Using Centralized Element in P2P Network for Better Community Management

Adam Morávek, Ivan Jelínek

Abstract: *In this paper we are focusing on description of a new possible Internet searching system. It uses Peer-to-Peer (P2P) model to build a clustered logical network using a multi-ring topology. We are also aware of the fact that without centralized element is the searching very difficult. For that we introduce central data-topics managing server as the intermediary between user (P2P node) and desired information. When using both models – P2P and Client/Server – we introduce two-way searching approach. For better node identification we use the meta-addressing which is also necessary for managing node profiles.*

Key words: *P2P, Peer-to-Peer Network, Searching, Community Management, Relevance of Search Results*

INTRODUCTION

Searching is one of the present Internet problems – the process of uncontrollable increasing of information amount means also an increase of time needed by the indexing subsystems of most centralized search engines (such as Google [1], Altavista [2], etc) to index new web pages – today, the typical indexing latency is approximately of order of days [3]. The centralized search engines have, except the fact of their non-actual indexes, more disadvantages: they are not able to interpret queries individually (depending on particular host) and they index only few resource types (web pages and common document formats). Centralized searchers are seldom focused on database search [4].

However, these problems can be solved by using Peer-to-Peer (P2P) distributed architecture [5]. Unfortunately, existing P2P file sharing systems also suffers by user- and information amount growth mentioned above – P2P networks are becoming more loaded, which highly affects the way of query and message propagation, data replication, etc [5]. To minimize these effects, it is, in our opinion, necessary to use appropriate P2P topologies in combination with centralized part, which can offer some advantages – speeding-up and central management.

1. EXISTING SYSTEMS

In this paper, we will focus on hybrid Peer-to-Peer system and for that it is useful to mention two systems – Napster [6] and Gnutella [7]. Both of them act as an inspiration to our approach. Even though the Napster is not pure P2P system (it uses centralized host database), it is here taken as P2P, because it offers similar functionality.

1.1 NAPSTER

Napster combines the advantage of both Client/Server and P2P model – it manages a central database of the data which is possible to download from registered hosts. The hosts upload the list of data they offer to the Napster server. The search process is then very quick – searching host asks the Napster server for locations of chosen data and Napster server then responds with IP-addresses of those locations (node addresses). But a problem would appear, when the Napster server shut down by various reasons – the whole system would become unserviceable.

1.2 GNUTELLA

Gnutella, on the other hand, is fully distributed (P2P) system – there doesn't exist any central point which may cause the whole system to shut down. Gnutella could not be shut down even at all [5]. However, recent snapshots of visualized Gnutella network shows, that there exist 'locally central' nodes (this appeared due to the new Gnutella response packet type, which is broadcasted instead of directly routed [5]).

As Gnutella is originally fully distributed, searching hosts have to propagate their queries over the whole network (or a part of it, depending on TTL value of queries). This process is much slower than the Napster search. For propagating queries, Gnutella uses the flooding algorithm, which causes the high network load.

1.3 GNUTELLA AND NAPSTER PROBLEMS

Napster and Gnutella characteristics show that Napster is fast, but not resistant to Napster server failure. On the other hand, Gnutella is slow, but, due to its architecture, resistant to node failures. Both of these systems do not offer community support – communities are very important, because in communities nodes with similar focus (i.e. data) are grouped together. This can qualitatively improve the search process [8]. In general, P2P model allows to react on a query individually, which means, for example, to answer the query '1 + 1' by returning the value '2' *or* returning some documents containing the string '1 + 1'. Even though Napster and Gnutella are P2P-based systems, they do not have very good support for this feature.

2. ANOTHER APPROACH

Summarizing all the problems mentioned above, we realized that it is time to develop a new system which will be able to solve them (or at least few of them) efficiently. We thought that mixing the Napster and Gnutella characteristics would bring some results. We also noticed that the power-law [9] behavior of Gnutella network is not very legitimate – some nodes are abused (overloaded) due to their content. Our research consists of some major improvements of Napster and Gnutella mixture.

2.1 MULTI-RING TOPOLOGY

As the underlying logical topology, we have chosen the multi-ring topology. The reasons for that were simple – a ring can be easily managed, i.e. nodes can be easily inserted and removed, we can implement some load-balancing [10] or caching techniques based, for example, on [11]. But the main and the most important reason was the fact, that nodes in a particular ring can represent community focused on some topic(s). Therefore, it should be possible to perform a directed community search, which will result in highly relevant data retrieval. Every ring may be connected to other rings to allow direct search in other communities. To make the system work, every node has to be equipped with special software which must implement several operations (see the next chapter).

2.1.1 NODE CHARACTERISTICS

Nodes are the most important parts of the whole system. Their software is responsible for indexing content, providing unified information access, adding and removing them from ring(s), detecting and bypassing node failures, which also implies rebuilding of broken ring. All communication is realized by passing messages to neighboring nodes. Every node can act as a member of multiple rings – it is human-like behavior: everybody of us is not focused only on one area/activity, but on multiple ones. Now, let's provide detailed node- and system description.

2.1.1.1 INTERNAL AND EXTERNAL LINKS

Every node maintains two link-types to connect to other nodes – internal and external. Internal links are used to interconnect nodes inside a ring and to lower the time needed to propagate a query around a ring – every node propagates a query to both its neighbors and to nodes connected by internal and external links, which causes the query to be propagated on more hosts at a time. Of course there is some limitation of internal links count which depends on the size of a ring [9].

External links interconnects nodes in different rings. Every node may have unlimited number of those external links. This link-difference give us the choice of determining where we want to search – inside the ring (we know that wanted information lies there) or also in other rings/communities (if we don't exactly know to location of the wanted information).

2.1.1.2 SEARCH-RESPONSES DATABASE

Another routing mechanism, which is however not new, is using search-responses database. It is a special database into which each node stores IP-addresses of nodes which replied with a positive search-response to equal or similar queries. This mechanism is used by almost all P2P systems to directly route queries – when node receives a query, look into its database to get some directions to nodes which have already answered that (or similar) query in the past.

2.1.1.3 LOCAL INDEX AND UNIFORM INFORMATION ACCESS

To reach the decentralization principle, each node use local index which stores information about all resources offered by that node. Each query will be processed over this index. By resource we mean every form of information/service that can be shared. This includes documents, various file-formats, databases or even CPU cycles, etc. However this feature requires a uniform access method to be developed. But our goal now is to answer the question HOW the system will work, not WHAT data will it share – this (development of uniform data access methods) is planned to be next research goal.

2.1.1.4 LOCAL TOPIC-TREE

Local index tells us what data offers the specified node. This may help, if we are searching for data containing for example given string. But what should we do if we are looking for anything covered by some topic? Using the brute-force method could help, but it requires the Internet to be much faster than today. The second way is to use directed index-search: every node assigns a categories to all records in its index. Categories can be divided into subcategories and so on. So it is created a node-specific topic hierarchy – local topic-tree. This topic-tree is then uploaded on special server (see below), to be inserted to global topic-tree. This mechanism will build a global topic-tree which will store the information about all shared topics and IP-addresses of sharing nodes.

2.2 TOPIC SERVER

As said above, the topic server will be gathering the local topic-trees of particular nodes. Then it will collect them into one global topic-tree (see fig. 1). As the domain of the topic names contain every word (even non-existing) of every language, it is really hard to build some topic-hierarchy. But, even though, we suppose that in real the topic names will be derived from real data object names or from the real interest-areas names. For that, particular local topic-trees should contain similar or equal topic-names, which will help to tie them together and build a hierarchy. To detect similar topic-names (they may contain mistakes or various attributes), we can use some hashing mechanism and base the decisions on the distance between the hashed topic-names. But what is the topic server for? It may be used for finding those communities (rings) which we need to search in.

Imagine the situation of node which is run for first time, it has no IP-addresses in its database – it knows no other nodes (rings) to connect to or search in. For that it should browse the topic server topic-hierarchy to find some suitable rings and IP-addresses of nodes from which are the rings built. Then our node inserts itself to that rings to be more accessible. Of course no node has to connect to any ring. Or even it could be in one ring while searching in another – it is sufficient to know few addresses (backup purposes – there may be node failures) of ring-nodes to propagate a query. But when some node is a member of a ring, it will not suffer by topic server failure that much as a standalone node will do. Topic server failure will not break any links so the search process can't be interrupted.

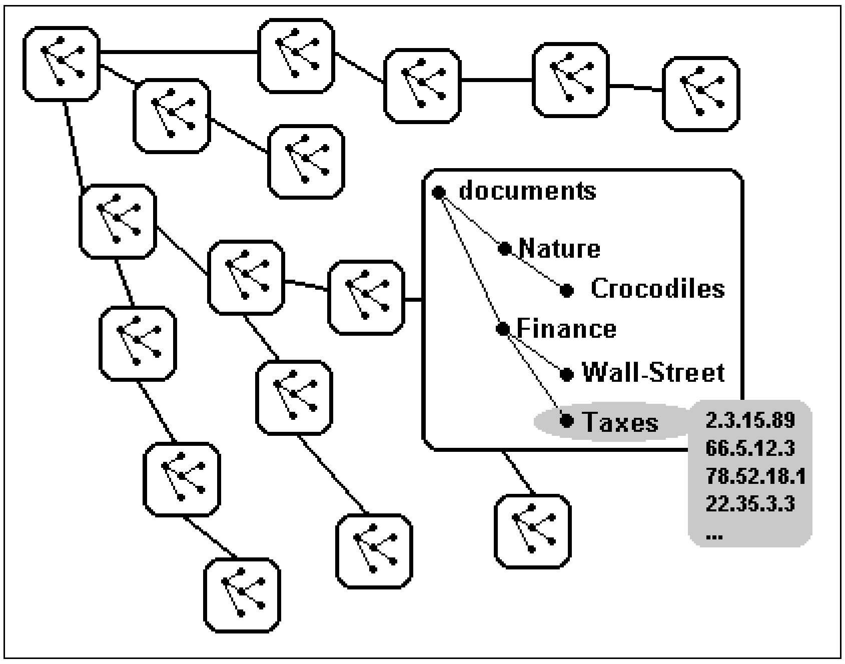

Fig. 1: Global topic-tree stored on topic server

2.3 META-ADDRESSING

To allow node the possibility to change its local topic-tree on the server or to change its online status, it is useful to manage node profiles. After the node's software starts, it may then log on the server and change the topic names. For that we should use protocol-centric addressing, while many users connect themselves using transient IP-addresses, which imposes the possibility to use IP-address as a unique identification. Protocol-centric addressing also offers independence on DNS – there is no need for it when using protocol-centric addresses. As an example of protocol-centric addresses could serve the ICQ messaging system [13] – it uses a unique number and password to identify user. Detailed description of protocol-centric addressing can be found for example in [5].

2.4 SEARCHING

When using a system with the parameters mentioned above, it could be possible to allow all online nodes to be searched by other nodes or web-users which visited some web-based topic-server search engine (it could search in topic names to gather IP-addresses of nodes) to pass them to some other search engine searching the nodes by given IP-addresses. Let's describe the first possibility – when a node wants to search other nodes in P2P network.

2.4.1 NODE-TO-NODE SEARCH

This type of search has various shapes: *purely-directed (PD) search*, *community (C) search*, *Omnidirectional (OD) search*. **Purely-directed (PD)** search means that searching node directly contacts only one node to perform a search. For example, this is the case of two friends when one was told (by the other) about some interesting data stored on friend's machine. **Community (C) search** means that node has at least one address of chosen community (ring) member and wants to search them all. This is possible due to the ring topology. Searching node propagates its query on every node in the ring. In this type of search, we do not use external links of chosen community nodes.

Omnidirectional (OD) search works the same way as community search, but the query is propagated also using the external links – for that it can reach other communities. But we suppose that every community member has similar interests and for that its external links will point (with high probability) to communities also with those or similar interests. In all these cases (see fig. 3), the usage of topic-server is not mandatory. IP-addresses can be obtained everywhere – told by a friend, on a community website, etc. But using topic server gives us the most updated and structured view – it gives us the information about what communities are oriented on what topic and, of course, their IP-addresses. Every node stores used IP-addresses to be independent on topic server in the case of its failure. No node has to be a member in any community (ring) – it may decide itself. The community membership is voluntary for everyone. But if someone wants to present his data to the world, he should join at least one ring to be accessible easily.

2.4.2 WEB-USER-TO-NODE SEARCH

This is another possible type of search. Its principle lies in fact, that not everyone wants to run special software to join the P2P network, but wants to search the nodes in it. For them it could be developed a special WWW interface to allow it. Web user will choose the category (topic), chooses the search-type (PD, C, OD – see above) and then types a search-keyword which will be encapsulated into a query and sent to IP-address(es) obtained from topic-server. All responses will be then returned in a form of webpage back to the user (see fig. 2).

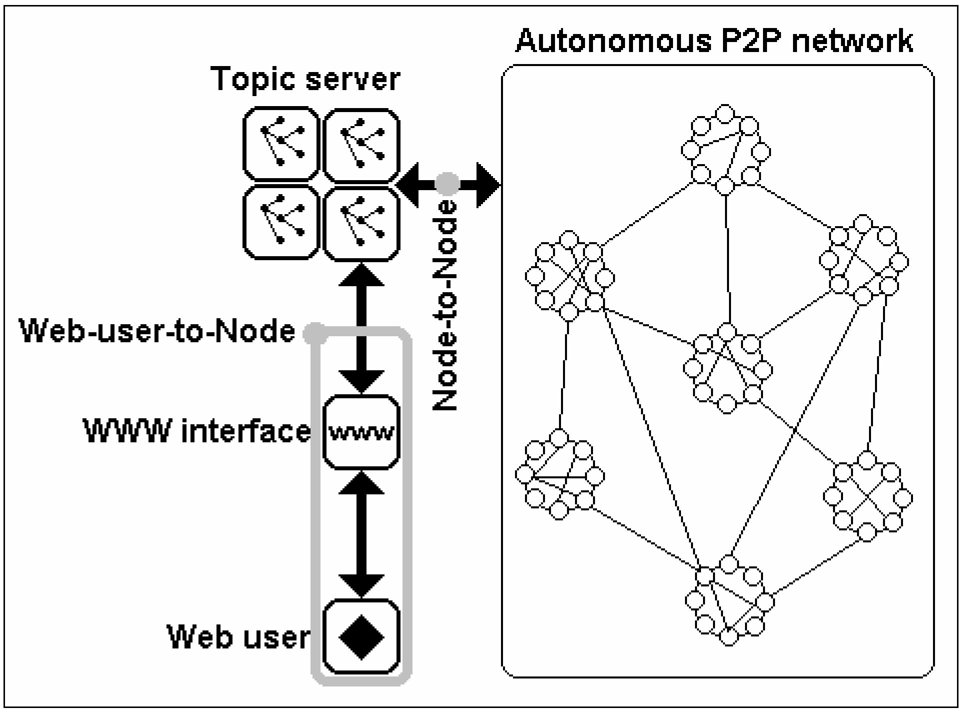

Fig. 2: Web-user-to-Node and Node-to-Node search scheme

CONCLUSION AND FUTURE WORK

This paper don't offer detailed description of the system – we do not discuss indexing and global topic-tree organization techniques or unified data access, which is needed to use in order to make all the possible node resources available for searching. All these areas are under our parallel research and will be also implemented.

As the indexing process does not require to be processed as fast as it is in the case of centralized servers (optimized for speed), we are free to experiment with some sophisticated indexing methods, which might be more effective and more data-descriptive – the index database could be more detailed. This is possible due to the decentralized manner of index database – every node maintains its own one and therefore it requires much less processing power to index the information content.

We think that the key role in directed searching has the topic-tree merging algorithm – the global tree structure is responsible for correct community detecting. The presence of the *central* topic server suggests also its many possible areas of usage – from specialized data clusters searching to various business applications. As the system of such parameters is quite hard to implement, we think that it would be better to simulate it first. We are working on such a simulator and in the time of the conference we may have the first results.

REFERENCES

[1] Google, http://www.google.com

[2] Altavista, http://www.altavista.com

[3] Esler, S. L., Nelson M. L., NASA Indexing Benchmarks: Evaluating Text Search Engines, Journal of Network and Computer Applications, 20(4), October, 1997, pp. 339-353.

[4] Bergman, M., The Deep Web: Surfacing Hidden Value, BrightPlanet Whitepaper, 2001, http://www.brightplanet.com

[5] Oram A., Peer-toPeer: Harnessing the Power of Disruptive Technologies, O'Reilly 2001, ISBN 0-596-00110-X

[6] Napster, http://www.napster.com

[7] Gnutella, http://www.gnutella.com

[8] Pant, G., Bradshaw, S., Menczer, F., Search Engine-Crawler Symbiosis: Adapting to Community Interests, Proceedings of ECDL 2003

[9] Ripeanu, M., Foster, I., Iamnitchi, A., Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design, IEEE Internet Computing Journal, vol. 6, no. 1, 2002.

[10] Rao, A., Lakshminarayanan, K., Surana, S., Karp, R., Stoica, I., Load Balancing in Structured P2P Systems, Proceedings of IPTPS 2003: Berkeley, CA, USA

[11] Blundell, N., Mathy, L., An Overview of Gnutella Optimisation Techniques, Proceedings of PGNet 2002

[12] Densmore, O., An Exploration of Power-Law Networks, Sun Microsystems Laboratories, www paper, http://backspaces.net/PLaw/

[13] ICQ Messaging System, http://www.icq.com

ABOUT THE AUTHOR

Ing. Adam Morávek, Faculty of Electrical Engineering, Department of Computer Sciences, Czech Technical University in Prague, Phone +420 604 538 070, E-Mail: morava3@fel.cvut.cz