Generating Internet Topologies with Highly Optimized Tolerance

Hao Chen and Wei Liu

April 16, 2002

1 Abstract

Due to the complexity of today's network, simulation tools are often used to evaluate new systems and protocols. In order to better analyze the performance, the topology models used in these tools are required to be as realistic as possible. Recent empirical studies have shown that Internet topologies exhibit power law statistics [6]. One theory which tries to explain this phenomenon is "self organized criticality" (SOC) [2]. Jean Carlson and John Doyle proposed another theory, called highly optimized tolerance (HOT) [4], which is a new mechanism used to generate power law distributions. Compared with SOC, HOT focuses on systems which are optimized, either through natural selection or engineering design, to provide robust performance despite uncertain environments. In this project, we will implement a topology generator based on HOT theory and verify that HOT can indeed generate topology with power-law distributions.

2 Introduction

Today simulation tools are widely used to test network protocols. These tools need network topologies as input data. In order to better evaluate the performance, the topology models used in these tools are required to be as realistic as possible. However, due to the lack of any centralized administration, it is very difficult to obtain complete topological descriptions of even a modest portion of the Internet. In fact, for reasons of security, some administrations take great effort to hide the topology of their networks from outside world. Modeling Internet topologies has become a major challenge for network researchers.

The Internet topology is usually modeled by an undirected graph where the network devices are modeled by the nodes of the graph and the communication links are modeled by the edges of the graph. One of the earliest and most famous topology models was designed by Waxman in 1988 [8]. This generator is a variant of the classical Erdos-Renyi random graph [5]; the nodes are randomly placed on an Euclidean plane and the links between each pair of nodes are created with probabilities calculated biased on the Euclidean distance between the link endpoints.

In a recent paper, Faloutsos *et al.* [6] discovered that the Internet topologies exhibit power laws of the form $y \propto x^{\alpha}$, where α is a constant, x and y are the measures of interest, and $\propto$ stands for

"proportional to ". This discovery has brought the arrival of a new kind of topology model, the power law model. As the name suggests, this model tries to use power laws to generate network topologies.

While various evidences showed the existence of power law statistics in the network topologies, it is not clear that what are the causes of this property. One theory which tries to explain this phenomenon is "self organized criticality" (SOC). Albert-Laszlo Barabasi and Reka Albert explained in their paper [2] that the scale free power law distributions found in many large networks are due to the following reasons (i) networks expand continuously by the addition of new vertices, and (ii) new vertices attach preferentially to sites that are already well connected. However, Chen *et al* [3] re-examined experimental results in BA model and found that the historical data does not support the results in [2]. At the end, they pointed out that criticality might not be the only possible origin of pow law distributions. The Internet topology might have developed over time following a very different set of growth processes, i.e., the Highly Optimized Tolerance (HOT) [4].

In this project, we will generate the WWW Web site topology using H.O.T. model and verify that H.O.T. model can indeed generate topology with power law distributions. This report is organized as follows: in section 3, we will briefly introduce the concept of the Highly Optimized Tolerance (HOT) model. In section 4, we will present the procedure of generating the Internet topology using HOT. In section 5, we will show some experimental results and finally, in section 6, we will summarize the project and point out the possible future work.

3 The Highly Optimized Tolerance (HOT) Model

The HOT model was introduced by Jean Carlson and John Doyle [4]. It was motivated by biological organisms and advanced engineering technologies. The HOT model focuses on systems which are optimized, either through natural selection or engineering design, to provide robust performance despite uncertain environments. Today's Internet is a prime example of a large-scale, highly engineered, yet highly complex system. This makes it a good candidate for analysis as a HOT system. The characteristic features of HOT systems include: (1) high efficiency, performance, and robustness to designed-for uncertainties, (2) hypersensitivity to design flaws and unanticipated perturbations, (3) nongeneric, specialized, structured configurations, and (4) power laws. These features arise as a consequence of optimizing a "design" objective in the presence of uncertainty and specified constrains.

The simplest example of HOT is Probability-Loss-Resource (PLR) problem. In the PLR problem, resources are allocated to limit average loss. For a set of abstract events with index i, $1 \leq i \leq N$, such as the occurrence of source symbols (DC), file accesses (WWW), and fire ignition and propagation (FF), we assume there is a relationship $l_i = f(r_i)$, with $l_i \propto r_i^{-\beta}$, which describes how the allocation of resources r_i limits the sizes/cost of events, l_i. Each event is assumed to be independent and initiated with probability p_i during some time interval of observation. There is an overall constraint on the resource availability: $\sum r_i \leq R$. The objective is to minimize the expected cost

$$J = \{\sum p_i l_i \mid l_i = f(r_i), \sum r_i \leq R\} \tag{1}$$

Assuming that the one-parameter resource vs loss function $l_i = f_\beta(r_i)$ is of the form:

$$f_\beta(r_i) = \begin{cases} -\log(r_i), & \beta = 0; \\ \frac{c}{\beta}(r_i^{-\beta} - 1), & \beta > 0. \end{cases} \tag{2}$$

The optimal solution minimizing J [equation (1) and (2)] is obtained using Lagrange multipliers and is given by:

$$r_i = R p_i^{\frac{1}{1+\beta}} \left(\sum_j p_j^{\frac{1}{1+\beta}} \right)^{-1} \tag{3}$$

and from (2) we get

$$l_i = \begin{cases} -\log(Rp_i) + \log(\sum_j p_j), & \beta = 0; \\ \frac{c}{\beta}[(Rp_i^{\frac{1}{1+\beta}})^{-\beta}(\sum_j p_j^{\frac{1}{1+\beta}})^\beta - 1], & \beta > 0. \end{cases} \tag{4}$$

reverting (4) yields the (noncumulative) probabilities of events of size l_i

$$p_i = c_1(l_i + c_2)^{-(1+\frac{1}{\beta})} \tag{5}$$

where

$$c_1 = (\frac{\rho}{\beta} c_2^{\frac{1}{\beta}})^{-(1+\frac{1}{\beta})}$$
$$c_2 = \frac{c}{\beta}$$
$$\rho = \sum_{j=1}^{N} p_i^{\frac{1}{\beta+1}}$$

Note that c_1 and c_2 can be treated as two arbitrary constants. As we can see, after the optimization, the relationship between p_i and l_i exhibits power-law statistics. In the next section, we will simulate the optimization process and verify the conjecture that heavy-tailed distributions result naturally from the tradeoff between the design objective and limited resources.

4 Generating Internet Topology Using HOT

Zhu *et al.* [9] proposed an algorithm of generating the WWW graph using HOT model. In this model, the Web topology is modeled by an undirected graph in which nodes represent Web pages and the edge between node i and node j represents the hyperlink between page V_i and V_j. The design objective in the Web layout model is to minimize the delay in download times and latency. Given a distribution of user interest, and a constraint on the total number of files, the average download times can be minimized by having the high hits be small files, allowing larger files for rarely requested portions of the Web site. The initial graph was generated using random graph model and the user navigation pattern was modeled as Markov chains. The optimization was done through a sequence of file splitting and merging processes, with a tradeoff between ease of navigation, which would favor fewer files, and having small files to download.

4.1 Generating Random Graph

The initial graph was generated using Waxman's random graph model. In this method, a (fixed) set of nodes is uniformly distributed in a 10×10 plane at random. A link is added between each pair of nodes with a probability given by:

$$P(u, v) = \alpha e^{(-d/(\beta L))}$$

where α and β are parameters in $(0, 1)$, d is the Euclidean distance (in the plane) between nodes u and v and L is the maximum distance between any two nodes in the plane.

4.2 The Markov Chain Model

A discrete-time Markov chain is used to model the behavior of network traffic, where all the nodes in the graph are considered as possible states for the random process, and the probability of going from node V_i to node V_j defines the transition probability p_{ij}. Let $M = [p_{ij}]$ be the transition probability matrix, then $M^n = [p_{ij}(n)]$ gives the n-step transition probabilities. The following theorem is from the standard theory of Markov chains.

Theorem 4.1 [7] For an irreducible, aperiodic, and positive recurrent Markov chain,

$$\lim_{n \to \infty} p_{ij}(n) = p_j, \qquad for all j,$$

Theorem 4.1 tells us that, after a long time, the Markov chain is going to settles down, and (independent of the initial state i) there is a probability P_j that we are in state j. The vector $P = [p_1 \ p_2 \ ... \ p_s]$ is often called the steady-state distribution.

For a given chain with transition probability matrix $M = [p_{ij}]$, the steady-state distribution p_is can be found by solving the following equations:

$$p_i = \sum_i p_i * p_{ji}, \tag{6}$$

$$\sum_i p_i = 1. \tag{7}$$

The initial transition probability matrix is set up as follows: the page V_1 is set as the entry point of every user's navigation, that is, we assume that a user starts from the front page and then proceeds to subsequent pages through hyperlinks. After downloading page V_i, the user follows hyperlinks E_{ij} to visit page V_j with probability p_{ij}. All $p_{i1}(i > 1)$ are viewed as probability of exiting the Web site from page V_i, which means that the user either stops navigating or goes on to other Web sites. For each node V_i simply assume that there is one common exiting probability p_e except for V_1, i.e., $p_{i1} = p_e(i > 1)$. The remaining probability $1 - p_e$ will be evenly distributed among all the outgoing links $E_{ij}(j > 1)$. Set $p_{ij} = 0$ if there is no link between nodes V_i and V_j. Note that the particular Markov chains defined here do not have self-loops, i.e. $p_{ii} = 0$. After

uniquely determining the entire transition probability matrix M, the access probabilities p_i can be computed through solving equation (6) and (7).

4.3 Optimization through Splitting and Merging

The heuristic optimization algorithm goes as follows:

- **Splitting** Pick node V_{i*} with the highest $p_i l_i$. Cut the node in the middle to produce two new nodes $V_{i'}$ and $V_{i''}$ with file size $l_{i'} = l_{i''} = l_i/2$. Any outgoing link E_{ij} becomes $E_{i'j}$ and $E_{i''j}$ without changing the outgoing probabilities, i.e., $p_{i'j} = p_{i''j} = p_{ij}$. Meanwhile, any incoming link E_{ji} becomes $E_{ji'}$ and $E_{ji''}$ with half incoming probability, i.e., $p_{ji'} = p_{ji''} = p_{ji}/2$. The access probability vector p remains almost the same except that $p_{i'} = p_{i''} = p_i/2$.

- **Merging** After splitting one node, find node V_{k*} with the lowest p_k and its least popular neighbor V_{k**}. Combine them into one node V_k with file size $l_k = l_{k*} + l_{k**}$. Copy all the outgoing links and incoming links for V_{k*} and V_{k**} into the new node, then merge redundant links and combine probabilities, i.e., $p_{jk} = p_{jk*} + p_{jk**}$, and $p_{kj} = \frac{p_{k*}}{p_{k*}+p_{k**}} P_{k*j} + \frac{p_{k**}}{p_{k*}+p_{k**}} p_{k**j}$. The self-loops produced by merging should be removed and all $p_{kj}, j \neq k$ need to be adjusted appropriately so that each row of M still sums up to 1. Recompute the access probability vector p.

- **Iteration** Update $J = \sum_i p_i l_i$. Repeat the above splitting-merging procedure until the improvement of J is within a certain tolerance level or the number of iterations reaches a present maximum.

The flow chart of HOT simulation is shown in figure 4.1.

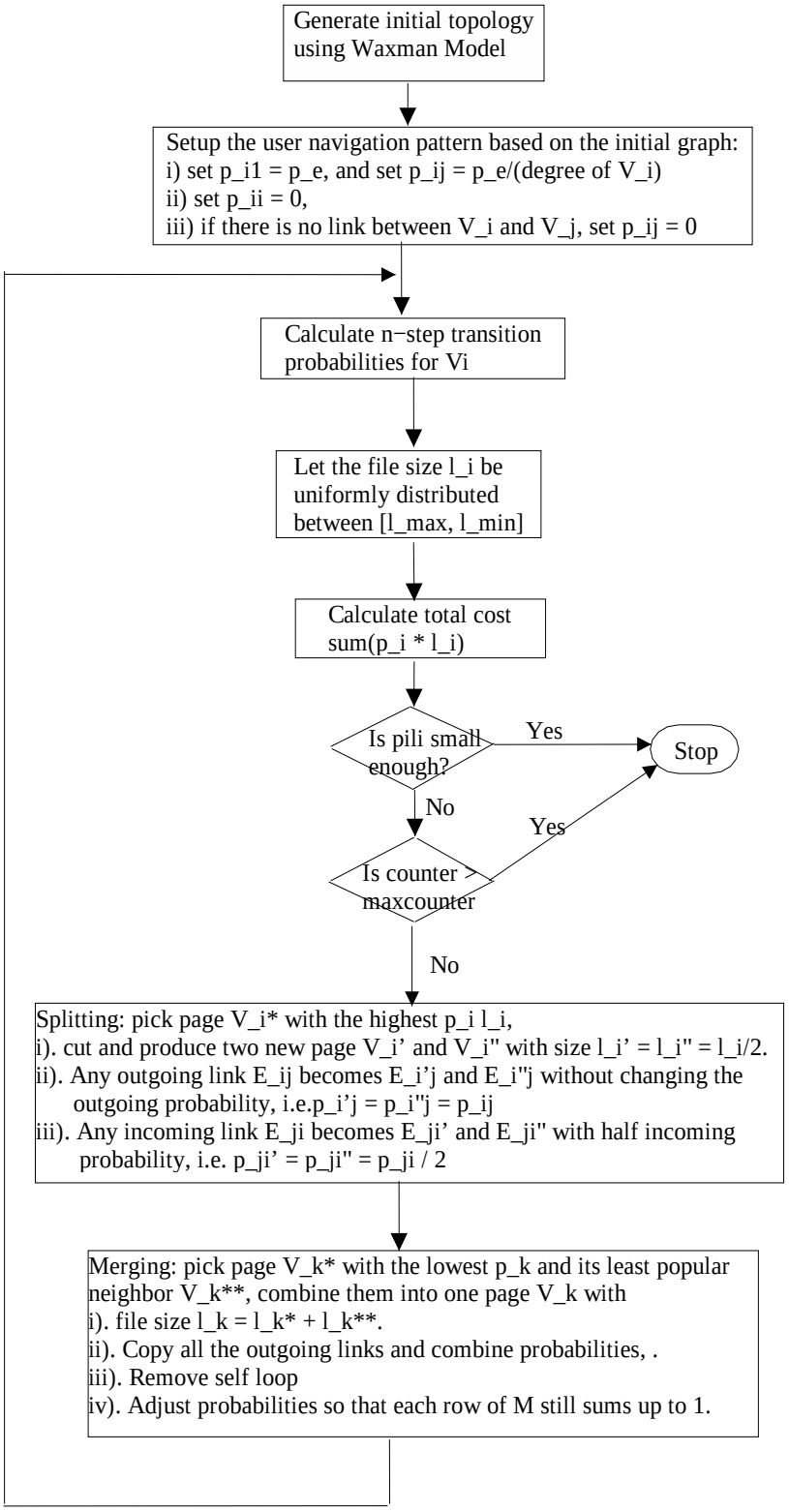

Figure 4.1 Flow chart for HOT simulation

5 Experimental Results

The simulation process is programmed using Java programming language. After user specifies the number of nodes required in the graph, the program generates the random graph using Waxman model. The random topology is going to be optimized using HOT mechanism and more specifically through a sequence of splitting and merging operations. And finally, after the total cost reaches the steady state, i.e., stop decreasing, the program stops. The output of the program include (i) the topology graph after optimization, (ii) the total cost at each iteration and (iii) the cumulative frequencies vs. file sizes before and after the optimization.

The simulation results with $N_{node} = 100$ are shown in figure 5.1, 5.2, 5.3, and 5.4. These results are very similar to the results given in [9]. Figure 5.3 illustrates the effectiveness of the splitting-merging process in reducing the cost J iteration by iteration. The improvement is quite dramatic at the beginning, then it slows down and finally reaches some steady state after 20 iterations.

In order to verify the relationship between cumulative frequencies and file sizes is power law distribution, i.e.

$$P_i^t = c_i(l_i + c_2)^{-1/\beta} \tag{8}$$

We calculate the logarithms of both sides of (8)

$$\log(P_i^t) = \log(c_1(l_i + c_2)^{-1/\beta})$$

i.e.

$$\log(P_i^t) = \log(c_1) + (-1/\beta)\log(l_i + c_2) \tag{9}$$

Let $y = \log(P_i^t)$, $x = log(l_i + c_2)$, $a = \log(c_1)$ and $b = -1/\beta$, (9) becomes

$$y = a + bx \tag{10}$$

as we can see, (10) is a linear equation with variables x, y and constants a, b, in which a represents the y-interception and b represents the slop. To determine whether the relationship between two sets of data is linear, we can apply a statistical treatment known as linear regression [1].

Given two sets of data (x_i, y_i) with n data points, the slope and y-interception can be determined using the following:

$$b = \frac{n\sum(xy) - \sum x \sum y}{n\sum(x^2) - (\sum x)^2}$$

$$a = \frac{\sum y - b \sum x}{n}$$

It is also possible to determine the correlation coefficient, r, which gives us a measure of the reliability of the linear relationship between the x and y values. A value of $r = 1$ indicates an exact linear relationship between x and y. Value of r close to 1 indicates excellent linear reliability. If the correlation coefficient is relatively far away from 1, the predictions based on the linear relationship, $y = mx + b$, will be less reliable. The correlation coefficient, r, can be determined by

$$r = \frac{n \sum(xy) - \sum x \sum y}{\sqrt{[n \sum(x^2) - (\sum x)^2][n \sum(y^2) - (\sum y)^2]}}$$

Figure 5.4 shows plot of file sizes vs the cumulative frequencies. We applied the linear regression processes on both initial data sets generated by Waxman model and the data sets after the HOT optimization. The resulting model (the gray line in figure 5.4) with $c1 = 1.034 \times 10^9$, $c2 = 318.28$, $\beta = 0.41$ well approximates the simulation results. For initial random graph the correlation coefficient is 0.67, which means that the relationship between log scale values of file sizes vs cumulative frequencies is not quite linear, so the relationship between original data, i.e., file sizes vs cumulative frequencies is not power law. After the optimization, the correlation coefficient is 0.96, which verifies that the distribution after the optimization displays power law statistics.

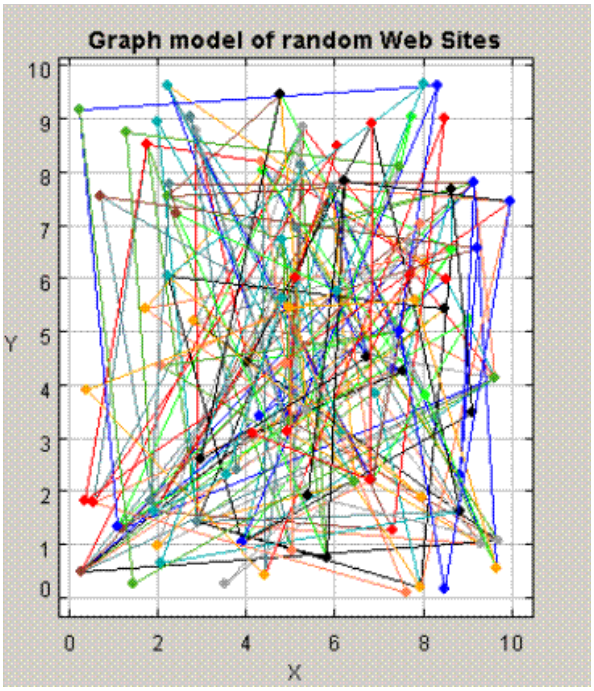

Figure 5.1 Initial random topology generated by Waxman model.

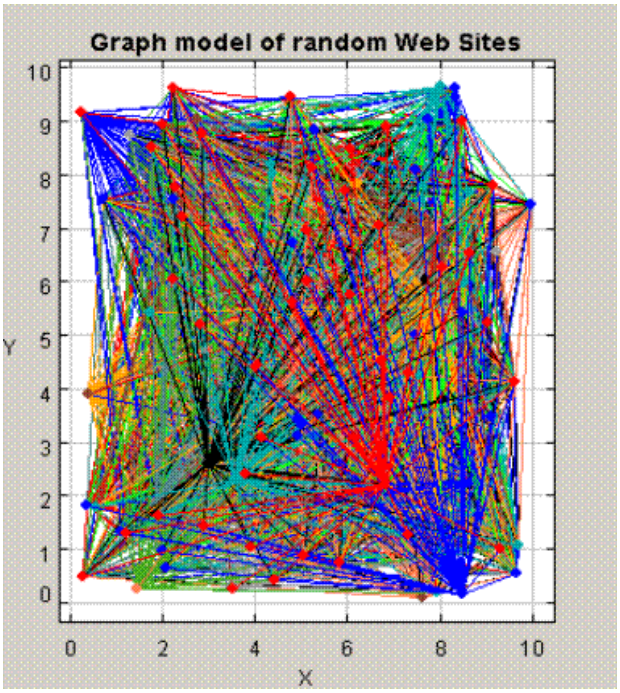

Figure 5.2 The internet topology after the optimization.

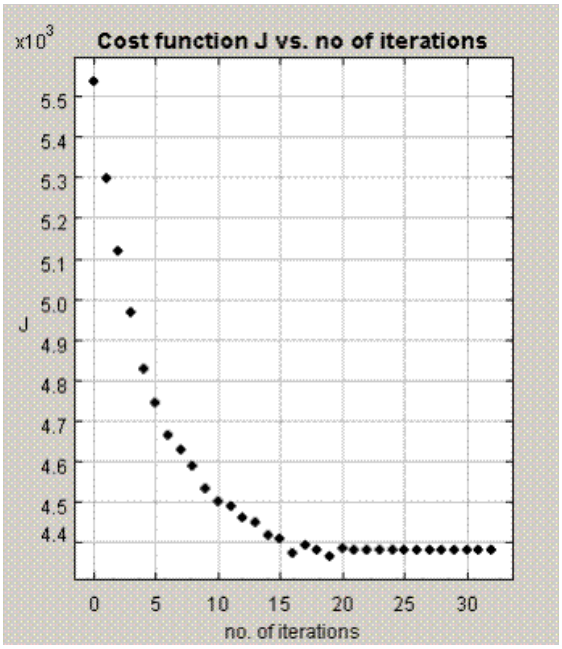

Figure 5.3 Cost function J vs. no of iterations, N_node = 100.

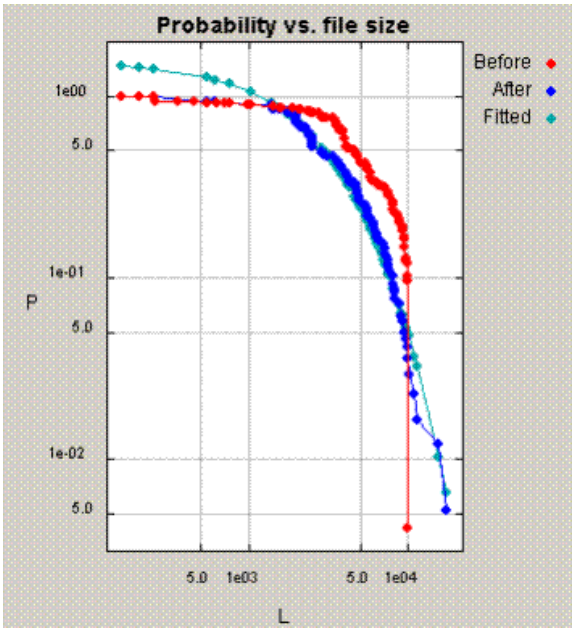

Figure 5.4 File size vs. cumulative probability for Web file transfers before optimization and after 20 iterations, N_node = 100.

6 Conclusions and future work

In this project, we verified that HOT mechanism can indeed generate power law distributions. However, the parameters used in this project are purely random. Verifying the truthfulness of this simulation model against the real Web site topology is still open to research.

References

[1] *http://physicsnt.clemson.edu/chriso/tutorials/regression/.* April 2002.

[2] R. Albert and A. L Barabasi. "Topology of Evolving Networks: Local Events and Universality". *Physical Review Letters*, 85, 2000.

[3] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. "The Origin of Power Laws in Internet Topologies Revisited". *Proc. of IEEE Infocom*, 2002.

[4] J. Doyle and J. M. Carlson. "Highly Optimized Tolerance: A Mechanism for Power-Laws in Designed Systems. *Physical Review Letters*, 1999.

[5] P. Erdos and A. Renyi. "On the Evolution of Random Graphs". *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5, 1960.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos. "On Power-Law Relationships of the Internet Topology". In *ACM SIGCOMM*, Cambridge, MA, September 1999.

[7] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering.* Addison-Wesley, 1994.

[8] B. M. Waxman. "Routing of Multipoint Connection". *IEEE Journal of Selected Areas in Communication*, 6(9), December 1988.

[9] X. Zhu, J. Yu, and J. Doyle. "Heavy-Tailed Distributions, Generalized Source Coding and Optimal Web Layout Design". Technical Memorandu CIT CDS 00-001, California Institute of Technology, Pasadena, CA 91125, 2000.