Network Topology Analysis and Internet Modelling with Nem

Damien Magoni

Network Topology Analysis and Internet Modelling with *nem*

Damien Magoni
Université Louis Pasteur – LSIIT
Boulevard Sébastien Brant
67400 Illkirch, France
magoni@dpt-info.u-strasbg.fr

***Abstract*— The design of network protocols is greatly accelerated by the use of simulators particularly when studying a protocol's behavior in a large internetwork topology. However, the accuracy of the simulation results is heavily affected by the input network topology. As taking a real map as an input is not always feasible, artificially created topologies are often used. There exist many network topology generators available but recent discoveries on the Internet topology have made most of them obsolete when it comes to modelling a typical part of the Internet. This paper presents a free open source software designed for network topology analysis and modelling called network manipulator (*nem*). It is capable of creating realistic Internet-like topologies and it can check proof them on the fly by a thorough topology analysis. The paper especially focuses on the architecture and the capabilities of the software.**

I. Introduction

The use of simulation for protocol design has increased the need for accurate network topologies. These topologies used as input for simulations must be realistic with respect to the OSI level of the protocol being evaluated. In a simulation, a layer 2 protocol for example should be evaluated on a topology similar to its real topology (e.g. a bus, a ring, etc.) [1]. Thus, as many of the layer 3 protocols are created in order to be deployed in the Internet (e.g. protocols for routing, multicasting, mobility, etc.), their simulation should be performed upon Internet-like topologies.

Early network topology generators were rather basic and not accurate in internetwork topology modelling [1], [2]. However, since the discovery of the power laws in the collected maps of the Internet, many topology generators have been created to fulfill this task. They use different method to reach their goals and the graphs that they can generate should still be checked against the highest number of criteria to ensure their accuracy in modelling the topological properties found in the Internet maps. We introduce a free open source software that not only generates Internet-like graphs (by several methods) but also analyzes their topological properties for an immediate control of their realism. The paper contains five sections. We briefly review other similar tools in section 2. The capabilities of our software are detailed in section 3. The architecture of *nem* is presented in section 4. Section 5 describes some important parts of the *nem* implementation. Finally, section 6 contains results showing the accuracy of the *nem* graphs *vs* the collected Internet maps and a performance comparison of *nem vs* other power law topology generators.

[1]This is an extended version of an article presented at the 10th IEEE/ACM *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* in 2002.

II. Previous Work

Among the topology generators, one of the earliest and most famous model was designed by Waxman in 1988 [1]. This kind of model is usually called flat topology model. The nodes are randomly placed on an Euclidean plane irrespective of any hierarchy order among them. This model was later replaced by hierarchical topology models such as the Tiers [3] and the Transit-Stub [2] models. These models try to enforce a multi-level hierarchy that can be found in the Internet (e.g. host-router-AS). The discovery of power laws in the Internet [4]–[6] has brought the creation of a new kind of topology model. We call it the power law topology model because, as the name suggests, this model tries to reproduce power law distributions in order to generate Internet-like graphs. *nem*, like all the tested generators presented in section 6, belongs to this topology model category and it can produce router level graphs like all the other generators or AS level graphs like the Inet2 generator [7]. A recent study by Radoslavov *et al.* [8] provides a detailed analysis of topology characterization of real, generated and canonical topologies. Generated

topologies include Tiers [3], Transit-Stub [2] and Waxman [1] graphs. Several multicast scenarios are then played on all these topologies to examine the influence of the network topology on protocol simulation. Unfortunately no power law topology model generators are evaluated in this work. We partly fill this gap by looking at the values of some topology properties measured in graphs produced by recent power law topology models such as Model A [9], BRITE [10], Extended Scale Free (ESF) [11], Inet2 [7], PLOD [12] and *nem* although we do not run multicast scenarios on the produced topologies. This is left for future work. Concerning the analysis of the topological properties of graphs, few network generation tools provide an in-depth topology analysis of the produced graphs and the researcher must usually use another software than the one which generated the graphs in order to obtain detailed topological information. As the aim of the topology models and their corresponding generators is the production of realistic network topologies (especially Internet-like) for feeding network simulators such as ns-2 [13], OPNET [14] and GloMoSim [15], it is interesting for the researcher to simultaneously get the generated graph and its topology analysis. The latter should be immediately comparable to an Internet map topology analysis in order to check proof the generated graph realism. *nem* provides this functionality thus making the validation of a generated graph easier. Internet reference maps include maps created by Govindan *et al.* at the Information Sciences Institute by the Mercator software [5] and by Burch *et al.* at Lucent labs [16].

III. Capabilities

In this section we present the capabilities of our software. *nem* can do three fundamental tasks:

1) Convert network files from one format to another.
2) Analyze the topology of networks.
3) Generate random networks.

nem has no user interface but works with an ASCII batch file called a process file. Each line contains one or more instructions to be executed on a network. The detailed syntax of a process file is described in the *nem* manual.

A. Conversion and Interaction with other Software

nem can handle a variety of network file formats generated by other software. *nem* is able to load measured Internet maps such as the ones produced by the famous inter-domain router *route-views* (Autonomous System level maps) or by the mapping software Mercator (router level maps). These maps can be converted into the format of the ns-2 or OPNET simulators to serve as input topologies although they are usually too big. Our map sampling algorithm enables the generation of graphs from these maps that keep their fundamental properties while having a much smaller size (see section 6). The graphs can also be viewed in 3 dimensions by choosing the HypViewer output format and by using the HypViewer software.

B. Topology Analysis

Running an analysis on a network produces a file that has the same prefix than the network filename and has the extension `*.analysis`. The analysis concerns only the topology of the network (seen as a graph). Among the most important topology properties whose values are calculated by *nem*, we have the degree, the tree, the mesh, the distance, the number of shortest paths and the bicomponent related properties.

Depending on the characteristics of the graph (directed or not, connected or not, etc.) some properties are not calculated (see section 5 for details on the course of an analysis). Also depending on the kind of graph (random graph, Internet-like graph, Internet map, etc.) some properties are not significant (e.g. power law related properties only make sense in Internet-like graphs and maps). However, many calculated properties are valid and interesting whatever the kind of topology of the analyzed network.

Furthermore, the Dijkstra's outer loop can be parallelized in order to analyze huge networks. In the analysis of a graph, this algorithm is particularly time consuming. The all-pairs Dijkstra algorithm calculates the shortest path from any one node to another. It is the Dijkstra algorithm [17] implemented with a binary heap which runs in $O(n\log_2 n)$ and which is run for every node in the graph. Thus the whole algorithm runs in $O(n^2\log_2 n)$. In *nem*, the outer loop of the algorithm can be cut into slices to enable parallelization. At the end of the slice analysis, a partial analysis dump file of the slice is produced. The latter use of the merge operator on all the partial analysis dump files produces a file which contains the final analysis of the properties depending on the Dijkstra's algorithm (distance properties).

C. Graph Generation

The generation of a network topology is done through the use of a specification file. It is an ASCII file containing the values of the parameters needed to create a network. This file must have the extension `*.specif`. Common parameters include the number of nodes, the number of edges, and the name of the network output file. The user has the choice between several generation methods given in table I. Only the map sampling method has been created by us and we describe it below. The other methods available in the *nem* generation module have been defined in research papers. However we found convenient to implement them in *nem* (i.e. because they were not implemented elsewhere or their implementations were not available) by carefully following the algorithms described in the corresponding papers. Some

TABLE I
GENERATION METHODS

Generation method name	Author(s)	Ref.
Extended scale-free model	Albert, Barabsi	[11]
Map sampling	Magoni, Pansiot	[18]
Model A	Aiello, Chung, Lu	[9]
Modified Waxman	Waxman	[1]
PLOD	Palmer, Steffan	[12]

parameters are common to all generation methods while others only make sense and work for a given generation method. Detailed explanations for the methods, their parameters and their effects on network generation are provided in their respective papers whose references are given in table I. It's worth noticing that the user can generate a random graph if needed instead of an Internet-like graph. For doing this we provide the modified Waxman method. It works like the regular Waxman [1] algorithm but connectivity is ensured by repeatedly linking connected components together until there is only one left. Another possibility is to use our map sampling algorithm on a topology that is not a specific Internet-like topology. The resulting graph will not a priori have an Internet-like layout.

We describe here our own graph generation method. It makes use of an algorithm that randomly extract a subgraph of a real Internet map. As we focus on the router level of the Internet, we use router level Internet maps as input graphs (but it works in the same way for the AS level). By extracting nodes in a mostly random fashion, we ensure the creation of a wide diversity of subgraphs. Fig. 1 shows how the tree creation algorithm works. Fig. 1 (a) shows a small part of the map being sampled. In the first step (b), node 1 is selected randomly among all the nodes of the map. Nodes 2 and 5 are then added to a candidate list (see below) with candidate link a and b respectively. In step (c), node 5 is selected randomly among the nodes of the candidate list. Having candidate link b, it is connected to the tree (composed only of node 1 for the moment) by this link. Nodes 4, 6 and 7 are added to the candidate list with candidate link e, f and g respectively. Node 2 is already in the candidate list so it is not added to it. As node 2 is connected to 5 by link d, a random roll is performed to check if its candidate link changes to d (it is currently a). The roll fails and the candidate link is not changed. In step (d), node 2 is selected randomly among the nodes of the candidate list. Having candidate link a, it is connected to the tree (composed of nodes 1 and 5 for the moment) by this link. Node 3 is added to the candidate list with candidate link c. As node 5 is already selected, link d is added to the redundancy link list for later use (after the tree construction). In step (e), node 6 is selected randomly among the nodes of the candidate list. Having candidate link f, it is connected to the tree (composed of nodes 1, 2 and 5 for the moment) by this link. Node 8 is added to the candidate list with candidate link i. Node 7 is already in the candidate list so it is not added to it. As node 6 is connected to 7 by

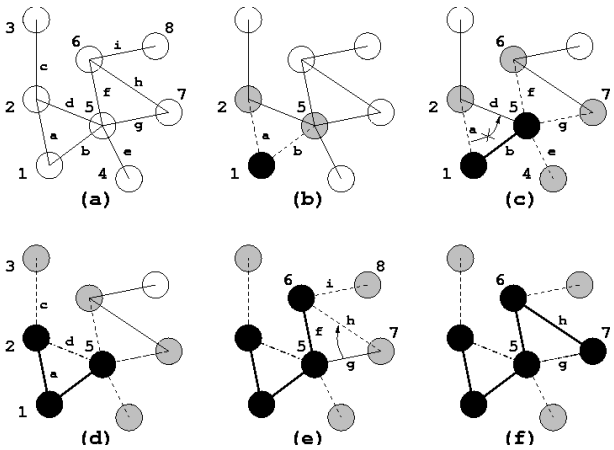


Fig. 1. tree creation process

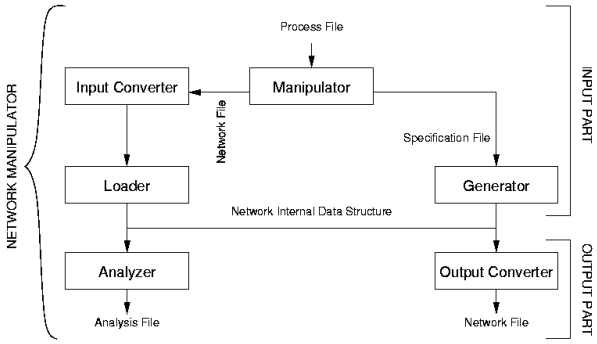


Fig. 2. *nem* architecture

link h, a random roll is performed to check if its candidate link changes to h (it is currently g). The roll succeeds and the candidate link is changed to h. In the last step shown (f), node 7 is selected randomly among the nodes of the candidate list. Having candidate link h, it is connected to the tree (composed of nodes 1, 2, 5 and 6) by this link. As node 5 is already selected, link g is added to the redundancy link list for use after the tree construction. This algorithm enables us to extract a tree having a number of nodes equal to the one of our future graph and to create a supply of redundancy links. The complete and detailed pseudo code of our map sampling method can be found in [18]. The amount of time needed to generate *nem* graphs with our map sampling method is in the order of seconds.

IV. ARCHITECTURE

The architecture of *nem* is shown in fig. 2. It is a schematic view of the modules constituting *nem* and how they interact with each other. This picture does not represent how *nem* is implemented.

The manipulator module takes as argument a process file. It is a batch file that contains a description of the actions to be carried out by *nem*. An action consists of one input part and one or more output parts. In the input part, *nem* can either load a network file (already produced by *nem* or another

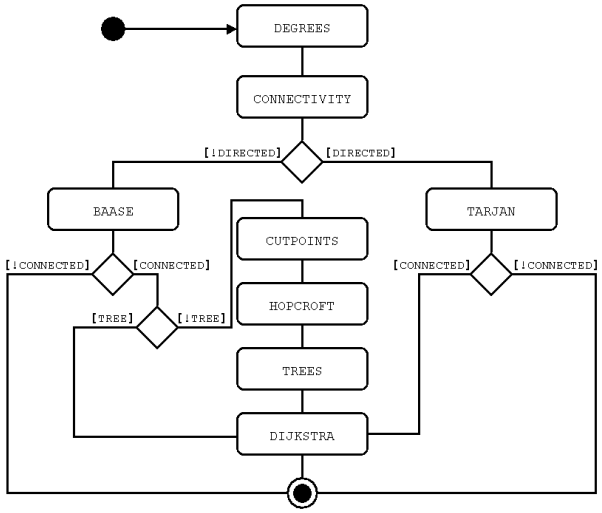


Fig. 3. UML activity diagram of the network topology analysis

generator) or generate a network from a set of parameters given in a specification file. The input converter module is controlled by the network filename extension (if known by *nem*). If the extension is `.specif`, the generator module is called, otherwise the loader module is called by the converter if the network file format is recognized (see table 1 for a list of the supported input file formats). In the output part, *nem* can analyze the graph and/or convert it in another format available for output.

V. Implementation

nem has been implemented in C++ and contains around 9000 lines of code. The C++ language has been chosen for the speed of the binary code and the efficiency of the source code management due to the object oriented nature of this programming language.

The function calls in the analysis module of *nem* is depicted as an UML activity diagram in fig. 3. The degrees' phase calculates the degree distribution and the related power law values. The connectivity phase determines if the graph is directed or not. This depends on the network input file (most of them are undirected though). If the graph is undirected, the Baase algorithm [19] is used to determine if the graph is connected otherwise the Tarjan algorithm [17] is used (although it also works for undirected graphs). If the graph is not connected, the number and the distribution size of the connected components are calculated and the analysis ends. Otherwise, if the graph is directed, the Dijkstra algorithm [17] is run upon it in order to determine the distance property values such as average path length, diameter, etc. If it is undirected and not a unique tree, the number of cut points is determined and the Hopcroft algorithm [17] is performed in order to obtain information on the biconnected components of the graph (i.e. number, distribution size, number of vertices on a cycle, number of vertices on a bridge, etc.).

The map sampling algorithm of the generator module of *nem* is described in the previous section and the explanations concerning its implementation can be found in our previous work [18].

VI. Performance Evaluation

In this section we present the performances of the graph generation module of *nem*. When we speak of *nem* graphs in this section, we speak about graphs generated by using our map sampling method (see table 1). It is important to explain this because *nem* can generate graphs by using the Extended Scale Free (ESF) method for example and in this case the graphs would be called ESF graphs (even though *nem* has produced them). We first look at the accuracy of the *nem* graphs in comparison to Internet maps and then we compare graphs generated by using several power law generation methods including ours.

A. Accuracy with Respect to Internet Maps

In order to evaluate the accuracy of the *nem* graphs in comparison to Internet maps, we have generated 20 graphs of sizes ranging from 125 nodes to 4000 nodes using the Mercator Internet map of 1999. The average analysis of the 20 graphs has been produced by *nem* just after their creation by the generation module. The figures contain the plots of these values as well as the minimum and maximum values observed in the 20 graphs (by looking in the individual analysis files). Most properties present in the Internet map are accurately reproduced in the *nem* graphs. This was the objective of our sampling method (i.e. the graphs are extracted from the map). In particular, we have looked at the average path length distributions of a sample of the extracted graphs and they all have a Gaussian shape like the one found in the Internet map (overlooking the small oscillations due to the sampling scale). This means that we can consider the average values of these distributions as being interesting distance indicators. Fig. 4 clearly shows that it is possible to extract graphs that have the same average path length than a 200 times bigger map (shown by the dotted line). All graphs of size equal or above 1000 nodes up to 4000 nodes have average distances between 8 and 11 which are typical values measured in Internet router level maps [5], [20]. For graphs of less than 1000 nodes it is hardly possible to obtain satisfying results.

It is worth noticing that our filtering function presented in [18] and which avoids extracting very high degree nodes, is necessary to achieve these results. The basic extraction is not enough to obtain accurate average path lengths at these size levels (a few thousand nodes). As we observed similar results for the node eccentricity distributions of the *nem* graphs, we can claim that the distance properties of the graphs extracted by our map sampling algorithm are satisfactory for graphs of sizes 1000 up to 4000, but we also found that these graphs do comply with the power laws found in the Internet map. As an example, fig. 5 shows the absolute correlation coefficients (ACC) for the rank and degree power laws [4] of the graphs extracted by the map sampling algorithm. All extracted graphs (from size 125 to 4000) comply with these laws for any of the generated sizes with values above the 0.95 commonly used threshold [4]. Also we found that all extracted graphs comply

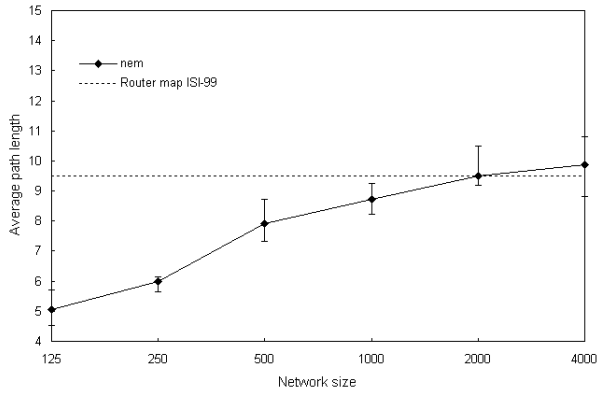


Fig. 4. average path length of the graphs

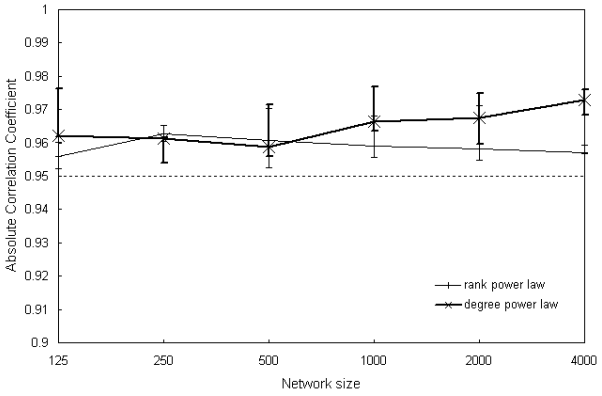


Fig. 5. rank and degree ACC of the graphs

with the tree size and tree rank power laws described in [6] when the graph size is equal or above 1000 nodes and equal or below 4000 nodes as shown in fig. 6.

To conclude, we showed that the graphs generated by *nem* comply with the rank and degree power laws found by Faloutsos *et al.* [4] for all evaluated sizes and comply with the tree power laws found in our prior work [6] for sizes comprised between 1000 and 4000 nodes. Concerning the distance properties (average path length and average eccentricity), we saw that accurate results can be achieved for evaluated sizes ranging from 1000 to 4000 nodes. These graph sizes should be appropriate for most protocol simulations (i.e. neither too small nor too big).

B. Comparison Between nem and Other Generators

In order to evaluate the accuracy of the *nem* graphs in comparison to other power law topology generators, we have generated graphs of sizes ranging from 500 nodes to 8000 nodes using the ESF, Model A, Inet2, PLOD, BRITE and *nem*. Inet2 uses an Autonomous System map of 3025 nodes collected in 1999 as a seed for its graph generation method and thus is not able to produce graphs containing less than 3025 nodes. That is why there are only 2 points for Inet2 plots in the figures. As the process of generating graphs involves random selection (and thus random rolls), we have used a sequential scenario of simulation [21] to produce the results shown in

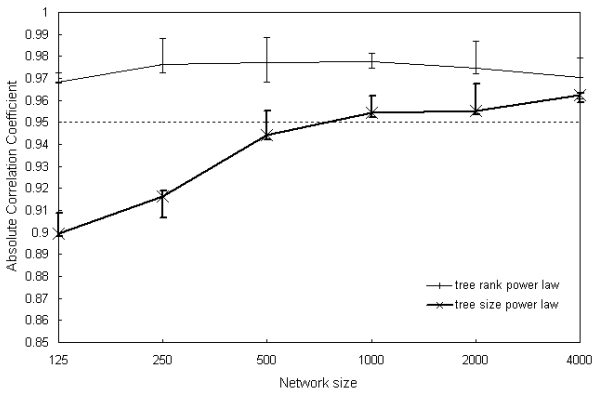


Fig. 6. tree rank and tree size ACC of the graphs

this section. We have used the Mersenne Twister code [22] for producing the random numbers needed in our simulation. As the random rolls are the only source of randomness in our simulation, we can reasonably assume that the simulation output data obey the central limit theorem. We have performed a terminating simulation where each run consists in generating a graph and determining its topological properties (i.e. one run is the time horizon). We have performed a sequential checkpoint each time after having created one graph. Convergence is reached between 8 to 21 graphs depending on the generation method and the topological property. All the simulation results have been obtained assuming a confidence level of 0.95 with a relative statistical error threshold of 5% for all measured metrics. We show the plots of the resulting values for the main topological properties concerning degree, distance, mesh and trees.

1) Degree Properties: Fig. 7 shows the degree absolute correlation coefficient (ACC) of the graphs. This indicator measures the compliance of the graph degree distribution to the degree power law (Faloutsos *et al.* [4] power law number 2). Only BRITE does not comply with this power law. BRITE uses a parameter m that indicates how new nodes connect to the graph during its construction. We have examined some degree distribution samples of BRITE graphs with $m = 2$. They contain a few degree 1 nodes that cause these graphs to not comply with power law 2. These few nodes of degree 1 are encountered in all the BRITE graphs with $m = 2$ whatever their size. We have verified that BRITE, unlike the other generators, generates graphs having a degree distribution that starts at m instead of one (when $m > 1$). A few outliers of degree less than m cause BRITE graphs to fail compliance with power law 2.

Fig. 8 shows the degree exponent of the graphs. This indicator measures the slope of the line given by the least squares fitting of the log-log plot of the node degree distribution. Like the average concisely describes a Gaussian distribution, the exponent concisely describes a power law distribution. As BRITE graphs do not comply with the degree power law, no exponent can be calculated and thus there is no plot for BRITE graphs in fig. 8. The values of most graphs are around the threshold but only ESF and *nem* graphs reach the target

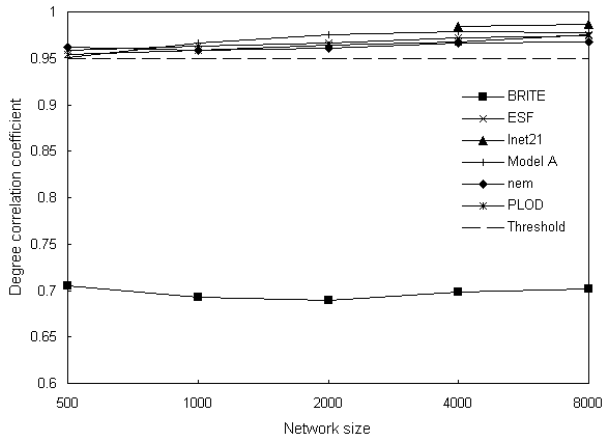


Fig. 7. degree ACC of the graphs

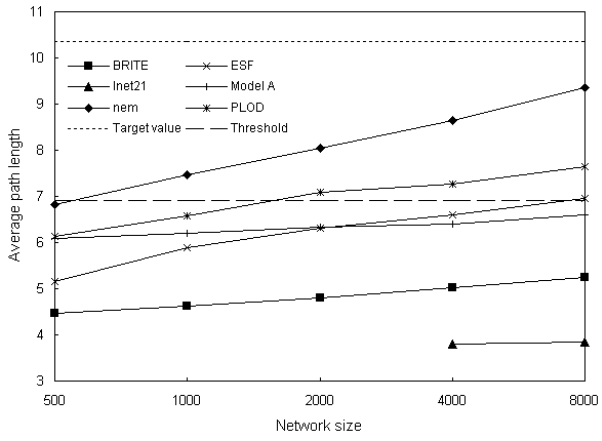


Fig. 9. average path length of the graphs

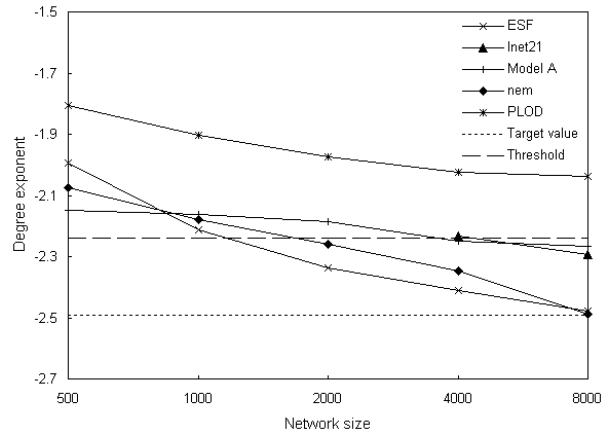


Fig. 8. degree exponent of the graphs

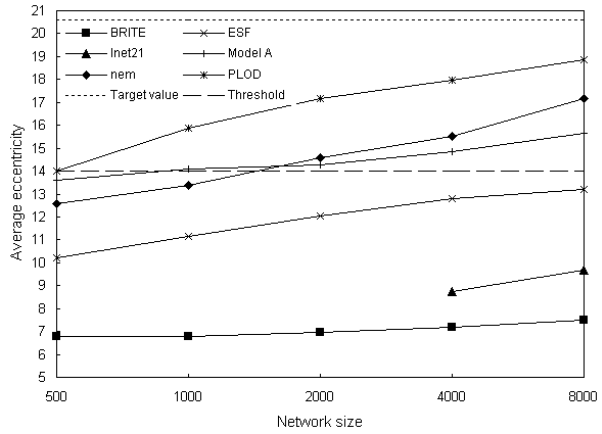


Fig. 10. eccentricity of the graphs

value when the graph size is high enough. PLOD graphs do not match this indicator.

2) Distance Properties: Fig. 9 shows the average path length of the graphs. Only *nem* graphs get close to the average path length measured in Internet maps. PLOD graphs also manage to get above the minimum threshold. All other graphs including ESF graphs clearly do not accurately model the average path length. It's worth noticing that this property is very important because the hop count is one of the only few values that can be measured and exploited by IP routing and transport protocols.

Fig. 10 shows the average eccentricity of the graphs. All graphs have difficulty reaching the target value of the indicator. However PLOD, nem and Model A manage to be around or above the minimum threshold. The other graphs are quite below it and thus their nodes are not spread as they could be in a real Internet map.

3) Mesh Properties: The definition given in [6] says that a node belonging to a cycle (i.e. a closed path of disjoint nodes) or lying on a path connecting two cycles is called an in-mesh node. The mesh is the set of all the in-mesh nodes of a graph. The mesh is important because this is the area where can be found all the redundant edges of the graph. Accurately modelling the mesh has a crucial impact on multiple shortest path or alternate path simulations. Fig. 11 shows the mesh degree exponent of the graphs. PLOD and Model A are out of the threshold area and thus do not match this indicator.

Fig. 12 shows the mesh average path length of the graphs. *nem* graphs are close to the target value especially when the graph size increases. All other graphs are below the minimum threshold of this indicator (except for the 8000-node ESF graphs) and thus do not comply with this indicator.

4) Tree Properties: We have also studied the trees in the graphs. The trees in an Internet map follow the tree size power law described in [6]. In the previous section we defined the mesh. The forest is simply the set of nodes of the graph that do not belong to the mesh. These nodes are located in trees and the union of these trees form the forest. The trees are connected to the mesh by special nodes called roots. We consider the roots as nodes belonging to the mesh. As trees are sensitive areas for link failure, it is important to model them with accuracy. We have not shown the tree rank ACC of the graphs because all graphs have values above the 0.95 threshold except BRITE graphs. Actually BRITE graphs do not comply with tree rank and tree size power laws because nearly 100% of their nodes are in the mesh. Thus there is not enough trees

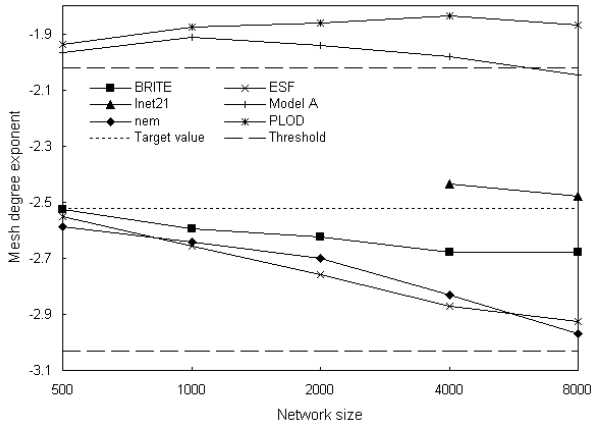


Fig. 11. mesh degree exponent of the graphs

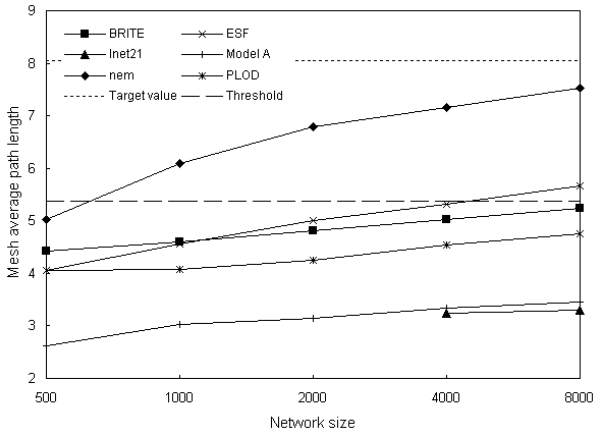


Fig. 12. mesh average path length of the graphs

in order to compute tree rank and tree size least squares fitting. Fig. 13 shows the tree size ACC of the graphs (except BRITE graphs for reasons previously explained). Model A graphs do not match the threshold of this indicator and PLOD is just close enough. Other generators have good results.

The careful study of all these indicators have shown us that only the topological properties of *nem* graphs do comply with all of them (or at least reach their thresholds) as shown in table II. A 'yes' entry means that the graphs produced by the generator comply with the indicator, a 'thre' entry means that they are close from the indicator's threshold and a 'no' entry means that they do not comply with the indicator.

VII. Conclusion

Accurate network topology generation is an important research field for protocol simulation. In this paper we have presented a software designed for network topology analysis and modelling. It is particularly suited for Internet-like topology modelling. We have detailed its capabilities, architecture and implementation and we have evaluated its performances. We have shown that *nem* is appropriate for analyzing graphs and maps obtained from various sources and that *nem* is accurate when modelling Internet-like graphs. We hope that

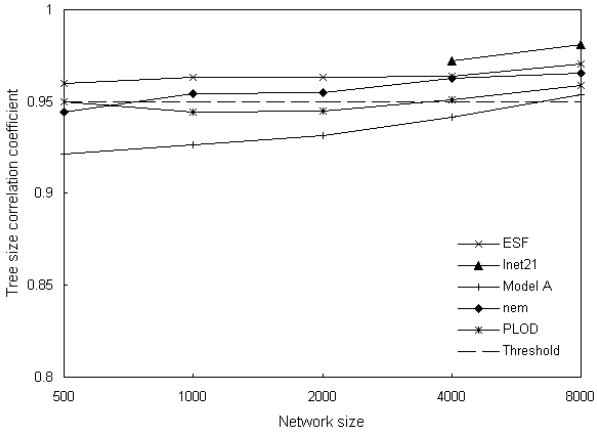


Fig. 13. tree size ACC of the graphs

this software will be a helpful tool for the network research community. *nem* is open source and available on the Web at `http://www-r2.u-strasbg.fr/nem`. For future work, we are planning to develop a graphical user interface for *nem* and to increase the number of topological parameters analyzed in the graphs (such as the clustering coefficient) We have also already added several features such as the ability to produce an overlay map from one router level map and one Autonomous System level map and the ability to load maps produced by our network cartographer software.

References

[1] B. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, December 1988.

[2] E. Zegura, K. Calvert, and M. Donahoo, "A quantitative comparison of graph-based models for internetworks," *IEEE / ACM Transactions on Networking*, vol. 5, no. 6, pp. 770–783, December 1997.

[3] M. Doar, "A better model for generating test networks," in *Proceedings of IEEE GLOBECOM'96*, November 1996.

[4] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of ACM SIGCOMM'99*, Cambridge, Massachusetts, USA, August 1999, pp. 251–262.

[5] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *Proceedings of IEEE INFOCOM'00*, Tel Aviv, Israël, March 2000.

[6] D. Magoni and J.-J. Pansiot, "Analysis of the autonomous system network topology," *ACM Computer Communication Review*, vol. 31, no. 3, pp. 26–37, July 2001.

[7] C. Jin, Q. Chen, and S. Jamin, "Inet: Internet topology generator," University of Michigan, Tech. Rep. CSE-TR-433-00, 2000.

[8] P. Radoslavov, H. Tangmunarunkit, H. Yu, R. Govindan, S. Shenker, and D. Estrin, "On characterizing network topologies and analyzing their impact on protocol design," University of Southern California, Tech. Rep., 2000.

[9] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *Proceedings of ACM STOC'00*, 2000, pp. 171–180.

[10] A. Medina, I. Matta, and J. Byers, "On the origin of power laws in internet topologies," *ACM Computer Communication Review*, vol. 30, no. 2, April 2000.

TABLE II
ACCURACY OF GENERATORS

Generator	degree ACC	degree exponent	ave. path length	eccentr.	mesh deg. exponent	mesh path length	tree size ACC
BRITE	no	n/a	no	no	yes	no	n/a
ESF	yes	yes	no	no	yes	thre	yes
Inet	yes	thre	no	no	yes	no	yes
Model A	yes	thre	thre	thre	no	no	no
nem	yes	yes	yes	thre	yes	yes	yes
PLOD	yes	no	thre	yes	no	no	thre

[11] R. Albert and A.-L. Barabási, "Topology of evolving networks: local events and universality," *Physical Review Letters*, no. 85, p. 5234, 2000.

[12] C. Palmer and G. Steffan, "Generating network topologies that obey power laws," in *Proceedings of IEEE GLOBECOM'00*, San Francisco, California, USA, November 2000.

[13] V. project, *network simulator (ns-2)*, UCB/LBNL, USC/ISI, Xerox PARC, http://www.isi.edu/nsnam/vint/.

[14] *network simulator (OPNET)*, OPNET Technologies Inc., http://www.opnet.com.

[15] *GloMoSim*, Parallel Computing Laboratory, http://pcl.cs.ucla.edu/projects/glomosim/.

[16] H. Burch and B. Cheswick, "Mapping the internet," *IEEE Computer*, vol. 32, no. 4, pp. 97–98, 1999.

[17] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*. Addison-Wesley, 1983.

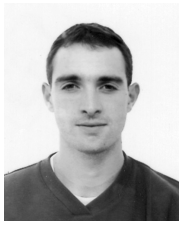
[18] D. Magoni and J.-J. Pansiot, "Internet topology modeler based on map sampling," in *Proceedings of the 7th IEEE Symposium on Computers and Communications*, Giardini Naxos, Sicily, Italy, July 2002, pp. 1021–1027.

[19] S. Baase, *Computer Algorithms*, 2nd ed. Addison-Wesley, 1988.

[20] M. Hoerdt and D. Magoni, "Completeness of the internet core topology collected by a fast mapping software," in *Proceedings of the 11th International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, October 2003, pp. 257–261.

[21] A. Law and W. Kelton, *Simulation Modelling and Analysis*, 3rd ed. McGraw-Hill, 2000.

[22] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, 1998.



Damien Magoni is an assistant professor in computer science at the Université Louis Pasteur located in Strasbourg, France. He obtained a PhD in computer science in 2002. His research interests include Internet topology, architecture and protocols.