2009 International Conference on Computer Engineering and Technology

Performance Analysis of Large Receive Offload in a Xen Virtualized System

Hitoshi Oi and Fumio Nakajima
The University of Aizu, Aizu Wakamatsu, JAPAN
{oi,f.nkjm}@oslab.biz

Abstract

System-level virtualization provides us various advantages including independent and isolated computing environments on which multiple operating systems can be executed, and improved resource utilization. These benefits come with performance overheads and network operation is one of most typical cases. In this paper, we first present the experimental results of network performance with LRO under varying message sizes and maximum transmission unit (MTU). We analyze the effectiveness of large receive offload (LRO) in a consolidated execution environment based on Xen. When SPECjbb, a CPU-intensive workload, and network receiver are executed on separated domains, LRO in physical and virtual NICs improve the throughput up to 8 and 14%, respectively.

Keywords Virtualization, Network, Performance Analysis

I. Introduction

System-level virtualization technologies provide multiple independent computing environments on a single physical platform. Each computing environment (virtual machine, or domain in Xen) can run a (possibly) different operating system (guest OS) and resource isolation between virtual machines (VMs) are guaranteed [1]. The advantages of virtualization come with performance overheads. With hardware supports for CPUs, switching between privileged and unprivileged execution modes is relatively fast [2], [3]. However, accesses to I/O devices are still costly operations and networking is one of most typical cases. In Xen, paravirtualization (PV) model is used for virtualizing I/O devices. With the PV device model, multiplexing of a single I/O device among guest domains is possible. In addition, the interface of the I/O device for each guest domain can be abstracted. However, the performance degradation of network interface in the PV model is significant and various optimization attempts have been made [4], [5].

Large receive offload (LRO) is a technique to reduce the overhead of handling received message packet [6], [7]. Previously, we ported the LRO into a Xen virtualized system and reported the results of preliminary performance measurements [8]. In this paper, we will analyze the performance of LRO in a Xen virtualized system in more detail and investigate the interference with the workload running on another domain on the same platform.

This paper is organized as follows. In the next section, we explain the technical background of this work. In Section III, experimental results and their analysis are presented. We show how network throughput is changed under varying application message size, maximum transmission unit and LRO scheme. We also show the effect of CPU utilization by another guest domain. Previous work related to this paper is presented in Section IV and we conclude the paper in Section V.

II. Background of This Work

In this section, we present the background of this work. First, we provide a brief description of the Xen internal network architecture. Next, Large Receive Offload (LRO), an implementation technique to reduce the network operation overhead, is presented. The objectives of this paper derived from our previous work are explained in Section II-C.

A. Xen Internal Network Architecture

Figure 1 shows the architecture of the Xen internal network. Each guest domain has zero or more virtualized network interface called netfront. In the privileged domain handling physical network interface (driver domain), there is a counterpart for each netfront (netback). Netfront and corresponding netback exchange packets by sharing page frames pointed to by the descriptor ring. For event notification, virtual interrupt mechanism (event channel) through Xen hypervisor is used. This paravirtualized network architecture of Xen has advantages, such as device

978-0-7695-3521-0/09 $25.00 © 2009 IEEE
DOI 10.1109/ICCET.2009.112

IEEE computer society

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:15:14 UTC from IEEE Xplore. Restrictions apply.

isolation or transparency to the guest domain, but it incurs high network performance overhead. In [5], it is reported that the per packet CPU overhead (in clock cycles) in Xen network is more than four times larger than that of native Linux.

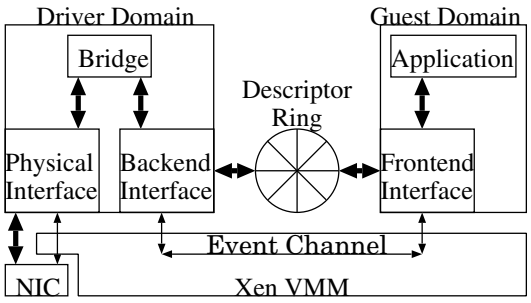

Fig. 1. Xen Internal Network Architecture

B. Large Receive Offload

Large receive offload (LRO) was initially implemented by hardware on the Neterion 10GbE Ethernet card [6]. Jan-Bernd Themann implemented it in software as a patch for the Linux kernel [7] and it has been adopted to the kernel tree from version 2.6.24. LRO combines the received TPC packets and passes them as a single larger packet to the upper layer in the network. By reducing the number of packet processing operations, the CPU overhead is lowered and the performance improvement is expected.

C. Objectives of This Paper

In our previous work, we ported LRO into the physical and virtual NICs of a Xen virtualized system and measured its effectiveness [8]. As shown in Table I, LRO in the physical NIC reduced the numbers of clock cycles and instructions for the same amount of message transfer while LRO in the virtual NIC increased the throughput.

Following these results, our objectives in this paper are as follows: (1) some NICs support larger maximum transmission units (MTUs) than standard 1500B. Larger MTUs can reduce CPU utilization and lead to more efficient message transfer. We show how a larger MTU affects the effectiveness of LRO in Xen virtualized system. (2) In the previous results shown in Table I, the LRO in the physical NIC reduced the CPU overhead of message receive operation. We show that how this reduction of CPU overhead affects the performance of the workload in other domain sharing the hardware platform. (3) Similarly, we investigate how the network performance is altered when the workload in another domain is varied and whether LRO can reduce the performance interference or not.

Config.	TP (Mb/s)	CLK	INST	LRO
Linux	557.0	6.27	6.30	N/A
Xen	505.4	9.71	7.16	N/A
pLRO	508.4	8.01	5.36	1.99
vLRO	615.3	9.87	7.49	4.92

TABLE I. Effectiveness of LRO [8]. For the abbreviations in Config. column, see Table III. TP stands for throughput in Mbps. CLK and INST stand for the numbers of clock cycles and instructions in 10^{10} for a 10GB message transfer. LRO is the number of packets combined by LRO.

III. Experiments and Results

In this section, we present the results and analysis of our experiments. First, the hardware and software environment used for the experiments are described. Next, we present the measurement results of throughput for varying MTU and LRO scheme. Finally, the performance interference between the packet processing overhead and the workload in other domain is investigated.

A. Experimental Environment

Component	Description
CPU	Xeon 3GHz
Memory	2GB
NIC	1Gbps
Operating System	Linux 2.6.18
VMM	Xen 3.1.1
Network Measurement	Netperf 2.4.4
Java workload	SPECjbb2001 v1.04
Guest Domains	
vCPU	1/domain
Memory	512MB/domain

TABLE II. Benchmarking Environments

Table II shows the details of the machine used as the message receiver. The hardware platform used has a Xeon processor, 2GB of memory and a Gigabiet NIC. This machine is connected to another machine used as a message sender by a cross-cable. The base operating system is Linux kernel 2.6.18 and Xen 3.1.1 is used for virtualization. This receiver machine is the same as the one used in our previous work [8]. The machine used as the message sender is also a Linux 2.6.18 based machine, but the main difference is the NICs. Previously, we used on-board NICs for both the sender and the receiver. In this paper, we use NICs that support larger MTUs which are connected through the PCI-Express bus. Also, we previously used a network switch but in this paper a

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:15:14 UTC from IEEE Xplore. Restrictions apply.

cross-cable connects two machines. For measurements of message throughput, we use the TCP STREAM TEST of netperf [9]. To control the CPU utilization of the receiver machine, we use SPECjbb2005 [10].

Abbrv.	Configuration
Linux	Native (Non-virtualized) Linux
Xen	Xen-virtualized (no LRO)
pLRO	LRO in Physical NIC
vLRO	LRO in Virtual NIC

TABLE III. Configurations and Their Abbreviations

B. Standalone Performance

Standard maximum transmission unit (MTU) on the Ethernet is 1500B. Some NICs support larger MTUs which improve the transmission efficiency. However, the larger the MTU, the fewer the packets to be sent which lead to the fewer chances of LRO. Also, the application level message sizes (e. g. the size of the message chunk with which `send()` is invoked) affect the CPU overhead for packet processing. In this section, we show the throughput and the LRO rate (the number of receive packets divided by the number of packets transferred to the upper network layer) for the varying MTU, message size and LRO schemes (Table III). In the virtualized configurations (Xen, pLRO and vLRO), only the guest domain for the message receiver is activated as the guest domain.

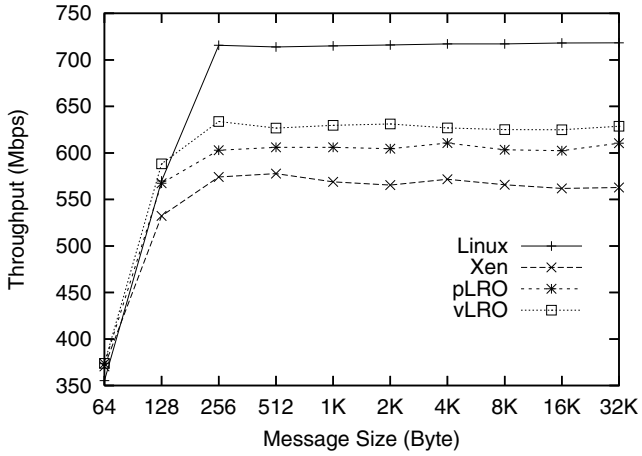

Fig. 2. Throughput (MTU = 1500B)

Figure 2 shows the throughput of four configurations for $MTU = 1500$. In the case of $MTU = 1500B$, throughput is quite low for the message sizes of 64 and 128B. However, for the message sizes of 256B or larger, the throughput is almost constant for all configurations. Throughput for all configurations are higher than those in Table I. The primary factor of these improved throughput should be the cross-cable connecting sender and receiver machines (avoid the network switch used in the previous work). The amount of improvement varies from 3% (vLRO) to 29% (Linux). The vLRO is least affected by (possibly) the use of the cross-cable. This is understandable because the LRO operations take place at the virtual NIC which is distant from the physical NIC.

We also note other differences between the results in this paper and those in the previous one. Previously, the improvement of throughput by pLRO was less than 1% but this time it is up to 8%. We may attribute this result to the LRO rate of pLRO in Figure 5. Previously the LRO rate in pLRO was 1.99 but now it is up to 2.70. Again, these higher LRO rates should be due to the cross-cable which transmits the packets at a higher rate than the network switch. The difference between pLRO and vLRO is shrunk compared to the previous one. vLRO achieved 22% and 21% higher throughputs than Xen and pLRO, respectively, in the environment for the previous paper. This time, throughput of vLRO is only 12% and 5% better than Xen and pLRO, respectively.

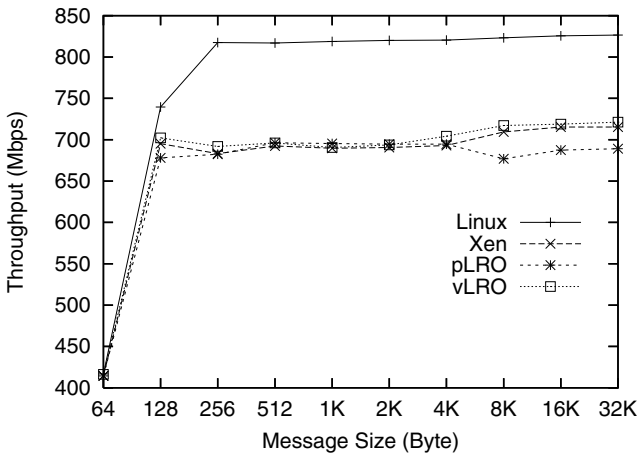

Fig. 3. Throughput (MTU = 4000B)

The throughput for MTU = 4000B are shown in Figure 3. We see great improvements of up to 15% in throughput over the MTU = 1500B cases in the native Linux. The larger MTU of 4000B also improves the network performance of Xen-based configurations: the throughputs of Xen, pLRO and vLRO are increased by up to 27%, 15% and 15%, respectively[1]. Unlike MTU = 1500B cases, LRO does not seem to improve the throughput of Xen-

[1]The cases for 128B or smaller messages sizes are excluded

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:15:14 UTC from IEEE Xplore. Restrictions apply.

based systems for MTU = 4000B. vLRO and Xen perform almost the same and pLRO is actually worse than Xen for the message sizes of 8KB or larger. The small benefit of LRO should be attributed by the small LRO rates as seen in Fig. 5: for the pLRO and vLRO cases, their LRO rates are 1.2 or smaller and 1.5 or smaller, respectively. As mentioned above, another thing we should note is that the throughput of pLRO becomes lower than Xen and vLRO for the message sizes of 8KB or larger. We do not have a clear answer to this question and further investigation is required. However, the Xen internal network passes network packets by exchanging page frames whose size is 4KB and the above phenomena may be related to this page frame size.

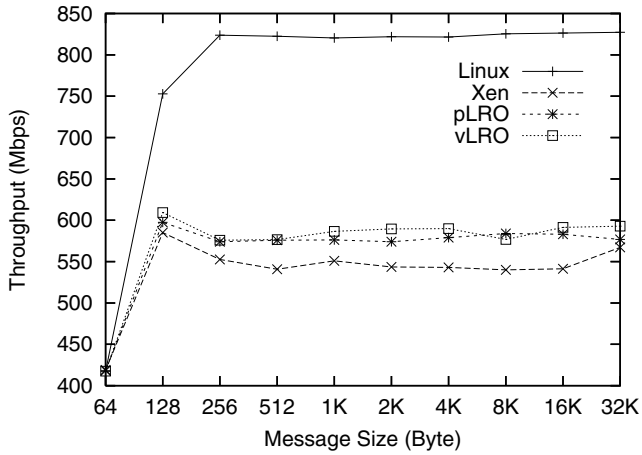

Fig. 4. Throughput (MTU = 4500B)

The throughput for MTU = 4500B are shown in Figure 4. This is the largest MTU we could set in our system. Similar to the MTU = 4000B cases, the native Linux takes advantage of MTU = 4500B and 15% of improvement in throughput is achieved. Unfortunately, Xen-based systems we used cannot take advantage of this yet larger MTU. In all three cases (Xen, pLRO and vLRO), the throughput are even lower than MTU = 1500B cases. Please note that, for the results shown in Figs. 3 and 4, MTUs of all Xen internal network components (bridge, netback and netfront) are set to 4000B and 4500B, respectively. We consider this poor network performance for MTU = 4500B as follows. As briefly mentioned above, the backend transfers the packet by exchanges page frames with the frontend of the target domain. Since MTU is slightly larger than the size of page frame, we have to use two page frames and only small fraction of the second page frame is filled with payload. These results suggest that even when the physical NIC supports the jumbo frame feature, we have to set actual MTU by taking the page frame size into consideration.

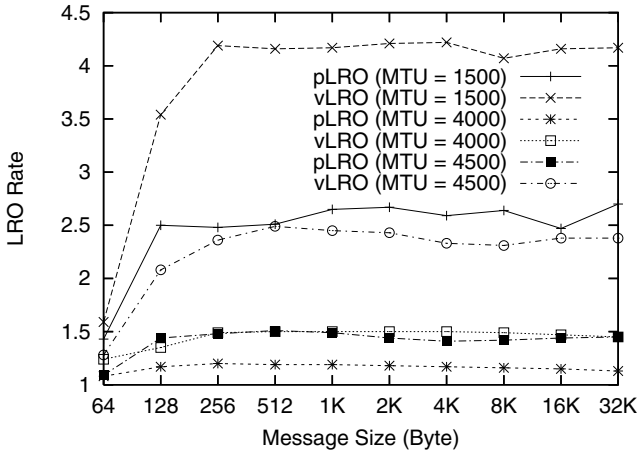

Fig. 5. LRO Rates

LRO rates for pLRO and vLRO with MTUs of 1500B to 4500B are shown in Figure 5. As mentioned above, for MTU = 1500B cases, pLRO rate is higher and vLRO rate is lower than those reported in our previous work, respectively. In all MTUs, vLRO achieves higher LRO rates than pLRO. MTU = 4500B cases achieve higher LRO rates than MTU = 4000B cases. Especially, vLRO in MTU = 4500B achieves LRO rates of around 2.5. One possible explanation is as follows. Again, due to the page frame exchanging mechanism of the Xen internal network, one packet of 4500B is divided into two part and then merged at the virtual NIC. The LRO rates of vLRO for MTU = 4000B (1.5 or smaller) support this idea because a packet of 4000B can fit in a single page frame. Again, further instrumentation and analysis are needed to validate this idea. In all other large MTUs, LRO rates are quite small and coincide with the little effectiveness of LRO in these configurations.

C. Performance Interference

In the previous subsection, only the guest domain for the message receiver was activated. In this subsection, we also activate another guest domain for running the SPECjbb benchmark to see the performance interference between two guest domains. SPECjbb is modeled after TPC-C benchmark [11] which is an OLTP workload, but the main difference is that all the database tables are stored in main memory (rather than HDDs in TPC-C). To control the CPU usage of the SPECjbb domain, we inserted short sleep periods in the transaction clients, which is the same methodology used in benchmark suites for consolidated systems [12], [13]. For the measurements in this subsection, we use the message size of 1KB and

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:15:14 UTC from IEEE Xplore. Restrictions apply.

MTU = 1500B. We set the scaling factor of SPECjbb $W = 4$, which is scaled down from the configuration of the SPECjbb in the VMmark (W = 8 for vCPU = 2) [12].

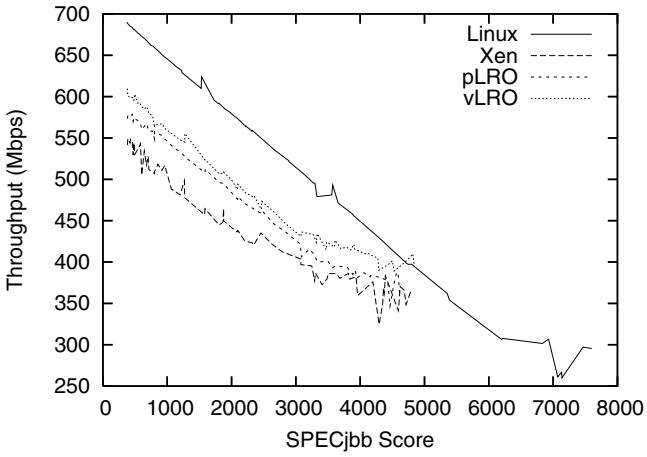

Fig. 6. Consolidated Performance

The relationship between the SPECjbb score and network throughput in four configurations are shown in Figure 6. The right-most point for each line represents the case where SPECjbb is executed without CPU usage control. Due to the discrete even nature of SPECjbb and unpredictability in system softwares (both Xen and Linux), there are several "noises" on each graph. However, we can find the followings from Figure 6: First, as expected, native Linux achieves higher throughput when SPECjbb consumes low CPU time, and vice verse. However, we can also say that in the multiprogramming environment of the native Linux, one CPU-intensive workload can use too much CPU time which may result in performance degradation of other workloads (in this case the lower network throughput). In Xen-based cases, resource isolation works as expected and such extreme case seen in the native Linux case is avoided.

Second, the throughput of all cases degrease linearly against the increasing SPECjbb scores. However, the rates of throughput decreases change at around SPECjbb score 3000 for the Xen-based cases and 6000 for the native Linux. These SPECjbb scores correspond to about 30% (Xen-based cases) and 60% (Linux) of the physical CPU time utilization. In the cases of Xen-based configurations, two guest domains are running, and it looks to imply that the network throughput decreases linearly until another domain uses around 60% of its CPU time share. Further investigation is necessary to confirm this hypothesis. Finally, the throughput improvement rate the pLRO ranges from 4% to 8% while that by vLRO rangres from 7% to 14%.

IV. Related Work

Meron et. al. evaluated three optimization techniques for the network performance in Xen [4]: the virtual network interface, the data path between guest and driver domains, and the virtual memory. They redefined the virtual network interface so that it could utilize the hardwares support from modern NICs such as TCP segmentation offloading (TSO). In [1], the design of Xen is described in detail. In this paper, they compared the network performance of Xen with native Linux, VMware and User-mode Linux. In addition to the standard MTU of 1500B, they used a smaller MTU of 500B to present the network connection in dial-up PPP client which is no longer used these days. Santos et. al. analyzied the Xen receiving network path by breaking down the CPU time into functional groups, such as network device driver, grant copy, etc. [5]. They applied various optimization techniques to these categories and reduced the overhead. Some of the techniques they used (such as reduction of packet copy operation) should be applicable to our cases and we plan to do so. When multiple domains on a single platform are configured to form a multi-tier system, inter-domain communication traffic can be large and it could be the performance bottle neck of the system. Jian Wang et. al. proposed XenLoop which is a inter-domain communication channel [15]. It bypasses the regular virtual network interfaces in Figure 1, yet provides transparency in the sense that the application programs can still use standard TCP/IP APIs. Vconsolidate is Intel's proprietary benchmark suite for evaluating consolidated server. Apparao, et. al., evaluated how performance metrics of each workload is changed under consolidation with varying cache sizes and CPU affinity [13].

V. Conclusion

In this paper, we presented and analyzed the network receiving performance of a Xen virtualized system with LRO. When only the guest domain for the network receiver was activated, LRO in the physical and virtual NICs improved the throughput by 5 and 12%, respectively. When another guest domain was activated and SPECjbb, a CPU-intensive workload, is executed, the network throughput was dropped linearly against the CPU utilization of SPECjbb. However, throughout the ranges of the workload mix used in this paper, LRO in physical and virtual NICs improved the throughput by 4% to 8% and 7% to 14%, respectively. Open questions and future work include the following. First, compared to the standard MTU of 1500B, we saw that MTU of 4000B improved the throughput of both native Linux and Xen-based systems by up to 27%. In the current system, we have to use the same MTU for all the network components within Xen. When we use a

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:15:14 UTC from IEEE Xplore. Restrictions apply.

standard MTU (1500B) for the physical NIC, we apply LRO there and use a larger MTU of the page frame size (4KB). In this configuration, we may reduce the packet handling overhead at the physical NIC and maximize the throughput of the Xen internal network. Second, we saw the changes in the relationship between the throughput and SPECjbb score around CPU utilization of 60% and reasons of this change should be identified. Lastly, various optimization techniques mentioned in Section IV should also be incorporated into our experimental system.

References

[1] Paul Barham et. al., "Xen and the Art of Virtualization," in *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP'03)*, pp164–177, October 2003.

[2] "AMD Virtualization," http://www.amd.com/virtualization .

[3] "Intel Virtualization Technology in Computing," http://www.intel.com/technology/virtualization/ .

[4] Aravind Menon, Alan Cox and Willy Zwaenepoel, "Optimizing Network Virtualization in Xen ," in *Proceedings of the 2006 USENIX Annual Technical Conference*, pp15–28, May–June 2006.

[5] Jose Renato Santos, et. al., "Bridging the Gap between Software and Hardware Techniques for I/O Virtualization," in *Proceedings of the Usenix '08*, pp29–42, June 2008.

[6] Leonid Grossman, "Large Receive Offload implementation in Neterion 10GbE Ethernet driver," in *Proceedings of the Ottawa Linux Symposium*, pp195–200, July 2005.

[7] Jan-Bernd Themann, "[RFC 0/1] lro: Generic Large Receive Offload for TCP traffic," Linux Kernel Mailing List archive, http://lkml.org/lkml/2007/7/20/250.

[8] Takayuki Hatori and Hitoshi Oi, "Implementation and Analysis of Large Receive Offload in a Virtualized System," in *Proceedings of the Virtualization Performance: Analysis, Characterization, and Tools (VPACT'08)*, April 2008.

[9] "Netperf: a network performance benchmark," Hewlett-Packard Company, Feb. 15, 1995.

[10] SPECjbb2005, http://www.spec.org/jbb2005/ .

[11] TPC-C, http://www.tpc.org/tpcc/ .

[12] VMmark, http://www.vmware.com/products/vmmark/

[13] Padma Apparao, et. al., "Characterization & analysis of a server consolidation benchmark," in *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp21–30, March 2008.

[14] Aravind Menon, et. al., "Diagnosing performance overheads in the Xen virtual machine environment," *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments (VEE05)*, pp13–23, June 2005.

[15] Jian Wang, Kwame-Lante Wright and Kartik Gopalan, "XenLoop: A Transparent High Performance Inter-VM Network Loopback," in *Proceedings of the 17th International Symposium on High Performance Distributed Computing (HPDC'08)*, pp109–118, June 2008.

Authorized licensed use limited to: Biblioteka Glowna i OINT. Downloaded on April 11,2010 at 21:15:14 UTC from IEEE Xplore. Restrictions apply.