Doubletree With Many Sources

Tony McGregor
Computer Science Department
The University of Waikato
Hamilton, New Zealand
tonym@cs.waikato.ac.nz

***Abstract*—Most active measurement projects are limited by the number of measurement points and, consequently, the number of perspectives they have on the Internet. The goal of the RIPE NCC Atlas project is to deploy up to 100,000 active measurement monitors around the Internet. An extended version of Hubble, which finds routing "black holes" is a motivating application. Increasing the number of measurement points by two orders of magnitude requires new measurement approaches. For example, Atlas Hubble needs to perform traceroute type path discovery from many sources to a small number of destinations. It is important to optimise the load placed on the destination monitors, especially if they are located at the edges of the Internet. Doubletree is a path discovery optimisation technique that may be applicable. To date, Doubletree has only been investigated for a small number of sources to many destinations. This paper reports on a simulation study of Doubletree for the many sources, few targets case. Initial results indicate that Doubletree may be very effective in this case but further work is needed to understand the impact of the sharing of coordination information.**

***Keywords*-Doubletree; traceroute; active measurement;**

I. Introduction

This paper explores the effectiveness of the Doubletree [1] algorithm in a Hubble-like [2] application where there are up to 100,000 monitors. Hubble looks for routing "black holes" in the Internet. A black hole is defined as a routing failure that persists for at least 15 minutes where some, but not all, parts of the Internet can not reach a given destination. The system operates in two main modes: target *discovery* mode, where "reachability events" of this type are found, and reachability *analysis* mode, where as much detail as possible about the location of the fault is found. In broad terms, discovery mode operates using reachability measurements while analysis mode uses path discovery.

To utilise a measurement system like Hubble in an environment where there are up to two orders of magnitude more monitors than existing measurement systems requires special attention to the load created by the measurement on the network. This paper looks at the part of the Hubble load that occurs once a reachability event has been found. In this phase, Hubble investigates the event by sending traceroute style probes from all its vantage points to the target of the reachability failure. In the case of Atlas/Hubble, there might be tens of thousands of distinct vantage points (e.g., one per Autonomous System). Running the original traceroute from this large number of sources has the potential to overwhelm the destination, especially if the destination probe is located with a home user or mobile device with limited bandwidth. We investigate whether Doubletree can reduce this load.

Doubletree is designed to reduce the number of probes sent when team probing occurs. It reduces the number of probes sent to discover hops close to the source when multiple destinations are explored from the same monitor by starting to probe with a Time To Live (TTL) > 1. Once the path from this mid-point to the destination has been discovered (by sending successive packets with incrementally greater TTLs), Doubletree sends probes with incrementally smaller TTLs starting with the original mid-point TTL less one. Probing stops when a node that has been discovered before is re-discovered. This aspect of Doubletree is useful for the Atlas/Hubble application but it is the other half of the Doubletree methodology (which reduces the number of probes sent to discover hops near the destination) that, at face value, appears particularly useful to Atlas/Hubble.

Doubletree reduces redundant link discovery near the destination in a similar way to how it reduces probing to hops near to the source except that a global list of hops discovered near destinations is shared by all monitors. In the original Doubletree work, simulations with 24 source monitors showed substantial savings. In Atlas, however, there will many more monitors. As a starting point, we assume one trace from every Autonomous System (AS) that hosts an Atlas monitor giving approximately 22,000 sources in our simulation.

Real-time behaviour influences the effectiveness of Doubletree. For example, if several probes reach a link at about the same time the second and subsequent monitors are unlikely to have already received the stop-set information indicating that this link is already known.

The rest of the paper is organised as follows: Section II describes the simulation of Doubletree, including the main assumptions and simplifications. Section III presents the main results of the investigation. Information required to reproduce this work is provided in Section IV. The paper ends with a review of the current results and further work needed to fully understand the use of Doubletree in this context.

II. Simulation

This Section describes the simulation of Doubletree undertaken for this project. It includes the derivation of the input data used and the major assumptions made.

A. Topology

To be realistic, the simulations need a topology that matches the Internet as well as possible. In particular, the length of paths and the branching nature (node in-degree and out-degree) of the topology should be as similar to the real Internet as possible. To this end, the simulator topology used is based on a map of the Internet derived from CAIDA's Archipelago [3] infrastructure running the Scamper [4] tool (from here on referred to as Scamper for simplicity although we note that Scamper can be run in other environments).

Scamper attempts to measure the global topology of the Internet. Traceroute style measurements are taken from a relatively small number of sources to many destinations. Scamper uses the intra-monitor part of Doubletree to reduce the cost of probing hops near a monitor but does not use the inter-monitor (global stop-set) part of Doubletree. The output from Scamper is a set of runs where each run contains a traceroute style probe to every destination address from one member of the team of monitors. A single run started on 3 Jan 2009 from a team of 13 monitors was used as the topology input to the simulator. In future work, we plan to repeat the simulations with other scamper data sets to determine the stability of our results against the particular scamper data set used.

The arrangement of nodes and links alone is not enough to route packets through a network; routing information is also required. In the simulator, routing information is represented in a table of destinations and next-hop at each node. This information is inherent in the scamper data set and we simply maintain it in the simulator topology data structure.

B. Interfaces vs Routers

Scamper, like all traceroute based tools, discovers interfaces not routers; the raw data does not show which interfaces are on the same router. The simulator topology is, therefore, also built in terms of interfaces not routers. This is not problematic for the simulation, which also proceeds in terms of packets being passed from interface to interface. When we refer to nodes in the topology model we are referring to a particular interface (not a router). Similarly, links are between interfaces.

C. Discarded paths

Some of the paths in the scamper data are not usable in the simulator, mostly because they are not well formed. For example, scamper discovers loops in some paths. In others, it abandons tracing because too many nodes do not reply with a TTL expired message. In these cases, a complete path from the source to the destination is not discovered and the path can not be used in the simulation. Of the 5.6 million paths discovered in part or full by scamper, 241,763 (4%) are omitted from the simulation topology because of these reasons.

D. Symmetric Paths

Scamper data is not a complete map of the Internet. In particular, it contains paths from the monitors to the destinations but not the reverse. It also does not measure paths between destinations. The first issue is resolved in the simulator by adding a symmetric path from the destination back to the source. From other work [5], we know that Internet paths are not always symmetric. We do not believe that the symmetric nature of paths in the simulation topology significantly affects our results because it is the overall structure of the Internet (i.e., path lengths and branching) not the exact details of particular paths that is important. However, without a measured non-symmetric topology, we have no way of demonstrating that this is the case.

The omission of paths between destinations is mostly not problematic for the simulator because packets are only sent between sources and destinations.

E. Alternative Paths

In some cases, Scamper discovers alternative paths between nodes within the network. In the Internet, alternative paths may arise because of load balancing or because the topology has changed during the measurement. As a consequence, Scamper may discover different alternatives between the same nodes within the network when it probes between different source/destination pairs.

In the simulator topology model, alternative paths between nodes are maintained. The same path variant is used when packets are sent from a source to a destination as was discovered when scamper measured the route between the two. In the simulator topology data structure, this is done by including a source as well as the destination in the next-hop table at a node.

This approach does not necessarily match the behaviour of the Internet in all cases, however it is likely to be correct in most cases. If the source of the alternative paths is a path change during scampers probing, either path is acceptable for the simulation. It is not required that we maintain both paths in this case but it is acceptable. In the case where there are alternative paths due to load balancing, using the same path as the one scamper discovered for this source/destination pair will mostly match the behaviour of the Internet. This is because per-destination and per-flow load balances are most common in the Internet while per-packet load balances are rare [6].

F. Missing hops

During traceroute style probing, it is common for some hops to not reply with a TTL expired message. Often the

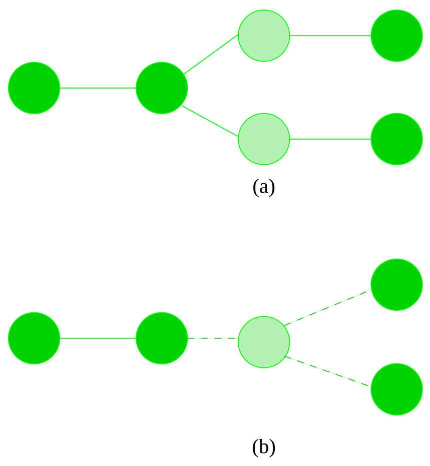


Figure 1. Modelling Missing Hops

hop is known to exist because later hops do respond but the address of the hop can not be discovered. Approximately 22% of hops are not identified in the traces we used. Within the simulator, these non-responding nodes are given a unique address. The addresses are constructed so that they do not collide with the IP addresses that most nodes have. To support this, the simulator has two classes of address, an IP address (printed as a doted quad) and a "missing" address (printed as an "m" followed by a unique id).

This procedure does not exactly replicate the structure of the Internet at the time scamper was probing because it is possible that a missing hop in two different paths might be the same interface but this approach inserts two different interfaces (see Figure 1). From the Scamper data, it is not possible to tell which scenario is correct.

G. *Missing Topology*

Although it is the most extensive macroscopic Internet topology discovery system available, Scamper does not discover all links in the Internet. The extent of missing topology is not currently known. What Scamper discovers is the path used from the vantage points it has at the time the measurement was taken. Because the simulation is based on edges measured by Scamper, the simulation matches how packets were routed at that time. If the paths Scamper finds are typical of all Internet paths, the missing topology has no significant impact on the simulations.

H. *Selection of Endpoints*

The simulation uses a sub-set of the possible end-points in the topology. The selection is based, in part, on the AS (Autonomous System) that an endpoint belongs to. The translation from IP address to AS number (ASN) is based on data collected by the RIPE NCC Routing Information Service (RIS) [7]. RIS data contains route dumps from specially deployed routers that passively peer with operation routers in many ISPs. The routing data collected describes Internet routing as a set of elements with an IP address range and a list of ASs that traffic may be router through to reach IP addresses in the range. For more details see [7]. A database of address ranges and the last ASN in the path was created from RIS data from 30 March 2009 to allow IP addresses to be matched with a host AS. On occasion, there is more than one last hop ASN for a given IP address. In this case, the first ASN discovered in the data set is used.

For simulations of many sources to a few destinations, the Scamper source monitor addresses are used for the *destination* addresses. One source address present in the Scamper data is selected from each AS for the destination addresses. If there is more than one source in the same AS, the first source discovered is used.

As explained in Section III (Results), some simulations were performed from a few sources to many destinations. In this case, the selection process is the same but with the source and destination roles reversed.

This methodology produces a topology with 4.2 million nodes (interfaces including sources and destinations) and 55 million links between nodes.

I. *Probing Strategy*

Several different strategies for when to start sending probes to a particular destination are modelled. The simplest is "1-stage" in which, all probing to all destinations starts at the beginning of the simulation. At the other extreme is "staggered" probing where, the next destination is not started until the previous one is complete and each monitor waits until the previous monitor is complete. Between there extremes we modelled "2-stage" and "10-stage" where the first two (or ten) destinations are probed sequentially (as in "staggered" probing) then the remaining probes are sent when they are complete.

The different probing strategies explore whether slower initial probing enhances the performance of Doubletree by allowing probes started later to benefit from more information from the early probes. In particular, the impact of probing on the last hops is particularly important because that is where there is the most potential for congestion from the traffic created by the large number of monitors. If the early probes learn the final hops, most later probes would not re-probe them. Staggered probing is not suitable for deployment because it will take too long to complete; it is simulated to provide a best case against which the performance of the other strategies can be compared.

Unlike the traditional traceroute, Doubletree does not send multiple probes per TTL value. For comparison the traceroute model in the simulator also sends one probe packet per TTL value.

III. Results

The following sections describe the results of simulation of Doubletree, first for the few sources case and then for many sources.

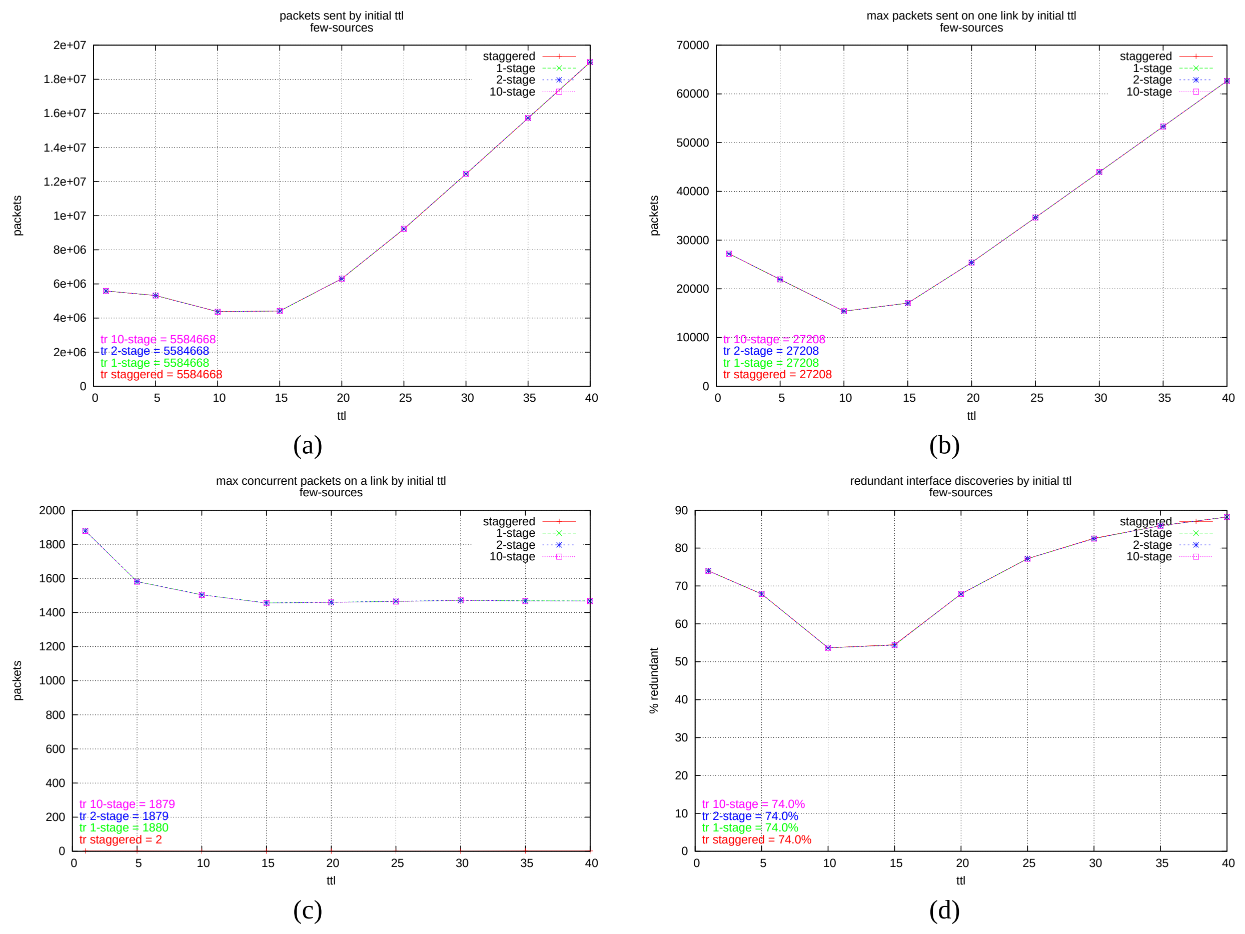


Figure 2. Few Sources, many destinations

A. Few monitors to many destinations

Although our motivation, the Atlas/Hubble application, leads us to be most interested in the performance of Doubletree with many monitors to a few destinations, we present the reverse arrangement so that our results can be compared with previous Doubletree simulations and because we explore more parameters than previous studies have.

Figure 2(a) shows the effect, on the total number of packets sent, of varying the starting TTL that Doubletree probes from. At low initial TTLs, the local stop set is less effective. At high starting TTLs, extra probes are sent for paths that are shorter than the starting TTL. The trade off between these factors, in this topology, suggests a starting TTL of between 10 and 15. There are no significant differences between the different probing strategies (staggered, 1-stage etc) and, as a consequence, the lines are indistinguishable on the graphs. The equivalent values for traceroute, which always starts probing from TTL 1, are shown in the bottom left of the graph.

The number of packets sent by Doubletree with an initial TTL of 15, is approximately 4.4 million compared with 5.5 million for traceroute. A similar trade off can be seen with the maximum number of packets sent on a single link and the highest number of concurrent packets on a link (see Figure 2(b) and (c)).

Even with Doubletree, there must be some redundant discoveries. These occur because Doubletree probing cannot stop in either direction until an interface that has already been seen is re-discovered or the source or destination is reached. When the initial TTL is too small for effective operation of the local stop set, or too big for the global stop set, further redundancies occur. Concurrent discoveries may also cause redundant probes. Figure 2(d) shows the percentage of interface discoveries that are redundant. Again, initial TTL around 10-15 is most effective with about half (55% for TTL=10) of the interface discoveries being redundant compared to 74% for traceroute.

B. Many monitors to a few destinations

The many sources to a few destinations case is shown in figures 3 and 4. In this case, the local stop set is lately ineffective and most optimisation comes from the global stop set. The cost of exchanging the global stop-set is not

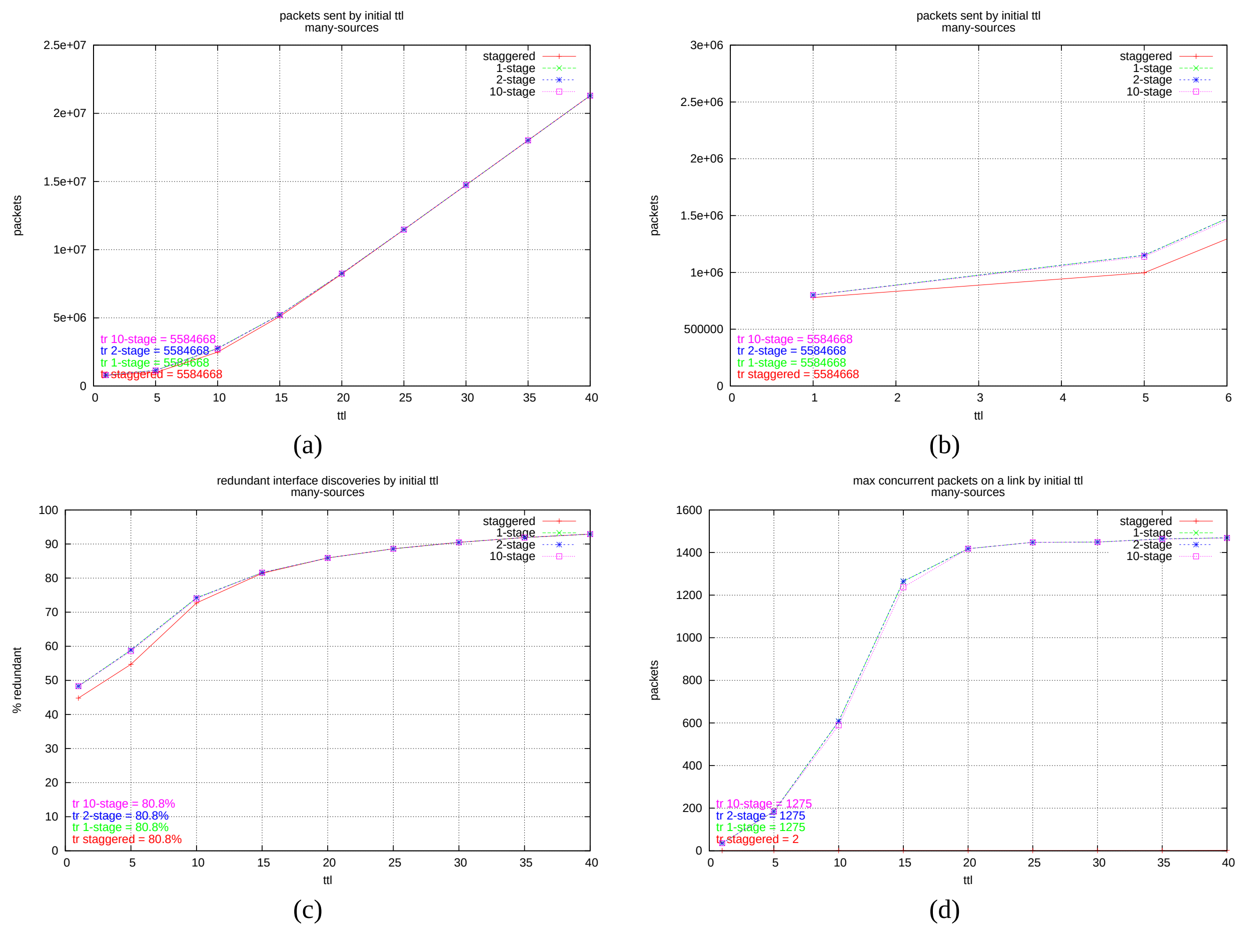


Figure 3. Many Sources, few destinations

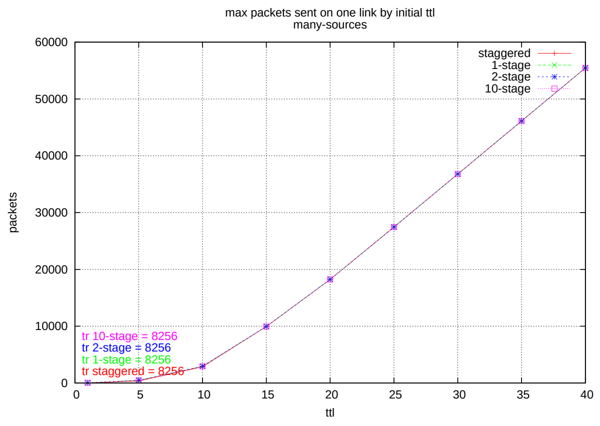
included in these results.

Figure 3(a) shows the effect of varying the initial TTL on the number of packets sent. Because of the predominant effect of the global stop set, which is most effective at smaller starting TTLs, there is no trade off between the effectiveness of the local and global stop sets. The number of packets increases as the starting TTL increases. Figure 3(b) shows an enlarged view of the area near the origin. At an initial TTL of 1, the number of packets sent using Doubletree is approximately 770,000 compared with 5,600,000 for traceroute. Note, however, that the effect of communicating the global stop set will add additional packets to the Doubletree case.

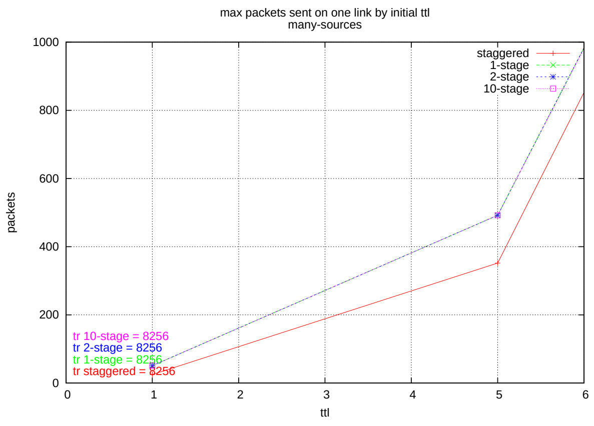
Figure 3(c) shows the percentage of redundant discoveries for the many sources case. Unlike the few sources case (Figure 2(d)), there is no trade off. At initial TTL of 1, there are less than 50% redundant discoveries (45% for staggered start times and 48% for the other probe starting strategies). Traceroute has 81% redundant discoveries for the many sources case.

The number of packets sent, especially the number of packets sent on the links close to the sources and destinations, is of particular interest in the Atlas/Hubble scenario that motivated this study. The total number of packets concurrently sent on a link is shown in Figure 3(d). For an initial TTL of 1, the largest number of concurrent packets on a link is 2 for staggered starts and 36 for the other probing strategies. The former is inherent in the staggered strategy, which does not send a new probe until the previous one has finished. The equivalent values for traceroute are 2 and 1275 respectively.

The maximum number of packets carried by any one (unidirectional) link during the course of the simulation is shown in Figure 4(a). An enlarged view of the origin in shown in Figure 4(b). For an initial TTL of 1, there are only 25 packets sent on the most most loaded link using staggered traces and 51 for the other cases. Classical traceroute (but with one probe per TTL, not 3) has a maximum load on a single link of approximately 8300. This is a very encouraging result but, again, we note the omission of the cost of exchanging the global stop set.



(a)



(b)

Figure 4. Maximum packets sent on a Link (many Sources, few destinations)

IV. Reproducibility

Reproducing the results described here requires the simulator code, the scamper and RIS data sets and some associated tools. The simulator code can be obtained from [8]. The simulator is written in C with some preprocessing of the topology files written in Perl. gcc 4.3.2 and Perl 5.10.0 were used to produce the results for this paper.

The scamper data set is not allowed to be redistributed but can be obtained from CAIDA via the request form at [9]. As previously noted, the data set used for this study is the Scamper run from 3 Jan 2009. The data set consists of 13 files with names in the pattern `daily.l7.t1.c000359.20090103.`mmm-cc`.warts.gz`. mmm-cc should be replaced with the monitor identification codes: `amw-us` `bcn-es` `cjj-kr` `dub-ie` `hel-fi` `hlz-nz` `laf-us` `lej-de` `mnl-ph` `nrt-jp` `san-us` `syd-au` `yto-ca`.

The files are decoded using the `sc_analysis_dump` tool from the scamper package. For this work version `scamper-cvs-20070523o` was used. Scamper [4] can be obtained from the WAND website [10]. The version used for this work is preserved along with the simulator code at the address given above.

For the IP address to ASN translation, RIS data from 30 March 2009 was used. RIS data may be fetched from the RIPE NCC at [11]. The RIS data that was used for this work is also preserved with the simulator code at the address given above.

V. Conclusions

If the cost of exchanging the global stop set is excluded, Doubletree is expected to be very effective at reducing the load on the monitors in an Atlas/Hubble system. Simulations suggest the load on the most heavily utilised link reduces from 8300 packets to just 51 packets at $TTL = 1$.

These simulations do not model the effect of sharing the global stop set. Sharing the global stop-set will increase the total number of packets sent (perhaps very significantly). However, this extra load will be most concentrated around the nodes that coordinate the exchange of global stop set data. It is possible that, an Atlas/Hubble system could be built with more capacity in this/these area(s). It is not yet clear to what extent communication of the global stop set reduces the Doubletree gains at interfaces near the destination probe.

A second outcome of this work is the demonstration that Internet scale simulation is now possible on commonly available hardware. In this study, available memory and single core execution speed where the limiting factors. The simulator was custom written for this application. Significant effort was made to optimise the implementation in both memory usage and execution time. A simulation run (representing a single point on the graphs) used around 8GB of RAM and took between one hour and three days depending on the number of packets sent. As noted, Scamper does not measure all of the Internet. However, commodity hardware with more cores and much larger memory are now common and within the budget of most academic computer science departments.

Acknowledgements

This work was undertaken while the author was on sabbatical with the RIPE NCC. My thanks for their support both practical and academic during this time. I am grateful to the RIPE NCC and CAIDA for making the data used in this work available. Support for the CAIDA IPv4 Routed /24 Topology Dataset is provided by the National Science Foundation, the US Department of Homeland Security, the WIDE Project, Cisco Systems, and CAIDA Members.

References

[1] B. Donnet, B. Huffaker, T. Friedman, and K. Claffy, *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4268, ch. Evaluation of a Large-Scale Topology Discovery Algorithm, pp. 193–204. [Online]. Available: http://dx.doi.org/10.1007/11908852_17

[2] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson, "Studying black holes in the internet with hubble," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 247–262. [Online]. Available: http://portal.acm.org/citation.cfm?id=1387589.1387607

[3] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: From art to science," *Conference For Homeland Security, Cybersecurity Applications & Technology*, pp. 205–211, 2009.

[4] M. Luckie, "Scamper: a scalable and extensible packet probet for active measurement of the internet," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 293–245.

[5] V. Paxson, "End-to-end routing behavior in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 41–56, October 2006. [Online]. Available: http://doi.acm.org/10.1145/1163593.1163602

[6] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with paris traceroute," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, ser. IMC '06. New York, NY, USA: ACM, 2006, pp. 153–158. [Online]. Available: http://doi.acm.org/10.1145/1177080.1177100

[7] RIPE NCC. (2010) RIS:Routing Information Service. [Online]. Available: http://www.ripe.net/projects/ris/

[8] T. McGregor. Dobuletree witn many sources: reproducability data. *Last Accessed 10 Jan 2011*. [Online]. Available: http://byerley.cs.waikato.ac.nz/~tonym/papers/icimp11/reproduction/

[9] CAIDA: Coperative association for Interent Data Analysis. Topology data request form. *Last Accessed 10 Jan 2011*. [Online]. Available: http://www.caida.org/data/active/topology_request.xml

[10] Waiakto WAND Network Research Group. Web site. *Last Accessed 10 Jan 2011*. [Online]. Available: http://wand.net.nz/

[11] R. N. C. Centre. Routing information service website. *Last Accessed 10 Jan 2011*. [Online]. Available: http://ripe.net/ris/