IXP1200 네트워크 프로세서를 이용한 Receive Thread의 설계

박우진[0] 이병재 왕명안 황광섭 안순신
고려대학교 전자공학과
{progress, airman01, wangma, hanriver, sunshin}@dsys.korea.ac.kr

The design of the Receive Thread in IXP1200 Network Processor

Woo-Jin Park[0] Byung-Jae Lee Ming-An Wang Kwang-Seop Hwang Sun-Shin An
Computer Network Lab. Dept. of Electronics Eng., Korea University

요 약

인터넷의 급격한 성장과 함께 네트워크 서비스에 대한 사용자의 요구도 점점 증대되고 있다. 이러한 시장의 요구에 빠르게 대응하고 새로운 특징에 대한 시스템의 수정과 보완이 용이하게 되도록 고안된 것이 네트워크 프로세서이며, 본 논문은 인텔의 IXP1200 네트워크 프로세서를 이용한 포워딩 엔진의 한 모듈인 Receive Thread를 설계한다.

1. 서 론

최근 수년간 네트워크 응용들은 점점 복잡해 지고 있으며, 이를 지원하기 위한 네트워크서비스의 필요성이 대두되고 있다. ASIC에 의한 하드웨어적인 방법은 고성능인데 반해, 일단 설계되면 수정이 불가능해서 새로운 특징을 시스템에 첨가하기 위해서는 많은 시간이 소요된다. 이러한 단점의 극복을 위해 탄생한 것이 네트워크 프로세서이며, 프로그램밍이 가능한 소프트웨어적인 방법을 사용하므로 시장의 요구에 빠르게 대응하고 새로운 특징에 대한 시스템의 수정과 보완이 용이하다. 또한, 집적회로 기술의 발전과 함께 성능이 계속 증대되고 있으며, 이러한 네트워크 프로세서를 이용한 네트워크 및 통신장비는 날로 확대될 전망이다.

2. 관련 연구

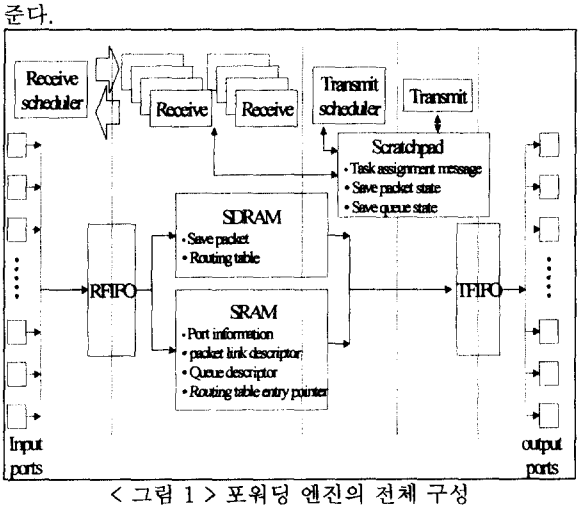
2.1 IXP1200 네트워크 프로세서의 개요

인텔사에 의해 개발된 IXP1200 네트워크 프로세서는 프로그래밍이 가능한 6개의 32비트 RISC 마이크로 엔진과 StrongARM 프로세서 코어를 탑재하고 있다. 6개의 마이크로 엔진은 패킷 포워딩을 담당하며, 3계층에서 64바이트 패킷을 초당 3백만개 까지 포워딩 할 수 있다(약, 1.5Gbps). 하나의 마이크로 엔진은 IXP 1200 core frequency (200MHz)에서 동작하고 모든 instruction은 한 cycle에 수행된다. 하나의 마이크로 엔진은 4개의 context를 지원하고 전체 24개의 context로 이루어져 있다.

StrongARM 프로세서는 포워딩 테이블의 유지 및 보수, 망관리, 라우팅 프로토콜의 처리등의 기능을 담당한다.

2.2 포워딩 엔진의 전체구성

포워딩 엔진의 설계을 위한 코드의 전체구성은 Receive Scheduler, Receive Thread, Transmit Scheduler, Transmit Thread의 네 모듈로 구성된다. Receive 부분은 입력포트에 수신된 64바이트 MAC Packet (MPKT)을 RFIFO로 임시저장하고, 헤더의 수정을 거친 후 하나의 완전한 패킷을 출력을 위해 SDRAM까지 저장하는 역할을 담당하며, Transmit 부분은 SDRAM에 저장된 패킷을 TFIFO에 임시저장하고, 출력포트로 패킷을 전송하는 역할을 담당한다. 그림 1은 포워딩 엔진의 전체구성을 보여준다.



< 그림 1 > 포워딩 엔진의 전체 구성

RFIFO와 TFIFO는 IX 버스를 통해 포트로부터 패킷을 보내고 받기 위해 사용되며, 각 FIFO는 16개의 element를 가지고 있으며, 각 element는 64바이트로 구성되어 있다. Scratchpad는 마이크로 엔진의 제어 및 동기를 맞추고, 프로세서간의 정보교환을 위해 사용되며, SRAM과 함께 StrongARM과의 통신을 위한 용도이다.

2.3 Receive Thread 설계시의 고려사항

Slow Port와 Fast Port는 패킷 수신률이 서로 다르고 Receive Thread가 수신한 데이터를 처리하는데 걸리는 시간이 다르기 때문에 패킷의 순서를 다른 방식으로 취급하며, 이들 각각의 처리방법은 다음과 같다.

Slow Port는 port blocking 방식을 사용하며, 이것은 하나의 Receive Request가 처리되고 나서 메모리 버퍼로 놓여진 후에야 다시 동일한 포트로의 Receive Request가 가능하도록 해주는 것이다.

Fast Port는 MPKT들이 다른 Receive Thread에 의해 병렬적으로 처리되기 때문에 3개의 sequence number을 사용하여 패킷의 순서를 유지하며, 이들이 유지되는 규칙은 다음과 같다.

- ✓ SOP(Start Of Packet) sequence number는 SOP신호가 탐지될 때 마다 증가된다.
- ✓ MPKT sequence number는 SOP신호가 탐지되면 SOP sequence number를 상속하며, 그렇지 않으면, Receive Request마다 증가된다.
- ✓ Enqueue sequence number는 완전한 하나의 패킷을 SDRAM의 Transmit Queue로 이동시키는데 사용된다. 즉, 이 sequence number가 SOP sequence number가 일치할 경우 Receive Thread가 Transmit Queue로 패킷을 이동시킬 수 있으며, 그렇지 않으면, 일치할 때까지 대기한다. 패킷이 Transmit Queued로 이동된 후에 이 sequence number는 증가된다.

3. 포워딩 엔진의 설계

3.1 설계 시나리오

본 논문의 설계에서는 8개의 Slow Port(100Mbps)와 1개의 Fast Port(1Gbps)가 사용되며, Receive Scheduler에 의해 패킷을 수신받아 RFIFO까지 옮겨졌다는 전제하에 Receive Thread에 중점을 두어 설명한다.

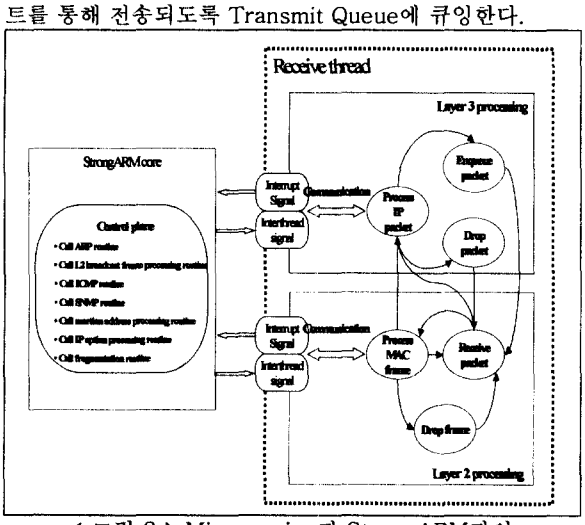
Receive Thread는 Layer 2와 Layer 3에 관련된 패킷 포워딩 기능을 주목적으로 하고 몇몇 예외들은 StrongARM core control plane으로 보내어 처리되도록 하는 기능을 수행한다. 그림 2는 Receive Thread에서의 마이크로 엔진과 StrongARM과의 상태천이 다이어그램이며, 간단히 설명하면 다음과 같다.

- Layer 2 processing

여기서, Receive Thread의 주기능은 Link Layer로부터 패킷을 받아 MAC 헤더를 증명하고 수정하는 것이다. 즉, 처음에 MAC layer protocol를 확인하고, 이더넷 프레임이라면 계속 처리를 위해 나아가고, 802.3 프레임이라면 그것을 폐기하고 다시 입력포트로부터 패킷을 수신하도록 한다. 그 패킷의 source address는 전송될 포트의 MAC 주소로 갱신되며, destination address는 route lookup에 의해 얻어진 next hop의 destination MAC address로 수정된다. 그리고, ARP패킷의 경우에는 StrongARM core control plane으로 보내어 그곳에서 처리되도록 한다.

- Layer 3 processing

Layer 2에서의 처리를 마친 후 건네진 패킷에 대해서 다음을 수행한다. IP헤더의 source와 destination address을 타당화하며, Martian address이거나 directed broadcast address이면 그 패킷을 폐기한다. 만약, destination address가 local host가 아니라면 TTL를 감소시키고, checksum을 재계산한다. 그 후, route lookup을 수행하며 다음 인터페이스를 결정하고, lookup에 근거하여 destination과 source MAC address를 수정한다. 또한, 패킷이 IP option을 포함하는 경우, 패킷 크기가 path MTU보다 큰 경우, 그리고, ICMP 패킷인 경우는 StrongARM core control plane으로 보내어 그곳에서 처리되도록 한다. 마지막으로 완전한 하나의 패킷이 출력 포트를 통해 전송되도록 Transmit Queue에 큐잉한다.



< 그림 2 > Microengine과 StrongARM과의 상태천이 다이어그램

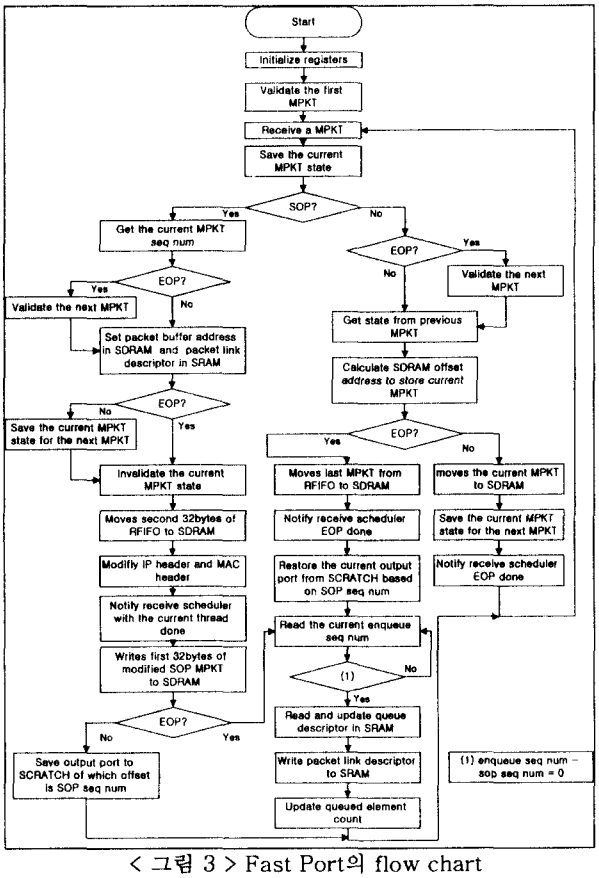
3.2 Receive Thread의 설계

Receive Scheduler에 의해 각 포트로부터 패킷이 64바이트씩 분할되어 수신되며, 포트에 따라 할당된 RFIFO로 임시저장된다. 본 논문의 설계에서는 Receive Thread에 3개의 마이크로 엔진이 할당되며, 8개의 Slow Port에는 각각 하나의 RFIFO element가 할당되며, 1개의 Fast Port는 Slow Port보다 패킷 수신률이 빠르므로 8개의 RFIFO element가 할당된다. Receive Thread는 이렇게 64바이트씩 분할되어 수신되는 각각의 MPKT들의 SOP와 EOP(End Of Packet)의 정보를 활용하여 분할된 MPKT를 원래 패킷의 시작과 끝을 판단하며, 헤더의 분석 및 수정 후 하나의 완전한 패킷이라고 판단되는 경우에만 SDRAM Transmit Queue의 적당한 위치로 저장한다.

또한, Receive Thread는 하나의 MPKT를 처리하고 나서 Receive Scheduler에게 다른 MPKT가 수신되도록 신호를 주며, 전송될 패킷이 발생할 때마다 Transmit Scheduler에게 알려준다. 각각의 입력 포트는 향후 QoS의 지원을 위해 8개의 큐를 두며, 이들 Transmit Queue들은 Queue Descriptor에 의하여 정의되며, SRAM에서 유지된다. Queue Descriptor에는 Transmit Queue의 처음과 마지막 MPKT에 대한 포인터와 MPKT의 개수에 대한 정보가 포함되며, 각각의 패킷은 Packet Link Descriptor의 linked list로서 유지된다.

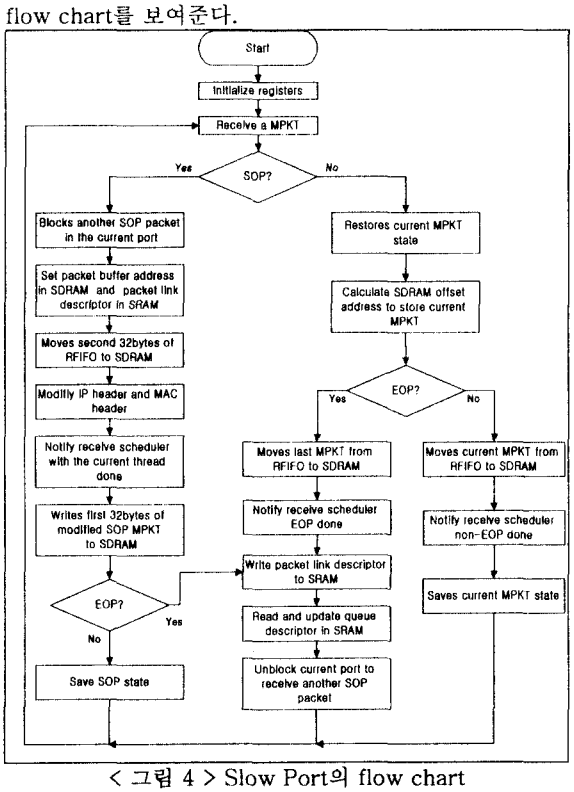
앞에서 언급한 바와 같이, Receive Thread는 Fast Port와 Slow Port에서 수신된 패킷에 대한 순서화를 다른 방식으로 처리하기 때문에 이들 각각을 분리하여 설명하며, 패킷이 처리되는 동작에 초점을 맞추어 설명한다.

Fast Port에서는 MPKT들이 여러 Receive Thread에 의하여 처리되므로 이들을 관리하기 위해 마이크로 엔진들이 공용으로 사용하는 Scratchpad영역에 상태정보를 기록하여 마이크로 엔진들간의 패킷의 동기를 맞추어야 하며, 3개의 sequence mumber를 이용하여 패킷의 순서를 맞춘다. 그림 3은 Fast Port에서의 flow chart를 보여준다.



< 그림 3 > Fast Port의 flow chart

Slow Port에서는 Fast Port와는 달리, 하나의 패킷이 하나의 Receive Thread에 의하여 처리되므로 sequence number를 유지하여 MPKT의 순서를 유지하는 것이 아니라 해당 포트로 SOP가 수신되면 그 포트로는 또 다른 패킷의 SOP가 수신되지 않도록 막고, EOP를 수신한 후 그 포트로 SOP가 수신될 수 있도록 함으로써 패킷의 순서를 유지한다. 그림 4는 Slow Port에서의 flow chart를 보여준다.



< 그림 4 > Slow Port의 flow chart

4. 결론 및 향후연구

본 논문에서는 인텔사의 IXP1200 네트워크 프로세서를 이용하여 라우터의 포워딩 엔진 구현을 위한 예비단계로 Receive Thread의 설계를 하였으며, Fast Port와 Slow Port에서 최적으로 패킷을 수신하게끔 설계하였다.

현재까지 추상화한 설계안을 토대로 실질적인 구현을 통해 전체적인 기능향상을 가져오는 기술을 적용하고, QoS등의 기능을 제공하도록 구현해야 하는 과제가 남아 있다.

5. 참조 문헌

[1] "IXP 1200 Hardware Reference Manual" Intel, June, 2000

[2] "IXP 1200 Programmar's Reference Manual" , Intel, May, 2000

[3] RFC 1812 "Requirements for IP Version 4 Routers"

[4] Comer,D.E, "Internetworking with TCP/IP" Volume 3, 4th edition, Pretice-Hall