UNIVERSIDAD AUTÓNOMA METROPOLITANA

What Users Want(WUW): un servicio de satisfacción de usuarios orientado a aplicaciones de distribución de contenidos

Tesis que presenta:
Adriana Pérez Espinosa
Para obtener el grado de
Maestra en Ciencias y Tecnologías de la Información

Asesoras:
Dra. Elizabeth Pérez Cortés
Dra. Patricia Serrano Alvarado

Jurado Calificador:

Dra. Reyna Carolina Medina Ramírez	UAM-I
Dr. José G. Rodríguez García	CINVESTAV-IPN
Dr. Manuel Aguilar Cornejo	UAM-I

México, D.F

Resumen

Las aplicaciones Par-a-Par (P2P) se han vuelto populares debido a su alta escalabilidad, y es gracias a la participación de los usuarios que el proceso de distribución se lleva a cabo, ya que estos comparten sus recursos (como el ancho de banda, almacenamiento, etc) e intercambian el contenido con otros usuarios.

Es por ello que hoy en día, muchas aplicaciones P2P pertenecen a la categoría de distribución de contenidos, las cuales van desde el intercambio de archivos hasta los sistemas que permiten crear una infraestructura P2P para organizar, buscar, y recuperar contenido [1].

No obstante, a pesar de su éxito, aún existen diferentes problemáticas relacionadas con el procesos de distribución. La mayoría de los trabajos relacionados con este tema enfocan sus esfuerzos en mejorar la arquitectura, la robustez en escenarios de alta dinamicidad o bien de mejorar la habilidad de descubrir nuevos pares para compartir contenido.

En otras palabras, el enfoque en estas propuestas se basa principalmente en la calidad de servicio ofrecida [2][3][4]. Debido a que gracias a los usuarios, la distribución de contenidos se lleva a cabo, este tipo de aplicaciones no sólo deben preocuparse por la calidad de servicio, sino también por la satisfacción de sus usuarios, ya que son los principales actores en estas aplicaciones.

Por lo tanto, este tipo de aplicaciones deberían hacer que los usuarios participen más en el proceso de distribución, teniendo en cuenta sus preferencias sobre el contenido y la información que comparten. Sin embargo, ninguna de las aplicaciones P2P de distribución de contenidos estudiadas en el presente trabajo considera las preferencias de los usuarios. Por lo que, se propone la utilización de un servicio que permita a los usuarios definir sus preferencias y tomarlas en consideración para la selección de los usuarios con los que desea intercambiar el contenido.

En este trabajo, se presenta el servicio WUW que permite a los usuarios expresar sus preferencias y que sean tomadas en cuenta durante el proceso de distribución. Se presenta su arquitectura e implementación. De igual forma, WUW permite medir la satisfacción de los usuarios y cómo la aplicación P2P utilizada considera las preferencias de los usuarios, proporcionando una retroalimentación inspirada en Satisfaction-Based Query Load Balancing framework (SQLB) [5], pero cuya definición formal en WUW es esencialmente diferente

[6] [7].

También como parte de nuestro trabajo de investigación se diseña una interfaz web, que actúa como un intermediario entre los usuarios y el servicio WUW, a través de la cual los usuarios definen sus preferencias para que WUW pueda considerarlas, y le muestra a los usuarios la retroalimentación proporcionada por WUW.

Finalmente, se realiza la evaluación del servicio WUW utilizando como aplicación P2P BitTorrent, el cual utiliza uno de los protocolos más popular para el intercambio de contenidos. Los resultados de nuestra evaluación, muestran que la presencia del servicio WUW no afecta el rendimiento de BitTorrent al considerar las preferencias de los usuarios al momento de seleccionar con quienes se desea compartir el contenido. Así mismo, se presenta una demostración del uso de la interfaz web, que permite a los usuarios definir sus preferencias.

UNIVERSIDAD AUTÓNOMA METROPOLITANA

What Users Want(WUW): a User Satisfaction Service Oriented to Content Delivery Applications

by
Adriana Pérez Espinosa

A thesis submitted in partial fulfillment of the requirements for the degree of Master in Science (Sciencies and Information Technologies)

Advisers:
PhD. Elizabeth Pérez Cortés
PhD. Patricia Serrano Alvarado

México, D.F

Abstract

At the present time, Peer-to-Peer (P2P) applications are frequently used to distribute different kinds of content, additionally in these applications we found a solution to reduce the maintenance costs and the drawbacks when there is a high demand of the content.

These applications are highly scalable and thanks to users participation the distribution process is carried out, because users share their resources (like bandwidth, storage, etc.) and they exchange the content with others.

Therefore is important to consider users preferences during the distribution process about the usage of their resources, the content that they are sharing or with whom they share the content. Nowadays, most of the efforts of these applications mainly address to improve QoS as the downloading time, the availability of content, the integrity of the content, the resolution, etc.

But none of the P2P content delivery applications (P2P-CDA) studied in the present work consider users preferences for sharing the content based on aspects like the geographical location, the kind of content, the interest in the content, the bandwidth that other users offer, etc.

In the present work, we propose WUW (What Users Want) a service that allows users to define their preferences and take them into consideration for selecting the neighbors with whom to trade with.

We tested our service using BitTorrent, which use one of the most popular protocol for sharing content. The results of the experiment shows that the presence of WUW service does not affect the performance of BitTorrent.

Acknowledgements

I want to thank my advisors: PhD. Elizabeth Pérez Cortés and PhD. Patricia Serrano Alvarado, for all the time dedicated to my work, their generous guidance and mentoring.

To the Consejo Nacional de Ciencia y Tecnología (CONACyT) for the finantial support to conduct this thesis.

To all the members of P2PWeb project for allowing me participate in this project and give me the great experiance of working in team.

To the Université de Nantes, the personal of Laboratoire Informatique de Nantes Atlantique, in special to the GDD team which made me feel at home during my stay in France.

Thanks to my parents and my sisters for their inconditional supported, I love you.

To Andrea and Aldo for being part of my life and give me great joys.

To my friends who have supported me throughout the process, in special to my best friends Luis A. Pérez and Luis A. Alarcón. I will always appreciate all they have done for me.

To PhD. Manuel Aguilar Cornejo for his guidance and advice not only as a professor but as a great friend.

Special thanks to José Luis, for his unconditional love, his words of encouragement and the support he gave me.

Contents

List of Figures

List of Tables

Acronyms

P2P Peer-to-Peer

VoD Video-on-Demand

CDN Content Delivery Network

HTTP Hyper Text Transfer Protocol

URL Universal Resource Locator

LSF Least-Shared First

RUp Random Useful peer

MDp Most Deprived peer

BAWp Bandwidth AWare Scheduler peer

WUW What User Wants

CDA Content Delivery Application

QoS Quality of Service

P2P-CDA Peer-to-Peer Content Delivery Application

Chapter 1
Introduction

Nowadays, Internet is being used for different purposes not originally envisaged. A challenge for the Internet infrastructure has been delivering data to a large number of users, resulting in the emergence of applications dedicated to content distribution. The goal of ***Content Delivery Applications*** or ***Content Distribution Applications*** **(CDAs)** is to distribute the content to end users with high availability and high performance.

Today, several CDAs exist as streaming applications, VoD applications and file sharing applications which consume the majority of the Internet bandwidth [8]. With the continuous demand for such applications, the problem of efficient content distribution increased importantly. Different architectures were proposed to content distribution. The goal is to handle high amounts of demands more efficiently, to offer customers a better service while reducing its resource requirements and delivery costs.

The basic architecture for content distribution over the Internet is the Client/Server architecture [9]. In this architecture, the content is stored in a source server, and the clients request the content to the server. However, this architecture has two main disadvantages: first, when the number of clients grows a bottleneck occurs and second, if the server crashes the distribution is stopped.

The Content Delivery Network (CDN) is a variation of the Client/Server architecture. In this architecture, the source server first pushes the content to a set of content delivery servers placed strategically at the network. This is known as *content replication*. This replication makes possible the content distribution to many users even when the levels of traffic are high.

When a client is interested in one content, instead of downloading it from the content source server, the client is directed to a nearby server that will provide the lowest latency and the best connection speed for the client. However, the bandwidth required in a Client/Server architecture or in a CDN, grows up proportionally with the client population, so this makes these architectures expensive [10][11].

The Peer-to-Peer *(P2P)* architecture emerged like a solution to the disadvantages shown by the previous architectures. The basic design of this architecture is to incite clients to act as content providers (servers) and content consumers (clients), namely as ***peers***. In a P2P architecture, peers not only download data from the network, but also upload the downloaded data to other users in the system [12][13].

The principal reasons that have fostered the explosive growth of use of P2P architectures are the low cost and high availability of resources, thanks to the fact that each peer shares

its resources, so if number of users grows, the amount of available resources will increased.

Usually, peers are under the control of users who autonomously decide when to join or to leave the P2P system. Each peer can join or leave the system at any arbitrary time, this means, the dynamicity of peers participation, which is also called *peer churn* [13][14], is an inherent property of P2P systems. Peer churn significantly affects the performance of P2P systems, therefore is essential to incorporate an adequate degree of robustness in order to ensure the content distribution.

1.1 Problem statement

Today, many P2P applications belong to the category of content distribution *(P2P-CDA)*, such applications vary from file sharing to systems that allow to create a P2P based infrastructure to organize, to search, to index and to retrieve content [1].

Despite the success of P2P applications, there are still many issues related to the content distribution. Most of the efforts mainly address the improvement of aspects like the overlay P2P topology, the robustness in high churn scenarios or the ability to discover new peers to share content. In other words, the focus in these proposals is the quality of service (QoS) of the content distribution [2][3][4].

The CDAs should worry not only about the QoS, but also for the satisfaction of their users, because they are the main players in these applications. Therefore, such applications should make users participate more in the content distribution, considering their preferences about the content and the information that they share.

In the present thesis, we present a service which allows users to express their personal preferences concerning their participation in the system. Preferences are considered to select the peers for the content exchange and the users obtain a feedback about how these preferences are respected by the application during the distribution process.

1.2 Objectives

This thesis propose a P2P service, which has no dependence with the P2P-CDA. Such service allows users to define their personal preferences to be considered by the P2P-CDA. For this, the main objectives are:

- To analyze some existing P2P content delivery applications in order to determine how they consider users preferences.
- To design and to implement a service that considers users preferences, other than QoS related, in order to improve their satisfaction.
- To design a web interface that allows the communications between to users and the service.

It is important to mention that the development of this service is a collaborative work in which each of the people who are part of the team performed specific tasks.

1.3 Thesis contributions

In this thesis, the contributions are:

1. A state of the art in P2P-CDAs, considering

   - The topologies of the overlay networks used by P2P-CDAs; a topology defines how peers are connected among them.
   - The strategies used by P2P-CDAs to select the data and the peers with whom the content will be shared.
   - A brief description of some P2P-CDA applications and a comparison of their characteristics.

   The elaboration of this state of the art was in charge of the student Adriana Pérez E.

2. Design and implementation of a P2P service which considers the user preferences during the distribution process. This service is able to interact with any P2P-CDA thanks to its well-defined interfaces and its modular design. This task was in charge of the Ph.D. Marco Biazzini.

3. Design and implementation of a web interfaces which allows to users define their preferences and receive the feedback provided by the WUW service. This task was in charge of the student Adriana Pérez E.

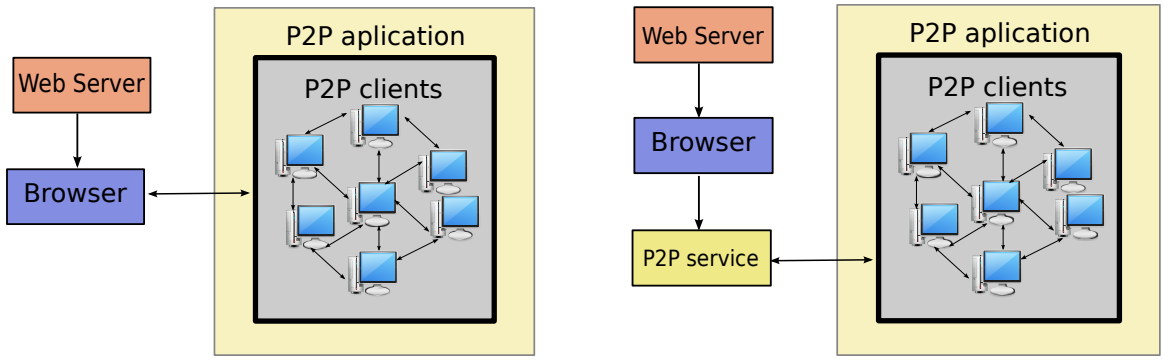
1.4 P2P service as part of a P2P-CDN architecture

The proposed service was developed in the context of the P2PWeb project, a project funded by the Region Pays de la Loire in France.

As we mention before, architectures like CDN and P2P architectures, were used. However, both technologies have their own limitations: CDN servers are expensive to maintain and generate costs for providers. On the other hand, a P2P-based architecture requires sufficient number of peers that have the complete content to jumpstart the distribution process.

The P2PWeb project propose to merge a CDN with a P2P distribution mechanism, such architecture will allow users become participants in the content distribution. The principal goals are to achieve higher scalability and to reduce structural costs for the content provider.

In the Figure 1.1, we present the global architecture of a P2P-based CDN application. Our service is located between the browser and the P2P application. The browser provides a user interface to access to the P2P-based CDN server. A P2P client is a component that implements the P2P communication protocol and we consider only one client by user. The web server contains all the information about the content provided by the CDN [6].



(a) Without a P2P service (b) With a P2P service

Figure 1.1: Global view of the P2P-based CDN application.

1.5 Thesis structure

The remaining of the thesis is composed by the following chapters:

Chapter 2 presents a the state of the art about the P2P-CDAs and, at the end, we present a comparison of the studied P2P-CDAs.

In the Chapter 3, we present the functional architecture of our service and then, we describe the modules through its functional and no-functional requirements.

Chapter 4 introduces details about the implementation of our service. In this chapter we describe the operation of a BitTorrent client, which is the chosen P2P application to test our service[1]. Then, we briefly describe how each module of WUW is implemented.

In Chapter 5 we present the evaluation of our service. We describe the experimentation and the explaination of our the results. Finally, Chapter 6 presents our conclusions and we identify some lines of the future work.

[1]This description was elaborate by the student Adriana Pérez E.

Chapter 2
State of the Art

The P2P-CDAs are usually used to distribute different kinds of contents over the Internet, as pictures, text files, real time audio and video streaming. These applications form a co-operative network where the participants share their resources in order to increase the total service capacity of the P2P system.

This chapter presents the basic concepts and strategies used by P2P-CDAs in order to understand the distribution process, and how such applications determine with whom and how the content is shared.

2.1 Overlay topologies in P2P-CDAs

Every peer in the P2P-CDAs is organized in an overlay network. The topology of this overlay defines how peers are connected, the use of a particular topology has a tight connection with the kind of application because it can influence the application aspects like performance, efficiency, robustness, etc.

According to several studies there are three different overlay topologies in P2P-CDAs: chain topology, tree-shaped topology and mesh-shaped topology [15][13][14][16]. Each of them can be more o less adequate according to the type of application, e.g., P2P live streaming, P2P Video-on-Demand, P2P file sharing, etc.

2.1.1 Chain overlay

The P2P-CDAs can organize the participant peers in a chain. This overlay is the simplest architecture that can be defined. Each peer downloads and uploads the content from/to exactly one peer as shown in Figure 2.1, and every peer can be part of a single chain. At the present, this kind of overlay are used for content sharing through Mobile Ad Hoc networks [17][18].

However all the peers take exactly the same time to download the content that they would take if the content were download from the source node, considering an homogeneous case

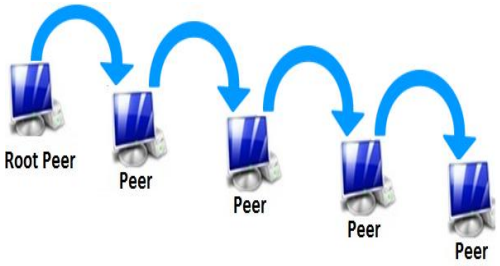


Figure 2.1: Chain overlay

of bandwidth. Also, this overlay does not withstand the minimal presence of churn, because any absence of a peer interrupts the content distribution to others peers that are in the rest of the chain.

2.1.2 Tree-shaped overlay

The tree-shaped overlay shown in Figure 2.2 organizes peers in a tree. The source of the content is the root, and the rest of peers are children of the source and in turn, they acts as children and parents among themselves. The content is distributed by pushing data from a peer to its children peers, in a continous way.

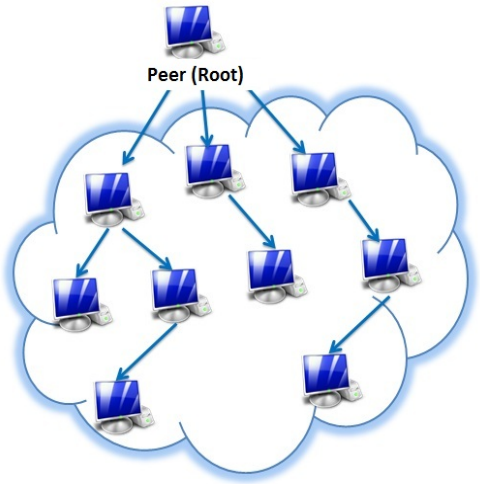


Figure 2.2: Tree-shaped overlay

The P2P content delivery applications that use a tree-shaped overlay are vulnerable to *peer churn*; when a peer leaves the network, all the peers in its sub-tree are temporarily disconnected of the distribution. So it is necessary to have mechanisms that help the recovery of the tree as soon as possible.

Another drawback in a tree-shaped overlay is that leaf peers do not contribute with their bandwidth for uploading. Since a large portion of peers in the systems are leaf peers, this behavior degrades the efficient utilization of the bandwith.

2.1.3 Mesh-shaped overlay

Many P2P content delivery applications have adopted the mesh-shaped overlay, shown in Figure 2.3, where at any given time peers can establish connections with multiple peers, called neighbors. One peer may simultaneously download content from multiple peers, and upload content to multiple peers. Regarding peer churn, the mesh-shaped overlay is most robust because if a peer's neighbor leaves the system, the peer can still download the content from remaining peers.

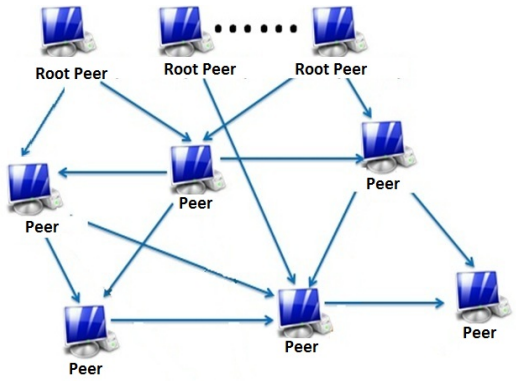


Figure 2.3: Mesh-shaped overlay

In some topologies each peer can select its neighbors and the order in which the content is shared. In the next section we describe the strategies which peers use for these purposes.

2.2 Chunk and peer selection strategies

In the P2P-CDA an important aspect is the data dissemination among the peers. In order to carry out the distribution process, the source peer splits the content into pieces called ***chunks***. Then, peers can exchange chunks with others, in order to speed up the distribution process [14] [16].

The chunk selection and peer selection are fundamental strategies for achieving content delivery, efficiently. On the one hand, an efficient chunk selection should guarantee that each peer can always find interesting chunks from its neighbors. On the other hand, an efficient peer selection should maximize the service capacity of the system, improving the downloading experience of peers that contribute to the content exchange by uploading chunks,

and punishing peers that download chunks from others without uploading any chunk. In the next sections, we briefly summarize the most used strategies by P2P-CDAs [19][20].

2.2.1 Chunk selection strategies

In this section, we present the strategies used for selecting chunks to download them from the set of available chunks in an uploader (peer that uploads chunks to others).

Random Selection

This selection strategy, chooses one of the available chunks for download in a randomly way. This strategy is easy to implement and does not require any knowledge about the content or the distribution of the chunks. However, the random selection causes problems in scenarios where peers go offline after finishing the download, or in networks with a high churn rate.

Eventually, during the random selection, some chunks are less distributed than others because only chunks that have been downloaded previously can be shared. Then, for many peers this least-shared chunk will be the last one that they need to complete a copy of the content. When a peer finally gets this last chunk, it could leave the system before being able to share it. If peers are selfish and leave the system, the number of copies of a rare chunk would never increase and could be lost. This problem is known as ***chunk starvation*** and it causes high download times or the failure of the download [21][22].

Least-Shared First

The Least-Shared First (LSF) strategy is a solution to the chunk starvation problem. This strategy selects the chunk with the fewest number of copies in the network from the available chunks of the uploaders. If there are several chunks which are least shared, then this strategy selects one of them according to some policy as priority-deadline [1], randomly, number of sequence, etc.

LSF prevents that any chunk could be shared more times than the others, as a result, the availability of every piece is the same. To achieve this strategy requires global knowledge about the distribution of all chunks in the whole network [23][19].

Sequential

The sequential strategy takes into account the number of sequence or the linear importance of chunks. In short, the peers always select the available chunk with the minimal sequence number, i.e., a peer never downloads the $chunk_{i+1}$ if it has not the $chunk_i$. Respectively peers always selects the available chunk with the higher priority, i.e., a peer never downloads

[1]The priority-deadline policy defines the chunk's priority according to the playback deadline of the chunk. Hence, a chunk with near playback deadline has a higher priority than a chunk with a far playback deadline. This policy is commonly used in P2P streaming or P2P-VoD applications.

a chunk with priority i if it has not a chunk with priority $i+1$. Given that the download/upload scheduling policy strictly obeys a sequential priority, peers can not accept out-of-order chunks even if the bandwith utilization allows it [22][23].

The use of a chunk selection strategy depends on the type of application. For applications oriented to the VoD, streaming or P2P-TV, it is not possible to download in an arbitrary order. For a quality of service, it is necessary that these applications download as many chunks as possible available for display at a particular moment in the time. As a consequence, some form of order has to be imposed in the download process. The applications are free to implement other chunk selection strategies or mergering some of the previous strategies.

2.2.2 Peer selection strategies

As we said previously, an efficient peer selection strategy should maximize the capacity of the service in the system, improving the downloading experience of those peers that contribute to the content exchange by uploading chunks with others. This strategy allows peers to select others in their neighborhood for exchanging chunks. Below, we present some peer selection strategies [24][25] [26].

Random Useful Peer (RUp)

In this strategy, $Peer_i$ randomly selects a peer in its neighbor set. The main advantage of the Random Useful peer strategy is that peers do not need information about the underlying network.

Most Deprived Peer (MDp)

In the MDp strategy, $Peer_i$ selects a $Peer_j$ form its neighborhood and that has the smallest number of chunks provided by $Peer_i$.

Bandwidth Aware Scheduler Peer (BAWp)

With this strategy, $Peer_i$ selects a $Peer_j$ according to the uploading rate of $Peer_j$. If $Peer_j$ has a high uploading rate, then it has more probability to be selected.

Minimum depth strategy

The minimum depth strategy is used in the tree-shaped overlay. In this strategy, the $Peer_i$ selects a $Peer_j$, if the $Peer_j$ has the bandwith capacity to admit a new peer. The $Peer_i$ searches from the root to the leafs of the tree overlay. If there are multiple peers that satisfy this condition, $Peer_i$ chooses the nearest peer in the overlay based in the network delay.

Longest-first strategy

In the longest-first strategy, the $Peer_i$ selects a $Peer_j$ according to the lifetime of the $Peer_j$. If $Peer_j$ is the longest-lived peer then, $Peer_i$ selects it to exchange chunks. This strategy follows the hypothesis that older peers keep the longer time than younger peers.

As in the chunk selection strategy, the peer selection strategy impacts the quality of service of the applications. So, the use of a particular strategy depends of the desired QoS and on the kind of application, i.e. streaming, VoD or file-sharing.

The success of P2P systems depends heavily on the contribution of resources from each peer that joins the system. Therefore, users must be motivated to stay longer and share their resources. In order to do that, several P2P-CDAs allow users to specify their preferences. In the next section, we present some P2P-CDAs and we analyse how user's preferences are considered.

2.3 Users preferences in P2P-CDAs

In P2P-CDAs, users are essential because their participation in the network and their resources allow to carry out the content distribution. Therefore, users should be able to specify their preferences concerning the usage of their resources, and to measure the satisfaction not only in terms of QoS but also in terms of how the system satisfies their personal preferences.

Some preferences that the user can define in the application is the highest value of bandwidth, the kind of content, the peers with whom she wants to share the content, the type of personal information that she wants to share (IP address, peer ID, geographic location, etc.). In Section 2.3.1, we describe different P2P-CDAs and in Section 2.3.2, we present a brief analysis of them, in particular, we studied their architecture and which users preferences they consider.

2.3.1 P2P-CDAs description

In this section, we briefly describe some P2P-CDAs ranging from file sharing to streaming and P2P-TV applications.

subsubsectionAnySee *AnySee* [27][28] is an application that distributes live media streaming. It was developed in the ***CERNET*** *(China Education and Research Network)* and released in the Summer of 2004. In *AnySee* peers can belong to multiple overlays, which improves the utilization of resources and the distribution of traffic.

BitTorrent

BitTorrent [20][29][30] is one of the most popular applications that allows file sharing, it was launched in 2002 by Bram Cohen. Its main objective is the efficient content delivery.

BitTorrent does not implement a function for content searching.

GoalBit

GoalBit [31] is an open source P2P system for the distribution of real-time video streams through the Internet. The *GoalBit* client, in the same way that other applications of streaming P2P networks, is a complete application for the distribution of live video. GoalBit implements its own transport protocol, ***GBTP*** *(GoalBit Transport Protocol)* and packetized protocol, ***GBPS*** *(GoalBit Packetized Stream).*

PPLive

PPLive [32][33] is one of the most popular P2P streaming software in China. It has two major communication protocols. Firstly, the registration and peer discovery protocol, and second one is the P2P chunk distribution protocol. This application was created as a project of the University of Science and Technology of Huazhong. The service is available only for distribution of live video or pre-recorded video.

PPStream

PPStream [34] is one of the most popular internet television distribution systems in China. It was released in early 2006 and its content is aimed at users of this country. Most of the content is broadcasted in China, Korea, Japan and Hong Kong. This application carries out the data distribution using the TCP protocol and, sometimes, the UDP protocol.

SopCast

SopCast [35] is a P2P-TV application developed at the University of Fudan, China in 2004. Both the application and its protocol are proprietary software. This application offers a variety of TV channels, where each channels is considered an overlay network.

SwarmPlayer

SwarmPlayer [36] is a plug-in for web browsers that allows users to view live video streaming. The plug-in was developed by the P2PNext Consortium as part of the Next-PC project. At the present, Wikipedia.org has enabled the SwarmPlayer for all its video content.

uTorrent

uTorrent [37] is a tiny BitTorrent client originally created by Ludvig Strigeus and currently managed by BitTorrent Inc. This application allows the distribution of content based on the BitTorrent protocol to share video, documents, music, etc. *uTorrent* is open source.

vidTorrent

vidTorrent [38][39] is a protocol for P2P live streaming based on BitTorrent. It was developed in the Viral Communication Group by Dimitris Vyzovitis and Ilia Mirkin in 2005-2006. *vidTorrent* is an open protocol and currently is not available for its use.

2.3.2 Analysis

In our research of the different P2P content delivery applications, we focus on the study of the overlay topologies used by such applications and in the information for assigning users preferences.

Table 2.1 shows a comparison of the studied P2P applications. The ✓ and × symbols refer to whether the applications allow to configurate or not the mentioned parameter. The ? symbol refers that the documentation does not provide this information.

Overlay Topology

As shown in Table 2.1, most of the P2P applications use a mesh-shaped topology, instead of a tree-shaped topology, because of its greater robustness against churn. AnySee and vidTorrent are the only two applications that use a tree-shaped overlay.

Users Preferences

As we have mentioned above, users are the essential players in this kind of application, that is why users must be allowed to define their preferences about the way their resources will be used. As you can see in the Table 2.1, the majority of the applications only permit to change the bandwidth for downloading or uploading data. This characteristic lets users to set the bandwidth in KB/s. Although the modification of these parameters is allowed, these applications do not warn users, if their choices may cause a poor quality of service.
As well, users can determine the maximum number of connections globally and by content and they also can modify the port number for the connections.

But, do these applications allow users to define which personal information or content they want to share?, for example, let be Alice a user, she wants to download content B but she is not interested in distributing it to others, or Alice wants to share content B, but she does not want to disclose her location.

We observed that these applications do not allow users to define their preferences concerning the conditions under they wish to participate in the system. It is not possible that users can set personal data, like ID, IP, location or type of content, they want to share or not with others.

System	Topology	Bandwidth	Number of connections	Personal Data	Content
BitTorrent	Mesh	✓	✓	×	×
GoalBit	Mesh	✓	✓	×	×
PPLive	Mesh	✓	✓	×	×
PPStream	Mesh	✓	✓	×	×
SwarmPlayer	Mesh	×	×	×	×
uTorrent	Mesh	✓	✓	×	×
AnySee	Tree	?	?	?	?
SopCast	Mesh	?	?	?	?
vidTorrent	Tree	?	?	?	?

Table 2.1: A comparison of P2P content delivery applications

For AnySee, SopCast and vidTorrent, we could not determinate what they allow or not, because we could not test them and their documentation does not provide more information about it.

As we can see, none of these applications allow users to define their personal preferences. They only allow to modify parameters that affect the QoS. For that reason, we propose a service where users can define their preferences for having an impact in the user's neighborhood. In the next Chapter we describe the What Users Want (WUW) service and its components.

Chapter 3
What Users Want(WUW): a User Satisfaction Service oriented to Content Delivery Applications

In this chapter we present our proposal, in Section 3.1 we remember the problem statement and the objective of this project. In Section 3.2, we present the principles used by our service to achieve the objective proposed and finally in Sections 3.3 and 3.4, we describe our WUW service and its architecture, respectively.

3.1 Problem statement and objectives

P2P-CDA is conformed of autonomous peers which exchange data among them. In principle, peers are free to determine how long they stay in the system. Hence, their decisions determine the QoS that application offers, because without the peers participation the content distribution would not happen.

We consider that the main actors in these applications are users, it is important to allow them to specify their preferences in the system concerning content that they share, as a consequence, we expect users will be more satisfied in a system that takes in consideration their preferences. In Chapter 2, we found that any of the reviewed P2P applications allow users to define their preferences about the content, their personal data or the users with whom they are interested to interact.

Based on the problematic presented above, in this thesis we raise the following objectives:

- To design and to implement a service which considers user preferences, other than QoS related, in order to improve their satisfaction.
- To design a web interface that allows the communications between to users and the service.

This service enables users to express their preferences concerning their participation during the distribution process. Another contribution of WUW is that it measures users satisfaction and how the P2P application used considers the users preferences.

In Section 3.2, we present some notions that WUW uses and adapts in order to compute the feedback. This notions are inspired in the Satisfaction-Based Query Load Balancing framework (SQLB)[5]. However, their formal definition in WUW is essentially different [6] [7].

3.2 Principles

In order to take into account the users preferences by the P2P-CDA, users preferences are expressed as couples $p=< label, value >$. WUW maps these preferences into real numbers using a function called ***strategy***; these real numbers are called ***intentions***. **Intentions** quantify how much a peer is happy to exchange content with others.

Each user builds her own ranking of neighbors locally. WUW computes the intentions related to every neighbor for every content which is trading. Every peeer exchanges the intentions of her neighbors via a dissemination protocol. The intentions of neighbors are in the interval [-1,1].

Given that one peer is a server and a client simultaneously, WUW calculates the intentions taking into account both roles. Firstly considering the peer as a server and secondly considering the peer as a client. The results of these computations involve two different ways for ranking peers, so WUW calculates the average of the two results to obtain a definitive result.

Neighbors are scored according to their intentions and the result is list of peers. Thus, WUW gives to the local P2P-CDA a subset of the best ranked peers, in that way the user preferences will have an impact in the overlay during the content exchange.

In order to evaluate the impact that users preferences have on the P2P-CDA's job, WUW provides a feedback to users, this is composed of three measures:***Satisfaction, Adecuation and System Evaluation***. These notions, presented below, are inspired by the SQLB framework [5] as well, however the formal definitions in WUW are substantially different.

WUW needs to get some information during the content exchange from the local P2P application for computting the feedback. For this reason, we considered that a content C is split in a set of non overlapping *items* $i_1, ..., i_n$. These items are the unit of measure for computing and updating the feedback. The notation for describing the formulas of every feedback measure is presented in the Table 3.1.

In the same way that we consider the peer's intentions as server and client, WUW calculates the ***Satisfaction*** as *a client (respectively as a server)*, this notion evaluates to which extent the P2P application prefers highly ranked users over others in the actual neighborhood, *to download (respectively upload)* a given content. Intuitively, a user will be more satisfied if she downloads from or uploads to users who have the best scores, according to the

Notation	Meaning
u	A remote user
P_i	The set of users who provided the item i *
I_C^u	The local user's intention toward the remote user u *
D_i^u	Set of all download events from user u related to item i *
D	Set of all download events *
LQ_i	Set of all the request events issued by the local user related to item i *
LQ	Set of the request event issued by the local user *
RD_i^u	Set of all the complete download events to user u related to item i *
RD_i	Set of all the complete download events to remote peers related to item i *
RD	Set of all the complete download events to remote peers *
H_i	Set of all remote users who currently have item i
RQ_i^u	Set of all the request events issued by user u related to item i *
RQ_i	Set of all the request events issued by remote users related to item i *
RQ	Set of all the request events issued by remote users *

The symbol ***** implies the constraint: *since the last time the measure was computed*

Table 3.1: Notation used to describe feedback measures.

local ranking provided by her strategy and preferences. The value of satisfaction can vary between 0 and 1.

The *Satisfaction* S_c of a (local) user as a client ("downloader") is computed as follows. For each item $i \in C$ whose download has been completed, let $S_c[i]$ be the sum of the local user's intentions towards users who have provided item i, multiplied by the number of successful download events related to item i, divided by the number of times item i, or part of it, has been requested. That is :

$$S_c[i] = \frac{\sum_{u \in P_i}(((I_C^u + 1)/2) \cdot \parallel D_i^u \parallel)}{\parallel LQ_i \parallel} \tag{3.1}$$

Then S_c is the (moving) average computed by aggregating the values $S_c[i]$ over the latest downloaded items:

$$S_c = \frac{\sum_{i \in D} S_c[i]}{\parallel D \parallel} \tag{3.2}$$

The *Satisfaction* S_s of a (local) user as a server ("uploader"), is instead computed as follows. For each item $i \in C$ whose upload has completed, let $S_s[i]$ be the sum of the local user's intentions towards users who have downloaded item i, multiplied by the number of successful upload events related to item i. That is :

$$S_s[i] = \sum_{u \in RD_i} (((I_C^u + 1)/2) \cdot \parallel RD_i^u \parallel) \tag{3.3}$$

Then S_s is the (moving) average computed by aggregating the values $S_s[i]$ over the latest uploaded items:

$$S_s = \frac{\sum_{i \in RD} S_s[i]}{\parallel RD \parallel} \tag{3.4}$$

The ***Adequation*** *as a client (respectively as a server)* evaluates to which extent the downloaded pieces *(respectively uploadeded pieces)* are *provided (respectively requested)* by highly ranked users.

The *Adequation* A_c of a (local) user as a client ("downloader") is computed as follows. For each item $i \in C$ for which a request has been issued, let $A_c[i]$ be the average of the local user's intentions towards all the users who currently have item i. That is :

$$A_c[i] = \frac{\sum_{u \in H_i}((I_C^u + 1)/2)}{\parallel H_i \parallel} \tag{3.5}$$

Then A_c is the (moving) average computed by aggregating the values $A_c[i]$ over the latest requested items:

$$A_c = \frac{\sum_{i \in LQ} A_c[i]}{\parallel LQ \parallel} \tag{3.6}$$

This measure determines to which extent the preferences of the local user and her strategy are assigning higher scores to useful peers to get a given content. Its value can vary between 0 and 1, with 1 denoting the best possible choices are always made.

The *Adequation* A_s of a (local) user as a server ("uploader"), is instead computed as follows. For each item $i \in C$ whose upload has been requested, let $A_s[i]$ be the sum of the local user's intentions towards users who have requested item i, multiplied for the number of upload events related to item i. That is :

$$A_s[i] = \sum_{u \in RQ_i} (((I_C^u + 1)/2) \cdot \parallel RQ_i^u \parallel) \tag{3.7}$$

Then A_s is the (moving) average computed by aggregating the values $A_s[i]$ over the latest requested items:

$$A_s = \frac{\sum_{i \in RQ} A_s[i]}{\parallel RQ \parallel} \tag{3.8}$$

The A_s measures to which extent the local application strive to distribute those contents that are more requested by the best ranked users. Its value can vary between 0 and 1.

The ***System Evaluation*** reflects how much users can be happy with the impact of her preferences during content exchange. This measure can vary into the interval $[0, \infty]$. When this measure has a value of 1, it means a neutral impact.

The $SystemEvaluation_c$ of a (local) user as a client ("downloader") is calculated as:

$$SystemEvaluation_c = \frac{S_c}{A_c} \tag{3.9}$$

The $SystemEvaluation_s$ of a (local) user as a server ("uploader") is calculated as:

$$SystemEvaluation_s = \frac{S_s}{A_s} \tag{3.10}$$

WUW calculates intentions, peer ranking and feedback every Δ time units such that, the user is kept informed about the impact of her preferences. While any QoS related performances issue is reported by the local application, the feedback measures allow users to evaluate the quality of their neighbors (related with their preferences) and the average quality of our service itself. Following, we present a brief general description of the proposed service.

3.3 WUW's description

Since users resources are an important element in P2P-CDA, we propose WUW, a P2P service that runs in each peer[1]. To reduce the impact of WUW on a P2P applications, this service is on the top a P2P overlay.

Therefore, WUW is an intermediary between the local instance of the P2P-CDA and the P2P overlay. WUW requests information to the P2P application to know information about the events of download/upload from/to any peer during the content exchange. In Figure 3.1, we present the global view of our service.

[1]We consider that each user is represented by one and only one peer in an overlay, thus we will use the terms *peer* and *user* to refer the same participant in the system.

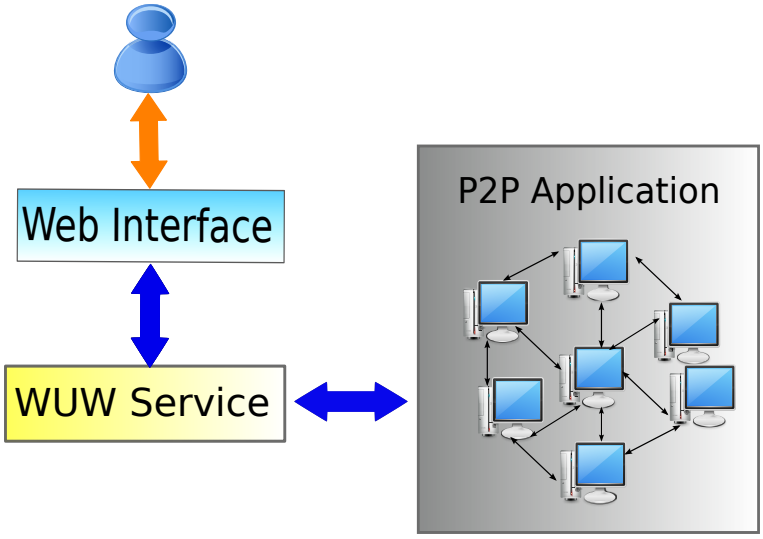


Figure 3.1: Global view of WUW.

The functionalities of WUW are:

- To get user preferences for a given content.
- To compute and update the feedback measures which allow users to measure their satisfaction and to evaluate at what degree the system considers their preferences.
- To retrieve information about the download/upload events and the neighbors.
- To calculate the local user intentions towards her neighbors for each content.
- To spread the users intentions among the peers in the system.

We describe the functional architecture of our service as follows.

3.4 WUW's functional architecture

In this section we describe the architecture of our service. WUW has different modules, each module can exchange data from other modules using well-defined interfaces. Thereby, the changes in the implementation of a single module do not require any change in the other modules. Figure 3.2 shows the general architecture of WUW. In the following subsections, we present the modules of WUW: User interface handler module, Core module, Communication module and P2P handler module.

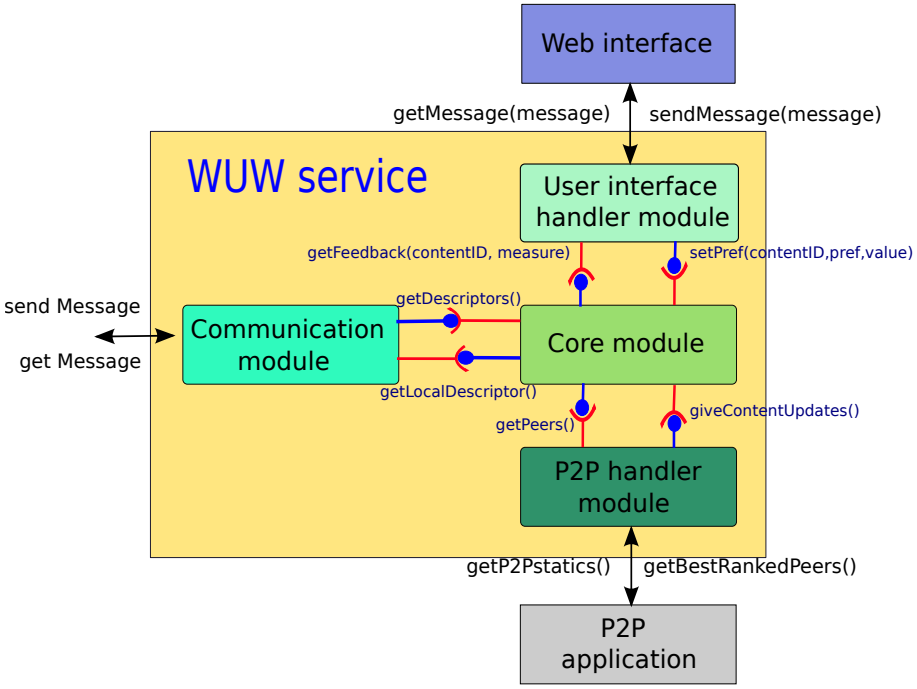


Figure 3.2: WUW's architecture

3.4.1 User interface handler module

Given that the main objective of our service is to consider the users preferences during the distribution process, our service must be able to interact with users in a simple way.
We propose an interface that lets to WUW receive the input data of users. The objective of this module is to get the user preferences and to retrieve the feedback measures
This module is independent from the other modules in WUW in order to have the possibility of using the implementation of different interfaces.

3.4.2 Core module

This module takes into acount the users preferences during the content exchange; the intentions and the feedback measures are calculated by this module. Its functionalities are:

- To compute intentions of every neighbor per content.
- To calculate the feedback measures related to the latest local activity using the neighbors intentions and the information about the content exchange.
- To update the feedback measures via the user interface handler module.

With the above, the user is kept updated about the impact that her choices have on the local computation. The feedback measures provided by WUW service allows the user to evaluate

the overall quality of her neighbors considering her preferences and the average quality of the WUW service.

3.4.3 Communication module

To calculate intentions, the feedback measures and the ranking of peers, WUW requires to know information about other peers, so this module has the task to communicate with other WUW instances which are running in other peers.

The goal is to collect information about the remote users intentions and the diffusion of any content, where the local user has a participation. All of this, without interfering in the execution of the P2P-CDA which is running on the remote peers.

3.4.4 P2P handler module

This module is the only part of the service that has knowledge about the P2P-CDA. Using this module WUW can retrieve data to compute intentions and the feedback measures. WUW needs to know information about download/upload events per neighbor and per content. Also, this module gives the P2P application a subset of the best ranked peers, which are chosen by the core module.

Chapter 4
Implementation

In this chapter, we present the technological platform used by the prototype of WUW and address the issues of integrating the different technologies in order to satisfy the objectives of the WUW service.

4.1 Technological platform

As we said before, our service evaluates a P2P-CDA based on users preferences.Our current implementation of WUW service is able to interact with a BitTorrent client (Mainline 3.9.1. version) [6]

According to Chapter 3, WUW requires to retrieve information about the P2P application, in our case BitTorrent, so we need to know how it works in order to get such information. Given that WUW needs to know which are the intentions of remote peers, we decide to use an epidemic protocol. Such protocols allow to disseminate information across a large set of peers, when there is a high dinamicity among them[40].

In Section 4.1.1 we describe how BitTorrent works. Section 4.1.2 presents the epidemic protocol used by WUW. Finally in Sections 4.1.3 and 4.2, we give details about the implementation of all the modules that compose our service.

4.1.1 BitTorrent

BitTorrent is a P2P application and the name of the P2P protocol, developed by Bram Cohen and BitTorrent Inc., for uploading and downloading files (file sharing). BitTorrent is called *Mainline* by the developers. An instance of BitTorrent is also known as a BitTorrent client.

Currently, many P2P-CDAs are based on the BitTorrent protocol. Due to its efficiency, scalability and robustness, it has become the most popular distribution protocol. On February 2009, it was estimated that BitTorrent protocol produced roughly 43% to 70% of all the traffic on Internet[41].

Applications based on BitTorrent are implemented in a hybrid P2P network. Such network uses both client-server and peer-to-peer architectures, as shown in Figure 4.1. The

centralized entity is called ***tracker***.

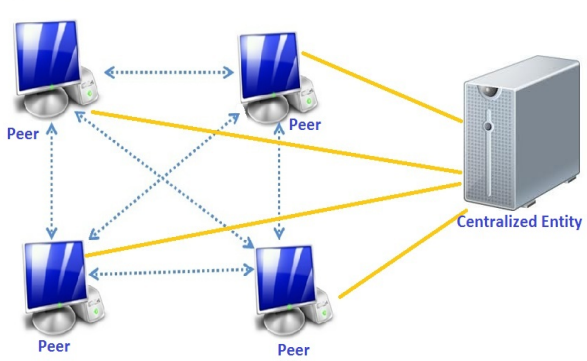


Figure 4.1: Hybrid P2P architecture

The ***tracker*** is a special node which stores meta-data about in the content exchange, it does not participate in the content distribution, it just enables peers to find each other by using the HTTP protocol. Peers interested in a content are organized into overlays network, called ***torrents***. Peers which form part of a *torrent* can be connected or not. One *torrent* is build per content. The connected set of peers in the torrent participate in the content distribution and they are referred as a ***swarm***.

The peers which are part of a *torrent* can be classified into two types: ***seeders*** and ***leechers***. A *seeder* has a complete copy of the file and its job in the torrent to serve other peers. For a torrent to get started, we need at least one *initial seeder* that provides the entire content for download. A *leecher* is interested on downloading the file.

For content distribution are using *swarming techniques*. In these techniques the content is split into fixed size pieces, typically 256 KB, and each piece is divided into sub-pieces, typically of 16 KB in size, which are called ***blocks*** or ***chunks***. When one content is available for its distribution through the P2P network, it has a ***descriptor***, also known as ***torrent file***. This file contains information about the content: length, name, number of chunks, size of chunks and the URL of the tracker.

As shown in Figure 4.2, there are 3 steps for being involved in the downloading [20]:

- **Step 1. Getting the File Descriptor.-** When $peer_i$ wants to download a $content_j$, it obtains the corresponding descriptor from a web server.

- **Step 2. Announcement.-** $Peer_i$ communicates with the tracker and asks for the peers that are involved in the distribution of $content_j$. Then, the tracker sends a list of peers to $peer_i$, this list is a subset of the *swarm*. This list typically consists of at most 50 peers chosen in a randomly way from the *swarm*.

- **Step 3. Exchange of chunks.-**$Peer_i$ selects some peers, they will become its *neighbors* and a request of chunks from the $content_j$ is sent to each peer in the neighborhood. When $peer_i$ obtains a new chunk, it informs to its *neighbors* the availability of that chunk. $Peer_i$ downloads pieces not only from the seeders, but also from the leechers, in that way seeders reduce their workload.

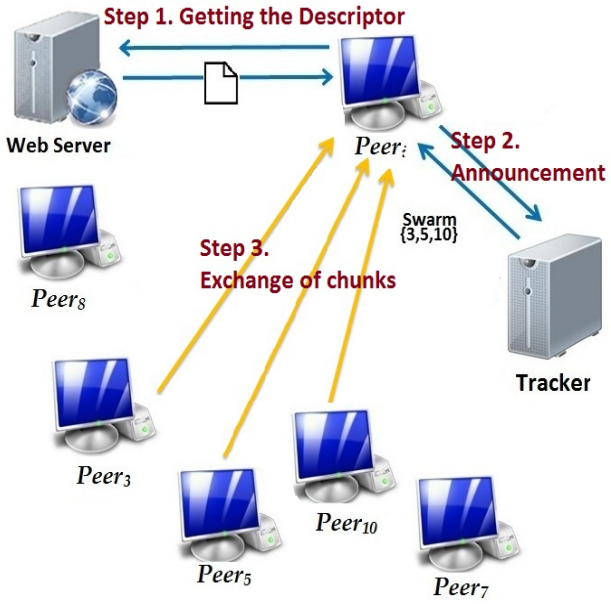


Figure 4.2: Content download

BitTorrent makes use of some principles defined in Chapter 3, the downloading process uses a mesh-shaped overlay where any peer can act as a server or as a client simultaneously.

For the pieces exchange, BitTorrent uses two key strategies, chunk and peer selection. The strategy for chunk selection is a modification of the Least-Shared First algorithm which is known as the *Rarest First Algorithm*.

The rarest first algorithm works as follows. Each peer in the swarm maintains a list of the number of copies of each piece in its neighborhood. This information lets define a rarest pieces set. The rarest pieces set of a peer is updated when a new piece is added or removed from its neighborhood. Each peer selects the next piece to download in its rarest pieces set, in a randomly way.

Additionally, the rarest first algorithm comprises the next three policies [19][30]:

- ***Random First Policy.-*** When a peer has downloaded less than 4 pieces, it chooses randomly the next piece to be requested. The aim of this policy is to permit a peer to

download its first pieces as soon as possible. This is important because it is necessary to have some pieces for sharing.

- ***Strict Priority Policy.-*** When at least one chunk of a piece has been requested, the other chunks of the same piece are requested with the *highest priority*. The aim of this policy is to complete one piece as soon as possible.

- ***End Game Mode.-*** When a content is almost complete, peers request every non received block to their neighbors. Each time a block is received, the pending requests are cancelled. This policy accelerates the end of the download.

One peer, as we said before, can be seeder or leecher and it must maintain state information for each connection that it has with a remote peer. Considering A and B as peers the possible states in a BitTorrent are:

- ***Interested:*** A is interested in B, if B has pieces that A does not have.

- ***Not interested:*** A is not interested in B, if B only has a subset of the pieces of A.

- ***Chocked:*** A is choked by B, when B decides not to share chunks with A.

- ***Unchoked:*** A is unchocked by B, when B decides to share chunks with A.

Peer can be in the next situations:

1. *Choked/Interested:* The peer is choked and interested.

2. *Unchoked/Interested:* The peer is unchoked and interested.

3. *Choked/Not interested:* The peer is choked and not interested.

4. *Unchoked/Not interested:* The peer is unchoked and not interested.

BitTorrent uses as peer selection strategy, the Bandwidth Aware scheduler (BAWp). Such strategy is included in its *Choke Algorithm*, with the aim to penalize peers that never upload pieces. These peers are known as *free riders*.

Below, we describe the Choke algorithm from the point of view of the local peer. In order to facilitate the understanding of such algorithm, we use the state *interested* for saing that a remote peer is interested in the local peer, and the state *choked* for saing that a remote peer is choked by the local peer.

The Choke algorithm differs between the two types of peer: leecher and seeder. Below, we describe the choke algorithm in the two cases [19][30]:

- **Choke algorithm (when the local peer is a leecher)**

  The local peer can serve at most 4 remote peers, i.e. at most 4 remote peers can be in the state <Unchoked/Interested>.

  1. Every 10 seconds, the interested remote peers are sorted according to their download rate[1] and the 3 fastest peers are unchoked.
  2. Every 30 seconds, one additional interested remote peer is unchoked at random. This step is called *The Optimistic Unchoke*. This peer selection has two purposes, firstly to evaluate the download capacity of new peers in the peer set and secondly, to allow to new peers obtain their first piece.

- **Choke algorithm (when the local peer is a seeder)**

  At most 4 remote peers can be in the state <Unchoked/Interested>.

  1. Every 10 seconds, the interested remote peers are sorted according to their upload rate[2] and the 3 fastest peers are unchoked.
  2. Every 30 seconds, one additional interested remote peer is unchoked at random.

As we mentioned above, one peer is willing to join the swarm, and retrieves the descriptor of the content, then it contacts the tracker by sending an *announce* message through a HTTP request. The parameters for this request are [29]:

- ***info_hash:*** ID of the content (20-byte SHA1), this is the same ID that we can find into the torrent file.
- ***peer_id:*** ID of the peer, generated by the client at start-up.
- ***port:*** The port number where the client is listening on.
- ***event:*** This field can take the following values:
  - *started:* If the peer sends the first request to the tracker.
  - *completed:* If the peer shuts down BitTorrent.
  - *stopped:* If the download is complete.

  This information helps the tracker to keep overall statistics about the torrent (e.g., number of seeders, number of leechers, life time of the seeders, etc.)
- ***uploaded:*** The total amount of uploaded bytes since the peer sends the *started* message to the tracker.

[1] The download rate is the speed at which data can be downloaded from a remote peer to the local peer
[2] The upload rate is the speed at which the data can go from the local peer and be send to a remote peer.

- ***downloaded:*** The total amount of downloaded bytes since the peer seds the *started* message to the tracker.
- ***left:*** Number of bytes that the peer still has to download.
- ***compact:*** If the value of this field is 1, the peer accepts a compact response. The announce-list is replaced by a string of 6 bytes per peer (4 for IP and 2 for port).
- ***IP (optional):*** The peer can send its IP address, it is optional because it is possible to identify it.
- ***numwant (optional):*** Number of peers that the client would like to receive from the tracker. By default it is 50 peers and it is possible to define zero peers.
- ***tracker_id:*** When the tracker receives the announce message, it can return a tracker identifier. If this happens, the client should include this value in the announce.

The tracker responds with a *text/plain* document, which includes a set of peers that are currently on-line. The information in this document is:

- ***interval:*** Time in seconds that the peer should wait to send requests to the tracker.
- ***tracker_id:*** A string that the client should send back in its next announcements.
- ***Completed:*** Number of seeders.
- ***Incomplete:*** Number of leechers.
- ***Peers list:***
  - *Dictionary mode (compact=0):* For each peer the tracker sends:$<$ *peer_id* $><$ *IP* $><$ *port* $>$
  - *Binary mode (compact=1):* String with 6 bytes per peer, 4 for IP and 2 for port.

The communication between peers is done by sending message over the TCP protocol. The range of TCP ports used by BitTorrent clients is [6881-6999]. The peers exchange messages about the contextual information and about the pieces they have. In Table 4.1, we describe the different messages exchanged between peers [42][43].

One peer knows the pieces that its neighbors have, so it is able to decide if it is interested or not in downloading from them. The messages are shown in Table 4.2 [42][43].

A block is download/upload by a peer if it satisfies the following conditions:

1. One block is downloaded by peer A when A is interested in a remote peer B, and B is not choking to A.
2. One block is uploaded by peer A when A is not choking a remote peer B, and B is interested on A.

It is important for the peers to keep their neighbors informed when they are in the interested state. This information should be kept up-to-date for each neighbor even if the peer is choked. This allow neighbors to know if the peer will download when it is unchoked (and vice-verse). In Figure 4.3, we present a state diagram of a peer, the initial state is Choked/Not interested.

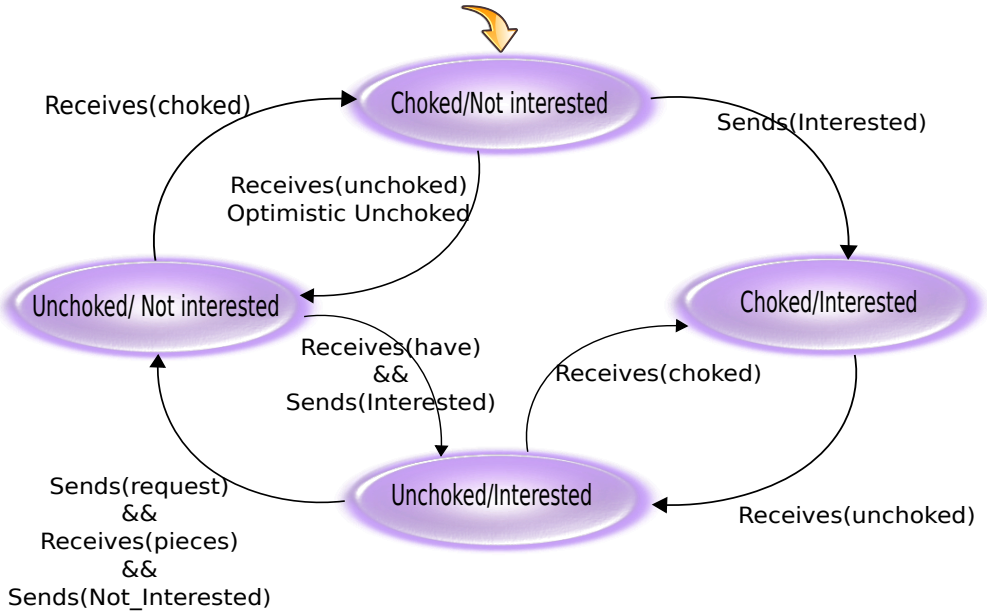


Figure 4.3: State diagram of a peer

Figure 4.4 shows an example of a possible message exchange between two peers. In the example, the connection is established after peer A sends an *handshake* message, and peer B responds with one as well. Initially, both peers are in the *choked/not interested* state.

Once the connection is established, peer B sends a *bitfield* message to peer A, and A sends an *interested* message. In this moment, the state of peer A changes to *choked/interested*. Peer A has no pieces, so it does not send the *bitfield* message. Then peer B sends a *not interested* message to A and, peer A sends a *choked* message to peer B.

Therefore, data is flowing from peer B to peer A because when the peer B receives the *interested* message from peer A, an *unchoked* message is sent to peer A. Now A is in *unchoked/interested* state. Finally, A requests a chunk of a particular piece sending a *request* message, and B responds with the *piece* message which contains the requested chunk.

Moreover, the peer A requests all the chunks that form one piece. Once A obtains one chunk, it sends a *not interested* message to peer B which responds with a *choked* message. So, the state of A is *choked/not interested*. When A has a completed piece, it sends *have* messages to all its neighbors.

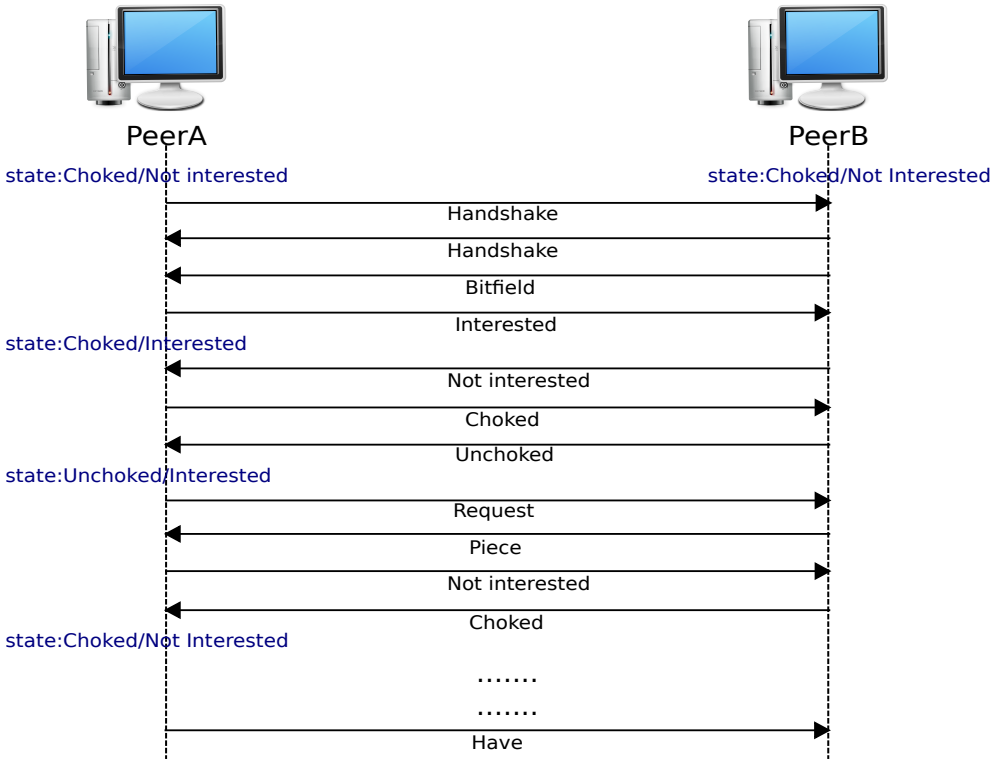


Figure 4.4: Example for changes of state and the message flow between two peers

Name	Format	Description
Handshake	$< pstrlen >< pstr >< reserved >< info\_hash >< peer\_id >$ • ***pstrlen:*** Length of *pstr* parameter • ***pstr:*** String identifier of the protocol • ***reserved:*** 8 bytes reserved for the protocol, each of these defines the behavior of the protocol. • ***info_hash:*** ID of the content • ***peer_id:*** ID of the peer, this is the same that in the *announce*.	The connection between two peers starts with this message, this creates a TCP connection between the peers. The peer A sends a Handshake to the peer B, if the peer B accepts the connection, then peer B has to send other Handshake as response.
Bitfield	$< len >< id = 5 >< bitfield >$ • ***len:*** Length of the bitfield. • ***id:*** Id of the message Bitfield, it is 5. • ***bitfield:*** The payload is a bitfield representing the pieces that have been successfully downloaded.	This message may only be sent immediately after the handshaking sequence is completed and before any other messages are sent. With this message, one peer knows the pieces that other peers have.
Have	$< len >< id = 4 >$<piece index> • ***len:*** Length of the message. • ***id:*** Id of the message, it is 4. • ***piece index:*** Index of the piece.	This message is sent when a peer receive a new piece. The payload is the index of the piece that has just been successfully downloaded.
Keep_alive	$< len = 0000 >$ • ***len:*** Length of the message.	This is a message with zero bytes, specified with the length prefix set to zero. Peers may close a connection if they receive no messages (keep_alive or any other message) for a certain period of time. This amount of time is generally two minutes.

Table 4.1: Messages exchange for sharing pieces

Name	Format	Description
Choked	$< len = 0001 >< id = 0 >$ • ***len:*** Fixed-length of this message • ***id:*** ID of the message, it is 0.	This is sent by a peer to inform one of its neighbors (remote peer) that it is choked and will not receive any data. i.e. the remote peer is choked. This message follows a peer selection.
Unchoked	$< len = 0001 >< id = 1 >$ • ***len:*** Fixed-length of this message • ***id:*** ID of the message, it is 1.	This is sent by a peer to inform one of its neighbors (remote peer) that it is unchoked and will receive any data. i.e. the remote peer is unchoked. This message follows a peer selection. When the peer A receives a *not_interested* message from a peer B, peer A sends a *choked* message to peer B. Thus, data will not flow from peer A to peer B until both messages are replaced.
Interested	$< len = 0001 >< id = 2 >$ • ***len:*** Fixed-length of this message • ***id:*** ID of the message, it is 2.	This message is sent by a peer to inform one of its neighbors (remote peer) that it is interested in one of its pieces. This message follows a *bitfield* or a *have* message.
Not interested	$< len = 0001 >< id = 3 >$ • ***len:*** Fixed-length of this message • ***id:*** ID of the message, it is 3.	This message is sent by a peer to inform one of itsneighbors (remote peer) that it is not interested in any of its pieces. This message follows a *bitfield* or a *piece* message.
Request	$< len = 0013 >< id = 6 >< index >< begin >< length >$ • ***len:*** Fixed-length of this message • ***id:*** ID of the message, it is 6. • ***index:*** Integer specifying piece index • ***begin:*** Integer specifying byte offset within the piece • ***length:*** Integer specifying the requested length	This message is sent by a peer to request a given data chunk; this message follows an *unchoked* message
Piece	$< len >< id = 7 >< index >< begin >< block >$ • ***len:*** Fixed-length of this message • ***id:*** ID of the message, it is 7. • ***index:*** Integer specifying piece index • ***begin:*** Integer specifying byte offset within the piece • ***block:*** Chunk of data, which is a subset of the piece specified by index.	This message is sent in response to a *request* message and contains the requested chunk
Cancel	$< len = 0013 >< id = 8 >< index >< begin >< length >$ • ***len:*** Fixed-length of this message • ***id:*** ID of the message, it is 8. • ***index:*** Integer specifying piece index • ***begin:*** Integer specifying byte offset within the piece • ***length:*** Integer specifying the requested length.	This message is used to cancel block requests. The payload is identical to that of the *request* message. It is typically used during *End game mode*.

Table 4.2: Messages exchange for sharing contextual information

4.1.2 Epidemiological protocol: Newscast

The design of protocols and models that address the problem caused by information exchange in P2P systems, has been one of the most active research areas in recent years[40]. The principle of this dissemination is based on imitating the expansion of epidemics. These algorithms reproduce how the infectious diseases or rumors are spread in a society, where an infected individual passes the germs or the rumor to others, keeping a direct or indirect contact.

Analogically in a distributed system, new information is randomly distributed to other nodes, and these nodes reproduce the same cycle, considering that only one single server has to transmit such information.

Newscast is an epidemiological protocol, its aim is to exploit randomness to disseminate information across a large set of nodes, maintaining these nodes connected without using any static structure or any kind of administration [40].

The basic idea of the Newscast protocol is that each peer maintains a cache containing *c Peer Descriptors (see Figure 4.5a)*. Each *Peer Descriptor* contains the peer address, a logical timestamp representing its creation time and all the information that must be spread. Periodically, a peer A must:

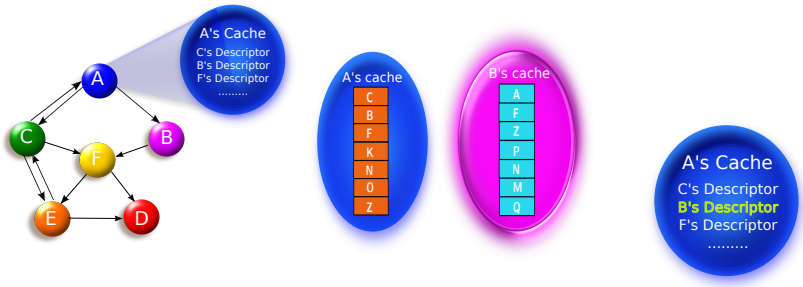
1. To select a random peer B from its local cache as shown in Figure 4.5b.
2. To updates its local timestamp.
3. To exchange its cache with B *(See Figure 4.5c)*. The exchange involves sending the cache of A with its ID and receiving the cache of B.

After the exchange, each peer merges both caches and keeps the most fresh IDs using the timestamp associated with each ID, as shown in Figure 4.5d. The only requirement imposed to a cache is that no multiple copies of the same ID are allowed.

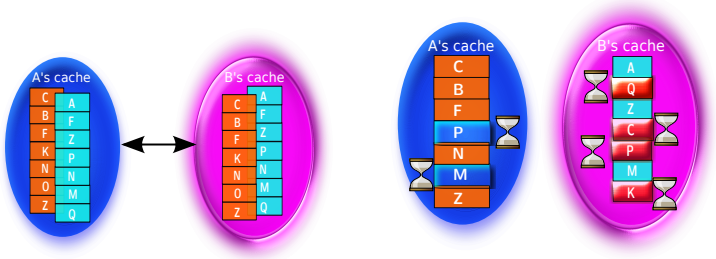
4.1.3 Implementation Decisions

In Chapter 3, we described the functional architecture of our service, as we mention in this chapter, WUW receives the preferences from the users. For this purpose, we implement a module which gets the users preferences and provides a feedback to users. In our current implementation, WUW features a web-based interface, this interface interacts with the User interface handler module.

WUW is a service developed in Java which consists of a set of modules where each one complies a specific task. The web interface lets users to specify their preferences that WUW uses during the distribution process.



(a) Representation and overview of caches (b) Peer selection



(c) Exchange of caches (d) Update caches

Figure 4.5: Phases of Newscast protocol

LiveConnect [44][45] is a technology which allows intercommunicate Java and JavaScript through a Web page. LiveConnect allows JavaScript programs to interact with Java applets [3]. Also LiveConnect allows to invoke public methods of applets from JavaScript programs. So, we decide to use LiveConnect in order to lets WUW gets the users preferences from the web page.

4.2 Implementation Description

In this section we present how each module of WUW was implemented and we describe the methods used in each module.

[3] A Java applet is a little application written in Java that performs interactive animations, immediate calculations, or other simple tasks. Usually, Java applets are embedded in web pages and can be run directly from a browser.

4.2.1 User interface handler module

In order to facilitate the interaction between the users and WUW, we designed a web interface which is easy-to-use. Through this interface, users can specify their preferences and shows the values of the feedback measures of WUW (See Section 3.2).

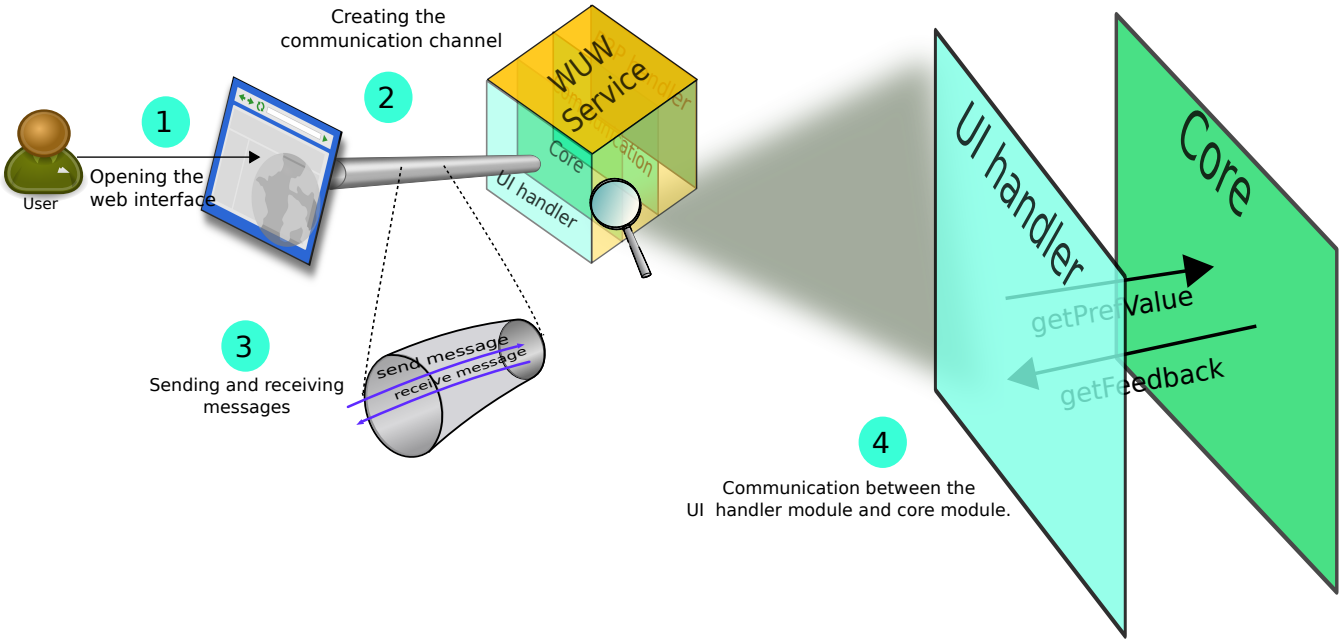


Figure 4.6: Communication between the user and WUW

As shown in Figure 4.6, the user opens the web interface, once the web interface is charged, the channel communication is created. So, it is possible to send and receive messages between the web interface and the User interface handler module of WUW.

The User interface handler module was implemented with the objective of allow WUW gets the preferences specified in the web interface. Subsequently, the core module considers these preferences during the distribution process and then the feedback measures are shown.

The information exchange between the web interface and the WUW service is carried out through an invisible applet that runs in the web interface. User interface handler module implements the following methods:

- **void initPref**(*String contentID, Preference[ ] Pref*);

  Description
  Initializes the preferences related to a content.

Input Parameters

contentID
The ID of the content to which the preference refers.
Pref
Set of preferences related to a content.

Output Parameters

- - -

- **void setPref**(*String contentID, String pref, Object value*);

  Description
  Sets a preference to a given value.

  Input Parameters

  contentID
  The ID of the content to which the preference refers.
  pref
  The name of the preference to be set.
  value
  The value the preference is to be set to.

  Output Parameters

  - - -

- **Object getPrefValue**(*String contentID, String pref*);

  Description
  Retrieves the value of a given preference.

  Input Parameters

  contentID
  The ID of the content to which the preference refers.
  pref

The name of the preference to be set.

Output Parameters
Returns the value of the preference.

- **void initFeedback**(*String contentID, String[] measures, double[] values*);

  Description
  Initializes the measures displayed to the users as a feedback of the ongoing computation.

  Input Parameters

  contentID
  The ID of the content to which the preference refers.
  measures
  The names of the measures.
  values
  The initial values of the measures, whose order matches with the order of occurrence of their name in the *measures* parameter.

  Output Parameters

  - - -

- **void setFeedback**(*String contentID, String measure, double value*);

  Description
  Sets a single feedback measure to a given value.

  Input Parameters

  contentID
  The ID of the content to which the preference refers.
  measure
  The name of the measure to be set.
  value
  The value to be set.

  Output Parameters

  - - -

- **double getFeedback**(*String contentID, String measure*);

  Description
  Gets a single feedback measure to a given value.

  Input Parameters

  contentID
  The ID of the content to which the preference refers.
  measure
  The name of the measure to be set.

  Output Parameters
  Returns the value of the measure specified.

These methods allow to initialize each content with the set of preferences and the set of measures of feedback. Moreover, the methods to set a specific value in one of the preferences or measures of feedback were implemented, as well as the methods to get the current value of each one.

During the communication with WUW, the web interfaces uses the next methods:

- **function getElementJava()**

  Description
  This function gets a reference to the applet object which is loaded in the web interface in order to allows the access to its methods and properties for establish the channel communication.

  Input Parameters
  - - -

  Output Parameters
  Returns a reference to the applet which is a HTML element in the web page.

- **function appletReady()**

  Description
  This function informs if the applet is ready to use, i.e. when the applet was charged, it sets a flag with a true or false value depending on the state of the applet.

Input Parameters
- - -

Output Parameters
- - -

- **function connectSocket(address, port)**

  Description
  This function creates a local channel communication through a socket using an specific port.

  Input Parameters
  address
  IP address of the machine to which it connects.
  port
  The port number to which it connects.

  Output Parameters
  - - -

- **function disconnectSocket()**

  Description
  This function closes the local channel communication.

  Input Parameters
  - - -

  Output Parameters
  - - -

- **function sendMessage(message)**

  Description
  This function allows to send messages through a socket, messages will contain the preferences set by the users.

Input Parameters
message
The information that is send to the WUW service from the web page.

Output Parameters
- - -

- **function getMessage(message)**

  Description
  This function receives the message sent through the socket by the WUW service, it contains the feedback.

  Input Parameters
  message
  The information that is received from the WUW service.

  Output Parameters
  - - -

These functions lets to send the users preferences and receive the feedback measure that WUW calculates. All these information is structured as a string that follows a pattern with the objective that the functions or methods can retrieve only the valuable information for them.

4.2.2 Core module

This module takes into consideration users preferences during the content exchange. The core module needs to obtain the preferences in order to calculate the intentions. Then this module gets the intentions of remote peers via the Newscast protocol through the communication module.

This module considers the definitions presented in Section 3.2 to calculate the feedback measures and the ranking of peers.During the content exchange users can see how the feedback evolves via the web interface. See Figure 4.7.
The core module performs its tasks using the following methods:

- **void computeIntentions()**;

  Description
  This method is the implementation of what we call a strategy. It gives as a result the user intention. In our current implementation, the strategy only takes into account the interest of the user in the content which is captured through the web interface.

  Input Parameters
  - - -

  Output Parameters
  - - -

- **void updateFeedback()**;

  Description
  This method computes the feedback measures. Such measures are computed in base to the formulas presented in Section 3.2.

  Input Parameters
  - - -

  Output Parameters
  - - -

- **void getEpidemicUpdates** (*Object upd[]*);

  Description
  This method gets the updated PeerDescriptors of the remote peers.

  Input Parameters

  upd
  Descriptors to get.

  Output Parameters
  - - -

- **void buildGlobalRanking()**;

  Description
  The global ranking of peers is build considering the intentions per neighbor and per content. Every neighbor is scored based in the average of the linear combination between intentions of the local peer and the intentions of remote peers.
  These scores are added together and then the average (global score) is calculated for each neighbor. Then the list of the scored neighbors is sorted by score. Finally the best score peers per content are given to the P2P application.

  Input Parameters
  - - -

  Output Parameters
  - - -

4.2.3 Communication module

In order to calculate intentions, feedback and ranking peers, WUW requires to communicate with other WUW instances. Such communication allows to collect information about intentions of remote peers and the diffusion of any content where the local user is participating. See Figure 4.8

The communication module implements the basic facilities to send and receive messages over TCP or UDP using the Newscast protocol. (See Section 4.1.2.

All the peers contribute in the dissemination of the information, but each peer uses and keeps in its memory only the information that concerns the local user. One peer can update only the information that concerns the local user, because it only has access to the information identified by its ID. So, it is not allow to change the remote users information.

With the dissemination of up-to-date information it is possible that each local user knows which are the intentions of remote users towards her and what is the state of the diffusion of a given content. The communication module uses a Java interface that implements the following methods.

- **int addMsgHandler** (*MsgHandler mh*);

  Description
  This method is called in order to any object uses the message delivery service.

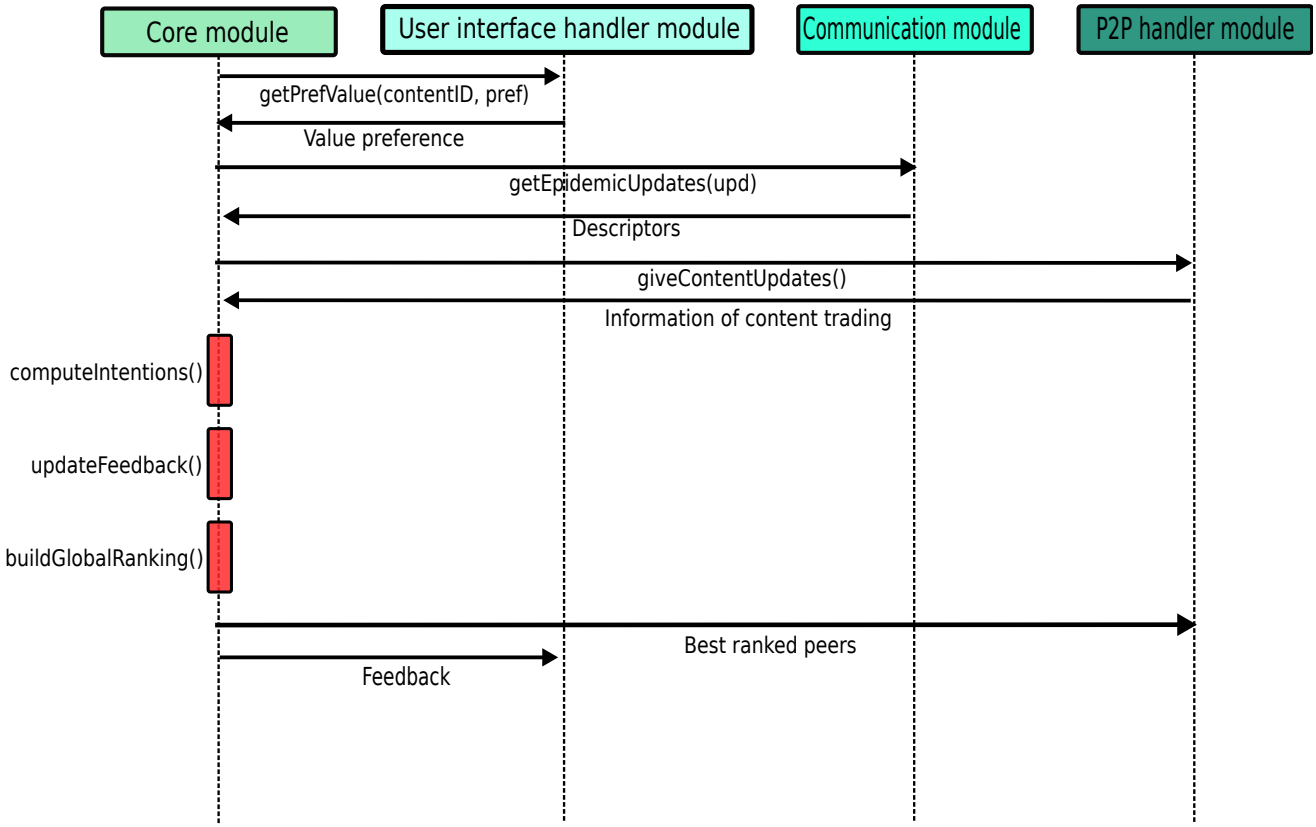


Figure 4.7: Sequence diagram to calculate feedback and ranking peers.

Input Parameters

mh

The receiver object that can handle the message

Output Parameters
Return the message ID.

- **void send** (*PeerID dest, int mid, Object msg*);

Description
Sends message *msg* to peer *dest*.

Input Parameters

dest

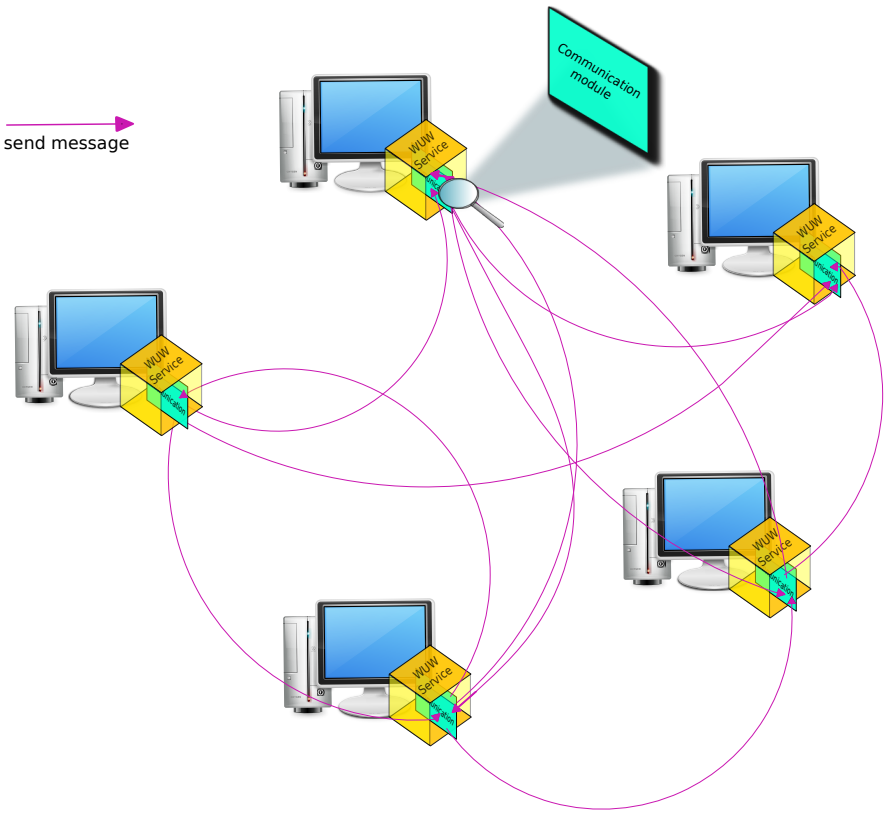


Figure 4.8: Communication between remote WUW services

The destination peer

mid

The ID of the message, as returned to the caller by addMsgHandler

msg

Message to be sent

Output Parameters

- - -

- **void dispatch** (*TMessage msg*);

Description

By calling this method whenever a message is received, the payload will be dispatch to the proper handler.

Input Parameters

msg
The message to be dispatch to the proper handler.

Output Parameters
- - -

4.2.4 P2P handler module

This module establishes a communication with the local P2P application, in our case BitTorrent. WUW needs to obtain data from BitTorrent in order to calculate the feedback.

For each content and for each neighbor in the overlay, WUW needs to know about any download/upload event from/to each neighbor. The basic data that WUW needs to send to BitTorrent is the peer list related to a given content. See Figure 4.9.

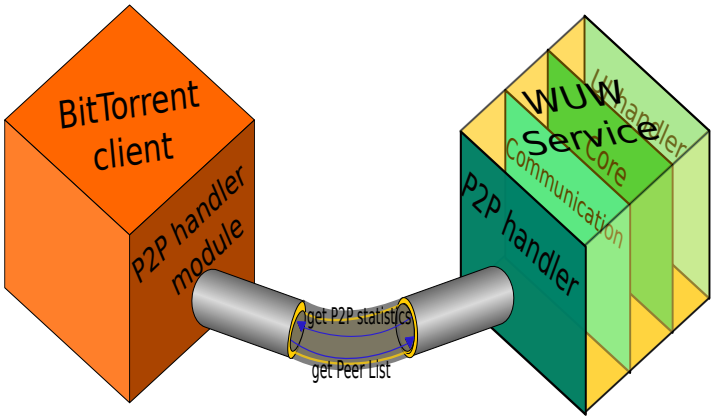


Figure 4.9: Communication between BitTorrent client and WUW service

The P2P handler module is able to send and retrieve data from BitTorrent, so that, we instrumented BitTorrent for getting that information periodically collected, via a local socket from the BitTorrent process. The methods used for this purpose are the following:

- **Transaction[ ] giveContentUpdates** ( );

  Description
  This method retrieves information about the download/upload events. This is the way WUW knows the details about content exchange.

Input Parameters
- - -

Output Parameters
Return the latest updates about the content trading on the local peer, or null if no update is available.

- **void getPeers**(*String contentID, PeerID[ ] peers*);

Description
This method gets the peers from WUW core and makes them available to the BitTorrent client updating its local neighborhood.

Input Parameters
contentID
The ID of the content which the peer list must be associated to.
peers
The peer list for the given content.

Output Parameters
- - -

- **BtWuwPeer**[ ] **getBestRankedPeers**( );

Description
This method gets an array of the current best ranked peers with the objective that the list of peers is sent to the local BitTorrent instance.

Input Parameters
- - -

Output Parameters
Array of the best ranked peers.

- **BtWuwPeer**[ ] **getPeersForAnnounce**( );

Description
This method gets an array of peers for answering the first announce message which has been sent by the local BitTorrent instance.

Input Parameters
- - -

Output Parameters
BitTorrent peer list for the first announce message.

Figure 4.10 shows a general way of the download process. Users willing to download content set their preferences through the web interface. These preferences are sent to WUW service. Then, the torrent file is requested to the web server which sends the torrent file to the P2P client, in our case BitTorrent.

WUW obtains the peer list from the tracker. WUW communicates with other WUW instances in order to obtain their descriptors through the Newscast protocol. After procesing all the information, WUW selects the best ranked peers and sends these peers to the P2P client. From here, the download process is performed as usual.

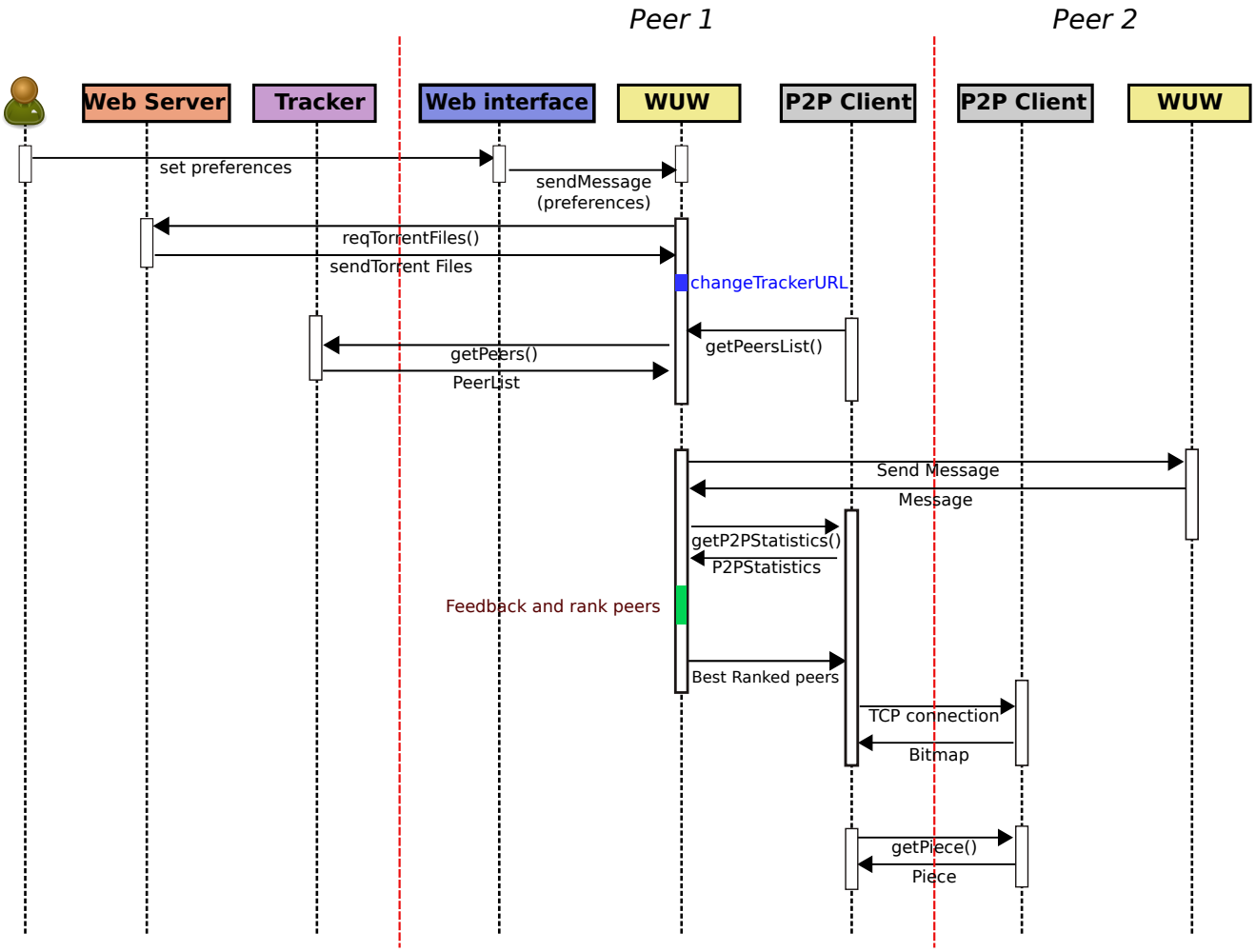


Figure 4.10: Sequence diagram to download content

Chapter 5
Evaluation

In this chapter, we present the evaluation of the impact of our service in the average download time of a content in BitTorrent. In the section 5.1, we present our testbed used, thereafter we present our experiment made, which was in charge of the Engineer Raziel Carvajal G. In section 5.3, we present the demostration of our web interface.

5.1 Experimental platform

In order to evaluate the impact of WUW, we carried out an experiment using the French tesbed **Grid5000**.

Grid5000 [46] borns as a project with the aim to build an experimental platform gathering 8 different sites, which are geographically distributed in France. The main objective is to offer an experimental testbed for research purposes. Grid5000 has the participation of 17 laboratories, nation wide. On April 5th 2011, 7244 cores distributed among 1500 nodes were available to the community, they had access to all the software layers between the network protocols and the applications. Grid5000 offers to the research community a dedicated platform where they can run their experiments.

5.2 Experimentation

This experiment has the objective of measure the impact of WUW service in the average download time.

We performed our experiment using 100 nodes, where each node represents one peer and each of them runs a BitTorrent process and a WUW process. The 10% of the nodes are *seeders* and 90% of them are *leechers*, and each node has a 256 Kb of bandwidth.

Every peer shares a content shared of 112MB, and we use the interest of the peer as a preference. So, we define that the 50% of peers have a ***primary*** interest, 30% have a

negligible interest, 20% have a ***unknown*** interest on the shared content.

With respect to BitTorrent process, each process update its peer list every 40 seconds, the Newscast protocol selects a peer to exchange information every 2 seconds in randomly way, and WUW service compute every 8 seconds the feedback measures and the ranking the peers.

To carry out the fulfillment of the objective of our experiment, we separate the experiment in two cases. The first case consists on running a BitTorrent (BT) without any modification and exist the presence of one tracker. In the second case, each node runs a BitTorrent and a WUW instance (BT+WUW).

For this experiment, we instrumented the BitTorrent with a class that collects some statistics about the content. WUW intercepts the messages between the peers and the tracker.

Instead, we implemented a tracker emulator that choses randomly the list of peers sent to each leecher, this list is intercepted by WUW service which modify this list and include the peers more interesting and excluding the less interesting ones from point of view of the user preference.

As we can see in the Figure 5.1 the red line shown that a leecher in average obtains the whole content in 384.21 seconds using the Bittorrent client. On the other hand, the blue line shown that a peer that use the BitTorrent client and the WUW service in average it got the content in 364.02 seconds.

These result shown that the use of BitTorrent and WUW has an improvement of 20 seconds i.e. 5.1% with respect to download the content using only the BitTorrent client. We consider that this improvement is due that in the download process exist less external communication with other entities, that not participate in the download, owing the absense of the tracker.

Whereby, the peers communicate only between them, even with the presence of the WUW service, the most of tasks that WUW makes involve a interprocess communication as the collection of statistics of BitTorrent, where both instances (WUW and BitTorrent) run in the same node.

5.3 Web interface demostration

In this section, we present a demostration of our web interface which through the users can define their preferences. In this demostration, we consider that a user is interested in a specific content and she wants to download it, using our WUW service. In Figure 5.2, we present the use case diagram implemented in the web interface.

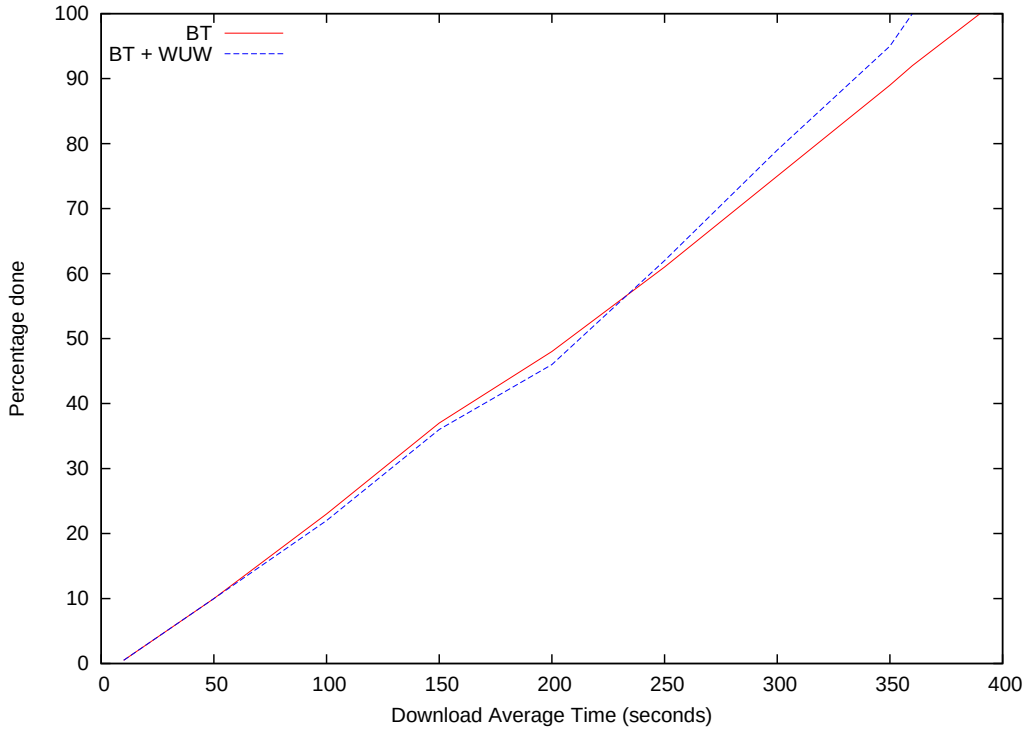


Figure 5.1: Performance of WUW in terms of average download time.

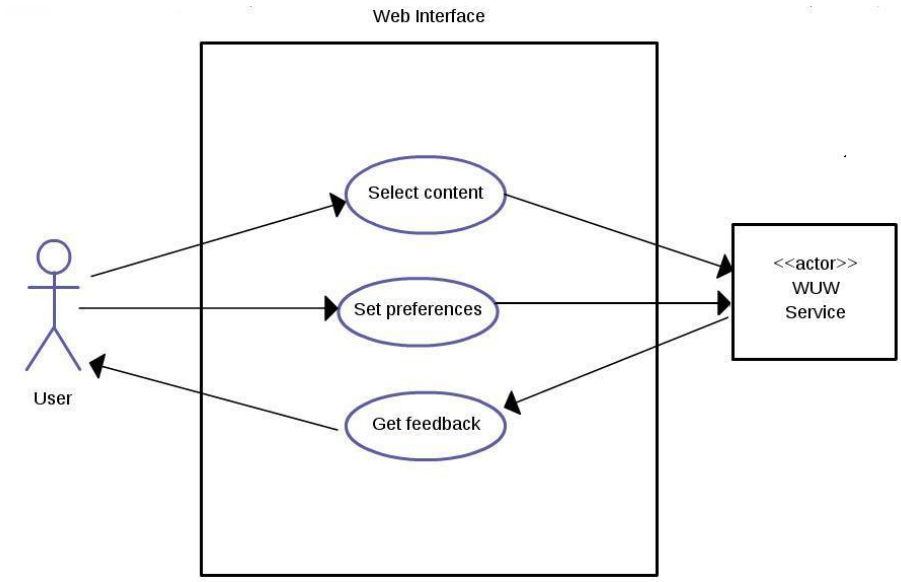


Figure 5.2: Use case diagram of the web interface

The web interface has three use cases: select content, set specification and get feedback. In Figure 5.3, we see that after to select a content (Figure 5.3a), it is shown the fields

related to the content selected (Figure 5.3b). The selection of a content implements the use case presented in Table 5.1.

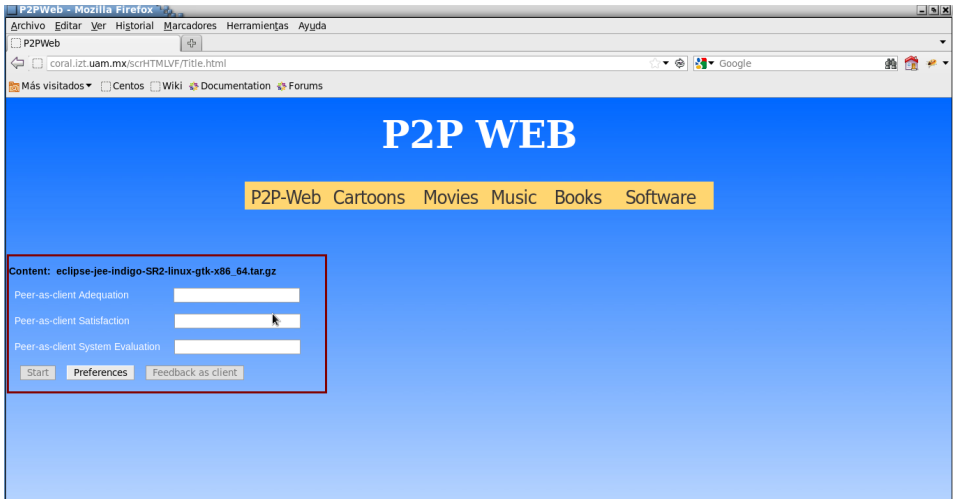
Name	Content selection
Objective:	Select a content to be downloaded
Actor	Users
Precondition	Open the web interface.
Sequence of interactions	User-Web interface
1.- The user selects a content among the different categories. 2.- The web interface shows the fields related to the content that the user wants to download.	
Postcondition	The web interface shows the name of the selected content. Also the web interface shows the feedback and the buttons of start, delete and preferences.

Table 5.1: Use case specification 1: Content selection

Name	Setting preferences
Objective	Allow users to define their prefereces related to selected content.
Actor	Users
Precondition	User selects a content.
Sequence of interactions	User-Web interface
1.- The user clicks on the preferences button. 2.- The web interface displays the area of preferences, this area presents differents preferences related with the selected content. 3.- The user defines the value of each preference. 4.- The user clicks on the send button. 5.- The web interface sends the preferences to the local WUW instance.	
Postcondition	The web interface sends preferences to the WUW service.

Table 5.2: Use case specification 2: Setting preferences

(a)



(b)

Figure 5.3: Content selection

In Table 5.2, we present the use case that our web interface implements for users can set their preferences about the content to be downloaded. Figure 5.4 shown the pop-up window that web interface shows.

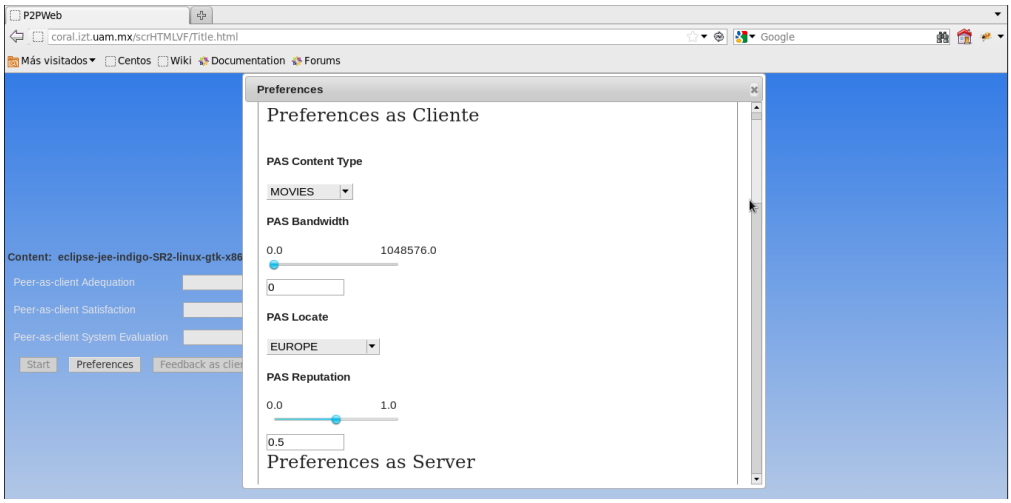


Figure 5.4: Pop-up window with the user preferences

Table 5.3 shows the last use case that the web interface implements, which name is *Getting feedback*, in Figure 5.5, we presents the feedback that WUW service sent to the web interface when the user is a client, i.e. when the user downloads a content. As we can see, in the right of the screen, a chart is generate with the values of the feedback that WUW calculates. In the same way, Figure 5.6 presents the feedback when the user is a server, i.e when the user uploads a content. This feedback is display when user clicks on the button *Feedback as server*.

When the user clicks on either of the two previous charts, a pop-up windows is displayed which shown the general feedback, both feedback as client and feedback as server (Figure 5.7).

Name	Getting feedback
Objective	Provide users a feedback related with the download process.
Actor	Users
Precondition	User sets her preferences.
Sequence of interactions	User-Web interface
1.- The user clicks on the start button in order to initialize the download process. 2.- The WUW service considers the user preferences in the download process. 3.- The WUW service calculates the feedback. 4.- The WUW service sends the feedback to the web interface. 5.- The web interface displays the feedback in the screen.	
Postcondition	The web interface displays values of the feedback measures. The interface provides a plot of the feedback measures.

Table 5.3: Use case specification 3: Getting feedback

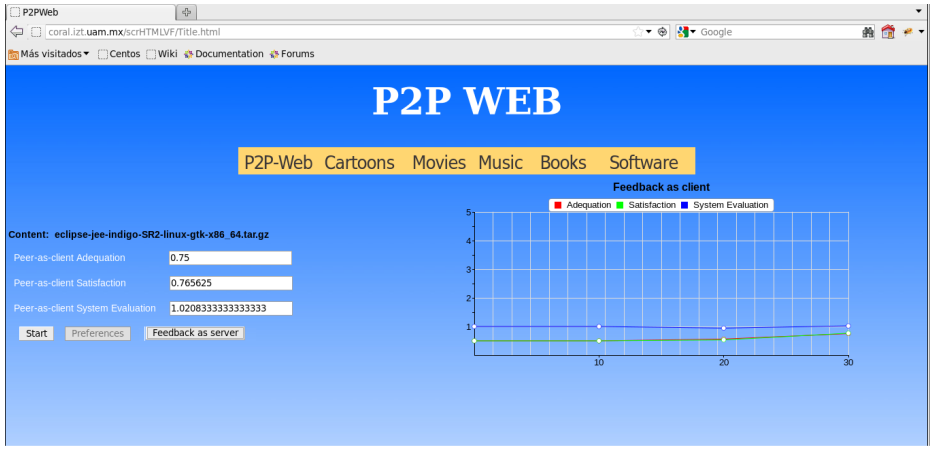


Figure 5.5: Feedback as client

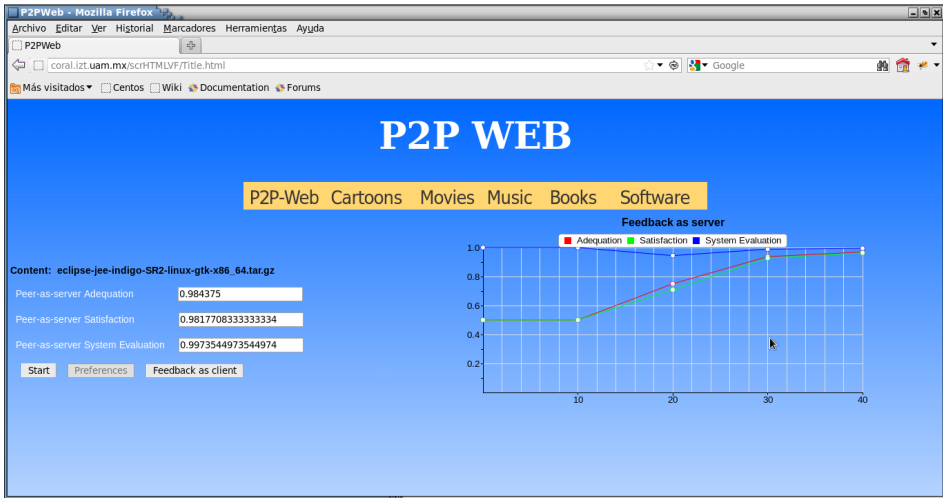


Figure 5.6: Feedback as server

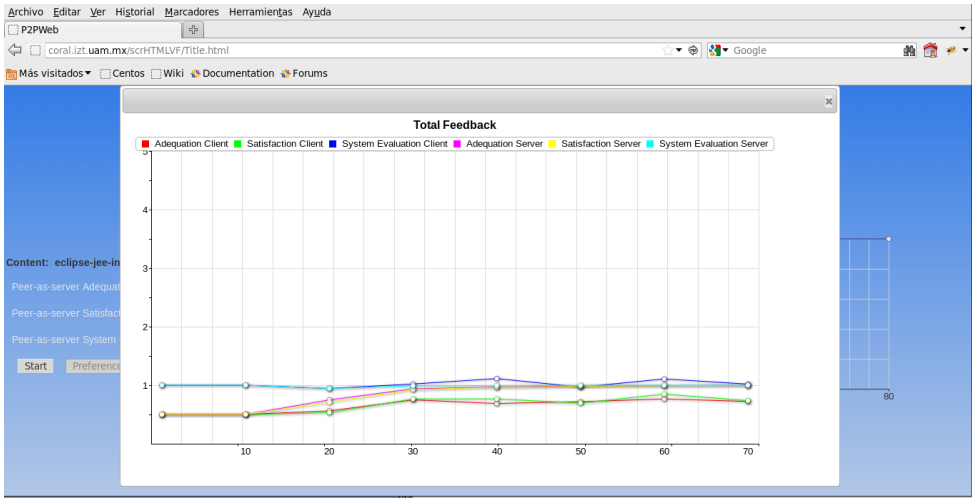


Figure 5.7: Pop-up window with the general feedback

Chapter 6
Conclusions and Future Work

Currently, there exist many different P2P applications that allow the content distribution among their users. However all of them do not take into account the users preferences that have not relation with QoS parameters. For that reason, in this thesis we evaluate the impact of WUW over BitTorrent through an experimental scenario and we show a demostration of the web interface that lets to users to define their preferences.

The objectives of this thesis, namely: to review and analyze some existing P2P content delivery applications in order to determine how they consider user preferences and how to test a service that considers users preferences, other than QoS related, in order to improve their satisfaction.

We reviewed the literature concerning the content distribution applications, this review allowed us to identify the architectures used in the distribution process, as well as their advantages and disadvantages. Considering the use of a P2P application as a specific context, we identified the different overlays used in these applications, as well as the different strategies used to select the chunks and peers.

We participated in the design of a service that considers the specified user preferences to determine the neighbors more interesting for for trading. We designed a web interface to allow the users to express their preferences. Subsequently, an experimentl evaluation was done with the objective to evaluate the impact of considering users preferences in the content distribution process using BitTorrent application.

The results of this experiment shown that the WUW service does not affect the performance of the BitTorrent application; despite the different tasks that the service realizes, reduce the communication with remote entities.

Below outlines the future work identified so far:

- Evaluate the performance of the WUW service, where users participate in the distribution process considering more than one content.
- Evaluate the performance considering more preferences like the geographic location,

the content type or reputation.

Appendices

Appendix A
User Interface-Java code

A.1 Preference.java

```
package wuw.ui;

public class Preference implements Comparable<Preference> {

private final String name;
private Object value;
private final Object defaultValue;
private final double start;
private final double end;
private final double step;
private final Object[] values;


/**
 * Constructor of a preference whose value belongs to a continuous range of
 * values
 *
 * @param name
 *          The (unique) name of the preference
 * @param start
 *          The lowest value the preference can be set to
 * @param end
 *          The highest value the preference can be set to
 * @param step
 *          The distance between two subsequent values that the preference can
 *          be set to
 * @param defaultValue
 *          The value by default of the preference
 */
public Preference(String name, double start, double end, double step, double defaultValue) {
  this.name = name;
  this.start = start;
  this.end = end;
  this.step = step;
  this.value = defaultValue;
  this.defaultValue = defaultValue;
  this.values = null;
}


/**
 * @param name
 *          the (unique) name of the preferences
```

```
 * @param values
 *          The values the preference can be set to
 * @param defaultValue
 *          The value of the preference by default
 */
public Preference(String name, Object[] values, Object defaultValue) {
  this.name = name;
  this.values = values;
  this.value = defaultValue;
  this.defaultValue = defaultValue;
  this.start = this.end = this.step = 0;
}


/**
 * @param value
 *          Set the value for the preference
 * @uml.property name="value"
 */
public synchronized void setValue(Object value) {
  this.value = value;
}


/**
 * @return the value of the preference
 */
public synchronized Object getValue() {
  return this.value;
}




/**
 * @return the default value of the preference
 */
public synchronized Object getDefaultValue() {
  return this.defaultValue;
}


/**
 * @return the value of the preference
 */
public synchronized Object[] getValues() {
  return this.values;
}


/**
 * @return the name of the preference
 * @uml.property name="name"
 */
public String getName() {
  return this.name;
}


/**
 * @return the lowest value the preference can be set to
 */
public double getStart() {
  return this.start;
```

```
}


/**
 * @return The highest value the preference can be set to
 **/
public double getEnd() {
  return this.end;
}


/**
 * @return The distance between two subsequent values that the preference can be
 *         set to
 */
public double getStep() {
  return this.step;
}


/**
 * @return The value by default of the preference
 */
public Object getDefaultVal() {
  return this.defaultValue;
}


/* (non-Javadoc)
 * @see java.lang.Comparable#compareTo(java.lang.Object)
 */
@Override
public int compareTo(Preference o) {
  return this.name.compareTo(o.getName());
}


/*
 * (non-Javadoc)
 * @see java.lang.Object#equals(java.lang.Object)
 */
@Override
public boolean equals(Object o) {
  if (o instanceof Preference) {
    return this.name.equals(((Preference) o).getName());
  }
  return false;
}

}
```

A.2 UIHandler.java

```
package wuw.ui;

import wuw.core.ContentData.Category;

public interface UIHandler {

public static final String[] feedback_measures = {
  "Peer-as-server Adequation",
  "Peer-as-server Satisfaction",
  "Peer-as-server System Evaluation",
  "Peer-as-client Adequation",
  "Peer-as-client Satisfaction",
  "Peer-as-client System Evaluation"
};

public static final double[] feedback_defaults = {
  0.5,
  0.5,
  1.0,
  0.5,
  0.5,
  1.0
};

//For web interfaz


public static  Preference[] preferences = {
new Preference(new String("PAS Content Type"),new Object[]{(Object)new String("MOVIES"),
(Object)new String("CARTOONS"),(Object)new String("NEWS"), (Object)new String("VIDEOCLIPS"),
(Object)new String("MUSIC"), (Object)new String("BOOKS")}, (Object)new String("MOVIES")),

new Preference(new String("PAS Bandwidth"), new Double (0), new Double(1048576),
new Double(0.1), new Double(0)),new Preference(new String("PAS Locate"),new Object[]{
(Object)new String("ASIA"),(Object)new String("EUROPE"),(Object)new String("NORTH AMERICA"),
(Object)new String("SOUTH AMERICA"), (Object)new String("AFRICA"), (Object)new String("AUSTRIA")}
,(Object)new String("EUROPE")),

new Preference(new String("PAS Reputation"), new Double (0), new Double(1), new Double(0.1),
new Double(0.5)),

new Preference(new String("PAC Content Type"),new Object[]{(Object)new String("MOVIES"),
(Object)new String("CARTOONS"),(Object)new String("NEWS"),(Object)new String("VIDEOCLIPS"),
(Object)new String("MUSIC"), (Object)new String("BOOKS")}, (Object)new String("MOVIES")),

new Preference(new String("PAC Bandwidth"), new Double (0), new Double(1048576),
new Double(0.1), new Double(0)),

new Preference(new String("PAC Locate"),new Object[]{(Object)new String("ASIA"),
(Object)new String("EUROPE"),(Object)new String("NORTH AMERICA"),
(Object)new String("SOUTH AMERICA"), (Object)new String("AFRICA"),
(Object)new String("AUSTRIA")}, (Object)new String("EUROPE")),

new Preference(new String("PAC Reputation"), new Double (0), new Double(1), new Double(0.1),
new Double(0.5)),new Preference(new String("sharePrefs"),new Object[]{(Object)new Boolean(true),
(Object)new Boolean(false)}, (Object)new Boolean(true)),

new Preference(new String("shareInts"),new Object[]{(Object)new Boolean(true),
(Object)new Boolean(false)}, (Object)new Boolean(true)),
```

```
new Preference(new String("saveAll"),new Object[]{(Object)new Boolean(true),
(Object)new Boolean(false)}, (Object)new Boolean(true)),

new Preference(new String("alpha"),new Double (0), new Double(1),
new Double(0.1), new Double(0.5)),

new Preference(new String("pref_weight"),new Double (0), new Double(1),
new Double(0.1), new Double(0.5)),

new Preference(new String("selfishness"),new Double (0), new Double(1),
new Double(0.1), new Double(0.5)),

new Preference(new String("maxNeighSize"),new Double (0), new Double(30),
new Double(1), new Double(10)),

//new Preference(new String("printLogs"),new Object[]{(Object)new Boolean(true),
(Object)new Boolean(false)}, (Object)new Boolean(true)),

new Preference(new String("localPort"),new Double (4000), new Double(6000),
new Double(1), new Double(5670)),

new Preference(new String("Interest"),new Object[]{(Object)new String("UNKNOWN"),
(Object)new String("NEGLIGIBLE"),(Object)new String("LOW"), (Object)new String("HIGH"),
(Object)new String("PRIMARY")}, (Object)new String("PRIMARY")),

new Preference(new String("protocol"),new Object[]{(Object)new String("UDP"),
(Object)new String("TCP")}, (Object)new String("UDP"))
};


/**
 * Initialize a numeric preference by specifying its validity range and the step
 * at which possible values must be set.
 *
 * @param contentID
 *          The ID of the content which the preference refers to
 * @param pref
 *          The (unique) name of the preference
 * @param start
 *          The lowest value the preference can be set to
 * @param end
 *          The highest value the preference can be set to
 * @param step
 *          The distance between two subsequent values that the preference can
 *          be set to
 * @param defaultValue
 *          The default value of the preference
 */
public void initPref(String contentID, String pref, double start, double end, double step,
    double defaultValue);


/**
 * Initialize a preference by specifying all possible values it can be set to.
 *
 * @param contentID
                        Socket socket = null;
 *          The ID of the content which the preference refers to
 * @param pref
 *          The (unique) name of the preference
 * @param values
 *          The values the preference can be set to
 * @param defaultValue
 *          The value by Default of the preference
```

```
 */
public void initPref(String contentID, String Pref, Object[] values, Object defaultValue);


/**
 * Initialize a preference by specifying all possible values it can be set to.
 *
 * @param contentID
 *          The ID of the content which the preference refers to
 * @param pref
 *                       Array of Preferences
 */
public void initPref(String contentID,  Preference[] Pref);


/**
 * Set a preference to a given value.
 *
 * @param contentID
 *          The ID of the content which the preference refers to
 * @param pref
 *          The name of the preference to be set
 * @param value
 *          The value the preference is to be set to
 */
public void setPref(String contentID, String pref, Object value);


/**
 * Retrieve the value of a given preference.
 *
 * @param contentID
 *          The ID of the content which the preference refers to
 * @param pref
 *          The name of the preference
 * @return The value of the preference
 */
public Object getPrefValue(String contentID, String pref);


/**
 * Initialize the measures displayed to the users as a feedback of the ongoing
 * computation.
 *
 * @param contentID
 *          The ID of the content which the measures refer to
 * @param measures
 *          The names of the measures
 * @param values
 *          The initial values of the measures, whose order matches the
 *          measures' names in <code>measures</code>
 */
public void initFeedback(String contentID, String[] measures, double[] values);


/**
 * Set a single feedback measure to a given value.
 *
 * @param contentID
 *          The ID of the content which the measure refers to
 * @param measure
 *          The name of the measure to be set
 * @param value
```

```
     *           The value to be set
     */
    public void setFeedback(String contentID, String measure, double value);


    /**
     * Get the concurrent value of a specific measure.
     *
     * @param contentID
     *           The ID of the content which the measure refers to
     * @param name_measure
     *           The name of the measure to be set
     *
     * @return  The value of the "name_measure"
     */
    public double getFeedback(String contentID, String name_measure);


}
```

A.3 Feedback.java

```
package wuw.ui.WebUI;

class Feedback {

String[] measures;
double[] values;


        /**
         * Constructor
         * @param measure
         *              Array of the set of names of the measures
         * @param values
         *              Array of the set of value of each measure
         */
        Feedback(String[] measure, double[] values) {
          this.measures = measure;
          this.values = values;
        }


        /**
         * Method that verify if the measure exist or not
         *
         * @param measure
         *                      Name of the measure
         * @return
         *                      The position of the measure if this exist or
         *                      -1 if the measure does not exist
         *
         */
        int existMeasure(String measure) {
          for (int i = 0; i < measures.length; i++) {
            if (measures[i].equals(measure)) {
              return i;
            }
          }
          return -1;
        }


        /**
         * Method that define the value of the measure in a specific position
         * @param index
         *                      The position of the measure
         * @param value
         *                      The value of the measure
         */
        void setValue(int index, double value) {
          values[index] = value;
        }

        /**
         * Method to get the value of a measure which is in a specific position
         * @param index
         *                Position of the measure
         * @return
         *                the concurrent value of the measure
         */
        double getValue(int index) {
                return values[index];
```

```
        }


        /**
         * Method used to get the name and values of the measure as a string well-formed
         * @return
         *              String with the name and values of the measure with "&" as separator
         */
        synchronized String getMeasure() {
          String feedback = "";
          for (int i = 0; i < measures.length; i++) {
            feedback = feedback + "&" + measures[i] + "&" + values[i];
          }
          return feedback;
        }
}
```

A.4 WebUIHandler.java

```
package wuw.ui.WebUI;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.ListIterator;
import java.util.Set;
import wuw.ui.Preference;
import wuw.ui.UIHandler;

public class WebUIHandler implements UIHandler {
	/**
	 * Each list of preferences is mapped against the unique ID of a content.
	 */
	public static LinkedHashMap<String, LinkedList<Preference>> preferences;
	public static LinkedHashMap<String, Feedback> feedback;


	public WebUIHandler() {
		preferences = new LinkedHashMap<String, LinkedList<Preference>>();
		feedback = new LinkedHashMap<String, Feedback>();
	}



	/*
	 * (non-Javadoc)
	 *
	 * @see www.ui.UIHandler#initPref(java.lang.String, java.lang.String, double,
	 * double, double, double)
	 */
	public void initPref(String contentID, String pref, double start, double end, double step,
			double defaultValue) {
		Preference item = new Preference(pref, start, end, step, defaultValue);
		getContents(contentID).add(item);
	}


	/*
	 * (non-Javadoc)
	 *
	 * @see www.ui.UIHandler#initPref(java.lang.String, java.lang.String,
	 * java.lang.Object[], java.lang.Object)
	 */
	@Override
	public void initPref(String contentID, String pref, Object[] values, Object defaultValue) {
		Preference item = new Preference(pref, values, defaultValue);
		getContents(contentID).add(item);
	}

	/*
	 * (non-Javadoc)
	 *
	 * @see www.ui.UIHandler#initPref(java.lang.String, java.lang.String,
```

```
         * java.lang.Object[], java.lang.Object)
         */
        @Override
        public void initPref(String contentID, Preference[] pref) {
                for(int i=0; i<pref.length; i++){
                        getContents(contentID).add(pref[i]);
                }
        }


        /*
         * (non-Javadoc)
         *
         * @see www.ui.UIHandler#setPref(java.lang.String, java.lang.String,
         * java.lang.Object)
         */
        public void setPref(String contentID, String pref, Object value) {
                LinkedList<Preference> list = preferences.get(contentID);
                if (list != null) {
                        ListIterator<Preference> itr = list.listIterator();
                        Preference aux;
                        while (itr.hasNext()) {
                                aux = itr.next();
                                if (aux.getName().equals(pref)) {
                                        aux.setValue(value);
                                }
                        }
                } else {
/System.err.println("WebUIHandler : ERROR : attempting to set a preference in an"
+ " uninitialized preference list (content ID = " + contentID + ")!!!");
                        System.err.flush();
                }

        }


        /*
         * (non-Javadoc)
         *
         * @see www.ui.UIHandler#getPrefValue(java.lang.String, java.lang.String)
         */
        public  Object getPrefValue(String contentID, String pref) {
                LinkedList<Preference> list = preferences.get(contentID);
                //System.out.println("TAM LIST ::::::::::"+list.size());
                if (list != null) {
                        ListIterator<Preference> itr = list.listIterator();
                        Object value = null;
                        Preference aux;
                        while (itr.hasNext()) {
                                aux = itr.next();
                                if (aux.getName().equals(pref)) {
                                        value = aux.getValue();
                                }
                        }
                        return value;
                } else {
 System.err.println("WebUIHandler : ERROR : attempting to get a value from an"
+ " uninitialized preference list (content ID = " + contentID + ")!!!");
                        System.err.flush();
                        return null;
                }
        }
```

```
	@Override
	public double getFeedback(String contentID, String name_measure) {
		double value;
		Feedback list = feedback.get(contentID);
		int i= list.existMeasure(name_measure);
		if (i!=-1) {
			value=list.getValue(i);
			return value;
		} else {
 System.err.println("WebUIHandler : ERROR : attempting to get a value from an"
 + " uninitialized preference list (content ID = " + contentID + ")!!!");
			System.err.flush();
			return 0;
		}
	}


	/*
	 * (non-Javadoc)
	 *
	 * @see UIHandler#initFeedback(java.lang.String[], double[])
	 */
	public void initFeedback(String contentID, String[] measures, double[] values) {
		if (measures.length == values.length) {
			Feedback instance = new Feedback(measures, values);
			feedback.put(contentID, instance);
		} else {
System.err.println("WebUIHandler : ERROR while intitializing feedback : arguments size mismatch.");
			System.err.flush();
		}
	}


	/*
	 * (non-Javadoc)
	 *
	 * @see UIHandler#setFeedback(java.lang.String, double)
	 */
	public void setFeedback(String contentID, String measure, double value) {
		if (feedback.containsKey(contentID)) {
			Feedback aux;
			aux = feedback.get(contentID);
			if (aux.existMeasure(measure) != -1) {
				aux.setValue(aux.existMeasure(measure), value);
			} else {
	System.err.println("WebUIHandler : ERROR : attempting to set an"
	+ " uninitialized measure (measure = " + measure + ")!!!");
				System.err.flush();
			}
		} else {
	System.err.println("WebUIHandler : ERROR : attempting to set in an"
	+ " uninitialized Feedback (content ID = " + contentID + ")!!!");
			System.err.flush();
		}

	}


	/**
	 * Function called to initialize the form in the Web Page.
	 *
	 * @param contentID
	 * @return String with a specific format that contains the name and value of
	 *         preferences used By Default Format of the String is "#" is a divider
```

```
         *          #NumberPrefereces#Preference1#Preference2#Preference3#.....# Each
         *          Preferences is integrate for 1)
         *          NamePreference,start,end,step,valueDefault 2) NamePreferences,Values
         *          that it can set to, valueDefault each data is separate for the
         *          character "&"
         */
        static String getPreferences(String contentID) {
                LinkedList<Preference> list = preferences.get(contentID);
                if (list != null) {
                        ListIterator<Preference> itr = list.listIterator();
                        Preference aux;
                        int numPref = list.size();
                        String answerRequest = "#" + String.valueOf(numPref);
                        while (itr.hasNext()) {
                                aux = itr.next();
                                if (aux.getValues() == null) {
                                        answerRequest = answerRequest + "#" +
                                        aux.getValues() + "&" + aux.getName() + "&"
                                        + aux.getStart() + "&" + aux.getEnd() + "&"
                                        + aux.getStep() + "&";
                                } else {
                                   answerRequest = answerRequest + "#" + "Combo"+
                                   "&" + aux.getName() + "&"+ aux.getValues().length+"&";

                                        for (int i = 0; i < aux.getValues().length; i++) {
                                                answerRequest =
                                                answerRequest + aux.getValues()[i] + "&";
                                        }

                                }
                                answerRequest = answerRequest + aux.getDefaultValue()+"&";
                        }
                        return answerRequest;
                } else {
System.err.println("WebUIHandler : ERROR : attempting to get a value from an"
+ " uninitialized preference list (content ID = " + contentID + ")!!!");
                        return "";
                }
        }


        /**
         *
         * @param contentID
         *              Reset the preferences of the specific content with the default values
         */
        static void reset(String contentID) {
                LinkedList<Preference> list = preferences.get(contentID);
                if (list != null) {
                        ListIterator<Preference> itr = list.listIterator();
                        Preference aux;
                        while (itr.hasNext()) {
                                aux = itr.next();
                                aux.setValue(aux.getDefaultValue());
                        }
                } else {
System.err.println("WebUIHandler : ERROR : attempting to reset preferences in an"
+ " uninitialized preference list (content ID = " + contentID + ")!!!");
                }
        }



        /**
```

```
        * @param contentID
        *       Display the list of Preferences of one Content with
                Id = contentID print the name of preferences,
        *       its default value and its value set up.
        */
       static void displayPreferences(String contentID) {
               LinkedList<Preference> list = preferences.get(contentID);
               if (list != null) {
                       ListIterator<Preference> itr = list.listIterator();
                       Preference aux;
                       while (itr.hasNext()) {
                               aux = itr.next();
System.out.println("Name:" + aux.getName() + "; Default value = " + aux.getDefaultVal()
+ "; Current value = " + aux.getValue() + ".");

                       }
               } else {
                  System.out.println("Uninitialized list.");
               }
       }

       /**
        * @param ID
        *          Identifier of the content to remove
        */
       void removeContent(String ID) {
               preferences.remove(ID);
               feedback.remove(ID);
       }


       /**
        * Delete the complete list of contents
        */
       void removeAll() {
               if (!preferences.isEmpty()) {
                       preferences.clear();
               }
               if (!feedback.isEmpty()) {
                       feedback.clear();
               }
       }


       /**
        * Show the id of the contents in the list
        */
       public static void displayContent() {
               Set<String> keys = preferences.keySet();
               System.out.println("List of Content");
               if (keys.isEmpty()) {
                       System.out.println("Empty!");
                       return;
               }
               Iterator<String> itrContent = keys.iterator();
               while (itrContent.hasNext()) {
                       System.out.println("Content:" + itrContent.next());
               }
       }


       /**
        * @param ID
        *          Identifier of the content
```

```
 * @return true if the content exist in the list false if don't exist
 */
public static boolean existContent(String ID) {
        return preferences.containsKey(ID) && feedback.containsKey(ID);
}


/**
 * @param idContent
 *          Id of Content
 */
public static String getFeeback(String idContent) {
        String FeedBack = idContent;
        Feedback aux = feedback.get(idContent);
        FeedBack = FeedBack + aux.getMeasure();
        return FeedBack;
}


/**
 * @param contentID
 *      Identificator of one content
 * @return
 *    A list with the name of the contents
 */
private LinkedList<Preference> getContents(String contentID) {
        if (!preferences.containsKey(contentID)) {
                LinkedList<Preference> list = null;
                list = new LinkedList<Preference>();
                preferences.put(contentID, list);
                return list;
        }
        return preferences.get(contentID);
}

/**
 * @param contentID
 *              Id of the content
 * @param Type
 *              Identify the preferences Type
 *              PAS (Preferences as Service) or
 *              PAC (Preferences as Client)
 * @return
 *              Array with the names of the preferences as Server and Client
 */
public static String[] getNamePreferencesPeer(LinkedHashMap<String,
LinkedList<Preference>> hashList, String contentID, String Type){
        LinkedHashMap<String, LinkedList<Preference>>  auxList=hashList;
        LinkedList<String> namePreferences = new LinkedList<String>();

        if (auxList.containsKey(contentID)) {
                LinkedList<Preference> list = auxList.get(contentID);
                if (list != null) {
                        ListIterator<Preference> itr = list.listIterator();
                        Preference aux;
                        while (itr.hasNext()) {
                                aux = itr.next();
                                if(aux.getName().contains(Type)){
                                        namePreferences.add(aux.getName());
                                }
                        }
        return (String[])namePreferences.toArray(new String[namePreferences.size()]);
                } else {
                        /**/System.out.println("Uninitialized list.");
```

```
                }

            }

            return null;
        }

        /**
         * @param contentID
         *              ID of the content
         * * @param Type
         *              Identify the preferences Type
                        Type= PAS (Preferences as Service) or
                              PAC (Preferences as Client)
         * @return
         *              Array with the values of the preferences
                        as Service and Client
         */
        public static Object[] getValuePreferencesPeers(LinkedHashMap<String,
        LinkedList<Preference>> hashList, String contentID, String Type){
                LinkedHashMap<String, LinkedList<Preference>> auxList=hashList;
                LinkedList<Object> namePreferences = new LinkedList<Object>();
                int i=0;
                if (auxList.containsKey(contentID)) {
                        LinkedList<Preference> list = auxList.get(contentID);
                        if (list != null) {
                                ListIterator<Preference> itr = list.listIterator();
                                Preference aux;
                                while (itr.hasNext()) {
                                        aux = itr.next();
                                        if(aux.getName().contains(Type)){
                                                namePreferences.add(aux.getValue());
                                        }
                                }
                                return namePreferences.toArray();
                        } else {
                                /**/System.out.println("Uninitialized list.");
                        }

                }
                return null;
        }

}
```

A.5 WebComHandler.java

```
package wuw.ui.WebUI;

import java.io.BufferedReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public interface WebComHandler {
        /*
         *This interface defines the communication between WUW service and browser
         *
         */


        /**
         *Retrieve information about content select by user in the browser,
         *this is the way WUW gets to know the user preferences.
         *@param input
         *    Reads text from a character-input stream which is storage in a buffer
         * @return
         *                A string with the user preferences
         *
         */
        public String Input(BufferedReader input);


        /**
         * This method send information from java side to the web page through applet
         * @param data
         *                Information to be sent
         * @param out
         *                Print formatted representations of objects to a text-output stream.
         */
        public void Response(String data, PrintWriter out);

        /**Method used to have access to the server socket
         * @return
         *                ServerSocket object
         */
        public ServerSocket getServerSocket();

        /**Method used to have access to the socket
         * @return
         *                Socket object
         */
        public Socket getSocket();

        /** Method used to have access to the printwriter of output
         * @return
         *                PrintWriter object
         */
        public PrintWriter getOutput();

        /**
         * Method used to decode the input information and response using printwriter
         * @param x
         *                Information received from web interface
         * @param out
         *                PrintWrited where WUW response to web interface
         * @return
         *                Information about the content (only for debug)
```

```
         */
        public String Decode(String x, PrintWriter out);

        /**Method used to have access to the  input buffer.
         * @return
         *              BufferedReader object
         */
        public BufferedReader getInputStream();

}
```

A.6 BrowserHandler.java

```
package wuw.ui.WebUI;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

import wuw.ui.UIHandler;

public class BrowserHandler implements WebComHandler{

private final UIHandler ui;
public static PrintWriter out = null;
public static ServerSocket serverSocket = null;
public static BufferedReader input = null;
public static Socket socket = null;


/**
 * Constructor
 */
public BrowserHandler(UIHandler ui) throws IOException{
        this.ui=ui;
        BrowserHandler.serverSocket = new ServerSocket(3030);
        BrowserHandler.socket = serverSocket.accept();
        BrowserHandler.input = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        BrowserHandler.out = new PrintWriter(socket.getOutputStream());
}


/* (non-Javadoc)
 * @see www.ui.WebUI.WebComHandler#getServerSocket()
 */
public ServerSocket getServerSocket(){
        return BrowserHandler.serverSocket;
}

/* (non-Javadoc)
 * @see www.ui.WebUI.WebComHandler#getSocket()
 */
public Socket getSocket(){
        return BrowserHandler.socket;
}

/* (non-Javadoc)
 * @see www.ui.WebUI.WebComHandler#getInputStream()
 */
public BufferedReader getInputStream(){
        return BrowserHandler.input;
}

/* (non-Javadoc)
 * @see www.ui.WebUI.WebComHandler#getOutput()
 */
public PrintWriter getOutput(){
        return BrowserHandler.out;
}

/* (non-Javadoc)
```

```
 * @see www.ui.WebUI.WebComHandler#Input(java.io.BufferedReader)
 */
public String Input (BufferedReader input) {
                String message="";
                        try {
                                message= input.readLine();
                                message=Decode(message,out);
                        } catch (IOException e) {
                                        e.printStackTrace();
                        }

                return message;
}


/* (non-Javadoc)
 * @see www.ui.WebUI.WebComHandler#Response(java.lang.String, java.io.PrintWriter)
 */
public void Response(String data, PrintWriter out){
        System.out.println("Envio"+ data);
        out.println(data);
        out.flush();
}

/**
 * Method to close the connection
 * @param serverSocket
 *                  A server socket which waits for requests to come in over the network.
 * @param input
 *                  Buffer that receive information by web interface
 * @param socket
 *                  The endpoint for communication between java and web interface
 */
public void Close(ServerSocket serverSocket, BufferedReader input, Socket socket){
        ServerSocket serversocket=serverSocket;
        BufferedReader inputs=input;
        Socket sockets=socket;
        try {
                serversocket.close();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        try {
                inputs.close();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        try {
                sockets.close();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}

/* (non-Javadoc)
 * @see www.ui.WebUI.WebComHandler#Decode(java.lang.String, java.io.PrintWriter)
 */
public String  Decode(String x, PrintWriter out){
        String result=x;
                if (x.contains("Request") == true) {
                        String message=Request(x);
```

```
                    System.out.println(message);
                    this.Response(message, this.out);
                } else if (x.contains("New#") == true) {
                     String idContent = obtainIdContent(x);

               if (WebUIHandler.existContent(idContent) == false) {
                  CreateContent(idContent);
                  WebUIHandler.displayPreferences(idContent);
                  WebUIHandler.displayContent();
               }

                } else if (x.contains("Init")){
                   String[]data=x.split("#");
                   String Interest=(String) ui.getPrefValue(data[1], "Interest");
                   result=result+"#"+Interest;
                   String feedback= "FeedBack#"+WebUIHandler.getFeeback(obtainIdContent(x));
                   this.Response(feedback, this.out);
                /*}else if (x.contains("Del#") == true) {
                }
                } else if (x.contains("Reset#") == true) {
                } else if (x.contains("exit") == true) {*/
                }else{
                    obtainValues(x);

                }
                return result;
        }

        /**
         * This method gets the id of the content
         * @param message
         *          Message received
         * @return The Id of the Content
         */
public static   String obtainIdContent(String message) {
          String[] IdContent = null;
          String content = new String(message);
          int pos = content.indexOf("#");
          if (pos != -1) {
            IdContent = message.split("#");
          }
          return IdContent[1].toString();
        }


        /* This function create a new content.
         *
         * @param idContent
         *          is the id content
         */
        public void CreateContent(String idContent) {
          ui.initPref(idContent,  UIHandler.preferences);
          ui.initFeedback(idContent, UIHandler.feedback_measures, UIHandler.feedback_defaults);
        }

        /**
         * This function is called when the user wants modify the preferences of the
         * content
         *
         * @param x
         *          is the id content
         * @return
         *  A string where are include all the names and values by default
         *          used in the preferences
```

```
     */
    private String Request(String x) {
      String content = obtainIdContent(x);
      String Form = "Form" + WebUIHandler.getPreferences(content);
      return Form;

    }

    /**
     * This methods obtains the values of the
     * user preferences received by the web interface
     * @param message
     *          message receive for the browser
     */
    private void obtainValues(String message) {
      String[] Values = null;
      int pos = message.indexOf("#", 0);
      if (pos != -1) {
        Values = message.split("#");
        setValues(Values);
      }
    }


    /**
     * This method used to call the @setPref method
     * @param idContent
     *          Id of the content
     * @param message
     *          String that include the name of the preferences and its value set up
     * @param numPref
     *          number of preferences
     */
    private void setValues(String[] Values) {
      String Preferences = Values[2].toString();
      String[] Pref = null;
      String[] SetValue = null;
      if (Preferences.indexOf("&", 0) != -1) {
        Pref = Preferences.split("&");
        for (int i = 0; i < Integer.parseInt(Values[1]); i++) {
          SetValue = Pref[i].split("=");
          ui.setPref(Values[0], SetValue[0], (Object)SetValue[1]);
        }

      }
      WebUIHandler.displayPreferences(Values[0]);
    }


    /**
     * This function sets the value of the preferences to its default value.
     *
     * @param idContent
     *          id of the content
     */
    private void resetValues(String idContent) {
            WebUIHandler.reset(idContent);
            WebUIHandler.displayPreferences(idContent);

    }
}
```

Appendix B
User Interface-HTML code

B.1 Title.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>P2PWeb</title>
<link rel="stylesheet" type="text/css" media="screen,
projection" href="./css/sliderbar.css" />
<link href="./css/style.css" rel="stylesheet" type="text/css" media="screen">
<link rel="stylesheet" href="./css/jquery-ui.css" />
<script src="http://code.jquery.com/jquery-1.8.3.js"></script>
<script type="text/javascript" src="./js/Title.js"></script>
<script src="http://code.jquery.com/ui/1.10.0/jquery-ui.js"></script>
<link href="./css/stylegraphics.css" rel="stylesheet" type="text/css" media="screen">
<script src="./RGraph/libraries/RGraph.common.core.js" ></script>
<script src="./RGraph/libraries/RGraph.common.dynamic.js" ></script>
<script src="./RGraph/libraries/RGraph.common.tooltips.js" ></script>
<script src="./RGraph/libraries/RGraph.common.effects.js" ></script>
<script src="./RGraph/libraries/RGraph.line.js" ></script>
<script src="./RGraph/libraries/RGraph.common.key.js" ></script>
<script>
        window.onbeforeunload=refreshTitle;
        window.onload=loading;
</script>
</head>
<body >
   <applet id="JavaSocket" archive="JavaSocket.jar"
    width="0" height="0">
</applet>
<div id="wrapper">
        <div id="logo">
          <h1><a href="#">P2P Web</a></h1>
        </div>
        <div id="menu">
          <center>
            <ul id="vbUL_wp36t" class="vbULwp36t">
                <li><a title="P2P-Web">P2P-Web</a></li>
                <li><a title="Cartoons">Cartoons</a>
                <div>
                   <ul id="vbUL_4p36t" class="vbULwp36t">
                        <li>The Flintstones</li>
                        <li>The Simpson</li>
                    </ul>
                </div>
                </li>
```

```
                <li><a title="Movies"> Movies</a>
                <div>
                   <ul id="vbUL_zp36t" class="vbULwp36t">
                        <li onMouseover="this.style.color='#FFFFFF'"
                          onMouseout="this.style.color='060606'"
                          onClick="createItem('IDContentA','IDContentA',);">The Artist</li>
                        <li >The Avengers</li>
                        <li >Departures</li>
                     </ul>
                </div>
                </li>
                <li><a title="VideoClips ">Music</a>
                <div>
                     <ul id="vbUL_up36t" class="vbULwp36t">
                        <li>Rolling In The Deep-Adele</li>
                        <li>The Cave - Mumford & Sons</li>
                      </ul>
                </div>
                </li>
                <li><a title="Books"> Books </a>
                <div>
                       <ul id="vbUL_up36t" class="vbULwp36t">
                          <li>Let's Pretend this never happened, by Jenny Lawson</li>
                          <li>The last boyfriend, by Nora Roberts</li>
                        </ul>
                </div>
                </li>
                <li><a title="News"> Software  </a>
                <div>
                        <ul id="vbUL_up36t" class="vbULwp36t">
                            <li onMouseover="this.style.color='#FFFFFF'"
                                onMouseout="this.style.color='060606'"
                                onClick="createItem('eclipse-jee-indigo-SR2-linux-gtk-x86_64.tar.gz',
'16920','SOFTWARE');">Eclipse 64 bits</li>
                          </ul>
                </div>
                </li>
            </ul>
<script type="text/javascript"> var vbImgPath="test2-files/"</script></center>
        </div>
        <div id="spacework">
             <div class="alignleft" id="leftcolumn">
             </div>
<canvas id="myLine1" width="600" height="300">[No canvas support]</canvas >
<canvas id="myLine2" width="600" height="300"  style="display:none;">[No canvas support]</canvas >
<div id="rightcolumn" title="Preferences" class="dialog">
        <form id="FormPreferences" >
                <fieldset id="LabelPref" name="LabelPref" >
                </fieldset>
        </form>
        <form id="bottom">
                <input type="Button" name="" value="Apply"
                  onClick="ValueSet();" size="20" align="left">
        </form>
</div>
</div>
</div>
<div id="general"  class="dialog">
<canvas id="line6" width="900" height="500">[No canvas support]</canvas >
</div>
</body>
</html>
```

B.2 Title.js

```
/*****************************************
        JAVASCRIPT VARIABLES GLOBAL
*****************************************/
//VARIABLES FOR CREATE SOCKET
// Global variables
var javaAppletReadyFlag = false;
var open_socket=false;
var window_refresh=true;
var flag_send_pref=false;
var times=0;

/********Global Array************************/
var feedbackasclient=new Array("Peer-as-client Adequation",
                               "Peer-as-client Satisfaction",
                               "Peer-as-client System Evaluation");
var feedbackasserver=new Array("Peer-as-server Adequation",
                               "Peer-as-server Satisfaction",
                               "Peer-as-server System Evaluation");

/*****************************************/
//GLOBAL ARRAYS FOR FEEDBACK
var Timeclient = new Array();
var Adecuationclient=new Array();
var Satisfactionclient=new Array();
var Evaluationclient= new Array();
var timeCl;
var time=0;
var TimeAux = new Array();
var AdecuationclientAux=new Array();
var SatisfactionclientAux=new Array();
var EvaluationclientAux= new Array();
var Adecuationserver=new Array();
var Satisfactionserver=new Array();
var Evaluationserver= new Array();
var AdecuationserverAux=new Array();
var SatisfactionserverAux=new Array();
var EvaluationserverAux= new Array();
/*****************************************
FUNCTIONS FOR CREATE THE SOCKET
FOR THE COMUNICATION AND ITS MESSAGE ERRORS
*****************************************/
// Applet reports it is ready to use
function appletReady(){
        javaAppletReadyFlag = true;
}

// Connect to a given url and port
function connectSocket(address, port){
        if(javaAppletReadyFlag){
                return getElementJava().connect(address, port);
        }
        else{
                on_socket_error("Cannot connect until the applet has loaded");
        }
}

// Disconnect
function disconnectSocket(){
        if(javaAppletReadyFlag){
                return getElementJava().disconnect();
        }
```

```
        else{
                on_socket_error("Cannot disconnect until the applet has loaded");
        }
}

// Write something to the socket
function sendMessage(message){
        if(javaAppletReadyFlag){
                return getElementJava().send(message);
        }
        else{
                on_socket_error("Cannot send a message until the applet has loaded");
        }
}



// Report an error
function on_socket_error(message){
        alert(message);
}

// Get the applet object
function getElementJava(){
        return document.getElementById('JavaSocket');
}

/*****************************************
Functions used for communication
whit java used by Title.html
*****************************************/
var flagTitle=false;

//Click  in one content of the list
function  CreateContent(idContent){
        var messageNew="New#"+idContent;
        //Sende message to Java for create the content
        runTitle(messageNew);
        //Create button for obtaint its preferences or delete the content
        //createItem(nameData,idContent);

}

//Function loading at the begining
        function loading(){
                if(javaAppletReadyFlag){
                   setTimeout("runTitle('init')",100);
                }else {
                        setTimeout("loading()",500);
                }
        }



//Function that send "message" to Java
function runTitle(message){
        if(flagTitle==false){
                socket_connect('coral.izt.uam.mx', 3030);
                flagTitle=true;
        }
        if(message!="init"){
                sendMessage(message);
        }
}
```

```
//Function that detect if a page is close or refresh
//send message to  java..
function refreshTitle(){
        if(flagTitle==true){
                runTitle("exit");
                flagTitle=false;
        }
}


//For receive message of the java application
function getMessage(message){
   var initial=0;
   var ending=0;
   ending=message.indexOf("#", initial);
   var data=message.substring(initial, ending);
   if(data == "Form"){
          Form(message.substring((ending+1),message.lenght ));
   else if (data=="FeedBack"){
        FeedBack(message.substring((ending+1),message.lenght ));
   }
}

/*****************************************
Functions used for create new item and
display button and label in the leftcolumn
and these cant execute the javascript
fucntion used by Title.html
*****************************************/

/*Function called by CreateContent for display
the name of content, button Preferences and button Close
Functio that create Dinaymically all
the items related to the content with id=idContent*/
function createItem(nameData, items,category){
//Only create the elemets if the elements don't exist
   if(document.getElementById(nameData) == null){
        var label = document.getElementById(nameData);
        addLabelBold(nameData, items,category);
        addBr(nameData);
        addBr(nameData);
        addLabel(nameData, "Peer-as-client Adequation");
        addLabell(nameData, "Peer-as-server Adequation");
        whiteSpace(nameData, 1);
        addInput(nameData, nameData, "Peer-as-client Adequation", "");
        addInputt(nameData, nameData, "Peer-as-server Adequation", "");
        addBr(nameData);
        addBr(nameData);
        addLabel(nameData, "Peer-as-client Satisfaction");
        addLabell(nameData, "Peer-as-server Satisfaction");
        whiteSpace(nameData, 1);
        addInput(nameData,nameData, "Peer-as-client Satisfaction", "");
        addInputt(nameData,nameData, "Peer-as-server Satisfaction", "");
        addBr(nameData);
        addBr(nameData);
        addLabel(nameData, "Peer-as-client System Evaluation");
        addLabell(nameData, "Peer-as-server System Evaluation");
        whiteSpace(nameData, 2);
        addInput(nameData,nameData, "Peer-as-client System Evaluation", "");
        addInputt(nameData,nameData, "Peer-as-server System Evaluation", "");
        addBr(nameData);
        addBr(nameData);
```

```
        whiteButton(nameData);
        addButtonStart(nameData,"Start",nameData);
        whiteButton(nameData);
        addButtonPref(nameData,"Preferences",nameData);
        whiteButton(nameData);
        addButtonFeedback(nameData,"Feedback␣as␣server");
        addBrTag(nameData);
        CreateContent(nameData)
    }
}

/*Function called by function createItem for construct the tag label
This tag is display in the div "leftcolumn"
the name of the tag is the name of the content
And write in document HTML this name
nameData = is the value that will take as Id the Label*/
function addLabelBold(nameData,items,category){
                var name= "<b>"+"<font␣color='black'>Content:  "+nameData+"␣</font></b>"
                var location= document.getElementById("leftcolumn");
                var label = document.createElement("label");
                label.id=nameData;
                label.title=items+"#"+category;
                label.innerHTML= name;
                location.appendChild(label);
}
//Function called by function createItem for construct the tag label
//This tag is display in the div "leftcolumn"
//the name of the tag is the name of the measure
//And write in document HTML this name
//nameData = is the value that will take as Id the Label
function addLabel(nameData, measure){
                var text= "  "+measure+" "
                var location= document.getElementById(nameData);
                var label = document.createElement("label");
                label.id=measure;
                label.innerHTML= text;
                location.appendChild(label);
}


//Function called by function createItem for construct the tag label
//The elements created by this function are not display but exist in the DOM
//This tag is display in the div "leftcolumn"
//the name of the tag is the name of the measure
//And write in document HTML this name
//nameData = is the value that will take as Id the Label
function addLabell(nameData, measure){
                var text= "  "+measure+" "
                var location= document.getElementById(nameData);
                var label = document.createElement("label");
                label.id=measure;
                label.innerHTML= text;
                label.style.display = "none";
                location.appendChild(label);
}

//Function called for create a label br
function addBr(parent){
        var location= document.getElementById(parent);
        var br= document.createElement("br");
        location.appendChild(br);
}

//Fucntion called for create white space according the name of feedback
```

```
function whiteSpace(parent, n){
                var text;
                switch(n){
                        case 1:
                          text=spaceAandS();
                          break;
                        case 2:
                           text=spaceSysEval();
                          break;
                }
                var location= document.getElementById(parent);
                var label = document.createElement("label");
                label.innerHTML= text;
                location.appendChild(label);
}


//Fucntion called for create white among the buttons
function whiteButton(parent){
                var text="    ";
                var location= document.getElementById(parent);
                var label = document.createElement("label");
                label.innerHTML= text;
                location.appendChild(label);
}


//White space for Adequation
function spaceAandS(){
        var text= "       
        "
        return text;
}


//White space for Adequation
function spaceSysEval(){
var text= "    "
return text;
}

//Function called by function addLabel for create Feedback button
//exectFunc is the function that this button execute
//nameData=Parent of the input tag
//idContent = Id Content with the button is associated
function addButtonFeedback(nameData, value){
                var exectFunc="buttonFeedback('"+nameData+"');"
                var location=document.getElementById(nameData);
                var button= document.createElement("input");
                button.type="button";
                button.id="buttonFeedback"+nameData;
                button.value=value;
                button.disabled=true;
                createEvent(button, exectFunc);
                location.appendChild(button);
}


//Function called by function addLabel for create Preferences button
//exectFunc is the function that this button execute
//nameData=Parent of the input tag
//idContent = Id Content with the button is associated
function addButtonPref(nameData, value,idContent){
                var exectFunc="buttonPreferences('"+idContent+"');"
```

```
                var location=document.getElementById(nameData);
                var button= document.createElement("input");
                button.type="button";
                button.id="buttonPref"+nameData;
                button.value=value;
                createEvent(button, exectFunc);
                location.appendChild(button);
        }
//Function called by function addLabel for create Delete button
//exectFunc is the function that this button execute
//Parent is  the tag label by this function is called
 function addButtonDel(nameData, value, idContent){
                var exectFunc="removeElement('"+nameData+"','"+idContent+"');"
                var location=document.getElementById(nameData);
                var button= document.createElement("input");
                button.id="buttonDel"+nameData;
                button.type="button";
                button.value=value;
                createEvent(button,exectFunc);
                location.appendChild(button);
}


//Function called by function addLabel for create Start button
//exectFunc is the function that this button execute
//Parent is  the tag label by this function is called
 function addButtonStart(nameData, value, idContent){
                var exectFunc="buttonStart('"+idContent+"');"
                var location=document.getElementById(nameData);
                var button= document.createElement("input");
                button.id="buttonStart"+nameData;
                button.type="button";
                button.value=value;
                button.disabled=true;
                createEvent(button,exectFunc);
                location.appendChild(button);
}

//Create the input tag that is associated with the SliderBar
//Parameters
//id = Parent of the input tag. Where the input tag will be generated
//idContent= ID of the content
//nameInput = name that will receive the InputTag
//valDefault = vale by dafault that the Preferences will be take
//This input tag only is for read
function addInput(id, idContent,nameInput, valDefault){
                var name=nameInput+idContent;
                var location=document.getElementById(id);
                var tag= document.createElement("input");
                tag.type="text";
                tag.name=name;
                tag.value=valDefault;
                tag.id=name;
                tag.size=20;
                tag.setAttribute("readonly", "true");
                location.appendChild(tag);
}

//Create the input tag that is associated with the SliderBar
//Parameters
//id = Parent of the input tag. Where the input tag will be generated
//idContent= ID of the content
//nameInput = name that will receive the InputTag
//valDefault = vale by dafault that the Preferences will be take
```

```
//This input tag only is for read
function addInputt(id, idContent,nameInput, valDefault){
                var name=nameInput+idContent;
                var location=document.getElementById(id);
                var tag= document.createElement("input");
                tag.type="text";
                tag.style.display="none";
                tag.name=name;
                tag.value=valDefault;
                tag.id=name;
                tag.size=20;
                tag.setAttribute("readonly", "true");
                location.appendChild(tag);
}

//Function called by function addLabel for create Label Tag
//Parent is  the tag label by this function is called
function addBrTag(idContent){
        var location= document.getElementById("leftcolumn");
        var br= document.createElement("br");
        br.id=idContent;
        location.appendChild(br);
}


//Action execute by the Feedback as client button

function buttonFeedback(nameData){
        var i;
        var parent=document.getElementById(nameData);
        var button;
        var exectFunc="buttonFeedback2('"+nameData+"');"
        document.getElementById("myLine1").style.display = 'none';
        document.getElementById("myLine2").style.display = '';
        for (i=0; i<feedbackasclient.length; i++){
                document.getElementById(feedbackasclient[i]).style.display = "none";
                document.getElementById(feedbackasclient[i]+nameData).style.display="none";
        }
        for (i=0; i<feedbackasserver.length; i++){
                document.getElementById(feedbackasserver[i]).style.display = '';
                document.getElementById(feedbackasserver[i]+nameData).style.display='';
        }
        button=document.getElementById("buttonFeedback"+nameData);
        button.value="Feedback as client";
        createEvent(button, exectFunc);
}


function buttonFeedback2(nameData){
        var i;
        var parent=document.getElementById(nameData);
        var button;
        var exectFunc="buttonFeedback('"+nameData+"');"
        document.getElementById("myLine2").style.display = 'none';
        document.getElementById("myLine1").style.display = '';
        for (i=0; i<feedbackasserver.length; i++){
                document.getElementById(feedbackasserver[i]).style.display = "none";
                document.getElementById(feedbackasserver[i]+nameData).style.display="none";
        }
        for (i=0; i<feedbackasserver.length; i++){
                document.getElementById(feedbackasclient[i]).style.display = '';
                document.getElementById(feedbackasclient[i]+nameData).style.display='';
        }
        button=document.getElementById("buttonFeedback"+nameData);
```

```
        button.value="Feedback as server";
        createEvent(button, exectFunc);
}



//Action execute by the Start button
//Send a message to java for obtaint the initialization to the form
function buttonStart(idContent){
        var items;
        var parent = document.getElementById("LabelPref");
        var nombre= parent.name;
        var feed="buttonFeedback"+idContent;
        if(parent.name==idContent  || parent.name=="LabelPref" ){
                if(document.getElementById('rightcolumn').style.visibility=='visible'){
                        document.getElementById('rightcolumn').style.visibility='hidden';
                        document.getElementById(feed).disabled=false;
                        delLabel();
                        parent.name="LabelPref";
                }
                parent.name=idContent;
                //CreateContent(idContent);
                var elements =document.getElementsByTagName("label");
                for(var k=0; k<elements.length; k++){
                        if(elements[k].getAttribute("id")==idContent){
                                items=elements[k].getAttribute("title");
                                break;
                        }
                }
        var request = "Init#"+idContent+'#'+items;
        runTitle(request);

        }else{
/*Because user wants open the preferences of the content A but in the screen
 are display the prefrences of the content B*/
                alert("Close Window Preferences");
        }
updates();
}

//Action execute by the Preferences button
//Send a message to java for obtaint the initialization to the form
function buttonPreferences(idContent){
        $( "#rightcolumn" ).dialog( "open" );
        var parent = document.getElementById("LabelPref");
        var nombre= parent.name;
        if(parent.name==idContent  || parent.name=="LabelPref" ){
                if(document.getElementById('rightcolumn').style.visibility=='visible'){
                        document.getElementById('rightcolumn').style.visibility='hidden';
                        delLabel();
                        parent.name="LabelPref";
                }else{
                        var request = "Request#"+idContent;
                        parent.name=idContent;
                        runTitle(request);
                }
        }else{
                //Because user wants open the preferences of the content A but in the screen are display the
                alert("Close Window Preferences");
        }
}

//Delete the Label Tags where the Preferences are display...
```

```
//Because only are generate when the user want modify the preferences
function delLabel(){
   if (document.all || document.getElementById) {
        var parent = document.getElementById("LabelPref");
        var elementosDiv = document.getElementById("LabelPref").getElementsByTagName('label');
        for ( var j = 0; j< elementosDiv.length; j+1){
                var id=elementosDiv[j].id;
                var temp=document.getElementById(id);
                parent.removeChild(temp);
        }
   }
}

//Function that receive the initialization of the form.
function Form(message){
  //Variables that allow to find the numbers of preferences that are recieved.
  //Function that find the first occurrens of "#"
                var reci=message;
                Create_Forms(reci);
                document.getElementById("rightcolumn").style.visibility="visible"
}


//Functioon that receive  the FeedBack
function FeedBack(message){
var myString = message;
var mySplitResult=myString.split("&");
for(i=1; i<mySplitResult.length;i=i+2){
        var nameLocation = mySplitResult[i]+mySplitResult[0];

        var input= document.getElementById(nameLocation);

        input.value= mySplitResult[i+1];
}
}

//Define How many Preferences are sending by the Java Application
function Create_Forms(message){
        var content;
        var numPreferences;
        var id;
        var id2;
        var id3;
        //split the string message and create a array called loging where is store the
        //total number preferences (in position 0)and all the preferences
        content= message.split("#");
        //Obtain the total number of preferences.
        numPreferences=content[0];
        //Generate the label tag one for each numPreferences.
        id= "PAC";
        id2="PAS"
        id3="Config"
        //Call to function to create a Label tag
        addLabelP(id);
        //Call to function to create a H3 tag
        addH3(id,"Preferences␣as␣Cliente" );
        CreateLabel(0,4);
        //Call to function to create a Label tag
        addLabelP(id2);
        //Call to function to create a H3 tag
        addH3(id2,"Preferences␣as␣Server" )
        CreateLabel(4, 8);
        //Call to function to create a Label tag
        addLabelP(id3);
```

```
        //Call to function to create a H3 tag
        addH3(id3,"Configuration" )
        CreateLabel(8, numPreferences);
        /*for each preferencees is called the function that
         identify the type of preference.*/
        for(var i=0; i<numPreferences; i++){
        //Call the function that identify the type of form that is used by the Preference
                TypeForm(i, content[i+1]);
        }
}


//Create Label for PAC y PAS
function addLabelP(idLabel, text){
                var location= document.getElementById("LabelPref");
                var label = document.createElement("label");
                label.id=idLabel;
                label.innerHTML= " ";
                location.appendChild(label);
}

//Parameter idLabel = identify that will take the label
//This Label are generate in the element with id= "LabelPref"
function addLabelPref(idLabel){
                var location= document.getElementById("LabelPref");
                var label = document.createElement("label");
                label.id=idLabel;
                label.innerHTML= " ";
                location.appendChild(label);
}


//Loop for Create  Label for construct the preferences in combo and sliderbar forms
//Parameter contDiv = number of Preferences received
function CreateLabel(init, contDiv){
        for(var i=init; i<contDiv; i++){
                addLabelP(i);
        }
}

//Create and include the Label Tag for Generate the Combo and sliderbar Forms
//Parameter idLabel = identify that will take the label
//This Label are generate in the element with id= "LabelPref"
function addLabelPref(idLabel){
                var location= document.getElementById("LabelPref");
                var label = document.createElement("label");
                label.id=idLabel;
                label.innerHTML= " ";
                location.appendChild(label);
}

//Param
//id_position = Id of the element where will create the form
//Preferences = Include the id, name and values that will take the Forms
function TypeForm(id_position, Preference){
var Preferences;
var Aux;
Aux=Preference.substring(0, Preference.lastIndexOf("&"));
Preferences=Aux.split("&");
//id= type of Prefences (sliderbar = 0 || Combobox =1)
var id=Preferences[0];
Preferences.splice(0,1);
        if(id=="null"){
```

```
                //Call the function that create Slider Bar
                Constructor( id_position , Preferences);

        }else{
                //Call the function that create the Combo box
                Constructor_Combo( id_position , Preferences);
        }
}

 //Constructor Combo, storage in the combo the possible values of the preferences
 //Parameters
 //id_position=Location where the Form must be constructed
 //Value= Value that have to take the Form
function Constructor_Combo(id_position , Preferences){
        var j;
        var nameData;
        var numValuesofData;
        //Obtain the name of preference
        nameData=Preferences[0];
        //Obtain the values that will able to select in the Combo.
        //Read the number of possible values that can take the preferences.
        numValuesofData=Preferences[1];
        //Create the tag H4 with the name of the Preferences
        Preferences.splice(0,2);
        addH4(id_position ,nameData);
        //this part create form and select tag
        mypara=document.getElementById(id_position);
        myselect = document.createElement("select");
        //the id and name of the tag select is the name of the preferences
        myselect.name=nameData;
        myselect.id=nameData;
        //Each possible value is add to the select form with the option tag
                for(j=0; j<numValuesofData; j++){
                        theOption=document.createElement("OPTION");
                        theText=document.createTextNode(Preferences[j]);
                        theOption.appendChild(theText);
                        myselect.appendChild(theOption);
                }
        Preferences.splice(0,numValuesofData);
         //The last data is the value that by defaul is select
        myselect.value=Preferences[0];
        //the  form tag is join as a child of the p tag
        mypara.appendChild(myselect);
}

//this function create a H3 tag, and is add in the "id" tag  with the value nameData
function addH3(id, nameData){
        var name=id;
        var parent = document.getElementById(name);
        var h3 = document.createElement("h3");
        h3.innerHTML = nameData;
        parent.appendChild(h3);

}

//this function create a H4 tag, and is add in the "id" tag  with the value nameData
function addH4(id, nameData){
        var name=id;
        var parent = document.getElementById(name);
        var h4 = document.createElement("h4");
        h4.innerHTML = nameData;
        parent.appendChild(h4);

}
```

```
//Function used to creade sliderbar
function addPrueba(id, min, max){
        var interval=min+"           ";
        interval+="            "
        interfal+="        "+max+"<br>"
        var nameParent=id;
        var parent=document.getElementById(nameParent);
        var label= document.createElement("label");
        label.id=id+"slide";
        label.innerHTML=interval;
        parent.appendChild(label);
}


//This function is responsable to create the Sliderbar with id=0
//Parameteres
//id=Element where must be created the Form
//values= include name, min, max, step , value_default for construct the SliderBar
function Constructor(id, values){
        var namePref=values[0];
        //Obtain the min value that will take the preferences
        var min=values[1];
        //Obtain the max value that will take the preferences
        var max=values[2];
        //Obtain the step value that will use the SliderBar
        var step=values[3];
        //Obtain the value select by default for this Preferences
        var value=values[4];
        //Create a h4 tag with the name of the Preferences
        addH4(id,namePref);
        addPrueba(id, min, max);
        //Create a Input tag associated with the SliderBar
        addInputTag(id, namePref, value);
        //Create the script tage responsable to generate the SliderBar
        addScriptTag(id, namePref, min, max, step);
        removingChild(id, namePref);
}


//Remove Element

function removingChild(parent, child){
        var Parent=document.getElementById(parent);
        var namechild="fd-slider-"+child;
        var children=document.getElementById(namechild);
        children.id=namechild+"Omit";
        Parent.removeChild(children);

}



//Create the ScriptTag that is responsable to generate the SliderBar
//Parameters
//id = Parent tag and Where the script tag will be generate
//namePref = name of the Preferences
//min = minimal value that can take the preferences
//max = maximal value that can take the preferences
//step= interval between two posible value
function addScriptTag(id,namePref, min, max, step){
        //alert("Valor "+ id);
        //fun= function that the script tag has to execute
                var fun="Init('"+namePref+"',"+min+","+max+","+step+");"
                var ss = document.createElement('script');
```

```
                //function that is executed for javascript is append in scr
                var scr = fun;
                //Add the function to the script tag
                var tt = document.createTextNode(scr);
                ss.appendChild(tt);
                //Obtain the element with ID="id"
                var hh = document.getElementById(id);
                //alert("Tag: "+hh.tagName);
                //Add the script tag to the element
                hh.appendChild(ss);
}


//Create the input tag that is associated with the SliderBar
//Parameters
//id = Parent of the input tag. Where the input tag will be generated
//nameInput = name that will receive the InputTag
//valDefault = vale by dafault that the Preferences will be take
//This input tag only is for read
function addInputTag(id,nameInput, valDefault){
                var name=nameInput
                var location=document.getElementById(id);
                var tag= document.createElement("input");
                tag.type="text";
                tag.name=name;
                tag.value=valDefault;
                tag.id=name;
                tag.size=10;
                tag.setAttribute("readonly", "true");
                location.appendChild(tag);

}

//Create Object that is send as parameter for create the SliderBar
//Parameters
//inputAssoc = input associated with the SliderBar for show the value
//minValue = minimal value that can take the preferences
//maxValue = maximal value that can take the preferences
//stepValue= interval between two posible value
function Init(inputAssoc, minValue, maxValue, stepValue){
        var newObj= new Object();
        newObj.inp = document.getElementById(inputAssoc);
        newObj.step= stepValue;
        newObj.min = minValue;
        newObj.max = maxValue;
        newObj.animation="jump";
        fdSlider.createSlider(newObj);
}



//Function executed by Button Delete
//Parameters
//nameData = name of the content
//idContent = Id of the content

function removeElement(nameData, idContent) {
        //if the div that display the preferences is visible then ask
        //if the preferences that display are to the content that wants to delete
        //then  close windows preferences and delete button
        //else  only delete buttons
        if(document.getElementById('rightcolumn').style.visibility=='visible'){
         var parent = document.getElementById("LabelPref");
```

```
         if(parent.name == idContent){
                document.getElementById('rightcolumn').style.visibility='hidden';
                delLabel();
                parent.name="LabelPref";
         }
        }
                var del="Del#"+idContent;
                var element="buttonPref"+nameData;
                var d = document.getElementById('leftcolumn');
                var tagbr = document.getElementById('leftcolumn').getElementsByTagName('br');
                var olddiv = document.getElementById(nameData);
                var b=document.getElementById(element);
                b.setAttribute("onClick","");
                d.removeChild(olddiv);
                for (var k=0; k<tagbr.length; k++){
                        if(tagbr[k].id == idContent){
                                d.removeChild(tagbr[k]);
                        }
                }
                socket_send(del);
}

//Change the function that input execute onClick
//input = Input element that wants modify
//nameFunction = function that will execute
function createEvent(input, nameFunction){
        input.setAttribute("onClick",nameFunction);
}

//Function execute by Button Close in the form
//hidde the div where is display the preferences and delete the label where the preferences are constructed
/*function closePref(){
                document.getElementById('rightcolumn').style.visibility='hidden';
                delLabel();
                var parent = document.getElementById("LabelPref");
                parent.name="LabelPref";
}*/


//Function Called by Button Apply in the Form
//This function obtain the value that the user set for the preferences
//These values are sending to java
function ValueSet(){
        var location =document.getElementById("LabelPref");
        var idContent = location.name;
        var buttonStart="buttonStart"+idContent;
        var buttonPref="buttonPref"+idContent;
        if(document.getElementById(buttonStart).disabled){
                document.getElementById(buttonStart).disabled=false;
                document.getElementById(buttonPref).disabled=true;
        }
        runTitle(ObtainValues("FormPreferences"));

}

function Reset(){
        var location =document.getElementById("LabelPref");
        var idContent = location.name;
        var messageReset="Reset#"+idContent;
        var request = "Request#"+idContent;
        runTitle(messageReset);
        delLabel();
        runTitle(request);
```

```
}
//Obtain name and value  of the preferences in the form that user define
function ObtainValues(theform) {
   if (document.all || document.getElementById) {
    var location =document.getElementById("LabelPref");
    var idContent = location.name;
    var elementForm = document.getElementById(theform);
    var numPref=elementForm.length-2;
    valSet=idContent+"#"+numPref+"#"
        for (var i = 0; i < elementForm.length; i++){
           if(elementForm[i].id != "LabelPref"){
                if(elementForm[i].id != "Config"){
                 valSet=valSet+elementForm[i].name+"="+elementForm[i].value+"&";
                }else{
                 valSet=valSet+"$";
                }
            }
        }
   }
   return valSet;
}

/*****************************************************************
*               Function that create the slider-bar
*               with support in differents browser
*****************************************************************/
var fdSlider = (function() {
  var sliders= {},
  uniqueid= 0,
  mouseWheelEnabled = true,
  ullARIA= true,
  describedBy= "fd-slider-describedby",
  varSetRules={
   onfocus:true,
   onvalue:true
  },
  noRangeBar= false,
  html5Animation= "jump",
  isOpera= Object.prototype.toString.call(window.opera) == "[object␣Opera]",
  fpRegExp= /^([-]{0,1}[0-9]+(\.[0-9]+){0,1})$/,
  stepRegExp= /^([0-9]+(\.[0-9]+){0,1})$/;
  var parseJSON = function(str) {
  // Check we have a String
  if(typeof str !== 'string' || str == "") {
    return {};
  };
  try {
  // Does a JSON (native or not) Object exist
  if(typeof JSON == "object" && JSON.parse) {
     return window.JSON.parse(str);
     // Genious code taken from: http://kentbrewster.com/badges/
  } else
    if(/mousewheelenabled|fullaria|describedby|norangebar|
        html5animation|varsetrules/.test(str.toLowerCase(){
    var f = Function(['var␣document,top,self,window,parent,Number,Date,Object,Function,',
'Array,String,Math,RegExp,Image,ActiveXObject;',
    'return␣(',str.replace(/<\!--.+-->/gim,'').replace(/\bfunction\b/g,'function-') ,
          ');'].join(''));
        return f();
    };
  }catch (e) { };
        return {"err":"Could␣not␣parse␣the␣JSON␣object"};
  };
```

```
var affectJSON = function(json) {
 if(typeof json !== "object") { return; };
  for(key in json) {
   value = json[key];
   switch(key.toLowerCase()) {
    case "mousewheelenabled":
          mouseWheelEnabled = !!value;
          break;
    case "fullaria":
          fullARIA = !!value;
          break;
    case "describedby":
          describedBy = String(value);
          break;
    case "norangebar":
          noRangeBar = !!value;
          break;
    case "html5animation":
          html5Animation = String(value).search(/^(jump|tween|timed)$/i) != -1 ?
              String(value).toLowerCase() : "jump";
          break;
    case "varsetrules":
     if("onfocus" in value) {
       varSetRules.onfocus = !!value.onfocus;
     };
     if("onvalue" in value) {
       varSetRules.onvalue = !!value.onvalue;
     };break;
   };
  };
 };

 // Classic event functions
var addEvent = function(obj, type, fn) {
 if( obj.attachEvent ) {
     obj.attachEvent( "on"+type, fn );
 } else { obj.addEventListener( type, fn, true ); }
};
var removeEvent = function(obj, type, fn) {
 try {
   if( obj.detachEvent ) {
     obj.detachEvent( "on"+type, fn );
   } else { obj.removeEventListener( type, fn, true ); }
 } catch(err) {};
};
var stopEvent = function(e) {
 e = e || window.event;
 if(e.stopPropagation) {
    e.stopPropagation();
    e.preventDefault();
 };
 return false;
};
var preventDefault = function(e) {
  e = e || window.event;
  if(e.preventDefault) {
     e.preventDefault();
     return;
  };
  e.returnValue = false;
};
// Add/Remove classname utility functions
var addClass = function(e,c) {
  if(new RegExp("(^|\\s)" + c + "(\\s|$)").test(e.className)) { return; };
```

```
        e.className += ( e.className ? "␣" : "" ) + c;
    };

 var removeClass = function(e,c) {
     e.className = !c ? "" : e.className.replace(new RegExp("(^|\\s)" +
        c + "(\\s|$)"), "␣").replace(/^\s\s*/, '').replace(/\s\s*$/, '');
 };

 // Returns an Object of key value pairs indicating which sliders have values
 // that have been "set" by the user
 var getValueSet = function() {
   var obj = {};
   for(id in sliders) {
      obj[id] = sliders[id].getValueSet();
   };
      return obj;
 };

// Sets the valueSet variable for a specific slider
 var setValueSet = function(sliderId, tf) {
  sliders[sliderId].setValueSet(!!tf);
 };
// Does the slider exist in memory
 var sliderExists = function(slider) {
  return !!(slider in sliders && sliders.hasOwnProperty(slider));
 };

// Javascript instantiation of a slider (input type="text" or select list)
var createSlider = function(options) {
 if(!options || !options.inp || !options.inp.tagName ||
     options.inp.tagName.search(/^input|select/i) == -1) {
       return false;
 };
 options.html5Shim = false;
 if(options.inp.tagName.toLowerCase() == "select") {
   if(options.inp.options.length < 2) {
     return false;
   };
   options.min= 0;
   options.max= options.inp.options.length - 1;
   options.step= 1;
   options.precision= 0;
   options.scale= false;
   options.forceValue= true;
 }else {
   if(String(options.inp.type).search(/^text$/i) == -1) {
     return false;
   };
   options.min= options.min && String(options.min).search(fpRegExp) != -1 ? +options.min : 0;
   options.max= options.max && String(options.max).search(fpRegExp) != -1 ? +options.max : 100;
   options.step= options.step && String(options.step).search(stepRegExp) != -1 ? options.step : 1;
   options.precision = options.precision && String(options.precision).search(/^[0-9]+$/) != -1 ?
       options.precision : (String(options.step).search(/\.([0-9]+)$/) != -1 ?
       String(options.step).match(/\.([0-9]+)$/)[1].length : 0);
   options.scale= options.scale || false;
   options.forceValue = ("forceValue" in options) ? !!options.forceValue : false;
 };
 options.maxStep    = options.maxStep && String(options.maxStep).search(stepRegExp) != -1 ?
                      +options.maxStep : +options.step * 2;
 options.classNames = options.classNames || "";
 options.callbacks  = options.callbacks || false;
 destroySingleSlider(options.inp.id);
 sliders[options.inp.id] = new fdRange(options);
 return true;
```

```
};

var getAttribute = function(elem, att) {
  return elem.getAttribute(att) || "";
};

// HTML5 input type="range" shim - called onload or onDomReady
var init = function() {
 var inputs = document.getElementsByTagName("input"),
 options;
 for(var i = 0, inp; inp = inputs[i]; i++) {
  if(inp.tagName.toLowerCase() == "input"
     &&
     inp.type.toLowerCase() == "text"
     &&
     (getAttribute(inp, "min") && getAttribute(inp, "min").search(fpRegExp) != -1
     ||
     getAttribute(inp, "max") && getAttribute(inp, "max").search(fpRegExp) != -1
     ||
     getAttribute(inp, "step")
     &&
     getAttribute(inp, "step").search(/^(any|([0-9]+(\.[0-9]+){0,1}))$/i) != -1
   )) {
   // Skip elements that have already been created are are resident in the DOM
     if(inp.id && document.getElementById("fd-slider-"+inp.id)) {
       continue;
     // Destroy elements that have already been created but not resident in the DOM
     } else if(inp.id && !document.getElementById("fd-slider-"+inp.id)) {
       destroySingleSlider(inp.id);
     };
     // Create an id for the form element if necessary
     if(!inp.id) {
        inp.id = "fd-slider-form-elem-" + uniqueid++;
     };
     // Basic option Object
     options = {
     inp:            inp,
     callbacks:      [],
     animation:      html5Animation,
     vertical:       getAttribute(inp, "data-fd-slider-vertical") ?
                      true : !!(inp.offsetHeight > inp.offsetWidth),
                      classNames:     getAttribute(inp, "data-fd-slider-vertical"),
                      html5Shim:      true
     };
     if(options.vertical && !getAttribute(inp, "data-fd-slider-vertical")) {
        options.inpHeight = inp.offsetHeight;
     };
     options.min = getAttribute(inp, "min") || 0;
     options.max = getAttribute(inp, "max") || 100;
     options.step = getAttribute(inp, "step").search(/^any$/i) != -1 ?
       options.max - options.min : getAttribute(inp, "step").search(stepRegExp) != -1 ?
              inp.getAttribute("step") : 1;
     options.precision= String(options.step).search(/\.([0-9]+)$/) != -1 ?
                        String(options.step).match(/\.([0-9]+)$/)[1].length : 0;
     options.maxStep= options.step * 2;
     destroySingleSlider(options.inp.id);
     sliders[options.inp.id] = new fdRange(options);
   };
 };
 return true;
};

 var destroySingleSlider = function(id) {
    if(id in sliders && sliders.hasOwnProperty(id)) {
```

```
        sliders[id].destroy();
        delete sliders[id];
        return true;
       };
       return false;
    };
var destroyAllsliders = function(e) {
 for(slider in sliders) {
    if(sliders.hasOwnProperty(slider)) {
      sliders[slider].destroy();
    };
 };
 sliders = [];
};
var unload = function(e) {
 destroyAllsliders();
 sliders = null;
};
var resize = function(e) {
 for(slider in sliders) {
  if(sliders.hasOwnProperty(slider)) {
   sliders[slider].onResize();
  };
 };
};
var onDomReady = function() {
 removeEvent(window, "load",   init);
 init();
};
var removeOnLoadEvent = function() {
 removeEvent(window, "load",   init);
};

function fdRange(options) {
 var inp= options.inp,
 disabled= false,
 tagName= inp.tagName.toLowerCase(),
 min= +options.min,
 max= +options.max,
 rMin= +options.min,
 rMax= +options.max,
 range= Math.abs(max - min),
 step= tagName == "select" ? 1 : +options.step,
 maxStep= options.maxStep ? +options.maxStep : step * 2,
 precision= options.precision || 0,
 steps= Math.ceil(range / step),
 scale= options.scale || false,
 hideInput= !!options.hideInput,
 animation   = options.animation || "",
 vertical= !!options.vertical,
 callbacks   = options.callbacks || {},
 classNames  = options.classNames || "",
 html5Shim   = !!options.html5Shim,
 defaultVal  = max < min ? min : min + ((max - min) / 2),
 resetDef    = tagName == "select" ? inp.selectedIndex : inp.defaultValue || defaultVal,
 forceValue  = html5Shim || !!options.forceValue,
 inpHeight   = html5Shim && vertical && ("inpHeight" in options) ? options.inpHeight : false,
 timer       = null,
 kbEnabled   = true,
 initialVal  = tagName == "select" ? inp.selectedIndex : inp.value,
 sliderH     = 0,
 sliderW     = 0,
 tweenX      = 0,
 tweenB      = 0,
```

```
tweenC      = 0,
tweenD      = 0,
frame       = 0,
x           = 0,
y           = 0,
rMaxPx      = 0,
rMinPx      = 0,
handlePos   = 0,
destPos     = 0,
mousePos    = 0,
stepPx      = 0,
userSet     = false,
touchEvents = false,
outerWrapper,
wrapper,
handle,
rangeBar,
bar;

// For the reset event to work we have set a defaultValue
if(tagName == "input" && forceValue && !inp.defaultValue) {
  inp.defaultValue = getWorkingValueFromInput();
};
// Make sure we have a negative step if the max < min
if(max < min) {
   step    = -Math.abs(step);
   maxStep = -Math.abs(maxStep);
 };
 // Add the 100% scale mark if needs be
 if(scale) {
   scale[100] = max;
 };
 // Set the "userSet" variable programmatically for this slider
 function valueSet(tf) {
   tf = !!tf;
   if(tf != userSet) {
    userSet = tf;
    valueToPixels(getWorkingValueFromInput());
   };
 };

 function disableSlider(noCallback) {
  if(disabled && !noCallback) {
   return;
  };
  try {
    setTabIndex(handle, -1);
    removeEvent(handle, "focus",      onFocus);
    removeEvent(handle, "blur",       onBlur);
    if(!isOpera) {
     removeEvent(handle, "keydown",   onKeyDown);
     removeEvent(handle, "keypress",  onKeyPress);
    } else {
     removeEvent(handle, "keypress",  onKeyDown);
    };
    removeEvent(outerWrapper, "mouseover",  onMouseOver);
    removeEvent(outerWrapper, "mouseout",   onMouseOut);
    removeEvent(outerWrapper, "mousedown",  onMouseDown);
    removeEvent(outerWrapper, "touchstart", onMouseDown);
    if(mouseWheelEnabled) {
      if (window.addEventListener &&
          !window.devicePixelRatio)
               window.removeEventListener('DOMMouseScroll', trackMouseWheel, false);
      else {
```

```
            removeEvent(document, "mousewheel", trackMouseWheel);
            removeEvent(window,   "mousewheel", trackMouseWheel);
        };
      };
    } catch(err) {};
      removeClass(outerWrapper, "fd-slider-focused");
      removeClass(outerWrapper, "fd-slider-active");
      addClass(outerWrapper, "fd-slider-disabled");
      outerWrapper.setAttribute("aria-disabled", true);
      inp.disabled = disabled = true;
      clearTimeout(timer);
      if(!noCallback) {
        callback("disable");
      };
  };
  function enableSlider(noCallback) {
    if(!disabled && !noCallback) {
      return;
    };
    setTabIndex(handle, 0);
    addEvent(handle, "focus",      onFocus);
    addEvent(handle, "blur",       onBlur);
    if(!isOpera) {
      addEvent(handle, "keydown",   onKeyDown);
      addEvent(handle, "keypress",  onKeyPress);
    } else {
      addEvent(handle, "keypress",  onKeyDown);
    };

    addEvent(outerWrapper, "touchstart", onMouseDown);
    addEvent(outerWrapper, "mousedown",  onMouseDown);
    addEvent(outerWrapper, "mouseover",  onMouseOver);
    addEvent(outerWrapper, "mouseout",   onMouseOut);
    removeClass(outerWrapper, "fd-slider-disabled");
    outerWrapper.setAttribute("aria-disabled", false);
    inp.disabled = disabled = touchEvents = false;
    if(!noCallback) {
      callback("enable");
    };
};

// Destroys a slider
function destroySlider() {
  // Clear any timeouts
  clearTimeout(timer);
  // Remove pointers to DOM nodes
  wrapper = bar = handle = outerWrapper = timer = null;
  // Call the "destroy" callback
  callback("destroy");
  //Delete the callback functions
  callbacks = null;
};
// Calculates the pixel increment etc
function redraw() {
  locate();
  // Internet Explorer requires the try catch as hidden
  // elements throw errors
  try {
    var sW  = outerWrapper.offsetWidth,
    sH  = outerWrapper.offsetHeight,
    hW  = handle.offsetWidth,
    hH  = handle.offsetHeight,
    bH  = bar.offsetHeight,
    bW  = bar.offsetWidth,
```

```
      mPx = vertical ? sH - hH : sW - hW;
      stepPx = mPx / steps;
      rMinPx = Math.max(scale ?
          percentToPixels(valueToPercent(rMin)) : Math.abs((rMin - min) / step) * stepPx, 0);
      rMaxPx = Math.min(scale ?
         percentToPixels(valueToPercent(rMax)) : Math.abs((rMax - min) / step) * stepPx,
         Math.floor(vertical ? sH - hH : sW - hW));
      sliderW = sW;
      sliderH = sH;
      // Use the input value
      valueToPixels(forceValue ?
         getWorkingValueFromInput() :
         (tagName == "select" ? inp.selectedIndex : parseFloat(inp.value)));
    } catch(err) {};
      callback("redraw");
  };

// Calls a callback function
function callback(type) {
  if(!html5Shim) {
   if(callbacks.hasOwnProperty(type)) {
     var cbObj = {"disabled":disabled, "elem":inp, "value":tagName == "select" ?
                  inp.options[inp.selectedIndex].value : inp.value};
     // Call all functions in sequence
     for(var i = 0, func; func = callbacks[type][i]; i++) {
      func.call(inp, cbObj);
     };
   };
  }else if(type.match(/^(blur|focus|change)$/i)) {
     if(typeof(document.createEventObject) != 'undefined') {
       try {
         var e = document.createEventObject();
         inp.fireEvent('on' + type.toLowerCase(), e);
       } catch(err){ };
     } else if(typeof(document.createEvent) != 'undefined') {
          var e = document.createEvent('HTMLEvents');
          e.initEvent(type, true, true);
          inp.dispatchEvent(e);
     };
   };
  };

  // FOCUS & BLUR events
  function onFocus(e) {
    addClass(outerWrapper, 'fd-slider-focused');
    // Is the value said to have been set by the user onfocus
    if(varSetRules.onfocus) {
      userSet = true;
      valueToPixels(getWorkingValueFromInput());
    };
    // If mousewheel events required then add them
    if(mouseWheelEnabled) {
       addEvent(window, 'DOMMouseScroll', trackMouseWheel);
       addEvent(document, 'mousewheel', trackMouseWheel);
       if(!isOpera) {
          addEvent(window,    'mousewheel', trackMouseWheel);
       };
     };

    // Callback...
     callback("focus");
       return true;
  };
```

```
 function onBlur(e) {
  removeClass(outerWrapper, 'fd-slider-focused');
  // Remove mousewheel events if necessary
  if(mouseWheelEnabled) {
    removeEvent(document, 'mousewheel', trackMouseWheel);
    removeEvent(window, 'DOMMouseScroll', trackMouseWheel);
    if(!isOpera) {
        removeEvent(window,   'mousewheel', trackMouseWheel);
    };
  };
  kbEnabled = true;
  // Callback...
  callback("blur");
 };
// MOUSEWHEEL events
function trackMouseWheel(e) {
 if(!kbEnabled) {
  return;
 };
 e = e || window.event;
 var delta = 0,
 value;
 if (e.wheelDelta) {
  delta = e.wheelDelta/120;
  // Older versions of Opera require a small hack to inverse the delta
  if (isOpera && window.opera.version() < 9.2) {
     delta = -delta;
  };
 } else if(e.detail) {
    delta = -e.detail/3;
 };
 if(vertical) {
   delta = -delta;
 };
 if(delta) {
   value = getWorkingValueFromInput();
   value += (delta < 0) ? -step : step;
   userSet = true;
   valueToPixels(getValidValue(value));
 };
 return stopEvent(e);
};

// KEYBOARD events
function onKeyPress(e) {
 e = e || window.event;
// Let all non-hijacked keyboard events pass
 if((e.keyCode >= 33 && e.keyCode <= 40) || !kbEnabled || e.keyCode == 45 || e.keyCode == 46) {
  return stopEvent(e);
 };
 return true;
};

function onKeyDown(e) {
  if(!kbEnabled) {
     return true;
  };
e = e || window.event;
var kc = e.keyCode != null ? e.keyCode : e.charCode,
    value;
    if(kc < 33 || (kc > 40 && (kc != 45 && kc != 46))) {
           return true;
    };
```

```
value = getWorkingValueFromInput();
if( kc == 37 || kc == 40 || kc == 46 || kc == 34) {
    // left, down, ins, page down
    value -= (e.ctrlKey || kc == 34 ? +maxStep : +step);
} else if( kc == 39 || kc == 38 || kc == 45 || kc == 33) {
    // right, up, del, page up
    value += (e.ctrlKey || kc == 33 ? +maxStep : +step);
} else if( kc == 35 ) {
   // max
   value = rMax;
} else if( kc == 36 ) {
  // min
  value = rMin;
};
 userSet = true;
 valueToPixels(getValidValue(value));
 callback("update");
 // Opera doesn't let us cancel key events so the up/down arrows
 //and home/end buttons will scroll the screen - which sucks
 preventDefault(e);
};
// MOUSE & TOUCH events
// Mouseover the slider
function onMouseOver(e) {
  addClass(outerWrapper, 'fd-slider-hover');
};
// Mouseout of the slider
function onMouseOut(e) {
 // Should really check we are not still in the slider
 removeClass(outerWrapper, 'fd-slider-hover');
};

// Mousedown on the slider
function onMouseDown(e) {
   e = e || window.event;
   // Stop page scrolling
  preventDefault(e);
  // Grab the event target
  var targ;
  if (e.target) {
      targ = e.target;
  } else if (e.srcElement) {
      targ = e.srcElement;
  };
  if(targ && targ.nodeType == 3) {
      targ = targ.parentNode;
  };
  // Are we using touchEvents
  if(e.touches) {
    // Skip gestures
    if(e.targetTouches && e.targetTouches.length != 1) {
        return false;
    };
    e = e.touches[0];
    touchEvents = true;
  };

  // Stop any animation timers
    clearTimeout(timer);
    timer = null;
 // Not keyboard enabled
    kbEnabled = false;
 // User has set a value
    userSet   = true;
```

```
  // Handle mousedown - initiate drag
    if(targ.className.search("fd-slider-handle") != -1) {
       mousePos  = vertical ? e.clientY : e.clientX;
       handlePos = parseInt(vertical ? handle.offsetTop : handle.offsetLeft)||0;
       // Set a value on first click even if no movement
       trackMouse(e);
         if(!touchEvents) {
           addEvent(document, 'mousemove', trackMouse);
           addEvent(document, 'mouseup', stopDrag);
         } else {
           addEvent(document, 'touchmove', trackMouse);
           addEvent(document, 'touchend', stopDrag);
        // Remove mouseEvents to stop them firing after the touch event
           removeEvent(outerWrapper, "mousedown", onMouseDown);
         };
         addClass(outerWrapper, 'fd-slider-active');
         addClass(document.body, "fd-slider-drag-" + (vertical ? "vertical" : "horizontal"));
         callback("dragstart");
        // Wrapper mousedown - initiate animation to click point
    }else {
        locate();
        var posx= 0,
        sLft= 0,
        sTop= 0;
        // Internet Explorer doctype woes
        if(document.documentElement && document.documentElement.scrollTop) {
         sTop = document.documentElement.scrollTop;
         sLft = document.documentElement.scrollLeft;
        } else if (document.body) {
         sTop = document.body.scrollTop;
         sLft = document.body.scrollLeft;
        };
        if(e.pageX) {
         posx = vertical ? e.pageY : e.pageX;
        } else if (e.clientX) {
         posx = vertical ? e.clientY + sTop : e.clientX + sLft;
        };
        posx -= vertical ?
        y + Math.round(handle.offsetHeight / 2) : x + Math.round(handle.offsetWidth / 2);
        posx = snapToPxValue(posx);
        // Tween animation to click point
        if(animation == "tween") {
           addClass(outerWrapper, 'fd-slider-active');
           tweenTo(posx);
           // Progressive increment to click point
        } else if(animation == "timed") {
           addClass(outerWrapper, 'fd-slider-active');
           addEvent(document, touchEvents ?
                'touchend' : 'mouseup',onDocMouseUp);
           destPos = posx;
           onTimer();
           // Immediate jump to click point
        } else {
           pixelsToValue(posx);
         //addEvent(document, touchEvents ? 'touchend' : 'mouseup', onMouseUp);
        };
   };
   return stopEvent(e);
};

 // Progressive increment to click point - clear the animation timer
//and remove the mouseup/touchend event
 function onDocMouseUp( e ) {
    e = e || window.event;
```

```
    preventDefault(e);
    removeEvent(document, touchEvents ? 'touchend' : 'mouseup', onDocMouseUp);
    removeClass(outerWrapper, "fd-slider-active");
    clearTimeout(timer);
    timer     = null;
    kbEnabled = true;
    return stopEvent(e);
};
// Mouseup or touchend event on the document to stop drag
    function stopDrag(e) {
      e = e || window.event;
      preventDefault(e);
      if(touchEvents) {
       removeEvent(document, 'touchmove', trackMouse);
       removeEvent(document, 'touchend',   stopDrag);
      } else {
       removeEvent(document, 'mousemove', trackMouse);
       removeEvent(document, 'mouseup',   stopDrag);
      };
      kbEnabled   = true;
      removeClass(document.body, "fd-slider-drag-" + (vertical ? "vertical" : "horizontal"));
      removeClass(outerWrapper, "fd-slider-active");
      callback("dragend");
      return stopEvent(e);
};
// Mousemove or touchmove event on the drag handle
function trackMouse(e) {
  e = e || window.event;
  preventDefault(e);
  if(e.touches) {
  // Skip gestures
    if(e.targetTouches && e.targetTouches.length != 1) {
      return false;
    };
    e = e.touches[0];
  };
  pixelsToValue(snapToPxValue(handlePos + (vertical ? e.clientY - mousePos : e.clientX - mousePos)));
    return false;
 };
  // Increments the slider by "inc" steps
 function increment(inc) {
   var value = getWorkingValueFromInput();
   userSet   = true;
   value += inc * step;
   valueToPixels(getValidValue(value));
 };
 // Attempts to locate the on-screen position of the slider
 function locate(){
  var curleft = 0,
  curtop  = 0,
  obj     = outerWrapper;
  // Try catch for IE's benefit
  try {
    while (obj.offsetParent) {
     curleft += obj.offsetLeft;
     curtop  += obj.offsetTop;
     obj = obj.offsetParent;
    };
  } catch(err) {};
    x = curleft;
    y = curtop;
  };
 // Used during the progressive animation to click point
 function onTimer() {
```

```
  var xtmp = parseInt(vertical ? handle.offsetTop : handle.offsetLeft, 10);
  xtmp = Math.round((destPos < xtmp) ?
          Math.max(destPos, Math.floor(xtmp - stepPx)) :
          Math.min(destPos, Math.ceil(xtmp + stepPx)));
  pixelsToValue(snapToPxValue(xtmp));
  if(xtmp != destPos) {
     timer = setTimeout(onTimer, steps > 20 ? 50 : 100);
  } else {
     kbEnabled = true;
     removeClass(outerWrapper, "fd-slider-active");
     callback("finalise");
  };
};
var tween = function(){
    frame++;
    var c = tweenC,
    d = 20,
    t = frame,
    b = tweenB,
    x = Math.ceil((t==d) ?
        b+c :
        c * (-Math.pow(2, -10 * t/d) + 1) + b);
    pixelsToValue(t == d ? tweenX : x);
    if(t!=d) {
     // Call the "move" callback on each animation increment
      callback("move");
      timer = setTimeout(tween, 20);
    } else {
      clearTimeout(timer);
      timer      = null;
      kbEnabled = true;
      removeClass(outerWrapper, "fd-slider-focused");
      removeClass(outerWrapper, "fd-slider-active");
    // Call the "finalise" callback whenever the animation is complete
      callback("finalise");
    };
};

function tweenTo(tx){
   kbEnabled = false;
   tweenX    = parseInt(tx, 10);
   tweenB    = parseInt(vertical ? handle.offsetTop : handle.offsetLeft, 10);
   tweenC    = tweenX - tweenB;
   tweenD    = 20;
   frame     = 0;
   if(!timer) {
      timer = setTimeout(tween, 20);
   };
};

// Returns a value within the range & sets the userSet var
// i.e. has the user entered a valid value
function checkValue(value) {
 if(isNaN(value) || value === "" || typeof value == "undefined") {
    userSet = false;
    return defaultVal;
 } else if(value < Math.min(rMin,rMax)) {
    userSet = false;
    return Math.min(rMin,rMax);
 } else if(value > Math.max(rMin,rMax)) {
    userSet = false;
    return Math.max(rMin,rMax);
 };
    userSet = true;
```

```
    return value;
};

// Returns a value within a range - uses the form element value as base
 function getWorkingValueFromInput() {
    return getValidValue(tagName == "input" ? parseFloat(inp.value) : inp.selectedIndex);
 };

// Returns a value within the range
 function getValidValue(value) {
   return (isNaN(value) || value === "" || typeof value == "undefined") ?
        defaultVal : Math.min(Math.max(value, Math.min(rMin,rMax)), Math.max(rMin,rMax));
 };

// Calculates value according to pixel position of slider handle
 function pixelsToValue(px) {
   var val = getValidValue(scale ? percentToValue(pixelsToPercent(px)) : vertical
       ? max - (Math.round(px / stepPx) * step) : min + (Math.round(px / stepPx) * step));
   handle.style[vertical ? "top" : "left"] = (px || 0) + "px";
   redrawRange();
   setInputValue((tagName == "select" || step == 1) ? Math.round(val) : val);
};

// Calculates pixel position according to form element value
function valueToPixels(val) {
   var clearVal = false,
   value;
   // Allow empty values for non-polyfill sliders
   if((typeof val == "undefined" || isNaN(val) || val === "")
       && tagName == "input" && !forceValue) {
     value    = defaultVal;
     clearVal = true;
     userSet  = false;
   } else {
     value = checkValue(val);
   };
   handle.style[vertical ? "top" : "left"] =
       (scale ? percentToPixels(valueToPercent(value)) : vertical ?
               Math.round(((max - value) / step) * stepPx) :
               Math.round(((value - min) / step) * stepPx)) + "px";
   redrawRange();
   setInputValue(clearVal ? "" :value);
};

// Rounds a pixel value to the nearest "snap" point on the slider scale
function snapToPxValue(px) {
  if(scale) {
       return Math.max(Math.min(rMaxPx, px), rMinPx);
  } else {
       var rem = px % stepPx;
       if(rem && rem >= (stepPx / 2)) {
          px += (stepPx - rem);
       } else {
          px -= rem;
       };
       if(px < Math.min(Math.abs(rMinPx), Math.abs(rMaxPx))) {
          px = Math.min(Math.abs(rMinPx), Math.abs(rMaxPx));
       } else if(px > Math.max(Math.abs(rMinPx), Math.abs(rMaxPx))) {
            px = Math.max(Math.abs(rMinPx), Math.abs(rMaxPx));
       };
    return Math.min(Math.max(px, 0), rMaxPx);
  };
};
```

```
 // Calculates a value according to percentage of distance handle has travelled
 function percentToValue(pct) {
   var st = 0,
   fr = min,
   value;
   for(var s in scale) {
    if(!scale.hasOwnProperty(s)) {
      continue;
    };
    if(pct >= st && pct <= +s ) {
     value = fr + ((pct - st) * (+scale[s] - fr) ) / (+s - st);
    };
    st = +s;
    fr = +scale[s];
   };
   return value;
 };

// Calculates the percentage handle position according to form element value
   function valueToPercent(value) {
      var st  = 0,
      fr  = min,
      pct = 0;
      for(var s in scale) {
         if(!scale.hasOwnProperty(s)) {
         continue;
      };
      if(value >= fr && value <= +scale[s]){
        pct = st + (value - fr) * (+s - st) / (+scale[s] - fr);
      };
      st = +s;
      fr = +scale[s];
   };

   return pct;
  };

  function percentToPixels(percent) {
     return ((outerWrapper[vertical ? "offsetHeight" : "offsetWidth"] -
                handle[vertical ? "offsetHeight" : "offsetWidth"]) / 100) * percent;
  };

  function pixelsToPercent(pixels) {
    return pixels / ((outerWrapper[vertical ? "offsetHeight" : "offsetWidth"]
        - outerWrapper[handle ? "offsetHeight" : "offsetWidth"]) / 100);
  };

  // Sets the form element with a valid value
  function setInputValue(val) {
  // The update callback doesn't mean the input value has changed
    callback("update");
  // If the user has not set this value or has entered an incorrect value then set a class
  // to enable styling of the slider
    if(!userSet) {
         addClass(outerWrapper, "fd-slider-no-value");
    } else {
         removeClass(outerWrapper, "fd-slider-no-value");
    };
    if(tagName == "select") {
     try {
      val = parseInt(val, 10);
      if(inp.selectedIndex === val) {
        updateAriaValues();
        return;
```

```
      };
      inp.options[val].selected = true;
    } catch (err) {};
  } else {
   if(val != "") {
     val = (min + (Math.round((val - min) / step) * step)).toFixed(precision);
   };
   if(inp.value === val) {
     updateAriaValues();
     return;
   };
   inp.value = val;
 };
 updateAriaValues();
 callback("change");
};
  function checkInputValue(value) {
     return !(isNaN(value) || value === "" ||
             value < Math.min(rMin,rMax) || value > Math.max(rMin,rMax));
  };

  function setSliderRange(newMin, newMax) {
    if(rMin > rMax) {
      newMin = Math.min(min, Math.max(newMin, newMax));
      newMax = Math.max(max, Math.min(newMin, newMax));
      rMin   = Math.max(newMin, newMax);
      rMax   = Math.min(newMin, newMax);
    } else {
      newMin = Math.max(min, Math.min(newMin, newMax));
      newMax = Math.min(max, Math.max(newMin, newMax));
      rMin   = Math.min(newMin, newMax);
      rMax   = Math.max(newMin, newMax);
    };
    if(defaultVal < Math.min(rMin, rMax)) {
      defaultVal = Math.min(rMin, rMax);
    } else if(defaultVal > Math.max(rMin, rMax)) {
      defaultVal = Math.max(rMin, rMax);
    };
    handle.setAttribute("aria-valuemin",  rMin);
    handle.setAttribute("aria-valuemax",  rMax);
    checkValue(tagName == "input" ? parseFloat(inp.value) : inp.selectedIndex);
    redraw();
  };

  function redrawRange() {
    if(noRangeBar) {
       return;
    };
    if(vertical) {
       rangeBar.style["height"] = (bar.offsetHeight - handle.offsetTop) + "px";
    } else {
       rangeBar.style["width"] = handle.offsetLeft + "px";
    };
  };

  function findLabel() {
    var label = false,
    labelList = document.getElementsByTagName('label');
    // loop through label array attempting to
    //match each 'for' attribute to the id of the current element
    for(var i = 0, lbl; lbl = labelList[i]; i++) {
    // Internet Explorer requires the htmlFor test
      if((lbl['htmlFor'] && lbl['htmlFor'] == inp.id) ||
        (lbl.getAttribute('for') == inp.id)) {
```

```
        label = lbl;
        break;
    };
  };
  if(label && !label.id) {
     label.id = inp.id + "_label";
  };
  return label;
};

function updateAriaValues() {
   handle.setAttribute("aria-valuenow",  tagName == "select" ?
             inp.options[inp.selectedIndex].value : inp.value);
   handle.setAttribute("aria-valuetext", tagName == "select" ?
             (inp.options[inp.selectedIndex].text ?
              inp.options[inp.selectedIndex].text :
              inp.options[inp.selectedIndex].value) : inp.value);
};

function onInputChange(e) {
   userSet = true;
   valueToPixels(tagName == "input" ? parseFloat(inp.value) : inp.selectedIndex);
   updateAriaValues();
};

function onReset(e) {
   if(tagName == "input") {
      inp.value = inp.defaultValue;
   } else {
      inp.selectedIndex = resetDef;
   };
   checkValue(tagName == "select" ? inp.options[inp.selectedIndex].value : inp.value);
   redraw();
   updateAriaValues();
};

function valueSet(tf) {
    userSet = !!tf;
};

// Sets a tabindex attribute on an element, bends over for IE.
function setTabIndex(e, i) {
   e.setAttribute(!/*@cc_on!@*/false ? "tabIndex" : "tabindex", i);
   e.tabIndex = i;
};

(function() {
   if(html5Shim || hideInput) {
     addClass(inp, "fd-form-element-hidden");
   } else {
     addEvent(inp, 'change', onInputChange);
   };
   // Add stepUp & stepDown methods to input element if using the html5Shim
   if(html5Shim) {
     inp.stepUp   = function(n) { increment(n||1); };
     inp.stepDown = function(n) { increment(n||-1); };
   };
   outerWrapper= document.createElement('span');
   outerWrapper.className = "fd-slider" +
             (vertical ? "-vertical " : " ") +
             (!html5Shim ? " fd-slider-no-value " : "") + classNames;
                    outerWrapper.id           = "fd-slider-" + inp.id;
```

```
 if(vertical && inpHeight) {
        outerWrapper.style.height = inpHeight + "px";
 };
 wrapper= document.createElement('span');
 wrapper.className = "fd-slider-inner";
 bar = document.createElement('span');
 bar.className= "fd-slider-bar";
 if(!noRangeBar) {
   rangeBar = document.createElement('span');
   rangeBar.className= "fd-slider-range";
 };
 if(fullARIA) {
   handle= document.createElement('span');
 } else {
   handle= document.createElement('a');
   handle.setAttribute("href", "#");
   addEvent(handle, "click", stopEvent);
 };
 setTabIndex(handle, 0);
 handle.className = "fd-slider-handle";
 handle.appendChild(document.createTextNode(String.fromCharCode(160)));
 outerWrapper.appendChild(wrapper);
 if(!noRangeBar) {
   outerWrapper.appendChild(rangeBar);
 };
 outerWrapper.appendChild(bar);
 outerWrapper.appendChild(handle);
 inp.parentNode.insertBefore(outerWrapper, inp);
// Add ARIA accessibility info programmatically
 outerWrapper.setAttribute("role",           "application");
 handle.setAttribute("role",           "slider");
 handle.setAttribute("aria-valuemin",  tagName == "select" ?
                    inp.options[0].value : min);
 handle.setAttribute("aria-valuemax",  tagName == "select" ?
           inp.options[inp.options.length - 1].value : max);
 var lbl = findLabel();
 if(lbl) {
   handle.setAttribute("aria-labelledby", lbl.id);
   handle.id = "fd-slider-handle-" + inp.id;
   /*@cc_on
   /*@if(@_win32)
   lbl.setAttribute("htmlFor", handle.id);
   @else @*/
   lbl.setAttribute("for", handle.id);
   /*@end
   @*/
 };
// Are there page instructions
if(document.getElementById(describedBy)) {
   handle.setAttribute("aria-describedby", describedBy);
};
// Is the form element initially disabled
if(inp.getAttribute("disabled") == true) {
       disableSlider(true);
} else {
       enableSlider(true);
};
// Does an initial form element value mean the user has set a valid value?
// Also called onload in case browsers have automatically set the input value
if(varSetRules.onvalue) {
    userSet = true;
    checkValue(tagName == "input" ? parseFloat(inp.value) : inp.selectedIndex);
};
if(inp.form) {
```

```
            addEvent(inp.form, "reset", onReset);
        };
        updateAriaValues();
        callback("create");
        redraw();
    })();
    return {
        onResize:function(e) { if(outerWrapper.offsetHeight != sliderH ||
                outerWrapper.offsetWidth != sliderW) { redraw(); }; },
        destroy:function()  { destroySlider(); },
        reset:function()  {
            valueToPixels(tagName == "input" ? parseFloat(inp.value) : inp.selectedIndex); },
        stepUp:function(n) { increment(Math.abs(n)||1); },
        stepDown:function(n) { increment(-Math.abs(n)||-1); },
        increment:function(n) { increment(n); },
        disable:function()  { disableSlider(); },
        enable:function()  { enableSlider(); },
        setRange:function(mi, mx) { setSliderRange(mi, mx); },
        getValueSet:function() { return !!userSet; },
        setValueSet:function(tf) { valueSet(tf); },
        checkValue:function() {
            if(varSetRules.onvalue) {
                    userSet = true;
                    checkValue(tagName == "input" ?
                            parseFloat(inp.value) : inp.selectedIndex);
            }; updateAriaValues(); redraw(); }
    };
};

addEvent(window, "load",   init);
addEvent(window, "load",   function() {
        setTimeout(function() {
           var slider;
           for(slider in sliders) {
                sliders[slider].checkValue();
           }
         }, 0);
});
addEvent(window, "resize", resize);
addEvent(window, "unload", unload);

// Have we been passed JSON within the including script tag
(function() {
  var scriptFiles= document.getElementsByTagName('script'),
  scriptInner= String(scriptFiles[scriptFiles.length - 1].innerHTML)
        .replace(/[\n\r\s\t]+/g, "␣").replace(/^\s+/, "").replace(/\s+$/, ""),
  json= parseJSON(scriptInner);
  if(typeof json === "object" && !("err" in json)) {
     affectJSON(json);
  };
})();

return {
  createSlider:function(opts) { return createSlider(opts); },
  onDomReady:function() { onDomReady(); },
  destroyAll:function() { destroyAllsliders(); },
  destroySlider:function(id) { return destroySingleSlider(id); },
  redrawAll:function() { resize(); },
  addEvent:addEvent,
  removeEvent:removeEvent,
  stopEvent:stopEvent,
  increment: function(id, numSteps) {
     if(!sliderExists(id)) {
        return false;
```

```
        };
        sliders[id].increment(numSteps);
    },
    stepUp:function(id, n) {
        if(!sliderExists(id)) {
            return false;
        };
        sliders[id].stepUp(Math.abs(n)||1);
    },
    stepDown:function(id, n) {
        if(!sliderExists(id)) {
            return false;
        };
        sliders[id].stepDown(-Math.abs(n)||-1);
    },
    setRange:function(id, newMin, newMax) {
        if(!sliderExists(id)) {
            return false;
        };
        sliders[id].setRange(newMin, newMax);
    },
    updateSlider:function(id) {
        if(!sliderExists(id)) {
          return false;
        };
        sliders[id].reset();
    },
    disable:function(id) {
        if(!sliderExists(id)) {
            return false;
        };
        sliders[id].disable();
    },
    enable:function(id) {
        if(!sliderExists(id)) {
            return false;
        };
        sliders[id].enable();
    },
    getValueSet:function() {
        return getValueSet();
    },
    setValueSet:function(a, tf) {
        if(!sliderExists(id)) {
            return false;
        };
        setValueSet(a, tf);
    },
    setGlobalVariables: function(json) {
        affectJSON(json);
    },
    removeOnload:function() {
        removeOnLoadEvent();
    }
 };
})
();

/*******************************************************
*   Functions used to show the plots of the feedback    *
*******************************************************/

/*Function called after the preferences are sent to Java
this function collects the feedback information in order
```

```
to draw a plot of the feedback*/
function updates(){
/*The setTimeout() method will wait the specified number
 of milliseconds, and then execute the specified function.*/
        setTimeout(feedclient,100);
        setTimeout(feedserver,100);
        clickgraphServer();
        clickgraphClient()
}

/*Function that get the feedback as client that Java retrieve to user
through the web interface*/
function feedclient(){
 //local variables
 var timeCl,valAdCl,valSatCl,valEvaCl;
 timeCl=time*10;
 valAdCl=parseFloat(document.getElementById("Peer-as-client
␣Adequationeclipse-jee-indigo-SR2-linux-gtk-x86_64.tar.gz").value);
 valSatCl=parseFloat(document.getElementById("Peer-as-client
␣Satisfactioneclipse-jee-indigo-SR2-linux-gtk-x86_64.tar.gz").value);
 valEvaCl=parseFloat(document.getElementById("Peer-as-client␣System
Evaluationeclipse-jee-indigo-SR2-linux-gtk-x86_64.tar.gz").value);
 //Storage the values in the total arrays
 Timeclient.push(timeCl);
 Adecuationclient.push(valAdCl);
 Satisfactionclient.push(valSatCl);
 Evaluationclient.push(valEvaCl);
 //We used an array whit size 5 to
 //show the partial feedback
 //If the size of the temporal array is 5, we need
 //eliminate the first element (pop)
 if(Timeclient.length>5){
    TimeAux.splice(0,1);
    AdecuationclientAux.splice(0,1);
    SatisfactionclientAux.splice(0,1);
    EvaluationclientAux.splice(0,1);
 }
 //And insert the last value at the end
 TimeAux.push(timeCl);
 AdecuationclientAux.push(valAdCl);
 SatisfactionclientAux.push(valSatCl);
 EvaluationclientAux.push(valEvaCl);
 time=time+1;
 //Call to the function to draw the plot in the canvas.
 graphicFeedbackClient(TimeAux,AdecuationclientAux,SatisfactionclientAux,EvaluationclientAux);
 setTimeout(feedclient,10000);
}

/*Function to use the RGraph tool to draw a plot of feedback
 as client in a canvas with id= "myLine1"*/
function graphicFeedbackClient(data2, data1,data3,data4){
 var line1;
 var b1= document.getElementById("myLine1");
 RGraph.Clear(b1);
 RGraph.ObjectRegistry.Clear(b1);
 line1 = new RGraph.Line("myLine1",data1,data3,data4);
 line1.Set('chart.labels',data2);
 line1.Set('chart.tickmarks', 'circle');
 line1.Set('chart.key', ['Adequation', 'Satisfaction', 'System␣Evaluation']);
 line1.Set('chart.gutter.top', 45);
 line1.Set('chart.gutter.bottom', 45);
 line1.Set('chart.key.position.graph.boxed', false);
 line1.Set('chart.key.position', 'gutter');
 line1.Set('chart.title', 'Feedback␣as␣client');
```

```
 line1.Set('chart.title.vpos', 0.3)
 line1.Draw();
}

/*Function that get the feedback as client that Java retrieve to user
through the web interface*/
function feedserver(){
//local variables
  var valAdSV,valSatSV,valEvaSV;
  valAdSV=parseFloat(document.getElementById("Peer-as-server
␣Adequationeclipse-jee-indigo-SR2-linux-gtk-x86_64.tar.gz").value);
  valSatSV=parseFloat(document.getElementById("Peer-as-server
␣Satisfactioneclipse-jee-indigo-SR2-linux-gtk-x86_64.tar.gz").value);
  valEvaSV=parseFloat(document.getElementById("Peer-as-server␣System
␣Evaluationeclipse-jee-indigo-SR2-linux-gtk-x86_64.tar.gz").value);
 //Storage the values in the total arrays
  Adecuationserver.push(valAdSV);
  Satisfactionserver.push(valSatSV);
  Evaluationserver.push(valEvaSV);
 //We used an array whit size 5 to
 //show the partial feedback
 //If the size of the temporal array is 5, we need
 //eliminate the first element (pop)
  if(Timeclient.length>5){
    AdecuationserverAux.splice(0,1);
    SatisfactionserverAux.splice(0,1);
    EvaluationserverAux.splice(0,1);
  }
  AdecuationserverAux.push(valAdSV);
  SatisfactionserverAux.push(valSatSV);
  EvaluationserverAux.push(valEvaSV);
 //And insert the last value at the end
  graphicFeedbackserver(TimeAux, AdecuationserverAux,SatisfactionserverAux,EvaluationserverAux);
  setTimeout(feedserver,10000);
}

/*Function that use the RGraph tool to draw a plot
 of feedback as server in a canvas with id= "myLine2"*/
function graphicFeedbackserver(data2, data1,data3,data4){
        var line;
        var b= document.getElementById("myLine2");
        RGraph.Clear(b);
        RGraph.ObjectRegistry.Clear(b);
        line = new RGraph.Line("myLine2",data1,data3,data4);
        line.Set('chart.labels',data2);
        line.Set('chart.tickmarks', 'circle');
        line.Set('chart.key', ['Adequation', 'Satisfaction', 'System␣Evaluation']);
        line.Set('chart.gutter.top', 45);
        line.Set('chart.gutter.bottom', 45);
        line.Set('chart.key.position.graph.boxed', false);
        line.Set('chart.key.position', 'gutter');
        line.Set('chart.title', 'Feedback␣as␣server');
        line.Set('chart.title.vpos', 0.3)
        line.Draw();
}

/*Function that use RGraph tool to draw a plot of the feedback
(general, as client and as server) in the canvas with id="line6"*/
function total(tiempo,adCl,saCl,sysCL,adSer,saSer,sysSer ){
   var line6;
   var b= document.getElementById("line6");
   RGraph.Clear(b);
   RGraph.ObjectRegistry.Clear(b);
   line6 = new RGraph.Line("line6",adCl,saCl,sysCL,adSer,saSer,sysSer)
```

```
    line6.Set('chart.labels',tiempo);
    line6.Set('chart.tickmarks', 'circle');
    line6.Set('chart.key', ['Adequation Client',
              'Satisfaction Client',
              'System Evaluation Client',
              'Adequation Server',
              'Satisfaction Server',
              'System Evaluation Server']);
    line6.Set('chart.shadow', true);
    line6.Set('chart.shadow.offsetx', 1);
    line6.Set('chart.shadow.offsety', 1);
    line6.Set('chart.shadow.blur', 3);
    line6.Set('chart.hmargin', 15);
    line6.Set('chart.gutter.top', 45);
    line6.Set('chart.gutter.bottom', 45);
    line6.Set('chart.key.position.graph.boxed', false);
    line6.Set('chart.key.position', 'gutter');
    line6.Set('chart.title', 'Total Feedback');
    line6.Set('chart.title.vpos',0.3);
    line6.Draw();
}

/*********************************************************
Functions used to open the window that shows the general
feedback, when the user click on the canvas of the partial
feedback no matter if is feedback as client or as server
*********************************************************/

/*Function executed when the user click on the canvas that
shows the feedback as client*/
function clickgraphClient(){
  var canvas=document.getElementById("myLine1");
  canvas.addEventListener("click", function (e) {
     total(Timeclient,Adecuationclient,Satisfactionclient,Evaluationclient,
           Adecuationserver,Satisfactionserver,Evaluationserver);
     $( "#general" ).dialog( "open" );
    }, true);
}

/*Function executed when the user click on the canvas that
shows the feedback as client*/
function clickgraphServer(){
   var canvas=document.getElementById("myLine2");
   canvas.addEventListener("click", function (e) {
       total(Timeclient,Adecuationclient,Satisfactionclient,Evaluationclient,
             Adecuationserver,Satisfactionserver,Evaluationserver);
       $( "#general" ).dialog( "open" );
   }, true);
}
/*********************************************************
        Function that use jquery ui
*********************************************************/
/*Function that uses jquery.ui, in order
to present a pop-up windows that contains
the preferences and the general feedback*/
$(function() {
   $("#rightcolumn").dialog({
     autoOpen: false,
     height: 300,
     width: 700,
     modal:true,
     show: {
       effect: "scale",
     },
```

```
      hide: {
        effect: "scale",
      }
    });
    $( "#general" ).dialog({
      autoOpen: false,
      height: 500,
      width: 1000,
      modal:true,
      show: {
        effect: "scale",
      },
      hide: {
        effect: "scale",
      }
    });
});
```

Appendix C
Applet Code

C.1 JavaSocket.java

```
import java.applet.*;
import javax.swing.*;
import netscape.javascript.*;
import java.net.*;
import java.io.*;

public class JavaSocket extends JApplet {
// Attributes Declaration
// Access to the browser
JSObject browser = null;
// socket to communication
Socket socket = null;
// Output (representations of objects to a text-output stream).
PrintWriter out = null;
// Instance of the Listener for input
Listener listener = null;
//Flag to identify state
boolean running = false;
// Address where the socket is  connected
String address = null;
// Number Port
int port = -1;
//Flag to identify if the connection is done
boolean connectionDone = false;

// Method to initialize the applet
public void init(){
/*Allows Java to manipulate objects
that are defined in JavaScript.*/
        browser = JSObject.getWindow(this);
}

// Method to stop the applet
public void stop(){
/*Change the value of the flag
 and call the method disconnect*/
        running = false;
        disconnect();
}

//Destroy the applet
public void destroy(){
/*Change the value of the flag
and call the method disconnect*/
```

```
        running = false;
        disconnect();
}

//Start the applet
public void start(){
/*Access to the browse and call
the function "appletReady" in
javascript and modify value of the running flag.*/
        browser.call("appletReady", null);
        running = true;
        while(running){
        // Wait
                try{
        //Sleep threat that execute this method.
                    Thread.sleep(100);
                }
                catch(Exception ex){
                    running = false;
                    return;
                }
        /*Verify the state of the socket,
        and the value of address and port*/
                if(address != null && port != -1 && socket == null){
                    do_connect(address, port);
                }
        }
}

// Connect the socket
public boolean connect(String url, int p){
        address = url;
        port = p;
/* Wait for the connection to
 happen in the main thread */
        connectionDone = false;
        while(!connectionDone){
                try{
        //Sleep the thead that execute this method.
                        Thread.sleep(100);
                }
                catch(Exception ex){
                     return false; }
                }
        connectionDone = false;
/*Verify the state of socket and return a
true or false, depend if the connection is done.*/
        if(socket != null) return true;
        return false;
}

//Create socket comunication
private void do_connect(String url, int p){
   if(socket == null){
     try{
       //Socket
       socket = new Socket(url, p);
       //Buffer socket
       out = new PrintWriter(socket.getOutputStream());
       //create a objet Listener (new Thread)
       listener = new Listener(socket, this);
       listener.start();
       log("Connected to: "+getUrl());
     }catch(Exception ex){
```

```
error("No connection to:"+url+" on port:"+p+"\n"+ex.getMessage());
        connectionDone = true;
      }
    }else{
        error("Already connected to "+getUrl());
    }
        connectionDone = true;
}

/*Disconnect Applet, this method close
the socket and reset the variables */
public boolean disconnect(){
    if(socket != null){
      try{
         log("Socket Disconnected of : "+getUrl());
         listener.close();
         out.close();
         socket = null;
         address = null;
         port = -1;
         return true;
      }catch(Exception ex){
error("An error occured while closing the socket\n"+ex.getMessage());
         socket = null;
         return false;
      }
}
      return false;
}

// Send a message to Java side
public boolean send(String message){
         if(out != null){
                  try{
          //write message in buffer
                         out.println(message);
          //clean buffer
                         out.flush();
                         log("Message sent: "+message);
                 }catch(Exception ex){
                         error("Not write to socket\n"+ex.getMessage());
                 }
                 return true;
         }else{
                 error("Not connected");
                 return false;
         }
}

// Get message from the socket
public void hear(String message){
//Objet type array
         Object[] arguments = new Object[1];
         arguments[0] = message;
         /*Call methods javascript and passing a
         arguments ( this arguments must be a array)*/
         browser.call("getMessage", arguments);
         log("Message Received: "+message);
}

// Report an error
public void error(String message){
         message = "Error Message: " + message;
         log(message);
```

```
        Object[] arguments = new Object[1];
        arguments[0] = message;
        /*Call methods javascript and passing a
        arguments ( this arguments must be a array)*/
        browser.call("on_socket_error", arguments);
}

// Log something
public void log(String message){
        System.out.println(message);
}

// Get the connected URL
private String getUrl(){
 if(socket == null) return null;
  return socket.getInetAddress().getHostName() +":"+socket.getPort();
 }
}
```

C.2 Listener.java

```
import java.io.*;
import java.net.*;

// Thread which listens for input
public class Listener extends Thread{

// Atribute
//Instance of JavaSocket
JavaSocket parent;
// Listen to this socket
Socket socket;
//Input
BufferedReader in;
//Flag to verify if is running
boolean running = false;

// Constructor
public Listener(Socket s, JavaSocket b) throws IOException{
   parent = b;
   socket = s;
   in = new BufferedReader(new InputStreamReader(s.getInputStream()));
}

// Close
public void close() throws IOException{
        if(running == false) return;
        running = false;
        socket.close();
        in.close();
}

// Main loop
public void run(){
   running = true;
   String str = null;
   while(running){
     try{
        str = in.readLine();
        if(str==null){
           parent.disconnect();
           close();
        }else{
           parent.hear(str);
        }
     }catch(Exception ex){
        if(running){
          parent.error("Error to read socket\n"+ex.getMessage());
          parent.disconnect();
          try{
             close();
          } catch(Exception ex2){}
        }
     }
   }
   try{ close(); } catch(Exception ex){}
}

}
```

Bibliography

[1] Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. **36**(4) (December 2004) 335–371

[2] Yang, Y., H.Chow, A.L., Golubchik, L., Bragg, D.: Improving QoS in Bittorrent-like VoD Systems. In: Proceedings of the 29th Conference on Information Communications, San Diego, California, USA (2010) 2061–2069

[3] Acunto, L.D., Andrade, N., Pouwelse, J.A., Sips, H.J.: Peer Selection Strategies for Improved QoS in Heterogeneous BitTorrent-like VoD Systems. In: 12th IEEE International Symposium on Multimedia. (2010) 89–96

[4] Xiong, X., Song, J., Yue, G., Liu, J., Xie, L.: Survey: Research on QoS of P2P reliable streaming media. JNW **6**(8) (2011) 1114–1121

[5] Quiané-Ruiz, J.A., Lamarre, P., Valduriez, P.: Satisfaction-based query replication-An automatic and self-adaptable approach for replicating queries in the presence of autonomous participants. Distributed and Parallel Databases **30**(1) (2012) 1–26

[6] Biazzini, M., Carvajal-Gomez, R., Pérez-Espinosa, A., Serrano-Alvarado, P., Lamarre, P., Pérez-Cortés, E.: WUW (What Users Want): A service to enhance users' satisfaction in content-based peer-to-peer networks. Technical report, Laboratoire d'Informatique de Nantes Atlantique - LINA , Universidad Autonoma Metropolitana Unidad Iztapalapa - UAM , Laboratoire d'Informatique en Images et Systèmes d'Information - LIRIS (October 2012) 20 pages.

[7] Rouibia, S., Ghareed, M., Parrein, B., Biazzini, M., Carvajal-Gomez, R., Perez-Espinosa, A., Serrano-Alvarado, P.: Towards a Hybrid Client/Server and P2P Architecture for Content Delivery over the Internet. In: Proceeding of the CFIP/NOTERE, Bayonne, France (October 2012) 1

[8] Sandvine:: Intelligent broadband networks., Global internet phenomena report, Fall 2011. `http://www.sandvine.com` (August 2012)

[9] Coulouris, G.F., Dollimore, J., Kindberg, T., Blair, G.: Distributed systems: concepts and design. 5th edn. Addison-Wesley Longman Publishing Co., Inc., USA (2011)

[10] Buyya, R., Pathan, M., Vakali, A.: Content delivery networks, Lecture Notes in Electrical Engineering. 1st. edn. Volume 2. Springer-Verlag, Germany (2008)

[11] Verma, D.C.: Content distribution networks: An engineering approach. 1st. edn. John Wiley & Sons, Inc., USA (2002)

[12] Chang, L.: A survey on modeling peer-to-peer video streaming systems. Technical Report V00687101, University of Victoria, Victoria, BC, Canada (2008)

[13] Liu, Y., Guo, Y., Liang, C.: A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications **1**(1) (March 2008) 18–28

[14] Gheorghe, G., Cigno, R.L., Montresor, A.: Security and privacy issues in P2P streaming systems: A survey. Peer-to-Peer Networking and Applications **4**(2) (2011) 75–91

[15] Biersack, E.W., Carra, D., Lo Cigno, R., Rodriguez, P., Felber, P.: Overlay architectures for file distribution: Fundamental performance analysis for homogeneous and heterogeneous cases. Comput. Netw. **51**(3) (February 2007) 901–917

[16] Seibert, J., Zage, D., Fahmy, S., Nita-Rotaru, C.: Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective. In: The 33rd IEEE Conference on Local Computer Networks, The Conference on Leading Edge and Practical Computer Networking. (2008) 20–27

[17] Klemm, A., Lindemann, C., Waldhorst, O.: A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In: Proceedings of 58th Vehicular Technology Conference. Volume 4. (2003) 2758–2763

[18] Abada, A., Cui, L., Huang, C., Chen, H.H.: A novel path selection and recovery mechanism for manets p2p file sharing applications. In: Procedings of the Wireless Communications and Networking Conference. (2007) 3472–3477

[19] Legout, A., Urvoy-Keller, G., Michiardi, P.: Rarest first and choke algorithms are enough. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, Rio de Janeriro, Brazil (2006) 203–216

[20] Xia, R.L., Muppala, J.K.: A survey of Bittorrent performance. Commun. Surveys Tuts. **12**(2) (April 2010) 140–158

[21] Schlosser, D., Hobfeld, T., Tutschku, K.: Comparison of Robust Cooperation Strategies for P2P Content Distribution Networks with Multiple Source Download. In: Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing, Cambridge, UK (2006) 31–38

[22] Abeni, L., Király, C., Cigno, R.L.: On The Optimal Scheduling of Streaming Applications in Unstructured Meshes. In: Proceedings of the 8th International IFIP-TC 6 Networking Conference, Atlanta, USA (2009) 117–130

[23] Kuo, J.L., Shih, C.H., Chen, Y.C.: A Comprehensive Study of Delivery Strategies with Chunk Scheduling for Mesh P2P live Streaming. In: International Conference on Ubiquitous Information Management and Communication. (2011) 107–113

[24] Abeni, L., Kiraly, C., Cigno, R.L.: Scheduling P2P Multimedia Streams: Can We Achieve Performance and Robustness? In: Proceedings of the 3rd IEEE International Conference on Internet Multimedia Services Architecture and Applications, Bangalore, India (2009) 54–59

[25] Shams, S.M.S., Engelstad, P.E., Kvalbein, A.: Analysis of Peer Selection Algorithms in Cross-Layer P2P Architectures. In: Proceedings of the 3rd IEEE International Conference on Internet Multimedia Services Architecture and Applications, Bangalore, India (2009) 274–279

[26] Sripanidkulchai, K., Ganjam, A., Maggs, B., Zhang, H.: The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points. In: Proceedings of ACM Special Interest Group on Data Communication, Portland, USA (2004) 107–120

[27] Sentinelli, A., Celetto, L., Lefol, D., Palazzi, C.E., Pau, G., Zahariadis, T., Jari, A.: Survey on P2P Overlay Streaming Clients. In: Future Internet Assembly, Prague, Czech Republic (2009) 273–282

[28] Liao, X., Jin, H., Liu, Y., Ni, L.M., Deng, D.: Anysee: Peer-to-peer live streaming. In: 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, Barcelona, Spain (2006)

[29] Wiki: BitTorrent protocol specification v1.0., TheoryOrg. `http://wiki.theory.org/BitTorrentSpecification` (April)

[30] Cohen, B.: Incentives Build Robustness in BitTorrent. In: Proceedings of the 1st. Workshop on Economics of Peer-to-Peer Systems. (2003)

[31] Bertinat, M.E., De Vera, D., Padula, D., Amoza, F.R., Rodríguez-Bocca, P., Romero, P., Rubino, G.: Goalbit: The first free and open source peer-to-peer streaming network. In: Proceedings of the 5th International Latin American Networking Conference, Pelotas, Brazil (2009) 49–59

[32] Ye, Q., Chen, C.: A Study on Topology Model and Data Contribution Strategy of PPLive. In: Proceedings of the 2010 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Washington, USA (2010) 301–304

[33] Li, R., Gao, G., Xiao, W., Xu, Z.: Measurement Study on PPLive Based on Channel Popularity. In: Proceedings of the 9th Annual Communication Networks and Services Research Conference, Washington, USA (2011) 18–25

[34] Liang, W., Bi, J., Wu, R., Li, Z., Li, C.: On Characterizing PPStream: Measurement and Analysis of P2P IPTV under Large-Scale Broadcasting. In: Proceedings of the 28th IEEE Conference on Global Telecommunications, Piscataway, USA (2009) 3552–3557

[35] SopCast. `http://www.sopcast.org/` (April 2013)

[36] Swarmplayer. `http://swarmplayer.p2p-next.org/` (April 2013)

[37] uTorrent. `http://www.utorrent.com` (April 2013)

[38] vidTorrent. `http://web.media.mit.edu/~vyzo/vidtorrent` (April 2013)

[39] Mirkin, I.: Reliable real-time stream distribution using an internet multicast overlay. Master's thesis, Massachusetts Institute of Technology, E.U (2006)

[40] Voulgaris, S., Jelasity, M., van Steen, M.: A robust and scalable peer-to-peer gossiping protocol. In: Proceedings of the 2nd International Conference on Agents and Peer-to-Peer Computing, Heidelberg, Alemania (2004) 47–58

[41] Schulze, H., Mochalski, K.: Internet study 2008/2009. `http://www.ipoque.com/userfiles/file/ipoque-Internet-Study-08-09.pdf` (July 2011)

[42] Park, H., Izhak-Ratzin, R., van der Schaar, M.: Peer-to-Peer Networks - Protocols, Cooperation and Competition. In Zhu, C., Li, Y., Niu, X., eds.: Streaming media architectures, techniques, and applications: Recent advances. IGI Global, Pennsylvania (2010) 262–294

[43] Marin, O., Monnet, S., Thomas, G.: Peer-to-Peer Storage. In Haddad, S., Kordon, F., Pautet, L., Petrucci, L., eds.: Distibuted Systems: Design and Algorithms. John Wiley & Sons, Ltd. (2011) 59–80

[44] LiveConnect: Support in the new java plug-in technology. `http://jdk6.java.net/plugin2/liveconnect/` (April 2013)

[45] Flanagan, D.: Javascript: The definitive guide. 4th edn. O'Reilly & Associates, Inc., Sebastopol, CA, USA (1998)

[46] Grid5000. `https://www.grid5000.fr` (April 2013)

Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA

What Users Want(WUW): un servicio de satisfacción de usuarios orientado a aplicaciones de distribución de contenidos

Tesis que presenta:
Adriana Pérez Espinosa
Para obtener el grado de
Maestra en Ciencias y Tecnologías de la Información

Asesoras:
Dra. Elizabeth Pérez Cortés
Dra. Patricia Serrano Alvarado

Jurado Calificador:

Dra. Reyna Carolina Medina Ramírez	UAM-I
Dr. José G. Rodríguez García	CINVESTAV-IPN
Dr. Manuel Aguilar Cornejo	UAM-I

México, D.F