ELG 5121 / CSI7631
Multimedia Communications
Ottawa, Ontario, Canada, Fall 2005

Mobile P2P: Data Retrieval and Caching

Dewan Tanvir Ahmed[1], Abdelfettah Diabi[2]
[1]dahmed@discover.uottawa.ca
[2]adiabi@discover.uottawa.ca

Abstract *– Peer-to-peer is a popular distributed overlay network due to its easy deployable property. Mobile computing and communications are improving day by day. With this advancement in technologies, mobile p2p has revealed its attractions. But resource constraints such as limited bandwidth, power and memory along with nodes mobility make it challenging to build a scalable and a robust mobile p2p network. Content discovery and locating are crucial that affects the performance of the system. Data caching and adopting dynamic topology of the mobile p2p systems are important to accelerate the effectiveness the system. In this project we discuss these issues to enhance the applicability of mobile p2p systems and give some guidelines to the researchers in this field.*

Keywords *– Mobile, peer-to-peer, resource locating, caching.*

I. INTRODUCTION

Peer-to-Peer is a distributed network architecture where each node shares its recourses such as processing power, bandwidth, storage capacity, etc. Usually nodes in peer-to-peer system are loosely coupled with each other and play an equal role (client/server) in the system. Moreover nodes are autonomous that makes it difficult to construct a robust peer-to-peer system. In a pure peer-to-peer network, there is no notion of clients or servers, but only equal peer nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network [1]. This model of network architecture differs from the client-server architecture where communication is usually to and from a server. FTP is an example of non peer-to-peer file transfer system where the client and server programs are quite distinct, and the clients initiate the download/uploads and the servers react to and satisfy these requests. In a peer-to-peer system the removal of any arbitrary chosen node from the network does significantly affect the offered service from the system while in client/server system it does not make in any sense without a server. Peer-to-Peer system has many significant applications ranging from file sharing to grid computing. Digital library can be built up on top of peer-to-peer network as well.

Mobile peer-to-peer (MP2P) system faces more constraints as compared to wired p2p system. Some problems in this network are the scarcity of bandwidth, short lifetime of the node due to power constraint, dynamic topology caused by the mobility of nodes. Due to these constraints, MP2P give rise to hard challenges to the researchers to research on routing, content location discovery, data retrieval and caching policy etc.

The rest of the paper is organized as follows. Section II describes wired p2p system. Mobile p2p architecture and mobile content addressable networks are mention in section III and IV respectively. Regional content searching and caching is covered in section V. Section VI describes dynamic indices in mobile p2p system. Finally in section VII we draw the conclusion.

II. WIRED P2P SYSTEM

Peer-to-Peer is a distributed overlay network. There should be no central node in p2p at least in theory. But in reality some of the p2p systems depend on central server or directory server. Some other p2p systems classify peers as super peers and client peers. For the sake of proper operation of p2p system, researchers have to sort out some problems such as efficient content searching, fair participation, poisoning attacks, denial of service attacks, spamming attack, identity attacks, and many others. There are many p2p systems. Among them Napster and Gnutella are described below.

A. Napster

Napster was the first widely used peer-to-peer music sharing service. This music file sharing system was created by Shawn Fanning in 1999. Napster technology allowed music lovers to easily share MP3 format song files with each other, thus leading to the music industry's accusations of massive copyright violations. “Although the original service was shut down by court order, it paved the way for decentralized P2P file-sharing programs, which have been much harder to control” [2].

Figure 1: Napster architecture

In Napster, there is directory server, see figure 1, which maintains a database of the content location. In ordering to

This research was part of the course Multimedia Communications (Fall 2005) given by Prof. A. EL Saddik

download the music file client has to know the location of the intended file. This directory server is responsible to provide this information. This is like a content location server. Obviously peer must provide this information to directory server to make this file available to its peers. Napster works in three steps,

1. Content search request
2. Search response
3. Download

When a peer would like to download a file he should make a content search request to the directory server. Upon receiving the request server checks its database and returns a response to the requester. If the response is positive, peer will establish a connection to the peer holding the file and download it from there.

Napster is not a pure p2p system as it depends on a central node, e.g. the directory server. Moreover it is not scalable. This architecture would be vulnerable at central node with a large number of peers. That is there is a chance of one point failure. But its key feature is its simplicity.

Broadband technology has made file sharing even more widespread. Increasing download speed implies distribution of the entire movies and other large files are now possible. Now it seems that it is quite impossible to interdict the emerging and cryptographically strong third generation of P2P protocols [2]. Thus, Shawn Fanning opened a Pandora's Box.

B. Gnutella

Gnutella is a pure peer-to-peer file sharing network that operates without a central server or directory server. According to the file sharing website Slyck.com, Gnutella is the fourth-most-popular file sharing network in the Internet, following eDonkey2000, BitTorrent, and FastTrack. Gnutella is thought to host on average approximately 1.8 million users, although around 400,000-500,000 at one time [3].

Different approaches have been developed to bootstrap client software. Usually client learns about other active nodes thorough IRC or Gwebcache sites on the web. Bootstrapped node tries to make a connection to one of the possible active nodes. This active node, upon connected, will return his own active connected nodes to the bootstrapped node. This learning process continues until active connections reach to a pre-assigned number. In this way, new node builds its own topology to work with other peer nodes to share files or something else.

When a node wants to do a search, it sends the request to all of its active connected nodes. Figure 2 is an example of Gnutella searching process. These active connected nodes will forward the request to their descendants. In this way it finds all the nodes in the network and hopes that the request will turn up a positive result. After that, client negotiates a connection and downloads the file from there. If there is multiple copy of the same file it can apply swarm downloading technique, partially from different seeders in parallel. This swarm downloading accelerates downloading rate.

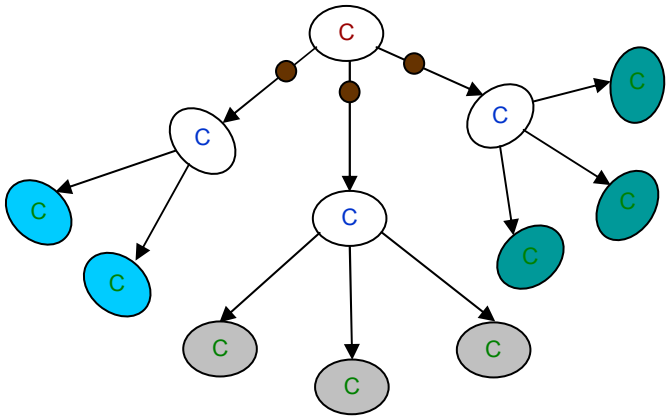

Figure 2: Gnutella content searching process

The GDF (Gnutella Developer Forum) currently leads the development of the Gnutella protocol. Many protocol extensions have been made and some others are being developed by the software vendors and free Gnutella developers of the GDF. These extensions include intelligent query routing, trust vector for fairness, checksums for correctness, file transfers via UDP, XML metadata and parallel downloading in slices (swarming), etc [3].

III. MOBILE P2P ARCHITECTURE

Peer-to-Peer can be benefited from some inherent features such as scalability, dynamic adaptation, permanency, fault tolerance, anonymity and self configuration, etc. But effectiveness of these features lies on how well system assigns peer names and locating the peers and the data items. The organization of p2p topology and efficient data placement on peers require some strategy. Hashing is the most popular method to do this 4], [5], [6]. Hashing assigns unique key to the peers and data items in the same space to control the topology and data retrieval. This technique protects anonymity of nodes and data items. These sorts of architectures are called hash-based structured peer-to-peer (HS-P2P) systems. In the following sub sections we explain a mobile p2p system named Bristle.

A. Bristle: An Overview

The general hash strategy is to store a data item with a hash value *k* at a node whose hash value is the closest to *k*. So when there is a request to that data item, request message is forwarded to the intermediate peers whose hash values are closer and closer to *k*. In this architecture each node maintains some state information for routing. This aids a local node to locate a node on the way to the destination. A state-pair has a form like *<hash key, network address>*. A scalable and robust p2p system depends on sophisticated state management which must be distributed [7].

Bristle recognizes participating nodes either as mobile nodes or stationary nodes that follows Tornado p2p architecture [4]. Stationary node provides location information of mobile nodes to an interested node. Bristle requires $O(\log N)^2$ hops to send a message from source to the destination, where N is the number of nodes in the system [7].

B. Bristle: Packet Forwarding

A simple Bristle architecture is given in figure 3. This architecture consists of two hash-based structured peer-to-peer systems. One of them is mobile layer and other is stationary layer. As the name suggests, in mobile layer, nodes are allowed to move by changing their attachment points while in stationary layer, nodes are fixed in the location. There is no restriction of HS-P2P on stationary layer. It may use any hash based p2p, for example, Chord [8], CAN [5], Pastry [6], Tapestry [9], Tornado [2], etc.

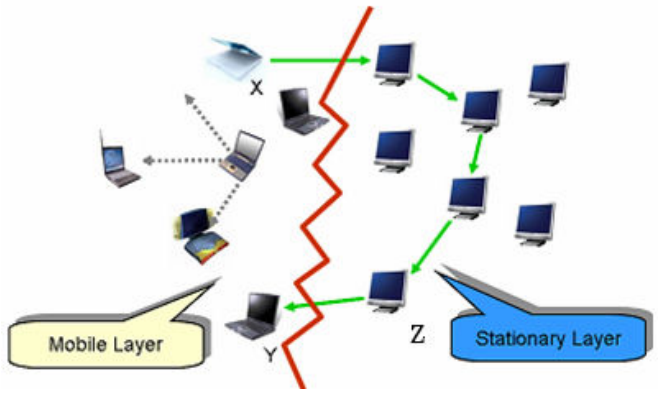

Figure 3: The Bristle Architecture

Say, node X and Y are in Mobile layer. X would like to forward a data packet to node Y as it is the next node in the routing path. If X knows the network address of Y, in that case, packet is directly forwarded to Y. But, if the network address of Y is unknown to X, it needs to resolve the network address first and then forwards the packet to Y. In order to learn the network address, X simply pushes the packet to the stationary layer. For the sake of proper operation each and every mobile node should publish or validate its state information to the new attachment point. The packet is being forwarded to the node Z, provided that Z knows network address of Y. Mobile layer node is allowed to distribute update information to the other mobile nodes. Since most of the HS-P2P systems distribute a node's state to $O(\log N)$ nodes in the system indicates that Bristle is scalable [7].

C. Bristle: Routing

The routing algorithm for Bristle is given in figure 4 [7]. Here a node's state-pair information is presented as *<hash key, network address>*. It may happen that for a particular node, say X, has sate-pair like *<k, null>*. Here null refers to invalid or unknown address. So it then seeks the help of stationary layer by sending a discovery message with a hash key *k*. This request is being forwarded within the stationary layer until it can be solved. One of the nodes in stationary layer, let Z, determines the network address of hash value *k*. Node Z then sends the message to Y, destination node, to retrieve the data. Z also returns the complete state-pair information, *<k, Address(Y)>* back to the requester on the reverse path.

```
_route (node i, key j, payload d)
//Does there exist a node closer to the desitnation key j ?
if (∃p ∈ state[i] such that p.key is closer to j)
  // Is the network address valid?
  if (p.addr = null or p.addr = invalid)
    // Resolve the address for p.key and forward packets
    // by a node in location management layer
    p.addr = _discovery(p.key);
  else
  // node i forwards packets to node p.key
  _forward(p.addr, j, d);
```

Figure 4: Bristle Routing Algorithm

Bristle uses *location advertisement tree* (LDT) rooted at every mobile node. LDT supports multicast communication for state advertisement. Moreover, Bristle supports four operation, register, update, join and leave, in order to manage mobile nodes.

D. Bristle: Features

Bristle is a HS-P2P system that allows nodes to change their attachment point without re-assigning new states. With the help of mobile IP old state can be retained. It enhances it scalability, reliability and performance as well.

IV. MOBILE CONTENT ADDRESSABLE NETWORK

Limited power, limited bandwidth and lower computing capability are the major concerns in the designing of mobile p2p system. Thus for every operation, such as device discovery, content location, usage of broadcast may result in poor performance because of resource constraints. But the characteristics of mobile devices, such as variable connectivity, location dependency and scarcity of resources make the design and implementation of client- server architecture really difficult and even would be inefficient due to inherent broadcast property of mobile devices. M-CAN (a lookup protocol) is an extension of Content Addressable Network [10]. Instead of file transferring and caching, M-CAN applies registering and grouping policy to avoid bandwidth reserving [11]. In literature, there are some p2p lookup protocols.

CDP, Centralized Directory Protocol), maintains a central server. The peer who wants to share a file publishes it to the central server. If a peer would like to use a shared content, it sends a request to the central server. Upon receiving the request server checks it directory and returns the best result back to the peer. After that, file accessing can be handled between the two peers directly. Figure 5 is a snapshot of CDP. Napster works on this concept. The central server would be point of bottleneck with an increased number of peers and hard to scale.

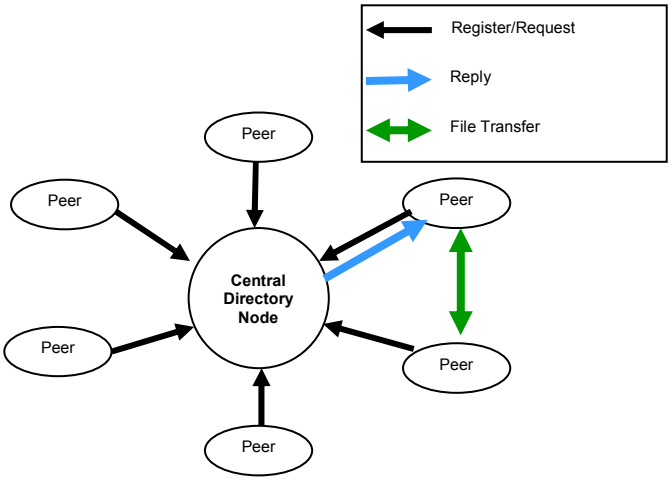

Figure 5: Centralized directory protocol

In flooded request protocol, every peer has the information of its neighbors. Publication of information is absent here. Thus every peer's request is flooded in the system until it is reached to it maximum hop value or it is answered (figure 6). But it is not efficient in mobile environment even though it is very popular in wired p2p system like Gnutella as it consumes a lot of bandwidth and power.

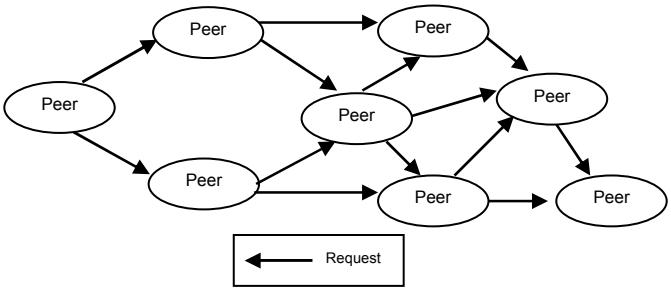

Figure 6: Flooded request protocol

On the other hand, in document routing protocol each peer is assigned with an ID chosen from a random pool. Moreover each peer has the content information of the neighbor peers. One of the key features of this approach lies on the document or content distribution. Here IDs are given to shared files based on the document it contains or its title. Then this file is stored on a peer that has an ID closest to the document ID. The copy of that file is also left on all the peers on the way to the targeted node. So the request of the file is being routed to the node that has an ID close to the file ID. Figure 7 illustrates document routing protocol. Even though the algorithm applied here seems good for wired system in terms of scalability but it is not suitable for mobile p2p system. This is because, due to storage and bandwidth limitations it is impractical to store the file on all the peers on the way to targeted node and wastages too much memory to be deployable in mobile p2p system.

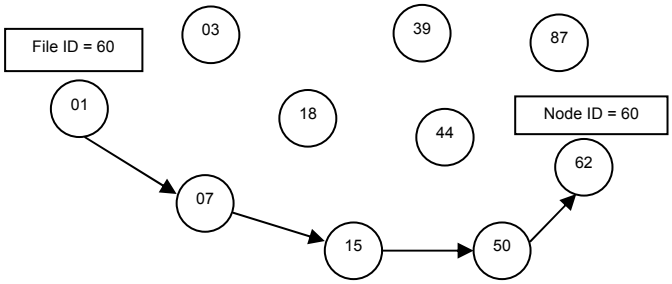

Figure 7: Document routing protocol

A. M-CAN: Hierarchy and Peer Community

M-CAN uses register approach to manage resource. It assigns an ID to each content file according to its contents and title before publishing to the super nodes. M-CAN consists of a collection of super nodes and ordinary nodes. Super nodes are selected considering their stronger computing capabilities, higher storage resources. These super nodes are responsible to handle a set of contents in fact a range of IDs. In this system, each node registers to some super nodes according to the IDs of the shared files [11].

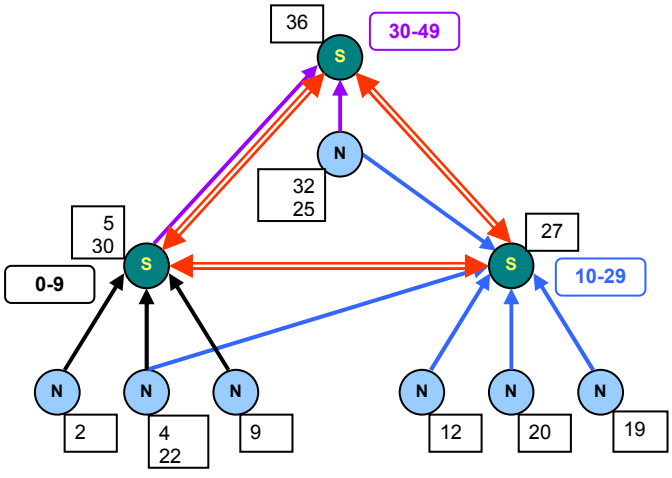

Figure 8: M-CAN: Group formation

M-CAN group formation process is shown in the figure 8. Here, S and N stand for super node and ordinary node respectively. In this figure there are three super nodes. They cover the files ranges from 0-9, 10-29, and 30-49 in three disjoint set. For example, if an ordinary node has two files with ID 4 and 22, it must register them onto two different super nodes according to the ID range. It may happen that a

super node has to register to another super node due to its shard file ID. These super nodes communicate with each other for the management of the M-CAN system.

M-CAN system composed of some groups. Groups are allowed to communication with each other through the special communicating nodes that are at the edges of the group. This group communication indirectly helps the system to scale. The structure of M-CAN system is depicted in the figure 9.

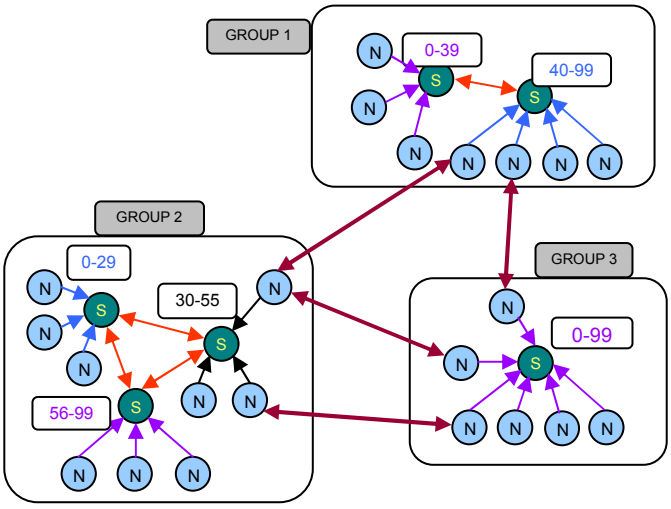

Figure 9: M-CAN: The whole peer community

B. M-CAN: A Lookup protocol

In M-CAN every node registers on one or more super nodes. When a node would like to get a file, it sends the request to its registered super node. If the requested file ID is within its range and actually exists there, on that case he will get a positive response. Otherwise request is directed to other the super node in the same group or other group. The request is forwarded to the super node of the same group if targeted ID is not on the registered super node. The request may require to be forwarded on other group if it is not in home group. Inter group requests are transported through the communicating nodes. If the requested ID is on other group, response is redirected to the original requester. After that, file accessing is conducted either directly or with the help of other intermediate nodes.

C. M-CAN: Performance issues and analysis

In order to ensure fairness and limit the excessive burden on a super node, M-CAN allows a super node to include another super node in the system if number of connections to ordinary nodes reaches to a pre-assigned value. In that case, range of file IDs will be distributed on them. Figure 10 gives an example of split operation. A departing super node triggers another super node to extend ID covering range. Due to space limitations we discard it explaining with an example.

M-CAN uses registering method to reduce bandwidth and storage capacity consumption. This is because file is not forward to other nodes to store there. Moreover it doesn't consider the caching to lessen the burden on storage capacity of the devices. But users have to know the file ID before sending the request. As the size of the system increases search latency also increases. In that sense we can say that it in not well scalable.

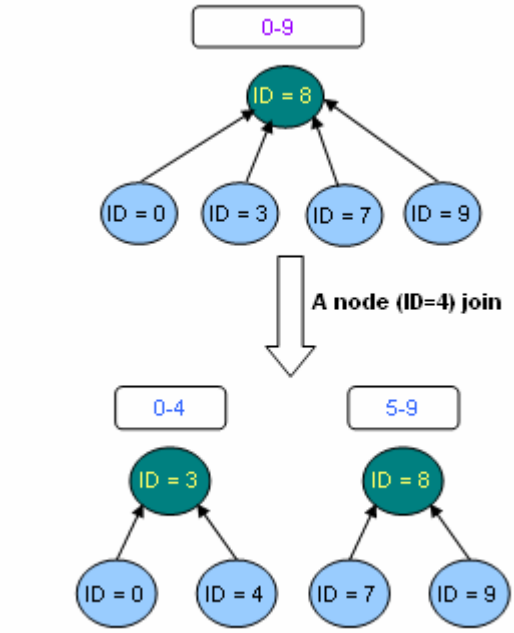

Figure 10: Super node split operation

V. COOPERATIVE CACHING IN MOBILE P2P

Caching improves data retrieval performance in mobile peer-to-peer system [12]. Because caching can save computing power and bandwidth as long as you have enough memory space. In cooperative caching scheme peers can access data from their neighbors. So, to accelerate the overall performance researchers are now focusing on designing new caching schemes suitable for mobile environment. Cooperative caching schemes have been studied for wire environment to improve the performance of web caching or web proxies [13] [14]. *Proximity Region for Caching in Cooperative* (PReCinCt) is a novel caching scheme in MP2P environment. It caches data among a set of peers based on some criteria such as popularity of the content, size of the content and distance to the owner of the item [15]. In the following subsections we describe PReCinCt in detail.

A. PReCinCt: Locating and Caching

PReCinCt cooperative caching divides the whole network topology in multiple geographical regions. Each region is responsible to deal with a set of keys. This hash based method uses a hash function $h(k_i) = L_j$ to locate the region

L_j where this key k_i resides. Actually this is a mapping from key to location. We can say, L_j is the home region R_j for the key k_i. So when a peer is looking for an item, it first uses hash function with the key to determine its home region R_j. This request in then forwarded to that home regions using geographical routing protocol like GPSR [16]. Figure 11 gives an example of PReCinCt. Localized flooding is used after the request has been reached in home region. So the performance of PReCinCt totally depends on how well hash function maps keys to regions.

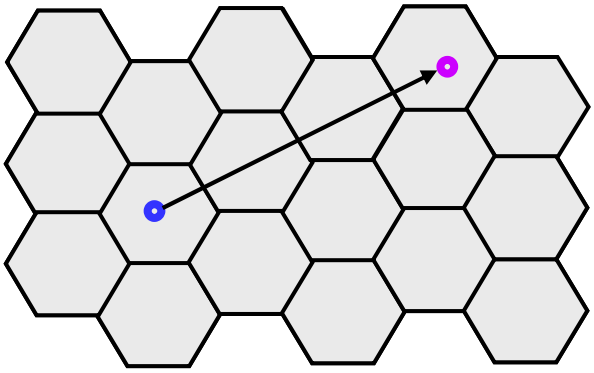

Figure 11: Content Locating in PReCinCt using hash function

After getting the data item, PReCinCt decides whether it will cache the item or not. The caching policy depends on three parameters namely popularity of the data item in the region, distance between the requesting and responding peer and size of the item. They named it *Greedy Dual Least Distance* method (GD-LD). The evaluation function is given below.

$$U = w_r \times ac_i + w_d \times req\_dst + w_s \times \frac{1}{size_i} \quad (1)$$

Here w_r, w_d and w_s represent weights of the three parameters. Obviously sum of the weights must be equal to 1, e.g. $w_r + w_d + w_s = 1$. Hence by changing weights peer can give more importance on one parameter than others.

B. PReCinCt: Handling Mobility

As PReCinCt divides the network topology in geographical regions, so mobility of peers has been divided into two categories that are intra-region mobility and inter-region mobility. Actually there is no over head in case of intra-region mobility as peer does not leave the region. On the other hand, in inter-region mobility peer crosses the region and being a member of new region. No reference is better than wrong reference. Believing this principle, peer has to distribute the keys for which he was responsible to the others peers of the region being departed. Departing peer hands over the keys to those peers whose have enough cache space to hold the keys, have low mobility rates, and are located near the center of the region. Peer that is close to the center of the region and has low mobility rate is less likely to move in near future.

C. PReCinCt: Other Issues and analysis

To manage the network topology PReCinCt defines four functions. These are *Add, Delete, Merge* and *Separate*. To expand and shrink the topology it uses *Add* and *Delete* function respectively. *Merge* combines two regions while S*eparate* splits a large region into two independent regions. For the sake of the data consistency among the replicas PReCinCt uses *Push with Adaptive Pull* scheme. As the name suggests it combines the features of both Push and Pull scheme. In order to check cached data consistency it uses pulling method. While in push method, a peer updating an old data item sends the updated data to the home region.

PReCinCt uses a composite caching policy. But it does not tell how to determine the mobility of the nodes and distance between the requesting and the responding nodes. Even though p*ush with adaptive pull* ensures consistency but again initiates the problem of excessive bandwidth consumption.

VI. DYANAMIC INDEX IN MOBILE P2P

In traditional peer-to-peer systems resource discovery and locating are most challenging. It is even more difficult in mobile p2p environment due to its limited resources. Indices are the ways to resort the problem to some extent. In the next sub-sections we elaborate dynamic indices for mobile p2p networks [17].

A. Dynamic Indices: Design and Implementation

Indices provide reference information of the resource location. Hence it improves the performance of the system. Dynamic index is a method that can adopt the sudden topological change in the environment due to nodes mobility. Here every node keeps an index. This index maintains reference information of the nearby resources of the closer nodes. In order to define scope of the reference dynamic indices use the concept of radius. Nodes are allowed to change their radius considering the mobility of the nearby nodes. Figure 12 is an example of changing the index scope. If the system static it will use outer radius otherwise it will use inner radius.

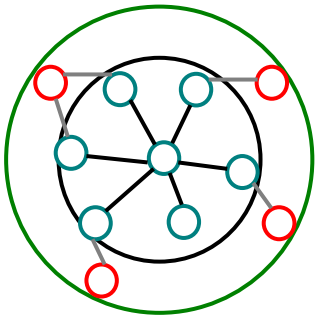

Figure 12: changing the index scope

When a node, say node A, would like to join the system, it simply broadcasts GET request message in the network (figure 13). The format of the GET request is given in figure 14. TTL (Time to Live) field refers to how many hops the message will travel in the network. Eventually, this message has been propagated in the network and requester will get the response indicating that what resources are stored on them. For example, in the table 1, requester A will form an index table based on the topology given in figure 13.

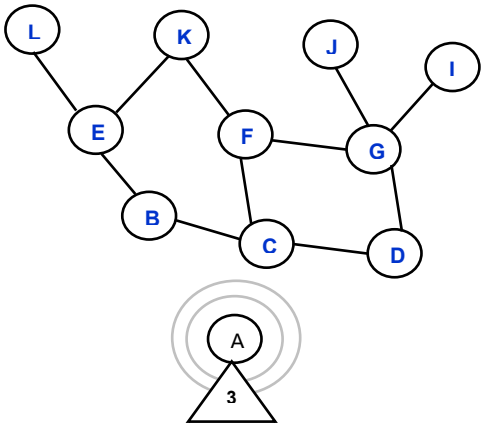

Figure 13: Creating Dynamic Index

CMD	TTL	SRC
GET	7	A

Figure 14: GET request data structure

Table 1: Index for node A

Owner	File name	Next Hop
B	1	B
B	2	B
C	2	C
D	3	D
E	2	B
E	4	B
F	3	C
…	…	…
K	10	C

This table indicates file no 2 is stored on node B, C and E. It also maintains next hop information to download the corresponding file. So if node A would like to download the file 3, it can forward the request to either node C or node D. What will happen if node A fails to locate the file in it index table? In this situation all R hop neighbors are requested to launch the same lookup process on their index tables.

Each node refreshes its index table at T time unit interval. If the new and the old index table defer to a large extent it will reduce its radius. It can increase the radius if some consecutive refreshes result in same index table, providing the clue of static environment.

B. Dynamic Indices: Analysis

One of the key features of dynamic indices is reduction in the number of broadcast requests. It accelerates content searching. Moreover, in the design process it considers the mobility of the nodes. But periodic refresh messages increase control and processing overhead. The searching method they deployed has no intelligence as it totally ignores the inherent benefits of hashing.

VII. CONCLUSION

With the advancement of mobile computing and the popularity of mobile devices, mobile peer-to-peer communication is likely to boom soon. But the limitations that mobile systems are now facing, such as limited computing power (handheld devices), limited bandwidth, limited space etc, make it challenging to design scalable, robust and bandwidth efficient communication platform for mobile devices.

Mobile P2P is the way to shared content resources among mobile devices. Resource discovery and location tracking are really difficult. Distributed hashing schemes are suitable for wired p2p system. These are a class of decentralized, distributed systems and algorithms developed to provide a scalable, self-configuring infrastructure with a clean programming interface for p2p systems [18]. It maps both nodes and data to a circle or hypercube and provides routing either in $O(1)$ or $O(\log n)$ hops. This distributed hashing can be adopted on mobile p2p systems.

Broadcasting is expensing in mobile or in ad hoc networks. Mobile devices always lose energy even at idle or sleeping state. Instead of applying pure broadcasting when it is required, dominating set with neighbor elimination method might be an alternative solution. That will reduce redundant message re-transmissions.

JXTA technology is a set of open protocols. With help of JXTA, mobile devices such as PDA, handheld devices or even servers are allowed to communicate in p2p manner [18]. So it is now becoming easier to design and implement mobile p2p protocols as compared to earlier dates.

VIII. REFERENCES

[1] http://en.wikipedia.org/wiki/Peer-to-peer
[2] http://en.wikipedia.org/wiki/Napster
[3] http://www.slyck.com
[4] H.-C. Hsiao and C.-T. King. “Capability-Aware Peer-to-Peer Storage Networks,” *In Int’l Parallel and Distributed Processing Symposium*, April 2003.
[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. “A Scalable Content-Addressable Network,” *In ACM SIGCOMM*, pages 161-172, August 2001.

[6] A. Rowstron and P. Druschel. “Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems,” *In Proc. of the IFIP/ACM Int’l Conf. on Distributed Systems Platforms (Middleware 2001),* Nov. 2001.

[7] Hung-Chang Hsiao, Chung-Ta King. "Bristle: A Mobile Structured Peer-to-Peer Architecture," *In International Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003.

[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications,” *In ACM SIGCOMM*, pages 149-160, August 2001.

[9] J. Li, J. Jannotti, D. De Couto, D. R. Karger, and R. Morris. “A Scalable Location Service for Geographic Ad Hoc Routing,” *In Proc. of the ACM/IEEE Int’l Conf. on Mobile Computing and Networking*, pages 120-130, August 2000.

[10] Sylvia Ratnasamy et al, “A Scalable Content-Addressable Network”, *In Proceedings of the ACM SIGCOMM*, 2001.

[11] Gang Peng, Shanping Li, Hairong Jin, Tianchi Ma. "M-CAN: a Lookup Protocol for Mobile Peer-to-Peer Environment," *In the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'04)*, 2004.

[12] H. Shen, M. Kumar, S.K. Das, and Z. Wang, “Energy-Efficient Caching and Prefetching with Data Consistency in Mobile Distributed Systems,” I*n Proc. of IEEE International Parallel and Distributed Processing symposium (IPDPS)*, Santa Fe, NM, April 2004.

[13] L. Fan, “Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol,” *In Proceedings of ACM SIGCOMM Conf., ACM Press,* 1998.

[14] P. Sarkar, and J. H. Hartman, “Hint-based cooperative caching,” *ACM Transactions on Computer Systems*, volume 18, number 4, pages 387–419, 2000.

[15] Huaping Shen, Mary Suchitha Joseph, Mohan Kumar, Sajal K. Das. "PReCinCt: A Scheme for Cooperative Caching in Mobile Peer-to-Peer Systems," *In the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005.

[16] B. Karp, and H. T. Kung, GPSR: Greedy Perimeter Stateless Routing for Wireless Networks In Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000), August 2000.

[17] Gang Peng, Shanping Li, Hairong Jin, Tianchi Ma. "Dynamic Indices for Mobile Peer-to-Peer Networks," *In The Fourth International Conference on Computer and Information Technology (CIT'04)*, 2004.

[18] http://www.answers.com