Towards Internet Scale Simulation

Anthony J McGregor
Computer Science Department, The University of Waikato
Hamilton, New Zealand
Email: tonym@cs.waikato.ac.nz

***Abstract*—Simulation of the Internet has long been understood to be very challenging mostly because of its scale, diversity and the lack of detailed knowledge of many of its components. However, two recent developments (macroscopic topology discovery and large memory servers) mean that some of these problems are now more tractable. Although problems like the lack of detailed link information remain, models are are useful for some problems that require an understanding of how an application interacts with the Internet as a whole. The paper presents is-0, an Internet Simulator. is-0 derives its model of Internet topology directly from the output of an Internet topology mapping project. Efficient design allows is-0 to simulate packet-by-packet, hop-by-hop behaviour at Internet scale. Validation of is-0, an example application and performance measurements are included.**

***Keywords*-Discrete Event Simulation, Internet Simulation.**

I. INTRODUCTION

Understanding the performance of the Internet and the way that new applications and protocols will perform and interact is a challenging problem that has been noted by many authors. For example, in 1997, Paxon and Floyd wrote:

> "As the research community begins to address questions of scale, however, the utility of small, simple simulation scenarios is reduced, and it becomes more critical for researchers to address questions of topology, trafc generation, and multiple layers of protocols." [1]

The main challenges they noted were: heterogeneity in nodes, links and protocols; rapid rate of change including size, applications, protocols and traffic characteristics; large size; and the difficulties of building traffic models [1], [2]. These problems are "moving targets". Not only is the Internet big and diverse, it is growing rapidly in both respects. The challenges of Internet simulation, especially the scale of the Internet, has been reiterated by many others including [3] [4] [5] [6] [7].

Two developments have made it possible to make progress towards simulation of the Internet as a whole. These are: Internet macroscopic topology discovery projects and a large increase in the memory capacity of commodity servers. In recent years, there have been several projects that are producing useful maps of the Internet. These include CAIDA's ARC infrastructure [8] running the Scamper mapping tool [9] and Dimes [10]. Further progress on mapping the macroscopic topology of the Internet can be expected in the coming years with projects like the RIPE NCC Atlas [11]. The entry of large memory servers to the commodity market is driven by the trend towards vitalisation. Computers with up to 512GB of RAM are now available at less than US$40,000.

Together these two developments mean, it is now possible to build a packet, node and link level simulation model for the Internet as a whole that will run on commodity server hardware. There are still many challenges that prevent high fidelity simulation of the Internet including lack of knowledge of the characteristics of each link and the cross traffic links carry. However, it is possible to make models that are useful for some problems including those where the topology of the Internet interacts with a system in complex ways but where fine grained temporal results are less important. Examples include content distribution, reducing peer-to-peer traffic loads and multicast optimisation.

In this paper, we describe the use of simulation to investigate the traffic load created by large scale use of the DoubleTree optimisation [12] for topology discovery. This was motivated by the desire to implement a Hubble [13] like application on Atlas [11]. We present a new, open source, simulation system, **is-0**. The system includes a discrete event simulator and surrounding infrastructure to support the process of converting the output of an Internet mapping project to a topology model suitable for simulation, running a set of simulations across a range of parameters values utilising the massively parallel nature of most simulation experiments and presenting outputs.

There are other open source, discrete event simulators. Some, like ns-2 [14] are well established and we might have based this project on one of them. However, the core of a discrete event simulator is simple and most of the contribution of this project lies in managing scale and in supporting the whole process of taking an Internet topology map through to the result of simulation experiment, possibly involving many individual simulations (see Figure 1). **is-0** must meet the needs of a large topology and, potentially, billions of packet events. On the other hand, it provides less fine grained temporal behaviour than many other simulation projects. The need to optimise performance in terms of both memory and CPU cycles leads to the requirement for an implementation tailored to meeting these needs in the

 ISBN: 978-1-61208-169-4

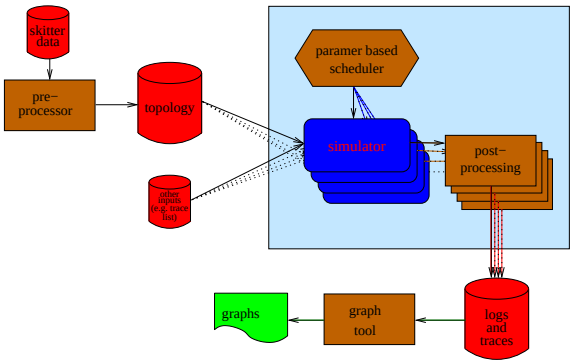

Figure 1. **is-0** Architecture

context of an Internet model.

There is a great deal more planned for **is-0**. We are pre-releasing it now because it has already demonstrated its usefulness to us in an important research area. We believed that, other researchers will find **is-0** useful. The rest of this paper describes the system in more detail. An example application and performance statistics are presented.

II. **is-0** OVERVIEW

A. Topology

The current version of **is-0** builds its topology from Scamper [9] output files. Scamper measures the global macroscopic Internet topology from a set of measurement points to one destination in every routed IPv4 `/24`. The output from Scamper is a set of runs where each run contains a traceroute style probe to every destination address from one monitor (team probing).

From this data, **is-0** builds a topology model of the Internet based on nodes and links between them. The nodes and links alone are not enough to route packets; routing information is also required. In the simulator, routing information at each node represented as a table of destinations and the associated next-hop. This information is inherent in the Scamper data set and it is simply maintain in the simulator topology. For conciseness, we refer to the data structure as the "topology model" even though it contains elements of topology and routing. The following sections refine the topology model as it is built from Scamper data.

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 normally problematic because simulations are performed in terms of packets being passed from interface to interface. References in this paper to nodes in the topology model are 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 simulation. The Scamper data set used for the example in section example had 5.6 million paths discovered in part or full by Scamper. Of these, 241,763 (4%) were omitted because of the reasons described above.

D. Alternative Paths

In some cases, Scamper discovers alternative paths between nodes. Alternative paths may arise because of load balancing or because the topology has changed during the measurement. Scamper may discover different paths between the same nodes when it probes between different source/destination pairs. These alternatives are maintained in the topology model. 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 topology data structure, this is done by including a source as well as the destination in the next-hop table.

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 more common than per-packet load balances in the Internet [15].

E. Unknown Paths

Scamper data does not provide a complete map of the Internet. While it contains paths from the monitors to most destinations it does not have the reverse paths or paths between destinations. The extent of this missing topology is not currently known.

The lack of return paths is resolved in the simulator by adding a symmetric path from the destination back to the source. It is known that Internet paths are not always symmetric [16] . For many simulations, it is the overall structure of the Internet (i.e., path lengths and branching) not the exact details of particular paths that is important. If this is the case, the symmetric nature of paths will not unduly influence simulation results. However, without a measured non-symmetric topology, there is no way of demonstrating that this is true for a particular experiment.

 ISBN: 978-1-61208-169-4

The omission of paths between destinations is not problematic for simulations where packets are only sent between sources and destinations (as is mostly the case in our example). If this is not the case, the simulator has the ability to add extra paths to the topology. These paths are, by necessity, not based on measured topology. Extra paths are discovered using a breadth first search from the source to the destination using links that were discovered by scamper. If paths from single source to multiple destinations are required, a single search can create all the paths.

Added paths do not necessarily follow the same route as in the Internet. The breadth first search finds the shortest path based on the hops that were discovered by Scamper. While Internet routing is designed minimise path length this is in terms of the ASs that a path passes through. This may use a link Scamper did not discover or, within an AS, the shortest path may not be followed. Internet routing may also includes policy which restricts the choices for a path.

An understanding of the extent to which this affects the results of a particular simulation may be gained from a comparison of the performance of Scamper measured paths and the equivalent paths formed by the methodology above. A future release of **is-0** will include automated sensitivity analysis including the effect of added paths (although cross traffic sensitivity is the highest priority for automated sensitivity analysis).

F. Missing hops

During traceroute style probing, it is common for some hops to not reply with a TTL expired message. Often the hop is known to exist, because later hops do respond, but the address of the hop is unknown. In the data set used for the example in section VII, approximately 22% of hops are not identified. Within the simulator, these non-responding nodes are given a unique address.

This procedure may not exactly replicate the structure of the Internet at the time Scamper was probing. It is possible that, a missing hop in two different paths might be the same interface, however, this approach inserts two different interfaces. Automated sensitivity analysis could also allow the impact of this effect to be determined.

G. Topology Data Structure

Within the simulator, the topology is represented in a data structure based on nodes and links. Links contain a reference to the node at each end of the link, the link latency, serialisation rate, and the current link state (queue length and when the current packet, if there is one, will have been completely added to the the link). Links also contain performance metrics including packets dropped, packets sent by packet type and the peak queue length.

Nodes include their address (IP address or missing node address) and a table of references to links that leave this node. The table is indexed by either the destination (of the path, not the next hop) or, where there is more than one path to a destination (see section II-D), the source (of the path) and the destination.

Type	**Hooks**
Packet Events	`newPacketHook`, `packetQueuedHook`, `packetArrivedHook`, `packetDropHook`, `ttlExpiredHook`, `changePacketTypeHook`
Simulation Start and Termination	`startHook`, `usageHook`, `argsHook`, `logConstantsHook`, `cleanupHook`, `heapMapValidHook`, `buildHook`, `newNodeHook`, `saveBuildGlobalsHook`, `restoreBuildGlobalsHook`
Hash Management	`newHashEntryHook`, `freeHashElementHook`
Reporting	`progressHeaderHook`, `progressHook`, `printStatsHook`, `packetInfoHook`, `summaryStatsHook`, `specialAddrHook`, `nodeSummaryHook`

Figure 2. API Event Hooks

H. Building The Data Structure

The raw Scamper data is pre-processed into a record for each node that contains the next hop links from that node. The resulting files are large. For the example application described in section VII, they total a little over 2GB. Before a particular simulation can be run, this information must be built into the internal simulator data structures including the hash tables, initialisation of performance metrics etc. Any additional paths must also be added to the topology. The resulting data structure is large (in the order of 8GB for the example application) and it takes several minutes to load and build. If many simulations are to be run (around 8,000 were needed for our example experiment) the total time taken to repeatedly load and build the data structure is significant and the size of the data structure may limit the number of simultaneous simulations that can be run on a machine.

is-0 can store the topology data structure in a memory mapped file. This has two advantages. Firstly, the data structure can be reused, avoiding most of the time otherwise required for building it. Secondly, memory mapped files can be shared and only a single copy kept for their read only components. This may allow more simulations to be run in parallel reducing the time required for large topologies on machines with many cores.

 ISBN: 978-1-61208-169-4

III. Parameter Exploration

Scripts that manage the process of concurrently running as many parallel simulations as the hardware supports with different parameters are included with the simulator. These scripts manage the process of running simulations, recording the results in appropriately named files, first cut checking that simulations complete successfully, re-starting batches after an interruption and logging and reporting overall progress.

Scripts that produce plots over different combinations of parameters are also included. Users can select the parameter for the x- and y-axes and also an additional parameter (and perhaps some specific values of this parameter) if more than one line is to be drawn per plot.[1] While much of this is mundane, in a typical simulation experiment considerable researcher time is spent on these mundane matters and **is-0** can significantly reduce this effort. A future release will include more sophisticated parameter space exploration inspired by Nimrod [17] and other similar tools. This will reduce the number of simulations that need to be run as part of an experiment by focusing attention on parts of the parameter space that cause significant variations.

IV. API

The simulator API has three components: data structure augmentation, event hooks and utility routines. These are described in the following sections.

A. Data Structure Augmentation

Application related data structures (like packets and nodes) can be extended. Our example application implements a traceroute like protocol so we have added a probe packet type that contains, amongst other things, the value that the TTL field had when the probe ended its outward journey and began to return to its source. The data structures that can be augmented in this way are: packets, which can have new packet types and/or additional generic fields in all packets; nodes; hash tables; and the set of global variables that is saved when a memory image is created and then restored when the image is reloaded for a particular simulation run (see section II-H above).

B. Event Hooks and Calls

The second component of the API is a set of event routines that can be called when simulation events occur that might be important to an application. For example, the example application uses `ttlExpiredHook`. There are currently 25 hooks as shown in Figure 2.

The API also includes 51 function calls (and related constants, macros and data structures) as shown in Figure 3.

[1]Currently, these scripts have not been fully generalised and are somewhat tailored to our example problem. However, changes required for other applications are not expected to be great.

Type	Calls
Sim. Infrastructure	`intArg boolArg usage queueEvent warpTime simMalloc simFree`
Hash	`makeHash freeHash makeSpaceInHash hashSize changeHashSize find dumpHashTable forEachHashEntry`
Address	`addr2str str2addr extractAddr addrEqual addrCpy setAddr2null`
Topology	`addLink queueLength findDistances makeDistancesFile processDistances addExtraPath`
Packet	`disposeOfPacket changePacketType makePacket packetInfo queuePacket4nextHop queuePacket addPacket2link swapAddrs`
Utility and progress reporting	`chopNl commas comment flag2bool processEachLine add2file lastModified fileSize skipUnknownTags getTaggedLine registerProgressReport unRegisterProgressReport recordTraceEvent`
For parameters	`newExploreNode freeExploreList packetArrivedEvent`

Figure 3. API Calls

V. Validation

The goal of the project is to simulate the Internet as a whole. As a consequence, it is not possible to compare the results of simulation with the real system. However, four other types of validation have been performed. These are: internal consistency, manual validation against a simple network scenario, external generic behaviour validation and external application specific validation.

A. Internal Validation

The simulator contains a substantial amount of internal consistency checking. There is liberal use of `asset` statements, particularly for pre-and post conditions (currently, there are > 400 asset statements in the C code base of 12,500 lines). The C code also contains a hierarchy of more extensive tests. These include, for example, checks on the consistency of data structures, sensibility checks on behaviours (e.g. that packets leave a FIFO queue in the order they entered) and event queue checks. There are none levels of the hierarchy and approximately 120 blocks of checking code. The simulator is also run under `valgrind` to ensure there are no uninitialised variables, accesses to freed memory or memory leaks.

 ISBN: 978-1-61208-169-4

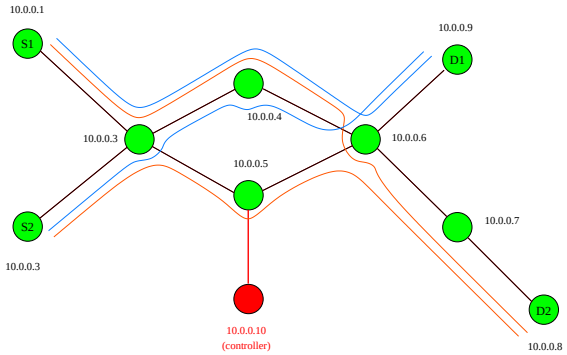

Figure 4. Network used for Hand Validation

B. Hand Validation

A simple network (see Figure 4) and test scenario was designed and the example application simulated. This involved nine nodes, nine links, including a diamond topology and alternative load balanced paths, and two traces. The simulator was configured to record a full trace of all events for each link and packet. These traces were then analysed by hand to check for consistency, errors and correct path discovery.

C. External Validation

Simulator trace files can also be checked for correct behaviour using a separate programme once a simulation is complete. Separate tests check that all packets are delivered exactly once, that they are delivered in sequence and that the transmission and queueing time is consistent with the links latency and serialisation rate.

The external validator was written by the same programmer who coded the main simulator but several months after the original coding and in a different language (perl). It is much slower and uses a lot more memory than the simulator so it is not suitable for use with every run of the simulator, rather it is normally used to check a representative set of results. It was used to check the simulations that were hand validated.

In addition to generic validation, which only relies on features of the base simulator, external validation was also applied to the example application. This involved running traceroutes over a network configuration and then running a program that checked that the paths discovered were correct. This program was coded in a similar way to the generic external validation.

VI. Performance

Achieving the required performance, including balancing memory use and CPU cycles, required careful implementation. This was informed by extensive use of the kcachegrind/valgrind tool set. The following sections describe two optimisations in the design (but there are may others). Section VII-A contains further performance statistics in the context of the example application.

The simulator uses five different types of hash table to improve performance. There are millions of hash tables in use in any particular simulation run. For example, any node can be found from its address via a hash table and the table of next hops within a node has an associated hash table.

A common hash mechanism is used across all hashes. It supports look up from one or two keys (e.g. the destination or source/destination pair), insertion and deletion, chaining through all valid entries, resizing of the hash table, and hash performance statistics. Collision resolution is managed via an alternate hash calculation and, if that also collides, by linear chaining. Use of indirection minimises the memory use of hash tables which normally have many unused entries.

The hash infrastructure includes (optional) performance metrics and automatic hash resizing to maintain sufficient head space for efficient operation. Resizing is relatively expensive (especially for large hashes) so it is avoided where possible. The topology pre-processing produces size hints that remove the need for most resizing.

A. Event Queue

The simulator event queue is optimised for Internet simulation. In particular, most events are added for times in the near future and there are often many events at the same time. A fixed size, event hint look-aside table allows the correct queue location for most events to be found with a single table look up.

VII. Example

The initial motivation for development of **is-0** was to investigate the potential for DoubleTree [12] to reduce the cost of measuring the path from many sources to a few destinations. This problem stems from the desire to design an implementation of an application like Hubble [13] on an infrastructure consisting of up to 100,000 vantage points (a design goal for the RIPE Atlas [11] project). In previous DoubleTree work, the few sources, many destinations scenario was investigated. [12]

The code for this application is included with the simulator as an example. It is divided into two parts, traceroute and extensions to traceroute for DoubleTree. Implementing traceroute took 950 lines of code including comments, checking and reporting code.. DoubleTree required an additional 1,300 lines of code.

Donnet built a simulator to explore DoubleTree's behaviour but was not totally happy with the level of detail that it provided. [18]. We believe that, had **is-0** been available at that time, he would have had access to detailed modelling with less effort. In turn, this may have allowed more development of DoubleTree for the same effort.

 ISBN: 978-1-61208-169-4

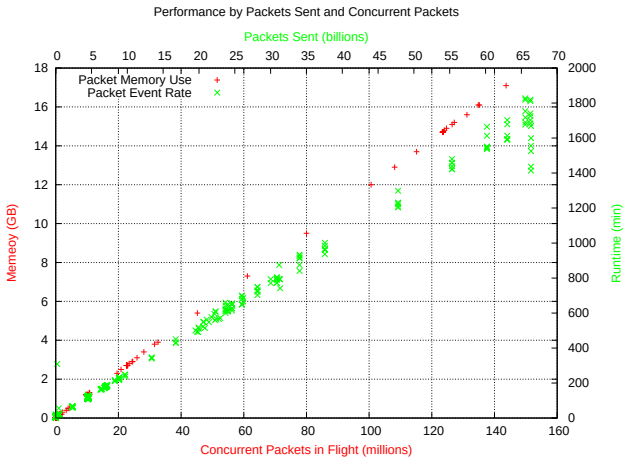

Figure 5. Resources by Number of Packets Sent and Concurrent Packets

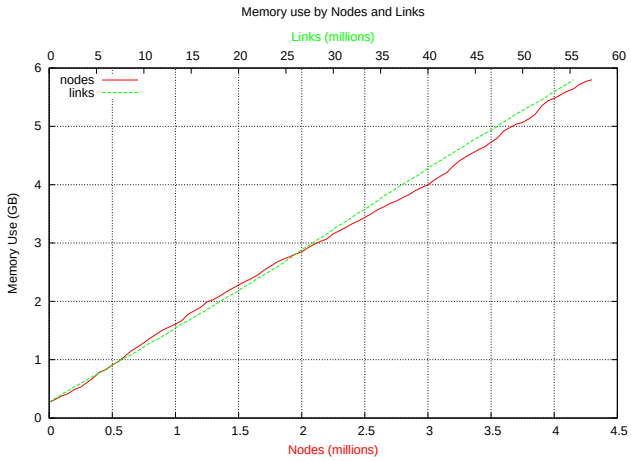

Figure 6. Memory Use Against Nodes and Links in the Topology

A. *Performance*

The Internet model in this example used a Scamper run from 3 Jan 2009. The typology had 4,300,000 nodes and 55,000,000 links (source/destination based next hops). 22,000 traceroutes were performed in each simulation. A total of 1835 simulations were run for each investigation with different parameters (e.g. starting TTL, trace schedule, stop set exchange).

The resources used per simulation varied as the simulation parameters were changed. A typical example `(DoubleTree=yes, sources=many, scheduling=1stage, probing=team, ttl=1 stopSetBin=250ms)` required simulation of sending 25,000,000 packets, had a peak and mean memory usage of 8GB and ran in 945s real time. A small number of (pathological) parameter combinations required far greater resources. For example, `(DoubleTree=yes, sources=many, scheduling=1stage, probing=team, ttl=1 stopSetBin=1ms)` required simulation of sending 2.3 billion packets, peak and mean memory usage of 15GB and 10GB (respectively) and ran for 4,103s. At the time of peak memory usage, there were 123 million packets in flight.

It is likely that, over time, growth of the Internet and more extensive topology discovery will result in larger topology models. **is-0** was designed to permit multiple CPU cores to be used in parallel on a single simulation. While this is not yet fully implemented, planning for it is well underway and it will be one of the first extensions implemented in future release.

Figure 5 shows the relationship between the number of events simulated and the memory use and real time taken for the example application. The graph demonstrated event rates of around 650,000 events/s on our hardware[2] Figure 6 shows the relationship between the number of nodes and links in the topology and the total simulator memory required for a null simulation. This data was collected by building the data structure for the first n nodes in the topology.

More details of this simulation and its results are available in [19].

VIII. Conclusion

is-0 only addresses a few of the challenges of simulation of the Internet as a whole. However, it has proven useful and we believe others will find it helpful in exploring problems that interact with the Internet as a whole. **is-0** supports simulations with millions of nodes and billions of packets on commodity hardware. It builds its topology model directly from the Scamper Internet macroscopic topology discovery project data. It also includes parameter exploration and graphing tools to reduce the time required to undertake Internet simulation experiments.

Some of the plans for extending **is-0** have already been mentioned. These include: using other sources of topology data (e.g. Dimes [10]); Nimrod [17] style parameter space exploration; and automated sensitivity analysis.

Currently, parallel use of **is-0** relies on the embarrassingly parallel nature of most simulation experiments by running multiple simulations concurrently. Support exists to reduce the memory overhead in this case. However, some simulations are long and this is expected to be more common as the Internet continues to grow and better models of the Internet become available. Support for employing multiple cores within a single simulation has been designed and will be included in a future release. The code for **is-0** is available from `http://research.wand.net.nz/software/`.

[2]Intel Xeon X5570, 2.93GHz, 8192KB cache, 800Mhz DDR3 triple channel memory, 12 concurrent simulations over 8 physical cores with two hyper-threads each.

 ISBN: 978-1-61208-169-4

ACKNOWLEDGEMENTS

This work was undertaken, in part, 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 Data set is provided by the National Science Foundation, the US Department of Homeland Security, the WIDE Project, Cisco Systems, and CAIDA Members.

REFERENCES

[1] V. Paxson and S. Floyd, "Why we don't know how to simulate the internet," in *Proceedings of the 29th conference on Winter simulation*, ser. WSC '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 1037–1044.

[2] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 392–403, August 2001.

[3] M. Liljenstam, Y. Yuan, B. J. Premore, and D. Nicol, "A mixed abstraction level simulation model of large-scale internet worm infestations," in *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, ser. MASCOTS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 109–. [Online]. Available: http://portal.acm.org/citation.cfm?id=882460.882592

[4] M. Crovella, C. Lindemann, and M. Reiser, "Internet performance modeling: the state of the art at the turn of the century," *Performance Evaluation*, vol. 42, no. 2-3, pp. 91 – 108, 2000.

[5] S. Wei, J. Mirkovic, and M. Swany, "Distributed worm simulation with a realistic internet model," in *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, ser. PADS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 71–79.

[6] J. H. Cowie, D. M. Nicol, and A. T. Ogielski, "Modeling the global internet," *Computing in Science and Engineering*, vol. 1, pp. 42–50, 1999.

[7] H. Ringberg, M. Roughan, and J. Rexford, "The need for simulation in evaluating anomaly detectors," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 55–59, January 2008.

[8] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: From art to science," in *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 205–211.

[9] M. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the internet," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 239–245.

[10] Y. Shavitt and E. Shir, "Dimes: let the internet measure itself," *SIGCOMM Comput. Commun. Rev.*, vol. 35, pp. 71–74, October 2005.

[11] R. N. C. Centre. The ripe atlas website. *Last Accessed 5 June 2011*. [Online]. Available: http://atlas.ripe.net/

[12] 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.

[13] 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

[14] S. McCanne and S. Floyd. ns–network simulator. *Last Accessed 5 June 2011*. [Online]. Available: http://www.isi.edu/nsnam/ns/

[15] 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.

[16] V. Paxson, "End-to-end routing behavior in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 36, pp. 41–56, October 2006.

[17] D. Abramson, J. Giddy, and L. Kotler, "High performance parametric modeling with nimrod/g: Killer application for the global grid?" in *Proceedings of the 14th International Symposium on Parallel and Distributed Processing*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 520–. [Online]. Available: http://portal.acm.org/citation.cfm?id=846234.849304

[18] B. Donnet, personal communication, 2011.

[19] T. McGregor, "DoubleTree with many sources," in *ICIMP11: The Sixth International Conference on Internet Monitoring and Protection*. Sint Maartin Island, The Netherland Antilies: IARIA, March 2011.

 ISBN: 978-1-61208-169-4