# AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems*

Yunhao Liu, Zhenyun Zhuang, Li Xiao
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
{liuyunha,zhuangz1,lxiao}@cse.msu.edu

Lionel M. Ni
Department of Computer Science
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
ni@cs.ust.hk

*Abstract*- **Peer-to-Peer (P2P) systems are self-organized and decentralized. However, the mechanism of a peer randomly joining and leaving a P2P network causes topology mismatching between the P2P logical overlay network and the physical underlying network. The topology mismatching problem brings great stress on the Internet infrastructure and seriously limits the performance gain from various search or routing techniques. We propose the Adaptive Overlay Topology Optimization (AOTO) technique, an algorithm of building an overlay multicast tree among each source node and its direct logical neighbors so as to alleviate the mismatching problem by choosing closer nodes as logical neighbors, while providing a larger query coverage range. AOTO is scalable and completely distributed in the sense that it does not require global knowledge of the whole overlay network when each node is optimizing the organization of its logical neighbors. The simulation shows that AOTO can effectively solve the mismatching problem and reduce more than 55% of the traffic generated by the P2P system itself.**

## I. INTRODUCTION

Peer-to-peer (P2P) systems have received much attention since the development of Gnutella. The P2P model aims to further utilize the Internet information and resources, complementing the traditional client-server services. P2P systems can be classified into structured and unstructured systems [1]. A major factor to determine the quality and performance of a P2P system is how effective is the searching and locating of information among the peers. Many search techniques have been proposed for structured P2P systems based on hash functions to tightly control file placement (and file locating) with the network topology (e.g., [2]). Although these designs are expected to dramatically improve the search performance, none of them is practically used due to their high maintenance traffic in delivering messages and updating the mapping. Furthermore, it is hard for structured P2P systems to efficiently support partially matched queries.

In an unstructured P2P system, file placement is random, which has no correlation with the network topology. Unstructured P2P systems are most commonly used in today's Internet. An unstructured P2P system floods queries among peers (such as in Gnutella) or among supernodes (such as in KaZaA). This paper is focusing on unstructured P2P systems.

In a P2P system, all participating peers form a P2P network over a physical network. A P2P network is an abstract, logical network called an *overlay network*. When a new peer wants to join a P2P network, a bootstrapping node provides the IP addresses of a list of existing peers in the P2P network. The new peer then tries to connect with these peers. If some attempts succeed, the connected peers will be the new peer's neighbors. Once this peer connects into a P2P network, the new peer will periodically ping the network connections and obtain the IP addresses of some other peers in the network. These IP addresses are cached by this new peer. When a peer leaves the P2P network and then wants to join the P2P network again (no longer the first time), the peer will try to connect to the peers whose IP addresses have already been cached. This mechanism of a peer joining a P2P network and the fact of a peer randomly joining and leaving causes an interesting matching problem between a P2P overlay network topology and the underlying physical network topology.

Figure 1 shows two examples of P2P overlay topology (A, B, and D are three participating peers) and physical topology (nodes A, B, C, and D) mappings, where solid lines denote physical connections and dashes lines denote overlay (logical) connections. Consider the case of a message delivery from peer A to peer B. In the left figure, A and B are both P2P neighbors and physical neighbors. Thus, only one communication is involved. In the right figure, since A and B are not P2P neighbors, A has to send the message to D before forwarding to B. This will involve 5 communications as indicated in Fig. 1. Clearly, such a mapping creates much unnecessary traffic and lengthens the query response time. We refer to this phenomenon as *topology mismatching problem*.

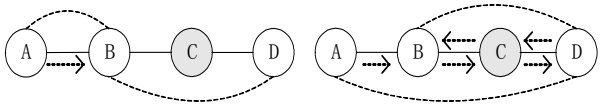


Figure 1: Two examples of P2P overlay networks.

* This work was partially supported by Michigan State University IRGP Grant 41114 and by Hong Kong RGC Grant HKUST6161/03E.

Studies in [3] show that only 2 to 5 percent of Gnutella connections link peers within a single autonomous system (AS). But more than 40 percent of all Gnutella peers are located within the top 10 ASes. This means that most Gnutella-generated traffic crosses AS borders so as to increase topology mismatching costs. The same message can traverse the same physical link multiple times, causing large amount of unnecessary traffic.

In order to reduce unnecessary flooding traffic and improve search performance, two approaches have typically been used to improve from the flooding-based search mechanism. Rather than flooding a query to all neighbors, the first approach routes queries to peers that are likely to have the requested items by some heuristics based on maintained statistic information [4]. In the second approach, a peer keeps indices of other peers' sharing information or caches query responses in hoping that subsequent queries can be satisfied quickly by the cached indices or responses [4,5]. The performance gains of these approaches are also seriously limited by the topology mismatching problem.

The objective of this paper is to minimize the effect due to topology mismatching. We propose the *Adaptive Overlay Topology Optimization* (AOTO) to alleviate the topology mismatching problem. AOTO is scalable and completely distributed in the sense that it does not require global knowledge of the whole overlay network when each node is optimizing the organization of its logical neighbors. Our simulation shows that the average cost of each query to reach the same scope of nodes is reduced by about 55% when using AOTO in a Gnutella-like P2P network without losing any autonomy feature, and the average response time of each query can be reduced by 40%.

## II. ADAPTIVE OVERLAY TOPOLOGY OPTIMIZATION

### A. Inefficient Scenarios

In most flooding-based decentralized P2P networks, such as Limeware (Gnutella), each peer forwards a query message to all of its logical neighbors. Most supernode-based P2P systems, such as KaZaA, also flood queries among supernodes. Figure 2(a) depicts an example of the underlying physical network topology, where the cost of each link is labeled by the link. Let node 1 be the source peer that will send flooding messages to other peers. For simplicity, we only consider total traffic (or cost) generated reaching nodes 2, 3, and 4 on three different P2P overlay topologies as shown in Fig. 2(b), 2(c), and 2(d), respectively. We assume that a node reaches to another node through a shortest physical path based on the link cost (metric). Note that the two shaded nodes in Fig. 2(a) are non-participating nodes in the P2P network.

In Fig. 2(b), nodes 2, 3 and 4 are immediate logical neighbors of node 1. The shortest physical path from node 1 to node 4 is 1 → 5 → 2 → s → 4 with a total cost of 9. Similarly, the costs from node 1 to nodes 2 and 3 are 3 and 15, respectively. Thus, the total cost of flooding a message from node 1 to nodes 2, 3, and 4 is 3+15+9=27. In Fig. 2(c), node 3 is the only immediate logical neighbor of node 1 and nodes 2 and 4 are immediate logical neighbors of node 3. A message will be flooded from node 3 to nodes 2 and 4. The total cost from node 1 to nodes 2, 3, and 4 is 15+12+6=33, which is worse than the case of Fig. 2(b). In Fig 2(d), node 1 can flood the message to all its neighbors, thus nodes 2, 3, and 4. However, node 2 does not know that node 3 will receive the message and will flood the message to node 3 as well. Similarly, node 4 will also flood the message to node 3. Thus, the total cost is 3+15+9+12+6=45.

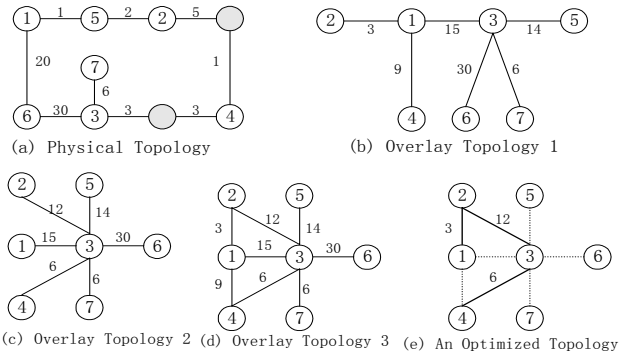


Figure 2: Examples of different overlay topologies.

Clearly, all the three inefficient overlay topologies generate a large amount of unnecessary traffic. Optimizing inefficient overlay topologies can fundamentally improve P2P search efficiency. One attempt is to build an overlay multicast tree among a node and its logical neighbors. In the case of Fig. 2(d), an improved mechanism is shown as thick lines in Fig. 2(e) in which the total cost from node 1 to nodes 2, 3, and 4 is 3+12+6=21. Although the cost is not as low as the optimal IP-level multicast, which is 15, the total cost has already been significantly reduced. This is the motivation that we propose the Adaptive Overlay Topology Optimization (AOTO) technique.

While retaining the desired prevailing unstructured architecture of P2P systems, the goal of AOTO is to dynamically optimize the logical topology to improve the overall performance of P2P systems, which can be measured as *query response time*. AOTO includes two steps: Selective Flooding (SF) and Active Topology (AT). Selective Flooding is to build an overlay multicast tree among each peer and its immediate logical neighbors, and route messages on the tree to reduce flooding traffic without shrinking the search coverage range. Thus, some neighbors become non-flooding neighbors. Active Topology is the second step in AOTO for each peer to independently make optimization on the overlay topology to alleviate topology mismatching problem by replacing non-flooding neighbors with closer nodes as direct logical neighbors.

### B. Selective Flooding

Instead of flooding to all neighbors, SF uses a more efficient flooding strategy to selectively flood a query on an overlay multicast tree. This tree can be formed using a minimum spanning tree algorithm among each peer and its immediate logical neighbors. In order to build the minimum spanning tree, a peer has to know the costs to all its logical neighbors and the costs between any pair of the neighbors. We use network delay between two nodes as a metric for measuring the cost between nodes. We modify the Limewire implementation of Gnutella 0.6 P2P protocol by adding one routing message type. Each peer probes the costs with its immediate logical neighbors and forms a *neighbor cost table*. Two neighboring peers exchange their neighbor cost tables so that a peer can obtain the cost between any pair of its logical neighbors. Thus, a small overlay topology of a source peer and all its logical neighbors is known to the source peer.

Compared with the flooding traffic, the traffic generated in SF due to exchanging neighbor cost tables is insignificant because such exchanges only occur between immediate neighbors. For a branching factor (i.e., average number of direct logical neighbors) of $m$ and TTL (the number of times a message will be forwarded) of $k$, the flooding traffic is O(mN) for each query, where N is the total number of peers, which is typical in the range of millions and m is in the range of tens. The traffic increased due to exchanging neighbor cost tables is O($m$) that is trivial. Based on obtained neighbor cost tables, a minimum spanning tree then can be built by simply using an algorithm like PRIM which has a computation complexity of O($m^2$). Now the message routing strategy of a peer is to select the peers that are the direct neighbors in the multicast tree to send its queries.

In the example of Fig. 2(e), node 1 sends a message only to node 2 and expects that node 2 will forward the message to nodes 3 and 4. Note that in this step, even node 1 does not flood its query message to nodes 3 and 4 any more, node 1 still retains the connections with nodes 3 and 4 and keeps exchanging the neighbor cost tables. We call nodes 3 and 4 *non-flooding neighbors*, which are the peers to be optimized in the next step.

### C. Active Topology

The second step of AOTO, AT, reorganizes the overlay topology. Note that each peer has a neighbor list which is further divided into flooding neighbors and non-flooding neighbors in SF. Each peer also has the neighbor cost tables of all its neighbors. In this step, it tries to replace those physically far away neighbors by physically close by neighbors, thus minimizing the topology mismatching traffic. An efficient method to identify such a candidate peer to replace a far away neighbor is critical to the system performance. Many methods may be proposed. In our approach, a non-flooding neighbor may be replaced by one of the non-flooding neighbor's neighbors. Let $C_{ij}$ represent the cost from peer $i$ to peer $j$. The proposed *Randomized AT algorithm* picks up a candidate peer at random among the non-flooding neighbor's neighbors. The following pseudo code describes the randomized AT algorithm for a given source peer $i$.

Pseudo Code of the Randomized AT Algorithm (peer $i$)

```
For each j in i's non-flooding neighbors
    Replaced = false;
    List = all j's neighbors excluding i;
    While List is not empty and Replaced = false
        randomly pick a peer h from List;
        measure C_ih;
        if C_ih < C_ij {replace j by h in i's neighbor list;
            Replaced = true; remove h from List}
        else if C_ih < C_jh {
            if C_jh is not disconnected
                    add h to i's neighbor list;
            else {remove j from i's neighbor list;
                    Replaced = true; remove h from List}
            };
    End While;
End For;
```

Note that $C_{ij}$ is known to the peer $i$. $C_{jh}$ is also known to the peer $i$ due to the exchange of neighbor cost tables between $i$ and $j$. The cases of $C_{ih} < C_{ij}$ and $C_{ih} \geq C_{jh}$ are quite obvious. Let's explain the case of $C_{ih} < C_{jh}$ using Fig. 2(d), where $i$=1 and $j$=3. Suppose $h$=6. From Fig. 2(d), we have $C_{1,3}$=15 and $C_{3,6}$=30. From Fig 2(a), we can measure $C_{1,6}$=20. When node 1 finds that the cost between nodes 3 and 6 is even larger the cost between node 1 and node 6, node 1 will keep node 6 as a new neighbor. Since the algorithm is executed in each peer independently, node 1 cannot inform node 3 to remove node 6 from node 3's neighbor list. However, as long as node 1 keeps both node 3 and 6 as its logical neighbors, we may expect that node 6 will become a non-flooding neighbor to node 3 after node 3's SF step since node 3 expects node 1 to forward messages to node 6 to reduce unnecessary traffic. Node 3 will try to find another peer to replace node 6 as its neighbor. After knowing that node 6 is no longer a neighbor to node 3 from periodically exchanged neighbor cost tables from node 3 (or from node 6), node 1 will remove node 3 from its neighbor list though node 1 has already stopped sending query messages to node 3 for a period of time since the spanning tree has been built for node 1.

## III. Performance Evaluation

Performance evaluation of the proposed AOTO method is described in this section. Both physical topologies and logical overlay topologies which can accurately reflect the topological properties of real networks in each layer are needed in the simulation study. Previous studies have shown that both large scale Internet physical topologies [6] and P2P overlay topologies [7] follow small world and power law properties. Power law describes the node degree while small

world describes characteristics of path length and clustering coefficient [9]. The study in [6] found that the topologies generated using the AS Model have the properties of small world and power law. BRITE is a topology generation tool that provides the option to generate topologies based on the AS Model. We generate 10 physical topologies each with 5000 nodes. The logical topologies are generated with the number of peers (nodes) ranging from 100 to 2000. For each given number of nodes, we generate logical topologies with average edge connections between 1 and 20.

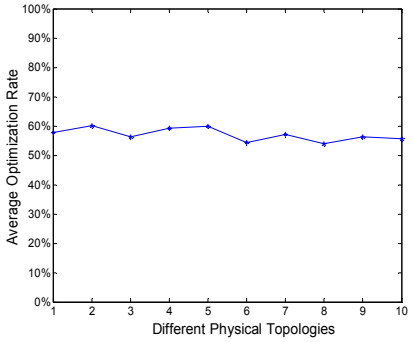

Figure 3: Effect due to different physical topologies for SF.

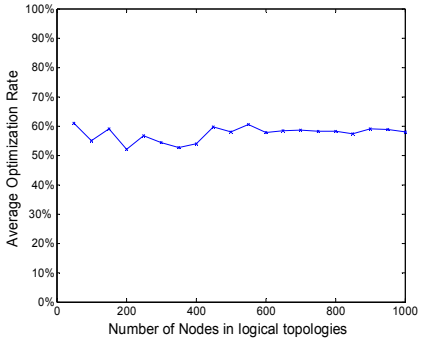

Figure 4: Effect due to different logical topologies for SF.

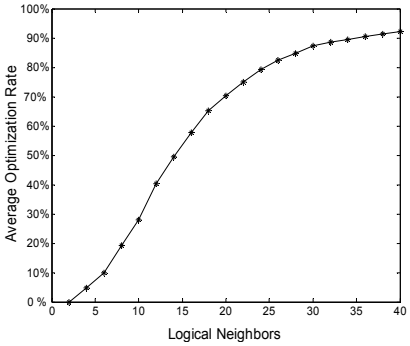

Figure 5: Effect due to different number of logical neighbors

### A. *Performance Metrics*

Let $T_k$ denote the cost of each query from the source to reach all its neighbors, where $T_{-1}$ is the cost based on the traditional blind flooding, $T_0$ is the cost using SF only in the first time, and $T_k$ is the k-th time applying the randomized AT algorithm. Let $\Delta_k = (T_{k-1} - T_k)/T_{k-1} \times 100\%$. Whenever a new neighbor cost table is received or there is a change of neighbors, the source peer has to re-calculate the multicast tree and apply the randomized AT algorithm. In theory, the source peer can continuously do this until no cost improvement is obtained, thus closing to a perfect topology matching. Obviously, this is unnecessary and creates too much overhead. We refer k as the number of *optimization steps*.

During the k-th time applying the AT algorithm, each non-flooding neighbor may be replaced by one of its neighbors. If the source has n non-flooding neighbors, the proposed randomized AT algorithm may have up to n replacements. The overhead to exhaust all n possible replacements may also be too high. In practice, after each replacement, the source peer will compute the cost improvement ratio and decide whether it needs to find another candidate peer to replace another non-flooding neighbor based on a *termination threshold*, $\Delta$. The optimization process will terminate if the improvement ratio is less than $\Delta$. Thus, the value of $\Delta$ is a factor to impact the effectiveness of AT. A smaller threshold causes larger overhead, but produces better optimization results, which will be shown in the next section.

To evaluate the optimization result of a logical topology, we use the metric, *average distance*, D. Let $D_i$ be the average distance between the source peer *i* and all its logical neighbors. The value D is defined as the average of all $D_i$'s (i.e., all peers in the P2P network). The ideal case would be that each node has physically closest P2P nodes as its logical neighbors with at least the same query coverage range. The topology mismatching problem is effectively solved in the ideal case. Since we assume that a peer can always reach its logical neighbors through the shortest physical path, our simulator calculates the physical shortest path for pairs of peers.

We have simulated AOTO for all the generated logical topologies on top of each of the 10 generated physical topologies with 5000 nodes. We have also simulated AOTO in a real-world P2P topology (based on DSS Clip2 trace). We obtained consistent results on the real-world topology and the generated topologies. In order to show a thorough performance discussion, we only present our performance on various generated topologies.

### B. *Effectiveness of Selective Flooding*

To evaluate the effectiveness of SF, we consider three factors. First, how do different physical topologies affect the effectiveness of SF? We compare the average $\Delta_0$'s of a 1000-node logical topology with average 8 edge connections on top of the 10 different physical topologies after selective flooding. Figure 3 shows consistent $\Delta_0$ results ranging from 55% to 60% for different physical topologies. Thus, the physical topology has little impact on the effectiveness of SF.

Second, how do different logical topologies affect the effectiveness of SF? For a given physical topology and a given average edge connection, we compare the average $\Delta_0$'s of SF on logical topologies ranging from 50 to 1000 nodes. The results in Fig. 4 show that the density of P2P nodes does not influence the effectiveness of SF. The average optimization rate, $\Delta_0$, for each topology is around 55%.

Third, how do different numbers of average logical neighbors affect the effectiveness of SF? We compare the average $\Delta_0$'s of 20 500-node logical topologies with different numbers of logical neighbors ranging from 2 to 40. Figure 5 show that SF is more effective with a large number of logical neighbors. For example, SF can achieve $\Delta_0$ as high as 87.4% on a logical topology with an average of 30 logical

neighbors. It is normal that the out-degree of a supernode reaches 30 in many P2P systems [4].

### C. Effectiveness of Active Topology

The average number of logical neighbors is a major factor in the effectiveness of SF, but not of AT. We compare the average reductions of D on three 500-node logical topologies with average 6, 12, and 18 logical neighbors as the number of optimization steps is increased.

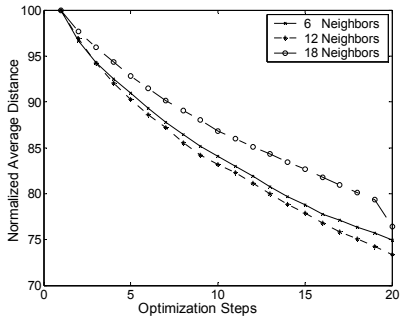


Figure 6: Effect on the number of logical neighbors.

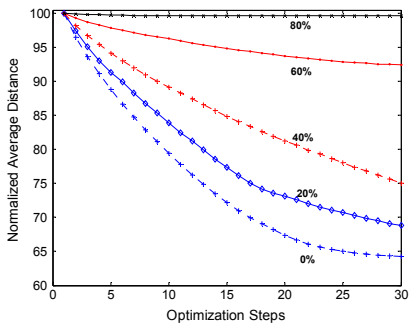


Figure 7: Comparison of different termination thresholds.

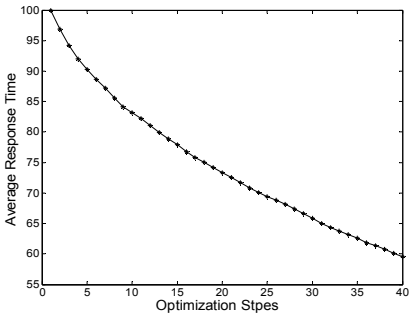


Figure 8: Normalized improvement of average query response time.

Results in Fig. 6 show that the number of logical neighbors has little impact to the effectiveness of AT.

The termination threshold Δ can be defined by each node independently. In above simulations, we have Δ =20% for each node. In order to show the trade-offs between overhead and effectiveness, Fig. 7 plots the normalized average distance D on different thresholds as the number of optimization steps is increased. As expected, D is reduced more slowly for a larger threshold, and a lower threshold leads to a better result.

### D. The Improvement of Query Response Time

Average response time of each query in a P2P system is what a user really cares about. In order to show the overall improvement of AOTO, Fig. 8 shows the normalized average query response times as the number of optimization steps is increased from 1 to 40. Given a source peer, the destination peer is randomly chosen. As shown in Fig. 8, the average query response time is significantly reduced.

## IV. Conclusions and Future Work

This paper aims at alleviating the topology mismatching problem by optimizing the overlay network using our proposed AOTO technique to reduce unnecessary traffic and improve query search efficiency. To the best of our knowledge, the topology mismatching problem has not been adequately addressed. The only related work is Narada [8]. Based on end system multicast, Narada first constructs a rich connected graph on which to further construct shortest path spanning trees. This approach introduces large overhead of forming the graph of trees in a large scope and does not consider peers' dynamic joining and leaving. The overhead of Narada is proportional to the multicast group size.

The proposed AOTO technique is easy to implement and adaptive to the dynamic nature of P2P systems. Furthermore, the overhead of the proposed AOTO algorithm is only proportional to the average number of logical neighbors. AOTO is more effective on logical topologies with large branching factors. It will make the decentralized flooding-based P2P file sharing systems more scalable and efficient.

Due to space limitation, many simulation results and design alternatives are not included in this paper. Instead of the randomized AT algorithm, we have studied the Closest AT algorithm to replace a non-flooding neighbor by its least cost neighbor, which will further optimize the overlay topology with a less number of optimization steps at the expense of more overhead.

More research is needed to study other AOTO policies, the frequency in invoking the AOTO algorithm, the impact due to different multicast tree algorithms, and the impact of implementation overhead.

## References


[1] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. "Search and replication in unstructured peer-to-peer networks," *Proc. of the 16th ACM Int'l Conf. on Supercomputing*, June 2002.

[2] I. Stocia, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications," *Proc. of SIGCOMM 2001*.

[3] M Ripeanu, A. Iamnitchi and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Computing*, January/February 2002, pp. 50-57.

[4] B. Yang and H. Garcia-Molina, "Designing super-peer network," *Proc. of the 22nd Int'l Conf. on Distributed Computing Systems* (ICDCS 2002), Vienna, Austria, July 2002.

[5] K. Sripanidkulchai, B. Maggs and H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," *Proc. of INFOCOM 2003*.

[6] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network Topology Generators: Degree-Based vs. Structural," *Proc. of SIGCOMM'02*, August 2002.

[7] S. Saroiu, P.K. Gummadi, and S. D. Gribble, “A measurement study of peer-to-peer file sharing systems,” *Proc. of Multimedia Computing and Networking 2002*, January 2002.

[8] Y. Chu, S. G. Rao and H. Zhang, “A Case For End System Multicast," *Proc. of ACM SIGMETRICS*, Santa Clara, CA, June 2000, pp. 1-12.

[9] T. Bu and D. Towsley, “On Distinguishing between Internet power law topology generators,” *Proceedings of INFOCOM 2002.*