2008

Overlay networks monitoring

Mohammad Fraiwan
Iowa State University

Overlay networks monitoring

by

Mohammad Amin Fraiwan

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Manimaran Govindarasu, Major Professor
Ahmed E. Kamal
Douglas W. Jacobson
Srikanta Tirthapura
Sivalingam Sritharan

Iowa State University

Ames, Iowa

2008

DEDICATION

To my mother and father for their love and support. All is due to their hard work and sacrifice...

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

All praise is due to ALLAH, the most merciful and the most compassionate, for bestowing this Ph.D. degree among other endless blessings on me. Along the journey to make this dissertation, many people came to my assistance and support, and offered their love and compassion.

First and foremost, I would like to express my deep gratitude, appreciation, and thanks to Dr. Manimaran Govindarasu for his valuable assistance in pursuing my Ph.D. degree. Dr. Manimaran has guided me to this achievement with patience, vigor, research insight, and kind advice both professionally and personally. Without his help and advice, this dissertation could not have been possible.

Also, I would like to thank my Ph.D. committee members, Professors Ahmed Kamal, Doug Jackbson, Tom Daniels, Srikanta Tirthapura, and Sivalingam Sritharan. They provided me with very helpful comments and feedback on my research proposal and dissertation, which greatly elevated the quality of my work and opened up future research directions for me.

As a member of the real-time systems and networking laboratory and the department of electrical and computer engineering , I benefited from the discussions, paper reviews, and presentations with my colleagues and friends. I would like to extend my gratitude to my lab mates Basheer Al-Duwairi, Anil Kumar, Muthuprasanna, Kayal, Srdjan Pudar, Naeem Odat, and Yoga Mahesh. Also, my deepest thanks go to my friends Jamal Al-Karaki, Bashsar Gharaibeh, Hisham Almasaeid, Osameh Al-Kofahi, Ashraf Hamad, and Mohammad Saleh. In addition, I express my best wishes and appreciation to the

wonderful ladies at the department's main and graduate offices: Sara, Laurie, Karen, Dana, Jess, Pam, Kara, Jean, and Ginny.

In addition, I would like to acknowledge and thank my loving friends in the Ames and ISU Community. In particular, the Dar Al-Arqum Mosque family, Mohammad Al-saqer, Nimer Mehyar, Ehab Abu Basha, Khalid Boushaba, Janni Abbasi, Mohammad Mujeeb, the Sermit's family, and so many others that supported and put up with me all these years.

Also, I would like to express my warm love, appreciation, and best wishes to the smart, beautiful, and charming Carly Riecken for her support and love.

Finally, I am full of pride and gratitude to my parents, sisters, and brother for their love, effort, endless support, and help throughout my life.

ABSTRACT

The phenomenal growth of the Internet and its entry into many aspects of daily life has led to a great dependency on its services. Multimedia and content distribution applications (e.g., video streaming, online gaming, VoIP) require Quality of Service (QoS) guarantees in terms of bandwidth, delay, loss, and jitter to maintain a certain level of performance. Moreover, E-commerce applications and retail websites are faced with increasing demand for better throughput and response time performance. The most practical way to realize such applications is through the use of overlay networks, which are logical networks that implement service and resource management functionalities at the application layer. Overlays offer better deployability, scalability, security, and resiliency properties than network layer based implementation of services.

Network monitoring and routing are among the most important issues in the design and operation of overlay networks. Accurate monitoring of QoS parameters is a challenging problem due to: (i) unbounded link stress in the underlying IP network, and (ii) the conflict in measurements caused by spatial and temporal overlap among measurement tasks. In this context, the focus of this dissertation is on the design and evaluation of efficient QoS monitoring and fault location algorithms using overlay networks.

First, the issue of monitoring accuracy provided by multiple concurrent active measurements is studied on a large-scale overlay test-bed (PlanetLab), the factors affecting the accuracy are identified, and the measurement conflict problem is introduced. Then, the problem of conducting conflict-free measurements is formulated as a scheduling problem of real-time tasks, its complexity is proven to be NP-hard, and efficient heuristic

algorithms for the problem are proposed. Second, an algorithm for minimizing monitoring overhead while controlling the IP link stress is proposed. Finally, the use of overlay monitoring to locate IP links' faults is investigated. Specifically, the problem of designing an overlay network for verifying the location of IP links' faults, under cost and link stress constraints, is formulated as an integer generalized flow problem, and its complexity is proven to be NP-hard. An optimal polynomial time algorithm for the relaxed problem (relaxed link stress constraints) is proposed.

A combination of simulation and experimental studies using real-life measurement tools and Internet topologies of major ISP networks is conducted to evaluate the proposed algorithms. The studies show that the proposed algorithms significantly improve the accuracy and link stress of overlay monitoring, while incurring low overheads. The evaluation of fault location algorithms show that fast and highly accurate verification of faults can be achieved using overlay monitoring. In conclusion, the holistic view taken and the solutions developed for network monitoring provide a comprehensive framework for the design, operation, and evolution of overlay networks.

CHAPTER 1. INTRODUCTION

Recent years have seen a tremendous growth in the number and quality of Internet-based multimedia applications (e.g., online gaming, video streaming, VoIP, etc.). In addition, conventionally text-based Web pages are being replaced by multimedia-rich content. Deploying and delivering such applications require Quality of Service (QoS) support from the underlying network infrastructure. The typical QoS metrics are available bandwidth, latency, loss rate, and jitter. As long as the multimedia application conforms to its QoS requirements, the network is supposed to provide for accepted requests while maximizing the acceptance rate and revenue.

As new network applications and services emerge, new network functionalities are required and new operating challenges face the network infrastructure. The IP layer of the TCP/IP network model has, for many years, successfully provided the necessary routing service. However, adding new applications and services (e.g., multicast, monitoring) has led to many security threats, deployment complexities and extra cost. In addition, new functional requirements are emerging quickly (e.g., object lookup). Implementing such requirements into routers, which are serving diverse network communities, requires high management overhead, prone to errors, and not scalable. For example, IP multicast [1] has been deployed on only a very small portion of the routers in the Internet [2].

An alternative solution to these challenges allows for the implementation of the routing services, among other application-specific services (e.g., robust monitoring), at the end nodes or the hosts rather than the routers. The end nodes along with the end-to-end paths form an overlay network, which is a virtual abstraction of the underlying

IP-level network implemented at the application level. Using such an overlay, complex routing algorithm, security protocols, monitoring algorithms, and network applications can be deployed with great versatility and ease, because we do not have to change the underlying routers. Overlay nodes need to maintain up-to-date information regarding the QoS metrics of overlay paths to the other overlay nodes, in order to forward packets along the appropriate paths.

Now that the overlays have been established as a viable and feasible solution to the aforementioned challenges facing today's networks, the focus of the literature has moved into two fronts. The first one is overlay design (i.e., choosing overlay nodes and paths) to optimize the performance of the specific network application. Researchers have advocated that using the underlying IP-level topological information will greatly enhance the performance of overlays [20, 26, 27, 28, 35]. Thus, tools have been developed to infer the underlying IP-level information and make it available to trusted applications that require them [56]. Albeit, at the expense of the layered design principles. The second research thrust deals with network monitoring. It has been quickly realized that new applications can not completely rely on the paths chosen by IP-level routing [4] for two main reasons. First, IP-level routing optimizes path selection for a system-wide criteria (e.g., minimize maximum link utilization) and is often sub-optimal in terms of user performance (e.g., policy routing, etc.) [29]. Second, it has relatively slow convergence when faced with faults, congestion, and degraded performance. Also, sophisticated monitoring tools and protocols can be readily deployed at the overlay level.

Overlay monitoring protocols have generally focused on exploiting the overlay and IP-level topologies to achieve high monitoring accuracy at a low overhead [4, 5, 6, 10, 26, 35]. This is accomplished by considering all the overlay paths jointly rather than individually, and exploiting the overlap at the IP and overlay levels. These protocols have abstracted out the specific measurement tools used on each overlay path, and thus overlooked the possible interference among measurements being run concurrently on overlapping

paths. On the other hand, measurement tools deal with estimating the QoS properties of individual paths, and hence do not address the benefits (i.e., lower overhead) and problems (i.e., link stress) caused by conducting measurements on multiple overlapping paths.

1.1 Thesis Statement

A comprehensive understanding of the overlay monitoring problem is provided in this dissertation. Relevant elements that factor into the accuracy, overhead, and impact of the monitoring infrastructure are identified. By integrating overlay monitoring protocols and the information about measurement tools, the high accuracy of a single lone path measurement and the scalability and efficiency of the monitoring protocols is achieved. Also, by integrating IP-level topological information, the overlay link stress caused by the monitoring packets is bound and controlled. In addition, overlay monitoring information is used in network fault management. Specifically, IP links fault location.

1.2 Organization

The remainder of this dissertation is organized as follows.

- In chapter 2, the related work in the literature is reviewed. The chapter focuses on overlay monitoring protocols, bandwidth estimation tools, and alarm correlation and fault detection algorithms.

- In chapter 3, experiments are conducted to evaluate the extent of measurement conflict and the factors affecting the accuracy of measurement. The measurement conflict problem is formulated and its complexity is proven to be NP-hard. Polynomial time scheduling heuristic algorithms are proposed to allow measurement tasks

to run in a non-conflicting manner. In addition, simulation studies are conducted to evaluate the performance of the proposed algorithms.

- In chapter 4, a scheme that controls the links stress caused by overlay monitoring protocols is proposed. The goal is minimize the monitoring overhead, while controlling the number of duplicate measurement packets that cross a certain link (overlay or IP). Simulation studies are conducted to evaluate the performance of the proposed scheme.

- In Chapter 5, the problem of link stress constrained fault location using overlay measurements is formulated and its computational complexity is proven to be NP-Hard. A relaxed problem is identified and a flow network solution is proposed. Experimental studies on real-life Internet topologies are conducted to evaluate the performance of the proposed solution.

- In chapter 6, the dissertation is concluded and interesting research directions are identified.

CHAPTER 2. BACKGROUND AND RELATED WORK

In this chapter, some background into the work related to our dissertation contribution is provided. In section 2.1, an introduction into overlay networks is given and the scene is set for the overlay monitoring environment and the associated problems. Some overlay networks applications are surveyed in section 2.2, as it is necessary to show the importance of the monitoring problem, its performance and economical impacts, and its pervasiveness into many aspects of the end-user's applications. In section 2.3, some of available bandwidth estimation tools and algorithms are discussed. These tools are used at a network-wide level by the overlay monitoring protocols, which will be discussed in section 2.4. Finally, some background into the fault management literature is given in section 2.5.

2.1 What is an overlay network?

An overlay network is a virtual abstraction of the underlying IP-layer network, see Fig 2.1. It can be represented as a graph $G_o = (V_o, E_o)$, where V_o is the set of all overlay nodes, and E_o is the set of overlay edges between the overlay nodes that are constructed using the specific overlay application. IP-level routers and other end-nodes are abstracted away from the vertex set V_o, while IP links are left out of the set E_o. Such an abstraction drastically simplifies the network graph, as well as the related network algorithms, which is one of the attractive characteristics of overlay networks [3, 5, 26].

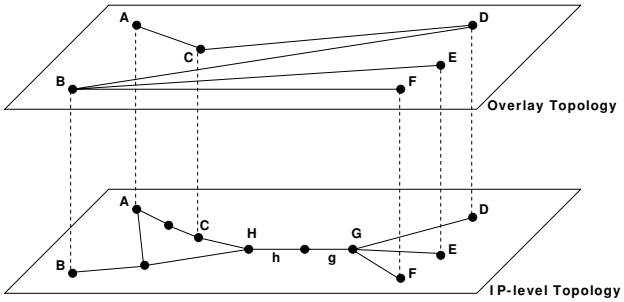


Figure 2.1 An example overlay network and the underlying IP-level topology.

Overlay networks have come to play an increasingly diverse role in today's network applications. Such applications include end-system multicast [5], peer-to-peer systems [6, 10], routing for high reliability [4, 20], storage and lookup systems (e.g., Akamai [21]), and security [24]. Monitoring overlays are also becoming more common, Internet service providers (ISPs) are deploying monitoring tools at specific nodes in their networks, as part of a Network Measurement Infrastructure (NMI) [25]. Overlays allow designer to implement and deploy their algorithms, applications, and services with great flexibility and versatility. However, maintaining the efficient and correct operation of these overlays requires regular probing of overlay links to measure available bandwidth, delay, and loss rate. Network monitoring is an important infrastructure service that helps in tracking network performance, security, and fault management.

One of the paramount issues to the overlay application is the construction of the overlay topology [3]. The topology consists of the set of the overlay nodes and the collection of direct or indirect neighbors. The choice of the overlay network topology, or the choice of neighbors at each node, has a significant impact on systems performance. For example, suboptimal multicast trees may result in high link stress and high relative delay penalty (RDP) [26], while a poor quality backup path will affect the fault tolerance of that application. In order to construct an efficient overlay topology, applications need

to carefully select paths between overlay node pairs. These paths need to be monitored to keep an up-to-date view of their quality.

To put these problems in perspective, and to design systematic and practical solutions, this dissertation envisage an overlay monitoring framework comprised of the following elements, see Fig. 2.2:

- `Monitoring requirements:` The information flow starts by identifying the monitoring requirements, these requirements are either specified by the user (i.e., network administrator), or the application. Realizing these requirements can be accomplished either by network tomography algorithms, or directly through measurement tools.

- `Network tomography:` The network tomography algorithms identify the most suitable (e.g., result in least overhead) paths and trees that achieve the monitoring goals, and process the intermediate monitoring results into final results that fulfill the user requirements. Traditionally, there has been a gap between choosing the paths to be monitored on one hand, and how the measurements are actually conducted on the other.

- `Measurement tasks generation:` Given a network topology, a monitoring algorithm, and a set of measurement tools, we are supposed to be able to generate measurement tasks based on this and possibly more information. Several issues and research tasks can be thought of in this context; specifically, the choice of the monitoring frequency of each task. Many factors affect this choice as given below.

  - The measurement overhead. Typically it is desirable to limit the measurement overhead to a certain amount. Also, how will this overhead be distributed among tasks?

  - Assigning frequency to the tasks based on the stability of the path through which this task will be run. Stable paths should be monitored less frequently.
  - Inter-AS (Autonomous System) monitoring traffic is usually bounded by contractual and managerial constraints (i.e., Service Level Agreements (SLA)).
  - The schedulability of the task set. What if the task set is not schedulable? How to modify the frequency, and for which tasks?

- `Measurement Scheduling:` The previous framework elements provide the input to the practical solution that needs implementation. As we will see later on in this dissertation, several problems arise in practice. Measurement scheduling presents a time dimension solution to the measurement conflict problem that is caused by overlap in the time and space dimensions (by space, we mean the network links).

- `Spatial processing:` In the case where time is not enough to solve the problems caused by the overlap in time and/or space dimensions, we can process the set of tasks so that they are schedulable or to produce low overhead, as we will see later in chapter 3.

- `Measurement tools:` These tools of choice conduct the actual measurements. However, they are not the subject of this dissertation. But, the nature of these tools and how they interact with each other is important in a network, where multiple tools are running on overlapping links and time periods.

- `Distributed coordination:` The monitoring framework will not be complete without the inclusion of a distributed coordination and communication module. The purpose of such module will be to communicate the monitoring requirements to the appropriate nodes, gather, aggregate, and redistribute monitoring results, in a scalable and timely fashion. Many algorithms have been proposed in the literature [27, 28]. This subject is out of the scope of this dissertation.

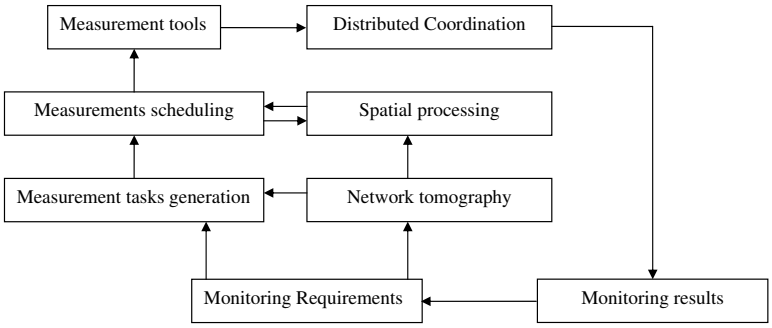


Figure 2.2 The various components of a monitoring framework.

- `Monitoring results:` The monitoring results may affect the monitoring requirements for subsequent monitoring cycles. For example, if the network conditions fluctuate rapidly then we may wish to increase the monitoring frequency, or reduce the frequency when the network is stable for long periods. More importantly, they will drive the overlay routing decisions and the topology of the overlay network.

2.2 Overlay Network Applications

In this dissertation, overlay network applications refer to those network services that run on top of an existing infrastructure and provide additional functionalities such as end-node routing. Since we address overlay monitoring, which is vital to many types of overlay applications, it is important to point out the variety of those applications and how they make use of the monitoring information to optimize their performance or even deliver an acceptable quality to the end user. Several categories of overlay applications are discussed; including content distribution and dissemination networks, and Peer-to-Peer system.

Content Distribution

Depending on the content type being delivered, the overlay applications under this category will require a varying set of QoS requirements. For example, a video conferencing application will require strict available bandwidth, latency, loss rate, and jitter requirements. Deviation from these requirements will greatly affect the perceived quality at the end-user. Another major example is e-commerce. Studies have shown that although the speed of Internet connections may be improving for an increasing number of end-users, the amount of time that they are willing to wait on a retailer's web page to load is decreasing. It has been reported that the average retail web page response time -for the shopper to abort- was 8 seconds about a decade ago, but now it is less than 4 seconds [21]. Hence, network performance greatly affects costumer satisfaction and revenues for the retailers using the content provider's overlay network. Another important overlay application is the publish-subscribe system, which requires secure and timely delivery.

Various enhancements, over the basic IP-level packet delivery, are possible only through extensive monitoring. For example, to increase reliability, multiple duplicate packets can be sent from one overlay node to the other along the source-destination path with different duplication level (i.e., number of duplicate packets) between each pair of overlay hops [21, 30]. In this case, monitoring will provide a map of available overlay paths and their QoS parameters (e.g., reliability) to choose the duplication level and the paths to send packets over. In another example, monitoring can help content providers achieve load balancing and reduce congestion by monitoring the available bandwidth on their overlay paths and direct traffic accordingly.

The overlay networks literature has witnessed the devolvement of a large number of applications, protocols, and algorithms that require monitoring information as input to optimize their performance. Example of these applications are: RON [4], Narada

[5], NICE [11], HyperCast [12], Overcast [13], ScatterCast [14], OMNI [15], TAG [27], Akamai [21], CoDeeN [63], BitTorrent [16], Globule [17], and Herald [18].

Peer-to-peer Systems

The traditional client-server communication model stores all the data on a centralized server or a set of servers, and the clients request data from the server. Such a model has been facing a lot of scalability problems, presents a single point of failure and a network resource bottleneck close to the server, and has high management overhead [3]. To counter these problems, the peer-to-peer communication model has been proposed. In this model, all the nodes are equivalent it functionality, with little or no centralized control or management overhead [6]. Peer-to-peer systems has witnessed a tremendous amount of research in object location and search, structuring, and distributed hashing [7].

A peer-to-peer system distributes objects (e.g., music files) and their indices across the network [6, 9, 10]. Multiple copies of the same object and its index may exist in the network, thus, adding more fault tolerance and reliability. Each peer contributes its resources (i.e., storage, computation, and bandwidth) and content to the whole system, hence, increasing scalability.

The cost of distributing objects and their indices is that they will be harder to locate. Peer nodes are expected to receive queries searching of a certain object or content, and based on that object or a search phrase determine the location of this object or the next peer to ask. Overlay networks can be used in solving this problem by providing complex functionalities like Distributed Hash Tables (DHT) that allow requests to be routed and redirected based on the object name, index, or content rather than the IP address where the object resides.

2.3 Estimating QoS properties of End-to-End Paths

As we have seen earlier, several QoS parameters are of interest to overlay applications (e.g., loss rate, bandwidth, etc). However, measuring or estimating these parameters turns out to be a challenging problem. Bandwidth estimation is one of the most important parameters, as it is the most difficult to obtain and it is necessary for many traffic engineering algorithms, overlay multicast, and many other network applications [19]. Understanding how these tools work in principle is essential to appreciate their interactions and effects on each other's accuracy when run at overlapping times and paths.

There are three categories of bandwidth measurement and estimation tools, as follows [47]:

- `Capacity`: refers to the maximum possible bandwidth that a link can deliver.
- `Available bandwidth`: refers to the maximum unused bandwidth of a link or path.
- `Bulk Transfer Capacity`: refers to the achievable throughput of a bulk-transfer connection using UDP or TCP.

In this section, we will discuss the later two categories as they are the most important for overlay applications.

2.3.1 Available Bandwidth Measurement

The available bandwidth of a link or path is the unused capacity of that link or path at the time of measurement. Clearly, available bandwidth depends on the traffic load at the time of measurement and thus it is a time varying metric. A number of techniques exist in the literature for measuring the available bandwidth of the path of interest [39, 40, 42, 43, 44, 59, 60, 61]. Some of these are [47]: Variable Packet

size (VPS) probing and its tailgating variation, Packet Pair/Train Dispersion (PPTD) probing, Self-Loading Periodic Streams (SLoPS), and Trains of Packet Pairs (TOPP). The majority of the available bandwidth measurement tools inject packet pairs or trains of packet pairs into the path of interest with predetermined gaps between each packet pair. They differ in the size of each packet in the pair, the rate at which the pairs/trains are sent, determining the time gap between the packet pair and how it is varied at the sender, and the way they estimate the available bandwidth based on the changes in the time gap separating the pair of packets at the receiver. We go into the details of one of the most commonly used measurement methods; Packet Pair/Train Dispersion (PPTD) [48].

In PPTD, the source sends multiple equally sized packet pairs into the path of interest. Each pair consists of two packets sent back to back with pre-determined rate, which determines the intra-pair time dispersion (i.e., the time difference between the last bit of each packet). Figure 2.3 shows how the intra-pair spacing changes based on the link's capacity. If the probe packet is of size S bytes and initial inter-packet dispersion $\Delta_{i_{in}}$, and the capacity of link i is C_i, then $\Delta_{i_{out}}$, the dispersion of the packet pair at link i, is given by:

$$\Delta_{i_{out}} = \max(\Delta_{i_{in}}, \frac{S}{C_i}) \tag{2.1}$$

Assuming no cross traffic and an N-hop path, then the dispersion Δ_R that the receiver will measure is given by [47]:

$$\Delta_R = \max_{i=0,\ldots,N}(\frac{S}{C_i}) = \frac{S}{\min_{i=0,\ldots,N}(C_i)} = \frac{S}{C} \tag{2.2}$$

After receiving the j^{th} train of packet pairs, the receiver will calculate the j^{th} estimate of the available bandwidth experienced over the entire path as, $e_j = \frac{S}{\Delta_{R_j}}$, where Δ_{R_j} is the mean of the intra-pair dispersion within the j^th received train. A major problem with this approach is that cross traffic can lead to an overestimation or underestimation

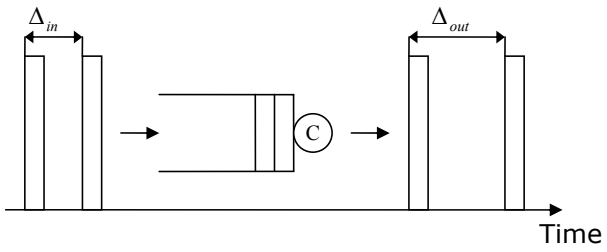


Figure 2.3 The dispersion of a packet pair going through a small capacity link.

of Δ_{R_j} depending on when the cross traffic packets are transmitted (i.e., before the first packet or in-between the packet pair) [39, 47].

Statistical methods (e.g., linear regression) can be used to filter out the erroneous estimates. However, it was shown that standard statistical methods do not always lead to correct estimation [47, 48]. In addition, the estimation accuracy also decreases as the number of congested links increases [39].

2.3.2 Bulk Transfer Capacity Measurement

The available bandwidth on a certain path does not necessarily mean that the end-user using this path will experience such a throughput. Thus it is necessary to measure the achievable throughput using the TCP/IP transport protocols (i.e., TCP or UDP). Tools under this category can be classified into TCP simulation tools (e.g., cap [60]) and path flooding tools (e.g., TReno [45], Iperf [59], Netperf [46]). In cap [60], the sender and the receiver exchange UDP data and acknowledgment packets that contain the information required to emulate TCP behavior. Flooding techniques, on the other hand, inject TCP/UDP packets into the network as fast as possible within a specific time period [59]. The user of the tools is given the freedom in choosing the specific transmission rates and the various TCP attributes (e.g., congestion window, and slow start threshold). For example, Iperf [59] using TCP and default settings, will repeatedly

inject sequences of 8 KB packets over a period of 10 seconds. Using UDP, the default bandwidth to send at is 1 Mbit/sec. Iperf allows the user to change these and other parameters (e.g., TCP window size, length of buffer to send or receive, and transmission duration), to achieve better measurement accuracy.

The reported bulk transfer capacity highly depends on a number of factors including the number of competing TCP flows, the transmission rate, the TCP parameters such as buffer and window size, the delay bandwidth product, the longevity of the TCP flow, the cross traffic, and many other factors [47].

The specific measurement tools are not the focus of this dissertation. The point to be made here has to do with the intrusive nature of these tools and their sensitive nature to the timing among successive probe packets. Some studies have developed queuing models for a packet pair sample being affected by the cross-traffic at a certain link [61]. Throughout this research the measurement tool is assumed to be a black box. Its computation and communication requirements are the only required input, in order to judge whether it affects and affected by other measurement tools.

2.4 Overlay Monitoring Protocols

Several studies in the literature have focused on deploying measurement tools on a large scale overlay-wide level. Although these tools had high accuracy and low overhead as their design objective, it has been quickly realized that more improvements are possible by utilizing the overlay and the underlying IP-level topologies. The main idea is to choose a small number of overlay links and paths as a probing set such that it is possible to infer or estimate the quality of the other unprobed paths. For these protocols to reduce the monitoring overhead, a significant number of overlay paths need to share the same set of IP links [26].

Monitoring Protocols

Some overlay applications (e.g., Chord [6] and Pastry [10]) introduced limited connectivity between overlay nodes as a way of reducing overhead. Albeit, it is possible that some good paths exist among overlay nodes, but they would be excluded from monitoring and thus from usage. Tang and Mckinley [26] have introduced the compromise between monitoring accuracy- as opposed to coverage- and overhead. Their rationale is that some applications (e.g., routing) may require bounded rather than completely accurate monitoring results. They achieve their goal by constructing an intermediate topology between the overlay and the IP-level topologies that consists of path segments (i.e., the longest subpath of an IP path that is not incident to any other IP or overlay paths leading to another node). Heuristics of the minimum set cover problem are then applied on the intermediate topology to find the minimum number of overlay links or paths to be probed. The quality of the unprobed paths is inferred from the bounds on the quality of their constituting segments. Additional unprobed overlay paths are chosen to refine the estimation accuracy. Still, high link stress have been reported by the authors. In another study [28], they formulated the problem of exchanging monitoring results as a link stress bounded overlay spanning tree.

Other methods have been proposed to select the probing set. Ozmutlu et al. formulated the constrained coverage problem to optimally select a subset of trace route probes to monitor the network [49]. They proposed the constrained coverage heuristic to predict the delay on the network paths as well as to identify anomalies. Chen et al. use algebraic method to choose a small subset of paths to be monitored and infer the loss rate and latency of the other paths [35]. They provided a method to update the probing set in face of topology changes and overlay nodes joining and leaving. However, their approach is not suitable for bandwidth measurements because it is a concave QoS metric.

To the best of our knowledge, no overlay monitoring protocol has tackled the interaction among the measurement probes, although some have noticed the problem and tried to reduce the overall overhead to solve the problem [49]. The same goes for the high link stress of these protocols. The goal of this dissertation is to consider time domain and space domain (i.e., topological) solutions to the measurement conflict and high link stress problems.

Monitoring Overlays and PlanetLab

PlanetLab is a globally distributed research overlay spanning over 917 nodes at 474 sites [58]. It provides a planetary-scale deployment platform for overlay applications and the supporting protocols and algorithms. One such overlay application is CoDeeN [63], which is a content distribution network running on top of PlanetLab. PlanetSeer [64], a monitoring overlay, has been developed to support CoDeeN functionalities. It works by combining passive and active measurement techniques to provide network-wide monitoring. More specifically, PlanetSeer uses passive monitoring of CoDeeN traffic to detect potential faults and anomalies in the network, while active measurements are used to verify whether the potential faults are real. PlanetSeer provides a relatively low overhead monitoring system, as it only initiates intrusive measurements when faults are detected from the large volume of the CoDeeN traffic.

2.5 Network Fault Management

Network fault management has gained an increasing momentum in the recent years as the communication systems continue to evolve and become more complex. The type of new network applications (e.g., video streaming) has imposed new requirements on fault management. Network faults need to be detected, located, and recovered from as fast as possible due to the severe financial ramifications of the loss of network services

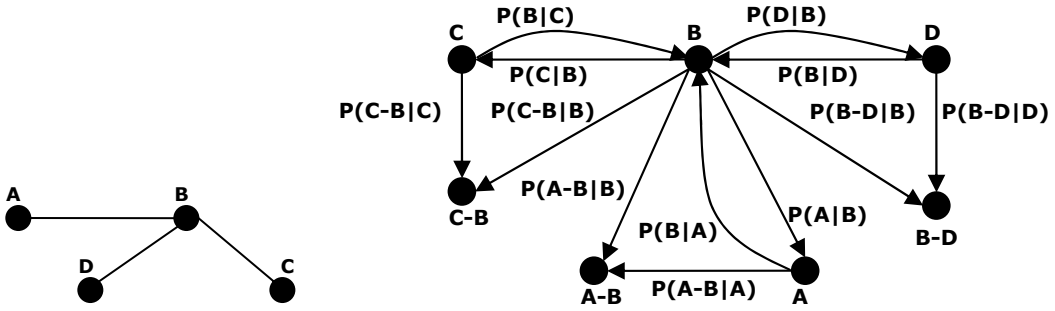
[21].

The process of fault management consists of three major aspects; fault detection, fault recovery, and fault localization [81]. Fault detection is the process of obtaining signs of the network service degradation. These are typically provided as alarms by the malfunctioning devices, protocol error messages, or simple customer complaints. Fault recovery aims at restoring the service or avoiding interruptions to the network connectivity [50]. Fault localization is the process of inferring the exact source of the fault from the set of observed alarms and symptoms. In this section, We will survey the later two aspects as they have witnessed the most research in the past decade.

Fault Localization and Alarm Correlation

Graph-theoretic techniques are one of the most commonly used in the literature. In these techniques, the system is modeled using a Fault Propagation Model (FPM) or a dependency graph, which describes symptoms and how they generate faults and affect each other. Observed symptoms are represented as FPM nodes. Two nodes share an edge if the failure of one node will cause the failure of the other node with a certain probability, see Fig 2.4. A deterministic model will have a probability of 1 on all edges. The problem boils down into analyzing the FPM to deduce the best possible explanation of the observed faults and symptoms [81]. It has been shown that this problem is NP-hard in general [82].

Heuristic algorithms have been proposed to solve this problem [81]. One such algorithm is based on the divide and conquer approach. The main idea is to cluster the dependency graph based on the maximum mutual dependency index [83], meaning that any intra-cluster edge label is larger than any inter-cluster label. The ultimate goal is to group all faults that are most dependent on each other in one cluster that explains the domain of observed alarms. This algorithm will always explain all the observed alarms, but may fail to give their most likely explanation in some cases [83]. Reference [81]



(a) An example network. (b) The corresponding dependency graph.

Figure 2.4 An example network and the corresponding dependency graph. Each $a \rightarrow b$ edge in the dependency graph has a label of the form $prob(a \text{ fails} \mid b \text{ fails})$.

discusses a number of fault localization methods.

Recently, Tang et al. [95] extended the dependency graph model to include a set of actions that can be used to best verify the most likely hypothesis or hypotheses. In another study [93], spatial correlation of the observed symptoms was used for detection of network black holes or silent failures.

Fault Recovery

The other aspect of the fault management literature to be discussed is the protection against faults and restoration of service. In protection schemes, backup paths are provisioned and discovered in advance. Upon detection of a failure, backup paths will be activated to route around failures as fast as possible (i.e., order of milliseconds). On the other hand, restoration schemes operate over a larger time scale and thus are more flexible in deciding the appropriate actions [50].

Depending on the amount of reserved resources, path protection schemes can be divided into the following categories [94].

- $1+1$ `Protection`: Packets are transmitted along the primary and backup paths.

The destination will choose the fastest transmission or the best signal.

- $1:1$ `Protection`. Resources are reserved along the backup path, but used only if the primary path fails.
- $1:N$ `Protection`. A single backup path is used to protect N primary paths. This scheme can not handle more than one path failure at a time.
- $M:N$ `Protection`. M backup paths are provisioned to protect N primary paths, where $1 \leq M \leq N$.

Each of these schemes represents a compromise between the cost, resource utilization, and the amount of protection needed.

A number of data plane recovery techniques have been proposed in the literature [50]. Mainly, 1+1 and M:N rings, disjoint paths, Protection Cycles, p-cycles, Redundant Trees, and IP Fast Reroute. We will not go into the details of those schemes in this dissertation as they are not essential for the dissertation contribution and are included here for the sake of completeness.

In this dissertation, the focus is on identifying the location of faults to enable the network administrator to bring those links or interfaces back online and operational. On the other hand, the focus of recovery techniques is continuing the transmission of packets in the face of failures (i.e., while they get fixed). In this sense, our goal and the fault recovery and protection schemes are complementary.

2.6 Discussion

The combined consideration of the overlay monitoring protocols and the measurement tools can significantly improve the monitoring accuracy. The reason is that the interaction among measurement tools running on overlapping overlay and physical paths

can lead to congestion delay, and possibly loss of measurement packets. Since the available bandwidth measurement tools are specifically sensitive to the timing between the measurement packets, it is imperative to study such an interaction. The same intuition applies to the other QoS metrics, which are affected by congestion and loss. Algorithms that significantly improve the measurement accuracy by scheduling conflicting measurement activities at non-overlapping time periods are proposed. Also, methods to reduce the link stress on highly shared paths segments, by optimizing monitoring protocols to avoid unnecessarily sending multiple measurement packets on the same set of links, are introduced. The former contribution can be viewed as a time domain solution to the conflict problem, while the later is a space domain (i.e., network topology) solution to the high link stress problem, with the two being related in that measurement conflict is caused by link stress, which in turn caused by overlapping measurement paths.

Finally, monitoring information available to the overlay are used to detect IP links faults by utilizing the underlay topological information and the availability of the overlay paths. Although obtaining the underling IP-level topological information incurs extra overhead, it has been shown that this information is available and stable for a reasonable amount of time [26]. Although, such a cross layer optimization methodology violates the network layering principles, an increasing number of overlay systems are using this information to improve performance [28].

CHAPTER 3. THE MEASUREMENT CONFLICT PROBLEM AND SOLUTION

In computer networks, the traditional measurement problem has typically focused on ways to measure a given path (or link) QoS parameters (e.g., bandwidth, delay, and jitter) to the best possible accuracy. Measurement methods are categorized as either active or passive. Passive measurements are non-intrusive, but have their limitations [51]. On the other hand, active measurements inject a non-negligible amount of probe traffic [43]. Such approaches have generally overlooked the limited capacities of the network links and the complex connectivity overlaps among different paths [25, 65]. Hence, causing concurrent measurements to compete for the network resources (*e.g.*, computation and communication resources), interfere with each other, and produce inaccurate results. In addition, Service Level Agreements (SLAs) usually limit the amount of monitoring traffic that can exist in the network at any point in time [25]. Thus, there is a need to coordinate simultaneous conflicting measurement activities in a way that the accuracy and timeliness properties are satisfied.

In this chapter, the measurement conflict problem is considered in the context of overlay networks. The proliferation of the Internet has led to many measurement-sensitive network applications. Such applications include overlay routing (*e.g.,* RON [4]), content distribution systems (*e.g.,* Akamai [21]), end-system multicast (e.g., Narada [5]), and security monitoring (*e.g.,* denial of service attacks detection). In addition, Internet Service Providers (ISPs) are deploying measurement overlays to monitor the health of

their networks.

A key aspect of the measurement conflict problem is the overlap among measurement paths. This overlap is an obvious observation at the IP-level of the network. However, measurements and the applications they support are typically deployed using overlays with specifically designed topologies on top of the IP-level network. The point to be noted here is that two seemingly disjoint overlay paths may actually be joined at the IP-level. This dependence may also change with time as the IP-level routing and the mapping to the overlay paths change.

Consider the example in Fig. 3.1, two measurement tasks T_1 and T_2, are being conducted concurrently. Task T_1 is running from source overlay node A to destination overlay node F, while task T_2 is running from source overlay node B to destination overlay node E. The overlay (or IP) routing has resulted in these two tasks sharing the overlay (or IP) link $C - D$. Now, depending on the bandwidth and computational requirements of each task and with sufficient synchronization, a conflict may occur wherein both tasks will inaccurately measure the bandwidth, loss, or latency of their respective paths due to their resource consumption at the shared link. Experiments conducted using PlanetLab [58] with various measurement tools [59, 42] confirm that such conflict does happen with a substantial effect on the measurement accuracy.

A globally distributed monitoring network have been constructed on top of PlanetLab. The network consisted of 25 nodes (15 in the USA, 5 in Europe, and 5 in Asia and Australia). Each node is programmed to conduct measurements every one hour, and later on, they are scheduled to conduct measurements in sufficiently non-overlapping time periods. The reported results use Iperf as the measurement tools, which reports the transfer throughput achievable using UDP or TCP. Experiments performed using other tools such as Pathload (measures available bandwidth), and H.323 beacon (measures support for Voice and Video over IP) results in drawing similar conclusions [25].

Fig. 3.2 shows the drastic effect that the conflict has on the accuracy of the measure-

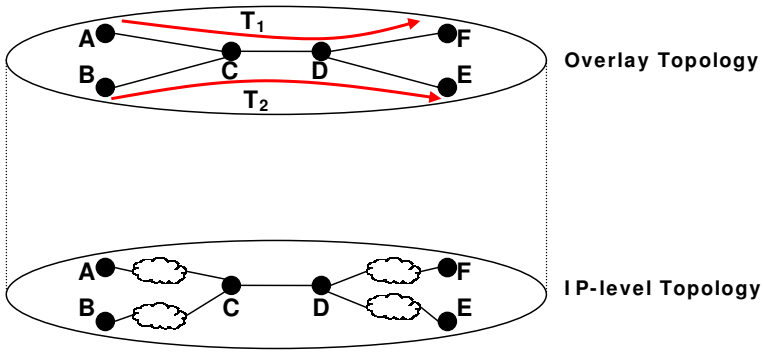


Figure 3.1 A measurement example, tasks T_1 and T_2 sharing overlay link $C - D$, which could have also been an IP link.

ment, as seen by any individual node. With 10 conflicting measurements, the reported throughput is only about $20 - 30\%$ of the reported value without conflict. The experiment had been repeated multiple interleaved times to ensure network dynamics are not the reason for the degradation. In addition, we have noticed the high stability of the PlanetLab paths, due probably to the Internet 2 connections.

The contributions of this chapter are as follows:

- The measurement conflict problem is formulated as a scheduling problem of periodic QoS real-time tasks, and the complexity of the problem is shown to be NP-hard [65].
- Two conflict-aware heuristic scheduling algorithms for uniform and no-uniform measurement tasks are proposed based on graph partitioning concepts [65].
- A topology-aware heuristic scheduling algorithm that increases the efficiency of producing feasible measurement schedules is proposed.
- The overlap among overlay paths, using various real-life Internet topologies of the two major service carriers in the U.S., is studied

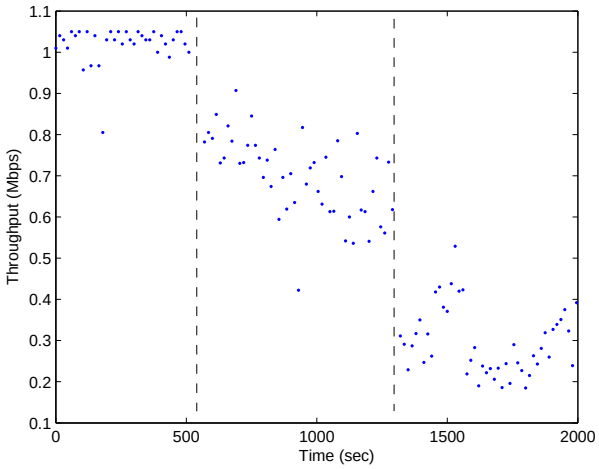


Figure 3.2 Degradation of measurement accuracy as the number of conflicting measurements is increased. We start with no conflicting tasks, then 5 conflicting tasks, and finally 10 tasks.

- The existence and the effect of measurement conflict is evaluated by constructing a globally distributed measurement overlay on top of PlanetLab and using various measurement tools.

The remainder of this chapter proceeds as follows. Section 3.1 presents our network monitoring model, its assumptions, and its computational complexity. Section 3.2 discusses the related work and motivation. Section 3.3 goes into the details of the conflict-aware scheduling algorithms. Section 3.4 presents the topology-aware scheduling approach and algorithm. We discuss some implementation issues in section 3.5. Section 3.6 presents our performance evaluation results. We conclude in section 3.7.

3.1 Network Monitoring Model

The network model is generalized to include overlay networks, as well as IP-level networks. As it has been mentioned earlier, network monitoring is considered part of a NMI (Network Measurements Infrastructure) employed by ISPs (Internet Service

Providers) at the IP-level network. While the rise of overlay networks and their varying application requires regular monitoring to achieve efficient and correct operation of these overlays.

Given an overlay network undirected graph $G_o = (V_o, E_o)$, where V_o is the set of overlay nodes and E_o is the set of overlay edges. The corresponding IP-level graph $G_{IP} = (V_{IP}, E_{IP})$ is also available, where V_{IP} is the set of physical nodes and E_{IP} is the set of physical edges. Note that $V_o \subseteq V_{IP}$, but $E_o \nsubseteq E_{IP}$ in general. Several previous studies have assumed and justified the knowledge of the underlying IP topology by the overlay network administrator [26].

Let $T = \{T_1, \ldots, T_n\}$ the set of tasks to be scheduled. $T_i = (s_i, d_i, c_i, p_i, tool_i)$, where s_i is the source of the measurement task, d_i is destination of the same task, c_i is the running time of the task, p_i is the period of task T_i, and $tool_i$ is the tool used by the task. The deadline of the task is the same as its period. We also define matrix M to be an $n \times n$ 0-1 matrix, representing the possible conflict between tasks if they are run at the same time, where $m_{ij} = 1$ if task i conflicts with task j, and 0 otherwise. The conflict matrix M captures the conflict among tasks based on the set of tasks and the computational and communicational conflict factors.

3.1.1 Problem Definition

We start by defining three important terms related to this problem:

- `Feasibility`: A feasible schedule is a schedule in which all tasks meet their deadlines and no two tasks are scheduled in a conflicting manner.

- `Optimality`: A scheduling algorithm is said to be optimal if no other algorithm can find a feasible schedule for a task set that this algorithm has failed to find a feasible schedule for.

- `Schedulability`: This defines the efficiency of the scheduling algorithm in terms of the ability to find a feasible schedule.

The measurement conflict scheduling problem can be defined as follows:

Given the set of tasks T and the conflict matrix M, find a feasible schedule for the task set.

We prove that this problem is NP-hard by reduction from Maximum Cardinality Independent Set [62], as follows:

Instance: A Graph $H = (W, F)$, where W is the set of vertices and F is the set of edges.

Question: Is there a subset $W' \subseteq W$ such that, for all $u, v \in W'$, $(u, v) \notin F$ and W' is of maximum cardinality? An independent set is said to be of maximum cardinality if it contains the largest possible number of vertices without destroying the independence property.

Theorem 1 *The optimal scheduling of measurement tasks is NP-hard.*

Proof: An instance of the problem is considered, where all the tasks have the same period and the same execution time (i.e., a uniform task set). An optimal scheduling policy will allow for the maximum concurrency of tasks, without violating the conflict constraint. Construct a task conflict graph $G = (V, E)$ as follows. Assign a node for each task in the set of tasks T, thus $|T| = |V|$. An edge is added between nodes $i, j \in V$ if the corresponding entry in the conflict matrix M, $m_{ij} = 1$.

Notice that finding a maximum independent set in the conflict graph will correspond to finding a maximum set of non-conflicting tasks that can execute at the same time. Finding the rest of the schedule can be done by repeatedly deleting the nodes in the previous maximum set and their incident edges, and finding the maximum independent set on the new graph. Thus, the problem is NP-hard. Note that the same proof can be obtained using the minimum graph coloring problem [62].

Although this problem sounds similar to the well known problem of offline single processor scheduling of real-time tasks, a major difference exists. The resource under consideration is the network on which probe conflicts occur. If the network is treated as a single processor system, then there is no parallelism in executing the tasks, this will lead to no conflicts, but poor schedulability. The problem is also different from the multiprocessor scheduling, because if the network is treated as a multiprocessor system, then the problem become identifying the processors that are active (i.e., the tasks that can execute concurrently) at any given time.

3.2 Related Work and Motivation

3.2.1 Related Work

Most of the research on network monitoring has focused on reducing the probing overhead [6, 26, 35]. Nonetheless, this reduction is a function of the number of nodes in the probing node set, with the best known algorithms having complexity of $O(n \log n)$. However, the constant factor in this function is high and depends on the type of measurement (e.g., bandwidth, loss, etc.) and other factors [85].

The measurement conflict problem was first introduced by Calyam et al. in [25]. They have observed such a problem while designing ActiveMon for the Third Frontier Network (TFN) project. ActiveMon is an NMI software framework to collect and analyze network-wide active measurements. In another study [55] they have developed a simple scripting language interface to specify various measurement requirements used in generating measurement timetables. Our work is different from theirs in that we showed the NP-hardness of the problem [65] and provided far superior alternative algorithms, while highlighting the availability of a performance bound. Also, we conduct experimental studies to show the existence and effect of the measurement conflict problem.

A token passing protocol has been used by related studies to minimize collisions

between probes [54], this protocol is used to generate time series of measurement data, which is then used in numerical forecasting models as part of a Network Weather Service. However, their approach does not allow for concurrent execution of multiple measurement tasks.

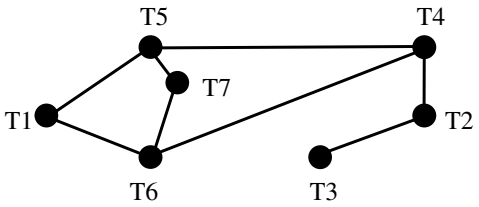
Periodic task scheduling is a well studied problem in the real-time scheduling literature. For example, EDF (Earliest Deadline First) scheduling is an optimal single processor scheduling algorithm [66]. As the name suggests, EDF scheduling gives higher priority to the task with the earliest deadline. In this study, we leverage some of the concepts used in real-time scheduling, without affecting the novelty of our approach.

3.2.2 Motivation

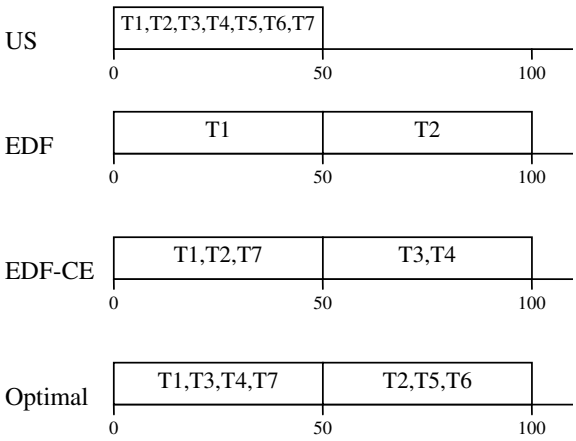
In this section, some motivational examples are given, where the shortcomings of existing solutions and the existence of a significant room for improvements, are shown. The algorithms to be considered are:

- `Unsynchronized scheduling (US)`: Tasks are run without regard to conflict. This algorithm achieves maximum schedulability, but with a lot of conflicts.
- `Non-preemptive EDF`: This algorithm will schedule tasks based on deadline, with higher priority given to earlier deadlines, with no regard to the benefits of concurrent execution. It permits no conflicts, but achieves worst schedulability.
- `EDF-CE`: EDF with Concurrent Execution, this is the algorithm proposed in [25]. Tasks are executed in EDF order, ready tasks are added randomly as long as they do not conflict with the currently executing tasks.

Example 1. Consider the following seven identical tasks, $T_i = (c_i, p_i) = (50, 100)$, $i = 1, 2, \ldots, 7$, where c_i is the execution time, and p_i is the period of task T_i. The conflict graph is shown in Fig. 3.3a. The schedule produced by the various algorithms is shown



(a) Task Conflict Graph

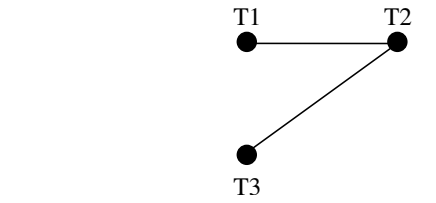


(b) The schedule produced by various algorithms.

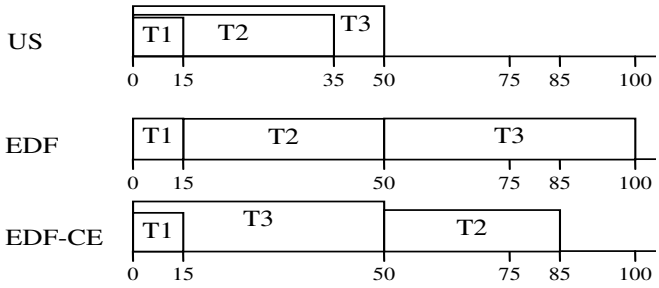
Figure 3.3 Example 1. Using Unsynchronized Scheduling (US) will schedule all the tasks, but EDF will schedule two tasks only, while 5 tasks will miss their deadlines. Tasks 5 and 6 will miss their deadlines under EDF-CE, while all tasks are scheduled with no conflict in the optimal solution.

3.3b. US and EDF provide us with the two extremes of the scheduling solution, the first achieves the least conflict, and the later achieves the best schedulability. EDF-CE on the other hand, breaks the deadline tie between tasks randomly. Thus, it gives no guarantees on the amount of parallelism. The optimal solution will find the largest possible set of non-conflicting tasks to be scheduled at any time. This feature is particularly important with larger, more complicated conflict graphs.

In the previous example, the task set was uniform (*i.e.,* tasks have the same period and the same execution time). However, non-uniform task sets may need additional attention.



(a) Task Conflict Graph



(b) The schedule produced by EDF-CE.

Figure 3.4 Example 2. Under EDF-CE, task 2 will miss its deadline due to a scheduling anomaly, and the algorithm aborts.

Example 2. Consider the following set of three non-uniform tasks. $T_1 = (15, 50)$, $T_2 = (35, 75)$, $T_3 = (50, 100)$. The conflict graph is given in Fig. 3.4a. This example shows another interesting fact. When faced with a non-uniform task set, blindly trying to increase task parallelism may lead to a scheduling anomaly, a situation in which a higher priority task misses its deadline due to a lower priority task execution. The schedules are shown in Fig. 3.4.

The previous examples have shown some of the shortcomings of the existing solutions. US and EDF are useless for this problem due to large conflicts and poor schedulability respectively. On the other hand, EDF-CE provides higher schedulability with fewer conflicts, but it does not provide any guarantees on the amount of parallelism, misses many chances for improvement, and may cause scheduling anomalies. The schedule

produced by EDF-CE exhibits this problem, see Fig. 3.4b. Task T1 has the highest priority (i.e., lowest deadline), to achieve maximum overlap, T3 was allowed to run concurrently with T1, but since T3 has a longer execution time, it will continue running past T1 causing T2 to miss its deadline.

3.3 The Scheduling Algorithms

Since the problem is NP-hard, we develop two heuristic algorithms based on graph partitioning. The algorithms generally has the following three steps:

1. Construct the task conflict graph.
2. Partition the task conflict graph into a least number of partitions.
3. Schedule each partition concurrently as long as their is enough slots in the time frame, where the time frame is the period of the uniform task set, or the least common multiple (LCM) of the periods in a non-uniform task set. The number of tasks scheduled to run concurrently in each time slot is equal to the cardinality of the partition.

The high level flow chart for our scheduling algorithms is shown in Fig. 3.5.

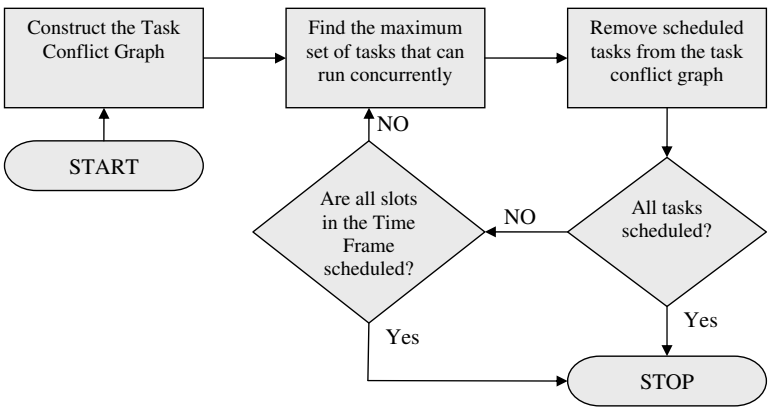


Figure 3.5 The Flow chart of the conflict-aware scheduling algorithms.

3.3.1 Conflict-Aware Scheduling of Uniform Tasks

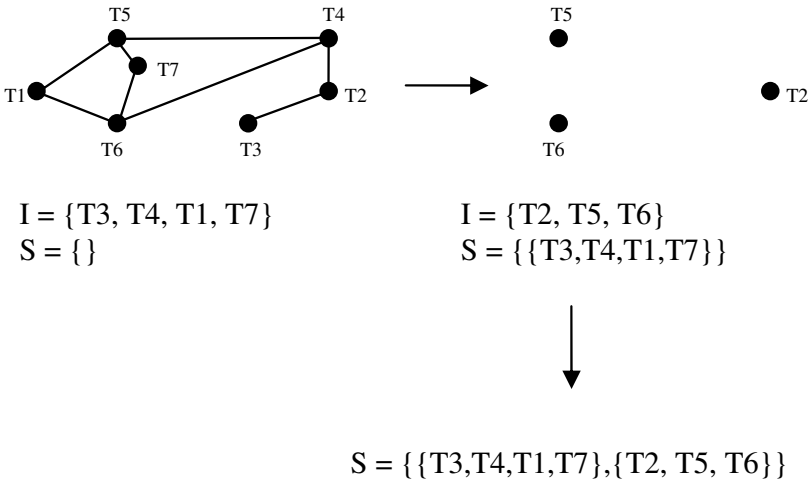
The conflict-aware scheduling algorithm for uniform tasks is shown in Algorithm 1. The algorithm assumes that the task conflict graph had been constructed, and it is provided as input. The central piece of this algorithm is the partitioning sub-routine. We use a least conflict first partitioning algorithm [57]. The task with the least number of conflicts is chosen first. Then, all of its neighboring (*i.e.,* conflicting) tasks in the graph are removed. We proceed until no more tasks can be added to the current set (step 1.4). The tasks from the resulting partition are removed from the graph, along with their incident edges (step 1.5). The process is repeated until all tasks are grouped. Each group of tasks can be run concurrently. In case the number of time slots available is not enough, the largest cardinality of task subsets can be run first to minimize the number of dropped tasks.

Going back to example 1, we can easily verify that our conflict-aware algorithm for uniform tasks will give the same solution as the optimal algorithm (for this example). Fig. 3.6 shows the working of the algorithm, and the resulting schedule.

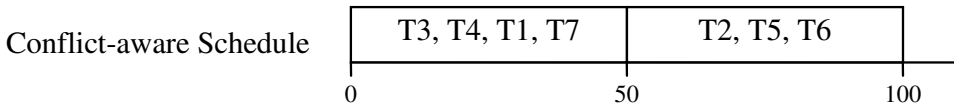
input : The Task Conflict Graph $G = (V, E)$, where all tasks are uniform.
output: A schedule of tasks

1.1 $S \leftarrow \phi$ // A collection of subsets
1.2 $I \leftarrow \phi$ // A subset of tasks
1.3 **while** $G \neq \phi$ **do**
1.4 $\quad I \leftarrow Partition(G)$ // Find the largest possible
$\qquad\qquad$ // non-conflicting subset of tasks
1.5 $\quad G \leftarrow G - I$
1.6 $\quad S \leftarrow S \cup I$
1.7 **end**
1.8 *Since all tasks have the same deadline, tasks within a partition are executed in parallel, while different partitions are executed sequentially.*

Algorithm 1: The conflict-aware scheduling algorithm for uniform tasks.



(a) The working of algorithm 1.



(b) The schedule produced by algorithm 1.

Figure 3.6 Under the conflict-aware algorithm for uniform tasks, all tasks meet their deadlines.

Since the problem is reducible into a minimum graph coloring (or the maximum independent set) problem [65], a wide variety of approximation algorithms can be used to solve this scheduling problem while providing performance bounds.

The above conflict-aware algorithm assumes a uniform task set. This assumption is not without merit. For example, RON uses pair wise measurements between the members of the overlay, so a central operator may be issuing measurement requests at regular intervals, while the execution time estimates is the worst case latency between any pair of nodes in the overlay times some tool specific factor. Thus, the resulting task set is uniform. The next section addresses the problem of non-uniform tasks.

3.3.2 Conflict-Aware Scheduling for Non-uniform Tasks

For the general case of a non-uniform task set scheduling, we use period transformation, and execution time transformation [67]. Tasks within each partition are transformed into a certain number of the same uniform task $\tau = (C, P)$. This uniform task can be the same across partitions or different. Transforming the tasks into a common task ensures that all the tasks are aligned with each other. Thus, achieving higher parallelism, while maintaining the original tasks' properties of deadline and utilization. Algorithm 2 shows the pseudo code for the slotted scheduling algorithm. The algorithm proceeds in the following steps.

1. The current task set is grouped into a collection of non-uniform task partitions, as in Algorithm 1 (steps 2.3-2.7).

2. Within each partition, tasks are divided into multiple uniform tasks $\tau = (C, P)$, where P is the common period and it is smaller than the period of any task in the task set, C is the common execution time and it is smaller than the execution time of any task in the task set, and the utilization $\frac{C}{P}$ is smaller than the utilization of any other task. Subtasks from the same parent task will form a complete subgraph in the task conflict graph (i.e., they pair-wise conflict), and each subtask will inherit the conflicts of its parent task. The number of subtasks generated from a given task is:

$$\lceil \frac{c_i/p_i}{C/P} \rceil$$

There may be some performance loss due to the rounding up. However, achieving higher parallelism and no scheduling anomalies overrides this slight loss of performance. Note that τ can be the same across all partitions, or unique to each partition.

Each subtask will inherit the deadline of its parent task. Hence, inherit the priority.

3. Each partition is further divided into sub-partitions, using the least conflicts first approximation algorithm (step 2.11).

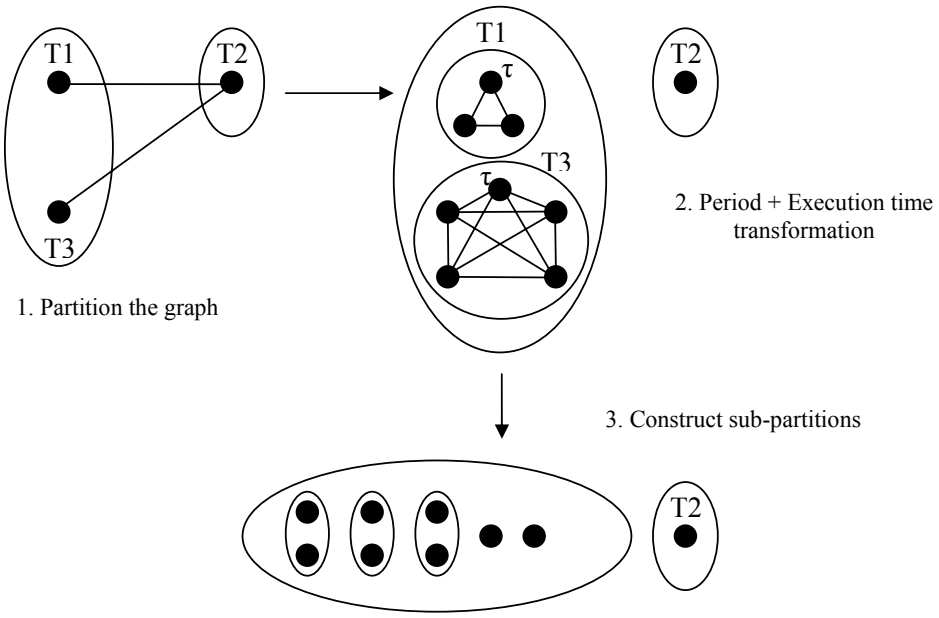
4. The sub-partitions from all partitions are arranged in increasing order of the earliest deadline of the parent task in the sub-partition. This is the schedule.

Going back to example 2, we can use $\tau = (5, 50)$ to normalize the tasks. Task T_1 will be split into 3 sub-tasks, while task T_3 will be split into 5 subtasks. This division is useful for the purpose of finding maximum overlap, while preserving task priorities. However, the tasks actual deadline is still the same. For example, in Fig. 3.7 at time 50, 3 sub-tasks of T_3 will miss their deadline imposed by τ, but the actual deadline is 100. Task T_2 was not split because it was in a different partition.

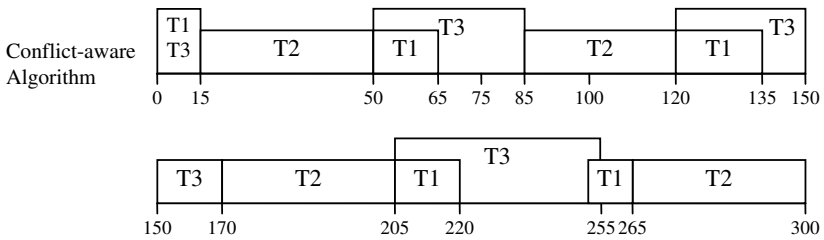
input : The Task Conflict Graph $G = (V, E)$.
$\tau = (C, P)$
output: A schedule of tasks

2.1 $S \leftarrow \phi$
2.2 $I \leftarrow \phi$
2.3 **while** $G \neq \phi$ **do**
2.4 $\quad I \leftarrow Partition(G)$
2.5 $\quad G \leftarrow G - I$
2.6 $\quad S \leftarrow S \cup I$
2.7 **end**
2.8 **foreach** *subset* $s \in S$ **do**
2.9 $\quad$ *Transform task subsets using* τ
2.10 $\quad$ *Construct a subset conflict graph* H*, where subtasks from the same parent task share an edge*
2.11 $\quad$ *Construct sub-partitions from* H
2.12 **end**
2.13 *The earliest deadline of a parent task in a sub-partition is the deadline of that sub-partition.*
2.14 *Sort all the resulting sub-partitions in increasing of deadline. Output the schedule.*

Algorithm 2: The conflict-aware scheduling algorithm for non-uniform tasks.



(a) The working of algorithm 2.



(b) The schedule produced by algorithm 2.

Figure 3.7 Algorithm 2 solution to example 2. Note that borders of consecutive slots assigned to the same parent task are merged together.

3.4 Topology-Aware Scheduling

The scheduling of measurements can be thought of as a time dimension solution for a space dimension problem, by space we mean the network path or link. However, depending on the task set, the scheduling algorithm may not be able to solve the problem within the given time constraints. In this case, we can combine space and time solutions to modify the task set in such a way that it makes the new task set schedulable, albeit at the expense of aggregation and coordination overhead. Consider the example in Fig. 3.1, if it is not possible to schedule tasks T_1 and T_2 at non-overlapping times, we employ the topology information to resolve the conflict among these tasks in the following manner: task T_1 is split into tasks T_{11} running from A to C, T_{12} running from C to D, and T_{13} running from D to F. Task T_2 is split into tasks T_{21} running from B to C, and task T_{22} running from D to E, in addition to task T_{12}. The topology-aware solution is shown in Fig. 3.8.

Note that nodes C and D are overlay nodes. All of the new tasks can be run concurrently at the expense of incurring an overhead associated with aggregating the measurement results. Thus, the spatial partitioning technique improves the schedulability while incurring some additional overheads. In this example, we were able to split the tasks because nodes C and D are overlay nodes. However, this could not have been possible if these nodes were, for example, core routers.

Aggregation of Monitoring Results

Let Task T_i be split into tasks $\{T_{i1}, T_{i2}, \ldots, T_{im}\}$, then the aggregation of monitoring results can be easily performed, as follows:

1. *Latency (L):* $L(T_i) = \sum_{j=1}^{m} L(T_{ij})$

2. *Loss rate (R):* $R(T_i) = 1 - \prod_{j=1}^{m} [1 - R(T_{ij})]$

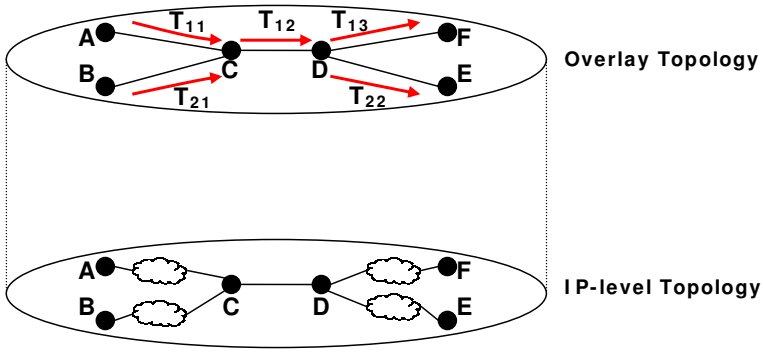


Figure 3.8 Splitting the measurement tasks in Fig. 3.1 into multiple non–conflicting tasks that can run concurrently.

3. *Bandwidth (BW):* $BW(T_i) = \min_{j=1}^{m} \ BW(T_{ij})$

As an example, consider task T_2 in Fig. 3.8. The results of task T_2 can be obtained as follows:

1. $L(T_2) = L(T_{21}) + L(T_{12}) + L(T_{22})$

2. $R(T_2) = 1 - [1 - R(T_{21})] \times [1 - R(T_{12})] \times [1 - R(T_{22})]$

3. $BW(T_2) = \min\{BW(T_{21}),\ BW(T_{12}),\ BW(T_{22})\}$

Which Tasks to Split?

The overlay topology and the underlying IP-level topology will put some restrictions on the pool of tasks that can be split. Out of these, there may be many heuristics for choosing the tasks to be split that can achieve the desired schedulability. Such heuristics include splitting the tasks with the most number of conflicts, and running the previously mentioned scheduling algorithms on the new conflict graph. In this chapter, we adopt a different approach, where the topology-aware splitting is done as a post processing step to the previous algorithms. The intuition behind such approach stems from the observation

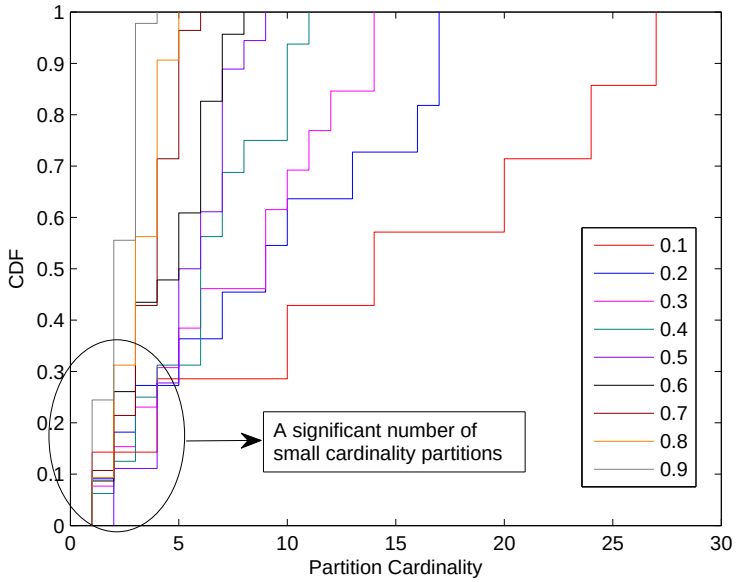


Figure 3.9 The cumulative distribution function (CDF) of the partitions' cardinality produced by the least conflicts first partitioning algorithm for various conflict factors $(0.1 - 0.9)$.

in Fig. 3.9, which plots the cumulative distribution function (CDF) of the partitions' cardinality produced by the least conflicts first algorithm (the exact setup is the same as Fig. 3.12 in section 3.6.2). We have noticed that, for most conflict factors (i.e., probabilities), partitioning results in a significant number of small cardinality partitions (i.e., 3 tasks or less). Such partitions serve as a perfect candidate for merger, as they can save a predictable number of scheduling time slots. The conflict among tasks belonging to different partitions is resolved by splitting conflicting tasks as mentioned in the example in Fig. 3.8.

The Topology-Aware Scheduling Algorithm

The Algorithm listing is given in Algorithm 3. The algorithm embodies the previous discussions. It starts by initializing the number of split tasks N to 0. This number

should not exceed a given overhead threshold K, which we choose to specify as the maximum number of tasks that we are allowed to generate by splitting existing tasks. The threshold K is related to, and depends on, the coordination and aggregation overhead associated with the monitoring system. Next, the algorithm chooses the smallest two partitions, resolves the conflict among their constituting tasks by performing topology-aware splitting, and merges them together into a single partition. The process is repeated until the overhead threshold is reached.

The solution for non-uniform tasks is a little bit tricky, as the produced schedule is in terms of subtasks. In this case, we first use Algorithm 1 to produce an intermediate schedule on which we apply Algorithm 3. The amount of saved time slots depends on the longest task in each partition. Then, we apply Algorithm 2.

input : Set of Tasks: T
The Task Conflict Graph $G = (V, E)$
Overlay and IP-level topologies
Overhead Threshold K
output: A feasible schedule of tasks.

```
3.1  Run Alg. 1 on Task Set T
3.2  Initialize Number of split tasks, N ← 0
3.3  while N ≤ K do
3.4  |  use the network topologies to resolve conflict among tasks in the
     |  smallest two partitions in the schedule.
3.5  |  Update N
3.6  |  if N > K then
3.7  |  |  STOP and exit loop.
3.8  |  end
3.9  |  Merge the two partitions.
3.10 end
3.11 if Schedule is feasible then
3.12 |  Output schedule
3.13 end
3.14 else
3.15 |  Algorithm Fails.
3.16 end
```

Algorithm 3: The topology-aware scheduling algorithm.

3.5 Implementation Issues

The scheduling algorithms are envisaged as a component of a monitoring infrastructure, which is part of a larger network management system, or a network application. A node is selected as a controller, which is responsible for collecting and scheduling measurement requests. To make the controller fault tolerant, well-known backup and leader election strategies can be used. As for the performance bottleneck concern, we argue that the communication and computation costs at the central node are small for a reasonably sized monitoring task set.

Topology changes are another valid concern. However, several related studies [25, 26, 56] have assumed that the network topology is stable for a reasonable amount of time that allow for at least one round of schedule execution. In addition, changes to the topology do not necessarily mean changes to the tasks conflict graph, especially if the conflict-causing overlap is at the overlay level.

Period and execution time transformation may lead to increased execution time of the scheduling algorithm. Since the scheduling algorithm is either run offline or run only when changes in the task set occur, this kind of increase is very moderate considering the fact that the least conflict first partitioning algorithm runs in linear time [57], and the processing capabilities of the centralized controller.

The design of the measurement tools should take into consideration the measurement conflict problem, and the potential scheduling. Network measurement is a sophisticated process that involves many design parameters that need to be considered in conjunction with scheduling. Several measurement techniques exit, and some of them are more suited than others for a scheduling environment. For example, to measure bandwidth, some tools (*e.g.,* Pathload [42], and PTR [44]) measure bandwidth over a short interval, which is more convenient. However, other tools (*e.g.,* pathChirp [41], Spruce [43]) use a relatively longer measurement interval, with statistically constructed probing gaps.

The tool designer needs to consider the effect of the possible disruption, caused by scheduling, of the gaps between probe sequences. The measurement interval may also affect the choice for C and P in τ.

Another implementation issue is that of the global synchronization of measurements schedules starts and stops. Achieving a 100% synchronization of a distributed system's clocking is a difficult task. However, considering Fig. 3.10, we can see that complete synchronization is not necessary, a useful property of this conflict problem is that measurements accuracy gradually degrades with the amount of overlap. Thus, a certain amount of synchronization imperfection can be tolerated.

3.6 Performance Evaluation

3.6.1 Experimental Evaluation

A globally distributed monitoring network is constructed on top of PlanetLab. The network consisted of 25 nodes (15 in the USA, 5 in Europe, and 5 in Asia and Australia). Each node is programmed to conduct measurements every one hour, and later on, they are scheduled to conduct measurements in sufficiently non-overlapping time periods. The reported results use Iperf as the measurement tools, which reports the transfer throughput achievable using UDP or TCP. Performed experiments using other tools such as Pathload (measures available bandwidth), and H.323 beacon (measures support for Voice and Video over IP) results in drawing similar conclusions [25].

Fig. 3.2 shows the drastic effect that the conflict has on the accuracy of the measurement, as seen by any individual node. With 10 conflicting measurements, the reported throughput is only about $20 - 30\%$ of the reported value without conflict. The experiment had been repeated multiple interleaved times to ensure network dynamics are not the reason for the degradation. In addition, we have noticed the high stability of the PlanetLab paths, due probably to the Internet 2 connections.

Measurement Accuracy vs. Schedulability

Fig. 3.10 highlights the opportunity for another area of research in this context. Certain applications may require bounded rather than fully accurate measurements [26]. Thus, we can change the task model into the well studied Imprecise Computation model [69]. The imprecise computation model logically divides the task into two parts, mandatory and optional. The mandatory path should be executed completely, while the optional part may be partially executed. The error in executing the task is the amount of unfinished optional execution time. We can bring this model into the problem by requiring a mandatory portion of a measurement task to be executed in exclusion from other conflicting tasks, while its optional part may be scheduled in a conflicting manner.

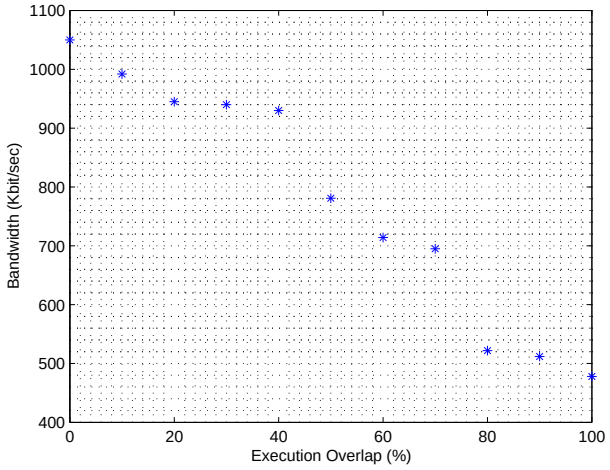


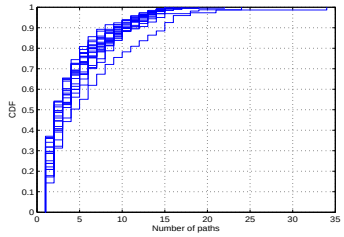
Figure 3.10 Degradation of measurement accuracy as a function of the percentage of of the task's execution time overlapping with other conflicting measurement tasks.

3.6.2 Simulation Results

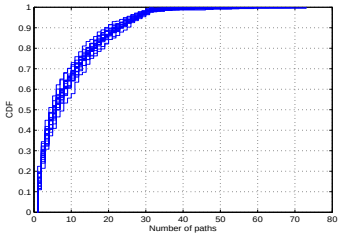
3.6.2.1 Path Sharing

Another aspect of the problem to be studied is the amount of sharing among overlay paths at the IP-level. This aspect is very important since it gives an incite into the overlay path overlap necessary to cause conflict in measurements. We use real Internet ISP topologies of two major U.S. carriers provided by RocketFuel [72]: AT&T (AS# 7018, 11800 nodes), and Sprint (AS# 1239, 10332 nodes).The mapping between an overlay link and the underlying IP level path has been done using shortest path routing on the IP network using hop count as the metric. The set of overlay nodes is uniformly selected at random. The size of the overlay (i.e., number of nodes N) is set to 16, 32, and 64. The overlay topology is varied in two ways: a complete graph (denoted as complete), where each nodes maintains overlay links to every other node, and a random graph (denoted as log) where each node maintains $\log_2 N$ neighbors.

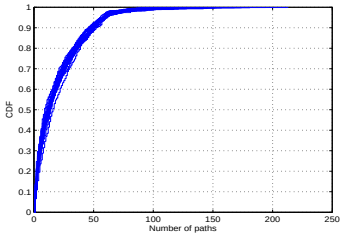
Fig. 3.11 shows the cumulative distribution function (CDF) of the number of links shared by a given number of overlay paths, for various network sizes, overlay topologies, and the two carriers mentioned above. It is clear from this figure that there is substantial sharing among overlay paths. For example, for a complete overlay graph of 64 nodes over the AT&T network, 65% of links have participated in about 25 paths or less, this number drops to 5 paths or less for an overlay of size 16, which is a very small overlay size. Note that the paths may or may not be the same. If we change the overlay topology to a random graph (at the expense of the coverage and debugging capabilities), the same numbers drop to 12 paths and 4 paths respectively. The Sprint network exhibits similar properties.



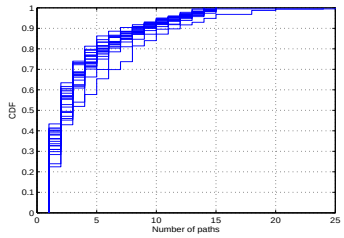
(a) AT&T, Complete, Network size = 16



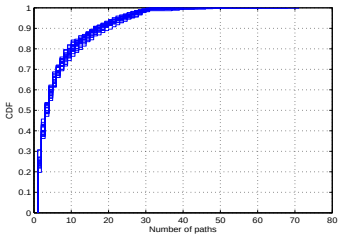
(b) AT&T, Complete, Network size = 32



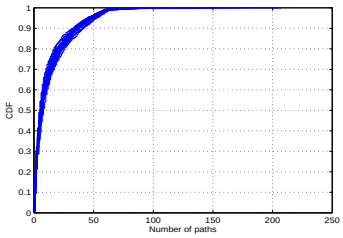
(c) AT&T, Complete, Network size = 64



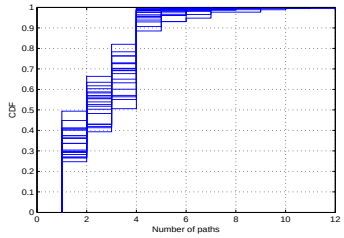
(d) Sprint, Complete, Network size = 16



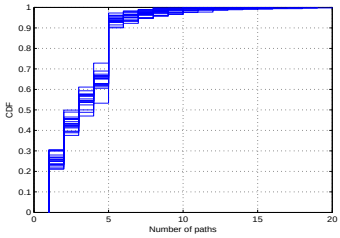
(e) Sprint, Complete, Network size = 32



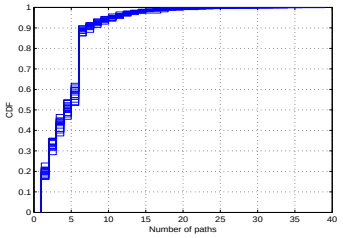
(f) Sprint, Complete, Network size = 64



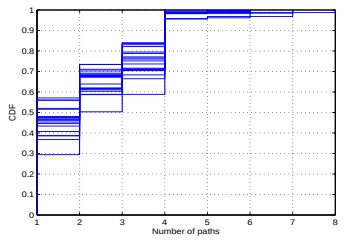
(g) AT&T, log, Network size = 16



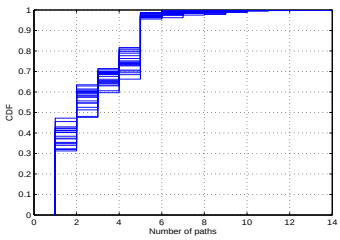
(h) AT&T, log, Network size = 32



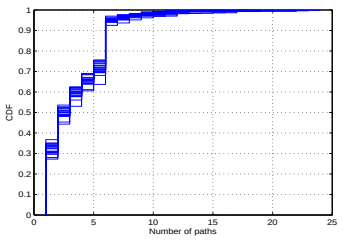
(i) AT&T, log, Network size = 64



(j) Sprint, log, Network size = 16



(k) Sprint, log, Network size = 32



(l) Sprint, log, Network size = 64

Figure 3.11 The cumulative distribution function (CDF) of the number of IP links being shared by a given number of overlay paths.

3.6.2.2 Schedulability of uniform task sets

Scheduling uniform task sets depends solely on how good the partitioning algorithm performs. To study this effect, we experiment with a task conflict graph consisting of 100 tasks, each pair of tasks share an edge based on a certain conflict probability. We vary the conflict probability from 0.1 to 0.9, and we report the average number of partitions generated over 20 runs. We find the optimal solution to the uniform scheduling problem by solving successive Maximum Independent Set problems on the task conflict graph. Given the task conflict graph $G = (V, E)$, $|V| = n$, the Integer Linear Program (ILP) formulation associates a binary variable x_v to each node $v \in V$. The ILP is as follows:

Maximize

$$\sum_{v \in V} x_v$$

Subject to

$$x_u + x_v \leq 1, \ \forall \{u, v\} \in E$$

$$x_v \in \{0, 1\} \ \forall v \in V$$

Fig. 3.12 shows that the proposed least conflicts first approach achieves about 10% less number of partitions than EDF-CE. We solve the ILP for the optimal solution using the CPLEX optimization tool [74]. Surprisingly, both solutions are close to the optimal. This is due to the fact that we are finding successive independent sets not just one. So the optimal solution may start by finding large independent sets, but as edges are eliminated from the graph, all the algorithms will end up with a significant number of pair-wise conflicting tasks, which they will handle similarly.

Fig. 3.13 compares the least conflicts first partitioning algorithm with the topology-aware approach. As expected, the topology-aware approach reduces the number of partitions by a decent number, and even surpasses the optimal solution numbers reported in Fig. 3.12. However, it incurs higher coordination and aggregation overhead.

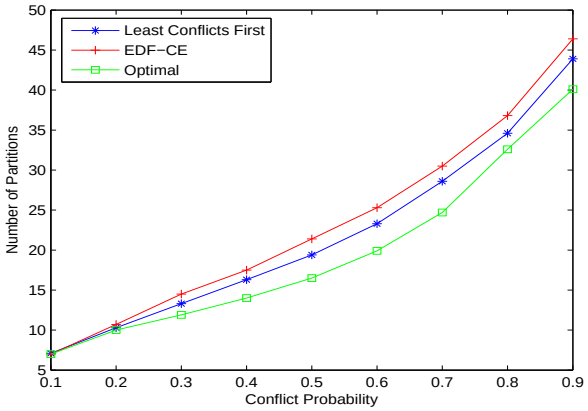


Figure 3.12 The number of partitions produced by the EDF-CE algorithm, and the least conflicts first partitioning algorithm.

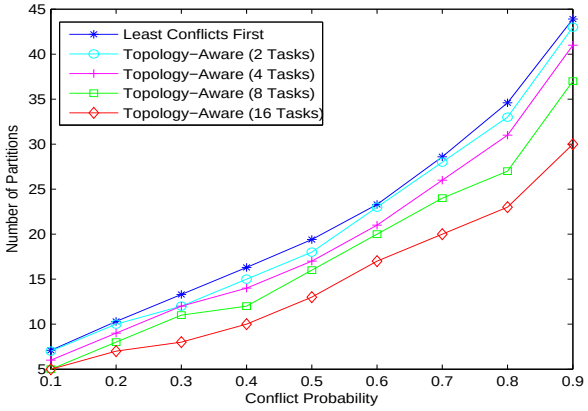


Figure 3.13 The number of partitions produced by the least conflicts first algorithm and the topology-aware scheduling with resolved conflicts among a various number of tasks.

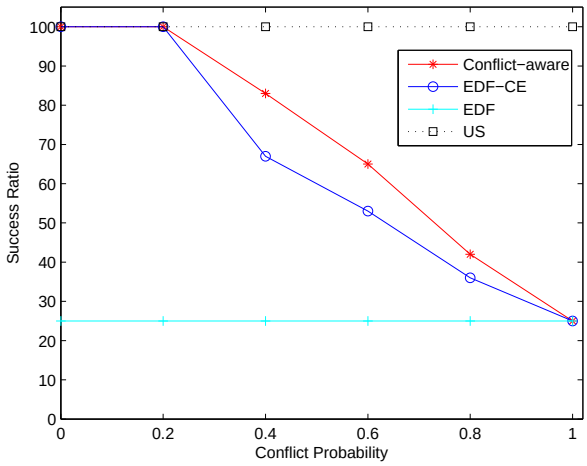
3.6.2.3 Schedulability of non-uniform task sets

We now examine the schedulability of the conflict-aware algorithm for non-uniform tasks. In particular, we study the success ratio of our approach in comparison to existing solutions. The success ratio is defined as:

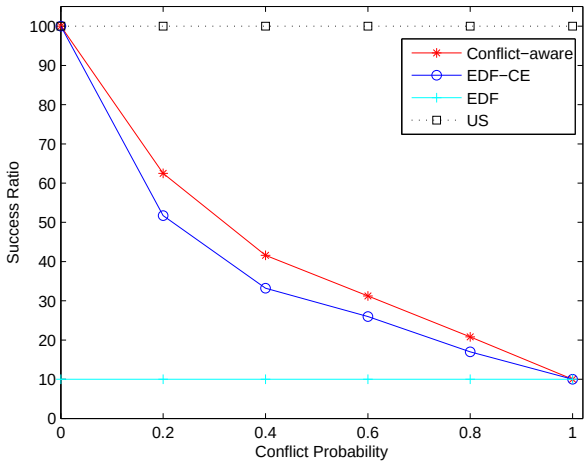
$$success\ ratio = \frac{\#\ of\ tasks\ successfully\ scheduled}{Total\ number\ of\ tasks}$$

We used a set of 20 tasks, and took the average of 10 runs. The period of each task is selected uniformly at random from $[100, 1000]$, we report the success rate as a function of the conflict probability among tasks for task utilization values of 0.2, 0.4, 0.6. The execution time of each task is the product of its period and the utilization value used in the corresponding figure.

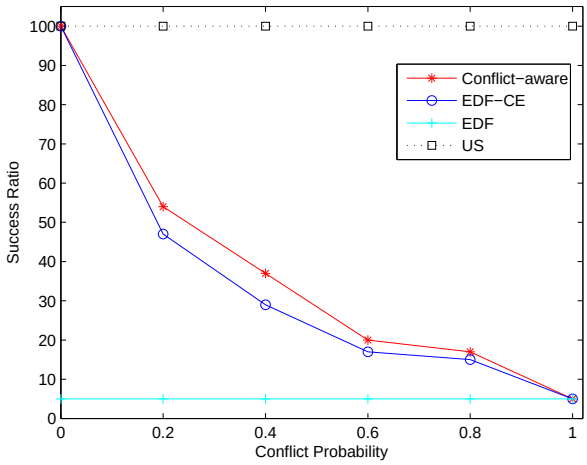
Fig. 3.14 shows that the conflict-aware scheduling algorithm tasks achieves up to 25% better success ratio. The main reasons for this improvement are the elimination of scheduling anomalies, higher parallelism achieved by our algorithm, and a higher density of tasks in each partition (*i.e.,* a small number of partition have a large number of tasks), which results in dropping low density partitions. In addition, it shows that our conflict-aware approach and the EDF-CE algorithm follow the same trend as the conflict probability increases. At low conflict probabilities (*i.e,* below 0.2), there is a great deal of possible tasks concurrency, and both algorithms achieve high schedulability. In the region between 0.2 and 0.8 conflict probabilities, the conflict-aware algorithm achieves superior schedulability compared to EDF-CE. This superiority decreases with increasing task utilization due to the smaller room for improvements. Both algorithm converge again at high conflict probabilities (*i.e.,* probabilities greater than 0.9), where most of the tasks conflict with each other, and they must be executed sequentially to avoid conflicts.



(a) Task utilization = .2



(b) Task utilization = .4



(c) Task utilization = .6

Figure 3.14 The success ratio as a function of the conflict probability.

3.7 Conclusion

Network measurement is an important part of any network application or network infrastructure. Traditionally, the focus has been on developing single-source single-destination measurement tools that will most accurately estimate the properties of the path of interest. Conduction network-wide measurements were then a matter of deploying these tools on strategic nodes in the network. The collective effect that these measurements have on each other has been largely ignored.

In this chapter, experimental studies that prove the existence and severity of the measurement conflict problem were conducted. The problem of conducting conflict-free measurements as a scheduling problem of real-time tasks was formulated and its complexity was proven to be NP-hard. Polynomial time heuristic scheduling algorithms were proposed based on a well-known Maximum Independent Set approximation algorithm, which achieves 10% less number of partitions. Simulation studies have shown that our algorithms improve schedulability by at least 25% compared to existing solutions. In addition, studies of the conflict-causing topology overlap on real Internet AS-level topologies was performed.

Future work includes evaluating the effect of measurement conflict on network applications, such as overlay multicast. For the scheduling part, concepts from the imprecise scheduling literature, where tasks are allowed to partially overlap, can be formulated and solved to increase the schedulability of measurement tasks. In addition, it is appealing to apply and compare several graph coloring heuristics and approximation algorithms to the measurement conflict problem. These algorithms can aid in obtaining bounds on the schedulability of measurement tasks.

CHAPTER 4. LINK STRESS CONTROL IN OVERLAY NETWORKS MONITORING

Monitoring algorithms typically choose the set of paths to be probed, or they probe all the overlay paths in the networks (i.e., pair-wise probing). As a consequence of the overlay and physical topologies, we might get in a situation, where certain physical and overlay links are shared among a large number of overlay paths. Once the set of paths to be probed has been chosen, members of the overlay will send probes along these paths regardless of the effort performed by other members. In other words, properties of shared path segments are being measured indirectly multiple times.

The maximum number of measurements going through a certain link can vary between $O(n)$ and $O(n^2)$, where n is the number of overlay nodes. Although the typical number of overlay nodes is small (*e.g.,* less than 250), the amount of injected traffic varies depending on the measurement application. For example, consider two PlanetLab [58] nodes; planetlab-3.ece.iastate.edu and planetlab1.ucsd.edu. Testing reachability with a simple ping will inject a handful of packets at regular intervals. On the other hand, measuring bandwidth or throughput, using a tool like Iperf [59], will repeatedly inject sequences of 8 KB packets over a period of 10 seconds. Our experiments show that a total of about 3.5 MB of packets were injected into the network for this measurement. This high overhead is not peculiar to Iperf. Other measurement tools generate comparable overhead [41]-[44]. It is of no surprise that this amount of overhead can cause delay, congestion, and loss, which results in reporting incorrect measurements to the network

administrator or the beneficiary application.

In this chapter, the goal is to reduce the link stress caused by monitoring algorithms. To this end, the concept of *Internal Probing*, which uses readily available physical topology information to discover highly shared links and construct network probing partitions, is introduced. Internal Probing tries to control the perimeter beyond which certain probes (*i.e.,* those probes crossing highly shared links) can not propagate. Simulation studies using GT-ITM [75] topology generator are presented. Results confirm that Internal Probing significantly reduces average link stress by up to 30%.

The remainder of this chapter is organized as follows. We present a motivational example in section 4.1. We discuss related work in section 4.2. We describe the Internal Probing approach in detail in section 4.3. We present our simulation results in section 4.4. We conclude in section 4.5.

4.1 Motivation

Consider the example shown in Fig. 4.1. If we were to use pair-wise probing to measure the bandwidth of all the overlay links, then physical links g and h would have a link stress of $18T$ bytes, where T is a tool-specific value (*e.g.,* 3.5 MB for Iperf in the previous example). However, the link stress can be greatly reduced by making use of the IP-level topology as follows: Let nodes A, B, and C probe node H, while nodes D, E, and F probe node G, only T bytes will be needed to measure the bandwidth of link HG as we will see in section 4.3. Thus, reducing the link stress on links g and h to T. The probing results can be aggregated and disseminated by a designated node. The key to the improvement achieved by Internal Probing is that we only need a small number of packets to report the results obtained from injecting a high volume of monitoring traffic (*e.g.,* 18T= 18×3.5 MB = 63MB). Assuming that each node generates a 10KB performance report, which is fairly large [28], then the aggregation overhead is 90KB.

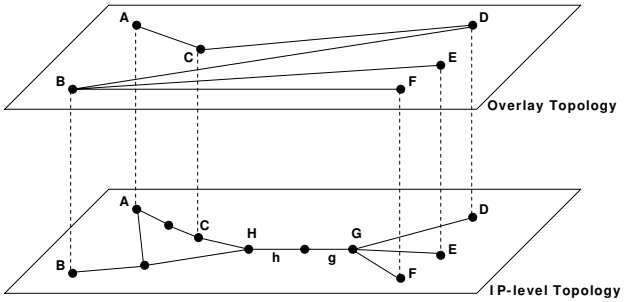


Figure 4.1 An example overlay network and the underlying IP-level topology. Overlay paths AD and CD overlap, while physical links h and g are shared among many overlay paths.

4.2 Related Work

Recent work on network monitoring has focused on reducing the total number of probes used (*i.e.,* probing overhead), as exemplified by Chord [6] and Pastry [10], as well as the approach described in [26]. In Chord and Pastry each node maintains connections to $O(\log n)$ other nodes. Hence, the probing overhead is reduced to $O(n \log n)$, but each node will have a partial view of the network. However, these methods still suffer from high link stress [26]. The authors in [26] suggest that each node should have a full view of the network at the expense of measurement accuracy. They show that an $O(n \log n)$ probing overhead can achieve a 90% accuracy. Nevertheless, a high link stress had also been reported. RON [4] and Narada [5] assume a small overlay network (*i.e.,* less than 50 nodes) and use pair-wise probing, with a total overhead of $O(n^2)$. We have seen earlier that the number of nodes is not the only factor affecting link stress and overhead, the type of measurement needed (*e.g.,* bandwidth or a simple reachability test) has a great effect. Otherwise, the monitoring bandwidth requirements will be small if we would only consider the number of nodes in typical overlay networks (*e.g.,* less than 250).

Network tomography of network links from end-to-end path measurements typically employs complex statistical methods (*e.g.,* The expectation-maximization (EM) and the

MVWA algorithms [32]). Our approach is similar to the network tomography schemes. However, we aim at finding the path characteristics as opposed to link characteristics, while reducing the link stress due to probing. In addition, we use simple algebraic manipulations.

In another related study, Kwon and Fahmy [27] have proposed exploiting the underlying network topology to construct overlay multicast networks that satisfy the application requirements. They have assumed that the underlying routes are of high quality, which requires constant monitoring of the overlay paths. Our approach is complementary in nature to their approach, as the input to our approach is the output of theirs.

In chapter 3, the problem of measurement conflict, due to the communicational and computational requirements of measurement tools and the network topology, was addressed. A scheduling algorithm that prevents conflicting measurements from executing at the same time was proposed. However, the total number of monitoring packets crossing each link is still the same. Depending on the number of measurement tasks, it may not be possible to schedule all the tasks in a non-conflicting manner. The solution that we proposed in [65] can be thought of as a time dimension solution. While the solution proposed in this chapter is a space dimension solution. By space, we mean a network link or path. Both solutions can be complementary, and an ongoing study is combining the benefits of both approaches.

4.3 Methodology

In this section, a formal network model is given first, followed by the Internal Probing approach.

4.3.1 Network Model

Given an overlay network undirected graph $G_o = (V_o, E_o)$, where V_o is the set of overlay nodes and E_o is the set of overlay edges. The corresponding IP (Internet Protocol) level graph $G = (V, E)$ is also available, where V is the set of physical nodes and E is the set of physical edges. Note that $V_o \subseteq V$, but $E_o \nsubseteq E$ in general. Instead of working directly on the physical graph G, we can construct a topology-aware overlay network $G_T = (V_T, E_T)$ [20, 68], where $V_T \subseteq V$, and E_T is the set of path segments. A path segment is defined as the maximal sub-path such that no two path segments share a physical edge, and every overlay path can be expressed in terms of elements in E_T [20, 68]. The graph G_T is not necessary for the correctness of this approach, but merely to work at a higher granularity than the physical network. For example, in Fig. 4.1, physical links g and h will be treated as one segment.

We make the following assumptions that are commonly used in the literature:

1. There is a great deal of sharing between overlay paths. Recent studies have shown that this assumption is reasonably true [26, 70].

2. Network level topology information is available. This assumption has been made by several studies [20, 26, 71]. Also, tools that provide topology information have been developed [56].

3. Overlay paths are stable for a reasonable amount of time (*i.e.,* order of minutes) [4] to allow for the aggregation of the probing results. This is true since path changes are triggered by path quality changes discovered using probe packets, while the opposite does not necessarily hold [26].

4. While overlay nodes are assumed to be able to inject and/or respond to probes, non-overlay nodes only respond to probes. This assumption is especially valid because of the specific application we are considering, where ISPs have full control of

the underlying IP network. Some researchers, with a similar deployment scenario, have gone to the extreme by actually deploying measurement servers on top of core routers [25]. Although ISPs may be able to optimize the overlay topology to serve their purpose, IP-level routing determines the mapping of overlay paths to the actual physical links. For example, link HG in Fig. 4.1 can be a BGP route or a satellite link. In addition, other deployment scenarios (*e.g.,* storage and lookup systems) optimize the overlay topology to serve the specific application requirements rather than the measurement requirements.

4.3.2 The Concept of Internal Probing

Internal Probing works by dissecting the current probing set (*i.e.,* the set of paths to be probed) to account for the sharing of path segments. Path segments exceeding a given sharing threshold will induce partitions on their respective paths to create a new probing set. We identify three kinds of probes needed for probing the new set, as shown in Fig. 4.2:

1. *End-to-End probes:* These probes originate from an overlay node and get replied by another overlay node at the other end of the overlay path. Unpartitioned overlay paths will use these probes (*e.g.,* in Fig. 4.2 path BC uses probe Prb_3).

2. *Internal-to-Internal probes:* The end points of these probes are non-overlay nodes, which is caused by partitioning (*e.g.,* Prb_2 in Fig. 4.2).

3. *End-to-Internal probes:* These probes originate from an overlay node, but get replied by an intermediate node. They will be used once for each highly shared path segment, and along with type 2 probes above, they are the key for achieving lower link stress. There are two cases where this kind of probes can be employed:

(a) Probing path segments with an overlay node at one end, while the other end is an intermediate node (*e.g.,* probe Prb_1 in Fig. 4.2).

(b) If Internal-to-Internal probes can not be used (*i.e.,* internal nodes unable to inject probes), then these probes are used to measure the properties of the inter-partitions segments.

Consider segment XY in Fig. 4.2. In order to measure its properties, node X (or Y) can send probe Prb_2. Alternatively, node A can send a probe to node Y and a sperate probe to node X (*i.e.,* Prb_1), then use the following equations to calculate the the various QoS metrics:

1. *Latency (L):* $L(XY) = L(AY) - L(AX)$
2. *Loss rate (R):* $R(XY) = 1 - \frac{1-R(AY)}{1-R(AX)}$
3. *Bandwidth (BW):*

$$\textbf{IF } BW(AX) > BW(AY) \textbf{ Then}$$
$$BW(XY) = BW(AY)$$
$$\textbf{ELSE}$$
$$BW(XY) \geq BW(AY)$$

Now, we can consider dissecting the probing set. Without Internal Probing, only probes of type 1 will be used and the resulting link stress is excessive. Another approach would probe each path segment separately using probes of type 3 and/or type 2, which results in a lower number of probes crossing each link. However, a higher number of nodes will participate in the measurement process, which increases the complexity and signalling overhead to coordinate the process and aggregate the results.

We aim at finding a compromise between link stress reduction and coordination complexity under the constraints imposed by the limited capabilities of the internal

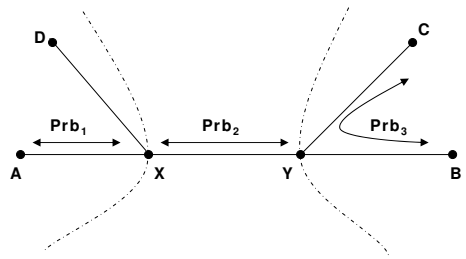


Figure 4.2 Highly shared path segment XY has induced a partition of the overlay paths. Overlay paths AB, AC have been partitioned into path segments AX, XY, YB, and YC.

(physical) nodes. To achieve this goal, we define a segment sharing threshold as the number of overlay paths sharing this segment. Previous studies [5][26] have shown that the average and worst case link sharing are generally considerably higher than $O(n)$. The optimal choice of a sharing threshold will depend on the number of overlay nodes, the overlay topology, the underlying network topology, the type of measurement being carried out, the maximum stress each segment is allowed to experience, and the bandwidth of each segment.

Before presenting the pseudo code for Internal Probing, we define some of the associated data structures and variables. Variable P is a collection of sets, where each set represent an overlay path p_i in terms of its constituent path segments. The notation $|p|$, means the number of elements in set p. Variable W is a set of weights, where weight w_{s_i} is associated with path segment s_i and represents the number of paths sharing s_i. Variable PS is the new probing set that achieves lower link stress. Variable K is the sharing threshold. The value of K can be the same for all links, or specific for each link. It is clear that the lower the value of K, the larger the number of segments being probed explicitly, which may increase the overhead and may not be physically possible to conduct the measurements due to accessibility limitations of the intermediate (physical) nodes.

input : The probing set $P = \{p_1, p_2, \ldots, p_{|P|}\}$ where
$p_i = \{s_1, s_2, \ldots, s_{|p_i|}\}$
$W = \{w_1, w_2, \ldots, w_m\}$
The sharing threshold: K
output: The new Probing set PS

```
4.1  foreach p in P do
4.2      S ← φ
4.3      for i = 1 to |p| do
4.4          S ← S ∪ s_i
4.5          if w_{s_i} ≥ K then
4.6              PS ← PS ∪ s_i
4.7              PS← PS ∪ S
4.8              S ← φ
4.9          end
4.10     end
4.11     if S ≠ φ then
4.12         PS← PS ∪ S
4.13     end
4.14 end
```

Algorithm 4: Internal Probing

The algorithm works by traversing each path in the probing set and if a segment has a weight higher than the threshold K, then the corresponding path will be split. At the end of the algorithm, the probing set would have been dissected into sub-paths. No segment s_i in PS would be shared among more than K paths. It is clear that the algorithm runs in order of the number of path segments in the probing set. Once the partitioned probing set is available, the previously identified probe types can be readily used to measure the properties of the paths and sub-paths in the new probing set. The end points of the path or path segment will determine the type of probe to be used. The Internal Probing algorithm is shown in algorithm 4.

To demonstrate the working of the algorithm, consider the simple network in Fig. 4.2. Let us have a probing set that consists of the set of paths $P = \{AB, AC, AD, DB, DC, BC\}$. The breakup of each path in terms of segments is as follows:

$AB = \{AX, XY, YB\}$.

$AC = \{AX, XY, YC\}$.

$AD = \{AX, XD\}$.

$DB = \{DX, XY, YB\}$.

$DC = \{DX, XY, YC\}$.

$BC = \{BY, YC\}$.

The weight of each path segment (i.e., the associated link stress in terms of the number of measurements using that segment) assuming symmetric links is $W = \{w_{AX} = 3, w_{XY} = 4, w_{XD} = 3, w_{YB} = 3, w_{YC} = 3\}$. Note that because of symmetry, a segment like XD will be regarded the same as DX. The extension to asymmetric segments is straightforward. The last input parameter is the sharing threshold K. Setting the value of K to 4 will result in dissecting or partitioning the probing set around segment XY. Thus, the new probing set is: $PS = \{AX, XY, XD, YB, YC\}$. Segments AX, XD, YB, YC are probed using type 3 probes. While segment XY is probed using either type 2 or type 3 probes. Aggregation is required to produce the measurements in terms of the original probing set, as follows:

Let path $p = \{s_1, s_2, \ldots, s_{|p|}\}$, then:

Loss rate: $R(p) \leq 1 - \prod_{s \in p} (1 - R(s))$

Latency: $L(p) \leq \sum_{s \in p} L(s)$

Bandwith: $BW(p) \geq \min_{s \in p} BW(s)$.

Implementation Issues

Throughout this chapter we have assumed the existence of the probing set. This probing set is generated by the monitoring algorithm. The Internal Probing approach is independent of the monitoring algorithm used. However, some of the monitoring algorithms are centralized (*e.g.,* [26, 71]), while others are distributed (*e.g.,* [4, 6]).

Internal Probing can be applied with centralized algorithms with no major modifications to the coordination and aggregation infrastructure. In distributed algorithms there is no coordination among nodes beyond the probing effort, so extra effort needs to be performed to disseminate the probing results. Developing appropriate multi-cast trees or distributed coordination systems will not be considered in this chapter. The authors in [28] describe an algorithm for the construction of spanning trees for distributed overlay path monitoring.

Using Internal Probing, a node is selected as a controller, which is responsible for collecting measurement requests and should have the physical and overlay topologies information available. The central node runs algorithm 4, and disseminates the information needed for probing to all other nodes (*i.e.,* who transmits what type of probes, and what type of probes are available to nodes), and aggregates the results from the various nodes. To make the controller fault tolerant, well-known backup and leader election strategies can be used.

4.4 Performance Evaluation

In this section, we present our experimental setup and discuss the simulation results.

4.4.1 Simulation Metrics and Setup

We compare the proposed methodology augmented with a distributed monitoring algorithm that uses pair-wise probing and an $O(n \log n)$ centralized probing algorithm proposed in [26], against the bare versions of these two algorithms, in terms of the following two metrics:

- Link stress. Defined as the number of duplicate probes crossing a path segment.
- Estimation accuracy. As we have discussed earlier in section 4.2, the algorithm described in [26] sacrifices estimation accuracy for probing coverage. This estimation

accuracy is peculiar to their algorithm, and it is different from the accuracy related to the measurement conflict problem described in [25]. We consider pair-wise probing as having a 100% accuracy, expressed in terms of the absolute relative error e , defined as: $e = \frac{|x-\bar{x}|}{x}$, Where:

- x, is the value obtained using pair-wise probing.
- $\bar{x}$, is the estimated value.

And the total estimation accuracy A is defined as:

$A = 1 - \sum_{p_i \in P} e_{p_i}$.

We performed our simulation on a transit-stub topology of 1500 node using GT-ITM topology generator [75]. The number of overlay nodes is varied between 4 and 256 nodes, selected uniformly at random in stub domains. We use Dijkstra's shortest path algorithm with latency as the links weight. The physical links latency values range from 1 ms to 60 ms. We use the LM1 Gilbert model [73] for setting the loss rate of the network layer links with the fraction of good links set at 90%. Available bandwidth is randomly chosen between 100 MB and 500 MB for non-stub edges, and 500 KB to 1 MB for stub edges [26]. We use two sharing threshold values, one for the non-stub edges and the other for stub edges due to the big difference in their available bandwidth. Unless otherwise stated, the sharing threshold for the high capacity non-stub edges is set to 250. Each simulation result is an average of 10 simulation runs, each of which with a different random seed and a random network topology.

4.4.2 Results

Effect of node degree: Fig. 4.3 shows the effect of node degree, in the underlying physical topology, on the average path segment stress for an overlay network of 64 nodes, and a sharing threshold of 50 for stub edges. For example, with an average node degree of 1.8, using pair-wise probing causes an average stress of 249, while the $O(n \log n)$

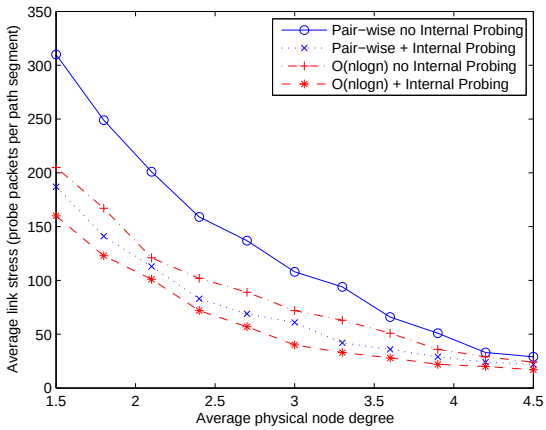
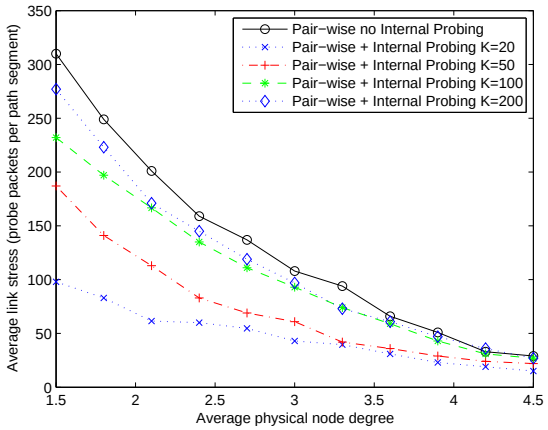


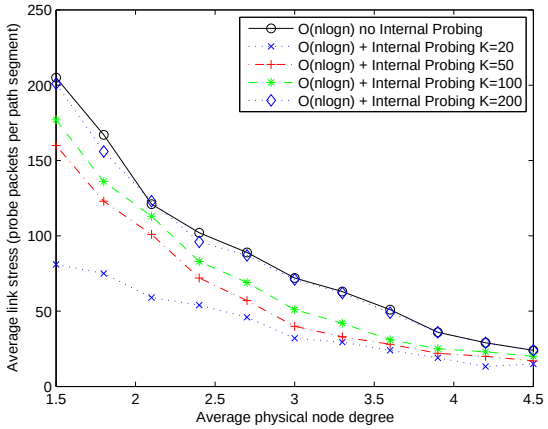
Figure 4.3 Effect of average node degree on link stress. The $O(n \log n)$ algorithm with 90% accuracy.

algorithm with Internal Probing caused an average stress of 123. As the density of the network increases, there will be more path diversity causing less sharing among overlay paths, which translates into less stress. For example, at an average node degree of 3.8, using the $O(n \log n)$ algorithm without Internal Probing will result in an average stress of 36, the same algorithm with Internal Probing will cause an average stress of 22. In general, Internal Probing produces 30% less link stress (using the same probing algorithm).

Effect of the sharing threshold (K): Fig. 4.4 shows the effect of the stub edges sharing threshold on the average link stress against the average physical node degree for the two monitoring algorithms. As the value of K is increased less and less segments will be partitioned, which leads to a lesser improvement. In Fig. 4.5 we set a global value of the sharing factor for both stub and non-stub edges. Thus, capping the maximum (and average) stress a link can experience (i.e., that of the sharing threshold value). This scenario corresponds to the case, where there is a limit on the bandwidth allocated for conducting measurements or any other similar constraints.



(a) Pair-wise probing



(b) O(nlogn) with 90% accuracy

Figure 4.4 Average link stress for various choices of the stub edges sharing threshold K.

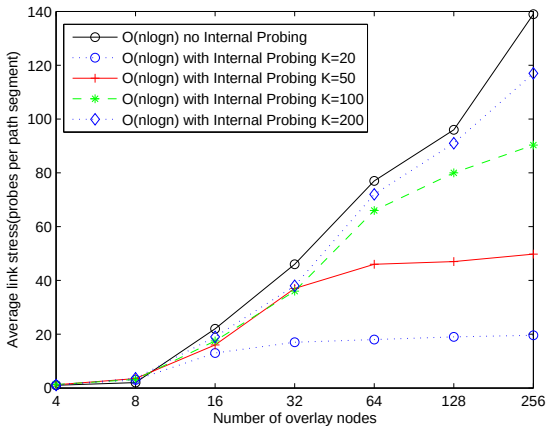


Figure 4.5 Capping the average link stress by the choice of the sharing threshold K (x axis in log scale). K here is the same for both stub and non-stub edges. Average node degree is 2.

Effect of the number of overlay nodes: For Fig. 4.6 and Fig. 4.7, the stub edges sharing threshold is set equal to the size of the overlay and 1.5 times that value for non-stub edges, and the average node degree is set to 2. Fig. 4.6 examines the effect that the number of overlay nodes has on the average path segment stress. Since the members of the overlay network are uniformly distributed across the stub domains, the larger the number of members the more overlay paths are there to step on each other. Again, Internal Probing achieves better link stress.

Effect on the probing accuracy: While Internal Probing is helpful in improving link stress, it can also achieve better accuracy for the centralized algorithm. As Fig. 4.7 shows, Internal Probing requires 5% less probes to achieve a 90% estimation accuracy for latency and loss rate measurements. However, there is no much improvement to be mentioned regarding bandwidth. This is related to the way path qualities are calculated from the bounds on the qualities of their constituent path segments, see section 4.3. So the fewer the number of segments, the better the accuracy for latency and loss rate estimations, but that is not necessarily true for bandwidth, because it uses minimization.

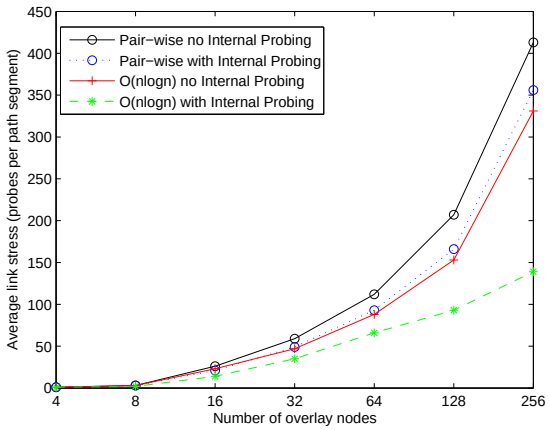


Figure 4.6 Effect of overlay network size on link stress (x axis in log scale). The $O(n \log n)$ algorithm with 90% accuracy.

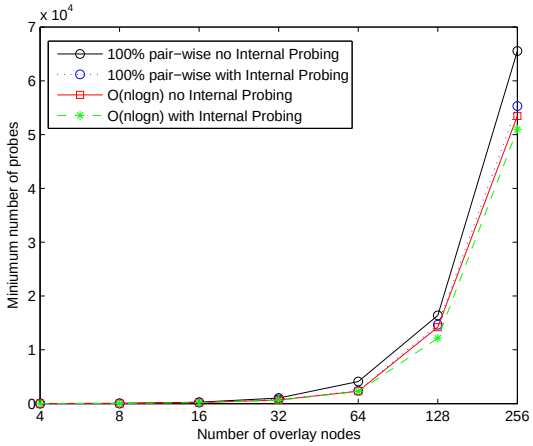
Monitoring results aggregation and distribution: It has been shown in a related study [28] that the overhead and link stress associated with aggregating and disseminating monitoring information is small compared to the overall monitoring overhead. Tang and Mckinley [28] have formulated such a problem and proposed several heuristic solutions. For example, their worst algorithm reported a worst case bandwidth consumption of 300KB, while their best algorithm achieved about 60KB worst case bandwidth consumption. Such research problem is important, but it is not the subject of this dissertation.

4.5 Conclusion

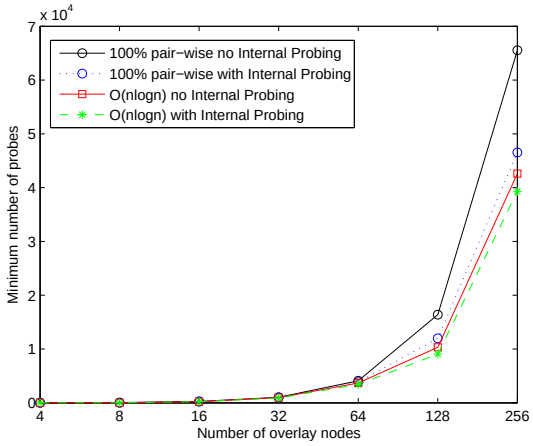
In this chapter, the issue of reducing link stress in topology-aware overlay networks was addressed. To achieve this objective, the *Internal Probing* scheme was proposed. Our approach employed readily available physical (*i.e.,* network-level) topology information coupled with existing probing algorithms to provide intelligent probing that avoids highly shared links. Simulation studies showed that the Internal Probing approach offer significant benefits over the traditional topology-oblivious probing methods. However,

such improvement needs to be balanced with the amount of overhead introduced in communicating and aggregating probing results from various partitions in the overlay network, and the probing capabilities of the intermediate (physical) nodes.

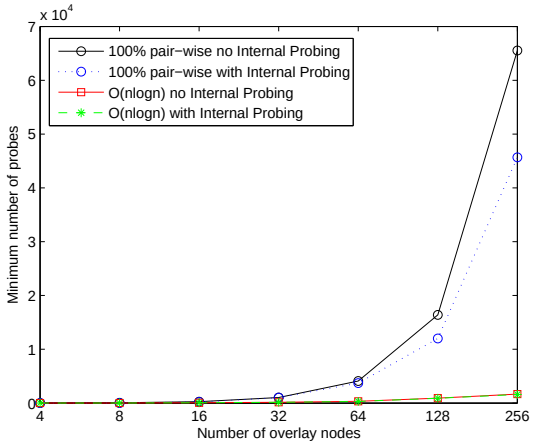
While our results showed that this approach already yields significant improvements, there are several directions that are possible in order to advance this work. One such direction is to study the effect of reduced buffering overhead caused by Internal Probing, and how it will affect the reported measurement results.



(a) Loss rate estimation



(b) Latency estimation



(c) Bandwidth estimation

Figure 4.7 Minimum number of probes required to achieve a 90% estimation accuracy (x axis in log scale). Also shown, the number of probes used by pair-wise probing with and without Internal Probing to achieve 100% accuracy.

CHAPTER 5. Network Fault Location

Accurate fault detection and location affects the performance of Internet applications (e.g., VoIP, video streaming, and online gaming), and ISP networks as a whole [77][78]. There is an ever increasing need to reduce rerouting and maintenance times. In order to achieve stable operating environments for ISP networks and the kind of emerging applications they support, we need to accurately locate IP links faults.

IP links faults can be caused by many factors, such as fiber cuts, router crashing or misconfiguration, very heavy congestion, or maintenance activities causing unintentional effects. These kinds of failures occur on a daily basis [79], and they may affect packet forwarding even with the existence of backup paths due to the overlap among network paths [80].

The process of fault monitoring goes through three steps. The first step is fault detection, which is done through IP-level management agents via management protocol messages (e.g., SNMP trap and CMIP EVENT-REPORT), or application-level overlay monitoring [81]. These agents generate a set of alarms. After that, fault identification through alarm correlation is performed. The output of the second step is a set of possible fault scenarios. The majority of fault identification algorithms and systems [81, 84] rely on spatial correlation of observed symptoms and possible fault scenarios. These systems typically generate a set of equally plausible fault locations. This set of possible faults is non trivial due to lost and spurious symptoms, the overlap among network paths, and the network heterogeneity. The final step, which traditionally has been done by the network operators or administrators, is fault verification through debugging. Fault

verification locates the exact faulty link(s). The focus of this research is fault verification using overlay networks to achieve accurate and fast IP links fault location.

The proliferation of infrastructure overlay networks allows the network administrator to deploy short-term and long term network solutions with great versatility and flexibility. Such overlays present a tremendous opportunity for the verification of suspected faulty IP links. However, several issues arise in the design of such overlays due to the IP and overlay link sharing among overlay paths. One of the major concerns is the IP link stress (i.e., number of packets crossing the same link). Such stress can cause measurement conflict [65], which in turn leads to congestion and packet loss. Thus, adding further confusion rather than verifying the faulty IP links. In addition, overlay paths may have varying properties such as the underling IP hop count or bandwidth capacity.

The contributions of this chapter are as follows:

- The problem of overlay design for IP links fault verification is formulated as a minimum cost link stress constrained path selection problem, and the corresponding flow network is constructed.
- The complexity of the problem is proven to be NP-hard.
- The IP links coverage of various overlay sizes and topologies is studied using real-life Internet topologies.
- Several interesting research problems are identified in this context.

The remainder of this chapter proceeds as follows. Section 5.1 gives a motivational example and explores the problem space. Section 5.2 discusses the related work. Section 5.3 presents our network model and its assumptions. The overlay design problem flow network formulation along with its complexity is presented in section 5.4. Section 5.5 presents the performance evaluation results. We conclude in section 5.6.

5.1 Motivation

In this section, some of the issue involved in the design of overlay networks for IP links fault verification are exposed, the problem space is explored. There are two aspects of the problem: how do we obtain IP links fault information? And how do we verify such information and what are the various performance issues that should be considered in this context?

Fig. 5.1 shows an example overlay network and the underlying IP network [86]. Let us assume that the overlay monitoring algorithm has chosen overlay paths E-C-A and B-A as a subset of paths that will cover most of the underlying IP links, or for any other consideration. The properties of the remaining overlay links can be calculated (*e.g.,* link AC), estimated [32], or probed explicitly (*e.g.,* link DC).

Let us go through a scenario wherein a fault has occurred in an IP link, which caused path $E - C - A$ to go down, then consider the following:

- Performing spatial correlation [84] using overlay paths $E - C - A$ and $B - A$, and the underlying IP links, will generate a set of equally "good" suspected list of faulty IP links. This set includes links $E - N_2$, $N_2 - C$, and $C - N_1$. Note that this set could also have been generated by a management system at the IP layer.

- A network administrator will need to debug these suspected faulty IP links. Either by checking each link physically one by one (i.e., white box testing), or through end-to-end measurements (i.e., black box testing).

- Probing overlay paths $D - C$ and $C - A$ would be sufficient. If path $D - C$ is faulty then so is link $N_2 - C$, and if path $C - A$ is faulty then link $C - N_1$ is faulty. If both paths are working then the assumption is that link $E - N_2$ is faulty.

From this simple example, we can identify several issues involving this problem:

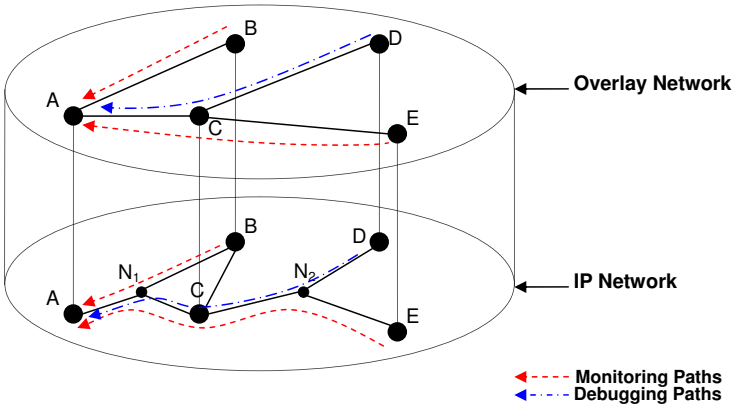


Figure 5.1 A motivational example.

1. An awful choice would have been for node B to probe nodes C and D. Since there is no overlay link between nodes B and C, and between nodes B and D. The paths will have to go through node A first (because of overlay routing), which will result in a higher link stress. For example, link $A - N_1$ would have a stress of 4 probes, and link $N_1 - C$ would have a stress of 2 probes. Such stress can cause measurement conflict [65], which in turn leads congestion and packet loss. Thus, adding further confusion rather than verifying the faulty IP links.

2. The overlay needs to cover the underlying IP network or the portion containing the suspected list of IP links. This coverage need to be sufficient for fault debugging (i.e., sufficient number of good paths).

3. Different overlay paths may have varying costs (e.g., hop count) and/or bandwidth capabilities. Such costs need to be minimized and/or the bandwidth constraints respected.

4. The number of overlay nodes involved in the measurement needs to be minimized. This has an effect on the management overhead.

In this chapter, the problem of choosing the overlay paths with minimum cost subject to IP links stress constraints is addressed assuming that we have a set of overlay nodes to choose from. The link stress constraint will capture the measurement conflict, path disjointedness, and bandwidth capacities of the underlying IP links. For example, a stress bound of 1 on each IP link means that all the overlay paths need to be completely disjoint.

5.2 Related Work

There have been many attempts to study the occurrence of failures in backbone networks. In [77] the authors study the impact of failures in Sprint's IP backbone on VoIP services. They observed that failures occur on a daily basis and they have a tangible impact on the operation of backbone networks. In another study [79], the authors classified probable cause of such failures. They had found that 49% of all failures affect a single link at a time, 21% of failures are caused by router-related problems or optical equipments, and 20% is due to planned maintenance activities.

The area of fault detection and localization has been very active in the past decade or so. An excellent survey of such algorithms has been presented by Steinder and Sethi in [81]. SCORE [84, 93] is one of the most recent related studies. In SCORE, spatial correlation on a bipartite fault graph is used to identify fault hypotheses that best explain the failure signature with the least number of candidate faults. Such systems where used to detect black holes in MPLS networks [93], or link fault detection in ATM networks. In another paper, wang and Al-Shaer [95], extend the bipartite fault propagation graph to a probabilistic model that can be used to handle lost and spurious symptoms. We use such spatial correlation engines as an input to our problem.

In the design of monitoring overlays literature, Cantieni et al. [90] revisit the problem of monitor placement, they propose an optimal algorithm to choose which monitors to

activate and at what sampling rate in order to achieve a given measurement accuracy, as opposed to maximizing the fraction of IP flows being monitored proposed by Suh et al. [87]. In our study, we use similar assumptions regarding the available IP network information. In another steady, Bejerano and Rastogi [88] propose a two phase approach to minimize the monitoring infrastructure cost and the probing overhead of link delays measurements even in the presence of link faults. Our contribution is different in that we aim at pinpointing the faulty link(s) out of the suspected faulty ones, while theirs is to ensure delay measurements are possible even in the presence of failures through bypassing suspected faulty links irrespective of their accurate location.

Traditionally, the problem of fault verification or debugging has been conducted by the network administrator. Approaches such as using the *traceroute* tool to locate the faulty links by traversing the faulty paths cause a large amount of stress, especially close to the source of these traceroute packets, due to the way traceroute works. Also, they will take longer time to conduct in comparison to our proposed approach. In our approach, we aim at using existing overlay measurement paths to locate these faulty links. Thus, achieve fast location, while respecting the cost and stress constraints of the underlying IP network.

5.3 Network Model and Assumptions

Throughout this chapter, the following model is used and assumptions that are commonly used in the literature are made:

1. We represent the network as an undirected graph $G = (V, E)$, where V is the set of physical nodes and E is the set of physical edges. From this graph a set of network nodes $V_o \subseteq V$ are available to be used in network debugging.

2. The routing paths from each node in V_o to every other node in V_o are known. This routing information is represented as a matrix $R = [r_{ij}]$, where r_{ij} is defined by:

$$r_{ij} = \begin{cases} 1 & \text{if path } i \text{ traverses edge } j \\ 0 & \text{otherwise} \end{cases}$$

Several related studies make this assumption [65, 87, 88].

3. We are given the set of suspected faulty IP links $F = \{f_1, f_2, \ldots, f_{|F|}\}$, where $F \subset E$. This set is generated by a fault detection and alarm correlation system. These alarms are produced by management agents via management protocol messages (e.g., SNMP trap and CMIP EVENT-REPORT) operating at the IP level, or using overlay monitoring as in Fig. 5.1. This suspected list will need to be debugged and verified operational.

4. The vector $D = [d_i]$ specifies the bound on the link stress experienced by each link $i \in E$. The bound is in terms of the number of paths using this link, but can also be in terms of the number of probes or the bandwidth allocated for conducting measurements on each link. By definition, the stress on faulty links should be 1 for the debugging to be correct (i.e., the debugging path should pass through only one suspected faulty link at a time).

5. The set of non-faulty IP links is $L = \{l \mid l \in E \setminus F\}$.

6. The set of debugging paths is $P = \{p_1, p_2, \ldots, p_{|P|}\}$, contains those overlay paths in $|V_o \times V_o|$ that pass through the set of faulty links. No path in P traverses two faulty links at the same time, as it will violate the stress constraint.

7. Each source-destination path $i \in |V| \times |V|$ is associated with a non-negative cost. This cost could represent the number of hops in each path, the bandwidth cost, or the operational cost of that path depending on its geographical location or accessibility [87]. For the sake of formulation simplicity, the path costs will be expressed in terms of link costs κ_i, $\forall l_i \in L$

5.4 Problem Formulation

Given the network graph G, the set of suspected faulty links F, the sets V_o and L, and links costs. The goal is to find the set paths that can debug the suspected faulty links at a minimum cost while respecting the link stress constraints for each link incident to the debugging paths.

This problem is formulated as a integer generalized flow problem [89] by constructing a flow network $G_f = (V_f, E_f)$, as follows:

1. Add a node for each suspected faulty link in F, a node for each debugging path in P, and a node for each IP link that we are interested in bounding its stress. In addition, add a dummy source s and a dummy terminal t.

2. Add an edge with (cost, upper bound, multiplier) label of (0, 1, 1) between the source s and every other faulty link node $f_i \in F$.

3. Add an edge with $(0, 1, |p_j|)$ label between each faulty link node $f_i \in F$ and the nodes $p_j \in P$ representing its debugging paths. $|p_j|$ is the number of links in path p_j.

4. Add an edge with $(0, 1, 1)$ label between each debugging path node and the non-faulty links' nodes $l_i \in L$ that are incident to the corresponding path.

5. Add an edge with $(\kappa_i, d_i, 1)$ label between each node representing links in L and the terminal node t. d_i represent the bound on the stress experienced by link i in terms of the number of paths sharing that link, and κ_i is the cost of link l_i.

The flow network $G_f = (V_f, E_f)$ corresponding to the minimum cost link stress constrained overlay design for fault verification problem is shown in Fig. 5.2. It is clear that finding the minimum cost maximum flow in this network will correspond to finding the least cost maximum number of debugging paths (i.e., one for each suspected

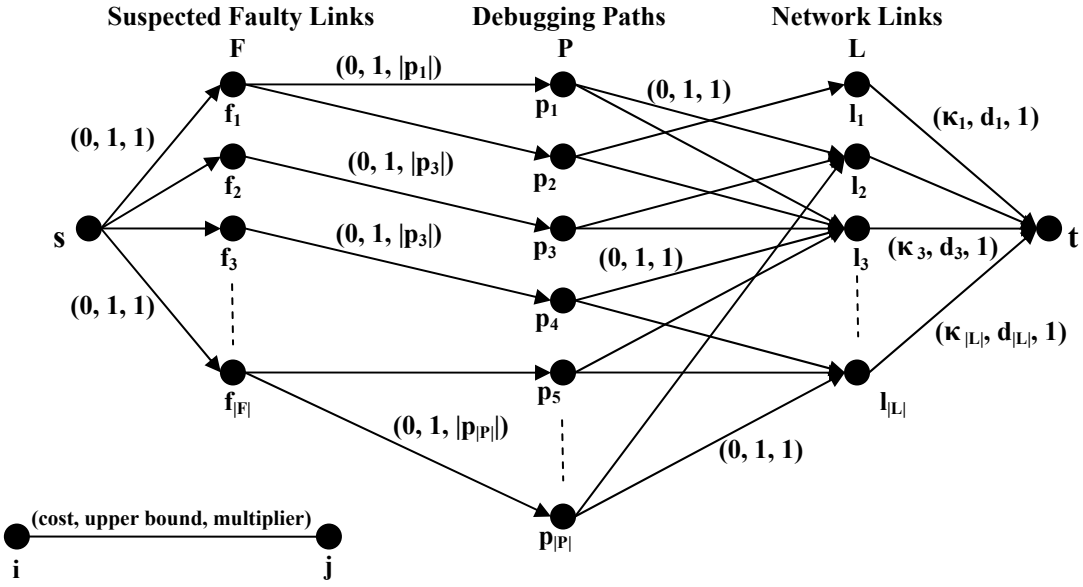


Figure 5.2 The flow network corresponding to the fault debugging problem. Not all edges are labeled to keep the figure clear.

faulty link), while respecting the link stress constraints. Note that the bound on the flow leaving the l nodes will limit the number of departing flow units. Hence, by flow conservation, the number of incoming flow units (i.e., paths using that same link in L) is also bounded by the stress constraint. Each $s - f_i$ edge carrying a flow means that fault f_i is verifiable by the path represented by node p_j that has the flow value on $f_i - p_j$ equal to 1.

5.4.1 Problem Complexity

The problem of minimum cost link stress constrained overlay design for fault verification is proven to be NP-hard by reduction from the 3-cover problem [62][89], as follows: The 3-cover problem is a known NP-hard problem.

Instance: Given a collection of m (possibly overlapping) sets $S_1, S_2, \ldots, S_m$ each with three elements drawn from the set $N = \{1, 2, 3, \ldots, n\}$.

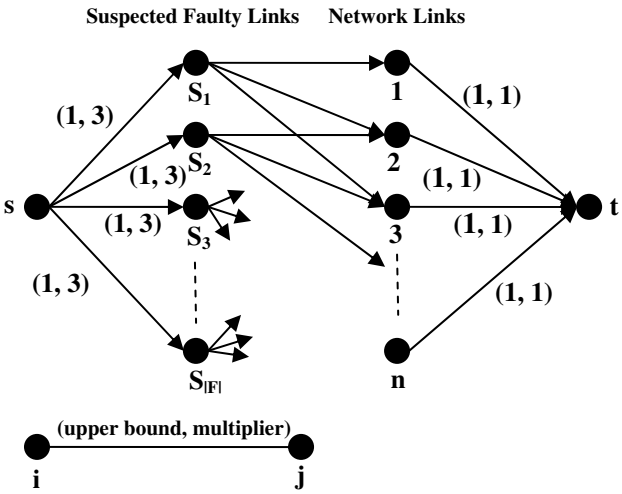


Figure 5.3 The reduction of the minimum cost link stress constrained fault verification problem from the 3-cover problem.

Question: Does there exist $n/3$ pairwise disjoint sets from this collection such that their union is equal to N.

Proof: We consider a simple instance of the minimum cost link stress constrained fault verification problem, where there are no costs and the link stress constraint is 1 for all links (i.e., disjoint debugging paths). Further assume that each fault is verifiable by one path only, which allows us to merge the f_i nodes with the corresponding p_j nodes: $\forall f_i \in F,\ \forall p_j \in P$, if $(f_i, p_j) \in E_f$, then merge f_i and p_j. Also, assume that each path consists of 3 links. We wish to find if at least ν faults can be verified.

Fig. 5.3 shows the flow network corresponding to the reduction from the 3-cover problem. In this network, $S_i = f_i\ \forall f_i \in F$, $|F| = m$, $|L| = n$, each link $|l_i| \in L$ will correspond to a number $i \in N$, and $\nu = n/3$ or the total flow value is n.

5.4.2 Relaxing the Stress constraint

Since the problem is NP-hard, we look for a relaxation of the problem that will make it computationally tractable. We choose to relax the link stress constraint, and

reformulate the problem with path costs λ_i, as a minimum cost circulation problem [89]. Fig. 5.4 shows the corresponding flow network. The main differences in this flow network are the removal of the path multipliers, and that we add an edge from t to s with capacity $|F|$, and a cost of $-C$, where C is larger than any path cost. It can be shown that finding the minimum cost circulation will correspond to finding the maximum number of least cost debugging paths that can verify the most possible number of suspected faulty links. The maximum flow (i.e. maximum number of faults to be verified) is achieved because of the very large negative cost on the $t - s$ edge. The more flow that is pushed on the $s - f_i$ edges, the least the cost will be, till we reach the upper bound of $|F|$ (i.e., the number of suspected faults) if possible, then the circulation stops. Each $s - f_i$ edge carrying a flow means that fault f_i is verifiable by the path represented by node p_j with the on $f_i - p_j$ equal to 1. The minimum cost is guaranteed by the elimination of all negative cost cycles (i.e., faults verifiable with less cost) in the residual network. Such a proof is well-known in the optimization literature [89].

Building upon this construction, we map the problem into an LP (Linear Program), which can be solved using common optimization tools (e.g., CPLEX [74]), as follows:

Let $G_f = (V_f, E_f)$ be the flow network in Fig. 5.4, let x_{ij} be the flow value along edge $(i, j) \in E_f$, c_{ij} is the cost per unit flow along edge (i, j), and u_{ij} is its upper bound, or the edge's capacity. Then, the corresponding LP formulation is:

Minimize

$$\sum_{(i,j)\in E_f} c_{ij}x_{ij}$$

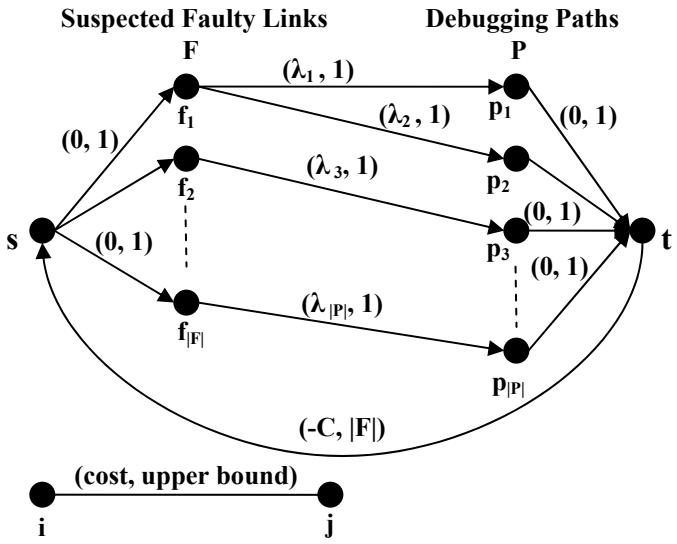


Figure 5.4 Relaxing the link stress constraints allowed us to model the problem as a minimum cost circulation.

Subject to

$$\sum_j x_{ij} = \sum_j x_{ji} \ , \ \forall i, j \in V_f \tag{5.1}$$

$$x_{sf_j} = \{0, 1\} \ \forall (s, f_j) \in E_f \tag{5.2}$$

$$x_{f_i p_j} = \{0, 1\} \ \forall (f_i, p_j) \in E_f \tag{5.3}$$

$$x_{p_i t} = \{0, 1\} \ \forall (p_i, t) \in E_f \tag{5.4}$$

$$0 \leq x_{ts} \leq |F| \tag{5.5}$$

In the above formulation, constraint 1 is the flow conservation constraint, constraints 2-4 are binary constraints (e.g., either use the link in the flow network or do not), and constraint 5 states that we can not verify more faults than what is given.

5.5 Performance Evaluation

The problem is studied on real Internet ISP topologies of two major U.S. carriers provided by RocketFuel [72]: AT&T (AS# 7018, 11800 nodes), and Sprint (AS# 1239,

10332 nodes). In addition, two smaller networks are considered: Ebone (AS# 1755, 300 nodes), and Above (AS# 6461, 654 nodes). A shortest path algorithm, based on hop count, is used for the IP-layer routing.

The set of overlay nodes is uniformly selected at random. The number of overlay nodes (N) is set to 16 and 128. The overlay topology is varied in two ways: a complete graph (denoted as complete), where each nodes maintains overlay links to every other node, and a random graph (denoted as log) where each node maintains $\log_2 N$ neighbors.

Coverage of IP links

The first aspect that should be considered when choosing the set of overlay nodes is how many IP links to they cover? Fig. 5.5 shows the percentage of the covered IP links against the number of overlay nodes and for the two overlay topologies previously mentioned. The results show that there is not much difference, in terms of coverage, between the *log* overlay and the complete overlay, as both achieve comparable results. In general, the percentage is good (i.e., 25-35% for 128 overlay nodes), however, it is so because of the small size of the underlying IP network. For the larger AT&T and Sprint networks, the percentage was close to 5% for 128 overlay nodes. The conclusion to be drawn here is that the choice of the overlay nodes (an ongoing and future research problem) should be dynamic and changing with the set of suspected faults.

Debugging Faulty Links

How much path disjointedness is it possible to get from an available set of overlay paths? Remember that in order to verify any two links we need two paths such that each path contains one of the links, but not the other. We show the results for the AT&T network, as the other networks follows the same trend. We set the number of links requiring debugging (from the currently covered set of IP links) to 5, 10, 15, and 20. The cost of the path is set to its IP hop count. Fig. 5.6a shows the normalized

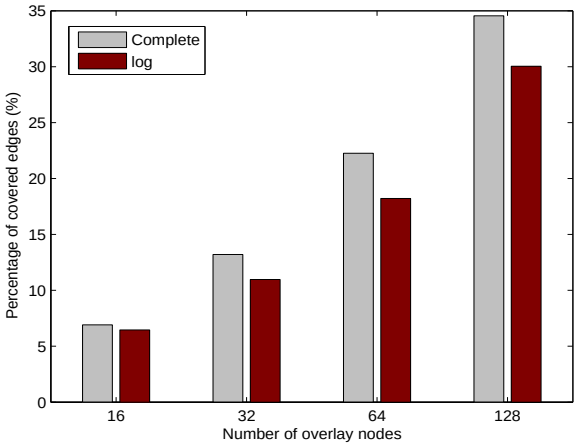
average number of successfully debugged IP links versus the number of suspected IP links for a complete overlay of sizes 16, 32, and 64 nodes. While Fig. 5.6b shows the same results for a random graph (denoted as log) for the same number of nodes. This figure shows that even though there is significant overlap in the overlay, we are still able to find a very good number of disjoint paths. For example, for 5 suspected links, close to 90% of the links were debugged successfully using a complete overlay of size 64 or 32, the percentage drops to 68% for the humble overlay size of 16. As for the random log graph, the numbers are slightly less impressing as compared to the complete graph.

5.6 Conclusion

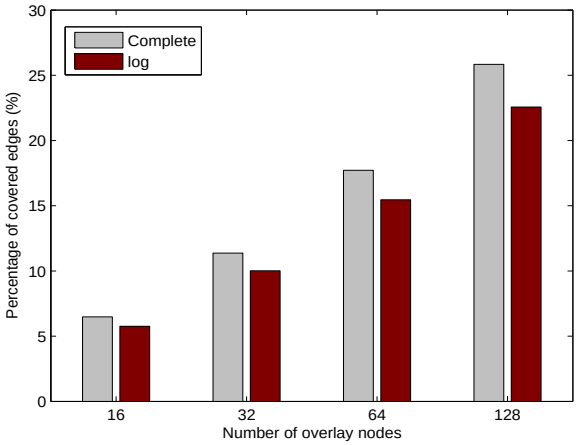
In this chapter, the problem of verifying IP links faults using overlay networks monitoring was introduced. The problem of verifying the maximum number of faults using the least cost paths under link stress constraints was proven to be NP-hard. In addition, a relaxation of the link stress constraints was given and the maximum number of faults that can be verified using the least cost paths was studied.

using experimental results, it was shown that it is possible to achieve good verification capabilities with a small number of overlay nodes (e.g., 64 nodes) due to the path diversity provided by the overlay network. However, the number of IP links covered by the overlay is small. Hence, dynamic selection of the overlay nodes, based on the network topology and the set of suspected faulty links, is needed.

For future work, heuristic solutions to the fault verification problem, under the stress constraints, can be developed. More importantly, choosing the set of overlay nodes to be used for fault verification, in addition to the network topology, is a problem worth investigating.

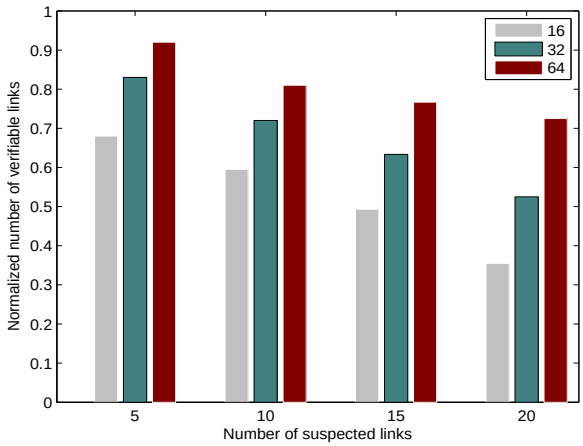


(a) Ebone

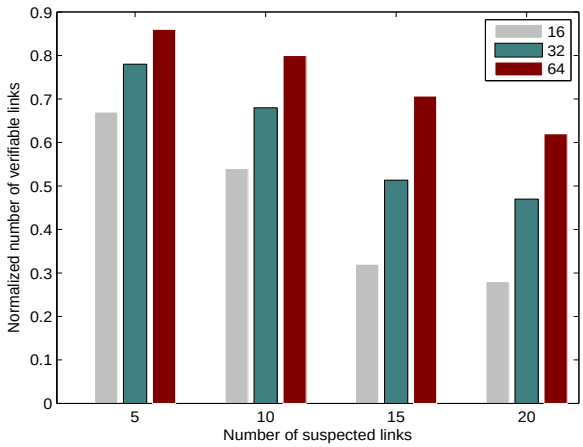


(b) Above

Figure 5.5 The percentage of IP links covered by the overlay network for various number of overlay nodes, and network topologies.



(a) A complete overlay network, with number of nodes 16, 32, and 64.



(b) A random overlay network, where each node maintains $\log_2 N$ neighbors.

Figure 5.6 The average link debugging capability for various sizes and topologies of the overlay network.

CHAPTER 6. Conclusions and Future Work

In this dissertation, the usefulness and feasibility of combining information about measurement tools and overlay monitoring protocols was demonstrated. The overall goal was to achieve accurate monitoring with low impact on the underlying IP network, and to use monitoring paths in accurate and timely IP links fault location. This goal was achieved by taking under consideration the various topological properties of the overlay and timing conflicts among different measurement tools. Our contributions was as follows:

1. The amount of overlap among different overlay paths was studied on real-life Internet topologies for various overlay sizes and topologies. For example, for a complete overlay network of 64 nodes over the AT&T network, 65% of links had participated in about 25 paths or less. Such numbers were sufficient to significantly degrade the quality of measurements.

2. A monitoring overlay on top of PlanetLab was constructed to demonstrate the effect of the measurement conflict problem on the accuracy of bandwidth estimation. Using the Iperf bandwidth measurement tool, the reported bandwidth values were 70% and 25% of the accurate bandwidth value for 5 and 10 conflicting measurement tasks respectively.

3. The problem of conducting conflict-free measurements was formulated as a scheduling problem of real-time tasks, and the complexity of the problem was proven to be NP-hard. The temporal and spatial properties of the overlay network were used

to develop scheduling heuristics that achieve up to 25% better schedulability over the existing algorithm.

4. The concept of internal probing was proposed to minimize the monitoring overhead, while capping the link stress caused by the overlay monitoring algorithms. Such an approach achieves 30% lower link stress than the topology-oblivious monitoring algorithms.

5. The validity and performance of using common overlay networks topologies in network fault location was evaluated.

6. Problems related to verifying IP links faults using overlay networks under a set of cost and link stress constraints were formulated and the complexity of the problem was proven to be NP-hard.

7. Using real-life Internet topologies to evaluate the performance of the proposed solutions, it was found that a typical overlay topology can verify up to 95% of IP links faults.

Broader Impact

Beside the intellectual and academic contributions of this dissertation, the following impact was expected on the research community and industry:

- Content Distribution service providers like Akamai (45000 globally distributed servers, contributing 20% of the Internet traffic) can benefit from measurement scheduling algorithms in orchestrating their continuous service-monitoring activities to achieve more accurate measurement results, which in turn translates into better services and more revenue. Other than the affect on measurement accuracy, controlling the link stress could be beneficial for this kind of overlays for

other reasons; In particular, the pricing of the underlying IP links may vary, and thus, high-priced links should experience less link stress.

- Internet Service Providers (ISPs) can benefit from the verification of IP links faults using monitoring overlays in a manner that allows for the fast and accurate fault detection, location, and maintenance. In addition, the same monitoring benefits enjoyed by the content distribution networks above can apply to ISPs.

Future work

The work presented in this dissertation opens up research in the following directions.

- Employing concepts from the real-time scheduling literature in achieving more efficient schedulability of measurement tasks. In particular, the imprecise computation model [69] can be used to allow measurement tasks to partially overlap, thus reducing the schedule time and allowing more tasks to execute, albeit at the expense of a reduction in the measurement accuracy depending on the amount of overlap. In addition, real-time multiprocessor scheduling algorithms can be adapted in a novel manner to fit the measurement scheduling problem.

- It was demonstrated that the accuracy of the measurement tasks degrades as the overlap in time and space increases. It would be appealing to study in more depth, how the behavior of dependent application will differ from the no-conflict case.

- Design and analysis of multi-source multi-destination bandwidth measurement tools. Does this approach achieve better performance than dividing the problems into monitoring protocols and measurement tools? Such a question is yet to be answered.

BIBLIOGRAPHY

[1] S. Deering, and D. Cheriton. Multicasting routing in datagram internetworks and extended LANs. *ACM Transactions in Computer Systems, 8(2):85-110*, May 1990.

[2] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefe. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1):88-98, January 2000.

[3] Chiping Tang. *Underlay-aware overlay networks*. Ph.D. dissertation, Michigan State University, 2005.

[4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. *Proceedings of the* 18^{th} *SOSP*, 2001.

[5] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for End System Multicast. *IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 8, October 2002.

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. *In Proceedings of ACM SIGCOMM'01*, August 2001.

[7] Javed I. Khan, and Adam Wierzbicki. Foundation of Peer-to-Peer Computing. *Sepcial Issue, Elsevier Computer Communications* 31 (2008) 187189.

[8] Napster (Last accessed October 20, 2008). http://www.napster.com.

[9] The Gnutella protocol specification (Last accessed October 20, 2008). www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[10] A. Rowstron, and P. Druschel. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. *In Proceedings of IFIP/ACM Middleware 2001*, November 2001.

[11] Suman Banerjee, Bobby Bhattacharjee, Christopher Kommareddy. Scalable Application Layer Multicast. *In proceedings of the ACM SIGCOMM*, 2002.

[12] Jorg Liebeherr, Tyler K. Beam. HyperCast: A protocol for maintaining multicast group members in a logical hypercube topology. *In Networked Group Communication*, 1999, pp 72-89.

[13] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. *In proceedings of OSDI*, 2000.

[14] Yatin Chawathe. Scattercast: an adaptable broadcast distribution framework. *In Multimedia Systems*, vol. 9, Issue 1, pp. 104-118, 2003.

[15] Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, and Samir Khuller. Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications. *In Proceedings of IEEE Infocom*, 2003.

[16] BitTorrent - The global standard for accessing rich media content over the Internet (Last accessed October 20, 2008). http://www.bittorrent.com/.

[17] Guillaume Pierre, Maarten Van Steen, and De Boelelaan. Design and implementation of a user-centered content delivery network. *In Proceedings of 3rd Workshop on Internet Applications*, 2003.

[18] Kenneth L. Calvert, James Griffioen, and Su Wen. Lightweight network support for scalable end-to-end services. *In ACM SIGCOMM Computer Communication Review*, Vol. 32, Issue 4, 2002.

[19] Alessio Botta, Antonio Pescape, and Giorgio Ventre. On the performance of bandwidth estimation tools. *In proceedings of the IEEE Systems Communications*, 2005.

[20] W. Cui, I. Stoica, and R. H. Katz. Backup path allocation based on a correlated link failure probability model in overlay networks. *In Proceedings of IEEE ICNP*, November 2002.

[21] Akamai Technologies. Freeflow content delivery system (Last accessed October 20, 2008). http://www.akamai.com.

[22] Vinay J. Ribeiro, Rudolf H. Riedi, and Richard G. Baraniuk. Spatio-Temporal Available Bandwidth Estimation for High-Speed Networks. *ISMA 2003 Bandwidth Estimation Workshop (Best)*, CAIDA, La Jolla, CA, December 2003.

[23] Kevin Lai, and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. *In Proceedings of ACM SIGCOMM*, 2000, pp. 283-294.

[24] Jun Li, Peter L. Reither, and Gerald J. Popek. Resilient self-organizing overlay networks for security update delivery. *In IEEE Journal on Selected areas in Communication (JSAC), Special Issue on Service Overlay Networks,* Vol. 22, No. 1, January 2004.

[25] P. Calyam, C. Lee, P. Arava, and D. Krymskiy. Enhanced EDF scheduling algorithms for orchestrating network-wide active measurements. *In Proceddings of the* 26^{th} *IEEE RTSS*, December 2005.

[26] C. Tang and P. K. Mckinley. On the cost-quality tradeoff in toplogy-aware overlay path probing. *In Procedings of IEEE ICNP*, 2003.

[27] M. Kwon, and Sonia Fahmy. Path-aware overlay multicast. *In Computer Networks, Vol. 47, Issue 1, pp. 23-45, 2005.*

[28] Chiping Tang, and P. K. Mckinley. A distributed approach to topology-aware overlay path monitoring. *In Proceedings of the 24th International conference on Distributed Computing Systems (ICDCS 2004)*, March 2004.

[29] L. Qiu, Y. Yang, Y. Zhang, and S. Shenker (ICSI). On Selfish Routing In Internet-like Environments. *ACM SIGCOMM 2003.*

[30] Anirban Chakrabarti, and G. Manimaran. Reliability constrained routing in QoS networks. *IEEE/ACM Transactions on Networking (TON) 13(3):662-675 (2005).*

[31] R. Caceres, N. G. Duuffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions in Information Theory, Vol. 45, pp. 2462-2480, 1999.*

[32] T. Bu, N. Duffield, F. Presti, and D. Towsley. Network tomography on general topologies. *In Proceedings ACM sigmetrics*, 2002.

[33] M. Rabbat, R. Nowak, and M. Coates. Multiple source, multiple destination network tomography. *In Proceedings of IEEE Infocom*, 2004.

[34] M. Coates, A. Hero, and R. Nowak, B. Yu. Internet tomography. *IEEE Signal Processing Magazine, 2002.*

[35] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebric approach to practical and scalable overlay network monitoring. *In Proceedings of ACM SIGCOMM*, August 2004.

[36] T. Ho, B. leong, Y Chang, Y. Wen, and R. Koetter. Network monitoring in multicast networks using network coding. *In Proceedings of the International Symposium on Information Theory (ISIT)*, 2005.

[37] Christina Fragouli, and Athina Markopoulou. A network coding approach to overlay network monitoring. *In Proceedings of 43rd Allerton Conference on Communication, Control, and Computing*, Sep. 2005.

[38] Y. Zhu, B. Li, J. Guo. Multicast with network coding in application-layer overlay networks. *In IEEE JSAC, Special Issue on Service Overlay Networks*, 4th Quarter, 2003.

[39] B. Melander, M. Bjorkman, and P. Gunningberg. A new end-to-end probing and anlaysis method for estimation bandwidth bottlenecks. *In Proceedings of IEEE GLOBECOM*, 2000, pp. 415-420, San Francisco, CA, USA.

[40] J. Navratil, and R. L. Cottrell. ABwE: A practical approach to available bandwidth estimation. *In Proceedings of Passive and Active Measurements (PAM) workshop*, 2003, La Jolla, CA, USA.

[41] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cotrell. Pathchirp. Effecient available bandwidth estimation for network paths. *In Passive and Active Measurement Workshop*, 2003.

[42] M. Jain, and C. Dovrolis. End-to-end available bandwidth measurement methodology, dynamics, and relation with TCP throughput. *In proceedings of SIGCOMM*, 2002.

[43] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. *In proceedings of Internet Measurement Conference (IMC)*, October 2003.

[44] Nangning Hu and Peter Steenkiste, Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, August, 2003.

[45] Traceroute Reno (Last accessed October 20, 2008). http://www.psc.edu/networking/treno_info.html

[46] Network performance benchmark (Last accessed October 20, 2008). http://www.netperf.org/netperf/NetperfPage.html

[47] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network, Nov.-Dec. 2003.*

[48] C. Dovrolis, P. Ramanathan, and D. Moore. What do Packet Dispersion Techniques Measure?. *In Proceedings of IEEE INFOCOM*, 2001, pp. 905-914.

[49] H. Cenk Ozmutlu, Natarajan Gautam, and Russell Barton. Managing End-to-End Network Performance via Optimized Monitoring Strategies. *Journal of Network and Systems Management*, pp. 107-126, 2004.

[50] Aun Haider, and Richard Harris. Recovery Techniques in Next Generation Networks. *IEEE communications surveys*, vol 9, No. 3, 2007.

[51] T. Zseby, S. Zander, G. Carle. Evaluation of Building Blocks for Passive One-Way-Delay Measurements. *In proceedings of Passive and Active Measurements*, Amsterdam, 2001.

[52] Chiping Tang, and P. K. Mckinley. A Distributed multipart computation framework for overlay network applications. *Technical report MSU-CSE-04-18, department of Computer Science and Engineering, Michigan State Univeristy*, May 2004.

[53] Rich Wolski, N. Spring, and Jim Hayes. The Network weather service: A Distributed Resource Performance Forecasting Service for Metacomputing. *In Journal of Future Generation Computing Systems*, Vol. 15, No. 5-6, pp. 757-768, Oct. 1999.

[54] B. Gaildioz, R. Wolski, and B. Tourancheau. Synchronizing network probes to avoid measurement intrusiveness with the Network Weather Service. *In proceedings of IEEE High-performance Distributed Computing Conference*, 2000.

[55] Prasad Calyam, Chang-Gun Lee, P. K. Arava, Dima Krymskiy, and David Lee. OnTimeMeasure: A Scalable Framework for scheduling active measurements. *IEEE Workshop on End-to-End Monitoring Techniques and Services, 2005.*

[56] A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, and K. Ramakrishnan. An OSPF topology server: Design and evaluation. *JSAC: Special Issue on Recent Advances on Fundamentals of Network Management*, May 2002.

[57] M. Halldorsson, and J. Radhakrishnan. Greed is Good: Approximating Indpendent Sets in Sparse and Bounded Degree Graphs. *In proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, 1994.

[58] PlanetLab (Last accessed October 20, 2008). http://www.planet-lab.org/

[59] A. Tirumala, L. Cottrel, and T. Dunigan. Measuring end-to-end bandwidth with Iperf using Web100. *In Passive and Active Measurements Workshop*, 2003.

[60] M. Allman. Measuring End-to-End Bulk Transfer Capacity. *In proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001, pp. 139-143.

[61] Rohit Kapoor, Ling-Jyh Chen, L Lao, Mario Gerla, and M. Y. Sanadidi. CapProbe: A simple and Accurate Capacity Estimation Techniques. *In proceedings of ACM SIGCOMM*, 2004.

[62] M. R. Garey, and D. S. Johnson. Computers and Intractability: A guide to the theorey of NP-completeness. *W. H. Freeman*, 1979.

[63] Limin Wang, Vivek Pai, and Larry Peterson. The effectiveness of Request Redirection on CDN Robustness. *In proceedings of OSDI*, 2002.

[64] Ming Zhang, Chi Zhang, Vivek Pai, Larry Peterson, and Randy Wang. PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-area Services. *In proceedings of OSDI*, 2004.

[65] M. Fraiwan, and G. Manimaran. On the Schedulability of Measurement Conflict in Overlay Networks. *In proceedings of IFIP Networking*, 2007: 1120-1131.

[66] C. L. Liu, and Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the Association for Computing Machinery* Vol. 20, 1973, pp. 46-61.

[67] Lui Sha, John Lehoczky, and Rangunathan Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling. *In 7th IEEE Real-Time Systems Symposium (RTSS)*, 1986.

[68] Y. Shavitt, X. Sun, A. Wool, and B. Yener. Computing the unmeasured: An algebraic approach to Internet mapping. *In Proceeding of IEEE INFOCOM*, 2001.

[69] I. K. Cheong. Scheduling imprecise hard real-time jobs with cumulative error. *Technical Report UIUCDCS-R-92-1758, Department of Computer Science, University of Illinois at Urbana-Champaign*, June 1992.

[70] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *In Proceedings of ACM SIGCOMM*, pp. 251-262, September 1999.

[71] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. *In Proceedings of ACM SIGCOMM*, August 2004.

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

[73] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Server-based inference of Internet link Lossiness. *In proceedings of IEEE INFOCOM*, 2003.

[74] CPLEX optimization package (Last accessed October 20, 2008). http://www.ilog.com/products/cplex/

[75] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to model and internetwork. *In proceedings of IEEE INFOCOM*, 1996.

[76] N. Spring, L. Peterson, A. Bavier, and Vivek Pai. Using PlanetLab for Network Research: Myths, Realities, and Best Practices. *In Proceedings of ACM SIGOPS Operating Systems Review*, Vol. 40, No. 1, January 2006.

[77] Gianluca Iannaccona, Chen-Nee Chuah, Richard Mortier, Supartik Bhattacharyya, and Christophe Diot. Analysis of link failures in an IP backbone. *In Proceedings of the 2nd Internet Measurement Workshop*, 2002.

[78] C. Boutremans, G. Iannaccone, and C. Diot. Impact of link failures on VoIP performance. *In Proceedings of NOSSDAV*, May 2002.

[79] A. Markopoulou, G. Iannaccona, S. Bhattacharyya, C. Chuah, and C. Diot. Characterization of failures in an IP backbone. *In Proceedings of IEEE INFOCOM*, 2004.

[80] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot. An approach to alleviate link overload as observed on an IP backbone. *In IEEE INFOCOM*, 2003.

[81] M. Steinder, and A. Sethi. A survey of fault localization techniques in computer networks. *Elsevier Science of Computer Programming* 53 (2004), 165-194.

[82] A.T. Bouloutas, S. Calo, A. Finkel. Alarm correlation and fault identification in communication networks. *IEEE Transactions on Communications* 42(2-4), 1994, pp. 523-533.

[83] I. Katzela, and M. Schwartz. Schemes for fault identification in communication networks. *IEEE/ACM Transactions on Networking* 3 (6), (1995), pp. 733-764.

[84] Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C. Snoeren. IP fault localization via risk modeling. *In Proceedings of Networked Systems Design and Implementation (NSDI)*, May 2005.

[85] M. Fraiwan, and G. Manimaran. Link stress reduction in topology-aware overlay path monitoring. *Elsevier Computer Communications*, vol. 31 (10), 2008, pp. 2086-2093.

[86] M. Fraiwan, and G. Manimaran. Localization of IP Links Faults Using Overlay Measurements. *In proceedings of IEEE ICC*, 2008.

[87] K. Suh, Yang Guo, Jim Kurose, and Don Towsley. Locating network monitors: complexity, heuristics, and coverage. *In proceedings of IEEE Infocom*, 2005.

[88] Yigal Bejerano, and Rajeev Rastogi. Robust Monitoring of Link Delays and Faults in IP Networks. *In Proceedings of IEEE Infocom*, 2003.

[89] R. K. Ahuja. T. L. Magnanti, and J. B. Orlin. Network Flows: Theorey, Algorithms, and Applications. *Prentice Hall Inc. 1993.*

[90] G. R. Cantieni, Gianluca Iannaccone, Chadi Barakat, Christophe Diot, and Patrick Thiran. Reformulating the Monitor Placement Problem: Optimal Network-Wide Sampling. *In proceeding of 40th Annual Conference on Information Sciences and Systems*, 2006.

[91] S. Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. On the Placement of Internet Instrumentation. *In proceedings of IEEE Infocom*, 2003.

[92] Sridhar Srinivasan, and Ellen Zegura. RouteSeer: Topological Placement of Nodes in Service Overlays. *Technical report, Georgia Institute of Technology*, 2003.

[93] Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C. Snoeren. Detection and Localization of Network Black Holes. *In Proceedings of IEEE INFOCOM*, 2007.

[94] J. P. Lang, and J. Drake. Mesh Network Resiliency Using GMPLS. *In proceedings of IEEE*, vol. 90, No. 9, 2002.

[95] Y. Tang, E. S. Al-Shaer, and Raouf Boutaba. Active Integrated Fault Localization in Communication Networks. *In proceedings of IEEE/IFIP Integrated Management*, May, 2005.