OMware: An Open Measurement Ware for Stable Residential Broadband Measurement[*]

Lei Xue, Ricky K. P. Mok, and Rocky K. C. Chang
Department of Computing
The Hong Kong Polytechnic University
Hong Kong, China
{cslxue|cskpmok|csrchang}@comp.polyu.edu.hk

ABSTRACT

A number of home-installed middleboxes, e.g., BISMark and SamKnows, and web-based tools, e.g., Netalyzr and Ookla's speedtest service, have been developed recently to enable residential broadband users to gauge their network service quality. One challenge to designing these systems is to provide stable network measurement. That is, the measurement results will not be fluctuated by sporadic overheads incurred inside the middlebox or web browser. In this poster, we propose a network measurement ware, OMware, to increase the stability of residential broadband measurement. The key feature is to implement the send and receive functions for measurement packets in the kernel. Our preliminary evaluation for an OpenWrt implementation shows that OMware provides very stable throughput and delay measurement, compared with typical socket-based measurement at the user level.

Categories and Subject Descriptors

C.4 [**Performance of system**]: Measurement techniques

General Terms

Measurement, Performance

Keywords

Network measurement, high performance, OpenWrt kernel module

1. INTRODUCTION

Residential broadband measurement is one of a few network measurement initiatives targeting at end users. It is carried out usually based on two approaches: host-based tool (e.g., Netalyzr [3] and Ookla's speedtest [4]) or a middlebox between a broadband router and a host (e.g., BISMark [1] and SamKnows [5]). Each approach has its own merits, and the advantages of the middlebox approach is already given in [1]. Regardless of which approach to use, a main challenge is to mitigate as much as possible self-biased measurement results. Web browsers may introduce very high overhead to the latency measurement in the host-based approach, and middleboxes must be carefully designed to allocate resources to the measurement and normal traffic.

This poster focuses on the middlebox approach. These middleboxes are usually run on OpenWrt, a popular Linux platform for home routers. Existing measurement tools are often implemented with the POSIX socket API, which can still be re-used on OpenWrt via cross compilation. However, comparing to commodity PC, an OpenWrt router has very limited resources. The tools running in the user space may not be able to compete with other kernel processes, such as `iptables`. Therefore, it is very challenging to mitigate self-biased results. In this poster, we propose a network measurement ware, OMware, to enhance the stability of residential broadband measurement by implementing the probe send and response receive functions in the kernel.

OMware offers a set of APIs for transmitting ICMP, UDP, and TCP packets with specified delay and receiving packets with matched addresses and ports in the Linux kernel. An active measurement tool can therefore easily access to these kernel services through OMware. The main challenge is the limited resources available in the router. Therefore, instead of using `libpcap`, we implement our own packet capture function through `netfilter`. The outgoing measurement packets are queued and sent out according to the delay specified by the measurement application. Our preliminary evaluation shows that OMware-based measurement provides more stable throughput measurement for a train of packets and accurate dispatching of the measurement packets.

2. OMWARE

Figure 1 shows the overall architecture of OMware. The OMware module is a loadable kernel module, which runs in the background and handles the required packet operations passed via the application interface. Through a set of APIs, OMware provides the services of dispatching measurement packets received from a measurement program according to the specified delay, and delivering to the measurement program measurement packets received from the network. These generic functions are often used in active measure-

[*]This work is partially supported by an ITSP Tier-2 project grant (ref. no. GHP/027/11) from the Innovation Technology Fund in Hong Kong.

ment tools. The probe packet's content and parameters are passed to the OMware module via a `netlink` socket.

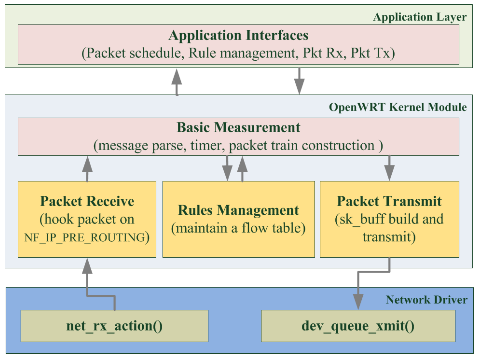


Figure 1: The architecture of OMware.

The OMware module comprises three components: `Packet Tx`, `Packet Rx`, and `Rule Management`. `Packet Tx` is responsible for queueing the outgoing measurement packets. The probe packets received from the application are prepared in the kernel module and setup kernel tasks according to their assigned transmission times. When a send event is triggered, the probe packet is copied to the network driver socket buffer directly to reduce the software overhead.

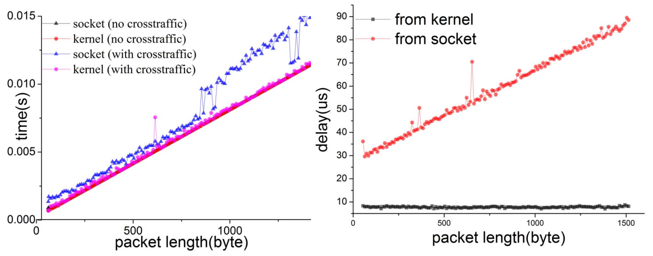
For packet reception, we implement `Packet Rx` by hooking to `NF_IP_PRE_ROUTING` in the `netfilter` library and copy the packet to the attached measurement flow. The packet timestamp is given by the network driver, which is the same source used in `libpcap`. As all incoming and outgoing packets pass the hook, OMware's `Rule Management` maintains a flow table which is used for identifying measurement packets. Hence, the impact on non-measurement traffic can be minimized.

3. EVALUATION

In this section we evaluate the performance of OMware-based measurement by comparing with socket-based measurement in the user space. We develop two corresponding measurement tools for sending out a train of packets using OMware and raw socket, respectively. We have implemented OMware in a Netgear WNDR3800 router. Our setup is consisted of a PC (source) connected to a LAN port of the router and a laptop (destination) connected to the WAN port of the router via a 100-Mbps link. We perform two sets of experiments: with and without cross traffic. We generate the cross traffic using the D-ITG tool [2], and the traffic is sent from the source host to the destination host through the router. The cross-traffic rate is 9000 packets per second and the packet size is uniformly distributed in the range of [100, 1500] bytes.

We first evaluate the throughput measurement conducted by both methods. The measurement programs generate a packet train of 100 packets with zero time gap between two consecutive packets with packet sizes ranging from 54 bytes to 1454 bytes with a 10-byte increment. We compute the duration required for sending the entire packet train. Figure 2a shows that both methods obtain the same throughput measurement in the absence of cross traffic for all packet sizes. However, in the presence of the cross traffic, the OMware-based measurement obtains a considerably higher throughout, especially when the packet size is greater than 800 bytes.

Besides, Figure 2b plots the inter-packet delay which is specified to be 0 s. The OMware-based measurement incurs a delay of around 5 μs which is very stable across the packet sizes. However, the delay for the socket-based measurement is higher and increases with the size of the packet.



(a) Throughput measurement. (b) Inter-packet delay measurement.

Figure 2: A comparison between socket-based and OMware-based measurement.

We also evaluate the accuracy of dispatching the measurement packets according to the specified delay for the OMware-based measurement. We experiment with inter-packet delay between 0.3 s to 0.9 s in an increment of 0.2 s and the packet's length being 54 bytes. We repeat each experiment for ten times and compute the mean, standard deviation, and a maximum deviation error which is the maximum of (|actual sent time − scheduled send time|/scheduled sent time)×100%. As shown in Table 1, the OMware-based measurement can dispatch the packets with a fairly high accuracy.

Table 1: An evaluation of the packet dispatching accuracy for the OMware-based measurement.

Schedule period	0.3 s	0.5 s	0.7 s	0.9 s
Mean period	0.299994	0.499993	0.799992	0.899992
Jitter	0.000011	0.000011	0.000010	0.000011
Max deviation	0.0117%	0.0074%	0.0047%	0.0041%

4. CONCLUSIONS

Our preliminary evaluation finds that the OMware-based measurement is very promising. However, more evaluation efforts are needed to draw sound conclusions. In our future work, we will evaluate BISMark's performance and compare it with OMware.

5. REFERENCES

[1] BISmark. http://www.projectbismark.net.
[2] A. Dainotti, A. Botta, and A. Pescapè. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531–3547, 2012.
[3] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *Proc. ACM/USENIX IMC*, 2010.
[4] Ookla. Speedtest.net. `http://www.speedtest.net/`.
[5] SamKnows. http://www.samknows.com.