UDP NAT and Firewall Puncturing in the Wild

Gertjan Halkes, Johan Pouwelse

UDP NAT and Firewall Puncturing in the Wild

Gertjan Halkes and Johan Pouwelse

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, P.O. Box 5031, 2600 GA, Delft, The Netherlands
science@ghalkes.nl, j.a.pouwelse@tudelft.nl

Abstract. Peer-to-Peer (P2P) networks work on the presumption that all nodes in the network are connectable. However, NAT boxes and firewalls prevent connections to many nodes on the Internet. For UDP based protocols, the UDP hole-punching technique has been proposed to mitigate this problem.
This paper presents a study of the efficacy of UDP hole punching on the Internet in the context of an actual P2P network. To the best of our knowledge, no previous study has provided similar measurements. Our results show that UDP hole punching is an effective method to increase the connectability of peers on the Internet: approximately 64% of all peers are behind a NAT box or firewall which should allow hole punching to work, and more than 80% of hole punching attempts between these peers succeed.

Keywords: UDP, NAT, Firewall, Puncturing, Measurements

1 Introduction

In Peer-to-Peer (P2P) systems, computers on the Internet connect with each other in a symmetrical fashion. The computers simultaneously assume both the role of client as well as the role of server. This requires that random computers on the Internet must be able to connect with each other. However, the deployment of firewalls and Network Address Translator (NAT) boxes creates obstacles.

By their very nature, firewalls are meant to regulate what connections are permitted. Moreover, firewalls are frequently configured to allow only outgoing connections, based on the assumption of the client-server model of communication. Of course, in a P2P setting connections may also be incoming, often on non-standard ports, which these firewalls don't allow.

NAT boxes pose a separate but related problem. Although NAT by itself is not meant as a connection filtering technology, it does present an obstacle for setting up connections: the publicly visible communications endpoint (IP and port combination) is not visible for the computer behind the NAT box, and

This work was partially supported by the European Community's 7th Framework Programme through the P2P-Next and QLectives projects (grant no. 216217, 231200).

may even be different for each remote endpoint. To make matters worse, NAT technology is often combined with firewalling to create an even bigger obstacle for connection setup.

The techniques for dealing with NATs and firewalls are well known. For example, the STUN [9] protocol details how a computer can determine what kind of NAT and firewall is between itself and the public Internet. Connection setup can be done through connection brokering or rendez-vous [3]. It should be noted though that the connection setup techniques are most useful for UDP traffic. Setting up a connection for TCP traffic when both computers are behind a NAT/firewall requires unusual or non-standard use of TCP and IP mechanisms, and may rely on specific NAT/firewall behaviour to work [2].

In this paper we present the results of a measurement study of UDP NAT and firewall puncturing "in the wild". Using the known techniques, we have implemented a P2P solution for NAT and firewall puncturing which we then used to measure connection setup success. Our results show that using puncturing, connections between many more peers can be set up, which should ultimately increase robustness in the P2P network.

2 Related Work

The problems related with NATs and firewalls and work-arounds for these problems have been described extensively in previous work. The STUN protocol [9] describes how to detect the type of NAT and firewall between a computer and the public Internet and determine its publicly visible address. To do so, it uses UDP messages (although TCP is also supported) sent to pre-determined STUN servers with public IP addresses. Although the STUN protocol does not provide guarantees about whether the address learnt through it is in fact usable for connection setup, the techniques described are the most well-known method for determining NAT and firewall types.

UDP hole punching was extensively described in [3] (although earlier descriptions exist). The idea is that to allow packets to come in from a remote endpoint, the computer behind the NAT or firewall needs to send something to the remote endpoint first. By doing so it creates a "hole" in the NAT or firewall through which communications can proceed. This even works when both sides are behind a NAT/firewall, when both sides start by punching a hole in their respective NAT/firewalls.

In this paper we only consider simple hole punching. More elaborate techniques exist to deal with NAT/firewalls with more complex behaviour [10]. However, these techniques only apply to a small percentage of the NAT/firewalls on the Internet, and are therefore less useful.

Previous studies have shown that a significant portion of the Internet hosts could be usable in P2P networks using the cited techniques [3, 4, 8]. However, none of these studies try to determine to what extent the theoretical ability to connect is actually usable. The study presented in this paper tries to fill that void.

3 Terminology

In the rest of this paper we will use the terminology introduced by the BEHAVE working group [1]. Specifically we will use the following terms and their respective abbreviations (see Figure 1 for a graphical explanation of the mapping types):

Endpoint-Independent Mapping (EIM) The NAT reuses the port mapping for subsequent packets from the same IP address and port to *any* remote IP address and port. So when sending packets to host A and B from the same internal IP address and port, hosts A and B will see the same external IP address and port.

Address-Dependent Mapping (ADM) The NAT reuses the port mapping for subsequent packet from the same internal IP address and port to *the same* remote IP address, *regardless of the remote port*. This means that host A will always see the same external IP address and port, regardless of the port on host A, but host B will see a different mapping.

Address and Port-Dependent Mapping (APDM) The NAT only reuses the port mapping for subsequent packets using the same internal and remote IP addresses and ports, for the duration of the mapping. Even when communicating from the same internal IP address and port, host A will see two different external IP addresses and/or ports when different ports on host A are used.

Endpoint-Independent Filtering (EIF) The NAT or firewall allows packets from any remote IP address and port, for a specific endpoint on the NAT or firewall.

Address-Dependent Filtering (ADF) The NAT or firewall allows packets destined to a specific endpoint on the NAT or firewall from a specific remote IP, after a computer behind the NAT of firewall has sent a single packet to the remote IP.

Address and Port-Dependent Filtering (APDF) The NAT or firewall allows packets destined to a specific endpoint on the NAT or firewall from remote endpoints only after a packet has been sent to that remote endpoint from inside the NAT or firewall.

For the purposes of our experiments there is no difference between ADM and APDM as only single endpoints on the different hosts are used, which we will therefore combine into the abbreviation A(P)DM. Similarly we collapse the definitions of ADF and APDF into A(P)DF if the distinction is irrelevant. In [10] a more extensive subdivision is made, but again the distinctions made are relevant only when considering multiple ports or port-prediction techniques.

4 Implementation

We have implemented a UDP NAT/firewall puncturing scheme as part of the Tribler BitTorrent client. Although the Tribler client does not currently use UDP for setting up connections or exchanging data, we use it as a vehicle to

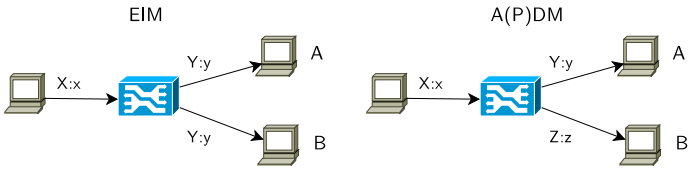


Fig. 1. When a NAT exhibits Endpoint-Independent Mapping (EIM) behaviour, hosts *A* and *B* see the same external IP address and port (Y:y) for a particular internal IP and port combination (X:x). If however the NAT type is Address (and Port)-dependent Mapping, hosts *A* and *B* see different IP/port combinations (Y:y and Z:z). Note that the IP addresses (Y and Z) can be the same, especially if the NAT has only a single external address.

deploy our puncturing test software on real users' machines. The puncturing test builds an swarm, much like the BitTorrent software. However, instead of having a separate tracker, we use a peer at a pre-programmed address and port and employ a form of Peer EXchange (PEX) to find new peers to connect with. The PEX messages include a (random) peer ID, the IP address and port at which the peer originating the PEX message communicates with the remote peer, and the NAT/firewall type of the remote peer. The latter is included so the receiving peer can determine whether it is useful to attempt to connect to the remote peer, as certain combinations of mapping and filtering are not able to connect to each other.

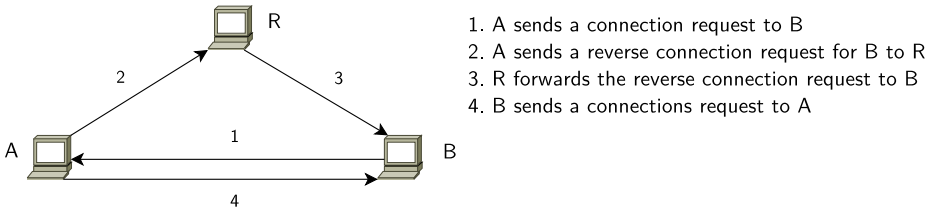


Fig. 2. Rendez-vous for connection setup.

Our implementation tries to make minimal use of centralised components. Therefore the detection of the NAT and firewall type are not done using STUN. Instead, the peers rely on information reported by other peers and by checking if other peers can connect to it without previous communication in the reverse direction. Furthermore, peers use other peers as rendez-vous servers (see Figure 2). If a peer *R* is connected to two other peers *A* and *B*, it can serve as rendez-vous server for them, even if it is itself behind a NAT box or firewall.

It should be noted that due to the generic audience which participated in our trial we expect these numbers to be reasonably representative for generic Internet users. Because using P2P file-sharing software is typically discouraged

in a corporate environment, we do expect a bias towards home users. Home users tend to use less professional equipment that is more prone to misbehaviour, which may negatively impact our connection success rate results.

In the following sections we will describe the tests used by peers to determine their NAT and firewall types.

4.1 NAT Type Detection

To allow determination of the NAT type of the NAT box (if any) that a peer is behind, all peers report the remote address and port they see when a connection is set up. So each peer will receive, from its communication partner, his own external address and port. If the reports from all communication partners are the same (or at least a large majority is the same), a peer will determine that there is either no NAT or the NAT has EIM behaviour. Note that the two cases (no NAT or EIM behaviour) are indistinguishable without a reliable local determination of the local address. Furthermore, the difference is mostly irrelevant. If, however, the reported external IP address and/or port are regularly different, then the peer concludes that it is behind a A(P)DM NAT.

4.2 Filtering Behaviour Detection

The filtering behaviour of a NAT/firewall is detected by checking whether a direct connection request arrives before a reverse connection request arrives. To enable this to work, when a peer tries to set up a connection using rendez-vous, it will always first send a direct connection request to the remote peer. In most cases this direct connection request will arrive at the remote peer before the reverse connection request from the rendez-vous, unless the NAT/firewall behaviour is A(P)DF. So when for a significant fraction of incoming requests the direct connection request arrives before any communication in the reverse direction, the peer concludes that the filtering behaviour is EIF (or there is no firewall, which again is indistinguishable). Otherwise it must conclude that the filtering type is A(P)DF.

In principle it would be possible for the clients to distinguish between ADF and APDF. For example, peers could try to deliberately send a direct connection request to the wrong port. If the filtering is of APDF type, no connection can be setup, and the attempt will always fail. However, if the firewall uses ADF type filtering, the attempt will still succeed (assuming the reverse connection request arrives at the remote peer). Another option is to try to connect to peers behind an A(P)DM type NAT. This should theoretically only succeed for peers behind ADF firewalls. These experiments should be performed several times before concluding one way or the other.

In our implementation however, we have not let peers try to distinguish between APDF and ADF type filtering. We did let A(P)DF peers attempt to connect to A(P)DM peers, such that from the collected logs we can later make the distinction. Because distinguishing between APDF and ADF filtering after the experiments should not provide different results, and the fraction of both

EIM-ADF and A(P)DM peers is small (see Section 5), we felt that including this distinction in the client software would provide little benefit.

5 Results

In this section we present the results for two trials. For the purposes of data collection, the peers in the network logged all interesting send and receive events to a local log file. In the first trial (907 peers), peers would not retry a failed connection attempt immediately. After analysing the results of this first trial, we conducted a second trial (1,531 peers) in which peers would perform up to three retries if connection attempts failed. In the first trial peers regularly sent the collected logs to a central collection server. However, in the second trial we used a less intrusive reporting method which lead to peers with an unfiltered Internet connection being favoured in the results. Therefore we report the market share results from the first test to ensure correct results. Both trials lasted several weeks.

5.1 Market Share

Figure 3 shows the detected NAT and firewall types for 646 out of the 907 peers in the first trial, for which were able to draw a conclusion about the NAT/firewall type from the connections they made. The most dominant type of NAT/firewall is EIM-APDF (52%). This includes both simple firewalls and EIM NATs. Theoretically these can connect to each other through a rendez-vous peer. The fraction of peers that are behind A(P)DM NATs is only 11%. These NAT/firewalls are the biggest obstacle, i.e. they can only connect to EIM-EIF and EIM-ADF peers. The fraction of A(P)DM NATs is expected to go down, as more and more vendors start complying with the BEHAVE RFC [1].

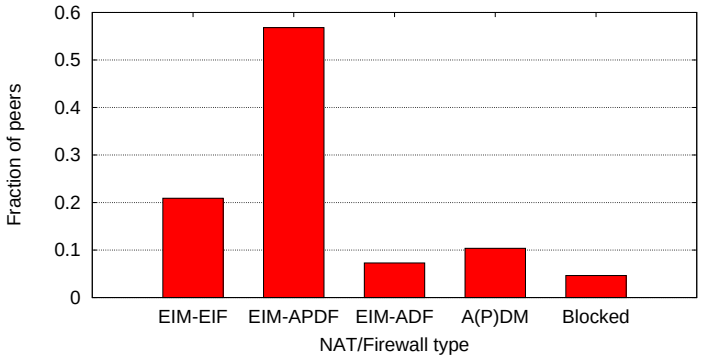
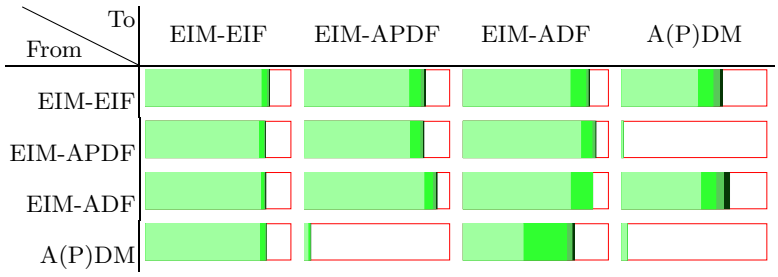


Fig. 3. NAT/firewall type market share

Table 1. Connection success rate per NAT/firewall type. Each bar shows the successful attempts, where retries are shown as increasingly darker shades of green.



5.2 Connection Success Rate

Next we look at the success rate in setting up a connection between two peers. For this we only consider those connections for which we have information on both ends of the connection, and for which we were able to determine the NAT/firewall type of both peers. Our results therefore represent 15,545 connections between 841 peers.

We have split the results out into the different NAT/firewall types we have distinguished (see Table 1). A first thing to note about the results is that even peers classified as EIM-EIF only show an 85% success rate. Detailed analysis of the results shows that this is caused by a small fraction of peers that consistently show poor connectability (see Figure 4). Perhaps they are experiencing high packet losses due to a saturated link. As the puncturing test is running in the background of a BitTorrent client this is certainly not impossible.

To test our hypothesis, we tried to reproduce the packet dropping behaviour in a local test environment using (new) NAT routers. Using TCP side traffic to saturate the links, we were only able to produce small packet loss rates ($\leq 5\%$). To see whether other methods of stressing the routers would produce different results, we also performed a similar test using UDP side traffic. Using large volumes of UDP packets we could get some routers to drop significant numbers of packets (50% or more), or even crash completely.

We must stress that we used newly acquired NAT routers, which may be more resilient to the BitTorrent-like stress test that we subject these routers to. Unfortunately, we currently do not have older router and modem models, which may be more susceptible to dropping packets under high TCP load. Also, although the consumer-grade NAT/firewalls are the prime suspect for dropping packets, they are by no means the only possible point at which packets may be dropped. Therefore we can not discount packet loss as a cause for reduced connection success rates.

Another possibility is that certain routers stop functioning correctly when their mapping tables are full. This is a well-known problem with some older types of routers, when used in combination with BitTorrent. Note that this does not mean they do not function at all, but new connections can usually not be

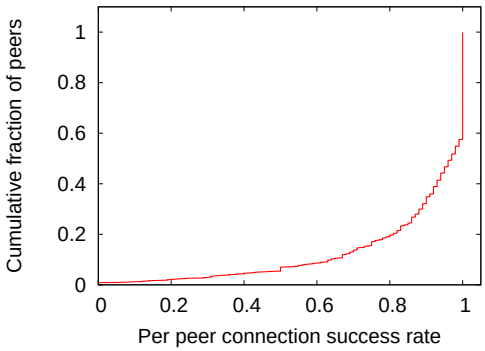


Fig. 4. Cumulative distribution of per peer connection success rate, counting only connections attempts between compatible peers. Connections to peers with lower success rates are not counted against peers with a higher connection success rate.

set up anymore, and existing connections can experience significant throughput reductions. A peer behind such a router may still be reachable over the existing connections, but all new connection attempts will fail, thereby reducing the connection success rate.

The connections between EIM-ADF peers and EIM-A(P)DF peers succeed a little more often than the connections between EIM-EIF peers, but this result is not statistically significant as only 36 peers in our test are behind a EIM-ADF type NAT/firewall.

Connections to and from peers behind EIM-APDF NAT/firewalls succeed a little less frequently than the connections between EIM-EIF peers. This is not entirely unexpected. As already noted in the RFC describing the STUN protocol [9], even though two peers should be able to communicate given the classification determined here, subtleties in the actual implementation may prevent actual connection setup (see also Section 5.2). Conversely, peers which should not be able to connect due to incompatible NAT/firewall types can on occasion setup a connection due to similar implementation subtleties.

Behavioural Subtleties In our analysis of the collected logs, we found several behaviours in NAT boxes that are not well described by the classification we have used so far. For example, we found that several EIM-APDF NATs would occasionally use a different external port (similar to A(P)DM NAT behaviour). This is one reason why these NATs have lower connection success rates.

A second notable behaviour occurs in A(P)DM NATs. The conventional idea is that these NATs will use a different port for each remote endpoint. However, some A(P)DM NATs appear to use a small set of ports repeatedly. So although it is uncertain which of these ports will be used, there is only a small set to choose from. This could explain why sometimes a connection between an A(P)DM NAT and an EIM-APDF NAT/firewall does succeed: because the set is small there is

a chance that the EIM-APDF NAT/firewall actually uses the port that is chosen by the A(P)DM NAT, allowing the packet to arrive. This may also explain why sometimes the connections between two A(P)DM NATs succeed.

A(P)DM to EIM-ADF First Attempt Success One very interesting result can be seen in the connections from A(P)DM peers to EIM-ADF peers. If we assume that a packet sent directly between peers A and B will arrive before a packet sent at the same time but via a third peer, then the direct connection request will always arrive before the reverse connection request has punched a hole. This results in the direct connection request being dropped at the NAT/firewall. If we consider the case where an A(P)DM peer is the originating peer, the reverse connection request sent from the EIM-ADF peer will also very likely not arrive at the port that the direct connection request was sent from, causing the packet to be dropped. This means that given the above assumption, we would expect that connection attempts from A(P)DM peers to EIM-ADF peers will always fail the first time. For the second and later attempts, the firewall on the EIM-ADF side will already have been opened for connections from the A(P)DM peer by the reverse connection request of the first attempt, allowing the direct connection request to pass through.

As we can see from the results, the first connection attempt does succeed in approximately 41% of the cases. The reason for this is that the direct connection request is not always faster than the reverse connection request which is sent through the rendez-vous peer. This is known as a Triangle Inequality Violation (TIV). It is well known that TIVs exist in the round-trip time between nodes [11–13]. Our connection setup is dependent on one-way delay, and not round-trip time, but many of the underlying causes such as routing policies and peering agreements will affect the one-way delay as well. Studies of TIVs in round-trip time seem to indicate that the occurrence of TIVs can be as high as 40%, although lower numbers are more common.

Another possible explanation for TIVs specific to our situation is that there is network equipment which requires some setup time when a packet needs to be sent to a host (or AS) with which it has not recently communicated. The consumer-grade NAT/firewalls are of course a first suspect for such behaviour, but an exploratory test with a small number of such boxes did not find any significant extra delay. The only source of extra delay in communicating with "new" IP addresses that we were able to find is the ARP protocol used in LANs. However, in our situation it is unlikely that the ARP protocol is the cause of TIVs.

Finally, there are other, albeit unlikely, possibilities for the first attempt to succeed. First, it is possible that some EIM-ADF firewalls have very long mapping/filter timeouts. If, after closing the connection, an A(P)DM peer tries to establish a new connection before the mapping/filter has timed out, the first attempt could succeed. To prevent such situations we did not allow new connections within 5 minutes, but this could be too short. Second, the two peers could simultaneously try to connect to each other. However, similarly to the pre-

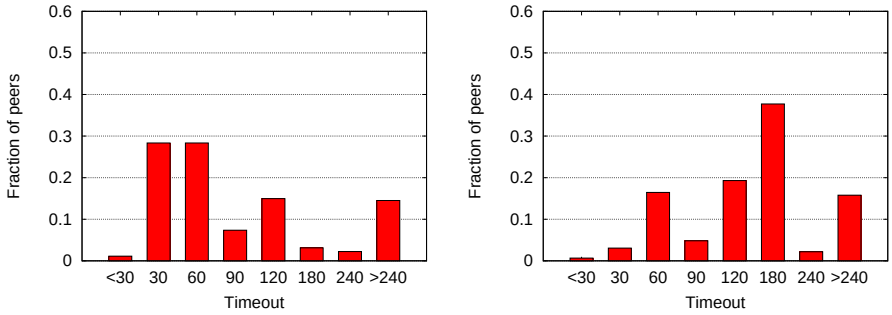


Fig. 5. Mapping/firewall hole timeout without (left) and with (right) handshake.

vious explanation, the chances of such an occurrence are very small. Third, the previously mentioned behavioural subtleties may explain a small fraction of the successful connections as well. If the A(P)DM NAT uses only a small number of ports, this significantly increases the chance that the EIM-ADF peer actually attempts the reverse connection to the port the A(P)DM NAT is expecting, in which case the connection setup will succeed.

5.3 Timeout

A final measurement we made was the timeout for the mappings/firewall holes. This parameter is important when setting the keep-alive interval. To measure the timeout, we sent UDP packets to a pre-programmed IP address and port, after which a reply was sent with a delay. If the reply was received, the timeout was determined to be larger than the delay of the reply. The largest timeout for which a message was received was determined to be the approximate timeout for the mapping/firewall hole.

To ensure a rapid measurement, several messages requesting replies at different delays were sent at the same time from different ports. This setup can give wrong results for some EIM-ADF NAT/firewalls which refresh the timeout for incoming traffic as well as for outgoing traffic. Doing so is a security risk, so we expect few firewalls to behave this way. However, we have for this reason excluded the (small number) of EIM-ADF NAT/firewalls from our analysis.

The graph on the left in Figure 5 shows the results of this measurement. These results indicate that many firewalls employ a fairly short timeout (1 minute or less) for the created mappings/firewall holes. However, some NAT/firewalls, particularly those based on the Linux kernel, use a longer timeout if more than two packets have been sent on a particular mapping/firewall hole, and at least one packet has passed in both directions. We therefore also measured the timeout where we include an initial handshake before requesting the delayed reply. The results are shown in the graph on the right in Figure 5. The results are markedly different. Most of the NAT/firewalls that have a 30 second timeout without the handshake now use a much longer timeout. The default value in the Linux

kernel is 180 seconds, which may explain the large fraction of peers with that timeout when using an initial handshake. These differences between single packet timeouts and timeouts after an initial handshake are also demonstrated in [5].

6 Discussion

In the previous section we have seen that NAT/firewalls do not always behave as simple as the traditional classifications have suggested. The result of this is that both connections that are expected to succeed do not, and vice versa. Although there seems to be a convergence towards BEHAVE compatible NAT behaviour (as can be seen by the very small number of A(P)DM NAT boxes that are available from stores today), it is unlikely that all behavioural anomalies will fully disappear.

Our results confirm that over three quarters of the peers on the Internet (79%) are not directly connectable. The lack of connectability has long been recognised as a problem in P2P networks [3, 6], especially in video streaming [7]. However, using a simple rendez-vous mechanism, a further 64% of all peers could potentially set up connections between themselves when using UDP. This would reduce the connectability problem to a mere 15% of peers, practically eliminating the problem for most P2P applications.

Some have suggested that the problem of NATs obstructing connectability will go away with the introduction of IPv6. However, it should be noted that most of the problems today are not so much caused by the NAT behaviour, but rather by the filtering. The vast majority of NAT/firewalls today are of the EIM-A(P)DF type, which means that their externally visible endpoint is constant and therefore not a big obstacle for connectability. The remaining problems are caused by the filtering behaviour, and it is likely that home routers will continue to include firewalling by default, even when the switch to IPv6 is made. As such, the connectability problems will continue to exist and the puncturing techniques studied in this paper will remain an important tool to overcome these problems.

7 Conclusions

In this paper we have presented the results of a real implementation of UDP NAT and firewall puncturing, running on random Internet users' machines. Our implementation makes minimal use of central components. Most notably, peers detect their NAT and firewall types solely through communication with other peers.

Our results show that connectable (EIM-EIF) peers form a small minority (21%) on the Internet, and that most NATed and firewalled peers (64% of all peers, EIM-A(P)DF) should theoretically be able to set up connections through a simple rendez-vous mechanism when using UDP. Our results also show that connections between these peers can indeed be set up, with a success rate only fractionally lower than for connectable peers. As these hosts are the majority of the peers in Peer-to-Peer networks, we conclude that there is a large opportunity

for UDP based Peer-to-Peer protocols to set up many more connections and therefore create a more robust network.

Finally, our measurements of the NAT mapping/firewall hole timeout show that keep-alive message should be sent at least every 55 seconds to ensure that mappings/holes will remain open on almost every NAT/firewall. This assumes that several messages are exchanged within the first 30 seconds, because some NAT/firewalls extend their timeout if there is more traffic than a simple request and reply.

References

1. Audet, F., Jennings, C.: Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787 (Best Current Practice) (Jan 2007), http://www.ietf.org/rfc/rfc4787.txt
2. Biggadike, A., Ferullo, D., Wilson, G., Perrig, A.: NATBLASTER: Establishing TCP connections between hosts behind NATs. In: SIGCOMM Asia Workshop (Apr 2005)
3. Ford, B., Srisuresh, P., Kegel, D.: Peer-to-peer communication across network address translators. In: USENIX 2005 (Apr 2005)
4. Guha, S., Francis, P.: Characterization and measurement of tcp traversal through nats and firewalls. In: Proc. of the 5th ACM SIGCOMM Conf. on Internet Measurement (IMC 2005). pp. 199–211. Berkeley, CA (Oct 2005)
5. Hätönen, S., Nyrhinen, A., Eggert, L., Strowes, S., Sarolathi, P., Kajo, M.: An experimental study of home gateway characteristics. In: Proc. of the 10th Internet Measurement Conference (IMC 2010). Melbourne, Australia (Nov 2010)
6. Liu, Y., Pan, J.: The impact of NAT on BitTorrent-like P2P systems. In: Proc. of the 9th Int. Conf. on Peer-to-Peer Computing (P2P'09). pp. 242–251. Seattle, WA (Sep 2009)
7. Mol, J., Bakker, A., Pouwelse, J., Epema, D., Sips, H.: The design and deployment of a bittorrent live video streaming solution. In: Proc. of the IEEE Int. Symp. on Multimedia (ISM2009) (Dec 2009)
8. Noh, J., Baccichet, P., Girod, B.: Experiences with a large-scale deployment of stanford peer-to-peer multicast. In: Proc. of the 17th Int. Packet Video Workshop (PV 2009). pp. 1–9. Seattle, WA (May 2009)
9. Rosenberg, J., Mahy, R., Matthews, P., Wing, D.: Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard) (Oct 2008), http://www.ietf.org/rfc/rfc5389.txt
10. Roverso, R., El-Ansary, S., Haridi, S.: NATCracker: NAT combinations matter. In: Proc. of the 18th Int. Conf. on Computer Communications and Networks (ICCN 2009). pp. 1–7. San Francisco, CA (Aug 2009)
11. Savage, S., Anderson, T., Aggarwal, A., Becker, D., Cardwell, N., Collins, A., Hoffman, E., Snell, J., Vahdat, A., Voelker, G., Zahorjan, J.: Detour: A case for informed internet routing and transport. IEEE Micro 19(1), 50–59 (1999)
12. Tang, L., Crovella, M.: Virtual landmarks for the internet. In: Proc. of the 3rd ACM SIGCOMM Conf. on Internet Measurement (IMC 2003). pp. 143–152. Miami Beach, FL (Oct 2003)
13. Zheng, H., Lua, E.K., Pias, M., Griffin, T.G.: Internet routing policies and round-trip-times. In: Proc. of the 6th Int. Workshop on Passive and Active Network Measurement. pp. 230–250. Boston, MA (Apr 2005)