International Journal of Engineering Applied Sciences and Technology, 2018
Vol. 3, Issue 7, ISSN No. 2455-2143, Pages 98-102
Published Online November 2018 in IJEAST (http://www.ijeast.com)

PEER-TO-PEER DISTRIBUTED FILE SHARING SYSTEM USING BINARY TREE TOPOLOGY

Ruchi Sodhi, Chakresh Kumar
University School of Information, Communication & Technology,
Guru Gobind Singh Indraprastha University, New Delhi-110078, India

Ghanendra Kumar
Department of Electronics and Communication Engineering
National Institute of Technology, Delhi-110040, India

Abstract— **One of the major disadvantages of a client-server network is its dependency on a central server. This dependence creates a single point of failure for the network i.e. If the central server fails, the network stops functioning. Though getting rid of the central server is not viable for some scenarios, there are some problems that could be solved more effectively with a server-less architecture. A Peer-to-Peer or adhoc network is formed when a collection of multiple nodes/computers offering a similar service come together eliminating the need of a central server. These Peer-to-Peer networks can be used for vast applications like video sharing, file sharing, distributed computation, etc.**

Here I aim at designing a Peer-to-Peer file sharing system for quick searching and download of a given file that is present over multiple nodes/computers on that network. The entire file can be taken from a single peer that hosts that file but since the upload speed is much slower than the download speed, the node downloading the file will be limited by the uploader peer's speed. To overcome this, the downloader node/peer can take parts of file from multiple uploader nodes that host the file. In this way, the downloader node/peer can attain its maximum download speed.

I am going to use binary tree as the Network topology. This binary tree topology will enable is in assigning logical addresses to the nodes which will help us in routing the file packet to the destination using an efficient path.

I. INTRODUCTION

In the most basic form, a Peer-to-Peer (P2P) network is established when two or more computers are connected with each other, and the resources are shared between the users without any need of having a server computer. P2P network can also be a permanent arrangement that can link half-dozen computers in a small workplace over copper wires, or a Peer to Peer network can be a network on a much bigger scale in which dedicated protocols and applications are used to set up an uninterrupted relationships between users over the Internet.

A. *Distributed file sharing system*

A typical implementation of a file sharing system would include a single downloader taking the entire file from a single uploader. Though this architecture is simple, it comes with its own drawbacks.

i. The transfer speed will always be limited by the upload speed of the uploader.

ii. If the uploader goes down while the transfer, the download process will fail.

Single Uploader Peer

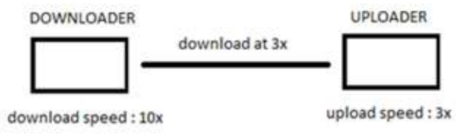


Fig. 1 single uploader peer

Multiple Uploader Peer

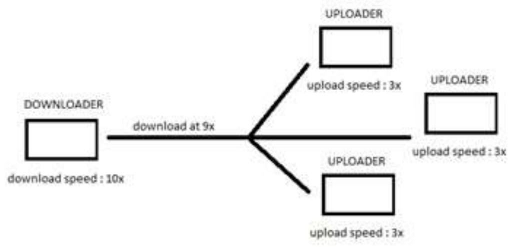


Fig.2 Multiple Uploader Peer

A distributed file sharing system helps us in overcoming these 2 disadvantages. In a distributed environment, a given file is downloaded by taking chunks of the file from multiple uploaders simultaneously. This will help us is attaining the maximum download speed and will also remove the dependency from a single uploader.

B. *Network Topology*

Network Topology

The logical and hierarchical manner in which computers are computers are connected is termed as Network topology. It defines the layout pattern of the interconnections between the computers. Network Topology is also called as "network architecture."

Devices on the network are known as 'nodes.' Network Topologies are classified on the basis of manners in which the nodes are connected with each other. Network topologies determine the way nodes communicate with each over the network. There are different types of network topologies such as bus, Point-to-point, star, mesh and ring.

1) Bus: One main cable is used in the Bus Topology. Nodes are directly connected to the cable. The main cable is the backbone of the network

2) Point-to-Point: Point-to-point topology is the most basic simplest network topology. The nodes have a direct link between them.

3) Star: Star Network Topology consists of a central hub to which each node is connected using point-to-point connection.

4) Mesh: Every node is connected to the every other node in the Mesh Network topology. This topology is not efficient in handling high volume of traffic

5) Ring: In ring topology, each node is connected to the other two nodes forming a ring like structure. Data travels node to node in one direction.

II. BINARY TREE NETWORK TOPOLOGY

Binary tree is a data-structure that is used mostly for quick lookup of data in memory. I am going to use this data structure as the topology of the network for quick lookup of a given file over the network. Having a tree like structured topology gives us the advantage of assigning addresses to the nodes and delivering packed from one uploader nodes/peers to the downloader node.

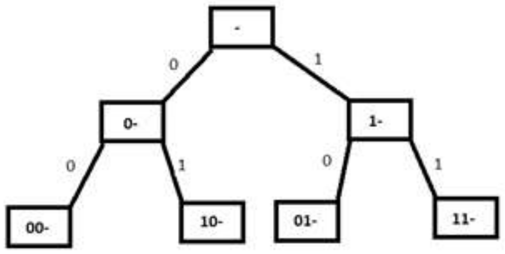


Fig. 3 Binary Tree Network Topology

Above is an example of a peer-to-peer network where different nodes come together to form a tree like structure. The root node is assigned the address “–“.

1. If a peer is added to the left of a node with id x, the address of the new node will be 0x.

2. If a peer is added to the left of a node with id x, the address of the new node will be 1x.

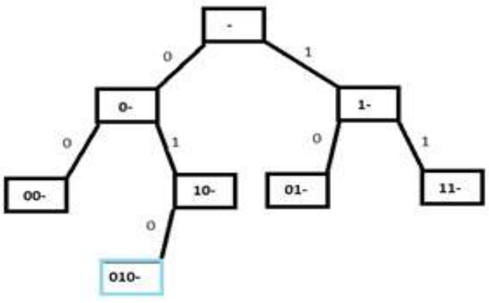


Fig. 4 Addition of new node

In the example above, since the new node(blue) is added to the left of 10-, its id is 010-

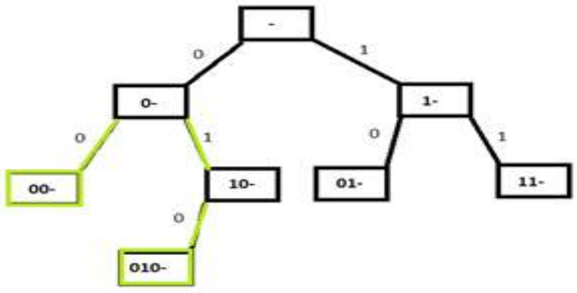


Fig. 5 Routing message from one peer to another

If (010-) wants to send message to (00-) then it will send it via (0-) which is their common parent node.

A. *Message Routing Algorithm*

010- (from)

00- (to)

We see that in (010-) and (00-), (0-) is the common part, which is the address of their common parent.

In 010- , 0- is common and 01 is different. Since length of 01 is 2, the message has to be sent to 2 levels above to reach 0-(common parent).

Now in (00-) 0- is common and 0 if different. But since the message has already reached (0-), we are only concerned about different. Since 0 represents left, the message has to be sent to the left child of 0-. That will be 00-

B. *Why Binary Tree Topology*

Binary tree give us an advantage that is not possible in many topologies. There is a hierarchical relationship between nodes. This relationship can be used to assign logical address to the nodes which can be used in routing of data packets.

C. *Socket*

A socket is bounded to a specific port number of a computer on which the program is running of a specific computer. Client then makes a connection request to the server, and server waits and listen to the socket when client makes a request. The client identifies itself to the server and is bound to a local port number which is used during the connection.

Connection request

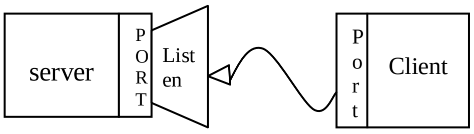


Fig. 6 Connection Request

Server then accepts the request, if everything is going as per the server' requirement. After accepting the request, the server catches a new socket destined to the same local port. The server's end point is set to the address and port of the client. It needs a new socket to continue listening to the original socket for connection requests.

Connection

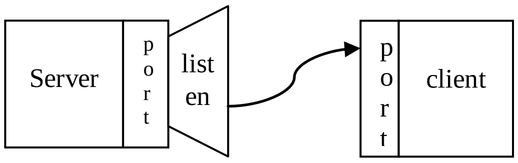


Fig. 7 Connection

If the connection is accepted on the client side, a socket is created successfully and the client then uses the socket to communicate with the server.

A communication is now build up between client and a server and they communicate by writing to and reading from their sockets. An end point is comprised of a port number and an IP address. Given these two end points, every TCP connection is uniquely identified. This gives us the liberty of having multiple connections between our host and the server.

III. IMPLEMENTATION

Technology used: JAVA Socket programming

There are different types of messages that will be transferred in our applications. These messages will be used for establishing connection, requesting for a file and sending the file.

Type of messages communicated:

- ADD_NEW_NODE
- ID_ASSIGNMENT
- ID_ASSIGNMENT_FAILURE
- FILE_REQUEST
- FILE_RESPONSE

A. *ADD_NEW_NODE*

This type of message is a request to connect to the network that is sent from a connecting/new node to a network/tree node (that is already part of the tree network). It is sent only once.

Message direction: NEW NODE -> TREE NODE

RESPONSE OF THE NODE THAT RECIEVES THE MESSAGE:

When a tree node receives this request/message it will try to add the new node as its left or right child. Once added successfully, it will send an ID_ASSIGNMENT message to the new node which means that the new node is now a part of the network. However, if It already have both right and the left child, it will send an ID_ASSIGNMENT_FAILURE message, which means the new node cannot be connected to the tree/network by requesting to this node.

MESSAGE FORMAT:

```
{
  "type": "ADD_NEW_NODE",
  "content": null
}
```

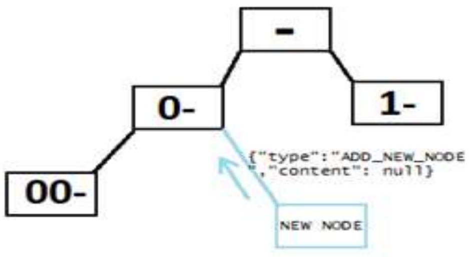


Fig. 8 Adding node

B. *ID_ASSIGNMENT*

This message is a response to the ADD_NEW_NODE message. When a node succeeds adding a new node as its left or right child, it sends this message to the new node. The content of this message is the Logical id of the new node that is being connected to the network and its value is created by adding 0 or 1 in front of the id of the existing node.

RESPONSE OF THE NODE THAT RECIEVES THE MESSAGE:
When the new node receives this message, it marks the node that sent this message as its parent.

MESSAGE FORMAT:

```
{
  "type": "ID_ASSIGNMENT",
  "content": "01-"
}
```

So, 1- adds the new node as its left child an sends the above message to that node. The new node now knows that its id is 01- from the content.

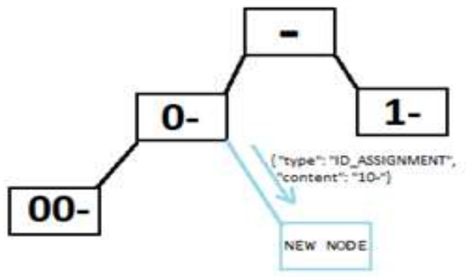


Fig. 9 Assigning ID

C. ID_ASSIGNMENT_FAILURE

This message is a response to the ADD_NEW_NODE message. When an existing node fails to add a new node as its left or right child (it already had a left/right children), it sends this message to the new node. The content of this message is null

RESPONSE OF THE NODE THAT RECIEVES THE MESSAGE:

Failed to connect to the network.

MESSAGE FORMAT:
```
{
  "type": "ID_ASSIGNMENT_FAILURE",
  "content": null
}
```

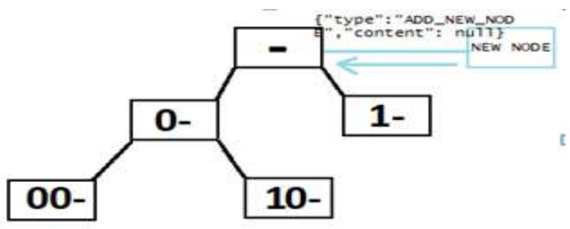


Fig. 10 ID assignment failure

D. FILE_REQUEST

Whenever a node wants a specific file, it will broadcast this type of message. Since a node only knows about 3 adjacent nodes(left child, right child and parent), it will send this message to them. The message content will contain the ID of the original node that requests that file and the file name.

RESPONSE OF THE NODE THAT RECIEVES THE MESSAGE:

Case 1: File is present with the node that receives the request:
Whenever a node receives file request message, it will 1st check if it has that file. If it is present, it will send the FILE_RESPONSE message to the requesting node(The file request message contains the address).

Case 2: File is not present with the node that receives the request:
If the file is not present, the node will just forward the message to its other 2 adjacent link. For example, if a node receives the request from its left child and it does not have that file, it will forward the request to its parent and its right child.

MESSAGE FORMAT:

```
{
  "type": "FILE_REQUEST",
  "content": “requesting_node:0-,file_name:photo.jpg”
}
```

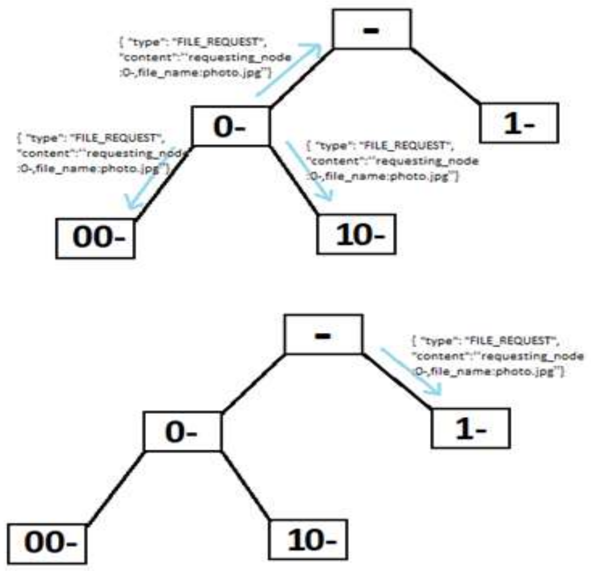

{"type": "FILE_REQUEST", "content":"requesting_node:0-,file_name:photo.jpg"}
-
0-
1-
00-
10-

Fig. 11 File Request

E. FILE_RESPONSE

Whenever a node receives a request for a file that it has, it will create a response message and send it on the appropriate path. The content of the message will be the destination address (which is the same as the requesting node in the FILE_REQUEST), the data of the file and file’s full name.

RESPONSE OF THE NODE THAT RECIEVES THE MESSAGE:

Case 1: The Destination address is the same as the receiving node address:

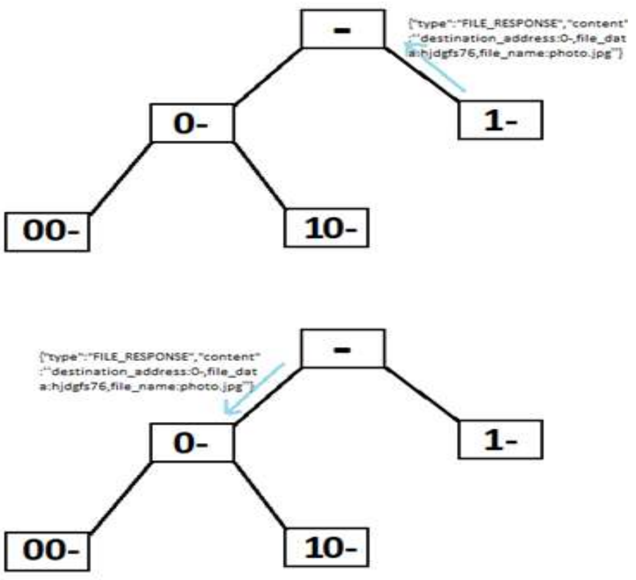
If the destination address in the message is the same as the address of the receiving node, it will save the file.

Case 2: The Destination address is different from the receiving node address:

If the destination address in the message is different from the address of the receiving node, it will forward the message to its appropriate adjacent node. This can be found by comparing the destination id with the receiving node's id. It will be done using the following algorithm:
We see that in (010-) and (00-), (0-) is the common part, which is the address of their common parent.
In 010- , 0- is common and 01 is different. Since length of 01 is 2, the message has to be sent to 2 levels above to reach 0-(common parent).
Now in (00-) 0- is common and 0 if different. But since the message has already reached (0-), we are only concerned about different. Since 0 represents left, the message has to be sent to the left child of 0-. That will be 00-.

MESSAGE FORMAT:

```
{
  "type": "FILE_RESPONSE",
  "content":                    "destination_address:0-
,file_data:"hjdgfs76",file_name:photo.jpg"
}
```



```
{"type":"FILE_RESPONSE","content":"destination_address:0-,file_data:hjdgfs76,file_name:photo.jpg"}
```

```
{"type":"FILE_RESPONSE","content":"destination_address:0-,file_data:hjdgfs76,file_name:photo.jpg"}
```

Fig. 12 File Response

IV. CONCLUSION

The major disadvantages of a client-server network are its dependency on a central server. This dependence creates a single point of failure for the network i.e. If the central server fails, the network stops functioning. Though getting rid of the central server is not viable for some scenarios, there are some problems that could be solved more effectively with a server-less architecture. In this paper I have successfully assigned address to each peer which is acting as a node of the binary tree, for the quick look up of the file and transferring the file from one peer to another.

V. FUTURE SCOPE

- Data from multiple source

Currently, data/file is taken from a single scope. The solution need to be enhanced to read file from multiple sources.

- Live Video streaming

The distributed binary tree solution can be used to get video data from more than one source for video playing. This will resolve the buffer issue due to slow download.

- Fault tolerance

The solution needs to be improved for fault tolerance. In case a node goes down, the system should be capable to handle such disaster.

- ID_ASSIGNMENT_FAILURE improvement

Currently, if can connect to the network only by requesting those nodes which don't have a child. This should not be the case. The network should be smart enough to add the node to the apt location.

- Use routing tables

The performance can hugely improve if we can use routing tables. With this the transfer complexity will be even less than log(n).

- Balanced binary tree

If we can manage to make the network a self -balanced binary tree, the complexity of file transfer will always be O(logn).

VI. REFERENCES

[1] Sodagar I., (2011) "The mpeg-dash standard for multimedia streaming over the internet," IEEE multimedia, vol. 18, no. 4, (pp. 62–67)

[2] Lua E.K., Crowcroft J., Pias M., Sharma R., Lim S. (2005) A Survey and Comparison of Peer to-Peer Overlay Network Schemes. IEEE Communications Surveys and Tutorials, Second Quarter, 7(2).

[3] Detti A., Pomposini M., Blefari-Melazzi N., Salsano S., and Bragagnini A., (2012) "Offloading cellular networks with information-centric networking: The case of video streaming," in 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), (pp. 1–3).

[4] Daswani N., Molina H.G., and yang B., (2003) "Open Problems in Data-sharing Peer-to-peer Systems," 9th International Conference on Database Theory.

[5] Harami C., Gazdar A., Jamelli I., and Belgith A., (2015) "Study of VOD streaming on BitTorrent," IEEE International Symposium on Networks, Computers and Communications (ISNCC).