THE IMPLEMENTATION OF THE NEW PROTOCOL FOR HIGH SPEED INTERNET ROUTERS/SWITCHES - ROACM

Abdullah Ali Bahattab
Computer Technology Department, College of Telecom. and Electronics, Jeddah, Saudi Arabia
abahattab@gmail.com

ABSTRACT: *In this paper, we focus in implementing the Route once and cross-connect many (ROACM) new protocol as a software based on Data Plane Development Kit (DPDK) application. We used the C programming language for coding. In the previous study, the network simulation was used to conduct the comparison between the TCP/IP and the ROACM. The ROACM protocol was considered from the source workstation to the destination workstation. This would enforce all users to update their web browser with ROACM. So, in this paper, we make the implementation of the ROACM starts from source edge router to destination edge router. The operating system for ROACM was built by using the open source network routing suite "Quagga" and the "DPDK". The results show that the ROACM outperforms the TCP/IP protocol.*

KEYWORDS: ROACM, high speed network, Forwarding Packet, Cross-connect, IP addresses, Routing

. 1 INTRODUCTION

ROACM is a new novel protocol [1,2], which does route once and cross connect many by creating dynamic virtual circuits for each connection/session. In [3], the research that we did was based on the comparison between the TCP/IP and ROACM protocols using the network simulation tool (NS3) to study different aspects of performance analysis. They are the average delay, and throughput. In [3] two network simulation scenarios were created. One is for the TCP/IP while the other is for the ROACM. Each scenario consists of a source and a destination workstation, and between them are twenty routers, see Figure. 1. In this paper, we decided to make the protocol starts the session from a source edge router to a destination edge router instead of the source and destination workstations. This would make the protocol transparent to the users as in MPLS, ATM, and Tag switching [4,5,6,7,8,10,15,16.17], see Figure 2. Thus, in this paper, the implementation will be based in this concept.

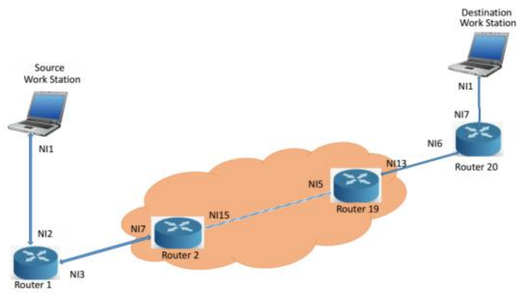


Figure 1. A network with twenty hopes from source to destination work stations.

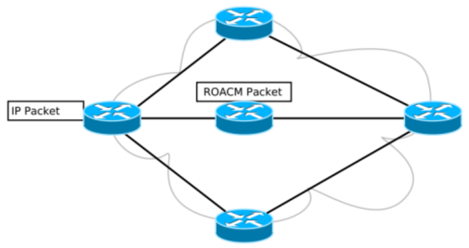


Figure 2. A network using ROACM edge to edge routers.

2 SYSTEM OVERVIEW

The proposed solution implements ROACM protocol as a software based on Data Plane Development Kit (DPDK) application, see Figure 3. It handles the ROACM protocol, synchronizes with kernel routing table, and forwards the packets either through the routing table look up at the call set up stage or ROACM specific local routing interface table lookup at the data transmission stage. DPDK is a set of data plane libraries and network interface controller drivers for fast packet processing. DPDK provides a programming framework for x86, ARM, and PowerPC processors and enables faster development of high speed data packet networking applications. The ROACM solution can run in both host and virtualized environment (VM), see Figure 4. The overall system consists of a) Quagga and b) ROCAM Application.

2.1 QUAQQA.

Quagga is used in this solution to provide control plane support for routing. It is an open source software-based package which provides TCP/IP based routing services with different routing protocols like OSPF, BGP, RIP, RIPng etc. for Unix-liked platform (Linux, Solaris, FreeBSD, and NetBSD). It uses an advanced software architecture which offers a high quality, multi-server routing engine.

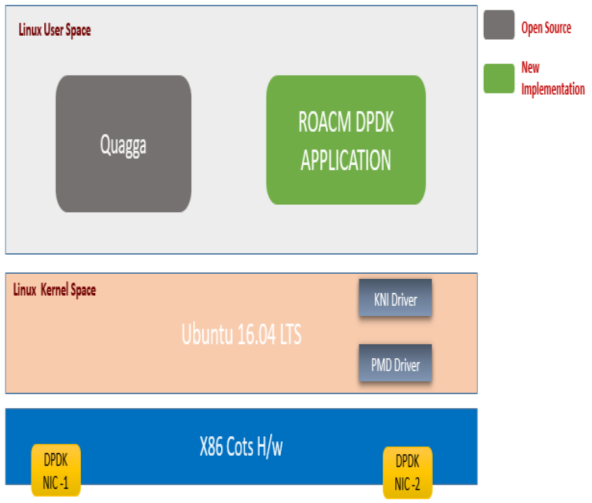


Figure 3. System Overview for Physical Environment.

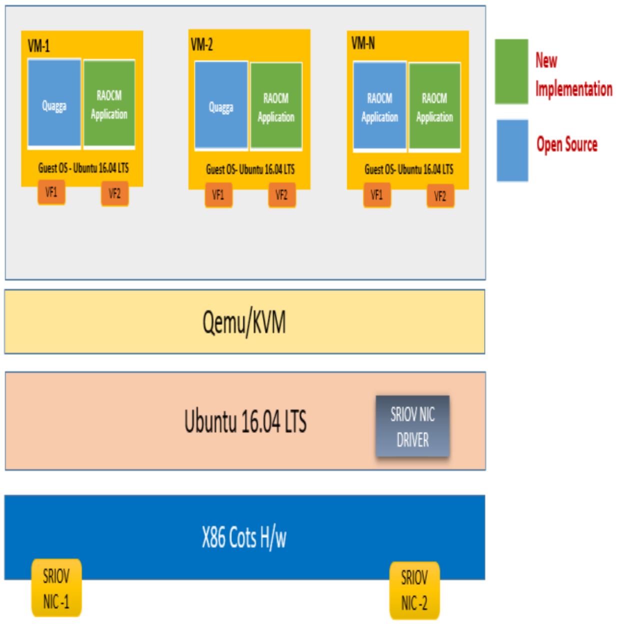


Figure 4. System Overview for Virtual Environment.

Quagga architecture consists of a core daemon(zebra) and several protocol specific daemons which work together to build the routing table, see Figure 5.

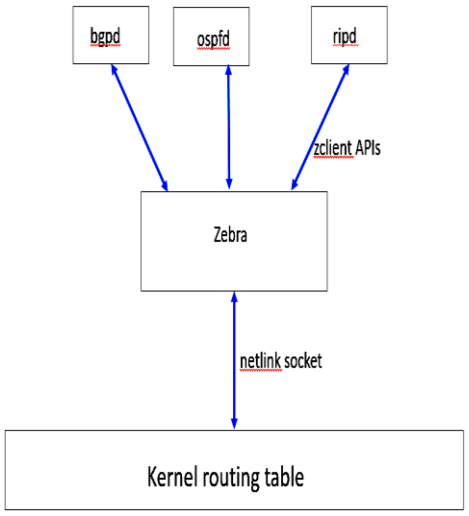


Figure 5. Quaqqa Architecture.

2.2 ROACM APPLICATION.

The ROACM solution is based on multicore DPDK application, see Figure 6. One core (Master/Sync core) is dedicated for synchronization activity like route sync up and other cores take part in fast path packet (ROACM Protocol) processing. The fast path cores or worker cores perform the packet processing in a run-to-completion mode and each of the worker cores are uniquely associated with RX/TX queues per NIC.

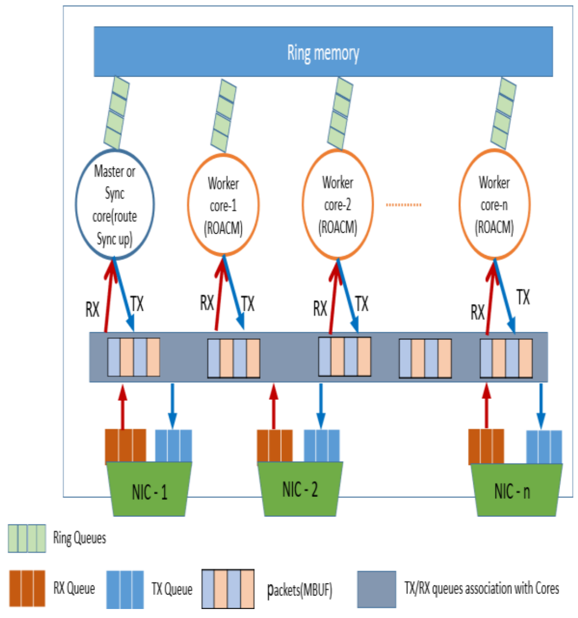


Figure 6. ROACM DPDK Application Components

2.3 SOFTWARE DESIGN.

The ROACM solution consists of the following software components:

Sync Module, Ingress Module, Egress Module, Parser Module, De-Parser Module, ARP Module, Routing Module, Kernel NIC Interface (KNI) Module, and ROACM Protocol Block, see Figure 7.

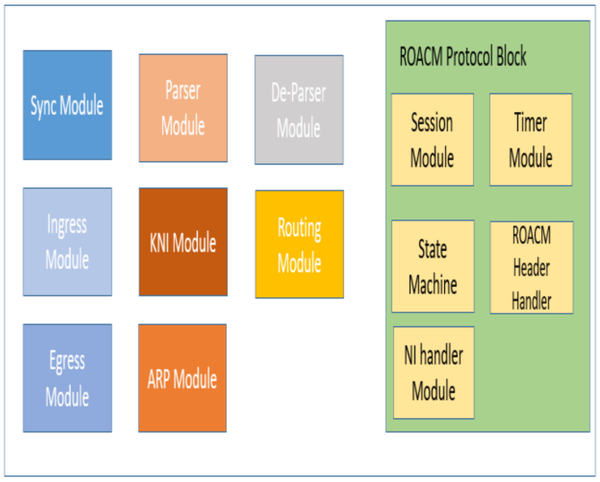


Figure 7. ROACM Software Components

2.3.1 SYNC MODULE

The functionality of the Sync Module is to synchronize with kernel routing table. Quagga updates the kernel routing table based on the control application like OSPF. Sync module will register with the kernel for route updates using net-link socket (RTMGRP_IPV4_ROUTE and RTMGRP_NOTIFY flags). These updates will be propagated to the routing module. The routing module will store the information in the routing table.

2.3.2 INGRESS MODULE

The ingress module will poll the RX queues of the ports to receive the packets in bursts. The burst size is preconfigured. The packets will be passed to parser module one by one.

2.3.3 EGRESS MODULE

The Egress module will receive the packets for transmission through a specific TX queue of a port. To achieve the best performance instead of sending a single packet, packets are buffered and sent out together once it reaches the preconfigured burst size. To handle TX failure scenario, packets can be retransmitted with the preconfigured retry count.

2.3.4 PARSER MODULE

This module will be used to parse the packet headers and forward to the appropriate module for further processing. The KNI module is running in both Sync and Worker Cores. All control packets (ARP, ICMP etc.) will be directly sent to the to the KNI module running in the Worker Core. A copy of an ARP Packet will be sent to the ARP module for MAC/IP learning. As ARP module runs on Master/Sync Core, ring buffer will be used to send the MBUF from Worker to Sync Core. All Other IP packets will be sent to ROACM protocol block for ROACM related processing. Before sending the packet to ROACM protocol block, parser module will remove L2 header. This can be implemented using DPDK API's for MBUF processing.

2.3.5 DE-PARSER MODULE

De-Parser Module will add the data link layer or L2 header in the packets before sending to the Egress Module for transmission and fetches the destination MAC address from IP/MAC Table and updates the L2 header.

2.3.6 ARP MODULE

The ARP module receives the ARP request/Response packets from the Parser Module. IP/MAC information is extracted from the packets and will be updated in the IP/MAC table. This table will be implemented using in built DPDK HASH library where IP address will be used as a key. This table will be used by the routers to fetch the MAC address for a given IP address.

2.3.7 ROUTING MODULE

The Routing module will add/update/delete routes from routing table. This module receives route information form the Sync Module. Other modules use the route module to fetch the routing information. This routing table will be implemented using in built DPDK LPM library. The DPDK LPM library component implements the Longest Prefix Match (LPM) table search method for 32-bit keys (IP Address).

2.3.8 KNI MODULE

Both Sync core as well as worker cores uses this Module. Sync Core uses this module to receive the packets from the KNI interfaces and will be sent directly to the Egress Module. Worker Core uses this module to send all the control packet (ARP, ICMP, etc.) to the KNI interface.

2.3.9 ROACM PROTOCOL BLOCK

The session module will be used to maintain session related information in the Edge routers. A session table will be used to keep the session Information. Edge routers will use the Source IP and Destination IP combination (OR operation) as Key for the session table. This table will be implemented using in-built DPDK HASH library. The State Module will maintain the state machine per flow-based session. ROACM has 4 different states:

a. Call set up
b. Data transmission
c. Path update
d. Path recovery.

Based on the state of a session, ROACM header and ROACM data is added to the packet for edge routers. Other routers will add or removes the
RACOM data based on the control field of the ROACM header. Timer module will handle the path update timeout for a session. Update timer is necessary to get the optimal path. Once the session gets created, the path update timer will start. The timeout value is preconfigured. Once the timeout happens, path update event will be triggered. This can be implemented by calculating the difference between previous and current CPU cycles. There are in-built DPDK APIs (rte_get_timer_hz(), unlikely() etc.)are available to achieve this functionality. The NI handler module will handle the NI (Network Interface) index table. Information on these tables are populated during the following stages.

3 PERFORMANCE ANALYSIS & RESULTS

In the previous research [3], we considered the comparison between the TCP/IP and ROACM protocols using the network simulation tool (NS3), but in this research we have coded the ROACM protocol using C programing language. Also, we implemented the virtual machines on DELL servers each one with the following specifications; PowerEdge R640-8 x 2.5 Chassis, Intel Xeon Silver 4110 – 2.1 GHz – 8 Core, and 8 GB DDR4. In fact, we have one server for the control and in the other two servers on which we install the virtual machines (virtual routers), and all servers are connected as a LAN. In each server of the two ones, we have 4 instances (virtual routers), a total of 8 virtual routers. We transferred a file with the size of 250GB from a source to a destination using 4, 5, 6 hops (routers). As we mentioned that each physical server has 4 virtual routers, therefore, when the traffic moves from router number 4 to router number 6, it goes via the physical links because every 4 virtual routers are in different physical server. Table 1 shows the results of the comparison of the delay between the TCP/IP and ROACM. This depicted in Figure 8. The x-axis represents the number of hops (virtual routers) and the y-axis represents the time delay in seconds. So, Figure 8 shows that at the hop number 6, the value of the delay time of TCP/IP is very high than the ROACM. Then we add The results of 8 hops, see table 2.

Table 1: The results of data transfer of file size 250GB for up to 6 hops.

No. Hops	TCP/IP	ROACM
4	1465	1449
5	1534	1498
6	1583	1407

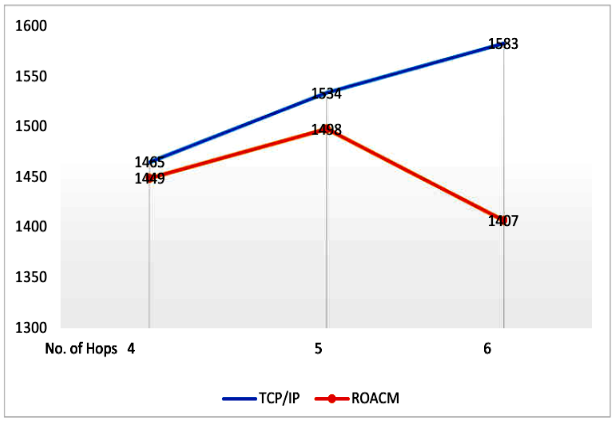
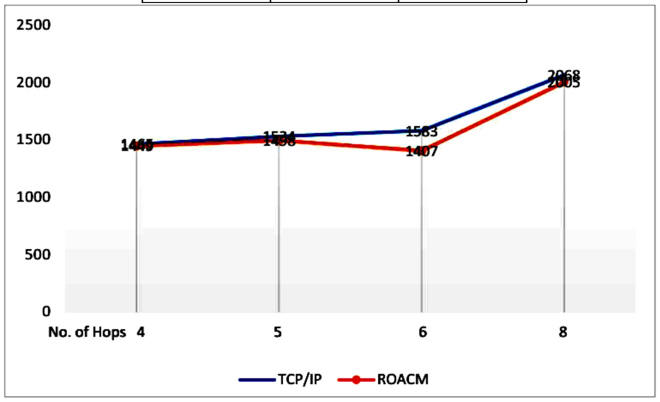


Figure 8: The delay for 6 hops

Table 2: The results of data transfer of file size 250GB for up to 8 hops.

No. Hops	TCP/IP	ROACM
4	1465	1449
5	1534	1498
6	1583	1407
8	2068	2005



Figur 9: The delay for 8 hops.

As we can see in Figure 9 that even thaogh the time delay for ROACM is lower than TCP/IP, but it was not as expected to be. This is because of the coding (programming) at the edge routers for the ROACM consumes time. If we move this code to a hardware as the Application Specific Integrated Circuit (ASIC), then the delay for the ROACM will be as in Figure 8, and it will keep low as there are more hops in the session/connection.

4 FUTURE RESEARCH

Since protocols on software takes time, therefore, the code of the implementation of the ROACM should be implemented in a hardware such the ASIC Hardware. This is our future research.

5 CONCLUSION

In this research, we focus in implementing the Route once and cross-connect many (ROACM) new protocol as a software based on Data Plane Development Kit (DPDK) application. The operating system for ROACM was built by using the open source network routing suite "Quagga" and the "DPDK". We use the C programming language for coding. In this research, we make the implementation of the ROACM starts from source edge router to destination edge router, not as in the previous study, where in the simulation, the ROACM protocol was considered from the source workstation to the destination workstation. In this research, we build the operating system for ROACM by using the open source network routing suite "Quagga" and the "DPDK". The results show that the ROACM outperforms the TCP/IP protocol when there are many hops (routers) between the source and the destination.

ACKNOWLEDGMENT

The author would like to thank "Badir", the technology incubator at King Abdulaziz City for Science and Technology (KACST) based in Riyadh, Saudi Arabia for funding the implementation stage of this research and facilitating their services to make this project successful. Also, the author would like to thank High Speed Networks Solutions LLC, Jeddah, Saudi Arabia, which is the development management company who oversaw the process of the protocol's operating system development in association with 3rd party developer.

REFERENCES

[1] Abdullah Bahattab, "Route once and cross-connct many", USA patent No. 7664108, Feb. 16, 2010.
[2] Abdullah Bahattab, "Route once and cross-connct many", European patent No. 2048835, March. 22, 2017.
[3] Abdullah Bahattab, "A Comparative Analysis of TCP/IP and ROACM Protocols- A Simulation Study", Indian Journal of Science and Technology, Volume 9. Issue 28, July, 2016.
[4] T. Ishihara, et al. "A Consideration of IX Architecture Using MPLS Based on Router Performance and QoS Requirements", IEICE Transaction Communication, Vol. E86-B, No.2, Feb.2003
[5] X. Xiao, L. Ni, V. Vuppala "An Overview of IP Switching and Tag Switching", ICPADS'97, Dec. 11-13, 1997, Seoul, KOREA.
[6] Y. Rekhther, et al. "Cisco Systems' Tag Switching Architecture Overview" RFC 2105, Feb. 1997.
[7] Y. Katsube, K. Nagami, and H. Esaki, "Toshiba's Router Architecture Extensions for ATM: Overview," IETF RFC 2098, April 1997.
[8] F. Baker, "Requirements for IP Version 4 Routers," IETF RFC 1812, Jun. 1995
[9] Cisco White Paper, "Understanding MPLS-TP and Its Benefits, © 2009 Cisco Systems, Inc. All rights reserved. This document is Cisco Public Information.
[10] Ali Diab, Rene Boringer, "Optimized I-MPLS: A Fast and Transparent Micro- Mobility-Enabled MPLS Framework", 1-4244-0398-7/06 §2006 IEEE
[11] Wenjun Xie, Shanguo Huang, Wanyi Gu, "AN IMPROVED RING PROTECTION METHOD IN MPLS-TP NETWORKS", Proceedings of ICNIDC2010, 978-1-4244-6853-9/10 ©2010 IEEE

[12] David Applegate, "Load optimal MPLS routing with N + M labels", 0-7803- 7753-2/03 (C) 2003 IEEE

[13] Yimin Qiu, "A Research of MPLS-based Network Fault Recovery", 2010 Third International Conference on Intelligent Networks and Intelligent Systems 978-0-7695-4249-2/10 © 2010 IEEE

[14] P. Newman, T. Lyon, and G. Minshall, "Flow Labelled IP: A Connectionless Approach to ATM," Proc IEEE Infocom'96, San Francisco, CA, March 1996, pp. 1251 - 1260.

[15] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow, "Cisco Systems' Tag Switching Architecture Overview," IETF RFC 2105, Feb. 1997.

[16] Peter Newman et al., "IP Switching and Gigabit Routers", IEEE Communications Magazine, Vol. 35, No. 1, January 1997, pp. 64-69.

[17] Kaur.G and Kumar.D, MPLS technology on IP backbone network, IJCA, 2010

[18] Dumka.A and Mandoria.H, Dynamic MPLS with feedback, IJCSEA, 2012

[19] Chan.C et al, High performance IP forwarding with efficient routing-table update, ELSEVIER, 2003