A NAT-ed Peer Organization Model in Kademlia Protocol

Bingqing Shen, Jingzhi Guo and C. L. Philip Chen
Faculty of Science and Technology
University of Macau,
Macau, China
{yb17412, jzguo, philipchen}@umac.mo

Abstract*—Peer-to-peer network and application have boomed for the recent decade. However, Network Address Translation (NAT) devices obstruct its wide application by restricting the access to peers from private network. In this paper, we devise a peer strongly connected model to adapt Kademlia protocol to NAT-involved environment to minimize the impact on routine algorithm and meanwhile to make the target system reliable and scalable. Evaluation shows that our organization model highly outperforms existing model in delivery rate and scalability.

***Keywords*—Network Address Translate (NAT); NAT Traversal; Peer-to-peer (P2P); Peer Organization; Kademlia**

I. INTRODUCTION

As peer-to-peer application has been studied and applied in large scale of fields, current Internet structure is not friendly to end-to-end connection. With the increase of home computer and personal mobile devices, IP address has become a shortage and valuable resource which is not able to support the one-machine-one-address structure because of its intrinsic limitation [1][16]. This makes network address translation (NAT) [1] the de facto standard interface between public and private network such as home LAN. Literatures [2, 19, 30] show more than 80% computers are behind NAT routers or firewalls. This results in that peer-to-peer application design needs more consideration in real life. Most of well-known peer-to-peer routing and searching algorithms such as flooding [3][4], random walk [5][6] and DHT [3] assume that peers are equally accessible to each other. They cannot be directly applied to the wild without the consideration of impact from NAT traversal. As illustrated in Fig. 1, the message not only has to choose a longer path to arrive at the destination but also has to return along the original path back since the destination peer cannot initiate a direct connection to a peer behind NAT. Some NAT traversal protocols [7, 8, 29] have already been enacted to achieve inter-connection between two NAT-ed devices. Yet, we have found literatures seldom consider the organization open peers and NAT-ed peers properly. Most of the NAT traversal solutions do not consider the impact of NAT traversal to the performance of a peer-to-peer system. Either hole punching [15-17, 31], port forwarding [18], or relay [27][28] requires a "Middle Man" to assist setting up direct connection between two NAT-ed nodes. This paper introduces a single point of failure degrading the efficiency and scalability of a peer-to-peer system in practice.

This paper aims to resolve the above challenging problem by properly accommodating NAT-ed nodes in a peer-to-peer routing system such that the performance degradation on the original protocol is minimal. To solve the problem, we devise a peer strongly connected model to coordinate NAT traversal and session maintenance such that the model can meets the following design requirements which are common to peer-to-peer system design:

- The increase of search hops by introducing NAT traversal should be minimized;
- The impact on existing routine algorithm and message delivery rate should be minimized;
- Design should be reliable and scalable.

We applied our open / NAT-ed peer organization methodology in a concrete peer-to-peer protocol – Kademlia which is a widely studied peer-to-peer routing protocol and mostly used in peer-to-peer file sharing applications.

The rest of the paper is organized as follows: Section 2 reviews the related work of peer-to-peer NAT traversal and NAT-ed peer-to-peer network. In section 3, system model is established which includes the conditions of models designed and a brief introduction of Kademlia protocol. The detail of our model design and key algorithm adaption are rendered in section 4, followed by the comparison of two existing models. Especially we give a mathematical definition of traffic load regarding message transmission on each open node to facilitate the comparison. Section 5 evaluates our model and compares it with other models in delivery rate, search hops and traffic load. The characteristics of our model are summarized and future improvements are indicated in the last section.

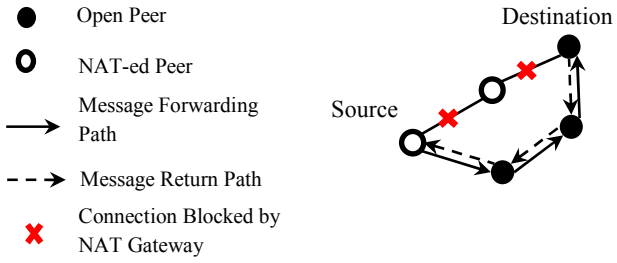


Figure 1. Illustration of inefficient routing in NAT environment.

II. Related Work

Specific design of NAT-ed peer integration scheme in peer-to-peer can be found in [21][13][30][14] and [22]. [21], [13] and [30] are similar in that open peers are fully connected and organized to a structured network in which messages can be efficiently delivered. We call it a rendezvous model. In this model, NAT-ed peers have to choose one or more open peers to attach. Communication between any two NAT-ed peers relies on the proxies of open peers the NAT-ed peers attach to. It will be discussed in section IV – E (2) in detail that in his model system load is heavily skewed to open nodes and the capacity of NAT-ed peers is not fully utilized. Distinction between [13] and [30] focuses on performance optimizations. [21] utilizes openDHT as a specific layer to store and retrieve NAT information for later traversal. Unfortunately, there is no detail about the organization of open nodes and NAT-ed nodes for efficient NAT traversal. Close-by relay is firstly proposed in [13]. It clusters open peers into autonomous system (AS) level, country level and continent level. In [13], location proximity algorithm is introduced to attach a new joined NAT-ed peer to a nearby open peer as its communication proxy of message transfer. SMBR system in [14] outstands in strategies of messages transfer between NAT-ed peers based on type of messages. Small size messages such as routing or control message are relayed by intermediate peers while big data transfer has to rely on direct connection between two NAT-ed peers. Authors in [22] suggest a hierarchical structure to divide the scope of application into several "communities" in which a NAT traversal can be handled by open peers in the same community. Inter-community NAT traversal can be achieved by the collaboration of any two open peers located in these two communities where a center server is required in this approach for open node query by NAT-ed client in the same community, creating a single point of failure in the peer-to-peer system.

Other proposals of NAT traversal in peer-to-peer network either requires changes of the NAT device setting [23], or re-designs the routing protocol in IP layer [24] [25] for NAT traversal friendly hardware solution. Hardware level solution assumes users are equipped with certain level of professional knowledge or a new protocol can be widely accepted by the industry. However, users may not allowed to configure the NAT device due to corporate policy and redesign of the routing protocol in IP layer will even incur a high cost on hardware redesign, manufacturing, upgrade and redeployment, which impacts the whole production chain.

III. System Model

A. NAT-ed Peer-to-peer Network

The object network consists of a set of computers which are called nodes or peers which are interchangeable. Some of them are behind NAT devices such as NAT routers or firewalls. Others are located in public network called open peers. In the system, we further differentiate proptery NAT devices. Some of the NAT devices are easily to be traversed. Others are not peer-to-peer friendly, and can only connect to the external network through server or computers in public domain as relays. Such dividing of NAT device characteristics leads us to further classifying the nodes into three types: open peers, NAT-traversable peers and non-NAT-traversable peers. One NAT-traversable peer is able to directly connect to another NAT-ed peer assisted by an open peer or a bootstrap server. Non-NAT-traversable cannot set up direct connection to any other NAT-ed peers. No peer can send message to a Non-NAT-traversable peer without solicited request. To participate in a peer-to-peer, a non-NAT-traversable peer has to attach to an open peer as a pure client. It does not possess server functions such as message routing. Thereafter, the NAT-ed peer in the models introduced below only denotes NAT-traversable peer if no explicit indication. According to literatures we have investigated [18-21], open peers take a small portion of the internet in real life around 10%-20%. In this paper we consider different ratio of open node to NAT-ed node from 0.001:0.999 to 0.2:0.8 to show the robust of our scheme over others.

Details of NAT traversal is not discussed in this paper. They can be referred to STUN protocol [7], TURN protocol [8] and ICE protocol [29]. We assume two NAT-traversable peer can be connected with any the traversal techniques. We also assume NAT type can be discovered by any means, which is out of the scope of this paper as well.

B. Briefing of Kademlia Protocol

Kademlia protocol is a DHT-based (distributed hash table) structured peer-to-peer overlay protocol which is very efficient in terms of <key, value> distribution and lookup. Detail of the protocol can refer to the original paper [10]. Here we give a concise introduction. In Kademlia, each peer is assigned with a unique 160-bit node ID. The distance of two nodes is calculated with XOR operation on the node IDs. Two distances thus are able to be compared with their absolute values. The neighbor node list in Kademlia protocol is composed of k-buckets. In the i-th k-bucket of a node, there records at most k neighbors' location sharing the same i prefix bits of the node ID. Theoretically there can be up to 160 k-buckets in each node. Limited by the network size, the actual number of k-buckets in node is expected to be log2n where n is the network size. Data will be hashed to a 160-bit-long key before it is distributed on the network. To store data into a Kademlia network, a node will select k neighbors whose node IDs are closest to the key and sends STORE message to these nodes. Each of them returns another list of k nodes closer to the key than themselves. The initial node then queries these new candidate nodes about their view of neighbors and receives k-node lists from them. The lookup process will continue iteratively until no node closer to the key can be found in the network. Then the initial node picks the k closest nodes from the search result and replicates the data to them. Later when another node needs to query the distributed data, it issues a FIND_VALE message to look up the same key. The lookup process is similar as storing except that the search will return immediately as long as the data can be found in one node. To prevent too much overhead from parallel searching process, each time the number of query is limited to α which is a system parameter for performance tuning.

Like other DHT-based protocol such as Chord or Pastry, the number of hops in Kademlia protocol is upper bounded to O(log2n) (n is the size of network) which makes it very efficient and scalable and thus widely used in real life application.

IV. NAT-ED PEER ORGANIZATION MODEL

A. *Peer Strongly Connected Model*

Peer strongly connected model is a symmetric model in which any two peers can communicate with each other by initiating a message from one open / NAT-ed peer to another open / NAT-ed peer, as shown in Fig. 2(a). In this model, NAT-ed peers do not rely on open peers. They can connect to other NAT-ed peers if both of them are NAT traversable. Each peer has a neighbor list which contains IDs and endpoints of other known open peers or NAT-ed peers. For NAT-ed peers in the neighbor list, a long term session is always maintained by sending keep-alive UDP packets periodically between each other after NAT traversal has been made for direct connection. Connections also need to be maintained between NAT-ed peers and open peers by sending keep-alive UDP packets since open peers cannot initiate a connection to any devices behind NAT. Otherwise, the symmetric property of this model will be lost.

There are two rules regulating the long term session maintenance.

1) Each NAT-ed node must periodically send keep-alive messages to all the neighbor nodes. In Kadmlia, they are all the nodes on the k-bucket table.
2) Each node receiving the keep-alive message has to replied it if the sender is on its neighbor list, too.

These design rules leads to an invariant below.

Invariant 1 *Direct connection can always be set up between two nodes if they have a common neighbor regardless of the type of the common neighbor whether it is an open node or a NAT-ed node.*

This invariant is proved by an illustration in Fig. 2(b). In this scenario, let us assume the node A, B and C are NAT-ed which is the most severe case. Both node A and B has a common neighbor C with which node A and B maintain long term session by exchanging keep-alive messages. If node A needs to directly connect to node B, it will send a request to node C. Node C will forward the request to node B which then responds to node A via node C to agree the connection. The endpoint information of node A and node B can be carried in the request and response messages allowing node A and B sending messages to the correct port number open on the NAT device of the destination node. Once the messages arrive at the correct port number which follows the mapping rule within the respective NAT devices, it will be forwarded to the node in the private network. (Usually the mapping rule needs an outgoing record from the corresponding node in the private network, which requires node A and node B sending multiple connection packets to each other simultaneously.)

B. *Routing Style Adaption*

Benefiting from Kademlia protocol's iterative routing style, routing in this model does not rely on open peer or specific NAT server. Iterative routing is different from recursive routing in how a routing message is forwarded to the next node approaching to the destination node [11]. They are shown in Fig. 3. In recursive routing (in Fig. 3(b)), the request message will be routed neighbor by neighbor until one node holds the target data and returns it to the source node via end-to-end connection. However in our model, direct connect cannot be set up between any two nodes if the initiator is behind NAT gateway and long term session is not maintained between them. Therefore the destination node can only communicate to the initiator with intermediation in recursive routing style, resulting in low efficiency.

In an iterative routing style such as Kademlia protocol (in Fig. 3(a)), direct connection can be established iteratively with the help of neighbor nodes in the peer strongly connected model. When a node queries one of its neighbors for a certain data and retrieves a node list of candidate nodes from the neighbor, this neighbor can help the source node and candidates establish direct connection since the neighbor node maintains both session to the source node and the candidate node. After the source node directly connects to the candidate, it adds the candidate node either into its neighbor list or temporary neighbor list if the regular one is full, and the routing operation proceeds by querying the candidate for the data. If the candidate does not have the data either, it will return another list of candidate nodes to the source. To be more specific, we call the nodes in the new candidate list second-level candidate and the one holding these candidates are first-level candidate. The first-level candidate can help the source node and second-level candidate nodes set up direct communication channel traversing through their NAT gateways.

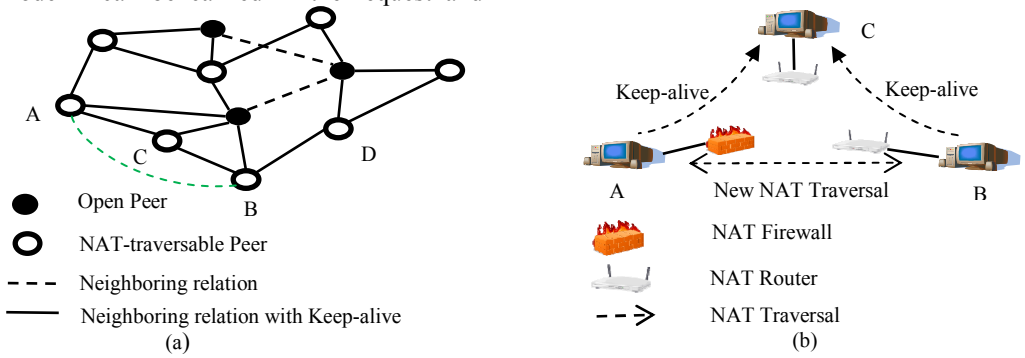


Figure 2. NAT-ed peer strongly connected model. (a) Peer organization structure, (b) NAT traversal illustration mediated by the common neighbor.

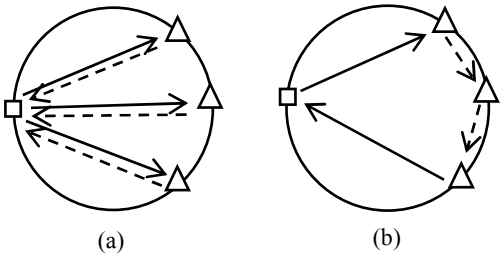


Figure 3. Routing style. (a) Iterative routing , and (b) and Recursive routing.

C. Neighbor-ship Adaption

The neighbor relationship established by temporary connection may not symmetric. One node can have the other node in its neighbor list while the other node may not have since the opposite's neighbor list may have already been full. In such case, one node is in the regular neighbor list and the other is in the temporary neighbor list. Temporary neighbor relation can expire either after the data has been transmitted, the "middle man" work has finished or the preset fixed period has been passed. In our model, we drop the temporary relation with a graceful approach. All NAT-ed nodes maintain connection with each other by exchanging keep-alive messages for long term session maintenance. A node will send keep-alive messages to only the nodes on the regular neighbor list. If a node does not receive a session maintenance message from a neighbor node, it means either the neighbor is failed (either gracefully or abruptly) or the temporary neighbor relationship is expired. The silent neighbor thus will be removed from the neighbor list.

D. Algorithm Adaption

We adapt the NAT traversal to Kademlia protocol to allow NAT-ed nodes participating routing activities. Kademlia protocol has algorithms on four types of messages: PING, STORE, FIND_VALE and FIND_NODE. Every node sends PING messages to neighbors to detect their up state. STORE message is to find a set of suitable nodes and deploy new data on these nodes. FIND_NODE and FIND_VALE messages are transmitted over the network to lookup k nodes which are closest to the given key. In FIND_NODE, the value of the key is the node ID which is hashed from the node's endpoint. In FIND_VALUE, the key is generated by hashing the value of the real data or its abstraction. Parameter k is preset as one of the system parameter of Kademlia protocol. Here we abstract the common part of STORE, FIND_NODE and FIND_VALUE algorithm in looking up k nodes closest to a hashed key and call it the FIND message routing algorithm. The FIND algorithm contains three key sub-routines which are FIND, ROUTE and ROUTE_RESPONSE.

- The FIND sub-routine initiates node finding messages (called FIND messages) to the k nodes whose IDs are closer to the key and sending α messages in parallel where α is a system parameter for performance tuning.
- The ROUTE_RESPONSE sub-routine replies the source node with k closest nodes whose IDs are closer to the key than itself from the local neighbor list.
- The ROUTE sub-routine is triggered when the response from neighbor nodes are receive. It adds the replied node list into candidate list and sorts the candidate list by calculating their distance to the key. It also checks whether all the candidates are queried. If not, it pops out the top member from the list and sends the FIND message to the candidate. Otherwise it returns k closest nodes to the key as FIND result.

The request – response in routing process will be iteratively executed until no node can be found that is closer to the key by ID.

The adapted FIND algorithm in our model mainly changes the ROUTE function. To handle temporary neighboring relation for NAT traversal, we also add two key routines: ADD_INTO_NAT_TRAVERSAL_LIST and RETRIEVE_NAT_TRAVERSAL_INTERMEDIATE.

- The new ROUTE sub-routine will now extract the list of candidate node returned by the neighbor from the message in the form of <candidate ID, candidate endpoint> after the source node retrieves the response from neighbor node. For each candidate node in the list, if it is not an open node, it will be added into a temporary neighbor list for NAT traversal later. When the local node needs to send a FIND message to a temporary neighbor, NAT traversal has to be made with the help of an intermediate node before sending the message.
- The ADD_INTO_NAT_TRAVERSAL_LIST sub-routine adds a NAT-ed candidate node into a temporary neighbor list which maps the candidate to an intermediate node. Here the intermediate is the responded neighbor.
- RETRIEVE_NAT_TRAVERSAL_INTERMEDIATE returns the intermediate node assisting NAT traversal from the temporary neighbor list by using the candidate node ID as the mapping key.

As in the original Kademlia algorithm, the ROUTE function pops up the first node from the candidate list for further query. Before sending the FIND message to the new candidate, a NAT traversal has to be made if the candidate is behind NAT gateway. The source node asks the intermediate node to set up a tunnel to the candidate node. As long as the tunnel is successfully established, the mapping of candidate to intermediate node can be removed from the temporary neighbor list and FIND message can be sent to the candidate node as usual.

E. Model Comparison

We compare our model with other two prevailing approaches of organizing open and NAT-ed nodes to show the advantage of our model over them.

1) Simple Integration Model

In Kademlia-based application like eMule [12] or BitTorrent (using MainlineDHT [15] which is a variant of Kademlia protocol), there is no special treatment of NAT-ed node different from open nodes. Following the original Kademlia protocol, each node can have both open nodes and NAT-ed nodes in its neighbor list whereas it can only

communicate with open nodes. In an un-adapted Kademlia protocol as shown in Fig. 4, search success rate and message delivery rate will be decreased to a very low extent due to inaccessibility of NAT-ed neighbors along the FIND message forwarding path, which will be shown in the evaluation section. Delivery rate degradation results in that data stored on the k node closest to the key of the data cannot be found by FIND operation, because the search path to these k closest nodes are largely blocked by NAT-ed nodes in the middle. Particularly in BitTorrent application, an NAT-ed node can only retrieve resource from neighboring open nodes and in reverse an open node is unable to initiate a connection to a NAT-ed node, which causes NAT-ed node hardly get chance to share their resource with other nodes and become resource leech.

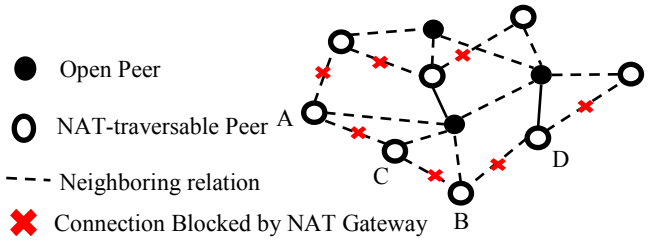


Figure 4. Simple integration model structure.

2) *Rendezvous Model*

Rendezvous model uses open peers to perform message routing and resource lookup. NAT-ed peers attach to open peers only and no two NAT-ed peers connect together, as illustrated in Fig. 5. An open peer connected by a NAT-ed peer is called a parent peer to the attached NAT-ed peer. This model resembles a two-layer or superpeer model and currently widely adopted in most of the peer-to-peer system [21][13][35] for its simplicity. When a new NAT-ed peer joins the network, it will ask the bootstrap node to find an open peer to become its parent peer and then attach to it. The bootstrap node either randomly selects an open peer [13][35] or looks up an open peer based on locality-awareness strategy [21] over the open peer network. The parent node location will then be replied to the new peer by the bootstrap node. On receiving the reply, the new sends request for affiliating to the parent node. The parent node will acknowledge the affiliation according to its own workload. In case the open node exceeds its maximum capacity, it will reject the request and the new node has to look for a new parent by delegating the bootstrap node. Usually to prevent bootstrapping overheads, the bootstrap node will search for several candidate parent nodes and the new node will ask them one after another until an open node is available.

Rendezvous model can minimize communication overhead by reducing new NAT traversal in routing. However, load of traffic is heavily skewed to open peers if the ratio of NAT-ed peer to open peer is high. (In fact, investigation [2] shows open peers only constitute a small portion of the whole network.) To facilitate the comparison of traffic load in different models, we define the traffic load with the rate of messages that is transmitted on each open node in one cycle of FIND operation in Kademlia network. We use a probability model below to estimate the relationship of the network size (n) to number message transmission (L):

$$L(n) = \left(\frac{1}{n} \times \log_2 n + \left(1 - \frac{1}{n}\right) \times \frac{\log_2 n}{n}\right) \times 2 \qquad (1)$$

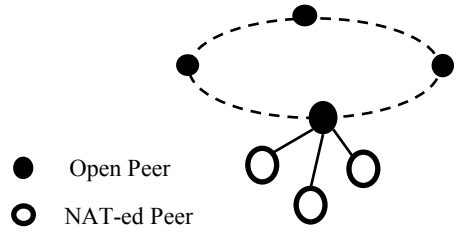


Figure 5. Rendezvous model structure.

When the of network size increases to a great order of magnitude, the factor $(1/n)$ is approaching 0 and function (1) can be simplified to

$$L(n) = \left(\frac{1}{n} \times \log_2 n\right) \times 2 \qquad (2)$$

In the rendezvous model, since most of the messages are transmitted by open nodes, we can approximate the load by substituting the network size n with open node number N. We can see L(N) is always greater than L(n) in function (2) since N is smaller than n. It means the rate of message transmitted over each open node (L(N)) in the rendezvous model is greater than the one in the peer strongly connected model (L(n)) in which the load is fairly distributed to all the peers (including open and NAT-ed peer). The load comparison of different model is shown in the evaluation section with our test data.

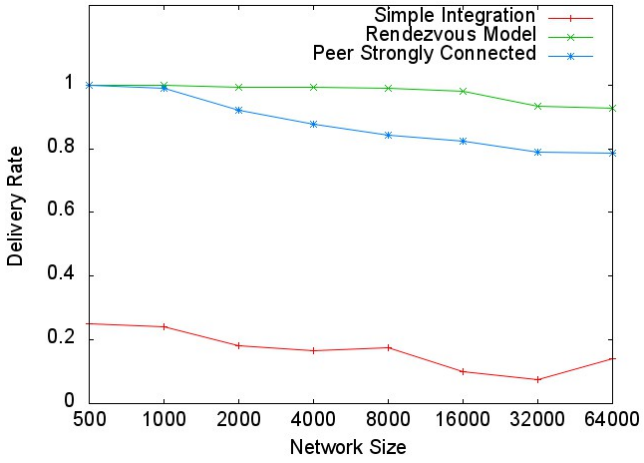
V. EVALUATION

We run our algorithm and do the comparison with other models in PeerSim simulator [17] with an extension version of Kademlia protocol implementation [20]. PeerSim can simulate up to 106 nodes in one experiment and can run on a single computer. Simulation in PeerSim can be implemented into one of the two models: the cycle-based model and event-based model. Event-based model is more controllable of configuration and closer to real life situations. We developed our simulation system with event-based model.

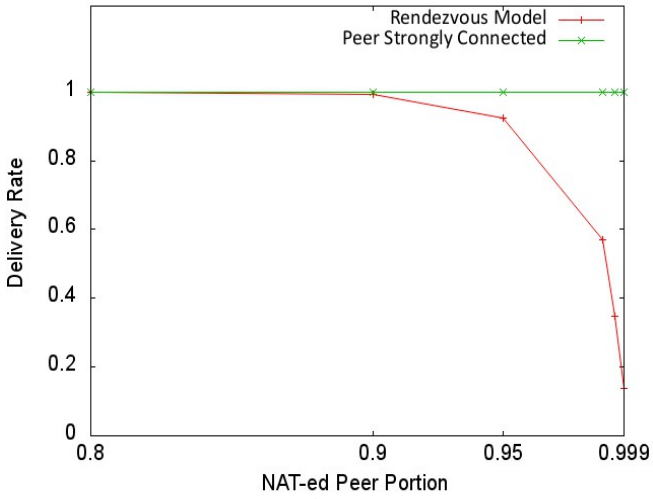
A. Delivery Rate

Delivery rate reflects the chance a message can reach the destination of a FIND operation. In Fig. 6 shows the delivery rate when the network is in unsteady state in which each node only has limited knowledge of other nodes. We firstly run the test in a NAT rate of 80% (of which 20% are open nodes) and change the network size from 500 to 64,000 exponentially in factor 2. Fig. 6(a) shows the result of the test case that the delivery rate in simple integration model is as low as 0.2 while the other two are closing to 1. In the Fig. 6(b) we add the test pressure by decreasing the number of open nodes. The delivery rate of rendezvous model drops sharply when the portion of open nodes decreases lower than 10%. Fig. 7 shows test result of the three models when the network is in steady state in which a node has a big and stable view of the whole network. It can be clearly seen that the delivery rate of the peer strongly connected model is much higher and more stable than the simple integration model and also

than the rendezvous model when the NAT-ed rate is higher than 90%.

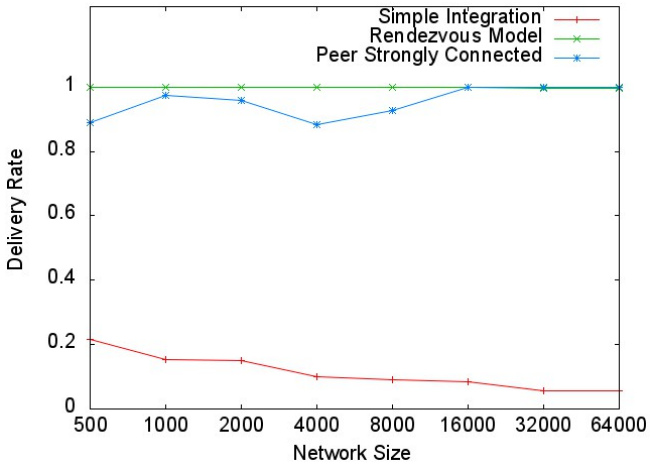


(a)

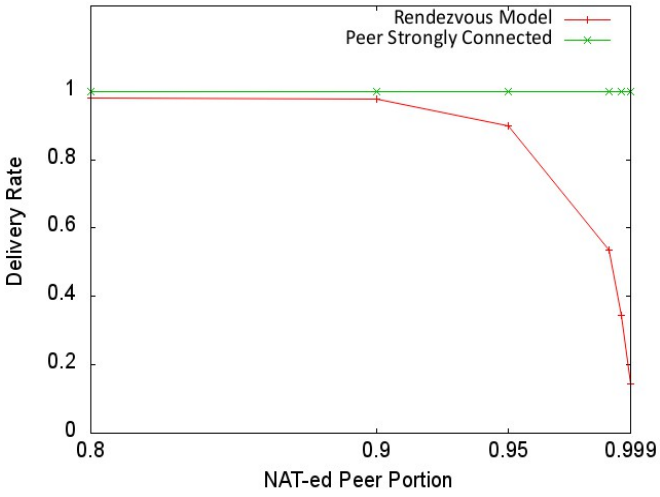


(b)

Figure 6. Delivery rate in network unsteady state. (a) Fix the NAT rate and change the network size; (b) Fix the network size and change the NAT rate.



(a)



(b)

Figure 7. Delivery rate in network steady state. (a) Fix the NAT rate and change the network size; (b) Fix the network size and change the NAT rate.

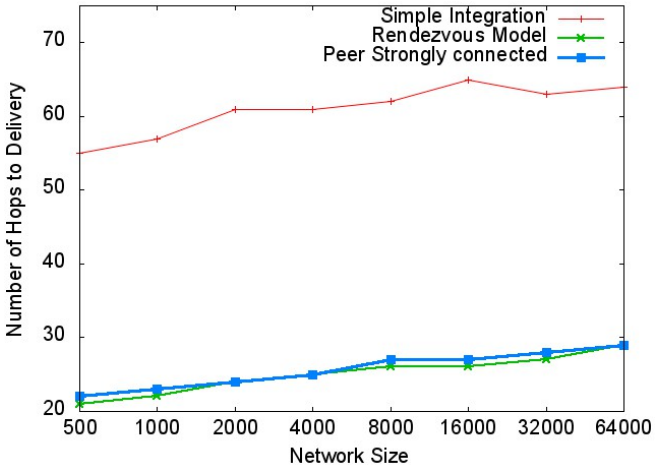


Figure 8. Search hops in steady state.

B. Search Hops

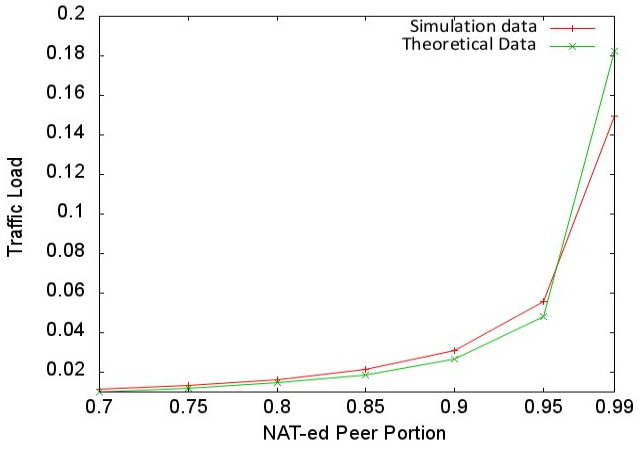
Test case of search hops measure is run in steady state condition and 80% of NAT rate. This test case is to validate the scalability of the three models. Following the original Kademlia algorithm, the number of hops an initial node needs contact scales in the form of $\lceil \log(n) \rceil + c$ where n is network size and c is a small constant. From Fig. 8 we can see that the hop number is proportional to the logarithmic value of network size (represented by the abscissa) and c is around 20 for peer strongly connected model and rendezvous model, and 55 for simple integration model.

C. Traffic Load

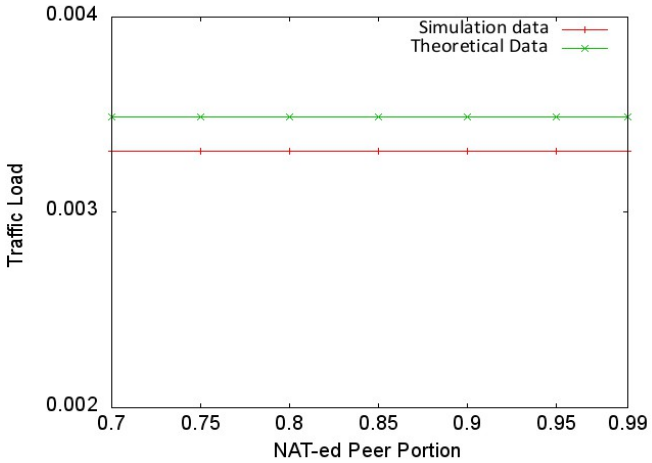
Load scalability is to test the rate of message transmitted by each open node per FIND operation. Fig. 9 shows the theoretical prediction generated from function (1) highly matches the measure data when the NAT-ed rate is lower than 95% in rendezvous model (Fig. 9(a)). In the peer strongly connected model, we can see the traffic load of opens does not related with the portion of open peers since it is shared by all the (open / NAT-ed) nodes in the

network such that the plotting of the data remains constant (Fig. 9(b)).

Fig. 10 shows the test scenario that portion of open nodes is fixed to 10% while the network size changes. In both model we can see that the traffic load decreases with the scaling up of the network size. This is because the number of messages increases in the form of log(n) where n is the network size. We can see the traffic load of open peers in peer strongly connected model is as one order of magnitude low as in the rendezvous model.

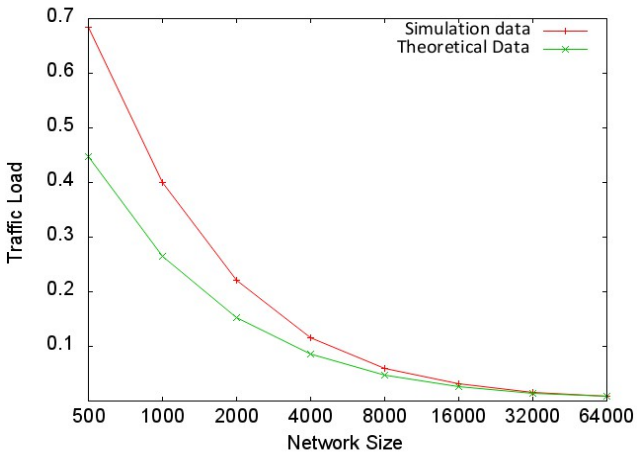


(a)

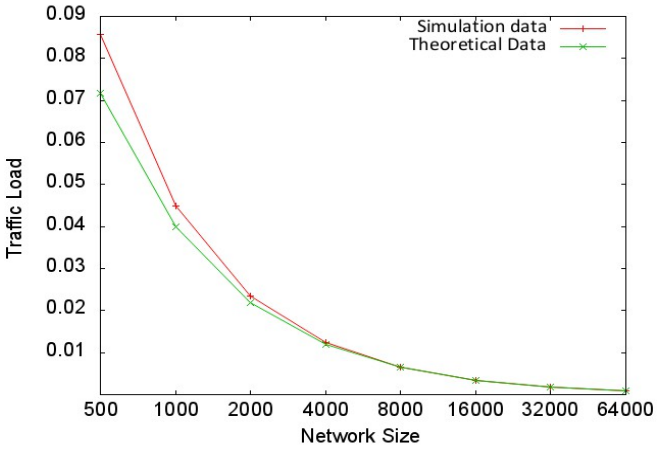


(b)

Figure 9. Open peer dynamic load to NAT-ed peer portion in a network with16000 nodes. (a) Rendezvous model and, (b) peer strongly connected model.



(a)



(b)

Figure 10. Open peer dynamic load to network size in with 10% NAT-ed peers. (a) Rendezvous model, and (b) peer strongly connected model.

VI. Conclusion & Future Work

This paper proposes a peer strongly connected peer organization model on top of Kademlia protocol in the environment where NAT gateway is functioned in a peer-to-peer network. In this model, NAT-ed peers participate routing and data lookup activities by maintaining long term sessions with neighbors after NAT traversals have been made. The long term maintained session can help two non-neighboring nodes establish direct connection regardless of whether they are open or NAT-ed nodes, distracting the load of open peers in rendezvous model.

Our contribution in this paper includes incorporating NAT traversal and NAT traversal session maintenance into the Kademlia protocol such that the impact to the original protocol is reduced to minimal. Evaluation result shows that in the proposed model the delivery rate and scalability are higher than the simple integration model and rendezvous model. Our model remains as good search performance as the theoretical bound of original Kademlia protocol even in a network with extremely high NAT-ed rate (which is higher than 90%)

Our future work includes reduction of the overhead incurred by new NAT traversal creation in routing process. Possible improvement is to cache NAT links which even represent a temporary neighboring relationship. Another undone part is the integration of NAT type check into part of NAT traversal and peer organization model for model completeness and minimum of the dependency of out-of-band component.

Acknowledgement

This research is partially supported by the University of Macau Research Grant No. MYRG156(Y2-L2)-FST11-GJZ.

Reference

[1] Audet, Francois, and Cullen Jennings. *Network address translation (NAT) behavioral requirements for unicast UDP*. BCP 127, RFC 4787, January, 2007.

[2] D'Acunto, L., J. A. Pouwelse, and H. J. Sips. "A measurement of NAT and firewall characteristics in peer-to-peer systems." Proc. 15-th ASCI Conference. Vol. 5031. Advanced School for Computing and Imaging (ASCI), 2009.

[3] Lua, Eng Keong, et al. "A survey and comparison of peer-to-peer overlay network schemes." *IEEE Communications Surveys and Tutorials* 7.2 (2005): 72-93.

[4] Ripeanu, Matei. "Peer-to-peer architecture case study: Gnutella network." Peer-to-Peer Computing, 2001. Proceedings. First International Conference on. IEEE, 2001.

[5] Jia, Zhaoqing, et al. "Random walk search in unstructured P2P." *Journal of Systems Engineering and Electronics* 17.3 (2006): 648-653.

[6] Gkantsidis, Christos, Milena Mihail, and Amin Saberi. "Random walks in peer-to-peer networks." INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. Vol. 1. IEEE, 2004.

[7] Rosenberg, Jonathan, et al. Session traversal utilities for NAT (STUN). Vol. 4. No. 8. RFC 5389 (Proposed Standard), 2008.

[8] Mahy, Rohan, Philip Matthews, and Jonathan Rosenberg. "Traversal using relays around NAT (TURN): relay extensions to session traversal utilities for NAT (STUN)." Internet Request for Comments (2010).

[9] Muller, Andreas, et al. "Autonomous nat traversal." Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on. IEEE, 2010.

[10] Maymounkov, Petar, and David Mazieres. "Kademlia: A peer-to-peer information system based on the xor metric." Peer-to-Peer Systems. Springer Berlin Heidelberg, 2002. 53-65.

[11] Kunzmann, Gerald. "Recursive or iterative routing? Hybrid!." KiVS Kurzbeiträge und Workshop. 2005.

[12] Kulbak, Yoram, and Danny Bickson. "The eMule protocol specification." eMule project, http://sourceforge. net (2005).

[13] Cuevas, Rubén, et al. "A collaborative P2P scheme for NAT Traversal Server discovery based on topological information." Computer Networks 54.12 (2010): 2071-2085.

[14] Yang, Pinggai, et al. "SMBR: A novel NAT traversal mechanism for structured Peer-to-Peer communications." Computers and Communications (ISCC), 2010 IEEE Symposium on. IEEE, 2010.

[15] Grunthal, Aaron. "Efficient indexing of the BitTorrent distributed hash table." arXiv preprint arXiv:1009.3681 (2010).

[16] Ford, Bryan, Pyda Srisuresh, and Dan Kegel. "Peer-to-peer communication across network address translators." USENIX Annual Technical Conference. Vol. 2005. 2005.

[17] Montresor, Alberto, and Márk Jelasity. "PeerSim: A scalable P2P simulator." Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on. IEEE, 2009.

[18] Holzapfel, Sebastian, et al. "A new protocol to determine the nat characteristics of a host." Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on. IEEE, 2011.

[19] Roverso, Roberto, Sameh El-Ansary, and Seif Haridi. "Natcracker: Nat combinations matter." Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on. IEEE, 2009.

[20] M. Bonani, D. FurlanA. "Kademlia module for Peersim." Technical Report on Web Site http://peersim.sourceforge.net. Written in Academic Year 2009/2010.

[21] Wang, Xugang, and Qianni Deng. "VIP: a P2P communication platform for NAT traversal." Parallel and Distributed Processing and Applications. Springer Berlin Heidelberg, 2005. 1001-1011.

[22] Zhang, Zepeng, Xiangming Wen, and Wei Zheng. "A NAT Traversal Mechanism for Peer-To-Peer Networks." Intelligent Ubiquitous Computing and Education, 2009 International Symposium on. IEEE, 2009.

[23] Wang, Bo, et al. "A novel nat traversal mechanism in the heterogeneous environment." Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on. IEEE, 2009.

[24] Suzuki, Hidekazu, Yuji Goto, and Akira Watanabe. "External dynamic mapping method for NAT traversal." Communications and Information Technologies, 2007. ISCIT'07. International Symposium on. IEEE, 2007.

[25] AVES: http://www.cs.rice.edu/~eugeneng/research/aves/

[26] Liu, Yangyang, and Jianping Pan. "The impact of NAT on BitTorrent-like P2P systems." Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on. IEEE, 2009.

[27] Guha, Saikat, and Neil Daswani. An experimental study of the skype peer-to-peer voip system. Cornell University, 2005.

[28] Baset, Salman Abdul, and Henning Schulzrinne. "Reliability and relay selection in peer-to-peer communication systems." Principles, Systems and Applications of IP Telecommunications. ACM, 2010.

[29] Rosenberg, Jonathan. "Interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal for offer/answer protocols." (2010).

[30] Niazi, Salman, and Jim Dowling. "Usurp: Distributed nat traversal for overlay networks." Distributed Applications and Interoperable Systems. Springer Berlin Heidelberg, 2011.