Eliminating Machine Duplicity in Traceroute–based Internet Topology Measurements

Hal Burch

May 2002
CMU-CS-02-146

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

One of the hurdles faced by Internet topology meauserments is machines appearing within the induced topology many times, each time with a different IP address, sometimes many hops apart. Most topology measurements are based on traceroute, which may result in a machine responding with different IP addresses in different traceroutes. There are three major techniques known for finding or detecting pairs of IP addresses belonging to the same machine. However, two of the three techniques naively require quadratic packets in the number of IP addresses to test. This paper presents practical, scaleable algorithms for each technique, using three novel methods to divide the input set to make the quadratic techniques practical on large sets. For each technique, the error is analyzed, looking at both the source and amount of error the technique exhibits, as well as looking at how responsive machines on the Internet are to the technique.

Keywords: Internet, network measurement, network topology, traceroute, IP aliasing

1 Overview

Several ongoing projects are attempting to determine the IP connectivity of the Internet or pieces of it [8, 2, 7, 3, 13], mostly based on traceroute [4]. Traceroute sends out packets with low time to live (TTL) and expects ICMP time exceeded responses in order to determine the machine k hops along a path. By varying k, one can determine the complete path. Presuming that hops collected represent a coherent path, adjacent pairs of machines are connected on the IP layer.

One of the main problems with this approach is that the responses do not contain the name of the machine. Rather, they contain an IP address. Of course, a machine often has multiple IP addresses, which can distort the resulting topologies. In particular, a router normally has one IP address for every physical or virtual interface. Machines are free to make the source IP address of the response be any of their IP addresses [10]. Some machines choose the responding IP address based on the interface the packet came in, some on the interface the ICMP error message goes out, and some always choose the same IP address. Except in the last case, this means that data collected could contain the same machine listed several times under different IP addresses. This problem, called the "IP aliasing problem" is exacerbated for scans separated by time or done from multiple locations.

Listing machines multiple times under different IP addresses causes a variety of problem with the topology produced. First, adjacency is incorrect, as some IP addresses belonging to adjacent machines may not be listed as adjacent, causing nodes to be farther apart in the topology than they are in reality. Second, the topology is larger than the network measured, both in terms of machines and links. Third, this causes problems in path prediction, as predicted path given in terms of IP addresses may not be the same as the one measured because the machines choose to return different IP addresses than expected.

This paper explores techniques for transforming IP address maps to machine maps. It compares several techniques and explores results from specific implementations. There are three basic ways to determine machines from IP addresses: direct methods, bucketing techniques, and implied methods.

Direct methods, such as UDP port unreachable [8], query the machine implicitly or explicitly for IP address of other interfaces. They send packets to a machine expecting another IP address to be listed within the reply.

Bucketing techniques look for similar behavior between IP addresses. Here, packets are sent to two IP addresses and the responses are compared for similarity. The IPid technique is one such example [13], looking for similarities in the IP identification field of the replies.

Implied techniques look for invoked behavior. Here, behavior that is expected within a machine is exploiting to reveal the matching. One example is using the rate limits within the machine, where the fact that a packet has been sent to one IP address of a machine causes the machine not to respond to a second packet sent to a different IP address.

Direct methods are generally linear time, as each machine is queried in order. Bucketing techniques and implied techniques, on the other hand, are quadratic. Since the input sets can be large (almost 380,000 for the data sets used within this paper), quadratic algorithms are too slow to work in practice. Thus, some division of the input data set may be required before the actual algorithm is run.

This paper will explore three algorithms for solving the IP aliasing problem and three techniques to split the input set, looking at the effectiveness of each. Section 2 explains the three IP aliasing techniques, while section 3 details the splitting techniques. Section 4 gives the details of the algorithms employed. Section 5 lists the results of the experiments, and section 6 discusses conclusions and future work.

2 General Techniques

2.1 UDP Technique

Pansiot and Grad observed the same problem of IP aliasing when mapping multicast trees [8]. They used UDP port unreachables to find pairs of IP addresses belonging to the same machine. They sent a UDP packet to a port where they did not expect a service to be listening. Machines receiving these packets then replied with ICMP unreachables, specifically port unreachable. They noted that the source IP address of the ICMP error packet was not always the same as the destination of the UDP packet. It is unlikely that a different machine would respond with an UDP port unreachable than the destination of the UDP packet. Therefore, they used such replies as evidence that the source of the ICMP error packet and the destination of the UDP packet are probably the same machine.

This direct technique is nice in that it requires sending only one packet to every IP address you wish to probe. Many machines pick the source of the ICMP error packet as the IP address of the outbound interface, so this source address is constant over all interfaces. However, this methodology does not work for machines that always pick the source of the reply to be the original destination.

Govindan and Tangmunarunkit [3] expanded on this idea by sending multiple packets to each IP address, spaced out in time. This improved the discovery by exploiting route changes. In addition, they employed source routing [11] to make the packets come into the machine from different interfaces. Barford, et al. [1] also used the UDP technique, although without the additional features of Govindan and Tangmunarunkit.

More recently, Keys wrote the program `iffinder` [5], which uses several techniques for finding interfaces, but the most successful technique by far is the UDP technique mentioned above.

The primary source of false postives in this technique is machines responding with source addresses in private address space as defined in RFC 1918 [12]. Such replies must be discarded, as there is an overlap of usage of this address space. In addition, some machines respond with canonically bad IP addresses, such as 0.0.0.0, or the IP address from which the original UDP packet came.

2.2 IPid Technique

Spring, et al. [13] used the IP identification field as a bucketing technique to detect multiple IP addresses belonging to the same machine. Many machines set this field using a global counter. Thus, if you send two packets to a machine, you will get back similar IPid values unless the machine is sending an extremely large number of packets. Spring, et al. designed their algorithm to work on the smaller data sets. It is not clear from the paper how they decide which pair of IP addresses should be tested. This paper will present an algorithm designed for larger data sets and an improved verification algorithm, and will also present a more formal analysis of the technique.

This IPid field provides a natural use in verifying that two IP addresses belong to the same machine, as you would expect the IPids within the responses to be similar. Discovering the pairs of IP addresses is a bit more difficult. Here, you must send packets to a large number of IP addresses and look at the IPid field values in the responses to determine which addresses may belong to the same machine. If you have a large number of IP addresses, this becomes difficult, as you want to be able to detect if the first and last IP address belong to the same machine. If the rate of IPid change multiplied by the time it takes to send packets to every address exceeds 65,536, it is impossible to garner data on whether the first and last IP addresses to which packets were sent belong to the

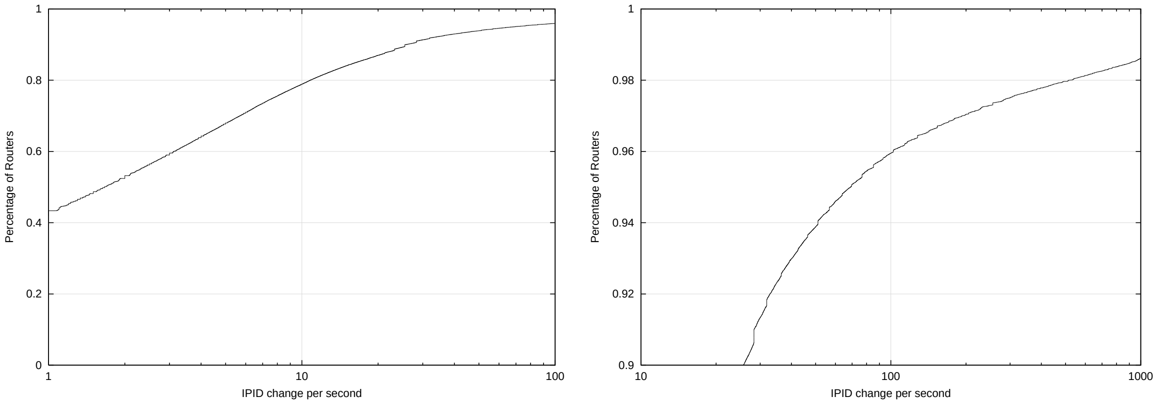

Figure 1: Cumulative distribution of IPid change rates for 102,225 IP addresses, mostly routers

same machine. Since the rate that packets can be sent out is limited by several factors, including the bandwidth of the connection to the Internet and the speed at which the testing machine can process packets, it is important that the rate of IPid change is small.

Figure 1 shows the distribution of IPid change rates of IP addresses reachable over the Internet. These are based on measurements done on November 5, 2001 at 19:00 GMT. The rate of change was measured over about 12 seconds, so rates above 5,000 were incorrectly measured, but the number of such machines is extremely small. Responders that send constant IPid values were discarded (total of 1,011 (1.0%), not included in the 102,225). Many of the higher rates are machines that increment in big-endian fashion or pick IPid randomly. This figure shows that the majority of the tested machines have slow IPid change rates: 79% change at slower than 10 per second, and 96% change at slower than 100 per second.

Let R be the maximum rate of IPid change you want to support, K be the number of IP addresses you wish to test, and P be the packet rate. Assume that the current IPid counters for all machines are initialized randomly, that there are no pairs of IP addresses belonging to the same machine, and that round trip times are zero. The probability of two packets being paired improperly is the product of the time between the queries of the IPid ($\frac{j}{P}$, where j is their distance apart in the order that the IP addresses are queried) and the maximum IPid change rate supported (R), divided by the range of possible IPid values, 65536. Thus, the total number of expected false positives is:

$$
\begin{aligned}
\sum_{i=0}^{K-1}\sum_{j=1}^{i}\frac{jR}{65536P} &= \frac{R}{65536P}\sum_{i=0}^{K-1}\sum_{j=1}^{i} j \\
&= \frac{R}{65536P}\sum_{i=0}^{K-1}\frac{i(i+1)}{2} \\
&\approx \frac{R}{131072P}\sum_{i=0}^{K-1} i^2 \\
&\approx \frac{R}{131072P}\frac{K^3}{3} \\
&\approx \frac{RK^3}{400000P}
\end{aligned}
$$

$$\approx \quad C\left(K,2\right)\frac{R}{200000P}$$

Note that this formula only works for $\frac{RK}{65536P} < 1$, since otherwise a machine's IPid counter may roll-over within the testing window. More involved techniques are necessary to derive any data if $K > \frac{65536P}{R}$.

In practical terms, if $R = 100$, $K = 100,000$, and $P = 1,000$, this means you will have around 250 million false positives, or about 5% of all pairs. Thus, multiple phases or splitting the set is essential to this method's success on medium to large collections of IP addresses.

2.3 Rate Limit Technique

One (very unfriendly) idea for determining if two IP addresses belong to the same machine is to bring down the machine belonging to one of the IP addresses and determine if the other IP addresses still responds. This has the obvious problem that it is extremely antisocial. More technically, you cannot distinguish an IP address belonging to the machine and an IP address that is beyond the machine unless you bring down the machine but not the routing level.

However, the entire machine does not need to be brought down; you need only that the machine not respond to your packets. Some machines on the Internet rate-limit the responses they send. The common restriction is to send no more than one UDP port unreachable response per second. These restrictions are often not interface specific, so they will only send one UDP port unreachable response per second, regardless of the source or destination of those packets. Thus, if you send two UDP packets to a machine in a second, you will only get one response.

This rate limitation is normally set-up as only sending a packet every second as determined by the internal clock. Thus, packets received 100 milliseconds apart will elicit a response only if the second ticks between receiving the packets. Thus, multiple experiments are required to avoid being fooled by this clock tick and drawing the false conclusion that the two IP addresses do not belong to the same machine.

The main source of false positive results in this implied technique is packet loss. If the second UDP packet is lost or the response is lost, the algorithm would falsely conclude that the two IP addresses belong to the same machine. Again, multiple experiments are required to avoid this problem.

This general technique was also mentioned by Spring, et al. [13]. However, they ran ran only one pass of this verification, running the IPid test twice, the second time in the reverse order of the first. If only the first packet elicited a reply both times, they concluded the IP addresses belonged to the same machine. In our experiments, an IPid test was inconclusive (did not get a response to both IP addresses) only 0.3% of the time, so it is not clear how much this was really employed.

3 Splitting Techniques

There were three splitting techniques employed: the null operation, TTL splitting, and adjacency splitting. The null operation just returns the original set, so it will not be discussed in any detail. Adjacency splitting does not produce disjoint sets.

3.1 TTL Technique

Each IP packet has an eight bit field called "Time to Live" (TTL). This field is decremented by one for each hop a packet traverses. If the field reaches zero before the packet reaches its destination, the packet is discarded. This insures that packets on the Internet have limited lifetime.

Most machines initialize the value of the TTL to a value that is independent of the destination. Moreover, most machines decide which interface to send out a response packet based purely on the destination (as opposed to the interface the original packet came in). Thus, sending a packet to two IP addresses of the same machine results in two packets with the same TTL. This provides evidence that the two IP addresses belong to the same machine.

There are two problems with this technique. The first problem with this approach is routes may change in the middle of an experimental run. Thus, the TTLs coming from the same machine may vary, either due to true routing change or load balancing. Fortunately, load balancing over paths of different lengths is unusual and most routes are static over reasonable time spans (minutes to hours) [9].

The second, and larger, problem is that the TTL field has only 255 possible values, and the distribution is highly skewed. Initial TTL values are not distributed uniformly: There are six common initial TTL values: 30, 32, 64, 128, 150, and 255, with 255 being the most common by a factor of almost four. Distances are clustered around 12 and 13 for the measuring point used, so there is a large spike of packets observed with TTLs of 242 and 243, with each value representing about 8% of the nodes. Performing additional tests, either at different times from the same location or from different locations, does not improve the situation enough. This is because the distance of two IP addresses within the same autonomous system to any measurement point is highly correlated.

The coarseness of this technique makes it unsuitable for direct measurement. However, it does provide a nice way to split the data set into sets, reducing the size of the largest set by about twelve.

3.2 Adjacency

Another method to split the IP addresses is based on the graphical union of the source data set. The IP addresses came from a collection of traceroutes done over many days. A topology is induced from these traceroutes by assuming that a hop in a traceroute represents direct IP connectivity. The adjacency splitting technique produces sets of size two representing all pairs of IP addresses that are two hops apart in the induced topology.

This works on the premise that two IP addresses of the same machine are likely to be adjacent to the same node. Consider adjacent machines A and B. On day one, the daily scan sees machine A uses A1 in its time-exceeded responses and machine B uses B1 along a traceroute. The next day, machine A changes its decision and uses A2 instead, but machine does not change its decision. Machine A may change which IP address to use because the traceroute comes in on a different interface or because its routing table has changed and now has a different next hop for the measurement point. Both of these can occur without a corresponding change in machine B. In such cases, A1 and A2 will both be adjacent to B1 within the resulting topology.

Note that this technique is expected to have worse coverage. In addition to having a smaller number of valid pairs remaining within the set of consideration, this technique will also not catch the case of two IP addresses of a machine appearing far apart within the topology. The major advantage of this technique is that the number of pairs to test is almost linear in the number of IP addresses.

4 Specific Algorithms

There are five algorithms of note: the packet collection algorithm, the TTL splitting algorithm, and the actual IP alias resolution algorithms. The adjacency and null splitting algorithms are straight-forward and need no further detail, so they are not covered.

Each algorithm takes a set of IP addresses as input. They can send packets of any type to any IP addresses in order. Their output is a set of pairs of IP addresses that are believed to belong to the same machine.

A set of pairs can be transformed as a set of groups of IP addresses, where each group of IP addresses is believed to belong to a single machine. These groups are created by first creating a group of size 1 for each IP address, and, for each pair, combining the groups to which they belong. Note that a pair of IP addresses within a group may not appear within the original set of pairs. Such pairs are known as "false" pairs. A "false" pair may be the result of a false negative (the algorithm did not output a correct pair) or false positives (the algorithm outputted one or more incorrect pairs). While a false negative produces at most one "false" pair, a single false positive can produce many.

These groups can be used to refine the deduced network topology within a network measurement. In particular, each of these groups is combined to a single node. After this combination, the graph is reduced to a simple graph by removing any self-loops and removing duplicate instances of any edge.

Note that pair of IP addresses belonging to the same machine but not within the output set is considered a false negative only if the machine is responsive to the technique. A false positive is any pair of IP addresses within the output set that belong to the two or more machines (one IP address may belong to multiple machines, although this is rare when the IP address is not within a private address ranges).

4.1 Packet Collection

All the techniques listed within this paper use UDP packets. Whenever packets are collected from a large set of IP addresses, special care must be taken to improve response rates.

When sending packets to a large set of IP addresses, there are two complicating features. First, replies must be matched with requests. The only type of packets used are UDP packets. UDP packets elicit ICMP port unreachable replies, which have a partial copy of the original packet within them, so this does not seem especially difficult. Unfortunately, the destination listed within these enclosed packets is not always the same as the destination of the original packet. The author is not certain what causes this, but one possibility is improper network address translation (NAT). This matching is therefore done by selecting the destination port in the original UDP packet.

Using high numbered ports (above 32,768) reduces the appearance of an attack, but this means that if more than 32,768 IP addresses are to be tested, port numbers must be reused (in fact, the results described in this paper use only 28,000 unique port numbers). The port numbers are used in rotating fashion, to maximize the time between reuse. Packet rates were all below 2,000 packets per second, so this meant that as long as round trip times were below 14 seconds, the matching could be done. We used a rotation window to match these up. Unfortunately, a very small number of machines did not properly list the original destination port in the reply. This was only observed twice within all the IPs tested, but it would only be detected 57% of the time if the source port is selected randomly (28,000 ports would be considered valid out of 65,536), so it probably occurred more times just the two detected, but definitely at a very low rate.

The second problem is that not all packets elicit responses. Non-response can be caused by routing problems, packet loss, rate limits (the same behavior the rate limit technique exploits), firewalls, and machines being disconnected from the network. Thus, obtaining reliable response rates from the addresses requires multiple passes. The algorithm continues to do passes as long as a third of the IP addresses are responding. This gives over a 93% response rate, with 83% of the addresses being reliable responders (responding to seven of seven attempts using this algorithm).

4.2 TTL Splitting

Splitting the input set based on TTL is relatively straight–forward. Using the packet collection technique described above, a response to a high–port UDP packet is collected for each IP address. The IP addresses are divided into sets based on the TTL of the responding packet. Any machine that does not respond to the packet collection attempt is discarded.

4.3 UDP Technique

As mentioned above, this algorithm is relatively straight-forward: send a UDP packet to each IP address and look at the source address of the reply packet. Due to rate limitations and packet loss, some destinations must be tried more than once, so the packet collection technique mentioned above must be employed. In addition, replies from private address space and other invalid IP addresses (specifically, 0.0.0.0 and the test machine's IP addresses in this implementation) are dropped to avoid false joins.

4.4 IPid Technique

The algorithm has four phases: TTL splitting, IPid splitting, IPid testing, and IPid verification. Any occurrence of an IP address that fails to respond to even one of the phases is discarded. Some phases have most IP addresses listed in several groups, reducing the impact of this discard. The test of each group is run independently, so not responding or responding to the test from one group an IP address is in does not affect the results from any other group it is also within.

4.4.1 TTL splitting

TTL splitting was done using the algorithm detailed in section 4.2. The groups collected are then tested individually. TTL splitting was selected because it was expected to have better coverage than adjacency splitting and the input set (null splitting) exceeded 200,000 in size, so IPid would not work directly.

4.4.2 IPid splitting

IPid splitting attempts to further divide the TTL-split groups into groups of size 2,000 or smaller, to simplify later testing. It sends packets to each IP address and collects the IPid of the response. The IPid range 0-65,535 is split into subranges based on the maximum rate supported and the time it took to collect the packets. The size of the range is the product of the maximum rate and the collection time (with round-off distributed roughly evenly across groups). A group consists of all IP addresses within a certain range and the preceding range (the ranges wrap, so the predecessor of the lowest range group is the highest range). This means that every IP address that responded is listed twice each time the splitting is done.

Presuming that the machine did not change at an average rate exceeding the maximum supported rate and the machine responded for all IP addresses, every pair of IP addresses belonging to that machine exists in some group. In particular, the first responding IP address of each pair is put into a group containing the second responding IP address. If any resulting group is still too large, the process is repeated. Note that the order of sending and receiving the packets is ignored. Also, in order to decrease the size of the largest group, at least three groups must be created, so the number of IP addresses within a single test group cannot exceed $\frac{65536P}{3R} \approx \frac{21845P}{R}$, where P is the average rate of packet transmission and R is the maximum supported IPid change rate. For the experimental results given, $R = 100$ and $P \approx 1000$, which means the groups inputted into this phase cannot exceed 218,450 IP addresses.

4.4.3 IPid testing

IPid testing takes the smaller sets and proposes pairs of IP addresses that may be aliases of each other. It takes groups smaller than 2,000 and finds pairs of IP addresses that may belong to the same machine. Packets are sent to every IP address and the IPid of the responses are collected. Pairs of IPids are then examined to see they are consistent with a machine whose IPid counter is changing no faster than the supported rate, with the assumption that the reverse path delays do not exceed 200 milliseconds (this allows for a good deal of jitter). Pairs of IP addresses that meet this criteria are then outputted. From the equation in the previous section, the percentage of false positives would be expected to be quite low. However, this ignores the fact that this phase is very dependent on the previous splitting work, since 40% of the machines have IPids that change at a rate slower than 1 per second (see figure 1). Due to the high number of false positives, IPid testing is done twice on each set and only pairs that appear in both tests of a set are considered. To decrease the dependence, all sets are tested once and then tested again (as opposed to testing the first set twice, then the second set twice, etc.).

4.4.4 IPid verification

IPid verification is done in sets of 1,000 pairs such that no IP address appears in two pairs. Packets are sent to the first IP address, then to the second, then to the first, and then to the second, with a one second delay between each packet sent to a single pair. The IPids of the responses must be valid with the assumption that the machine responding to both addresses does not change its IPid very quickly. This interlacing ensures that regardless of the initial values, the pair will be discarded if the machines' IPids are not moving at similar rates. For IP addresses with IPids changing at very slow speeds, pairs with similar initial IPid values (most of the false pairs tested), the interlacing is likely to reveal two responses with the same IPid value.

4.5 Rate Limit Technique

To utilize the rate limitations to determine if two IP addresses belong to the same machine, packets must be sent to both IP addresses in less than a second. Since the packet rate was limited to 1,000 packets per second, this means that completely testing a data set of 380,000 IP addresses would require over a year, so pruning is required. To get the experiment to run on the entire data set in less than a day would require reducing the data set to less than 30,000 IP addresses, which greatly reduces the effectiveness and rules out direct use of TTL splitting. Thus, adjacency splitting was employed. This removes the problem of discovering possible pairs and reduces the problem to verification only.

After adjacency splitting is done, each IP address is tested to see if it has any rate limitations. Any pair where at least one of the IP addresses does not have rate limitations is discarded as untestable. The set of pairs is divided into subsets of 150 pairs. Each of these subsets is tested separately. UDP packets are sent to the first IP address in each pair and then to the second. Then, 1,200 milliseconds after the first test started, the test is repeated with the IP addresses swapped. The packets in each test pair are sent out approximately 150 milliseconds apart. The 1,200 millisecond gap ensures that regardless of when the second tick is for the hypothetical machine, one of the tests will not span that tick, assuming limited jitter. A test is considered successful if the first packet invokes a response and the second does not. The test is negative if four packets are received or responses are garnered both times for the second packet. Otherwise, the test is inconclusive.

This test is subject to large error, due to packet loss, interference between pairs, and jitter, so this test is repeated eight times. After the third test, any pair that consistently yields three negative results is discarded, and any pair that yields three positive results is accepted without further testing. The other pairs are accepted only if there were at least two positive results and more than twice as many positive results as negative results. IP addresses that appear in more than 20 accepted pairs and groups with more than 20 IP addresses are eliminated to avoid excessive false positives due to lossy connections.

5 Experimental Results

5.1 Experimental Setup

All experiments were done over a T1 connection connected to UUNet via their NYC PoP. The T1 was used by two dozen people, so it had a background usage level that was relatively low. Other measurements were periodically run on the T1, causing higher utilization. The other measurements were designed not to exceed half the capacity of the T1, so the experiments detailed here were also designed not to exceed half the capacity, to alleviate packet loss concerns.

The input set of IP addresses came from hops within 77 days of traceroute data. After discarding unroutable and private address space, the original input set consisted of 587,828 IP addresses, although only 386,157 were found to be responsive to UDP packets. The traceroutes were daily runs from May 2002 to July 2002, consisting of about 240,000 individual traces per run. This input set was expected to have many machines listed multiple times under different IP addresses, due to the time range covered and collection method.

That only 66% of the IP addresses responded to UDP packets was somewhat surprising. The traceroutes used were ICMP Echo Requests, not UDP packets, so firewalls were one source. 92% of the IP addresses were responsive ICMP Echo Requests. The other 8% are a mixture of machines turned off or retired and routers that forward packets but cannot be directly addressed, either due to firewalls or residing on unannounced networks.

5.2 UDP Results

UDP packets were sent to all of the IP addresses, which took 15 minutes. 385,821 of those responded with valid responses (ICMP port unreachable). Of those responses, several pathologies were detected and ignored. 573 packets contained different destination addresses in the enclosed packet than the original destination. 2,471 of the responses had source addresses in private address space, five had source addresses of 0.0.0.0, and three had responses with a source address equal to

Phase	Total IPs	Groups	Largest Group	Pairs
TTL Splitting	379,843	160	34,954	3,676,963,861
IPid Splitting	375,867	16,969	1,989	233,821,741
IPid Testing, pass 1	356,064	12,115,122	2	12,115,122
IPid Testing, pass 2	280,576	2,965,950	2	2,965,950
IPid Validation	203,698	226,653	2	226,653

Figure 2: Summary of IPid Results after each phase. Pairs is the number of pairs of IP addresses still being considered

the test machine's. Removing these packets left a total of 383,105 responses, of which 67,735 had different source addresses than the original destination.

For each reply whose source address differs from the destination of the corresponding request, the reply source address and original destination address were combined into one group. This created 45,087 groups, the largest having 94 IP addresses within it. Collapsing the IP addresses within each group to a single IP address removed 61,652 IP addresses. 5,878 of the IP addresses in groups were not in the original list. Removing these "new" IP addresses from the groups resulted in the number of groups with at least two IP addresses dropping to 39,931.

This method can produce false positives when the IP address listed as the source of the replies may not belong to the machines within the Internet address space. This is most commonly true when responses are within private address space (which is why the are removed), but some machines respond with other invalid addresses, such as 0.2.0.0, 1.255.1.110, and 2.2.2.5. Besides dark addresses (addresses in use, but not globally routed) [6], network address translation (NAT) can also cause problems.

When a machine is responsive to this technique, false negatives can occur either when a machine does not respond to the queries or when routing changes during the experiment. Either may cause the set of IP addresses belonging to the same machine to be split into multiple sets. Neither is believed to occur often, especially when compared to the number of machines not responsive to this technique.

5.3 IPid Results

Figure 2 summarizes the results from running the algorithm on the input. Note that the drop in the number of IP addresses after IPid testing and validation is due to discarding pairs, not lack of responses. The first two phases serve only to decrease the size of the largest group, to make IPid testing possible as well as to decrease the number of false positives obtained during it. Unfortunately, in order to do this, IP addresses are duplicated. The algorithm itself causes a factor of 2 increase, but the group size limitation means that occasionally the splitting algorithm must recurse, so the average duplication was 2.28, with some IP addresses appearing over a hundred times. The author believes this could be improved with algorithmic improvements, but this level of duplication was deemed acceptable.

The original TTL splitting took 95 minutes. The subsequent IPid splitting took three hours, while each testing run took approximately three hours of data collection and 85 minutes of data processing. IPid validating the implied pairs took 360 minutes (approximately 140 pairs per second).

It is simpler to determine the number of false negatives for this technique than for the UDP methodology. A validation test has four possible results: acception, rejection, failure (did not receive responses to all packets), and skipped (could not find a test set to add the pair to that did not already contain one of the IP addresses of the pair). Validation sweeps are done through the incomplete (not accepted or rejected yet) pairs until at least half of the non-decisions are failures, at which point the pass is stopped, and the remaining unvalidated pairs are considered rejected. Of the 2,965,950 pairs to be validated, 108,005 were not validated because of this. Without the relaxed stopping condition, however, the validation would never complete.

Examining the groups induced by the output pairs, 7,797 pairs of IP addresses belonging the same group did not appear in the list of accepted pairs. Of these, 5,018 appeared in the set of pairs to be validated, giving a false negative rate of the validation phase of approximately 1 in 45 (2.2%), and a false negative rate of the combination of all the phases up to then of about 1 in 84 (1.2%), with a combined false negative rate of about 1 in 30 (3.3%).

This false negative and false positive analysis assumes that the machines are responsive to this technique. As mentioned before, this is not always true, as some machines respond with constant IPid to the UDP test packets. More difficult to detect are machines that maintain separate IPid fields for each interface.

5.4 Rate Limit Results

Five UDP packets were sent in quick succession to each of the 386,157 IP addresses. 377,169 (98%) IP addresses responded to at least one of the packets, with another 686 (0.2%) of the IP addresses resulted in error responses. 139,133 (36%) IP addresses responded to all five of the packets, leaving 238,036 (62%) IP addresses that responded to at least one but not all of the packets. Of these 238,036 IP addresses, 227,368 (96%) responded to only one packet.

Adjacency splitting gave 5,620,847 pairs to test. Eliminating pairs where one or both IP addresses were not limited reduced the set to 2,908,033. The rate limit verification algorithm, as described above, was run on these pairs. After running three verification passes, 84,190 of the pairs were accepted and 2,607,481 pairs were rejected, leaving 216,362 pairs to test further. An additional 23,886 pairs were accepted after the completion of all eight runs.

Of the 69,635 pairs, 606 IP addresses and 20 groups were removed, leaving 55,363 pairs accepted. This defined 29,183 groups, the largest with 20 (of course). The groups represented 81,637 pairs, of which 55,363 (67.8%) were the accepted set. Of the other 26,274 pairs, 6,918 (26.3%) were in the output of the adjacency splitting. This gives an approximate false negative rate of 11.1% for rate limit validation separate from adjacency splitting, ignoring false positives. The false positive rate is probably higher, but difficult to quantify. Over half of the accepted set was discarded by the set size cut-off. Taking the union of the results from the other methods, 3,050 of the pairs are invalid, giving a false positive rate of about 0.10 %. Without the removal of the bad IP addresses and groups, the false positive rate jumps to 0.27 %.

6 Conclusions

6.1 Splitting Comparison

There are three major goals of the splitting algorithms: reduce the set size, minimize the set overlap, and minimize the induced error. Figure 3 summarizes the three techniques. The null splitting clearly does not split many pairs and does not have any overlap, but the set it produces is very large. Adjacency splitting optimizes the largest set, but covers less than half of the pairs. TTL splitting is a nice middle ground, reducing the largest set by a factor of ten while still keeping most of the pairs.

Technique	No. Sets	Total Size	Pair Count	Largest Size	Coverage	Pkts Sent
Null	1	386,157	74,558,421,246	386,157	100.0 %	0
Adjacency	5,620,847	11,241,694	5,620,847	2	59.7 %	0
TTL	160	379,833	3,676,963,861	34,954	84.7 %	466,756

Figure 3: Summary of results for each splitting technique. Pair count is the total number of pairs left to test. Coverage is the percentage of pairs found by the combination of the techniques that appeared in at least one of the sets.

Technique	# Sets	Implied Pairs	IPs Rem.	Coverage	Pkts Sent	"False" Pairs
UDP	45,087	131,518	59,705	38.2 %	469,824	21,465
IPid w/ TTL Split	53,736	234,450	97,719	68.2 %	5,861,007	27,068
IPid Ver. w/ Adj. Split	50,318	203,541	89,674	59.2 %	6,621,023	9,913
Rate Limit	29,176	81,637	44,707	23.7 %	9,805,909	13,887
Combined	63,875	343,720	112,851	100.0 %	22,757,763	N/A

Figure 4: Summary of results for each general technique.

6.2 Technique Comparison

Figure 4 summarizes the results of the general techniques. An IP removal (IP Rem.) is the collapse of two IP addresses to one. An implied pair is a pair of IP addresses that end up in the same group after joining each end of an accepted pair. For comparison purposes, IPid verification was run on the pairs proposed by adjacency splitting. The false pair count is the number of implied pairs that was not discovered by the other three runs. This includes machines not responsive to the other techniques as well as incorrect pairings.

The IPid technique is clearly the most successful technique of the three, finding the largest number of sets and pairs, removing the most IP addresses, and having the smallest number of false positives. However, it required many more packets than the UDP technique.

The rate limit technique required the most number of packets and found the least number of sets. Moreover, it had the largest estimated percentage of false positives. However, it did find pairs that were valid that the others were unable to discover.

The UDP technique scales best, growing linearly with the input size. It is also the easiest to implement and has low false positive rates (after private addresses and common invalid responses are removed).

As the experimental machine was limited to T-1 access, packet rates were around 1,000 packets per second for most experiments in order to avoid flooding the T-1 and causing both experimental error and colleague upset. Faster transmission speeds would reduce the times, quadratically for some techniques. However, rate limits on responses bound the fastest desirable rate of all techniques with the exception of the rate limit technique.

Based on DNS labeling techniques (specifically, doing reverse lookups of the IP addresses and assuming that starting with `fe`n, `pos`n, or `s`n-n, where n is any number, but otherwise the same, belong to the same machine), the IPid techniques with TTL splitting did a good job, eliminating 1,550 out of the 1,765 IP addresses that could be eliminated. IPid with adjacency splitting found 1,537, while rate limit and UDP found a small number (131 and 220 respectively). Given the bias of this DNS labeling technique to certain classes of routers, the rate limit and UDP numbers more likely suggest that this class of machines is not responsive to the techniques, rather than the techniques failed. However, this does demonstrate that the IPid has approximately 88% correctness, at least within this sample set.

Overall, the IPid technique revealed more IP aliasing than both the UDP and rate limit techniques. However, it still had relatively low coverage. The combined results of the two IPid experiments still missed 12,820 of the pairs found by the UDP technique. All three of the techniques used are based on machine behavior that is not specified by RFC or other standards, so using multiple techniques is important in order to maximize coverage.

References

[1] P. Barford, A. Bestavros, J. Byers, and M. Crovella. On the Marginal Utility of Network Topology Measurements. In *ACM SIGCOMM Internet Measurement Workshop 2001*, Nov. 2001.

[2] W. Cheswick, H. Burch, and S. Branigan. Mapping and Visualizing the Internet. In *Proceedings of 2000 USENIX Annual Technical Conference*, June 2000.

[3] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet Map Discovery. In *INFOCOM (3)*, pages 1371–1380, 2000.

[4] V. Jacobson. traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.Z, 1989.

[5] K. Keys. iffinder. http://www.caida.org/tools/measurement/iffinder/, 2000.

[6] C. Labovitz, A. Ahuja, and M. Bailey. Shining Light on Dark Address Spaces. Nanog 23, Nov. 2001.

[7] D. McRobb, K. Claffy, and T. Monk. Skitter: CAIDA's macroscopic Internet topology discovery and tracking tool. http://www.caida.org/tools/skitter/, 1999.

[8] J. Pansiot and D. Grad. On Routes and Multicast Trees in the Internet. *ACM Computer Communication Review*, 28(1):41–50, Jan. 1998.

[9] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.

[10] J. Postel. Internet Control Message Protocol. RFC 792, Sept. 1981.

[11] J. Postel. Internet Protocol. RFC 791, Sept. 1981.

[12] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address ALlocation for Private Internets. RFC 1918, Feb. 1996.

[13] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *ACM SIGCOMM 2002*, Aug. 2002.