NEW APPROACH TO DEVELOP THE MESSENGER APPLICATION: FROM CLIENT-SERVER DESIGN TO P2P IMPLEMENTATION

Ha Quoc Trung[1]

[1]Department of Data Communication and Computer Network, School of Information and Telecommunication Technology, Hanoi University of Science and Technology, Hanoi, Vietnam
trunghq@soict.hut.edu.vn

ABSTRACT

Client server application architecture is widely used in design and development of distributed application. Its advantage is the protocol design simplicity. However the architecture has several drawbacks such as server bottleneck and weak scalability. P2P architecture resolves these drawbacks by distributing computing tasks on both the client and server, making them equal in the system. However, P2P application development and protocol design are far difficult than client server model. This article proposes a solution to get the advantages of both models: the distributivity of the P2P architecture and the simplicity of the client server architecture. The solution uses local proxies, which interact with the client as the server and sharing information among them via P2P system. The solution is implemented and experimented for text information and services: messaging application.

KEYWORDS

Client Server, P2P, Messenger Application, distributed system.

1. INTRODUCTION

Most of recent applications and protocols are developed base on client-server architecture, due to its simplicity. Computational task in the client-server architecture is concentrated on the server, so the server will quickly become bottleneck in term of processing power and communication bandwidth. Extended client-server models are proposed to improve the performance. Load balancing and replication techniques allow distributing load on number of servers [1,2]. Caching is a kind of asymmetric replication of the server's data on client side. Caching is used exclusively in the proxy model, where the proxy is delegated from many clients to access server's data, and can provide information from proxy cache when possible. The proxy can be used in reversed direction, i.e. delegated from the server part. These approaches still remain in the client server model, so they cannot resolve the server's bottleneck problem [1,2].

When design application with client-server architecture, the designer has to consider interaction between multiple clients and a server. So in fact, the only kind of interaction between a client and a server need to be considered, taking account of the fact that multiple clients can interact with the server at the same time. The fact that data is concentrated on the server makes many problems of distributed system trivial: synchronization, replication, security, access control and so on.

Natarajan Meghanathan, et al. (Eds): SIPM, FCST, ITCA, WSE, ACSIT, CS & IT 06, pp. 529–534, 2012.
 DOI : 10.5121/csit.2012.2351

P2P model resolves these disadvantages by distributing the computational workload on peers (i.e. the host – members of the system). Using this model, the interaction of the system's component is changed conceptually, so the protocols and applications design is complicated. Instead of designing a protocol between client and server, now a complicated interaction between peers must be considered.

In fact, P2P architecture resolves the problem of communication and processing performance of the hosts, but it doesn't concern the interaction between peers. That explains that it is possible to mix P2P model and client server model so that the client server model is used for interaction between user agents, while the P2P model is used for communication between hosts. The system then consists of the client-server component and the P2P subsystem. The problems remained to be resolved are: (i) interaction between the client-server and P2P subsystem; (ii) adapting the system with data and application specific requirement.

In this article, a solution using local proxy agent is presented. The solutions are implemented, experimented and adapted with text data and application: the messaging application.

The article is organized as the following: section 2 introduces some related works. Section 3 introduces the Client-Server based design for P2P Application using Local Proxy. Section 4 presents the implementation experiments of the messaging application using the local proxy model. Section 5 concludes and presents some future works.

2. RELATED WORKS

Client server is most used architecture in distributed application. In [1] a review of client-server is presented. Several solutions for performance, scalability and fail tolerance are discussed: application layering; multi-tiered architecture. Other methods such as application gateway and replication are found in [2]. These methods cannot resolve the problem of scalability. P2P architecture has better performance in large scale, but it is far difficult to design a P2P protocol than a Client-server protocol with the same functionality. In [3] the P2P architecture and several P2P protocol and system are discussed and compared.

For applying P2P in client-server application, in [4] a software system allowing using P2P file subsystem for distribution of software sources. In this solution, client remains the same, but a local agent that can share software sources with similar agents on other hosts replaces the server. The problem of apt-p2p is that the client cannot function in offline mode, event if there are multiple apt-p2p agents in the same LAN.

In [5] P2P Skype is presented and analyzed in details. P2P subsystem with super peers is used to transfer multimedia data between hosts in an efficient way.

Some common characteristics of reviewed works can be summarized here:

- Proposed architectures are suitable for different types of information
- Application protocol design depends is P2P based design.
- No common way for design P2P application and protocol.

The summary shows that we need a solution, an approach to design application with client server architecture, but then implement them with P2P architecture.

3. CLIENT SERVER BASED DESIGN FOR P2P APPLICATION USING LOCAL PROXY

Consider the design of a distributed application. User needs to communicate each other by some protocol, regardless the fact that the system is P2P or Client-Server. In Client-Server architecture, this protocol defines interaction between each client and the server. Client to client interaction is realized based on client-server communication. In P2P application, host-to-host communication is point to point, no need of a central intermediate host (server).

To profit the advantages of both client server and P2P architecture, our goal is to:

- Design the protocol using the client server architecture for simplicity,
- Implement the protocol using P2P communication for better performance.

Our approach is to keep the protocol design by client-server architecture unchanged, while the implementation will adapt with P2P communication system. So the client remains unchanged. The server will accept request from client and transfers it to P2P system; it acts as a proxy. The location of the server must be the same as the client; it will be a local server. This is the reason why the server is called local proxy and the model is called the local proxy model. The model is described in

Figure 1: Local Proxy Model. The system consists of following components:

- The client modules: the same as the client modules in classical client-server architecture.
- The local proxy module: each client host has an install of this local proxy module. The module interacts with client module and updates local data.
- The P2P subsystem agent: accept request for sharing and updating data, propagandas it on the P2P network.

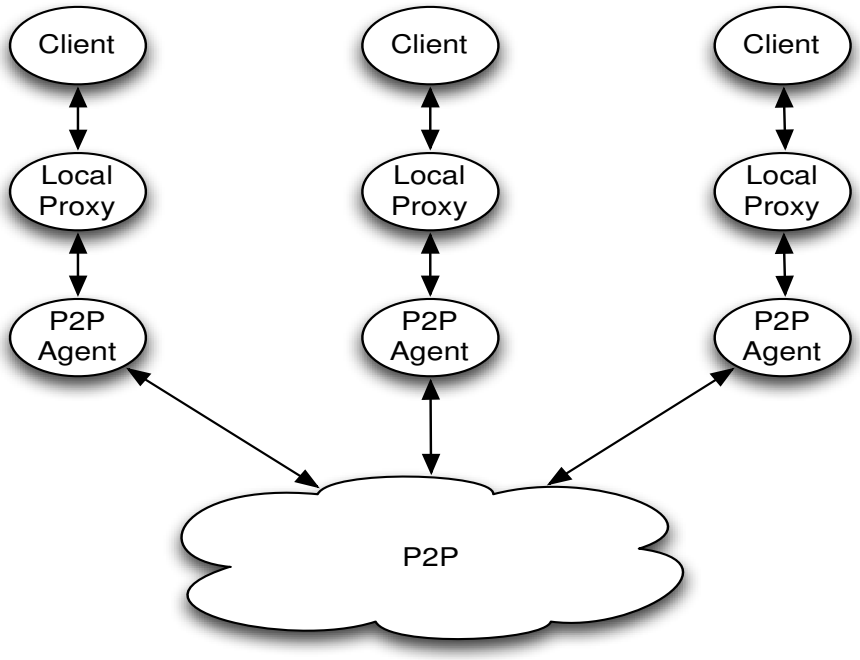


Figure 1: Local Proxy Model

The local proxy model gives the opportunity to develop a P2P application using client-server design. Of course, the protocol and application logic, in point view of host-to-host communication is realized by communication between P2P Agent and P2P subsystem. This interaction can be trivial as a simple data sharing mechanism, but to have a good efficiency, the data structure has to be well designed following the logic of the application and the P2P protocol. In next section, via the case study of messenger application, this point will be discussed in details

4. THE MESSENGER APPLICATION

The client server messenger protocol and application

The comparison of messenger protocol can be found in [6] and in [7]. Of course, what we are interested in is how to design the application with P2P communication, so we'll chose minimal functionality of the protocol.

The main functions of the messenger application and the information exchanges between client and server are described in Table 1: Messenger Protocol Functional description.

Table 1: Messenger Protocol Functional description

No	Name	Description
1	Login, Registration	Accept login request and register user in the whole system
2	Get logged user lists	Get the list of 'Online' users that can receive messages
3	Get messages from inbox	Get all the message from inbox
4	Read a messages	Get a specific message from inbox and visualize it for user.
5	Send messages to user	Send a message to specific users
7	Delete messages	Delete specified messages
8	Logout, Unregistered	Logout and remove users from the whole system

From this description a client and a server are built following the classic client-server model. This application is for testing only, so we use a very simple, command line interface.

The local proxy P2P messenger application design

Our goal is to build a P2P messenger application, but the interaction between client and server remains the same so that we can profit the description in Table 1: Messenger Protocol Functional description. We follow the model in
Figure 1: Local Proxy Model. The client module remains unchanged. No server module is needed. A local proxy module will be implemented, accept all requests from client module as a server, but share it via P2P with local
proxy on other hosts. For simplicity, our technical choices are:

- Information exchanges are organized in files and folders.
- Files and folders are synchronized between local proxies using P2P file sharing system as a communication base.

With these assumptions, each client has a shared folder: inbox and outbox. In the folder, messages are stored as a file. When user logged in for the firs time, o token file with the name

<logged><user-id> is created and stored in the shared folder. Getting list of users is done via P2P-search function for the filename <logged>. When the user logs out, the local proxy will delete the corresponding file.

The same mechanism is used for sending and receiving messages. When the user send a message, the content of the message is stored in a file with the form of the name: <message><sender><receiver><subject><controlling information>. Each time user read a message; the local proxy will send a query <message><sender> to the P2P system for downloading from other hosts' local proxies.

The functions in the Table 1 will be implemented by creating and modifying files as in Table 2: Local Proxy Data.

Table 2: Local Proxy Data

No	Name	Local proxy action
1	Login, Registration	Create a token <logged-in><user-id> file
2	Get logged user lists	Download by query "logged-in" from P2P system.
3	Get messages from inbox	Download from P2P by searching with <message><receiver-id>
4	Read a messages	Get the message from the shared folder
5	Send message to user	Put a message with file name: <message><sender><receiver><subject> to the shared folder.
7	Delete messages	Delete specified file from P2P system
8	Logout, Unregistered	Remove the user token from the shared folder.

The local proxy P2P messenger application implementation and experimentation

The application is built using JXTA library. For P2P subsystem we use Chord protocols, but for the ease of integration of local proxy, we re-code the chord file-sharing agent by our-self. This helps us to experiment the system without interference of peer not participating in our system.
For experimentation and evaluation, we face a problem of lacking a large-scale test environment. This is the reason why our experimental result is only qualitative. It proves only that the P2P messenger application has the same functionality of the client-server one. From analytic point of view, when the system become large, the P2P application should have better performance that the client server application.

5. CONCLUSIONS

In this article, we have presented an approach to implement P2P application with client-server design. The main idea of the approach is to use a local proxy in each host-participant in P2P application, keeping client unchanged. This local proxy will accept request from client, modified local data and shared it with other hosts' local proxies. This approach has several advantages for development of P2P application:

- It makes the protocol design unchanged (same as the client-server protocol design).
- It provides the performance and scalability of P2P application.
- The local proxy model can be extended to LAN proxy model, in which local proxy will serve not only for local client, but any client in the LAN section.

Based on this approach, from the client-server design of messenger application, a P2P messenger application is developed and implemented. The applications are tested and the results show that the functionality of both applications is the same as the design. No quantitative and performance evaluation is executed.

In the future, the following works should be completed:

- Joining the test-bed community such as Planet-Lab, G-Lab to have the possibility to evaluate the P2P and Client-Server application in large-scale system.
- Applying the method for other application with different kind of information exchanged: audio, video, raw and so on.

REFERENCES

[1] Tanenbaum, A. S. & Steen, M. v. (2006) *Distributed Systems: Principles and Paradigms (2nd Edition)*. Prentice Hall

[2] Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011) *Distributed Systems: Concepts and Design*. Addison Wesley

[3] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., Lim, S., & Lim, S. (2005) *A Survey and Comparison of Peer-to-Peer Overlay Network Schemes*. Communications Surveys & Tutorials, IEEE, 72--93

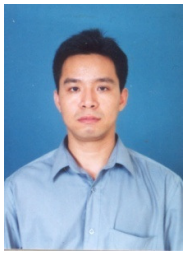
[4] Dale, C. & Liu, J. (2009) *apt-p2p: A Peer-to-Peer Distribution System for Software Package Releases and Updates*. INFOCOM 2009, 864-872.

[5] S. A. Baset, H. G. Schulzrinne, 2006. *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol*. INFOCOM 2006

[6] Reed, D. (1992) *A Discussion on Computer Network Conferencing*., Request for Comments: 1324

[7] Carl von Loesch, 2003, *Functionality provided by systems for synchronous conferencing.* http://http://www.psyc.eu/ ,

Author



Ha Quoc Trung (Ph. D.) Graduated from Technical University of Sofia in 1996, speciality “Electronic, Microelectronic and Automation”. In 2004 he completed his Ph. D. degree in School of Advance Research Practice (Ecole Pratique de Hautes Etudes-Paris, France). From 1999 to recent he is lecturer at department of data communication and computer network. His research interests are: distributed algorithm, system and application, ubiquitous computing.