Using Realistic Internet Topology Data for Large Scale Network Simulations in OMNeT++

Roland Bless

Institute of Telematics, Universität Karlsruhe (TH),
Zirkel 2, D-76128 Karlsruhe, Germany
Phone: +49 721 608 6411, Fax: +49 721 388097, `bless@tm.uka.de`

Abstract. Results from simulation models can be used to derive implications for real scenarios only when the model is designed very close to reality. Besides solving the question of how to generate data traffic in the simulation, one has to solve also the problem of how to generate larger realistic topologies. The latter are required in many cases to evaluate protocols and mechanisms with respect to scalability. Instead of using algorithms for generating Internet-like topologies artificially, it is possible to use sources of the Internet's real Autonomous System topology for generating the simulation topology. This paper describes a first approach of how to make use of such data for OMNeT++ simulations and presents first experiences with this approach.

1 Introduction

Results from simulation models can be used to derive implications for real scenarios only when the model is designed very close to reality. Network simulations for Internet-related protocols require several special considerations [2]. Besides solving the question of how to generate data traffic in the simulation [7], one has to solve also the problem of how to generate larger realistic topologies. The latter are required in many cases to evaluate protocols and mechanisms in respect of scalability.

The real Internet topology consists of at least two different hierarchy levels. The coarser first level consists of several interconnected *Autonomous Systems (ASs)* (cf. fig. 1). An Autonomous System constitutes an administrative domain, usually operated by an Internet service provider (ISP). The second hierarchy level can be found within an AS. It comprises all the routers and their interconnections within an AS (strictly speaking this level can consist of intra-domain hierarchy levels, too, e. g., OSPF routing areas for instance). This topology is usually neither propagated nor directly visible outside the AS in order to limit the propagated routing information and its updates.

Algorithms for generating Internet-like topologies have to obey some rules which have recently been formulated as several "power laws" [1,4] for the first hierarchy level of interconnected ASs. More detailed analysis of the Internet topology by a different approach can be found in [8].

Instead of using algorithms for generating Internet-like topologies artificially, it is possible to use sources of the real AS topology for generating the simulation topology.

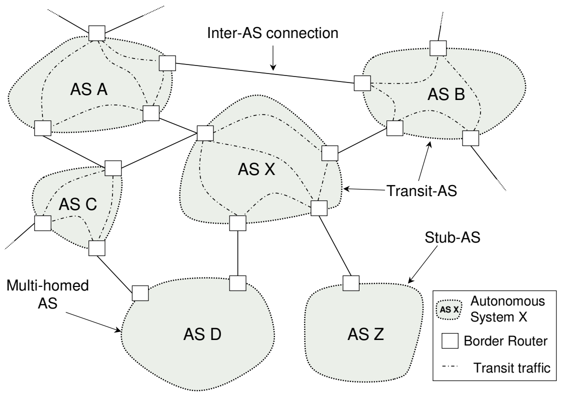


Fig. 1. Sample of connectivity between different types of Autonomous Systems

This paper describes a first approach of how to make use of such data for OMNeT++ simulations and presents first experiences.

2 Sources of Topology Information

The AS topology level can be used for evaluating different aspects of inter-domain protocols, e. g., stability of the border gateway protocol BGP [3] or scalability of new signaling protocols for resource reservation [6]. Currently, there exist more than 12000 different ASs which have each a unique assigned AS number. One can distinguish several different AS types: a *stub AS* is only source or sink of data, whereas a *transit AS* also carries and forwards traffic that has neither its destination nor its origin in this AS. Stub ASs can be *single-homed* or *multi-homed*, i. e., having only one or more connections to different providers respectively. Furthermore, there may exist *pure transit ASs* or *mixed ASs*. The latter carry transit traffic and comprise also networks that are sources and sinks of traffic.

One source for getting information about the real Internet AS level topology are routing tables for inter-domain routing generated by the routing protocol BGP. A routing table constructed by BGP within an AS represents only a particular view of this AS, because BGP is a path vector protocol. The routing table contains a set of AS paths for each destination network prefix address (cf. fig. 2). Therefore, not all connections between other ASs are visible for this particular AS. However, there exists a “route-views” project [5] at the University of Oregon in order to enable service providers to check their own routing information from another AS’s view. Currently, over 50 providers are contributing their routing information to a dedicated BGP router. BGP routing tables are archived automatically several times a day and are made available for download.

In comparison to BGP tables other sources of AS topology information, e. g., gathered by router-level path traces, show usually a smaller degree of coverage. Therefore, a small program written in Perl was used to produce a graph representation for the AS

```
route-views.oregon-ix.net>show ip bgp
BGP table version is 694918, local router ID is 198.32.162.100
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*  3.0.0.0          204.42.253.253                         0 267 1225 701 80 i
*                   129.250.0.1              6             0 2914 701 80 i
*                   129.250.0.3              0             0 2914 701 80 i
...
*  128.134.151.128/25 195.66.225.254                         0 5459 3549 4766 9526 i
```

Fig. 2. A small excerpt from a BGP table dump

topology from the BGP routing table data archived at route-views. The BGP routing table data is usually dumped in a human readable ASCII format generated by a CISCO router as output from the command sequence `show ip bgp`. The routing table is quite huge in this representation: it comprises currently (as of date January 7th, 2002) more than 360 MB in its uncompressed form.

The script also analyzes the type of ASs and creates a topology data file which is suitable for input by OMNeT++ simulations. When constructing the AS graph, an important simplification is made: it is assumed that links are bidirectional, so the graph is *undirected*. Furthermore, due to the fact that the assigned AS numbers are not always continuous, the real AS numbers are mapped to a continuous number space of unsigned integers starting at 1.

Source	Number ASs (transit, mixed, stub)	Conn-ections	Net-works	Paths	Max. path-length
route-views.oregon-ix.net (7.1.2002)	12446 (82,2919,9445)	28872	115156	4423339	24
route-views.oregon-ix.net (18.8.2001)	11621 (85,2637,8899)	26963	112276	4239791	31
route-views.oregon-ix.net (18.8.2000)	8326 (77,1894,6355)	18857	93513	1772139	20
route-server.ip.att.net (18.8.2000)	8221 (52,1107,7062)	12710	84116	1524708	17
route-server.cerf.net (18.8.2000)	8223 (47,1057,7119)	12382	83942	331700	17

Table 1. Characteristics of Autonomous Systems in the Internet

3 Implementation in OMNeT++

The objective of the now discussed OMNeT++ simulation was to evaluate a new inter-domain signaling protocol for resource reservation. In this particular case, it was assumed that there is only one signaling entity within each AS. Signaling messages were generated and consumed by hosts located in non-pure transit ASs. Furthermore, each signaling entity has to store a complete routing table to all other signaling entities. It was assumed that the routing tables are static, i. e., no link failures between ASs were simulated. Nevertheless, this implies a huge amount of memory usage for routing table data.

The design process in OMNeT++ was as follows: no module uses `activity()`, only `handleMessage()` was used in order to avoid problems with stack memory consumption when having several thousand modules. Instead of generating NED code from the topology data, OMNeT++'s feature for dynamic module creation was used (it is described in the OMNeT++-manual). If you look at output from the `nedc` compiler you see many repeated calls similar to those caused by unrolling loops. Therefore, it is more efficient to directly create the modules within OMNeT++ dynamically.

The OMNeT++ network description file contains only one simple module which reads in the topology data and creates all further modules dynamically during its `initialize()` call. Its `handleMessage()` method can be left empty. The dynamically created modules are subsequently initialized automatically by the OMNeT++ kernel. A topology file has the following format (assuming that total number of ASs is N):

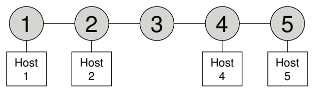
```
N
asnum1 (type,outdegree)
...
asnumN (type,outdegree)
asnum1 : ASa ASb ASc .... ASz ;
...
asnumN : ASv ASw ASx .... ASr ;
```

The total number N of nodes in the graph is given in the first line of the topology data file. This is useful in order to know how many times one has to loop over the lines in the file, and, to temporarily store the actual module pointers in an array. Subsequently, a list of all nodes follows, describing its type (1 = single-homed, 2 = multi-homed, 3 = mixed, 4 = pure transit) and the outdegree, i. e., the number of outgoing links (or gates in OMNeT++ terminology). While parsing this list, the necessary modules are created dynamically (using `cModuleType::create()`) with their respective incoming and outgoing gates (using `cModule::setGateSize()`). The second list consists of an adjacency list which describes the interconnections between all ASs. Therefore, after the modules have been created, all necessary connections between the modules are established (using `connect()`). Modules for hosts that initiate signaling messages are additionally generated and connected for each AS that is not a pure transit AS. A topology data file for the Internet topology from August 18th 2001 needs approximately 500 kB of file space.

A very simple 5 node topology with a pure transit AS in the middle is described by the following topology file in figure 3:

```
5
1 (1,1)
2 (3,2)
3 (4,2)
4 (3,2)
5 (1,1)
1 : 2 ;
2 : 1 3 ;
3 : 2 4 ;
4 : 3 5 ;
5 : 4 ;
```

(a) Topology description file example



(b) Respective topology example

4 Experiences

The previously mentioned simulation model was used to run a simulation under Linux with an Internet topology from march 2000 consisting of 7183 ASs (53 transit, 1053 mixed and 6077 stub ASs). The routing tables were calculated by using the `cTopology` class (using `unweightedSingleShortestPathsTo()`) for each AS and dumped into a file in order to save calculation time when doing several simulation runs. The file size of all routing tables is around 738 MB in this case. The actual memory consumption is considerably higher because each routing tables entry needs additional pointers (each consuming 4 Bytes of memory). The current Linux kernel 2.4 limits the memory size of a user space process to 3 GB on Intel platforms so newer and current AS topologies could not be used yet. After simulation initialization the memory usage was around 2.2 GB. As hardware platform a two processor 1 GHz Pentium III with 4 GB RAM was used.

Creating and connecting all the modules (more than 14000, counting signaling nodes and hosts) from the topology data file is very fast (only 2 seconds), whereas initializing them requires some tens of seconds. The routing table calculation was still reasonably fast (around 20 minutes), but reading the precalculated routing tables from a file reduces the setup time considerably (down to 3 minutes). However, during all the time the OMNeT++ simulation kernel was stable.

5 Conclusion and Outlook

OMNeT++ is a very efficient simulation environment when doing large simulations with several thousands of modules. This makes it possible to use “real” large scale topologies for network simulations. It is planned to run simulations with more current AS Internet topologies on 64-bit platforms in order to break the 3 GB process memory limit on Intel Linux platforms. Furthermore, it is planned to generalize the presented first approach for topology data file input and to integrate it into the OMNeT++ kernel.

References

1. M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proceedings of the ACM symposium on Communications architectures & protocols*. ACM, 1999. SIGCOMM 1999.
2. S. Floyd and V. Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, Aug. 2001.
3. S. Halabi and D. McPherson. *Internet Routing Architectures, Second Edition*. Cisco Press, 201 West 103rd Street, Indianapolis, IN 46290, USA, 2 edition, 2000. ISBN 1-57870-233-X.
4. A. Medina, I. Matta, and J. Byers. On the Origin of Power Laws in Internet Topologies. *ACM Computer Communications Review*, 30(2), 2000.
5. D. Meyer. Route Views Project Page. `http://www.routeviews.org/`, Sept. 2000.
6. Next steps in signaling (nsis) charter. `http://www.ietf.org/html.charters/nsis-charter.html`, Dec. 2001.
7. K. Park and W. Willinger, editors. *Self-Similar Network Traffic and Performance Evaluation*. Wiley, 2000.
8. D. Vukadinović, P. Huang, and T. Erlebach. A Spectral Analysis of the Internet Topology. Technical Report ETH TIK-Nr. 118, Inter-Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, July 2001. `http://www.tik.ee.ethz.ch/~huang/publication/nls-tr.pdf`.