제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

TCP/IP offloading 기술에 관한 연구

김재열, 차규일, 정성인
한국전자통신연구원
e-mail : {gauri, gicha, sijung}@etri.re.kr

A Study on TCP/IP offloading

Chei-Yol Kim, Gyu-il Cha, Sung-In Jung
Electronics and Telecommunications Research Institute

요 약

TCP/IP 의 성능을 향상 시키기 위하여 주로 커널 레벨에 구현되어 있는 TCP/IP 프로토콜 계층을 네트워크 인터페이스 카드로 이전시키는 기술을 TCP/IP offloading 기술이라고 한다. 본 논문은 최근의 TCP/IP offloading 기술의 동향을 살펴보고 또한 TCP/IP offloading 기술을 시스템에 적용할 때 고려해야 할 점등을 기술한다.

1. 서론

이더넷 기술은 현재 일반적인 LAN 환경에서 가장 많이 사용되는 기술이다. 시간이 흐를수록 네트워킹 기술의 발전으로 이더넷 네트웍의 속도 또한 꾸준히 증가하여 왔으며 이는 일반 서버 컴퓨터 프로세서의 능력을 능가하는 10Gbps 까지 이르고 있다. 하지만, 이러한 네트워크의 발전은 서버 컴퓨터가 근본적으로 입출력 능력보다 컴퓨팅 능력을 기본으로 설계되었기 때문에 한계를 가지게 되었다. 보편화된 인터넷의 영향으로 서버 컴퓨터의 병목지점은 컴퓨팅 부분이 아닌 네트워크를 포함한 입출력 부분에서 발생하고 있으며 네트워크에서의 병목현상의 가장 큰 이유는 네트워크의 속도를 따라가지 못하는 TCP/IP 스택의 오버헤드 때문이다. 이와 함께 발생하는 문제점으로는 TCP/IP 스택의 처리를 위하여 호스트의 프로세서를 사용하게 됨으로써 프로세서 상에서 돌아가고 있는 응용프로그램의 호스트 프로세서 점유율을 빼앗아 응용프로그램의 처리능력 감소를 들 수 있다. 네트워크의 속도가 느렸던 과거에는 이러한 문제점을 해결하기 위해서 TCP 프로토콜 자체를 연구하여 오버헤드를 줄이는 노력이 많았다[1]. 하지만 고성능의 네트워크 성능을 요구하는 오늘날에는, 이러한 문제점을 해결하기 위한 방법으로 호스트 프로세서 상에서 처리되는 TCP/IP 스택을 이더넷 네트웍 카드상으로 내려 호스트 프로세서의 부담을 덜어주어 전체적으로 네트워크와 호스트의 성능을 동시에 향상 시킬 수 있다. 이러한 기술을 TCP/IP offloading 이라고 부른다. 그림 1 은 일반적인 네트워크 구조와 TCP/IP offloading 기술을 적용한 경우의 네트워크 구조를 보여준다.

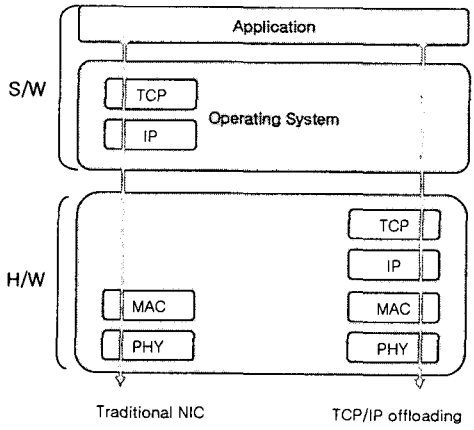

그림 1. TCP/IP offloading 의 적용

2 절에서는 TCP/IP 의 성능을 향상시키기 위한 방법에 대해서 설명하며, 그 중 Zero-copy 기법에 대해서는 3 절에서 자세히 알아볼 것이다. 4 절에서는 네트워크 환경에 따른 TCP/IP offloading 의 고려사항을 설명하고, 5 절에서는 구현방법에 따른 특징을 말하며, TCP/IP offloading 기술의 응용에 대해서는 6 절에서 언

급하고, 7 절에서는 결론을 말한다.

2. TCP/IP 성능 향상

TCP/IP 의 성능을 향상시키기 위해서 많은 연구가 진행되어 왔으며 이를 통해 TCP/IP 처리를 위한 오버헤드를 줄이는 방법으로는 크게 패킷 관점에서의 접근법과 바이트 관점에서의 접근법으로 크게 나눌 수 있다.

2.1. 패킷 관점에서의 TCP/IP 성능 향상

패킷 관점에서 TCP/IP 의 성능을 향상시킬 수 있는 방법은 첫째로 먼저 주고 받는 패킷의 크기를 크게 하여 데이터 전송 횟수를 줄이는 방법이 있다. IEEE802.3 이더넷 표준에서는 MTU(Maximum Transfer Unit)의 크기를 1500 바이트로 하고 있으나 현재는 많은 기가비트 이더넷 제공자들이 Alteon 네트웍스사가 제안한 9000 바이트의 MTU 크기를 지원한다. 다른 한 가지는 패킷을 주고 받을 때 발생하는 인터럽트에 의한 성능 저하를 방지하는 기법이다[2]. 데이터를 보내거나 받을 때 호스트는 인터럽트를 통해 이를 감지한다. 네트워크가 고속화 됨에 따라 인터럽트를 처리하는데 발생하는 오버헤드가 커지면서 네트웍 속도를 높이는데 장애물이 되고 있다. 인터럽트 처리시 부하를 줄이는 방법은 크게 2 가지로 아래와 같다.

- Interrupt Coalescing : 발생하는 인터럽트를 모아서 호스트에 전달
- Interrupt Suppression : 선택적으로 인터럽트를 지연시켜서 호스트에 전달

이상과 같은 패킷 관점에서의 오버헤드를 줄이는 방법은 TCP/IP offloading 기술을 필요로 하지는 않는다.

2.2. 바이트 관점의 TCP/IP 성능 향상

바이트 관점에서의 오버헤드는 2 가지로 나누어 볼 수 있다. 첫째는 호스트의 사용자 메모리영역에 있는 패킷 데이터를 네트워크 인터페이스 카드의 메모리로 이동하는데 드는 오버헤드이다. 사용자 영역의 데이터를 네트워크를 통해서 보내기 위해서는 사용자 영역에서 커널 메모리 영역으로의 복사와 커널 영역에서 네트워크 인터페이스의 메모리로의 복사가 일어난다. 이러한 복사의 회수를 줄이는 방법을 zero-copy 기법이라 한다. 이러한 zero-copy 기법을 사용하는데 있어 커널 영역의 메모리를 건너뛰는 경우에는 커널 영역에 구현된 TCP/IP 계층을 통과할 수 없으므로 이때는 TCP/IP 계층을 네트워크 어댑터로 내리는 TCP/IP offloading 이 반드시 필요하다. 3 절에서는 이러한 zero-copy 의 다양한 기법을 설명할 것이다.

바이트 관점에서의 두 번째 오버헤드는 네트워크를 통해 전달된 데이터의 오류 여부를 판단하기 위해서 사용하는 데이터의 checksum 을 수행하는 데 드는 비용이다. 일반적으로 호스트 프로세서에서 수행하는 checksum 기능을 네트워크 어댑터에서 수행하게 되면 호스트 프로세서의 부담을 줄일 수 있으며 네트워크의 성능도 높일 수 있으므로 TCP/IP offloading 기술의 큰 장점이 된다.

3. Zero-copy 기법

다수의 zero-copy 기법이 연구되어 왔으며[3], 크게 4 가지의 기법으로 분류될 수 있다.

3.1. 사용자가 접근 가능한 네트워크 인터페이스 메모리

이것은 사용자가 네트워크 인터페이스의 메모리에 직접 접근할 수 있도록 해주는 하드웨어의 지원을 받아서 네트워크 인터페이스의 메모리를 사용자 메모리 공간으로 미리 매핑시켜 사용하는 기법이다. 이 기법을 사용하기 위해서는 응용프로그램은 인터페이스의 메모리를 할당하고 사용하는데 있어서 특별한 버퍼 관리기를 사용해야 하며 이로 인해 기존의 응용프로그램에 대해서 호환성을 제공하지 못한다. 또한 이로 인해 인터페이스 카드가 자체적으로 사용하는 메모리의 크기가 줄어 메모리 자원 부족 현상을 보일 수도 있다.

3.2. 커널-네트워크 공유 메모리

위 3.1 에서 언급된 메모리 자원 부족 문제를 완화하기 위한 방법으로 운영체제의 커널이 DMA(Direct Memory Access)나 PIO(Program I/O)를 이용해 인터페이스 카드의 메모리와 응용프로그램의 버퍼 메모리 사이의 데이터를 전송하는 방식이다. 이 기법은 응용프로그램의 소켓 인터페이스를 그대로 사용할 수 있기 때문에 응용프로그램을 고치지 않아도 되는 호환성이 제공되지만, 커널의 네트워크 버퍼관리 부분은 공유메모리 제공을 위해 고쳐져야만 한다.

3.3. 사용자-커널 공유 메모리

먼저 사용자 주소 공간과 커널의 주소 공간에서 같이 접근할 수 있는 공유 메모리를 설정하여 사용자 영역과 커널 영역간의 데이터 복사를 방지하는 방법으로, 응용프로그램에게는 공유메모리를 접근할 수 있는 API(Application Program Interface)를 제공한다. 그리고 공유 메모리와 네트워크 인터페이스간의 데이터 전송은 DMA 를 이용하는 방식이다. 이 방식의 가장 큰 단점은 소켓 형식의 동일한 인터페이스를 제공하지 못하기 때문에 기존 응용프로그램에 대해서 호환성이 없다는 점이다. 또한 3.2 에도 적용되지만, 공유메모리의 특성상 응용프로그램의 에러에 매우 취약한 구조를 가지고 있다.

3.4. 사용자-커널 page remapping

이 기법은 먼저 MMU 테이블을 수정하여 커널 버퍼 메모리를 재매핑 시키고, DMA 를 사용해서 네트워크 인터페이스 메모리와 커널 버퍼 메모리간의 데이터를 전송하는 방식이다. 송신 측에서는 CoW(Copy on Write) 기법을 사용하여 소켓 인터페이스의 복사 형식

을 그대로 만족시킨다. 이 기법을 사용하기 위해서는 네트워킹 코드 중 일부를 수정해야 하지만, 응용프로그램의 인터페이스에 대해서는 호환성을 제공할 수 있다.

4. 사용환경에 따른 고려사항

일반적인 LAN 환경에서는 데이터 전송 중에 소실되는 패킷의 수는 매우 드물며 호스트간의 연결 또한 장시간 이루어진다. 이러한 상황에서는 TCP/IP 의 연결 설정 및 해제 부분과 데이터 전송 중 에러가 발생했을 때의 처리 부분을 호스트에서 처리하고 데이터 전송 부분만을 이더넷 어댑터에서 처리해 주는 국부적 offloading 기법을 사용해도 좋은 성능을 얻을 수 있다. 이러한 예로 현재 저장장치 분야에서 많이 사용되는 SAN(System Area Network)을 대표적인 응용예로 들 수 있을 것이다. 이와는 반대로 전송중의 에러율이 상대적으로 훨씬 크고 호스트간의 연결 시간 또한 상대적으로 짧은 인터넷 환경에서는 데이터 전송 부분만을 이더넷 어댑터에서 처리해 주는 국부적 offloading 기법을 사용해서는 큰 효과를 볼 수 없으며 TCP/IP 프로토콜의 모든 동작을 이더넷 어댑터에서 지원해 주는 전면적 TCP/IP offloading 기술이 유효하다. 그리고 일반적으로 국부적 offloading 기법을 사용하는 경우는 TCP/IP 연결을 제어하는 제어부분은 커널에 남아있으므로 통상의 사용자 인터페이스(소켓 인터페이스)를 비교적 쉽게 제공해 줄 수 있는 반면, 전면적 TCP/IP offloading 기술은 소켓 사용자 인터페이스를 그대로 제공하기 힘들다는 단점이 있다.

5. TCP/IP offloading 구현방법

TCP/IP offloading 기술은 근본적으로 호스트 프로세서에서 수행되는 TCP/IP 프로토콜 처리를 네트워크 어댑터에서 수행함으로써 호스트 프로세서의 부하를 줄여 네트워크 성능과 호스트 프로세서의 효율성을 높이는 것이다. TCP/IP offloading 기술을 구현하는 방법은 크게 네트워크 프로세서 상의 펌웨어에 구현하거나 하드웨어적으로 ASIC 에 구현하는 방법으로 나눌 수 있으며, 이를 혼용하는 경우도 있다. 이전에도 네트워크 프로토콜상의 작업 중 부하가 많은 부분을 네트워크 어댑터에 구현한 경우도 있었다. 이러한 예로 IP 의 checksum 기능을 몇몇의 서버 네트워크 어댑터에서 처리해 주었으나 이더넷 네트워크의 속도가 100Mbps 를 넘어 서면서 IP checksum 이상의 기능을 처리해 줄 필요가 생기게 되었다.

TCP/IP offloading 설계에 있어서 어떤 방법으로 구현하는지에 따라 네트워크 어댑터의 유연성과 성능이 달라지므로 네트워크 어댑터의 필요에 따라 이를 고려하여야 한다. 일반적으로는 TCP/IP offloading 을 ASIC 으로 구현하면 성능은 높일 수 있으나 유연성이 떨어져 사용범위에 제한을 받을 수 있으며 프로세서를 사용한 펌웨어 형식으로 구현하면 용도에 따라 유연하게 사용할 수 있으나 ASIC 에 구현하는 방식에 비해 성능이 떨어지고 비용이 올라가는 단점이 있다. 각 구현에 따른 단점을 극복하기 위하여 현재는 성능에 큰 영향을 미치는 부분은 ASIC 으로 구현하고 나머지 부분은 다수의 프로세싱 엔진을 사용하여 유연성과 성능을 모두 추구하는 방법 또한 사용되고 있다.

5.1. ASIC 기반 TCP/IP offloading

ASIC 기반이라 함은 일반적으로 하나의 반도체 칩 위에 구현하는 것을 말한다. 이와 반대로 펌웨어 기반이라 함은 네트워크 프로세서나 일반 범용 프로세서를 장착한 프로그래밍 가능한 네트워크 카드를 이용하는 것을 말한다. ASIC 기반의 TCP offloading 기술의 특징을 정리하면 아래와 같다.

① 필요한 내용만을 구현
② 외부 버스를 통한 입출력 작업 최소화
③ 병렬처리
④ 다수의 하드웨어 타이머
⑤ CAM(Content Addressable Memory)의 효율적 적용
⑥ 낮은 비용(cost)

①은 펌웨어 기반 구현에 비하여 꼭 필요한 기능만을 ASIC 에 구현할 수 있다는 것을 말한다.

②는 펌웨어 기반위에 TCP/IP offloading 을 구현하게 되면 보드상에 있는 메모리를 통해 작업을 수행하게 되며 이때는 보드의 버스 속도에 영향을 받아 성능이 떨어지지만, ASIC 기반에는 ASIC 칩속에 메모리를 내장하여 외부 버스를 이용할 필요가 없어진다.

③ TCP/IP 연결은 하나의 호스트에서 여러 개가 동시에 연결 될 수 있으며 또한 동시에 처리되어야 한다. 이러한 처리를 ASIC 을 이용하면 ASIC 내의 처리장치(CPU)의 수를 필요에 맞게 설계해 병렬로 처리할 수 있으므로 더 좋은 성능을 나타낼 수 있다..

TCP/IP 연결을 유지하기 위해서는 다수의 타이머가 필요하다. 이러한 타이머를 소프트웨어가 아닌 하드웨어로 각각 구성하게 되면 타이머 운영에 따르는 오버헤드를 줄여 TCP/IP 성능을 향상 시킬 수 있다. ④는 이를 말한다.

하나의 TCP/IP 연결 마다 각 연결설정에 관한 데이터 구조체가 할당된다. 서버 컴퓨터의 경우 여러 클라이언트의 요청을 받으면 아주 많은 수의 연결들을 유지하고 있어야 한다. 그러면 각 연결마다 유지해야 하는 데이터의 양도 많아지며 각 데이터를 접근하는데 따르는 부하도 커지게 된다. 대체로 각 연결 데이터를 유지하는 데는 테이블을 사용하게 되는데 테이블 접근시 걸리는 비용을 줄이는 방법의 하나로 CAM(Content Addressable Memory)을 사용한다. CAM 은 테이블을 사용하지 않고 내용기반의 인덱싱 기법을 사용하여 한번의 메모리 접근으로 데이터를 얻을 수 있도록 보장해 준다. 물론 이 기법은 ASIC 수준에서 구현할 수도 있고 펌웨어 수준에서 구현할 수도 있다. 하지만, 기술의 특성상 ASIC 수준에서 구현하는 것이

일반적으로 효과적이며 ⑤는 이와 같은 내용을 말한다.

⑥에서는 ASIC 은 대량생산시 아주 작은 비용으로 생산할 수 있으며 이는 펌웨어 기반 방법에 비해 아주 큰 장점이다.

ASIC 의 가장 큰 단점은 개발에 따르는 비용이 크며 한번 개발한 내용을 고치고 다시 생산하는데 따르는 시간과 비용이 펌웨어 기반의 제품보다 나쁠 수 밖에 없다는 것이다. 또한 대부분의 ASIC 은 특정 응용에 적합하게 설계되기 때문에 다른 응용에 사용하기 위해서는 많은 비용과 노력이 필요하다.

이상과 같은 ASIC 기반의 TCP/IP offloading 기술을 구현한 TOE(TCP/IP Offloading Engine)으로는 Adaptec(www.adaptec.com)사와 Alacritech(www.aracritech.com)사의 제품을 들 수 있다. 이중 Adaptec 사는 모든 TCP/IP 프로토콜을 내린 전면적 TCP/IP offloading 으로 구현되었으며 Alacritech 사의 경우는 데이터 전송 부분만을 내린 국부적 TCP/IP offloading 을 구현하고 있다.

5.2. 펌웨어 기반 TCP/IP offloading

펌웨어 기반의 TCP/IP offloading 기법의 특징을 정리하면 아래와 같다.

① TCP/IP 스택 자체가 소프트웨어적으로 설계
② 장기적으로 비용이 낮음
③ 새로운 기능 추가가 용이
④ 유지 보수가 용이

TCP/IP 스택은 근본적으로 소프트웨어적으로 설계되어 있어 이를 하드웨어적으로 구현하기는 쉽지 않으며(①), 단기적으로는 ASIC 기반의 구현이 비용이 적게 들지 모르나 장기적으로 제품을 업그레이드 한다면 펌웨어 기반의 구현이 적은 비용이 든다(②). 또한 구현된 TCP/IP 계층위에 새로운 계층을 올리거나 하는 경우에 ASIC 기반의 구현은 새로운 칩을 설계해야 하는 반면 펌웨어 기반의 구현은 소프트웨어적으로 해결할 수 있어 새로운 기능이나 스택의 추가가 용이하다(③). 또한 이미 생산한 제품에 문제가 발생한 경우 펌웨어 기반의 구현은 소프트웨어 개선으로 문제를 해결할 수 있으나 ASIC 기반은 전체 하드웨어를 교체하는 방법밖에는 없다(④).

Windriver(www.windriver.com)사는 TINA(Tornado for Intelligent Network Acceleration)라는 펌웨어 기반의 TCP/IP offloading 개발 플랫폼을 제공한다. TINA 는 TCP/IP offloading 과 더불어 TCP/IP 프로토콜의 성능 향상을 위해 필요한 부분은 하드웨어적으로 구현하였다. 또한 Intel(www.intel.com) 에서 개발한 IXP1200 네트워크 프로세서의 경우에는 TCP/IP offloading 기술 등을 적용하기 쉽도록 하나의 네트워크 프로세서 내에 다수의 마이크로 엔진과 메모리를 포함시켜 네트워크 트래픽의 처리 성능을 향상시킬 수 있는 환경을 제공한다.

6. TCP/IP offloading 기술의 응용

앞서 살펴본 TCP/IP offloading 기술들은 현재 저장장치분야에서 많은 관심을 보이고 있다. 최근의 네트워크 환경의 발달로 IP 네트워크 상에서 운용할 수 있는 저장장치에 대한 관심과 수요가 늘어나면서 iSCSI 의 하위 계층으로 TCP/IP 를 사용하기 위해서는 고속의 네트워크 어댑터 기술이 필요하게 됨에 따라 많은 네트워크 디바이스 제공업체들이 TCP/IP offloading 관련 제품을 개발하고 있는 상황이다[4]. 이와 더불어 서버 컴퓨팅 분야에서 날로 늘어나는 네트워킹 요구를 처리하기 위해서 TCP/IP offloading 기술을 적용하려는 추세가 나타나고 있다.

7. 결론

급속히 증가하는 네트워크 수요를 만족시키기 위해서는 TCP/IP 의 성능을 올릴 수 있는 TCP/IP offloading 기술은 반드시 필요한 기술이 될 것이다. 하지만, TCP/IP offloading 기술을 적용하기 위해서는 각각의 구현에 따른 특징을 명확하게 분석하여 적용대상에서 가장 좋은 성능과 기능을 발휘할 수 있는 방법을 찾고 이를 실제적으로 구현할 수 있는 기술이 있을 때에만 기대하는 효과를 볼 수 있을 것이다. 가장 간단한 예로 일반적인 인터넷 환경에서 사용할 것이라면 데이터 전송 시 에러가 거의 없는 환경에서 좋은 성능을 발휘하는 국부적 TCP/IP offloading 기술을 선택하는 것은 그렇게 좋은 방법이 아니다. 물론 실제 구현을 어떻게 하느냐에 따라서 성능이 결정될 수 있지만, 일반적인 경우에 적합한 결정이라고는 볼 수 없다. 또한, 기존의 응용프로그램과의 호환성을 제공할건지의 여부도 중요한 사항이다. 특정한 응용만을 위한 시스템이라면 호환성은 중요한 요소가 아니지만, 그렇지 않은 경우에는 반드시 고려해야 할 대상이며, 또한 구현하는 방법에 따라 많은 성능 차이를 보일 수도 있기 때문이다.

8. 참고문헌

[1] D. D. Clark, V. Jacobson, J. Romkey, H. Salwen. An analysis of TCP processing over-head. In IEEE Communications, vol.27, no.6, pp.23-29, 1989.

[2] Peter Druschel, Larry L. Peterson, and Bruce S. Davie. Experiences with a high-speed network adaptor: A software perspective. In Proceedings of the ACM SIGCOMM Conference, pages 2-13, August 1994

[3] J. Chu. Zero-copy TCP in Solaris. In Proceedings of the 1996 Usenix Technical Conference, pages 253--64, San Diego, CA, USA, Jan. 1996.

[4] Delivering High-Performance Storage Networking, Alacritech Inc, (http://www.alacritech.com/html/whitepapers.html), 2001