Distributed Solution Approaches for Large-scale Network Measurement Exploiting Local Information Exchange

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2014

Tien Hoang DINH

List of publication

Journal papers

1. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "A low-cost, distributed and conflict-aware measurement method for overlay network services utilizing local information exchange," *IEICE Transactions on Communications*, vol. E96-B, no. 2, pp. 459–469, February 2013.

Refereed Conference Papers

1. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "A distributed measurement method for reducing measurement conflict frequency in overlay networks," in *Proceedings of the 2011 International Communications Quality and Reliability Workshop (IEEE CQR 2011)*, pp. 1–6, May 2011.

2. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "A distributed and conflict-aware measurement method based on local information exchange in overlay networks," in *Proceedings of the 2012 Australasian Telecommunication Networks and Applications Conference (ATNAC 2012)*, pp. 1–6, November 2012.

3. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "A distributed measurement method exploiting path overlapping in large scale network systems," in *The 1st Workshop on Large Scale Network Measurements (31st NMRG meeting)*, Zurich, Switzerland, October 2013.

4. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "Monitoring available bandwidth in

overlay networks using local information exchange," in *Proceedings of the 2013 Australasian Telecommunication Networks and Applications Conference (ATNAC 2013)*, November 2013.

Non-Refereed Technical Papers

1. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "A distributed measurement method for reducing measurement conflict in overlay networks," *Technical Report of IEICE (CQ2010-57)*, vol. 110, no. 287, pp. 49–54, November 2010 (in Japanese).

2. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "A distributed and conflict-aware measurement method using local information exchange in overlay networks," *Technical Report of IEICE (IN2012-35)*, vol. 112, no. 134, pp. 13–18, July 2012.

3. Dinh Tien Hoang, Go Hasegawa and Masayuki Murata, "A distributed method for measuring available bandwidth in overlay networks exploiting path overlap," *Technical Report of IEICE (NS2013-71)*, vol. 113, no. 205, pp. 1–6, September 2013.

Preface

The traditional IP networks have shown many technical issues and have been vulnerable against network congestion, degraded performance, failures and malicious attacks. Furthermore, the conservative server-client communication model has many limitations, such as performance bottleneck of the server, huge management cost and lack of scalability. To overcome these problems, many techniques based on application-level communication between end systems have been developed. Some typical applications of these techniques include end system multicast, peer-to-peer (P2P) network applications, content distribution network (CDN) applications, global event notification, resilient routing, and denial-of-service attack prevention. These systems are constructed over a large number of end hosts or servers distributed at multiple network domains. These end hosts or servers cooperate with each other to dynamically select their communication paths at application layer, without affecting the IP-level routing mechanism. Furthermore, because these techniques do not require standardization processes, they can be deployed with ease and low cost.

To maintain and improve the performance of these network services, the end hosts need the information related to the quality of the paths connecting them, in terms of latency, loss rate, jitter, bandwidth, connectivity, etc. In general, these information can be obtained using measurements performed by the end nodes. Measurement methods can be divided into two groups: active and passive. Passive measurement methods can collect traffic information at end hosts for measurements without creating any traffic on the network, but require a long time for data collection. Active measurement methods exert measurement traffic to the network, but can obtain timely and accurate quality information. To conduct active measurements, the end nodes send probe packets to each other, and cooperate to obtain the measurement results from the characteristics of the probe packets.

To obtain accurate results, the measurements should be performed frequently. However, because the end-to-end paths often overlap with each other, frequent measurements over these paths can cause heavy traffic load at the overlapping segments. Existing measurement methods can not solve the trade-off between obtaining high measurement accuracy and reducing measurement traffic load. Another issue of these methods is that they are mostly centralized approaches, which use a master node to coordinate the measurements. In the case of large networks that contain largely distributed nodes, the master node becomes a potential performance bottleneck and a vulnerable point of the network.

This thesis presents a comprehensive solution for measuring the quality of the end-to-end paths to support large-scale distributed network systems. We focus on the methods for measuring the most important quality metrics including latency, loss rate, jitter, available bandwidth, and connectivity. We take a distributed approach to tackle the problem of scalability. We also exploit information exchange between end nodes to resolve the trade-off problem mentioned above. All of the proposed methods start with the phase of detecting the overlapping status, using information exchange of route information. Then, the algorithms for improving measurement efficiency are proposed based on the characteristics of the metrics.

We first introduce a method for measuring the metrics that the measurement result of a path can be accumulatively calculated from the measurement results of included segments of the path. These metrics, called additive metrics, include latency, loss rate and jitter, which are important for distributed systems, such as VPN and CDN, and streaming media. The basic idea is that the end nodes exchange measurement results of the overlapping segments between end-to-end paths, and use statistical processing to improve the measurement accuracy of these segments, thus consequently improve the measurement accuracy of the whole path. The measurement result of a segment can be obtained by sending probe traffics to the two end nodes or routers of the segment. Simulation results show that the relative error in the measurement results of our method can be decreased by half compared with the existing method when the total measurement overheads of both methods are equal.

In the second part of this thesis, we produce a method for measuring the end-to-end available bandwidth, which is important for applications that require the transmission of large data, such as

storage area networks and P2P networks. Because the measurement results of the overlapping segment of two end-to-end paths can not be obtained from the measurements of the end nodes, we can not apply the method for measuring the additive metrics mentioned above. We therefore aim at reducing measurement time and traffic of each measurement, thus can reduce the measurement conflict, and consequently enhance measurement accuracy. To achieve this goal, in our method, the end nodes share the measurement results of overlapping paths to configure parameter settings for available bandwidth measurements. Simulation results show that the relative errors in the measurement results of our method are approximately only 65% of those of the existing method. Furthermore, the measurement accuracy of our method remains better than the existing method when the total measurement traffic loads of both methods are equal.

We also propose a method for diagnosing failures, which can help the distributed network systems to rapidly detect and bypass the problematic network segments. A typical fault diagnose procedure contains two phases: the fault detection phase, whose purpose is to periodically check if there are some problems in the network, and the fault localization phase, whose purpose is to rapidly locate the components responsible for the detected problems. We propose two algorithms that not only can rapidly detect network failures and locate faulty components, but also can reduce the total measurement traffic and well balance the traffic between the links of the network. Similar to our previous researches, we also utilize information exchange of measurement results of overlapping paths to reduce the measurement traffic. In the fault detection phase, we set constrains of measurement traffic for each link of the network, and probe the paths that satisfy the constrains with maximum measurement efficiency. On the other hand, in the fault localization phase, we choose the paths for measurements so that the expectation value of suspected faulty components after probing the paths is minimum. Simulation results show that our method can detect failures much faster than existing method while keeping the measurement traffic well-distributed between the links.

Throughout the researches, we have confirmed that the measurement efficiency, in terms of enhancement of measurement accuracy and reduction of time requiring for fault diagnosis, can be improved by slightly shifting overhead from measurement to information exchange. The idea of reusing measurement results of overlapping segments is also utilized in other approaches of the

literature, such as network tomography. However, the basic difference is that, in the network tomography, the measurement results of each overlapping segment are basically obtained by one end node and are used to calculate measurement results of multiple paths traversing the segment, while in our approach, they are measured and shared by multiple end nodes. Therefore, our approach can avoid some biases causing by one measurement agent, and has larger chance to obtain accurate measurement results.

Acknowledgments

This thesis would be impossible without the assistance of many people. First of all, I would like to express my deepest gratitude to my superviser, Professor Masayuki Murata, for giving me the chance to pursue a new way in my career. His generous guidance, insightful comments and patient encouragement have been essential throughout my Ph.D.

I am also heartily grateful to the members of my thesis committee, Professor Teruo Higashino, Professor Toru Hasegawa and Professor Takashi Watanabe of Graduate School of Information Science and Technology, Osaka University, and Professor Morito Matsuoka of Cyber Media Center, Osaka University, for their multilateral reviews and perceptive comments.

I would like to especially express my deepest appreciation to Associate Professor Go Hasegawa of Cyber Media Center, Osaka University. Without his continuous advices and supports, I would not have accomplished the Ph.D. program.

I also acknowledge Professor Naoki Wakamiya, Associate Professor Shin'ichi Arakawa, Assistant Professor Yuichi Ohsita, Assistant Professor Yoshiaki Taniguchi, Assistant Professor Masafumi Hashimoto, Assistant Professor Daiichi Kominami of Osaka University and Dr. Kenji Leibnitz of National Institute of Information and Communications Technology for their valuable comments and suggestions on my study.

I would like to sincerely thank to Dr. Le Thanh Man Cao, who introduced me to the Advanced Network Architecture Research Laboratory, and continuously gave me useful advices and encouragements. I also express my appreciation to colleagues and secretaries of the Advanced Network Architecture Research Laboratory, Graduate School of Information Science and Technology, Osaka University, for their support.

Finally, I am indebted to my parents, my sister and my brother for their understanding, endless support and encouragement. Especially, I would like to give my thanks to my dearest wife Minh Trang for her sacrifice and endless patience. I also always feel happy and motivated whenever I am with my lovely daughter Ha Phuong.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 End-to-end path quality in distributed network systems

The Internet is a large network that connects a huge number of autonomous systems (AS), which are operated by different internet service providers (ISP). Because of the interest of policy enforcement, ISPs share little information about the network topology and traffic conditions of the AS's under their management. These information are exchanged using the Border Gateway Protocol (BGP-4) running at the border routers between the AS's. The simple and lightweight of the exchanged data brings the scalability for the Internet, which has extended to millions of networks nowadays. However, as the Internet expands, the failures at many network components also occur more frequently. The failures may be due to hardware problems, e.g., fiber cuts and malfunctioned router interfaces, software errors, e.g., router software bugs, or network misconfiguration, and malicious attacks. When a failure occurs, because of the lack of detail information about traffic conditions, the fault recovery mechanisms of BGP may take many minutes before routes converge to a stable form [1]. These delays disrupt most of the communications between Internet hosts. For example, a TCP connection will time out because the ACK packets do not come after a certain time.

To solve this problem, many calls for change have been voiced and solutions have been proposed. However, because any little change at the IP or lower layers can lead to the reconstruction of an extremely huge number of network hardwares, it takes a long time before an innovation at these

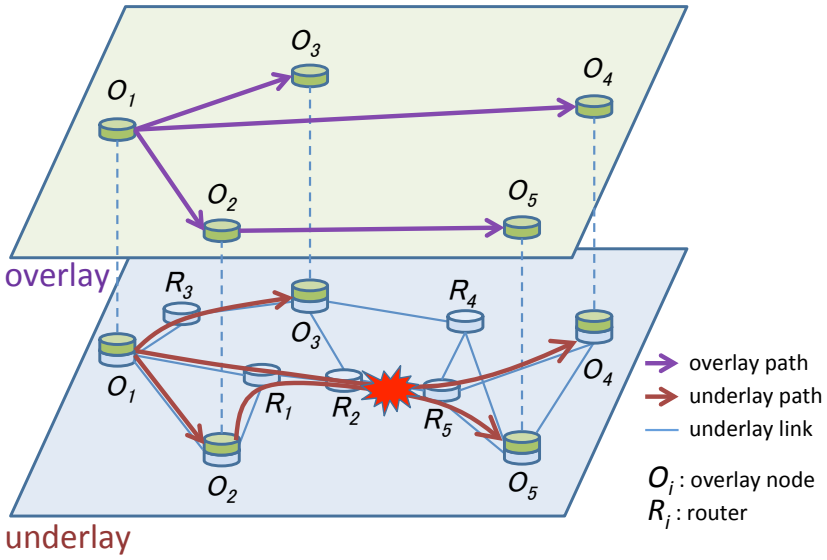


Figure 1.1: Example of an overlay network

layers can be accepted. Therefore, many solutions that not require changes at network hardwares have been proposed. These solutions include Peer-to-Peer (P2P) systems, content delivery network (CDN), resilient overlay routing,... Figure 1.1 shows an example of overlay network, which is defined as an application level logical network constructed over the underlay IP network.

In these systems, the end nodes can dynamically choose their communication paths to overcome various problems occurred in IP-layer network. To achieve this end, the end nodes must frequently monitor the quality of the IP level paths connecting them. There are many metrics that relate to the quality of a path, such as connectivity, latency, jitter, loss rate, available bandwidth, throughput, ... Depending on the characteristics of the service that the distributed network delivers, a varying set of quality metrics will be required. For example, in file sharing application based on P2P technique, the information of available bandwidth of the paths between a client and all the peers can help to download the wanted file with smallest time. A streaming media application may require strict quality related to latency, loss rate and jitter. Connectivity may be the most important metric, because the most concern of the end users is the ability to connect to the network.

1.2 End-to-end path quality measurements and limitations of current solutions

Most of the attention in the literature is to reduce the measurement traffic load, because monitoring all of the end-to-end paths will become too costly. For example, because all of the paths between overlay nodes in RON are extensively monitored, in order to not cause much effect to normal traffic, the number of overlay nodes in RON is limited to under fifty nodes [1]. Therefore, some solutions for reducing measurement traffic load has been proposed based on the characteristics of the metrics.

For the latency, some solutions based on network coordinate systems have been proposed [2,3]. These methods bases on the fact that latency between two end hosts is approximately proportional with their physical distance. Each end node is assigned a logical coordinates, and these data are used to calculate the latency between two arbitrary end nodes. Although this approach works well with latency, it can not be applied for measurements of loss rate. Network tomography [4–6], which infers link quality from end-to-end measurements, is an efficient approach for measurements of both latency and loss rate. In this approach, a group of end-to-end paths that covers all the links of the network are chosen for measurements, and from these measurement results, quality of links are inferred, and from these information, the measurement results of unprobed end-to-end paths can be calculated. This means this approach utilizes the overlaps between end-to-end paths to reduce the measurement traffic load. Although network tomography is considered as the state-of-the-art approach for measuring additive metrics, such as latency and loss rate, we have found that this approach may suffer from some bias between measurement agents (end nodes). That is, because the overlapping segments between end-to-end paths are basically measured by only one measurement agent and the results are used to estimate the measurement results of all paths sharing the segment, some small measurement error causing by one measurement agent can lead to the degradation in measurement accuracy of multiple paths.

BRoute [7] is one of the earliest methods aiming at reducing measurement traffic load in measurements of end-to-end available bandwidth. This method relies on two characteristics of overlay networks constructed over the Internet: (1) bottleneck links exist from both ends of the overlay path in roughly four hops or less, and (2) path overlappings often exist near both ends of the overlay

path. Therefore, the available bandwidth of a segment near both ends of each overlay path can be used to get the available bandwidth of the entire path, which greatly reduces the measurement traffic load. However, this method requires BGP routing information in advance to infer the AS-level paths between end hosts. Currently proposed solutions [8–10] rely on the observation that the measurement of available bandwidth can be approximately embedded to metric spaces, and thus it can be estimated using the concept of distance in metric space. Because embedding the measurement of available bandwidth to a metric space is only approximately justified by some real Internet datasets, there is a concern related to the measurement accuracy of these methods.

Although monitoring and diagnosing network faults at IP level is the responsibility of ISP, providers of distributed network services can also benefit by monitoring network faults by themselves. For example, if they detect that the problem is at the IP level, they can decide to require the ISP to repair the faulty components or compensate them, or simply bypass the problematic network segments. If they found that the problem occurs at some heavily overloaded servers in their network, they can decide to reroute some communication paths or replace those servers by the others at better locations. For the problem of diagnosing faulty components, network tomography is a straightforward solution. However, this approach is more costly than the approach that divides diagnose process into two phases: fault detection and fault localization. In the fault detection phase, a small number of probe packets are sent over the network so that all the link of interest can be covered. If some probes fails, that means some problems have occurs in the network, the fault localization will be initiated, and more probe packets will be sent to the problematic segments to exactly locate the faulty components. Most of existing methods focus only on minimizing the total monitoring overhead. However, we urge that the balance of overhead between network links is also of important. Another problem is that, none of existing methods deliver a temporal solution, that is the time table at which the monitoring tasks should be conducted.

Another issue in measurements of end-to-end path quality is the contention between measurements, which causes high traffic load and degrades measurement accuracy. Many researchers have pointed out that the end-to-end paths in distributed networks often overlap with each other [11]. Because the quality of end-to-end paths fluctuate with time, measurements should be conducted

with high frequency to obtain a good measurement accuracy. However, this will lead to the conflicts between concurrent measurements of overlapping paths. For example, in Fig. 1.1, the paths O_1O_4 and O_2O_5 overlap at the underlay level, i.e., they share links and routers on the path between routers R_1 and R_5. Therefore, the concurrent measurement tasks of paths O_1O_4 and O_2O_5 compete on the common links for network resources (e.g., processing power at routers and link bandwidth), causing high load on the common links and additional error in the measurement results. Some of existing methods [11–13] have addressed the problem, and try to schedule different timings for measurements of overlapping paths. Although the measurement conflicts can be avoided completely, it comes with the cost that measurement frequency is limited, thus the measurement accuracy is not high in overall.

1.3 Organization of the present thesis

The present thesis consists of three distributed measurement methods for three type of quality metrics of end-to-end paths in large-scale distributed network systems. Each of these methods is described together with the simulation results for evaluations and comparison with existing methods. In the remaining of the thesis, because we will not refer to other networks besides the distributed network systems, for simplicity, except explicitly mentioned, we simply refer to the term "the distributed network" as "the network".

1.3.1 Measurement method for end-to-end additive quality metrics

In Chapter 2, we introduce a distributed method for measuring the additive metrics, include latency, loss rate and jitter, etc. The method consists of three phases, in which we proposed some original algorithms based on information exchange between the end nodes of the network.

In the first phase, end nodes exchange route information to detect overlapping status between the paths. This phase is also the initial phase of the following two methods in Chapters 3 and 4. In this phase, each end node first detects the overlapping status between the paths starting from itself, by simply conducting the traceroute command to all of other end nodes. Then, based on the difference between the lengths of overlapping segments between these overlapping paths, the end

node infers the overlapping paths that have different source nodes. The end node then exchanges route information with these source nodes to confirm the overlapping status between the paths. Simulation results suggest that the method can detect over 90% of the actual overlapping paths.

In the second phase, based on the degree of the overlapping status, measurement frequency of each path are decided. The end nodes then decide measurement timings for each path, trying to reduce measurement conflicts between overlapping paths. That is, the overlapping paths with the same source node are measured sequentially, and the measurement timings of each path are randomly decided to reduce conflicts with the measurements of the overlapping paths that have different source nodes.

In the third phase, the end nodes exchange measurement results of the overlapping segments between end-to-end paths, and use statistical processing to improve the measurement accuracy of these segments, thus consequently improve the measurement accuracy of the whole path. The measurement results of the overlapping segments are obtained by sending probe traffics to the two end nodes or routers of the segments. Simulation results show that the relative error in the measurement results of our method can be decreased by half compared with the existing method when the total measurement overheads of both methods are equal. We also confirm that the overhead of information exchange is very small and negligible comparing to the measurement overhead.

1.3.2 Measurement method for end-to-end available bandwidth

We produce a method for measuring the end-to-end available bandwidth in Chapter 3. Because the measurement results of the overlapping segment of two end-to-end paths can not be obtained from the measurements of the end nodes, we can not apply the method for measuring the additive metrics in Chapter 2. We therefore take a different approach, by trying to reduce the measurement time and traffic of each measurement, thus can help to reduce the measurement conflict, and consequently enhance measurement accuracy. For each measurement, we apply an algorithm similar to that of Pathload [14], one of the most efficient tools for measuring end-to-end available bandwidth. This tool finds the range of available bandwidth between a predetermined initial search range, by repeatedly sends probe packets with the sending rates vary based on the changes of the intervals

between probe packets, according to a binary search procedure. In our method, the end nodes share the measurement results of overlapping paths and related information to configure an initial search range, that is narrow and near the actual available bandwidth. By doing this, our method can reduce the number of iterations of sending probe packets, thus reduce measurement time and traffic of each measurement. Our method bases on two observations. First, the available bandwidth varies gradually. Therefore, we can use recent measurement results to estimate the initial search range. Second, if the bottleneck links of two overlapping paths belong to their overlapping segment, then the measurement results of two path are equal. Thus, we can also use the recent measurement results of overlapping paths for estimating the initial search range. Our method is obvious from these observations. That is, the end nodes save recent measurement results and exchange measurement results of overlapping paths, as long as the probability that bottleneck link belongs to the overlapping segment. The end nodes then use statistical processing for these data to calculate the initial search range. Simulation results show that the initial search range estimated by our method is much narrower that the default value of initial search range in Pathload. We also compare our method with an existing method using simulations, and the results suggest that the relative errors in the measurement results of our method are approximately only 65% of those of the existing method. Furthermore, the measurement accuracy of our method remains better than the existing method when the total measurement traffic loads of both methods are equal.

1.3.3 Measurement method for link fault diagnosis

We also propose a method for diagnosing network failures in Chapter 4. A typical fault diagnose procedure contains two phases: the fault detection phase, whose purpose is to periodically check if there are some problems in the network, and the fault localization phase, whose purpose is to rapidly locate the components responsible for the detected problems. In the fault detection phase, most of the existing researches only produce a spatial solution. That is they only focus on how to select a set of probe paths that can cover all of the links of the network. In this thesis, we propose not only the spatial solution but also a temporal solution: an algorithm for dynamically determined the probe timings that can distribute the probe timings all over the monitoring period, thus can

help to detect faulty components faster. We propose two algorithms that not only can rapidly detect network failures and locate faulty components, but also can reduce the total measurement traffic and well balance the measurement traffic load between the links of the network. Similar to our previous researches, we also utilize information exchange of measurement results of overlapping paths to reduce the measurement traffic. In the fault detection phase, we set constrains of measurement traffic for each link of the network, and probe the paths that satisfy the constrains with maximum measurement efficiency. On the other hand, in the fault localization phase, we choose the paths for measurements so that the expectation value of suspected faulty components after probing the paths is minimum. Simulation results show that our method can detect failures much faster than existing method while keeping the measurement traffic well-distributed between the links.

Chapter 2

Measurement method for end-to-end additive quality metrics

2.1 Introduction

Recently, overlay networks have attracted much attention as a technology that enables early deployment of new network services without standardization processes. Applications of overlay networks include end-system multicast (e.g., Narada [15]), P2P systems (e.g., Skype [16], KaZaA [17], BitTorrent [18]), content distribution systems (e.g., Akamai [19]), and resilient routing (e.g., RON [1]).

In overlay networks, the overlay nodes are often installed on end hosts as an application program. In this case, routing and traffic control at the overlay detecting level are conducted at the end hosts, and such controls cannot be activated inside the network. On the other hand, the overlay routing inside the network becomes possible by installing overlay nodes on the routers in the network. This installation has been simplified with such techniques as network virtualization [20] and software defined network [21]. In this chapter, to realize efficient routing control by overlay networks, we consider an overlay network in which the overlay nodes are deployed on the routers.

An overlay network should obtain the network resource information of the underlay network, including available bandwidth, propagation delay, and packet loss ratio, to maintain and improve the performance of network service. These metrics should be measured frequently to obtain high

measurement accuracy. RON [1] is one early-stage instance that measures all paths among overlay nodes. The measurement overhead becomes $O(n^2)$, where n is the number of overlay nodes. Therefore, [22] pointed out that the number of overlay nodes that can be applied is up to around fifty. Many solutions have been proposed to reduce measurement overhead [4–7, 23–25]. However, these methods have shortcomings in terms of measurement accuracy [23] or available measurement metrics [7, 24].

Measurement accuracy is affected not only by the way measurements are performed but also by the overlap of underlay paths among overlay nodes. Fig. 1.1 illustrates an example of overlapping paths. O_i and R_i ($i = 1, ..., 5$) represent overlay nodes and routers. Although paths O_1O_4 and O_2O_5 are disjointed at the overlay level, they overlap at the underlay level, i.e., they share links and routers on the path between R_1 and R_5. Therefore, the concurrent measurement tasks of paths O_1O_4 and O_2O_5 compete on the common links for network resources (e.g., processing power at routers and link bandwidth), causing high load on the common links and additional error in the measurement results.

[12] addresses this problem and proposes a method that schedules the timing of the measurement tasks of the overlay paths so that measurement conflicts can be avoided completely. However, the measurement frequency in this method is limited because of the heuristic behavior of the proposed scheduling algorithms [26]. Moreover, the methods in [4–6, 12, 23, 25] require a master node to aggregate the complete topology information of the underlay (IP) network, decide measurement timings, and give instructions to each overlay node. Therefore, the amount of time and network traffic for the aggregation of topology information and instructions are large, and the performance of overlay networks decreases when changes occur in the underlay or overlay networks.

In this chapter, we propose a distributed measurement method that can reduce measurement conflicts and obtain high measurement accuracy. In our proposed method, each overlay node exchanges route information with its neighboring overlay nodes to detect the overlapping paths. Overlapping paths with the same source node are measured sequentially to completely avoid measurement conflicts. Overlapping paths having different source nodes are randomly measured to reduce measurement conflicts. The overlay node then exchanges the measurement results with its neighboring overlay nodes to statistically improve measurement accuracy. Our method can also lower

the measurement frequencies to reduce overhead and measurement conflicts.

We make the following contributions in this chapter:

- We propose two algorithms for detecting the overlapping paths that do not require complete topology knowledge of the IP network at each node.
- We propose a method for determining the measurement frequencies and timings of the overlapping paths to reduce measurement conflicts.
- We evaluate our method and compare it with the method in [12] by simulations with both generated and real Internet topologies.

From the simulation results, we reach the following conclusions:

- Our method detects more than 90% of the overlapping paths with less than 30% of the information exchanges of the full-mesh method.
- When the overheads of our method and the method in [12] are equal, the relative error of the measurement results of our method is less than half of the method in [12].

The remainder of this chapter is organized as follows. Section 2.2 describes related work. In Section 2.3, we explain our method for detecting the overlapping of overlay paths. Section 2.4 describes our technique for reducing measurement conflicts and improving measurement accuracy. In Section 2.5, our proposed method is evaluated by simulations. We conclude this chapter and discuss future work in Section 2.6.

2.2 Related work

RON [1] can measure many network resource information of the underlay network such as available bandwidth, propagation delay and packet loss ratio, but it suffers from a lack of scalability. Therefore, the measurement methods proposed later tried to reduce the measurement overhead from the $O(n^2)$ overhead of RON. Network tomography [4–6, 23, 25] is an effective approach to achieve this goal. The main idea of these methods is that they monitor only a few paths that cover all the links

of the overlay network and use the measurement results of the collected paths to infer the measurement results of the remaining paths. However, the centralized behavior of these methods makes it hard for them to cope with changes or troubles that occur in the underlay network.

The measurement conflict problem, which was first addressed in [11], is considered in later work [12, 13]. The main idea of these studies is that they use heuristic algorithms from graph theory to schedule the measurement timings of paths so that the overlapping paths are measured at different timings. Although measurement conflicts can be avoided completely, the measurement frequencies are limited, so measurement accuracy is not high. We also point out that when the measurement traffic is not so intrusive, for example, when the measurement metric is latency, it is not necessary to completely avoid measurement conflicts.

Only a few measurement methods work in a distributed fashion [3, 24], and they have their own limits. The authors in [24] proposed a measurement system for available bandwidth, called ImSystemPlus, that can reduce measurement conflicts without using a master node by randomly deciding the measurement timing of overlapping paths. However, this method requires complete topology knowledge of the IP network at each overlay node. [3] proposed a measurement system in which overlay nodes estimate their virtual coordinates and exchange with each other to calculate the distances between them and infer latencies from those distances. However, this method cannot be applied to measure packet loss and bandwidth.

2.3 Detecting overlapping paths

2.3.1 Network model and definitions

We consider a network with m routers, denoted by R_i ($i = 1, ..., m$). We denote the underlay path between two routers R_i and R_j as R_iR_j. If two different paths R_iR_j and R_sR_t share at least one link, we say that R_iR_j and R_sR_t overlap with each other, or R_iR_j (R_sR_t) is an *overlapping path* of R_sR_t (R_iR_j).

Suppose that there are n ($n \leq m$) overlay nodes deployed on n routers. Density σ of the overlay nodes is defined as the ratio of the number of overlay nodes to the number of routers, i.e., $\sigma = n/m$.

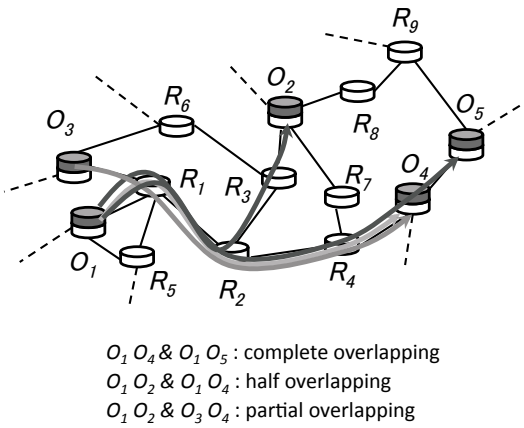


Figure 2.1: Classification of path overlapping

We denote the overlay nodes as O_i ($i = 1, ..., n$) and call the path between two overlay nodes an *overlay path*. For overlay path O_iO_j, O_i is the *source node*, and O_j is the *destination node* of the overlay path.

Figure 2.1 shows a classification of the overlapping state of overlay paths. In this chapter, we classify overlapping states into the following three types:

- Complete overlapping: One overlay path completely includes another overlay path.
- Half overlapping: Two overlay paths share a route from the source node to a router that is not an overlay node.
- Partial overlapping: Two overlay paths share a route that does not include the source node.

For example, in Fig. 2.1, path O_1O_4 is a complete overlapping path of O_1O_5. Paths O_1O_2 and O_1O_4 have a half overlapping relation. Path O_1O_2 is a partial overlapping path of O_3O_4.

2.3.2 Methods for detecting complete and half overlapping paths

Complete overlapping and half overlapping can be detected by the source node of the overlay path using `traceroute`-like tools, as described in [27]. For example, in Fig. 2.1, when overlay node O_1 issues `traceroute` to O_4 and O_5, complete overlapping of paths O_1O_4 and O_1O_5 can

be detected. Similarly, the shared route from O_1 to router R_2 by paths O_1O_2 and O_1O_4 can be detected when O_1 issues `traceroute` to O_2 and O_4.

2.3.3 Method for detecting partial overlapping paths

Detecting algorithms

Partial overlapping cannot be precisely detected only by `traceroute`-like tools, because the source nodes of the partial overlapping paths are different. Therefore, in this subsection, we propose the following method for detecting partial overlapping paths.

We demonstrate how an overlay node O_i detects the partial overlapping paths. We denote the set of overlay paths whose source nodes are O_i, which contain at least two links and do not completely include other overlay paths as $\mathcal{S}_{O_i}$. We also denote the set of overlay paths whose destination nodes are O_i, which contain at least two links and do not completely include other overlay paths as $\mathcal{D}_{O_i}$. Note that we exclude one-link paths when defining $\mathcal{S}_{O_i}$ and $\mathcal{D}_{O_i}$ since they do not have partial overlapping paths. Also, we do not directly measure the paths that completely include other overlay paths, as described in Subsection 2.4.1.

Our method consists of two steps that detect the partial overlapping paths of each path in $\mathcal{S}_{O_i}$ and $\mathcal{D}_{O_i}$, respectively. In the first step, O_i finds the candidates of the partial overlapping paths of the paths in $\mathcal{S}_{O_i}$. O_i then exchanges the path information with the source nodes of the candidates to confirm whether they are actually partial overlapping paths. In the second step, O_i exchanges the information of the paths in $\mathcal{D}_{O_i}$ with their source nodes to detect their partial overlapping paths.

Algorithm 1 shows the details of the first step. Function $OverlapLength$ returns the length (number of hops) of the overlapping part between two paths. In this algorithm, O_i finds the candidates of the partial overlapping paths of each path O_iO_j in $\mathcal{S}_{O_i}$ by utilizing the information of its half overlapping paths. In detail, when O_iO_s and O_iO_t are half overlapping paths of O_iO_j and when the length of the overlapping part of O_iO_j and O_iO_s is smaller than the length of the overlapping part of O_iO_j and O_iO_t, we infer that O_sO_t is a candidate of the partial overlapping path of O_iO_j. O_i then exchanges path information with O_s to determine whether O_iO_j and O_sO_t

Algorithm 1 O_i detects the partial overlapping paths of the paths in $\mathcal{S}_{O_i}$

1: //initilization
2: **for** $O_iO_j \in \mathcal{S}_{O_i}$ **do**
3: $\mathcal{C}_{O_iO_j} \leftarrow \emptyset$ //set of candidates of partial overlapping paths of O_iO_j
4: $\mathcal{N}_{O_iO_j} \leftarrow \emptyset$ //set of nodes that receives information of O_iO_j
5: **end for**
6: **for** $O_j \neq O_i$ **do**
7: $\mathcal{T}_{O_i}^{O_j} \leftarrow \emptyset$ //set of paths that O_i sends to O_j
8: $\mathcal{R}_{O_i}^{O_j} \leftarrow \emptyset$ //set of paths that O_i receives from O_j
9: **end for**
10: //find candidates of partial overlapping paths
11: **for** $O_iO_j \in \mathcal{S}_{O_i}$ **do**
12: **for** each pair O_iO_s, O_iO_t of half overlapping paths of O_iO_j **do**
13: **if** $OverlapLength(O_iO_j, O_iO_s) < OverlapLength(O_iO_j, O_iO_t)$ **then**
14: $\mathcal{C}_{O_iO_j} \leftarrow \mathcal{C}_{O_iO_j} \cup \{O_sO_t\}$
15: **else if** $OverlapLength(O_iO_j, O_iO_s) > OverlapLength(O_iO_j, O_iO_t)$ **then**
16: $\mathcal{C}_{O_iO_j} \leftarrow \mathcal{C}_{O_iO_j} \cup \{O_tO_s\}$
17: **end if**
18: **end for**
19: **end for**
20: //update set of paths that O_i sends to other nodes
21: **for** $O_iO_j \in \mathcal{S}_{O_i}$ **do**
22: **for** $O_sO_t \in \mathcal{C}_{O_iO_j}$ **do**
23: $\mathcal{T}_{O_i}^{O_s} \leftarrow \mathcal{T}_{O_i}^{O_s} \cup \{O_iO_j\}$
24: **end for**
25: **end for**
26: //O_i exchanges information of paths with other nodes
27: **for** $O_j \neq O_i$ **do**
28: **loop**
29: **for** $O_iO_s \in \mathcal{T}_{O_i}^{O_j}$ **do**
30: O_i sends information of O_iO_s to O_j
31: $\mathcal{N}_{O_iO_s} \leftarrow \mathcal{N}_{O_iO_s} \cup \{O_j\}$
32: **end for**
33: $\mathcal{T}_{O_i}^{O_j} \leftarrow \emptyset$ //clear the set $\mathcal{T}_{O_i}^{O_j}$
34: O_i receives information of paths from O_j and adds it to set $\mathcal{R}_{O_i}^{O_j}$
35: O_i detects partial overlapping between the paths in $\mathcal{S}_{O_i}$ and the paths in $\mathcal{R}_{O_i}^{O_j}$
36: //update the set $\mathcal{T}_{O_i}^{O_j}$
37: **if** there are some paths in $\mathcal{S}_{O_i}$ that overlap with at least one path in $\mathcal{R}_{O_i}^{O_j}$ and have not been sent to O_j **then**
38: Add these paths to $\mathcal{T}_{O_i}^{O_j}$
39: **end if**
40: //stop if there is no more information of paths to send
41: **if** $\mathcal{T}_{O_i}^{O_j} = \emptyset$ **then**
42: exit loop
43: **end if**
44: **end loop**
45: **end for**

Algorithm 2 O_i detects the partial overlapping paths of the paths in $\mathcal{D}_{O_i}$

1: //O_i sends path information
2: **for** $O_iO_j \in \mathcal{S}_{O_i}$ **do**
3: O_i sends information of O_iO_j and $\mathcal{N}_{O_iO_j}$ to O_j
4: **end for**
5:
6: //O_i receives path information
7: $\mathcal{D}_{O_i} \leftarrow \emptyset$
8: **for** $O_j \neq O_i$ **do**
9: O_i receives information of O_jO_i and set $\mathcal{N}_{O_jO_i}$ from O_j
10: $\mathcal{D}_{O_i} \leftarrow \mathcal{D}_{O_i} \cup \{O_jO_i\}$
11: **end for**
12:
13: //O_i detects partial overlapping paths and sends to other nodes
14: **for** each pair $O_sO_i, O_tO_i \in \mathcal{D}_{O_i}$ **do**
15: **if** O_sO_i and O_tO_i overlap with each other **then**
16: **if** $O_t \notin \mathcal{N}_{O_sO_i}$ **then**
17: O_i sends information of O_sO_i to O_t
18: **end if**
19: **if** $O_s \notin \mathcal{N}_{O_tO_i}$ **then**
20: O_i sends information of O_tO_i to O_s
21: **end if**
22: **end if**
23: **end for**
24:
25: O_i receives the partial overlapping paths of paths in $\mathcal{S}_{O_i}$ from other nodes

actually have a partial overlapping relation. In this way, O_i exchanges path information with the source nodes of the candidates to decide their overlapping states. Furthermore, when receiving path information from other nodes, O_i may find new candidates of the partial overlapping paths. In that case, O_i repeats the information exchange and the decisions of the overlapping states.

We use Fig. 2.1 to explain how Algorithm 1 works for path O_1O_2. Set $\mathcal{S}_{O_1}$ includes O_1O_2, O_1O_3, and O_1O_4 and does not include O_1O_5 because it completely contains O_1O_4. We infer that path O_3O_4 is a partial overlapping path of O_1O_2, because the length of the overlapping part of O_1O_2 and O_1O_3 is smaller than the length of the overlapping part of O_1O_2 and O_1O_4. O_1 then exchanges path information with O_3 to confirm whether O_1O_2 and O_3O_4 actually have a partial overlapping relation.

Algorithm 2 shows the details of the second step. In this algorithm, O_i exchanges path information with other nodes to detect the partial overlapping paths of the paths in $\mathcal{D}_{O_i}$ as follows.

1. O_i receives information of each path in $\mathcal{D}_{O_i}$ from the source node (referred to as O_s) of the path.

2. O_i detects the partial overlapping paths of each path O_sO_i in $\mathcal{D}_{O_i}$ and sends information of these paths to O_s.

We also use Fig. 2.1 to explain how Algorithm 2 works for path O_2O_4. Set $\mathcal{D}_{O_4}$ includes O_1O_4, O_2O_4, and O_3O_4 and does not include O_5O_4 because it contains only one link. First, O_4 receives the information of paths O_1O_4, O_2O_4, and O_3O_4 from O_1, O_2, and O_3, respectively. O_4 then detects that O_1O_4, O_2O_4, and O_3O_4 are in a partial overlapping relation and sends the information of O_1O_4 and O_3O_4 to O_2.

Evaluation of detecting algorithms

We evaluate our proposed algorithms for detecting partial overlapping paths by simulations with two metrics, defined as follows:

- detection ratio: ratio of the number of detected partial overlapping paths to the actual number of partial overlapping paths.

- number of path information exchanges: number of times that the information of overlay path was exchanged among the overlay nodes.

Algorithm 1 includes iterations for information exchange and the decision of the overlapping states. When the number of iterations increases the detection ratio is enhanced, while the overhead of the information exchange among the overlay nodes also increases. In addition, since Algorithms 1 and 2 can be conducted independently, we set the following four detecting levels to conduct Algorithms 1 and 2 to investigate the trade-off relationships between the detection ratio and the information exchange overhead.

- detecting level 1: run Algorithm 1 with one iteration.
- detecting level 2: run Algorithm 1 with two iterations.
- detecting level 3: run Algorithm 1 completely.
- detecting level 4: run Algorithms 1 and 2 completely.

For the underlay network topology, we used the AT&T topology obtained from [28]. We also utilized generated topologies based on BA [29] and random models [30]. We generated ten topologies for each model using the BRITE topology generator [31]. All topologies have 523 nodes and 1304 links. We set the density of the overlay nodes to 0.2 and randomly chose them. For averaging the results, the choice of the overlay nodes was taken 100 times for the AT&T topology and ten times for each topology of the BA and random models.

We compared our method with the full-mesh method when evaluating the number of path information exchanges. In the full-mesh method, each overlay node sends information of all overlay paths departing from it to all other overlay nodes. When the number of overlay nodes is n, the number of path information exchanges of the full-mesh method is $n(n-1)^2$, which becomes $1{,}103{,}336$ in the evaluation results.

Figures 2.2 and 2.3 show the average values and 95% confidence intervals of detection ratio of the partial overlapping paths and the number of path information exchanges, respectively. The black, gray and white bars show the results of the AT&T topology, the BA topologies, and random

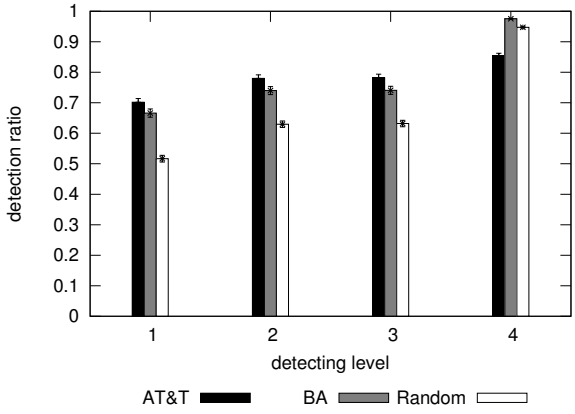


Figure 2.2: Average detection ratio of partial overlapping paths

topologies, respectively. The line in Fig. 2.3 represents the number of path information exchanges of the full-mesh method. As shown in these figures, our method needs only $1/6$ and $1/3$ of the path information exchanges to detect about 60% and 90% of the partial overlapping paths at detecting levels 1 and 4, respectively. The results of detecting levels 2 and 3 are very close, meaning that we only need to run two iterations of the exchange loop of Algorithm 1.

Solution for topology measurement errors

The proposed method relies on the assumption that all of the routers on the paths between overlay nodes appropriately respond to `traceroute`. In the case that some of the routers do not respond to `traceroute`, we apply a method similar to the one proposed in [4]. More specifically, if path O_iO_j contains some routers between R_s and R_t that do not respond to `traceroute`, then we consider the path between R_s and R_t as a "virtual link", and apply the proposed method as usual.

2.4 Measurement method for overlay paths

In this section, we propose a method for reducing the measurement conflicts based on the status of the path overlapping detected by the method in Section 2.3. We explain the proposed method by describing the detailed behavior for an overlay path O_iO_j. First, node O_i detects the overlapping

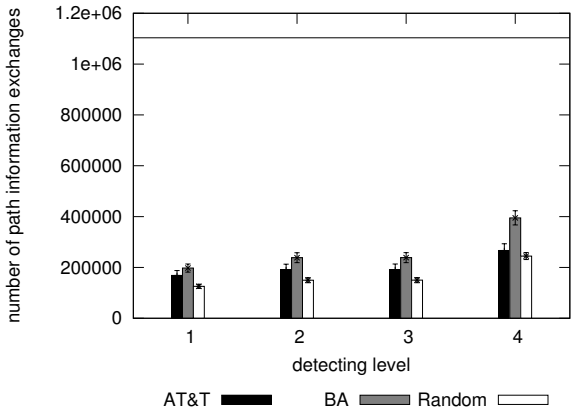


Figure 2.3: Average number of path information exchanges

paths of path O_iO_j with the method described in Section 2.3. If path O_iO_j has no overlapping paths, it is unnecessary to consider a method for reducing measurement conflicts. Therefore, we are only concerned with the case when path O_iO_j overlaps with other overlay paths.

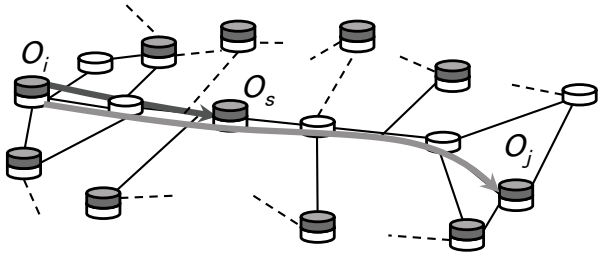
We consider the following two cases of overlapping states:

1. When path O_iO_j completely includes other overlay paths, overlay path O_iO_j is not measured directly.

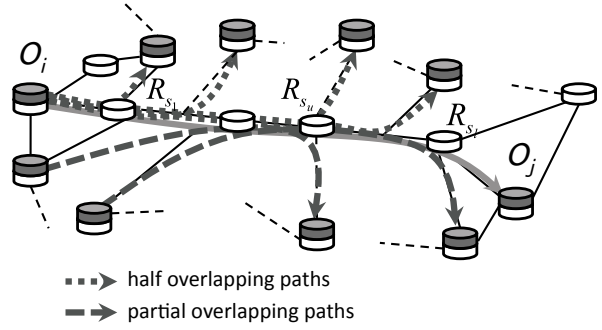
2. When path O_iO_j does not include other overlay paths, we adjust the frequency and timing of the measurements to reduce the measurement conflicts.

The detailed mechanisms for the above two cases are described in Subsects. 2.4.1 and 2.4.1, respectively. In Subsection 2.4.2, we propose a statistical method for improving the accuracy of the measurement results.

Finally, in Subsection 2.4.3, we describe the entire procedure for each overlay node to measure the overlay paths departing from it.



(a) Complete overlapping



(b) Half and partial overlapping

Figure 2.4: Examples for explaining the proposed measurement method

2.4.1 Reducing measurement conflicts

Complete overlapping

In this case, the overlay path that includes the other overlay paths is not measured directly. Instead, the measurement result is estimated based on the measurement results of the overlay paths included in it.

We use Fig. 2.4(a) to explain this method. As shown in Fig. 2.4(a), path O_iO_j completely includes path O_iO_s. When O_i issues `traceroute` to O_j, the `traceroute` packet goes through O_s, which learns that it is on path O_iO_j. O_s then measures path O_sO_j and transmits the result to O_i, which also learns that O_s is on path O_iO_j, based on the `traceroute` result. Then O_i does not directly measure path O_iO_j; it only measures path O_iO_s. O_i estimates the measurement result of path O_iO_j from the measurement result of path O_iO_s and that of path O_sO_j received from O_s. See [27] for details. Note that this method dramatically reduces the number of measurement

paths, especially when the density of the overlay nodes is large [27]. Furthermore, the reasonable measurement accuracy of such a spatial composition method has been confirmed [32].

Half and partial overlapping

Here, we assume that O_iO_j has $(G_{i,j} - 1)$ half overlapping paths ($G_{i,j} \geq 1$), as shown in Fig. 2.4(b). For simplicity, we rewrite $G_{i,j}$ as G. We denote path O_iO_j as path 1, and each of its half overlapping paths as path p ($2 \leq p \leq G$). Furthermore, we assume that, with the method described in Section 2.3 to detect partial overlapping paths, path p ($1 \leq p \leq G$) has $(K_p - 1)$ partial overlapping paths ($K_p \geq 1$).

Overlay node O_i can avoid the measurement conflicts between half overlapping paths 1, 2, ... and G simply by measuring them sequentially. On the other hand, because the source nodes of the partial overlapping paths of path p are different, measurement conflicts between them cannot be avoided completely. Therefore, we propose a technique that combines a sequential measurement for half overlapping paths and a random measurement for partial overlapping paths.

We define the *measurement frequency* as follows. We assume that the time required for each measurement task is identical for all overlay paths and denote it as τ. We also assume that the measurement results of path p are aggregated in the time duration of T_p ($T_p \geq \tau$). We call T_p an *aggregation period*. When a path is measured q ($q \leq T_p/\tau$) times at an aggregation period, its measurement frequency at that aggregation period is defined as $f_p = q\tau/T_p$.

We introduce β_p as a value that reflects the dispersion of the measurement results of path p at an aggregation period. Note that the method to determine β_p is beyond the scope of this thesis. β_p can be calculated based on the statistics of the measurement results or using the method in [24]. We set measurement frequency f_p proportional to β_p for all paths, i.e., $f_1/\beta_1 = f_2/\beta_2 = ... = f_G/\beta_G$. To avoid measurement conflicts between half overlapping paths, the sum of their measurement frequencies should be equal to or less than one, i.e., $\sum_{p=1}^{G} f_p \leq 1$. So we have $f_p \leq \beta_p/(\sum_{s=1}^{G} \beta_s)$.

To reduce the probability of measurement conflicts between path p and its $(K_p - 1)$ partial overlapping paths, we set the measurement frequency of path p to a value equal to or less than $1/K_p$, i.e., $f_p \leq 1/K_p$. In addition, we keep the measurement frequencies as large as possible to

obtain as many measurement results as possible. Therefor, the measurement frequency of path p is decided based on the following equation:

$$f_p = \min\{\beta_p/(\sum_{s=1}^{G}\beta_s), 1/K_p\}. \tag{2.1}$$

Next, we explain our method for randomly deciding the measurement timings of path p so that the probability that the measurement of path p is carried out becomes f_p. We define a *measurement cycle* for the measurements of paths 1, 2, ... and G. We also divide the measurement cycle into multiple *measurement time slots*, each of which is assigned to the measurement of each path. We consider a scheme for allocating the measurement timings of paths p to these measurement time slots as follows.

When a path is measured at one measurement time slot of the measurement cycle, the probability that the measurement of the path is carried out becomes $1/G$. Therefore, we compare f_p with $1/G$ when considering the measurement timings of path p. We assume that $f_1 \geq f_2 \geq ... \geq f_G$ without loss of generality. For convenience, we define dummy value $f_0 = 1$. Since $\sum_{s=1}^{G} f_s \leq 1$, $0 \leq l < G$ exists, such that $f_0 \geq ... \geq f_l \geq 1/G \geq f_{l+1} \geq ... \geq f_G$.

If $l = 0$, meaning $f_p \leq 1/G, \forall 1 \leq p \leq G$, one measurement time slot in the measurement cycle is enough to allocate measurement timings for each path p.

On the other hand, $l > 0$ means that for path s where $s > l$, one measurement time slot is enough to allocate its measurement timings. For path t where $t \leq l$, one measurement time slot is not enough for allocating its measurement timings to satisfy its measurement frequency. In this case, the measurement time slot allocated to path s where $s > l$ is also used to measure path t where $t \leq l$ when path s is not measured.

In detail, we propose the following scheme for allocating the measurement timings of all paths.

1. Randomly decide the measurement order of path p ($1 \leq p \leq G$) at one measurement circle, and allocate the measurement time slot for each path.

2. - If $l = 0$,

     We measure path p with the probability of Gf_p at the measurement time slot allocated

to it.

- If $l \geq 1$,
  - For path t where $t \leq l$, we measure it at the measurement time slot allocated to it.
  - For path s where $s > l$, we measure it with the probability of Gf_s at the measurement time slot allocated to it.

    If path s ($s > l$) is not measured, the measurement time slot is used to measure path t ($t \leq l$) with the probability of $(f_t - 1/G)/\delta$, where $\delta = \sum_{s=l+1}^{G} (1/G - f_s)$.

2.4.2 Statistical method for improving the accuracy for measurement results

In the proposed measurement methods in Subsection 2.4.1, because it is impossible to completely avoid measurement conflicts with partial overlapping paths, the accuracy of the measurement results decreases due to measurement conflicts. Therefore, in our proposed method, overlay nodes exchange measurement results and use statistical processing to improve measurement accuracy. We assume the measuring metric is delay.

We use Fig. 2.4(b) to explain the method for path O_iO_j. We assume that the overlapping parts of O_iO_j and its half and partial overlapping paths are divided by routers R_{s_1}, R_{s_2}, ..., R_{s_l}. In the proposed method, the delay measurements are individually conducted for overlapping parts $R_{s_1}R_{s_2}$, $R_{s_2}R_{s_3}$, ..., $R_{s_{l-1}}R_{s_l}$ as well as for end-to-end path O_iO_j. In detail, O_i measures the delays to routers R_{s_1}, R_{s_2}, ..., R_{s_l} and calculates the delay of $O_iR_{s_1}$, $R_{s_1}R_{s_2}$, ..., $R_{s_{l-1}}R_{s_l}$ and $R_{s_l}O_j$ as follows, where the delays of $O_iR_{s_1}$, $O_iR_{s_2}$, ..., $O_iR_{s_l}$, and O_iO_j are denoted as $t_{O_iR_{s_1}}$,$t_{O_iR_{s_2}}$,...,$t_{O_iR_{s_l}}$,$t_{O_iO_j}$, respectively.

$$\begin{aligned} t_{R_{s_k}R_{s_{k+1}}} &= t_{O_iR_{s_{k+1}}} - t_{O_iR_{s_k}} \quad , k = 1, ..., l-1 \\ t_{R_{s_l}O_j} &= t_{O_iO_j} - t_{O_iR_{s_l}} \end{aligned} \tag{2.2}$$

When part $O_iR_{s_1}$ or $R_{s_k}R_{s_{k+1}}$ is the overlapping part of O_iO_j and its half overlapping path O_iO_s, $t_{O_iR_{s_1}}$ or $t_{R_{s_k}R_{s_{k+1}}}$ is used to calculate the measurement results of both paths O_iO_j and O_iO_s. When part $R_{s_k}R_{s_{k+1}}$ or $R_{s_l}O_j$ is the overlapping part of O_iO_j and its partial overlapping path O_uO_v, O_i sends $t_{R_{s_k}R_{s_{k+1}}}$ or $t_{R_{s_l}O_j}$ and its measurement timing to O_u, so that O_u can use

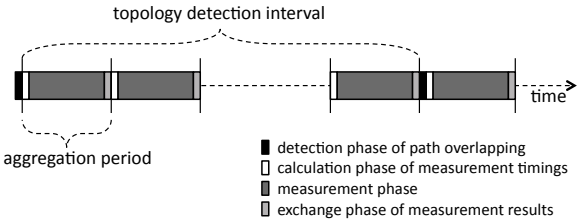


Figure 2.5: Measurement procedure

$t_{R_{s_k}R_{s_{k+1}}}$ or $t_{R_{s_l}O_j}$ to calculate the measurement result of path O_uO_v.

Finally, we use statistical processing for the data obtained by information exchange to calculate the measurement result of path O_iO_j. First, using the gathered values with the above method, we obtain the average value of the measurement results of $O_iR_{s_1}$, $R_{s_1}R_{s_2}$, ..., $R_{s_{l-1}}R_{s_l}$, and $R_{s_l}O_j$, which are denoted as $\bar{t}_{O_iR_{s_1}}$, $\bar{t}_{R_{s_1}R_{s_2}}$, ..., $\bar{t}_{R_{s_{l-1}}R_{s_l}}$, and $\bar{t}_{R_{s_l}O_j}$, respectively. The measurement result of path O_iO_j is then calculated as follows.

$$\bar{t}_{O_iO_j} = \bar{t}_{O_iR_{s_1}} + \sum_{k=1}^{l-1} \bar{t}_{R_{s_k}R_{s_{k+1}}} + \bar{t}_{R_{s_l}O_j} \tag{2.3}$$

The main idea of the above method is that source nodes of partial overlapping paths exchange measurement results of the overlapping parts to improve the measurement accuracy of these parts, and consequently improve the measurement accuracy of the whole path. Therefore, this method can be applied similarly to the metrics that the measurement results of overlapping parts can be obtained from the measurement results of the paths from the source node to the routers in the overlapping parts. These metrics include latency, loss rate, jitter, etc.

However, when the metric is bandwidth-related information such as available bandwidth or throughput, because the measurement results of overlapping parts can not be obtained, we can not apply the above method. The methods for bandwidth-related metrics are our future work.

2.4.3 Measurement procedure

The measurement procedure of an overlay node includes the following four phases:

- Detection phase of path overlapping

  The overlay nodes detect the path overlapping using the method described in Section 2.3.

- Calculation phase of measurement timings

  The measurement frequencies and timings are calculated based on the status of the path overlapping, as described in Subsection 2.4.1.

- Measurement phase

  The measurements are performed at the calculated measurement timings.

- Exchange phase of measurement results

  The overlay nodes exchange measurement results and calculate the measurement results of the overlay paths, as described in Subsection 2.4.2.

Figure 2.5 illustrates the relationships among phases. The phases of the calculations of measurement timings, the measuring, and the measurement results exchange are performed at each aggregation period. Because the frequency of the change in the underlay network is generally smaller than the frequency of the change in the measurement results, the interval between two phases of path overlapping detection is larger than an aggregation period. We call this interval a *topology detection interval*. In general cases, the length of detection phase of path overlapping is much smaller than that of measurement phase, because in detection phase of path overlapping, the actions of detecting and exchanging path information are performed immediately with no waiting time, while in measurement phase, measurements are performed several times, and there are large intervals between measurements to reduce measurement conflicts. The overheads of these phases are evaluated and discussed in Subsection 2.5.2.

2.5 Performance evaluation

In this section, we evaluate the performance of our proposed method by simulation experiments. We explain the evaluation method in Subsection 2.5.1 and present evaluation results and discussions in Subsection 2.5.2.

2.5.1 Evaluation method

We compared the proposed method with an existing method [12], which we briefly explain and make some assumptions about for comparison. We then explain the evaluation metrics and the simulation settings.

Existing method [12]

In the method in [12], a measurement task on an overlay path is represented by a vertex in a graph. Two vertexes that represent the measurement tasks on overlapping paths are connected by an edge. The authors proposed some heuristic algorithms from graph theory to divide the vertexes into some groups, so that each group contains only disconnected vertexes which represent measurement tasks of non-overlapping paths. The measurement tasks represented by vertexes in the same group are simultaneously performed, while the measurement tasks represented by vertexes in the different groups are sequentially performed. Therefore, measurement conflicts between overlapping paths are avoided completely.

However, in [12], a detail measurement method for applying these algorithms is not mentioned. Therefore, to compare it with our method, we assume that the method in [12] is applied to a centralized measurement system like the one described in [4]. In this system, a *master node* aggregates the information of overlay paths from other overlay nodes, schedules measurement timings for the overlay paths using the method in [12], instructs other overlay nodes to measure, and aggregates the measurement results from the other overlay nodes.

Evaluation metrics

Here, we assume the measuring metric is delay. We compare the proposed method and the method in [12] with the following metrics:

- Measurement accuracy

  We use the relative error of the measurement results as a metric to evaluate the measurement accuracy of the methods.

- System overhead

  We consider the following three kinds of overheads in conducting the measurements.

  - Path information accessing overhead

    This is caused when each overlay node uses `traceroute`-like tools to access the information of the overlay paths.

  - Measurement overhead

    This is caused when performing measurements on the overlay paths.

  - Information exchange overhead

    This is caused when overlay nodes exchange information of overlay paths and measurement results with other overlay nodes.

The relative error of the measurement result is calculated by:

$$\epsilon = \frac{|\bar{t} - t^*|}{t^*} \tag{2.4}$$

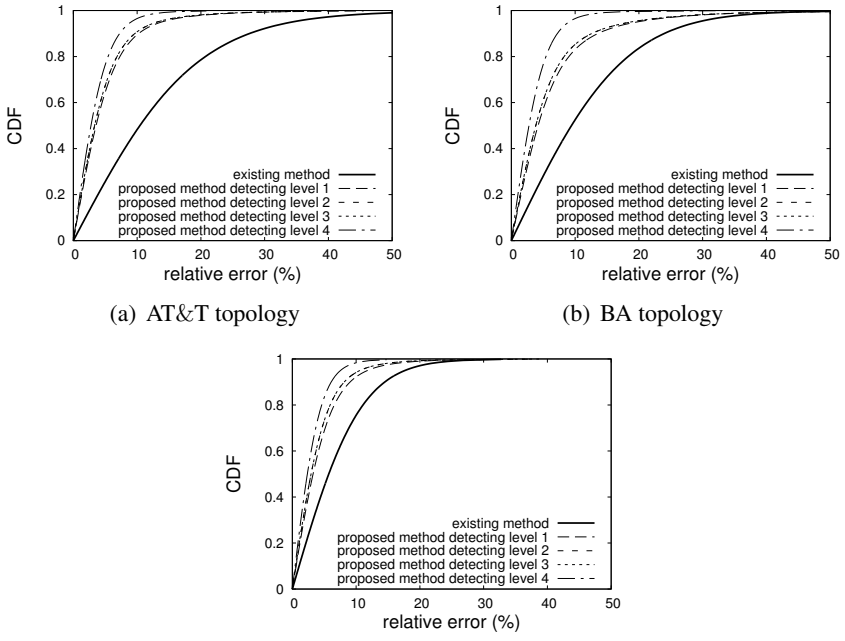
where t^* and $\bar{t}$ are the real delay and average values of the measurement results, respectively.

We use the M/M/1 queueing model for each link in the network to calculate t^* and $\bar{t}$. We assume that each measurement on a link causes the increase in the link utilization, that results in the increase of the delay and delay jitter at the link. When the number of concurrent measurements on a link increases, the link utilization also greatly increases, causing additional error in the delay measurements.

The system overhead, denoted by A, is calculated by:

$$A = \frac{s_a + s_m + s_e}{d} \tag{2.5}$$

where d is the duration during which the measurements were performed, and s_a, s_m and s_e are the sizes of the data packets used for accessing the path information, measuring, and exchanging the path information and the measurement results, respectively. We use second as the unit of d and bit as the unit of s_a, s_m and s_e. Therefore, the unit of A is bit per second (bps).



(a) AT&T topology (b) BA topology

(c) Random topology

Figure 2.6: Relative error of measurement results

Simulation settings

In obtaining the following simulation results, our assumptions on the network topologies, the number and the distribution of overlay nodes are the same as those mentioned in Subsection 2.3.3.

Value β_p, which is used for calculating the measurement frequencies by Eq. (2.1), is determined based on the coefficient of variance of the measurement results. Furthermore, we adjust the measurement frequencies in our method so that the system overheads of the proposed method and the method in [12] are the same.

We assume that we utilize `traceroute` to access information of overlay paths, and use `ping` to measure their delays. The size of each `traceroute` packet and `ping` packet is 28 and 475 bytes, respectively. We set the time of each measurement task $\tau = 1$ (second). An aggregation period is set to one hour, and an topology detection interval is set to ten hours. We set the utilization of each link in the network to 0.5 and assume that each measurement task increases the link utilization by 0.005.

Table 2.1: Average number of measurements during an aggregation period

Method \ Topology	AT&T	BA	Random
existing method	10.626	16.034	37.753
proposed method detecting level 1	7.918	10.531	20.424
proposed method detecting level 2	8.213	10.593	20.538
proposed method detecting level 3	8.211	10.602	20.538
proposed method detecting level 4	6.798	9.286	19.162

2.5.2 Evaluation results and discussions

Measurement accuracy

Figure 2.6 shows the distribution of the relative error in the measurement results. The relative errors in our method are about half of those in the method in [12]. In our method, the relative errors decrease from detecting levels one to four, and the measurement accuracy of detecting level four greatly surpasses the other detecting levels.

To explain these results, we use the evaluation results of the parameters related to measurement accuracy. Tables 2.1 and 2.2 show the average number of measurements of an overlay path and the average number of the measurement results of a link (in our method) or a path (in the method in [12]) gathered during an aggregation period, respectively. In our method, as explained in Subsection 2.4.2, the aggregated measurement results of each link of an overlay path include the results obtained from the measurements performed by its source node and the results received from other overlay nodes. On the other hand, in the method in [12], because measurement results are not exchanged among overlay nodes, the number of aggregated measurement results of an overlay path equals its measurement times. Table 2.3 shows the average number of concurrent measurements

Table 2.2: Average number of measurement results received during an aggregation period

Method \ Topology	AT&T	BA	Random
existing method	10.626	16.034	37.753
proposed method detecting level 1	130.323	141.577	187.547
proposed method detecting level 2	136.889	148.918	203.068
proposed method detecting level 3	136.852	149.077	203.199
proposed method detecting level 4	168.294	210.704	277.161

performed at a link. In the method in [12], because the measurement conflicts are avoided completely, this value remains one for all links. In our method, although the measurement conflicts cannot be avoided completely, we reduce them by adjusting the measurement frequencies based on the status of the path overlapping. Therefore, the average number of concurrent measurements of a link is very close to one.

As shown in these tables, in our method, although the number of measuring times is smaller than that in the method in [12], the number of aggregated measurement results is much larger, while the number of measurement conflicts is small. Therefore, the measurement accuracy of our method surpasses the method in [12].

We also observe in Tables 2.2 and 2.3 that when the detecting level of the proposed method is four, the number of measurement results is the largest, whereas the number of concurrent measurements is the smallest. This results in that the measurement accuracy at detecting level four is better than those at other detecting levels.

System overhead

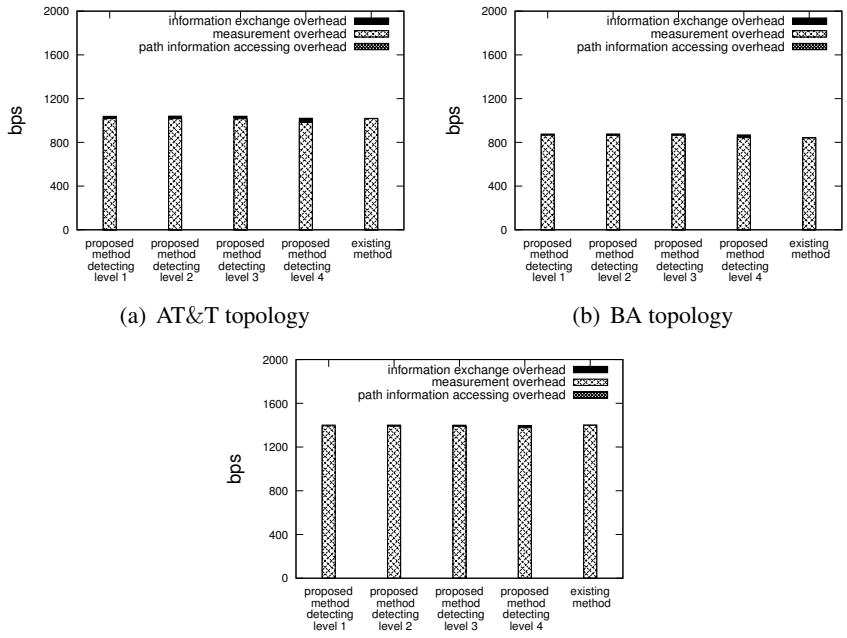
Figure 2.7 shows the average values of the system overhead of the method in [12] and our proposed method with four detecting levels. The system overheads of these methods are almost equal.

Table 2.3: Average number of concurrent measurements of one link

Method \ Topology	AT&T	BA	Random
existing method	1.000	1.000	1.000
proposed method detecting level 1	1.031	1.030	1.040
proposed method detecting level 2	1.029	1.027	1.036
proposed method detecting level 3	1.029	1.027	1.036
proposed method detecting level 4	1.022	1.018	1.022

Furthermore, the measurement overhead occupies the most part of the system overhead, and the information exchange overhead is very small while the path information accessing overhead is negligible. This is because of the following two reasons. First, the size of the measurement traffic is much larger than the size of the traffic of information exchange and path information accessing. Second, the access frequency of path information is smaller than the measurement frequency, because the frequency of the change in the underlay network is generally smaller than the frequency of the change in the measurement results. In our method, the information exchange overhead of detecting level four is slightly larger while the measurement overhead is smaller than those of the other detecting levels. This means that by shifting some amount of overhead from measurement to information exchange, we can significantly improve the measurement accuracy.

We also observe in Figs. 2.6 and 2.7 that random topology has the smallest relative error but the largest system overhead compared with AT&T and BA topologies. We explain these results as follows. From the simulation results, we found that the number of half overlapping paths and partial overlapping paths in random topology is smaller than that in AT&T and BA topologies [26]. Therefore, in the method in [12], the number of overlay paths that can be measured concurrently is the largest, meaning that the measurement frequency and the measurement overhead are the largest in random topology. Because the system overhead is occupied mostly by the measurement



(a) AT&T topology (b) BA topology

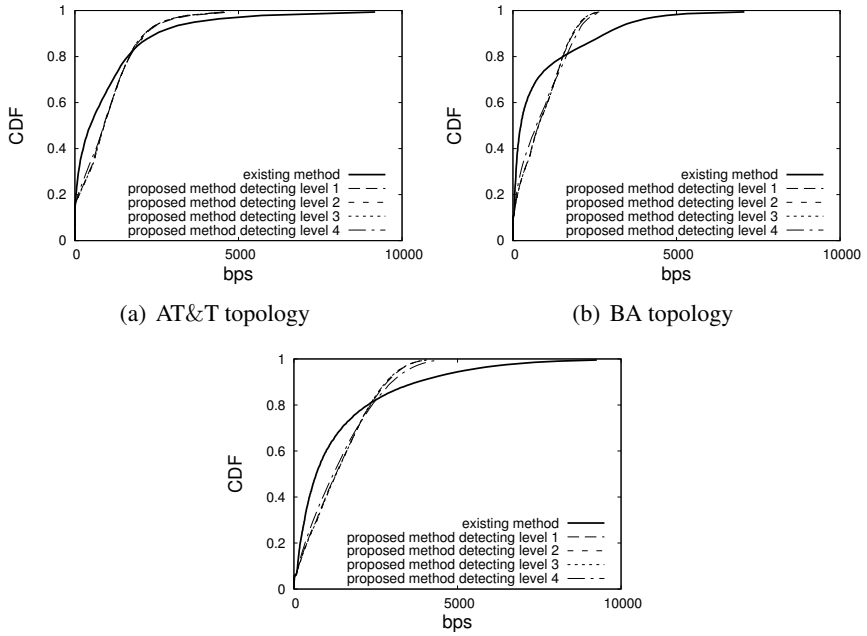
(c) Random topology

Figure 2.7: Average system overhead of one link

overhead, the system overhead is also the largest in random topology. Furthermore, because the measurement frequency in random topology is the largest among three network topologies, the relative error becomes the smallest. In our method, because we adjust measurement frequency of our method so that the method in [12] and our method have the same system overhead, we have the same result with the method in [12].

Figure 2.8 shows the distribution of system overhead on the links in the network. In the method in [12], the overhead is concentrated at several links, while in our method, the overhead is better balanced between links. This is one of side-effects of our hop-by-hop delay measurement method explained in Subsection 2.4.2.

We finally conclude that from the results in Figs. 2.6, 2.7 and 2.8, in our method, the detecting level four is the most effective for improving measurement accuracy. Note that the detecting levels one and two are still useful, because of the following two reasons. First, although measurement accuracy in detecting level one or two is slightly worse than that in the detecting level four, it is still



(a) AT&T topology (b) BA topology

(c) Random topology

Figure 2.8: Distribution of system overhead of all links in network

much better than that of the method in [12]. Second, it is easier to implement the proposed method at detecting level one or two since we only need to run Algorithm 1 with one or two iterations.

2.6 Summary

In this chapter, we proposed a distributed overlay network measurement method that reduces the measurement conflicts by detecting the path overlapping and adjusting the measurement frequencies and the measurement timings of overlay paths. We also proposed a method to improve measurement accuracy by exchanging measurement results among neighboring overlay nodes. Simulation results show that the relative error in the measurement results of our method can be decreased by half compared with the existing method when the total overheads of both methods are equal. We also confirmed that exchanging measurement results contributes more to the enhancement of measurement accuracy than performing measurements.

In the future, we plan to construct a measurement system that applies the proposed method and investigate its effectiveness in real environments.

Chapter 3

Measurement method for end-to-end available bandwidth

3.1 Introduction

The estimation of available bandwidth is crucial for many overlay network applications. For example, available bandwidth information allows the construction of an efficient overlay topology for video on demand [33] and peer-assisted streaming [34].

However, in general, measuring available bandwidth in overlay networks is expensive, not only because of the huge amount of pair-wise measurements but also because of the large traffic load of each measurement. In particular, for an overlay network that contains n overlay nodes, the number of pair-wise measurements is $O(n^2)$, which is unacceptable in large-scale overlay networks. Furthermore, the traffic load of each measurement of the available bandwidth is much larger than other metrics, such as latency or packet loss rate. This is because latency or packet loss rate can be measured by such lightweight tools as `ping`, and measuring the available bandwidth requires more complicated and costly mechanisms. For example, for Pathload [14, 35], which is one of the most accurate tools for measuring end-to-end available bandwidth, groups of packet streams called packet fleets are sent at various rates within a large range that contains the real value of available bandwidth. The traffic load of one Pathload measurement is very large and can reach 10 MB, based

on one study [36]. However, most existing solutions focus on decreasing the number of pair-wise measurements [7–10, 23] rather than reducing the traffic load of each measurement.

Another measurement issue in overlay networks is measurement conflict, which degrades measurement accuracy. This problem occurs when measurement tasks of overlapping paths are performed simultaneously. Previous studies have addressed this problem, and algorithms for avoiding concurrent measurements of overlapping paths have been proposed [11–13]. However, in these methods, although measurement conflicts can be avoided completely, measurement frequency is small, that leads to inaccurate measurement results [37]. Furthermore, the concurrent measurements of overlapping paths do not always cause conflict, depending on the mechanism of the measurement tools. For example, in the case of Pathload, because the interval between two consecutive packet streams is set to a value not smaller than one RTT, if the sending time of one packet stream is smaller than one RTT, the probability that a conflict occurs is smaller than that of non-conflict.

In Chapter 2, we proposed a distributed method for measuring additive metrics such as latency or packet loss rate, that can reduce measurement conflict and improve measurement accuracy. In this method, overlay nodes exchange route information to detect overlapping paths. Based on the overlapping state, the measurement frequency and timing of each path are determined to reduce the measurement conflict. Overlay nodes then exchange the measurement results of the overlapping parts to improve the measurement accuracy of these parts and improve the measurement accuracy of the whole path. However, we cannot apply this method when the metric is bandwidth-related information such as available bandwidth or throughput, because the measurement results of the overlapping parts cannot be obtained.

In this chapter, we propose a distributed method for measuring the available bandwidth that can also reduce the measurement conflict while decreasing the traffic load of each measurement. Even though we use the same mechanism as in Chapter 2 to detect the overlapping paths, we introduce a novel method that determines the measurement frequencies and timings to better measure the available bandwidth. To obtain accurate measurement results, we adopt some mechanisms similar to induced-congestion-based end-to-end available bandwidth tools such as Pathload or pathChirp [38] for measuring end-to-end available bandwidth. To reduce the measurement traffic load, the overlay nodes exchange the measurement results of the overlapping paths to calculate the parameters for

each measurement.

We make the following contributions in this chapter:

- We propose an algorithm that determines the measurement frequencies and timings of the overlapping paths to reduce measurement conflicts.
- We propose a method for calculating the parameters of each measurement to reduce the measurement traffic load.
- We evaluate our method and compare it with a previous method [12] by simulations with both generated and real Internet topologies.

From the simulation results, we reach the following conclusions:

- The relative errors in the measurement results of our method are approximately only 65% of those of the previous method [12].
- The measurement accuracy of our method is still better than that of the method in [12] when the total measurement traffic loads of both methods are equal.

The rest of this chapter is organized as follows. Section 3.2 describes related work. Definitions related to overlay networks are presented in Section 3.3. In Section 3.4, we explain our method that reduces measurement conflicts while decreasing the traffic load of each measurement. We evaluate our method in Section 3.5 and conclude this chapter in Section 3.6.

3.2 Related work

Measuring end-to-end available bandwidth has been extensively studied, and many measurement tools have been proposed so far. These tools can be mainly divided into two categories: Probe Gap Model (PGM) tools, e.g., IGI [39] and Spruce [36], and Probe Rate Model (PRM) tools, e.g., Pathload and pathChirp. PGM tools set an initial time gap between consecutive probing packets at the sender and observe the changes of the time gaps at the receiver to infer the cross traffic rate. They then subtract the cross traffic rate from the physical capacity of the bottleneck link to obtain

the available bandwidth. IGI finds an initial time gap that makes the queue of the bottleneck link full but not overflowed, while Spruce sets the initial time gap to the bottleneck link transmission time of a 1500 Byte packet. PRM tools rely on the idea of induced congestion: the delays of packets in a stream show an increasing trend when the probing rate exceeds the available bandwidth. These tools send packet streams at various rates and try to determine the rate at which the delays of the probe packets start increasing. This rate is an estimation of the available bandwidth. Pathload uses periodic packet streams, while pathChirp exponentially increases the probing rate within one stream. In general, PRM tools are more accurate but also more intrusive than PGM tools [40]. In this chapter, to obtain accurate measurement results, we use PRM tools to measure the end-to-end available bandwidth. However, because the default configurations of these tools make them intrusive, we propose methods for configuring the parameters of each measurement to reduce the measurement traffic load.

Many solutions have been proposed for effectively measuring the available bandwidth in overlay networks, and most focus on decreasing the number of pair-wise measurements from the $O(n^2)$ traffic load of full-mesh measurements [7–10, 23]. The method in [23] selects and measures only some paths that cover all the links of the paths between the overlay nodes and bases on these results to roughly estimate the results of remaining paths. The measurement traffic load is reduced to $O(n \log n)$, but the accuracy of the measurement results obtained by this technique is not high. BRoute [7] relies on two characteristics of overlay networks constructed over the Internet: (1) bottleneck links exist from both ends of the overlay path in roughly four hops or less, and (2) path overlappings often exist near both ends of the overlay path. Therefore, the available bandwidth of a segment near both ends of each overlay path can be used to get the available bandwidth of the entire path, which greatly reduces the measurement traffic load. However, this technique requires BGP routing information in advance to infer the AS-level paths between end hosts. Currently proposed solutions [8–10] rely on the observation that the measurement of available bandwidth can be approximately embedded to metric spaces, and thus it can be estimated using the concept of distance in metric space. In [8], the available bandwidth between two arbitrary nodes is calculated based on the measurement results of the incoming and outgoing paths between these nodes and predetermined hosts called landmarks. Sequoia [9] embeds nodes in the leaves of a weighted tree

and uses the distances in it to estimate the available bandwidth. Another method, which is a decentralized version of Sequoia, reduces the number of measurements [10]. Although these methods show better results than previous coordinate-based solutions [2, 3], their measurement accuracy remains insufficient because embedding the measurement of available bandwidth to a metric space is only approximately justified by some real Internet datasets. We take a contrasting approach to the existing solutions [7–10, 23] and focus on decreasing the traffic load of each measurement. Our approach not only reduces the total measurement traffic load but also helps mitigate the measurement conflict.

In general, to improve measurement accuracy, we must obtain as many accurate measurement results as possible. Since the accuracy of each measurement can be seriously affected by the conflicts between the concurrent measurements of overlapping paths [11], we should reduce the measurement conflict while maintaining high measurement frequency. However, existing solutions [11–13] focus on avoiding the measurement conflict by sacrificing measurement frequency. The main idea of these studies is that they use heuristic algorithms from graph theory to schedule the measurement timings of the paths so that the overlapping paths are measured at different timings. Although measurement conflicts can be completely avoided, the measurement frequencies are limited, and so the measurement accuracy is not high. Furthermore, the concurrent measurements of overlapping paths do not always cause measurement conflict; the probability that conflicts occur depends on the mechanism of the measurement tool and may be small in some cases. In particular, for such tools as Pathload or pathChirp [38], to obtain accurate results, many packet streams are redundantly sent, and the interval between packets is carefully calculated. Thus the number of packets that experience conflict may be so small that the measurement results are not affected. Our method does not completely avoid concurrent measurements of overlapping paths, as in previous solutions [11–13]; instead it reduces them while maintaining high measurement frequency to improve measurement accuracy.

In Chapter 2, we proposed a method for measuring such additive metrics as latency or packet loss rate that improve the measurement accuracy utilizing measurement exchanges between overlay nodes. The method contains three parts:

- an algorithm for detecting overlapping paths,
- an algorithm for determining the measurement timings that reduce measurement conflicts while maintaining high measurement frequency,
- and an algorithm for exchanging the measurement results to improve the measurement accuracy.

In this chapter, we only use the same algorithm for detecting overlapping paths, but propose two new algorithms for determining the measurement timings and exchanging the measurement results. This is because the two algorithms in Chapter 2 can only be applied for additive metrics; they cannot be applied for such bandwidth-related information as available bandwidth or throughput for the following two reasons.

First, in Chapter 2, the source nodes of the overlapping paths exchange the measurement results of the overlapping parts to improve their measurement accuracy and consequently improve the measurement accuracy of the whole path. However, when the metric is bandwidth-related information, we cannot obtain the measurement results of the overlapping parts. In this chapter, we propose a method for exchanging the measurement results of the whole path and the related information. Furthermore, because the measurement results of overlapping paths are equal only when these paths have the same bottleneck links, we cannot use such data to directly improve the measurement accuracy. We instead use them to configure parameters for each measurement to reduce the measurement time and traffic load. This not only reduces the total measurement traffic load but also mitigates measurement conflicts, indirectly improving the measurement accuracy.

Second, in Chapter 2, because the measurement results of one path can be used directly to improve the measurement accuracy of its overlapping paths, the number of measurements and measurement timings of each path can be roughly determined, as long as the total number of the measurements of these overlapping paths is maintained. However, in this chapter, as explained above, because the measurement results of one path cannot be used directly for improving the measurement accuracy of its overlapping paths, the number of measurements and the measurement timings of each path must be determined more strictly. We thus propose a novel method for this end.

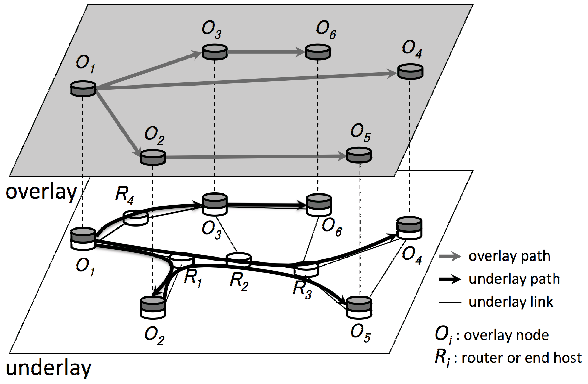


Figure 3.1: Example of overlay network and path overlapping

3.3 Network model and definitions

In this chapter, we define an overlay network as a logical network constructed on an under-layer IP network. In overlay networks, overlay nodes are often installed on end hosts as an application program. In this case, both routing and traffic control at the overlay level are conducted at the end hosts, and such controls cannot be activated inside the network. On the other hand, the overlay routing inside the network becomes possible by installing overlay nodes on the routers in the network. This installation can be done in the networks that support configurations at the application level in the routers. If the network supports such techniques as network virtualization [20] and software defined networks [21], which enable the settings of all network components at some devices called controllers, this installation can be further simplified. In this chapter, to realize efficient routing control by overlay networks, we consider an overlay network in which the overlay nodes are deployed on both routers and end hosts.

We call an under-layer IP network an *underlay network* and consider an underlay network with m end hosts or routers, denoted by R_i $(i = 1, ..., m)$. For simplicity, we call an end host or router an *underlay node*. Suppose that n $(n \leq m)$ overlay nodes are deployed on n different underlay nodes. We denote the overlay nodes as O_i $(i = 1, ..., n)$. Density σ of the overlay nodes is defined as the ratio of the number of overlay nodes to the number of underlay nodes, i.e., $\sigma = n/m$. Figure 3.1 shows an example of an underlay network and the overlay network constructed on it.

The gray arrows show the overlay paths, and the black arrows illustrate the underlay paths of the corresponding overlay paths. We assume the shortest path algorithm for routing in the underlay network and denote the underlay path between underlay nodes R_i and R_j as R_iR_j. For path R_iR_j, R_i is the *source node*, and R_j is the *destination node* of the path. If different paths R_iR_j and R_sR_t share at least one link, R_iR_j and R_sR_t overlap each other, or R_iR_j (R_sR_t) is an *overlapping path* of R_sR_t (R_iR_j). We define a *route* from R_i to R_j as a sequence of underlay nodes that construct an underlay path from R_i to R_j.

As in Chapter 2, we classify the overlapping states into the following three types:

- Complete overlapping: One path completely includes another path. The path that includes the other is called the *longer path*, and the other path is called the *shorter path*.
- Half overlapping: Two paths share a route from the source node to a router that is not an overlay node.
- Partial overlapping: Two paths share a route that does not include the source node.

For example, in Fig. 3.1, path O_1O_3 is a complete overlapping path of O_1O_6. Paths O_1O_2 and O_1O_4 have a half overlapping relation. Path O_1O_4 is a partial overlapping path of O_2O_5.

3.4 Proposed method

3.4.1 Overview

Our solution is built in a completely distributed fashion, in which each overlay node measures the paths starting from itself, based on the information it exchanges with neighboring overlay nodes. The measurement procedure of each overlay node includes the following three phases:

- Detection phase of path overlappings

  The overlay nodes detect path overlappings using a previously described method in Chapter 2.

- Calculation phase of measurement timings

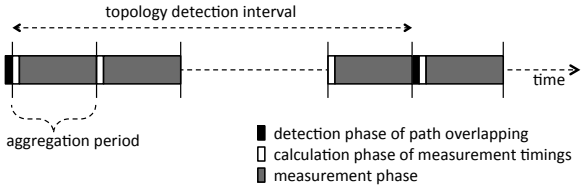


Figure 3.2: Measurement procedure

The measurement frequencies and timings in a predetermined duration are calculated based on the path-overlapping status.

- Measurement phase

  At each measurement timing in the predetermined duration, the overlay node calculates the parameters of the end-to-end measurement using the previous measurement results received from other nodes. The measurement is performed or omitted based on the calculation results. The overlay node then sends the measurement results and related information to the neighboring overlay nodes.

Figure 3.2 illustrates the relationships among the three phases. We call a duration that contains one calculation phase of measurement timings and one measurement phase an *aggregation period*. Because the change in the routing information of the underlay network is generally less frequent than the change in the measurement results, the interval between two successive phases of path overlapping detection is larger than an aggregation period. We call this a *topology detection interval*. In general cases, the lengths of the detection phase of the path overlappings and the calculation phase of the measurement timings are much smaller than that of the measurement phase. This is because the overheads of the exchanging path information and the calculating measurement timings are very small compared to that of the measurements [37]. Therefore, we ignore the time of the detection phase of the path overlappings and the calculation phase of the measurement timings and only consider the time of the measurement phase when calculating the measurement frequencies.

3.4.2 Detection phase of path overlappings

We use a previous method proposed in Chapter 2 for detecting complete, half, and partial overlapping paths on overlay networks. In particular, arbitrary overlay node O_i can detect complete and half overlapping paths of path O_iO_j by issuing `traceroute` to all the other nodes. To detect the partial overlapping paths of O_iO_j, O_i first utilizes the overlapping status of the half overlapping paths to find the candidates of partial overlapping paths and then exchanges the routing information with the source nodes of the candidates to determine their overlapping states. For example, in Fig. 3.1, we infer that path O_2O_5 is a partial overlapping path of O_1O_4, because the length of the overlapping part of O_1O_4 and O_1O_2 is smaller than the length of the overlapping part of O_1O_4 and O_1O_5. O_1 then exchanges routing information with O_2 to confirm whether O_2O_5 is actually a partial overlapping path of O_1O_4. Our simulation results show that our method can detect about 90% of the partial overlapping paths with relatively small overhead [37].

3.4.3 Calculation phase of measurement timings

We propose a method for calculating the measurement timings of the paths that can reduce measurement conflicts while maintaining high frequencies to improve measurement accuracy. Our method utilizes the overlapping status of the paths.

For complete overlapping paths, to avoid measurement conflict, we only measure the shorter path; the longer path is not directly measured, and its measurement result is estimated based on the measurement results of the shorter paths included in it [37].

We explain the method for half and partial overlapping paths as follows. Consider path O_iO_j that has half and partial overlapping paths (Fig. 3.3). We denote the number of half overlapping paths of O_iO_j as $(G_{i,j} - 1)$ $(G_{i,j} \geq 1)$. For simplicity, we rewrite $G_{i,j}$ as G. We denote path O_iO_j as path 1, and each of its half overlapping paths as path p $(2 \leq p \leq G)$. We also assume that path p $(1 \leq p \leq G)$ has $(K_p - 1)$ partial overlapping paths $(K_p \geq 1)$.

Overlay node O_i can avoid the measurement conflicts among half overlapping paths 1, 2, ... and G simply by measuring them sequentially. On the other hand, because the source nodes of the partial overlapping paths of path p are different, measurement conflicts between them cannot be

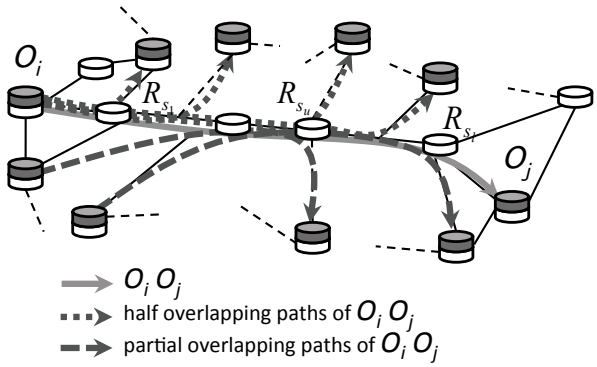


Figure 3.3: Example for explaining the proposed measurement method

completely avoided. Therefore, we propose a technique that combines a sequential measurement for half overlapping paths and a random measurement for partial overlapping paths. We set the time required to measure a single path to a predetermined parameter τ. We divide each aggregation period into T ($T \geq 1$) *measurement time slots*, whose length is τ. We denote the measurement times of path p at an aggregation period as h_p ($h_p \leq T$) and calculate h_p as follows.

We introduce β_p as a value that reflects the variability of the measurement results of path p at an aggregation period. Note that the method to determine β_p is beyond the scope of this thesis. For example, β_p can be calculated based on the statistics of the measurement results or using a previous method [24]. We set measurement times h_p proportional to β_p for all paths, i.e., $h_1/\beta_1 = h_2/\beta_2 = ... = h_G/\beta_G$. To avoid measurement conflicts between half overlapping paths, the sum of their measurement times should be equal to or less than T: $\sum_{p=1}^{G} h_p \leq T$. We have $h_p \leq T\beta_p/(\sum_{s=1}^{G} \beta_s)$.

To reduce the measurement conflicts between path p and its $(K_p - 1)$ partial overlapping paths, we set the measurement times of path p to a value equal to or less than T/K_p, i.e., $h_p \leq T/K_p$. In addition, we keep the measurement times as large as possible to obtain as many measurement results as possible. Therefore, we set $h_p = min\{T\beta_p/(\sum_{s=1}^{G} \beta_s), T/K_p\}$.

However, the measurement times of path p should be large enough to avoid serious degradation of the measurement accuracy. To avoid this unexpected situation, we set the minimum of the measurement times to parameter D ($D \geq 2$) and adjust h_p so that $h_p \geq D$, $1 \leq p \leq G$, and $\sum_{p=1}^{G} h_p \leq T$. In detail, for paths p where $h_p < D$, we set $h_p = D$. As a result, $\sum_{p=1}^{G} h_p$ increases and

may exceed T. In that case, for paths p where $h_p > D$, we reduce h_p one by one until $\sum_{p=1}^{G} h_p = T$ while keeping $h_p \geq D$. Note that if we set the aggregation period large enough, that is, when we set $T \geq D(n-1)$, because $G \leq n-1$, the above adjustment of h_p can be satisfied.

Algorithm 3 Method for allocating measurement timings

```
1: function AllocMeasTime()
2: for p = 1 to G do
3:     Set c (c ≤ T) as the number of slots that have not been allocated to any path
4:     Divide these c slots into h_p groups, so that each group contains c/h_p continuous slots
5:     Randomly choose one slot from each group and allocate it for path p
6: end for
7: end function
```

Next, we explain our method for randomly deciding the measurement timings of path p so that the measurement times of path p become h_p. The main idea of our method is that we divide T measurement time slots of an aggregation period into h_p groups and randomly choose one slot from each group to allocate for path p. Algorithm 3 shows the details of our method.

3.4.4 Measurement phase

In this section, we explain our method that sets the parameters for each end-to-end measurement to reduce the measurement traffic load. We first present the method in the case that end-to-end measurement tool is Pathload [14, 35]. We then briefly explain the methods for other tools such as pathChirp [38].

Calculating parameters for available bandwidth measurement

To obtain accurate measurement results, we adopt a mechanism similar to Pathload for measuring the end-to-end available bandwidth. However, the default settings of parameters in each Pathload measurement makes its traffic load very large. Therefore, we propose a statistical method for calculating these parameters to reduce the measurement traffic load.

We first describe the mechanism of Pathload and explain why its measurement traffic load is large. Pathload relies on the fact that the one-way delays of a periodic packet stream show an increasing trend when the stream rate exceeds the available bandwidth. It first sets a large range

(R_{min}, R_{max}), and uses a binary search algorithm to find the value of the available bandwidth inside this range. In detail, at each iteration of the measurement, the source node sends a string of packet streams called a *packet fleet* at the rate $R^* = (R_{min} + R_{max})/2$ and checks whether there is an increasing trend in the one-way delays to judge if the real value of the available bandwidth is larger or smaller than the rate. If it found that the real value of the available bandwidth is larger than the rate, then it updates R_{min} to R^*, while in other case, it updates R_{max} to R^*, and repeats the search procedure. It stops when the width of the search range (R_{min}, R_{max}) is smaller than a predetermined threshold ω, and reports (R_{min}, R_{max}) as the measurement result. It is obvious that the traffic load of each measurement depends on the width of the initial search range. The initial value of R_{min} is set to 0, and that of R_{max} is set to a large value, for example, the capacity of the path; thus the measurement traffic load is very large [36].

To reduce the traffic load of each measurement, in our method, overlay nodes exchange measurement results of overlapping paths and related information to calculate a search range (R_{min}, R_{max}) that is narrow and near the real value of the available bandwidth. We rely on the observation that when the tight links of two overlapping paths belong to their overlapping part, the measurement result of one path can be used as the measurement result of the other.

We explain our proposed method by describing the detailed behavior for path O_iO_j. We first assume that path O_iO_j has K partial overlapping paths ($K \geq 1$), denoted as $O_{u_s}O_{v_s}$ ($1 \leq s \leq K$). O_i receives the following information from O_{u_s} ($1 \leq s \leq K$).

1. Measurement result

2. Probability that a tight link of $O_{u_s}O_{v_s}$ belongs to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$.

We denote the probability that the tight link of $O_{u_s}O_{v_s}$ belongs to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$ as $\Phi_{O_{u_s}O_{v_s},O_iO_j}$ and calculate it using a previous method [24] as follows:

$$\Phi_{O_{u_s}O_{v_s},O_iO_j} = \frac{Latency(Overlap(O_iO_j, O_{u_s}O_{v_s}))}{Latency(O_{u_s}O_{v_s})}.$$

Here, $Overlap(O_iO_j, O_{u_s}O_{v_s})$ is the overlapping part of paths O_iO_j and $O_{u_s}O_{v_s}$.

After receiving the above data, O_i also estimates $\Phi_{O_iO_j,O_{u_s}O_{v_s}}$, which is the probability that

the tight link of O_iO_j belongs to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$. Then it calculates $\alpha_s = \Phi_{O_iO_j,O_{u_s}O_{v_s}}\Phi_{O_{u_s}O_{v_s},O_iO_j}$, which is the probability that the tight links of O_iO_j and $O_{u_s}O_{v_s}$ belong to the overlapping part of O_iO_j and $O_{u_s}O_{v_s}$. That means that α_s is the probability that the measurement results of O_iO_j and $O_{u_s}O_{v_s}$ are equal.

O_i stores its measurement results and the information received from other nodes. It uses the stored data to calculate R_{min} and R_{max} and discards them when it decides that these data are no longer useful for this calculation.

We assume that at measurement timing t^*, O_i stored G measurement results of O_iO_j and its half and partial overlapping paths, denoted as $(A_L^1, A_U^1), (A_L^2, A_U^2), ..., (A_L^G, A_U^G)$. We also denote the probabilities that these results equal the measurement results of O_iO_j as $\alpha_1, \alpha_2, ..., \alpha_G$, respectively. Note that α_s $(1 \leq s \leq G)$ corresponding to the measurement result of O_iO_j is set to 1.

We calculate the lower bound of the 95% confidence interval of A_L^s $(1 \leq s \leq G)$, denoted as S_L^*, and the upper bound of the 95% confidence interval of A_U^s $(1 \leq s \leq G)$, denoted as S_U^*, as follows:

$$S_L^* = \bar{A}_L - 1.96\sqrt{\frac{V_L}{G}} \quad , \quad S_U^* = \bar{A}_U + 1.96\sqrt{\frac{V_U}{G}}. \tag{3.1}$$

Here, $\bar{A}_L$, V_L, $\bar{A}_U$, and V_U are the weighted means and variances, calculated as follows:

$$\begin{aligned} \bar{A}_L &= \sum_{s=1}^{G} \beta_s A_L^s \quad , \quad V_L = \sum_{s=1}^{G} \beta_s {A_L^s}^2 - \bar{A}_L^2 \\ \bar{A}_U &= \sum_{s=1}^{G} \beta_s A_U^s \quad , \quad V_U = \sum_{s=1}^{G} \beta_s {A_U^s}^2 - \bar{A}_U^2, \end{aligned} \tag{3.2}$$

where $\beta_s = \alpha_s / \sum_{w=1}^{G} \alpha_w$ $(1 \leq s \leq G)$ is the weight of result (A_L^s, A_U^s).

When the stored data of O_i contain some measurement results of O_iO_j, we can infer that range (S_L^*, S_U^*) is near the real value of the available bandwidth and set $R_{min} = S_L^*$ and $R_{max} = S_U^*$. However, when they only contain measurement results of overlapping paths of O_iO_j, it is possible that these measurement results are much different from those of O_iO_j. As a result, the probability that range (S_L^*, S_U^*) is near the real value of the available bandwidth of O_iO_j is small and we can

Algorithm 4 Measurement algorithm for path O_iO_j

function MeasureOnePath()
//initialization
if stored data of O_i contain measurement results of O_iO_j **then**
  $R_{min} \leftarrow S_L^*$
  $R_{max} \leftarrow S_U^*$
else
  $R_{min} \leftarrow 0$
  $R_{max} \leftarrow C_{O_iO_j}^0$
end if
$ub\_found \leftarrow 0$
$lb\_found \leftarrow 0$
$meas\_time \leftarrow \tau$

//find a range (R_{min}, R_{max}) that includes available bandwidth
while $(ub\_found = 0 \,||\, lb\_found = 0) \,\&\&\, (R_{min} > 0 \,||\, R_{max} < C_{O_iO_j}^0) \,\&\&\, meas\_time > 0$ **do**
  //test if R_{max} is an upper bound of the available bandwidth
  if $ub\_found = 0$ **then**
    $TestUB(R_{min}, R_{max}, lb\_found, ub\_found, meas\_time)$
  end if
  //test if R_{min} is a lower bound of the available bandwidth
  if $lb\_found = 0 \,\&\&\, meas\_time > 0$ **then**
    $TestLB(R_{min}, R_{max}, lb\_found, ub\_found, meas\_time)$
  end if
end while

//measure available bandwidth between the range (R_{min}, R_{max})
if $R_{max} - R_{min} > \omega \,\&\&\, meas\_time > 0$ **then**
  $RuntimeLimitedPathload(R_{min}, R_{max}, meas\_time)$
end if
return R_{min}, R_{max}
end function

Algorithm 5 Test of upper bound of search range

function TestUB($R_{min}, R_{max}, lb\_found, ub\_found, meas\_time$)
Set current time to $start\_time$
Send a packet fleet with rate R_{max}
if increasing trend **then**
    $ub\_found \leftarrow 1$
else
    $R_{min} \leftarrow R_{max}$
    $lb\_found \leftarrow 1$
    $R_{max} \leftarrow min(R_{max} + (S^*_U - S^*_L)/2, C^0_{O_iO_j})$
end if
Set current time to end_time
$meas\_time \leftarrow meas\_time - (end\_time - start\_time)$
return $R_{min}, R_{max}, lb\_found, ub\_found, meas\_time$
end function

Algorithm 6 Test of lower bound of search range

function TestLB($R_{min}, R_{max}, lb\_found, ub\_found, meas\_time$)
Set current time to $start\_time$
Send a packet fleet with rate R_{min}
if non increasing trend **then**
    $lb\_found \leftarrow 1$
else
    $R_{max} \leftarrow R_{min}$
    $ub\_found \leftarrow 1$
    $R_{min} \leftarrow max(R_{min} - (S^*_U - S^*_L)/2, 0)$
end if
Set current time to end_time
$meas\_time \leftarrow meas\_time - (end\_time - start\_time)$
return $R_{min}, R_{max}, lb\_found, ub\_found, meas\_time$
end function

not use it as the search range. In such cases, we set $R_{min} = 0$ and $R_{max} = C^0_{O_iO_j}$, where $C^0_{O_iO_j}$ is the capacity of the IP link of path O_iO_j that connects to O_i.

Performing measurement

Note that when the stored data of O_i contain some measurement results of O_iO_j, the width of range (S^*_L, S^*_U) reflects the variability of the available bandwidth of O_iO_j at the time near t^*. Therefore, O_i decides whether it will perform or omit the measurement of O_iO_j at measurement timing t^* based on this value as follows:

- If $S^*_U - S^*_L < \epsilon$, where ϵ is a predetermined parameter, we conclude that the variability of the available bandwidth of O_iO_j at the time near t^* is small and infer that this trend does not change at timing t^*. This means that (S^*_L, S^*_U) can be approximately considered as the measurement result at t^*. Therefore, O_i omits a measurement at timing t^* and uses (S^*_L, S^*_U) as the measurement result. This omission helps to decrease the measurement traffic load of O_i and reduce the conflicts between the measurements of O_iO_j and its partial overlapping paths.

- If $S^*_U - S^*_L \geq \epsilon$, O_i decides to perform a measurement at timing t^*.

Next, we explain how O_i performs a measurement when $S^*_U - S^*_L \geq \epsilon$. Although we infer that the real value of the available bandwidth is near range (S^*_L, S^*_U), we are not sure whether the real value of available bandwidth is inside this range. Therefore, O_i first sends probing packet streams with rates S^*_L and S^*_U to determine if the real value of the available bandwidth is between S^*_L and S^*_U, based on the status of the increasing trend in the one-way delays. If the real value of the available bandwidth does not exist between S^*_L and S^*_U, we infer that it has changed greatly and discard the stored measurement results because these data have become unreliable. We also infer that its real value exists outside but near range (S^*_L, S^*_U). We then set a new search range at a neighboring range of (S^*_L, S^*_U) and check whether the real value of the available bandwidth belongs to this new range. This procedure is repeated until we find a search range that includes the real value of the available bandwidth. We then apply a similar algorithm with Pathload to search for the value of the available bandwidth.

In the case of Pathload, the search procedure stops when the width of the search range is smaller than the threshold ω. In our proposed method, we add another termination condition for the search procedure; our search procedure also stops when the measurement time exceeds τ.

Algorithms 4, 5, and 6 show the details of our method. Procedure *RuntimeLimitedPathload* is a search procedure that resembles Pathload with limited search time.

At the end of an aggregation period, the measurement result of O_iO_j at that aggregation period is calculated as follows. Assume that during that aggregation period, O_i obtained F measurement results of O_iO_j, denoted as $(A_L^1, A_U^1), (A_L^2, A_U^2), ..., (A_L^F, A_U^F)$. The measurement result of O_iO_j at that aggregation period is calculated by Eq. (3.3):

$$A_{meas} = \frac{1}{F}\sum_{s=1}^{F}\frac{A_L^s + A_U^s}{2}. \tag{3.3}$$

Sending measurement results and related information

If the measurement is performed, O_i sends the result and probabilities $\Phi_{O_iO_j,O_{u_s}O_{v_s}}$ to nodes O_{u_s} $(1 \leq s \leq K)$. On the other hand, if the measurement is omitted, search range (S_L^*, S_U^*) is not sent to other nodes, although it is used as a measurement result of O_iO_j. This is because (S_L^*, S_U^*) is calculated based on the results of the measurements before timing t^*, that have already been sent to other nodes. Furthermore, since (S_L^*, S_U^*) is not the result of an actual measurement, it might not be near the real value of the available bandwidth of O_iO_j. Therefore, not sending it to nodes O_{u_s} $(1 \leq s \leq K)$ helps to avoid degradation of the measurement accuracy of $O_{u_s}O_{v_s}$.

Methods for other end-to-end measurement tools

Our proposed method that sets parameters for each end-to-end measurement can be applied for other induced-congestion-based measurement tools such as pathChirp [38]. Similar to Pathload, pathChirp also uses several packet streams, called *chirps*, to probe the available bandwidth. However, the probing rates within a chirp increase exponentially from a lower rate L to an upper rate $U = L\gamma^{(N-1)}$, where N is the number of packets in the chirp, and γ is the *spread factor* ($\gamma > 1$). We use the same approach with the case of Pathload to set the values of L and U. We only make

the following two changes for pathChirp.

First, in contrast to Pathload, the result of each pathChirp measurement is a point. Suppose that O_i stored the G measurement results of O_iO_j and its half and partial overlapping paths, denoted as $A^1, A^2, ..., A^G$ at measurement timing t^*, then values S_L^* and S_U^* in Eq. (3.1) are modified as follows:

$$S_L^* = \bar{A} - 1.96\sqrt{\frac{V}{G}} \quad , \quad S_U^* = \bar{A} + 1.96\sqrt{\frac{V}{G}}. \tag{3.4}$$

Here, $\bar{A}$ and V are the weighted means and variances, calculated as follows:

$$\bar{A} = \sum_{s=1}^{G} \beta_s A^s \quad , \quad V = \sum_{s=1}^{G} \beta_s A^{s2} - \bar{A}^2, \tag{3.5}$$

where $\beta_s = \alpha_s / \sum_{w=1}^{G} \alpha_w$ $(1 \leq s \leq G)$ is the weight of result A^s. When the stored data of O_i contain some measurement results of O_iO_j, we can infer that range (S_L^*, S_U^*) is near the real value of the available bandwidth and set $L = S_L^*$ and $U = S_U^*$.

Second, although we also infer that search range (S_L^*, S_U^*) is near the real value of the available bandwidth, we do not check whether its real value is inside this range, as in the case of Pathload. This is because pathChirp ' s algorithm does not include sending packet streams at a constant rate to check whether the real value of the available bandwidth is smaller or larger than that rate.

3.5 Performance evaluation

In this section, we evaluate the performance of our proposed method. We focus on evaluating how our proposed method of local information exchange relatively improves measurement accuracy rather than absolutely evaluating the measurement accuracy. Therefore, we only use evaluations based on simulations. Evaluation based on experiments in real networks is our future work. We explain the evaluation method in Subsection 3.5.1 and present our evaluation results and discussions in Subsection 3.5.2.

3.5.1 Evaluation method

We compared our proposed method with an existing method [12], which we briefly explain, and make some assumptions for comparison in Subsection3.5.1. Because measurement conflict is not avoided completely in our method, we describe a statistical model for simulating the effect of measurement conflict on the results of the end-to-end measurement in Subsection 3.5.1. We then explain the evaluation metrics and the simulation settings in Subsects. 3.5.1 and 3.5.1. Throughout the simulation, we assume that Pathload is used for the end-to-end measurement of the available bandwidth. However, we expect the same trend in the evaluation results in the cases of other measurement tools.

Existing method [12]

In [12], the authors proposed heuristic algorithms from graph theory to schedule different timings for the measurement tasks of overlapping paths. Although measurement conflicts are completely avoided, the measurement frequency is greatly limited, as we show below in Subsection 3.5.2. Two algorithms have been proposed for uniform and non-uniform measurement tasks. Uniform measurement tasks have the same running time and period, but non-uniform measurement tasks have different running times or periods. As explained in Subsection 3.4.3, the running time of all the measurement tasks is set to the same value τ. To keep the measurement frequency as large as possible, we set the periods of all the measurement task to the running time. That means that all the measurement tasks are uniform, and we apply the corresponding algorithm to obtain the measurement timings for each task. Because the overlay nodes do not exchange information with each other, the search range in each end-to-end measurement cannot be estimated, as in our proposed method. We therefore set the search range for path O_iO_j to $(0, C_{O_iO_j})$, where $C_{O_iO_j}$ is the capacity of the first IP link of path O_iO_j.

Simulating the effect of measurement conflict on the end-to-end measurement result

A measurement conflict may occur when the measurements of overlapping paths are performed concurrently. When it occurs, we cannot determine exactly the effect it causes on the measurement results. However, we can statistically simulate this effect based on the mechanism of end-to-end

measurement as follows.

As mentioned above, we adopt a mechanism similar to Pathload for measuring the end-to-end available bandwidth. In particular, at each iteration of the measurement, the source node sends a string of packet streams (a packet fleet) with a predetermined rate and checks whether there is an increasing trend in the one-way delays to judge if the real value of the available bandwidth is larger or smaller than the rate. This judgement may become incorrect if the durations of sending packet streams overlap with those of overlapping paths.

We assume that at measurement timing t^* of path O_iO_j, H ($H \geq 1$) overlapping paths of O_iO_j also perform measurements. For simplicity, we denote path O_iO_j as path 1 and each of these H overlapping paths as path p ($2 \leq p \leq H+1$). For path p ($1 \leq p \leq H+1$), the duration for sending one packet stream is denoted as θ_p, and the interval between two successive packet streams in each packet fleet is set to ψ_p. $\pi_p = \theta_p + \psi_p$ is the period of one packet stream of path p. Set $\phi_p = \theta_p/\pi_p = \theta_p/(\theta_p + \psi_p)$, $0 < \phi_p < 1$. In general, θ_p is much smaller than ψ_p [35], and thus $\phi_p \ll 1$.

We calculate the probability that the duration of sending packet streams of path 1 overlaps with the durations of sending packet streams of any path p ($2 \leq p \leq H+1$) as follows. During one period π_1 of one packet stream of path 1, the duration of sending packet streams of path p ($2 \leq p \leq H+1$) becomes $t_p = \phi_p\pi_1$. The probability that the durations of sending packet streams of paths 1 and p overlap with each other during one period π_1 is $\gamma = \theta_1/(\pi_1 - t_p) = \phi_1/(1-\phi_p)$. Since $\phi_p \ll 1$, we have $\gamma \approx \phi_1$. Therefore, the probability that the duration of sending packet streams of path 1 overlaps with the durations of sending packet streams of any path p ($2 \leq p \leq H+1$) becomes $1-(1-\phi_1)^H$.

We then simulate the effect of the measurement conflict on the result of the end-to-end measurement of path O_iO_j at measurement timing t^* as follows. We randomly decide that the durations of sending packet streams of path O_iO_j overlaps with the durations of sending packet streams of overlapping paths of O_iO_j with probability $1-(1-\phi_1)^H$. In this case, we randomly decide whether the sending rate is larger or smaller than the real value of the available bandwidth.

For simplicity, we use the default settings of Pathload [35] to set $\phi_p = 0.1$ for all paths.

Evaluation metrics

We compare our proposed method and the method in [12] with the following metrics:

- Metrics related to measurement accuracy

  We use the relative error of the measurement results as a metric to evaluate the measurement accuracy of the methods. As explained in Section 3.2, the measurement accuracy is largely decided by the measurement frequency and conflict. Therefore, we evaluate the following metrics:

  - *Relative error*

    The relative error is calculated by:

    $$e = \frac{|A_{meas} - \bar{A}|}{\bar{A}}, \tag{3.6}$$

    where A_{meas} is the average of the measurement results in an aggregation period, defined by Eq. (3.3), and $\bar{A}$ is the average of the real value of the available bandwidth in that aggregation period.

  - *Measurement frequency* and *number of measurements*

    The measurement frequency of a path at an aggregation period is calculated by:

    $$f = \frac{h\tau}{\Delta}, \tag{3.7}$$

    where h is the number of measurements in that aggregation period, τ is the duration of one measurement, and Δ is the duration of that aggregation period.

  - *Measurement conflict rate*

    The measurement conflict rate is defined as the ratio of the number of measurements that experience conflicts to the number of measurements, at one aggregation period.

- Metrics related to measurement traffic load

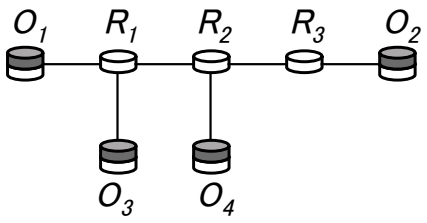


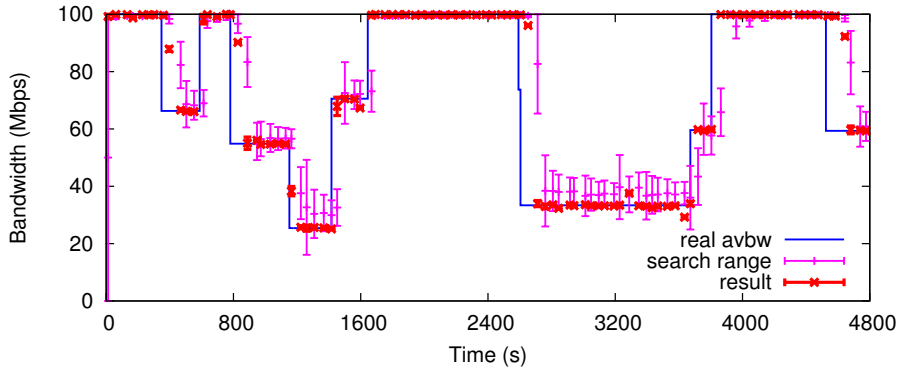
Figure 3.4: Small network topology

We use the average number of packet fleets traversing one link to evaluate the measurement traffic load.

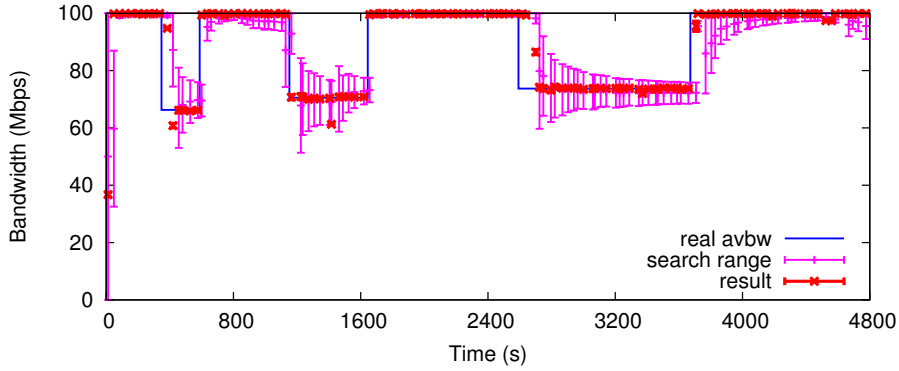
Simulation settings

The purposes of our simulations are to confirm that the proposed method works properly as designed and to compare its performance and that of the method in [12]. For the former purpose, we applied our method to the small network topology shown in Fig. 3.4 and observed its detailed behavior. For the latter purpose, we used three types of large network topologies: the AT&T topology [28], generated topologies based on the Barabasi-Albert (BA) [29], and the Waxman models [30]. We generated ten topologies for each model using the BRITE topology generator [31]. All topologies have 523 nodes and 1304 links. We set the density of the overlay nodes to 0.2 and randomly chose them from 523 nodes. For averaging the results, the choice of the overlay nodes was taken 100 times for the AT&T topology and ten times for each topology of the BA and Waxman models. For simplicity, we assume that the capacity of all IP links in the network is C and set $C = 100$ [Mbps].

We made the following assumptions about the temporal changes in the traffic amount between the overlay nodes. Assume that cross traffic occurs at fraction α ($0 < \alpha \leq 1$) of the paths. For the small network topology, α was set to 0.2, and in the large network topologies, it was set to 0.02. For path O_iO_j where cross traffic occurs, denote its IP links as $l_1, l_2, ..., l_r$. Assume that among the paths where cross traffic occurs, the number of paths that share the link l_t ($1 \leq t \leq r$) is b_t. Set $b_{max} = \max\{b_1, b_2, ..., b_r\}$. Furthermore, set $s_{max} = 0.9C/b_{max}$, $s_{min} = 0.5s_{max}$. The rate of cross traffic of O_iO_j was then randomly chosen in range $[s_{min}, s_{max}]$. Furthermore, the intervals where traffic occurs and does not occur were randomly chosen in range $[120\text{s}, 1200\text{s}]$.



(a) Path O_1O_4



(b) Path O_3O_1

Figure 3.5: Measurement result in small network topologies

For the parameters related to end-to-end measurement, because we adopt a method similar to Pathload, we set the parameters based on the suggestions of the authors of Pathload [35]. In particular, we set $\tau = 12$ [s] and $\omega = 400$ [Kbps]. The parameter for omitting measurement ϵ was set to 2ω. Minimum value D of the measurement times of one path at an aggregation period was set to 2. For the small network topology, we set the measurement duration to 400τ. On the other hand, in the large network topologies, the measurement duration contained ten aggregation periods, and each aggregation period was set to 1200τ.

Our simulation program was written in C language and run on commodity Linux machines with default settings.

Table 3.1: Distribution of relative errors (AT&T topology)

Method \ Relative error	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	56.600%	32.184%	9.576%	1.432%
Proposed method	41.999%	18.087%	3.260%	0.194%

Table 3.2: Distribution of relative errors (BA topology)

Method \ Relative error	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	50.994%	29.480%	9.542%	1.153%
Proposed method	35.472%	14.161%	2.546%	0.105%

3.5.2 Evaluation results and discussions

Evaluation results in small network topologies

Figure 3.5 shows the measurement results of paths O_1O_4 and O_3O_1 of the network in Fig. 3.4. The measurement results of other paths exhibited similar trends, and are thus omitted in the interest of saving space. The blue lines show the real values of available bandwidth, the pink bars show the search ranges, and the red bars show the measurement results. Because our measurement accuracy depends on the search range, we consider the variation of the search range to evaluate the effectiveness of our method. As shown in Fig. 3.5, the search range varies based on the real value of the available bandwidth and tends to approach it. When its real value changes greatly, the width of the search range is large at first, but it gradually becomes small, and the search range quickly approaches the real value of the available bandwidth. These results show that our proposed method for calculating search range is efficient for measuring available bandwidth.

Evaluation results in large network topologies

Measurement accuracy Tables 3.1, 3.2 and 3.3 show the distribution of the relative errors in the measurement results in the AT&T, BA, and Waxman topologies, respectively. In particular, it shows

Table 3.3: Distribution of relative errors (Waxman topology)

Method \ Relative error	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	33.411%	15.373%	3.418%	0.167%
Proposed method	26.841%	9.492%	1.385%	0.024%

Table 3.4: Average relative errors

Method \ Topology	AT&T	BA	Waxman
Existing method	0.088	0.081	0.049
Proposed method	0.058	0.049	0.039

the percentage of relative errors in the measurement results that are not smaller than 0.05, 0.1, 0.2, and 0.4. Table 3.4 shows the average values of the relative errors in the measurement results. The relative errors in the measurement results of our method are approximately only 65% of those in the method in [12]. To explain these results, we use the evaluation results of the average number of measurements, the average measurement frequencies and the average measurement conflict rates of an overlay path during an aggregation period, shown in Tabs. 3.5, 3.6 and 3.7, respectively. The number of measurements and measurement frequency in our method are much larger than those of the method in [12], but only about 12% of the measurements experience conflicts. Therefore, our method ' s measurement accuracy surpasses the method in [12]. We also observe in Tabs. 3.1, 3.2, 3.3 and 3.4 that the Waxman topology has smaller relative error than the AT&T and BA topologies for the following reason. From the simulation results, we found fewer half and partial overlapping

Table 3.5: Average number of measurements during an aggregation period

Method \ Topology	AT&T	BA	Waxman
Existing method	3.287	5.912	13.202
Proposed method	11.050	20.259	28.388

Table 3.6: Average measurement frequencies during an aggregation period

Method \ Topology	AT&T	BA	Waxman
Existing method	2.739×10^{-3}	4.927×10^{-3}	11.002×10^{-3}
Proposed method	9.208×10^{-3}	16.883×10^{-3}	23.657×10^{-3}

Table 3.7: Average measurement conflict rates during an aggregation period

Method \ Topology	AT&T	BA	Waxman
Existing method	0.000	0.000	0.000
Proposed method	0.141	0.107	0.119

paths in the Waxman topology than in the AT&T and BA topologies. Therefore, the measurement frequency is the largest, meaning that the number of measurements is the largest, and thus the relative error is the smallest in the Waxman topology.

Measurement traffic load Tables 3.8 and 3.9 show the average number of packet fleets traversing one link during an aggregation period and the average number of packet fleets traversing one link at one measurement. As shown in Tab. 3.8, in our method, the average number of packet fleets traversing one link during an aggregation period is larger than that in the method in [12]. This is because the measurement frequency in our method is much larger (Tab. 3.6). However, the average number of packet fleets traversing one link at one measurement in our method is smaller than that of the method in [12] for the following reason. In the method in [12], because the search range in each

Table 3.8: Average number of packet fleets traversing one link during an aggregation period

Method \ Topology	AT&T	BA	Waxman
Existing method	634.483	510.533	828.918
Proposed method	1508.375	1374.447	1411.387

Table 3.9: Average number of packet fleets traversing one link at one measurement

Method \ Topology	AT&T	BA	Waxman
Existing method	6.000	6.000	6.000
Proposed method	5.814	5.784	5.674

Table 3.10: Distribution of relative errors when measurement traffic loads of existing and proposed methods are identical (AT&T topology)

Method \ Relative error	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	56.524%	32.274%	9.7203%	1.404%
Proposed method	54.095%	28.301%	6.960%	0.621%

end-to-end measurement of the available bandwidth is set to $(0, C)$, the number of packet fleets at one measurement is constant for all measurements. On the other hand, in our proposed method, since the search range is calculated based on the measurement results exchanged between overlay nodes, it is narrower and nearer the real value of the available bandwidth than the search range in the method in [12]. Therefore, the number of packet fleets at one measurement is smaller, meaning that the traffic load of each measurement is smaller in our proposed method.

Evaluation results with equal measurement traffic load Next, we adjust the measurement frequencies in our proposed method so that its measurement traffic loads and those of the method in [12] are the same. Tables 3.10, 3.11, 3.12 and 3.13 show the evaluation results of the relative

Table 3.11: Distribution of relative errors when measurement traffic loads of existing and proposed methods are identical (BA topology)

Method \ Relative error	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	51.792%	30.487%	10.095%	1.244%
Proposed method	50.120%	26.604%	6.923%	0.534%

Table 3.12: Distribution of relative errors when measurement traffic loads of existing and proposed methods are identical (Waxman topology)

Method \ Relative error	≥ 0.05	≥ 0.1	≥ 0.2	≥ 0.4
Existing method	34.835%	16.453%	3.848%	0.188%
Proposed method	34.283%	13.938%	2.279%	0.048%

Table 3.13: Average relative errors when measurement traffic loads of existing and proposed methods are identical

Method \ Topology	AT&T	BA	Waxman
Existing method	0.088	0.083	0.051
Proposed method	0.077	0.073	0.046

errors in the measurement results in the AT&T, BA, and Waxman topologies. Table 3.14 shows the average number of packet fleets traversing one link during an aggregation period of the method in [12] and our proposed method, which are almost the same, as expected. Table 3.15 shows the average number of packet fleets traversing one link at one measurement of the method in [12] and the proposed method. Even though the number of packet fleets traversing one link at one measurement in [12] is unchanged, that value in our proposed method is slightly larger than before adjusting the measurement frequencies. This is because the measurement frequency in our method is reduced, thus the number of measurement results used for calculating parameters of each measurement also decreases, and the resulting search ranges are not near the real value of available bandwidths as

Table 3.14: Average number of packet fleets traversing one link during an aggregation period when measurement traffic loads of existing and proposed methods are identical

Method \ Topology	AT&T	BA	Waxman
Existing method	655.325	527.841	822.997
Proposed method	652.432	526.810	821.838

Table 3.15: Average number of packet fleets traversing one link at one measurement when measurement traffic loads of existing and proposed methods are identical

Method \ Topology	AT&T	BA	Waxman
Existing method	6.000	6.000	6.000
Proposed method	5.875	5.867	5.730

Table 3.16: Average number of measurements during an aggregation period when measurement traffic loads of existing and proposed methods are identical

Method \ Topology	AT&T	BA	Waxman
Existing method	3.562	5.777	12.713
Proposed method	3.736	5.966	13.345

before adjusting the measurement frequencies.

However, as shown in Tabs. 3.10, 3.11, 3.12 and 3.13, the measurement accuracy of the proposed method is slightly better than that of the method in [12]. This is because the number of measurements and the measurement frequency in our method remain larger than those of the method in [12], as shown in Tabs. 3.16 and 3.17, the measurement conflict rate is small, as shown in Tab. 3.18, and the search range in our proposed method is narrower and nearer the real value of the available bandwidth than the method in [12], as explained above.

Table 3.17: Average measurement frequencies during an aggregation period when measurement traffic loads of existing and proposed methods are identical

Method \ Topology	AT&T	BA	Waxman
Existing method	2.968×10^{-3}	4.814×10^{-3}	10.594×10^{-3}
Proposed method	3.113×10^{-3}	4.972×10^{-3}	11.121×10^{-3}

Table 3.18: Average measurement conflict rates during an aggregation period when measurement traffic loads of existing and proposed methods are identical

Method \ Topology	AT&T	BA	Waxman
Existing method	0.000	0.000	0.000
Proposed method	0.124	0.095	0.114

3.6 Summary

In this chapter, we proposed a distributed measurement method for available bandwidth in overlay networks that reduces measurement conflicts by detecting path overlappings and adjusting the measurement frequencies and the measurement timings of overlay paths. We also proposed a method to improve measurement accuracy while reducing the traffic load of each measurement by exchanging the measurement results among neighboring overlay nodes. Simulation results show that the relative errors in the measurement results of our method are approximately only 65% of those of the existing method. The measurement accuracy of our method remains better than the existing method when the total measurement traffic loads of both methods are equal.

Chapter 4

Measurement method for link fault diagnosis

4.1 Introduction

Fault diagnosis is important for efficient operation of overlay network services. Overlay service providers can use fault diagnosis tools to locate faulty components in the end-to-end paths to repair or bypass them. Faults in overlay networks can occur at IP level components (e.g., routers or IP links) or at overlay nodes for a variety of reasons. For instance, faults in the underlying IP layer may be due to: (1) hardware problems e.g., fiber cuts and malfunctioned router interfaces, (2) software errors, e.g., router software bugs, and (3) network misconfiguration, e.g., BGP misconfiguration. On the other hand, complicated functions (e.g., virtualization, caching, or probing) commonly adopted on overlay nodes increase the chance of depletion of their resource (e.g., CPU or bandwidth depletion).

Solutions for detecting faults can be mainly divided into two categories : passive measurement and active measurement. Passive measurement methods [41–43] use network level and end user level negative symptoms to identify faulty components. These passive methods require very small active probes, thus have an advantage of non-intrusiveness. However, because they rely on symptoms observed by end users, which can be lost or spurious, they may take a long time to discover

faulty components or make wrong detection (misjudge a good component as faulty one, and vice versa).

To avoid these problems, an alternative diagnostic approach has been developed that is based on active probing technology [4, 44–60]. A typical active probe solution involves two phases. In the first phase, called fault detection phase, a set of paths between measurement agents (end nodes) is selected such that these paths cover all of the links. These paths are periodically probed until some failures on them are detected. When failures are detected, the second phase, called fault localization phase, is initiated, and more probes are sent to the problematic network segments to identify exactly the locations of the faulty components. Most of the current solutions only focus on how to select a minimum set of end-to-end paths that can cover all the links of interest. However, the probe frequency and probe timings for these paths are not provided. Furthermore, most of existing methods do not care the problem of well-balancing probe overhead over the links. Another setback is that most of these solutions use centralized approaches; they assume that a Network Operation Center (NOC) will aggregate the topology of the underlay network, choose probe stations and probing paths, calculate probing timings for each path, gather probed results from probe stations and locate the faulty links. Therefore, these methods have difficulties to deploy in large scale networks.

Our aim in this paper is to propose an active probing method for rapidly diagnosing faulty components in overlay networks. To make probing overhead of overlay nodes well-distributed, we apply a distributed method, in which each overlay node monitors the paths starting from itself, and exchanges the result of each probe immediately with the source nodes of overlapping paths. Furthermore, to balance the probing overhead well over the links, we set a constraint of probing overhead for each link. Overlay node then chooses the probing paths that contain the links whose overhead are smaller than their constraints. If we can not select a path that satisfies all constraints of its links, we gradually increase the constrained overheads, and choose the paths with the updated constraints. After probing the selected path, overlay node exchanges the monitoring result of each probe immediately with source nodes of overlapping paths. When no fault is detected, overlay node reduces the number of the paths to be probed in one probing period, using the probing results received from other nodes. This helps to minimize the probing overhead. Furthermore, the probe

timings of each overlay node are dynamically decided based on the number of the paths that the overlay node must probe. Therefore, when the number of probe paths are reduced, the probe timings are also spread over each period of fault detection phase. This helps to detect faulty components rapidly. Simulation results show that our method is much more effective than an active measurement method proposed in [57], which is one of the newest methods in the literature and is the nearest with our approach.

The rest of this chapter is organized as follows. Section 4.2 describes related work. Definitions and assumptions related to overlay networks are presented in Section 4.3. In Section 4.4, we explain our method that reduces measurement conflicts while decreasing the overhead of each measurement. We evaluate our method in Section 4.5 and conclude this chapter in Section 4.6.

4.2 Related work

Some of the overlay network monitoring methods utilize passive measurements to reduce measurement overhead [41–43]. The method in [41] obtains statistical information for end node faults and observes network symptoms of overlay links to calculate the set of prior fault probabilities of overlay components (end nodes or links), called Overlay Network Profile (ONP). It then uses ONP to construct a graph that demonstrates the relations between the negative symptoms (high loss rate or latency) and the overlay components. From this graph, it calculates the contribution of fault probabilities of overlay components, and chooses the set of components with largest fault probabilities as the candidates of faulty components. Finally, active probes are performed to confirm the faulty components. DigOver [42] also uses end-user observed negative symptoms to identifies a set of potential faulty components. It quantifies fault likelihood and the corresponding uncertainty of each component, and constructs a plausible fault graph to locate the most likely faulty components. The method in [43] utilizes end user observed packet loss and route information of overlay paths to construct a set of constraints related to packet loss rates of overlay components. By solving this set of constrains, the method obtains packet loss rate of each component and infers faulty ones. These methods are non-intrusive but require a long time to detect faulty components. Furthermore, the methods in [41, 42] require predetermined fault probability of each component, while the accuracy

of the method in [43] heavily relies on the number and settings of constrains. Contrast to these method, we adopt active probing approach, thus can rapidly detect faulty components. Furthermore, our method does not require predetermined statistical information of overlay components as the methods in [41, 42].

The tomography-based approach, which only rely on end-to-end measurements to infer link properties, is another efficient way for detecting faulty components in overlay networks. The methods of this approach can be divided into two categories [61]: analog and binary methods. In the former methods [4, 58–60], the link properties such as loss rate or latency are accurately calculated from the measurement results of some paths that cover all the links of the network. However, these methods are error-prone in practice, because they require high accuracy of the end-to-end measurements. Therefore, the binary methods [50–52, 55–57], which estimate link as either "good" or "bad", are expected to be more practical. Similar to analog methods, the status of links are also inferred from the estimated results of end-to-end paths. Although these method are more robust, they produce less information of link properties than the analog ones. The main idea of the binary methods is to find a minimum set of end-to-end paths that can cover all of the links of interest. The minimal set covering problem have been proved as NP-hard [51], and some heuristic algorithms have been proposed so far [50, 51, 55, 57]. Duffield [55] considers a topology consisting of paths from a single source to multiple destinations, and presents the "Smallest Common Failure Set" algorithm, which defines as bad only those links nearest the source. Nguyen and Thiran [50] propose an algorithm based on max-plus algebra that can detect multiple simultaneous link faults. The methods in [52, 56] use full-mesh probing in the phase of fault detection and focus on rapidly locating fault links. Dhamdhere et. al. [56] propose a heuristic algorithm that find a minimum set of fault links that can explain the failures of a probed paths. The method in [52] first calculates the probabilities that links become faulty through few initial measurements, and uses these probabilities and the current probing results to estimate the most possible faulty links. These methods assume that per-path overhead is negligible so that the links that are probed by more than one path do not suffer high probing overhead. However, because many probing algorithms can produce 10s to 100s of Kbps of traffic or more, this assumption can become problematic in operational networks [57]. The first method that considers the problem of balancing the probing overhead over the network

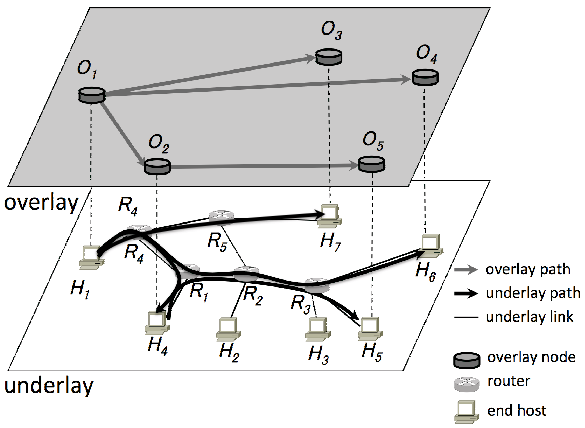


Figure 4.1: Example of overlay network and path overlapping

links is the method proposed in [57], which is the nearest with our approach. In this method, each link is attached with a weight that reflects the length of duration from the last probing time of the link, and the weight of a path is calculated as the sum of the weights of its links. At each probing interval of this method, a set of k paths, whose weights are the largest, is chosen to be probed. Although this algorithm seems to balance the probing overhead over the links better than previous methods, the goal of well-balancing overhead is still far from being reached.

Contrast to the passive measurement methods, our solution does not require heavy instrumentation; it relies only on end-to-end measurements. Our method also differs from current tomography-based approach: it tries to choose probing paths that can keep the probing overheads of it links under some constrains rather than choosing the paths that can cover the most unprobed links. Furthermore, our method works in a completely distributed fashion, while all of the above methods are centralized methods. This makes our method more suitable to be deployed in large scale systems.

4.3 Definitions and assumptions

In this paper, we define an overlay network as a logical network constructed on an under-layer IP network in which the overlay nodes are installed on end hosts as an application program.

We call an under-layer IP network an *underlay network* and consider an underlay network with

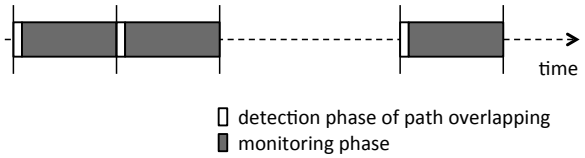


Figure 4.2: Monitoring procedure

m end hosts, denoted by H_i ($i = 1, ..., m$), and l routers, denoted by R_i ($i = 1, ..., l$). Suppose that n ($n \leq m$) overlay nodes are deployed on n different end hosts. We denote the overlay nodes as O_i ($i = 1, ..., n$). Density σ of the overlay nodes is defined as the ratio of the number of overlay nodes to the number of end hosts, i.e., $\sigma = n/m$. Figure 4.1 shows an example of an underlay network and the overlay network constructed on it. The gray arrows show the overlay paths, and the black arrows illustrate the underlay paths of the corresponding overlay paths. We assume the shortest path algorithm for routing in the underlay network and denote the underlay path between overlay nodes O_i and O_j as O_iO_j. For path O_iO_j, O_i is the *source node*, and O_j is the *destination node* of the path. If different paths O_iO_j and O_sO_t share at least one link, O_iO_j and O_sO_t overlap each other, or O_iO_j (O_sO_t) is an *overlapping path* of O_sO_t (O_iO_j). We define a *route* from O_i to O_j as a sequence of underlay nodes that construct an underlay path from O_i to O_j.

We classify the overlapping states into the following two types:

- Half overlapping: Two paths share a route from the source node to a router that is not an overlay node.
- Partial overlapping: Two paths share a route that does not include the source node.

For example, in Fig. 4.1, paths O_1O_2 and O_1O_4 have a half overlapping relation. Path O_1O_4 is a partial overlapping path of O_2O_5.

4.4 Proposed method

The monitoring procedure of our method contains two phases: detection phase of path overlappings and monitoring phase, as shown in Fig. 4.2. In the first phase, we use the same mechanism as a Chapter 2 to detect overlapping paths.

The second phase contains two steps: probing the paths and localizing the faulty links when some failures are detected. In the first step, the overlay node periodically probes the paths starting from itself and shares probing results of overlapping paths with other nodes and uses these data to reduces the number of paths to be probed. If some failures are detected, the source node performs the second step: locates the faulty links using exchanged information.

We choose a distributed approach for our method to apply to large scale networks, because of the following reasons. First, because the problem of selecting probing paths is very time consuming, as explained later, in a centralized solution, a master node must have a very large computing capacity. Furthermore, the traffic between the master node and other overlay nodes will concentrate at the links near the master node. Thus, constructing and maintaining the master node and the network around it will require a huge cost. Second, a distributed approach is especially efficient for locating faulty links, because when a probe of a path fails, we only need to gather the probing results of the paths related to the failed probe at its source node to locate the faulty components, rather than aggregating probing results all over the network at the master node in centralized approach.

4.4.1 Fault detection

Problem statement

We define a network that contains N overlay nodes, and consider the problem of detecting faults in the links contained in all the paths between the overlay nodes with the smallest possible overhead. Denote the total number of links of interest as M. This problem can be consider as an instance of the *Weighted Set Cover Problem*, which is defined as follows.

Definition 1 *Weighted Set Cover Problem (WSC) :*

Given a universe X of elements, and a collection F of subsets $S \subset X$, where each $S \in F$ has an associated non-negative cost, find a subcollection $C \subset F$ of minimum total cost that covers X, assuming one exists.

The WSC problem has been proved to be an NP-hard problem [62], and a typical centralized solution for this problem can be the greedy algorithm 7 [63]. To apply to our problem of selecting

probing paths, we consider F as the set of all paths S between overlay nodes, X as the set of all the links of these paths, and C as the set of selected probing paths. $cost(S)$ is the probing overhead of the path S. The term $\frac{|S \cap U|}{cost(S)}$ illustrates the "efficiency" of the selected path, which is the number of uncovered links per unit cost. This means that we try to select the paths with maximum probing efficiency.

Algorithm 7 Greedy algorithm for the Weighted Set Cover Problem

```
1: function GreedyWSC()
2: //initilization
3: C ← ∅
4: U ← X //set of uncover links
5: while U ≠ ∅ do
6:     Find set S ∈ F \ C that maximizes |S∩U| / cost(S)
7:     C ← C ∪ {S}
8:     U ← U \ S
9: end while
10: return C
11: end function
```

Algorithm 7 requires $O(|X||F|) = O(MN^2)$ of time [64], thus has the difficulty to apply to large scale networks. Furthermore, because the overhead of each link is not considered, the distribution of overhead may not be well spaced. We therefore propose a novel probing method to tackle these problems.

Overview of our method

To reduce the time complexity, we apply a distributed method, in which each overlay node autonomously select the paths to probe. Furthermore, we propose a heuristic scheme for selecting probing paths that can reduce the total probing overhead while keeping the probing overhead well-distributed between links. More specially, we first set a constrain of overhead for each link of interest, and find the paths whose probing overhead satisfies all constrains of its links. Among these paths, we choose the paths that satisfies the following criteria.

- The paths that maximize the "probing efficiency"

Similar to the solution of the WSC problem (Algorithm 7), we use this criterion to keep the total probing overhead as small as possible.

- The paths that has smallest number of partial overlapping paths

  Because the partial overlapping paths can be probed simultaneously, we choose the path with smallest number of partial overlapping paths, to reduce the overhead of the links in their overlapping parts.

If there are more than one path satisfying the above two criteria, then we randomly choose one of them and probe it. The probing result is exchanged with the source nodes of its partial overlapping paths. After probing each path or receiving probing result from other nodes, the probing overheads of its links are updated, and are used to select the next probing paths. If we can not select a path that satisfies all constrains of its links, we gradually increase the constrained overheads, and choose the paths with the updated constrains. This process is repeated until all the links of interest are probed. The number of iterations is about $N - 1$ in the worst case, but normally much smaller than this value.

We also propose a scheme for dynamically determining the probing timings of the paths based on the exchanged probing results, to balance probing overhead over time, and reduce the probability of simultaneous probing of overlapping paths.

Details of our solution

Method for determining probing timings We explain the method for an overlay node O_i. O_i has to probe the paths starting from itself to all of other nodes, denoted as O_iO_j, $(1 \leq j \leq N, j \neq i)$. The total number of these paths is $N - 1$, and these paths together construct a tree with the root node is O_i. We denote the set of links of this tree as Φ_{O_i}.

Assume that O_i must probe all the links in Φ_{O_i} at each duration of T (seconds), called a *probing period*. If O_i does not receive any probing results from other nodes during a probing period, O_i must probe all of these $N - 1$ paths. This means O_i must probing one path for each period of $T/(N - 1)$ (seconds). For simplicity, we assume that the duration of each probe is τ. The value of τ depends on the type of probing metric and tool. Because the probability of fault occurring is

quite small [55], and we need to keep the probing traffic small enough to not disturb other normal traffic in the network, the probing period should be selected such that $T/(N-1) \gg \tau$.

Because in our method, the overlay nodes exchange probing results to reduce probing overhead, the probes of some paths can be omitted, as explained latter. Therefore, the number of paths that O_i must probe at one probing period may be much smaller than $N-1$. We therefore propose a dynamic scheme of determining probing timings of O_i, based on the number of unprobed paths. More specially, at the time t^* in one probing period, assume that O_i must probe k ($k \leq N-1$) paths. We then divide the remaining time $T - t^*$ of the probing period into k time slots. In each time slot, we randomly select a timing, which is used as the start of probing time of one unprobed path. Note that we do not decide the order of the probing paths at this moment: the probing path is dynamically chosen at the time just before the decided probing timing based on the probing results of O_i and the results received from other nodes.

Because the intervals between two consecutive probes are rapidly increased, this method has the following advantages. First, the probing overhead of each overlay node is well-distributed over the duration of each probing period. Second, the probability of simultaneous probing of overlapping paths is gradually reduced, thus help to keep the probing overhead of each links small. Third, overlay nodes can receive more probing results from other nodes, thus can reduce probing overhead of each node. Fourth, if the probability of fault occurring is uniformly distributed over the time, spreading probes over the probing period helps us detect faulty components early.

Method for selecting probing paths Algorithm 8 shows the probing procedure of overlay node O_i. For each link $l \in \Phi_{O_i}$, we set an initial constrained overhead of $w = 1$, and denote the current probing overhead of h_l. The initial value of h_l is set to 0. At each decided probing timing, O_i chooses randomly one unprobed path whose satisfies two criteria: maximizes the probing efficiency, and has the least number of partial overlapping paths. O_i probes this path, and immediately sends the result to the source nodes of its partial overlapping paths. After each time O_i probes a path or receives one probing result, denote the set of links of this probed path as P. The probing overhead h_l of each link l in $\Phi_{O_i} \bigcap P$ is updated as $h_l + 1$. O_i then marks the links in the set $\Phi_{O_i} \bigcap P$ as probed links, and check if there are some paths whose probe can be omitted. These paths are the

Algorithm 8 Probing algorithm of overlay node O_i

function DistributedProbing()
//initilization
Add the paths that O_i needs to probe to the set F
$C \leftarrow \emptyset$
Add the links of all the paths in F to the set Φ_{O_i}
$w \leftarrow 1$ // initial probing constrain
for $l \in \Phi_{O_i}$ **do**
  $h_l \leftarrow 0$ // initial link overhead
end for
$U \leftarrow \Phi_{O_i}$ //set of uncover links
while $U \neq \emptyset$ **do**
  Find the paths $S \in F \setminus C$ whose link overheads are all smaller than w, and add to the set V
  if $V \neq \emptyset$ **then**
    Find the paths $S \in V$ that maximizes $\frac{|S \cap U|}{cost(S)}$, and add to the set T
    Find the paths $S \in T$ whose number of partial overlapping paths are smallest, and add to the set Q
    Randomly choose one path, denoted as S, from Q and probe path S
    for $l \in S$ **do**
      $h_l \leftarrow h_l + 1$ // increase link overhead
    end for
    $C \leftarrow C \cup \{S\}$
    $U \leftarrow U \setminus S$
  else
    $w \leftarrow w + 1$ // increase probing constrain
  end if
  Receive probing result of some path P from other overlay node
  for $l \in P \cap \Phi_{O_i}$ **do**
    $h_l \leftarrow h_l + 1$ // increase link overhead
  end for
  $U \leftarrow U \setminus P$
  for $S \in F \setminus C$ **do**
    if $S \equiv \emptyset$ **then**
      $F \leftarrow F \setminus S$ // omit probe of path S
    end if
  end for
end while
end function

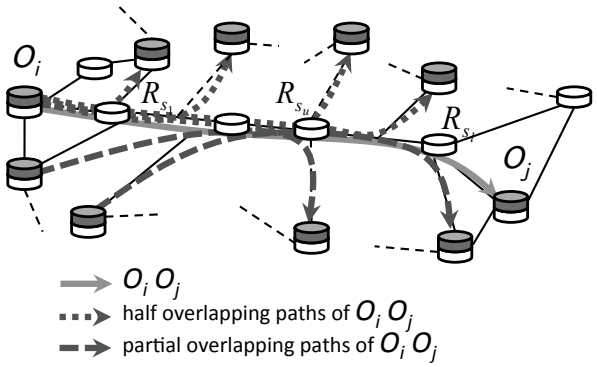


Figure 4.3: O_iO_j and its overlapping paths

paths that contain only probed links. The procedure of selecting probing path is repeated with the updated link overheads. If O_i can not find an unprobed path whose link overheads are below the constrained overhead, the constrain w of each link l are updated as $w + 1$. The probing procedure of O_i at one probing round will be stopped when all the links in Φ_{O_i} are marked as probed links.

Our method can be also considered as a heuristic solution of the minimal set covering problem. However, contrast to the existing method, the overhead between links and overlay nodes is better balanced. Therefore, if the probabilities of fault occurring are identical between all links, we can expect that our method detects faulty links faster than existing method, when the probing overheads are equal.

4.4.2 Fault localization

Once a probe of a path has been failed, the localization phase is initiated in order to find the links responsible for the fault. For simplicity, we assume that only one link on the path is problematic. This assumption is quite common in the literature, and is reasonable because the probability of fault occurring is small [51, 55, 57]. In this section, we formalize the problem of fault localization and propose a method that can rapidly locate the faulty links.

Problem statement

Assume that at time t^* in a probing period, a probe of path O_iO_j failed, because some faults had occurred in the links contained in this path. We assume that the overlapping parts of O_iO_j and its half and partial overlapping paths are divided by routers R_{s_1}, R_{s_2}, ..., R_{s_l}, as shown in Fig. 4.3. Define $\mathcal{H}_{O_iO_j} = \{O_iR_{s_1}, R_{s_1}R_{s_2}, ..., R_{s_{l-1}}R_{s_l}, R_{s_l}O_j\}$. Because we use end-to-end probing between overlay nodes to locate faulty links, we can only identify which segment in $\mathcal{H}_{O_iO_j}$ is responsible for the fault. The problem is that we must select a minimum set of overlapping paths of O_iO_j to rapidly locate the faulty segment.

Overview of our method

We propose a distributed method for fault localization, in which O_i exchanges probing results with other nodes to identify the faulty segment of O_iO_j. O_i first uses some probing results of half and partial overlapping paths of O_iO_j it obtained up to the time t^*, to narrow down the set $\mathcal{H}_{O_iO_j}$ to a smaller set of suspected segments, denoted as $\mathcal{G}_{O_iO_j}$. If $\mathcal{G}_{O_iO_j}$ contains only one segment, then it is the faulty one, and O_i stops the phase of fault localization. Otherwise, O_i selects some overlapping paths of O_iO_j for additional probes. If the selected paths are half overlapping paths of O_iO_j, O_i probes them by itself. If the selected paths are partial overlapping paths of O_iO_j, O_i requires the source nodes of these paths to probe them and report results to it. O_i then uses the probing results to locate the faulty segment. The basic idea of our method for selecting additional probing paths is that we choose and probe the paths one by one so that after each probe, the expected value of the number of remaining unprobed suspected segments is smallest. Our method is similar to the scheme proposed in [57]. However, contrast to our method, the method in [57] chooses the probing paths so that after each probe, the expected value of the number of remaining unprobed suspected links is smallest. This scheme may be efficient for locating faulty links, but not best fit for the problem of locating faulty segments.

Details of our solution

We first explain how O_i uses some probing results of half and partial overlapping paths of O_iO_j

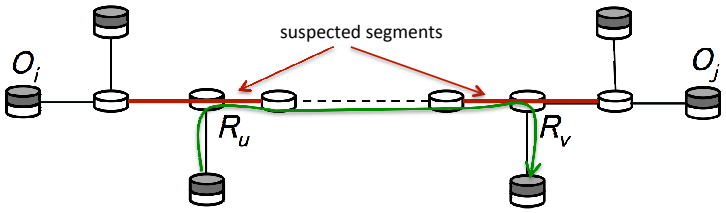


Figure 4.4: O_iO_j and suspected segments

it obtained up to the time t^*, to narrow down the set $\mathcal{H}_{O_iO_j}$ to the set of suspected segments $\mathcal{G}_{O_iO_j}$. The probing results that can be used is the ones whose probing timings are between the interval $[t^* - \epsilon, t^*]$, where ϵ is a parameter. ϵ can be set based on the probing frequency in the fault detection phase and the average duration of faults. Denote the set of half and partial overlapping paths of O_iO_j whose probing timings are between the interval $[t^* - \epsilon, t^*]$ as $V_{O_iO_j}$, and the set of other half and partial overlapping paths of O_iO_j as $W_{O_iO_j}$. First, set $\mathcal{G}_{O_iO_j} = \mathcal{H}_{O_iO_j}$. Then for each probing result of path $S \in V_{O_iO_j}$, we repeat $\mathcal{G}_{O_iO_j} = \mathcal{G}_{O_iO_j} \setminus S$.

If $\mathcal{G}_{O_iO_j}$ contains only one segment, then this is the faulty one and O_i stops the phase of fault localization. Otherwise, O_i selects the paths from $W_{O_iO_j}$ for extra probes. We explain the detail method for selecting paths as follows. Consider a path S from $W_{O_iO_j}$ that overlaps with O_iO_j at the overlapping part between R_u and R_v, as shown in Fig. 4.4. Denote $\mathcal{B}_S = \mathcal{G}_{O_iO_j} \bigcap R_uR_v$ as the set of suspected segments contained in path S. Denote the number of segments in $\mathcal{G}_{O_iO_j}$ and $\mathcal{B}_S$ as m_G and m_S ($m_S \leq m_G$), respectively. Furthermore, denote the number of links contained in segments in $\mathcal{G}_{O_iO_j}$ and $\mathcal{B}_S$ as l_G and l_S ($l_S \leq l_G$), respectively. Similar to [57], we assume that each link has an equal chance to exhibit a fault. Therefore, the probability that the probe of S fails is $\alpha_S = l_S/l_G$. If the probe of S fails, the set of suspected segments reduces to the set $\mathcal{B}_S$, while if the probe of S succeeds, the set of suspected segments reduces to the set $\mathcal{G}_{O_iO_j} \setminus \mathcal{B}_S$. Therefore, the expected value of number of suspected segments after a probe of S is the value E_S given in Eq. (4.1).

$$E_S = \alpha_S m_S + (1 - \alpha_S)(m_G - m_S) \tag{4.1}$$

Thus, to rapidly locate faulty segment, we choose and probe the path S that has the minimum value

Algorithm 9 Probing algorithm of overlay node O_i to locate faulty links of O_iO_j

```
1: function LocatingFaults()
2: //initilization
3: F ← W_{O_iO_j} // set of unprobed paths
4: U ← G_{O_iO_j} //set of uncovered segments
5: while |U| > 1 && F ≠ ∅ do
6:    Find the path S ∈ F that minimizes the value E_S
7:    if S is a half overlapping path of O_iO_j then
8:       O_i probes S
9:    else
10:      O_i requires the source node of S to probe it and report the result to O_i
11:   end if
12:   if probe of S fails then
13:      U ← U ∩ S
14:   else
15:      U ← U \ S
16:   end if
17:   F ← F \ {S}
18: end while
19: return U
20: end function
```

of E_S.

Algorithm 9 shows the details of the procedure for selecting and probing paths from $W_{O_iO_j}$. O_i chooses randomly one unprobed overlapping path S of O_iO_j that minimizes the value E_S, calculated by Eq. (4.1). If path S is a half overlapping path of O_iO_j, then O_i probes this path. Otherwise, if path S is a partial overlapping path of O_iO_j, O_i requires the source node of S to probe it and report the result to O_i. If the probe of path S fails, $\mathcal{G}_{O_iO_j} = \mathcal{G}_{O_iO_j} \bigcap S$. Otherwise, $\mathcal{G}_{O_iO_j} = \mathcal{G}_{O_iO_j} \setminus S$. The probing procedure of O_i is repeated until $\mathcal{G}_{O_iO_j}$ contains only one segment, or all the paths in $W_{O_iO_j}$ are probed.

4.5 Performance evaluation

In this section, we evaluate the performance of our proposed method. We focus on evaluating how our proposed method of selecting probing paths relatively reduces the detection time, localization time and probing overhead, rather than absolutely evaluating the accuracy of fault detection and

localization. Therefore, we only use evaluations based on simulations, and assume that when a fault occurs on a link, we can correctly detect the failures of the probes of the paths containing that link. Evaluation based on experiments in real networks is our future work. We explain the evaluation method in Subsection 4.5.1 and present our evaluation results and discussions in Subsection 4.5.2.

4.5.1 Evaluation method

We compared our proposed method with an existing method [57], which we briefly explain, and make some assumptions for comparison in Subsection 4.5.1. We then explain the evaluation metrics and the simulation settings in Subsections 4.5.1 and 4.5.1.

Existing method [57]

The method in [57] is the only existing method so far that also considers the problem of balancing the probing overhead over the network links as our method. In this method, in the phase of fault detection, each link is attached with a weight that reflects the length of duration from the last probing time of the link, and the weight of a path is calculated as the sum of the weights of its links. At each probing interval of this method, a set of k paths, whose weights are the largest, is chosen to be probed. The key advantage of this method compared with minimum set cover based approaches is that the computations are divided over a series of probing intervals, thus reducing overhead at each interval. In the phase of fault localization, the method in [57] chooses the probing paths one by one so that each probing path covers about half of remaining unprobed suspected links.

To compare our method with the method in [57], in the phase of fault detection, we must carefully choose the value of k so that the two methods have equal conditions. We urge that the best suit value of k is $k = N$, which is the number of overlay nodes. This is because, in our method, each overlay node probes paths one by one, therefore we can approximately consider that our method chooses N paths at one probing interval over the entire network. Furthermore, in order to make the two methods run in as similar condition as possible, in the method in [57], the paths in a selected group of N paths are probed sequentiallly with the same intervals. The methods will be stopped as soon as all of the links of interest are covered, or the probe of one path is failed.

Note that the method in [57] proposes another option, called *k-max*, that does not use a fixed value of k, but selects all of maximum weighted paths that do not overlap with each other from the full-mesh $N(N-1)$ paths. However, we do not use this option to compare with our method, because this option must select paths from $O(N^2)$ paths, thus is very time-consuming in large networks, which is contradictory with the advantage of this method explained above.

Furthermore, we try to set the same total probing overhead for the two methods: we first set a period for the method in [57] to monitor all of the links of interest, and then adjust the monitor period in our the method so that the probing overheads of two method can be nearly equivalent.

In the phase of fault localization, we do not need any settings like the above one, because the method in [57] and our method both basically select paths one by one.

Evaluation metrics

In the phase of fault detection, we compare our proposed method and the method in [57] with the following metrics:

- Detection time

  This is defined as the average duration from the time when a fault occurs to the time it is detected.

- Total traffic load

  This is defined as the total traffic load for probing and information exchange over all links in the network, until a fault is detected.

- Distribution of probing overhead over the links of interest

  We use this metric to evaluate the load-balancing effectiveness.

In the phase of fault localization, we use the total number of extra probing paths to locate the faulty path segments, to evaluate the efficiency of our method and the existing method. Note that we do not evaluate the localization accuracy, because we assume that we can correctly detect the failures of the probes of paths containing the faulty link.

Simulation settings

We used three types of large network topologies: the AT&T topology [28], generated topologies based on the Barabasi-Albert (BA) [29], and the Waxman models [30] for the router level topology. We generated ten topologies for each model using the BRITE topology generator [31]. All topologies have 523 routers and 1304 links. We randomly selected 20% of the 523 routers, and assume that each of them is directly connected to one end host that is an overlay node. For averaging the results, the choice of the routers that directly connects to overlay nodes was taken 100 times for the AT&T topology and ten times for each topology of the BA and Waxman models. We set the time of each probe to 5 [s] and assume that the duration for monitoring all the links of interest is one hour. For simplicity, we assume fault occurs with the same probability for all the links of interest, and at each fault detection phase, only one link experiences fault. For averaging the results, with each topology, the choice of the faulty link was taken 5 times. Furthermore, for each selected faulty link, the time of fault occurring was randomly taken 5 times. Finally, for each scenario of faulty link and fault occurring time, we run our method and the method in [57] 5 times and take the average results. We assume that probing of packet loss rate is used to detect faulty links. The packet loss rate in each probe is simulated based on transmission of 10000 packets as in [4]. The probing result of one path, which is the binary value of "fault" or "no fault", is exchanged between overlay nodes using one packet. The size of each packet in information exchange and probing is 50 bytes.

Our simulation program was written in C language and run on commodity Linux machines with default settings.

4.5.2 Evaluation results and discussions

Fault detection

Figures 4.5 and 4.6 show the average value of detection time and the total traffic load over all the links in the network in AT&T, BA and Waxman topologies, respectively. We first see in Fig. 4.6 that the traffic load of probing occupies the most part of the total traffic load. This is because the traffic load of each probe is much larger than the traffic load of one time of information exchange. We can also see in Figs. 4.5 and 4.6 that the detection time in our method is much smaller than

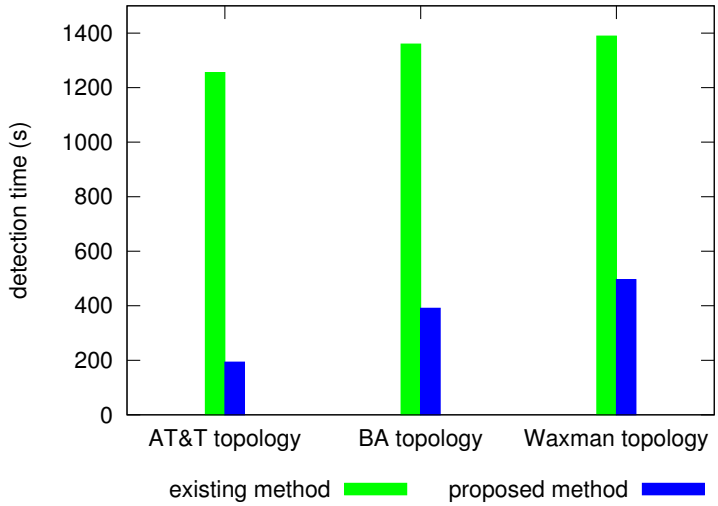
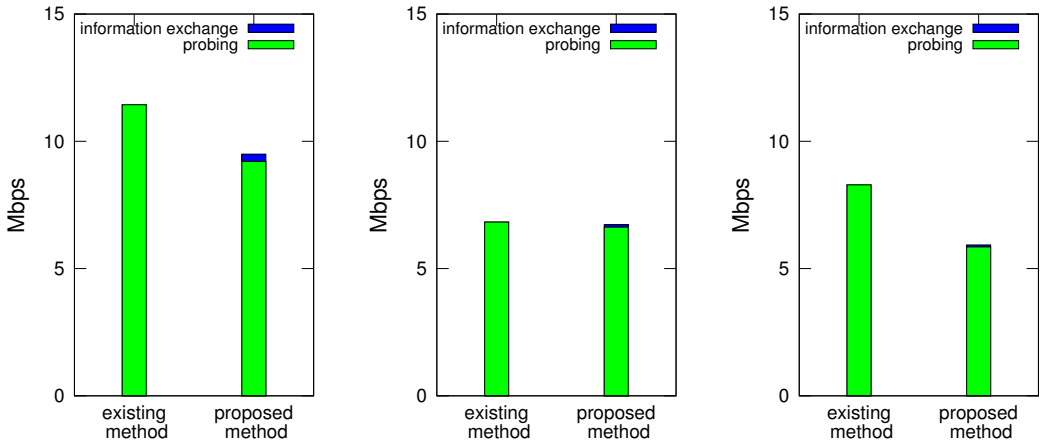


Figure 4.5: Average of detection time (seconds)

that in the method in [57], while the total traffic load of our method is still smaller than that of the method in [57]. We use the distribution of the probing overhead per link and the distribution of the probe timings, shown in Figs. 4.7 and 4.8 respectively, to explain this result. As we can see in Fig. 4.7, the distribution of probing overhead per link in BA and Waxman topologies of our method and the method in [57] are almost the same. However, in AT&T topology, the probing overhead per link in the method in [57] is especially badly distributed compared to our method. This is because in AT&T topology, there are some very long paths (contains many hops), and the method in [57] tends to probe these paths more frequently than other paths. This means that some links are frequently, while others are rarely probed. Therefore, it takes many probes to cover all the links of interest. On the other hand, in our method, because we choose probing paths based on the probing overhead of the links, our method is not effected by the bias of the path lengths. More specially, as we can see in Fig. 4.8, the probe timings in the method in [57], the probe timings concentrates at the beginnings of two consecutive aggregation periods. On the other hand, because we adaptively change the intervals among probe timings, based on the number of remaining probing paths, as explained in section 4.4.1, the probe timings cover all over the aggregation periods. Therefore, our method can detect the faulty links much faster with smaller overhead.

From the simulation results, we found that the ratios between the traffic load of information



(a) AT&T topology (b) BA topology (c) Waxman topology

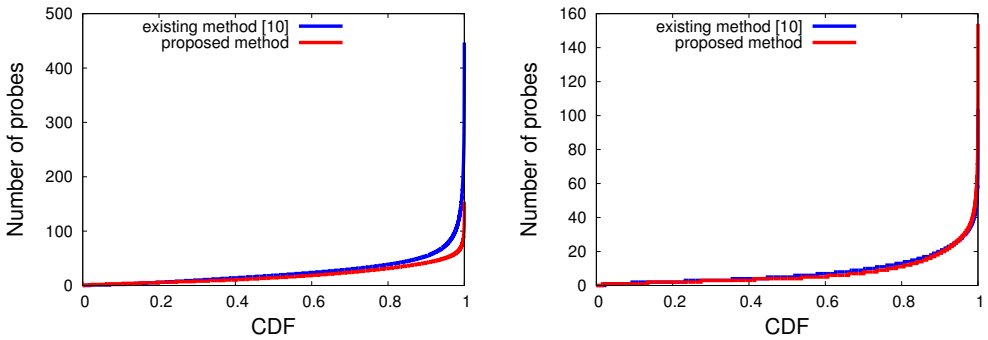
Figure 4.6: Total traffic load over all the links in the network

exchange and the probing traffic load in AT&T, BA and Waxman topologies are 0.031, 0.014 and 0.013, respectively. Because the number of half overlapping paths and partial overlapping paths is the smallest in Waxman topologies and the largest in AT&T topology [26], we conclude that the traffic load of information exchange is proportional to the degree of overlapping status.

Finally, we can see in Fig. 4.6 that traffic load of information exchange in our method is larger than in the method in [57]. This means that by shifting some amount of overhead from probing to information exchange, we can significantly reduce the detection time of faulty links.

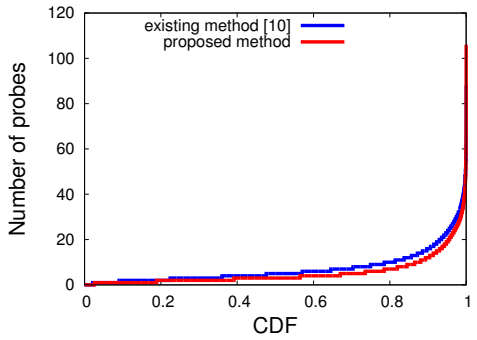
Fault localization

Table 4.1 shows the number of probe paths to locate the faulty links. As we can see, the number of probe paths in our method and the method in [57] are very small and almost the same. This result is obvious, because the number of hops and segments in these topologies is very small, as shown in Tab. 4.2. We believe that in a network with much longer paths, which contains many path segments, our method will surpass the method in [57].



(a) AT&T topology

(b) BA topology

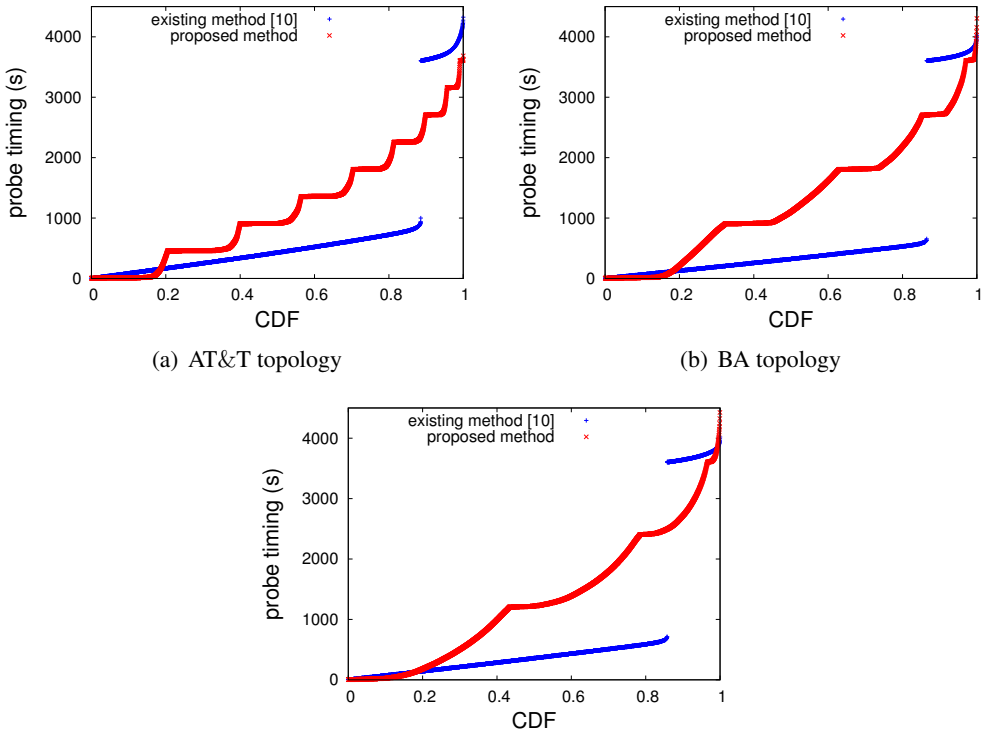


(c) Waxman topology

Figure 4.7: Distribution of probes per link

4.6 Summary

In this chapter, we introduce a comprehensive solution with both spatial and temporal aspects: we not only propose a method for effectively selecting monitoring paths, but also provide a method for adaptively determining the probing timings of selected monitoring paths to rapidly detect the faulty network components. Our solution utilizes a distributed approach: neighboring monitoring agents exchange route information to detect overlapping paths and share the probing results of overlapping paths to reduce the number of probing paths, and adaptively decide the probing timings based on the number of remaining probing paths so that the probing timings can cover large range of time. Our simulation results show that the proposed methods can detect faulty components much faster than the existing method when the probing overhead are equal.



(a) AT&T topology

(b) BA topology

(c) Waxman topology

Figure 4.8: Distribution of probe timing (s)

Method \ Topology	AT&T	BA	Waxman
Existing method	2.6552	2.4842	2.6085
Proposed method	2.6567	2.4844	2.6086

Table 4.1: Total number of probes for locating faulty links

Topology	AT&T	BA	Waxman
# of path hops	7.020	5.475	5.966
# of segments	6.402	5.408	5.921

Table 4.2: Average number of path hops and segments

Chapter 5

Conclusion

In this thesis, we introduced three methods for measuring the quality of the end-to-end paths to support large-scale network systems. We focus on the methods for measuring the most important quality metrics including latency, loss rate, jitter, available bandwidth, and connectivity. All of the proposed methods are built in a distributed approach in which the end nodes exchange route information and measurement results to enhance measurement accuracy and reduce measurement traffic load.

We first proposed a method for measuring the additive quality metrics, such as latency, loss rate and jitter. The basic idea is that the end nodes exchange measurement results of the overlapping segments between end-to-end paths, and use statistical processing to improve the measurement accuracy of these segments, thus consequently improve the measurement accuracy of the whole path. The measurement result of a segment can be obtained by sending probe traffics to the two end nodes or routers of the segment. Simulation results show that the relative error in the measurement results of our method can be decreased by half compared with the existing method when the total measurement overheads of both methods are equal.

We next produced a method for measuring the end-to-end available bandwidth, which is important for applications that require the transmission of large data, such as storage area networks and P2P networks. The proposed method aims at reducing measurement time and traffic of each measurement, thus can reduce the measurement conflict, and consequently enhance measurement

accuracy. To achieve this goal, in the proposed method, the end nodes share the measurement results of overlapping paths to configure parameter settings for available bandwidth measurements. Simulation results suggest that the relative errors in the measurement results of our method are approximately only 65% of those of the existing method. Furthermore, the measurement accuracy of our method remains better than the existing method when the total measurement traffic loads of both methods are equal.

We also introduced a method for diagnosing failures, which can help the distributed network systems to rapidly detect and bypass the problematic network segments. We proposed two algorithms that not only can rapidly detect network failures and locate faulty components, but also can reduce the total measurement traffic and well balance the traffic between the links of the network. Similar to our previous researches, we also utilize information exchange of measurement results of overlapping paths to reduce the measurement traffic. In the fault detection phase, we set constrains of measurement traffic for each link of the network, and probe the paths that satisfy the constrains with maximum measurement efficiency. On the other hand, in the fault localization phase, we choose the paths for measurements so that the expectation value of suspected faulty components after probing the paths is minimum. Simulation results show that our method can detect failures much faster than existing method while keeping the measurement traffic well-distributed between the links.

Through the researches, we have confirmed that exchanging measurement results contributes more to the enhancement of measurement accuracy than performing measurements. Compared to directly conducting measurements, exchanging measurement results not only requires small overhead, but also has no concern about conflicts. The idea of reusing measurement results of overlapping segments in our research is also utilized in other approaches of the literature, such as network tomography. However, in the network tomography, the measurement results of each overlapping segment are basically obtained by one end node and are used to calculate measurement results of multiple paths traversing the segment, while in our approach, they are measured and shared by multiple end nodes. Therefore, our approach can avoid some biases causing by one measurement agent, and has larger chance to obtain accurate measurement results. We believe that this idea can also be applied to other research areas, which have the same problem of resource competition, such

as wireless networks.

Bibliography

[1] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. SOSP 2001*, Oct. 2001.

[2] T. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proc. IEEE INFOCOM 2002*, Jun. 2002, pp. 170–179.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. the ACM SIGCOMM 2004*, Aug. 2004, pp. 15–26.

[4] Y. Chen, D. Bindel, H. Song, and R. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *Proc. ACM SIGCOMM 2004*, Aug. 2004.

[5] Y. Gu, G. Jiang, V. Singh, and Y. Zhang, "Optimal probing for unicast network delay tomography," in *Proc. IEEE INFOCOM 2010*, Mar. 2010.

[6] A. Krishnamurthy and A. Singh, "Robust multi-source network tomography using selective probes," in *Proc. IEEE INFOCOM 2012*, Mar. 2012.

[7] N. Hu and P. Steenkiste, "Exploiting internet route sharing for large scale available bandwidth estimation," in *Proc. IMC 2005*, Oct. 2005.

[8] C. Xing, M. Chen, and L. Yang, "Predicting available bandwidth of internet path with ultra metric space-based approaches," in *Proc. IEEE GLOBECOM 2009*, 2009, pp. 1–6.

[9] V. Ramasubramanian, D. Malkhi, F. Kuhn, M. Balakrishnan, A. Gupta, and A. Akella, "On the treeness of internet latency and bandwidth," in *Proc. ACM SIGMETRICS 2009*, 2009, pp. 61–72.

[10] S. Song, P. Keleher, B. Bhattacharjee, and A. Sussman, "Decentralized, accurate, and low-cost network bandwidth prediction," in *Proc. IEEE INFOCOM 2011*, 2011, pp. 6–10.

[11] P. Calyam, C. Lee, E. Ekici, M. Haffner, and N. Howes, "Orchestration of network-wide active measurements for supporting distributed computing applications," *IEEE Transactions on Computers*, vol. 56, no. 12, pp. 1629 –1642, Dec. 2007.

[12] M. Fraiwan and G. Manimaran, "Scheduling algorithms for conducting conflict-free measurements in overlay networks," *Computer Networks*, vol. 52, pp. 2819–2830, 2008.

[13] Z. Qin, R. Rojas-Cessa, and N. Ansari, "Task-execution scheduling schemes for network measurement and monitoring," *Computer Communications*, vol. 33, no. 2, pp. 124 – 135, 2010.

[14] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proc. ACM SIGCOMM 2002*, aug 2002.

[15] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456 – 1471, Oct. 2002.

[16] Skype Home Page, available at http://www.skype.com.

[17] KaZaA Home Page, available at http://www.kazaa.com.

[18] BitTorrent Home Page, available at http://www.bittorrent.com.

[19] Akamai Home Page, available at http://www.akamai.com.

[20] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862 – 876, 2010.

[21] J. Rubio-Loyola, A. Galis, A. Astorga, J. Serrat, L. Lefevre, A. Fischer, A. Paler, and H. Meer, "Scalable service deployment on software-defined networks," *IEEE Communications Magazine*, vol. 49, no. 12, pp. 84 –93, Dec. 2011.

[22] A. Nakao, L. Peterson, and A. Bavier, "Scalable routing overlay networks," *ACM SIGOPS Operating Systems Review*, vol. 40, pp. 49–61, Jan. 2006.

[23] C. Tang and P. McKinley, "On the cost-quality tradeoff in topology-aware overlay path probing," in *Proc. ICNP 2003*, Nov. 2003.

[24] C. L. T. Man, G. Hasegawa, and M. Murata, "Monitoring overlay path bandwidth using an inline measurement technique," *IARIA International Journal on Advances in Systems and Measurements*, vol. 1, no. 1, pp. 50–60, 2008.

[25] D. Ghita, H. Nguyen, M. Kurant, K. Argyraki, and P. Thiran, "Netscope: Practical network loss tomography," in *Proc. IEEE INFOCOM 2010*, Mar. 2010.

[26] D. T. Hoang, G. Hasegawa, and M. Murata, "A distributed measurement method for reducing measurement conflict frequency in overlay networks," in *Proc. IEEE CQR 2011*, May 2011, pp. 1–6.

[27] G. Hasegawa and M. Murata, "Scalable and density-aware measurement strategies for overlay networks," in *Proc. ICIMP 2009*, May 2009, pp. 21–26.

[28] N. Spring, R. Mahajan, and C. Wetherall, "Measuring isp topologies with rocketfuel," in *Proc. ACM SIGCOMM 2002*, Jan. 2002.

[29] A. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, Oct. 1999.

[30] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, Dec. 1988.

[31] BRITE: Boston university Representative Internet Topology gEnerator, available at http://www.cs.bu.edu/brite/index.html.

[32] G. Hasegawa and M. Murata, "Accuracy evaluation of spatial composition of measurement results in overlay networks (in Japanese)," *IEICE technical report*, vol. 110, no. 39, pp. 1–6, May 2010.

[33] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-peer video-on-demand system: Design and evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1706–1716, 2007.

[34] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 1, pp. 313–324, Jun. 2008.

[35] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth," in *Proc. PAM 2002*, 2002, pp. 14–25.

[36] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proc. ACM SIGCOMM 2003*, 2003, pp. 39–44.

[37] T. H. Dinh, G. Hasegawa, and M. Murata, "A low-cost, distributed and conflict-aware measurement method for overlay network services utilizing local information exchange," *IEICE Transactions on Communications*, vol. E96-B, no. 2, pp. 459–469, Feb. 2013.

[38] V. J.Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell, "pathchirp: Efficient available bandwidth estimation for network paths," in *Proc. PAM 2003*, 2003.

[39] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879 – 894, aug. 2003.

[40] C. D. Guerrero and M. A. Labrador, "On the applicability of available bandwidth estimation techniques and tools," *Computer Communications*, vol. 33, no. 1, pp. 11–22, 2010.

[41] Y. Tang, E. Al-Shaer, and R. Boutaba, "Efficient fault diagnosis using incremental alarm correlation and active investigation for internet and overlay networks," *IEEE Transactions on Network and Service Management*, vol. 5, no. 1, pp. 36–49, 2008.

[42] Y. Tang, E. Al-Shaer, and K. Joshi, "Reasoning under uncertainty for overlay fault diagnosis," *IEEE Transactions on Network and Service Management*, vol. 9, no. 1, pp. 34–47, 2012.

[43] F. Gillani, E. Al-Shaer, M. Ammar, and M. Demirci, "Fine-grain diagnosis of overlay performance anomalies using end-point network experiences," in *Proc. CNSM 2012*, 2012, pp. 91–99.

[44] M. Brodie, I. Rish, and S. Ma, "Optimizing probe selection for fault localization," *Distributed Syst. Oper. Manage*, 2001.

[45] I. Rish, M. Brodie, and S. Ma, "Optimizing probe selection for fault localization," *IBM Systems Journal*, 2001.

[46] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik, "Real-time problem determination in distributed systems using active probing," in *Proc. IEEE/IFIP NOMS 2004*, vol. 1, 2004, pp. 133–146.

[47] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez, "Adaptive diagnosis in distributed systems," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1088–1109, 2005.

[48] Y. Tang, E. Al-Shaer, and R. Boutaba, "Active integrated fault localization in communication networks," in *Proc. IM 2005*, 2005, pp. 543–556.

[49] M. Natu and A. S. Sethi, "Active probing approach for fault localization in computer networks," in *the 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, 2006, pp. 25–33.

[50] H. X. Nguyen and P. Thiran, "Active measurement for multiple link failures diagnosis in ip networks," in *Proc. PAM 2005*. Springer, 2005, pp. 185–194.

[51] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in ip networks," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 1092–1103, 2006.

[52] H. X. Nguyen and P. Thiran, "The boolean solution to the congested ip link location problem: Theory and practice," in *Proc. INFOCOM 2007*, 2007, pp. 2117–2125.

[53] M. Natu and A. Sethi, "Probabilistic fault diagnosis using adaptive probing," in *Managing Virtualization of Networks and Services*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4785, pp. 38–49.

[54] M. Natu and A. S. Sethi, "Probe station placement for robust monitoring of networks," *Journal of Network and Systems Management*, vol. 16, no. 4, pp. 351–374, 2008.

[55] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5373–5388, 2006.

[56] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *Proc. ACM CoNEXT 2007*, 2007, p. 18.

[57] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," in *Proc. INFOCOM 2009*, 2009, pp. 1377–1385.

[58] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, 2006, pp. 219–230.

[59] J. Ni, H. Xie, S. Tatikonda, and Y. R. Yang, "Network routing topology inference from end-to-end measurements," in *Proc. INFOCOM 2008*, 2008, pp. 36–40.

[60] J. Ni and S. Tatikonda, "Network tomography based on additive metrics," *IEEE Transactions on Information Theory*, vol. 57, no. 12, pp. 7798–7809, 2011.

[61] S. Zarifzadeh, M. Gowdagere, and C. Dovrolis, "Range tomography: combining the practicality of boolean tomography with the resolution of analog tomography," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, 2012, pp. 385–398.

[62] D. S. Johnson, "Approximation algorithms for combinatorial problems," *Journal of Computer and System Sciences*, vol. 9, pp. 256–278, 1974.

[63] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.

[64] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer, 2000.