J Inf Process Syst, Vol.13, No.1, pp.95~103, February 2017
https://doi.org/10.3745/JIPS.01.0013

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Feasibility and Performance Analysis of RDMA Transfer through PCI Express

Min Choi* and Jong Hyuk Park**

Abstract
The PCI Express is a widely used system bus technology that connects the processor and the peripheral I/O devices. The PCI Express is nowadays regarded as a *de facto* standard in system area interconnection network. It has good characteristics in terms of high-speed, low power. In addition, PCI Express is becoming popular interconnection network technology as like Gigabit Ethernet, InfiniBand, and Myrinet which are extensively used in high-performance computing. In this paper, we designed and implemented a evaluation platform for interconnect network using PCI Express between two computing nodes. We make use of the non-transparent bridge (NTB) technology of PCI Express in order to isolate between the two subsystems. We constructed a testbed system and evaluated the performance on the testbed.

Keywords
Interconnection Network, Non-transparent Bridge, PCI Express, RDMA

1. Introduction

Today, as the computing technology and network technology development, the amount of data to be processed by processors was increased, resulting in high-performance computing. Recent high-performance computing system are based on a system interconnect technology such as InfiniBand and Ethernet, and PCI Express.

PCI Express bus is originally a technology for the connection among processors, memory, and peripheral I/O devices. PCI Express has been developed from 2.5 Gbps Gen1 specification to Gen3 specification technology. With the increase of speeds and performance needs, there were many requests to make use of bus bandwidth efficiently on system interconnection network [1,2].

In field of high-performance computing and cluster computing, interconnection network technology has been widely studied for a long time. Majority of interconnection network technologies are InfiniBand and Ethernet. [3-5] use the PCI Express non-transparent bridge (NTB) to implement a multi-host system. The PCI Express NTB solves clock synchronization issues and provides isolation between hosts. [6] implemented the DMA transfer with a contiguous buffer using the PCI Express bus. This approach enables a DMA transfer for non-contiguous memory. This paper design and implemented

Manuscript received August 31, 2016; first revision November 11, 2016; accepted December 28, 2016.
Corresponding Author: Jong Hyuk Park (jhpark1@seoultech.ac.kr)
* Dept. of Information and Communication Engineering, Chungbuk National University, Cheongju, Korea (mchoi@cbnu.ac.kr)
**Dept. of Computer Science, Seoul National University of Science and Technology, Seoul, Korea (jhpark1@seoultech.ac.kr)

PC cluster system based on PCI Express interconnection network technology. In this paper, we use the PCI Express NTB to implement a DMA transfer of non-contiguous memory between two systems.

This paper is organized as follows: Section 1 provides introduction, In Section 2, we show our implementation of RDMA transfer on PCI Express NTB. In Section 3, we analyze and compare the existing and proposed scheme. Section 4 concludes this paper.

2. Implementation of RDMA Transfer on PCI Express

PCI Express is a further improved version than PCI bus technology developed by INTEL. PCI Express is a technology by which can be connected with various peripherals or host, support high I/O scalability and high-performance, high-speed communication. PCI Express NTB is widely used in the multi-host and host-failover. Fig. 1 shows the system architecture in which two hosts are connected through PCI Express NTB [7,8].

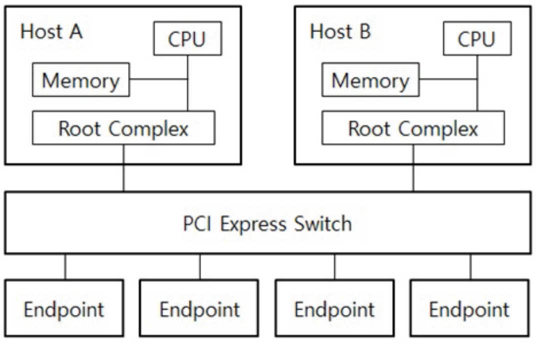


Fig. 1. Multi-host system hierarchy.

PCI Express is transmitting and receiving data (full-duplex) through an independent link tied to an integrated line in one lanes of transmit and receive. PCI Express may be used by a plurality of lanes (×2, ×4, ×8, ×16) to control the links from a single lane width (X1). PCI Express has the 8 Gbps performance of a single-lane Gen3 data rate. Table 1 represents the data transfer rates depending on the generation and link width [6,9].

Table 1. Transfer speed of PCI Express

Generation	Lane (width)				
	X1	X2	X4	X8	X16
Gen1 (Gbps)	2.5	5	10	20	30
Gen2 (Gbps)	5	10	20	40	60
Gen3 (Gbps)	8	16	32	64	96

PCI Express NTB is used for the mutual isolation in the PCI Express based interconnection network. NTB has integrated two conceptual interfaces (endpoints) for two roles in one port. We cannot see the fact that the two systems are connected in the BIOS emulation process due to the logically separated interface of PCI Express NTB. Fig. 2 depicts an example in which two systems are connected through PCI Express based interconnection network [10].

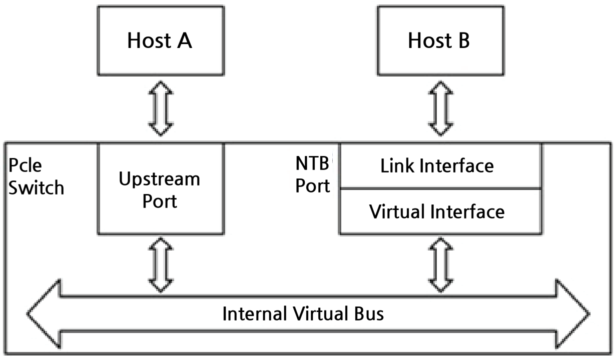


Fig. 2. Interconnect system with non-transparent bridge.

In Fig. 2, Host A and B can communicate via logical endpoint interface of virtual side and link side through address translation. It is necessary to know the address of the external memory when the host address translation. There are some scratchpad registers that can be accessed from each of the interface. Two hosts may share data with each other via the scratchpad register. It can also cause an interrupt to the other party through the Doorbell register. The interconnection network determines where the packet is directed through the bus and device information field in the packet header.

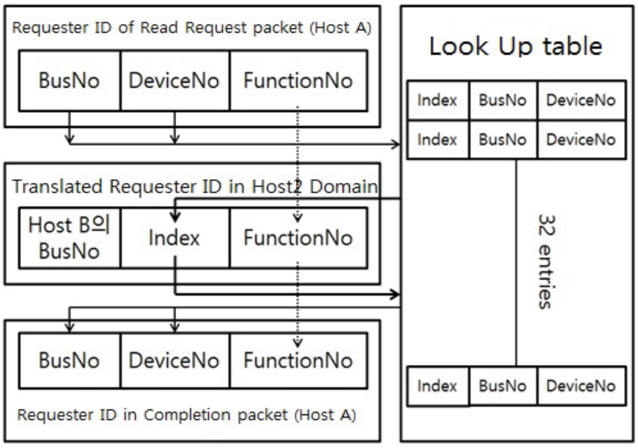

Requester ID of Read Request packet (Host A)
BusNo
DeviceNo
FunctionNo
Translated Requester ID in Host2 Domain
Host B의 BusNo
Index
FunctionNo
BusNo
DeviceNo
FunctionNo
Requester ID in Completion packet (Host A)
Look Up table
Index
BusNo
DeviceNo
32 entries

Fig. 3. Requester ID translation sequence on PCI Express bus.

The bus number and device number are different, so the interconnection network returns a proper ID depending on the bus number and device number during address translation. In order to do this, there is an address-translation lookup table (LUT) to which we can register an ID, lookup the ID by the index. Fig. 3 shows the address translation process of a PCI Express packet. There are also necessary to translate memory address on remote DMA transfer, because the destination address on target machine has totally different address management from that on source machine. We make use of direct address translation in that the offset address are maintained but only the base address is replace by that of the target machine as shown in Fig. 4.

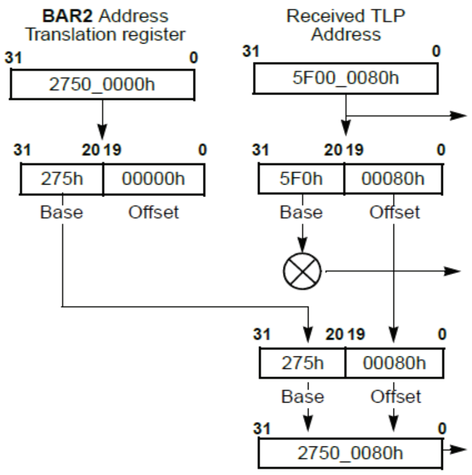


Fig. 4. Memory address translation sequence.

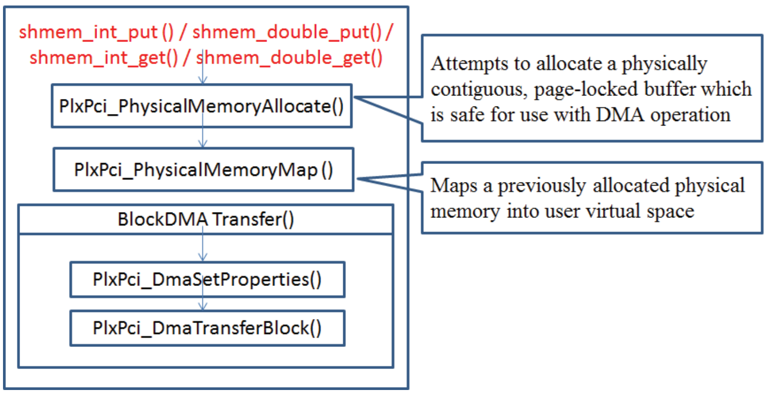

shmem_int_put () / shmem_double_put() / shmem_int_get() / shmem_double_get()
PlxPci_PhysicalMemoryAllocate()
PlxPci_PhysicalMemoryMap ()
BlockDMA Transfer()
PlxPci_DmaSetProperties()
PlxPci_DmaTransferBlock()
Attempts to allocate a physically contiguous, page-locked buffer which is safe for use with DMA operation
Maps a previously allocated physical memory into user virtual space

Fig. 5. Our RDMA implementation.

After the address translation as shown in Fig. 4, we implemented the remote DMA (RDMA) functions as the process in Fig. 5. The standard interface of RDMA is described in red colored in Fig. 6, for example, shmem_int_put(), shmem_double_put(), shmem_int_get(), and shmem_doule_get(). After those APIs are called, the first task is attempt to allocate a physically contiguous, page-locked buffer which is safe for use with DMA operation. Another step is to map a previously allocated physical memory into user virtual space. Then, the memory allocation and mapping are done, so the blockDMATransfer() function will be called and executed.

We also implemented interrupt messaging mechanism using doorbell interrupt. To realize this functionality, we have to access the NT0 link side configuration space and NT0 virtual side configuration space as shown in Fig. 6. For sending an interrupt on link side, we make use of address 0x3FC5C. For sending an interrupt on virtual side, we make use of address 0x3EC4C.

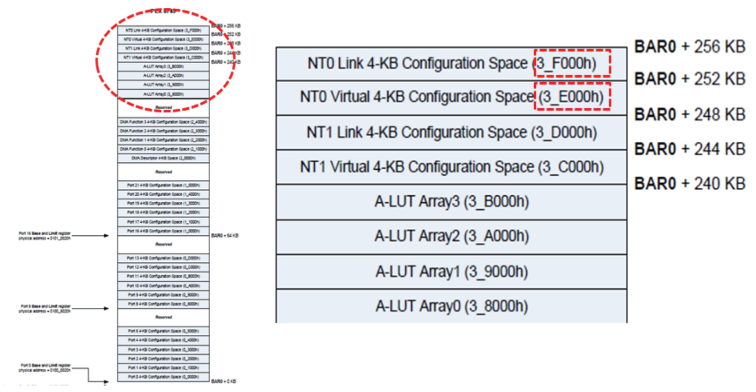

NT0 Link 4-KB Configuration Space (3_F000h)
NT0 Virtual 4-KB Configuration Space (3_E000h)
NT1 Link 4-KB Configuration Space (3_D000h)
NT1 Virtual 4-KB Configuration Space (3_C000h)
A-LUT Array3 (3_B000h)
A-LUT Array2 (3_A000h)
A-LUT Array1 (3_9000h)
A-LUT Array0 (3_8000h)
BAR0 + 256 KB
BAR0 + 252 KB
BAR0 + 248 KB
BAR0 + 244 KB
BAR0 + 240 KB

Fig. 6. Our RDMA implementation.

3. Experimental Analysis

This section provides the design and implementation of evaluation system. Table 2 shows the experimental environment of the system. We make use of Core-i5 3.2 GHz, with 2 GB 1333 MHz DDR memory. The OS is Centos Linux 7 64 bit, kernel 3.10.0. The interconnection network is based on PEX8749 Chipset of PLX Technology.

Table 2. Experiment environment

Host PC hardware environment	
M/B	GIGABYTE GA-H61M-DS2V
Processor	Intel Core i5-3470 CPU @ 3.20GHz X 4
Memory	Samsung DDR3 1333 MHz 2 GB
O/S	Centos Linux 7 64 bit / Kernel: 3.10.0-327.el7
Interconnect hardware environment	
PCI Express NTB evaluation board	PLX PEX8749 RDK / 48 Lane, 18 Port / PCI Express Gen3 Switch
NIC	PLX PEX8732 Cable Adapter X 4
Driver	PLX SDK Device Driver

Table 3. PCI Express link performance

PCI Express version	Lind code	Transfer rate (GT/s)	Bandwidth	
			Per lane (Gbit/s, MB/s)	In a ×16 (16-lane) slot (Gbit/s, GB/s)
1.0	8b/10b	2.5	2, 250	32, 4
2.0	8b/10b	5	4, 500	64, 8
3.0	128b/130b	8	7.877, 984.6	126.032, 15.754
4.0	128b/130b	16	15.754, 1969.2	252.064, 31.508

Table 3 provides the performance of PCI Express link. These days we usually utilize the PCI Express Gen3, so we can get link performance up to 8 Gbit/s, theoretically. But, the actual performance depends on variable situation such as link status, protocol overhead, and so on.

To connect a host PC to PEX8749 evaluation board, we need a PEX8732 chipset based network interface card for each side. Since the evaluation host supports up to PCI Express Gen2 specification, we make use of Gen2 specification both for host and PCI Express NTB evaluation board. Evaluation software runs on Centos 7.0 Linux operating system (kernel 3.10.0). For device control, PEX8749 device driver was installed as shown in Fig. 7.

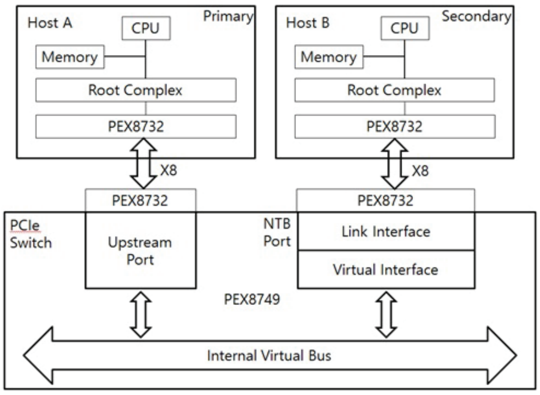


Fig. 7. Implemented interconnection network system.

First, we show the performance difference between non-DMA transfer using ordinary memcpy and PCI Express RDMA transfer and we also evaluate the performance improvement between block RDMA transfer and scatter-gather RDMA.

In each experiment, each host transmits the data through buffer located in BAR (base address register). The actual physical address, the virtual address in kernel, and buffer size are send and received to each other via scratchpad register in host. In this case, the host buffer ware assigned to the same size. We transmit the buffered data to the target host after address translation by using the BAR register.

Since the BAR memory size is 1 MB, we limit the buffer size to a maximum of 1 MB. In this experiments, we transfer the data to Host A from Host B by using one DMA channel. We measured the transmission rate with respect to the size and number of buffers. When an interrupt occurs at the end of each transfer, we measured the average data transfer rate as a way to send data back and forth.

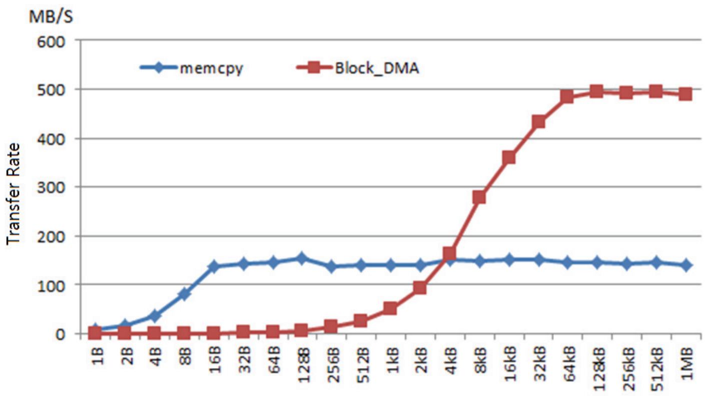


Fig. 8. Result of non-DMA and block DMA transfer (MB/s).

Fig. 8 depicts the evaluation between non-DMA transfer and block RDMA transfer. The x-axis shows the size of transfer data and the y-axis represents the transfer rate as MB/s, commonly in both graphs. When the size of the buffer is small, memcpy and DMA transfer shows low performance at the same time. The larger the size of the buffer gets better performance in DMA transfer. This is because the data transfer at a time to increase. When the data size is below 2 kB, memcpy showed higher transmission rate than DMA transfer. Even if we increase the buffer size in memcpy, it tends to be satisfied at the transfer rate of up to 150 MB/s. In more than 4kB buffer size, we got a relatively high performance of up to about 500 MB/s in DMA transfer.

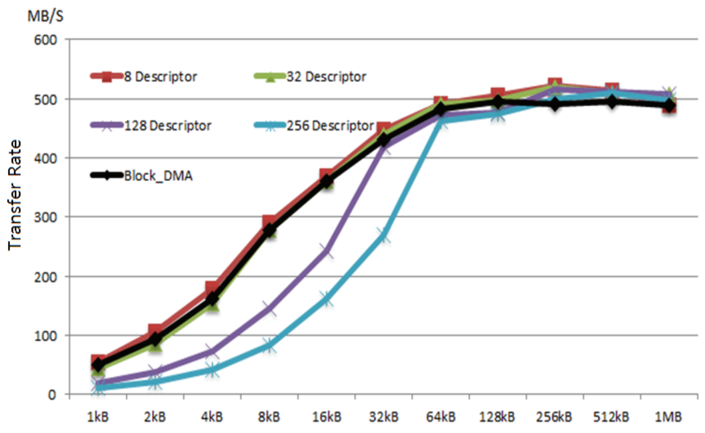


Fig. 9. Result of block DMA and SGDMA transfer (MB/s).

Scatter-gather DMA (SGDMA) transfer descriptor has the information which is required for DMA such as source address, destination address, and byte count. This descriptor enables us to transmit multiple DMA buffers or discontinuous address spaces. Fig. 9 shows the performance of SGDMA transfer. Fig. 9 shows the evaluation between block RDMA transfer and scatter-gather RDMA transfer. The x-axis shows the size of transfer data and the y-axis represents the transfer rate as MB/s, commonly in both graphs. We used the 8, 32, 128, 256 descriptors for SGDMA transfer. We specify the data size for each descriptor to a value obtained by dividing evenly the same size to the buffer size. The smaller the size of the buffer, the more the number of descriptor, we got lower the transmission rate of the SGDMA. As a result of execution with 8 descriptors, we got up to 520 MB/s speed. This is slightly higher than the performance block DMA.

4. Conclusions

Non-DMA (memcpy) and DMA block exhibited a maximum transmission speed of each to increase the larger the size of the buffer transfer rate of about 150 MB/s and 500 MB/s. For SGDMA larger the size of the buffer, the number of descriptor was confirmed to represent a slightly better performance than the DMA block. The transfer rate is low in transmission rate of up to about 520 MB/s. Due to the DMA engine device drivers or user-level overhead of the application due to the limitation of the size of the buffer compared to the theoretical performance shows a lower performance.

By reducing the overhead due to the processing of the next buffer size and if the PCI Express software that take advantage of the bandwidth that is determined to theoretically support can significantly improve the performance of the DMA transfer.

Evaluation result shows that the non-DMA transfer performance is 150 MB/s and the block RDMA transfer is improved to 500 MB/s. Finally, SGDMA performance is shown up to 520 MB/s. In this paper, it delivers high-performance computing and widely used interconnect technology of the non-transparent PCI Express switch and connect two PC interconnect system using cluster computing bridging that the DMA transfer performance were analyzed between two PC.

Acknowledgement

This research was jointly supported by a Research and Development of Dual Operating System Architecture with High-Reliable RTOS and High-Performance funded by ETRI (No. R0101-16-0081) and a Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. NRF-2016R1C1B1012189).

References

[1] Y. W. Kim, Y. Ren, and W. Choi, “Design and implementation of an alternate system interconnect based on PCI express,” *Journal of the Institute of Electronics and Information Engineers*, vol. 52, no. 8, pp. 74-85, 2015.

[2] J. Liu, A. Mamidala, A. Vishnu, and D. K. Panda, “Evaluating InfiniBand performance with PCI express,” *IEEE Micro*, vol. 25, no. 1, pp. 20-29, 2005.

[3] H. Wong, “PCI express multi-root switch reconfiguration during system operation,” Ph.D. dissertation, Department of Electrical Engineering and Computer, Massachusetts Institute of Technology, Cambridge, MA, 2011.

[4] L. Mohrmann, J. Tongen, M. Friedman, and M. Wetzel, “Creating multicomputer test systems using PCI and PCI express,” in *Proceedings of 2009 IEEE AUTOTESTCON*, Anaheim, CA, 2009, pp. 7-10.

[5] J. Gudmundson, “Enabling multi-host system designs with PCI Express technology,” 2004 [Online]. Available: http://www.rtcmagazine.com/articles/view/100015.%208.

[6] L. Chen, W. Zhou, and Q. Wu, “A design of high-speed image acquisition card based on PCI EXPRESS,” in *Proceedings of 2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, Taiyuan, China, 2010, pp. 551-554.

[7] Top500.org, “2015 Interconnect Family Statistics,” [Online]. Available: http://top500.org/statistics/list.

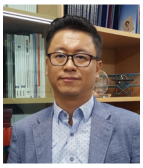
[8] V. Krishnan, “Towards an integrated IO and clustering solution using PCI express,” in *Proceedings of 2007 IEEE International Conference on Cluster Computing*, Austin, TX, 2007, pp. 259-266.

[9] L. Rota, M. Caselle, S. Chilingaryan, A. Kopmann, and M. Weber, “A new DMA PCIe architecture for Gigabyte data transmission,” in *Proceedings of 19th IEEE-NPSS Real Time Conference*, Nara, Japan, 2014, pp. 1-2.

[10] H. Kavianipour and C. Bohm, “High performance FPGA-based scatter/gather DMA interface for PCIe,” in *Proceedings of IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, Anaheim, CA, 2012, pp. 1517-1520.

Min Choi

He received the B.S. degree in Computer Science from Kwangwoon University, Korea, in 2001, and the M.S. and Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 2003 and 2009, respectively. From 2008 to 2010, he worked for Samsung Electronics as a Senior Engineer. Since 2011 he has been a faculty member of Department of Information and Communication of Chungbuk National University. His current research interests include embedded system, microarchitecture, and cloud computing.



James J. (Jong Hyuk) Park http://orcid.org/0000-0003-1831-0309

He received Ph.D. degrees in Graduate School of Information Security from Korea University, Korea and Graduate School of Human Sciences from Waseda University, Japan. From December 2002 to July 2007, Dr. Park had been a research scientist of R&D Institute, Hanwha S&C Co., Ltd., Korea. From September 2007 to August 2009, he had been a professor at the Department of Computer Science and Engineering, Kyungnam University, Korea. He is now a professor at the Department of Computer Science and Engineering and Department of Interdisciplinary Bio IT Materials, Seoul National University of Science and Technology (SeoulTech), Korea. Dr. Park has published about 200 research papers in international journals and conferences. He has been serving as chairs, program committee, or organizing committee chair for many international conferences and workshops. He is a founding steering chair of some international conferences—MUE, FutureTech, CSA, UCAWSN, etc. He is editor-in-chief of *Human-centric Computing and Information Sciences (HCIS)* by Springer, *The Journal of Information Processing Systems (JIPS)* by KIPS, and *Journal of Convergence (JoC)* by KIPS CSWRG. He is Associate Editor / Editor of 14 international journals including 8 journals indexed by SCI(E). In addition, he has been serving as a Guest Editor for international journals by some publishers: Springer, Elsevier, Wiley, Oxford University press, Hindawi, Emerald, Inderscience. His research interests include security and digital forensics, human-centric ubiquitous computing, context awareness, multimedia services, etc. He got the best paper awards from ISA-08 and ITCS-11 conferences and the outstanding leadership awards from IEEE HPCC-09, ICA3PP-10, IEE ISPA-11, and PDCAT-11. Furthermore, he got the outstanding research awards from the SeoulTech in 2014. Dr. Park's research interests include human-centric ubiquitous computing, vehicular cloud computing, information security, digital forensics, secure communications, multimedia computing, etc. He is a member of the IEEE, IEEE Computer Society, KIPS, and KMMS.