HomeMaestro: Order from Chaos in Home Networks

Thomas Karagiannis,[1] Elias Athanasopoulos,[2] Christos Gkantsidis,[1] Peter Key[1]
Microsoft Research, Cambridge, UK
[1] {thomkar, chrisgk, peter.key}@microsoft.com, [2] elathan@ics.forth.gr

May 2008

Technical Report
MSR-TR-2008-84

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

Abstract

We present HomeMaestro, a distributed system for monitoring and instrumentation of home networks. HomeMaestro performs extensive measurements at the host level to infer application network requirements, and identifies network-related problems through time-series analysis. By sharing and correlating information across hosts in the home network, our system automatically detects and resolves contention over network resources among applications based on predefined policies. Finally, HomeMaestro implements a distributed virtual queue to enforce those policies by prioritizing applications without additional assistance from network equipment such as routers or access points. We outline the challenges in managing home networks, describe the design choices and architecture of our system, and highlight the performance of HomeMaestro components in typical home scenarios.

1. INTRODUCTION

The number and complexity of home networks continues to grow, tracking the deployment and penetration of broadband [20] and driving customer premise equipment (CPE) spending [11]. According to Gartner, the number of worldwide consumer broadband connections is expected to reach 364 million by 2010[1], while Park Associates estimates that there will be some 72 million residential gateways by 2012. Indeed, as this trend persists, home networks may resemble small replicas of complex enterprise networks, with multiple interconnected heterogeneous devices in the near future. Typical examples include desktops, laptops, IP phones, game consoles, home servers, and media centers, running email, web, peer-to-peer, social networking, voice-over-IP, video streaming, on-line gaming, media sharing, and teleworking applications. Despite their wide-spread use and increasing practical significance, the study of managing and organizing home networks has been largely ignored by the research community, in contrast to other types of networks, such as enterprise networks where numerous solutions have been proposed (e.g., [7, 23]). Commercially, the only process currently that relates to network administration in a home ecosystem is the configuration of the home router through a web-based interface.

Home networks have idiosyncrasies and specific requirements that make them unique. Firstly, in contrast to enterprise networks, where policies and objectives are well defined and proscribed by a single authority, home networks connect users who not only have different requirements and performance expectations but also conflicting priorities. Even if a nominal network administrator is elected among the home users to resolve conflicts, it is unclear whether this administrator will have the most knowledge about the network and will be able to prevent other home users from bypassing her administrative rules (e.g., contrast parents with tech-savy children). Secondly, home networks have typically one central open channel, namely the gateway, that interconnects all home network devices to the rest of the world. This implies that all home devices and their applications will share and most likely compete for the available network capacity. Competition for resources also exists between subscribers, since broadband providers frequently oversubscribe their access network. Within the home, the rising number of devices, coupled with the growing trends in inter-connecting these devices (e.g., home servers) and the proliferation of bandwidth-intensive applications introduces in-home bottlenecks.

Thirdly, most home users are unaware of how different network configurations will affect their perceived network performance (and they should be!). In practice, however, minimal changes can lead to high performance differences [17]. Finally, the home network is also the place where a large number of diverse technologies meet and interact, from broadband technologies such as cable and DSL, to local access technologies such as Wireless, Wired or Powerline, some of which with various undesirable side-effects such as large queues and delays [8].

Our work strives to put an order in the chaos of Home networking, by introducing *HomeMaestro*, a distributed system for the instrumentation and monitoring of home networks. *HomeMaestro* tackles the challenges of home networks by automatically identifying competition and allowing for connection and application coordination in a collaborative manner across hosts. Besides their challenges, home network offer opportunities that *HomeMaestro* exploits, as typical networking problems (e.g., network management, bottleneck sharing detection, resource allocation) are strictly simpler than in the general case (e.g., in an ISP backbone). Home networks comprise a relative small number of devices with the sufficient context and extensive information about network usage, allowing advanced algorithms to be developed that infer network properties and enforce desired performance. Additionally, certain levels of trust do exist between home devices.

HomeMaestro monitors the performance of all network applications at every participating host, and employs time-series analysis to infer performance problems. Performance may be defined depending on the application through a variety of metrics such as rate or latency. Tracking such properties, *HomeMaestro* further detects whether these network problems are related to competing traffic flows across hosts or applications, and through priority-based mechanisms and traffic shaping assigns the available bandwidth to applications.

Overall, our contributions can be summarized as follows:

• We describe the architecture of *HomeMaestro*, a distributed system for the monitoring and organization of home networks (§ 3). We implement *HomeMaestro* and test its performance in realistic home scenarios.

• We present techniques to identify performance issues and detect whether these problems are caused by competing traffic flows through time-series and statistical analysis (§ 4).

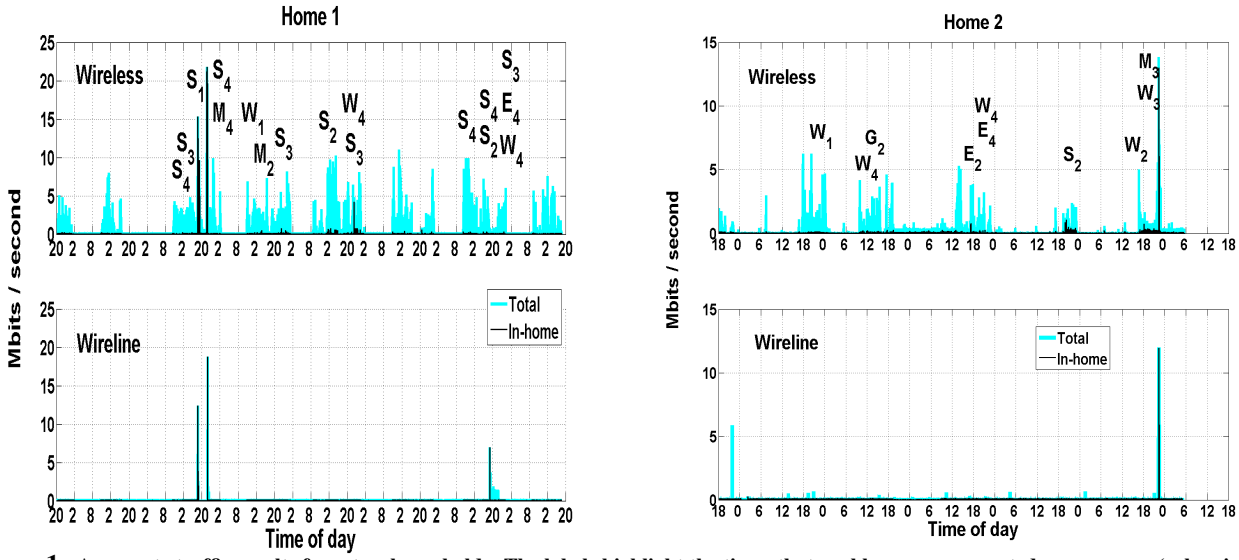


Figure 1: Aggregate traffic results from two households. The labels highlight the times that problems were reported across users (subscripts) and applications (W:WEB, S:Streaming, G:Games, M:Media applications, e.g., image upload/searching, E:email).

• We apply algorithms to enforce traffic priorities through virtual queue [14] and priority-based mechanisms. We distribute information amongst cooperating hosts, and create a distributed proportionally fair allocation, using stable algorithms based on congestion control ideas previously applied to the wide-area [22](§ 5).

• We report evidence from a small-scale user-study of typical households. Our study involved monitoring of home traffic for roughly 3 weeks, while at the same time home-users reported on their perceived user experience (§ 2).

Our vision is to provide a seamless users experience to the home user in the evolving home ecosystem. *HomeMaestro* is a first step towards this direction, sheding at the same time light on the requirements, challenges, open research problems, but also opportunities of current and future home networks.

2. USERS AND THE HOME NETWORK

While we did not expect an ideal world in current home networks, before undertaking this work, we had limited intuition of how users perceive the performance of their home networks. It was also unclear how the various application and traffic properties manifest themselves in the home network. Projects such as Neti@home[21] have presented some data regarding traffic characteristics within the home; however, we needed a more detailed characterization.

Specifically, we wanted to know, *What does home network traffic look like? Does the competition over network resources affect user experience? If so, how do users perceive network-related problems?* To answer these questions, we appealed to sociology researchers of a large European institution. With their help, we identified a small set of typical households and designed the following study. First, we instrumented their home networks without affecting any of their devices or their location within the home to capture all traffic in the network, wireless or wireline. Second, at the same time, users would keep detailed diaries of specific application usage throughout the study period, where they would also comment on their experience, on possible problems they faced, and their assumed cause of the problem. Our goal was to correlate user perception of the network and potential problems with actual measurable networking events. Each study lasted 7 days. Even though we cannot claim that the results are representative, we do believe that they highlight interesting trends and motivate the need for better understanding of home networks.

Sample aggregate results for two of households are presented in Fig. 1. Each household had four users who connected using their personal laptops or desktops to the home access point (AP). Other devices such as gaming consoles and/or media centers also existed. The two figures show the aggregate wired and wireless traffic, as well as the "in-home" traffic, i.e., traffic where both the source and destination IPs were part of the home network. We further annotated the figure with labels to denote the times were users reported problems, and the type of problem (W:WEB, S:Streaming, G:Games, M:Media applications, e.g., image upload or searching, E:email). A number of interesting observations can be drawn from Fig. 1.

Problems were reported almost daily! While we expected that users do face network-related problems, their number highlights significantly poor user experience. Users reported problems at times when the network was moderately loaded. In their descriptions of the assumed cause of the problems, most users do recognize that "problems might be caused by other home users" competing for local resources. However, in many cases the assumed cause was "No idea."

Streaming applications, both video and audio, were particularly popular, and also faced numerous problems. Such

latency sensitive applications suffer even with moderate load, since packet spacing affects their performance. A range of streaming-related applications existed in all homes.

The majority of the problems occurred during the weekends when most users were present. Interestingly, a typical comment in users' diaries was that "Internet in general slow at weekends." Traffic also follows diurnal patterns that are out-of-phase with normal working hours.

In some cases, users reported problems while the network was lightly loaded (e.g., W_4 in the second household, around 1pm in the second day, in-between the large spikes). In some of those cases, the users were running a large number of applications concurrently, which may have affected the responsiveness of their computer.[1] It appears that users "blame" the network for any underperforming network application. This suggests that simply informing the user of where the performance problem lies (e.g., host vs. the network) is valuable.

The vast majority of the traffic within the examined home networks was wireless, and downstream, coming in the home network. We observed low volume of upstream traffic probably due to the absence of observable peer-to-peer (p2p) file-sharing applications in the households during our study. We expect that users would report more problems in the presence of p2p traffic.

In-home network traffic is limited but when present creates significant spikes. In both homes in Fig. 1, spikes were created by large-volume transfers between a wireless and a wired host in the home network. We believe that such spikes are important, and network effects will become more pronounced in the future, as communication between home devices increases (e.g., with the use of home servers, media centers, etc.).

Overall, this user-study highlighted numerous problems with current home networks. As home devices increase in number and advance in functionality, we believe that management and troubleshooting of home networks will be even more problematic.

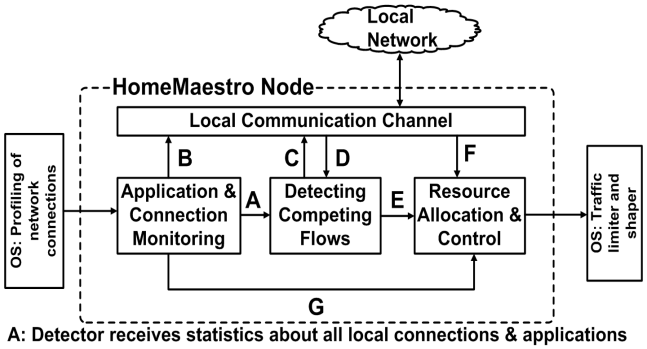
3. HOMEMAESTRO ARCHITECTURE

We now describe the basic architecture of *HomeMaestro*, together with our assumptions and specific design choices.

Our overarching goal is to automatically allocate network resource across hosts and applications in order to meet user's expectations. *HomeMaestro* does not assume assistance from any network equipment, or from the applications and the transport protocol. *HomeMaestro* is targeted at small networks in the order of tens of hosts. Overall, *HomeMaestro* attempts to infer competing flows or applications, and then configure network resources accordingly by applying priorities.

The design of *HomeMaestro* is depicted in Fig. 2. We favored a modular design that decomposes the problem into various modules that could be developed independently of

[1] Another reason may be due to problem over-reporting.



A: Detector receives statistics about all local connections & applications
B: Broadcast statistics of important flows to other HomeMaestro nodes
C: Broadcast the ids of connection that experience problems
D: Detector collects information about remote important connections
E: Detector determines constrained resources, their capacities, and competing connections
F: Collect traffic statistics of remote competing flows (broadcasted in B)
G: Collect traffic characteristics of local competing flows

Figure 2: Architecture of HomeMaestro

one another, and could potentially be modified without affecting the whole solution (e.g., *HomeMaestro* could allow for various inference techniques besides the ones described in the following sections). Application and connection monitoring uses extensive measurements at the end hosts to collect raw data, and summary statistics are occasionally and selectively broadcasted to participating hosts. A detection module then uses time-series analysis to detect applications that compete for the same resource, such as the broadband link, or the wireless network. The detection module also estimates the capacity of the constrained resource. The resource allocation module devises a nominal allocation amongst competing applications using a feedback control loop with real measurements, and enforces these allocations through rate control.

Our approach is reactive. Congestion may temporarily build up and users will suffer some performance degradation for a few seconds; however, we do control the extent of such congestion events, and eventually the system converges to the desired configuration. This design choice was dictated by the lack of explicit signaling from the applications regarding their network requirements, that would enable proactive solutions. Additionally, usage demands and behavior can be quite unpredictable favoring reactive designs. In the following section, we provide a detailed description for each module of our design.

3.1 Modules

Application monitoring. Application and connection monitoring is the first important module of *HomeMaestro*. Measuring at the hosts is attractive for several reasons: It provides simplicity compared to measuring within the network and enables collection of certain data types only accessible at the host. Note that while monitoring statistics in the middle of the network might provide other advantages such as the aggregate view of the home network, we believe that host

participation is essential for the management of the home network. This is because hosts have the necessary context (e.g., application and user specific information), which is difficult to have available in the network. In the ideal scenario, *HomeMaestro* would coordinate with home networking equipment such as routers to exert control. These possibilities are further discussed in Section 6.

Specifically, we first monitor all read and write operations at the socket level to infer network-related activity. Second, we monitor the internal TCP state of all connections[2] and collect extensive measurements, including TCP's estimation of the RTT, the total number of bytes/packets in and out, the number of congestion events, and others. This information is collected at fixed time intervals[3]. Third, we collect other application-specific information such as its process name and the libraries it is using; in principle we could even detect when an application is using devices like the microphone (which may indicate a VoIP application), or displays video (which would hint that the application is streaming video). At its current implementation, *HomeMaestro* matches the process name to a database of well-known applications and determines priorities based on static rules; such rules may be modified by the users.

Detecting Competition for Resources. The detection module uses the collected measurements to identify connections that compete for the same network resource. We first use time-series analysis to detect connections that experience significant changes in their network performance (e.g., throughput drop, or RTT increase). The detection module specifies what we refer to as the *change points*, which reflect the time and type of such changes. This detection process is described in detail in Section 4.1. Change points and a snapshot of the time series shortly after the change are communicated to all *HomeMaestro* participating hosts.

Hosts correlate change points across local or remote connections in order to detect applications that compete for network resources. Section 4.2 describes this inference process in detail. The correlation module assumes some type of weak synchronization (as we operate at second timescales) since cross-host correlations are estimated through time-series analysis. For example, in our current implementation, it is sufficient that *HomeMaestro* hosts are synchronized through NTP (Network Time Protocol) every 10-15 minutes. The output of the correlation module also allows us to estimate the capacity of the constrained resource, for example the rate of the access link. In summary, the correlation module produces sets of competing connections and provide estimates for the capacities of the constrained resources. The detection mechanism is flexible enough to be applied either at the flow or at the application level when deemed appropriate by aggregating all flows for the specific application (e.g., for peer-to-peer file-sharing applications that maintain a large set of connections active). While the question of applying per-application or per-flow control in home networks is interesting, it is outside the scope of this work.

Resource Allocation and Control. The resource allocation module divides the capacity of the resources among the competing connections, according to some predefined priorities, and enforces the allocation by rate limiting the connections. The allocations need to utilize the network resources efficiently and at the same time provide a good experience to the end-users. Moreover, the mechanism should also work well with existing transport protocols, and the congestion control algorithm of TCP. The resource allocation and enforcement details are described in Section 5, where the components form a feedback controller.

Communication channel. The algorithms for detecting change points, correlating connections and deciding the rate allocations executes at each node separately, using summary knowledge about selected connections from all local machines. This design choice is attractive due to its simplicity since no complicated coordination schemes are required; rather, each host reaches the same decision locally, by executing the same algorithms. Such a scheme results in weak coordination, where some hosts may apply policies a few time intervals before others. However, for our purposes this is perfectly acceptable since such time differences are in the order of a few seconds (typically less than 10).

Each host broadcasts information about selected connections, such as their rates, change points, and priorities, to all other nodes in the network. This broadcast communication requires an efficient and reliable communication channel, with modest capacity for control traffic, for timely delivery of the information. Those constraints are generally satisfied in a home environment. We also assume a certain amount of trust between the hosts, since we exchange detailed information about the network connections of each host; moreover, the hosts should trust the information received from the broadcast channel. Such a degree of trust typically exists in small local networks.

3.2 Implementation

We have implemented *HomeMaestro* in the Windows Vista operating system. Our design is generic, and can in principle, be ported to other modern operating systems.

We have used the Extended TCP statistics interface [16], to probe for all TCP connection statistics. Additionally, we use Event Tracing for Windows (ETW[4]) to collect detailed information with regards to the send and receive operations to and from a socket buffer.

The traffic controller of the Windows OS[5] is used to en-

[2]We shall use connection and flow interchangeably through the paper. In both cases, we will refer to the commonly-used 5-tuple of source and destination IPs and Ports and transport protocol.

[3]In our current configuration collection occurs once per second; the time interval is user-specified.

[4]`http://msdn2.microsoft.com/en-us/library/aa468736.aspx`

[5]`http://msdn2.microsoft.com/en-us/library/aa374468(VS.85).aspx`

force rate limits on the connections. The controller implements a token bucket algorithm for traffic limiting and shaping. The token rate is set to the desired traffic rate, and the token bucket size to five to ten times the token rate. (The depth of the token bucket was chosen heuristically, and experiments suggest the exact value was not very important; however, we avoided large values to control queuing delay and bound RTTs.)

We used the Windows Communication Foundation (WCF[6]) to implement the local broadcast channel. This channel is established over a message queuing middleware (MSMQ[7]) that uses reliable multicast (PGM[8]). We have not used any custom transport encodings, and as a result, the size of the broadcast packets is much larger than necessary. Yet, the total overhead traffic observed was less than 40kbps. The use of WCF was chosen to allow flexibility in the design.

A straightforward implementation (in C#) turned out to be reasonably efficient, despite the large amount of collected information and the time-series analysis used to detect change points and correlations. The total memory consumption (both code and data) is typically less than 50MB, and the CPU utilization (of our application) was usually less than 5% on our low-end 2GHz/1.5GB RAM laptop. Memory and CPU requirements are restricted by not monitoring connections that are idle or have very low data rates.

In the following sections due to space limitations, we will focus on the detection and resource allocation modules which present the most interesting challenges of the design, although we also faced intriguing problems during the design and implementation of the other modules. The description and evaluation of both modules will be described simultaneously to facilitate discussion.

4. DETECTING COMPETING FLOWS

This section describes how *HomeMaestro* identifies traffic flows competing for the same resources. First, we focus on detecting candidate flows by identifying performance problems. Then, by sharing and correlating information among the hosts, we identify competing flows and applications across hosts.

Throughout this and the following sections, we present results obtained by analyzing data collected from experiments in realistic home settings. Specifically, Fig. 3 describes the setup that was used for experimentation in one of the authors' home network. While running our experiments, other home network traffic was transmitted over the network at the same time (e.g., browsing, emailing, etc.) to ensure that the collected traces would reflect realistic scenarios. The home was connected to the Internet through a consumer Internet Gateway Device (IGD) (with 3Mbps downstream and

[6] http://msdn2.microsoft.com/en-us/netframework/aa663324.aspx

[7] http://www.microsoft.com/technet/archive/winntas/deploy/confeat/msmqapin.mspx?mfr=true

[8] http://www.faqs.org/rfcs/rfc3208.html

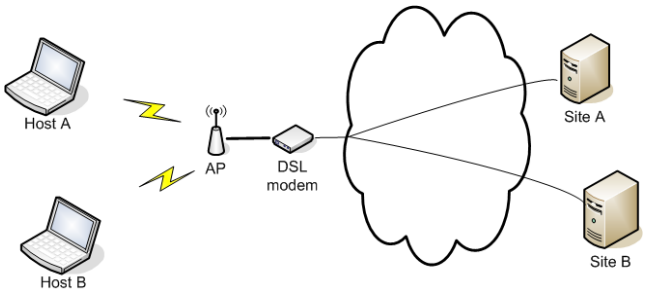


Figure 3: Experimental setup within a real home network. Two *HomeMaestro* instances were running on two laptops that are connected by wireless to the home access point.

800Kbps upstream nominal capacity) comprising a DSL modem, a wireless Access Point (AP), and a router, having common features and configuration such as NAT services, a firewall, etc. We used two laptop computers for our experiments; they connected to the local network over wireless interface. The experiments in this work mostly feature connections to two other hosts in the Internet one in a local institution, the other in another home network (sample RTTs were in the order of 25-30ms). *HomeMaestro* was running in both laptops and messages were transmitted between the two *HomeMaestro* instances over the wireless medium. Overall, our goal was to create an experimental setup as representative as possible when compared to existing configurations in home networks.

4.1 Distinguishing candidate flows

To detect competing traffic flows, *HomeMaestro* first attempts to detect flows that are likely to either experience or cause performance problems.

Candidate flows are identified by detecting *Change Points* (*CP*s), that reflect significant performance change according to some metric. We define three *CP* types: *DOWN, UP*, and *NEW*, to signal the direction in performance change or the arrival of a new significant flow.[9] *CP*s are identified using time-series analysis applied to various monitored connection metrics. We used metrics that relate to application specific network requirements, such as latency and bandwidth. In Section 4.2, we show that the instantaneous rate metric, where the incoming or outgoing observed rate is measured for a predefined time interval, is particularly attractive. Unless otherwise stated, the default measurement interval is set to one second.

Fig. 4 describes in detail how *CP*s are identified. The main idea is to identify base performance using early measurements of the metric of interest, and then detect changes when the instantaneous performance diverges significantly from the base one. Because instantaneous measurements can be quite noisy, both the current and the base performance (currentPerf and basePerf respectively in Fig. 4) are smoothed

[9] A new flow can be defined either as a new flow that is in the top-N flows in terms of bandwidth or through a threshold, e.g., a new flow with at least 5KBps rate.

```
Change Point Detector (currentvalue)
begin
    window = 5 /* in seconds */
    α_fast = 0.2; /* fast moving average*/
    α_slow = 0.005; /* slow moving average*/
    timeseries.add(currentvalue);
    check = window;
    n = timeseries.Count; /* number intervals so far*/
    if n==1 then
        basePerf = currentvalue;return false;
    if n < window then
        basePerf = α_slow * currentvalue +(1- α_slow )*basePerf;
        currentPerf = basePerf; return false;
    else
        currentPerf = α_fast*currentvalue+(1- α_fast)*currentPerf;
        if Abs(currentPer - basePerf) > Threshold then
            check = check - 1;
            if check==0 then
                check = window;
                return true;
        else
            basePerf = α_slow*currentvalue+(1- α_slow)*basePerf;
            check=window; return false;
    return false;
end
```

Figure 4: Detection of Change Points.

using exponential moving averages, the former tracking the latest trends of the time-series, while the latter long-term fluctuations. Once the absolute difference between the two moving averages is above a threshold (which could be defined relative to the base performance, e.g., $0.1 \cdot \text{basedPerf}$), we trigger a *CP* to signal change in performance. To avoid considering short-term performance drops or peaks as *CP*s, we apply a sliding window (enforced by the *check* variable in Fig. 4) before signaling a *CP* event. Thus, only one *CP* exists per window, which also minimizes the communication overhead since *CP*s are communicated to other hosts.

Note that the base performance variable offers context for the flows' long-term performance and does not act as a predictor for the flow's real requirements. Therefore, underestimating or overestimating the base performance is not critical, as we reset this long-term average depending on how the *CP* is handled (Section 4.3).

Fig.5 demonstrates the operation of our detector in one of our experiments. The time-series of interest here is the incoming rate in KBytes per second. Fig.5 shows how the detector doesn't signal events caused by short-term peaks in the first few seconds, and how *CP*s are fired once every window (here equal to 5 seconds), while the rate is significantly different from the base performance.

The detector algorithm in Fig.5 signals changes that are deemed significant. Flows with associated *CP*s are marked as candidate competing flows. However, the detector offers no clues at the cause of the change in performance, which could be the result of changing network conditions somewhere or even just reflect application behavior. The following section describes how *HomeMaestro* identifies changes

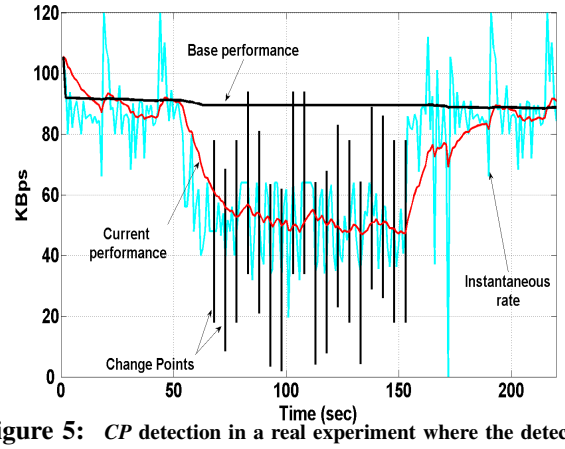


Figure 5: ***CP*** **detection in a real experiment where the detector is applied to the incoming rate.** ***CP*****s are signaled for as long the rate is far from the base performance. The lines highlighting the** ***CP*****s are only shown for visualization purposes.**

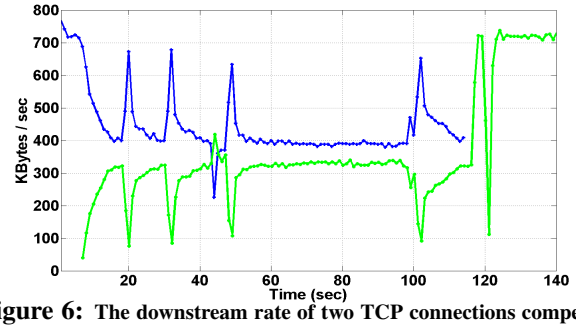


Figure 6: The downstream rate of two TCP connections competing over the access capacity.

caused by competition for network resources.

4.2 Identifying correlated flows

HomeMaestro detects competition for resources by correlating performance metrics across flows and across hosts. Intuitively, all flows competing for a local resource, such as the access link or wireless, should observe performance problems during high contention periods. On the contrary, problems that only affect individual flows likely reflect application behavior or changing WAN conditions.

By way of example, Fig. 6 highlights a sample of two TCP flows competing for the access capacity during one of our experiments. Fig. 6 depicts the incoming rate of two web transfers, one starting a few seconds before the other, and finishing roughly 20 seconds earlier, at which point the second flow manages to get all the spare capacity. As expected, not only does the rate of one flow directly affect the other, but also their behavior appears to exhibit strong negative correlation.

To confirm the findings of this example, we performed the following experiment: Once every hour for two days, the laptops in our home setup would initiate web downloads from two different locations simultaneously, resulting in 48 different sessions. We also experimented with upstream flows by configuring web servers on both laptops and repeating the same experiment as above. In the latter case, clients outside

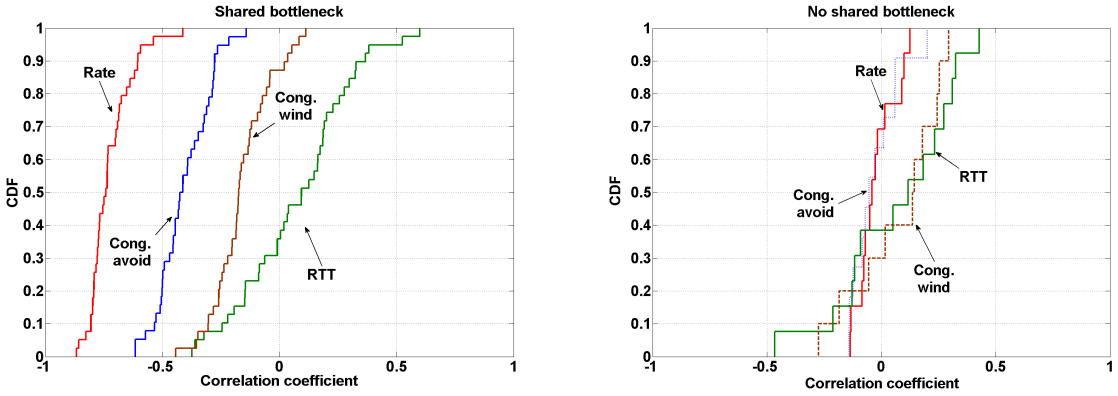


Figure 7: CDFs of the Spearman's coefficient across four metrics for all experiments of the upstream scenario.

the home network would initiate web downloads for files served by the two hosts within our home network. Finally, we artificially limited the server capacity and repeated both experiments; our intention was to emulate scenarios where the two flows would not compete for the same resources, as each TCP flow would be restricted by the server traffic cap.

During these experiments, we configured *HomeMaestro* to log four different metrics in the upstream case (rate, RTT, current congestion window, and congestion avoidance counter,[10] in order to capture different properties of the connection). We also monitored the packet interarrival times and rate for the downstream scenarios. We then used several well-known correlation algorithms to examine the correlation between the various time-series. Specifically, we tested the Pearson's cross-correlation, Spearman's rank correlation coefficient, Kendall's tau coefficient, and Wilcoxon's signed rank test [10]. Overall, we observed that Spearman's rho coefficient produced the most robust set of results. The coefficient is defined as follows:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where n is the number of values, d_i is the difference between each corresponding rank, and ρ ranges from -1 to 1.

Fig. 7 presents the Cumulative Distribution Function (CDF) for Spearman's coefficient across the four metrics, with respect to the 48 experiments of the two upstream scenarios. The plot on the right presents the case where TCP flows are rate limited, and thus not competing for the local resources. Since the TCP flows lasted for several minutes, we applied a sliding window of 30 seconds, for which the correlation was estimated using only data within the window (effectively 30 values), and then averaged the results for each time-series. That is, if an experiment lasted for 300 seconds (5-minutes), we would estimate 270 different coefficients by sliding the 30-second window over the time-series, and finally the end correlation result would be the mean of these 270 values. Thus, the input distributions for the CDFs in Fig. 7, correspond to 48 coefficients, each of which reflects the mean that resulted from the previous process.

We applied this procedure for two reasons: Firstly, we want to detect correlations as soon as possible by collecting a small number of measurements. Secondly, we wanted to evaluate whether the two flows correlate at any point in time, and not only at particular instances of the experiment.

Fig. 7 clearly highlights the differences between the cases where i) flows compete for the same resources, and ii)flows do not compete for the resources but still share part of the same path. In the former case, two of the metrics, the rate and the congestion avoidance counter[11], display significant negative correlations, with the correlation for rate in all experiments being less than -0.4. In the latter case, most metrics are concentrated around zero as expected, showing no significant correlations. RTT in both figures exhibits both positive and negative correlation for certain experiments. We believe that this is caused by flows sharing part of the path, where trends would affect both flows. Similar results hold for the downstream scenarios.

Motivated by these observations, we selected the rate as a distinguishing feature to identify flows competing for the same resource in the home network. An advantage of applying such techniques in home networks is that hosts are typically only a few hops (less than three) away from potential local bottlenecks, and therefore bottleneck effects will not be smoothed out before we have a chance to observe them.

For the receiving scenarios, we also examined techniques that focus on the distribution of packet interarrival times both in the time or the frequency domain (e.g.,[6]). The intuition behind such methodologies is that similar peaks in the histograms of the packet intearrival times would reflect common bottlenecks. We observed that these techniques do indeed produce good results when using a large number of packet arrivals, but constraining the histogram for packet arrivals to small time windows introduces noise and bias. Hence, we selected the rate to be the distinguishing feature for the receiving case as well.

Finally, we examined correlations when three flows were active at the same time. We observed that overall the nega-

[10] defined in [16] as the number of times the congestion window has been increased by the Congestion Avoidance algorithm.

[11] This metric is monotonically increasing and thus correlation is applied to the differenced series.

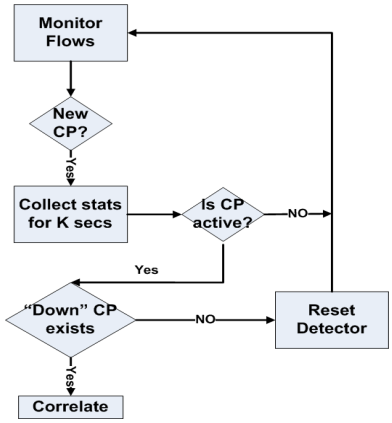


Figure 8: **Competing flows detection flowchart**

tive correlation identification still held when examining flows per pair. However, occasionally flow-pairs could also exhibit strong positive correlations (effectively two flows correlated positively with each other and negatively with the third).

Note that our correlation algorithm might also identify correlated flows that compete for a non-local resource, i.e., correlations in the WAN, should these correlations survive smoothing effects along the path. In *HomeMaestro* (and possibly from the actual users's point of view), these flows are still competing and *HomeMaestro* will assign priorities to such flows.

4.3 Determining competing flows

Fig. 8 outlines our algorithm for identifying competing flows, which combines the ideas presented in the two previous sections. The detection algorithm monitors all flows and identifies the candidate flows through *CP*s. Then, correlation determines whether the candidate flows are really competing for a resource.

Once a *CP* for a flow is detected, *HomeMaestro* collects statistics for a predetermined interval K.[12] Note that we cannot use previously collected statistics for the flow, since we are interested in the interval *after* the problem is detected. If the *CP* is still active after K seconds, we examine whether other *CP*s exist either locally or remotely from other hosts, and the type of these *CP*s.

If no "DOWN" *CP* exists, then we simply reset the detector, so that its base performance is set to the currently observed one. Since "DOWN" *CP*s are the ones that really signal network problems, we ignore cases where only other types of *CP*s exist. Finally, if such a *CP* does exist, we correlate all current *CP*s, and if the correlation score is less than a threshold (Fig. 7 suggests -0.4), we regard these flows as competing.

This procedure (Fig. 8) effectively produces sets of competing flows, e.g., flows {A,B,C} are correlated and thus competing, while C is also competing with D in another set of correlated flows {C,D}. This is attractive since we can identify competition at different points in the network. Hence, *HomeMaestro* can distinguish between two flows competing for the wireless medium from others that compete for the access capacity.

An example highlighting the functionality of the detection and correlation module in *HomeMaestro* is presented in Fig. 9. The figure shows results from a real experiment in a home network, where three upstream flows competed for the access capacity. Two of the flows (the first and third rows in Fig. 9) were initiated at host A, while the third at host B. The dashed lines reflect times where *CP*s were first detected for a flow, while solid lines show correlation events. The letters in each line represent the chronological order of the events as seen in the two hosts.

The arrival of the flow in Host B resulted in two *CP*s. First, a *NEW*-type (point A) *CP*, and then a *DOWN*-type one (point B) from host A. Notice that the *DOWN* event is fired a few seconds later than the noticeable drop in the flow performance, caused by the effect of the smoothing and the window used in the detector. From this point, the two hosts exchange messages with statistics for the corresponding *CP*s, and once enough values have been collected, each host separately evaluates the correlation score, since there exists an active *DOWN CP*. The time difference in the correlation evaluation reflects the distributed nature of *HomeMaestro*, where each host locally evaluates correlations. At point E, the flow at Host B, experiences further performance drops, and another *CP* is fired which however is not handled since no other *CP*s exist in that time interval. Once the third flow is initiated, *CP*s are fired at points F (*NEW*),G (*DOWN*) and I (*DOWN*) and correlation occurs once the sufficient number of statistics is collected.

In summary, the simplicity and local execution of the detection and correlation algorithms allow us to correlate events and identify competing flows across applications and across hosts with a minimal network overhead (see Section 3.2).

5. RESOURCE ALLOCATIONS

When resources are constrained and limit performance, we are faced with an archetypal resource allocation problem. This section discusses how we allocate resources to connections (and effectively applications). We propose a sharing mechanism that is based on weighted proportional fairness [12]. The proposed mechanism respects TCP's congestion control loop, and is provably stable.

For simplicity, we concentrate on a single constrained resource (e.g., the upstream of the Internet access link) which has an (estimated) capacity of $\hat{C}$ and "real" capacity C. *HomeMaestro* estimates the capacity of the constrained resource as a side-effect of the correlation mechanism described in Section 4.2. Intuitively, the sum of the rates of all competing connections provides information on the capacity of the constrained resource. For example, using Fig. 9 we estimate the congestion of the upstream link by adding the instantaneous rates. In practice, instead of using the sum of the current instantaneous rates, a noisy measurement, *HomeMaestro* uses

[12]We use thirty seconds in our experiments.

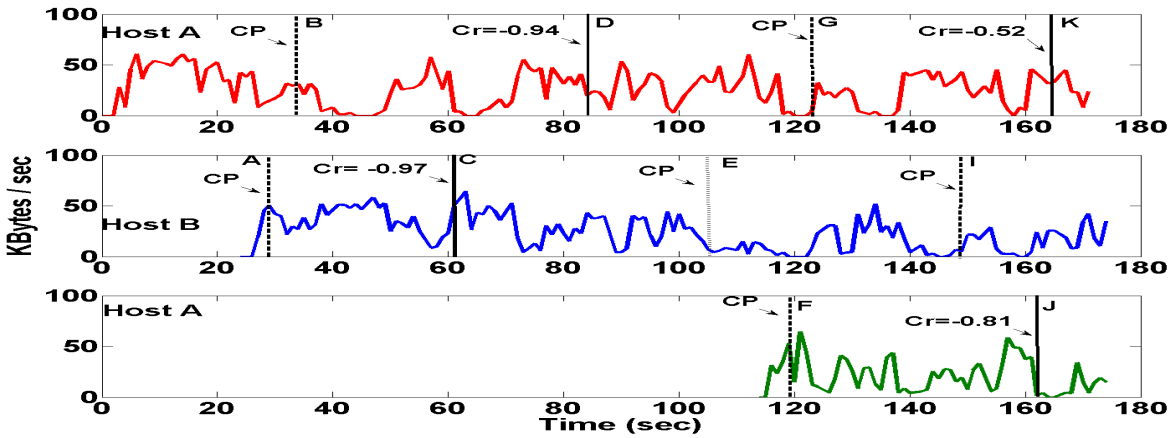


Figure 9: **Detection and correlation events in a real experiment where three flows compete for the upstream access capacity across two hosts.**

the 95^{th} percentile of the observed distribution as the capacity measure. We deliberately use a small underestimation of the total capacity ($\hat{C} = 0.95C$ in expectation) and, thus pay a small utilization penalty to ensure stability of the control algorithm and to minimize queuing delay.

Additionally, we assume that all hosts have information for all connections i that use the resource and their associated rates r_i. The r_i is collected and broadcasted periodically, so the r_i is a delayed estimate. Our goal is to adapt the rates of the connections (through rate limiting) in order to achieve a target resource utilization of individual allocations that satisfies weighted proportional fairness.

5.1 Utilization of the constraint resource

If we knew the rate requirements and the relative importance of different connections, the capacity allocation problem would have been "straightforward." However, predicting bandwidth demand for each connection is difficult without explicit signaling from the applications. As a result, we do the following. Assume that at time t, we have rate limited each connection i to a value $x_i(t)$, and then observe congestion. Notice that a connection may actually use rate $r_i(t) < x_i(t)$. Then, the ratio $\rho = \sum_i r_i(t)/\hat{C(t)} = R(t)/\hat{C(t)}$, is the estimated utilization, where R is the sum of rates. If ρ is below (above) a target objective, then we increase (decrease) the $x_i(t+1)$ (and hence induce changes in the $r_i(t+1)$). The particular mechanism for increasing the x_i, which guarantees weighted proportional fairness, is described in Section 5.2. For now, assume that the change in R, results in a new utilization ρ'. By changing the x_i, we can achieve the target utilization (assuming sufficient demand).

Rate control mechanism. Changing the rate cap $x_i(t)$ in direct proportion to the observed changes in ρ has practical disadvantages, including high oscillations in the assignments and instability, which result in bad performance. Instead, we use a probability of congestion, the so-called marking probability $p = \rho^B$ to control the rate limits. B here is a small constant, typically $B = 5$ in our experiments. If we model the constraint resource as an M/G/1 queue with arrival rate ρ and service rate $\hat{C}$, then the marking probability expresses the probability that an arriving packet finds more than B other packets in the queue. Since $E[\hat{C}] = 0.95$, this is an example of a so-called *Virtual Queue* strategy [9, 14], and is used to signal early warnings of congestion for resource with capacity C. Every *HomeMaestro* node simulates the operation of a *virtual queue* for every constraint resource.[13] We simulate the virtual queue by using the rates of the competing flows to determine the utilization ρ of the resource, and the marking probability p that a queue with the same average ρ would observe.

Our goal is to adapt the x_i to achieve utilization ρ that satisfies the following objective:

$$\rho = \alpha \frac{1-p}{p} = \alpha \frac{1-\rho^B}{\rho^B} \tag{1}$$

where α is a small scaling parameter that gives us flexibility in determining the desired rate (we use $\alpha = 1$ in our experiments). If we view our rate controller as a feedback control system, then the fixed point solution of Eq. 1 expresses the desired stable point for the system. Observe that it is possible that $(1-p)/p > 1$, during periods of underutilization; the response of our rate controller would be to increase the rate caps and in doing so allows more traffic.

The objective expressed in Eq. 1 can be motivated as follows. Assume that the total utility of achieving a rate R from the constrained resource is $U(R) = \alpha' \log(\alpha' + R)$. We assume that the cost of operating the constraint resource at rate R is proportional to the rate of packets that suffer excessive queuing delay, i.e., finding more than B packets in the queue, which we can model as $R \cdot p^B$. Then, the net benefit of using the resource is $U(R) - Rp^B$. The net benefit is maximized when $R = \alpha'(1-p)/p$, or $\rho = \alpha(1-p)/p$, by setting $\alpha' = \alpha\hat{C}$, where $\hat{C}$ is the estimated capacity.

From a practical point of view, Eq. 1 expresses a quantifiable operational target as a function of configuration parameters B and α. For example, when $\alpha = 1$ and $B = 5$,

[13]The flows that compete for the resource and the capacity of the resource are determined by the algorithms of Section 4.1.

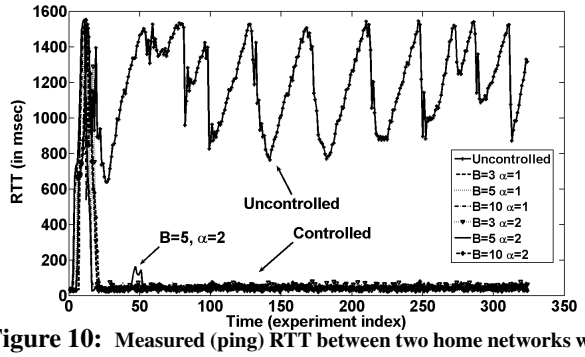


Figure 10: Measured (ping) RTT between two home networks while performing three long web transfers.

the target utilization is $\rho \approx 0.88$, which is a good compromise between efficiency and network responsiveness (i.e., small queuing delay). We can always use an Adaptive Virtual Queue [14] which adapts $\hat{C}$ to increase utilization.

The Virtual Queue affect on RTT. To evaluate the performance of the rate controller with respect to utilization and queue sizes, we performed the following experiment. We initiated three upstream web connections in our home setup, where the estimated uplink capacity was around 800kbps. For each connection, we measured the goodput, i.e. number of bytes sent, and throughput, i.e., total number of bytes sent including retransmissions. We also used ping measurements of RTT delay between the two endpoints, which was in the order of 25msec-30msec before initiating the connections. The same experiment was repeated several times, first without rate-control and then for a number of configurations of the rate controller for various values of B and α.

Fig. 10 presents the measured RTT for a representative run (we observed similar behaviors in multiple runs of the experiment). Without rate control, the accumulated load resulted in large queuing delays (RTTs $\approx$ 1sec), in agreement with previous observations for broadband networks [8]. Enabling rate control resulted in RTT values dropping significantly to around $40 - 50$msec. Since our rate controller is reactive, we also observed high RTT values for a few seconds before our system detects the congestion and enforces the rate caps. After that initial period, the RTT values remained small.

Virtual queue and utilization. The price for controlling the queuing delay at the access link is a small reduction in the utilization of the constrained resource. As Table 1 indicates, this penalty is negligible. Moreover, rate control brings advantages with respect to fairness and efficient use of resources. Table 1 shows that the effective rate (i.e., *Total*, measured in application kBytes/sec) reduced from 76.83kB/s (uncontrolled) to 66.65kB/s $-$ 75.46kB/s, depending on the controller. Examining the ratio of bytes sent over the total bytes (G/T), we observe that rate control reduces the overhead by $\approx$ 10% (goodput increases from 0.78 to 0.84, see ratio T/C). In other words, rate control reduced the amount of capacity spent in retransmissions. Observe in Table 1 that in the uncontrolled case, connection A received a higher rate than the other two connections. This

Table 1: Performance of three web transfers (of equal priority) [in kBytes/sec] and network utilization

	Connection					
Controller	A	B	C	Total	G/T	T/C
Uncontrolled	43.62	15.75	17.46	76.83	0.78	0.97
$B = 3\,\alpha = 1$	26.69	22.35	21.61	66.65	0.84	0.79
$B = 3\,\alpha = 2$	24.03	23.17	22.83	70.02	0.84	0.83
$B = 5\,\alpha = 1$	25.12	23.70	26.64	75.46	0.84	0.87
$B = 5\,\alpha = 2$	25.66	24.14	24.21	74.01	0.84	0.88
$B = 10\,\alpha = 1$	25.82	24.86	24.40	75.08	0.83	0.90
$B = 10\,\alpha = 2$	26.27	24.45	24.92	75.64	0.83	0.91

G/T is the ratio of unique bytes transferred (goodput) over total transfers (including retransmissions). T/C is the ratio of total transfers over (our estimate of) the capacity of the uplink (100kBytes/sec).

unfairness was common in this particular scenario (though not universal, i.e., in some cases TCP allocated the capacity fairly). We never observed such unfairness with rate control.

The ratio T/C of Table 1 is the network utilization ρ of our virtual queue. It is interesting to observe that the measured ρ is close to the theoretical values predicted by Eq. 1. For example, optimal $\rho = 0.881$ for $B = 5$ and $\alpha = 1$; the observed utilization was 0.867.

A larger B increases utilization at the cost of responsiveness. Overall, the particular configuration choices of the virtual queue size B were not very important (with the exception of $B = 3$ that is a bad choice even theoretically). Such robustness is attractive from an engineering point of view, since it negates the need for fine-tuning.

5.2 Allocating resources to connections

Hard priority mechanisms, such as strict priorities or preemptive priorities have several disadvantages, such as possible starvation or unbounded performance for lower priority traffic, and potential priority inversion when there are several resources. Instead, we use relative priorities, that enforce a form of weighted proportional fairness [12]; informally, we associate a weight w_i with each application, and if there is a single resource, then the amount of resource allocated to each application is proportional to w_i. Specifically we want to relate x_i to w_i via

$$x_i = w_i \frac{1-p}{p}. \tag{2}$$

(Eq. 2 is the per-connection equivalent to Eq. 1.)

To achieve the objective utilization of Eq. 1 and rate assignments proportional to w_i, we adapt the rate cap x_i of connection i as follows:

$$x_i(t+1) \leftarrow x_i(t) + \kappa x_i(t) \left[(1 - p(t)) - p(t) \frac{x_i(t)}{w_i} \right] \tag{3}$$

where κ is a gain parameter that determines the rate of convergence, and typically $\kappa \leq 1/(B+1)$, and the time t to $t+1$ is the control interval M. Observe that the w_i should be scaled to satisfy $\sum_i w_i = \alpha C$. We guarantee that x_i is always positive by enforcing a minimum value rate of $x_{min} = 2KB/s$,

$$x_i(t+1) \leftarrow \max\{x_{min}, x_i(t+1)\}. \tag{4}$$

Observe that when the system converges, i.e., $x_i' = x_i$, then $x_i = w_i \frac{1-p}{p}$. If all connections use their allocated capacity, then indeed Eq. 1 is satisfied.

From a utility maximization point of view, we can express the per-connection utility function as ([12]):

$$U_i(x_i) = w_i \log(w_i + x_i)\,. \tag{5}$$

Again, the objective is to maximize user utility minus network cost, which can be modeled as px_i. The marking probability $p = \rho^B$ can be regarded as the price for the resource with capacity $\hat{C}$ and offered load ρ. If there are several resources ℓ with associated marginal "prices" p_l, then p is the aggregate price, given by $p = \sum_\ell p_l$.[14] Modeling competition for multiple resources, e.g., the wireless medium and the access link, is an interesting problem, which we defer to future work.

For a fixed population of users and demands, Lyapunov techniques (see e.g., [22]) show that each x_i converges to its target rate. However, we not know the instantaneous load at the resource; instead, the nodes determine the utilization of the resources every M seconds (typically M=4 seconds which includes measuring the rates of the local connections and broadcasting them). It is straightforward to show (e.g., using control theory techniques, [22]) that the delayed feedback algorithm (Eq. 3) is stable provided that κ is sufficiently small, i.e., provided that

$$\kappa \leq \min\left(\frac{1}{B+1}, \frac{M}{RTT}\right).$$

Given our large observation window (M), we just need to pick $\kappa < 1/(B+1)$.

Evaluation of connection prioritization. To evaluate the performance of the system under Eq. 3, we repeated the same experiment as in Section 5.1, but we configured one of the connections to have three-times the priority relative to the other two (i.e., $w_1 = 3 \cdot w$, $w_2 = w_3 = w$). As before, we used ping to get an estimate of the queuing delay in the access link; again, the queuing delay was small. The main objective of this experiment was to determine whether the rate controller would give a higher rate to the high priority flow, whether the rate caps and effective rates approximated the configured w_i, as well as to examine the utilization and performance of the network.

Fig. 11(A) depicts the rates allocated by the rate controller to the three flows. As expected, the rate caps (x_i) for the high priority flow is roughly three times higher than the rate caps of the other two. The low priority flows received similar rates. Observe that the rate controller spent roughly 20 to 30 seconds converging to the desired rate, and after that initial period the rate remained relatively stable.

The lower three plots in Fig. 11 depict the actual rates (goodputs) received by each of the three flows. In this particular experiment, the low priority Flow B started first and

[14]Note that we sum prices across resources, where prices are derivatives of costs. See [22] for a discussion of aggregating probabilistic signals of prices.

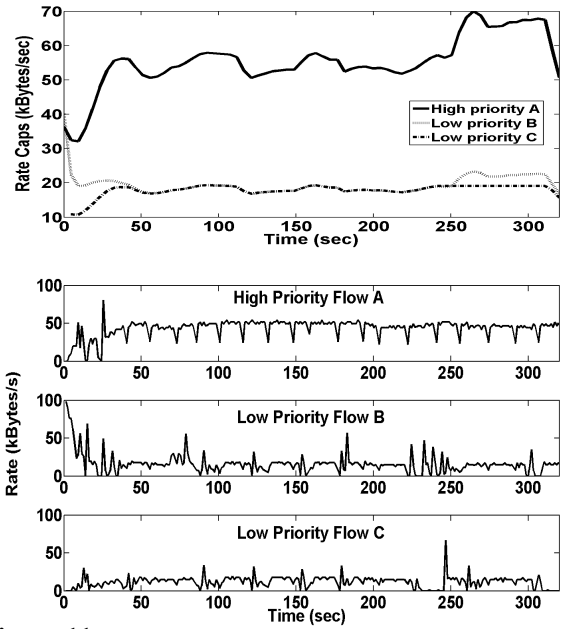
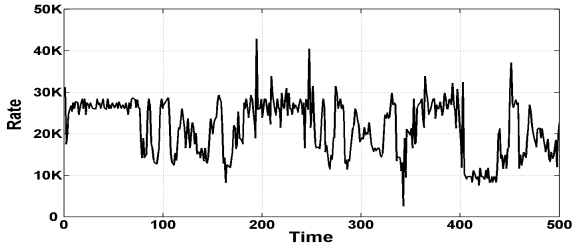


Figure 11: **Rate caps and actual rates (goodputs) of a mix of one high ($w_1 = 3w$) and two low ($w_2 = w_3 = w$) priority flows**

(A) Rate of media encoder without competing traffic (kB/s)



(b) Rates of three web transfers and a high priority media streaming

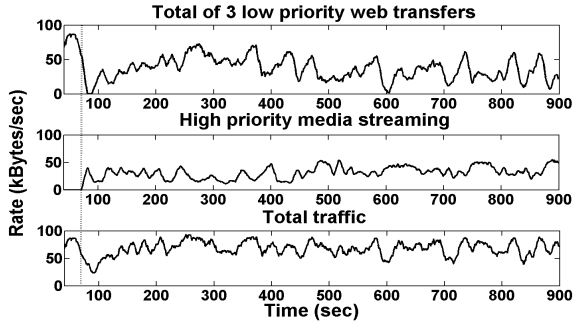


Figure 12: **Rate caps and actual rates (goodputs) of a mix of one high ($w_1 = 3w$) and two low ($w_2 = w_3 = w$) priority flows**

initially used the entire upload capacity. During the observation period, Flow A indeed experienced a better rate than Flows B and C; the actual ratios of rates were 3.8 and 3.6. While the ratios were not equal to the target ratio of 3, they were a close approximation. Observe that this difference was not due to the rate controller, which estimated the correct values for the rate caps. From a practical point of view, the system indeed favored the high-priority connection at the expense of the rest. The fluctuations observed in the figure relate to TCP behavior and not to the limits set by the rate control mechanism. For example, at around $t = 225$sec, we observed a congestion event for the low priority flows in this experiment. Since both of them reduced their rates, capacity was freed for the high priority rate and our controller smoothly adapted the limits accordingly, avoiding further instabilities. When congestion ended, flows returned to their intended rates.

In this particular experiment, we used a virtual queue with $B = 5$ and $\alpha = 1$. Overall, goodput and total rate observed were $G/T = 0.81$ and $T/C = 0.82$. These numbers are somewhat smaller than those of Table 1, which may indicate a small decrease in performance when introducing priorities. We observed a similar trend in other experiments.

We have performed similar experiments, where high priority flows were rate limited to values below their fair share. In such cases, the low priority flows achieved rates higher than their fair-share as it would be desirable in a realistic scenario. Due to space limitations, we omit this set of results and instead describe a more challenging scenario.

Finally, to stress-test our control mechanisms, we tested the performance with highly-variable rate connections, such as the media streaming sessions. Similar to the previous experiment, we have initiated three upstream connections and after a short period we initiated a media streaming session, which produced an average rate of 377kbps. The media server (a Windows Media Encoder) produced a stream of variable bit rate (Fig. 12(A)). The media stream connection was configured with $w_{\text{media}} = 6$, while the web transfers with $w_1 = w_2 = w_3 = 1$. Indeed, during the experiment the video played out at the full rate without any performance problems.[15] During the contention period, the low priority flows received an aggregate rate of $\approx$ 33kB/s. The video received on the average $\approx$ 40kB/s. The overall performance was sufficient to support the video streaming without blocking the web downloads; however, the utilization was smaller than in the case of only web transfers due to the high variability of the media stream. An open problem is to design a rate controller that performs well even with highly variable high-priority connections.

5.3 Setting weights w_i

Setting the weights w_i is a challenging problem that should take into account user priorities and intentions that are very difficult to infer even at the host-level. As an example, in our user study (Section 2), a family member insisted that his work related file transfers were more important than his daughter's YouTube video downloads. Our system allows users to adapt the w_i to express intentions. This however presents a complicated user interaction problem that we defer to future work. Instead, we describe how to set connection weights w_i as functions of the desired connection rates.

The weights w_i can be interpreted as the "willingness-to-pay" [4] of an application. If x_s is in bytes, then w represents the per-byte willingness to pay. It follows from (1) that

$$w_i = x_i \frac{p}{1-p} \tag{6}$$

which implies that w_i should be of the same order as the rate we want to achieve for x_i. To allow the system to be run at full rate, we need to allow $\sum x_i = \alpha \hat{C}$, as described above. One way to set initial weights is to set $w_i = M x_i$, where x_i^{target} is a nominal target rate for the connection, and $M = E[p/(1-p)]$, related to the target utilization via Eq. 1. For example, with $B = 5, \alpha = 1$ this implies $\rho = 0.88$ and $M = 7.4$. In our experiments, we used arbitrary weights. Target rates could be set using history, or application characteristics, and with relative weights adjusted by the user.

6. DISCUSSION

Throughout this paper, we have stressed the problems of home network management and configuration. *HomeMaestro* is a first step towards simplifying these processes by monitoring the performance of applications across hosts and automatically assigning relative priorities in order to improve user experience. However, *HomeMaestro* by no means addresses all the problems of the home network. This study suggests a number of challenging issues; here we discuss a few of them.

Non-compliant devices. The diversity of home devices is one of the most important hurdles to overcome when deploying home solutions. Can we control hosts or devices that are part of the home network, but do not participate in the *HomeMaestro* virtual network? While priorities cannot be enforced for non-compliant devices, our detection algorithm will still identify performance problems and could potentially report these back to the user.

Control could be imposed, if future home networking devices such as home routers or APs, expose APIs for hosts to query for network statistics, and set parameters on a per-flow basis. Then, *HomeMaestro*-enabled hosts could infer the devices causing the problem using techniques similar to the ones presented in the paper, and possibly exert control by using the home-networking device to shape traffic sourced at or destined for non-compliant devices.

Multiple subnets. The number of home networks with multiple subnets grows. Our experience with this study and discussion with vendors suggest that this is already occurring, with most users being unaware that multiple NAT devices introduce multiple subnets in their home network. *Home-*

[15] Notice also that in addition to our rate protection, the media player buffers for 5sec.

Maestro requires an efficient mechanism for machines to discover and communicate with each other, which is currently implemented using multicasting. However, multicast will not cross over multiple subnets without further sophisticated configuration.

UDP traffic. We have not explicitly referred to UDP traffic. While some applications such as gaming do use UDP, the vast majority of home network traffic is still TCP. This was also confirmed by our household study. However, there may still be cases where control of UDP flows is required. Since receiver-side control of UDP is not possible due to the absence of a feedback loop, we will restrict our attention to the sender. If a new UDP flow significantly affects existing traffic, *CP*s will detect the performance degradation and the new UDP flow. The two flows will be correlated, however, our detection may fail to spot the correlation (since UDP does not "fairly" compete for the bandwidth[16]). We can use specific rules to address such issues, and enforce priorities. On the other hand, if UDP flows are already taking a significant amount of the total bandwidth, new TCP flows might not trigger any CPs and thus applying the correct priorities may not be possible.

Delay sensitive applications. By rate controlling the high rate flows, we limit the amount of queuing in the access routers. As a result, most interactive and low-rate applications do not experience long delays and perform as desired. The same is true for media streaming applications, as we demonstrated in Section 5.2. However, we have not explored how medium rate, interactive applications (such as online gaming) would perform. Such applications are becoming increasingly common in home networks and pose their own set of constraints. The goal here is to provide good performance to those applications, without significantly underutilizing the home network.

Soft guarantees or reservation. The current implementation of *HomeMaestro* employs a differentiated services type mechanism to provide soft relative priority guarantees. Future applications may explicitly inform lower transport layers of their networking requirements, thus facilitating reservations or an Integrated Services type solution. The modular design of *HomeMaestro* can incorporate either scheme.

The effect of broadband technologies. Local networking technologies for home networks, such as Wireless and Powerline, as well as access technologies such as Cable or DSL, exhibit diverse network properties and can be very unpredictable [8]. While our experiments covered only DSL technologies, we believe that *HomeMaestro* should be largely unaffected by low-level variability, and that sufficient smoothing of the various metrics should account for such effects. Further testing is needed to confirm this hypothesis.

Prioritization support in lower-layers. There is a recent move for supporting prioritization in local networks especially for wireless (e.g., 802.11e). Certain operating systems and devices already support such services, through 802.11e priority assignments. While such solutions could be of use, it is unclear as to how effective they would be in cases where the bottleneck is in the access link. Moreover, strict priorities bring their own problems, alluded to above. Integrating such mechanisms in *HomeMaestro* is perfectly feasible.

[16]Note however that most high-performance applications that run over UDP, such as gaming, do implement some form of congestion control over UDP. Thus, correlations may still exist.

7. RELATED WORK

To the best of our knowledge, studies of home networks have been extremely limited. Papagiannaki et. al [17] describe the complications that arise in wireless home networking environments. The authors explain in detail how small changes in configurations can have dramatic impact on the performance of the network. Dischinger et al. [8] examine the networking characteristics of various broadband technologies, describing how these technologies can exhibit high delay variation (jitter), with large queues significantly affecting expected performance. *HomeMaestro* takes such problems into account by using smoothing in evaluation of the various metrics, while its virtual queue control mechanism mitigates the unwanted effects created by queuing, and reduces queuing delays, thereby improving the performance.

The work that is closest to our in spirit is the Congestion Manager (CM) module proposed by Balakrishnan et al. [3], however not in the context of home networks. The authors propose CM, a novel framework, for the control of the network congestion from an end-to-end perspective to account for network properties of applications. The CM module requires communication between the application and the CM module using a predefined API to enforce traffic shaping for congestion avoidance. In contrast, *HomeMaestro* does not require any modification of protocols or applications, and is targeted at application monitoring and control in home environments, not end-to-end congestion avoidance. Furthermore, *HomeMaestro* infers competing connections passively without introducing any probing. Congestion control from the end host has also been proposed by Bhandarkar et al. [5], while Anderson et al. [2] propose a novel approach for endpoint congestion control. Again, the scope and goal of such schemes are quite different than ours. Copperative congestion avoidance and "willingness-to-pay" mechanisms have also been explored to reduce queuing delays and provide priorities in [4].

Inferring shared bottlenecks through correlation analysis [19, 13], or available capacity in broadband networks [15] has mostly been based on analysis of delay, loss or idle times involving active probing. Instead, *HomeMaestro* evaluates shared congestion passively by exploiting its close proximity and small hop-count to the bottleneck most of the times. Finally, distributed rate control has been applied in a different context by Raghavan et al. [18] to provision the demands in resources of cloud-based services with a aim of enforcing a global rate limit across multiple sites.

8. CONCLUDING REMARKS

Home networks are increasing in number and complexity; yet, there is a surprising lack of tools for automating their management, and providing an acceptable user experience. We have proposed *HomeMaestro*, a distributed system for the instrumentation and monitoring of home networks. By collecting extensive information and correlating data across hosts, we detect performance problems, and identify their origin. *HomeMaestro* ensures that high-priority connections and effectively applications perform well through prioritization and traffic shaping, without sacrificing resource utilization. We believe that our most important contribution in this work is that we demonstrate the feasibility of automated management mechanisms in the home network. This is achieved by exploiting the rich context available at hosts which allows for sophisticated inference and resource allocation schemes. Our study is only a first step towards understanding and managing the rapidly evolving, modern home network ecosystem; indeed, various challenging questions still remain unanswered in this direction.

Acknowledgements

The authors are thankful to Richard Harper, Abigail Sellen, Tim Regan and the Socio-Digital Systems group at Microsoft Research, Cambridge for their valuable directions and help in organizing the household user study.

9. REFERENCES

[1] Worldwide Consumer Broadband Penetration Sees Rapid Growth but Current Price Strategy Alone is Not Sustainable for Telecom Carriers Says Gartner. http://www.gartner.com/it/page.jsp?id=501276.

[2] T. Anderson, A. Collins, A. Krishnamurthy, and J. Zahorjan. PCP: efficient endpoint congestion control. In *NSDI*, 2006.

[3] H. Balakrishnan, H. S. Rahul, and S. Seshan. An integrated congestion management architecture for internet hosts. In *SIGCOMM*, 1999.

[4] P. Barham. Explicit Congestion Avoidance: Cooperative Mechanisms for Reducing Latency and Proportionally Sharing Bandwidth. Technical Report MSR-TR-2001-100, 2001.

[5] S. Bhandarkar, A. L. N. Reddy, Y. Zhang, and D. Loguinov. Emulating AQM from end hosts. In *SIGCOMM*, 2007.

[6] A. Broido, R. King, E. Nemeth, and k. claffy. Radon spectroscopy of inter-packet delay. In *High Speed Networking (HSN) 2003 Workshop*, 2003.

[7] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. In *ACM SIGCOMM*, 2007.

[8] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing Residential Broadband Networks. In *IMC, AM SIGCOMM Internet Measurement Conference*, 2007.

[9] R. J. Gibbens and F. P. Kelly. Resource pricing and the evolution of congestion control. *Automatica*, 35:1969–1985, 1999.

[10] R. Hogg and A. Craig. *Introduction to Mathematical Statistics*. Prentice Hall, 1995.

[11] In-Stat. Global home networking & broadband CPE outlooks. In-Stat #IN0703433RC, July 2007.

[12] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 1998.

[13] M. Kim, T. Kim, Y. Shin, S. Lam, and E. Powers. A wavelet based approach to detect shared congestion. In *ACM SIGCOMM*, 2004.

[14] S. Kunniyur and R. Srikant. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. In *ACM SIGCOMM*, 2001.

[15] K. Lakshminarayanan, V. Padmanabhan, and J. Padhye. Bandwidth estimation in broadband access networks. In *IMC, ACM/USENIX Internet Measurement Conference*, 2004.

[16] M. Mathis, J. Heffner, and R. Raghunarayan. RFC 4898 - TCP Extended Statistics MIB. `http://www.ietf.org/rfc/rfc4898.txt`.

[17] K. Papagiannaki, M. Yarvisand, and W. S. Conner. Experimental Characterization of Home Wireless Networks and Design Implications. In *Infocom*, 2006.

[18] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren. Cloud control with distributed rate limiting. In *ACM SIGCOMM*, 2007.

[19] D. Rubenstein, J. F. Kurose, and D. F. Towsley. Detecting shared congestion of flows via end-to-end measurement. In *Measurement and Modeling of Computer Systems*, 2000.

[20] K. Scherf. Networks in the home: The global service provider play. Park Associates report, January 2008.

[21] J. Simpson, C. Robert, and G. F. Riley. NETI@home: A Distributed Approach to Collecting End-to-End Network Performance Measurements. In *Passive and Active Measurements Workshop*, 2004.

[22] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkausser, 2004.

[23] H. Yan, D. A. Maltz, T. E. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4D network control plane. In *NSDI*, Cambridge, MA, May 2007.