Quality-of-Service Routing for Voice-over-IP in Service Overlay Networks

Hong Li

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

November 2009

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

2009/11/11

Abstract

Voice-over-IP (VoIP) becomes more and more popular with the development of service convergence in the next generation network. This thesis focuses on improving VoIP quality with application-layer routing in service overlay networks.

Internet end-to-end delay is one of the most important impairments on VoIP quality. We therefore analyze, model and simulate it to better understand it and thus to discover potential advantages of routing in service overlay networks.

Based on the investigation of the Internet end-to-end delay, we find that VoIP quality on a pair of diverse paths is better and more stable than that on a single path. We therefore propose a novel centralized data fusion approach to search for the best pair of diverse paths. This method jointly optimizes source routing with adaptive play-out scheduling at the receiver. It requires transmitting the delay distributions of all the overlay links for estimating the delay distributions of diverse paths. We propose to transmit only the model parameters of the link delay distributions to reduce the communication overhead. It is shown that the best pair of diverse paths can be estimated with a small error.

Nonetheless, the centralized approach is computationally expensive. We therefore propose an online diverse routing method, which uses distributed learning automata to actively probe path delays and to determine the best pair of diverse paths for VoIP based on the state of the learning automata. We have demonstrated the scalability and the optimality of the approach by simulations, and proven the optimality of the approach using Kushner's weak convergence method. VoIP quality has been shown to improve from unsatisfactory levels to satisfactory levels. In addition, we propose a method to detect and recover from link failures based on the state of the learning automata. Considerable improvement in link failure recovery time has been achieved.

In sum, this work demonstrates that the proposed centralized diverse routing approach is effective to improve VoIP quality in terms of R-factor for small overlay networks, and that the proposed distributive diverse routing approach together with the link failure detection scheme provides a scalable, effective and robust solution to VoIP routing for large overlay networks.

Sommaire

Voix sur IP (VoIP) est un service dont la popularité croît avec le développement de la convergence entre les services dans les réseaux dits de nouvelle génération. Dans cette thèse, nous nous appliquons à améliorer la qualité des services de VoIP grâce au routage au niveau la couche application en utilisant des réseaux dédiés.

Dans cette thèse, nous procédons à l'étude des délais de bout en bout du réseau Internet, qui sont le facteur impactant le plus sur la qualité de la VoIP. Nous analysons, modelons et synthétisons des traces de délais de bout en bout, afin de découvrir un potentiel intérêt relatif à leur utilisation dans le cadre du routage au niveau le la couche application utilisant des réseaux dédiés.

En nous appuyant sur l'étude des traces de délais de bout en bout, nous montrons que la qualité de la VoIP peut être améliorée et stable en utilisant un couple de routes diverses, au lieu d'une seule route. Nous donc proposon un centre de fusion de données qui utilise notre approche pour trouver le meilleur couple de routes diverses. Cette méthode optimise le routage source conjointement avec adaptation play-out au niveau du récepteur. Il faut transmettre des délais des-dites distributions de tous les liens au niveau le la couche application pour estimer des distributions de tous les couples de routes possibles. Nous proposons de transmettre uniquement les paramètres du modèle de la distribution des délais, afin de réduire des coûts de communication. Nous prouvons que cette méthode peut trouver le meilleur couple de routes à une faible erreur.

Comme la technique centralisée requiert une grande puissance de calcul, nous proposons une solution de routage divers extensible en ligne, qui utilise l'apprentissage distribué automata activement sonde des délais de bout en bout et détermine la meilleure paire de diverses voies de VoIP basé sur l'état de l'apprentissage d'automates. Nous avons démontré l'extensibilité et l'optimalité de cette approche par les simulations, et démontré l'optimalité de l'approche par l'utilisation de la méthode de convergence faible de Kushner. Nous montrons que la qualité de la VoIP est ameliorée, passant d'une qualité inacceptable à une qualité acceptable. De plus, nous proposons une méthode pour détecter les défaillances du lien et de sa récupération sur la base de paramètres de l'apprentissage d'automates, qui permettent une réduction considérable du temps de récupération à la suite de la défaillance d'un lien.

En somme, cette thèse démontre que la proposition de la diversité de routage centralisé approche est efficace pour améliorer la qualité de la VoIP en termes de R-facteur pour les petits réseaux de la couche application, et que l'apprentissage de un couple de routes diverses avec des méthodes de détection de défaillance offre un extensible, efficace et robuste solution pour services de VoIP grâce au grands réseaux de la couche application.

Acknowledgments

I would like to express my gratitude to my supervisors, Prof. Lorne Mason and Prof. Michael Rabbat, for their guidance over the course of my studies at McGill University. Through their wisdom, they have provided me with insights and feedback that have proven to be invaluable; I am greatly appreciative of their contributions. I also thank the other members of my Ph.D. committee, Prof. Mark Coates, Prof. Peter Kabal and Prof. Muthucumaru Maheswaran. Financial support for this thesis was provide by Prof. Lorne Mason and Prof. Michael Rabbat via grants from NSERC. I wish to thank them for their assistance.

I would like to recognize the graduate students in the McGill ECE department for their help and friendship. I would like to thank my colleagues Yvan Pointurier and Boris Oreshkin, who have been a sounding board for discussion. In particular, I thank Dr. Yvan Pointurier and my friend JuJing Tan for proofreading this manuscript. Further thanks to Yvan Pointurier for translating the abstract. Likewise, I thank my fellow lab members Frederic Thouin, Fariba Heidari and Alex Lam and Mohammed Ghanassi for their kind support and discussion. I would also like to thank my dear friend Dr. Huiqun Deng for encouraging and supporting me.

I would also like to thank my family for their vital support. My parents always provided me the love and encouragement to complete my education, and greatly contributed to my success at McGill. Thanks to Meizheng Jia and Huimin Zeng for their love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work investigates quality-of-service routing methods for improving Voice-over-IP quality. In this chapter, I first present the background and the big picture of the thesis work. Then I show the motivation for each part of the work. After that, my novel contributions and related publications are also presented.

1.1 Background and Big Picture of the Work

Voice-over-IP has its origins in the early days of packet switching when in 1973 the Network Voice Protocol (NVP) specification was defined in RFC 741 [5]. NVP was implemented in the ARPANET, the ancestor of today's Internet, however service quality was not satisfactory. From its early origins to the present day, packet voice quality is impaired by the statistical multiplexing inherent in packet switching, whereby random delay is introduced as packets traverse routers or packet switches. To remove this jitter in the arrival times of voice packets at the receiver, it is necessary to buffer and then read the packets out in a manner that preserves the constant bit rate voice packet stream had at its point of origin. Some of the mechanisms considered are to mitigate delay and loss impairments within the network while other mechanisms reside at the receiver. Before describing the mechanisms in detail, we briefly review the diversity of approaches employed to transport voice packets. With the appearance of the global Internet in the mid-1990s, hobbyists experimented with sending voice packets over the Internet and in 1994 the first commercial service was offered by Vocaltec Inc. [6]. Following that event many different implementations of VoIP have appeared. A general setting for Voice-over-IP is shown in Fig. 1.1, where the Internet, instead

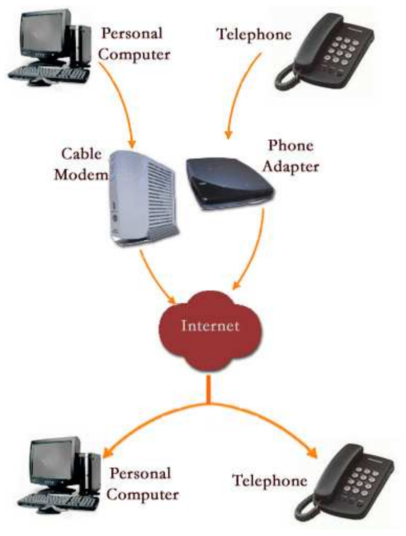


Fig. 1.1 A general setting for VoIP [1]. It shows that the Internet, instead of the traditional telephone network, carries voice calls between computers and telephones.

of the traditional telephone network, carries voice calls between computers and telephones. Many large corporations introduced VoIP in their private Intra-nets using RSVP [7] and IntServ [8] to provide good quality of service among their global branch offices. While the Quality of Service (QoS) is acceptable as a result of the bandwidth reservation inherent in the IntServ model, the approach does not scale to the global Internet. DiffServ [9] was proposed as a scalable alternative to provide differentiated service over the Internet and many Internet Service Providers (ISPs) have adopted the complementary combination of DiffServ-aware MPLS [10] in their networks. This approach does not employ signalling and bandwidth reservation associated with IntServ and thereby avoids the scalability problem at the expense of relinquishing QoS guarantees. DiffServ relies on priority queueing, and traffic engineering, including capacity provisioning and routing to provide service quality, which by its very nature relies on accurate forecasts of traffic demand and timely capacity provisioning, a very challenging task in today's environment, where new services appear overnight.

In addition to the challenge of good traffic engineering there is a more fundamental issue which impedes high quality end-to-end QoS necessary for toll quality VoIP. The current Internet is structured as a hierarchy of interconnected Autonomous Systems (ASes), where no single ISP has control over the end-to-end service quality. Indeed a given VoIP call might originate within a wireless LAN offering only a best effort service (no QoS guarantees), then transit a local ISP before reaching a tier one or backbone ISP which implements MPLS/DiffServ, before going down the hierarchy of local ISP, and perhaps an ADSL loop and then reaching a standard fixed telephone. Packet delay, loss and jitter are introduced within each segment of the end-to-end path with no one being responsible for the end-to-end quality of the call.

Apart from variations in quality resulting from statistical multiplexing, abrupt degradation in quality also occurs as a result of link and node failures on the transmission path, as well as path changes invoked by the Internet's routing protocols [11]. This is the current state of affairs and in spite of the long sought goal of constructing an Integrated QoS enabled Internet, this vision still remains to be realized.

Many traditional carriers and cable operators offer VoIP over their MPLS DiffServ-aware networks, however off-net calls can still encounter quality impairments. Apart from the large players in the market place, there have recently been a spate of phone card operators who provide VoIP service to their customers by having them dial, for example, a toll-free number which connects a circuit switched voice call to a gateway, where PCM to packet conversion takes place before the call enters the public Internet.

While the initial attraction of VoIP was inexpensive or free calling, there are now numerous other benefits associated with VoIP, since it is much easier to create new multimedia service combinations if every component service employs the IP protocol. Indeed this is a primary driver of service convergence long sought by network operators. When VoIP was in its infancy, users were prepared to tolerate poor quality as the service was effectively free, however as more and more voice traffic is being transported as VoIP, customers now demand toll quality service.

As the current Internet cannot guarantee Quality-of-Service for VoIP, hence, an alternative approach, service overlay network has been proposed to provide end-to-end quality-of-service as shown in Fig. 1.2.

Service overlay networks are logical networks formed by connecting nodes/gateways via logical links/tunnels through the underlying network, as shown in Fig. 1.2. The key to improve VoIP quality via service overlay networks is to choose the paths that provide the best VoIP quality from all the paths that are provided by the virtual network. The goal of this work is to invent new routing methods to improve VoIP quality.

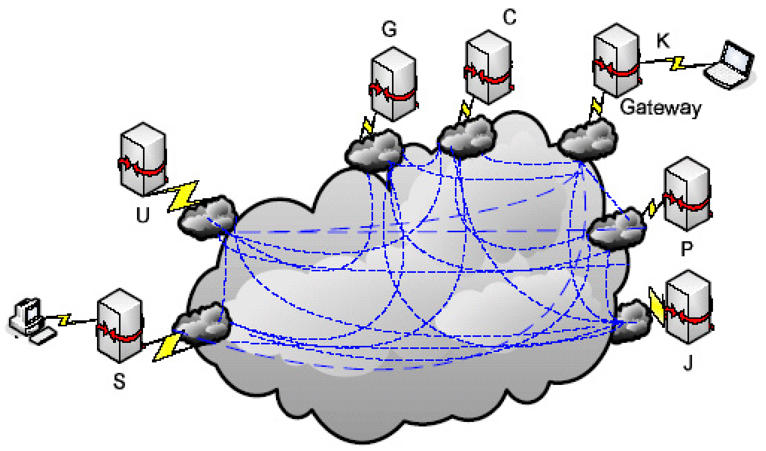


Fig. 1.2 Service Overlay Network. The overlay nodes/gateways are interconnected via virtual connections as shown by the dashed lines.

The challenge for selecting the best paths for VoIP is to determine which paths provide the best R-factor. A number of previous work [12–17] has shown that diverse routing, i.e. sending redundant voice packets to two diverse paths, can improve VoIP quality and improve robustness against link failures. However, the methods for finding the best diverse paths have been heuristics and no optimal solution is available. Moreover, the previous diverse routing approaches do not scale with network sizes and are thus inapplicable to large overlay networks.

This work proposes a new method to select the best diverse paths for VoIP and we evaluate this method based on modeling and simulating end-to-end delays of each path. We also propose a scalable distributed routing approach to adaptively learn the best pair of diverse paths. In addition, we propose a distributive scheme to detect and recover from link failures.

1.2 Motivation

In this thesis, I present my work on the fundamental study of real end-to-end delay measurements, the design and simulation of a centralized VoIP QoS routing approach for small-scale service overlay networks, the proposal and experiment of a distributed routing approach for choosing the single best paths and a pair of diverse paths in large-scale service overlay networks. In the following, I am going to first present the motivation for these works.

1.2.1 End-to-end Delay Analysis, Simulation and Sampling

End-to-end delay is one of the most important impairments for VoIP quality. It is also an important measure of network performance, useful for network diagnosis, application performance optimization, and network design. However, it is expensive to measure end-to-end delays for all the source-destination pairs in a rapidly growing large network. Nowadays, there are over 53247 Autonomous Systems (ASes) [18] and the number keeps increasing each year. To measure end-to-end delays for such a large network and to store all the measurements require a large amount of monitoring facility and storage space. Hence, it is very important to be able to reduce the amount of measurements and storage, while still maintaining a good understanding of the end-to-end performance.

To reduce the amount of monitoring and thus to reduce the amount of storage required, many work on compressed monitoring and network tomography [19–22] has been investigated, which infer network performance based on a reduced amount of measurement. Network performance characteristics inferred with these techniques are mostly statistics of the network performance and are useful for network diagnosis.

However, for the purpose of real-time application performance optimization, which usually involves measuring end-to-end delay for each application packet, the previous network tomography or compressed monitoring techniques are not sufficient to infer end-to-end delay time series. Hence, in this case, an approach that can synthesize or infer end-to-end delay time series is required.

Suppose we only store the mean value of an end-to-end delay time series and that the other statistics of the end-to-end delay time series are missing, is it possible to reconstruct or synthesize the end-to-end delay time series? If this can be done, it means only the mean of the end-to-end delays needs to be stored, which will significantly reduce the amount of storage for end-to-end delay measurements. Another approach is to find an optimal sampling rate for end-to-end delays, such that the sampling rate guarantees a maximum sampling accuracy at a minimum sampling cost. This approach also reduces network monitoring and storage, while still allowing reconstruction of the end-to-end delay time series. Hence, the purpose of understanding, simulating and sampling end-to-end delay time series forms the motivation of this work on end-to-end delay analysis, simulation and sampling. Chapter 3 will present the details of the investigations on end-to-end delay analysis, simulation and sampling.

1.2.2 R-factor based Diverse Routing for VoIP: a Centralized Solution

With the analysis of real end-to-end delay time series in the fundamental study just mentioned, we observe triangle inequality violations [23], i.e. the delay on an alternative path through an intermediate hop not on the shortest-path route can be shorter than that on a direct path (a direct path between a source-destination pair is the path determined by the underlying network). Hence, there are benefits to route voice packets via the alternative paths instead of the direct paths. Selecting the best alternative paths requires measuring overlay link performances and computing the paths that provide the best VoIP quality. A standard measure for VoIP quality, R-factor is then chosen as the routing metric.

In previous work [12–15], diverse routing, i.e. sending redundant voice packets to two diverse paths, has suggested successful quality improvement for VoIP in overlay networks. However, the methods for finding the best diverse paths have been heuristics and no optimal solution is available. Hence, we are motivated to choose the best pair of paths to maximize R-factor for VoIP calls, which is very challenging because R-factor is related to both the routing path and the packet play-out scheduling scheme [16] (i.e. the scheme to determine when a packet is decoded and played out at the receiver). In Chapter 4 we tackle this problem with a novel centralized method to estimate the diverse paths with the best R-factor, which shows improved VoIP quality.

1.2.3 Learning Minimum Delay Paths: a Decentralized Solution

In the centralized approach above, estimating R-factor for all the possible pairs of paths is computationally expensive, which limits its applicability to large overlay networks. Hence, we are motivated to design a distributed routing approach that is scalable with network sizes, to provide the best quality for VoIP calls in service overlay networks. In this distributed scheme, no routing update message or overlay link performance characteristics are communicated among the overlay nodes. Routing decisions are made locally at each overlay node.

As VoIP quality is sensitive to end-to-end delay, we choose paths to minimize end-to-end delays. As previous minimum delay routing methods [24–28] are usually very complex, our approach is also motivated to have low computational complexity. In sum, the desirable features of our approach include low computational complexity, distributed routing decision making, scalability with network sizes and adaptivity to dynamic network environments.

In Chapter 5, we present this completely distributed approach. It is able to discover the minimum delay paths in around 5 seconds if the probing interval is 5 ms, and the method scales well with network sizes.

1.2.4 Online Distributed Diverse Routing for VoIP

Section 1.2.2 has mentioned a centralized approach to determine the best pair of diverse paths for VoIP in small-scale service overlay networks. For large-scale service overlay network, since the distributed approach mentioned in Section 1.2.3 is able to learn the minimum delay paths and scalable to network sizes, we extend it to learn the best pair of diverse paths. However, this online distributed diverse routing introduces new challenges, for example, how to guarantee link disjointness between the diverse paths distributively and adaptively to network dynamics. No distributive approach has been seen dealing with such challenges. We tackle this issue with a max-rule based approach, detailed in Chapter 6, to choose the best pair of diverse paths that have minimum delays and maximum link disjointness. Improved VoIP quality and robustness against link failures on the selected diverse paths are also shown in Chapter 6.

1.2.5 Link Failure Detection in Service Overlay Networks

In the current best-effort Internet, all the packets on a link are dropped if that link fails. Overlay networks are known to be able to provide better alternate paths to avoid link failures. For the distributed minimum delay path learning and diverse path learning methods mentioned in Sections 1.2.3 and 1.2.4, we are also motivated to propose a distributed link failure detection algorithm to provide robustness against link failures. For VoIP sessions, it is an important objective to detect and recover from a link failure in 2 seconds to avoid sessions being dropped. If we recover in less than 2 seconds the VoIP user will only experience a click but the session continues [29]. It will be shown that this objective can be achieved with the proposed link-failure detection and recovery approach. The detail of this mechanism is also presented in Chapter 6.

1.3 Novel Contributions

1.3.1 Analyzing, modeling, simulating and sampling end-to-end delays

- Detailed analysis of end-to-end delay traces.
- Suggested ways to synthesize end-to-end delay traces and ways to design the capacity of a network to simulate realistic delays.
- Proposed a fair sampling solution to resolve the trade-off between the sampling accuracy and the sampling cost.

1.3.2 Improving R-factor with Diverse Routing: a Centralized Approach

- Suggested jointly optimizing routing with adaptive play-out scheduling to choose a pair of diverse paths that maximizes R-factor for VoIP.
- Proposed to use a centralized data fusion center to estimate R-factor for all the possible pairs of paths based on the summarized performance characteristics of all the overlay links.

1.3.3 Learning Minimum Delay Paths: a Decentralized Approach

- Proposed to use distributive learning automata and cross-correlation learning algorithm to learn minimum delay paths distributively. Four variations of this approach are proposed.
- Simulation results showing the scalability and optimality of the proposed approach.
- Proved the cross-correlation learning algorithm converges to a globally stable user equilibrium solution by Kushner's weak convergence method [30] and by finding a Lyapunov function for end-to-end delays.
- Analyzed the probing overhead for the proposed learning automata based approaches.

1.3.4 Online Distributed Diverse Routing for VoIP

- Proposed to learn a pair of diverse paths for VoIP with the distributive learning automata based approach above.
- Proposed a max-rule based approach to determine the primary-optimal paths and the secondary-optimal paths for VoIP based on the state of the distributive learning automata.
- Suggested ways to choose the secondary-optimal paths to be disjoint to the primary-optimal paths.

1.3.5 Link Failure Detection in Service Overlay Networks

- Proposed to detect and recover from link failures based on the state of the distributive learning automata.

- Simulation results showed considerable reduction in link failure recovery time comparing to that when the link failure detection mechanism was not applied.

1.4 Related Publications

1.4.1 Journal Paper

- Hong Li, Lorne Mason and Michael Rabbat, "Learning Optimal Diverse Paths for Voice-Over-IP in Service Overlay Networks: a Distributed, Scalable and Robust Solution," to appear in IEEE Trans. Network and Service Management, 2009.

1.4.2 Report of Invention

- Hong Li, Lorne Mason and Michael Rabbat, "Online Distributed Diverse Routing for Voice-over-IP in Service Overlay Networks.", 2009.

1.4.3 Conferences

- Hong Li, Lorne Mason and Michael Rabbat, “Learning minimum delay paths for VoIP in Service Overlay Networks,” IEEE NCA’08, Jul. 10 - Jul. 12 2008, Cambridge, MA, USA.

- Hong Li and Lorne Mason, “Optimal multipath routing with adaptive playback scheduling for VoIP in Service Overlay Networks,” IEEE Sarnoff’08, Apr. 28-Apr. 30, 2008, Princeton, NJ, USA.

- Hong Li and Lorne Mason, “Synthesis of network delays for voice packets in Service Overlay Networks,” IEEE/ACM Qshine’07, Aug. 14-Aug. 17, 2007, Vancouver, Canada.

- Hong Li and Lorne Mason, “Estimation and Simulation of Network Delay Traces for VoIP in Service Overlay Network,” IEEE ISSSE’07, Jul. 31-Aug. 2, 2007, Montreal, Canada.

1.5 Thesis Organization

This chapter is the introduction of the whole thesis work. Chapter 2 reviews related work. Chapter 3 presents our contribution on end-to-end delay analysis, simulation and sampling. The work on improving R-factor with a centralized diverse routing approach is presented in Chapter 4. Chapter 5 shows our contribution on learning minimum delay paths with a distributed approach. The work on online distributed diverse routing and link failure detection are presented in Chapter 6. Chapter 7 concludes the whole work.

Chapter 2

VoIP and its Quality: Background and Related Work

2.1 Introduction

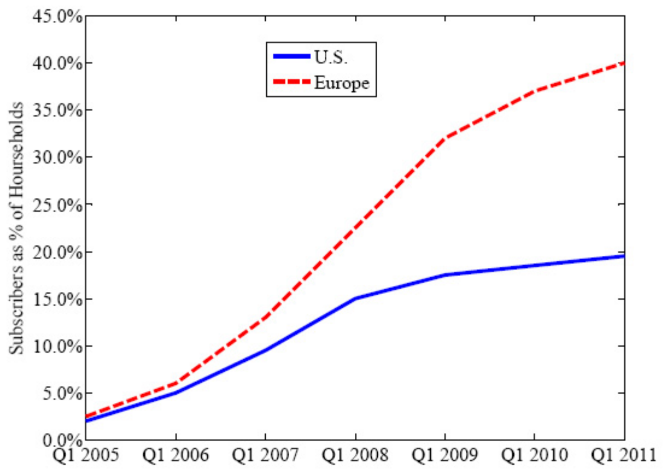


Fig. 2.1 VoIP penetration in U.S. and Europe [2]. It shows the increasing trend of VoIP subscribers as percent of households in U.S. and Europe from 2005 to 2011.

As mentioned in Section 1.1, due to low cost and service flexibility of VoIP, the number of VoIP subscribers has been increasing rapidly over the past few years, and it is expected to continue increasing in the future, as illustrated in Fig. 2.1. Then the immediate problem following the increase of VoIP subscribers is how to provide toll-quality voice calls.

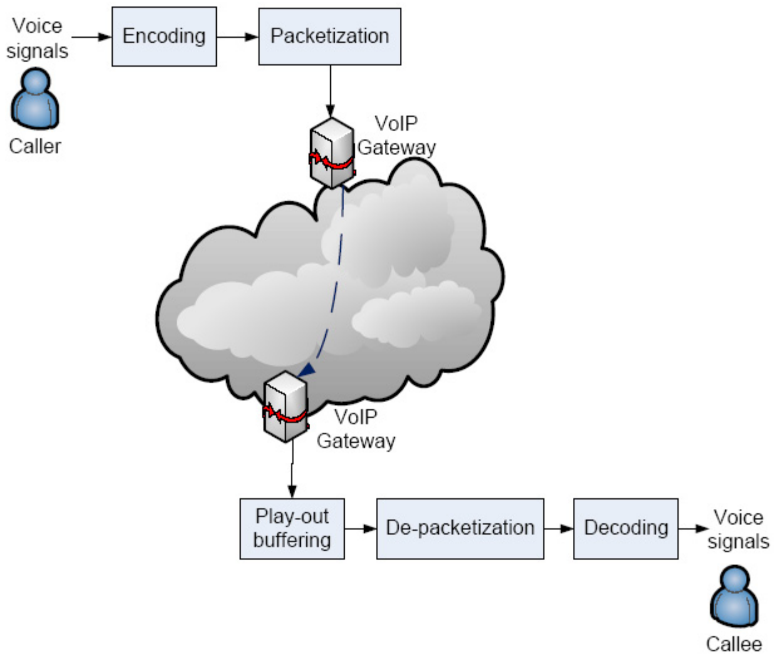


Fig. 2.2 A simple VoIP system. Voice signals are encoded and packetized before being sent to the Internet via a VoIP gateway. The packetized voice goes through a path selected by the Internet routing protocol to reach its destination/receiver. At its receiver, the voice packet is buffered at the play-out/de-jitter buffer until the time scheduled to play it out (i.e. de-packetization and decoding).

Fig. 2.2 illustrates a simple VoIP system. For a toll-quality VoIP call, ITU G.114 recommends the mouth-to-ear delay, which includes the delays for voice encoding and packetization, the network delay and the play-out delay at the receiver, to be less than 150 ms. The mouth-to-ear loss, which includes the network loss and the play-out loss at the receiver, is recommended to be less than 1%, The delay jitter is expected to be as small as possible. However, the current Internet is unable to provide such Quality-of-service for Voice-over-IP which leads to our investigation on Quality-of-service routing in Chapters 4, 5 and 6. Before describing our work on VoIP QoS routing in the following chapters, we first review the background on VoIP and present related work on VoIP QoS routing in this chapter.

2.1.1 Chapter structure

The organization of the chapter is as follows. Section 2.2 introduces VoIP quality measure. Section 2.3 reviews the previous work on Quality-of-service routing for VoIP. Section 2.4 presents integration of VoIP in service overlay networks with exising standard signalling protocol. At the end, Section 2.5 summarizes this chapter.

2.2 Voice-over-IP (VoIP)

2.2.1 R-factor: VoIP Quality Measure

R-factor is a standard measure for VoIP quality defined in ITU-T G.107 E-model [3]. This model maps various network impairments to speech quality and provides a scalar quality rating value R, ranging from 0 to 100. It combines all the transmission parameters related to a connection under consideration. By definition, $R = R_o - I_s - I_d - I_{ef} + A$ [3]. R_o refers to the basic signal-to-noise ratio. It is a function of the circuit noise, send-loudness rating, room noise and receive-loudness rating. I_s refers to the impairments that occur more or less simultaneously with voice transmission, such as too low loudness rating, non-optimum side-tone and quantizing distortion. I_d represents the impairments caused by voice signal delay, including the impairments by talker echo, listener echo, and unacceptably long absolute delay. I_{ef} is the equipment impairment due to low bit rate codecs and packet losses. ITU has defined a few codecs for VoIP, as given in Table 2.1. A is the advantage factor, which is 0 in the conventional environment. For a multi-hop satellite connection A can be as high as

20. If all the parameters are set to the default values given in Table 2 of ITU-T G.107 [3], the rating R is 93.2.

Codec	Bit Rate (Kbps)	Sample Size (Bytes)	Sample Interval (ms)	Voice Payload Size (ms)	Packet Per Second
G.711	64	80	10	20	50
G.729	8	10	10	20	50
G.723.1	6.3	24	30	30	34
G.723.1	5.3	20	30	30	34
G.726	32	20	5	20	50
G.726	24	15	5	20	50
G.728	16	10	5	30	34

Table 2.1 Codec Information [4]

Denote the mouth-to-ear delay as d and the mouth-to-ear loss as ℓ, then R-factor can be written as a function of d and ℓ by setting all the other impairments to their default values [3], as follows.

$$R = 93.2 - I_d(d) - I_{ef}(\ell), \tag{2.1}$$

where $I_d(d)$ is the impairment caused by mouth-to-ear delay d an $I_{ef}(\ell)$ is the impairment caused by mouth-to-ear loss ℓ. $I_d(d)$ and $I_{ef}(\ell)$ can be computed as in ITU-T G.107 [3] or as follows [31]:

$$\begin{aligned} I_d(d) &= 0.024d + 0.11(d-177.3)\mathbb{I}(d-177.3), & (2.2)\\ I_{ef}(\ell) &= \gamma_1 + \gamma_2 \ln(1+\gamma_3\,\ell), & (2.3)\end{aligned}$$

where $\mathbb{I}(x) = 0$ when $x < 0$ and $\mathbb{I}(x) = 1$ when $x \geq 0$. The parameters $\gamma_1, \gamma_2, \gamma_3$ vary among codec and loss types as given in Table 2.2.

Codec	γ_1	γ_2	γ_3
G.711	0	30	15
G.729	11	40	10

Table 2.2 Parameter values for computing $I_{ef}(\ell)$

A *subjective* rating system for received media (voice, audio or video) after compression and/or transmission is Mean Opinion Score (MOS). This score is given by averaging the scores provided by all the individual testers. R-factor can be related to the subjective rating MOS by the following expression: for $R < 0$, $MOS = 1$; for $R > 100$, $MOS = 4.5$; for $0 < R < 100$, $MOS = 1 + 0.035R + 7 \cdot 10^{-6} \cdot R \cdot (R - 60)(100 - R)$. A commonly recognized quality of voice rating with the value of R-factor and MOS is shown in Fig. 2.3. It can be seen that voice quality is rated high or best when R-factor is above 80. Thus, the target R-factor for VoIP quality of service provision can be set to be above 80.

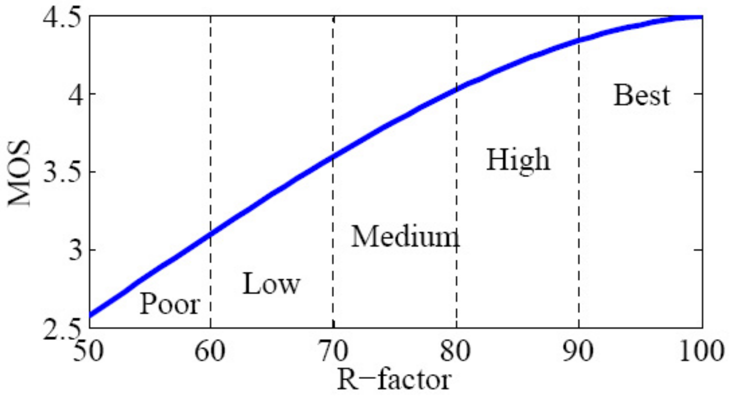


Fig. 2.3 Quality of voice rating V.S. R-factor and MOS values [3]

2.2.2 VoIP Quality Degradation with Mouth-to-ear Delay and Losses

R-factor as a function of mouth-to-ear delay d and loss ℓ [3] for the standard voice codec G.711 is illustrated in Fig. 2.4. The reader should note that it is imperative to maintain a low mouth-to-ear delay d and loss ℓ to achieve high VoIP quality. In the following, we show how the mouth-to-ear delay d and loss ℓ are computed.

The mouth-to-ear delay d includes encoding delay d_{enc}, network delay d_{net} and play-out delay d_{play}, i.e. $d = d_{enc} + d_{net} + d_{play}$. The encoding delay d_{enc} is usually 20 ms for G.711, G.729 and G.726 codec, 30 ms for G.723 and G.728 codec. The network delay d_{net} equals the end-to-end delay of the selected voice path. The characteristics of the network delay d_{net} is analyzed in detail in Chapter 3. The play-out delay d_{play} depends on the play-out scheduling algorithm at the receiver, which is detailed in Section 2.2.3.

The mouth-to-ear loss ℓ includes end-to-end network loss l_{net} and play-out loss l_{play}, i.e. $\ell = l_{net} + (1 - l_{net}) * l_{play}$. The end-to-end loss l_{net} is the end-to-end loss rate on the selected

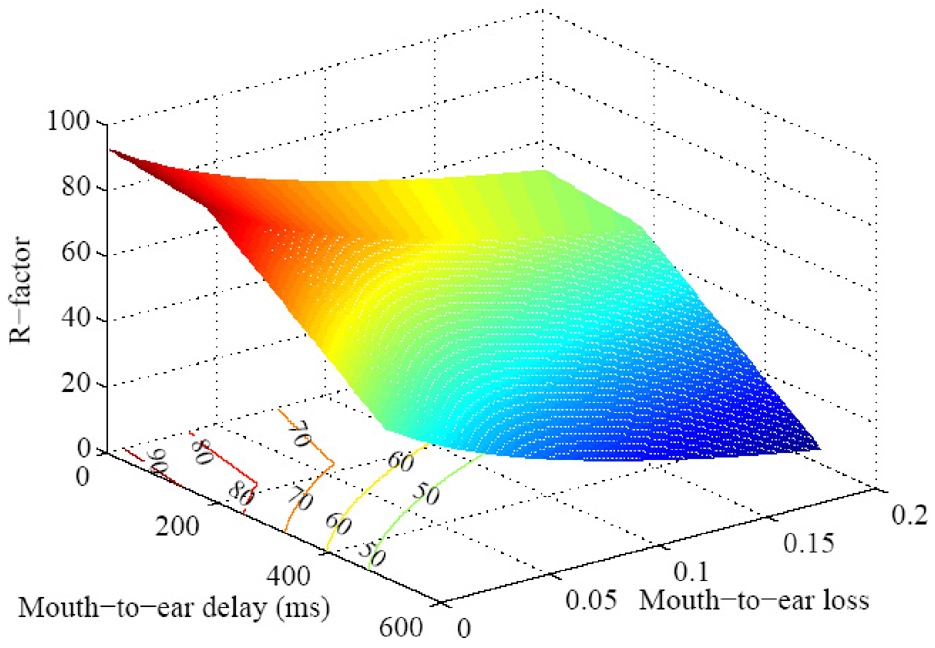


Fig. 2.4 R-factor as a function of mouth-to-ear delay and loss for the codec G.711 [3]. The maximum R-factor for this codec is 93.2. We adopted G.711 codec in our simulations at Chapter 6 because this codec is commonly used in practice.

voice path. The play-out loss l_{play} is determined by the play-out scheduling scheme at the voice call receiver, as detailed in Section 2.2.3.

2.2.3 Adaptive Play-out vs. Fixed Play-out Scheduling

At the VoIP call receiver, the decoder expects a smooth voice packet input for a smooth voice signal output. However, due to network delay jitters in the Internet, voice packets arrive at the receiver at a variable rate. Thus, a play-out buffer is applied at the receiver to temporarily store voice packets and then play them out at a scheduled time. In this way, the decoder can play the voice packets out at a relatively smooth rate. However, this also introduces an additional delay and loss for voice packets, i.e. the play-out delay d_{play} and loss l_{play}. The play-out delay d_{play} is the delay from the time a voice packet arrives at the receiver till the time it is played out. The play-out loss l_{play} refers to the probability that a voice packet arrives at the receiver later than the scheduled time for it to be played out. The values of d_{play} and l_{play} depend on the play-out scheduling algorithm. In general, there are two types of play-out scheduling algorithms, fixed play-out scheduling and adaptive play-out scheduling [16, 32] as follows.

1. Fixed play-out scheduling
   Fixed play-out scheduling means that the play-out buffer delay is set to be fixed during a call or during a talk-spurt. Suppose the fixed play-out delay is 120 ms and the first received packet of a talk spurt has a network delay of d_1 ms, then the total delay for all the packets in this talk spurt is $d_1 + 120$ ms, as illustrated in Fig. 2.5, and all the packets in this talk spurt with a network delay over $d_1 + 120$ ms are lost. Denote the network delay cumulative distribution as $F(x)$, where unit of x is ms, then the play-out loss probability is $l_{play} = P(d_{net} > d_1 + 120) = 1 - F(d_1 + 120)$. Such fixed play-out delay setting can be computed for a whole call, (i.e. d_1 is the network delay of the first packet of a call), or calculated for each talk spurt, (i.e. d_1 is the network delay of the first packet of each talk spurt).

2. Adaptive play-out scheduling
   In adaptive play-out scheduling, the play-out delay for the first packet of each talk spurt can vary with time. An example of the play-out delay in [32] is $\hat{d}_i + \alpha \hat{v}_i - n_i$ if packet i is the first packet of a talk-spurt. n_i is the network delay of packet i. α is a

scalar parameter. $\hat{d}_i$ and $\hat{v}_i$ are the estimates of the mean and standard deviation of the end-to-end delay for this talk-spurt, which are updated whenever a packet arrives. For the other packets in this talk-spurt, the play-out delay is $\hat{d}_i + \alpha\hat{v}_i - n_j$, where n_j is the network delay of packet j. In this method [32], the sum of the network delay and the play-out delay for packets in a talk spurt is $\hat{d}_i + \alpha\hat{v}_i$. The play-out loss for packets in this talk-spurt is therefore given by $Pr\{n_j > \hat{d}_i + \alpha\hat{v}_i\}$. On one hand, if this estimate $\hat{d}_i + \alpha\hat{v}_i$ is too large, there is zero play-out loss but long play-out delays; on the other hand, if the estimate is too small, there is a large play-out loss probability but small play-out delays. Hence, the play-out delay and loss depend on the "goodness" of the estimate $\hat{d}_i + \alpha\hat{v}_i$. Therefore, the tradeoff between the play-out delay and play-out loss has to be tuned by setting α optimally. A more elegant way to determine the play-out delay for packets in a talk spurt is to use the distribution of the network delays of the previously received packets. Let the network delay distribution be $F(x)$. The 99th percentile delay value $F^{-1}(99\%)$ indicates only 1% packets have a network delay over $F^{-1}(99\%)$. Therefore, given the delay distribution $F(x)$ and a certain play-out loss probability l_{play}, one can compute the network and play-out delay for a talk spurt as $d_{net} + d_{play} = F^{-1}(1 - l_{play})$, i.e. the adaptive play-out delay is $d_{play} = F^{-1}(1 - l_{play}) - d_{net}$. The play-out loss parameter l_{play} can actually be optimized so that best R-factor can be obtained for the adaptive play-out scheme [16].

In a dynamic network environment, especially when network delays vary a lot, the play-out delay d_{play} should not be set as a fixed value, otherwise, either a packet is delayed very long before being played out (when d_{play} is set too large), or all packets that arrive later than the fixed deadline will be discarded (when d_{play} is set too small) which can lead to a large play-out loss. Fig. 2.5 illustrates two play-out scheduling schemes for three consecutive talk spurts. It is evident that the network delay for talk spurt 2 is longer than that for talk spurts 1 and 3. If fixed play-out scheduling is applied, we can see that either a packet is delayed considerably before being played out (for the case play-out delay equals $D1_{play}$), or all packets that arrive later than the scheduled play-out time are discarded (for the case play-out delay equals $D2_{play}$). However, if adaptive play-out scheduling is applied, the play-out delay of each packet is adapted to the latest network delay statistics to keep both the play-out delay and loss low.

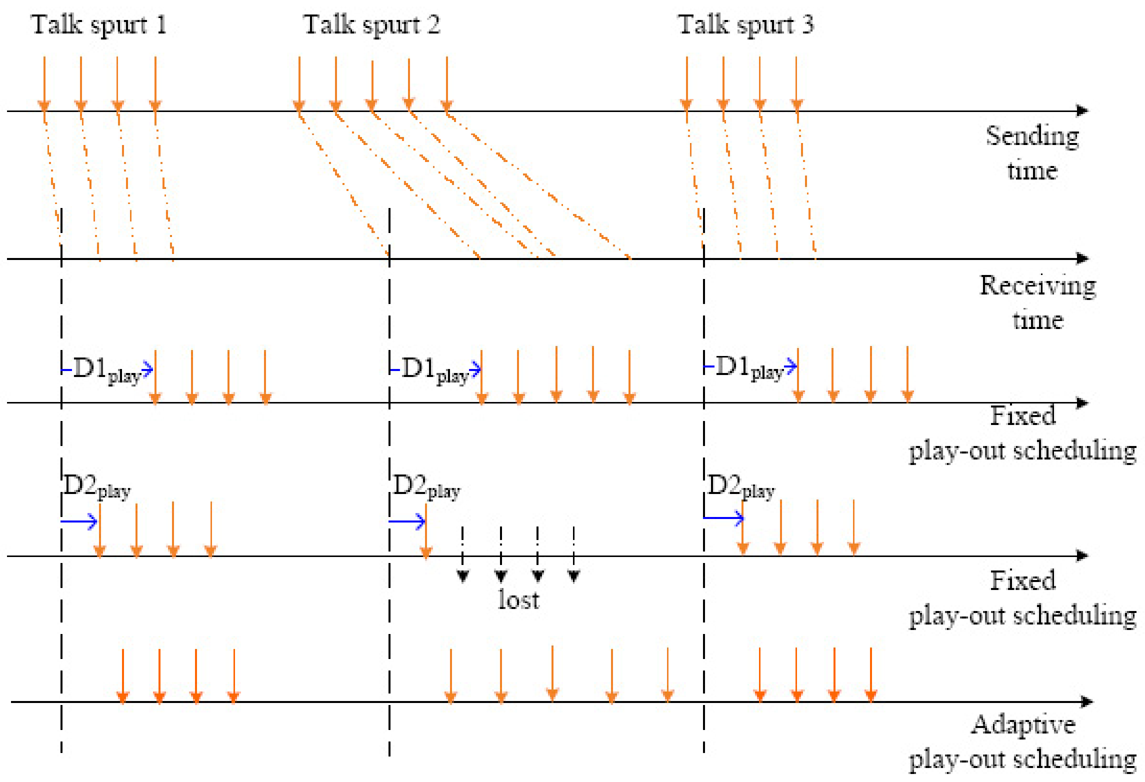


Fig. 2.5 Fixed play-out scheduling VS. Adaptive play-out scheduling. This figure shows the sending time, receiving time and the play-out time for three talk spurts. With fixed play-out scheduling, if the play-out delay is set to be as large as $D1_{play}$, all packets arrive before their scheduled play-out time, so there is no play-out loss, but the play-out delay for the packets in talk spurts 1 and 3 is long. However, if the play-out delay is set to be $D2_{play}$, 4 packets of the talk spurt 2 are dropped, which results in a play-out loss of 80%. In contrast, if adaptive play-out scheduling is employed, the play-out time of each packet is adapted based on the order statistics of the previously received packets. Hence, it is possible that both the play-out delay and play-out loss are minimal.

Previous work has also shown that adaptive play-out scheduling performs better than fixed play-out scheduling [16, 32], and for this reason we assume adaptive play-out scheduling is applied at the receiver when we propose to jointly optimize routing and play-out scheduling scheme in Chapter 4.

2.3 QoS Routing in Service Overlay Networks

In the current best-effort Internet, shortest hop routing is widely deployed by Internet Service Providers (ISPs) for intra-AS routing [33]. However, the shortest hop paths may not provide the best quality for VoIP. A link on a shortest hop path can be heavily loaded during peak hours and experiences large queuing delays and jitters[1], which are detrimental to VoIP quality. For inter-AS routing, it is also known that the policy-based BGP routing [34] is unable to provide best quality routes [35–39] because routing decisions are not made based on path performance. It has been reported that the Internet loss can be as large as 13% or even more when equipment failure happens [40]. Table 2.3 summarizes the reported Internet outage and failure characteristics. As can be seen, the link failure duration and network outage duration/down time can range from several minutes to many hours [36, 38, 41–44], and it usually takes tens of minutes or longer for the inter-domain routers to converge to a new consistent view of the network topology after a fault [39].

Link/node outage duration	very long, sometimes follow Pareto distribution [41], 5% last more than 2 hours, can last more than 20 hours [36]
Link failure duration	10% greater than 20 minutes, 40% between 1 to 20 minutes, 50% below 1 minute [42]
Classification of failures	20% due to planned maintenance activities. Of the unplanned failures, 70% affect a single link at a time [43].

Table 2.3 Internet outage and link failure characteristics

Therefore, considering these poor Internet delay and loss performances, we can see that it is hard for the current best-effort Internet to provide QoS for VoIP. Hence, it is important to develop QoS routing schemes to select high quality paths for VoIP and develop good link

[1]A detailed analysis of Internet delay characteristics is presented in Chapter 3.

failure detection and adaptive routing schemes to avoid those paths that encounter outages and link failures.

As network level QoS remains a goal of network engineers, another approach, referred to as Service Overlay Networks (SONs), has been employed as an alternative to provide QoS routing for VoIP [12–14, 45–48]. As illustrated in Fig. 1.2, a Service Overlay Network (SON) consists of gateways that are interconnected via logical links over the underlying network.

2.3.1 QoS Routing in Service Overlay Networks

Internet measurement research [23] has shown triangular inequality violation in the Internet, i.e. the delay on an alternative path through an intermediate hop not on the shortest-path route can be shorter than that on a direct path. As a consequence, there may be advantage to search for alternative paths in service overlay networks to leverage VoIP quality (this is verified in our preliminary research on QoS routing in Chapter 4).

In general, QoS routing for VoIP in service overlay networks includes the following two types of methods: single best path routing and diverse routing. Single best path routing is to choose only one path that optimizes a quality metric for VoIP. Diverse routing refers to improve VoIP quality by sending voice packets through more than one paths. In the following, I will present related work on these two routing methods in Section 2.3.2 and Section 2.3.3, respectively.

2.3.2 Single Best Path Routing

In this part, we discuss previous work that studied how a single best path can be determined for overlay routing to improve quality of service, which include a brute-force search method [48, 49], a modified Dijkstra's algorithm [13, 14] and a modified distance-vector algorithm [12].

Brute-force Search Method

Resilient overlay network (RON) [48, 49] is a pioneering work that adopts application layer routing to improve reliability for distributed Internet applications. It uses brute-force search method to search for latency, loss or throughput optimized paths. Andersen et.al. [49] have shown that path loss and latency can be considerably reduced by using a relay node.

The optimal path selection of RON is based on measuring and estimating overlay link performances. The size of a RON is usually small. It allows the overlay network to perform "aggressive" monitoring and path computation (at the cost of scalability). In a RON, the health of an overlay link is monitored and inferred by measuring packet loss rate, path latency or available bandwidth. By periodically or randomly sampling link performance with probes, an estimate of the link latency, loss and throughput can be computed, and then broadcast to all the overlay nodes. At the end, the best paths are selected based on brute-force search with these link quality estimates, which is not very costly in this case because the alternate routes under consideration are only one-hop. The selected route with the maximal routing metric estimate is then broadcast to other RON nodes every **ROUTING_INTERVAL**. To determine if a link is down, probes are sent per **PROBE_INTERVAL**. A probe is lost if it does not return in **PROBE_TIMEOUT**.

Such measurement and estimation based overlay routing method in RON is also seen in many other overlay networks [12–14]. Therefore, it is of interest to have a better understanding of the measurement and routing cost of such methods. For an N-node RON, the probing and routing traffic grows with $O(N^2)$ [48]. For a 10-node RON with **ROUTING_INTERVAL** = **PROBE_INTERVAL** = 5 ms, the aggregate probing and routing traffic at each node is 0.65 MBps. The probing and routing overhead and the time required to recover from link outages are both related to the probing frequency and the routing update frequency. For a RON with **ROUTING_INTERVAL** = 5 ms and **PROBE_INTERVAL** = 5 ms, the link failure detection time is a function of the timer setting, i.e. **PROBE_TIMEOUT**. When **PROBE_TIMEOUT** is 150 ms, 500 ms, 1 sec and 10 sec, the link failure detection time will be 606 ms, 2.005 sec, 4.005 sec and 40.005 sec, respectively. Also note that a reasonable **PROBE_TIMEOUT** cannot be too small, e.g. to be less than 150 ms. It will be shown in Section 5.5 that under the same probing rate, the communication overhead and the link failure detection time of our methods presented in Sections 5.5 and 6.5 are less than those of RON.

The limitation of RON is that it only considers one-hop alternate paths and the routing metrics of RON are targeted for general applications, which limit its capability to improve VoIP quality. In the following, we will discuss routing methods that are specifically designed for VoIP application and are not limited to one-hop paths, which include a modified Dijkstra's method and a modified distance-vector method.

Modified Dijkstra's Algorithm

Paper [13] proposes to use modified Dijkstra's algorithm to choose the best path with the maximum packet delivery ratio. Packet delivery ratio refers to the percent of packet arrivals before the play-out deadline. Maximizing packet delivery ratio is equivalent to minimizing the play-out loss. By assuming a fixed deadline for playing-out each voice packet, packet delivery ratio for partial paths can be computed based on the combination of link delay distributions. Dijkstra's algorithm is then applied to select the best path with the maximum packet delivery ratio.

The limitation of this method is that the quality of the selected path is highly related to the setting of the fixed play-out deadline and it only minimizes the play-out loss, which does not necessarily maximize R-factor for VoIP.

Modified Distance Vector Algorithm

The routing algorithm in OverPhone [12] system is a modified distance-vector algorithm. In this algorithm, link quality is measured and broadcast for computing the best paths. 1000 Probes, which are 132-byte data packets, are sent at 15 ms intervals to emulate VoIP talk-spurts. Link quality is estimated based on the delays and losses of the 1000 probing packets. For an overlay link a, the link quality of interest includes link delay d^a, link loss l^a, play-out loss j^a and conditional loss probability c^a. Note that d^a represents the delay d on link a instead of d to the power a, and similarly for l^a, j^a and c^a. Given these measured link quality, multi-hop path quality can be estimated. For a concatenated path with link a and link b, the network delay, network loss, play-out loss, and conditional loss probability of this path are estimated as $d^{ab} = d^a + d^b$, $l^{ab} = 1 - ((1 - l^a)(1 - l^b))$, $j^{ab} \approx j^a + j^b$, and $c^{ab} = \frac{(l^a \times c^a + l^b \times c^b)}{l^{ab}}$, respectively. Note that the authors use $j^a + j^b$ to approximate j^{ab} due to the complexity of computing j^{ab}. It is claimed that such approximation gives close or over estimation when the play-out loss rate of a link is greater than 1%. Then a modified Distance-Vector algorithm is used to select the best path with the highest R-factor, which is computed from the estimated network delay, network loss, play-out loss and conditional loss probability.

The limitation of this work is that they assume fixed play-out scheduling at the receiver, and there exists a better estimation of the play-out loss j^{ab} when the delay distributions of link a and b and the network delay of the first packet of each talk spurt are known.

Therefore, the selected best R-factor path is in fact sub-optimal and the quality of the selected path is related to the setting of the fixed play-out deadline.

Tracking the Best Path for VoIP

Paper [46] proposes an optimal path switching scheme based on path quality estimation. UDP probes are sent regularly to measure the network performance of each available path. R-factor is then estimated based on the probing delays and losses. It is expected that the optimal R-factor path varies with time in a dynamic network environment. This paper proposes an adaptive path switching algorithm that determines when to switch to a new path and which path the application should switch to.

For a predefined set of time scales, the proposed solution gives an optimal time scale for path switching, and chooses the path that gives sufficient gain over the current optimal path. It works as follows. (1) Predefine a set of time scales $\{t_n\}$, e.g $\{5, 100, 200, 500\}$ ms, each time scale t_n is a time unit for path switching. (2) For each time scale t_n, a baseline is defined as the maximum R-factor computed based on the measurements in the past T time slots (each slot is of size t_n). (3) In the time slot k at time scale t_n, evaluate all the path qualities, and compute the gain of switching to a new optimal path versus the quality of the baseline, compute the gain $\tilde{\rho}_k(t_n)$ based on the gain of the predicted path switching. (4) Compute the average gain $\tilde{\rho}(t_n)$ for the last T time slots for the time scale t_n. (5) The best time scale is $t = \arg\max \tilde{\rho}(t_n)$. (6) For the best time scale t, switch to another path if that path is sufficiently better than the current path in terms of R-factor.

In sum, the paper suggests the potential benefit of path switching for exploiting the Internet path diversity. It switches to a better path at an optimal time scale among a predefined set of time scales. The results it obtains can be sub-optimal because of the quantization of path switching time scales and a path only switches when the gain is sufficiently large. In addition, the method did not show how the candidate paths are determined. The applicability of the method relies on the accuracy of path quality prediction[2], which can be error-prone in a dynamic network environment.

As this method is also an adaptive routing method and track the optimal as our routing methods in Chapters 6 and 4 do and it also adopts adaptive play-out scheduling, we compare the performance of this method with that of our methods in Chapter 6. The probing

[2]The predictor they use is simply the future R-factor equals the current R-factor.

overhead of this method (to measure the quality of a predefined set of candidate paths) is also compared in Section 5.5. For this method, the probing overhead at each node is $\frac{S((N-1)+(N-2)(N-1))}{\textbf{PROBE\_INTERVAL}}$[3], where S is the size of a probing packet, for an N-node full mesh overlay network and considering only direct or one-hop candidate paths. If we choose the same setting as in RON with $S = 69$ bytes, **PROBE_INTERVAL** $= 5$ ms and $N = 10$, the probing overhead is 1.1178 MBps.

2.3.3 Diverse Routing

Having discussed the related work on single best path routing, we continue to discuss another VoIP QoS routing method, i.e. diverse routing method. Diverse routing for VoIP and media transmission has been studied in some previous work [12–16], in which packets can be fully or partially duplicated for transmission on diverse paths. In this thesis we assume voice packets are fully duplicated[4] for diverse routing.

In [40], different combinations of diverse paths are experimented, which have been shown that diverse routing is able to reduce packet loss rate and packet delay more than that one single best path can do. When considering the effect of reactive best path routing and diverse routing on losses and delays, the authors [40] suggest that reactive routing avoids paths with long-term pathologies, and that diverse routing masks transient congestion-related losses and delays. Thus, intuitively, a method combining reactive routing and diverse routing is able to perform better in both cases, which becomes one motivation for our work in Chapter 6.

Paper [12] shows that diverse routing can improve VoIP quality. It was shown that in general VoIP quality improves mostly from one path to two-path diversity, while when four-path diversity is used, a quality decrease can happen due to link capacity saturation. Therefore, we only consider two-path diversity in this thesis.

Having shown the advantages of diverse routing above, next, I will discuss previous work on diverse path selection, which include brute-force search method [51], active path

[3]The probing overhead refers to the probing traffic received at each node. Each node receives (N-1) probing packets from direct path probing and (N-2)(N-1) probing packets from one-hop path probing.

[4]In cases when network bandwidth is limited, one can always apply multiple-description codec [50] or only duplicate the important voice packets (i.e. the packets containing transitional (unvoiced-to-voiced) frames and unvoiced frames that proceed voiced frames, which account for only around 11% of all the voice packets [16]). The quality for VoIP calls in this case is then highly related to the actual coding method, e.g. the specific multiple description codec, which is out of the scope of this thesis.

first method [17] and diverse path selection method in physical networks [52, 53].

Brute-force Search Method

Paper [51] uses brute-force search method to determine the diverse paths by solving the problem in (2.4) for each source-destination pair (u, v).

$$\begin{aligned} O' &= \arg\min_{k \in O} |p'(u, k, v) \cap p^*(u, v)| \\ k' &= \arg\min_{z \in O'} w(p'(u, z, v)) \end{aligned} \tag{2.4}$$

traceroute is used to measure links and link latencies between all overlay node pairs, which can be inaccurate. In (2.4), O is the set of candidate next-hop nodes for the source-destination pair (u, v), $p^*(u, v)$ is the direct Internet path for (u, v). $p'(u, k, v)$ denotes the one-hop redundant path via k. $|p'(u, k, v) \cap p^*(u, v)|$ denotes the number of joint links between paths $p'(u, k, v)$ and $p^*(u, v)$ in the underlying network. $w(\cdot)$ represents the cost or delay of a path. The redundant paths considered are only one-hop redundant paths. Thus, the first step to solve the problem is to maximize link disjointness between path $p'(u, k, v)$ and path $p^*(u, v)$, i.e. to find the set of next hop nodes O' that minimize the number of joint links between the direct path and the redundant path. Once O' is found, the optimal redundant path is selected as the one with the minimum cost.

One limitation of this brute-force search method is that it requires identifying the number of joint links between all the candidate secondary paths and the direct Internet path, which is not scalable with network sizes. The other limitation is that it only considers the combination of a one-hop path with the direct Internet path which may not provide better VoIP quality than the combination of two one-hop paths or two paths with more than one hop. In our work, we consider all the possible combinations of one-hop paths with or without the direct path in Chapter 4 and consider paths with more than one hop in Chapter 6.

Active Path First (APF)

Active path first method is in general known as using the Dijkstra's algorithm to find the shortest path first, and then finding a backup path from the remaining network using

Dijkstra's algorithm again. Cui, et al., [17] propose two APF methods to choose the backup paths based on a correlated link failure probability model in overlay networks. The goal of [17] is to minimize the link failure probability on the chosen pair of paths, which is an NP-hard integer quadratic programming problem. The authors propose two heuristics to solve the problem: (1) choose the primary path as the minimum delay path and choose the secondary path to minimize the joint failure probability with the primary path; (2) choose the primary path as the minimum delay path and choose the secondary path as a minimum delay path that is link disjoint to the primary path.

The limitation of this work is that it is a centralized method and it requires knowing link failure probability model for computing the optimal paths. In addition, it cannot adaptively route voice calls to avoid link failures. In our work in Chapter 6, we propose a distributed and adaptive routing approach that requires no prior knowledge about link failure model and it is able to adaptively route voice calls to avoid link failures.

Finding Diverse Paths in Physical Networks

Diverse path selection has also been studied in the context of survivable networks [52, 53], for example, to provide reliability against failures in optical networks. The problem of finding disjoint paths that minimizes/maximizes a specific objective can be formulated as an Integer programming problem [17, 54–56], which is usually NP-hard [57–59]. There are three well known problems, the Min-Sum problem (minimize the sum of the cost of the selected routes) [56, 60, 61], the Min-Max problem (minimize the maximum cost of the selected routes) [58] and the Min-Min problem (minimize the minimum cost of the selected routes) [59]. Li, McCormick and Simchi-Levi [58] have developed a heuristics to the NP-complete Min-Max problem. Suurballe, et al., [60, 61] have proposed algorithms for solving the Min-Sum problem. Xu, et al., [59] have proposed a heuristics to the NP-complete Min-Min problem. Book [52] has also given a thorough review of previously investigated diverse routing algorithms, where given the topology and the cost of each link, edge/node disjoint shortest pair of paths can be found. However, the algorithms in [52] are centralized algorithms which requires to know network topology and link cost. In our work, we propose a distributive method to minimize delay or maximize R-factor on disjoint paths, which is different from the Min-Max, Min-Min or Min-Sum problem. No distributive solution is known for such a problem in previous work.

2.3.4 Minimum Delay Routing

In this part, we discuss minimum delay routing, which is related to our work in chapter 5. To find the minimum delay paths, two categories of minimum delay routing methods have been reported. One category is to find the user optimal paths that minimize the estimated end-to-end delay for each packet [62] [25] [63]. The other category is to find the system optimal paths that minimize the total delay of all packets [64]. Both the system optimal minimum delay routing [64–68] and the user optimal minimum delay routing [24–26] have been studied extensively in the literature. The approach commonly used for system optimal routing problem requires estimating delay gradients [64–68], and the approach for user optimal routing problem usually requires estimating marginal delays [24–28].

In our work in Chapter 5, we also minimize delays for each source-destination pair, which belongs to the user optimal routing problem. These previous work on user optimal routing in packet switching networks [24–28] may be modified and applied for minimum delay routing in overlay networks. However, their complexity limits their practical values and they are more complex than our method in Chapter 5.

2.3.5 Reinforcement Learning (RL) for Adaptive Routing

Reinforcement learning methods [27, 28, 69–71] have been applied to adaptive routing in dynamic network environments. These methods can distributedly learn QoS-aware paths by online measurement. Q-routing [27, 28] is a value-search algorithm, which is basically a modified distance-vector algorithm, where Q-functions are estimated for finding the minimum cost paths. [71] is a policy-search algorithm, where an optimal (stochastic) routing policy is learned with a gradient ascent algorithm by trying and evaluating a set of policies. They both show the promise in providing QoS with RL-based adaptive routing control, however, the simulation settings of [27, 28, 71] are far from realistic.

Erol Gelenbe's Cognitive Packet Network (CPN) [69, 70, 72] learns QoS-aware paths by using "smart" packets for route discovery where the route of the "smart" packets are chosen based on Random Neural Network(RNN) based algorithms. The successful paths found by the "smart" packets are exploited by the "dumb" packets. This method is very similar to our learning automata based adaptive routing in Chapters 5 and 6 in spirit, however, the route learning algorithms of our method are much simpler and provably converge to the minimum delay paths in a stationary network environment. In addition, the experiments of [69, 70] are based on very simple network topologies, while the simulations in Chapters 5 and 6 take into account more aspects of the complex behaviour witnessed in realistic network traffic, including non-homogeneous networks with different link bandwidth, routing node buffer size limits and link propagation delays are simulated. The final difference between our work and CPN is that CPN method is not resilient to link failures since it only learns a single route between source and destination, while we develop a diverse routing method and a link failure detection method based on the reinforcement learning algorithm to provide resilience to link failures in Chapter 6.

2.4 VoIP in Service Overlay Networks

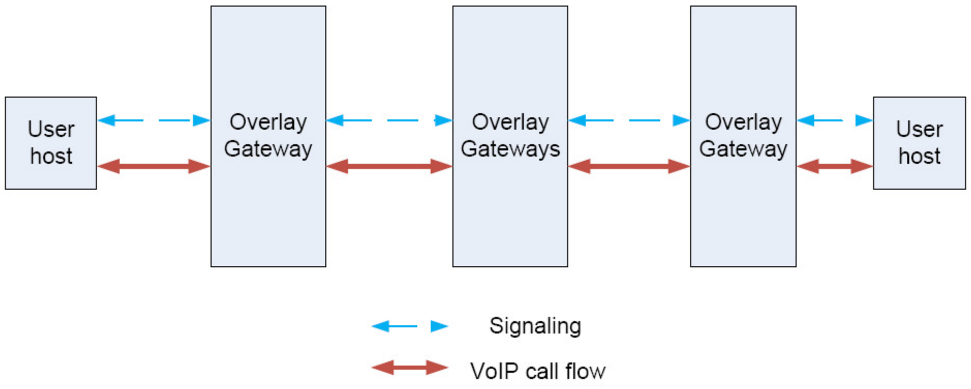


Fig. 2.6 Integrate call signaling and voice traffic transmission in the service overlay network.

Fig. 2.6 illustrates the signaling and voice traffic transmission in an overlay network, where the overlay gateways function as proxies or relays for VoIP calls. For implementation in the overlay gateways, the proposed routing method should be integrated with the known VoIP framework of SIP [73] or H.323 [74], i.e., the signaling functions including call establishment, access control and disconnection.

The call signalling and voice call transmission sequence are illustrated in Fig. 2.7. The first phase is the call establishment period, illustrated by the dashed arrows in the top, where a session is initialized for the caller and callee, while the route for this call is also determined in the calling period. Once a voice session is established, voice packets can be transmitted over the overlay network, as illustrated by the solid arrows in the middle.

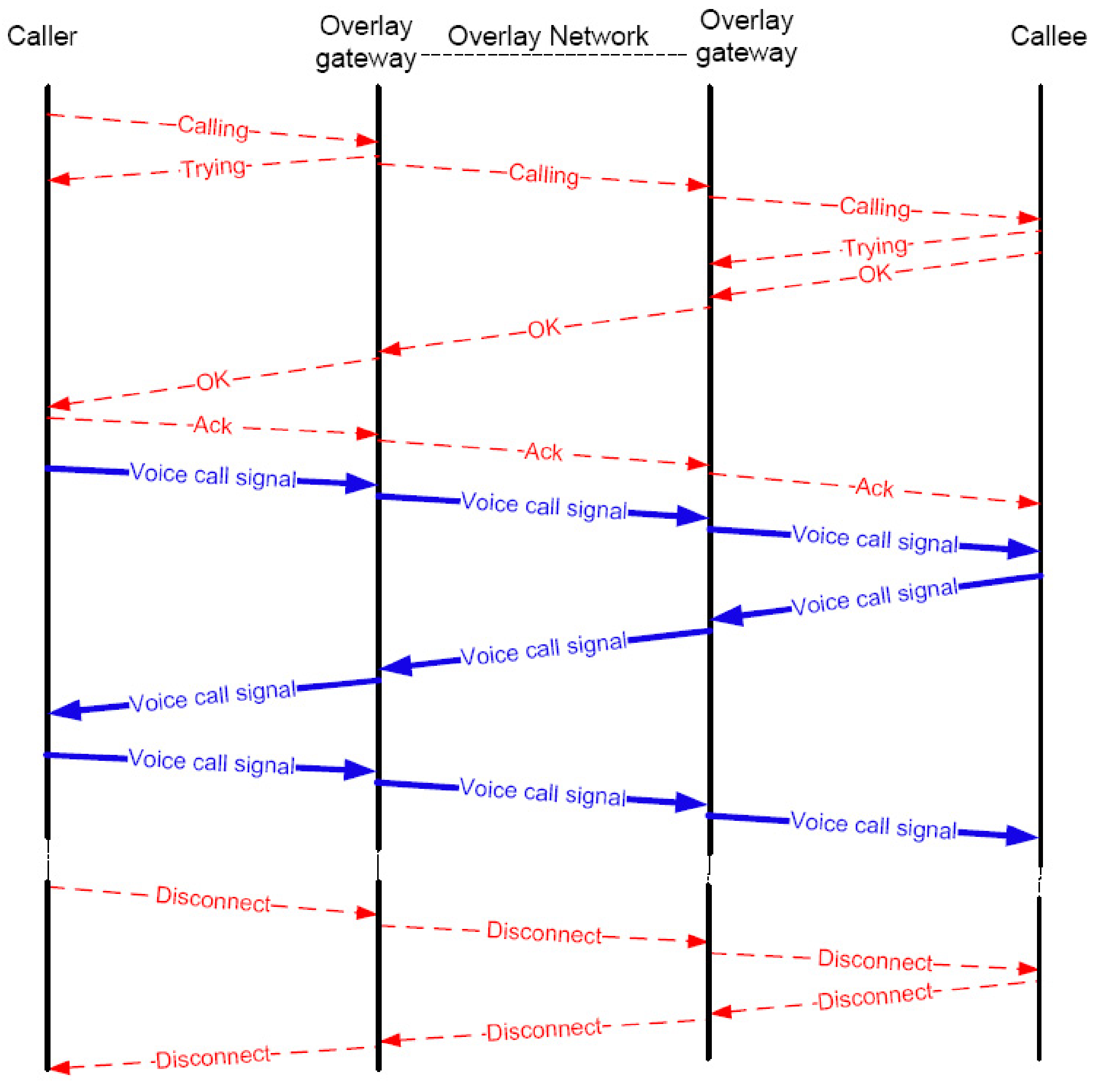


Fig. 2.7 Call signaling and call transmission process. The "Calling" message refers to the initialization of call establishment. The "Trying" message indicates the gateway is trying to connect to the callee. When the callee picks up the phone, an "OK" message will be sent to the caller, and the caller replies with a "Ack" message to complete the call establishment process. Then the conversation between the caller and callee can start. When one side hangs up the phone call, a "Disconnect" message is sent to the other side and the other side responds with a "Disconnect" message to end the voice session.

The last phase is the call disconnection period, where the voice session is terminated, as illustrated by the dashed arrows in the bottom.

The underlying transport layer that supports the VoIP service over a Service Overlay Network can be TCP [75] or UDP [76]. However, we consider using UDP protocol to transmit voice calls in this work since TCP introduces delays at source. The potential losses due to using UDP protocol can be masked by using diverse paths.

2.5 Summary

Service overlay networks, driven by the fast development of VoIP, video streaming and content distribution, provide a cost-effective way to support Quality of Service over the current best-effort Internet. Previous work on QoS routing in overlay networks [12–14, 40, 45, 46, 48, 51, 77–79], has shown that alternative paths can provide better quality for applications than the direct paths. Brute-force search method, modified Dijkstra's algorithm and modified distance-vector algorithm, et. al, have been studied in previous works [12–14, 40, 45, 46, 48, 51, 55, 77, 78] for finding the single best paths and diverse paths. However, these works all have their limitations as mentioned in each section. For example, they usually consider one-hop or two-hop paths that are not necessarily the best path for VoIP, and they are often centralized methods which require link cost broadcasts or require a central server to collect link costs when estimating path quality. These constraints limit the optimality and the scalability of those approaches. Therefore, in our work, we investigate how to select the optimal diverse paths for VoIP and how to achieve a scalable distributed overlay routing method.

A preview of our work on overlay routing is as follows. Chapter 4 presents an optimal R-factor-based diverse routing approach; Chapter 5 proposes a scalable minimum delay path learning approach; Chapter 6 proposes a distributive approach to learn diverse paths. In the next chapter, we will first present our work on end-to-end delay analysis, simulation and sampling, which are important for understanding the end-to-end delay characteristics and thus to discover potential method for QoS routing in service overlay networks.

Chapter 3

End-to-end Delay Trace Analysis, Simulation and Sampling

3.1 Introduction

End-to-end delay is one of the most important impairments for VoIP phone calls. Thus,this chapter is dedicated to the investigation on end-to-end delays. For the preliminary research on VoIP QoS routing, we require end-to-end delay traces for a long period of time for many source-destination pairs. To save the substantial cost of gathering traces, especially for large networks with hundreds of nodes, a practical way is to generate end-to-end delay traces that are statistically similar to the real end-to-end delay traces. Thus, in Section 3.2, we analyze the statistical characteristics of real end-to-end delay traces to understand end-to-end delays in depth so that we can simulate end-to-end delays for VoIP QoS routing research, which is given in Section 3.3 and Appendix C. Another practical problem is the optimal sampling problem. Supposing we have collected a large quantity of end-to-end delay traces that require a large amount of storage, we may desire to only store parts of the traces while still be able to reconstruct the original end-to-end delay traces. In this case, an optimal sampling rate is desired such that the trade-off between the reconstruction accuracy and the sampling/storage cost can be fairly resolved, which is solved in Section 3.4.

As end-to-end delay analysis, end-to-end delay simulation and end-to-end delay sampling all focus on investigating end-to-end delays. Therefore we combine them into one chapter. We also place this chapter before Chapters 4, 5 and 6 on VoIP QoS routing because understanding end-to-end delay characteristics is key to VoIP QoS routing.

3.1.1 Chapter Structure

This Chapter is divided into three parts. Section 3.2 analyzes real end-to-end delay traces. Section 3.3 presents a network model designed for simulating realistic end-to-end delay traces. Finally, Section 3.4 shows the work on end-to-end delay sampling. At the end, Section 3.5 summarizes the three parts of work.

3.2 Analysis of real end-to-end delay traces

We first analyzed real end-to-end delay traces collected by an Internet Service Provider (ISP) in Asia. The network delay measurements under study are Round-Trip-Time (RTT). The measurement method the ISP adopted was to send 100 consecutive probing packets(with 30 milliseconds (ms) interval) per 2.5 minutes. The consecutively sent probing packets are used to emulate a talk spurt. We obtained two sets of raw data from the Internet Service Provider. The first set of data is the RTT measurement between Singapore and 15 other network nodes (located in the USA, China, Korea, Japan, Singapore and Malaysia) in a time period T_1, which lasts 6.8 hours. Denote this set of measurement data as $X_{K\times L}$, where $K = 16300$ is the number of samples, $L = 15$ is the number of overlay links. The second set of data are sample means of each 100 RTT measurements in a different time period T_2, which lasts 277.5 hours, among network nodes (again located in the USA, China, Korea, Japan, Singapore and Malaysia). Denote this set of measurement data as $Y_{N\times M}$, where $N = N1 \times N2$ represents the number of overlay links with $N1 = 14$ source nodes and $N2 = 11$ destination nodes, and $M = 6660$ is the number of sample means per overlay link.

As the first set of data mentioned above is at finer time granularity, we use it to analyze the property of network delay trace[1]. The second set of data will be used for end-to-end delay synthesis in Appendix C.

[1]As network is dynamic and keeps evolving, we cannot claim the traces we analyzed here represent all the delay traces in the Internet. However, the properties we obtained in the first and second order statistical analysis, e.g. relationship between the propagation delay delay and geographical distance, originate in either network design (e.g. network topology and routing protocol [33]) or network traffic dynamics (self-similarity [80]). Some similar properties (e.g. triangle inequality, Gamma distribution, delay spikes) have also been observed in traces collected by other work [16,81–83]. Also note that I focus on analyzing delays instead of loss patterns in this work because a good analysis of loss pattern requires a large amount of measurement, while we only have 16300 delay samples for each of the 15 links.

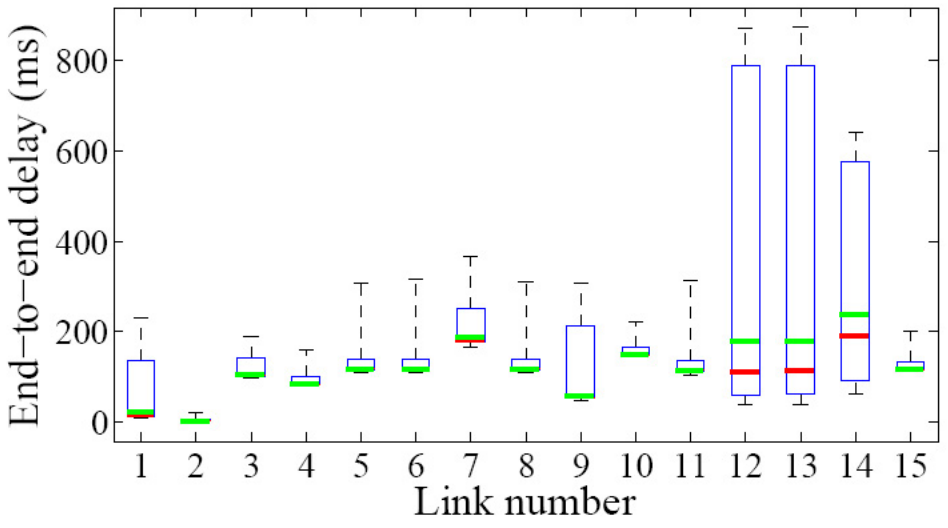


Fig. 3.1 This figure shows, from bottom to top, the minimum, 25% quantile, median, mean, 99% quantile and the maximum of end-to-end delay values on each of the 15 overlay links. Each box contains delay values between quantiles 25% and 99%. Note that on some links, the minimum, 25% quantile, median and mean values are so close that their difference is indistinguishable.

First, let us consider the minimum, 25% quantile, median, mean, 99% quantile, and the maximum of the RTT measurements in the first set of data, as given in Fig. 3.1. This figure shows the RTT range from less than 10 ms to more than 800 ms although the median and mean of the delay values are mostly less than 250 ms. Furthermore, the large difference between 99% and 50% quantiles of the delay values shows the existence of large delay spikes, which can be detrimental to VoIP quality. This reinforces the importance of designing QoS routing schemes to choose paths with low end-to-end delays and jitters. In Chapter 4, I will present a QoS routing scheme to deal with this issue.

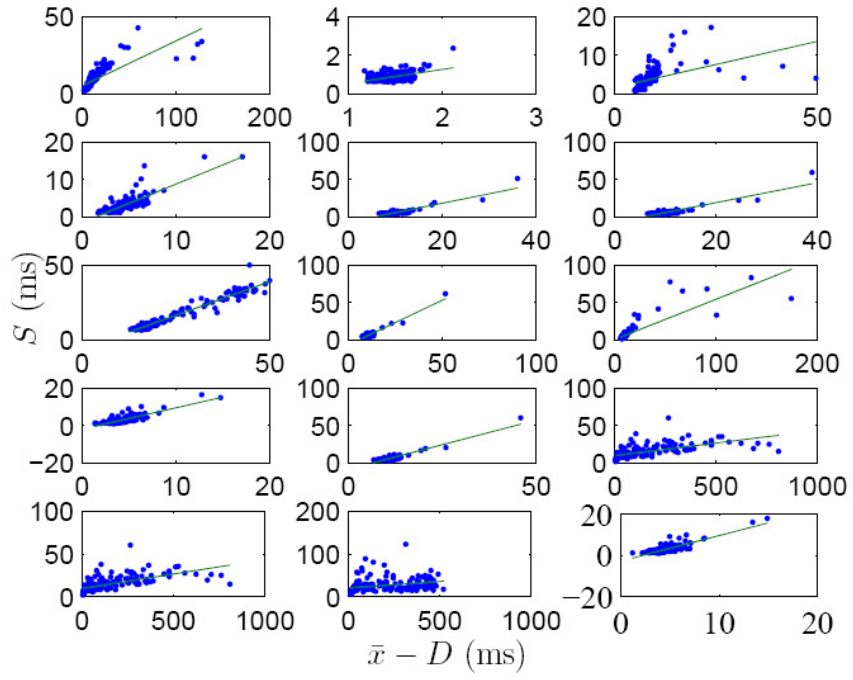


Fig. 3.2 Scatter plot for S and ($\bar{x}$ − D), and the linear fit to $\frac{\mathrm{S}}{\bar{x}-\mathrm{D}}$ on 15 different links (the links are ordered from the left to the right and from the top to the bottom). S is the standard deviation per 100 network delay samples. $\bar{x}$ is the average network delay per 100 network delay samples on a link. D is the minimum of all the network delay samples on a link. $\bar{x}$ − D can be considered as the average queueing delay per 100 network delay samples on a link.

3.2.1 The sample mean and the sample standard deviation of a delay trace

Given a random sequence $\{X_1, X_2, ..., X_M\}$, its sample mean $\bar{x}$ and its unbiased standard deviation S are

$$\begin{aligned} \bar{x} &= \frac{1}{M}\sum_{i=1}^{M} X_i \\ \mathrm{S} &= \sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(X_i - \bar{x})^2} \end{aligned} \tag{3.1}$$

. Let the minimum of the random sequence be $\mathrm{D} = \min\{X_1, X_2, ..., X_M\}$. The relation between S and $\bar{x} - \mathrm{D}$ for measured delay is shown in Fig. 3.2 (M=100 for the figure). There is evidently a strong linear correlation between S and $\bar{x} - \mathrm{D}$ in most cases. A similar result can be found for S and $\bar{x}$. Such a relation maybe used to linearly estimate S from $\bar{x} - \mathrm{D}$.

Fig. 3.3 shows two representative network delay traces[2] that have relatively strong correlation between their sample mean and sample standard deviation. The links that have similar property as link 3 and 4 are links 1, 2, 3, 4, 5, 6, 7, 8, 9, 11 and 15, as indicated in Fig. 3.2. The figures on the right of Fig. 3.3 show the sample mean $\bar{x}$ and the sample standard deviation S per 100 delay samples for the delay traces on the left. By comparing Fig. 3.2 and Fig. 3.3, we can see that the correlation on link 4 is a little higher than that on link 3[3], but both have short delay spikes, which can be due to short bursts in traffic arrival. The delay traces of the other links in this group all appear similar to those in Fig. 3.3 and are not shown here to save space. Due to the strong correlation between S and $\bar{x} - D$ for this type of links, one may linearly estimate S from $\bar{x} - \mathrm{D}$ or $\bar{x}$ as D is a constant for a specific link.

For links 12, 13, and 14, we observe much longer spikes, which can be interpreted as the result of long bursts in traffic arrival and link congestions. For example, Fig. 3.4 shows the delay traces of link 12 and link 14, (link 13 performs similar to link 12). On links 12 and 14, the sample standard deviation does not increase drastically with the sample mean[4], in

[2]Note that in the network delay trace, we miss 147 seconds of measurements per 2.5 minutes because 100 probing packets with 30 ms interval are sent per 2.5 minutes.

[3]Note that there is a short period of route change on link 3 at time steps 5300 and 10500.

[4]The network delay traces shown in Fig. 3.3 and Fig. 3.4 are only “continuous” per 3 seconds. Although

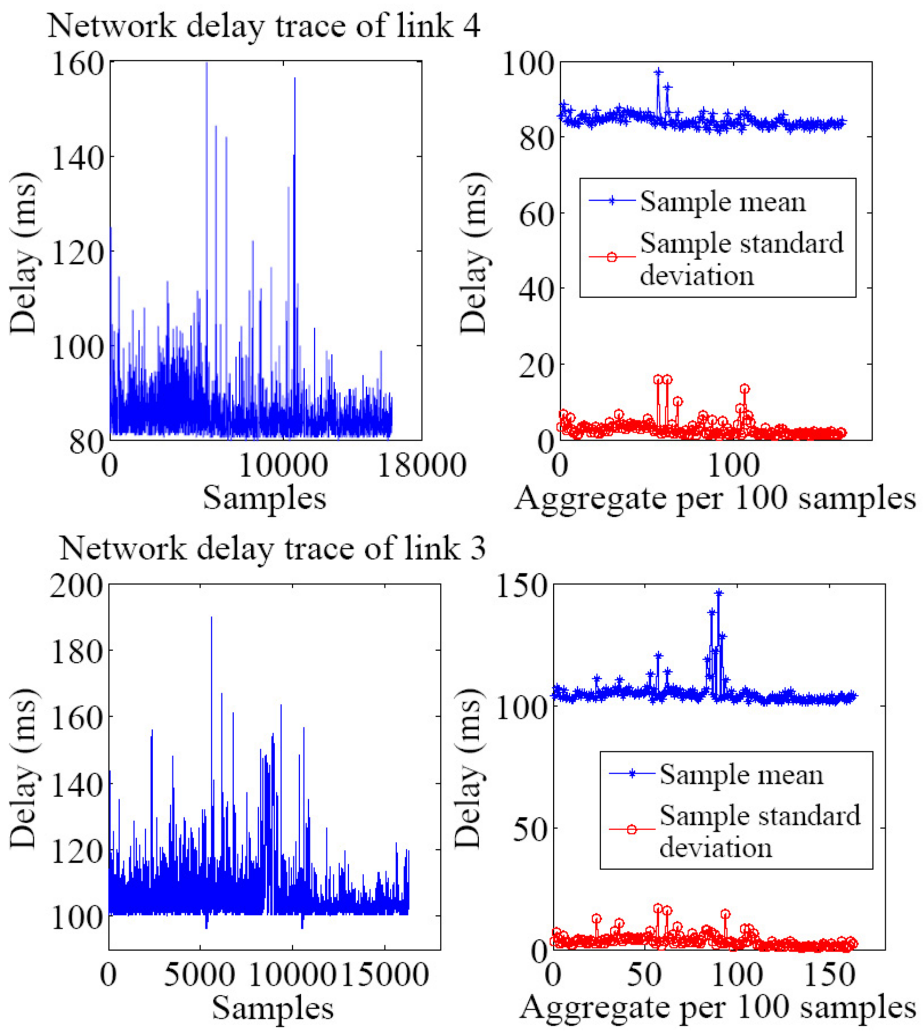


Fig. 3.3 Representative network delay traces for links with random delay spikes and strong correlation between the sample mean S and the sample standard deviation $\bar{x}$, where S and $\bar{x}$ are calculated based on per 100 network delay samples, i.e. M=100 in (3.1).

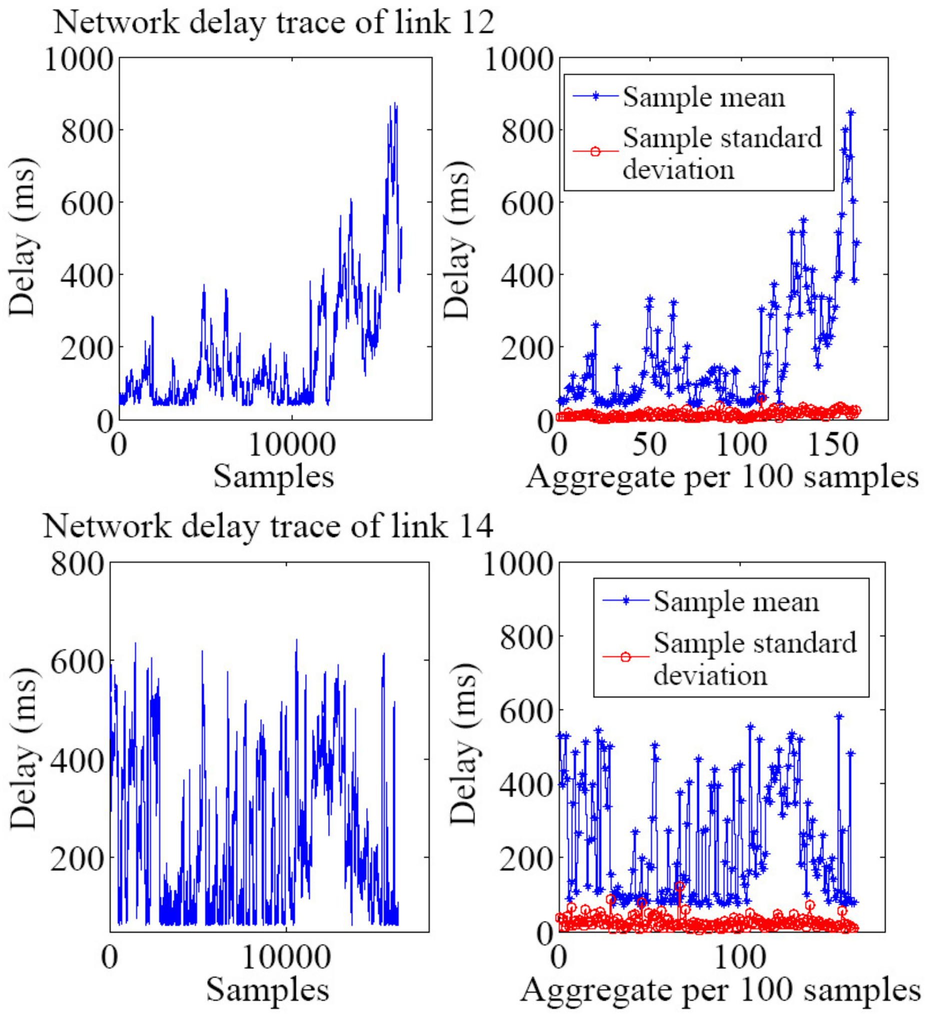


Fig. 3.4 Representative network delay traces for links with long delay spikes and weak correlation between the sample standard deviation S and the sample mean $\bar{x}$, where S and $\bar{x}$ are calculated based on per 100 network delay samples, i.e. M=100 in (3.1).

other words, they are not strongly correlated; thus, it is hard to linearly estimate S from $\bar{x} - D$ or $\bar{x}$ for links that show long delay spikes such as links 12, 13 or 14.

3.2.2 Probability distribution

End-to-end network delay consists of three components: propagation delay, transmission delay and queuing delay. Propagation delay is a constant value due to signal propagation between two ends, and is the minimum delay that a voice packet can experience. Transmission delay is considered to be negligible for high speed links, while queuing delay is a random variable, which follows a certain distribution.

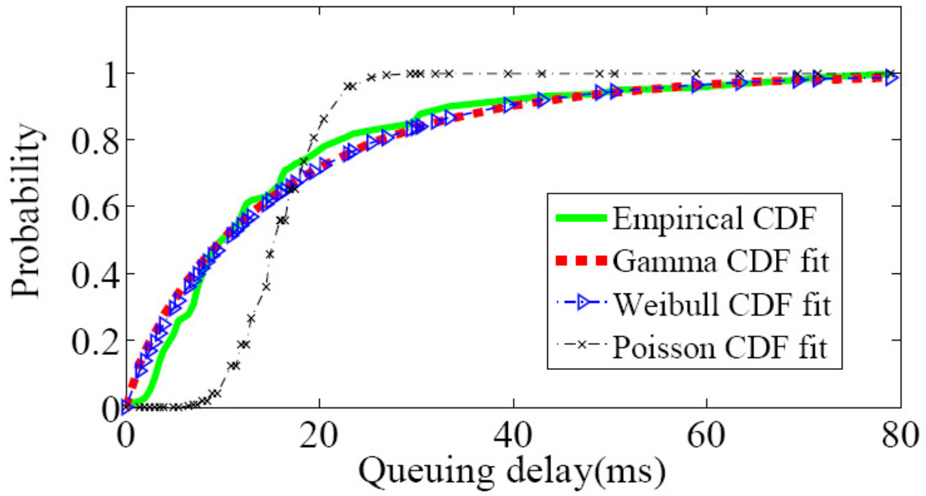


Fig. 3.5 Cumulative Distributions Function (CDF) fitted to 100 queuing delay samples (the propagation delay has been removed from the measured network delay samples). In this figure, the Gamma distribution [83] and Weibull distribution [80] both give good fit to the empirical distribution, and their difference is visually indistinguishable.

	Gamma	Poisson	Weibull
KL-divergence	0.4166	1.4485	0.3537
ℓ_1 norm	0.6293	1.4285	0.6020

Table 3.1 KL-divergence and the L_1 norm of the difference between the empirical PDF and the fitted distribution.

they have gaps in time, they do show the bursts and congestion characteristics as these characteristics appear very frequently in the samples.

We fit various commonly known positive-valued parametric distributions to the real network delay measurements obtained from the ISP mentioned before. We find that Gamma and Weibull distributions are good approximations to the empirical distribution of the queuing delays [5]. As shown in Fig. 3.5 and Table. 3.1, Gamma distribution is a good fit to the 100 queuing delay samples, and it is also true for a larger number of delay samples on many different links[6].

In fact, Gamma distribution can represent many commonly seen distributions. For example, by setting $\gamma = 1$, we get an exponential distribution; by setting γ to be an integer, we get an Erlang distribution; by setting $\gamma = v/2$ and $\beta = 2$, we get a Chi-square distribution. Therefore, according to the distribution fitting above and due to the amenable properties of Gamma distribution, we use Gamma distribution to model the queuing delay distribution. Then the distribution of the total network delay can be modeled as a shifted Gamma distribution, whose Probability Distribution Function (PDF) is given by:

$$f(x) = \frac{(\frac{x-\theta}{\beta})^{(\gamma-1)} \cdot exp(-\frac{x-\theta}{\beta})}{\beta \cdot \Gamma(\gamma)}; x \geq \theta; \theta, \gamma, \beta > 0, \tag{3.2}$$

in which γ is the shape parameter, β is the scale parameter, and θ represents the minimum network delay experienced by a voice packet, i.e. the propagation delay. As a result, the mean network delay μ is $\mu = \theta + \gamma\beta$, the standard deviation $\sigma = \sqrt{\gamma}\beta$, and the skewness k is $k = \frac{2}{\sqrt{\gamma}}$. In fact, when $\theta = 0$, f is a Gamma distribution which models the queuing delay distribution. Such shifted Gamma distributed model for end-to-end delays can also be found in [83, 84].

3.2.3 Relationship between the minimum delay and the geographical distance

By locating each network node based on its IP address[7], we can compute the geographical distance between nodes. As the scatter plot of Fig. 3.6 shows, we find that the correlation between the minimum delay and the geographical distance between two nodes is low because

[5]The distribution fitting is performed as follows. Given the real data and a parametric distribution, the maximum likelihood estimate of the parameters of the parametric distributions can be computed.

[6]Gamma distribution gives relatively large log-likelihood comparing to that of other commonly known positive-valued parametric distributions. It can also be expected that a non-parametric distribution will provide higher log-likelihood of estimation comparing to that of any parametric distribution, but a non-parametric distribution is hard to generalize to all the delay traces on different links.

[7]The location resolution is at a city level.

the physical length of an overlay link between two nodes is not necessarily equal to the geographical distance between them.

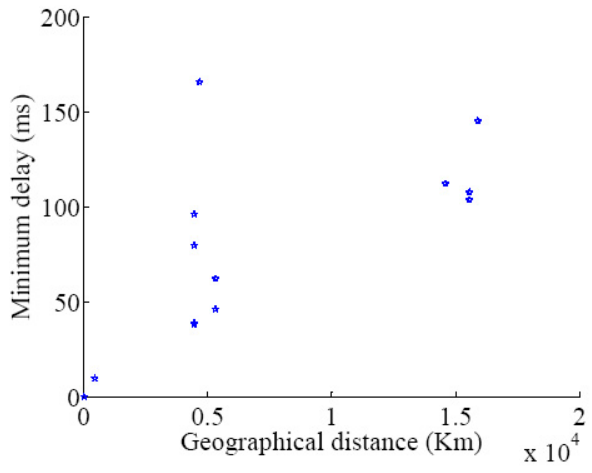


Fig. 3.6 Scatter plot for the minimum delay D and the geographical distance of 15 links. It shows there is low correlation between them.

3.2.4 Autocorrelation

The autocorrelation coefficient r_k of lag k for a time series $\{X_1, X_2, ..., X_M\}$ can be computed as[8]:

$$r_k = \frac{\sum_{i=1}^{M-k}(X_i - \bar{x})(X_{i+k} - \bar{x})}{\sum_{i=1}^{M}(X_i - \bar{x})^2}, \tag{3.3}$$

where $\bar{x}$ is the sample average of X_i, with $\bar{x} = \frac{\sum_{i=1}^{M} X_i}{N}$.

By plotting the autocorrelation function of the real network delay measurements on different links in the first group represented by Fig. 3.3, we see that the autocorrelation is usually negligible when there are only random delay spikes, as in the examples of Fig. 3.7. Therefore, an identical and independently distributed shifted Gamma distribution model can be used to simulate the network delays when the time series has no spike or random spikes. However, if there is strong or medium autocorrelation in the time series, as for the second group of links shown in Fig. 3.4, the model would need to be modified to take

[8]Note that autocorrelation function computed with finite size of samples can result in oscillation in the autocorrelation function.Similar oscillation can also be observed in computing cross-covariance with finite size of samples. An analysis of the effect can be made using Fourier theory, which is left for future work.

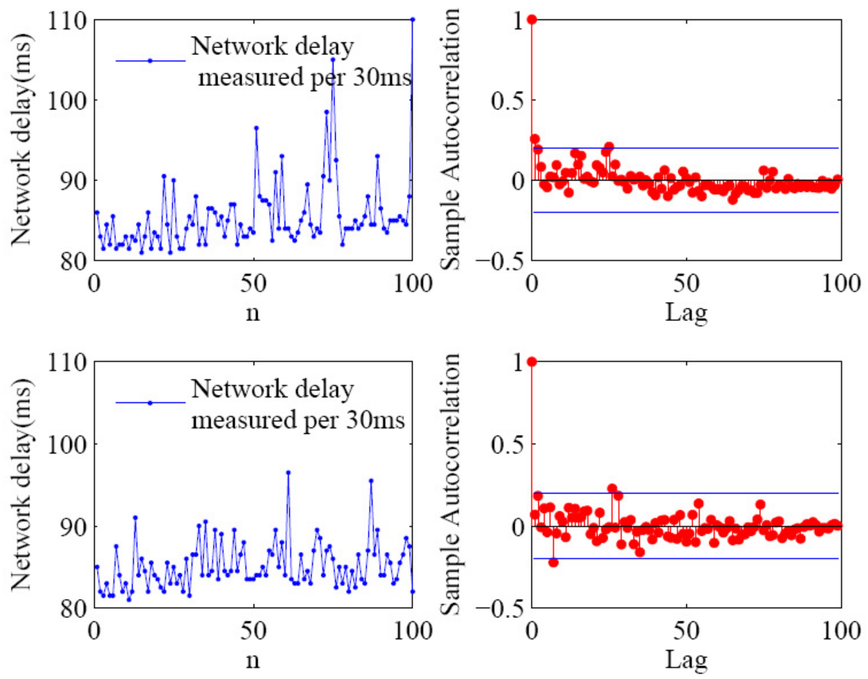


Fig. 3.7 The figures on the left show the real network delay measurements. The figures on the right show the autocorrelation computed for the 100 network delay samples in the left figures, which are negligible.

account of autocorrelation. In this case, a possible model would be an Auto Regressive (AR) model with a Gamma distributed residual, as with the ARTA process [85].

3.2.5 Cross-covariance between delay traces

The cross-covariance of two random sequences $X = \{X_1, ..., X_M\}$ and $Y = \{Y_1, ..., Y_M\}$ can be computed as follows. Let the sample mean of $X_k, k = \{1, ..., M\}$, be $\mu_X = \frac{1}{M}\sum_{k=1}^{M} X_k$, and the sample mean of $Y_k, \{k = 1, ..., M\}$, be $\mu_Y = \frac{1}{M}\sum_{k=1}^{M} Y_k$. Denote the conjugate of Y_k as Y_k^*. Then, the cross-covariance between X and Y is:

$$C_{XY}(n) = \begin{cases} \frac{1}{C_{XY}(0)} \sum_{k=1}^{M-k} (X_{k+n} - \mu_X)(Y_k^* - \mu_Y), n \geq 0 \\ C_{YX}^*(-n), n < 0 \end{cases} \tag{3.4}$$

The cross-covariance is computed between all the link delay traces on the measured 15 links of the ISP. We see that the cross-covariance between some link delay traces is relatively strong, as shown in Fig. 3.8, which is because a large delay spike occurs on each of two links under consideration with a time difference of around 150 seconds. Thus it is evident that there can be relatively strong correlation between delay spikes on different overlay links due to correlated traffic congestion on these links.

3.3 End-to-end Delay Trace Simulation with Fluid Network Model

Based on the analysis above, we can synthesize delays that are statistically similar to the real end-to-end delay traces that have weak or no autocorrelation (see Appendix C). However, it is not easy to synthesize traces with strong autocorrelation or cross-correlation among different links. In this section, we therefore use a traffic model that incorporates time zone information and design a simple capacity assignment method to simulate realistic network delays.

Fluid network simulation [86] has been shown to be much faster than NS2 [87] simulation. It is more scalable to simulate large scale networks than the packet level simulation does [86]. The simulation result for discrete time fluid network is very similar to that of packet level simulation [88].

The original contribution in this section is to apply fractional Brownian motion traffic

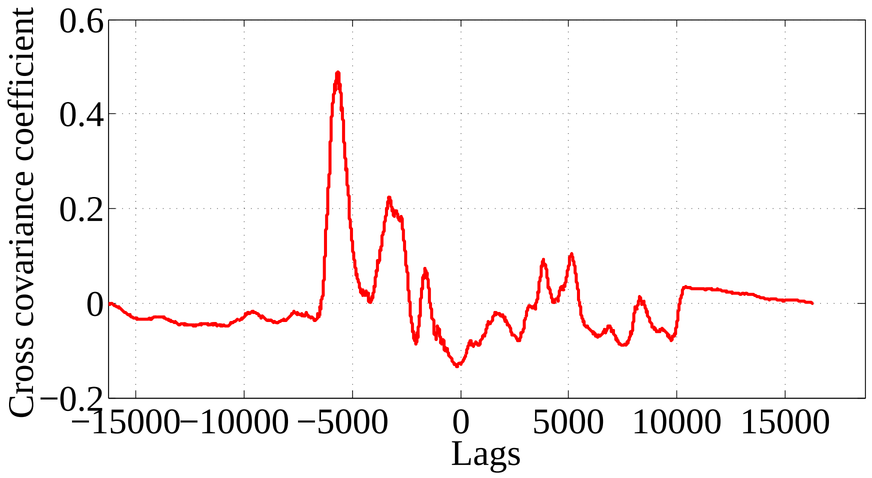


Fig. 3.8 Cross covariance coefficient between the delay trace of link 9 and that of link 13. As can be seen, the cross-covariance between the delays on the two links is relatively strong. This large cross covariance coefficient at around -5000 lags is because two large delay spikes on the two links occur with 5000 lags of samples, as illustrated in Fig. 3.9. Note that this cannot be interpret as a traffic surge passing two links consecutively since the time different for the 5000 lags is $2.5 \times \frac{5000}{100} = 125$ minutes.

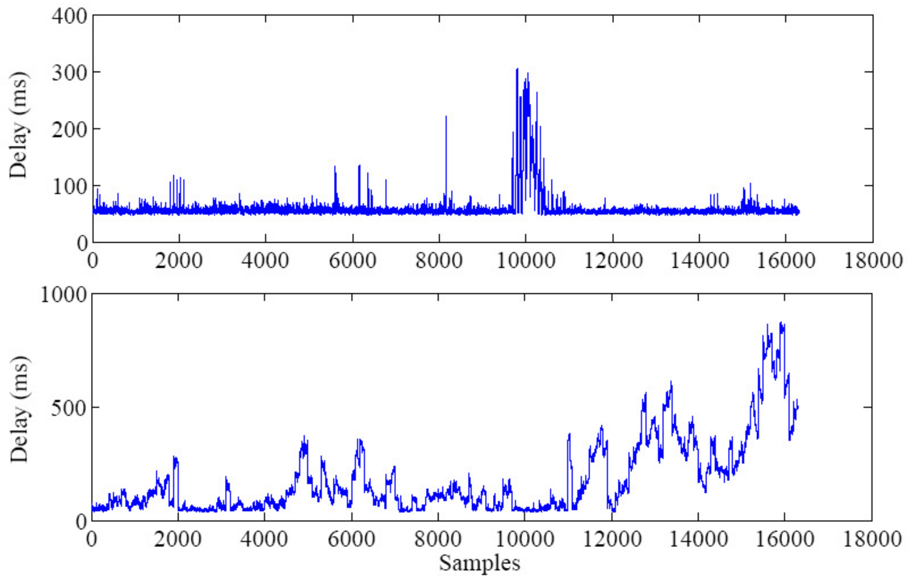


Fig. 3.9 Delay traces of link 9 (top) and link 13 (bottom). The cross-covariance between the two delay traces is shown in Fig. 3.8.

model [80] and the gravity traffic model [89] to simulate dynamic traffic among network nodes, and link capacities are also chosen to closely simulate realistic network design.

Note that the goal of the delay simulation work is to simulate network delay traces that show autocorrelation and cross-covariance, as well as delay spikes and triangle inequality[9], as observed in the real network delay traces (see Section 3.2 and [16, 82]).

3.3.1 Fluid Network Model

Fig. 3.10 illustrates a fluid network model [88]. In this figure, the on-the-fly time of a link l, with propagation delay D_l, queuing capacity q_l^{max} and bandwidth c_l, is divided into $d_l + \frac{q_l^{max}}{c_l \tau}$ time slots with each time slot being τ, where d_l is the quantized propagation delay (i.e. $\frac{D_l}{\tau}$). Then each flow transmitting on this link is moved forward in these time slots, as shown in Fig. 3.10. In the fluid network model, the end-to-end delay of each flow can be

[9]Triangle inequality means that the delay on an alternate path can be less than that on the direct Internet path, which is the key why overlay routing can provide better path than the direct Internet path. This property is not only observed in the MediaRing trace mentioned in Section 3.2. We also observed it in collected delay traces from PlanetLab [81]. It is also observed in other research [82].

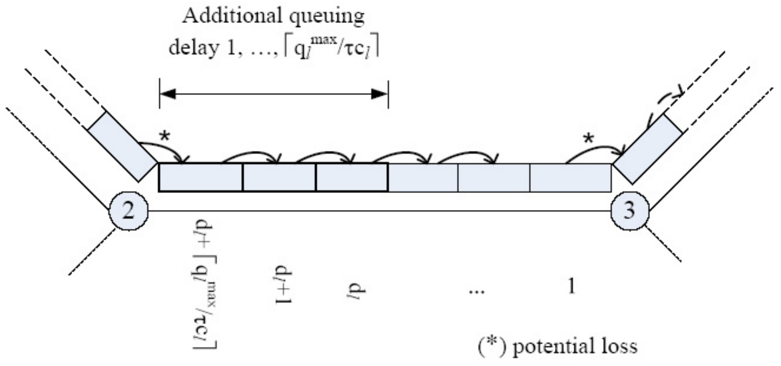

Additional queuing delay 1, …, ⌈$q_l^{max}/\tau c_l$⌉
*
*
2
3
$d_l+\lceil q_l^{max}/\tau c_l \rceil$
d_l+1
d_l
…
1
(*) potential loss

Fig. 3.10 Fluid network model [88]. d_i is the quantized propagation delay. q_l^{max} is the queuing capacity. c_l is the link bandwidth. τ is the time slot size.

tracked as in (3.5) and (3.6) [88].

$$Delay^a_{next(a,l)}[n + d_l + j]^+ = Delay^a_l[n] + d_l + j, \forall j, i < j \le k. \tag{3.5}$$

$$\begin{aligned} Delay^a_{next(a,l)}[n + d_l + i]^+ &= Delay^a_{next(a,l)}[n + d_l + i]^- \cdot \Big(\frac{\lambda^a_{next(a,l)}[n + d_l + i]^-}{\lambda^a_{next(a,l)}[n + d_l + i]^+}\Big) + \\ & (Delay^a_l[n] + d_l + i) \cdot \Big(1 - \frac{\lambda^a_{next(a,l)}[n + d_l + i]^-}{\lambda^a_{next(a,l)}[n + d_l + i]^+}\Big). \end{aligned} \tag{3.6}$$

$next(a, l)$ is the next hop after link l for flow a, $\lambda^a_{next(a,l)}$ is the traffic arrival of flow a at link $next(a, l)$, and $q_l[n]$ is the queuing length at time step n for link l, $k = \frac{q_l[n+1]}{c_l\tau}$ and $i = \frac{q_l[n]}{c_l\tau}$.

Similar to the above, we can derive a function for tracking the end-to-end loss of each flow as (3.7).

$$Loss^a_{next(a,l)}[n + d_l + j]^+ = Loss^a_{next(a,l)}[n + d_l + j]^- + \frac{\lambda^a_l[n]p_l(n)}{k - i + 1}, \forall j, k \ge j \ge i. \tag{3.7}$$

where $p_l(n)$ is the loss rate on link l at time step n. (3.7) means that the loss on link $next(a, l)$ at step $n + d_l + j$ for flow a is the sum of the loss before adding the loss from link l, $Loss^a_{next(a,l)}[n + d_l + j]^-$ and the loss from link l at time step n, which is divided into $k - i + 1$ parts, $\frac{\lambda^a_l[n]*p_l(n)}{k-i+1}$.

The accuracy of the discrete time fluid network simulation can be tuned by changing time step size τ. A large step size of 10 ms can still capture the basic network behavior [88]. In our simulation, we set $\tau = 5$ ms.

Note that we built our own fluid network simulation platform in MATLAB because previous work [86,88] is used mainly for TCP flow simulation and has not considered relatively realistic network topologies for simulating relatively realistic network delays. In my work, I simulated self-similar traffic, i.e. fractional Brownian motion [80] traffic, as background traffic[10] in a representative network topology, i.e. AT&T backbone network, with relatively realistic network setting in the fluid network model, in order to simulate delays that show autocorrelation, cross-covariance, large delay spikes, and triangle inequality, as observed in the real delay traces in Section 3.2.

3.3.2 Network Setting

To test this, we simulated a model of the AT&T backbone network, as shown in Fig. 3.11. It includes 50 nodes located in the major cities of the United States and 196 directed links. Each node represents a Point of Presence (PoP). The network topology is inferred from Rocketfuel data [91] and the AT&T optical network topology [92]. From Rocketfuel data, we extracted a network topology with 103 network nodes (including stub nodes and transit nodes). The traffic from leaf nodes can also be combined with the traffic in transit nodes if single-homing for stub nodes is employed, allowing us to remove the leaf nodes from the network. However, if multi-homing is enabled, the stub nodes should not be removed. We finally obtain the 50-node network topology by doing single-homing and multi-homing for the stub nodes according to the AT&T link level optical network topology.

Aggregate Background Traffic Model

Research has shown that Internet traffic shows self-similarity. The fractional Brownian motion process is a well known long range dependent self-similar process. In Norros' paper [80], the accumulated traffic arrival in time $(-\infty, t)$ is represented by a fBm process A_t, where $A_t = mt + \sqrt{am}B_H(t)$, $t \in (0, \infty)$, with m being the mean traffic arrival rate, a being the variance coefficient for traffic arrival, $B_H(t)$ being a normalized fBm process and H being the Hurst parameter of $B_H(t)$. The increment process for this fBm process is a fractional

[10]Internet traffic has been found to be self-similar [80,90].

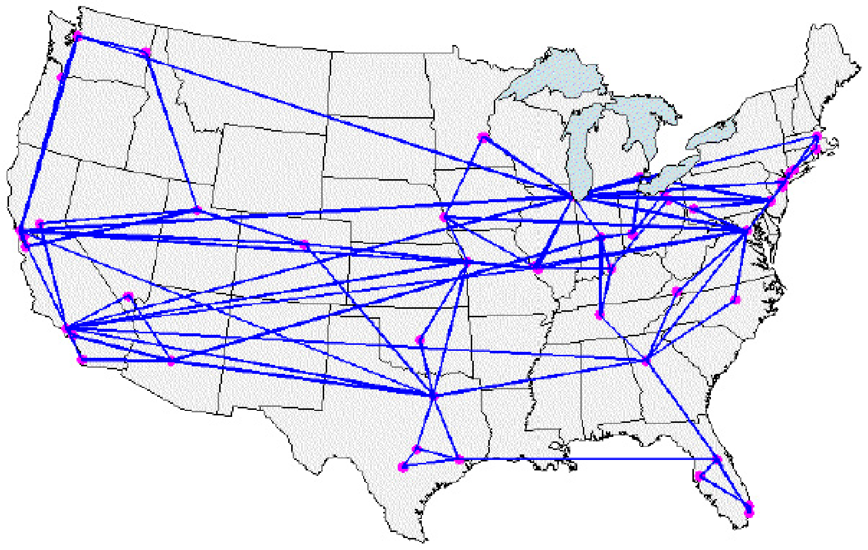


Fig. 3.11 50-node model of AT&T backbone network with 196 directed links.

Gaussian noise (fGn) process, which is a long range dependent, self-similar and stationary process. Given parameters m, a, H, a fGn process can be generated. Thus, we can use the fGn process to model the dynamics of traffic demand, and study the stationary network behavior for long range dependent and self-similar traffic arrivals. The autocorrelation function for a fractional Gaussian noise (fGn) with Hurst parameter H is $\rho(s) = \frac{1}{2}[(s+1)^{2H} - 2s^{2H} + (s-1)^{2H}]$, $0.5 < H < 1$.

Traffic Demand

Gravity model [89] [93] has been used to model mean network traffic demands between ingress and egress points of a network. Let $\mathrm{M}_{SD}(t)$ be the mean traffic demand from node S to node D at Greenwich Mean Time (GMT) time t, and let $\mathrm{P}(t, S)$ be the number of active population at node S as a function of GMT time t as measured in [89]. Given that the average traffic generated per person is a constant α, the mean traffic originating from node S is set to be proportional to the total active population $\mathrm{P}(t, S)$ of the metropolitan area where node S is located, i.e. $\alpha\mathrm{P}(t, S)$. In addition, the variance of the traffic demand between a source-destination pair is set to introduce realistic queuing delays in the network. In the gravity model, the mean traffic demand $\mathrm{M}_{SD}(t)$ from an origin node S to a destination node D is proportional to the total traffic leaving node S and that entering node D. Let $\mathrm{G}_{SD}(t)$

be the fan-out factor of traffic originating from node S and entering node D, proportional to the number of active population $\mathrm{P}(t, D)$ of node D at time t, generating the following two equations:

$$\mathrm{M}_{SD}(t) = \alpha \mathrm{P}(t, S)\mathrm{G}_{SD}(t), \tag{3.8}$$

$$\mathrm{G}_{SD}(t) = \frac{\mathrm{P}(t, D)}{\sum_D \mathrm{P}(t, D)}. \tag{3.9}$$

In the present simulation, the total population at each node is obtained from census data [94]. The average traffic demand per person is set to $\alpha = 0.01$ kb/5ms/person, i.e. 5.184 Gb or 648 MB per month per person, the Hurst parameter $H = 0.8$.

Capacity assignment

As we do not know link capacities in the AT&T network, in order to simulate realistic end-to-end network delays, the capacity of each link must be set properly. We develop a simple discrete optimization method described below.

Let the backbone network be modeled as a directed graph $G = (V, E)$. The number of nodes in G is $|V| = n$. E is the set of links in graph G. To formulate a capacity assignment problem, we consider the following two points. First, link delays increase with link utilization rates. Thus, if we can control the maximum link utilization rate of a network, the link delays can also be controlled. Therefore, we set a maximum link utilization rate ρ_{max} for capacity assignment. Second, in a realistic network, especially a backbone network, there are physical limitations on link capacity values, e.g. OC-3 (155 Mbps), OC-12 (622 Mbps), OC-192 (10 Gbps). Thus, it is reasonable to set a minimum capacity $C_{\min}$ for the capacity assignment problem. In our simulation, we choose the minimum capacity of a backbone network link as 155 Mbps (OC-3), and assume that the capacity a link can have are multiples of OC-3. Then we use the following algorithm to determine capacity of each link.

3.3.3 Delay simulation results

The following presents the simulated network delays. As mentioned before, we model the AT&T backbone network by a discrete time fluid network model with finite queuing capacity. The gravity model and fractional Brownian motion (fBm) traffic model presented in Section 3.3.2 is applied to generate aggregate fluid traffic arrivals for all source-destination pairs in the 50-node AT&T network shown in Fig. 3.11. Traffic flows are routed through minimum hop paths [33]. The aggregate traffic input at each link is then the sum of the external fBm traffic input and the cross-traffic from upstream links. In the following subsections, we show the simulated delays in the fluid network with the time resolution of 5 ms.

In simulating the following delays, I set the link utilization rate to be relatively high in order to simulate delay spikes, autocorrelation and cross-covariance that are observed in the real network delay traces in Section 3.2. I have also simulated cases when link utilization rate is very low, for which queuing delays or delay spikes are rarely seen, and thus I would not show them here. In the following, I show the simulation results when all the times zones in USA are in their peak hours [89], i.e. GMT 21:00 or Eastern Time 16:00, and when the western time zone is not yet in its peak hour, i.e. GMT 16:00 or Eastern Time 11:00. By intuition, there are more traffic load in the network at GMT 21:00 than that at GMT 16:00, especially for links connecting the western time zone to the eastern time zone, and thus we can expect to observe higher link delays at GMT 21:00 than those at GMT 16:00, which is validated in the simulation results as follows. Also note that, at other time, e.g. at GMT 10:00 or EST 05:00, it might still be advantageous to use overlay routing from a global point of view since there are some other countries at their peak hours although all the time zones in the USA are at low traffic load and one can still see triangle inequality [82] (note that overlay routing is beneficial when there is triangle inequality), which is left for future work.

```
Input: Traffic arrival $m_l$ at each link $l$.
Initialization: Set all link capacities equal $C_{\min}$=155 Mbps ( OC-3 ) ;
Iteration: for each link $l$
(i) Compute link utilization $\rho_l$, $\rho_l = \frac{m_l}{C_l}$
(ii) If $\rho_l > \rho_{max}$, let $C_l = C_l + 155$ Mbps;
If $C_l > C_{\min}$, $C_l = C_{\min}$
(iii) Stop if no more updates for $C_l, \forall l$;
```

Table 3.2 Algorithm for Capacity Assignment

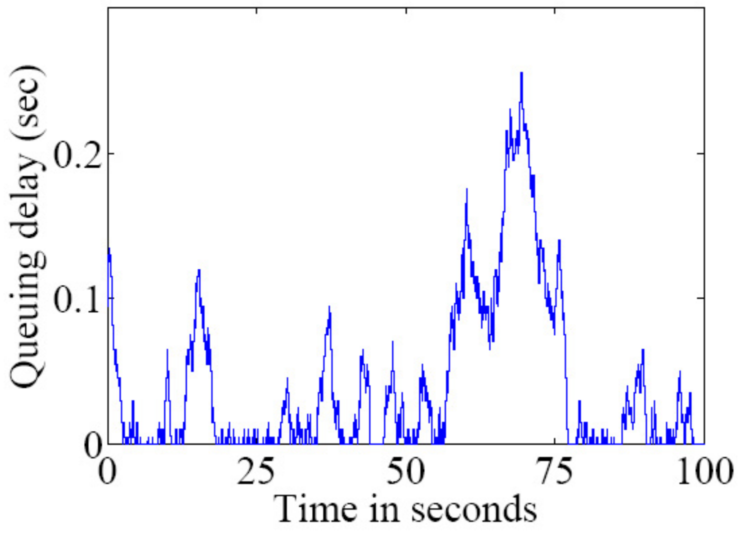


Fig. 3.12 Queuing delay trace on a link (between two major cities located in the Central time zone and the Eastern time zone respectively) at GMT 21:00 (i.e. EST 16:00) when all the time zones in the US are at peak hours.

Queuing delays

We simulate the network at GMT 21:00 (i.e., EST 16:00), as is a time when the traffic in all time zones in the United States is at its peak [89]. Thus, the links are all relatively heavily loaded, comparing to other times. However, as there are a total of 196 directed links, we cannot show all the queuing delays on each of them. Consequently, we show only an example of the link queuing delay trace as in Fig. 3.12 for a relatively heavily loaded link NC, which connects a node N in the eastern time zone to a node C in the central time zone. Among all the simulated queuing delays, we see that some links have relative large queuing delays, as in Fig. 3.12, while some links have zero or very small queuing delays even at peak times. This demonstrates that the shortest hop routing is unable to provide balanced traffic on different links. For VoIP service providers, service overlay networks can be employed to route traffic around heavily loaded links and choose links with low queuing delays.

We also simulated the network at GMT 16:00 (i.e. EST 11:00), to compare with traffic at GMT 21:00. At GMT 16:00, traffic originating from the western time zones is not at its peak. Fig. 3.13 shows the queuing delay trace at GMT 16:00 for the same link between the central time zone and the eastern time zone. Evidently, the queuing delay is very low

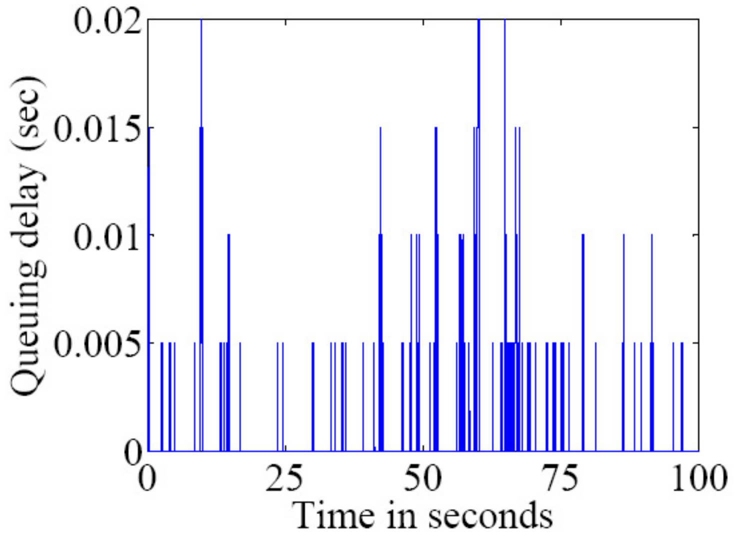


Fig. 3.13 Queuing delay trace on a link (between two major cities located in the Central time zone and the Eastern time zone respectively) at GMT 16:00 (i.e. EST 11:00) when the western time zones in the US are not at its peak hours, demonstrating that the queuing delay is very low because much less traffic transits this link at this time compared to that at GMT 21:00.

because much less traffic transits this link at this time. Therefore, we can see that the quality of a link varies with time and that the overlay network routing scheme should be able to learn and adapt to such changes.

The cross-covariance among link queuing delays is also evident with the fluid network simulation. Fig. 3.15 illustrates the cross-covariance between the queuing delay traces of link CN and NW, (which connects two nodes N and W in the Eastern Time zone, with a queueing delay as in Fig. 3.14) at GMT 21:00. As can be seen, there is a strong cross-covariance between the queuing delay traces at around the lag of 50 seconds, which is due to delay spikes in both queuing delay traces. At GMT 16:00, we can compute the cross-covariance similarly for the two links, and we find that there is no such strong cross-covariance, as in Fig. 3.16. The difference between Fig. 3.15 and Fig. 3.16 is due to the traffic level at different times. When the network traffic level is high, there can be more cross-covariance between the queuing delays on different physical links. A possible reason for such behavior is that both links are shared by some of the shortest hop paths in the network.

As is also observable in Fig. 3.12 to Fig. 3.16, the fluid network model based simulation

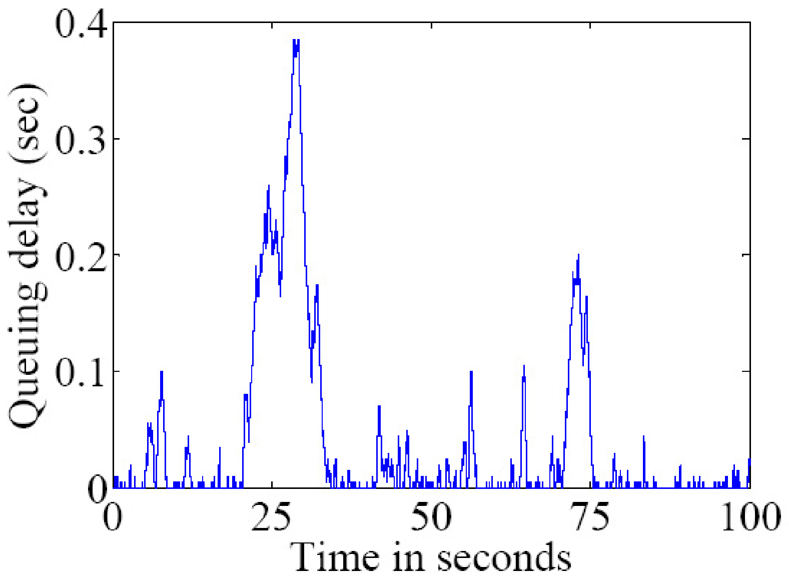


Fig. 3.14 Queuing delay trace on a link (between two major cities located in the Eastern time zones) at GMT 21:00 (i.e. EST 16:00) when all the time zones in the US are at peak hours.

is able to generate realistic network delays.

Warm-up period

In the fluid network model based simulation, it takes time for the queuing network to reach its stationary state from the time traffic is injected into it. The time taken to reach the stationary state is called the warm-up period, which is estimated by simulating the network with different initial states. Fig. 3.17 shows an example for estimating the warm-up period. The bottom graph in Fig. 3.17 shows a link queuing delay trace when the queuing network is initialized with empty queues and links. The bottom graph shows the queuing delay on the same link while the queuing network is initialized with fully loaded queues and links.

We also compare all of the other link queuing delays for the two different network state initializations, and find that the warm-up period can be approximated as 50 seconds for the simulated network. We have determined the duration of the warm-up period based on experimental results. A theoretical derivation of the warm-up period is outside the scope of the thesis and can be explored in future work.

The active probing and learning process, which I present in Chapter 5, is simulated on

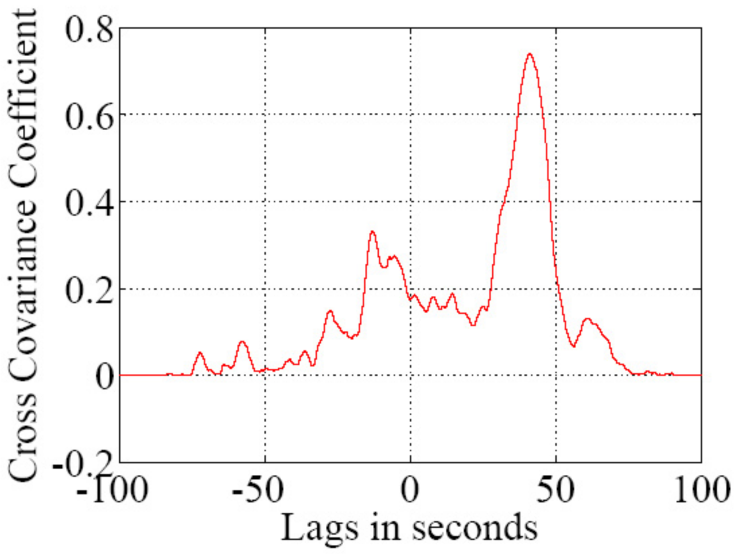


Fig. 3.15 Cross covariance for the queuing delays on the two links CN and NW at GMT 21:00 (i.e. EST 16:00) when all the time zones in USA are at peak hours. There is a strong cross-covariance between the queuing delay traces at around a lag of 50 seconds, which is due to delay spikes in the queuing delay traces on link CN and NW, as shown in Fig. 3.12 and Fig. 3.14, respectively.

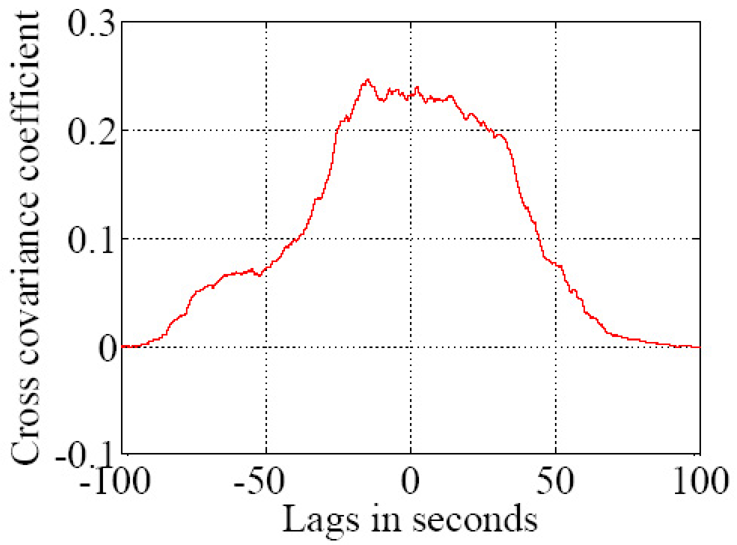


Fig. 3.16 Cross covariance for the queuing delays on two links CN and NW at GMT 16:00 (i.e. EST 11:00), when traffic originating from the western time zones is not at its peak. No strong cross-covariance is observable for the two queuing delay traces.

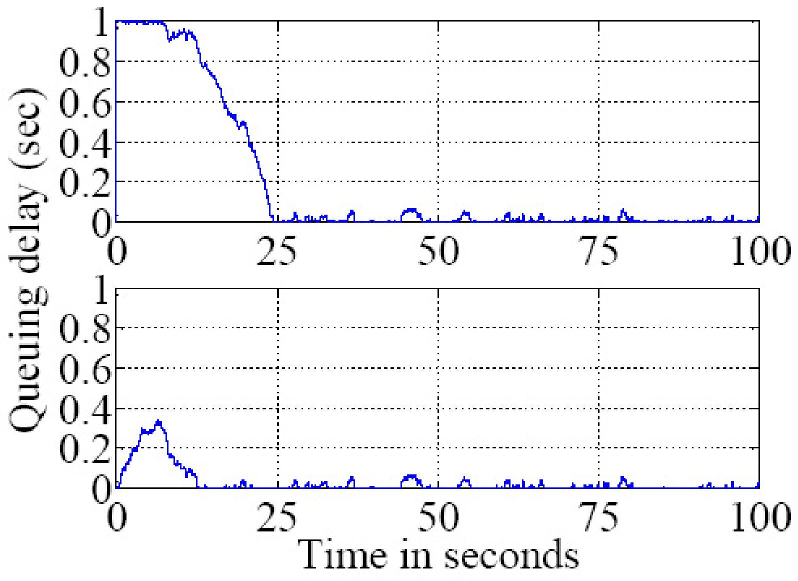


Fig. 3.17 Queuing delay traces for a link during the warmup period with different initial network states. The upper figure illustrates the situation when the queuing network is initialized with fully loaded queues and links; in the bottom figure the queuing network is initialized with empty queues and links. In this example, the warm-up period is around 25 sec.

top of this fluid network model, and is started after the warm-up period.

Round trip time

Assuming unicast traffic and shortest hop routing [33], we simulated the end-to-end delays for all 2450 source-destination pairs of the AT&T topology shown in Fig. 3.11. Fig. 3.18 shows an example of the round trip time for a source-destination pair at GMT 21:00. Similar to the queuing delays experienced at GMT 16:00, the round trip time at GMT 16:00 would also be lower (I have not shown this in order to save space). The trace is very similar to the real delay traces collected by the ISP, that were discussed in Section 3.2. There are also large delay spikes that can be detrimental to VoIP quality.

The autocorrelation of the round-trip-time is plotted in Fig. 3.20. Obviously, the trace in Fig. 3.18 has relatively strong autocorrelation. The small peak at around 20-second lag is due to the 20 seconds gap between two large delay spikes shown in Fig. 3.18.

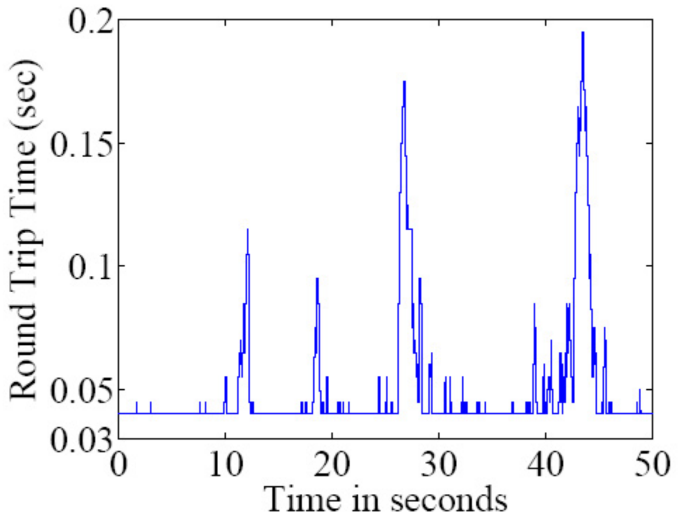


Fig. 3.18 Round Trip Time for a source-destination pair at GMT 21:00, when network traffic level is relatively high (i.e. when all the time zones in USA are at their peak hours).

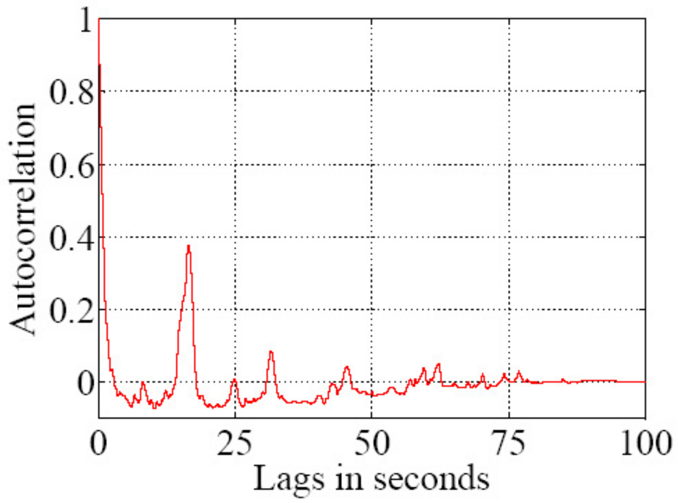


Fig. 3.19 The autocorrelation for the delay trace in Fig. 3.18. It is evident that the trace in Fig. 3.18 has relatively strong autocorrelation. The small peak at around 20-second lag is due to the 20 seconds gap between two large delay spikes in Fig. 3.18.

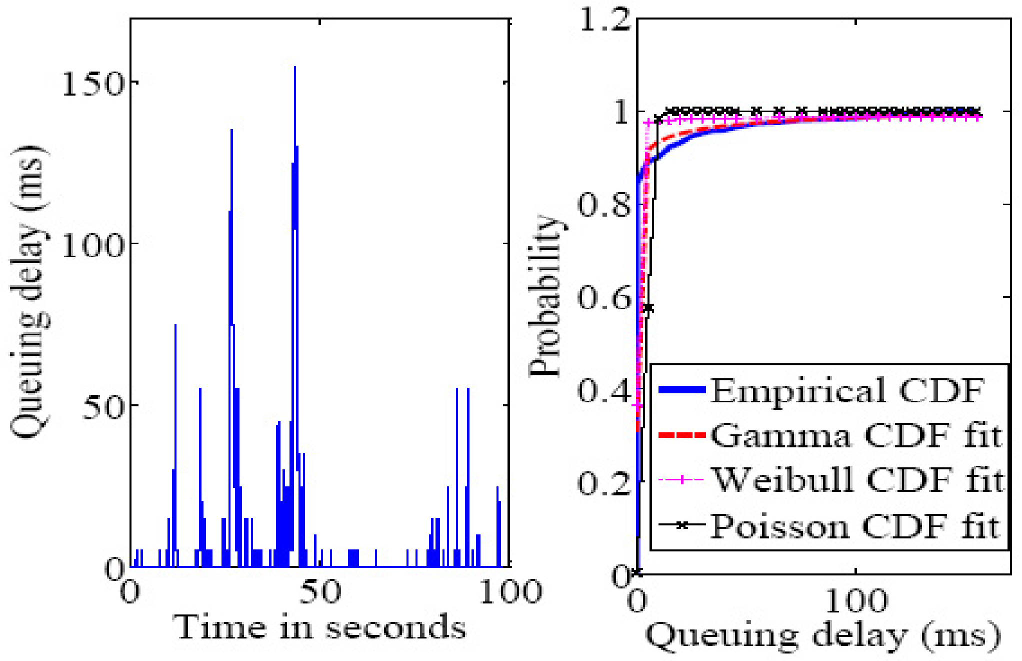


Fig. 3.20 Total queuing delay on an end-to-end path and its distribution fitting to a Gamma distribution.

3.4 Sampling End-to-end Delay Traces

The last two sections presented our work on end-to-end delay trace analysis and simulation. This section focuses on how to sample end-to-end delay traces. The goal of this work is to find a method to determine the sampling rate that resolves the tradeoff between sampling cost and sampling accuracy. As mentioned in Section 1.2.1, it is expensive to collect all the end-to-end delay traces for all the source-destination pairs. Therefore, we want to collect a minimum amount of end-to-end delay samples, while still being able to sufficiently accurately reconstruct the delay trace from the collected minimum amount of delay samples. The question is how to determine the sampling frequency that minimizes sampling overhead while maximizing sampling accuracy.

3.4.1 Introduction

Signal sampling is a fundamental issue for statistical signal processing. Generally speaking, there are two basic sampling approaches: deterministic sampling and random sampling. In the deterministic sampling approach, the sampling intervals are fixed; while in random sampling, as the name implies, the sampling intervals are random. The deterministic sampling method is specifically useful for measuring network impairments for Constant Bit Rate (CBR) applications, e.g. VoIP. For example, probing packets sent with the same bit rate as that of a CBR application can emulate the CBR application, with the result that the end-to-end delay measured by a probing packet equates that experienced by a CBR application packet. The deterministic sampling approach can thus obtain a more accurate measurement of the end-to-end delays for VoIP traffic. However, for statistical signal sampling that involves either random or deterministic sampling, there is always a tradeoff between sampling cost and sampling accuracy. Previous studies have acknowledged the fact that higher sampling accuracy is usually obtained with higher sampling cost [95]. However, it is unclear how this tradeoff can be optimally resolved. This study develops an original approach that suggests an optimal sampling rate that both maximizes sampling accuracy and minimizes sampling cost. The traditional way usually formulates the problem as maximizing sampling accuracy with constraint to a certain sampling cost. However, when we we are not constrained to a sampling cost but we still want to minimize sampling cost and maximize sampling accuracy, we need a different problem formulation, i.e. a bi-objective optimization problem.

3.4.2 Bi-objective Optimization Problem

Denote the network delay trace as $x(t)$. The Fourier transform of $x(t)$ is represented by $X(f)$. Then, $|X(f)|$ is the amplitude spectrum of $x(t)$, and $\Phi(f) = |X(f)|^2$ is the power spectrum of $x(t)$, where $\Phi(f) = \Phi(-f)$ for real valued $x(t)$.

Let the cut-off frequency of $X(f)$ be F_m, $F_m \in (0, \infty)$. Define the signal-to-noise ratio (SNR) as the ratio of the signal energy in $x(t)$ to the noise energy introduced by aliasing when $x(t)$ is sampled at frequency $2f$, where $f < F_m$. Then the signal-to-noise ratio SNR is a function of f as follows.

$$\mathrm{SNR}(f) = \frac{\int_{-F_m}^{F_m} \Phi(u)\mathrm{d}u}{\int_{-F_m}^{-f} 2\Phi(u)\mathrm{d}u + \int_{f}^{F_m} 2\Phi(u)\mathrm{d}u}. \tag{3.10}$$

The sampling cost $\mathrm{c}(f)$ is a monotonous increasing function of the sampling frequency f. It is defined as $\mathrm{c}(f) = f$, for simplicity. Define the sampling error $\mathrm{Er}(f)$ as the noise-to-signal ratio $\frac{1}{\mathrm{SNR}(f)}$ and the sampling accuracy $\mathrm{Ar}(f)$ as $1 - \mathrm{Er}(f)$. Then,

$$\mathrm{Ar}(f) = 1 - \frac{1}{\mathrm{SNR}(f)}. \tag{3.11}$$

As defined above, the sampling accuracy $\mathrm{Ar}(f)$ and the sampling cost $c(f)$ both increase with the sampling frequency $2f$, where $2f \leq 2F_m$. Hence, the maximization of the signal-to-noise ratio $\mathrm{SNR}(f)$ or the sampling accuracy $\mathrm{Ar}(f)$ conflicts with the minimization of the sampling cost $c(f)$. Let the fair sampling frequency that resolves the conflict be $2f^*$ and the optimal sampling accuracy be Ar^*. Thus, the problem of maximizing the sampling accuracy $\mathrm{Ar}(f)$ while minimizing the sampling cost $\mathrm{c}(f)$ can be formulated as a bi-objective optimization problem in (3.12):

$$\begin{aligned} f^* &= \begin{cases} \arg\max_f \ \mathrm{Ar}(f), \\ \arg\min_f \ \mathrm{c}(f), \end{cases} \\ s.t. &\quad 0 < f \leq F_m. \end{aligned} \tag{3.12}$$

Since $0 < f < F_m$, the maximum sampling cost equals $\mathrm{c}(F_m)$. Hence, we can rewrite the bi-objective optimization problem as follows:

$$f^* = \begin{cases} \arg\max_f \ \mathrm{Ar}(f), \\ \arg\max_f \ \max(\mathrm{c}(f)) - \mathrm{c}(f), \end{cases} \quad s.t. \begin{cases} 0 < f \leq F_m \\ \max(\mathrm{c}(f)) = c(F_m) \end{cases} \tag{3.13}$$

3.4.3 Solution to the Bi-objective Optimization Problem

Related work

To solve a multi-objective optimization problem the weighted sum method [96] has been proposed to reduce a multi-objective optimization problem to a single-objective optimization problem, for which a Pareto front (for bi-objective optimization) or a Pareto surface (for multi-objective optimization) can be approximated by changing the weights among the objective functions. The present work proposes a new approach to solving the bi-objective optimization problems.

A proportional fair solution

Solving bi-objective problems is typically done through the provision of a compromise that maintains a balance between two competing interests. The balance can be obtained when measures (such as the value or the rate of change of each competing interest) corresponding to the competing interests are given equal weight. However, in the case where two competing interests are measured with different metrics, it makes more sense for the competing interests to possess the same rate of change at balance instead of possessing the same value. For the problem in (3.12), the sampling accuracy $\mathrm{Ar}(f)$ and the sampling cost $\mathrm{c}(f)$ are different physical quantities. Hence, we seek a balance with the same rate of change. Such a balance can be obtained with a proportional fair solution.

Definition 1 (Proportional fairness [97–99]). *An allocation of rates $\vec{x} = [x_1, ..., x_s, ..., x_N]$ is "proportional fair" if and only if for any other feasible allocation $\vec{y} = [y_1, ..., y_s, ..., y_N]$, $\sum_{s=1}^{N} \frac{y_s - x_s}{x_s} \le 0$.*

The concept of proportional fairness has been applied to solving the network optimal flow control problem in [98]. The present study applies the concept of proportional fairness to solve the bi-objective optimization problem, which provides a sampling frequency f^* that guarantees the same rate of change for $\mathrm{Ar}(f)$ and $\mathrm{c}(f)$ at the balance, i.e. for any other sampling frequency $f \neq f^*$:

$$\frac{\mathrm{Ar}(f) - \mathrm{Ar}(f^*)}{\mathrm{Ar}(f^*)} + \frac{\mathrm{c}(f) - \mathrm{c}(f^*)}{\mathrm{c}(f^*)} \le 0. \tag{3.14}$$

This means the rate of increase in $\mathrm{Ar}(f)$ or $\mathrm{c}(f)$ cannot compensate for the rate of decrease in $\mathrm{c}(f)$ or $\mathrm{Ar}(f)$ at any other sampling frequency f other than f^*. In [99], it has been proven that there exists one unique proportional fair allocation achieved by maximizing $\sum_{s=1}^{N} ln(x_s)$ over the set of feasible allocations. Therefore, we obtain the following proportional fair solution to the bi-objective optimization problem.

Lemma 3.4.1. *The proportional fair optimal solution f^* to the bi-objective optimization problem in (3.12) maximizes the product of the two objectives in (3.13).*

Proof. Paper [99] shows that a proportional fair allocation can be obtained by maximizing $\sum_s \ln(x_s)$ over the set of feasible allocations x_s, where x_s is the share allocated to user s.

Therefore, following the approach of [99], a proportional fair solution to the problem (3.13) or (3.12) can be given by:

$$\begin{aligned} \max_f \quad & \ln(v_1) + \ln(v_2) \\ v_1 \quad & = \mathrm{Ar}(f) \\ v_2 \quad & = \max(\mathrm{c}(f)) - \mathrm{c}(f) \\ s.t. \quad & 0 < f \leq F_m \\ & \max(\mathrm{c}(f)) = c(F_m) \end{aligned} \tag{3.15}$$

□

In fact, if we formulate the bi-objective optimization problem as a bargaining problem between two players, with the set of payoffs under agreement being the points interior to and on the boundary of Ar(f) and the payoff under disagreement being (0,0), we can obtain a Nash bargaining solution [100] that is the same as that for (3.15).

The solution to the bi-objective optimization problem also possesses the properties of a Nash Bargaining solution [100], i.e. 1. Pareto efficiency[11], 2. invariant to affine transformations or invariant to equivalent utility representations, 3. independence of irrelevant alternatives[12], 4. symmetry[13]. By proving the following lemmas, the equivalence between the proportional fair solution to the bi-objective optimization problem and the Nash Bargaining solution is established. From the definition of Ar(f) and c(f), it is easy to see that the property 3, independence of irrelevant alternatives, is satisfied. The symmetry is also guaranteed if the two objectives are indistinguishable. Consequently, we only prove properties 1 and 2.

Lemma 3.4.2. *The optimal solution f^* to the bi-objective optimization problem (3.12) is a Pareto efficient solution.*

[11]Pareto efficient is an important concept in economics with broad application in game theory, engineering and social science [101, 102]. A situation is Pareto efficient if any change to make one person better off (i.e. put in a preferred position) would cause someone else to be worse off [100–102]. Any Pareto inefficient system can be improved to Pareto efficient by making some person better off without causing others to be worse off.

[12]If A is preferred to B out of the choice set {A,B}, then introducing a third alternative X, thus expanding the choice set to {A,B,X}, must not make B preferable to A [100, 103].

[13]If two players are indistinguishable, they should achieve the same utility at the NBS [100].

Proof. Supposing the sampling cost function $\mathrm{c}(f)$ and the sampling accuracy function $\mathrm{Ar}(f)$ are differentiable, we can write $\mathrm{c}'(f^*) = \lim_{\Delta f \to 0} \frac{\mathrm{c}(f^*+\Delta f)-\mathrm{c}(f^*)}{\Delta f}$.
Similarly, $\mathrm{Ar}'(f^*) = \lim_{\Delta f \to 0} \frac{\mathrm{Ar}(f^*+\Delta f)-\mathrm{Ar}(f^*)}{\Delta f}$. It is easy to see that $\mathrm{Ar}'(f) \geq 0$ and $\mathrm{c}'(f) \geq 0$. Now consider two situations:
(a) If we increase f^* by Δf, $\Delta f > 0$, and $0 < f^* + \Delta f \leq F_m$, then $\mathrm{c}(f^* + \Delta f) > \mathrm{c}(f^*)$ and $\mathrm{Ar}(f^* + \Delta f) > \mathrm{Ar}(f^*)$, i.e. $\mathrm{c}(f)$ becomes worse off if we make $\mathrm{Ar}(f)$ better off by increasing the sampling frequency.
(b) Similar to (a), if we decrease f^* by Δf, $\Delta f > 0$, and $0 < f^* - \Delta f \leq F_m$, then $\mathrm{c}(f^* - \Delta f) < \mathrm{c}(f^*)$ and $\mathrm{Ar}(f^* - \Delta f) < \mathrm{Ar}(f^*)$, i.e. $\mathrm{Ar}(f)$ becomes worse off if we make $\mathrm{c}(f)$ better off by decreasing the sampling frequency.

In sum, no single objective can be better off without making the other objective worse off, i.e. the optimal solution f^* is a Pareto efficient solution. □

Lemma 3.4.3. *The optimal solution f^* to the bi-objective optimization problem in (3.12) is invariant to the linear equivalent representation of* $\mathrm{Ar}(f)$ *and* $\mathrm{c}(f)$.

Proof. Let $\tilde{\mathrm{Ar}}(f)$ be $\alpha_1 \mathrm{Ar}(f) + \beta_1$ and $\tilde{\mathrm{c}}(f)$ be $\alpha_2 \mathrm{c}(f) + \beta_2$. The transformation maintains the order of preference in $\mathrm{Ar}(f)$ and that in $\mathrm{c}(f)$, i.e. $\alpha_1 > 0, \alpha_2 > 0$. The original bi-objective optimization problem in (3.12) becomes:

$$\begin{cases} \max_f & \tilde{\mathrm{Ar}}(f) \textit{ or } \alpha_1 \mathrm{Ar}(f) + \beta_1, \\ \min_f & \tilde{\mathrm{c}}(f) \textit{ or } \alpha_2 \mathrm{c}(f) + \beta_2, \end{cases} \quad s.t.\ 0 < f \leq F_m. \tag{3.16}$$

Let a Pareto efficient solution to the problem in (3.12) be f^*. Then it is easy to see that $\tilde{f}^* = f^*$ is also a Pareto efficient solution to problem (3.16) since the order of preference in $\tilde{\mathrm{Ar}}(f)$ and $\tilde{\mathrm{c}}(f)$ is the same as that in $\mathrm{Ar}(f)$ and $\mathrm{c}(f)$. □

3.4.4 Experimental results

We generated a stationary network delay trace $x(n)$ with the fBm traffic, as shown on the top left graph of Fig. 3.22. The parameters of the fBm traffic are from the Bellcore data [80], where Hurst parameter is H=0.86, the mean input rate is m=12.3 kbit/sec, and the variance coefficient is a=68.6 kbit· sec. Then we apply the optimal sampling method proposed in Section 3.4.3 to sample the network delay trace $x(n)$. The sampling accuracy $\mathrm{Ar}(f) = 1 - \frac{1}{\mathrm{SNR}(f)}$ is a concave function, as shown in Fig. 3.21.

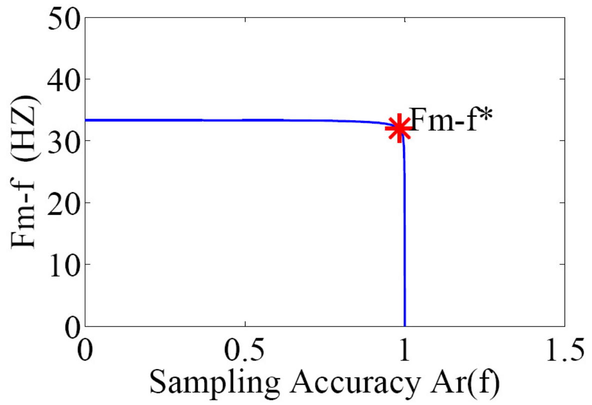


Fig. 3.21 Trade-off between sampling frequency and sampling accuracy. In this example, sampling accuracy Ar(f) is plotted on the x-axis, the cut-off frequency $F_m = 33.3$Hz. The sampling cost is $c(f) = f$. The y-axis is the residual sampling cost $F_m - f$. f^* is the fair sampling frequency. The star point represents the optimal residual sampling cost $F_m - f^*$.

The optimal solution to the bi-objective optimization problem in (3.13) is the point on the curve in Fig. 3.21 which maximizes the product of the two coordinates. The resulting proportional fair sampling frequency given by the solution to (3.15) is $f^* = 1.3$HZ. This reduces the original sampling frequency ($F_m = 33.3$HZ) by about 30 times. Signal $y(k)$, which is sampled with frequency f^* from $x(n)$, and signal $z(n)$, which is reconstructed from the sampled signal $y(k)$, are also shown on the top right and bottom left of Fig. 3.22, respectively. The noise in the reconstructed signal, i.e. $x(n) - z(n)$ is also plotted on the bottom right of Fig. 3.22. The corresponding signal-to-noise ratio SNR for the reconstructed signal is about 20dB.

In sum, we have resolved the trade-off between the sampling accuracy and the sampling cost by solving a bi-objective optimization problem. We show that the proportional fair and Pareto efficient solution to the bi-objective optimization problem is given by the point that maximizes the product of the objectives. This proportional fair solution is equivalent to a Nash bargaining solution. It provides relatively accurate estimation and reconstruction of the original delay trace at low sampling cost, and it also shows that significant savings in monitoring costs (as quantified by the number of samples) are possible. The proposed method gives a fair sampling rate that addresses the fairness of the tradeoff [104]. The end-to-end delay sampling method considered here is deterministic in methodology, but the approach to tradeoff resolution can be extended to many other sampling methods.

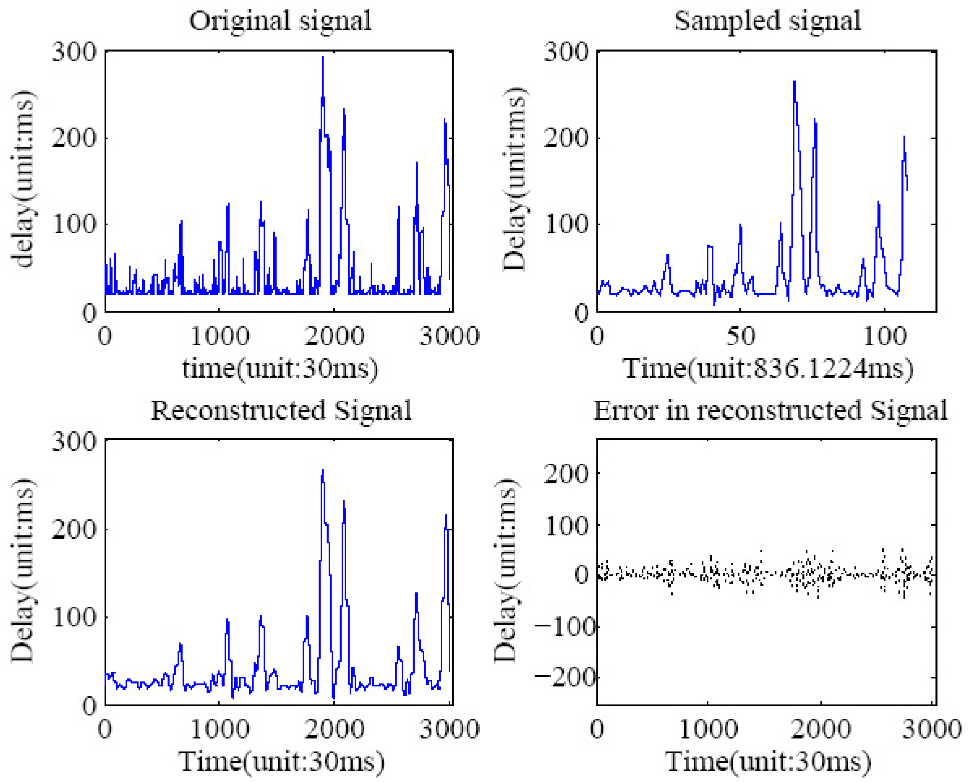


Fig. 3.22 Sample network delay with the proportional fair sampling frequency. The top left figure is the original delay trace $x(n)$, which is sampled every 30 ms. The top right figure shows the delay trace $y(k)$ after downsampling $x(n)$ at a ratio of $\frac{30}{763}$ (the ratio is decided by the proposed optimal sampling method). The bottom left figure is the delay trace $z(n)$ reconstructed from $y(k)$ by upsampling $y(k)$ at a ratio of $\frac{763}{30}$. As can be seen, the delay trace on the top left figure and that on the bottom left figure looks very similar. The noise in the reconstructed signal $z(n)$, i.e. $x(n) - z(n)$ is plotted on the bottom right. The corresponding signal-to-noise ratio SNR for the reconstructed signal is about 20 dB.

3.5 Summary

End-to-end delay is one of the most important performance characteristics of the Internet. It can affect the quality of a VoIP call significantly. In this chapter, I have presented new contributions on end-to-end delay analysis, simulation and sampling.

In the end-to-end delay analysis section, I analyzed the relation between the sample mean and the sample standard deviation of end-to-end delay traces, the relation between the minimum delay and the geographical distance between two nodes, the marginal probability distribution and the autocorrelation of end-to-end delay traces. A shifted gamma distribution model is then fitted to the real network delay measurements. Based on the analysis, delay traces with random delay spikes and no/weak autocorrelation can be synthesized.

To simulate more realistic network delays, where strong auto-correlation and cross-correlation among delay traces are also maintained, we simulate a 50-node model of the AT&T backbone network, with the fluid network model implemented in MATLAB, a gravity model for traffic demand, a fractional Brownian motion process for traffic dynamics, and well-designed link capacities. Simulation results show that the simulated end-to-end delays are within a reasonable range. Autocorrelation and cross-correlation among delay traces are also observed in the simulated network delays. Such a carefully designed network model and simulated delays will be used for distributed QoS routing schemes presented in Chapters 5 and 6.

Measuring end-to-end delays requires sending probing packets to the network, which adds additional overhead. Storing these measurements also puts pressure on the storage system. To find a proportional fair sampling frequency that minimizes sampling cost (and thus saves storage) and also maximizes sampling accuracy, we formulated a bi-objective optimization problem with two conflicting objectives: sampling cost minimization and sampling accuracy maximization. The tradeoff is resolved in a solution that provides the same rate of change for both objectives. This is a proportional fair solution that maximizes the product of the two objectives, which is the same as the Nash Bargaining solution if we formulate the two objectives as two selfish players. Finally, this method for resolving the trade-off between conflicting interests can also be extended to solve other multi-objective optimization problems.

Chapter 4

Improving R-factor with Diverse Routing: A Centralized Approach

4.1 Introduction

Following the end-to-end delay study in the last chapter, I conducted preliminary research on diverse routing in small scale overlay networks. This chapter presents a centralized solution for the diverse routing problem based on R-factor estimation.

In general, the network layer of the Internet only provides a single route (e.g., shortest hop path) between a given source-destination pair. This route is calculated to minimize criteria specified by the ISP(s) owning the networking infrastructure between the source and destination, and their criteria do not necessarily attempt to minimize the latency experienced by packets traveling from source to destination. For latency-sensitive applications, such as voice-over-IP or other streaming multimedia, the "given route" provided by the network layer may not be adequate, and it may be possible to improve the quality of service by routing the traffic along an alternate route through the nodes of a service overlay network. However, this requires identifying the optimal route through the overlay network.

As mentioned in Section 2.3, previous studies [12, 14, 15, 47, 48] have demonstrated the benefit of path diversity in overlay networks — using two disjoint paths between the origin and destination improves the perceived end-user performance when one path temporarily experiences congestion, as illustrated in Fig. 4.1. In practice, a pair of paths has been often used to provide a good tradeoff between performance improvement and cost of transmitting

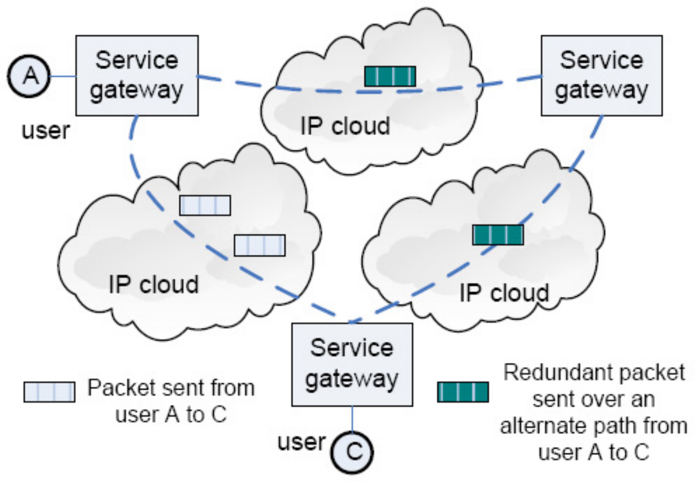

A
user
Service gateway
IP cloud
Service gateway
IP cloud
IP cloud
Service gateway
Packet sent from user A to C
Redundant packet sent over an alternate path from user A to C
user
C

Fig. 4.1 Diverse routing in service overlay networks.

on two paths [12, 48, 49]. However, the pair of paths is usually selected arbitrarily. In this chapter, we develop a method to choose the best pair of paths for VoIP, by selecting that with the best R-factor to route VoIP packets to their destination.

As discussed in Section 3.2, we have obtained end-to-end delay traces measured between gateways of an Internet Service Provider (ISP). The gateways are located in the USA, China, Korea, Japan, Singapore and Malaysia. By considering these gateways of the ISP as overlay nodes of a full mesh overlay network, I conducted a preliminary research on how VoIP can benefit from routing on a pair of paths. In this chapter, we propose a novel mechanism to select the optimal pair of paths that provides the best R-factor estimate for VoIP phone calls when adaptive play-out scheduling is applied at the receiver. The paths considered have a maximum of two hops due to the computational complexity of estimating R-factor on multi-hop paths. This method also proposes a framework for communicating network performance characteristics, where pre-processing of network measurements and a data fusion center are involved. In the data fusion center, the optimal pair of paths is determined based on the pre-processed overlay link performance characteristics. The proposed mechanism is evaluated in the small-scale overlay network of the ISP referred to before by comparing the R-factor of the VoIP calls sent through the optimal pair of paths with that of the VoIP calls sent through the direct path[1]. The simulation results show that

[1]The direct path is the shortest hop path decided by the underlying network

the proposed method can improve VoIP quality and provide much more stable quality for VoIP calls.

4.1.1 Chapter Structure

The rest of the chapter is organized as follows: Section 4.2 presents the routing problem to be solved, the proposed solution and the simulation results. Section 4.3 summarizes the chapter.

4.2 Selecting the Optimal Pair of Paths for VoIP

4.2.1 Problem statement

For a centralized routing problem, it is practical to consider only overlay paths that are comprised of either a one-hop overlay link, which can be called a direct path, or a two-hop overlay path, which is a concatenated path. In this study, a pair of overlay paths is used to route voice calls from a source to a destination. We also assume that the overlay links of a pair of overlay paths are independent. This can be assumed for a well designed overlay network whose overlay links are disjoint in the physical layer, e.g. when each overlay node is located in a different country and in a different Internet Service Provider.

Let $\hat{d}$ and $\hat{l}$ represent the estimate of the total end-to-end delay d and loss l, respectively. Then, the estimate of the R-factor is:

$$\hat{\mathrm{R}}(\hat{d}, \hat{l}) = 93.2 - \mathrm{I}_d(\hat{d}) - \mathrm{I}_e(\hat{l}). \tag{4.1}$$

Denote the set of the candidate overlay paths for a source-destination pair as S. This includes all the concatenated paths and the direct path between the source-destination pair. Thus, the set of all the possible path pairs between the source-destination pair is $S \times S$. Here, a pair of paths is used interchangeably with diverse paths. Denote a pair of diverse path as $[s; t]$, where $s, t \in S$ and $s \neq t$. Note that $[t; s]$ is the same as $[s; t]$. The optimal diverse path selection problem can be written as:

$$(s^*, t^*) = \arg \max_{(s,t) \in S \times S, s \neq t} \hat{\mathrm{R}}(\hat{d}_{[s;t]}, \hat{l}_{[s;t]}) \tag{4.2}$$

where $\hat{d}_{[s;t]}$ and $\hat{l}_{[s;t]}$ are the estimate of the end-to-end delay and loss, respectively, for the pair of paths s and t, $s, t \in S$. s^* and t^* represent the selected optimal pair of paths.

4.2.2 Solution: Optimal path pair selection for VoIP

To solve the problem in (4.2), we propose a novel diverse routing mechanism to select the optimal multipath with the best R-factor estimate. The advantage of the proposed method is that the selected optimal path not only maximizes the R-factor but also automatically guarantees a more stable quality. This is because the proposed end-to-end delay estimation method accounts for network delay variations by cooperating with the adaptive playback scheduling at the receiver.

There are two challenges in estimating R-factors on diverse paths and choosing the best pair of paths accordingly: estimation of the end-to-end delay and loss on a pair of paths, and the communication of the network performance characteristics for optimal diverse path decisions.

I present the details of the challenges and propose solutions below.

Estimation of the end-to-end delay and loss on a pair of paths

The first challenge in solving the optimization problem is the estimation of the end-to-end delay $\hat{d}$ on a pair of paths. The end-to-end delay d of a voice call includes not only the network delay d_{net}, which can be measured by sending active probes between a source-destination pair, but also the playback delay d_{play} at the receiver of the voice call. The probing overhead for measuring network delays is the same as that for RON as discussed in Section 2.3.

As mentioned in Section 2.2.3, adaptive play-out scheduling performs better than fixed play-out scheduling in a dynamic network environment, especially when network delay varies considerably.

- Adaptive play-out scheduling
  In this work, we consider adaptive playback scheduling [16, 105] at the receiver. The empirical Complementary Cumulative Distribution Function (CCDF) $\mathrm{H}^r_{d_{net}}(x)$ for network delay is stored for the last N received voice packets. The play-out delay for the next packet is decided according to $\mathrm{H}^r_{d_{net}}(x)$. With this approach, if the acceptable play-out loss rate is l_{play}, the receiver automatically adjusts the play-out

delay as $d_{play} = \mathrm{H}^{r(-1)}_{d_{net}}(l_{play}) - d_{net}$ given that the next packet received has a network delay of d_{net} and that the inverse of the function $\mathrm{H}^{r}_{d_{net}}$ is $\mathrm{H}^{r(-1)}_{d_{net}}$. It can be seen that $\mathrm{E}(d_{play}) = \mathrm{H}^{r(-1)}_{d_{net}}(l_{play}) - \mathrm{E}(d_{net})$ if the network delay is stationary with CCDF $\mathrm{H}^{r}_{d_{net}}(x)$ and the mean network delay is $\mathrm{E}(d_{net})$. Suppose that the CCDF $\mathrm{H}_{d_{net}}(x)$ for the network delay d_{net} measured by the sender and the CCDF $\mathrm{H}^{r}_{d_{net}}(x)$ measured at the receiver are the same and slowly varying, then the sender is able to estimate the play-out delay d_{play} at the receiver with $\hat{d}_{play}$ based on $\mathrm{H}_{d_{net}}(x)$.

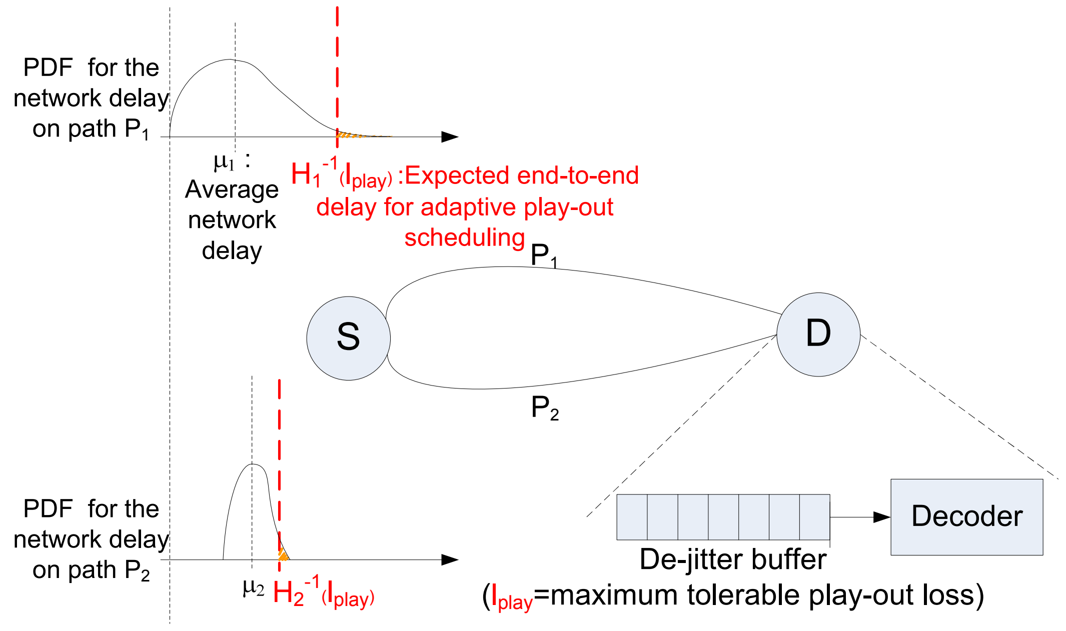

PDF for the network delay on path P1
μ1 : Average network delay
H1^-1(lplay) :Expected end-to-end delay for adaptive play-out scheduling
P1
S
D
P2
PDF for the network delay on path P2
μ2
H2^-1(lplay)
Decoder
De-jitter buffer
(lplay=maximum tolerable play-out loss)

Fig. 4.2 Illustration of the expected end-to-end delays when adaptive play-out scheduling is adopted at the receiver. Given a tolerable play-out loss l_{play} at the receiver and the Probability Density Function (PDF) for the network delay on path P_1 or its Complementary Cumulative Distribution Function (CCDF) $H_1(x)$, the expected end-to-end delay for path P_1 is $H_1^{-1}(l_{play})$. Similarly, the expected end-to-end delay for path P_2 is $H_2^{-1}(l_{play})$. Hence, path P_2 is better than path P_1, although the average network delay on path P_1 is less than that on path P_2.

It is important to include $\hat{d}_{play}$ in the end-to-end delay estimate $\hat{d}$ for the following reasons.

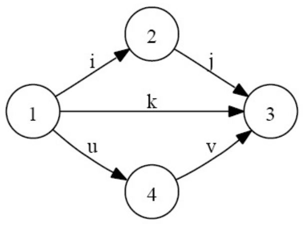


Fig. 4.3 Illustration of the direct and concatenated paths between nodes 1 and 3

– As illustrated in Fig. 4.2, for two paths with the same mean network delay $\mathrm{E}(d_{net})$, the path with heavier tail distribution, i.e. with larger $\mathrm{H}^{r(-1)}_{d_{net}}(l_{play})$, will have a larger expected play-out delay, since $\mathrm{E}(d_{play}) = \mathrm{H}^{r(-1)}_{d_{net}}(l_{play}) - \mathrm{E}(d_{net})$, and thus a longer expected end-to-end delay. Therefore, the heavy tail of a network delay distribution indicates a large delay variation on a path, which should be taken into consideration when the routing decision is made.

– The computation of R-factor requires knowledge of the total end-to-end delay and loss, which by definition include the play-out delay and loss.

Let $\hat{d}_{play}(l_{play})$ represent the estimate of the play-out delay d_{play} for a certain tolerable play-out loss l_{play} at the receiver. Then, $\hat{d}_{play}(l_{play}) = \mathrm{H}^{-1}_{d_{net}}(l_{play}) - d_{net}$. The estimated end-to-end delay $\hat{d}$ on the path is:

$$\hat{d} = d_{net} + \hat{d}_{play}(l_{play}) = \mathrm{H}^{-1}_{d_{net}}(l_{play}) \tag{4.3}$$

In order to estimate $\hat{d}$, we must first estimate the CCDF $\mathrm{H}_{d_{net}}(x)$ of the network delay for each pair of diverse paths, which is discussed below.

- Estimation of network delay and loss on a concatenated path.

  To estimate $\mathrm{H}_{d_{net}}(x)$ for a pair of diverse paths, we must first consider estimating $\mathrm{H}_{d_{net}}(x)$ for each concatenated path, (i.e. two-hop overlay path). Let $d^{[j]}_{net}$ and $l^{[j]}_{net}$ represent the network delay and loss on an overlay link j.

  Given the network delay distribution of each overlay link, we can find the network delay distribution on a concatenated overlay path. As shown in Fig. 4.3, link i represents an overlay link between overlay node 1 and 2. For the purposes of representation we denote the concatenated path formed by overlay link i and j as $[i, j]$. Let $\mathrm{F}_{d^{[i]}_{net}}(x)$ and $\mathrm{F}_{d^{[j]}_{net}}(x)$ represent the Cumulative Distribution Function (CDF) of the network delays on overlay link i and j. Then the CCDF $\mathrm{H}_{d_{net}^{[i,j]}}(x)$ for the network delay on a concatenated path $[i,j]$ can be computed from $\mathrm{F}_{d_{net}^{[i]}}(x)$ and $\mathrm{F}_{d_{net}^{[j]}}(x)$ by (4.4), assuming the independence of $d_{net}^{[i]}$ and $d_{net}^{[j]}$, where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier transform and the inverse Fourier transform, respectively. Similarly, the network loss $l_{net}^{[i,j]}$ on the concatenated path $[i,j]$ can also be computed from $l_{net}^{[i]}$ and $l_{net}^{[j]}$ as in (4.4).

$$\begin{aligned} d_{net}^{[i,j]} &= d_{net}^{[i]} + d_{net}^{[j]} \\ \mathrm{H}_{d_{net}^{[i,j]}}(x) &= \Pr\{d_{net}^{[i,j]} \geq x\} \\ &= 1 - \mathcal{F}^{-1}(\mathcal{F}(\mathrm{F}_{d_{net}^{[i]}}(x)) \cdot \mathcal{F}(\mathrm{F}_{d_{net}^{[j]}}(x))) \\ l_{net}^{[i,j]} &= 1 - (1 - l_{net}^{[i]}) \cdot (1 - l_{net}^{[j]}) \end{aligned} \tag{4.4}$$

- Estimation of the network delay and loss for a pair of paths.

  For purposes of representation, we denote a pair of paths that consists of a pair of diverse paths s and t as $[s;t]$, where s and t are two candidate paths for a source-destination pair. If a voice packet is received from both paths, the one that arrives later is discarded. Thus, the network delay for a voice packet transmitted on the pair of paths $[s;t]$ is the minimum of network delays on paths s and t. The network loss on the pair of paths is the product of losses on two paths as shown in (4.5):

$$\begin{aligned} d_{net}^{[s;t]} &= \min(d_{net}^{[s]}, d_{net}^{[t]})(1 - l_{net}^{[t]})(1 - l_{net}^{[s]}) + (1 - l_{net}^{[t]})l_{net}^{[s]}d_{net}^{[t]} + (1 - l_{net}^{[s]})l_{net}^{[t]}d_{net}^{[s]}; \\ l_{net}^{[s;t]} &= l_{net}^{[s]} \cdot l_{net}^{[t]} \ . \end{aligned} \tag{4.5}$$

When the path loss rates $l_{net}^{[t]}$ and $l_{net}^{[s]}$ are small, we approximate $d_{net}^{[s;t]}$ as $\min(d_{net}^{[s]}, d_{net}^{[t]})$. Let the CCDF of the network delay on the pair of paths $[s;t]$ be $\mathrm{H}_{d_{net}^{[s;t]}}(x)$.

$$\begin{aligned} \mathrm{H}_{d_{net}^{[s;t]}}(x) &= \Pr\{d_{net}^{[s;t]} \geq x\} \\ &\approx \Pr\{\min(d_{net}^{[s]}, d_{net}^{[t]}) \geq x\} \\ &\triangleq \mathrm{h}_{d_{net}^{[s;t]}}(x) \\ &= (1 - \mathrm{F}_{d_{net}^{[s]}}(x)) \cdot (1 - \mathrm{F}_{d_{net}^{[t]}}(x)) \end{aligned} \tag{4.6}$$

- Estimation of the end-to-end delay and loss on a pair of paths.

  Using (4.4)–(4.6), the estimated end-to end delay $\hat{d}_{[s;t]}$ on the pair of paths $[s;t]$ can be estimated according to (4.3), as shown in (4.7), where l_{play} is the pre-set tolerable play-out loss rate at the receiver, e.g. $l_{play} = 0.01$. The end-to-end loss on the pair of paths $[s;t]$ is the sum of the network loss and the play-out loss as given in (4.8):

$$\hat{d}_{[s;t]} \approx \mathrm{h}^{-1}_{d^{[s;t]}_{net}}(l_{play}); \tag{4.7}$$

$$\hat{\ell}_{[s;t]} = l^{[s]}_{net} \cdot l^{[t]}_{net} + (1 - l^{[s]}_{net} \cdot l^{[t]}_{net}) l_{play}. \tag{4.8}$$

Communication of network performance characteristics

In the service overlay network, each overlay node measures only the network delays and losses from itself to all the other overlay nodes. Therefore, in order to estimate the end-to-end delays and losses on the concatenated overlay paths, the delay distributions on other overlay links has to be communicated efficiently. However, the communication cost would be high when delay distributions are non-parametric.

We propose a method whereby feature vectors of the network performance characteristics are sent to a data fusion center, as illustrated in Fig. 4.4, where R-factors on diverse paths are estimated and the optimal diverse paths are selected. The optimal diverse path selection algorithm is given in Algorithms 1 and 2.

```
Require: Service overlay network G=(V,E), at each overlay node $i \in V$
1: Perform active probing from $i$ to all $j$, $j \in V$, per $\tau$ time interval;
2: Get N consecutive network delay measurements $T^N_{ij} = \{d^1_{net_{ij}}, ..., d^k_{net_{ij}}, ..., d^N_{net_{ij}}\}$
   between node $i$ and $j$;
3: Fit a shifted Gamma distribution with parameters $(\mu_{ij}, \alpha_{ij}, \beta_{ij})$ to $T^N_{ij}$ using the
   maximum likelihood method;
4: Compute the loss rate $l_{ij}$ from $T^N_{ij}$;
5: Send the feature vector $[\mu_{ij}, \alpha_{ij}, \beta_{ij}, l_{ij}]$ to the data fusion center.
```

Algorithm 1: Operation at each overlay node

- Feature vector for network performance characteristics

  We have shown the formula for computing the network delay distribution on concatenated paths in (4.4). For a non-parametric representation of the distributions $\mathrm{F}_{d^{[i]}_{net}}(x)$

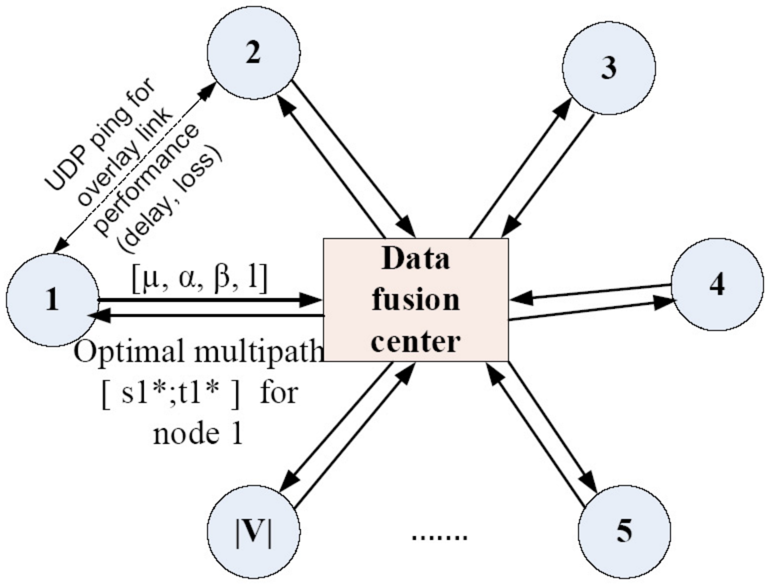

2
3
UDP ping for overlay link performance (delay, loss)
1
[μ, α, β, l]
Data fusion center
4
Optimal multipath [s1*;t1*] for node 1
|V|
....
5

Fig. 4.4 Illustration of the centralized data fusion based diverse routing method. Each overlay node collects and pre-processes network performance measurements, then sends the pre-processed network performance feature vectors $[\mu, \alpha, \beta, l]$ to the data fusion center for making optimal diverse routing decisions.

```
Require: Service overlay network G=(V,E), feature vectors $[\mu_{ij}, \alpha_{ij}, \beta_{ij}, l_{ij}]$ for all $i, j \in V$
1: Estimate end-to-end delay and loss for all candidate paths with (4.3)–(4.6);
2: Find the optimal pair of paths that solves the optimization problem in (4.2) by brute-force search for each source-destination pair;
3: Send the selected optimal pair of paths $[s^*; t^*]$ for each source-destination pair to the corresponding source overlay node.
```

Algorithm 2: Operation at the data fusion center

and $\mathrm{F}_{d_{net}^{[j]}}(x)$, it is difficult to evaluate $\mathrm{H}_{d_{net}^{[i,j]}}(x)$ from the convolution of $\mathrm{F}_{d_{net}^{[i]}}(x)$ and $\mathrm{F}_{d_{net}^{[j]}}(x)$, and thus to evaluate the delay distributions on concatenated paths and those on a pair of paths.

As referred to in Section 3.2.2, network delay distributions are slowly varying and network delay distributions on different links at different times can be fitted to a shifted Gamma distribution [**?**, 106]. Therefore, we use the parameters of a shifted gamma distribution to represent the network delay distribution of an overlay link. The feature vector for the network delay and loss l on an overlay link i can be written as $[\mu_i, \alpha_i, \beta_i, l_i]$.

- Optimal diverse path selection in data fusion center

  Once the data fusion center receives feature vectors of network performance on all the overlay links, the parameters of the delay distributions on all concatenated paths can be approximated. For the shifted Gamma distributed network delays d_{net_i} and d_{net_j} with parameters $(\mu_i, \alpha_i, \beta_i)$ and $(\mu_j, \alpha_j, \beta_j)$, respectively, paper [107] stated that the distribution of the sum of two Gamma distributed random variables can be approximated as a Gamma distribution. Therefore, a shifted Gamma distribution can be used to approximate the distribution for the network delay $d_{net}^{[i,j]}$ on the concatenated path $[i, j]$. with parameters (μ, α, β) as follows [107]:

$$\begin{aligned} \mu &= \mu_i + \mu_j, \\ \alpha &= \alpha_i + \alpha_j, \\ \beta &= \frac{\alpha_i \cdot \beta_i + \alpha_j \cdot \beta_j}{\alpha_i + \alpha_j}, \end{aligned} \tag{4.9}$$

The end-to-end delay and loss can be estimated for each possible path pairs between a source-destination pair from (4.3)–(4.6). The R-factor can then be evaluated according to (4.1). The optimization problem given in (4.2) can be solved by brute-force search where the entire set of possible path pairs is searched and compared. The selected optimal pair of paths will then be sent to the corresponding source nodes.

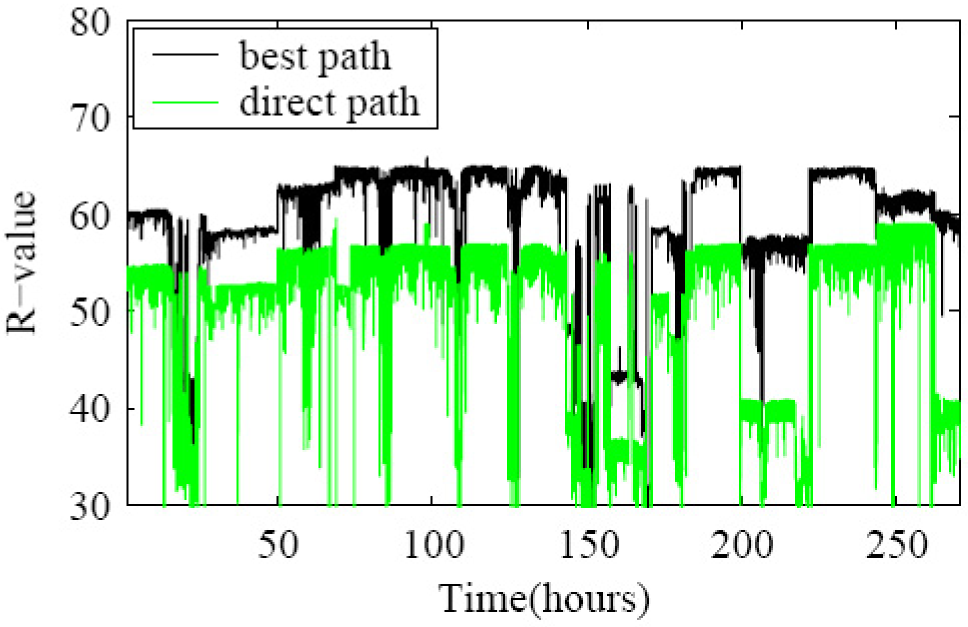


Fig. 4.5 R-factor for the best R-path (top) versus that for the direct path (bottom) between two service gateways over the whole measurement period (277.5 hours). G.723 codec is assumed for R-factor computing.

4.2.3 Simulation

A preliminary simulation on diverse path routing

With measured data in an implemented SON with 7 service gateways and 42 overlay links, we simulate the quality of a VoIP call encoded with the G.723.1-A codec [3, 108]. The adaptive play-out delay at the receiver is set to correspond to 1% play-out loss. Then we calculate the R-factor for all different paths between two service gateways. Fig. 4.5 shows the R-factor results for the direct path and the best R-factor among all paths. As shown in the figure, choosing the best R-factor path for routing packets can significantly improve the R-value, especially for periods where the direct path has a low R-value.

We selected a period of 5 hours where the network performance is poor and another similar period where it is acceptable, and calculated the R-factor for the direct path, the best single path and the best combination of two paths. Results are shown in Fig. 4.6. It is apparent that the R-factor in the two-path scheme is more stable than that in the single path cases. Compared to using only a single best R-factor path, there is a noticeable improvement in R-factor when path diversity is used for the period of poor performance,

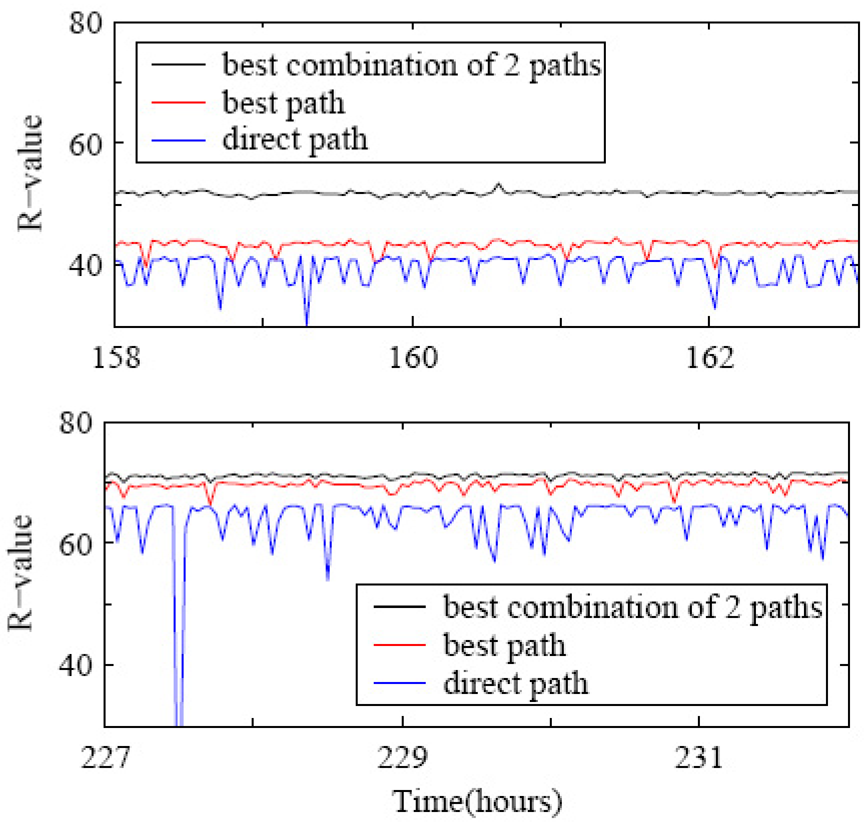


Fig. 4.6 R-factor calculated over two different periods for the direct path (bottom plot), the best-R path (middle plot) and the best-R combination of two paths (top plot). G.723 codec is assumed for R-factor computation.

as shown in the top graph of Fig. 4.6, whereas for the period of acceptable performance, R-factor value is not largely improved, as shown in the bottom of Fig. 4.6. An intuitive explanation for this difference is that in the second case the performance of the single best path is much better than that of any other path in the network, and therefore adding a second path cannot largely improve R-factor.

Simulation of the proposed optimal diverse routing

The proposed novel optimal diverse routing scheme presented in Section 4.2.2 is also simulated for the 7-node overlay network of the ISP described above. Feature vectors are computed from the end-to-end delay traces for all overlay links, and then sent to the data fusion center. The communication overhead between the overlay nodes and the data fusion center is kept small by transmitting only the feature vectors and the optimal diverse path decisions. The data fusion center computes the optimal diverse paths for all source-destination pairs.

The real optimal multipath refers to the pair of paths that gives the best R-factor when voice quality is evaluated at the receiver. We also evaluate R-factor for voice calls that are sent through the estimated optimal pair of paths. At the receiver of the voice call, adaptive play-out scheduling is applied.

Fig. 4.7 shows an example where the optimal pair of paths increases R-factor by a minimum of 5 and a maximum of 32 when the direct overlay path is in poor condition. The R-factor on the optimal pair of paths is also much more stable compared to that for the direct path. Fig. 4.8 shows the difference between the R-factor on the real optimal diverse paths and that on the estimated optimal diverse paths. The discrete difference in Fig. 4.8 is because R-factors on the best multipath and on the estimated best multipath are stable during the simulation time. For other source-destination pairs, the difference is not always discrete. The average difference for all the source-destination pairs is shown in Fig. 4.9. This estimation loss is due to the approximation of the network delay distribution on the concatenated paths and that on the pair of paths. As can be observed, the difference is very small in this network, which means that the estimated optimal pair of paths, which is selected based on the approximated shifted Gamma distributions, can provide a voice quality close to that given by the real optimal pair of paths. It is important to note that the method assumes there is no correlation between the pair of selected paths. The end-to-end delay follows the shifted Gamma distribution and the end-to-end loss is small. When these assumptions are not satisfied, we would see larger estimation losses. In that case, we would have to transmit the whole end-to-end delay measurements to the data fusion center to obtain a better estimate of the best pair of paths, which would incur a high communication cost.

4.3 Summary

This chapter is a preliminary study on diverse routing in a small-scale service overlay network based on the end-to-end delay trace measured by an Internet Service provider in Asia. Path diversity is shown to provide a stable and higher R-factor for VoIP in the service overlay network under study. We have also explored the potential use of shifted Gamma

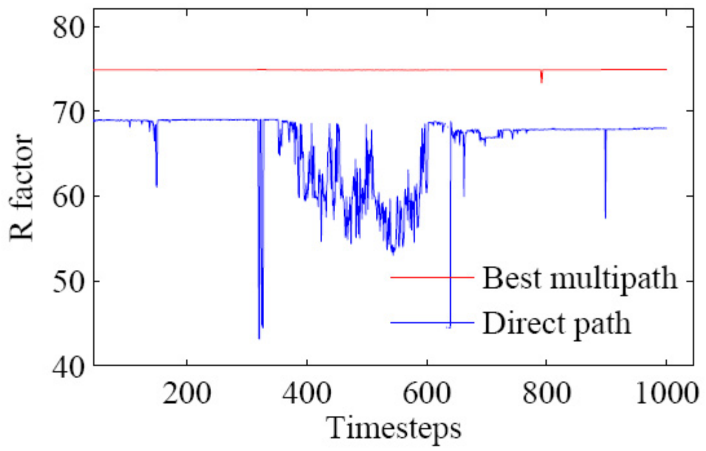


Fig. 4.7 R-factor on the optimal pair of paths (upper line) vs. R-factor on the direct path for a source-destination pair. G.729 codec is assumed for R-factor computation.

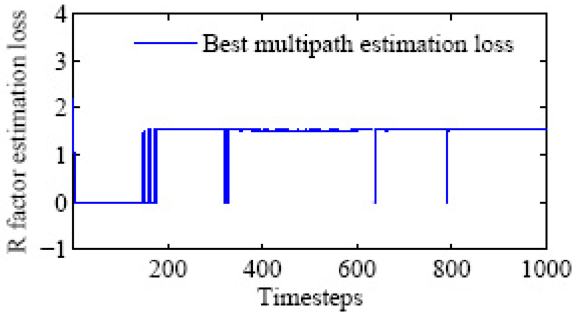


Fig. 4.8 Difference between the R-factor on the real optimal pair of paths and that on the selected optimal pair of paths.

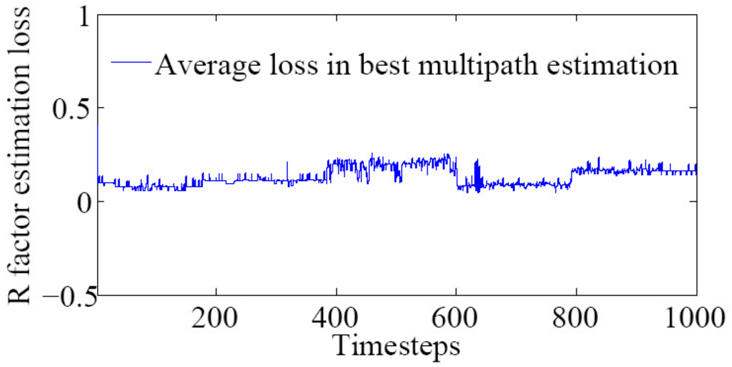


Fig. 4.9 Difference between the average R-factor on the real optimal pair of paths and that on the selected optimal pair of paths for all the source-destination pairs in the 7-node service overlay network.

distribution to estimate end-to-end network delay distribution. We proposed an optimal R-factor diverse path selection scheme, which cooperates with an adaptive playout scheduling scheme at the receiver to provide optimal quality for voice calls. The simulation shows that the proposed optimal diverse path selection algorithm gives a much more stable R-factor than that on the direct paths, and it also shows that the proposed method can improve R-factor as in the example. We found that the paths with the best performance vary depending on the time. As shown in Fig. 4.8, the shifted Gamma distribution approximation for the network delay distributions on the concatenated paths results in acceptable errors, in choosing the optimal diverse paths. Therefore, the estimated optimal diverse paths can provide as satisfactory quality for VoIP calls as can the real optimal diverse paths.

When network bandwidth resources are limited, multiple-description-coded voice packets can be sent instead of duplicate voice packets in the two-path diversity scheme. In this study, we considered only two-path diversity and it can be expected that with more than a pair of diverse paths, VoIP quality may be improved. However, as mentioned in [12], when four-path diversity is used, a quality decrease can occur due to link capacity saturation.

This centralized brute-force search method requires computation of all the end-to-end delay distributions for all pairs of candidate paths, which is computationally very expensive, and limits the scalability of this approach. Due to the complexity of computing end-to-end delay distributions, the candidate paths considered are only one-hop or two-hop paths for each source-destination pair. However, it can be expected that VoIP quality may be further improved with more than two-hop paths. In Chapters 5 and 6, we propose a method to find paths that are not limited to one or two hops, in order to improve VoIP quality. Moreover, we seek a method that is scalable, distributed, adaptive, and has low computational cost. In addition, in the simulation presented in Fig. 4.5, Fig. 4.6, we also saw that the resultant R-factor ratings are not very satisfactory with G.723 and G.729 codec when the network performance is poor. In the next stage of simulations, we use G.711 codec, which has much lower codec impairment [3, 108].

Chapter 5

Learning Minimum Delay Paths: A Distributed Approach

5.1 Introduction

In the previous chapter, we have shown how much VoIP quality can be improved by diverse routing in a small scale service overlay network based on the collected network delay measurements from an Internet Service Provider. This chapter presents a scalable QoS routing scheme for larger overlay networks.

In a dynamic network environment, it is challenging to adaptively find the minimum delay paths for service overlay networks. In this work, we propose a novel approach to actively probe and learn the minimum delay paths for VoIP calls. Learning automata are applied to probe unwastefully and learn the optimal paths in a distributed manner. We proposed four strategies to actively probe and learn the optimal paths. The performance of the four strategies is then evaluated in a fluid model of the AT&T backbone network. Simulation results show that the proposed active probing and learning strategies can converge to the minimum delay paths very quickly. We also prove the convergence of the learning automata and show that they converge to the user equilibrium for minimum delay routing.

5.1.1 Chapter Structure

The rest of the chapter is structured as follows. Section 5.2 shows the architecture of the proposed method. Section 5.3 presents the proposed active probing and learning method.

The simulation results for the proposed method are shown in Section 5.4. Section 5.5 analyzes the probing overhead of this method. Section 5.6 proves its convergence to the minimum delay paths. Section 5.7 summarizes this chapter.

5.2 Architecture

The architecture of the proposed QoS routing solution consists of three layers: service overlay network layer, probing layer and VoIP routing service layer, as shown in Fig. 5.1.

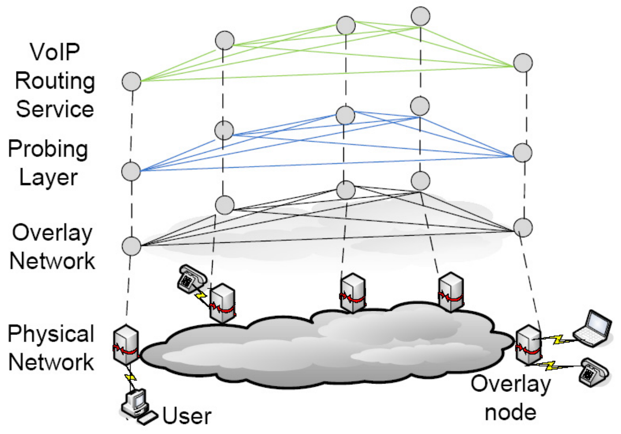


Fig. 5.1 Proposed network architecture for VoIP service provision in overlay networks. An overlay network is formed by interconnected overlay nodes. The probing layer and the VoIP routing service layer are the novel design in our work, which are presented in detail in Sections 5.3 and 6.4, respectively.

The service overlay network layer consists of two types of nodes: user nodes and overlay nodes. User nodes run VoIP applications. Each user node typically connects to an overlay node that is geographically closest to it, which we call its local overlay node. All the overlay nodes are interconnected via virtual connections. During the call initialization process, a caller connects to its local overlay node and uses the paths provided by the overlay network to communicate with its callee via the callee's local overlay node.

Overlay nodes are responsible for identifying routes that provide high quality of service through the overlay network. As the performance of the overlay links is random and unknown to the overlay nodes, it is necessary to actively probe the network performance for finding the minimum delay overlay paths. To accomplish this, the overlay nodes periodically probe alternative paths through the overlay network via a measurement plane to support VoIP routing decisions. UDP packets are sent periodically between each source and destination overlay nodes to measure round trip time of all the possible candidate paths. For the remainder of the thesis, we refer an overlay node as a node, unless otherwise specified. Probes are used to measure the .

Rather than probing all the candidate paths in a brute-force fashion, the probing process is controlled by learning automata. Detail of the probing algorithm is provided in Section 5.3. By using learning automata (a form of reinforcement learning agent), we quickly learn which paths have little potential to provide sufficient quality of service for a given destination. This allows us to significantly reduce the amount of probing traffic on low-quality paths, and focus on probing a smaller subset of paths that will likely have the optimal performance. The learning automata parameters are probability distributions over which each overlay node is to be probed for a given destination. Each overlay node maintains its own independent learning automata, so that the algorithm is completely decentralized. When a node receives a probe from another node, it randomly selects a next hop for the probe according to its learning automata parameters.

Each probe follows a round-trip path through the overlay, from origin to destination and back to its origin. When the probe returns to its origin node, it updates the learning automata parameters based on the measured Round-Trip Time (RTT). As RTT feedback is continuously-valued, the commonly used learning automata algorithms that are based on binary or discrete-valued feedback, such as LRI, LReP [109], are not applicable. Instead, we adopt the cross-correlation learning algorithm [110] that has been proposed for positive finite continuous feedback environments.

5.3 Active Probing and Learning (APL)

This section presents the detail of the probing layer in Fig. 5.1. The overlay network in Fig. 5.1 is modeled as a graph $G = (V, E)$, where $|V| = m$ and $|E| = m(m-1)$. Each overlay node S, $S \in V$, runs $(m-1)$ learning automata, with each automaton being responsible for actively probing the RTT from S to each destination D, $D \in V, D \neq S$. As the total number of paths for each source-destination pair is very large, it is not efficient to always probe all the paths for determining the minimum delay path. Thus, a learning algorithm is employed to save the cost on probing paths with large delays.

In the following subsections, we first give a brief description of the learning automaton in Section 5.3.1, then we present the active probing process in Section 5.3.2. The detailed learning algorithm is given in Section 5.3.3.

5.3.1 Preliminary: Learning Automaton

A learning automaton is a form of reinforcement learning agent. It can be represented by a tuple $\{Z, U, \Phi, f, \underline{\pi}, T\}$ [110, 111], where Z is the set of inputs to the automaton, U is the set of outputs of the automaton, Φ is the set of states that the automaton can take, f is the function that maps Φ to U, $\underline{\pi}$ represents the probability vector that governs the states of the automaton, and T is an operator that updates the probability vector $\underline{\pi}$. The probability vector $\underline{\pi}$ is updated as

$$\underline{\pi}(t+1) = T(z(u(t), t), \underline{\pi}(t)), \tag{5.1}$$

where $z(u(t), t)$, is the feedback from the environment to the automaton when it outputs $u(t)$ at time t as shown in Fig. 5.2. For the next time instant, the output of the automaton is given by

$$u(t+1) = f(\underline{\pi}(t+1)). \tag{5.2}$$

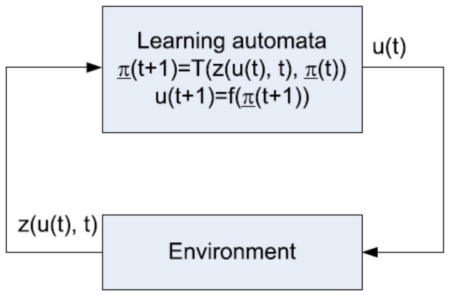


Fig. 5.2 Learning automaton. When the automaton outputs $u(t)$ at time t, the environment produces a feedback $z(u(t), t)$ to the automaton. Then the probability vector of the learning automaton is updated as in (5.1). Then for the next time instant, the output of the automaton $u(t+1)$ is given by (5.2).

In Fig. 5.2, the input to the learning automaton is in fact the feedback from the environment. It is called a P-model environment when the set of feedback Z equals {0,1}; it is called a S-model environment when Z is positive and continuous [110].

5.3.2 Active Probing Process

Before describing the active probing process, we first clarify the notation. Suppose a probing packet $pp(S,D)$ is scheduled to be sent from a source node S, $S \in V$, to a destination node D, $D \in V$. Let $\pi^D_{Sj}, \forall j \in V$, denote the probability that node j is the first hop of the probing packet $pp(S,D)$. For all the possible choices of j, $j \in V$, we define the probability distribution $\underline{\pi}^D_S = [\pi^D_{S1}, \pi^D_{S2}, ..., \pi^D_{Sj}, ..., \pi^D_{Sm}]$, with $\sum_{j=1}^{m} \pi^D_{Sj} = 1$ and $1 \geq \pi^D_{Sj} \geq 0$. The probability distribution $\underline{\pi}^D_S$ is maintained only at the source node S.

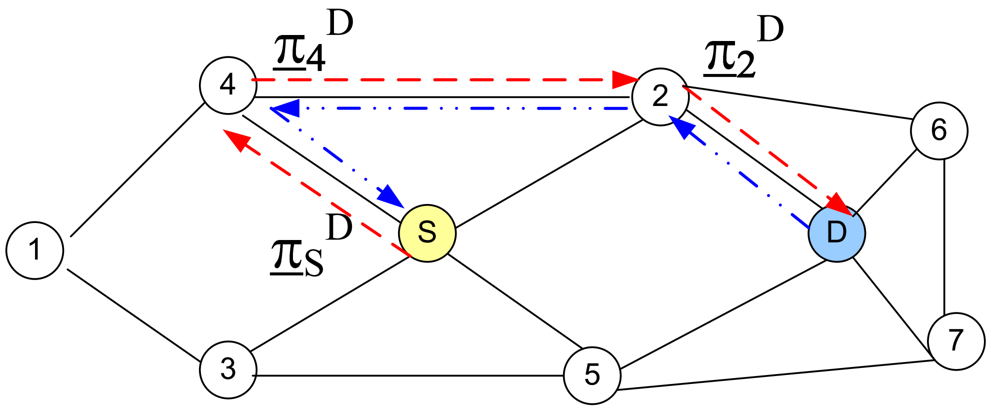


Fig. 5.3 Hop-by-hop learning of the minimum delay path from source S to destination D. The probability distribution parameters of each learning automaton are initialized with the uniform initialization. Each next-hop node of a probing packet is chosen randomly according to its previous hop node's learning automaton's parameters. The identifier and the arrival time at each intermediate node is recorded by the probe. Once the probe reaches its destination D, D sends the probe back immediately to its previous hop, i.e. node 2 in this example. When node 2 receives the probe, it updates its learning automaton according to the RTT from 2 to D. Then node 2 passes the probe to its previous hop, i.e. node 4, 4 updates its corresponding learning automaton and then passes the probe to its previous hop, and so on, until the probe returns to its source S, and S updates its corresponding learning automaton.

With the $\underline{\pi}^D_S$ well defined for all the source-destination pairs S and D in the overlay network, the active probing process for a probing packet $pp(S,D)$ is as follows. As illustrated in Fig. 5.3, the first hop of the probing packet is selected randomly based on the probability distribution, e.g. node 4 is selected with probability π^D_{S4}. In this example, node 4 is selected as the first-hop of $pp(S, D)$. Then, the probing packet is passed to node 4. The random next-hop node selection process is repeated at node 4 based on the distribution $\underline{\pi}^D_4$. Such a procedure continues until the probing packet reaches its destination D. Then the probing packet is sent back from D to S.

The active probing method above is a distributed and scalable method, as the next-hop node for each probing packet $pp(S, D)$, $\forall S, D \in V$, is determined locally at each intermediate node. The RTT measured by the probing packet depends on the random path formed by the randomly selected next hops.

5.3.3 Active learning algorithm

The probability distribution $\underline{\pi}^D_S$ is updated during the active learning process, thus we rewrite $\underline{\pi}^D_S$ as a function of time t, i.e. $\underline{\pi}^D_S(t)$. To update the probability distribution $\underline{\pi}^D_S(t)$, we adopte the cross-correlation learning algorithm [110] that works for S-model environment[1].

Suppose at time t, a probing packet $pp(S, D)$ returns to its source S with RTT feedback $\mathrm{d}^D_S(u)$. $\mathrm{d}^D_S(u)$ is a function of u, where u is the first hop of the probing packet. We then normalize $\mathrm{d}^D_S(u)$ with respect to the maximum RTT d_{max}[2] as in (5.3) to obtain the environment reward $z^D_S(u)$, $z^D_S(u) \in [0, 1]$ for the S-model environment [110],

$$z^D_S(u) = (1 - \frac{d^D_S(u)}{d_{max}})^+, \tag{5.3}$$

where $(1 - \frac{d^D_S(u)}{d_{max}})^+$ is non-negative part of the function $1 - \frac{d^D_S(u)}{d_{max}}$. Suppose the current

[1]RTT is a positive continuous feedback to the automata. Thus it can be represented by S-model [110].

[2]d_{max} is predefined to be a constant value that is reasonably large for VoIP calls. $d_{max} = 2$ seconds in our simulations, which means the maximum tolerable RTT for a probing packet is 2 seconds. If d_{max} is set small, e.g. $d_{max} = 0.5$ second, the initial learning will be slow because the automata parameters are not updated from RTT feedback larger than 0.5 second, which is highly probably during the initial learning process due to initial long random path. If d_{max} is large, e.g. $d_{max} = 5$ seconds, the whole learning process will be faster because the reward for the RTT feedback will be relatively large, which is similar to using a larger learning gain, where the update step size is large. However, d_{max} cannot be too large because otherwise the resolution of path delay will be small. One should also note that d_{max} and the learning gain should be kept constant during the whole learning process for each learning automaton, which is required for the convergence of the cross-correlation learning algorithm [110]. However, one may choose different d_{max} and learning gain for different learning automata in the overlay network, although I set them to be the same in all my simulations.

probability distribution is $\underline{\pi}_S^D(t) = [\pi_{S1}^D(t), \pi_{S2}^D(t), ..., \pi_{Sj}^D(t), ..., \pi_{Sm}^D(t)]$. Applying the cross-correlation learning algorithm [110], we have

$$\pi_{Sj}^D(t^+) = \pi_{Sj}^D(t) + g\ z_S^D(u)\ (\delta_{ju} - \pi_{Sj}^D(t)), \quad \forall j = 1, \ldots, m, \tag{5.4}$$

where t^+ is the time instant right after the update at time t, g is the learning gain, δ_{ju} satisfies

$$\delta_{ju} = \begin{cases} 1, \text{if } j = u; \\ 0, \text{else.} \end{cases} \tag{5.5}$$

The update in (5.4) increases $\pi_{Su}^D(t)$ according to the value of $d_S^D(u)$. The smaller $d_S^D(u)$ is, the more increase for $\pi_{Su}^D(t)$, while $\pi_{Sj}^D(t)$, $j \neq u$, are decreased accordingly. When the learning gain is sufficiently small, the higher the learning gain is, the more increase for $\pi_{Su}^D(t)$ and the more decrease for $\pi_{Sj}^D(t)$, $j \neq u$, and thus the faster the learning automata converge[3]. A theorectical method for deriving a sufficiently small learning gain is presented in Appendix A.2.

With the updating formula in (5.4), we can show that the probability distribution $\underline{\pi}_S^D(t)$ converges to a distribution that satisfies the property given in (5.6), i.e. if the minimum delay path from S to D is unique, and link (S, j^*) is on the minimum delay path, then

$$\pi_{Sj}^D = \begin{cases} 1, \text{if } j = j^*; \\ 0, \text{otherwise.} \end{cases} \tag{5.6}$$

The proof of the convergence is given in Section 5.6. Note that in practice, we will set a bound on the values of π_{Sj}^D, which is detailed in Section 5.6.2, in order to avoid being trapped at any local minimum, so that the learning automata can adapt to network changes in a non-stationary network environment.

Also note that the learning algorithm converges to paths with the minimum mean delay. In the simulations of Section 5.4, we can see that it can converge in 5 seconds when each node send a probe per 5 ms. If there are abrupt network delay changes that last for a short time of period, e.g. less than 5 seconds, it is hard for the algorithm to adapt to such

[3]I started from a small learning gain 0.0001, and increased it to 0.0005, 0.001, 0.005, 0.01, 0.02. For learning gains smaller or equal to 0.01, I observed convergence of the learning algorithm and that the learning speed increases proportionally with the learning gain, but when the learning gain equals 0.02, I observed false convergence.

changes. In this case, it might be advantageous to maintain second order staistics, i.e. not just the mean performance, but also the variance, to be able to react faster to sudden severe link quality degradations, which is left for future work.

5.3.4 Initialization of the Learning Automata

One issue for using the cross-correlation learning algorithm is how to initialize the probability distributions $\underline{\pi}_S^D(t)$. For each source-destination pair (S, D), $\forall S, D \in V$, the probability of choosing the first hop as node j at the starting time 0 is $\pi_{Sj}^D(0)$.

Uniform initialization

The uniform initialization method allows all possible paths to be scanned fairly at the beginning of probing. Uniform initialization of the probability distribution $\underline{\pi}_S^D(0)$ allows all nodes in V except node S equally probably being the next-hop of node S, as shown in (5.7):

$$\pi_{Sj}^D(0) = \frac{1}{m-1}, j \in \{1, ..., m\}\text{S}. \tag{5.7}$$

When a node j, $j \neq S$, receives a probing packet $pp(S, D)$, it may send $pp(S, D)$ back to node S, as the next hop choice is made independently at each node. This can introduce random loops on the probing path, as detailed in Appendix A.3. It can be shown that, random loops occur with high probability at the starting stage, and the probability increases with network sizes. However, as paths with loops have higher delays than loop-free paths, the learning automata will automatically learn to avoid these loops [63].

Geographical location aware initialization

As mentioned above, uniform initialization method can result in a high probability of random loops at the initial stage of the probing. While this is not detrimental to the algorithm, probing resources are needed to learn not to use paths with loops. In order to avoid these random loops, we propose a geographical location aware initialization method. Let $\mathcal{L}(S, D)$ be the Euclidean distance (or the great circle distance on a sphere) between nodes S and D. Geographical-location-aware initialization, as given by:

$$\pi_{Sj}^D(0) = \frac{\text{I}(\mathcal{L}(j, D) < \mathcal{L}(S, D))}{\sum_j \text{I}(\mathcal{L}(j, D) < \mathcal{L}(S, D))}, j = 1, ..., m \tag{5.8}$$

where I$(\cdot)$ is an indicator function, guarantees that only nodes with distances to the destination D less than that from the origin S are explored. In this way, loops can be avoided in the whole probing process and the initial RTT can be reduced. However, one should note that this method could potentially rule out paths that are better than those which satisfy the condition of always moving closer to the destination.

5.3.5 Hop-by-hop learning

We have mentioned how a probing packet is sent from its source to its destination in Section 5.3.2. Once a probe reaches its destination, it needs to return feedback to the sender. There can be multiple ways to send it back. One way is to select the backward path randomly as what we did in the forward path selection. However, this is not efficient for updating the probability distributions (in terms of the number of updates per second). The most efficient way is to use the hop-by-hop learning, as illustrated in Fig. 5.3 and Fig. 5.4 for the uniform initialization and the geographical-location-aware initialization, respectively.

In hop-by-hop learning, feedback from a probing packet $pp(S, D)$ is sent back to its source S by the exact reverse path of its forward path from S to D, which requires the probing packet records all the intermediate nodes on its forward path. For example, in Fig. 5.3, the forwarding path for the probing packet $pp(S, D)$ is $\{(S, 4), (4, 2), (2, D)\}$. Then its backward path is $\{(D, 2), (2, 4), (4, S)\}$. Thus, all the probability distributions $\underline{\pi}_S^D(t)$, $\underline{\pi}_4^D(t)$ and $\underline{\pi}_2^D(t)$ are updated according the probing packet's RTT measurements $d_S^D(4)$, $d_4^D(2)$ and $d_2^D(D)$, respectively, as given in (5.4). This hop-by-hop learning algorithm is shown in Algorithm 3. A more clear view of the algorithm is shown by the flow chart in Fig. 5.5.

The difference between Fig. 5.3 and Fig. 5.4 is that the uniform initialization method considers all the possible paths for learning the minimum delay path, while the geographical location aware initialization considers only a subset of the possible paths. Moreover, as mentioned before, the uniform initialization method leads to random loops at the initial stage, although the loops eventually disappear [63]. The geographical location aware initialization method has no loop throughout the whole process. However, the optimal paths learned may be a suboptimal path if the real minimum delay path is excluded by the geographical location aware initialization.

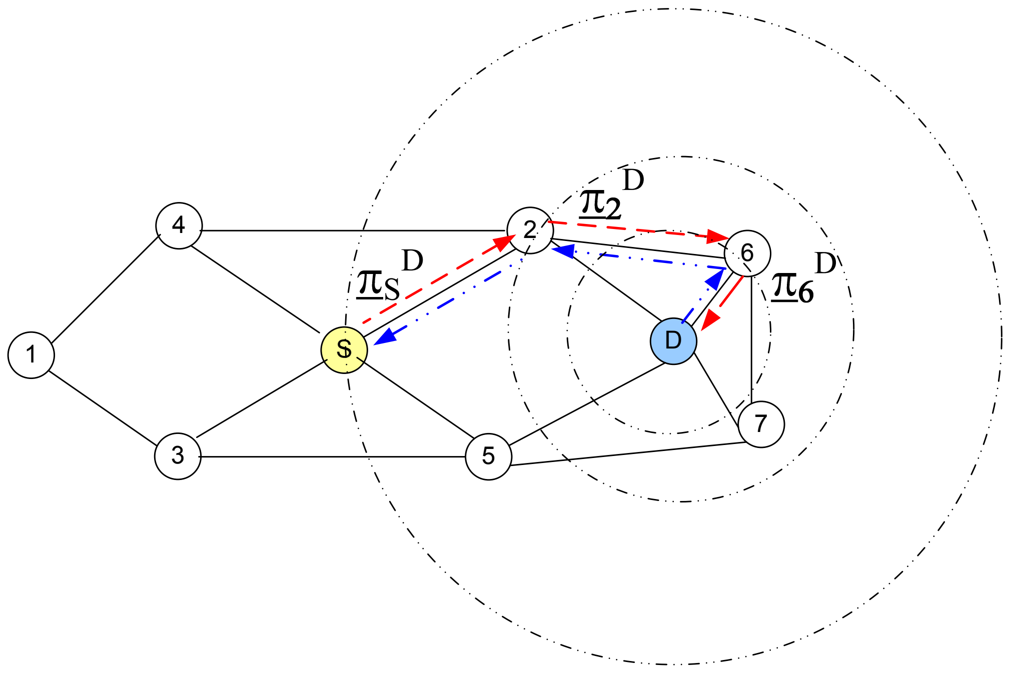


Fig. 5.4 Hop-by-hop learning of the minimum delay path from source node S to destination node D. The active probing and learning process is similar to that in Fig. 5.3, that except this method works on a smaller set of next-hop nodes, as defined by the geographical location aware initialization.

Require: Probing packet $pp(S, D)$, current node ID k.
Ensure: Forward the probing packet to a random next hop overlay node, or update the probability distribution $\underline{\pi}_k^D$.

```
1: if The probing packet is on its forward path from $S$ to $D$ and $k == D$ then
2:     Send the probing packet back to its source node $S$, choose its next hop as the previous hop where it comes from;
3: else if $pp(S, D)$ is on its forward path from $S$ to $D$, and $k \neq D$ then
4:     Send $pp(S, D)$ to a next hop node chosen randomly from distribution $\underline{\pi}_k^D$, and append node $k$ in the probing packet's intermediated node list;
5: else
6:     Compute the RTT from $k$ to $D$, update the distribution $\underline{\pi}_k^D$, send $pp(S, D)$ to node $k$'s last hop on the forward path.
7: end if
```

Algorithm 3: Hop-by-hop learning algorithm

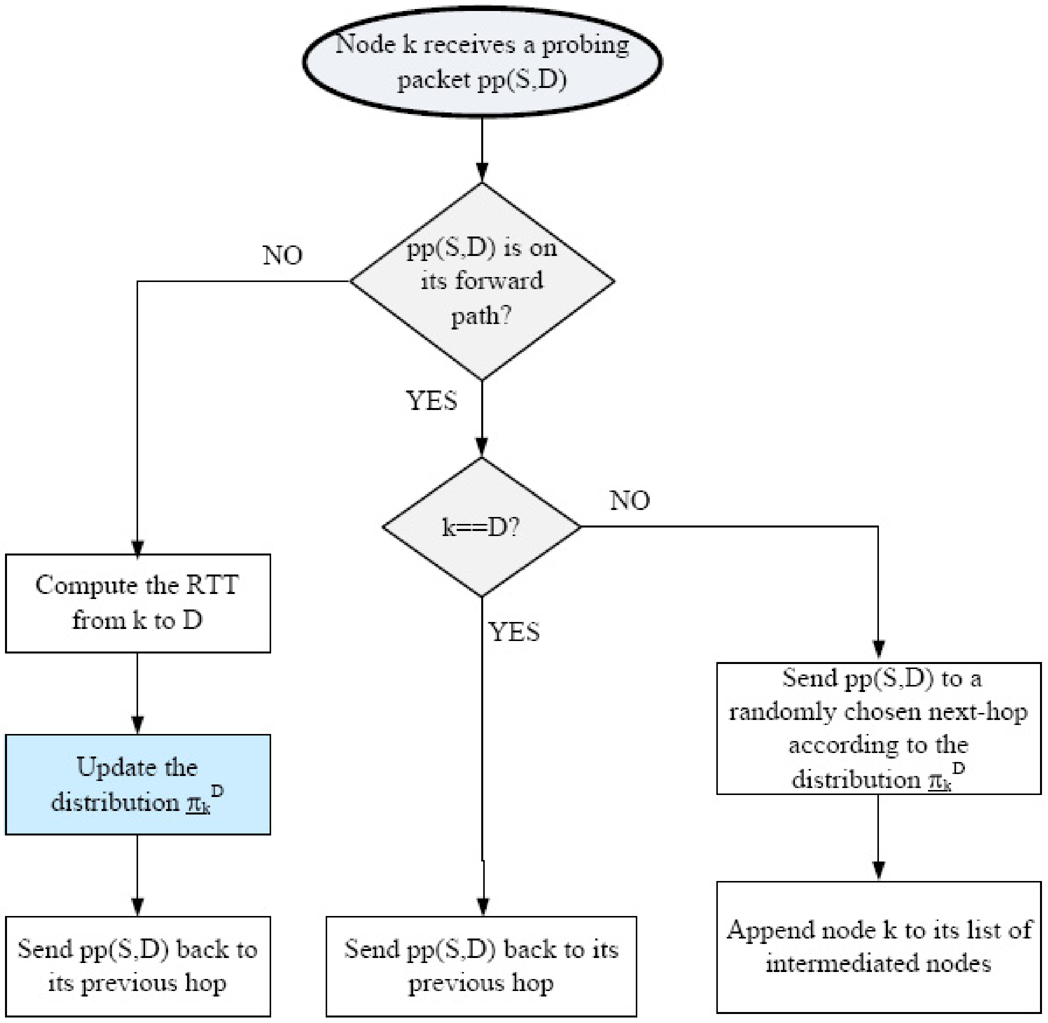

Node k receives a probing packet pp(S,D)
pp(S,D) is on its forward path?
NO
YES
k==D?
NO
YES
Compute the RTT from k to D
Update the distribution π_k^D
Send pp(S,D) back to its previous hop
Send pp(S,D) back to its previous hop
Send pp(S,D) to a randomly chosen next-hop according to the distribution π_k^D
Append node k to its list of intermediated nodes

Fig. 5.5 Flow chart for hop-by-hop learning algorithm. This process is started whenever a probing packet is received.

End-to-end learning

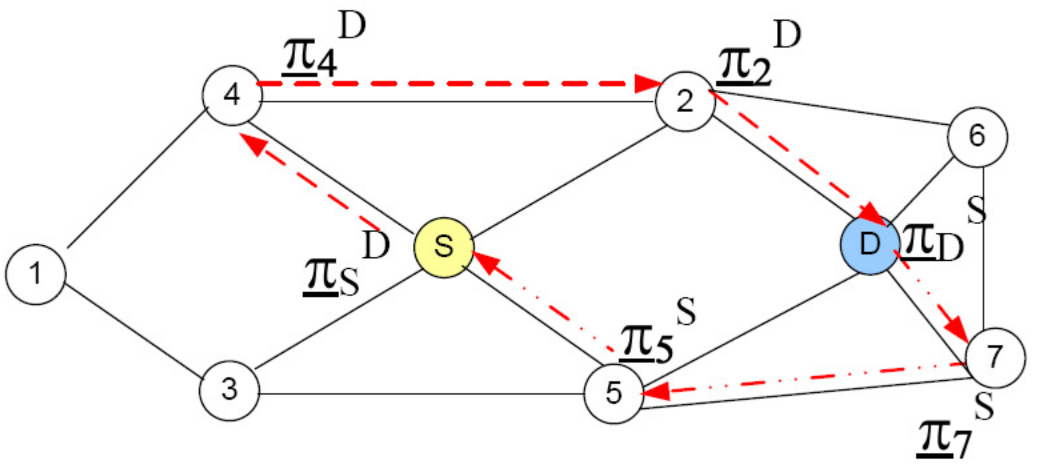


Fig. 5.6 End-to-end learning of the minimum delay path from S to D. The parameters of the learning automata at each node are initialized with the uniform initialization method defined in section 5.3.4.

In end-to-end learning, the backward path of a probing packet is also a random path. Each intermediate node i on the forward path from S to D and on the backward path from D to S chooses its next hop node randomly according to its own probability distribution $\underline{\pi}_i^D$ on the forward path or $\underline{\pi}_i^S$ on the backward path. Therefore, only the source node S can get the RTT measurement from the probing packet and update its learning automaton, i.e. when S receives the probing packet $pp(S, D)$ coming back from D to S, it updates the probability distribution $\underline{\pi}_S^D$.

For example, Fig. 5.6 and Fig. 5.7 illustrate the active probing and end-to-end learning process for a probing packet $pp(S, D)$ sent from source S to destination D. In Fig. 5.6, the source node S initializes its probability distribution $\underline{\pi}_S^D$ uniformly according to (5.7); in Fig. 5.7, the probability distribution $\underline{\pi}_S^D$ is initialized according to (5.8). The first hop of the probing packet is selected randomly according to the probability distribution $\underline{\pi}_S^D$, i.e. node 4 in Fig. 5.6. Similarly, all the following hops are selected. This process continues until the probing packet reaches its destination D. D will then send it back to S by randomly selecting the reverse path in a similar way as in selecting the forward path. The probing packet is passed to the randomly selected next hops until it returns to its origin node S. When S receives the probe, it updates the probability distribution $\underline{\pi}_S^D$ from the RTT measurement according to the cross-correlation learning algorithm.The algorithm for the end-to-end learning method is given in Algorithm 4. The flow chart for the algorithm is shown in Fig. 5.8.

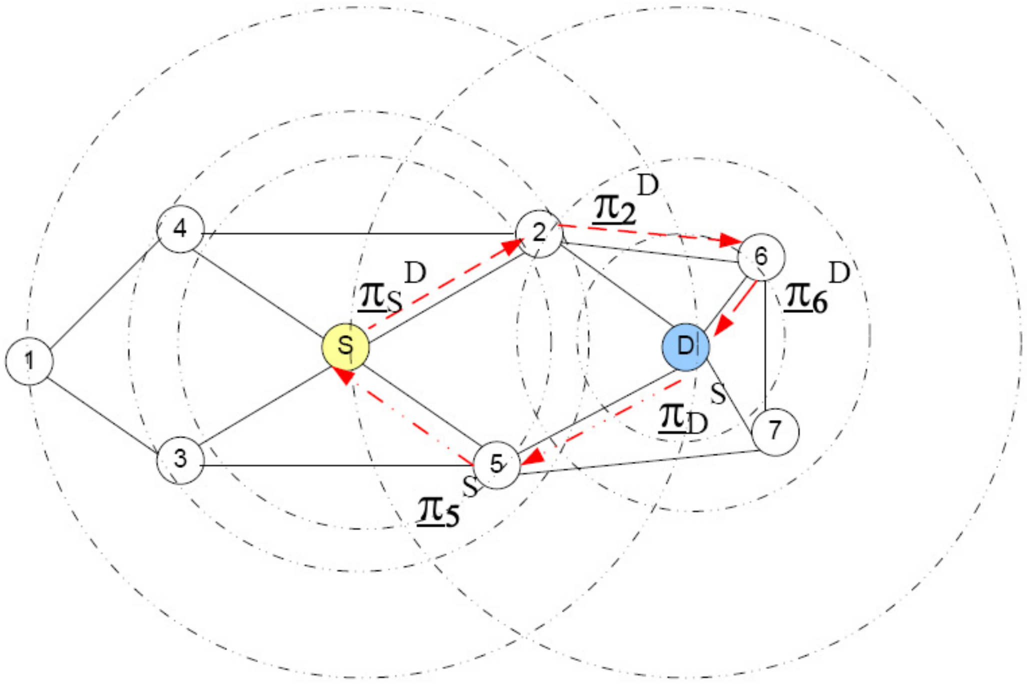


Fig. 5.7 End-to-end learning of the minimum delay path from source S to destination D. The learning automata are initialized using the geographical location aware initialization method. Therefore the candidate next hop nodes of each node are restricted to a smaller set of network nodes compared to that in the uniform initialization method. The method of probing and probability distribution updating are the same as that in Fig. 5.6.

In end-to-end learning, the probing packet does not need to record each intermediate node on its forward path, which makes it easier to implement at the cost of slower learning speed and the size of a probing packet is smaller than that in the hop-by-hop learning method. One thing to note is that the end-to-end learning method will be very inefficient if precise one-way delay measurement is available and used as feedback to the learning automata; while in this case, the hop-by-hop learning method is still very efficient in updating the probability distributions.

```
Require: Probing packet $pp(S, D)$, current node ID $k$
Ensure: Forward the probing packet to a random next hop overlay node, or update
    the probability distribution $\underline{\pi}_S^D$.
1: if The probing packet is on its forward path from $S$ to $D$ and $k == D$ then
2:     Send the probing packet back to its source node $S$, choose a random next hop
       node based on distribution $\underline{\pi}_D^S$;
3: else if $pp(S, D)$ is on its forward path from $S$ to $D$, and $k \neq D$ then
4:     Send $pp(S, D)$ to a next hop node chosen randomly from distribution $\underline{\pi}_k^D$;
5: else if The probing packet $pp(S, D)$ is on its backward path from $D$ to $S$, and
   $k == S$ then
6:     Compute the RTT from $S$ to $D$, update the distribution $\underline{\pi}_S^D$;
7: else if The probing packet $pp(S, D)$ is on its backward path from $D$ to $S$, and $k \neq S$
   then
8:     Compute the RTT from $k$ to $D$, update the distribution $\underline{\pi}_k^D$;
9: end if
```

Algorithm 4: End-to-end learning algorithm

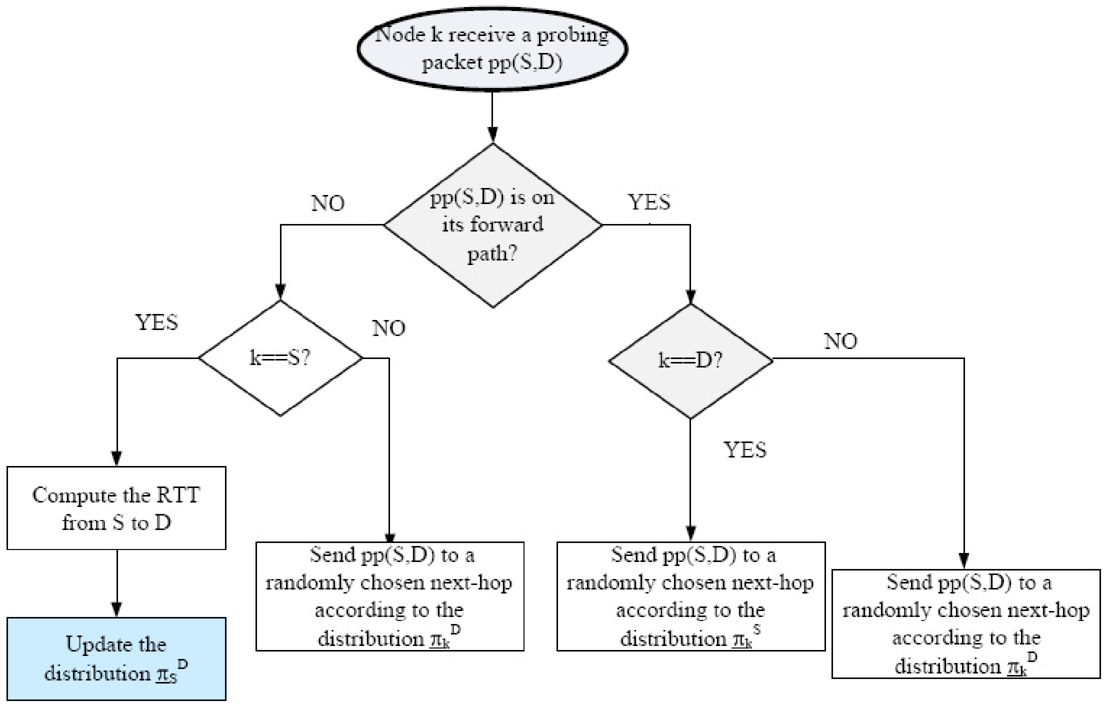

Node k receive a probing packet pp(S,D)
pp(S,D) is on its forward path?
NO
YES
k==S?
YES
NO
k==D?
NO
YES
Compute the RTT from S to D
Update the distribution π_S^D
Send pp(S,D) to a randomly chosen next-hop according to the distribution π_k^D
Send pp(S,D) to a randomly chosen next-hop according to the distribution π_k^S
Send pp(S,D) to a randomly chosen next-hop according to the distribution π_k^D

Fig. 5.8 Flow chart for end-to-end learning algorithm. This process is started whenever a probing packet is received.

5.4 Experiments

5.4.1 3-node network

We first learn the minimum delay paths in a 3-node fully connected network, as shown in Fig. 5.9. Suppose the three nodes are also overlay nodes. Active probing and learning automata, specifically hop-by-hop learning with uniform initialization, are implemented at all three nodes.

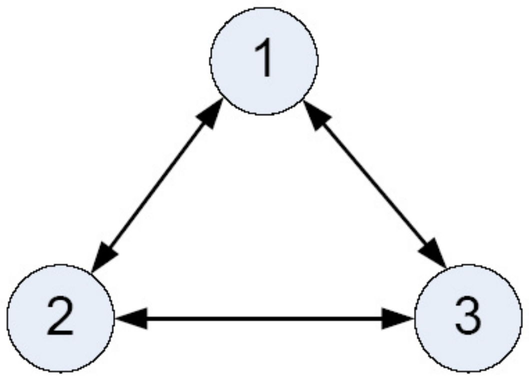


Fig. 5.9 Fully connected 3-node network

The goal of the experiment is to understand whether the cross-correlation learning algorithm is able to converge to the minimum delay path when the network is in its transient state. We set the network as follows. We know that the queuing delay on a link with high link utilization rate can increase until reaching its maximum queuing capacity during the network warm-up period. In this period, the network performance is dynamic and transient. Thus, we tune one link in the 3-node network to be heavily loaded to see if the algorithm converges to the minimum delay paths. Note that the actual values of the network setting are not important as long as they can be used to generate the desired end-to-end delay traces. Suppose the capacities of all links are 622.08 kbps. There is fractional Brownian motion traffic only from node 1 to node 2, with the mean traffic demand 604 kbps. i.e. the utilization rate for the directed link (1,2) is 97.09%. Then the queuing delay at link (1,2) can be close to infinite for infinite queuing capacity. In our simulation, we set queuing capacity to be finite. Then, traffic will be dropped when the queue is full. The propagation delays of all links are set to be 0.06 sec. As there is no background traffic on links other than link (1,2), we only show the queuing delay on link (1,2) in Fig. 5.10. It can be seen

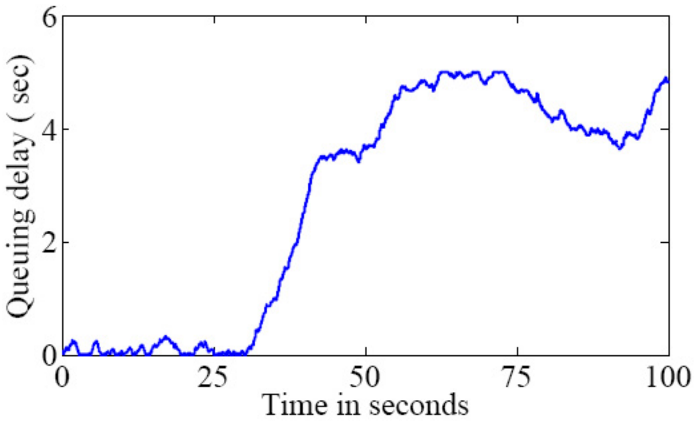


Fig. 5.10 The queuing delay on the heavily loaded link (1,2). The delay value does not increase infinitely because the queuing capacity of the link is set to be finite.

that the time period from 0 second to 50 seconds is the transient warm-up period for the network. The traffic arriving at link (1,2) can still be emptied from 0 second to 30 seconds. The mean queuing delay on link (1,2) from 0 to 30 seconds is 0.05 sec, which is less than the propagation delay on link (1,3). During this period, the minimum delay path from node 1 to node 2 is thus the direct link (1,2). The 30 seconds is a turning point. From 30 seconds to 50 seconds, the queue at link (1,2) starts to build up, as indicated by the increase of queuing delay. From 50 seconds to 70 seconds, the queue keeps on accumulating until it reaches its maximum queuing capacity, and afterwards, the queueing delay drops again to the delay level at around 50 seconds. As the queuing delay from 50 seconds to 100 seconds appear stationary, we consider the transient warm-up period ends at 50 seconds for this network. The queuing delay on link (1,2) is very high in the stationary state from 50 seconds on (much larger than the end-to-end delay on path {(1,3),(3,2)}), then the minimum delay path for a voice call originating from node 1 and destined for node 2 should be {(1,3),(3,2)}, on which the end-to-end delay is 0.06+0.06=0.12 second.

Figure. 5.11 shows the evolution of the probability distribution $\underline{\pi}_1^2(t)$. This figure shows only $\pi_{12}^2(t)$ (thick line) and $\pi_{13}^2(t)$ (thin line) as $\pi_{11}^2(t) = 0$ all the time. It can be seen that the probability $\pi_{12}^2(t)$ for sending a probing packet to node 2 is higher than that to node 3 and it keeps increasing until around 30 seconds. This is because the mean delay on path {(1,2)} (0.05+0.06=0.11 sec) is lower than that on path {(1,3),(3,2)} (0.06+0.06=0.12 sec). From 30 seconds on, the mean delay on path {(1,2)} increases and surpasses that on path

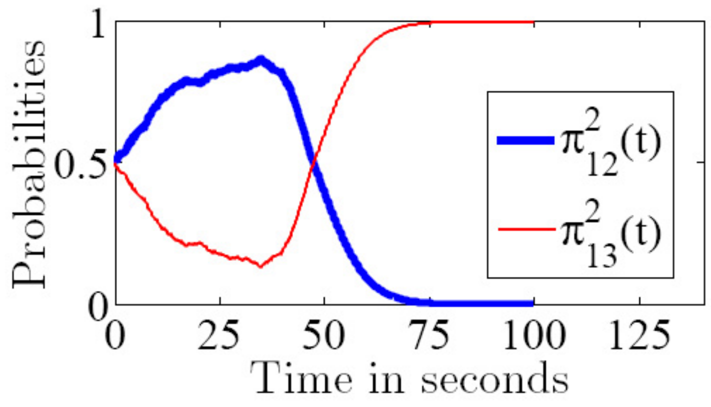


Fig. 5.11 The evolution of the probability distribution $\underline{\pi}^2_1(t)$ when the learning gain $g = 0.001$. This figure only shows $\pi^2_{12}(t)$ (thick line) and $\pi^2_{13}(t)$ (thin line) as $\pi^2_{11}(t) = 0$ all the time.

{(1,3),(3,2)}. Then the probability $\pi^2_{12}(t)$ starts to decrease and $\pi^2_{13}(t)$ starts to increase, until they converge to $\pi^2_{12}(t) = 0$ and $\pi^2_{12}(t) = 1$ at around t=100 seconds. This shows that before the learning automata converge, (e.g. at t=30 seconds), the cross-correlation learning algorithm is able to learn the current minimum delay path adaptively in a transient network environment.

Fig. 5.12 shows the average delays for all source-destination pairs on the learned paths (the paths taken by the probing packets) when the learning gain $g = 0.001$ (on the left) and $g = 0.01$ (on the right), on the shortest hop paths and on the optimal paths. It can be seen that the average delay on the shorted hop paths increases drastically at around 30 seconds due to the delay increase on link (1,2) shown in Fig. 5.10. On the optimal paths, the average delay is 0.153 second from time 0 to 30 seconds, while it is around 0.16 second from the time 30 seconds to 100 seconds. On the learned path when the learning gain $g = 0.001$, it can be seen that the average delay starts to increase at 30 seconds, and starts to decrease at around 45 seconds. The increase at 30 seconds is due to the increase in queuing delay on link (1,2), as shown in Fig. 5.10 and the high probability of probing path {(1,2)}, as shown in Fig. 5.11. The decrease at around 45 seconds is because the probability of probing path {(1,3),(3,2)} starts to be higher than that of probing path {(1,2)}. However, when the learning gain is 0.01, as shown on the right of Fig. 5.12, the learning algorithm converges to a suboptimal value.This is becuase the learning automata have already converged to the current minimum delay path, i.e. $\pi^2_{12}(t) = 1$ and $\pi^2_{13}(t) = 0$ before the mean delay

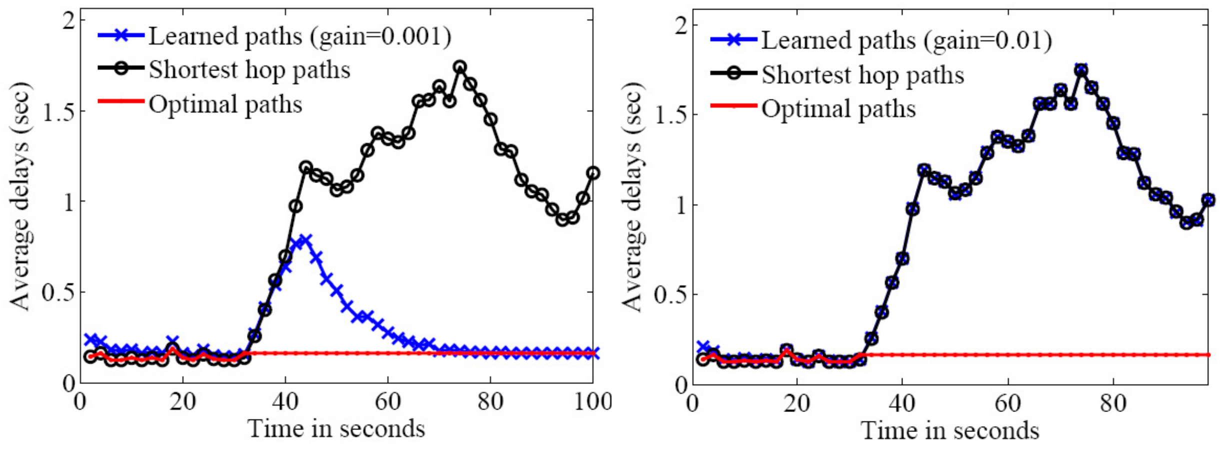


Fig. 5.12 The average delay for all source-destination pairs when the learning gain $g = 0.001$ (on the left) and $g = 0.01$ (on the right). This shows that the learning algorithm can keeps track of the current minimum delay path when the learning gain g is small enough, e.g. $g = 0.001$ in this example.

on path {(1,2)} starts to increase at 30 seconds. Therefore, when the network dynamics changes, the learning algorithm cannot track the new minimum delay path. To avoid this, we can set a threshold on the maximum and minimum values of the probabilities, as given in Section 5.6.2.

By varying the learning gain in this network environment, we also find that the convergence speed of the learning automata is sensitive to the learning gain. The general trend is that the learning algorithm converges to the optimum faster for larger gains. The sensitivity of the learning speed to the learning gain setting is also shown in the simulation of a larger network in Fig. 5.13 of Section 5.4.2[4].

5.4.2 Experiment in a 50-node model of the AT&T backbone network

The 50-node model of the AT&T backbone network and its setting have been described in Section 3.3. We chose 10 of the 50 nodes in Fig. 3.11 to form a full mesh overlay network. The overlay nodes were chosen to be geographically distributed across the network[5]. The

[4]In Fig. 5.13, the setting of learning gain $g = 0.01$ does not lead to false convergence because its network environment does not experience drastic change during the learning process and we set a bound on the probability values.

[5]The selected 10 overlay nodes are: Sacramento, CA, Las Vegas, NV, Houston, TX, Raleigh, NC, Dayton, OH, Phoenix, AZ, Nashville, TN, Pittsburg, PA, New York, NY, Anaheim, CA. A more strategic overlay network construction method is left for future work

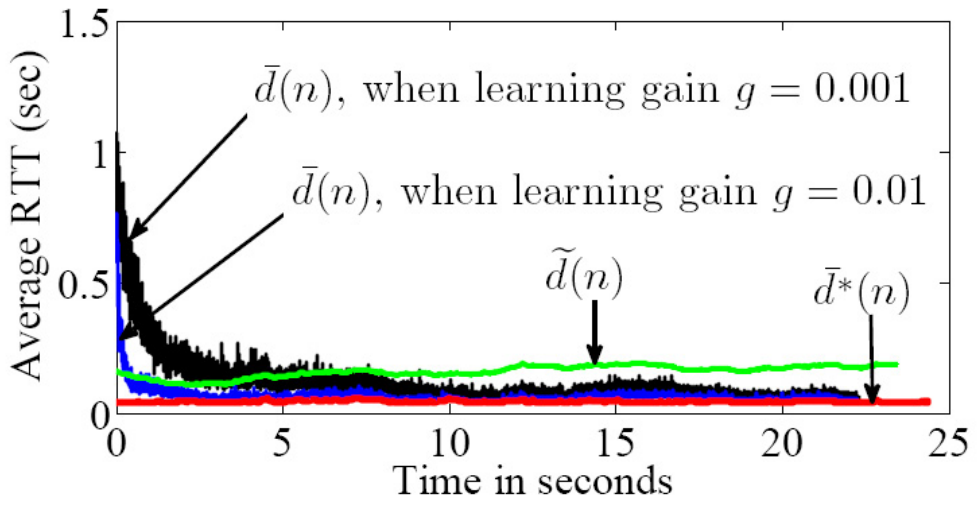


Fig. 5.13 Average RTT measured by the probing packets between all source-destination pairs, versus that on the minimum delay paths and that on the minimum hop paths.

active probing and learning algorithm described in Section 5.3 is simulated in this overlay network[6]. For each source-destination pair in the overlay network, a probing packet is sent every 5 ms to measure RTT. Simulation of the active probing and learning process is based on the fluid network simulation described in Section 3.3, where time is slotted into 5 ms bins. At the nth time slot, each overlay node S send one probing packet $pp_n(S, D)$ to each destination nodes D in the overlay network. Note that only one potential next-hop of all the possible next-hop options from S to a given destination D is probed during each time-slot. Also note that when the learning algorithm converges, the paths that the probing packets transit are the minimum delay paths, as given in (5.6).

In the following, I present the simulation results on the average delay of all source-destination pairs in the overlay network.

Average Latency

Let $d_n(S, D)$ denote the RTT measured by the probing packet $pp_n(S, D)$, $S, D \in V$, and let $\bar{d}(n)$ denote the average of the RTT for all source-destination pairs. Then,

$$\bar{d}(n) = \frac{1}{|V|(|V|-1)} \sum_{i \neq j} d_n(i, j) \tag{5.9}$$

Similarly, the average RTT for the minimum hop paths and that for the minimum delay paths can be defined[7]. Denote them as $\widetilde{d}(n)$ and $\bar{d}^*(n)$, respectively. Let ${d^d}_n(S, D)$ denote the RTT measured for the shortest hop paths (i.e. the direct paths determined by the underlying network) and ${d^*}_n(S, D)$ denote the RTT measured for the minimum delay paths.

$$\widetilde{d}(n) = \frac{1}{|V|(|V|-1)} \sum_{i \neq j} {d^d}_n(i, j) \tag{5.10}$$

$$\bar{d}^*(n) = \frac{1}{|V|(|V|-1)} \sum_{i \neq j} {d^*}_n(i, j) \tag{5.11}$$

Then, a comparison of $\bar{d}(n)$, $\widetilde{d}(n)$ and $\bar{d}^*(n)$ is shown in Fig. 5.13. The average delay $\bar{d}(n)$ is then shown in black line or blue line for the learning gain of 0.001 and 0.01. It can be seen that $\bar{d}(n)$ converges to $\bar{d}^*(n)$ as n increases, which indicates that the probing paths converge to the minimum delay paths, i.e. the probability distribution $\underline{\pi}_S^D(t)$ converges to the distribution that satisfies property (5.6). Fig. 5.13 also shows that the convergence speed increases proportionally as the learning gain increases from 0.001 to 0.01. We have also simulated the learning algorithm for larger gains and observed false convergence when the learning gain is greater than 0.01. Comparing with the example on the right of Fig. 5.12, we can state that the learning gain has to be sufficiently small to avoid false convergence in a dynamic network environment, as derived in Section A.2, and a bound should be set on the probability values to be able to adapt to network changes.

[6]The advantage of using the simulated network is that we can always verify that the learning algorithm converges to the minimum delay paths since we can compute the real minimum delay paths in the simulated (stationary) network. This is one reason why we do not run the algorithm in the real Internet because it is impossible to determine which path is the minimum delay path in the non-stationary real Internet, e.g. in the PlanetLab [81] environment.

[7]The minimum hop paths and the minimum delay paths are computed with Dijkstra's Algorithm [112], with the overlay network adjacency metric and the overlay link latency metric respectively.

Scalability

We then chose the overlay network size to be 15, 20 and 25. The overlay nodes were chosen to be geographically distributed across the network. The same setting (i.e. hop-by-hop learning with uniform initialization, learning gain $g = 0.01$) is applied for the 10, 15, 20, 25 node overlay networks. Similar convergence results as in Fig. 5.13 are shown for all 4 overlay networks.

Let the difference between $\bar{d}(n)$ and $\bar{d}^*(n)$ be the absolute error $\varepsilon(n)$, i.e.,

$$\varepsilon(n) = \bar{d}(n) - \bar{d}^*(n) \tag{5.12}$$

For overlay network size of 10, 15, 20 and 25, we compute their absolute errors and combine them in Fig. 5.15. It can be seen that the convergence speed remains similar although the overlay network size increases from 10 to 25. This figure also shows that the convergence can be very fast, which is about 5 seconds in this example. Note that the absolute error is driven down to around 10 ms. Our simulations are conducted at time-scale $\tau = 5$ ms, so this is effectively as good as we can hope to achieve. The remaining error can be thought of as noise, due to quantizing time in the fluid network simulation. Also note that larger overlay networks (20 or 25 nodes) converge at a slightly slower rate, but that in general, the size of the overlay network does not dramatically impact the time required to learn minimum delay paths.

End-to-end learning

There are four factors that may affect the learning speed: learning method, initialization method, network size and the learning gain. An example of the simulation results for end-to-end learning is shown in Fig. 5.16. In this example, we use end-to-end learning, geographical location aware initialization for a 15-node network. As can be seen, the initial RTT measurement for geographical location aware initialization in Fig. 5.16 is less than that for uniform initialization in Fig. 5.13. However, it takes around 60 seconds to converge to the optimal result, which is much longer than the case in Fig. 5.13. We also know that, for end-to-end learning, the probability distribution update only occurs at the source node of a probing packet, hence, it is almost the same as the probing rate. When the probing rate is fixed, the update rate will not be much affected by the network size. Then the

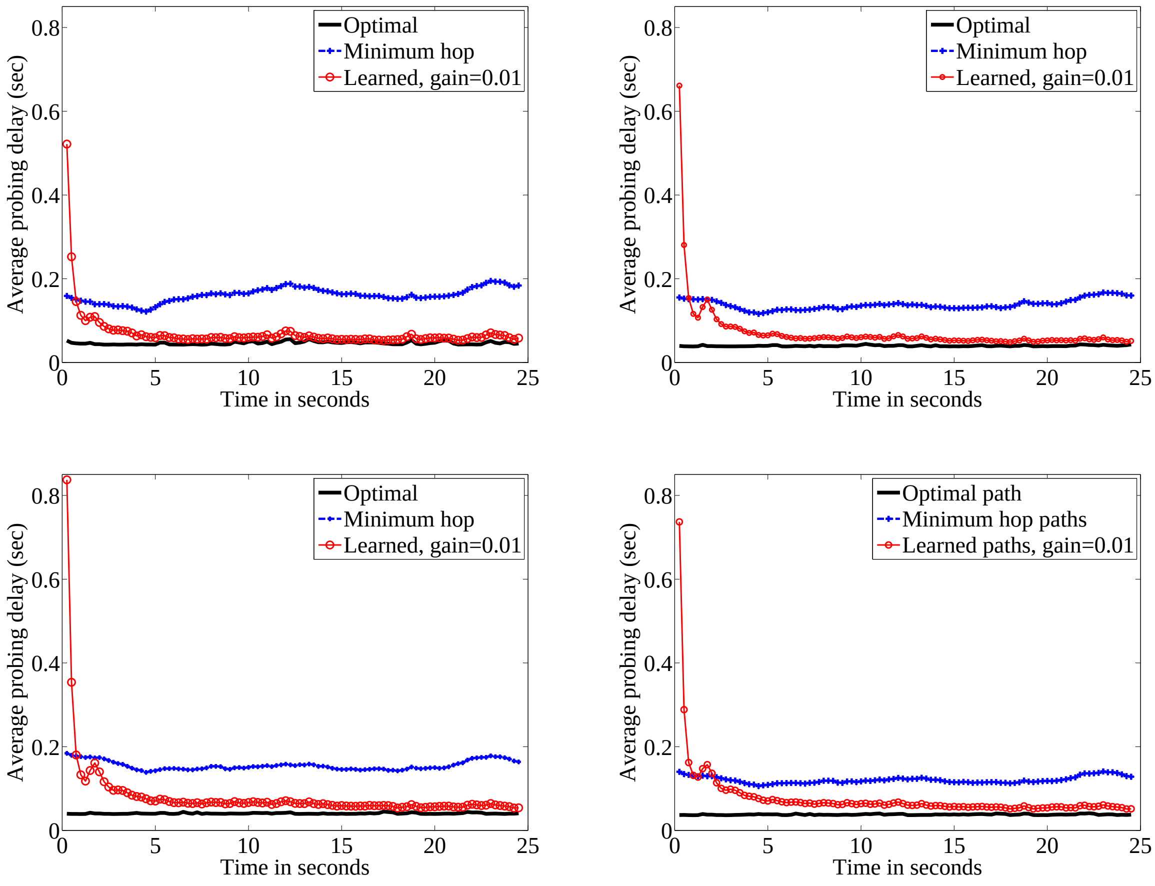


Fig. 5.14 Comparison of the average probing delay for three different routing: optimal (i.e. minimum delay path), minimum hop (i.e. shortest hop path), learned (i.e. the path determined by the distributed learning automata). The comparison is among 4 different overlay network sizes: 10 (**top left**), 15 (**top right**), 20 (**bottom left**) and 25 (**bottom right**) node overlay networks. The learning gain is 0.01. It can be seen that the learning speed is fast and similar for all the 4 figures. The small bump at around 2 seconds in the four figures is due to a temporary delay increase in the network environment, which causes a small purturbation to the learning algorithm; but the learning automata converge to the real optimal paths afterwards. This is similar to the bump on the left of Fig. 5.12.

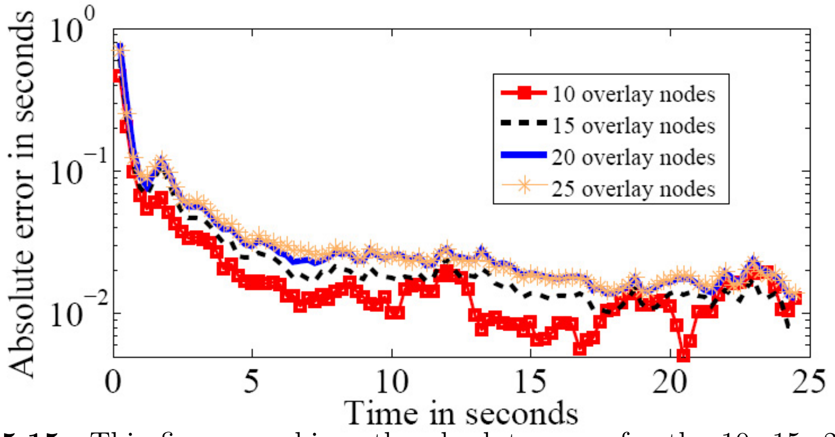


Fig. 5.15 This figure combines the absolute error for the 10, 15, 20 and 25 node overlay networks. The absolute error is the difference between the average probing delay on the paths determined by the learning automata and that on the minimum delay paths.

dominant factor for learning speed is the learning method and the learning gain. While for the same learning gain 0.001, end-to-end learning has much slower learning speed, as illustrated in Fig. 5.16, because it has lower probability distribution update rate.

5.5 Probing Overhead Analysis

This section analyzes the amount of probing traffic received at each node. It determines the probing overhead, as well as the probability distribution update rate of each learning automaton. It can be computed as follows. First, we define a routing matrix for each destination v in the overlay network, i.e. $\Pi^{(v)} = [\pi_{ij}^v]$, with $\pi_{ij}^v \in [0,1], i,j,v \in V$, $|V| = m$, denoting the probability that a probing packet $pp(i,v)$ is sent to a neighboring node j. Let the probing interval be **PROBE_INTERVAL**. Then the probing rate between each source i and destination v is $\lambda_i^v = \frac{1}{\textbf{PROBE\_INTERVAL}}$. Denote γ_i^v as the aggregate probing packet arrival rate at node i for destination v. Let $\underline{\lambda}^v$ be $\underline{\lambda}^v = [\lambda_1^v, ..., \lambda_m^v]^T$ and $\underline{\gamma}^v$ be $\underline{\gamma}^v = [\gamma_1^v, ..., \gamma_m^v]^T$. Then $\underline{\gamma}^v$ can be computed as [63]:

$$\begin{aligned} \underline{\gamma}^v &= [I - (\Pi^{(v)})^T]^{-1}\underline{\lambda}^v \\ &= \frac{1}{\textbf{PROBE\_INTERVAL}}[I - (\Pi^{(v)})^T]^{-1}(1 - [\delta_{1v}, \delta_{2v}, ..., \delta_{mv}])^T \end{aligned} \quad (5.13)$$

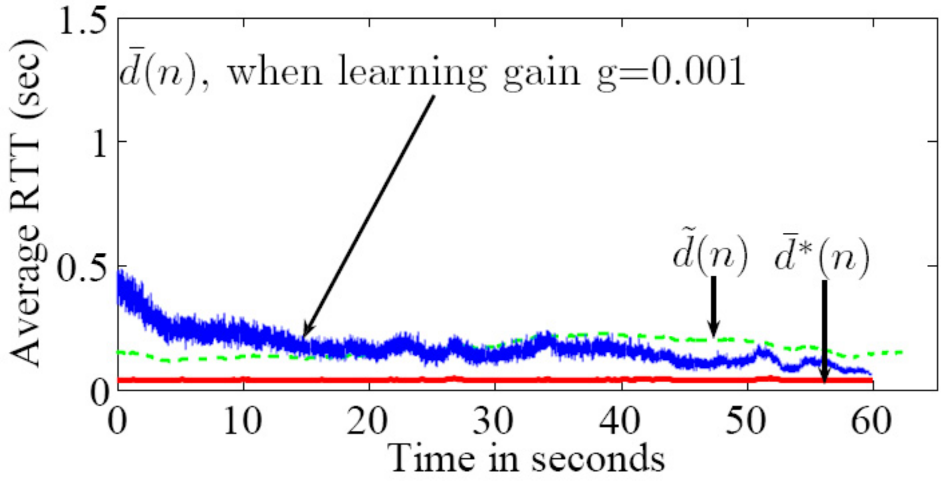


Fig. 5.16 Average RTT measured by the probing packets between all source-destination pairs, versus that on the minimum delay paths and that on the minimum hop paths for a 15-node overlay network. With geographical aware location initialization method and a gain of 0.001, the end-to-end learning is close to the optimal result.

where $\delta_{iv} = 1$ if $i = v$; otherwise, $\delta_{iv} = 0$.

5.5.1 Probing overhead for hop-by-hop learning

In hop-by-hop learning, the size of a probing packet grows along its forward path. As shown in Fig. 5.17, the length of the header of a probing packet is 28 bytes. The probing data is $13 + 9(N - 1)$ bytes long, where N is the number of hops that the probing packet has gone through. For example, if $N = 1$, the probing packet has just been sent to its first hop from its source node. If this first hop equals the destination, we do not record this hop; Otherwise, we record this hop and the time the probing packet is forwarded to a new hop, which will add 9 bytes to the probing data, and we set $N = 2$. Adding up the headers and the probing data, the probing packet is $B = 41 + 9(N - 1)$ bytes long.

For any $v \in V$, the expected probing path length N to a destination v is a function of the probing probability $\Pi^{(v)} = [\pi_{ij}^{v}]$. π_{ij}^{v} is the probability of sending a probing packet from node i via node j to destination node v. An example for computing the expected probing path length N is as follows. For a three-node network, $\Pi^{(1)}$ is the probing probability

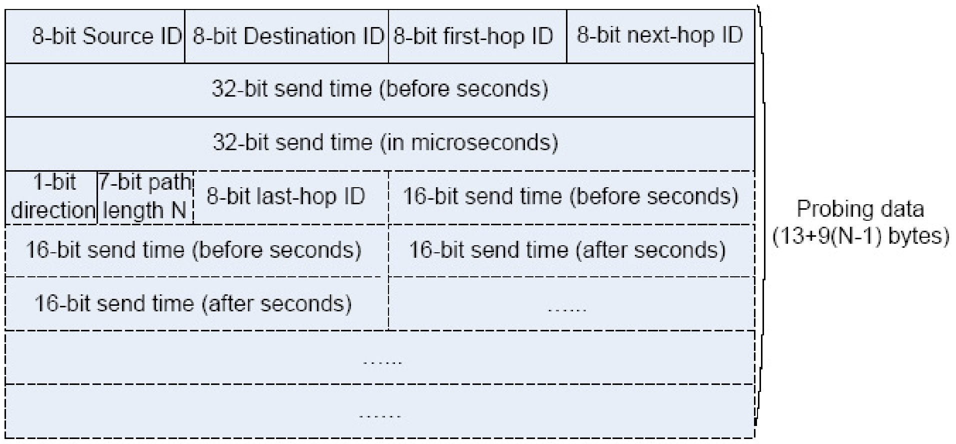

8-bit Source ID
8-bit Destination ID
8-bit first-hop ID
8-bit next-hop ID
32-bit send time (before seconds)
32-bit send time (in microseconds)
1-bit direction
7-bit path length N
8-bit last-hop ID
16-bit send time (before seconds)
16-bit send time (before seconds)
16-bit send time (after seconds)
16-bit send time (after seconds)
……
……
……
Probing data (13+9(N-1) bytes)

Fig. 5.17 The data format of a probing packet for hop-by-hop learning. The source ID and destination ID are identifiers for overlay nodes. The number of nodes can be extended if necessary. The 8-bit next-hop ID is the identifiers of the next-hop overlay node for this probing packet. The sending time field is 64 bits, which has a precision of microsecond. 1-bit direction is 0 if the probe is on the forward path to its destination; this field is 1 if the probe is on the backward path to its source. The maximum of the 7-bit overlay path N is $2^7 = 128$. The 8-bit last hop ID is the last hop overlay node that this probing packet has passed. The following 64-bit sending time is the time the probing packet is sent from the last hop overlay node. The content in dash lines grows as the probing packet travels through the network before reaching its destination and shrinks as it is sent back to its source.

matrix for destination node 1, as follows.

$$\Pi^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0.7 & 0.3 & 0 \end{pmatrix} \tag{5.14}$$

The data in the diagonal of $\Pi^{(1)}$ are all zeros because the probability of sending a probing packet from a node to itself is zero. The data in the first row of $\Pi^{(1)}$ are all zeros because the probabilities of sending a probing packet from the destination node, i.e. node 1 in this example, to other nodes are zeros. Similarly, the second row of $\Pi^{(2)}$ will be all zeros if the destination under consideration is node 2. Then, from $\Pi^{(1)}$, we can get

$$\Pi^{(1)^2} = \Pi^{(1)}\Pi^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ 0.35 & 0.15 & 0 \\ 0.15 & 0 & 0.15 \end{pmatrix}$$

The 1st column of $\Pi^{(1)^2}$ represents the probabilities of reaching destination 1 from node 1, 2 and 3 in two steps.

$$\Pi^{(1)^3} = \Pi^{(1)^2}\Pi^{(1)} \begin{pmatrix} 0 & 0 & 0 \\ 0.075 & 0 & 0.075 \\ 0.105 & 0.045 & 0 \end{pmatrix}$$

Similarly, the 1st column of $\Pi^{(1)^3}$ represents the probabilities of reaching the destination node 1 from node 1, 2 and 3 in three steps; and the 1st column of $\Pi^{(1)^k}$ represents the probability of reaching the destination node 1 from any source node in k steps. Thus, the expected number of steps from each source node to a destination node can be computed as follows.

Let

$$X^{(v)} = 1 \cdot \Pi^{(v)} + 2 \cdot \Pi^{(v)^2} + 3 \cdot \Pi^{(v)^3} + \ldots = \sum_{k=1}^{\infty} k \cdot \Pi^{(v)^k} \tag{5.15}$$

Then, the v^{th} column of $X^{(v)}$ contains the expected numbers of steps to destination v from all the source nodes. i.e., the expected probing path length from any source node i to the destination node v is given by $X^{(v)}_{iv}$. $X^{(v)}$ can be computed as follows.

$$X^{(v)} \cdot \Pi^{(v)} = {\Pi^{(v)}}^2 + 2 \cdot {\Pi^{(v)}}^3 + 3 \cdot {\Pi^{(v)}}^4 + \ldots = \sum_{k=1}^{\infty} k \cdot {\Pi^{(v)}}^{k+1} \tag{5.16}$$

$$X^{(v)} \cdot (\mathrm{I} - \Pi^{(v)}) = \Pi^{(v)} + {\Pi^{(v)}}^2 + {\Pi^{(v)}}^3 + \ldots = \sum_{k=1}^{\infty} {\Pi^{(v)}}^{k} \tag{5.17}$$

For a deadlock-free routing pattern $\Pi^{(v)}$, $(I - \Pi^{(v)})^{-1} = \sum_{k=0}^{\infty} {\Pi^{(v)}}^{k}$ [63]. Then, $X^{(v)} \cdot (\mathrm{I} - \Pi^{(v)}) = (I - \Pi^{(v)})^{-1} - I$. Thus,

$$X^{(v)} = \Big((I - \Pi^{(v)})^{-1} - I\Big)(\mathrm{I} - \Pi^{(v)})^{-1} \tag{5.18}$$

Hence, the expected probing packet length from any source node i to destination v is $B_{iv} = 41 + 9(X^{(v)}_{iv} - 1)$ bytes. Combining (5.13) and (5.18), the aggregate probing traffic at each node i for destination v denoted by ξ^v_i bytes/sec can be computed with (5.19). The factor 2 in (5.19) is because the probing packet is sent back to its origin for measuring RTT.

$$\xi^v_i = 2\Big(41 + 9(X^{(v)}_{iv} - 1)\Big)\gamma^v_i \tag{5.19}$$

From (5.13), (5.18) and (5.19), we can see that ξ^v_i is a function of $\Pi^{(v)}$ and the probing rate $\underline{\lambda}^v$. As $\Pi^{(v)}$ is updated with the cross-correlation learning algorithm in the active probing and learning process, ξ^v_i is also a function of time.

From the deduction above, we can also find the update rate for probability distribution $\underline{\pi}^v_i$ at node i for destination v. As each intermediate node on the forward path from source i to destination v receives a RTT measurement, thus, the probability distribution update rate of each node is the same as the aggregate probing packet arrival rate, i.e. γ^v_i.

Examples

For the example three node network with $\Pi^{(1)}$, as given in (5.14),

$$X^{(1)} = \Big((I - \Pi^{(1)})^{-1} - I\Big)(\mathrm{I} - \Pi^{(1)})^{-1} = \begin{pmatrix} 0 & 0 & 0 \\ 1.7647 & 0.4152 & 0.7958 \\ 1.5294 & 0.4775 & 0.4152 \end{pmatrix}$$

Then the expected probing path length from node 1, 2, 3 to destination 1 is $X_{11}^{(1)} = 0$, $X_{21}^{(1)} = 1.7647$ and $X_{31}^{(1)} = 1.5294$. It is reasonable that $X_{11}^{(1)} = 0$ since no probing packet is sent from node 1 to itself. It is also reasonable that $X_{21}^{(1)} > X_{31}^{(1)}$ since node 3 has a larger transition probability to node 1 than node 2 does in $\Pi^{(1)}$.

If **PROBE_INTERVAL**=5 ms, we have $\underline{\gamma}^1 = [0, 352.9412, 305.8824]^T$. Then, for the 3-node network, we have the probability distribution update rates at node 1, 2, and 3 are $[0, 352.9412, 305.8824]^T$ times per second. The expected aggregate probing traffic at node 2 and 3 for destination 1 are: $\xi_2^1 = 2\Big(41 + 9(X_{21}^{(1)} - 1)\Big)\gamma_2^1 = 33.7991$ kBps and $\xi_3^1 = 2\Big(41 + 9(X_{31}^{(1)} - 1)\Big)\gamma_3^1 = 27.9978$ kBps.

For the 10-node overlay network as in Fig. 5.13, the average probing traffic overhead can be computed from (5.13), (5.18) and (5.19). For comparison, the average probing and routing traffic overhead for path switching method (TaoPS) [46] and that for RON [48] with the same probing and routing update interval is also computed and plotted in Fig. 5.18. Note that in the Active Probing and Learning scheme (APL), similar for TaoPS, there is no routing update message required for routing decisions as in RON. Therefore, the routing traffic in APL and TaoPS is zero. As can be seen in Fig. 5.18, the average probing traffic for APL initializes with around 4 MBps, and quickly reduces to around 0.35 MBps, which is almost half of the probing and routing traffic in a same size RON (i.e., 0.65 MBps as mentioned in Section 2.3.2) and much lower than that of "TaoPS" (i.e., 1.1178 MBps as mentioned in Section 2.3.2).

5.5.2 Probing overhead for end-to-end learning

For end-to-end learning, a probing packet only has the 28 bytes IP and UDP headers, and the fixed size 13 byte probing data. The expected probing path length from any source i to destination v is the same as that in the hop-by-hop learning case, i.e. $X_{iv}^{(v)}$. The expected aggregate probing traffic ξ_i^v at each node i for destination v is:

$$\xi_i^v = 2 \cdot 41\gamma_i^v = 82\gamma_i^v. \tag{5.20}$$

The factor 2 is because the probing packet is sent back to its origin for measuring RTT.

For end-to-end learning, similarly, we can find the probability distribution update rate at node i for destination v. As only the source node of each probing packet receives a RTT

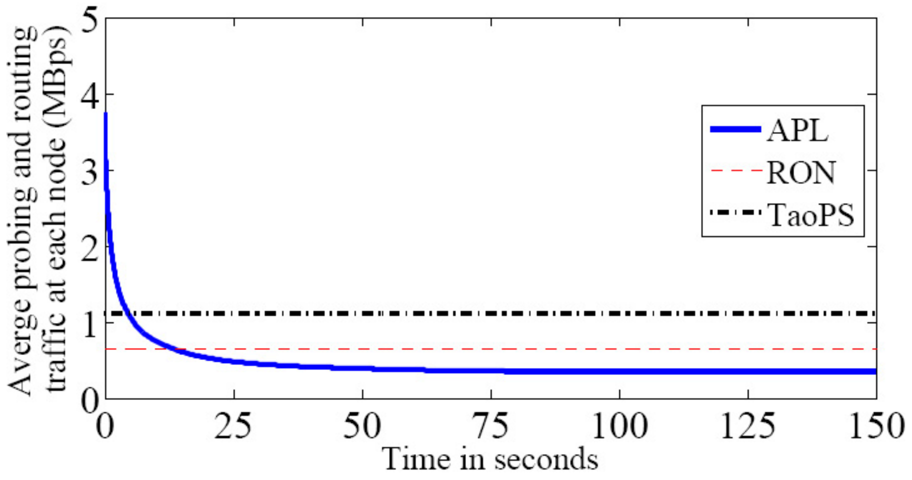


Fig. 5.18 Average probing and routing traffic received at each node. The average probing and routing traffic at each node represents the overhead for routing decisions in an overlay network. Note that in the Active Probing and Learning scheme (APL), there is no routing update message required for routing decisions as in RON, similar for TaoPS. Hence the routing traffic in APL and TaoPS is zero. The probing and routing update traffic for the centralized routing method of Chapter 4 is the same as that for RON. TaoPS has the most probing overhead because it has to probe the performance of each candidate path for each source-destination pair.

measurement, the probability distribution update rate for $\underline{\pi}_i^v$ is the same as the probing rate, i.e. $\underline{\lambda}_i^v = \frac{1}{\mathbf{PROBE\_INTERVAL}}$. For the 3-node network, we have the probability distribution update rates at node 1, 2, and 3 are $[0, 200, 200]^T$ times per second for the end-to-end learning, which are less than that for hop-by-hop learning. This explains why the learning speed of hop-by-hop learning is faster than that of end-to-end learning.

5.5.3 Remarks

Since the learning automata learn the minimum delay paths, all the probing traffic will concentrate on the minimum delay paths when the learning automata converge. Therefore, the average amount of probing traffic at each node depends only on the average minimum-delay-path length (in terms of the number of hops). This is because the probing traffic for each source-destination pair goes through its minimum delay path. Suppose the minimum-delay-path length between a source-destination pair s and d is h_{sd}. Then h_{sd} nodes receives the probing traffic from s to d. Thus, the average amount of probing traffic at each node is $\frac{\sum_{s,d} h_{sd}\lambda_s^d}{N}$ for a N-node overlay network. Note that $\frac{\sum_{s,d} h_{sd}}{N(N-1)}$ is the average minimum-delay-path length. Denote it as K. Then the average amount of probing traffic at each node when the learning automata converge is $K(N-1)\lambda_s^d = \frac{K(N-1)}{\mathbf{PROBE\_INTERVAL}}$.

Therefore, the scalability of the amount of probing traffic with network size depends on how the average minimum-delay-path length K scales with network size. Although we can not determine how K grows network size N, however, it is easy to understand that $K < \frac{N-1}{2}$ for $N > 2$ since the worst case is that the minimum delay path tree of the network forms a chain, for which the average path length is less than $\frac{N-1}{2}$. Therefore, we can say that K grows with less than $O(\frac{N-1}{2})$ and thus the probing traffic grows with less than $O(\frac{(N-1)^2}{2})$when the automata converge, which is less than those of RON and TaoPS (see Section 2.3, [48] and [46]). In fact, as the probing traffic focuses on the minimum delay paths when the automata converge, one may prefer to reduce the probing frequency to reduce the amount of probing traffic injected into the network when the learning automata converge.

When the automata haven't converged, e.g. during the intial stage, the amount of probing traffic can be simply computed from (5.19)(5.18) for uniform initialized learning automata. It can be expected that the initial probing traffic is large due to random loops, however, we can expect that it reduces very quickly (in around 5 seconds) as the learning algorithm converges, as illustrated in Fig. 5.14 and Fig. 5.18.

5.6 Convergence of the cross-correlation learning automata

In this section, we prove that the learning algorithm defined by (5.3) and (5.4) converges to the optimal solution given by (5.6). Our approach is to first find the Karush-Kuhn-Tucker (KKT) conditions [113] for the delay minimization problem, and then show that asymptotically, the learning algorithm satisfies the KKT conditions.

5.6.1 Proof of Convergence

The minimum delay routing problem for a commodity/user with source-destination pair (S, D) can be formulated as follows. The graph model for the overlay network is $G = (V, E)$, $V = \{1, ..., m\}$, $S, j, D \in V$. Assume the network is stationary during the active probing and learning process. Let $\Delta_{Sj}^D(t) = \Delta_{Sj} + \Delta_j^D(t)$ be the expected delay from node S to node D via node j at time t, Δ_{Sj} is the expected link delay from node S to node j and $\Delta_j^D(t)$ is the expected delay from node j to node D at time t. Let θ_{Sj}^D denote the probability for sending a probing packet from source S to destination D via j. Note that θ_{Sj}^D is different from $\pi_{Sj}^D(t)$ because θ_{Sj}^D is a scalar parameter while $\pi_{ij}^k(t)$ is a function of time t. Then the user optimization problem for the minimum end-to-end delay routing problem can be formulated as follows in (5.21).

$$\begin{aligned} \min_{\theta_{Sj}^D} \quad & \sum_j \theta_{Sj}^D \Delta_{Sj}^D \\ s.t. \quad & \sum_j \theta_{Sj}^D = 1, \\ & \theta_{Sj}^D \geq 0, \forall j. \end{aligned} \tag{5.21}$$

The KKT conditions for the user optimization problem in (5.21) are as follows. Let μ_S^{D*} be the multiplier for the constraint $\sum_j \theta_{Sj}^D - 1 = 0$ in the KKT condition. Then the KKT condition for $\theta_{Sj}^{D*}, j = 1, ..., m$, being the optimal solution to (5.21) is:

$$\begin{cases} \theta_{Sj}^{D*} \geq 0, \text{if } \Delta_{Sj}^D = \mu_S^{D*}; \\ \theta_{Sj}^{D*} = 0, \text{if } \Delta_{Sj}^D > \mu_S^{D*}. \end{cases} \tag{5.22}$$

where $\mu_S^{D*} = \sum_{u=1}^{m} \Delta_{Su}^{D} \theta_{Su}^{D*}$. Thus, the user optimal point satisfies:

$$\theta_{Sj}^{D*}(\Delta_{Sj}^{D} - \sum_{u=1}^{m} \Delta_{Su}^{D} \theta_{Su}^{D*}) = 0. \tag{5.23}$$

In the stochastic network environment, according to the Kushner's weak convergence method [30] and following the proof in Vázquez-Abad and Mason's work [114, 115], we can derive from the cross-correlation algorithm that as $t \to \infty$ and the learning gain goes to zero, $\lim_{t\to\infty} \pi_{Sj}^{D}(t) = \theta_{Sj}^{D*}$, where θ_{Sj}^{D*} satisfies the following equation:

$$\frac{\mathrm{d}\pi_{Sj}^{D}(t)}{\mathrm{d}t} = -\beta \frac{\pi_{Sj}^{D}(t)}{\mathrm{d}_{max}} (\Delta_{Sj}^{D}(t) - \sum_{u} \Delta_{Su}^{D}(t) \pi_{Su}^{D}(t)) \tag{5.24}$$

in which $\beta > 0$ corresponds to an update rate.

For θ_{Sj}^{D*} to be locally stable, it should satisfy $\frac{\mathrm{d}\pi_{Sj}^{D}(t)}{\mathrm{d}t}|_{\theta_{Sj}^{D*}} = 0$ according to the appendix (A.2), which is true given the KKT conditions in (5.23) and (5.22).

To show the solution is globally stable, let $M_S^D(t) = \sum_{j=1}^{m} \pi_{Sj}^{D}(t) \Delta_{Sj}^{D}$. $M_S^D(t)$ can be thought as the objective in (5.21) being a function of time t. From the cross-correlation learning algorithm, $\pi_{Sj}^{D}(t^+) = \pi_{Sj}^{D}(t) + gz(u,t)(\delta_{ju} - \pi_{Sj}^{D}(t))$, we can write:

$$M_S^D(t^+) - M_S^D(t) = -\sum_{j} \pi_{Sj}^{D}(t) \Big((\Delta_{Sj}^{D})^2 - (\sum_{j} \pi_{Sj}^{D}(t) \Delta_{Sj}^{D})^2 \Big) \tag{5.25}$$

i.e. $M_S^D(t^+) - M_S^D(t) \le 0$, since $\sum_j \pi_{Sj}^{D}(t) \Big((\Delta_{Sj}^{D})^2 - (\sum_j \pi_{Sj}^{D}(t) \Delta_{Sj}^{D})^2 \Big)$ equals the variance of Δ_{Sj}^{D}. Let $M(t) = \sum_S \sum_D M_S^D(t)$. Then $M(t)$ is monotonically decreasing with each update of $\underline{\pi}_S^D(t)$, $\forall S, D \in V$.

Thus, when the learning gain $g > 0$ is sufficiently small, the expected delay $M_S^D(t)$ keeps decreasing with time until $\pi_{Sj}^{D}(t) = \theta_{Sj}^{D*}$, where $M_S^D(t^+) - M_S^D(t) = 0$, and θ_{Sj}^{D*} minimizes $M_S^D(t)$ or equivalently the expected delay from node S to node D. In fact, since the user optimization problem in (5.21) is a linear programming problem, the optimal solution θ_{Sj}^{D*} is unique and stable if $\Delta_{Su}^{D}, u = 1, ..., m$, are distinct. If Δ_{Su}^{D} are not distinct, the optimal solution may be not unique, but all the optimal solutions will give the same optimal value for the objective function.

5.6.2 Practical Considerations

This section discusses practical concerns regarding applying the active probing and learning process for VoIP routing in a dynamic network environment.

When the cross-correlation learning algorithm converges to the optimal solution given in (5.6), the values of $\pi_{Sj}^{D}(t)$ cannot be changed any more, i.e. they are stuck in the current (local) optimum. For example, if at time t, $\pi_{Sj}^{D}(t) = 0$, then link (S, j) will be eliminated from being a candidate next-hop on the optimal path from S to D; or if $\pi_{Sj}^{D}(t) = 1$, then link (S, j) becomes the only candidate next-hop for that. In practice, we would not allow $\pi_{Sj}^{D}(t) = 0$ or $\pi_{Sj}^{D}(t) = 1$, so that it can adapt to network changes in a non-stationary network environment. Thus, we constrain all $\pi_{Sj}^{D} \geq \epsilon$, for a small $\epsilon > 0$, so that changes in network environment can be tracked over time.

We have proved in Section 5.6 that the learning automata can converge to the minimum delay paths. However, in a dynamic network environment, there exist situations when VoIP routing decisions have to be made before the learning automata converge. If voice calls are routed in the same way as that for the probing packets, random path selection before the learning automata converge, can lead to choosing different paths for voice packets of a single voice call, and thus results in out-of-order arrivals. In addition, fair quality provision for all users requires us to avoid the risk of choosing a bad random path[8] for some users. Thus, we need a mechanism to choose a fixed path for each new incoming voice call, which is detailed in Chapter 6.

Also note that the probing packets measure round-trip-time instead of one-way delay. This means that the minimum delay paths learned with the learning automata are in fact paths with minimum round-trip-time. The reason we measure round-trip-time is that in practice there is clock drift in computer systems which can lead to inaccurate one-way delay estimation. The other reason is that the conversational delay (i.e. the delay from the time one speaker says something till the time this speaker hears the response from the other side), which is highly related to the round-trip-time, is also very important for a voice call. If accurate one-way delay estimation is available, for example when Global Positioning System (GPS) is installed on all the overlay nodes, we can certainly use one-way delay as feedback $d_{S}^{D}(u)$ in (5.3) to the cross-correlation learning algorithm. Then the minimum delay paths learned with the cross-correlation learning algorithm will be paths

[8]Loops might still exists with very small probability, as we disallow $\pi_{Sj}^{D}(t) = 0$ or $\pi_{Sj}^{D}(t) = 1$

with minimum one-way delays.

5.7 Summary

In this chapter, we proposed a novel method to learn minimum delay paths for each source-destination pair in service overlay networks. Based on the cross-correlation learning algorithm, we proposed four active probing and learning strategies to learn the optimal paths, which are uniformly initialized hop-by-hop learning, geographical location aware initialized hop-by-hop learning, uniformly initialized end-to-end learning, and the geographical location aware initialized end-to-end learning. The performance of the proposed active probing and learning strategies is simulated in service overlay networks over a model of the AT&T backbone network. The simulation results show that the learning method converges to the minimum delay paths very quickly (around 5 seconds for hop-by-hop learning in a 10-node overlay network), and the convergence speed scales well with the overlay network size. We then analyzed the overhead for the hop-by-hop learning and end-to-end learning methods, and proved that the cross-correlation learning algorithm converge to the user equilibrium. At the end, we pointed out a practical concern that is treated in the next chapter.

Chapter 6

Online Distributed Diverse Routing for VoIP

6.1 Introduction

The previous chapter has shown that minimum delay paths can be learned with the proposed learning method. This chapter presents an online distributed diverse routing methods to route voice calls based on the parameters of the learning automata.

We have seen improved VoIP quality with path diversity in Chapter 4, where a novel centralized approach was proposed for VoIP routing in small-scale overlay networks. In this chapter, we propose a *distributed and scalable* approach to find the diverse paths to improve VoIP quality in service overlay networks. We demonstrate improved VoIP quality for single best path routing and diverse routing by answering the following two questions:
(1) How can we determine and track optimal diverse paths for VoIP routing in a distributed, scalable and efficient way?
(2) How can we detect and recover from a link failure quickly?

To answer these two questions, we apply the cross-correlation learning automata framework to guide a probing process that learns primary and secondary delay-optimal paths for VoIP path diversity; we present a novel link failure detection mechanism based on the states of the learning automata, which allows routes in our overlay to rapidly recover after link failures.

6.1.1 Chapter Structure

The rest of the chapter is organized as follows. Section 6.2 provides a formal problem statement. Section 6.3 presents the methodology for learning diverse paths. Section 6.4 presents a scheme for determining the primary and secondary optimal paths based on learning automata parameters. Section 6.5 presents a novel the method for link failure detection. Section 6.6 gives simulation results. Section 6.7 discusses the stability of the proposed routing method. Section 6.8 presents the implementation considerations for the routing method. Section 6.9 compares the overlay performance between RON, our method and BGP routing. Section 6.10 summarizes the chapter.

6.2 Diverse routing for VoIP in SONs

The aim of this work is to distributively find and track the two best disjoint paths so that VoIP quality can be improved with path diversity in overlay networks.

6.2.1 Problem Formulation

To formulate the optimal diverse routing problem, the overlay network is modeled as a graph $G = (V, E)$ as before, where $V = \{1, 2, .., m\}$ is the set of overlay nodes and E is the set of overlay links. Each overlay link is represented by two end nodes. It may consist of multiple physical links. An overlay path is represented by a set of overlay links.

Let $\mathcal{P}$ represent the set of all possible overlay paths from a source overlay node $S \in V$, to a destination overlay node $D \in V$. For any pair of overlay paths $p_i, p_k \in \mathcal{P}, p_i \neq p_k$, define $R(p_i, p_k)$ as the R-factor when voice packets are sent on both paths p_i and p_k. Then the R-factor $R(p_i, p_k)$ for the pair of paths can be evaluated based on the performance of the two paths. If the same voice packets are received from both paths, the network delay of this voice packet would be the minimum of the network delays on the two paths, and the loss rate would be the product of the loss rates on the two paths (assuming independence of the losses on p_i and p_k). Suppose the network delays on paths p_i and p_k are d_{p_i} and d_{p_k}, respectively, and the network loss rates on the two paths are l_{p_i} and l_{p_i}, respectively. Then, the end-to-end network delay d_{net} can be computed as the minimum of d_{p_i} and d_{p_k} when a voice packet is duplicated and received from both paths p_i and p_k.

In order to maximize the quality of a voice call by duplicating voice packets and sending them on diverse paths p_i and p_k, we need to find a pair of paths p_i and p_k that maximizes the R-factor $R(p_i, p_k)$. Then the optimal diverse path routing problem can be formulated as follows:

$$\begin{aligned} &\max R(p_i, p_k), \\ &s.t. \;\; p_i, p_k \in \mathcal{P}. \end{aligned} \tag{6.1}$$

6.2.2 Approximation Problem

The problem (6.1) is in fact an NP-hard problem (refer to Section 2.3.3 and [57–59] on this). A brute-force search solution to the problem (6.1) is to search all the possible pairs of disjoint paths, which can be exponentially large. Such a method is straightforward, but it is not scalable, as it is computationally expensive to calculate R-factor for all the possible disjoint pairs of paths in $\mathcal{P}$ for a large and dynamic network.

Thus, a solution, that requires no prior knowledge of the path performance characteristics, with only small computation and probing cost, adaptive to network dynamics, and scalable with network size, is the goal we aim for in this work. However, it is hard to find such a solution for (6.1) directly. Thus, we solve an approximation problem, as given below.

First, we differentiate the two paths $p_i, p_k \in \mathcal{P}$ in (6.1). We call p_i be a candidate for the primary optimal path and p_k a candidate for the secondary optimal path. The end-to-end path delay on p_i is expected to be less than that on p_k, i.e. $d_{p_i} \leq d_{p_k}$. The second path p_k is used to prevent voice calls from being dropped during link losses or failures on path p_i. We expect the performance on paths p_i and p_k to be uncorrelated when there is no joint link between p_i and p_k. Thus, we also try to minimize the number of joint overlay links[1] between p_i and p_k, i.e. $|p_i \cap p_k|$.

Then we convert problem (6.1) into the following two problems (6.2) and (6.3):

$$p_i^* = \arg\min_{p_i \in \mathcal{P}} d_{p_i} \tag{6.2}$$

[1]Completely disjoint paths cannot be guaranteed if we are considering overlay networks built in a single ISP network, since we only have control over routing at the overlay level and not actual router-level routing. Also note that zero link jointness may produce paths with large delays. Therefore, one may argue that it is better to find a set of secondary minimum delay paths and choose one path that has minimum link jointness with the primary path. However, it is also possible that the set of secondary minimum delay paths may share many links with the primary path. As we are looking for a distributed solution, we rely on the secondary learning automata to find a minimum delay path that shares as few overlay links with the primary path as possible.

$$\begin{cases} p_k^* = \arg\min_{p_k \in \mathcal{P}'} d_{p_k} \\ \mathcal{P}' = \{p_k \in \mathcal{P}, |p_k \cap p_i^*| = \min_{p_k \in \mathcal{P}} |p_k \cap p_i^*|\} \end{cases} \tag{6.3}$$

Problems (6.2) and (6.3) are an approximation of the original problem (6.1). The benefit of such an approximation is that it is easier to estimate additive delay than to estimate R-factor for a path[2], especially for the adaptive learning algorithm, as used in this work. The effectiveness of the approximation is further illustrated in the simulation results presented in Section 6.6.

Solving problem (6.2) is equivalent to determining the primary optimal path $p_i^* \in \mathcal{P}$ that minimizes end-to-end delay from source S to destination D. Then, we solve problem (6.3) to determine the secondary optimal path $p_k^* \in \mathcal{P}'$. p_k^* is a path with secondary minimum end-to-end delay from S to D and with a minimum number of joint overlay links with p_i^*. We cannot assume that two distinct overlay links do not share physical links without knowing the router-level topology and policy. Hence, for overlay routing control, it is only feasible to control the disjointness at the overlay link level.

6.3 Methodology for Determining the Optimal Diverse Paths

To get a distributed and scalable diverse routing solution, the active probing and learning automata described in the previous chapter is implemented at each overlay node, as illustrated in Fig. 6.1. It consists of the necessary components for making local decisions on the primary and the secondary optimal next hop nodes for VoIP calls. As illustrated in Fig. 6.1, the probing layer is responsible for probing all the candidate pairs of next-hop nodes. The VoIP routing service layer makes decisions on the current primary and secondary optimal next-hop nodes for each new incoming call.

To determine the primary and secondary optimal paths for each source-destination pair, i.e. to solve the problems (6.2) and (6.3), the following two steps should be followed.

[2]Note that it is hard to guarantee that paths with minimum end-to-end delays always maximize the R-factor, but mostly we would expect to see better VoIP quality on paths with low end-to-end delays. We have not taken loss into consideration in the problem formulation because diverse paths are usually able to reduce losses when the performance of the two diverse paths are uncorrelated, which is why we desire the primary and the secondary paths to be disjoint.

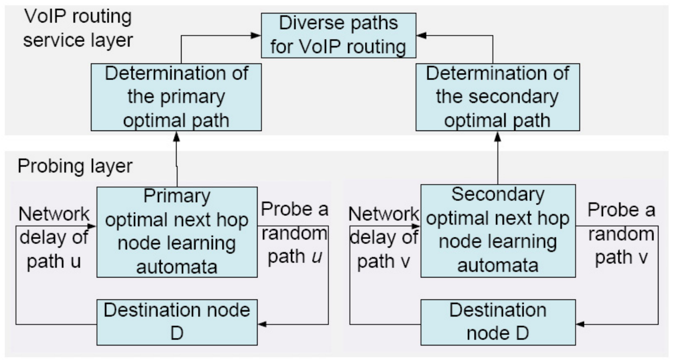

VoIP routing service layer
Diverse paths for VoIP routing
Determination of the primary optimal path
Determination of the secondary optimal path
Probing layer
Network delay of path u
Primary optimal next hop node learning automata
Probe a random path *u*
Destination node D
Network delay of path v
Secondary optimal next hop node learning automata
Probe a random path v
Destination node D

Fig. 6.1 Components for learning the primary-optimal and secondary-optimal next-hop nodes for VoIP routing. An end-to-end route can be set up for VoIP calls based on the local primary-optimal and secondary-optimal next hop decisions during the call session initialization phase.

- Step 1:
  Each node S probes all its neighbors j to learn if the link (S, j) is on its minimum delay path from S to a destination D.

- Step 2:
  The minimum delay path from a node S to D is determined based on the primary optimal next-hop or secondary optimal next-hop node decisions at all nodes.

Both steps are necessary for determining the primary optimal path p_i^* or the secondary optimal path p_k^*. However, there are still some differences in determining p_i^* and p_k^*, as follows.

- To find the primary optimal path p_i^*, i.e. to solve problem (6.2), we need to consider all $p_i \in \mathcal{P}$ so that the minimum delay path found is a global minimum solution.

- To find the secondary optimal path p_k^*, i.e. to solve problem (6.3), we must find a minimum delay path p_k^*, which has the minimum number of joint overlay links with p_i^*.

The next section presents how to determine the primary and the secondary optimal paths. The detail on how to efficiently search all $p_i \in \mathcal{P}$ to determine the primary optimal path is presented in section 6.4.1. Section 6.4.2 gives detail on how to determine the secondary optimal path.

6.4 Determining the Primary and Secondary Optimal Paths

This section presents the detail of the VoIP routing service layer in Fig. 5.1. As discussed in the last chapter, assuming that the network is stationary, the solutions to problems (6.2) and (6.3) are obtained when the primary and the secondary optimal next hop learning automata converge. However, in a *dynamic* network environment, we are more concerned with how to *track* the primary optimal and the secondary optimal paths, and *always* choose the best possible pair of paths for new incoming voice calls. This is a more challenging problem than the problems in (6.2) and (6.3) for a stationary network environment.

To solve it in a dynamic network environment, the most important and practical question to answer is which pair of paths we should select when the learning automata have not yet converged. That is to say, a rule for predicting the primary optimal path p_i^* and the secondary optimal path p_k^* has to be determined based on the current probability distributions $\underline{\pi}_S^D(t)$, $S, D \in V$.

In this section, we present the proposed solution to problems (6.2) and (6.3), i.e. tracking the best diverse paths. Section 6.4.1 shows how to determine the instantaneous *primary* optimal path p_i^* for incoming VoIP calls. Section 6.4.2 shows how to determine the instantaneous *secondary* optimal path p_k^* for incoming VoIP calls.

6.4.1 Determine and Track the Primary Optimal Paths for Routing VoIP Calls

To determine and track the primary optimal paths, as illustrated in Fig. 6.1, active probing and learning presented in Section 5.3 is applied to track the probability of a potential next-hop node being the optimal next hop at any time instant t. Let the probability distribution at node S for destination D at time t be $\underline{\pi}^{(p)}{}_S^D(t)$. The superscript (p) of $\underline{\pi}^{(p)}{}_S^D(t)$ indicates that $\underline{\pi}^{(p)}{}_S^D(t)$ is used for determining the primary optimal next-hop node of S for destination D.

In a stationary network environment, we can run the learning automata for a sufficiently long period of time. Then the primary optimal next-hop for each source-destination pair is determined based on the maximum of the probability distribution $\underline{\pi}^{(p)}{}_S^D(t)$. In fact, as implied by the updating scheme in (5.4), the value of $\pi^{(p)}{}_{Sj}^D(t)$ gives some indication on the ranking of the link quality. In a dynamic network environment, the optimal next hop j^* for a source node can vary with time, i.e. it is a function of time and the network dynamics. When $\underline{\pi}^{(p)}{}_{S}^{D}(t)$ is updated with (5.4), Algorithm 5 is used to track the primary optimal path. Thus, for an incoming voice call session initiated at time t, with source S and destination D and candidate next hop node set $U = \{1, 2, ..., m\} \setminus \{S\}$, the primary optimal next hop node for node S is given by (6.4):

$$j^* = \arg\max_{j \in U} \pi^{(p)}{}_{Sj}^{D}(t) \tag{6.4}$$

Then similarly, the primary optimal next hop node for node j^* with destination D is $u^* = \arg\max_u \pi^{(p)}{}_{j^*u}^{D}(t)$. In this way, the primary optimal path from source S to destination D can be determined.

Note that in (6.2), the candidate path set $\mathcal{P}$ for learning p_i^* includes all the possible paths between that source-destination pair. That is to say, to solve problem (6.2), the primary optimal next-hop node learning process has to be initialized with uniform initialization, given in Section 5.3.4. As mentioned before, such initialization introduces undesirable loops before final convergence. However, loops can be avoided by adding an additional loop check in the primary optimal path determination algorithm. Whenever a loop is found, the primary optimal path is reset to the default path, e.g. the shortest hop path, between that source-destination pair, or the path previously used. The algorithm for determining the primary optimal path for a VoIP call is shown in Algorithm 5. As Algorithm 5 runs at each overlay node *independently and distributedly*, it is a *scalable* solution for problem (6.2).

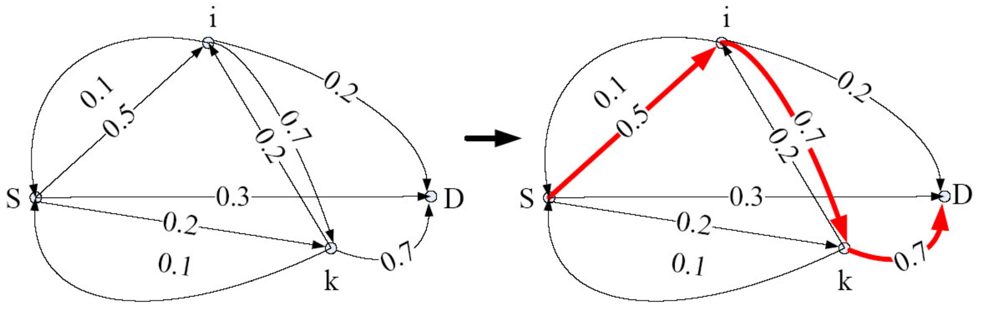


Fig. 6.2 Illustration of the primary-optimal path determination: scenario 1. The left figure shows the network with probabilities $\pi^{(p)}{}_{sj}^{D}(t)$, labeled on each corresponding link (s, j), $s, j \in \{S, i, k\}$. The thick arrows in the right figure shows the primary optimal paths from node S, i, k to destination D determined by the algorithm in Algorithm 5.

```
Require: Source-destination pair $(S, D)$, current time $t$
Ensure: The primary optimal path $p_i^*$ for an incoming voice call at time $t$.
 1: Set the last hop node $u = S$.
 2: while $u \neq D$ do
 3:     Set the primary optimal next-hop:
        $j^* = \arg\max_j \pi^{(p)}{}_{uj}^{D}(t)$
 4:     if Node $j^*$ already used in $p_i^*$, i.e. there will be a loop if $j^*$ is used then
 5:         $j^* = D$, $p_i^* = \{(S, D)\}$, Return.
 6:     else
 7:         $p_i^* = p_i^* \cup \{(u, j^*)\}$.
 8:         $u = j^*$.
 9:     end if
10: end while
```

Algorithm 5: Primary optimal path determination algorithm

An example of the primary optimal path determination process is illustrated in Fig. 6.2. The left figure shows a network with nodes $\{S, i, k, D\}$, and the probability distributions $\underline{\pi}^{(p)}{}_{v}^{D}(t)$ at each node v, $v \in \{S, i, k\}$, for a certain destination D at time t. For instance, $\underline{\pi}^{(p)}{}_{i}^{D}(t) = [0.1, 0, 0.7, 0.2]$. Thus the primary optimal next hop of node i is k, since $k = \arg\max_j \pi^{(p)}{}_{ij}^{D}(t)$. Applying Algorithm 5, the primary optimal path from each node v, $v \in \{S, i, k\}$, to destination D is found, as illustrated by the thick arrows on the right of Fig. 6.2.

6.4.2 Determine and Track the Secondary Optimal Paths for Routing VoIP Calls

Once the primary optimal path is determined, the secondary optimal path that has minimal overlap with the primary optimal path can also be determined based on the parameter of the learning automata.

The active probing and learning presented in Section 5.3 is also applied here for learning the secondary optimal paths. Let the probability distribution at node S for destination D at time t be $\underline{\pi}^{(s)}{}_{S}^{D}(t)$. The superscript (s) of $\underline{\pi}^{(s)}{}_{S}^{D}(t)$ indicates $\underline{\pi}^{(s)}{}_{S}^{D}(t)$ is used to determine the *secondary* optimal next-hop node of S for destination D.

Ideally, the primary optimal next hop node j^* should not be a candidate for the secondary optimal next-hop node. By setting the probability $\pi^{(s)}{}_{Sj^*}^{D}$ to be zero, the corresponding link (S, j^*) is then removed from the original graph G.

To satisfy the maximum disjointness constraint for p_k^* and p_i^*, as required in (6.3), where p_k^* and p_i^* are the secondary and the primary optimal path from S to D, respectively, we set up the following procedure. If link (S, j^*) is on the primary optimal path from S to D at time t, we divide the value of ${\pi^{(s)}}_{Sj^*}^{D}(t)$ into all other ${\pi^{(s)}}_{Sj}^{D}(t)$, $\forall j \in \mathcal{J}$, where $\mathcal{J}$ represents the set of candidate nodes for the secondary optimal next-hop, and we set the probability ${\pi^{(s)}}_{Sj^*}^{D}(t) = 0$. Then with the new distribution ${\underline{\pi}^{(s)}}_{S}^{D}(t)$, the secondary optimal next hop at node S for destination D is:

$$j'^* = \arg\max_{j' \in \mathcal{J}} {\pi^{(s)}}_{Sj'}^{D}(t) \tag{6.5}$$

With the above secondary optimal next-hop selection procedure, as detailed in Algorithm 6, the secondary optimal path can be decided.

```
Require: Source-destination node pair $(S, D)$
Ensure: The secondary optimal path $p_k^*$ for an incoming voice call at time $t$.
1: Set last hop node $u = S$;
2: while $u \neq D$ do
3:     Set $j' = j^*$, where $j^*$=the primary optimal next-hop of $u$
4:     if There exists a node $j' \neq j^*$ such that ${\pi^{(s)}}_{uj'}^{D}(t) > 0$ then
5:         Divide the value of ${\pi^{(s)}}_{uj^*}^{D}(t)$ into all the other positive ${\pi^{(s)}}_{uj'}^{D}(t)$ to keep
           $\sum_j {\pi^{(s)}}_{uj}^{D}(t) = 1$, i.e. ${\pi^{(s)}}_{uj'}^{D}(t) = {\pi^{(s)}}_{uj'}^{D}(t) + \frac{{\pi^{(s)}}_{uj^*}^{D}(t)}{|J'|}$, for each $j \in J'$,
           $J' = \{j | {\pi^{(s)}}_{uj'}^{D}(t) > 0\}$.
6:         Set ${\pi^{(s)}}_{uj^*}^{D}(t) = 0$;
7:         $j'^* = \arg\max_{j'} {\pi^{(s)}}_{uj'}^{D}(t)$;
8:         $p_k^* = p_k^* \cup \{(u, j'^*)\}$,$u = j'^*$;
9:     end if
10: end while
```

Algorithm 6: Secondary optimal path determination algorithm

As we are striving for a *scalable and distributed* solution to problem (6.3), we assume that there is no communication between different overlay nodes on their optimal next hop nodes. Thus, the secondary optimal next-hop node decision at node S for destination D is made *locally* and *independently* based solely on the local probability distributions ${\underline{\pi}^{(p)}}_{S}^{D}(t)$ and ${\underline{\pi}^{(s)}}_{S}^{D}(t)$. However, there is one thing to note about the initialization of ${\underline{\pi}^{(s)}}_{S}^{D}(t)$. In (6.3), in order to search all the possible paths in $\mathcal{P}$ that are disjoint with p_i^*, ${\underline{\pi}^{(s)}}_{S}^{D}(t)$ should be uniformly initialized. For the scenario of Fig. 6.2, it is easy to show how the method in Algorithm 6 can find secondary optimal paths that are completely disjoint, whether $\underline{\pi}^{(s)}{}_S^D(t)$ is initialized with uniform initialization or geographical-location-aware initialization. With the primary optimal paths being determined, as in the example of Fig 6.2, those links on the primary optimal paths are removed from the original graph when we chose the secondary optimal paths. Then we set $\pi^{(s)}{}_{vj^*}^D(t) = 0$ if j^* is the primary optimal next-hop of node v. Then the probability distribution parameters $\pi^{(s)}{}_{vj}^D(t)$ are re-normalized and labeled on each link (v, j), as illustrated in Fig. 6.3, with $\pi^{(s)}{}_{vj}^D(t)$ adopts uniform initialization in the left figure and geographical-location aware initialization in the right figure.

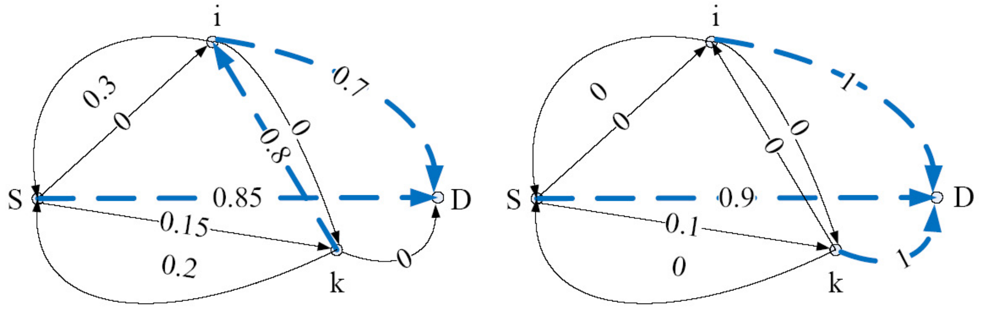


Fig. 6.3 Illustration of the secondary-optimal path determination: scenario 1. The left figure shows the determined secondary optimal path for uniformly initialized secondary optimal learning automata; The right one shows the determined secondary optimal path for geographical-location aware initialized secondary optimal learning automata. The thick dashed arrows show the secondary optimal paths from source S, i, k to destination D determined with the algorithm in Algorithm 6.

Note that, at any time, the values of $\underline{\pi}^{(s)}{}_v^D(t)$ depend on the network dynamics and the setting of the secondary optimal next-hop learning automata. It is not necessary that they are the same as those of $\underline{\pi}^{(p)}{}_v^D(t)$ and the values of $\underline{\pi}^{(s)}{}_v^D(t)$ can also differ for different initialization methods, which is also indicated in the values labeled on each link of Fig. 6.2, the left and the right side of Fig. 6.3. For example, in Fig. 6.2, at node k, $\pi^{(p)}{}_{kS}^D(t) = 0.1$; on the left side of Fig. 6.3, $\pi^{(s)}{}_{kS}^D(t) = 0.2$; while on the right side of Fig. 6.3, $\pi^{(s)}{}_{kS}^D(t) = 0$ due to geographical-location aware initialization.

If uniform initialization is used for there are cases when no completely disjoint secondary optimal path can be found, as shown in the example of Fig. 6.4. The left graph shows the network with probability distributions $\pi^{(p)}{}_{vj}^D(t)$ on each link (v, j). The right graph shows

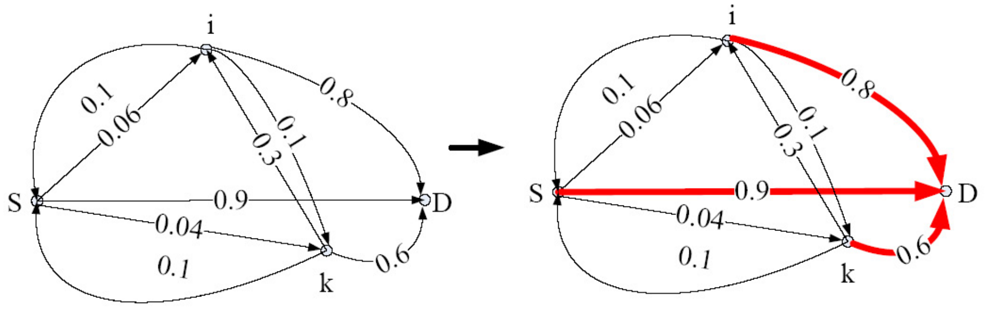


Fig. 6.4 Illustration of the primary-optimal path determination: scenario 2. The left graph shows the network with probabilities ${\pi^{(p)}}^D_{sj}(t)$, $s, j \in \{S, i, k\}$, labeled on each corresponding link (s, j). The thick arrows on the right side show the primary optimal paths from nodes S, i, k to destination D, which are determined with the algorithm in Algorithm 5.

the primary optimal path determined by Algorithm 5, as in the example of Fig. 6.2. Once the primary optimal paths are determined, the secondary optimal paths can be determined. However, if ${\underline{\pi}^{(s)}}^D_v(t)$ are uniformly initialized, no secondary optimal path can be found, as illustrated in the left graph of Fig. 6.5, as all the direct paths from v, $v \in \{S, i, k\}$ to destination D are disallowed for disjointness, i.e. ${\pi^{(s)}}^D_{vD}(t) = 0$.

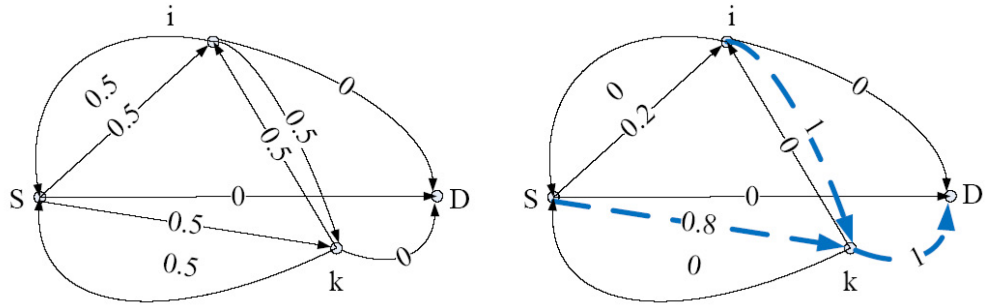


Fig. 6.5 Illustration of the secondary optimal path determination for the scenario in Fig. 6.4. The probabilities ${\pi^{(s)}}^D_{vj}(t)$, $v, j \in \{S, i, k\}$, are labeled on each corresponding link (v, j) in both the left and right figures. On the left figure, all distributions ${\underline{\pi}^{(s)}}^D_v(t)$ are uniformly initialized. No secondary optimal path can be determined by the method given in Algorithm 6. While on the right, all distributions ${\underline{\pi}^{(s)}}^D_v(t)$ are initialized with geographical-location-aware initialization. The thick dashed arrows show the secondary optimal paths from source S, i, k to destination D determined with Algorithm 6.

In this case, we use geographical-location-aware initialization for all distributions ${\underline{\pi}^{(s)}}^D_v(t)$, which means that the set of candidate next hop nodes for the secondary optimal next hop determination only includes those closer to destination D than v. Then Algorithm. 6 can find the secondary optimal paths, which are minimally joint with the primary optimal paths, as shown by the thick dashed arrows on the right of Fig. 6.5.

Thus, combining Algorithm 5 and Algorithm 6, we have the diverse routing approach for VoIP calls, implemented in overlay networks as in Fig. 6.1. Next, I am going to show how to detect link failures to improve the robustness of the proposed routing method. Simulations of the diverse routing method and the link failure detection approach are presented in Section 6.6.

6.5 Resiliency against Link Failures

We have presented the work on determining diverse paths based on the parameters of the learning automata. This section presents a novel link failure detection method for enhancing resiliency against link failures.

When a link fails, all the voice packets on it are dropped. Thus, to improve the robustness of Algorithm 5 and Algorithm 6, we consider how to detect overlay link failures as early as possible to reduce the number of lost calls. The primary and the secondary optimal next-hop learning embedded in Algorithm 5 and Algorithm 6 are able to detect a link failure after the probability of choosing the failed link drops below that of another choice. To improve this, we propose to detect link failures based on the change of the probability distribution $\underline{\pi}_S^D(t)$.

6.5.1 Link failure detection

Suppose at time t, the primary optimal next-hop of node S for destination D is j^*, i.e. $j^* = \arg\max_j \pi^{(p)}{}_{Sj}^{D}(t)$.

Proposition 6.5.1. *For* $n(\epsilon) = \lceil log(\epsilon)/log(1 - \pi^{(p)}{}_{Sj^*}^{D}(t))\rceil$*, if the sequence* $\{\pi^{(p)}{}_{Sj^*}^{D}(t - n(\epsilon)+1), ..., \pi^{(p)}{}_{Sj^*}^{D}(t)\}$ *is monotonically decreasing, a link failure can be declared with* $(1-\epsilon)$ *confidence.*

Proof. As defined in Section 5.3, the probability that node j^* is probed at time t is $\pi^{(p)}{}_{Sj^*}^{D}(t)$. Then, the probability that node j^* was not chosen in the past consecutive n time steps $f(n) = \prod_{\tau=t-n+1}^{t}(1 - \pi^{(p)}{}_{Sj^*}^{D}(\tau))$. If node j^* was probed and rewarded positively during the last n time steps, at least one increase should be observed in the sequence $L = \{\pi^{(p)}{}_{Sj^*}^{D}(t - n + 1), ..., \pi^{(p)}{}_{Sj^*}^{D}(t)\}$. However, if a physical link between j^* and destination D fails, there would be no positive increase observed in $L = \{\pi^{(p)}{}_{Sj^*}^{D}(t-n+1), ..., \pi^{(p)}{}_{Sj^*}^{D}(t)\}$, because no probing packet returned. In this case, $\pi^{(p)}{}_{Sj^*}^{D}(t-n+1) > ... > \pi^{(p)}{}_{Sj^*}^{D}(t)$. Then $\hat{f}(n) = (1 - \pi^{(p)}{}_{Sj^*}^{D}(t))^n \geq f(n)$. Thus, for a sufficiently small $\epsilon > 0$, $n = n(\epsilon) = \lceil log(\epsilon)/log(1 - \pi^{(p)}{}_{Sj^*}^{D}(t))\rceil$ guarantees $f(n) \leq \hat{f}(n) \leq \epsilon$. Or equivalently, the probability that sequence L should have at least one positive increase is $1 - f(n) > 1 - \epsilon$ if there is no link failure on (j^*, D). Therefore, no positive increase in sequence L indicates a link failure, with $(1 - \epsilon)$ confidence. Otherwise, if there is positive increase in the sequence L, this is not a necessary condition for link failure, thus, no failure alarm will be reported. □

A link failure event is defined as an event when the condition in *Proposition 6.5.1* is satisfied. According to Proposition 6.5.1, for a sufficiently small $\epsilon > 0$, a link failure event can be declared with confidence $(1-\epsilon)$ when sequence $L = \{\pi^{(p)}{}_{Sj^*}^{D}(t-n+1), ..., \pi^{(p)}{}_{Sj^*}^{D}(t)\}$ shows consecutive decrease. This method can be applied to link failure detection for both the primary and the secondary optimal next-hop node decision. As they are similar, we only show the link failure detection algorithm for primary optimal next-hop node decisions in Algorithm 7. In Algorithm 7, the variable *count* is used to count the number of consecutive link failure events. If such link failure events happen consecutively for N times, a link failure alarm is reported, i.e. we set $FailAlarm(S, D, j^*) = 1$. The value of ϵ and N can be adjusted to trade off the link failure detection time and the probability of a false alarm. The link failure detection method can be run after each update with (5.4) or less frequently, for example per 1 second interval if a gap of 2 seconds is tolerable[3].

Note that we desire a completely distributed link failure detection method, thus we do not consider propagating link failure information across the network, because it requires additional routing update overhead as RON does [48], and it is not necessarily useful since link failures can be quickly detected, recovered and avoided by re-routing voice calls on that failed link.

[3]2 seconds is the time taken by voice network signalling protocols to release a connection if there is a failure and so it is an important objective to meet to avoid dropping connections when a failure occurs, since if we recover in less than 2 seconds the user will only experience a click but the session continues [29].

```
Require: Source-destination node pair $(S, D)$, $\pi^{(p)}{}_{Sj^*}^{D}(t)$, $\epsilon$, $L$, $count$.
Ensure: Link Failure Alarm $FailAlarm$.
 1: $n(\epsilon) = log(\epsilon)/log(1 - \pi^{(p)}{}_{Sj^*}^{D}(t))$
 2: if Sequence $L = \{\pi^{(p)}{}_{Sj^*}^{D}(t - n(\epsilon) + 1), ..., \pi^{(p)}{}_{Sj^*}^{D}(t)\}$ shows consecutive decreases, i.e.
    $\pi^{(p)}{}_{Sj^*}^{D}(t - n(\epsilon) + 1) > ... > \pi^{(p)}{}_{Sj^*}^{D}(t)$ then
 3:     $count = count + 1$
 4: else
 5:     $count = 0$
 6: end if
 7: if $count >= N$ then
 8:     $FailAlarm(S, D, j^*) = 1$
        //A link failure alarm is reported.
 9: else
10:     $FailAlarm(S, D, j^*) = 0$
        //No link failure.
11: end if
```

Algorithm 7: Link Failure Detection Algorithm

6.5.2 Modification to the Primary and Secondary Optimal Path Determination

To apply the link failure detection Algorithm 7 into the primary and secondary optimal path determination algorithms, we need to add a rule to recover from a link failure when it is detected, i.e. to determine which link should replace the failed link. Note that, although the probability of two physical links failing at the same time is small, one cannot assume the probability of two overlay links failing at the same time is equally small, since two overlay links may share the same failed physical link. Hence, the new optimal next hop must be without link failure alarm and with as high $\pi^{(p)}{}_{Sj}^{D}(t)$ as possible. In other words, the rule for link failure recovery is:

- the new primary optimal next hop j must satisfy $FailAlarm(S, D, j) = 0$;
- j should have as high $\pi^{(p)}{}_{Sj}^{D}(t)$ as possible, since the probability $\pi^{(p)}{}_{Sj}^{D}(t)$ implicitly indicates the ranking of path quality through next-hop node j.

For the secondary optimal path determination, modification to Algorithm 6 is also needed to detect and recover from link failures, and to maintain link-disjointness if possible.

Once a link failure is detected with Algorithm 7 for the secondary optimal path, the link failure recovery rule is as follows.

- Choose the secondary optimal next hop to be the one without link failure alarm, with as high ${\pi^{(s)}}_{ij}^{k}(t)$ as possible and disjoint with the primary optimal next hop if possible.
- To be disjoint with the primary optimal next hop, whenever the primary optimal next hop changes, the secondary optimal next hop has to be changed so that the primary optimal next hop j^* is not equal to the secondary optimal hop j'^*. There are only two cases where j^* can be equal to j'^*, i.e. when all other choices except j^* are not available or have link failures.

Examples

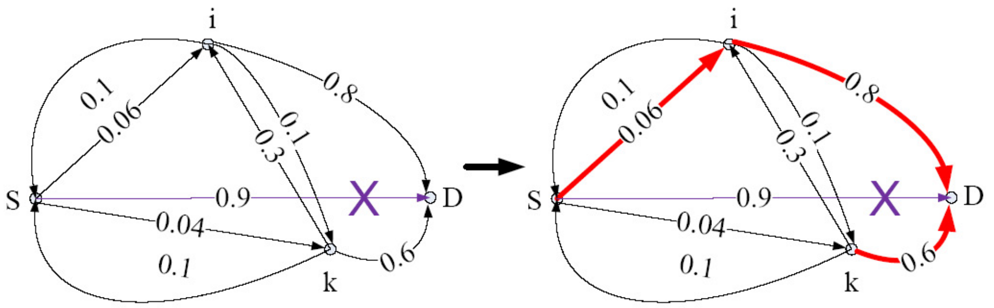


Fig. 6.6 Illustration of the primary optimal path determination when a link (S,D) fails. The left graph shows the network with probabilities ${\pi^{(p)}}_{vj}^{D}(t)$, $v,j \in \{S,i,k\}$, labeled on each corresponding link (v,j). The thick arrows on the right show the new primary optimal paths from S,i,k to destination D after the link failure on (S,D) is detected.

An example for link failure recovery for primary optimal path determination is illustrated in Fig. 6.6. The left graph of Fig. 6.6 shows the network with probabilities ${\pi^{(p)}}_{vj}^{D}(t))$, $v,j \in \{S,i,k\}$, labeled on each corresponding link (v,j). It also shows the detected link failure on link (S,D). The thick arrows on the right side of Fig. 6.6 show the new primary optimal paths from S,i,k to destination D after the link failure on link (S,D) is detected and recovered. The failed link (S,D) is detected by Algorithm 7, and replaced by link (S,i) based on the link failure recovery rule above. This example illustrates that a link

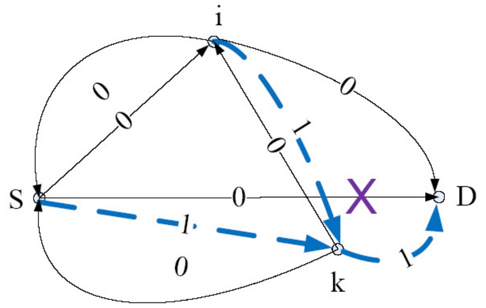


Fig. 6.7 Illustration of the secondary optimal path determination when a link (S, D) fails. The probabilities ${\pi^{(s)}}_{sj}^{D}(t)$, $s, j \in \{S, i, k\}$ are labeled on each corresponding link (s, j). Note that ${\underline{\pi}^{(s)}}_{S}^{D}(t)$ are initialized with geographical-location-aware initialization. The thick dashed arrows show the secondary optimal path from source nodes S, i, k to destination D.

failure can be recovered with the proposed link failure recovery rule once it is detected. Simulations for the link failure detection and recovery method are presented in Section 6.6.

Continuing with the example of Fig. 6.6, where the primary optimal path has recovered from the link failure on (S, D), Fig. 6.7 shows the secondary optimal path determination. As the failed link is not on the secondary optimal path, the secondary optimal path is the same as that in Fig. 6.5.

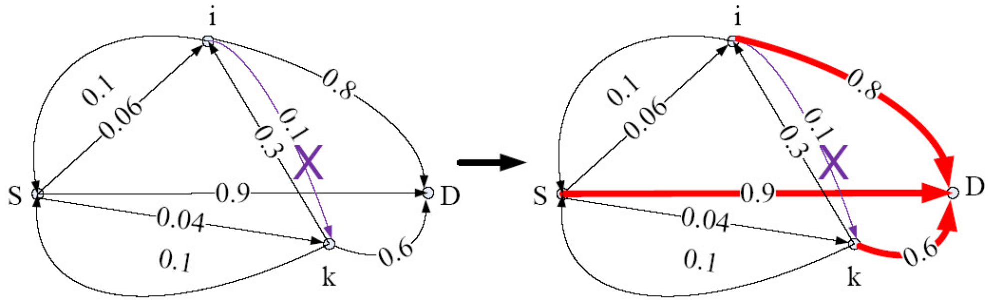


Fig. 6.8 Illustration of the primary optimal path determination when a link (i, k) fails. The left side shows the network with probabilities ${\pi^{(p)}}_{vj}^{D}(t)$, $v, j \in \{S, i, k\}$, labeled on each corresponding link (v, j). The thick arrows on the right show the new primary optimal paths from S, i, k to destination D after the link failure is detected.

In the next example, we show the link failure recovery on the primary optimal and the secondary optimal path when link (i, k) fails, as in Fig. 6.8 and Fig. 6.9. In this case, the failed link does not affect the primary optimal path, as illustrated in Fig. 6.8, but it affects the secondary optimal path selection, as shown in Fig. 6.9. When a failure is detected for link (i,k) the secondary optimal next-hop of node i has to be D, rather than k.

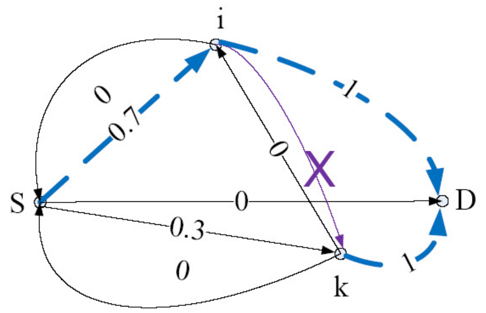


Fig. 6.9 Illustration of the secondary optimal path determination when a link (i,k) fails. The probabilities $\pi^{(s)}{}_{vj}^{D}(t)$, $v,j \in \{S,i,k\}$ are labeled on each corresponding link (v,j). The thick dashed arrows show the secondary optimal path from source S, i, k to destination D.

6.6 Performance Evaluation

This section evaluates VoIP quality on the primary optimal paths and on the diverse paths, when there is no link failure and when there are link failures.

In the following, we simulate a physical network with 50 nodes, which is a model of the AT&T backbone network, as in Fig. 3.11. Overlay nodes are selected to be geographically distributed over the network, and all the overlay nodes are interconnected. In the overlay network, we run the primary and the secondary optimal next hop learning automata detailed in Sections 5.3, 6.4 and 6.5. Based on the values of the probability distributions $\underline{\pi}^{(p)}{}_{S}^{D}(t)$ and $\underline{\pi}^{(s)}{}_{S}^{D}(t)$, $\forall S, D \in V$, the primary and the secondary optimal paths are determined and applied to route VoIP calls, as illustrated in Fig. 5.1 and Fig. 6.1.

6.6.1 Simulation Results

In this section, we compare VoIP quality, i.e. R-factor, for source routing on the learned primary optimal path (abbreviated "APL1"), diverse routing with the learned primary and secondary optimal paths (abbreviated "APL2") with three other VoIP routing methods: routing on the direct paths (i.e. paths determined by the underlying network, i.e. the Shortest Hop Paths, abbreviated "SHP"), the routing method proposed in Section 4.2 of Chapter 4 (abbreviated "CDR"), and the path switching method proposed in [46] and

Abbreviation	Routing Method	Reference	Features
APL2	Routing with the learned primary and secondary optimal paths.	Section 6.4	Decentralized, diverse, adaptive, flow-based routing
APL1	Routing with the primary optimal paths.	Algorithm 5	Decentralized, single path, adaptive, flow-based routing
APL1+LFD	The APL1 method plus the link failure detection algorithm.	Sections 6.4 and 6.5	Decentralized, single path, adaptive, flow-based routing, link failure detection
CDR	Routing with two diverse paths.	Chapter 4	Centralized, diverse, adaptive, flow-based routing
TaoPS	Path switching algorithm.	Section 2.3.2 and [46]	Decentralized, single path, adaptive, packet-based routing
SHP	Shortest hop routing.	[33]	Decentralized, single path, fixed, packet-based routing
APL1+SP	Routing with the primary optimal next-hops determined by the max-rule in (6.4).	Equation (6.4)	Decentralized, single path, adaptive, packet-based routing

Table 6.1 Abbreviations for different routing methods simulated in this section.

described in Section 2.3.2 (abbreviated “TaoPS”). The goal of this comparison is to understand if the proposed distributed diverse routing method “APL2” can provide the same competitive and stable VoIP quality as the centralized diverse routing method “CDR”, and also to show the gain of using the proposed methods over the existing methods ”SHP” and “TaoPS”. We compare our methods with “TaoPS” because “TaoPS” is also an adaptive routing method for VoIP traffic.

With background traffic generated for the 50-node model of AT&T backbone network, we evaluate VoIP quality for the primary optimal path and the diverse paths in the same 10-node overlay network as that for Fig. 5.13. In the simulation, the voice codec is G.711 [3] and the learning gains for APL1 and APL2 are set to be sufficiently small with $g = 0.0005$. The time scales for the path switching method TaoPS are set to be similar to [46] as $t_n = 5 \times 2^n$ ms for $n \in \{0, 5, 10, 12, 15\}$. As the routing method TaoPS can switch the path of an on-going call, i.e. re-routing a call when a better path is found, we also compute the R-factor for APL1 when distributed re-routing is applied (abbreviated APL1+SP). The abbreviations of all the simulated routing methods are also listed in Table. 6.6.1.

A Scenario When There is No Physical Link Failure

We first simulated the VoIP routing methods CDR, APL2, APL1, TaoPS and SHP for a scenario with no physical link failure. The average R-factors for all voice calls in the overlay network are shown in Fig. 6.10. The top three lines are the R-factors for CDR, APL2 and APL1. We can see that the three routing methods CDR, APL2 and APL1 provide competitive “High” VoIP quality, while APL2 performs slightly better than APL1. The R-factor for the path switching method TaoPS is not as high as those of CDR, APL2 and APL1, but it is still much higher than that of the shortest hop path routing SHP.

	CDR	APL2	APL1	TaoPS	SHP
R-factor	90.79±0.88	91.09±0.57	91.01±0.67	89.31±1.96	71.79±4.62

Table 6.2 Average R-factor over the whole simulation time for all the source-destination pairs when there is no link failure. Note that the number after each ± refers to standard deviation of R-factor.

Table. 6.2 shows the average R-factor of the five routing methods CDR, APL2, APL1, TaoPS and SHP when there is no link failure. It also shows that the diverse routing method APL2 performs slightly better than the single path routing APL1, TaoPS and

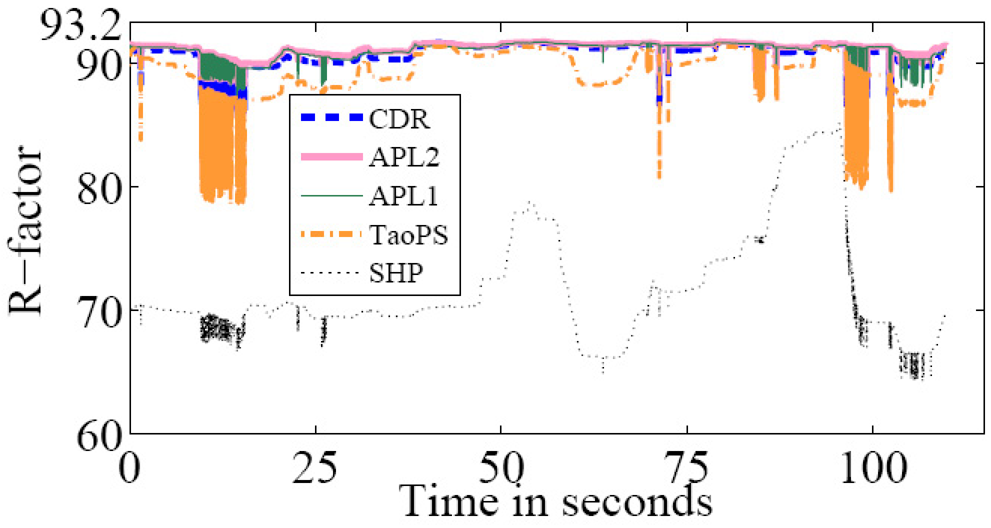


Fig. 6.10 Average R-factor for all the voice calls in the overlay network when there is no physical link failure. The lines from top to bottom represent the R-factors for APL2, APL1, CDR, TaoPS and SHP, respectively. It can be seen that the three routing methods APL2, APL1 and CDR provide competitive performance and that they are better than TaoPS and much better than SHP.

SHP on average in this case. We can see that APL1 performs slightly better than CDR, which is because CDR only considers single-hop paths. The relative increases in average R-factor for APL2, APL1 and CDR comparing to that of TaoPS are 2%, 1.91%, and 1.66%, respectively. When comparing to the R-factor on the direct paths, the relative increases for APL2, APL1, CDR and TaoPS are 26.9%, 26.78%, 26.47% and 24.41%, respectively.

A Scenario When There is a Physical Link Failure

Fig. 6.10 has shown that there is no significant gain by using diverse routing APL2 or CDR when there is no link failure. We are now interested in if there are benefits by using diverse routing when there is a physical link failure. Thus, we disable a physical link that is on the primary optimal paths of four source-destination overlay node pairs. This is a similar but worse case than that described in Fig 6.2, since four primary optimal paths fail due to this single physical link failure, which can result in Algorithm 5 to take longer to learn new optimal paths. Fig. 6.11 shows the simulation results when this link fails at 50 seconds. It can be observed that the R-factor for the diverse routing method APL2 is not affected by the link failure at 50 seconds, while that of APL1 and APL1+SP drops significantly

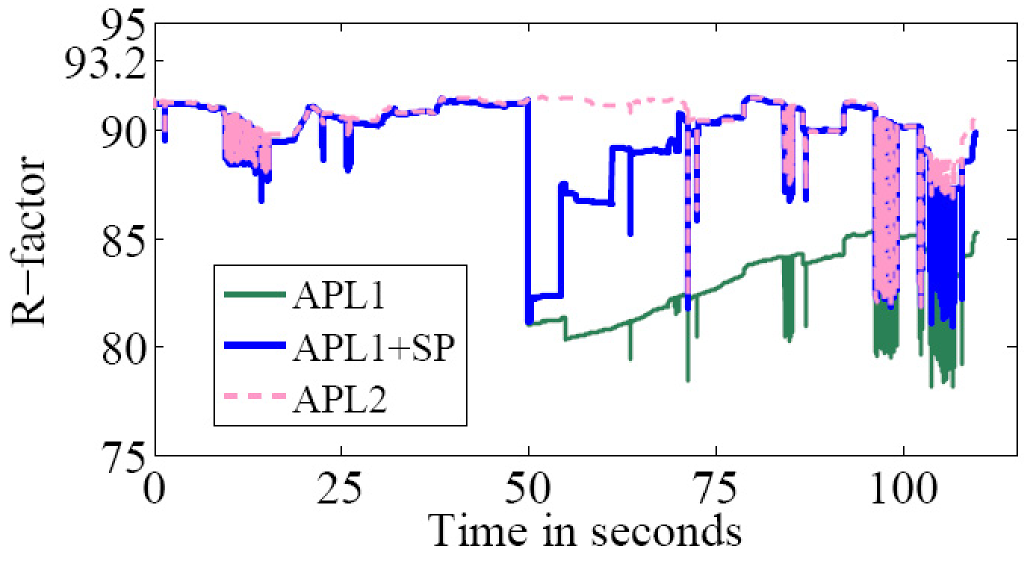


Fig. 6.11 Average R-factor for all the voice calls in the overlay network when there is a physical link failure. This link failure disables 4 primary optimal paths. From 50 to 72 seconds, we can see 4 jumps in R-factor that are due to the recovery of the 4 paths.

at 50 seconds. After that, the R-factors for APL1+SP and APL1 grow slowly[4], and that of APL1+SP gets close to the value of APL2 at around 70 seconds, i.e. the learning automata finish learning four new primary optimal paths at around 70 seconds. Note that the average R-factor for APL1+SP is higher than that for APL1 after 50 seconds. This is because APL1+SP allows packet re-routing for on-going voice calls while APL1 only allows flow-based source routing. Table 6.3 lists the average R-factor over the whole

	CDR	APL2	APL1	APL1+SP	TaoPS	SHP
R-factor	90.46±1.22	90.79±1.02	85.08±2.96	90.41±1.50	86.65±2.71	71.48±4.52

Table 6.3 Average R-factor over the whole simulation time for all source-destination pairs when there is a physical link failure. Note that the number after each ± refers to standard deviation of R-factor.

simulation time for all source-destination pairs for the six routing methods CDR, APL2, APL1, APL1+SP, TaoPS and SHP. It can be seen that the diverse routing methods CDR and APL2 performs much better than the single path routing APL1, TaoPS and SHP. We can also see that APL1+SP performs similar to the diverse routing methods. The relative

[4]This is because it takes time for Algorithm 5 to learn four new primary optimal paths.

increases in R-factor for APL2 and APL1+SP comparing to that of APL1 are 6.71% and 2.3%, respectively. In this example, we see that re-routing voice packets to newly learned primary optimal paths as in the case of APL1+SP and TaoPS provides higher VoIP quality than the case of APL1 where packets of the on-going voice calls on the link are dropped when the link failure occurs at 50 seconds. However, one should note that the packet re-routing for APL1+SP and TaoPS can result in out-of-order arrivals and may lead to route oscillations when the amount of voice traffic is non-negligible[5].

Link Failure Detection and Recovery

In this section, we evaluate the efficiency of the proposed novel link failure detection method detailed in Section 6.5. In this simulation, we assume that the same physical link failure as in Fig. 6.11 fails. The simulation result is shown in Fig. 6.12, where R-factor of an example source-destination pair that is affected by the link failure is compared for APL1 (i.e. link failure detection is not implemented), APL1+LFD (i.e. link failure detection and recovery is implemented) and APL2 (i.e. both the primary and the secondary optimal paths are used).

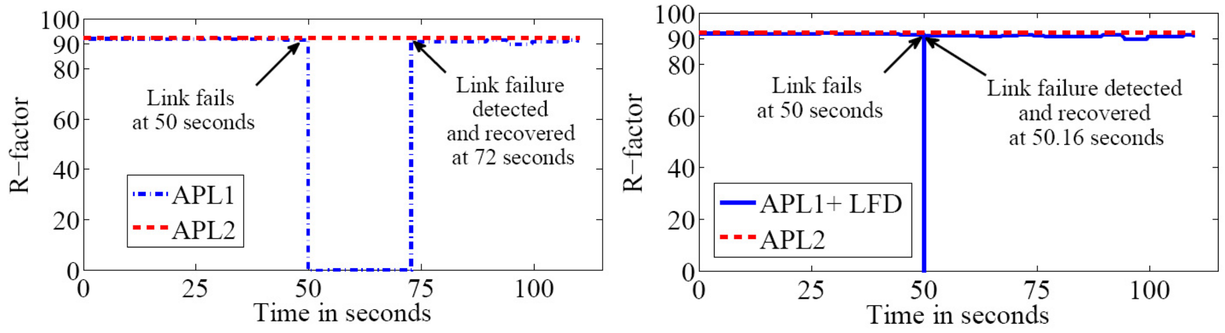


Fig. 6.12 R-factor comparison for three routing methods: APL2 (both the primary and the secondary optimal paths are used), APL1 (i.e. link failure detection is not implemented) and APL1+LFD (i.e. link failure detection is implemented with $N = 1$, $\epsilon = 10^{-6}$ in Algorithm 7) for an example source-destination pair. The figure on the left shows that VoIP quality is not affected by the link failure when diverse routing is adopted, while the APL1 method takes 22 seconds to detect and recover from the link failure. The figure on the right shows that APL1+LFD method takes only 0.16 second to detect and recover from the same link failure.

[5]In this example, we assume the amount of voice traffic is small compared to other background traffic on a link, as it is currently and likely to be in the future.

As shown in Fig. 6.12, when no link failure detection is applied, Algorithm 5 discovers a new primary optimal path at $t = 72$ seconds. Consequently, all voice calls/packets on this optimal path will be dropped due to the 22 second gap[6]. When the proposed link failure detection algorithm is implemented, with the same link failure setting, the link failure is detected and recovered at $t = 50.16$ seconds, just a few tenths of a second after the failure occurs. This indicates that the link failure detection method can significantly reduce the gap from 22 seconds to 0.16 seconds. It should also be noted that when both the primary and the secondary optimal paths are used, no gap is observed in R-factor for APL2.

Fig. 6.12 has shown the link failure detection time for an example source-destination pair. The link failure detection time is in fact related, the probing rate and the parameters (ϵ>0 and N>0) in Algorithm 7. The higher the probing rate is, the faster the link failures are detected. The lower the ϵ is and the higher the N is, the slower the link failures are detected. One should also note that the link failure detection time is not affected by the learning gain, as defined in Algorithm 7. According to (5.13), we can compute the probing packet arrival rate γ_i^v at a node i for a destination v, the link failure detection time for a failed overlay link (i, v) can then be estimated as $\frac{n(\epsilon)+(N-1)}{\gamma_i^v}$.

The Average Result with Random Physical Link failures

We also simulated the six different routing methods CDR, APL2, APL1, APL1+SP, TaoPS and SHP for 20 different scenarios with a random physical link failure [17]. The random physical link failures are generated from a link failure model similar to [17], where each link fails with a probability uniformly distributed between $[10^{-4}, 10^{-3}]$. Fig. 6.13 shows the average R-factor for all calls over all scenarios for the five routing methods CDR, APL2, APL1, TaoPS and SHP from top to bottom[7]. R-factors for the centralized and the distributed diverse routing methods CDR and APL2 are shown by the top two lines. It is evident that R-factor for these two routing methods are very close to 93.2, i.e. the maximum R-factor for the G.711 codec, which means they can provide "best"[8] quality VoIP and thus both are good heuristics for the optimal diverse routing problem in (6.1). It is also observable that CDR performs slightly better than APL2 because the diverse paths chosen

[6]This gap can be reduced by increasing probing rate or learning gain.

[7]The R-factors for APL1+SP and APL1+LFD are between that of APL2 and APL1. They are not shown in Fig. 6.13 because they are very close to that of APL2 and APL1.

[8]The VoIP quality level has been illustrated in Fig. 2.3.

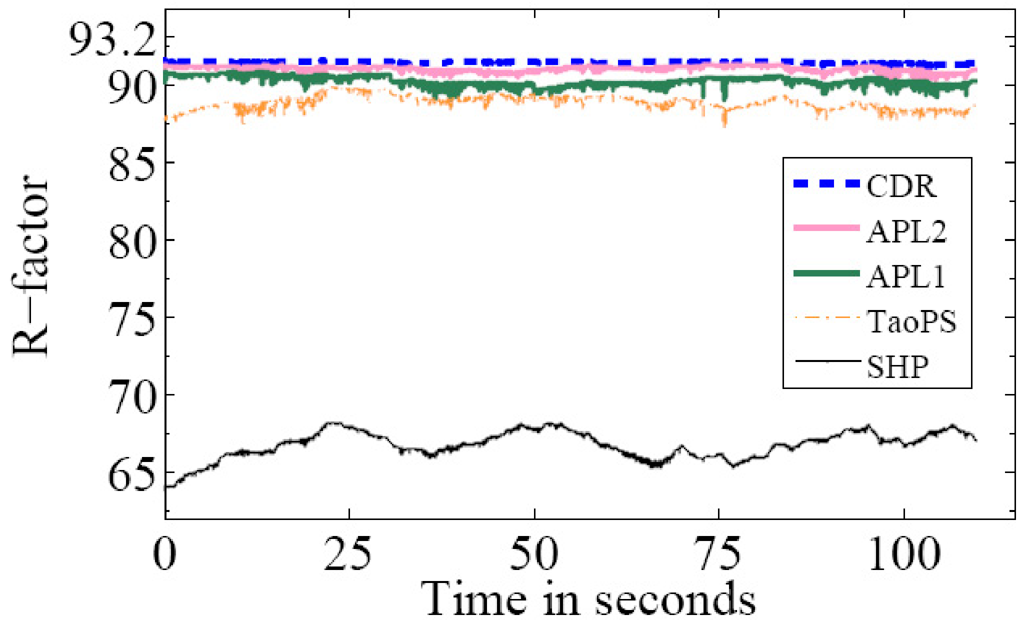


Fig. 6.13 Average R-factor of all the voice calls in the overlay network for 20 different scenarios (with random link failures). The lines from top to bottom represent the R-factor for CDR, APL2, APL1, TaoPS and SHP, respectively.

by CDR are selected based on mouth-to-ear latency and loss, while those of APL2 are based on network latency only. R-factor for the single path routing method APL1 is lower than those of the diverse routing methods CDR and APL2, which shows the advantage of diverse routing over single path routing. It is observable that APL1 performs better than TaoPS because the path switching method is sub-optimal as discussed in Section 2.3.2.

	CDR	APL2	APL1	APL1+SP	TaoPS	SHP
R-factor	91.54±0.09	91.14±0.18	90.35±0.31	90.81±0.27	88.98±0.44	66.80±0.89

Table 6.4 Average R-factor for six routing methods: CDR, APL2, APL1, APL1+SP, TaoPS, SHP, when there is a random link failure. Note that the number after each ± refers to standard deviation of R-factor.

The average R-factors over time for the six routing methods are listed in Table. 6.4. It can be seen that the four routing methods CDR, APL2, APL1+SP and APL1 provide competitive VoIP quality, with the diverse routing CDR and APL2 performing slightly better than the single path routing APL1+SP and APL1. APL1+SP performs slightly better than APL1 because the APL1+SP method adaptively learns a better path when a link failure happens on an overlay link and it re-routes voice calls to the better path. The difference between APL1 and APL1+SP is small because the probability a link fails in 100 seconds is still small although we have set the link failure probability to be relatively large (within $[10^{-4}, 10^{-3}]$ comparable to that in [17]). The relative increases in R-factor for CDR, APL2, APL1+SP and APL1 comparing to that of TaoPS are 2.88%, 2.43%, 2.06% and 1.54%, respectively[9]. When comparing to the R-factor of the voice calls on the direct paths, we see a big gap between those of the adaptive routing methods CDR, APL2, APL1+SP, APL1 and TaoPS and that of SHP, where the relative increases in R-factor are 24.74%, 24.34%, 24.01%, 23.55% and 22.19%, respectively, which shows significant advantage of adaptive routing for improving VoIP quality in overlay networks.

Fig. 6.12 has shown the link failure detection time for an example source-destination pair. For the example with 20 scenarios, we also compute the average link failure detection time for the two methods APL1 and APL1+LFD[10], as given in Table. 6.5. We see that the link failure detection time when the link failure detection algorithm is adopted is 8 times faster than that when it is not used.

	APL1	APL1+LFD
Link failure detection time (second)	$3.67\pm$ 2.91	0.46 ± 0.31

Table 6.5 Average Link Failure Detection for APL1 and APL1+LFD. Note that the number after each ± refers to standard deviation of R-factor.

Remarks

From the simulations in Fig. 6.10, Fig. 6.11, Fig. 6.12 and Fig. 6.13 we can reach the following conclusions. First, when there is no link failure in the network, the single path routing method APL1 provides competitive performance to the diverse routing methods CDR and APL2 in terms of R-factor. Second, when there are link failures, the diverse

[9]The difference is not as large as that in Table. 6.3 because 13 out of 20 scenarios are without overlay link failure during the simulation time.

[10]Link failures that are not detected before the simulation time ends for APL1 are not counted in the average link failure detection time computation. That is why we do not show the link failure detection time for TaoPS, since it only detects one link failure for all the 7 scenarios that have overlay link failures and the detection time was 4.27 seconds for the one case). However, we can still estimate the link failure detection time for TaoPS as longer than 36.8 seconds since we know each link failure time and the simulation ending time.

routing methods CDR and APL2 provide higher R-factor than the single path routing method APL1. When path switching is allowed for APL1, the single path routing method APL1+SP can provide competitive R-factor to the diverse routing methods. Third, in the simulations, we also observe that the proposed three methods CDR, APL1 and APL2 provide higher R-factor than the path switching method TaoPS on average. Fourth, in terms of scalability, the distributed routing methods APL1 and APL2 are better than the centralized routing method CDR and the brute-force search method TaoPS.

6.7 Stability Analysis

Simulation results in Section 6.6 have shown that the learned primary optimal path and diverse paths are able to improve VoIP quality. The other issue of interest is the stability of the routing approach given in Algorithm 5 and Algorithm 6. As both the primary and secondary optimal path determinations are based on the max-rule in (6.4) or (6.5), we only analyze the stability of the primary optimal path determination in this section.

For an overlay network with m overlay nodes and $m(m-1)$ overlay links, we have the matrix $\Pi^{(v)} = \left[\pi_{ij}^{(v)}\right], v = 1, ..., m$, with $0 \leq \pi_{ij}^{(v)} \leq 1$ denoting the probability that voice arrivals with source i and destination v should be sent to node j. Let $\Pi = [\Pi^{(v)}]$. Then the user optimization problem for each source-destination pair (i, j), $\forall i, j \in \{1, ..., m\}, i \neq j$, is as follows.

$$\Pi^* = \arg\min_{\Pi} D(i, j) \tag{6.6}$$

where $D(i, j)$ is the end-to-end delay from node i to j[11]. If there exists a feasible solution to the above delay minimization problem in (6.6), one can claim that there is an optimal load sharing solution for the problem, and it should be stable. However, it is hard to find the optimal solution for the minimization problem analytically. Paper [63] suggests a learning algorithm to find the user equilibrium solutions. Thus, we can check if the average delay for the paths determined by the max-rule in (6.4) converges to that given by the optimal solution found by the learning algorithm in [63].

This framework is simulated in the following setting. For the same 50 node model of the AT&T backbone network and the same overlay network, we simulated the max-rule based primary optimal path determination algorithm, as in Algorithm 5, and the user equilibrium

[11]The computation of D is given in Appendix D.

learning algorithm in [63].The average delay over all source-destination pairs as a function of time is shown in Fig. 6.14. Note that the delay function $\underline{d}(\underline{f}, \underline{c})$ is determined according to a piecewise linear fit to link delay function as in paper [116]. The absolute value for the delays can be tuned. However, as we are only concerned with the stability of the adaptive routing approach, we did not optimize the parameter setting for the delay function in this simulation.

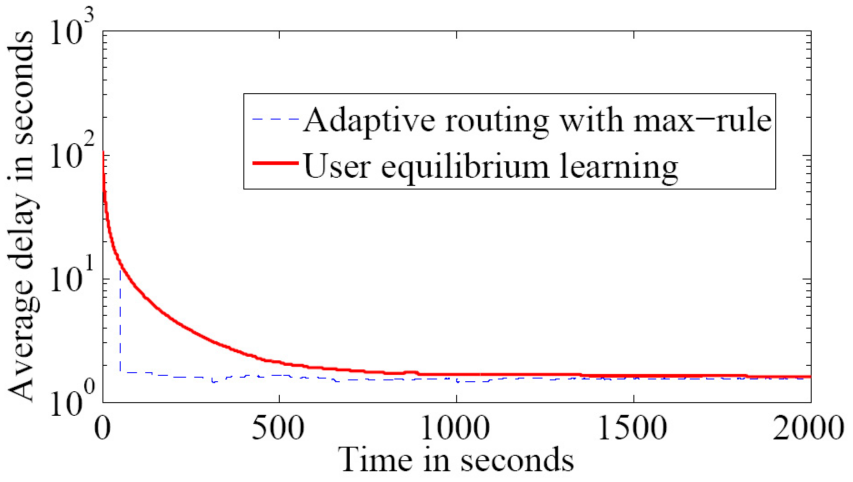
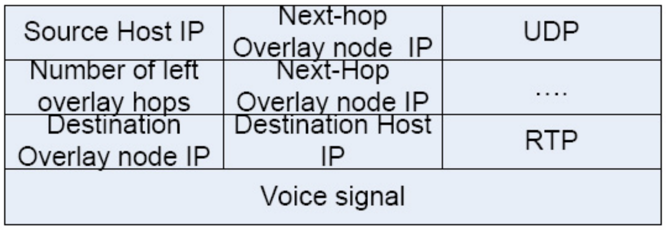


Fig. 6.14 The average delay over all source-destination pairs as a function of time.

Fig. 6.14 shows the simulation results for the max-rule based adaptive VoIP routing method and the user equilibrium learning method as in paper [63]. The thick line shows the average delay for the user equilibrium learning method. At the beginning, the average delay is very large because uniform random routing is assumed for the initial stage of the method. The thick line suggests the downward trend for average delays with marginal fluctuation due to call departure. It can also be seen that the slope of the line decreases with time and the line becomes almost flat at the end of the simulation. The thin dash-line shows the average delay for the max-rule based adaptive routing method; before time 50 seconds, the thin dash line and the thick line overlaps because the user equilibrium learning method is also used for the warm-up period of the max-rule based method; the thin dash line shows lower average delay from the time the max-rule base method is applied, i.e., the time 50 seconds; the average delay after time 500 seconds remains steady, with small magnitude of fluctuation due to new call arrivals and departures; the average delay value is lower than that of the method in [63] because the max-rule based method predicts the
]

best route even before the user equilibrium learning method converges. When the user equilibrium is reached for the method in [63], the average delays for both methods are very close. As suggested by the figure, the effect of routing new calls adaptively with the max-rule cannot significantly affect system performance; it is because the number of new call arrivals at each time instant is much less than that of the on-going calls. Hence, the system performance is more determined by the ongoing calls than by the adaptive routing of new calls.

6.8 Implementation Considerations



Source Host IP	Next-hop Overlay node IP	UDP
Number of left overlay hops	Next-Hop Overlay node IP	....
Destination Overlay node IP	Destination Host IP	RTP
Voice signal		

Fig. 6.15 Voice packet format

For implementation, each voice packet is encapsulated as in Fig. 6.15. In each packet, the IP addresses of all the intermediate overlay nodes are included for source routing. The field "Number of left overlay node" represents the number of remaining overlay hops before the voice packet reaches its destination overlay node. It starts with the length of the source-destination routes and decreases 1 each time it arrives at an intermediate overlay node. When the packet reaches its destination, this field equals zero. The "Destination Host IP" is the callee's IP address, which is the final destination of the packet. Voice signals are encapsulated using the RTP protocol [117].

6.9 Overlay performance comparison with BGP and RON for VoIP routing

In this section, we consider the performance of the proposed overlay routing framework with that of BGP [34] and RON [48, 49], as shown in Table 6.6. BGP is a completely distributed routing protocol and does not require any network monitoring mechanism;

	BGP	RON	our work
Scalability	++	−	+
Path optimality	−	+	++
Failure detection speed	−	+	++

Table 6.6 Overlay Performance Comparison. "+" means "good", "−" means "bad".

thus, it has excellent scalability. However, it has been performing poorly with its policy based path selection schemes, e.g. AS-hop based routing and hot-potato routing, which can not choose the paths with optimal performance for applications and are usually slow to detect path failures [35–39,48,49]. On the other hand, RON requires aggressive monitoring of overlay link performance, and a central performance database is required for storing and distributing link performance measurements to all the overlay nodes, which limits its scalability. Optimal path selection in RON is based on inaccurate overlay link performance estimation, and only one-hop overlay paths are considered. RON is able to detect path failures in a few seconds, which is faster than that with BGP [48,49]. In our work, minimum delay paths and diverse paths can be learned with a reasonable amount of probing overhead; the probing and minimum delay path learning scheme is completely distributed, yielding good scalability as shown in Fig. 5.13 of Chapter 5. The optimality of the minimum delay paths has also been shown. The proposed link failure detection scheme is able to detect and recover from a failure in less than 2 seconds on a typical backbone network, which can prevent a voice connection from being dropped. Thus, our work provides better path quality and failure detection speed, as indicated in Table 6.6.

6.10 Summary

This chapter presented a distributed method to identify and track the primary and the secondary optimal paths for VoIP in service overlay networks. Our approach is to deploy learning automata at each overlay node. The learning automata are responsible for controlling the probing process, with the idea being to quickly rule out routes that will not lead to optimal performance so that probing resources can be focused on promising paths. Ideally, the primary and secondary optimal paths we learn should be statistically independent, however this is a challenging task to accomplish in practice. Instead, we learn the secondary optimal path as a minimum delay path that are maximally disjoint with the primary optimal path in the overlay network. Simulations on a model of the AT&T backbone network indicate that our algorithm can considerably improve VoIP quality from medium to high level, and by using diverse paths, in combination with a novel algorithm for detecting link failures, we can make VoIP overlay networks more resilient to link failures.

Chapter 7

Conclusion

This chapter summarizes the original contributions of this study and the advancements it makes over previous research. First, I developed a QoS routing method to optimize R-factor by jointly considering routing and the play-out scheme of the receiver. In addition, I developed a distributive minimum delay path learning method that is scalable with network sizes, and a distributive diverse path learning method that improves VoIP quality. Finally, I developed a link failure detection and recovery method that improves routing robustness against link failures.

7.1 End-to-end delay analysis, simulation and sampling

Understanding Internet end-to-end delay characteristics is essential for VoIP QoS routing. In this context, we studied end-to-end delay characteristics by modeling and simulating end-to-end delay traces and found the potential advantage of VoIP QoS routing. Using the real end-to-end delay traces measured by an Internet Service Provider, we analyzed the marginal probability distribution for delay samples, the autocorrelation of delay traces, the relation between the sample mean and the sample standard deviation of delay traces, and the relation between the propagation delay and geographical distance. We found that there is no strong correlation between the propagation delay of a path and the geographical distance between two ends of the path. We also observed that for most of the links analyzed, the sample mean and the sample standard deviation are more correlated when the autocorrelation of a delay trace is weak than when it is strong. Based on these analyses, we suggested that a delay trace with weak or no autocorrelation can be synthesized using its sample mean, sample standard deviation and minimum delay. We also simulated delay traces based on a fluid model of the 50-node model of the AT&T backbone network, in which we incorporated a fractional Brownian motion traffic model, a gravity traffic demand model and a maximum link-utilization-rate based link capacity assignment technique. Similar delay spikes, autocorrelation and cross-covariance were observed in the simulated delay traces as those in the real end-to-end delay traces we obtained from an ISP.

We also investigated the bi-objective optimization problem arising from end-to-end delay sampling, i.e. finding a fair sampling frequency to both maximize the sampling accuracy and minimize the sampling cost. A proportional fair solution, which equalizes the rate of change for the two competing objectives, was proposed to solve this problem. Any deviation from this solution would result in one side losing proportionally more than the other side gained.

7.2 R-factor based diverse routing: a centralized approach

Based on the analysis of end-to-end delay characteristics, we suggested a routing method that maximizes R-factor for VoIP by jointly considering routing and the adaptive play-out scheduling at the receiver. The advantage of this method over previous studies is that we adapted the routing control to provide paths with low mouth-to-ear delay and loss instead of just low network delay and loss, as in other research. Using this method, improved VoIP quality in terms of R-factor was observed in simulations.

This method requires transmitting delay distributions of all the overlay links to a central node for estimating the delay distributions of diverse paths. We proposed to transmit only the model parameters of the link delay distributions to reduce the communication overhead, and demonstrated that the selected diverse paths based on these model parameters provide comparable VoIP quality as that of the real best diverse paths.

7.3 Learning minimum delay paths and diverse paths: a distributed approach

The centralized method is useful primarily for small scale overlay networks. For large-scale overlay networks, we developed a scalable distributed approach to adaptively learn the minimum delay paths and the best pair of diverse paths. The minimum delay paths are learned with active probing and learning automata. We presented four variations of active probing and learning methods, i.e. hop-by-hop learning with uniform or geographical-location-aware initialization and end-to-end learning with uniform or geographical-location-aware initialization. Simulations on these schemes showed that hop-by-hop learning converges faster to the minimum delay paths than end-to-end learning. For uniform initialization, when the learning gain of the learning algorithm is sufficiently small, the minimum delay paths are learned; while with geographical-location-aware initialization, there is no random loops on the probing paths from the beginning of learning. We have also demonstrated, by simulations, the scalability and the optimality of the approach. Convergence to the minimum delay paths was proved using Kushner's weak convergence method.

We also applied the active probing and learning algorithm to learn a pair of diverse paths for VoIP routing. This is a distributed approach since VoIP routing decisions were made locally at each overlay node based on the state of the learning automata. VoIP quality was shown to improve from unsatisfactory levels to satisfactory levels. A much more stable performance with diverse routing was also observed compared to that seen on a single path.

Finally, we proposed a novel link failure detection and recovery scheme based on the changes of the parameters of learning automata. In this respect, simulation results have shown considerable improvement in link failure recovery time.

7.4 Remarks and future work

Quality of Service routing for VoIP investigated in this work allows VoIP applications to take advantage of network measurements to obtain better quality. To achieve "high" or "best" VoIP quality, we recommend the centralized routing method proposed in Chapter 4 for small-scale service overlay networks. When the network size is large, we recommend the distributed diverse routing method investigated in Chapter 6 if bandwidth is not a concern; otherwise, the primary path routing with the link failure detection method can perform similarly well when bandwidth is a concern.

In fact, the proposed methods in this thesis are not limited to the service overlay networks although the thesis work is conducted in the context of service overlay networks. The insights gained here regarding end-to-end delay characteristics, the solution to the bi-objective optimization problems in end-to-end delay sampling, the consideration of adaptive play-out scheduling in the centralized R-factor-based diverse routing problem, active probing and learning, online distributed diverse routing and link failure detection and recovery approaches can all be applied to solve problems in other types of networks. In future work, we would like to explore these potential applications. For example, we can investigate how the active probing and learning based routing method can be applied to find bandwidth-optimal paths for video-streaming services. In that case, the feedback to the learning automata would be bandwidth estimation of different paths. Another interesting problem to study is how a strategic overlay network design, e.g. overlay node selection, can affect the performance of the diverse routing. This is because the bandwidth, delay, loss and failure probability of the overlay links are determined by the choice of overlay nodes for a full-mesh overlay network. We would expect that an optimal overlay network design changes with time since network traffic varies with time in a low frequency. The method for finding a fair operation point for the bi-objective optimization problem can also be extended to solve other multi-objective optimization problems. The diverse routing methods and link failure detection method have been shown to provide better resiliency than single path routing methods for failure-prone networks. It is thus also interesting to investigate the performance of these methods in wireless networks.

Appendix A

A.1 ODE approximation of the cross-correlation learning algorithm

In this section, we prove the cross-correlation learning algorithm weakly converges to an Ordinary Differential Equation (ODE) with Kushner's method [30] following the proof in Vázquez-Abad and Mason's work [114, 115].

For each automaton with source-destination pair (S,D), $\tau_k^{(S,D)}$ is the epoch of the k_{th} local update. $\underline{\pi}_S^D$ is the state probability vector of the automaton. Then the cross-correlation learning automata can be expressed as $\underline{\pi}_S^D(k+1,g) = \underline{\pi}_S^D(k,g) + gX_k^g$, where g is the learning gain, $X_k^g = (\delta_{iu} - \underline{\pi}_i^k(k,g))s(u,k)$, action u is a random action drawn from $\underline{\pi}_S^D(k,g)$ and $s(u,k)$ is the network reward for action u.

We sort the epoch of the updates at all learning automata in the network, and let τ_k be the epoch of the k_{th} global update. Let k be the k_{th} global update epoch. We define θ as the vector of all $\underline{\pi}_S^D, \forall S, D$, then $\theta_{k+1}^g = \theta_k^g + gY_k^g$. Y_k^g depends on the feedback function. This process can be embedded in a Markov decision process (MDP) (θ_k^g, ξ_k^g). The state ξ_k^g is the vector of queue lengths and local information at time τ_k.

Let $\mathrm{G}(\theta,\xi) = \mathrm{E}(Y_k^g|\theta_k^g = \theta, \xi_k^g = \xi)$. The σ-algebra related to the MDP up to the k_{th} update will be denoted by $\mathcal{F}_k^g$. By the Markovian property, $\mathrm{G}(\theta,\xi) = \mathrm{E}(Y_k^g|\mathcal{F}_k^g)$, which is a random variable depending on the distribution of (θ_k^g, ξ_k^g).

In our problem, the fixed control process $\xi(\theta)$ is Markovian with transition probability $\mathrm{P}(\mathrm{d}x, x) = \mathrm{P}(\xi_{k+1}(\theta) \in x + \mathrm{d}x|\xi_k(\theta) = x)$, which is weakly continuous in θ. The network is stable for every possible θ since the probing traffic is negligible, and even if it is not negligible, the finite buffer network guarantees the network is stable for every possible value of θ.

Let $\mu_\theta(\mathrm{d}x)$ be the invariant measure of the fixed control process and

$$\begin{aligned}\mathrm{g}(\theta) &= \int \mu_\theta(\mathrm{d}\xi)\mathrm{G}(\theta,\xi) \\ &= \lim_{m\to\infty}\frac{1}{m}\sum_{k=0}^{m-1}\mathrm{E}(\mathrm{G}(\theta,\xi_k(\theta))) \\ &= \lim_{m\to\infty}\frac{1}{m}\sum_{k=0}^{m-1}\mathrm{E}_\xi(\mathrm{E}_{Y_k^g}(Y_k^g|\theta_k^g=\theta,\xi_k^g(\theta)=\xi))\end{aligned}$$

The random variables Y_k^g are uniformly integrable, since they are uniformly bounded by construction of the feedback function. Furthermore, the sequence $\{(\theta_k^g,\xi_k^g),k\geq 0,g>0\}$ is tight. That is, for every $\alpha>0$, there exists a compact set K_α such that $\sup_{g,k}\mathrm{P}((\theta_k^g,\xi_k^g)\notin K_\alpha)<\alpha$. This follows because θ_k^g are probability vectors and ξ_k^g are uniformly tight: queue sizes are all bounded by the buffer size. Tightness is the stochastic analog of compactness. Define the control interpolation process: $\theta^g(t)=\theta_k^g$ for $t\in[kg,(k+1)g]$. From this definition and the stochastic approximation form, for $t=gk$, we have: $\frac{\theta^g(t+g)-\theta^g(t)}{g}=Y_k^g$, and the conditional expected behavior of Y_k^g is related to $\mathrm{E}\{\frac{\theta^g(t+g)-\theta^g(t)}{g}|\mathcal{F}_k^g\}=\mathrm{G}(\theta^g(t),\xi_k^g)$. From Vázquez-Abad et.al [114] proposition 1, it follows that every subsequence of $\theta^g(\cdot)$ as $g\to 0$ has a further weakly convergent subsequence and all weak limits are Lipshitz continuous a.s. All the assumptions of Kushner and Vázquez-Abad et.al [30] are satisfied, therefore, any such limit satisfies the ODE $\frac{\mathrm{d}\theta(t)}{\mathrm{d}t}=g(\theta(t))$. If the ODE has a unique solution for each initial condition, the limit does not depend on the subsequence and therefore $\theta^g(\cdot)$ converges to $\theta(\cdot)$. As θ is a vector of all the control parameters $\pi_{Sj}^D,\forall S,j,D$,

$$g(\theta)=\lim_{m\to\infty}\frac{1}{m}\sum_{k=0}^{m-1}\mathrm{E}_\xi\mathrm{E}_{Y_k^g}(Y_k^g|\theta_k^g=\theta,\xi_k^g(\theta)=\xi) \tag{A.1}$$

is also a vector. Let the control parameter corresponding to $\pi_{Sj}^D(k,g)$ and $\underline{\pi}_S^D(k,g)$ be θ_{Sj}^D and θ_S^D, respectively. Denote the expected reward strength and the expected delay of sending a probe from S to D through j as s_{Sj}^D and Δ_{Sj}^D, respectively. Supposing $\theta^d(t)$ represent the routing pattern for destination d, we have $\theta(t)=\{\theta^1(t),...,\theta^N(t)\}$. For an element of $g(\theta)$, it can be shown that for the cross-correlation learning algorithm,

$$\begin{aligned}
\mathrm{g}(\theta_{Sj}^D)(t) &= \lim_{m\to\infty}\frac{1}{m}\sum_{k=0}^{m-1}\mathrm{E}_\xi(\mathrm{E}_{Y_k^g}(Y_k^g|\theta_{Sj}^D(k,g)=\theta_{Sj}^D(t),\xi_k^g(\theta)=\xi)) \\
&= Q_j(\theta(t))\Big(\lambda_S^D\theta_{Sj}^D(t)s_{Sj}^D-\theta_{Sj}^D(t)\sum_u s_{Su}^D\lambda_u^D\theta_{Su}^D(t))\Big) \\
&= Q_j(\theta(t))\lambda_S^D\Big(\theta_{Sj}^D(t)(1-\frac{\Delta_{Sj}^D(t)}{\mathrm{d}_{max}})-\theta_{Sj}^D(t)\sum_u(1-\frac{\Delta_{Su}^D(t)}{\mathrm{d}_{max}})\theta_{Su}^D(t)\Big) \\
&= -\frac{Q_j(\theta(t))\lambda_S^D\theta_{Sj}^D(t)}{d_{max}}\Big(\Delta_{Sj}^D(t)-\sum_u\theta_{Su}^D(t)\Delta_{Su}^D(t)\Big) \qquad \text{(A.2)}
\end{aligned}$$

where λ_S^D refers to the probing rate from source S to destination D. $Q_j(\theta(t))\lambda_S^D$ is the parameter update rate for $\theta_{Sj}^D(t)$, which can be computed as the probing packet arrival rate γ_S^D according to (5.13).

A.2 Learning gain

Paper [110] points out that the learning gain of the cross-correlation learning algorithm must be sufficiently small for the learning automata to converge. However, it is not clear how small the learning gain should be used in practice. For the cross-correlation learning algorithm, we can derive an ordinary differential equation (ODE) that approximates the behavior of the algorithm. Then the solution to the ODE implies the performance of the algorithm when it converges.

The general framework for such an ODE approximation is [109]: given a learning algorithm of the form $\theta_{k+1}^g=\theta_k^g+g\mathrm{G}(\theta_k^g,\xi_k^g)$, $\theta_0^g=\theta_0$, where $\theta_{k+1}^g\in\Re^N$ is the state vector and $\xi_k^g\in\Re^{N'}$ is the noise vector, $\mathrm{G}(\cdot,\cdot)$ is a function that maps $\Re^N\times\Re^{N'}$ to $\Re^N$. The approximating ODE for this algorithm is $\frac{dz}{dt}=\mathrm{g}(z)$, with $z(0)=\theta_0$. It is proven in [109] that for any initial condition x_0 and any given finite T, ϵ, $\delta>0$, there exists a $g^*>0$ such that for all $0<g\le g^*$, the probability that the maximum difference between the value of the learning algorithm θ_k^g and that of the ODE $z(kg)$ is greater than ϵ is less than δ, i.e.,

$$Prob\{sup_{0\le k\le\frac{T}{g}}\|\pi_{Sj}^D(k)-z(kg)\|\ge\epsilon\}\le\delta, \qquad \text{(A.3)}$$

In page 234 of [109], g^* can be set as $\frac{\delta\epsilon^2}{K_3T}$, where K_3 satisfies $kK_3\ge(k+1)\eta$ and η is the maximum of $\mathrm{E}\{\|\mathrm{G}(\theta_k^g, \xi_k^g) - \mathrm{g}(\theta_k^g)\|^2\}$.

Hence, if we can derive η for the cross-correlation learning algorithm, we can find the appropriate learning gain $g \leq \frac{\delta\epsilon^2}{K_3 T}$ to guarantee ϵ-optimality. Derivation of η, which requires analyzing the dynamics of a Markov decision process, is left for future work. One intuition is that the more variant the network performance is, the higher η will be, and the smaller $g^* = \frac{k\delta\epsilon^2}{(k+1)T\eta}$ should be.

A.3 Some properties of the random paths determined by learning automata

Lemma A.3.1. *For a full mesh service overlay network with m overlay nodes, at the beginning of active probing with the uniform initialization, the probability of there being at least a random loop is* $1 - \sum_{k=1}^{m-1} \frac{\mathrm{P}_{m-2}^{k-1}}{(m-1)^k}$*, where* $\mathrm{P}_{m-2}^{k-1} = \frac{(m-2)!}{(k-1)!}$.

Proof. Let event $A_k, k = 1, ..., m-1$, be there is no loop in k hops. Let event $B_k, k = 1, ..., m-1$, be the total number of hops from source to destination is k. Then we have:

$$\Pr\{A_k\} = \frac{\mathrm{P}_{m-2}^{k-1}}{(m-2)^{k-1}} \tag{A.4}$$

$$\Pr\{B_k\} = \frac{(m-2)^{k-1}}{(m-1)^k}, k = 1, ..., m-1 \tag{A.5}$$

$$\Pr\{\textit{there is no loop}\} = \sum_{k=1}^{m-1} \Pr\{A_k\}\Pr\{B_k\} = \sum_{k=1}^{m-1} \frac{\mathrm{P}_{m-2}^{k-1}}{(m-1)^k} \tag{A.6}$$

$$\Pr\{\textit{there is at least a loop}\} = 1 - \Pr\{\textit{there is no loop}\} = 1 - \sum_{k=1}^{m-1} \frac{\mathrm{P}_{m-2}^{k-1}}{(m-1)^k} \tag{A.7}$$

□

For example when m=3, the probability of at least a loop in a random path is 0.25; when m=5, the probability of at least a loop is 0.5117; when m=10, the probability is 0.7419. It shows that the more nodes in the network, the higher probability there is a loop in the random path, which is also true by intuition.

Lemma A.3.2. *For a mesh network with m nodes,* $m \geq 3$*, let the number of hops from a source node i to a destination node* $d, d \neq i$*, be N. At the beginning of the probing* *with the uniform initialization, the expectation of* N *is* $m-1$*, and the variance of* N *is* $(m-1)^2-(m-1)$*. The probability of* N *being over* $(k+1)(m-1)$ *is less than* $\frac{1}{k^2}$*.*

Proof.

$$\begin{aligned}
\Pr\{N=n\} &= \frac{(m-2)^{n-1}}{(m-1)^n}, n=1,...,\infty \\
\mathrm{E}\{N\} &= \sum_{n=1}^{\infty}\Pr\{N=n\}\cdot n = m-1 \\
\mathrm{E}\{N^2\} &= \sum_{n=1}^{\infty}\Pr\{N=n\}\cdot n^2 \\
&= 2(m-1)^2-(m-1) \\
\mathrm{Var}\{N\} &= \mathrm{E}\{N^2\}-\mathrm{E}\{N\}^2 \qquad (A.8) \\
&= (m-1)^2-(m-1)
\end{aligned}$$

By Chebyshev inequality, we know $\Pr\{|N-\mathrm{E}\{N\}|>k\sqrt{Var\{N\}}\}\leq\frac{1}{k^2}$,
i.e. $\Pr\{N>(m-1)+k\sqrt{(m-1)(m-2)}\}\leq\frac{1}{k^2}$.
Therefore, $\Pr\{N>(k+1)(m-1)\}\leq\frac{1}{k^2}$. For example, if $k=10$, $\Pr\{N>11(m-1)\}\leq$ 0.01. □

Further, given the queuing delay probability on one link
$\Pr\{Any\ one\ hop\ delay>x\}<h(x)$, we can find the end-to-end delay probability
$\Pr\{End-to-end\ delay>(k+1)(m-1)x\}\leq\frac{1}{k^2}h(x)$. This result is useful since we need to set an appropriate maximum Round Trip Time (RTT) in computing the reward strength in Eq. (5.3). For example, if the tolerable error for the maximum RTT estimation is ε and $h(x)=0.01$, i.e. $\Pr\{End-to-end\ delay>(k+1)(m-1)x\}\leq\frac{1}{k^2}h(x)=\varepsilon$, then $k=\sqrt{\frac{h(x)}{\varepsilon}}$. Let $h^{-1}(\cdot)$ be the inverse of $h(\cdot)$. The maximum RTT can be set as $2(\sqrt{\frac{\delta(x)}{\varepsilon}}+1)(m-1)x$.

Appendix B

Table B.1 Nodes in the 50-node AT&T backbone network in Fig. 3.11

City	latitude (n)	longitude (w)	Population
Seattle, WA	47.616665	122.35	582454
Portland, OR	45.533333	122.65	537081
San Francisco, CA	38	122.55	744041
Oakland, CA	37.8044	122.2697	397067
Santa Clara, CA	37.35	121.96667	1731281
Sacramento, CA	38.5817	121.4933	1374724
los angeles,CA	34.1	118.4	9948081
Anaheim,CA	33.8353	117.9136	334425
San Diego, CA	32.8	117.13333	2941454
Spokane, WA	47.666668	117.4	446706
Las Vegas, NV	36.2	115.21667	552539
Salt Lake City, UT	40.766666	111.916664	178858
Phoenix, AZ	33.533333	112.066666	1512986
Tucson, AZ	32.2217	110.9258	518956
Denver, CO	39.766666	104.86667	566974
Colorado Springs, CO	38.85	104.75	372437
Dallas, TX	32.783333	96.75	2345815
Fort Worth, TX	32.7253	97.3206	653320
Austin, TX	30.3	97.75	26407
San Antonio, TX	29.45	98.5	1296682
Houston, TX	29.766666	95.38333	23044
Oklahoma City, OK	35.4675	97.5161	537734
Tulsa, OK	36.1539	95.9925	577795
Omaha, NE	41.2586	95.9375	419545

City	latitude (n)	longitude (w)	Population
Kansas City, MO	39.116665	94.55	447306
St. Louis, MO	38.633335	90.23333	1000510
Springfield, MO	37.2153	93.2981	150797
Little Rock, AR	34.7464	92.2894	184422
Memphis, TN	35.1494	90.0489	670902
Louisville, KY	38.2542	85.7594	554496
Bridgeton, MO	38.7535	90.4162	15173
Des Moines, IA	41.6006	93.6089	40885
Minneapolis, MN	44.98	93.2636	372833
St. Paul, MN	44.9444	93.0931	273535
Chicago, IL	41.85	87.65	2833321
Galva, IL	41.1648	90.0437	2676
Champaign, IL	40.1164	88.2433	185682
Indianapolis, IN	39.7683	86.1581	785597
New Orleans, LA	30.05	89.916664	223388
South Bend, IN	41.6833	86.25	104905
Grand Rapids, MI	42.9633	85.6681	193083
Milwaukee, WI	43.05	87.96667	915097
Madison, WI	43.0731	89.4011	223389
Atlanta, GA	33.75	84.416664	486411
Birmingham, AL	33.5206	86.8025	229424
Nashville, TN	36.1658	86.7844	552120
Orlando, FL	28.5381	81.3794	220186
Jacksonville, FL	30.333334	81.65	794555
Tampa, FL	27.95	82.46667	332888
Fort Lauderdale, FL	26.133333	80.13333	185804
Miami, FL	25.7739	80.1939	404048
West Palm Beach, FL	26.715	80.0536	98774
Columbia, SC	34.033333	80.88333	119961
Raleigh, NC	35.816666	78.65	356321
Charlotte, NC	35.2269	80.8433	630478
Detroit, MI	42.366665	83.1	871121
Cincinnati, OH	39.1619	84.457	332252
Cleveland, OH	41.4994	81.6956	444313
Dayton, OH	39.766666	84.183334	156771
Plymouth, MI	42.3711	83.4861	9037
Brookhaven, MI– Traverse city	44.7333	85.5525	14407
Washington DC	38.9	77	581530

City	latitude (n)	longitude (w)	Population
Arlington, VA	38.8864	77.0946	199776
Abingdon, VA	36.7097	81.9775	7933
Dunwoody, GA	33.946	84.334	32808
Greensboro, NC	36.0725	79.7922	236865
Norfolk, VA	36.916668	76.23333	229112
Baltimore, MD	39.3	76.6	787384
Pittsburgh, PA	40.4406	79.9961	312819
Akron, OH	41.0814	81.5192	209704
Harrisburg, PA	40.2736	76.8847	47164
Philadelphia, PA	40	75.13333	1448394
Wayne, PA	40.0439	75.3881	50929
Camden, NJ	39.9258	75.12	517001
New York, NY	40.7142	74.0064	8214426
Rochelle Park, NJ	40.9066	74.0784	5528
Newark, NJ	40.7356	74.1728	281402
Cedar Knolls, NJ	40.823	74.4523	3882
Richmond, VA	40.516666	74.21667	9142
New Brunswick, NJ	40.4861	74.4522	50172
Hamilton Square, NJ	40.2272	74.6536	26419
Freehold, NJ	40.2418	74.2768	11394
Bohemia, NY	40.7689	73.1108	9871
White Plains, NY	41.0339	73.7633	57081
Stamford, CT	41.083332	73.55	119261
Bridgeport, CT	41.1669	73.2053	137912
Hartford, CT	41.7636	72.6856	876927
Albany, NY	42.6525	73.7567	297556
Syracuse, NY	43.0481	76.1478	140658
Rochester, NY	43.166668	77.6	1611581
Buffalo, NY	42.8864	78.8786	276059
Cambridge, MA	42.375	71.1061	101365
Framingham, MA	42.3	71.433334	66910
Worcester, MA	42.266666	71.8	784992
Providence, RI	41.8239	71.4133	635596

Appendix C

End-to-end delay synthesis

Based on the analysis in Section 3.2, we attempt to synthesize network delay traces for trace based simulations in our preliminary research on VoIP QoS routing. A related work on delay synthesis is SS-SVM [118], which is a non-parametric method for generating network delays. [118] claims that the synthetic delay trace generated by SS-SVM is statistically similar to the real trace in terms of the marginal probability distribution of the real delays. Papers [82, 119] investigate simulating Internet delay space by modeling the Internet delay space. However, the previous work [82, 118, 119] cannot synthesize delay traces given only the mean delays, which is the problem we need to solve, i.e. we need to synthesize delay traces with the second set of data (mean delays) obtained from the ISP mentioned in Section 3.2. Therefore, we developed a parametric model of the end-to-end network delay based on the analysis discussed in Section 3.2 to synthesize delay traces. As a result, given the sample mean, we estimate the standard deviation and the minimum delay of a network delay trace, then we estimate the parameters and synthesize new delay traces with the parametric model.

As mentioned in Section 3.2, shifted Gamma distribution gives a good fit to the real delay measurements in the ISP aforementioned. From the same ISP, we obtained a second set of raw measurement data which are 6660 sample mean values (of each 100 RTT measurements) for 154 links (whose end nodes are located in the USA, China, Korea, Japan, Singapore and Malaysia). In order to synthesize a network delay trace that is statistically similar to the original network delay trace based on these 6660 sample mean values, we first estimate the parameters of the shifted Gamma distribution, and rewrite the problem in a formal way as: reconstruct a stationary and ergodic random sequence $\{X_1, X_2, ..., X_M\}$ from the sample mean $\bar{X}$, where X_i, $i = 1, ..., M$, are independently and identically distributed with shifted Gamma distribution with parameters θ, γ, β. To estimate the parameters θ, γ, β, we can first estimate the sample standard deviation S and minimum delay D from $\bar{X}$. Admittedly, this is an under-constrained problem, but we can infer S and D from $\bar{X}$ based on the analysis in Section 3.2, and then estimate the parameters θ, γ, β from $\bar{X}$, S and D. First, since we cannot linearly estimate the minimum delay D from the geographical distance of an overlay link, the minimum delay D can be estimated as the minimum of the sample means in the night and at dawn. Let the estimator of S be $\hat{\mathrm{S}}$. Given the sample mean $\bar{X}$, the best estimator of the sample standard deviation $\hat{\mathrm{S}} = E\{\mathrm{S}|\bar{X}\}$, which is the minimum mean square estimate. It is therefore possible to determine the conditional distribution $Pr\{\mathrm{S}|\bar{X}\}$ from the real delay measurements of the first set of raw data. However, the learned conditional distribution $Pr\{\mathrm{S}|\bar{X}\}$ is inapplicable when a $\bar{X}$ value in the second set of raw data does not occur in the training data. An alternative is to linearly estimate the sample standard deviation S from the sample mean $\bar{X}$ since they have shown strong correlation for most links as seen in Section C. We synthesize $\hat{\mathrm{S}}$ by $\hat{\mathrm{S}} = \alpha(\bar{x} - \mathrm{D}) + k + \omega$, in which the parameter α is tuned to be mostly constant and to vary with $\bar{x} - \mathrm{D}$ for outliers shown in Fig. 3.2, and ω is a very small Gaussian noise to add perturbation to the linear function (refer to the linear fitting in Fig. 3.2). With $\hat{\mathrm{S}}$, $\bar{x}$ and D, we can estimate the parameters of a shifted Gamma distribution as follows:

$$\begin{aligned} \hat{\theta} &= \mathrm{D} \\ \hat{\gamma} &= \frac{(\bar{x} - \mathrm{D})^2}{\hat{\mathrm{S}}^2} \\ \hat{\beta} &= \frac{\hat{\mathrm{S}}^2}{\bar{x} - \mathrm{D}} \end{aligned} \tag{C.1}$$

We can then generate end-to-end delay traces with the shifted Gamma distribution with the parameters $\hat{\theta}, \hat{\gamma}, \hat{\beta}$. As we only know the mean delays, the parameters estimated from the mean delays cannot be very accurate, but the key to the delay synthesis here is to provide spiky delay traces for QoS routing research. Fig. C.1 is an example of the synthetic network delay trace, which is visually similar to the real measured trace in that it also contains delay

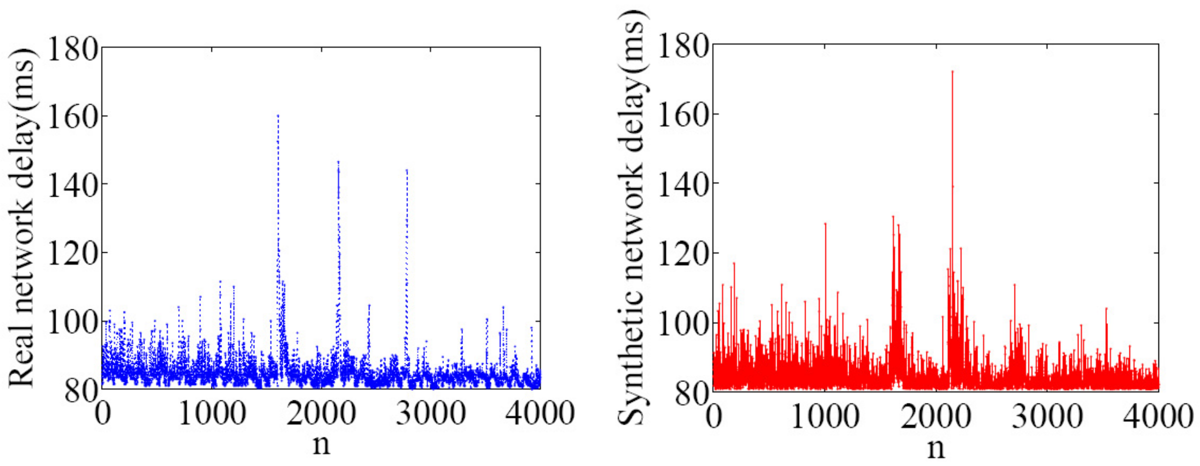


Fig. C.1 Real delays vs. synthetic delays (4000 samples). The synthetic delay trace is visually similar to the real delay trace in that it also contains delay spikes, and therefore can be used in our preliminary research for evaluating QoS routing mechanisms. The KL-divergence and the L_1 norm of the difference between the marginal distributions of the real and synthetic delays are 0.2845 and 0.4374, respectively.

spikes[1], and therefore can be used in our preliminary research for evaluating QoS routing mechanisms. The KL-divergence and the L_1 norm of the difference between the marginal distributions of the real and the synthetic delays are 0.2845 and 0.4374, respectively. The difference in the two delay traces of Fig. C.1 and the KL-divergence measure shows that the method based on the linear estimation is hard to accurately reconstruct the delay trace provided that we only store the sample means of that delay trace. This is a difficult and open problem that could be addressed in future studies.

[1]It is important to synthesize delay spikes because delay spikes can significantly degrade VoIP quality.

Appendix D

Analytical end-to-end delay computation for VoIP SONS

Let the number of nodes in the underlying physical network be n and the underlying background traffic demand be $\mho = [\mho_i]$, which is a 1-by-$n(n-1)$ vector, with $\mho_i$ denoting the traffic demand for the i_{th} source-destination pair in the underlying physical network. Suppose the number of physical links in the underlying router-level network is k. U is an $n(n-1)$-by-k matrix, where $U_{ij} = 1$ if a physical link j is on the shortest hop path for a source-destination pair i in the underlying physical network; otherwise $U_{ij} = 0$. Then the vector $\underline{e} = \mho \cdot U$ represents the background traffic on each physical link.

Suppose call arrival of each user is a Poisson process with arrival rate λ, and the call departure rate is μ. If there are a maximum of N users, and if the number of inactive users at a time instant is K, the total call arrival rate at that time instant is then $K\lambda$, and the total call departure rate is $(N-K)\mu$. Then the number of on-going calls for a source-destination pair forms a birth-death process. The voice traffic demand for all source-destination pairs in the overlay network is denoted by a 1-by-$m(m-1)$ vector $\Lambda = [\Lambda_i]$, with Λ_i representing the voice demand for the i_{th} source-destination overlay node pair.

Let $Q = [Q_{ij}]$ be an $m(m-1)$-by-k matrix, with $Q_{ij} = 1$ if the j_{th} physical link is on the i_{th} overlay link, otherwise, $Q_{ij} = 0$.

Then, given Λ, Π[1] and Q, the nodal voice traffic at each node for destination v is a m-by-1 vector $\gamma^{(v)} = (I - \Pi^{(v)})^{-1}\Lambda_v$. Given $\Gamma = [\gamma^{(v)}]$, the voice traffic on all the physical

[1]It is defined in Section **??**

links can be computed. Let it be denoted as $\underline{b}$.

The traffic on all the physical links is then a vector $\underline{f} = \underline{b} + \underline{e}$. Denote physical link delay as a function of link load $\underline{f}$ and capacity $\underline{c}$, i.e. $\underline{d}(\underline{f}, \underline{c})$, which is a k-by-1 vector. Then the overlay link delay in the overlay network is $L = Q \cdot \underline{d}(\underline{f}, \underline{c})$, which is a $m(m-1)$-by-1 vector. Convert L to an m-by-m matrix A with the rows and columns representing source and destination respectively and $A = [A^{(v)}]$. For destination v, the overlay path delay from all source nodes to destination v is $D^{(v)} = (I - \Pi^{(v)})^{-1} A^{(v)}, v = 1, ..., m$. Then $D = [D^{(v)}]$ is the end-to-end delay matrix of the overlay network.

References

[1] "FCC." http://www.fcc.gov/voip/.

[2] "TeleGeoGraphy." http://www.telegeography.com.

[3] "The E-Model, a computational model for use in transmission planning," *ITU-T Recommendation G.107*, Mar. 2003.

[4] "Cisco." http://www.cisco.com/.

[5] "Specifications for the network voice protocol (NVP)," *IETF RFC 741*, Jan. 1976.

[6] "Vocaltec Inc." http://www.vocaltec.com.

[7] "Resource reservation protocol (RSVP)," *IETF RFC 2205*, Sep. 1997.

[8] "Integrated services in the Internet architecture: an overview," *IETF RFC 1633*, Jun. 1994.

[9] "An architecture for differentiated services," *IETF RFC 2475*, Dec. 1998.

[10] G. R. Ash, "Performance evaluation of QoS-routing methods for IP-based multiservice networks," *Computer Communications*, vol. 26, pp. 817–833, May 2003.

[11] F. Tobagi, "Voice over IP: the challenges behind the vision," in *Proc. Asilomar Conference on Signals, Systems and Computers*, vol. 1, (Monterey , USA), pp. 410–414, Nov. 2004.

[12] R. K. Rajendran, S. Ganguly, R. Izmailov, and D. Rubenstein, "Performance optimization of VoIP using an overlay network," in *Proc. IEEE Infocom*, (Barcelona, Spain), Apr. 2006.

[13] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "1-800-OVERLAYS: using overlay networks to improve VoIP quality," in *Proc. NOSSDAV*, (Skamania, USA), pp. 51–56, Jun. 2005.

[14] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "An overlay architecture for high-quality VoIP streams," *IEEE Trans. Multimedia*, vol. 8, pp. 1250–1262, Dec. 2006.

[15] P. Frossard, J. de Martin, and M. R. Civanlar, "Media streaming with network diversity," *Proc. the IEEE*, vol. 96, pp. 39–53, Jan. 2008.

[16] M. Ghanassi and P. Kabal, "Optimizing Voice-over-IP speech quality using path diversity," in *Proc. IEEE MMSP*, (Victoria, Canada), pp. 155–160, Oct. 2006.

[17] W. Cui, I. Stoica, and R. Katz, "Backup path allocation based on a correlated link failure probability model in overlay networks," in *Proc. ICNP*, (Paris, France), pp. 236–245, Nov. 2002.

[18] "Internet assigned numbers authority (IANA)." http://www.iana.org/assignments/as-numbers/.

[19] M. Coates, A. Hero, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, vol. 19, pp. 47–65, May 2002.

[20] N. G. Duffield and F. L. Presti, "Network tomography from measured end-to-end delay covariance," *IEEE/ACM Trans. Netw.*, vol. 12, pp. 978–992, Dec. 2004.

[21] M. Coates, Y. Pointurier, and M. Rabbat, "Compressed network monitoring for IP and all-optical networks," in *Proc. IMC*, (San Diego, USA), pp. 241–252, Oct. 2007.

[22] H. H. Song, L. Qiu, and Y. Zhang, "Netquest: a flexible framework for large-scale network measurement," *IEEE/ACM Trans. Netw.*, vol. 17, pp. 106–119, Feb. 2009.

[23] G. Wang, B. Zhang, and T. S. E. Ng, "Towards network triangle inequality violation aware distributed systems," in *Proc. IMC*, (San Diego, USA), pp. 175–188, Oct. 2007.

[24] V. S. Borkar and P. R. Kumar, "Dynamic cesaro-wardrop equilibration in networks," *IEEE Trans. Autom. Control*, vol. 48, pp. 382–396, Mar. 2003.

[25] V. Raghunathan and P. R. Kumar, "On delay-adaptive routing in wireless networks," in *Proc. IEEE CDC*, vol. 5, pp. 4661–4666, Dec. 2004.

[26] P. Gupta and P. R. Kumar, "A system and traffic dependent adaptive routing algorithm for ad hoc networks," in *Proc. IEEE CDC*, vol. 3, pp. 2375–2380, Dec. 1997.

[27] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. NIPS 6*, (Denver, USA), pp. 671–678, Dec. 1994.

[28] S. P. M. Choi and D. Yeung, "Predictive q-routing: A memory-based reinforcement learning approach to adaptive traffic control," in *Proc. NIPS 8*, (Denver, USA), pp. 945–951, Dec. 1996.

[29] M. N. Ellanti, S. S. Gorshe, L. G. Raman, and W. D. Grover, *Next Generation Transport Networks: Data, Management, and Control Planes.* Springer, Apr. 2005.

[30] H. J. Kushner and F. J. Vazquez-Abad, "Stochastic approximation methods for systems over an infinitehorizon," *SIAM J. Control Optim.*, vol. 34, pp. 712–756, Mar. 1996.

[31] H. Xie and Y. R. Yang, "A measurement-based study of the Skype peer-to-peer VoIP performance," in *Proc. IPTPS*, (Bellevue, USA), Feb. 2007.

[32] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proc. IEEE Infocom*, (Toronto, Canada), pp. 680–688, Jun. 1994.

[33] "OSPF version 2," *IETF RFC 2328*, Apr. 1998.

[34] "A border gateway protocol 4 (BGP-4)," *IETF RFC 1771*, Mar. 1995.

[35] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of Internet path selection," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 289–299, 1999.

[36] M. Dahlin, B. B. V. Chandra, L. Gao, and A. Nayate, "End-to-end WAN service availability," *IEEE/ACM Trans. Netw.*, vol. 11, pp. 300–313, Apr. 2003.

[37] A. Markopoulou, F. Tobagi, and M. Karam, "Loss and delay measurements of Internet backbones," *Computer Communications*, vol. 29, pp. 1590–1604, Jun. 2006.

[38] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Trans. Netw.*, vol. 6, pp. 515–528, Oct. 1998.

[39] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 293–306, Jun. 2001.

[40] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-path vs. multi-path overlay routing," in *Proc. IMC*, (Miami, USA), pp. 91–100, Oct. 2003.

[41] Y. Zhang and N. Duffield, "On the constancy of internet path properties," in *Proc. ACM IMW*, (San Francisco, USA), pp. 197–211, Nov. 2001.

[42] G. Iannaccone, C. nee Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, “Analysis of link failures in an IP backbone,” in *Proc. ACM IMW*, (Marseille, France), pp. 237–242, Nov. 2002.

[43] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, “Characterization of failures in an IP backbone,” *IEEE/ACM Trans. Netw.*, vol. 16, pp. 749–762, Aug. 2008.

[44] C. Labovitz, A. Ahuja, and F. Jahanian, “Experimental study of Internet stability and backbone failures,” in *Proc. FTCS*, (Madison, USA), pp. 278–285, Jun. 1999.

[45] V. Hilt, A. Hari, and M. Hofmann, “An efficient and robust overlay routing scheme for VoIP,” in *Proc. ICICS*, (Bangkok, Thailand), pp. 508–512, Dec. 2005.

[46] S. Tao, K. Xu, A. Estepa, T. Gao, R. Guerin, J. Kurose, D. Towsley, and Z. L. Zhang, “Improving VoIP quality through path switching,” in *Proc. IEEE Infocom*, vol. 4, (Miami, USA), pp. 2268–2278, Mar. 2005.

[47] H. Li and L. Mason, “Optimal multipath routing with adaptive playback scheduling for VoIP in service overlay networks,” in *IEEE Sarnoff Symposium*, (Princeton, USA), pp. 1–5, Apr. 2008.

[48] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient overlay networks,” *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 131–145, 2001.

[49] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, “The case for resilient overlay networks,” in *Proc. HotOS*, (Elmau, Germany), pp. 152–157, May 2001.

[50] E. Orozco, S. Villette, and A. Kondoz, “Multiple description coding for Voice-over-IP using sinusoidal speech coding,” vol. 1, pp. I–9–I–12, May 2006.

[51] T. Nguyen and A. Zakhor, “Path diversity with forward error correction system for packet switched networks,” in *Proc. IEEE Infocom*, (San Francisco, USA), pp. 663–672, Mar. 2003.

[52] R. Bhandari, *Survivable networks: Algorithms for Diverse Routing.* Springer, Jan. 1999.

[53] G. Li, D. Wang, C. Kalmanek, and R. Doverspike, “Efficient distributed path selection for shared restoration connections,” in *Proc. IEEE Infocom*, vol. 1, (New York, USA), pp. 140–149, Jun. 2002.

[54] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," *IEEE/ACM Trans. Netw.*, vol. 13, pp. 198–211, Feb. 2005.

[55] Z. Ma, H.-R. Shao, and C. Shen, "A new multi-path selection scheme for video streaming on overlay networks," vol. 3, pp. 1330–1334, Jun. 2004.

[56] T. Gomes, J. Craveirinha, and L. Jorge, "An effective algorithm for obtaining the minimal cost pair of disjoint paths with dual arc costs," *Computers and Operations Research*, vol. 36, pp. 1670–1682, May 2009.

[57] C. L. Li, S. McCormick, and D. Simchi-Levi, "Finding disjoint paths with different path costs: complexity and algorithms," *Networks*, vol. 22, pp. 653–667, Oct. 1992.

[58] C. L. Li, S. T. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with Min-Max objective function," *Discrete App. Math.*, vol. 26, pp. 105–115, Oct. 1990.

[59] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On finding disjoint paths in single and dual link cost networks," vol. 1, (Hong Kong, China), pp. 705–715, Mar. 2004.

[60] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, pp. 125–145, Oct. 1974.

[61] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, Oct. 1984.

[62] F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, "The interface message processor for the arpa computer network," in *Proc. AFIPS Joint Comput. Conf.*, (Atlantic City, USA), pp. 551–566, May 1970.

[63] L. G. Mason, "Equilibrium flows, routing patterns and algorithms for store-and-forward networks," *Large Scale Systems*, vol. 8, no. 3, pp. 187–209, 1985.

[64] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. 25, pp. 73–85, Jan. 1977.

[65] S. Vutukury and J. J. Garcia-Luna-Aceves, "A simple approximation to minimum-delay routing," in *Proc. ACM SIGCOMM*, (Cambridge, USA), pp. 227–238, Aug. 1999.

[66] C. G. Cassandras, M. V. Abidi, and D. Towsley, "Distributed routing with on-line marginal delay estimation," *IEEE Trans. Commun.*, vol. 38, pp. 348–359, Mar. 1990.

[67] D. P. Bertsekas, E. M. Gafni, and R. Gallager, "Second derivative algorithms for minimum delay distributed routing in networks," *IEEE Trans. Commun.*, vol. 32, pp. 911–919, Aug. 1984.

[68] T. Guven, C. Kommareddy, R. J. La, M. A. Shayman, and B. Bhattacharjee, "Measurement based optimal multi-path routing," *Proc. IEEE Infocom*, vol. 1, pp. 187–196, Mar. 2004.

[69] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "Cognitive packet networks: Qos and performance," in *Proc. IEEE MASCOTS*, (Fort Worth, USA), pp. 3–9, Oct. 2002.

[70] E. G. Ricardo, R. Lent, and Z. Xu, "Reliable networking with cognitive packets," in *Proc. IEEE MASCOTS*, (San Francisco, USA), pp. 3–12, Aug. 2000.

[71] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Proc. IJCNN*, (Hawaii, USA), pp. 1825–1830, May 2002.

[72] G. Sakellari, "The Cognitive Packet Network: A Survey," *The Computer Journal: Special Issue on Random Neural Networks*, pp. 1–12, Jun. 2009.

[73] "SIP: Session initiation protocol," *IETF RFC3261*, Jun. 2002.

[74] "Packet-based multimedia communications systems," *ITU-T Recommendation H.323*, Jun. 2006.

[75] "Transmission Control Protocol," *IETF RFC 793*, Sep. 1981.

[76] "User Datagram Protocol," *IETF RFC 768*, Aug. 1980.

[77] A. C. Begen, Y. Altunbasak, O. Ergun, and M. A. Begen, "Real-time multiple description and layered encoded video streaming with optimal diverse routing," in *Proc. ISCC*, (Kemer, Turkey), pp. 887–892, Jun. 2003.

[78] S. Qazi and T. Moors, "Scalable resilient overlay networks using destination-guided detouring," *Proc. ICC*, pp. 428–434, Jun. 2007.

[79] J. G. Apostolopoulos and M. D. Trott, "Path diversity for enhanced media streaming," *IEEE Communications Magazine*, vol. 42, pp. 80–87, Aug. 2004.

[80] I. Norros, "On the use of fractional brownian motion in the theory of connectionless networks," *IEEE J. Sel. Areas Commun.*, vol. 13, pp. 953–962, Aug. 1995.

[81] "PlanetLab." http://www.planet-lab.org/.

[82] B. Zhang, T. S. E. Ng, A. Nandi, R. Riedi, P. Druschel, and G. Wang, “Measurement based analysis, modeling, and synthesis of the internet delay space,” in *Proc. IMC*, (Rio de Janeriro, Brazil), pp. 85–98, Oct. 2006.

[83] A. Mukherjee, “On the dynamics and significance of low frequency components of Internet load,” *Internetworking: Research and Experience*, vol. 5, pp. 163–205, Dec. 1994.

[84] C. J. Bovy, H. T. Mertodimedjo, and G. Hooghiemstra, “Analysis of end-to-end delay measurements in internet,” in *Proc. Passive Active Measurement Workshop*, (Fort Collins, USA), pp. 26–33, Mar. 2002.

[85] M. C. Cario and B. L. Nelson, “Autoregressive to anything: Time-series input processes for simulation,” *Oper. Res. Lett.*, vol. 19, pp. 51–58, Aug. 1996.

[86] Y. Liu, F. L. Presti, V. Misra, D. F. Towsley, and Y. Gu, “Scalable fluid models and simulations for large-scale IP networks,” *ACM Trans. Model. Comput. Simul.*, vol. 14, no. 3, pp. 305–324, 2004.

[87] “ns2.” http://www.isi.edu/nsnam/ns/.

[88] L. Jansen, I. Gojmerac, M. Menth, P. Reichl, and P. Tran-Gia, “An algorithmic framework for discrete-time flow-level simulation of data networks,” in *Proc. ITC20*, (Ottawa, Canada), pp. 865–877, Jun. 2007.

[89] A. Gunnar, M. Johansson, and T. Telkamp, “Traffic matrix estimation on a large IP backbone: a comparison on real data,” in *Proc. IMC*, (Taormina, Italy), pp. 149–160, Oct. 2004.

[90] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, “On the self-similar nature of ethernet traffic (extended version),” *IEEE/ACM Trans. Netw.*, vol. 2, pp. 1–15, Feb. 1994.

[91] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, “Measuring ISP topologies with rocketfuel,” *IEEE/ACM Trans. Netw.*, vol. 12, pp. 2–16, Feb. 2004.

[92] G. R. Ash, *Traffic Engineering and QoS Optimization of Integrated Voice & Data Networks (Morgan Kaufmann Series in Networking (Hardcover))*. Morgan Kaufmann Publishers Inc., Oct. 2006.

[93] Y. Zhang, M. Roughan, C. Lund, and D. L. Donoho, “Estimating point-to-point and point-to-multipoint traffic matrices: an information-theoretic approach,” *IEEE/ACM Trans. Netw.*, vol. 13, pp. 947–960, Oct. 2005.

[94] “US Census Bureau.” http://www.census.gov.

[95] N. G. Duffield, C. Lund, and M. Thorup, "Learn more, sample less: control of volume and variance in network measurement," *IEEE/ACM Info. Theory*, vol. 51, pp. 1756–1775, May 2005.

[96] I. Kim and O. de Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Struct Multidisc Optim*, vol. 29, pp. 149–158, Sep. 2004.

[97] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability," in *Journal of the Operational Research Society*, vol. 49, pp. 237–252, Mar. 1998.

[98] R. Mazumdar, L. G. Mason, and C. Douligeris, "Fairness in network optimal flow control: Optimality of product form," *IEEE/ACM Trans. Comm.*, vol. 39, pp. 775–782, May 1991.

[99] J. Y. Boudec, "Rate adaptation, congestion control and fairness: A tutorial," Dec. 2006. http://icapeople.epfl.ch/leboudec.

[100] M. J. Osborne and A. Rubinstein, *A Course in Game Theory.* The MIT Press, Jul. 1994.

[101] D. Fudenberg and J. Tirole, *Game Theory.* The MIT Press, Aug. 1991.

[102] Y.-K. Ng, *Welfare Economics: Towards a More Complete Analysis.* Palgrave Macmillan, Apr. 2004.

[103] P. Ray, "Independence of irrelevant alternatives," *Econometrica*, vol. 41, pp. 987–991, Sep. 1973.

[104] H. Li and L. Mason, "Synthesis of network delays for voice packets in service overlay networks," in *Proc. IEEE/ACM Qshine*, (Vancouver, Canada), Aug. 2007.

[105] Y. J. Liang, E. G. Steinbach, and B. Girod, "Real-time voice communication over the internet using packet path diversity," in *Proc. ACM MULTIMEDIA*, (Ottawa, Canada), pp. 431–440, Sep. 2001.

[106] H. Li and L. G. Mason, "Estimation and simulation of network delay traces for VoIP in service overlay network," in *Proc. IEEE ISSSE*, (Montreal, Canada), pp. 423–425, Jul. 2007.

[107] H. C. S. Thom, "Approximate convolution of the Gamma and mixed Gamma distributions," *Monthly weather review*, vol. 96, pp. 883–886, Dec. 1968.

[108] "Transmission impairments due to speech processing," *ITU-T Recommendation G.113*, Feb. 2001.

[109] M. Thathachar and P. Sastry, *Networks of learning automata : techniques for online stochastic optimization.* Springer, Oct. 2003.

[110] L. G. Mason, "An optimal learning algorithm for s-model environments," *IEEE Trans. Autom. Control*, vol. 18, pp. 493–496, Oct. 1973.

[111]

[112] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms.* The MIT Press, 2 ed., Sep. 2001.

[113] J. Nocedal and S. Wright, *Numerical Optimization.* Springer, Apr. 2000.

[114] F. J. Vázquez-Abad, C. G. Cassandras, and V. Julka, "Centralized and decentralized asynchronous optimization of stochastic discrete event systems," *IEEE Trans. Autom. Control*, vol. 43, pp. 631–655, May 1998.

[115] F. J. Vázquez-Abad and L. G. Mason, "Decentralized adaptive flow control of high-speed connectionless data networks," *Operations Research*, vol. 47, pp. 928–942, Jun. 1999.

[116] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, vol. 40, pp. 118–124, Oct. 2002.

[117] "RTP: A transport protocol for real-time applications," *IETF RFC 1889*, Jan. 1996.

[118] J. Hernandez, I. Phillips, and J. Moguerza, "A SS-SVM approach to generate synthetic network delays," in *Proc. ASMTA*, (Riga, Latvia), pp. 125–131, Jun. 2005.

[119] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson, and R. Steinmetz, "Modelling the internet delay space based on geographical locations," in *Proc. PDP*, (Weimar, Germany), Feb. 2008.