

Topology Discovery for Public IPv6 Networks

Daniel G. Waddington, Fangzhe Chang, Ramesh Viswanathan, and Bin Yao
Networking Research Laboratory
Bell Laboratories
101 Crawford's Corner Road, Holmdel, NJ 07733
{dwaddington, fangzhe, rv, byao}@dnrc.bell-labs.com

ABSTRACT

In just three decades the Internet has grown from a small experimental research network into a complex network of routers, switches, and hosts. Understanding the topology of such large scale networks is essential to the procurement of good architectural design decisions, particularly with respect to address allocation and distribution schemes.

A number of techniques for IPv4 network topology already exist. Of these ICMP-based probing has shown to be most useful in determining router-level topologies of public networks. However, many of these techniques cannot be readily applied to IPv6 because of a changes in the addressing scheme and ICMP behaviour, and also increased proliferation of equal-cost multi-path routing.

This paper presents Atlas, a system that facilitates the automated capture of IPv6 network topology information from a single probing host. It describe the Atlas infrastructure and its data collection processes and discusses identified IPv6 network phenomena that must to be taken into account by the probing scheme. We also present some initial results from probing the 6Bone, currently the largest public IPv6 network. The results illustrate the effectiveness of the probing algorithm and also identify various trends in prefix allocation and routing policy.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Network Topology;
C.2.3 [Network Operations]: Network Monitoring and Public Networks

General Terms

IPv6 Network Topology Discovery

Keywords

Network probing, topology inference, IPv6, network measurement

1. INTRODUCTION

A fundamental identifying characteristic of any network is its connectivity topology, typically modeled as a graph with nodes

representing devices (e.g., routers, switches, and end hosts) and edges representing links (physical or logical). Additional information about the network, such as the delay or loss, can be represented as annotations on the graph [1]. However, many large networks do not have a single administrative body and their behavior continuously changes—their complexity and dynamic nature make the manual collection of topology information infeasible.

Many tools have already been developed to collect IPv4 network topology information [2, 3, 4, 5]. Of these, approaches that rely upon ICMP-based (Internet Control Message Protocol) probing have proven most useful in collecting topology information from public networks, where access to routing resources is restricted. However, applying existing IPv4 probing techniques to IPv6 networks is not straightforward; IPv6 has a wider address space, a different scope-based addressing scheme and modified protocol fields. For example, the Rocketfuel [6] probing tool uses the IPv4 header identifier field to help identify addresses belonging to the same router. The IPv6 header does not include such a field (since fragmentation is not performed) and therefore this technique simply cannot be used. Another probing technique used in many IPv4 discovery tools is address guessing. The Mercator system [4] uses ‘informed random address probing’ to guess which portions of the entire IP address space contain addressable nodes. This technique cannot be applied to IPv6 networks since the address space is significantly larger. Consequently, assigned prefixes are numerically sparse and therefore guessing adjacent prefixes is not effective.

Topology discovery in IPv6 networks also merits additional interest. IPv6 is the predicted successor to the current Internet Protocol, IPv4. Today, the proliferation of IPv6 is relatively small and therefore provides an opportunity to more fully understand the evolution of an inter-network from its early stages. By capturing and understanding the topologies of deployed large scale IPv6 networks over time, we can better prevent undesirable trends and realize good architectural decisions. However, the deployment of IPv6 demands a number of new and untested technologies. For example, the majority of IPv6 networks are not built from scratch, but rather as an overlay network supported by existing IPv4 infrastructure. Consequently, required transitioning technologies, such as tunnels and translators, significantly impact network topology (at least from the point of IP). It is these such aspects that make the collection of topology information from IPv6 networks an interesting and challenging problem.

Probe deployment generally take one of two approaches. The first is to deploy multiple probe engines throughout the network [2]. Paths are then probed from the individual points and the data later unified into a single topology graph. This approach does rely upon the ability to deploy probe engines throughout the network, which is many cases is not possible.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2002 ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

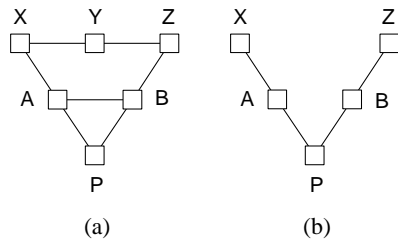


Figure 1: Increased coverage from source-routed traceroutes.

Alternatively, a single probe engine can be used in conjunction with *source routing* facilities. Both IPv4 and IPv6 provide the ability to perform source routing, which allows the packet sender to specify one or more intermediate nodes that should be visited en route to the final destination. However, source routing in the IPv4 Internet is largely disabled (other work has shown that only 8% of routers are source-route capable [4]). Conversely, source routing support in IPv6 routers is a mandated feature and hence mostly enabled. With respect to traceroute, source routing allows one to discover paths between arbitrary pairs of network nodes, neither of which need be the originating node. To illustrate the increased coverage of the topology that source routing offers, consider the network of Fig. 1(a), where P is the probing node and, X and Z are nodes whose addresses are known beforehand. By using a simple traceroute (i.e. without source routing) to destinations X and Z, one obtains the paths P–A–X and P–B–Z, and discovers routers A and B. Probing for paths to the newly found nodes A and B does not result in the discovery of any additional nodes or links and therefore the topology given in Fig. 1(b) will result. On the other hand, with source-routed traceroutes one can probe from P for paths between A and B (revealing link A–B) and between X and Z (revealing an additional router Y together with links X–Y and Y–Z). Consequently, the use of source routed traceroute results in the additional discovery of nodes (Y) and links (A–B, X–Y, and Y–Z).

While source-routed IPv6 traceroute forms the underlying basis of our work, our work additionally addresses several challenges that we have identified arising from observed network phenomena, which affect the path information gathered by traceroute. One issue is that a router manifests itself in paths by the IP address of one of its interfaces. Each router is likely to have multiple physical interfaces. Therefore, one must identify different addresses that belong to the same physical router and therefore correspond to a single node in the topology. A second issue is the presence of router interfaces that are *anonymous*, in that they do not have configured network addresses and therefore cannot be uniquely identified. As a result, each occurrence of an anonymous interface across multiple paths must be treated as a distinct node in the topology graph. This means that a naive graph construction would include multiple distinct nodes for each anonymous interface, resulting in an inaccurate and inflated topology. A third issue is the existence of unstable routing, which can significantly compromise the accuracy of path information generated by traceroute. Without careful examination, routers not directly connected may appear as adjacent due to probe packets traversing different paths.

While the prevalence of these phenomena in an inter-network is perhaps not surprising, their consequent effect on traceroute results and the fidelity of the topology generated doesn’t seem to have been explicitly recognized before. Some of our techniques to account for these phenomena exploit special capabilities afforded by IPv6 but most of them can be easily adapted to an IPv4 setting as well. For some of the phenomena, the techniques we propose are the first

ones to our knowledge, and for others our techniques are better or different than previously proposed solutions. One of the significant differentiators of Atlas as a topology discovery tool is, therefore, its comprehensive accounting of network phenomena in ensuring the accuracy of its inferred topologies.

The rest of the paper is structured as follows. Section 2 describes the background for topology discovery in IPv6. Section 3 then introduces the Atlas probing and topology inference system. Section 4 discusses the network phenomena encountered by Atlas, together with solutions to handle them. Section 5 then presents preliminary results from simulation and probing the 6Bone. Finally, Section 6 reviews related work and Section 7 summarizes the paper and discusses future directions.

2. BACKGROUND

Internet Protocol version 6 (IPv6, IETF RFC 2373) is designed to resolve address space shortages and provide fundamental support for security and Quality-of-Service (QoS). The deployment of IPv6 is becoming more widespread. Among the many new features it provides, the most relevant to this paper are its addressing scheme and its source routing capability.

The protocol provides different types of addresses, including those for different casting (e.g., multicast, anycast, and unicast) and for different scopes (e.g., global, site-local, and link-local). Furthermore, not all router interfaces need be configured with a unique global address, such as those connected by point-to-point links.

IPv6 packets consist of a basic header and a number of optional extension headers. One such extension header is the *routing header* (RFC 2460), which supports the source routing feature as mentioned earlier. The basic header contains the address of the next specified router in the path, while the routing header contains a list of all other routers including the final destination. Each intermediate router (herein termed *via-router*) swaps the next address in the list with the current destination address in the basic header. This process is performed according to an embedded counter (known as the *Segments Left* field) that is decremented at each swap. After a swap has occurred, the packet is then forwarded out to its next destination (providing the next destination is not local).

With respect to source-routed probing of path P to Z via X in Fig. 1(a), packets on the segment P–A–X include X in the basic header and Z in the routing header. Those on the segment X–Y–Z include Z in the basic header and X in the routing header (even though X no longer serves any purpose). Router X is responsible for swapping the address in the basic header with the appropriate address in the extension header. We refer to the segment from the probe engine to the via-router (i.e., P–A–X) as the first leg and the segment from the via-router to the destination (i.e., X–Y–Z) as the second leg.

An integral part of both IPv4 and IPv6 is ICMP. ICMPv4 and ICMPv6 are in many ways similar, but are nevertheless different protocol standards. ICMP defines basic control messages that are used by routers and hosts to signal errors in packet handling. Two specific message types relevant to traceroute are “hop limit exceeded in transit” and “port unreachable”. In ICMPv6 (RFC 2463) the former is sent when a router detects a packet that has a hop limit (as part of the basic header) that is no bigger than one. The latter is sent when either a host or router’s TCP/UDP protocol layer receives a TCP/UDP packet with an invalid port number (invalid in that there is no listener present). The traceroute algorithm works by repeatedly sending UDP probe packets to a given destination with a progressively increasing hop limit. As the probe packets fall short of the final destination, in terms of insufficient hop limit, “hop limit exceeded in transit” messages are returned by the intermedi-

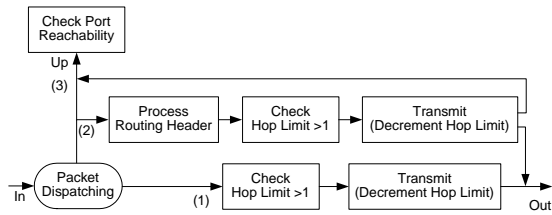


Figure 2: Packet processing model. “Check” boxes potentially generate ICMPv6 responses when the router (1) forwards a packet, (2) processes the routing header, and (3) passes the packet up the stack.

ate routers. When a traceroute packet reaches its final destination a “port unreachable” message is returned by the host (ensured by using a reserved UDP port number that should not have any active listener). The responding ICMPv6 messages contain as much of the invoking packet as will fit without exceeding the minimum IPv6 MTU (defined as 1280 bytes). This means that the headers of the probe packets will usually be included in the responses.

The source address used by ICMPv6 responses is defined as either (1) the unicast address to which the traceroute packet is destined, providing it belongs to the responding node, or (2) an address belonging to the responding node that will be most helpful in diagnosing the error. However, it is not clear which address (belonging to the ingress or egress interfaces) is in fact more helpful to ICMPv6 messages such as “hop limit exceeded in transit”. This policy is different from that of ICMPv4, which specifies that the IP address belonging to the egress interface should be used, or the router identity number in its absence.

Fig. 2 shows a typical relationship between source routing and ICMPv6 message generation. There are two important points illustrated here. Firstly, a router will generally process the routing header before checking the hop limit. Secondly, after an address swap has occurred, if the new destination address belongs to the processing router, then the packet will be forwarded immediately to the upper layers. As a result, a single packet may in fact cause the router to generate both “hop limit exceeded in transit” and “port unreachable” messages together.

3. THE ATLAS SYSTEM

Atlas consists of four components: a probe engine, a topology constructor, a topology verifier, and an interactive visualization program (see Fig. 3). The probe engine collects raw path information by exhaustively using source-routed traceroutes between all known addresses. The main challenge of the probe engine is to correctly identify the network phenomena and to retrieve accurate path information.

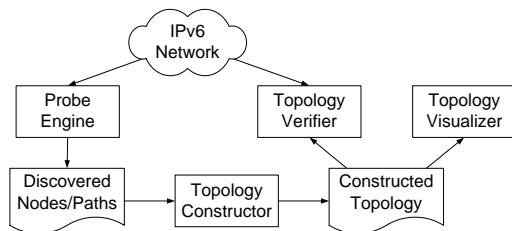


Figure 3: Overview of the Atlas system.

The topology constructor builds the connectivity topology graph

based upon the path information gathered by the probe engine. In cases where it is unable to restore the exact “actual” topology (see the discussion about anonymous nodes in Section 4), it attempts to construct a minimal graph that preserves the discovered node and path information. Constructing such a graph is provably NP-hard and cannot be approximated within certain bounds. Detailed discussion of the topology constructor can be found in [7].

The topology verifier looks at the constructed topology and re-asserts (through additional probing) the existence of the routers and links modeled by the graph. For each node, the verifier examines the reachability of the respective router and also its support for source routing. For each edge, it asserts the existence and direction by re-probing the path between the two nodes. It also tries to verify the path across anonymous nodes and connections within restricted routing areas (see Section 4).

Finally, the interactive visualization program extends the 3D hyperbolic space layout tool H3 [8]. This tool allows interactive navigation of topology graphs that are laid out according to the address prefix hierarchy. Enhancements have been made to include the ability to interactively locate elements, isolate and expand branches of the prefix tree and filter nodes from a given search. The tool has also been ported to a web-based environment.

3.1 Seed Selection

To initiate the discovery process, Atlas relies on probing paths among a set of addresses known *a priori* which we term “seeds”. It maintains a list of address pairs, each specifying a single via-router and destination for a source-routed traceroute session. The list initially contains only pairs of seeds and later grows as new router addresses are discovered. For efficiency, pairs contained in a discovered path do not need to be re-probed (they are pruned). This does assume that the pruned pairs will not reveal a path that is not already a segment of the discovered path. However, this assumption may not be true due to policy-based routing in which case some coverage may be sacrificed; Atlas can be configured to trade off between coverage and performance.

The choice of seeds directly affects the coverage one can hope to achieve with the Atlas probe engine. The search algorithm is inherently only able to discover routers that exist in the transit part of the network. By this, we mean routers that are traversed by packets en route to some other destination. Hence the optimal choice of seeds is all routers that exist at the edge of the network (i.e. non-transit). Fig. 4 gives some idea of the relationship between seed placement and achievable coverage.

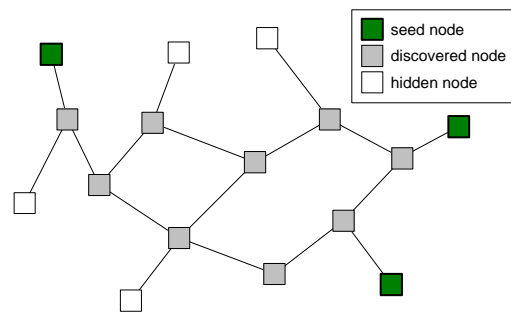


Figure 4: Seed placement and its effectiveness.

Currently, seeds are derived from the information in the 6Bone registry [9], which is a public database of sites and their respective address allocations. Atlas generates “likely” IPv6 addresses from the prefixes listed in this registry. Simple traceroute packets are

then sent to these addresses with a large hop limit value. Providing that the generated address does not belong to an actual router, then typically the router closest to the sub-network that has been allocated the associated prefix will generate an ICMPv6 message indicating that the address is not reachable. In this case, the ICMP message source address can be used as a candidate seed. However, if a generated address is actually used by an active router or host, then itself or an adjacent router address can be used.

Further tests must be performed before these candidate addresses can be used as seeds by the probe engine. Atlas first checks that they are source routing capable by using them as a via-router in a test trace. The system then checks that the router’s ICMP handling is conformant and ensures that they do not behave irregularly by comparing the behavior against several reported bugs. For example, some routers copy the remaining values of the hop limit field to the responding ICMP messages. Their responses would not be received without a much larger hop limit value than their real distances; a behaviour ultimately resulting with incorrect paths.

Of course, seeds should preferably be assigned to routers (as opposed to end-systems), since we are trying to build a router-level topology of the network. However, many of today’s end-systems are capable of performing IP routing and are in fact configured to do so. For example, Microsoft Windows 2000 machines installed with the IPv6 add-on, enable source routing by default and are auto-configured with a routing gateway. To avoid probing end-systems that behave as routers, we ensure that there is at least one route through the given device.

3.2 Increasing probe engine performance

Probing a large network is time-consuming; even performing a single traceroute can take tens of seconds. To enhance probing performance Atlas employs both caching and parallelism. For each trace, the probe engine caches the hop distance to the via-router. If the same via-router is used in subsequent traces, which will occur because of the exhaustive nature of the process, then the cached distance provides the initial hop limit (as opposed to simply starting from one) and thus alleviates the need to re-probe the first leg.

The probe engine also performs multiple traces concurrently. The number of traces that can be performed in parallel depends upon the probe timing (described later in Section 4.1) and also network bandwidth restrictions toward the probing node. Our initial experiments ran with 50 concurrent traces which gave significant improvement over the serialized version.

Finally, the Atlas probe engine uses checkpointing to enable fast recovery if failures occur. In the development phase, this also reduced the time to debug program code.

4. NETWORK PHENOMENA

During our experiments we observed a number of network phenomena that typically arise from different router implementations and configurations. The rest of this section reviews network phenomena identified by Atlas, including differing ICMP handling, anonymous interfaces, instable routing, address equivalence, and restricted routing areas. For each of the phenomena we discuss how the Atlas implementation addresses the problem.

4.1 Intermittent Router Non-Responsiveness

In order to limit the bandwidth and forwarding costs of ICMP, particularly to avoid ICMP Denial-of-Service (DoS) attacks, all IPv6 nodes must implement rate limiting (also termed *quenching*) on their outgoing ICMP messages. As a consequence, a router may appear to issue intermittent responses to the probing.

Quenching is generally timer-based (limiting the minimum interval between sending messages) or bandwidth-based (limiting messages to a fraction of the available bandwidth). The effect of rate limiting depends upon the specific router implementation. Some implementations may queue any pending ICMP messages, whilst others may simply drop them. In any case, the Atlas probe engine must avoid sending probes in a manner that might trigger ICMP quenching.

Currently we use two techniques to avert this phenomenon. The first is to use a timer back-off mechanism. That is, identical probe packets are repeatedly sent with a given hop limit, according to a time delta that exponentially increases until a given timeout period is reached. This scheme offers simple adaptive retransmission and avoids prolonged ICMP quenching. It is possible that the exponential back-off is not optimal. However, we are not overly concerned with the choice of back-off algorithm since gains in performance are not critical.

The second mechanism is dispersion of concurrent traces. The key to this approach is to avoid sending too many probes in relative locality within the search space. The Atlas probe engine attempts to disperse traces by randomly taking pairs from the list of pairs that still require probing. Nevertheless, this solution is limited in its effectiveness since congestion is still likely to arise where the traces converge close to the probing node. Should excessive congestion occur, ICMP messages will be dropped in the network and the resulting paths will be incomplete.

4.2 Prolonged Router Non-Responsiveness

In some cases routers will not respond to probes over a prolonged length of time. Our observations have shown that some routers in fact have ICMP disabled for security reasons or that they are non-responsive due to long-term congestion. Once a timeout has occurred, the probe engine attempts to *push through* the non-responsive router to the next router in the path; that is, probes are sent with an increased hop limit (see Fig. 5). Any routers that are non-responsive are treated as anonymous and labelled accordingly. However, this is only useful if the complete path for the trace is received; a path must begin and end with a non-anonymous router.

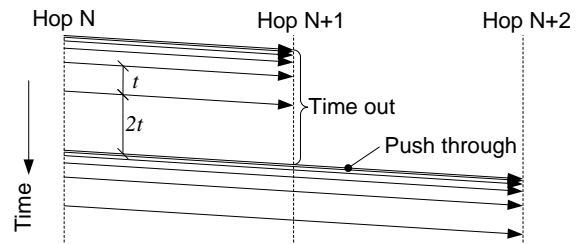


Figure 5: Exponential timer back-off between traces together with the push through process.

Attempting to push through a non-responsive router is important since it may only be a single router in the path that is non-responsive. If the push through succeeds (i.e. ICMP responses are received from router $N + x$, where router N was the last to respond) then the process continues for router $N + x + 1$ and so forth. The push through approach does, however, have its limitations. The problem exists in distinguishing between a router that is non-responsive permanently, or just intermittently (as previously described). If the maximum timeout period is not sufficiently large, an intermittent non-responsive router (which responds at a later point in time) may in fact be mistaken for a permanently non-

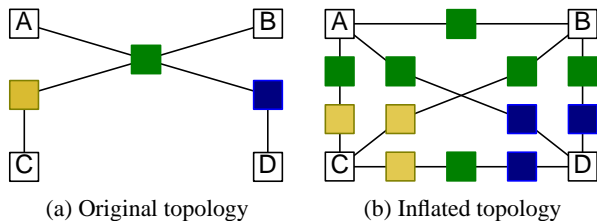


Figure 6: Inflationary effects on the topology graph at the presence of anonymous nodes.

responsive router. The resulting effect on the constructed topology is that some routers may be duplicated by a “phantom” anonymous router, which ultimately ensues as an inaccurate topology.

4.3 Anonymous Interfaces

Normal behavior of a router is to use one of its global unicast addresses as the source address of its ICMP responses. This address disclosure is the cornerstone behind using traceroute as a path discovery tool. However, not all interfaces may have a configured global unicast address; they may be configured with IPv6 site-local or link-local addresses only. Furthermore, such an interface may be selected to originate the ICMP packet’s source address (according to the rules described in Section 2). In this instance, our observations have shown that the majority of routers copy the destination address of the original probe packet into the source address of the responding ICMP packet. For example, if node Y in Fig. 1 is such a router, traceroute would report the second leg as X–Z–Z, as opposed to X–Y–Z. In our testbed, we have two routers that behave anonymously (Cisco 2600); they have a point-to-point link between them where the respective ports have not been assigned a global unicast address. The anonymous interfaces phenomenon has not been reported previously in IPv4 networks. Its existence arises from the availability of scoped addressing in IPv6. Furthermore, what the correct behaviour in this case actually is, is unclear from the ICMP and IPv6 standards.

If the Atlas system receives a “hop limited exceeded” message that appears to be from the destination router, but which is of course a response from an anonymous interface, the node is labeled with a unique name (e.g., ANON1). Hence in our example, path X–Z–Z would be altered to X–ANON1–Z.

Since anonymous routers can occur in multiple paths, each of which is treated as distinct node, a straightforward construction of the topology would include multiple nodes for each “real” anonymous node, resulting in an inflated topology. Fig. 6 illustrates how probing the topology in Fig. 6(a) would yield the reconstructed graph in Fig. 6(b). The shading of the nodes illustrates the mapping between the real node and its representation in the probed paths. Since the topologies in both figures would result in the same traces, they are inherently indistinguishable. A key challenge in topology construction is to minimize the inflationary effect by merging anonymous nodes into a minimal graph. The merging heuristic is essentially reliant upon like neighbors of the anonymous nodes.

It is easy to see that more than one topology may generate the same set of probe traces in the presence of anonymous interfaces. Our heuristic, in the topology construction phase, aims at finding a minimum admissible topology among the candidates by merging anonymous interfaces. The heuristic observes trace preservation and shortest path preservation. The former requires the resultant topology to be able to generate all the paths in the trace, and possibly more (since the original probing might not be complete). The

latter requires that no shorter distances are created between any two non-anonymous addresses due to the merging of anonymous interfaces. In addition, two anonymous interfaces are not mergeable if they are generated in a same probe path since the path resulted from a same traceroute session should be simple and contain no loop. We have proven that finding a minimum topology under these constraints is an intractable problem and have shown the heuristic works well in realistic cases where the percentile of anonymous interfaces is no more than 5%. A more detailed discussion about merging is presented in [7].

4.4 Instable Routing

One of the fundamental assumptions that traceroute-based probing techniques make is that packets sent between any two nodes in a given direction will always take the same route; that is, the path is stable. However, this assumption may not hold if there are multiple alternate paths between pairs of nodes. From our own experience with the 6Bone, and IPv4-based experiments performed by others [10], routing is stable in most of the cases. However, some instances of instable routing (sometimes termed *fluttering* [10]) have been detected where each packet takes one of multiple possible paths. Alternate paths exist typically for traffic engineering or load balancing purposes. The more common routing protocols (e.g., OSPF and IS-IS) support the notion of equal weighted multiple paths for a given destination. Multi-path routing policies are not standardized across implementations. Example policies include the use of hashing algorithms on source and destination addresses, round-robin or some load-relative algorithm (RFC 2328).

Multi-path routing fundamentally affects the path information collected by traceroute. Fig. 7(a) illustrates a simple example where router A makes dynamic routing decisions to destination D, following either path A–C–D or A–B–C–D. A trace between P and D, via A, could result in ICMP responses from nodes A, B, and D (for an increasing hop limit of one to three). The path thus appears to be A–B–D which does not exist. Atlas will eventually discover the links A–C, B–C, and C–D because the search is exhaustive. Nonetheless, an erroneous link B–D would still be introduced.

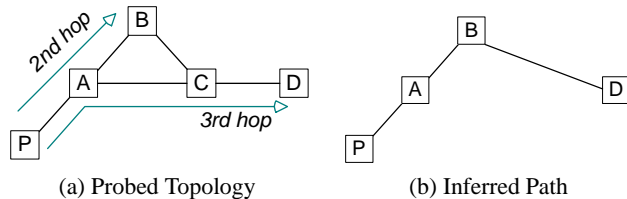


Figure 7: Traceroute packets (from P to D via A) may follow different paths for the second and the third hop as in (a), and produce an erroneous path A–B–D as in (b).

Our current solution to this problem is relatively straightforward. Multiple probes are dispatched for each hop in *batches*. For a hop result to be considered stable (and hence accurate) Atlas must receive all replies for a given batch from the same router. It is however not necessary to wait for a response to all packets in a batch, since ICMP quenching and congestion may be taking effect. If responses are received from different routers then instable routing is evident and the path is invalidated. Although the probe engine tries to avoid the erroneous effects of instable routing, the technique used cannot assure that all such cases will be detected; it can only make probabilistically guarantees.

A more complete solution might be to eliminate the side effects of routing instability with additional source routing indirect-

tion. That is paths are verified by forcing a traceroute through all members of the known path. This approach would all the identification of points of instability and furthermore allow the stable sub-paths to still be admitted to the topology data.

4.5 Address Equivalence

As discussed previously, routers follow certain rules in selecting source addresses for their ICMP messages. Because routers typically have multiple global unicast addresses, ICMP messages that are generated by a single router may not always use the same source address. As a consequence, two different traces across the same router may reveal different addresses and thus appear as two unrelated entities. To reflect the original topology, it is important to find a way to determine address equivalence (i.e. those belonging to the same router).

Unifying equivalent addresses is also discussed in Mercator [4]. Their solution sends a probe to one of the potentially equivalent addresses and expects to receive a “port unreachable” response from a different address. As described in their paper, this approach relies on the assumption that all ICMP responses use the address of the interface closest to the probe engine. For IPv6 routers this assumption does not hold: the ICMPv6 specification explicitly states that the address being probed should be used as the source address for ICMPv6 responses, providing it belongs to the responding node (RFC 2463). The Rocketfuel [6] work relies upon the correlation of IPv4 packet sequence numbers. However, this approach cannot be directly applied to IPv6 since fragmentation is not supported and thus the sequence number field no longer exists in the IP header.

Atlas attempts to find addresses belonging to the same router based upon the processing model shown in Fig. 2. It relies on the assumption that routing header processing is separate from, and occurs in advance of, delivering packets to the TCP/UDP layers. To ascertain the equivalence of two addresses X and Y, Atlas first performs a traceroute to Y via X. When the first probe packet reaches router X, at hop distance h , it first swaps the address X in the destination field with the final address Y contained in the routing header. Next the hop limit is checked. Assuming the value is 1 an ICMPv6 “hop limit exceeded in transit” message response is triggered. Because the destination address field of the probe packet is now Y, the source address of the ICMPv6 response also becomes Y. The next probe packet with hop limit $h + 1$ is delivered to the UDP layer, causing a “port unreachable” response. Thus, if X and Y belong to the same router, the trace (X to Y) will report Y–Y, and the trace (Y to X) will report X–X.

This scheme works for most cases, except for routers not complying with the ICMP reference model and for a specific topology that we will now discuss. Fig. 8 shows two interconnected routers with anonymous interfaces that are egress toward the probe engine. Suppose both routers select these anonymous interfaces for their ICMP responses, then the equivalence test applied to X and Y would report X–X and Y–Y, and therefore falsely conclude that they belong to the same router. Atlas checks additional information in the ICMPv6 response to reduce cases of this false equivalence. Even if the reported paths are Y–Y and X–X, these two addresses are not perceived as equivalent if the remaining hop limit values of the ICMPv6 packet are different (i.e., the response packets travel paths of different lengths back to the probe engine).

4.6 Restricted Routing Areas

In our experiments we have also observed areas of the 6Bone network that include routers that are not directly addressable from the probing node. That is, given a previously received path U–V–Y–Z–W, addresses Y and Z may not be directly routable from P;

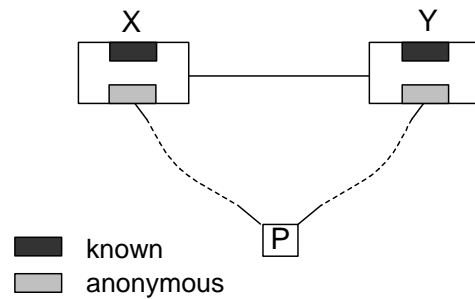


Figure 8: A topology based on two interconnected routers which proves problematic to the address equivalence test.

they exist in a *restricted routing area* (see Figure 9). Consequently, a probe between these and a router revealed by some other path will fail. Thus, links X–Y and X–Z cannot be discovered.

The phenomenon is indicated by the receipt of the ICMPv6 messages “communication with destination administratively prohibited” and “no route to destination”. There are two principal causes of this behavior. The first is the existence of a firewall that is preventing routers being directly addressed in order to decrease the likelihood of being targeted in Denial-of-Service attacks. The second cause of restricted routing is when addresses that are assigned to router interfaces in a given administrative domain are not aggregated within routes that are propagated through exterior routing protocols such as BGP. For example, a given administrative domain may be assigned the address space $3ffe:80a::/32$. The administrator may then decide to assign $3ffe:80a::1-3ffe:80a::9$ to the routers and $3ffe:80a:1::/48$ to the hosts. The cause of the restricted routing area is the decision to only advertise network prefix $3ffe:80a:1::/48$. Hence, the rest of the network cannot determine a route to the routers within this administrative domain. These routes are, of course, locally advertised through interior routing protocols or through the use of static routes.

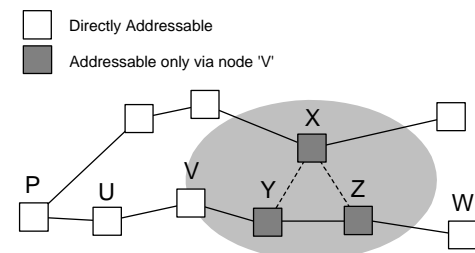


Figure 9: Restricted routing areas.

The restricted routing area phenomena is also identified by Mercator [4]. However, unlike Mercator, Atlas is able to determine connectivity within restricted routing areas. We have observed that routers at the edge of the administrative domain often maintain routes to routers within restricted areas. For instance, router V in Fig. 9 may participate in both interior and exterior routing domains; it is both globally accessible and able to forward packets to routers within the restricted routing areas. Atlas exploits this by performing an additional level of source routing, that is, applying two via-routers to the probe packets. For example, to discover links X–Y and X–Z, probes are sent to Y and Z firstly via V and then via X. The first level directs the packet through the border router and

the second is used to reveal connectivity within the restricted routing area. This technique does depend on locating the appropriate border routers, which is only possible after the probe engine has collected the initial data. Of course, this solution is not effective against firewalls since they are likely to filter out all source-routed packets.

4.7 Routing Loops

Atlas has also proved to have uses outside of simple topology data retrieval. One interesting side effect of the probing scheme is the ability to detect routing loops within the network. Although routing loops do not directly lead to erroneous paths, they must be handled by the implementation to avoid infinite processing.

Routing loops occur as a result of incorrect routing tables, in most cases caused by incorrect static routes (as opposed to routes learnt through routing protocols such as BGP, RIP and OSPF). This phenomenon is shown as repeating patterns in paths. For example, a path U-V-Y-X-Y-X-Y-X-Y,... indicates a routing loop between routers X and Y. Of course, the loop may exist between more than two routers, but in our experience two is the common case. The size of the path returned is bound by a parameter defining the maximum hop limit of the probe packets. This parameter should be equal to the maximum diameter of the network being probed multiplied by two (since we are source routing). In our initial experiments a value of 25 seems to be adequate.

Paths that include loops can be pruned and handled in the normal way, except when anonymous nodes are present. For instance, if in the previous example Y had actually been anonymous, the resulting path would be U-V-Y1-X-Y2-X-Y3..., where Y1, Y2 and Y3 appear to be different anonymous nodes. In this case, the repeating pattern must be extracted, leaving the correct path U-V-Y1-X.

5. PRELIMINARY RESULTS

The Atlas probe engine, topology constructor, and topology verifier are currently implemented on a Linux platform. The visualization tool runs on Microsoft Windows as an ActiveX object, so that it can be embedded in a web page.

5.1 Assessment of Effect Coverage

Using a simulated network allows us to directly compare the actual topology with that inferred by Atlas. The NS-2 simulator was extended to support source routing and the Atlas probing algorithm. Random topologies of certain configurations were generated, varying in terms of the number of non-anonymous nodes, anonymous nodes, links, and seeds. To ensure an appropriately connected graph, the topology generation process first creates a spanning tree and then introduces additional links between pairs of nodes. Note that the current simulation does not incorporate any of the network phenomena as previously described, except anonymous nodes, as this would significantly increase the complexity of the simulation.

A representative set of parameters were chosen for the simulation, although a more extensive assessment would require additional configurations to be simulated. Simulating a single network topology of 1000 nodes and 3000 links takes approximately three days. Therefore, the simulated network was limited to 50 nodes, with 10% of these anonymous. Number of links was varied from 50 to 120, and connectivity varied from tree-like to well-connected structures (average node degree of 4.8).

Fig. 10 shows the results from the simulation. The graph illustrates the relationship between number of seeds used by the probing algorithm and the resulting effective node coverage. Data points were derived by taking average values over 30 random graphs of

equal size (in terms of number of nodes and links). A similar relationship exists for link coverage (as opposed to node coverage) although this graph is not shown.

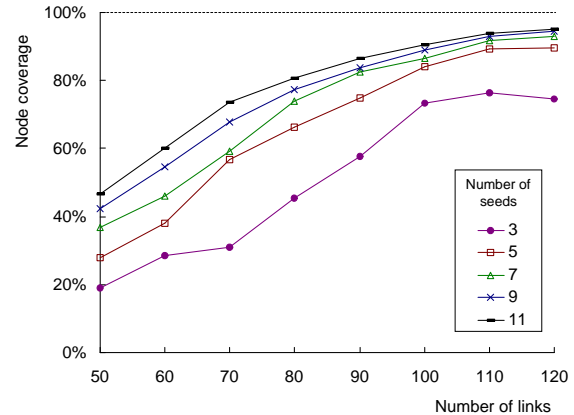


Figure 10: Coverage of probed networks of different connectivity as the number of seeds vary.

These results illustrate that coverage increases when more seeds are used. Coverage also increases according to the degree of connectivity of the topology being probed. It is evident that only a small number of seeds (approximately 10% of the total number of nodes) are needed to discover a large portion (approximately 80%) of a well-connected topology. Nonetheless, the same number of seeds result in a much smaller effective coverage for tree-like structures (less than 30%). As a consequence, Atlas needs only a small number of seeds to discover the transit part of a network and many more seeds to reveal a comprehensive topology of any edge networks that are typically less connected.

As previously mentioned in Section 3.1, seed selection directly affects the effective coverage of the discovery algorithm. More specifically, seeds selected in close proximity to each other reveal less of the topology than an equal number of “well-spaced” seeds. This is partially reflected in Fig. 10 where the curves for the larger seed counts are smoother than those of a smaller count; coverage is more sensitive to seed selection when fewer seeds are used. Therefore, random selection is less optimal than some more careful algorithm. Investigation into possible heuristics for seed selection given partial topology information is a topic of future work.

5.2 Results from the 6Bone

The original motivation for this work was to collect data from the 6Bone network to assess its evolving topological characteristics. To begin with, 426 seeds were gathered by probing addresses that were derived from prefixes listed in the 6Bone registry. Starting with these seeds, the probe engine ran for eight weeks (starting July 2002) and discovered approximately 2420 routers. The following table lists some numerical results from this trace:

5.2.1 Graph Diameter and Maximum Path Length

One can observe that the constructed graph diameter (i.e. the maximum shortest path between any two nodes) is relatively small. We believe that the likely cause of this is extensive deployment of IPv6-in-IPv4 tunneling. Tunneling mechanisms essentially provide ‘virtual hyper-links’. That is, a single IPv6 hop may correspond to multiple underlying IPv4 hops. Consequently, large physical distances are often mapped to a relatively small IPv6 hop distance. For

Table 1: Summary of results from the 6Bone trace.

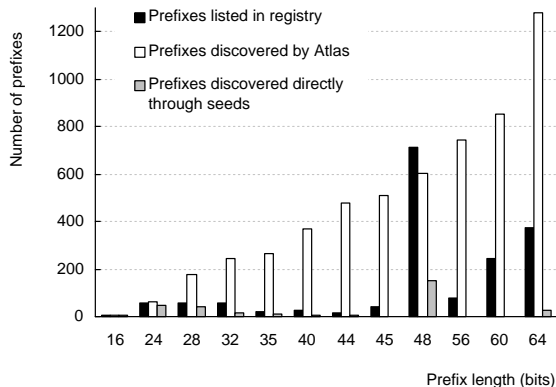
Seeds	426
Paths probed	308887
Non-anonymous interfaces discovered	2951
Anonymous interfaces discovered	44
Routers discovered	2420
Links discovered (before merging)	822189
Links discovered (after merging)	2665
Anonymous interfaces after merging	44
Constructed graph diameter	7
Maximum path length	17
Instances of inconsistent ICMP response	139070
Instances of address unreachable (not unique)	35823
IPv4 compatible addresses	2

example, a traceroute from our testbed to *alt.ipv6.sics.se* reveals 21 hops by IPv4 and only 6 by IPv6.

It is also interesting to note that the maximum length of any given path in the 6Bone trace is 17. This appears to conflict with the graph diameter and suggests that routing is not always shortest path. At first glance, one might infer from this that routing policies are leading to inefficient routing (providing that hop distance is considered a useful metric in determining the best path). However, whilst the effect is most certainly a consequence of routing policies, one cannot necessarily infer inefficient routing. We believe that this result reflects the deployment of link-state routing protocols whereby metrics other than the IPv6 hop distance are being used. More importantly, we believe that IPv6 path weights are likely being derived from the underlying IPv4 tunnel connectivity.

5.2.2 Comparison with Known Prefixes

To understand what proportion of the 6Bone our trace represents, Fig. 11 gives data that defines the relationship of discovered prefixes to those with *matching* prefixes listed in the 6Bone registry (a match is defined by equal bits up to the respective prefix length). The data points on the *x*-axis are derived from the most frequently appearing prefixes in the registry.

**Figure 11: Coverage of 6Bone registry**

It is evident that most of the prefixes listed in the registry have a match in the collected data. However, Atlas appears to discover many addresses that do not have a matching registry entry. We observe that the Atlas system discovered many addresses that have prefixes different from those of the seeds (recall that the seeds were derived from entries in the registry). The graph also shows that

the registry has some 48-bit prefixes which do not seem to be discovered by Atlas even though they were given as seeds. Careful examination reveals that a small percentage of these prefixes are not reachable, i.e. Atlas receives “destination unreachable” messages before the probes are able to reach many of these networks. A likely reason for this is that it is currently relatively simple for a user to get allocated a 48-bit and 64-bit prefix free of charge (e.g. through a public tunnel broker). As a consequence, these network and site-level prefixes are being assigned to very small networks and even single end-systems. Hence, in many cases there is not even a single router handling these assigned prefixes. Of course, the 6Bone address space is considered experimental (RFC 2450). Future IPv6 address allocations are likely to be more controlled.

Another cause of this unreachability is that IPv6 supports multi-homing (i.e. a single site can use multiple prefixes) which may not be configured appropriately. In fact, our testbed has two prefixes; the one listed in the 6Bone registry was first obtained and then later left unused. Probing an address with such a prefix would be unsuccessful because the prefix is not actually being used.

The data used to construct the graph in Fig. 11 offers some indication of the number of different sub-networks in existence at the time the traces were made. In order to analyze the evolution of the 6Bone, Atlas will need to continue probing the network over time. The data collected can be used to calculate given characteristics such as degree of connectivity (average and frequency), maximum hop distance (graph diameter), and address allocations (prefix frequency). However, further work is needed to interpret this data and ultimately assess issues such as the impact of multi-homing and IPv4/IPv6 tunneling.

Finally, to assess the advantages gained from the use of source routing, simple traceroutes were made to the generated seeds and the subsequently discovered addresses. This approach proved to be much less effective than that of the Atlas system. In fact, simple traceroute was able to discover only 617 nodes, just 45% of those revealed by Atlas.

6. RELATED WORK

Path or network topology information can be obtained from different sources, including routing protocols (e.g., BGP, OSPF, IGMP), management protocols (e.g., SNMP), and control protocols (e.g., ICMP). Each source typically has various advantages and disadvantages, which we now examine.

Topology information is an integral part of many routing protocols. These protocols maintain a partial or complete topology according to some model of the network. For example, BGP peers model the network through an AS-level topology (RFC 1771), whereas OSPF does so through a router-level topology (RFC 2328). The Route Views project [11] collects BGP routing protocol information by the distribution of BGP peers (known as “looking glass” servers). The data collected by these servers is then used to construct an AS-level network topology. Results from this work have been used by many other research efforts concerning BGP protocol and general IP routing analysis. The Route Views data has also been applied to work on multicast with respect to efficient spanning tree construction [12].

One of the most commonly used network management protocols is SNMP (RFC 1157). This protocol is typically deployed on routers and devices to facilitate retrieval of their traffic and status information. SNMP data is held within a Management Information Base (MIB). MIBs belonging to routers are a useful source of connectivity information and have been used by other work to derive both logical and physical network topologies [13, 5]. The principal drawback of using SNMP-based data collection for topology

construction is the inherent need for administrative access to the routers. Hence the technique is only suited to private networks whereby appropriate access privileges can be assumed. The usefulness of SNMP is further limited within the context of IPv6 networks, since many IPv6 routers do not yet support the IETF defined IPv6 MIB (RFC 2465).

More closely related to Atlas, other work has focused on the use of ICMP-based tools (such as *ping* and *traceroute*) to collect topology information. The Internet Mapping project [2] uses IPv4 traceroute (without source routing) to retrieve path information from probing points distributed throughout the Internet. Again, this approach does rely upon some access to the network infrastructure so that probing devices can be suitably deployed. However, the projects algorithm is not exhaustive and omits to reveal many cross links in the network. This is partially evident from the maps that resulted from this work, in which independent trees are readily identifiable. Work from the Mercator project [4] is also relevant. Like Atlas, Mercator uses source-routed traceroutes to collect path information, although they do limit this to IPv4. Certain network phenomena, as described earlier in Section 4, have been partially identified by this work. However, many of the heuristics in Mercator cannot be readily applied to IPv6 networks and many of the problems arising from network behavior have not been fully addressed.

Finally, it is worth noting that many existing topology discovery techniques that have been developed for IPv4 networks cannot be applied directly to IPv6. Reasons for this include changes in the relevant protocols (e.g. ICMP, SNMP) and a larger address space (e.g. making exhaustive address space probing infeasible).

7. CONCLUDING REMARKS

This paper describes Atlas, an automated topology discovery system for large-scale public IPv6 networks. Through careful deployment of ICMP-based probing and source routing, the system's probe engine is able to collect raw path information for a large part of the network. In turn, Atlas is able to construct a router-level topology of the underlying network. The data collection process itself has proved to be an intricate problem because of the complex and unpredictable behavior of the networks being probed. This paper has highlighted some network phenomena that were encountered in our own experiments, including varied router responsiveness, anonymous interfaces, instable routing, address equivalence, restricted routing, and routing loops. Furthermore, it has been shown that correct handling of these phenomena is vital to the integrity of the topology data.

Although deployment of IPv6 is still relatively small, it is nevertheless gaining pace. The ability to accurately monitor the changing topology of IPv6 networks, such as the 6Bone, will lead to better understanding and decision making in the development of this new protocol. Our preliminary results already indicate extensive use of tunneling and routing policies, and highlight the effects of uncontrolled prefix allocation.

Existing IPv4 topology probing techniques cannot be readily applied to IPv6, and therefore new solutions are required for this domain. The contributions made by this work are novel in both realizing a system that is able to probe IPv6 networks and construct their respective topology, but also in understanding IPv6 network behavior in general. Future work of Atlas may extend in number of directions. These include mapping IPv6 topologies to underlying IPv4, analyzing path information in order to infer routing policies and address aggregation, and also studying network dynamics such as stability and growth.

8. REFERENCES

- [1] A. Bestavros, J. Byers, and K. Harfoush, "Inference and labeling of metric-induced network topologies," in *Proc. IEEE INFOCOM 2002*, 2002.
- [2] B. Cheswick, H. Burch, and S. Branigan, "Mapping and visualizing the Internet," in *Proc. USENIX 2000*, June 2000.
- [3] J. Pansiot and D. Grad, "On routes and multicast trees in the Internet," *ACM Computer Communications Review*, vol. 28, no. 1, 1998.
- [4] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet map discovery," in *Proc. INFOCOM 2000*, March 2000.
- [5] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz, "Topology discovery in heterogeneous IP networks," in *Proc. INFOCOM 2000*, March 2000.
- [6] N. Spring, R. Mahajan, and D. Wetherall, ,
- [7] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology inference in the presence of anonymous routers," To Appear in INFOCOM'2003.
- [8] T. Munzner, "H3 : Laying out large directed graphs in 3D hyperbolic space," in *Proc. IEEE Symposium on Information Visualisation*, October 1997.
- [9] "6Bone Registry," <http://www.viagenie.qc.ca/en/ipv6/registry/>.
- [10] V. Paxson, "End-to-end routing behavior in the Internet," in *Proc. SIGCOMM 1996*, August 1996.
- [11] University of Oregon, "Route views project," <http://www.routeviews.org/>.
- [12] N. Duffield, J. Horowitz, and F. Lo Presti, "Adaptive multicast topology inference," in *Proc. IEEE INFOCOM 2001*, 2001.
- [13] G. Mansfield, M. Ouchi, K. Jayanthi, Y. Kimura, K. Ohta, and Y. Nemoto, "Techniques for automated network map generation using SNMP," in *Proc. INFOCOM 1996*, April 1996.
- [14] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. INFOCOM 1996*, April 1996.
- [15] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *Proc. SIGCOMM 1999*, 1999.
- [16] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "The origin of power laws in Internet topologies revisited," in *Proc. INFOCOM 2002*, 2002.
- [17] P. Francis, "Comparison of geographical and provider-rooted Internet addressing," *Computer Networks and ISDN Systems*, vol. 27, no. 3, 1994.
- [18] D. C. Wood, S. Coleman, and M. F. Schwartz, "Fremont: A system for discovering network characteristics and problems," in *Proc. USENIX 1993*, 1993.
- [19] V. Jacobson, "Traceroute software," Lawrence Berkeley Laboratories, December 1998.
- [20] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A global internet host distance estimation service," *IEEE/ACM Transactions on Networking*, vol. 9, no. 5, 2001.
- [21] T.S. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc. IEEE INFOCOM 2002*, June 2002.
- [22] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *Proc. ACM Symposium on Theory of Computing*, 2000.

- [23] E. Zegura, K. Calvert, and M. Donahoo, "A quantitative comparison of graph-based model for Internet topology," *ACM/IEEE Transactions on Networking*, vol. 5, no. 6, 1997.
- [24] T. Bu and D. Towsley, "On distinguishing between Internet power law topology generators," in *Proc. IEEE INFOCOM 2002*, June 2002.
- [25] C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkatachary, "The impact of Internet policy and topology on delayed routing convergence," in *Proc. IEEE INFOCOM 2001*, 2001.
- [26] S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 3, 1997.
- [27] "6Bone," <http://www.6bone.net>.