Tracking application network performance in Home Gateways

Ahlem Reggani, Fabian Schneider, Renata Teixeira

To cite this version:

Ahlem Reggani, Fabian Schneider, Renata Teixeira. Tracking application network performance in Home Gateways. 4th International Workshop on TRaffic Analysis and Classification (TRAC 2013), Jul 2013, Cagliari, Italy. pp.1150-1155, 10.1109/IWCMC.2013.6583719 . hal-00825277

HAL Id: hal-00825277

https://hal.science/hal-00825277

Submitted on 23 May 2013

Tracking application network performance in Home Gateways

Ahlem Reggani
UPMC Sorbonne Universités and CNRS,
LIP6, Paris, France
ahlem.reggani@lip6.fr

Fabian Schneider
NEC Laboratories Europe,
Heidelberg, Germany
fabian@ieee.org

Renata Teixeira
UPMC Sorbonne Universités and CNRS,
LIP6, Paris, France
renata.teixeira@lip6.fr

Abstract—**Home gateways offer Internet connectivity for all devices in the home, allowing services such as telephony or gaming. However, typical home gateways do not include any mechanism to guarantee optimal performance when applications are competing for the same resources. In this paper we outline an application performance optimization approach for home networks. In particular we study the feasibility of application performance tracking on home gateways, which involves both identification of active applications and monitoring their performance. Our results show that although the home gateway has limited resources, it still has the capacity to do more than just forwarding packets. It can collect and export all the information needed to perform our application performance optimization.**

Keywords—Network performance diagnosis, Home gateway, passive measurements

I. Introduction

With the spread of broadband Internet access [11], more and more people have Internet at home. Various services allow users in a household to perform professional and personal tasks. But running different network services simultaneously can lead to performance degradation. Home users face many performance problems due to various reasons [9], [21]. For instance, a kid downloading a big file can disturb the quality of his parents conference call over Skype. Although, the download only marginally suffers from the minimal bandwidth requirements of the Voice over IP call, the latter requires low latency, which is negatively affected by the download if both share a single queue. The problem is twofold: First, most home users only have limited technical skills and thus have no understanding of performance degradation reasons. Second, even if these skills are present contemporary home network devices offer almost no options to prioritize traffic and identify and resolve resource conflicts.

As for the previous example, when network performance degrades users can only wonder why their conference call quality is bad? Are there competing applications? Is it the router? Or the ISP? Maybe the VoIP server is overloaded …In this paper we want to propose a new approach to help users in such situations by tracking home network application performance. Our solution leverages the home gateway as the tracking point. Not only is all home network traffic passing through the home gateway. It is also the ideal point to tell apart the traffic from different user devices and network services as well as to distinguish between internal and external problems (home/ISP).

Home performance tracking consists of two main parts, identifying active applications and monitoring their performance. The first step is application identification, where we analyze traffic to identify the set of active applications. The second, performance monitoring, is about extracting performance metrics from the flows belonging to each individual application. Applications have different requirements on the network. If we consider the previous example, using the performance tracking metrics we can evaluate the actual bandwidth used for download and the impact on the latency required for the Skype conference call. Thus, we can optimize these two parameters to ensure that the quality of the Skype call will not degrade by enforcing a maximum delay for its packets. Based on the application type and the performance metrics an optimal network resource utilization can be determined and enforced. The resource requirements of applications can either be manually configured or learned from past measurements (e.g. when an application was the sole active application at a time).

Solutions for application performance optimization exists but only based on end-hosts [6], [16]. Performance optimization in home networks however requires a complete view of all home network traffic. This can either be achieved by placing the monitor on the home gateway or by putting a monitor on each end-host. With the increasing number (tablets, smartphones) of more and more different types (laptops, gaming consoles, set-top-boxes, smart-home, eHealth, etc.) of network devices at home, the task of developing a solution that runs on every platform appears non-viable. Our home gateway based solution only needs to support a single platform. It can easily replace the existing home gateway, given that home gateways are small and cheap and mainly passive devices without user's data stored on them. On the other hand this also means it has limited resources which makes it challenging to run computationally expensive performance optimization algorithms on them. In this paper we:

1) Present our overall approach on how to optimize the utilization of network resources in home networks (in Section II) consisting of several components including monitoring, application detection, metric computation, resource optimization and traffic shaping;
2) Discuss which metrics are required and review existing measurement tools that provide these metrics (in Section III);

978-1-4673-2480-9/13/$31.00 © 2013 IEEE

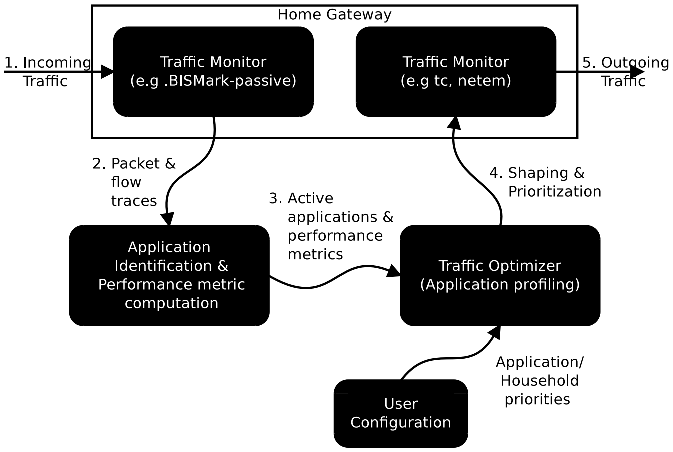
Home Gateway
1. Incoming Traffic
Traffic Monitor (e.g .BISMark-passive)
Traffic Monitor (e.g tc, netem)
5. Outgoing Traffic
2. Packet & flow traces
3. Active applications & performance metrics
4. Shaping & Prioritization
Application Identification & Performance metric computation
Traffic Optimizer (Application profiling)
Application/ Household priorities
User Configuration

Fig. 1. Performance Optimization System.

3) Study the CPU and I/O capacity of typical home gateways beyond pure forwarding of traffic (Section IV) and assess the resource consumption of the traffic monitoring component of our solution (Section V);
4) Examine the overhead involved in exporting the monitored data to an external metric computation and optimization element (in Section VI).

Moreover we discuss how to best perform application identification (in Section VII, present our current state of implementation, (in Section VIII and conclude (in Section IX).

II. The Overall Approach

We envision a system that will control traffic in the home network according to application requirements and household priorities. The network resource requirements vary from one application to another. Some will need low delay while others require high throughput. Household priorities also change depending on activity and over time. A user can give more importance to his mails while working, his file download while installing software or his video streaming when relaxing. Both application requirements and household priorities help to decide how to allocate resources. For this optimization system to exercise the required level of control, it must be installed mainly in the home gateway. Figure 1 shows the complete approach of our performance optimization system at home.

When traffic traverses the home gateway (step 1), the *traffic monitoring* module records flow and packet information (step 2) for the purpose of determining current active applications along with their performance metrics (*application identification and performance metric computation*). This information is send to the *traffic optimizer* (step 3). The optimizer processes this information and assigns the corresponding application traffic profiles. Further it identifies the resource requirements for a given profile and according to the *user configuration* (application/household priorities) sends control parameters (download/upload speed, queue length) to the *traffic controller* (step 4) which will tune the traffic to prevent performance degradation.

In the following we explain each module of this system in more detail:

- **Traffic Monitor**, this module is composed of a modified version of the BISMark-passive [25] function to perform passive traffic measurement. BISMark-passive was originally developed by researcher from Georgia Tech for the purpose of passively monitoring network traffic and periodically sending small anonymized updates to a central server for analysis to help understand home network usage. The recorded packet and flow information is send to the …
- **Application identification & performance metric computation**, a process to detect applications and their performance metrics. For any incoming traffic, this process will identify active applications and their corresponding metrics (bandwidth, packet loss, latency, etc.) and send them to the …
- **Traffic Optimizer**, this module takes as input the active applications with their performance metrics. In addition, it needs existing knowledge (e. g., learned before) about application performance profiles (ranges of acceptable performance for different metrics). This module will combine these information and the user priority to give as output the optimal shaping parameters for the …
- **Traffic Controller**, this module is designed to apply traffic shaping and traffic prioritization in order to forward the traffic in the best form that avoids performance degradation. Some tools as tc [1] and/or netem [2] could be used for traffic shaping.
- **User configuration**, a learning module that records the priority that the user assigns to each application (class of applications). Its output will be included in the *Optimizer* decision process.

While it is clear that the traffic monitor and the traffic controller need to run inside the home gateway, there is some design space on where to run the application identification and performance metrics calculation, the traffic optimizer, and the user configuration.

The proposed modules and their functions need a considerable processing power to be performed. In the remainder of this paper we want to address the question how much processing can be done in the home gateway and discuss the overhead of exporting information from the gateway. In this paper, we focus primarily on the Traffic Monitor and Application identification and metric computation modules. In the following section we will discuss which performance metrics we need to monitor.

III. Collection of Application Performance Metrics

Understanding network application performance is a prerequisite to allocating network resource in a way that users are satisfied. In networking application performance is represented with performance metrics such as *throughput*, *delay* and *jitter*. Other metrics such as the number of *retransmissions* in a connection as well as the number of *concurrent connections* will help us better diagnose the current situation. In addition we need to be able to identify the (type of) *application* which causes a certain piece of traffic. Thus we aim to collect all these metrics on a gateway.

Inferring and monitoring network performance metrics has been well studied. Different tools have been developed to help users and researchers measure simple network metrics. In the following we explore the suitability of existing tools for our purposes.

a) Dedicated metric measurements:: For instance, Pathload measures bandwidth [12], T-rat evaluates the rates at which flows transmit data [27], or King estimates delay by measuring the delay between the closest DNS servers [24]. These tools are actively monitoring a dedicated metric by issuing probes. Despite being fairly accurate the required measurement overhead is a concern to our approach which is supposed to operate permanently.

b) Network performance diagnose:: There are also passive tools that extract network performance metrics. For example, *tcptrace* uses traces collected on end-hosts to compute a set of metrics for each observed connection such as amount of bytes sent and received, number of retransmissions, throughput and others [18]. Similarly, HostView is a monitoring tool [14] that records packet header traces and information about applications and user environment. HostView relies on `gt` [8], for application identification.

As we can see, many tools already exists to measure various metrics. But, these tools have been developed to perform measurements on end-hosts. Moreover, they are not optimized for low resource consumption but rather for high accuracy happily investing more resources. For our performance tracking technique however, we need gateway-based monitoring tools. We can not directly apply the existing solutions for end-hosts in gateways. First, because of the limited resources in home gateways. Second, because on the gateway we do not have access to the same information as we have on end-hosts (e. g., process executable or network stack details).

c) In-network traffic monitoring: An approach closer to our needs is collecting network flow measurements. Protocols like NetFlow collect IP traffic information, for instance source and destination IP's, ports, Timestamps for the flow start and finish time, number of bytes and packets observed in the flow and so on. NetFlow is powerful for collecting IP traffic statistics on all interfaces where it is enabled. Some of our required metrics can be computed from such information. However, it does not collect all the information our tracking needs (e.g. domain names) and is hard to extend.

This is why we base our work on the BISMark firmware [25]. While monitoring a similar set of information compared to NetFlow it is tailored for the use on home gateways (low resource consumption), and easily extensible. The firmware includes the BISMark-passive function which passively collects traces including flow and packet records (timestamps, size, ports, IP addresses, transport protocol and IP to domain name mappings from DNS traffic). These (anonymized) traces are periodically send to a central server for analysis.

Looking at the list of metrics we are interested in at the beginning of this section, we can already calculate many but not all of our desired metrics. In fact, the collection process does not include sequence numbers, acknowledgment numbers and TCP flags which are required to compute round-trip-time, jitter, and retransmissions. For that purpose, we extend the BISMark-passive software to collect the missing metrics by directly changing the implementation to fit our needs.

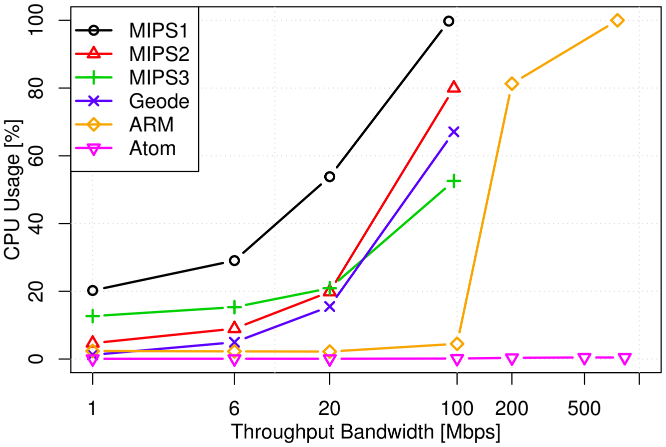

Fig. 2. CPU consumption for six home gateways (MIPS1: Linksys WRT54GL, MIPS2: Dlink DIR-615, MIPS3: Netgear WND3700, AMD Geode LX, OpenRD Kirkwood ARM, Intel Atom 330)

BISMark achieves low resource utilization through three major mechanisms. First, no computation of metrics is performed on the gateway. Second, it only sends incremental updates to the server, e. g., , the flow information (IPs, ports, transport protocol) is send only once and referred to with a connection identifier afterwards or most timestamps are only relative timestamps and thus require less bits to encode. Third, Bismark limits it memory usage for trace data. It records traces in chunks of 30 seconds and each chunk will not store more than a preset number of packets and flows (default: 2^{16}). Then, each 10 minutes all chunks are compressed, sent to a server and deleted on the gateway.

IV. Home Gateway Constrains

Home gateways typically come as small and cheap boxes build out of embedded hardware with low resources. Their limitations include low processing power, small memory size and most times no storage. Different types and brands of gateways are available.

Given the low resources in home gateways, our first question is: Can we do more than just forward packets through the gateway? Figure 2 shows the CPU consumption for 6 different gateways in the market, while forwarding packets at different bandwidth (x-axis). We refer to our technical report [22] for the detailed test setup, measurement procedure, and further results. Each gateway has a maximum forwarding bandwidth, beyond which no measurements are reported in this plot.

We observe that even at 100 Mbps most gateways have CPU resources left to capture and process packets. Only the aged WRT54GL (MIPS1) already reaches its CPU limit. While the Atom and ARM boxes easily achieve several hundreds of Mbps, with today's Internet access link speeds, 50 Mbps should suffice for almost all users. The Netgear WNDR3700 (MIPS3) is a good compromise between cost (80 USD) and performance (60 % remaining CPU @ 50 Mbps). We conclude that more than half of the CPU cycles are left for, e. g., packet capture, network metrics computation, or application detection. Thus we select it for further study.

Although CPU capacity is the prime concern, home gateways also have only a small amount of flash memory and

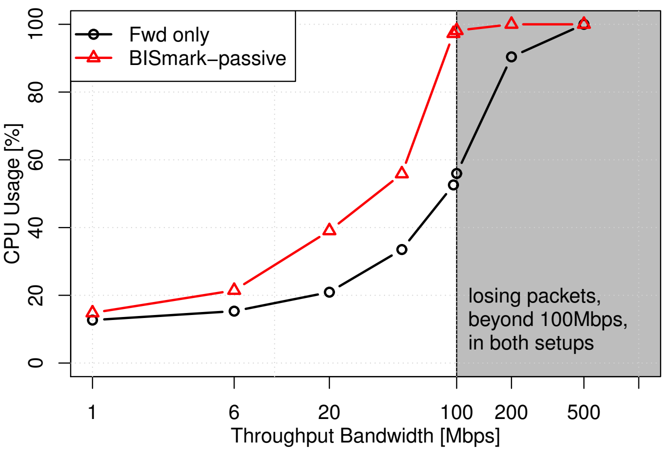

Fig. 3. CPU consumption for forwarding only and running Bismark-passive. Note that the gateway starts losing packet in both setups beyond 100 Mbps.

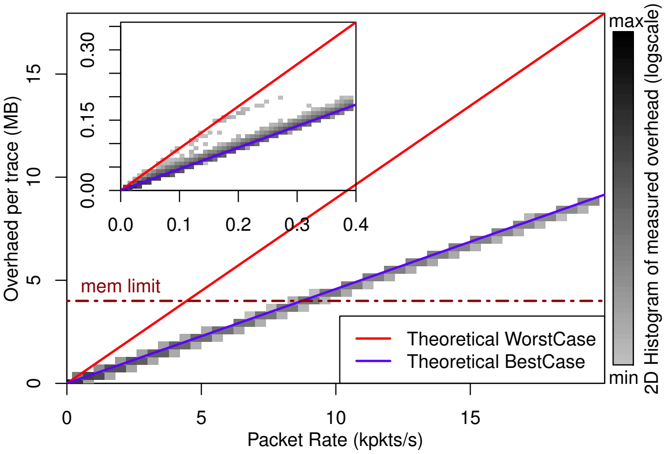

Fig. 4. Distribution of the real collected traces overhead as a function of packet rate

typically no disk storage space. Thus memory management is an important concern for our approach. The WNDR3700v2 ships with 64 MB of RAM and 16 MB of flash memory. After this general performance evaluation we now continue evaluating BISMark-passive's performance on the selected home gateway.

V. BISMark Passive Evaluation

As explained in Section III, we rely on BISMark-passive software to build an extended technique for application performance tracking. Bismark-passive captures all traffic passing through the gateway using the de facto standard libpcap library. It then analyzes the recorded packets and extracts the information described in Section III. Periodically differential updates of the extracted information [4] are gzipped and exported to a server. In Figure 3 we evaluate the BISMark-passive software by comparing its resource consumption against the case of just forwarding packets from last section, when running in the MIPS3 gateway.

As expected the additional task of capturing packets and analyzing them causes noticeably higher CPU consumption at bandwidths beyond 5 Mbps. At typical home network speed (20Mbps), BISMark-passive requires twice as much resources (40% in total) than just forwarding the packets. However the additional CPU requirements of BISMark-passive remain almost constant when looking at 50 Mbps still leaving 40 % of the CPU for other tasks. Yet, at maximum forwarding rate (96 Mbps) BISMark-passive uses the entire CPU capacity.

This evaluation tells that BISMark-passive already consumes a decent amount of CPU. Thus, if in addition we compute all our required metrics, run the performance optimization and the resulting traffic shaping in the gateway, it will generate a high overhead in terms of CPU/memory usage and is likely to run out of resources at high rates.

To avoid a possible lack of resources, our idea is to split the application performance tracking between the home gateway, which is mainly monitoring and extracting packet and flow information, and an additional computation element, which will analyze the compute the performance metrics and determine an optimized resource utilization. The result of this optimization is a set of policies (prioritization rules, QoS parameters) that the home gateway will need to apply to the traffic passing through it. The computation element, could be a laptop or media server at home or "cloudified" resources of a dedicated performance management service.

However with this approach, we need to export information from the gateway in order to be processed outside (computation element). This incurs additional bandwidth requirements, which we will study in the next section. But with this approach we can trade-off home gateway CPU utilization for increased upload bandwidth.

VI. Overhead of Performance Tracking

With the insights from the last section on scarce CPU resources and our approach to ship of the collected information from BISMark to a computation element, we need to ensure the offloading overhead is manageable. Recall that we extended BISmark-passive to collect more traffic information in order to enable our application performance tracking. This causes additional overhead which has to be taken into account.

The difficulty is now that our deployment of the extend BISMark software has not left the lab experimentation stage, and we have no data from a real world deployment. Yet, from analyzing the code of BISMark we know that the overhead which is equivalent to the size of the collected traces only depends on the number of packets and the number of flows that it contains. We also know how much data is produced per packet and per flow seen by the home gateway. Thus, we can combine these constants with real world data collected with the unmodified BISMark firmware, to estimate the required overhead of our extended version. This data was made available to us by the researcher from the BISMark project, which collected these real world traces from 24 different home users using BISMark gateways.

Figure 4 shows the distribution of the estimated overhead (MB) from real world traces as a function of packet rate (K packets/s) in a two-dimensional histogram, where darker regions indicate high numbers of traces matching the region. In addition we show theoretical boundaries (worst and best case). At a given packet rate these boundaries represent the two extreme cases: *(i)* Either every packet in the trace belongs to a different (and new) traffic flow. This represents the worst case as we need both a packet and a flow entry for each packet. Or *(ii)* where every packet in the trace belongs to the very same flow. In which case we still need to store a packet entry for each packet but only one flow entry. Finally we assume that we use no more than 4MB of the gateway memory (RAM only) which is shown as a discontinuous helper line.

As a first takeaway from Figure 4, we find that the real world data matches closely with the theoretical best case. This means that we observe many more packets than different flows, which matches our expectation. If we look at the zoomed inset (small picture, <400 packet/s, 97 % of all the traces), we see a higher variation in the overhead, with some traces reaching the worst case boundary. Yet, at low packet rates also the absolute overhead is small.

To better understand the packet rate, we transform (K packets/s) to (Mbps) assuming an average packet size of 660 bytes from [5]. Thus, based on average packet sizes 5K, 10K, and 15K packtes/s would translate into 26, 53, and 79 Mbps. The figure shows that the memory limitation is respected up to 8K packets/s, as the best case is prevailing at these packet rates. Although this would translate to 42 Mbps, using average packet sizes, we already know that in the best case we see many more packet than flows. Thus we have less small TCP maintenance packets (i. e., SYN, ACK, FIN) and likely many full sized packets. Using full sized (1500 bytes) packets 8K packets/s translate into 90 Mbps.

Looking at our overall approach the findings from this overhead estimation show that it is no problem to collect and export the collected data. Although a worst case of 4 MB memory limit every 30 sec would translate into an uplink bandwidth requiredment of 1 Mbps, this case is highly unlikely. On the one hand the traces are compressed once they are collected which reduces their size to 25 %–50 %. On the other hand from the distribution of packet rates we observe in the data, high packet rate occur only in very few cases (<0.1 %).

VII. Application identification

Many projects already map network flows to application names. What makes application identification in our case more challenging is that we need to develop techniques that offer fast application identification and consume small amount of resources if we want to run it in the gateway. On the other hand for our purpose it might suffice to only detect a certain class of traffic as defined by its network performance profile, i. e., the classes requirements on the network.

In general like for the network performance measurement tools there are two major classes of tools for labeling network traffic with their corresponding application. There are end-host based tools such as ETW [10] (Windows only), *lsof* or *GT* [8] (for UNIX based systems) which associate flows with the appropriate socket entry from the kernel socket tables. Despite their high accuracy, these tools are not possible to use in our gateway-based approach. Thus we need to resort to non end-host based application identification techniques such as port based classification or deep packet inspection (DPI). As an additional requirement for our application performance tracking, we need to identify active applications in real-time.

The BISMark project already tested a DPI solution, namely the TIE tool [7] which proved to consume too much resources for permanent operation. Given that other DPI solutions such as *snort* [23] or *bro* [19], [20] have a extended functionality over TIE, those will likely also consume too much resources. Thus, a gateway-based application identification requires a simpler and less heavy solution.

Considering our data collection process which does not include full packet headers, real-time application classification [17] can be a suitable solution. Although, this technique has a training phase that might be resource consuming, we believe it is a good start because it only needs the timings and sizes of the first packets of a TCP connection [3]. This solution could also be implemented on a computational element outside the home gateway, and the information collected with our extended BISMark software suffices as input. Our idea is to combine this application classification tool with port-based classification and the analysis of domain names collected from BISMark-passive. For HTTP traffic, destination names are especially useful to identify related services. Then to distinguish between different HTTP services, we can also use content-type. If any traffic carries content-type, we can identify whether it is an image, a video, or simple text.

VIII. Implementation

Given the implementation challenges that home gateways bring, the evaluation in this paper is crucial for any implementation decision we choose. In order to track application performance, we modify BISMark-passive implementation. For our collection process we add sequence numbers, acknowledgment numbers and TCP flags in the collected traces. They will be valuable to compute round-trip-time, jitter, and retransmissions.

We are working on a first small deployment of BISMark boxes to collect traffic using our modified BISMark-passive function. The dataset will allow us to deepen our understanding of home traffic and help avoid performance degradation issues. We aim at performing the collection period for a long time in different homes. We already started this step by changing the BISMark-passive code, setting up the gateways and preparing the agreement for the users who will be hosting the gateways at their homes.

We also have already investigated implementation solutions of our final optimization solution. We will start by testing available tools such as WonderShaper [26]. We choose this tool because it already provides low latency for interactive traffic, allows web surfing at reasonable speeds while uploading/downloading, and ensures uploads/downloads do not hurt one another.

IX. Conclusion

Home networks and application performance are two challenging areas for study. In our work, we aim at avoiding performance degradation of all active applications in a home network by tracking their performance from home gateways. We discuss the gateway resource limitations along with the trade-off between different implementation strategies (end-host vs home gateway). Besides, we introduce our modified version of BISMark-passive that collects valuable information for application identification. We show that even if it generates an additional overhead, the results are promising for a use of an application identification technique along with this traffic collection process. We explain possible application identification techniques and discuss our solution implementation. Our overall evaluation shows that it is possible to perform an application performance degradation technique from the home gateway following our guidelines.

Future work. We are currently working on the BISMark-passive software implementation changes. When the extended version will be ready, we intend to use it on our Netgear boxes. We hope to recruit many users to deploy our home gateways and start monitoring traffic.

Later, we plan the build the complete system that controls and avoids performance degradation in home networks. Here, we need to perform another resource consumption study for the traffic control box on the home gateway.

Our final goal is to improve user perception. We believe that our system combined with end-hosts measurements is an efficient solution to achieve our goal. In fact, work on the correlation between user perception and application performance from [13] is complementary to our system and can be used to improve user experience.

An additional idea for the future is to integrate our home performance optimization approach with a prediction technique for user (dis-)satisfaction [15], using the same data that we already collect.

Acknowledgments. We thank Sam Burnett and Srikanth Sundaresan for their help with the BISMark part.This work was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) no. 258378 (FIGARO) and carried out at LINCS (www.lincs.fr).

References

[1] http://lartc.org/manpages/tc.txt.

[2] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[3] Bernaille, L., Teixeira, R., and Salamatian, K. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference* (New York, NY, USA, 2006), CoNEXT '06, ACM, pp. 6:1–6:12.

[4] BISMark-passive. https://github.com/projectbismark/bismark-passive.

[5] Borgnat, P., Dewaele, G., Fukuda, K., Abry, P., and Cho, K. Seven years and one day: Sketching the evolution of internet traffic. In *INFOCOM* (2009), pp. 711–719.

[6] Chetty, M., Banks, R., Harper, R., Regan, T., Sellen, A., Gkantsidis, C., Karagiannis, T., and Key, P. Who's hogging the bandwidth: the consequences of revealing the invisible in the home. In *Proceedings of the 28th international conference on Human factors in computing systems* (New York, NY, USA, 2010), CHI '10, ACM, pp. 659–668.

[7] Dainotti, A., Donato, W., and Pescapé, A. Tie: A community-oriented traffic classification platform. In *Proceedings of the First International Workshop on Traffic Monitoring and Analysis* (Berlin, Heidelberg, 2009), TMA '09, Springer-Verlag, pp. 64–74.

[8] Gringoli, F., Salgarelli, L., Dusi, M., Cascarano, N., Risso, F., and claffy, k. c. Gt: picking up the truth from the ground for internet traffic. *SIGCOMM Comput. Commun. Rev. 39*, 5 (Oct. 2009), 12–18.

[9] Grinter, R. E., Edwards, W. K., Chetty, M., Poole, E. S., Sung, J.-Y., Yang, J., Crabtree, A., Tolmie, P., Rodden, T., Greenhalgh, C., and Benford, S. The ins and outs of home networking: The case for useful and usable domestic networking. *ACM Trans. Comput.-Hum. Interact. 16*, 2 (June 2009), 8:1–8:28.

[10] Insung, Park and Ricky, Buch. Improve Debugging And Performance Tuning With ETW. http://msdn.microsoft.com/en-us/magazine/cc163437.aspx.

[11] Internet World Stats. http://www.internetworldstats.com/dsl.htm.

[12] Jain, M., and Dovrolis, C. Pathload: A measurement tool for end-to-end available bandwidth. In *In Proceedings of Passive and Active Measurements (PAM) Workshop* (2002), pp. 14–25.

[13] Joumblatt, D., Goga, O., Teixeira, R., Chandrashekar, J., and Taft, N. Characterizing end-host application performance across multiple networking environments. In *INFOCOM, 2012 Proceedings IEEE* (March), pp. 2536–2540.

[14] Joumblatt, D., Teixeira, R., Chandrashekar, J., and Taft, N. Hostview: annotating end-host performance measurements with user feedback. *SIGMETRICS Perform. Eval. Rev. 38* (January 2011), 43–48.

[15] Joumblatt, D., Teixeira, R., Chandrashekar, J., Taft, N., and Branislav, K. Predicting user dissatisfaction with internet application performance at end-hosts. In *INFOCOM, 2013 Proceedings IEEE* (march 2013). (accepted for publication).

[16] Karagiannis, T., and Peter, K. Homemaestro: Distributed monitoring and diagnosis of performance anomalies in home networks. Tech. rep., Microsoft Research, 2008.

[17] L.Bernaille. *Real-time application classificationin the Internet*. Ph. d. thesis, Université Pierre et Marie Curie, 2007.

[18] Ostermann, S. Tcptrace: A tcp connection analysis tool. *URL: http://www. tcptrace. org* (2000).

[19] Paxson, V. Bro: a system for detecting network intruders in real-time. *Computer Networks 31*, 23-24 (1999), 2435–2463.

[20] Paxson, V. Bro intrusion detection system, 2013. http://www.bro-ids.org.

[21] Poole, E. S., Chetty, M., Grinter, R. E., and Edwards, W. K. More than meets the eye: Transforming the user experience of home network management, 2008.

[22] Reggani, A., and Schneider, F. Packet capture on home gateways: Is it feasible? Tech. Rep. hal-00763742, UPMC Sorbonnes Universites and CNRS, Paris, France, Mars 2011.

[23] Snort, 2013. http://www.snort.org.

[24] Stefan, K. G., Saroiu, S., and Gribble, S. D. King: Estimating latency between arbitrary internet end hosts. In *SIGCOMM Internet Measurement Workshop 2002* (2002).

[25] Sundaresan, S., de Donato, W., Feamster, N., Teixeira, R., Crawford, S., and Pescapè, A. Broadband internet performance: a view from the gateway. In *Proceedings of the ACM SIGCOMM 2011 conference* (New York, NY, USA, 2011), SIGCOMM '11, ACM, pp. 134–145.

[26] Wonder Shaper. http://lartc.org/wondershaper/.

[27] Zhang, Y., Breslau, L., Paxson, V., and Shenker, S. On the characteristics and origins of internet flow rates. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications* (New York, NY, USA, 2002), SIGCOMM '02, ACM, pp. 309–322.