HomeMaestro: Distributed monitoring and diagnosis of performance anomalies in home networks

Thomas Karagiannis, Christos Gkantsidis, Peter Key
Microsoft Research, Cambridge, UK

Elias Athanasopoulos, Elias Raftopoulos
Foundation for Research and Technology, Hellas

October 2008

Technical Report
MSR-TR-2008-161

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

Abstract

Home networks are comprised of applications running over multiple wired and wireless devices competing for shared network resources. Despite all the devices operating in a single administrative domain in such networks, applications operate independently, and users cannot express or enforce policies. By studying multiple households' network performance at the packet-level correlated with diaries capturing user experiences, we show that the lack of cooperation across applications leads to observable performance problems and associated user frustration.

We describe HomeMaestro, a cooperative host-based system that monitors local and global application performance, and automatically detects contention for network resources. HomeMaestro is designed to manage home and small networks and requires no modification to routers, access points, applications, or protocols. At each host, it transparently monitors per-flow and per-process network usage statistics, such as throughput, RTT, and loss rates. We propose novel algorithms for detecting performance anomalies and application contention using time-series and cross-correlation analysis.

Through the collected household traces, we show that HomeMaestro effectively identifies roughly 95% of the user reported problems. In addition, we evaluate the processing and communication overhead of HomeMaestro, which typically are below 5% CPU utilization and 10kbps signalling traffic per host respectively.

1 Introduction

Home networks connected to the Internet through wired or wireless broadband are ubiquitous. These networks are typically small in scale, with heterogeneous wired and wireless devices supporting a rich variety of applications with different characteristics and requirements such as email, web, peer-to-peer file sharing, voice-over-IP, live video streaming, on-line gaming, media sharing, and teleworking applications. While all devices operate in a single administrative domain, home networks lack the network administrative entity who would clearly proscribe policies. Indeed, each member of the household not only has different requirements and performance expectations from the network, but also conflicting priorities. Thus, prioritizing applications or traffic may be highly subjective, or context and user dependent. For example, a father may believe that running remote desktop is the most important application, while his teenage son may believe that playing online games on his game-console is the most important application.

In order to understand this tension in home networks and identify potential problems and their root causes, we performed a study of typical households. A company experienced in user studies and focus groups selected three representative families, not including IT professionals. For these three households we collected packet-level traces for a period of a week per household, and asked home users to maintain a diary containing information about application usage and potential perceived issues associated with their home network. Post-processing of the packet-level traces allows us to correlate user reported problems with actual measurable events observed in the traces. In all the diaries, it is clear that users often experience periods of great frustration.

Closer examination of the household traces revealed that user reported events coincide with periods when bandwidth resources were insufficient to support every application. Each household was connected to the Internet using commodity broadband ADSL which appeared to be the bottleneck. Problems were also reported during periods were the in-home wireless was highly utilized. Motivated by the type and magnitude of these performance problems we have designed HomeMaestro, a distributed system that allows real-time monitoring and identification of application problems in home networks.

HomeMaestro is a host-based system that enables the instrumentation and monitoring of home networks at the application-level granularity. In particular, HomeMaestro monitors the performance of all network applications at every host, and employs time-series analysis to infer performance problems. Performance is defined using a variety of metrics such as rate or latency. Tracking and sharing this information across hosts allows HomeMaestro to detect whether network problems are related to competing traffic flows across applications or hosts.

Our system is based on the idea that shared performance problems are observable at each host competing for network resources. HomeMaestro uses time-series analysis to track short- and long-term application per-

formance; significant deviations between these two metrics flag performance anomalies. HomeMaestro hosts exchange information about the performance anomalies, and cross-correlate them in order to identify competing flows. We found experimentally that competing flows directly influence each other's performance, with the time-series of the flow rate exhibiting significant negative correlation. HomeMaestro exploits this negative correlation to identify cases of competition. The proposed algorithms are simple enough to allow for real-time identification with minimal CPU and network overhead.

Our design is well-suited for and exploits the properties of home networks, where a small number of trusted devices incur the majority of network usage. HomeMaestro operates at the host and can be deployed without modifying networking protocols, and applications, or networking equipment, such as routers or access point. Hosts are best-placed to identify application performance problems; they have easy access to extensive information about network usage, and to important context such as user ids, process names, and network connections. Modern hosts have also significant processing power available, and can both track performance and implement complex traffic analysis algorithms.

We evaluate HomeMaestro by replaying the packet-level traces collected at the household studies, and show that HomeMaestro was able to identify approximately 95% of user-reported problems. Additionally, HomeMaestro identified cases of problems that were not reported in the diaries, because they were difficult to spot by non-expert users (for example, competition between peer-to-peer applications and file transfers). We further evaluate the system performance of HomeMaestro using micro-bench marks, and show that it is able to successfully identify problems with low operating overhead even with a large number active flows per host (e.g., typically 5%-10% CPU, and a few Kbps signaling traffic for 50 concurrent high-volume flows).

Our contributions can be summarized as follows:

- We present a user study of three typical households in § 2. We collected packet-level traces of all intra- and inter-network traffic and also household diaries reflecting the users perceived experiences.
- We identify performance issues and detect whether these are caused by competing traffic flows using novel time-series and statistical analysis techniques (§ 3).
- We present the design of HomeMaestro, a distributed system for the monitoring and instrumentation of home networks in § 4. We evaluate its performance using the traces collected and with synthetic workloads.

2 Users and the home network

In order to understand the problems and characteristics of home networks, we studied three typical households. In contrast to other types of networks such as Enterprise networks or WANs, studies of real home networks have been limited (e.g., Neti@home [12]). Besides capturing networking data, our study further explores user experiences and perceptions of home networks. In this section, we provide a detailed description of this study and our findings.

Current home networks are typically small, with many having two or three devices linked together. At the other end of the spectrum, in an informal survey of tech-savvy users, we discovered one home network consisting of upwards of thirty connected devices! In view of this variety of home networking environments, we worked with sociology researchers and a company that specializes in user surveys and focus group studies in order to identify typical samples of home networks. The company selected three households in the greater London area, none of which included an IT professional. The first two consisted of four professionals living together, while the third comprised a family of four. All three homes were connected through ADSL broadband to the Internet, and all featured a single subnet, and a wireless AP which provided access to the majority of the devices in the homes (laptops and desktops). In two of the homes, there were also wired devices, such as game consoles and media centers.

The study was performed as follows. We instrumented each of the households to capture all traffic within the network, wireless or wireline. Specifically, our wired monitors captured all intra- and inter-home wired traffic. Additionally, we placed two wireless monitors physically close to the home AP in order to capture packets sourced at or destined for the AP. We believe that this configuration captured the vast majority of wireless packets in the network. We possibly missed some packets due to interfer-

Table 1: Application usage according to the user diaries

	Home 1		Home 2		Home 3	
Application	Instances	Problems (%)	Instances	Problems (%)	Instances	Problems (%)
Streaming	63	18 (75)	37	18 (66.6)	32	8 (50)
Web	166	3 (12.5)	72	7 (26)	67	6 (27.5)
File Transfer	23	1 (4.2)	9	0 (0)	4	2 (12.5)
Email	55	2 (8.3)	35	2 (7.4)	25	0 (0)
P2P	1	0 (75)	12	0 (0)	3	0 (0)

Table 2: General workload characteristics as observed in the packet-level traces

	Hosts	Total flows	Flows/day	Total Bytes	Rate (Kbps)	Max Rate (Mbps)	Daily Activity(hours)
Home 1	1	3,403 K	680 K	5,261 M	218.7	0.44	10.68
	2	38 K	7 K	83 M	25.3	0.39	1.46
	3	40 K	8 K	198 M	98.7	0.67	0.89
	4	98 K	19 K	158 M	37.5	0.72	1.87
	5	31 K	6 K	157 M	92.9	1.47	0.75
Home 2	1	177 K	35K	7,770 M	628.3	3.81	5.49
	2	142 K	28K	2,878 M	338.9	2.52	3.77
	3	183 K	36K	4,130 M	351.8	9.83	5.21
	4	24 K	4K	237 M	160.0	0.62	0.65
Home 3	1	11 K	2.2 K	44 M	0.01	0.04	1.40
	2	2 K	0.4 K	31 M	0.03	0.07	0.35
	3	155 K	53 K	673 M	0.07	0.64	4.27

ence, e.g., packets transmitted by hosts but never received by our monitors or the home AP due to wireless effects. We captured the first 100 bytes of all packets crossing our monitors.

At the same time, users were instructed to keep detailed diaries of specific application usage throughout the study period (a week per household). The users would further comment on their experiences by reporting potential problems they faced, and also their assumed cause of the problem. This was crucial to our study since our goal was to understand and correlate user perception with measurable networking events.

Tables 1 and 2 present general workload characteristics of the traces and the diaries collected at the three households. Table 1 displays application-related information from the diaries coupled with the number of reported problems for each case. Table 2 highlights network-related statistics per host from the packet-level traces. Two characteristics stand out. First, application usage diversity is evident across hosts. For example, in home 1, one heavy user accounted for the majority of the traffic through peer-to-peer, streaming and gaming applications, while the rest of the users mostly browsed the web. On the contrary, traffic and daily activity was balanced across three users in home 2. Second, a variety of streaming applications (Internet radio, video-on-demand, YouTube, etc.) were extremely popular in all homes. In some cases, content was also streamed between two home devices. This is consistent with trends showing growing demand for media networking within the home to make the content accessible from a variety of devices. These streaming applications seem to suffer the most according to user reported problems (Table 1). Such latency sensitive applications suffer even with moderate load, since packet spacing affects their performance, producing effects which are easily observed by the user. Finally, home 3 users observed only a small number of problems, probably due to the limited amount of traffic in the network compared to homes 1 and 2. Traffic in home 3 mostly involved a desktop that was operated by one home user.

Fig. 1 further highlights how reported problems correlated with network usage. The three figures show the aggregate 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 when users reported problems, and

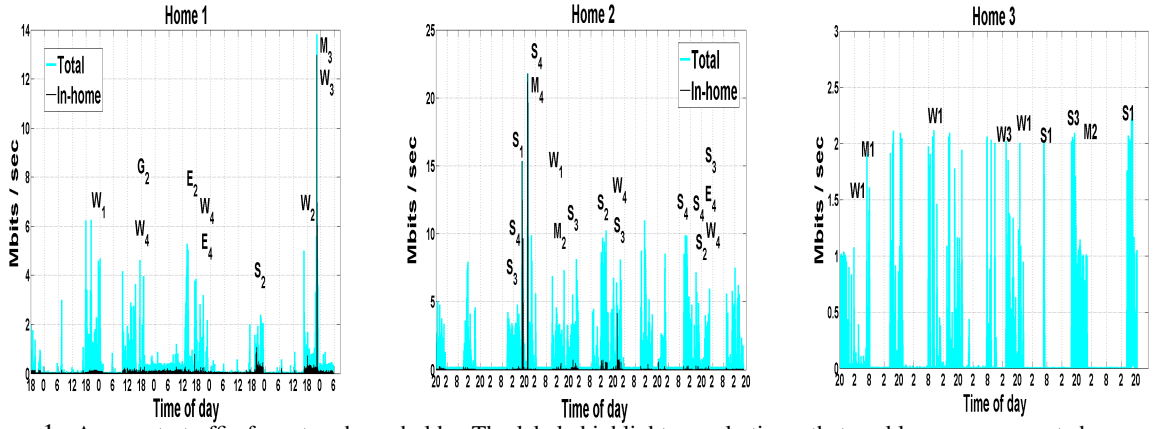

Figure 1: Aggregate traffic from two households. The labels highlight sample times that problems were reported across users (subscripts) and applications (W:WEB, S:Streaming, G:Games, M:Media applications, e.g., image upload/searching, E:email). The occasional dark lines denote "in-home" traffic between two local devices. In all cases, reported problems correlated with increased traffic volumes.

the type of problem (W:WEB, S:Streaming, G:Games, M:Media applications, e.g., media browsing or uploading, E:email). For visualization purposes, only a sample of the reported problems is displayed. Fig. 1 clearly shows that user reported problems correlate with the traffic volume observed in the packet-level traces. Evenings and weekends were particularly problematic, mostly because all users were active simultaneously highlighting *competition across hosts for network resources*. This was also evident from the diaries with several users reporting problems at the same time instances, or commenting that "Internet is in general slow at weekends".

We expected that users face network-related problems; yet, *their number highlights significantly poor user experience*. To further relate these issues to networking events, we examined properties such as TCP retransmissions and flow RTTs. Specifically, we examined how the number of TCP retransmissions and the value of RTT varies between "good" and "bad" periods (i.e, periods when no complaints were reported, versus periods during which problems were reported in the diaries). For the RTT comparison to be meaningful, we aggregated remote IPs (i.e., non-home IPs observed in the trace) in /16 prefixes. This aggregation was essential in order to observe enough flow samples for a particular destination IP space across both periods. We then compared the mean RTT per prefix for the prefixes that presented at least 20 samples both in the "good" and the "bad" periods. Since we were only interested in a rough RTT estimate, RTTs were estimated through the SYN-SYN/ACK packets of the TCP flows.

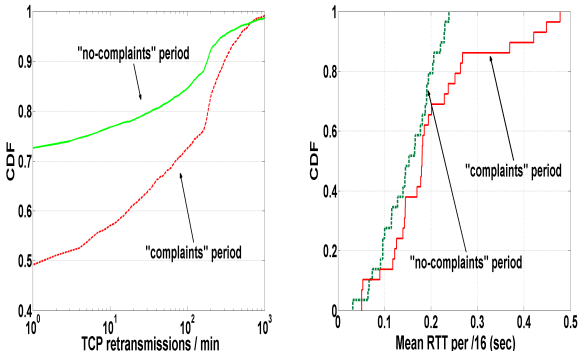

Figure 2: CDFs of TCP retransmissions per minute (left) and RTTs per /16 (right) for good and bad periods according to the user diaries. During periods with reported problems, the number of TCP retransmissions and the RTTs increase significantly.

Fig. 2 presents the Cumulative Distribution Function (CDF) for the number of TCP retransmissions per minute and RTTs per prefix comparing the two periods. Clearly, problematic periods in the diaries correlate also with problematic periods in the network, with TCP retransmissions increasing by more than an order of magnitude (especially for the small values of the x axis). Large number of TCP-retransmissions are also observed for some periods where no issues were reported in the diaries.

We found that these cases relate mostly to high-volume streaming applications at times where only one user was active, or large local file transfers where performance issues may be harder for a non-expert user to spot. Similarly, the RTTs significantly increase for problematic periods, especially for values larger than 200msec.

The vast majority of the traffic for homes 2 and 3 was downstream (86% and 99% respectively), while all traffic in home 2 involved at least one host connected over wireless. Traffic was more balanced in home 1 due to the continuous use of peer-to-peer applications with 50% of the traffic being downstream, 48% upstream and 2% in-home traffic. As shown in previous studies, the home is not a clean environment for WiFi [10], with potential interference from other neighborhood WiFi APs as well as other devices operating in the unlicensed band. This implies that the wireless medium is an additional potential bottleneck with variable capacity, and applications compete for bandwidth. Indeed, examination of wireless characteristics showed high MAC-level retransmissions when multiple hosts were active, implying contention for wireless access.

In-home network traffic is limited but when present creates significant spikes (occasional dark lines in Fig. 1). Indeed, a number of problems were always reported during local file transfers (typically from a wired to a wireless host). We believe that this effect will be further pronounced in the future as communication and networking between home devices increases.

Our study highlights that *user awareness* is substantially limited. Assumed causes for the problems were typically "No idea" in the diaries. Similarly, OS events such as "blue-screens" were attributed to networking. This emphasizes that *management solutions have to be simple, intuitive, and to some extent automatic*.

The results presented over the previous paragraphs emphasize that application performance is a significant, daily problem in today's home networks related to a large extent to applications competing for bandwidth. We believe that despite the small sample of three homes, our findings are representative for a large fraction of today's home networks. We conjecture that even with increasing access speeds, performance issues will persist in view of the growing number of networked devices in the home, and the proliferation of bandwidth-intensive applications. Additionally, *as home networks lack a well-defined management structure, prioritizing applications or traffic may be highly subjective or time, context, and user dependent.* Dynamic policies imply that the effectiveness of centralized approaches, such as applying QoS (Quality of Service) at home routers is questionable, since the necessary context is not available in the network.

Motivated by these observations, we believe that home management solutions should be host-centric with user-centric information presentation and control based on three key ingredients: (i) host monitoring of applications to detect performance problems, (ii) traffic information dissemination by host's sharing information, (iii) semi-automatic host traffic control, with a user interface design displaying information to enable users decide priorities, which are then enforced automatically by the host. The limited size and scope of home networks makes such an approach feasible. In the following sections, we describe HomeMaestro which targets the first two of the three points mentioned above. We leave traffic control for future work.

3 Detecting competing flows

The overarching goal of HomeMaestro is the identification of competing applications, which translates to traffic flows competing for bandwidth in the home network. We define a *flow* as a directed data transfer; a TCP connection has two associated flows, one for each direction.

In this section, we describe in detail the process of detecting the set of competing flows over time. We motivate our algorithms and illustrate their functionality using examples of real experiments performed in our testbed described in Fig. 3. We concretely describe the HomeMaestro system design and the implementation details in the following section (§ 4).

The detection process tackles three main challenges:

1. Detect problematic flows. Since applications do not explicitly signal the OS when they experience performance issues, our detection mechanism needs to identify such applications and the corresponding flows. We call these *candidate* flows, implying that these flows could be facing performance problems. The set of candidate flows is a small subset of all active local host flows.

2. Identify "correlated" flows. Our detector attempts to detect flows that affect each other's performance.

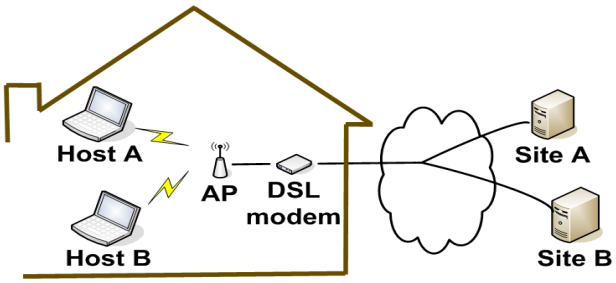

Figure 3: Experimental setup within a home network. Two HomeMaestro enabled laptops connected to the access point with wireless. The nominal capacity of the Internet ADSL connection is 3Mbps downstream and 800kbps upstream.

3. Determine competing flows locally or across hosts by sampling the correlated flows.

In the following subsections, we describe in detail these three processes.

3.1 Detecting candidate flows

HomeMaestro first detects local host flows that either experience or cause performance problems.

Candidate flows are identified by detecting *Change-Points* (*CP*s), defined as points that reflect abrupt and significant performance changes (anomalies), with performance measured by some metric. We define three *CP* types: *DOWN, UP*, and *NEW*, to signal the direction in performance change (down or up), or the arrival of a new flow. HomeMaestro considers as "active," flows that sustain a rate of at least 1kBps. *CP*s are identified using time-series analysis applied to application specific metrics, such as bandwidth and latency. In Section 3.2, we show that the instantaneous rate, where the observed rate is measured over a predefined time interval typically set to 1sec, is a particularly attractive metric. At this time scale we can separate short-term bursts that reflect transient changes in network performance from changes in application performance.

The *CP* detection algorithm as implemented in HomeMaestro is described in Fig. 4. The essence of our algorithm is to detect *CP*s that capture the divergence between long-term and short-term application performance (*basePerf* and *currentPerf* in Fig. 4). Intuitively, divergence should reflect changing network conditions.

We compute these two performance metrics by using low-pass filters to filter out high frequency variation. This

Change Point Detector(flow, value)
input : flow to search for change points
input : the latest performance value for flow
output : TRUE if *CP* detected, else FALSE
constant : α_f = 0.2 // fast moving average
constant : α_s = 0.005 // slow moving average
constant : δ = 0.1 // relative perf change
constant : window = 5
flow related variables (input and output)
basePerf : long term performance of flow
curPerf : short term performance of flow
timeseries : time series of measurements for flow
anomalies : # consecutive anomalies for flow

```
begin
    timeseries.Add(value);
    N ← # of samples in timeseries;
    if N ≤ window then
        basePerf ← basePerf + value / window;
        curPerf ← basePerf;
        return FALSE;
    else
        curPerf ← α_f · value + (1−α_f)·curPerf;
        Threshold ← δ ·basePerf;
        if Abs(curPerf − basePerf ) > Threshold then
        anomalies ++;
        else
            basePerf ← α_s · value + (1-α_s)·basePerf;
            anomalies ← 0;
        if anomalies ==window then
            anomalies ← 0;
            return TRUE;
        else
            return FALSE;
end
```

Figure 4: Detection of Change Points.

is effectively performed through exponential moving averages on the instantaneous measurements. The two metrics differ in the weights used to perform the averaging (α_f and α_s in Fig. 4, with $\alpha_s << \alpha f$). Divergence is detected when the absolute difference of the two metrics is above a threshold relative to the long-term performance for at least a $window$ of consecutive samples. The threshold is set at $0.1 \cdot \mathrm{basedPerf}$, and represents the smallest acceptable magnitude of change to be detected. This value provided satisfactory performance in our experiments.

If there is a level shift in the underlying process of magnitude larger than the threshold, we trigger a *CP*.

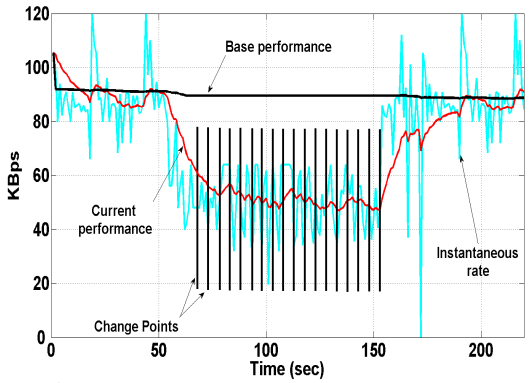

Figure 5: *CP* detection in a real experiment where the detector is applied to the incoming rate. *CP*s are signaled for as long as the rate is far from the base performance. The vertical lines highlighting the *CP*s are purely for visualization.

The threshold detector will trigger after time $window + O(\frac{1}{|\log(1-\alpha_f)|})$, where the second term is the reaction time (under the assumption that $\alpha_s << \alpha_f$). The *window* suppresses short-term performance variations and very transient bursts. Moreover, we generate at most one *CP* per flow per *window* time intervals and hence limit the network overhead, caused when *CP*s are communicated to all other hosts (§ 3.3). The *window* is also used to initialize the detector by averaging measurements over the first samples of the flow ($N \leq window$ case in Fig. 4).

Note that the base performance variable offers context for a flow's 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, since we reset this long-term average depending on how the *CP* is handled (§ 3.3).

Fig. 5 visually demonstrates the operation of our detector during one of our experiments at the testbed shown in Fig. 3. The time-series of interest here is the incoming rate in KBytes per second. Fig. 5 shows that the detector does not signal events caused by short-term peaks in the first few seconds, and that *CP*s are fired once every window (here equal to 5 seconds), while the rate is significantly different from the base performance.

Flows with associated *CP*s are marked as *candidate competing flows*. However, the detector offers no clues as to the cause of the change in performance, which could be the result of changing network conditions or even just reflecting normal application behavior. The following section describes how HomeMaestro identifies changes caused by competition for network resources.

3.2 Identifying correlated flows

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

Competition is detected in HomeMaestro by identifying negative correlations across specific flow metrics. These negative correlations will be especially strong for two TCP flows sharing a bottleneck link before TCP stabilizes, e.g., when the first flow is already active and a second flow starts.

More specifically, consider the case of two TCP flows sharing a bottleneck link of nominal capacity C, with background traffic b, and rates per measurement interval T of $X_1(t), X_2(t)$ and $X_b(t)$. If the bottleneck buffer size is large, or if both TCP flows are limited by their receive window, then no loss occurs and $X_1(t)$ and $X_2(t)$ will be positively correlated. Congestion windows will increase up to receive window sizes, RTTs will increase, hence be positively correlated (caused by queuing delay), and achieved rates X_i converge to constant values (see [8] for steady state analysis).

Once losses occur, provided the bottleneck buffer does not empty, we have $X_1(t)+X_2(t)+X_b(t) = C$ (in expectation). If the number of losses per measurement period is small, and we assume $X_b(t)$ is approximately constant (compared to X_1, X_2) then

$$Cor(X_1, X_2) = -1 \quad , \tag{1}$$

where Cor is the correlation coefficient. This will hold for small buffers. For large buffers, the loss processes will become positively correlated over the order of time T when TCP stabilizes; in this regime, the conditional probability of a loss from stream 1 given loss from stream 2 is high, hence X_1 and X_2 show positive correlation. It is possible to make these arguments rigorous using differential equations for TCP rate evolution [6] with a queuing model for the loss process. However, even for the

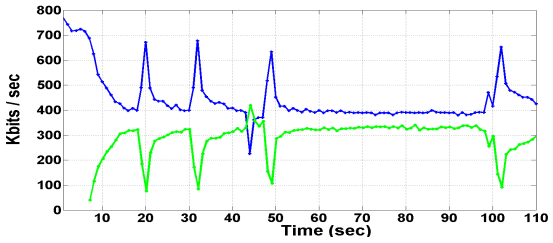

Figure 6: Downstream rate of two competing TCP flows.

large-buffer case, negative correlations will occur should the conditional probability of loss be small, a case which often occurs in practice. Similarly, at the early stages of the lifetimes of the flows, before TCP stabilizes, $X_1(t)$ and $X_2(t)$ will be negatively correlated. This is especially true if the two flows do not start at the same exact moment in time, as it is the case in practice (e.g., the active flow will see its rate decreasing with time as the new flow picks up bandwidth). This is indeed the case that HomeMaestro exploits for the large buffer case.

By way of example, Fig. 6 exactly highlights this scenario, by a sample of two TCP flows competing for the access capacity during two concurrent downloads in the testbed of Fig. 3. Fig. 6 depicts the upstream rate of the two flows for roughly two minutes; one flow started a few seconds before the other. As expected, not only does the rate of one flow directly affect the other, but also their behavior appears to exhibit strong negative correlation, especially during the early seconds of the transfer.

To confirm the intuitive observation of this example, we performed the following experiment: Once every hour for two days, the laptops in our testbed (Fig. 3) initiated Web downloads from two different locations simultaneously, resulting in 48 different sessions (scenario 1). 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 the home network would initiate Web downloads for files served by the two hosts within our home network. We then artificially limited the server capacity and repeated both experiments (scenario 2). Our intention for scenario 2 was to emulate cases where the two flows would not compete for the same resources, since each TCP flow would be restricted by the server traffic cap.

During these experiments, we configured HomeMaestro to log four different metrics per second in the upstream case (rate, RTT, current congestion window, and congestion avoidance counter[1]) 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 cross-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 [4]. Overall, we observed that Spearman's rho coefficient produced the most robust set of results. The coefficient operates on the ranks of the values in the time-series (position of the value once the series is sorted), and 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 $\rho \in [-1, 1]$.

Fig. 7 presents the 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 scenario 2, i.e., the case where TCP flows are rate limited at the servers, 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, 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 used 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 see whether the two flows correlate at *any* point in time, and not just at particular instances.

Fig. 7 clearly highlights the differences between the cases where i) flows compete for the same resources, and

[1]Defined in [9] as the number of times the congestion window has been increased by the Congestion Avoidance algorithm.

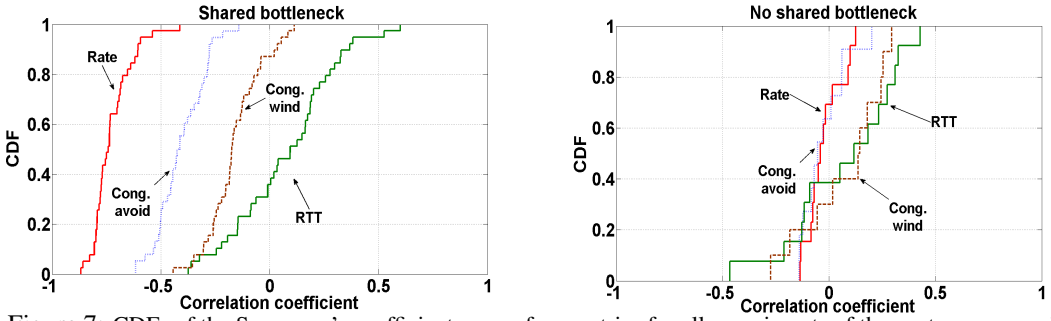

Figure 7: CDFs of the Spearman's coefficient across four metrics for all experiments of the upstream scenario.

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[2], display significant negative correlations, with the correlation for rate in all experiments being less than -0.4. In scenario 2, most metrics are concentrated around zero as expected, showing no significant correlations. Motivated by these observations, *we selected the rate as a distinguishing feature to identify flows competing for the same resource in the home network.*

RTT in both figures exhibits both positive and negative correlation. We believe that this is caused by flows sharing part of the path, where trends would affect both flows. Such positive correlation effects for the WAN have been observed before for RTT or packet loss [7, 11, 5]. There are two main differences between the WAN and our case (leaving aside the fact that different metrics are used): in the WAN, apart from having large buffers, a large number of flows compete, hence $X_b(t)$ is not constant, which can cause $X_1(t)$ to be positively correlated with $X_2(t)$. Second, our monitors are very close to the bottleneck (typically one or two hops away in home networks); indeed we assume here that the limiting bottleneck for the flows is either at the edge or within of the home network. In the WAN, measurements are smoothed by passing several queues before being observed.

For the receiving scenarios, we also considered using the distribution of packet interarrival times both in the time or the frequency domain (e.g.,[1]). The intuition behind such methodologies is that similar peaks in the histograms of the packet interarrival times would reflect common bottlenecks. We observed that these techniques 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 negative 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.

Summarizing, our detector is based on identifying strong negative correlations for candidate flows with the rate as a distinguishing metric.

3.3 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 via *CP*s. These and a small sample of the time-series after the *CP* are shared across hosts. Correlation tests determine 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 of T seconds. We used thirty seconds in our experiments. Note that we cannot use previously collected statistics for the flow, since we are interested in the interval *after* the problem is detected (i.e., during the anomaly). If the *CP* is still active after T seconds, we examine whether other *CP*s exist ei-

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

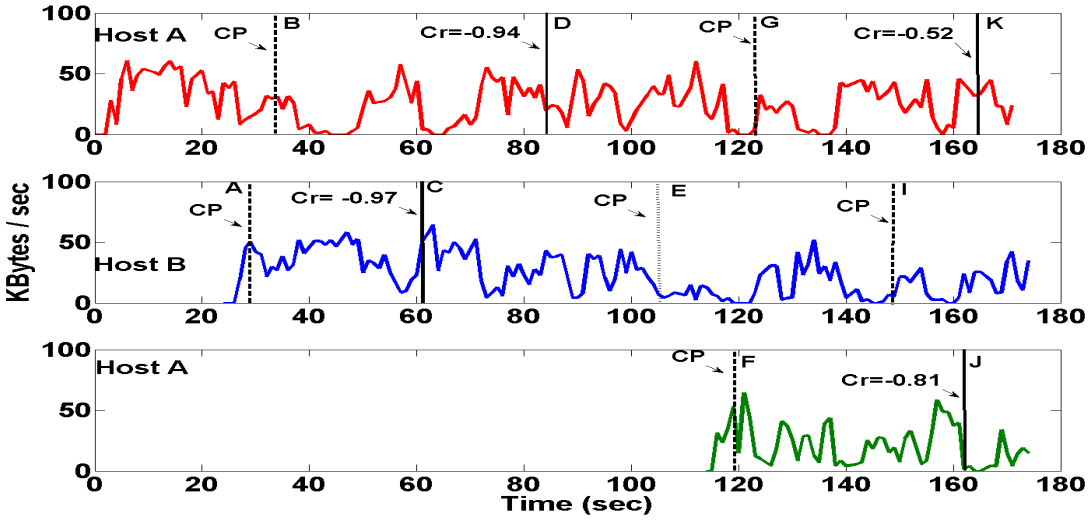

Figure 9: Detection and correlation events in a real experiment where three flows compete for the upstream available bandwidth.

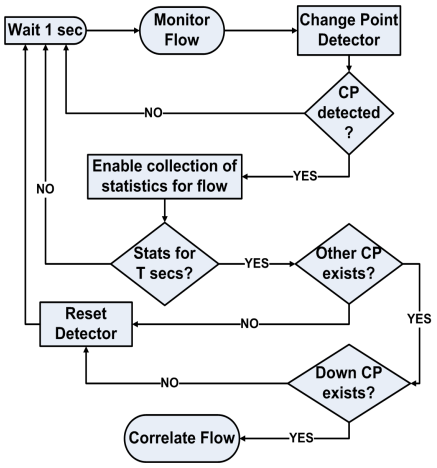

Figure 8: Competing flows detection flowchart

ther 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.

Fig. 9 demonstrates by example the functionality of the detection and correlation modules. The figure shows results from an experiment using the testbed of Fig. 3, 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.

Connection bundles. Certain applications operate using mostly short-lived flows. HomeMaestro heuristically bundles related connections to deal with such cases. This is possible since the HomeMaestro agent has access to host knowledge, such as the process and user ids. For example, we can bundle together and treat as one, all connections that belong to the same process or to processes with the same image name. This is practically infeasible to do inside the network (e.g., at a home router).

Typically, connections related to peer-to-peer applications or Web browsers should be bundled as they refer to the same application. However, care needs to be taken when bundling web connections, since several different applications could be incorrectly treated as one (e.g., web email and YouTube streams). Evaluating the efficiency of the bundling process is an open question. We would like however to stress here that our detection algorithms also perform well for bundles of flows since they operate at multi-second timescales (i.e., smoothing-out short-term TCP bursts, such as short web flows). Overall, *by bundling connections, we significantly reduce the number of active flows fed to the detection algorithms, and we can capture the effect of small flows in the network.*

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 hosts with a small network overhead (§ 5.2).

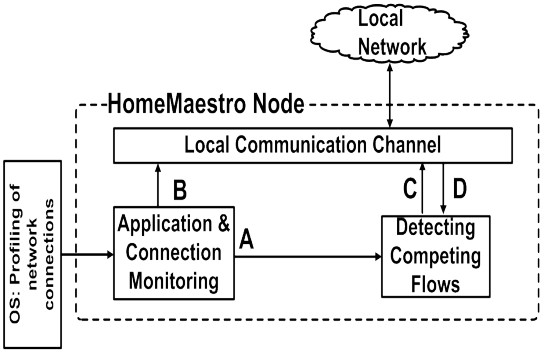

Figure 10: Architecture of HomeMaestro

4 HomeMaestro architecture

The architecture of the HomeMaestro agent is depicted in Fig.10. HomeMaestro uses standard OS interfaces to periodically monitor the performance of all local connections. We use the Extended TCP statistics interface [9] to collect metrics such as total number of bytes and packets uploaded and downloaded per connection, the estimated RTT, number of congestion events, and others. Simultaneously, we track the process that owns each connection and the associated application (i.e., image name of the process); this is useful not only for providing more meaningful information about performance problems, but also in the case of connection bundling. We have also used the Event Tracing for Windows (ETW, [3]) infrastructure to monitor UDP connections. However, due to the limited UDP traffic in our traces, we shall skip the discussion of this part of the system.

Additionally, we monitor other process characteristics, such as the user-id of the owner of the process, or the libraries and peripherals used by the process, in order to infer the purpose of the connection. For example, processes using the microphone may have VoIP connections. Observe that neither the bundling of connections per process and application, nor application inference can be easily performed inside the network. Indeed, this limitation was evident to us when trying to simulate the operation of HomeMaestro in the household traces (e.g., we were unable to bundle peer-to-peer connections together).

The collected measurements are then fed to the detection module that (a) identifies flows that experience performance changes, and (b) infers competing flows. Even though the detection of candidate flows and the generation of *CP*s uses local information only, the identification of competing flows needs information from all hosts. Indeed, as observed in § 2, flows typically compete across hosts which emphasizes the need for a distributed design.

The correlation module assumes some type of weak synchronization across hosts (because we operate at second timescales) since correlations across hosts are estimated through time-series analysis. We have found that synchronization every 10-15 minutes using the Network Time Protocol (NTP) was sufficient for our purposes.

The local communication module broadcasts the *CP*s (step C of Fig.10) and other statistics (step B of Fig.10) for the candidate flows to all other nodes. Further, it collects statistics broadcasted by remote HomeMaestro agents and forwards them to the detection engine (step D of Fig.10). Hence, the detection engines operates as if all the required statistics were locally available. As we shall see in § 5, the network overhead is relatively small, since the *CP*s are compact representations of the important characteristics of a flow. Even though the amount of communication is small, it still needs to be communicated reliably and timely to the other hosts in the network. To achieve this, we use reliable multicasting (PGM [13]) in HomeMaestro. Due to the low volume of traffic, we did not experience any network performance problems, even in the case of wireless networks, where multicast uses a very low transmission rate (1Mbps).

We have implemented HomeMaestro in the Windows Vista operating system. In principle, our design can be easily adapted for other modern operating systems. A straightforward implementation (of $\leq$5K lines of code in C#) turned out to be reasonably efficient. 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 a Pentium 4 (3.20GHz with 3GB RAM). We evaluate in detail CPU requirements and other overhead in the following section.

5 Evaluation

We evaluate HomeMaestro's performance across two main dimensions. First, we examine the ability of the proposed algorithms to identify user reported problems by applying HomeMaestro to the packet-level traces. Second, we examine the processing and network overhead incurred by HomeMaestro.

5.1 Household data

To evaluate the effectiveness of our detection algorithms, we replayed the household packet-level traces and instrumented an "off-line" version of HomeMaestro. In this version, all HomeMaestro system modules remain unchanged with the exception of the monitoring module. This was replaced with a pcap input module that allows us to directly process the trace files. We then examined whether the problems reported in the diaries were identified through negative correlation events (section 3.2). Further, we examined whether the correlated flows in the traces reflect the applications actually reported by the users through whois lookups of the non-local IPs. From the set of problems reported in the diaries, we removed issues unrelated to performance, such as "blue-screens" or "problem detecting wireless signal".

HomeMaestro detects approximately 85% of the problems repoted by the users. Specifically, HomeMaestro identifies 29 out of the 34 reported problems for the households shown in Fig. 1. Table 3 breaks down user reported problems per applications for household 2, and also displays competing applications as identified by HomeMaestro. The first column describes the problematic application according to the diaries, while the rows display competitions according to HomeMaestro. Looking at the individual correlated flows further reveals that indeed they correspond to the applications and even websites reported by the users (e.g., Hotmail, YouTube). For the remaining 15% of problems that went undetected, we discovered that these occurred with applications that were the only ones active at that time in our traces (i.e., no competition could occur). Hence these reports either relate to users "over-reporting" ,which is common in such user studies, or were related to the WAN. After manually examining all undetected reported problems, we were able to identify only two occurrences of valid competition. These involved competition between Hotmail and peer-to-peer flows in household 1. Thus, HomeMaestro appears to identify the vast majority of competition-related problems that were observable by the users.

Table 3: Problems detected by HomeMaestro and instances of competitions across applications. The first column describes applications for which users reported problems. Rows represent the competing applications as shown by HomeMaestro.

App.	stream.	web	file	email	updates
stream.	14	1	1	-	2
web	1	-	-	-	2
file	-	-	-	1	-
email	-	-	1	-	1

The vast majority of the problems detected were due to application competition across hosts. Especially in household 2 where most users were active simultaneously, 92% of the problems were between two or more hosts, while the rest constituted application competition at the same host (e.g., streaming and web applications). This implies that purely local mechanisms, which for example detect only local, per-host competition across applications, cannot detect the majority of the problems. HomeMaestro further identified several instances where competition existed between three or more flows at the same time. Indeed, for home 1, at least three flows were negatively correlated with each other at the same time for 70% of the correlation instances due to the heavy usage of peer-to-peer applications. For home 2, this is true for 13% of the correlations.

HomeMaestro flagged several correlations at times when no user reports existed in the diaries. This was mostly true for household 1, where one heavy user was running peer-to-peer applications and was performing several remote and local file transfers during the period of the study. Most correlations were thus related to competition between file transfer and peer-to-peer applications for which the user did not report any issues. Examining all such cases showed that they indeed correlated with high utilization of the home network (e.g., last day of home 1 in Fig. 1), and appeared to be valid competing flows. Performance issues in such applications are not as evident, and perhaps this lead to no reported problems.

In household 2, HomeMaestro flagged correlations for update services for a variety of applications. Such update services typically run when the users were idle, thus no problems were reported. As with file transfers, observing performance issues for update services as a user is difficult. These update services competed with other applications across hosts resulting in only one of the users reporting issues. This is a particularly interesting scenario as "cloud" back-up services are increasingly introduced for home users. These services mostly operate during user idle time at the specific host trying to be transparent to the user. However, such services have no way of tracking network usage across hosts in the home network, and being bandwidth intensive they could cause applications at other hosts to under-perform.

Summarizing, HomeMaestro was able to identify all but two user reported problems that related to application competition for network resources. Additionally, we were able to identify several cases where no user reports existed, but applications indeed appeared to be competing for bandwidth. Exploiting this information could be invaluable to the user, as HomeMaestro can readily provide root-cause analysis for an issue (e.g., a specific application at a remote host). This is feasible as our distributed design exploits both the local host context and at the same time has global knowledge of relevant information across hosts.

Host vs. in-network monitoring. The evaluation presented in this section focused on traces collected inside the home network, i.e., as if HomeMaestro was hosted in the home gateway; our study was limited by the fact that we were not allowed to install our agent on the hosts of the study. It is natural, however, to question whether such analysis should indeed take place inside the network, instead of the hosts. A network-centric solution is definitely possible, even advantageous in the presence of uncooperative hosts; however, it has also several shortcomings. First, bundling connections per process inside the network is rather challenging. We suspect that this limitation was the main reason of missing the two reported problems mentioned above. Second, filtering low-rate connections is rather challenging to perform in low-cost home networking equipment, since it requires per-connection state; on the other hand, such information is readily available at the hosts. Third, a centralized solution will miss any host-to-host local traffic not crossing the specific network equipment (e.g., in-home traffic).

5.2 System overhead

HomeMaestro runs as a background process at each host. It collects statistics about all local network flows, de-

tects changes in the network performance per flow or application, and communicates with the other HomeMaestro agents. Hence, our system incurs storage, processing, and network overheads respectively, which we evaluate below.

Processing overhead. The most expensive processing overhead of HomeMaestro is due to the algorithms described in § 3. In the worst case, those algorithms scale as the square of the number of active flows, since we may need to pairwise correlate every active flow.

To evaluate the processing overhead, we have used both the data collected from the users, and a synthetic benchmark. To stress the detection algorithms, we disabled connection bundling (§ 3) for the results reported henceforth. Specifically, we isolated the busiest 20-minute time interval in our traces (in terms of active flows) and recorded the start and end times of all TCP connections. Then, we started bulk file transfers from well-provisioned servers, respecting the inter-arrival times of the connections in the trace and their durations. As we do not follow the actual rates seen in the traces but rather all our transfers are high-rate flows, the contention is significantly more severe compared to the original trace. The goal is to study the processing overhead using the connection arrival and departure process seen in the busiest period of the trace. We have used one host to perform this experiment; in the original trace, there were three active hosts at the same time, thus using a single host further stresses the detection algorithms.

According to this trace, the peak number of active connections is 28 and typically, there are 15-25 active connections simultaneously. For this workload, we measured the processing overhead as the CPU utilization of the HomeMaestro process as reported by the operating system on a Pentium 4 with 3.2GHz/3GB RAM. We measure the utilization every 500ms, and the CDF of the entire distribution is given in Fig. 11. The average and median utilization are 1.5% and 1% respectively.

To further study the scalability of the detection algorithms, we examine the following scenarios. We simulate a dynamic environment of N concurrent bulk file transfers; for each, we assign a lifetime following an exponential distribution with a mean of R seconds. We then run 20-minute experiments with N=20, 50, and 100 and R =10sec, 30sec, 1min, and 5min, collecting statistics of CPU utilization every 500ms. We do not try to simulate "realistic" workloads here, but rather stress-test our detection engine.

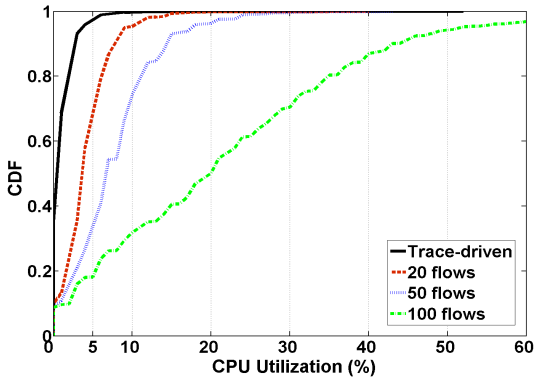

Figure 11: CPU overhead of HomeMaestrofor the busiest 20-minute interval of the household traces, and for three synthetic workloads. The overhead appears negligible for the real trace workload, and acceptable for 20 or 50 concurrent flows.

As discussed, the choice of N has a significant impact on the processing overhead. Fig. 11 shows the CDF of CPU utilization for R=30sec for all values of N, as well as the trace driven simulation. The average CPU utilization is 4.8%, 8.2%, and 22% for N=20, 50, and 100, respectively. The results for the other values of R are similar, since in all experiments all flows were continuously competing with each other (i.e., the flow arrival/departure process is not as significant as the number of active flows). For N=20 and 50, Fig. 11 suggests that the CPU overhead is below 10% most of the time. Even though there are many ways to reduce the overhead and our implementation is hardly optimized, we believe that the current performance is quite acceptable. In practice, we expect that each host rarely has more than 20 "active" connections, a number which connection bundling significantly reduces. The processing overhead is rather large for N=100. It is an open question whether we can scale our detection algorithms for larger networks, i.e., make it linear to the number of active connections; however, we believe that our approach performs well for a typical host of a home network.

As a final note, we have found that the overhead of periodically polling (every second in HomeMaestro) the OS for connection statistics is negligible. This is even with hundreds of connections running in the system.

Network overhead. The network overhead is dominated by the broadcasting of change points and associated traffic information. When we broadcast a *CP*, the message contains a short description of the flow (currently 16 bytes), and, more importantly the measurements of the metric of interest collected in the last 30 seconds (in total roughly 500 bytes). The latter are used to perform the correlation of flows across hosts. While detecting *CP*s for a connection, we also broadcast new measurement updates for that connection. Each host collects these updates for all flows of interest and broadcasts every second (e.g., 1-2 update messages when up to 30 connections are of interest). In summary, the network overhead is determined by the number of connections that generate *CP*s, plus some additional update traffic. Hence, the network overhead both in terms of messages and bytes is a function of the number of *CP*s per second.

We used the traces to simulate the generation of *CP*s, and to estimate the network overhead. Most of the time, HomeMaestro did not generate *CP*s (89% for Home 1 and 95% of Home 2). We ignore these periods, and instead study only intervals that generated at least one *CP*; this gives an upper bound on the network overhead. The average number of *CP*s was 2-3 per second (conditioning on observing at least one *CP*). In the busiest interval, HomeMaestro signaled 41 *CP*s. Even in this extreme case, the network overhead is rather small and can be easily handled by home networks — less than 50 messages per second and around 200kbps. In the typical case, the network overhead is negligible.

We have also measured the network overhead in our benchmark experiments. As expected, the network overhead was typically much less than 50-100kbps, even with 100 concurrent bulk downloads.

Memory overhead. One concern in the design of our system is the amount of memory required to store information about the past performance of all connections. Recall that we collect those statistics every second for active flows, information which is used by the correlation algorithms. Our current implementation stores around 40 bytes per connection per second (timing information, total bytes and packets incoming and outgoing) plus some extra bytes for maintaining internal data structures. The amortized memory overhead for storing connection statistics is a few tens of KBytes per connection, or typically less than a few tens of MBytes per host.

6 Discussion

HomeMaestro is a first step toward simplifying the detection and reporting of performance problems. However, HomeMaestro by no means addresses all such problems in the home network. We discuss the most important of these open issues here.

Non-compliant devices. The diversity of home devices is one of the most important hurdles to overcome when deploying home solutions. Can we monitor hosts or devices that are part of the home network, but do not participate in the HomeMaestro virtual network? Our detection algorithm could still work independently in all participating hosts. However, correlation across non-participating devices would not be feasible. Our algorithms could be enabled, if future home networking equipment such as home routers expose APIs for hosts to query for network statistics. Then, HomeMaestro-enabled devices could infer competition with techniques similar to the ones presented in the paper, although limitations still exist (e.g., inferring application types).

UDP traffic. UDP typically amounts to only a small fraction of the total traffic. This was also confirmed by our household study. If a new UDP flow significantly affects existing TCP traffic, *CP*s will detect the performance degradation of TCP and the new UDP flow. The two flows will be competing, however, our detector may fail to spot the correlation, since UDP does not "fairly" compete for bandwidth. However, 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.

Privacy. As HomeMaestro broadcasts signaling traffic, every home user can easily discover the network activity of every other user. This is especially true since we envisage a system that notifies a user that her applications are not performing well due to excessive traffic from another user. This assumes that users are willing to exchange or broadcast such information. Such an assumption we believe is acceptable in home networks (actually, the parents in our study found this feature very desirable!).

Prioritizing applications. Detecting performance problems is the first step toward a home management solution. As discussed in § 2, we envision that users will be included in the loop by providing preferences over applications and then a distributed system will automatically

configure the allocation of network resources, typically by rate limiting the least important flows during periods of contention. We believe that rate limiting should take place also at the end-hosts, possibly in cooperation with network equipment for non-participating devices. Rate-limiting at the host is feasible in modern operating systems, and allows for complex policies by exploiting the rich context available at the host, e.g., per-application, or per user policy enforcement.

Including the user in the loop implies that users would be able to dynamically configure priorities. Presenting such information to the user in an intuitive way is by itself a challenging problem. During interview sessions, users argued that priorities relate more to the task at hand (e.g. a business call, web surfing for school homeworks) than to the application itself. In such a system, the interaction with the user could be fairly regular, but it should also be meaningful and non-intrusive.

7 Related work

To the best of our knowledge, studies of home networks have been extremely limited. Papagiannaki et. al [10] 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. Neti@home[12] monitors aggregate statistics of performance metrics at the host similarly to HomeMaestro. However, the focus of Neti@home is targeted more toward collecting data for general research on Internet performance as seen from the edge of the network. Dischinger et al. [2] 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 the evaluation of the various metrics.

Inferring shared bottlenecks through correlation [11, 7] has mostly been based on analysis of delay, or loss involving active probing. In contrast, HomeMaestro evaluates shared congestion passively by exploiting its close proximity and small hop-count to the bottleneck. Passive evaluation of shared bottlenecks has also been examined through clustering of inter-packet spacings by Katabi et al. [5]. This methodology is based on minimizing the entropy of packet spacings across pairs of flows from different sources as these are observed at the same receiver. Instead as discussed in § 3.2, we attempt to identify shared bottlenecks with only a few measurements across different receiving hosts, whereas inter-packet timings show acceptable performance only after a significant number of samples has been collected.

8 Concluding remarks

Home networks are ubiquitous and their complexity is growing. Yet, there is a surprising lack of tools for automating their management, and providing an acceptable user experience. We have introduced HomeMaestro, a distributed system for monitoring and diagnosis of performance anomalies in home networks. By collecting information at the host and correlating data across hosts, we detect performance problems, and identify whether they are related to competition for network resources. This is achieved by exploiting the rich context available at hosts which allows for sophisticated inference and passive detection of shared problems. We believe that HomeMaestro is a step toward providing transparent management mechanisms in the evolving home network ecosystem.

References

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

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

[3] Event Tracing for Windows. `http://msdn2.microsoft.com/en-us/library/aa468736.aspx`.

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

[5] D. Katabi, I. Bazzi, and X. Yang. A Passive Approach for Detecting Shared Bottlenecks. *ICCCN*, 2001.

[6] F. Kelly. Mathematical modelling of the Internet. In *Mathematics Unlimited - 2001 and Beyond*, pages 685–702. Springer-Verlag, Berlin, 2001.

[7] M. Kim, T. Kim, . Shin, S. S. Lam, and E. J. Powers. A wavelet-based approach to detect shared congestion. In *ACM SIGCOMM*, pages 293–306, 2004.

[8] L. Massoulié and J. Roberts. Bandwidth sharing: objectives and algorithms. *IEEE/ACM Trans. Netw.*, 10(3):320–328, 2002.

[9] M. Mathis, J. Heffner, and R. Raghunarayan. RFC 4898 - TCP Extended Statistics MIB.

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

[11] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Trans. Netw.*, 10(3):381–395, 2002.

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

[13] T. Speakman et al. RFC 3208 - PGM Reliable Transport Protocol Specification.