Experimental Study on the Performance of Linux Ethernet Bonding

Hoang Tran-Viet[1(✉)], Toan Nguyen-Duc[1], Kien Nguyen[1,2], Quang Tran Minh[2], Son Hong Ngo[1], and Shigeki Yamada[2]

[1] Hanoi University of Science and Technology, 1 Dai Co Viet Road, Hanoi, Vietnam
{hoang.tranviet,toan.nguyenduc1}@hust.edu.vn, sonnh@soict.hust.edu.vn

[2] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
{kienng,quangtran,shigeki}@nii.ac.jp

Abstract. Linux bonding is a feature allowing to group multiple physical network interfaces into a logical one on Linux machines. Known as a low-cost method to improve fault tolerance and network throughput, the Linux bonding with seven supported modes is increasingly deployed in various scenarios such as datacenters, home networks, etc. However, the strengths and weaknesses of different modes have not been well investigated. While previous works mostly pay attention on the performance of the popular round-robin mode, this work extensively and additionally evaluates other modes based on three major criteria: throughput improvement, load balancing, and fault tolerance. To the best of our knowledge, this is the first work investigating the capabilities of fault tolerance using Linux bonding. The evaluation results show that the active-backup mode achieves the flow switch-over time, which is the duration of traffic flow discontinuation due to a network failure, as small as 10 milliseconds. Moreover, in the round-robin mode with two bonded network interfaces, Linux machines can achieve the maximum throughput close to double of that in case of non-bonding. However, the out-of-order and switch compatibility issues may limit the utilisation of the round-robin mode in certain scenarios. In the 802.3ad mode, the out-of-order issue can be avoided, although load balancing is not always optimal.

Keywords: Linux bonding · Link aggregation · Fault tolerance · Switch-over · Throughput improvement · Load balancing

1 Introduction

With the continuous development of online applications and services, network operators and service providers have to deal with an ever-increasing bandwidth demand [1]. On the other hand, the customers expect their network services always available, despite the fact that every element in network can fail.

V.C.M. Leung et al. (Eds.): TridentCom 2014, LNICST 137, pp. 307–317, 2014.
DOI: 10.1007/978-3-319-13326-3_29

A common approach for increasing network resilience and throughput is to combine multiple communication links. In the transport layer, there are many recent works focus on multipath TCP [2,3]. In the network layer, equal cost multipath routing is widely adopted [4]. In the lower layers, the combination is usually done by *interface aggregation*, which means grouping multiple physical network interfaces to form a single logical one. There are several flavours of interface aggregation in networks today. On networking devices, this feature is called EtherChannel [5] by Cisco, or Link Aggregation in 802.3AD standard [6] by IEEE. On servers, this feature is implemented as *Ethernet bonding* [7] in Linux environment, or NIC teaming in Windows environment.

Ethernet bonding feature is currently deployed in various datacenters, server systems, and even in home networks [8]. Additionally, the Ethernet bonding provides a valuable tool for research, for example in building high performance systems [9] or maintaining connectivity during virtual machine migration [10,11]. Despite the popularity of Linux bonding in industrial and research environments, its performance has not been well investigated. For that reason, we extensively measure the performance of different Linux bonding modes. Further, this work is potentially a base for network designers and researchers to choose the right bonding mode for their specific network scenarios and requirements.

Different to previous evaluation works [12,13], we evaluate the performance of Linux bonding based on three major criteria: throughput enhancement, load balancing efficiency, and fault tolerance. To the best of our knowledge, this is the first work evaluating the fault tolerance capability of Linux bonding. Moreover, apart from round-robin mode, several bonding modes are also investigated in our evaluation. The experiment results show that the UDP flow switch-over time is in the range of 5 - 20 milliseconds with active-backup bonding mode. In another experiment, by bonding two physical interfaces in round-robin mode, the transfer duration of a file in FTP is decreased by nearly a half.

To accomplish our evaluations, there are several technical challenges must be overcome. First, measuring the switch-over time of traffic flow is not trivial. In a high-speed network, measurement tools may not work fast enough in recording network events or capturing packets. On the other hand, we need to make sure that our measurements do not impose heavy load on system and affect network performance. Another issue is related to failure detection feature of Network Interface Cards (NICs) hardware. Many NICs do not detect link failures in a timely manner, resulting in a very long flow switch-over time. This requires testing NICs carefully before using them for bonding both in experiment networks and in production networks.

The rest of paper is outlined as follows: after Section 2 mentioning the related work, Section 3 provides a brief overview on Linux bonding feature. In Section 4, after the description of experiment set-up, we present our evaluation results and discussions. Finally, conclusion remark is given in Section 5.

2 Related Work

There are various works related to investigating Linux bonding. The most close ones to our work are [12] and [13], in which the authors evaluated the performance of wired and wireless bonding accordingly. However, the fault tolerance, which is an important aspect of Linux bonding, was not considered in these works. Moreover, the works only examined the round-robin mode although there are seven different modes in Linux bonding. In [8], the authors evaluated Linux bonding in a home network environment. This work analysed the packet loss and throughput improvement of round-robin mode and broadcast mode with varied attenuation level.

3 Overview of Linux Bonding

Linux bonding means aggregating several NICs together into a group on Linux machines. This group of interfaces appears as a single interface to higher network layers. In the Linux bonding, physical interfaces in the group are called *slaves*, while the logical interface is called *master*. When a packet is sent from higher layer to the master interface, the bonding driver will deliver this packet to one (or more) slave interface. Packets can be delivered in several ways, depending on which mode the bonding driver is in. The process of receiving packets is similar to sending process. When a packet comes to a slave from a remote host, the bonding driver will decide to direct this packet to the master or to drop it, depending on the bonding mode.

Linux bonding provides fault tolerance by monitoring link or NIC failure and switching traffic from failed slave to other slaves. Linux bonding uses two methods for link monitoring. The first one is MII monitoring, which relies on NIC driver to maintain its knowledge about local link status. MII monitoring is the default option in Linux bonding and can be used with all modes. The second method is ARP monitoring. With this method, the bonding driver maintains a list of remote peers, periodically sends ARP requests to these peers and monitors whether slave interfaces still send or receive traffic recently. At the time of writing, the latest version of Linux bonding (v3.7.1) supports 7 different bonding modes. Table 1 shows a brief overview [7] of each mode.

Round-Robin Mode. In this mode, each packet is sent to one slave interface in turn. For example, in case of a bonding group with two slave interfaces, the first packet will be sent through slave 1, the second one will be sent through the slave 2, the third one will be sent through slave 1, and so on.

Active-Backup Mode. In this mode, there is one interface is active at a time. Other slaves are in standby state and do not send or receive packets. When the active one is failed, one backup slave will be chosen as the new active one.

Table 1. List of bonding modes

Mode	Fault tolerance	Throughput enhancement
round-robin	Yes	Yes
active-backup	Yes	No
balance-xor	Yes	Yes
broadcast	Yes	No
802.3ad	Yes	Yes
balance-tlb	Yes	Yes
balance-alb	Yes	Yes (for both incoming and outgoing traffic)

Balance-Xor Mode. In this mode, outgoing traffic is balanced per flow of packets. While different flows may be sent over different slaves, all packets belongs to a flow will be sent through the same slave. This ensures all packets of a flow will in order at destination host. Specifically, for each packet coming to bonding group, the bonding driver will decide to send it through which slave by using a XOR hash function on its header information. By default, the outgoing traffic is balanced based on layer 2 information only. Each packet will be sent through the slave with the $SlaveID$ determined by:

$$SlaveID = (srcMAC \oplus destMAC) \mod N$$

where $srcMAC$ and $destMAC$ are the source MAC address and destination MAC address; N is the number of live slaves in a bonding group. This algorithm let all traffic between two MAC clients go through the same slave. Linux bonding also supports load-balancing based on layer 3 (IP addresses) or layer 4 (TCP or UDP port numbers) information:

$$SlaveID = (srcPort \oplus destPort) \oplus (srcIP \oplus destIP) \wedge FFFFh) \mod N.$$

Broadcast Mode. Each outgoing packet is copied and broadcast on all bonded interfaces. Currently, this mode is not widely used in real networks.

802.3ad Mode. 802.3ad is an IEEE standard for link aggregation [6], which also includes Link Aggregation Control Protocol (LACP), between two networking devices. In this mode, a Linux host can connect to a LACP-enabled switch through a group of aggregated links. The limitation of this mode is that the 802.3ad requires all links running in full duplex mode at the same speed. Outgoing traffic from Linux host is distributed by the same algorithm as in the balance-xor mode.

Balance-tlb Mode. In this mode, outgoing traffic is balanced per remote host. However, incoming TCP/UDP traffic is always received on one slave.

Balance-alb Mode. This mode is almost similar to the balance-tlb mode, but has a difference in balancing the incoming traffic. By intercepting ARP packets, the bonding driver can decide that traffic from a specific IP address will be

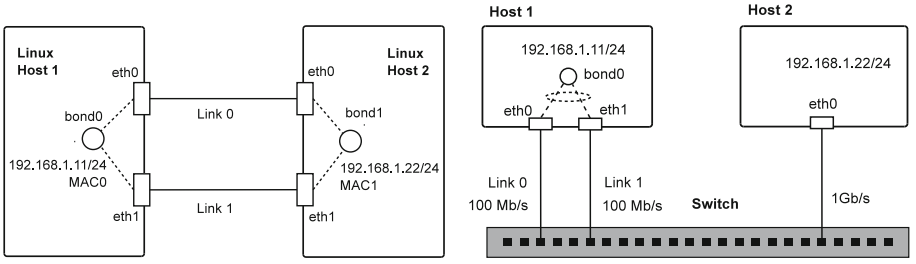
received on which slave. However, the interfaces must be able to change their MAC addresses while they are running. The feature that is not supported by commodity NICs is also a big disadvantage of balance-alb mode.

4 Evaluation Method

In this section we present the evaluation of three bonding modes: round-robin mode, active-backup mode, and 802.3ad mode. These modes are chosen because they represent well for three major criteria on which we want to focus: throughput improvement, fault tolerance, and load balancing.

4.1 Experiment Set-Up

Each bonding mode has been evaluated using two network topologies. The first one is host-to-host bonding, as illustrated in Fig. 1a, in which two Linux computers are connected directly through a group of bonded links (i.e., CAT5e Ethernet cables). Each computer (Core i5 Ivy Bridge, 4GB RAM, Ubuntu 13.10 64-bit, kernel version 3.11.0-14, with the latest version of bonding drivers) has several network cards with the same negotiated speed at 100 Mbps. This topology allows us to conveniently monitor how traffic is distributed among bonded links. In real deployments, high performance clustering systems can use this kind of bonding topology. The second topology is shown in Fig. 1b, in which the same Linux computers are used. However, the computers connect to a switch, and the bonding configuration is enabled on one computer (i.e., host 1). In the topology, we use the Pica8 P3295 switch, which has 48 Gigabit Ethernet Ports and supports LACP. The topology allows us to explore how computers with bonding interoperate with networking devices.



(a) Topology 1 (host-to-host) (b) Topology 2 (host-switch-host)

Fig. 1. The evaluation topologies

During the bonding initialisation process, it is essential to specify the link monitoring interval; otherwise the mechanism of link failure detection will be disabled. In our evaluations, the monitoring interval is set to 1 ms, which is the minimum value supported on Linux.

4.2 Round-Robin Mode Evaluation

Round-robin is the only mode allowing a TCP/UDP flow to be spread across multiple links. In this evaluation, we use Iperf [14] to generate a UDP flow with 500 Mbps offered rate from host 1 to host 2. Figure 2 shows the total throughput with different UDP datagram sizes in case of topology 2 (Fig. 1b). In case of topology 1 (Fig. 1a), the results are similar. As shown in the figure, when the size of UDP datagram is less than 1472 bytes, the total throughput provided by two NICs or three NICs bonding is nearly double or triple compared to that of non-bonding scenario, respectively. However, the UDP datagrams larger than 1472 bytes are fragmented. With round-robin bonding, different parts of a UDP datagram are sent through different links. Combining with out-of-order issue, these fragments are not re-assembled correctly at the receiver. We also investigate the behavior of round-robin mode in response to link failures. The results for TCP traffic with topology 2 are shown in Fig. 3a. A failure of link 1 (eth1) occurs at $t = 5s$, and then the link is recovered after 5 seconds. We observed

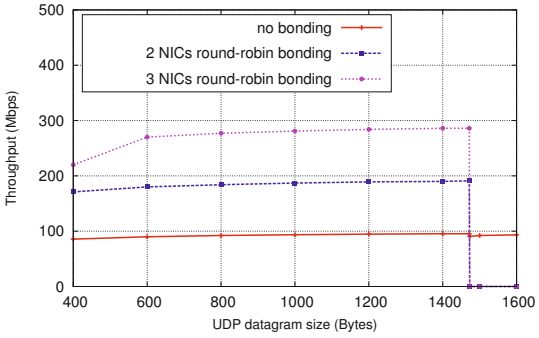
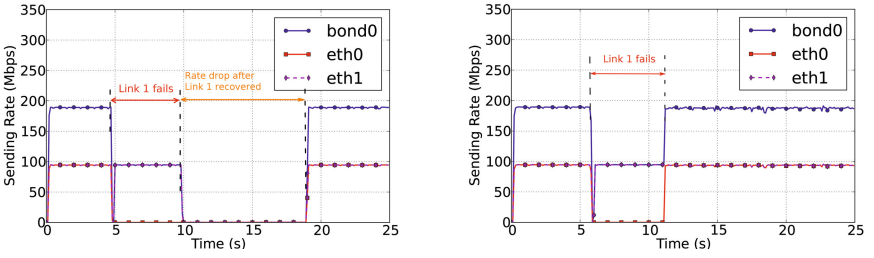


Fig. 2. Round-robin mode, UDP traffic, throughput on Iperf server



(a) Default switch configuration (b) Static Link Aggregation enabled

Fig. 3. Round-robin mode, TCP traffic with a link failure on link 1

that even after this link is recovered, from $t = 10s$ to $t = 19s$, the transfer rate is dropped nearly to zero on both slaves. This issue does not happen with topology 1. The root cause of this issue is that the Pica8 switch is not aware that two links are bonded at Linux host, hence the packets go through newly recovered link are dropped at the switch. Although link 0 is still alive, the packet loss on link 1 triggers TCP congestion control to reduce congestion window to 1, hence reducing the sending rate close to zero. To solve this problem, it is necessary to aggregate these switch ports by using Static Link Aggregation (SLA), that is known as 802.3ad without LACP. With SLA enabled, the switch can quickly determine that an aggregated link is recovered, and correctly receive packets from recovered link, as the results of same evaluation in Fig. 3.

Table 2. Round-robin mode, 361 MB file transfer time in FTP (average in 10 runs)

	361 MB file transfer time
no bonding	32.12 seconds
round robin	16.20 seconds

In the case of TCP, significant out-of-order issue will trigger congestion control at the sender to reduce the sending rate. To investigate how this issue affects the throughput performance from application level, we conduct a simple FTP traffic test. We send a 361 MB file in the binary mode of FTP from host 1 to host 2 in topology 2 and measure the file transfer time (FTT). Table 2 shows the FTT value in two cases: host 1 connects to switch by a single link (no bond) or by two bonded links. The results show that FTT has been reduced to nearly a half with round-robin mode, and out-of-order issue only imposes a minor overhead on network performance in this application.

4.3 Active-Backup Mode Evaluation

Active-Backup mode is specifically designed for fault-tolerance, so we focus on evaluating how fast this bonding mode can switch traffic to a backup link when the active link fails. For this purpose, we use flow switch-over time (FST), which is defined as the duration of traffic flow discontinuation due to a network failure, as a criterion to assess fault-tolerance capability. We determine FST as the duration from the time of last data packets on old active slave (that is, just before link failure) to the time of the first data packets on new active slave.

We do the experiments with both topology 1 and 2. While traffic is being sent, failures on the active link happen for every 10 seconds. With the topology 1, we capture all incoming data packets at the receiver (host 2) using Wireshark [15]. Figure 5a shows the FST values of UDP and TCP traffic, each case is measured in 20 times. In the case of UDP, the FST values fall in the range of 5 - 20 ms, however in the case TCP flow, the FST values are divided into two groups:

normal and huge. The huge group contains the values higher than 200 ms. In these cases, the congestion window is reduced to 1, which means that the TCP transmission timeout (RTO) has occurred. This happens if the TCP sender has sent all packets in congestion window, and it has not received acknowledgements during the RTO. It is noted that the minimum value of RTO in Linux TCP is 200 ms by default.

With topology 2, above method cannot be used to measure FST, since all packets are received on a single NIC at the receiver (host 2). In this case, we use another method which based on the port-mirroring feature. Specifically, we mirror the traffic from the port which associated with link 0 to another NIC (eth3) of host 1, as shown in Fig. 4. By using port mirroring, we can determine correctly the FST as the duration from the last packet captured on eth3 and the first packet on eth1. As shown in Fig. 5, the FST results measured with two topologies are similar.

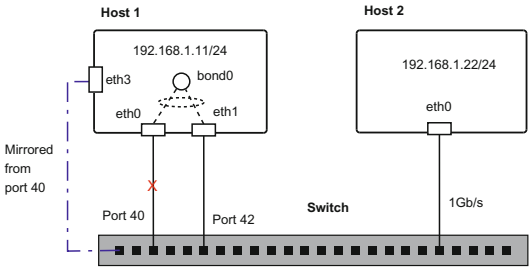
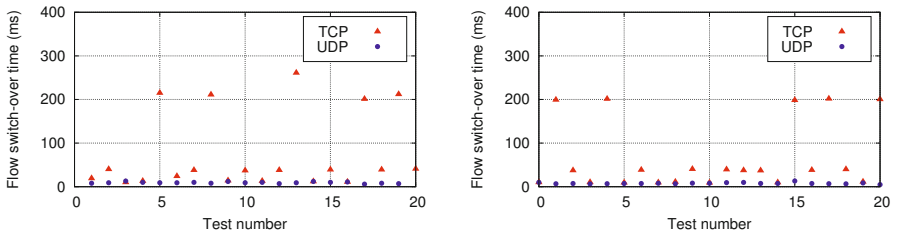


Fig. 4. Measuring FST in topology 2 with port mirroring



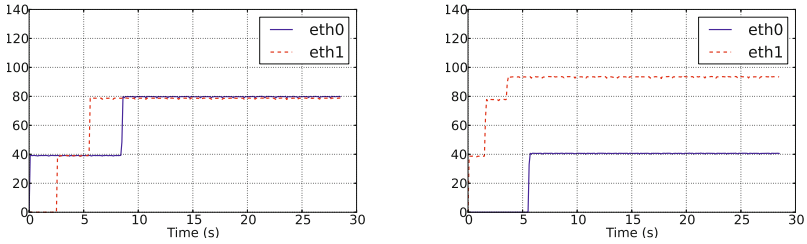
(a) Topology 1 (b) Topology 2

Fig. 5. Flow switch-over time, active-backup mode

4.4 802.3ad Mode Evaluation

This evaluation focuses on load balancing capability supported by hashing algorithms. In the evaluation, we use "layer 3+4" load balancing option, which is based on both IP addresses and layer 4 port numbers. This allows different

TCP/UDP flows between two hosts to be distributed among multiple slaves. The 802.3ad bonding mode works in the two topologies with similar behaviors. The traffic is set up and load-balanced toward the direction from host 1 to host 2. We create four 40 Mbps UDP flows with random UDP port numbers. After that, we measure the receiving rate at both slave interfaces on the host 2, and the results are shown in Fig. 6a. The figure clearly shows the load balancing of the traffic. We repeat same procedure but using the different traffic flows. The link 1 is saturated with one full 100 Mbps flow and three other 40 Mbps flows are on link 0. The results in Fig. 6b show that the load balancing is not optimal. Even though, the link 1 is full, other traffic flows are not switched to the other available link.



(a) 2 flows on each link (b) 1 flow on link 0, 3 flows on link 1

Fig. 6. Receiving rate on two links in the Topology1 of 802.3ad evaluation

To further understand the efficiency of the load balancing mechanism, we conduct experiments using two different traffic patterns: pattern 1 with four 10 Mbps flows and pattern 2 with sixteen 2.5 Mbps flows. The direction of flows is from host 1 to host 2. In each case of traffic pattern, we measure the sending and receiving rate in 100 different runs to see how load balancing mechanism allocates flows on each link. Figure 7 shows the number of flows on link 1 with each traffic pattern. For the 4-flow pattern, the worst situation is that all 4 flows run on only one link happened in 10% (10/100 runs). For the other pattern, the situation that all flows run on a single link did not happen in 100 runs. These results indicates the load balancing mechanism of this mode works better in case

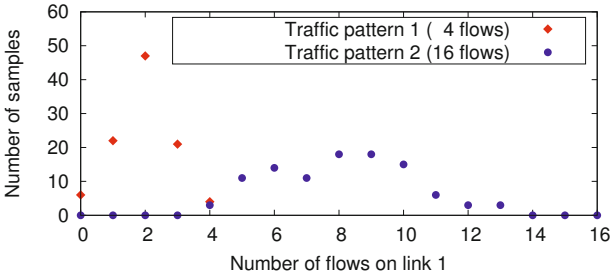


Fig. 7. Distribution of flows on 2 links, 100 runs in 802.3ad mode

of a large number of flows. Furthermore, the evaluation suggests that when the UDP/TCP port numbers are random, the number of flows on a link follows a binominal distribution.

5 Conclusion

In this paper, we have presented the performance evaluation of the three typical Linux bonding modes (i.e., round-robin, active-backup, and 802.3ad modes) on three major criteria: throughput improvement, fault tolerance, and load balancing. With no packet fragmentation, the round-robin mode with two bonded NICs can provide the throughput nearly double of that in the non-bonding case, even for a single flow. However, when the fragmentation happens, the out-of-order issue will affect the throughput performance severely. Moreover, we point out that this mode requires the adjacent peer (i.e., switch) to aggregate ports in order to achieve fault tolerance. In contrast to the round-robin mode, the active-backup mode does not require any special support by the switch and neither suffer from out-of-order issue. The active-backup mode, which is originally designed for the fault tolerance purpose, can provide the FST in the range of 5 - 20 ms with UDP traffic and less than 300 ms with TCP traffic. In the 802.3ad mode, the Linux host can dynamically cooperate with an LACP-enabled switch following in a 803ad standard (or LACP). The hashing algorithms allow the Linux host to load balance outgoing traffic among several physical links; however, the traffic distribution is not optimal.

References

1. Armbrust, M., et al.: A view of cloud computing. ACM Communications **53**(4), 50–58 (2010)
2. Ford, A., Raiciu, C., Handley, M., Bonaventure, O., et al.: TCP Extensions for Multipath Operation with Multiple Addresses. RFC6824 (IETF) (2013)
3. Barré, S., Paasch, C., Bonaventure, O.: MultiPath TCP: From Theory to Practice. In: IFIP Networking, Valencia (May 2011)
4. Augustin, B., Friedman, T., Teixeira, R.: Measuring multipath routing in the Internet. IEEE/ACM Transactions on Networking **19**(3), 830–840 (2011)
5. Cisco EtherChannel - White Paper (accessed January 18, 2014), http://www.cisco.com/en/US/tech/tk389/tk213/tech_white_papers_list.html.
6. IEEE Std 802.3ad-2000 (2000)
7. Davis T., et al.: Linux Ethernet Bonding Driver HOWTO (2011), http://www.kernel.org/doc/Documentation/networking/bonding.txt
8. Yu, Y., Pan, J., Lu, M., Cai, L., Hoffman, D.: Evaluating no-new-wires home networks. In: 33rd IEEE Conference on Local Computer Networks, pp. 869–875. IEEE (2008)
9. Cope, J., Oberg, M., Tufo, H.M., Woitaszek, M.: Shared parallel filesystems in heterogeneous Linux multi-cluster environments. In: Proceedings of the 6th International Conference on Linux Clusters (2005)

10. Dong, Y., Yang, X., Li, J., Liao, G., Tian, K., Guan, H.: High performance network virtualization with SR-IOV. Journal of Parallel and Distributed Computing **72**(11), 1471–1480 (2012)
11. Zhai, E., Cummings, G.D., Dong, Y.: Live migration with pass-through device for Linux VM. In: Proceedings of the 2008 Ottawa Linux Symposium, pp. 261–268 (2008)
12. Aust, S., Kim, J.-O., Davis, P., Yamaguchi, A., Obana, S.: Evaluation of Linux Bonding Features. In: Proceedings of IEEE International Conference on Communication Technology, pp. 1–6 (2006)
13. Jayasuriya, A., Aust, S., Davis, P., Yamaguchi, A., Obana, S.: Aggregation of Wi-Fi links: When does it work? In: Proceedings of IEEE 15th International Conference on Networks, pp. 318–323 (2007)
14. Iperf: The TCP/UDP bandwidth measurement tool (2014)
15. Wireshark - a network protocol analyzer (2014)