ISBN 978-89-6211-311-2

기술 분석: InfiniBand 기술 동향 분석

부 서: 슈퍼컴퓨팅센터
작성자: 조혜영, 차광호, 김성호

한국과학기술정보연구원

Korea Institute of Science and Technology Information

목 차

제 1장 서론

1.1 개요

인피니밴드(Infiniband)는 컴퓨터 서버가 입출력 장치 및 타 서버와 상호 연결하는 방법을 획기적으로 개선한 시스템 연결망 기술이다. 인피니밴드 아키텍처는 인피니밴드통상협회(Infiniband Trade Association : IBTA)가 2000년 10월 발표한 규격서에 최초로 정의된 후 개정되어 오고 있다[1].

초기 인피니밴드 기술은 PCI(Peripheral Component Interconnect) 버스의 대체 기술로 출발하였다[3]. 인피니밴드의 1차적 목표는 PCI 버스의 한계(성능 병목현상, 확장성 등)을 극복하는 것에서 시작되었으나 현재는 타 서버와의 상호 연결하는 시스템 연결망 기술, 여러 서버들 간의 클러스터링을 위한 시스템 연결 기술을 향상시키는 쪽으로 발전하고 있다. 이러한 인피니밴드 기술은 클러스터 컴퓨팅을 위한 통합 연결망을 목적으로 개발되고 있는 차세대 시스템 연결망 기술로 슈퍼컴퓨팅센터에서 보유해야 할 핵심기술이라 판단되며, 이에 본 문서에서는 인피니밴드에 대한 기술 및 시장 동향을 기술한다.

1.2 인피니밴드 역사

인피니밴드 기술이 출현한 역사에 대해서 간단히 알아보면 인피니밴드 아키텍처는 관련 업계를 중심으로 두 개의 표준화를 진행해 온 두 개의 연결망 기술인 NGIO(Next Generation I/O)와 FIO(Future I/O) 기술로부터 비롯된다[4]. 1996년 인텔은 VI(Virtual Interface) 아키텍처를 기반으로 한 입출력 기술 개발을 주도하였고 이 기술은 NGIO(Next Generation I/O)라고 명명되었고, 델 컴퓨터, 인텔, 썬, NEC, 히다찌, 후지쯔, 지멘스 등이 기술 개발에 동참하였다. 거의 비슷한 시점에, 또 하나의 표준화 노력으로서 FIO(Future I/O)가 IBM, 컴팩, 3Com, 시스코, HP, 어탭텍 등을 중심으로 개발되기 시작하였다. NGIO는 대량의 소규모 서버를 초점으로 맞추었고, 반면에 FIO는 고성능 플랫폼에 초점을 맞추었다. 이후 양 진영은 통합이 더 바람직하다고 판단하고, 199년 10월 IBTA를 결성하였다. 2000년 10월 IBTA는 개발자 회의를 개최하고 인피니밴드 표준 규격서(버전 1.0)를 완성하

여 업계에 공표하였다.[2]

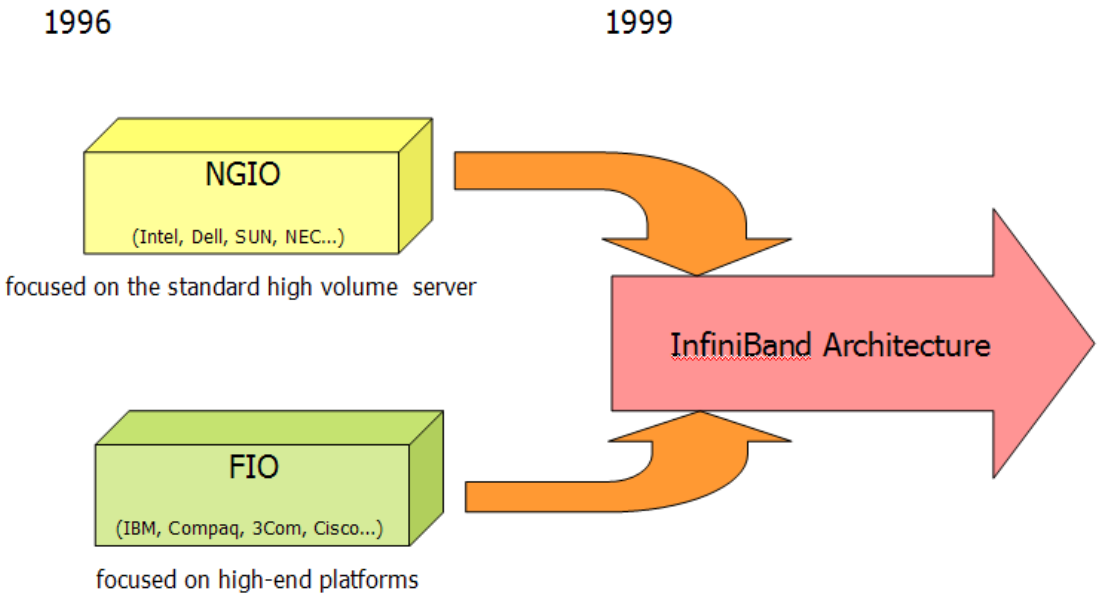


그림 1. History of the InfiniBand Architecture

제 2장 InfiniBand 연결망 구조

2.1 개요

InfiniBand 구조(InfiniBand Architecture: IBA)는 입출력 처리에 최적화된 입출력 연결망과 프로세서 상호간 통신에 최적화된 클러스터 연결망을 통합한 차세대 클러스터 연결망 기술이다. IBA는 통신 및 관리 메커니즘을 포함하고 있으며, 소규모 컴퓨터 시스템에서 대규모 인터넷 서버 및 슈퍼컴퓨터에 이르기까지 다양한 영역에서 사용이 가능하다.

IBA는 복수 개의 디바이스간에 동시 전송을 가능하게 하며, 높은 전송 대역폭과 낮은 전송 지연시간을 제공하고 높은 확장성 보장을 위하여 스위치 기반형 연결망으로 정의되었다. IBA는 통신 처리에 있어서 프로세서의 부하를 최소화할 수 있도록 프로토콜 및 프로토콜 처리 하드웨어를 정의하고 있으며, 이를 통하여 무복사(zero processor-copy) 데이터 전송, 커널 오버헤드 최소화, DMA(Direct Memory Access), 고확장성 및 고가용성 기능을 제공하고 있다.

2.2 토폴로지와 연결망 구성 요소

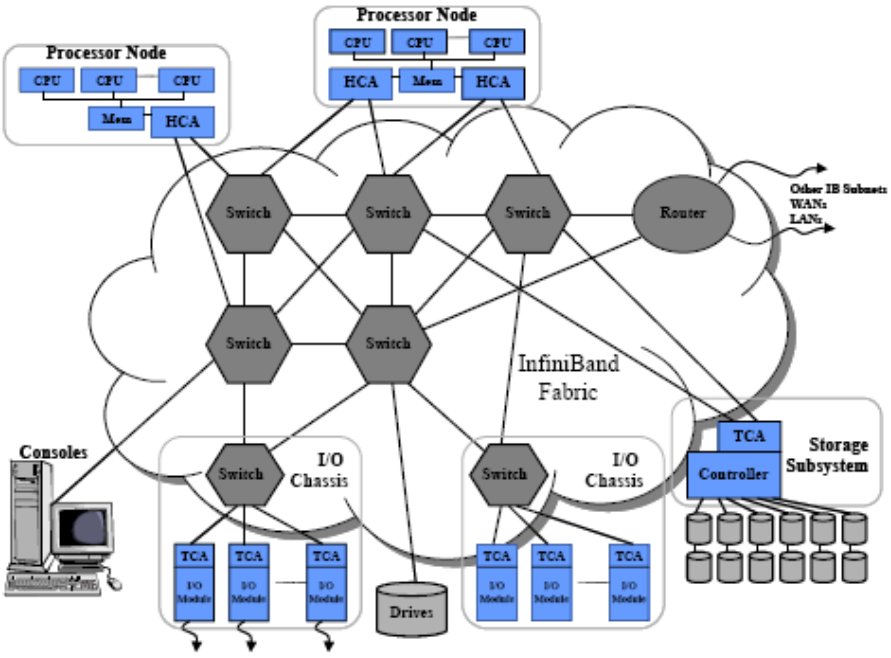


그림 2. 인피니밴드 시스템 연결망

인피니밴드 연결망에는 호스트 채널 어댑터(HCA: Host Channel Adapter), 타겟 채널 어댑터(TCA : Target Channel Adapter), 스위치(switch), 그리고 라우터(router)로 이루어져 있다. HCA는 서버 내에 존재하는 인터페이스로서, IBA 연결망은 물론 서버 메모리 및 프로세서와 직접 통신한다. TCA는 스트리지 어탭터와 네트워크 어댑터 두 종류로 크게 분류되며, 호스트 컴퓨터와 입출력 장치 사이의 일종의 브리지 역할을 수행한다. 스위치는 HCA와 TCA를 연결하여 서브넷을 구성하고 네트워크 트래픽을 제어한다. 라우터는 서브넷과 타 서브넷을 연결해 주는 일종의 브리지이다. 위의 그림은 인피니밴드 기반 시스템 연결망의 구성 예를 나타낸 것이다.

채널 어댑터는 지역 또는 원격 DMA 기능을 수행하는 프로그래머블 DMA 엔진으로 정의될 수 있으며, 가상 메모리 보호 메커니즘을 포함하고 있다. 전송 요구를 해독하여 해당 전송 요구를 처리하기 위한 패킷을 발생시키고 수신하는 기능을 수행하며, 기능에 따라서 HCA와 TCA로 구분된다. HCA는 다른 호스트 또는 입출력 장치와 통신을 위하여 호스트에게 채널 인터페이스를 제공하는 장치로 사용자 수준 프로그램에서 Verb 라는 소프트웨어 인터페이스를 사용하여 메시지를 송출하고 수신하는 기능을 수행한다. 호스트 프로세서에 의해서 발생된 메시지 전송 요구를 해독하여 호스트 메모리에서 데이터를 읽어서 IBA 연결망으로 송출하고 또한 수신된 메시지를 해독하여 호스트 메모리에 직접 쓰는 작업을 수행한다. HCA는 스위치와 TCA 모두에 직접 연결할 수 있으므로 시스템 구성시 규모에 따라 유연하게 시스템을 구성할 수 있다. TCA는 입출력 장치와 연결망을 이어주는 장치로 디스크 컨트롤러, 네트워크 컨트롤러, RAID(Redundant Array of Inexpensive Disks) 컨트롤러 등과 같은 다양한 입출력 장치를 IBA 연결망에 정합할 수 있게 한다.

채널 어댑터는 그림3과 같은 개념적 구조를 갖는다. 채널 어댑터(HCA)의 구성, 관리 및 동작 명령 전달을 위해서 Verb를 정의하고 있다. Verb는 사용자(응용 프로그램)의 메시지 전송 또는 데이터 서비스 요구를 채널 어댑터에 전송하기 위해서 다양한 기능을 정의하고 각 기능에서 사용되는 파라미터를 정의하고 있다. 채널 어댑터는 복수 개의 포트를 가질 수 있으며, 각 포트는 하나의 LID(Local ID) 또는 특정 영역의 LID를 할당 받는다. 각 포트는 송신 및 수신을 동시에 수행할 수 있도록 독립적인 송신 버퍼와 수신 버퍼를 가지며, 독립적인 흐름 제어에 의해서 동작하는 가상 레인(Virtual Lane: VL)을 통해서 데이터

송수신을 위한 채널을 구성한다. 채널 어댑터는 가상 주소와 물리 주소간의 변환 및 접근 권한을 관리하기 위한 MTP(Memory Translation & Protection) 메커니즘을 지원하며, 이 기능을 통하여 채널 어댑터는 운영체제의 간섭없이 메시지 버퍼로 사용되는 사용자 영역의 호스트 메모리를 직접 접근한다. 서브넷 매니저는 IBA 연결망의 형상 관리 및 채널 어댑터 형상을 제어하며, 각 포트에 LID를 할당한다. 이를 위해서 채널 어댑터는 서브넷 매니지먼트 에이전트(Subnet Management Agent: SMA)를 가지고 있으며, 이를 통해서 서브넷 매니저의 요청에 응답한다.

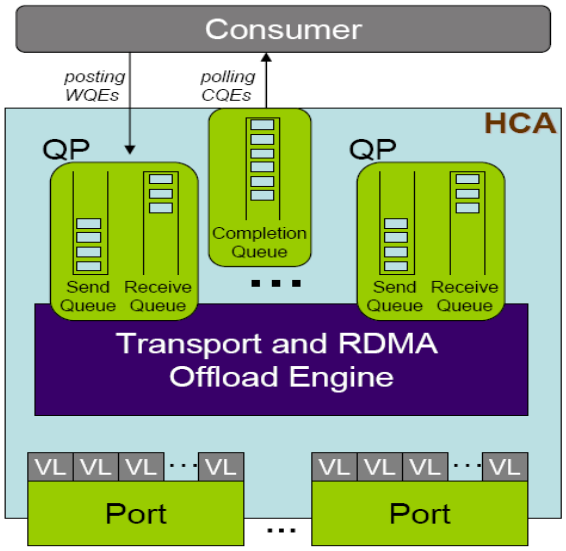


그림 3. Channel Adapter 개념도

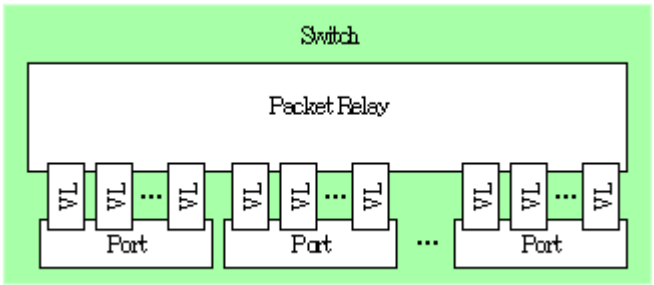


그림 4. 스위치 개념도

스위치는 서브넷을 구성하여 경로를 제공하는 장치로서 경로 설정 정보에 따라 패킷을 전달하는 기능을 수행하며, 패킷을 발생시키거나 직접 수신하지 않는다. 스위치는 내부에 패킷 포워딩을 위한 테이블을 구성하고 이를 이용하여 패킷의 목적지 LID에 따라서 패킷 경로 설정

및 전송을 수행한다. 패킷 포워딩 테이블은 서브넷 매니저에 의해서 구성되며 다중 경로가 존재할 경우 서브넷 매니저는 부하 분산 또는 오류에 의한 자동 경로 변경이 가능하도록 패킷 포워딩 테이블의 재설정을 수행한다. 스위치는 위의 그림과 같이 복수 개의 포트를 가지며, 하나의 패킷을 하나의 목적지로 전송하는 유니캐스트 전송은 물론 사용자 구현 선택으로 하나의 패킷을 복수의 목적지로 전송하는 멀티캐스트도 구현할 수 있다. 라우터는 서브넷과 서브넷을 연결하는 장치로 기능 및 구조는 스위치와 유사하다. 패킷 경로 설정에 있어서 스위치와 달리 GID(Global Identification)를 사용한다.

IBA는 서브넷 매니저(Subnet Manager: SM)와 제너럴 서비스(General Service)로 구성되는 관리 형상을 제공하며, 각 매니저는 해당 에이전트와 MAD(Management Datagram) 패킷을 사용하여 연결망 관리 동작을 수행한다. 서브넷 매니저는 스위치, 라우터, 채널 어댑터 등 IBA 구성 요소의 형상 관리를 수행하며, IBA 연결망 상의 모든 노드는 SM과의 통신을 위해서 SMA를 탑재하고 있어야 한다. 예를 들어 SM은 서브넷 토폴로지를 검출하고, 어댑터 포트에 LID와 GID 등의 정보를 할당하고, 스위치에 LID 할당하고 패킷 포워드 테이블을 구성하는 등의 기능을 수행한다.

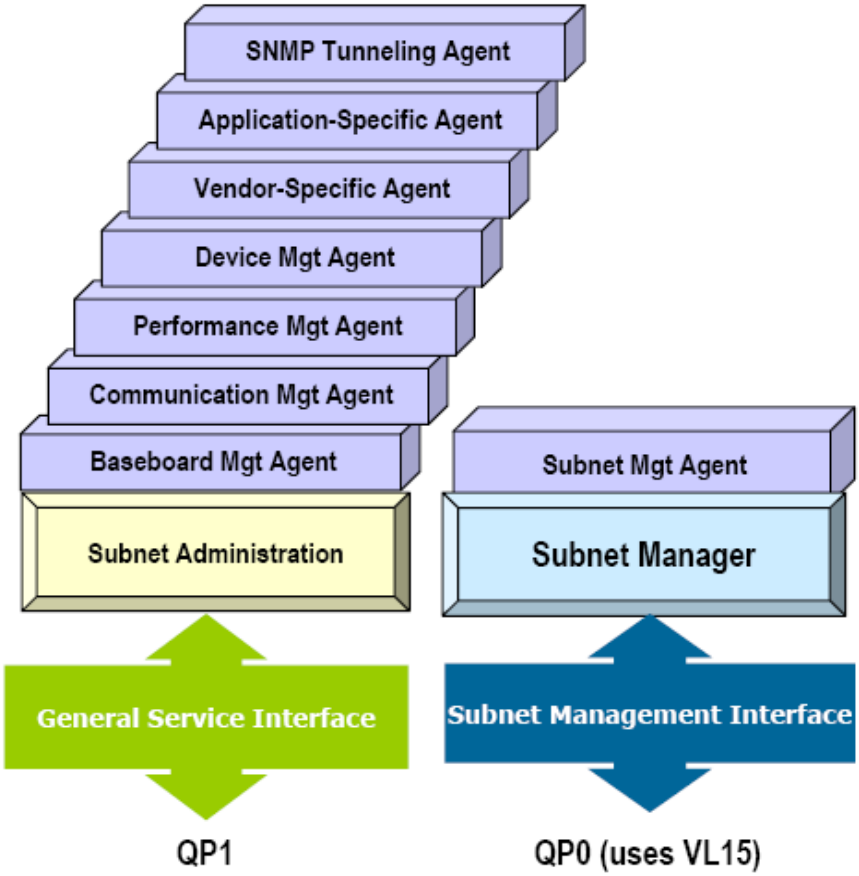


그림 5. InfiniBand Management Model

모든 종단 노드는 SMA를 제공해야 한다. SMA는 SM이 SMI(Subnet Management Interface)를 통해서 접근하는 기능 모듈이다. SMI는 LID를 통한 경로 설정 방법과 직접 경로 설정 방법을 지원하며, 직접 경로 설정 방식은 스위치나 종단 노드가 초기화 되기 전에 MAD 패킷을 전송할 수 있게 한다. 종단 노드는 SMA 외에 부가적인 관리 형상 지원을 위하여 GSA(General Service Agent)를 가질 수 있으며, LID를 통한 경로 설정 방법을 사용하는 GSI(General Service Interface)를 통해서 통신한다.

2.3 통신 방법

IBA 연결망에 연결된 노드는 아래 그림과 같은 방식에 의해서 통신하게 된다. 사용자 프로그램은 메시지 패싱을 위한 메시지 전송 요구와 전송 종료 정보 취득을 Verb를 통해서 수행한다. Verb를 통해서 생성된 메시지 전송 요구는 호스트 메모리에 위치한 작업 큐(송신큐와 수신큐로 구성되는queue pair)에 FIFO(First-in First-out) 방식으로 저장된다. HCA 또는 TCA는 작업 큐의 내용을 검색하여 해당 메시지를 호스트 메모리에서 읽어내어 패킷으로 변환 후에 IBA 연결망으로 전송하게 된다. 전송된 패킷은 목적지 노드에서 다시 메시지로 조립되어 작업 큐에 저장된다. 노드의 전송 요구가 작업 큐에 저장될 때는 해당 전송 요구의 전송 상태 정보를 관리하는 완료 큐를 하나 할당 받게 되며, 전송이 종료되면 해당 완료 큐의 내용에 전송 완료 정보를 기록한다. 노드는 전송 요구 후 전송 완료 여부를 완료 큐의 상태를 확인함으로써 전송의 완료 여부 및 상태를 알 수 있다.

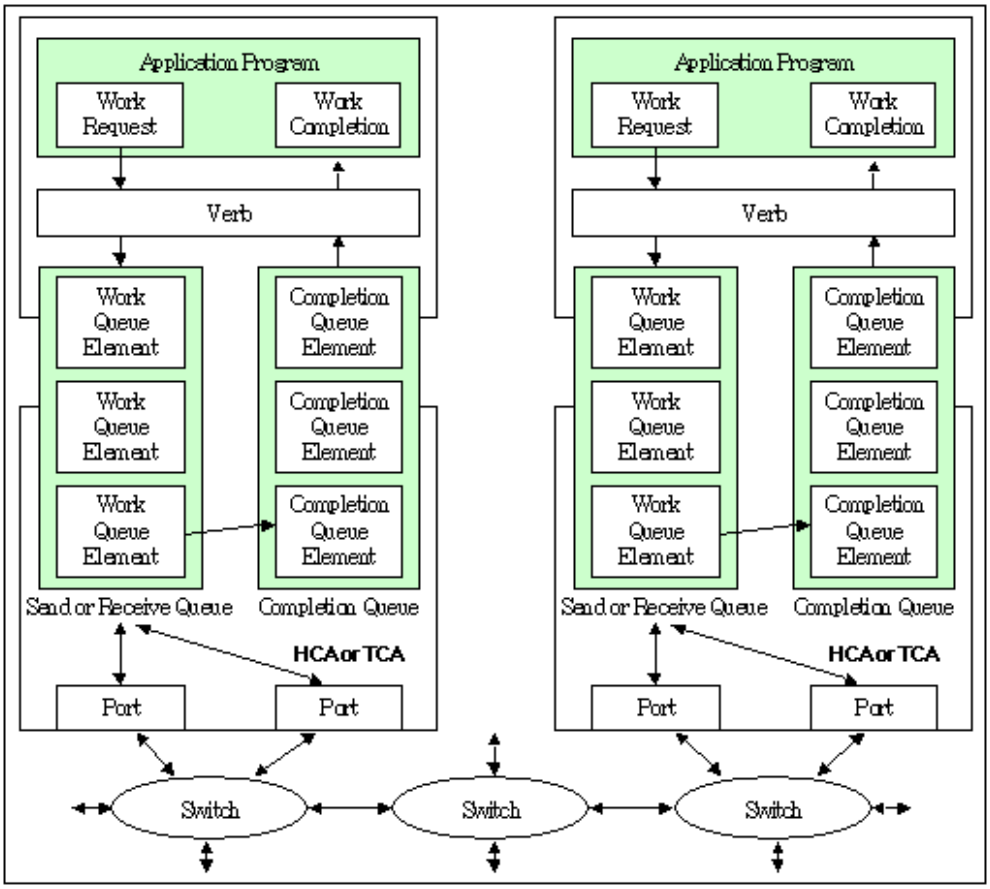


그림 6. InfiniBand 통신 방법

각 노드는 메시지 전송을 위하여 사전에 호스트 메모리에 메시지 전송을 위해서 사용할 영역을 가상주소를 사용하여 지정해야 한다. 사용자 프로그램은 메시지 전송 요구 및 메시지 데이터를 지정된 영역에 위치시키게 되고, HCA 또는 TCA는 운영체제의 간섭없이 지정된 영역에서 메시지 전송 요구와 해당 데이터를 가상 주소를 사용하여 읽어내어 IBA연결망으로 전송한다. 따라서 메시지 전송에 있어서 통신 계층을 통과하며 발생하는 데이터 복제 현상을 제거하여 메시지 전송 처리 속도를 향상시키고 있으며, 데이터 복제 시에 발생하는 프로세서의 오버헤드도 제거한다.

IBA 연결망을 통한 통신은 송신/수신 프리미티브(send/receive primitive)를 사용하는 메시지 패싱 기법 이외에 RDMA(Remote Direct Memory Access) 전송을 제공하여 대용량 데이터의 전송을 지원하고 있으며, 응용 프로그램의 특성에 따라서 메시지의 신뢰도와 대역폭을 조절할 수 있도록 다양한 형태의 전송 서비스를 제공한다.

제 3장 InfiniBand 프로토콜 계층

InfiniBand architecture는 아래 그림과 같이 다중 계층으로 나누어지며 각 계층간은 독립적으로 동작한다. 본 장에서는 인피니밴드를 Physical, Link, Network, Transport, Upper Layer로 나누어 설명하고자 한다.

IBA 연결망을 구성하는 HCA, TCA 및 스위치와 라우터는 아래 그림과 같은 통신 프로토콜 계층을 처리한다. HCA와 TCA는 Verb를 통하여 전달되는 메시지를 IBA 연결망에 패킷 단위로 전송해야 하므로, 메시지 전송 요구를 입력으로 받아서 IBA 패킷으로 변환하여 IBA 연결망으로 전기적 신호를 전송하는 기능을 수행해야 한다. 따라서 HCA와 TCA는 트랜스포트, 네트워크, 링크 및 물리 계층을 모두 가져야 한다. 스위치와 라우터는 InfiniBand 연결망에서 패킷의 경로 설정 및 전송을 담당하며, 네트워크, 링크, 물리 계층으로 구성된다.

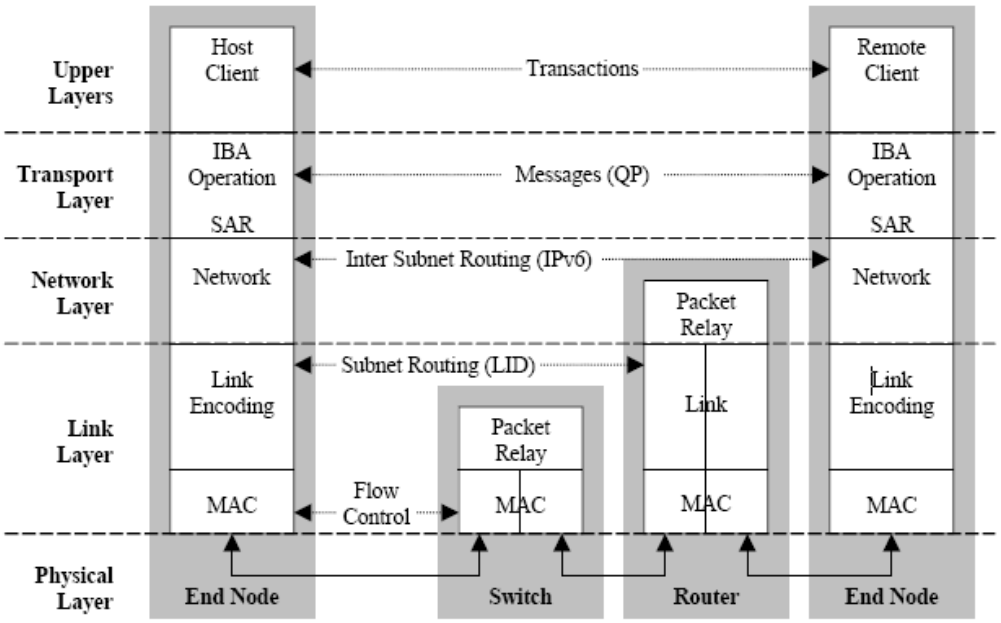


그림 7. InfiniBand Layers

물리 계층은 연결 매체상에 비트로 표현되는 데이터 정보가 심볼로 구성되어 전달되는 계층으로 비트 전송 속도, 전송 매체, 신호 전달 방식 등을 정의한다. 8b/10b 코딩 방식에 의한 직렬 차동 신호 전송 방식을 사용하여

동축 케이블이나 광 케이블 매체를 통해서 2.5Gbps 이상의 전송 속도를 제공하고 있다.

참고로 전자회로 장치에서 반도체 칩(chip)과 반도체 칩 사이의 chip-to-chip 인터페이스는 한 신호선이 연결된 칩 개수에 따라 point-to-point 방식과 multi-drop 방식의 두 가지로 분류되고, 한 번에 전송되는 신호 개수에 따라 직렬(serial) 링크와 병렬(parallel) 링크의 두 가지로 분류된다. 차동 전송 방식은 한 개의 신호를 두 개의 신호선을 사용하여 전송하는 방식으로 두 신호선의 전압이 어느 쪽이 더 큰지를 판별하면 되므로 별도의 기준전압을 필요로 하지 않는다.

사용자의 성능 요구에 따라 포트 당 연결 매체의 개수를 1x, 4x, 12x 중에서 선택할 수 있게 한다. 1x 인피니밴드는 4개의 serial connection로 이루어져 있으며 (2개씩 송, 수신으로 이루어짐) 대역폭(bandwidth)는 2.5Gb/s이고, 이중 20%는 인코딩에 사용되기 때문에 실제 대역폭은 2.0Gb/s이다. 양방향 링크(bi-directional)이기 때문에 한 버스의 총 대역폭(aggregate bandwidth)는 4.0Gb/s이다. full duplex connection 2.5Gb/s를 지원한다.

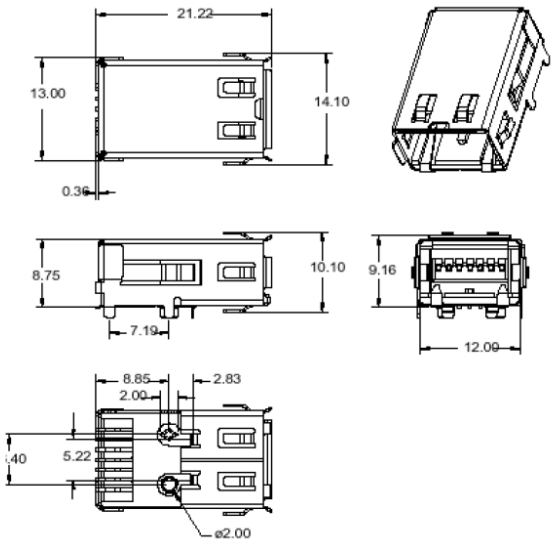


그림 8. InfiniBand Architecture Specification v1.o, Sample Connector

– Mechanical Characteristics

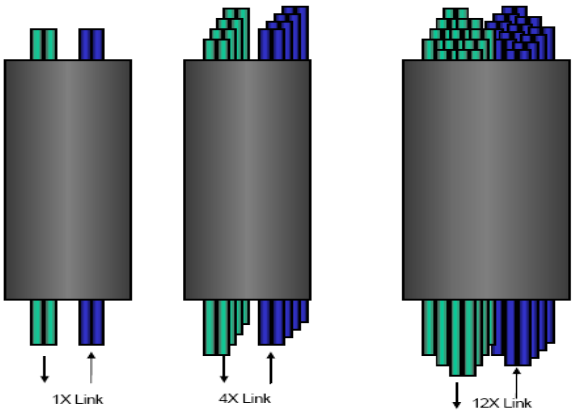


그림 9. InfiniBand Physical Link

Cable은 크게 구리 케이블(Copper Cables)과, 광 케이블(Fiber Optics)이 있으며, 다음 표와 같이 케이블 종류에 따라 전송 길이 제한이 있다.

그림 10. InfiniBand Cable Type

표 1. Copper Cables

Width	Speed	Connector	Min Reach	Type / Power**
4X	SDR/ DDR	Micro-GiGaCN	20m/ 10m	Passive
4X	DDR	Micro-GiGaCN	15-25m	Active 0.5-0.75W
12X	SDR/ DDR	24pin Micro-GiGaCN	20m/ 10m	Passive

표 2. Fiber Optics

Width	Speed	Connector	Type	Min Reach	Power ***	Fiber Media
4X	SDR/ DDR	Micro-GiGaCN	Media Converter	300m/ 150m	0.8-1W	12 strand MPO
4X	DDR	Micro-GiGaCN	Optical Cable	100m	1W	12 strand attached

표 3. InfiniBand Link Rates

InfiniBand Link Width	Signal Count	Signalling Rate	Data Rate	Fully Duplexed Data Rate
1x	4	2.5Gb/s	2.0Gb/s	4.0Gb/s
4x	16	10Gb/s	8.0Gb/s	16.0Gb/s
8x	32	20Gb/s	16.0Gb/s	32.0Gb/s
12x	48	30Gb/s	24.0Gb/s	48.0Gb/s

표 4. Link Speed(Gb/s)

Link Speed / LinkWidth	SDR (2.5GHz)	DDR (5GHz)	QDR (10GHz)
1x	2.5	5	10
4x	10	20	40
8x	20	40	80
12x	30	60	120

링크 계층은 패킷 포맷과 패킷 단위 흐름제어, 서브넷 내부에서의 패킷 경로 설정을 포함하는 프로토콜을 정의한다. 패킷은 링크 관리(management) 패킷과 데이터(data) 패킷으로 구분된다. 링크 관리 패킷은 포트와 포트간의 점대점 연결로 구성된 링크상에서 전송 속도, 전송 데이터 폭 등을 상호 협상하는 링크 트레이닝 동작에 사용된다. 또한 링크상에서 흐름제어 정보를 전달하고 링크의 무결성을 유지 관리한다. 데이터 패킷은 IBA 연결망을 통해서 전송 서비스 동작을 수행하기 위해서 사용하는 패킷으로 전송 서비스

종류에 따라서 다양한 헤더를 가질 수 있다. 서브넷 내부에서의 경로 설정을 위해서 기본적으로 LRH(Local Routing Header)를 가져야 한다. 또한, 오류 방지를 위해서 ICRC(Invariant CRC)와 VCRC(Variant CRC)를 사용하여 데이터 페이로드를 포함한 패킷 전체를 보호한다. 데이터 패킷은 transport payload에서 4K byte까지 보낼 수 있다.

● 패킷(Packets)

링크 계층에서는 링크 관리(management) 패킷과 데이터(data) 패킷으로 구분된다. Management 패킷은 link의 유지와 관리를 위해 사용된다. Virtual Lane의 device 정보는 management 패킷에 포함된다. 데이터 패킷은 transport payload에 4K까지 보낼 수 있다.

● 스위칭(Switching)

링크 계층에서 서브넷(subnet) 안에서 패킷 forwarding과 switching을 처리한다. 서브넷에서 모든 디바이스에 Subnet Manager로부터 16bit Local ID(LID)를 받는다.

Addressing을 위해 Subnet 안에는 LID가 사용된다. 링크 계층 스위칭(Link Level Switching)은 패킷 안에 Local Route Header(LRH)에 있는 목적지(Destination) LID에 의해 정해진 디바이스로 보내어진다. 이 LRH는 모든 패킷에 존재한다.

● QoS

QoS는 Virtual Lanes(VL)을 통해 제공된다. 이 VL들은 하나의 물리적 Link가 논리적으로 통신 링크(logical communication links)로 분리된다. 각 링크는 1개의 관리 lane과 15개까지의 VL을 가질 수 있다. VL15는 높은 우선순위 VL0는 낮은 우선순위를 가진다. 관리 패킷은 독점적으로 VL15를 사용한다. 각 디바이스는 최소한 VL0와 VL15를 제공해야 하며, 다른 VL은 옵션이다. 각 스위치와 라우터는 SL와 VL의 매핑 테이블을 가지며, 그 매핑 테이블은 서브넷 매니저(subnet manager)에 의해 각 링크에서 제공할 VL의 수와 적절한 우선순위를 지키기 위해 제공된다.

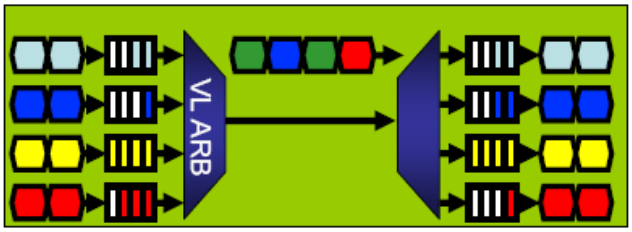
● Credit Based Flow Control

흐름 제어(Flow Control)는 두 포트간의 점대점 연결(point-to-point links) 사이의 data flow를 관리하는데 쓰인다. flow control은 각 VL이 같은 물리적 매체(media)를 사용하여 분리된 virtual fabrics처럼 통신을 유지하는 것을 가능하도록 해 준다. 링크 계층은 각 링크의 수신이 데이터의 손실

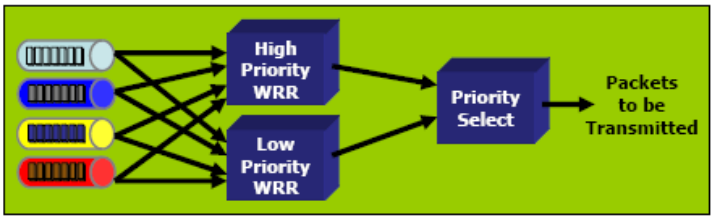
(loss) 없이 정해진 량의 데이터가 송신되었다는 신뢰를 제공한다. 각 디바이스 사이의 credit passing을 관리하기 위해 수신자가 수락할 수 있는 데이터 패킷의 수를 업데이트 한다. 만약 수신자가 가능한 수신 버퍼 공간(receive buffer space)을 알려서 신뢰(credit)를 제공하지 않으며, 데이터는 전송될 수 없다.

● Data integrity

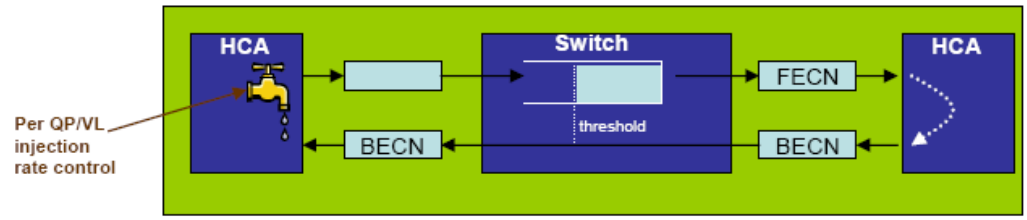
링크 계층에는 패킷마다 데이터의 무결성(integrity)을 확보하기 위해 Variant CRC(VCRC)와 Invariant CRC(ICRC) 2개의 CRC가 있다. 16bit VCRC는 패킷의 모든 필드를 포함하고 각 홉(hop)마다 제 계산한다. 32bit ICRC는 hop에서 hop으로 수정되지 않는 부분만을 포함한다. VCRC는 두 hops 사이의 데이터 무결성을 링크 계층에서 제공하고, ICRC는 end-to-end 데이터 무결성을 제공한다. 이더넷과 같이 오직 하나의 CRC를 정의한 프로토콜에서는, 오류는 디바이스에서 CRC를 재 계산할 때 발견할 수 있다. 비록 데이터가 오류가 있더라도 다음 hop에서 오류 체크는 타당한 CRC로 드러날 수 있다. InfiniBand는 ICRC를 포함해서 한 bit 오류가 발견되었을 때, 오류가 항상 감지될 것이다.



Independent Virtual Lanes (VLs)



H/L Weighted Round Robin (WRR) VL Arbitration



Efficient FECN/BECN Based Congestion Control

그림 11. Link Layer

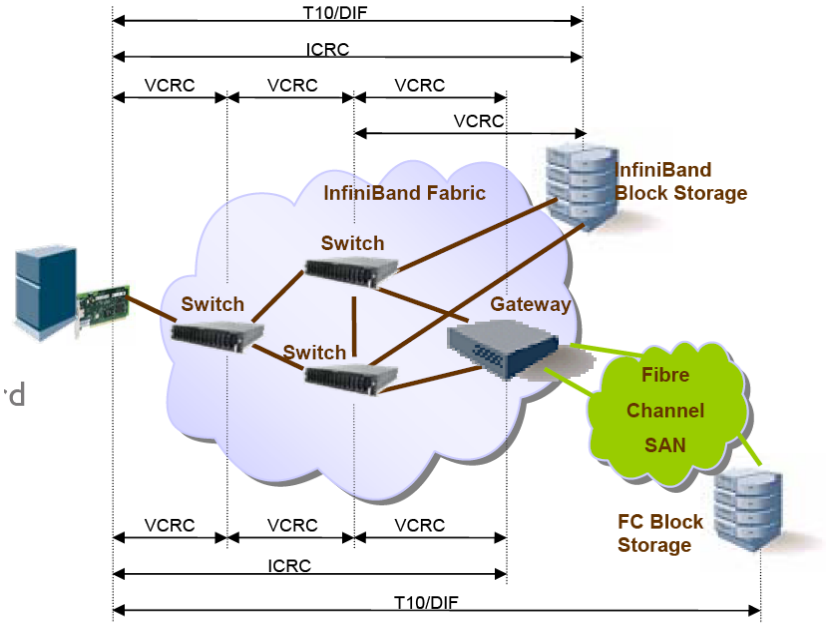


그림 12. InfiniBand Data Integrity

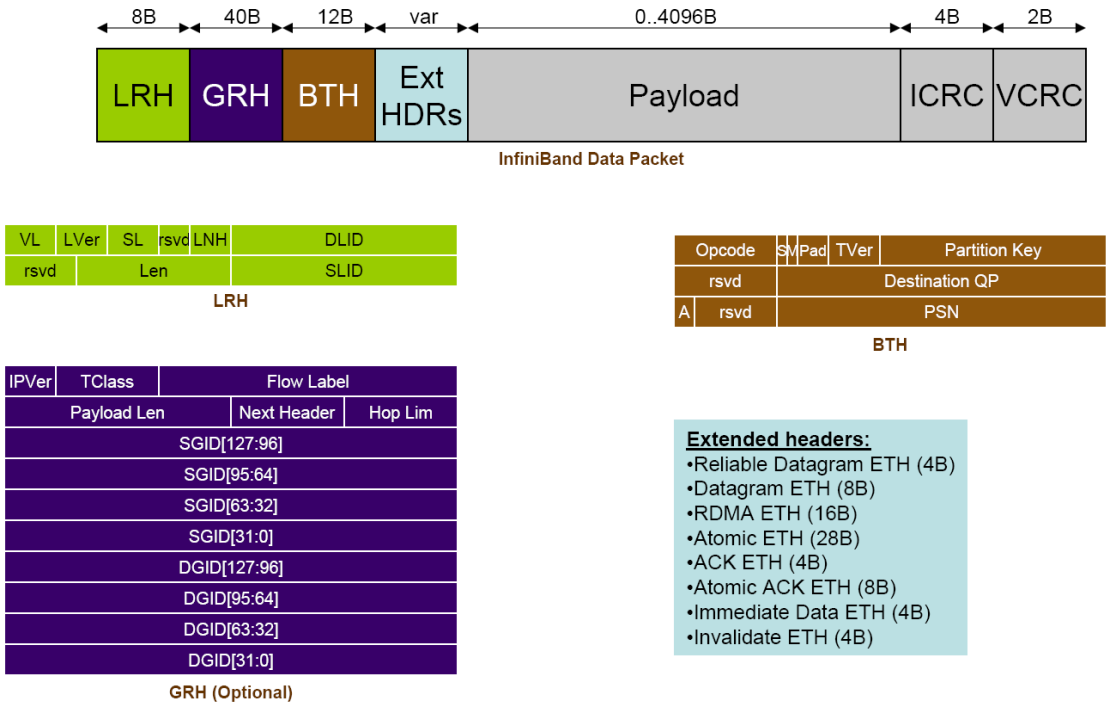


그림 13. InfiniBand Packet Format

네트워크 계층은 서브넷간 패킷 경로 설정 및 패킷 전달을 정의하는 계층으로 GRH(Global Routing Header)를 사용한다. GRH는 IPv6 어드레스 포맷으로 표현되는 GID를 사용하며 GRH의 내용을 사용하여 서브넷간 경로 설정 및 패킷 전달을 수행한다.

트랜스포트 계층은 데이터 전송 서비스의 종류별로 전송 동작을 정의하고, 전송 메시지를 패킷으로 분할 또는 재결합하는 기능을 수행한다. 트랜스포트 계층의 송신과 수신시 데이터를 재결합(reassemblely)할 때 transaction data segmentation를 다룬다. Maximum Transfer Unit(MTU)을 기준으로 패킷을 적절한 크기로 나눈다. 수신자를 Base Transfer Header(BTH)의 목적지 QP(Queue Pair)와 PSN(Packet Sequence Number)를 기반으로 패킷을 재통합한다. 수신자를 acknowledge(패킷을 받았음)를 보내고, 송신자를 acknowledge를 받고 completion queue의 operation 상태를 변경한다. 전송 서비스 종류에 따라서 ETH(Extended Transport Header)를 사용하여 전송 서비스별 상세 정보를 제공한다. IBA는 두드러진 특징은 이러한 트랜스포트 계층의 모든 기능이 하드웨어로 구현되었다는 점이다.

아래 표에는 서비스 신뢰도에 따른 다양한 트랜스포트 서비스를 기술했다.

한 개의 Queue Pair에 한 개의 Transport level이 사용된다.

표 5. Support Services

Class of Service	Description
Reliable Connection	Acknowledged – connection oriented
Reliable Datagram	Acknowledged - multiplexed
Unreliable Connection	Unacknowledged – connection oriented
Unreliable Datagram	Unacknowledged - connectionless
Raw Datagram	Unacknowledged - connectionless

상위 계층은 IBA 전송 서비스를 사용하는 사용자 프로그램과 관리 형상 프로그램을 IBA 연결망 사용자에 해당하며, Verb를 통해서 IBA 연결망을 사용하거나 관리하는 작업을 수행한다.

제 4장 프로토콜 스택

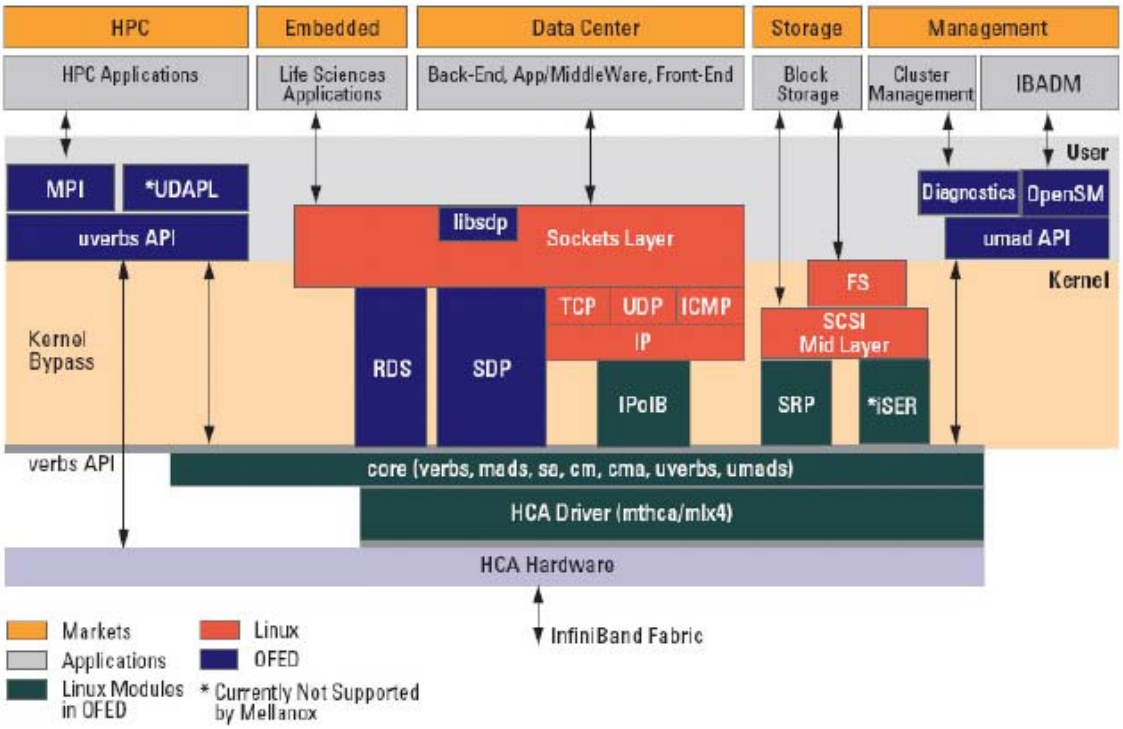


그림 14. Mellanox OFED Stack

그림 14는 Mellanox OFED 스택을 나타낸 것이다. 아래에 각 요소에 대하여 간단히 설명을 기술한다.

- HCA Drivers

  Mellanox는 다음과 같이 두 가지 드라이버를 제공한다.

  ✓ mthca

  Mellanox HCA 디바이스(InfiniHost, InfiniHost III Ex, InfiniHost III Lx)를 위한 low level 드라이버

  ✓ mlx4

  Mellanox ConnectX adapter 기술을 위한 low level 드라이버.

ConnectX는 InfiniBand HCA로서 기능할 수 있음

- Mid-layer Core

  코어 서비스는 management interface (MAD), connection manager (CM) interface, and subnet administrator (SA) interface를 포함한다. 이 스택은 user-mode와 kernel-mode 프로그램들을 위한 요소이다. 코어 서비스는 커널에서 실행되어 verbs와 CM와 관리를 위해 user-mode로 확장된다.

- ULPs

  ✓ IPoIB

  IP over IB(IPoIB) 드라이버는 인피니밴드 위에 구현된 네트워크 interface로서, IPoIB는 IP 패킷을 IB 프로토콜로 encapsulate한다. IP datagram을 헤더를 덧붙여 IPoIB로 변환하여, 인피니밴드 전송(Transport) 서비스를 적용할 수 있다. 기본적으로 Transport 서비스는 표5에서 보인 Reliable Connected(RC) 서비스가 제공되고 configuration을 통해 Unreliable Datagram(UD)를 제공할 수 있다.

  ✓ RDS

  Reliable Datagram Sockets은 RC나 TCP/IP 위의 소켓 사이에서 순차적인 datagram 전송으로 reliable(신뢰성)을 제공하는 소켓 API이다.

  ✓ SDP

  Sockets Direct Protocol은 TCP stream 서비스를 제공하는 byte-steam transport protocol이다. SDP는 IinfinBand의 진보된 offload 기술의 프로토콜을 이용하다. offload 기술 때문에 SDP는 기존의 TCP 구현과 비교했을 때, 낮은 CPU 사용율과 메모리 사용율을 가진다.

  ✓ SRP

  SCSI RDMA 프로토콜은 offload 프로토콜과 infiniBand 구조에서

제공되는 RDMA의 장점을 포함하도록 디자인되었다. SRP initiator라고 알려진 SRP 드라이버는 기존의 리눅스에서 low-level SCSI와는 다르다. SRP initiator는 로컬 HBA를 제어하지 않는다. SRP initiator는 SRP Target이라는 I/O controller와의 connection을 제어한다. SRP Target은 infiniband fabric에 연결된 원격 스토리지 디바이스에 access를 제공하다. SRP Target은 하나의 I/O unit 안에 존재하며, 스트리지 서비스를 제공한다.

- MPI

  Message Passing Interface(MPI)는 병렬 컴퓨터들, 클러스터, heterogeneous networks를 이용하기 위한 병렬 소프트웨어 라이브러리이다. 멜라녹스 OFED는 infiniband 위에서 다음과 같은 두 가지 MPI 구현을 제공한다.

  ✓ Open MPI – Open MPI 프로젝트에 의해 구현된 오픈 소스 MPI-2

  ✓ OUS MVAPICH – Ohio State University에서 구현된 MPI-1

- InfiniBand Subnet Manager

  모든 infiniband 관련된 Upper Layer Protocols (ULPs)은 infiniBand fabric 위에 Subnet Manager(SM)의 적절한 동작이 요구된다. 하나의 SM은I infiniband 스위치나 하나의 노드에 실행되어야 한다. OpenSM은 하나의 Infiniband-compliant Subnet Manager이며 Mellanox OFED의 한 부분으로 설치된다.

- Diagnostic Utilities

  네트워크과 데이터 센터 관리자를 위해 두 가지 진단 툴을 제공한다.

  ✓ ibutils – Mellanox Technologies diagnostic utilities

  ✓ infiniband-diags – OpenFabrics Alliance InfiniBand diagnostic tools

제 5장 InfiniBand 성능 측정

```
[root@bambi03 log]# ibv_devinfo

hca_id: mthca0

        fw_ver:                         4.6.0

        node_guid:                      0005:ad00:0005:14b4

        sys_image_guid:                 0005:ad00:0005:14b7

        vendor_id:                      0x05ad

        vendor_part_id:                 25208

        hw_ver:                         0xA0

        phys_port_cnt:                  2

                port:   1

                        state:                  PORT_ACTIVE (4)

                        max_mtu:                2048 (4)

                        active_mtu:             2048 (4)

                        sm_lid:                 8

                        port_lid:               8

                        port_lmc:               0x00


                port:   2

                        state:                  PORT_DOWN (1)

                        max_mtu:                2048 (4)

                        active_mtu:             512 (2)
```

```
                    sm_lid:                 0
                    port_lid:               0
                    port_lmc:               0x00
```

```
[root@bambi03 log]# ibstat
CA 'mthca0'
        CA type: MT25208 (MT23108 compat mode)
        Number of ports: 2
        Firmware version: 4.6.0
        Hardware version: a0
        Node GUID: 0x0005ad00000514b4
        System image GUID: 0x0005ad00000514b7
        Port 1:
                State: Active
                Physical state: LinkUp
                Rate: 10
                Base lid: 8
                LMC: 0
                SM lid: 8
                Capability mask: 0x00510a6a
                Port GUID: 0x0005ad00000514b5
        Port 2:
                State: Down
```

```
Physical state: Polling

Rate: 10

Base lid: 0

LMC: 0

SM lid: 0

Capability mask: 0x00510a68

Port GUID: 0x0005ad00000514b6
```

■ Netperf 설치

상기의 site에서 다운로드 받은 파일(netperf-2.4.1.tar.gz)을 압축을 해제하고, 설치 및 컴파일하는 과정은 아래와 같으며, root 사용자로 로그인하여 작업을 수행하였다.

```
# gunzip netperf_2.4.1.tar.gz

# tar xvf netperf_2.4.1.tar.gz

# cd netperf_2.4.1

# ./configure

# make

# make install
```

■ 설치 확인

상기와같이 압축을 해제하고 컴파일을 수행 후 아래와 같이 실행 파일들이

생성되었는지 확인한다.

cd /usr/local/bin

ls -l net*

-rwxr-xr-x 1 root root 153986 3월 3 17:23 netperf

-rwxr-xr-x 1 root root 159404 3월 3 17:23 netserver

■ 환경 설정

Netperf를 정상적으로 설치후 시스템내에 Netperf에서 사용하는 TCP 12865 서비스 포트를 추가하는 과정이 필요하다.

vi /etc/services

netperf 12865/tcp ----> 추가

vi /etc/xinetd.conf ----> 아래 서비스 그대로 추가

```
service netperf
{
socket_type          = stream
protocol     = tcp
wait                    = no
user                    = root
server                  = /usr/local/bin/netserver
server_arg = netserver
}
```

■ 실행 확인

가. Server에서 netserver 실행

/usr/local/bin/nerserver

Starting netserver at port 12865
Starting netserver at hostname 0.0.0.0 port 12865 and family AF_UNSPEC

나. Client에서 netperf 실행

/usr/local/bin/netperf -H hostname (netserver가 동작중인 Hostname)
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to port 0 AF_INET
Recv Send Send
Socket Socket Message Elapsed
Size Size Size Time Throughput
bytes bytes bytes secs. 10^6bits/sec

87380 16384 16384 10.01 77.51

상기와 같은 결과값이 나오면 정상적으로 실행이 되는 것이다.

[root@tom4 netperf-2.4.1]# netperf -H 100.0.0.7

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 100.0.0.7 (100.0.0.7) port 0 AF_INET

Recv Send Send

Socket Socket Message Elapsed

Size Size Size Time Throughput

bytes bytes bytes secs. 10^6bits/sec

87380 16384 16384 10.02 941.19

[root@tom4 netperf-2.4.1]# netperf -H 110.0.0.7

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.7 (110.0.0.7) port 0 AF_INET

Recv Send Send

```
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.00     7828.54
[root@tom4 netperf-2.4.1]# netperf -H 110.0.0.7
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.7 (110.0.0.7) port 0 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec
 87380  16384  16384    10.01     7828.43
```

```
[root@tom7 ~]# LD_PRELOAD=/usr/lib64/libsdp.so LIBSDP_CONFIG_FILE=/etc/libsdp.conf netperf -H 110.0.0.4 -c -C -- -m 65536
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.4 (110.0.0.4) port 0 AF_INET
Recv   Send    Send                          Utilization       Service Demand
Socket Socket  Message  Elapsed              Send     Recv     Send     Recv
Size   Size    Size     Time     Throughput  local    remote   local    remote
bytes  bytes   bytes    secs.    10^6bits/s  % S      % S      us/KB    us/KB

 87380  16384  65536    10.00      5364.59   12.68    13.51    1.549    1.650
```

```
[root@tom7 ~]# LD_PRELOAD=/usr/lib64/libsdp.so LIBSDP_CONFIG_FILE=/etc/libsdp.conf netperf -H 110.0.0.4 -c -C -- -m 65536

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.4 (110.0.0.4) port 0 AF_INET

Recv   Send    Send                          Utilization       Service Demand

Socket Socket  Message  Elapsed              Send     Recv     Send    Recv

Size   Size    Size     Time     Throughput  local    remote   local   remote

bytes  bytes   bytes    secs.    10^6bits/s  % S      % S      us/KB   us/KB


 87380  16384  65536    10.00      5443.94   12.76    13.13    1.536   1.581

[root@tom7 ~]# netperf -H 110.0.0.4 -c -C -- -m 65536

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.4 (110.0.0.4) port 0 AF_INET

Recv   Send    Send                          Utilization       Service Demand

Socket Socket  Message  Elapsed              Send     Recv     Send    Recv

Size   Size    Size     Time     Throughput  local    remote   local   remote

bytes  bytes   bytes    secs.    10^6bits/s  % S      % S      us/KB   us/KB


 87380  16384  65536    10.01      7826.17   10.66    13.43    0.893   1.124
```

```
[root@tom7 ~]# LD_PRELOAD=/usr/lib64/libsdp.so LIBSDP_CONFIG_FILE=/etc/libsdp.conf netperf -H 110.0.0.4 -c -C -- -m 65536

[root@tom7 ~]# netperf -H 110.0.0.4 -c -C -- -m 65536

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.4 (110.0.0.4) port
```

```
0 AF_INET

Recv    Send    Send                          Utilization       Service Demand

Socket Socket  Message  Elapsed              Send     Recv     Send     Recv

Size    Size    Size     Time     Throughput  local    remote   local    remote

bytes  bytes   bytes    secs.    10^6bits/s  % S      % S      us/KB    us/KB


 87380  16384  65536    10.00      3919.21   14.80    18.81    2.474    3.146

[root@tom7 ~]# netperf -H 110.0.0.4 -c -C -- -m 65536

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.4 (110.0.0.4) port
0 AF_INET

Recv    Send    Send                          Utilization       Service Demand

Socket Socket  Message  Elapsed              Send     Recv     Send     Recv

Size    Size    Size     Time     Throughput  local    remote   local    remote

bytes  bytes   bytes    secs.    10^6bits/s  % S      % S      us/KB    us/KB


 87380  16384  65536    10.00      3769.48   15.09    18.38    2.623    3.195

[root@tom7 ~]# service openibd restart


[root@tom7 ~]# netperf -H 110.0.0.4 -c -C -- -m 65536

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 110.0.0.4 (110.0.0.4) port
0 AF_INET

Recv    Send    Send                          Utilization       Service Demand

Socket Socket  Message  Elapsed              Send     Recv     Send     Recv

Size    Size    Size     Time     Throughput  local    remote   local    remote
```

```
bytes  bytes   bytes     secs.    10^6bits/s  % S     % S     us/KB   us/KB

 87380  16384  65536     10.01       7825.69   11.39   14.70   0.954
```

```
[root@tom7 ~]# vi /etc/infiniband/openib.conf

# Start HCA driver upon boot

ONBOOT=yes


# Load UCM module

UCM_LOAD=no


# Load RDMA_CM module

RDMA_CM_LOAD=yes


# Load RDMA_UCM module

RDMA_UCM_LOAD=yes


# Increase ib_mad thread priority

RENICE_IB_MAD=no


# Load MTHCA

MTHCA_LOAD=yes
```

```
# Load IPATH

IPATH_LOAD=yes



# Load MLX4 modules

MLX4_LOAD=yes



# Load CXGB3 modules

CXGB3_LOAD=yes



# Load IPoIB

IPOIB_LOAD=yes



# Enable IPoIB Connected Mode

SET_IPOIB_CM=yes

#SET_IPOIB_CM=no



# Load SDP module

SDP_LOAD=yes



# Load SRP module

SRP_LOAD=no



# Load SRP Target module
```

```
SRPT_LOAD=no



# Enable SRP High Availability daemon

SRPHA_ENABLE=no

SRP_DAEMON_ENABLE=no
```

성능 결과

Protocol	Network Bandwidth (16MB)	Network Bandwidth (32MB)	비고
IPoIB (SET_IPOIB_CM=no)	3769.48 Mbps (471.185MB)	4077.43 Mbps	IPoIB non CM 의 성능
IPoIB (SET_IPOIB_CM=yes)	7825.69 Mbps (978.21MB)		OFED 1.1->1.2 IPoIB Connection Mode 가 생기면서 성능 향상 1.2 부터는 default 로 IPoIB CM 이 사용됨
SDP	5443.94 Mbps (680.49MB)	7,320.04 Mbps	

제 5장 참고자료

[1] InfiniBand Trade Association(IBTA) Official Homepage, http://www.infinibandta.org, Mar., 2008.

[2] D.Pendery and J. Eunice, “InfiniBand Architecture : Bridge over Troubled Waters”, Research Note, http://www.infinibandta.org/ newsroom/whitepapers/illuminata.pdf

[3] “Introduction to InfiniBand”, whiter paper, http://www.infinibandta.org/newsroom/whitepapers/intro_to_infiniband_1207.pdf, 2001.

[4] W.T. Futral, InfiniBand Architecture Development and Deployment : A Strategic Guide to Server I/O Solution, Intel Press, 2001.

[5] S. Moh, Y, Y, Kim, S.-H. Yoon, M.-J. Kim and K.-W. Rim, “InfininBand Technology : The Next-Generation System Interconnect,” Proc. of the 1st World Korea Business Convention, pp.C.2-8, Oct., 2002.

[6] 모상만, 박경, 김성남, 김명준, 임기욱, “고성능 클러스터 시스템을 위한 인피니밴드 시스템 연결만의 설계 및 구현”, 한국정보처리학회 논문지, 2003.

[16] InfiniBand Architectural Technology, Technical White Paper from Compaq, ftp://ftp.compaq.com/ pub/supportinformation/papers/tc000702tb.pdf, June 2000.

[19] InfiniBand Technology Prototypes White Paper Spring 2000, Technical White Paper from Intel, ftp://download.intel.com/design/servers/ future_server_io/documents/Final_Whitepaperxx.pdf, Feb. 2000.

[20] InfiniBand Architecture: Next-Generation Server I/O, Technical White Paper from Dell, http:// www.dell.com/downloads/global/vectors/infiniband.pdf, 2000.

[21] Get on the Fabric: InfiniBand Fabric Prototype Demonstration, Technical White Paper from Intel, ftp://download.intel.com/design/servers /future_server_io/documents/get_on_fabric.pdf, Aug. 2000.

[22] To InfiniBand and Beyond, The Presentation from IBM, http://www.chips.ibm.com/products/ infiniband/presentations/To_InfiniBand_and_Beyond.pdf, 2000.
[23] T. Shanley, InfiniBand Network Architecture, Edited by J. Winkles, Addison-Wesley, 2002.