INT J COMPUT COMMUN, ISSN 1841-9836
Vol.7 (2012), No. 4 (November), pp. 696-708

P2P Resource Sharing in Wired/Wireless Mixed Networks

J. Liao

Jianwei Liao
College of Computer and Information Science
Southwest University of China
400715, Beibei, Chongqing, China
E-mail: liaojianwei@il.is.s.u-tokyo.ac.jp

Abstract: This paper presents a new routing protocol called Manager-based Routing Protocol (MBRP) for sharing resources in wired/wireless mixed networks. MBRP specifies a manager node for a designated sub-network (called as a group), in which all nodes have the similar connection properties; then all manager nodes are employed to construct the backbone overlay network with ring topology. The manager nodes act as the proxies between the internal nodes in the group and the external world, that is not only for centralized management of all nodes to a certain extent, but also for avoiding the messages flooding in the whole network. The experimental results show that compared with Gnutella2, which uses super-peers to perform similar management work, the proposed MBRP has less lookup overhead including lookup latency and lookup hop count in the most of cases. Besides, the experiments also indicate that MBRP has well configurability and good scaling properties. In a word, MBRP has less transmission cost of the shared file data, and the latency for locating the sharing resources can be reduced to a great extent in the wired/wireless mixed networks.
Keywords: wired/wireless mixed network, resource sharing, manager-based routing protocol, backbone overlay network, peer-to-peer.

1 Introduction

Peer-to-Peer technology (P2P) is a widely used network technology, the typical P2P network relies on the computing power and bandwidth of all participant nodes, rather than a few gathered and dedicated servers for central coordination [1, 2]. According to the research and analysis on Internet traffic management conducted by ipoque Germany, P2P applications dominate Internet traffic from 50% to 90%, and the statistics from Chinese sources reveal that P2P traffic currently accounts for 70% of China's total network traffic [3]. This indicates that resource sharing via various P2P techniques contributes to the major part of resource sharing on the Internet [4].

In general, P2P systems implement an abstract overlay network, built at application layer on top of the native or physical network topology [2]. From the architecture view [5], P2P systems are generally divided into structured systems, unstructured systems [6] and hybrid systems [7]. A structured P2P system employs a globally consistent protocol to ensure that any node can efficiently find some of the peers that have the desired resources, even though the file is extremely rare. However, since DHT-like (Distributed Hash Table, DHT) data structure is employed for maintaining the whole structured P2P system, the scalability is a critical problem. Chord [8] is a typical structured P2P system. The unstructured P2P system is formed when the overlay links are established arbitrarily. In an unstructured P2P system, if a peer wants to lookup a piece of desired resources in the network, the query has to be flooded through the network to find the peers who have the desired sharing resources as many as possible. However, the unstructured P2P system uses flooding queries to discover the target objects, which may introduce lots of network traffic. Bittorrent [10] is a well-known unstructured P2P system. In addition, many structured P2P systems use stronger peers (super-peers or super-nodes [11]) as servers, and the client-peers are connected in a star-like fashion to a single super-peer. This architecture can simplify the

network architecture, but super-peers hold all routing information, even though a local search is also conducted by a relevant super-peer, thus the super-peers are apt to be overloaded. As examples for hybrid networks can be named modern implementations of Gnutella2 [12] and the eDonkey network [13].

Nowadays, various modern devices can access Internet by different resorts, and these devices also want to share resources with others, it is quite clear they can employ the P2P techniques. But different kinds of devices have the different properties and purposes of use, thus the sharing targets are inequality. For instance, widely used handheld devices may share a several megabytes mp3 audio file, and rarely share a size up to several gigabytes video file. But for normal desktops and laptops, the latter sharing is quite common; therefore, treating different kinds of peers as same is not an ideal strategy in the heterogeneous networks.

In this paper, we propose, implement and evaluate a new routing protocol called MBRP (Manager-based Routing Protocol) for constructing P2P resource sharing networks, in which there are many disparate devices. As a matter of fact, MBRP has been inspired by the hybrid P2P architecture, but in MBRP, the target sharing resources might be replicated and stored on the manager nodes, and the message forwarding may not conducted by manager nodes. In addition, although the P2P protocols mentioned above are scalable and efficient, they were designed originally for wired networks and are generally not suitable for wireless networks, in which nodes join and depart much more frequently. The main idea of MBRP is to organize the diverse devices into different groups according to their properties such as location, wired or wireless etc., and then appoint a manager node for each group to communicate with other groups. Since the devices in the same group have the similar properties, the expected resources are stored and shared in the same group with quite high probability, only the desired resource is not in the group, the inter-group communication is launched.

This paper is organized as follows: we present the design and implementation of MBRP in Section 2; the evaluation experiments and results are shown in Section 3; finally, we make concluding remarks in Section 4.

2 The Design and Implementation of MBRP

As various Internet-connected devices have the different properties including bandwidth, connection resorts and storage capacity, the proposed MBRP first divides all participant nodes into several groups according to their properties such as wired or wireless connection, then elects a manager node for each group for communicating with the external nodes belonging to other groups. Different from traditional hybrid P2P systems, in MBRP, the nodes may communicate with other internal nodes belonging to the same group directly without any intervention from the manager node. Besides, the manager nodes might cache the replicas of hot sharing resources to reduce the lookup and transmission overhead for the sharing objects.

2.1 The Architecture of MBRP Network

Figure 1 shows the topology of a heterogeneous network built by resorting to MBRP. All nodes are divided into several groups according to their connection property, i.e. wired or wireless access. In each group, there is only one proxy node called manager node, which is responsible for the management of other internal nodes in the same group. For instance, the internal nodes who want to communicate with other nodes belonging to the different groups, are supposed to resort to their manager node. In addition, all manager nodes are connected into a ring, a similar topology to Chord [8], but all internal nodes in the same group can connect to each other via different topologies.

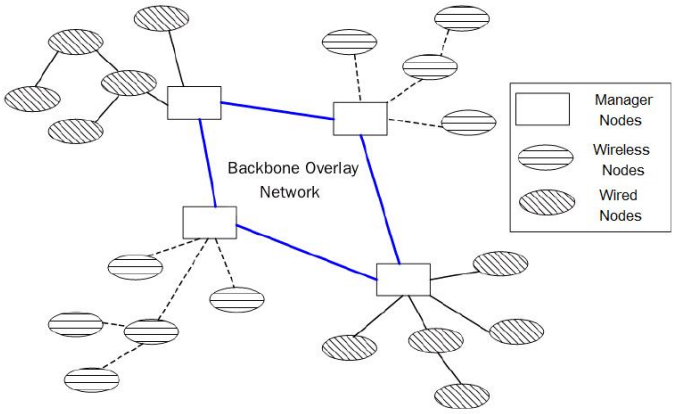

Figure 1: The Topology of Wireless/Wired Mixed Network via MBRP

2.2 Application Routing Protocol

We assume that MBRP is built on IP-based wide area network, we use a m-bit hash function to calculate the unique m-bit group ID. All internal nodes in the same group have their unique group ID, which is also used by the manager node as its own private ID to communicate with other manager nodes. For the purpose of routing in the whole network, like routing in Chord, each manager node holds its predecessor and finger table [14].

$$T_k = BNID + 2^{k-1} \ mod \ 2^m \quad (\forall k, where \ 1 \leq \ k \leq \ m) \tag{1}$$

Equation 1 is employed to calculate the manager node's successors, where *BNID* is the manager node ID, and m is the number of bits of manager node ID, T_k is the ID of the first successor; then, it calls the **find_successor**(T_k) function, which is shown in Figure 2, to calculate the next successor $T_k\prime$; finally, each manager node has m successors in its finger table when the **find_successor** function has been called *m-1* times. Figure 3A shows the finger table of node *N5512*, in which some example successors of node *N5512* are demonstrated.

```
n.find_successor (id)
    if (id ∈ (n, suceessor))
        return successor;
    else
        return successor.find_successor (id);
```

Figure 2: Find Manager Node's Successor

For the nodes who want to publish some sharing files after joining into the group, they are supposed to add the group ID as a part of the identification of the sharing files. Figure 3B shows that an internal node belonging to group *N2124* has published a file named as *K4814*, but the published name sent to the other manager nodes is *N2124#K4814*. For other manager nodes, that means the sharing file *K4814* is located in the *N2124* group.

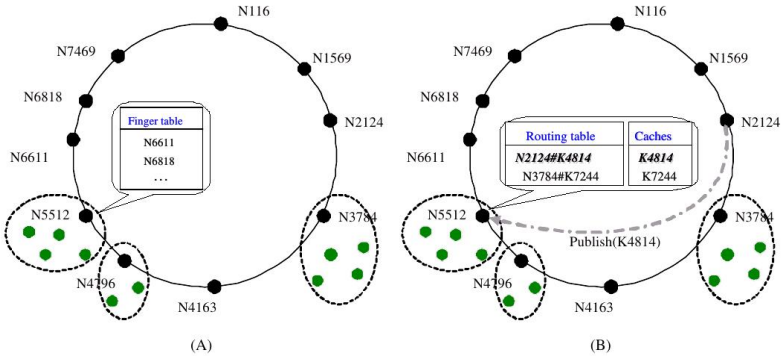

Figure 3: (A) Finger Table of *N5512*; (B) Routing Table and Cache Table of *N5512*

Creating Routing Table

As shown in Figure 3B, for publishing file *K4814*, the manager node *N2124* computes the first successor according to Equation 1, therefore, *N2124* publishes *K4814* to its successor manager node, i.e. *N5512*; then *N5512* adds the new entry *N2124#K4814* to its own routing table; finally, it makes a replica of *K4814* and an associated cache entry in the cache table if the cache strategy is enabled.

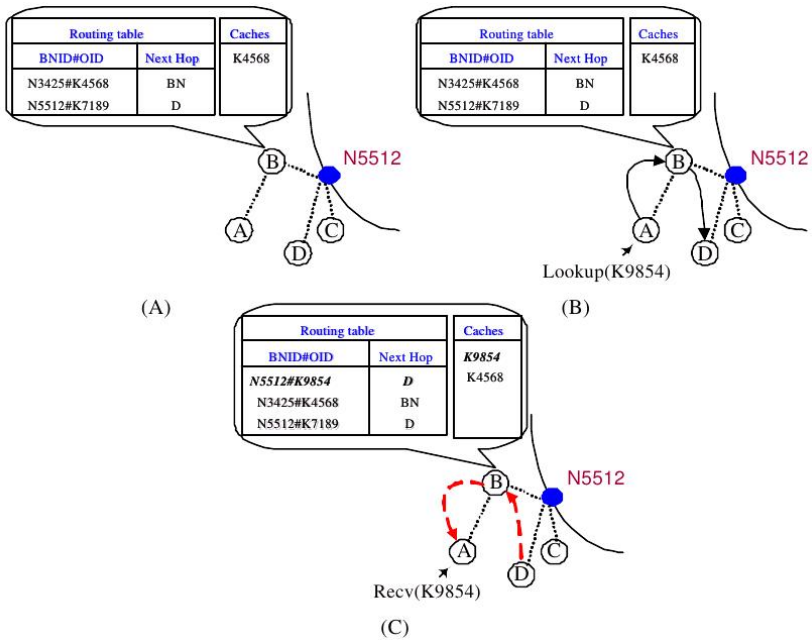

Figure 4: (A) Routing Table of Internal Node B; (B) Lookup Originated from Internal Node A to B and D; (C) Lookup Succeed Message Passed Back From D to B and A

While an internal node lookups a piece of sharing resource, which is in the different groups, then the lookup process can be taken over by its manager node. Otherwise, while the sharing file is in the same group, the desired file data can be transferred within the group directly. Thus, each internal node also holds a routing table to show the routines within the group. However, quite different from the manager node, the IDs of the internal nodes are their IP addresses; thus,

the routing table is different as well. The routing table of the internal node B (B represents node's IP address) is shown in the Figure 4. *BNID#OID* stands for the manager node ID and the sharing resource ID, and the Next Hop means the next node for getting the sharing resource. For instance, in order to obtain the sharing resource *K7189* located in group *N5512*, internal node B should send the lookup request to the next hop, i.e. the internal node D. The symbol *BN* in the routing table represents the ID of the manager node that in the same group.

Lookup Algorithm

The lookup algorithm will be described from 4 parts: sending a request, receiving a request, manager node routing on the backbone overlay network and receiving a lookup hit message. Figure 5A shows the algorithm of sending a lookup request. If a node wants to locate a piece of sharing resource labeled as *obj_ id*, then it calls **Send_LookupReq(obj_id)** to send the lookup request. First, the **find_cache(obj_id)** function is called to make sure whether the sharing object is in its own cache or not; if not,the **find_routetable(obj_id)** function will be called to obtain the related routing information; while there is no related routing information, it calls **get_approximated(obj_id)** to obtain the feasible routing information to find the next hop n'. Finally, **Send_Req(obj_id, n′)** is called to send the lookup request to the next hop n'. Figure 4A shows an example of sending a lookup request for the sharing object *K9854*, since there is no corresponding entry in the route table, the **get_approximated(K9854)** function is called to obtain the feasible routing entry and the lookup request is forwarded to the selected node D.

```
// send a lookup request to find the obj_id
Send_LookupReq (obj_id)
    r = nil;
    n' = nil;
    if (find_cache (obj_id) is not nil)
        r = get_routetable (obj_id);
    else
        r = get_approximated (obj_id);
    n' = r.nexthop;
    Send_Req (obj_id, n');
```

```
// receive a lookup request from the predecessor
Recv_LookupReq (obj_id)
    r = nil;
    n"= nil;
    if (find_cache (obj_id) is not nil)
        Send_Lookup_Hit (src, obj_id, data);
        return;
    if (find_routetable (obj_id) is not nil)
        r = get_routetable (obj_id);
    else
        r = get_approximated (obj_id);
    n" = r.nexthop;
    Send_Forward (obj_id, n");
```

Figure 5: (A) Sending a Lookup Request; (B) Receiving a Lookup Request

The algorithm of receiving a lookup request from the predecessor is almost same to that of sending a lookup request. It receives a lookup request, and then processes like sending a lookup request, the pseudo-code of the algorithm is shown in Figure 5B. While the target sharing object is found, then the **Send_LookupHit(obj_id, src, data)** function will be called to transfer the resource to the request node. Figure 4B shows an example of the procedure while the internal node B handles the received lookup request from node A. Since there is no corresponding routing entry in node B's routing table, it calls **get_approximated(obj_id)** to find the feasible next hop, i.e. node D, and forwards the request to it. The process of handling a received lookup request does not stop until the resource is found or timeout (i.e. maximum hop count exceeded).

While the lookup request is not fulfilled within the group, it will be forwarded to the manager node, the **Core_Ring_Route(obj_id)** function shown in Figure 6A, will be called by the group's manager node to handle the request from other manager nodes. After receiving the

```
// routing in backbone overlay network
Core_Ring_Route (obj_id)
    r = nil;
    n'= nil;
    b' = nil;
    if (find_cache (obj_id) is not nil)
        Send_Lookup_Hit (src, obj_id, data);
        return;
    if (find_local (obj_id) is not nil)
        n' = get_local (obj_id);
        Send_Forward (obj_id, n');
    else
        if (find_routetable (obj_id) is not nil)
            r = get_routetable (obj_id);
            b' = r.bnid;   // the manager ID
        else
            b' = find_successor (obj_id);
    Send_Core_Ring_Forward (obj_id, b');
```

```
// receive a object hit message
Recv_LookupHit (src, obj_id, data)
    update_routetable (src, obj_id);
    if (src is not equal to me)
        Send_Lookup_Hit _BackRoute
            (src, obj_id, data);
    if (find_caches (obj_id) is nil)
        append_caches (data);
    return;
```

Figure 6: (A) Routing in Backbone Overlay Network; (B) Algorithm of Receiving a Lookup Hit

lookup request, the manager node checks the target object is in its own group or not. If the object is in its group, then the request is forwarded to the corresponding internal node; otherwise, it forwards the request to other corresponding manager node whose group has the target object or the successor manager node in the finger table.

The algorithm of receiving a lookup hit message is shown in Figure 6B, while the manager node receives the lookup hit message, it first updates its routing table to label the new routine to the target object; then if the cache mechanism is enabled, it also makes a replica of the target object on its local disk.

From the above description, we can see that while the target sharing object is found, then the expected file will be transferred to the request node by the reverse routing path. At the same time, if cache strategy is enabled, one replica of this target object is made and stored in each manager node on the routing path for quick responses to the future lookup requests. We should mention that the number of cached replicas in the manager node is configured and limited, LRU is used to evict an existing replica and append the new replica into the local cache.

2.3 Dynamic Registration

Since the handheld devices have the roaming property, that means they might change their groups and manager nodes frequently, we have adopted a mechanism like IP mobility support [15] called dynamic handover, to allow the handheld device to register to a new manager node after the roaming. Figure 7 illustrates the dynamic registration in following steps:

1. The handheld device issues a registration request with its current IP address, the former IP address and the former manager node ID (hashed value from the manager node's IP address) to the new manager node for registration after it roams to another group.

2. The new manager node creates a new internal node record, and replies the handheld device with message about the registration has been handled. The handheld device updates its manager node and tries to re-build its routing table again.

3. The new manager node notifies the device's former manager node that the device has joined into its group and the old record should be removed; then the previous manager

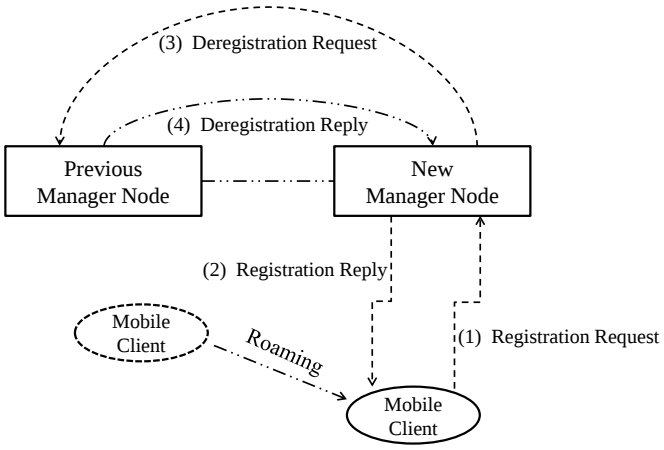

Figure 7: Dynamic Handover Algorithm

node deletes the corresponding record, and broadcasts that de-registration message to its all internal nodes to require them to update their routing tables.

4. The previous manager node replies to the new manager node with the message of the de-registration has been handled.

2.4 Manager Node Election

The manager node plays a critical role in the MBRP network, while the manager node fails or departs from the group, a new manager node is supposed to be elected. As a matter of fact, the main principle for electing a new manager node is the well relaying property, which means lots of internal nodes choose it as the next hop in their routing table entries; if more than one internal nodes have the same relaying property, then one of them will be selected randomly.

On the premise that network is still working when the manager node has exited or failed, the internal node, who first detects the failure or exit of the manager node, broadcasts the election request to other internal nodes. We assume the messages related to election are never lost, then all internal nodes belonging to the same group should take part in the process of election.

1. After receiving the election request, all internal nodes check the possible manager node candidate(s) according to the relaying property, and then reply to the issuer of the election request with the candidate(s). It is possible that the internal nodes may send several candidates who have the same relaying property;

2. The issuer of election request collects all replies from the internal nodes, then determines which node is the unique manager node;

3. The result of election will be broadcast to the corresponding internal nodes;

4. The new created manager node should be insert into the backbone overlay network simply like inserting a node into a ring topology network. In general, the new manager node replaces the failure one in the ring of the backbone overlay network;

5. All nodes in the group (including the manager node), which have a new manager node, remove the records with previous manager node and update the manager node information;

6. The new manager node broadcasts that the former manager node is not working, to other manager nodes on the backbone network and requires them to update the routing table. At the same time, the new manager node re-builds its routing table and finger table.

We should mention that because all nodes in the group should re-build their routing tables, and routines on the backbone overlay network are supposed to be updated as well, the overhead brought by electing a manager node is not trivial.

3 Experiments and Evaluation

The NS2 [16] was employed as our experimental platform while analyzing the performance and overhead on both the MBRP system and its comparison counterpart. Much more exactly, the module Gnutellasim in NS2 is used for our experiments. The hybrid P2P system Gnutella2, which uses super-peers (called hubs) to manage the internal nodes in the same group, has been selected as our comparison counterpart while evaluating the overhead, such as network traffic, in the manager nodes in the MBRP network system.

Moreover, the manager nodes play significant roles in our proposed mechanism MBRP, so that they are supposed to have enough bandwidth and processing power. Because wireless network is connected to the wired network via a gateway (also called access point), in our experiments, we selected such nodes as manager nodes for wireless groups. Regarding to wired groups, we simply appointed the fixed servers as manager nodes. Thus, all nodes can join the whole system by registering to their own manager nodes.

3.1 Overhead on Manager Node

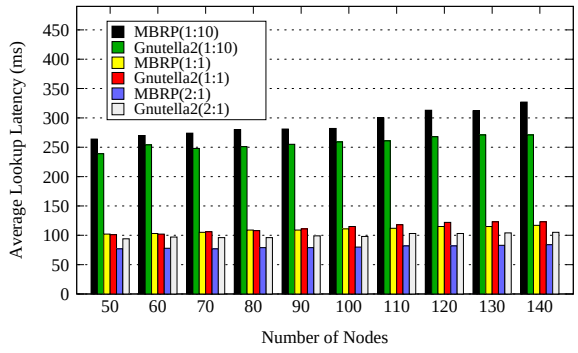

Figure 8: The Average Lookup Latency

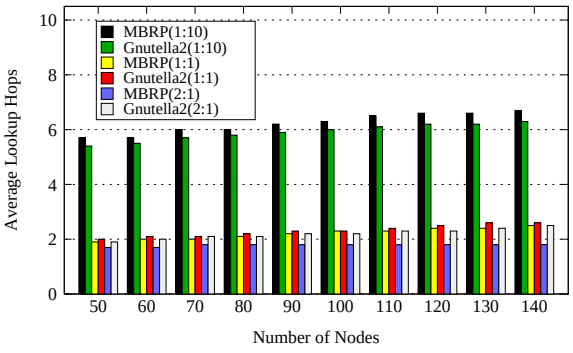

Figure 9: The Average Lookup length

We have conducted the comparison experiments between Gnutella2 and the proposed MBRP, to show the overhead of introducing the manager nodes in MBRP. For both target systems, the overhead of locating the local objects (belonging to the same group) and locating the external objects (belonging to the different groups) are different, we adopted different Internal/External Access Ratios, i.e. 1:10, 1:1, 2:1, to show the different overhead; each group has 50 internal nodes and each experiments stands 200 seconds.

We measured both the average lookup latency and the average lookup hops for Gnutella2 and MBRP with different access ratios. Figure 8 reports the results of average lookup latency (lower is better), except for the case of Internal/External Access Ratio is 1:10, while no less than half of lookups for sharing resources are hit within the group (i.e. access ratios are 1:1 and 2:1), the lookup latency introduced by MBRP is less than Gnutella2, especially while the access

ratio is 2:1, MBRP reduces around 20% lookup latency. That is because all communications within the group should be handled by the super-peers in Gnutella2, but the internal nodes may communicate with each other in MBRP.

Figure 9 presents the results of average lookup length(i.e. hot count, lower is better), the similar tendency to the average lookup latency between Gnutella2 and MBRP. In addition, from Figures 8 and 9, we can conclude that MBRP needs just a little more time while increasing the total number of nodes, this shows that MBRP has well scalability.

3.2 Overhead on Backbone Overlay Network

NS2-Gnutellasim was also employed to show the traffic on the backbone overlay network with the different network properties. In order to accelerate the access speed to the sharing resources, MBRP applies cache mechanism to store the hot resources in the manager nodes when these resources are transferred via/to them. In this section, we will inspect that different cache strategies bring about what kind of benefits to the lookup latency and negative effect on the traffic overhead in backbone overlay network respectively. The following cache strategies have been taken into consideration:

1. **No Cache**, which represents that the cache mechanism is disabled.

2. **Unlimited Cache**, which means the manager nodes can cache unbounded replicas.

3. **Weighted Cache**, which indicates that the manager nodes can cache limited resources, while the ceiling is reached, some existing cached resources should be swapped out to make space for the new replicas with LRU cache algorithm.

Because in both of Gnutella2 and the MBRP mechanisms, all inter-group messages are handled by the manager nodes or super-peers, the traffic overhead on backbone overlay network of Gnutella2 is same to that of MBRP without cache. We do not report the experimental results regarding to Gnutella2 in the following experiments.

In these experiments, GT-ITM [17] is used to construct the network topology, which has 500 manager nodes, the size of the sharing resource is 1024 byte, Wired/Wireless access Ratio is 1:1, and the duration of each experiment is 500 seconds. Since P2P is an application level protocol, we only care about application level packets rather than other level packets.

Average Traffic Overhead

We defined the average traffic overhead as the traffic on backbone overlay network divided by the size of sharing resource. For instance, in order to transfer a sharing file (default size as 1024 byte), which introduces 4096 byte total traffic on backbone overlay network, then the average traffic overhead is 4.

Figure 10 shows the average traffic overhead by using different cache strategies, the X axis stands for the number of successful lookups, in other words, during 500 seconds for conducting an experiment, how many successful lookups have been completed; the Y axis represents the average traffic overhead. Without doubt, No Cache strategy works worst, meanwhile Unlimited Cache mechanism works best. Figure 11 shows the total traffic in the all manager nodes, which has the similar trend to that of average traffic, In addition, Figures 10 and 11 also show that with the increasing the maximum number of caches, the total traffic goes down slowly.

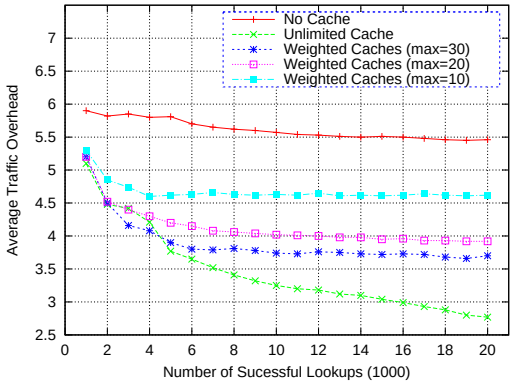

Figure 10: Average Traffic Overhead

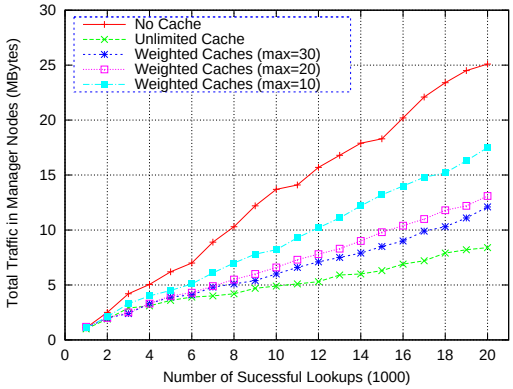

Figure 11: Total Traffic in Manager Nodes

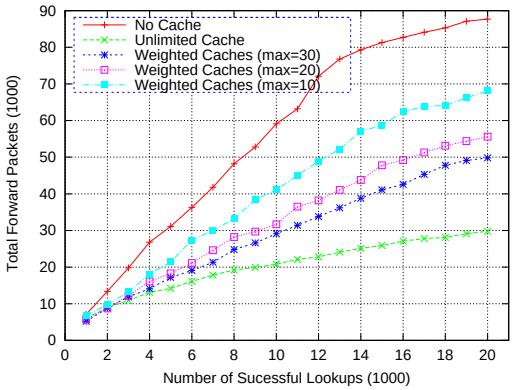

Figure 12: Total Forward Packets on Backbone Overlay Network

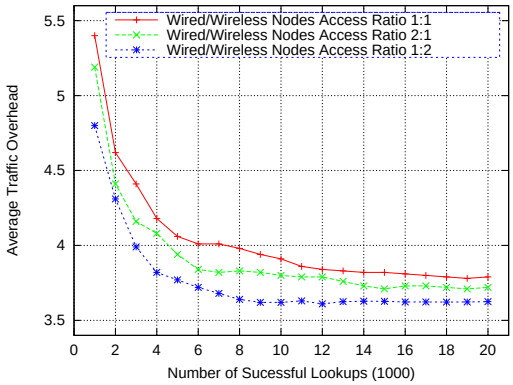

Figure 13: Average Traffic Overhead with Different Wired/Wireless Access Ratio

Total Traffic Overhead

Figure 12 shows the total forward packets on the backbone overlay network. While the number of cached resources is increasing, then less packets is forwarded on the backbone overlay network. That is because the more cached replicas, the more lookup hits can be obtained within the group.

Overhead with Different Wired/Wireless Access Ratios

In the previous experiments, we fixed the wired/wireless access ratio as 1:1. In this section, we will discuss the ratios are 1:1, 2:1 and 1:2 respectively. First, we configured the cache strategy as Weighted Caches (max=30). Then we repeated to measure the average traffic overhead, total traffic in manager nodes and total forward packets on the backbone overlay network.

Figures 13, 14 and 15 show the relevant results of overhead on backbone overlay network with different wired/wireless access ratios separately. From these figures, we can see that while the ratio is 1:2, the MBRP with cache mechanism can achieve considerable performance improvement because of a major part of lookup hits occurred in the groups.

3.3 Discussion

From the above experiments, we can see that MBRP has low latency, and the cache mechanism is also suitable for a large amount of accessing to the sharing resources in wireless/wired mixed networks. In addition, MBRP keeps a good scaling property, it employs manager nodes

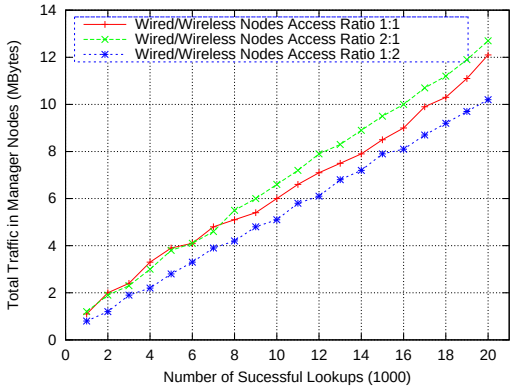

Figure 14: Total Traffic in Manager Nodes with Different Wired/Wireless Access Ratio

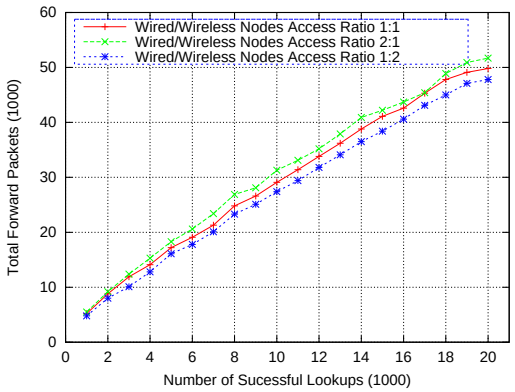

Figure 15: Total Forward Packets with Different Wired/Wireless Access Ratio

to construct the backbone overlay network, and then other nodes can register to the manager node to join into the whole network. The manager nodes are responsible for the communication between the different groups, therefore, both traffic overhead and lookup hops do not increase drastically even though lots of the new nodes join into the whole network suddenly.

4 Concluding Remarks and Future Work

A new routing protocol named as Manager-based Routing Protocol (MBRP) has been proposed, implemented and evaluated in this paper. All nodes in the network have been partitioned into several groups according to their properties, wired or wireless access for instance; then a manager node is elected for each group and in charge of communication between the internal nodes in the group and external nodes belonging to other groups. From our experimental results, compared with Gnutella2, which has super-peers for different groups, except the Internal/External Locating Ratio is 1:10, in which MBRP performs a little worse than Gnutella2; in other cases, MBRP outperforms than Gnutella2. In addition, since the nodes in the same group, which have the similar properties, are mostly like to share the same kind of resources, the most of resource sharing cases may occur within the group. Namely, MBRP can work well in the heterogeneous networks, in which internal accesses might more than external ones.

Furthermore, for responding quickly to the lookup requests for the hot resources, MBRP adopts caching the hot resources in the group while transferring the target objects from external groups. Therefore, the future lookups for these cached resources can be fulfilled in the local group. The cache strategy reduces not only the lookup latency and lookup node hops, but also

the network traffics on the backbone overlay network. Consequently, the system performance can be upgraded greatly.

However, the current design of MBRP still has it limitations, although we assume that the sharing resources are not modified frequently, modification of the resources really happen, thus we need to consider how to maintain the consistency between the cached copies and the original ones in the near future. In addition, to determine which manager nodes for storing the copies, and make the cache mechanism much more effective is another aspect of our future work.

Bibliography

[1] G. Fox, Peer-to-peer networks, Computing in Science and Engineering, ISSN 1521-9615, 3(3):75-77, 2001.

[2] R. Schollmeier , A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications, Proceedings of the First International Conference on Peer-to-Peer Computing, pp.101-102, 2001.

[3] Eric Bangeman, P2P responsible for as much as 90 percent of all Net traffic (http://arstechnica.com/old/content/2007/09/p2p-responsible-for-as-much-as-90-percent-of-all-net-tra_c.ars, 2007

[4] M. Parameswaran, A. Susarla, A.B. Whinston, P2P networking: an information sharing alternative, Computer, ISSN 0018-9162, 34(7):31-38, 2001.

[5] Analoui M., Sharifi M., Rezvani M.H., Probabilistic Proximity-aware Resource Location in Peer-to-Peer Networks Using Resource Replication, International Journal of Computers Communications & Control, ISSN 1841-9836, 5(4):447-457, 2010.

[6] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker , Search and replication in unstructured peer-to-peer networks, Proceedings of the 16th international conference on Supercomputing,ICS'02, pp.84-95, 2002.

[7] Beneventano, Domenico and Bergamaschi, Sonia and Guerra, Francesco and Vincini, Maurizio , Querying a super-peer in a schema-based super-peer network, Proceedings of the 2005/2006 international conference on Databases, information systems, and peer-to-peer computing, pp.12-25, 2007.

[8] M. Kelaskar, V. Matossian, P. Mehra, D. Paul, M. Parashar, A Study of Discovery Mechanisms for Peer-to-Peer Application, Proceedings of the 2nd IEEE/ACM Intenational Symposium on Cluster Computing and the Grid,IEEE Computer Society Washington, DC, USA, pp.444, 2002.

[9] Sameh Elfiansary , Luc Onana Alima , Per Brand , Seif Haridi , Efficient Broadcast in Structured P2P Networks, Proceedings of 2nd International Workshop On Peer-To-Peer Systems, 2003.

[10] BitTorrent, http://www.bittorrent.com

[11] J. Lin, M. Yang, Robust Super-Peer-Based P2P File-Sharing Systems, the Computer Journal, ISSN 0010-4620, 53(7):951-968, 2010.

[12] M. Ripeanu , Peer-to-peer architecture case study: Gnutella network, Proceedings of the First International Conference on Peer-to-Peer Computing, pp.99-100, 2002.

[13] S.B. Handurukande et al , Peer sharing behaviour in the eDonkey network, and implications for the design of server-less file sharing systems, Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006, pp. 359-371, 2006.

[14] Stoica, Ion and Morris, Robert and Karger, David and Kaashoek, M. Frans and Balakrishnan, Hari, Chord: A scalable peer-to-peer lookup service for internet applications, SIGCOMM Comput. Commun. Rev. ISSN 0146-4833, 31(4):149-160, 2001.

[15] IP Mobility Support for IPv4, http://tools.ietf.org/html/rfc3344

[16] Rupali Bhardwaj and V.S. Dixit and Anil Kr. Upadhyay, An Overview on Tools for Peer to Peer Network Simulation, International Journal of Computer Applications, ISSN 0975-8887, 1(1):70-76, 2010.

[17] E. W. Zegura, GT-ITM: Georgia Tech internetwork topology models (software), http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz, 1996.