International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.2, No.1, January 2013 DOI:10.15693/ijaist/2013.v2i1.17-20

Implementation of File Sharing in Unstructured Peer to Peer Networks

P.Vignesh,
Post Graduate Student
Department of Computer and Communication Engineering,
Sethu Institute of Technology
Virudhunagar – 626115, Tamilnadu, India

K.Priyadharshini,
Assistant Professor,
Department of Computer Science Engineering,
Sethu Institute of Technology
Virudhunagar – 626115, Tamilnadu, India

***Abstract*- Peer-to-peer (P2P) streaming has been widely deployed over the Internet. A streaming system usually has multiple channels, and peers may form multiple groups for content distribution. In this project, we propose a distributed overlay framework (called SMesh) for dynamic groups to make content distribution with switched timing fashion. P2P overlay network, hosts are responsible for packets replication and forwarding. A P2P network only uses unicast and does not need multicast capable routers. The existing networking infrastructure are multicast capable. Emerging commercial video transport and distribution networks heavily make use of IP multicasting. However, there are many operational issues that limit the use of IP multicasting into individual autonomous networks. Furthermore, only trusted hosts are allowed to be multicast sources. Thus, while it is highly efficient, IP multicasting is still not an option for P2P streaming at the user level. In this project, we use SMesh (Stable Mesh) first builds a relatively stable mesh consisting of all hosts for control messaging. The mesh supports dynamic host joining and leaving, and will guide the construction of delivery trees. In this project, we consider building a data delivery tree for each group. To reduce tree construction and maintenance costs, we build a single shared overlay mesh. The mesh is formed by all peers in the system and is, hence, independent of joining and leaving events in any group. This relatively stable mesh is used for control messaging and guiding the construction of overlay trees. With the help of the mesh, trees can be efficiently constructed with no need of loop detection and elimination. Since an overlay tree serves only a subset of peers in the network, we term this framework Subset-Mesh, or SMesh.**

Keywords- Unstructured P2P networks, Cluster.

I. INTRODUCTION

A computer network, often simply referred to as a network, is a collection of computers and devices interconnected by communication channels that facilitate communications among users and allows users to share resources. Networks may be classified according to a wide variety of characteristics. A computer network allows sharing of resources and information among interconnected devices.

Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

Peers make a portion of their resources, such as processing power, disk storage or network bandwidth directly available to other network participants, without the need for central coordination by servers or stable hosts. Peers are both suppliers and consumers of resources, in contrast to the traditional client–server model where only servers supply, and clients consume.

The peer-to-peer application structure was popularized by file sharing systems like Napster. The concept has inspired new structures and philosophies in many areas of human interaction. Peer-to-peer networking is not restricted to technology, but covers also social processes with a peer-to-peer dynamic. In such context, social peer-to-peer processes are currently emerging throughout society.

In computer science, a thread of execution is the smallest unit of processing that can be scheduled by an operating system. It generally results from a fork of a computer program into two or more concurrently running tasks. The implementation of threads and processes differs from one operating system to another, but in most cases, a thread is contained inside a process. Multiple threads can exist within the same process and share resources such as memory, while different processes do not share these resources. In particular, the thread of a process share the latter's instructions and its context. To give an analogy, multiple threads in a process are like multiple cooks reading off the same cook book and following its instructions, not necessarily from the same page.

On a single processor, multithreading generally occurs by time-division multiplexing the processor switches between different threads. This context switching generally happens frequently enough that the user perceives the threads or tasks as running at the same time. On a multiprocessor or multi-core system, the threads or tasks will actually run at the same time, with each processor or core running a particular thread or task.

File sharing is the practice of distributing or providing access to digitally stored information, such as computer programs, multimedia (audio, images, & video), documents, or electronic books. It may be implemented through a variety of ways. Storage, transmission, and distribution models are common methods of file sharing that incorporate manual sharing using removable media, centralized computer file server installations on computer networks, World Wide Web-based hyperlinked documents, and the use of distributed peer-to-peer networking.

Users can use software that connects in to a peer-to-peer network to search for shared files on the computers of other users connected to the network. Files of interest can then be downloaded directly from other users on the network. Typically, large files are broken down into smaller chunks, which may be obtained from multiple peers and then reassembled by the downloader. This is done while the peer is simultaneously uploading the chunks it already has to other peers.

II.EXISTING METHODS

E.Cohen et.al (2003) [4] developed a new class of decentralized P2P architectures. Their design is based on unstructured architectures such as gnutella and fast track, and retains many of their appealing properties including support for partial match queries, and relative resilience to peer failures. Yet, they obtain orders of magnitude improvement in the efficiency of locating rare items. Their approach exploits associations inherent in human selections to steer the search process to peers that are more likely to have an answer to the query. They demonstrate the potential of associative search using models, analysis, and simulations.

K. Sripanidkulchai et.al (2003) [5] explored how to retain the simplicity of Gnutella, while addressing its inherent weakness: scalability. They proposed a content location solution in which peers loosely organize themselves into an interest-based structure on top of the existing Gnutella network. Their approach exploits a simple, yet powerful principle called interest-based locality, which posits that if a peer has a particular piece of content that one is interested in, it is very likely that it will have other items that one is interested in as well. When using their algorithm, called interest-based shortcuts, a significant amount of flooding can be avoided, making Gnutella a more competitive solution. In addition, shortcuts are modular and can be used to improve the performance of other content location mechanisms including distributed hash table schemes. They demonstrate the existence of interest-based locality in five diverse traces of content distribution applications, two of which are traces of popular peer-to-peer file-sharing applications. Simulation results show that interest-based shortcuts often resolve queries quickly in one peer-to-peer hop, while reducing the total load in the system by a factor of 3 to 7.

I. Stoica et.al (2003) [6] suggested a fundamental problem that confronts peer-to-peer applications is the efficient location of the node that stores a desired data item. This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord provides support for just one operation: given a key, it maps the key onto a node. Data location can be easily implemented on top of Chord by associating a key with each data item, and storing the key/data pair at the node to which the key maps. Chord adapts efficiently as nodes join and leave the system, and can answer queries even if the system is continuously changing. Results from theoretical analysis and simulations show that Chord is scalable: communication cost and the state maintained by each node scale logarithmically with the number of Chord nodes.

B. Yang et.al (2002) [7] presented three techniques for efficient search in P2P systems. They present the design of these techniques, and then evaluate them using a combination of experiments over Gnutella, the largest open P2P system in operation, and analysis. While their techniques maintain the same quality of results as currently used techniques, our techniques use up to 5 times fewer resources. In addition, they designed their techniques to be simple in design and implementation, so that they can be easily incorporated into existing systems for immediate impact.

A. Rowstron et.al (2001) [8] presented the design and evaluation of Pastry, a scalable, distributed object location and routing substrate for wide-area peer-to-peer applications. Pastry performs application-level routing and object location in a potentially very large overlay network of nodes connected via the Internet. It can be used to support a variety of peer-to-peer applications, including global data storage, data sharing, group communication and naming. Pastry is completely decentralized, scalable, and self-organizing; it automatically adapts to the arrival, departure and failure of nodes. Experimental results obtained with a prototype implementation on an emulated network of up to 100,000 nodes confirm Pastry's scalability and efficiency, its ability to self-organize and adapt to node failures, and its good network locality properties.

M.E.J.Newman et.al (2000) proposed the small-world network model, which mimics the transition between regular-lattice and random-lattice behavior in social networks of increasing size is studied. They contend that the model displays a critical point with a divergent characteristic length as the degree of randomness tends to zero. They proposed a real-space renormalization group transformation for the model and demonstrate that the transformation is exact in the limit of large system size. This result is used to calculate the exact value of the single critical exponent for the system, and to derive the scaling form for the average number of 'degrees of separation' between two nodes on the network as a function of the three independent variables. They confirm their results by extensive numerical simulation.

III.OVERVIEW

3.1. Description

This paper describes the peer to peer streaming by means of file sharing from the source peer to the group peer under the control of control peer. The source peer activates the group peer after the successful standard login verification. Once it is verified, the control peer stores that group peer entry in the vector table. The file is to be shared to the group peer from the source peer. Threads are responsible to upload data chunks to Peers using timing fashion method. If a group peer is in active state, then its details are stored in the report which is produced in the control peer. After the content distribution, if the group peers wish to leave the network, it leaves by intimate to the control peer. The control peer enters that group peer detail in the leave peer report.

3.2 Existing System

System uses random walks to send queries across the overlay. When a peer receives the query, it sends a list of all the content matching the query to the originating peer. There is no control of joining and leaving nodes through control peer. Each peer maintains a small routing table (neighboring peers Node IDs and IP addresses). System is composed of peers joining the network with some loose rules, without any prior knowledge of the topology. Network uses flooding as the mechanism to send queries across the overlay with a limited scope. During the lookup process, any node encountered along the path is checked for availability and can be selected as a servant for the querying user.

3.3. Proposed System

A streaming system usually has multiple channels, and peers may form multiple groups for content distribution. In this paper, we propose a distributed overlay framework (called SMesh) for dynamic groups to make content distribution with switched timing fashion. SMesh (Stable Mesh) first builds a relatively stable mesh consisting of all hosts for control messaging. The mesh supports dynamic host joining and leaving, and will guide the construction of delivery trees. In this paper, we consider building a data delivery tree for each group. To reduce tree construction and maintenance costs, we build a single shared overlay mesh. The mesh is formed by all peers in the system and is, hence, independent of joining and leaving events in any group. This relatively stable mesh is used for control messaging

and guiding the construction of overlay trees. With the help of mesh, trees can be effectively constructed without the need of loop detection and elimination.

3.4 Algorithm

The Proposed Method algorithm as follows,
1.Group peers detection at CP
2.Peer finder at source
3.Peer communicative tasks
4.If (groupfound)
Add in NodeMap
End IF
5.Distribution using switched process to the nodes of NodeMap

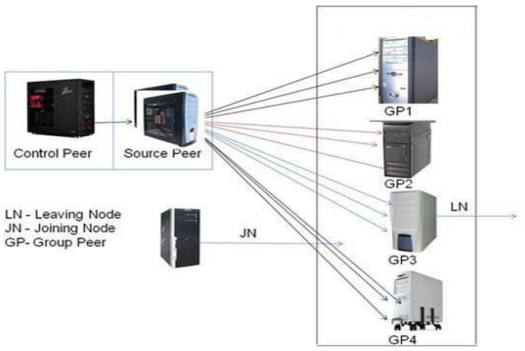


FIG 1. Proposed system design

3.5Algorithm Description

Control peer, source peer and group peer are formed. Group peers are detected at control peer. Peers are found near the source. Then peer communication tasks are performed. Check whether the particular group is foung. If the group is found add it in NodeMap. Atlast distribution using switched process to the nodes of NodeMap is done.

IV.IMPLEMENTATION AND METHODOLOGY

4.1 DATA FLOW DIAGRAM

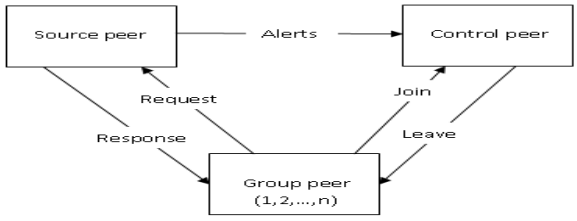


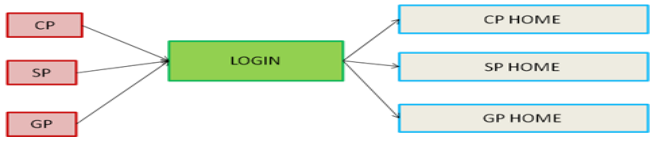
FIG 2. Peer to peer process

4.2 MODULE DESCRIPTION:

4.2.1 Process Level Validation

Control Peer, Source Peer, Group Peers are the three layer of the system. The Layers are started after the successful standard login verification. Control Peer has to do the authentication task to get the Control Peer Home, and then Source peer and Group peer get the home task after the successful validation. The default validation tasks are performed in this scenario. Project relevant information is managed in this module to describe the synopsis, developer information. Three layers are executed in three different process levels. It means the layers are executed in different machines. Control Peer gets one process, Source peer gets a process and Group Peers can be started in different process levels.

FIG 3. Process level validation in different layers.



4.2.2 Group peer messaging

Group Peer is the application level process and the process can be started in wired environment system. After Starting GPA (Group Peer Application), the GPASI (Group Peer Application System Identifier) is the process to transfer GP status (IP Address Object) to Control Peer. It is the joined status. The GPASI indicates the Control Peer about leaving the GP from the Group. Based on the GPASI the CP indicates the Source Peer to define the data transferring task. All the messaging is performed automatically without doing any manual process. The CP listens the GPASI to manage the GPV (Group Peer Vector). The GPV is the collection of GPE (Group Peer Elements). The number of elements can be varied depending upon the joining or leaving process. Finally Control Peer dynamically manages the list to assist the Source Peer.

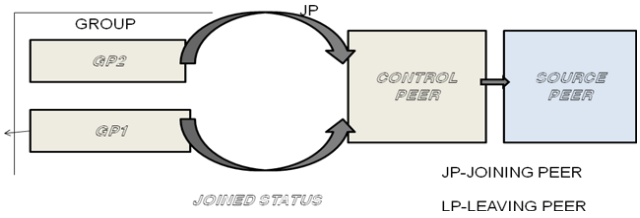


FIG 4. Group peer message formation

4.2.3 Group peer vector formation

Source Peer maintains the GPV based on Control peer messaging to define the task preparation list. The task preparation list defines the uploaded data to various systems with parallel approach. GPV is the dynamic collection maintained in SP which has the IPO (IP Object of peers). By using IPOE (IP Object Element) information the Source peer starts the uploaded process. When one peer joined or leaved the IPO is modified and tasks are remapped**.**

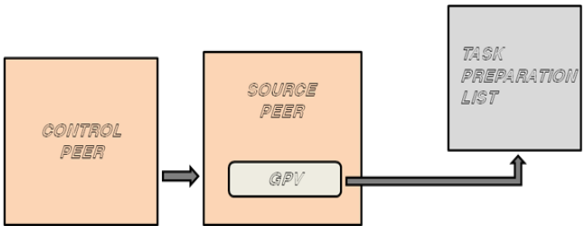


FIG 5. Source peer GPV based on control peer messaging.

4.2.4 Data transferring process

Based on the GPV (collection of IPO. IPOE is the object element indicates the peer), the Source peer starts different process to upload the data to different peers. P1, P2…Pn are peers. The T1 connects with P1, T2 connects with P2, Tn connects with Pn. T1 consists T1a, T1b, T1c. Here T1 is the thread process and T1a is the child thread of T1 and so on. T1a, T1b, Tnn are the threads responsible to upload data chunks to Peers using timing fashion method. The each thread

will be connected with peers for defined time for example 0.05 seconds to transfer the data to Group peer. After the time out next child threads starts the same process and continue the task of previous thread. If previous thread gets failure, the next thread continues the same work of previous thread. So the system assures the data loss prevention.

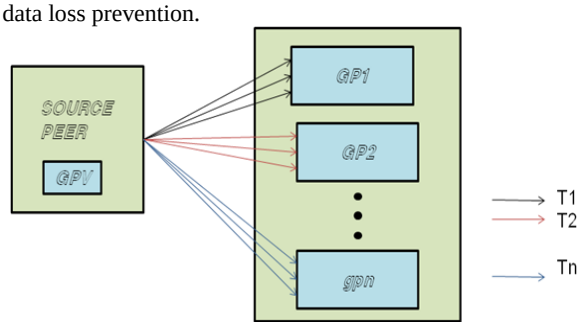


FIG 6. Parallel data transferring to different group peers

4.2.5 Peer Assembler

The assembler assembles the uploaded content from various processes. Assembler creates the container to put the various data chunks in order. To successful data assembling the assembler gets the data about uploaded data before starting the uploading task. After receiving the uploaded finished signal from the source peer, the data is assembled. Then the assembled content will be managed in group peer system.

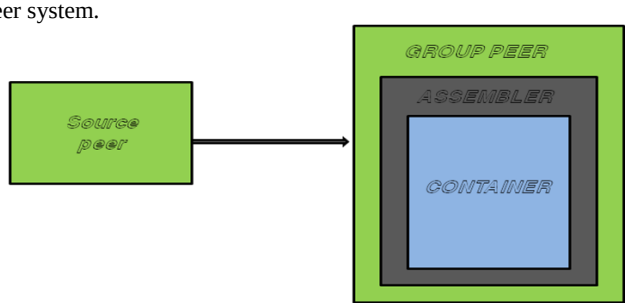


FIG 7.Peer assembler

V.RESULTS AND DISCUSSIONS

In existing system context switching is used and repeatable data are sent. But in proposed method, we use multithreading technique to transmit data and non repeatable data are sent to the selected peers.

VI.CONCLUSION AND FUTURE ENHANCEMENTS

The Group peer register before it startup. The startup and leaving processes are monitored by the control peer. The state changes of Group Peers are intimated to Source Peer. SP manages the Group Peer map and the map reflects with CP state changes. The merit of this system is that, the active nodes in the networks are dynamically updated in the vector table maintained by the control peer, without any manual updating. The RMI is used to develop the network environment and GUI is designed using Swing and AWT techniques. Oracle is the backend used to manage the persistence mechanism. Datas are sent in Multi threaded fashion (multicast to more than one group peer).

The system builds the self informative group peer cluster with source peer which is controlled by control peer. Control Peer controls the joining and leaving nodes, T and T1. Tn processes, Gp1.Gpn peer counts. Dynamic messages generated by the control peer will control the source peer reaction to the situations like starting and stopping processes. Each node must have the knowledge of neighbors. This protects the system from untrusted nodes. Resource discovery has also been overcome.

Number of source peer can be increased in a stable network. If the source peer gets failed, then automatically other source peer acts as next available server. A group peer can have many child peers. The main group peer receives the data from source peer and sends data to child peers. It increases the source peer's task environment. Control peer indicates the source peer about its system stability.

REFERENCES

[1] Zhen Zhang, "Approach to Construct Cluster in Unstructured P2P Networks Based on Small World Theory" IEEE, 2011.
[2] Gnutella development page, http://gnutella.wego. com, 2005.
[3] Kazaa home page, http://www.kazaa.com, 2005.
[4] E. Cohen, A. Fiat, and H. Kaplan, "Associative Search in Peer-to-Peer Networks: Harnessing Latent Semantics," Proc. IEEE INFOCOM, 2003.
[5] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," Proc. IEEE INFOCOM, 2003.
[6] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Trans. Networking, vol. 11, no. 1, pp. 17-32, 2003.
[7] B. Yang and H. Garcia-Monlina, "Efficient Search in Peer-to-Peer Networks," Proc. 22nd IEEE Int'l Conf. Distributed Computing Systems (ICDCS), 2002.
[8] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware), 2001.
[9] Freenet home page, http://www.freenet.sourceforge. com, 2005.
[10] D.J. Watts, S.H.Strogatz, Collective dynamics of 'small-world' networks, Nature 393(1998) 440-442.

Author Profile

Vignesh.P received his B.tech degree in Information Technology from P.S.N.A College of Engineering and Technology, India, in 2010. He is Currently pursuing his M.E. in computer and communication engineering from Sethu Institute of Technology, India. His research interests include networking, Cryptography, Data Mining and Artificial Intelligence.