International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
*ISSN (Online)*2349-6967
Volume 2 , Issue 1(Jan-Feb 2015), PP223-230

CLIENT SERVER BASED SECURE CHAT APPLICATION USING PEER-TO-PEER NETWORK ARCHITECTURE

Rushabh Balpande[1], Chetan Dusane[2], Khushboo Kashyap[3], Nandkumar Patil[4]

[5] Prof. Sunita Patil

[1 2 3 4 5] *(Dept. of Information Technology, Dr.D.Y.Patil College of Engg. Pune, Savitribai Phule Pune University, India)*

Abstract: This paper proposes a system that can be used for communication between peoples located at different places using a chat application which is secure and it uses client server architecture to connect peer-to-peer networks. The system is proposed for an organization in which the administrator can communicate with the project leader and programmer. The administrator which acts as server can keep track of the login time and logout time of the employee which acts as a client. The proposed system uses Node.js technology also known Node and it is a server-side JavaScript environment. Node.js is based on single as well as multiple thread by thread execution. The proposed system uses CompuP2P uses peer-to-peer networks for sharing of computing resources. The proposed system uses Collaborative Locality-aware Overlay Service (CLOSER), an architecture whose aims is to lessen the usages of expensive international links by exploiting traffic locality. It also includes the privacy module that may arouse the user interest and encourage them to switch to the new architecture. The proposed method based on sending/receiving messages in intranet through intranet server via Wi- Fi connection without the need of taking any service from mobile service provider and without the use of internet connections.

Keywords: Administrator, Architecture, Bluetooth, Chat, Client, CLOSER, communication, employee, internet, network, Node.js, peer-to-peer, server, technology, Wi-Fi.

I. INTRODUCTION

Internet computing is a distributed computing paradigm that uses the Internet as a single large virtual computer. Internet computing promises to fulfill the vision of "anytime" "anywhere" computing. Application benefiting from this paradigm range simple data sharing to ones using the Internet as processing engine for large scale data and distributed task execution. Internet computing is challenging to realize primarily because of its sheer size and open unworthy environment. In the last decade, the concept of Internet computing has been revolutionized due to application such as file sharing developed around peer-to-peer (P2P) paradigm. We believe that P2P paradigm has the potential to serve as a platform for developing several "killer-apps" for making true Internet computing a reality (and also affordable). Peer-to-peer (P2P) networks are flexible distributed system that allows nodes (also called peers) to act as both client and server and provide service to each other. P2P is powerful emerging networking paradigm that permits the sharing of virtual unlimited data and computational resources in distributed, fault-ignorant, measurable, and in a flexible way. Node.js also called Node is a server side JavaScript environment (see http:// nodejs.org). It is based on Google's runtime implementation — the aptly named "V8" engine. V8 and Node are mostly implemented in C and C++ language, concentrating on performance and low memory consumption, But, whereas V8 supports mainly JavaScript's in the browser. Node aims to support long-running server processes. Current Peer-to-Peer (P2P) file makes use of a considerable percentage of Internet Service Providers (ISPs) bandwidth. This paper present the Collaborative Locality-aware Overlay SERvice (CLOSER), an architecture that

aims at lessening the usage of expensive international links by exploiting traffic locality (i.e., a resource is downloaded from the inside of the ISP as required). The paper also shows the effectively of CLOSER by analysis and simulation, as well as comparing the proposed architecture with existing solution for traffic locality in P2P system. While saving on international links can be very good for ISPs, it is essential to provide some unique features that can be of interest for users to favor a wide adoption of the application. Due to this very reason, a privacy module is introduced by CLOSER that may arouse the user's interest and encourage them to switch to the new architecture. Some interesting application for CompuP2P is giver as follows: 1. Allowing processing limited devices, such as wireless client, to distribute their processing requirements to other machine in a network..2. Utilizing the storage capacity of virtually millions of machines connected to the world wide web, etc.

II. LITERATURE SURVEY

The following papers are used as part of references and which provide us such ideas about your proposed system.

2.1 CompuP2P: architecture for internet computing using Peer-to-Peer Networks.

The research reported in this paper was funded in party Jerry R. Junkins Endowment at Iowa State University. It is published in IEEE Transactions on Parallel and Distributed Systems, Vol. 17, No.11. The authors are Rohit Gupta,Varun Sekhri and Arun K. Somani.

Idea of extraction: Internet computing is emerging as an important new distributed computing paradigm in which resource intensive computing is integrated over Internet-scale networks. Over these large networks, different users and organizations share their computing resources, and computations takes place in distributed fashion. In such an environment, a framework is needed in which the resource providers are given incentives to share their resources.CompuP2P is a lightweight architecture for enabling Internet computing. It uses peer-to-peer networks for sharing computing resources such as processing powers, memory storage, disk space, etc. in a completely distributed, scalable, and fault-tolerant manner. This paper discusses the system architecture, functionality, and applications of the proposed CompuP2P architecture.

2.2 CLOSER: A Collaborative Locality-Aware Overlay Service.

It is published in IEEE Transactions on Parallel and Distributed System, Vol.23, No.6. The authors are Marco Papa Manzillo, Luigi Ciminiera, Member, IEEE, Guido Marchetto, Member, IEEE, and Fulvio Risso.

Idea of extraction: Current Peer-to-Peer (P2P) file sharing system make use of considerable percentage of Internet Service (CLOSER),an architecture that aims at lessening the usage of expensive international links by exploiting traffic locality (i.e., a resource is downloaded from inside of the ISP whenever possible). The paper proves the effectiveness of CLOSER by analysis and simulation, also comparing the architecture with existing solution for traffic locality in P2P systems. While saving on international links can be attractive for ISPs,it is necessary to offer some features that can be interest for users to favor a wide adoption of the application.

2.3 Node.js: Using JavaScript to Build High-Performance Network Programs

It is published by the IEEE Computer Society IEEE Internet Computing. The authors are Stefan Tilkov and Steve Vinoski.

Idea of extraction: Node.js also called Node is a server-side JavaScript environment (see http:// nodejs.org). It is based on C++, focusing on performance and low memory consumptions But, whereas V8 supports mainly JavaScript in the browser (most notably, Google Chrome), Node aims to support long-running server processes. Node is one of the better known frameworks and environments that support server-side JavaScript development. The

community has created a whole ecosystem of libraries for, or compatible with, Node. Among these, tools such as node-MySQL or node-couch dB play an important role by supporting asynchronous interaction with relational and NoSQL data stores, respectively. The Node package manager.npm enables installation of libraries and their dependencies. Finally, many libraries available for client side JavaScript that were written to comply with the CommonJS module system also work with Node.

2.4 Design of Chatting Application Based on Android Bluetooth

This paper was published in International Journal of Computer Science and Mobile Computing.ISSN 2320–088X Vol. 3, Issue. 3, March 2014, pg.712 – 717. The authors are Nikita Mahajan, Garima Verma, Gayatri Erale, Sneha Bonde, Divya Arya.

Idea of extraction: Bluetooth provides the communication on low-power low-cost basis. Wireless communication can also be done with the help of Bluetooth Technology in a Mobile communication. Short-range establishment of two way communication has occurred without any support of the networks. Bluetooth is integrated into Android which is mainstream Smartphone platform as a mean of Mobile communication. Nowadays android becomes a latest technology in a Smartphone's which provides the open sourcing and powerful application API. Thus we design a chatting application based on Android Bluetooth which established a connection between smartphones using Bluetooth and send message are exchanged between them.

2.5 Instant Messaging Service on Android Smartphones and Personal Computers.

This paper was published in International Journal of Computer Science and Mobile Computing. ISSN 0974-2239 Volume 4, Number 3 (2014), pp. 265-272. The authors are Priya Mehrotra, Tanshi Pradhan and Payal Jain.

Idea of extraction: The main objective of this paper is introduced a methodology to provide instant Messaging Service over the Intranet which is addressed to android based smart phones and tablet user connected over the intranet via Wi-Fi. The proposed method is based on Sending/receiving message in intranet server via Wi-Fi connection without the need of taking in service from mobile service provider and without the use of internet connection. In this paper we shall be discussing the pros and cons Blue Stacks App Player which has been designed to enable Android application to run on Windows PC. We will show or discuss this by using BlueStakcs software which will provide and efficient and fast way to perform instant messaging which will further increase the performance. With IM, you can keep a list of people you interact with. You can IM with anyone on your buddy list or contact list as long as the person is online. You type messages to each other into a small window that show up on both of your screens.

III. COMPUP2P: SYSTEM OVERVIEW

CompuP2P is designed to take into account users selfishment, and use ideas game theory and microeconomics for pricing of computing resources. CompuP2P allows users to define their policies regarding what, when, how and by whom their resources can be used. Moreover it can allow users to specify their task requirement while accessing the system resources. For example, a user can specify the timeliness and reliabilityrequirements regarding the received results. Figs 1 depict the layer constituting the CompuP2P architecture. The functionality of this layer are explained in the following sections. Fig. 1. depicts the conceptual view of the operation of CompuP2P system.

3.1 Computing Resource Layer:

This layer refers to various distributed resources, such as compute power, disk space and filesetc. that exist in large Internet-scale system. This resource belongs to different nodes that are part of the underlying P2P networks. In our prototype implementation of CompuP2P, nodes joining the system are organized in a chord ring. Although, we have used Chord as the underlying P2P protocol, the architecture of CompuP2P is generalized enough to be built on top of other structured P2P networked, such as CAN.

3.2 Resource Trading Layer:

As shown in Fig. 1, the functionality of this layer can be further divided into three sub layers:

- Markets lookup protocol: it ensure that seller a buyers looking to trade a commodity converge on the same market.
- Resource pricing protocol: The pricing mechanisms ensure that both seller and MOs are suitably compensated for the service they provide to client.
- Dynamic market creation protocol: It is used for selecting nodes that act as MOs for specific commodities. The protocol is robust against MOs failing and new nodes joining the system.

3.3 Service Layer:

The service layer accepts service requests from a user. A service can be a computation task or a data storage request. A computation task is submitted by a user in the form of an XML task file. The task file is parsed and appropriate computing nodes in the network are determined that can execute the associated subtasks. The service layer allows a user to specify reliability and timeliness requirement on the result of computation, while accessing the system resources. Moreover, a user interested in backing up the local data can also request the service layer to search for appropriate storage nodes in the networks. The data is usually replicated at multiple remote nodes and is stored in either plain-text or encrypted format.

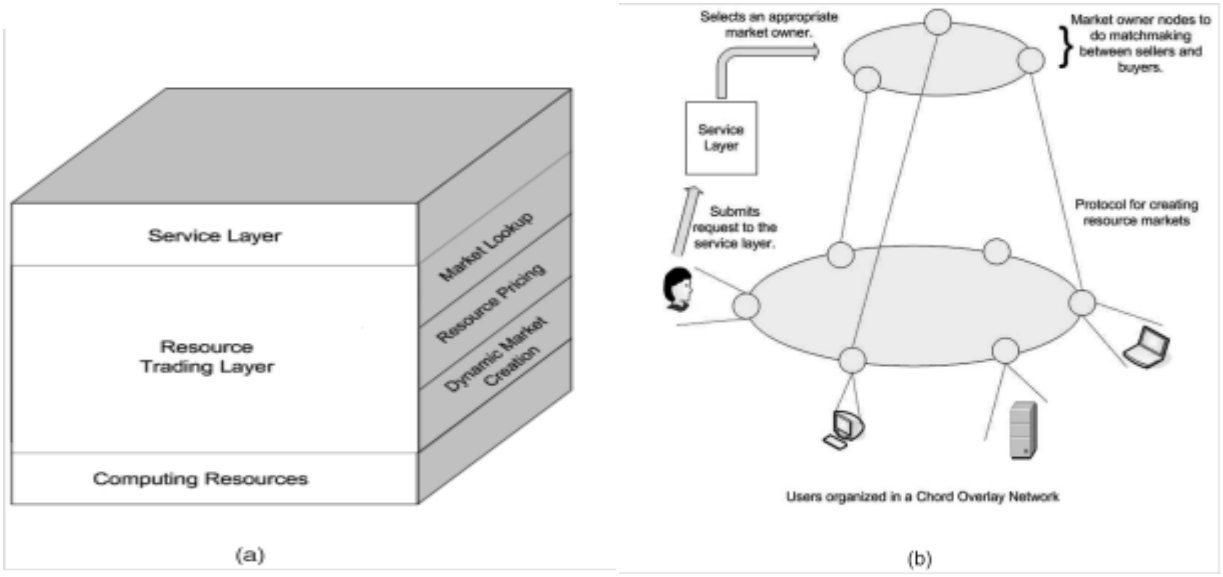

Fig. 1. (a) CompuP2P: system architecture. (b) A conceptual view of CompuP2P.

3.4 CompuP2P Usage Steps:

To demonstrate the working of the overall system, we briefly describe the step taken by a user to carry out a distributed computation.

1. The user submits an XML formatted task file to the service layer. The task file specifies for each subtask various attribute, such as input parameters, CPU cycles required, maximum offered price for successful execution, etc.
2. The service layer parse the tasks file and query the resource trading layer for the appropriated seller nodes.
3. The resource trading layer looks up the markets(s) that trade in resource required for the successful completion of the subtasks(s).
4. The MOs node is queried about the available sellers. The MO acts as a matchmaker for the finding the best seller, for example, the seller that offers the needed computer power at minimum cost. The MO is compensated on the marginal cost of the sellers listed in the markets.
5. The resource trading layer returns the information, like the IP address of a selected seller, to the service layer.
6. The service layer submits the subtask to the seller. Depending on a user's requirement, the service layer also provides various fault-tolerance features such as check pointing and replicated computing. The seller node is compensated after the computation is completed.
7. Finally the user is notified of the execution result with the outputs(s) being stored in appropriate file(s), as specified by the user in the task file for the subtask.

3.5. Peer to Peer Architecture:

Peer-to-peer networking is a network architecture which allows a group of nodes (peers) to connect with each other and share resources, and any node can operate as either a 12 server or a client. Hence participants in a P2P networks do not need a central server to communicate like the traditional client-server architecture which has existed for many years. Unlike client-server architecture, a P2P networks is considered alive even if only one peer is active. The network is unavailable only when no peers are active. Nowadays the most popular P2P networks file sharing system such as Napster, Ares, LimeWire, and Gnutella use decentralized topology, Instant messaging (ICQ) and distributed computing. Though peers all have equal status in the network, they do not all necessarily have equal physical capabilities. A P2P network might consist of pees with varying capabilities from mobile devices to mainframes. A mobile peer might not be able to act as a server due to its intrinsic limitations. Peers in P2P network have equal chance in such a way that any peer can act as a server or a client at the same time.

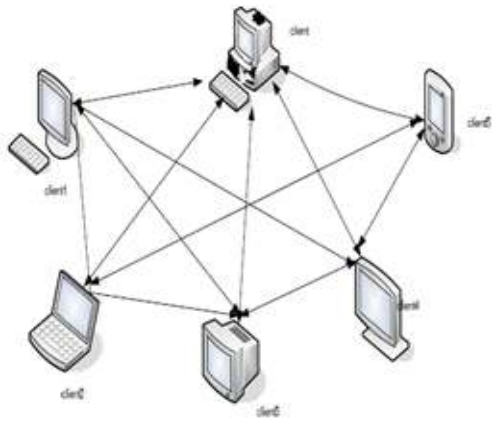

Fig. 2: Peer-to-Peer architecture

3.6. Client Server Architecture:

Client-server architecture is the oldest technology where a client machine contacts the server when the services are needed. In other words it is called centralized architecture where the whole network depends on the central point. If the central point fails, the entire system will collapse. With no server the network would make no sense.

The procedure is as follows:

- Client sends a request for a service to a server.
- The server receives the request and processes the request, and then sends back the response to the client.
- The client receives the response.
  Some of the servers existing on the Internet are web servers, mail server, FTP and so on. The communication between the server and the client.

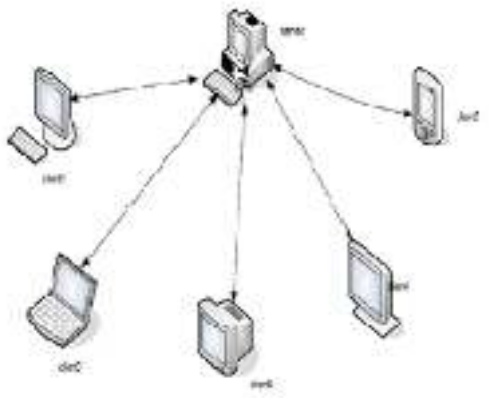

Fig. 3: Client Server Architecture

IV. RESOURCE TRADING

At the heart of the CompuP2P is its resource trading layer. This layer is responsible for creation and management of markets. For strengthening, here we use compute power as the resource under consideration. However, the proposed mechanisms for resource trading are equally applicable to other resource types, such as disk space, memory, bandwidth, etc. Each node based on its past load and current estimates the average number of resources that would remain idle in future, for example, on UNIX platform a user can use commands “top” and “uptime”. Suppose a node determines markets to permit buyers and sellers to come together and trade the amount of compute power that is needed by them. For a buyer to sequentially search the total network for the best available deal is a very time consuming and costly operations. Also, selecting one node, say the successor of ChordID zero, for trading all available compute power in the network is a bad idea. This is due to relying on one node can lead to extreme scalability, fault-tolerance and security problems. For efficient creation and lookup of compute power markets, we propose two schemes that attempt to uniformly distribute the location of and responsibility for maintaining those markets across the network. Both the schemes use Chord for market assignment and lookup, however, they differ from each other in the overhead involved and the manner in which nodes are selected for running markets for various commodities. The term commodity as used here represents a range of idle CPU cycles/sec values. Each market deals in only one type of commodity (i.e. homogeneous market). A single physical node may be responsible for more than one market. Figure depicts how nodes with different values of idle compute power C join different markets. Although, for simplicity of discussion, we have used C as discrete value, in actual

practice it refers to a well-defined range of values within which a node's idle processing capacity can lie. Thus, nodes with different but close enough idle processing capacities trade in the market.

V. CLOSER OVERVIEW

Table 1 summarizes the modifications needed by current P2p systems to be complaint with the analyzed locality-aware techniques, including CLOSER. It is worth noticing that, although two separated columns are shown in the table for the modifications required by the P2P application and the indexing system, these two components intersects when the indexing system is distributed as the indexing system is built and maintained by the application itself. This is the case for the majority of modern P2P systems which tend to migrate to decentralized approaches. In CLOSER, the localization information has to be stored at the indexing system together with itself the resource. However, the little information makes this to result in a negligible cost if we consider the storage capabilities of modern computer architectures.

TABLE 1
Summary of Modifications Needed by Locality-Aware Systems

System	ISP Support	Modified P2P Application	Modified Indexing System
oracle	Required	Required	No
P4P	Required	No	Required
Ono	No	Required	No
Kontiki	No	Required	No
CLOSER	Optional	Required	Required

TABLE 2
Model Notation

Symbol	Meaning
N	# of resources in the P2P system
M	# of resources downloaded
$f(i)$	Probability that a user requests a resource of popularity rank i
$\text{size}(i)$	Size of resource of poularity rank i
P_{ISP_j}	Prob. user belongs to the ISP_j
P	# of users in the P2P system
Ω	Average # of shared resources per user
L	# of results obtainable by a real indexing system

Given the notation described in Table 2, the percentage traffic reduction on inter-ISP links offered by CLOSER with respect to legacy systems can be obtained by

$$G_{C/L}\% = (1 - R_{C/L}) \cdot 100, \qquad (1)$$

where

$$R_{C/L} = \frac{\sum_{i=1}^{N} s(i) \cdot f(i) \cdot (1 - P_{\text{ISP}_j})^{f(i) \cdot P \cdot \Omega} \cdot \text{size}(i)}{\sum_{i=1}^{N} s(i) \cdot f(i) \cdot (1 - P_{\text{ISP}_j}) \cdot \text{size}(i)}.$$

CLOSER/ELA reduction. Similarly to the previous case, we have

$$G_{C/E}\% = (1 - R_{C/E}) \cdot 100, \qquad (2)$$

where

$$R_{C/E} = \frac{\sum_{i=1}^{N} s(i) \cdot f(i) \cdot (1 - P_{\text{ISP}_j})^{f(i) \cdot P \cdot \Omega} \cdot \text{size}(i)}{\sum_{i=1}^{N} s(i) \cdot f(i) \cdot (1 - P_{\text{ISP}_j})^{L_R(i)} \cdot \text{size}(i)}.$$

VI. CONCLUSION

In this paper, we have discussed CompuP2P, which enables Internet computing. CompuP2P is significantly different from other from other large-scale distributed computing projects which have been implemented in the arena of grid or public resource sharing computing. Users of CompuP2P can harness almost unlimited processing capacity of the entire network in a complete disturbed manner without using any centralized administrative authority. As part of our future work, we would design and implement a workflow engine and integrate it with CompuP2P. This paper presents the Collaborative Locality-aware Overlay Service, an architecture that aims at lessening the usage of expensive international links in P2P file-sharing systems. This is obtained by exploiting traffic locality (i.e. a resource is downloaded from the inside of the ISP whenever possible) and generates significant cost savings for ISPs. Analytical and simulation results show the effectiveness of CLOSER, also with respect to other proposed techniques for traffic locality in P2P systems. Unlike other approaches, CLOSER can discriminate among all complete list over the network, thus also the sampled list problem. This is obtained without transferring the complete list over the network, thus also preserving the scalability of the system.

REFERENCES

[1] Rohit Gupta,Varun Sekhri, and Arun K. Somani, "CompuP2P: An architecture for internet computing using Peer-to-Peer Networks",*IEEE Transactions On Parallel And Distributed Systems, Vol. 17, No. 11, November 2006*.

[2] Marco Papa Manzillo, Luigi Ciminiera, Member, IEEE, Guido Marchetto, Member, IEEE, and Fulvio Risso," CLOSER: A Collaborative Locality-Aware Overlay SERvice",*IEEE Transactions On Parallel And Distributed Systems, Vol. 23, No. 6, June 2012.*

[3] Stefan Tilkov and Steve Vinoski," Node.js: Using JavaScript to Build High-Performance Network Programs",*IEEE Computer Society in IEEE Internet Computing*.

[4] Nikita Mahajan, Garima Verma, Gayatri Erale, Sneha Bonde, Divya Arya," Design of Chatting Application Based on Android Bluetooth",*International Journal of Computer Science and Mobile Computing, Vol.3 Issue.3, March- 2014, pg. 712-717.*

[5] Priya Mehrotra, Tanshi Pradhan and Payal Jain," Instant Messaging Service on Android Smartphones and Personal Computers",*International Journal of Information and Computation Technology.ISSN 0974-2239 Volume 4, Number 3 (2014), pp. 265-272*.

[6] B. Liu, Y. Cui, Y. Lu, and Y. Xue, "Locality-Awareness in Bittorrent-Like P2P Applications," *IEEE Trans. Multimedia, vol. 11, no. 3, pp. 361-371, Apr. 2009*.

[7] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications." *Proc. 2001 ACM SIGCOMM Conf., 2001*.

[8] Cherry, S.M "Talk is cheap; text is cheaper [mobile messaging]", *Spectrum, IEEE, vol.39, no.5, pp.55, May 2002*.