PCI Express를 통한 Socket 통신 구현에 대한 연구

심철*, 최민*
*충북대학교 정보통신공학부
e-mail:eisen@cbnu.ac.kr

A Study on Implementation of Socket communication through PCI Express

Cheol Sim*, Min Choi*
*Dept. of Information and Communication, Chungbuk University

요 약

오늘날 PCI Express는 프로세서와 시스템 장치들과 연결을 위한 표준 I/O 인터페이스로 널리 사용되고 있다. 고속, 저전력, 고효율의 특성을 가진 PCI Express는 기존 네트워크 연결망의 대안으로 고려되고 있다. 본 논문에서는 이러한 PCI Express를 서로 다른 시스템 간에 통신을 도와주는 PCI Express Interconnect를 통해 네트워크 연결망을 형성하고, 기존의 TCP/IP 프로토콜 스택을 거쳐 Socket 통신을 하는 Application을 PCI Express를 거쳐 통신할 수 있도록 하는 네트워크 모듈을 개발해보고자 한다. 이를 위해 관련 연구를 조사하여 네트워크 Family를 새로 정의하여 TCP/IP 프로토콜 스택을 거치지 않는 PCI Express 통신 프로토콜이 구현 중에 있다.

1. 서론

오늘날, PCI Express는 프로세서와 시스템 장치들과의 연결을 위한 표준 I/O 인터페이스로 널리 사용되고 있다. PCI Express의 고속, 저전력, 고효율 등의 특성은 기존의 네트워크 구조의 대안으로 고려되고 있다[1].

PCI Express는 기존의 고성능 시스템에서 사용되고 있는 인피니밴드(InfiniBand)와 이더넷(Ethernet) 연결망과 비교하였을 때 약 6% 정도 높은 프로토콜 효율을 보이며, End-to-End 지연시간도 이더넷에 비하면 10배에서 30배 정도로 작다[2]<표 1>. 이러한 PCI Express의 특성을 활용할 수 있도록 PCI Express를 시스템 간의 네트워크로 만들어주는 인터커넥터들이 개발되고 있다.

<표 1> PCIe와 Ethernet, InfiniBand의 효율

	PCI Express	Ethernet	InfiniBand
프로토콜 효율*	93%	86%	88%
End-to-End Latency	~1μs	~10μs to 30μs	~1μs to 2μs

*Payload : 256Byte

본 논문에서는 Socket Application으로 통신을 할 때, TCP/IP 프로토콜 스택이 아닌 PCI Express 인터커넥터를 통해 RDMA 통신이 가능한 네트워크 모듈을 연구해보고자 한다.

2. PCI Express Interconnect

PCI Express에서는 NTB(Non-Transparent Bridge) 기술이 도입되었다. PCI Express를 통해 서로 다른 시스템을 연결하여 네트워크를 형성할 때, SSC(Spread Spectrum Clock)을 사용하는 특성이 있다. 이러한 특성 때문에 클록 타이밍 충돌과 버스주소체계 충돌하는 문제가 생긴다[1]. NTB 포트는 각 노드들을 논리적으로 분리시켜 각각의 시스템들은 격리(Isolation)되고, 서로 다른 클록 타이밍과 버스주소체계의 변환을 통해 앞서 언급한 문제점을 해결할 수 있다.

각 Non-Transparent 노드들은 BAR(Base Address Register)들을 가지고 있으며, 일부 메모리 영역이 노드의 메모리에 연관되어 있어서 다른 원격 노드들의 메모리 영역에 주소 변환을 통해 직접 접근할 수 있다(그림 1)[3].

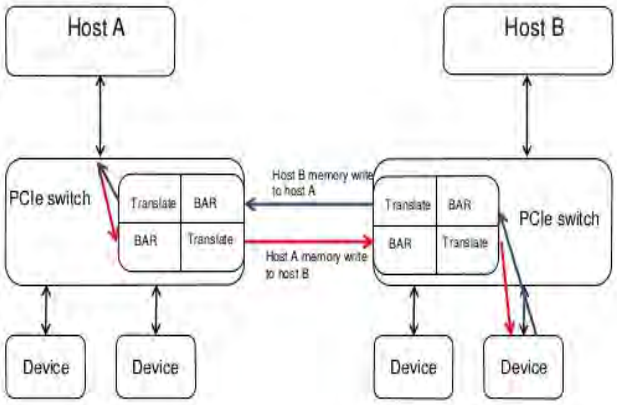
그림 2는 PCI Express Interconnect를 이용한 네트워크 연결망을 표현한 도식이다.

<표 2> PCIe의 Gen과 Lane에 따른 전송 속도[3]

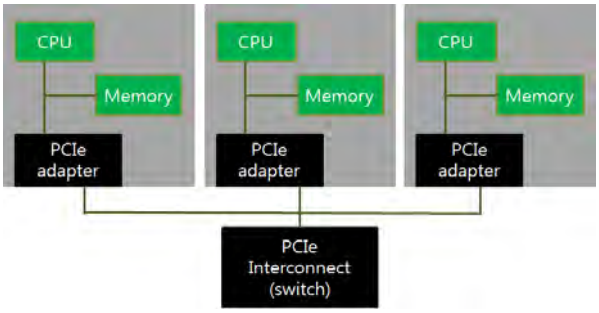
Generation	Lane(Width)				
	x1	x2	4	x8	x16
Gen1 (Gbps)	2.5	5	10	20	60
Gen2 (Gbps)	5	10	20	40	80
Gen3 (Gbps)	8	16	32	64	128

본 논문에서 제안하는 PCI Express Interconnect switch는 Broadcom 社에서 제작한 PEX8749CA

RDK를 사용할 것이다. PEX8749CA RDK는 총 48-lane, 18개의 port, PCIe Gen3 전송속도를 지원한다<표 2>.



(그림 1) PCI Express Interconnect를 이용한 원격 노드의 자원(Resource) 사용



(그림 2) PCI Express Interconnect 연결망 도식

3. Socket Direct Protocol

기존의 TCP/IP 프로토콜은 인피니밴드나 PCI Express를 통한 HPC(High Performance Computing)에 적합하지 않다. 커널 영역과의 헤더와 데이터를 처리하는데 있어 메시지 전송에 지연시간이 발생하게 된다[3].

SDP(Socket Direct Protocol)란, 이러한 문제를 RDMA(Remote Direct Memory Access)를 사용해서 해결한 프로토콜이다[4][5].

4. PCI Express 통신을 위한 네트워크 모듈 구현

유저 영역에서 Server-Client Socket Application을 실행하면 각 Socket API에 대응하는 커널 영역 API에 시스템 콜을 호출하여 PCI Express를 통해서 데이터를 송수신해야 한다. 커널 영역에서 Socket API가 TCP/IP 프로토콜을 거치지 않고 본 논문에서 제시하는 네트워크 모듈을 통해 통신하기 위해 PF_PACKET이라는 프로토콜 패밀리를 새로 정의하였다.

그림 3은 socket(), bind(), connect(), accept() 등의 유저 영역 Socket API에 대응하여 구현해야 될 커널 API들이다.

Socket Application에서 PCI Express를 사용하는 네트워크 커널 모듈을 사용하기 위해 Socket 생성 시 앞서 제시한 PF_PACKET 프로토콜 Family를 사용해야 된다. 이는 family 필드가 PF_PACKET으로 선언된 net_proto_family 구조체를 sock_register() 커널 API를 사용하여 등록하는데, 그림 4와 같이 작성할 수 있다.

```
static struct proto_ops packet_ops_spkt = {
        .family         = PF_PACKET,
        .owner          = THIS_MODULE,
        .release        = packet_release,
        .bind           = inet_bind,
        .connect        = packet_connect,
        .socketpair     = packet_socketpair,
        .accept         = packet_accept,
        .poll           = packet_poll,
        .ioctl          = packet_ioctl,
        .listen         = packet_listen,
        .shutdown       = packet_shutdown,
        .setsockopt     = packet_setsockopt,
        .getsockopt     = packet_getsockopt,
        .sendmsg        = packet_sendmsg,
        .recvmsg        = sock_common_recvmsg,
        .sendpage       = packet_sendpage,
};

static struct net_proto_family packet_family_ops = {
        .family     = PF_PACKET,
        .create     = packet_create,
        .owner      = THIS_MODULE,
};
```

(그림 3) Socket API에 대응하는 커널 영역에서 구현할 함수들

```
static int __init packet_init(void)
{
        int rc = proto_register(&packet_proto, 0);

        if (rc != 0)
                return rc;

        rc = sock_register(&packet_family_ops);

        if (rc != 0) {
                proto_unregister(&packet_proto);
                return rc;
        }

        printk("%s: initialized at %lu.\n", __func__, jiffies);
        return rc;
}
```

(그림 4) 커널 모듈 삽입 시 수행되는 family 등록 루틴 구현 코드

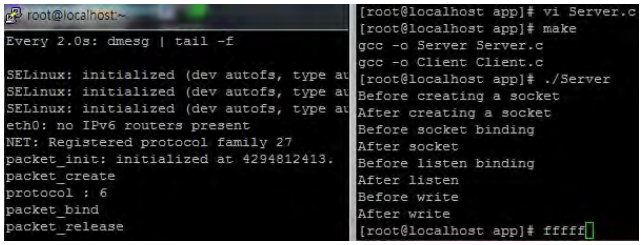
유저 영역에서 Socket 생성 시 호출하는 socket() API가 호출될 때, 그림 5와 같이 커널에서는 Socket을 위한 메모리 할당과 해당 Socket이 수행하는 함수들을 등록해 준다. 이렇게 작성한 커널 모듈을 삽입하고 Socket Application에서 사용할 Socket의 Family를 커널 모듈에서 사용하는 Family로 등록하여 실행시키게 된다.

```
static int packet_create(struct socket *sock, int protocol)
{
        printk("packet_create\n");
        struct sock *sk;
        struct packet_sock *po;

        if (sock->type != SOCK_STREAM) {
                printk("SDP: unsupported type %d.\n", sock->type);
                return -ESOCKTNOSUPPORT;
        }

        //IPPROTO_IP is a wildcard match
        if (protocol != IPPROTO_TCP && protocol != IPPROTO_IP) {
                printk("SDP: unsupported protocol %d.\n", protocol);
                return -EPROTONOSUPPORT;
        }

        printk("protocol : %d\n", protocol);

        sk = sk_alloc(PF_INET_SDP, GFP_KERNEL, &packet_proto, 1);

        if (!sk) {
                printk("SDP: failed to allocate socket.\n");
                return -ENOMEM;
        }

        sock->ops = &packet_ops_spkt;
        sock_init_data(sock, sk);
        sk->sk_protocol = 0x0;
        sk->sk_family = PF_INET;
        po = pkt_sk(sk);
        po->num = protocol;
        sk->sk_destruct = packet_sock_destruct;
        sock->state = SS_UNCONNECTED;
        return 0;
}
```

(그림 5) socket() API 호출 시 커널 영역에서 Socket 생성 구현 코드

5. 결론 및 향후 연구계획

본 논문은 Socket Application이 TCP/IP 프로토콜 스택을 거치지 않고 PCI Express를 통해 RDMA로 데이터를 송수신하는 네트워크 모듈의 구현 방법에 대해 조사하여 제안한다. 현재 Socket Application이 4장에서 제시한 프로토콜 Family를 사용하는 네트워크 모듈에 접근하는 단계까지 구현이 진행되었다(그림 6). 앞으로 4장에서 제시한 커널 함수들을 구현할 것이고, 기존의 이더넷 통신, 인피니밴드와의 비교를 통해 실제 성능 향상이 얼마나 이루어졌는지 기대한다.



(그림 6) PF_PACKET 프로토콜 패밀리를 사용하는 네트워크 모델에 접근한 결과

참고문헌

[1] Young Woo Kim, Ye Ren, WonHyuk Choi. "Design and Implementation of an Alternate System Interconnect based on PCI Express." Journal of the Institute of Electronics and Information Engineers, 52.8 (2015.8): 74-85.

[2] Vijay Medury, "PCI Express in Clustering," High Speed Interconnects Seminar, Linley Group, Nov., 2010.

[3] Ahmed Bu-Khamsin, "Socket Direct Protocol over PCI Express Interconnect: Design, Implementation and Evaluation," MS Thesis, Simon Fraser University, 2012

[4] RDMA Consortium., http://www.rdmaconsortium.org/.

[5] J. Pinkerton. Sockets Direct Protocol (SDP) for iWARP over TCP (v1.0), 2003.