An Abstract Internet Topology Model for Simulating Peer-to-Peer Content Distribution

Ryan LaFortune
Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180, U.S.A.
email: {laforr, chrisc}@cs.rpi.edu

William D. Smith
Michael Hartman
Global Research Center
General Electric
Niskayuna, NY 12309, U.S.A.
email: {smithwd, hartman}@research.ge.com

Abstract

In recent years, many researchers have run simulations of the Internet. The Internet's inherent heterogeneity and constantly changing nature make it difficult to construct a realistic, yet computationally feasible model. In the construction of any model, one must take into consideration flexibility, accuracy, required resources, execution time, and realism. In this paper, we discuss the methodology and creation of a model used to simulate Internet content distribution, and the rationale used behind its design. In particular, we are interested in modeling the in-home consumer broadband Internet, while preserving geographical market relationships. In our performance study, our simulations experience tremendous speedups, and require a fraction of the memory of other models, without sacrificing the accuracy of our findings. Specifically, our piece-level model achieves the accuracy of a packet-level model, while requiring the processing of 40 times fewer events.

1 Introduction

File-sharing using peer-to-peer (P2P) protocols such as BitTorrent [4] has become the "killer application" for the consumer broadband Internet. CacheLogic's [15] monitoring of tier 1 and 2 Internet Service Providers (ISPs) in June of 2004 reports that between 50% and 80% of all traffic is attributed to P2P file-sharing. In 2005, those numbers appear to have been holding steady at 60% of all network traffic on the reporting consumer-oriented ISPs. Consequently, the impact of the P2P movement on the Internet has been quite staggering, and broadband ISPs have not been pleased with this rogue utilization of their network resources because the cost to them is prohibitive – *on the order of $1 billion U.S. dollars* [15], and the ISPs are not making any additional revenue from these network intensive applications. Thus, there is a need to design better, more ISP-friendly P2P protocols. To meet this design goal, models need to be created of both the P2P protocols and the networks they use. For these applications, most of the network traffic and bottlenecks exist in the in-home consumer broadband Internet. *With the key assumption that such traffic has a negligible impact on the core of the Internet [27], our model is designed to not only provide results from peers' perspectives, but also allow us to analyze how the traffic affects the "last-mile" of a particular ISP's network.*

In this paper, we discuss the methodology and creation of an Internet topology model for use in simulating large-scale Internet content distribution. The specific P2P protocol in our study is BitTorrent. Our choice to examine the BitTorrent protocol as opposed to others such as eDonkey2k [7] is because BitTorrent's design appears to be the most amenable approach for legal content distribution. Unlike other P2P protocols, BitTorrent's tracker is a central point that allows content owners to ensure that only legitimate content is served in the distribution network.

Our topology model is comprised of several components that are largely independent of each other. The components include: the Internet connectivity model (Section 3), the population model (Section 4), the delay model (Section 5), the technology model (Section 6), and the bandwidth model (Section 7).

In the design of these models, many decisions were made whether or not to include certain features. Considered in this process are the model's overall realism, its accuracy, the data collection and maintenance required, the execution time of the simulations, and the required system resources. In some cases, a model can be unnecessarily complex, and produce results that either cannot be analyzed, or are no better than those of a simpler version [12]. In Section 8, we demonstrate the efficacy of our model, and discuss some of the benefits that we reap as a result of our decisions.

We now give a brief description of the BitTorrent protocol, our simulator, and prior work that has been done in this area.

2 BitTorrent

BitTorrent is a robust P2P protocol that takes advantage of peers' bandwidth to efficiently replicate content. It is a scalable mechanism that has been successful at distributing large files quickly and efficiently without overwhelming the capacity of the origin servers [2]. There are many BitTorrent clients in existence, and the implementations vary greatly [17]. In this description, we consider the open source *mainline* client developed by Bram Cohen, the protocol's inventor.

2.1 BitTorrent Preliminaries

As the focus of this paper is the Internet topology model, we will just briefly describe the BitTorrent protocol. Please note that any settings specified hereinafter are default settings of the mainline client. Also, many "behind the scenes" details from the implementation have been purposely left out in order to focus on, and simplify, the main ideas of the protocol.

In BitTorrent, files transferred are split into 256 KB *pieces*. These pieces are further divided into *blocks* (blocks are the transmission unit of the network). A *torrent* is a transfer session of one or many files, and is active if it contains at least one *seed* (a peer possessing all files). A peer still downloading content is called a *leecher*. Users can join an existing torrent by downloading a *.torrent* file from a web server. This server stores file information, including SHA-1 hash values for each piece (to check their integrity), as well as the IP address of the torrent's *tracker*. The tracker is not involved in the actual distribution of the files, but instead is a centralized system that keeps track of peers involved in that particular torrent, and statistics relating to it. A tracker can provide peer-set services to a number of torrents [7].

When a user joins the torrent, it contacts the tracker to obtain a list of IP addresses of 50 random peers, forming its initial *peer-set* (the peers it can potentially transfer data to). If the cardinality of this set ever drops below 20 (due to disconnects), the local peer once again contacts the tracker for a new list of IP addresses. The maximum size of the peer-set is 80, allowing for connections that are initiated by remote peers (there are also restrictions regarding the number of open connections). Whether or not pieces are transferred depends on piece interest, how rare pieces are, and a concept of choking [7].

Every peer in the torrent has knowledge regarding which pieces each peer in its peer-set have. We say that peer X is *interested* in peer Y, if Y has at least one piece that X does not have. Conversely, we say that peer X is *not interested* in peer Y, if X has every piece that Y has. That is, the pieces possessed by Y are a subset of the pieces possessed by X. For a peer X to be able to send data to a peer Y, Y must be interested in X, and Y must be unchoked by X. We say that peer X *chokes* peer Y if X is unable to send data to Y. Conversely, we say that peer X *unchokes* peer Y if X can send data to Y. The choke algorithm is designed to guarantee a reasonable level of download and upload reciprocation, thus penalizing those who infrequently upload and rewarding those who frequently upload [7]. This algorithm will be described in greater detail in Section 2.3.

2.2 Rarest Piece First Algorithm

To give each peer a chance to reciprocate faster (instead of waiting to be unchoked), the first four pieces are provided to a peer at random (the *random first policy*). Towards the end of a peer's download, *end game mode* is initiated. In end game mode, all remaining pieces are requested from all peers. The communication complexity of this mode is high, but allows the last pieces to be obtained quickly. Generally, the peers run a *rarest first policy*. In rarest first, a peer always requests the piece from its peers that it needs, and is possessed by the fewest number of them. This allows rare pieces to be replicated and more readily available [7].

2.3 Choke Algorithm

Another important algorithm used by BitTorrent is the choke algorithm. This algorithm is intended to guarantee a reasonable level of upload and download reciprocation. As a consequence, *free-riders* (peers that never upload) are penalized [7].

There are three different ways the choke algorithm is initiated by a peer in the leecher state: at a regular interval (10 seconds), when a peer leaves the peer-set, and each time the interest of an unchoked peer changes. At the beginning of every third round of the algorithm, one interested and choked peer is chosen at random, called the *planned optimistic unchoked peer*. If a peer has not sent at least one block in the last 30 seconds, it is excluded from the unchoking process. The peers that become unchoked are the three fastest peers and the planned optimistic unchoked peer. If the planned optimistic unchoked peer is one of the three fastest, another is chosen at random. As a consequence, four interested peers are unchoked in each round [7].

The algorithm run by a seed is slightly different, but is initiated the same way. Peers that are unchoked and interested are ordered based on when they were last unchoked (their upload rates break the ties). All other peers are ordered only by their upload rate, and are added to the bottom of the first list. During two out of three unchoking rounds, the first three peers are kept unchoked, and a fourth interested peer is randomly unchoked. During the third round, the first four peers are kept unchoked. This algorithm is designed to frequently change a seed's active peer-set, and not favor peers with high download rates [7].

2.4 Implications for Network Model Design

As one can see, the dynamics and causal relationships among peers is extremely complex. Consequently, we are limited to the extent with which we can abstract away such interactions without incurring losses with respect to peer-protocol interactions. For example, a peer need not receive a full 256 KB piece from a single peer, nor is it guaranteed to receive blocks within a piece in-order, or pieces themselves in any particular order. Additionally, the pattern with which pieces are received impacts the "rarest piece" within

a peer-set. This rarest piece will vary among peer-sets as their view of the available pieces changes over time. This in turn impacts which pieces a peer will request, and ultimately determines the download completion time along with other network effects. This point is especially critical if we attempt to make any sort of cross-P2P model performance comparisons. Thus, it is imperative that any abstraction preserve the dynamics between peers, peer-sets, available pieces, and rarest pieces. Because of this, we are forced to model this protocol at the level of a piece. However, as we will show, this level affords a 40x event reduction over a pure packet-level model.

Because of this abstraction, our model, described in greater detail in [31], efficiently simulates the BitTorrent protocol with a memory footprint 30 to 1,000 times less than the operational BitTorrent client, using commodity hardware. The system is built on the ROSS discrete-event simulation system [32, 30], which provides a logical process world view. Here, we map the active peer and tracker objects to logical processes that exchange time-stamped event messages, which models the scheduling of real messages between peers as prescribed by the BitTorrent protocol. Our model is capable of scaling to 100's of thousands of peers. Model performance results and analysis for large torrents ranging from 128 to 128,000 clients, with files up to 4,092 pieces have been conducted, but details on those results have been reported elsewhere, and are beyond the scope of this paper.

2.5 Related Work

2.5.1 Internet Mapping Projects

Caida [5] used *skitter* data that they collected over time to construct a connectivity model between registered Autonomous Systems (AS) on the Internet. This model captures the connectivity between groups of networks, however, leaves internal network structure unknown. This model is not suitable for our simulations because realistic hop counts cannot be determined (AS can be disconnected or even span the country in 1 hop). Further, we have no data regarding the location of nodes or their corresponding bandwidths.

Lumeta [9] also created an Internet map using trace data. The map is very large-scale, and does give a notion of location. However, probes were initiated from a single source, thus the map is very tree-like, and hop counts cannot be accurately inferred. Rocketfuel [21] is an Internet mapping tool that allows for direct measurements of router-level ISP topologies. The number of required traces is significantly reduced by exploiting BGP routing tables, using properties of IP routing to eliminate redundant measurements, alias resolution, and using DNS to divide maps into POPs and backbone. Using 300 sources and 800 sinks, Rocketfuel creates extremely detailed maps of specific ISPs [22]. We use a similar technique to map parts of the backbone and POPs, however, we abstract out specific ISPs. This allows us to scale to larger simulations while keeping realistic ISP properties.

Mercator [24] is a similar tool that uses informed random-access hop-limited probes to explore the IP address space. Targets are informed by the results of earlier probes as well as IP address allocation policies. Mercator is deployable anywhere because it makes no assumptions about the availability of external information to direct the probes. It uses alias resolution and a technique called source-routing to direct probes in non-radial directions from the source in order to discover cross-links that would not have otherwise been found. In our model, we use carefully chosen addresses and ranges to probe in order to guarantee the coverage of certain key geographic regions.

[23] describes a model of the U.S. Internet backbone constructed using merged data sets from the existing Internet mapping efforts Rocketfuel and Mercator, and identifies areas where the research community lacks data, such as link bandwidth and link delay data.

2.5.2 Abstractions

Presented in [29] are fluid models used to study the scalability, performance, and efficiency of BitTorrent-like file-sharing mechanisms. The idea is to approximate a system through theoretical analysis, rather than a detailed simulation.

In [16], *NIx-Vector* routing (short for Neighbor-Index Vector) is introduced. Typically, routing of packets on the Internet consists of a series of independent routing decisions made at each router along the path between any source and destination. Hence, when many packets are sent between the same pair of nodes, the same decisions are made repeatedly and independently, without knowledge of any previous decisions. A NIx-Vector is a compact representation of a routing path that is small enough to be included in a packet header. Once this vector exists, routing decisions can be made at each router in constant time, without requiring caching or state saving. This technique can significantly reduce the burden on routers.

Staged Simulation [19] is a technique for improving the runtime performance and scale of discrete-event simulators. It works by restructuring discrete-event simulators to operate in stages that pre-compute, cache, and reuse partial results to drastically reduce the amount of redundant computation within a simulation. Like all abstraction techniques, there are advantages and trade-offs. Experiments show that this technique can improve the execution time of the NS2 simulator considerably.

One of the first flow-based network models was reported in [1]. Here, a two order of magnitude speedup is achieved over a pure packet-level model by coarsening the representation of the traffic from a packet-basis to a "cluster" of closely spaced packets called a *train*. Narses [25] and GPS [26] are other flow-based network simulators that approximate the low-level details such as the physical, link, network, and transport layers. A similar framework is presented in [28]. Our simulator is also flow-based performing at the piece-level without neglecting low-level details, allowing us to analyze application layer behavior as well as the effects on the underlying network.

Most recently, [13] reports a new method for periodically computing traffic at a time scale larger than that typically used for detailed packet simulations. This is especially useful for large-scale simulations where the execution cost is

exceedingly expensive. Results suggest huge speedups are possible when comparing background flows to those simulated in pure packet simulators. In addition, comparing the foreground interactions verifies the accuracy of the technique.

[18] discusses a novel approach for scalable and efficient network simulation, which partitions the network into domains and the simulation time into intervals. Each domain is simulated concurrently and independent of the others, using only local information of the interval. At the end of each interval, simulation data is exchanged between domains. When the exchanged information converges to a value within a prescribed precision, all simulators progress to the next time interval. This approach results in speedups due to the parallelization with infrequent synchronization.

Common to all of these approaches is the trade-off of accuracy for a decrease in computational complexity. In many cases, that trade-off must be made in order to make the model computationally tractable. Our plight is no different here. Large-scale P2P protocol sessions exist for many hours to days. To capture the larger-scale session dynamics within a tractable computational budget on common hardware is not possible at the packet-level. What makes our approach different are the constraints P2P protocols and BitTorrent in particular place on our network abstraction coupled with the in-home broadband usage model.

3 Internet Connectivity Model

The Internet connectivity model defines all the nodes and links present in the simulated network. As the Internet is constantly changing, a true-to-life connectivity graph of the Internet does not exist. Our model features two key components: the Internet backbone, and the neighborhood-level networks of lower-tiered ISPs. The Internet backbone contains many of the key links that glue the Internet together. The backbone is very non-uniform, and has evolved slowly over time. The neighborhood-level networks on the other hand, are very uniform, and have evolved based on the current Internet connection technology trends (i.e. cable or DSL). In particular, these two device technologies have different performance characteristics that need to be considered when distributing large video content to in-home audiences via the Internet.

In order to preserve realism and accuracy in our simulations, the model must capture many properties of the Internet, especially those in the "last mile" where most of the delay and congestion for in-home broadband networks is likely to occur. Additionally, our model must allow for a configurable number of nodes. Thus, we have developed a hybrid abstraction connectivity model to do just that.

3.1 Backbone

The importance of the Internet backbone is obvious, and because of its non-uniformity, cannot be generated easily. Because of this, our model uses a subset of the actual backbone. These nodes and connections were obtained by performing thousands of traces from 15 *sources* to 99 *sinks* all over the U.S. (see Figure 1). We reached 3,331 distinct nodes, and covered 6,239 edges. The maximum experienced degree (number of links connecting a single node) was 36, and the average degree was 3.746. When data is sent across the backbone in the simulation, we can use typical delays based on the path length to estimate its total backbone delay. Figure 2 shows the lengths of distinct shortest paths in the simulated backbone. Within the modeled backbone, low-tiered ISPs were located in many of the designated market areas defined by Nielsen Media Research [14]. These markets are driven by the Nielsen Rating System, which is used to determine viewing rates of cable and broadcast television shows by location. We use the Nielsen market data to provide a distribution of potential home viewers of content received over the Internet. This aspect is discussed in the sections below. By design, these nodes border the backbone, and can therefore be used to expand to the particular ISP's neighborhood-level networks.

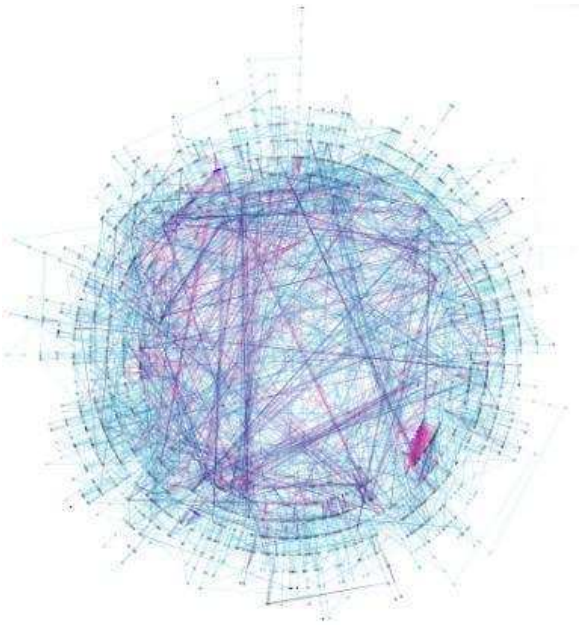

Figure 1. This figure is the connectivity graph of the backbone of the connectivity model. The nodes represent sources, sinks, intermediate backbone routers, and identified low-tiered ISP routers. The edges represent links between respective nodes.

3.2 Neighborhood-Level

Having up-to-date trace results for all ISPs would allow for maximum realism in our simulations. However, this would require constant data gathering, and the memory required to store such data (typically an adjacency matrix or adjacency list) can be on the order of gigabytes for large simulations like the ones we study. For example, a 100,000

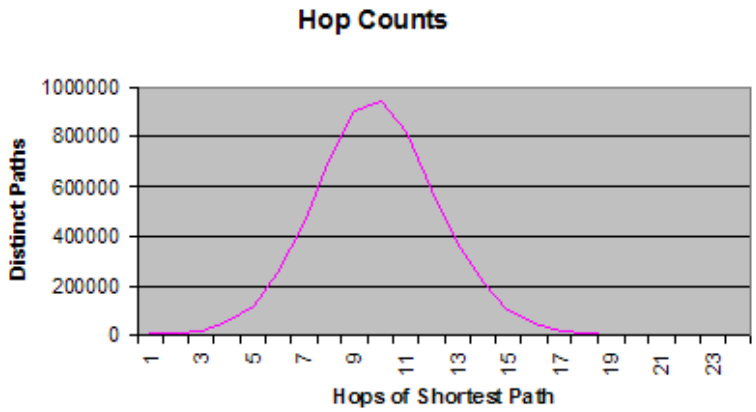

Figure 2. This figure shows the distribution of shortest path lengths for distinct paths in the backbone of the connectivity model. This curve is typical of the Internet, demonstrating that we have preserved the required path properties.

by 100,000 matrix with 32-bit entries would require approximately 37 GB of memory (see Table 1 for memory comparisons). Luckily, network design theory implies, and traces confirm, that neighborhood-level networks have similar structures regardless of the particular ISP (in particular we looked at cable and DSL ISPs). Because of this, specific ISPs have been abstracted out of the model, and we can dynamically generate these types of networks in a realistic manner. In this case, the speedup, the reduction of required system resources, and the elimination of the need to maintain an up-to-date connectivity model is worth the slight degradation of system realism.

Figure 3 shows the connectivity graph resulting from one set of traces to a popular cable ISP. From the figure, we can see how the routers are interconnecting at the different network levels, and also the *fan-outs* at the network's edge connecting to home computers/networks. This figure includes a total of 21,146 nodes resulting from responses from 21,037 homes and 109 intermediate routers. From this set of traces, the average fan-out size is approximately 540 nodes.

Our market/neighborhood-level model allows us to take advantage of symmetries that exist at the consumer broadband level of the Internet. This allows us to route without using any adjacency-storing data structures. For example, all peers in the same neighborhood have common routers (usually a few hops away) used to route within the neighborhood. Similarly, peers within the same market area have common routers used to route between neighborhoods and the ISP's backbone. Thus, an individual peer's adjacencies are unimportant. Whether a message is being sent to the same neighborhood, a different neighborhood within the same market, or to a completely different market, hops along the paths to common routers can be accounted for, and the message can be forwarded to the appropriate peer or backbone router. Although asymptotically the same, this technique provides a space and computational complexity improvement over the popular adjacency list data structure, while providing the same routes.

4 Population Model

As previously mentioned, the backbone portion of the connectivity model has identified ISPs by location. Using our current population statistics of the given designated market areas [14], we can generate realistic neighborhood-level networks. For example, if a city has 1-Million cable Internet subscribers, it is unrealistic to generate 5-Million such nodes within the neighborhood-level networks of that city. In terms of abstraction, the population data for specific cities allows us to take into consideration time zones and the targeting of certain populations. Since we are mostly concerned with media distribution, which may include streaming, this level of fidelity is required to give realistic simulation results.

5 Delay Model

Our Internet connectivity model (discussed in Section 3) provides our simulations with realistic hop counts. As previously mentioned, we are simulating Internet content distribution, thus, we must measure time in order for our experiments to be useful for analysis. Therefore, we must have appropriate delay and bandwidth models. In this section, we describe our delay model.

Research has shown that compared to the delay at the first and last links on a packet's course, the delay through the Internet's core is negligible [27]. Because of this, we can use estimates of the core delays without significantly impacting our results. Our estimates come from live measurements. Figure 4 shows the average delay experienced at each of the first 18 hops along a packet's trajectory for roughly 100,000 performed traces. The curve suggests

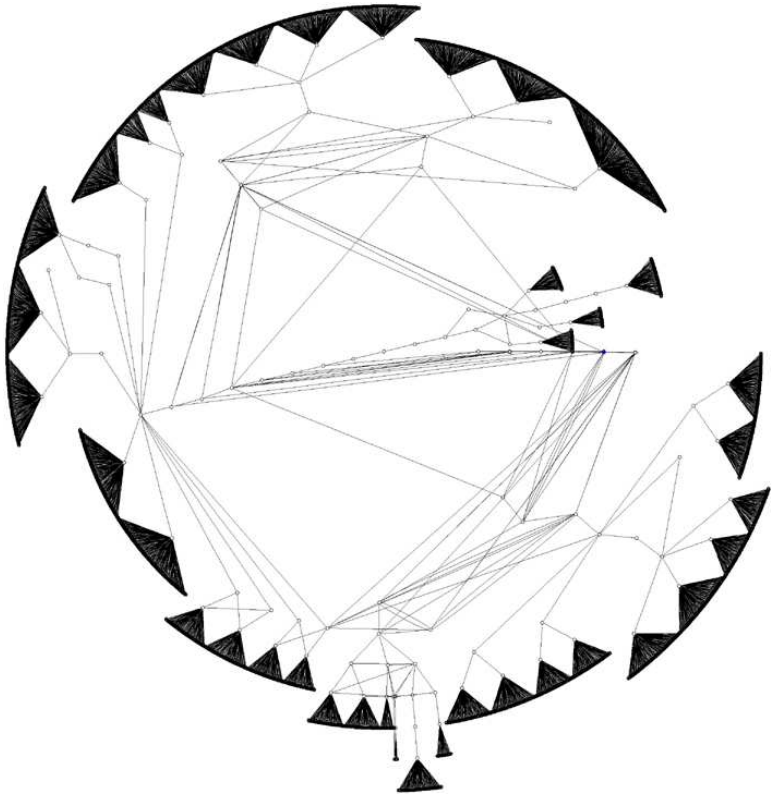

Figure 3. This figure is the connectivity graph resulting from one set of traces to a popular cable ISP.

that even though many factors, both predictable and unpredictable, contribute to delay, it generally increases at later hops (and decreases closer to the destination). Because of this, we believe that using average delays and distributions around core hops is realistic. Further, the traces were performed on the live Internet over several days. Thus, the averages inherently capture the effects of background traffic, while reducing computational costs and data-gathering needs. For the first and last links, the delays and available bandwidths are specified by the technology model.

6 Technology Model

The technology model describes what type of device a home user uses to connect to the Internet. Research has shown that long delays exists at the first and last links along a packet's path. Thus, the technology model affects the delay model. According to [20], depending on the DSL provider, service levels can range from 128 Kbps to 7 Mbps downstream from the Internet to the user while upstream service levels from the user to the Internet can range from 128 Kbps to 1 Mbps. Cable service levels can range from 400 Kbps to 10 Mbps downstream and 128 Kbps to 10 Mbps upstream. Service levels depend on service agreements offered by each cable system operator per market, and depend on whether the access is for residential or commercial use. But typically, cable (hybrid-fiber coax) has more bandwidth available than DSL.

Since we are interested in simulating cable and DSL users, we will generate the nodes in our connectivity model according to the national percentages of home users that connect using the two technologies. The delay model can therefore include the delays at the first and last links based on the device being used. These delays have been observed in our traces. Figure 5 shows the national averages of cable and DSL users from 2003 and 2006 [6].

Depending on the simulation needs, more devices can be used in the technology model, and the other topology components should be updated appropriately.

7 Bandwidth Model

The last major component of our topology model is the bandwidth model. Equation 1 [10] provides an upper bound estimate on the bandwidth for delivering a 16 KB block. In Equation 1, BW is the bandwidth; MSS is the maximum segment size (which is 1,460 bytes in default TCP, and 1,380 bytes in BitTorrent); RTT is the round trip time; and p is the probability of packet loss.

To calculate the delay of a path, we apply a truncated (values below zero are not used) normal distribution to the

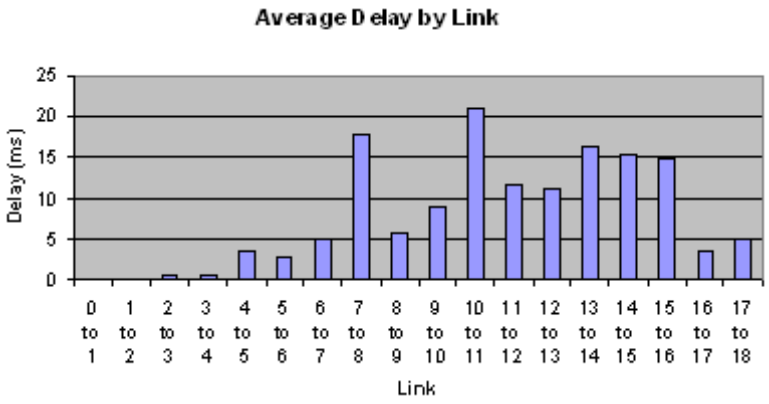

Figure 4. This figure shows the average delays experienced at each of the first 18 links along a packet's path from our traces.

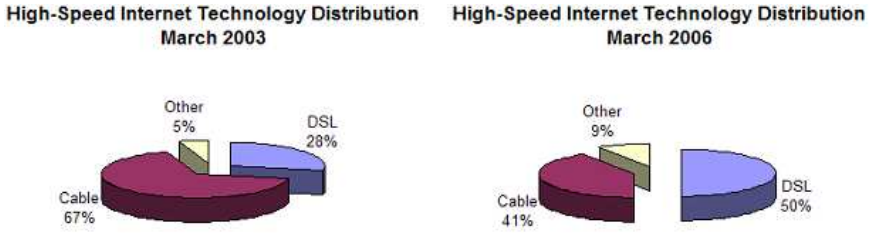

Figure 5. This figure shows the national technology distribution for home high-speed Internet connections for March of 2003 and March of 2006.

observed average delay. We use this distribution because we observed a Gaussian curve in the real trace delays similar to Figure 2. From here, RTT is set to twice the sum of the overall path delay (accounting for the path and the return path).

The probability of packet loss is set to 0.05, which is based on the loss rates of the ISPs we observed. The final bandwidth is rate-shaped based on the available bandwidth remaining along the pipe.

$$BW < \frac{MSS}{RTT}\frac{1}{\sqrt{p}} \quad (1)$$

Simulated Peers	Memory Required (MB)	Memory Required (MB) With Matrix
10,000	654	1,035
20,000	1,308	2,833
50,000	3,270	12,806
100,000	6,540	44,686

Table 1. Approximate memory required for simulation runs.

8 Results

In this section, we provide experimental results that defend our topology model design. In Table 1, we compare the amount of memory required by our connectivity model versus a model with all real nodes stored in an adjacency matrix, for several simulation runs with a varying number of nodes. The results are based on a memory footprint of 67 KB per peer (which has been achieved in [31]). It is obvious that the savings are drastic, and it is not shown, but the memory accesses may also increase the simulation time significantly for the adjacency matrix model.

In Table 2, we revisit some simulation runs published in [31]. In particular, we look at a swarm of 1,000 peers, and files with the following number of 256 KB pieces: 128, 256, 512, and 1,024. We compare the number of events processed in each simulation to a lower bound estimate of the number of events required in the equivalent packet-level simulations. The lower bound takes into consideration the actual data broken up into packets and the TCP acknowl-

edgments. TCP control messages are ignored, and BitTorrent protocol messages are not broken up (both would increase the number of events further for the packet-level simulator). For each transmission mentioned, an event is created at each hop through the network. As shown, the number of events increases by a factor of 34 to 41 for the given simulation runs. Note that this event increase significantly increases the execution time of the simulations, as the event-rate is likely to remain roughly the same, hence, our simulations experience a tremendous speedup as a result of the reduced number of events processed.

Pieces	Piece-Level Events	Packet-Level Events
128	19M	650M
256	36M	1.3B
512	66M	2.56B
1,024	122M	5B

Table 2. Number of events generated in the piece-level simulations and lower bound on the number of events generated in the packet-level simulations.

Figure 6 shows the download completion times across all peers for a modified version of the INRIA/PlanetLab testbed scenario [8]. Here, 40 peers are divided into 2 groups, fast peers and slow peers. The fast peers have a 200 KBps upload capacity while the slow peers have only a 20 KBps upload capacity. The download capacity is set in our simulation to 100 MBps. The primordial seeder has the same upload capacity as a fast peer. There are a few key differences between our scenario and the INRIA/PlanetLab scenario. First, the PlanetLab network topology is less complex than our topology. Our 40 peer scenario is distributed across a network that spans the top 31 television markets in the U.S. Next, because our model is currently only able to support cable and DSL devices, we only have two speed classes of users at this time. Lastly, because of the radically different random number generation seed-sets used across the 10 experiments, our range of different peer-sets and piece selection is much greater. However, despite these differences, we observe that our download completion times, in terms of shape, are similar to what they report – i.e., the conical S-shape. This shape has also been reported by [3] in their emulation of BitTorrent for a 700 peer scenario. This result provides confidence that both our network and BitTorrent models are behaving as expected.

9 Conclusion

We have created an Internet topology model that preserves the Internet's structure, and provides a reasonable traffic model for our simulation studies of P2P file distribution strategies. Our piece-level model can achieve the same accuracy as a packet-level model while requiring 40 times fewer events to be processed. Because of the components

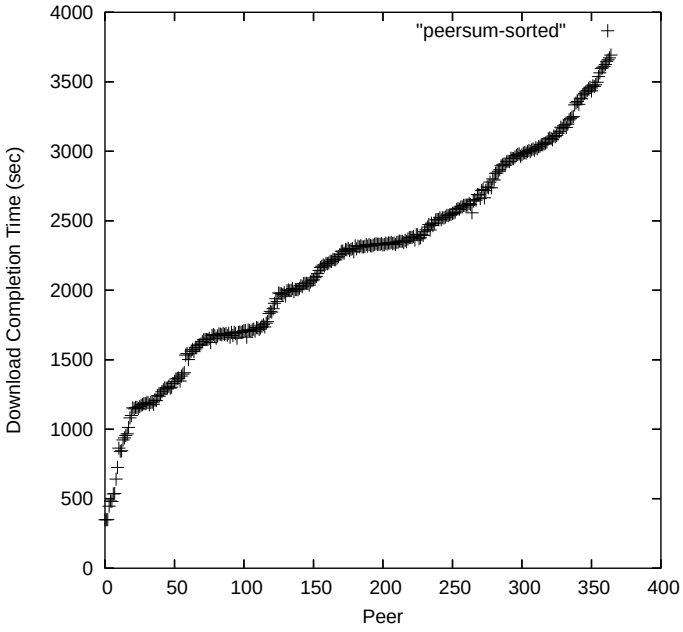

Figure 6. This figure shows the download completion times of the modified INRIA/PlanetLab scenario taken from [8]. In our case, we varied the random number seed-sets across 10 separate runs of the 40 peer, 1 seeder scenario. Thus providing us with 400 peer data points.

and levels of abstraction used, the model can easily adapt to accept changes. As the model is compact, we have been able to simulate swarms of over 200,000 peers. This paper describes how an acceptable model can be created that does not waste precious time and resources by including details that are not required for analysis of the P2P protocol understudy.

References

[1] Jong. S. Ahn and Peter B. Danzig. Packet Network Simulation: Speedup and Accuracy Versus Timing Granularity. *ACM Transactions on Network (TON)*, Volume 4, Number 5, October, 1996.

[2] Ashwin R. Bharambe, Cormac Herley, and Venkata N. Padmanabhan. "Analyzing and improving bittorrent performance". Technical report, MSR-TR-2005-03, Microsoft Research, February 2005.

[3] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates and A. Zhang. "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection", In *Proceedings of the 2006 International Conference on Distributed Computing Systems*, July 2006, Spain.

[4] BitTorrent – Home Page. http://www.bittorrent.org.

[5] CAIDA – Home Page. http://www.caida.org.

[6] John B. Horrigan. "Home broadband adoption 2006". Pew Internet & American Life Project, May 2006.

[7] A. Legout, G. Urvoy-Keller, and P. Michiardi. "Understanding BitTorrent: An Experimental Perspective". Technical report, INRIA, Eurecom, France, November 2005.

[8] A. Legout, N. Liogkas, E. Kohler, L. Zhang Cluster and Sharing Incentive in BitTorrent Systems Technical report, INRIA, #inria-00112066, version 1 Eurecom, France, November, 21, 2006.

[9] Lumeta – Research Mapping Home Page. http://www.lumeta.com/research/mapping.asp.

[10] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), 1997.

[11] Network Simulator (NS) – Home Page. http://www.isi.edu/nsnam/ns/ns.html.

[12] D. Nicol. Tradeoffs between model abstraction, execution speed, and behavioral accuracy. In *European Modeling and Simulation Symposium*, 2006.

[13] David M. Nicol and Guanhua Yan. Simulation of network traffic at coarse timescales. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 141–150, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Nielsen Media – Home Page. http://www.nielsenmedia.com/dmas.html.

[15] A. Parker. P2P in 2005 http://www.cachelogic.com/research/2005_slide01.php.

[16] G. Riley, E. Zegura, and M. Ammar. Efficient routing using nix-vectors. Technical report, GIT-CC-00-13, March 2000.

[17] Slyck – Home Page. http://slyck.com/bt.php?page=2.

[18] Boleslaw K. Szymanski and Yu Liu. Simulation of large scale networks iii: loosely-coordinated, distributed, packet-level simulation of large-scale networks. In *WSC '03: Proceedings of the 35th conference on Winter simulation*, pages 712–720. Winter Simulation Conference, 2003.

[19] K. Walsh and E. Sirer. Staged simulation: A general technique for improving simulation scale and performance. ACM TMACS, 14(2):170–195, April 2004.

[20] Time Warner Cable. "Cable Vs. DSL", http://raleigh.twcbc.com/about/cable_vs_dsl.cfm.

[21] Rocketfuel – Home Page. http://www.cs.washington.edu/research/networking/rocketfuel.

[22] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. *In Proceedings of ACM/SIGCOMM '02*, August 2002.

[23] M. Liljenstam, J. Liu, and D. Nicol. Development of an Internet Backbone Topology for Large-Scale Network Simulations. *In Proceedings of the 35th Conference on Winter Simulation: Driving innovation*, pages 694–702, New Orleans, Louisiana, December 2003.

[24] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet Map Discovery. *In Proceedings of IEEE INFOCOM*, pages 1371–1380, Tel Aviv, Israel, March 2000.

[25] Narses Network Simulator – Home Page. http://sourceforge.net/projects/narses.

[26] Weishuai Yang and Nael Abu-Ghazaleh. GPS: A General Peer-to-Peer Simulator and its Use for Modeling BitTorrent. *In Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 425–434, Washington, DC, USA, 2005.

[27] TJ Giuli and Mary Baker. Narses: A Scalable Flow-Based Network Simulator. *ArXiv Computer Science e-prints*, CS0211024, November 2002.

[28] Hannes Birck, Oliver Heckmann, Andreas Mauthe, and Ralf Steinmetz. Analysis of Overlay Networks at Message- and Packet-Level. Technical report, KOM-TR-2004-03, Darmstadt University of Technology, June 2004.

[29] Dongyu Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. *SIGCOMM Comput. Commun. Rev.*, pages 367–378, October 2004.

[30] C. Carothers, D. Bauer, and S. Pearce. ROSS: Rensselaer's Optimistic Simulation System User's Guide. Technical report, #02-12, Department of Computer Science, Rensselaer Polytechnic Institute, 2002.

[31] C. D. Carothers, R. LaFortune, W. D. Smith, and M. R. Gilder. A Case Study in Modeling Large-Scale Peer-to-Peer File-Sharing Networks using Discrete-Event Simulation. *In Proceeding of the International Mediterranean Modeling Multiconference*, pages 617–624, Barcelona, Spain, October 2006.

[32] C. Carothers, D. Bauer, and S. Pearce. ROSS: A High-Performance, Low Memory, Modular Time Warp System. *In Proceeding of the 14th Workshop on Parallel and Distributed Simulation*, pages 53–60, May 2000.