Design and Implementation of a extensible Peer–to–Peer Content Recommendation System

Mangesh V. Bedekar
CS-IS Group,
BITS-Pilani, K. K. Birla, Goa Campus,
Zuarinagar, Goa, INDIA. Pin – 403 726.
mangesh.bedekar@gmail.com
[Corresponding Author]

Dr. Bharat Deshpande
CS-IS Group,
BITS-Pilani, K. K. Birla, Goa Campus,
Zuarinagar, Goa, INDIA. Pin – 403 726.
bharatmsu@yahoo.com

***Abstract*— many file sharing systems exists today. The abundance of content on the web is in itself a huge problem to comprehend.**
The existence and wide scale use of peer-to-peer file sharing systems, which let users share files, folders, directors to entire drives makes the problem of finding relevant information a daunting task.
With the increasing number of file sharing systems, there are challenges in providing users with useful recommendations about interesting products and services, which suit their tastes and behaviors.
In this paper we propose a simple, extensible Peer-to-Peer (P2P) based content recommendation architecture.

Keywords*—*Peer-to-peer file sharing systems, content recommendation systems, DC++, NS2, TCL, log mining

I. Introduction

With the increasing number of file sharing systems, there are challenges in providing users useful recommendations about interesting products and services. Peer-to-peer file sharing systems, just due to their distributed nature operate at high speeds so transmission of data is usually easy difficult is the task of finding relevant information. The main advantage of such systems is their high availability, independent of geographical location and time.

Peer-to-peer file sharing systems can be classified as,

- • Pure peer-to-peer systems
- • Hybrid peer-to-peer systems

Peer-to-peer systems pose unique challenges, in that the volume of data is enormous [1]. Identifying relevant information can be done effectively using content recommender systems.

The rest of the paper is organized as follows. Section-2 describes the Motivation and Requirement. Section-3 describes the System Architecture of the system. Section-4 describes the Architecture Views. Section-5 describes the implementation of the system; Section-6 describes Visualization sub-system, followed by the conclusion, future work and acknowledgements.

II. Motivation And Requirement

A modification to the currently working version of a popular open source peer-to-peer file sharing system was thought of. DC++ is the system into which addition of the functionality to get recommended downloads is envisioned [2].

Recommendations are planned to be provided on the basis of the users association with other users on the network. Associations are extracted from the client log files and server log files.

Based on the past history of downloads and users association with other users, relevant files are to be recommended for users to access / download.

III. System Architecture

The platform used for the p2p content recommendation systems implementation is DC++, Direct Connect++. Direct Connect is a peer-to-peer file sharing network but it uses a central server.

A. The Users

The users of file sharing systems seldom reveal any information about themselves. The usual information revealed is only the users IP Address and the users Nick name.

As an input, the current Nick name of the user, who is requesting for the download, is given along with the level of associations he is looking for.

In DC++ terminology, there are two types of users that may be present on the hub at any time,

- • Active users
- • Passive users

Connectivity modes too differ; there are two types of connection methods,

- • Passive mode
- • Active mode

Passive mode is the simplest mode to use but it has a drawback, it has limitations in connectivity (passive users cannot connect to other passive users). It also causes strain on hubs as all passive mode communication must go through the hub. This mode is therefore a compatibility mode and is only used if active mode cannot be used at any time.

Active method is the "standard" way of connecting to others but depending on the network topology, it may be difficult to enable.

B. The Association between Users

Level of association recursively defines friends, friends of friends etc.

Example – Say user A is associated with user B, user B is associated with user C and so on.
Then for user A, his first level associations includes user B, his level-two associations include user C and so on.

In order to provide content recommendations to user, the associations of the user are identified. Further associations are recursively found as 1st, 2nd level and so on till the level specified, associated nick-names are identified and their downloaded files are provided to the user in the order as follows-

- • 1st level association downloads,
- • 2nd level association downloads,
- • 3rd level association downloads, and so on.

C. The Platform
DC++ is a free open-source, ad and spy-ware-free client, written in C++ for the Direct Connect protocol. DC++ allows users to share files and chat over the Internet / Intranet with other users.

Direct Connect defines the servers as Hub's. Clients connect to a central hub and that hub feature a list of clients or users connected to it. Users can then search for files to download, or chat with other users present (on that server).
Clients connect to a central hub and that hub features a list of clients or users connected to it.

Therefore DC is a combination of client server as well as peer to peer. Initially users connect to a central server wherein all shared files and all online users at that time can bee seen. All private chats are also routed through the hub/server. When the user has to download a file, it is downloaded directly from the client, at this point of time; it becomes a peer-to-peer network.

A "hub" is a piece of software that routes chat and search requests / results and facilitates clients to connect to each other. It's not called a server because it doesn't share any files. All file transfers are being made directly between respective clients, not through the hub.

D. The Log Files

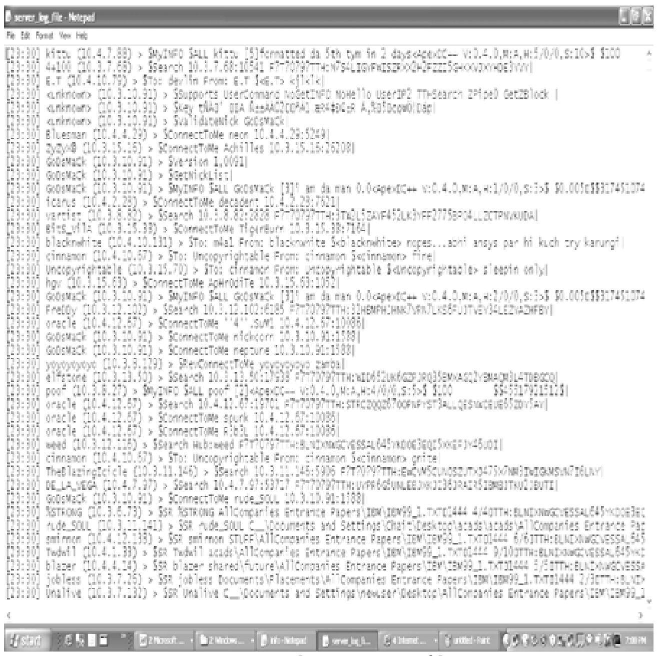
The DC++ system has a logging feature both at 'Hub' / Server side as well as at individual user 'client' level. Log files are regularly generated and are updated on any activity by the users on the file-sharing network. The logs files are used to generate a set of user communications based on private messages as well as the files they have downloaded.

The private message details are obtained by parsing the server log file while the download associations are obtained by parsing the individual client log files.

Downloads are one way associations wherein one user downloads from the other while private message details are logged only if there has been at least one two way communication (association) between the two users, i.e. if user- A sends a message to user B and user B replies back to user A.

E. The Server Log File

The 'Hub' (server) logs all requests for search, download and chat requests-response sessions.



```
[23:30] kittu (10.4.7.88) > $MyINFO $ALL kittu [5]formatted da 5th tym in 2 days<ApexDC-- V:0.4.0,M:A,H:5/0/0,S:10>$ $100
[23:30] 44100 (10.3.7.68) > $Search 10.3.7.68:10541 F?T?0?9?TTH:[illegible]|
[23:30] E.T (10.4.10.79) > $To: devilin From: E.T $<E.T> kjlklk|
[23:30] <unknown> (10.3.10.91) > $Supports UserCommand NoGetINFO NoHello UserIP2 TTHSearch ZPipe0 GetZBlock |
[23:30] <unknown> (10.3.10.91) > $Key [illegible]|
[23:30] <unknown> (10.3.10.91) > $ValidateNick GoDsMaCk|
[23:30] Bluesman (10.4.4.29) > $ConnectToMe neon 10.4.4.29:5249|
[23:30] ZyZyx@ (10.3.15.15) > $ConnectToMe Achilles 10.3.15.15:26208|
[23:30] GoDsMaCk (10.3.10.91) > $Version 1,0091|
[23:30] GoDsMaCk (10.3.10.91) > $GetNickList|
[23:30] GoDsMaCk (10.3.10.91) > $MyINFO $ALL GoDsMaCk [3]' am da man 0.0<ApexDC++ V:0.4.0,M:A,H:1/0/0,S:3>$ $0.005d$$317451074
[23:30] icarus (10.4.2.28) > $ConnectToMe decadent 10.4.2.28:7621|
[23:30] vertist (10.3.8.82) > $Search 10.3.8.82:2828 F?T?0?9?TTH:[illegible]|
[23:30] BitS_vIlA (10.3.15.38) > $ConnectToMe TigerBurn 10.3.15.38:7164|
[23:30] blacknwhite (10.4.10.131) > $To: m4al From: blacknwhite $<blacknwhite> nopes...abhi ansys par hi kuch try karungi|
[23:30] cinnamon (10.4.10.57) > $To: Uncopyrightable From: cinnamon $<cinnamon> fine|
[23:30] Uncopyrightable (10.3.15.70) > $To: cinnamon From: Uncopyrightable $<Uncopyrightable> sleepin only|
[23:30] hgv (10.3.15.63) > $ConnectToMe ApHrOdiTe 10.3.15.63:1052|
[23:30] GoDsMaCk (10.3.10.91) > $MyINFO $ALL GoDsMaCk [3]' am da man 0.0<ApexDC++ V:0.4.0,M:A,H:2/0/0,S:3>$ $0.005d$$317451074
[23:30] FreDDy (10.3.12.102) > $Search 10.3.12.102:6185 F?T?0?9?TTH:[illegible]|
[23:30] oracle (10.4.12.67) > $ConnectToMe ''4''.SuM1 10.4.12.67:10086|
[23:30] GoDsMaCk (10.3.10.91) > $ConnectToMe nickcorn 10.3.10.91:1588|
[23:30] GoDsMaCk (10.3.10.91) > $ConnectToMe neptune 10.3.10.91:1588|
[23:30] yoyoyoyoyo (10.3.8.129) > $RevConnectToMe yoyoyoyoyo zamba|
[23:30] elfstone (10.3.13.50) > $Search 10.3.13.50:17938 F?T?0?9?TTH:[illegible]|
[23:30] poof (10.3.8.27) > $MyINFO $ALL poof [2]<ApexDC-- V:0.4.0,M:A,H:4/0/0,S:5>$ $100        $$[illegible]$|
[23:30] oracle (10.4.12.67) > $Search 10.4.12.67:19701 F?T?0?9?TTH:[illegible]|
[23:30] oracle (10.4.12.67) > $ConnectToMe spunk 10.4.12.67:10086|
[23:30] oracle (10.4.12.67) > $ConnectToMe RibiL 10.4.12.67:10086|
[23:30] weed (10.3.12.115) > $Search Hub:weed F?T?0?9?TTH:[illegible]|
[23:30] cinnamon (10.4.10.57) > $To: Uncopyrightable From: cinnamon $<cinnamon> gnite|
[23:30] TheBlazingIcicle (10.3.11.146) > $Search 10.3.11.146:5906 F?T?0?9?TTH:[illegible]|
[23:30] DE_LA_VEGA (10.4.7.97) > $Search 10.4.7.97:53717 F?T?0?9?TTH:[illegible]|
[23:30] GoDsMaCk (10.3.10.91) > $ConnectToMe rude_SOUL 10.3.10.91:1588|
[23:30] %STRONG (10.3.6.73) > $SR %STRONG AllCompanies Entrance Papers\IBM\IBM99_1.TXT01444 4/40TTH:[illegible]
[23:30] rude_SOUL (10.3.11.141) > $SR rude_SOUL C__\Documents and Settings\Chaitu\Desktop\acads\acads\AllCompanies Entrance Pa
[23:30] sminnon (10.4.12.138) > $SR sminnon STUFF\AllCompanies Entrance Papers\IBM\IBM99_1.TXT01444 6/6TTH:[illegible]
[23:30] Twdwfl (10.4.1.38) > $SR Twdwfl acads\AllCompanies Entrance Papers\IBM\IBM99_1.TXT01444 9/10TTH:[illegible]
[23:30] blazer (10.4.4.14) > $SR blazer shared\Future\AllCompanies Entrance Papers\IBM\IBM99_1.TXT01444 5/5TTH:[illegible]
[23:30] jobless (10.3.7.26) > $SR jobless Documents\Placements\AllCompanies Entrance Papers\IBM\IBM99_1.TXT01444 2/3TTH:[illegible]
[23:30] Unalive (10.3.7.132) > $SR Unalive C__\Documents and Settings\newuser\Desktop\AllCompanies Entrance Papers\IBM\IBM99_1
```

Figure 1. The Server Log file

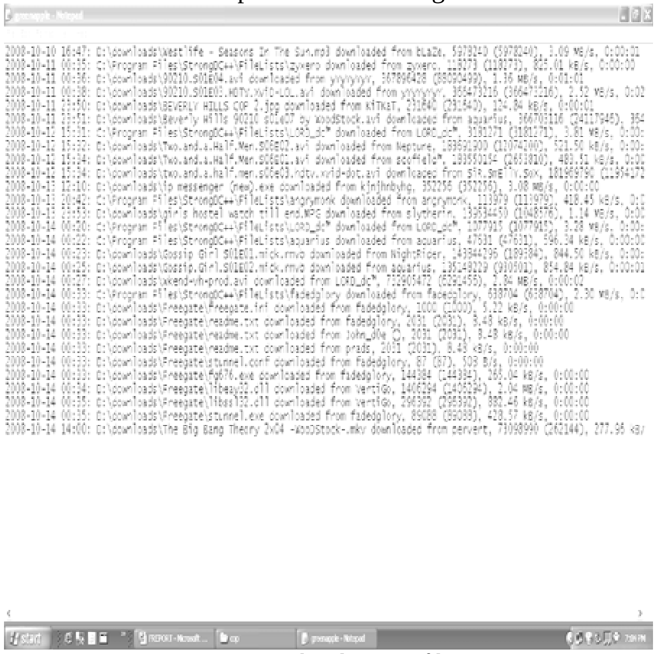
The server logs all the commands that are issued as response to user generated requests. A sample set of commands are as under,

$To: <othernick> From: <nick> $<<nick>> <message>|
$Version <version>|
$Key <calculated key>|
$ValidateNick <nick>|
$Search <ip>:<port> <searchstring>
$Search Hub:<requestornick> <searchstring>
$MyINFO $ALL <nick> <description>$
$<connection><flag>$<e-mail>$<sharesize>$|
$GetNickList|
$Supports <feature1>[<feature2>[<featureN>]]|
$ConnectToMe <SenderNick> <RemoteNick>
<SenderIp>:<SenderPort>
$RevConnectToMe <ClientNick> <TargetNick>|

F. The Client Log File

The client also has a log feature to log the users actions invoked through the client software. The log file in the 'hub' server only contains the download information for the passive user as it is routed through the hub.
In passive connection mode, DC++ will only make outbound connections to other users. All searches will be sent through the hub, and search results will be returned through the hub as well.

Since the server files contains information only of the passive user downloads, the client log files were also required so as to tap all the files downloaded from other users, which would later be used in generating content to be recommended. A sample of the client log file is as under.



```
2008-10-10 16:47: C:\downloads\Westlife - Seasons In The Sun.mp3 downloaded from bLaZe, 5978240 (5978240), 3.09 MB/s, 0:00:01
2008-10-11 00:35: C:\Program Files\StrongDC++\FileLists\zyxero downloaded from zyxero, 118273 (118273), 825.01 kB/s, 0:00:00
2008-10-11 00:36: C:\downloads\90210.S01E04.avi downloaded from yyyyyyy, 367896428 (38050499), 1.36 MB/s, 0:01:01
2008-10-11 00:38: C:\downloads\90210.S01E03.HDTV.XviD-LOL.avi downloaded from yyyyyyy, 366473216 (366473216), 2.52 MB/s, 0:02
2008-10-11 23:50: C:\downloads\BEVERLY HILLS COP 2.jpg downloaded from KiTKaT, 231640 (231640), 124.84 kB/s, 0:00:01
2008-10-11 23:51: C:\downloads\Beverly Hills 90210 s01e07 by WoodStock.avi downloaded from aquarius, 366703116 (24117945), 354
2008-10-12 15:31: C:\Program Files\StrongDC++\FileLists\LORD_dc* downloaded from LORD_dc*, 3181271 (3181271), 3.81 MB/s, 0:00:
2008-10-12 15:32: C:\downloads\Two.and.a.Half.Men.S06E02.avi downloaded from Neptune, 183591300 (12074200), 521.50 kB/s, 0:00:
2008-10-12 15:34: C:\downloads\Two.and.a.Half.Men.S06E01.avi downloaded from scoffield*, 183550154 (2653810), 483.51 kB/s, 0:0
2008-10-12 15:34: C:\downloads\two.and.a.half.men.s06e03.hdtv.xvid-dot.avi downloaded from SiR.SmElly.SoX, 181969790 (11954172
2008-10-13 12:10: C:\downloads\ip messenger (new).exe downloaded from kjnjhnbyhg, 352256 (352256), 3.08 MB/s, 0:00:00
2008-10-13 20:42: C:\Program Files\StrongDC++\FileLists\angrymonk downloaded from angrymonk, 113979 (113979), 408.45 kB/s, 0:0
2008-10-13 23:53: C:\downloads\girls hostel watch till end.MPG downloaded from slytherin, 139534450 (1048576), 1.14 MB/s, 0:0
2008-10-14 00:20: C:\Program Files\StrongDC++\FileLists\LORD_dc* downloaded from LORD_dc*, 1077915 (1077915), 3.28 MB/s, 0:00:
2008-10-14 00:22: C:\Program Files\StrongDC++\FileLists\aquarius downloaded from aquarius, 47631 (47631), 596.34 kB/s, 0:00:0
2008-10-14 00:23: C:\downloads\Gossip Girl S01E01.mick.rmvb downloaded from NightRider, 143844296 (189384), 844.50 kB/s, 0:00:
2008-10-14 00:25: C:\downloads\Gossip.Girl.S01E02.mick.rmvb downloaded from aquarius, 135149229 (910501), 854.84 kB/s, 0:00:01
2008-10-14 00:27: C:\downloads\xkend-vh-prod.avi downloaded from LORD_dc*, 732905472 (6291455), 2.84 MB/s, 0:00:02
2008-10-14 00:33: C:\Program Files\StrongDC++\FileLists\fadedglory downloaded from fadedglory, 638704 (638704), 2.30 MB/s, 0:0
2008-10-14 00:33: C:\downloads\Freegate\Freegate.ini downloaded from fadedglory, 1000 (1000), 5.22 kB/s, 0:00:00
2008-10-14 00:33: C:\downloads\Freegate\readme.txt downloaded from fadedglory, 2031 (2031), 3.48 kB/s, 0:00:00
2008-10-14 00:33: C:\downloads\Freegate\readme.txt downloaded from John_d0e (), 2031 (2031), 3.48 kB/s, 0:00:00
2008-10-14 00:33: C:\downloads\Freegate\readme.txt downloaded from prads, 2031 (2031), 3.48 kB/s, 0:00:00
2008-10-14 00:33: C:\downloads\Freegate\stunnel.conf downloaded from fadedglory, 87 (87), 503 B/s, 0:00:00
2008-10-14 00:33: C:\downloads\Freegate\fg676.exe downloaded from fadedglory, 144384 (144384), 265.04 kB/s, 0:00:00
2008-10-14 00:34: C:\downloads\Freegate\libeay32.dll downloaded from VertiGo, 1406294 (1406294), 2.04 MB/s, 0:00:00
2008-10-14 00:35: C:\downloads\Freegate\libssl32.dll downloaded from VertiGo, 296392 (296392), 882.46 kB/s, 0:00:00
2008-10-14 00:35: C:\downloads\Freegate\stunnel.exe downloaded from fadedglory, 89088 (89088), 428.57 kB/s, 0:00:00
2008-10-14 14:00: C:\downloads\The Big Bang Theory 2x04 -WooDStock-.mkv downloaded from servert, 73098990 (262144), 277.96 kB/
```

Figure 2. The Client Log file

Each record/entry in the client log file has the following fields,

- Timestamp of the download
- File downloaded along with the exact extension from
- where it is downloaded
- name of the person (DC nick) from whom the file is being downloaded
- speed of download
- time for the download

IV. System architecture Views

The system functionality on both the Server side and the Client Side is depicted as below.

A. Server Side functionality

The Server Log file is processed to identify various components namely, commands, associations between users, private messaging patterns between users, the Nickname to IP Address mapping, Active users, Passive users as follows.

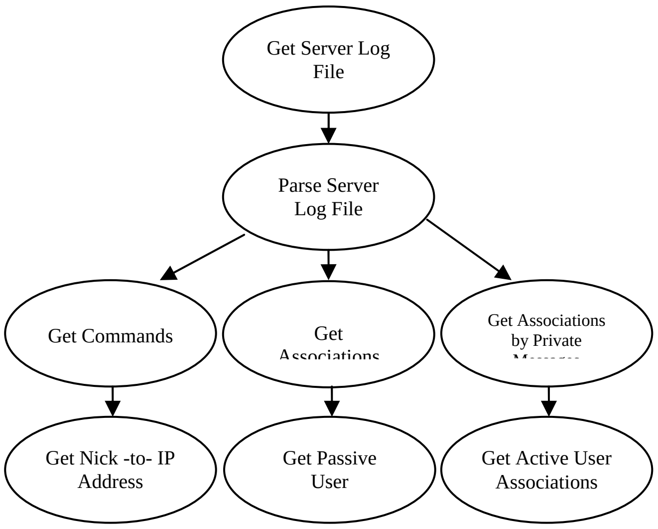


Figure 3. The Server side Functionality

The Client Log file is also processed to identify the remaining user specific components namely, downloaded files, which file downloaded from which user, users association with other users in the system, file lists from client machine, users which have been marked as favorite, as shown below.

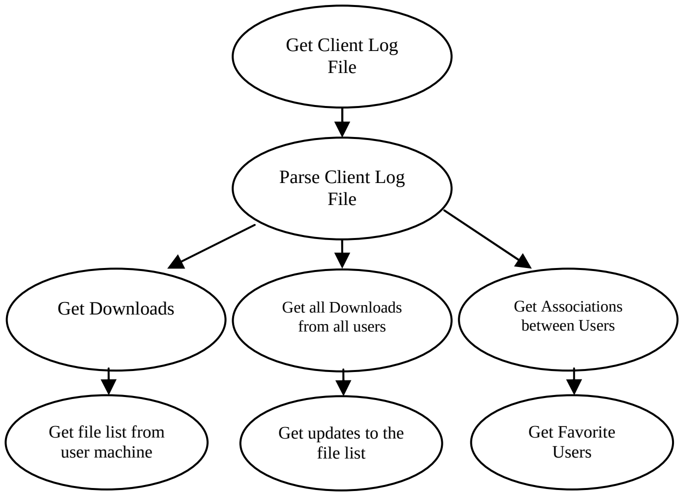

Get Client Log File
Parse Client Log File
Get Downloads
Get all Downloads from all users
Get Associations between Users
Get file list from user machine
Get updates to the file list
Get Favorite Users

Figure 4. The Client side Functionality

V. IMPLEMENTATION DETAILS

The entire system was implemented in C++, so as be compatible with the existing DC++ system. The system was implemented in three stages as explained below.

STAGE 1 – Server Side Log Parsing
1. Get all the unique commands in the Log File.
2. Get the association between the <Nick—Name> and their corresponding <IP—Address>.
3. Get the associations between users who are communicating using private messaging system.

STAGE 2 – Client Side Log Parsing
1. Get all the download association from the particular user with other users.
2. Get all the downloaded files from all users, it is retrieved in the format, <nick-name> and <Downloaded-File>
3. Identify two-way associations between users.

STAGE 3 – Generating Content Recommendations
From the results of processing the Server Log and Client Log,
1. Generate Recommendations to the user in question.
2. Remove duplicates from the recommendation system.
3. Give suggested content recommendations to the user.

All the processing is done using C++ programming language to make it directly compatible with the existing code of DC++.

The server module runs on the server and communicates with the client modules running on various client machines.

The code has been incorporated as a separate module in the DC++ source code. On the User Interface, a separate button has been created, which allows the user to invoke the recommendation system and also view content recommended for access / download.

VI. GRAPHICAL VISUALISATIONS OF ASSOCIATIONS AND RECOMMENDATIONS

The Visualizations involved are depicted using the NS2 Network Simulator [3, 4]. This simulator is used with C++ and TCL (Tool Command Language) [5, 6].

We wrote an interface which takes in the details as mentioned above and generates the resultant TCL code to be used with NS2.

The TCL file has two parts, namely,
- The Structure (Static) part
- The Behavior (Dynamic) part

A. Architecture of the Visualisation sub-system

The visualization sub-system is developed in two stages, the ‘structure part’ and the ‘behavior part’.
- The structure part is static; as in the information herein does not change over time.
- The behavior part is dynamic; the information herein depicts the associations (both one and two way) between users.

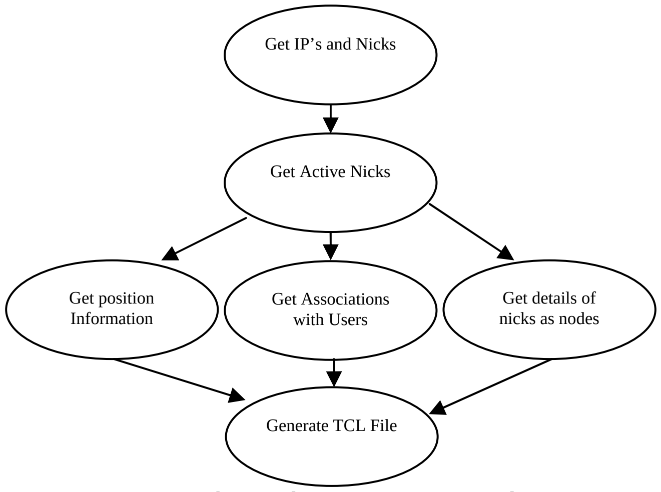
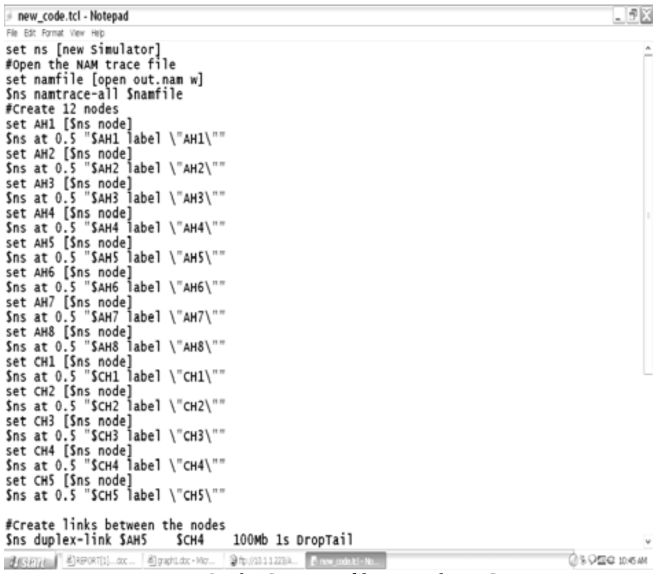


Figure 5. The Visualisation system Functionality

The structure (static) part consists of,
- The nodes (users)
- The positioning of the nodes on the graph spatially as per their geographic locations
- Names of the nodes

The behavior (dynamic) part consists of,
- Plotting one-way associations between nodes
- Plotting two-way associations between nodes

The C++ code results in a TCL file as shown below.



```
set ns [new Simulator]
#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
#Create 12 nodes
set AH1 [$ns node]
$ns at 0.5 "$AH1 label \"AH1\""
set AH2 [$ns node]
$ns at 0.5 "$AH2 label \"AH2\""
set AH3 [$ns node]
$ns at 0.5 "$AH3 label \"AH3\""
set AH4 [$ns node]
$ns at 0.5 "$AH4 label \"AH4\""
set AH5 [$ns node]
$ns at 0.5 "$AH5 label \"AH5\""
set AH6 [$ns node]
$ns at 0.5 "$AH6 label \"AH6\""
set AH7 [$ns node]
$ns at 0.5 "$AH7 label \"AH7\""
set AH8 [$ns node]
$ns at 0.5 "$AH8 label \"AH8\""
set CH1 [$ns node]
$ns at 0.5 "$CH1 label \"CH1\""
set CH2 [$ns node]
$ns at 0.5 "$CH2 label \"CH2\""
set CH3 [$ns node]
$ns at 0.5 "$CH3 label \"CH3\""
set CH4 [$ns node]
$ns at 0.5 "$CH4 label \"CH4\""
set CH5 [$ns node]
$ns at 0.5 "$CH5 label \"CH5\""

#Create links between the nodes
$ns duplex-link $AH5    $CH4    100Mb 1s DropTail
```

Figure 6. The Structure file created in TCL

The behavior (dynamic) part is plotted over the static part.

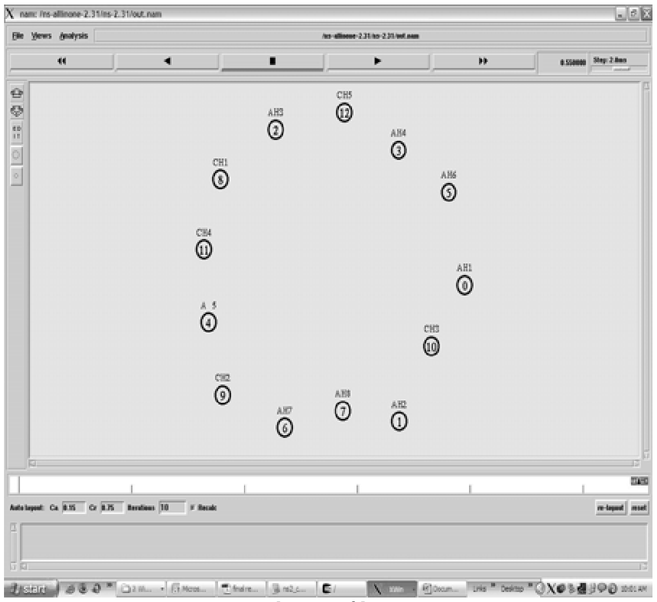


Figure 7. The TCL file run in NS2

The dynamic plot with associations between users is as shown below.

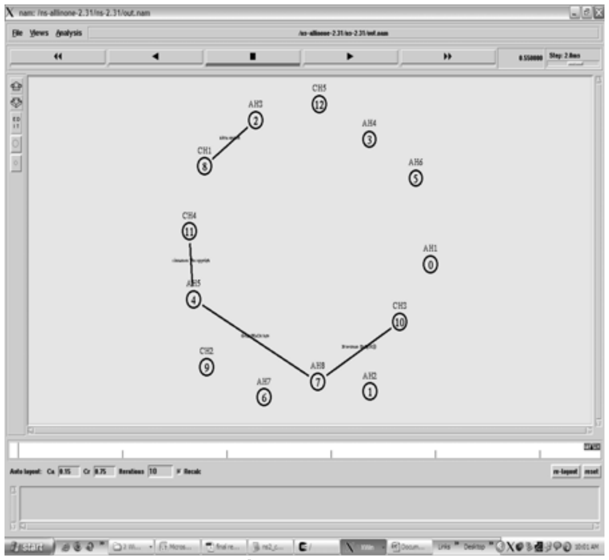


Figure 8. The run time view in NS2

The real time operation of the system is as depicted below.

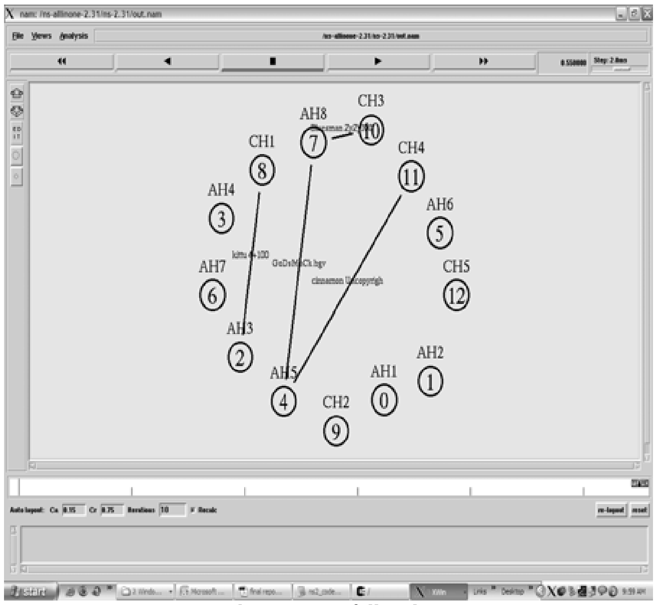


Figure 9. The system in full scale operation

CONCLUSION

In this paper we have proposed the design and shown the implementation of a simple, extensible P2P content recommendation system which is very scalable.
The key point in the system is identifying various levels of associations between users. We have identified four types of associations that exist between users of any P2P system. These associations are effectively and efficiently to generate user focused content recommendations.

FUTURE WORK

Future work includes implementing the same system for users on the move. The system is very light in weight which makes it very easy to implement the same on Laptops, portable devices and mobile phones.

For laptops, the system can be used to recommend available wireless networks, product recommendations etc. For mobile phones, the system can be used to recommend ringing tones, wallpapers, logos etc.

ACKNOWLEDGMENT

We wish to thank two of our ex-students who helped a lot in implementation of the system namely, Ms. Aakriti Agrawal [2005T6PS453G] and Ms. Gargi Apte [2005C6PS500G].

REFERENCES

[1] A. Oram, editor. Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O'Reilly and Associates, 2001.
[2] DC++, Available at http://www.dcplusplus.sourceforge.net/download/
[3] NS2, Available at http://www.isi.edu/nsnam/ns/
[4] Using NS Simulator, Available at,
http://www.cs.sunysb.edu/~samir/cse590/ns-simulator.htm
[5] TCL, Available at http://www.tcl.tk/
[6] TCL-Source forge, Available at http://tcl.sourceforge.net/