# Peer-to-Peer clustering of Web-browsing users

Patrizio Dazzi
ISTI-CNR
Pisa, Italy
p.dazzi@isti.cnr.it

Pascal Felber
University of Neuchâtel
Neuchâtel, Switzerland
pascal.felber@unine.ch

Le Bao Anh
EPFL
Lausanne, Switzerland
lbanh@ifi.edu.vn

Lorenzo Leonini
University of Neuchâtel
Neuchâtel, Switzerland
lorenzo.leonini@unine.ch

Matteo Mordacchini
ISTI-CNR
Pisa, Italy
m.mordacchini@isti.cnr.it

Raffaele Perego
ISTI-CNR
Pisa, Italy
r.perego@isti.cnr.it

Martin Rajman
EPFL
Lausanne, Switzerland
Martin.Rajman@epfl.ch

Étienne Rivière
NTNU
Trondheim, Norway
etriviere@gmail.com

## ABSTRACT

For most users, Web-based centralized search engines are the access point to distributed resources such as Web pages, items shared in file sharing-systems, etc. Unfortunately, existing search engines compute their results on the basis of structural information only, e.g., the Web graph structure or query-document similarity estimations. Users expectations are rarely considered to enhance the subjective relevance of returned results. However, exploiting such information can help search engines satisfy users by tailoring search results. Interestingly, user interests typically follow the clustering property: users who were interested in the same topics in the past are likely to be interested in these same topics also in the future. It follows that search results considered relevant by a user belonging to a group of homogeneous users will likely also be of interest to other users from the same group. In this paper, we propose the architecture of a novel peer-to-peer system exploiting collaboratively built search mechanisms. The paper discusses the challenges associated with a system based on the interest clustering principle. The objective is to provide a self-organized network of users, grouped according to the interests they share, that can be leveraged to enhance the quality of the experience perceived by users searching the Web.

## Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering, Information filtering*

## General Terms

Algorithms, Measurement, Performance

## Keywords

Peer-to-peer, Interest-based clustering, User profiling

## 1. INTRODUCTION

The access to distributed resources such as Internet pages or shared files usually require the use of a search tool, e.g., a centralized search engine such as Yahoo! or Google. These search engines compute and rank their results on the basis of several different pieces of information taken from the Web graph structure, such as document PageRank [5], and from statistical estimates of query-document similarities like the TF/IDF metric. The profile of users and their tastes are rarely taken into account to enhance the users' search experience, although they provide more accurate results with respect to the interests of that particular user and such a profiling would yield better results in many situations. A first such situation occurs when there exist results along several different domains for a given ambiguous query, e.g., the request for the keyword "jaguar" can give results about cars, animals, operating systems, and several other unrelated subjects. The fact that the user issuing the query usually browses the Internet for new car models would help in determining automatically the *interest domain* to which her query likely belongs. The second and more common situation where user-centric profiling information would be effective is when a search system attempts to offer *suggestions* along with the results of a given query, e.g., when the keywords were not selective enough. Users typically use ambiguous and too general queries, instead of selective ones that would filter out the results. Some tools included in modern search engine (such as the "Searches related to:" tool in Google) propose more targeted searches, but do not take into consideration the users' expectation and search history in the process. Only the frequency of the queries is used. For many users, and given the typically skewed, long-tail distribution of interests observed for Web content, the suggested queries may not give more satisfactory results than the ones that are already proposed on the first results pages of the search engine. This calls for new tools that can take advantage of user-centric and interest-profiling information to enhance the search engines capabilities with interest-awareness for better-tailored search.

An interesting observation is that, among groups of Internet users, interests for data typically follow the clustering property [1,2,14,19]: two users who were interested to same topics in the past are more likely to be interested to the same topics in the future. More, it is likely that elements searched by users interested in one such domain will also interest other users from that group in their future searches, either as additional results, or as suggestions to replace the missing selectiveness of their query by the mean of interest scoping based on collaboratively-built knowledge. Grouping users with similar interests has already been successfully exploited for greatly increasing the chances to locate new data when using unreliable search mechanisms such as flooding or random walks [7,10,14,20,23].

In this paper, we propose the general architecture of a system that pushes further the idea of exploiting collaboratively built search mechanisms, based on interest clustering and obtained through recommendations among users. The envisioned system is meant to be used either as a stand-alone search engine, e.g., for a peer-to-peer file sharing system, or perhaps more interestingly, as a companion for some existing search engine. We will concentrate on the description of the challenges that the design of such a companion system raises and on the discussion of the corresponding technical choices. Our overall objective is to construct a self-organized network of peers, each peer being attached to a single-user, with users (i.e., peers) grouped according to their shared interests. Obviously, each peer can participate and belong to several groups, for each of the main interest domains it has been assigned to. Thereafter, the knowledge at the neighbors from that peer is leveraged (1) to enhance the search recall by proposing the resources that were deemed interesting for the same requests by interest-neighbors, and (2) to deal with ambiguous queries by comparing the results return by some search engine with the interest-communities the querying user belongs to, and by re-ranking results accordingly.

The proposed system is a two layers, peer-to-peer (P2P), fully decentralized system. This choice is due a series of considerations about the system goals. First of all, such systems allow proposing a service without any centralized authority (e.g., a single server that would store all profiles and browsing histories of users) with the service implemented through the collaboration of the peers. It is also potentially more difficult for a node to cheat and bias the results given by the system, as a single node will only have limited impact for suggesting search results to its neighbors. Moreover, it is possible, as we will see, to prevent peers from disturbing the system by faking statistics about sites' popularity, while it is impossible to detect that form of cheating with a centralized server that could modify the order of sites, e.g., based on commercial reasons. Next, P2P systems allow us to solve the important problem of scale, as they do not require the over- and proportional-provisioning of resources that would be required with a centralized approach. The more peers participate, the more power is added to the system. A P2P approach scales well to large numbers of peers and it does not suffer from the *bootstrap problems* [12] (i.e., the difficulty, for a centralized and stand-alone service, to attract enough users to fully sustain its specific functionalities—here, the P2P system can be used in conjunction with an existing search engine). Finally, P2P systems are known to deal gracefully with system dynamism at no or very little additional cost, whereas centralized systems need expensive and complex techniques to ensure continuous operation under node and link failures.

Such a system poses several design and engineering challenges. This is why our envisioned system is based on a two level P2P organized network. First, it is necessary to construct the interest-based network, so that peers are effectively grouped with other peers that share similar interest in their various interest domains. This requires maintaining a representation of these interests (*user profiling*), to compare these profiles to determine their similarity (*similarity metrics*) and finally to propose distributed algorithms that cluster peers in interest-based groups based on this metric (*clustering algorithm*). Moreover, from an orthogonal point of view, the system has to care about security and privacy. Indeed, users would not want to use such a system if it allows others to spy on their browsing activity, or if malicious peers can extract the content of their cache in plain text during proximity evaluation.

The remaining of this paper is as follows. First, Section 2 discusses related work. Next, Section 3 reviews the various issues posed by the system construction and Section 4 elaborates on what should be the adequate system architecture, as well as the role of each of its components. Section 5 presents in more details each component and discusses the different issues that are to be faced by the implementer. Section 6 presents future work and concludes.

## 2. RELATED WORK

Many independent studies have observed the characteristics of accesses to distributed data in various contexts. The most important of these characteristics in the context of this paper are: clustering of the graph that links users based on their shared interests, correlation between past and futures accesses by users or by groups of users that share similar interests, skewness of the distribution of interests per peer, skewness of the distribution of accesses per data element. Skewness usually relates to Zipf-shape distributions, which are a feature of access behaviors amongst large groups of humans [28]. We first review the work related to the detection and use of interest correlation between users in large-scale systems.

The presence of communities amongst user interests and accesses in Web search traces [1,2], peer-to-peer file sharing systems [14] or RSS news feeds subscriptions [19] can be exhibited.

The existence of a correlation of interests amongst a group of distributed users has been leveraged in a variety of contexts and for designing or enhancing various distributed systems. For peer-to-peer file sharing systems that include file search facilities (e.g., Gnutella, eMule, ...), a sound approach to increase recall and precision of the search is to group users based on their past search history or based on their current cache content [10,13,23]. Interestingly, the small-world [18] aspects of the graph of shared interests[1] linking users with similar profiles is observed and can be exploited not only for file sharing systems, but also in re-

[1]Small-world aspects for the shared interests graph are: (i) a high clustering, (ii) a low diameter due to the existence of a small proportion of long links, i.e., links to *exotic* domains that are distant from the common interests of the node and its regular neighbors and that act as cross-interest-domain links, and (iii) the possibility to navigate through the graph of interest proximity amongst peers and effectively find short path between two interest domains based only on the one-to-one distance relationships amongst these domains, i.e., without global knowledge of the graph.

searcher communities or in web access patterns [14]. Another potential use of interest clustering is to form groups of peers that are likely to be interested in the same content in the future, hence forming groups of subscribers in a content-based publish-subscribe [7]. Finally, interest correlation can be used to help bootstrapping and self-organization of dissemination structures such as network-delay-aware trees for RSS dissemination [20]. Finally, user interest correlation can be used for efficiently prefetching data in environments where access delays and resource usage constraints can be competing [26], as it is an effective way of predicting future accesses of the users with good accuracy.

The correlation between the users' past and present accesses has been used for user-centric ranking. In order to improve the personalization of search results, the most probable expectations of users are determined using their search histories stored on a centralized server [24,25]. Nevertheless, the correlation between users with similar search histories is not leveraged to improve the quality of result personalization, hence making the approach sound only for users with sufficiently long search histories.

An alternative class of *clustering search engines* uses semantic information in order to *cluster* results according to the general domain they belong in (and not as in our approach to cluster users based on their interests). This can be seen as a centralized, server-side and user-agnostic approach to the use of characteristics of distributed accesses to improve user experience. The clustering amongst data elements is derived from their vocabulary. It presents the user with results along different interest domains and can help her to disambiguate these results from a query that may cover several domains, e.g., the query word "apple" can relate to both *food/fruits* and *computers* domains. Examples of such systems are EigenCluster [8], Grouper [27], SnakeT [9] or TermRank [11]. Nonetheless, these systems simply modify the presentation of results so that the user decides herself in which domain the interesting results may fall–these results are not in any way automatically tailored to her expectations. They do not also consider the clustering of interest amongst users, but only the clustering in content amongst the data.

Aspects related to the distribution of the popularity of te elements or to the number of interest domains of the users are of particular importance in the peer-to-peer context, where the responsibility for these elements (or for these users) has to be distributed amongst a large set of nodes or servers. To achieve scalability, it is necessary to balance the load evenly amongst nodes. This is usually achieved by letting all peers interested in one element serve that element (e.g., as in the first versions of Gnutella) at the cost of reducing availability of unpopular resources, or in a more structured manner, to map elements to one or several nodes [17]. An example of a system using reorganization of the data responsibility to cope with skewed dynamic load is Mercury [3]. An example of using data replication for load balancing is given by the Beehive [21] system. An example of replication or split of the responsibility for a group of users is the publish/subscribe system SplitStream [6], which is based on the Pastry [22] DHT.

## 3. CHALLENGES

This Section lists the various research challenges that are associated with our system proposal. A clear definition of these challenges helps in defining and justifying the corresponding architecture, described in the next Section.

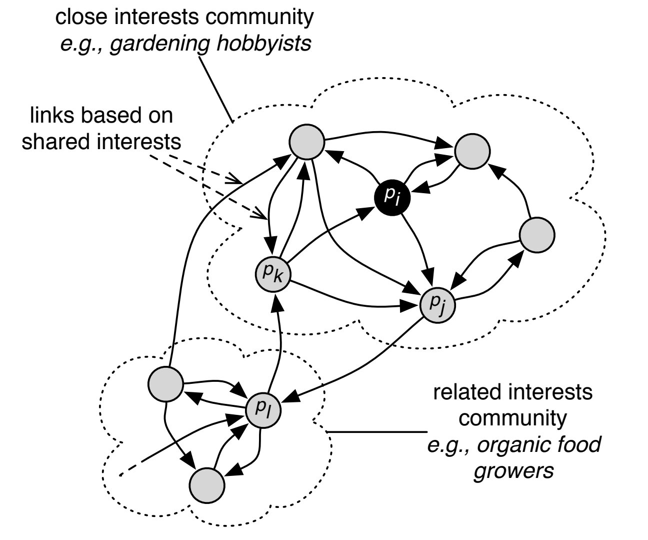
close interests community
e.g., gardening hobbyists
links based on shared interests
$p_i$
$p_k$
$p_j$
$p_l$
related interests community
e.g., organic food growers

**Figure 1: Interest-based network: general principle**

### 3.1 Construction of an interest-based network

We present the general principle of the construction and use of a network of peers based on shared interest, in the context of Web search enhancing mechanisms. Figure 1 presents a coarse view of such a network. This example is purely fictional but helps for presenting the global idea.

Peers (e.g., peer $p_i$ and $p_j$) are linked as they share interests for the same kind of content (or, more correctly, have been interested in the same content in the past and therefore are considered to have a high probability to be again common interests in content in future, i.e. when issuing searches for new content). Users are grouped by the means of a clustering protocol, in two different ways. First, each peer decides independently to which peer it is linked. These one-to-one relationships are chosen based on an interest-based distance, amongst the peer it encounters. Second, based on these one-to-one relationships and on their associated distances, peers are grouped in collectively-known and maintained interest groups (i.e., as collectively recognized communities of interests). An important point here is that a peer is not part of one single group but can participate in as many groups it requires to *cover* its interests. For instance, peer $p_j$ is interested in, i.e. has been accessing resources about, both gardening and to a lesser extent to organic food production. It hence has links to members of these two groups and is "officially" part of one, but may as well be part of a completely different group, say, one grouping researchers that often search the web for new information retrieval papers.

Note that the labels given here are only for the sake of simplifying explanations: there is no automatic labeling, nor is there any ontology, in the system. The process of creating, deleting, merging or splitting interest-based clusters is completely automatic and solely based on statistical properties.

The task of creating such a network requires the following mechanisms: creating profiles of users that represent their interests, finding a way to construct the one-to-one links with peers that are “close” in terms of shared interest, and to that extent, to define a metric of interest proximity.

### 3.1.1 User Profiling

User interests ideally represent the comprehensive set of thematic areas that are (most often) covered by the documents or items the user is accessing or is likely to access. The automated detection of interest domains is not easy, indeed, as typically user interests are dynamic and time-dependent. Typically, users can be interested in a topic only for a certain period due to either some personal reasons (e.g. who recently lost her/his job looks for a new position) or environmental ones (e.g. who lives or will visit Italy looks at Italian weather forecasting sites). Moreover, due to the fact users have different interests, users can, in their daily conducted browsing activity, access to web resources that are very different in content and heterogeneous in type. As a consequence, it makes the user behavior analysis for interest detection even more complex.

For the implementer, user profiles also have to respect two important properties: they have to be small and lightweight to allow a fast transmission and computation, and they have to hide as much as possible the plain content that is represented while allowing the comparison of what they represent (as part of the similarity metric computation). This calls for the use of space-constrained representation. A typical example is to use Bloom filters [4] that map a large set of elements[2] to a fixed size bit vector. Each element is hashed using $k$ hash functions, setting the corresponding bits. Inclusion tests are made by testing for these same $k$ bits, which require to know the resource name beforehand (hence adding intrinsic privacy support to the structure: it is impossible to reverse the process and obtain a plain text list of the visited web pages, for instance). Interestingly, while inclusion tests can yield false positives, comparing the size of two sets encoded with bloom filters (or, the size of their union/intersection, or their Jaccard similarity) gives good approximations. Counting filters and compact approximators are two possible alternatives that gives better precision (at the cost of a larger space usage). Time issues also have to be taken into account for the profiles: how much time, or how many elements, are to be kept in one profile, are particularly sensitive settings.

### 3.1.2 Similarity Metrics

The measure of similarity between two users, represented by some interest profiles, will eventually be used to form clusters of users, grouped together based on affinities. In this process, what matters is to be able to distinguish between two potential neighbors, which one is *closest* in terms of interest. The presence of interest, first, is denoted by accesses by both peers to the same elements (e.g. two users frequently visit a gardening-related webpage after looking up for information on their favorite search engine, or access Web pages that are described by similar keywords). Simply using the number of common elements has been successfully used in the context of P2P file sharing systems [23] or Web cache design [13]. Nevertheless, this poses the problem of the skewness in the number of elements represented by the profiles (which is due to the skewness in the number of accesses by each peer [1]), unless the profiles are kept to a fixed (or maximum) number of represented elements.

Moreover, it is important to note that a common interest for non-popular resources, or to several of them, represents future shared interests with higher accuracy. Therefore, a good metric has to take into account the popularity of each element that is encoded in each profile to weight the calculation. Nevertheless, this information is not available only at the couple of peers that are computing their interest-distance. The information about their local accesses would bias unpopular elements that are by indeed popular amongst these two peers, consider them as popular and reduce their weight, hence loosing the benefits of using a popularity-aware similarity metric. This requires some *global knowledge* about accesses, i.e., statistics about each page usage based on all accesses from all peers (or from an unbiased subset of these accesses).

## 3.2 Membership, Trust and Privacy

Other important challenges that are faced in constructing the envisioned system lie in the three closely related aspects of membership, trust and user privacy. All require carefully algorithmic designs that take them into account from the beginning.

Membership relates to the following problem: it is necessary to restrict peers from sending arbitrary data to the network to bias the view of other peers, e.g. by participating in multiple interests domains which they would not normally be part of. Moreover, each peer (user) has to be given limited (but fairly distributed) *credits* to participate in the creation of the global statistics.

Trust is linked to membership and is twofold: (1) for the collection of global data coming from interest based communities (2) for peers and users, the confidence they have in using these statistics as a basis for creating one-to-one relationships.

Finally, privacy is of particular concern. As the information that is shared to allow the creation of interest-based links is typically personal (URLs of visited Web pages, bookmarks, etc.) it is required that no peer can easily gather statistics about one particular user, in particular recreating in plain text the list of visited Web pages. This means that (1) a peer that manages information for one given page (typically, as a result of a routing process in the distributed index) does not need to know the original peer's IP who issued the information. Also, (2) a peer on that routing path should not be able to spy on the information that is sees when sending it to next hops.

## 4. SYSTEM ARCHITECTURE

Based on the challenges presented in the previous Section, we sketch here a distributed system architecture that has the necessary features and solidity. An overview of the architecture is given by Figure 2.

## 4.1 Network Architecture

We consider the following network setting: a large number of *regular peers* are simply accessing the network resources and issuing the queries, and some peers that dedicate some of their processing and network capacities to the well-functioning of the network, that we denote *backbone peers.* Regular peers are unreliable, not trustable and can leave

[2]These elements can be visited Web pages URL or their representing keywords (*snippet*), bookmark tags from an online annotation service such as `http://delicious.com/`, or any other information that represents the user interests.

| **Criteria** | Information kept | Memory | Structure principle | View, reach | Churn rate | Trustability | Usage |
|---|---|---|---|---|---|---|---|
| **Indexing layer** | Pages frequencies, membership info., group mngmt info. | Long-lasting | Indexing distributed data structure (e.g. DHT) | Global view, routable | Low and fair | Correct | Reliable services for interest-proximity layer |
| **Interest-proximity layer** | Local browsing history, queries cache | Null to short-lasting | Self-emerging, proximity-based | Random-walks, local explorations | High and unfair | Unknown | Statistics collection for indexing layer, querying |

**Table 1: Interest-proximity layer and indexing layers: different peer characteristics and service objectives call to adapted structures.**

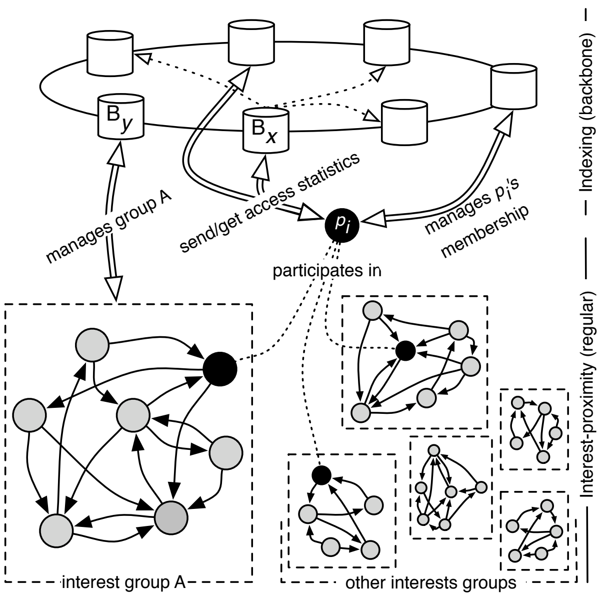

**Figure 2: Two tier architecture and functional relationships between the interest-based layer and the more stable indexing layer using backbone peers.**

or join the network at any time. Backbone peers are more reliable, and tend to leave the system gracefully. In their majority they are considered to be well behaving and trustable. Backbone peers can be, for instance, machines dedicated by ISPs or companies to the well functioning of the system. As the number of the regular peers grows, the number of backbone peers is expected to grow accordingly. We recap in Table 1 the characteristics of the two sets of peers, where we also derive the characteristics of the infrastructure-related aspects that better tie to each of them. The remaining of this Section explains each such decision and characteristic.

In our envisioned scenario, we face the problem of dealing with two types of information. The first level of information is related with global statistics (i.e., globally maintained Web sites popularity measurement). This information is used to bootstrap the interest-proximity layer, by allowing nodes to find similar peers when they are not still part of any group. These statistics are derived from data about groups. We also need to keep at a global level the identifiers and the signatures of existing groups, i.e.,the more relevant sites that distinguish the groups members. This data is used to ease all the global operation related with groups, e.g. retrieving existing groups and their features (i.e. signature), easy join/leave operations and derive global visit scores for the sites. The other level of information that is kept in the system is the local data associated with every user. This data is exploited by users to join to the network and find other group of peers sharing similar interests. Due to privacy concerns and the high variability of this information, it is not maintained at a global level. Instead, it is stored locally in every peer and only what the users allow to use is shared and used to compute similarities and contribute to the computation of group signatures.

Given the above remarks, we believe that the different nature and use of the two kinds of information present in the system require different ways of dealing with them.

In order to reflect these different needs, the overall network architecture of the system is composed of two different layers. One layer is the *backbone* layer, composed of the eponym stable and reliable peers. Due to their stability, they constitute the layer upon which global operations are possible. They are in charge of maintaining a long-lasting index that stores information concerning the global data of the network (hence, called *global* information).

In particular, the backbone layer stores the statistics about visited sites both at a global and at community levels. For this purpose it makes use of structured indices (i.e. DHT), in order to have an easy deterministic global store and retrieve operations. This information are critical to allow fast and fair community creations and maintenance.

The backbone network does not take in charge the formation and maintenance processes for the registered communities itself. Instead, this is delegated to the interest-proximity layer. Indexing the regular peers content is not possible, nor would be routing deterministically amongst those. More, the churn rate of these peers can be high enough to incur significant costs for the maintenance of such an index. But, while being self-structured and with no global view of the network at any, or from any, of its peer, this layer can still successfully leverage the indexed information from the backbone layer. The backbone layer has the responsibility to maintain this global data and the overall information (i.e. the signature) of each community. This data changes over time and is thus periodically refreshed. Nonetheless, is is not changed each time a peer performs a new action (i.e., visits a new site or increases the statistics about old sites).

The second layer is composed of more volatile, unreliable peers. These peers are attached to the users of the system. They usually have a high churn rate, since they constantly and unpredictably connect and disconnect to the network. Due to their nature, they are better organized using a self-structured network, i.e., not trying to implement a globally coherent routing substrate amongst them. Those networks are more suitable to deal with less reliable peers since they can be more easily maintained. We are interested in forming community of users based on their respective profiles. Performing such a task using only a index-based substrate (i.e. the backbone layer) will involve a very high degree of requests and update activities for this layer. Network organizations based on a self-emerging paradigm have proven to be more suitable for the creation of spontaneous communities. Hence, the network is based on a gossip-style management system. P2P Gossip-based solutions for membership management have proved to be very efficient [15, 16]. This communication and information dissemination style is used by volatile peers to start building the "core" of a community that can be later used to build a stable community, whose data can then be stored in the backbone layer, allowing peers joining the system later on to speed-up their search for suitable communities. Since this network is formed by grouping together peers with similar interests, we call it the *interest-proximity* layer.

In the interest-proximity layer, communities are formed using similarities among volatile peer profiles. More similar peers can get in touch and connect to form neighborhood of similar users. Similarity is computed on the basis of user profiles, that consist of the sites visited by them during their browsing history.

## 5. COMPONENTS AND ALGORITHMS

### 5.1 Profile Creation

Profiles of peers are created based on the users' interests, represented by recently accessed resources.[3] The popularity of resources in a distributed system, e.g. the Web, is usually very sparse, typically following a Zipf-like [28] distribution: the popularity of the $i^{\text{th}}$ least popular element is proportional to $i^{-\alpha}$ (the bigger $\alpha$ is, the sparsest is the distribution—$\alpha$ is typically around 1 for web pages popularity [1]). This means that in order to effectively decide that two users both accessing the same resource denote some kind of proximity in interest, one needs to make sure that that particular resource is not simply a vastly popular resource (e.g., `www.weather.com` or some search engine), that denotes less shared interest than mid-popular ones [2].

As a result of the aforementioned observations, it is necessary to track statistics about the popularity of pages to carefully select and weight more those that that convey more proximity of interest. To that extent, it is necessary to keep pages frequencies, based on the number of accesses by users (or on an unbiased sample or these). Clearly, the loose self-emerging but not controlled structure, with no indexing or routing mechanism, of the interest-proximity layer is not adapted for this matter: the accesses of neighbors in the interest-vicinity of some peer to Web content is itself biased by that interest proximity used to build the network. Instead, it is necessary to propose a more global and organized view of the system that would allow disposing of this information (equivalent to inverse document frequencies for data mining). This requires the different architectural choices presented in the previous section.

Based on Web sites' popularity that follow Zipf-like distributions, we extract and exploit what we call the MRFVS: the middle-range frequently visited sites. Namely, the sites that a user visited with high frequency individually but that are not the most frequent ones over all accesses by all users. The surrounding idea is that on one side the sites that are too frequently visited are not eligible for representing the user because they are accessed nearly by everyone but on the other side, the sites that are accessed only a few time are not sufficiently frequent in the user browsing activity for being considered as interesting from the point of view of the user.

The assumptions stating that MRFVS are the most relevant among the whole set of sites is not new, indeed, it is well-stated that the significance of the items in a Zipf-like distribution follows a Gaussian distribution that is maximized in the area we are considering for extracting the MRFVS. Nevertheless, in order to be able to exploit that information two issues have to be addressed: i) to decides the proper range of sites to consider and ii) to store the global statistical information for allowing to each single peer to extract from her/his browsing history the sites belonging to the ones globally considered as relevant. The former issue is a still open issue; the naive solution for finding the thresholds indicating the range is an iterative process that empirically decide the proper values. The latter issue can be addressed storing global statistical information about the MRFVS in the global index structure, which stores the (compacted and anonymized) list of sites belonging to at least one community signature, i.e., the sites representing the interests of that community.

### 5.2 Profile Similarity

Once a suitable method to describe each user interests is found and is coded in the peer profiles, we have to put attention on choosing a proper function to compare profiles. This is a particular relevant point, since this function determines the relationships between peers on the basis of their interests. As cited in the introduction, using simple functions, like counting the number of common items in the profiles, is not enough.

Instead, more accurate, although simple functions, should be used. Our proposal is to use a metric that takes into account the size of each profile, such as the Jaccard similarity, $\frac{|A \cap B|}{|A \cup B|}$, that have proven to be effective for that matter [10, 20].

We propose also to weight the mid-popular (MRVFS) elements in the profile at the moment of the calculation of the similarity metric. The ratio between popularity and weight in the similarity computation for pages is similar to the use TF-IDF (Term Frequency - Inverse Document Frequency) in data mining algorithms, except that the content of the document itself is not indexed.

### 5.3 Community Creation

As soon as each peer is able to compute its interest-based

[3]The information sources we use for extracting the data related to user browsing activity are, essentially, the history of visited web sites, the bookmarks saved by the user, the submitted queries and a either implicit or explicit user provided relevance feedback. Such gathered information have to be considered in a proper way, hence taking care of the time elapsed since the site have been visited. Indeed most recently visited sites are very important ones whereas the ones visited since a long time can be considered as not very important ones.

distance to any other peer, based on both its profile and that peer's profile, and based on information about the popularity of elements that compose the profiles (either directly or indirectly), its objective is to *group* with other peers that have close-by interests, in order to form the basis for *interests communities*. This process is done in a self-organizing and completely decentralized manner. Each peer knows a set of other peers, called its interest neighbors, and tries periodically to choose new such neighbors that are closer to its interest than the previous ones. This is simply done by learning about new peers from some other peer or from a bootstrap mechanism, then retrieving their profile, and finally choosing the $C$ nearest neighbors in the union of present and potential neighbors.

The bootstrap mechanism leans on the backbone network. A peer $p$ that joins the network can ask to the backbone layer to send it links to peers belonging to the most similar communities available in the network. The similarity is computed between $p$'s profile and the communities' signatures. The backbone layer selects the most suitable candidate communities, sends their IDs and the computed similarity to $p$, which, in turn, select the best and proper communities to join, on the basis of thresholds on the similarity score. For the selected communities, the backbone layer retrieves some contact peers inside each community and puts $p$ in contact with those. The join process can then continue using the volatile network. Indeed, the selected peers send to $p$ some other links from their neighbor list, these links lying inside the chosen community. The process stops once $p$ reaches the desired number of neighbors for all the communities it has joined, and the gossip exchanges of neighborhoods information helps with maintaining a high quality of peers neighborhoods thereafter.

When a peer enters the network, it is put in contact with one or more peers already taking part in the interest-proximity network. They use the profile similarity function to compute how similar they are. Moreover, the peers contacted by $p$ use the same similarity function to determine which are, among their neighbors, the most similar to $p$ and route the join request of $p$ toward them. All the peers that receive that request will react using the same protocol described above. This mechanism will lead $p$ to learn the existence of its most similar nodes in the network and allow it to connect with them. In doing this process, the involved peers can only use global usage statistics to compare their respective profiles. Since close similarity scores can be obtained by using different sets of sites in the peers' profiles, $p$ can use the information given by its newly added neighbors to group them on the basis of the most common visited sites. These groups try to reflect how neighbors are divided, considering $p$'s different interests.

Since $p$ is not yet part of any community, it can try to create a new one, starting from its neighbors' groups. Groups represent the seeds of new possible communities. If the cardinality of the group neighborhood exceeds a given threshold, $p$ can start a new community creation election process. Using the public profiles of its neighbors, it constructs the signature of the new potential community. Then, it asks its neighbors whether or not they want to join the new community. In the case votes for the adhesions are over a given threshold, the new community can be built. $p$ can request a new identifier to the backbone network, spreads it among the other community members and then sends the signature to the backbone layer. This layer will then keep the information about the signature and the frequencies associated with the signature's sites.

Related with the communities' maintenance processes are also the split and fusion processes. The split process happens when a community has grown too big. This kind of evaluation is performed locally, at the interest-proximity layer level. The initiative can be taken by any peer of the community that, by simply checking its neighbors table, discovers that it has too high a number of neighbors for that community. It can then decide to initiate a split.

The split proceeds as follows. The first step consists in computing the similarity with the other community members and tox take the first $C$ (with $C$ large enough) of them as the possible candidates for building the new community. Then, it computes the signature of the new community and sends it to the new potential members asking them to cast votes for the community creation. If the number of adhesions is sufficient, the new community can be created, by communicating it to the backbone layer. As a consequence, the signature of the old community has to change. This operation could be done by the backbone layer. Updates of local routing tables can be done through the interest-proximity layer, via messages propagated by the community members to their old neighbors. In principle, a node can take part to both the old and the new communities. The split simply give more "specialization" to the neighbors, by better focusing their interests and, thus, the relationship among them.

The opposite operation of a split is a fusion or merge operation. In this case, a peer may observe that the number of neighbors, over some period of time for a given community have fallen under a given threshold. Hence, it may start a merge process. It works in a similar way as the join process done by a single peer. The difference, in this case, is that instead of using the peer profile, the community signature is used. Once the most similar other community is found, the new signature is computed and an election request to the peers of both communities is sent. In case the two communities decide to merge, a request for a new community identifier is sent to the backbone network that register also the new community signature.

## 5.4 Community Signature

The signature of a community represents the cumulative (related with the neighborhood clustering for dealing with heterogeneity) profile of the set of peer that has joined that community. As for a single-peer's profile, it is represented by the sites with mid-frequency, considering all the sites visited its members.

The signature is created at the creation of a community. Peers that have agreed to become part of it communicate their visited sites (again, in an aggregated and privacy-preserving form) to the backbone network. In this case, they communicate also the sites that have no relevance for the community, with a frequency equal to 0. This is done just to avoid further coming nodes to be restricted to the nodes added so far and be able to increase the relevance of nodes that have not yet considered relevant inside a community.

A community signature is maintained by the backbone layer. It stores the community identifier and associates to it the information about the community-visited sites. This information consists in the identifiers of the sites in the backbone network, since other stable peers are in charge maintaining up-to-date data about sites frequencies.

Every time a peer joins or leaves a community, it may introduce changes to the community signature. For this purpose, every given amount of time $T$, members of a community have to start a renewal process of the community signature. The signature has to be re-computed using the new data and, when done, peers check whether they still belong to that community or not. In case they do not, they start searching new suitable communities, using the interest-proximity layer.

## 6. CONCLUSIONS AND FUTURE WORK

The focus of this paper is on addressing the problem of clustering Web users in a purely decentralized way. This is particularly useful for enabling an automated creation of communities made from users sharing common interests. In this paper we presented the overall architecture of a peer-to-peer system exploiting collaboratively built search mechanisms. The architecture is based on two different network layers: the indexing layer and the interest-proximity layer. The first one represent the so-called backbone of the network, namely a set of "institutional", reliable and trusted peers provided by ISPs that are not expected to churn or to exploit in an improper way the privacy related data. The second layer consists of churn-prone unreliable and untrusted peers connected in a self-emerging topology according to interest-based communities of users.

We have discussed in this paper the main challenges faced in designing the architecture of our collaborative search system. These issues include the creation of user profiles and their maintenance, the similarity metrics for computing how close users are in terms on interests, and issues related to the security and privacy. We presented our main vision and proposed a set of solutions for each challenge. Yet, much work remains to be performed, both from a design and implementation point of view, toward our vision of a peer-to-peer system maintaining Web-user clusters in a scalable, precise, secure, and privacy-preserving manner. Besides the actual implementation of the system, we are currently validating our algorithms using real-world Web browsing data and working on improving the privacy-preserving and security features.

### *Acknowledgments*

The research leading to these results has received funding from the European Community's Seventh Framework Program FP7/2007-2013 under grant agreement 215483 (S-Cube). This work was conducted during the tenure of Étienne Rivière's "Alain Bensoussan" ERCIM fellowship.

## 7. REFERENCES

[1] L. A. Adamic and B. A. Huberman. Zipf's law and the internet. *Glottometrics*, 3:143–150, 2002.

[2] R. Akavipat, L.-S. Wu, F. Menczer, and A. Maguitman. Emerging semantic communities in peer web search. In *Proc. of P2PIR'06*, pages 1–8, 2006.

[3] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 353–366, New York, NY, USA, 2004. ACM Press.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 1998.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 298–313, New York, NY, USA, 2003. ACM Press.

[7] R. Chand and P. A. Felber. Semantic peer-to-peer overlays for publish/subscribe networks. In *Proceedings of Europar'05, European Conference on Parallel Processing*, pages 1194–1204, Lisboa, Portugal, Sept. 2005.

[8] D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-and-merge methodology for clustering. *ACM Trans. Database Syst.*, 31(4):1499–1525, 2006.

[9] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. *Softw. Pract. Exper.*, 38(2):189–225, 2008.

[10] P. Fraigniaud, P. Gauron, and M. Latapy. Combining the use of clustering and scale-free nature of user exchanges into a simple and efficient p2p system. In *Proc. of EuroPar'05*, 2005.

[11] F. Gelgi and H. D. S. Vadrevu. Term ranking for clustering web search results. In *Proceedings of the 10th International Workshop on Web and Databases (WebCD 2007)*, Beijing, China, jun 2007.

[12] H. Gylfason, O. Khan, and G. Schoenebeck. Chora: Expert-based p2p web search. In *Proc. of AAMAS'06*, Hakodate, Japan, may 2006.

[13] S. B. Handurukande, A.-M. Kermarrec, F. Le Fessant, L. Massoulié, and S. Patarin. Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems. In *Proceedings of Eurosys'06*, Leuven, Belgium, apr 2006.

[14] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. *ArXiv Computer Science e-prints*, July 2003.

[15] M. Jelasity and O. Babaoglu. T-Man: Gossip-based overlay topology management. In *Proceedings of Engineering Self-Organising Applications (ESOA'05)*, July 2005.

[16] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM Trans. Comput. Syst.*, 25(3):8, 2007.

[17] D. R. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '04)*, 2004.

[18] J. Kleinberg. Navigation in a small world. *Nature*, 406, 2000.

[19] H. Liu, V. Ramasubramanian, and E. G. Sirer. Client behavior and feed characteristics of RSS, a publish-subscribe system for web micronews. In *Proceedings of the 2005 Internet Measurement Conference (IMC 2005)*, Oct. 2005.

[20] J. A. Patel, E. Rivière, I. Gupta, and A.-M. Kermarrec. Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*, 2009. In Press.

[21] V. Ramasubramanian and E. G. Sirer. Beehive: O(1)lookup performance for power-law query distributions in peer-to-peer overlays. In *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2004.

[22] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware'01*, pages 329–350, Nov. 2001.

[23] K. Sripanidkulchai, B. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *IEEE Infocom, San Francisco, CA, USA*, Mar. 2003.

[24] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proc. of SIGKDD'06*, pages 718–723, Philadelphia, PA, USA, 2006.

[25] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of SIGIR-IR'05*, pages 449–456, Salvador, Brazil, 2005.

[26] T. Wu, S. Ahuja, and S. Dixit. Efficient mobile content delivery by exploiting user interest correlation. In *Proc. WWW (Poster)*, 2003.

[27] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to web search results. *Computer Networks*, 31(11-16):1361–1374, 1999.

[28] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Harvard University Press, 1949.