Latency Equalization as a New Network Service Primitive

Minlan Yu, *Student Member, IEEE*, Marina Thottan, *Member, IEEE, ACM*, and Li (Erran) Li, *Senior Member, IEEE*

***Abstract*—Multiparty interactive network applications such as teleconferencing, network gaming, and online trading are gaining popularity. In addition to end-to-end latency bounds, these applications require that the *delay difference* among multiple clients of the service is minimized for a good interactive experience. We propose a *Latency EQualization (LEQ) service*, which equalizes the perceived latency for all clients participating in an interactive network application. To effectively implement the proposed LEQ service, network support is essential. The LEQ architecture uses a few routers in the network as *hubs* to redirect packets of interactive applications along paths with similar end-to-end delay. We first formulate the hub selection problem, prove its NP-hardness, and provide a greedy algorithm to solve it. Through extensive simulations, we show that our LEQ architecture significantly reduces delay difference under different optimization criteria that allow or do not allow compromising the per-user end-to-end delay. Our LEQ service is incrementally deployable in today's networks, requiring just software modifications to edge routers.**

***Index Terms*—Algorithm, interactive network applications, latency equalization (LEQ), next-generation network service.**

I. Introduction

THE INCREASED availability of broadband access has spawned a new generation of netizens. Today, consumers use the network as an interactive medium for multimedia communications and entertainment. This growing consumer space has led to several new network applications in the business and entertainment sectors. In the entertainment arena, new applications involve multiple users participating in a single interactive session, for example, online gaming [1] and online music (orchestra) [2]. The commercial sector has defined interactive services such as bidding in e-commerce [3] and telepresence [4]. Depending on the number of participants involved, interactive applications are sensitive to both end-to-end delay and delay difference among participants. Minimizing the delay difference among participants will enable more real-time interactivity.

End-to-end delay requirements can be achieved by traffic engineering and other QoS techniques. However, these approaches are insufficient to address the needs of multiparty interactive network applications that *require bounded delay difference across multiple clients to improve interactivity* [5]. In online gaming, the delay difference experienced by gamers significantly impacts game quality [6]–[8]. To improve the interactive experience, game servers have even implemented mechanisms by which participating players can vote to exclude players with higher lag times. In distributed live music concerts [2], individual musicians located at different geographic locations experience perceptible sound impairments introduced by latency differences among the musicians, thus severely degrading the quality of the music. In e-commerce, latency differences between pairs of shopping agents and pricing agents can result in price oscillations leading to an unfair advantage to those pairs of agents who have lower latency [3].

Previous work on improving online interactive application experiences considered application-based solutions either at the client or server side to achieve equalized delay [9]–[11]. Client-side solutions are hard to implement because they require that all clients exchange latency information to all other clients. They are also vulnerable to cheating [7]. Server-side techniques rely on the server to estimate network delay, which is not sufficiently accurate [12] in some scenarios. Moreover, this delay estimation places computational and memory overhead on the application servers [13], which limits the number of clients the server can support [1]. Previous studies [8], [14]–[16] have investigated different interactive applications, and they show the need for network support to reduce delay difference since the prime source of the delay difference is from the network. The importance of reducing latency imbalances is further emphasized when scaling to wide geographical areas as witnessed by a press release from AT&T [17].

In this paper, we design and implement *network-based* Latency EQualization (LEQ), which is a service that Internet service providers (ISPs) can provide for various interactive network applications. Compared to application-based latency equalization solutions, ISPs have more detailed knowledge of current network traffic and congestion, and greater access to network resources and routing control. Therefore, ISPs can better support latency equalization routing for a large number of players with varying delays to the application servers. This support can significantly improve game experience, leading to longer play time and thus larger revenue streams.

Our network-based LEQ service provides equalized-latency paths between the clients and servers by redirecting interactive application traffic from different clients along paths that minimize their delay difference.[1] We achieve equalized-latency

Manuscript received March 25, 2010; revised October 07, 2010 and December 14, 2010; accepted May 09, 2011; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z. M. Mao.

M. Yu is with Princeton University, Princeton, NJ 08540 USA (e-mail: minlanyu@cs.princeton.edu).

M. Thottan and L. Li are with Bell Laboratories, Murray Hill, NJ 07974 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2155669

[1]Only interactive application traffic uses LEQ routing; other traffic can use the default routing mechanism.

paths by using a few routers in the network as *hubs*, and interactive application packets from different clients are redirected through these hubs to the servers. Hubs can also be used to steer packets away from congested links. Since the redirection through the hubs is implemented through IP encapsulation, our hub routing mechanism can be deployed in today's existing routing infrastructure.

Our LEQ architecture provides a flexible routing framework that enables the network provider to implement different delay and delay difference optimization policies in order to meet the requirements of different types of interactive applications. In one policy scenario, latency equalization among different interactive clients can be achieved without compromising the end-to-end delay of individual clients. This is because in some networks where OSPF weights are typically used to optimize traffic engineering objectives and not just delay [18], the default network path may not correspond to the lowest end-to-end latency due to path inflation [19] or due to transient network congestion. Akamai's SureRoute service [20] and other research [21], [22] show that overlay paths can be used to reduce end-to-end latency by getting around congestion and network failures. Similar to these works, our LEQ routing can minimize delay difference without compromising the end-to-end delay. In the other policy scenario, if the application can tolerate some moderate increase in the end-to-end delay, it is possible to achieve even better latency equalization among clients.

To achieve LEQ routing, we formulate the *hub selection problem*, which decides which routers in the network can be used as hubs and the assignment of hubs to different client edge routers to minimize delay difference. We prove that this hub selection problem is NP-hard and inapproximable. Therefore, we propose a greedy algorithm that achieves equalized-latency paths. Through extensive simulation studies, we show that our LEQ routing significantly reduces delay difference in different network settings (e.g., access network delay and multiple administrative domains).

The paper is organized as follows. Section II motivates the need for network support for interactive applications. Section III describes the LEQ architecture and its deployment issues. Section IV provides our theoretical results and algorithms for the hub selection problem. Section V evaluates the LEQ architecture and algorithms in different network settings and under both static and dynamic scenarios. Sections VI and VII discuss related work and conclude the paper.

II. Motivation for Network-based Latency Equalization Support

To achieve equalized delay for interactive applications, previous approaches are implemented either at the client or server side without any network support. We use online gaming as an example to discuss the limitations of these approaches.

Client-side latency compensation techniques are based on hardware and software enhancements to speed up the processing of event updates and application rendering. These techniques cannot compensate for network-based delay differences among a group of clients. Buffering event update packets at the client side is hard to implement because this requires the coordination of all the clients regarding which packets to buffer and for how long. This leads to additional measurement and communication overhead and increased application delay [23]. Some gaming clients implement *dead reckoning*, a scheme that uses previously received event updates to estimate the new positions of the players. Dead reckoning has the drawback that the prediction error increases significantly with increasing network delays. In one racing game, where estimating the position of the players is critical, it was shown that the average prediction error using dead-reckoning was 17 cm for a delay of 100 ms and 60 cm for a delay of 200 ms, a factor of 3.5 [24]. Client-side solutions are also prone to cheating. Players can hack the compensation mechanisms or tamper with the buffering strategies to gain unfair advantage in the game [7].

Due to the problems of client-side solutions, several delay compensation schemes are implemented at the server side. However, while introducing CPU and memory overhead on the server, they still do not completely meet the requirements of fairness and interactivity. For example, with the *bucket synchronization* mechanism [9], the received packets are buffered in a bucket, and the server calculations are delayed until the end of each bucket cycle. The performance of this method is highly sensitive to the bucket (time window) size used, and there is a tradeoff between interactivity versus the memory and computation overhead on the server. In the *time warp synchronization* scheme [10], snapshots of the game state are taken before the execution of each event. When there are late events, the game state is rolled back to one of the previous snapshots, and the game is reexecuted with the new events. This scheme does not scale well for fast-paced, high-action games because taking snapshots on every event requires both fast computation and large amounts of fast memory, which is expensive [23]. In [11], a game-independent application was placed at the server to equalize delay differences by constantly measuring network delays and adjusting players' total delays by adding artificial lag. However, experiments in [12] suggest that using server-based round-trip-time measurements to design latency compensation across players fails in the presence of asymmetric latencies.

Based on the above limitations of the end-system based techniques, we conclude that it is difficult for end-hosts and servers to compensate for delay differences without network support. In a survey of online gamers [16], 85% of the users requested additional network state information to improve game quality, clearly demonstrating the inefficiency of existing techniques. Thus, there is a pressing need to provide a network support for latency equalization as a general service to improve user experience for all interactive network applications. With network support for LEQ, the network delay measurement can be offloaded from the application server and performed more accurately. The network-based solutions can achieve LEQ and also react faster to network congestion or failure. By providing LEQ service, ISP can gain larger revenue through significantly improving the user experience of interactive applications.

Network support for LEQ is complementary to server-side delay compensation techniques. Since network-based LEQ service can reduce both delay and delay difference among participants of the interactive applications, the application servers can better fine-tune their performance. For example, in the case

of gaming applications, the servers can use smaller bucket sizes in bucket synchronization, or use fewer snapshots in time warp synchronization. Therefore, for the same memory footprint, servers can increase the number of concurrent clients supported and improve the quality of the interactive experience.

III. Latency Equalization Architecture

In this section, we first present the deployment scenario of LEQ routing in a single administrative domain. We achieve LEQ routing by selecting a few routers as hubs and directing interactive application traffic through these hubs. Next, we extend the basic LEQ architecture to support access network delay and multiple administrative domains (e.g., across a content distribution network and ISPs).

A. Illustration of Basic LEQ Hub Routing

Our deployment scenario is within an ISP network. The ISP can leverage the proposed LEQ routing architecture to host multiple interactive applications or application providers on the same network. The network-based LEQ architecture is implemented using a hub routing approach: *Using a small number of hubs in the network to redirect application packets, we equalize the delays for interactive applications*. To explain the basic LEQ architecture, we consider a single administrative domain scenario and focus on equalizing application traffic delays between the different client edge routers and the server edge routers without considering access delay. Based on the application's LEQ requirements, the application traffic from each client edge router is assigned to a set of *hubs*. Client edge routers redirect the application packets corresponding to the LEQ service through the hubs to the destined servers. By redirecting through the hubs, application packets from different client edge routers with different delays to the servers are guaranteed to reach the servers within a bounded delay difference.

For example, in Fig. 1, client traffic from an interactive application enters the provider network through edge routers $R1$ and $R2$. The server for the interactive application is connected to the network through edge router $R10$. Using the LEQ routing architecture, $R6$ and $R7$ are chosen as hubs for $R1$, and $R7$ and $R8$ are chosen as hubs for $R2$. Using redirection through hubs, $R1$ has two paths to the server edge router $R10$: $R1 \rightarrow R6 \rightarrow R10$ and $R1 \rightarrow R7 \rightarrow R10$, both of which have a delay of 10 ms. $R2$ also has two paths: $R2 \rightarrow R7 \rightarrow R10$ and $R2 \rightarrow R8 \rightarrow R10$, whose delay is also 10 ms. Thus, LEQ is achieved by optimized hub selection and assignment.[2] Each client edge router is assigned to more than one hub, so it has the flexibility to select among its assigned hubs to avoid congestion. For example, in Fig. 1, $R1$ and $R2$ are both assigned two hubs.

To illustrate the advantage of our LEQ routing concept on real networks, we conducted experiments on the Abilene [25] network in VINI test bed [26] as shown in Fig. 2. We set up a server at the Washington DC node and measured the delay

[2]In contrast, since OSPF weights are set for traffic engineering goals [18] and not necessarily the shortest latency paths, if we assume the default paths based on OSPF weights are $R1 \rightarrow R3 \rightarrow R9 \rightarrow R10$ (14 ms) and $R2 \rightarrow R5 \rightarrow R8 \rightarrow R10$ (10 ms), the delay difference is 4 ms. Our evaluations on large ISP networks in Section V show that we can reduce delay difference compared to latency-based shortest path routing.

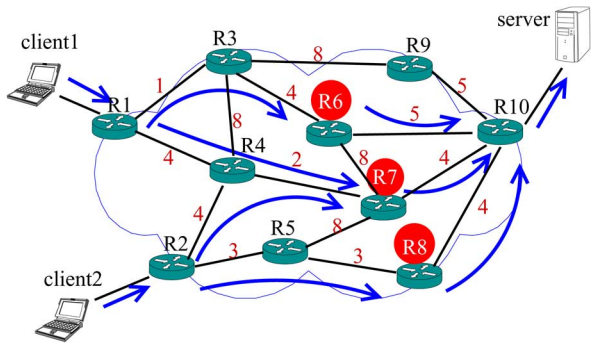


Fig. 1. LEQ hub routing example ($R1 \ldots R10$ are routers in the network. For simplicity, we only consider client edge routers $R1, R2$ and server edge router $R10$. The numbers on the links represent the latency (ms) of each link. $R6$, $R7$ and $R8$ are the hubs for $R1$ and $R2$.)

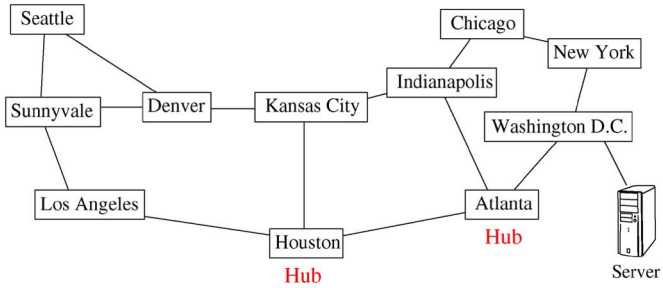


Fig. 2. LEQ routing on Abilene network.

difference among players at the 11 different sites using ping packets. The average delay difference using the default shortest path routing was 58 ms. We then evaluated the performance of LEQ routing using two hubs, one at Atlanta, GA, and the other at Houston, TX, and measured the average delay difference for packets from different sites to the server through these two hubs. The average delay difference is 25 ms using the LEQ architecture, which is a 56% reduction compared to shortest path routing. A more detailed analysis of the LEQ architecture using various network settings is provided in Section V.

B. LEQ Routing Architecture

To implement the hub routing idea, our LEQ architecture involves three key components.

LEQ Service Manager: The LEQ service manager serves as a centralized server to decide hub selection and assignment. We choose an *offline* hub selection algorithm. This is because an online hub selection algorithm would require significant monitoring overhead and fast online path calculation to keep pace with client dynamics (clients join and leave applications) and network dynamics (failures and transient network congestion). The offline algorithm assumes the presence of clients at all edge routers. The inputs to the algorithm are the server edge router locations, network topology, and the propagation delay. The service manager selects a group of routers to serve as hubs for each client edge router and sends this information of the assigned hubs (IP addresses) to the client edge routers.

Hubs: Hubs are just conventional routers that are either in the core network or at the edge. Packet indirection through hubs

is implemented through IP encapsulation without any changes to today's routing architecture. Since we consider all edge routers as potential client edge routers, the selected hub nodes can be shared among different interactive applications. Our evaluations in Section V on ISP network topologies show that we are able to find enough equalized-delay paths by redirecting packets through hubs, and that we only require a few (5–10) hubs to achieve latency equalization. The evaluations also show that the LEQ architecture with a few hubs scales well with increase in the number of servers.

Edge Routers: The edge router first identifies interactive application packets by their port number and server IP addresses (known to the service provider in advance), and then redirects the packets via one of its assigned hubs. Each edge router monitors the end-to-end delay to the server edge routers through each of its assigned hubs. When congestion is detected along a path, the client edge router can redirect packets to another assigned hub to get around the point of congestion.

Using the above LEQ routing architecture, we can implement different delay and delay difference optimization policies to meet the requirements of different types of interactive applications. We can either reduce delay difference without compromising the end-to-end delay of individual application users, or further improve the overall latency equalization performance for the application at the cost increasing the end-to-end delays of a few application users.[3] The evaluations of LEQ routing performance under different optimization policies are shown in Section V.

C. Benefits of LEQ Architecture

No Significant Impediments to Deployment: Our analysis on ISP network topologies shows that the LEQ architecture requires only a few routers to serve as hubs. Hubs are just conventional routers in the network, and it is not necessary to modify any underlying routing protocol. The architecture only requires minimal functions on the edge router such as application packet identification and end-to-end path monitoring. The LEQ architecture can even be implemented as an overlay on the underlying routing infrastructure. Our LEQ architecture is incrementally deployable. We show that even with one hub, we can reduce the delay difference by 40% on average compared with shortest path routing.

Handling Network Failure and Congestion: The LEQ architecture assigns multiple hubs for each client edge router so that the client edge routers can select hubs to get around network failure and congestion. Since the LEQ service is supported by the service provider, we assume that congestion detection is possible with lightweight probes between the server and the edge router. The hub-based packet redirection will not cause congestion at the hubs or the network because interactive application traffic is small (e.g., in gaming, heavy-duty graphics are loaded to the clients in advance, while the interactive traffic is small) [27].

In fact, in the LEQ architecture, there is a tradeoff regarding the appropriate number of hubs for each client. More hubs would lead to more diversity of equalized-latency paths for one client, and thus provide more reliable paths in the face of transient congestion or link/node failure. However, these additional equalized-latency paths are realized by a small compromise in the delay difference that can be achieved. We study this tradeoff through our dynamic simulation setting in Section V-F.

[3]The increased end-to-end application delay for some clients is a small price to pay for a richer interactive session.

D. Comparison to Alternative Network-Based Solutions

The LEQ architecture is scalable to many clients and applications with only minor modifications to edge routers. We compare LEQ architecture to other possible network-based solutions to implement latency equalization.

Buffering by Edge Routers: One obvious approach of using the network to equalize delays is to buffer packets at the edge routers. This would require large buffers for each interactive application, making the router expensive and power inefficient [28]. Edge routers also need complex packet-scheduling mechanisms that: 1) take into account packet delay requirements, and 2) cooperate with other edge routers to decide how long to buffer these packets. These modifications introduce significant changes to the normal operation of today's routers. Our LEQ architecture can reduce the delay difference (with and without compromising delay) without any modification of the routing infrastructure.

Source Routing: One could use source routing to address the problem of latency equalization. Source routing [29] can be used by the sender (i.e., the client or the client edge router) to choose the path taken by the packet. However, this requires that all clients are aware of the network topology and coordinate with each other to ensure that the delay differences are minimized. This function is harder to implement than our proposed LEQ architecture.

Set Up MPLS Paths: We can set up MPLS paths with equalized latency between each pair of client and server edge routers. This approach is more expensive than our LEQ architecture in that it requires $N_{\mathrm{C}} \times N_{\mathrm{S}}$ MPLS paths to be configured. (N_{C} and N_{S} are the number of client and server edge routers, respectively.) This solution does not scale well for large numbers of client and server edge routers.

E. LEQ in the Presence of Access Network Delay

The latency difference in interactive applications also arises from the disparity in the access network delays. Multiple clients may connect to the same client edge router through different access networks. Access network delay depends on the technology used, and the current load on the last mile link [30], [31]. For different access network types, the average access network delay can be: 180 ms for dial-up, 20 ms for cable, 15 ms for asymmetric digital subscriber line (ADSL), and negligible for fiber optic service (FiOS).[4]

In our LEQ architecture, we account for this disparity of access network types by grouping clients into latency equivalence groups.[5] We provide different hubs for each latency group to achieve latency equalization among all the clients. When a client

[4]We assume servers are connected to the network on dedicated high-speed links and thus do not have access delay.

[5]Latency equivalence groups could be set up for delay variations within an access network type if stateful delay measurements are implemented at the edge router.

connects to a game server, the edge router can determine the incoming access network type of the client. The router then identifies the appropriate latency group the client belongs to and then forwards the application traffic to the corresponding hub. This implies that multiple clients connected to the same edge router but with widely different access delays are assigned to different hubs to achieve latency equalization. The proposed LEQ routing in the backbone network can also be used in conjunction with QoS mechanisms in the access networks to provide equalized latency for interactive application traffic.

F. Hosting Applications in a Content Distribution Network

In today's Internet, many content distribution networks (CDNs) have become the major contributor for interdomain traffic [32]. These CDNs may also host servers for interactive applications. In this scenario, the application traffic from the clients must traverse a transit ISP and a CDN to reach the application server. Achieving LEQ under these two different administrative domains is challenging. There are two possible scenarios. The first scenario is a cooperative environment, where the ISP and the CDN cooperate to provide LEQ service within their respective domains. In this cooperative environment we consider the application of the LEQ architecture over the combined topology of both providers. Therefore, similar to the single administrative domain, the LEQ architecture can significantly reduce delay differences.

The second scenario is the service agnostic peering environment where the CDN and the transit ISP do not have any knowledge of topology and routing in the other domain and do not cooperate in placing hubs. In this case, the CDN treats users coming from the transit ISP with differing delays at a border router as similar to users with different access delays. Our evaluation in Section V shows that we can indeed reduce delay differences significantly with only the application hosting provider supporting the LEQ routing service.

IV. Algorithms for Latency Equalization

The key component of our LEQ architecture is the hub selection algorithm, which focuses on the problem of hub selection and the assignment of hubs to the client edge routers. Hubs are selected with the goal of minimizing the delay difference across all client edge routers. We first formulate the basic hub selection problem without considering access delay and prove that it is NP-hard and inapproximable. Therefore, we propose a greedy heuristic algorithm to solve this basic problem and extend the algorithm to handle access delays. We show that delay differences can be significantly reduced using the selected hub nodes as compared to shortest-path routing.

A. Formulating the Basic Hub Selection Problem

We use an undirected graph $G = (V, E)$ to represent the network. The node set V consists of client edge routers, candidate hubs, and servers. Let $V_{\mathrm{C}} \subseteq V$ denote the set of client edge routers, $V_{\mathrm{S}} \subseteq V$ denotes the set of server nodes, V_{H} denotes the set of routers that can be chosen as hub nodes. We denote $d(u, v)$, $u, v \in V$ as the propagation delay of the underlying network path between routers u and v. In order to balance the load among these $N_{\mathrm{S}} = |V_{\mathrm{S}}|$ servers, we associate each client edge router with its r closest servers (in terms of propagation delay). We denote by S_{c_i} the set of servers that are associated with client edge router c_i. Note that the choice of S_{c_i} is independent of the hub locations.

TABLE I
Notations of Basic Hub Selection Problem

$d(u,v)$	Propagation delay between routers u and v
S_{c_i}	Set of servers associated with client edge router c_i
H_{c_i}	Set of hub nodes assigned to client edge router c_i
N_s	Number of servers in the network
r	Number of servers associated with each client edge router
M	Total number of hubs
m	Number of hubs selected for each client edge router
D_{ik}	Maximum delay bound between client edge router c_i and its associated server s_k
D_{max}	Maximum delay bound of all end-to-end paths
δ	Delay difference

We also define D_{ik} as the maximum delay each client edge router c_i can tolerate on its path to the server in S_k. If we set D_{ik} to be the delay experienced by the default routing path, then we enforce that the latency for each end-to-end path should not get worse. Our algorithm will try to equalize path delay difference while making sure that no path's absolute delay increases. If we set $D_{ik} = D_{\max}$ for all i, k, then we allow path delay to increase. Thus, the D_{ik} parameter allows us to set different policies depending on application needs.

To reduce the deployment and management overhead, we set up at most M hubs in the network. For reliability, we require that each client edge router has at least m hubs chosen from M hubs. Thus, each client edge router has m different paths to the servers. The notations are summarized in Table I.

Given $M, m, r, D_{\max}, D_{ik}$, our goal is to find a set H_{c_i} of m hubs for each client edge router c_i so that we can minimize the delay difference δ. Let $d(c_i, h_j)$ denote the delay from a client edge router c_i to a hub h_j. Similarly let $d(h_j, s_k)$ denote the delay from a hub h_j to a server s_k. We use the notation d_{ijk} for $d(c_i, h_j) + d(h_j, s_k)$. Let $y_j = 1$ denote router h_j is a hub, 0 otherwise. $x_{ij} = 1$ denotes router h_j is a hub for client edge router c_i, 0 otherwise. We present the integer programming formulation for the *hub selection problem* as follows:

Minimize delay difference

$$\delta = \max_{c_i \in V_{\mathrm{C}}, h_j \in H_{c_i}, s_k \in S_{c_i}} (d_{ijk}) - \min_{c_i \in V_{\mathrm{C}}, h_j \in H_{c_i}, s_k \in S_{c_i}} (d_{ijk})$$

such that

$$\sum_{j \in V_{\mathrm{H}}} y_j \leq M$$

$$x_{ij} \leq y_j \qquad \forall c_i \in V_{\mathrm{C}},\ h_j \in V_{\mathrm{H}}$$

$$\sum_{j \in V_{\mathrm{H}}} x_{ij} \geq m \qquad \forall c_i \in V_{\mathrm{C}}$$

$$d_{ijk} x_{ij} \leq D_{ik} \qquad \forall c_i \in V_{\mathrm{C}},\ h_j \in V_{\mathrm{H}},\ s_k \in S_{c_i}$$

$$|d_{ijk} - d_{i'j'k'}|(x_{ij} + x_{i'j'} - 1) \leq \delta, \qquad \forall c_i, c_i' \in V_{\mathrm{C}}, h_j, h_j' \in V_{\mathrm{H}}, s_k, s_k' \in S_{c_i}$$

$$y_j \in \{0, 1\} \qquad \forall j : 1 \leq j \leq |V_{\mathrm{H}}|$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j : 1 \leq j \leq |V_{\mathrm{H}}|,\ 1 \leq i \leq |V_c|$$

The first equation in the formulation is the constraint that the total number of hubs cannot be more than M. The second

equation means that each client edge router can only select its hubs from the hub set V_{H}. The third equation is the constraint that each client must have at least m hubs. The fourth equation bounds the delay between an edge router c_i and its associated server s_k. D_{ik} is defined with different values to address different policy requirements. The fifth constraint specifies that pairwise delay differences between pairs x_{ij} and $x_{i'j'}$ cannot exceed δ. It takes effect only when $x_{ij} = 1$ and $x_{i'j'} = 1$, otherwise the constraint is trivially true. The last two equations indicate that y_j and x_{ij} are binary variables.

B. Complexity of the Basic Hub Selection Problem

We now prove that the basic hub selection problem is NP-hard and inapproximable even for a single server.

Theorem 1: When $m < M$, the basic hub selection problem is NP-hard and is not approximable.

Proof: We reduce the hub selection problem to the well-known set cover problem that is NP-hard.

Set Cover Problem: Consider a ground set $U = \{e_1, e_2, \cdots, e_n\}$, and a collection of n subsets $B_i \subseteq U$ of that ground set. Given an integer M, the set cover problem is to select at most M subsets such that taken together they "cover" all the elements in U. In other words, is there a set $C \subseteq \{B_1, \cdots, B_n\}$ such that $|C| \leq M$ and $\cup_{B_i \in C} B_i = U$?

Hub Selection Is NP-Hard: Given an instance of the set cover problem, we construct an instance of the hub selection problem. We map each element e_i to a client edge router c_i, and map each subset B_j to a candidate hub h_j. If $e_i \in B_j$, we set $d(c_i, h_j) = \epsilon$; otherwise, we set it to $q_{ij}D$, where $D > D_{\max}$ and q_{ij} is a positive integer. In addition, $q_{ij} = q_{i'j'}$ if and only if $i = i'$ and $j = j'$. We set the same delay from each candidate hub to each server ($d(h_j, s_k) = \alpha, \forall j, k$).[6] Let $m = 1$, i.e., each client edge router has to have at least one hub. Because a valid hub selection cannot use edges with delay $D > D_{\max}$, all valid delays from any given client edge router to any given server are equal to $\epsilon + \alpha$. That is, the construction ensures that the maximum delay difference of a valid hub selection is zero. Thus, there is a set cover of size M if and only if we have M hubs with delay difference equal to zero where each client gets assigned at least one hub. Thus, we reduce the set cover problem to the hub selection problem. Therefore, the hub selection problem is NP-hard.

The proof for $m \neq 1$ is similar, as long as m is a small constant, we can construct a reduction from set cover problem where each element must be covered m times.

Hub Selection Is Inapproximable: Suppose we can approximate the problem within a λ factor. Let the maximum delay difference of this algorithm be APX and the optimal delay difference be OPT. Then $\mathrm{APX} \leq \lambda \mathrm{OPT}$ since we cannot pick links with delay $q_{ij}D > D_{\max}$ (since it exceeds maximum delay), and the rest of the paths (from client edge routers through candidate hubs to servers) all have $\epsilon + \alpha$ delay. Thus, if a valid solution exists, OPT must be zero. This means $\mathrm{APX} \leq \lambda \mathrm{OPT} = 0$, so the algorithm gives a valid solution for the set cover problem (i.e., a set cover of size $\leq M$ exists). If there is no valid solution, the maximum delay must be at least $D - \epsilon$. Therefore, if $\mathrm{APX} > D - \epsilon$, then there is no solution to the set cover problem. Thus, an approximate solution of hub selection would yield a solution to the set cover problem. This is a contradiction. ■

[6]This proof holds for both one server or multiple-server cases.

TABLE II
HUB SELECTION PROBLEM SUMMARY

Hub Constraint	Complexity	Algorithm
$m < M$	NP hard Inapproximable	Greedy (Algorithm 1)
$m = M$	P	Optimal (Algorithm 2)

Algorithm 1 Greedy algorithm for basic hub selection

Step 1. Sort all the delays from client edge router c_i to server s_k through hub h_j in increasing order, which is denoted as array A.

Step 2. For each A[t], binary search to find the min delay difference:
for each delay $A[t]$
$left = 0$, $right$=$D_{max} - A[t]$
while($left$ not equal $right$)
$\delta_t = (left + right)/2$
L_t = $greedycover$($A[t]$, δ_t, m, $\{d(u, v)\}$, $D_{i,k}$)
if ($|L_t| > M$) $left = \delta_t$ else $right = \delta_t$.

Step 3. Pick L_t with smallest δ_t. If there are multiple solutions that achieve the minimum δ_t, pick the smallest $A[t]$. If $\delta_t = D_{max}$, then output no solutions found.

Fig. 3. Pseudocode of greedy algorithm for basic hub selection.

C. Greedy Hub Selection Algorithm and a Special Case

We first provide a greedy algorithm for the basic hub selection problem and then show that when $m = M$, there exists a polynomial-time optimal solution (Table II).

1) Greedy Algorithm for $m < M$ (Fig. 3): To solve the hub selection problem, we design a simple greedy heuristic algorithm to pick the M hubs. Our algorithm first sorts in increasing order all the delays from each client edge router through each possible hub to its associated servers (Step 1). This sorted list is denoted by the array A. For example, in Fig. 1, the delays from client 1 through hubs R6, R7, and R8 are 10, 10, and 11. The delays from client 2 through hubs R6, R7, and R8 are 18, 10, and 10. We set the following variables: $A[0] = 10$, $A[1] = 11$, $A[2] = 18$. We select the hubs to optimize for delay difference.

We use a binary search to find a feasible solution with minimum delay difference δ (Step 2). For each possible minimum delay value in $A[t]$, we perform a binary search. The search range for the maximum delay difference is $[0, D_{\max} - A[t]]$. The goal of the binary search is to find a solution such that the maximum delay difference bound δ_t is within the preset range, the minimum absolute delay is no smaller than $A[t]$, and other constraints of our hub selection problem are satisfied (e.g., the per-client server delay bound). In each round of the binary search, given a possible minimal delay $A[t]$ and a maximum delay difference bound δ_t, we use the conventional greedy set cover algorithm [33] ($greedycover$ in Algorithm 1) to pick the M hubs. That is, each time we pick the hub that "covers" the maximum number of uncovered client edge routers until all the

Algorithm 2 Optimal algorithm for $m = M$
Step 1. Let B_1, B_2 be the candidate hub records sorted by their min delay and max delay in increasing order.
Step 2. for each $b_t^1 \in B_1$ in sorted order
$L_t = \{b_t^1.h_{id}\}$, $\delta_t = D_{max}$
for each $b_j^2 \in B_2$ in sorted order
if ($b_j^2.min_d > b_t^1.min_d$ and $|L_t| < M$)
$L_t = L_t \cup \{b^2.h_{id}\}$
$\delta_t = b_j^2.max_d - b_t^1.min_d$
Step 3. Pick L_t such that δ_t is the smallest.

Fig. 4. Pseudocode of the optimal algorithm for $m = M$.

client edge routers are covered. By "covering," we mean all the constraints associated with that client are satisfied. For example, if there is a bound D_{ik} for client i, then this constraint is satisfied by the chosen hub. A client must be covered at least m times. L_t is the set of hubs returned by $greedycover$ when the minimal absolute delay is $A[t]$. Note that a client edge router will not be covered in $greedycover$ by a hub if its inclusion causes the maximum delay difference to exceed the preset bound δ_t. If $m > 1$, each client edge router has to be covered m times. If no feasible solution exists, $greedycover$ will set L_t to some value larger than M, and binary search will output $D_{\max}$ as the δ_t.

Finally, we pick the solution with the minimum δ_t (Step 3). If there are multiple optimal solutions, we pick the one with smallest min delay $A[t]$ among them because applications may also be sensitive to delay.

2) Optimal Algorithm for the Special Case $m = M$ (Fig. 4): Since each client edge router is allowed to use all M hubs, we know the minimal delay and maximum delay for all the paths going through a given hub h_j (paths from any c_i via h_j to any $s_k \in S_{c_i}$). This is in contrast to the general case where these two delays depend on the assignment of hubs to client edge routers. This is the intuition for why we can design an optimal algorithm for the special case, but not for the general case. Let B be a set of all candidate hub records where each record $b \in B$ has three fields: hub ID h_{id}, minimal delay min_{d}, and maximum delay max_{d}. We prune a candidate hub if the delay from edge router c_i to server edge router s_k through it exceeds the D_{ik} bound. Denote the records sorted in increasing order of min_{d} and max_{d} by B_1 and B_2, respectively (Step 1). For each candidate hub record b_t^1, the algorithm computes a candidate solution L_t by adding in M hubs with smallest $b_j^2.max_{\mathrm{d}}$ whose $b_j^2.min_{\mathrm{d}} > b_t^1.min_{\mathrm{d}}$ (Step 2). The algorithm then picks the L_t with minimal δ_t (Step 3).

Theorem 2: Algorithm 2 is optimal when $m = M$.

Proof: Note that L_t is optimal for each possible min_{d}. Therefore L_t with minimal δ_t must be an optimal solution for the problem. ∎

D. Hub Selection With Access Delays

The basic hub placement problem can be easily extended to account for access delays by extending the definition of the nodes V in the general graph $G(V, H)$ to include client groups. For the clients that connect to the same client edge router, we divide them into groups based on their access delay. For example, we can partition clients of an edge router into four groups with access delays in [0, 30), [30, 60), [60, 100), [100, ∞). For each client group g_i, we define access delay $a(g_i)$ as the median delay between clients in the group and their associated edge router. We use the median delay to characterize the client groups in our algorithm. Furthermore, we define the delay between the client groups and the hubs as $d(g_i, h_j)$, which consists of two parts: the access delay and the delay from the edge router to hub h_j. Similarly the delay from h_j to the server s_k denoted by $d(h_j, s_k)$ consists of the delay from the hub h_j to the server edge router and the delay from the server edge router to s_k. We extend the greedy hub placement algorithm in Fig. 3 to consider access delay as shown in Fig. 5.

Algorithm 3 Hub selection algorithm with access delay
Step 1.a For each client group, calculate the median access delay and the delay from the edge router to the hubs.
Step 1.b Sort all the delays no larger than D_{max} from *client group* g_i to server s_k through candidate hub h_j in increasing order, which is denoted as array A
Step 2 and 3: Same as **Step 2. and 3.** in **Algorithm 1**.

Fig. 5. Pseudocode of hub selection algorithm with access delay.

The use of client groups simplifies the management of the LEQ architecture. For a newly arrived client, we only need to determine the client group it belongs to. The client edge router forwards the packets from the new client to the hubs associated with its client group. The access delay for a client may change over time based on the access link load. We can periodically measure access delay changes [34] and assign the client to different delay groups accordingly.

V. Evaluation

We evaluate our LEQ routing architecture using both static and dynamic scenarios on ISP network topologies. In the static case, we only consider propagation delays, and this corresponds to the scenario of a lightly loaded network. We also evaluate the delay difference under different optimization policies both with and without compromising the delay of individual clients, and different network settings such as considering access network delay and multiple administrative domains. In the dynamic case, we evaluate the LEQ routing architecture under transient congestion. In each simulation scenario, we compare the performance of the LEQ routing scheme to that of shortest-path routing (OSPF).

A. Simulation Setup

For our network simulations, we use large ISP network topologies such as AT&T and Telstra. These topologies were obtained from Rocketfuel [35]. For the dynamic case, we consider the Abilene network topology [25]. The key characteristics of these networks are summarized in Table III.

Our evaluation uses several parameters that define the LEQ architecture: the total number of hubs M, the number of hubs selected for each client edge router m, the number of servers in the network N_{S}, and the number of servers allocated for each client edge router r. We evaluate the LEQ routing architecture with and without access delay. We use "acd" to denote the range of

TABLE III
MAIN CHARACTERISTICS OF EXAMPLE NETWORKS

ISP	AT&T	Telstra	Abilene
Number of nodes	391	97	11
Number of links	1280	132	14

access delay. The performance metric is the delay difference δ, which is the maximum difference in delay among all the selected paths.

We use all the edge nodes in the backbone topology as client edge routers and randomly choose N_S edge nodes as the location of servers. Each client would communicate with r servers that are nearest to it in terms of propagation delay. We then run the LEQ routing and shortest-path routing algorithms to compute the paths between these clients and servers. Note that, in the static case, the LEQ path computation is based on the propagation delay in the network. We compute the propagation delay of these networks based on the geographical distances between any two nodes. To eliminate the bias introduced by server location, when $N_S = 1$, we test all the possible locations of the servers; when $N_S > 1$, we run each simulation 1000 times with randomized the server locations.

B. LEQ Without Compromising End-to-End Delay

We first explore the potential of the LEQ routing architecture to discover latency equalized paths, under the optimization constraint that the *end-to-end delays of individual clients are not compromised*.

In a typical service provider network, OSPF weights are optimized for traffic engineering [18], and thus OSPF paths may not always correspond to lowest delay paths [19]. Our evaluation uses OSPF weights from Rocketfuel [35]. The delay and delay differences obtained using OSPF routing are compared to that obtained using LEQ routing. To not compromise on end-to-end delay, in our LEQ path calculation algorithm, we add the optimization constraint that we do not increase the delay of any client. Fig. 6(a) shows that LEQ reduces the average delay difference by 60%. This latency equalization is achieved without sacrificing the end-to-end delay obtained using OSPF. Fig. 6(b) shows that the average end-to-end delay of LEQ and OSPF are similar, but LEQ reduces the maximum end-to-end delay compared to OSPF since some end-to-end path delays have been improved by the LEQ scheme. This is consistent with the work in [22], where the authors show that end-to-end delay can be reduced by packet indirection.

C. LEQ With Compromising End-to-End Delay

In the scenarios where the interactive application permits increasing the delay of some clients application traffic in order to reduce the overall delay difference, a further reduction in the delay difference can be achieved. Under this optimization policy, we do not have any delay constraint for individual client's application traffic in our algorithm. To highlight the main features of the LEQ architecture, we first consider the provider network without network access delays.

1) LEQ Routing Architecture Reduces Delay Difference Significantly Compared to Shortest-Path Routing and Only Requires a Few Hubs: Fig. 7 shows the average delay difference

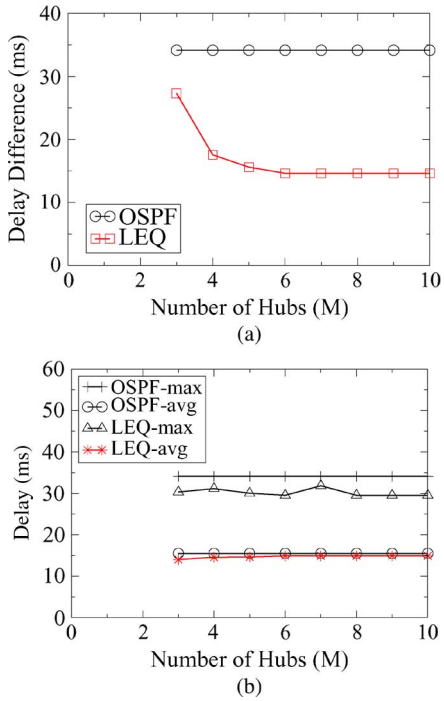


Fig. 6. Comparison of OSPF and LEQ without compromising delay (AT&T network, $N_S = 1$, $m = 3$). (a) Delay difference. (b) Delay.

between all the client paths to any one server for both LEQ routing and OSPF. In all the networks we tested, we find that LEQ routing with a single hub per client $(m = 1)$ reduces delay difference to 5 ms, which is an 85% reduction for AT&T, 85% for Telstra, and 90% for Abilene over the default OSPF routing. Even with just one hub in the entire network $(M = 1)$, LEQ routing has on average 40% reduction in delay difference. Also, the best performance for LEQ routing is achieved when the number of hubs per client (m) is set to 1. As the number of hubs per client increases, we find that, due to the increased path diversity, the average delay difference increases. However, even with three hubs per client, the performance of LEQ routing is significantly better than OSPF.

From Fig. 7, we note that increasing the total number of hubs in the network to more than five does not provide any significant improvement in the delay difference measurements. This result holds with varied topologies (AT&T with 391 nodes, and Abilene with 11 nodes). This reduction in the delay difference significantly improves application interactivity as seen later in Fig. 11.

2) Improvement in Overall Delay Difference Can be Achieved With Some Compromise in End-to-End Delay: In Fig. 8, we first compare the maximum, average, and median delay of the selected paths of LEQ routing with those of OSPF whose weights are optimized for minimizing end-to-end delay. We can see that the maximum delay of LEQ is similar to that obtained with OSPF. However, the average delay of LEQ is larger than that of OSPF. This is because to achieve better LEQ (as compared to the case in Fig. 6), the end-to-end delay of some paths has to be compromised.

3) With Multiple Servers, LEQ Architecture Also Reduces the Delay Difference: In some applications such as P2P gaming

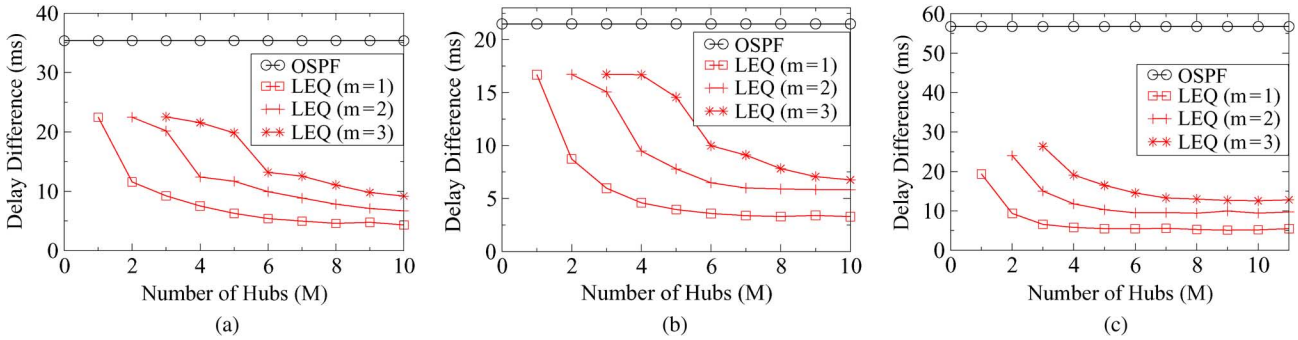


Fig. 7. Delay difference evaluation of OSPF and LEQ routing with compromising delay. ($N_S = 1$). (a) AT&T network. (b) Telstra network. (c) Abilene network.

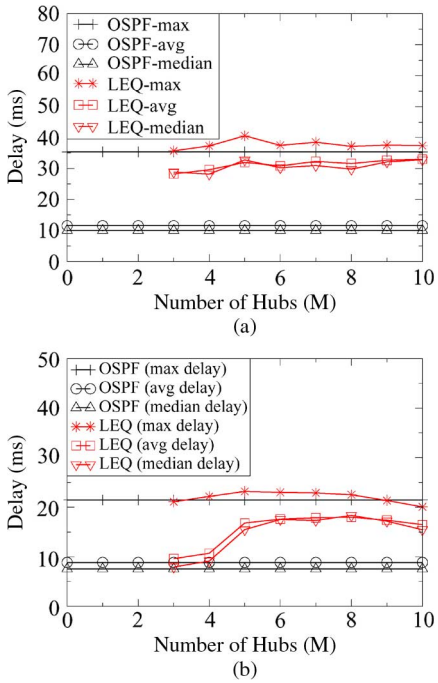


Fig. 8. Delay evaluation of OSPF and LEQ routing with compromising delay. ($m = 3$, $N_S = 1$). (a) AT&T network. (b) Telstra network.

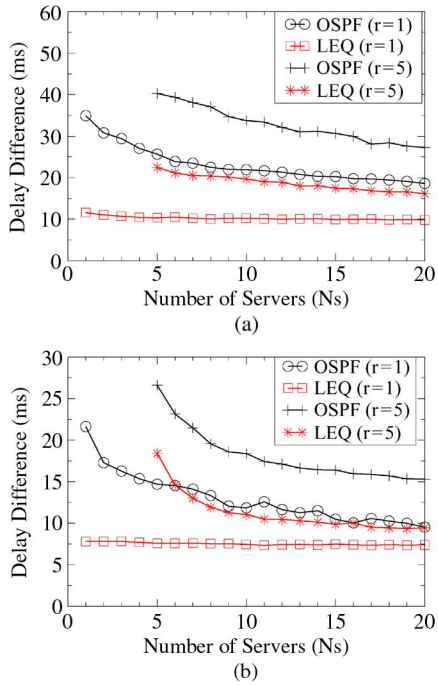


Fig. 9. Influence of number of servers (N_S) and servers per client r in LEQ routing ($M = 5, m = 2$). (a) AT&T network. (b) Telstra network.

and distributed live music performances, there are multiple servers at different places, and each client is associated with several servers. Fig. 9 shows the influence of the number of servers (N_S) and servers per client r on delay difference δ. With OSPF, keeping the number of servers per client (r) fixed, increasing the total number of servers reduces the average delay difference. This is because with more servers, the clients can choose nearer servers. Thus, their shortest path delay to the servers decreases, and correspondingly the delay difference also decreases. As compared to OSPF, with LEQ routing, increasing the total number of servers does not have as significant an impact. Even with 20 servers, LEQ routing still performs better than OSPF (e.g., 50% delay difference reduction when $r = 1$).

D. LEQ With Access Delay

The grouping of access delay is discussed in Section III. For simplicity, we consider two ranges of access network delays (acd): [0 ms, 50 ms] and [0 ms, 100 ms] (we omit ms in the following text). These large delay ranges were chosen since even within a single access network technology, the delays experienced by the clients are highly variable and dependent on network load [31]. We assume 20 clients on each edge router with access delays that are randomly chosen within the range.

1) Improvement in Delay Difference Depends on the Range of Variation in the Access Network Delays: Fig. 10(a) shows that in AT&T network, with access delay ranging from 0 to 50 ms, the delay difference can be reduced by 45%. However, with access delays ranging from 0 to 100 ms [Fig. 10(b)], the delay difference is reduced only by 35%. Similarly in the Telstra network, the delay difference is reduced by 50% for $acd : [0 - 50]$ and 25% for $acd : [0 - 100]$. These results show that when the access network delay difference is very large, LEQ routing performance will benefit from improvements in access network technologies that reduce access delay variations for interactive applications by giving them higher priority.

2) LEQ Improves User Experience—An Important Criteria for Interactive Applications: The user experience of interactive applications relates to both delay difference among the in-

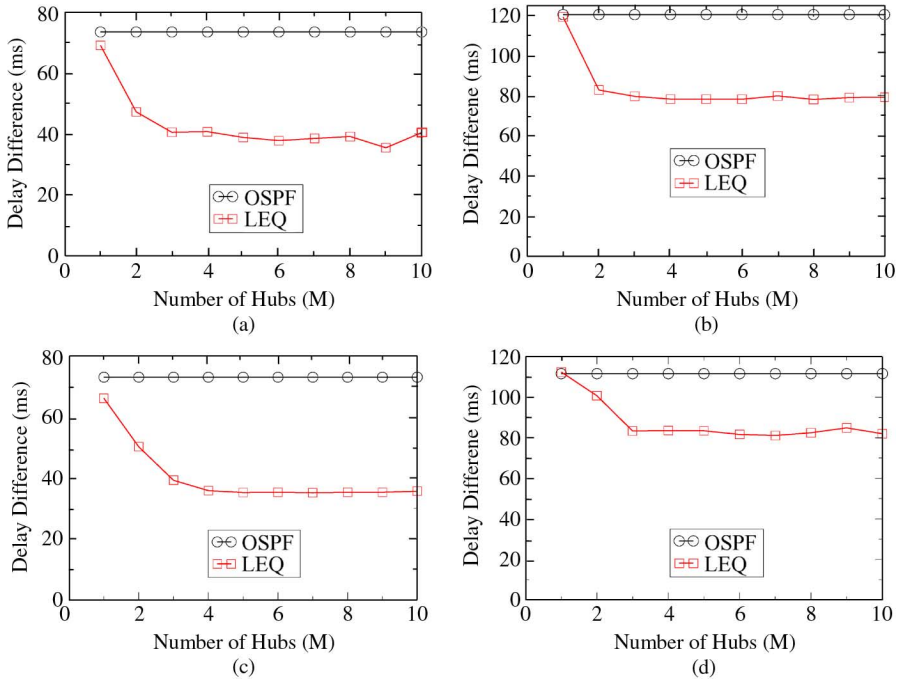


Fig. 10. Delay difference with access delay ($m = 2, N_S = 1, r = 1$). (a) AT&T network, $acd : [0 - 50]$. (b) AT&T network, $acd : [0 - 100]$. (c) Telstra network, $acd : [0 - 50]$. (d) Telstra network, $acd : [0 - 100]$.

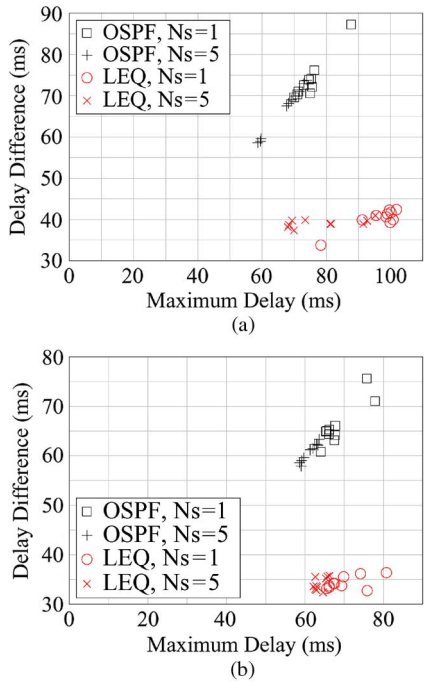


Fig. 11. User experience comparison of max delay and delay difference ($acd = [0, 50], M = 5, m = 2, r = 1$). (a) AT&T network. (b) Telstra network.

teracting parties and the individual end-to-end delay. Fig. 11 shows that in both Telstra and AT&T networks, LEQ routing has smaller delay difference and thus better interactivity while providing the same maximum end-to-end delay as with OSPF routing. The increased delay difference in OSPF reduces the interactive experience. In the case when there are multiple servers ($N_S = 5$), and each client is associated with just one server, the maximum end-to-end delay for both OSPF and hub routing is reduced. However, as compared to OSPF, LEQ routing still has significant improvement in user interactive experience due to reduced delay difference among clients.

E. LEQ With a CDN and a Transit ISP

We consider the case of supporting LEQ on a CDN network (AT&T[7]) over a transit ISP (Sprint), where the server resides on the CDN and the clients exist on both the transit ISP and the CDN with different access delay ranges. We investigate the following scenarios for the two autonomous systems (ASs): 1) ASs cooperate on routing (labeled as "joint OSPF," "joint LEQ"); and 2) ASs make routing decisions independently (labeled as "OSPF-LEQ," "LEQ-LEQ"). We also investigate different server location scenarios: 1) the servers are located close to the peering nodes of the two networks (labeled as fixed server); and 2) the servers are randomly located (labeled as "rand server"). For each scenario, we randomly choose client and server locations and plot one point of delay and delay difference for each choice. Our observations are as follows.

1) If the CDN and the Transit ISP Cooperate, LEQ Architecture Can Improve User Experience: Fig. 12 plots the delay difference and maximum delay for several scenarios. In the joint OSPF and joint LEQ scenarios, we assume that the routing scheme has complete knowledge of both network topologies. Joint LEQ routing always performs significantly better than joint OSPF.

2) If the CDN and the Transit ISP Do Not Cooperate, Applying LEQ Routing in the CDN Is Enough to Improve User Experience Significantly: As shown in Fig. 12, even without the

[7]Since we do not have a CDN network topology, we use AT&T's topology as the CDN network.

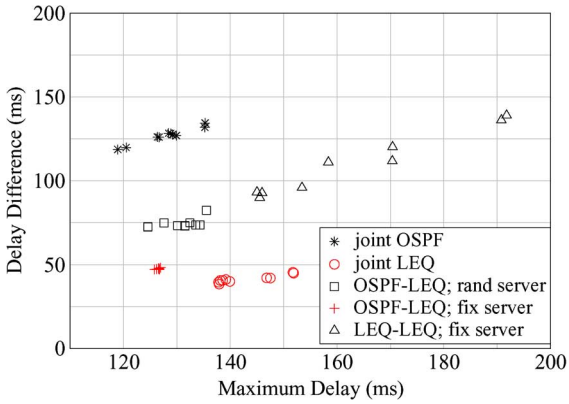


Fig. 12. LEQ in peering ASs with access delay (acd: [0–50]%, $N_S = 1$) (OSPF-LEQ means applying OSPF in Sprint, LEQ in AT&T; LEQ-LEQ means applying LEQ separately in both Sprint and OSPF.)

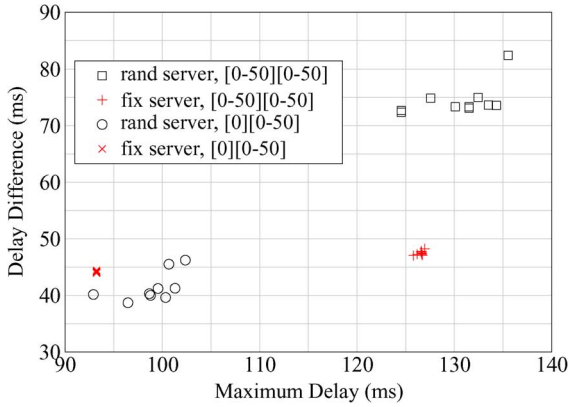


Fig. 13. Effect of access delay in peering ASs. (The first range is Sprint acd, which is either close to 0 or [0–50]; the second range is AT&T acd range, which is always [0–50]. We apply OSPF in Sprint, LEQ in AT&T.)

knowledge of the peering Sprint network topology, the AT&T network can use LEQ in its own network (OSPF-LEQ, which also provides better performance than using the standard OSPF routing on the joint topology (joint OSPF).

When the server is placed near the peering nodes (fixed server case), applying LEQ only in AT&T achieves better user experience than applying LEQ independently in both Sprint and AT&T. This is because the two networks do not cooperate but independently optimize for LEQ. By applying LEQ routing in Sprint, within the Sprint network the clients may experience equal delay with each other, but when they come in to the AT&T network, their delay difference with clients in AT&T is worse than if they had taken the shortest path through the Sprint network.

3) Placing the Server Close to Peering Nodes (Labeled "Fixed Server" in Figs. 12 and 13) Improves User Experience, When Two ASs Have Equal Access Delay: From Fig. 12, we see that when the server is placed close to the peering node,[8] and AT&T applies LEQ routing (OSPF-LEQ, fixed server case), we can achieve similar delay difference to the case when we have full knowledge of both topologies (joint LEQ case). Furthermore, in Fig. 13, when the access delay range of Sprint and AT&T are both [0–50], we note that the application performance is improved by placing servers close to the peering node locations.

The proximity of the server to peering nodes helps only when access delays from both networks are in the same range. Fig. 13 shows that when the access delay of players in Sprint is close to 0,[9] the location of the servers does not affect performance. This is because although players from the Sprint edge router have less access delay, they take longer paths to reach the AT&T network where the server is located. Therefore, they can be viewed as players in the AT&T network with some access delay. When the access delays across the networks have large variability, LEQ routing does not benefit from placing servers near the peering nodes.

[8]Peering nodes are the nodes that connect the two ASs.

[9]Some application servers may only allow players with low access delay in Sprint to join the application in order to improve user experience.

F. Dynamic Analysis

In a typical service provider network, links are usually provisioned at below 50% average utilization. However, it has been observed that in the presence of traffic bursts, it is possible that on the timescale of minutes, the average utilization could be close to 90%–95% of the link capacity [36]. Under these conditions, the queue size builds up at the links and contributes to the overall delay between the clients and the server. Therefore, we investigate the performance of LEQ architecture under these dynamic traffic conditions.

We implemented *LEQ routing* as a new routing module in *ns-2* for packet-level simulations. We generated two classes of packets: packets of background traffic and probing packets. The background traffic denotes traffic of all the other applications in the network. Since the interactive application traffic such as gaming traffic is much smaller than background traffic and will not influence the network conditions, we do not simulate them explicitly in our experiment. Instead, we use small probing packets that go through LEQ routing paths, and we measure the actual latency experienced by these packets. To force some of the probing packets to go through LEQ routing paths, each client edge router marks the probing packets with the address of the destination node and sends them to a hub. The hub is selected in a round-robin schedule among the m hubs allocated to the client edge router through the hub selection algorithm. Upon receiving the probing packets, the hub looks up the destination server from the packet and redirects it to the server. For comparative purposes, we also send probing packets through shortest-path routes computed using Dijkstra's algorithm parametrized by the propagation delay metric.

For the dynamic analysis, we use the Abilene network topology (Fig. 2). We use a single server that is located at Washington, DC. All the 11 edge nodes have clients. The bandwidth capacity of each link in Abilene is fixed at 10 Gb/s. For background traffic, we generated real traffic matrices based on the Abilene Netflow data [25]. The size of the probing packet is 48 B, which is similar to the size of general UDP packets in interactive applications [27]. Probing packets are sent at a fixed interval of 0.01 s.

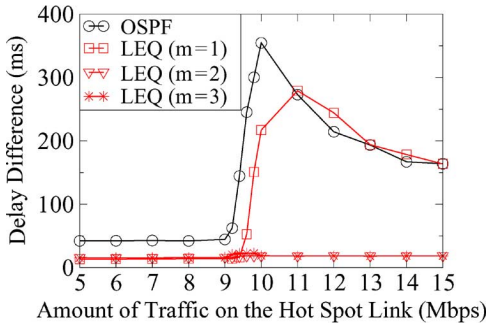


Fig. 14. Abilene network: single-bottleneck link.

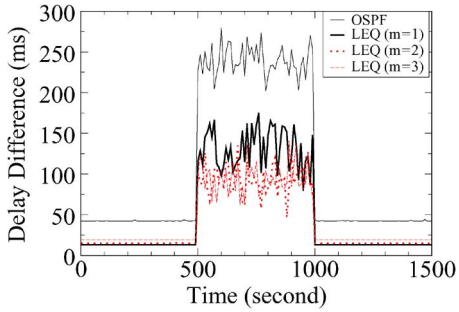


Fig. 15. Transient congestion in LEQ routing.

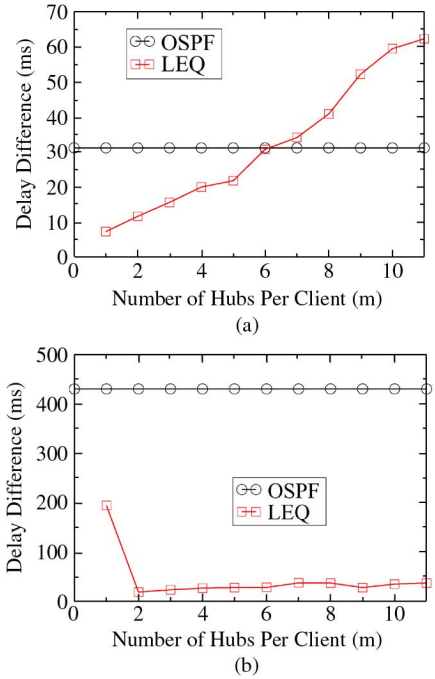


Fig. 16. Influence of hubs per client (m) on LEQ. (a) Static analysis of propagation delay. (b) Packet-level simulation of delay during congestion.

1) LEQ Achieves Reliability in the Single-Bottleneck-Link Scenario by Providing Multiple Hubs for Each Client: We first experiment with a single-bottleneck link. We start with a uniform traffic matrix (0.1 Gb/s), and then increase the traffic load on the hotspot link between Denver, CO, and Kansas City, MO. Fig. 14 shows the real-time evolution of the delay difference of hub routing and shortest-path routing. When the amount of traffic on this link approaches the capacity of the link, both OSPF and LEQ routing with $m = 1$ experience significant queuing delay. Using LEQ routing with $m = 2$ or 3, we maintain the delay difference due to path diversity. When the amount of traffic continues to increase, packets from different clients are also influenced by this heavy traffic, and thus the delay difference among all the clients decreases in both cases: OSPF and LEQ routing with $m = 1$. From this study, we show that with two or three hubs per client, LEQ routing can get around transient congestion in the network and reduce the corresponding queuing delay.

2) LEQ Also Achieves Reliability Under Transient Network Congestion: To evaluate the performance of hub routing under more realistic scenarios where the traffic matrix is not uniform and transient congestion may happen at any place, we run the experiment with 150 realistic Abilene traffic matrices, each applied for 10 s. During the time interval of 500–1000 s, we insert traffic burst on several selected links by increasing the utilization to 90% of the link capacity. The overloaded links were chosen based on a snapshot of the Abilene network operation center [37]. As shown in Fig. 15, when transient congestion happens, LEQ routing with $m = 2, 3$ has alternate routes to get around the congested link. Thus, the impact of transient congestion is less prominent on LEQ routing than it is on shortest-path routing.

3) Considering the Tradeoff of Robustness and Performance, We Need to Assign Each Client Edge Router Two or Three Hubs From the Hub Set: Fig. 16 compares the performance from both the static and dynamic analysis. In the static analysis in Fig. 16(a), with one hub per client, we can achieve the minimal delay difference in terms of propagation delay. However, in Fig. 16(b), during transient congestion, queuing delay becomes more critical than propagation delay. Thus, adding one more hub (changing m from 1 to 2) provides more path diversity while reducing the average delay difference. Robustness to traffic variability and transient network congestion are important requirements for any real-time interactive application. In LEQ architecture, this robustness can be achieved with $m = 2$ or 3 without severe impact on delay difference. This is consistent with the work in [38], where they prove that load balancing becomes much easier by just having two routing paths and the flexibility to split traffic between them.

VI. Related Work

Network support for gaming and other interactive services is a relatively new topic since the scalability and commercial significance of these services has only recently become important. In [1], the authors provide the motivation for network support for gaming and design a game booster box—a network-based game platform that combines low-level network awareness and game-specific logic. The goal of the booster box is to offload network functions from the game server, specifically network monitoring. Shaikh *et al.* [39] and Saha *et al.* [40] proposed an online game hosting platform, a middleware solution based on existing grid components. The platform performs functions such as addition and removal of servers and administrative tasks such as new game deployment, directory management, player redirection to server, and game content distribution. Our preliminary

study in [41] presented the basic idea of LEQ as a service that could run on programmable routers. In this paper, we consider the deployment of LEQ architecture on today's Internet, provide hub selection algorithms for different network settings, and present a complete simulation and evaluation of LEQ in various scenarios (e.g., with access network delay, multiple administrative domains).

The authors in [42] provide a method to predict latency between machine pairs and use this information to help players *preselect* the other players that have similar delay. In contrast, the LEQ architecture aims at improving the game experience for players already participating in the game session by routing the game packets through equalized latency paths during the game session. Our work can leverage the techniques in [42] to provide better prediction of the delay metric used in our hub selection algorithm.

From the network performance optimization perspective, previous work focused on reducing delay in the overlay network (e.g., RON [22]) or reducing bandwidth costs with bounded delay (e.g., VPN tree routing [43]). There are no theoretical results thus far aimed at optimizing latency difference in the network. In this paper, we formulate the hub selection problem that is a critical component for optimizing latency differences and show its NP-hardness. We also design and implement algorithms for hub selection that achieves latency-equalized paths in the network. Cha *et al.* [44] proposed a strategy to place relay nodes in the intradomain network. However, their selection algorithm is aimed at reducing cost, not delay difference.

VII. Conclusion

The LEQ routing architecture and algorithms presented in this paper clearly provide a pathway for networks to support scalable and robust multiparty interactive applications. Based on the evaluation of our LEQ architecture, we conclude that, with only minor enhancements to the edge routers, provider networks can easily support and enhance the quality of multiparty interactive applications. We show that the LEQ scheme can support different optimization policies that can achieve overall application performance in terms of latency equalization both with and without compromising end-to-end application latencies.

References

[1] D. Bauer, S. Rooney, and P. Scotton, "Network infrastructure for massively distributed games," in *Proc. NetGames*, 2002, pp. 36–43.

[2] A. Kapur, G. Wang, P. Davidson, and P. R. Cook, "Interactive network media: A dream worth dreaming?," *Organized Sound*, vol. 10, no. 3, pp. 209–219, 2005.

[3] A. R. Greenwald, J. O. Kephart, and G. Tesauro, "Strategic pricebot dynamics," in *Proc. ACM Conf. Electron. Commerce*, 1999, pp. 58–67.

[4] "Cisco telepresence solutions," Cisco, San Jose, CA [Online]. Available: http://www.cisco.com/en/US/netsol/ns669/networking_solutions_solution_segment_home.html

[5] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Proc. NOSSDAV*, New York, 2002, pp. 23–29.

[6] S. Zander and G. Armitage, "Empirically measuring the QoS sensitivity of interactive online game players," in *Proc. ATNAC*, Dec. 2004, pp. 511–518.

[7] J. Brun, F. Safaei, and P. Boustead, "Managing latency and fairness in networked games," *Commun. ACM*, vol. 49, no. 11, pp. 46–51, Nov. 2006.

[8] M. Dick, O. Wellnitz, and L. Wolf, "Analysis of factors affecting players' performance and analysis of factors affecting players' performance and perception in multiplayer games," in *Proc. NetGames*, 2005, pp. 1–7.

[9] C. Diot and L. Gautier, "A distributed architecture for multiplayer interactive applications on the Internet," *IEEE Netw.*, vol. 13, no. 4, pp. 6–15, Jul.–Aug. 1999.

[10] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin, "An efficient synchronization mechanism for mirrored game architectures," in *Proc. NetGames*, 2002, pp. 67–73.

[11] S. Zander, I. Leeder, and G. J. Armitage, "Achieving fairness in multiplayer network games through automated latency balancing," in *Proc. Adv. Comput. Entertain. Technol.*, 2005, pp. 117–124.

[12] J. Nichols and M. Claypool, "The effects of latency on online Madden NFL football," in *Proc. NOSSDAV*, 2004, pp. 146–151.

[13] A. Abdelkhalek and A. Bilas, "Parallelization and performance of interactive multiplayer game servers," in *Proc. IPDPS*, 2004.

[14] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool, "The effects of loss and latency on user performance in unreal tournament 2003," in *Proc. NetGames*, 2004, pp. 144–151.

[15] P. Quax, P. Monsieurs, W. Lamotte, D. D. Vleeschauwer, and N. Degrande, "Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game," in *Proc. NetGames*, 2004, pp. 152–156.

[16] M. Oliveira and T. Henderson, "What online gamers really think of the Internet?," in *Proc. NetGames*, 2003, pp. 185–193.

[17] "AT&T renews hosting agreement with Blizzard Entertainment Inc. for online games," Press release, 2009 [Online]. Available: http://www.reuters.com/article/2009/03/03/idUS159848+03-Mar-2009+PRN20090303

[18] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, 2000, vol. 2, pp. 519–528.

[19] N. Spring, R. Mahajan, and T. Anderson, "The causes of path inflation," in *Proc. SIGCOMM*, 2003, pp. 113–124.

[20] "SureRoute," Akamai, Cambridge, MA, 2003 [Online]. Available: http://www.akamai.com/dl/feature_sheets/fs_edgesuite_sureroute.pdf

[21] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: Informed internet routing and transport," *IEEE Micro*, vol. 19, no. 1, pp. 50–59, Jan.–Feb. 1999.

[22] D. G. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. Symp. Oper. Syst. Principles*, Banff, AB, Canada, 2001, vol. 35, no. 5, pp. 131–145.

[23] L. Gautier and C. Diot, "Design and evaluation of MiMaze, a multi-player game on the internet," in *Proc. IEEE Int. Conf. Multimedia Comput. Syst.*, 1998, pp. 233–236.

[24] L. Pantel and L. C. Wolf, "On the suitability of dead reckoning schemes for games," in *Proc. NetGames*, 2002, pp. 79–84.

[25] "Internet2 network," Internet2, Ann Arbor, MI [Online]. Available: http://abilene.internet2.edu/

[26] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," in *Proc. ACM SIGCOMM*, Pisa, Italy, Sep. 2006, pp. 3–14.

[27] J. Farber, "Network game traffic modelling," in *Proc. NetGames*, 2002, pp. 53–57.

[28] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," in *Proc. ACM SIGCOMM*, 2002, pp. 251–264.

[29] J. Postel, "Internet protocol: DARPA Internet program protocol specification," RFC 791, 1981.

[30] T. Jehaes, D. D. Vleeschauwer, B. V. Doorselaer, E. Deckers, W. Naudts, K. Spruyt, and R. Smets, "Access network delay in networked games," in *Proc. NetGames*, 2003, pp. 63–71.

[31] N. Barakat and T. E. Darcie, "Delay characterization of cable access networks," *IEEE Commun. Lett.*, vol. 11, no. 4, pp. 357–359, Apr. 2007.

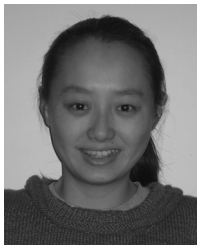
[32] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," in *Proc. ACM SIGCOMM*, 2010, pp. 75–86.

[33] V. Chvatal, "A greedy heuristic for the set-covering problem," *Math. Oper. Res.*, vol. 4, no. 3, pp. 233–235, 1979.

[34] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proc. ACM SIGCOMM IMC*, 2007, pp. 43–56.

[35] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.

[36] R. Prasad, C. Dovrolis, and M. Thottan, "Router buffer sizing revisited: The role of the output/input capacity ratio," in *Proc. ACM CoNEXT*, 2007, Article no. 15.
[37] "Indiana University Global Research Network Operations Center weathermaps," Global Research Network Operations Center, Indiana University, Indianapolis, IN [Online]. Available: http://weathermap.grnoc.iu.edu/
[38] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, Oct. 2001.
[39] A. Shaikh, S. Sahu, M. Rosu, M. Shea, and D. Saha, "Implementation of a service platform for online games," in *Proc. NetGames*, 2004, pp. 106–110.
[40] D. Saha, S. Sahu, and A. Shaikh, "A service platform for on-line games," in *Proc. NetGames*, 2003, pp. 180–184.
[41] M. Yu, M. Thottan, and L. Li, "Latency equalization: A programmable routing service primitive," in *Proc. ACM PRESTO*, 2008, pp. 39–44.
[42] S. Agarwal and J. R. Lorch, "Matchmaking for online games and other latency-sensitive P2P systems," in *Proc. ACM SIGCOMM*, 2009, pp. 315–326.
[43] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: A network design problem for multicommodity flow," in *Proc. ACM STOC*, 2001, pp. 389–398.
[44] M. Cha, S. Moon, C. D. Park, and A. Shaikh, "Placing relay nodes for intra-domain path diversity," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.



Minlan Yu (S'10) received the B.A. degree in computer science and mathematics from Peking University, Beijing, China, in 2006, and the M.A. degree in computer science from Princeton University, Princeton, NJ, in 2008, and is currently pursuing the Ph.D. degree in computer science at Princeton University.

She has interned at Microsoft, Redmond, WA; AT&T Laboratories Research, Florham Park, NJ; and Bell Laboratories, Murray Hill, NJ. Her research interest is in network virtualization, enterprise, and data center networks.

Marina Thottan (M'00) received the Ph.D. degree in electrical and computer engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2000.

She is Director of the Mission-Critical Communications and Networking Group, Bell Laboratories, Murray Hill, NJ. Most recently, she has been leading work on smart grid communication networks. She has published over 40 papers in scientific journals, book chapters, and refereed conferences.

Dr. Thottan is a member of the Association for Computing Machinery (ACM).

Li (Erran) Li (M'99–SM'10) received the B.E. degree in automatic control from Beijing Polytechnic University, Beijing, China, in 1993, the M.E. degree in pattern recognition from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1996, and the Ph.D. degree in computer science from Cornell University, Ithaca, NY, in 2001.

Since graduation, he has been with the Networking Research Center, Bell Laboratories, Murray Hill, NJ. He has published over 40 papers.

Dr. Li is an Editor of *Wireless Networks* and the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. He was a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATION Special Issue on Non-Cooperative Behavior in Networking.