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

RUM: 미리넷을 위한 신뢰성 있는 UDP

김진욱°, 진현욱, 유혁
고려대학교 컴퓨터학과
{jukim°, hwjin, hxy}@os.korea.ac.kr

RUM: Reliable UDP on Myrinet

Jin-Ug Kim° Hyun-Wook Jin Hyuck Yoo
Dept. of Computer Science and Engineering
Korea University

요 약

클러스터와 같은 네트워크 컴퓨팅 환경에서는 신속하고 신뢰성이 보장되는 데이터 전송이 요구된다. 신뢰성 보장을 위해서 일반적으로 사용되는 전송 프로토콜은 TCP 이다. 그러나 클러스터의 하부 네트워크로서 많이 사용되는 Myrinet 은 cut-through 스위칭 방식을 기반으로 하기 때문에 네트워크 혼잡(congestion)이 발생하지 않는다. 따라서 TCP 의 혼잡 제어(congestion control) 등과 같은 루틴들은 Myrinet 상에서 불필요한 오버헤드를 발생시킨다. 본 논문은 Myrinet 네트워크에서 흐름 제어(flow control)만으로도 신뢰성을 보장할 수 있음을 보이고 TCP 보다 오버헤드가 적은 UDP 에 흐름 제어를 구현한 RUM(Reliable UDP on Myrinet)을 제안한다. 성능 측정 결과, RUM 은 신뢰성을 보장함과 동시에 TCP 보다 최대 34% 더 높은 처리량(throughput)을 보이며, UDP 와 비슷한 낮은 단방향 지연시간(one-way latency)을 보장함을 알 수 있다.

1. 서론

고속의 네트워크를 기반으로 하는 클러스터링 환경에서는 신뢰성이 보장되는 빠른 데이터 전송이 필요하다. 일반적으로 신뢰성이 보장되는 데이터 전송이 필요한 환경에서는 TCP 가 사용이 된다.

TCP 는 윈도우(window)를 기반으로 전송율을 조절하며, 네트워크 혼잡(congestion)으로 인하여 패킷 손실이 발생한 경우 해당 패킷을 재전송하는 혼잡 제어(congestion control) 메커니즘을 통하여 신뢰성을 보장한다. 네트워크 혼잡은 store-and-forward 방식을 기반으로 하는 라우터에서 입력과 출력 링크의 서로 다른 데이터율에 의한 버퍼 부족 현상이 그 원인이다.

Myrinet[1]과 같이 cut-through 스위칭[2]을 기반으로 하는 네트워크에서는 혼잡이 발생하지 않는다. 따라서 신뢰성을 보장하기 위한 TCP 의 재전송과 혼잡 제어 등과 같은 메커니즘[3]은 Myrinet 네트워크에서는 불필요한 오버헤드를 발생시켜 응용 프로그램이 얻을 수 있는 성능을 제약하는 한 요인이 된다.

본 연구는 2002 년 정보통신부의 대학 S/W 연구 센터지원 사업에 의해 연구됨.

이와 같은 문제점을 해결하기 위하여 본 논문에서는 cut-through 스위칭 네트워크와 같이 혼잡이 없는(congestion-free) 환경에서 신뢰성이 보장되는 프로토콜인 RUM(Reliable UDP on Myrinet)을 제안한다. RUM 은 수신측의 가용 메모리 상태를 기반으로 송신측이 흐름 제어를 하는 UDP 기반의 전송 프로토콜이다. 성능 측정을 통해서 RUM 은 TCP 보다는 높은 처리율(throughput)을 성취하면서, UDP 와 같이 낮은 단방향 지연시간(one-way latency)을 갖음을 보인다.

결과적으로 본 논문은 클러스터를 위한 전송 프로토콜의 신뢰성을 위한 연구에 기여한다. 기존의 연구들은 신뢰성을 위해서 재전송 등을 포함한 여러 가지 기능을 구현하지만, 본 연구는 하부 네트워크의 특성을 이해함으로써 최소한의 기능으로 신뢰성을 보장할 수 있음을 보인다. 또한 본 논문에서 제시하는 기법은 VIA(Virtual Interface Architecture)[4]와 같은 사용자 수준의 프로토콜(user-level protocol)에도 적용 가능하다.

본 논문은 다음과 같이 구성되어 있다. 서론에 이어 2 장에서는 cut-through 스위칭을 기반으로 하는 Myrinet 상에서 TCP 의 특성을 분석한다. 그리고 3 장에서는 관련 연구를 살펴본다. 4 장에서는 신뢰성을 보장하면서도 빠른 데이터 전송을 위하여 고안된 RUM

을 소개한다. 5 장에서는 RUM 의 성능을 기존의 TCP, UDP 와 비교 분석한다. 마지막으로 6 장에서 결론 및 향후 연구 계획을 기술한다.

2. Myrinet 에서의 TCP 특성 분석

TCP 는 데이터 전송 시 신뢰성을 보장하는 대표적인 종단 간 전송계층 프로토콜이다. TCP 는 네트워크를 일종의 블랙박스(black box)로 간주하고, 수신측에서 보내는 ACK 을 기반으로 네트워크 상황을 판단한다. Store-and-forward 방식을 사용하는 일반적인 라우터에서는 메모리 부족에 의해서 패킷 손실이 발생한다. 즉, 라우터로 들어오는 패킷의 비율이 나가는 패킷의 것보다 큰 경우에 할당할 메모리가 부족하게 되어 패킷을 버리게 된다. 패킷 유실로 인하여 발생된 타임아웃과 중복된 ACK 을 통하여 TCP 는 네트워크가 혼잡한 것으로 판단하고 혼잡제어(congestion control)기법을 수행한다.

하지만 클러스터링 환경에서 많이 사용 되는 Myrinet 은 store-and-forward 가 아닌 cut-through 스위칭 방식을 사용한다. Cut-through 방식은 수신되는 패킷의 목적지 주소를 확인하자마자 아직 수신 완료가 되지 않았더라도 목적지 포트로 패킷을 전송하는 방식이다. 그러므로 store-and-forward 방식의 네트워크와는 다르게 cut-through 스위치 네트워크에서는 패킷 전송 대기 시간이 최소화된다.

이와 같은 Myrinet 에서 TCP 의 특성을 분석하기 위하여 그림 1 과 같은 실험 환경을 구성하였다. 각각의 종단 노드는 Pentium III 450MHz 프로세서와 Myrinet LANai-4 NIC(M2F-PCI32C)을 장착하여 8 port 의 Myrinet 스위치(M2F-SW8)로 연결하여 실험 환경을 구성하였다. 그리고 실험 노드의 운영체제로 Linux (커널 버전 2.4)를 사용하였으며, 디바이스 드라이버와 NIC 펌웨어를 위해서 GM (버전 1.5.1)을 사용하였다.

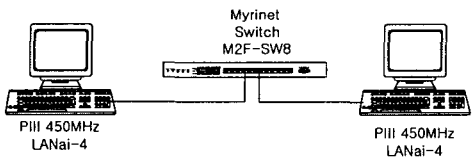

그림 1. 실험환경 구성도

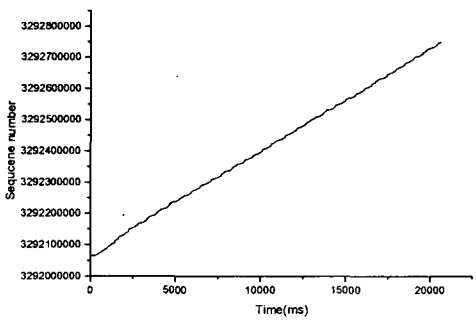

그림 2. 수신 패킷 순차번호

측정된 값은 시간에 따른 패킷 순차번호, *congestion window*, 그리고 수신측 가용 메모리 상태를 알리는 advertised window 등이다.

그림 2 는 수신측에서 수신된 패킷의 순차번호를 보여준다. 그림과 같이 Myrinet 에서는 패킷의 순서가 뒤바뀌는 현상이 발생하지 않음을 알 수 있다. 따라서 재순서화(reordering)와 같은 기법은 필요하지 않다.

TCP 는 윈도우를 기반으로 수신측 가용 메모리 상태를 고려하는 흐름 제어와 네트워크 혼잡에 대처하는 혼잡 제어를 수행한다. 그리고 전송 가능 윈도우(available window)는 수식 1 에 의해서 결정된다. 중간 노드에서 발생하는 혼잡으로 인하여 패킷 손실이 빈번하게 발생되는 일반적인 네트워크에서는 congestion window 가 TCP 의 전송 가능 윈도우 크기를 결정하게 된다.

Available window = min (cwnd, awnd)
cwnd : congestion window
awnd: advertised window

수식 1. TCP 의 전송 가능 윈도우

그러나 실험을 통하여 알아본 Myrinet 에서의 TCP 는 advertised window 가 전송 가능 윈도우 크기를 결정하는 것을 그림 3 과 같이 알 수 있었다. 그리고 수신측은 메모리 고갈 상태를 알리는 ZWA(Zero Window Advertisement)를 자주 발생시킨다. 따라서 송신측은 사용 가능한 대역폭을 효율적으로 이용하지 못하고, 수신측 가용 메모리에 제약을 받게 된다.

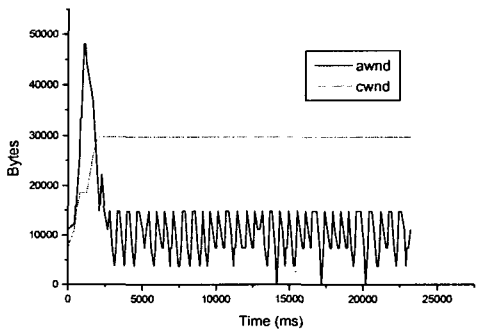

그림 3. Congestion window 와 Advertised window 비교

이와 같이 네트워크 혼잡이 발생하지 않고 수신측 가용 메모리, 즉 advertised window 가 전송율을 결정하는 이유는 Myrinet 이 cut-through 스위칭 방식을 사용하기 때문이다. 결과적으로 Cut-through 스위칭 기반의 네트워크에서는 흐름제어만을 이용하여 신뢰성이 보

장될 수 있음을 알 수 있다.

3. 관련 연구

FM(Fast Message)[5]은 신뢰성 보장을 위해서 return-to-sender 프로토콜을 구현하였다. Return-to-sender 프로토콜은 수신측 가용 메모리 부족으로 패킷 손실이 발생하는 경우를 대비하여 송신측이 거부 큐(reject queue)를 유지하도록 한다. 수신측이 받은 패킷에 대하여 사용할 가용 메모리 공간이 없는 경우, 해당 패킷을 버리고 송신측에 다시 전송을 하도록 알린다. 이와 같이 FM 은 흐름 제어 외에도 재전송 기법을 구현하여 신뢰성을 보장하고 있다.

RBUDP(Reliable Blast UDP Protocol)[6]는 사용자가 명시한 전송 속도로 데이터를 UDP 로 전송한다. 수신측 메모리 부족에 의하여 손실된 패킷에 대한 정보는 TCP 를 이용하여 송신측에 비트맵 정보 형태로 알린다. 이것은 대량의 데이터를 전송한 후 손실된 패킷을 ACK 정보를 바탕으로 재전송하는 메커니즘이다. 따라서 RBUDP 도 FM 과 같이 재전송 기법을 이용하여 신뢰성을 보장하고 있다.

Trapeze[7]는 TCP 를 성능 최적화한 프로토콜로서 TCP 가 작동하는 방식과 동일하게 혼잡제어와 흐름제어를 수행한다. 따라서 혼잡이 발생하지 않는 Myrinet 환경에서는 불필요한 오버헤드들이 존재한다.

4. RUM – Myrinet 을 위한 신뢰성 있는 UDP

2 장에서 언급한 바와 같이 혼잡이 존재하지 않는 Myrinet 에서는 흐름 제어가 신뢰성을 보장하는 요인이 된다. 본 장에서는 UDP 를 기반으로 흐름 제어를 구현하여 신뢰성이 보장되는 프로토콜인 RUM(Reliable UDP on Myrinet)을 제안한다.

RUM 이 사용하는 데이터 패킷과 제어 패킷의 헤더는 그림 4 와 같다. RUM 은 송신측이 보내는 패킷 헤더에 순차 번호를 부여한다. 그리고 수신측이 보내는 제어 패킷은 헤더에 가장 최근에 받은 패킷의 번호와 수신측 가용 메모리 크기를 포함한다. 이와 같이 신뢰성 보장을 위해서 추가되는 헤더의 크기가 작기 때문에 추가되는 헤더 오버헤드는 미약하다.

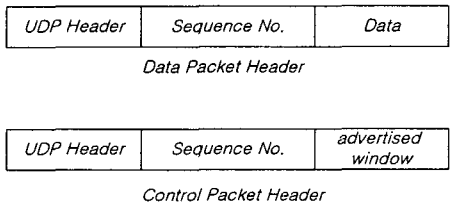

그림 4. RUM 프로토콜 헤더 구조

송신측은 수식 2 와 같이 수신측에서 알려주는 가용 메모리 상태를 바탕으로 흐름제어를 수행한다. 수식 2 의 흐름 제어는 보낸 메시지가 수신측에 전달 된 후 가용 메모리 부족으로 패킷 손실이 발생하지 않는 것을 보장한다.

available memeory = awnd – (txed – acked)
txed : 최근 보낸 패킷 번호
acked : 마지막으로 받은 ACK 패킷 번호
awnd : advertised window

수식 2. RUM 의 흐름 제어

5. 실험 및 성능 평가

실험 환경은 2 장에서 언급한 것과 동일하게 구성을 하고 RUM 을 구현하였다. 성능 분석을 위해서 기존 TCP 와 UDP, 그리고 RUM 의 처리량 및 지연시간을 측정하여 비교하였다. 처리율은 패킷의 크기를 1.5KB 에서 7KB 까지 변화시키며 각각 1000 회 전송하여 측정하였다. 그리고 단방향 지연시간은 ping-pong 프로그램을 사용하여 측정했다.

그림 5 와 같이 RUM 과 TCP 의 처리율 비교를 통하여, RUM 은 데이터 크기가 작을수록 좋은 처리율을 보임을 알 수 있다. 1.5KB 의 데이터에 대하여 RUM 은 TCP 에 비하여 34% 더 높은 성능을 보이고 있다. 데이터 크기가 증가할수록 TCP 와 RUM 간의 처리량 차이는 작아지고 있다. 이것은 데이터를 전송하는 과정에서 per-byte 오버헤드가 per-packet 오버헤드 보다 커지기 때문이다. 따라서 RUM 프로토콜은 특히 작은 데이터의 패킷을 전송할 때 효율적임을 알 수 있다. RUM 의 처리량은 데이터 복사 제거 등의 기법들[8, 9]을 적용하여 더 향상될 수 있을 것으로 기대된다.

UDP 는 전송 과정에서 수신측 메모리 부족으로 패킷 손실이 발생한다. 1.5KB 의 데이터를 UDP 를 사용하여 전송할 경우 최대 71.5%의 손실률을 보였다. 이와 같이 정확한 처리량을 측정하기에 어려움이 따르기 때문에 그림 5 에서 UDP 의 처리율은 포함하지 않았다.

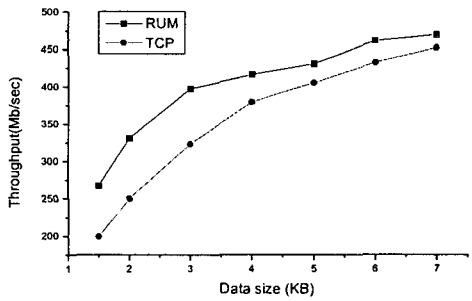

그림 5. TCP 와 RUM 의 처리율 비교

클러스터 환경에서 단방향 지연시간은 프로토콜 성능을 비교하는 주요한 요소 중의 하나이다. RUM 은

그림 6 에서 보는 것처럼 UDP 와 유사한 단방향 지연시간을 보이고 있다. 반면 TCP 는 UDP 나 RUM 보다 단방향 지연시간이 현저하게 크다.

결과적으로 RUM 은 흐름제어만을 구현함으로써 신뢰성을 보장하고 TCP 보다 높은 처리율을 성취할 수 있으며, 흐름 제어의 구현으로 인한 오버헤드는 미약함을 알 수 있다.

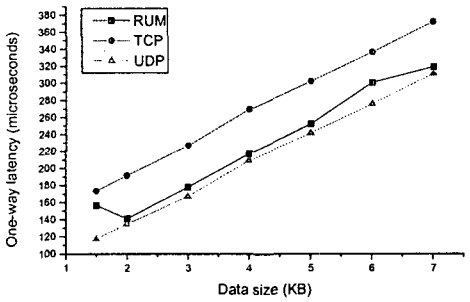

그림 6. **TCP, UDP, RUM** 의 단방향 지연시간 비교

6. 결론 및 향후 계획

클러스터링 환경에서 신뢰성이 보장되는 데이터 전송은 필수적인 요소이다. Myrinet 과 같은 cut-through 스위칭을 기반으로 하는 네트워크에서는 혼잡이 발생하지 않는다. 따라서 Myrinet 에서는 신뢰성 보장을 위해서 네트워크 혼잡 등을 제어하는 TCP 의 모듈들은 불필요한 오버헤드를 발생한다.

이를 해결하기 위해서 본 논문은 수신측 메모리 상태에 기반을 두고서 흐름 제어를 하는 RUM 을 개발하였다. RUM 은 흐름제어만으로도 신뢰성을 보장하며 TCP 프로토콜보다 높은 처리량을 보인다. 그리고 UDP 프로토콜과 유사한 지연시간을 보이고 있다. Myrinet 에서 흐름 제어만으로도 신뢰성을 보장할 수 있다는 RUM 의 기본 틀은 Myrinet 을 대상으로 개발된 많은 프로토콜에 응용될 수 있다.

향후 연구로서 효율적인 흐름제어가 가능하도록 하는 ACK 전송 메커니즘을 고안 중이다. ACK 을 보내는 시점은 RUM 프로토콜의 성능과 밀접한 관계를 갖는다. ACK 을 자주 발생시키면 단방향 지연시간은 짧아지지만 처리율은 낮아진다. 이와 반대로 ACK 의 간격을 크게 설정하면 처리율에서는 이득이 있지만 단방향 지연시간은 커진다. 따라서 이 둘의 균형을 잘 맞출 수 있는 ACK 기법이 요구된다.

참고문헌

[1] N. Boden, D. Cohen, R. Feldeman, A. Kulawik, C. Seitz, J. Seizovic, and W-K Su, "Myrinet – a gigabit-per-second local area network," IEEE Micro, February 1995.

[2] P. Kermani and L.Kleinrock, "Virtual cut-through: A new computer communication switching technique," Computer Networks, Vol. 3, pp. 267-286, 1979.

[3] W. Stevens, TCP/IP Illustrated, Volume I: The Protocols, Addison-Wesley, 1994.

[4] Dave Dunning, Greg Regnier, Gary McAlpine, Don Cameron, Bill Shubert, Frank Berry, Anne Marie Merritt, Ed Gronke, and Chris Dodd, " The Virtual Interface Architecture," IEEE Micro, Vol. 18, No. 2, pp. 66-76, March/April 1998.

[5] S. Pakin, Lauria, and A. Chien, "High Performance Messaging on Workstations: Illinois Fast Messages (FM) For Myrinet," Proceedings of Supercomputing '95, 1995.

[6] E. He, J. Leigh, O. Yu, and T. A. DeFanti, " Reliable Blast UDP: Predictable High Performance Bulk Data Transfer," Proceedings of IEEE Cluster Computing 2002, Chicago, Illinois, September 2002.

[7] K. G. Yocum, J. S. Chase, A. J. Gallatin, and A. R. Lebeck, "Cut-through delivery in Trapeze: An exercise in low-latency messaging," Proceedings of 6th IEEE International Symposium on High Performance Distributed Computing (HPDC-6), pp. 243-252, August 1997.

[8] Chuck Yoo, Hyun-Wook Jin, and Soon-Cheol Kwon, "Asynchronous UDP," IEICE Transactions on Communications, Vol. E84-B, No. 12 pp. 3243-3251 December 2001.

[9] Hyun-Wook Jin, Chuck Yoo, and Sung-Kyun Park, "Stepwise Optimizations of UDP/IP on a Gigabit Network," Proceedings of 8th International Euro-Par Conference, LNCS, Vol. 2400, pp. 745-748, August 2002.