An automated testbed for profiling the packet send-time accuracy of embedded devices

Ricky K. P. Mok, Weichao Li, Rocky K. C. Chang,
Kwok-Wun Yung, Ching-Ho Chan, and Yat-Sing Poon
Email: {cskpmok|csweicli|csrchang}@comp.polyu.edu.hk,
{cskwyung|cschchan}@comp.polyu.edu.hk,13040015d@connect.polyu.hk

ABSTRACT

Embedded devices, such as home routers and single-board computers, are becoming more powerful and affordable. Many of the existing Linux-based network measurement tools can also be run on these devices through cross-compilation. These features provide more incentive for network administrators and researchers to employ them as network measurement probes. Large-scale measurement projects, such as BISMark and RIPE, have already deployed more than hundreds of these devices to measure the Internet from the edge.

Our previous work shows that these embedded devices have much lower packet send-time accuracy compared to commodity PC. The lower accuracy limits their ability to acquire sound measurement results in high-speed networks. In this poster, we present an automated testbed to systematically benchmark the performance of four popular embedded devices (Raspberry Pi I & II, ECS LIVA, and TP-LINK Travel Router) and determine the network types that they can accurately measure. We employ `OMware`, a kernel-space module, to measure their performance in the testbed. The results can be visualized through a web interface, thus enabling users to compare various devices easily.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*Measurement techniques*

Keywords

Embedded devices, network measurement, packet send-time accuracy

1. INTRODUCTION

Embedded devices are ubiquitous nowadays, as they are affordable, low-energy consumption, and reasonably powerful. A well-known example, Raspberry Pi, only costs about $30 US. These devices are usually Linux-based and are able to provide many kinds of network services. Through cross-compilation, many of the existing Linux-based network measurement tools can also run on these devices.

TRIDENTCOM'15, June 24–25, 2015, Vancouver, Canada.
ACM .

Large-scale measurement platforms (e.g., [11, 10]) often require multiple probing machines to measure and monitor a network's performance from different vantage points in order to diagnose network faults [7]. However, the scalability of these platforms is often limited by the cost of the machines. The low-cost embedded devices, therefore, provide an inexpensive option for network administrators and researchers to employ them as lightweight measurement probes. For example, Project BISMark [14], RIPE [1], and CAIDA's Archipelago (Ark) [2], have already deployed more than hundreds of them as vantage points to measure the Internet from the edge.

However, these embedded devices have limited resources. We find that they have much lower packet send-time accuracy compared to commodity PC [12]. The lower accuracy limits their ability to acquire sound measurement results. In particular, some network metrics, such as capacity and available bandwidth, are very sensitive to timing accuracy, as they rely on accurate packet sending pattern. For example, the minimum packet-pair dispersion can limit the maximum measurable capacity. However, it is not straightforward to effectively compare the accuracy of different devices.

In this poster, we present an automated testbed to systematically benchmark the performance of four popular embedded devices, including ECS LIVA, Raspberry Pi I & II, and TP-LINK Travel Router. We implement a testing tool using `OMware` [12], which is a Linux loadable kernel module for improving packet send-time accuracy. The tool can send prescribed packet patterns used by network measurement tools and record the packets' sending times. Based on the results which can be visualized through a web interface, the testbed can report the network types that each device can accurately measure.

2. THE TESTBED

Figure 1 shows the setup and test flow of the testbed. The devices under test and the controller PC are connected through a Gigabit switch. The controller provides a web interface, which displays the available devices and allows users to choose one of them for evaluation in a browser. When the experiment starts, the controller remotely executes the tests on the device. The results are collected and visualized by the controller. The testbed will also summarize the results to help users understand the accuracy of the device.

The functional blocks of the testbed are depicted in Figure 2. We have implemented a web interface based on Apache and PHP to handle the interaction between users and the testbed. The controller consists of three major components:

TRIDENTCOM 2015, June 24-25, Vancouver, Canada

DOI 10.4108/icst.tridentcom.2015.259826

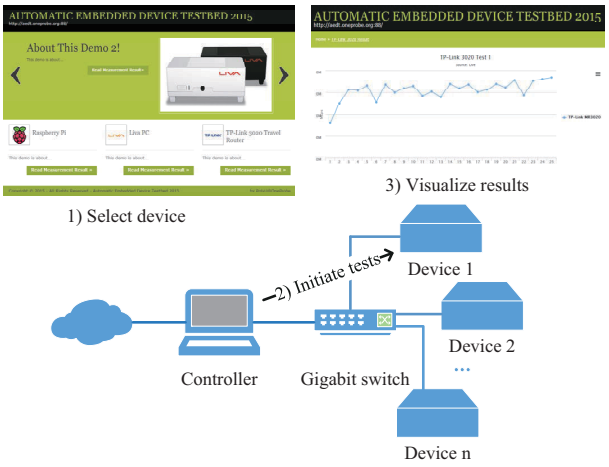


Figure 1: The testbed setup and test flow.

device manager, *experiment manager*, and *data manager*. These components are implemented with Perl. The device manager maintains the information of each connected device (e.g., the model, IP address, and authentication information), while the experiment manager accepts a user's instructions and deploys evaluation tasks to the corresponding device through SSH. When receiving the request from the controller, the device under test then launches the *executer* and performs a set of tests (as described in §3). The experiment results are collected by a *data collector* and sent back to the controller. After being processed by the data manager, the data are visualized in terms of a set of metrics. At the same time, the best network types that the device can accurately measure can also be determined based on the metrics. For the purpose of further analysis, both the raw data and the metrics are stored in a database.

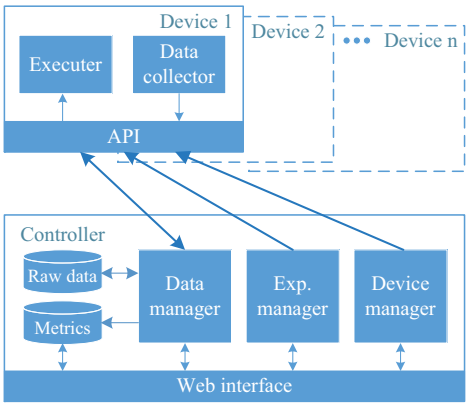


Figure 2: The implementation details of the testbed.

3. EVALUATING THE DEVICES

We evaluate the devices by sending two packet patterns—packet pair and packet train. These two patterns are often used in measuring network capacity [6, 8, 4, 3] and available bandwidth [13, 9]. In our evaluation, we emulate these tools by programming the devices to send these packet patterns. At the same time, the packets are captured and timestamped by the devices. We then compute the following three metrics to benchmark the performance of each device.

1) Minimum inter-packet gap, denoted by g, is obtained by using packet-pair test. We subtract the timestamp of the first packet from that of the second in each packet pair. This metric can be used to infer the maximum network capacity the device can measure. A smaller value implies a higher measurable capacity.

2) Average packet sending rate, denoted by r, is obtained by sending packet trains with pre-defined sending rates. This metrics is to evaluate how close the actual sending rate to the expected. We can compute the actual average packet sending rate by Eqn. (1).

$$r = \frac{s \times 8}{\frac{1}{n-1}\sum_{i=1}^{n-1} t_i - t_{i-1}}, \quad (1)$$

where s is the packet size, n is the total number of packets in the packet train, and t_i refers to the timestamp of i^{th} packet in the packet train.

3) Average inter-departure time variability, denoted by j, is computed by difference in the timestamps of packets in a packet train as shown in Eqn. (2). This metrics is particularly useful in estimating the stability of inter-departure time in a packet train when the device is under loading.

$$j = \frac{\sum_{i=1}^{n-1} |t_i - t_{i-1}|}{n-1}. \quad (2)$$

We have implemented a testing tool to conduct two tests named *packet-pair* and *packet-train* tests. Our tool is built based on `OMware` [12] to measure the best possible performance of the devices. As a packet timestamping device, such as endace DAG, may not be readily available in a measurement tool, we use the packet sending timestamps reported by `OMware`. Our previous results [12] show that these timestamps are the same as using `tcpdump`.

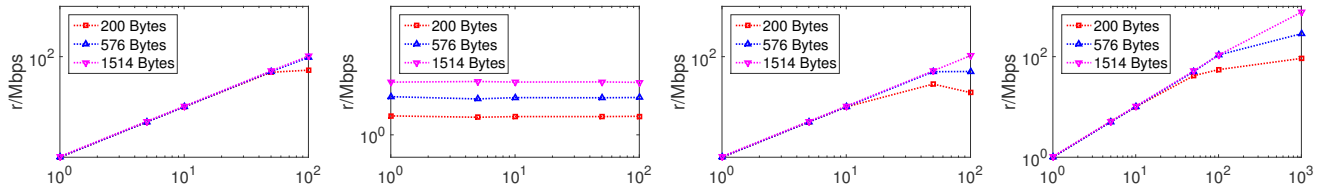
Packet-pair test: This tool sends 25 pairs of back-to-back packet pairs with a specified packet size using the packet-pair sending function in `OMware` to achieve a smallest inter-packet gap. We insert a 1-ms delay between two adjacent pairs to avoid interference from the previous pair.

Packet-train test: This tool sends a packet train consisting of 50 packets with different packet sending rates. We adjust both the packet inter-departure time and packet sizes to achieve different sending rates. In order to obtain the best performance of the device, all the packets are first predispatched into `OMware` to mitigate the overhead from the user space. Table 1 lists the parameters used in our tests. Besides, to induce loading to the device, the device is set up to also generate different numbers of cross-traffic flows using D-ITG [5]. The cross-traffic flows are UDP flows with the packet inter-departure time following the Pareto distribution.

Table 1: Parameters tested in the packet-train test.

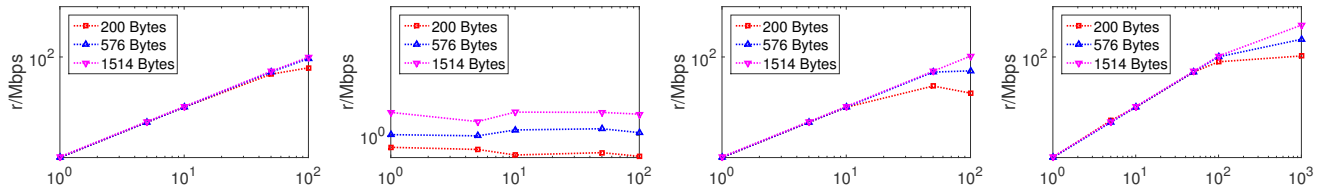
Parameters	Values
IP Packet size (Bytes)	200, 576, 1514
Packet sending rate (Mbps)	1, 5, 10, 50, 100, 1000
No. of cross-traffic flows	0, 5, 15, 25

4. EXPERIMENT RESULTS



(a) TP-LINK. (b) Raspberry Pi I. (c) Raspberry Pi II. (d) ECS LIVA.

Figure 3: The log-log plot of measured packet sending rate r against the expected without cross-traffic.



(a) TP-LINK. (b) Raspberry Pi I. (c) Raspberry Pi II. (d) ECS LIVA.

Figure 4: The log-log plot of measured packet sending rate r against the expected with cross-traffic (25 flows).

To show the performance difference among devices, we evaluate four devices with the price ranged from $20 to $130 US. All of them are very compact in size. ECS LIVA is a x86-based mini-PC, while the other three use ARM-based CPU. The TP-LINK travel router and Raspberry Pi II are employed by RIPE and Ark, respectively, as measurement probes. Their detail configurations are listed in Table 2. Due to the page limit, we can only show part of the results in this poster. Figure 3 plots the average packet sending rate of the four devices without cross traffic. The x-axis is the expected packet sending rate, while the y-axis is the measured one. We do not test 1 Gbps in the TP-LINK router and Raspberry Pi I & II, because only ECS LIVA supports 1-Gbps Ethernet.

Table 2: The configurations of the four devices.

	TP-LINK ML-3020	Raspberry Pi I (B)	Raspberry Pi II (B)	ECS LIVA
CPU Freq.	400 MHz	700 MHz	900 MHz	2 GHz
RAM	32 MB	512 MB	1 GB	2 GB
LAN †	FE	FE	FE	GE
OS∗	W	W	W	U
Price (USD)	∼$20	∼$35	∼$35	∼$130

Note: † FE: 10/100Mbps; GE: 1 Gbps
∗ W: OpenWrt 12.09; U: Ubuntu 14.04.

The P-LINK router and ECS LIVA are quite accurate in all packet sending rates less than 100 Mbps when the packet size is reasonably large (i.e., >=576 Bytes). However, when the packet size is small (e.g., 200 Bytes), the actual sending rate cannot reach the expected rate (i.e., 100 Mbps). This suggests that the devices reach their performance bottlenecks when processing a large number of packets. Surprisingly, Raspberry Pi I performs poorly in all the tests. It sends full-sized packets at around 17 Mbps for all sending rates (see Figure 3(b)). Raspberry Pi II, the latest model of this product, has a similar design (using USB to provide the Ethernet interface), but it can send packets with accurate send time in most of the sending rates less than 50 Mbps. We believe that the poor time accuracy in Pi I could be due to the Ethernet chip (Microchip LAN9512 in Pi I vs. LAN9514 in Pi II). Therefore, we recommend that Raspberry Pi I should not be used for network performance measurement.

To analyze the impact of cross traffic, we plot the sending rates with 25 additional UDP flows in Figure 4. The average packet sending rate of LIVA is dropped from 760 Mbps to 430 Mbps when sending a 1-Gbps packet train with full-sized packets. On the other hand, we cannot observe significant performance degradation for both TP-LINK and Raspberry Pi II. Based on our experiment results, the best network speed that TP-LINK, Raspberry Pi II, and LIVA can measure are less than 50 Mbps, 10 Mbps, and 50 Mbps, respectively, for all packet sizes. The measurable network speed for Raspberry Pi II and LIVA can be increased to 50 Mbps and 100 Mbps, respectively, when large packets are used.

5. CONCLUSIONS

In this poster, we introduced an automatic testbed to evaluate the performance of embedded devices for network performance measurement. With the testbed, we can easily identify that Raspberry Pi is not suitable for performing network performance measurement. More generally, the testbed results can be used to determine which types of networks each device can measure accurately.

In the future work, we will compare the results with an external packet capturing device (e.g., DAG card) to validate the accuracy of the timestamps reported by `tcpdump`. Besides, we will introduce more embedded devices and tests to the testbed.

6. ACKNOWLEDGEMENTS

We thank three anonymous reviewers for their valuable comments. 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.

7. REFERENCES

[1] RIPE Atlas. https://atlas.ripe.net/.

[2] CAIDA. Archipelago (ark) measurement infrastructure. http://www.caida.org/projects/ark/.

[3] L.-J. Chen, T. Sun, B.-C. Wang, M. Sanadidi, and M. Gerla. PBProbe: A capacity estimation tool for high speed networks. *Computer Communications*, 31(17):3883–3893, 2008.

[4] D. Croce, T. En-Najjary, G. Urvoy-Keller, and E. Biersack. Capacity estimation of ADSL links. In *Proc. ACM CoNEXT*, 2008.

[5] 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.

[6] C. Dovrolis, P. Ramanathan, and D. Moore. Packet-dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Trans. Netw.*, 12(6):963–977, 2004.

[7] W. Fok, X. Luo, R. Mok, W. Li, R. Chang, Y. Liu, and E. Chan. MonoScope: Automating network faults diagnosis based on active measurements. In *Proc. IFIP/IEEE IM*, 2013.

[8] R. Kapoor, L. Chen, L. Lao, M. Gerla, and M. Sanadidi. CapProbe: A simple and accurate capacity estimation technique. In *Proc. ACM SIGCOMM*, 2004.

[9] J. C. Kim and Y. Lee. An end-to-end measurement and monitoring technique for the bottleneck link capacity and its available bandwidth. *Computer Networks*, 58:158–179, 2014.

[10] A. Kvalbein, D. Baltrūnas, K. Evensen, J. Xiang, A. Elmokashfi, and S. Ferlin-Oliveira. The Nornet Edge platform for mobile broadband measurements. *Computer Networks*, 2014.

[11] W. Li, W. Fok, E. Chan, X. Luo, and R. Chang. Planetopus: A system for facilitating collaborative network monitoring. In *Proc. IFIP/IEEE IM*, 2011.

[12] R. Mok, W. Li, and R. Chang. Improving the packet send-time accuracy in embedded devices. In *Proc. PAM*, 2015.

[13] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proc. ACM IMC*, 2003.

[14] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband Internet performance: A view from the gateway. In *Proc. ACM SIGCOMM*, 2011.