Author et al. 2019, 7:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Understanding the Flash Crowd in P2p Live Video Streaming Systems

M.Tech. Scholar Mahazbeen Khan[1], Prof. Rahul Pandey[2]
Department of Electronics& Communication Engineering
School of Research & Technology
People's University, Bhopal (MP), India

Abstract- In this research, a distributed algorithm with minimal central control is presented which organizes the newly arrived peers into hierarchical positions to reduce competition among them. This hierarchical rank is then used to construct different sub-stream trees. The video stream is divided into sub-streams and each sub-stream is pushed over a separate sub-stream tree. Only the peers at the top of the hierarchy will directly access the scares initial resources and in turn forward the stream to those below them in the hierarchy. Thus, by utilizing the resources provided by the newly arrived peers in handling flash crowd, better system scale can be achieved.

Keywords- P2P Network, Content Delivery Networks (CDN), Distributed Hash Table (DHT).

I. INTRODUCTION

Client-server architecture has been the modus operandi of the internet for a long time. In such systems, there is a clear distinction between the role of client and server nodes. The client node is the requester of the resource (which may be a webpage, music file, video stream or even a service) and the server node is responsible for processing these requests and replying with the desired resources. All the processing is done at the server.

This architecture worked well as long as the capabilities of client and server nodes were of different orders of magnitude. But improvement in processing capabilities of local machines and the bandwidth capability of internet reduced this difference significantly. The client nodes, equipped with powerful processors and faster internet, became capable of evolving from passive recipient to an active participant in the system. Traditional client-server architecture was unable to tap in to this potential. Moreover, as the number of devices connected to the internet increased, the client-server architecture suffered from scalability and single point of failure issues. Hence, peer-to-peer architecture was developed as an alternative.

1. DHT Overview

A Distributed Hash Table (DHT) based protocol is used to provide the overlay over which the streaming trees can be constructed. In DHT based systems, each resource is identified by a <key,value> pair. The set of all possible keys forms the key space. For e.g., if each key is a m-bit long sequence, the keyspace would comprise of all possible 2m combinations.

The keyspace is partitioned among the nodes of the network. Each node is responsible for maintaining the resources whose key lies in its region of the keyspace. Finally, a routing algorithm describes how the queries are routed in the network to locate the resources. Different implementations give rise to different systems. Chord [10] is the preferred choice for this thesis but other DHT based protocols such as Tapestry [12] or Pastry [11] can also be used.

2. CHORD

Chord is a scalable peer-to-peer look up protocol designed to construct an overlay network based on DHT. As described earlier, each node as well as resource in the system is identified by a m-bit long node-id and resource-id (key) respectively, with each node maintaining a subset of the resources. By constructing and maintaining efficient routing structures, a resource can be located in a maximum of

$O(\log_2(N))$ hops, where N is the number of nodes in the Chord overlay.The mechanism for mapping resources to nodes and routing queries over the network. On bootstrapping, each node contacts the bootstrapping server and receives a list of existing nodes. It then connects itself in the chord overlay and starts to populate its routing tables by exchanging information with other nodes. Chord supports multiple joins and can bootstrap large- scale DHT networks.

II .CONSTRUCTION OF STREAMING TREES

In this case, the parent Node accepts child n Node as its child and the process is completed for child Node. If the number of fertile children of parent Node becomes k, it sends a deregistration message to #(stream[s] level[l-1]) to remove itself from the feed-forwarder list.

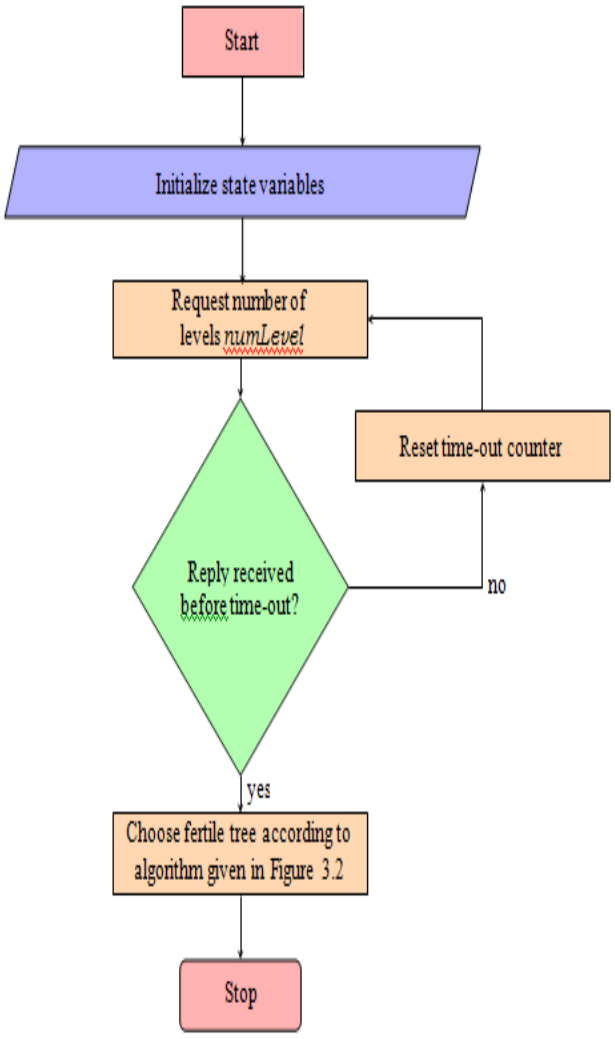

Figure 1 Flow chart for selecting fertile tree.

III. RESULTS

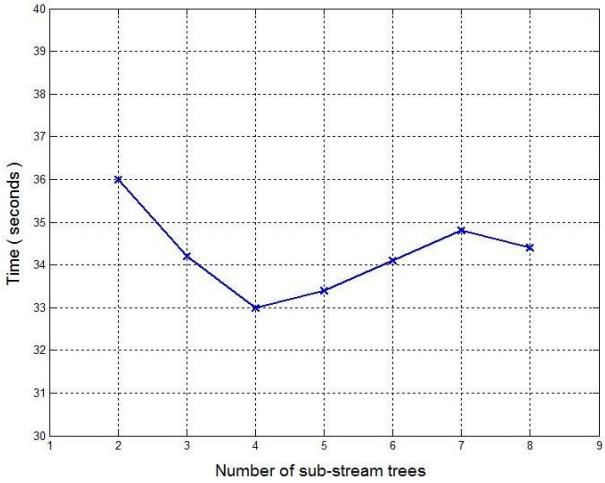

Figure 2 Time taken for 90% of the nodes to get all the sub-streams (stabilization time) for different values of t.

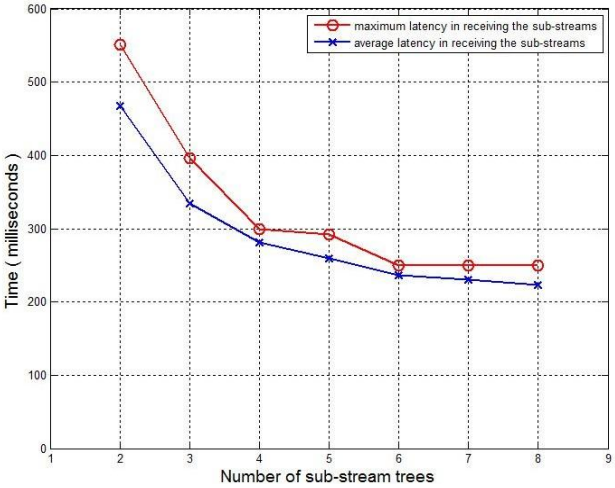

Figure 3 Average and maximum value of average latency at each node for different values of t

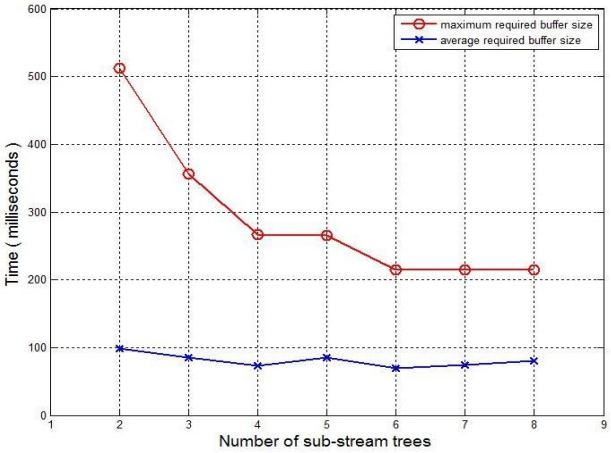

Figure 4 Average and maximum buffer size required at each node.

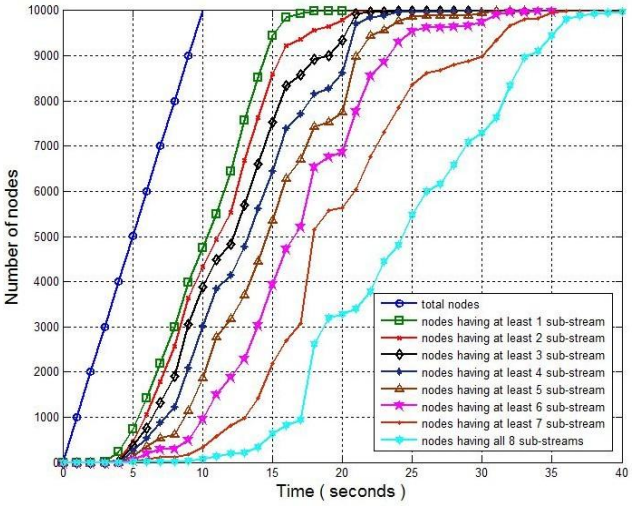

Figure 5 Number of sub-streams received by nodes at different times.

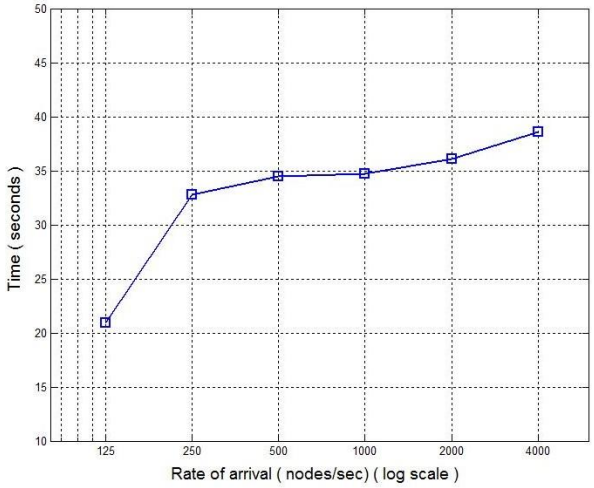

Figure 6 System stabilization time for different node arrival rates.

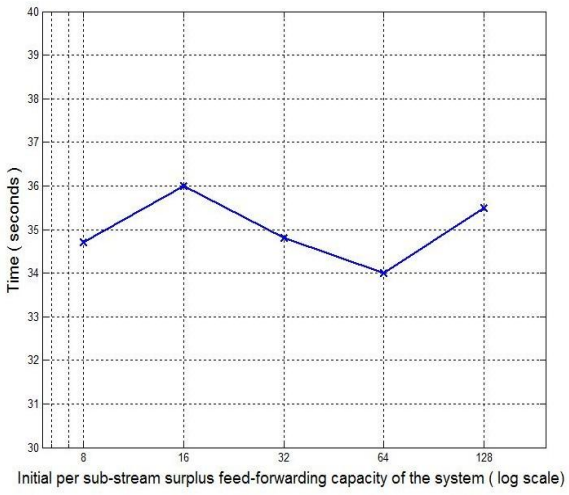

Figure 7 Stabilization time of the system for different values of U_s.

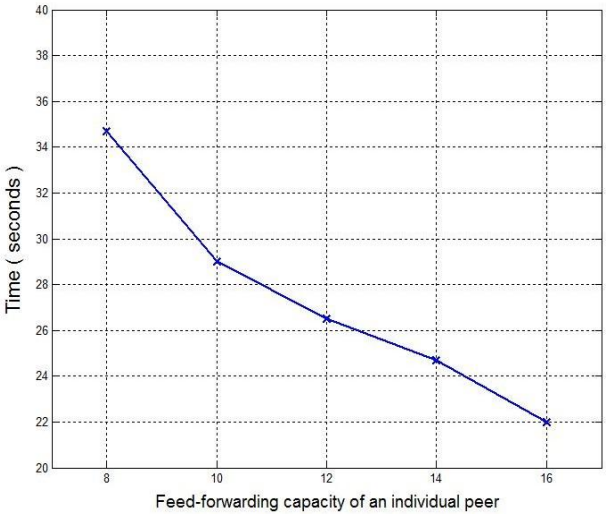

Figure 8 Stabilization time of the system for different values of k.

IV.CONCLUSION

The proposed method consists of two major steps (1) Arrange the newly arrived peers in different levels of different sub-stream trees. (2) Connect the peers with each other to distribute the stream. The number of levels that are constructed depends upon the node arrival rate. Hence, the design is able to scale to very large number of nodes. From the results, we can see that on increasing the node arrival rate from 2000 nodes/sec to 4000 nodes/sec, the stabilization time of the system increases by only 2.5 seconds.

The slight increase in stabilization time can be attributed to two factor. With higher rate the underlying Chord overlay used for communication takes more time to stabilize. The number of peers in a level increases exponentially. Hence, if the node arrival rate increases, more levels would be created. As a result, number of peers in the lowermost level would be more. This increases competition among the peers belonging to the same level. This can be reduced by creating sub-levels with in levels.

REFERENCES

[1] R. Schollmeier. "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications". In: Peer-to-Peer Computing, 2001. Proceedings. First International Conference on. 2001, pp. 101–102. DOI: 10.1109/P2P.2001.990434.

[2] BitTorrent. URL: http://www.bittorrent.com/.

[3] Gnutella.URL: http://en.wikipedia.org/wiki/Gnutella.

[4] Skype — Free Calls to friends and family. URL: http:// skype.com.

[5] Xinyan Zhang et al. "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming". In: INFOCOM 2005. 24th Annual Joint Con- ference of the IEEE Computer and Communications Societies.Proceedings IEEE.Vol. 3.IEEE. 2005, pp. 2102–2111.

[6] David P Anderson et al. "SETI@ home: an experiment in public-resource comput- ing". In: Communications of the ACM 45.11 (2002), pp. 56–61.

[7] Ian Clarke et al. "Freenet: A distributed anonymous information storage and re- trieval system". In: Designing Privacy Enhancing Technologies. Springer. 2001, pp. 46–66.

[8] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: Consulted1.2012 (2008), p. 28.

[9] Christos Gkantsidis, Milena Mihail, and Amin Saberi. "Random walks in peer-to- peer networks". In: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. Vol. 1.IEEE. 2004.

[10] Ion Stoica et al. "Chord: A scalable peer-to-peer lookup service for internet ap- plications". In: ACM SIGCOMM Computer Communication Review 31.4 (2001), pp. 149–160.

[11] Antony Rowstron and Peter Druschel. "Pastry: Scalable, decentralized object lo- cation, and routing for large-scale peer-to-peer systems". In: Middleware 2001. Springer. 2001, pp. 329–350.

[12] Ben Y Zhao et al. "Tapestry: A resilient global-scale overlay for service deploy- ment". In: Selected Areas in Communications, IEEE Journal on 22.1 (2004), pp. 41– 53.

[13] XiaojunHei et al. "A measurement study of a large-scale P2P IPTV system". In:Multimedia, IEEE Transactions on 9.8 (2007), pp. 1672–1687.

[14] Bo Li et al. "Inside the new coolstreaming: Principles, measurements and performance implications". In: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE. IEEE. 2008.

[15] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. SScalable application layer multicast.Vol. 32. 4. ACM, 2002.

[16] Duc A Tran, Kien A Hua, and Tai Do. "Zigzag: An efficient peer-to-peer scheme for media streaming". In: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies.Vol. 2.IEEE. 2003, pp. 1283–1292.

[17] Vivek K Goyal. "Multiple description coding: Compression meets the network". In:Signal Processing Magazine, IEEE 18.5 (2001), pp. 74–93.

[18] Miguel Castro et al. "Split Stream: high-bandwidth multicast in cooperative envi- ronments". In: ACM SIGOPS Operating Systems Review. Vol. 37. 5. ACM. 2003, pp. 298–313.

[19] Dejan Kosti´c et al. "Bullet: High band width data dissemination using an over- lay ". In: ACM SIGOPS Operating Systems Review. Vol. 37. 5. ACM. 2003, pp. 282–297.

[20] Venkata N Padmanabhan, Helen J Wang, and Philip A Chou. "Resilient peer-to- peer streaming". In: Network Protocols, 2003. Proceedings.11th IEEE International Conference on.IEEE. 2003, pp. 16–27.

Author's details

1Author Designation, Name of the Department, Institute Name, State, Country, author1@email.com

2Author Designation, Name of the Department, Institute Name, State, Country, author2@email.com

3Author Designation, Name of the Department, Institute Name, State, Country, author3@email.com

4Author Designation, Name of the Department, Institute Name, State, Country, author4@email.com