Scaling Properties of the Internet Graph

Arvind Kannan Advisor: Prof. Srinivasan Seshan

Carnegie Mellon University

1 Introduction

The Internet grows in size every day. As time progresses, more end-hosts are added to the edge of the network. Correspondingly, to accommodate these new end-hosts, Internet Service Providers (ISPs) add more routers and links. History has shown that the addition of these links maintains the power-law properties of the Internet topology [8]. Power-laws are expressions of the form $y \propto x^a$, meaning y is directly proportional to x raised to a constant a. As an example, one observed power-law in the Internet relates the outdegree of a node to a constant power of the node's rank among all outdegrees. Another power-law relates the frequency of a node's outdegree to a constant power of the outdegree. The addition of new end-hosts places a greater load on the network as a whole. As these end-hosts are added, their connectivity to all other nodes in the graph must be upheld. This means the network traffic carried over the routers and links added by ISPs will significantly increase. Fortunately, the improvement of network technology operates over the same time period. We expect the network links at the edge and core of the network to improve by a similar performance factor as the growth of traffic over time, since they both typically follow similar Moore's Law-like technology trends. These trends stem from the theory that there is an exponential growth of computing power in the network suggested by Moore's Law, which states that the speed of electronic components doubles every 18 months.

Unfortunately, due to the topology of the network and the behavior of Internet routing, the increase in load may be different on different links. As a result, it may be necessary for the speed of some hot-spot links in the network to improve much more quickly than others. If this is true, then these parts of the network are likely to eventually become bottlenecks and the network would have poor scaling properties. In such a situation, we would either need to adjust the routing behavior, remove the power-law nature of the topology or accept that end-to-end network performance of the network will not improve as rapidly as individual links. If, on the other hand, the worst congestion scales well with the network size then we can expect the network to continue to operate as it does now.

In this paper, we use extensive simulations to address this issue of how the maximum congestion in the Internet scales with the network size. Our large set of detailed simulations compute congestion on links in the network, based both on real and synthetically generated *AS-level topologies*[1]. Through our simulations, we also investigate the impact of several key factors on the worst congestion in the network. These include: (1) the routing algorithm employed by the nodes in the network (BGP policy-driven routing vs. shortest-path routing) (2) sophisticated models of communication, modeling realistic traffic demands and factoring in the popularity of a few nodes in the network over others and (3) Alternate degree distributions, e.g., an exponential distribution and power law trees evolving from Preferential Connectivity.

The key contribution of our paper is to show that maximum congestion scales poorly with the growing size of the Internet graph. Our simulations show that policy routing in the AS graph results in roughly

[1] Although we show experimental results for AS graphs, our results for maximum congestion hold equally well for router-level graphs.

the same maximum congestion as shortest path routing, *but certainly not worse.* When more realistic, non-uniform traffic models are considered, the congestion scaling properties of power-law graphs worse substantially. We also show that in terms of the maximum congestion, power law trees are considerably worse than power law graphs. In contrast, graphs with exponential degree distribution have very good congestion properties. Another key contribution of our paper is the discussion of simple guidelines on adding redundant edges between nodes that result in a dramatic improve in the congestion scaling properties of the Internet graph.

The rest of the paper is structured as follows. We discuss related work in Section 2. In Section 3, we discuss our simulation set-up. Section 5 presents the results from our simulations. In Section 6, we discuss the implications of our results on network design. Finally, in Section 7, we conclude the paper.

2 Related Work

In the past, there have been several research efforts aimed at studying the properties of large-scale, Internet-like graphs. Of these, one class of studies has proposed various models of graph evolution that result in a power law degree distribution. Notable examples include the power law random graph model of Aiello *et. al.* [3], the bicriteria optimization model of Fabrikant *et. al.* [7] and the Preferential Connectivity model of Barabasi and Albert [6, 4]. Another class of studies in this category [8, 16, 18] is aimed at analyzing the properties of power law graphs. However, most of these are based on inferences drawn from measurements of real data. Other efforts [13, 19, 18] have used these inferences to construct realistic generators for Internet-like graphs. Our simulations use topologies generated synthetically using Inet-3.0 [19].

The problem of characterizing congestion in graphs, and specifically designing routing schemes that minimize congestion, has been studied widely in approximation and online algorithms. The worst congestion in a graph is inversely related to the maximum concurrent flow that can be achieved in the graph while obeying unit edge capacities. The latter is, in turn, related to a quantity called the cut ratio of the graph. Aumann *et. al.* [5] characterize the relationship between maximum concurrent flow and cut ratio[2] and Okamura *et. al.* [15] give bounds on the cut ratio for special graphs. Algorithmic approaches to the problem (see [11, 12] for a survey) use a multi-commodity flow relaxation of the problem to find a fractional routing with good congestion properties. Although fairly good approximation factors have been achieved for the problem, most of these are not distributed, involve a lot of book-keeping, or involve solving large linear programs, which makes them impractical from the point of view of routing on the Internet. Therefore, we choose the approach of analyzing the congestion achieved from using widely implemented routing schemes such as shortest path or BGP-policy based routing.

Perhaps the work that bears closest resemblance to ours is that of Mihail *et. al.* [10]. Using arguments from max-flow min-cut theory, their paper shows that graphs obeying power law degree distribution have good expansion properties, in that they *allow* routing with $O(n \log^2 n)$ congestion, which is close to the optimal value of $O(n \log n)$ achieved by regular expanders. In addition, based on simulations run over Inet-generated topologies, the paper concludes that the congestion in power law graphs scales almost as $O(n \log^2 n)$, *even when shortest path routing is used.* The paper also shows that policy routing results in worse congestion.

Our work is different from [10] in several key aspects, a few of which we identify below. First, the theoretical analysis in [10] does not restrict the routing to shortest path and, in fact, assumes an optimal routing algorithm that minimizes congestion. We show that, in fact when shortest path routing is employed, power law graphs exhibit poor scaling properties in terms of congestion. We confirm this via detailed simulation experiments. In addition, our simulations also show that policy routing *does not* worsen the

[2] The maximum concurrent flow that can be achieved in a graph is always within a factor of $O(\log n)$ of the cut ratio, where n is the number of nodes.

maximum congestion in the network contrary to the conclusion in [10]. The evaluations of policy routing and shortest path routing in [10] only consider graphs with a small number of nodes, approximately 10,000 nodes for policy routing graphs (due to the dependence on real AS graphs) and only 23,000 for the shortest path routing graphs. Our simulations, on the other hand, consider graphs of up to 50000 nodes. Finally, we also consider the impact of different traffic workloads and deployments of parallel links on the scaling properties of the network.

3 Methodology

We use extensive simulations to understand the congestion properties of the Internet graph. In what follows, we describe the set-up for the simulations we use to corroborate and extend analytical arguments.

3.1 Problem Statement

Let $G = (V, E)$ be an unweighted graph, representing the Internet AS-AS graph, with $|V| = n$. Let d_v denote the total degree of a vertex v in G. We are given three key aspects pertaining to the graph G: the degree distribution of the graph, the routing algorithm used by the nodes in the graph to communicate with each other and the traffic demand matrix determining the amount of traffic between pairs of nodes in the graphs. We give precise definitions of these three aspects, in turn, below.

In our paper we will mostly be concerned with graphs having a power law degree distribution, defined below.

Definition 1 *We say that an unweighted graph G has a power law degree distribution with exponent α, if for all integers d, the number of nodes v with $d_v \geq d$ is proportional to $d^{-\alpha}$.*

Similarly, graphs with exponential degree distribution are those in which the number of nodes v with $d_v \geq d$ is proportional to $e^{-\beta d}$, for all integers d. Henceforth, we will refer to such graphs as power law graphs and exponential graphs respectively.

Let $\mathcal{S}$ denote a routing scheme on the graph with $\mathcal{S}_{u,v}$ representing the path for routing traffic between nodes u and v. We consider two different routing schemes in this paper:

1. **Shortest Path Routing:** In this scheme, the route between nodes u and v is given by the shortest path between the two nodes in the graph G. When there are multiple shortest paths, we consider the maximum degree of nodes along the paths and pick the one with the highest maximum degree. This tie-breaking rule is reflective of the typical policies employed in the Internet—higher degree nodes are typically much larger and much more well-provisioned providers than lower degree nodes and are in general used as the primary connection by stub networks. Additionally, to be complete in the situations that we assess, we consider breaking ties by favoring lower degree nodes and also by choosing random shortest paths.

   We report results for all three of the above tie-breaking rules and show that the results are fairly consistent across the three schemes.

2. **Policy Routing:** In this scheme, traffic between nodes u and v is routed according to BGP-policy. We classify edges as peering edges or customer-provider edges (that is, one of the ASes is a provider of the other). Typically, ASes in the Internet only provide transit for traffic destined to their customers, if any. This implies that no AS will carry traffic from its peer to another of its peers or to its provider. Similarly, no AS will carry traffic from one of its providers to one of its peers or to its provider. These rules together give rise to "valley-free" routing, in which each path contains a sequence of customer to provider edges, followed by at most one peering edge, followed by provider to customer edges. For a detailed description of the mechanism, the reader is referred to [17].

A traffic vector τ is a vector containing $\binom{n}{2}$ non-negative terms, with the term corresponding to (u, v) signifying the amount of traffic between the nodes u and v. The congestion on an edge e due to traffic vector τ and routing scheme $\mathcal{S}$ is given by the sum of the total amount of traffic that uses the edge e: $\mathcal{C}_{\tau,\mathcal{S}}(e) = \sum_{(u,v):e\in\mathcal{S}_{u,v}} \tau(u, v)$.

We define the edge congestion due to traffic vector τ and routing scheme $\mathcal{S}$ to be the maximum congestion on any edge in the graph:

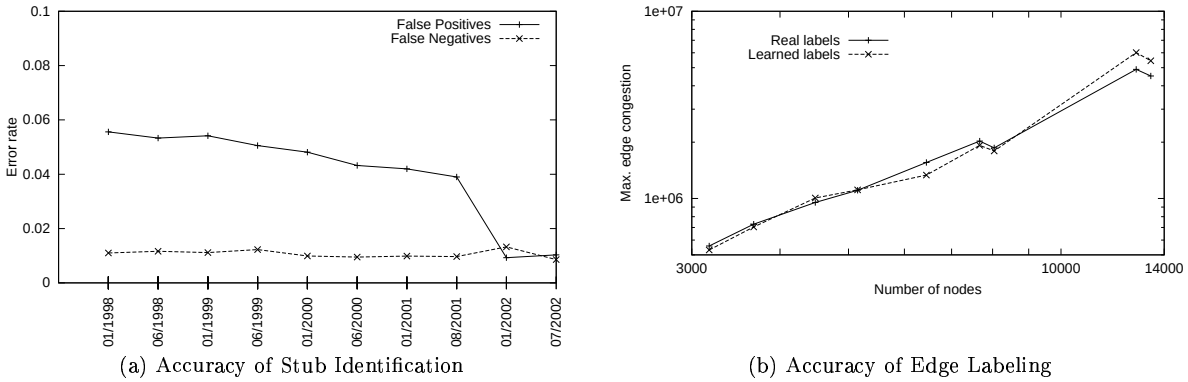
$$\text{EDGE-CONGESTION}_{\tau,\mathcal{S}}(G) = \max_{e\in E} \mathcal{C}_{\tau,\mathcal{S}}(e)$$

In this paper, we are interested in quantifying the congestion in a graph with power law degree distribution, for shortest path and policy routing schemes, due to various different traffic vectors. Additionally, we also consider the congestion across graphs with exponential degree distributions, as mentioned above, and power law trees, based on the Preferential Connectivity model [6]. Specifically, we consider the following three traffic vectors:

1. **Any-2-any:** This corresponds to the all 1s traffic vector, with a unit traffic between every pair of nodes.

2. **Leaf-2-leaf:** In order to define this model, we classify nodes in the graph as *stubs* and *carriers*. Stubs are nodes that do not have any customers. In other words, consider directing all customer-provider edges in the graph from the customer to the provider. Peering edges are considered to be bidirected edges. Then, vertices with no incoming edges (corresponding to ASes with no customers) are called stubs or leaves in the graph. In this model, there is a unit of traffic between every pair of stubs in the graph.

3. **Clout:** This model is motivated by the fact that "well-placed" sources, that is, sources that have a high degree and are connected to high degree neighbors, are likely to send larger amounts of traffic than other sources. Accordingly, in this case, $\tau_{(u,v)} = f(d_u, c_u)$, where u and v are stubs, c_u is the average degree of the neighbors of u and f is an increasing function. As in the previous case, there is no traffic between nodes that are not stubs. In this paper, we only use the function $\tau_{(u,v)} = f(d_u, c_u) = d_u c_u$ for stubs u, v.



(a) Accuracy of Stub Identification

(b) Accuracy of Edge Labeling

Figure 1: **Accuracy of heuristics**: The graph on the left shows the accuracy of our simple stub identification algorithm. The graph on the right shows the error in the maximum congestion due to our machine-learning based edge-classification algorithm.

3.2 Simulation Set-up

Our simulations serve two purposes: (1) to corroborate our theoretical results, and, (2) to characterize the congestion in more realistic network models than those considered in our analysis.

Our simulations are run on two different sets of graphs. The first set of graphs contains maps of the Internet AS topology collected at 6 month intervals between Nov. 1997 and April 2002, available at [2]. The number of nodes in any graph in this set is at most 13000, the maximum corresponding to the April 2002 set. The second set of graphs contains synthetic power law graphs generated by Inet-3.0 [19]. In this set, we generate graphs of sizes varying from $n = 4000$ to 50000 nodes. In all our simulations, for any metric of interest, for each n, we generate 5 slightly different graphs of n nodes[3] and report the average of the metric on the 5 graphs.

As pointed out in Section 3.1, in order to implement the leaf-2-leaf and clout models of communication, we need to identify stubs in the network (note that these might have a degree greater than 1). Additionally, in order to implement policy routing, we need to classify edges as peering or non-peering edges. In order to do so, for the real AS graphs, we employ the relationship inference algorithms of Gao [9] to label the edges of the graphs as peering or customer-provider edges. These algorithms use global BGP tables [1] to infer relationships between nodes. Then, we use these relationships to identify stubs (as nodes that are not providers of any other node). Henceforth, we shall refer to the real AS graphs as *accurately labeled real* graphs (ALRs). Labeling edges and identifying stubs in the synthetic graphs of Inet is more tricky since we do not have the corresponding BGP information. We will refer to synthetic graphs, labeled using the algorithms described below, as *heuristically labeled synthetic* graphs (HLSs). We use different algorithms for classifying nodes (this is key to implementing leaf-to-leaf communication) and edges (this is key to implementing policy routing in synthetic graphs). We discuss each in turn below.

Stub Identification

Here is how we identify stubs in synthetic graphs: For any edge $e = (v_1, v_2)$, we assign v_1 to be the provider of v_2 whenever $degree(v_1) \geq degree(v_2)$. Notice that we do not explicitly identify peering edges (although edges between nodes of identical degree will be bidirectional). We then identify stubs in graphs labeled as above.

We test the accuracy of this stub-identification algorithm on real AS graphs by comparing the labels produced by our algorithm to the true labels of ALRs, and compute the fraction of false positives and false negatives[4] in these. The results (see Figure 1(a)) show that our simple algorithm has very low error rate. Notice that the inference algorithms of Gao [9] have some error intrinsically and hence some of the labels on the ALRs might actually be inaccurate.

Edge Classification

Although for the purpose of classifying nodes, we simply consider all edges in the graph to be customer-provider edges, this simple scheme is not useful for the purposes of edge classification – it results in a significant error on the maximum congestion in real graphs employing policy routing. In order to improve the accuracy of labeling edges, we resort to machine learning algorithms.

Employing a good machine learning algorithm for the classification proves to be a tough task, because there is a huge bias towards customer-provider edges in the graphs (roughly 95% of the edges are customer-provider edges). We use the 3-Nearest Neighbor [14] algorithm for classifying edges as peering or non-peering: each edge in the unlabeled graph is classified as a peering edge if among the three edges most

[3] By varying the random seed used by the Inet graph generator.

[4] False positives are nodes that are identified as stubs by the algorithm, but are not stubs in the ALR. False negatives are stubs in the ALR that are not identifies as stubs by the algorithm.

similar to it in the labeled graph, at least two are peering edges. Similarity between edges is judged based on the degrees of their respective end points and neighboring vertices. We measure the accuracy of the procedure by applying it to real graphs and then comparing the classification with true labels.

Our machine learning algorithm gives only 20% accuracy on peering edges and about 95% accuracy on customer-provider edges. However, for the purposes of computing the worst congestion in the graph, this low accuracy of labeling is in fact enough. Indeed, as shown in Figure 1(b), labeling real graphs using our algorithm results in an error of less than 10% in the worst congestion (while employing policy routing) in comparison with the congestion computed on ALRs. More importantly, the growth in congestion is identical in the two cases.

We also report simulation results for congestion in power law trees and exponential topologies. A comparison of the former with power law graphs gives an insight into the significance of density of edges in the graph. The latter model is interesting because most generative models for power law topologies result in exponential distributions in the "fringe" cases. Our tree topologies evolve according to the Preferential Connectivity model [6].

To generate exponential degree distributions, we modify Inet-3.0 to generate an exponential degree distribution first and then add edges in Inet's usual way. For a given n, the exponent β for the exponential graphs on n nodes is chosen such that the total number of edges in the exponential graph is very close to that of the corresponding power law graph on n nodes[5]. To generate power law tree distributions, we use an algorithm based on the Preferential Connectivity Generative Model of Barabasi and Albert [6]. For completeness, we give a brief description of the model. The Preferential Connectivity model is as follows: We use a fixed constant parameter k. We start with a complete graph on $k+1$ nodes. We call this set of nodes the **core** of the graph. Let the graph at time i be denoted G^i. At time step $i+1$, one node is added to the network. This node picks k nodes at random and connects to them. Each vertex v has a probability $\frac{d_v^i}{D^i}$ of getting picked, where d_v^i is the degree of the vertex at time i, and D^i is the total degree of all nodes at time i. At the end of n steps, with $k = 3$, this process is known to generate a power law degree distribution.

Note that due to a lack of real data for exponential graphs and power law tree topologies, we do not have a good way of labeling edges and nodes in them. We do not perform experiments with policy routing or the leaf-2-leaf and clout traffic models for them.

4 Simulation Algorithm

To carry out the simulations, software tools were developed to take a topology graph as input and output one of several options as specified by the user. The tool was primarily developed in the C Programming Language, but it was complemented with several Perl scripts to extract the necessary information. The Perl scripts also aided in facilitating the creation of data sets that could be easily processed into graphs. These graphs are included in Section 5.

The specifications for the tool are as follows.

4.1 Name

`shortest` - A traffic simulator employing shortest path routing.
`policy` - A traffic simulator employing BGP based policy routing.

4.2 Synopsis

```
shortest [-p] [-t <rootfreq>] [-r <root>] [-m <mode>] N
```

[5] We perform heuristic hill-climbing to estimate the value of the exponent β that minimizes error in the number of edges.

```
policy [-p] [-t <rootfreq>] [-r <root>] [-m <mode>] N
```

4.3 Overview

`shortest` and `policy` employ shortest path routing and BGP based policy routing respectively to simulate Internet traffic flow across a given topology. It is assumed that the supplied topology is an AS-level topology of the format output by the `inet` Internet topology generator [19]. Based on the specified options, `shortest` and `policy` provide the corresponding output. This can be one of the following.

1. The total number of paths from every node to every other node.
2. The path trees rooted at many nodes.
3. A tabulation of every edge in the graph along with its total stress as experienced under the specified model of traffic flow.

4.4 Options

`-p`: Enable calculation of number of paths. Cannot use with `-t` or `-r`. No stresses are calculated when `-p` is specified.
`-t <rootfreq>`: Enable path tree computation for every one in `rootfreq` nodes considered.
`-r <root>`: Same as `-t`, except path tree is only computed for node specified (`root`).
`-m <mode>`: Specifies mode of simulation. Choices are

- 0: Compute leaf-2-leaf paths (default)
- 1: Compute leaf-2-any paths
- 2: Compute any-2-any paths

4.5 Output Format

The output file of `shortest` or `policy` depends on the specified options. The output file can be one of the following:

1. **shortestnumpaths.N.M**: when `-p` is chosen
2. **shortestbfstrees.N.M.freqF**: when `-t` is chosen
3. **shortestbfstrees.N.M.rootR**: when `-r` is chosen
4. **stress.N.M**: when neither `-p, -t`, or texttt-r are chosen. This is the default.

In all of the above filenames, N represents the number of nodes and M represents the mode (as designated in Section 4.4). F represents the frequency at which to choose nodes to expand into path trees, and R represents the particular node at which to expand a path tree.

The output of the **shortestnumpaths.N.M** follows the following format:

```
numnodes numpaths
```

There is only one line in this file. `numnodes` is the number of nodes in the input topology, and `numpaths` is the number of paths found with the specified mode of communication.

The output of the **shortestbfstrees.N.M.freqF** files follow the following format:

```
Root root1
node parent stress
...
--------
Root root2
...
```

`root1` and `root2` are the roots of the path trees in the file. There may be many path trees, as given by the frequency specified with the `-t` flag. For each root, there is a listing of all of the nodes in the graph and its respective parent node in the path tree constructed at the given root node. Additionally, the stress carried by that edge (child-parent) solely according to this path tree is shown. If a node is not connected to the root node, then in that root node's path tree, the disconnected node will have a parent of -1. This may occur in policy routing since some nodes may be unreachable if they violate the routing policy, but this scenario will never occur in shortest path routing.

The output of the **shortestbfstrees.N.M.rootR** files is as follows:

```
Root root
node parent stress
...
--------
```

This is the same format as the **shortestbfstrees.N.M.freqF** files, except that the files for the single root path tree only contain one path tree.

The output of the **stress.N.M** files is as follows:

```
node1 node2 stress
...
```

This is a list of edges in the graph and their accompanying total stresses. If an edge is a *peering* edge, i.e. bidirectional, between two nodes a and b, then the edge appears twice in the list: once listing the stress across $b - a$ and once across $a - b$.

4.6 Input

`shortest` and `policy` both assume their input to be of similar formats. They expect two input files each, one entitled **degrees** and the other **adj**.

The file **degrees** should be as follows:

```
id degree numleaves
...
```

The file contains many lines, each of the same format. The first field, `id`, refers to the node's id. The nodes in the graph should be ordered according to degree, and the node with the highest degree should be given the id 0. The next field, labeled `degree`, specifies the degree of the node. Since the nodes are identified according to degree, the degrees will appear in decreasing order in the file. The third field, `numleaves`, gives the number of leaves, or degree-1 nodes, adjacent to the given node.

The file **adj** should be of the following format:

```
id1 id2 weight
...
```

In this format, `id1` and `id2` refer to the node's id as mentioned above, and the `weight` field is ignored (but necessary). Both `shortest` and `policy` will use the above two files to simulate traffic flow across the given network topology.

4.7 Description

Our simulations employ several algorithms to determine the traffic congestion on each of the edges in each topology. We consider the different scenarios based on the different modes of communication.

Any-2-any

In both shortest path routing and policy routing, we are simulating all paths from one node in the graph to another node. We are also looking for the shortest path between two nodes in both routing schemes, the difference however is that policy routing specifies certain restrictions on the allowable paths while shortest path routing does not. We iterate over all the nodes in the topology and construct a breadth-first-search tree rooted at each of these nodes. During construction of this tree, we push each visited node onto a stack. Thus when we have found the shortest path to each node, we simply pop the nodes off the stack and percolate the stresses up the shortest path tree. Additional optimizations are also performed. In this manner, we efficiently label the stresses for each node in the graph. Upon iterating over all the nodes, we obtain stresses for all possible paths in the graph.

Leaf-2-leaf

In simulating leaf-2-leaf communication, we employ the same procedures as any-2-any simulation, but the calculated stresses differ. We still go through the procedure of constructing a shortest path tree and unrolling the nodes from the stack for stress computation, however the details of the computation are modified. We also add further optimizations by only considering shortest path trees at neighbors of leaf nodes (nodes with degree 1).

Clout

When simulating this model of communication, we pre-label our graph by replacing each unit traffic with the degree of the stub node multiplied by the average degree of its neighbors. We then perform normal leaf-2-leaf simulation. The resulting stresses on each link will now reflect the updated traffic matrix.

4.8 Optimizations

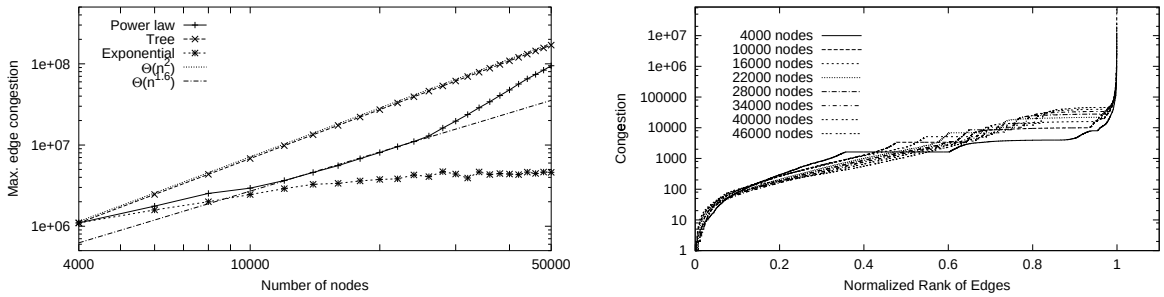
We perform a variety of optimizations on our methods as outlined above. The algorithm specified is itself a result of numerous optimizations to an original idea that was first executed. Prior to constructing a breadth-first search tree, a recursive routine was adopted. This proved to be sluggish and space intensive as well. Thus running our simulations for large datasets up to graphs of size 50,000 nodes, we experienced significant delays and initially had to await days for simple results. Once the breadth-first search was implemented, as well as the stack approach to label the stresses in a post-search fashion, dramatic increases in efficiency and speed were noted. What formerly took at least a day to compute was now possible within the span of an hour. Given such significant speed-ups, we were able to successfully conduct large dataset simulations to achieve our desired effect.

Additionally, we used simple optimizations to avoid redundant and excess computation. One such example is the fact that we do not iterate over leaf nodes (nodes with single degree). Instead, when we iterate over the neighbors of these nodes, we simply add a correction factor to the stress calculation at

the end of the computation cycle. This saves an extra iteration for every leaf node in the graph. Upon observation of these graphs, we can see that a large number of nodes have single degree, so this is indeed a fruitful optimization.

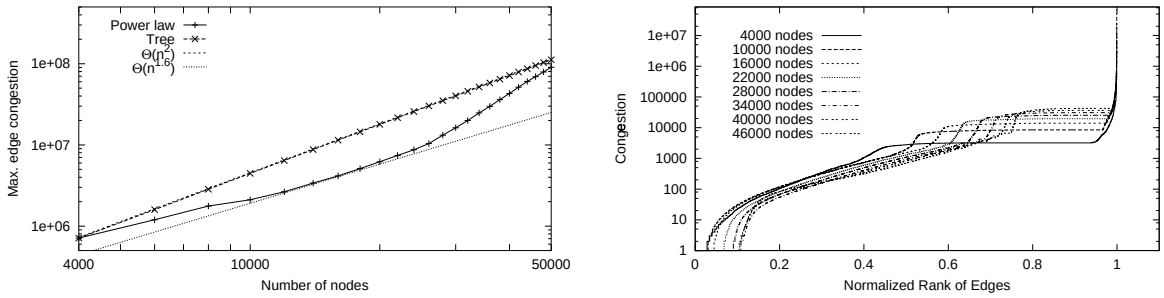
4.9 Using the Tool

To make use of the above mentioned tool, we used the output generated by the program `inet` [19]. We used `inet` to generate topologies according to power-law distributions as well as exponential distributions. We then used Perl scripts to process the output of `inet` and to generate the files `degree` and `adj` to give as input to our tools. Perl scripts were also used to automate the running of the tool and to perform several runs at once and collect the data into presentable formats upon completion.



(a) Maximum edge congestion (b) Distribution of edge congestion

Figure 2: **Edge congestion with shortest path routing and any-2-any communication**: The figure on the left shows the maximum edge congestion. The figure on the right shows the distribution of congestion over all links, with the number of links normalized to 1 in each case. The figure on the left also plots the worst congestion for exponential graphs and preferential connectivity trees.

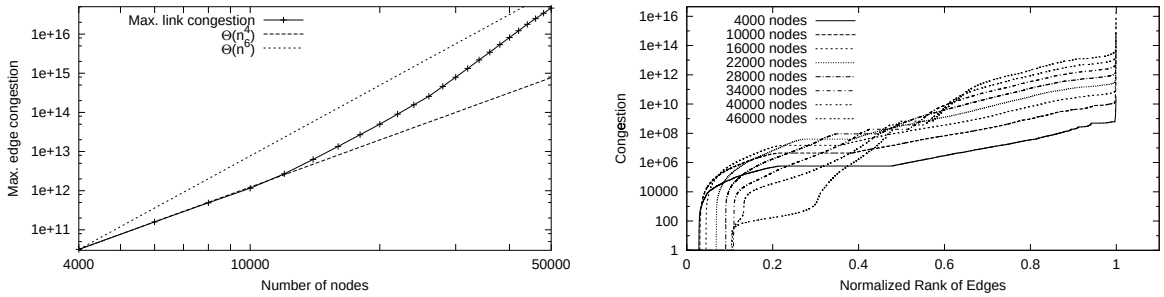


(a) Maximum edge congestion (b) Distribution of edge congestion

Figure 3: **Edge congestion with shortest path routing and leaf-2-leaf communication**

5 Simulation Results

In this section, we present the results from our simulation study over Inet-generated graphs. Henceforth, we shall use the graphs generated by Inet 3.0 *as is*, that is, we do not alter the way Inet chooses α to depend on n. (Recall that, in contrast, the simulation results in the previous section use the modified Inet 3.0 code which employs the same value of α for all n. We *do not* show results for such graphs.) In what



(a) Maximum edge congestion

(b) Distribution of edge congestion

Figure 4: **Edge congestion with shortest path routing and clout model of communication**

follows, we first show results for shortest-path routing, followed by policy-based routing. In both cases, we first present results for the any-2-any communication model, then for the leaf-2-leaf model and finally for the clout model.

5.1 Shortest-Path Routing

Figure 2(a) shows the maximum congestion in power law graphs generated by Inet-3.0 as a function of the number of nodes. We use the any-2-any model of communication here. From the trend in the graph, it is clear that *the maximum congestion in Internet-like graphs scales worse that* $n^{1+\Omega(1)}$. Notice also that the slope of the maximum congestion curve is slightly increasing. This may be explained as follows. As mentioned earlier, Inet-3.0 chooses the exponent of the power law degree distribution as a function of the number of nodes n: $\alpha = at + b$, where $t = \frac{1}{s} log \frac{n}{n_0}$, $a = -0.00324$, $b = 1.223$, $s = 0.0281$ and $n_0 = 3037$.[6] Notice that the absolute value of α decreases as n increases, and so, as our lower bound of $\Omega(n^{1+1/\alpha})$ suggests, the *slope* of the function on a log-log plot should steadily increase. In fact around $n = 28000$, α becomes less than 1 and at this point we expect the curve to scale roughly as n^2, which is the worst possible rate of growth of congestion.

The figure also shows the maximum congestion in power law trees and exponential graphs. The power law trees we generate, have the exponent α between 1.66 and 1.8, the value increasing with the number of nodes in the tree. These exponents are significantly higher than those of the corresponding power law graphs. Notice that the edge congestion on power law trees grows much faster as compared to graphs which is expected since trees have much fewer edges. Our lower bound on the maximum congestion, which holds equally well for trees satisfying power law degree distributions, predicts the slope of the curve for trees to be at least 1.5, which is consistent with the above graph.

On the other hand, we notice that edge congestion in *exponential graphs is much smaller compared to power law graphs.* In fact, edge congestion in exponential graphs has a less than linear growth (i.e., scales as $O(n)$). This could be explained intuitively as follows: Recall that for each n, we choose the exponent β of the exponential distribution so as to match the total number of edges of the corresponding n-node power law graph. Because the power law distribution has a heavier tail compared to the exponential distribution, the latter has more edges incident on low degree nodes. Consequently, low degree vertices in an exponential graph are better connected to other low degree vertices. Edges incident on low degree nodes "absorb" a large amount of congestion leading to lower congestion on edges incident on high degree nodes. As n increases the degree distribution becomes more and more even, resulting in a very slow increase in congestion.

[6] a, b and s are empirically determined constants. n_0 is the number of ASes in the Internet in November 1997.

In Figure 2(b), we show the congestion across all links in a power law graph for varying numbers of nodes. Notice that at higher numbers of nodes, the distribution of congestion becomes more and more uneven.

The corresponding set of graphs for the leaf-2-leaf communication model is shown in Figure 3. The worst congestion is consistently about 0.8 times the worst congestion for the any-2-any model (not explicitly shown in the graph). The congestion across all the edges, plotted in Figure 3(b), also displays a similar trend as for the any-2-any model – the distribution becomes more uneven as the number of nodes increases.

The results for the clout model are more interesting with the resulting maximum congestion in the graph scaling *much worse than before*. Indeed, as Figure 4(a) shows, the maximum congestion scales worse than n^5. This is because the total traffic in the graph also grows roughly as $O(n^4)$. Again, as with the any-2-any model, the smaller absolute values of α in the graphs generated by Inet-3.0 for larger values of n is a plausible explanation for the increasing slope of the curve.

The graph of the congestion across all edges in this model, shown in Figure 4(b), is equally interesting. Compared to Figure 3(b) of the leaf-2-leaf model, Figure 4(b) looks very different. Visual inspection of the two figures reveals that the unevenness in congestion is much more pronounced in the clout model of communication. To summarize, *the non-uniform traffic demand distribution in the Internet only seems to exacerbate the already poor congestion scaling of the Internet.*

5.1.1 Shortest Path Routing Variations

As mentioned in Section 3.1, we chose shortest path routing with three possible variations: favoring paths on which the nodes have higher degrees, favoring paths where the nodes have lower degrees, and choosing a random shortest path when there is a choice of more than one.

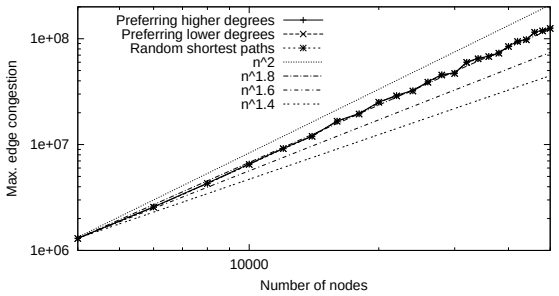


Figure 5: **Edge congestion with shortest path routing and any-2-any communication, with** $\alpha = 1.23$. The figure plots the three different variations of breaking ties in shortest path routing.

In assessing the deviation between the three different means of tie-breaking for shortest path routing, we chose to simulate these graphs by setting α to be a constant value in Inet 3.0 and compared the resulting relations between maximum edge congestion and the number of nodes.

As Figure 5 depicts, the difference between the three types of tie-breaking methods is hardly noticeable. In fact the graphs seem to coincide, but upon close inspection of the generated data slight differences are noted due to variation. The same case holds true for Figures 6 and 7. We thus conclude that our scheme of breaking ties by favoring paths containing nodes with greater degree does not skew our results.

5.2 Policy-Based Routing

Figure 8 shows the maximum edge congestion for the three communication models when policy based routing is used. For the any-2-any and leaf-2-leaf models, shown in Figure 8(a), the maximum edge

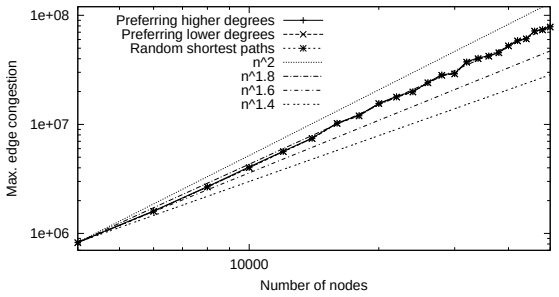


Figure 6: **Edge congestion with shortest path routing and leaf-2-leaf communication, with** $\alpha = 1.23$. The figure plots the three different variations of breaking ties in shortest path routing.

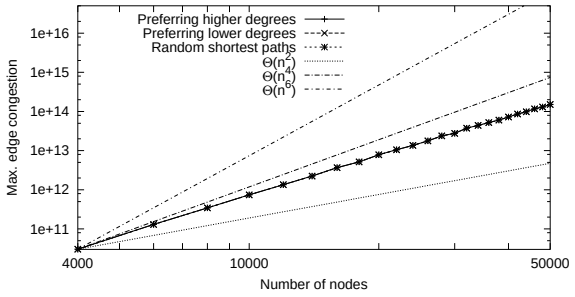
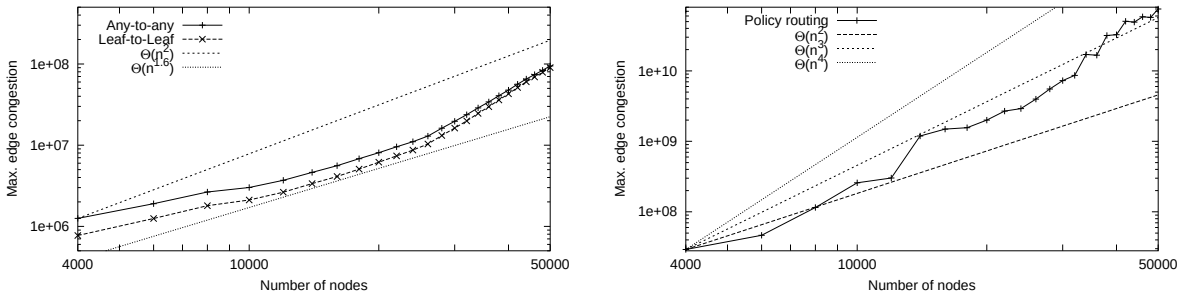


Figure 7: **Edge congestion with shortest path routing and clout model of communication, with** $\alpha = 1.23$. The figure plots the three different variations of breaking ties in shortest path routing.



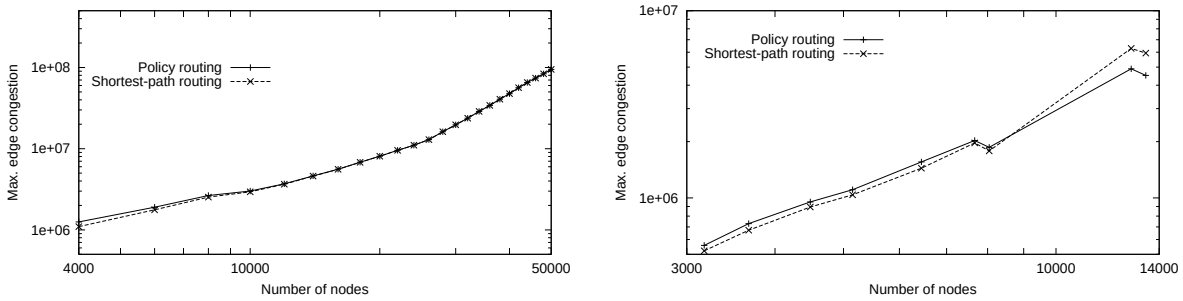
(a) Any-2-any and Leaf-2-leaf communication

(b) Clout model

Figure 8: **Maximum Edge congestion with policy-based routing in HLSs**

congestion scales almost identically to that for shortest path routing (compared with Figure 2(a) and 3(a)). However, somewhat surprisingly, for the clout model, congestion under policy based routing scales only as n^3 compared to over n^5 for shortest-path routing.

Figure 9(a) compares maximum congestion obtained for policy routing to that for shortest path routing. Notice that the two curves are almost overlapping, although policy routing seems to be slightly worse when the graph is small and gets better as the graph grows larger. This observation can be explained as follows: policy routing disallows certain paths from being used and could thus, in general, force connections to be routed over longer paths. This would increase the overall traffic in the network leading to higher congestion,



(a) Edge congestion on synthetic graphs

(b) Edge congestion on real graphs

Figure 9: **Comparison of edge congestion for shortest path and policy based routing in the any-2-any model**

especially for smaller numbers of nodes. However, as the size of the graph grows, there are more and more shortest paths available. As a result, the constraints placed by policy-based routing might not have any significant impact on the path lengths in the graph. In fact, at higher numbers of nodes, policy routing could provide better congestion properties, albeit only marginally different, than shortest path routing. This is because while shortest path routing always picks paths that go over high degree nodes, a fraction of these paths might not be allowed by policy routing as they could involve more than one peering edge. In this case, policy routing moves traffic away from the hot-spots, thereby, partially alleviating the problem.

In order to verify that the above observation is not just an artifact of our machine learning-based labeling algorithms, we plot the same curves for ALRs in Figure 9(b). These display exactly the same trend—policy routing starts out being worse than shortest path, but gets marginally better as n increases. To summarize, *policy routing does not worsen the congestion in Internet like graphs, contrary to what common intuition might suggest. In fact, policy routing might perform marginally better than shortest path routing.*

6 Discussion

Our simulation results have shown that the power law nature of the Internet graph causes the maximum congestion in the network to scale rather poorly. As mentioned in Section 1, this implies that *as the Internet grows in its size, the uniform scaling in the capacities of all links in the Internet graph according to Moore's Law, might not be enough to sustain the increasing congestion in the graph.* Our results show that the high degree nodes, which are typically in the core of the Internet, will get congested more quickly over time than the edges. In such a situation, to enhance the scaling properties of the network, it might become necessary to either change the routing algorithm employed by the nodes or alter the macroscopic structure of the graph. We address the latter issue in this section.

6.1 Adding Parallel Network Links

In this section, we examine ways in which additional links can be placed in the network, so as to contain the effect of bad scaling of the maximum congestion. Specifically, we consider the model in which each link can be replaced by multiple links (between the same pair of nodes) that can share the traffic load. Ideally, we would like to provide sufficient parallel links between a pair of nodes, so that the total congestion on the corresponding edge divided equally among these parallel links, even in the worst case, grows at about the same rate as the size of the network. The number of parallel links between a pair of nodes may need to

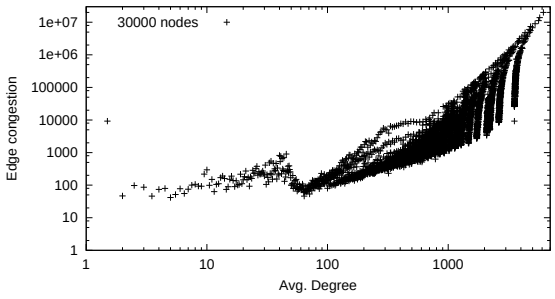


Figure 10: **Edge Congestion versus the average degree of the nodes incident on the edge (any-2-any model with shortest path routing).** The congestion is higher on edges with a high average degree.

change as the network grows to achieve this goal. Notice that this change does alter the degree-structure of the graph, but the alteration is only due to increased connectivity between *already* adjacent nodes[7]. This does not require new edges between nodes that were not adjacent before.

In some ways, the network already incorporates this concept of parallel links. For example, the power law structure of the AS graph only considers the adjacency of ASes: the link between Sprint and AT&T, for instance, is modeled by a single edge. However, in the real world the Sprint and AT&T ASes are connected to each other in a large number of places around the world. However, not much is known about the degree of such connectivity in the Internet today.

In order to guide the addition of parallel edges between adjacent nodes, we first observe that there is clear correlation between the average degree and edge congestion. Figure 10 plots the congestion of each edge against the average degree of the nodes on which it is incident, for shortest path routing on an Inet generated graph of 30000 nodes. The form of communication used here is any-2-any. The figure shows that edges incident on high degree nodes have much higher congestion than those incident on lower degree nodes. This suggests that *a good choice for the number of parallel links substituting any edge in the graph, could depend on the degrees of nodes which an edge connects.*

We examine several ways of adding parallel links based on the above observation. In particular, we let the number of links between two nodes be some function of the degrees of the two nodes and we consider the following functions: (1) sum of degrees of the two nodes, (2) product of the degrees of the two nodes, (3) maximum of the two degrees and, (4) minimum of the two degrees. For each of these functions, we compute the maximum relative congestion, that is, the maximum over all edges, of the congestion on the edge divided by the number of parallel links corresponding to each edge. In what follows, we show simulation results about how the maximum relative congestion scales for shortest path routing on power law graphs within the any-2-any model of communication.

The results are shown in Figure 11. Notice that, surprisingly, when parallel links are added according to *any* of the above four functions *the maximum relative congestion in the graph scales linearly.* This implies that adding parallelism in the edges of Internet-like graphs according to the above simple functions is enough to ensure that the scaling of link capacities in the Internet according to Moore's law can maintain uniform levels of congestion in the network and avoid any potential hot-spots.

[7]Note that the routing is still done based on the original degrees of nodes.

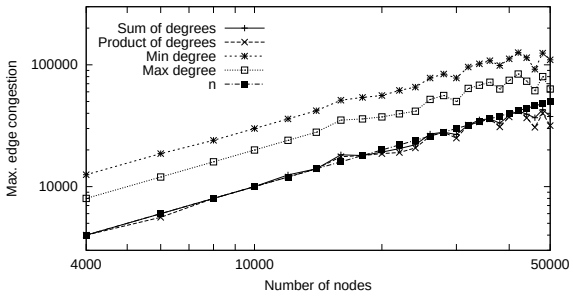


Figure 11: Maximum relative congestion for shortest path routing, any-2-any model, when parallel links are added to the graph using the sum, product and max functions.

6.2 Random Edge Addition

Additionally, various other methods of altering the structure of the graph may also be examined. This paper contains the results of adding parallel network links according to the above mentioned four functions, however two additional methods were conjectured and are currently being tested for the scaling properties they imply. One such method involves adding random edges to the topology. For instance, a percentage of node pairs that are not connected in the current topology would have an edge added between them. This could potentially alleviate congestion due to the fact that some of the stub nodes would not need to route their traffic through the core of the graph to communicate to other stub nodes.

Another model that is being considered is one where edges are added between nodes with high degree. Consider the core of the topology, this is where the highest degree nodes tend to be. We would then like to examine randomly adding edges between two nodes of high degree that are not connected in the original topology. This could help reroute some of the traffic that goes through the core, particularly through the nodes with very high degree. If edges were added between the high-degree nodes, perhaps the traffic could become more distributed leading to a reduction in the maximum edge congestion and better scaling of the worst edge congestion in the topology.

7 Summary

In this paper, we addressed the question of how the worst congestion in Internet-like graphs scales with the graph size. Using extensive simulation studies, we show that the maximum congestion scales poorly in Internet-like power law graphs. Our simulation results show that the non-uniform demand distribution between nodes only exacerbates the congestion scaling. However, we find, surprisingly, that policy routing may not worsen the congestion scaling on power law graphs and might, in fact, be marginally better when compared to shortest-path routing.

Our results show that, with the current trend of the growth of the Internet, some locations in the network might eventually become perpetual hot-spots. Fortunately, however, there is an intuitively simple fix to this problem. Adding parallel links between adjacent nodes in the graph according to simple functions of their degrees will help the maximum congestion in the graph scale linearly. In this case, it might not be necessary for some links in the graph to grow in capacity at a faster rate than the others.

References

[1] BGP Tables from the University of Oregon RouteViews Project. `http://moat.nlanr.net/AS/data`.

[2] National laboratory for applied network research. Routing data. http://moat.nlanr.net/Routing/rawdata/.

[3] Aiello, W., Chung, F., and Lu, L. Random evolution in massive graphs. In *FOCS* (2001), pp. 510–519.

[4] Albert, R., and Barabasi, A.-L. Topology of evolving networks: local events and universality. *Physical Review Letters 85(24)* (2000), 5234–5237.

[5] Aumann, Y., and Rabani, Y. An $o(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing 27(1)* (1998), 291–301.

[6] Barabasi, A.-L., and Albert, R. Emergence of scaling in random networks. *Science 286* (1999), 509–512.

[7] Fabrikant, A., Koutsoupias, E., and Papadimitriou, C. Heuristically optimized trade-offs: A new paradigm for power laws in the Internet. In *ICALP* (2002), pp. 110–122.

[8] Faloutsos, M., Faloutsos, P., and Faloutsos, C. On power-law relationships of the Internet topology. In *Proceedings of the SIGCOMM '99 Symposium on Communications Architectures and Protocols* (1999), pp. 251–262.

[9] Gao, L. On inferring autonomous system relationships in the Internet. *IEEE/ACM Transactions on Networking 9(6)* (December 2001).

[10] Gkantsidis, C., Saberi, A., and Mihail, M. Conductance and congestion in power law graphs. In *Proceedings of the ACM Sigmetrics 2003* (2003).

[11] Leighton, F. T. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, 1992.

[12] Leonardi, S. On-line network routing. In *Online Algorithms - The State of the Art* (1998), Springer, pp. 242–267.

[13] Medina, A., Lakhina, A., Matta, I., and Byers, J. Brite: An approach to universal topology generation. In *MASCOTS* (2001).

[14] Mitchell, T. M. *Machine Learning.* McGraw-Hill Companies, Inc., 1997.

[15] Okamura, H., and Seymour, P. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory B 31* (1981), 75–81.

[16] Park, K., and Lee, H. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *Proceedings of the SIGCOMM '01 Symposium on Communications Architectures and Protocols* (San Diego, California, August 2001).

[17] Subramanian, L., Agarwal, S., Rexford, J., and Katz, R. H. Characterizing the Internet hierarchy from multiple vantage points. In *Proceedings of IEEE INFOCOM* (June 2002).

[18] Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S., and Willinger, W. Network topology generators: Degree-based vs structural. In *Proceedings of the SIGCOMM '02 Symposium on Communications Architectures and Protocols* (Pittsburgh, Pennsylvania, August 2002).

[19] Winick, J., and Jamin, S. Inet-3.0: Internet topology generator. Tech. rep., University of Michigan, 2002. Technical Report, CSE-TR-456-02.