The Lonely NATed Node

Chad Yoshikawa
University of Cincinnati
yoshikco@ececs.uc.edu

Brent Chun
Intel Research Berkeley
bnc@intel-research.net

Amin Vahdat
U.C. San Diego
vahdat@cs.ucsd.edu

Abstract

In this paper we take the position that current research in the area of distributed systems has all but forgotten about one of the largest collective Internet resources - the NATed node. These are hosts that are behind Network Address Translation (NAT) gateways and are hidden by the fact that they have private IP addresses. We argue that Distributed-Hash Tables [18], P2P systems [5], and Grid Computing[9] could greatly benefit by tapping into this forgotten pool of resources. Also, we give an outline of a service, the Distributed-Hash Queue (DHQ), that can enable these NATed resources to be exploited.

1 Introduction

At least since the days of Networks of Workstations [1], researchers have been attempting to harvest the idle cycles of the desktop. Now, however, these desktop machines are no longer accessible due to the proliferation of NAT gateways. In fact, some studies have shown that there exists a large population of users (in some cases a majority) which is hidden due to NAT gateways. In [11], it was shown that the percentage of *public* hosts that partipated in a video multicast event was only between 7% and 57%. In addition, LimeWire network measurements [12] typically show that the number of hosts accepting incoming connections is less than half that of all total hosts. Of course, this number may be due to a number of factors including firewalls and the personal preference of the user. However, the ubiquity of NAT gateways almost certainly has an impact.

1.1 What Effect Does this Have?

The most obvious and direct effect is that the NATed node's CPU, disk, and (sometimes limited) network bandwidth is not being used by current distributed systems. For example, the Grid [9] applications could greatly increase performance by utilizing the large collective computing power of the NATed nodes. In fact, some specialized Grid applications such as the GIMPs [10] prime search have make use of NATed resources to great effect by finding the largest known Mersenne prime number to date. This application, however, makes use a specialized client which coordinates through a central server. What is needed is a distributed, public service which can enable NATed nodes to coordinate and communicate with one another.

Distributed-hash tables (DHTs) have been used as the basis for P2P filesystems [15] and multicast [3] applications. DHTs themselves are based on key-based routing (KBR) [7] protocols including Chord [21] and Pastry [18], among others. The key facet of KBRs is that they enable large scale networks to be grown while limiting the number of network hops needed for communication to $O(log(N))$. While KBRs and the applications that are built on top of them have proven extremely useful, it has been shown these multi-hop routing algorithms matter most when the network size exceeds a certain threshold. Depending on the application, the threshold can be as small as 100,000 nodes or as large as a billion [17]. In fact, research on KBR protocols [18] regularly report on experimental results from network emulators of 100,000 nodes. Given that one of the largest network testbeds, PlanetLab [4], has 500 nodes, how are we going to build a testbed of 100,000 nodes? Accesing the large pool of NATed nodes may be the answer.

1.2 A Solution

We argue for a general purpose solution to these problems which enables bidirectional communication to NATed nodes. This will make these nodes available to Grid computing engines, DHTs and other P2P systems. The solution has several requirements that we identify below:

- It must be publicly addressible. In other words, there must be a fault-tolerant set of front-end nodes by which private nodes can gain access.

- It must provide network storage. NATed nodes are presumably more dynamic than their public counterparts. Thus, the solution must provide a reliable store-and-forward messaging service.
- It must scale. In order to handle a large population of NATed nodes, the solution must be scalable.

This proposed service shares many of the qualities found in a Delay-Tolerant Network (DTN) [8] including addressibility and network storage/retransmission. This is not a coincidence; we argue that the Internet (due to the NATed nodes) is in fact a challenged network and requires a DTN to function properly.

To accomplish this, we advocate building a DTN for the Internet using a new distributed data structure, the distributed-hash queue (DHQ). The distributed-hash queue is similar to another popular distributed data structure, the distributed-hash table [18]. Both operate on top of a key-based routing protocol, meaning that they provide a mapping from large K-bit keys to data. The main difference is that the distributed-hash queue provides a *push* and *pop* interface vs. the traditional DHT *put* and *get* operations. This allows the DHQ to support messaging from NATed node to NATed node. For example, the sender pushes a message to the receiver's queue and the receiver subsequently performs the corresponding pop operation. More sophisticated request-reply matching can be done by including a tag field on all queue elements, so that a request and its corresponding reply tags match. We have created a prototype implementation of the DHQ system and describe it in the following sections.

2 Background

The DHQ system makes extensive use of the Pastry key-based routing (KBR) protocol. Pastry is used to implement the DHQ name service and to help in replicating queue state. While Pastry is used for the implementation, any KBR protocol would be sufficient. In this section, we give a brief background of the Pastry system. For a complete description, please see [18].

In the most basic sense, Pastry maps 160-bit keys to IP addresses. Thus, given any 160-bit key, Pastry will return the closest IP address to that key. This provides the basis of the DHQ name service, since we need to map queue names (160-bit keys) to the host that owns the queue state.

In the Pastry system, the 160-bit key space is configured in a ring (from 0 to $2^{160} - 1$) and the nodes are distributed along the ring. All nodes are assigned a node ID which consists of a 160-bit key and a IP address. Using a consistent hashing algorithm (e.g. SHA), the IP address is deterministically hashed to a key. In addition to being deterministic, the hashing algorithm also generally provides a uniform distribution of keys. So the nodes are roughly distributed in the 160-bit key space in a uniform manner. For an actual distribution of 8 nodes, see Figure 1.

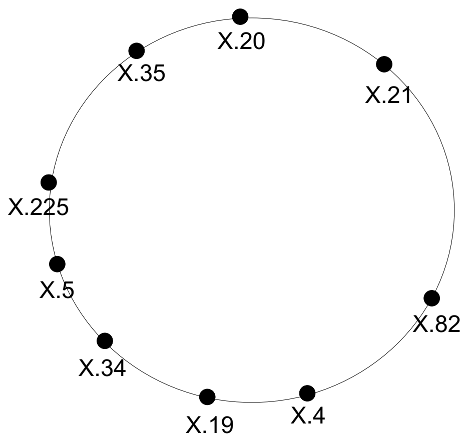

Figure 1: This figure shows eight Pastry nodes with a nearly uniform distribution along the 160-bit ID space. Only the last part of each node's IP address is shown.

Our DHQ exploits two important features of Pastry: the $O(log(N))$ number of hops between sender and receiver and its location-independent names. This allows the DHQ service to scale to a large number of nodes and to proactively move data around faults in the network.

3 Distributed Hash Queues

A DHQ provides durable network storage that can be used to facilitate communication between disconnected hosts. (Think of it as moving send queues and receive queues into the network.) A sending host places network packets into the DHQ and a receiving host subsequently pulls packets from the DHQ. All queues are named using 160-bit keys and a queue lookup (naming) service has been built on top of the Pastry key-based routing protocol. The DHQ prototype runs on top of the PlanetLab network testbed using Java.

The DHQ service consists of N nodes running on the PlanetLab which are publically addressable (i.e. have public IP addresses) and participate in a single Pastry ring (group of cooperating nodes). See Figure 2.

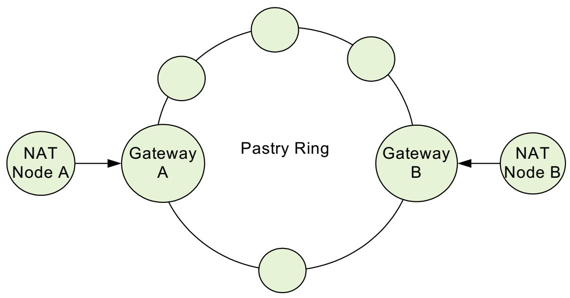

Figure 2: This figure shows the logical structure of the DHQ service. Two communicating NATed nodes, A and B, connect to the DHQ service via the closest respective gateway node.

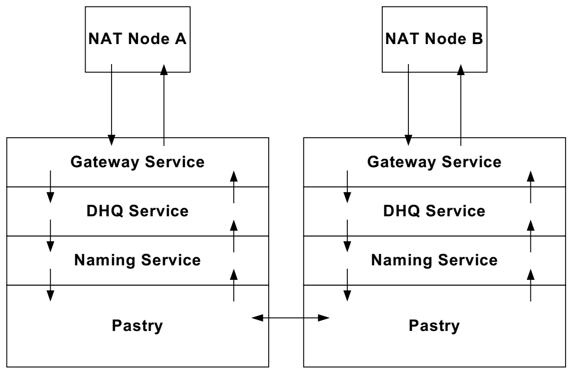

Figure 3: This figure shows the layered structure of the DHQ system. The arrows indicate communication between layers and between entities.

The DHQ system consists of three services: a reliable naming service, a gateway service (for accepting requests from NATed nodes), and the core reliable queueing service. See Figure 3 which shows the layered structure of the DHQ system.

3.1 Reliable Naming Service

All queue operations operate on named queues and must use the naming service in order to locate the queue owners. The naming service provides a mapping from queue names (160-bit keys) to a set of K locations which replicate the queue state for redundancy. In addition, in order to prevent the naming service itself from becoming a single point of failure in the system, names are replicated across K nodes for fault-tolerance. (In practice, K is chosen to be 3.) The name-to-queue-owners binding is replicated by making use of the Pastry *replica-set* feature which finds the K closest nodes to a particular ID. A queue name is first converted to a Pastry key key, and then the Pastry system is used to locate the K node handles which may contain the name binding.

For example, consider a lookup of the queue named "foo". First, the name "foo" is converted into a Pastry key foo_{key} which begins with the hex digits $0x338A$.... A request message for a list of name replicas ($LookupReplicasMessage$) is then sent to the Pastry node with ID closest to the key $0x338A$.... This closest node responds with a list of K replica node IDs. A name lookup is then attempted in parallel to each of these replicas, and the first valid response is returned to the caller. (A similar mechanism is used by the PAST storage system [19].)

3.2 Gateway Service

The NATed nodes do not participate in the Pastry ring, i.e. they do not own a part of the Pastry ID space. This is by design since NATed nodes are assumed to be highly dynamic and would introduce a high churn rate [16] into the system which would decrease stability. Instead, NATed nodes communicate to the Pastry ring nodes using a Gateway Protocol over standard TCP/IP. Commands are sent as human-readable single-line ASCII strings in order to ease parsing and debugging. The Gateway supports commands including the basic *push* and *pop* operations.

NATed nodes attach to Gateway nodes by using a bootstrap process that is based on the nearby-node algorithm from [2] in order to find the closest Gateway node. In our experience, the nearby-node algorithm tended to be biased towards returning the seed node and an improved algorithm based on Vivaldi [6] network coordinates is currently underway.

3.3 Reliable Queue Service

Queues are replicated across a set of K nodes (K is 3 in practice) which are specified upon creation of the queue. They are implemented as priority queues where the message timestamps denote priority. This provides a total ordering on messages given synchronized global clocks. Given weaker time synchronization, however, the priority queues still serve a purpose: they provide a consistent ordering of packets in replicated queues. Therefore, if messages are replicated across a set of K queues, the priority feature ensures that messages will be seen by queue readers in the same order. Queue operations (e.g., push, pop, peek) are multicast to the queue owners. Creating strongly consistent replicated queues is the focus of our current work.

In order to preserve the queue state over long delays, the queue name bindings and queue state are periodically re-replicated. The invariant that we try to maintain in the system is that queues and name bindings have 3 replicas at all times.

4 Related Work

In this paper, we have described a mechanism for allowing communication to a NATed network node with a private IP address. Some related work in this area has attempted to tackle this very problem including AVES [13] i3 [20], and IPNL[14]. The main difference between our approach and these approaches is that the DHQ provides stable network storage.

In AVES, the NAT gateway (and DNS server for performance reasons) is modified in order to support incoming connections to private IP hosts. A public network waypoint address serves as the virtualization of the private IP address, and relays IP packets from a public IP address to the private IP address through the modified AVES NAT gateway. The main constaint on the AVES solution is that it requires gateway software modifications which may not be administratively possible by all NATed clients.

The Internet Indirection Infrastructure (i3) is another possible choice as a substrate for building a DTN. In i3, packets are sent not to an IP address but rather to a rendezvous node identified by an m-bit key, called k. An overlay network then routes data packets to the node associated by $successor(k)$ in the Chord system. Any interested parties can register triggers with the rendezvous node (again, using the key k to identify the rendezvous node). The triggers then forward packets to the interested nodes. What i3 provides through this indirect communication is the ability for recipients to be mobile. However, i3 does not provide network storage for packets as is required by a DTN - packets are simply forwarded by a trigger as soon as they arrive.

The IP Next Layer (IPNL) system [14] provides connectivity to NATed hosts by extending IP addresses to be a triple of a public IP address, realm ID, and private IP address. Other network communication remains the same, so that the IPNL does not handle the long storage delays that are inhernet to DTNs. Also, while IPNL provides a general purpose NAT-to-NAT communication mechanism, it does so by modificating the IP layer and therefore requires router modifications.

5 Conclusion

In this paper we have motivated the need for a new service, the distributed-hash queue (DHQ) which enables NATed hosts to join the traditional Internet. This distributed data structure has been described and shown to support the operations needed for NAT-to-NAT communication. It is our hope that the DHQ can enable distributed systems, such as DHTs and Grid computations, to reach a multitude of new hosts that were previously inaccessible.

References

[1] T. E. Anderson, D. E. Culler, D. A. Patterson, and and the NOW team. A case for now (networks of workstations). *IEEE Micro*, 15(1):54–64, 1995.

[2] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in distributed hash tables. In O. Babaoglu, K. Birman, and K. Marzullo, editors, *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, pages 52–55, June 2002.

[3] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An evaluation of scalable application-level multicast built using peer-to-peer overlays. In *Infocom'03*, Apr. 2003.

[4] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.

[5] L. P. Cox and B. D. Noble. Samsara: honor among thieves in peer-to-peer storage. In *ACM Symposium on Operating Systems Principles*, pages 120 – 132, 2003.

[6] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical, distributed network coordinates. In *Proceedings of the Second Workshop on Hot Topics in*

Networks (HotNets-II), Cambridge, Massachusetts, November 2003. ACM SIGCOMM.

[7] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common api for structured peer-to-peer overlays. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, Berkeley, CA, 2003.

[8] K. Fall. Delay-tolerant networks. In *Proceedings of ACM SIGCOMM 2003*, Karlsruhe, Germany, Aug. 2003.

[9] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150, 2001.

[10] P. Golle and I. Mironov. Uncheatable distributed computations. *Lecture Notes in Computer Science*, 2020, 2001.

[11] Y. hua Chu, A. Ganjam, T. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early experience with an internet broadcast system based on overlay multicast. Technical Report CMU-CS-03-214, CMU, Dec. 2003.

[12] LimeWire Host Count. http://www.limewire.com/english/content/netsize.shtml, 2004.

[13] T. S. E. Ng, I. Stoica, and H. Zhang. A waypoint service approach to connect heterogeneous internet address spaces. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 319–332. USENIX Association, 2001.

[14] P. F. Ramakrishna. Ipnl: A nat-extended internet architecture. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 69–80. ACM Press, 2001.

[15] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz. Pond: The oceanstore prototype. In *Proceedings of USENIX File and Storage Technologies (FAST)*, 2003.

[16] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. Technical Report CSD-03-1299, UCB, Dec. 2003.

[17] R. Rodrigues and C. Blake. When multi-hop peer-to-peer routing matters. In *3rd International Workshop on Peer-to-Peer Systems (IPTPS'04)*, Feb. 2003.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Nov. 2001.

[19] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 188–201. ACM Press, 2001.

[20] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proceedings of ACM SIGCOMM Conference (SIGCOMM '02)*, Aug. 2002.

[21] I. Stoica, R. Morris, D. Karger, M. Kaashock, and H. Balakrishman. Chord: A scalable peer-to-peer lookup protocol for internet applications. In *Proceedings of ACM SIGCOMM*, Aug. 2001.