Social Peer-to-Peer for Social People

Njål Borch

Norut IT, Tromsø, Norway
Njaal.Borch@itek.norut.no

Abstract

With increasing amounts of information digitally available, computerized searching has become a commodity. Typical computer search services work by creating a searchable, central index equivalent to the Yellow Pages. Humans will often use such central indexes, but we are also likely to exploit our social networks. We can directly contact someone we believe has knowledge about the resources we seek. This social network is built and maintained by socializing with and talking about other people.

In this paper we present how we use a social Peer-to-Peer search infrastructure, ***The Socialized.Net****, to give computers a rudimentary social network. This allows applications to work in a more humanly natural way, seamlessly integrating centralized services and distributed, direct contact. Specifically, we demonstrate applications within Instant Messaging, Shared bookmarks and Distributed BitTorrent key files and how* ***The Socialized.Net*** *makes the creation of such applications much easier than other currently used approaches.*

Keywords

Social P2P, P2P search, Ad-hoc instant messaging, Shared bookmarks, Distributed BitTorrent files.

1. Introduction

The increasing amount of information available electronically has turned searching for information into a daily event. Typical search services work by creating an index of as many resources as possible, stored in a central database. These databases can then be queried by clients or users via a computer network. We can think of this approach as the Yellow pages, listing all known businesses. When a user wants to find a plumber, he can look up “plumbers” in the Yellow pages. While computers are very efficient at searching through large databases, this way of searching only maps to a part of how humans locate information.

Even though humans often use such central indexes, we are also likely to exploit our social networks. We can directly contact someone we believe has got knowledge about the resources we seek. Using our Yellow page example, our user can contact a friend that is a plumber. If he has no such friend he can ask a neighbor that just had her pipes redone.

Computers typically lack a social network, making it a complex task to know who to ask. Early Peer-to-Peer (P2P) networks[1] were implemented by having each node forward queries to all neighbor nodes. In this fashion, all nodes are reached by every query. This is known as “flooding”, and it scales poorly due to it's vast amount of overhead. Another approach, distributed hash tables (DHT)[2,3], use a mathematical lookup function to only contact the node(s) responsible for a resource. DHT does however require a relatively stable infrastructure in order to function, and the keys of the resources must be known in advance.

Semantic routing is somewhat more humanly inspired, in that it learns other nodes' interests. When routing a query, only nodes with interests matching the query are selected. This limits the search tree to nodes likely to provide the wanted resource. Semantic routing shows promise of both speed and scalability [4]. A social network is however much more than just common interests. Humans can like each other, they can have similar cultural, social, religious or educational backgrounds, they can meet often etcetera. In the next section, we will briefly describe the social P2P search infrastructure *The Socialized.Net*. This infrastructure give computers a rudimentary social network, allowing us to demonstrate applications that can work in a more humanly natural way. We then present our demonstrations within Instant Messaging, Bookmark sharing and Distributing BitTorrent key files. We conclude with a summary of our experiences with socializing computers, and discuss the usability of the demonstrators.

2. The Socialized.Net

The Socialized.Net (*TSN*)[5] is a p2p search infrastructure based on ideas from human social networks. The basic idea is to only send queries to nodes likely to have interesting resources. For example, if your car breaks down, you can contact a friend with an interest in cars. If he can not help you himself, he is likely to know someone who can. This creates an overlay network based on interests, where nodes seek neighbors with similar interests. This is known as semantic routing[6,7], and shows promise to scale very well [4].

TSN seeks to take semantic routing further, providing computers with a rudimentary social network[8]. This is realized by allowing nodes to interact in a more human way. When two *TSN* nodes physically meet, they will discover each other. By physical we mean that they can communicate directly, such as via a wireless ad-hoc network or a LAN. Nodes are also introduced via other nodes, similar to human introduction to friends' friends. Nodes are allowed multiple addresses, based on the idea that nodes are typically in one of multiple known locations. For example, a laptop is likely to be either at home, at work or somewhere in between (likely unreachable). Nodes also observe how their neighbors behave, such as if they actively participate in the network, spread bogus information etcetera. These observations, together with possible user interaction, are used to calculate a *preference* for nodes. Nodes also spread reputations about other nodes, making it possible to learn from other nodes' experiences.

TSN can be pictured as a very subjective personal assistant, socializing with other such assistants in order to primarily help it's own user. Objective results should therefore not be expected.

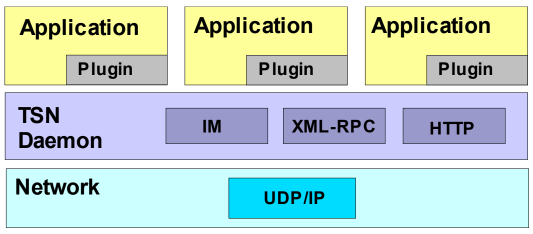


Figure 1. The Socialized.Net prototype design overview.

We have implemented a *TSN* prototype, shown in Figure 1. The prototype works by having each participating node running a *TSN Daemon*. Local applications can then connect to the various interfaces of the daemon. The http interface is a web based user interface, giving the user the possibility to directly access the daemon. The other interfaces (Instant Messaging and XML-RPC) allow applications to integrate with TSN. For simplicity the prototype only runs on IP networks, although supporting other protocols (such as Bluetooth and IR) is feasible.

TSN was designed to be configurable and dynamic as well as scalable. Applications can both specify their own meta-data structures and matching policies for the meta-data. There are also policies describing how resource descriptions and queries are handled. For example, limiting caching of resource descriptions, limiting the scope of messages etcetera.

A search performed by TSN is done in two steps. First a local filtering of incoming resource descriptions is set up based on a given query. Resource announcements and incoming replies are processed by the filter. A second, optional step is to send the query to other nodes, who in turn will forward the query to more nodes. Resource descriptions matching the query are then routed back. TSN can automatically resend the query at a given interval, allowing the daemon to actively keep searching. Note that the query can be routed differently each time it is sent, as the daemon learns more of which nodes are likely to provide results. A search is not terminated until the user (or an application) explicitly removes it from the daemon.

We also provide an optional “copy-to” field in messages. This allows applications to specify specific recipients regardless of how the daemon itself routes the message. This is a useful feature to utilize centralized services when they are available. Servers can thus be thought of as well known, powerful nodes. The “copy-to” field does not disturb distributed searches, which will be routed and performed as normal.

TSN can provide a flexible search infrastructure with good scalability. It works well in both single-hop ad-hoc networks, mobile ad-hoc networks (MANETs) and on the global Internet. TSN is however not able to give guarantees that all possible resources are found, as only a (likely good) subset of nodes are queried. Also, changes in connectivity might limit the amount of reachable nodes, possibly affecting the search results. It provides a “best effort” service, and will not guarantee message delivery.

Regardless of it's imperfection, we believe that TSN allows us to create a set of applications that inspire spontaneous communication and collaboration, while being able to utilize the services and global range of the Internet. These applications can exploit any available computer network, reducing the need for multiple, parallel solutions. The familiar conceptual ideas of TSN is likely to be easily understandable for users, making these new applications both powerful and easy to use.

Instant Messaging

Instant messaging (IM) has become very popular as an informal communication method. It allows direct, immediate communication while avoiding the disruption of voice conversations[9]. Instant messaging first became widely available to the public with the ICQ network. Other public IM networks have been established by large actors such as America on-line (AOL), Microsoft and Yahoo!. Most IM protocols require use of the owner's central IM servers. Other protocols, such as Jabber[10], permits privately hosted servers, making it possible to implement corporate internal instant messaging. All these IM protocols require centralized servers, making them unable to gracefully function in ad-hoc networks. The servers also create a single point of failure, making clients unable to communicate if the server is not reachable.

Some multi-protocol instant messengers are able to use both global, server based protocols and ad-hoc protocols. Among others, iChat and Gaim can use the Rendezvous protocol[11] in order to find local instant message users. This allows users to send instant messages in ad-hoc networks. Rendezvous does not work in multi-hop ad-hoc networks, such as MANETs and routed LANs.

Ueda, Bandyopadhyay and Hasuike[12] propose an IM protocol for Wireless ad-hoc networks based on proxies in the network. Similarly, Green and O'Mahony demonstrate Dawn IM[13], providing Instant Messaging in ad-hoc wireless networks, though without the use of proxies. They suggest that the spontaneous communication in ad-hoc wireless networks is a natural and efficient way to communicate without the need for expensive infrastructure support. These protocols do however not support global communication over the Internet, making it necessary to use different applications or protocols for ad-hoc and global instant messaging.

We use *The Socialized.Net* to provide Instant Messaging in ad-hoc wireless networks, with a seamless transition to global communication over the Internet. All communication is decentralized, enabling spontaneous, local communication and global instant messaging within a single familiar tool. Removing the need for servers omits the single point of failure seen in other, global chat applications. Both one to one messages and multi-user group chats are provided.

We have implemented a plugin for the Gaim multi-protocol instant messenger application[14]. Gaim provides a rich albeit very server centered plugin interface, enabling us to quickly implement a demonstration. We also get the benefit of testing our protocol in the same application that the traditional, server based protocols use. We should therefore get a good impression of how our protocol fares without having to take different graphical user interfaces into consideration.

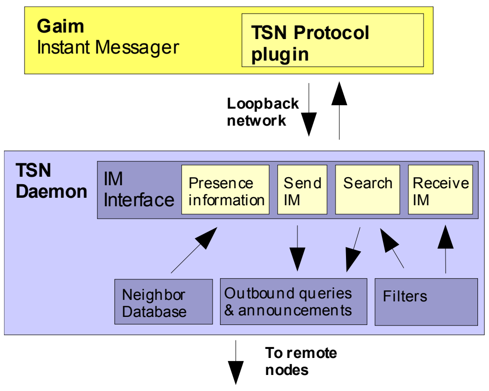


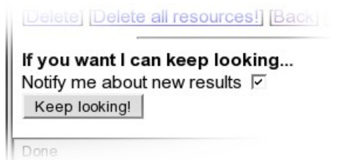
Figure 2. Design of the TSN plugin for the Gaim Instant Messenger

As Gaim is very server centred, we had to implement a simulated server in the TSN daemon. This is shown in Figure 2 as the “IM Interface”. Using this interface, the Gaim protocol plugin connects to the *TSN daemon*, which is responsible for all outwards communication. TSN is a search infrastructure and is as such not designed for generic message passing. It does however support announcements and filtering of announcements. To receive messages, Gaim therefore creates a search for instant messages, without actively sending any queries. In order to deliver messages, we send the messages as announcements to the wanted recipients. We use presence information directly from TSN, relieving our IM protocol from doing any liveness checks. Group chats are implemented by sending chat messages (including “join” and “leave” messages) to all known participating nodes. Each participating node will keep an updated resource description of the chat, containing all known participants.

To demonstrate locality awareness, we have implemented local listing of group chats in addition to global searching for both users and group chats. This allows users to find and join group chats based on being close to one of the participants. For example, a professor can create a group chat for a particular lecture. Attendants can easily find the chat, based on their presence in the lecture room. Creating the group chat is done with a few mouse clicks, as is locating it. Participants in the chat can continue communicating electronically at a later time, over the Internet or when they meet.

Our TSN plugin makes it very easy to locate users when we meet them, or when they are friends of friends. However, some times we want to find users unknown to TSN. For example, when users first start using our IM protocol, they should not need to physically meet before they can start sending messages to each other. IM protocols based on central servers can solve this easily by allowing searches through their central index. We have implemented a central register of users by allowing our protocol to send an announcement of it's user to a well known, central node. When a user searches for another user, the central node is included in the *copy-to* option. If the register is reachable, it will get the query and return any matches. Note that even if the central repository is not available, the search will still be performed as a normal distributed search. We get the advantage of centralized lookup, but do not depend on it.

As an added bonus, we also implemented instant message notifications of new hits to searches. This is specially useful when the daemon periodically retransmits queries. When new results arrive, the TSN daemon sends a local instant message to inform the user. This feature, shown in Figure 3, is surprisingly helpful as the user no longer has to manually check for new replies. For example, it can notify the user when someone has added a new bookmark that matches the user's field of interest.


[Delete] [Delete all resources!] [Back
Notify me about new results ☑
Keep looking!
Done

If you want I can keep looking...

Figure 3. The daemon can notify users of new replies via local Instant Messages

The TSN plugin for Gaim has been tested in daily use. Figure 4 shows presence information in Gaim. Several users have portable laptop computers, which are used at different locations. As expected, the IM application works whether the computers are disconnected from any other computer (although no other users are available), in ad-hoc networks or on the Internet. After being suspended for transportation, the laptops immediately upon resuming operation detect which remote nodes are off-line. All reachable nodes are typically showed on-line within 30 seconds. The Internet based protocols do at the same time use several minutes in order to detect a disconnection from their server, and must be reconnected in order to resume operation. Having the instant messenger gracefully handling mobility and off-line periods gives an increased feeling of robustness, as clearly no error has occurred.

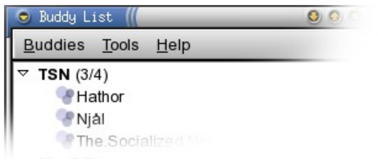


Figure 4. Reachable users in the Gaim Instant Messenger

While our IM plugin works rather well, the lack of guaranteed message delivery can affect the usage. Messages can be lost without any notifications, making us occasionally wonder whether the user is simply not replying or if the messages are lost. By extending TSN with acknowledged message transmission, we can ensure that transmitted messages have arrived. We have not implemented this, as it does not affect our demonstration to any large degree.

Group chats can also be slightly troublesome as each node is itself in charge of knowing who all the nodes in the chat are. This is not likely a problem in single-hop networks, but might be troublesome in multi-hop networks. We also do not provide any re-synchronization of splitted chats. We could minimize the synchronization risk by regularly spreading group chat information between all known participating nodes. This would also help re-synchronize slitted chats. As we have primarily used one-on-one messaging, we have not yet implemented this information spreading.

Distributed Bookmarks

Most modern web-browsers has support for bookmarking web pages. The bookmarks are typically arranged in folders by the user, and might be searchable. There is however little support for distributing bookmarks between different users. Personal web pages has historically been a place for shared bookmarks (or "links"). Such web pages must however typically be manually updated, making them inefficient for casual, quick bookmark sharing. Personal web pages are also not easily searched, making web search engines the only choice.

A more dynamic type of personal page, the weblog (or "blog"), provides a somewhat better tool for bookmark sharing. "del.icio.us"[15] can be described as a bookmarking weblog, or a "centralized social bookmark manager". It allows users to "post" bookmarks by associating them with descriptive keywords. Users can then list and surf not only their own, but also other user's bookmarks. The bookmarks are kept on central servers, introducing a single point of failure. If the central server is not available, neither are the bookmarks. Users of "del.icio.us" do in fact not have any bookmarks locally (if they do not manually add them in both places). This makes even pen-and-paper transfer of bookmarks impossible without a fully operational Internet connection. As well, it is possible to have web links to local services that are available regardless of Internet access. For example company Internet sites are likely locally available even if the company uplink (or some other, vital backbone link) is temporarily disabled.

The Safari browser can integrate Rendezvous bookmarks automatically. This is however limited to list locally available web resources. A local web server might for example notify Safari browsers about web resources it makes available. Multiple Safari browsers are unable to share bookmarks with each other. A commercial product [16] will allow LAN sharing of bookmarks between users via Rendezvous, but it is not able to share bookmarks in global or multi-hop networks.

By using TSN we can allow users to exploit their computer's social network in order to locate bookmarks. While this is similar to central bookmark sharing sites, our solution is fully distributed. Due to the seamless transition between local and global operation, we can expand our bookmark base by physically meeting other users. For example, if several users meet at a conference, their computers can learn about the other users' interests. When the conference is over, some of the TSN daemons might continue to communicate with each other. The users have thus extended their searchable bookmark base to include the bookmark of several other experts within their field of expertise.

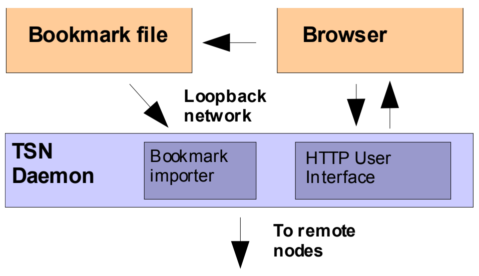


Figure 5. Bookmark integration with web browsers

We have implemented importing of bookmarks from both Mozilla Firefox and Microsoft Internet Explorer, as shown in Figure 5. When a user finds an interesting web page, she creates a bookmark with some descriptive keywords. We use the internal bookmarks of the browsers, so they will be browsable and available even off-line or in ad-hoc operation. The user can group bookmarks in folders, and the scope of distribution is configured individually for each folder. This allows the user to have private (non-shared) bookmarks or to share bookmarks to the local host only, the local network only or globally. Giving the user a choice of which bookmarks should be shared is vital, as users likely have private bookmarks. Exporting all bookmarks could violate the privacy of the user. TSN has also been integrated as a search engine for Firefox, which access the web interface of the TSN daemon. To search for a bookmark, keywords are entered into the search field, as shown in Figure 6. In Internet Explorer, the user must enter the search directly into the web interface.

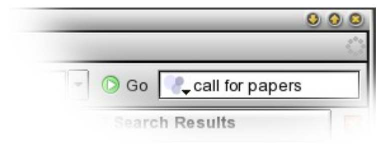
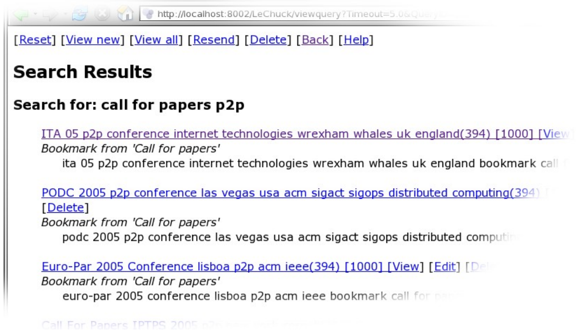


Figure 6. Integrated bookmark searching in the Firefox web browser

While the distributed bookmark plugin has partially re-vitalized bookmarks for the test users, it is only used as an extension of centralized search engines, such as Google. With a larger user base, we expect the plugin will prove even more useful. It is however important to think of TSN as a very subjective, personal assistant. This means that search replies are not likely objective. As such, our plugin is only meant to be an addition to centralized search engines, specially when searching for objective information. Sharing bookmarks with friends and colleagues is still likely powerful, as we often want subjective answers when looking for resources. Our plugin also allows users to search their own bookmarks in a convenient way. An example of search results is shown in Figure 7.


http://localhost:8002/LeChuck/viewquery?Timeout=5.0&Query

[Reset] [View new] [View all] [Resend] [Delete] [Back] [Help]

Search Results

Search for: call for papers p2p

ITA 05 p2p conference internet technologies wrexham whales uk england(394) [1000] [View
Bookmark from 'Call for papers'
ita 05 p2p conference internet technologies wrexham whales uk england bookmark call

PODC 2005 p2p conference las vegas usa acm sigact sigops distributed computing(394)
[Delete]
Bookmark from 'Call for papers'
podc 2005 p2p conference las vegas usa acm sigact sigops distributed computi

Euro-Par 2005 Conference lisboa p2p acm ieee(394) [1000] [View] [Edit] [De
Bookmark from 'Call for papers'
euro-par 2005 conference lisboa p2p acm ieee bookmark call for pa

Call For Papers IPTPS 2005

Figure 7. Results from a query

Distributed BitTorrent key files

When traditional web servers make large files available, it must transmit the entire file to all clients that request the file. For popular files the aggregated bandwidth consumption can become substantial. This makes distribution of large, popular files expensive. A higher popularity than expected can also lead to service disruption based on lack of server power, lack of bandwidth or bandwidth caps being hit. The BitTorrent[17] protocol was designed as a tool to let the downloading clients help in the distribution process. It was thus designed for a central distribution point, with a per-resource ad-hoc P2P file sharing network to lower the distribution cost.

BitTorrent works by having a torrent key file that contains cryptographic hashes of a set of files, as well as contact information for the associated "Tracker". The "Tracker" is a central node, which is responsible for synchronizing the download operation. The Tracker does not itself have any data, but it lets peers know about other peers participating in the distribution. The peers then contact each other directly, seeking to exchange data blocks. BitTorrent is as such not a P2P network by itself, but rather one distinct ad-hoc network for each torrent.

In order to share a resource with BitTorrent, the torrent key file must first be spread. This is typically done via web pages. In order to search for torrent files, large websites has appeared, such as FileSoup, Legaltorrents, LokiTorrents and Suprnova. These sites list a great number of torrent key files, and provide centralized search capabilities for their users. The more popular sites use excessive amounts of bandwidth, even though the torrent files are very small in comparison to the data shared by the Bittorrent protocol. While torrent files are typically less than a hundred kilobytes in size, the files shared are often several gigabytes. Another issue has been the large amount of copyright protected material shared via Bittorrent. This has led to the closing of several web sites listing bittorrent files. Partially centralized file sharing networks are also prone to these matters, as shown by the Napster case[18].

Other P2P applications also require keys similar to Bittorent key files. The eDonkey 2000 network is based on a distributed hash table (DHT). By looking up the key of a file, the responsible nodes for the resource can easily be found. In order to implement efficient searching of resources eDonkey 2000 use servers to index all resources.

By exploiting the social networks of TSN nodes, a search is largely kept within a group of nodes that are likely to share interests and backgrounds. For example, a viewer of the popular web-series "Red vs. Blue" is likely to look for every new episode of the series. The social network creates implicit virtual groups of users. This is not only efficient, but it also makes the groups more resistant to malicious nodes or bad content. If for example a node tries to spread pornographic movies by masquerading them as "Red vs. Blue" episodes, the node will not have the same interests as the "Red vs. Blue" virtual group. The node will get a bad reputation quickly, and is not likely able to keep spamming the group. A fully distributed file sharing network with personalized quality control would be a powerful tool. We believe our solution demonstrates the feasibility of such a tool.

The social grouping can also make it easier to find relevant resources, simply by reducing the amount of reachable resources to match the user's interests. For example, searching for "jazz" is likely to give the right styles of jazz when only asking fellow "bebop" fans. Searching for "jazz" in a server based system, is just as likely to return "acid jazz" resources, which are not likely of interest.

We can also envision new ways to cooperate. For example, (amateur) artists could make available uncompressed sound tracks under the Creative Commons licence (or similar) to other members of open "virtual groups". By using automatic query retransmissions, the other members of the virtual group would notice any new tracks, and within the hour they could be modifying and extending the track. The resulting track could then be re-released in the same manner, further triggering remixing and extension. TSN would handle the virtual group concept, allowing any interested user to join without any human intervention.

Using social networks to perform decentralized search and distribution of Bittorrent key files seems promising. We have implemented a TSN plugin for the Azureus Bittorrent client and tracker[19]. Azureus is written in Java, and has a powerful support for 3rd party plugins.

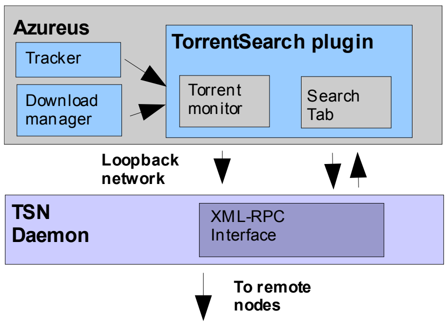
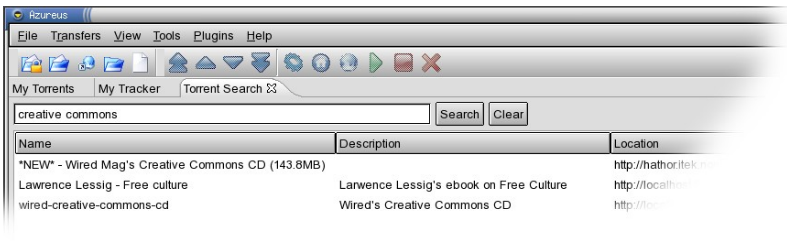


Figure 8. The design of the Torrent Search plugin

As seen in Figure 8, the *Torrent Search* plugin monitors Azureus for active and known torrent key files. It will automatically share these torrent files via the TSN daemon. While TSN is designed as a search infrastructure, an internal web server does have the possibility to serve files to any node within direct IP range of the daemon. It will therefore only work between single-hop (possibly ad-hoc) networks and between fully connected Internet nodes. MANETs and

NAT will efficiently stop these file transfers. When Azureus removes torrents, they are also removed from TSN. Our solution does not demand any extra participation from the user, creating a powerful integration with the Azureus client.

Searching is made available via a simple search tab, shown in Figure 9. The tab triggers a global query for "application/bittorrent" files with the specified keywords. As it is only created as a demonstrator, it does not handle any advanced queries, scope limitations or automatic retransmissions of queries.



Name	Description	Location
NEW - Wired Mag's Creative Commons CD (143.8MB)		http://hathor.itek
Lawrence Lessig - Free culture	Larwence Lessig's ebook on Free Culture	http://localho
wired-creative-commons-cd	Wired's Creative Commons CD	http://loc

Figure 9. Using the Torrent Search plugin

During our tests, we have had great success with the Torrent Search plugin. Sharing files, whether on the local network or globally, is simply a matter of dragging a file and dropping it over the Azureus client. The torrent key file is instantly made available and searchable. Searching is very accessible, by allowing the user to simply enter a few keywords, press "search" and double-click on any interesting results. The search process only takes a few seconds, with the first replies being listed virtually immediately. It is on par with or better than most hybrid P2P applications. The only feature we missed in the Torrent Search plugin is automatic retransmissions of queries. We got around this by using the TSN web user interface. As these searches are not often changed, this minor inconvenience does not severely reduce the usability of our demonstration.

We have not been able to verify the quality of replies due to only small scale testing. Our plugin also lack explicit user feedback to TSN. We thus only have preliminary results indicating that this works in practice.

Conclusions

A set of demonstrators emphasising and exploiting social networks and mixed ad-hoc and global communication has been created. Somewhat surprising, we found that the automatic retransmission of queries in combination with instant message notifications is perceived as powerful by the user. This is largely due to the feeling of being actively helped by the computer.

Our demonstrators show that even rudimentary emulated social skills allow us to efficiently distribute services that are otherwise difficult to decentralize. We feel confident that many applications can benefit from this concept, vastly enhancing the user experience. We also believe that many new and exiting applications can be created using *The Socialized.Net* concept.

Acknowledgements

The author would like to thank Stian Henriksen for his work on the Gaim Instant Message plugin, and Anders Andersen, Elisabeth Borch, Lars Kristian Vognild and Pål Currie for valuable feedback.

References

[1] Kan (2001) *Gnutella,* Peer-to-peer: Harnessing the benefits of distruptive technologies, ed. Oram, O'Reilly & Associates: 94-122.

[2] Stoica, Morris, Karger, Kaashoek and Balakrishnan (2001) *Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications,* in Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, 2001. ISBN 1-58113-411-8.

[3] Gupta, Birman, Linga, Demers and van Renesse *Kelips (2003): Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead,* in Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS).

[4] Joseph, S. (2002) *NeuroGrid: Semantically Routing Queries in Peer-to- Peer Networks,* in Proceedings of the International Workshop on Peer-to-Peer Computing, Pisa, Italy

[5] *The Socalized.Net*, http://the.socialized.net

[6] Crespo and Garcia-Molina (2002): *Semantic Overlay Networks for P2P Systems,* Technical report, Computer Science Department, Stanford University.

[7] Tempich, Staab and Wranik (2004): *REMINDIN': Semantic Query Routing in Peer-to-Peer Networks based on Social Metpahors,* Proceedings of the 13th international conference on World Wide Web. ISBN:1-58113-844-X

[8] Borch and Vognild (2003) *Searching in variably connected P2P Networks,* in Proceedings of the Internation Conference on Pervasive computing and communications. ISBN 1-932415-39-4.

[9] Nardi, Whittaker, Bradner (2000) *Interaction and Outeraction: Instant Messaging in Action,* in Proceedings of ACM 2000 Conference on Computer Supported Cooperative Work.

[10] *Jabber community site*, http://www.jabber.org

[11] *Rendezvous Technology Brief,* http://www.apple.com/macosx/pdfs/ Rendezvous_TB.pdf

[12] Ueda, Bandyopadhyay and Hasuike (2002) *Implementing Messaging Services on Ad Hoc Community Networks using Proxy Nodes,* in Proceedings of 2nd Swedish Workshop on Wireless Ad-hoc Networks, Stockholm.

[13] Greene and O'Mahony (2004) *Instant Messaging & Presence management in Mobile Ad-Hoc Networks,* in Proceedings of 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops.

[14] *Gaim multi-procotol Instant Messager,* http://gaim.sourceforce.net

[15] *del.icio.us socialized bookmark mangager*, http://del.icio.us

[16] Freshly Squeezed Software *Booklet,* http://freshsqueeze.com/products/booklet/

[16] Cohen (2003) *Incentives Build Robustness in BitTorrent,* http://bittorrent.com/bittorrentecon.pdf

[17] U.S. Court of Appeals for the Ninth Circuit (2001) *A&M RECORDS V NAPSTER,* Case Number: 00-16401.

[18] *Azureus Bittorrent client*, http://azureus.sourceforge.net