

Distributed Adaptive Diverse Routing for Voice-over-IP in Service Overlay Networks

Hong Li, *Student Member, IEEE*, Lorne Mason, *Senior Member, IEEE*, and Michael Rabbat, *Member, IEEE*

Abstract—This paper proposes a novel mechanism to discover delay-optimal diverse paths using distributed learning automata for Voice-over-IP (VoIP) routing in service overlay networks. In addition, a novel link failure detection method is proposed for detecting and recovering from link failures to reduce the number of dropped voice sessions. The main contributions of this paper are a decentralized, scalable method for minimizing delay on both a primary and secondary path between all pairs of overlay nodes, while at the same time maintaining the link disjointness between the primary and the secondary optimal paths. Simulations of a 50-node model of AT&T's backbone network show that the proposed method improves the quality of voice calls from unsatisfactory to satisfactory, as measured by the R-factor. With the proposed link failure detection mechanism, the time to recover from a link failure is considerably reduced.

Index Terms—Diverse routing, reinforcement learning, voice-over-IP, overlay networks.

I. INTRODUCTION

VOICE-over-IP is an important value-added service on the best effort Internet. The number of Voice-over-IP (VoIP) subscribers has skyrocketed over the past few years, and it is expected to continue increasing in the future. 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 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.

In the current best-effort Internet, shortest hop routing is widely deployed by Internet Service Providers (ISPs) for intra-AS routing. However, shortest hop paths may not provide the best quality for VoIP. A link on the shortest hop paths can be heavily loaded during peak hours, and then voice packets on these paths will experience large queuing delay, jitter, or loss. When a link failure happens, it usually takes tens of minutes or longer for the routers to converge to a new consistent view of the network topology [1]. In addition, variations in network traffic can lead to large delay jitter, and sometimes sudden

delay increases, which are detrimental to VoIP quality. As the current Internet cannot guarantee quality-of-service for VoIP, service overlay networks have been proposed as an alternative approach to provide high quality end-to-end service.

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 (QoS) over the current best-effort Internet. Previous work on QoS routing in overlay networks[2]–[9], have shown the benefit of alternative paths in providing better quality for applications than the direct paths and have shown the benefit of path diversity in overlay networks — using two disjoint paths between the origin and destination improves the perceived performance when one path temporarily experiences congestion.

A. Main Contribution and Organization of the Paper

In this paper, we propose a novel online-distributed-diverse-routing method by answering the following 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?

The main contributions of the work are the following: (1) We present a distributed approach to learn primary and secondary delay-optimal paths for VoIP path diversity using learning automata. (2) We present a novel link failure detection mechanism based on the automata parameters that allows routes through our overlay to rapidly recover after link failures.

The remainder of the paper is organized as follows. Section II presents the related work on QoS routing for VoIP. Section III gives an overview of our approach. Section IV provides a formal problem statement. Section V presents the methodology for learning diverse paths. Section VI describes the active probing and learning method. Section VII presents a scheme for determining primary and secondary optimal paths from learning automata parameters. Section VIII presents the method for link failure detection. Section IX gives the simulation results, and we conclude in Section X.

II. RELATED WORK

In this section, we briefly review the related work on QoS routing for VoIP. In the literature[2]–[6], [8], [10]–[12], brute-force search method, modified distance-vector, or link-state based algorithms are widely used for finding the single-best or diverse paths in overlay networks. Due to combinatorial explosion of the solution space, brute-force search methods typically only consider one-hop overlay paths[6], [10], which are not necessarily the best paths for VoIP. Moreover, these

Manuscript received January 24, 2009; revised August 4, 2009; accepted September 30, 2009. The associate editor coordinating the review of this paper and approving it for publication was M. Brunner.

The authors are with the Electrical and Computer Engineering Department, McGill University, Montreal, Canada (e-mail: viselia.li@mail.mcgill.ca, {lorne.mason,michael.rabbat}@mcgill.ca).

This work was supported by the *Natural Sciences and Engineering Research Council of Canada* (NSERC) under grant STPSC 356934 - 07 and grant SP 208010.

Digital Object Identifier 10.1109/TNSM.2009.P0303.

methods are often centralized and require that link costs be broadcast to central server for estimating path quality[5], [6], [10] or require a-prior knowledge of network model[12], [13]. Consequently, the central server is a single point of failure. Thus, these brute-force search methods are often computationally expensive and unscalable. The distance-vector[2] or link-state[3], [14] based methods require path quality estimation and link quality broadcast. Both approaches require heavy probing to assess path quality, and are thus unscalable also. The work[15] is a scalable by clustering peer users of one Autonomous system to one delegate node, which acts as a relay node for other users in this cluster. It is not a greedy method and seems a scalable solution. However, the performance of this method is far from the optimal.

Diverse path selection has also been studied in the context of survivable networks[13], [16]–[19], [19]–[22], 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[13], [18], [19], which is usually NP-hard[20], [21]. The book [16] gives a thorough review of previously investigated diverse routing algorithms and describes algorithms which, given the topology and the cost of each link, find pairs of edge/node disjoint shortest paths. However, the algorithms in [16] are centralized algorithms which require global knowledge of the network topology and link costs.

The Cognitive Packet Network (CPN) architecture[23]–[25] 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 algorithms. The successful paths found by the “smart” packets are exploited by the “dumb” packets. This method is very similar in spirit to the learning automata-based adaptive routing architecture presented in this paper. However, the route learning algorithm of our framework is much simpler to implement, we prove that the learning parameters converge to the minimum delay paths in a stationary network environment. In addition, the experiments of [23], [24] involve extremely simplistic network scenarios. In contrast, the simulations in this work incorporate non-homogeneous links with different bandwidths, routing node buffer size limits and link propagation delays. Moreover, there is no work on diverse routing or link failure detection based on these reinforcement learning adaptive routing methods[23], [24], [26]–[28], which is the focus of this paper.

III. OVERVIEW OF THE PROPOSED APPROACH

The proposed network architecture is shown in Fig. 1. Service overlay networks consist of two types of nodes: user nodes and overlay nodes. User nodes run VoIP applications. A user node typically connects to the overlay node that is geographically closest. We call this the local overlay node. Overlay nodes are interconnected via virtual connections. During call initialization, a user node connects to its local overlay node and uses the paths provided by the overlay network to communicate with the destination user via the destination’s local overlay node.

As illustrated in Fig. 1, overlay nodes are responsible for identifying routes through the overlay that will provide high

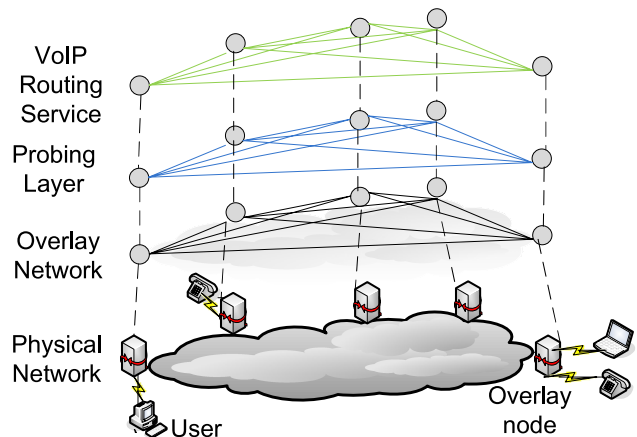


Fig. 1. Proposed network architecture for VoIP QoS provision in overlay networks. Overlay network is formed by interconnected overlay nodes. The probing layer corresponds to the active probing and learning, as described in more detail in Section VI. The VoIP routing service layer corresponds to determining/predicting the optimal *diverse* paths for routing VoIP traffic adaptively, as detailed in the Sections VII and VIII.

quality of service. To accomplish this, they periodically probe alternative paths through the overlay via a measurement plane that feeds information to the VoIP routing service layer. This work builds on our previous work [29], where stochastic automata are used to probe and learn a single minimum delay path in a decentralized manner. Whereas the algorithm proposed in [29] focused on learning a single minimum delay overlay route, in this paper we extend the active probing and learning approach to learn a primary and secondary optimal *diverse* paths. We describe a non-linear rule for predicting the best diverse path before the stochastic automata converges, allowing us to learn and adapt both the primary and secondary paths in an online fashion. In [29], we have proposed four variations on active probing and learning strategies. In this work, two out of the four strategies are found suitable to learn the diverse paths for VoIP QoS routing. To measure overlay path quality, UDP packets are sent periodically between each source and destination overlay node. For the remainder of the paper, when we refer to a node we implicitly mean an overlay node, unless otherwise specified. Probes are used to measure the round trip time of all the possible candidate paths.

Rather than probing all candidate paths in a brute force fashion, the probing process is controlled by a learning automaton. Details of the probing algorithm are provided in Section VI. By using a learning automaton (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 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 selects the next hop on which to forward the probe according to the learning automata parameters.

Each probe follows a round-trip path through the overlay, from origin to destination and back to the origin node again. When the probe returns to the origin node, it updates the learn-

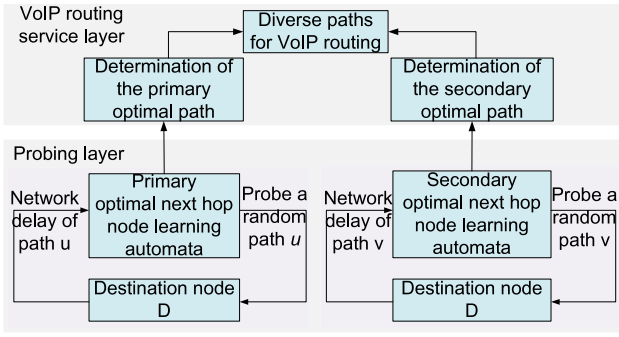


Fig. 2. Proposed overlay architecture block diagram.

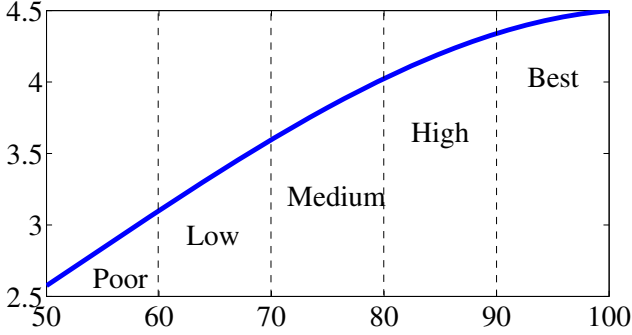


Fig. 3. Quality of voice rating V.S. R-factor and MOS values

ing automata parameters based on the measured Round-Trip Time (RTT). Since round trip time feedback is continuous-valued, typical learning algorithms that are based on binary or discrete-valued feedback, such as the LRI, LReP [30], are not applicable. Instead, we adopt the cross-correlation learning algorithm [31] that has been proposed for positive finite continuous feedback environments.

Such probing and learning, as mentioned above, is implemented for both the primary and the secondary optimal next hop learning automata, as illustrated in Fig. 2. VoIP routing is carried out using the primary and secondary optimal next hop nodes. An end-to-end route can then be set up for VoIP calls during the session initialization phase by choosing each next hop to be the one currently perceived to be optimal, according to the learning automata parameters.

IV. PROBLEM STATEMENT

A. Background

The R-factor[32] is a quantitative metric for measuring the perceived quality of a VoIP conversation, based on characteristics of the underlying connection. It combines transmission parameters related to a connection under consideration such as delay, jitter, and loss rate. A subjective rating system for media (voice, audio or video) after compression and/or transmission is the Mean Opinion Score (MOS), obtained by averaging the scores provided by all the individual testers. A commonly recognized quality of voice rating with the value of R-factor and MOS is shown in Fig. 3. It can be seen that voice quality is rated high or best when R-factor is above 80. Hence, the target R-factor for VoIP quality of service provision can be set to be above 80.

Given that the mouth-to-ear delay and loss are affected by network routing and play-out schemes, the formula for R-factor computation is

$$R = 93.2 - I_d(d) - I_{e-eff}(l) \quad (1)$$

where I_d is a function of the mouth-to-ear delay d and I_{e-eff} is a function of the mouth-to-ear loss l . $I_d(d)$ and $I_{e-eff}(l)$ represent the impairments caused by d and l , respectively.

The mouth-to-ear delay d includes end-to-end network delay d_{net} , play-out buffer delay $d_{playout}$, and codec delays. The mouth-to-ear loss l includes the end-to-end network loss l_{net} and play-out buffer losses $l_{playout}$. Let d_{codec} represent the coding delay. For G.729A, $d_{codec} = 25ms$. Then the mouth-to-ear delay d and loss l can be calculated as follows.

$$\begin{aligned} d &= d_{codec} + d_{net} + d_{playout} \\ l &= l_{net} + (1 - l_{net}) \cdot l_{playout} \end{aligned} \quad (2)$$

When a single path is used for routing, the end-to-end network delay d_{net} and loss l_{net} are simply the path delay and path loss probability. Computation of play-out delay or de-jitter buffer delay $d_{playout}$ and play-out loss or de-jitter buffer loss $l_{playout}$ depends on the implementation of play-out scheduling schemes[33], [34]. From the discussion on R-factor computation above, we can see that it is imperative to choose VoIP paths with low delay, loss and jitter.

B. Problem Formulation

The goal of this research is to find and track the two best disjoint paths so that VoIP quality can be improved with path diversity in overlay networks. To formulate the problem, an overlay network is modeled as a graph $G = (V, E)$, where V is the set of overlay nodes, $V = \{1, 2, \dots, m\}$, 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 with 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 $p_i, p_k \in \mathcal{P}, p_i \neq p_k$, the R-factor $R(p_i, p_k)$ for the pair of paths can be evaluated based on the performance on the two paths. If voice packets are sent on both paths, the network delay of a voice packet would be the minimum of the network delay 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 end-to-end delays on the paths p_i and p_k are d_{p_i} and d_{p_k} respectively, and the end-to-end loss rate on the two paths are l_{p_i} and l_{p_k} respectively. Hence, the end-to-end network delay d_{net} in (2) 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 the pair of paths p_i and p_k that maximize the R-factor $R(p_i, p_k)$, where $R(p_i, p_k)$ is computed with the equations (1) and (2). Then the optimal diverse path routing problem can be formulated as follows:

$$\begin{aligned} &\max R(p_i, p_k), \\ &s.t. \quad p_i, p_k \in \mathcal{P}. \end{aligned} \quad (3)$$

A brute-force solution to (3) is to search over all the possible pairs of disjoint paths, which can be exponentially large. Although straightforward, this approach is not computationally tractable for a large and dynamic network. Hence, a solution that requires no prior knowledge of 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 the paper.

C. Approximation Problem

To differentiate the two paths $p_i, p_k \in \mathcal{P}$, let p_i be a candidate for the primary optimal path and p_k be a candidate for the secondary optimal path, where the end-to-end path delay on p_i is less than that on p_k , i.e. $d_{p_i} \leq d_{p_k}$.

Note that the second path p_k is used to prevent voice calls from being dropped due to link losses or failures on the path p_i . Hence, we expect the performance on the paths p_i and p_k to be uncorrelated when there is no joint link between p_i and p_k . However, completely disjoint paths cannot be guaranteed since we only have control over routing at the overlay level and not actual router-level routing.

Thus, we also try to minimize the number of joint overlay links between p_i and p_k , i.e. $|p_i \cap p_k|$.

Then we convert problem (3) into the following two problems as in (4) and (5):

$$p_i^* = \arg \min_{p_i \in \mathcal{P}} d_{p_i} \quad (4)$$

$$\begin{cases} p_k^* = \arg \min_{p_k \in \mathcal{P}'} d_{p_k} \\ \mathcal{P}' = \{p_k \in \mathcal{P}, |p_k \cap p_i^*| = \min_{p_k \in \mathcal{P}} |p_k \cap p_i^*|\} \end{cases} \quad (5)$$

The problems (4) and (5) are an approximation to the original problem in (3). The benefit of such an approximation is that it is easier to estimate additive delay than to estimate R-factor for a path, especially for the adaptive learning algorithm, as is used in this work. R-factor is often highly correlated with network delay, and losses tend to be highly correlated with delay spikes. The effectiveness of the approximation is validated by the simulation results presented in Section IX.

Solving problem (4) is equivalent to determining the primary optimal path $p_i^* \in \mathcal{P}$ that minimizes the end-to-end delay from the source node S to the destination node D . Then, we solve problem (5) to determine the secondary optimal path $p_k^* \in \mathcal{P}'$. p_k^* is the path with the secondary minimum end-to-end delay from node S to D that has 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.

V. METHODOLOGY FOR DETERMINING THE OPTIMAL DIVERSE PATHS

Fig. 2 illustrates the method for solving the problems in (4) and (5), i.e. the method to determine the optimal diverse paths. It includes two steps. *Step 1*: At each node $S \in V$, probe all its neighbors $j \in V$ to learn if a link (S, j) is on the minimum delay path from S to each destination node D ,

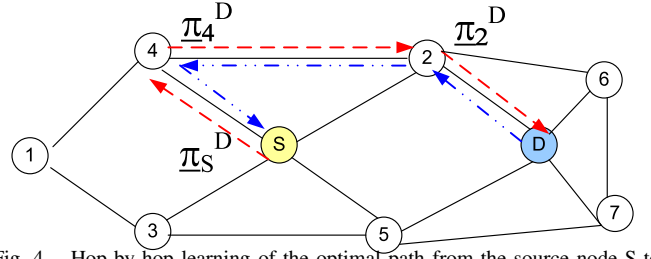


Fig. 4. Hop-by-hop learning of the optimal path from the source node S to the destination node D . The state probability vector at each node is initialized with uniform initialization.

$D \in V$. *Step 2*: Determine the minimum delay path from S to D .

The two steps just mentioned are similar for determining either the primary minimum delay path p_i^* or the secondary minimum delay path p_k^* . However, there are still certain differences in determining p_i^* and p_k^* . To find the primary optimal path p_i^* , which corresponds to solving problem (4), we need to consider all $p_i \in \mathcal{P}$ so that we find a global minimum. To find the secondary optimal path p_k^* (solving (5)) we must find a minimum delay path p_k^* , such that p_k^* has minimum number of joint links with p_i^* .

The next section presents the first step for determining either the primary or the secondary optimal path, The details on how to efficiently search all $p_i \in \mathcal{P}$ to determine the primary optimal path is presented in section VII-A. Section VII-B gives the detail on how to determine the secondary optimal path.

VI. ACTIVE PROBING AND LEARNING

In the overlay network of Fig. 1, every overlay node $S \in V$ runs $(|V|-1)$ or $(m-1)$ active probing and learning automata with each automata responsible for probing the RTT from S to a destination $D \in V, D \neq S$. As the total number of paths from the source S to the destination D is usually very large, it is not efficient to always probe all the paths for determining the minimum delay path. Hence, a learning algorithm was employed to save the cost on probing the bad paths.

In the following subsections, we first present the active probing process in the section VI-A, then we describe the learning algorithm in section VI-B.

A. Active Probing Process

Before describing the active probing process, let us 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 $\pi_{S_j}^D, \forall j \in V$, 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 can define the probability distribution $\underline{\pi}_S^D = [\pi_{S_1}^D, \pi_{S_2}^D, \dots, \pi_{S_j}^D, \dots, \pi_{S_m}^D]$, with $\sum_{j=1}^m \pi_{S_j}^D = 1$ and $\pi_{S_j}^D \geq 0$. The probability distributions $\underline{\pi}_S^D$, are maintained only at the source node S .

With these $\underline{\pi}_S^D$ well defined $\forall S, D \in V$, the active probing process goes as follows. For example, in Fig. 4, the set of nodes $V = \{S, D, 1, 2, 3, 4, 5, 6, 7\}$, a probing packet is sent from the source node S to the destination node D and then back to the source node S to measure the RTT from S to D . The first hop of the probing packet is selected based on the probability distribution $\underline{\pi}_S^D$. In the example of Fig. 4, node 4

is selected with probability π_{S4}^D . Then the probing packet is sent from node S to node 4. At node 4, the above random next hop node selection process is repeated, with a different probability distribution $\underline{\pi}_4^D$. Node 2 is then selected as the next hop of node 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 just mentioned is distributed and scalable. The next hop node of a probing packet is determined locally at each intermediate node. Then the probing packet transits all the next hop nodes to reach its destination. The RTT that the probing packet measures depends on the path formed by the next hop nodes.

B. Cross-correlation learning algorithm

In order to learn the probability distributions $\underline{\pi}_S^D, \forall S, D \in V$, we define $\underline{\pi}_S^D$ as a function of time t , and represent it with $\underline{\pi}_S^D(t)$. Suppose at time t , a probing packet with destination D returns to its source S . The time different between the time it is received and that it was sent is $d_S^D(u)$. u is the first hop that the probing packet transits on its forwarding path from S to D . Then the RTT measurement $d_S^D(u)$ is normalized with respect to a maximum RTT d_{max} .

$$z_S^D(u) = \left(1 - \frac{d_S^D(u)}{d_{max}}\right)^+ \quad (6)$$

to obtain a reward, $z_S^D(u)$ between 0 and 1. Suppose the current probability distribution $\underline{\pi}_S^D(t) = [\pi_{S1}^D(t), \pi_{S2}^D(t), \dots, \pi_{Sj}^D(t), \dots, \pi_{Sm}^D(t)]$. Applying the cross-correlation learning algorithm[31], we have

$$\pi_{Sj}^D(t^+) = \pi_{Sj}^D(t) + g z_S^D(u) (\delta_{ju} - \pi_{Sj}^D(t)), \quad (7)$$

where t^+ is the time instant right after the update at time t , $j = 1, \dots, m$ and g is the learning gain, δ_{ju} satisfies

$$\delta_{ju} = \begin{cases} 1, & \text{if } j = u; \\ 0, & \text{else.} \end{cases} \quad (8)$$

The update in (7) 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 $\pi_{Su}^D(t)$. In (6), d_{max} is predefined to be a value that is suitably large for VoIP calls; we use $d_{max} = 2seconds$ in our simulations. Note that a path receives zero reward when a probe returns with delay greater than or equal to d_{max} on that path.

With the updating formula in (7), we can show that the probability distribution $\underline{\pi}_S^D(t)$ converges to the distribution that satisfies the property given in (9). i.e. If the minimum delay path from S to D is unique, and link (S, j^*) is on the minimum delay path, then

$$\pi_{Sj}^D = \begin{cases} 1, & \text{if } j = j^*; \\ 0, & \text{else.} \end{cases} \quad (9)$$

The convergence is proved in the Appendix. In a non-stationary network environment, we constrain all $\pi_{i,j}^k \geq \epsilon$ for small $\epsilon > 0$ so that changes in network delays can be tracked over time. (If at some point t_0 , $\pi_{i,j}^k(t_0) = 0$ then j will be eliminated as a next hop option on the path from i to k for all future $t \geq t_0$.)

C. Initialization of the Learning Automata

One issue with using the cross-correlation learning algorithm is how to initialize the probability distributions $\underline{\pi}_S^D(t)$. At the starting time 0, the probability of sending a probe from a source node S to a destination node D via a neighboring node j is determined by $\pi_{Sj}^D(0)$.

For probing packets sent from a node $S \in V$ to a destination node $D \in V, D \neq S$, uniform initialization of the probability vector $\underline{\pi}_S^D(0)$ makes it equally likely that any node will be the next hop from S , except S itself;

$$\pi_{Sj}^D(0) = \frac{1}{m-1}, \quad j \neq S, j = 1, \dots, m. \quad (10)$$

When node j receives the probing packet from node S , it may send the probe back to node S , since the next hop choice is made independently at each node. This can introduce random loops in the probing path. It can be shown that there is high probability of random loops on the probing path at the starting stage, and the probability increases with the network size. However, because routes with loops always have higher delay than loop-free paths, the learning automata will automatically learn to avoid these loops [35].

As mentioned above, the uniform initialization method results 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,

$$\pi_{Sj}^D(0) = \frac{I(\mathcal{L}(j, D) < \mathcal{L}(S, D))}{\sum_j I(\mathcal{L}(j, D) < \mathcal{L}(S, D))}, \quad j = 1, \dots, m \quad (11)$$

where $I(\cdot)$ is an indicator function. This initialization guarantees only the nodes whose distances to the destination node D are less than that from the origin node S are explored. In this way, loops are avoided during the whole entire probing process and the initial round trip time can be reduced. As with other iterative optimization algorithms, starting from an initial condition with lower average delay means that we typically converge to a solution quicker than, e.g., from the uniform initialization. However, one should note that this method could potentially eliminate paths which have lower delay, but require taking a step geographically further from the destination in order to achieve this delay.

D. Hop-by-hop learning

We have mentioned how a probing packet is sent from its source to its destination in Section VI-A. Once probe reaches 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 also randomly as what we did in the forward path selection[29]. However, it is not efficient for updating the probability distributions. The most efficient way is to use the hop-by-hop learning strategy illustrated in Fig. 4.

In hop-by-hop learning, a probing packet is sent back to its source by the exact reverse path of its forward paths, which

requires the probing packet records all the intermediate nodes on its forward path. For example, in Fig. 4, the forwarding path for a probing packet from S to D is $\{(S, 4), (4, 2), (2, D)\}$. Then its backward path is $\{(D, 2), (2, 4), (4, S)\}$. Hence, all the probability distributions $\underline{\pi}_S^D(t)$, $\underline{\pi}_4^D(t)$ and $\underline{\pi}_2^D(t)$ are updated according to the probing packet's RTT measurements $d_S^D(4)$, $d_4^D(2)$ and $d_2^D(D)$ respectively as given in (7).

VII. DETERMINING THE PRIMARY AND SECONDARY OPTIMAL PATHS IN A DISTRIBUTED FASHION

The active probing and learning framework presented in the previous section is implemented for both the primary and the secondary optimal next-hop learning, as illustrated in Fig. 2. Assuming that the network is stationary, the primary and the secondary optimal next hops are found when the learning algorithm in Section VI-B converges. Thus the solutions to the problems in equations (4) and (5) are obtained. However, in a *dynamic* network environment, we are more concerned with how to *track* the primary optimal and secondary optimal paths, and *always* choose the best possible pair of paths for voice over IP calls. This is a harder and more important problem to solve than just solving the problem in the equations (4) and (5) 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 has not 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 $\underline{\pi}_S^D(t)$, $\forall S, D \in V$.

In this section, we present the proposed solution to the problem in the equations (4) and (5) that determine (i) the instantaneous primary optimal path p_i^* and (ii) the instantaneous secondary optimal path p_k^* for incoming VoIP calls in a distributed manner.

A. Determine and Track the Primary Optimal Paths

To determine and track the primary optimal path, active probing and learning presented in section VI 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 $\underline{\pi}_S^{(p)D}(t)$. The superscript (p) of $\underline{\pi}_S^{(p)D}(t)$ indicates $\underline{\pi}_S^{(p)D}(t)$ is used for determining the optimal next-hop node of S that is on the *primary* optimal path p_i^* from S to D .

In a stationary network environment, we can run the learning automata for a sufficiently long period of time. Then the primary optimal path for each source-destination pair is determined based on the maximum of the probability distribution $\underline{\pi}_S^{(p)D}(t)$. In fact, as implied by the updating scheme in (7), the value of $\pi_{S_j}^{(p)D}(t)$ give some indication of the ranking of link quality. Therefore, for an incoming voice call session initiated at time t , with source S and destination D and candidate next hop node set $U = \{1, 2, \dots, m\} \setminus \{S\}$, the primary optimal next hop node for node S is given by equation (12):

$$j^* = \arg \max_{j \in U} \pi_{S_j}^{(p)D}(t) \quad (12)$$

Require: Source-destination node pair (S, D) , current time t

- 1: Set the last hop node $u = S$.
- 2: **while** $u \neq D$ **do**
- 3: Set the primary optimal next hop node:
 $j^* = \arg \max_j \pi_{u_j}^{(p)D}(t)$
- 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**

Fig. 5. Primary optimal path determination algorithm

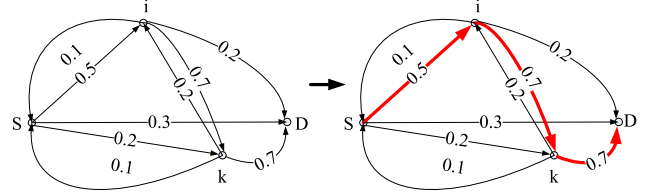


Fig. 6. Illustration of the primary optimal path determination: scenario 1. The left side shows the network with probabilities $\pi_{s_j}^{(p)D}(t)$, labeled on each corresponding link (s, j) , $s, j \in \{S, i, k\}$. The thick arrows in the right side shows the primary optimal paths from node S, i, k to destination D determined by the algorithm in Fig. 5.

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

Note in equation (4), the candidate path set \mathcal{P} for the primary optimal path of a source-destination pair includes all the possible paths between that source-destination pair. That is to say, to solve the problem in equation (4), the primary optimal next hop node learning process has to be initialized with uniform initialization, as described in section VI-C. 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 shortest hop path between that source-destination pair.

Then the algorithm for determining the primary optimal path for VoIP routing with source S and destination D is given in Fig. 5. In a dynamic network environment, the optimal next hop j^* can vary with time, i.e. it is a function of time and the network dynamics. Therefore, while $\underline{\pi}_S^{(p)D}(t)$ is updated with (7), the algorithm in Fig. 5 is able to track the primary optimal path. Then at a time instant t , the primary optimal path for an incoming call is able to be determined. All the voice packets of that call go through this path. As the algorithm in Fig. 5 runs at each overlay node *independently and distributedly*, it is a *scalable* solution for solving the problem (4).

An example of the primary optimal path determination process is illustrated in Fig. 6. The left side of the Fig. 6 shows the network with nodes $\{S, i, k, D\}$ and the probability distributions $\underline{\pi}_v^{(p)D}(t)$ at each node $v \in \{S, i, k\}$ for a

Require: Source-destination node pair (S, D) , primary optimal next hop node j^*

- 1: Set last hop node $u = S$;
- 2: **while** $u \neq D$ **do**
- 3: Set $j' = j^*$;
- 4: **if** There exists a node $j' \neq j^*$ such that $\pi^{(s)}_{uj'}(t) > 0$ **then**
- 5: Divide the value of $\pi^{(s)}_{uj^*}(t)$ into all the other positive $\pi^{(s)}_{uj'}(t)$ to keep $\sum_j \pi^{(s)}_{uj}(t) = 1$.
- 6: Set $\pi^{(s)}_{uj^*}(t) = 0$;
- 7: $j'^* = \arg \max_{j'} \pi^{(s)}_{uj'}(t)$;
- 8: $p_k^* = p_k^* \cup \{(u, j'^*)\}, u = j'^*$;
- 9: **end if**
- 10: **end while**

Fig. 7. Secondary optimal path determination algorithm

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 . Applying the algorithm in Fig. 5, the primary optimal path from each node $v \in \{S, i, k\}$ to the destination D is found, as shown in the thick arrows of Fig. 6.

B. Determine and Track the Secondary Optimal Paths

This section describes how to determine the secondary minimum delay path that has minimal overlap with the primary path. The active probing and learning presented in Section VI is also applied here. 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 track the probability of a potential next-hop node being the *secondary* optimal next hop of S at any time instant t .

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

To satisfy the maximum disjointness constraint as required by (5), where p_k^* and p_i^* are the secondary and the primary optimal path from S to D respectively, we do the following steps. If link (S, j^*) is on the primary optimal path from S to D at time t , we divide the value of $\pi^{(s)}_{Sj^*}(t)$ into all other $\pi^{(s)}_{Sj}(t)$ to ensure $\sum_j \pi^{(s)}_{Sj}(t) = 1, \forall j \in \mathcal{J}$, where \mathcal{J} represents the set of candidate next-hop nodes for the secondary optimal next hop. Then we set the probability $\pi^{(s)}_{Sj^*}(t) = 0$. With the new distribution $\underline{\pi}^{(s)}_S^D(t)$, the secondary optimal next hop at node S for destination D is

$$j'^* = \arg \max_{j' \in \mathcal{J}} \pi^{(s)}_{Sj'}(t). \quad (13)$$

With the above secondary optimal next-hop selection method, the secondary optimal path can then be determined as in the algorithm in Fig. 7.

As we are striving for a *scalable and distributed* solution to the problem in (5), we assume there is no communication between different overlay nodes on their choices of the optimal next hop nodes. Hence, the secondary optimal next-hop node

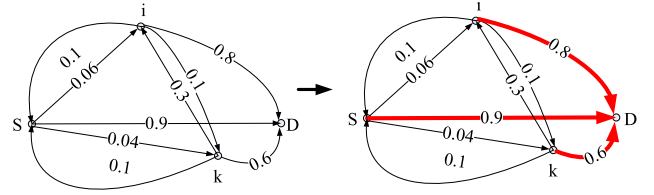


Fig. 8. Illustration of the primary optimal path determination: scenario 2. The left side shows the network with probabilities $\pi^{(p)}_{sj}^D(t)$, $s, j \in \{S, i, k\}$, labeled on each corresponding link (s, j) . The thick arrows in the right side shows the primary optimal paths from S, i, k to destination D determined with the algorithm in Fig. 5.

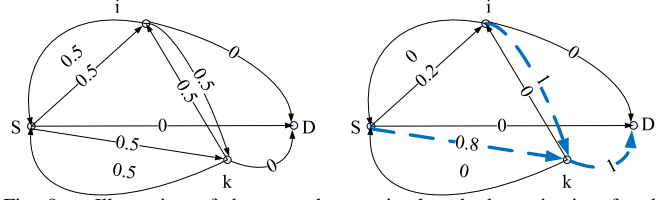


Fig. 9. Illustration of the secondary optimal path determination for the scenario in Fig. 8. The probabilities $\pi^{(s)}_{sj}^D(t)$, $s, j \in \{S, i, k\}$, are labeled on each corresponding link (s, j) in both the left and right figures. In the left side figure, all distributions $\underline{\pi}^{(s)}_S^D(t)$ are uniformly initialized. No secondary optimal path can be determined by the method given in Fig. 7. While in the right side, all distributions $\underline{\pi}^{(s)}_S^D(t)$ are initialized with geographical-location-aware initialization. The thick dashed arrows shows the secondary optimal path from source S, i, k to destination D determined with the algorithm in Fig. 7.

decision at node S for destination D are made *locally and independently* based solely on the local probability distributions $\underline{\pi}^{(p)}_S^D(t)$ and $\underline{\pi}^{(s)}_S^D(t)$.

The advantage of the method is its scalability. However, there is one thing to note about the initialization of $\underline{\pi}^{(s)}_S^D(t)$. In (5), in order to search all the possible paths in \mathcal{P} , which are disjoint with p_i^* , $\underline{\pi}^{(s)}_S^D(t), \forall S, D \in V$, should be uniformly initialized. However, there are cases when no disjoint solution exists, as shown in the example below. Fig. 8 illustrates the primary optimal path decision for a new scenario. The probability distributions $\underline{\pi}^{(p)}_S^D(t)$ are shown in the left side of the figure. The right side shows the primary optimal path determined with the algorithm in Fig. 5. For the scenario of Fig. 6, it is easy to show the method in Fig. 7 can find secondary optimal paths that are completely disjoint with the primary optimal paths with either uniform initialization or geographical-location-aware initialization. However, for the scenario in Fig. 8, no secondary optimal path can be found with the algorithm if $\underline{\pi}^{(s)}_S^D(t)$ is uniformly initialized, as illustrated in the left side of Fig. 9. In this case, we use geographical-location-aware initialization for all $\underline{\pi}^{(p)}_S^D(t)$, which means the set of candidate next hop nodes \mathcal{J} for the secondary optimal next hop determination only include those closer to the destination D than S . As a result, for the scenario in the right side of the Fig. 9, the algorithm in Fig. 9 find the secondary optimal paths that are minimally joint with the primary optimal paths, which is shown in thick dashed arrows.

The algorithm in Fig. 7 is able to track and determine the secondary optimal paths. Combining with the algorithm in Fig. 5, the diverse paths for VoIP routing are determined, as illustrated in Fig. 2. Hence, we have described the proposed solution to the problems in (4) and (5). Simulations of the proposed method are presented in section IX.

Next, we are going to see how to detect link failures to improve the robustness of the proposed routing methods.

VIII. LINK FAILURE DETECTION

When a voice call only uses a single path, if a link on that path fails, then the call will be dropped if the link failure is not repaired in 2 seconds¹. It is important to be able to detect the link failure as early as possible. The primary and the secondary optimal next hop learning algorithms in Fig. 5 and Fig. 7 are able to detect a link failure after the probability of choosing the failed link drops to be smaller than that of other choices. The question here is if we can do better. To improve the robustness of the routing methods in Fig. 5 and Fig. 7, we consider how to detect overlay link failures as early as possible to reduce the number of lost talk spurts. The idea is to detect link failure based on consecutive probability decrease in the distribution $\pi_S^D(t)$.

Suppose at time t , the primary optimal next hop for S with destination D is j^* . i.e. $j^* = \arg \max_j \pi_{Sj^*}^D(t)$.

Proposition 1: For $n(\epsilon) = \log(\epsilon)/\log(1 - \pi_{Sj^*}^D(t))$, if the sequence $\{\pi_{Sj^*}^D(t - n(\epsilon) + 1), \dots, \pi_{Sj^*}^D(t)\}$ is monotone decreasing, a link failure can be declared with $(1 - \epsilon)$ confidence.

Proof: As defined in section VI, the probability that the node j^* is probed at time t is $\pi_{Sj^*}^D(t)$. Hence, the probability that node j^* was not chosen in the past consecutive n time steps $f(n) = \prod_{\tau=t-n+1}^t (1 - \pi_{Sj^*}^D(\tau))$. If the node j^* was probed and rewarded positively during the last n time steps, at least one increase should be observed in the sequence $L = \{\pi_{Sj^*}^D(t - n + 1), \dots, \pi_{Sj^*}^D(t)\}$. However, if a link from j^* to destination D fails, there would be no positive increase observed, because no probing packet returned, in $L = \{\pi_{Sj^*}^D(t - n + 1), \dots, \pi_{Sj^*}^D(t)\}$. In this case, $\pi_{Sj^*}^D(t - n + 1) > \dots > \pi_{Sj^*}^D(t)$. Then $\hat{f}(n) = (1 - \pi_{Sj^*}^D(t))^n \geq f(n)$. Hence, for a sufficiently small $\epsilon > 0$, $n = n(\epsilon) = \log(\epsilon)/\log(1 - \pi_{Sj^*}^D(t))$ guarantees $f(n) < \hat{f}(n) \leq \epsilon$. Or equivalently, the probability that the sequence L has at least one positive increase is $1 - f(n) > 1 - \epsilon$. Therefore, no positive increase in the sequence L indicates a link failure with $(1 - \epsilon)$ confidence. Otherwise, if there is positive increase in the sequence L , it is not a necessary condition for link failure, hence, no failure alarm will be reported. ■

According to *Proposition 1*, for a sufficiently small $\epsilon > 0$, a link failure event can be declared with confidence $(1 - \epsilon)$ when the sequence $L = \{\pi_{Sj^*}^D(t - n + 1), \dots, \pi_{Sj^*}^D(t)\}$ shows consecutive decrease. In Fig. 10, the variable *count* is used to count the number of consecutive link failure events. If such link failure event happens consecutively for N times, a link failure alarm is reported, i.e., $FailAlarm(S, D, j^*) = 1$. The value of ϵ and N can be tuned to trade off the link detection time and the probability of false alarm. The link failure detection method can be run after each update with (7) or less frequently, for example per 1 second interval if a voice gap of 2 seconds is tolerable. This method is applied for link failure detection on secondary optimal paths too.

¹2 seconds is the time taken by voice network signaling protocols to release a connection if there is a failure, so it is an important objective to meet to avoid dropping connections when a failure occurs. If we recover in less than 2 seconds the user will only experience a click but the session continues[36].

Require: Source-destination node pair (S, D) , $\pi_{Sj^*}^D(t)$, ϵ , L , *count*.

Ensure: Link Failure Alarm $FailAlarm$.

```

1:  $n(\epsilon) = \log(\epsilon)/\log(1 - \pi_{Sj^*}^D(t))$ 
2: if Sequence  $L = \{\pi_{Sj^*}^D(t - n(\epsilon) + 1), \dots, \pi_{Sj^*}^D(t)\}$ 
   keeps decreasing then
3:    $count = count + 1$ 
4: else
5:    $count = 0$ 
6: end if
7: if  $count \geq N$  then
8:    $FailAlarm(S, D, j^*) = 1$ 
   //indicate a link failure alarm is reported
9: else
10:   $FailAlarm(S, D, j^*) = 0$ 
   //indicate no link failure
11: end if

```

Fig. 10. Link Failure Detection Algorithm

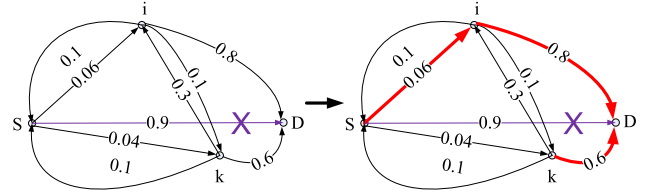


Fig. 11. Primary optimal paths determined when link (S, D) fails. The left side shows the network with probabilities $\pi_{Sj}^D(t)$, $s, j \in \{S, i, k\}$, labeled on each corresponding link (s, j) . The thick arrows in right side show the new primary optimal paths from S, i, k to D after the link failure is detected.

1) Modification on the Primary and Secondary Optimal Path Determination: When a link failure on the primary optimal path is detected at time t with the algorithm in Fig. 10, another rule is needed to recover from the link failure, i.e. to determine which link will be the new primary optimal next hop. The idea is to select the new optimal next hop to be the one without link failure alarm and with as high $\pi_{Sj}^D(t)$ as possible. Although the probability of two physical links fail at the same time is small, one cannot say the probability of two overlay links fail at the same time is equally small, since two overlay links may share the same failed physical link. Therefore, to recover from the link failure, the new primary optimal next hop j must satisfy $FailAlarm(S, D, j) = 0$, and j should have as high $\pi_{Sj}^D(t)$ as possible, since the probability $\pi_{Sj}^D(t)$ implicitly represents the quality of the path with next hop j .

With the link failure recovery rule above, the primary optimal path can be determined for the example in Fig. 11. The left side of the Fig. 11 shows the network with probabilities $\pi_{Sj}^D(t)$, $s, j \in \{S, i, k\}$, labeled on each corresponding link (s, j) . The thick arrows in right side of the Fig. 11 shows the new primary optimal paths from S, i, k to destination D after the link failure is detected. It also shows the detected link failure on link (S, D) . In the right side, the optimal next-hop node for node S is determined as i instead of D in this case. Hence, the link failure is recovered with the new rule when it is detected with the algorithm in Fig. 10.

Modification for the secondary optimal path learning is also

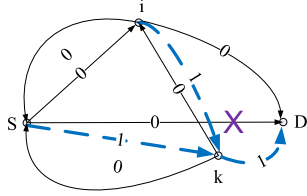


Fig. 12. Secondary optimal path determined when link (S, D) fails. The probabilities $\pi^{(s)D}_{sj}(t)$, $s, j \in \{S, i, k\}$ are labeled on each corresponding link (s, j) . Note that $\underline{\pi}^{(s)D}_S(t)$ are initialized with geographical-location-aware initialization. The thick dashed arrows shows the secondary optimal path from source S, i, k to destination D .

needed to maintain link-disjointness if possible, as well as to detect and recover from link failure. The idea is as follows. First, choose the secondary optimal next hop to be the one without link failure, that has highest $\pi^{(s)k}_{ij}(t)$ and is disjoint with the primary optimal next hop, if possible. Then, to be disjoint with the primary optimal next hop, when the primary optimal next hop changes, the secondary optimal next hop must change also; j^* can be equal to j'^* when all other choices except j^* are not available or have link failures.

Continuing with the example in Fig. 11, the secondary optimal paths when link (S, D) fails is illustrated in Fig. 12. As the failed link is not on the secondary optimal path, the secondary optimal path is the same as that in Fig. 9.

IX. PERFORMANCE EVALUATION

This section evaluates the performance of the VoIP quality on the primary optimal path and on the diverse paths when there is no link failure and when there is a physical link failure.

We simulated a physical network with 50 nodes, modeled after AT&T's backbone network. Overlay nodes are selected randomly to be geographically distributed over the network, and all the overlay networks are interconnected. In the overlay network, we run the primary and secondary optimal next hop learning processes described in section VI, VII and VIII. Based on the probability distribution $\underline{\pi}^{(p)D}_S(t)$ and $\underline{\pi}^{(s)D}_S(t)$, $\forall S, D \in V$, the primary and secondary optimal paths are determined and used to route VoIP calls.

A. 50-node Model of AT&T's Network

1) *Network Topology*: The network topology under study is derived from AT&T's backbone network [37], [38]. It includes 50 Point-of-Presence nodes located in major U.S. cities.

2) *Aggregate Background Traffic Model*: Fractional Brownian motion is a long range dependent self-similar process. In [39], fractional Brownian motion traffic A_t is the accumulated traffic arrival in time $(-\infty, t)$. $A_t = mt + \sqrt{am}B_H(t)$, $t \in (0, \infty)$, where $B_H(t)$ is a normalized fBm process, m is the mean traffic arrival rate, a is the variance coefficient for traffic arrival, and H is the Hurst parameter of $B_H(t)$.

The gravity model[40][41] can be used to model the mean network traffic demand between ingress and egress points of a network. In the gravity model, the traffic demand from an origin node S to a destination node D is proportional to the total traffic leaving node S and that entering node D . Assuming the average traffic generated per person is a constant α , we let the mean traffic demand originating from node S be proportional to the total active population of the metropolitan

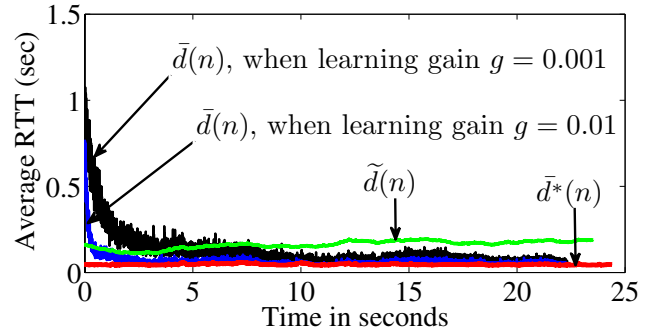


Fig. 13. Average RTT measured by the probing packets between all source-destination pairs versus that on the minimum delay paths and that on the minimum hop paths.

area where node S is located. The variance of the traffic demand between a source-destination pair is set appropriately to introduce realistic queuing delays in the network. The active population is a function of the GMT (Greenwich Mean Time) time, which varies with time zone [40]. Let $M_{SD}(t)$ be the mean traffic demand between node S and node D at GMT time t , and let $P(t, S)$ be the number of active population at node S as a function of GMT time t as measured in [40]. Then the total traffic originating from node S at time t is $\alpha P(t, S)$. Let $G_{SD}(t)$ be the fan-out factor of traffic originating from node S and entering node D ; it is proportional to the number of active population $P(t, D)$ of node D at time t .

$$M_{SD}(t) = \alpha P(t, S) G_{SD}(t) \quad (14)$$

$$G_{SD}(t) = \frac{P(t, D)}{\sum_D P(t, D)} \quad (15)$$

B. Convergence

The active probing and learning process given in Section VI was simulated in the 50-node model of the AT&T's network. When the learning algorithm converges, the paths that the probing packets transit will be equal to the minimum delay paths, as shown in (9). We randomly chose 10 of the 50 nodes to form a full mesh overlay network. The remaining 40 nodes are not part of the overlay network.

Our simulation uses a fluid model, described in more detail below, where time is slotted into 5ms bins. During the n th time slot, each overlay node S send one probing packet $pp(S, D, n)$ to all other destination nodes D in the overlay network. Note that of all the possible next-hop options from S to a given destination D , only one potential next-hop is probed during each time-slot. Let $d_n(S, D)$ denote the RTT measured for probing packet $pp(S, D, n)$, and let $\bar{d}(n)$ denote the average of these delays over all source-destination pairs. Then

$$\bar{d}(n) = \frac{1}{|V|(|V|-1)} \sum_{i \neq j} d_n(i, j) \quad (16)$$

The average RTT for minimum hop paths and minimum delay paths are defined similarly. Denote them by $\bar{d}(n)$ and $\bar{d}^*(n)$, respectively. The mean delays $\bar{d}(n)$ with the learning gain of 0.001 and 0.01 are shown in Fig. 13. It can be seen in Fig. 13 that $\bar{d}(n)$ converges to $\bar{d}^*(n)$ as n increases, which indicates that the probability distributions $\underline{\pi}^{(p)D}_S(t)$ converges to the distribution that satisfies property of 9. The figure also

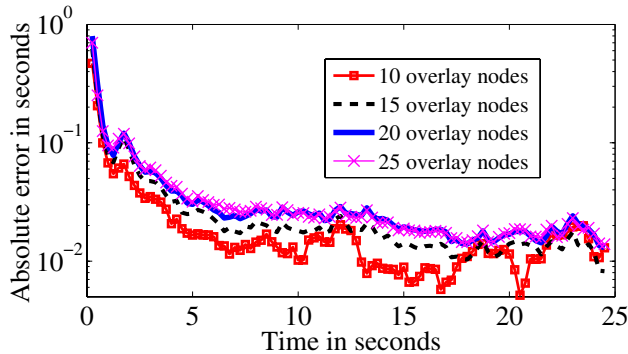


Fig. 14. The difference between the average probing delay on paths determined by learning automata and that on the minimum delay paths. This figure combines the results for the 10, 15, 20 and 25 node overlay networks.

shows the convergence speed increases proportionally as the learning gain g increases from 0.001 to 0.01.

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 network.

Let the difference between the average probing delay on the learned paths $\bar{d}(n)$ and that on the optimal paths $\bar{d}^*(n)$ be the absolute error $\varepsilon(n)$. Then,

$$\varepsilon(n) = \bar{d}(n) - \bar{d}^*(n). \quad (17)$$

For overlay network size of 10, 15, 20 and 25, we computed their absolute errors and combine them into one figure, as given in Fig. 14. It can be seen that the convergence speed remains similar although the overlay network size increases from 10 to 25. The 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^{-2} seconds, or 10ms. Our simulations are conducted at time-scale $\tau = 5\text{ms}$, 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 number of probes required to learn minimum delay paths.

C. VoIP Quality on Primary and Secondary Paths

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 (paths determined by the underlying network, i.e. the Shortest Hop Paths, abbreviated “SHP”), the centralized routing method proposed in [10] (abbreviated “CDR”), and the path switching method proposed in [5] (abbreviated “TaoPS”). The goal of this comparison is to understand if the proposed distributed diverse routing method APL2 can provide competitive and stable VoIP quality as the centralized diverse routing method CDR, and also to show the gain of using the proposed methods in this paper over the existing methods SHP and TaoPS. We

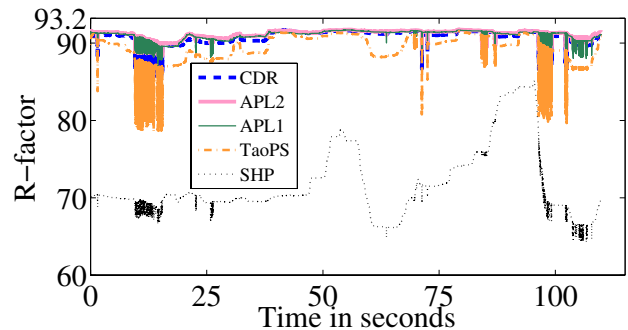


Fig. 15. Average R-factor for all the voice calls in the overlay network when there is no physical link failure. The lines from top to bottom represent the R-factors for APL2, APL1, CDR, TaoPS and SHP, respectively. It can be seen that the three routing methods APL2, APL1 and CDR provide competitive performance and that they are better than TaoPS and much better than SHP.

compare our adaptive methods with TaoPS because TaoPS is also an adaptive routing method for VoIP. 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 instead of source routing is applied (abbreviated APL1+SP). When there are link failures, we also implemented the proposed link failure detection method for APL1 (abbreviated APL1+LFD).

With background traffic generated for the 50-node mode of AT&T network, we evaluate the VoIP quality for the primary optimal path and the diverse paths in the same 10-node overlay network as that for Fig. 13. In the simulation, the voice codec is G.711[32] 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 are set to be similar to [5] as $t_n = 5 \times 2^n$ ms for $n \in \{0, 5, 10, 12, 15\}$.

1) A Scenario When There is No Physical Link Failure:

We first simulated the five VoIP routing methods CDR, APL2, APL1, TaoPS and SHP for a scenario with no physical link failure. The average R-factor for all the voice calls in the overlay network are shown in Fig. 15. 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 because TaoPS switches paths only when the estimated switching gain is sufficiently large and at specific time intervals, but it is still much higher than that of the shortest hop path routing SHP.

The second row of Table. I shows the the average R-factor over the whole simulation time for all source-destination pairs 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 SHP on average. We can also see that APL1 performs slightly better than CDR, which is because CDR only considers single-hop paths. The relative increases in 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.

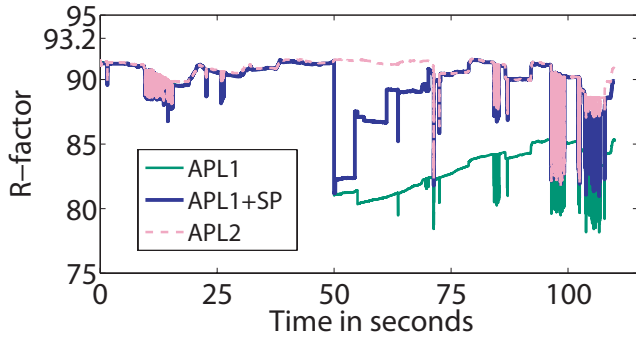


Fig. 16. Average R-factor for all the voice calls in the overlay network when there is a physical link failure. This link failure disables 4 primary optimal paths. From 40 to 72 seconds, we can see 4 jumps in R-factor that are due to the recovery of the 4 paths.

2) A Scenario When There is a Physical Link Failure:

Fig. 15 has shown that there is no significant gain by using diverse routing APL2 or CDR when there is no link failures. We are now interested in if there are benefits by using diverse routing when there is a physical link failure. We disable a physical link 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, as four primary optimal paths fail due to this single physical link failure, which can result in the algorithm in Fig. 5 to take longer to learn new optimal paths. Fig. 16 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 50s, while that of APL1 and APL1+SP drops significantly at 50s. After that, the R-factors for APL1+SP and APL1 grow slowly², 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.

The third row of Table I lists the average R-factor 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, however, as mentioned before, it can lead to out-of-order arrivals and potential route oscillations. The relative 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 provides higher VoIP quality than the case of APL1 where the on-going voice calls are dropped when the link failure occurs at 50 seconds³. However, one should note that the packet re-routing for APL1+SP can result in out-of-order arrivals and may lead to route oscillations when the

²This is because it takes time for the algorithm in Fig. 5 to learn four new primary optimal paths.

³We did not simulate the routing method APL1+SP for the case when there is no link failure. This is because the R-factor for APL1 alone is already close to 90. There is no significant gain to re-route the on-going voice calls.

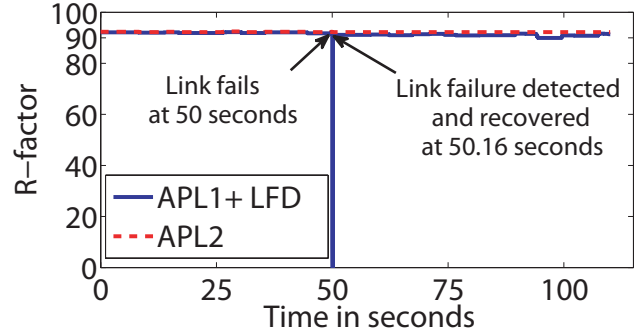
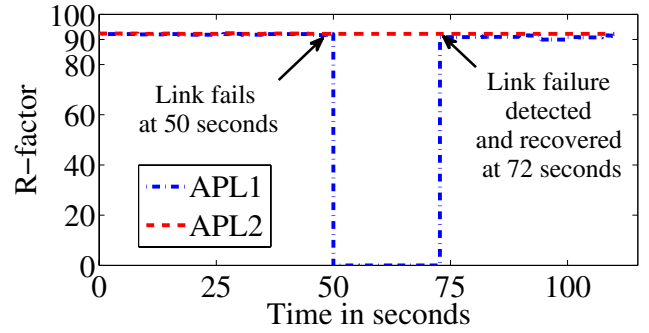


Fig. 17. R-factor comparison for an example source-destination pair for three routing methods: APL2 (both the primary and the secondary optimal paths are used), APL1 (i.e. link failure detection is not implemented) and APL1+LFD (i.e. link failure detection is implemented with $N = 1$, $\epsilon = 10^{-6}$ in Fig. 10). 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.

amount of voice traffic is non-negligible⁴.

3) *Link Failure Detection and Recovery*: In this section, we evaluate the efficiency of the proposed novel link failure detection method proposed in Section VIII. In this simulation, we assume that the same physical link failure as in Fig. 16 fails. The simulation result is shown in Fig. 17, where R-factor of an example source-destination pair is compared for APL1 (i.e. link failure detection is not implemented), APL1+LFD (i.e. link failure detection is implemented) and APL2 (i.e. both the primary and the secondary optimal paths are used).

As shown in Fig. 17, without failure detection, our algorithm discovers a new primary optimal path at $t = 72$ seconds. Consequently, all calls on this optimal path are dropped due to the 22 second gap⁵. When link failure detection is implemented, the failure is detected 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 time to detection, saving potentially dropped calls. It should also be noted that when both the primary and the secondary optimal paths are used, no gap is observed in the R-factor because no voice call is dropped.

Fig. 17 has shown the link failure detection time for an example source-destination pair. This link failure detection time is related the learning gain, the probing rate and the parameters ($\epsilon > 0$ and $N > 0$) in algorithm in Fig. 10. 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

⁴In this example, we assume the amount of voice traffic is small compared to other background traffic, as it is currently and likely to be in the future.

⁵This gap can be reduced by increasing probing rate or learning gain.

	CDR	APL2	APL1	APL1+SP	TaoPS	SHP
R-factor (a scenario when no link failure)	90.79±0.88	91.09±0.57	91.01±0.67	X	89.31±1.96	71.79±4.62
R-factor (a scenario with one fixed link failure)	90.46±1.22	90.79±1.02	85.08±2.96	90.41±1.50	86.65±2.71	71.48±4.52
R-factor (scenarios with one random link failure)	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 I
AVERAGE R-FACTOR OVER THE WHOLE SIMULATION TIME FOR ALL SOURCE-DESTINATION PAIRS.

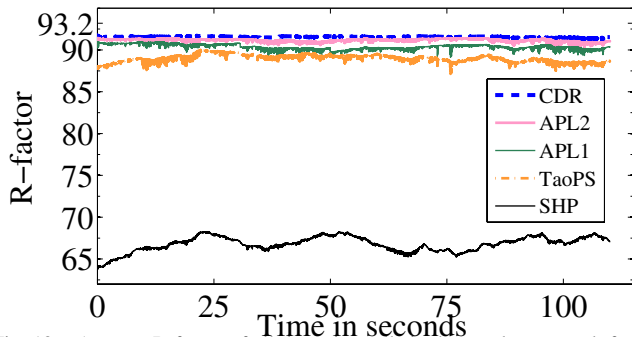


Fig. 18. Average R-factor of all the voice calls in the overlay network for 20 different scenarios (with random link failures). The lines from top to bottom represent the R-factor for CDR, APL2, APL1, TaoPS and SHP, respectively.

the link failures are detected.

4) *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 single physical link failure[13]. The random physical link failures are generated from a link failure model similar to [13], where each link fails with a probability uniformly distributed between $[10^{-4}, 10^{-3}]$. Fig. 18 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 bottom⁶ The 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 the 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” quality VoIP (as illustrated in Fig. 3) and both are good heuristics for the optimal diverse routing problem in (3). It is also observable that CDR performs slightly better than APL2 because the diverse paths chosen by CDR are selected based on mouth-to-ear latency and loss, while those of APL2 are based on network latency only. The R-factors 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.

The average R-factor over time for the six routing methods is listed in the fourth row of Table. I. 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 of [13]). The relative

⁶The R-factors for APL1+SP and APL1+LFD are between that of APL2 and APL1. It is not shown in Fig. 18 because it is very close to that of APL2 and APL1.

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%, respectively⁷. 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 the significant advantage of adaptive routing for improving VoIP quality in overlay networks.

Fig. 17 has shown the link failure detection time for an example source-destination pair. For the example in the 20 scenarios, we also compute the average link failure detection time for the two methods APL1 and APL1+LFD⁸. For APL1, the link failure detection time is $3.67±2.91$ and for APL1+LFD, the link failure detection time is $0.46±0.31$. This shows that the average link failure detection time when the link failure detection algorithm is adopted is 8 times faster than that when it is not used, and that the standard deviation of the link failure detection time is also smaller.

D. Discussion

The advantage of this method over other related work is that it is completely distributed. It does not require any link state broadcast or route update. It is also simple and easy to implement. By sending packets on diverse paths, we can avoid paths that experience temporary congestion or disconnection. If bandwidth of diverse routing is a concern, we can always choose to send only important packets on the secondary path or apply multiple-description encoding on diverse paths.

Similar to other diverse routing problems in optical network design, the R-factor optimization problem is NP-hard. However, simulation results indicate that the R-factor values achieved with our routing method are very close to the maximum R-factor value for G.711 codec, which suggests that our method is a good heuristic for R-factor optimization.

Regarding stability of the approach, flow-based adaptive VoIP routing will not significantly change system stability when VoIP traffic is negligible. It is possible that VoIP paths can oscillate a little when VoIP traffic is significant. However, such oscillation can be regarded as load balancing and in fact, no stable routing pattern actually exists if we consider the dynamic nature of the network environment. The scalability of the approach is mainly derived from the distributed nature

⁷The difference is not as large as that in the third row of Table. I because the random physical link failure in 13 out of 20 scenarios has not resulted in overlay link failure during the simulation time.

⁸Link failures that are not detected before the simulation time ends 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 the link failure time and the simulation ending time.

of the method⁹ and shown by the simulation results in Fig. 14.

X. CONCLUSION

This paper presented a distributed scalable method to identify and track the primary and secondary optimal paths for VoIP in service overlay networks. Our approach is to deploy a learning automata at each overlay node. The learning automata is 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 would be statistically independent, however this is a challenging task to accomplish in practice. Instead, we learn secondary optimal paths that have minimum delay while also being maximally disjoint with the primary optimal path. Simulations on a model of AT&T backbone network indicate that our algorithm can learn the minimum delay path in roughly 5 seconds, while probing at a rate of roughly 32kbps per learning automata. We also demonstrate that, by using diverse paths, in combination with a novel algorithm for detecting link failures, we can robustify the performance of our system to outages.

APPENDIX A

CONVERGENCE OF THE CROSS-CORRELATION LEARNING AUTOMATA

In the appendix, we prove that the learning algorithm defined by equations (6) and (7) converges to the optimal solution defined by (9). Our approach is to first find the Karush-Kuhn-Tucker (KKT) conditions for the delay minimization problem, and then show that asymptotically, the learning algorithm satisfies these conditions.

The minimum delay routing problem for a commodity/user with source-destination pair (S,D) can be formulated as follows. As before, let the graph model for the network be $G = (V, E)$, $V = \{1, \dots, m\}$, $S, j, D \in V$. 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 , 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 . θ_{ij}^k denotes the probability of sending a probing packet from source S to destination D via j . Note 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. Then the user optimization problem for the minimum end-to-end delay routing can be formulated as follows in (19).

$$\begin{aligned} \min_{\theta_{Sj}^D} \quad & \sum_j \theta_{Sj}^D \Delta_{Sj}^D \\ \text{s.t.} \quad & \sum_j \theta_{Sj}^D = 1 \\ & \theta_{Sj}^D \geq 0, \forall j \end{aligned} \quad (18)$$

The KKT conditions for the user optimization problem in (19) are as follows. Let μ_{Sj}^{D*} be the multiplier for $\sum_j \theta_{Sj}^D - 1 =$

0 in KKT condition. Then the KKT condition for θ_{Sj}^{D*} , $j = 1, \dots, m$ to be the optimal solution to 19 is:

$$\begin{cases} \theta_{Sj}^{D*} > 0, \text{ if } \Delta_{Sj}^D = \mu_{Sj}^{D*}; \\ \theta_{Sj}^{D*} = 0, \text{ if } \Delta_{Sj}^D \geq \mu_{Sj}^{D*}. \end{cases} \quad (19)$$

where $\mu_{Sj}^{D*} = \sum_{u=1}^m \Delta_{Su}^D \theta_{Su}^{D*}$. Hence, the user optimal point satisfies

$$\theta_{Sj}^{D*} (\Delta_{Sj}^D - \sum_{u=1}^m \Delta_{Su}^D \theta_{Su}^{D*}) = 0 \quad (20)$$

In the stochastic network environment, according to the Kushner's weak convergence method [43] and following the proof in Vázquez-Abad and Mason's work [44][45], we can derive from the cross-correlation algorithm that as $t \rightarrow \infty$ and the learning gain goes to zero, $\lim_{t \rightarrow \infty} \pi_{Sj}^D(t) = \theta_{Sj}^{D*}$, where θ_{Sj}^{D*} satisfies the following equation:

$$\frac{d\pi_{Sj}^D(t)}{dt} = -\beta \frac{\pi_{Sj}^D(t)}{d_{max}} (\Delta_{Sj}^D(t) - \sum_u \Delta_{Su}^D(t) \theta_{Su}^D(t)) \quad (21)$$

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

For θ_{Sj}^{D*} to be locally stable, it should satisfy that $\frac{d\pi_{Sj}^D(t)}{dt} \Big|_{\theta_{Sj}^{D*}} = 0$ according to the equation (21), which is true given the KKT conditions in (20).

To show the solution is globally stable, let $M_S^D(t) = \sum_{j=1}^m \pi_{Sj}^D(t) \Delta_{Sj}^D$. $M_S^D(t)$ is in fact the dynamic of the objective function in (19). 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

$$\begin{aligned} M_S^D(t^+) - M_S^D(t) &= - \sum_j \pi_{Sj}^D(t) \left((\Delta_{Sj}^D)^2 - \left(\sum_j \pi_{Sj}^D(t) \Delta_{Sj}^D \right)^2 \right) \end{aligned} \quad (22)$$

i.e. $M_S^D(t^+) - M_S^D(t) \leq 0$, since $\sum_j \pi_{Sj}^D(t) \left((\Delta_{Sj}^D)^2 - \left(\sum_j \pi_{Sj}^D(t) \Delta_{Sj}^D \right)^2 \right)$ 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 $\pi_{Sj}^D(t)$, $\forall S, j, D \in V$. Let $\Delta M(t) = \sum_S \sum_D (M_S^D(t^+) - M_S^D(t))$. When $\pi_{Sj}^D(t) = \theta_{Sj}^{D*}$, $\forall S, j, D \in V$, $\Delta M(t) = 0$ and reaches a stationary state.

Hence, when the learning gain g is sufficiently small, the expected delay $M(t)$, 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 (19) is a linear programming problem, the optimal solution θ_{Sj}^{D*} is unique and stable if Δ_{Su}^D , $u = 1, \dots, m$, are distinct. If Δ_{Su}^D are not distinct, the optimal solution may be not unique, but these optimal solutions will give the same value for the objective function.

REFERENCES

- [1] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 293-306, June 2001.
- [2] 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.

⁹It can be shown that the amount of probing traffic for APL1 is about half of the brute-force search method in RON[6] with the same probing rate in a 10-node overlay network[42].

- [3] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "An overlay architecture for high-quality VoIP streams," *IEEE Trans. Multimedia*, vol. 8, no. 6, pp. 1250-1262, Dec. 2006.
- [4] V. Hilt, A. Hari, and M. Hofmann, "An efficient and robust overlay routing scheme for VoIP," in *Proc. ICICS*, Bangkok, Thailand, Dec. 2005, pp. 508-512.
- [5] 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, Mar. 2005, pp. 2268-2278.
- [6] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 131-145, 2001.
- [7] T. Nguyen and A. Zakhor, "Path diversity with forward error correction system for packet switched networks," in *Proc. IEEE Infocom*, San Francisco, USA, Mar. 2003, pp. 663-672.
- [8] S. Qazi and T. Moors, "Scalable resilient overlay networks using destination-guided detouring," *Proc. ICC*, pp. 428-434, June 2007.
- [9] J. G. Apostolopoulos and M. D. Trott, "Path diversity for enhanced media streaming," *IEEE Commun. Mag.*, vol. 42, no. 8, pp. 80-87, Aug. 2004.
- [10] H. Li and L. Mason, "Optimal multipath routing with adaptive playback scheduling for VoIP in service overlay networks," in *IEEE Sarnoff Symp.*, Princeton, USA, Apr. 2008, pp. 1-5.
- [11] M. Seshadri and Y. H. Katz, "Dynamics of simultaneous overlay network routing," *UCB/CS03-1291, Tech. Rep.*, Nov. 2003.
- [12] C. Tang and P. K. McKinley, "Improving multipath reliability in topology-aware overlay networks," in *Proc. ADSN*, Columbus, USA, June 2005, pp. 82-88.
- [13] 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, Nov. 2002, pp. 236-245.
- [14] S. Misra and B. J. Oommen, "Dynamic algorithms for the shortest path routing problem: Learning automata-based solutions," *IEEE Trans. Sys. Man, Cybernetics-Part B*, vol. 35, no. 6, pp. 1179-1192, Dec. 2005.
- [15] S. Ren, L. Guo, and X. Zhang, "Asap: An as-aware peer-relay protocol for high quality voip," in *Proc. ICDCS*, Lisboa, Portugal, July 2006, pp. 70-70.
- [16] R. Bhandari, *Survivable networks: Algorithms for Diverse Routing*. Springer, Jan. 1999.
- [17] G. Li, D. Wang, C. Kalmanek, and R. Doverspike, "Efficient distributed path selection for shared restoration connections," in *Proc. IEEE Infocom*, vol. 1, NY, USA, June 2002, pp. 140-149.
- [18] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 198-211, Feb. 2005.
- [19] T. Gomes, J. Craveirinha, and L. Jorge, "An effective algorithm for obtaining the minimal cost pair of disjoint paths with dual arc costs," *Computers Operations Research*, vol. 36, no. 5, pp. 1670-1682, May 2009.
- [20] 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, no. 1, pp. 105-115, Oct. 1990.
- [21] 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, Mar. 2004, pp. 705-715.
- [22] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125-145, Oct. 1974.
- [23] E. Gelenbe, R. Lent, A. Montuori, and Z. Xu, "Cognitive packet networks: Qos and performance," in *Proc. IEEE MASCOTS*, Fort Worth, USA, Oct. 2002, pp. 3-9.
- [24] E. G. Ricardo, R. Lent, and Z. Xu, "Reliable networking with cognitive packets," in *Proc. IEEE MASCOTS*, San Francisco, USA, Aug. 2000, pp. 3-12.
- [25] G. Sakellari, "The Cognitive Packet Network: A Survey," *Computer J.: Special Issue Random Neural Netw.*, pp. 1-12, June 2009.
- [26] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Proc. IJCNN*, HI, USA, May 2002, pp. 1825-1830.
- [27] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. NIPS 6*, Denver, USA, Dec. 1994, pp. 671-678.
- [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, Dec. 1996, pp. 945-951.
- [29] H. Li, L. Mason, and M. Rabbat, "Learning minimum delay paths for VoIP in service overlay networks," in *Proc. IEEE NCA*, Cambridge, USA, July 2008, pp. 271-274.
- [30] M. Thathachar and P. Sastry, *Networks of learning automata: techniques for online stochastic optimization*. Springer, Oct. 2003.
- [31] L. G. Mason, "An optimal learning algorithm for s-model environments," *IEEE Trans. Autom. Control*, vol. 18, no. 5, pp. 493-496, Oct. 1973.
- [32] "The E-Model, a computational model for use in transmission planning," *ITU-T Recommendation G.107*, Mar. 2003.
- [33] 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, June 1994, pp. 680-688.
- [34] M. Ghanassi and P. Kabal, "Optimizing Voice-over-IP speech quality using path diversity," in *Proc. IEEE MMSP*, Victoria, Canada, Oct. 2006, pp. 155-160.
- [35] 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.
- [36] 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.
- [37] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2-16, Feb. 2004.
- [38] 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.
- [39] I. Norros, "On the use of fractional brownian motion in the theory of connectionless networks," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 6, pp. 953-962, Aug. 1995.
- [40] 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, Oct. 2004, pp. 149-160.
- [41] 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, no. 5, pp. 947-960, Oct. 2005.
- [42] H. Li, "Qos routing for voice-over-ip in service overlay networks," Ph.D. dissertation, McGill University, Montreal, Canada, Oct. 2009.
- [43] H. J. Kushner and F. J. Vazquez-Abad, "Stochastic approximation methods for systems over an infinite horizon," *SIAM J. Control Optim.*, vol. 34, no. 2, pp. 712-756, Mar. 1996.
- [44] 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, no. 5, pp. 631-655, May 1998.
- [45] F. J. Vázquez-Abad and L. G. Mason, "Decentralized adaptive flow control of high-speed connectionless data networks," *Operations Research*, vol. 47, no. 6, pp. 928-942, June 1999.

Hong Li is a PhD candidate of Electrical and Computer Engineering at McGill University. Her research interest is multimedia communication, statistical signal processing, game theory, time series analysis and machine learning.

Lorne Mason is a Professor in the Electrical and Computer Engineering department, McGill University, and Professor Honouaire at INRS-EMT, University of Quebec. He has taught at the undergraduate and post graduate level and conducted research in the areas of network control, performance analysis, network design and planning, with more than 100 publications in leading journals and conferences. His current research interests include Internet traffic engineering and network control, networking games, and optical network design. He was co-recipient of the 2006, NSERC Synergy Award for research on Agile-All Photonic Networks. He was a co-recipient of the prestigious STENTOR research award for collaborative research in telecommunications in 1993, for his contribution to state dependent routing. He is the Canadian representative to the Telecommunications Committee of IFIP, and a senior member of the IEEE.

Michael Rabbat is an Assistant Professor of Electrical and Computer Engineering at McGill University. His research interests are in the areas of computer networking, statistical signal processing, and statistical machine learning.