EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

An FPGA-based high-speed network performance measurement for RFC 2544

Yong Wang[1*], Yong Liu[2], Xiaoling Tao[3] and Qian He[3]

Abstract

Aiming at the problem that existing network performance measurements have low accuracy for (Request for Comments) RFC 2544, this paper proposes a high-speed network performance measurement based on field-programmable gate array (FPGA). The active measurement method is used to generate probe data frames, and a passive measurement method is employed to count network traffic. According to the statistical laws based on throughput variation, interval stretching mechanism is used to dynamically adjust interframe gap. When our approach approaches the maximum throughput, the network performance parameters are achieved. A prototype based on NetFPGA is also implemented for evaluation. Experimental results show that our approach can be applied in high-speed network and the latency can be accurate to the nanosecond. Compared with network performance measurement using software to send probe data frames and a similar work based on FPGA, our approach can be more flexible and the evaluation data are more accurate.

Keywords: RFC 2544; Active measurement; Passive measurement; Traffic generator; Interval stretching; Network performance measurement

1 Introduction

The explosive growth in Internet deployment for a constantly growing variety of applications has created a massive increase in demand for network performance parameters, such as throughput, latency, and packet loss rate [1-6], which are very important for providing differentiated network services. Accurate network performance parameters can help to improve the quality of network services, including active and passive resource management, traffic engineering, as well as providing quality of service (QoS) guarantees for end-user applications. In particular, as modern network management systems shift their focus forward service-level and application-level management, the network monitoring process requires more data to be collected in a higher frequency.

With the development of new network applications and value-added services, network traffic characteristics have become more and more complex. Different applications have different flow characteristics and behavioral characteristics, so it is not enough to analyze only using mathematical simulation and the classical queuing theory mode. But through the result of the network performance parameters and its analysis, we could simulate the Internet environment accurately, which helps us to optimize the network and design the network equipment to a certain extent.

Different network applications have different requirements for QoS. For example, file transfer services require low packet loss rate and high throughput, and real-time multimedia services demand low latency [7]. Through network measurement, the users can detect network congestion, locate network performance bottlenecks, and provide the basis for the network resource optimization.

Faced with an increasingly serious threat to network security, the large-scale network measurements are used to analyze and assess the network performance in abnormal circumstances so early warning can be provided to prevent large-scale network attacks. Therefore, network measurement method also has become an important mean to protect the network security and prevent large-scale network attacks.

The most frequent used methods are ping and traceroute. By calculating the time between sending the

* Correspondence: wang@guet.edu.cn
[1]CSIP Guangxi Center, Guilin University of Electronic Technology, Guilin 541004, China
Full list of author information is available at the end of the article

Internet Control Message Protocol (ICMP) or User Datagram Protocol (UDP) packet and receiving the response packet, the end-loop time can be obtained, but the processing rate restricts the performance of the above solutions, which only enables the latency to be accurate to the milliseconds. Transmission Control Protocol (TCP) state detection [8] performs analysis of TCP flows. It utilizes TCP data packets and its acknowledgement (ACK) packets to measure the throughput and latency. Unfortunately, the results are only closer to the actual network. Using the software [9,10] to generate test flow can be easy to achieve, but its results are difficult to make the users satisfy to a certain extent because it could create an extra overhead when generating the test flows constantly. It usually is not suitable to be deployed in high-speed network. In addition, common network performance analyzers, such as SmartBits and TestCenter, are too expensive to be suitable for general performance benchmarks.

In this paper, an FPGA-based high-speed network performance measurement for RFC 2544 [11] is proposed. The active measurement method is employed to generate a lot of probe data frames set by the user. The passive measurement method is used to precisely count network traffic of each Ethernet interface and other related parameters through *Register I/O* in real time. According to the change of throughput, it dynamically adjusts the interframe gap to reach the limit of network performance. Our approach not only gets the latency that is accurate to the nanosecond but also can be applied in high-speed network.

The rest of this paper is organized as follows. Section 2 discusses related work on network performance measurement. In Section 3, we put forward the network performance measurement benchmarks. The design of our approach is shown in Section 4. Then in Section 5, experimental tests and performance analysis are given. Section 6 concludes the paper.

2 Related work

Currently, the majority of existing network performance measurement methods is software-based and concentrates on the measurement of throughput, latency, packet loss rate, and so on. The network performance measurement techniques are divided into two categories [12]. One is the active measurement method and the other is passive measurement method. The active measurement method usually does not require collaboration among multiple nodes, which is very flexible. It is also easy to operate. A methodology that estimates packet loss rate between arbitrary end hosts without control on either end is developed in [13]. Using UDP by default to transmit Domain Name Service (DNS) queries actively, it takes advantage of the retransmission behavior of deployed DNS servers to conduct measurements, but it needs much more resources than others, which could bring some difficulties to some extent. Sending probe frames or Internet Protocol (IP) packets to conduct the measurements is another solution. The network traffic configured by the users is needed to generate in the client windows. When the data flows are sent or received, it should count the network traffic. Then, the related parameters, such as latency, packet loss rate, and throughput, are calculated. The hardware-based method can process network traffic at much higher data rates. A hardware circuit is tested using an FPGA device which conducts an Ethernet tester [14] compliant with the throughput and latency tests specified by the RFC 2544 for 10/100 Mbps Ethernet networks; the usual limitations added by several hardware and software layers can be overcome by implementing a frame generator directly in an FPGA device, but it is only deployed in 10/100 Mbps Ethernet networks. It is also not flexible enough. A *traffic generator* [15,16] based on FPGA can achieve full gigabit link utilization, which motivates us to do it better.

The passive measurement method does not take up network bandwidth and has less impact on the network. It can also help us to get accurate results. To calculate per-application packet loss, [17] periodically retrieves the expired flows from the two passive monitoring sensors. Allowing pinpointing loss events for specific classes of traffic, the packet loss rate of individual traffic flows can be measured. Since it gets the related information periodically, the packet loss rate is difficult to be achieved in real time. The recognition of lost and retransmitted packets and segments for packet-based methods is one of challenges in passive round-trip time (RTT) measurement. To overcome this problem, [18] proposes an estimation technique for RTT measurement based on flow monitoring. It calculates RTT from five-tuple flow measurement, which identifies the flow, as well as start time and end time for each flow record, but it should run much more time and its results are not accurate.

3 Benchmarks

RFC 2544 provides a lot of parameters applied in different network equipment test. In this paper, the latency, throughput, and packet loss rate in our approach are provided to conduct network performance measurement.

3.1 Latency

The frame forwarding delay can be divided into queuing delay, transmission delay, and the propagation delay three parts. The transmission process of a frame is shown in Figure 1. For a frame k, the queuing delay is $t_{iq}^{(k)}$, the transmission delay is $t_{id}^{(k)}$, and the propagation delay is $t_{ld}^{(k)}$. The node in Figure 1 is a target device such

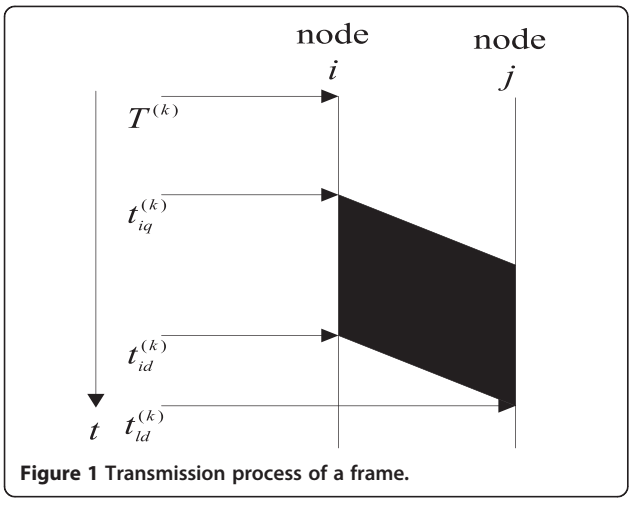


Figure 1 Transmission process of a frame.

as firewall, switch, router, and so on or an equipment conducting measurement.

Supposing that the delay of the frame k at the node i is $T^{(k)}$, we can calculate it as follows:

$$T^{(k)} = t_{iq}^{(k)} + t_{id}^{(k)} + t_{ld}^{(k)} \tag{1}$$

$t_{id}^{(k)}$ and $t_{ld}^{(k)}$ are certain under a specific measurement condition. Due to the network traffic congestion and other factors in the cache of the node i, the length of the queue is dynamic, which makes the $t_{iq}^{(k)}$ changed.

The current network performance measurement method usually uses round-trip delay to evaluate the network performance. However, the one-way delay may be more meaningful for network services and its error is smaller than the round-trip delay [19]. Therefore, we use one-way delay to reveal the latency in this paper. We assume that $T_t^{(k)}$ is the delay of the frame k at the node j and $T_o^{(k)}$ is the overlap between the node i sending the data frame and the node j receiving the data frame; Then, we can get the one-way delay $T_a^{(k)}$, which is stated as follows:

$$T_a^{(k)} = T^{(k)} + T_t^{(k)} - T_o^{(k)} \tag{2}$$

3.2 Throughput

When a sequence of frames $\{P_1,P_2,\ldots,P_n\}$ are transmitted, δ is interframe gap between P_i and P_{i+1} $(1 \le i \le n-1)$. The forwarding delay of the above frames is clearly illustrated in the interframe gap model shown in Figure 2.

S_i is the frame size corresponding to the frame P_i. The total size $S_a^{(n)}$ of the above frames sequence can be set as follows:

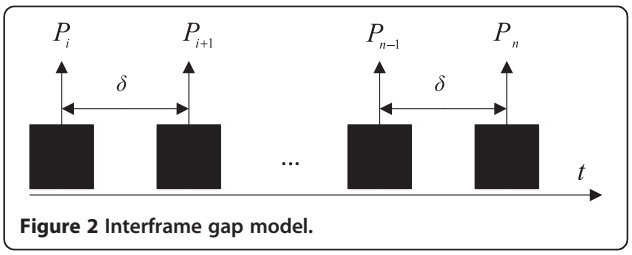


Figure 2 Interframe gap model.

$$S_a^{(n)} = \sum_{i=1}^{n} S_i \tag{3}$$

The time t_i required for processing the frame P_i is divided into two parts. One is the interframe gap δ and the other is the delay of forwarding frame. The t_i and the total forwarding delay $T_a^{(n)}$ can be obtained as follows:

$$t_i = \begin{cases} \dfrac{S_i}{C}, & i = 1 \\ \delta + \dfrac{S_i}{C}, & 1 < i \le n \end{cases} \tag{4}$$

$$T_a^{(n)} = \sum_{i-1}^{n} t_i, \tag{5}$$

where C is the bottleneck bandwidth. The first frame P_1 does not need to wait for the interframe gap. Thus, we can calculate the throughput T_B at the above nodes denoted as follows:

$$T_B = S_a^{(n)} / T_a^{(n)} = \sum_{i=1}^{n} S_i / \sum_{i=1}^{n} t_i \tag{6}$$

We define that the throughput of entering the node represents the downlink throughput, while the throughput of leaving the node indicates the upstream throughput.

Assuming that $V = \{T_{BI},T_{B2},\ldots,T_{Bm}\}$ is a set consisting of valid measurement of throughput on the sequence of frames $\{P_1,P_2,\ldots,P_n\}$for m times, we can get the bottleneck bandwidth C as follows:

$$C = \max_{T_{Bi} \in V} (T_{Bi}) \tag{7}$$

From Equations 4, 5, and 6, it is obvious that if we reduce the δ, then the $T_a^{(n)}$ will also be reduced so that the T_B will increase correspondingly; but conversely, if we increase the δ, it will make the T_B decreasing in turn. Therefore, it is theoretically proved that we can dynamically adjust the interframe gap to gradually approach to the maximum throughput, which enables us to implement the measurement of the bottleneck bandwidth.

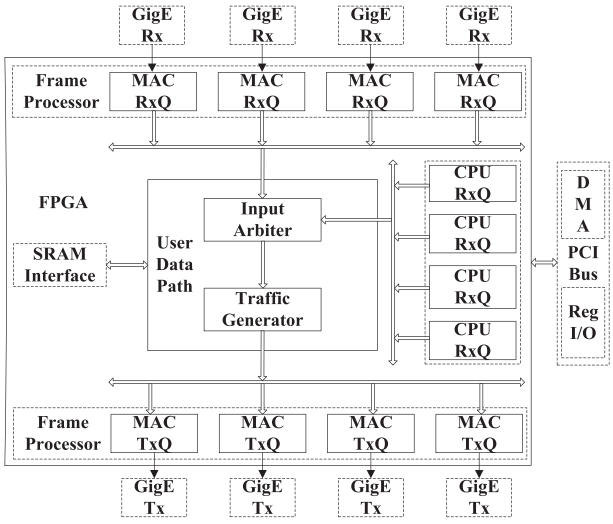


Figure 3 Layout of the network performance measurement.

3.3 Packet loss rate

For the above sequence of frames $\{P_1,P_2,\ldots,P_n\}$ sent, the number of the frames sent and the total bytes of the sequence sent are $S_{aout}^{(n)}$ and $S_{bout}^{(n)}$ at the node *i*, respectively. On other hand, at the node *j*, the number of the frames stored and the total size of the sequence stored are $S_{ain}^{(n)}$ and $S_{bin}^{(a)}$, respectively. Thus, frames' loss rate L_f and bytes' loss rate L_b can be set as follows:

$$L_f = \frac{S_{aout}^{(n)} - S_{ain}^{(n)}}{S_{aout}^{(n)}} \times 100\% \tag{8}$$

$$L_b = \frac{S_{bout}^{(n)} - S_{bin}^{(n)}}{S_{bout}^{(n)}} \times 100\% \tag{9}$$

4 The FPGA-based network performance measurement

4.1 Architecture

The layout of the components along with data frame and component interactions is shown in Figure 3. It mainly includes traffic generation, traffic statistics, and communication interaction.

The active measurement method is used to generate probe data frames. *user data path* is the core of data path processing, which includes an *input arbiter* and a traffic generator. The input arbiter chooses a data frame from a *MAC RxQ* or a *CPU RxQ* and then makes it entry into the traffic generator. By accessing the static random access memory (SRAM), a lot of data frames are generated by the traffic generator according to the parameters configured by the user.

The passive measurement method is employed to count network traffic. The MAC RxQ and TxQ are part of a *frame processor*, which receives and transmits the data frames from the Ethernet interface *GigE Rx* and *Tx*, respectively. The network traffic stored and sent according to its Ethernet interface and the frame forwarding delay are also counted by it.

The host computer often communicates with the FPGA to complete the performance measurement. Through Peripheral Component Interconnect (PCI) bus interface, not only the host computer can send data frames to FPGA through the direct memory access (DMA)-engine, but also the host computer can access the internal registers of the FPGA any time. A CPU RxQ receives the data frames configured by the user from the host computer via PCI bus, which is the source of the probe data frame.

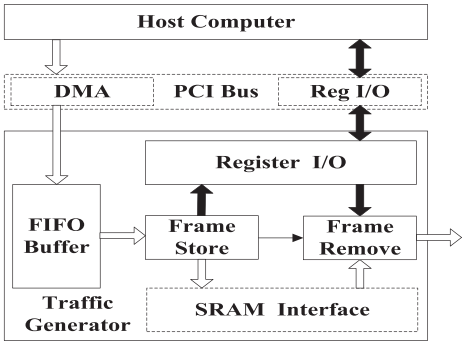


Figure 4 Layout of the traffic generator.

4.2 Traffic generator

The traffic generator uses the active measurement to generate network traffic. The original probe data frame configured by such parameter as the frame length is generated by the libnet [20] in the host computer, which installs the NetFPGA board. Through PCI bus interface, it is transmitted to a CPU RxQ. Next, the input arbiter chooses a data frame from the above CPU RxQ. Then, the data frame enters the traffic generator via a 64-bit wide data bus. The layout of the traffic generator is shown in Figure 4. The data frame traverses the circuit following the path indicated by the bold and white arrows, while the register values crossing the circuit are shown by the bold and black arrows.

The data frame is initially processed by *FIFO buffer* which buffers some bytes if there are any downstream processing delays. Then, the frame flows into *frame store*. Next, the frame is stored in SRAM via *SRAM interface*. The associated frames with the appropriate frame context information are retrieved via Register I/O so that the data frame in the SRAM can be monitored. When the last word of the data frame is written into the SRAM, the frame store will send a control signal to make *frame remove* informed, which tells the frame remove to begin reading the data frames in the SRAM.

The frame remove provides a client interface and passes the data frame in the SRAM to a *MAC TxQ*. It obtains output Ethernet interface x, the number of data frames y, and interframe gap z via Register I/O, which are configured by the user through PCI bus interface at the beginning of the FPGA working. Using the active measurement method, the frame remove reads the data frame in the SRAM and sends it to a MAC TxQ according to x for y times. When reading two adjacent frames in the SRAM, it should wait for z clock cycles. In order to identify the success of the measurement, it allows setting the different 64 bits of special tags in the first and last data frame. The traffic generator process is shown in Figure 5.

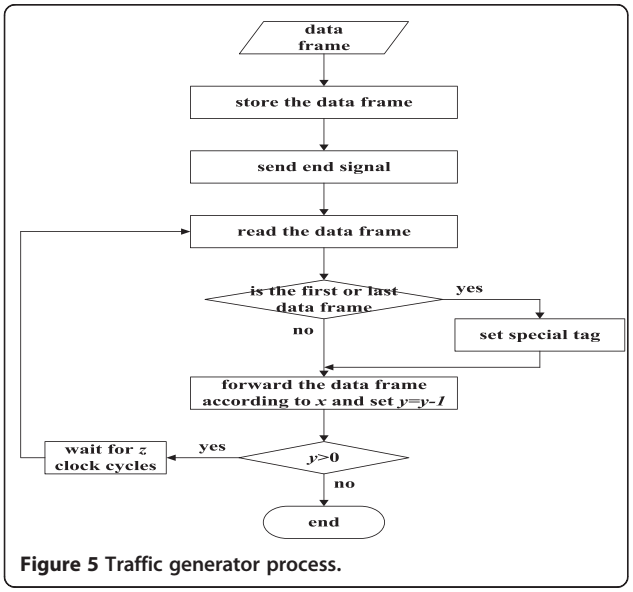


Figure 5 Traffic generator process.

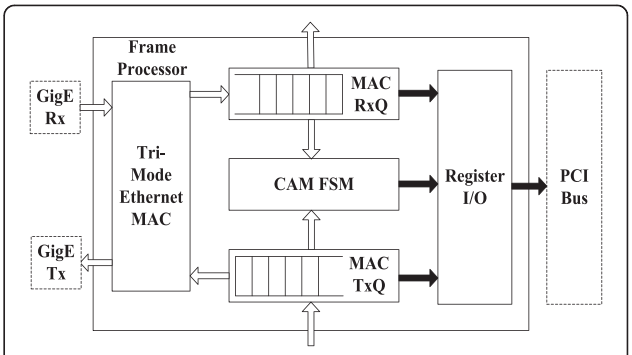


Figure 6 Layout of the frame processor.

4.3 Frame processor

The frame processor uses the passive measurement to count the network traffic, and it also transmits and receives the probe frame. Our approach has four gigabit Ethernet interfaces corresponding to each frame processor. The layout of the frame processor is shown in Figure 6. The MAC TxQ receives the data frames from the traffic generator and also records a tag of each new frame and the frame length simultaneously. When getting the tag and the frame length, it will accumulatively calculate the number of the data frames and total bytes received, etc., which are used separately in Equations 6, 8, and 9. Those enable us to achieve the network traffic through Register I/O in real time. Besides, the MAC TxQ needs to reassemble the data frames from 64-bit to 8-bit wide. Next, the data frames will be sent to *CAM FSM*, and the data frames checked by cyclical redundancy check (CRC) are transmitted to *Tri-Mode Ethernet MAC* (TEMAC) via an 8-bit wide data bus.

The content addressable memory (CAM) is the core of the CAM FSM. At the beginning of the CAM FSM working, the CAM is initialized firstly. With two stage pipelines, the data formed by accumulating the byte which is received by the CAM FSM are used to match the special tags (64 bits) set in the traffic generator. Through the above matching, the matching

mark of the data frame and its valid mark are recorded. For one thing, the total data frame transmission time can be achieved by clock cycles, which is used in Equation 6. For another thing, using the valid mark can help us to avoid invalid statistics about clock cycles when the tag in the data frames is missing.

The TEMAC generated by the intellectual property (IP) core of Xilinx is used to process the data frames. It sends data frames to the *GigE Tx* and receives data frames from the *GigE Rx*. The MAC RxQ is similar with the MAC TxQ, which is just in a different transmission direction. When the MAC TxQ sends a data frame, a counter is opened; while the MAC RxQ finishes receiving that data frame, the above counter is closed by the MAC TxQ immediately. By their collaboration, the latency shown in Equation 2, which is accurate to the nanosecond, can be obtained.

4.4 Interval stretching

The interval stretching is used to dynamically adjust the interframe gap, which enables the traffic generator to generate the perfect network traffic, so the accurate measurements can be conducted. According to the statistical laws based on throughput variation, our approach dynamically adjusts the interframe gap to generate probe data frames and count network traffic again and again. When it approaches the limit of the network, the network performance parameters are achieved. To find an optimized interval is important to get accurate evaluation results. The interval $[a, b]$ represents the interframe gap interval determined by the above sequence of frames $\{P_1,P_2,...,P_n\}$, which is measured for a certain time. The a and b show the clock cycles of the interframe gap. After the measurement, four kinds of rational situations can be determined. The model of interval stretching is shown in Figure 7. The arrows show the changes of the interframe gap.

The first one is that when $\delta = a$, our approach fails to conduct the measurement but while $\delta = b$, the measurement is successful. In Figure 7a, it will adjust to the interframe gap interval $[b^l, b]$. The second one is that when both $\delta = a$ and $\delta = b$, our method fails to perform the measurements. From Figure 7b, it will adjust to the interframe gap interval $[b, b^r]$ after expanding to $[a, b^r]$. The above two situations usually appear early with binary search. The third one is also that when $\delta = a$, our approach fails to conduct the measurement but while $\delta = b$, the measurement is successful. It usually should adjust to the interframe gap interval $[a^r, b]$, that is to say, compressing the interframe gap interval, which is shown in Figure 7c. The last one is that when both $\delta = a$ and $\delta = b$, the measurements are successful. It often should adjust to the interframe gap interval $[a^l, a]$ after extending to the interval $[a^l, b]$, as we can see in Figure 7d, but the above two situations often occur in the late with fine-tuning. The interval stretching is stated in Algorithm 1. In Step 6, the measurement expected represents that when $\delta = a$, it fails but while $\delta = b$, it conducts measurement successfully.

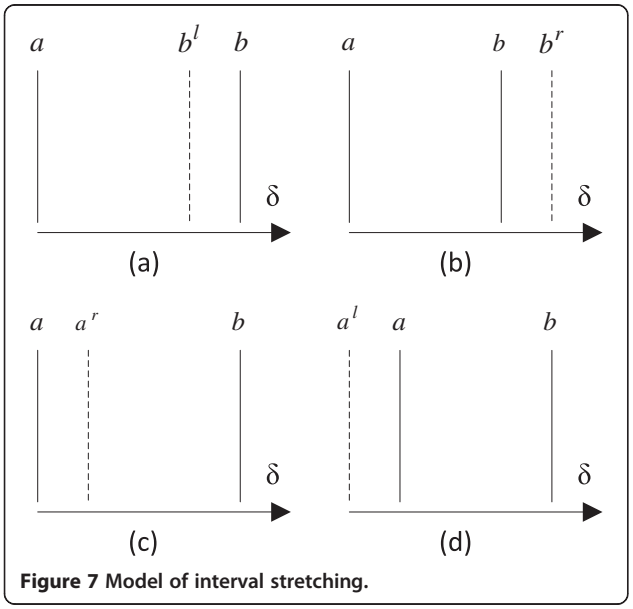


Figure 7 Model of interval stretching.

Algorithm 1 Interframe gap interval stretching	
1.	Get a random value *r* and set *a=0*, *b=r*.
2.	Change the interval with binary search shown in (a) and (b).
3.	Count the number of binary search. If it does not reach the expected value calculated by the initial interval length and frame length, go to Step 2.
4.	Change the interval with fine-tuning shown in (b), (c) and (d) by the step set by the user down to 1 clock cycle.
5.	Get the ideal intervals. If it finds that for 2 intervals of the same *a*, with interval length under 2 and 1 clock cycle, both of them conduct measurement expected, then go to Step 6; otherwise, go to Step 4.
6.	Get the ideal interframe gap *a+1* and the network performance parameters.

5 Experiments and analysis

In order to verify the feasibility of our approach, a high-speed network performance measurement system is implemented. The FPGA-based network performance measurement platform is designed on NetFPGA card version 2 with Virtex II-Pro FPGA device (Xilinx, San Jose, CA, USA), which is shown in Figure 8. The NetFPGA is a PCI card which provides a low-cost reusable hardware platform for network researchers. The bottom card is the NetFPGA in Figure 8. The host computer operates at Intel Pentium Dual E2200 (2.20 GHz) and Cent OS 5.4 (Linux kernel 2.6.18).

We connect one network interface to another on NetFPGA board in the experiments for the ideal network bandwidth 1 Gbps. To facilitate the analysis of the results, the visualization of network traffic information for each Ethernet interface and network performance parameters information are achieved by the software. And we compare our system with another system using the software to send data frames for the same one communication path. Besides, a similar approach based on FPGA [14] is also compared with our approach on the NetFPGA.

5.1 Network performance

For the data frame of different length under different interframe gap, it is sent for 20,000 times. We compare it with the same one communication path for latency, throughput, and packet loss rate.

From Figure 9, the latency of the given frame length converges to a fixed value. With the growth of the frame length, the latency increases accordingly. It can also be found that the latency is not directly proportional to the frame length according to the statistics.

Figure 8 The hardware of measurement platform.

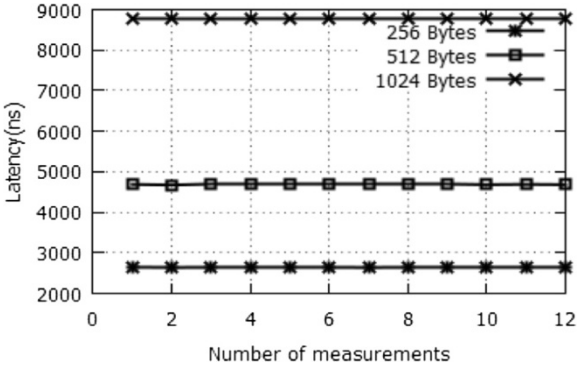


Figure 9 Latency under different frame length.

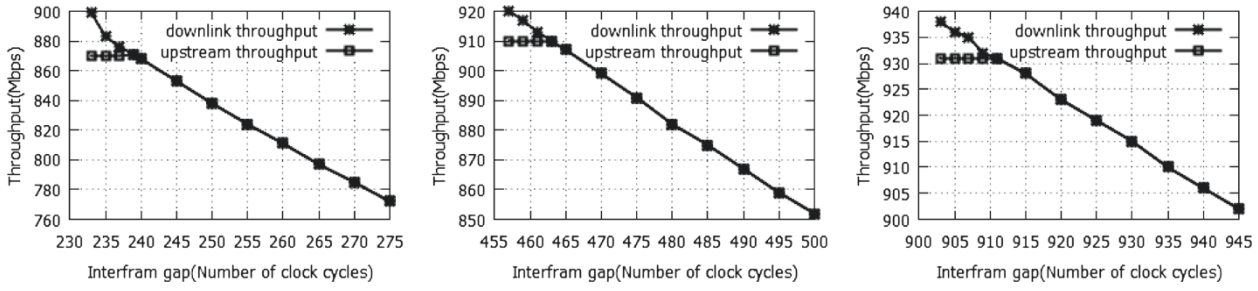
The measurements of throughput for a given frame length under different interframe gap are shown in Figure 10. It can be found that for a given frame length under a specified interframe gap, the throughput decreases clearly with the growth of interframe gap. It is obvious that the downlink throughput is never less than the upstream throughput. From the intersection between the downlink throughput and upstream throughput, the maximum throughput of the communication path can be determined. So, the accurate bottleneck bandwidth can be gained. The bottleneck bandwidth is 871, 910, and 931 Mbps when the data frame length is 256, 512 and 1,024 bytes, respectively. Besides, the frame length under the specific throughput is approximately proportional to the interframe gap, which is drawn by the scope of the interframe gap under different frame length.

Figure 11 plots the correspondence of interframe gap and packet loss rate for a given frame length. It is clear to see that with the growth of interframe gap, the packet loss rate drops to zero and then tends to be stable under the fixed frame length. To put it simply, the bytes' loss rate is only be shown because in the above experiments, the frames' loss rate are all zero. Moreover, with the frame length increasing, the packet loss rate will decline when the measurements approach the limit of network; the packet loss rate also reveals the relationships between the downlink throughput and upstream throughput, which tells us why the downlink throughput is never less than the upstream throughput.

On the whole, with the frame length growing, the network performance of the communication path is improved, that is to say, leaving the throughput increasing and making the packet loss rate decreasing. But the increasing latency is also paid.

5.2 Performance stability

The network performance measurement under the special frame length and the fixed interframe gap is



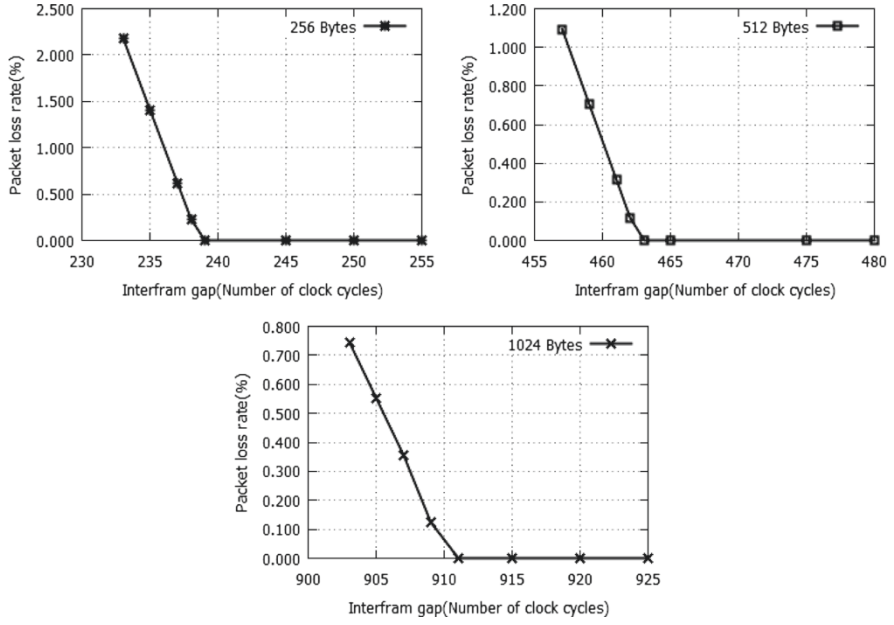
(a) *length = 256 Bytes* (b) *length = 512 Bytes* (c) *length = 1024 Bytes*

Figure 10 Throughput under different frame length.

shown in Table 1. It can be found that for a given interframe gap under the maximum throughput, the performance parameters of throughput, latency, and packet loss rate are stable with the growth of the number of frames. It is obvious that the network performance measurement parameters under different frame length are credible.

5.3 Performance comparison

In order to fully evaluate our approach, the network performance measurement system of our approach is compared with it using software to send probe data frames for the same one communication path. The software system operates at the host computer. It uses the libnet to cyclically encapsulate data frame according to the user's configurations. Then, it forwards those data frames immediately. Meanwhile, in order to monitor the network traffic transmitted and received for each network interface, the above NetFPGA is also used to count the network traffic in the software system. Moreover, we do not wait for the interframe gap when sending those data frames so as to maximize the network performance, namely sending back-to-back data frames to reveal the network performance.



(a) *length = 256 Bytes* (b) *length = 512 Bytes* (c) *length = 1024 Bytes*

Figure 11 Packet loss rate under different frame length.

Table 1 Network performance measurement under special frame length and interframe gap

Frame length (Bytes)	Interframe gap (number of clock cycle)	Number of frame (10,000)	Downlink throughput (Mbps)	Upstream throughput (Mbps)	Latency (ns)	Frames' loss rate (%)	Bytes' loss rate (%)
		1	871	871	2,632	0	0
		2	871	871	2,640	0	0
256	239	3	871	871	2,640	0	0
		4	871	871	2,632	0	0
		5	871	871	2,640	0	0
		1	910	910	4,688	0	0
		2	910	910	4,688	0	0
512	463	3	910	910	4,688	0	0
		4	910	910	4,688	0	0
		5	910	910	4,688	0	0

The probe data frames sent in the NetFPGA and the software for maximum throughput are presented in Table 2. When the frame length is 256 bytes, the maximum throughput is 132 Mbps in the software, which is far below it in the NetFPGA; even if the frame length is increasing to 1,024 bytes, the maximum throughput is merely 220 Mbps, which is far less than the ideal network bandwidth 1 Gbps. On the contrary, our approach can be applied in high-speed network and it also can be migrated to ten gigabit high-speed networks.

A similar work based on FPGA is also compared with our approach. The channel utilization under different platforms is shown in Table 3. For the frames per second, the maximum throughput of 71% of the ideal one is given for 64 bytes in [14]. Its latency is only accurate to microsecond and it is merely presented for an Ethernet 100 Mbps. But the channel utilization of our approach on the NetFPGA can be 100% of the ideal one for 64 bytes under 100 Mbps. And the latency can be accurate to the nanosecond. In addition, when it is deployed in an Ethernet 1 Gbps, the maximum throughput of our approach is 721 Mbps under 66 clock cycles for 64 bytes when the data frames are not missing. It is obvious that our approach has better performance than it.

Table 2 The probe data frames sent in the NetFPGA and software for maximum throughput

Frame length (Byte)	NetFPGA	Software
	Maximum throughput (Mbps)	Maximum throughput (Mbps)
256	871	132
512	910	180
1,024	931	220

6 Conclusions

In this paper, an FPGA-based high-speed network performance measurement solution for RFC 2544 is proposed. The active measurement method is employed to send probe data frames. The passive measurement method is used to monitor network traffic of each Ethernet interface. Our architecture also supports CAM interface to match the special tags with two stage pipelines, which helps us to calculate the throughput and latency precisely. A SRAM interface is provided to access the data frame, which enables our approach to be applied in high-speed network.

An FPGA-based prototype is also implemented for evaluation. It also can be migrated to ten gigabit high-speed networks. The experimental results show that the network performance parameters can be measured accurately. Comparing with the network performance measurement using software to send probe data frames and a similar work based on FPGA, it can be used in high-speed network and the latency can be accurate to the nanosecond.

In order to be applied in the real network widely, generating more complex and real network traffic according to the users' choice will be considered in our future research. Extending our method reliably also will be taken into account. Thus, a more perfect performance will be provided for the users.

Table 3 The channel utilization under different platforms

Frame length (Byte)	Performance index	FPGA	NetFPGA
64	Channel utilization	71%	100%

Competing interests
The authors have declared that no competing interests exist.

Authors' information
Yong Wang was born in 1964. He received his PhD in East China University of Science and Technology, Shanghai, China, in 2005. He is currently a professor at Guilin University of Electronic Technology. His research interests mainly include intelligent control application and network management.
Yong Liu was born in 1987. He received his B.S in Wuhan Textile University, Wuhan, China, in 2011. He is currently a graduate student in the Department of Computer Science and Engineering at the Guilin University of Electronic Technology. His research interest is mainly on compute network and application.
Tao Xiaoling was born in 1977. She received her B.S and M.S in Guilin University of Electronic Technology, Guilin, China, in 1999 and 2008. She is currently an associate professor at Guilin University of Electronic Technology. Her research interest is mainly on network security.
Qian He was born in 1979. He received his PhD in Beijing University of Posts and Telecommunications, Beijing, China, in 2011. He is currently a professor at Guilin University of Electronic Technology. His research interests lie in network security and distribute computing.

Acknowledgements
The manuscript was received last November 2013; accepted February 2014. This work is supported by the National Natural Science Foundation of China (No. 61163058, 61172053, and 61201250).

Author details
[1]CSIP Guangxi Center, Guilin University of Electronic Technology, Guilin 541004, China. [2]College of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China. [3]College of Information and Communication, Guilin University of Electronic Technology, Guilin 541004, China.

Received: 23 December 2013 Accepted: 28 August 2014
Published: 26 June 2015

References

1. YT Han, IY Hwang, CC Kim et al., A new attainable TCP throughput measurement tool for long distance high speed networks. IEEE Commun Lett **14**(10), 990–992 (2010)
2. Y Zhao, M Song, J Wang et al., *Throughput Measurement-Based Access Point Selection for Multi-rate Wireless LANs, Wireless Algorithms, Systems, and Applications* (Springer Berlin Heidelberg, Berlin, Germany, 2009), pp. 509–518
3. S Meiling, TC Schmidt, M Wahlisch, *Large-scale Measurement and Analysis of One-Way Delay in Hybrid Multicast Networks* (IEEE LCN 2012, Clearwater, Florida, USA, 2012), pp. 513–520
4. L Angrisani, D Capriglione, L Ferrigno et al., *Type A Uncertainty in Jitter Measurements in Communication Networks* (IEEE I2MTC 2011, Hangzhou, China, 2011), pp. 1–6
5. F Baccelli, S Machiraju, D Veitch et al., *On Optimal Probing for Delay and Loss Measurement*. Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, San Diego, CA, USA, 2007, pp. 291–302
6. C Wang, J Liu, B Li et al., LRED: a robust and responsive AQM algorithm using packet loss ratio measurement. Parallel Distribute Syst IEEE Transac **18**(1), 29–43 (2007)
7. A Kovac, M Halas, Analysis of influence of network performance parameters on VoIP call quality. Knowledge in Telecommun Technol Optics, 26–30 (2010)
8. H Oda, H Hisamatsu, H Noborio, *A New Available Bandwidth Measurement Method Based on ImTCP* (DICTAP2012, Bangkok, Thailand, 2012), pp. 343–347
9. F Lifu, Y Dongming, T Bihua et al., *Technique for Network Performance Measurement Based on RFC 2544* (IEEE CICN2012, Mathura, Uttar Pradesh, India, 2012), pp. 200–204
10. V Tanyingyong, M Hidell, P Sjodin, *Using Hardware Classification to Improve PC-Based OpenFlow Switching* (IEEE HPSR 2011, Cartagena, Spain, 2011), pp. 215–221
11. S Bradner, J McQuaid, *Benchmarking Methodology for Network Interconnect Devices* (RFC 2544, Fremont, 1999)
12. Z Cai, W Zhao, J Yin et al., *Using Passive Measuring to Calibrate Active Measuring Latency, Information Networking, Convergence in Broadband and Mobile Networking* (Springer Berlin Heidelberg, Berlin, Germany, 2005), pp. 198–206
13. YA Wang, C Huang, J Li et al., *Queen: Estimating Packet Loss Rate between Arbitrary Internet Hosts, Passive and Active Network Measurement* (Springer Berlin Heidelberg, Berlin, Germany, 2009), pp. 57–66
14. CB Both, C Battisti, FA Kuentzer et al., FPGA implementation and performance evaluation of an RFC 2544 compliant Ethernet test set. Int J High Perform Syst Architecture **2**(2), 107–115 (2009)
15. J Wu, J Zhang, Z Han et al., The Implementation of a High Speed Ethernet Traffic Generator Based on FPGA. Adv Mater Res **433**, 7530–7534 (2012)
16. A Tockhorn, P Danielis, D Timmermann, *A Configurable FPGA-Based Traffic Generator for High-Performance Tests of Packet Processing Systems* (ICIMP 2011, St.Maarten, The Netherlands Antilles, 2011), pp. 14–19
17. A Friedl, S Ubik, A Kapravelos et al., *Realistic Passive Packet Loss Measurement for High-Speed Networks, Proceedings of the First International Workshop on Traffic Monitoring and Analysis* (Springer Berlin Heidelberg, Berlin, Germany, 2009), pp. 1–7
18. F Strohmeier, P Dorfinger, B Trammell, *Network Performance Evaluation Based on Flow Data* (IEEE IWCMC 2011, Istanbul, Turkey, 2011), pp. 1585–1589
19. T Jones, *Network Performance Measurement: Accuracy in Highspeed Provide Networks* (IEEE MIPRO 2011, Opatija, Croatia, 2011), pp. 489–493
20. MD Schiffman, *The Libnet Packet Construction Library*, 2005. http://libnet.sourceforge.net/

doi:10.1186/1687-1499-2015-2
Cite this article as: Wang *et al.*: **An FPGA-based high-speed network performance measurement for RFC 2544.** *EURASIP Journal on Wireless Communications and Networking* 2015 **1**:2.