Optimizing Cost and Performance for Multihoming*

David K. Goldenberg⋆ Lili Qiu† Haiyong Xie⋆ Yang Richard Yang⋆ Yin Zhang§
AT&T Labs – Research§ Microsoft Research† Yale University⋆

{david.goldenberg,haiyong.xie,yang.r.yang}@yale.edu liliq@microsoft.com yzhang@research.att.com

ABSTRACT

Multihoming is often used by large enterprises and stub ISPs to connect to the Internet. In this paper, we design a series of novel *smart routing* algorithms to optimize cost and performance for multihomed users. We evaluate our algorithms through both analysis and extensive simulations based on realistic charging models, traffic demands, performance data, and network topologies. Our results suggest that these algorithms are very effective in minimizing cost and at the same time improving performance. We further examine the equilibrium performance of smart routing in a global setting and show that a smart routing user can improve its performance without adversely affecting other users.

Categories and Subject Descriptors

C.2.6 [**Computer-Communication Networks**]: Internetworking—*Internet*

General Terms

Algorithms, Performance

Keywords

Multihoming, Smart Routing, Optimization, Algorithms

1. INTRODUCTION

Multihoming [31] is often used by large enterprises and stub ISPs to connect to the Internet because of its perceived benefits in reliability, cost, or performance. A customer or ISP network (also called a user) with multiple external links (either to a single ISP, or to different providers) is said to be *multihomed* [31]. When a user actively controls how its traffic is distributed among its multiple links to the Internet, we refer to it as implementing *smart routing*. Smart routing is also referred to as route optimization, or intelligent route control.

Smart routing can potentially be useful in the following ways. First, smart routing may help to improve network performance and reliability. Recent studies [27, 32, 33] have shown that when compared with the ideal routing, network-level routing often yields suboptimal user performance due to routing hierarchy and BGP policy routing. Equipment failure, transient instability, and network congestion may also affect user performance. Smart routing offers a way for end users to exercise control over routes. In [1], Akella *et al.* quantify the potential benefits of smart routing and suggest that selecting the right set of providers can yield some performance improvement. In [2], Akella *et al.* observe that the latency and throughput achieved by multihoming to three ISPs are within 5-15% of overlay routing employed in conjunction with 3-multihoming. Second, when taking into account specific charging models, smart routing can potentially reduce users' financial cost. A recent economic analysis shows that smart routing has the potential to benefit not only the end users, but also the service providers [8].

Given the potential benefits of smart routing and the large number of multihomed users, many companies are actively developing software to implement smart routing, *e.g.*, [12, 19, 21, 24]. However, since these are commercial products, their technical details are not available, and their performance and impact on the Internet are not well understood. While there are a few research studies on smart routing, *e.g.*, [1, 11], the focus of these studies is on network performance only; users' cost, which is another major incentive to use multihoming, is not considered. In addition, previous studies focus on the *potential* performance benefits, not on the design of algorithms; it remains an open question how such potential benefits can be achieved in practice.

In this paper, we attempt to realize some of the benefits of smart routing by developing a series of novel algorithms for optimizing both cost and performance for multihomed users. We first demonstrate that optimizing network performance alone can significantly increase the cost of a user, thus rendering smart routing less attractive. To address this issue, we propose novel offline and online routing algorithms to minimize a user's cost under common usage-based charging models. Using realistic pricing data and traffic demand traces from universities and enterprises, we show that despite fluctuations in traffic, our online algorithm can significantly reduce a user's cost, compared with using dedicated links or applying round robin or equal split algorithms to burstable links. We also design online and offline algorithms to optimize the network performance of a smart routing user under cost constraints. Using realistic pricing data, traffic demand traces, and latency traces, we show that our online algorithm achieves performance within 10–20% of the optimal offline algorithm.

In this paper we assume that the user is already multihomed to a set of ISPs. Thus, we focus on how to dynamically assign traffic among them to optimize cost and performance. The business decisions of whether to use multihoming and which ISPs to choose are by themselves very complicated and may involve many technical and non-technical factors, which we do not attempt to address in

*David Goldenberg is supported in part by NSF Graduate Research Fellowship DGE0202738. Haiyong Xie is supported in part by NSF grant ANI-0238038. Yang Richard Yang is supported in part by NSF grants ANI-0207399 and ANI-0238038.

SIGCOMM'04, Aug. 30–Sept. 3, 2004, Portland, Oregon, USA.

this paper. We also assume that cost and performance are the main factors of interest to the user. For many real Internet services such as Virtual Private Networks (VPNs), however, optimizing cost and performance alone may not be enough. Other factors such as ease of management, ease of trouble-shooting, security, and Quality-of-Service (QoS) also play critical roles in users' business decisions. So our techniques are not directly applicable in such contexts. Nevertheless, we believe that in order to better understand the potential role of smart routing in the future Internet, it is important to go beyond previous performance-centric studies by placing both cost and performance in a common optimization framework.

Besides developing techniques for optimizing cost and performance, we also evaluate the global effects of such optimization. We note that smart routing becomes a *selfish* routing scheme when each individual smart routing user adaptively changes its routes to optimize its own metrics without considering its effects on the network. Such adaptation changes network performance and may cause self-interference or interference with other smart routing or regular (*i.e.*, single-homed) traffic. It remains to be seen whether smart routing can retain its performance benefits in the presence of such forms of interference.

We use extensive simulations to study the global effects of smart routing. We first examine the equilibrium performance of smart routing in the presence of self-interference (*i.e.*, when the routing decisions made by the smart routing user change the network performance, which in turn interferes with the decisions involved in smart routing). Our results suggest that even in the presence of self-interference, our algorithms still achieve good equilibrium performance. We then evaluate how smart routing traffic interacts with other smart routing traffic as well as with single-homed traffic. Our evaluations are based on an inter-domain network topology and user demands from real traffic traces. Our results show that smart routing improves performance without degrading the performance of other traffic.

Our key contributions can be summarized as follows:

- We design offline and online algorithms to minimize cost based on realistic usage-based charging models.
- We design offline and online algorithms to optimize network performance under cost constraints.
- We use both analysis and simulations based on realistic traffic and performance data to demonstrate that our algorithms yield good performance and low cost.
- We evaluate the performance of smart routing when multiple users selfishly optimize their own cost and performance. We find that under this setting, smart routing traffic interacts well with other traffic under traffic equilibria.

The rest of this paper is organized as follows. In Section 2, we review related work. In Section 3, we discuss our network and charging models. In Section 4, we motivate the importance of optimizing cost and present novel cost optimization algorithms. In Section 5, we optimize network latency under given cost constraints. We present the methodology and results of our evaluations in Section 6. In Section 7, we study the global effects of smart routing and evaluate its interactions with other traffic. We conclude and discuss future work in Section 8.

2. RELATED WORK

Several recent studies have shown that Internet routing often yields sub-optimal user performance, *e.g.*, [4, 27, 32, 33]. There are a number of contributing factors, including routing hierarchy, policy routing, and slow reaction (if any) to transient network congestion or failures. BGP routing instabilities can further exacerbate the problem. These observations have generated considerable research interest in offering end-users more control in route selection. For instance, the authors in [4, 27] propose using overlay routing to improve user performance. Achieving a large scale deployment with this approach is challenging, as cooperation among multiple organizations is not easy to arrange in practice.

Multihoming is an alternative way to enable users to control routes. Many large enterprises, stub ISPs and even small businesses already use multihoming as a way to connect to the Internet.

Much of the previous work on multihoming focuses on how to design protocols to implement multihoming, *e.g.*, [5, 7, 11, 30]. For example, the authors of [5, 7, 12, 24, 30] use BGP peering as an implementation technique. Another technique is through DNS or NAT, which is used in [9, 21]. Our work differs from the above in that we do not focus on the implementations, but instead on designing algorithms to determine when and how much traffic a user should assign to different ISPs to optimize both performance and cost. Consequently, our work is complementary to the above.

There are several papers that evaluate the benefits of smart routing, including [8, 28, 29]. More recently, Akella *et al.* [1] quantify the potential performance benefits of multihoming using real Internet traces. Their results show that smart routing has the potential to achieve an average performance improvement of 25% or more for a 2-multihomed user in most cases, and that most of the benefit can be achieved using 4 providers. Motivated by these results, we seek to develop routing schemes to achieve such benefits in practice. In addition, we study the effects of un-coordinated route optimization by multiple mutually interfering smart routing users.

Finally, there are a few research studies on designing algorithms for smart routing, *e.g.*, [1, 15, 17]. For example, Orda and Rom [17] investigate where to place multihomed users and show that the problem is NP-hard. Cao *et al.* [6] propose using hash functions to achieve load balancing among multiple links. In [11], the authors compare several route selection schemes in a local area network and show that hashing can achieve performance comparable to load-sensitive route selection. Our work differs from these studies in that we use both cost and network performance as metrics of interest. We also study the interactions between multiple smart routing users, and between smart routing and single-homed users.

3. NETWORK AND CHARGING MODELS

In this section, we describe our network model, ISP charging models, and the performance metric we use.

3.1 Network Model

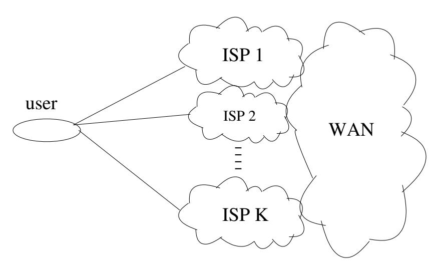


Figure 1: An illustration of a user with K service providers.

A multihomed user has multiple links to the Internet for sending and receiving traffic, as shown in Figure 1. The implementation techniques of distributing traffic to the links are different for outgoing and incoming traffic. For outgoing traffic, a border router inside the user's network can actively control how traffic is distributed. For incoming traffic, a user can use NAT or DNS to control the routes. For more detailed discussions about the implementations, we refer the readers to [1, 5, 7, 11, 30].

Note that the implementations of multihoming are complementary to our study, as our focus is on determining when and how much traffic should be assigned to each link. Consequently, our algorithms can be applied to a wide variety of multihoming implementations, and work for both out-going and incoming traffic. Since our traffic traces, as described below, consist of only outgoing traffic, we evaluate traffic assignment only in the out-bound direction in this paper.

3.2 Charging Models

Users pay ISPs for using their service. The cost incurred is usually based on the amount of traffic a user generates, *i.e.*, $cost = c(x)$, where x is a variable determined by a user's traffic (which we will term the *charging volume*) and c is a non-decreasing function that maps x to cost. Various charging models differ from one another in their choices of charging volume x and cost function c.

Usually, the cost function c is a piece-wise linear (non-decreasing) function, which we will use for our design and evaluation. There are several ways in which the charging volume x can be determined. Percentile-based charging and total-volume based charging are both in common use.

- *Percentile-based charging*: This is a typical usage-based charging scheme currently in use by ISPs [26]. Under this scheme, an ISP records the traffic volume a user generates during every 5-minute interval. At the end of a complete charging period, the q-th percentile of all 5-minute traffic volumes is used as the charging volume x for q-percentile charging. More specifically, the ISP sorts the 5-minute traffic volumes collected during the charging period in ascending order, and then computes the charging volume x as the $(q\% \times I)$-th volume, where I is the number of intervals in a charging period. For example, if 95th-percentile charging is in use and the charging period is 30 days, then the cost is based on the traffic volume sent during the 8208-th ($95\% \times 30 \times 24 \times 60/5 = 8208$) sorted interval.

- *Total-volume based charging*: This is a straightforward charging model, where the charging volume x is the total volume of traffic a user generates during the entire charging period.

In this paper, we focus primarily on percentile-based charging. We describe how to deal with total-volume based charging in Appendix C. In our evaluations, we use two sets of pricing functions. The first set of functions are simple pricing functions: if the charging volume is 0, the price is 0; otherwise, the price is a constant value. We pick the values from the entries in Table 1, which is published in [25].[1] In this table, a burstable link is a link whose price is determined by the percentile-based charging model; a full-rate link is also called a dedicated link and has a fixed price independent of usage. To evaluate the sensitivity of our algorithms to cost functions, we also use another set of functions shown in Figure 2. These functions are the more complex step functions. The prices at 24 Mbps for DS3 and at 100 Mbps for OC3 match those in Table 1. The overall trend of the pricing curves reflects the general pricing practice of decreasing unit cost as bandwidth increases; it is also consistent with the pricing curves we are aware of, *e.g.*, [3, 18].

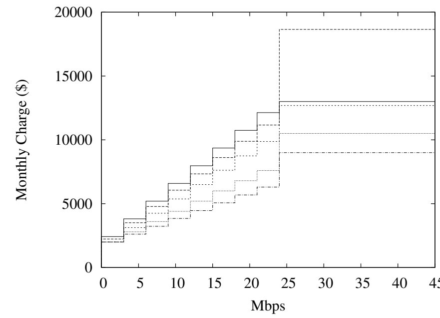
3.3 Network Performance Metric

There are several ways to measure network performance. In our evaluations, we use end-to-end latency as the metric. As shown in [24], latency not only reflects network response time but also serves as a measure of availability, as users often consider large delay or rapidly increasing latency as a sign of a potential availability problem. Our algorithms can be easily extended to work with alternative metrics, *e.g.*, a combination of latency and loss rate: $latency + w \cdot \log(\frac{1}{1-lossRate})$, where w is the relative weight of loss rate.

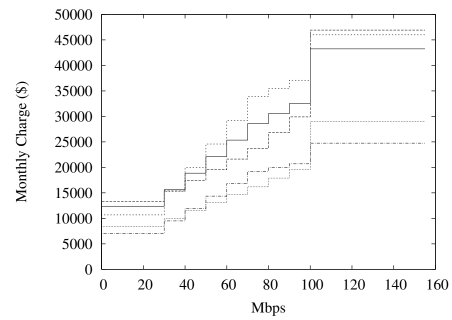
[1] Prices are based on a February 2002 Blind RFP. All prices listed are discounted, based on a two-year, $240,000 annual commitment with installation in a San Jose, CA facility. Prices do not include recurring and non-recurring local access fees. Given a set of dedicated links available, the cost of using dedicated links is the minimum cost of links such that their total capacity exceeds the maximum traffic load. Note that the price of ISP5 is corrected from $23,750 in the original publication to $28,750.

ISP	Burstable DS3 (24 Mbps)	Full-rate DS3 (45 Mbps)	Burstable OC3 (100 Mbps)	Full-rate OC3 (155 Mbps)
ISP1	$12,131	$13,000	$32,500	$43,245
ISP2	$11,160	$18,652	$29,900	$46,930
ISP3	$ 9,870	$12,690	$37,088	$45,989
ISP4	$ 7,600	$10,500	$19,600	$29,000
ISP5	$ 6,300	$ 9,000	$24,700	$28,750

Table 1: The pricing data.



(a) DS3 pricing functions



(b) OC3 pricing functions

Figure 2: The more complex pricing functions.

4. MINIMIZING TOTAL COST

Since the previous studies focus on improving network performance without considering cost, we first motivate the need to optimize cost. We show that by optimizing network performance alone, users may incur high cost. Since the percentile-based charging model is in common use, we illustrate our point below using a simple example under this model. Our performance results in Section 6 will further support this point using real data.

Consider a user with K identical links to K ISPs. Suppose the user has one unit of traffic to send at each interval, and the latency of each link at each interval is drawn uniformly from a common range. In each interval, to minimize latency, the user sends all of its traffic through the link with the lowest latency while the other links receive no traffic. Since the latency through different links are identically distributed, each link receives traffic for approximately $1/K$ of the intervals. Therefore when K is less than 20, *e.g.*, $K = 4$, the 95th-percentile of each link is one. This implies that by optimizing performance, the user pays K times the cost of using a single link. This K-fold increase in cost is clearly unacceptable to most users.

Given this potential of a large cost increase, in this section we study how to design effective smart routing algorithms to optimize cost. As mentioned in Section 3, we focus on the percentile-based charging model. We present an algorithm for dealing with the total-volume based charging model in Appendix C.

4.1 Problem Specification

We first introduce the following notation.

K	The number of ISPs. We use k as the index.
c_k	The cost function of ISP k. Without loss of generality, we assume that c_k is a piece-wise linear non-decreasing function.
I	The number of time intervals in a charging period. We use i as the index.
F	The number of flows. We use f as the index.
$v_f^{[i]}$	The traffic volume of flow f during interval i.
$v^{[i]}$	The total traffic volume during interval i, *i.e.*, $v^{[i]} = \sum_f v_f^{[i]}$. Let time series $V = \{v^{[i]} \mid 1 \le i \le I\}$.
$t_k^{[i]}$	The volume of traffic distributed to ISP k during interval i. Let time series $T_k = \{t_k^{[i]} \mid 1 \le i \le I\}$. Note that $V = \sum_k T_k$ (with vector summation).
q_k	The charging percentile of ISP k, *e.g.*, $q_k = 0.95$ if an ISP charges at 95th-percentile.
$\mathtt{qt}(X, q)$	The $\lceil q * \lvert X \rvert \rceil$-th value in X_{sorted} (or 0 if $q \le 0$), where X_{sorted} is X sorted in non-decreasing order, and $\lvert X \rvert$ is the number of elements in X.
p_k	The charging volume of ISP k, (*i.e.*, $p_k = \mathtt{qt}(T_k, q_k)$). For example, if ISP k charges at 95th-percentile, then p_k is the 95th-percentile of the traffic assigned to ISP k.

We now formally specify the *flow assignment problem*: Given cost functions c_k, the flow assignment problem is to find $t_k^{[i]}$ that minimizes the total cost $\sum_{k=1}^{K} c_k(p_k)$.

We consider two cases: *fractional flow assignment* and *integral flow assignment*. Under fractional flow assignment, flows are infinitely splittable. In contrast, integral flow assignment assumes that during each interval each flow is assigned to only one ISP. In the latter case, flows can be naturally defined using destination prefixes when BGP is used to implement smart routing.

The traffic assignment problem, be it fractional or integral, can be further classified into two categories: *offline* or *online*. The offline version assumes that $v_f^{[i]}$ are given in advance, whereas the online version needs to predict $v_f^{[i]}$ and deals with prediction errors. The online integral algorithms are more practical and have lower control overhead. The offline fractional algorithms are also important because they provide a lower bound on cost, and further serve as a basis for designing our online algorithms.

4.2 Offline Fractional Flow Assignment

We begin by solving the offline fractional flow assignment problem. We first present an efficient algorithm to compute an optimal traffic assignment when ISPs do not have capacity constraints. We then extend the algorithm to deal with capacity constraints.

4.2.1 An Optimal Algorithm for Percentile-based Charging Without Capacity Constraints

A key to optimizing cost is to determine the charging volumes. For example, when ISPs use the 95th-percentile charging model, we need to determine the 95th-percentile traffic volume for each ISP. Once we know the charging volume for each ISP, we can assign traffic by ensuring that the number of intervals in which ISP k serves more than its charging volume of traffic does not exceed $(1 - q_k) * I$ (*e.g.*, $5\% * I$ for 95th-percentile charging).

Based on the above observation, we develop an efficient algorithm that computes an optimal traffic assignment in two steps: (i) compute the charging volume for each ISP, and (ii) assign traffic based on the charging volumes.

4.2.2 Computing Charging Volumes

In this section, we describe how to compute the optimal charging volumes to minimize total cost. We show that the charging volumes can be derived in two steps: (i) compute the sum of the charging volumes, namely $\sum_k p_k$, and (ii) compute individual p_k values based on the sum.

4.2.2.1 Computing the sum of charging volumes.

We first describe how to compute the sum of charging volumes to minimize cost. This is based on the following two important observations, which we will formally prove below. Our first observation is that the total cost has a monotonicity property with respect to the sum of the charging volumes. This monotonicity property suggests that to minimize the total cost $\sum_k c_k(p_k)$, we need to minimize the value of $\sum_k p_k$. Our second observation is that the minimum value of $\sum_k p_k$ is equal to $\mathtt{qt}(V, 1 - \sum_k (1 - q_k))$. As an example, suppose we have 4 ISPs and all of them charge based on the 95th-percentile volume; then the minimum $\sum_k p_k$ is equal to the 80th-percentile of the total traffic (since $1 - 4 * (1 - 95\%) = 0.80$). The two observations together suggest that to minimize cost, we need to have $\sum_k p_k = \mathtt{qt}(V, 1 - \sum_k (1 - q_k))$, which is easy to compute given V and q_k.

Now we formally prove the above two observations. Define $c_{min}(s) = \min\{\sum_k c_k(p_k) \mid \sum_k p_k = s\}$. We have

THEOREM 1. *If* $s_0 \ge s_1 \ge 0$, *then* $c_{min}(s_0) \ge c_{min}(s_1)$.

PROOF. Suppose the set p_k minimizes $\sum_k c_k(p_k)$ subject to $\sum_k p_k = s_0$. We have $c_{min}(s_0) = \sum_k c_k(p_k) \ge \sum_k c_k(p_k \cdot s_1/s_0) \ge c_{min}(\sum_k p_k \cdot \frac{s_1}{s_0}) = c_{min}(s_1)$, where the first inequality is because the cost functions c_k are monotonically non-decreasing, and the second inequality is by the definition of c_{min}. $\square$

The second observation, which is formalized in Theorem 2, establishes the (reachable) lower bound of $\sum_k p_k$.

THEOREM 2. $\sum_k p_k \ge \mathtt{qt}(V, 1 - \sum_k (1 - q_k)) \stackrel{\text{def}}{=} V_0$.

We need the following lemma to prove the above theorem. The proof of the lemma is in Appendix A.

LEMMA 3 (QUANTILE INEQUALITY). *Given* K *equal-length time series* $T_k = \{t_k^{[1]}, t_k^{[2]}, \cdots, t_k^{[n]}\}$, *where* $n = \lvert T_k \rvert$ *and* $0 \le a_k \le 1$, *we have*

$$\sum_k \mathtt{qt}(T_k, 1 - a_k) \ge \mathtt{qt}(\sum_k T_k, 1 - \sum_k a_k).$$

Given the above lemma, we can prove Theorem 2 as follows.

$$\begin{aligned} \sum_k p_k &= \sum_k \mathtt{qt}(T_k, 1 - (1 - q_k)) \\ &\ge \mathtt{qt}(\sum_k T_k, 1 - \sum_k (1 - q_k)) \\ &= \mathtt{qt}(V, 1 - \sum_k (1 - q_k)) \stackrel{\text{def}}{=} V_0. \end{aligned}$$

Note that in the above proof, we implicitly assume that all $q_k * I$ are integers, where $I = \lvert V \rvert$. When $q_k * I$ is not an integer, we can easily enforce its integrality by readjusting q_k to $\lceil q_k * I \rceil / I$. Clearly such readjustment does not affect the outcome of $\mathtt{qt}(V, q_k)$ (*i.e.*, $\mathtt{qt}(V, q_k) = \mathtt{qt}(V, \lceil q_k * I \rceil / I)$, where $I = \lvert V \rvert$). Throughout the rest of the paper, we assume that such readjustment has been made for every q_k in advance. For example, when we discuss 95th-percentile charging with charging period of one week (*i.e.*, $I = 7 \times 24 \times 60/5 = 2016$), we are really using $q_k = \lceil 0.95 * I \rceil / I = \lceil 1915.2 \rceil / 2016 = 1916/2016$ (as opposed to $q_k = 0.95 = 1915.2/2016$).

4.2.2.2 Computing individual charging volumes.

Once V_0 is determined, the next step is to compute the optimal charging volumes p_k, which minimize $\sum_k c_k(p_k)$ subject to $\sum_k p_k = V_0$.

Theorem 4 shows that the optimal charging volumes p_k are easy to derive when all c_k are concave (proof omitted for the interest of brevity).

THEOREM 4. *If all cost functions c_k are concave, then an optimal solution is one in which the charging volumes are 0 for all but one ISP. More specifically, let $k_0 = \text{argmin}_k[c_k(V_0) - c_k(0)]$. Define $p_k^* = V_0$ when $k = k_0$ and 0 otherwise. We have $\sum_k c_k(p_k^*) \leq \sum_k c_k(p_k)$ for any p_k satisfying $\sum_k p_k = V_0$.*

For general cost functions (*e.g.*, non-decreasing step functions), it is more involved to determine the optimal charging volumes p_k (which minimize $\sum_k c_k(p_k)$ subject to $\sum_k p_k = V_0$). Below we introduce a dynamic programming algorithm to solve this problem. Let $opt(v, k)$ be the optimal cost for serving traffic volume v by the first k ISPs. We have:

$$opt(v,k) = \begin{cases} c_1(v) & k = 1 \\ \min\limits_{0 \leq x \leq v} \{opt(v-x, k-1) + c_k(x)\} & k > 1. \end{cases}$$

We can start from $opt(v, 1)$ and compute $opt(v, k)$ based on the above recurrence relation, while keeping track of the corresponding allocations. The value of $opt(V_0, K)$ gives the optimal cost, and its corresponding allocation determines p_k. The time complexity of the algorithm is $O(K \cdot V_0^2)$; the space complexity is $O(K \cdot V_0)$. Note that the above algorithm assumes that the desired precision is one. In practice, this may not be necessary, since the cut points of the pricing curve are often very coarse-grained. It is easy to handle any desired precision through discretization. For example, if we want V_0 and p_k to be accurate up to 100, we only need to compute $opt(v, k)$ when v is a multiple of 100. This reduces both time and space complexity. More precisely, with precision P, the time and space complexity of the algorithm will be $O(K \cdot (V_0/P)^2)$ and $O(K \cdot V_0/P)$, respectively. In practice, we typically only need to handle $K \leq 10$ and $V_0/P \leq 1000$, so the complexity of the algorithm is quite low.

4.2.3 Traffic Assignment Given Charging Volumes

Given the charging volumes, namely p_k for ISP k, next we describe how to assign traffic during each time interval. The goal of traffic assignment is to ensure that p_k is the charging volume for ISP k; that is, for $q_k * I$ intervals, the traffic volumes assigned to ISP k are less than or equal to p_k, and ISP k is only allowed to serve more than p_k for the remaining $(1 - q_k) * I$ intervals. This can be achieved by dividing intervals into non-peak intervals and peak intervals.

According to the definition of V_0, during the intervals when total traffic volumes are no larger than V_0, all traffic can be assigned without having any ISP receiving traffic more than its charging volume. Therefore, we call these intervals *non-peak intervals*. For the remaining intervals, at least one ISP needs to serve traffic more than its charging volume. As a result, we call the latter intervals *peak intervals*. We will use this terminology throughout this paper.

Based on the above definitions of peak and non-peak intervals, we assign traffic in the following way. If an interval is a non-peak interval, we assign traffic such that the traffic assigned to ISP k is less than or equal to p_k. There are multiple ways to assign traffic to satisfy the above constraint, and all of these assignments give the same cost. Therefore, we can pick any one of them. In Section 5, we will take advantage of such flexibility to improve performance. For a peak interval, we randomly select an ISP k to burst (*i.e.*, its assigned traffic exceeds p_k). This is done by assigning each of the remaining ISPs its charging volume p_k, and then assigning all remaining traffic to the burst ISP. This is feasible because we assume that ISPs do not have capacity constraints. (We will study the case of limited capacity in the following section.)

Putting everything together, we have the algorithm shown in Figure 3 to minimize cost for splittable flows. It is easy to see that p_k is ensured as the charging volume for ISP k, since ISP k serves more than p_k for exactly $(1 - q_k) * I$ intervals. Since the sum of the achieved p_k is equal to V_0, according to Theorem 2, the algorithm achieves minimum cost.

```
find V_0
find p_k by minimizing Σ_k c_k(p_k) subject to Σ_k p_k = V_0
for each (1 − Σ_k(1 − q_k)) ∗ I non-peak interval
    traffic assigned to ISP k is less than or equal to p_k
for each Σ_k(1 − q_k) ∗ I peak interval
    pick ISP k that has bursted fewer than (1 − q_k) ∗ I intervals
    assign p_k + v^i − V_0 to ISP k
    assign p_k' to ISP k', where k ≠ k'
```

Figure 3: An offline optimal flow assignment algorithm for splittable flows under the percentile-based charging model and without capacity constraints.

4.2.4 Dealing with Capacity Constraints

The previous algorithm assumes that ISPs do not have capacity constraints (*i.e.*, they each can carry all traffic in an interval). This is a reasonable assumption as multihoming is often used to provide high reliability—even if all other ISPs fail, a user can still send out traffic using the single remaining ISP. However, it is still possible that a single ISP may not always have enough capacity to handle all of the traffic.

```
f = Σ_k(1 − q_k)   // initialize the fraction of peak intervals
assignable = false
while assignable is false
    V_0 = qt(V, 1 − f)
    find p_k by minimizing Σ_k c_k(p_k) subject to Σ_k p_k = V_0
    assignable = IsPeakAssignable(V, V_0, f, {p_k})
    reduce f by Δ if assignable is false
assign f ∗ I peak intervals such that
    each ISP k bursts in at most (1 − q_k) ∗ I intervals, and
    there is enough total capacity for each peak interval
```

Figure 4: The global fractional offline assignment (GFA-offline) algorithm: an algorithm for percentile-based charging with link capacity constraints. The cost function $c_k(x)$ is assumed to be ∞ if x exceeds the capacity of ISP k. The constant Δ controls the step size when we search for f, the largest assignable fraction of peak intervals ($\Delta = 0.01$ in our evaluations).

We use the algorithm in Figure 4 to accommodate such capacity constraints. The basic idea is to properly choose the fraction of peak intervals, denoted as f, so that there are multiple burst ISPs during each peak interval that together provide enough total capacity to carry all of the traffic. More formally, given f and the corresponding V_0 and p_k (computed inside the **while**-loop in Figure 4), we need to know $IsPeakAssignable(V, V_0, f, \{p_k\})$, *i.e.*, whether it is possible to assign different ISPs to burst in $f * I$ peak intervals so that (i) no ISP k bursts more than $(1 - q_k) * I$ intervals, and (ii) there is enough total capacity in each peak interval.

Let g denote a set of ISPs that when bursting together can carry traffic in any peak interval. A sufficient condition for g is $\sum_{k \in g} C_k + \sum_{k' \notin g} p_{k'} \geq maxLoad$, where C_k is the capacity of link k and $maxLoad$ is the maximum load of a charging period. Let t_g denote the number of intervals during which the ISPs in group g burst. Let G be the set of all (2^K) possible ISP groups. When the following conditions hold, there exists a peak interval assignment and $IsPeakAssignable(V, V_0, f, \{p_k\})$ returns true.

$$\max \sum_{g \in G} t_g \geq f * I$$

$$\sum_{g:\ k \in g} t_g \leq (1 - q_k) * I \text{ for all } k.$$

A few comments follow. First, K is usually small (*e.g.*, below 10), so the number of variables is manageable. Second, the above conditions are sufficient but not necessary, because the conditions ensure that we have an assignment even when the traffic load during a peak interval is always equal to the maximum load. However, since the load during a peak interval may be smaller than the maximum load (*e.g.*, the 95th-percentile load is smaller than the maximum load), it is possible to have a peak-load assignment even when the above conditions are not satisfied. When the difference between the maximum load and the smallest peak load is small, the conditions are tight. Third, all these constraints are linear constraints, so we can determine the existence of a peak load assignment by solving an integer programming problem. Since the number of intervals is large, in practice we first solve the problem without the integer constraints and then use rounding to derive the results.

We refer to this assignment algorithm as global fractional offline assignment (GFA-offline).

4.3 Online Integral Assignment Algorithms

The offline fractional assignment algorithms described in the previous sections assume that traffic patterns are known in advance and that flows are splittable. In practice, traffic patterns are not given *a priori*. Moreover, one may prefer not to split flows (to reduce control overhead, *e.g.*, when BGP is in use).[2] In this section, we present online integral assignment algorithms to address both issues. Our solution consists of two steps:

1. Predict the traffic and V_0 in the next interval.
2. Compute an integral assignment based on predicted traffic.

We will now describe each step in detail.

4.3.1 Predicting Traffic and V_0

First, as shown in Figure 5, we predict total and per-flow traffic using an exponentially weighted moving average (EWMA). That is, $Prediction = \beta * currTraffic + (1-\beta) * Prediction$. Note that $\beta = 1$ corresponds to predicting traffic using only the preceding interval. Our evaluation shows that the predictions with $\beta < 1$ and $\beta = 1$ yield very similar performance.

There are several technical details about traffic prediction worth mentioning. First, to avoid keeping history for too many flows, we periodically remove the flows with the smallest predicted traffic volumes. Second, when a flow appears for the first time, we will directly use its traffic volume in the current interval to predict its traffic in the next interval (since it does not have any other history yet). Third, since the predicted total traffic may not match the sum of the predicted traffic of the flows that we keep track of, we add a normalization step shown in the algorithm.

Besides the traffic, we also need to predict V_0 in order to decide whether the next interval is a peak interval. Clearly, if we underestimate V_0, then we may end up exhausting the quota of peak intervals too early, thus increasing the total cost due to increased charging volumes of individual ISPs. To avoid this penalty, we update V_0 in the following conservative way. We use the V_0 in the past charging period as an initial estimate of V_0. We also maintain a sliding window (with length equal to the charging period) and after each interval we compute the V_0 value for the most recent sliding window, denoted as V_0'. Whenever V_0' exceeds V_0, we increase V_0 to $\gamma \cdot V_0'$ and recompute all the charging volumes based on the new V_0. For our traces, with $\gamma = 1.05$ we are able to track increases in V_0 quickly without overshooting too much.

[2] Avoiding splitting may cause packet losses. Our evaluations show that these loss rates are very low.

```
// update traffic prediction at interval i using EWMA
PredictTraffic() {
  PredictedTotal = β * v^[i] + (1 − β) * PredictedTotal
  for each flow f appearing in interval i or in PredictedFlow
    if flow f does not appear in PredictedFlow
      PredictedFlow(f) = v_f^[i]
    else
      PredictedFlow(f) = β * v_f^[i] + (1 − β) * PredictedFlow(f)
  if PredictedFlow has more than 2*MAX_FLOW_NUM flows
      keep only the MAX_FLOW_NUM largest flows
  normalize traffic in PredictedFlow such that
      Σ_f PredictedFlow(f) = PredictedTotal
}
```

Figure 5: The `PredictTraffic()` subroutine: predicting total and per-flow traffic volumes.

When recomputing the charging volumes, we need to ensure that for every k the new charging volume p_k' is no less than the old charging volume p_k. Otherwise, with $p_k' < p_k$, there may be many (possibly more than $(1-q_k)I$) past intervals in which we assign more than p_k' (but less than p_k) amount of traffic to ISP k, thus making it difficult to ensure $\mathtt{qt}(T_k, q_k) = p_k'$. We can apply the same dynamic programming algorithm as in Section 4.2.2.2 to compute $\{p_k'\}$; the lower bounds $\{p_k\}$ can be easily enforced by setting $c_k(x) = \infty$ for all $x < p_k$.

4.3.2 Performing Offline Integral Assignment

We first note that even in an offline setting with perfect knowledge of traffic, the integral assignment problem is already hard. More specifically, we have the following negative result (please see Appendix B for its proof):

THEOREM 5. *There is no polynomial-time algorithm that can achieve a constant approximation ratio for integral assignment with general cost functions, unless P=NP.*

The above negative result makes it very natural to consider approximation algorithms. We propose the following (offline) greedy algorithm for integral assignment. As shown in Figure 6, we first run the offline fractional flow assignment algorithm to find the charging volumes p_k. Based on p_k for ISP k, we then compute the targeted amount of traffic to be assigned to it; we call this value its *pseudo capacity* during the interval (abbreviated as $PseudoCap$). For a non-peak interval or a peak interval in which ISP k is not a burst ISP, the pseudo capacity of ISP k is its charging volume computed by the fractional assignment algorithm; for a peak interval in which ISP k is a burst ISP, its pseudo capacity is its link capacity C_k. Our goal is to ensure that the traffic assigned to any ISP does not exceed its pseudo capacity.

Conceptually, this is a problem similar to bin packing, and can therefore be solved using a greedy heuristic. Specifically, we can initialize each ISP with its pseudo capacity, sort the flows in descending order of the traffic volumes they generate, and then iteratively assign the flows to the ISP with the largest remaining pseudo capacity. The actual algorithm in Figure 6 splits this conceptual greedy assignment process into two separate steps. It first tries to assign traffic using p_k as the bin size, and then refills the bin size by $(PseudoCap_k - p_k)$ and assigns the remaining traffic. We find that using such a two-step approach makes it more likely for there to be ISP with large remaining bin size. This ISP can then be used in an online setting to accommodate traffic for prefixes not seen before.

4.3.3 Accommodating Prediction Errors

The integral assignment algorithm presented in Figure 6 works well for offline traffic demands. However, in an online setting, the

```
OfflineIntegral(NumPeaks, V_0, {p_k}, TotalTraffic, FlowTraffic) {
    // compute pseudo capacities
    for each ISP k, PseudoCap_k = p_k
    if TotalTraffic >= V_0 and NumPeaks < Σ_k (1 - q_k) · I
      for each burst ISP k, PseudoCap_k = C_k
      NumPeaks = NumPeaks + 1

    // try to assign min(V_0, TotalTraffic) amount of traffic
    for each k, Bin_k = p_k
    SortedFlowList = sort flows in descending order of FlowTraffic
    for each flow f in SortedFlowList
        find ISP k with the largest bin size Bin_k
        if Bin_k >= FlowTraffic(f)
          Assignment(f) = k
          Bin_k = Bin_k - FlowTraffic(f)

    // assign the remaining traffic
    for each k, Bin_k = Bin_k + PseudoCap_k - p_k
    RemainingFlowList = SortedFlowList - flows in Assignment
    for each flow f in RemainingFlowList
        Assignment(f) = k, where ISP k has the largest Bin_k
        Bin_k = Bin_k - FlowTraffic(f)

    // return the result
    MaxISP = argmax_k Bin_k
    return Assignment, MaxISP
}
```

Figure 6: The `OfflineIntegral()` subroutine: an offline greedy integral flow assignment algorithm.

predicted traffic may not match the real traffic due to prediction errors. If we are too greedy in filling up pseudo capacities of the links, then the prediction error may cause the actual usage to exceed the target pseudo capacities, thereby significantly increasing the actual cost. Our solution is to add some margin when computing the charging volumes and then trim them down afterward; our adjustment algorithm is shown in Figure 7. We find that setting $margin = 0.05 * V_0$ works well for the traces we have.

```
{p_k} = OfflineFractional(V_0 + margin * K)
for each ISP k, p_k = max{0, p_k - margin}
```

Figure 7: Dealing with prediction errors by adjusting p_k.

4.3.4 Final Algorithm

Putting everything together, we have the final online algorithm shown in Figure 8. This algorithm is also referred to as global integral online assignment (GIA-online).

```
// compute assignment for current interval
for each interval i
    // update V_0 based on most recent I intervals
    V_0' = FindV_0(TotalTraffic[i - I..i - 1], NumPeaks)
    if V_0' > V_0
      V_0 = 1.05 · V_0'
      margin = 0.05 * V_0
      {p_k'} = OfflineFractional(V_0 + margin * K, {p_k})
      for each k
          p_k = max(0, p_k' - margin)

    // perform integral assignment using predicted traffic
    (Assignment, MaxISP) =
          OfflineIntegral( NumPeaks, V_0, {p_k},
                           PredictedTotal, PredictedFlow)

    // actual assignment:
    for every flow appearing in interval i
        if flow appears in PredictedFlow
          use pre-computed Assignment
        else // this is a flow not seen before
            assign to MaxISP
    PredictTraffic()
```

Figure 8: The global integral online flow assignment (GIA-online) algorithm.

5. OPTIMIZING PERFORMANCE UNDER COST CONSTRAINT

In the preceding section we studied how to optimize cost for a user. To be practical, a sensible smart routing algorithm needs to consider both cost and performance.

5.1 Problem Formulation and Overview

There are multiple ways to formulate the problem of optimizing both performance and cost. For example, one possibility is to design a metric that is a combination of both cost and performance. However, it may be unclear to users exactly how to determine the relative weights between cost and performance. A more intuitive approach, which we propose in this paper, is to optimize performance under a given cost constraint.

As before, we design both offline and online algorithms. Both algorithms consist of two key components. The first component is a building block of the second one.

1. Given the pseudo capacity of each ISP during each interval, namely an upper bound on the traffic that can be assigned to an ISP, we assign flows in such a way that the total delay is minimized. We call this component *Flow Assignment Given Pseudo Capacities*.

2. Since a given cost constraint allows multiple pseudo capacity assignments and these different assignments give different delays, we will need to select the assignment that yields good performance. We call this component *Pseudo Capacity Selection*.

5.2 Offline Traffic Assignment

We first present an offline algorithm.

5.2.1 Flow Assignment Given Pseudo Capacities

The goal of flow assignment given pseudo capacities is to minimize the total latency such that the traffic assigned to each ISP does not exceed its pseudo capacity.

We solve the flow assignment problem as a minimum-cost multicommodity flow (MCMCF) problem by constructing a graph as shown in Figure 9. In the graph, each node in the top row represents the source of a flow and the destination of the flow is in the bottom row. The nodes in the middle two rows are ISP nodes. The cost $perf(k, f)$ of the link from the source node of flow f to ISP node k on the next row is the latency incurred by assigning flow f to ISP k; the costs of other links are zero. The link capacity of each ISP node on the second row to the corresponding ISP node on the third row is the pseudo capacity of the ISP; the capacities of other links are unlimited.

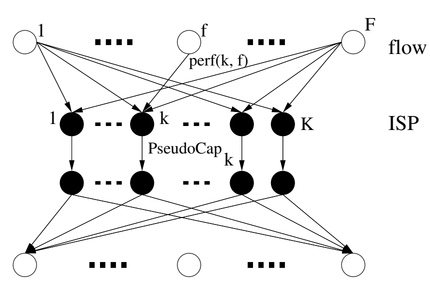


Figure 9: MCMCF formulation of the flow assignment problem.

5.2.2 Pseudo Capacity Selection

Given pseudo capacities, the above algorithm computes flow assignment to optimize latency. Next we study how to determine the pseudo capacities of the links during each interval.

Pseudo capacities are determined by cost constraints. Without consideration of cost, each ISP can allocate traffic up to its link capacity, *i.e.*, a link's pseudo capacity is its raw capacity. However, since our goal is to optimize performance under cost constraints, we apply the algorithms described in Section 4, which impose constraints on how much traffic each link can carry. More specifically, we obtain the charging volume p_k for ISP k based on cost optimization. Then, during a non-peak interval, each ISP's traffic should be no higher than p_k (*i.e.*, the pseudo capacity of ISP k is p_k).

The pseudo capacities of peak intervals are not completely determined by cost optimization. The only constraint from cost optimization is that each ISP can exceed p_k for only $(1 - q_k) * I$ intervals, so we still have flexibility in picking the burst ISPs for each individual peak interval. Below we describe the algorithms to determine the pseudo capacities for peak intervals under the cost constraints.

A key step in determining the pseudo capacities of a peak interval is to decide which ISP or set of ISPs to burst. Selecting burst ISPs for a given peak interval can be done in two steps. First, we derive the best performance achieved by bursting any set of ISPs at a given peak interval. This step can be achieved by first setting the pseudo capacities of the chosen burst ISPs to their link capacities, the pseudo capacities of the remaining links to their charging volumes, and then calling the algorithm developed in Section 5.2.1.

Next, we optimize performance across the entire charging period while preserving the cost constraint (*i.e.*, each ISP can burst up to $q_k * I$ time intervals). Let $BestPerf(g, i)$ denote the best performance computed by the algorithm in Section 5.2.1, when ISP set g bursts at interval i. Then the step of determining which ISPs to burst at each peak interval can be cast into a mixed integer programming (MIP) problem as shown in Figure 10. The MIP can be solved using LP software such as lp_solve [14].

$$\begin{aligned}
\text{minimize} \quad & \sum_{g,i} BestPerf(g,i) * IsPeak(g,i) \\
\text{subject to} \quad & \sum_{g:\ k \in g, i} IsPeak(g,i) \le (1 - q_k) * I \ \ \forall \text{k} \\
& \sum_{g} IsPeak(g,i) \ge 1 \text{ for any peak interval } i \\
& IsPeak(g,i) \in \{0,1\}
\end{aligned}$$

Figure 10: MIP formulation to determine which ISPs to burst.

5.3 Online Algorithms

Next we present the online algorithms. There are two new problems that we need to address in designing an online algorithm. First, we need to predict both traffic and performance. Second, we need an efficient algorithm to select pseudo capacities and assign flows to ISPs.

5.3.1 Predicting Traffic and Performance

We predict traffic patterns in the same way as shown in Figure 8. To predict performance, we again use the exponentially weighted moving average.

5.3.2 Performing Traffic Assignment

We use the following greedy heuristic to assign traffic online. During a time interval i, a flow is assigned to the ISP that has the best predicted performance among all of the ISPs with sufficient pseudo capacities. We observe that the ordering of flow assignment affects the performance. In particular, we find that assigning flows in order of descending $DiffPerf(f) * v_f^{[i]}$ performs very well, where $DiffPerf(f)$ is the predicted performance difference between the best performing ISP and the worst performing ISP, and $v_f^{[i]}$ is flow f's volume during time interval i. Similarly to Figure 6, we split the greedy assignment process into two separate steps so that we can better accommodate traffic that has not appeared before.

6. EVALUATIONS

In this section, we evaluate the performance of the algorithms developed in the preceding sections. We obtain two sets of traffic traces: Abilene traces and a large Web server trace. The Abilene traces contain netflow data from a number of universities and enterprises on the Internet-2 from Oct. 8, 2003 to Jan. 6, 2004. We select traffic traces from the organizations, shown in Table 2, for our evaluations. To speed up our evaluations, during each 5-minute interval, we only use the 2000 destination prefixes with the largest volumes. We call these prefixes *top prefixes*. Note that in different time intervals, the sets of top prefixes are different, but they always account for over 90% of the total traffic in an interval.

AS	Organization	Traffic Rate (Mbps)
3582	University of Oregon	215.576 (202.527)
3	MIT Gateways	64.598 (64.587)
52	UCLA	52.245 (52.234)
59	Univ. of Wisconsin, Madison	33.333 (33.253)
237	NSF (MERIT-AS-14)	117.366 (108.621)
6629	NOAA Silver Springs Lab	62.340 (62.335)
70	National Library of Medicine	72.810 (72.691)
1701	NASA/GSFC (Goddard Space Flight Center)	37.451 (37.448)
22753	Red Hat Inc.	33.241 (33.238)
Anonymized	Commercial Web Server	156.231 (64.124)

Table 2: Traffic traces used in our evaluation, where the last column shows the original traffic rates averaged over 91 days, and the traffic rates after filtering, which are shown in parentheses.

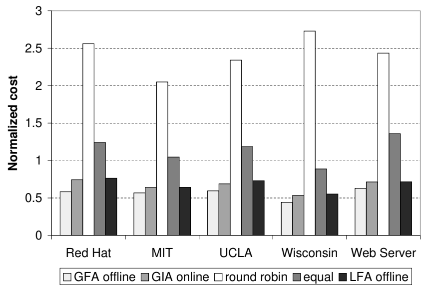
For diversity, we also use the trace collected from a large commercial Web server from Oct. 1, 2003 to Dec. 31, 2003. This is one of the busiest Web sites. The trace contains IP addresses of hosts that issue Web requests, along with time-stamps and sizes of requested files. Note that for efficiency, a set of proxy caches are deployed in front of the Web server. About half of the requests seen at the Web server are re-directed from the proxies with the IP addresses replaced by the proxies' IP addresses. Since we are interested only in wide-area network traffic, we filter out the re-directed requests. In addition, as with Abilene traces, we only consider the traffic contributed by the top 2000 prefixes during each 5-minute interval. The last column in Table 2 shows the mean traffic volume before and after filtering. Note that the difference between filtered traffic and original traffic of the Web server is larger than that of the Abilene traces due to the filtering of the redirected requests.

6.1 Evaluation of Cost Optimization

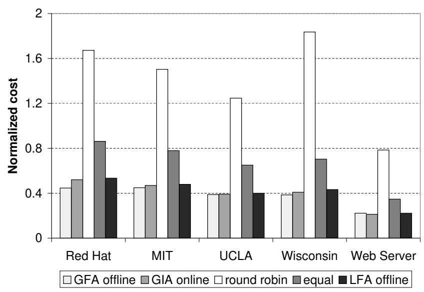
We compare the performance of our cost optimization algorithms described in Section 4 (*i.e.*, GFA-online in Figure 4 and GIA-online in Figure 8), with the following alternatives:

- Round robin: in each time interval, traffic is assigned to a single ISP, and we rotate the responsible ISP in a round robin fashion. If the chosen ISP does not have enough capacity to carry all of the traffic, the remaining traffic is assigned to the other ISPs in the same round robin manner.

- Equal split: in each time interval, traffic is split equally among all ISPs. When there are capacity constraints, we first sort links in order of ascending capacity. In this order, we assign ISP k an amount of traffic which is the minimum of C_k and rem_traf/rem_nisps, where rem_traf is the amount of traffic that remains to be assigned, and rem_nisps is the number of ISPs that have not yet been assigned traffic.



(a) Charging period = 1 week



(b) Charging period = 1 month

Figure 11: Comparison of the total cost across different traces, where each user has 4 links to the Internet, and each link's cost is determined by a simple pricing function.

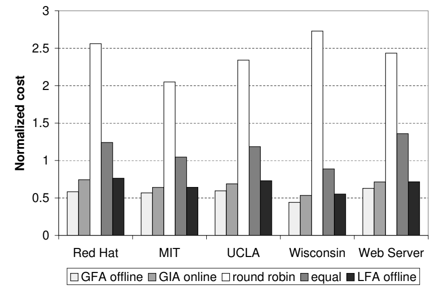
- Local fractional offline (LFA offline): in each interval i we determine the traffic allocation $t_k^{[i]}$ such that $\sum_k c_k(t_k^{[i]})$ is minimized. This essentially minimizes the total cost assuming that the cost is a function of the traffic in the current interval (instead of based on q_k percentile traffic volume). To determine the optimal allocation, we apply the dynamic programming algorithm described in Section 4.2, which takes the total traffic in the current interval as input to derive an allocation that leads to the minimum cost.

- Dedicated links: in today's market there is an option to purchase dedicated links besides burstable links. Dedicated links have flat rates which are independent of usage, even when the assigned traffic is 0.

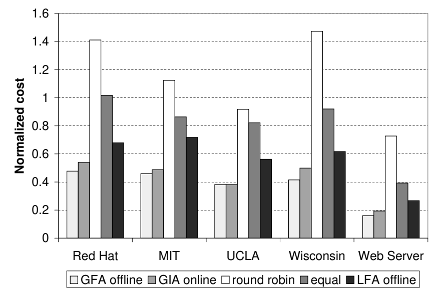
We derive the cost of a burstable link based on the 95th-percentile charging model throughout our evaluation. For a given trace, we determine the cost of using dedicated links by finding the cheapest set of links whose total capacity can accommodate the maximal load in the current charging period.

We start our evaluation by considering simple pricing functions: the price of an ISP link is a constant value if the charging volume is greater than 0, and the value is one of the entries shown in Table 1. In our first experiment, we consider a user with 4 links connected to the Internet. We randomly pick 4 burstable links from the ten links shown in Table 1 with the corresponding capacity constraints. We allow duplicates, since it is possible to have multiple links of the same type. Figure 11 shows the normalized cost achieved using different traffic assignment algorithms across 5 sets of traces. Here normalized cost is defined as the ratio between the cost of a burstable link based on a specific traffic assignment algorithm and that of using the dedicated links. Note that except for GIA-online, all of the algorithms are offline algorithms; thus they know traffic patterns in advance.

We make the following observations. First, as expected, GFA-offline yields the best performance. GIA-online incurs a moderately higher cost than its offline version due to prediction errors.



(a) Charging period = 1 week



(b) Charging period = 1 month

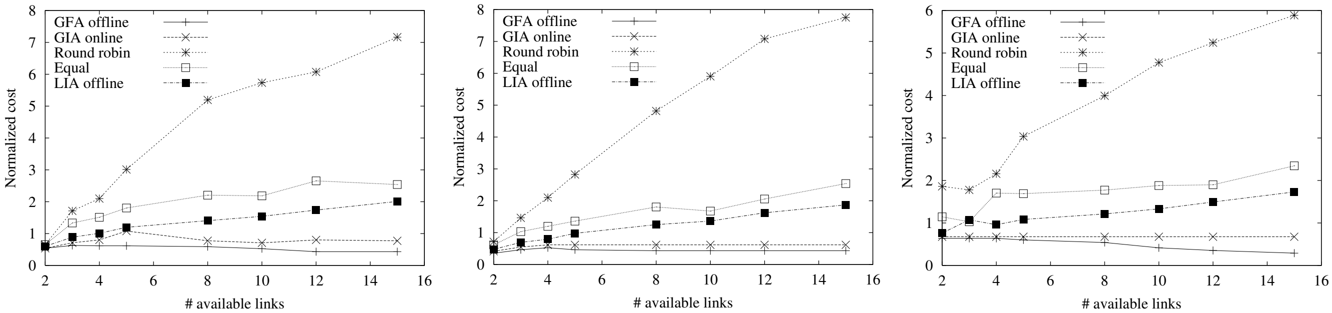
Figure 12: Comparison of the total cost across different traces, where each user has 4 links to the Internet, and each link's cost is a piece-wise linear function of traffic volume as shown in Figure 2.

Nevertheless, it is still able to yield cost comparable to (and often slightly lower than) LFA-offline, and much lower cost than the round robin and equal split. Second, we observe that applying GFA-offline, GIA-online or LFA-offline to burstable links can result in lower cost than using dedicated links. On the other hand, applying round robin or equal split to burstable links can incur significantly higher cost than using dedicated links. Finally, we observe that the relative ranking among these algorithms remains the same as the charging period changes from one week to one month.

We next use the more complex pricing functions, described in Section 3, to evaluate the robustness of our algorithms to varying pricing functions. Figure 12 summarizes the results. We observe that GFA-offline continues to perform the best. Its online version performs slightly worse due to prediction errors, but still out-performs the other algorithms.

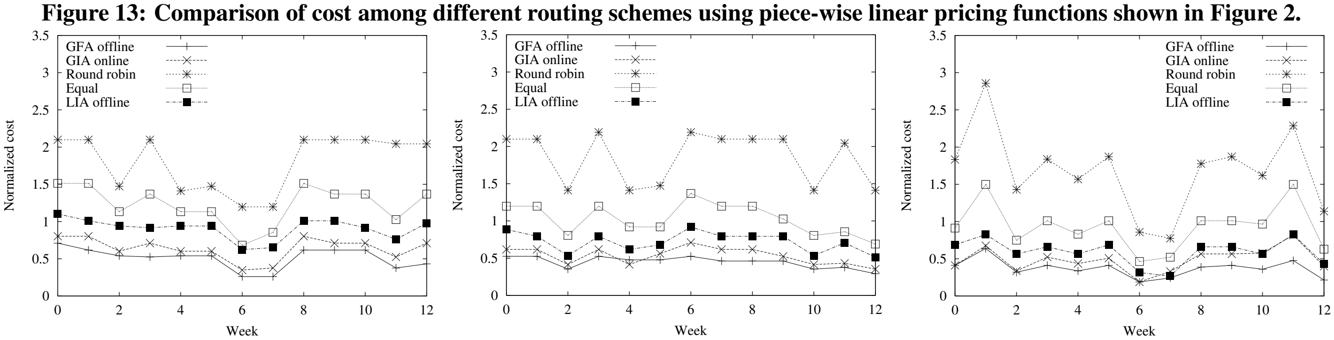
Next, we study the impact of varying the number of available links. Figure 13 shows the cost as we vary the number of links from 2 to 15. As before, GFA-offline yields the best performance, with GIA-online closely following it. We observe that the normalized cost of GFA-offline and GIA-online tend to decrease with the number of available links. This is because they can burst ISPs at their full capacities during peak load without incurring additional cost. In comparison, we observe that the normalized costs of the round robin, equal split, and LIA-offline algorithms increase with the number of links.

Finally, we look at the dynamics of cost over time. Figure 14 plots how cost varies over a period of 13 weeks, where the charging period is one week. As shown, GFA-offline and GIA-online perform significantly better than the other three algorithms. Since the normalized costs of GFA-offline and GIA-online are often much lower than 1, the cost of using these algorithms on burstable links is significantly lower compared with using dedicated links. We also observe that GFA-online can sometimes outperform GFA-offline, *e.g.*, week 4 in Figure 14 (b). This is because GFA-offline is not guaranteed to be optimal when there are capacity constraints.



(a) Red Hat Inc. (b) University of Wisconsin, Madison (c) Web server

Figure 13: Comparison of cost among different routing schemes using piece-wise linear pricing functions shown in Figure 2.



(a) Red Hat Inc. (b) University of Wisconsin, Madison (c) Web server

Figure 14: Time series plots of cost across different traces, where each user has 4 links to the Internet, and each link's cost is a piece-wise linear function of its charging volume shown in Figure 2.

Summary: Our evaluation results show that the GFA-offline algorithm achieves the lowest cost, as we expect. Moreover, its online version is also competitive despite fluctuations in traffic — it is often able to out-perform the other alternatives by a significant amount.

6.2 Evaluation of Performance Optimization Under Cost Constraints

Next we evaluate latency optimization under cost constraints. In this section, we mainly focus on evaluating our online algorithm in the presence of realistic RTT variations in the Internet. In the next section, we will further examine the performance of smart routing when multiple users interact with each other.

To evaluate the performance benefits of smart routing for a given traffic demand trace, we would ideally use round-trip time (RTT) measurements between the sources and destinations in the traffic traces during the period of trace collection. Due to a lack of such measurement data, we use the measurements published by NLANR [16] for our evaluation. The NLANR traces consist of RTT measurements between pairs of 140 universities from Oct. 2003 to Jan. 2004. In order to get the delay for a flow in the traffic trace, we first construct virtual ISPs in the following way. We map ISPs from the Rocketfuel dataset [22] to a set of universities by assigning each of their nodes to the geographically closest university in the NLANR trace. In addition, we map the origin of each of our Abilene traces to the closest university in the NLANR trace. Using a database from CAIDA's NetGeo project, we obtain physical coordinates for each destination prefix in our Abilene traces. We map each prefix to the closest node of each ISP. The delay through a given ISP from origin to prefix is then assigned to be the RTT between the universities in the NLANR trace representing the origin and the node of the ISP assigned to the prefix. We also add a last hop delay based on the speed of light and the distance between a prefix and its ISP node. In this way, we obtain delay traces reflecting realistic Internet RTT variations and geographically correlated performance variations across ISPs.

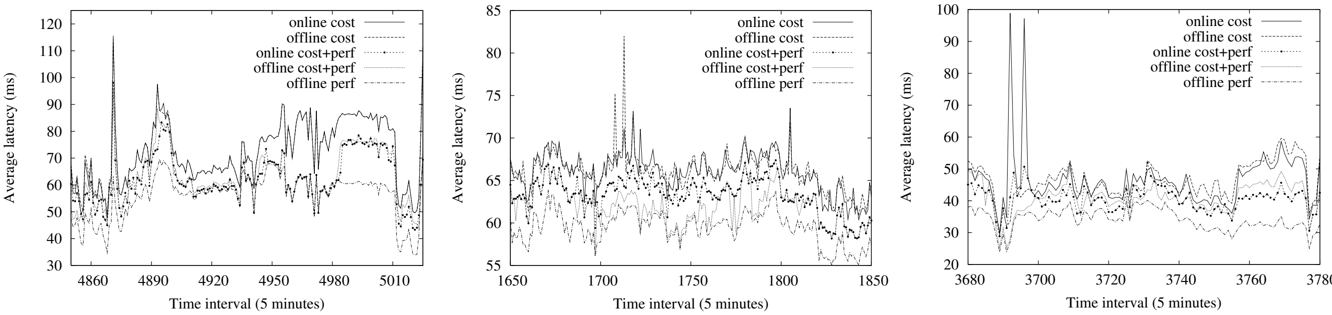
Note that the delay traces from NLANR are mostly between hosts within the US, so we filter out traffic with destination prefixes that are outside the US. Such filtering reduces traffic by 20% - 60%, and increases traffic variability (due to smaller aggregation). This increased variability will further stress-test our online algorithms.

Figure 15 compares the cost and performance of different routing schemes, where the cost in Figure 15 (a) is normalized by the cost of the offline cost optimization scheme. We make the following observations.

First, comparing the three offline schemes, we observe that optimizing performance alone increases cost by up to a factor of 2.75 compared with optimizing cost alone, whereas optimizing cost alone increases latency by up to 33% compared with the performance optimal. In contrast, the offline cost-performance scheme achieves the best of both worlds: it yields low cost and low latency.

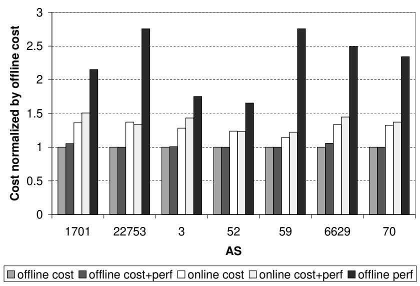
Second, comparing the offline schemes with their corresponding online versions, we observe that the online versions incur higher cost due to prediction errors. Note that the cost differences between the offline and online versions are larger than those in the previous sections, because here we filter out a significant amount of non-US traffic and thus increase variability. Nevertheless, the online cost-performance optimization yields much lower cost than optimizing performance alone, while achieving similar latency (within 10-20%).

Figure 16 further compares the latency of different schemes using time series plots. As it shows, in most cases the latency achieved using the online cost-performance scheme follows that of the offline performance optimization scheme. This suggests that the online cost-performance algorithm can effectively track variations in latency and traffic volume. Sometimes its latency is noticeably higher than pure performance optimization. This is due to the cost constraints, and indicates that there is a trade-off between optimizing cost versus optimizing performance. But the performance difference between the two is usually small (below 10%). When compared with optimizing cost alone, the online cost-performance algorithm often avoids delay spikes that pure cost-optimization can produce.

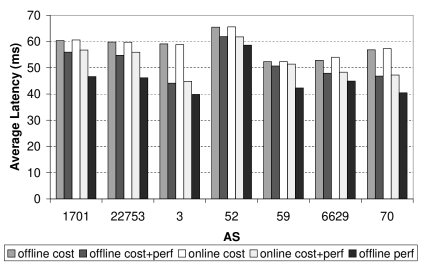


(a) Red Hat Inc. (AS 22753) (b) UCLA (AS 52) (c) National Library of Medicine (AS 70)

Figure 16: Performance comparison of different routing schemes.



(a) Comparison of total cost



(b) Comparison of performance

Figure 15: Comparisons of the total cost and performance across different traces during four weeks, where each user has 4 links to the Internet, and each link's cost is a piece-wise linear pricing function shown in Figure 2 based on a one-week charging period.

7. GLOBAL EFFECTS OF SMART ROUTING

In the preceding sections, we have investigated how an individual user can use smart routing to optimize cost and performance. Such optimization is selfish, since an individual user tries to optimize its own metrics without considering its impacts on other traffic. Moreover, the traffic of an individual user may self-interfere if traffic assignment may change link latency. Therefore, a comprehensive evaluation on the global effects of smart routing should address the following issues: (i) how well the smart routing algorithms perform when traffic assignment can affect link latency; (ii) how well different smart routing users co-exist; and (iii) how well smart routing users co-exist with single-homed users whose routing is controlled by the network. Below we investigate these issues, focusing on the performance at traffic equilibria.

7.1 Evaluation Methodology

Our topology is constructed using the Rocketfuel inter-domain topology data [22]. To make our simulations scalable, we select 4 ASes (to simulate ISPs) in the United States from the Rocketfuel data to construct a network topology of over 170 nodes and 600 edges. For each intra-domain link, we use the inferred OSPF weight and propagation delay from the data; for each peering link, we use the estimated propagation delay from the data. Once a user selects an ISP, its inter-domain route is determined based on the shortest AS hop count, and its intra-domain route follows the shortest OSPF path. Since the Rocketfuel data do not contain link bandwidth, we set the peering links to be OC3 (155 Mbps) and intra-domain links to be OC12 (622 Mbps). We use the M/M/1 latency function (i.e., $l(x) = \frac{1}{\mu - x} + prop$, where $l(x)$ is the latency for traffic load of x, μ is the link capacity, and $prop$ is the propagation delay) for all links in the network to capture the effect of traffic load on link latency.

We evaluate smart routing by connecting users to a varying number of ASes in the topology. For each smart routing user, its first-hop nodes in different ASes are geographically co-located. Further, we create traffic demands for each user using one of the nine Abilene traces described in Section 6. During each time interval, we select the top 100 destination prefixes in the traces, which account for over 90% traffic, and randomly map them to nodes in the simulation topology. The user sends traffic to the destination nodes at the traffic rate specified by the trace.

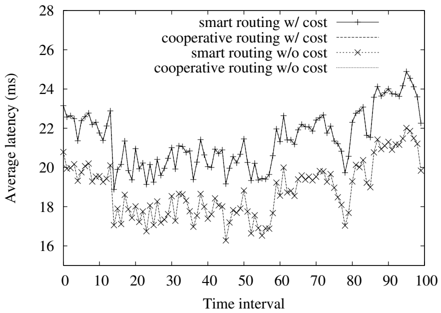
During every 5-minute time interval, we derive the latency under different routing schemes by computing the traffic equilibria based on the current topology and traffic demands using the approach in [20].

7.2 Smart Routing with Self-Interference

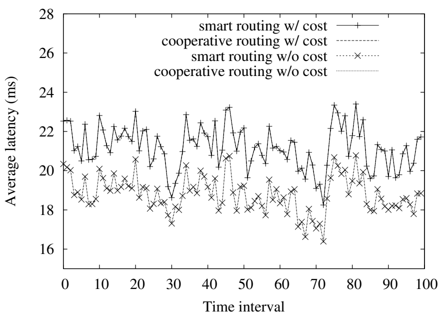
We start with an evaluation on the effects of self-interference. The online smart routing algorithm described in Section 5 assumes that traffic assignment would not affect link latency. If this is not the case (*i.e.*, the latency of a link depends on traffic assignment), the above algorithm results in selfish routing, since each flow is routed without considering its effects on other flows. In contrast, to optimize the total latency of all flows, a smart routing algorithm ideally needs to explicitly take into account this self-interference, and route traffic cooperatively to minimize the overall latency. As shown in [13, 23], the theoretical worst case performance difference between cooperative routing and selfish routing at traffic equilibria[3] can be quite large. Below we quantify the difference through simulations, and show that the impact of self-interference is small.

In our evaluations, the smart routing user has 4 ISPs and the burstable links to the ISPs are randomly selected from Table 1; the topology and real traffic traces are described in Section 7.1. In the interest of clarity, throughout this section we plot the results for only a small number of time intervals. The results for other time periods are consistent.

[3] A traffic equilibrium is defined as a state in which no traffic can improve its latency by unilaterally changing its link assignment. We adopt the approach in [20] to compute traffic equilibria.



(a) Red Hat Inc.



(b) University of Wisconsin, Madison

Figure 17: Evaluation of the effects of self-interference.

Figure 17 compares the latency of optimal routing versus that of smart routing at traffic equilibria, with and without enforcing the cost constraints. When there are no cost constraints, link capacities are always equal to their raw bandwidth; and when there are cost constraints, link capacities are equal to their pseudo capacities during the interval. We observe that under the same cost constraint, smart routing and optimal routing achieve similar latency. This suggests that ignoring self-interference incurs little performance penalty. In addition, removing the cost constraint yields slightly lower latency. This is consistent with the results in Section 6, which show that there is a trade-off between optimizing cost versus optimizing performance, but the trade-off is usually small.

7.3 Evaluation of Smart Routing in a Global Setting

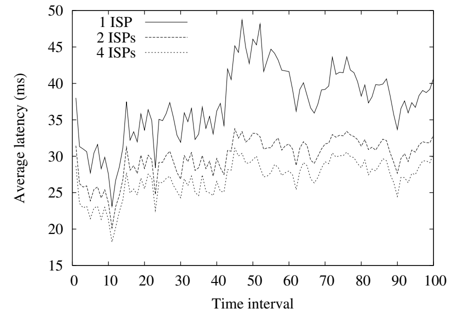
Having established the robustness of our smart routing algorithms against self-interference, we next evaluate smart routing when there are multiple users.

7.3.1 Performance Benefits of Smart Routing

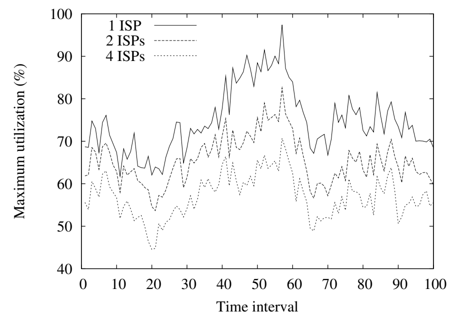
We start by studying the performance benefits of smart routing in the presence of other traffic. In our first experiment, we have 3 users generating traffic, where user 1 is a smart routing user subscribing to a varying number of ISPs, and users 2 and 3 are both single-homed. We observe that user 1 improves its performance by 10% when it changes from using one ISP to two ISPs, and further improves its performance by 8% when it uses four ISPs.

In our second experiment, we scale up the traffic by a factor of 3 to examine how smart routing performs in a highly utilized network. Figure 18 (a) shows the latency of the smart routing user over 100 time intervals. We observe that when user 1 changes from using one ISP to two ISPs, its performance is improved by 19%; when the number of ISPs increases to four, a further improvement of 9% is achieved. Smart routing achieves higher performance benefits under higher load, since it is able to route around network congestion whereas single-homed traffic follows a fixed path.

In addition, as shown in Figure 18 (b), increasing the number of ISPs also helps to reduce maximum link utilization. In particular, we observe a 12% reduction when user 1 changes from subscribing to one ISP to two ISPs, and an additional 10% reduction when the user subscribes to four ISPs. Similar results are observed when we use other traffic traces or vary the user's first-hop nodes.



(a) User latency



(b) Maximum link utilization

Figure 18: Performance benefit of smart routing when the latency is a function of traffic load.

7.3.2 Interactions Among Multiple Smart Routing Users

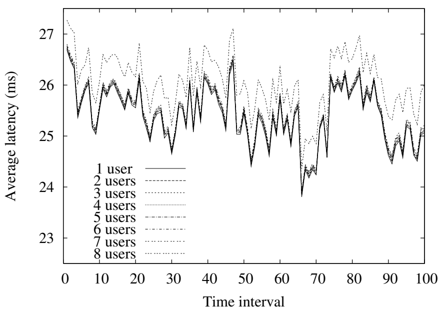
Next we examine how traffic from different smart routing users interacts. In our experiment, we start with a single smart routing user (called user 1) subscribing to two ISPs. We incrementally add new smart routing users (each subscribing to two ISPs) to the network to examine the interactions of smart routing traffic. We scale up the traffic by a factor of 3 to examine how smart routing performs in a highly utilized network. In the interest of clarity, we plot only the performance of user 1 in Figure 19 (a). The results for other users are consistent. As we can see, the performance degradation of user 1 remains less than 2 ms as the number of competing smart routing users increases. These results suggest that smart routing users can co-exist well.

Next, we repeat the above experiment, where each smart routing user subscribes to all four ISPs. Again, in the interest of clarity, we plot only the performance of user 1 in Figure 19 (b). We observe that an increase in the number of competing smart routing users has little effect on the performance of user 1. Moreover, user 1 improves its performance by about 8% when the number of its ISPs increases from two to four. The other users see a similar level of improvement (5–10%) when subscribing to two additional ISPs. This result is consistent with our findings in the previous subsection.

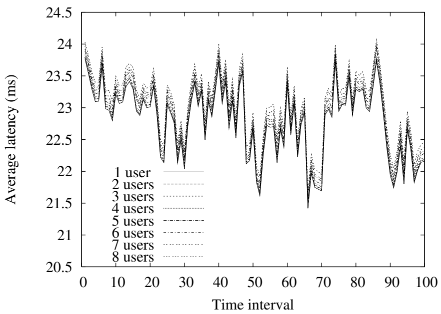
7.3.3 Interactions between Smart Routing Users and Single-homed Users

Finally, we study the interactions between smart routing users and single-homed users through the following two experiments.

In our first experiment, we start with two single-homed users. We examine the effects of adding one more ISP subscription to user 1. Figure 20 summarizes the results. As shown in Figure 20 (a), the performance of user 1 improves with the additional ISP, while the performance of user 2 remains almost the same after user 1 subscribes to one more ISP. This result indicates that a multi-

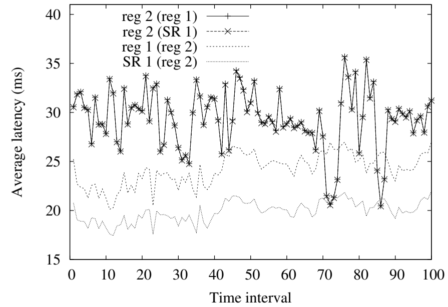


(a) Latency of user 1 (all users subscribe to 2 ISPs)

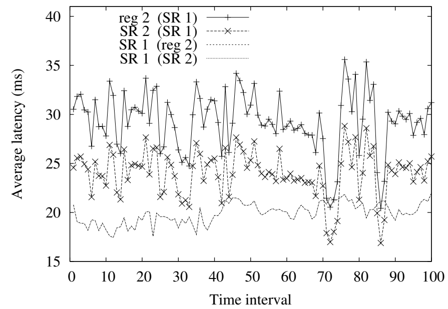


(b) Latency of user 1 (all users subscribe to 4 ISPs)

Figure 19: Latency of user 1 when it interact with multiple smart routing users, where all the users subscribe to 2 ISPs in (a), and all the users subscribe to 4 ISPs in (b).



(a) User 1 becomes multihomed while user 2 remains single-homed



(b) User 2 becomes multihomed while user 1 remains multihomed

Figure 20: Interaction of smart routing traffic and background traffic. Here `reg` i `(SR` j`)` denotes the performance of a regular (or single-homed) user i when user j uses smart routing.

homed user can improve its performance without adversely affecting a single-homed user. Note that the average latency of user 2 is lower than that of user 1 (with and without smart routing) in some intervals, although user 1 outperforms user 2 most of the time.

Next, we add one more ISP to user 2 as well. Figure 20 (b) shows that the latency of user 2 decreases without affecting user 1. In addition, we observe that smart routing users can take advantage of additional connections to smooth out traffic and reduce maximum link utilization by up to 10%.

In our second experiment, we have 5 users, which include one user that switches from single-homed to multihomed (referred to as user 1), two smart routing users, and two single-homed users. We compare the performance of all users before and after user 1 becomes multihomed. Our results show that user 1 improves its performance by 18%, whereas the latency of the other users changes within 1%.

Summary: Our evaluation results based on realistic settings show that the effect of self-interference is very small under traffic equilibria. In addition, we show that smart routing improves performance by 10–20%. Moreover, smart routing users co-exist well with other smart routing users and single-homed users.

8. CONCLUSIONS

In this paper, we design a series of novel smart routing algorithms to optimize cost and performance for multihomed users. Using both analysis and extensive simulations based on realistic traces, we show that our algorithms are very effective in minimizing cost and improving performance. We further examine the global effects of smart routing using simulations based on realistic topologies and traffic. Our results show that under traffic equilibria smart routing can improve performance without hurting other traffic.

There are several avenues for future work. In this paper, we focus on algorithmic design and evaluation through analysis and simulation. A natural next step is to implement the algorithms and conduct experiments in the Internet. In addition, we only study the interactions among multiple users under traffic equilibria. It is also interesting to investigate the dynamics of such interactions. Finally, increasingly wide deployment of smart routing poses new challenges to ISPs by intensifying the competition among different ISPs and making traffic less predictable. How ISPs should address these challenges is an open issue.

Acknowledgments

Jason Bender made the Web traces available to us and was very patient with our questions. Rick Summerhill and Mark Fullmer provided us research access to Abilene's traffic data. Young Hyun at CAIDA was very helpful in providing us access to the NetGeo database. NLANR publishes Internet performance data which ultimately made it possible for us to evaluate our schemes under realistic scenarios, and we are grateful to their providers. We would also like to thank James Aspnes, Arvind Krishnamurthy, Theodore Jewell, Jian Yin, and the anonymous reviewers for giving us valuable comments.

9. REFERENCES

[1] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A measurement-based analysis of multihoming. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.

[2] A. Akella, J. Pang, A. Shaikh, S. Seshan, and B. Maggs. A comparison of overlay routing and multihoming route control. In *Proceedings of ACM SIGCOMM '04*, Portland, Oregon, Aug. 2004.

[3] Amextel. `http://www.amextel.com/dedicated.htm`.

[4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th Annual ACM Symposium on Operating Systems Principles*, Banff, Canada, Oct. 2001.

[5] T. Bates and Y. Rekhter. *Scalable Support for Multi-homed Multi-provider Connectivity, RFC 2260*, Jan. 1998.

[6] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for Internet load balancing. In *Proceedings of IEEE INFOCOM '01*, Anchorage, AK, Apr. 2001.

[7] Cisco Inc. Sample configurations for load sharing with BGP in single and multihomed environments. Available at http://www.cisco.com/warp/public/459/40.html.
[8] R. Dai, D. O. Stahl, and A. B. Whinston. The economics of smart routing and QoS. In *Proceedings of the Fifth International Workshop on Networked Group Communications (NGC'03)*, 2003.
[9] F5 Networks, Inc. http://www.f5networks.com/.
[10] M. Garey and D. Johnson. *Computers and Intractability*. W.H. Freeman and Co., New York, NY, 1979.
[11] F. Guo, J. Chen, W. Li, and T. Chiueh. Experiences in building a multihoming load balancing system. In *Proceedings of IEEE INFOCOM '04*, Hong Kong, China, Apr. 2004.
[12] Internap Networks, Inc. http://www.internap.com.
[13] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, 1999.
[14] lp_solve. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.
[15] A. Mihailovic, G. Leijonhufvud, and T. Suihko. Providing multi-homing support in IP access networks. In *PIMRC 2002*, 2002.
[16] NLANR. Round-trip time measurements. Available at http://watt.nlanr.net/Active/raw_data/cgi-bin/data_form.cgi.
[17] A. Orda and R. Rom. Multihoming in computer networks: A topology-design approach. *Computer Networks and ISDN Systems*, 18(2):133–141, 1989.
[18] Pacific Bell. https://ebiznet.sbc.com/calnetinfo/RiderC.
[19] Proficient Networks, Inc. http://www.proficientnetworks.com.
[20] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in Internet-like environments. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.
[21] Radware, Inc. http://www.radware.com/content/products/pd/default.asp.
[22] Rocketfuel. PoP-level ISP maps. Data file policy-dist.tar.gz available from http://www.cs.washington.edu/research/networking/rocketfuel/, 2003.
[23] T. Roughgarden and E. Tardos. How bad is selfish routing? *Journal of ACM*, 49(2):236–259, 2002.
[24] RouteScience Technologies, Inc. http://www.routescience.com, June 2003.
[25] RouteScience Technologies, Inc. Reengineering ISP connectivity to lower bandwidth costs. White Paper. Available at http://www.routescience.com, Apr. 2002.
[26] RouteScience Technologies, Inc. Route optimization for ebusiness applications. White Paper. Available at http://www.routescience.com, 2003.
[27] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proceedings of ACM SIGCOMM '99*, Cambridge, MA, Aug. 1999.
[28] G. Schreck, C. Rustein, and M. Porth. The end of the private WAN. Forrester Brief, Mar. 2002.
[29] P. Sevcik and J. Bartlett. Improving user experience with route control. Technical Report NetForecast Report 5062, NetForecast, Inc., 2002.
[30] S. Sharma, J. Chen, W. Li, K. Gopalan, and T. Chiueh. Duplex: A reusable fault tolerance extension framework for network access devices. In *Proceedings of 2003 International Conference on Dependable Systems and Networks (DSN 2003)*, June 2003.
[31] P. Smith. BGP multihoming techniques. NANOG 23. http://www.nanog.org/mtg-0110/smith.html, Oct. 2001.
[32] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.
[33] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proceedings of SPIE ITCom*, Denver, CO, Aug. 2001.

APPENDIX

A. PROOF OF LEMMA 3

PROOF. We assume that $a_k n$ is always an integer, as justified at the end of Section 4.2.2.1. Let $x_k = \mathtt{qt}(T_k, 1 - a_k)$. Then the left hand side (LHS) of the lemma is $\sum_k x_k = \sum_k \mathtt{qt}(T_k, 1 - a_k)$. To prove $LHS \geq \mathtt{qt}(\sum_k T_k, 1 - \sum_k a_k)$, according to the quantile definition, we need to show that $|\{i| \sum_k t_k^{[i]} > LHS\}| \leq \sum_k a_k n$, *i.e.*, $|\{i| \sum_k t_k^{[i]} > \sum_k x_k\}| \leq \sum_k a_k n$.

By the definition of x_k, we have $|\{i|t_k^{[i]} > x_k\}| \leq a_k n$. Then we have

$$\begin{aligned} & |\{i| \sum_k t_k^{[i]} > \sum_k x_k\}| \\ \leq & \quad |\{i|t_k^{[i]} > x_k, \text{for at least one } k\}| \\ = & \quad |\bigcup_k \{i|t_k^{[i]} > x_k\}| \\ \leq & \quad \sum_k |\{i|t_k^{[i]} > x_k\}| \quad \text{(by Union Bound)} \\ \leq & \quad \sum_k a_k n. \end{aligned}$$

□

B. PROOF OF THEOREM 5

PROOF. The proof is by reduction from the NP-complete *set-partition problem* [10]. Specifically, the set-partition problem is to determine whether the numbers in a given set S (with non-negative integer values) can be partitioned into two subsets A and $S - A$ such that the sum of the elements in A is equal to half of the total sum of all elements in S.

Given an instance of the set-partition problem (a set S), we construct an instance of the integral assignment problem as follows. First, the assignment problem has two ISPs, each with a cost function $c(x)$ equal to 0 when $x \leq \frac{1}{2}\sum_{y \in S} y$, and 1 when $x > \frac{1}{2}\sum_{y \in S} y$. We assume that both ISPs use 100th-percentile charging. Second, we map each element in S to a flow in the assignment problem, with the size of each flow (at all intervals) being the value of the corresponding element. Then if we have any polynomial-time constant-approximation-ratio algorithm to the assignment problem, we can decide the original set-partition problem in polynomial-time by checking whether the cost returned by the approximation algorithm is equal to 0 or not. □

C. MINIMIZING COST OF TOTAL VOLUME BASED CHARGING

The problem of minimizing cost when ISPs use total-volume based charging can be cast into the following linear program, where c_k denotes the cost function of ISP k, t_k is the total traffic assigned to ISP k during its charging period, and $TotalTraffic$ is the total volume of traffic during the charging period. This problem can be readily solved using LP software such as lp_solve [14].

$$\begin{aligned} \text{minimize } & \sum_k c_k(t_k) \\ \text{subject to } & \sum_k t_k = TotalTraffic \end{aligned}$$

Figure 21: LP formulation for total volume-based charging.