Volume 5, Issue 12, December 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Minimize Server Load Using Efficient Social Network-Aided Live Streaming System (Save)

Surya M*, Shanthalakshmi K
ECE & Adhiyamaan College of Engineering
Tamil Nadu, India

Abstract*— *In peer-to-peer (P2P) live streaming systems, which refer to content deliver live and sharing video, hence each node required to form an overlay. A node triggers to watch a new channel depends upon centralised server. Live streaming applications make users to watch multiple channel simultaneously, which if widely used, server poses heavy load. To improve efficient live streaming systems, we proposed a Social-network-Aided efficient liVe strEaming system (SAVE). SAVE collects information of nodes interests, channel's watching time and channel interactions. It forms an overlay between nodes with frequent interaction; identify nodes similar interest and watching times. And also builds bridges between overlay of channel with less frequent interactions. We have designed database structure for SAVE systems using MYSQL and constructed templates for user and admin login, video uploads and created friendlist using Net Beans IDE 7.

Keywords*— *friendlist, channel clustering, P2P networks, SAVE, live streaming

I. INTRODUCTION

Most of P2P live streaming applications such as P2PIPTV & PPstream are centralised server. The success of these applications made the need of decentralized server, which posses to reduce load capacity on centralised server. In P2P live streaming system, which refers to streaming video sharing between nodes through P2P overlay formed during watching all nodes a channel. Each node contact the centralized server for every new channel wants to watch. The support of successive and simultaneously watching of multiple channels at a time is difficult in current P2P live which allow users to share stream in one channel. A node watching multiple channels which required forming multiple P2P overlay and it maintains cost is increased. As a node opens more channels, it leads to heavy burden on centralized server and delay response makes inefficiency in P2P live streaming systems.
SAVE proposed to improve the efficiency of live streaming systems. The design of this project is based on utilization of social network. The two main schemes of SAVE system are: channel clustering and friendlist.

Channel clustering scheme:

Most of nodes watch similar interest channel at same time and node watching limited to small number of channels. Hence SAVE performs channel clustering with frequent interactions of channels. It forms overlay between channels with frequent interactions and constructs bridges between overlay of channel with less frequent interaction. Thus, the successive or multichannel watching of users in its current overlay or bridges form new overlay with interfering server.

Freindlist scheme:

A node reaches its destination node with few numbers of steps which implies that a node in channel can knows another channel in few steps through friend connections. Thus every node in SAVE maintains friendlist to save the details of nodes sharing common channels interest and watching time period. If a node wants to watch a channel that is not in present overlay, it refers to friendlist to switch nodes in desired overlay of channel.

II. DESIGN OF SAVE SYSTEM

Fig. 1 shows a high level view of the SAVE structure. The centre of the entire network is the server node and it is denoted by ns. Initially, it forms overlay by all nodes in each channel. Each channel overlay has a stable node with highest capacity and longest lifetime in a channel is a channel head denoted by *hc*. The two main schemes of SAVE: channel clustering and friendlist.
Channel clustering scheme: This scheme considers connecting a group of nodes with frequent interacted channels. It collects the information of watching activities of nodes and single channel grouped to form channel clusters. Channel overlays in one channel cluster are merged into one overlay or bridged. In cluster, each channel overlay head (*hc*) is connected with other channel overlays head.

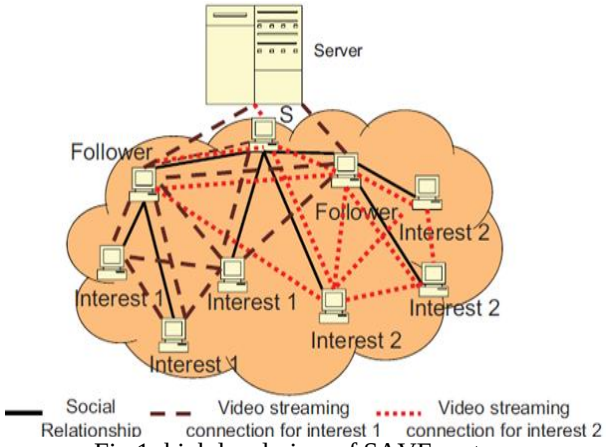

Fig 1: high level view of SAVE system

In each channel cluster, has a cluster head denoted by *hcr* and connects all possible hc in its cluster. The nodes with great probability in each channel cluster can switch to watch their desired channels without interfering ccentral server.

Friendlist scheme: This scheme collects similar interest channels and watching time to maintain a freiendlist. By using friendlist, a node not in its current overlay can join a channel. When a non- first time user can join its desired channel via friendlist without relying the server.

The design format of SAVE structure using JDK.7 was developed coding as shown below:

```
<%@page import="package com.oreilly.servlet.*,java.sql.*,databaseconnection.*,java.util.*,java.io.*,javax.servlet.*,
javax.servlet.http.*"%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="keywords" content="" />
<meta name="description" content="" />
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>A Social-Network-Aided Efficient Peer-to-Peer Live Streaming System</title>
<link href="http://fonts.googleapis.com/css?family=Oxygen" rel="stylesheet" type="text/css" />
<link href="style.css" rel="stylesheet" type="text/css" media="screen" />
<script language="JavaScript" type="text/javascript">
function valid()
{
var a=document.f.n.value;
if(a=="")
{
alert("Enter your Name");
document.f.n.focus();
return false;
}
var b=document.f.p.value;
if(b==""){
alert("Enter password");
document.f.p.focus();
return false;
}
}
</script>
</head>
<body>
<div id="wrapper">
        <div id="header-wrapper">
  <div id="header">
        </br>
        </br>
    <h3 align="center"> <font color="#FFFF00" size="+3"><b>A Social-Network-Aided Efficient Peer-to-Peer Live
Streaming System        </b></font></h3>
                        <div id="logo">
                        </div>
                </div>
        </div>
```



```
        <!-- end #header -->
 <div id="menu">
  <div align="center">
    <h2><font color="#FF0000">Admin Login</font> </h2>
  </div>
 </div>
        <!-- end #menu -->
        <div id="page">
                <div id="page-bgtop">
<div id="page-bgbtm"><table align="center"><tr><td width="400"><p><img src="images/adminlogo.jpg" width="300"
height="150"></p></td><td>                                                <table width="400">
                <form action="adminlo.jsp" name="f" onSubmit="return valid()">
                <tr>
                <td><p><font color="#3300FF">User Name</font></p></td>
                <td><p><input type="text" name="n"></p></td>
                </tr>
                <tr>
                <td><p><font color="#3300FF">Password</font></p></td>
                <td><p><input type="password" name="p"></p></td>
                </tr>
                <tr>
                <td><p></p></td>
<td><p><input type="submit" style="font-face: 'Comic Sans MS'; font-size: larger; color: teal; background-color: #FFFFC0;
border: 3pt ridge green" value="Submit">  
<input type="reset" style="font-face: 'Comic Sans MS'; font-size: larger; color: teal; background-color: #FFFFC0; border: 3pt
ridge green" value="Clear"></p>
                </td></tr></form></table>                                        </td></tr></table>
                <div id="content">
                <div class="post">
                        <div class="entry">                                                </div>
        </div>
        <div style="clear: both;"> </div>
                        </div>
<!-- end #content --><!-- end #sidebar -->
<div style="clear: both;"> </div>
                        </div>
                </div>
        </div>
<!-- end #page -->
</div>
<div id="footer">
        <p></p>
</div>
<!-- end #footer -->
</body>
</html>
```

III. RESULTS AND DISCUSSIONS

The design templates and database structures for user's and admin login, friend's detail, video uploads and new users details was developed using MYSQL and Net Beans IDE software is shown in fig:

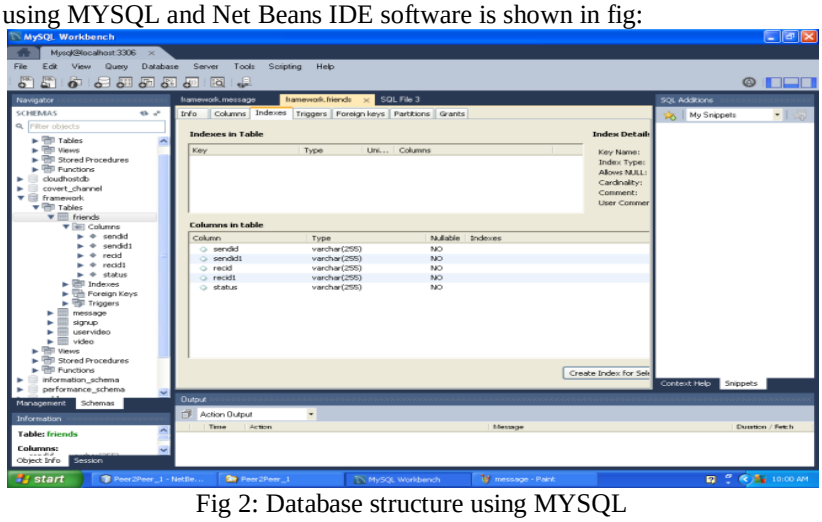

Fig 2: Database structure using MYSQL

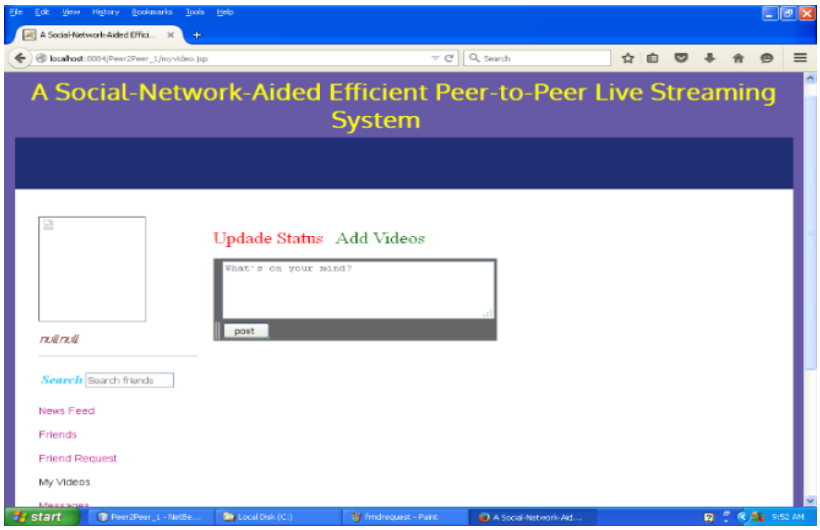

Fig 3: Design page of SAVE system.

IV. CONCLUSIONS AND FUTURE ENHANCEMENT

In this paper, we propose SAVE to improve the efficiency and scalability of P2P live streaming systems. We have designed database structure to maintain a list of user's similar interest, watching time, and new video uploaded, etc. for channel clustering and friendlist using MySQL. Uploaded video depends on server capacity and processing speed, watching time of video is not specific. Hence more number of users can be frequently interacted with this SAVE (Social network-Aided liVe streaming) systems.Our future work lies in writing JAVA coding for each designed buttons to be process the SAVE(Social-network Aided efficient liVe strEaming) system and to evaluate the performance analysis.

REFERENCES

[1] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proc. IEEE*, vol. 96, no. 1, pp. 11–24, Jan. 2008.

[2] C. Wu and B. Li, "Exploring large-scale peer-to-peer live streaming topologies," *Trans. Multimedia Comput.*, vol. 4, no. 3, 2008, Art. no.19.

[3] F. Dobrian, V. Sekar, I. Stoica, and H. Zhang, "Understanding the im- pact of video quality on user engagement," in *Proc. ACM SIGCOMM*, 2011, pp. 362–373.

[4] X. Cheng and J. Liu, "NetTube: Exploring social networks for peer-to-peer short video sharing," in *Proc. IEEE INFOCOM*, 2009, pp. 1152–1160.

[5] A. Fast, D. Jensen, and B. Levine, "Creating social networks to improve peer-to-peer networking," in *Proc. ACM SIGKDD*, 2005, pp. 568–573.

[6] A. Iamnitchi, M. Ripeanu, and I. Foster, "Small-world file-sharing communities," in *Proc. IEEE INFOCOM*, 2004, pp. 952–963.

[7] J. Kleinberg, "The small-world phenomenon: An algorithmic perspec- tive," in *Proc. STOC*, 2000, pp. 163–170.

[8] S. Milgram, "The small world problem," *Psychol. Today*, vol. 2, no. 1, pp. 60–67, 1967.

[9] I. Bermudez, M. Mellia, and M. Meo, "Investigating overlay topologies and dynamics of P2P-TV systems: The case of SopCast," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1863–1871, Oct. 2011.

[10] F. Wang, J. Liu, and Y. Xiong, "Stable peers: existence, importance, and application in peer-to-peer live video streaming," in *Proc. IEEE INFOCOM*, 2008, pp. 2038–2046.

[11] J. Mendes, P. Salvador, and A. Nogueira, "P2P-TV service and user characterization," in *Proc. IEEE CIT*, 2010, pp. 2612–2620.

[12] Y. Chen, E. Merrer, Z. Li, Y. Liu, and G. Simon, "OAZE: A network- friendly distributed zapping system for peer-to-peer IPTV," *Comput. Netw.*, vol. 56, no. 1, pp. 365–377, 2012.

[13] H. Shen, Z. Li, and J. Li, "A DHT-aided chunk-driven overlay for scal- able and efficient peer-to-peer live streaming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2125–2137, Nov. 2012.

[14] Y. Chu, A. Ganjam, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhang, andH. Zhang, "Early experience with an internet broadcast system based on overlay multicast," in *Proc. USENIX*, 2004, p. 12

[15] Z. Liu, C. Wu, B. Li, and S. Zhao, "UUSee: Large-scale operational on-demand streaming with random network coding," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9

