IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-3, Issue-5 E-ISSN: 2347-2693

Latency Equalization as a New Network Service Primitive in an Interactive Network Application

RAJU V

Department of Computer Science, Arignar Anna Govt. Arts College, Villupuram – 605 602

www.ijcseonline.org

Received: Apr /25/2015 Revised: May/07/2015 Accepted: May/19/2015 Published: May/30/ 2015

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.

Keywords : Latency equalization, Interactive network, Delay difference in networks, NP-Hardness

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 and online music (orchestra) . The commercial sector has defined interactive services such as bidding in e-commerce and telepresence. 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.

LATENCY EQUALIZATION (LEQ)

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.

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.

Example - In online gaming, the delay difference experienced by gamers significantly impacts game quality 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 , 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.

PREVIOUS WORK

In online gaming, the delay difference experienced by gamers significantly impacts game quality.

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 , 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.

PREVIOUS TECHNIQUES USED

Dead reckoning, bucket synchronization mechanism, time warp synchronization scheme were used in the previous works.

ISSUES IN THE PREVIOUS WORK

DEAD RECKONING

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.

BUCKET SYNCHRONIZATION MECHANISM

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.

TIME WARP SYNCHRONIZATION SCHEME

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.

PROBLEM DEFINITION

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]. Clientside 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.

REQUIREMENT SPECIFICATION

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

1.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. 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. 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.

2.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 . 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.

3.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.

4. 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 MPLS paths to be configured. (and 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.

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 lastmile 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

3The increased end-to-end application delay for some clients is a small price to pay for a richer interactive session. 4We assume servers are connected to the network on dedicated high-speed links and thus do not have access delay. 5Latency equivalence groups could be set up for delay variations within an access network type if stateful delay measurements are implemented at the edge router.

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.

LATENCY EQUALIZATION ARCHITECTURE

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).

ILLUSTRATION OF BASIC LEQ HUB ROUTING

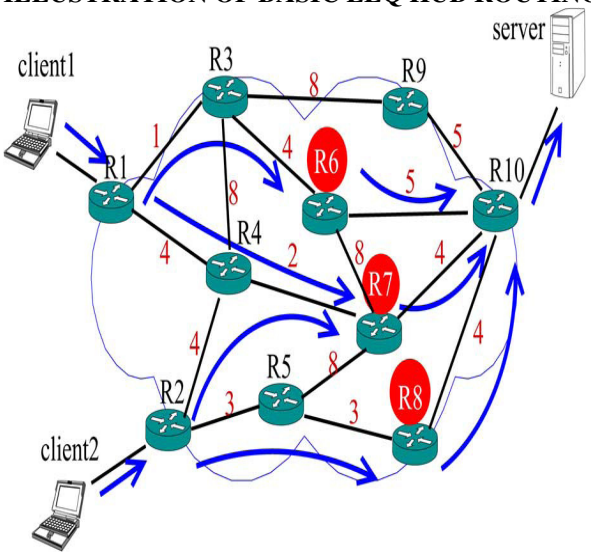
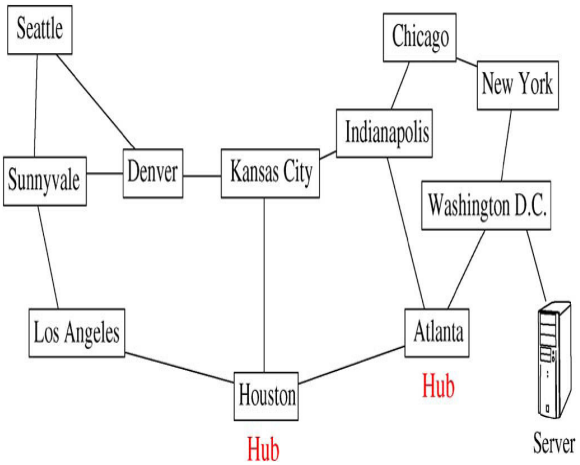


Fig. 1, and are both assigned two hubs. To illustrate the advantage of our LEQ routing concept on real networks, we conducted experiments on the Abilene [25] network in VINI test bed [26] as shown in Fig. 2. We set up a server at the Washington DC node and measured the delay 2In contrast, since OSPF 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 (14 ms) and (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.

LEQ ROUTING ON ABILENE NETWORK.



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.

Formulating the Basic Hub Selection Problem
Complexity of the Basic Hub Selection Problem
Greedy Hub Selection Algorithm and a Special Case
Hub Selection With Access Delays

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 $right$)
$\delta_t = (left + right)/2$
$L_t = greedycover(A[t], \delta_t, m, \{d(u,v)\}, D_{i,k})$
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.

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.

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**.

DISCUSSION AND ANALYSIS

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 andmultiple 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).

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]. 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.

CONCLUSION AND FURTHER WORK

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].

[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.

SAMPLE CODE

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Client
{
   public partial class CLIENT : Form
   {
      public CLIENT()
      {
         InitializeComponent();
      }

      private void CLIENT_Load(object sender, EventArgs
e)
      {

      }
  private void button1_Click(object sender, EventArgs e)
      {
         //TimeSpan t = TimeSpan.FromSeconds( secs );

         if (lsthub.SelectedItem == null)
            MessageBox.Show("Select Hub");
         else
         {
            lblmes1.Text = "Client Node String Time:" +
System.DateTime.Now.ToString("h:m:s:ms");
            lblmes2.Text = "Message Send Successfully";
            DateTime clienttime = System.DateTime.Now;
            string node = "";
     for (int i = 0; i < lsthub.SelectedItems.Count; i++)
            {
              if (lsthub.Items[i] == "191.128.2.1")
               {
                  System.Threading.Thread.Sleep(500);
                  node = "1";

((Hub1)Application.OpenForms["Hub1"]).lab1.Text      =
"Last Update receieved from " + lblip.Text;
  ((Hub1)Application.OpenForms["Hub1"]).lab2.Text      =
"Client Node #:" + node + txtmessage.Text;
 ((Hub1)Application.OpenForms["Hub1"]).lab3.Text      =
"Message Send Successfully";
                  //lblmes1.Text = "Client Node String
Time:" + System.DateTime.Now.ToString();
                  //lblmes2.Text      =      "Message      Send
Successfully";
               }
    if (lsthub.Items[i] == "191.128.2.2")
               {
                  System.Threading.Thread.Sleep(500);
                  node = node + ":2";
  ((Hub2)Application.OpenForms["Hub2"]).lab1.Text      =
"Last Update receieved from " + lblip.Text;

((Hub2)Application.OpenForms["Hub2"]).lab2.Text      =
"Client Node #:" + node + txtmessage.Text;

((Hub2)Application.OpenForms["Hub2"]).lab3.Text      =
"Message Send Successfully";
                  //lblmes1.Text = "Client Node String
Time:" + System.DateTime.Now.ToString();
                  //lblmes2.Text      =      "Message      Send
Successfully";
               }
               if (lsthub.Items[i] == "191.128.2.3")
               {
                  System.Threading.Thread.Sleep(500);
                  node = node + ":3";

((Hub3)Application.OpenForms["Hub3"]).lab1.Text      =
"Last Update receieved from " + lblip.Text;

((Hub3)Application.OpenForms["Hub3"]).lab2.Text      =
"Client Node#:" + node + txtmessage.Text;

((Hub3)Application.OpenForms["Hub3"]).lab3.Text      =
"Message Send Successfully";
      }
               if (lsthub.Items[i] == "191.128.2.4")
               {
                  System.Threading.Thread.Sleep(500);
                  node = node + ":4";
  ((Hub4)Application.OpenForms["Hub4"]).lab1.Text      =
"Last Update receieved from " + lblip.Text;
 ((Hub4)Application.OpenForms["Hub4"]).lab2.Text      =
"Client Node #:" + node + txtmessage.Text;
 ((Hub4)Application.OpenForms["Hub4"]).lab3.Text      =
"Message Send Successfully";
                  //lblmes1.Text = "Client Node String
Time:" + System.DateTime.Now.ToString();
                  //lblmes2.Text      =      "Message      Send
Successfully";
               }
            }
            System.Threading.Thread.Sleep(500);
            node = node + ":4";

((Server)Application.OpenForms["Server"]).lab1.Text      =
"Last Update receieved from " + lblip.Text;
```

```
((Server)Application.OpenForms["Server"]).lab2.Text = "Client Node #:" + node + txtmessage.Text;

((Server)Application.OpenForms["Server"]).lab3.Text = "Client to Server Receiving Time:" + System.DateTime.Now.ToString("h:m:s:ms");
            DateTime servertime = System.DateTime.Now;
            TimeSpan diff = (clienttime - servertime).Duration();
((Server)Application.OpenForms["Server"]).lab4.Text = "Minimal Time=:" + diff.Hours.ToString() + ":" + diff.Minutes.ToString() + ":" + diff.Seconds.ToString() + ":" + diff.Milliseconds.ToString();
           // lblmes2.Text = "Message Send Successfully";
         }


         //if (lsthub.SelectedItem == null)
         //    MessageBox.Show("Select Hub");
         //else
         //else if (lsthub.Text == "191.128.2.2")
         //{
((Hub2)Application.OpenForms["Hub2"]).lab1.Text = "Last Update receieved from " + lblip.Text;
         //
((Hub2)Application.OpenForms["Hub2"]).lab2.Text = "Clent Node #:1:" + txtmessage.Text;
         //
((Hub2)Application.OpenForms["Hub2"]).lab3.Text = "Message Send Successfully";
         //     lblmes1.Text = "Client Node String Time:" + System.DateTime.Now.ToString();
         //    lblmes2.Text = "Message Send Successfully";
         //}
         //else if (lsthub.Text == "191.128.2.2")
         //else if (lsthub.Text == "191.128.2.3")
         //else if (lsthub.Text == "191.128.2.4")
       }
   }
}
```