

Flow Scheduling for End-host Multihoming

Nathanael Thompson, Guanghui He, and Haiyun Luo
Dept. of Computer Science, University of Illinois at Urbana-Champaign
Email: {nathomps,ghe,haiyun}@cs.uiuc.edu

Abstract— Fueled by the competing DSL and Cable technologies, residential broadband access has seen a significant spread in availability to the point that many users have a choice from several ISPs. At the same time, 802.11 networks have spread rapidly in the residential area, and it is common for neighbors to be able to access each other’s wireless routers. End-users can leverage this diversity to improve their Internet connectivity at no additional cost by pooling all available Internet connections, both their own and their neighbors’ via wireless. In this paper we present our design and evaluation of flow scheduling algorithms in PERM, a framework for practical end-host multihoming. PERM scheduler employs automated on-line analysis of the end-users’ networking behaviors, and exploits the recognized patterns to achieve high-performance scheduling at flow level. We verify our models of end-user’s network traffic with large residential TCP traces. Based on these models we propose algorithms for scalable pre-probing and hybrid flow scheduling. Intensive experiments in our prototype testbed show that PERM scheduler reduces the latency by up to 50% for light-volume flows, and reduces the mean transmission time of heavy-volume flows by nearly 28% and 62% compared with a single Cable or DSL connection respectively. The PERM scheduler also out-performs algorithms for enterprise multihoming by up to 15% and 27% in mean transmission time for light- and heavy-volume flows respectively.

I. INTRODUCTION

Significantly improved speed and the “always on” feature have been driving the deployment of broadband Internet access, predominantly DSL (Digital Subscriber Line) and Cable, in the residential area. In the US as of April 2005, broadband reaches 57% of homes, with the market shared around 1:2 between DSL and Cable [1]. Additional access links are also growing in popularity. For example, Internet data connections can be made through many cellular phones via their 2.5/3G data subscriptions and their built-in Bluetooth interfaces. Corporations and universities often provide free 56k dial-up for their employees. Other alternatives, e.g. Fiber-to-Home, WiMAX, and 4G, are also under aggressive development.

The number of Internet connections in residential area is further increased by the widespread installation of high-speed 802.11 home networks¹. Figure 1 shows the plat map of the subdivision where one of the authors lives. The area is served by three ISPs: Mc Leod USA and SBC with DSL, and InsightBB with Cable. As we can see from the map an average number of 6.4 802.11 networks were detected in the drive ways of 74 houses, using a ThinkPad T41 laptop with integrated 802.11 interface and internal antenna on a

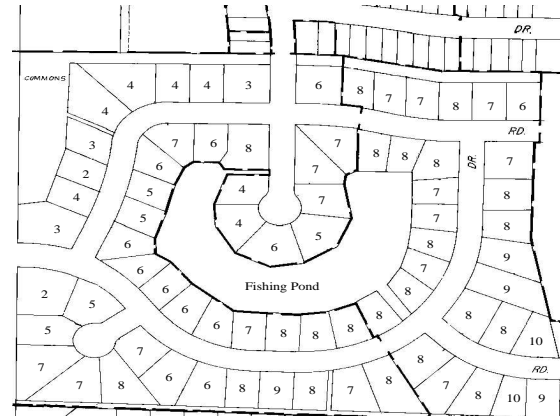


Fig. 1. Number of per-household 802.11 networks detected in a residential area

sunny afternoon. In such scenarios a residential user can easily connect to neighbors’ 802.11 wireless routers², expanding the set of Internet connections with multiple ISPs.

Although this fragmented market for last-mile Internet access is desired for fostering competition among ISPs, the fragmentation is against the spirit of statistical resource sharing in data networking. For a typical residential user, while his broadband connection is always on, it is probably idle for most of the time. When the user actually starts Web surfing, P2P downloading, and/or video streaming, popular DSL or Cable’s average 100~700Kbps downlink speeds and even lower uploading speeds can hardly support more than one flow. By pooling all locally available Internet connections with neighboring households’ access (amongst those users whose service agreements allow non-profit sharing), Internet connectivity can be improved in terms of lower latency, higher throughput and resiliency, etc. Moreover, on-going research and developments on 802.11-based community mesh could further extend the range of the last-mile Internet access sharing from one-hop neighboring households to the entire local community.

In light of the preceding observations it is our belief that *multihomed collaborative Internet access will become the norm in the residential area instead of niche exceptions*. The technology development and deployment are mature, and the potential benefits are tangible. What is missing is an architectural framework to enable multihomed Internet access at end-hosts. We answer this call with our design

¹A January 2005 survey by Parks Associates found that 52% of US households with a home network were using 802.11.

²Through multiple or a single 802.11 interface [2].

of PERM, a framework for practical end-host multihoming (PERM) in the residential area. PERM exploits broadband Internet access diversity in residential area for better last-mile Internet connectivity, and can be immediately deployed at no additional cost.

For immediate deployability in residential end-hosts, the design options are confined. Specifically, the design must not rely on any support outside the target end-host, i.e. network support beyond best-effort unicast, or support from the remote host. Link multiplexing (e.g., R-MTP [3], p-TCP [4]) and/or connection migration (e.g., through Mobile-IP [5], IPv6, MSOCKS [6], MAR [7], Migrate [8], SCTP [9] and HIP [10]) have been used to increase throughput and improve resiliency. These designs, although powerful, necessitate support from both ends of the connection and/or intermediate proxies. For a residential end-host most connections are established with a remote Internet host (e.g., www.yahoo.com). Thus it is unrealistic to assume the necessary support be available there. Moreover, we cannot assume routing control beyond first-hop route selection at the PERM-enabled end-host, which distinguishes PERM from source routing (e.g., IP Loose Source Routing, Nimrod [11], and SOSR [12]) and overlay routing (e.g., Detour [13] and RON [14]). IP source routing requires special router support, while other designs require the deployment of one or more Internet intermediaries.

The above analysis leads us to a framework where all the intelligence resides at the end-host. PERM schedules traffic through different ISPs³ at the flow⁴ level, in contrast to packet level stripping. Furthermore, a flow does not migrate once scheduled, since it is probably impossible to move a connected flow unilaterally. These constraints imply that PERM only enforces its scheduling control at flow connection time. Finally, we cannot rely upon user applications to explicitly specify the characteristics of their flows. It is therefore challenging for PERM to effectively schedule flows with different demands to the Internet connections with time-varying resource availability, while keeping the overhead negligible.

We address these challenges through novel end-host traffic modeling and prediction based on automated on-line analysis of the end-user's networking patterns, a unique opportunity in end-host systems. We use the 4-month (Nov. 2003 - Feb. 2004) 8.6GB network traffic traces collected in *residential* buildings at Dartmouth College campus [15]. The traces include traffic from 1345 unique MAC addresses. We analyze the networking traffic of the top 100 users with the largest number of daily connections (more than 105). Our major discoveries include a strong temporal correlation among the sets of destination addresses a user visits at adjacent time intervals, and the long-range dependency of the traffic volume when a user communicates with a specific remote host. Based on these

findings we design on-line prediction algorithms to infer the set of remote host IP addresses that a user will most likely communicate with in the next time interval, as well as the traffic volume associated with each connection. With those models and prediction algorithms we then design *scalable pre-probing* and *hybrid flow scheduling* to achieve zero-latency, traffic-aware, non-preemptive scheduling for end-host multihoming.

We evaluate the performance of PERM scheduling in our prototype testbed at a residential home with both DSL and Cable subscriptions. Compared with a single Cable or DSL, PERM scheduling decreases the latency by up to 31% and 29% respectively for light-volume flows (with volume less than 8KB), and decreases the transmission time of heavy-volume flows by up to 28% and 65% respectively. PERM scheduler also out-performs existing enterprise multihoming scheduling algorithms, e.g., hash-based multihoming, load balancing multihoming [16], and multihoming based on sliding-window active probing [17]. For heavy-volume flows PERM scheduling reduces average transmission time by 28% compared with the hash-based scheduling and 12% compared with the load balancing scheduling. These multihoming schedulers are optimized for enterprise traffic aggregates, therefore do not benefit from the awareness of individual users' traffic patterns as PERM schedulers do.

Our contributions in this paper are three-fold. First, we identify the opportunity of end-host multihoming for last-mile collaborative Internet access. Our design incurs no additional cost and requires no support outside the target host, therefore is immediately deployable. Second, we identify the need of modeling individual users' network activities for high-performance PERM scheduling, and demonstrate the effectiveness of such models in scalable pre-probing and hybrid flow scheduling. To the best of our knowledge, we are the first to study the patterns of *individual* users' networking behaviors, and exploit those patterns to optimize the perceived performance of broadband Internet connection at a residential user. Finally, we implement the PERM schedulers for the Linux operating system, and present performance evaluation results.

The rest of this paper is organized as follows. We begin the discussion with a review of related work in Section II. In Section III we describe the PERM framework and identify six important components. We present our analysis of user networking patterns and the design of scalable pre-probing in Section IV, and design of hybrid flow scheduling in Section V. In Section VI we describe our prototype implementation. Section VII presents our performance evaluation. We discuss the future work of incentive management and security enforcement, as well as other design options in Section VIII. Finally Section IX concludes this paper.

II. RELATED WORK

In this section we first examine existing solutions for improving end-host Internet performance using multiple links. We then compare PERM scheduling with existing enterprise

³Note that multiple connections do not necessarily mean the same number of ISPs, as neighbors may subscribe to the same service provider. An end-host can choose (e.g., through reverse DNS lookup) to collaborate with neighbors that are with different ISPs.

⁴In this paper we identify a flow by the end-host and a remote host IP address.

multihoming algorithms. Finally we summarize wireless traffic modeling and analysis.

Bandwidth aggregation has been intensively studied for mobile devices with multiple interfaces to increase throughput and connection reliability. Existing work can be classified by the layer of the protocol stack where it is implemented. Link layer solutions coordinate multiple underlying radios and present a virtual interface to higher levels [18], [19], [20]. At some point outside of the end-host the traffic has to be recombined into a single stream. Transport layer bandwidth aggregation, including [4], [3], [9] supports multiple access links. However, it requires modification at both ends of the connection. Another approach relies on the deployment of intermediate proxies (e.g., aggregation proxy in [21], and Mobile-IP agent in [5]). However, the availability of such middleboxes and their appropriate placement cannot be assumed. Because all of these solutions require some support from a second host, they are impractical in the residential end-host which has no control over the outside network.

Enterprise multihoming is probably closest to our research on end-host multihoming. Akella *et al.* first quantified the performance gain of multi-homing a subnetwork in [22]. They show more than 40% improvement using multiple providers, assuming perfect knowledge about the providers and ability to change routes arbitrarily. Following work [23], [24] reveals that multihoming could achieve the same level of performance as overlay routing. Their experiments on enterprise multihoming [17] provide empirical performance studies of NAT-based in-bound ISP link selection using latency-based scheduling. Latency is measured using passive monitoring, or active probing with the help of either a sliding window or frequency statistics of the historical records. Techniques based on DNS or HTTP-redirect are also used for inbound traffic control. Multihoming scheduling for load balancing is summarized by Guo *et al.* in [16].

However, DNS and HTTP redirect approaches involve the explicit cooperation of multiple entities, therefore are inapplicable to end-host multihoming due to the sheer number of end-hosts and the fact that an end-host generally does not provide content or services. The NAT-based approach requires an external device, which is unlikely available for every end-host. Moreover, the scheduling algorithms used in Internet multihoming are optimized for the enterprise traffic aggregates, where the traffic rate is a lot higher than that at an individual end-host. The higher traffic rate softens the effects of mis-scheduling and enables passive monitoring used in latency-based scheduling. End-host multihoming, nevertheless, does not have such a luxury. Finally, latency-based scheduling is less effective when throughput becomes the dominating factor [16]. PERM adopts a hybrid latency/load-balancing scheduling scheme, and adapts based on the expected volumes of individual flows. It performs uniformly well for both latency-sensitive light-volume flows and throughput-sensitive heavy-volume flows.

Wireless traffic modeling and analysis is very useful in facilitating traffic engineering and resource management. Early

studies based on the Dartmouth College 802.11 WLAN wireless traces [15], also used in this paper, analyzed the temporal and spatial distributions of all users, their networked applications, and their traffic aggregate [25], [26]. A recent work [27] proposes a Weibull regression model to characterize static and roaming flows on a per-AP aggregate basis. Specifically for HTTP traffic, the authors of [28] show that the information accessed by HTTP requests holds the property of spatial locality. These previous efforts provide insight and direction to our analysis and modeling. To the best of our knowledge, none of the existing work has studied networking activities from an individual user's perspective.

III. PERM FRAMEWORK

There are three general functionalities PERM must support: detecting new flow requests and binding them to a link, monitoring performance of available links, and deciding which link to use. In addition incentive management to promote collaborative sharing and security management to protect user privacy are also required. We have identified six components for the PERM framework which will cooperate to perform the above functions: 1) a connection manager which unobtrusively collects information and binds new flows, 2) a monitoring component which passively monitors each link and actively measures Internet destinations, 3) a prediction component to build the networking pattern model used to control probing and aid scheduling, 4) an incentive manager which regulates neighbor collaboration, 5) a security component to protect privacy, and 6) a scheduler for assigning flows to links based on the information from all other components.

Connection Manager: New flows generated by applications must be detected in time and bound to the appropriate connections. The PERM Connection Manager intercepts calls to the socket API and triggers the flow scheduling. After a connection has been chosen the Connection Manager binds the new flow accordingly. Because the majority of the flows at the residential end-host are user initiated, both inbound and outbound traffic can easily be controlled. As the majority of traffic is user-initiated in the case of an end-host, PERM Connection Manger does not control externally initiated flows.

Monitor: A profile of each available Internet connection must be maintained to enable proper scheduling. This includes monitoring the links for connection failures, estimating link capacity and measuring the latency to remote hosts. PERM scheduling uses active probing to measure latency and passive monitoring to track link status. Note that PERM scheduling does not use passive monitoring techniques for latency measurements because end-host traffic volume is usually too sparse.

User Traffic Prediction: Given the popularity of TCP flows, PERM scheduling benefits from knowing round trip time to a remote host as well as expected flow volume. Typically these metrics are not known before a connection is established or closed, too late for proper flow scheduling to take place. Our analysis of residential Internet connection traces from the Dartmouth campus shows that both remote host IP addresses

and flow volume are predictable. PERM scheduling utilizes the developed models to determine flow volume before the flow is scheduled, and to select targets for active probing.

Incentive Management: Although we are optimistic of inter-neighbor reciprocal sharing, there needs to be a generic mechanism for creating incentives to share, accompanied by an enforcement policy to ensure that the incentive guidelines are being followed. A broadband subscriber shares his Internet connection based on his subscription method. For example, a subscriber charged with a flat fee may share his connection only when he is idle, while a subscriber charged by usage may share his connection only if he is confident in receiving the same amount of traffic back at higher rate (some users are unable to share based on the ISP user agreement). The PERM incentive manager monitors usage of the Internet connections it owns, interacts with other managers, and provides the scheduler with schedulability information of each connection.

Privacy and Security: By accessing the Internet through a neighbor's Internet connection, a user exposes his own private data for inspection, reveals his traffic patterns, and opens himself for spoofing attacks through his own shared connection. To combat these vulnerabilities PERM includes a privacy enforcement component. The main function of the privacy enforcement component is to protect sensitive data from leaving the trusted network and to prevent neighbors from gaining unintended access to important sites through the local connection. PERM also provides capabilities to hide a user's traffic pattern at the neighbor access point.

Flow Scheduling: PERM implements a novel hybrid scheduler which schedules flows based on the expected flow volume. The important factors when scheduling a TCP flow are round-trip time, competing traffic volume at bottleneck link, and bottleneck link bandwidth. Our measurements to the top 500 web sites listed at Alexa.com show that in more than 80% of scenarios the access link (DSL or Cable) is the bottleneck, and the link bandwidth is constantly around 175KB for DSL and 500KB for Cable. The fact that an end-host controls the traffic at its own broadband connection combined with RTT probing and traffic volume estimates make it feasible to faithfully predict the TCP flow transmission time. According to the estimated flow volume, PERM adapts between scheduling based on latency for light-volume flows and scheduling based on estimated transmission time for load balancing among heavy-volume flows.

For UDP flows that are TCP-friendly [29], they can be treated like a TCP flow. Depending on the incentive management the non-regulated UDP flows are isolated onto dedicated links, e.g., a user's own Internet connection, to protect competing TCP flows.

IV. END-HOST TRAFFIC MODELING

The PERM scheduler resides in end-hosts. The exposure to end-user's networking behaviors provides a unique opportunity. The PERM scheduler exploits this opportunity with a designated component that analyzes end-hosts' traffic and

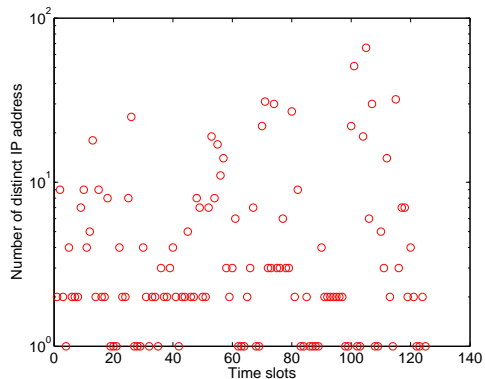


Fig. 2. Number of distinct remote IPs

extracts valuable information to enable high-performance non-preemptive flow scheduling. For light-volume flows whose performance is dominated by round-trip time (RTT), PERM maintains the current RTT on each link by pre-probing the destination. This eliminates the latency of post-facto probing after the connection request is made. For heavy-volume flows whose performance is limited by the available bandwidth PERM schedules the flows based on their predicted volume and the current load on each access link. Since PERM cannot probe all remote Internet hosts, the following two questions have to be answered for scalable pre-probing and hybrid flow scheduling: (1) What is the set of remote IP addresses that a user is most likely to communicate to within the next time interval and is it predictable? (2) What is the distribution of the traffic volume associated with a remote host and is it predictable?

In this section we answer these questions using our analysis of the 4-month (Nov. 2003 - Feb. 2004) 8.6GB network traffic traces collected in *residential* buildings at Dartmouth College campus [15]. We focus our attention on users with the most busy network activity: those 100 users with the largest numbers of daily connections (more than 105 flows per day).

A. Remote IP Addresses

We first study the patterns of remote IP addresses⁵ that an individual user connects with. Figure 2(a) shows the number of distinct remote IP addresses connected to in 10-second time slots. We can see that most of the time the user connects to less than 10 distinct IP addresses, although the number of distinct IP addresses occasionally can be as large as 70.

In Figure 3 (a) we show the remote IP addresses that a user connects to during one day. In the figure, each circle represents a sanitized IP address. We can observe obvious horizontal "lines" which implies that some specific IP addresses are frequently accessed by the user. Another interesting observation is that those "lines" are clustered into two groups, a pattern usually called *bi-modal*. This bi-modal pattern exists daily over 80% of the 100 busy users, while the others exhibit a uni-modal pattern that is not shown here.

⁵Note that IP address sanitizing in Dartmouth traces is prefix preserving.

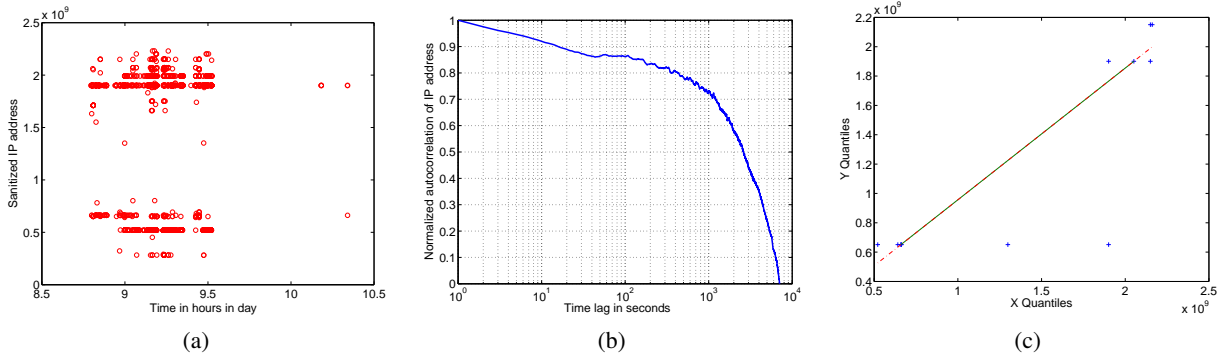


Fig. 3. (a) IP addresses v.s. time, (b) Autocorrelation and (c) QQ-plot for two adjacent slots

To further quantify the temporal correlation of the remote IP addresses visited by the user we show the auto-correlation of IP addresses in Fig. 3(b). We can see that the normalized correlation is above 0.85 even when the time lag is increased to as large as 100 seconds. Figure 3(c) shows the QQ-plot of the two sets of IP addresses. The majority of samples ($\sim 80\%$) are in the vicinity of the straight line, which indicates a good match of the IP address distributions in those two time slots.

The uni-modal/bi-modal patterns and the strong temporal correlation suggest that it is possible to predict online the IP addresses a user is most likely to visit based on the near history. Specifically, for each destination address, we can keep a list of IP addresses sorted based on the number of connections within a recent time window. The list is updated as new connections are established. With high probability the user will connect to one of the first m IP addresses in the near future following a connection request, where m is a tunable parameter based on our analysis. This prediction strategy is preferred to more sophisticated ones (e.g., Markov model) for its simplicity.

B. Flow volume

For each remote IP address, we extract the traffic volume of all the connections directed to that IP address as a time series. We use one typical traffic volume series as shown in Figure 4(a) to facilitate our presentation. Our results apply to 82.3% of the remote IP addresses with heavy traffic volume. We can observe that the traffic exhibits burstiness over several orders of magnitude. This burstiness is a hindrance for volume prediction. The dramatic change of up to several orders of magnitude is hard to predict using traditional prediction methods. We address this challenge with a novel method that predicts the order of the traffic volume using an on-line LMMSE predictor. Given the traffic properties we analyze below, our predictor achieves a mean 80% prediction accuracy. The main power of prediction is not to keep track of accurate instantaneous value of the real result. Instead, we seek to characterize stochastic properties such as the mean value.

We define the flow volume series associated with a remote IP address for a typical user as a time series $x(t)$ and define $y(t) = \log x(t)$. Our major finding is that the autocorrelation

function of $y(t)$, denoted as $R(\tau) = \int_0^\infty y(u)y(u+\tau)du$ obeys the following rule:

$$R_y(\tau) \sim -a\tau + b,$$

as shown in Figure 4(b) where a and b are two constants such that $R_y(\tau) \geq 0$. The linearly decaying autocorrelation indicates that process $y(t)$ is strongly autocorrelated and therefore predictable with high accuracy. We choose to employ a LMMSE predictor for on-line estimation of $y(t)$, since it directly capitalizes on the auto-correlation structure and avoids the two-fold prediction errors of other model-based predictors such as FARIMA and FBM [30].

Specifically, given the time series $y(k), k = 1, \dots, n$, we predict the next series sample, $\hat{y}(n+1)$, in the next interval as a weighted sum of the past n history samples:

$$\hat{y}(n+1) = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} \begin{bmatrix} y(1) \\ y(2) \\ \dots \\ y(n) \end{bmatrix}$$

where a_1, a_2, \dots, a_n are the LMMSE coefficients and can be expressed as

$$\begin{bmatrix} a_1 & \dots & a_n \end{bmatrix} = \begin{bmatrix} R(n) & R(n-1) & \dots & R(1) \end{bmatrix} \times \begin{bmatrix} R(0) & R(1) & \dots & R(n-1) \\ R(1) & R(0) & \dots & R(n-2) \\ \dots & \dots & \dots & \dots \\ R(n-1) & \dots & \dots & R(0) \end{bmatrix}^{-1}$$

where $R(n)$ is the covariance function of the time series. In practice this can be estimated as:

$$R(i) = \frac{1}{n} \sum_{t=i+1}^n y(t)y(t-i), \quad 0 \leq i \leq n-1$$

where n is the number of the time series samples kept and is a tunable parameter.

Figure 4(c) shows the on-line predicted flow volume versus the real traffic volume over a period of 10 seconds. We set $n = 20$, since the performance improvement becomes marginal as n exceeds 20. When applied against the Dartmouth traces our LMMSE predictor achieves a relative average error of $\eta_y = 2.5\%$ for $y(\tau)$, i.e., $y(\hat{t}) = (1 \pm \eta_y) \cdot y(t)$. The $x(t)$ estimate, denoted as $\hat{x}(t)$, is $x(\hat{t}) = \exp[(1 \pm \eta_y)y(t)] =$

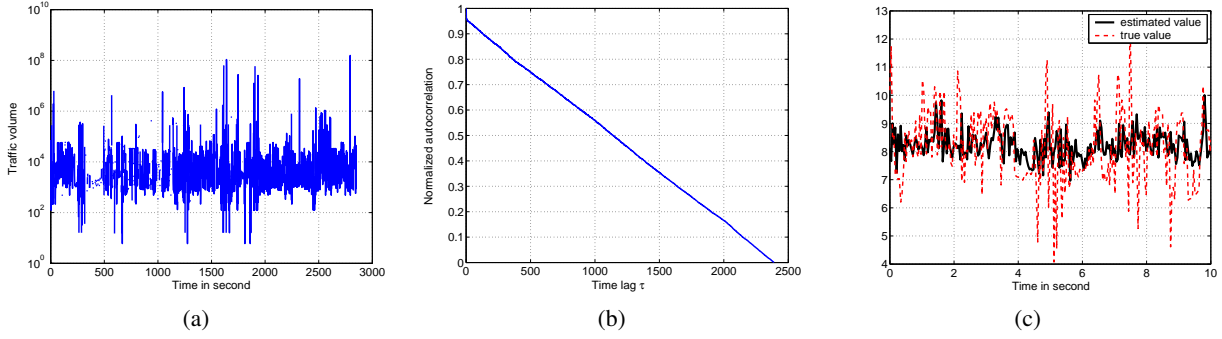


Fig. 4. (a) Traffic volume associated with one IP, (b) Autocorrelation of $y(t)$ and (c) Predicted volume v.s. real volume

$x(t) \cdot \exp[\pm\eta_y y(t)]$ The relative prediction error for $x(t)$ therefore is:

$$\eta_x = \frac{\|x(\hat{t}) - x(t)\|}{x(t)} = \|\exp[\pm\eta_y y(t)] - 1\|$$

In our study, the average order of flow volume $y(t)$ is around 8, and $\eta_y \in [2\%, 3\%]$. Hence the obtained $\eta_x \in [14\%, 20\%]$ as compared with the mean prediction error of around 1000% when directly applying LMMSE to $x(t)$.

V. HYBRID FLOW SCHEDULING

In this section we study how analysis results obtained in Section IV can be used for high-performance flow scheduling. We first evaluate what kind of flows are usually generated in a residential networking setting. We found that for all the traces we obtained, WWW flows dominate (76% in terms of the number and 67% in terms of the volume). Therefore we focus on metrics useful for web traffic, including RTT for light-volume flows and throughput for heavy-volume flows.

On flow establishment phase we classify a flow based on its predicted volume (Section IV-B). If the volume is light (less than 8KB as in Section VI) the flow is scheduled to the connection with the smallest pre-probed RTT (Section IV-A). Otherwise the flow is considered to be a heavy-volume flow and is scheduled based on its predicted volume and the current load on each access link.

Assume the number of available access links is M , and each has an available bandwidth of $B_j, j \in \{1, 2, \dots, M\}$. The number of flows being scheduled is N , and each has a predicted traffic volume $V_i, i \in \{1, 2, \dots, N\}$. Let the scheduling matrix be C , an $N \times M$ array, and the element $c_{i,j} = 1$ denotes that flow i is scheduled to link j , otherwise $c_{i,j} = 0$.

We assume all the flows are TCP flows with pre-probed round trip times $RTT_i, i \in \{1, 2, \dots, N\}$. Since in steady state TCP flows share the bottleneck bandwidth inversely proportional to their RTT^2 [31], [32], we can obtain that for link B_j , the achievable bandwidth $A_{i,j}$ for flow i is $A_{i,j} = (B_j \cdot c_{i,j}) / (Z \cdot RTT_i^2)$, where $Z = \sum_{i=1}^{i=N} \frac{c_{i,j}}{RTT_i^2}$. Given $A_{i,j}$ and V_i , the expected lifetime of flow i is therefore $V_i/A_{i,j}$.

Then the scheduling problem can be formulated in the following two ways:

Minimizing total transmission time

$$\begin{aligned} \min \quad & \sum_{i=1}^N \frac{V_i}{A_{i,j}} \quad \text{for } j \text{ such that } A_{i,j} > 0 \\ \text{s.t.} \quad & c_{i,j} \in \{0, 1\} \end{aligned}$$

Min-Max transmission time

$$\begin{aligned} \min_{c_{i,j}} \max_i \quad & \frac{V_i}{A_{i,j}} \quad \text{for } j \text{ such that } A_{i,j} > 0 \\ \text{s.t.} \quad & c_{i,j} \in \{0, 1\} \end{aligned}$$

“Minimizing the total transmission time” is equivalent to minimizing the time to finish all existing backlogged downloading as a group, treating all flows equally. “Min-Max transmission time”, on the other hand, weighs more on individual flows’ finish time and favors short flows with light volume. Both problems are integer non-linear programming and are NP-complete. Although the number of concurrent heavy-volume flows initiated by an end-host is unlikely to be big, N , C and $A_{i,j}$ are all functions of time as flows dynamically arrive and depart. The time-evolving N , C and $A_{i,j}$ make the above optimization problem prohibitively expensive to solve. We therefore implement heuristic-based greedy algorithms in the PERM framework to ensure low scheduling delay.

Note that for the “Min-Max transmission time” scheduling discipline $\max_i \frac{V_i}{A_{i,j}} = (\sum_i V_i) / B$. It is therefore straightforward to add the volume of the new flow to the total remaining load on each link, and divide the total load by the link bandwidth. The new flow will then be scheduled to the link with the earliest finish time. The competitive ratio is at least $\frac{(3M)^{2/3}}{2}(1 + o(1))$ according to [33].

For the “Minimizing total transmission time” scheduling discipline we need to calculate the finish time of the new flow efficiently. To this end we maintain a sorted list of flows that are currently scheduled on a link, according to the flows’ current finish time. Without loss of generality, we pick link 1 with available bandwidth B_1 , and assume that before f_{new} comes, there are N_1 scheduled flows. Let $A_i, i \in \{1, 2, \dots, N_1\}$ be the achievable bandwidth for flow i and $A_i = B_1 / (\sum_{j=1}^{N_1} \frac{1}{RTT_j^2} \cdot RTT_i^2)$. Since TCP flows share the bandwidth proportionally, adding new flows will not change the order of existing flows’ finish time. Based on this observation we have the following theorem:

Theorem 1: Let V_i be the traffic volume of flow i , RTT_i be the current RTT, B is the available bandwidth for the link and N_a flows share the link including the new one. Then, the k th flow in the sorted list will finish transmission at time T_k :

$$T_k = \frac{\sum_{i=1}^{k-1} V_i}{B} + \frac{\sum_{i=k}^{N_a} \frac{1}{RTT_i^2} \cdot V_k}{\frac{1}{RTT_k^2} B} \quad (1)$$

The list of flows is sorted in ascending order of $V_i \cdot RTT_i^2$.

Proof: Since flows share the link bandwidth proportionally, they finish in the order of $V_i \cdot RTT_i^2$. For those $k-1$ flows that end before the k th flow, their transmissions cause a total delay of $\frac{\sum_{i=1}^{k-1} V_i}{B}$ for the k th flow. For the remaining $N_a - k + 1$ flows, they share the link bandwidth proportionally until the k th flow ends, which bring another delay for the k th flow as the second item in Eq. 1 indicates. \square

Given Eq. 1 the scheduler explicitly calculates the total increase of flow finish time at each link, and schedules the flow to the link with minimum increase.

VI. IMPLEMENTATION

We have implemented the PERM scheduler for Linux end-hosts. The scheduler is installed as a software package requiring no modifications to the kernel or existing applications. The software consists of two pieces: a user-level library which overrides the existing socket API to intercept function calls to the socket library, and a system daemon which maintains PERM states and performs flow scheduling.

The user-level library is loaded using the `LD_PRELOAD` environment variable. This approach allows flexibility of when PERM scheduling is actually used. For applications that do not need or cannot use multihoming, it is simply a matter of unsetting the environment variable to disable PERM. If PERM should be used for all connections the environment variable can be set at login or even system wide. After the library is loaded it intercepts the application’s calls to the socket API. The library communicates via a UNIX socket to the PERM daemon.

The daemon maintains states about remote hosts in a database. This includes most recent RTTs to the remote host, associated IP addresses for IP prediction, and volume estimate. For each available network interface on the system PERM also maintains link capacity, local IP address, flows currently active on the link, and whether the link is active or not. Finally, the daemon maintains a list of predicted upcoming remote-host IP addresses. The prober uses the list to select the pre-probing targets.

As described in Section IV PERM scheduler uses flow volume history to estimate future flow volume. Once a flow has completed the volume of the flow is reported by the library to the daemon. The new data is incorporated into the historical data stored in a log, and the flow volume estimate is updated using the LMMSE estimator. The new estimate is stored in the host database.

For each remote host in the host database, a list of “popular” host IP addresses is also stored. These are the addresses that

are expected to be visited in the near future after visiting the specific remote host. The list is maintained based on historical records. After connecting to a remote host, the next m connections are flagged and added to the host list. For each address in the host list, the frequency of visits is stored. The most frequently visited hosts are considered to be the most likely to follow. Time-stamps are used to phase out older remote hosts. All of the information is stored on disk, along with the historical flow volume estimates, to survive host reboots.

To measure round trip time the daemon probes each of the hosts in the target list every T seconds, set to 60 in our implementation. The probing is done by sending a series of `<SYN,ACK>` packets to the target host, as in [17]. The `<SYN,ACK>` packets cause a `<RST>` packet to be sent from the remote host which is used to measure the RTT. Probes retransmit up to three times before declaring the host is unreachable, with an exponential timeout starting with 750ms and finishing with a timeout of 3sec.

Failures are detected by passively monitoring application and probe traffic. After three consecutive failed connection attempts, the link itself is marked as down. In the future probes will continue to be sent on each link. Once a successful probe runs on the failed link, the link status is reset to active.

When a new `connect` call is intercepted, the library signals the daemon with the remote host address. The scheduler first looks at the estimated flow volume for the remote host. If the volume is under the threshold the flow is considered light volume and scheduled based on latency. In measuring the flow volumes of HTTP GET requests to 500 different popular web sites we found that the median flow volume was 7600 bytes. Based on this result we choose 8 KB as the volume threshold.

A flow estimate may not exist, e.g., when a remote host is being connected to for the first time. Since heavy-volume flows impact the scheduling performance for a longer period of time than light-volume flows, we choose to err on the conservative side and schedule all first-time flows as heavy-volume⁶. We set the initial volume estimate to an arbitrary 1MB for the first-time scheduling, which works well in our experiments. The flow volume estimate will be set to the exact measured volume of the first-time connection to bootstrap our flow volume predictor for future scheduling.

After the scheduler selects the best connection for the new flow the library binds the new flow to the appropriate link via the `bind` call. While the flow is running the library counts the number of bytes transmitted. Upon flow completion the bytes are reported to the daemon and the flow volume estimate for the remote host is updated.

VII. PERFORMANCE EVALUATION

In this section we evaluate the benefits of PERM compared with the base case of a single broadband link (DSL or Cable), and compare PERM scheduling with the ones proposed for

⁶Another observation is that p2p downloads, such as BitTorrent that accounts for more than one-third of Internet traffic [34], tend to connect with arbitrary remote peers and are heavy-volume.

enterprise multihoming [16], [17]. To this end, we implement both the hash-based and the load balancing scheduling algorithms from [16] as well as the sliding window active probing from [17]. Note that it is shown in [17] that the other scheme, namely frequency counts, performs comparably for enterprise multihoming. We choose to compare with the sliding window active probing because in the scenarios of end-host multihoming the traffic is sparse. The number of samples for the frequency counts scheme would be low and an appropriate threshold hard to define. Furthermore, we expect that the sliding window active probing will perform better for web browsing which usually involves repeated connections to the same remote hosts within a small time window.

We subscribed to two local broadband ISPs at the same residence: one Cable provider and one DSL provider. The peak achieved incoming throughput for the Cable modem is around 500KB and for DSL 175KB. Our host PC was connected directly to one broadband modem via Ethernet. A second connection was made to an 802.11 home router which was connected to the other broadband modem.

We first looked at the PERM scheduler’s ability to improve the transmission times for light-volume flows that are sensitive to RTTs. Second, we examined the performance of PERM schedulers when scheduling heavy-volume flows. Finally we analyze the potential reliability benefits of PERM.

A. Light-volume Flows

Because light-volume flows usually do not fill the bottleneck pipe, the most important factor to decrease the transmission time is the round trip time. We studied the behavior of the Cable and DSL links to find what performance benefits were possible. To measure RTT characteristics we repeatedly probed the top 150 web sites as listed at Alexa.com in January 2005 using the probe method described in Section VI. We chose to limit the number of hosts we probed so that every host could be probed in about 5 minutes. Probes were initiated simultaneously on both connections to ensure that Internet-wide conditions were similar. Therefore, each host was probed on each interface around once every five minutes over a period of one week resulting in about 2000 probes to each host.

By probing each connection simultaneously we are able to look at instantaneous RTT on each connection and compare the relative performance of each connection. Our analysis showed that there is a possible average improvement in RTT of 17% by choosing the better connection. The performance of the two different connections was fairly evenly balanced. Across all probes, the Cable has the lower instantaneous RTT for 132198 of the total 299159 probes (44%) and DSL for the remaining 166961 (56%) of the probes. On a per-host basis, for 83 of the 150 web sites (55%) the majority of the instantaneous RTT measurements are lower on the Cable. For the remaining 45% of the remote hosts the majority of the lowest instantaneous RTT measurements are on the DSL. This balance in terms of which connection is overall better for which host means it is necessary to probe the connections on a per-host basis to determine where to schedule the new flows.

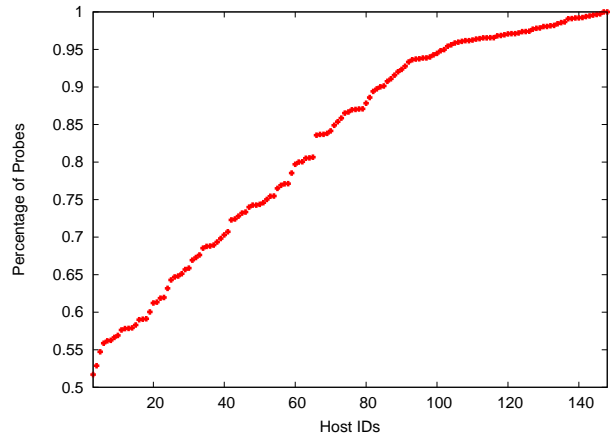


Fig. 5. The percentage of probes that the majority connection had the lower instantaneous RTT for each host (x-axis). Host IDs sorted by ascending percentage.

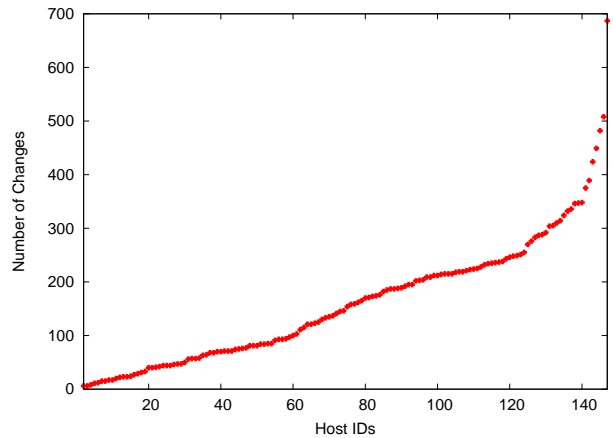


Fig. 6. The number of times the connection with lower instantaneous RTT changed over a 7-day period for each host. Host IDs sorted by ascending number of changes.

For each remote host we also looked at the relative performance between the two connections. The connection which had the majority of lowest instantaneous RTT measurements for a host is labeled the “majority connection”. The other connection is labeled “minority connection” for the host. Figure 5 shows for each host the percentage of probes the majority connection had the lower RTT. On average, the majority connection is the connection with lower RTT for 80% of the time. This result shows that there is a considerable amount of time that the majority connection is the wrong choice when scheduling a light-volume flow. Each host on average spent 6452.1 minutes total on the majority connection and 1296.55 minutes on the minority connection. During the 7-day probing period the lower-RTT connection changed 161 times on average. Figure 6 shows the number of such changes for each host. The majority connection remained the connection with lower RTT on average 110 minutes before a change. On the other hand the minority connection lasted on average 7.8 minutes before a change.

Scheduling scheme	# of hosts w/ min transmission time	Mean transmission time (sec)
PERM	147	2.790
Sliding Window	87	3.278
Load Balancing	80	3.052

TABLE I

THE NUMBER OF HOSTS FOR WHICH A SCHEDULER OUTPERFORMED OTHERS AND THE MEAN TRANSMISSION TIME FOR EACH SCHEDULER.

Although our measurements showed that RTT is fairly stable over the long term, dynamics in latency is high enough in the short term to necessitate continuous RTT measurements. To demonstrate the advantage of PERM’s scalable pre-probing mechanism we used the Dartmouth traces to drive a simulation of PERM’s scalable pre-probing and the sliding window active probing. The destination addresses from the traces were given to the scheduler as if a new flow was initiated. For each new flow if the destination address was in the list of probe targets from the previous probing period, we considered it a hit. PERM’s scalable pre-probing achieved an overall 53% hit ratio, in contrast to the 28% hit ratio of the sliding window active probing. When only HTTP flows were considered PERM’s probing had 92% hit ratio compared with 54% for sliding window active probing.

To quantify the benefits coming from PERM’s more accurate probing, we measured the transmission times of small web downloads with different multihoming schedulers. We compared the PERM scheduler, the scheduler based on sliding window active probing, and the load balancing scheduler against the baseline of a single Cable or DSL connection. For each scheduler we downloaded the index page of each of the 500 hosts listed at Alexa.com traffic rankings which had flow volume below the threshold of 8KB. We performed 15 different iterations for each scheduler at different times of the day. Table I shows the number of hosts for which a scheduler had the lowest mean transmission times as well as the overall mean transmission time for all hosts. From Table I we can see that PERM scheduler indeed has more hosts for which it achieved the lowest mean transmission times (nearly twice as many). Moreover, PERM scheduler has the lowest overall mean transmission time, 15% lower than the sliding window active probing based scheduler and 9% lower than the load balancing scheduler. Surprisingly, the load balancing scheduler out performs the PERM scheduler and the scheduler based on sliding window active probing for a considerable number of hosts and actually has lower overall mean transmission time than the scheduler based on sliding window active probing. The probable cause is the extra overhead of the PERM and sliding window active probing compared with the load balancing (analyzed in Section VII-D). When the difference in overhead is removed the load balancing scheduler loses more than half of its winning hosts.

Figure 7 shows the distribution of per-host relative mean transmission time for PERM scheduler with both Cable and DSL. Compared with a single Cable or DSL connection, for

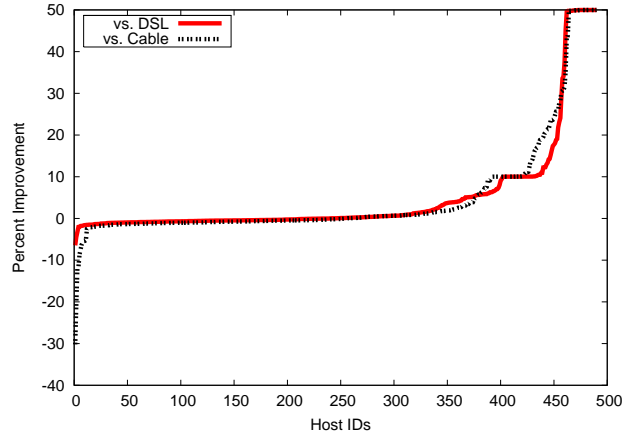


Fig. 7. Mean transmission time improvement percentage of PERM scheduler compared with the mean transmission time of a single DSL or Cable. Host IDs sorted by ascending mean transmission time.

113 and 123 hosts respectively PERM scheduler improves transmission times by more than 5%. For 10 hosts and 1 host respectively, PERM scheduler’s transmission times are worse by more than 5%, consistent with the mis-scheduling rate expected from PERM’s prediction mechanism. For the rest of the the hosts the change in performance was less than 5%. Overall PERM had lower mean transmission times for 235 and 250 hosts compared with a single Cable and DSL respectively, which is an expected result considering that the lowest RTT was divided between each connection nearly equally.

B. Heavy-volume Flows

The transmission time of heavy-volume flows are constrained more by available bandwidth rather than RTT. To evaluate the effectiveness of the PERM scheduler for managing the available bandwidth we initiated several simultaneous downloads of large files ranging in size from 2MB to 30MB. Three different files were downloaded at a time, a realistic scenario in the end-host. The files were hosted on different servers from each other and the downloads combined saturated the connections. The mean transmission times were computed for all the downloads measured over nearly 20 iterations. We compared the two PERM schedulers described in Section V against the load balancing scheduler, the scheduler based on sliding window active probing, and a hash-based scheduler which selects a connection by hashing the destination address and port.

Table II shows the mean transmission times for each scheduler. As seen, the two PERM schedulers out-perform all others, although the Min-Max scheduler is very even with the scheduler based on sliding window active probing. The Min-Total scheduler out performs the hash-based scheduler by 27.7%, the load balancer scheduler by 11.9% and the sliding window active probing based scheduler by 7.4%. The Cable and DSL were improved by 28% and 62%. More insight into the relative performance of each scheduler can be gained by examining the utilization of each connection, as shown

Scheduler	Mean Completion Time (sec)
Min-Total	186.21
Min-Max	199.14
Sliding Window	201.23
Load Balancing	211.37
Cable	257.21
Hash-Based	257.57
DSL	500.54

TABLE II
MEAN TRANSMISSION TIME OF HEAVY-VOLUME DOWNLOADS.

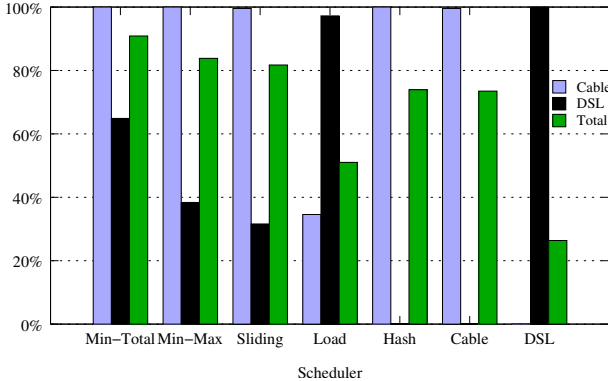


Fig. 8. Utilization of the Cable, DSL, and total capacity (Cable + DSL) with each scheduler.

in Figure 8. Again, the two PERM schedulers were able to make most use of both connections simultaneously. Moreover, the Cable with its larger capacity is favored by both PERM schedulers. Although the DSL was under-utilized (38% for Min-Max and 65% for Min-Total scheduler), the utilization of the combined total capacity is highest. Figure 8 also shows that the hash-based scheduler uses the Cable exclusively. This highlights that hash-based scheduler does not achieve a good balance of load when the number of connections is small and the traffic is sparse. Because there are only two connections in our testbed, the hash algorithm only has two buckets to hash into. Therefore the distribution of flows is skewed. It is possible that when connecting to a different set of remote hosts the hash scheduler would have put all of the flows on the DSL, leading to even worse load balancing.

C. Resiliency

Finally we quantify the failure characteristics of the Cable and DSL connections and analyzed the potential improvement in resiliency with PERM scheduling. We used the same probe output as used in Section VII-A. The total failure rate of our probes was 20.3% on Cable and 12.8% on DSL i.e., 88994 and 56114 probes out of the total 438397 on each connection failed. Of those failures only 3292 (4% of Cable, 6% of DSL) were simultaneous on both connections, meaning that around 94% of the failures could be avoided with PERM scheduling.

D. Overhead

The PERM scheduling itself imposes additional latency in connection set up and tear down. We performed a micro-

benchmark using the socket API functions to measure the overhead of the PERM scheduling. In specific, we measured the amount of time to complete the major function calls of the socket library with and without PERM scheduling. Our measurements show that for `recv`, `send`, `setsockopt` and `getsockopt` there were minor increases in latency: 33% on receive, 16% on send, and negligible on the other two. Higher latency comes from the `socket` and the `connect` calls, since these functions encapsulate the majority of the functionality of PERM scheduling. The latency of `connect` call was increased by 1.40ms or 159%, while the `socket` call was more severe adding 0.050ms or 931%. Considering the typical web page loading time at seconds, those additional latency can hardly be perceived by the user.

VIII. DISCUSSIONS

It is unlikely that a user will be willing to share his broadband access without some form of incentive, despite the performance benefits shown in Section VII. The goal of the incentive manager is to ensure that neighbors will mutually benefit from each other. We are in the process of implementing a tit-for-tat credit based incentive model similar to that used in BitTorrent [35]. In our model every user has a home wireless router which he has full control. On each remote wireless router the user has certain amount of credit. Credit is exchanged between users via the wireless routers whenever a user sends traffic across someone else's router. If no credit remains, any new flows can be rejected by the remote wireless router. When a remote user, A , sends traffic across the local user, B 's wireless router, B is given credit on A 's home router.

The incentive management starts with a registration process. A user contacts his neighbor to register himself, for example by MAC address filtering or by establishing an 802.11i or WPA password. The specific approach taken is not important, as long as the association between remote client and remote wireless router can be made. The neighbor then stores the association between wireless router and user and gives some initial credit, say 10MB to the new user. A user spends credit by scheduling flows on the remote router and gains credit by allowing the remote owner to schedule flows on the local router. Depending on the pricing scheme of the local subscription, an owner may choose to alter his incentive policy to manage its idle usage.

According to the desires of the owner, a wireless router may enforce traffic shaping on the remote flows, giving higher priority to the owner's traffic. This gives the local owner control over the scheduling. Any overflowing connections are allowed to continue, but the offending host is penalized by suspending its usage for a period of time.

This simple tit-for-tat model allows users to share their wireless connections with the confidence that they will gain at least whatever they lose in terms of responsiveness of their Internet connection. Note that incentive management is enforced by security mechanisms to defend malicious attacks. Instead of authenticating using MAC address an 802.11i (or WPA) key would be safer to prevent an attacker from gaining free Internet

access by constantly changing his MAC address. Because the incentive manager is part of the distributed software binary, cheating is not possible unless the user rewrite the software. For the majority of users, this is not an issue. We note that this model does not take into account host mobility or users with no home wireless router in communication range to offer, two aspects we plan to address in the future.

Collaboration creates new security concerns for network owners by opening the local router. Not only are local computers connected to the network more vulnerable, but private online data may be vulnerable to spoofing attacks. MAC filtering can be used to combat these vulnerabilities. Remote MAC addresses can be blocked from accessing local resources and also blocked from sensitive Internet destinations. The registration process needed for incentive management can serve to establish MAC filtering, which is readily available in most home routers. For example, the owner might block his neighbors from accessing his bank's web site through his own wireless router. Other destinations that might be in the filtered list include government sites, financial institutions or medical repositories. Link layer privacy is achieved by encrypting communication to the neighbor's wireless router with an 802.11i (or WPA) key managed by the incentive module. Note that the security module protects the wireless link and does not replace higher layer security functions (e.g., IPsec and SSL).

Another privacy threat is the analysis of the traffic that passes through a neighboring wireless router. To address this concern PERM can randomly generate Internet traffic, for example web requests to randomly selected IP addresses, and send it across the link. PERM generates traffic by replaying connections initiated earlier by the client and also by randomly selecting IP addresses to make new connections. The extra traffic masks the user's real traffic making traffic analysis more difficult. There is a tradeoff between the resiliency against traffic analysis and credit consumption.

Our analysis showed that web and email traffic is very predictable. However, for some types of traffic, notably P2P traffic, the destination address might be quite random. We are currently working on an approach to incorporate application information into prediction, for example by classifying a flow based on the local port it uses. Other types of traffic, like VoIP and RTP traffic can be scheduled based on other metrics, like jitter. The behavior of these metrics must be further studied to examine if performance improvement remains stable over the lifetime of those connections.

Our PERM implementation is largely contained in the end-host. However, the incentive and privacy components must be implemented in the wireless router. The question then arises, can all of the PERM functionality be moved to the wireless router? While this eases implementation, there are still some benefits to keeping PERM scheduling in the end-host. First of all, moving PERM completely to a separate device would require users to have that device available to multihome, even if the end-host device is directly connected to multiple providers. It is possible that the device might not be in range of

the neighboring wireless router while the end-host device is. Of course this effect could work both ways which suggests an implementation that spans *both* the client and wireless router. Second, in order to support client mobility there must be support in the mobile end-host device to associate and coordinate with the incentive managers. Finally, by scheduling at the end-host there is no need to exchange instantaneous flow states for the scheduling of future flows.

In this paper we did not evaluate the effects of cross traffic on scheduling and performance. Although the owner has control of the local router, the exact amount of traffic at the remote router is not known and might lessen scheduling accuracy. Also, whenever separate hosts are scheduling the same resource independently of each other there is a threat of route flapping. However, because PERM schedules on the flow level rather than per-packet the conditions for route flapping should be avoided. We also did not quantify the benefits of prefetching content based on PERM's predictions. Although it complicates security issues, further performance improvements could be achieved if PERM's predictive mechanisms are exploited for aggressive content prefetching and caching.

IX. CONCLUSION

Competition for last mile Internet access coupled with widespread deployment of home wireless networks presents an opportunity for end-host collaborative multihoming. In this paper we present our design and evaluation of flow scheduling algorithms in PERM, a framework for practical end-host multihoming in residential areas. Instead of relying on any support outside of the target host, PERM scheduling uses predictive mechanisms to schedule flows at connection time. PERM's prediction algorithms are based on extensive analysis of residential Internet access traces from the Dartmouth College campus. Scalable probing and flow volume estimation based on new models enable high-performance scheduling. Taking advantage of the modeling and analysis of end-host's Internet traffic, PERM scheduling achieves nearly 15% improvement in mean transmission time for light-volume flows and up to 27% improvement for heavy-volume flows compared with schedulers proposed for enterprise multihoming. Compared with a single Cable or DSL, PERM scheduling reduces latency up to 50% for light-volume flows, and reduces the mean transmission time of heavy-volume flows by up to 28% and 62% respectively.

ACKNOWLEDGMENT

We thank David Kotz for making the Dartmouth campus wireless traces available. Thank you also to Dawid Piwinski for his help with the initial implementation. Finally, many thanks to the anonymous reviewers for their constructive feedback.

REFERENCES

- [1] "April 2005 bandwidth report," www.websiteoptimization.com/bw/0504/.
- [2] R. Chandra, P. Bahl, and P. Bahl, "MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card," in *Proceedings of IEEE INFOCOM*, 2004.

- [3] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Proceedings of IEEE ICNP*, 2001.
- [4] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-home mobile hosts," in *Proceedings of ACM MobiCom*, 2002.
- [5] K. Chebrolu and R. Rao, "Communication using multiple wireless interfaces," in *Proceedings of IEEE WCNC*, March 2002.
- [6] D. A. Maltz and P. Bhagwat, "MSOCKS: An architecture for transport layer mobility," in *Proceedings of IEEE INFOCOM*, March 1998.
- [7] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee, "MAR: A commuter router infrastructure for the mobile internet," in *Proceedings of ACM MobiSys*, 2004.
- [8] A. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proceedings of ACM MobiCom*, 2000.
- [9] R. Stewart, "RFC 2960: Stream control transmission protocol," October 2000.
- [10] R. Moskowitz, "Host identity payload and protocol," Internet-Draft, November 2001, work in Progress.
- [11] I. Castineyra, N. Chiappa, and M. Steenstrup, "The nimrod routing architecture," RFC 1992, August 1996.
- [12] K. P. Gummadi, H. Madhyastha, S. D. Gribble, H. M. Levy, and D. J. Wetherall, "Improving the reliability of internet paths with one-hop source routing," in *Proceedings of USENIX OSDI*, 2004.
- [13] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: Informed internet routing and transport," *IEEE Micro*, vol. 19, no. 1, pp. 50–59, January/February 1999.
- [14] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of ACM SOSP*, 2001.
- [15] "Dartmouth college campus-wide wireless network," <http://www.cs.dartmouth.edu/campus/>.
- [16] F. Guo, J. Chen, W. Li, and T. cker Chiueh, "Experiences in building a multihoming load balancing system," in *Proceedings of IEEE INFOCOM*, 2004.
- [17] A. Akella, S. Seshan, and A. Shaikh, "Multihoming performance benefits: An experimental evaluation of practical enterprise strategies," in *Proceedings of USENIX Annual Technical Conference*, 2004.
- [18] H. Adishesu, G. Parulkar, and G. Varghese, "A reliable and scalable striping protocol," in *Proceedings of ACM SIGCOMM*, August 1996.
- [19] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A multi-radio unification protocol for IEEE 802.11 wireless networks," Microsoft Research, Tech. Rep. MSR-TR-2003-44, July 2003.
- [20] A. Snoeren, "Adaptive inverse multiplexing for widearea wireless networks," in *Proceedings of IEEE GLOBECOM*, 1999.
- [21] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin, "Handheld routers: Intelligent bandwidth aggregation for mobile collaborative communities," in *Proceedings of IEEE BROADNETS*, 2004.
- [22] A. Akella, B. Maggsy, S. Seshan, A. Shaikh, and R. Sitaraman, "A measurement-based analysis of multihoming," in *Proceedings of ACM SIGCOMM*, 2003.
- [23] A. Akella, J. Pang, B. Maggsy, S. Seshan, and A. Shaikh, "A comparison of overlay routing and multihoming route control," in *Proceedings of ACM SIGCOMM*, 2004.
- [24] S. Tao, K. Xu, Y. Xu, T. Fei, L. Gao, R. Guerin, J. Kurose, D. Towsley, and Z.-L. Zhang, "Exploring the performance benefits of end-to-end path switching," in *Proceedings of IEEE ICNP*, 2004.
- [25] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campuswide wireless network," in *Proceedings of ACM MobiCom*, 2004.
- [26] D. Kotz and K. Essien, "Analysis of a campus-wide wireless network," in *Proceedings of ACM MobiCom*, 2002.
- [27] X. G. Meng, S. H. Wong, Y. Yuanz, and S. Lu, "Characterizing flows in large wireless data networks," in *Proceedings of ACM MobiCom*, 2004.
- [28] F. Chinchilla, M. Lindsey, and M. Papadopouli, "Analysis of wireless information locality and association patterns in a campus," in *Proceedings of IEEE INFOCOM*, 2004.
- [29] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of ACM SIGCOMM*, 2000.
- [30] Y. Gao, G. He, and J. C. Hou, "On exploiting traffic predictability in active queue management," in *Proceedings of IEEE INFOCOM*, 2002.
- [31] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proceedings of ACM SIGCOMM*, 1998.
- [32] R. Shorten, F. Wirth, and D. Leith, "A positive systems model of tcp-like congestion control: Asymptotic results," Hamilton Institute, Tech. Rep. 2004-1, April 2004.
- [33] Y. Azar, A. Z. Broder, and A. R. Karlin, "On-line load balancing," in *Proceedings of ACM Symposium on Foundations of Computer Science*, 1992.
- [34] "File-sharing network thrives beneath the radar," <http://in.tech.yahoo.com/041103/137/2ho4i.html>.
- [35] B. Cohen, "Incentives build robustness in bittorrent," <http://www.bittorrent.com/bittorrentecon.pdf>, 2003.