Ethernet Bonding on a Bare PC Web Server with Dual NICs

Faris Almansour
Department of Computer & Information Sciences
Towson University
Towson, MD 21252
falmansour@towson.edu

Ramesh Karne
Department of Computer & Information Sciences
Towson University
Towson, MD 21252x 6221
rkarne@towson.edu

Alexander Wijesinha
Department of Computer & Information Sciences
Towson University
Towson, MD 21252
awijesinha@towson.edu

Bharat Rawal
IST Department
Penn State Abington
Abington, PA 19001
bsr17@psu.edu

ABSTRACT

Bare PC[1] applications run without the support of an operating system (OS) or kernel and include the necessary hardware interfaces and network device drivers with each application. We describe a novel implementation of Ethernet bonding on a bare PC Web server using dual NICs, where both NICs can send but only one NIC can receive. The split send-receive design is easily extended to more than two NICs and other send/receive NIC configurations. Conventional Ethernet bonding requires some form of OS or kernel support. In the bare PC implementation, OS overhead and OS-related vulnerabilities are eliminated. We describe the dual NIC bare server architecture and implementation, and present experimental results to measure server performance. The results confirm that client connection and response times are better than for a bare server with two cards that each receive and send. This implementation of Ethernet bonding on a bare PC Web server is the first step towards building secure bare PC servers that can optimize performance using multi-core processors and multiple NICs.

CCS CONCEPTS

• **Networks** • **Networks~Network adapters** • *Networks~Link-layer protocols* • *General and reference~Performance*

KEYWORDS

Bare machine computing, bare PC, operating system, Ethernet bonding

SAC'18, April 9-13, 2018, Pau, France

DOI: xx.xxxx/xxx_x

ACM Reference format:

F. Almansour, R. Karne, A. Wijesinha, and B. Rawal. 2018. In *Proceedings of ACM SAC Conference, Pau, France, April 9-13, 2018 (SAC'18)*, 8 pages. DOI: xx.xxxx/xxx_x

1 INTRODUCTION

Minimalist platforms are characterized by low cost and system simplicity [1]. Bare machine computing (BMC) [2] is a novel minimalist approach that has been used to build applications such as Web servers [3], mail servers [4], high performance servers [5], VoIP systems [6], IPv4-IPv6 gateways [7] and SQLite databases [8] without using any OS or kernel. Bare applications run on ordinary desktop hardware without using a hard disk (i.e., on a bare PC). The main advantages of BMC applications are the elimination of both OS overhead and OS-related vulnerabilities. BMC applications are also easier to analyze for security vulnerabilities because of their simple design and small footprint.

Ethernet NIC bonding [9] combines network interfaces on a device. It is widely used in practice, for a variety of reasons including load balancing and reliability. Ethernet bonding on Web servers typically requires some form of OS or kernel support and has not been previously done on a bare PC. We implement a novel form of Ethernet bonding using dual NICs with a bare PC Web server application. In the bare PC Ethernet bonding implementation, both NICs can send but only one NIC can receive. The approach can be extended to work with more than two NICs and a variety of send/receive NIC configurations. In future, NIC bonding can be used with bare PC Web servers running on multi-core architectures and with other high performance or high security BMC applications.

To illustrate the dual NIC split send-receive approach in a BMC application, consider the intertwined HTTP/TCP protocols as implemented in a bare PC Web server with a single NIC [3]. Protocol intertwining is inherent in BMC applications due to their underlying task design discussed later. While a bare Web server can use TLS for security, in this paper (for reasons of simplicity),

we only discuss HTTP connections to port 80. Fig. 1 shows the intertwined protocol messages exchanged by client and server due to a single request from a client, which can be any conventional OS-based Web browser. A bare PC NIC driver consists of send and receive data structures as shown in Fig. 2. In the driver, the send path using the transcript descriptor list (TDL) and the receive path using the receive descriptor list (RDL) are separate and parallel paths in the hardware. Send and receive controls such as enable and disable, and associated configuration parameters are also different in the driver. In essence in the NIC and also in the driver, send and receive paths can be treated as two separate entities.

In the bare Web server single NIC implementation shown in Fig. 3, send and receive paths are also different. When a packet is received in RDL, its Ethernet header is removed and IP processing is done on the packet as usual. After the IP header is removed, TCP and HTTP processing are done in an intertwined manner on the packet. For a given client's IP address and port, a unique entry for the request is created in the TCP table. This unique entry is kept in the table until the completion of the client's request.

A client HTTP Get or Post requires a server response, which consists of the response data prefixed with an HTTP header. The response is inserted into one or more packets with TCP, IP and Ethernet headers. These packets are inserted into the TDL. The Ethernet NIC sends packets from the TDL and receives packets from the RDL. All of these steps are part of the Web server application since there is no OS or kernel in a BMC system. The monolithic executable, unique tasking (discussed later), and integration of all protocol code with the application facilitates the separation of send and receive paths in the design of the bare PC Web server. In effect, send and receive paths are naturally disjoint in a bare Web server at the task level, NIC level, driver level, protocol level and application level. This enables Ethernet bonding to be implemented without any OS or kernel support.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the dual NIC architecture, design and implementation. Section 4 gives experimental results and Section 5 concludes the paper.

2 RELATED WORK

The growing popularity of virtualized OSs and containers have resulted in the recent introduction of several so-called minimalist OSs [10], including the Docker platform, Ubuntu Core, Core OS, and Atomic Host. Minimalist OSs may focus on protecting against failures and attacks, target embedded systems, IoT and the cloud, or be designed for speed. There are also many lightweight Linux distributions that run on older x86 hardware. One of the earliest attempts to provide a small kernel with minimal functionality for applications is Exokernel [11]. Subsequently, lightweight OSs supporting high-performance applications were introduced as in [12]. In contrast, BMC systems are non-virtual and non-embedded. They enable applications to run without any OS or kernel support by providing direct interfaces to the hardware [13].

Ethernet bonding on Linux systems [9] has numerous driver options and allows bonding configurations to be based on requirements of high availability or maximum throughput for single and multiple switches. The performance of Ethernet bonding is studied in [14], and it is found that active backup mode has low switch-over time in case of failure, while round robin mode with dual NICs almost doubles the throughput achieved without bonding. In Oracle VM, network bonding is designed primarily for redundancy although increased throughput requirements are also supported [15].

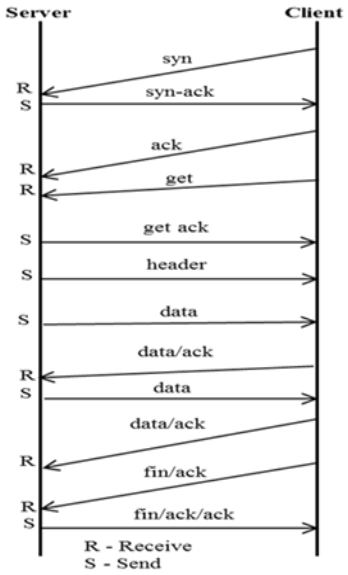

Figure 1: Integrated HTTP/TCP protocol.

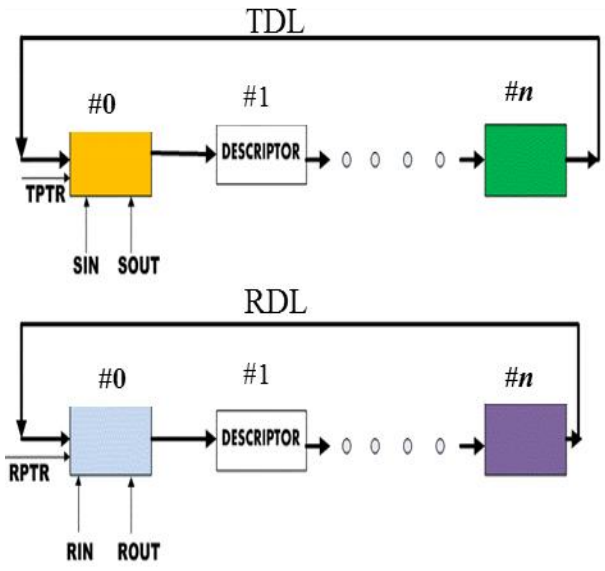

Figure 2: Transmit and receive descriptors.

3 DUAL NIC BARE PC WEB SERVER

3.1 Architecture

As noted earlier, it is natural to split send and receive logic in a BMC application from the NIC hardware level to the application level. We modify the single NIC bare PC Web server enabling it to use dual NIC Ethernet bonding by splitting the send and receive paths. Fig. 4 shows the bare PC Web server architecture for Ethernet bonding with dual NICs. Many clients can connect to the Web server where each client request is identified as usual by the unique IP address and port number combination. The two NICs are associated with respective IP addresses IP1 and IP2, and MAC addresses MAC1 and MAC2. The server uses only one MAC address to receive packets, which is the server's MAC (MAC1) for the receive NIC (R-NIC). All packets from the clients are thus received at the server on MAC1. ARP broadcasts are used to ensure that the default gateway or any local clients only send packets to the server using MAC1. No switch configuration is needed. The server can use either NIC for sending data based on load balancing requirements.

When a BMC system is booted, it goes to the MAIN task, which runs continually whenever no other task is running. This is different from the way an OS/kernel-based system works: in a bare PC, tasks are created and used as needed by a given BMC application suite. When a packet arrives, the RCV task is run to read a packet from the RDL and it continues running while Ethernet, IP and TCP processing is done on the packet until the state is updated in the TCP table. When an HTTP Get or Post command is received, a unique HTTP task is created to process the client request. A connection may be alive for many GET or POST requests. However, a unique HTTP task handles a given request.

The relation between the MAIN, RCV and HTTP tasks used in the bare Web server is shown in Fig. 5. The RCV and HTTP tasks process client packets that are received or that need to be sent. In an OS-less BMC system, tasks run to completion and only suspend themselves when waiting for an event. Suspended tasks are resumed when they are ready to run. Notice that the bare server has an inherently parallel design with respect to client requests.

3.2 Design

In a dual NIC system, there are many ways to send and receive packets using the NICs. In Fig. 1, the server receives SYN, SYN-ACK-ACK, GET, DATA-ACK and FIN-ACK from the client; and sends SYN-ACK, GET-ACK, Data and FIN-ACK-ACK to the client. The HTTP and TCP protocols can be split based on send and receive interfaces. Also, for HTTP requests, packets received by the server except for GET are small (about 60 bytes), while data packets sent to the client are large. Thus, when dual NICs are used for send and receive separately, the load is not balanced with respect to the two cards.

One approach with dual NICs is to dedicate one NIC for sending and one NIC for receiving. Sent packets use IP2 and MAC2 but the send NIC does not receive any data from a client. Similarly, a receive NIC does not send any data. This simplex connection of dual NICs may have potential security benefits due to isolation of send and receive paths. We do not investigate such benefits in this paper.

Another alternative is to receive on one card and send on two cards to load balance the data. When two cards are used for sending, small packets can be sent on one card and large data packets on the other. The decision of which card to use for sending a given packet can be made based on load balancing requirements.

We investigated a variety of strategies for load balancing NICs based on the novel bare PC Web server design. As shown in previous studies using OS-based systems, simple load balancing based on the data alone is not sufficient to achieve optimal performance in a dual NIC architecture. It is also known that using two different cards for the same request does not improve any performance, as the client has to handle two NICs simultaneously (data level parallelism). In order to achieve optimal dual NIC performance using the bare PC Web server design, we exploited the parallelism in the client requests, where each request can be treated as being independent of the others (request level parallelism). Although not investigated in this paper, the single core dual NIC design can be extended to multi-core level parallelism using essentially the same approach.

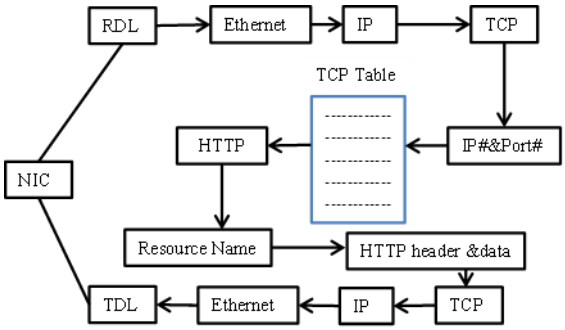

Figure 3: Single NIC bare PC Web server flow.

3.3 Implementation

Implementation of Ethernet bonding using dual NICs in a bare PC Web server for optimal performance requires that the NIC for sending data to clients be dynamically adapted based on the load. To simplify the design, we used two identical Ethernet NICs. One NIC is enabled for receive and send, and the other NIC is only enabled for sending data. This strategy is simple to implement in the bare PC Web server as the NIC driver code is part of an application program.

In the bare PC Web server design, two instances of the Ethernet object are created to interface with the two NIC drivers. When a SYN packet arrives from a client, the load balancing strategy is implemented based on request level parallelism. In addition, for a given client, a dedicated NIC will send the data and all other packets on the connection. For example, if there are two

clients sending requests, client 1 will get data from NIC 1 and client 2 from NIC 2. If a third client comes at the same time, half of its responses are sent using NIC 1 and the other half using NIC 2. If there are n simultaneous requests then n/2 use NIC 1 and remainder use NIC 2. This strategy is easily implemented in the bare PC application itself (without any OS or kernel involvement).

When a TCP SYN packet arrives, a send-id (1 or 2) is set in the TCP table to handle sending data using NIC 1 or NIC 2 respectively. A new function is written to send data based on the send-id. Minor modifications in the code enabled us to extend the existing single NIC Web server to a dual NIC architecture with load balancing implemented in the application itself. As the client requests are parallel, the two NICs and the two HTTP tasks run in parallel to provide a parallel path from the NIC hardware to the application. With a single core, this optimizes the use of dual NICs.

The Web server and the NIC driver are implemented using C/C++. Intel gigabit NICs are used to implement the Web server system. About 20 lines of assembly code are used in the NIC driver, which required two functions to read and write to the host controller configuration registers.

To further understand the details, recall that bare PC applications include the necessary network protocols and directly use the Ethernet interface without going through intermediate layers. This optimization avoids the need for buffer copying and also reduces procedure call and other overhead. An API is provided for bare PC applications to access the Ethernet interface as there in no centralized OS or kernel in the system.

The code segment in Fig. 6 illustrates how the TCP level code in a bare PC Web server application using an Intel gigabit NIC interacts with the Ethernet interface. This approach is not the same as offloading TCP functions to a NIC in an OS-based system. In the BMC system, calls are made directly by the TCP code (which is intertwined with code for HTTP and the bare PC Web server application) to methods that format the TCP, IP, and Ethernet headers before sending the packet. These calls are made within a single thread of execution avoiding the need for switching between layers. The Ethernet buffers are accessed directly and the send buffer address is formed in step 1. Formatting headers and aligning the data is done in steps 2 through 7. In the above code, EO is EO1 for NIC 1 and EO2 for NIC 2 based on send-id 1 or 2 respectively. These seven steps enable the bare PC Web server application to send a packet using an Ethernet NIC without any OS or kernel overhead.

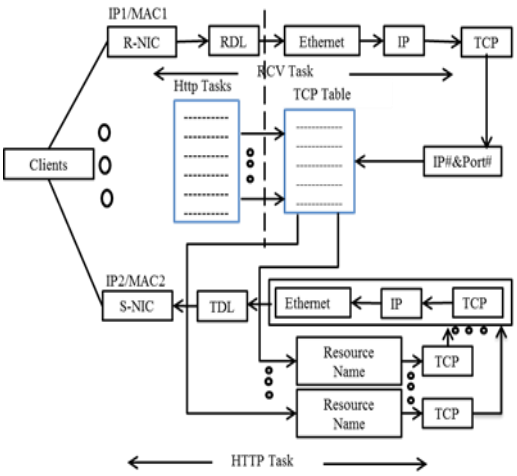

Figure 4: Dual NIC Web server architecture.

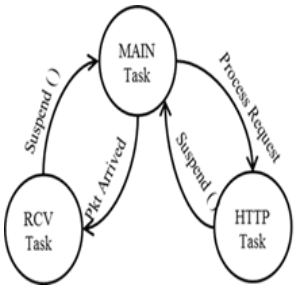

Figure 5: Task state transition diagram.

4 PERFORMANCE MEASUREMENTS

This section gives performance measurements for single and dual Intel external gigabit NICs [16] installed on a bare PC Web server running on an Optiplex 960 desktop machine. The HTTP stress tool http_load [17] is used to send requests and collect measurement data. The experiments are used to study bare PC Web server performance differences between single and dual NICs for different values of the load parameters. The requested file sizes vary from 4K to 1MB. Connection time and initial response times are used as performance metrics.

```
//1. Get Transmit Buffer Pointer
x = EO.getTDLPointer() + EO.getSendInPtr() * 16;
p1 = (long*)x;
send_buffer = (char*)*p1;

//2. Add Ethernet and IP Header Lengths
send_buffer = send_buffer + 14 + 20;

//3. Format TCP Packet
TCPPack_size = FormatTCPPacket(send_buffer, tcb->IP,
      tcb->PORT, tcb->tempflags,tcb->RCVWND,
      tcb->tempSeqNum,tcb->RCVNXT, sendbuffer, TCPSegSize,
     tcb->temprIndex, currenttask);

//4. Adjust for IP Header
send_buffer = send_buffer - 20;
//5. Format IP Packet
retcode = ip.FormatIPPacket(send_buffer, TCPPack_size, tcb->IP,
       tcb->destmac, TCP_Protocol, currenttask);

//6. Adjust for Ethernet Header
send_buffer = send_buffer - 14;
//7. Format Ethernet Header and Send Data
retcode = EO.FomatEthPacketN(send_buffer, TCPPack_size+20,
    IP_TYPE , tcb->destmac,InPtr, tcb->count1, tcb->sendtype,
    currenttask);
```

Figure 6: Application directly accessing Ethernet buffers.

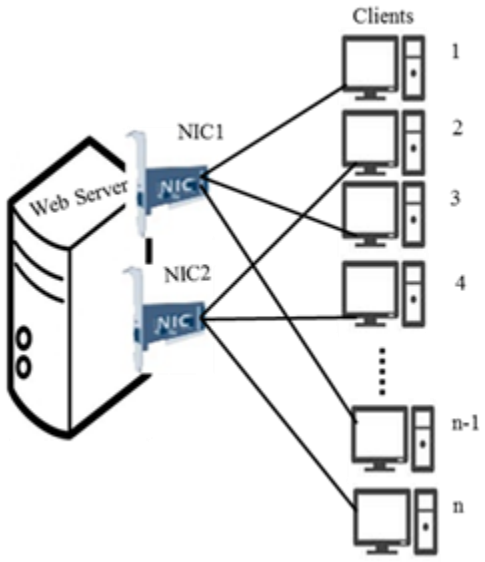

Figure 7: Each client has a dedicated send NIC.

4.1 Load Balancing Strategy

Two NICs can be used in a variety of ways to send and receive packets. We consider load balancing for dual NICs with split send and receive paths, where one NIC is dedicated for receiving packets and either NIC is able to send packets to a client.

Several strategies can be easily implemented to choose the NIC for sending packets. For example, packets can be sent by randomly picking the NIC to use for a given client, or alternatively, the NIC to send can be chosen based on throughput, packet size, connection time and/or response time. We consider a simple strategy, where a given NIC is dedicated for sending data to a given client as shown in Fig. 7. For an even number of clients, both NICs send data by sharing the number of requests equally. For an odd number of clients, the server uses NIC 1 for half of the responses to send data and NIC 2 for the other half.

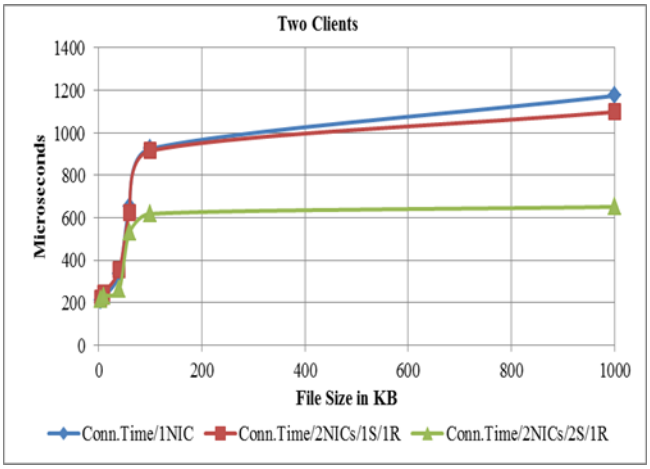

Figure 8: Connection time (two clients).

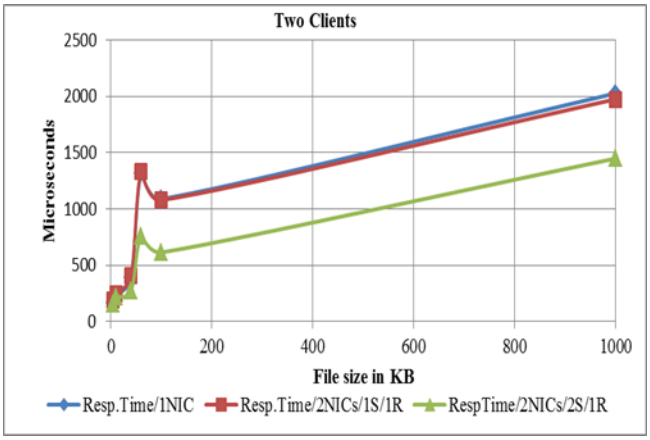

Figure 9: Response time (two clients).

4.2 Two Clients

Fig. 8 shows connection times for a single NIC (1 NIC) and dual NICs: (1 send, 1 receive) or (2 send, 1 receive). Connection time with one NIC increases as the file size increases, which is expected as it takes more time to send larger files. When dual NICs are used, one for sending only and one for receiving only, the performance is very close to one NIC. This is because the receive card is not as heavily loaded as the send card. Also, received packets associated with GET requests are small (about

60 bytes each except for the GET request itself). Since the two NICs are not sharing the load equally, there is no significant performance improvement.

When two NICs are used, where both can send data, the lowest connection time is seen because the NICs are now parallelized with respect to clients (each NIC serves one individual client). For 4K and 1 MB file sizes, we used respective rates of 1000 and 30 requests per second for each client. For large files, the request rate is limited by buffers allocated at the server. The maximum improvement in connection time is about 44% for a 1 MB file. We do not see any connection time improvement for small file sizes as the NICs are not load balanced. As shown in Fig. 9, the improvement in initial response time for a 1 MB file is 29%.

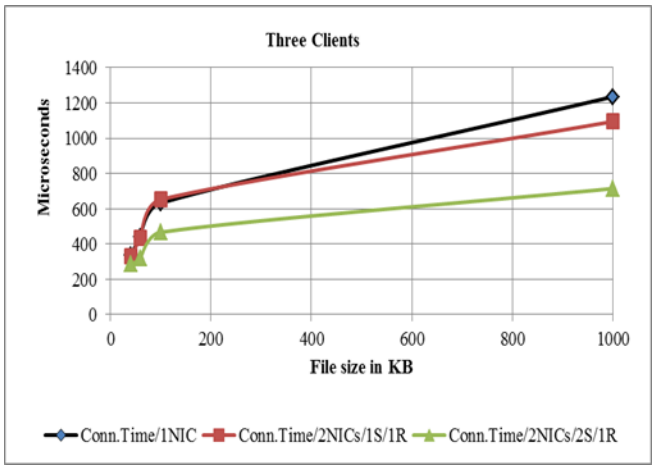

Figure 10: Connection time (three clients).

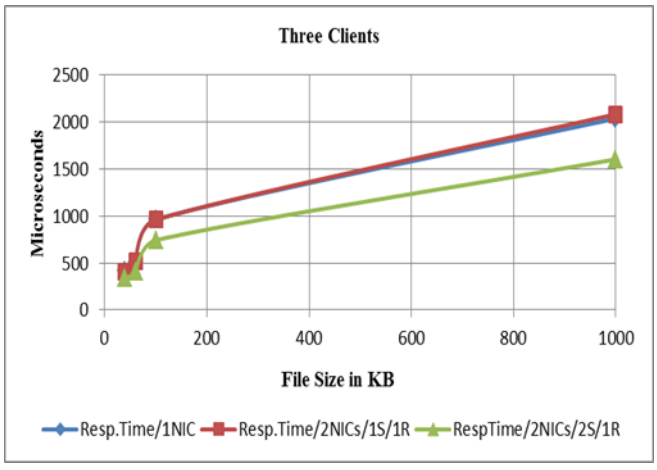

Figure 11: Response time (three clients).

4.3 Three Clients

Fig. 10 shows connection times for three clients. Client 1 is served by NIC 1 and Client 2 is served by NIC 2. Half of Client 3's requests are served by NIC 1 and the other half are served by NIC 2. For a 1 MB file, the connection time improvement is 42%, and the initial response time improvement (shown in Fig. 11) is 21%.

5 CONCLUSION

We described the architecture, design and implementation of a dual NIC Web server with split send-receive paths that runs on a bare PC. Performance measurements were given to illustrate the improvement in connection and initial response times as the load is increased. Associating a given client's request with a given NIC for sending data was found in our studies to be optimal for load balancing with dual NICs in a bare PC Web server. Other strategies for load balancing multiple NICs can be investigated in the future.

The dual NIC approach is shown to exploit the natural partition on send and receive logic at the NIC level, Ethernet driver level and the Web server implementation level. Isolating send and receive paths provides simplicity and better performance. Although not investigated here, multiple NICs can be used to completely separate the send and receive channels for security purposes. This approach for Ethernet bonding on servers without any OS or kernel support extends to multicore systems. With a single NIC for communication (as in an ordinary desktop), the multi-core processors share memory and the NIC. When dual or multiple NICs are used, they can be allocated to cores thus improving performance. It is also possible to integrate NICs with multicores on the same chip.

REFERENCES

[1] S. Soumya, R. Guerin and K. Hosanagar, "Functionality-rich vs. Minimalist Platforms: A Two-sided Market Analysis", ACM Computer Communication Review, vol. 41, no. 5, Sept. 2011, pp. 36-43.

[2] R. K. Karne, K. V. Jaganathan, N. Rosa, and T. Ahmed, "DOSC: dispersed operating system computing", 20th Annual ACM Conference on Object Oriented Programming, Systems, Languages, and Applications (OOPSLA), 2005, pp. 55-61.

[3] L. He, R. K. Karne, and A. L. Wijesinha, "The design and performance of a bare PC Web server", International Journal of Computers and Their Applications, IJCA, Vol. 15, No. 2, June 2008, pp. 100-112.

[4] G. H. Ford, R. K. Karne, A. L. Wijesinha, and P. Appiah-Kubi, "The design and implementation of a bare PC email server", 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC), 2009, pp. 480-485.

[5] B. Rawal, R. K. Karne, and A. L. Wijesinha. "Mini Web server clusters for HTTP request splitting," IEEE Conference on High Performance, Computing and Communications (HPCC), 2011, pp. 94-100.

[6] R. Yasinovskyy, A. Alexander, A. L. Wijesinha, and R. K. Karne, "Bare PC SIP user agent implementation and performance for secure VoIP", International Journal on Advances in Telecommunications, vol 5 no 3 & 4, 2012, pp. 111-119.

[7] A. Tsetse, A. Wijesinha, R. Karne, A. Loukili and P. Appiah-Kubi, "An experimental evaluation of IP4-IPv6 IVI translation", ACM SIGAPP Applied Computing Review, March 2013, vol. 13, no. 1, pp. 19-27.

[8] W. Thompson, R. Karne, A. Wijesinha, and H. Chang, "Interoperable SQLite for a bare PC", Beyond Databases, Architectures and Sturctures Conference (BDAS), 2017, pp. 177-188.

[9] Linux Ethernet bonding driver HOWTO, https://www.kernel.org/doc/Documentation/networking/bonding.txt, accessed: Sep 2017.

[10] P. Salvatore, "The new minimalist operating systems", https://blog.docker.com/2015/02/the-new-minimalist-operating-systems/, accessed: Sep 2017.

[11] D. R. Engler and M.F. Kaashoek, "Exterminate all operating system abstractions", Fifth Workshop on Hot Topics in Operating Systems, USENIX, 1995, p. 78.

[12] J. Lange, et. al, "Palacios and Kitten: new high performance operating systems for scalable virtualized and native supercomputing," 24th IEEE International

Parallel and Distributed Processing Symposium, 2010.
[13] R.K. Karne, K. V. Jaganathan, and T. Ahmed, "How to run C++ applications on a bare PC," 6th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD), 2005, pp. 50-55.
[14] H. Tran-Viet, et. al, "Experimental study on the performance of Linux Ethernet bonding", Inernational Conference on Testbeds and Resaerch Infrastructures: Development of Networks and Communities (TridentCom), 2014, pp. 307-317.
[15] Network Bonding, https://docs.oracle.com/cd/E27300_01/E27309/html/vmusg-network-bonding.html, accessed: Sep 2017.
[16] Intel; PCI/PCI-X Family of Gigabit Ethernet Controllers Software Developer's Manual.
[17] http_load-multiprocessing http test client, http://www.acme.com/software/http_load/, accessed: Sep 2017.