NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

Technical Report
NWU-EECS-09-05
April 16, 2009

EmNet: Satisfying The Individual User Through Empathic Home Networks

J. Scott Miller John R. Lange Peter A. Dinda

Abstract

Current network control systems, including quality of service systems in general, base optimization decisions on assumptions about the needs of a *canonical user*. We propose focusing optimization on the needs of *individual users*. This approach has the potential to increase overall user satisfaction while using fewer resources. We apply the approach to home networks. A careful user study clearly demonstrates that measured end-user satisfaction with a given set of home network conditions is highly variable—user perception and opinion of acceptable network performance is very different from user to user. To exploit this fact we design, implement, and evaluate a prototype system, EmNet, that incorporates direct user feedback from a simple user interface layered over existing web content. This feedback is used to dynamically configure a weighted fair queuing (WFQ) scheduler inside a broadband router that schedules the wide-area link. We evaluate EmNet in terms of the measured satisfaction of end-users, and in terms of the bandwidth required. We compare EmNet with an uncontrolled link (the common case today), as well as with statically configured WFQ scheduling. On average EmNet is able to increase overall user satisfaction by 24% over the uncontrolled network and by 19% over static WFQ. EmNet does so by only increasing the average application bandwidth by 6% over the static WFQ scheduler.

Effort funded by the National Science Foundation under awards CNS-0720691, CNS-0721978, and CNS-0709168. Lange was partially funded by a Symantec Research Labs Fellowship.

Keywords: Network Management, Human-Computer Interaction, Networks, Empathic Systems, Human Factors

EmNet: Satisfying The Individual User Through Empathic Home Networks

J. Scott Miller John R. Lange Peter A. Dinda
{jeffrey-miller,jarusl,pdinda}@northwestern.edu
Northwestern University

ABSTRACT

Current network control systems, including quality of service systems in general, base optimization decisions on assumptions about the needs of a *canonical user.* We propose focusing optimization on the needs of *individual users*. This approach has the potential to increase overall user satisfaction while using fewer resources. We apply the approach to home networks. A careful user study clearly demonstrates that measured end-user satisfaction with a given set of home network conditions is highly variable—user perception and opinion of acceptable network performance is very different from user to user. To exploit this fact we design, implement, and evaluate a prototype system, EmNet, that incorporates direct user feedback from a simple user interface layered over existing web content. This feedback is used to dynamically configure a weighted fair queuing (WFQ) scheduler inside a broadband router that schedules the wide-area link. We evaluate EmNet in terms of the measured satisfaction of end-users, and in terms of the bandwidth required. We compare EmNet with an uncontrolled link (the common case today), as well as with statically configured WFQ scheduling. On average EmNet is able to increase overall user satisfaction by 24% over the uncontrolled network and by 19% over static WFQ. EmNet does so by only increasing the average application bandwidth by 6% over the static WFQ scheduler.

1. INTRODUCTION

Increasingly, computer systems and networks are used to run interactive applications, where the effects of control decisions are perceived directly by the user. Of course, a large body of work in quality of service exists that investigates how to control systems and networks such that they provide acceptable application-level performance to the user. However, it has been demonstrated in contexts outside of networks, such as CPU scheduling and power management, that actual measured user satisfaction with any given operating point exhibits considerable variability across users [9, 19, 6]. Does user satisfaction with network control decisions also exhibit this property?

Measuring *individual user satisfaction* online and then using such measurements directly in the control process makes it possible to exploit this variation to the mutual benefit of users and systems. This result has been demonstrated, again, through user studies, in systems-level control as diverse as power management, scheduling, and remote display systems [21, 18, 16, 17, 14]. Further, it is becoming increasingly clear that such measurement can be done with minimal intrusion on the user, for example through the use of biometric information [22]. Does it make sense to leverage direct user feedback in controlling networks?

Effort funded by the National Science Foundation under awards CNS-0720691, CNS-0721978, and CNS-0709168. Lange was partially funded by a Symantec Research Labs Fellowship.

We claim that the control of networks can benefit from the idea of exploiting measurements of individual user satisfaction. In particular, we propose to associate a user satisfaction level with each network flow, or identifiable group of flows, relevant to an interactive application. Conceptually, picture each packet as having a user satisfaction level stamped onto it, much like an ECN bit, and network elements reading this information as the packet passes.

To evaluate this claim, we consider the context of the home network and its connection to the larger Internet. Previous studies have shown that this is a challenging and important environment in an of itself [1, 11]. In particular, hosts and devices within the home network talk to the Internet via a broadband router, where the upstream link typically has much lower bandwidth than either the home network, or the rest of the path through the Internet [7, 13]. Furthermore, the size of the home network in terms of devices and users is growing, especially with the explosion of wireless. Home users are also increasingly running applications that use many long-lived and high-bandwidth flows alongside their interactive applications, exacerbating the bottleneck upstream link. How to schedule this link to provide a high degree of user satisfaction for interactive applications, while providing excellent service for the other applications, is the problem we consider. We elaborate on the issues of shared home networks, and related work in this area, in Section 2.

Does scheduling this link really need to take into account the individual interactive user or users in the household? We conduct a carefully constructed user study in which participants use a range of interactive web-based and other interactive applications while we vary the nature and degree of cross-traffic representing the traffic of other users in the household. Our participants represent a broad spectrum of individuals at a private university. Section 3 describes our study in detail.

Although our study includes numerous qualitative and quantitative contributions, one result is clearly the most important. That result is that there is a large degree of variation in expressed user satisfaction for identical application/cross-traffic scenarios. This holds across the wide range of application/cross-traffic scenarios we studied. It is an inescapable conclusion that users react in distinct and highly individual ways to cross traffic. It is not the case that a per-application utility model, or other aggregate, captures this variation.

Optimizing a home network by using an objective function that does not express this variation among interactive users misses two opportunities. First, it is likely to result in some interactive users being more dissatisfied than they need be, while others are overprovisioned. Second, it is likely to result in lower performance for non-interactive flows than is necessary.

Having found opportunity in this variation across individual users, we next design and implement a system, EmNet, that provides a mechanism for individual user satisfaction feedback for web-based and other interactive applications. The system, which is described in detail in Section 4, schedules the outgoing link using weighted fair queuing [2, 23]. Non-interactive traffic is given a specific weight, and the weight of each user's traffic has upper and lower bounds. The user can at any time adjust a slider control that is overlaid on top of their current web page or desktop environment. By raising the slider, the user increases the weight of her traffic, but at a cost. The slider control displays the current monetary cost of the user's bandwidth usage and weight/provisioning.

We evaluate EmNet through an extensive user study on a different set of participants from a wide range of backgrounds, as well as other methods. Our evaluation is presented in detail in Section 5. The most important results are the following.

- EmNet increased the average user satisfaction by 24% over an unprovisioned network, and by 19% over a statically provisioned network.
- EmNet's increase in satisfaction came at the cost of only a 6% increase in application bandwidth compared to a statically provisioned network.

These results suggest that EmNet successfully allows individual users to personalize their home network performance, trading off perceived satisfaction and cost. This personalization is beneficial both for the individual users and for the network as a whole.

The lessons learned in our study of user satisfaction with home network performance, and from our EmNet system, suggest a principle. Perhaps measurements of individual user satisfaction, or other individual user feedback could be incorporated into network processing more broadly.

A summary of this work can be found elsewhere [15].

2. CHALLENGES OF HOME NETWORKS

We characterize a home network as a small collection of end-hosts connected by a high-speed wired or wireless network and sharing a residential broadband connection (e.g., DSL, Cable, etc.). Home networks are ubiquitous: in 2007, the Consumer Electronics Association reported that 30% of households in the United States own home networking equipment [5]. The Pew Internet and American Life Project found that in 2008, 55% of adult Americans have broadband Internet access at home [10]. It is safe to say that the quality of the application-level experience of millions of Internet users is in some way influenced by the quality and performance of a home network.

Home networks are a microcosm of the broad range of network applications available to end-users. The Pew study also revealed that in a typical day, 58% of respondents "Use an online search engine," 20% "Watch a video on a video-sharing site like YouTube or Google Video," 16% "Send instant messages," 4% "Download a podcast," and 3% "Download or share files using peer-to-peer networks such as BitTorrent or LiveWire." Other applications found on a home networks include wide-area networked gaming, remote storage and backup, software updates, and voice over IP (the study lacked data on these applications).

A large scale measurement study of broadband hosts conducted by Dischinger et. al. [7] found that the complexity of residential connections extends beyond their throughput characterization. In addition to their (varied) limits in throughput, broadband links typically impose an additional delay (with most residential links adding an additional 100ms to packet RTT when compared to the last-hop router), and substantial loss rates. Throughput is often heavily asymmetric and oftentimes unstable. Using a smaller data set, Claypool et. al. [4] found evidence of substantial variation between the queue size of DSL and Cable providers, leading to varying loss rates and packet delay between ISPs. The work suggests that network provisioning strategies targeted at the wide-area might be insufficient to provide high-quality service for the home network.

HomeMaestro. The role of the home network is to provide network connectivity for a small number of users and their heterogeneous set of applications under varied and at times unreliable constraints. HomeMaestro [1, 11] seeks to mitigate home network contention by using a distributed approach to identify competing flows and allocate network resources fairly to end-hosts. To motivate the need for special intervention, the authors conduct a study of several representative home networks, collecting network traces along with user accounts of dissatisfaction. Surprisingly, the households report daily issues with the network across a range of applications. The HomeMaestro approach is able to provide approximately fair allocations according to a set of application level weights. The setting of the application weights is left as an open question and the author's acknowledge that these weights are likely to be user and application dependent.

We demonstrate one possible way such weights can be derived directly from end-user feedback.

OneClick. Recently, Tu et. al. [24] have described OneClick, a system allowing end-users to signal dissatisfaction with network performance. The feedback is used to construct bounds on network-level quality of service metrics required for acceptable application-level performance. The approach can be applied to applications in which it is difficult to quantify the effects of the network on performance. While the authors' analysis point out some of the challenges of collecting user satisfaction levels, the individual user feedback is largely ignored with the goal being to inexpensively measure *mean* satisfaction level, the Mean Opinion Score concept from various ITU recommendations on digital audio and video.

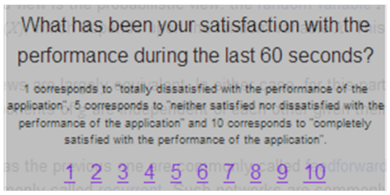
Our work characterizes the extent to which *variation* across users can be seen in user satisfaction with home network conditions. Further, we also demonstrate how to leverage this variation in controlling the home network to improve overall satisfaction.

3. VARIATION IN USER SATISFACTION

Given the intriguing results in other domains mentioned in Section 1, a natural question is whether actual measured user satisfaction given particular network conditions also exhibits significant variation. To answer this question in the context of home networks, we conducted a rigorous user study in which a wide range of users participated. Each was asked to run a set of network applications on an emulated home network, where a variety of cross-traffic scenarios were applied, and prompted for their satisfaction. We found the question could easily be answered in the affirmative—variation in user satisfaction is large. Furthermore, this variation is not explained by variation in application-level QoS measures.

3.1 Instrumentation

We created a user interface that can be transparently applied to existing web applications. Our approach is inspired by AjaxScope [12], a tool for dynamically instrumenting client-side, web-based JavaScript code for Web 2.0 applications. AjaxScope was designed to capture application-level performance metrics aggre-



What has been your satisfaction with the performance during the last 60 seconds?

1 corresponds to "totally dissatisfied with the performance of the application", 5 corresponds to "neither satisfied nor dissatisfied with the performance of the application" and 10 corresponds to "completely satisfied with the performance of the application".

1 2 3 4 5 6 7 8 9 10

Figure 1: User interface: The satisfaction prompt appears over the web page periodically during the studies.

gated from a large number of users, with each user only running a portion of the application with instrumented code. We applied a simplified version of this technique to inject our own instrumentation into existing web pages and applications, with each page and application receiving identical modifications. Like AjaxScope, we use a web proxy to modify existing pages before they are sent to the client. The user need only point her existing web-browser to the proxy server in order to receive the instrumented interface.

The code added to the client page exists in a single JavaScript closure and, aside from the UI elements added to the page's document object model, shares no global state with other code on the page. We wired our code to browser events as to minimize the potential disruption to code executed either before or after our own. The UI communicates with the proxy server using asynchronous call-backs to well-known control URLs that are intercepted by the proxy. From the browser's point of view, these URLs exist under the current page's domain and thus do not pose a security threat that might prevent the asynchronous calls from being executed. The instrumentation loop is wholly separate from the existing application.

When the proxy server, which consists of 1400 lines of Python code, receives a request, the URL is first examined to determine if the request is from the injected instrumentation. If it is, the request, along with any posted data, is dispatched to a handler to either log the data or perform any necessary computation or network control. If this is not the case, then the proxy recursively requests the target URL with the appropriate HTTP method (GET, POST, etc.). When the request returns, the proxy examines the HTTP headers. If the "content-type" header indicates that the requested item is an HTML page, the proxy buffers the page and calls a handler to add the necessary instrumentation. The final page is then returned to the client. If the page is of a different content type (e.g., binary data), then the proxy merely forwards the data directly to the client.

We use this instrumentation in the study described in this section, and in the study described in Section 5. For the present study, we created a simple user interface that asks the user to give their satisfaction with the network performance during the last 30 seconds. The prompt is placed over the current web document on top of an obscuring layer that prevents the user from browsing while the prompt is visible, as can be seen in Figure 1. During our experimentation, we have control over when the prompt is presented via an API exposed by the proxy server. The prompt does not disappear until the user has selected a satisfaction level.

3.2 Testbed

Our testbed is designed to emulate present-day home-networking configurations with the additional abilities to (a) add instrumentation as described in the previous section, and (b) add cross traffic. The testbed is further extended with traffic scheduling capabilities for the study of the study of Section 5.

The testbed is shown in Figure 2. We emulate a broadband router using a layer 2 Ethernet bridge configured to match the bandwidth characteristics of a typical home broadband connection. The

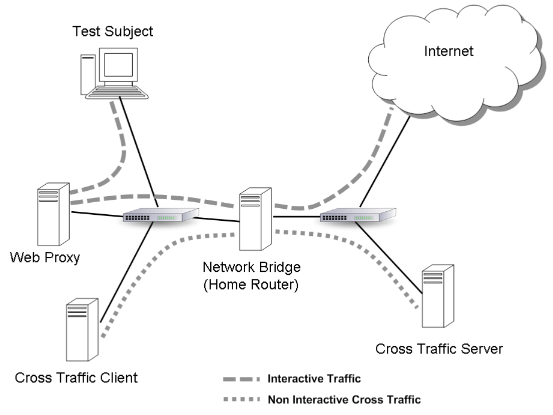


Figure 2: Network testbed used for the user studies. The testbed is configured in a standard dumbbell topology with web traffic traveling through a proxy server before reaching the bottleneck link.

bandwidth on the bridge router is restricted to a 3 Mbps download limit and a 796 Kbps upload limit (i.e., DSL speeds). The bridge server runs RedHat Enterprise Linux ES 4.7 and is configured using IPRoute2. We chose not to emulate a NAT based router to avoid any performance penalties, but instead filter out all traffic not destined to internal machines using IP Address based firewall rules.

We emulate the home user's LAN environment using a 1 Gigabit switch uplinked to the bridge's internal NIC. Another 1 Gigabit switch is placed between the bridge and the Internet link for reasons we will discuss shortly. The uplink port on the external switch was connected to a university network for Internet connectivity.

The LAN environment consists of 3 machines: (1) a client machine used by the test subject, (2) a web proxy server that implements instrumentation as described in the previous section, and (3) a cross traffic generator used to provide configurable background traffic. The client machine is configured to use the web proxy. Because the web proxy is on the internal network, it is not subject to external network fluctuations. Further, it can cleanly measure both client-side response time, and variation in performance of the web servers it interacts with. The proxy server runs Red Hat Enterprise Linux ES 4.7.

The cross-traffic in our testbed is created using a client machine on the internal network and a server machine connected to the switch on the external network. A local cross-traffic server allows us to accurately repeat specific cross-traffic scenarios. The cross-traffic is generated using a modified version of IPerf [20, 3] that provides support for parallel connections, time based traffic generation, and specific bandwidth usage. Both the cross-traffic client and server were configured with RedHat Enterprise Linux ES 4.7.

All of the servers are IBM x335 computers (Dual 2.0 GHz Xeons, 1.5 GB RAM, 40 GB IDE disk, dual Gigabit NICs). The switches are Netgear GS108 8 port Gigabit switches. The test subject machine is a Lenovo Thinkpad T61 (T7500, 2 GB RAM, 100 GB IDE disk, 14.1 inch 1400x1050 display, Gigabit NIC) which runs Windows XP Professional SP2. The web browser used is Firefox 3.0.2. Caching is disabled so that fetches always reach the web proxy.

3.3 Users, applications, and scenarios

We recruited subjects from the general population within a medium-sized private university. Formal approval by our Institutional Review Board allowed us to advertise widely across the university

Application	Familiarity	% Usage
Peer-to-peer	5.05	83%
Web Browsing	6.74	100%
Voice over IP	4.32	67%
Streaming video	5.84	94%
Interactive web apps	6.53	94%
Instant messaging	6.63	100%

Figure 3: Application familiarity and usage information for our study population. Application familiarity was rated on a 1–7 scale, with 1 meaning "Never Used/Not Familiar" and 7 meaning "Very Familiar". We report usage as the percentage of respondents who use the application at least monthly.

using a combination of fliers and email advertisements. Subjects were paid $20 for their time.

The study[1] involved 20 participants including 13 men and 6 women (one participant did not choose to answer the optional demographic questions). Each participant filled out a short survey measuring their use and familiarity with using various applications on a home network. Overall, 19 of the 20 participants have experience using a home network (11 report connecting using a Cable Modem while 8 report DSL). We present a detailed view of the participants experiences in Figure 3. The participants frequently use a wide-range of applications on home networks.

Each of our subjects used three different network applications, as explained below.

Wikipedia: Participants are given a set of thirty questions to answer using Wikipedia [26]. Participants are instructed to complete as many as they can before time runs out. Both in-memory and disk-based caching are disabled during the task to minimize ordering effects within the study. The task is representative of common web-browsing, where traffic is aperiodic and bursty.

Image labeler: Participants play the Google Image Labeler game [8], a two-player web-based game in which players assign labels to images found on the Web. Players attempt to agree on a label for an image without either player having knowledge of the labels chosen by the other user. The next image is presented when both players give the same label and players are scored based on the number and specificity of labels agreed upon. The application generates a large number of small, asynchronous web requests that communicate the game state.

Streaming video: Participants watch a streaming video. The video is compressed using the MPEG-4 codec such that the resulting bandwidth averages 2.9Mbps. Both the client and server are running the Video LAN Client (VLC) [25], version 0.8.5. The video is streamed over UDP without buffering using the MPEG-TS encapsulation method. The stream is unbuffered, such that packet loss introduces immediate effects that vary depending on the amount of data lost.

We consider 16 different cross-traffic scenarios, which are presented in Figure 4. All scenarios are bulk transfers, that is, we do not explore the effect of competing interactive traffic.

3.4 Study design

For each subject, a random ordering of applications was chosen, and then, for each application, a random ordering of scenarios was chosen. The subject began by reading and signing a consent form.

[1]The formal study documents, including protocol and survey instruments, will be made available online for the final version of the paper.

		No. of	
Name	Direction	Flows	Bandwidth
Up.1	Upload	1	100Kb
Up.2	Upload	1	300Kb
Up.3	Upload	1	500Kb
Up.4	Upload	1	768Kb
Up.5	Upload	4	100Kb/con
Up.6	Upload	4	300Kb/con
Up.7	Upload	4	500Kb/con
Down.1	Download	1	100Kb
Down.2	Download	1	500Kb
Down.3	Download	1	1Mb
Down.4	Download	1	3Mb
Down.5	Download	4	100Kb/con
Down.6	Download	4	500Kb/con
Down.7	Download	4	1Mb/con
Mixed.1	Download	4	1Mb/con,
	Upload	4	200Kb/con
Mixed.2	Download	8	1Mb/con,
	Upload	8	200Kb/con

Figure 4: Cross-traffic scenarios.

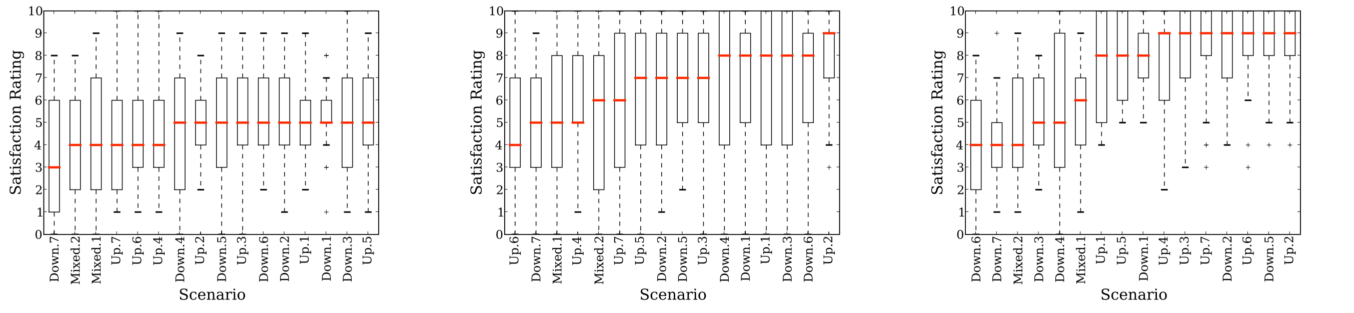
Next, he filled out an introductory questionnaire including demographic information and network application / home networking familiarity. Next, he was given a constrained period of time to familiarize himself with the study setup. He then proceeded to the first application task. He was given a written description and given a constrained period to read it. Then the task would begin. As he worked on the task, each cross traffic scenario would be applied for a fixed period of time. At the end of this period, the user was be prompted for his satisfaction level. This process repeated for the other two application tasks. A final debriefing then occurred.

3.5 Results

Figure 5 presents Box plots of the prompted user satisfaction, one graph for each application, with the graphs broken down further by cross-traffic scenario. The bold lines indicate the medians, while the boxes extend from the 25th to 75th percentiles.

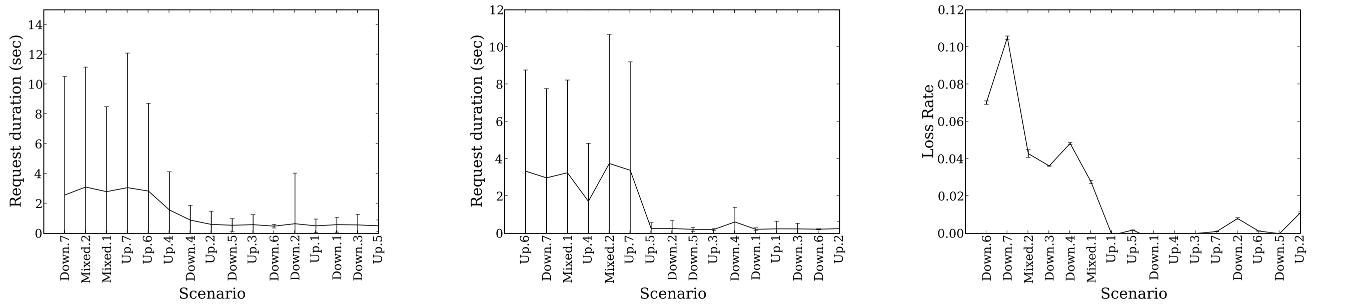
It is abundantly clear that the data shows that for every application/scenario pairing, there exists substantial variation in satisfaction across users. Indeed, this variation for the most part swamps the aggregate differences between the scenarios and across applications. The video application (Figure 5(c)) shows the *least* per-scenario variation, but it is still clearly dominant.

One question the reader may have is whether we are observing the effects of different performance induced by network characteristics on the broader Internet that are not under our control. This is not the case. We measure the response time at the client side and at the network side (from the proxy's request to the ultimate web server). In Figure 6, we present the effect of our cross traffic on the response time of HTTP transfers for both the Wikipedia and Google Image Labeler tasks as well as the packet loss of the streaming video. Note that in Figure 6(a) and Figure 6(b), there is considerable variation in the average response time under the scenarios with the most network contention. However, we see very little response time variation in the scenarios with the least contention, even though considerable variation in user satisfaction remains. In the case of the streaming video, shown in Figure 6(c), several scenarios resulted in substantial packet loss and subsequently low levels of user satisfaction. However, the scenarios with the least contention still produced highly variable satisfaction ratings. This in-



(a) Wikipedia (b) Image Labeler (c) Video

Figure 5: Prompted user satisfaction under different cross-traffic scenarios sorted by increasing median satisfaction. Notice the considerable variation for each scenario, which often swamps aggregate differences between scenarios.



(a) Wikipedia (b) Image Labeler (c) Video

Figure 6: The effect of our cross traffic on the user experience under different scenarios sorted by increasing median satisfaction. Average and variation in page request latency are given in (a) and (b) while average and variation in video packet loss is given in (c). Notice that for both Wikipedia (a) and the Image Labeler (b) tasks, the scenarios impacting satisfaction the least introduced very little average and variation in latency. These scenarios still exhibit considerable variation in user satisfaction. A similar trend can be seen in the packet loss rate of the streaming video (c), though the variation in the of cross traffic is less pronounced.

dicates two things. First, the vast majority of application variation is under our control. Second, the variation in satisfaction cannot be attributed to variation in application-level performance. Clearly, different users react differently to particular application-level QoS level.

Our study shows that users have a wide range of expectations of application-level performance. This implies that an optimal network provisioning strategy–that is, one that maximized the aggregate user satisfaction–is unlikely to be an even or static allocation of network resources.

4. EmNet SYSTEM DESIGN

Based on the results from our initial study we designed a system, EmNet, that is capable of shaping network traffic based on individual user satisfaction. We target EmNet specifically at web-based applications, but believe that it can easily be extended to other application environments. The essential idea in EmNet is that the user is presented with a throttle overlay that also notes the cost of the current throttle setting. The user can change the throttle setting at any point. The throttle setting is then an input to the link provisioning algorithm running on the edge router. The throttle setting can be associated with a set of one or more flows. Our link provisioning algorithm uses the throttle inputs of the different flow sets, and network-wide parameters, to derive weights for WFQ scheduling the Internet link.

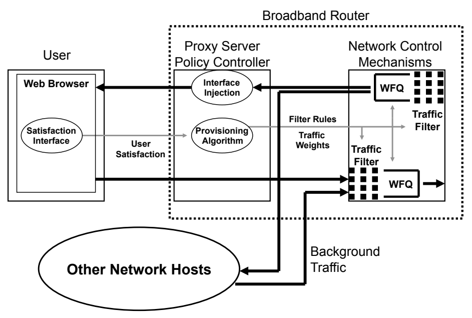


Figure 7: The EmNet Architecture

4.1 Architecture

EmNet is designed to be fully implemented inside a commodity broadband router. Its architecture is shown in Figure 7. Conceptually, the system can be separated into four components: (1) a user satisfaction sensor (the throttle in our implementation); (2) a proxy server that injects the user interface into the user experience (the throttle display injected by the web proxy in our implementation) and tracks managed traffic; (3) a policy controller that uses a provisioning algorithm to configure the network control policies, and (4) a set of network control mechanisms that implement the control policies (WFQ in our implementation).

4.2 Satisfaction sensor

The purpose of the satisfaction sensor is to provide periodic measurements of the individual user satisfaction in real time. These sensor readings are used to derive inputs for the policy controller by determining what needs to be optimized to maximize the satisfaction. Essentially the satisfaction values form a time series that needs to be analyzed to determine whether adjusting the user's network performance will improve satisfaction, and, if so, by how much. In EmNet, we express the network performance as a bounded scalar value greater than or equal to 0. The possible performance values form a linear scale, with 0 meaning minimum performance and the maximum value meaning maximum performance. The collection of these performance values for all users is used as the input to the policy controller.

In the present work, our satisfaction sensor is simply an on-screen throttle combined with a cost display. We require the user to explicitly gauge their own level of satisfaction, and derive the performance values themselves. However, it is important to note that considerable progress is being made in achieving accurate satisfaction measurements with minimal user intrusiveness, for example using biometrics [22].

4.3 Proxy server

The proxy server is responsible for tracking network flows and clustering them with the appropriate satisfaction sensor readings. Many network applications operate with multiple simultaneously open network connections, each of which contributes to the overall performance. In EmNet we assume that user satisfaction is dependent on overall application performance. Specifically, EmNet targets web applications that can use multiple connections to multiple servers to complete a request, and, because of the increased usage of AJAX, applications that use multiple persistent connections. The proxy server is responsible for tracking connections and associating each one with the appropriate performance value calculated from the user's satisfaction.

We refer to each group of connections that the proxy server tracks as a $FlowSet$. Each $FlowSet$ is a collection of network connections that is associated with a single satisfaction reading/performance value. Usually, each $FlowSet$ consists of all the network connections being used by a single application. Each $FlowSet$ is treated as a single unit and the flows within it are operated on in aggregate by EmNet. The proxy server is also responsible for creating descriptions of each $FlowSet$ that can be used as packet classifiers by the network control mechanism. We discuss $FlowSet$s in more detail in the following section.

4.4 Policy controller

The policy controller is responsible for creating the network control policy that will be implemented by the network control mechanism. The policy controller is responsible for ensuring that the link bandwidth is allocated such that user satisfaction is maximized while overall network performance is not significantly degraded. The inputs to the policy controller are the performance values that were previously calculated from the satisfaction of each user. EmNet assumes that not all traffic belongs to interactive applications. We refer to traffic that does not directly affect user satisfaction as *background traffic*. The goal of the policy controller is to find the optimal tradeoff between user satisfaction and the amount of bandwidth allocated for the background traffic.

EmNet is unique in that it gives individual network users easily accessible and direct control over traffic shaping and scheduling policy. However by doing so, its interaction with previous work done to ensure network fairness could be problematic. For example, TCP strives to ensure fairness among multiple network flows, and EmNet could allow users to bypass those mechanisms as well as others. In an idealized setting it should be possible to optimize the network such that overall satisfaction is maximized, however this assumes that all satisfaction measurements are both accurate and truthful. Clearly, if our system maximized bandwidth for users with low satisfaction ratings then users would quickly learn to always indicate that they were unsatisfied in order to maximize their performance. Left unchecked this could allow a single user to monopolize the link bandwidth, causing other users to express their dissatisfaction and leading to a situation where the satisfaction meanings were uniformly bad, and thus useless.

For EmNet to be usable in a multi-user context, there must be mechanisms in place to not only stop a single user from monopolizing the network but also to encourage honesty when users report their satisfaction. To address this issue we rely on two components. First, we developed an algorithm for network provisioning that prevents bandwidth monopolization by a small number of misbehaving users. Second we developed a method of associating a real world cost with lower satisfaction ratings.

4.4.1 Preventing Link Monopolization

The algorithm we use for driving the control policy ensures that no single user or group of users has the power to completely shut off the link for the other users. We do this by partitioning the link such that every user is guaranteed a portion of the link bandwidth. As we noted previously, EmNet groups network flows into one of two types: traffic that is associated with a satisfaction rating and everything else. Each satisfaction value collected by the system is an independent entity. We denote traffic associated with a satisfaction value as a $FlowSet$. N is then the number of $FlowSet$s active in the network. It bears repeating that the policy controller does not operate on individual network flows or the actual satisfaction measurements, but instead on aggregates of flows associated with derived individual performance values.

The output of our algorithm is a partitioning of the network link such that each $FlowSet$ is allocated a portion of the link based on the performance values, while the background traffic is given the remainder. This output takes the form of a set of ratios where the link bandwidth is the uniform denominator, and the individual $FlowSet$ bandwidths are the numerators. The portion of bandwidth that was not allocated to the $FlowSet$s is reserved for the background traffic.

It is important to note that the algorithm operates on the assumption that the network link is under-provisioned, meaning that each network flow is in contention with every other flow. This represents a worst case scenario, and is often avoided due to how the network control mechanism implements the results of the policy controller. We discuss this further in Section 4.5.

4.4.2 Provisioning algorithm

The inputs to the algorithm consist of the set of performance values derived from the satisfaction measurements. Each performance value is associated with a specific $FlowSet$. We refer to the performance value associated with a given $FlowSet$ as $Perf_{FlowSet}$. As we discussed earlier, each $Perf_{FlowSet}$ is a value taken from a bounded linear scale. We denote the maximum scale value as

$Perf_{MAX}$, and set the minimum scale value to 0, so that for every $FlowSet$, $0 \leq Perf_{FlowSet} \leq Perf_{MAX}$.

The first part of our algorithm splits the link bandwidth into two components. One component is reserved and statically allocated between the background traffic and each $FlowSet$, while the other component is dynamically allocated. We denote the static component as $StaticBW$ and the dynamic component as $DynBW$. The total link bandwidth we denote as BW_{Total}. We denote the ratio used to split BW_{Total} as the $SplitRatio$. Its value is constant and set such that $0 \leq SplitRatio \leq 1$.

The amount of bandwidth reserved for each component is thus given as:

$$\begin{aligned} StaticBW &= BW_{Total} \times SplitRatio \\ DynBW &= BW_{Total} \times (1 - SplitRatio) \end{aligned}$$

The $StaticBW$ and $DynBW$ components are then segmented equally between the background traffic and each $FlowSet$. The size of each segment is given as:

$$\begin{aligned} StaticBW_{Seg} &= \frac{StaticBW}{N+1} \\ DynBW_{Seg} &= \frac{DynBW}{N+1} \end{aligned}$$

$StaticBW_{Seg}$ is then immediately allocated to the background traffic and to each $FlowSet$. These static allocations represent the minimum amount of bandwidth guaranteed to each group, and do not change. Initially, $DynBW_{Seg}$ is allocated to the background traffic and to each $FlowSet$ as well, however these allocations are not guaranteed. As the algorithm progresses, the allocation of $DynBW$ is constantly recomputed based on the $Perf_{FlowSet}$ values.

$DynBW$ is allocated depending on all of the $Perf_{FlowSet}$ values. The algorithm calculates the $DynBW$ allocations such that if every $Perf_{FlowSet}$ value was set to $Perf_{MAX}$ (i.e. as if each $FlowSet$ demanded maximum performance) then 100% of $DynBW$ would be allocated to them, with the background traffic receiving only its minimum guaranteed bandwidth. Conversely, if every $FlowSet$ minimized their $Perf_{FlowSet}$ values ($Perf_{FlowSet} = 0$) then the background traffic would receive 100% of $DynBW$. The initial calculated values of $DynBW_{Seg}$ thus represent the case where every $FlowSet$ sets its $Perf_{FlowSet}$ value to $0.5 \times Perf_{MAX}$. In general $DynBW$ is allocated using:

$$DynBW_{FlowSet} = DynBW \times \frac{Perf_{FlowSet}}{\sum_0^N Perf_{MAX}}$$

This formula ensures that the $DynBW$ is allocated amongst the $FlowSet$s as a function of the their $Perf_{FlowSet}$ values, while ensuring that no single $FlowSet$ can monopolize the network link. This method of allocation provides a given $FlowSet$ with a segment of the $DynBW$ bounded by the maximum value of $Perf_{FlowSet}$ it is allowed to set, and determined by the $Perf_{FlowSet}$ value. Because $Perf_{FlowSet}$ can be set to zero, the minimum bandwidth available to either a $FlowSet$ or the background traffic is given by $StaticBW_{Seg}$. Also, the maximum bandwidth available to each $FlowSet$ is given by:

$$MaxBW_{FlowSet} = StaticBW_{Seg} + (1 + \frac{1}{N})DynBW_{Seg}$$

The maximum possible bandwidth allocated to the background traffic is given by:

$$MaxBW_{background} = StaticBW_{Seg} + DynBW$$

In general this algorithm gives each $FlowSet$ control over an equal portion of $DynBW$ that it can either use itself or give to the background traffic. A $Perf_{FlowSet}$ value of 5 out of 10 would mean that the $FlowSet$ is giving half its dynamic bandwidth to the background traffic. However $FlowSet$s are not allowed to allocate another $FlowSet$'s share of $DynBW$. An alternative view is that the link is allocated equally at first, but each $FlowSet$ has the ability to steal its share of bandwidth from the background traffic.

The result generated by the algorithm is a set of values equal to portions of the total bandwidth. The size of the output set is $N + 1$, one value for each $FlowSet$ plus one for the background traffic. Taken together the output set represents the allocated percentage of total bandwidth for each $FlowSet$ as well as the background traffic. The output value for a given $FlowSet$ is represented by $Output_{FlowSet}$, while the output for the background traffic is given by $Output_{background}$:

$$\begin{aligned} Output_{FlowSet} &= \frac{DynBW_{FlowSet} + StaticBW_{Seg}}{BW_{Total}} \\ Output_{background} &= 1 - \sum_{FlowSet=0}^{N} Output_{FlowSet} \end{aligned}$$

While enforcing network partitioning is a valid method for ensuring fair bandwidth allocation under contention, it is suboptimal in situations where network contention is not occurring. In the case of bursty interactive traffic it becomes wasteful to permanently reserve a portion of the bandwidth for each flow. For this reason we implemented our provisioning algorithm on top of weighted fair queuing (WFQ). WFQ allows network fairness to be protected in worst case scenarios when there is contention, while not placing an upper limit on the amount of available bandwidth when there is no contention. We discuss our use of WFQ in Section 4.5.

4.4.3 Cost function

While our control algorithm prevents any single network user from monopolizing the link bandwidth for themselves, it does nothing to prevent users from being dishonest in reporting their satisfaction and thus maximizing their $Perf_{FlowSet}$ values. Ideally, there would be a mechanism which can accurately detect a user's true satisfaction. In practice, we rely on an outside force that provides an incentive for the user to be honest and exerts a downward pressure on the user's satisfaction rating and performance value. This force takes the form of a performance cost function. By associating a given cost to the performance that a user requests, our system can encourage users to be honest in their satisfaction rating.

We do not mandate a particular cost function, we only require that it encourage users to be honest in their allocation settings. The choice of cost function is dependent on the local policy of the network using EmNet . In the context of the home network, this might take the form of a head of household setting per-user limits on network usage with access incurring a cost that grows in proportion to the weight. Perhaps no cost function is required if social pressure is an adequate mediator. A public wireless network may impose a financial cost on usage, once again scaled according to the weight selection.

4.5 Network control mechanism

The network control mechanism uses the results of both the proxy server as well as the policy controller to configure a network scheduler. EmNet uses weighted fair queuing as the mechanism for controlling network performance. All incoming and outgoing traffic is placed into the specific queues and then processed according to the queues' configured weights. By modifying the queue weights and altering the queues that a network connection uses, the network controller is able to optimize specific connections or groups of connections.

The inputs to the network control mechanism are the descriptions of the $FlowSet$s provided by the proxy server, and $Output_{FlowSet}$ ratios from the policy controller. The $FlowSet$ descriptions are used to generate packet classifiers that are capable of routing traffic belonging to a given $FlowSet$ into that $FlowSet$'s queue. The output from the policy controller is used to configure the appropriate weights for each $FlowSet$'s queue.

The queue weights are set by transforming the output ratios from the policy controller (the percentages of the network bandwidth available to each $FlowSet$ as well as the background traffic) into a set of weights that are assigned to the appropriate queue. The transformation is done by simply choosing a maximum weight value, W, that will represent 100% of the link bandwidth. W is then divided between the $FlowSet$s and the background traffic in the same proportions as the bandwidth ratios given by the policy controller. Because the partitioning of the link is implemented using weighted fair queuing, anytime a $FlowSet$ is not sending or receiving network traffic its queue is simply ignored. This has the effect of dropping that weight from the overall fairness calculation resulting in a larger bandwidth share for the remaining $FlowSet$s as well as the background traffic.

As an example, if EmNet was running with 2 $FlowSet$s, each with their performance values set to the half of $Perf_{MAX}$, the link would be equally partitioned into thirds, one third each for each $FlowSet$ and the background traffic. However if one $FlowSet$ was not sending traffic, weighted fair queuing would ignore its queue and process packets from the remaining two queues in proportion to the given weights. This would result in the other $FlowSet$ and the background traffic each getting half of the network bandwidth. If the silent $FlowSet$ began sending network traffic the allocations would return to one third each.

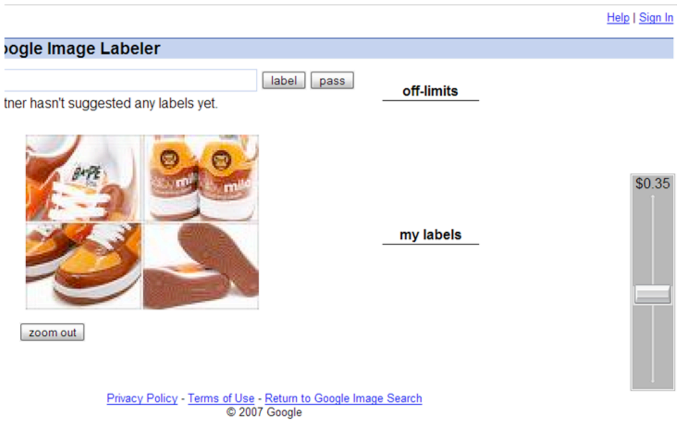
Implementing bandwidth partitions on top of WFQ allows our provisioning algorithm to aggressively impose limits on network bandwidth without worrying whether those limits will cause underutilization of the available bandwidth.

4.6 Prototype implementation

In order to study the feasibility and effectiveness of EmNet, we built a prototype system that could be evaluated on our testbed (described in detail in Section 3.2). Much of our implementation used components developed during the initial user study, and was deployed on several server machines in the testbed. As a result, our implementation is not a monolithic system running inside an emulated broadband router but a distributed system with components at various locations in the network. There are no technical obstacles to building a system that runs on a commodity broadband router.

4.6.1 *User interface*

The EmNet implementation does not directly measure user satisfaction, but instead relies on the user to choose a preferred performance value, $Perf_{FlowSet}$, based on their satisfaction. We collect this value from the user by using techniques discussed in Section 3.1. As shown in Figure 8, a slider control is added to every web page by the proxy. By setting the slider to a given value the user directly indicates their $Perf_{FlowSet}$. The slider's scale is linear with the range 0 to 10, and the initial slider value was set to the middle (5). The interface also includes a display of the current cost value computed by the policy controller, displayed slightly above the slider. The slider setting is sampled every second and the cost is updated every 3 seconds.



tner hasn't suggested any labels yet.

Figure 8: User interface: The slider overlay (right side of screen) hovers over a web page. Users are able to move the slider to reflect the amount of performance they would like to receive. The price is shown.

4.6.2 *Cost function*

We implement a simple cost function that accumulates a price as the user makes use of the bottleneck link. The user effectively "pays" for each byte being transferred. We created a simple server application that monitors all traffic associated with each $FlowSet$, counting the number of bytes transferred. This value is multiplied by the current $Perf_{FlowSet}$ value—the higher the user's performance value, the dearer each byte is. Because the cost is proportional to the amount of data transferred, this tends to bias applications that consume more bandwidth and thus negatively effect our evaluation of EmNet . To limit the effects of this, EmNet multiplies the derived cost by a scaling factor. We tuned this parameter for each application such that the total cost of using the application during the study was within some range salient to the participants (around $10).

4.6.3 *Proxy server*

The EmNet implementation uses the proxy server implementation discussed in Section 3.1. The proxy is still responsible for injecting the interface into web requests and tracking network connections from the user's browser, as well as receiving measurements from the user interface. We extended the proxy to calculate the appropriate values for the cost function and send those values to the user. We also implemented the policy controller in Python inside the proxy server. The proxy server continues to operate on a separate physical machine connected the testbed's internal network. The policy controller implements the provisioning algorithm explained in Section 4.4.

4.6.4 *Network controller*

The network controller was implemented on top of a FreeBSD Server running DummyNET. The server was configured with an Ethernet bridge that matched the performance of a typical home network (3Mbps down, 796Kbps up). We used the FreeBSD Weighted Fair Queue implementation and relied on the kernel firewall to handle packet classification. Our configuration used a pair of rate limited upload and download *pipes* that were fed by multiple WFQs. Each FreeBSD WFQ corresponded to one WFQ from the EmNet design.

Running on the bridge machine is a network control server written in Perl that was responsible for handling control requests coming from the proxy server. The network control server applied the policy configuration given by the proxy using the *ipfw* command line toolset. The commands sent to the network control server consist of a tuple $\langle srcIP, srcport, dstIP, dstport\rangle$ that can specify one or many connections, and a weight that is to be applied to traffic matching the given tuple. The network controller supports wildcard matching for each field in the tuple.

The proxy server is responsible for generating tuples that match the packets belonging to each $FlowSet$ and calculating the appropriate weights to apply. The network control server translates the tuples into packet classification rules used by the firewall to route packets to the correct WFQ configured with the given weight. The network control server creates two WFQs with identical weights for each tuple, connecting one to the upload *pipe* and the other to the download *pipe*. An inverted packet classification rule is applied to the WFQ attached to the upload pipe to match a connection's outbound packets.

5. EVALUATION

To evaluate our EmNet prototype we conducted a second user study with a different set of participants. The participants repeated the same tasks as in the study of Section 3, but they were able to use EmNet to change their network performance. Our goal was to determine if users are able to use EmNet to increase their satisfaction with the network performance.

5.1 Study design

The study we conducted to evaluate EmNet was a modified version of the previous study, described in detail in Section 3. We now summarize the differences.

Instrumentation. The EmNet instrumentation, described in Section 4, was used, in addition to the prompted user satisfaction instrumentation of the original study.

Testbed. The testbed of the previous study, described in Section 3.2 was reused, but with EmNet running on it. This implies a significant change to the data path, in that the bridge server now runs FreeBSD and DummyNet, instead of Linux and IPRoute2.

Users, applications, and scenarios. The study was done using a new set of 18 participants drawn from the same university population as the initial study, using the same recruitment methods. The 18 participants consisted of 11 women and 5 men (once again, 1 participant chose not to answer the demographic questions). Each participant was given the same set of survey questions used in the first study. 16 participants reported experience using a home network. The per-application familiarity and usage questions yielded results similar to those of first study (see Section 3.3) so we omit them here.

The application tasks were the same as in the previous study.

A subset of the cross-traffic scenarios from the earlier study was chosen. These are summarized in Figure 9. The subset was chosen to contain those that had significant effects on median user satisfaction in the first study. A placebo scenario (no cross-traffic) was also included.

EmNet configuration. The EmNet implementation requires the setting of global parameters, as described in Section 4.4. The values used for the study are given in Figure 10. Additionally, the initial performance value $Perf_{FlowSet}$ was set to $0.5 \times Perf_{MAX}$. This is the middle of the scale shown to the user. By forcing the performance value to stay at this position, we can also consider a static WFQ case.

Name	Direction	No. of Flows	Bandwidth
Up.4	Upload	1	768Kb
Up.7	Upload	4	500Kb/con
Down.4	Download	1	3Mb
Down.7	Download	4	1Mb/con
Mixed.2	Download	8	1Mb/con,
	Upload	8	200Kb/con

Figure 9: The subset of network scenarios used for the second user study.

Parameter	Value
$SplitRatio$	0.10
$Perf_{MAX}$	10
N	1
W	100

Figure 10: Provisioning algorithm parameters used by the EmNet Prototype in the second user study.

Study design. The overall study design was identical to that described in Section 3.4, with several differences, summarized here.

In addition to the prompt for satisfaction after each network scenario, users were also given the slider interface. Second, each user was exposed to a given network scenario for 60 seconds (as opposed to 30 seconds in the earlier study). The purpose was to give them time to find a satisfactory setting of the EmNet slider control.

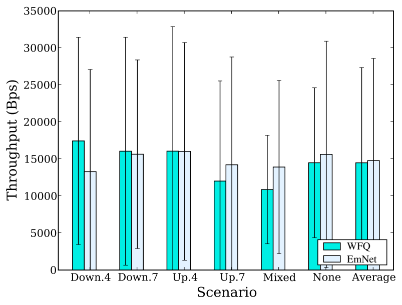
The users were told that moving the slider up was used to increase performance while moving it down decreased performance. To provide an incentive for the users to move the slider down the cost penalty was described to them, and the cost display was always visible as part of the interface. The users were given a goal of minimizing cost while keeping network performance at a level that they were satisfied with.

Every user experienced every network scenario twice; once where the slider control actually controlled his network performance, and once where the slider control did nothing. For the latter cases, a hard-coded $Perf_{FlowSet}$ value of 5 was used in the policy controller, as described earlier. This resulted in a network controller with a statically configured WFQ for these cases.

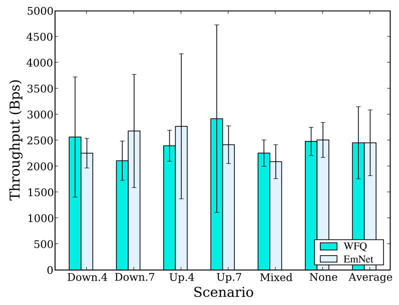
For each user, a random ordering of application tasks, a random ordering of network scenarios within each task, and a random ordering of static WFQ and dynamic WFQ (EmNet) cases were used.

5.2 Results

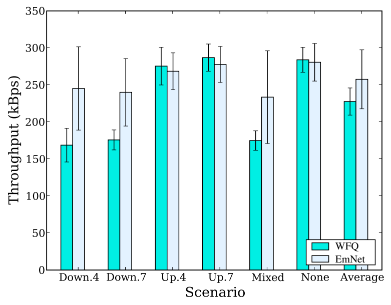
We consider the bandwidth used, the prompted user satisfaction, and the users' chosen performance values. Figure 11 shows the average download bandwidth used by the applications with EmNet and static WFQ, grouped by scenario. The whiskers represent the overall standard deviation, not the confidence interval for the mean. Figure 12 shows the average user satisfaction, and the standard deviation, for EmNet, static WFQ, and for the earlier observational study. Finally, Figure 13 shows the distribution of the performance values ($Perf_{FlowSet}$) chosen by the users.



(a) Wikipedia

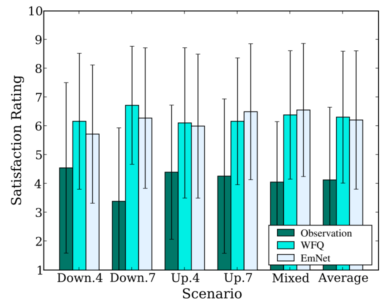


(b) Image Labeler

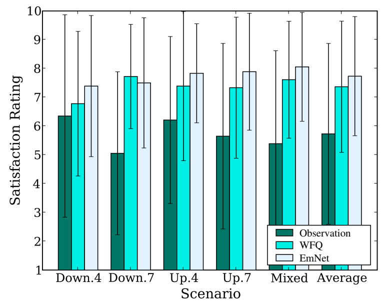


(c) Video

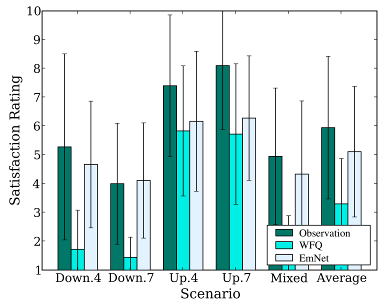
Figure 11: Average downstream throughput under each scenario and system.



(a) Wikipedia



(b) Image Labeler



(c) Video

Figure 12: Average satisfaction level under each scenario and system.

Caveats. Our studies are the first large scale user studies of their kind. Although data points such as ours are extremely expensive to get in terms of funds and manpower, it *is* important to note that we will examine what amounts to about 20 data points per application/scenario/network control configuration triplet. Further, comparisons between the observational study and the evaluation study should take into consideration their differences, which are summarized in Section 5.1.

5.2.1 Wikipedia

In terms of bandwidth consumed, the static WFQ and EmNet results are similar. As seen in Figure 11(a), for both, Wikipedia consumed a moderate amount of bandwidth, but with high variability across the users. The variation in application bandwidth that we see in scenarios where there is significant contention for download bandwidth is generally slightly higher with EmNet than with static WFQ.

Figure 12(a) shows the prompted user satisfaction numbers for Wikipedia. Note that unlike the bandwidth numbers, here we also compare with the observational study results. The overall shape of the results are similar across the cross-traffic scenarios. The average satisfaction levels for static WFQ and EmNet are comparable, and both are much higher than those in the observational study. On average, static WFQ and EmNet users are $\sim$ 33% more satisfied than the observation study participants. For scenarios with cross traffic in the download direction, EmNet shows a very slight average drop in satisfaction compared to static WFQ, which corresponds to the drop in the average bandwidth seen in Figure 11(a).

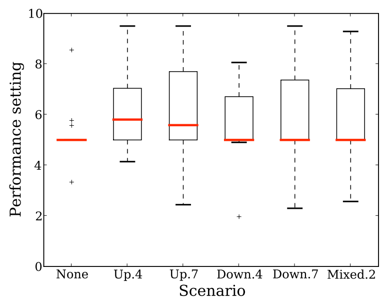
When using EmNet, many users did not change their setting from the default, but those that did increased it, and presumably reaped higher satisfaction. Figure 13(a) shows the distribution of performance values ($Perf_{FlowSet}$) that users chose for each cross-traffic scenario. For each scenario the median value was very nearly 5, which would result in performance very near to that of static WFQ. However, the distribution of the performance values shows considerable variation, where almost all variation is concentrated above the median.

Taken together the results show that for most people, the addition of network control, such as static WFQ or EmNet, dramatically increases satisfaction with the performance of Wikipedia. With such control, cross traffic has little effect on the satisfaction. The average bandwidth consumed by Wikipedia under network control is largely independent of the amount of cross traffic, suggesting that the necessary bandwidth was almost always available to the user. This in part explains the similarity between the static WFQ and EmNet satisfaction results.

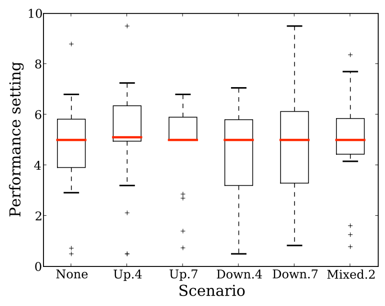
5.2.2 Image labeler

Figure 11(b) shows that the required download bandwidth was small. Furthermore while the bandwidth variability is not insignificant it is generally much less than in Wikipedia. Because the game is interactive this suggests that performance was strongly dependent on latency rather than available bandwidth. Like the Wikipedia results, the average download bandwidths for static WFQ and EmNet change little across different cross-traffic scenarios.

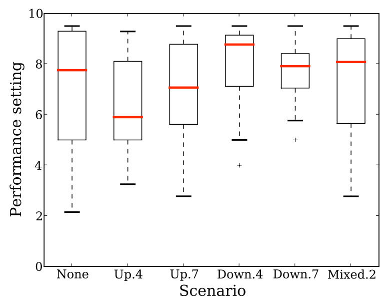
Average satisfaction increases dramatically going from the observational study to static WFQ, and then slightly again going to EmNet. Furthermore, the variation in satisfaction decreases significantly. The satisfaction measurements are given in Figure 12(b). Both WFQ and EmNet increase average satisfaction on the order of $\sim$ 27% compared to the observation study. It is important to point out, however, that due to the very small demand for bandwidth and



(a) Wikipedia



(b) Image Labeler



(c) Video

Figure 13: Variation in user-selected performance values ($Perf_{FlowSet}$) under EmNet for different cross-traffic scenarios.

the significant amount of interaction required, static WFQ and EmNet are likely improving satisfaction largely through providing latency bounds.

Compared to Wikipedia or Video, users of Image Labeler are far more likely to *decrease* their $Perf_{FlowSet}$ values. The distributions of $Perf_{FlowSet}$ values are shown in Figure 13(b). Given the low bandwidth demands of Image Labeler, adequate performance can be had at a lower setting. However, the median *is* 5—we had expected users would choose even lower $Perf_{FlowSet}$ values.

Intuitively, highly interactive low bandwidth applications can deliver very high satisfaction when provided with guaranteed latency bounds. Both static WFQ and EmNet provide this, while the configuration of the observation study does not. Additionally, the results also show that EmNet provides a consistent if small increase in average satisfaction over the static WFQ configuration.

5.2.3 Video

Unlike Wikipedia and Image Labeler, the Video task uses a large portion of the link's download bandwidth. The bandwidth for Video is given in Figure 11(c). Unlike the earlier tasks, the video stream bandwidth was significantly different across the various cross-traffic scenarios, while at the same time showing much less variance in relative terms. Because the video stream was sent over UDP it was less susceptible to upload cross traffic. Both static WFQ and EmNet show similar bandwidth when there is no contention on the download link, but with contention EmNet provides considerably more bandwidth to Video than static WFQ does. With download contention, EmNet provides $\sim 25\%$ more bandwidth than static WFQ. That is, the users are able to demand more bandwidth for Video with EmNet.

Video shows some of the most dramatic differences in satisfaction and the largest differences between static WFQ and EmNet. Figure 12 shows the satisfaction results. Here, the results for the observational study are almost all better than for both static WFQ and EmNet. This is due to the interaction of the video's UDP traffic with the cross traffic and the network control system. In the observation study the UDP-based video stream was free to compete with the TCP cross traffic, resulting in the congestion control algorithm being activated for the TCP flows. This effectively choked off the cross traffic while the video stream monopolized the link bandwidth. With static WFQ the video stream was limited to 50% of the link bandwidth, which resulted in a very large decrease in the video quality that was immediately noticeable to users.

Users report much higher satisfaction with EmNet than with static WFQ. With EmNet users are able to increase the bandwidth Video gets, resulting in large increases in satisfaction. Considering the satisfaction results (Figure 12(c)) in light of the the application bandwidth results (Figure 11(c)). A large increase in satisfaction is purchased with only a tiny increase in application bandwidth. Note that it is probably not possible for EmNet to achieve the same satisfaction results as in the observational study, nor should it be the case—performance there was due to choking the competing TCP flows.

Figure 13 shows the distributions of the users' $Perf_{FlowSet}$ values. As we might expect, most users moved the slider up to improve performance—the medians are all > 5. Further, the median $Perf_{FlowSet}$ values for the cross-traffic scenarios with download contention were the highest. It is interesting to note that, however, that even when there is no cross traffic, the median value remains high, which is unexpected, which could be an artifact of the small number of data points.

Video was unique among the three tasks by the nature of its large bandwidth usage that did not vary much between users. This meant that EmNet would often throttle Video's bandwidth. Thus, the user was in the position of having to increase his $Perf_{FlowSet}$ value for adequate Video performance. He was then forced to carefully choose a balance between cost and performance, namely to find the lowest $Perf_{FlowSet}$ value that was acceptable. While the large variation in user satisfaction shows that there were mixed results in the users' ability to do this, it is also clear that EmNet does a better job of optimizing for user satisfaction than static WFQ. This is best seen in the scenarios with download contention, where a 44% and 37% increase in bandwidth resulted in a 268% and 267% increase in user satisfaction respectively.

5.2.4 General Results

Comparing all three network control configurations (None (observational study), WFQ, and EmNet) across both studies and all cross-traffic scenarios, our results are that EmNet increases average user satisfaction by 24% compared to a configuration without network control. Further, EmNet increases average satisfaction by 19% compared to static WFQ. Finally, EmNet achieves these increases in satisfaction with only a 6% increase in application bandwidth compared to static WFQ.

Interestingly, we find that, overall, users are reluctant to decrease their $Perf_{FlowSet}$ value once it has been raised. While the per-byte cost displayed in the user interface of EmNet is intended to encourage users to decrease their setting as circumstances permit, it is not clear that this cost was a powerful enough incentive. Of course, in the study, the cost reflect no real world monetary value. We speculate that a real cost such as would be expected in a deployment would do better. Alternatively, pressure from other users in the home network could act as an additional cost function. We hope to explore both of these in our future work.

6. CONCLUSIONS

We introduced a new method of optimizing home network broadband connections by using individual user satisfaction as an input. We conducted an observational user study that demonstrated that user satisfaction shows a high degree of variance, meaning that each user's perception of network performance is very different. Optimizing for a canonical user is not sensible. We designed and implemented EmNet, a system that optimizes a home network broadband connection based on measurements of individual user satisfaction. We evaluated EmNet in a second user study that demonstrated that individualized optimizations can considerably improve user satisfaction with low resource cost. On average EmNet is capable of increasing user satisfaction by 24% over an uncontrolled link, and by 19% over a simple static configuration using weighted fair queuing. It does so by increasing average application bandwidth by only about 6%.

7. REFERENCES

[1] ATHANASOPOULOS, E., KARAGIANNIS, T., GKANTSIDIS, C., AND KEY, P. Homemaestro: Order from chaos in home networks. In *Proceedings of ACM SIGCOMM (demo)* (August 2008). Also see Microsoft Research Technical Report MSR-TR-2008-84.

[2] BENNETT, J., AND ZHANG, H. Worst-case fair weighted fair queueing. In *Proceedings of IEEE INFOCOM 1996* (March 1996).

[3] BLOM, H. Modified iperf implementation. (web page). http://staff.science.uva.nl/ jblom/gigaport/tools/test%5Ftools.html.

[4] CLAYPOOL, M., KINICKI, R., LI, M., NICHOLS, J., AND WU, H. Inferring queue sizes in access networks by active measurement. In *Proceedings of the 4th Passive and Active Measurement Workshop (PAM)* (April 2004).

[5] CONSUMER ELECTRONICS ASSOCIATION. CEA finds american adults spend $1,200 annually on consumer electronics products, April 2007.

[6] DINDA, P., MEMIK, G., DICK, R., LIN, B., MALLIK, A., GUPTA, A., AND ROSSOFF, S. The user in experimental computer systems research. In *Proceedings of the Workshop on Experimental Computer Science (ExpCS 2007)* (June 2007).

[7] DISCHINGER, M., HAEBERLEN, A., GUMMADI, K. P., AND SAROIU, S. Characterizing residential broadband networks. In *Proceedings of the 7th ACM SIGCOMM / USENIX Internet Measurement Conference (IMC)* (October 2007).

[8] GOOGLE, INC. Google image labeler. (web page). http://images.google.com/imagelabeler/.

[9] GUPTA, A., LIN, B., AND DINDA, P. A. Measuring and understanding user comfort with resource borrowing. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC 2004)* (June 2004).

[10] HORRIGAN, J. B. *Home Broadband Adoption 2008*. Pew Internet and American Life Project, 2008.

[11] KARAGIANNIS, T., ATHANASOPOULOS, E., GKANTSIDIS, C., AND KEY, P. Homemaestro: Order from chaos in home networks. Tech. Rep. MSR-TR-2008-84, Microsoft Research, 2008.

[12] KICIMAN, E., AND LIVSHITS, B. Ajaxscope: a platform for remotely monitoring the client-side behavior of web 2.0 applications. In *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP)* (October 2007).

[13] LAKSHMINARAYANAN, K., AND PADMANABHAN, V. N. Some findings on the network performance of broadband hosts. In *Proceedings of the 3rd ACM SIGCOMM / USENIX Internet Measurement Conference (IMC)* (October 2003).

[14] LANGE, J., DINDA, P., AND ROSSOFF, S. Experiences with client-based speculative remote display. In *Proceedings of the USENIX Annual Technical Conference (USENIX)* (2008).

[15] LANGE, J., MILLER, J. S., AND DINDA, P. EmNet: Satisfying the individual user through empathic home networks: Summary. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)* (June 2009).

[16] LIN, B., AND DINDA, P. Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. In *Proceedings of ACM/IEEE SC (Supercomputing)* (November 2005).

[17] LIN, B., MALLIK, A., DINDA, P., MEMIK, G., AND DICK, R. Power reduction through measurement and modeling of users and cpus: Summary. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)* (June 2007). Extended version appears as Northwestern EECS technical report NWU-EECS-06-11.

[18] MALLIK, A., COSGROVE, J., DICK, R., MEMIK, G., AND DINDA, P. Picsel: Measuring user-perceived performance to control dynamic frequency scaling. In *Proceedings of the 13th International Conference in Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (March 2008).

[19] MALLIK, A., LIN, B., MEMIK, G., DINDA, P., AND DICK, R. User-driven frequency scaling. *Computer Architecture Letters 5*, 2 (2006).

[20] NATIONAL LABORATORY FOR APPLIED NETWORK RESEARCH. Iperf network performance measurement tool. (web page). http://iperf.sourceforge.net/.

[21] SHYE, A., OZISIKYILMAZ, B., MALLIK, A., MEMIK, G., DINDA, P., AND DICK, R. Learning and leveraging the relationship between architectural-level measurements and individual user satisfaction. In *Proceedings of the 35th International Symposium on Computer Architecture (ISCA)* (June 2008).

[22] SHYE, A., PAN, Y., SCHOLBROCK, B., MILLER, J. S., MEMIK, G., DINDA, P., AND DICK, R. Power to the people: Leveraging human physiological traits to control microprocessor frequency. In *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (November 2008).

[23] STILIADIS, D., AND VARMA, A. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking (ToN) 6*, 5 (1998), 611–624.

[24] TU, C.-C., CHEN, K.-T., CHANG, Y.-C., AND LEI, C.-L. Oneclick: A framework for capturing users network experiences. In *Proceedings of ACM SIGCOMM 2008 (poster)* (August 2008).

[25] VIDEOLAN TEAM. Vlc media player. (web page). http://www.videolan.org/vlc/.

[26] WIKIMEDIA FOUNDATION, INC. Wikipedia, the free encyclopedia. (web page). http://en.wikipedia.org/.