

Title of the paper: Towards A Peer-To-Peer Simulator

Contact Author:

Dwight Deugo
The School of Computer Science
Carleton University
1125 Colonel By Drive,
Ottawa, Ontario,
K1S 5B6 Canada
Email: deugo@scs.carleton.ca
Office: (613)-520-2600 ext. 8438

Presenting Author

Jon Harris
The School of Computer Science
Carleton University
1125 Colonel By Drive,
Ottawa, Ontario,
K1S 5B6 Canada
Email: jbharris@magma.ca

Keywords: Peer-to-Peer, Simulation, Topology Generation

Towards A Peer-To-Peer Simulator

Jon Harris, Dwight Deugo

The School of Computer Science, Carleton University, 1125 Colonel By Drive., Ottawa, Ontario,
K1S 5B6 Canada, jbharris@magma.ca, deugo@scs.carleton.ca

Abstract— We are in the process of building a large-scale peer-to-peer simulator. In this paper, we discuss common aspects of existing peer-to-peer systems that required our attention in the design of our simulator. We provide details on peer-to-peer classification, network topologies, evolution, event handling and communication. We also describe features that figured prominently in the design of our simulator.

I. INTRODUCTION

As with peers in real life, peers in a community must speak a common language or protocol in order to communicate amongst each other. Moreover, understanding peer-to-peer protocols is of critical importance if you wish to accurately simulate them, compare them and use them.

We are in the process of building a large-scale peer-to-peer simulator. In this paper, we discuss common aspects of existing peer-to-peer systems that required our attention in the design of our simulator.

A real world peer-to-peer system or network is made up of a collection of peers that interact with each other through an underlying infrastructure called a network topology. Real world peer-to-peer networks are dynamic in nature. The structure of these networks is constantly evolving as new peers join or leave the network, forging or breaking links between peers. In order to simulate a peer-to-peer network, it is important to use an accurate and realistic representation of the network topology and its evolution.

Peer-to-peer protocols define the language that is spoken in conversations between peers, and the network topology provides the details of whom a peer may converse with. What are missing are the details about how the conversations get sent across the network and how these events are processed.

In the following sections, we provide more details on each of the above aspects. The classification sections gives the reader some background about different peer-to-peer protocol schemes. The network topology section describes issues related to accurately and realistically representing the network topologies underlying real world peer-to-peer communities. The evolution section defines how peers join or leave the network, forging or breaking links between other peers. The communication sections describes the issues involved in simulating the communication that occurs over a network topology. The events section describes methods for generating and propogating events through the simulator. The last section provides our conclusions.

Our goal by this discussion is to generate debate and consolidate those aspects that are essential ingredients in any peer-to-peer system and critical to the features, design and implementation of any peer-to-peer simulator.

II. CLASSIFICATION

Peer-to-peer protocols can be classified in a variety of ways. The main classifications are centralized or decentralized and structured or unstructured. Most structured or unstructured protocols fall into the decentralized classification, although there is no hard and fast rule that states that this must be the case.

A. Centralized or Decentralized

Peer-to-peer protocols can be centralized or decentralized [Ora01]. Centralized or more specifically centrally coordinated systems, such as instant messaging networks and Napster, make use of some form of central server that acts as a broker between peers. In decentralized systems, no such intermediary exists.

B. Structure or Unstructured

There are two different types of decentralized peer-to-peer protocols, Structured and Unstructured protocols [CRB03]. Structured protocols are often called Distributed Hash Table protocols or Second Generation protocols. Unstructured protocols are often called Broadcast Flooding protocols or First Generation protocols.

Distributed Hash Table (DHT) or structured protocols such as Chord or Pastry use the idea that the realm of searchable data can be split up into roughly equal sized pieces. Each document is mapped to a key, and each piece of the search space is uniquely related to a range of keys in a manner similar to conventional hash tables.

Broadcast Flooding (BF) or unstructured protocols such as Gnutella or Fasttrack are much more relaxed about the organization of searchable data. Searching with a BF is almost haphazard; each peer typically forwards a search to each of its neighbours, if they have not already seen it. Some BF protocols use a selective broadcast approach in an attempt to give peers some better knowledge about peers to whom a search should be forwarded.

DHT protocols are often touted for their efficient search capabilities, and BF protocols for their ability to adapt efficiently to changes in the structure of the peer community. Chawathe et al. [CRB03] relate DHT and BF protocols analogously to the needle and a haystack scenario. DHT protocols excel at always finding any file that exists on the network, provided you know exactly what you are looking for (finding a needle). BF protocols on the other hand, cannot always guarantee finding a specific file, yet most users typically search for files that are well replicated across the network, and thus easily found (the hay). Based on empirical data observed from real peer to peer networks, Chawathe et al.

conclude that most users are ‘looking for the hay, not the needle’, making the use of BF protocols a more viable approach for certain applications.

III. NETWORK TOPOLOGY

A real world peer-to-peer system or network is made up of a collection of peers which interact with each other through an underlying infrastructure called a network topology. In order to simulate a peer-to-peer network, it is important to use an accurate and realistic representation of the network topology. This section describes the different data structures that can be used to represent the network topology. Later on in this section, some of the known structural properties of real-world peer-to-peer network topologies are described, followed by a description of some of the methods for generating network topologies that exhibit these properties.

A. Representation

In order to simulate a peer-to-peer network, it is necessary to somehow represent this network topology inside the simulator. The standard approach for representing network topologies is to make use of the mathematical discipline of graph theory. In graph theory, a graph is defined as a collection of points or vertices (nodes) and lines connecting these points (edges). In the context of peer-to-peer or other networks, peers or computers in the network can be represented by nodes, and the communication links among them can be represented by edges. [Ora01].

Two popular data structures that exist for representing graphs are The Edge-List and that Adjacency Matrix .

1) Edge-List Data Structure

As its name implies, the Edge-List (sometimes called an Adjacency-List or Incidence-List) data structure represents the graph using lists of edges for each node. Each of the N nodes in the graph maintains a list, containing references to each of the edges connected to that node. Fast edge insertions and removals can be achieved with this data structure through the use of doubly linked lists.

2) Adjacency Matrix Data Structure

For the Adjacency Matrix data structure, an $N \times N$ matrix is used to store the M edges between the N nodes. Each node is indexed to both a row and column of the matrix. For example, if node 4 has an (undirected) edge with node 7, then the appropriate data (usually a Boolean flag) is placed in the matrix at row 4, column 7 and row 7, column 4. Node insertions and deletions are costly in this structure since the entire matrix must be copied in order to insert or delete rows and columns.

3) Comparison

The Edge-List is the more popular of these two data structures largely due to its more efficient storage. Node insertions and deletions also can be performed much more efficiently with the Edge-List as illustrated by the Table 1.

It is important to note that some protocols require custom data structures for representing the network topology. In most cases, this will be an extension of the Edge-List data structure.

Table 1 Comparison of run times of operations on Edge-List versus Adjacency Matrix

Operation	Edge-List	Adjacency Matrix
Storage	$O(N+M)$	$O(N^2)$
Node insertion time	$O(1)$	$O(N^2)$
Node deletion time	$O(1)$	$O(N^2)$
Edge finding time for a node with E edges. (other end node provided)	$O(E)$	$O(1)$
Edge insertion time	$O(1)$	$O(1)$
Edge deletion time for a node with E edges.	$O(1)$ + edge finding time if necessary	$O(1)$

B. Structural Models

The recent rise in popularity of peer-to-peer networks and the internet has motivated researchers from backgrounds ranging from mathematics to the social sciences to study the structural properties of the network topologies that these networks form. The models devised by these researchers provide valuable insight for simulating or approximating real networks.

1) Small World Networks

Researchers in the peer-to-peer field have observed that in practice, peer-to-peer networks tend to form Small World Networks [Ora01]. Small world networks is a term coined by Watts and Strogatz in their paper Collective dynamics of ‘small-world’ networks [WS98]. The authors determined that there are two measurements of structural properties of these types of graphs that are particularly interesting, the characteristic path length, and the clustering coefficient. The characteristic path length is defined by Watts and Strogatz as ‘the number of edges in the shortest path between two vertices, averaged over all pairs of vertices’. The clustering coefficient is a measure of the average ‘cliquishness’ of all the vertices. Cliquishness is measured for a vertex v with k neighbours by calculating how many edges exist between all k vertices in the neighbourhood, divided by the total possible number of edges between vertices in the neighbourhood. The total possible number of edges in a neighbourhood of k vertices is $k(k-1)/2$ when every vertex is connected to every other. Watts and Strogatz define small world graphs as graphs which exhibit a low characteristic path length, and high clustering coefficient.

2) Power Law Networks

Researchers in the peer-to-peer field have also noticed that peer-to-peer networks tend to form power law networks. The line between small world and power law networks is grey as opposed to black and white. Graphs can fall into any combination of these categories (both, one or other, or none).

The first connection between internet topologies and power law networks was made by Faloutsos et al. in their

paper On Power-Law Relationships of the Internet Topology [FFF99]. A power law graph (network) is one in which the majority of nodes have small degree, and only a few nodes have high degree (The degree of a node is the number of edges connected to it.).

More formally, the fraction of nodes N in a power law random graph with degree greater than D can be expressed by the function $N = D^{-k}$, where k is a network specific constant. It has been observed in practice that the value of k is typically between the values of 0.5 and 4 [BT02] [BA99]. Such Power law functions are recognizable by the appearance of a straight line when N is plotted versus D on a log-log scale. In [FFF99] they also show that the number of edges in a power law graph is linear in the number of nodes.

It should be noted that there has been some debate as to whether peer-to-peer networks form power law networks. For example Mihajlo et al. write: "Upon analysing the Gnutella topology data obtained using our network crawler, we discovered it obeys all ... these power-laws" [JAB01]. While Ripneau et al. using their own Gnutella crawler observe that "there are too few nodes with low connectivity to form a pure power law network." [RFI02].

Power law networks are often criticized for their vulnerability to malicious attacks on highly connected nodes, yet fare much better in the face of random node failures. Saroiu et al. observe from their Gnutella crawler data that "the network fragments only when 60% of the nodes breakdown", but also mention that "This removal of 5% of the more highly connected nodes has effectively shattered the overlay into a large number of disconnected pieces" [SGG02].

C. Topology Generators

In their paper [TGJ+01], Tanguminarunkit et al. propose grouping topology generators into three families: random graph generators, structural generators and degree-based generators. Random graph generators are the simplest and oldest family of generators. Structural generators are the next oldest family group, and generators from this family focus on attempting to create a hierarchical topology structure. Degree-based generators are typically newer, and generators from this family focus on the degree distribution of nodes in the generated topology.

1) Random Graph Generators

Primitive Random graph generators generate graphs by assigning a uniform probability for creating a link between any pair of nodes. The Waxman generator is perhaps the most well known random graph generator. The Waxman generator is fairly simple, nodes are assigned randomly to positions in the plane, and the probability of connecting any two pairs of nodes is proportional to their Euclidean distance from each other. [WAX88]. Several variations of the Waxman generator have been developed which make slight alterations to the probability distribution [ZCB96].

2) Structure-Based Generators

Structure-Based Generators attempt to enforce a rigid structure or hierarchy onto the graphs they generate. Two models have been proposed for developing structure-based

generators; the Transit-Stub model [ZCB96] and the Tiers model [Doa96].

The Transit-Stub model uses the idea of constructing portions of the graph randomly and then connecting these portions together in a fashion that mimics the structure of the internet. More specifically, the internet can be viewed as a collection of routing domains, where most traffic between two nodes in a domain stays inside that domain. The Transit-Stub model emulates this by classifying portions of the generated graph as transit domains (no restrictions) or stub domains (every path between any pair of nodes must be internal).

Generating graphs using the Transit-Stub model involves generating a random connected graph to represent the connections between transit domains. Each node in this graph is then replaced by a random connected graph which makes up the content of the transit domain. The next step is to generate a number of random connected graphs (stub domains) for connection to each transit domain node. In a final step, a number of extra links are created between transit domains and stub domains or between stub domains.

The Tiers model is similar to the Transit Stub model. In fact, the authors of both the Transit-Stub and Tiers models compare their models in [CDZ97]. The Tiers model replaces the transit and stub domains with 3 levels: WAN, MAN and LAN¹. The Tiers model also employs a minimum spanning tree algorithm to connect the nodes in sub-graphs and nodes at the LAN level are organized into a star topology. In the final step, 'redundant' edges are added according to a distance metric instead of randomly.

3) Degree-Based Generators

Several models have been proposed for degree-based or power law topology generators, many of which are summarized by Bu et al. [BT02] and Tangmunarunkit et al. [TGJ+01]. The two main types of models are the growth model and the distribution model.

Power law topology generators using the growth model typically start off with a small graph, and then grow this graph incrementally until the desired number of nodes is reached. The basic ideas behind this model are those of incremental growth and preferential connectivity.

Incremental growth refers to graphs that are formed by the continual addition of new nodes [MMB00]. At each step, a probability function is used to determine which node(s) a new node will connect to. There are several variations of incremental growth which specify different rules for how many edges to form with the new node, and where to connect them. In some variations of the model, edge addition or rewiring can occur independently of node additions [BA99] [AB00].

Preferential connectivity refers to the preference of new nodes to be connected to existing nodes which are highly connected (have high degree) or popular [MMB00]. Variations on preferential connectivity include schemes for incorporating distances between nodes as part of the probability function.

Power law topology generators using the distribution model differ from those using the growth model in that all of

¹ Wide, Metropolitan and Local Area Networks

the nodes have pre-defined degrees (drawn from a power law distribution) and are present from the graph from the start. Building the graph is now a question of how to distribute the edges to the nodes in order to match their degrees.

One approach used by Aiello et al. [ACL00] is to make x copies of each node, where x is the degree of each node, and then compute and use a matching of this set in order to determine which nodes to connect with edges. The disadvantage of this approach is that the generated graph may contain duplicate or redundant edges, and the graph is not guaranteed to be connected.

4) *Small World Generators*

Small World networks are often lumped into the same category as power law networks, but the generation of small world networks does not seem to fit into any of the previous families of generators. It is interesting to note that Bu and Towsley use the characteristic path length and clustering coefficients as ‘metrics for distinguishing between power law graphs produced by different topology generators’ [BT02]. For completeness, this section will describe one technique for generating small world graphs.

In order to bridge the gap between regular and random graphs, Watts and Strogatz [WS98] propose a technique for randomly ‘rewiring’ a regular (ring lattice) graph. Each edge in the regular graph is rewired at random with probability p ($0 < p < 1$). Rewiring an edge at random involves randomly selecting at least one new end node for that edge. This technique was modified by the authors in a later work [NW99] to eliminate the possibility of creating disconnected graphs. To this end, the rewiring procedure was modified so that edges are added instead of moved.

Regular graphs typically have long characteristic path lengths and low clustering coefficients, and vice versa for random graphs. Watts and Strogatz noticed that sampling the probability p in the range between 0 and 1, they obtained (small world) graphs that were ‘highly clustered like a regular graph, yet with small characteristic path length, like a random graph’.

IV. EVOLUTION

Real world peer-to-peer networks are dynamic in nature. The structure of these networks is constantly evolving as new users join or leave the network, forging or breaking links between the peers. The following sections describe the two scenarios that cause the network topology to evolve: Peer arrivals and departures, and Peer discovery.

A. *Peer Arrivals and Departures*

Peers are typically free to join or exit the community at any time. This corresponds in real life to users opening or exiting a peer-to-peer application, or rebooting or powering off their computers. Several events (computer or network errors) can even cause users to disconnect unintentionally.

An accurate peer-to-peer simulator must provide mechanisms for representing this form of peer behaviour. This can be accomplished via node insert/delete operations on the corresponding graph. A recent measurement study of the Gnutella network showing that most users connect to the

network for a period of less than an hour [SGG02], emphasizes the need to make the implementation of these operations as efficient as possible in the simulator.

During a simulation run, we should not be solely concerned with modeling the rate of peer arrivals and departures, but also with how they select a connected peer with which to make first contact. Ideally, node insert operations should be orchestrated so as to preserve the existing structural properties of the graph (eg. power law). However most existing peer-to-peer protocols make use of a host cache containing a fixed list of peers with which to connect.

B. *Peer Discovery*

Peers cannot form a community network without some knowledge of each other’s presence. In real peer-to-peer applications, each peer is aware of a (small) subset of peers from the community, referred to as that peer’s neighbourhood. Changes to a peer’s neighbourhood can occur for a variety of reasons, such as new peers joining or leaving the network, peers contacting another peer during a search, periodic search for new or faster peers to connect to (a.k.a. topology adaptation) or periodic pruning of unresponsive neighbours.

A peer’s neighbourhood is analogous to the set of edges stored by its node in the graph. Over time, peers can ‘discover’ new peers or drop existing peers from their neighbourhood. The specifics of this type of behaviour can vary, depending on which peer-to-peer protocol is in use.

Peer discovery or dropping of peers from a peer’s neighbourhood can be represented in a simulator via edge insertion/deletion operations on the corresponding graph. A higher frequency of changes to peers’ neighbourhoods than of arrivals or departures of peers from the network, would suggest that more priority should be given to the efficiency of edge operations than to node operations.

V. COMMUNICATION

Peer-to-peer protocols define the language that is spoken in conversations between peers, and the network topology provides the details of whom a peer may converse with. What are missing are the details about how the conversations get sent across the network.

A. *Overlay Network*

Peer-to-peer networks on the internet form what is called an overlay network [Ora01]. An overlay network is a network which is overlaid on top of another network, essentially forming a subset of this network. At the internet level, a link connecting two nodes on the internet may pass through several other nodes, and many such links or paths may exist between two specific nodes. In an overlay network, these two nodes appear to be connected directly by one link. In reality, any communication over this single link will follow one or many of the paths available between the two nodes at the internet level.

This design means that the speed of communicating a message between two nodes in an overlay network is dependant to a large extent on the amount of message traffic on the underlying network (the internet). At any given time, the underlying network may be in use by other protocols, such

as for browsing the World Wide Web or downloading files. This type of activity can be simulated by periodically increasing or decreasing the bandwidth (see the next section) of certain links of the overlay.

For the purposes of accurately simulating a peer-to-peer system, the question is whether the goal is to measure the absolute or relative performance of different protocols [CRB03]. In most cases, the primary concern is to measure the relative performance of protocols, and as such modeling the background traffic on a network overlay is not necessary.

B. Bandwidth Modeling

In real communication networks, we must define the concept of the bandwidth [Ora01] of connections between any two nodes. If we imagine that the underlying network topology is a system of interconnected water pipes, then the bandwidth is the width of these pipes. The bandwidth of a pipe determines the maximum amount of water (data) that can pass through it at any given moment in time, which in turn determines how long water (data) takes to get from one end to the other. The available bandwidth of a pipe corresponds to how much room is left in the pipe when some water is already flowing through it. When pipes are interconnected in sequence, the amount of water that can be channelled from one end to the other is limited by the narrowest (least available bandwidth) pipe. In communication networks, these ‘narrow pipes’ that limit the flow of water are called bottleneck links [GB02].

A typical approach to representing the bandwidth available to inter-node links is to associate a maximum and available value with the corresponding edge in the network topology representation. Initializing the representation with maximum bandwidths can be done at random, or can be biased towards certain capacities. In today’s world, this might correspond to allocating a certain percentage of the bandwidths to model dial-up users and another percentage items such as cable modem/DSL users.

In the context of an overlay network, the bandwidth available for a connection between two nodes corresponds to the sum of the bottleneck bandwidths of each of the distinct paths between the same two nodes in the underlying network. Deciding whether or not to account for this added complexity in modeling bandwidth is again a question of whether absolute or relative performance of a protocol is to be measured. It should be noted that the extra storage and processing required for such detailed bandwidth modeling may present a significant barrier to the scalability of the simulator.

During simulation, consumption of bandwidth occurs as nodes progressively engage in more and more communication. The time for each communication to be completed can be determined by the amount of available bandwidth. When communication begins the available bandwidth on a link is decremented. When the communication is complete, the available bandwidth is incremented. The trick is to share the available bandwidth fairly when more than one message is being communicated.

C. Transporting Messages

On the internet, messages between nodes (peers) are transported using network sockets. Network sockets are

essentially channels of communication that are opened between two nodes, using the TCP/IP protocol. There is a large amount of overhead associated with the use of sockets, and as such their use makes little sense in the context of a simulation where all the peers reside on one computer. The only exception might be if the goal is to build a simulator that is also deployable as a real application.

In simulation, the typical approach for transporting messages between nodes is to use simple method or function calls. This simple approach can be extended by using Remote Method Invocations (RMI) or other similar standards for simulators in distributed, multi-threaded or parallel settings.

D. Communication Errors

On the internet or other similar networks, communication errors can occur at any time, greatly impacting the time taken to transport messages. In some extreme cases, communication errors may actually prevent a message from being transported at all.

When simulating peer-to-peer protocols, it is important to evaluate their resilience in the face of communication delay or failure. Communication delay on a link between peers is relatively easy to simulate by simply increasing the completion time of the relevant message. Critical communication errors can be simulated by simply suppressing a message while it is in transit.

E. CPU Delay

Some scenarios require processing a message before or after its communication between nodes. For example, a peer receiving a search message may require time to check if it has the matching document, or to search through its routing table to find the appropriate peer to forward the search to. Another example is the encryption and decryption of messages before and after its transmission. Encryption can be particularly computationally expensive, resulting in lengthy delays.

Such delays incurred by the processing speed of the CPU of a node on the network can also be simulated by increasing the time to completion of the appropriate message.

F. File Transfers

The transfer of files between nodes on a network is usually the lengthiest form of communication that can occur. As such, this form of communication is typically most prone to failure of some form. Deletion of the file by the transmitter or cancellation by the receiver can also cause the communication to be terminated. In some cases, files can be transferred simultaneously from multiple sources providing extra resilience, yet requiring careful management. The time taken to transfer a file between one peer and another can be simulated proportionally to the bandwidth of the link(s) it traverses.

VI. EVENTS

At the heart of any simulator is a mechanism for generating events and propagating them through the entities that form the test data set.

In the peer-to-peer context, there is a need to generate the number of peer arrivals, peer departures and query requests that occur at any point in (simulated) time. To this end, the

logical approach is to use a Poisson Process to generate a Poisson distribution of event times [BNC00].

A Poisson process with rate λ allows us to determine the number of random events that will occur in a fixed time interval of length t . The number of random events per time interval will be biased towards the value λt . A convenient feature of generating Poisson distributions is that they have the ‘memory-less’ property, that is that the numbers of events in separate time intervals are independent of each other.

In the field of discrete-event simulation, there are two principle approaches for scheduling and processing events, *event scheduling* and *activity scanning* [BNC00].

Event scheduling is the predominant approach; it is based on the concept of maintaining a *future event list* (FEL) which stores all the events that are scheduled to occur in the future [BNC00]. All events in the future event list are arranged chronologically by time. While some events start and end at the same time, other events denote the beginning of an activity (such as downloading a file) which will end at some time in the future. The processing of an event which starts an activity requires placing an event which marks the end of the activity at the appropriate (future) time in the future event list.

Event scheduling continually advances a simulation clock to the time of the event at the front of the FEL. The event(s) whose times match that of the simulation clock are removed from the FEL, and executed. This usually results in new events being placed near the end of the FEL.

An important issue when using the event scheduling approach is efficiently maintaining the sorted order of the FEL. A chronologically sorted FEL allows the next imminent event to be found quickly, but may require scanning the entire list when adding, deleting or rescheduling the time of events. A variety of data structures have been suggested for the FEL such as arrays, linked lists or binary heaps. Binary heaps offer the best average performance as the FEL grows large.

Activity scanning is the event handling approach which focuses on the activities or events of a model and the conditions which must be met for them to occur. At each time step, all of the entities (i.e. peers) in the simulation model are scanned to check if they satisfy the conditions for the occurrence of an event, in which case that event is executed. For simulations involving a high number of peers, this may pose serious barriers to scalability. In contrast to the event scheduling world view which uses a variable time increment, the activity scanning world view “uses a fixed time increment and a rule-based approach to decide whether any activities can begin at each point in time” [BNC00].

It should be noted that with the activity scanning world view, the execution of an event for an entity may change the conditions of other entities, requiring repeated scanning of these entities. To combat this inefficiency, some hybrid approaches have been developed which borrow some properties of the event scheduling approach to reduce the number of scanning passes.

VII. CONCLUSION

An understanding of peer-to-peer classification, network topologies, topology evolution, event handling and communication is important for the successful design and

implementation of a peer-to-peer simulator. Equally important is a review and comparison of existing peer-to-peer simulators which we are currently undertaking, and plan to publish in the near future. In this paper, we have described items from each of these aspects that featured prominently in the design of our simulator. Our goal is to provide the reader with items that were important for us, so that when the reader attempts to construct their own simulator, he or she can use our focus as a starting point.

REFERENCES

- [AB00] R. Albert and A.-L. Barabasi. Topology of Evolving Networks: Local Events and Universality. *Physical Review Letters*, 85(24):5234-5237, 2000.
- [ACL00] W. Aiello, F. Chung and L. Lu, A random graph model for massive graphs, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, (2000) 171-180.
- [BA99] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, October 1999.
- [BNC00] Jerry Banks, Barry Nelson, and John Carson. *Discrete-Event System Simulation*. 3rd Edition, Prentice Hall, 2000.
- [BT02] Tian Bu and Don Towsley. On Distinguishing between Internet Power Law Topology Generators, *Proceedings of IEEE INFOCOM '02*, 2002.
- [CRB03] Yati Chawathe, Sylvia Ratnasamy, Lee Breslau, “Making Gnutella-like P2P Systems Scalable”, *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*, p. 407-418, 2003.
- [CDZ97] K. L. Calvert, M. B. Doar, and E. W. Zegura. "Modeling Internet Topology." *IEEE Communications Magazine* 35, 6 June 1997.
- [Doa96] M. Doar, "A Better Model for Generating Test Networks," in *Proceeding of IEEE Global Telecommunications Conference (GLOBECOM)*, November 1996.
- [FFF99] Michalis Faloutsos, Petros Faloutsos and Christos Faloutsos. On Power-Law Relationships of the Internet Topology. In *SIGCOMM*, pages 251-262, 1999.
- [GB02] TJ Giuli and M. Baker. Narses: A Scalable, Flow-Based Network Simulator. Technical Report cs.PF/0211024, Computer Science Department, Stanford University, Stanford, CA, USA, Nov 2002.
- [JAB01] M. Jovanović, F. Annexstein, and K. Berman: Modeling peer-to-peer network topologies through “small-world” models and power laws, *IX Telecommunications Forum TELFOR 2001*, Belgrade.
- [MMB00] A. Medina, I. Matta, and J. Byers. On the Origin of Power Laws in Internet Topologies. *Computer Communication Review*, 30(2):18–28, 2000.
- [NW99] M. E. J. Newman and D. J. Watts. *Renormalization group analysis of the small-world network model*. *Phys. Lett. A* 263 (1999) 341.
- [Ora01] Andy Oram, editor. *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O’reilly, Sebastopol, CA, March 2001.
- [RFI02] M. Ripneau, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6(1), January/February 2002.
- [SGG02] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*
- [TGJ+01] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topologies, power laws and hierarchy, *Tech. Rep. TR01-746*, Technical Report, University of Southern California, 2001.
- [WS98] D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440-442, June 1998.
- [Wax88] B. M. Waxman. Routing of Multipoint Connections. *IEEE Journal of Selected Areas in Communication* 6, 9 (December 1988) 1617-1622.
- [ZCB96] Ellen W. Zegura, Keneth L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of the INFOCOM '96*, 1996. 18