A novel Peer to Peer framework for media sharing applications

G.S. Mamakis and A.G. Malamos

Multimedia Lab, Dep. Of Applied Informatics and Multimedia, TEI Crete
fuabap@hotmail.com, amalamos@ie.teicrete.gr

ABSTRACT

P2Pservent is a novel framework based on the Gnutella approach of peer to peer communication systems. It presents the user with hopped search capabilities based on the Gnutella protocol. It can be extended to enable the user to remotely access applications. The system uses a novel, full-XML protocol, which provides the required data verification and extensibility. The protocol structure enables remote file access, can be extended to support messaging and can be customized to adapt to the needs of a variety of applications including data sharing. It can be easily extended to provide the required safety by means of password authentication and server authorization. Moreover, it is designed to bypass any existing firewall structure easily as it operates on port 80 just like any Gnutella network, thus providing ease of use and configuration.

Keywords: Agents, Gnutella, File-sharing

INTRODUCTION

One of the breakthroughs of the last decade, regarding Internet, was the development of agents. Agents are software developed to help the user accomplish a variety of tasks when using the Internet, in a more relaxing and reliable way, than before. Almost every task one performs when using the Internet (e-mail reading, news reading, financial transactions etc.), can be done by means of agents. Towards the last years of the bygone century, a new kind of agent emerged, named filesharing agent. It is used to help the user find and download files shared by other users, who, like him, have the same agent enabled at the time. The structure developed at the early stages of the filesharing agents history was a number of servers interconnected, while the users-clients connected to them. This connection enabled the users to search and download files, and to be more specific, to share files between themselves. The servers used databases with the names of the files shared throughout the network, and provided the users requesting files with all the information required to initiate transactions with other users sharing these files. This centralized approach of filesharing agents had its drawbacks, and all the agents that promoted this scheme of communication, eventually faded away due to law issues. At that point only a decentralized version of filesharing agents seemed to be operable. The first approach to decentralized filesharing agents was the development of Gnutella protocol. Gnutella protocol was originally developed by Nullsoft, in order to fulfill the needs of their employees to share the files among the network operating inside the company. It uses a hop-by-hop technology to promote the commands specified by the users using a rather complex structure shown on Figure1. As one can understand a user is directly connected to another and through him he is able to forward and accept messages to and from other users on the same network.

Gnutella uses a variable named Time-To-Live (TTL) to specify the number of levels of users that will accept the message. TTL = 7, for example, represents that the message will be forwarded through 7 levels of users, one at a time. The user creating this message specifies TTL and forwards the message to the user he is directly connected to. The latter then decrements TTL value by 1 and in turn forwards the message to the user he is directly connected to and so on. The users that are rather remote to the initial user would never get the message. The reply messages use the same path to reach the initial user. This technique is called hopped or hop-by-hop and is the basis of this protocol. The protocol used to uploading and downloading files in Gnutella applications is HyperText Transfer Protocol (HTTP). Once a user requesting a file gets a result, through the technique already discussed, he creates a message and directly sends it to the user that provided the result, requesting the specific file via HTTP on communication port 80. The user receiving this message becomes a web server and provides the user with the specific file. It is understood that each user is a server for another user, while still being a client himself to another user. This is the way de-centralization is achieved in Gnutella protocol. While Gnutella protocol was originally developed for filesharing purposes, the structure provided, can be extended to a variety of uses. Still, the main drawback of Gnutella protocol and applications using it (Gnutella-clones) is the fact that it is rather slow, due to the messages being passed on from user to user. In addition to that, the need to promote messages from one user to another results in a lot of data being distributed over the network for simple operations.

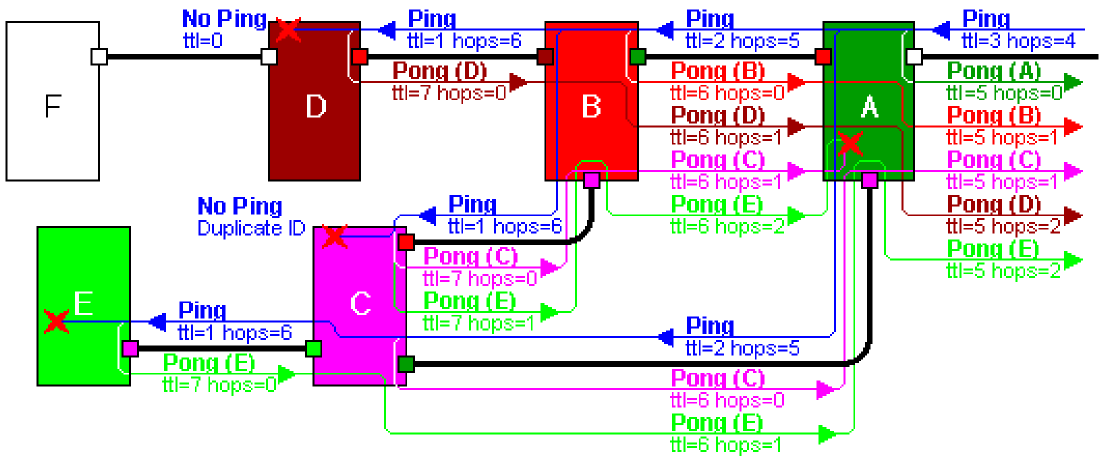

Figure1: Ping/Pong packet routing, with filtering of double messages, taken from the Gnutella Reference File

2. DATA SHARING

From early on in mankind's history, when the first communities where created, one of the primary characteristics in order for them to be viable, was sharing the required tasks to be done to the people forming these communities. This primary need for communities still applies to modern, well-organized societies. Each employee is assigned to perform some concrete tasks within a company, according to his qualifications. As experience has shown, an employee is more productive when the working environment is appealing. Since companies, are interested in getting the most of their employees one of their primary targets is to provide the employees with such an environment. The latest fashion in this approach, determines that the best way to get the most out of the employees, is through data sharing collaboration. Data sharing is generally defined as providing information through different channels to users under specific contracts. This implies, that users work at their own system, in their own environment and access data without prerequisites. Even though, data sharing has little in common with filesharing agents, the technology acquainted through peer to peer systems, which are primarily used for filesharing purposes, can be extended to satisfy the needs of specific areas of data sharing[2][3]. The main topic of interest regarding data sharing and peer-to-peer systems is to provide with a system, based on p2p, capable of handling the remote connection between user and the data repository, whilst, at the same time, satisfying the basic requirements of such a connection. The basic needs of data sharing can be described as follows: A data sharing application has to be secure, meaning that only the users, might have access to the specific system, efficient, that is able to handle without crashing or errors in connection, and easily extensible, in order to satisfy the increasing needs of data sharing.

3. P2PSERVENT

P2PServent is a novel application, based on a new protocol, resembling at some point Gnutella protocol. It was originally developed as a filesharing agent, with embedded de-centralization and hop-by-hop technology. It uses both Transfer Control Protocol (TCP) and User Datagram Protocol (UDP) as transmission protocols depending on the kind of message to be transmitted. It is designed to create Virtual Local Area Networks (VLANs) of different users connecting to a specific server, thus enabling them to exchange files between themselves. The scheme presenting this capability is shown on the Figures 2-5.

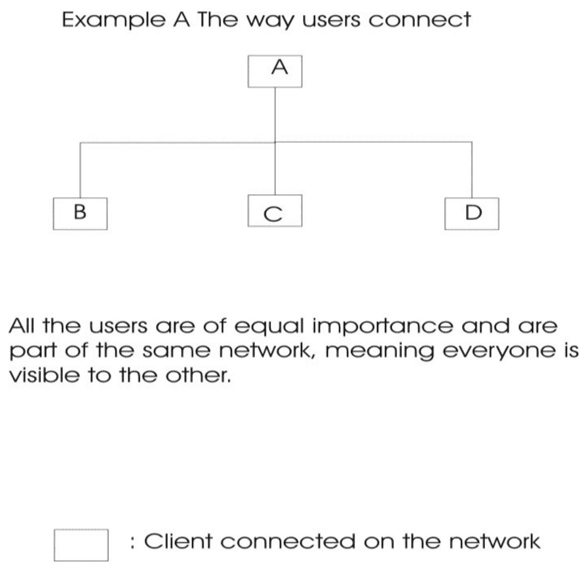
Example A The way users connect
A
B
C
D
All the users are of equal importance and are part of the same network, meaning everyone is visible to the other.
: Client connected on the network

Figure2: Primary Connection between users

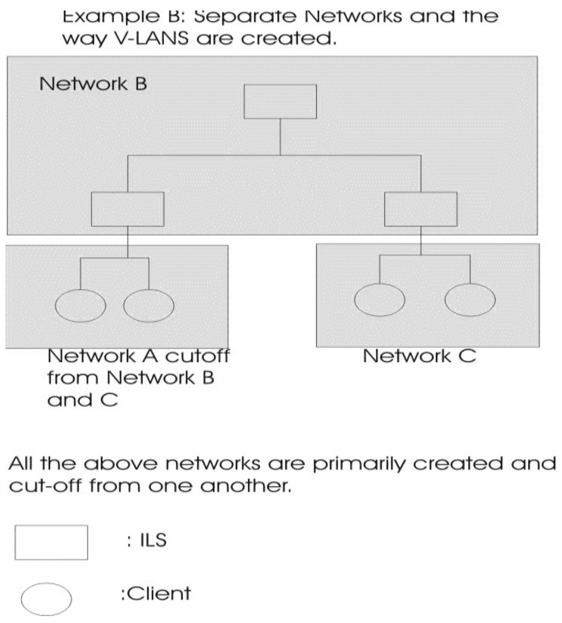
Example B: Separate Networks and the way V-LANS are created.
Network B
Network A cutoff from Network B and C
Network C
All the above networks are primarily created and cut-off from one another.
: ILS
:Client

Figure3: VLAN creation

At that point, it can be said that it operates in the way File Transfer Protocol (FTP) does. Each user can connect to one of the known users that are already connected to the same server. Moreover, after the connection, he can browse through the remote user's files, just like any FTP client on an FTP server. The application apart from the described capability enables the user to search for required files through a hop-by-hop technology similar to Gnutella. The routine responsible for file searching is also responsible for interconnection between VLANs, taking advantage from the fact that the application has a dual nature (both server and client). This is accomplished as shown in the pictures.

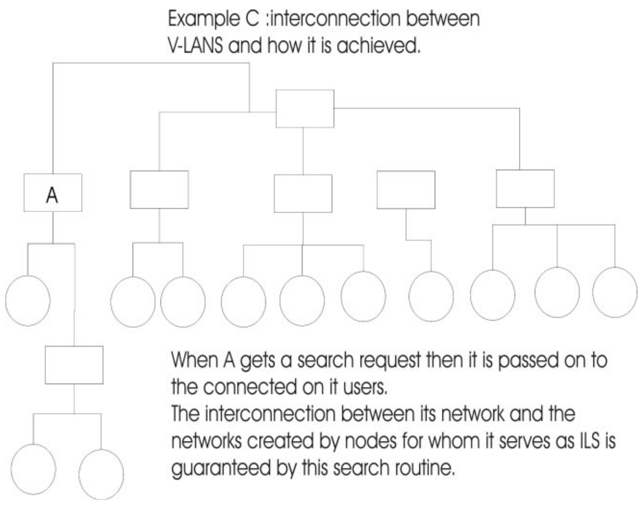
Example C :interconnection between V-LANS and how it is achieved.
A
When A gets a search request then it is passed on to the connected on it users.
The interconnection between its network and the networks created by nodes for whom it serves as ILS is guaranteed by this search routine.

Figure4: Interconnection between VLANs and search routine

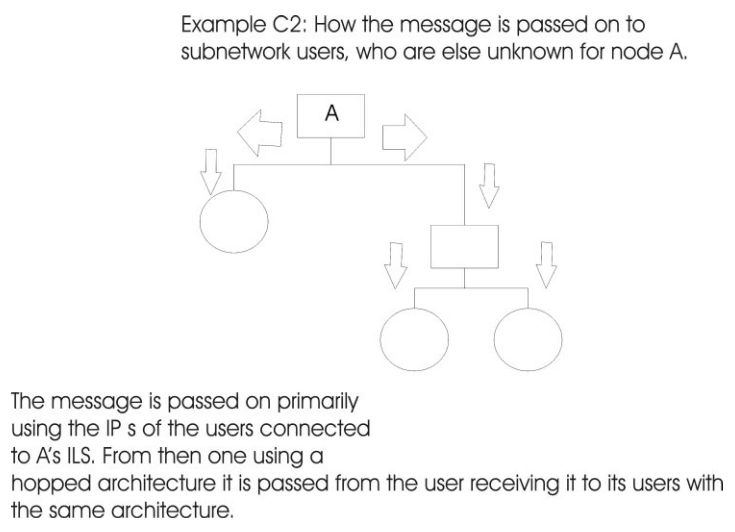
Example C2: How the message is passed on to subnetwork users, who are else unknown for node A.
A
The message is passed on primarily using the IP s of the users connected to A's ILS. From then one using a hopped architecture it is passed from the user receiving it to its users with the same architecture.

Figure5: Transmission of Search messages inside the network

As illustrated in these figures, the messages at the point of them being created are passed on to all the known users of the specific users. When received something like that is not necessary. The only thing necessary is to forward these messages to the users the embedded server of each client can recognize. This hop-by-hop technique is only used for the search routine. All the other commands of the protocol are directly sent to the desired client. The reason is to make the system more efficient on terms of bandwidth cost. The same applies to the replies of search messages. Each reply is sent directly to user requesting a file and not by following the same path the search message followed. The application has an integrated chat system, allowing the user to send public messages to all the users known and private messages to specific users as well. This is achieved by using multicasting technologies. The protocol discussed is a novel full-XML[4] protocol designed to provide the required efficiency to the system, top-level extensibility, in terms of adding new capabilities to the already present protocol, and backwards compatibility. The nature of eXtensible Markup Language (XML) enables the potential developer to use it for different purposes and, despite what one develops and how the protocol is extended to suit one's need, to be compatible with the initial application. The same applies to next versions of the application.

3.1 Technical Information

3.1.1 General

The structure implemented, as probably understood, is similar to the Gnutella approach of filesharing agents. P2PServent uses a variety of commands, in order for the communication to be accomplished in a satisfactory way. When entering the system, the user selects the server he wishes to connect to. This can be achieved by means of setting his own PC as server, connecting to one he has already connected to in the past, (while getting some interesting information about him), and by setting the Internet Protocol (IP) address by himself. He, then, specifies his unique username, and is connected to the specified server. User verification by means of password is yet to be implemented. Upon connection, the user learns who his colleagues are. At that point, the user can communicate with any of the existing users, by means of chatting, file searching, and file downloading. All file transactions are observed and stored for future use. File searching is one of the interesting parts of the application as it is fully controlled by the user. The user specifies which files he wishes to be available for search by other users, he provides a short description about the file, and, while searching himself, he even specifies the number of hops of the forwarded search message (how many levels of users will get the message). Moreover, file searching enables the user to communicate with other VLANs. The SEARCH message is the only message throughout the protocol that has the HOPS feature, which specifies the levels of users (VLANs) receiving this message. This is all achieved by extensive use of XML, from the protocol itself, to file uploading and downloading statistics.

3.1.2 Protocol Structure

The protocol itself is divided into 2 categories, based on the kind of transfer protocol used and into another 2 categories based on the complexity of the command. As for the first kind of division, TCP and UDP commands exist. TCP commands are utilized for the connection and safe data transfers between users. TCP connection is required for the users to connect to the server and acquire the list with the users that are online at the time, the connection to a single remote user, in order to browse through his files and initiate a file transfer and for the file transfer itself. The connection to a server demands a series of commands to be utilized in order for the users to actually register to the server. First of all, a HI command is sent upon connection to the server, forcing him to reply with a NAME command, requesting the remote user's data. This data is his unique name and IP address and are registered in a dynamic dictionary object. The reply is a NAME command formed by this data. After the user has finished up with connection, automatically he requests the list with the users that are online at the time, using a RETRIEVE command. The server is then obliged to provide the user with this data in an organized way so that the user can easily distinguish which user has provided this data, which are also their username and IP address. This command is called LIST. These elements are registered and ready for use. This is the primary procedure needed for a user to become an active member of the system. In order for the user to connect to another member of the system he must select him from a list and another procedure like the one described before is initialized. Upon connection with the remote user, the latter provides the former with the file structure (the files and folders) that he shares which reside in a predefined folder inside the install folder of the system. The request message is called FILELIST, while the reply message is called FILEANSWER. While browsing through the folders, PATHFILELIST messages are created and sent, that specify the folder the user wishes to browse through, getting the same reply message as in FILELIST. If the user wishes to download a specific file, the application creates a command called FILEGET, which specifies the file to be downloaded. The reply is FILESEND, and initializes the actual data transfer.
On the other hand UDP commands are used for multicast messages (messages that are to be delivered to a lot of users at the same time) and therefore are used to implement the chat system, file searching, and access control of the users making up the system. Access control is the sole way the system can verify whether a user is online or not. There are two chat systems available, one for private and one for public chatting. The command used for chat communication is CHAT and is formed up by the users' IP and address and the type of chat the user wishes to enable. File searching is implemented using a SEARCH command made up by the user data (IP address and username), the keyword specified and the number of HOPS, declaring the number of levels of VLANS that will acquire the SEARCH message. The remote users acquiring this file register the sender if not already registered for future use and try to find the keyword in a specific XML file where are the files shared for file searching exist. The file searching is not implemented only using the filename but is also implemented taking advantage of the fact that each file registered in this XML file has a small description provided by the user himself. The reply message to SEARCH is a UDP message, which is called RESULT, directly sent to the sender, with all the information needed so as the sender can initialize the file transfer of one of the files. Important commands to the system are the ones used for access control. These are STATUSCHECK and USEREXIT. The first is created and sent to a server when a user tries to connect to a member of the network not available for some reason. The server then tries to establish a connection with the specified user. If this is not possible then the specified user is unregistered and the STATUSCHECK is forwarded as is to the sender to all the users that belong to the same VLAN. On the other hand USEREXIT is the notification sent to the server forcing the latter to unregister the sender as he is exiting the system. This command is not only sent to the server one belongs, but also to all the users he is aware of.
The second division described above refers to command complexity and creation. There are commands, like HI, that apply to a generic rule of creation and some others, like LIST, that use a dedicated rule for their initialization. The generic rule specifies that a XML command is made up by only the root element, called COMMAND. Therefore, any command following the generic rule looks like this:

```
<XML>
<COMMAND>Name of command </COMMAND>
</XML>
```

That being the case, HI command is formed as follows:

```
<XML>
<COMMAND>HI</COMMAND>
```

```
</XML>
```

These commands are simple and are used to initialize connections and force the remote nodes to take specific actions. They do not provide the remote users with any interesting data, and therefore are used only for basic actions.
In the system more complex commands are specified, each of them having their own distinctive way of forming. Still, they conform to the rules specified by the XML reference file. The way they are formed is illustrated below:

```
<XML>
<ELEMENT>
        <SUBELEMENT>Value of parameter</SUBELEMENT>
</ELEMENT>
</XML>
```

where element defines the name of the command, subelement specifies the names of certain parameters used while the values of these parameters reside inside the subelement nodes. Therefore, LIST command which is made up of the users belonging to a specific server looks like this:

```
<XML>
<LIST>
        <PEER>
                <IP>IP address</IP>
                <USERNAME>Username</USERNAME>
        </PEER>
        <PEER>
                <IP>IP address</IP>
                <USERNAME>Username</USERNAME>
        </PEER>
</LIST>
</XML>
```

Sub-element PEER, as one may notice, is used as many times as the number of users registered to the specified server, and are made up by the data needed for one to be able to connect to each and every one of these users.
These commands are used to provide the user with all the data needed to achieve a task, as opposed to simpler ones which are used to perform tasks. This is the reason why there is a distinction between simple and complex commands.
The nature of the protocol enables it to be easily extended in the future, while still being compatible with earlier versions. The reason for that is the use of XML which implies that unknown sub-elements are just ignored and do not interfere with the rest of the known protocol or the application itself.

3.2 Potential extensibility-Drawbacks

The application can be extended to include remote desktop connections using either the Microsoft's approach or VNC extensions to accomplish that, easily without losing potential compatibility with older versions. Moreover, P2PServent features all the required stability, without any obvious bugs, apart from the ones applicable to all peer-to peer systems. These bugs have to do with the communication between users that use proxy servers to connect to the Internet. A user connected to the Internet through a proxy server (user having an internal IP address), is not able to provide any files to a user who is directly connected to the Internet through a common Internet Service Provider (ISP) (user having an external IP). In Gnutella-like applications, something like this is possible by means of uploading requested files, but this option is not yet implemented on P2PServent. The last bug is that users using different proxy servers (both users having internal IP addresses) cannot communicate at all on P2PServent. In Gnutella-like applications something like this is possible, but file uploading and downloading is out of the question as well. Still, this should not be a problem when referring to data sharing because using such software implies that the users at home have a way of accessing the Internet, most likely through an ISP, regardless of the type of the connection. So the only real need is the users on the company not to actually use a proxy to connect to the Internet. As for the safety, P2PServent provides data verification, acquired because of the use of XML. Also the protocol can be extended to provide access through password authentication, and server-sided user verification (provided that the server is one and located inside the company) based on information sent by the user to the server. Also, the embedded chat system can be used either to promote employees communication, or to provide the employees with the employer's instructions on specific subjects of interest. If needed the system can be extended to provide communication through videoconference, by employing already known technologies on this specific field. Filesharing can be used to exchange data required between the employees. As a result this system can be adopted by companies to enable the users to contact with the company's servers at any time and to present their work and progress on a more regular basis. In addition to that, the minimum requirements of such a system are far less than any telecommuting system available, thus providing greater interaction and covering the needs of any modern data sharing system.

4. CONCLUSION

Peer-to-peer systems remains a point of interest in modern computer science when referring to Internet technology. They are the main reason behind the vast development of Internet users in many parts of the world. The fact that these systems are members of the wide-spread agent family implies that they can take full advantage of the technology already applicable to the latter.

REFERENCES

[1] Gnutella Reference File, *http://rfc-gnutella.sourceforge.net/developer/stable/index.html*

[2] Khan, M.A., Yeh, L., Zeitouni, K. et al. Peer-to-Peer Netw. Appl. (2016). doi:10.1007/s12083-016-0450-7

[3] A New Document Sharing System Based on a Semantic Hierarchical Peer-to-Peer Network Anis Ismail, Aziz Barbar, Mohammad Hajjar, and Mohamed Quafafou International Journal of Engineering and Technology, Vol. 9, No. 2, April 2017

[4] Extensible Markup Language (XML) 1.0 (Third Edition), *http://www.w3.org/TR/2004/REC-xml-20040204/*

[5] Hofstatter Q, Zols S, Michel M, Despotovic Z, Kellerer W (2008) Chordella-a hierarchical peer-to-peer overlay implementation for heterogeneous, mobile environments. In: 8th international conference on peer-to-peer computing, 2008. P2P'08. IEEE, pp 75– 76

[6] Spyros Panagiotakis, Ioannis Vakintis, Haroula Andrioti, Andreas Stamoulias, Kostas Kapetanakis, Athanasios Malamos, "Towards ubiquitous and adaptive web-based multimedia communications via the cloud", Book: "Resource Management of Mobile Cloud Computing Networks and Environments" , pp307, IGI Global, 2015

[7] K Kapetanakis, S Panagiotakis, AG Malamos, M Zampoglou, “Adaptive video streaming on top of Web3D: A bridging technology between X3DOM and MPEG-DASH” Telecommunications and Multimedia (TEMU), 2014 International Conference on, 2014

[8] Markos Zampoglou , Athanasios G. Malamos, Kostas Kapetanakis , Konstantinos Kontakis, Emmanuel Sardis, George Vafiadis, Vrettos Moulos, Anastasios Doulamis, “iPromotion: A Cloud-Based Platform for Virtual Reality Internet Advertising” in “Big Data and Internet of Things: A Roadmap for Smart Environments Volume 546 of the series Studies in Computational Intelligence pp 447-470, 2014