sección acerca de la planificación en tiempo real proporciona la teoría necesaria para comprender como este tipo de sistemas funcionan, proporcionando el conocimiento necesario para realizar posteriormente la simulación.
Finalmente, la simulación ha sido implementada en MATLAB mediante la lista de sensores obtenidas en la primera sección y también haciendo uso de la teoría y los algoritmos de planificación tratados en la segunda sección.
Los siguientes cuatro algoritmos de planificación fueron implementados:

- FIFO (First In First Out): Los paquetes de información no se ordenan en el buffer de ninguna manera específica. Con este algoritmo se realizan cambios en el ancho de banda del buffer para observar el efecto que esto tiene en la pérdida de paquetes. Si bajamos el tamaño del ancho de banda, a 220 kb/s como en la Figura 1, la cantidad de información que no ha sido enviada se amontona en el buffer segundo tras segundo llevando finalmente a producirse un fallo del sistema.

- RM (Rate Monotonic): Los paquetes de información son ordenados con periodos de control decrecientes. Dos criterios son implementados para definir cuando ocurren las pérdidas de paquetes:

    1. Deadline loss: Esta pérdida de paquetes está asociada con el tiempo de caducidad tras el cuál un paquete deja de tener utilidad para la unidad de control y es por tanto eliminado.
    2. Bandwidth+Buffer loss: Implementando esta pérdida, se consigue poner una limitación en la cantidad de información que puede ser almacenada en el buffer (después de que el ancho de banda permitido haya sido enviado) y pueda seguir disponible en el siguiente escalón temporal.

- EDF (Earliest Deadline First): Con este algoritmo es posible priorizar los paquetes con menor tiempo de expiración. En este caso es posible observar que la pérdida por caducidad desaparece como era de esperar.

- HPF (High Priority First): El algoritmo HPF permite al usuario cambiar el valor de un parámetro para decidir el orden en que la prioridad será establecida. Para ello, el siguiente algoritmo fue implementado:

$$P = \alpha * K_1 * \left(\frac{ps}{K_2} + pr\right) + (1-\alpha) * (dl), \alpha \in [0,1] \quad (1)$$

donde el parámetro α puede se ajustado para decidir el porcentaje en que la función de planificación de la prioridad P depende de el tamaño del paquete ps y en su periodo de control pr y en que proporción depende del tiempo de expiración absoluto dl. $K_1 = 1{,}125$ y $K_2 = 5kb/s$ son dos contantes elegidas para mantener una dimensión correcta de los sumandos.
Cambiado el valor de α el usuario puede asignar el peso en la parte de la ecuación deseada. Por ejemplo si $\alpha = 1$ la pérdida individual de paquetes es reducida debido a que preferimos dejar fuera los paquetes más grandes y menos frecuentes como en la Figura 2.

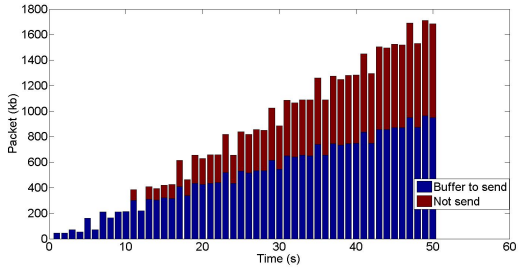

Figura 1. Como resultado de bajar el tamaño del ancho de banda el algoritmo FIFO es incapaz de enviar todos los paquetes en cada escalon temporal.

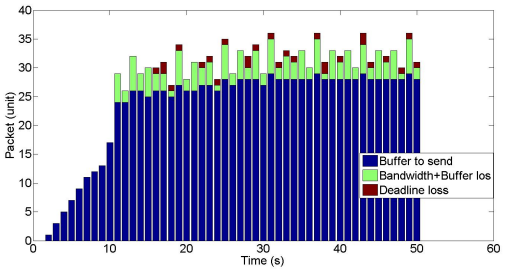

Figura 2. La planificación HPF con $\alpha = 1$ minimiza la pérdida de paquetes ya que se prefiere perder los paquetes de mayor tamaño.

Conclusión

La exitosa planificación de toda la información proveniente de los sensores implantados en el interior de un edificio inteligente es esencial para ser capaz de controlar de un modo correcto todas las aplicaciones instaladas y de este modo coordinar las funciones de seguridad o monitorización y hacer la estancia en el interior del edificio mas confortable.
Este proyecto clasifica las distintas aplicaciones en el edificio inteligente que pueden verse beneficiadas de una red de control inalámbrica y demuestra que usando distintos algoritmos de planificación los paquetes de información enviados por estos sensores pueden ser ordenados atendiendo un atributo específico u otro. Esto tiene importancia en el caso de que existan limitaciones en el tamaño del buffer de planificación o en el ancho de banda de los canales de comunicaciones que puedan provocar la pérdida de información.
En estudios posteriores la implementación de algoritmos de planificación que puedan ser controlados externamente por un usuario podría ser desarrollado para tener en cuenta otras características de los paquetes enviados por diversos sensores. Sin embargo, un campo de estudio muy interesante consistiría en probar si estos parámetros de control pudieran ser controlados

de forma dinámica y ser cambiados en cada escalón temporal de la simulación basandose en retroalimentación continua para reducir el número total de pérdida de paquetes.

Acknowledgement

I would like to thank KTH University of technology in Stockholm for allowing me to develop my bachelor thesis project during the second half of the academic course 2013/2014.
More specifically, I would like to thank my supervisor Carlo Fischione for providing me with feedback and teacher assistant Yuzhe Xu for keeping track of my progress, giving me useful feedback and code developing advice throughout my thesis work.

DOCUMENTO I

MEMORIA

Index

List of Figures

Table Index

Acronyms

KTH	Kungliga Tekniska Hgskolan
FCFS	First Come First Served
FIFO	First In First Out
RM	Rate Monotonic
EDF	Earliest Deadline First
EDD	Earliest Due Date
LLF	Least Laxity First
HPF	Highest Priority First

Simbols

a_i	Arrival time
C_i	Computation time
d_i	Absolute deadline
D_i	Relative deadline
s_i	Start time
f_i	Finishing time
R_i	Response time
v_i	Value
L_i	Lateness
E_i	Tardiness
X_i	Laxity
P	Scheduling priority function
α	Manually adjustable prioritizing parameter
ps	Packet size
pr	Period of control
dl	Absolute deadline
K_1	Dimensioning constant 1
K_2	Dimensioning constant 2

PART I

PROJECT DESCRIPTION

Chapter 1

Introduction

ENGINEERING has always focused on trying to make "life easier", or at least more comfortable, for humankind.
People spend a lot of time inside buildings at home, working place, etc. This is why implementing sensor networks that will allow for control, actuation and communication attending peoples needs: comfort, security, metering and other aspects inside buildings would result very useful and satisfactory.
Small devices, such as wireless sensors and actuator networks, can be distributed all over the building in order to [4]:

- Monitor physical variables, such as temperature, light intensity, humidity, etc.
- Control by means of a central controller automatically or physically by a person: air conditioning, lights, door opening systems and other electronic appliances.

These networked control systems are characterized for being wireless and for being powered by batteries. These characteristics makes their deployment in the building fast, easy and relatively inexpensive.
It can be stated that these sensors are dynamic systems as they depend on both time and input data. To make the system work properly the different tasks should be scheduled attending to several criteria. Depending on the varying sampling rate and the size of the data a flexible schedule will be designed by implementing a control algorithm. For example, Earlies Deadline First (EDF), Least Laxity First (LLF), Highest Priority First (HPF), etc. [8].
Some important aspects to take into consideration are that these networked control systems suffer from: delay, as it takes some time for the information to reach the controller which can be unacceptable in some cases, and packet losses, which consist of a loss of information mainly produced by interference with other networks. For this reason and also because our power supply is usually limited by a battery, the sensors must work in an efficient way and transmit information in the proper and exact amounts [10].
In this report the packet loss of a group of sensors when using different scheduling algorithms is analyzed. One algorithm is proposed and formulated that allows the user to adjust the priority when scheduling the tasks attending to several criteria in order to minimize and control the overall packet loss.
This project will cover the different fields of application of sensor networks which can be deployed inside a smart building attending their basic properties as they appear in Section II. Later on, the theory of real-time scheduling, its basic parameters and algorithms are exposed in section III. In section IV the loss of packets is empirically investigated by means of implementing and simulating several scheduling algorithms. Finally, an insight on future work plausible in the

smart building is exposed in section VI.

Chapter 2

Control applications inside the Smart Buildings

In order to correctly monitor and control some applications inside a building, such as air conditioning, door opening, light regulation, etc, it is necessary to identify the frequency with which these applications should be controlled, the deadlines in the latency with which the information must reach the controller and the size of the data packages that should be transmitted. In this section each of these aspects will be treated to be capable at the end of attributing each application the most suitable classification in TABLE I. Some examples of several of these applications are presented in TABLE II.

2.1. Control period

Some applications need to be checked and controlled more often than others. In order to assign the proper time frame by which each group of applications should be checked; it should be considered how, and in what proportion, this time affects the: comfort, security and the rest of the different fields of application to the people inside the smart building at each time.
For example it can be deduced that a door must open when a person is intending to trespass it or that the temperature inside the building must change and be adjusted continuously to keep an acceptable comfort level for the people inside it.
These decisions will be made by inspection, taking a ?pessimistic? approach (worst-case scenario). The time stated for each sensor to transmit data will be done periodically and it will be classified for the different applications, as can be seen in TABLE I, attending to four levels:

- Intensive: This level covers the applications that need to be controlled in time frames of a couple of seconds up to 10 seconds.
- Moderate: For sensors which should transmit information every 10 seconds and up to 2-3 minutes.
- Relaxed: For the applications that need from a couple of minutes up to 15 minutes to periodically send data.
- Sporadic: This level is to be applied to those activation times above 15 min that can go up to a couple of hours.

It is important to note that this control period may vary when looking at different specific sensors. This is because each individual sensor should be looked at independently and can have particular constraints depending on its specific application.

2.2. Latency in deadlines

The information sent by the sensors must reach the control unit in a specific amount of time in order to serve properly the duty they were designed for.
The applications should be classified attending the consequences that would derive if their information arrived later than it is specified. This can sometimes have bad side-effects and in other occasions these consequences are of minor importance. To attend this issue a further classification for the proposed sensors will be considered as can be seen in TABLE I.
The deadlines can be classified as:

- Hard: When delivering the objective information after the deadline may have catastrophic impacts. For these applications around 1-5% of the control period time will be established as the maximum latency for each deadline.
- Firm: When delivering the objective information after the deadline is useless but does not cause damage to the system, to its surrounding or to the people. For these applications around 5-15% of the control period time will be established as the maximum latency for each deadline.
- Soft:When delivering the objective information after the deadline has still some utility, causing on the long or the short run performance degradation. For these applications around 15-30% of the control period time will be established as the maximum latency for each deadline.

2.3. Size of the packets

Depending on the type of sensor and on its field of applications the data that each sensor sends will have varying size.
Each packet sent occupies a determined memory allocation which is measured in kb. This information contains among others a header which identifies its source node, destination node, the length of the data field, and other data. In addition to these contents, sometimes the nodes can transmit as well special frames to report and identify fault conditions [5].
To classify the packet size estimated for the different applications, the following distinction can be made:

- Small packets$\langle$S$\rangle$: Packets with sizes below 5kb.
- Medium packets $\langle$M$\rangle$: Packets with sizes from 5-10kb.
- Large packets$\langle$L$\rangle$: Packets with sizes larger than 10kb.

To assign the proper packet size to each category in TABLE I, the general complexity and content size of the messages for each application was estimated and taken into account.

2.4. Wireless communication protocols

In order to successfully implement a controlled wireless sensor network in a smart building, each group of sensors should be assigned the right sensor network technology. This will be done by taking a look to different protocols and deciding which one suits better for each of the different applications.

Table 1. Smart building applications that can benefit from a wireless sensor network

Category	Period	Deadline	Packet size	Protocol
Safety	Intensive	Hard	S/M	IEEE 802.15.4
Security	Intensive	Hard	M/L	ZigBee
Comfort	Moderate	Firm	M/L	ZigBee
Metering	Relaxed	Soft	S/M	WirelessHart
Monitoring	Sporadic	Soft	S/M	ISA SP-100

Table 2. Examples of specific sensors for each application

Category	Examples
Safety	Shower accident sensor, Co2 detector for garage, Smoke detector.
Security	Glass break sensor, Presence sensor, Alarm sensor Automatic lock for doors and windows sensor.
Comfort	Window shades, HVAC, Central heating, Door opening, Irrigation controller.
Metering	Light sensor, Temperature sensor, Underground humidity sensor (backyard).
Monitoring	Electrical properties (Power flow, voltage,etc.), Water quality conditions of the supply to the building.

2.4.1. IEEE 802.15.4

This protocol should be used for low-throughput, low-cost wireless communications links such as home automation, security, gaming, etc.
Some of its most important characteristics are that IEEE 802.15.4 can be used for distances up to 10-20 meters, its latency comes down to 15 milliseconds and its raw data rates can go up to: 20 kb/s for the case of 868 MHz, 40 kb/s for 915 MHz and 250 kb/s for 2.4 GHz.
Some interesting features are that the IEEE 802.12.4, supports both star and peer-to-peer network topologies and also that due to its use of low transmit power and very low duty cycle operation excellent battery life is achieved [3].

2.4.2. WirelessHart

This network layer supports self-organizing and self-healing mesh networking topologies; this allows that messages can be routed around interferences and obstacles.
The WirelessHart protocol operates in a 2400 MHz radio band with a data rate up to 250 kbits/s. WirelessHart has very stringent requirements on each network device allowing a 10 ms time slot, be further split into several time intervals (each of range 100 micros- 4.5 ms). The topology of a WirelessHart network can be a star a cluster or a mesh. This provides a much better scalability for different applications.
WirelessHart also distinguishes itself from other public standards by maintaining a central network manager. The network manager is responsible for maintaining up-to-date routes and communication schedules for the network, thus guarantee the network performance.
In the WirelessHart networks, sensors are usually attached to field devices to collect specific environmental data, such as flow speeds, fluid levels, or temperatures [11].

2.4.3. ZigBee

ZigBee is a wireless networking technology developed for low-data-rate and short-range applications. So it can also be useful for the proposed smart building control sensor network.
ZigBee can work in the 868/915/2400 MHz radio frequency bands, providing a bit rate of 20/40/250 kb/s respectively. It can provide service for up to 10-100 m apart and carry message with a size up to 127 bytes.
ZigBee can support mesh routing, tree routing and source routing and the main application areas to be considered in a house environment should be for the areas of light control, HVAC, window shades and security [4].

2.4.4. ISA SP-100

The ISA-SP100.11a standard is intended to be used for noncritical applications that can withstand delays of a maximum of 100 ms. It is conceived to serve lower implementation complexity devices and can provide low rates up to 250 kb/s.
This protocol may adopt a frequency hopping scheme in several frequency channels, which can be useful to avoid interferences with other signals. The ISA-SP100.11a can have a mesh topology built between different devices and it can as well interconnect several meshes [12].

Chapter 3

Real-time scheduling

In real-time systems, the correct performance of the system depends on the results obtained after a computation process as well as on the time that it takes for these results to be completed. In these type of systems obtaining results on time is as important as the actual result itself. Hence, it can stated that a system failure occurs when the wrong results are obtained, but also when the timing constraints are not fulfilled [7].
More specifically, the basic properties that define and make real-time systems different from other control systems are [2]:

- Timeliness: As mentioned before, in real-time systems it is as important that the result has its corresponding correct value, as that this result is obtained within a specified time domain.
- Predictability: When taking scheduling decisions it is of great importance to predict the possible consequences and plan alternative actions if needed.
- Efficiency: The characteristics of the system have constrains in terms of space, weight, energy, memory, computational power, etc.
- Robustness: Real-time systems should no collapse when they are subjected to peak-load conditions and they should be designed to be able to carry out a safe and controlled overload management.
- Fault tolerance: Single hardware and software failures should be handled by the system, which should be resistant and stand without crashing, being able to carry on working.
- Maintainability: Real-time systems typically have modular structure in order to ensure and ease possible system modifications.

Real-time systems can be classified in 3 different groups attending to the consequences derived from not fulfilling their individual time requirements. These groups are [2]:

- Hard: When producing results after their specific deadline may cause catastrophic consequences.
- Firm: When producing results after their specific deadline is useless but does not cause damage.
- Soft: When producing results after their specific deadline has still some utility, causing in the long run performance degradation.

3.1. Scheduling decisions

In real-time systems the information traffic is modeled as a stream of packets, with each packet having an expiry time beyond which the information is of no use to the end user. The objective of the scheduler is to transmit each packet before its expiry. When this is not possible, the scheduler should try then to minimize the number of lost packets due to deadline expiry.
The scheduler of a real-time system, as can be seen in Fig. 1, should focus on the admission control, allowing a certain number of streams to enter the network, as well as scheduling the packets, allocating the bandwidth for them and making sure that some specified quality of service constraints are successfully met [9].
In the activation phase of the tasks, two different groups can be identified according to their process structure. These tasks can be separated into tasks with a periodic structure or tasks with an aperiodic structure.
The periodic processes are characterized for being executed on a regular basis. These processes are defined by their period, the time specification within which these tasks must be executed repeatedly and their required execution time per period which is usually given attending an average measurement and a worst case execution time and it determines the time it takes for the packet to be dispatched after it has exited the scheduler.
In contrast, the aperiodic or non-periodic processes are those which are activated randomly following an action independent from the considered system, for example a Poisson or a Normal distribution. In a vast number of cases these processes deal with critical events occurring in the system's environment. Thus, it is usually not possible to perform worst-case analysis successfully and, as a result, it can be stated that aperiodic processes cannot have hard deadlines [1].

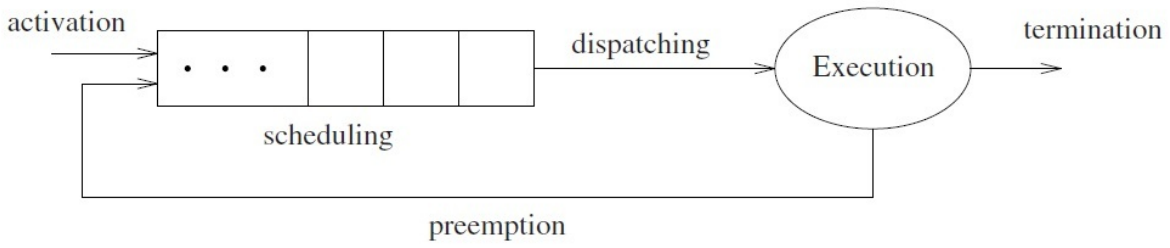

Figure 1. Scheduling process to lead tasks towards execution.

3.2. Real-time parameters

To understand how the scheduling of real-time systems is carried out it is necessary to define some parameters, represented in Fig.2, which will allow for the correct and successful performance of the system. The decision of the values for these parameters will impose the constraints that will determine the working conditions for each specific system.

Some of these task parameters are [2]:

- Arrival time $\langle a_i \rangle$: Is the time at which a task gets ready for execution, it is also known as release time or request time.
- Computation time $\langle C_i \rangle$: Is the time necessary for the processor for executing the task with no interruption.

- Absolute deadline $\langle d_i \rangle$: Is the time when the task should be completed to avoid causing damage to the system.
- Relative deadline $\langle D_i \rangle$: Is the difference between the absolute deadline and the request time. $D_i = d_i - a_i$.
- Start time $\langle s_i \rangle$: Is the time when the task starts its execution.
- Finishing time $\langle f_i \rangle$: Is the time when the task finishes its execution.
- Response time $\langle R_i \rangle$: Is the difference between the finishing time and the request time. $R_i = f_i - a_i$.
- Criticality: Is a parameter that states the consequences for missing a deadline(Hard, firm, soft).
- Value $\langle v_i \rangle$: Is the relative importance of a task with respect to the group of the other tasks inside a certain system.
- Lateness $\langle L_i \rangle$: Is the delay of a specific task completion with respect to its deadline. $L_i = f_i - d_i$.
- Tardiness $\langle E_i \rangle$: Is the time a task stays active after its deadline. $E_i = max(0, L_i)$.
- Laxity $\langle X_i \rangle$: Is the maximum time a task can be delayed on its activation to be completed within its deadline. $X_i = d_i - a_i - C_i$.

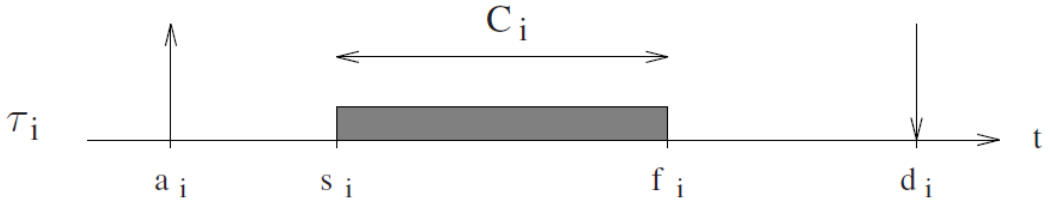

Figure 2. Structure and parameters that determine a real-time task.

3.3. Real-time algorithms

In order to perform a successful scheduling of the tasks the proper and correct algorithm should be implemented. This algorithm will be responsible for setting the tasks, processors and type of resources as well as choosing the order and moment in which each individual task will be initialized.
The scheduling algorithms can be classified attending different aspects [2]:

- Preemptive vs. Non-preemptive: For preemptive algorithms, tasks can be interrupted at any time to assign the processor another active task, while for non-preemptive algorithms the tasks which are started are executed until completion.

- Static vs. Dynamic: For static algorithms, scheduling decisions are based on fixed parameters while for dynamic these are dynamical parameters that may change during the evolution of the system.
- Off-line vs. Online: For the off-line algorithms, they are executed on the entire task set before tasks activation, while for the online, the scheduling decisions are taken at runtime every time a new task enters or is terminated in the system.
- Optimal vs. Heuristic: The optimal algorithms try to minimize some cost function defined over the tasks set and their main objective is to be optimal and achieve a feasible schedule. The heuristic approach takes the scheduling decisions following an heuristic function and it tends towards the optimal schedule but there are no guarantees that this will be eventually achieved.

There are many different scheduling algorithms depending on the strategy or approach they use to schedule the information and select the position for each packet in the sending buffer. Some of the most used scheduling algorithms are [8]:

- First Come First Served (FCFS): This algorithm is also known as First In First Out (FIFO). It sorts out the packets as they reach the buffer from oldest to newest.
- Rate Monotonic (RM): This algorithm assigns static priorities to tasks depending on their period. Moreover this algorithm assigns higher priorities to the tasks with the smallest periods.
- Earliest Deadline First (EDF): This algorithm focuses on minimizing the maximum latency by prioritizing the execution of the tasks with the smallest absolute deadline.
- Earliest Due Date (EDD): This algorithm focuses on minimizing the maximum latency by executing the tasks in the order of nondecreasing deadlines.
- Least Laxity First (LLF): This scheduling algorithm assigns the priority to the tasks according to their executing urgency. The smaller the laxity value of a task is, the higher priority it will be assigned for execution.
- Highest Priority First (HPF): For this algorithm a priority parameter is given or is implemented manually, to sort the tasks in a certain determined configuration.

The ideal "Clairvoyant" algorithm is the one that knows the future as it knows in advance the arrival times of each individual task. To evaluate the quality of the different algorithms listed before, one should compare them with the "Clairvoyant" algorithm for each specific application. For very strict applications the feasibility of the algorithm should be guaranteed in advance, looking ahead in the future and assuming a worst-case scenario.
The algorithms for real-time systems are usually "open loop" algorithms. These algorithms create schedules which are executed from beginning to completion, without being adjusted at any time by means of their feedback. The open loop algorithms are usually designed following a worst-case scenario analysis which can be vague when confronting highly unpredictable environments. For these cases it can be interesting to consider using an adaptive real-time algorithm which can handle these effects dynamically as described more detailed in [6].

Chapter 4

Simulation

WHEN implementing a scheduling algorithm, decisions are made concerning the order in which the set of task will be executed. The impossibility of executing all these tasks will be translated into a loss of data and more precisely a loss of individual packets.
In the conducted simulation the loss of packets when using different scheduling algorithms is studied and analyzed. In particular four scheduling algorithms mentioned previously in section III where implemented: The First In First Out (FIFO), the Rate Monotonic (RM), the Earliest Deadline First (EDF) and the Highest Priority First (HPF).

4.1. Setup and assumptions

The simulation was implemented in MATLAB and consists of a list containing the information for each sensor and a script code in which each second is modeled as a loop and treated independently. Thus in each second the proper information is initialized, loaded in a buffer, scheduled following an algorithm and sent.
To perform the simulation some assumptions and decisions were made in order to obtain a reduced example on how the scheduling is carried out in a smart building.
Firstly, the simulation time frame considered goes from 1s to 50s. To obtain reasonable results within this time frame the values for the start time, the deadline and the period of control were given smaller values than those stated in section II.
Secondly, a list containing the information for each sensor: start time, packet size, deadline and period of control was implemented. This information is computed randomly within specific ranges for each characteristic in order to obtain a wide combination among the 4 aspects taken into account for the list of sensors. The ranges for each characteristic are obtained randomly between:

- Start time: From 1 through 10 seconds.
- Packet size: 5, 10, 20 kb.
- Deadline: From 1 through 10 seconds.
- Period of control: 1, 2, 3 seconds.

To obtain feasible simulation times and manageable information sizes, the experiment was conducted with 30 different sensors.
Thirdly, the ZigBee protocol is implemented to limit the bandwidth up to 250 kb/s. Afterwards, this parameter is changed to study and quantify the packet loss attending to different scheduling

situations.
Finally, the buffer limit, which sets memory constraints on the scheduler and moreover in the quantity of tasks that will be allowed to form the waiting list, will be changed to visualize its effect on the loss of data.

4.2. Scheduling algorithm: FIFO

This algorithm orders the information as it enters the buffer. When the start time is reached for each sensor, its corresponding packets will be added to the buffer without taking into account any other parameters from the sensor. When the buffer is filled with all the packets being activated in each second, the buffer is then emptied up to its bandwidth limitation to simulate the effect of sending the packets. If other packets remain in the buffer after it is emptied up to its bandwidth possibilities, they will remain in the front positions for the next second.
A simple simulation was executed just taking into account the bandwidth constraints to see how it affects the delivery of the information packets.
As it can be seen in Fig.3 when the bandwidth is set at 250 kb/s all the packets stored in the buffer are sent after each second and in addition there is no packet loss in the scheduling procedure.

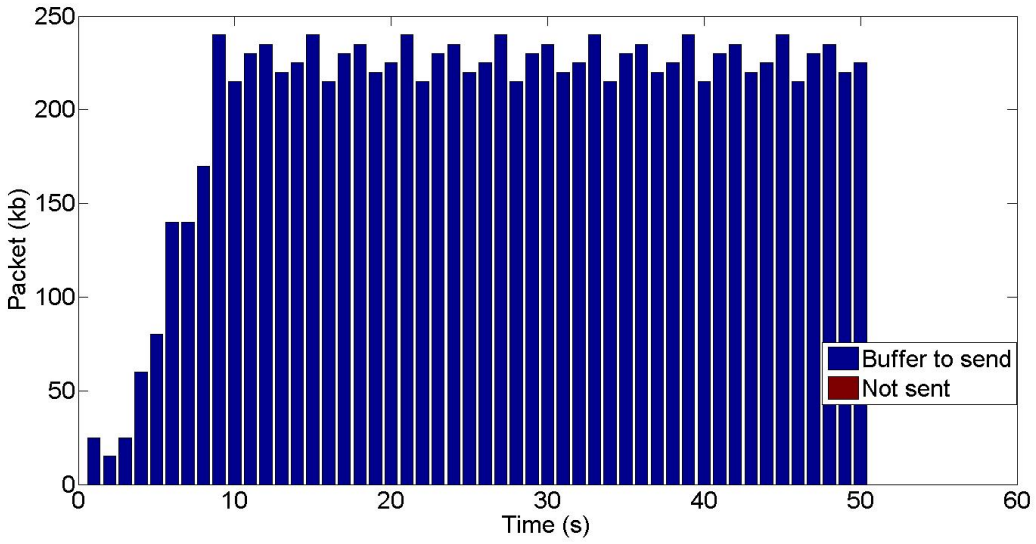

Figure 3. FIFO successfully sends all packets from the buffer in each second.

In contrast, when the buffer is reduced to a lower value for instance 220 kb/s, as can be seen in Fig.4, some cases can be observed where the amount of information in the buffer increases after each second as it is impossible for the scheduler to sent all the packets. These packets are procrastinated to the following second.

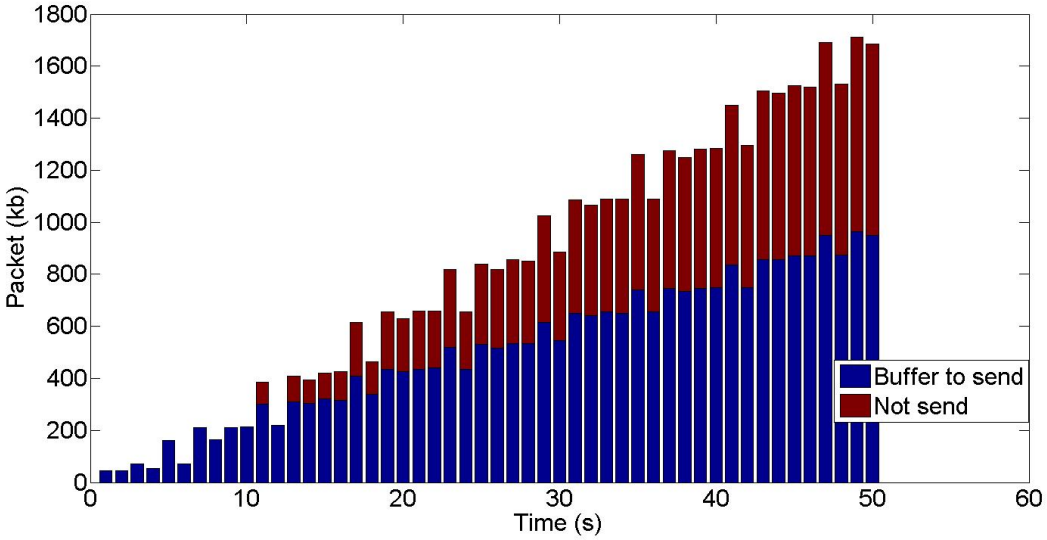

Figure 4. FIFO as a result of lowering the bandwidth is unable to send all the packets at each time step.

The amount of packets not sent has an increasing tendency. It can be noted that this will lead to failure of the system because of its inability to send the data successfully.
The FIFO algorithm has been implemented to study how the available bandwidth affects the capability of the scheduler to send the information successfully.
From this simulation it can be stated that the FIFO scheduling strategy can be a useful algorithm to be implemented in a real-time system for a smart building if the available bandwidth is big enough to send all the packets each second and avoid the accumulation of packets in the buffer second after second.

4.3. Scheduling algorithm: RM

The Rate Monotonic algorithm assigns a higher priority to those tasks with lower period. This means that the tasks being executed more often will be preferred.
In this simulation the packet loss due to expiry of the deadlines for each individual packet was implemented, as well as a buffer limitation in the amount of information above the bandwidth that is allowed to enter the buffer the following second. Thus, two different kind of packet loss are considered: The loss due to deadline expiry and the loss due to buffer size limitation after the bandwidth restriction applies.
The bandwidth and the buffer size were given smaller values, 120 kb and 100 kb respectively, than it is stated in the ZigBee protocol in order to observe and quantify the different packet loss. As it can be seen in Fig.5, in each second the buffer is filled with the information collected from the sensors in blue and the deadline expiry and buffer limiting loss of information are depicted in red and green respectively. As it was stated earlier, the data loss during the overall scheduling process will correspond in greater quantities to packets with larger periods.

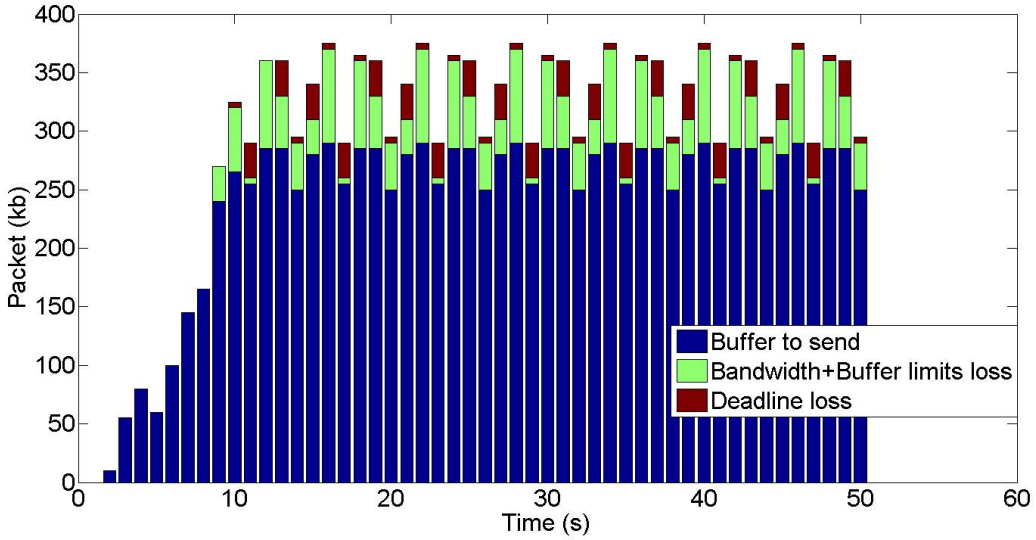

Figure 5. RM scheduling with losses due to deadlines expiry and to the limited bandwidth combined with the buffer size constraints.

It is important to realize that by having a criteria to order the tasks the loss of packets in a real-time system can be aimed to a certain objective to make sure that the packets with a specific characteristic are given priority to be sent successfully.

4.4. Scheduling algorithm: EDF

The Earliest Deadline First algorithm schedules the packets in the buffer in addition with the newly initialized packets from the sensor list attending to its associated absolute deadline.
For this simulation the bandwidth and the buffer size were fixed to the same values as the ones used for the RM algorithm in order to observe the differences among the scheduling output obtained in both of them.
As it can be seen in Fig.6 the scheduling output looks very similar to the one obtained with the RM algorithm in Fig.5. The main difference is that in Fig.6 no loss due to deadline expiry appears. This is as expected because the EDF algorithm gives priority to the packets which are close to their deadline expiry time to avoid having this sort of loss.

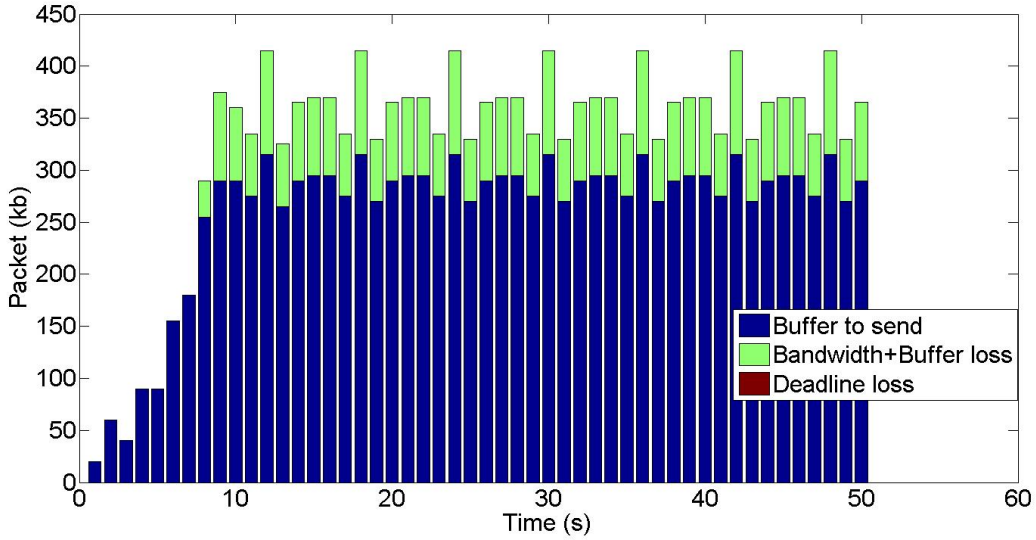

Figure 6. EDF scheduling focuses in sending the packets with lowest deadline expiry time.

It can also be observed by comparing the RM and the EDF algorithm in Fig.5 and in Fig.6 respectively, that a reduction in the deadline loss in red is achieved, however this algorithm provides more bandwidth and buffer loss in green so in reality the overall packet loss does not seem to be reduced as it is only discriminated attending the algorithm chosen.
The EDF algorithm is suitable for real-time systems in which the packet loss due to deadline expiry should to be minimized.

4.5. Scheduling algorithm: HPF

The High Priority First allows the user to choose manually how to prioritize the tasks.
In a smart buildings many different data packets are being sent by the sensors to the central control unit containing information about the different applications which are deployed in it. Being able to manually select which packets should be prioritized attending to a specific characteristic can be a very interesting approach for a real-time system where the loss of information could be kept within a margin in order to maintain a certain level to provide a better quality of service.
To ease and limit the users ability to decide how and in what proportion this can be done, the following algorithm is proposed with the objective of allowing the user to select the weight between having the EDF algorithm and a strategy to send many small and low period packets to minimize the overall individual packet loss

$$P = \alpha * K_1 * \left(\frac{ps}{K_2} + pr\right) + (1-\alpha) * (dl), \alpha \in [0,1] \tag{1}$$

where the parameter α can be manually adjusted to decide in which percentage the scheduling priority function P depends on the packet size ps and the period of control pr and in which proportion it depends in the absolute deadline dl. $K_1 = 1.125$ and $K_2 = 5kb/s$ are two constants designed to keep a proper dimensioning of the addend.
If the value of α is kept low the scheduling is made by prioritizing the lowest deadline expiry time, obtaining a very similar case as the one considered when implementing the EDF algorithm. In contrast, if the value of α is raised to values closer to 1, as in Fig.7, the buffer will be sorted attending to the lowest combination of packet size and control period.

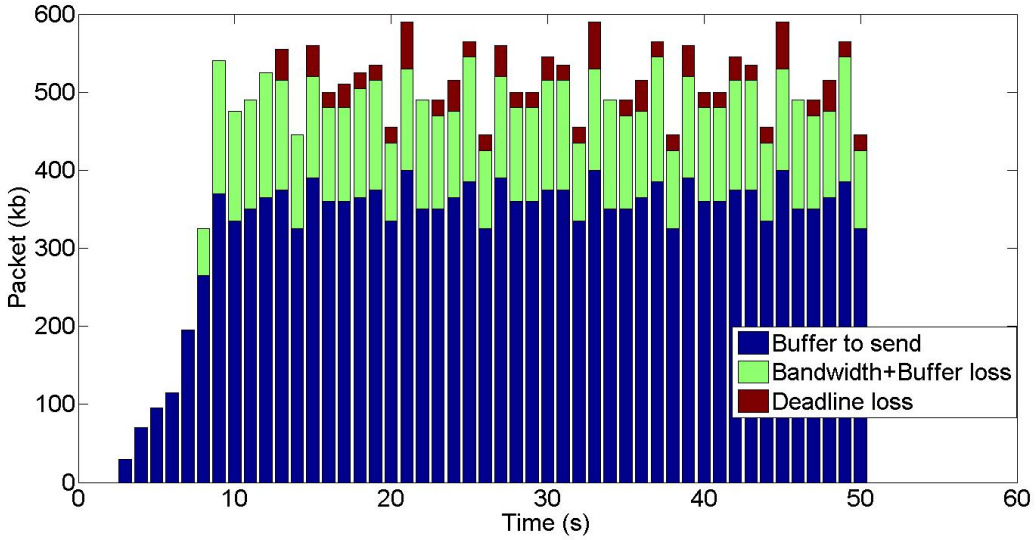

Figure 7. HPF scheduling with $\alpha = 0.9$ focuses in sending low deadline expiry time combined with starting time packets first.

Another interesting value to test is to approach α to a value close to 0.5. For this value all the data is given the same relevance so the tasks will be ordered according to their smallest mean value as can be seen in Fig.8.

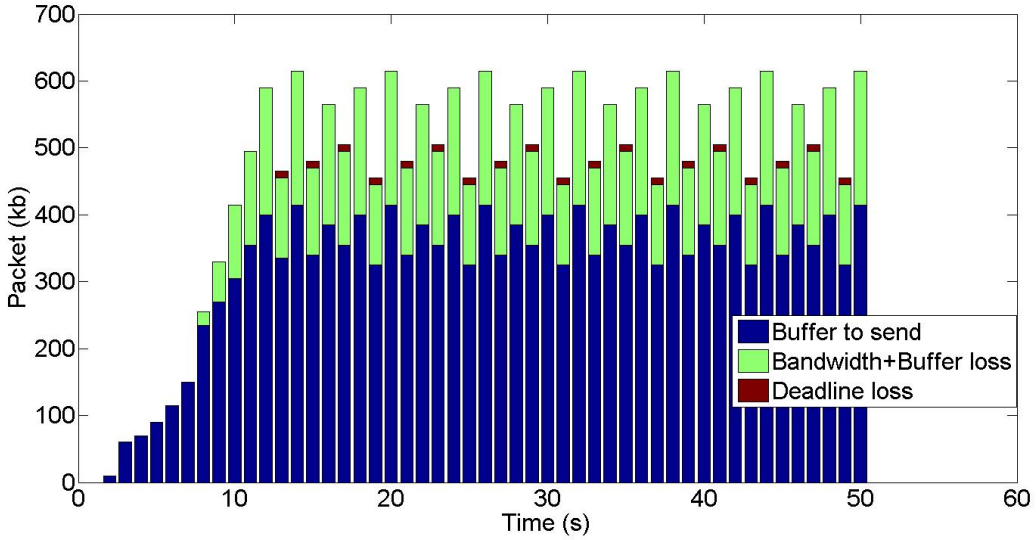

Figure 8. HPF scheduling with $\alpha = 0.5$ organizes the data attending to the mean value of all 3 properties considered.

The proposed algorithm allows the user to select the characteristics of the packets that will be offered a better position inside the buffer by changing the value of α.
Instead of looking at the overall information loss in kb, a different approach can be taken in which the packet loss is regarded attending to the quantity of lost packets as units.
For $\alpha = 0.5$ the three characteristics considered in the prioritizing algorithm are given the same relevance and the loss of many packets is avoided as can be seen in Fig.9. If this situation is taken to an extreme by making $\alpha = 1$ as in Fig.10, the small and the more frequent packets will be

sent earlier and this will result in a lower overall packet loss. The largest amount of information loss will be covered by the largest sized packets so at the end less packets will be lost.

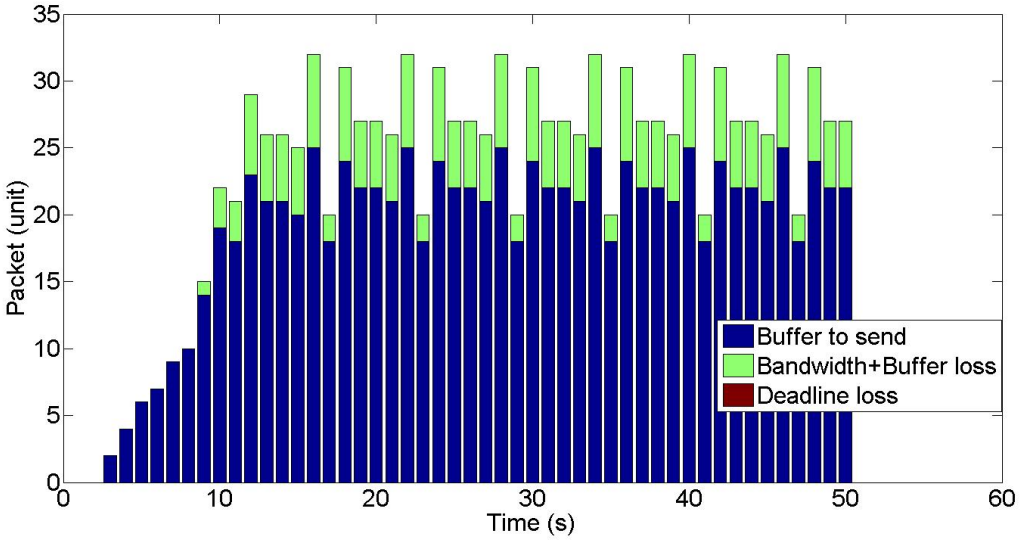

Figure 9. HPF scheduling with $\alpha = 0.5$ in terms of individual packet loss.

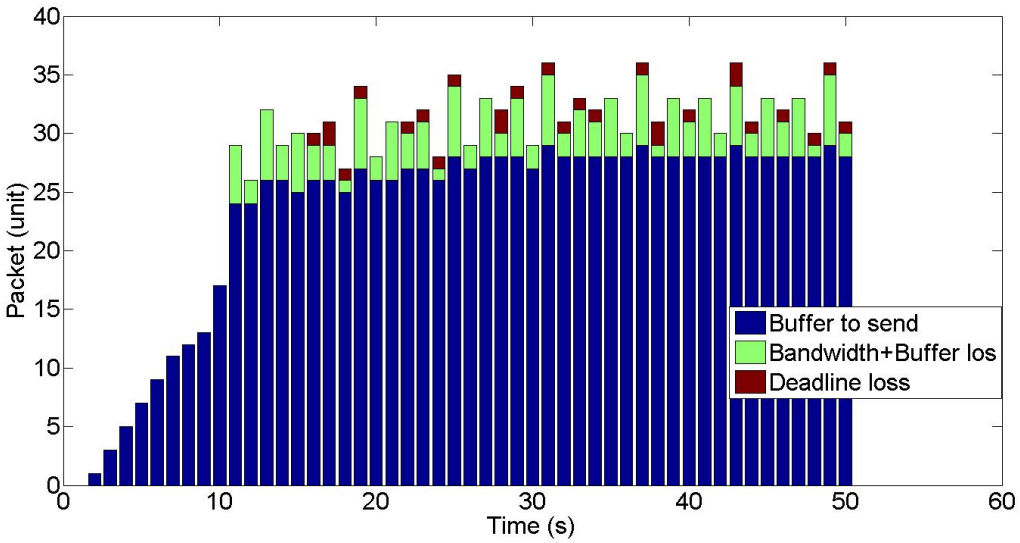

Figure 10. HPF scheduling with $\alpha = 1$ minimized packet loss since the bigger packets are the ones preferred to lose.

From a practical point of view this algorithm can be adjusted depending on the users specific needs and it provides a new approach in which smaller sized and less frequent packets are prioritized in order to diminish the unitary packet loss.

Chapter 5

Conclusions

SUCCESSFUL scheduling of all the information coming from the sensors deployed inside a smart building is essential to be able to perform and control in a proper way all the applications installed in order to coordinate security or safety functions and to make the stay inside the building more comfortable.
In this report the applications which could benefit from a wireless sensor network inside a smart building are identified, their basic parameters: Control period, latency in deadlines and size of the packets are covered and several wireless protocols are introduced. The different applications with their typical parameters are summarized in TABLE I and some sensor specific examples are illustrated in TABLE II.
In the second section the theory about real-time scheduling is presented, giving an overview on real-time systems, defining its basic parameters and explaining the distinction among different scheduling algorithms.
The last section consists of a simulation in which the loss of packets when executing several scheduling algorithms is studied and compared for different and varying bandwidth and buffer size. Later on an algorithm combining several characteristic of the information from each sensor is implemented allowing the user to decide, by a α factor, which parameters should precise a larger weight when the scheduling process is carried out. Finally, The effect of individual packets loss was depicted, commented and compared for different values of α.

Chapter 6

Future Work

THE implementation of scheduling algorithms which can be controlled by an external user could be further developed attending to many other characteristics of the packets sent by different sensors. Nevertheless, a very interesting field of study would consist in testing if these control parameters can be controlled dynamically and changed each second of the simulation based on continuous feedback to reduce the overall packet loss.

Bibliography

[1] Neil Audsley, Alan Burns, Rob Davis, Ken Tindell, and Andy Wellings. Real-time system scheduling. Springer, 1995.

[2] Giorgio C Buttazzo. Hard real-time computing systems: predictable scheduling algorithms and applications, volume 24. Springer, 2011.

[3] Ed Callaway, Paul Gorday, Lance Hester, Jose A Gutierrez, Marco Naeve, Bob Heile, and Venkat Bahl. Home networking with ieee 802. 15. 4: a developing standard for low-rate wireless personal area networks. IEEE Communications Magazine, 40(8):70–77, 2002.

[4] Carles Gomez and Josep Paradells. Wireless home automation networks: A survey of architectures and technologies. IEEE Communications Magazine, 48(6):92–101, 2010.

[5] Franck L Lewis. Wireless sensor networks. Smart environments: technologies, protocols, and applications, pages 11–46, 2004.

[6] Chenyang Lu, John A Stankovic, Sang H Son, and Gang Tao. Feedback control real-time scheduling: Framework, modeling, and algorithms*. Real-Time Systems, 23(1-2):85–126, 2002.

[7] Arezou Mohammadi and Selim G Akl. Scheduling algorithms for real-time systems. School of Computing Queen's University, Tech. Rep, 2005.

[8] Krithi Ramamritham and John A Stankovic. Scheduling algorithms and operating systems support for real-time systems. Proceedings of the IEEE, 82(1):55–67, 1994.

[9] Sanjay Shakkottai and Rayadurgam Srikant. Scheduling real-time traffic with deadlines over a wireless channel. Wireless Networks, 8(1):13–26, 2002.

[10] Amit Sinha and Anantha Chandrakasan. Dynamic power management in wireless sensor networks. Design & Test of Computers, IEEE, 18(2):62–74, 2001.

[11] Jianping Song, Song Han, Aloysius K Mok, Deji Chen, Mike Lucas, and Mark Nixon. Wirelesshart: Applying wireless technology in real-time industrial process control. In Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS'08. IEEE, pages 377–386. IEEE, 2008.

[12] Andreas Willig. Recent and emerging topics in wireless industrial communications: A selection. Industrial Informatics, IEEE Transactions on, 4(2):102–124, 2008.

PART II

SOURCE CODE

Scheduling algorithms

FIFO (First In First Out)

```
% clear variables
clear;
close all;

% Basic setting:
N = 30;
Tmax = 50;
BandWidth = 220;

% Generate Matrix: List_Nodes
ListSensor = zeros(N,4);                % pz, freq, start, deadline

% pz
ListSensor(:,1) = randi(3,N,1);
% freq
ListSensor(:,2) = randi(3,N,1);
% start
ListSensor(:,3) = randi(10,N,1);
% deadline
ListSensor(:,4) = randi(10,N,1);


% Buffer B
B.pz = [];              % package size

% initial results vectors:
resultB = zeros(1,Tmax);        % size of Buffer
resultP2 = zeros(1,Tmax);       % size of package loss Bandwith

for t = 1:Tmax

    Bt = [];        % new arrive packets
    % check which sensor send the packet
    % increase of the buffer: receive
    for i = 1:N
        deadline = ListSensor(i,4);
        starti = ListSensor(i,3);
        freqi = ListSensor(i,2);
        pzi = ListSensor(i,1);
            switch pzi
                case 1
                   pzi = 5;
                case 2
                   pzi = 10;
                case 3
                   pzi = 20;
            end
        if ~mod(t-starti,freqi)==1 && t>=starti
            B.pz = [B.pz, pzi];

        end
    end


    resultB(t) = sum(B.pz);           % what we have to deal with dark blue
```

```
    %decrease of the buffer: transmition
    Bt = B;


    % FIFO - First in, first out

    Btpz = cumsum(Bt.pz);
     IdxTmp = find(Btpz > BandWidth, 1);
     if isempty(IdxTmp)
         Bt.pz = [];              % package size

     else
        Bt.pz = Bt.pz(IdxTmp:end);

     end

    resultPl(t) = sum(Bt.pz);       % Loss due to bandwith

    B = Bt;


end

bar([resultB', resultPl'],'stacked');
```

RM (Rate Monotonic)

```
% clear variables
clear;
close all;

% Basic setting:
N = 30;
Tmax = 50;
BandWidth = 120;
BufferLimit = 100;

% Generate Matrix: List_Nodes
ListSensor = zeros(N,4);                % pz, freq, start, deadline

% pz
ListSensor(:,1) = randi(3,N,1);
% freq
ListSensor(:,2) = randi(3,N,1);
% start
ListSensor(:,3) = randi(10,N,1);
% deadline
ListSensor(:,4) = randi(10,N,1);

% Buffer B
B.pz = [];              % package size
B.st = [];              % start time
B.dl = [];              % deadline
B.fr = [];              % period
B.pl = cell(Tmax,1);    % package loss due to Deadlines expiry
B.plb = cell(Tmax,1);    % package loss due to bandwith and Buffer Limitations

% initial results vectors:
resultB = zeros(1,Tmax);        % size of Buffer
resultPl = zeros(1,Tmax);       % size of package loss buffer
resultP2 = zeros(1,Tmax);       % size of package loss Bandwith

for t = 1:Tmax

    Bt = [];        % new arrive packets
    % check which sensor send the packet
    % increase of the buffer: receive
    for i = 1:N
        deadline = ListSensor(i,4);
        starti = ListSensor(i,3);
        freqi = ListSensor(i,2);
        pzi = ListSensor(i,1);
            switch pzi
```

```
            case 1
               pzi = 5;
            case 2
               pzi = 10;
            case 3
               pzi = 20;
        end
    if ~mod(t-starti,freqi)==1 && t>starti
        B.pz = [B.pz, pzi];
        B.st = [B.st, t];
        B.dl = [B.dl, deadline+t];
        B.fr = [B.fr, freqi];
    end
end

resultB(t) = sum(B.pz);

%decrease of the buffer: transmition
Bt = B;


% RM: schedule attending smaller period.

[stTmp,stIdx] = sort(Bt.fr);
Bt.pz = Bt.pz(stIdx);
Bt.st = Bt.st(stIdx);
Bt.dl = Bt.dl(stIdx);
Bt.fr = Bt.fr(stIdx);


idxTmp = find(Bt.dl >= t);
idxLoss = find(Bt.dl < t);

Bt.pl{t,1} = Bt.pz(idxLoss);

resultP2(t) = sum (Bt.pl{t,1});

Bt.pz = Bt.pz(idxTmp);
Bt.st = Bt.st(idxTmp);
Bt.dl = Bt.dl(idxTmp);
Bt.fr = Bt.fr(idxTmp);


Btpz = cumsum(Bt.pz);
 IdxTmp = find(Btpz > BandWidth, 1);
 if isempty(IdxTmp)
     Bt.pz = [];              % package size
     Bt.st = [];              % start time
     Bt.dl = [];              % deadline
     Bt.fr = [];              % period
 else
    Bt.pz = Bt.pz(IdxTmp:end);
    Bt.st = Bt.st(IdxTmp:end);
    Bt.dl = Bt.dl(IdxTmp:end);
    Bt.fr = Bt.fr(IdxTmp:end);

    Btbz = cumsum(Bt.pz);
    IdxTmp2 = find(Btbz > BufferLimit, 1);
    if ~isempty(IdxTmp2)
        Bt.plb{t,1} = Bt.pz(IdxTmp2:end);
        Bt.pz = Bt.pz(1:IdxTmp2-1);
        Bt.st = Bt.st(1:IdxTmp2-1);
        Bt.dl = Bt.dl(1:IdxTmp2-1);
        Bt.fr = Bt.fr(1:IdxTmp2-1);

    end


 end

 resultPl(t) = sum(Bt.plb{t,1});

B = Bt;
```

```
end

bar([resultB', resultPl', resultP2'],'stacked');
```

EDF (Earliest Deadline First)

```
% clear variables
clear;
close all;

% Basic setting:
N = 30;
Tmax = 50;
BandWidth = 120;
BufferLimit = 100;

% Generate Matrix: List_Nodes
ListSensor = zeros(N,4);                % pz, freq, start, deadline

% pz
ListSensor(:,1) = randi(3,N,1);
% freq
ListSensor(:,2) = randi(3,N,1);
% start
ListSensor(:,3) = randi(10,N,1);
% deadline
ListSensor(:,4) = randi(10,N,1);


% Buffer B
B.pz = [];              % package size
B.st = [];              % start time
B.dl = [];              % deadline
B.pl = cell(Tmax,1);    % package loss due to deadline expiry
B.plb = cell(Tmax,1);    % package loss due to Bandwith and buffer limitations

% initial results vectors:
resultB = zeros(1,Tmax);        % size of Buffer
resultPl = zeros(1,Tmax);       % size of package loss buffer
resultP2 = zeros(1,Tmax);       % size of package loss Bandwith

for t = 1:Tmax

    Bt = [];        % new arrive packets
    % check which sensor send the packet
    % increase of the buffer: receive
    for i = 1:N
        deadline = ListSensor(i,4);
        starti = ListSensor(i,3);
        freqi = ListSensor(i,2);
        pzi = ListSensor(i,1);
            switch pzi
                case 1
                   pzi = 5;
                case 2
                   pzi = 10;
                case 3
                   pzi = 20;
            end
        if ~mod(t-starti,freqi)==1 && t>=starti
            B.pz = [B.pz, pzi];
            B.st = [B.st, starti];
            B.dl = [B.dl, deadline+t];
        end
    end


    resultB(t) = sum(B.pz);           % what we have to deal with dark blue


    %decrease of the buffer: transmition
    Bt = B;
```

```
    % EDF
    idxTmp = find(Bt.dl >= t);
    idxLoss = find(Bt.dl < t);

    Bt.pl{t,1} = Bt.pz(idxLoss);

    resultP2(t) = sum (Bt.pl{t,1});         % Loss due to deadlines red

    Bt.pz = Bt.pz(idxTmp);
    Bt.st = Bt.st(idxTmp);
    Bt.dl = Bt.dl(idxTmp);



    [dlTmp,dlIdx] = sort(Bt.dl);
    Bt.pz = Bt.pz(dlIdx);
    Bt.st = Bt.st(dlIdx);
    Bt.dl = Bt.dl(dlIdx);

    Btpz = cumsum(Bt.pz);
     IdxTmp = find(Btpz > BandWidth, 1);
     if isempty(IdxTmp)
         Bt.pz = [];              % package size
         Bt.st = [];              % start time
         Bt.dl = [];              % deadline
     else
        Bt.pz = Bt.pz(IdxTmp:end);
        Bt.st = Bt.st(IdxTmp:end);
        Bt.dl = Bt.dl(IdxTmp:end);

        Btbz = cumsum(Bt.pz);
        IdxTmp2 = find(Btbz > BufferLimit, 1);
        if ~isempty(IdxTmp2)
            Bt.plb{t,1} = Bt.pz(IdxTmp2:end);
            Bt.pz = Bt.pz(1:IdxTmp2-1);
            Bt.st = Bt.st(1:IdxTmp2-1);
            Bt.dl = Bt.dl(1:IdxTmp2-1);

        end


     end

    resultPl(t) = sum(Bt.plb{t,1});       % Loss due to bandwith application and the maximum
        buffer green

    B = Bt;


end

bar([resultB', resultPl', resultP2'],'stacked');
```

HPF (High Priority First)

```
% clear variables
clear;
close all;

% Basic setting:
N = 30;
Tmax = 50;
BandWidth = 120;
BufferLimit = 100;
alpha = 0.9;                              % should be a number between [0,1] it allows us to
    prioritize  between small and fast against deadlines and executing more frequently tasks.

% Generate Matrix: List_Nodes
ListSensor = zeros(N,4);                % pz, period, start, deadline

% pz
ListSensor(:,1) = randi(3,N,1);
```

```
% freq
ListSensor(:,2) = randi(3,N,1);
% start
ListSensor(:,3) = randi(10,N,1);
% deadline
ListSensor(:,4) = randi(10,N,1);


% Buffer B
B.pz = [];              % package size
B.st = [];              % start time
B.dl = [];              % deadline
B.pr = [];              % priority parameter
B.pl = cell(Tmax,1);    % package loss due to Bandwith
B.plb = cell(Tmax,1);    % package loss due to Buffer Limit

% initial results vectors:
resultB = zeros(1,Tmax);        % size of Buffer
resultPl = zeros(1,Tmax);       % size of package loss buffer
resultP2 = zeros(1,Tmax);       % size of package loss Bandwith

for t = 1:Tmax

    Bt = [];        % new arrive packets
    % check which sensor send the packet
    % increase of the buffer: receive
    for i = 1:N
        deadline = ListSensor(i,4);
        starti = ListSensor(i,3);
        freqi = ListSensor(i,2);
        pzi = ListSensor(i,1);
            switch pzi
                case 1
                   pzi = 5;
                case 2
                   pzi = 10;
                case 3
                   pzi = 20;
            end
        if ~mod(t-starti,freqi)==1 && t>starti
            B.pz = [B.pz, pzi];
            B.st = [B.st, starti];
            B.dl = [B.dl, deadline+t];
            B.pr = [B.pr, alpha*1.125*(pzi/5+freqi)+(1-alpha)*(deadline+t)]
        end
    end

    resultB(t) = sum(B.pz);


    %decrease of the buffer: transmition
    Bt = B;

    % HPF - assign priority manually, by changing alpha

    [stTmp,stIdx] = sort(Bt.pr);
    Bt.pz = Bt.pz(stIdx);
    Bt.st = Bt.st(stIdx);
    Bt.dl = Bt.dl(stIdx);
    Bt.pr = Bt.pr(stIdx);


    idxTmp = find(Bt.dl >= t);
    idxLoss = find(Bt.dl < t);

    Bt.pl{t,1} = Bt.pz(idxLoss);

    resultP2(t) = sum (Bt.pl{t,1});

    Bt.pz = Bt.pz(idxTmp);
    Bt.st = Bt.st(idxTmp);
    Bt.dl = Bt.dl(idxTmp);
    Bt.pr = Bt.pr(idxTmp);
```

```
    Btpz = cumsum(Bt.pz);
     IdxTmp = find(Btpz > BandWidth, 1);
     if isempty(IdxTmp)
         Bt.pz = [];              % package size
         Bt.st = [];              % start time
         Bt.dl = [];              % deadline
         Bt.pr = [];              % priority
     else
        Bt.pz = Bt.pz(IdxTmp:end);
        Bt.st = Bt.st(IdxTmp:end);
        Bt.dl = Bt.dl(IdxTmp:end);
        Bt.pr = Bt.pr(IdxTmp:end);

        Btbz = cumsum(Bt.pz);
        IdxTmp2 = find(Btbz > BufferLimit, 1);
        if ~isempty(IdxTmp2)
            Bt.plb{t,1} = Bt.pz(IdxTmp2:end);
            Bt.pz = Bt.pz(1:IdxTmp2-1);
            Bt.st = Bt.st(1:IdxTmp2-1);
            Bt.dl = Bt.dl(1:IdxTmp2-1);
            Bt.pr = Bt.pr(1:IdxTmp2-1);

        end


     end


     resultPl(t) = sum(Bt.plb{t,1});


    B = Bt;
end

bar([resultB', resultPl', resultP2'],'stacked');
```