Simulation and Topology Generation for Large-Scale Distributed Systems

by

Lechang Cheng

M. A. Sc., University of British Columbia 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2009

Abstract

Simulation of P2P systems at large scale is important because some problems with the protocols or their implementations might not appear at smaller scales. In this work, a parallel message-level simulator, P2PNet, is proposed, which can simulate P2P systems with up to tens of thousands of nodes. P2PNet applies the technique of time expansion and uses real time to synchronize the processing of events among the participating processors. Simulation results show that P2PNet has small overhead compared with a single-processor event-driven simulator, a large speedup when multiple computers are used and no late events. One of the other challenges of large-scale network simulations is the lack of scalable and realistic Internet topology generators. In this work, we propose a topology generator which can generate accurate large-scale models of the Internet. We extract the AS (autonomous system) level and router level topology of the Internet with Internet measurement data. A compact routing core is built with the AS topology and router cluster topology. Each generated topology consists of the routing core and a set of end nodes connected to router clusters. The generated topology is realistic since its routing core is extracted from Internet. We also propose efficient algorithms to compute AS level path. The current routing algorithms of DHT-based P2P systems have a large end-to-end delay and inconsistent routing performance because of their random selection of identifiers (IDs). In this paper, an Internet topology based overlay construction method is proposed for tree-based DHTs. The node ID is divided into three parts and assigned according to the autonomous system (AS), IP network prefix, and IP address of the node. This

algorithm assigns the AS ID prefix based on the AS-level Internet topology. The assignment of AS ID prefixes also takes into account the node densities of ASes to alleviate the ID space load imbalance. Simulation results show that this method can reduce the routing stretch and the standard deviation of the routing stretch without introducing any single points of failure.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am extremely thankful to my supervisor Dr. Mabo Ito for giving me this opportunity to work with him. I still remember nine years ago when he accepted me as his student and helped me to step into a new area other than my background. His guidance and support has helped to learn and grow both as an engineer and a researcher. His patience and kindness gave me the maximum freedom in my research and my career. Without his support, this work would not have been possible.

I am indebted to Dr. Norm Hutchinson for his invaluable inspiration and guidance during my research. His kindness, encouraging words have helped me when I met with various obstacles and difficulties.

I also want to thank Dr. Steve Wilton, Dr. Jose Marti and Dr. Alan Wagner who kindly agreed to be member of my exam committee. Their comments are valuable to make this thesis better. I also want to thank Dr. Victor Leung and Dr. Son Vuong for guiding my in the beginning of my research and charing my qualify exam committe.

My graduate study would not have been so pleasant and memorable without the following friends, Jack Chen, Dennis Yim, Naizhi Li, Jian Chen and Jefferoy Tan.

I would want to thank the members of Bethel Chinese Christian MB Church. Their love, support and prayer have been kept me and my family through difficult times in my graduate study life. They are like my family while we were in Vancouver. I also want to thank brothers and sisters in Evangelical Chinese Church, East King County Seattle, especially our cell group. Their uncessing support, prayer and encouragment helped me make through difficult times when I have to work full time

and do research as the same time in the past two years.

I would like to give my earnest appreciation to my parents for raising me and my sisters and my brother for loving me and supporting me. The same appreciation goes to my parents-in-law my and sister and brother in-law for their love and support. Words can not describe my deepest appreciation to my wife Joy. Her unconditional love and support has been the light and joy in my long journey of graduate study and my life forward. She has been giving me support and encouragement wheneven I need them and encouraging me to finish my study even she has to sacrifice the time we spent together and endure various pressure. I would like to thank my son Elijah and daugther Evangeline. They are the joy in my life.

Last but the most, I want to thank God and my Lord Jesus Christ for his salvation, love, guidance, provision. I want to give all the glory to Him.

To my parents and my wife Joy.

Chapter 1

Introduction

1.1 Motivation

In recent years, there has been considerable research effort in the area of peer-to-peer (P2P) systems and applications since the appearance of the music sharing application Napster. A P2P system is defined as a class of systems and applications that employ distributed resources in a decentralized manner to perform some functionality such as file sharing, collaboration, communication, distributed computing, etc. Conceptually, the P2P model could be regarded as an alternative to the client-server model. It differs from the client-server model in that every node in the system is both a client and a server and there is no centralized control. The development of P2P systems is largely motivated by the trend of decentralization in building large-scale Internet applications and sharing of distributed resources, such as storage, bandwidth and computation power of end users. The advent of the Internet, widely available wireless connectivity and ubiquitous computing provides great opportunities for such large-scale Internet applications.

This thesis is aimed to address some of the issues usually encountered in designing and evaluating P2P systems. Firstly, the simulation is important to any network research, especially for P2P network research. As P2P systems usually involves tens

of thousands, even millions of nodes, mathematical analysis of a system of such scale is impractical. Experiments of the designed system are also impossible before the actual deployment. Therefore, simulation plays a critical role in the evaluation of P2P systems in large scale. Currently, there are many general purpose discrete event simulators. Sequential simulators, such as NS-2 [2], OPNET [4], etc., have the limitation of performance. A lot of parallel simulators have been proposed. However, they are usually packet level simulators and not suitable for message level simulation in large scale. In this work, we try to design a parallel overlay simulator which can simulate P2P systems with hundreds of thousands of nodes.

Another challenge of simulation of P2P systems is the Internet topology. In evaluation of network research, researchers are commonly forced to question the representativeness of their topology used in the simulation. Without a representative topology which can capture the main characteristics of the Internet, the simulation results are always less convincing. Currently, there have been a handful of topology generators. Random generators failed to capture any characteristics of the Internet. Recently, there have been many power law topology generators. However, it is not widely agreed that if power laws truly exist in the Internet topology. Hierarchical topology generators such as GT-ITM [25] and Tier [46] capture the hierarchical structure of the Internet. However, the domain level topology and router level topologies inside each domain are randomly generated. By far, all the topology generators are focused on the characteristics of the topologies generators and none of them consider the routing path computation. When the number of nodes reaches a million, the computation must be efficient in terms of both message and CPU usage. It is equally important to design a topology generator which can be efficient in computating routing paths. This work is also aimed to address this challenge by proposing a topology generator which can generate topologies that are both representative and efficient in computing routing path.

Structured P2P systems based on DHTs (Tapestry [150], Pastry [125], Chord

[132], CAN [119], etc.) have attracted a lot of research and many applications have been developed based on DHT. However, there are a lot of open issues with DHT based system. Routing issues are one of them. The current routing algorithms of DHT-based P2P systems have a large end-to-end delay and inconsistent routing performance because of the randomness in generating node identifiers (IDs). A structured P2P system relies on node ID to determine routing paths. However, the node identifiers are not related to the position of the node in the Internet, which can cause high end-to-end latency. Improve the routing efficiency is important to the adaption of DHT based P2P systems.

1.2 Thesis Overview and Contributions

The work and contribution are summarized in the following.

1.2.1 Parallel Discrete Event Simulator for P2P

In chapter 3, the design of an overlay simulator (P2PNet) which can simulate P2P systems at large scale is proposed. The main idea of P2PNet is to use real time to synchronize all the participating processors so as to reduce the synchronization overhead. P2PNet executes on a cluster of computers, each of which runs an event-driven scheduler. Each event scheduler uses real time to calculate an estimate of the global minimum simulation time in the system which is called the lower bound time-stamp (LBTS). The scheduler only processes events that have timestamps less than LBTS. LBTS has a piece-wise linear relationship with real time and the ratio between real time and the rate at which LBTS changes is dynamically adjusted so as to eliminate or minimize the occurrence of late messages while minimizing the total simulation execution time. In this part, the novel contributions are as follows:

- The design of a new mechanism for parallel distributed event-driven simulation in which real time is used to synchronize the event processing of the participating

processors.

- An algorithm to adjust the rate at which logical time advances relative to real time which is adaptive to the varying workload experienced by the processors.
- Implementation of the simulator in Java (the simulator and its source code is ready for public use).
- Evaluation of the simulator on a cluster of distributed memory parallel computers. The simulation results show that P2PNet can provide speedup close to optimum value without introducing late messages.

1.2.2 Scalable and Representative Topology Generator

In this part, a topology generator is proposed to generate accurate large-scale models of the Internet. The AS-level (autonomous system) and router-level topologies of the Internet are extracted with various data sources such as BGP routing tables and traceroute records. With the real Internet topology, we infer the AS topology and the commercial relationship among ASes. Furthermore, routers are grouped into clusters according to their position in the Internet. A compact routing core is built with the AS topology and router cluster topology. Each generated topology consists of the routing core and a set of end-hosts connected to router clusters. The generated topology is realistic since its routing core is extracted from the Internet. With the assumption of uniform routing policy within an AS, the routing path calculation of any source/destination pair consists of finding the AS path for the source/destination ASes and finding the router level path within each AS in the AS path. Since the routing path depends only on the routing core, its size is independent of the number of end-hosts in the generated topology. In this part, the novel contributions are as follows:

- The design of a topology generator which based on Internet topology.

- An improved algorithm to compute AS path given AS relationship matrix.
- An algorithm to combine data from various data source and complete AS level topology and router level topology.
- The data processing algorithm and topology computation algorithm is implemented in Java and it is available for public use.
- Evaluation of the network distance characteristics of the topology generated and comparison of them with other topologies and the results show that RealNet is more representative and is efficient in terms of routing path computation.

1.2.3 Improvement of Routing in DHT-based Systems

In this part, an Internet topology based overlay construction method is proposed for tree-based DHTs such as Tapestry and Pastry. The node ID is divided into three parts and assigned according to the autonomous system (AS), IP network prefix, and IP address of the node. This algorithm assigns the AS ID prefix based on the AS-level Internet topology. Proximity Neighbor Selection (PNS) is used with topology based ID assignment so that the overlay routing can match the underlying IP routing path. The assignment of AS ID prefixes also takes into account the node densities of ASes to alleviate the ID space load imbalance. Simulation results show that this method can reduce the routing stretch and the standard deviation of the routing stretch without introducing any single points of failure. In this part, the novel contributions are as follows:

- The design of a new ID generation algorithm for tree-based DHTs which can improve the routing efficiency and keep the load balance of nodes.
- Evaluation of the routing performance of Tapestry with the topology based identification. The simulation results show that the topology based identifier

assignment can reduce the routing stretch and the variance of routing stretch of Tapestry and Pastry.

Chapter 2

Related Work and Background Knowledge

In this chapter, previous work on Internet topology generators, network simulation, and routing schemes for Peer-to-Peer networking will be summarized and discussed.

2.1 Large-Scale Parallel Discrete Simulator

Simulations are very important to network research since conclusions cannot be drawn without careful evaluation. As P2P systems usually are intended to involve large amount of nodes, it is important that the simulations verify the correctness of the systems to that scale. Large-scale simulations are very challenging due to lack of efficient parallel overlay simulators. In this section, previous work on network simulators will be discussed.

2.1.1 Sequential Discrete Event-driven Simulation

For network research, discrete event driven simulation (DES) is the main simulation method. In DES model, the states of the system are discrete and the changes of states of the simulated system are driven by events which also happen in discrete time.

Each sequential DES simulator has an event scheduler and an event list. Each event is assigned with a time stamp and the set of events is stored in the event list based on time stamp. The event schedule is responsible for getting events from the event list and finding the event handler to handle the event. After an event is processed, the virtual time is changed to the time stamp of the next event in the event list. As the sequential DES does not have any parallelism in computation, it suffers from performance issues and cannot effectively and efficiently simulate complex systems or large scale systems. Currently, there are a number of general purpose event-driven simulators such as NS-2 [2], J-Sim [1], OPNET [4], and OMNeT++ [3], etc. NS-2 [2] is the most popular object-oriented discrete event network simulator. It is a packet-level simulator to simulate network protocols, such as TCP, routing and multicast, on small networks. J-Sim [1] (formerly called JavaSim) is another component-based network simulation framework which is similar to NS-2. OPNET [4] modeler is the leading commercial network simulator for studying and evaluating networks and distributed systems. OMNeT++ [3] is an open-source simulation environment that is similar to OPNET. All the above general purpose simulators can simulate Internet protocols in detail. However, it cannot scale to networks with more than a few hundreds of nodes.

2.1.2 Parallel Discrete Event-driven Simulation

Parallel DES has been proposed to solve the performance limitations of sequential DES. It has been researched for more than 30 years and a lot of papers have been published on this subject. In PDES, a simulator generally has multiple logical processes (LPs) and the logical processes handle event independently. In parallel simulation, the critical aspect is to ensure the chronological order of processing events so that the simulation results produced by parallel simulation is the same as that by a sequential simulator [143]. In PEDS systems, LPs generally use synchronization mechanisms to communicate with each other to ensure the causality constraint. Based on the synchronization algorithm, parallel EDS can be classified into two categories: con-

servative and optimistic. Conservative methods try to guarantee chronological order of processing any events and optimistic methods process events aggressively and roll back in case that any causality error occurs.

The conservative algorithm proposed by Chandy-Misra-Bryant [27][22] was the first synchronization algorithm for parallel discrete simulation. In the conservative approach, the chronological order of the processing of events is guaranteed. A LP will not process an event until it is certain that it will not receive any event with smaller timestamp in the future. Each LP sends out events through its output channel in non-decreasing order based on time stamp. The events received from the input channel are stored in a FIFO queue. As long as every input channel has at least one event, a LP will process the event with the smallest timestamp. However, if any one of the input channels is empty, the LP must wait until there are events in all channels. This algorithm can guarantee that no causality error will happen. The performance of conservative algorithm is largely controlled by the so-called look-ahead value [53] which is determined by the simulated applications.

In contrast to conservative method, Jefferson [68] proposed a synchronization algorithm called Time Warp. This is the first and the most well-known optimistic algorithm. In an optimistic approach, the LPs optimistically process events even if it is possible that events might be processed out of order. When straggler event (event that is processed out of order) is detected, the LP performs a roll-back to a saved state of previous simulated time and restarts the simulation from that time. The LPs save the states periodically for the purpose of rollback.

2.1.3 Discrete Event Simulators

As the implementation of a sequential event driven simulator is straight forward, many researcher do the simulation with their own simulator. Besides, there have been many general purpose event driven simulators.

2.1.3.1 Parallel Discrete Event Simulators

In addition, a lot of parallel discrete simulators have been developed to address the scaling problem of sequential simulators. The most popular parallel discrete simulators include PDNS [49], SSFNet [9], DaSSF [83], GloMoSim [146], and GTNetS [122].

PDNS (parallel distributed NS) [49] is an enhancement to NS-2 to utilize parallel computation on multiple machines. The main design goal of PDNS is to reduce the memory requirement by distributing the simulation load, and thus improve the overall simulation speed. It also provides compatibility in that existing NS code can run on PDNS with minimum modification. PDNS uses a conservative method for synchronization. ***GloMoSim***/Parsec [146][135] is a scalable simulation framework to simulate wireless network systems. It applies the parallel simulation functionality provided by Parsec. GloMoSim uses a layered model similar to the OSI layers. It can simulate a network up to thousands of nodes. The Georgia Tech Network Simulator (***GTNetS***) is a component-based network simulator aimed to provide a structured simulation environment that is like the actual networks. GTNets is completely written in C++ and network elements are modeled as C++ objects. GTNetS uses many techniques used in PDNS to enable parallel computation. ***SSFNet*** is a scalable high performance simulation platform. It provides a unified interface for discrete-event simulation (the SSF API). SSF also provides a high-level modeling language to describe modeling environments. SSFNet is developed in Java and the C++ variant ***DaSSF*** is also available.

2.1.3.2 Peer-to-Peer Simulators

There have been many discrete event simulators designed specifically for P2P systems [105], such as P2PSim [5], PeerSim [69], Overlay Weaver [128], and PlanetSim [59].

P2PSim is a open-source discrete event simulator developed in C++ for P2P system simulations. It supports several existing P2P protocols: Chord [132], Accor-

dion [79], Tapestry [150] etc. But the implementations of these P2P protocols are simplified compared with the real implementations. P2PSim provides several node failure models for users to simulate node failures and network dynamics. However, it does not support distributed computation. For underlying network topologies, P2PSim supports a number of topology models including constant distance topology, DV graph, end-to-end time graph, Euclidean graph, G2 graph, GT-ITM[25], etc.

PeerSim is a Java-based P2P simulator. It can be used to simulate both structured and unstructured P2P protocols. PeerSim provides two simulation models: cycle-based model and event-driven model. PeerSim uses a simplified Internet model of the message passing and it can simulate networks with large sizes in the cycle based model (up to 1 million nodes). PeerSim does not support distributed simulation.

PlanetSim is another discrete-event overlay network simulation framework developed in Java for simulation of both overlay networks and services. PlanetSim implements two DHT protocols: Chord and Symphony. It also implements the Common API (CAPI) [39]. PlanetSim has a modular and layered design. With PlanetSim, users can not only study new overlay network designs, but also evaluate network services built on existing overlay networks.

Overlay Weaver is designed to be a simulation tool for easy development and testing of P2P protocols and services. Overlay Weaver implements common APIs such as DHT and multicast so that users can study and evaluate services built on top of these APIs. It implements Chord [132], Pastry [125], Tapestry [150], etc. Overlay Weaver has a modular architecture. The routing layer is composed of several components such as routing algorithm, message service, etc.

2.1.4 Discussion

As discussed above, the current general purpose simulators (both sequential and parallel) are focused on providing a simulation framework to study the network protocols of the Internet in great detail. They are not suitable for simulations of overlay net-

Simulator	Supported P2P protocol	Scalability	Support for parallel execution
P2PSim	Chord, Accordion, Tapestry	No	No
PeerSim	Structed and Unstructured P2P	No	No
PlanetSim	Chord, Symphony	No	No
Overlay Weaver	Chord, Pastry, Tapestry	Yes	No
P2PNet [33]	None	Yes	Yes

Table 2.1: Comparison of P2P System Simulators

works in large scale. The current overlay simulators are not scalable in simulating large overlay networks since they do not support distributed simulation.

Lin. et. al. [81] proposed a simulation architecture for P2P systems which was the first to take advantage of the features of P2P systems. It is based on the conservative method and introduces a slow-message relaxation optimization which trades simulation accuracy for speed. However, the simulation usually has a large number of late messages and it is not certain whether this will affect the correctness of the simulation results. It is our object to provide a simulation framework for P2P systems which is both scalable, efficient and has rare late messages so as to guarantee the correctness of simulation results.

2.2 Representative Topology Generators

Internet topology generators are very important to Internet application simulations and the lack of representative Internet topologies is one of the key challenges. Simulations can present different results when different type of topologies are used [52]. Therefore, representative Internet topologies that can capture the key characteristics of the Internet are essential to extract valuable insights from simulation results. This work of designing a representative Internet topology benefits from results from three research areas on Internet struture: 1) Internet topology modeling; 2) Internet

topology inference and analysis; and 3) Internet topology geneation.

2.2.1 Internet Topology Inference

The Internet consists of millions of hosts such as computers or routers, connected through various types of networks. Hosts in the Internet are grouped into Autonomous Systems (AS), or routing domains. The internal structure of an ISP and the commercial relationships among ASes are usually considered as financial secrets and operators of ISPs are not willing to reveal them. Furthermore, the Internet is evolving constantly and rapidly in terms of structure and participants. Therefore, it is impossible to obtain a complete and up-to-date map of the Internet. As the Internet structure is important to the understanding of the behavior of the Internet and its applications, a lot of research has been done on indirectly inferring the structure of the Internet from various data sources. Researchers have developed algorithms to infer both the AS level and router level structure of the Internet from data sources. In this section, we will discuss some of the previous research results on inference of the Internet structure.

2.2.1.1 AS Topology Inference

The AS topology of the Internet includes ASes and edges between ASes. In the Internet AS topology graph, an edge between two ASes (nodes) represents a commercial relationship. The main data sources for AS topology inference is the BGP routing table dump and Internet routing registry. Currently, there are two research projects which use BGP trace collectors to collect BGP routing tables: the Routeview [6] project of University of Oregon and the RIPE RIS [7] project. The data is open to public for resaerch purpose. The IRR (Internet Routing Registry) is a distributed database to store routing information. With BGP routing table dumps and IRR combined together, we can find most of the links of the AS topology. However, there are still many missing links due to the limited number of views of the BGP dump.

Most of those missing links are Peer-to-Peer link since Peer-to-Peer links can only be discovered by the routing table of one of the peering ASes. IXPs (Internet Exchange Points) play a major role in providing Peer-to-Peer links for ASes. Some research has been done to infer Peer-to-Peer links by studying the participants and properties of IXPs [65].

2.2.1.2 AS Relationship Inference

The commercial agreements between ASes also affect the Internet routing and traffic in the AS level. A pair of ASes can have one of the three main relationships: customer-to-provider (C2P), provider-to-customer (P2C), sibling-to-sibling (S2S) and peer-to-peer (P2P) relationships. In [56][134][42][43], researchers have proposed algorithms to infer AS relationship based on BGP routing tables dump data.

The first work of inferring AS relationship from a BGP dump is done by Gao [56]. The proposed algorithm is based on the so-called valley-free property of AS paths in BGP routing tables. It also made the assumptions that a provider is usually larger than its customers and two peers are usually of the same size. The basic algorithm iterates through all AS paths. For each AS path, it finds the AS with the largest degree and sets it as the top provider. Each uphill link pointing to the top provider is assigned a transit number. Then each AS link is roughly classified as either a C2P link or a S2S link based on the transit number. A refined algorithm was also proposed to improve the accuracy of relationship inference by taking router mis-configuration into account. The final step of the algorithm is to go through all the AS paths and decide if a link is possibly a P2P link. A link is classified as a P2P link if it is possibly a P2P link and the degree ratio of the two ASes of the link is within a threshold.

Subramanian et. al. [134] proposed an algorithm for inferring peering relationships from BGP routing dumps from multiple vantage points. The proposed algorithm generates a directed AS-level graph from each vantage point. Based on the location of an AS in the graph, a rank is assigned to it. Then each AS is represented

by the rank vector from multiple vantage points. The relationship of the AS is inferred by comparing the vectors. It also proposed a mechanism to divide the Internet hierarchy into levels based on AS relationships and node connectivity.

Battista et. al. [42] proposed a method to compute the AS relationships given a set of AS paths based on the valley-free property. An AS path is valid if it is valley-free. The algorithm models the problem of inferring relationship as a Type-Of-Relationship (ToR) which determines the relationship of AS links to minimize the total number of invalid paths. As the problem is NP-hard, it also provides a heuristics to solve the ToR problem with small number of invalid paths.

Dimitropoulos et al. [43] proposed a new algorithm to improve the accuracy of inferring AS relationships. This algorithm uses IRR to infer S2S links. Besides, a new algorithm was proposed to improve C2P relationship inference. In this algorithm, a weight is computed for each AS edge. The weight is the function of the degrees of the two nodes of the edge. The C2P inference problem is transformed to maximizing the number of valley free paths and the sum of weights of all edges.

2.2.1.3 Router Level Topology Inference

In this section, we discuss some of the work on inferring the router-level structure of the Internet. Traceroute is the main tool to collect router level data of the Internet. It sends multiple Internet Control Message Protocol (ICMP) packets with increasing TTL in the IP header. With the addresses of the returned ICMP packets, it will record the list of hosts the packets have reached on their paths to the destination. There are many projects which utilize traceroute to map router-level internet topology [47].

Mercator [60] is the first research to use the traceroute tool to infer a router-level map of the Internet from a single, arbitrary location without any external information. It uses a technique called informed random address probing to improve probing efficiency. Mercator also uses source-routing to guide the probes to discover

cross links. As one router can have multiple interfaces, and thus multiple IP addresses, Mercator also provides a heuristic to resolve aliases.

Rocketfuel [131] uses traceroute data from multiple sources to find the router-level map of ISPs. It provides a list of heuristic techniques to improve measurement efficiency such as directed probing and path reduction. The author claims that these two techniques reduce the number of measurement by three orders of magnitude without sacrificing the measurement accuracy. **Skitter** (and its successor **Ark**) [8] has the most widely used traceroute dataset. Skitter also tried to map IP addresses into ASes and discover the AS topology. After 10 years of operation, Skitter is replaced by its successor Ark. The focus of Ark is to coordinate large-scale traceroute-based topology measurements using a process called team probing. With team probing, Ark is able to obtain a complete measurement to all routed /24 network prefixes in a short period of time.

DIMES [127] is the first distributed scientific research project to rely on the contribution from volunteers to study the structure and topology of the Internet. In this project, participant users install the DIMES agent in their computers. The DIMES agent will recurrently perform measurements on the Internet at a low rate and sends the results back to a DIMES server. Currently, there are close to 18900 DIMES agents spread in 113 countries and there have been more than 5 billion measurements. With the help from users spread in the Internet, DIMES is able to obtain a more detailed map of the Internet router level topology.

The closest work to our research is iPlane [86]. iPlane provides a service to predict Internet path performance (latency, bandwidth, loss rate, etc.). It uses distributed sites in the Internet to probe the router-level map of the Internet. It resolved the aliases and grouped the routers into PoP (point of presence). Based on available PoP level path of the Internet, it can estimate the end-to-end path performance by combining measured segments of Internet paths. Our work differs from iPlane in that it calculates Internet path performance based on inferred Internet

topology. Nevertheless, the raw traceroute data, clustering algorithm and inferred data of iPlane can be used in our work.

2.2.2 Internet Modeling

Mathematical modeling of the Internet is critical to successful generation of realistic topologies. However, the Internet is a very complex network. It has evolved into a large-scale system with billions of participating entities and complex structure from a small research network. The heterogeneity of its participants, its complex structure and dynamic nature make it extremely difficult to accurately model the Internet. Researchers have proposed a lot of mathematic models of the Internet.

Random Model: Traditionally, the Internet is described as a random graph. Given the average degree of a topology, edges are created randomly between nodes to guarantee the average degree.

Power Law: In [48], Faloutsos et al. reported the Internet is scale-free and there were three power laws in degree of the AS topology of the Internet. The power laws are:

> 1) Rank exponent: Out-degree of a node is proportional to its rank to the power of a constant; 2) Out-degree exponent: The frequency of an out-degree is proportional to the out-degree to a constant power. 3) Eigen-exponent: The eigenvalues of the adjacency graph are proportional to the order i to a constant power [48].

Some research also tried to find the root cause of power laws in the Internet topology. Medina et. al. [98] describes four factors that lead to power laws: 1) Preference connectivity of nodes to nodes with high degree; 2) Incremental growth of the Internet; 3) Distribution of nodes in space; and 4) Locality of edge connection. After the discovery of the power law, there have been many algorithms proposed to generating topologies that follow the power law. Among them are PLOD [109], PLRG [12], BA [18], ESF [13], and GLP [23].

Recently, some researchers also question if power law truly exists or it is the result of incomplete measurement. Chen et. al. [31] claimed that due to the limitation of measurement point, BGP data only represents a incomplete view of the Internet. There are many links missing from the AS topology, mainly P2P links and the topology with the missing links will not strictly follow power law. In [91], Mahadevan et. al. found that the node degree distribution of WHOIS graph does not follow power laws due to high percentage of medium degree nodes.

2.2.3 Topology Generators

Currently, there are a good number of topology generators. They can be classified as: random topology generators, structural topology generators, power-law topology generators and sampling topology generators.

WaxMan: The earliest topology generator was the random topology generator introduced by Waxman [140]. The Waxman model randomly places the nodes in a two-dimensional Euclidean space. An edge is randomly placed between a pair of nodes based on the Euclidean distance. The probability of an edge existing between two nodes is related to the distance. the smaller the distance, the higher the probability is.

Tier [46] and **GT-ITM** [25] consider the hierarchical structure of the Internet in generating topologies. Tier divides the Internet into three layers: LAN, MAN and WAN. In Tier, there is only one WAN. Users can specify the number of LANs and MANs and the nodes within LANS and MANs to build a topology. GT-ITM uses a transit-stub model. In GT-ITM, the Internet is regarded as a set of domains. Each domain can be a transit domain or a stub domain. Nodes represent routers or switches. Transit domains have routers which connect to other transit domains and stub domains. Each stub domains has one or more gateway nodes which connect to a transit domain.

Inet [70] is a AS level topology generated developed by University of Michigan

Generator	Model	AS level	Router level	Scalable (network size)	Scalable (routing computing)
Waxman	Random	No	Yes	No	No
GT-ITM	Hierarchy	Yes	Yes	Yes	No
Tier	Hierarchy	Yes	Yes	Yes	No
Inet	Power law	Yes	No	No	No
nem	Power law, Random	Yes	Yes	No	No
BRITE	Random, power law	Yes	Yes	No	No
RealNet [34]	Inferred Internet model	Yes	Yes	Yes	Yes

Table 2.2: Comparison of Topology Generator

to generate topolgies that obey the power law. It uses the Internet BGP data to determine the parameters in the power law. Inet uses the PLGR algorithm and preferential attachment to produce the power law.

nem [88] (network simulator) is a topology generator developed by University of Louis Pasteur Strasbourg). It is a general purpose topology generator which has three basic functionalities: converting network topology files from one format to another, analyzing network topology characteristics, and generating network topologies. In generating topologies, it supports the Waxman model, the AB model, the PLOD model and map sampling. Users can use nem to measure topology properties such as distance, connectivity, etc.

BRITE [97] (Boston University Representative Internet topology generator) is a general purpose topology generator developed by Boston University. Brite can be used to create router-level topology and AS-level topology. It implements the Waxman model, BA model, and GLP model. It also supports to generate hierarchical topology from top-down or bottom-up. User can also specify bandwidth and delay for links in the topologies. Furthermore, it provides interface for a list of network simulators such as ns-2, SSF, Omnet++, and JavaSim.

2.3 Peer-to-Peer Networking

Milojicic et al. [99] define P2P systems as a class of systems and applications that employ distributed resources in a decentralized manner to perform some functionality such as file sharing, collaboration, communication, distributed computing, etc. Conceptually, the P2P model could be regarded as an alternative to the server-client model. It differs from the client-server model in that every node in the system is both a client and a server and there is no centralized control. Existing P2P systems can be classified into two categories: unstructured P2P systems and structured P2P systems.

The first generation of P2P systems such as Naspter and Gnutella [51] are unstructured systems which are built on random graphs. In unstructured P2P systems, the new nodes join the network by connecting to one of the known available nodes. Thus, the participant nodes form a dynamic, self-organizing network. Unstructured P2P systems generally use a flooding protocol in search for data. To locate a data item, a peer sends queries to its neighbors. The neighbor will forward the query to the neighbors and the query will be flooded to all neighbors within a certain radius. The flooding protocol is resilient to node failure and churn of the system. However, it is not scalable and generates many extra messages on the network.

The second generation of P2P systems are based on distributed hash tables (DHTs) and have many applications. There has been a lot of work to improve many aspects of P2P systems, such as the routing, security, robustness, and search ability. In this section, canonical P2P systems will be reviewed and ongoing research of improving P2P systems routing will be discussed.

2.3.1 Structured Peer-to-Peer Systems

In order to solve the scalability issue of the first generation P2P systems, a lot of research has been done on structured P2P systems using distributed hash tables

(DHTs). Typical structured P2P systems include Chord [132], Pastry [125], Tapestry [150], CAN [119], etc. In structured P2P systems, the overlay network topology is tightly controlled and the files are placed at precisely specified locations. Every object or node is assigned an identifier. The systems provide a mapping between the file identifier and node identifiers. Queries can be efficiently routed to the node with the desired file with the neighboring table of each node. Structured P2P systems can provide a scalable solution to distributed routing and location. Many applications have been developed using structured P2P: archival stores, file systems, web caches, application-level multicast, etc.

CAN (Content Addressable Network) [119] implements a distributed hash table using a virtual d-dimension Cartesian coordinate space on a d-torus. The coordinate space is dynamically divided into unique zones among all the nodes. Each CAN node maintains a routing table of $2 * d$ entries. To route a key to a node, Chord uses a greedy forwarding algorithm which chooses the next hop as the neighbor closest to the destination.

Chord [132] builds distribued hash table on a circular m-bit identifier space. It uses consistent hashing such as SHA-1 to generate identifiers for keys and nodes. A key k is mapped to the node with the same identifier as the key k or the closest node with identifier larger than the key k in the identifier space. Each peer maintains a routing table with up to m entries. The i^{th} entry in the routing table at node n contains the identifier and IP address of the first node that follows n with a distance of at least 2^i on the identifier space.

In Pastry [125], each node is assigned a 128-bit identifier. The node identifier and keys can also be considered a sequence of digits with base b. Each Pastry node maintains a routing table, a neighborhood set, and a leaf set. The routing table has $log_{2^b}N$ rows and each row holds b entries. The entries at row n of the routing table each holds the nodes whose identifier is the same as that of the current node's identifier in the first $n-1$ digits, but different on the n^{th} digit. The neighborhood

	Hop	End-to-end latency	Routing	Object publishing	Data locality	Path locality
Tapestry	O(logN)	Medium	Good	Good	—	Medium
Pastry	Olog(N)	Medium	Good	Good	—	Medium
Chord	O(logN)	Large	Good	Good	—	—
CAN	dN1/d	Large	Good	Good	—	—
SkipNet	O(logN)	Large	Low	Constrained	Good	Good

Table 2.3: Comparison of DHT-based systems

set contains the identifier and IP addresses of a number of peers that are closest to the local peer accoring to proximity metric. The leaf set contains a number of nodes numerically closest to current node (either larger or smaller).

Tapestry [150] is very similar to Pastry. Tapestry has the same routing table as that of Pastry except that Tapestry does not have a leaf set or a neighborhood set. It uses so called surrogate routing. When a node's routing table does not have an entry for a node that matches a key's n^{th} digit, the message is forwarded to the node in the routing table with the next higher value.

There are some other structured P2P system such as Kademlia, Viceroy, and SkipNet [64]. ***Kademlia*** [96] uses the XOR metric for measuring the distance between points in the identifier space. ***Viceroy*** [93] organizes the participant nodes into an approximate butterfly network. The main design goal is to handle the discovery and location of data and resources. The SkipNet system uses two address spaces: a lexicographic space and a numeric space to achieve efficient routing and content placement. SkipNet clusters peers and organizes data according to their lexicographic address and names. Table 2.3 compares the DHT-based systems on their routing performance. It shows that none of them has small end-to-end latency.

2.3.2 Improvement to DHT-based Peer-to-Peer Networking

Recently, there has been a lot of research on improving the routing efficiency and locality of DHTs [118][120][148][57]. Some work proposed incorporation of topology information in overlay construction to improve the routing efficiency. Ratnasamy et. al. [120] proposed three methods: Proximity Neighbor Selection (PNS), Proximity Routing Selection (PRS) and geographic layout. Tapestry and Pastry implement PNS. For nodes with the same prefix, they chose the closest one as the primary neighbor. However, even with PNS, the average routing stretch (overlay end-to-end delay divided by the underlying IP network end-to-end delay) is still high. Furthermore, as the node identifier has no relationship with the position of the node in the Internet, it is difficult to provide data locality.

Ratnasamy et. al. [118] proposed a binning method for CAN to infer network information and used it to construct a topology-aware overlay. As CAN lacks the ability to use PNS, its routing stretch remains significantly high.

Some other work introduced the hierarchical structure to the DHT-based systems and leveraged the heterogeneity of the nodes to improve routing efficiency [148][57]. In Brocade [148], a set of super-nodes are selected to form a secondary overlay on top of the existing DHT. The second-layer overlay provides a shortcut routing algorithm to quickly route to a remote network. Garces-Erice et al. [57] proposed the idea of a hierarchical DHT system. Instead of building a flat overlay, the participating nodes are divided into a set of groups and each group uses a DHT to build its own intra-group overlay network and look-up service. In each of the groups, the nodes with high bandwidth and capacity are selected as super-nodes to form a top-level overlay network. This two-tier structure can help improve the routing efficiency and locality of structured DHTs, however it also introduces single points of failure as they rely more on super-nodes to forward traffic. With the topology-based identifier assignment, our work introduces the hierarchical structure to DHTs without using super-nodes.

The closest work to ours is Zhou et. al.[156]. They also propose a hierarchical location-based identifier generation method for DHTs. This method divides IDs into pre-defined prefix and random suffix parts. The pre-defined prefix is assigned hierarchically according to the geographical location of the node in the Internet. The length of the prefix varies among different groups and a Huffman like algorithm is used to encode the prefix. The length of the prefix determines the number of keys that the group is responsible for. This method provides coarse grain identifier space load balance. Our method differs from [156] in that we try to reduce the routing stretch by matching the overlay routing path with the underlying Internet routing path as much as possible. Instead of assigning identifier prefixes with respect to geographical positions, identifier prefixes are assigned with respect to the AS-level Internet structure. Furthermore, in order to achieve a low routing stretch, proximity neighbor selection must be used in combination with topology-based identifier assignment.

Chapter 3

P2PNet: A Simulation Architecture for P2P Systems

3.1 Introduction

Currently a lot of effort is being put into developing large-scale Internet applications based on P2P technology [132][150][119][125]. As P2P systems utilize the resources of end user systems, they usually involve hundreds of thousands of nodes or even more. For systems on such a scale, it is impossible to test the designed protocols with real, large-scale implementations in the Internet. Therefore, simulation is critical to the building and understanding of these systems. Furthermore, as errors in the protocols or their implementations might only appear at large scale, simulation at large scale is important for the thorough evaluation of P2P systems before their deployment[81].

There are two challenges to simulating new Internet protocols at large scales: the size and complexity of the topology of the Internet, and the complexity of the layered Internet protocols (TCP, UDP, IP, etc.) that form the underpinning of the new application protocol. It is extremely difficult to evaluate any Internet protocol at both the full complexity of the Internet topology and the complexity of the underlying Internet protocols. Simulations are normally forced to trade off accuracy

for complexity by modeling either the Internet topology or the underlying Internet protocols in a very simple way. Packet-level simulation can evaluate the performance of the application protocol with detailed models of the underlying Internet protocols but with small scale models of the Internet's topology. Message-level simulations contain more precise and large-scale models of the Internet's topology but are able to do so only by using a very simple application model of message delivery.

Most of the current simulation tools utilize a centralized event-driven scheduler which is not scalable in simulating large scale networks. In order to facilitate simulations for large-scale networks, parallel distributed event-driven simulations (PDES) have been developed. The central problem for PDES is to synchronize the event schedulers and thus guarantee that events are processed in chronological order. Existing PDES systems can be divided into two types: conservative, where simulation time moves in lock-step on all the processors, or optimistic, where each processor advances simulation time in isolation but must retain sufficient state to roll-back should it get too far ahead of its peers. Both methods experience high overheads: the conservative method exhibits high communication costs for the synchronization of simulation time and the optimistic method has high state saving and rollback costs.

In this section we present the design of a message-level simulator which can simulate P2P systems at a scale of hundreds of thousands of nodes. We take advantage of a few characteristics of message-level simulations of P2P systems to improve the performance of the simulation. In message-level simulations, the message delay (the delay between the logical time at which the message is generated and the logical time at which it should be processed) is primarily caused by latency in the Internet, which is on the order of 10s or 100s of milliseconds. Therefore, the precision of the synchronization of logical time can be less exact.

In this work, we propose a new architecture (P2PNet) for simulating P2P applications. The main idea of P2PNet is to use real time to synchronize all the participating processors so as to reduce the synchronization overhead. P2PNet executes

on a cluster of computers, each of which runs an event-driven scheduler. Each event scheduler uses real time to calculate an estimate of the global minimum simulation time in the system which we call the lower bound time-stamp (LBTS). The scheduler only processes events that have timestamps less than LBTS. LBTS has a piece-wise linear relationship with real time and the ratio between real time and the rate at which LBTS changes is dynamically adjusted so as to eliminate or minimize the occurrence of late messages while minimizing the total simulation execution time. The novel contributions of this chapter are as follows:

- A new mechanism for parallel distributed event-driven simulation is developed which uses real time to synchronize the event processing of the participating processors.
- A new algorithm is proposed to adjust the rate at which logical time advances relative to real time which is adaptive to the varying workload experienced by the processors.
- The simulator was implemented in Java and was tested on a P2P protocol.

The structure of this chapter is as follows. Section 3.2 presents the details of P2PNet. Section 3.5 presents some simulation results. Section 3.6 concludes and discusses future work.

3.2 P2PNet Architecture

The P2PNet simulator is designed to simulate P2P protocols on network topologies with hundreds of thousands of nodes. As one single computer lacks the physical memory and processing power to simulate protocols in such large-scale networks, multiple computers must be used. It can be either a cluster of computers or a distributed memory multiprocessor connected through a high throughput network. Figure 3.1

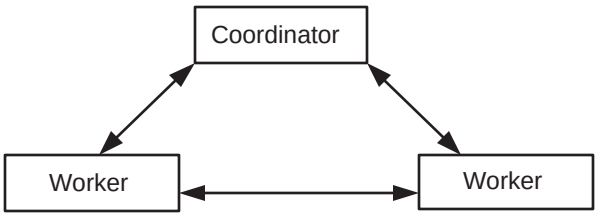

Figure 3.1: Tow-tier structure of P2PNet

illustrates the architecture of P2PNet. The simulator applies a two-tier parallel distributed event-driven simulation (PDES) architecture. One computer is the coordinator of the simulation and all the remaining computers are workers.

In P2PNet, each worker hosts a logical process (LP) and a number of simulated protocol instances. An LP consists of three components: 1) an event scheduler, 2) a simulation network component, and 3) a message delivery component. Each of the protocol instances is denoted a virtual node (VN).

The event scheduler is in control of the execution and the execution of the virtual nodes is driven by events. The event scheduler contains an event queue. Each event is assigned a time stamp which indicates the logical time at which the event should be processed. Events are stored in the event queue in increasing time stamp order. In the simulation, the event scheduler iteratively fetches events from the event queue and calls the event handler to handle each event. Event processing can trigger other events. Periodically the event scheduler will send and receive synchronization messages to the coordinator to synchronize the event processing process.

The simulation network component delivers events between virtual nodes. It abstracts the underlying Internet protocols up to the transport layer. It contains the topology of the simulated network. Based on the latency between virtual nodes in the underlying network topology, it builds a routing table. It also provides the mapping between virtual nodes and LPs. To deliver an event, the simulation network calculates the time stamp of the event and finds out the LP which hosts the destination VN. If the destination LP is the same as the current LP, it will schedule the event into the

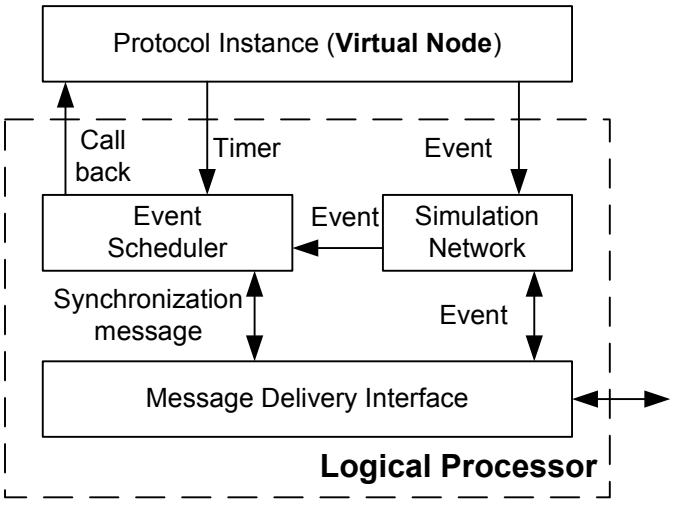

Figure 3.2: Structure of P2PNet

event scheduler. If the destination LP is in another machine, it will call the message delivery component to send the event out. When the destination LP receives the event, the simulation network component will insert it into the event queue of the event scheduler.

The message delivery component deals with message transmission between LPs. It can utilize socket, RMI, message passing interface (MPI) or any other message passing service. It delivers both the events from the simulation network component and the synchronization messages from the event scheduler. It can also provide message buffering and message retransmission should reliable message delivery be needed.

3.2.1 Time Systems

In PDES, each LP processes events and adjusts its local logical time independently. Each LP sets its local logical time to the time stamp of the current event it is processing. In order to synchronize the event processing of LPs, a global lower bound time stamp ($LBTS$) is calculated. At any time, LPs only process events that have time stamps which are less than $LBTS$. Therefore, for any LP, the local logical time

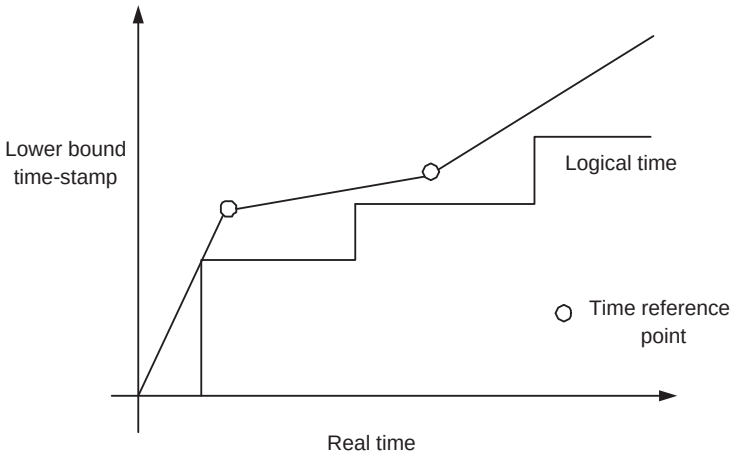

Figure 3.3: Real time, logical time and lower bound time-stamp

is always less than $LBTS$.

In this work, real time is used to synchronize the event processing of the LPs. There are three time systems in P2PNet, local logical time $\hat{t}$, real time t, and LBTS $\bar{t}$. Real time is the real execution time of the simulator since the simulation starts. LPs are assumed to have their physical clocks synchronized, and so the real time is the same for all LPs (within the precision of the clock synchronization algorithm).

In P2PNet, the LBTS has a (piece-wise) linear relationship with real time.

$$\bar{t} = (t - t_0)/K + \bar{t}_0$$

When real time advances Δt, LBTS advances $\Delta t/K$. K is called the time expansion factor (TEF). As the application behavior changes during the simulation, the TEF is adjusted dynamically. The pair $(t_0, \bar{t}_0)$ is called a time reference point; each time the TEF is adjusted, a new time reference point is established. As long as the time reference point and TEF are the same for all the LPs, the LBTS will be the same as well. Figure 3.3 illustrates the relationship between logical time, real time, and LBTS.

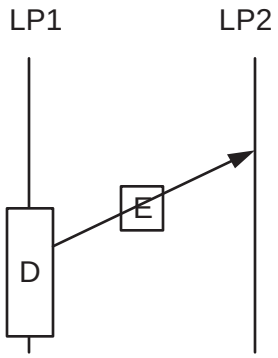

Figure 3.4: Time analysis of event processing in P2PNet

3.2.2 Time Management Algorithm

The primary task of a PDES is to guarantee the chronological order of event processing since events which happen in one LP might be processed in another LP. An event E is processed in *chronological order* if its time stamp is larger than the local logical time of the LP when it is scheduled for processing. Otherwise, it is called a late event.

Figure 3.4 shows an example of event processing in P2PNet. Supposes that LP1 starts to process event D when the real time is t_1, local logical time is $\hat{t}_1$ and the LBTS is $\bar{t}_1$. After some processing time Δt_p (in real time), it triggers event E and the time stamp of event E is $\hat{t}_1 + \Delta\hat{t}_l$ ($\Delta\hat{t}_l$ is the latency between the source and destination of event E in the simulated network topology). Suppose that event E has to be processed by LP2 and LP1 sends event E to LP2 right after it is generated. Suppose that LP2 will receive event E after a communication delay Δt_c. When LP2 receives the event, the $LBTS$ is $\bar{t}_2 = \bar{t}_1 + (\Delta t_p + \Delta t_c)/K$.

If order to guarantee that event E is processed in chronological order, the local logical time $\hat{t}_2$ of LP2 when it receives event E must be less than the time stamp of event E ($\hat{t}_2 \leq \hat{t}_1 + \Delta\hat{t}_l$). Recall that $\hat{t}_2 \leq \bar{t}_2$. Therefore, if

$$\bar{t}_2 = \bar{t}_1 + \frac{\Delta t_c + \Delta t_p}{K} \leq \hat{t}_1 + \Delta\hat{t}_l \tag{3.1}$$

we will have $\hat{t}_2 \leq \hat{t}_1 + \Delta\hat{t}_l$ and event E will be processed in chronological order.

Inequality 3.1 is equivalent to

$$\bar{t}_1 - \hat{t}_1 \leq \Delta\hat{t}_l - \frac{\Delta t_c + \Delta t_p}{K} \tag{3.2}$$

Taking the minimum of the right side of inequality 3.2 over all events generated by LP1, we obtain:

$$(\bar{t}_1 - \hat{t}_1) < \min(\Delta\hat{t}_l) - \frac{\max(\Delta t_c) + \max(\Delta t_p)}{K} \tag{3.3}$$

where $\min(\Delta\hat{t}_l)$ is the minimum network latency between any virtual node in LP1 and any virtual node not in LP1. $\max(\Delta t_c)$ is the maximum transmission delay between LP1 and other LPs. $\max(\Delta t_p)$ is the maximum event processing time in LP1. In the following, $\bar{t}_1 - \hat{t}_1$ is denoted the local logical time delay. $T_D = \min(\Delta\hat{t}_l) - \frac{\max(\Delta t_c)+\max(\Delta t_p)}{K}$ is denoted the logical time delay threshold of LP1. For LP1, if inequality 3.3 is true, then all events triggered by it will be processed in chronological order.

The local logical time delay $\bar{t}_1 - \hat{t}_1$ is the delay between the$LBTS$ and the local logical time of LP1. It becomes positive when the time expansion factor is too small and consequently the LP is unable to process all the events it should within a logical time interval. It is zero if the time expansion factor is large enough. Therefore, if the LBTS is the same for all LPs, and for each LP $(\bar{t}_1 - \hat{t}_1)$ is controlled within the threshold defined in the right side of inequality 3.3, all messages will be processed in chronological order. The time synchronization problem becomes:

- To adjust the time expansion factor K of each of the LPs so that $(\bar{t}_1 - \hat{t}_1)$ can be within the threshold defined in the right side of inequality 3.3.
- To synchronize the time expansion factor and time reference point periodically so that the $LBTS$ is the same for all LPs.

The event processing time and message transmission time are on the order of 0.1ms. With careful partitioning of the simulated network topology, $\min(\Delta\hat{t}_l)$ can be

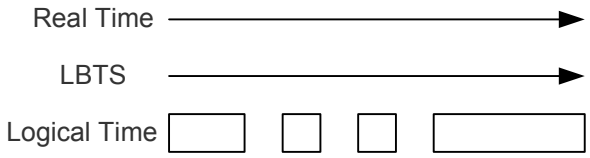

Figure 3.5: Illustration of time expansion factor

on the order of 10s or 100s of a millisecond. As the time expansion factor is always large than 1, the logical time delay threshold T_D is mainly determined by $\min(\Delta\hat{t}_l)$ and can be on the order of 10s or 100s of a millisecond. The threshold T_D provides a lot of flexibility in adjusting TEF. Insufficient TEF for a certain amount of time will not cause any late events.

3.2.3 Adaptation of TEF for a Single LP

The primary goal of synchronization is to find an appropriate TEF so that $(\bar{t}_1 - \hat{t}_1)$ will be within the threshold. In fact, the time expansion factor represents the ratio of the execution time taken by the simulation versus the simulation logical time. Figure 3.5 gives an example. Suppose that there are 4 events in the logical time interval [0, 10ms] and it takes 100ms to process them. Therefore, the minimum time expansion factor should be 10.

In the simulation, the event arrival rate and the event processing rate may be different for different LPs, and may change during the simulation. Therefore, the TEF calculation algorithm needs to be adaptive. In this section, we describe our time expansion factor adaptation algorithm. Periodically, the event scheduler measures the following three factors:

- **Logical time adjustment ratio**: This is calculated as the logical time adjustment versus the $LBTS$ adjustment in the monitoring period. If this is smaller than 1, then it means that the time expansion factor is not large enough. If this is equal to 1, this means that the time expansion factor is sufficiently large.

- **Logical time delay**: It is the difference between LBTS and logical time. This delay will be caused by an insufficient time expansion factor in this or a previous monitoring period.

- **Empty loop rate**: During the simulation, the scheduler iteratively fetches a suitable event from the event queue and processes it. There are loops in which no suitable events are found, which are called empty loops. The empty loop rate is the ratio of observed empty loops to the maximum number of empty loops a scheduler can execute in the monitoring period. This indicates the percentage of time that is wasted in a monitoring period because the scheduler found no suitable event to process. With this percentage we will be able to estimate the lowest time expansion factor that is necessary for the scheduler to process the events in the monitoring period.

The algorithm to adjust the time expansion factor is as follows. In any monitoring period, if the time adjustment ratio is less than some threshold α_l, or the logical time delay is larger than some threshold β_h, the time expansion factor will be multiplied by 2. However if the time adjust ratio is larger than α_h, and the logical time delay is less than threshold value β_l, we calculate the new TEF as follows:

TEFnew=TEF*(empty loop rate) * (1+θ)

We use a TEF that is (1+θ) times of the minimum TEF to accommodate the fluctuation of the simulation conditions. In other cases, the TEF remains unchanged. The thresholds α_l, α_h, β_l, β_h are used to accommodate the fluctuation and calculation errors of the logical time adjustment ratio and logical time delay. θ is used to force the setting of the TEF to be slightly conservative.

3.2.4 Synchronization of TEF of LPs

The algorithm for synchronizing TEF is performed periodically. At the end of each monitoring period, the coordinator sends a TEF inquiry message to all the workers.

Upon receiving the inquiry message, each LP will calculate its new minimum TEF with the algorithm described in Section 3.2.3. The workers will send an inquiry reply message to the coordinator. After collecting all the reply messages from the workers, the coordinator LP will find the maximum of TEF for all the LPs and broadcast a factor adjustment message which contains the global minimum TEF. The workers will adjust the TEF according to the factor adjustment message.

In some cases, there is long delay between sending the inquiry message and receiving the reply. In order to process reply messages in a timely manner, we set the maximum allowed delay. If a message is received after the delay, it is discarded and the coordinator will calculate the global time expansion factor based on the response messages it has received.

3.3 Discussion

In this part, we will discuss some issues related to the applicability of P2PNet, namely: adaptability, performance and the scalability.

3.3.1 Adaptability

P2PNet is designed to be a general purpose message-level simulator for P2P systems. In real situations, different P2P applications have different event arrival rates and event processing rates. The event arrival and processing rates may also change during the simulation. We argue that with the TEF adjustment algorithm, P2PNet is able to adapt to the various applications with the different event arrival rates and the event processing rates.

The simulation of P2PNet can be divided into two states: start up state and adaptive state. In the start up state, P2PNet tries to find a TEF that is larger than necessary so as to reduce late events. Currently, P2PNet starts with a default TEF. The TEF can be set as a very large number (such as 1000 or even 10000) so that it

will be large enough for most simulation situations. Furthermore, the algorithm can exponentially increase TEF if the logical time falls behind the LBTS (increase by 2 for every monitoring period). Therefore, P2PNet can adapt to the applications with a broad range of simulation workloads. As the TEF will be adjusted down when a larger than enough TEF is found, this large initial value of TEF only have a small effect on the overall simulation execution time if the simulation duration (logical time) is large enough. Further work might be to investigate adjustment algorithms which can adjust TEF more rapidly. We can either choose a higher increasing factor (large than 2) or adjust the TEF according to the logical time adjustment ratio.

In the adaptive state, the adjustment algorithm tries to adjust the TEF according to simulation situations. P2PNet can exponentially increase the TEF when the logical time falls behind LBTS. By default, P2PNet can change the TEF by 2 for every monitoring period (20ms). As discussed in Section 3.2, the logical time delay threshold T_D can be on the order of 100s of millisecond. The P2PNet can run with insufficient TEF for 5 monitoring periods without causing any late event. Therefore, the adjustment algorithm can gradually increase TET by 32 times in 100ms without causing any late events. It is unusual that the average event arrival rate and event processing rate change of large-scale simulations to change rapidly in the time scale of tens of milliseconds. Therefore, the adjustment algorithm can work for reasonable simulation workloads. In the future work, we will also investigate proactive adjustment algorithm which predict the future event arrival rate to deal with abrupt changes in the event arrival rate.

3.3.2 Performance Analysis

For parallel distributed event simulations, the communication cost involved in exchanging events among LPs is the main overhead compared with sequential event-driven simulation. In a sequential event–driven simulation, one can send an event from one virtual node to another by inserting it back into the event queue. In PDES,

if the destination virtual node of an event is in a different LP from the source virtual node, the event has to be physically sent from one computer to another. Suppose there are N total virtual nodes in the simulation and each virtual node generates R events every millisecond (in logical time). Let a be the average processing time for an event and b be the communication cost of an event (including the cost of sending and receiving an event), the simulation duration be P (in logical time). The total execution time with sequential event driven simulation is $NRPa$. With W workers, the total simulation execution time is

$$NRP(a + \frac{W-1}{W}b)/W \approx NRP(a+b)/W$$

The minimum time expansion factor is

$$NR(a + \frac{W-1}{W}b)/W \approx NR(a+b)/W$$

The speed up is

$$Speedup = \frac{W}{1 + (W-1)/W * (b/a)} \tag{3.4}$$

Where W is large, the speedup rate is

$$Speedup \approx \frac{W}{1 + (b/a)}$$

The overhead rate is about b/a. One can see from equation 3.4 that the number speedup is linearly related to the number of workers, restrained by the relative communication cost. Given the number of workers and the relative communication cost b/a, equation 3.4 gives the maximum speed up of any parallel simulator.

3.3.3 Scalability

P2PNet is designed to be scalable to simulate P2P applications on network topologies with hundreds of thousands of nodes. A regular PC can usually host thousands or tens

of thousands of application instances. Thus, P2PNet should also scale to hundreds or a few thousands of processors.

Each of the workers processes events and sends and receives events from other workers. The workload of a worker depends on the average event arrival rate R of virtual nodes and the number of the virtual nodes hosted by the worker. Therefore, we can simulate larger networks by simply adding more workers. Furthermore, each worker and sends and receives 2 synchronization messages for every monitoring period. The overhead of synchronization is both constant and negligible.

The possible bottleneck of P2PNet architecture is the coordinator as it communicates with all workers for synchronization. The coordinator processes $3W$ synchronization messages for every monitoring period T. Recall that when W is large, the time expansion factor is approximately $NR(a+b)/W$. Therefore, the message processing rate for the coordinator in real time is $\frac{3W}{(N/W)R(a+b)T}$. When the number of average virtual nodes per worker is fixed, the work load of the coordinator is proportional to the number of workers.

When N=106, W =1000, R= 1/10ms, $a+b$ = 0.2ms, T=20ms, the event processing rate is 7.5/ms. This is an affordable work-load by regular PCs.

The internal network bandwidth is another issue related to the scalability of P2PNet. The total number of messages generated during the simulation isNRP. Suppose that the average message length is L bytes. Since the minimum simulation execution time is$NRP(a+b)/W$, therefore, the average bandwidth needed for P2PNet is

$$\frac{NRPL}{NRP(a+b)/W} = \frac{WL}{a+b}$$

The required internal bandwidth does not depend on the size of the simulated network. It increases linearly with the number of workers. When W =1000, L is 40 bytes, $a+b$ = 0.2ms. The bandwidth required is 200 MB/s. Therefore, P2PNet can scale to thousands of worker with moderate requirement on the internal bandwidth.

```
public void sendMessage(QuickSerializable message, int dest)
public void broadcastMessage(QuickSerializable msg)
public Object[] receiveMessages();
public void sendPacket(Packet msg, int srcRank);
public void enQueueMessage(QuickSerializable msg, int dest);
```

Figure 3.6: API of the Lightweight Message Passing Interface

3.4 Implementation Details

This distributed simulation tool has been implemented in pure Java. It can be used in either a parallel computing environment or a set of connected computers. As discussed in Section 3.2, P2PNet is mainly composed of three parts: event scheduler, simulation network and the message delivery interface. The event scheduler is responsible for event processing and scheduling, and synchronization with the coordinator. The simulation network represents the underlying Internet and the message delivery interface is responsible for sending and receiving messages between LPs. In this section, the implementation details of each of the three components will be discussed.

3.4.1 Light-weight Message Delivery Interface

The message delivery interface builds communication channels between LPs. Each LP is assigned a unique id called its rank. LPs can send and receive events or synchronization messages. Currently, there is only one MPI (message passing interface) binding for Java (mpiJava). However, the serialization and de-serialization overhead of mpiJava is very high. In order to reduce communication overhead and thus improve simulation efficiency, a light-weight message passage interface was developed. This interface is using TCP as the underlying transport protocol. On top of the TCP protocol, we create a new library by which a host can send messages and receive messages from multiple hosts. The API is shown in Figure 3.6.

All the calls are asynchronous call. Function sendMessage is for sending message to a specific LP which is identified by the rank. It is not buffered in this layer. A

quick serialization message can contain a group of messages so as to reduce communication cost. Function broadcastMessage is used to send the message to all other LPs in the system. In this layer, we provide a buffered communication call enqueueMessage. Buffering the messages will increase the communication delay and reduce the overall messages. The users can specify the buffer size (the default buffer size is set as 5) through configuration file. In the simulation, all synchronization messages are sent through non-buffered calls and the event messages are sent through the buffered call.

The initialization of the message delivery interface needs to find the IP address and port of other hosts. An easy solution is to use a pre-determined configuration file. For computing environments where processors are assigned dynamically, we provide an initialization and host discovery method based on a shared file system. We treat each process as a host. Each host is identified by an IP address and a port. The port number is the socket server port each other host will connect to. We use a pair (address and port) to identify a host since a physical machine and have multiple processes. The initialization has three steps. Firstly, each host tries to bind a socket server port from a default port (10000). If binding fails (this means that some other process is using the current port), the host will increase the port number by an incremental interval (1000), sleep for a period time and bind again. The process will continue until the binding succeeds. After binding is successful, the host will create a new file in a predefined directory. The name of the file is IP address plus port. The host will constantly monitor the host list directory. If the number of files in the directory reaches the total number of hosts, it will read the files names and build a host list. Each of the hosts is assigned a rank number. Each host will use the host list to find and connect to other hosts.

```
public double getDistance(int src, int dest);
```

Figure 3.7: Topology Interface

3.4.2 Event Scheduler

The event scheduler is implemented as a single threaded application in order to avoid thread level context switch since it will interfere the time system. It is executing a loop to fetch an event from the event queue, process the event, get incoming messages from other LPs and process the messages. The message can be a synchronization message or an overlay message sent by the simulation instances.

As starting point, each worker will send a ping message to the coordinator. After the coordinator receives all messages from the workers, it will send a simulation start message to all workers and start the simulation. During the simulation, the coordinator will periodically send inquiry messages to all hosts to inquire about time expansion factors. In receiving the inquiry message, the worker will calculate the suggested time expansion factor and send it back to the coordinator. The coordinator collects the response messages from all workers, computes the maximum factor and the reference point and send them back to the hosts. The workers will update the time expansion factors and reference points based on the message from the coordinator. In sending factor inquiry reply messages to the coordinator, each worker also includes the number of events left in the event queue in the response message. If the number of events left is zero for all hosts, then the coordinator will send a simulation end message to all workers and the workers will stop execution upon receiving such a message.

3.4.3 Communication Network

The communication network represents the Internet in the simulation. As this simulation tool is only designed for overlay network simulation, it uses a simple internet

```
public void registerNode(EventHandlerIF node, int nodeId);
public double getDistance(int src, int dest);
public void deliverMessage(MessageIF msg, int destNodeId);
public void schedule(Object event, EventHandlerIF handler,
   double time);
public double currentTime();
public void start();
public int getSize();
```

Figure 3.8: The API for simulation

topology model. Internally, it uses an Internet topology model which can be generated by any topology generator. The topology provides a common interface to compute the distance between any pair of nodes. Users can implant the topology by implementing the following interface. Therefore, all the topology generators can be used including GT-ITM, Tier, Inet, etc. We also implement the topology generator designed in Chapter 4.

3.4.4 API of the Simulator

This simulator provides a rather generic interface for the simulation of overlay network applications although its initial purpose is for Peer-to-Peer simulation. The simulated instance registers itself with the simulator. The simulator will deliver messages to other nodes. Simulated instances can also schedule events with the simulator at certain time stamps.

3.5 Experiments

In this section, some simulation results are presented.

3.5.1 Performance Evaluation

We evaluated with a simple overlay application in which each node has a ping client. In the simulation, the ping client periodically picks a random node and sends a ping message to it. In receiving a ping message, the destination client performs some calculation (to simulate the event processing time) and replies to the ping message. The delay between two consecutive ping messages sent by a ping client follows a uniform distribution. The processing time of the event also follows a uniform distribution. As the topology has no effect of the simulation, we use a simple star topology and network size varies from 100 to 1000000.

3.5.1.1 P2PNet versus regular event scheduler

First, we compare the performance of P2PNet with one worker versus a sequential event-driven simulator (SEDS). As the sequential event-driven simulator will experience no communication overhead and thus takes the minimum time to perform a simulation, it is used as the benchmark. Compared with sequential event-driven simulator, the overhead caused by using real time to guide the event processing includes:

- Empty loops due to larger than necessary TEF.
- Monitoring of the execution of the event scheduler, especially the system calls to get the system time.

In the experiment, the simulation is run for 1000ms (logical time). In order to eliminate the effect of the initial value of TEF toward the simulation execution time, the initial value of the TEF is set as 1. The other parameters are set as: $\alpha_l = 0.9$, $\alpha_h = 0.95$, $\beta_l = 1.0$, $\beta_h = 1.0$, $\theta = 0.2$. The monitoring period is set as 20 ms.

Table 3.1, 3.2, and 3.3 show the total execution time (real time) for both P2PNet and a sequential event-driven simulator. It shows that the ratio between the execution time of P2PNet and that of a sequential event-driven simulator decreases

Table 3.1: Simulation time of the P2PNet versus SEDS (Network Size 1000, average between-message delay 50 ms)

Average message processing time (ms)	Simulation execution time (ms)		Ratio
	SEDS	P2PNet	
0.01	848	1432	1.689
0.02	1214	2035	1.677
0.05	2481	3549	1.430
0.1	4331	6606	1.525
0.2	8317	10877	1.308
0.5	18766	26310	1.402
1	36107	50031	1.386
2	71822	93551	1.303

Table 3.2: Simulation time of the P2PNet versus SEDS (Network Size 1000, average event processing time 0.1 ms)

Average between message delay (ms)	Simulation execution time (ms)		Ratio
	SEDS	P2PNet	
200	950	1872	1.971
100	2360	3237	1.371
50	4523	6540	1.446
20	10762	13877	1.289
10	20930	30368	1.451
5	43083	57911	1.344
2	105600	131352	1.244

as the simulation load increases, approaching 1.35 in high work load situations. As P2PNet is aimed at high work load situations, an overhead of 35% is acceptable. Future work will be investigating techniques to reduce the overhead.

3.5.1.2 Speedup of P2PNet

Secondly, we compare the performance of P2PNet with multiple workers to that of a sequential event-driven simulator. The simulation was conducted on a cluster of computers. In this simulation, the network size is 1000000 and the average event arrival rate is 1/100ms. As discussed in Section 3.3, the speedup rate is mainly

Table 3.3: Simulation time of the P2PNet versus SEDS (Network Size 1000, Average between-message delay 50 ms, average message processing time 0.1ms)

Network size	Simulation execution time (ms)		Ratio
	SEDS	P2PNet	
100	527	1537	2.917
200	808	1874	2.319
500	2138	2984	1.396
1000	4457	6540	1.467
2000	8546	12008	1.405
5000	20981	30400	1.449
10000	42672	57591	1.350

determined by the number of workers and the relative communication cost (the cost of sending and receiving an event divided by the event processing time). In this experiment, average event processing times of 0.1ms and 0.5ms. The cost of sending and receiving a message is around 0.35ms. We use a buffer size of 5. Therefore the average communication cost for an event is 0.07ms. Figure 3.9 shows the speedup of P2PNet versus the number of workers with different average relative communication cost. In the range of 1 to 30 computers, the speedup increases linearly with the number of workers at both situations. In all the simulations, P2PNet has no late events.

Due to lack of computing facilities, we were not able to evaluate the speedup of P2PNet on larger computer clusters. Given a simulation workload, the speedup rate of P2PNet will eventually reach its limit when the number of workers increases. The limit of the speedup rate of P2PNet is determined by the communication cost and the fluctuation of time expansion factor. Future work is to evaluate the speedup rate in larger cluster to fully understand the speedup rate of P2PNet.

The slopes of the speed up are different. Table shows the estimated speed up slope versus the measured slope. The estimated slope is calculated as$1/(1 + b/a)$. Compared with the estimated speed up slope, P2PNet has a low overhead of around 10% in either case.

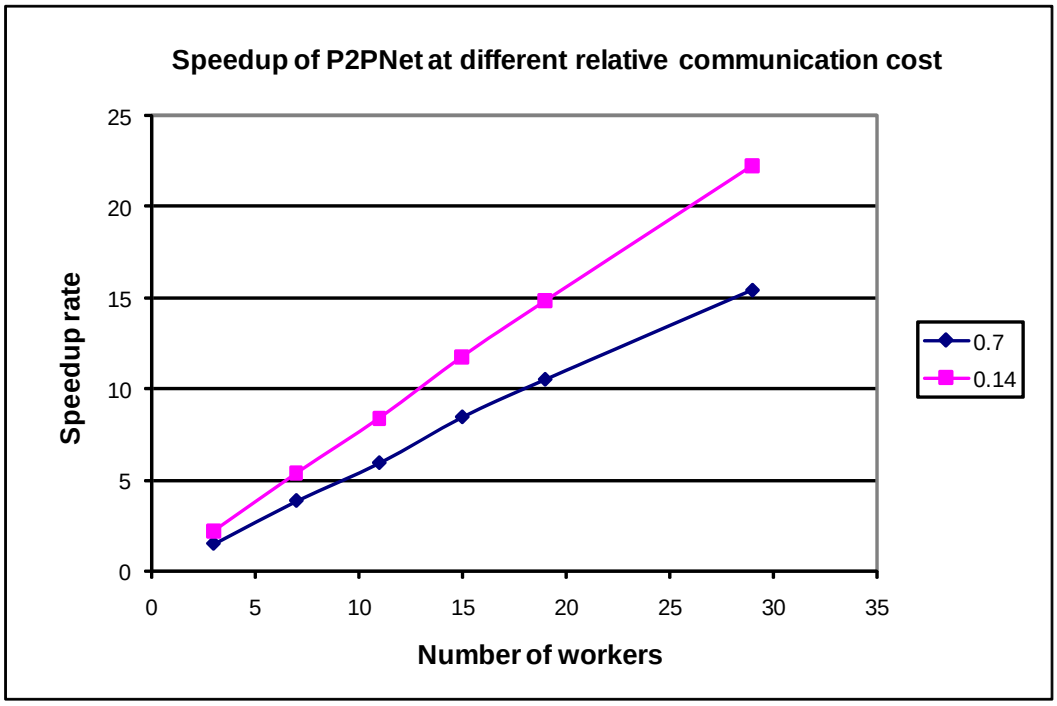

Figure 3.9: Speedup of the P2PNet versus regular event scheduler (network size 1000000, average event arrival rate 1/100ms).

Table 3.4: Estimated speed-up slope versus measured speed up slope

Average event processing time	0.1ms	0.5ms
Speed up slope	0.537	0.774
Estimated speed up slope	0.588	0.877
Overhead	9.54%	13.3%

3.5.2 Simulation of Pastry with P2PNet

In order to test the usability of P2PNet, we use it to simulate a typical Peer-to-Peer application, Pastry in a network of 100,000 nodes. In this simulation, 20 processes were used and each process hosted 5000 Pastry nodes. The experiment was carried in glacier of Westgrid high performance computation facility [10]. Westgrid is a large computing facility operated across west Canada. It provides a high performance computing (HPC), collaboration and visualization infrastructure for research purposes. Glacier provides an IBM blade server cluster of 840 servers with dual processors connected through gigabit Ethernet.

The source code of Pastry 1.14 was modified so as to use P2PNet. The modification of Pastry is mainly the implementation of an object serialization interface on the object types that are used in events transferred among LPs. It does not involve significant change in the algorithm or data structure of the Pastry source code. It demonstrates that P2PNet can be used to simulate other P2P systems or overlay networks without major modification to the original system. The simulation has two stages. In the first 10 seconds, 100,000 nodes will join the network randomly. After 20 seconds in simulation time, the nodes will send 10000 ping messages to random destination nodes. The simulation is carried round by round and each round is 20 ms in virtual time. The simulation results are as follows:

Event Insert: Figure 3.10 shows the events inserted, processed and total events in the queue at each round. The events processed in each round represents the simulation work load at each of the rounds. The more events the simulator processes in a round, the heavier the work load in that round, and the higher the expansion factor should be in that round. Figure 3.10 also shows that there is a steep increase of the work load at $0 - 500$ ms and a steep decrease of work load at around 10000ms. The simulator should adapt to these work load variations by adjusting the time expansion rate.

Time System: Figure 3.11 shows the time system of the simulation process.

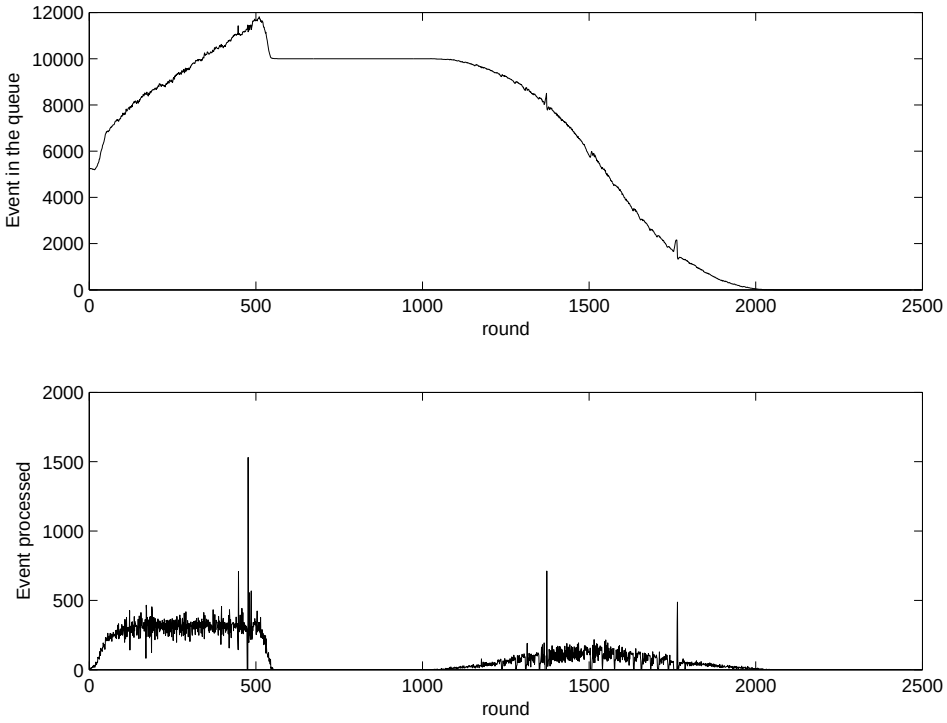

Figure 3.10: Events processed in each round during the simulation

Figure 3.11 shows that during the simulation, the virtual time closely follows the lower bound time stamp.

Adaptation Process: The adaptation process (adjusting the time expansion factor based on the work load) determines the performance of the simulator. The calculation of time expansion factor relies on two factors: logical time delay and empty rounds. Figure 3.12 shows the logical time delay and the empty round rate of LP 0 in the simulation. It can be seen that in most simulation rounds (98.4%), the logical time delay is close to zero. However, when the time expansion factor is low and the working thread is not executed for a long period of time, high logical time delay occurs. For instance, at the 1366th round, the working thread of the simulation was not run for 2828 milliseconds. As the time expansion factor is only 16, this caused a logical time delay of 176 milliseconds for 6 rounds. It is difficult to deal with this situation since the adaptation algorithm cannot distinguish between high simulation

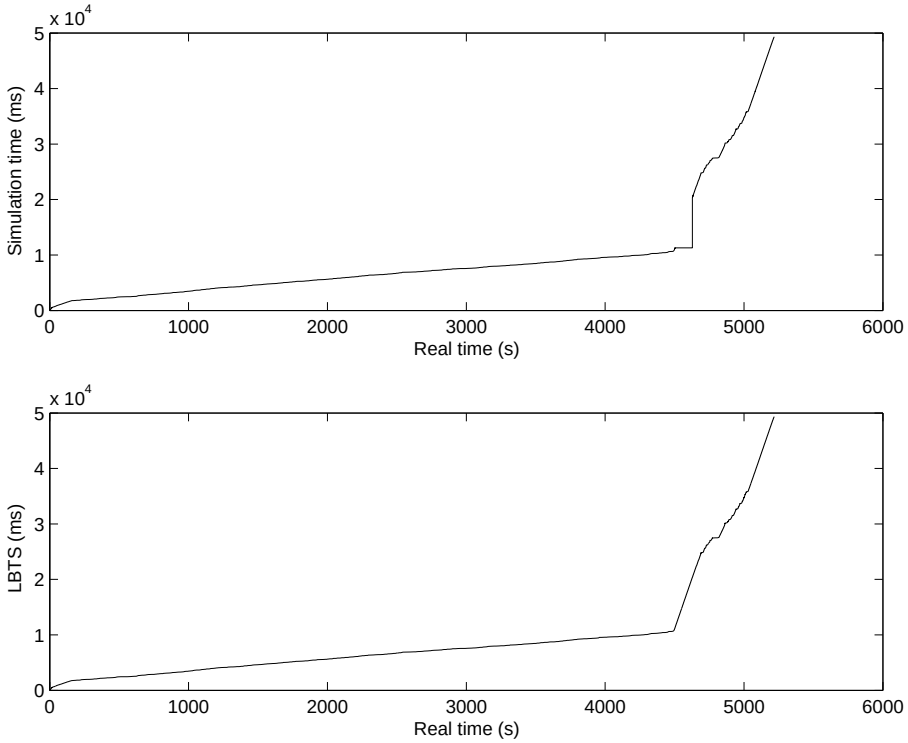

Figure 3.11: Simulation time and up-bound versus real time

Table 3.5: Logic time delay distribution

Logic time delay	< 1	$1 - 20$	20 -100	100+	Total
Rounds	2428	17	7	17	2467
Percentage	98.4%	0.7%	0.2%	0.7%	100%

work-load from a thread not running. However, when the time expansion factor is high, the logical time delay will become a lot less severe.

Figure 3.12 shows the empty loop rate. There are two time periods (600 – 1000 rounds, and 1000 – 2400 rounds) during which the work load is low. The expansion rate is kept as the minimum value, the empty rate is high (98%). During the first 600 rounds when the simulation load is high, the average empty rate is about 78%. As we set the redundancy rate as 1.2, the average empty rate should be close to 0.17. However, there is extra redundancy due to other reasons. As the time expansion rate is set as the maximum of all LPs, it is very likely that the actual expansion rate is a lot higher than necessary. Besides, the uneven distribution of events in the virtual time can also cause some empty rounds.

Synchronization of LPs: The synchronization of LPs is the critical part of the simulator. Figure 3.13 shows the time expansion factor versus simulation round. It shows that the time expansion factor adapts well to the simulation work load. When the simulation work load is high during rounds 0 – 500, the time expansion factor is high. The time expansion factor is kept at the minimum allowed value 3.3 during rounds 600 - 1000 when there are no events to be processed. However, this only accounts for 100 seconds in real time in the overall simulation (the total simulation takes about 5316 seconds). As discussed in Section 3.3, the overall time expansion factor is set as the maximum of the suggested time expansion factor returned by all LPs. The lower part of Figure 3.13 shows the ratio of the maximum and the minimum time expansion factor returned by LPs. Table 3.6 shows that the average ratio of maximum versus minimum time expansion factor is only 2.58. This shows the work load balance of different LPs.

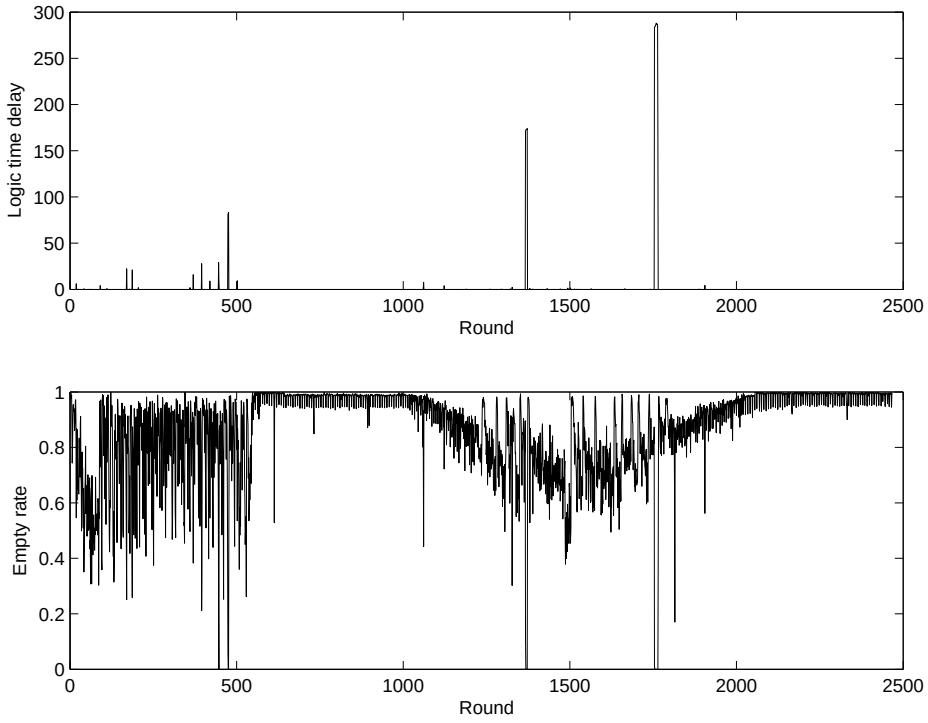

Figure 3.12: Logical time delay and simulation empty rate

Table 3.6: Mean and variance of time expansion factor ratio

Round	0-600	600-1000	1000-2000	2000-2318
Average ratio	2.54	1	1.28	1.01
Variance of TEF ratio	2.81	0	1.12	0.07

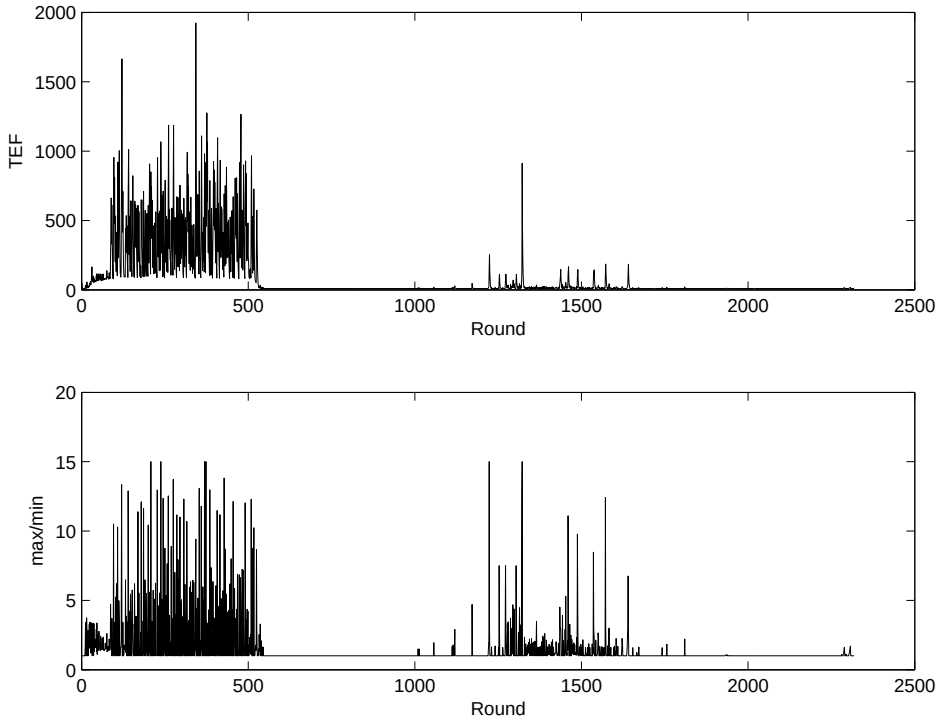

Figure 3.13: Time expansion factor versus simulation round

Communication between workers and the coordinator: The synchronization of P2PNet is carried out periodically for every monitoring period (round). At each round the coordinator sends query messages to the workers and the workers send back response messages to the coordinator. The coordinator will discard late response messages based on the configurable maximum allowed delay. The synchronization algorithm will fail if the delay of the response messages is too large. Figure 3.14 shows the delay between the query message and the last response sent by the workers and the number of discarded response messages. Table 3.7 shows the distribution of response delay and discarded response messages. It shows the synchronization delay is high when the simulation work is high. As the minimum allowed delay is set proportional to time expansion rate (and thus the simulation work load), the rate of discarded messages is very low. Therefore, the delay of response messages will not affect the synchronization algorithm.

Table 3.7: Average response delay and discarded response messages

Round	0-600	600-1000	1000-2000	2000-2318
Average response delay	324.8	4.0	16.8	3.74
Average discard message	0.05	0	0.013	0

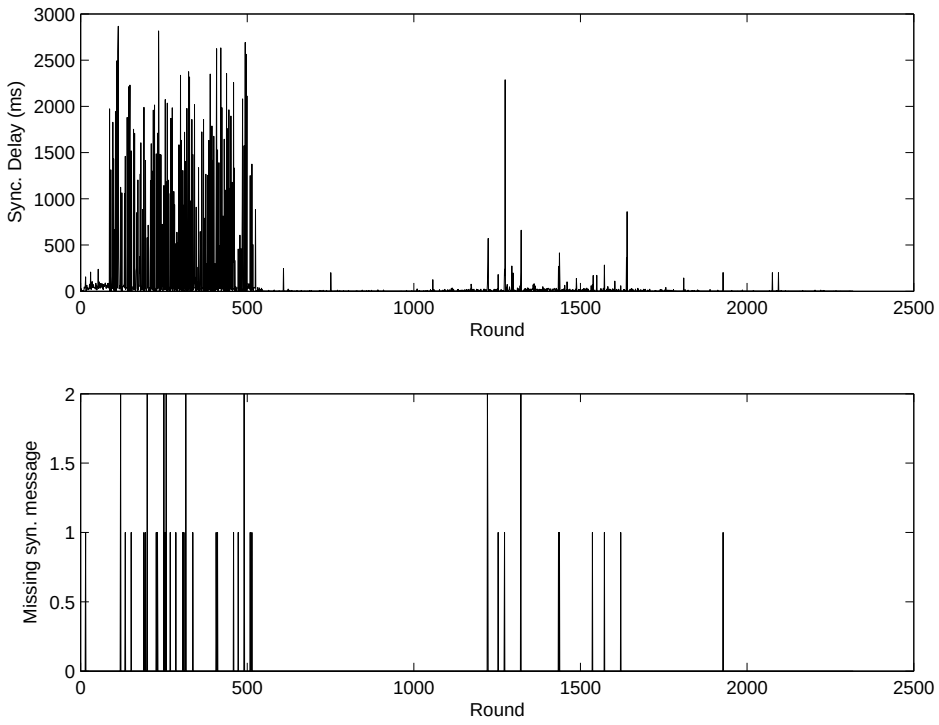

Figure 3.14: Synchronization delay and discarded messages

3.6 Conclusions

In this part, we describe a message-level simulator for P2P systems that uses real time to synchronize the event processing of the processors. The technique dynamically adapts to different simulation situations. Simulation results show that the simulator does not have large overhead compared with a sequential event-driven simulator. When used in a cluster of computers, it provides high speed-up in high work load situations without introducing any late events. P2PNet is also used to simulate Pastry at a large scale. The simulation results demonstrate the usability of P2PNet. Currently, there are no message level parallel simulators. Implementing the current parallel simulation algorithms and compare them with P2PNet will be time consuming. Therefore, in evaluating the performance of P2PNet, we only compare P2PNet with the benchmark.

There is much more work to be done on the foundation that we have laid. First, we observe some oscillation in the value of the TEF during simulation, which leads to performance degradation. We would like to understand the cause of this oscillation and find a better algorithm for adjusting the TEF. Second, more extensive evaluation needs to be done. We will be testing the simulation on various overlay applications such as Pastry and Chord, and using it to evaluate some optimization to their protocols. Third, because of lack of computer facilities, we have to date only performed simulations on a small cluster of 30 processors. We plan to test the simulator in a larger cluster to evaluate its extensibility to hundreds of processors.

Chapter 4

RealNet: A Topology Generator Based on Internet Topology

4.1 Introduction

In designing Internet applications such as peer-to-peer file sharing and content distribution applications [132][150][119][125], extensive simulation are usually performed to evaluate the efficiency and performance of the designed applications. Simulations usually use an abstraction or model of the Internet. In order to draw accurate conclusions from the simulations of an Internet application, the chosen topologies must capture fundamental properties of the Internet which are relevant to the application.

This section is focused on providing a scalable and representative topology generator for the large-scale simulation of Internet applications, mainly P2P applications. Large-scale message-level simulations are important for large-scale Internet applications because problems may only appear at scales of millions of participating end-hosts. For most P2P applications, an abstraction model of the Internet is used to calculate the routing path performance (latency, packet loss rate, etc.) between a pair of nodes.

4.2 Requirements

The requirements of such topology generator are:

1. **Scalable:**When the number of nodes in the network reaches millions, the memory and computation efficiency generation of topologies and calculation of routing paths become a very important issue. Currently, most topology generators only aim at providing synthetic topologies for small scale simulations. In a synthetic topology generator, edges between nodes are assigned weights and an edge-state based algorithm or a distance vector algorithm is used to find the routing path from one node to another. The routing table can be computed before-hand and loaded into memory or computed on-demand. In a topology with N nodes and E edges, the computation cost of the Dijkstra algorithm for calculating the routing table is $O(N^2 + E)$.

2. **Representativeness:**In order to draw accurate conclusions from the simulation results, the topology used in the simulations must capture the fundamental properties of the application. However, it is very hard to evaluate the representativeness of a topology. The reasons are: The real structure of the Internet is not well known. It is well known that at the AS level structure, there are many missing edges and routing policies. In the router level, it is very difficult to find all the routers. It is very hard to extract properties from the Internet. From graph theory, there are many different metrics to measure the properties of a graph. We generally use a subset of the possible metrics to measure; the only decision we have is to find the properties that are important for the simulation results. However, this is also challenging. For most P2P applications, the delay between participant nodes as well as their geographical distribution is important.

Currently, there are many synthetic Internet topology generators. GT-ITM[25] and Tiers[46], Nem[88] and BRITE[97] etc. Structural topology generators such as

GT-ITM tried to capture the hierarchical structure of the Internet. However, they generate domains and edges among domains and routers purely randomly. They cannot reflect connectivity characteristics of the Internet either in AS level or router level. Power-law based topology generators [109][12][18][13][23] try to generate topologies which reflect the power-law property in AS-level connectivity. These topologies fail to reflect either the layered structure of the Internet or the routing policy among ASes, or other connectivity characteristics yet to be discovered.

4.2.1 Our Solution

We consider this work to be the first effort to develop a representative topology generator which can generate large-scale Internet topologies with up to millions of nodes. As the computation of routing paths is important for large-scale simulations, the computation of routing paths should be efficient. Our main method is to extract the AS-level, and router-level Internet topology, and access links of the Internet and simplify them to obtain a simplified Internet model. The topology generator also considers the Autonomous System (AS) level routing policy. The AS topology and router topology form the Internet routing core. The topology generator can generate topologies with any number of nodes by attaching end-hosts to gateway routers. The hierarchical routing also guarantees that the routing path computation is scalable and effiicient.

4.3 Modeling the Internet

The Internet consists of a large collection of hosts interconnected by networks of edges and routers. It is divided into tens of thousands of ASes. Routing between the ASes is determined by inter-domain routing protocols such as the Border Gateway protocol (BGP). BGP employs the simplest path-vector protocol and selects a route with the shortest AS path. However, BGP allows each AS to choose its own routing policy

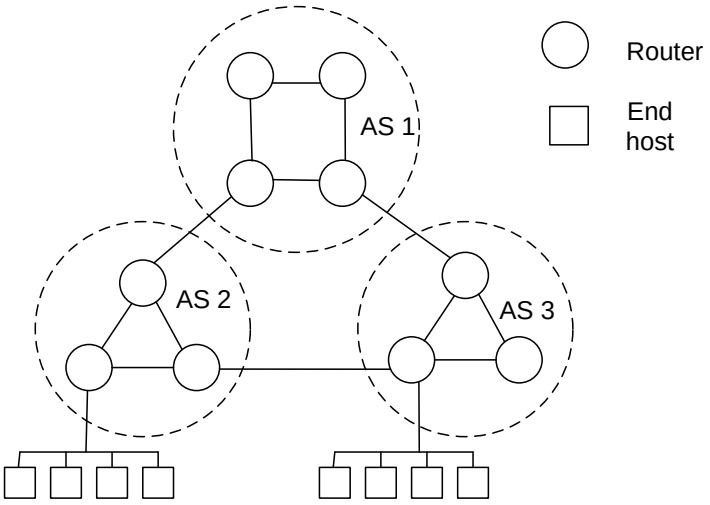

Figure 4.1: The Model of the Internet

in selecting the best route, announcing and accepting routes. Routing policies are usually determined by the commercial contractual relationhips between pairs of ASes. Because of the local routing policies in BGP routers, the AS level paths from a source AS to different IP prefixes in the same destination AS can be different.

Each domain consists of a set of routers and end hosts. Routers form the routing core. The edge between routers are assigned weights. Routing within an AS is controlled by some intra-domain routing protocols such as OSPF, IS-IS, and RIP. Intra-domain routing protocols use either distance vector algorithms or edge state algorithms to compute routing paths. End hosts are grouped into IP network prefixes. Each end host connects to the Internet through a gateway router. Figure 4.1 illustrates the Internet topology.

For modeling the Internet structure, the most accurate Internet topology model is the Internet itself. However, Internet structure is too complicated to be captured by a simple model. There are tens of thousands of ASes and millions of routers. The routing path not only depends on the router-level connectivity graph, it also depends on the routing policies of ASes. Besides, some AS edges, routers or local routing policies in BGP routers cannot be observed or inferred from Internet measurement

or published documents.

The focus of this work is to build a simplified model of the Internet and use the inferred model to generate topologies for computing routing path performance (latency, loss rate, etc.) in message-level simulations. In modelling the Internet, we try to capture the main chararteristics of both the structure and the routing. Our model is as follows:

Structure: Our model has the same hierarhical structure as that of the Internet. It is composed of a number of ASes. Each AS consists of a number of routers and end-hosts. End-hosts are grouped together to form IP networks. Each network is connected to the Internet through a gateway router.

In the Internet, there are tens of thousands of ASes. The number of ASes and edges between ASes are relatively stable. Instead of extracting AS connectivity characteristics of (such as power-law in out degree) the Internet and generate ASes based on the extract characteristics, we directly use all the ASes and AS edges in the Internet. By doing so, we preserve all the connectivity characterisitcs of the Internet in the model.

In each of the ASes, there are a number of routers. Overall, the Internet can have tens of millions of routers. A big ISP such as AT&T can have tens of thousands of routers. This will cause high computation cost in calcuating routing paths within the AS. Besides, some routers and router edges can not be revealed by traceroute probing. However, almost all PoPs and PoP edges can be revealed by traceroute data. From the routing and performance perspective, routers in the same PoP (point of presence) have similar behavior. In order to simplify the topology, we cluster router interfaces into PoPs. iPlane proposed a clustering algorithm and a mapping from router interfaces to PoPs. We adopt both the algorithm and the mapping in this work. After clustering the IPs into PoPs, we obtain a compact Internet routing core with PoPs and PoP edges.

Routing Policy: The routing of our model is also hierarhical. In order to

simplify the routing path calculation, we make the simplifying assumption of uniform routing policy within an AS. With this assumption, the routing path from a source AS to all IP prefixes in the destination AS will be the same. The interdomain routing is mainly determined by the contractual relationship between ASes. We ignore the local routing policies in BGP routers and assume that two neighboring ASes usually have one of four common relationships: customer-to-provider (C2P), provider-to-customer (P2C), sibling-to-sibling (S2S), or peer-to-peer (P2P). The routing algorithm will pick the shortest policy path. Within an AS, PoP edges are assigned weight based on the latency. An edge state routing algorithm is used to find the PoP routing path within the AS.

4.4 AS Graph

In this section, the fundamental about AS topology is defined. Some theorems are presented so as to provide theoretical foundation for the following design of algorithms.

Definition 4.1: An **AS graph** $G(V, E)$ is a directed graph in which $V(G)$ is the set of ASes and $E(G)$ is the set of AS edges (ordered pairs of ASes). AS graphs are ***symmetric graphs*** in that if an edge $e = (x, y)$ is in G, then the inverted edge $\overline{e} = \{y, x\}$ is also in the graph. AS graphs are ***weighted graphs***and every edge is assigned with one of the four types of relationship (P2C, C2P, S2S or P2P). A relationship function $R(e)$ represents the relationship of an edge e. $R(e)$ has the following property:

$$R((y,x)) = \begin{cases} P2C, & if\ R((x,y)) = C2P \\ C2P, & if\ R((x,y)) = P2C \\ P2P, & if\ R((x,y)) = P2P \\ S2S, & if\ R((x,y)) = S2S \end{cases}$$

An edge e with relationship S2S can be replaced with two edges $e1 = (x, y)$ and

$e2 = (y, x)$ and $R(e1) = C2P$ and $R(e2) = P2C$. A graph $G_1(V_1, E_1)$ is a ***subgraph*** of graph $G(V, E)$ if $V_1 \subset V$, $E_1 \subset E$.

Each AS is assigned a number called ASN (Autonomous System Number) which is an unsigned integer of 2 bytes (0-65535). In the following ASN will be used interchangeably with AS.

Definition 4.2: An AS path is a unique AS sequence, i.e., $P = (v_0 v_1 v_2 \cdots v_n)$ and for any two ASes $v_i \neq v_j, if\ i \neq j$. The length of the path is the number of ASes in the path. A path with only one AS has the length of 1. In an AS path P, (v_i, v_{i+1}), $i = 0, 1, \cdots, n-1$ are AS edges. For two paths $P1 = (v_0 v_1 v_2 \cdots v_n)$ and $P1 = (u_0 u_1 u_2 \cdots u_m)$ and $u_0 = v_n$, the ***union*** of two paths is $P1 \cup P2 = (v_0 v_1 v_2 \cdots v_n u_1 u_2 \cdots u_n)$. The ***union*** of a path P and an edge $e = (x, y)$ is $P \cup e = (v_0 v_1 v_2 \cdots v_n y)$ if $v_n = x$. The union of edge e and path P is $e \cup P = (x v_0 v_1 v_2 \cdots v_n y)$ if $v_0 = y$.

Definition 4.3: An AS path is valid (valley-free) if any P2P edges or P2C edges are only followed by P2C edges. A path with only C2P edges is called an ***uphill path*** and a path with only P2C edges are called a ***downhill path***. A valid path can be of one of the following two patterns:

Pattern A: $P = P1 \cup P2$, where P1 is an uphill path and P2 is a downhill path.

Pattern B: $P = P1 \cup e \cup P2$, where P1 is an uphill path and P2 is a downhill path and e is a P2P edge. Either P1 or P2 can be of length 1.

Definition 4.4: In a AS graph $G(V, E)$ AS u is called ***reachable*** from node v if there is a valid path P existing in G from u to v. If for any two nodes u and v in AS graph $G(V, E)$, u is reachable from v, then we call graph G ***connected***.

Theorem 4.1: Given a valley-free path $P = (v_0 v_1 v_2 \cdots v_n)$, the inverse path $\overline{P} = (v_n v_{n-1} v_{n-2} \cdots v_0)$ is also a valid path.

Proof: According to definition 4.3, a valid can be of two patterns. If P is of pattern A, then $P = P1 \cup P2$. $\overline{P} = \overline{P2} \cup \overline{P1}$. As $P1$ is an uphill path, $\overline{P1}$ is a downhill. As $P2$ is a downhill path, $\overline{P2}$ is an uphill path. Therefore, $\overline{P}$ is of pattern

A and it is a valid path.

If P is of pattern B, then $P = P1 \cup e \cup P2$. $\overline{P} = \overline{P2} \cup \overline{e} \cup \overline{P1}$. $\overline{P2}$ is an uphill path, $\overline{P1}$ is a downhill path, and $\overline{e}$ is a P2P edge. Therefore, $\overline{P}$ is of pattern B and it is a valid path.

Theorem 4.2: Given a valley-free path $P = (v_1 v_2 \cdots v_n)$, and an edge $e1 = (x1, y1)$ $e2 = (x2, y2)$. If $R(e1) = C2P$ and $y1 = v_1$, then the union of e1 and $e1 \cup P$ is also a valid path. If $R(e2) = P2C$ and $v_n = x2$, then the union of e1 and $P \cup e2$ is also a valid path.

Proof: According to definition 4.3, a valid path can be of two patterns. If P is of pattern A, then $P = P1 \cup P2$. $e1 \cup P = e1 \cup P1 \cup P2$. As $P1$ is an uphill path, e$\cup P1$ is also an uphill path. As $P2$ is a downhill path, $e1 \cup P$ is of pattern A and it is a valid path. If P is of pattern B, $e1 \cup P = e1 \cup P1 \cup e \cup P2$. Since $e1 \cup P$ is an uphill path and $P2$ is downhill path. $e1 \cup P$ is of pattern B and still a valid path. It can be proven with the same reasoning as above that $P \cup e2$ is a valid path.

Theorem 4.3: For a ASes graph $G(V, E)$, any two nodes are reachable. If a new AS N has a C2P edge to one node M in AS set V, then the new graph $G(V \cup \{N\}, \quad E \cup (N, M))$ is fully reachable.

Proof: Suppose that N has a provider M in set S. Therefore, for any other node B in V, there is always a valid path $P = (M v_1 v_2 \cdots v_n B)$ from M to B. Based on theorem 4.2, the new path $(N, M) \cup P = (N M v_1 v_2 \cdots v_n B)$ is a valid path. Therefore, any node in B is reachable from N. For the same reasoning, there is a valid path $P_2(B, M) = (B r_1 r_2 \cdots r_n M)$. The new path $P \cup (N, M) = (B r_1 r_2 \cdots r_n M N)$ is a valid path. Therefore, N is reachable from any AS in V. Therefore the new graph on vertex set $N \cup S$ is connected.

Theorem 4.4: Given a valley-free path P. If P is of pattern A, where $P = P1 \cup P2$, $P1 = (s r_1 r_2 \cdots r_n p)$ and $P2 = (p r_2 \cdots r_n d)$. P1 is the shortest uphill path from s to p and P2 is shortest downhill path from p to d. If P is of pattern B, where

$P = P1 \cup e \cup P2$, $P1 = (sr_1r_2 \cdots r_np)$ and $P2 = (mr_2 \cdots r_nd)$. $P1$ is the shortest uphill path from s to p and P2 is the shortest downhill path from m to d.

Proof: If P is of pattern A, $P = P1 \cup P2$. If there is another uphill path $P3$ which is shorter than $P1$, then the path $P3 \cup P2$ is a valid path which is shorter than P. This conflicts with that assumption that P is the shortest valid path. Therefore, P1 is the shortest uphill path from s to p. The other conclusion can be proven similarly.

4.5 Inferring Internet Topology

In order to build a network topology that is similar to the Internet, we will first try to infer the Internet structure from various data sources. In the AS level, we need to find ASes and edges and also infer the relationship between ASes. In the router-level, we need to find the routers and router edges and infer the edge weight among the routers. For the end-host, we need to find the gateway router it connects to.

4.5.1 Data Source

Data concerning Internet routing falls into three categories: routing data collected through BGP trace collectors, trace-route based data and Internet routing registry data.

BGP trace:A BGP trace collector is a BGP router which sets up peering sessions with commercial ISP BGP routers and receives BGP messages from its peers. Periodically, the collector saves its full routing tables and routing updates received from its peers. RouteView [6] and RIPE RIS [7] are two major projects that deploy BGP collectors and make their BGP trace data available to public. The BGP routing table provides the AS path to a particular IP prefix. From the AS path, we can find ASes and AS edges. AS paths can also be used to infer the relationship between AS peers. Table 4.1 shows the number of BGP collectors and peering AS and routers

	Ripe RIS	RouteView	Total
Collectors	15	7	22
Peer AS	366	81	447
Peering Router	619	147	763

Table 4.1: Number of BGP collectors and Peering AS of RouteView and Ripe RIS

	Monitors	Countries
iPlane	186	36
Dimes	17233	111
Caida	22	16

Table 4.2: Traceroute Monitors of iPlane, Dimes and Caida

of RouteView and Ripe RIS. The more peering ASes and routers we have, the more ASes and AS edges we can find for the Internet. In this work, IPv4 peers are counted.

Traceroute based data: Traceroute captures the IP path from the source to a given destination by sending either UDP or ICMP probe packets. Various projects make their trace-route records available to the public or only researchers. In this work, trace-route data from DIMES [127] and iPlane [86] are used to infer both AS level topology and router level topology. iPlane provides raw traceroute data. Dimes aggregates the router level path and provide a list of routers, nodes, and edges between nodes. We intend to use trace-route data from more source in the future work so as to improve the accuracy of inference of the Internet's strucuture. The more data we have, the more AS and router edges we can find. Traceroute data serves three purposes: 1) AS level paths can be deduced from router level paths given mappings from IP to AS. These AS paths can be used together with BGP traces and IRR data to infer AS level topology and relationships. 2) Traceroute can be used to find the router-level topology of the Internet. 3) It can be used to infer access edges (edges between an end host and its gateway router).

Internet Routing Registries: The purpose of the IRR is for operators to coordinate global policy settings. Network operators may register routing policies with the IRR. The databases that form the IRR are manually maintained by opera-

tors, mostly on a voluntary basis. Information therein may be incorrect, incomplete, or out-dated. The RIPE portion of the IRR is actively used by ISPs in Europe to filter route announcements and many European exchange points require operators to register with RIPE. Consequently, it is considered the most reliable information in the IRR.We use only the RIPE portion of the IRR for the topology in this paper.

4.5.2 Inferring AS-level Topology

From the BGP routing table, we are able to find a list of AS level paths to a network prefix. As the network prefix is in the last AS in the AS path, we can also build a mapping from network prefix to AS. Another source of AS level paths is the traceroute data. Each traceroute record provides a router level path from a source IP address to a destination IP address. With a mapping from IP address to AS, we will be able to find the corresponding AS level path. Providing an accurate IP to AS mapping is a very challenging task. Mapping IP addresses into AS numbers (using RouteViews BGP data) involves potential distortion due to IP prefixes advertised by: 1) AS-sets (an aggregated set of ASes advertises the prefix); 2) multi-origin ASes (MOASes), which means that several separate ASes advertise the same prefix; 3) missing IP to AS (some IP addresses appear in topology probes but are not advertised by any AS). In this work, the mapping from IP prefix to AS-set is discarded. With the collection of AS routing paths, one will be able to find the collection of ASes and edges between ASes.

In this work, AS topology is inferred from 4 data resources: Caida, Routeview, Dimes, and iPlane. Table 4.2 shows the AS nodes and edges combined from these 4 data sources. RouteView topology is used as the bench mark since it is the most accurate. Our mission is to collect a topology with as many AS nodes and edges as possible.

Table 4.3 shows that data from iPlane and Dimes contribute a significant number of AS edges. It shows that by using traceroute data, one can find a lot

	Data sources	AS nodes	AS edges	Unique AS	Unique edges
Routeview	BGP table	28200	58171	458	2360
Caida	Traceroute	27554	55481	40	705
iPlane	Traceroute	21373	81517	30	9774
Dimes	Traceroute	21223	61399	618	23062
Total		29132	97598		

Table 4.3: AS topologies inferred from various data sources

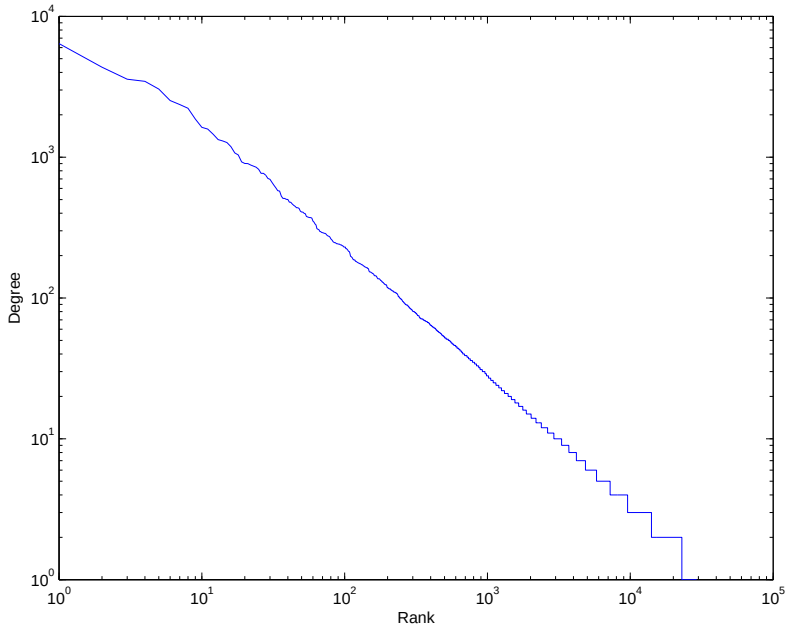

Figure 4.2: Distribution of ASes Rank

of AS edges that can not be found from BGP trace data. This also confirms the value of using traceroute data to infer the AS level topology of the Internet. Figure 4.2 shows the distribution of the ASes ranks.

4.5.2.1 Inferring AS-level Relationships

In the Internet, ISPs are rarely willing to reveal their commercial relationships with their neighbors. Several algorithms have been proposed for inferring AS relationships from AS paths. Most of the algorithms are based on the valley-free property of AS paths. A valid AS-level path is valley-free, which means that after traversing a P2C

	Caida	RouteView	iPlane
Caida		54238/51708 (95.3%)	30613/28880 (95.7%)
Routeview	51708/54238 (95.3%)		31667/30221(95.4%)
iPlane	30613/28880 (95.7%)	31667/30221 (95.4%)	

Table 4.4: Common AS paths and relationships extracted from various data sources

	C2P	S2S	P2P
Caida	51738	214	3529
Routeview	54942	575	2654
iPlane	43279	563	5125
Combined	68216	888	5432

Table 4.5: AS paths and relationships extracted from various data sources

or P2P edge, the AS path can not traverse a C2P or P2P edge. It can be also inferred that any AS path always has one or zero P2P edge. With the AS paths in hand, we are able to infer the relationships between neighboring ASes. Any AS relationship algorithm can be used.

In the AS topology inferred, Caida provides the AS relationship of all the edges. For the AS topology inferred from RouteView and iPlane, we use the algorithm in [56] to infer the AS relationships. Table 4.4 and 4.5 show the AS paths and relationships from the Caida, RouteView and iPlane data. Table 4.4 shows that although the algorithm used in Caida and RouteView are different, the inference results are very close. The majority (95.3%) of the relationships are the same. iPlane and RouteView uses the same algorithm but different data sets; 95% of the relationships are the same. This suggests the inner upperbound of the accuracy of AS relationship inference based on the valley-free property. In order to aggregate the AS topology, we use Caida as the base and add new edges from RouteView, and then iPlane to the topology as the Caida algorithm seems to be more accurate than that in [56] and the AS path in RouteView is more accurate than that of iPlane.

	T1	**T2**	**Comp**	**Edu**	**NIC**	**IX**
T1	Unknown	Unknown	P2C	P2C	P2C	P2C
T2	Unknown	Unknown	P2C	P2C	P2C	P2C
Comp	C2P	C2P	Unknown	Unknown	Unknown	P2P
Edu	C2P	C2P	Unknown	P2P	P2P	P2P
NIC	C2P	Unknown	Unknown	P2P	P2P	P2P
IX	C2P	C2P	P2P	P2P	P2P	P2P

Table 4.6: Relationship Estimation Rules

4.5.2.2 Estimation of Unknown AS Relationship

Table 4.3 shows that Dimes provides a lot of unique AS edges not seen by other data sources. However, they did not provide the relationship of the AS edges that they discover. As Dimes does not provide AS path information, it is impossible to infer the relationship using existing AS relationship inference algorithms. In order to infer AS paths, we need to estimate the relationship of those edges with unknown relationship.

So all AS relationship inference algorithms are based on three patterns of information: routing policy inferred from Internet Routing Registry, degree and description of ASes, or AS paths. Without information about the routing policy and AS paths, we will solely rely on degree and role of the AS to infer the relationship about the relationship about two ASes. For each AS, we classify the AS into a set of categories: large ISPs (T1), small ISPs (T2), customer networks (comp), universities (edu), Internet exchange points (IX), and network information centers (NIC). [44] provides both the algorithms and the classification results of ASes. The classification is based on the description of the ASes extracted from IRR, and the number of customers, and the number of claimed IP prefixes.

Table 4.6 provides the rules for estimating relationship of an edge based on the categories of the peering ASes. The rows represent the source AS of an AS edge and the columns represent the destination AS of an AS edge. If the relationship in one cell in the table is "unknown", it means that the relationship cannot be decided based solely on the categories of the source and destination AS.

	Total	C2P or P2C		S2S		P2P	
Estimated	45771	37470	81.9%	0	0%	8301	18.1%
Inferred	74536	68216	91.5%	888	1.2%	5432	7.3%
Total	120307	105686	87.8%	888	0.7%	13733	11.5%

Table 4.7: Inferred and Estimated AS Relationship

With those unknown relationships, we will use the second method: the degree based method. We set up a threshold R and compare the degree of the two nodes. For a edge with two ASes u and v,

if $\frac{deg(u)}{deg(v)} > R$, then $R(u, v) = P2C$

if $\frac{deg(u)}{deg(v)} < 1/R$, then $R(u, v) = C2P$

else $R(A, B) = P2P$

where $deg(u)$ is the degree of AS u.

For example, If AS u is classified as T1 (large ISP) and AS v is classified as a university (Edu), the relationship between the u and v is provider-to-customers (P2C). However, if u is classified as T1 and v is T2, we can not decide the relationship between these two ASes based on the categories. We will rely on the degrees of u and v to decide. Table 4.7 shows the estimation results.

4.5.2.3 Completion of AS Topology

In AS topology, reachability is determined not only by the connectivity of the ASes, but also by the relationship between the ASes. As in the Internet, it is extremely difficult to find all edges. Besides, relationships between ASes are not 100% accurate. This will result in some ASes being unreachable from other ASes. For simulation purposes, the topology must be connected between any two ASes of the topology. Therefore, we need to add a minimum number of AS edges to the AS topology in order to make it connected. Our algorithm is as follows:

We divide the ASes into two categories: transit ASes and stub ASes (including multi-homed ASes). Transit ASes are those ASes with siblings and customers and

Core ASes	31
Transit ASes	4696
Stub ASes	24456
Added edges from core AS to core AS	0
Added edges from transit AS to core AS	3
Added edges from stub AS to transit AS	643

Table 4.8: AS topology completion results

stub ASes are ASes without siblings or customers. First, we want to make sure that subgraph of transit ASes is connected. Secondly, we want to make sure that each of the stub ASes has at least one provider to one transit AS. The details of the algorithm of making the AS topology graph connected is as follows:

1. Build internet core AS set (only around 10, tier-1 ISPs, degree is larger than 500). If any one of them not reachable to any other ASes, a P2P edge is added between these two ASes. Thus, the subgraph on core ASes is connected.

2. Transit ASes. If a transit AS u is not reachable from any other transit AS v, a C2P edge is build from u to one of the core ASes. From Theorem 4.3, one can infer that by adding the C2P edge from u to the core ASes, u is reachable from core ASes and all the ASes that are reachable from the core ASes. After this step is done, the transit AS set is connected.

3. For the stub ASes, if a stub AS u does not have any provider, a transit AS v will be randomly selected from the transit AS set and a new edge (u, v) will be added. After this step is done, the whole AS graph is connected.

Table 4.8 shows the results of completing AS topologies. In total, there are only 3 transit edges added to the topology. It is unlikely that adding those edges will significantly change the characteristics of the AS topology.

	iPlane	**Dimes**	**Combined**
Routers	105019	587303	720308
Edges	528887	2973228	3311992
Nodes	195840	1384673	1443817
PoP	102366	88138	176183
PoP Edges	333673	308886	560905

Table 4.9: Router-level topology inferred from iPlane and Dimes data

4.5.3 Inferring the Router-level Topology

Our primary data source for building the router-level topology is traceroute. Traceroute allows us to identify the forwarding path from the source IP address to the destination IP addresses. Traceroute also provides us with hop-by-hop round-trip times. For a router path with k hops $P = (n_1 \cdots n_k)$, the intermediate nodes in the path are router interfaces. The IP addresses of routers are extracted from the router level path except the first hop and the last hop. The router edge delay is estimated as the difference of the round-trip time from the sender to the source and destination router of the edge. However, for security and privacy reasons, many Internet sites have blocked traceroute requests. As a result, a "*" may appear in the router path which represents a router with unknown IP address and name. In this cases, we only use the hop without "*" in between. In Internet measurement, there is a lot of measurement error which might result in computing negative delays for some router edges. In this case, we simply record the delay as zero. From tracetroute data, we can infer a list of nodes, edges and routers. Dimes also aggregate the traceroute data into a list of nodes, edges and routers. We can combine data from iPlane and Dimes together. For the delay of edges, we just average the delay of all repeated edges.

In the next step, we use the clustering algirthm in iPlane to cluster router interfaces into PoPs and mapping router interface edges into PoP edges. First of all, we find the geographical location of all router interfaces. We use DNS name to infer the physical location. iPlane provides a mapping from IP address to PoP. We can use this IP to PoP map to map IP edges to PoP edges.

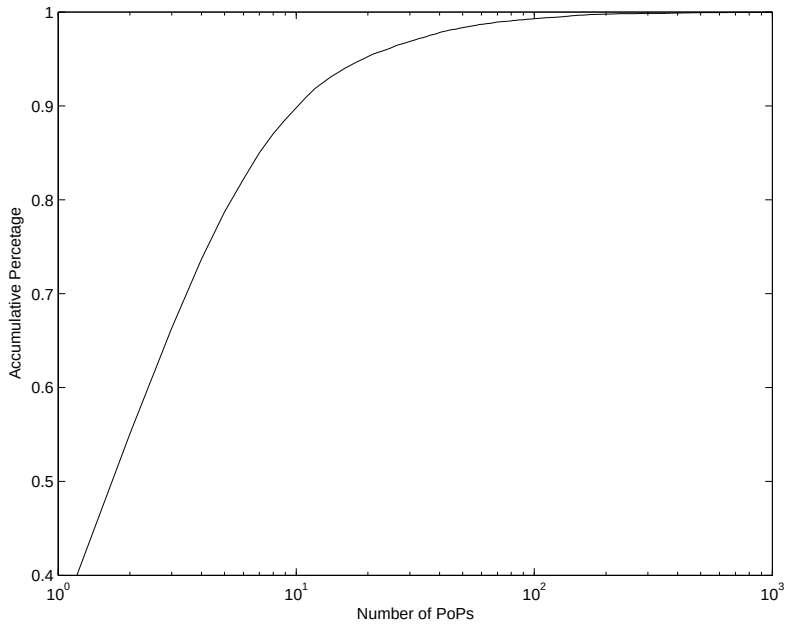

Figure 4.3: PoP number distribution in ASes

The delay of the router-level edges affects the Internet network distance. Due to measurement error, there are some negative edge distancies. In these cases, we simply regard it as zero. In other cases, some edge latencies are very large. We set 2000ms as the threshold and any edge with latency large than 2000ms is regarded as invalid edges. Table 4.10 and Figure 4.4 show the distribution of latencies of router edges. It shows that in traceroute data from iPlane, there are a large part of router edges with latency of zero. These is due to the closeness of routers in those edges (edges between an access router and a core router). For those edges with non-zero latency, the latency follows an exponential distribution. Figure 4.4 shows the distribution of non-zero edge latencies. It also shows that an exponential curve fits the distribution very well. The latency of PoP edges is estimated as the average edge distance of all those IP edges mapped to the PoP edge. We fit it with an exponential function. Figure 4.4 shows that it fits the data very well.

0	**0 – 2000ms**
iPlane87.5%	12.5%
iPlane PoP30.2%	69.8%

Table 4.10: Latency of router edges and PoP edges in iPlane data

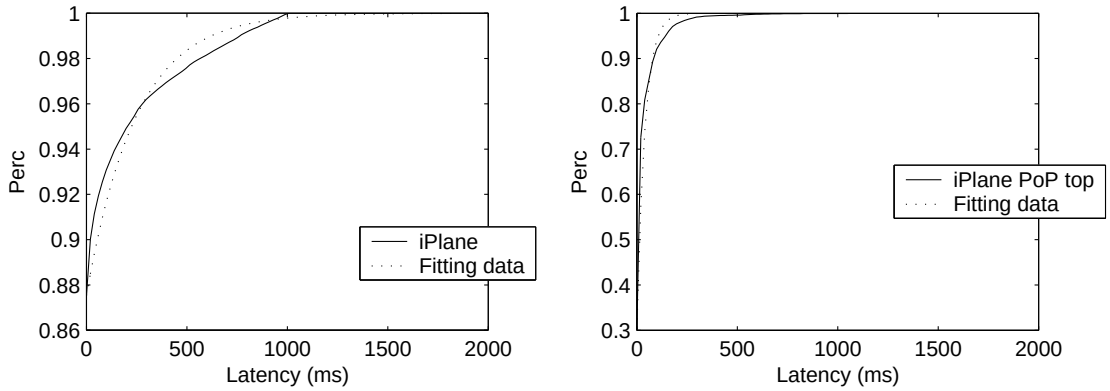

Figure 4.4: Latency distribution of iPlane

4.5.4 Completion of PoP-level Topology

With TraceRoute data, one cannot find all PoP level edges. With missing PoP edges, the PoP topology graph in some ASes might not be connected. Besides, some inter-AS edges might not have corresponding PoP edges. These will make the PoP level topology not connected. As Internet application simulation always requires a connected topology, we need to add minimum PoP edges to make the topology connected.

The algorithm for completing the PoP topology is as follows:

1. For each AS in the AS topology, find if there is a PoP in the PoP edge belong to the AS. If no, add one new PoP in the PoP topology belonging to the AS.

2. For every AS edge, if there is no edge between these two Ases in the PoP topology, randomly pick one PoP from each of the two Ases and add one PoP edge in the PoP topology.

3. For each AS, we check if the PoP graph is connected. The algorithm is as follows.

Added PoPs	0
Added inter-AS edges	3
Added intra-AS edges	643

Table 4.11: PoP topology completion results

(a) Compute the distance table for PoP topology. Suppose that there are N PoPs $p_i, (i = 1, \cdots, N)$. We call the connected sub graph vertex set $T = \{p_1\}$; the remaining set is $S = \{p_2, \cdots, p_N\}$.

(b) Get a node p_j in S, if the distance from p_1 to p_j is not infinity, remove p_j from S and add it to T. Otherwise, randomly pick a node p_i from T and add a PoP edge (p_i, p_j) to the topology. The algorithm of estimating the edge latency is discussed later.

(c) Repeat step 3b until the set S is empty.

One more detail is to determine the latency of the added edges. It approximately follows a exponential distribution $f(x) = ae^{-ax}$. We can estimate the parameter a with the inferred PoP edge latencies. The added edge latency is randomly generated with an exponential distribution. Table 4.11 shows the added PoPs and PoP edges.

4.5.5 Inferring Access Edges

An access edge is the IP level logical edge between an end-host and a gateway router. From traceroute data, we collect all the intermediate nodes from the traceroute paths as the set of router interfaces. We classify the rest of nodes that appear in the traceroute routing paths as end-host nodes. A edge is an access edge if only one of the nodes in the hop is a router. As it is usually the access router and latencies are similar across end-hosts within the same /24 prefix, we group the access edges into a list of /24 prefixes and a corresponding gateway router. Futhermore, with the mapping from router IP to PoP, we will be able to compute a mapping from a /24

prefix to a PoP.

4.6 Topology Generation

The topology generator is driven by three configuration files: an AS topology file which contains all the ASes and AS edges with their assigned relationships, a PoP toplogy file which contains the list of PoPs, their coresponding ASes and the delay of each PoP edge, and an access file which contains a list of /24 prefixes and PoPs (PoPs that connect to end-hosts) and the capacity of each PoP.

When users want to generate a topology with a particular size, the topology generator randomly generates a list of end-hosts and connects them to PoPs. The output of the topology generator is a file which contains the list of end-hosts, their corresponding access PoP and the delay to the access PoP. The AS topology file, PoP file and end host files makes up the complete definition files of a topology.

4.7 Calculation of Routing Path

The calculation of a routing path given the source and destination node consists of three steps: 1) find the access PoP and ASes of the source and the destination node, 2) with the source and destination AS, find the AS level path; 3) with the AS level path and source PoP and destination PoP, find the PoP path in each of the ASes.

4.7.1 AS-level path Inference

[56] provides an algorithm to compute the shortest valley-free path from a source AS to a destination AS. However, the computation cost is $N^2(N + E)$ where N is the number of ASes and E is the number of AS edges. In this section, we presented an algorithm to compute shortest valid path in AS graphs based on Dijkstra's algorithm with computation cost of $N^2 + E$. Dijkstra's algorithm is used to compute a best

routing path in a directed or undirected graph. However, it can not be applied directly to AS graphs since a connected path does not mean a valid path in an AS graphs.

We infer the shortest AS-level routing paths between pairs of source and destination ASes based on the valley-free property. A valid AS path can be of one of two possible patterns: a) an uphill path followed by a downhill path; b) an uphill path followed by a P2P edge and a downhill path. From Theorem 4.4, we know that for any shortest valley-free path, the uphill path is the shortest uphill path and the downhill path is the shortest downhill path.

Given a source node s, the new algorithm is as follows. The algorithm has four steps. Figure 4.1 shows the algorithm in pseudo code.

1. First, we determine all shortest uphill paths from s to all other ASes u. We do so by only considering the C2P edges in the AS graph. Each C2P is assigned a weight of 1 and Dijkstra's algorithm is used to compute the distance and predecessor table.

2. For each of the ASes u in the AS graph, if $\mathrm{d_u(s, u)}$ is not not infinity, we add a virtual P2C $(s,\ u)$ edge to the AS graph with weight equal to $\mathrm{d_u(s, u)}$.

3. For each of the ASes u in the AS graph of which $\mathrm{d_u(s, u)}$ is not not infinity, we search all peers of u. For every peer v of u, add a virtual P2C edge (s, v) to the AS graph with weight equal to $\mathrm{d_u(s, u)}$+1.

4. Use Dijkstra's algorithm to compute the shortest distance and path from s to any other node with all virtual P2C edges and P2C edges.

Listing 4.1: The Algorithm for Computing Shortest Valley-free Paths for an AS graph

```
function Dijkstra(Graph, source)
  // initialize distance table, peek table and peek neighbor table
  for each vertex v in Graph
    dist[v] = infinity
```

```
  pred[v]= undefined
  pathPattern[v] = undefined
  peekPoint = undefined
      peekNeighbor = undefined
end
// compute the shortest uphill path
dist[source] = 0
Q = copy(Graph)
while Q is not empty:
  u = extract_min(Q)
        for each provider p of u:
          alt = dist[u] + 1
          if alt < dist[p]
            dist[p] = alt
            pathPattern[p] = A
            peekPoint[p] = p
            peekNeighbor[p] = undefined
            pred[p] = u
    end
        end
      end
// compute the shortest uphill path plus one or zero peer edge
      for each vertex v in Graph
  if dist[v] < infinity
    for each peer q of v:
      alt = dist[v] + 1
          if alt < dist[q]
        dist[q] = alt
        pathPattern[q] = B
        peekPoint[q] = q
```

```
            peekNeighbor = v
          end
        end
      end
    end

    // compute the shortest valid (uphill path plus one or zero
    // peer edge plus downhill path
    Q = copy(Graph)
    while Q is not empty:
      u = extract_min(Q)
      for each customer c of u
        alt = dist[u] + 1
        if alt < dist[c]
          dist[c] = alt
          pathPattern[c] = pathPattern[u]
          peekPoint[c] = peekPoint[u]
        end
      end
    end
```

To compute the AS path, we will save the shortest uphill distance table and predecessor table in memory. By doing so, we build an uphill reachability matrix $D_u(G)$ and $P_u(G)$ where $\mathrm{d_u(i,j)}$ is the distance from i to j and $\mathrm{p_u(i,j)}$ is predecesorr of the shortest path from i to j. If the shortest uphill path from i to j is P , then $\overline{P}$ is the shortest downhill path from j to i. Therefore, the downhill shortest path can be computed with the uphill distance and predecessor matrix.

For the shortest valid path, we also save the distance of shortest valley-free path and peek table in memory. Give a source and destination pair of AS (s, d), peek is the peer edge in the shortest path if the shortest path is of type B or the last

AS in the uphill path if the path is of type B. Given the peek of a source destination pair (s, d), we can compute the uphill and downhill path accordingly. The overall path is the union of the uphil path, the peer edge (if the overall path is of type B), and the downhill path.

4.7.2 Revised AS Path Inference Algorithm

In order to find the shortest valid AS path, we need to store the uphill distance table (1 byte per pair of nodes), uphill path predecessor table (2 bytes per pair of nodes), valid path distance table (1 byte per pair of nodes) and valid path peek table (4 bytes per pair of nodes) in memory. This requires significant memory. For instance, if the total number of AS is 20000, the memory requirement is 20*20*(1+2+1+4) = 3.2G bytes.

In the Internet, the ASes can be classified as transit and stub ASes. A transit AS provides traffic transit between its neighbors. It either has providers, peers or siblings. A stub AS only has one provider and peers, no siblings. We divide the ASes into two groups: transit ASes and non-transit ASes (stub or multi-home ASes). For any AS level path, only the first and last AS can be non-transit ASes and intermediate ASes are all transit ASes. Therefore, the algorithm for finding the shortest path can be modified as:

1. to find the set of providers or peers $P(S)$ for the source AS S; if the source is a transit AS, then $P(S) = \{S\}$;

2. to find the set of providers or peers $P(S)$ for the destination AS S; if the destination is a transit AS, then $P(D) = \{D\}$;

3. to find the shortest transit path from any provider or peer in $P(S)$ to any provider or peer in $P(D)$. The final shortest path will be the shortest transit path augmented with the source and destination ASes.

AS	Stub AS	Transit AS	AS edges	PoP	PoP edges	End Prefix
27554	4119	20534	4119	20534	4119	20534

Table 4.12: Summary of AS nodes in the AS toplogy used in evaulation

4.7.3 Router-level Path Inference

With the AS level path, we still need to find the router-level path inside each AS. Within each AS, a routing path is calculated using the shortest distance algorithm such as Dijkstra's algorithm. If one AS has multiple PoP edges with the next AS in the AS path, the routing algorithm will choose the egress PoP with the shortest distance from the ingress PoP.

4.8 Evaluation

RealNet has been implemented in Java. In this section, some evaulation results will be presented. The evaulation has been focused on the scalability of routing path computation and the representativeness of the topologies generated through Realnet compared with those generated by other topology generators such as GT-ITM, Nem, IGen and BRITE, etc.

In the simulation, the AS topology was obtained by combining edges and ASes from Dimes, iPlane, RouteView and Caida. The algorithm presented in Secton 4.6.2 is used to estimate the relationships of those edges with missing relationship information and add missing edges to make the AS topology fully connected. PoP level topology from iPlane was used and the algorithm in Section is used to add PoPs and PoP edges to make the PoP topology fully connected. Table 4.12 summarizes the AS topology and PoP topology used in the evaulation. In the following simulation, an end-host list of 1000000 nodes is created.

4.8.1 Routing Path Computation

For RealNet, a topology parser was implemented which loads the configuration files (AS topology file, the PoP topology file and the end-host file) and computes a routing path for any given pair of source and destination nodes. In this section, we compare the memory requirement, routing path computation time of RealNet with that of other topology generators, such as GT-ITM, Waxman, and the AB model. The average degree of the topologies is 3. We use Dijkstra's algorightm to compute the distance table. As the memory requirement and compuation time are approximately the same for topologies with the same network size and average edge degree, we show only the result for topologies generated with Waxman's model.

To use routing path information of a topology in a simulation, people can either compute the distance table before simulation and save it in memory or compute network distance on-demand. The first method requires a large amount of memory and a long setup time (computing the whole routing table) with the benefit of zero overhead for computing network distance during simulation. The second method only needs to store the network topology in the memory and it does not compute the whole distance table. It comes with the price of overhead of computing network distance during simulation. This can significantly increase the simulation time.

Table 4.13 shows that for a topology with one million nodes, it takes 1629 hours (67 days) to compute the whole routing table and it takes 8000G (4000G if only distance table is stored) of memory. It is clearly not feasible for any moderate computers. Even using distributed simulation with 100 machines, it takes 16 hours to compute the routing table and 80G of memory to store routing table for each machine. In this case, memory is still a big issue. For the on-dmand computation of routing path for a topology with 1 million nodes, computing one single routing path takes 7 minutes. For message-level simulation of Internet computation, computation of routing path and network distance is a major task. For instance, for a node with 10000 simulation instances and each instance computes network distance of one pair

	Size	Routing table computation	Memory usage (MB)	On demand Routing path computation for one pair
Random model	1,000	1.66 sec	8	1.2 ms
	2,000	8.88 sec	32	3.2 ms
	3,000	25.0 sec	72	5.8 ms
	4,000	50.9 sec	128	8.9 ms
	5,000	93.0 sec	200	12.9 ms
	7,000	224.2 sec	392	23.9 ms
	10,000	586.5 sec	800	46.4 ms
	100,000	16.3 hours	80,000	4.6 s
	1,000,000	1629 hours	8000,000	460 s
RealNet	1,000,000	583.7 sec	1200	0.11ms

Table 4.13: Memory Usage and Computation Time of Routing Path

of node for every 1000 seconds (simulation time). The computation of a new routing table for each simulation second is 10000/1000*4600 = 46000s. This means that it takes takes 766 hours to simulate the behavior of an internet application for 1 minute.

With RealNet, we save the AS topology and PoP topology distance table in memory. In total it takes about 1.2GB of memory. Computation of the AS topology distance table and PoP distance table only takes 10 minutes. The computation of each routing path takes 0.11ms. This is affordable for one single machine. If we divide the memory requirement to 100 machines, each machine only needs 12 MB.

4.8.2 Representativeness

RealNet is intended to be used to generate topologies for large-scale message-level simulation of Internet applications. For message-level simulation, the network distance distribution affects the performance. In this section, we compute the distance distribution of generated topologies and compare them with distance distribution of Internet measurement data. We randomly choose 10000 pairs of source and destination hosts and calculate the distance between those pairs.

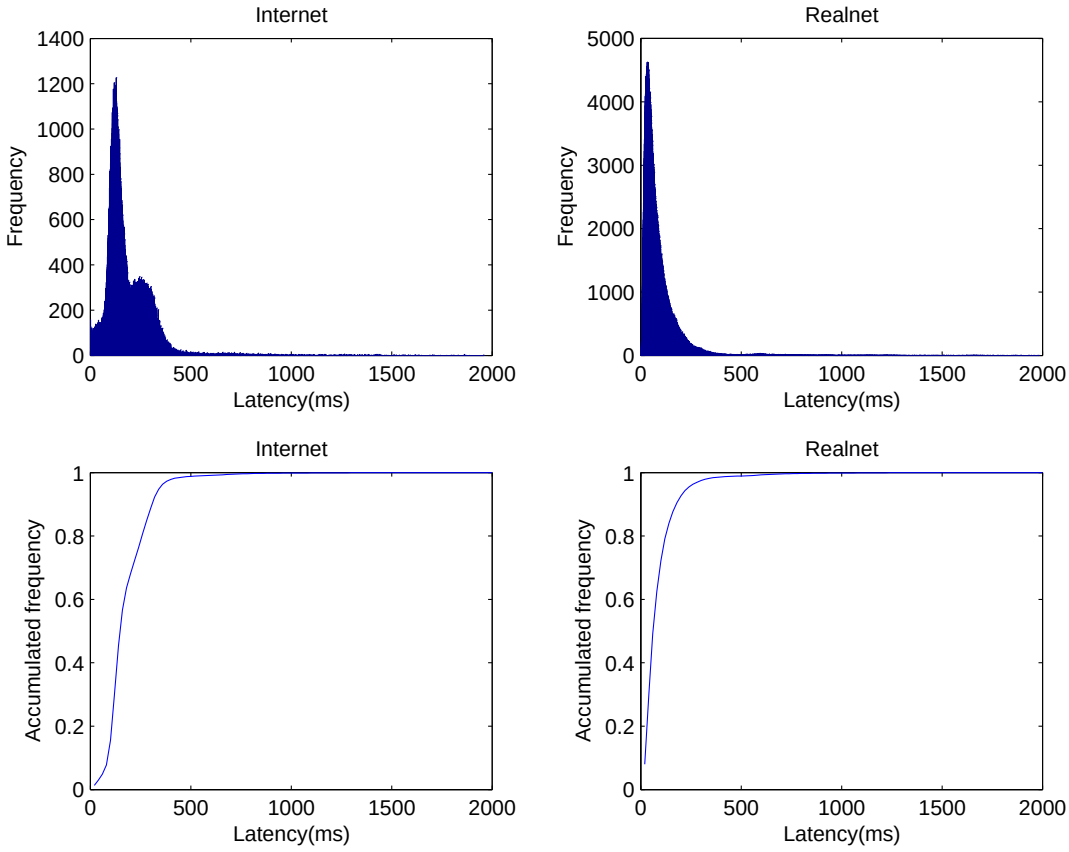

Figure 4.5: Network Distance Distribution of Internet and RealNet topology

Figure 4.5 shows the distance distribution of the Internet traceroute data. We randomly take 10000 distance data from iPlane raw traceroute data. It shows that the network distance of the Internet approximately follows an exponential distribution with an extra peak. Figure 4.5 also shows the data from RealNet. It also follows an exponential distribution. The average distance in RealNet is only half of that Internet traceroute data. This is due to the fact that the distance of PoP links in a RealNet topology is computed by averaging the routing level link distance. This filters out the noise or inaccurarcy of router link distances in actual Internet traceroute data.

Figures 4.6, 4.7 and 4.8 show the network distance distribution of the Waxman model, AB model and GT-ITM model. Waxman model has two parameters α, β. We vary the parameter α, β and network size. The network distance approximately

	Average	**Variation**	**Maxium**
Internet	188	187	1979
RealNet	97	134	2216

Table 4.14: Network Distance Distribution of Internet and RealNet topology

Topology	**T**	$\mathbf{N_t}$	**K**	$\mathbf{N_s}$	**Network size**
1	25	5	8	8	8040
2	5	25	8	8	8200
3	5	5	40	8	8200
4	5	5	8	40	8040

Table 4.15: Parameters of GT-ITM model

follows a normal distribution. The AB model has four parameters. We vary the parameter p, q and network size. Figure 4.7 shows that the parameter does not change the shape of the network distance distribution. Figure 4.8 shows the distance distribution of GT-ITM model. We vary the parameters T, N_t, K, N_s and obtain four topologies. Table 4.15 shows the value of the parameters in three topologies. From Figure 4.8, one can see that the distribution of GT-ITM topologies varied with the parameter of the topology.

4.9 Conclusion and Future Work

In this work, we propose a network topology generator which can generate representative topologies for large-scale Internet simulations. We infer the Internet structure from various available Internet measurement data. The infered AS level topology and clustered router level topology becomes the routing core and users can generate topologies with any desired size by adding end-hosts and connecting them to the PoPs. The calculation of the routing path is independent of the network size since it only depends on calculation of paths within the routing core. The topology generator is very efficient in computation, space and memory requirement. We compute the network distance distribution of a RealNet topology and that of topologies generated

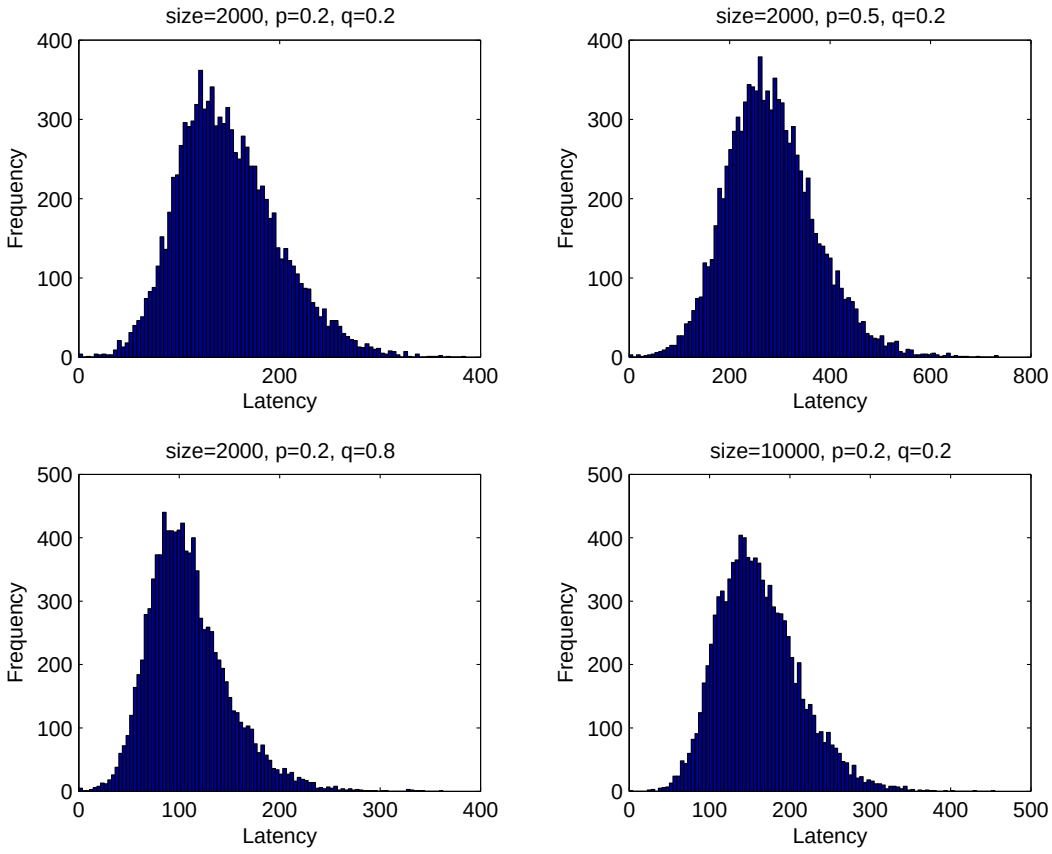

Figure 4.6: Network distance distribution of the BA model

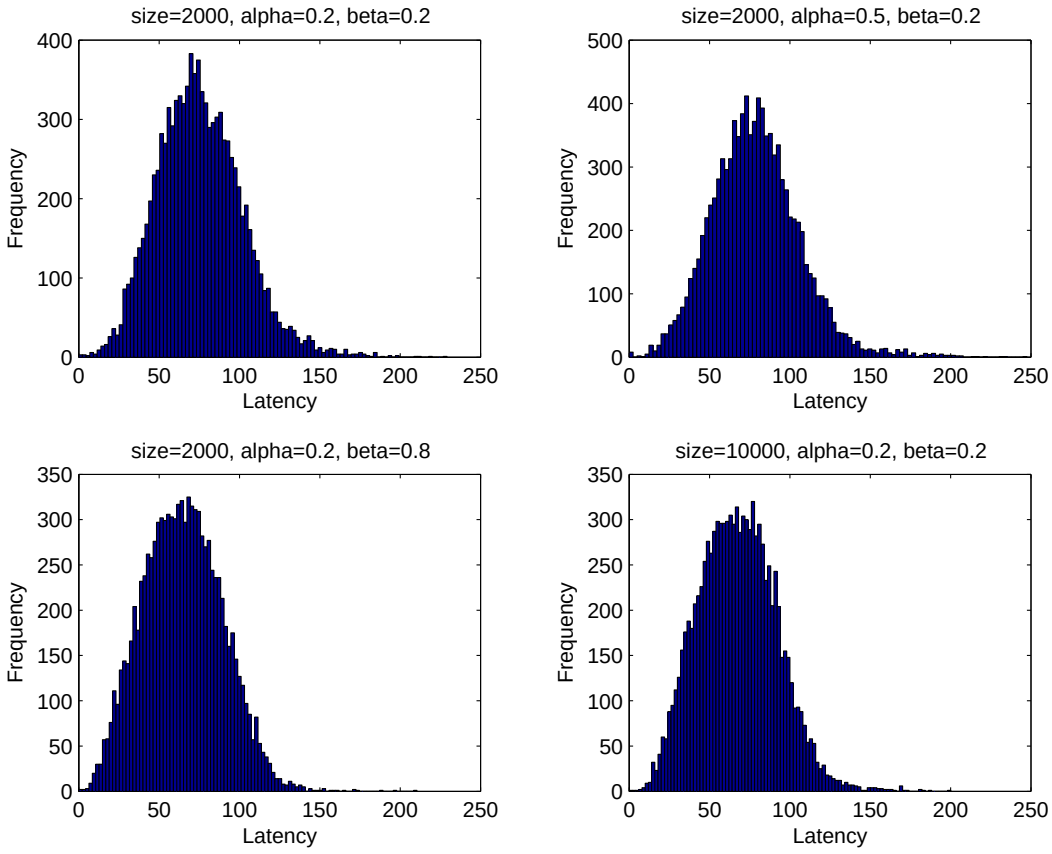

Figure 4.7: Network distance distribution of Waxman Model

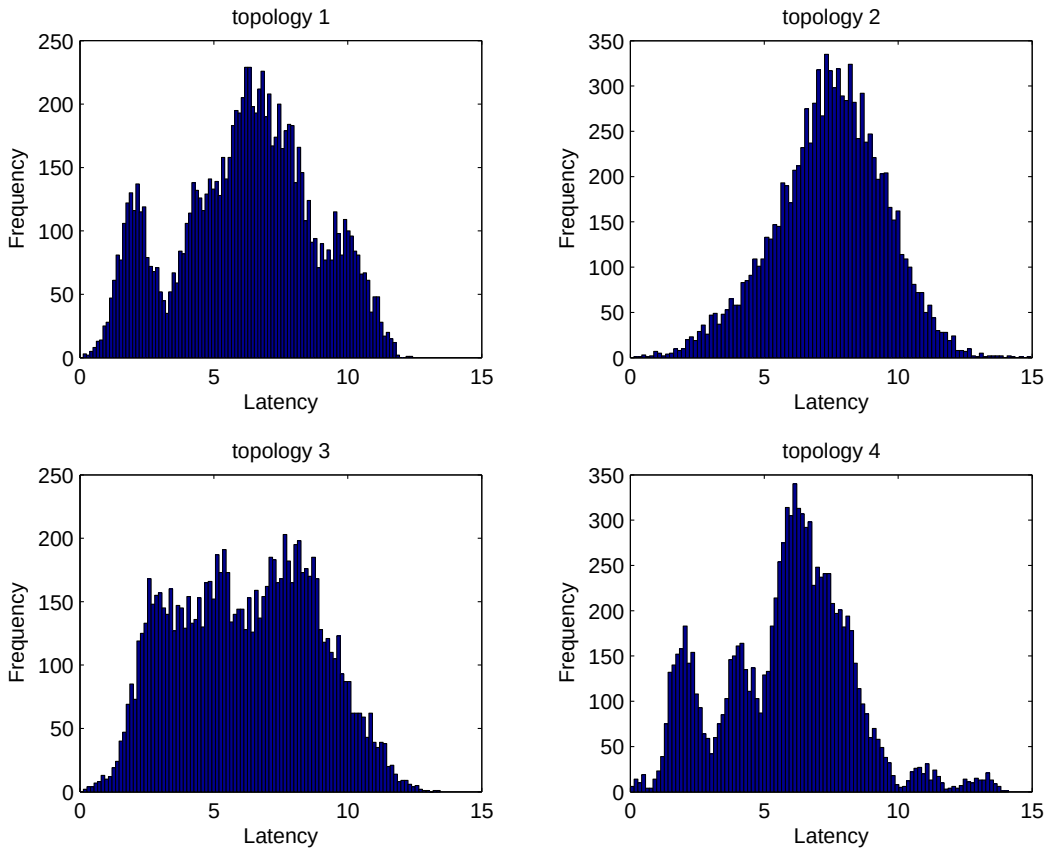

Figure 4.8: Network distance distribution of GT-ITM Model

by the GT-ITM, Waxman, and BA models. The results show that network distrance distrubtion of RealNet is very close to that of the Internet while the network distance distribution of topologies generated from GT-ITM, Waxman or BA model are very different from that of the Internet.

For the generated topology to be representative, the infered routing core has to reflect the real Internet structure. In the future, we plan to use more data sources and improve our AS relationship inference algorithm so as to find a more accurate AS level topology. There are a lot of research efforts trying to infer the Internet structure. We can constantly apply the new Internet measurement data to improve the accuracy of the routing core and make the resulting routing core available. Users can download and update their routing core before generating topologies.

Chapter 5

Internet Topology Based Identifier Assignment for Tree-based DHTs

5.1 Introduction

In recent years, there has been considerable research effort in the area of structured P2P systems using distributed hash tables (DHTs). Typical structured P2P systems include Chord[132], Tapestry[150], CAN[119], Pastry[125]. In structured P2P systems, every node and object is assigned a random ID from a numeric space using a consistent hashing algorithm. An object is usually mapped to the node with the closest numeric ID to that of the object. In structured P2P systems, nodes connect to each other based on their IDs to form a routing mesh and every node has a neighbor table of size O($\log N$). With the neighbor tables of the participating nodes, messages can be efficiently routed to the destination node through O($\log N$) hops.

The advantage of structured P2P systems is that their routing and locating methods are scalable. However, they also introduce some problems. Firstly, DHT-based systems assign IDs to nodes randomly. The nodes with close IDs might be widely separated in the Internet. Ignoring the positions of nodes causes high end-to-end delay and inconsistent routing performance. Secondly, the randomness in

assigning IDs to nodes and files destroys data locality. Data might be stored far from its users and outside the administrative domain that it belongs to.

P2P systems are usually overlay networks built on top of the Internet. In order to obtain an efficient routing, the overlay routing path must match the underlying Internet routing path as much as possible. Fundamentally, the Internet can be regarded as a decentralized network with a flat structure. However, in order to make routing efficient and scalable, a hierarchical structure has been introduced. The IP address is divided between the network address part and the computer address part. The Autonomous System (AS) was introduced to make the Internet appear as a three-tier structure. This hierarchical structure improves the routing scalability, efficiency and locality of the Internet.

In fact, tree-based DHTs (Pastry and Tapestry) can be considered as having a similar hierarchical routing architecture. To illustrate, if one wants to route a message to the node with ID 356, the message will first be sent to one of the nodes in the group of nodes with prefix 3**, then the group of 35* and finally 356. Therefore, if node IDs can be assigned according to the hierarchical structure of the Internet, the routing of the upper layers can be expected to match the underlying IP routing path. If the node ID bears the information of the network topology, it will become easy to provide data locality.

In this work, an Internet topology based ID assignment algorithm is proposed for tree-based DHTs, such as Pastry and Tapestry. Proximity Neighbor Selection (PNS) is used in combination with the topology based ID method to improve the routing efficiency for DHTs. With topology-based ID assignment, the hierarchical routing structure is introduced into DHTs without using super-nodes. Since the IDs bear information of the location of the nodes, it can provide explicit locality.

The rest of the chapter is organized as follows. Section 5.2 presents the details of our algorithm. Section 5.3 presents some simulation results. Section 5.4 concludes and presents some future work.

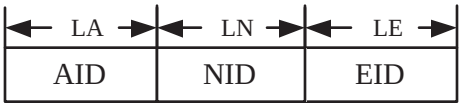

Figure 5.1: Node Id Structure

5.2 Topology based Node ID Assignment

In this section, the idea of Internet topology based node ID assignment is proposed for tree-based DHTs such as Tapestry and Pastry. Instead of generating the node IDs randomly using a consistent hashing function, IDs are assigned to the nodes according to their IP address, port number, IP network prefix and AS number. Let the length of node IDs be L and the base of ID be b. The node ID is divided into three parts. The first L_A digits are assigned according to the AS that the node resides in. The next L_N digits are assigned according to the node's network prefix. The last L_E digits are assigned according to the IP address and port number. These three parts are called the AID (AS ID), NID (Network ID) and EID (End node ID) respectively. The AID, NID and EID can be of a fixed length or a variable length. The algorithms to generate the AID, NID or EID are illustrated as follows.

5.2.1 AID Assignment Algorithm

On the Internet, an AS is the unit of router policy, either for a single network or a group of networks that is controlled by a common network administrator. ASes are usually stable and the connectivities among ASes do not change much with time. An inter-domain routing protocol such as BGP is used to determine the best path among ASes. For IP level routing, a packet is usually first routed to the destination AS, then its destination network prefix and finally its destination node.

Tree-based DHTs such as Pastry and Tapestry use prefix-based routing. The participating nodes forward a message by gradually resolving the prefix until getting to the node with the closest ID. In order to match the overlay routing path with the

underlying Internet routing path, it is desirable to assign AIDs so that the overlay routing mesh will reflect the connetivities among ASes and the first several hops of prefix routing will match the AS-level routing path.

In recent years, a lot of research has been done to infer the AS-level structure of the Internet. Ge et. al. [59] proposed a hierarchical structure to model the AS-level Internet topology. In the Internet, the majority of the relationships between ASes can be classified as the provider-customer relationship, and ASes in the Internet can be organized into a hierarchical structure with several levels. Each level is called a tier. An AS can be classified as either a tier-k provider or tier-k customer. It is assumed that the provider ASes only contain the core routers, not end nodes.

In this work, a hierarchical AIN assignment algorithm is proposed. The basic idea is to model the ASes in the Internet with a hierarchical structure and assign AIDs according to the position of the AS in the hierarchical structure. In addition, Proximity Neighbor Selection guarantees that the routing path among AS follows the underlying AS-level path.

Firstly, the AID is divided into K sections. Therefore the AID will be in the format of: $D^1 : \cdots : D^K$. D^i is called section-i of the AID and $D^1 : \cdots : D^m$ is called the level-m prefix of the AID. Every section of an AID can be represented by a sequence of digits with base b or the numeric value of the sequence. The AID of a tier m AS will have the format of $D^1 : \cdots : D^m : 0 : \cdots : 0$. The format is the same for both the provider and customer ASes.

In assigning IDs to the nodes, it is assumed that all the child nodes of a tier-m provider AS share the level-m prefix of their father node. To illustrate, if the AID of a tier-m provider is $D^1 : \cdots : D^m : 0 : \cdots : 0$, all its child ASes will have ID $D^1 : \cdots : D^m : D^{m+1} : 0 : \cdots : 0$. Section D^i of the AIDs is calculated iteratively from the top to the bottom. Figure 5.2 illustrates an example of the hierarchical structure of a simple network.

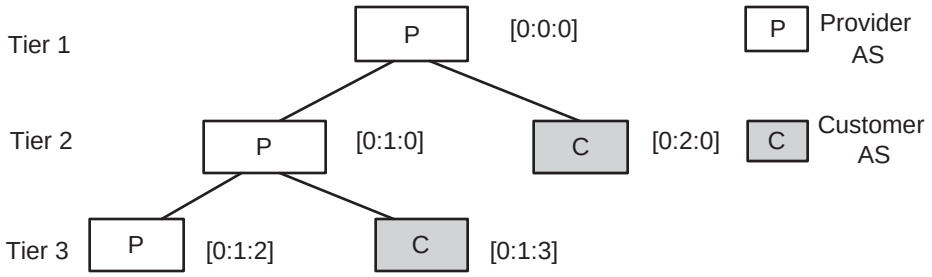

Figure 5.2: Id Assignment

5.2.2 Address-space Load Balancing

DHTs provide a mapping between keys (object IDs) and nodes. Every node is responsible for the set of keys that are numerically closest to itself. For simplicity, it is assumed that closest node ID of a key is the closest node ID that is larger than the key circularly. Random node IDs and object keys will guarantee that the keys are evenly distributed among nodes.

Our AID assignment method could cause imbalance in the key distribution for two reasons. 1) ASes in the Internet have different node densities and the number of keys assigned to each of the ASes is not proportional to its node density. 2) Some high level ID prefixes might be empty (not assigned to any AS). This will cause one of the nodes with the closest prefix to be responsible for the entire group of keys with the empty prefix.

We propose a fixed length method which provides fine grain ID space load balance. Every section of the AID is of fixed length. We try to carefully put the AID prefix in the ID space so that every prefix would be responsible for a portion of the key space proportional to its node density. With the hierarchical structure of ASes, the sections of AIDs are calculated iteratively from the top to the bottom. For the ASes that have the same father node, we sort them with respect to their densities in increasing order. Suppose there are M ASes and their node densities are $C_0 \leq C_1 \leq \cdots \leq C_{M-1}$. The node density of a provider AS is the sum of the node densities of its children ASes.

Let the digit length of the section-m of the AID be p. Then the number of possible p-length digit sequences will be b^p. The portion of sequences that the i^{th} ASes should be responsible for is:

$$B_i = C_i / \sum_{j=0}^{M-1} C_j$$

And the $m+1$ section of the AID of the i^{th} AS will be:

$$D_0^{m+1} = 0, D_i^{m+1} = [b^p * \sum_{j=0}^{i-1} B_j]$$

where [] means the closest integer.

To illustrate, assume that there are three ASes (A1, A2, A3) in a one tier Internet structure. The node densities of the ASes are 100, 600 and 900 respectively. If the base is 16 and the AID length is 1, the AID of A1, A2 and A3 will be 0, 6 and 15 respectively. Therefore, A1 will be responsible for all the keys with prefix 0, and A2 will be responsible for keys with prefix 1-6 and A3 will be responsible for keys with prefix 7-15. This warrants that the keys assigned to A1 and A2 are proportional to their node densities.

In order to eliminate the load imbalance inside each prefix, the mapping function has to be changed so that a key is mapped to the node with the closest AID, closest NID and closest EID. For closest AID, it means that every section of the AID has to be the closest so that the key space of the empty prefix will be evenly distributed among the nodes with the closest prefix. The mapping is deterministic since there is only one node with the closest AID, NID and EID to that of the desired key.

5.2.3 NID and EID Assignment Algorithm

In the Internet, different IP network prefixes have different node densities. In order to keep the address-space load balancing, NID is generated using the same method as that of AID in Section 3.2. The only difference is that within an AS, IP network prefixes have 1-tier hierarchical structure. The EID is generated using consistent hashing to preserve randomness.

5.2.4 Neighbor Table and Routing

With Internet topology based node ID assignment, the routing table will also reflect the hierarchical structure of the system. In the routing table, the last L_E columns contain nodes whose IDs match both the AID and the NID of the current node. Therefore, they are pointers to nodes in the same network. The entries of column L_A+1 to L_A+L_N contain nodes whose IDs match only the AID of the current node. They point to nodes in the same AS but in different networks. The first L_A columns contain nodes whose IDs do not match either the AID or the NID of the current node. They point to nodes in other ASes.

For simplicity, we use the same digit based incremental routing as that of Pastry. This is equivalent to gradually routing a message to the AS of the destination node, the network of the destination node and the destination node.

5.2.5 IP Prefix and AS Extraction

In order to calculate the AID for a node, a mechanism is needed to determine the AS number of a node given its IP address and to estimate the node density of every AS. We use the method similar to that mentioned in Sen et. al. [126]. Router level data is collected using BGP dump data such as Route-view [6]. With this data, all the network prefixes and masks and the corresponding AS can be found, thus building a mapping table. With an IP address, longest prefix matching is used to find the IP prefix, mask and AS number. As the mapping between an IP address and an AS is not changing with time, the process of creating the mapping table can be done offline. A mapping table could be provided to the user when they join the overlay network. With the mapping table, the node density of each of the ASes can also be estimated.

5.2.6 Discussion

The main concerns for topology based ID assignment for DHT are its side effects. One is the load imbalance. Some nodes might become the hotspot for both overlay routing and object storage. However, with topology based ID assignment, there is no super-peer. Nodes are regarded as the same during routing construction. Therefore, there is no routing hotspot. The simulation in the following section will also show that distribution of the in-coming link degrees is the same as that with random IDs. Besides, by taking the node density into consideration, the ID assignment method will alleviate the load imbalance with respect to object storage.

The second concern is the failure resilience. In the Internet, correlated failures are not unusual. In this method, the EIDs are assigned randomly. This keeps a certain level of randomness and thus improves the resilience to correlated failures. Another method to improve the failure resilience is to use multiple object IDs. An object can obtain more than one object ID using multiple hash functions. One of the object IDs is the primary ID and the rest are backups. In searching for an object, one can use the primary object ID to search for the object. If the object can not be found, then the backup object ID could be used. With multiple object IDs, one can improve the resilience to correlated failures.

5.3 Simulation Results

In this section, some simulation results are presented to demonstrate that Internet topology based ID assignment can effectively reduce the end-to-end delay of DHTs. The topology-based ID assignment method is applied to Pastry. In this simulation, the routing stretch is measured for 500 randomly selected pairs of nodes. Routing stretch is defined as the overlay routing delay divided by the underlying IP network delay. The average routing stretch is calculated as the metric of the average overlay routing performance. The standard deviation of routing stretch is also computed to

Table 5.1: The incoming-link degree versus network size

Size	RandID			TopID		
	Avg	Max	Min	Avg	Max	Min
100	89.7	103	83	89.9	103	82
200	91.1	103	83	90.8	105	83
500	93.1	102	84	93.2	115	82
1000	94.2	109	81	94.3	114	84
2000	95.6	113	86	93.8	116	85
5000	96.8	111	85	95.2	107	87

find out how the routing stretch varies among different paths in the overlay. The base b of IDs is set as 4.

The simulated network is generated using Georgia Tech's transit-stub network topology model (GT-ITM). As GT-ITM usually generates only routers, not nodes in the stub LANs, L_N (the length of NID) is set as zero. We will compare the routing performance of Pastry with random IDs and topology based ID in network topologies with increasing size.

5.3.1 Routing Performance

Figure 5.3 shows the average routing stretch versus network size. In the following figures, TopID represents Pastry with topology based ID assignment and RandID represents Pastry with randomly selected ID.

Figure 5.4 shows that using TopID decreases average routing stretch by 33% – 43%. The standard deviation of routing stretch reflects the consistency in the routing performance. A large standard deviation of routing stretch means that some routing paths will have large routing stretch. Figure 5.4 shows that the standard deviation of the routing stretch of the original Pastry is 2 – 4, while the routing stretch standard deviation to topology ID assignment is less than half of that of the original Pastry.

The incoming-link degree of a node D in an overlay network counts the number of nodes that have D in their routing tables. The larger incoming-link degree that a

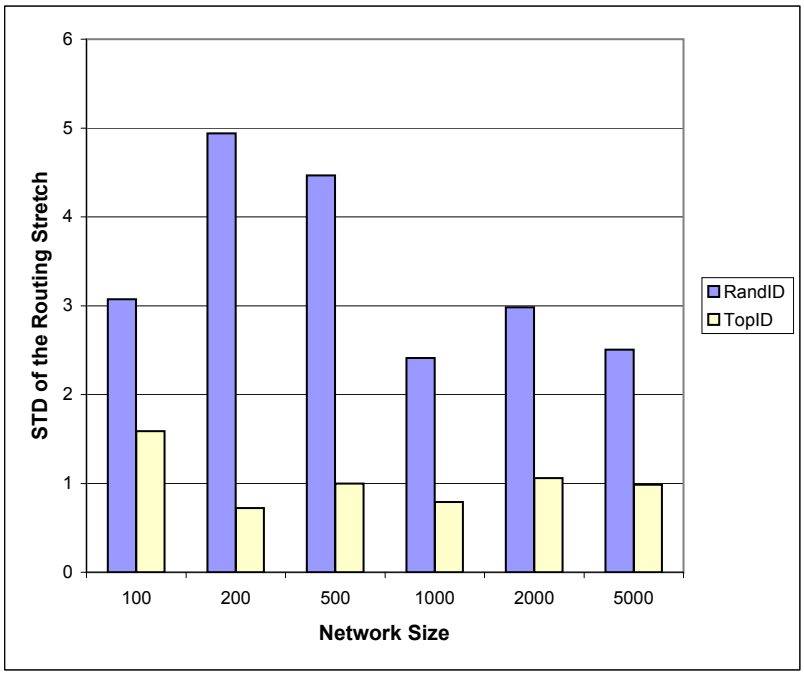

Figure 5.3: Average routing stretch versus network size

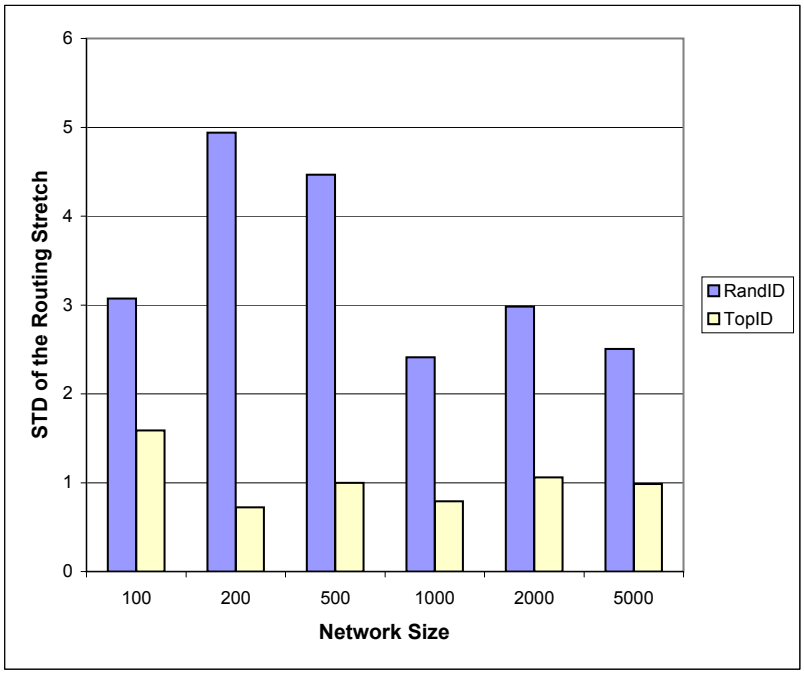

Figure 5.4: Routing stretch standard deviation versus network size

node has, the more routing traffic the node will have. Table 5.1 shows the average, minimum and maximum of in-coming link degrees of all the nodes in Pastry. With topology based ID assignment, the average, the minimum and maximum of the in-coming link degrees are almost the same as that with random IDs. The variance of in-coming link degrees is small compared to the average in-coming link degrees. This shows that topology based ID assignment does not introduce routing hotspots.

5.4 Conclusions and Future Work

In this work, a topology based ID assignment algorithm is proposed for tree-based DHTs such as Tapestry and Pastry. Simulation results show that the proposed method can reduce the routing stretch and routing stretch standard deviation of Pastry significantly in a variety of network topologies. This work can be applied to delay sensitive P2P applications such as video streaming. As the simulations in this work are only conducted to the scale of 5000 nodes, future work might be larger-scale simulations on networks with millions of nodes.

Chapter 6

Conclusions

In this section, we summarize the contributions, limitations and future work of this thesis.

6.1 Thesis Summary

Overall, the following contributions are made in this thesis:

- A topology based ID assignment algorithm is proposed for tree-based DHTs such as Tapestry and Pastry. Simulation results show that the proposed method can reduce the routing stretch and the routing stretch standard deviation of Pastry significantly in a variety of network topologies. This work can be applied to delay sensitive P2P applications such as video streaming.

- A message-level simulation framework is proposed for P2P systems. In designing this simulation work, an algorithm to use real time to synchronize the event processing of the processors is also proposed. This technique dynamically adapts to different simulation situations and helps reduce communication overhead of parallel event-driven simulations.

- The parallel simulator (P2PNet) was implemented in Java. Simulation results show that the simulator does not have large communication overhead compared

with a sequential event-driven simulator. When used in a cluster of computers, it provides high speedup in high work load situations without introducing any late events.

- For simulation of large-scale Internet application, a network topology generator is proposed which can generate scalable topologies for large-scale Internet simulations. The topology generated is based on the Internet structure inferred from various available Internet measurement data. The inferred AS level topology and clustered router level topology become the routing core and users can generate topologies with any desired size by adding end-hosts and connecting them to the PoPs. The calculation of the routing path is independent of the network size since it only depends on calculation of paths within the routing core.

- The Internet structure inferred from measurement data is usually inaccurate and incomplete. In the AS level, this will result in unmarked AS links (without relationship) or missing AS links. An algorithm is also provided to estimate the relationships of peering AS based on AS description information and neighboring degrees. As missing links can cause some ASes to be unreachable from other ASes, an algorithm is also proposed to add a minimum number of AS links to the inferred AS topology to make it fully connected. For the PoP topology inside each AS, an algorithm is also provided to add the minimum PoP links to make the PoP topology fully connected.

- In this thesis, we also studied the distance distribution of Realnet, Internet traceroute data, and some other topology generators such as GT-ITM. The results shows that the distance distribution of Realnet is very close to that of the real Internet data.

6.2 Future Work

There are some research areas that are suggested by our research:

6.2.1 Large-Scale Simulation Study of P2P Applications

Currently, there are more and more applications based on P2P technology. However, there are very few studies of large-scale (to millions of nodes) simulation studies of those systems due to lack of simulation tools and computational facilities. As scalability and performance are critical to those systems, we believe that large-scale simulation are very important for the understanding of those systems at real Internet scales.

6.2.2 Internet Topology Generator

Internet topology inference, modeling and topology generation are still a very active research areas. Understanding of the characteristics of the Internet topology is vital to the development of Internet applications. In the Internet topology inference research area, more research is still needed to get an accurate map of the Internet, both at router level and the AS level. From traceroute data and Internet data alone research can not infer accurate and detailed map of the Internet, new infrastructure and data collection facilities are needed to improve the accuracy of Internet topology inference. So far, most of the router level topology inference algorithms developed only infer link latency. Inference of other meta-data, such as bandwidth and delay remains unsolved issue.

Currently, there are many different types of topology generators. Future research is needed for an extensive comparative study of different topology generators. For comparison of topology generators, the most important factor is to find representative graph metrics. This is still an open research problem.

□

Bibliography

[1] J-Sim. http://www.j-sim.org.

[2] The network simulator - ns-2. http://isi.edu/nsnam/ns/.

[3] OMNet+. http://www.omnetpp.org.

[4] OPNET. http://www.opnet.com.

[5] P2PSim, a simulator for peer-to-peer protocols. http://pdos.csail.mit.edu/p2psim/.

[6] RouteView. http://www.routeviews.org.

[7] Routing information service. http://www.ripe.net/ris.

[8] Skitter. http://www.caida.org/tools/measurement/skitter/.

[9] SSFNet. http://www.ssfnet.org.

[10] Westgrid. www.westgrid.ca.

[11] Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling: Or, power-law degree distributions in regular graphs. In *Proceedings Of The Thirty-Seventh Annual Acm Symposium On Theory Of Computing*, pages 694–703, 2005.

[12] William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000.

[13] Reka Albert and Albert-Laszlo Barabas. Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85(24):5234–5237, 2000.

[14] David Alderson. Toward an optimization-driven framework for designing and generating realistic internet topologies. In *Proceedings of ACM HotNets-I*, 2002.

[15] David Alderson, Lun Li, Walter Willinger, and John C. Doyle. Understanding internet topology: Principles, models, and validation. *IEEE/ACM Transactions on Networking*, 13(6):1205–1218, Dec. 2005.

[16] Kostas G. Anagnostakis, Michael B. Greenwald, and Raphael S. Ryger. On the sensitivity of network simulation to topology. In *Proceedings of the Tenth IEEE/ACM Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 117–126, 2002.

[17] David G. Andersen, Nick Feamster, Steve Bauer, and Hari Balakrishnan. Topology inference from BGP routing dynamics. In *Proceeding of 2nd ACM SIGCOMM Internet Measurement Workshop*, Nov. 2002.

[18] Albert-Laszlo Barabas and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[19] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[20] Yi bing Lin, Paul A. Fishwick, and Senior Member. Asynchronous parallel discrete event simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 26, 1996.

[21] Charles Blake and Rodrigo Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. In *Proceedings of Ninth Workshop on Hot Topics in Operating Systems (HotOS-IX)*, 2003.

[22] Randal Everitt Bryant. Simulation of packet communication architecture computer systems. Technical Report TR-188, Massachusetts Institute of Technology, Cambridge, MA, USA, 1977.

[23] Tian Bu and Don Towsley. On distinguishing between internet power law topology generators. In *Proceedings of IEEE INFOCOM*, 2002.

[24] K. Calvert, J. Eagan, S. Merugu, A. Namjoshi, J. Stasko, and E. Zegura. Extending and enhancing GT-ITM. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, 2003.

[25] Kenneth L. Calvert, Matthew B. Doar, Ascom Nexion, Ellen W. Zegura, Georgia Tech, and Georgia Tech. Modeling internet topology. *IEEE Communications Magazine*, 35:160–163, 1997.

[26] Miguel Castro, Manuel Costa, and Antony Rowstron. Performance and dependability of structured peer-to-peer overlays. In *Proceedings of the 2004 International Conference on Dependable Systems andNetworks(DSN'04)*, 2004.

[27] K. Mani Chandy and Jayadev Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*, SE-5:440–452, 1979.

[28] H. Chang, S. Jamin, and W. Willinger. To peer or not to peer: Modeling the evolution of the internet's AS-level topology. In *Proceedings of 25th IEEE International Conference on Computer Communications*, 2006.

[29] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger. Towards capturing representative AS-level internet topologies. *Computer Networks*, 44:737–755, 2002.

[30] Hyunseok Chang, Sugih Jamin, and Walter Willinger. Internet connectivity at the AS-level: An optimization-driven modeling approach. In *Proceedings Of The Acm Sigcomm Workshop On Models, Methods And Tools For Reproducible Network Research*, 2003.

[31] Qian Chen, Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger. The origin of power laws in internet topologies revisited. In *Proceedings of IEEE INFOCOM*, 2002.

[32] Lechang Cheng, Norman Hutchinson, and Mabo R. Ito. Internet topology based identifier assignment for tree-based DHTs. In *Proceedings of New Technologies, Mobility and Security*, 2007.

[33] Lechang Cheng, Norman Hutchinson, and Mabo R. Ito. P2PNet: A simulation architecture for large-scale P2P systems. In *Proceedings of New Technologies, Mobility and Security*, 2007.

[34] Lechang Cheng, Norman C. Hutchinson, and Mabo R. Ito. RealNet: A topology generator based on real internet topology. In *Proceedings of 22nd International Conference on Advanced Information Networking and Applications - Workshops*, 2008.

[35] Byung-Gon Chun, Ben Y. Zhao, and John D. Kubiatowicz. Impact of neighbor selection on performance and resilience of structured P2P networks. In *Proceedings of IPTPS*, 2005.

[36] Rami Cohen and Danny Raz. The internet dark matter - on the missing links in the AS connectivity map. In *Proceedings of 25th IEEE International Conference on Computer Communications*, 2006.

[37] James H. Cowie, David M. Nicol, and Andy T. Ogielski. Modeling the global internet. *Computing in Science and Engg.*, 1(1):42–50, 1999.

[38] Jim Cowie and Hongbo Liu. Towards realistic million-node internet simulations. In *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications*, 1999.

[39] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and IonStoica. Towards a common API for structured P2P overlays. In *Proceedings of IPTPS*, 2003.

[40] Luca DallAstaa, Ignacio Alvarez-Hamelina, Alain Barrata, Alexei Vzquezb, and Alessandro Vespignani. Exploring networks with traceroute-like probes: Theory and simulations. *Theoretical Computer Science*, 355(1):6–24, 2006.

[41] Vasilios Darlagiannis, Andreas Mauthe, and Ralf Steinmetz. Overlay design mechanisms for heterogeneous, large scale, dynamic P2P systems. *Journal of Network and Systems Management, Special Issue on DistributedManagement*, 12(3):371–395, 2004.

[42] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *Proceedings of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, 2003.

[43] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, kc claffy, and George Riley. AS relationships: Inference and validation. *Acm Sigcomm Computer Communication Review*, 37:2007, 2006.

[44] Xenofontas Dimitropoulos, Dmitri Krioukov, George Riley, and Kc Claffy. Revealing the autonomous system taxonomy: The machine learning approach. In *Proceedings of Passive and Active Measurement (PAM) Workshop*, 2006.

[45] Xenofontas Dimitropoulos and George Riley. Modeling autonomous-system relationships. In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation*, 2006.

[46] Matthew B. Doar and Ascom Nexion. A better model for generating test networks. pages 86–93, 1996.

[47] B. Donnet and T. Friedman. Internet topology discovery: a survey. *IEEE Communications Surveys and Tutorials*, 9(4):2–15, December 2007.

[48] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of SIGCOMM*, pages 251–262, 1999.

[49] Steve L. Ferenci, Kalyan S. Perumalla, and Richard M. Fujimoto. An approach for federating parallel simulators. In *Proceedings of the fourteenth workshop on Parallel and distributed simulation*, pages 63–70, Washington, DC, USA, 2000. IEEE Computer Society.

[50] Ronaldo A. Ferreira, Suresh Jagannathan, and Ananth Grama. Plethora: A locality enhancing peer-to-peer network. In *Proceedings of Conference on Parallel and Distributed Systems (CPDS)*, 2004.

[51] A. Fisk. Gnutella dynamic query protocol v0.1, 2003.

[52] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. *IEEE/ACM Transaction on Networking*, 9(4):392–403, 2001.

[53] Richard Fujimoto, Kalyan Perumalla, Alfred Park, Hao Wu, Mostafa Ammar, and George Riley. Large-scale network simulation – how big? how fast? In *Proceedings of IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems*, 2003.

[54] Richard M. Fujimoto. *Parallel and Distributed Simulation Systems.* Wiley Interscience, 2000.

[55] RichardM. Fujimoto, Kalyan S. Perumalla, and George F. Riley. *Network Simulation (Synthesis Lectures on Communication Networks).* Morgan and Claypool Publishers, 2007.

[56] Lixin Gao. On inferring autonomous system relationships in the internet. In *Proceedings IEEE Global Internet Symposium*, 2000.

[57] L. Garces-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, and G.Urvoy-Keller. Hierarchial peer-to-peer systems. *Parallel Processing Letters*, 13(4):643–657, 2003.

[58] Pedro Garcia, Carles Pairot, Ruben Mondejar, Jordi Pujol, Helio Tejedor, , and Robert Rallo. Planetsim: A new overlay network simulation framework. In *Proceedings of the 19th IEEE International Conference on Automated Software Engineering*, 2004.

[59] Zihui Ge, Daniel R. Figueiredo, Sharad Jaiswal, and Lixin Gao. On the hierarchical structure of the logical internet graph. In *Proceedings SPIE ITCOM*, 2001.

[60] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for internet map discovery. In *Proceedings of IEEE INFOCOM*, March 2000.

[61] Mehmet H. Gunes and Kamil Sarac. Inferring subnets in router-level topology collection studies. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement.* ACM, 2007.

[62] Diwaker Gupta, Ken Yocum, Marvin McNett, Alex C. Snoeren, Amin Vahdat, and Geoffrey M. Voelker. To infinity and beyond: Time-warped network em-

ulation. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation*, 2006.

[63] Hamed Haddadi, Miguel Rio, Gianluca Iannaccone, Andrew Moore, and Richard Mortier. Network topologies: Inference, modeling, and generation. *IEEE Communications Surveys*, 2:48–69, 2008.

[64] Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, MarvinTheimer, , and Alec Wolman. SkipNet: A scalable overlay network with practical locality properties. In *Proceedings of Fourth USENIX Symposium on Internet Technologies and Systems(USITS '03)*, 2003.

[65] Yihua He, Georgos Siganos, Michalis Faloutsos, and Srikanth Krishnamurthy. A systematic framework for unearthing the missing links: Measurements and impact. In *Proceedings of the USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[66] Kirsten Hildrum, John D. Kubiatowicz, Satish Rao, and Ben Y. Zhao. Distributed object location in a dynamic network. *Theory of Computing Systems*, (37):405–440, March 2004.

[67] S. Jaiswal, A.L. Rosenberg, and D. Towsley. Comparing the structure of power-law graphs and the internet as graph. In *Proceedings of the 12th IEEE International Conference on Network Protocols*, Oct. 2004.

[68] David R. Jefferson. Virtual time. *ACM Transaction on Programming Languages and Systems*, 7(3):404–425, 1985.

[69] Márk Jelasity, Alberto Montresor, Gian Paolo Jesi, and Spyros Voulgaris. The Peersim simulator. `http://peersim.sf.net`.

[70] C. Jin, Q. Chen, and S. Jamin. Inet: Internet topology generator. Technical Report CSE-TR443 -00, Department of EECS, University of Michigan, 2000.

[71] Balachander Krishnamurthy, Jia Wang, and Yinglian Xie. A early measurements of a cluster-based architecture for P2P systems. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[72] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J h Cui, and A. G. Percus. Reducing large internet topologies for faster simulations. In *Proceedings of IFIP Networking 2005*, Waterloo, Ontario, Canada, May 2005.

[73] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, DennisGeels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon andWestly Weimer, Christopher Wells, and Ben Zhao. OceanStore: An architecture for global-scale persistent storage. In *Proceeding of ASPLOS*. ACM, Nov 2000.

[74] Anukool Lakhina, John W. Byers, Mark Crovella, and Peng Xie. Sampling biases in IP topology measurements. In *Proceedings of IEEE INFOCOM*, pages 332–341, 2003.

[75] Simon S. Lam and Huaiyu Liu. Failure recovery for structured P2P networks: Protocol design and performance evaluation. In *Proceedings of SIGMETRIS/Performance*, 2004.

[76] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978.

[77] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings Of The 12th Acm Sigkdd International Conference On Knowledge Discovery And Data Mining*, pages 631–636, New York, NY, USA, 2006. ACM.

[78] Jinyang Li, Jeremy Stribling, Thomer M. Gil, Robert Morris, and M. Frans Kaashoek. Comparing the performance of distributed hash tables under churn. In *Proceedings of the 3rd International Workshop on Peer-to-peer systems (IPTPS'04)*, Feb 2004.

[79] Jinyang Li, Jeremy Stribling, Robert Morris, and M. Frans Kaashoekand Thomer M. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In *Proceedings of the 24th IEEE Infocom*, Feb 2004.

[80] Jinyang Li, Jeremy Stribling, Robert Morris, and M. Frans Kaashoek. Bandwidth-efficient management of DHT routing tables. In *Proceedings of 2nd Symposium on Networked System Design and Implementation(NSDI'05)*, May 2005.

[81] Shiding Lin, Aimin Pan, Rui Guo, and Zheng Zhang. Simulating large-scale P2P systems with the WiDS toolkit. In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2005.

[82] Shiding Lin, Aimin Pan, Zheng Zhang, Rui Guo, and Zhenyu Guo. WiDS: An integrated toolkit for distributed system developmen. In *Proceedings of the 10th USENIX Workshop on Hot Topics in Operation System*, June 2005.

[83] J. Liu and D. M. Nicol. DaSSF 3.0 user's manual. http://www.cs.dartmouth.edu/research/DaSSF/Papers/dassf-manual.ps, 2001.

[84] Xin Liu and Andrew A. Chien. Realistic large-scale online network simulation. In *Proceedings of the ACM/IEEE SC2004 Conference on Supercomputing*, 2004.

[85] Margaret L. Loper and Richard M. Fujimoto. A case study in exploiting temporal uncertainty in parallel simulations. In *Proceedings of the 2004 International Conference on Parallel Processing*, pages 161–168, 2004.

[86] Harsha V. Madhyastha, Thomas Anderson, Arvind Krishnamurthy, Neil Spring, and Arun Venkataramani. A structural approach to latency prediction. In *Proceedings of Internet Measurement Conference*, 2006.

[87] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iPlane: An information plane for distributed services. In *Proceedings of OSDI*, 2006.

[88] Damien Magoni. nem:a software for network topology analysis and modeling. In *Proceedings of the 10th IEEE International Symposium on Modeling,Analysis,and Simulation of Computer and Telecommunications Systems*, 2002.

[89] Damien Magoni and Jean Jacques Pansiot. Analysis of the autonomous system network topology. *ACM SIGCOMM Computer Communication Review*, 31(3), July 2001.

[90] Damien Magoni and Jean-Jacques Pansiot. Internet topology modeler based on map sampling. In *Proceedings of the ISCC*, 2002.

[91] Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. Systematic topology analysis and generation using degree correlations. *SIGCOMM Computer Communication Review*, 36(4):135–146, 2006.

[92] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Xenofontas Dimitropoulos, kc claffy, and Amin Vahdat. The internet AS-level topology: Three data sources and one definitive metric. *ACM SIGCOMM Computer Communication Review*, 36:2006, 2005.

[93] Dalia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proceedings of the ACM PODC02*, 2002.

[94] Z. Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. On AS-level path inference. In *Proceedings Of The 2005 Acm Sigmetrics International Conference On Measurement And Modeling Of Computer Systems*, 2005.

[95] Peter Martini, Markus Rümekasten, and Jens Tölle. Tolerant synchronization for distributed simulations of interconnected computer networks. In *Proceedings Of The Eleventh Workshop On Parallel And Distributed Simulation*, pages 138–141, Washington, DC, USA, 1997. IEEE Computer Society.

[96] Petar Maymounkov and David Mazires. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the IPTPS*, 2002.

[97] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: Universal topology generation from a user's perspective. Technical Report BUCS-TR-2001-003, Computer Science Department, Boston University, 1 2001.

[98] Alberto Medina, Ibrahim Matta, and John Byers. On the origin of power laws in internet topologies. *ACM SIGCOMM Computer Communication Review*, 30(2)(2000-004):18–28, 20, 2000.

[99] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja1and Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, March 2002.

[100] Z. Morley, Mao David, Johnson Jennifer, Rexford Jia, and Wang Randy Katz. Scalable and accurate identification of AS-level forwarding paths. In *Proceedings of IEEE INFOCOM*, 2004.

[101] Zhuoqing Morley, Mao Jennifer, Rexford Jia, Wang Randy, and H. Katz. Towards an accurate AS-level traceroute tool. In *Proceedings of ACM SIGCOMM*, 2003.

[102] Wolfgang Mühlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Building an as-topology model that captures route diversity. *SIGCOMM Computer Communication Review*, 36(4):195–206, 2006.

[103] Wolfgang Mühlbauer, Steve Uhlig, Bingjie Fu, Mickael Meulle, and Olaf Maennel. In search for an appropriate granularity to model routing policies. *SIGCOMM Computer Communication Review*, 37(4):145–156, 2007.

[104] Alper Tugay Mýzrak, Yuchung Cheng, Vineet Kumar, and Stefan Savage. Structured superpeers: Leveraging heterogeneity to provide constant-timelookup. In *The Third IEEE Workshop on Internet Applications*, 2003.

[105] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *SIGCOMM Computer Communications Review*, 37(2):95–98, 2007.

[106] Hector Garcia-Molina Neil Daswani and Beverly Yang. Open problems in data-sharing peer-to-peer systems. In *Proceedings of ICDT*, 2003.

[107] David M. Nicol. Principles of conservative parallel simulation. In *Proceedings of Winter Simulation Conference*, 1996.

[108] Ricardo V. Oliveira, Beichuan Zhang, and Lixia Zhang. Observing the evolution of internet AS topology. *SIGCOMM Computer Communication Review*, 37(4):313–324, 2007.

[109] Christopher R. Palmer and J. Gregory Steffan. Generating network topologies that obey power laws. In *Proceedings of GLOBECOM*, November 2000.

[110] Alfred Park, Richard Fujimoto, and Kalyan Perumalla. Conservative synchronization of large-scale network simulations. In *Proceedings of ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation (PADS)*, 2004.

[111] Kalyan S. Perumalla. Parallel and distributed simulation: Traditional techniques and recent advances. In *Proceedings of the 2006 Winter Simulation Conference*, 2006.

[112] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

[113] Jian Qiu and Lixin Gao. AS path inference by exploiting known AS paths. In *Proceeding of IEEE GLOBECOM*, 2006.

[114] Bruno Quoitin. Towards more representative internet topologies. *Computer Networks*, 44:737–755, 2004.

[115] Bruno Quoitin. Topology generation based on network design heuristics. In *Proceedings Of The 2005 Acm Conference On Emerging Network Experiment And Technology*, 2005.

[116] Murali Krishna Ramanathan, Vana Kalogeraki, and Jim Pruyne. Finding good peers in peer-to-peer networks. Technical Report HPL-2001-2711, Software Technology Laboratory, HP Laboratories Palo Alto, October 2001.

[117] Dhananjai Madhava Rao and Philip A. Wilsey. Simulation of ultra-large communication networks. In *Proceedings of MASCOTS*, 1999.

[118] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM*, 6 2002.

[119] Sylvia Ratnasamy, Paul Francis, Mark Handley, and Richard Karp andScott Schenker. A scalable content-addressable network. In *Proceedings Of The 2001 Conference On Applications, Technologies, Architectures,And Protocols For Computer Communications*, 2001.

[120] Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Routing algorithms for DHTs: Some open questions. In *Proceedings of First International Workshop on Peer-to-Peer Systems*, 2002.

[121] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference*, June 2004.

[122] George F. Riley. The georgia tech network simulator. In *Proceedings Of The Acm Sigcomm Workshop On Models, Methods And Tools For Reproducible Network Research*, pages 5–12, New York, NY, USA, 2003. ACM.

[123] George F. Riley, Richard Fujimoto, and Mostafa H. Ammar. A generic framework for parallelization of network simulations. In *Proceedings of MASCOTS*, 1999.

[124] John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: search methods. Technical Report UNSW-. EE-P2P-1-1, University of New South Wales, 2004.

[125] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scalepeer-to-peer systems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[126] Subhabrata Sen, IEEE, and Jia Wang. Analyzing peer-to-peer traffic across large networks. *IEEE Transaction on Networking*, 12(2):219–232, April 2004.

[127] Yuval Shavitt and Eran Shir. DIMES: Let the internet measure itself, 2005.

[128] Kazuyuki Shudo, Yoshio Tanaka, , and Satoshi Sekiguchi. Overlay weaver: An overlay construction toolkit. *Computer Communications (Special Issue on Foundations of Peer-to-Peer Computing)*, 31:402–412, 2007.

[129] Georgos Siganos. Analyzing BGP policies: Methodology and tool. In *Proceedings IEEE INFOCOM*, 2004.

[130] Georgos Siganos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. Power-laws and the AS-level internet topology. *IEEE/ACM Transactions on Networking*, 11:514–524, 2003.

[131] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. In *Proceedings of ACM SIGCOMM*, 2002.

[132] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and HariBalakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[133] Jeremy Stribling, Isaac G. Councill, Jinyang Li, M. Frans Kaashoekand David R. Karger, Robert Morris, and Scott Shenker. OverCite: A cooperative digital research library. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, Feb 2005.

[134] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proceedings of IEEE INFOCOM*, Jun 2002.

[135] Mineo Takai, Yu An Chen, Xiang Zeng, and Jay Martin. Parsec: A parallel simulation environment for complex systems. *IEEE Computer*, 31:77–85, 1998.

[136] Hongsuda Tangmunarunkit, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger. Network topology generators: Degree-based vs. structural. In *Proceedings of ACM SIGCOMM*, pages 147–159, 2002.

[137] Ruixiong Tian, Yongqiang Xiong, Qian Zhang, Bo Li, and Ben Y. Zhaoand Xing Li. Hybrid overlay structure based on random walks. In *Proc. of IPTPS*, 2005.

[138] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large scale network emulator. In *Proceedings of 5th OSDI*, 2002.

[139] Helen J. Wang, Bhaskaran Raman, Chen N. Chuah, Rahul Biswas, Ramakrishna Gummadi, Barbara Hohlt, Xia Hong, Emre Kiciman, Zhuoqing Mao, Jimmy S. Shih, Lakmi Subramanian, Ben Y. Zhao, Anthony D. Joseph, and Randy H. Katz. ICEBERG: An internet-core network architecture for integrated communications. *IEEE Personal Communications*, 2000. Special Issue on IP-based Mobile Telecommunication Networks.

[140] B M Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, 1988.

[141] Jianhong Xia. On the evaluation of as relationship inferences. In *Proceedings of IEEE GLOBECOM*, pages 1373–1377, 2004.

[142] Kuai Xu, Zhenhai Duan, Zhi-Li Zhang, and Jaideep Chandrashekar. On properties of internet exchange points and their impact on as topology and relationship. In *Proceedings of Sigmetrics*, 2002.

[143] Voon yee Vee and Wen jing Hsu. Parallel discrete event simulation: A survey. Technical report, 1999.

[144] Ken Yocum, Ethan Eade, Julius Degesys, David Becker, Jeff Chase, and Amin Vahdat. Toward scaling network emulation using topology partitioning. In *Proceedings of MASCOTS 2003*, 2003.

[145] Ellen W. Zegura, Kenneth L. Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE Infocom*, 1996.

[146] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Proceedings of Workshop on Parallel and Distributed Simulation*, 1998.

[147] Beichuan Zhang, Raymond Liu, Daniel Massey, and Lixia Zhang. Collecting the internet AS-level topology. *SIGCOMM Computer Communication Review*, 35(1):53–61, 2005.

[148] Ben Y. Zhao, Yitao Duan, Ling Huang, Anthony Joseph, and John Kubiatowicz. Brocade: Landmark routing on overlay networks. In *Proc. of IPTPS*, pages 34–44, Mar 2002.

[149] Ben Y. Zhao, Ling Huang, John D. Kubiatowicz, and Anthony D. Joseph. Exploiting routing redundancy using a wide-area overlay. Technical Report CSD-02-1215, U. C. Berkeley, Nov 2002.

[150] Ben Y. Zhao, Ling Huang, Sean C. Rhea, Jeremy Stribling, AnthonyD Joseph, and John D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE J-SAC*, 22(1):41–53, January 2004.

[151] Ben Y. Zhao, Ling Huang, Jeremy Stribling, and Anthony D. Joseph andJohn D. Kubiatowicz. Exploiting routing redundancy via structured peer-to-peer overlays. In *Proc. of ICNP*, 2003.

[152] Ben Y. Zhao, Anthony D. Joseph, and John Kubiatowicz. Locality-aware mechanisms for large-scale networks. In *Proc. of International Workshop on Future Directions in Distributed Computing*, 2002.

[153] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report CSD-01-1141, U. C. Berkeley, Apr 2001.

[154] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang. An analysis of BGP multiple origin AS (moas) conflicts. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, pages 31–35, 2001.

[155] Shi Zhou and Raúl J. Mondragón. Accurately modeling the internet topology. *Phys. Rev. E*, 70(6):066108, Dec 2004.

[156] Shuheng Zhou, Gregory R. Ganger, and Peter Steenkiste. Balancing locality and randomness in DHTs. Technical report, School of Computer Science Carnegie Mellon University, November 2003.

[157] Shuheng Zhou, Gregory R. Ganger, and Peter Steenkiste. Location-based node IDs: Enabling explicit locality in DHTs. Technical report, School of Computer Science Carnegie Mellon University, September 2003.

[158] Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, and Randy H. Katzand John D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. of NOSSDAV*, pages 11–20. ACM, June 2001.