INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 10, NO. 1, MARCH 2005, 1-10

Overcoming NAT And Firewall Connectivity Restrictions In Overlay Multicast

Aditya GANJAM and Hui ZHANG

Abstract- **A large number of overlay multicast protocols have been developed, almost all of which assume universal connectivity between end hosts. However, in reality, this assumption is not valid with widespread use of Network Address Translators (NAT) and firewalls. The impact of NAT and firewall connectivity restrictions on overlay multicast, especially in the application-endpoint setting, has not been seriously considered. In this paper, we argue that it is critical to consider connectivity restrictions because NAT and firewall hosts make up a large fraction of the endpoints, affecting proper functionality as well as performance of overlay multicast protocols. We present several design enhancements that explicitly consider connectivity restrictions in overlay multicast and evaluate the design space and tradeoffs based on real Internet broadcasts and Internet testbed experiments.**

***Index Terms*—Overlay multicast, Network Address Translator, NAT, firewall, peer-to-peer systems**

1. INTRODUCTION

Enabling ubiquitous live video broadcast over the Internet is a key application motivating overlay multicast research. Overlay multicast is an architecture to deploy multicast functionality over the Internet where end hosts, instead of routers, perform all multicast operations. A key strength of this architecture is its flexibility in deployment. One could envision, at one extreme, deployment using highly provisioned and reliable end hosts performing multicast operations, as done in Akamai [2], or the other extreme, deployment using a purely *application-endpoint* model. In the latter scenario, only the end hosts participating in the application perform multicast operations. The application-endpoint model is very encouraging due to its low deployment cost, namely there is no need for infrastructure deployment or maintenance. A number of overlay multicast protocols have been developed, many of them in the context of the application-endpoint model [3-5,7-9,11-13,15-16,18,20]. However, a critical issue amplified by the application-endpoint model that these protocols do not seriously consider are the connectivity restrictions imposed by Network Address Translators (NAT) and firewalls, which hinder and even block communication between certain pairs of hosts.

In [6] we describe our experience with deploying a video broadcast application using overlay multicast. Through several live Internet broadcasts of video content to a real audience, we have observed environments where the fraction of NAT and firewall hosts in an overlay multicast group is as high as 80%. Given that most overlay multicast protocols assume universal connectivity, this property of the Internet poses an important question: *is the performance of an overlay multicast protocol significantly affected by the existence of hosts behind NAT and firewall and if so what techniques an be used to improve performance?*

Through results from 9 events from our Internet deployment and Internet testbed experiments, we show that performance is severely affected if NATs and firewalls are not seriously considered. In this paper, we describe three solutions to accommodate hosts behind NAT and firewall in overlay multicast: Strawman, Basic Contributor and Enhanced Contributor. We observe that in a commercial environment, the Strawman solution, which treats hosts behind NAT and firewall as leaf nodes only, has a high rejection ratio(.43), or fraction of hosts not able to connect to the tree. We show that the Basic Contributor and Enhanced Contributor solutions, which increase connectivity between hosts and make NAT and firewall hosts capable of begin interior nodes in the tree, significantly improve the rejection ratio, where Enhanced Contributor achieves a rejection ratio of 0. In addition, we propose and show the benefits of an optimization to overlay tree construction, called Connectivity-Aware Structuring, that uses knowledge of NAT and firewall hosts to construct trees with increased available resources for these hosts.

2. BACKGROUND

In this section we provide a background on NATs and firewalls, discuss existing solutions to connect hosts behind NAT and firewall and outline a representative overlay multicast protocol which we use in our evaluation.

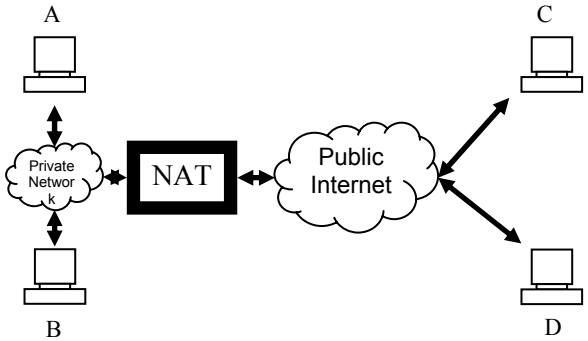

Fig. 1. Addressing issues with NAT.

2.1 NAT and Firewall

There are two network devices that hinder universal connectivity of the Internet: Network Address Translators (NAT) and firewalls. NATs are widely deployed and are used to extend the IP address space by using private addresses and ports within a network and translating to a smaller set of public addresses/ports when communicating outside the network. The private addresses can be replicated at any number of private networks, whereas the public addresses must be unique over the Internet. The translation between a private address/port to a public address/port is created in the NAT when a packet originating within the network passes through the NAT to reach its destination outside the private network. At this point, a packet arriving at the NAT from outside the network that matches the translation inserted into the NAT, will be reverse translated and sent to the destination. For example, consider Figure 1. Host A behind the NAT can send a packet to host C, creating a translation in the NAT, after which, Host C can send a packet back to host A by using the public address and port that the NAT used to translate A's private address and port. These values appear in the header of the packet that C first receives from A. The direct effect of this is that communication cannot be initiated from a host outside a NAT to a host behind a NAT (C cannot initiate communication to A).

Firewalls, in contrast to NATs, are used for security reasons. Firewalls use filtering rules to determine if a packet should be allowed into or out of a private network. The rules include filters for fields such as port number, IP address, transport protocol and can be different for incoming and outgoing packets.

NATs and firewalls impose fundamental restrictions on pair-wise connectivity of hosts in the Internet. In many cases, it is not possible for hosts behind different NATs and firewalls to communicate directly with one another. However, there are specific exceptions, depending on the transport protocol (UDP or TCP), and the exact behavior of the NAT/firewall. Adopting the classification from STUN [10](see Section 2.2, *Full Cone NATs* can receive incoming packets to a port from any arbitrary host once the NAT host sends a packet on that port to any destination. Many hosts can address a host behind a full cone NAT using the same port number. In contrast, *Symmetric NATs* allow incoming packets only from the host that it has previously sent a packet to. Different hosts address a host behind a symmetric NAT using different port numbers. To illustrate, consider Figure 1. In the case that the NAT is a Full Cone NAT, if host A sends a packet to host C resulting in the translated IP and port (IPA,PortA), then host D could use (IPA,PortA) to communicate with host A without host A ever directly sending a packet to D. In the other case,

Host 1	**Host 2**			
	Public	**Full Cone NAT**	**Symmetric NAT**	**Firewall**
Public	TCP	TCP	TCP	TCP
Full Cone NAT	TCP	UDP	UDP	UDP
Symmetric NAT	TCP	UDP	X	?
Firewall	TCP	UDP	?	?

Table. 1. Connectivity Matrix. TCP denotes communication is possible through both UDP and TCP, UDP denotes communication is possible through UDP only, X denotes communication is not possible and ? denotes depending on the properties of the firewall any of the three cases are possible.

where the NAT is a Symmetric NAT host D would not be able to use (IPA,PortA) to communicate with A; in fact D would have to receive a new (IP, port) touple directly from A before communicating with it.

Table 1 characterizes the pair-wise connectivity restrictions. For example, communication is not possible between two hosts behind different NATs using TCP, however certain combinations of NATs can communicate with UDP. Specifically, hosts behind different *Full Cone NATs* can communicate using UDP, but host behind different *Symmetric NATs* cannot. Firewalls are much more diverse in their configurations so it is difficult of characterize their connectivity constraints, however we assume the following common properties of firewalls: (i) most firewalls only allow TCP connections to be initiated from behind the firewall and (ii) UDP communication may follow *Full Cone* or *Symmetric* properties exhibited by NATs or may block UDP traffic in either direction. We currently do not consider firewalls that block UDP traffic in either direction. However, this is an important case that we will be considering in the future.

2.2 Existing Solutions for NAT and Firewall

Several general network level solutions to connect NAT and firewall hosts have been proposed [1,14,19]. Of these solutions, Universal Plug and Play (UPnP) [1] is the only one being aggressively deployed by a number of companies. UPnP NATs allow an application to add or remove translations in the NAT and learn the public address and port of a translation. The application dvertises this information allowing external hosts to initiate communication to it. In general, UPnP NATs behave like Full Cone NATs once the translation is created. Even with the current UPnP deployment, there are still reasons why all pairs of NAT and firewall hosts will not be connected:

- The incremental deployment of UPnP can take several years and may never reach the entire population of deployed NATs. A UPnP NAT

allows a single user within a private network to open a port to the rest of the Internet. Corporations, which use mostly Symmetric NAT will likely view this as a security hole and not switch to one that uses UPnP.

- Firewalls will spread with increased security risk. Firewall administrators want useful applications to work through their firewalls without sacrificing security. It is unlikely that they will allow applications that require universal connectivity to all Internet hosts. A more reasonable model is to allow only outgoing connections, which is currently the most common model. The result is hosts behind firewalls will be able to communicate with public hosts but not hosts behind different firewalls.

Clearly, the Internet will not be able to provide universal connectivity semantics in the foreseeable future. We approach this problem from the application level, with no changes to network elements and leveraging existing application level solutions for NAT and firewall. Simple Traversal of UDP through NAT(STUN) [10] is one such protocol that allows a host behind a NAT to determine its public address and port and the type of NAT. Our work is complementary to STUN, where we develop mechanisms to use and distribute the information learned through STUN's mechanisms.

2.3 Overlay Multicast

In overlay multicast, end hosts form a data distribution structure, usually a tree, and forward the multicast data over this structure. The protocol we use is distributed, self-organizing and performance-aware. The protocol builds and maintains a tree, possibly rooted at the broadcast source in a video broadcasting application. The tree is optimized primarily for bandwidth, and secondarily for delay. To maintain good performance, hosts monitor their receiving bandwidth and switch parents if they observe a performance drop. When selecting parents, the child looks for one that can provide better performance. Parents maintain a degree bound of the number of children to accept. A group management protocol is used to learn about other members that could be used as parents. We present details of the protocol in [6].

3. ACCOMMODATING NAT/FIREWALL IN OVERLAY MULTICAST

In this section we describe three solutions to accommodate NAT and firewall hosts: Strawman, Basic Contributor and Enhanced Contributor. We also describe a mechanism to optimize the structure of the overlay tree for NAT and firewall hosts called Connectivity-Aware Structuring. Implementation details are presented illustrating the required complexity. The rest of this paper uses the terms, *restricted* and *public* hosts. A restricted host can only communicate with public hosts, while a public host can communicate with either a restricted or a public host. Table 2 shows the types of hosts that belong to these groups depending on the transport protocol. Unless specified, we assume TCP.

	Restricted	Public
TCP Protocol	Full Cone NAT Symmetric NAT Firewall	Public
UDP Protocol	Symmetric NAT Firewall *	Public Full Cone NAT Firewall*

Table. 2. Definition of restricted and public hosts with UDP and TCP protocols. *Depending on the type of firewall it can be restricted or public.

3.1 Solutions Details

The **Strawman** solution makes restricted hosts leaf nodes only, thereby not utilizing available upstream bandwidth at these hosts. To implement this design the following requirements must be met: (i) each host must have a unique and persistent identifier throughout the session to allow members to differentiate between other known members, (ii) a valid mapping must be created between the identifier and the IP address and port and (iii) the identifiers and mappings of public hosts must be distributed to members in the group. Note that members in the group will not need to know or initiate any communication with restricted hosts for the purpose of parent selection since restricted hosts can only be leaf nodes and not contributors (or members who can be interior nodes in the tree).

IP addresses are no longer unique and the use of public IP and port or public and private IP addresses as a unique identifier does not serve this purpose because they are not persistent when considering Symmetric NATs. To resolve this, we assign a unique four byte flat overlay identifier(OID) to each host and decouple it from its IP address, separating overlay naming from addressing. When a host A joins the group, it is assigned an OID by the source. The source creates a binding that maps the OID of A to its public and private addresses and ports. This binding is distributed as part of the group membership management protocol.

There are two ways for a host B to learn bindings for host A. First, it can learn the binding as part of the group

membership operations. Second, it may receive packets directly from A. Bindings learned by the second method are prioritized because they are the only ones that can be used to talk to a host behind a Symmetric NAT. Each host B maintains the OID and associated binding for every other member A that it knows.

The OID is translated into the appropriate binding when B wishes to send a packet to A. In some cases A and B may be behind the same private network, but have different public IP addresses. This is common in the case of large corporations that use multiple NAT gateways. We use a simple heuristic where the public IP address prefixes are compared. Equal prefixes indicate the hosts are in the same private network. False positives are detected and removed if B does not receive packets from A after a short while.

To receive data the child always initiates a TCP connection for data transfer since the parent is guaranteed to be a public host.

The **Basic Contributor** solution allows restricted hosts to be contributors if they have upstream bandwidth available. However, they can only be contributors to public hosts. This solution directly increases the number of contributors for public hosts, and indirectly increases the number of contributors for restricted hosts since public hosts can choose restricted hosts as parents.

Implementing this solution requires all the mechanisms described for Strawman and a mechanism that allows public hosts to initiate communication with restricted hosts. This problem manifests itself in two issues: (i) a host A cannot communicate with Symmetric NAT host S, until S initiates communication with A and (ii) TCP connections cannot be initiated from public hosts to restricted hosts. Two mechanisms can be used to address the first issue: (i) hosts behind Symmetric NATs can periodically send messages to other members in the group allowing those members to communicate with it and (ii) each host S behind a Symmetric NAT can be associated with a public rendezvous host, which always maintains communication with S and helps any other host to initiate communication to S. The overlay multicast protocol we use performs group membership management using a gossip protocol. Periodically hosts tell a random member about a subset of hosts it knows about. In addition, the protocol periodically probes a subset of known members to search for a parent or a replacement parent if a better one than its current parent can be found. These two mechanisms effectively implement the first solution to address the issue for hosts behind Symmetric NAT. In the absence of such protocols, a rendezvous mechanism will be needed.

Since restricted hosts can be parents as well as children, TCP connections cannot always be initiated by the children as done in the Strawman solution. We use bi-directional connection initiation, by which both parent and child attempt to open a connection to the other. If one is a public and the other is NAT/firewall, then only one of the connections will be successful. If both are public, then both connections will be successful and we arbitrarily close the connection initiated by the host with higher IP address.

The **Enhanced Contributor** solution directly increases the number of contributors for restricted hosts. This is accomplished by using a UDP based transport protocol for all communication instead of TCP. The key point with this solution is that this allows direct communication between Full Cone NAT hosts and between Full Cone and Symmetric NAT hosts, as shown in Table 1. Therefore this solution moves Full Cone NAT hosts from the restricted host set to the public host set as illustrated in Table 2.
Implementing this solution requires the use of UDP for all communication. Our implementation uses a single port at each host over which it multiplexes all control traffic and data streams. We chose this port multiplexing mechanism to simplify management of external ports that the NAT gateway knows about. The alternative solution is to explicitly open and maintain ports in the NAT gateway complicating the overall system.

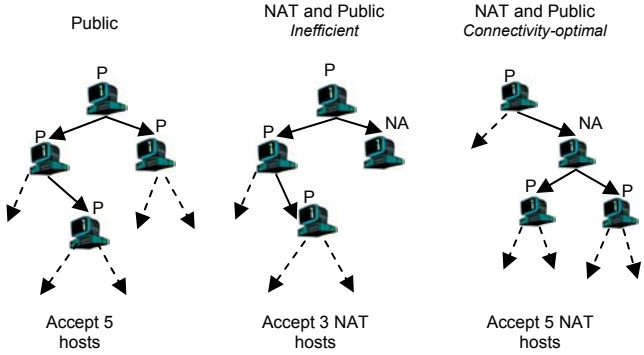

Fig. 2. Connectivity-Optimal Structuring.

2.1.1. Connectivity-Aware Structuring

Connectivity-Aware Structuring increases the number of available contributors for restricted hosts by optimizing the structure of the tree and can improve upon Basic and Enhanced Contributor. Figure 2 illustrates this concept. Figure 2 (a) shows a simple multicast tree with only public hosts where each host has degree 2. Assuming no incoming hosts can support children, this tree can support 5 more hosts. Now consider figure 2 (b), where one of the hosts is a restricted host. This tree can only accept three more restricted hosts, however if we restructure the tree as shown in 2 (c), we can support 5 more restricted hosts. We call the tree shown in 2 (c) a connectivity-optimal structure

for the set of hosts present. Notice, there may be many connectivity-optimal structures for a set of hosts. We use two heuristics to implement this in the overlay protocol: (i) **Passive Structuring**, where public hosts preferentially choose restricted hosts as parents when they make a parent switch and (ii) **Active Structuring**, where public hosts make a parent switch from a public host to a restricted host for the sole purpose of increasing the resources. While the active structuring heuristic should result in a more optimal structuring than passive, it induces more dynamics that will negatively impact performance.

4. IMPLEMENTATION

We have implemented the Basic Contributor design in the system described in [6]. NAT/firewall detection and OID creation and mapping are performed in the *Connectivity Service* shown in Figure 3. The figure shows the three layers used in the system. The overlay layer contains the overlay protocol and maintains overlay addressing. The transport layer at the bottom runs either TCP or UDP and uses IP addressing. In between is the connectivity layer, which learns mappings and translates between overlay addressing and IP addressing.

4.1 NAT/Firewall Detection

When a host joins the multicast group, it sends a join message to the source. The join message includes the IP address and port the host thinks it has. The connectivity service at the source compares the IP address and port inserted by the host with the IP address and port in the IP header. If they are different, then the host is behind a NAT. It relays this information back to the host in the join response message and is relayed in each membership message. The source and joining host performance firewall detection by initiating a TCP connection from the source to the joining host. The joining host checks for the existence of a TCP connection 1.5 seconds after sending the join message. If no TCP connection was established from the source, then the joining host is behind a firewall.

4.2 Translating OID to IP

On the send path, the overlay layer will call the SendMessage() function at the connectivity service with an destination OID argument. The connectivity service will translate the OID to an IP and port and send this to the transport protocol.

As described in Section 3, there are two ways to learn mappings from OID to IP and port. Both are illustrated in Figure 3. The first method is through group membership distribution. This is handled through the UpdateMapping() function. When the overlay protocol receives a group membership message containing an OID to IP and port mapping, it will use the UpdateMapping() interface to send this information to the connectivity service. All OID to IP mappings are maintained in the connectivity service. Therefore, to send a group membership message, the overlay protocol must lookup the mapping in the connectivity service. The LookupMapping() function is used to retrieve the mapping from an OID to IP and port from the overlay protocol when sending a group membership message. Mappings are also learned when packets are received from the network. When ReceivePacket() is called by the transport protocol, the connectivity service will first update its mapping for the sender of the packet by calling UpdateMapping(), and then send the packet up to the overlay protocol.

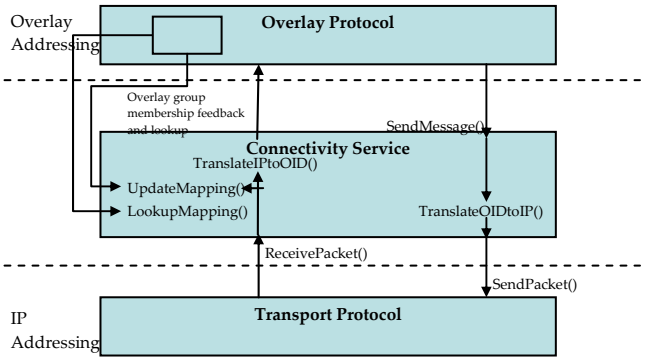

Fig. 3. Connectivity Service.

5. EVALUATION METHODOLOGY

Our evaluation consists of real Internet deployment and Internet experiments using the Emulab [21] testbed. This section describes the performance metrics and the experiment methodology.

5.1 Metrics

NATs and firewalls impose connectivity restrictions that can affect the proper functionality and performance of an overlay multicast system in several possible ways: (i) the system can get saturated for NAT and firewall hosts where new NAT and firewall hosts cannot join even though new public hosts can, (ii) NAT and firewall hosts have a reduced set of parent choices, specifically they cannot pick certain other NAT/firewall hosts and (iii) public hosts have a slightly reduced set of parent choices because they can choose a Symmetric NAT host as a parent only if the Symmetric NAT host has previously sent a message to them.

Saturation of the system may lead to several hosts that never connect to the tree and other hosts that intermittently connect to the tree. A reduced set of parent choices may lead to a longer time spent in searching for a parent. To capture these issues we use the *Rejection Ratio*, or the fraction of hosts that were never able to connect to the tree and the *Disconnect Ratio*, or the fraction of a host's stay

time spent disconnected from the tree. Stay time of a host is defined as the duration of time a host is in the group and is either connected to the tree or is attempting to connect to the tree.

In addition to these metrics we use the *Resource Index*. This is a general metric, independent of the overlay protocol that captures the resource capacity of the system considering bandwidth and NAT/firewall connectivity restrictions. *Resource Index* is defined as the ratio of the supply of bandwidth to the demand for bandwidth for a particular source rate. Supply is the sum of the degrees over all hosts participating and demand is the number of hosts participating. For example, consider Figure 2 (a), where each host has degree 2. The supply is 8 and the demand is 3 resulting in a Resource Index of 8/3. Considering restricted hosts however makes the Resource Index more complicated. In particular, the Resource Index becomes sensitive to the structure of the overlay, for the same set of hosts. For example, in Figure 2(b), the supply for newly joining restricted hosts is 3, and since there are three hosts receiving data already, the Resource Index is 6/3. In contrast, Figure 2(c), which has the same set of hosts, we find the Resource Index is 8/3. We observe that the optimal structure in terms of accommodating restricted hosts is one where public hosts preferentially choose restricted hosts as parents. Based on this observation, the *optimal* Resource Index for a set of hosts involving restricted hosts is defined as $\frac{S(t)}{|P(t)+R(t)}$, where

$$S(t) = \sum_{i \in P(t)} S_i + Min\left(\sum_{j \in R(t)} S_j \mid P(t) \mid \right)$$

where, *P(t)* is the set of public hosts at time *t* and *R(t)* is the set of restricted hosts at time *t*, and S_i is the number of children host *i* can support. The following is an informal derivation of this formula:

Since we can always use resources at public hosts, we have at least $\sum_{i \in P(t)} S_i$ resources. The optimal usage of restricted host resources is when all existing public hosts are children of restricted hosts. Therefore, if |*P(t)*| is greater than $\sum_{j \in R(t)} S_j$, then we can use all restricted host resources, however if |*P(t)*| is less than $\sum_{j \in R(t)} S_j$, we can use only |*P(t)*| resources from the restricted hosts. Figure 2(c) is an optimal structure for the set of hosts, and it can be verified that the formula confirms to the result stated above.

5.2 Experiment Methodology

Evaluation of the solutions for NAT and firewall consists of analysis of real world application deployment of overlay multicast and experiments using Emulab [21].

Event	Type	Incarnations	Entities	Peak
1	Lecture	75	38	37
2	Lecture	138	104	84
3	Conference	162	113	75
4	Conference	324	131	40
5	Conference	721	500	101
6	Conference	272	117	52
7	Conference	148	88	56
8	Lecture	94	66	50
9	Slashdot	2178	1316	160

Table. 3. Subset of broadcast events with *type*, *incarnations*, entities, and peak group size. Entities are the number of unique hosts that join the broadcast, where as the incarnations are the total number of joins. For example, one entity may join 3 times resulting in 3 incarnations.

4.2.1 Internet Deployment

To gain insight and real world operational experience with overlay multicast we have deployed a video broadcasting system and have conducted several broadcasts of actual content to real audiences on the Internet [6]. These broadcasts include technical conferences and lectures, small-scale sporting events and one broadcast to readers of the Slashdot [17] Web-based discussion form. The Slashdot broadcast is our largest to date and attracted many home and commercial viewers. In addition, it has the largest percent of NAT and firewall hosts. Table 3 characterizes the type and size of a subset of events labeled 1 through 9.

When considering supporting NAT and firewall hosts in the system we were met with several tradeoffs. We hypothesized early on the need to utilize the upstream bandwidth at NAT and firewall hosts. Not having any evidence of the performance improvement with Enhanced Contributor or Connectivity-Aware Structuring and the well incorporated use of TCP's congestion control for data distribution on the overlay, we were reluctant to change to UDP for the Enhanced Contributor solution and thus incorporated the Basic Contributor solution.

While we have not deployed Enhanced Contributor or Connectivity-Aware Structuring, we can still use static analysis with the Resource Index equation to evaluate other

design alternatives using traces from the broadcasts. In the Strawman solution, the $\sum_{j \in R(t)} S_j$ is always 0 since restricted hosts cannot be contributors. With Basic Contributor, this sum increases to the number of children that the restricted hosts can support. Finally, by using the Enhanced Contributor solution, the set *P(t)* of public hosts is increased based on the number of Full Cone NATs in the system and the set *R(t)* of restricted hosts is reduced appropriately. Note that the Resource Index computation assumes Connectivity-Aware Structuring with an optimally structured tree. Therefore, this result is an upper bound on the achievable Resource Index. To evaluate using rejection ratio and disconnect ratio we conduct Internet experiments.

4.2.2 Internet Experiments

Our Internet experiments are used to evaluate the Strawman, Basic Contributor and Enhanced Contributor solutions and Connectivity-Aware Structuring in a real protocol using the *Rejection Ratio* and *Disconnect Ratio* metrics. These experiments are run over the Emulab [21] testbed and do not consider network congestion or latencies. Therefore, the factors affecting the protocol are group dynamics, bandwidth resources and NAT/firewall connectivity restrictions. The experiments are setup as follows: Up to 100 simultaneous overlay hosts are multiplexed over 40 physical Emulab nodes. Each host is given the properties of a host from a 1 hour trace taken from the Slashdot event, where the group size ranges from 40 to 100 with a total of 553 individual joins. These properties include the join and leave times, available upstream bandwidth and NAT/firewall connectivity restrictions. The Bandwidth restriction is imposed by limiting the number of children each host can accommodate to the measured value in the trace and the connectivity restriction is imposed by an emulator within the protocol that restricts communication between two hosts that are assigned to be behind a NAT or firewall. We do not emulate the additional connectivity restrictions imposed by Symmetric NAT, where a host A can only send a message to a host S behind a Symmetric NAT, if S recently sent a message to A.

6. EVALUATION RESULTS

In this section we present results from the 9 events described in Table [3] in terms of the environment factors that affect system performance and performance results from experiments comparing the three solutions for supporting hosts behind NAT or firewall.

6.1 Internet Deployment

We summarize the following environment factors: (i) percent of NAT/firewall hosts and percent of Symmetric NAT hosts (ii) percent of public contributors and NAT/firewall contributors and (iii) Resource Index. Contributor hosts are hosts that can support children where we cap the number of children at 6.

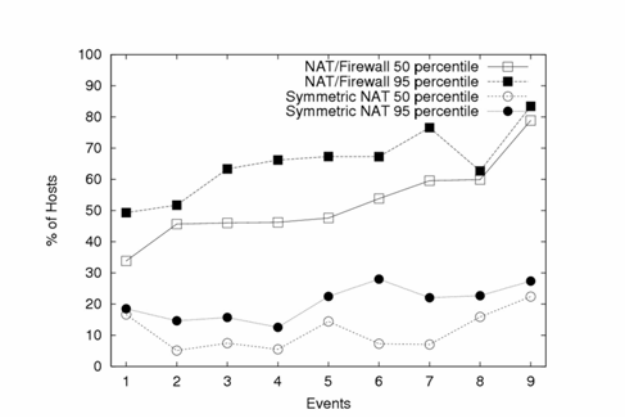

Figure. 4. 50 and 95 percentile of NAT and Symmetric NAT hosts over time.

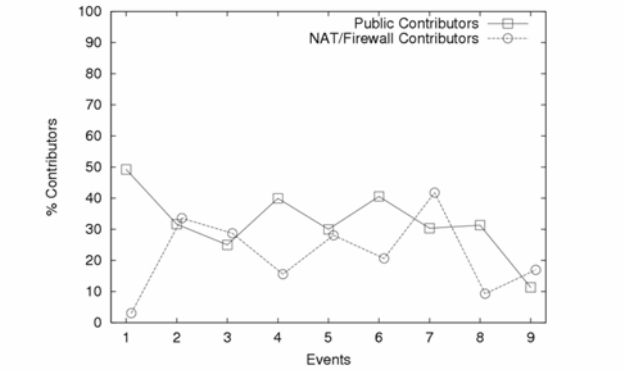

Figure. 5. 50 percentile contributor hosts for each event over time.

The percent of NAT/firewall hosts varies throughout the duration of an event. To capture this we compute the percent of NAT/firewall hosts every 5 minutes and extract the 50 and 95 percentile values. Figure 4 shows the 50 and 95 percentile NAT/firewall hosts over time for the nine different events. The 50 percentile NAT and firewall hosts ranges from about 35% to 80% while the 95 percentile ranges from about 50% to 80%. Figure 4 also shows the same information for Symmetric NAT, where the percent of Symmetric NAT drops to between 5% and 25%. The top two curves represent the percent of restricted hosts if the Basic Contributor Solution is used while the bottom two curves represent the optimistic estimate of the percent of restricted hosts if the Enhanced Contributor solution is

used. This is optimistic because we assume firewalls that are non-NATed can be connected just as Full Cone NATs, however this may not be true in general. Figure 5 shows the 50 percentile public and NAT/firewall contributor hosts for all events. Except for event 1, which had the lowest percent of NAT/firewall hosts, NAT/firewall hosts could contribute a large fraction of bandwidth resources and in our events, including Slashdot, they could contribute more than public hosts.

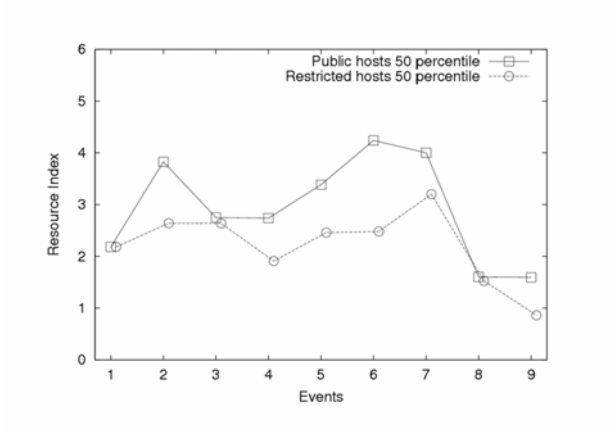

Figure. 6. 50 percentile resource index.

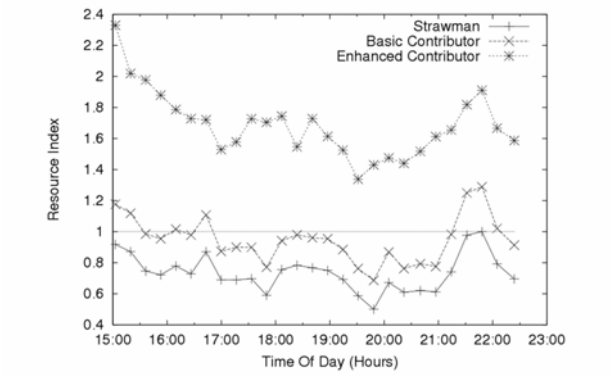

Figure. 7. Resource index for the Slashdot broadcast over time comparing the Strawman, Basic Contributor and Enhanced Contributor solutions with Connectivity-Aware Structuring.

Figure 6 shows the 50 percentile *Resource Index* over time for public hosts and restricted hosts using the Basic Contributor solution for the same 9 events. For example, in the Slashdot event, Resource Index for public hosts is about 1.6 indicating the bandwidth supply for public hosts is 1.6 times the demand, however the Resource Index for restricted hosts is about .9 indicating the supply for restricted hosts only accommodates .9 of the demand. Figure 7 shows the *Resource Index* for restricted hosts only for the Slashdot broadcast over time comparing the Strawman, Basic Contributor and Enhanced Contributor solutions assuming an optimally structured tree. The Strawman solution is always below 1 meaning a connected tree is not feasible. The Basic Contributor solution over long time periods is also below 1, however the Enhanced Contributor solution is significantly above 1. On average, Basic Contributor has a 29% improvement over Strawman and Enhanced Contributor has a further 75% improvement over Basic Contributor. More significantly, Enhanced Contributor is the only solution that has a Resource Index greater than 1 throughout the duration of the Slashdot broadcast.

To summarize our results, we make the following observations: (i) the percent of hosts behind NAT or firewall can reach up to 80% in commercial Internet environments (ii) hosts behind NAT or firewall contribute a significant amount, more than 50% in some cases, of bandwidth resources, (iii) close to 50% of restricted hosts are behind Full Cone NAT and (iv) the Resource Index for restricted hosts is much more likely to drop below an infeasibility level if their bandwidth is not utilized. These results indicate that both Basic Contributor and Enhanced Contributor have good potential to increase the Resource Index and improve host performance by harnessing the resources from hosts behind NAT of firewall.

NAT/Firewall Solution	Structuring	Rejection Ratio	Disconnect Ratio Average / 95 Pct
Strawman	N/A	.43	.05/.32
Basic Contributor	None	.08	.02/.12
	Passive	.05	.02/.10
	Active	.02	.01/.05
Enhanced Contributor	None	0	~0/.02
	Passive	0	~0/.02
	Active	0	~0/.02

Table 4. Rejection ratio and disconnect ratio for different NAT/firewall solutions.

6.2 Internet Experiments

Table 4 shows the rejection ratio for the different solutions. Each value is the average of three runs. Both Strawman and Basic contributor reject a large fraction of hosts where Strawman's rejection ratio is .43. The connectivity-aware structuring mechanisms help reduce the rejection ratio from .08 to .02 in the Basic Contributor solution. However, Enhanced Contributor does well without any Connectivity-Aware Structuring so there is no room for it to help. Table 4 also shows the average and 95 percentile disconnect ratio. The trend shown by this metric is similar to rejection rate, with Strawman having a large disconnect ratio and Basic Contributor having shorter but still significant disconnect

ratio. Enhanced Contributor sufferers very few disconnects at the tail. This tail is inherent to the parent search algorithm and available resources in the system.

7. CONCLUSION

Through several live Internet broadcasts of video content to a real audience, we have observed environments where the percent of hosts behind NAT or firewall ranged between 35% and 80%. In addition, 5 of the 9 events were in environments were hosts behind NAT or firewall were capable of contributing as much or more bandwidth than public hosts.

We present two solutions, Basic Contributor and Enhanced Contributor that significantly improve overlay multicast performance when compared with the Strawman solution, which does not utilize bandwidth resources at hosts behind NAT or firewall. In particular, Internet experiments show that in the Slashdot event, Basic Contributor eliminates most of the rejections suffered by Strawman and Enhanced Contributor performs very well without suffering any rejections. We have proposed Connectivity-Aware Structuring to increase the available resources for restricted hosts. This mechanism improves performance in the Slashdot environment when using the Basic Contributor solution, reducing the rejection ratio from .08 to .02 and having a similar impact on the disconnect ratio.

A key lesson from our results is the importance of the transport protocol for accommodating NAT and firewall hosts. In this paper we have quantified the performance improvement by using UDP to increase pairwise connectivity. However, using UDP alone is not a complete solution since there do exist hosts behind firewalls that can only communicate using TCP. Therefore, accommodating NAT and firewall hosts will require the overlay protocol to negotiate the appropriate protocol (UDP or TCP) between pairs of end systems.

REFERENCES

[1] Understanding Universal Plug and Play. Microsoft White Paper.

[2] Akamai. http://www.akamai.com/.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of (ACM) SIGCOMM*, Pittsburgh, PA, August 2002.

[4] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Content Distribution in Cooperative Environments. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, Bolton Landing, NY, October 2003.

[5] Y. Chawathe. Scattercast: An architecture for Internet broadcast distribution as an infrastructure service. Fall 2000. Ph.D. thesis.

[6] Y. Chu, A. Ganjam, T.S. Eugene Ng, S.G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early Experience with an Internet Broadcast System Based on Overlay Multiast. To Appear In *Proceedings of USENIX Annual Technical Conference*, Boston, MA, June 2004.

[7] Y. Chu, S.G. Rao, and H. Zhang. A Case for End System Multicast. In *Proceedings of (ACM) International Conference on Measurements and Modeling of Computer Systems (Sigmetrics)*, Santa Clara, CA, June 2000.

[8] J. Albrecht D. Kostic, A. Rodriguez and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, Bolton Landing, NY, October 2003.

[9] P. Francis. Yoid: Your Own Internet Distribution, http://www.aciri.org/yoid/. April 2000.

[10] J. Rosenberg, J. Weinberger, C. Huitema and R. Mahy. STUN - Simple Traversal of UDP Through Network Address Translators. RFC 3489, March 2003.

[11] J. Jannotti, D. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, San Diego, CA, October 2000.

[12] J. Liebeherr and M. Nahas. Application-layer Multicast with Delaunay Triangulations. In *IEEE Global Telecommunications Conference (Globecom)*, San Antonio, TX, November 2001.

[13] A.M. Kermarrec M. Castro, P. Druschel and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. In *IEEE Journal on Selected Areas in Communications Vol. 20 No. 8*, Oct 2002.

[14] T.S. Eugene Ng, I. Stoica, and H.Zhang. A Waypoint Service Approach to Connect Heterogeneous Internet Address Spaces. In *Proceedings of USENIX Annual Technical Conference*, Boston, MA, June 2001.

[15] V.N. Padmanabhan, H.J. Wang, and P.A Chou. Resilient Peer-to-peer Streaming. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, Atlanta, GA, November 2003.

[16] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level Multicast using Content-Addressable Networks. In *Proceedings of Third International Workshop on Networked Group Communication (NGC)*, London, UK, November 2001.

[17] Slashdot. http://slashdot.org/.

[18] S.Q.Zhuang, B.Y.Zhao, J.D.Kubiatowicz, and A.D.Joseph. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination, April 2001. In *Eleventh International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001)*, Port Jefferson, NY, June 2001.

[19] Z. Turanyi and A Valko. IPv4+4. In *Proceedings of 10thInternational Conference on Network Protocols (ICNP)*, November 2002.

[20] W. Wang, D. Helder, S. Jamin, and L. Zhang. Overlay optimizations for end-host multicast. In *Proceedings of the Fourth International Workshop on Networked Group Communication (NGC)*, October 2002.

[21] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI)*, pages 255–270, Boston, MA, December 2002.

Aditya Ganjam received the B.S. degree in computer science, the B.S. degree in electrical and computer engineering, and the M.S. degree in computer science, all from Carnegie Mellon University, Pittsburgh, PA, in 2002 and 2003. He is currently working toward the Ph.D. degree at
Carnegie Mellon University. His research interests include design and deployment of overlay multicast and management of fiber to the home access networks.

Hui Zhang received the B.S. degree in computer science from Beijing University, Beijing, China, the M.S. degree in computer

engineering from Rensselaer Polytechnic Institute, Troy, NY, and the Ph.D. degree in computer science from the University of California, Berkeley, in 1988, 1989, and 1993, respectively. He has been on the faculty of the School of Computer Science, Carnegie Mellon University, since 1995. His current research interests include exploring new approaches that can dramatically enhance the scalability, robustness, security, and manageability of broadband access networks, enterprise networks, and the Internet. Previously, he did research on Internet QoS, multicast, and peer-to-peer systems. Algorithms and software packages resulting from his research have been widely adopted by industry and academic institutions. He held the CMU SCS Finmeccanica Junior Faculty Chair from 1998 to 2002. He was the Chief Technical Officer of Turin Networks from 2000 to 2003. Dr. Zhang was the recipient of the National Science Foundation CAREER Award in 1996 and the Alfred Sloan Fellowship in 2000.