CONCEPTUAL DEVELOPMENT OF RESOURCES DISCOVERY IN THE PROPOSED HYBRID P2P VIDEO STREAMING

PROFESOR DR. NANNA SURYANA

Paper Published in Journal Of Advanced Manufacturing Technology
Special Edition International Conference On Engineering And ICT (ICEI 2007)
Vol. 1, No. 1, Nov.-Dec. 2007

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

CONCEPTUAL DEVELOPMENT OF RESOURCES DISCOVERY IN THE PROPOSED HYBRID P2P VIDEO STREAMING

Tajuddin, A.S.,[1] and Nanna, S.H.[2]

[1]Security, Intelligent Application and Multimedia Program
TM Research and Development, Serdang, Malaysia and
[2]Faculty Information Technology and Communication (FTMK)
Technical University of Malaysia Melaka (UTeM)
Melaka, Malaysia
[2]nsuryana@utem.edu.my

ABSTRACT

We present the design of a hybrid Peer-to-Peer (P2P) system for video streaming. In this paper, we address the availability, accessibility and lookup service of files. We use the advantages of server-client business model to search and retrieve the information. We implement the base ontology of video domain repository so that the final result may be different and provide more results from the keyword search. To provide the dynamic standby peer, we use checksum value as an indicator to search an identical content in the Peer-to-Peer network. We hypothesize that, by using server-client searching in Peer-to-Peer application, we can reduce the latency lookup services, path length, peer load and network traffic.

Keywords: hybrid Peer-to-Peer, lookup service, information retrieval, multimedia distribution

1.0 INTRODUCTION

File sharing becomes more trendy when peer-to-peer (P2P) application continues to emerge and give an option for the community to share their content, such as BitTorrent [5] , eDonkey2000 [11], Gnutella [12], FreeNet [14], KaZaA [17] and Napster [28]. Nowadays, the P2P technology is not only used for downloading but also for streaming, and many researchers have tried to implement video streaming in P2P environment, such as ZIGZAG [10], SplitStream [25], CollectCast [27], GnuStream [35] and CELL [36]

In our solution, we designed a P2P service for media streaming. The streaming session is a unicast session and consists of one sender to a single receiver peer, similar to a video on demand application. The only difference is that peers act both as a server and a client, thus, eliminates the use of dedicated video server to manage and stream the video. To backup the sender, we proposed the dynamic standby peer to replace the failure active sender. In our previous paper [1], we have presented a multiple distributed server on top of structured overlay network topology, which is composed of one nucleus supervisor and one or many child supervisor(s). This supervised peer will manage peers, and the peer topology itself is an unstructured overlay network topology. We rank the peer from bad peer to good peer. We believe that, the selection of the best peer to serve a streaming session is vital in providing good quality of environment for video streaming. The objective of this paper is to preserve the quality of the P2P video streaming application by:

i. Designing the topology of overlay for dynamic servers and peers;
ii. Improving the availability of files, lookup service and control delegation; and
iii. Maintaining a good real-time playback to requesting peer.

This paper is organized as follows: In section 2, we discuss the Related Work. Discussions on existing works particularly on availability, accessibility, searching and retrieval of the information will be presented. Section 3, the Research Design, discusses on how the information of file will be pushed into supervised peer, queries and matching, and the ontology connected to video realm. In section 4, we briefly discuss our simulation finding on path length for file availability and accessibility in different P2P model such as proposed design, Gnutella, BitTorrent, Chord and FreeNet. Finally, the conclusion of this paper is presented in Section 5.

2.0 RELATED WORK

Decentralized or pure P2P system, such as Pastry [2], Tapestry [4], Viceroy [7], Chord [15], Kademlia [30], CAN [33], Gnutella, KaZaA, CollectCast, FreeNet and CELL does not have a server to provide a location of data, whereas centralized P2P system, such as Napster, BitTorrent, eDonkey2000 and SPON [6],[18] uses a server as an advantage tool to locate data. CAN, Chord, Tapestry, Pastry, Kademlia, Viceroy and CollectCast offer Distributed Hash Table (DHT), in which DHT can guarantee to locate data within their overlay network topology. The application performs a query to match a key and/or NodeID over DHT routing information. Table 1 shows the various P2P systems which have different routing journey paths for lookup services.

2.1 Structured Pure P2P Application

Chord is a structured P2P application that is completely decentralized and symmetric, and it can find data using *O(log N)*. Each Chord node needs "routing" information about a few other nodes only. It uses DHT in which a Chord node communicates with other nodes in order to perform a lookup. Each node only needs to know how to contact its current successor's node on the identifier circle. Queries for a given identifier could be passed around the circle via these successor pointers using finger table until they encounter a pair of nodes that overlap with the desired identifier. The finger table has been generated using a formula as described in (1). In the steady state, in an N-node system, each node maintains information about only *O(log N)* other nodes, and resolves all lookups via *O(log N)* messages to other nodes.

$(n + 2^{k-1}) \bmod 2^m$, 1 d" k d" m (1)

n = number of nodes in the system; k = key; m = bit identifiers.

FreeNet also implements DHT but of a loosely type. FreeNet uses keys (keyword-signed key and signed-subspace key) and descriptive text strings to identify data objects. It will search from peer to peer until the requests exceed the Hops-To-Live limits and reach specified number of result set. If a file is found during a lookup service, the identified file will be successfully retrieved by the original requester, and it will cache on sequence upstream requester data-store. If there is not enough data-store space for the newer data item, it will remove the lowest data item and insert a new one. Thus, this method will increase the availability and accessibility of popular files by shortening the path length and results in the time lookup to decrease rapidly over time. Unfortunately, non-popular files will have a shorter life in FreeNet network.

In CollectCast, the authors used Tapestry as a lookup services mechanism. They modified the Tapestry so that it can return one or more supplier peer. CollectCast has a delegation control module to select which peer(s) should be able to become the supplier. They use offer rate and available bandwidth over topology-aware selection technique to choose the active and standby sender. The problem in this system is that their standby sender lists are static. They do not have a solution to overcome the situation when all standby peers leave the network.

2.2 Unstructured Pure P2P Application

Gnutella uses flooding technique which requires *O(N)* steps with a limit Time-To-Live (TTL). Unfortunately, Gnutella cannot guarantee to locate data although its searching request will saturate the internet network. Gnutella introduced Ultrapeer [3] to improve the lookup services. The peer itself will become an Ultrapeer automatically if they meet the minimal constraint, such as not under firewall, suitable operating system, sufficient uptime, etc. Once they become an Ultrapeer, they should be able to receive and store the meta-data of shared data, and also process the incoming query requests from connected peers and another Ultrapeer.

In KaZaA, peers are connected to their Super-Peers [23]. The functionality of Super-Peer is similar to Ultrapeer. The difference is that the user of KaZaA can choose to become the Super-Peer. When a peer joins the network, it tries to connect to the existing Super-Peer, and starts to push the metadata to Super-Peer.

CELL is principally used to cache any amount of video data, to reduce the search scope of a video lookup and to enhance the availability of a video through caching coordination. Supplying and requesting peer will determine whether the requesting peer should be able to cache some of the segmented video file, and become caching host. They use Gnutella-like technique to locate data, and the searching will stop when one caching host is found. From there, by using CacheTable, they can locate complete sets of video segments.

GnuStream is built on top of Gnutella, and it integrates dynamic peer location and streaming capacity aggregation. Each GnuStream streaming session is controlled by the receiver peer and involves a dynamic set of peer senders instead of one fixed sender. The receiver aggregates streaming bandwidth from the multiple senders, achieving load distribution and fast reaction to sender capacity and on/off-line status changes.

2.3 Hybrid P2P Application

In BitTorrent network, central server (i.e. Supernova [32]) will collect and store a *.torrent* file. Peers do a search with *O(1)* step via server and get a reply which consists of one file with multiple peers and multiple files with identical content in one result set. Once a peer selects and downloads *.torrent*, it also gets Universal Resource Locator of selected tracker. When BitTorrent client extracts the information in .torrent, it will start to initiate and download the data segmented of 256kb size from one or multiple peers and at the same time update his activity to tracker. The functionality of this tracker is to keep track download/upload activity by peers so that indirectly the tracker knows where to find a new supplier. Downloading process will be completed after all segmented files are downloaded and integrated into one file.

Napster is a pioneer in introducing a centralized server. The server will collect all meta-data and provide a location of data to peers. Meta-data stored at a server with *O(N)*, where N is the number of peers. The peer will search the file with *O(1)* step via server, and return with possible multiple files with identical content in one result set. Hence, with the server storing all information about the shared file, the Napster network can guarantee to locate data. Basically, eDonkey2000 follows the business model of Napster.

Table 1: Routing journey in various P2P systems

Application	**Routing step in lookup services**	**Application**	**Routing step in lookup services**
CAN	$O(d.N^{1/d})$	Gnutella	O(N)
Chord	O(log N)	GnuStream	O(N)
Tapestry	$O(\log_B N)$	CELL	O(N)
Pastry	$O(\log_{2^b} N)$	Napsters	O(1)
Kademlia	$O(\log_B N) + c$	BitTorrent	O(1)
Viceroy	O(log N)	eDonkey2000	O(1)
CollectCast	$O(\log_{2^b} N)$	Proposed Design	O(n)
SPON	O(1)		

Legend:
N- number of peers in network
d - number of dimensions
B- base of the chosen peer identifier
b- number of bits
c- small constant
n- number of child supervisor

2.4 Search Engine

Besides P2P application, we can look at an example in web search engine [31], such as Google and Yahoo! Google has a PageRank system which rates the relevancy of Web pages to queries based not only on whether they include keywords but also by how many other relevant pages link to them, whereas Yahoo! uses human editors to provide metadata about many Web sites, giving the system a way to judge the relevancy of potential search results.

3.0 RESEARCH DESIGN

We design an unstructured peer overlay network topology over structured distributed supervised peer. The distributed supervised peer system consists of one nucleus supervisor and one or many child supervisors. The function of these servers are to manage the accountability and availability of peers, such as joining/leaving request, content shared registration, rank predictions, files delegation and meta-data searching, whereas each peer will manage, interact and stream the content. We design the system so that we can do a dynamic child supervisor, which means that they can join or leave any time without degrading the performance of the system (of course, we still need at least one child supervisor in the system). Each supervisor has it own database and the nucleus server as well.

3.1 Network Topology

Each peer needs to register with the nucleus supervisor for the first time. Once the registration is done, the nucleus supervisor will push the information (e.g. IP address and NodeID) about the existing child supervisors(s) to the peer. The peer will create the Identifier Table to store all information regarding the child supervisor(s). The nucleus supervisor also alerts all child supervisors about a new peer. Then, the peer should be able to join the P2P network anytime using the identifier table. If the login process fails (e.g. timeout or overloaded), it will re-connect to another child supervisor. Once the peer successfully joins the system, the Transmission Control Protocol (TCP) connection between the child supervisor and peer will be closed. The child supervisor will assume that the peer(s) is still online based on the periodic message received from peer(s). We choose this mechanism to make sure that other peer(s) can perform join request concurrently by reducing the symptom of bottleneck. The child supervisor can also re-direct the join request to another child supervisor if it is overloaded with peers.

When a new child supervisor wants to join the system, the child supervisor will communicate with the nucleus supervisor, and ask for the identifier table information.

Once the child supervisor peer receives the information, it will create a new identifier table, and start to request and establish the TCP connection with other child supervisors, and tell them, "I'm Your Sibling". Each recipient child supervisor will update his/her identifier table and push the new child supervisor information to online peer. The peer should be able to insert a new record inside the identifier table. For those offline peers, they will receive it after logging into the system.

Figure 1. shows three distributed child supervisors S1, S2 and S3 that have to manage different peers. S1 monitors P1, P2 and P3; S2 monitors P5, P6 and P8; and S3 monitors P7, P9 and P4. S1, S2 and S3 are connected to each other so as to be aware of and replicate updated information about peer(s) to Sn (e.g. bandwidth and packet loss rate). If one child supervisor leaves the network, then all his peers have to be re-assigned to another child supervisor. Let say, S1 leaves the network, then P1, P2 and P3 have to find a new closet child supervisor (e.g. S2) to make sure all new interval information can be received by the nucleus.

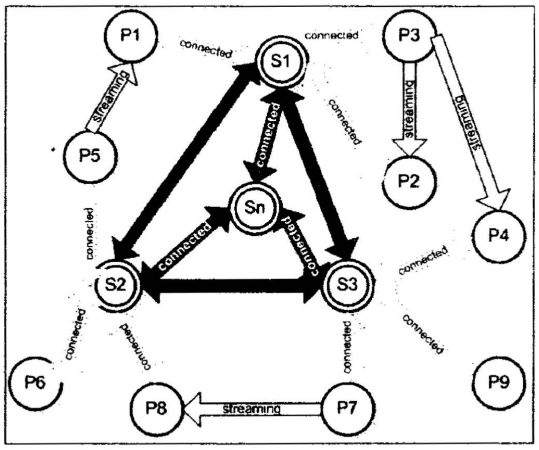

Figure 1. Hybrid Peer-to-Peer Overlay Network Topology

Each peer is not connected directly to another peer, and their overlay network topology is unstructured. When a peer is looking for a file, he/she queries the child supervisor(s), whether the data is available or not, the child supervisor will respond with a list of peers that contains the requested file. After that, if any file is listed, the peer can directly download the file from the source. Now, we will continue to discuss on the availability, searching, matching and location of files in hybrid P2P video streaming.

3.2 Availability And Accessibility Of File

There are two policies which the peer should meet before the peer can share his content. The first policy is that the peer should be allowed to share his files if he/she is in online mode. The second is the peer can share the contents if the peer has the current offer rate above 128 kbps.

User needs to login into the network first, and start to push a metadata of his/her shared files to the child supervisor through TCP. If in Yahoo!, human editors provide the metadata about the web sites, but in our case the user himself/herself should be able to provide the correct information about his/her shared content. If anything changes in his/her contents, then the system should be able to upload an amended metadata to the child supervisor. When a peer leaves the network, the child supervisor should be able to delete all metadata related to that peer. Hence, only the metadata owned by online peers should be visible to other peers. Furthermore, it will increase the speed of the search in the database, and it satisfies the user's need to watch it spontaneously after the file is spotted. However, it has to be reminded that this is streaming, not downloading.

What sort of data in metadata should be pushed to the child supervisor? We propose to transmit the explicit file name, title, synopsis, checksum and extension of file such as avi, mpeg, mp3, etc. We will use Md5sum [34] technique to hash video content to get the checksum value of 32 characters.

3.3 Lookup And Matching

This service attempts to satisfy users' queries primarily by looking for occurrence on metadata. Metadata that includes keywords are considered good matches. Lorenzo Thione in [31] said that, "We have to train ourselves, out of necessity, to translate our needs into keywords as successfully as we can". The proposed system can guarantee to locate data from *O(1)* to *O(n)* step via child supervisor, where n is the number of child supervisors. In theory, the system should be able to reduce the latency lookup services, search path length and peer load as compared to other P2P applications as exhibited in Table 1. We have fixed the number of records that should be answered back to requesting peer, and in this case, X is used as a fixed number. As we can see in Fig. 2, when the peer starts to search and send the query to his/her child supervisor, the child supervisor will match the query with metadata. If the number of record found does not reach X or no record is found, then the originator child will start to ask other child supervisors using an identifier table. When the originator child supervisor has the X records or received replies from all child supervisors, the originator child supervisor will push the result to the requesting peers. Each record should consist of a title, checksum value, size file, IP address, port number and rank. All records will be sorted by rank, current offer rate and balance of limitation.

There are many ways to perform retrieval information. We propose to use the exact and truncation keyword, extension file, checksum value and video/audio base ontology repository.

3.3.1 Keyword Search

User can choose to use exact word or truncation. The search will retrieve information that contains the exact keywords used anywhere in a metadata, such as in the file name, title and synopsis. Truncation in a search is done by using characters (e.g. % or *) to retrieve several words that begin with the same word. This method is effective when the words that have different spelling or different suffixes. In other words, truncation helps to retrieve more results or broaden the search.

Exact word

```
SELECT *
FROM metadata_table
WHERE file_name='keyword' OR file_title='keyword' OR
file_synopsis='keyword'
```

Truncation

```
SELECT *
FROM metadata_table
WHERE file_name LIKE '%keyword%' OR file_title LIKE
'%keyword%' OR file_synopsis LIKE '%keyword%'
```

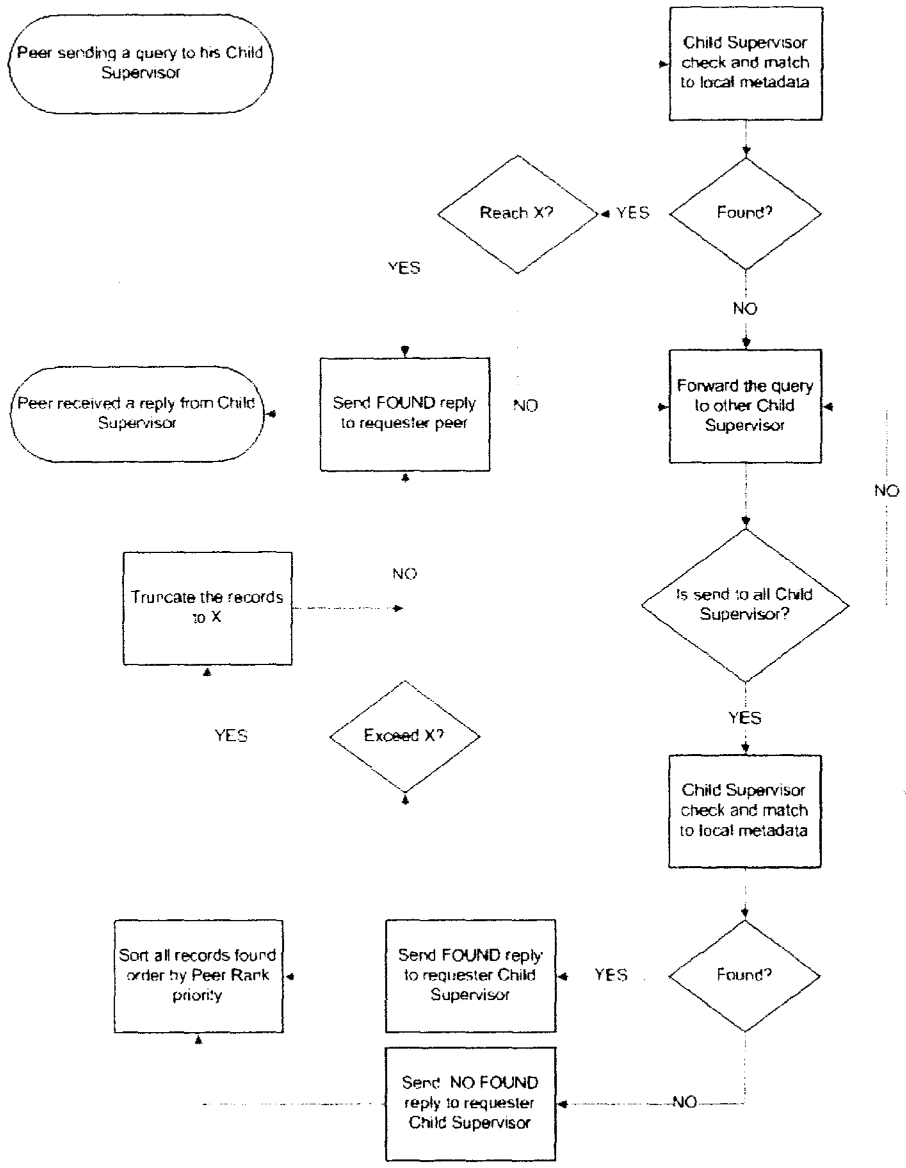
Peer sending a query to his Child Supervisor
Child Supervisor check and match to local metadata
Found?
YES
Reach X?
YES
NO
Send FOUND reply to requester peer
NO
Peer received a reply from Child Supervisor
Forward the query to other Child Supervisor
NO
Truncate the records to X
NO
Is send to all Child Supervisor?
YES
Exceed X?
YES
Child Supervisor check and match to local metadata
Sort all records found order by Peer Rank priority
Send FOUND reply to requester Child Supervisor
YES
Found?
Send NO FOUND reply to requester Child Supervisor
NO

Figure 2: Process flow in *O(n)* step searching

3.3.2 Extension Of File

We also merge a keyword search with an extension file (e.g. avi, mpeg, mp3) by using Boolean operator, such as AND, so that we can narrow the search and potentially get a short list of results with a better accuracy.

```
SELECT *
FROM metadata_table
WHERE file_name='keyword' OR file_title='keyword' OR
file_synopsis='keyword'
AND file_ext='extension_keyword'
```

3.3.3 Checksum

We compare the MD5sum hash of targeted file, to the known hash of the file we are searching in metadata. If the values match, we have an exact copy of the original file. This query mechanism will be triggered by two options. The first mechanism is triggered by the peer itself. Once the peer gets the result set from keyword search, he can continue to search again using checksum value. By doing this, the peer will receive another set of result which provides the location and file with identical checksum value. Second, the system itself will do the automatically re-query when the peer starts the Real Time Protocol (RTP) session. The system will locate the exact content using checksum value. This process will be done at occurrence in t time. It also depends on the duration of the movie. The result set will be used as a new standby peer list, and it will replace the previous standby peer record. If there is a problem in active sender, the peer with the highest order in standby peer list will take over.

```
SELECT *
FROM metadata_table
WHERE file_checksum='checksum_keyword'
```

3.3.4 Ontology Video Repository

Can the exact or truncation keyword search be considered as the best method for matching? What about using the ontology matching? Ontology is defined as shared conceptualization of domain. They capture background knowledge by providing relevant concepts and relations. The role of ontology is to provide in tensional knowledge in a machine process able way, thus enables automatic aggregation, and the proactive use of distributed data sources [8].

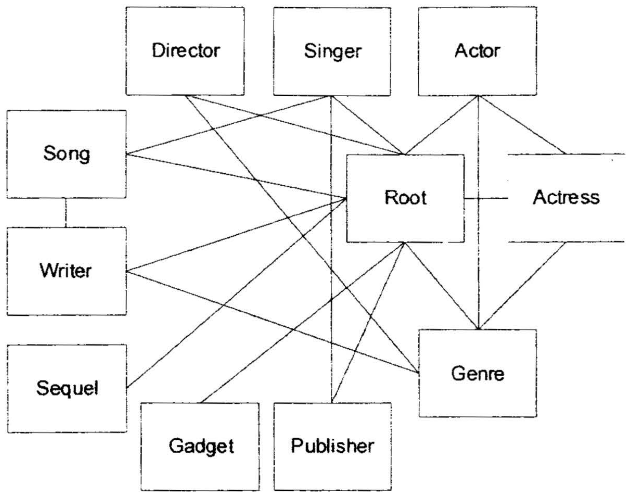

Figure 3: Base ontology for video/audio domain

In our case, we create base ontology referring to video/audio domain. Fig. 3 shows that the root "Title of video/audio" has broad relationship (or attributes) with Director, Actor, Actress, Singer, Writer, Publisher, Sequel, Gadget, Scene, Song and Genre. Using ontology matching, the final result may be different from the keyword search. First, we need to create the ontology repository or pool. Then, the data will be able to be populated by peers themselves. We have to make an assumption that all instances and relation words are well defined and constructed correctly in ontology repository.

We merge the base ontology with keyword and extension. When a peer sends the keyword search (and pick the base ontology as a selector criteria), the child supervisor will attempt to match (step 1) the keyword with the root, and retrieve the attribute of the root. Secondly, it will match (step 2) the keyword with the attribute, and get the root. After that the child will match (step 3) the keyword, attribute and root with the metadata. Because we have limited records to be displayed, the first priority is the records obtained from the keyword and attribute search.

Step 1: retrieve an attribute of the root

SELECT *attribute* AS *keyword_1*
FROM *ontology_table*
WHERE *root='keyword'*

Step 2: retrieve a root

SELECT *root* AS *keyword_2*

```
        FROM ontology_table
        WHERE attribute='keyword'

Step 3: retrieve files using keyword, keyword_1 and keyword_2
        SELECT *
        FROM metadata_table
        WHERE file_name='keyword' OR file_title='keyword' OR
        file_synopsis='keyword'
        OR file_name='keyword_1' OR file_title='keyword_1' OR
        file_synopsis='keyword_1'
        OR file_name='keyword_2' OR file_title='keyword_2' OR
        file_synopsis='keyword_2'
```

3.4 Delegation Rule

We believe that the selection of the best peer to serve a streaming session is vital in providing good quality video streaming. For that reason, we introduce *Six Rule* in our proposed design. This rule also can avoid delegating popular file to peers who have low resources, such as available bandwidth, CPU power, memory, etc, in order to avoid network saturation and flash crowd. The first rule is the availability of peer himself/herself. The offline peer should be removed from the result set so that the records returned to requester peer will consist of online peers. The second rule is a peer's rank. We classify the peer into eleven groups {0,1,2,3,4,5,6,7,8,9,10}, that are from *bad peer* to *good peer*. We choose six factors to perform a ranking, which are bandwidth, CPU power, packet loss rate, uptime, memory and storage. We will not discuss in detail on how the peer ranking is done. However, if you need further details, you may refer to [1]. The third rule is a balance of limitation. From the peer's rank, we set the limitation of each peer to serve other peer(s). For example; if a peer's rank is zero, his value of his limitation is zero. This means that the peer cannot stream his shared content. On the other hand, if a peer is ranked as number three, he, thus, has a limitation value of five. In this situation, he/she can only stream at the most to five peers concurrently. If he/she at present is serving two peers, he has another three spaces to let other peers to connect to him/her. The fourth rule is the current offer rate or Internet connection speed. Once a peer joins the P2P network, his/her current offer rate will be compared to the minimum offer rate system which is fixed to 128 kbps. If his/her current offer rate is more than 128 kbps, he/she will be allowed to share the content. The fifth rule is that the returning result is limited up to X number. We still have not decided how many files that should be returned, but we believe that it will give benefit to requester and the file owner as well. Finally, the sixth rule is the rights. We want to protect the legal distribution for video/audio content amongst the P2P network so that illegal copies or distribution can be minimized and infringement of rights will not occur in our proposed design.

1.0 SIMULATION AND RESULTS

We have done a simulation to compare the path length between the proposed designs, Chord, FreeNet, BitTorrent and Gnutella. Path length is the steps needed between two peers of the network. It is defined as the hops needed from a querying peer to another peer with the desired information. We used P2PNetSim tool and JAVA to write an algorithm for each protocol. We generate 8,192 peers and set 100 time step. Time step is a loop cycle, so 100 time step means 100 loops. Each peer has a capability to start a query with a probability of 0.10 for every time step. Each query has a probability of 0.50 to get an answer (data found) from one peer. We make an assumption that every query contains a valid keyword, checksum, etc and should be able to match any file in the P2P network. If there is no reply (not found) for a query, it is because of the routing step, TTL and network topology. We deployed 13 simplified algorithms in eight simulators to represent 13 different cases in one scenario, and each simulator runs 1024 peers concurrently. We consider 13 cases which are

a) Gnutella without TTL and has 10 neighbours;
b) Gnutella with TTL=15 and has 10 neighbours;
c) Gnutella with UltraPeer and has 10 neighbours;
d) Gnutella with UltraPeer, TTL=15 and has 10 neighbours;
e) Chord without TTL;
f) FreeNet without TTL and has 2 neighbours;
g) FreeNet with TTL=15 and has 2 neighbours;
h) BitTorrent with 1 Supernova and 10 Trackers (T);
i) BitTorrent with 1 Supernova and 25 Trackers (T);
j) BitTorrent with 1 Supernova and 50 Trackers (T);
k) Proposed design with 10 Child Supervisor (CS);
l) Proposed design with 25 Child Supervisor (CS); and
m) Proposed design with 50 Child Supervisor (CS)

The simulation result shows that the hybrid peer-to-peer model is suitable for searching in the P2P network. From Fig. 4, we can see that the BitTorrent, FreeNet and the proposed design have a lower average path length as compared to Chord and Gnutella. So, we can predict that the searching latency for the proposed design, FreeNet and BitTorrent should be lower than Chord and Gnutella. Fig. 5 shows that the proposed design and BitTorrent have a higher accessibility in comparison to FreeNet. FreeNet with or without TTL=15 has a possibility of 62% to find and locate the file, whereas the BitTorrent and the proposed design is 100%. Furthermore, the proposed design and BitTorrent have a small fluctuation in path length as compared to Chord, FreeNet and Gnutella. Therefore, the searching latency is likely constant and predictable.

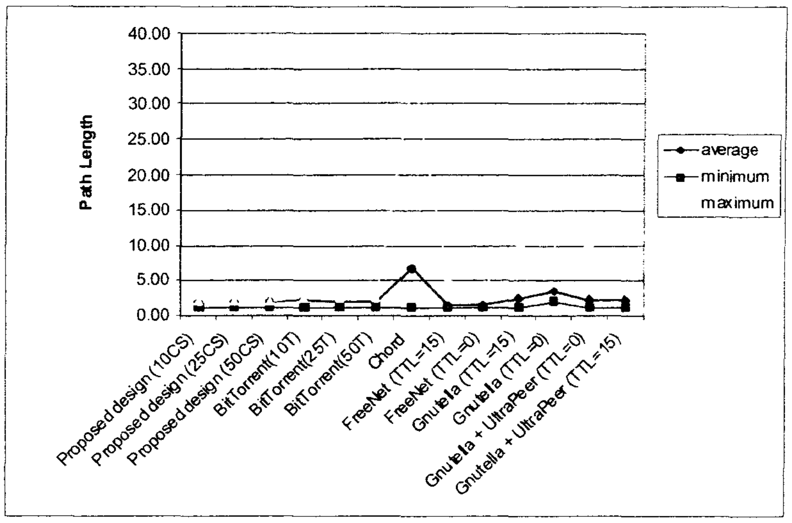

Figure 4. Comparison path length for various P2P applications

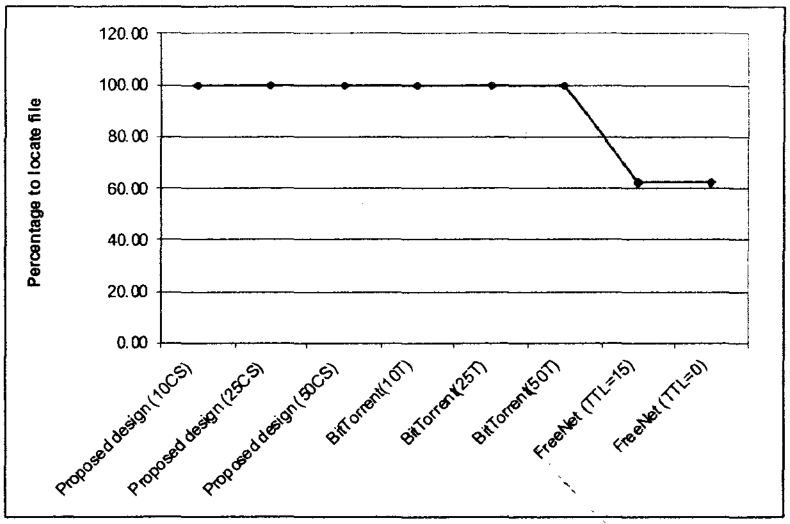

Figure 5. File accessibility for various P2P applications

5.0 CONCLUSION

This paper presents the hybrid peer-to-peer model for video streaming which concentrates on the availability, lookup and retrieval of information about the media contents. We hypothesize that, by using a server-client model to do a search and retrieve information, we can reduce the latency lookup services, path length, peer load and network traffic. We presented that, the proposed design has a small range path length from 1.00 to 2.00 and indirectly it shows that the searching latency is lower and constant as compared to FreeNet, Chord and Gnutella. Our future work is to do a simulation and make a comparison on peer load (incoming and outgoing messages) and network traffic with Gnutella, FreeNet, Chord and BitTorrent. The next plan is to analyze the result set based on the proposed design, and compare it to a typical searching, using server-client model without the exact and truncation keyword, extension file, checksum value and base ontology repository.

REFERENCES

[1] Ahmad Tajuddin, S., Nanna Suryana, H., and Mat Kamil, A., 2007, "The Ranking Peer for Hybrid Peer-to-Peer Real Time Video Streaming", *In Proceeding of IEEE ICACT2007*, Feb 12-14, 2007.

[2] Antony, R., and Peter, D., 2001, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems", *In Proceeding of the 18th IFIP/ aCM International Conference on Distributed Systems Platforms (Middleware 2001).*

[3] Ben Y. Zhao et al., 2004, "Tapestry: A Resilient Global-Scale Overlay for Service Deployment", *IEEE Journal on Selected Areas in Communication*, Vol. 22, No. 1, pg. 41-53.

[4] Chris, R., and Christian, S., 2004, "Guarantee Broadcasting Using SPON: Supervised P2P Overlay Network", In International Zurich Seminar on Communications (IZS), February 18-20, 2004.

[5] Dahlia, M., Moni, N., and David, R., 2002, "Viceroy: A Scalable and Dynamic Emulation of the Butterfly", *In Proceeding 21st ACM Symposium on Principles of Distributed Computing (PODC 2002),* July 21-24.

[6] Denny, V., 2006, "Ontology Evaluation for the Web", Extended Abstract – PhD proposal, Institute AIFB, University of Karlsruhe, Germany, 2006.

[7] Dongyan Xu et al., 2002, "On Peer-to-Peer Media Streaming", *In Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

[8] Duc,A. T., Kien, A. H., and Tai D., 2003, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming", *IEEE INFOCOM*, 2003.

[9] Ian, C., et al., 2000, "Freenet: A Distributed Anonymous Information Storage and Retrieval System", *In Proceeding of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA.

[10] Ion Stoica et el., 2001, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *In Proceeding of SIGCOMM*, p.p. 161-171, August 27-31, 2001.

[11] Pouwelse, J.A, et al., 2005, "A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System", *In 4th International Workshop on Peer-to-Peer Systems* (IPTPS), February 2005.

[12] Kishore, K., and Christian, S., 2005, "Supervised Peer-to-Peer Systems", *In proceeding of the 8th International Symposium on Parallel Archictecture, Algorithms and Network (ISPAN'05).*

[13] Krishna, P. Gummadi, et al., 2003, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload", *In Proceeding of SOSP'03,* New York, October 19–22, 2003.

[14] Garces-Erice, L. et al., 2003, "Hierarchical Peer-to-Peer Systems", *In Proc. of the International Conference on Parallel and Distributed Computing,* Klagenfurt, Austria.

[15] Li, X., Zhenyun, Z., and Yunhao, L., 2005, "Dynamic Layer Management in Superpeer Architecture", *In IEEE Transactions on Parallel and Distributed Systems*, pp. 1078-1091, Vol. 16, No. 11, November 2005.

[16] Meng, Z. et al., 2005, "Large-Scale Live Media Streaming over Peer-to-Peer Networks through Global Internet", *In Proceeding of P2PMM'05*, Singapore, November 2005.

[17] Michael, K., Eng, K. L., and Xiaoming, Z., 2005, "A Case for Lightweight SuperPeer Topologies", *In KiVS Kurzbeitrage und Workshop*, Germany, pp. 185-188, 2005.

[18] Michael Kleis, Eng Keong Lua and Xiaoming Zhou,, "Hierarchical Peer-to-Peer Networks using Lightweight SuperPeer Topologies" *In Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)*, Spain, 27-30 June 2005.

[19] Miguel Castrol et al., "SplitStream: High-Bandwidth Multicast in Cooperative Environments", *In Proceeding of SOSP'03,* New York, October 19–22, 2003.

[20] Mohamed Hefeeda et al., "CollectCast: A Peer-to-Peer Service for Media Streaming", *ACM/Springer Multimedia Systems Journal*, October 2003

[21] Peter Maymounkov and David Mazieres, "Kademlia: A Peer-to-peer Information System based on the XOR Metric", *In Proceeding of Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, March 7-8, 2002.

[22] Steven J. Vaughan-Nichols, "Researchers Make Web Searches More Intelligent", *Computer Magazine, IEEE Computer Society*, p.p. 16-18, December 2006.

[23] Supernova.org for BitTorrent central server, available at http:// www.supernova.org

[24] Sylvia Ratnasamy et al., "A Scalable Content-Addressable Network", *In Proceeding of SIGCOMM*, p.p. 161-171, August 27-31, 2001.

[25] Xuxian Jiang et al., "Gnustream: A P2p Media Streaming System Prototype", *In Proceedings of IEEE International Conference on Multimedia & Expo (ICME 2003)*, July 2003

[26] Ying Cai, Zhan Chen and Wallapak Tavanapong, "Video Management in Peer-to-Peer Systems" *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, 2005

[27] Using MD5 Checksums, available at http://www.openoffice.org/dev_docs/using_md5sums.html

[28] Napster website, available at http://www.napster.com

[29] Patrick Kirk, "Gnutella: file sharing and distribution network", 2003, available at http://rfc-gnutella.sourceforge.net/index.html

[30] MLDonkey Project Wiki website, available at http://mldonkey.sourceforge.net/Main_Page

[31] KaAzA website, available at http://www.kazaa.com

[32] eDonkey2000 website, available at http://www.edonkey2000.com

[33] Gnucleus: An Open-Source Gnutella Client, available aat http://www.gnucleus.com

[34] Gnutella Protocol Development, available at http://rfc-gnutella.sourceforge.net/developer/testing/Ultrapeers_1.0.html

[35] Anurag Singla and Christopher Rohrs, "Ultrapeers: Another Step Towards Gnutella Scalability" Lime Wire LLC, Nov 26, 2002, available at http://rfc-gnutella.sourceforge.net/src/Ultrapeers_1.0.html

[36] BitTorrent website, available at http://www.bittorrent.com