Multipath Protocol for Delay-Sensitive Traffic

Umar Javed, Martin Suchara, Jiayue He, and Jennifer Rexford,
Princeton University, USA
Email: {ujaved, msuchara, jhe, jrex}@princeton.edu

Abstract—**Delay-sensitive Internet traffic, such as live streaming video, voice over IP, and multimedia teleconferencing, requires low end-to-end delay in order to maintain its interactive and streaming nature. In recent years, the popularity of delay-sensitive applications has been rapidly growing. This paper provides a protocol that minimizes the end-to-end delay experienced by inelastic traffic. We take a known convex optimization formulation of the problem and use an optimization decomposition to derive a simple distributed protocol that provably converges to the optimum. Through the use of multipath routing, our protocol can achieve optimal load balancing as well as increased robustness. By carrying out packet level simulations with realistic topologies, feedback delays, link capacities, and traffic loads, we show that our distributed protocol is adaptive and robust. Our results demonstrate that the protocol performs significantly better than other techniques such as shortest path routing or equal splitting among multiple paths.**

I. Introduction

With the rapid proliferation of broadband Internet access in the consumer market, new video communications and entertainment are gaining in popularity. Interactive applications such as live video streaming, voice over IP, multimedia conferencing, or online gaming are promoted by the ISPs who are often offering IPTV, VoIP, and Internet service in one bundle. In their recent report [1], Cisco predicts that in 2012, Internet video traffic will account for nearly 90% of all consumer IP traffic. Representative of this trend, Internet video has jumped to 22% of the global consumer Internet traffic in 2007 from 12% in 2006.

To preserve the interactive nature of the video applications, and to enable real-time-playback, data delivery with a low latency is required. As even a temporary delay in playback is likely to harm user experience, preventing delay jitter is equally important. Despite the growing popularity of the new interactive applications, the current Internet is not well equipped to support the delivery of delay-sensitive traffic. First, minimization of end-to-end latency by utilizing shorter, more direct routes has not received enough attention, and the existing solutions are not practical. Second, transmissions are vulnerable to transient periods of congestion that cause temporary delays or losses which degrade the quality of live audio or video streams.

Previous work in the Quality of Service (QoS) area [2] guarantees data delivery with the bandwidth and latency specified by the applications. However, this approach requires coordination in the network because it uses circuit reservations. Moreover, admission control needs to be used and if the delays increase above certain value, some sources are not allowed to transmit. We believe that a simpler more practical solution that does not require per-flow resource reservation and allows users to transmit at all times is needed.

A popular technique that improves reliability and robustness of data delivery is multipath routing [3]. This technique is beneficial because it allows rerouting of traffic if a path becomes congested or unavailable. Much of the previous work studies how multipath routing can improve the throughput of the end hosts. See e.g. [4] [5] [6] [7] [8]. While the key technique used in this paper is also multipath routing, the focus of our work is different.

In this paper, we try to address the issues faced by delay-sensitive traffic by redesigning routing and congestion control. The goals are to (i) minimize the delay of the network traffic, (ii) if possible, meet the rate demands of the users while avoiding congestion, and (iii) improve the robustness of the network and resilience to transient performance degradation. The desired design will ensure that the traffic uses paths with low delay while preventing congestion on the most popular shortest paths even when the loads in the network change.

Optimization theory is a powerful tool that has yielded remarkable results in networking research. It played a crucial role in the design of new distributed protocols for Network Utility Maximization (NUM) [9], [6], [10], reverse-engineering of existing protocols, particularly congestion control [11], [12], [13], and cross-layer optimized architectures in general [14]. In this paper we use convex optimization to optimize the delivery of delay-sensitive traffic.

Our optimization problem is a special case of the formulation that appears in [15]. We chose an objective that penalizes a combination of the delay experienced by the traffic sources and excessive link utilization in the network. The same problem receives treatment in [16] and [17]. However, our solution is much simpler and more practical. We use optimization decompositions [9] to obtain a *simple* distributed solution that can be carried out by the routers and traffic sources in the network. We also translate the distributed solution into a *practical protocol* and evaluate it through simulations in NS-2. These packet level simulations allow us to demonstrate the performance of the protocol in an environment with realistic network topologies and feedback delays.

The assumptions of our design are explained and justified in Section II. Delay minimization is formulated as a centralized optimization problem in Section III and the centralized formulation is translated into a distributed network protocol in Section IV. Finally, performance, robustness and dynamic properties of the protocol are evaluated in Section V and related work is discussed in Section VI.

II. Design Assumptions

Before we formulate the optimization problem that our protocol should be solving, let's consider the design decisions we need to make. This section describes these design decisions and justifies them. On one hand, we choose to use some simplifying assumptions that allow us to obtain an efficient distributed network protocol. For example, we assume that all the traffic in our network is delay sensitive. On the other hand, we need to extend the design of current networks to ensure that we have access to low delay paths. This is achieved by allowing the sources to use multipath routing with flexible splitting of traffic over these paths.

A. Network Serving Delay-Sensitive Traffic

The objective of our optimization is to minimize the average delay that a bit of data delivered in the network experiences. This objective implicitly assumes that all the traffic in the network is delay sensitive.

We choose to minimize the *average* delay for several reasons. First, minimizing the average delay will result in shorter delay for more traffic than, say, minimizing the maximum delay. The second reason is mathematical convenience that allows formulating the problem as convex optimization and solving it more easily.

The assumption that *all* traffic in the network is delay sensitive is justified for two reasons. First, as the proportion of video and audio transmissions increases, our protocol could be used to improve the delay experienced by all traffic. Second, the protocol could be used in a virtual network [18] that carries delay-sensitive traffic exclusively. Virtual networks subdivide network resources such as link capacities and router time among multiple virtual networks, each of which is used for a different application. For example, one virtual network can carry delay-sensitive traffic, another bandwidth intensive traffic, etc.

B. Sources with Traffic Splitting Over Multiple Paths

The traffic sources in the network have by assumption the ability to split the traffic between multiple paths. Moreover, the splitting ratios can change dynamically.

The main benefit of having *multiple paths* is an improved ability to decrease the end-to-end delay, and better reliability. Because the mathematics used in this paper does not specify whether the traffic sources are edge routers or end hosts, and whether the protocol is used in an interdomain or intradomain setting, we discuss the feasibility of using multiple paths and the changes required to the network architecture for these cases separately.

Using multiple routes in the intradomain setting is easier since all the traffic originates and terminates within a single autonomous system. Inside the network, routers can compute k shortest paths, or a management system can set-up multiple tunnels between the end hosts or edge routers. Multipath routing in the interdomain case is also possible. Today, most routing protocols only forward packets on a single path between a source and destination. However, as is explained in [6], the sources can direct traffic on a particular path by, e.g., participating in a routing overlay or, if they are multihomed, simply directing the traffic to one of the available upstream providers.

Dynamic load splitting improves performance as the sources are able to choose the path that has momentarily the lowest delay. Moreover, in case of failures, the source can avoid paths that do not work or that became congested due to a sudden traffic shift. If the sources are edge routers, they can rate limit the hosts and split the traffic among the available paths. If the sources are end hosts, they have to be given access to the multiple paths across which the splitting occurs.

C. Cooperative Sources and Routers

Developing a practical protocol that scales with the size of the network requires that we obtain a distributed protocol that can be performed by the sources and routers. We assume that the routers in the network are cooperative and they are willing to share information about the network, such as the link propagation delays and current level of congestion, with the sources. We assume that this signalling is performed honestly. We also assume that the sources obey the signals provided by the routers.

Cooperation is easy to achieve in the intradomain case where the routers are managed by a single network operator. However, honest signalling may not be incentive compatible in the interdomain case where routers are managed by multiple competing entities. This work could be extended to deal with misbehaving routers and traffic sources, but this is beyond this paper's scope.

D. Inelastic Traffic Demands

In our optimization problem formulation we assume that the demands of the sources are inelastic, i.e., they do not change as the conditions in the network change. It is important to notice that even with this assumption, the traffic sources are free to change their demands at any time. When the demands change, the algorithm re-converges to the new optimal solution.

Allowing the sources to change the demand is important as this corresponds to the requirements of applications in practice. It was observed that the bandwidth requirements of streamed video remains relatively constant in 10-30 second intervals, but can vary significantly between these intervals [19] due to factors such as variable rate compression. Our experimental results in Section V reflect this observation and demonstrate the performance of our protocol both for the case when the demands remain constant, and when the sources change them.

III. Optimization Problem Formulation

After introducing the notation used throughout the paper, this section formulates a central optimization problem that minimizes the average delay of network traffic delivery. We show that this formulation is equivalent to the 'optimal routing' formulation presented in [15] and [16].

A. Notation

A particular link in our network is denoted by l, its capacity is denoted by c_l, and its propagation delay by p_l. The network offers multipath routing, i.e., there are multiple paths connecting each source-destination pair. The paths are encoded in routing matrices. H^i is the routing matrix for a source-destination pair i, where $H^i_{lj} = 1$ if path j connecting source-destination pair i uses link l, and $H^i_{lj} = 0$ otherwise. $\mathbf{H}$ is the collection of all routing matrices H^i. $\mathbf{H}$ does not necessarily represent all possible paths in the physical topology, but a subset of paths chosen by the network operators.

The traffic between each source-destination pair is split among the multiple paths. z^i_j denotes the rate on path j connecting the source-destination pair i, and $(\mathbf{Hz})_l$ denotes the total load on link l. Each source-destination pair i has by assumption constant-bit-rate demand x_i. Since p_l is the propagation delay, it does not include the queueing delay that traffic may encounter when traversing link l. Therefore, we introduce function $f(\cdot)$ that (i) models the queueing delay of a link, and (ii) penalizes a link's utilization that approaches its capacity. How exactly is $f(\cdot)$ calculated is discussed later in this section.

B. Minimizing Aggregate Delay

The objective of our optimization problem is to minimize the aggregate end-to-end network delay encountered by the traffic. In another words, we need to distribute the inelastic traffic among the available paths in the network in such a way that the end-to-end delay in seconds/bit for all bits in the network is minimized. We also need to ensure that the constant demands x_i of the traffic sources are met, and the link loads do not exceed link capacity.

Since our algorithm needs to calculate the path rates on the paths that each source-destination pair uses, the variables are z^i_j. The delay on a path is the sum of the link propagation delays and queueing delays for all links on the end-to-end path. Therefore, the end-to-end delay on path j of source i is $\sum_l H^i_{lj}(p_l + f(\cdot))$. The objective of the optimization which captures the average delay for all traffic is obtained by minimizing the products of the path rates and the associated delays, i.e., minimize $\sum_i \sum_j z^i_j \left(\sum_l H^i_{lj}(p_l + f(\cdot))\right)$. This is equivalent to the minimization from [15]: minimize $\sum_l D_l(L_l)$ where $L_l = \sum_i \sum_j z^i_j H^i_{lj}$ is the load on link l and $D_l(L_l) = L_l(p_l + f(\cdot))$ is the cost associated with link l.

The solution to the optimization problem, therefore, assigns path rates in a way that takes maximum advantage of low propagation-delay paths while keeping the queues small to minimize overall delay. To ensure that link loads do not exceed link capacity, we introduce the constraint $(\mathbf{Hz})_l \preceq c_l$. We also need to enforce that the sum of the path rates between each source-destination pair i be equal to the demand x_i, i.e., $\sum_j z^i_j = x_i$. Our central optimization is in Figure 1.

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_i \sum_j z^i_j \left(\sum_l H^i_{lj}(p_l + f(\cdot))\right) \\ \text{subject to} & (\mathbf{Hz})_l \preceq c_l, \quad \forall l \\ & \displaystyle\sum_j z^i_j = x_i, \quad \forall i \\ & \mathbf{z} \succeq \mathbf{0} \\ \text{variables} & \mathbf{z} \end{array} \tag{1}$$

Fig. 1. Central problem for optimized delivery of delay-sensitive traffic.

The demands x_i in the constraints $\sum_j z^i_j = x_i$ are constant. First, this follows from the problem formulation which assumes inelastic traffic. Second, if x_i were variables, the objective of (1) would have to change to include the demands. Otherwise, since no traffic implies zero delays, all path rates would become zero.

It remains to explain how to choose $f(\cdot)$. We want to choose $f(\cdot)$ such that it represents queueing delays and heavily penalizes links that approach or exceed their capacity. This penalty is introduced with the goal of avoiding long queues as well as improving the robustness of the solution. For mathematical convenience, we also need the function be convex, non-decreasing and twice-differentiable. Previous optimizations in traffic engineering [20], [21] use the M/M/1 queueing formula

$$f(L_l, c_l) = \frac{1}{c_l - L_l}, \tag{2}$$

where L_l is the load on link l, and c_l its capacity. Unfortunately, (2) is not defined for overutilized links. Overutilization may occur during convergence. This problem is solved by letting $f(u_l)$ be a piecewise linear function [21] or an exponential approximation of the M/M/1 formula [22]. Here $u_l = \frac{L_l}{c_l}$ is the utilization of link l. Both the functions are well defined for any non-negative values of u_l. In our work, we chose the piecewise linear approximation for mathematical convenience and the ability to obtain closed-form rate updates.

C. Optimization Problem is Convex

In order to have a unique solution to (1), we need to show that the problem is convex. Since all the inequality constraints, i.e. the capacity constraints

$$(\mathbf{Hz})_l \preceq c_l, \quad \forall l$$

are linear and therefore convex, and all the equality constraints, i.e the demand constraints

$$\sum_j z^i_j = x_i, \quad \forall i$$

are affine, we only need to show that the objective

$$\sum_i \sum_j z^i_j \left(\sum_l H^i_{lj}(p_l + f(u_l))\right) \tag{3}$$

is convex over $z^i_j \in \mathbb{R}^+$. As pointed out above, (3) is equivalent to $\displaystyle\sum_l L_l(p_l + f(u_l))$, which is convex since $f(u_l)$ is non-decreasing, twice differentiable and convex. Hence (3) is convex.

IV. Distributed Multipath Protocol

In this section we will derive a practical protocol that converges to the solution of (1). First, we will use an optimization decomposition to convert the central optimization derived in the previous section into a protocol that solves the problem in a distributed fashion. In the course of finding the distributed solution, we will explain how mathematical convenience motivates our choice of $f(u_l)$. Finally, we will translate the distributed solution into a practical protocol which can be carried out by the routers and traffic sources.

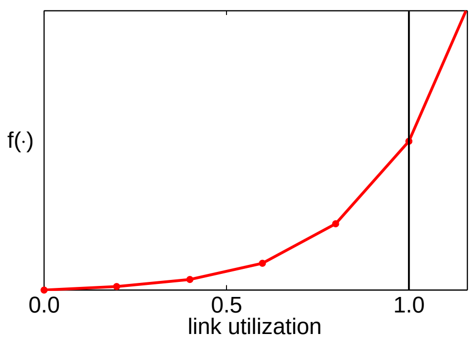

Fig. 2. The shape of the piecewise linear penalty function.

A. Optimization Decomposition

We begin by taking the Lagrangian of (1):

$$\begin{aligned} L(\mathbf{z},\lambda,\mathbf{q}) &= \sum_i \sum_j z_j^i \left(\sum_l H_{lj}^i (p_l + f(u_l)) \right) + \\ &\quad \sum_l \lambda_l ((\mathbf{Hz})_l - c_l) + \\ &\quad \sum_i q_i \left(x_i - \sum_j z_j^i \right) \\ &= \sum_i \sum_j z_j^i \left(\sum_l H_{lj}^i (p_l + \lambda_l + f(u_l)) - q_i \right) + \\ &\quad \sum_i q_i x_i - \sum_l \lambda_l c_l. \end{aligned} \quad (4)$$

Here λ_l is the dual variable for link l associated with the constraint $\sum_i \sum_j H_{lj}^i z_j^i \preceq c_l$, and q_i is the dual variable for source i associated with the constraint $\sum_j z_j^i = x_i$. Similarly as in [23], these dual variables can be thought of as prices for violating the respective constraints. In the distributed algorithm, these prices can be calculated through a subgradient method by the sources and the routers. The routers can then communicate the calculated prices to the traffic sources.

Since the problem is convex, we will use the KKT conditions [24] to find the optimal assignments of rates z_j^i. At time t in the distributed algorithm, the source i receives link prices $\lambda_l(t)$ from all distinct links in its paths and computes its own source price $q_i(t)$. Then by the KKT condition for the lagrangian the optimal path rate $z_j^{i\star}(t)$ on path j must satisfy

$$\frac{\partial}{\partial z_j^i} \left(L(\mathbf{z},\lambda,\mathbf{q}) \right) = 0. \quad (5)$$

For real-time computations, we are interested in a solution of (5) that gives us a closed form expression for $z_j^i(t)$. This depends on the choice of $f(u_l)$, and is not possible if we take $f(u_l)$ to be the M/M/1 queueing formula or an exponential. Therefore we approximate the M/M/1 formula with a piecewise linear function similar to the one introduced in [21].

The piecewise linear function is illustrated in Figure 2. $m(\frac{L_l}{c_l})$ is the function that determines the slope of the region which corresponds to the link utilization level $\frac{L_l}{c_l}$ of link l and $k(\frac{L_l}{c_l})$ is the function that determines the associated y-intercept, where $L_l = (\mathbf{Hz})_l = \sum_i \sum_j z_j^i H_{lj}^i$ is the link load. For simplification let $m_l = \frac{1}{c_l} m(\frac{L_l}{c_l})$ and $k_l = \frac{1}{c_l} k(\frac{L_l}{c_l})$. Therefore, $f(u_l) = m_l L_l + k_l$. Similarly as in [21], the values of $f(u_l)$ are small for low utilizations but increase rapidly as the load approaches or exceeds the capacity of the link.

Note that the piecewise linear model is still convex. To make it differentiable at the cutoff points, we need to define slopes at these points. This can be easily done by letting them take the slope of either of the adjoining lines. Now (4) can be rewritten as:

$$\begin{aligned} L &= \sum_i \sum_j z_j^i \left(\sum_l H_{lj}^i (p_l + \lambda_l + m_l (\mathbf{Hz})_l + k_l) - q_i \right) \\ &\quad + \sum_i q_i x_i - \sum_l \lambda_l c_l \end{aligned} \quad (6)$$

Differentiating (6) w.r.t. z_j^i and after a few simple algebraic manipulations we obtain:

$$\begin{aligned} \frac{\partial L}{\partial z_j^i} &= \sum_l H_{lj}^i (p_l + \lambda_l + m_l (\mathbf{Hz})_l + k_l) - q_i + \\ &\quad z_j^i \left(\sum_l H_{lj}^i m_l \right) + \sum_l H_{lj}^i m_l \left((\mathbf{Hz})_l - z_j^i \right). \end{aligned} \quad (7)$$

To compute optimal z_j^i, we set (7) to 0. After rearranging, introducing iteration index t, and denoting $(\mathbf{Hz})_l$ as L_l we obtain:

$$z_j^i(t+1) = z_j^i(t) + \frac{q_i(t) - \sum_l H_{lj}^i (p_l + \lambda_l(t) + 2m_l(t) L_l(t) + k_l(t))}{\sum_l H_{lj}^i m_l(t)}. \quad (8)$$

Similarly as in [23] and [12], the link feedback subgradient update for the capacity constraint in (1), $\lambda_l(t)$, is:

$$\lambda_l(t+1) = \left[\lambda_l(t) - \beta_\lambda(t) (c_l - L_l(t)) \right]^+. \quad (9)$$

Similarly, the source subgradient update for the demand constraint, $q_i(t)$, is:

$$q_i(t+1) = \left[q_i(t) - \beta_q(t) \left(\sum_j z_j^i(t) - x_i \right) \right]^+. \quad (10)$$

Here $\beta_\lambda(t)$ and $\beta_q(t)$ are the stepsizes at iteration t for the link price updates and the demand price updates respectively. For diminishing step size, i.e $\beta \rightarrow 0$ as $t \rightarrow \infty$, the distributed algorithms's objective will converge to the global objective [24].

B. Translation to a Network Protocol

Equations (8), (9) and (10) define a mathematical algorithm which has to be converted into a network protocol that can be performed by the routers and traffic sources.

The role of the routers is to monitor the performance on the links they are serving, calculate the prices associated with these links, and communicate these prices to the traffic sources. The routers update the link prices iteratively with granularity T. The prices are obtained by measuring the number of bits N_l^T that arrive on the link l during a time interval T, and comparing them to the link capacity c_l.

The sources collect the feedback from the routers and adjust the rates z_j^i accordingly. It is important to notice that the sources receive the feedback from the routers with a delay. Source i receives the feedback from the links in path j with delay RTT_j^i, the round-trip-time of that path. Therefore, we let source i update all its path rates at the timescale of the longest RTT. We denote this time interval $T_i = \max(\text{RTT}_j^i), \forall j$.

A closer inspection of (8) reveals that it is not sufficient for the source to learn $\lambda_l(t)$ from the routers. The sources are unable to calculate $F_l(t) = p_l + \lambda_l(t) + 2m_l(t)L_l(t) + k_l(t)$ and $G_l(t) = m_l(t)$ without the help of the routers. Therefore, we let the routers also calculate $F_l(t)$ and $G_l(t)$ along with the price $\lambda_l(t)$. Therefore, combining (8), (9) and (10) we obtain the protocol in Figure 3.

Feedback price update at link l:
$\lambda_l(t+T) = [\lambda_l(t) - \beta_\lambda (c_l - L_l(t))]^+$,
where β_λ is the feedback price stepsize, and $L_l(t) = \frac{N_l^T}{T}$.

Feedback computed at link l:
$F_l(t) = p_l + \lambda_l(t) + 2m_l(t)L_l(t) + k_l(t)$
$G_l(t) = m_l$.

Demand price computed at source i:
$q_i(t+T_i) = \left[q_i(t) - \beta_q \left(\sum_j z_j^i(t) - x_i\right)\right]^+$,
where β_q is the demand price stepsize.

Path rate update at source i, path j:
$z_j^i(t+T_i) = z_j^i(t) + \dfrac{q_i(t+T_i) - \sum_l H_{lj}^i F_l(t)}{\sum_l H_{lj}^i G_l(t)}$,
where $T_i = \max(\text{RTT}_j^i), \forall j$.

Fig. 3. Network protocol.

In the protocol in Figure 3, we dropped the dependence of the stepsizes β_λ and β_q on time. Implementation of diminishing stepsizes would be impractical as the stepsizes would have to increase whenever a new flow enters or leaves the network. Choosing a constant stepsize simplifies the protocol, but requires finding the proper values of the stepsizes. [24] shows that even with constant stepsizes, the subgradient method converges to within ϵ of the optimal value, where ϵ is a decreasing function of the stepsize. However, if the stepsizes are too large, the solution may end up too far from the optimum, while if the stepsizes are too small, the convergence of the protocol may be very slow. The choice of the best numerical values of β_λ and β_q is further discussed in Section V. To avoid division by zero, i.e. $\sum_l H_{lj}^i G_l(t) \neq 0$, we have $m_l \geq 0$ at all times.

The protocol in Figure 3 does not always assign the source rates z_j^i such that $x_i = \sum_j z_j^i(t)$. While after the protocol converges the equality holds, during transients the price q_i can be non-zero, and hence the demand constraint may be violated. We solve this problem by projecting the source rates z_j^i onto the feasible region during the transients so that the demands are met and all rates are non-negative. As shown in [15], projection onto the feasible region doesn't affect optimality of convergence. In particular, after each path rate update, we project the rates of each source in the j dimensional space onto the closest point that satisfies the constraints.

V. Evaluation

The goal of this section is to demonstrate the performance of the protocol and its dynamic properties. We use NS-2 to perform simulations on realistic network topologies and in the presence of feedback delay. First, we will describe our experimental setup in NS-2 and the topologies we used. Second, we will describe how we selected the values of the two stepsizes, β_λ and β_q. Third, we will compare the performance of our protocol against two other simple schemes. Finally, we will test the dynamic properties of the protocol by performing simulations with stochastic traffic.

A. NS-2 Simulations and Topologies

We implement the protocol of Figure 3 in NS-2. The link prices are updated every 5 ms and fed back to the sources in acknowledgement packets. It is important to notice that the prices are delayed because they are attached by the routers to packets and returned to the traffic sources in the acknowledgements. The sources then update their rates every $\max(\text{RTT}_j^i)$ ms. Large router buffers increase propagation delays during congestion. Since Section V-C compares our protocol against two schemes which are more likely to cause congestion, we chose short buffers (5 packets) for all evaluations.

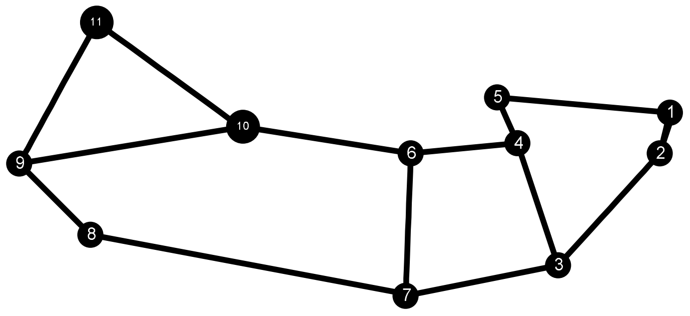

Fig. 4. Abilene topology used in NS-2 simulations.

In our simulations, we used both simple synthetic and more complicated realistic topologies. One of the topologies, Abilene, is depicted in Figure 4. The link capacities in Abilene were 100 Mb/s and delays approximated realistic values between the respective pairs of cities. The routing in the network was defined as follows. We selected 4 source-destination pairs, and connected each pair by 4 distinct paths. The round trip delays on these paths ranged between 28 ms and 82 ms.

Since the protocol can be easily deployed inside of a single autonomous system, we obtained autonomous systems' topologies along with link delays from the Rocketfuel topology mapping service [25]. The link capacities were 100 Mb/s if neither endpoint of the link has degree larger than 7, and 52 Mb/s otherwise. We experimented with the topology of the Sprint network, using 25 source-destination pairs connected by 4 paths. To obtain the paths connecting a source and destination in the Sprint topology, a third transit node was selected, then a shortest path connecting all the three nodes was calculated, and finally we removed any cycles from the paths.

B. *Tuning the Stepsizes*

The protocol has two step sizes: β_λ, associated with the link capacity constraint, and β_q, associated with the source demand constraint. In order to find the proper value of β_λ, we experimented with the protocol in MATLAB. We chose MATLAB rather than NS-2 because it allowed us to sweep the parameter space quicker. We experimented with link capacities of 1Mbps, 10Mbps, and 100Mbps, and chose demands that were sufficiently high to create congestion. We observed the speed of convergence for different values of β_λ. The results are summarized in Table I.

	10 Mbps	**100 Mbps**	**1000 Mbps**
$\beta_\lambda = 1 * 10^{-9}$	very slow	slow	slow
$\beta_\lambda = 5 * 10^{-9}$	very slow	slow	slow
$\beta_\lambda = 1 * 10^{-8}$	slow	slow	fast
$\beta_\lambda = 5 * 10^{-8}$	slow	slow	no
$\beta_\lambda = 1 * 10^{-7}$	slow	fast	no
$\beta_\lambda = 5 * 10^{-7}$	slow	no	no
$\beta_\lambda = 1 * 10^{-6}$	fast	no	no

TABLE I
RATE OF CONVERGENCE FOR DIFFERENT β_λ VALUES AND DIFFERENT LINK CAPACITIES.

Taking a close look at the step sizes and capacities, we observe that higher link capacities require smaller β_λ to achieve quick convergence. We conclude that the optimal value of the stepsize is $\beta_\lambda = 1/c_l$. We used the same method to find the optimal value for β_q and found out that the stepsize of each source depends on its demand. The optimal value is $\beta_q = 0.5/x_i$ where x_i denotes the demand of source i.

C. *Performance Comparisons*

To objectively assess the performance, we implemented two other simple protocols in NS-2 and compared the performance against our protocol on the Abilene topology. The first protocol we compare against uses shortest path routing, and the second uses equal traffic splitting among multiple paths. We describe these protocols next.

In the shortest path routing protocol, each source i measures the propagation delays on the j paths that are available to it. Then, all traffic is simply sent on the path with the lowest delay. The protocol that uses equal splitting splits traffic equally among the j paths that are available to source i.

In this experiment, we slowly increase the traffic demands from 20 Mbit/sec to 60Mbit/sec and observe the resulting losses and delays. The losses are depicted in Figure 5. We observe that the performance of the protocol that uses shortest path routing is the worst. This is expected because the protocol cannot spread the loads on the longer less used paths when the loads increase beyond 30 Mbit/sec and congestion occurs. The performance of the two protocols which use load balancing is nearly identical. Our protocol performs slightly better because it dynamically adjusts the fraction of traffic sent on each path.

The average propagation delay of the delivered traffic measured during the same experiment is depicted in Figure 6. The delay of shortest path routing is the same as our protocol for demands below 30 Mbit/sec, i.e., in the region where the shortest path algorithm can meet the demands. For larger demands, shortest path routing cannot deliver all the traffic. The delay for equal splitting is larger because, unlike our protocol, it does not place greater load on the shorter paths during periods of low congestion. We conclude that our protocol is able to balance the load in such a way so that the propagation delays are low when the network is not congested, but shifts traffic to the longer less used paths as congestion increases.

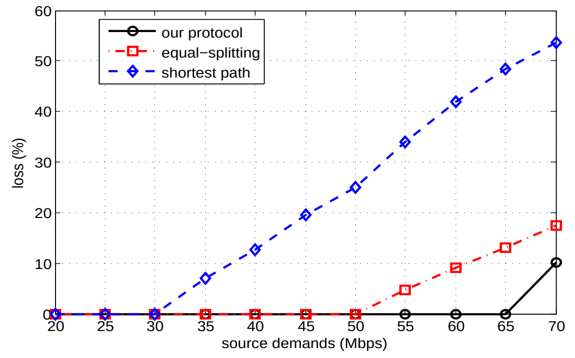

Fig. 5. Packet loss. Our protocol, equal splitting and shortest path routing are used respectively. The source rates increase from 20 Mbit/sec to 60 Mbit/sec.

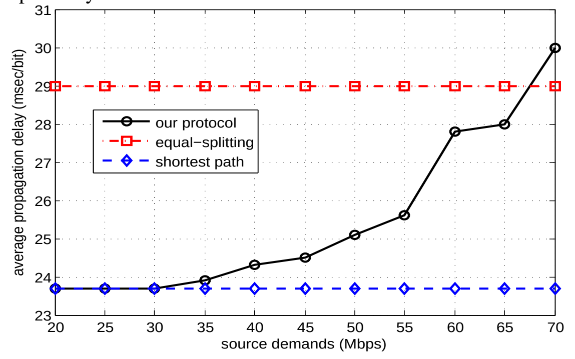

Fig. 6. The average propagation delays. Our protocol, equal splitting and shortest path routing are used respectively. In the region of the graph without packet losses, our protocol has the same or lower delay than the other two.

D. *Convergence to Optimum*

To assess the speed of convergence of the protocol, we compared the optimal value of the optimization objective with the achieved value after the sources start the transmission. The simulation was performed on the Abilene topology using three different traffic demands for all the sources: 30 Mbit/sec, 45 Mbit/sec and 60 Mbit/sec respectively. The results are depicted in Figure 7. The plot shows that the objective achieved by our protocol converges quickly to within a few percent of the optimal value. The optimal value was obtained by solving the optimization problem from Figure 1 numerically.

We observe that the speed of convergence is better for slower demands. This is because the amount of traffic that needs to be split among the available paths is smaller. At first glance, the speed of convergence is slow. However, this is not a major concern for two reasons. First, in this experiment, all flows start at time 0, which makes the scenario more challenging. In practice, such a synchronization is not likely. Second, as mentioned in Section IV, our algorithm uses projections to meet the demands of the sources. Therefore, even shortly after the start of the experiment, the demands of the sources are met, although the distribution of the load among the available paths is suboptimal with respect to delay during the transient period. The convergence of the average delays is depicted in Figure 8.

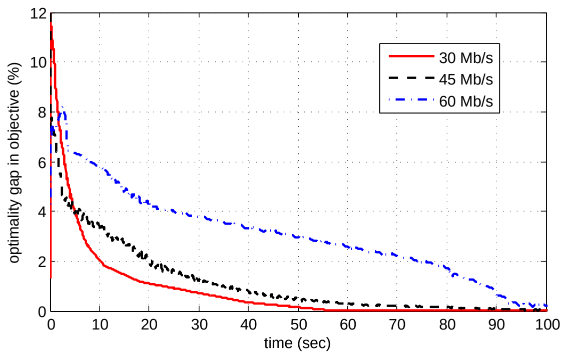

Fig. 7. The convergence to optimum is faster for smaller demands.

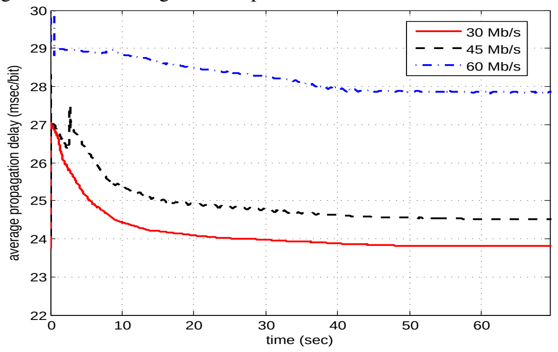

Fig. 8. The delays converge to smaller values in configurations with lower demands.

E. *Dynamic Properties*

In the previous simulations we assumed that the traffic demands do not change. Now we will relax this assumption and observe the behavior of the protocol when the demands change randomly. We start at time 0 by choosing the demand of each source uniformly at random between 40 Mbit/sec and 60 Mbit/sec. We keep the demands constant for a time interval of length t. After the time interval elapses, we resample all demands from the same distribution. This is repeated for each time interval. The length of each time interval is selected uniformly at random between 35 sec and 60 sec.

In this simulation, the demand adjustments are synchronized. In practice, the demand adjustments may or may not be synchronized. We selected synchronization for two reasons. First, this is a more arduous test than changing the demands for each source independently. Second, the optimal value of the objective changes less frequently, which makes it easier to compare against the achieved value of the objective.

The simulation was performed on the Abilene topology, and the results are depicted in Figure 9. The black curve depicts the optimal value of the objective, which was obtained by solving the optimization problem from Figure 1 numerically. The red curve depicts the value achieved by our protocol. We observe that after each demand adjustment, the achieved value quickly converges to the optimum.

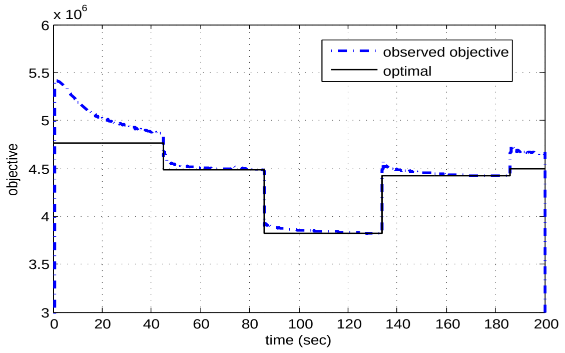

Fig. 9. The source demands change. After each change, the optimal value of the objective changes (black curve), and our protocol converges to the new optimum (red curve).

VI. Related Work

Previous work that improves the latency of Internet traffic falls into three groups. The first group consists of papers whose primary goal is to minimize these delays in context of 'optimal routing'. The second and larger group consists of papers that decrease the delays as a byproduct of designs that address other problems. Finally, the third group uses admission control to guarantee a desired quality of service (QoS).

[16] and [17] minimize propagation delays in the network by specifying for each node i what fraction of the traffic should leave node i on each of the links emanating from it. While this work uses multipath routing and theoretically achieves the optimal values of delays, it has several drawbacks. First, in [16] it is not possible to find a stepsize that would offer good convergence for a range of traffic conditions. This is solved in [17], but the work requires more complicated calculations, such as estimation of both the lower and upper bound of

the second derivative of the delays. Second, the protocols require synchronized message passing where upstream nodes need message updates from downstream nodes before they can compute their own updates. This also requires that the network be 'loop-free': node i cannot be upstream and downstream of node j at the same time to avoid deadlocks. In comparison, our approach is more practical and does not require significant changes in current routing protocols.

Proposals that help to improve end-to-end delay also include TCP protocols such as Vegas [11], or FAST [10] that decrease the amount of queueing in the router buffers. Decreasing the physical length of the buffers [26] also decreases the delays. Since the goal of these papers is not to optimize the delivery of delay-sensitive traffic, they do not offer optimal delays and robustness.

QoS approaches are able to guarantee the delay and bandwidth requirements to the applications. This is achieved by using admission control to select sources that are allowed to use the network. For a survey of the work in this area see e.g. [2]. While our work does not provide delay or bandwidth guarantees, we are able to minimize the aggregate delays without using admission control.

VII. Conclusion

In this paper we identified the need to develop a new protocol that would better serve the needs of interactive applications such as IPTV, VoIP or videoconferencing. We started with the observation that these applications would benefit from low latency as well as better robustness and resilience to transient performance degradation. With these goals in mind, we used optimization theory to design a new protocol that modifies routing and congestion control.

We took a convex optimization problem that minimizes the latency by splitting traffic on multiple paths while making sure that the shortest most popular paths do not become congested. Then we used optimization decomposition to translate the optimization problem into a distributed algorithm and a practical protocol with two tunable parameters. Simulations in NS-2 allowed us to demonstrate the robustness and low latency of our solution.

The robustness of the protocol was demonstrated by comparing it against two other simple algorithms that can be used to reduce latency, namely shortest path routing and equal splitting among paths. We concluded that our protocol is able to shift traffic on longer paths before the shortest paths become congested. We also observed that the protocol achieves a much lower latency than equal splitting, and is able to operate without any losses at higher loads than the other schemes.

We see several possible extensions of this work. Economic interpretation of prices in our protocol would provide monetary incentives for the sources and routers to behave honestly. This would be especially helpful in interdomain deployments that span networks belonging to competing entities. Another extension could simplify the protocol by having the sources to obtain the feedback from the network implicitly by measurement rather than having to rely on the explicit feedback from the routers.

References

[1] "Cisco Visual Networking Index – Forecast and Methodology, 2007 – 2012."

[2] X. Xiao and L. Ni, "Internet QoS: a big picture," *IEEE Network*, vol. 13, pp. 8–18, March 1999.

[3] J. He and J. Rexford, "Towards Internet-wide Multipath Routing." IEEE Network Magazine, Special Issue on Internet Scalability, March 2008.

[4] J. He, M. Chiang, and J. Rexford, "TCP/IP Interaction Based on Congestion Price: Stability and Optimality," in *Proc. International Conference on Communications*, June 2006.

[5] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 5–12, April 2005.

[6] J. He, M. Suchara, M. Bresler, J. Rexford, and M. Chiang, "Rethinking Internet traffic management: From multiple decompositions to a practical protocol," in *Proc. CoNEXT*, December 2007.

[7] X. Lin and N. B. Shroff, "Utility Maximization for Communication Networks with Multi-path Routing," *IEEE Trans. Automatic Control*, vol. 51, May 2006.

[8] F. Paganini, "Congestion Control with Adaptive Multipath Routing Based on Optimization," in *Proc. Conference on Information Sciences and Systems*, March 2006.

[9] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. on Selected Areas in Communications*, vol. 24, pp. 1439–1451, August 2006.

[10] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. IEEE INFOCOM*, March 2004.

[11] S. H. Low, L. Peterson, and L. Wang, "Understanding Vegas: A duality model," *J. of the ACM*, vol. 49, pp. 207–235, March 2002.

[12] S. H. Low, "A duality model of TCP and queue management algorithms," *IEEE/ACM Trans. Networking*, vol. 11, pp. 525–536, August 2003.

[13] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[14] M. Chiang, S. H. Low, R. A. Calderbank, and J. C. Doyle, "Layering as optimization decomposition," *Proceedings of the IEEE*, January 2007.

[15] D. Bertsekas and R. Gallager, *Data Networks*, ch. 5.4. Prentice Hall, second ed., 1992.

[16] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 73–85, 1977.

[17] D. Bertsekas, E. Gafni, and R. Gallager, "Second derivative algorithms for minimum delay distributed routing in networks," *IEEE Transactions on Communications*, vol. 32, no. 8, pp. 911–919, 1984.

[18] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "DaVinci: Dynamically Adaptive Virtual Networks for a Customized Internet," in *Proc. CoNEXT*, December 2008.

[19] T. Borsos, "A Practical Model for VBR Video Traffic with Applications," in *MMNS '01: Proceedings of the 4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, pp. 85–95, Springer-Verlag, 2001.

[20] K. G. Ramakrishnan and M. A. Rodrigues, "Optimal routing in shortest-path data networks," *Lucent Bell Labs Technical Journal*, vol. 6, no. 1, 2001.

[21] B. Fortz and M. Thorup, "Optimizing OSPF weights in a changing world," *IEEE J. on Selected Areas in Communications*, vol. 20, pp. 756–767, May 2002.

[22] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards Robust Multi-layer Traffic Engineering: Optimization of Congestion Control and Routing," *IEEE J. on Selected Areas in Communications*, June 2007.

[23] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. of Operational Research Society*, vol. 49, pp. 237–252, March 1998.

[24] S. Boyd and L. Vanderberghe, *Convex Optimization*. Cambridge University Press, 2004.

[25] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel," *IEEE/ACM Trans. Networking*, vol. 12, pp. 2–16, February 2004.

[26] I. Keslassy and N. McKeown, "Sizing router buffers," in *Proceedings of ACM SIGCOMM*, pp. 281–292, 2004.