WUW (What Users Want): A Service to Enhance Users' Satisfaction in Content-Based Peer-to-Peer Networks

Marco Biazzini[1] Raziel Carvajal-Gómez[1] Adriana Pérez-Espinosa[3] Patricia Serrano-Alvarado[1] Philippe Lamarre[2] Elizabeth Pérez Cortés[3]

[1] {*Name.LastName*}@univ-nantes.fr
[2] Philippe.Lamarre@liris.cnrs.fr
[3] {*pea, pece*}@xanum.uam.mx

[1] LINA/Université de Nantes
2, rue de la Houssinière
44322 Nantes, France

[2] Liris/Université de Lyon
37, Rue du Repos
69007 Lyon, France

[3] Universidad Autónoma Metropolitana
Av San Rafael Atlixco No.186
09340 Iztapalapa, México

Peer-to-Peer (P2P) architectures are more and more used in Content Delivery Networks (CDN), because the traditional client-server architectures are burdened by high distribution and maintenance cost, whereas in P2P systems those costs are almost negligible. In general, such applications do not take into account user preferences, other than QoS-related parameters. As users resources are the richness of P2P systems, we think it is important to satisfy their preferences concerning the usage of their resources. In this work we propose WUW (What Users Want), a service to improve users' satisfaction in a personal way. WUW runs on top of unstructured P2P systems, and its main goal is to allow users to strategically impact their local neighborhoods according to their own personal preferences. We present first results of experiments, deployed in a cluster, obtained with the prototype implementation of our service, which runs on top of BitTorrent, the most used file sharing protocol. We show that BitTorrent performances are not affected by the users strategic choices introduced by WUW. The advantage of our approach is that, without loosing performance, users can chose the peers they want to collaborate with according to their personal preferences.

1 Introduction and motivation

Content Delivery Networks (CDN) [1] distribute content to end users as files (multimedia, software, documents), live-streaming, on demand streaming, etc. Peer-to-Peer (P2P) architectures are increasingly used in CDN because traditional client-server architectures generate high distribution and maintenance cost, whereas in P2P systems those costs are almost negligible. Besides, P2P architectures are more performing than client-server ones when high demanded content has to be distributed in short period of time.

P2P systems are highly scalable because in those systems peers share their resources automatically (bandwidth, storage, etc.) and not only download content but also upload content to other peers. Thus the more peers are in the system, the more resources the system has and the more performing it is. The P2P architecture is built on top of a

physical network. Peers are organized in overlays (neighborhoods) composed of peers sharing the same resources, for instance, peers downloading the same file or watching the same tv program.

We consider that peers are under control of users that are autonomous and free persons having rights, preferences, interests, etc. As users' resources are the richness of P2P systems, we think it is important to satisfy their preferences concerning the usage of their resources. Concretely, in a P2P system, users should be able to define the behavior of their software beyond parameters related to the Quality of Service (QoS), like the available bandwidth or the maximum number of connections. By letting the users express their preferences and interests, we make it possible for them to shape their contribution in terms of a "strategy". The concept of strategy defines the basic way for a user to determine which neighbors she considers interesting to trade with. Obviously, strategies may reveal sensitive information about user preferences and wills. So, it is of our concern that the strategies effectively impact the overlays' organization while invading as less as possible the users' privacy.

In general, CDN applications based on P2P architectures do not take into consideration user preferences, but only QoS-related parameters. In this paper we propose WUW (What Users Want), a service that runs on top of unstructured P2P systems, which main goal is to allow the users to strategically impact their local neighborhoods, according to their own personal preferences. In order to both, analyze the way the applied strategies satisfy users expectation and limit the extent to which bad strategies may affect performance, WUW gives a feedback to the users. Users are thus able to evaluate the ongoing behavior of the system with respect to their preferences and modify their strategies to meet their goals.

Our work uses and adapt the Satisfaction-based Query Load Balancing framework (SQLB) [2], originally proposed as a mediator-based mechanism to balance the load of service allocation among consumers and providers. All the same to our purposes, the main goal of this framework is to increase users' satisfaction and motivate them in staying longer in the system. Our objective is to provide an autonomous and distributed service that can be used to support various P2P systems, without negatively affecting their performance.

This research repport is organized as follows. Section 2 introduces related works and gives the necessary concepts to understand the key aspects of our service. Section 3 describes how concepts of the SQLB framework have been used and which are the peculiarities of WUW. In Section 4, we propose the design and the algorithms of the WUW service, while Section 5 details the global system architecture we target. Section 6 shows first experimental results. Finally, Section 7 concludes and draws our ongoing work.

2 Related works

This section presents a brief overview of how existing P2P applications treat users preferences (Section 2.1), an overview of the SQLB, the framework (Section 2.2) and a very basic description of the BitTorrent protocol to summarize the main concepts considered by the current implementation of our service (Section 2.3).

2.1 User preferences on P2P-based applications

As we know, in P2P-based applications users are the main players of the game. Thus, it is important to allow them to express their preferences about which part of the user/peer information (IP address, name, location, etc.) may be revealed to other users/peers, along with their interest in the contents they are exchanging (for instance, movies genre, type of tv programs, etc.).

From the analysis we made of AnySee [3], GoalBit [4], PPLive [5], PPStream [6], SopCast [7], SwarmPlayer [8], uTorrent [9] and vidTorrent [10], we conclude that they only allow users to decide/modify (1) the bandwidth amount for download and upload in KBs, (2) the global maximum number of connections, (3) the maximum number of connections per content and (4) the port number used for incoming connections.

None of these applications allows a users to define a personal strategy (based on personal preferences other than QoS-related) that influences her local neighborhood. None of them alerts the user if her local settings result in a poor QoS neither. In general, the P2P applications choose peers based on their QoS-related performance characteristics.

2.2 SQLB overview

In the context of large-scale open distributed systems, participants generally have different individual interests and are autonomous, meaning that they can join or leave the system at any time, and on their own decision. These two characteristics make inadequate to build a system that only focus on performance; except if each participant is only interested in performance, which usually is not the case. In this context, building a system based on a static *strategy* ((the same for every participant all the time) seems doomed to failure. Certainly, if a participant does not find her personal interests satisfied within the system, there is a good chance she leaves.

In systems relying on the resources provided by participants, it is challenging to make them loyal. Participants who leave take away their resources from the system. If these resources interest other participants they may leave as well. Clearly, it is of main concern to avoid such a "domino effect". In order to do this, systems need to take care of their participants' intentions.

The "satisfaction approach", named SQLB (Satisfaction based Query Load Balancing), has been introduced to address this need [2] oriented to the "query allocation" problem. In this context, each participant computes her *intention* to treat a query, (respectively, to obtain her query be treated by some provider) if she is a provider (respectively a consumer). To do so, it is up to each participant to take into account any information she considers as relevant: general objectives, local context (capacity, actual load, task's load...), history (past jobs, satisfaction...)[1]. All these information is kept private and only the intention is disclosed. Once the system has collected participants' intentions, it is able to take an informed decision about whom will treat which query. Such decision may please some users and displease others. If we assume participants

[1] In some architectures, the intention computation task is delegated to an dedicated component to avoid an excessive network load. In such cases, the intention computation is simplified.

to be able to understand they cannot be satisfied every time, a singular episode of dissatisfaction bears no bad consequences. What is important is to avoid dissatisfaction to repeatedly occur on the long term.

A contribution of SQLB is to propose a model to define several fundamental notions, in particular, concerning adequation and satisfaction.

Adequation analyses how well a system fits participant expectations, which could motivate a decision of a participant to stay in the system. This can be measured considering how a participant respond to what the system proposes. In addition, *system adequation to the participant*, is measured considering the average of intentions expressed by the participant with respect to what the system proposes. If a participant exhibits positive intention to all system solicitation, it means clearly that the system is adequate to the participant. Symmetrically, it is possible to measure the *participant adequation to the system* by considering the intentions of other participants to be involved in activities proposed by this one.

Satisfaction analyzes how the system take into account the intention of the user. If the user is (respectively is not) involved by the system in an activity for which she has expressed a positive (respectively a negative) intention, she should be satisfied. Participant *satisfaction with the system* measure is obtained on the long run considering more than one satisfaction decision. Going further, it is possible to understand which is the role of the system strategy in participant's satisfaction. For a strategy to achieve a participant be involved only in activities for which she has expressed positive intention it can be more or less difficult. If this participant always expresses positive intentions it is quite obvious to satisfy her. Conversely, if this participant is very selective and expresses positive intention only exceptionally, the system strategy hardly will satisfy her. From the participant point of view, *satisfaction within the system strategy* measures the system effort to satisfy her considering her *satisfaction* and *system adequation*. From the system point of view, the *strategy efficiency* measures the same notion but also takes *participant adequation to the system* into account [2].

The satisfaction approach subsumes many strategies. For example, if all participants' interests focused on performance, the Satisfaction based Query Allocation acts as a Capacity Based Query Allocation, but conversely to this strategy, it adapts to any change on participant intentions. This allows providers to trade their preferences for their utilization while keeping their strategic information private. It accords to consumers the flexibility to trade their preferences for providers reputation. It makes possible to trade consumers' intentions for providers' intentions. It strives to balance queries at runtime via the participants satisfaction, thus reducing starvation. The satisfaction approach ensures a good level of satisfaction, as far as the system is adequate to participants and conversely.

It is also important to note, that even if trivially adapted to centralized architectures where a mediator regulates the system and can use the satisfaction approach to manage the query allocation process, it can also be adapted to fit distributed architectures [11].

[2] A participant may not be aware about others' intentions which makes the computation of *strategy efficiency* impossible for her.

This satisfaction approach has also been studied to address the problem of query replication in the context of open distributed systems [12]. In such a context, query replication is a preventive way to avoid long reply delay in case of provider failure. Getting answers to the same query from different providers is also a way to detect byzantine failures. In this case the winning strategy is to replicate the query as much as it is needed to overcome failure events, but as few as possible, to avoid system overload. The satisfaction-based approach allows to adapt the replication dynamically per each query, according to the capacity of the system and the interests of the participants.

2.3 BitTorrent overview

In BitTorrent, one of the most efficient protocols for file sharing and content distribution, all nodes are equal, thus they are called peers, and each one can communicate directly with its neighbours. In its mostly used implementation, the index of the ressources each peer holds is centralized in a node called *tracker* which facilitates peers' organisation. The downloading process uses a mesh-shaped overlay, where at any time peers can download/upload simultaneously with multiple neighbors. In BitTorrent each content being distributed determines an overlay called *torrent*. Peers can be either *leechers* or *seeders*. Seeders have a complete copy of the file and leechers are still downloading the file. Churn does not affects too badly peers' download, as long as enough uploaders are in the overlay.

Files in BitTorrent are split in equal-sized pieces, typically 256 KB, and each piece is split in equal-sized blocks (sometimes called chunks), typically of 16 KB. The block is the transmission unit of the content in the BitTorrent overlay network and one piece can be provided to a peer by several uploading peers. Typically, when a peer wants to download a content, it obtains from a web server a *.torrent* file. This file contains meta-information about the content (name, length, size, pieces hashings, etc.) and the url of the tracker. The peer contacts the tracker and requests a list of peers participating in the torrent. Usually the tracker sends to the new peer a list of 50 peers, selected randomly from the torrent. Then the new peer contacts about 20-40 peers to add them to its local neighborhood. Finally the peer begins to exchange pieces with its neighbors.

The download/upload of pieces is defined by means of selection strategies for peers and pieces. The peer selection strategy for uploading, better known as *choking algorithm*, aims at improving the downloading experience of peers that contribute more to the file exchange and at penalizing *free riders*. The main goal of the piece selection strategy is instead to maximize the content diffusion among the participants, by prioritizing the distribution of rarest pieces and maintaining a good average throughput among peers.

The literature about BitTorrent is huge. The reader interested in more details can find some useful insights in [13,14,15,16], among many other publications.

In this work we are not interested in modifying the BitTorrent protocol and we do not pretend to improve its performances. Our goal is to provide a mechanism to let users express their personal preferences and give them a feedback about how these preferences are taken into account in the P2P content sharing BitTorrent (or any other application) implementation.

3 WUW (What Users Want)

The objective of this work is to take into account users' preferences during their participation in the P2P system. Those preferences are used by *strategies* to compute intentions towards other users according to the satisfaction-based approach (see Section 2.2). WUW, the service we propose, provides users some control to influence their neighborhoods based on their personal strategies. It is located on top of the P2P layer, so it is generic enough to be adapted to different unstructured P2P protocols. The main objectives of WUW are: (i) to evaluate the job of the P2P application in use with respect to the preferences expressed by the local user; (ii) to rank the peer list associated to a content and give to the P2P layer a subset of this list consisting in the peers whose users most satisfy the expressed preferences.

3.1 Context constraints and main functionalities

The WUW service has been conceived to be independent from the P2P application that is used to trade the content. While these applications focus on performance issues, thus taking into account *peers*' settings and QoS-related parameters, our service is instead concerned about providing *users* a way to express personal preferences and impact the way the P2P applications works. Then, by measuring to which extent the preferences of the users are being considered, WUW is also able to give users a feedback about the overall quality of the P2P application job, from the point of view of their personal preferences, regardless any technical and architectural parametrization.

This first objective of WUW is achieved by borrowing notions from the SQLB framework and adapting them to this peculiar environment. As in the original framework, preferences are users' personal choices and the way they are used to evaluate other peers (the *strategy*) can be personalized by the local user at will. Using this local information, WUW computes the *intentions* of the local user at each peer, and use the values of these intentions to evaluate the neighbors of the local peer, in order to differentiate them according to the preferences each user expresses. In the following, we recall some definitions belonging to SQLB and used by WUW with no modification.

- A user preference $p \in P$ is a couple $p =< label, value >$.
- A strategy $s \in S$ is a function that maps a set of preferences into real numbers: $s : P \rightarrow \mathbb{R}$. These numbers are called intentions, as they quantify in a compact way the attitude of a user towards other users.

SQLB was designed to be implemented as a centralized mechanism. A mediator, a logically single component, collects intentions from providers and consumers and find the best match between them, trying to maximize their satisfaction. In this work, we focus on decentralized P2P networks in which no central component is given, or, even if it is, it cannot reasonably bear the burden of a combinatorial computation that should be repeatedly performed considering each peer with respect to all the others for every shared content.

It is thus essential that each user builds her own ranking locally. WUW computes intentions related to every neighbor for every content the local user is trading via the

P2P application being used. The neighbors are then scored according to the intentions associated to each of them and this turns the usually flat lists of peers that compose the P2P overlays for any shared content into rankings, where peers are not generically equal to each other, but different, reflecting a personal user evaluation and a given strategy.

An important difference between SQLB and WUW is related to the very nature of the actors. While SQLB's main actors are producers and consumers, typical connotations of a client-server system design, the WUW service operates with peers. In its very nature, every peer is simultaneously a client and a server, with respect to the other peers sharing a given content. Thus, when considering users intentions, it is due to take into account both faces of the peer nature.

For this very reason, intentions, user scores and peer ranking are computed twice:

- a first time considering the local peer as a client and the remote peers as servers and
- a second time considering the local peer as a server and the remote peers as clients.

The two possibly different rankings are then merged (and for each neighbor the average of the two scores is computed) to obtain the definitive one.

3.2 WUW features and dynamics

In this section we give further details about the decentralized computation and information exchange performed by WUW. The kind of information shared by users and the way this information is exchanged and used on each peer is described in the following.

Gossip-based information dissemination To make meaningful choices by computing scores in a fully decentralized fashion, the users must be informed on what the other users think and how they behave. WUW disseminates the intentions of the users via an epidemic protocol. Epidemic protocols are known to be efficient and robust ways to spread information in decentralized networks [17]. Each instance of WUW, running on each peer, participates in the dissemination of information coming from all the other peers, according to the epidemic paradigm. Every user is thus able to know the intentions of her neighbors towards her. By combining her own intentions and the neighbors' intentions in the ranking phase we explained above, the resulting ranks turn out to be an informed choice. The user is free to decide to balance anyone's intentions, or to be more "altruistic" and prioritize others', or to be "egoist" and consider her intentions only. More details on how information is spread among WUW instances are given in Section 4.3.

It is important to notice that the amount and the kind of information to be shared among users are entirely a users' choice. The more information is shared, the more informed will be the choice of each user, whenever the neighbor ranking is locally computed. Users may choose to share not only intentions, but also specific preferences. The only constraint is that the strategy applied by each user must be able to make use of the available information, which could not always be the case, given that users can locally apply strategies that differ from one another. The users are able to express their preferences via a proper interface. Details are given in Section 4.2.

Of course users may be concerned about the fact that revealing too much information may be detrimental for their privacy. Regarding this, it is worth underlining that, to make non-trivial and informed choices, sharing intentions only is enough. Intentions are real numbers that quantify how much a remote user is happy to share a given content with the local user at the present time. They reveal nothing about the reasons behind the number, which are most probably the sensitive information that may arise privacy concerns.

Feedback and self-evaluation Once the local ranking has been computed on each peer, the P2P application in use is feeded with only a subset of best ranked peers, rather than with the original peer list, so that the user preferences will directly impact the way the P2P application overlay evolves during the content sharing task.

Of course, pre-selecting those peers that the P2P application will use could have an undesired impact on performances. If this is the case, the P2P application will certainly report that the QoS is decreasing. What is important in our service, though, is to understand if the choices of the users are actually effective in making her sharing content with the users she like the most or not. Therefore WUW provides a feedback to the user, quantified in the *satisfaction*, *adequation* and *system evaluation* measures. These notions are inspired by the SQLB framework as well, although their formal definition in WUW is substantially different, reflecting the diverse context and actors.

The feedback measures are periodically recomputed by each peer and are related to the preferences of the local user and to the job the WUW service is doing for her. The main goal of a clever service design is thus maximizing the values of these feedback measures while minimizing any negative impact on the P2P application's QoS.

To be able to compute the feedback measures, WUW gets some information from the local instance of the P2P application about the status of the content sharing tasks. Intentions are computed by evaluating remote users with respect to given contents. Thus if a remote user shares several contents with the local user, she will be assigned distinct intentions for each content. Anyway, waiting for a whole content to be shared to be able to produce some feedback makes the aggregate feedback measures themselves almost irrelevant for at least two reasons: first, because they will be available to the user after a quite long wait, thus making it impossible to understand the dynamics occurred while the sharing was ongoing. Second, because they will have been measured only once, taking into account all the exchanges occurred with all the neighbors, thus giving a too coarse-grained evaluation of each of them.

Thus we assume that any content can be logically split in a set of non overlapping *items*. The items are the units of measure used to compute and update the feedback measures. WUW lets the precise definition of item depend on the P2P application being used. Details about the interaction between WUW and the P2P application in use are given in Section 4.4. Considering then a content C as a set of items $i_1, ..., i_n$ and with the help of the notation defined in Table 1, we give the following definitions of feedback measures related to this content.

The *Satisfaction* S_c of a (local) user as a client (that means as a "downloader") is computed as follows. For each item $i \in C$ whose download has completed, let $S_c[i]$ be the sum of the local user's intentions towards the users who has provided these items,

Notation	Meaning
u	A remote user
P_i	The set of users who provided the item i *
Ic_C^u	The local user's intention "as a client" toward the remote user u *
Is_C^u	The local user's intention "as a server" 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 *

Table 1: Notation used to describe feedback measures computation on each peer. All the events are intended to be to or from the local user and computed based on the locally available information.
The symbol "*" implies the constraint: "since the last time the measure was computed".

multiplied for the number of successful download events related to each item, divided by the number of times each item, or part of it, has been requested. That is :

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

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

$$S_c = movAvg\left(\frac{\sum_{i \in D} S_c[i]}{\parallel D \parallel}\right) \tag{2}$$

Intuitively, S_c measures to which extent the P2P application prefers "good users" over the others, to get a given content. Its value can vary between 0 and 1, with 1 denoting the best possible choices are always made.

The Satisfaction S_s of a (local) user as a server (that means as an "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 the users who have downloaded these items, multiplied for the number of successful upload events related to each item. That is :

$$S_s[i] = \sum_{u \in RD_i} (((Is_C^u + 1)/2) \cdot \parallel RD_i^u \parallel) \tag{3}$$

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

$$S_s = movAvg\left(\frac{\sum_{i \in RD} S_s[i]}{\parallel RD \parallel}\right) \tag{4}$$

Intuitively, S_s measures to which extent the P2P application prefers "good users" over the others, to distribute content. Its value can vary between 0 and 1, with 1 denoting the best possible choices are always made.

The *Adequation* A_c of a (local) user as a client (that means as a "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 these items. That is :

$$A_c[i] = \frac{\sum_{u \in H_i}((Ic_C^u + 1)/2)}{\parallel H_i \parallel} \tag{5}$$

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

$$A_c = movAvg\left(\frac{\sum_{i \in LQ} A_c[i]}{\parallel LQ \parallel}\right) \tag{6}$$

Intuitively, A_c measures to which extent the preferences of the local user and her strategy are assigning higher scores to useful peers over the others, 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 (that means as an "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 the users who have requested these items, multiplied for the number of upload events related to each item. That is :

$$S_s[i] = \sum_{u \in RQ_i} (((Is_C^u + 1)/2) \cdot \parallel RQ_i^u \parallel) \tag{7}$$

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

$$A_s = movAvg\left(\frac{\sum_{i \in RQ} A_s[i]}{\parallel RQ \parallel}\right) \tag{8}$$

Intuitively, A_s measures to which extent the P2P application strive to distribute those contents that are more requested by the "good users". Its value can vary between 0 and 1, with 1 denoting the best possible choices are always made.

At any given time, the *System Evaluation* of a (local) user as a client (respectively: server) is the ratio S_c/A_c (respectively: S_s/A_s). Intuitively, the System Evaluation measures how much the local user can be happy about the impact of her preferences and the applied strategy on the ongoing sharing task, both as a client and as a server. Its value can vary in the interval $[0...\infty]$ (the highest, the better), with 1 denoting a neutral impact.

User scores and peer ranking The scores assigned to each remote user and the ranking of the neighbors are computed at each peer as weighted averages.

For each remote user u and for each content traded by the local user l, let us call $I_c(u)$ the intention of u as a client towards l as a server and $I_s(u)$ is the intention of

u as a server towards l as a client. Conversely, let $I_c(l)$ be the intention of l as a client towards u as a server and $I_s(l)$ the intention of l as a server towards u as a client. The score assigned to u is then defined as

$$s(u) = (S * I_c(l) + (1 - S) * I_s(u) + S * I_s(l) + (1 - S) * I_c(u))/2 \quad (9)$$

The weight S is a real parameter in [0..1] decided at initialization time. It basically express the "selfishness" of the local user in considering her own intention as more important than the remote users' in trading the given content.

Based on the scores assigned to each remote user, a general ranking is build on each peer. The positions of the remote peers in the general ranking is computed by averaging all the scores assigned to u for all the contents and sorting the list of users according to these average values. Then at most K peers, picked from the general ranking in order, starting from the best ranked, are assigned to the peerlists associated to each content they are trading. These peerlists are finally given to the P2P application as local neighborhoods.

4 WUW design and implementation

In this section we detail the architectural design of our service. WUW functionalities are separated in different modules. Each module is able to get input from and give output to the other modules through well-defined interfaces. Thus changing the implementation of a single module does not require any change in the other modules, as long as the the proper interfaces are correctly implemented. Being WUW a multi-thread application, any information exchange among the modules is assumed to be asynchronous and possibly concurrent. WUW is implemented in Java and requires JVM 1.7 or higher. The sources are under the GPL license.

This section presents an overview of the WUW architecture in Section 4.1, and details the architectural modules in Sections 4.2 to 4.5.

4.1 WUW architecture

The WUW architecture is shown in Figure 1. Conceptually WUW acts as a man in the middle between the local instance of the P2P application and the overlay management system of the P2P network. This means that the P2P application communicates with WUW to know about other peers in the overlay and WUW communicates with an overlay coordinator (e.g., a tracker), a DHT, or other, depending on the P2P system being used) to get information about the state of the overlay.

Thus the typical sequence of actions of our service in an unstructured P2P content sharing system is the following:

- retrieve information about the content to be shared and the peer list to share it with;
- get the user preferences about the given content;
- periodically exchange messages with the remote users to share intentions and information about the state of the task;

- periodically retrieve data from the local P2P application instance about the state of the task;
- periodically compute/update the feedback measures by consuming, at each iteration, newly available data coming from the local P2P application and the remote WUW instances;
- update local user's intentions towards the neighbors for each content, by applying the given strategy to the current user's preferences;
- score the neighbors according to the intentions associated to each of them, then build a global ranking with all neighbors;
- for each content, build a peer list composed at most by the K best ranked peers who are sharing the content;
- send the newly created peer lists to the P2P application, and make the freshly updated feedback available to the user.

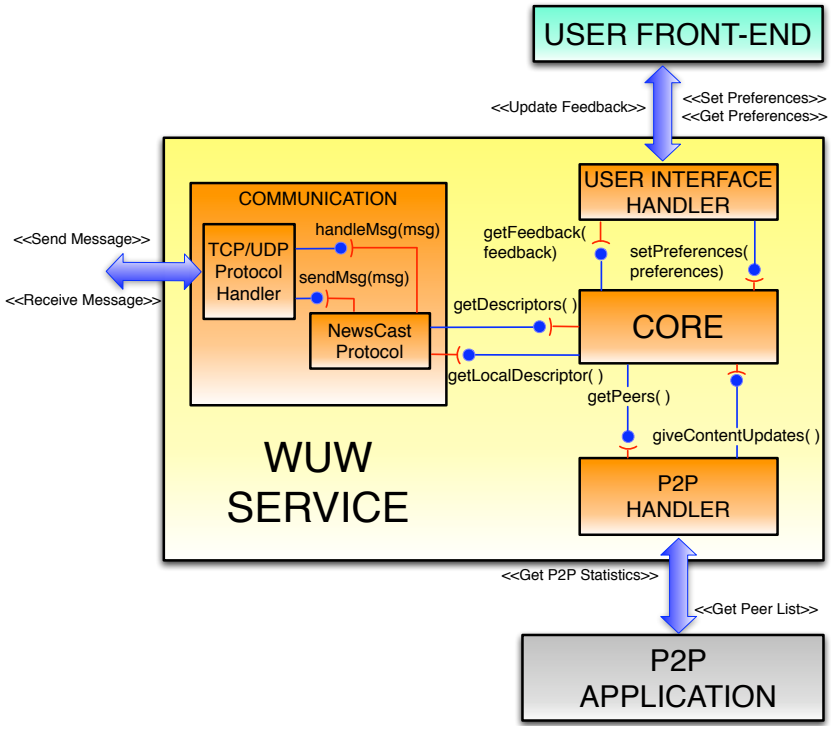


Fig. 1: WUW architecture.

4.2 The User Interface Handler module

The primary source of information for WUW is of course the local user. The User Interface Handler (UI Handler) module takes care of getting input from the user and outputs

useful feedback. The current implementation features a simple web-based interface that let the user add and change contents and preferences and reports the feedback computed along the computation. The total independence of the module from the rest of WUW makes it possible to implement different user interfaces at will and easy to improve the existing one.

4.3 The Communication module

The Communication module implements basic functions to facilitate communication among WUW instances over TCP or UDP.

Moreover, the NEWSCAST [18] epidemic protocol is provided to disseminate information in the P2P overlay. The information to be spread is periodically updated by each peer and G-zipped in a data structure called *PeerDescriptor*. All the peers contribute to the dissemination of descriptors of any other peer, but each of them uses and keeps in its working memory only the information that concerns the local user. This is done both to reduce the memory footprint of the service and to only disclose information to the users whom it is meant to be addressed to. A peer can only update the descriptor that concerns the local user. No change to remote users' descriptors is allowed.

The epidemic dissemination of up-to-date information makes it possible to each local user to know what are the recently computed intentions of the remote users towards her and what is the state of the diffusion of a given content. While it is common for P2P applications to be able to provide this latter information, WUW does not rely on a particular P2P application and it is able to retrieve this data autonomously.

Of course the epidemic diffusion of useful data is not instantaneous. A certain delay between the local generation of an updated descriptor and its fruition by remote peers is unavoidable and may have consequences that must be considered and discussed while drawing experiments with WUW.

4.4 The P2P Handler module

This module is the only part of WUW that is aware of the specific P2P application being used by the local user. This means that all that concerns the way WUW interacts with it and the specific way information can be retrieved and given to it is secluded in this module.

The data WUW need to retrieve from the P2P application are essentially the basic information we can reasonably expect to be possible to retrieve from any application. For each content being shared and for each neighbor in the overlay, WUW needs to know of any distinct download of upload event from/to to a given neighbor related to a given item. If these events can be time-stamped, they can be also used to estimate performances and build a "local" opinion about the remote users, beyond what concerns the local user's preferences and intentions. The basic data that WUW needs to send to the P2P application concerns the periodically renewed peer list to be associated to a given content.

Thus the main tasks of the P2P Handler module are:

– to retrieve from the P2P application overlay management system the list of peers associated to a given content;

- to give to the P2P application the subset of peers that are optimal according to local user's preferences and strategy;
- to periodically retrieved from the P2P application the needed data and process them to produce data structures that are usable by the rest of the service to perform the proper computations.

The current implementation of WUW features a P2P Handler module that is able to send and retrieve data from BitTorrent. Besides implementing what is needed in this module, we produced an instrumentation of the MainLine 3.1.9 version of BitTorrent, such that the required information can be periodically collected and retrieved via a local socket from the working memory of the BitTorrent process, thus minimizing the performance impact due to the interaction overhead.

4.5 The Core module

The main functionalities of WUW are placed in this module. This module is the orchestrator of the service. The intentions and the feedback are computed in a timely way. Every n seconds, a routine is started which performs the following steps:

- Get updates about the remote users from the Communication module.
- Get the latest information about the activity of the local P2P application from the P2P Handler Module.
- Get the latest changes in the users preferences from the UI Handler module.
- Update the local state with the collected information.
- Compute the feedback related to the latest local activity, considering the current intentions for all the neighbors and the contents.
- Make the newly computed feedback available to the local user via the UI Handler module.
- Compute the new intention values to be associated to every neighbor for every content, according to the applied strategy.
- Build a global ranking of all the neighbors, considering the associated intentions for all the contents.
- For each content, add neighbors sharing that content to its peer list, starting from the best ranked on, until at most K peers are added to each list.
- Send the newly created peer lists to the P2P application, to be associated with the respective contents, via the P2P Handler module.

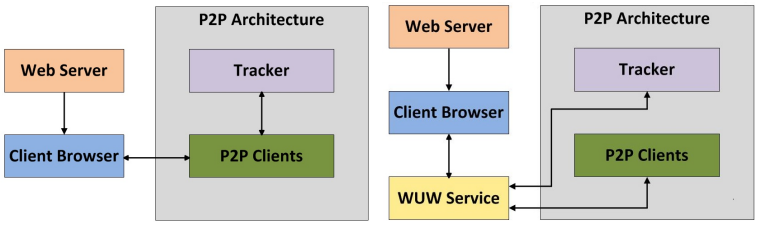
By repeating this routine regularly, the user is kept updated about the impact that her choices have on the local computation. While any QoS-related performance issue is usually reported by the P2P application in use, the feedback measures provided by WUW allow users to evaluate the overall "quality of their neighbors" (measured according to the local users preferences) and the average quality of the WUW service itself.

5 WUW as an added value in a CDN

In this section we discuss a proof of concept use case based on a CDN. We consider that a CDN consists in a centralized content provider that is able to distribute simultaneously several contents to a large number of clients. By hybridizing a CDN with a P2P distribution mechanism, the servers off-load part of the network traffic to the participants, who become active actors in the distribution task. The obvious goals are to achieve higher scalability and to reduce structural costs for the content provider.

Such a hybridized CDN could exploit the services of WUW at best. In a CDN there is a central authority that guarantees the quality and the safety of the content. The same central entity can also provide interesting metadata to help users making informed choices. By requiring a secure and unique identification of the user, the authority can provide useful characterization about the participants, with no need to disclose sensitive data. All this information could be taken into account by users, who may have some requirements whenever they are willing to download and share contents. This is precisely the purpose for which WUW has been conceived.

The global architecture of a P2P-based CDN application is shown in Figure 2 (a). When WUW is included, it is located between the browser and the P2P layer as seen in Figure 2 (b). The browser provides a user interface to access the P2P-based CDN server. The P2P application layer is BitTorrent-based and is composed of a tracker and P2P clients. The tracker maintains the set of peers participating in each torrent. A P2P client is a component implementing the P2P communication protocol and we consider there is only one client by user. The web server contains all the .torrent files of the content provided by the CDN. To simplify, we consider one web server and one tracker but several can exist to improve performances and they can be organized, by instance, trough a DHT (Distributed Hash Table)..



(a) Without WUW (b) With WUW

Fig. 2: Global view of the P2P-based CDN application.

Figure 3 shows, in a very general way, the content download process. Roughly speaking, users willing to download content introduce their preferences trough a web interface (User Front-End). Preferences can be by session, by type of content or by content. This granularity decision depends on the personal strategy of the user. Preferences are sent to WUW and not to the web server to respect users' privacy. Then, through the

web interface, users request content to the web server which sends the .torrent file to the P2P client. This file contains the address of the tracker. The P2P client request of the peer list is intercepted by WUW which communicates directly with the tracker. WUW obtains the peer list from the tracker and ranks the peers. To rank peers, WUW needs to know some information about those potentially neighbor peers, so it contacts other WUW components who are free to send or not the required information. Then, WUW selects the peers that fits the best its preferences and send this modified list to the P2P client. Then the download process is made like usual depending on the P2P protocol (this is maked in red in the figure).

Preferences may include class of content (e.g., movies, songs, tv programs), the type of content (e.g., for movies it can be action, comedy, political, children, etc.), minimal level of reputation of neighbor peers, location (e.g., particular city, region, continent), upload bandwidth, number of connections, etc.

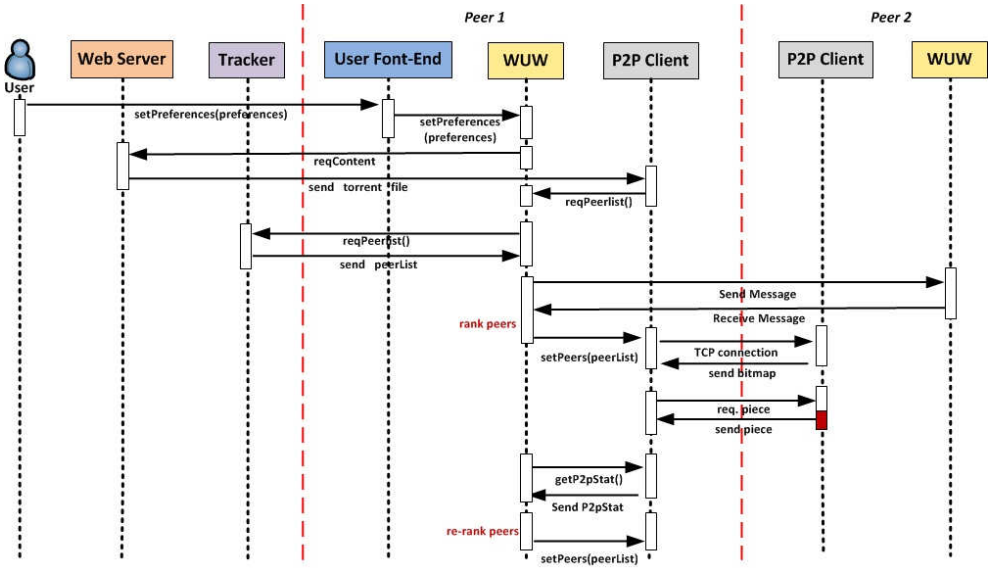


Fig. 3: Sequence diagram example to download content.

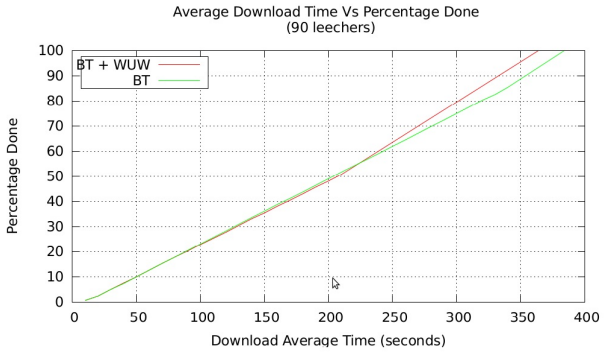


Fig. 4: Performance of WUW in terms of download average per second.

6 Experimental results

We are currently running and analyzing WUW experiments using Grid5000 as testbed. In the next, we show results only for two of those experiments. The first one measures the content distribution performance of WUW and the second one presents the peers' rankings made by WUW for 3 different types of preferences.

The parameters of the experiments are the following:

- A torrent of 100 nodes.
- 256 Kb of bandwidth. This bandwidth allows more or less 15 to 20 connections by peer.
- 10% of nodes are seeders and 90% are leechers. Those nodes have the content before launching the experiment.
- Content size of 112 MB.
- BitTorrent instances (leechers) ask for more peers every 40 seconds, the tracker answer with a list of 10 peers.
- The NEWSCAST algorithm sends the descriptor to one randomly chosen peer every 2 seconds.
- WUW computes every 8 seconds the feedback measures and the ranking of peers.
- Concerning users' preferences for simplification reasons we chose only the interest of peers in other peers. Each peer has a kind of tag with the type of peer it is: *normal peer*, *touchy peer* or *exigent peer*. We consider a system composed of 50% of normal peers, 30% of touchy peers and 20% of exigent peers.

The objective of the first experiment is to measure performance of WUW in terms of download average per second. This experiment consists of two steps, in the first step each node runs a BitTorrent (BT) instance and in the second step, each node runs a BitTorrent and a WUW instance (BT+WUW). For the first step, we use the BitTorrent implementation Mainline 3.1.9 without any modification and there is one tracker. In the second step we include in BitTorrent a class to collect some statistics needed by WUW

and there is no central tracker. By its design, WUW intercepts communication between peers and the tracker. In the experiment, WUW knows all the peers in the system and there is no churn, so the tracker is useless. Instead, we implement a tracker emulator that meanly choses randomly the list of peers sent to each leecher. WUW intercepts and modifies this list by including the peers more interesting and excluding the less interesting ones from the users' preference point of view .

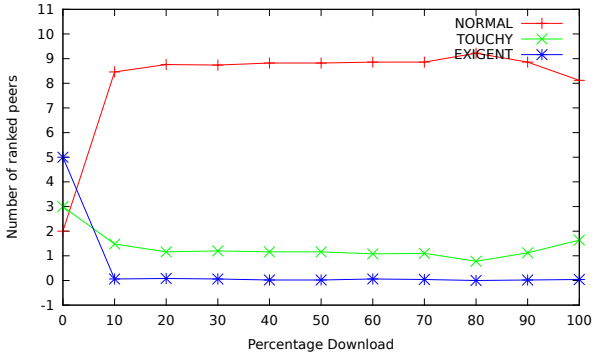


Fig. 5: Average ranking of *normal peers*.

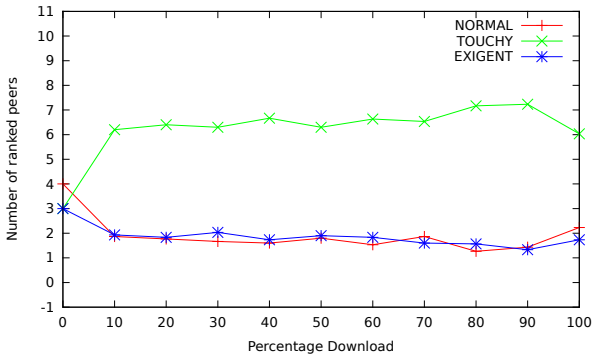


Fig. 6: Average ranking of *touchy peers*.

Figure 4 shows that in average a leecher got the hole content in 384.21 seconds when using BitTorrent and in 364.02 seconds when using WUW with BitTorrent. Results show an improvement of 5.1% (20 seconds) for WUW+BT, this improvement is due to the trackerless approach used.

Figures 5, 6 and 7 present in average the ranking of the top 10 peers made by WUW by percentage of download. Each figure presents the group of peers of same type. The

strategy used to compute intentions as server and as client, roughly speaking, gives better score to peers whose type is close in descending then in ascending order, i.e., a normal peer prefers peers like it then touchy peers then exigent peers; conversely, an exigent peer prefers touchy peers then normal peers, after peers of its same type.

The results are obviously influenced by the number of peers having same type in the system. For normal peers (i.e., Figure 5) is easy to find peers like them (50% of peers are of that type). Touchy peers (i.e., Figure 6), after ranking all possible peers of same type, prefer exigent peers, but they are only 20% of the system so they are not always available. Exigent peers (i.e., Figure 7), after ranking peers of same type, prefer touchy peers.

We can notice that at the beginning of the download process, the ranking is not meaningful. That is because the ranking is based on the information disseminated by the epidemic protocol and, in average, after 10% of the download WUW has enough information (i.e., knows enough remote peers' intentions) to apply its strategy and rank peers.

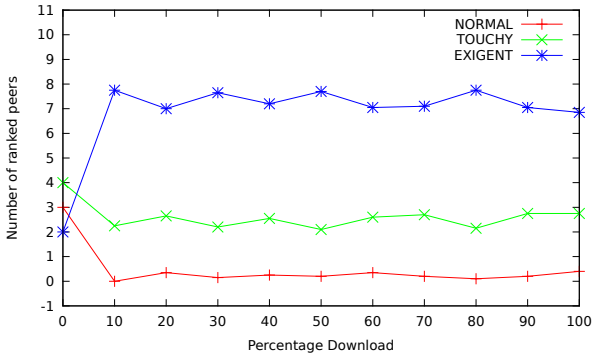


Fig. 7: Average ranking of *exigent peers*.

7 Conclusion and future work

This paper presented WUW (What Users Want), a P2P service that allows to enhance users' satisfaction based on personal strategies. This work was inspired by the SQLB approach [2] where a mediator (a centralized entity) collects preferences from providers (servers) and consumers (clients) and finds the best match between them. In the P2P context, every peer is simultaneously a client and a server and we do not consider a centralized entity. The main objectives of WUW are: (i) to evaluate the job of the P2P application in use with respect to the preferences expressed by the local user; (ii) to rank the peer list associated to a content and give to the P2P layer a subset of this list consisting in the peers whose users most satisfy the expressed preferences. In addition, WUW provides feedback to the user about satisfaction, adequation and system evaluation. These notions are inspired by the SQLB framework as well, although their formal

definition in WUW is substantially different, reflecting the diverse context and actors. We presented first experimental results of a WUW implementation where the P2P layer used is BitTorrent. Those experiments showed that without loosing performance WUW improves users' satisfaction.

References

1. Buyya, R., Pathan, M., Vakali, A.: Content Delivery Networks. Lecture Notes in Electrical Engineering. Springer (2008)
2. Quiané-Ruiz, J.A., Lamarre, P., Valduriez, P.: A Self-Adaptable Query Allocation Framework for Distributed Information Systems. Very Large Databases Journal (VLDB) **18**(3) (June 2009) 649–674
3. Liao, X., Jin, H., Liu, Y., Ni, L.M., Deng, D.: AnySee: Peer-to-Peer Live Streaming. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, Barcelona, Catalunya, Spain (April 2006) 1 –10
4. 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: Latin American Networking Conference (LANC), Pelotas, Brazil, ACM (September 2009) 49–59
5. Ruixuan, L., Guoqiang, G., Weijun, X., Zhiyong, X.: Measurement Study on PPLive Based on Channel Popularity. In: Communication Networks and Services Research Conference (CNSR), Ottawa, Ontario, Canada (May 2011) 18–25
6. Wei, L., Jingping, B., Rong, W., Zhenyu, L., Chen, L.: On Characterizing PPStream: Measurement and Analysis of P2P IPTV under Large-Scale Broadcasting. In: Global Telecommunications Conference (GLOBECOM), Honolulu, Hawaii, USA, IEEE (December 2009) 3552–3557
7. SopCast. `http://www.sopcast.org/` (July 2012)
8. Swarmplayer. `http://swarmplayer.p2p-next.org/` (July 2012)
9. uTorrent. `http://www.utorrent.com` (July 2012)
10. `http://web.media.mit.edu/~vyzo/vidtorrent` (July 2012)
11. Quiané-Ruiz, J.A., Lamarre, P., Cazalens, S., Valduriez, P.: Scaling Up Query Allocation in the Presence of Autonomous Participants. In: Conference on Database Systems for Advanced Applications: Part II (DASFAA). Lecture Notes in Computer Science, Hong Kong, China, Springer (April 2011) 210–224
12. 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 (DPD) **30**(1) (2012) 1–26
13. Legout, A., Urvoy-Keller, G., Michiardi, P.: Rarest First and Choke Algorithms are Enough. In: SIGCOMM Conference on Internet Measurement (IMC), Rio de Janeriro, Brazil, ACM (October 2006) 203–216
14. Xia, R.L., Muppala, J.K.: A Survey of BitTorrent Performance. Communications Surveys and Tutorials (COMSUR) **12**(2) (April 2010) 140–158
15. Wiki: BitTorrent Protocol Specification v1.0. TheoryOrg http://wiki.theory.org/BitTorrentSpecification.
16. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Workshop on Economics of Peer-to-Peer Systems, Berkley, CA, USA (June 2003)
17. Pittel, B.: On Spreading a Rumor. SIAM Journal on Applied Mathematics **47**(1) (February 1987) 213–223
18. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based Peer Sampling. ACM Transactions on Computer Systems (TOCS) **25**(3) (August 2007)