글로벌 캐시를 이용한 네트워크 병렬 프로세서 구조 연구

준회원 박 재 원*, 정회원 정 원 영*, 김 현 필*, 이 정 희**, 이 용 석***

Study of Parallel Network Processor using Global Cache

Jae-won Park* *Associate Member*,
Won-young Chung*, Hyun-pil Kim*, Jung-hee Lee**, Yong-surk Lee*** *Regular Members*

요 약

현재 광대역 통합망의 사용으로 인해 확장된 망을 사용하는 트래픽의 양이 많아지고, 어플리케이션의 발달로 인해 트래픽의 종류도 증가하고 있다. 특히 IPTV, VOD, 온라인 게임 등의 멀티미디어 속성을 가진 트래픽의 증가가 두드러지고 있다. 이러한 멀티미디어 트래픽은 페이로드의 크기가 클 뿐만 아니라 실시간 처리를 요하기 때문에 라우터에서 트래픽 속성에 따라 차등한 대역폭을 지원하는 연구가 진행 중에 있다. 트래픽의 속성을 정확히 구분하기 위해선 어플리케이션 계층을 분석하여야 하는데, 기존의 네트워크 프로세서 구조에선 L2-4 처리와 L7처리를 순차적으로 처리하고 있다. 본 논문에서는 L2-4와 L7을 병렬로 처리하기 위해 글로벌 캐시를 둔 새로운 병렬 네트워크 프로세서 구조를 제안한다. 제안하는 구조를 검증하기 위해 기존의 네트워크 시스템과 제안한 구조의 네트워크 시스템을 SystemC로 모델링하였으며, L2-4, L7 처리 시간을 측정하기 위해 EEMBC와 SNORT를 이용하여 동일한 시스템에서 시뮬레이션 하였다. 멀티미디어 속성의 동일한 트래픽이 연속적으로 입력될 경우 제안한 구조에서 약 85%의 성능 향상을 보였다.

Key Words : L2~4 and L7 parallel processing, Global Cache, SNORT, EEMBC, Multimedia traffic

ABSTRACT

The mount of network traffic from the Internet is increasing because of the use of Broadband Convergence Networks(BcN). Network traffic is also increasing because of the development of application, especially multimedia traffic from IPTV, VOD, and online games. This multimedia traffic not only has a huge payload but also should be considered a threat in real time. For this reason, this study examines the ways that routers distribute the bandwidth in accordance to traffic properties. To classify the property of the traffic, it is essential to analyze the application layer. However, the general network processor architecture serially processes the L2-4 and L7 layer. We propose a novel parallel network processor architecture with a global cache that processes L2-4 and L7 in parallel. To verify the proposed architecture, we simulated both of the architecture with SystemC. EEMBC and SNORT was used to measure L2-4 and L7 processing time. When multimedia traffic was entered into the network processor in the same flow, the proposed architecture showed about 85% higher performance than general architecture.

※ 본 연구는 지식경제부 및 정보통신연구진흥원의 IT산업원천기술개발사업의 일환으로 수행하였습니다. [2010-8-0986, Scalable 마이크로 플로우 처리 기술 개발]
* 연세대학교 전기전자공학과 프로세서 연구실({jwpark, wychung, hpkim}@mpu.yonsei.ac.kr),
** 한국전자통신연구원 옴니 플로우 프로세서 개발팀(jhlee@etri.re.kr),
*** 연세대학교 전기전자공학과 프로세서 연구실(yonglee@yonsei.ac.kr)
논문번호 : KICS2010-08-391, 접수일자 : 2010년 8월 10일, 최종논문접수일자 : 2011년 1월 4일

Ⅰ. 서 론

현재 BcN(Broadband Convergence Network)이 가속화됨에 따라 확장된 망을 사용하는 트래픽의 양이 증가하고 있다. 특히 IPTV, VOD, 온라인 게임 등 디지털 어플리케이션의 증가로 인해 멀티미디어 트래픽의 양적 증대가 두드러지게 나타나고 있다. 그림 1과 같이 트래픽 변화 예상 추이를 살펴보면, 2012년에 약 20,000 PETA byte까지 트래픽의 양이 증가할 것으로 예상하고 있으며, 특히 IPTV, VOD 등의 멀티미디어 속성을 가진 대용량 데이터의 비중이 약 50%까지 늘어날 것으로 예상하였다.

네트워크의 물리적 속도 또한 빨라지고 있으나, 멀티미디어 트래픽의 증가로 처리해야할 IP 패킷의 수적 증가, 그리고 그에 따른 페이로드(payload)가 증가하고 있다. 이로 인해 라우터 및 스위치에서 폭주(burst) 현상이 잦아져서 버려지는(discard) 패킷이 빈번히 발생하고 있다. 따라서 최신 라우터 및 스위치에선 네트워크에 특화된 명령어들을 갖고 있는 네트워크 프로세서를 개발하고[1], 여러 개의 네트워크 프로세서로 멀티 프로세서 시스템을 이루어 처리량(throughput)을 늘리고 있다.[2]

또한 제한된 네트워크의 대역폭을 효율적으로 사용하기 위해 대용량의 데이터를 가진 멀티미디어 트래픽을 분류하여 특정 대역폭을 할당해주는 연구가 진행 중에 있다. 트래픽을 속성별로 분류하여 같은 속성을 지닌 트래픽을 하나의 플로우(Flow)로 정의하여 처리하면 대역폭을 효과적으로 사용할 수 있다. 이와 같은 연구는 Cisco사의 Quantum Flow[3], Sable network사의 μ-flow, Stanford Nick Mckeown 교수팀의 Open Flow[4] 등이 대표적이다.

이러한 트래픽을 속성에 따라 분류하기 위해서는 OSI(Open Systems Interconnection : 개방형 시스템 상호 접속) 7계층 중 7계층(application Layer : 응용계층)을 분석하여야 한다. 7계층을 분석하기 위해서는 패킷의 헤더뿐만 아니라 페이로드 부분까지 함께 분석해야 하므로 DPI(Deep Packet Inspection) 처리를 요한다. 현재 7계층 분석을 통해 트래픽을 분류하고 같은 속성의 트래픽을 플로우로 정의하여 처리하는 프로세서는 L2-4와 L7을 순차적으로 처리하고 있으며, 특히 연산이 많은 L7처리로 인해 실시간 처리에 어려움을 안고 있다.

따라서 본 논문에서는 과거 L7처리 결과를 글로벌 캐시로 정의한 히스토리 테이블(history table)에 저장하고 현재 입력된 패킷이 히스토리 테이블에 있으면, 과거 L7 처리결과를 사용하여 L2-4와 L7처리를 병렬로 하여 성능을 높이는 구조를 제안한다.

2장에서는 일반적인 네트워크 프로세서의 구조를 설명하고, 3장에서 본 논문에서 제안하는 구조와 동작을 설명한다. 4장에서는 EEMBC와 SNORT 시뮬레이션을 통해 수치화한 지연시간을 SystemC로 모델링한 각 구조에 적용하여 성능 평가를 하였다. 5장에서 결론을 맺는다.

Ⅱ. 일반적인 네트워크 프로세서의 구조

일반적인 형태의 플로우(Flow) 기반의 고성능 네트워크 프로세서 구조는 하나의 플로우를 처리하기 위

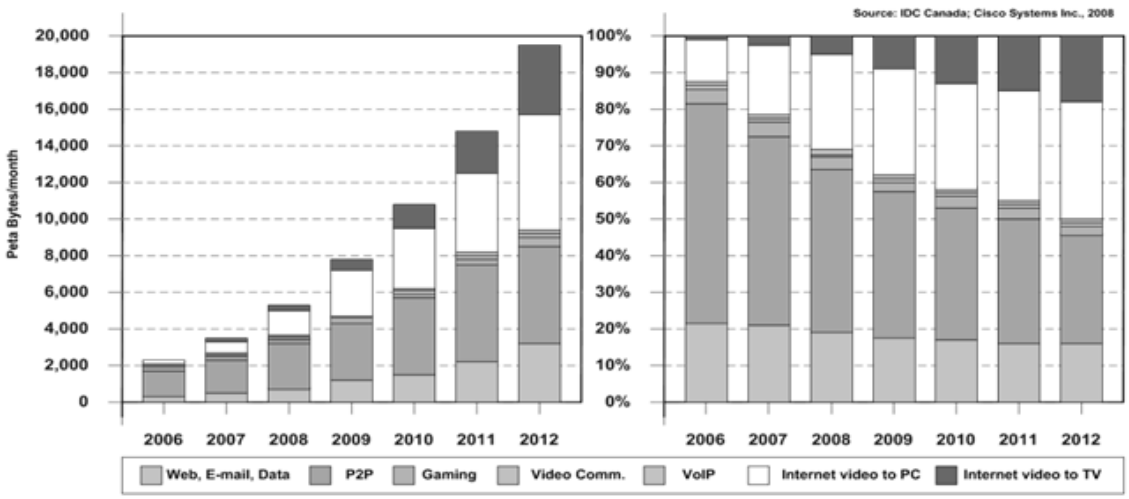

그림 1. 트래픽 변화 예상 추이 (IDC Canada 2008)

해 L2~7의 모든 계층을 순차적으로 처리한다.[5] 그림 2는 일반적인 네트워크 프로세서에서 사용되는 보편적인 구조를 나타낸 것이다. 일반적인 네트워크 프로세서는 패킷의 처리를 효율적으로 하기 위해 계층별로 파이프라이닝을 하거나 패킷을 여러 프로세서에서 동시에 처리하는 방식을 사용한다. 일반적인 구조에서는 다음과 같은 순서로 패킷을 처리한다.

1. 패킷이 입력 큐를 통해서 입력되면 전처리 프로세서(Preprocessor)에서 어떤 플로우에 속했는지 판단하고, 패킷을 프로세서가 처리하기 쉬운 크기를 분할한다.
2. 전처리 프로세서는 L2~4 프로세서에 분할한 패킷을 전달한다.
3. L2~4 처리가 끝나면 패킷을 L7 프로세서에 전달한다. L7프로세서에서는 패킷은 속성별로 분류한다.
4. L7프로세싱이 끝난 패킷은 L2~4 프로세서로 다시 보내진다.
5. L2~4 프로세서에서 속성별로 분류한 정보를 바탕으로 트래픽 관리를 하여 출력 큐로 보낸다.

앞에서 언급한 구조에서는 처리시간이 긴 L7처리가 끝날 때까지 L2~4 프로세서는 기다려야 하는 문제점이 발생하게 된다. 이를 해결하기 위한 구조를 다음 장에서 제안한다.

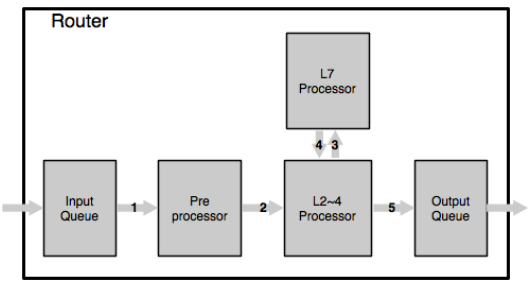

그림 2. 일반적인 네트워크 프로세서의 구조

Ⅲ. 제안하는 네트워크 프로세서 구조

일반적인 플로우 기반의 고성능 네트워크 프로세서의 구조에서는 모든 플로우가 L2~L7의 처리 과정을 거친다. 이를 빠르게 처리하기 위해 패킷 헤더 처리(L2~L4)와 패킷 처리를 포함한 페이로드 처리(L7)를 병렬적으로 처리할 수 있다. 하지만 L2~4와 L7을 병렬로 처리하는 것은 패킷 헤더 처리 시간은 비교적 일정한 반면 페이로드 처리는 패킷의 크기에 비례하여 선형적으로 증가하기 때문에 패킷의 크기가 증가하는 미래 인터넷에서는 단순히 L2~4와 L7을 병렬처리 하는 것으로는 높은 성능 향상을 기대하기 어렵다.

따라서 본 논문에서는 그림 3과 같이 글로벌 캐시를 이용하여 L7의 처리 이력을 기록하고, 이를 이용하여 처리된 이력이 있는 플로우의 패킷은 L2~4에서 처리하여 속도를 높이는 방법을 제안한다. 그림 4는 제안하는 네트워크 프로세서에서 패킷을 처리하는 흐름도이다.

제안하는 구조에서 각 모듈은 다음과 같은 작업을 한다. 전처리 프로세서에서는 패킷 헤더와 페이로드의 일부로 해시 값(hash value)을 만들고, 이를 이용해서 해쉬 키(hash key)를 생성하는 작업과 패킷을 처리하기 쉬운 크기로 나누는 일이다. L2~4프로세서는 정해진 정책에 따라서 패킷을 분류하고 분석하여 트래픽 관리를 하는 일을 수행하게 된다. DB는 페이로드가 저장되는 공간이다. DB는 L2~4프로세서의 처리가 모두 끝날 때 까지 페이로드를 가지고 있고, 처리가 끝나면 패킷을 다시 하나로 만드는 작업을 할 때 L2~4프로세서로 보내는 역할을 수행하게 된다. 글로벌 캐시는 해쉬 키를 주소로 사용하여 L7프로세서에서 처리한 플로우 값으로 정보를 가지고 있고, L2~4 프로세서는 전처리 프로세서에서 받은 해쉬 키를 이용하여 글로벌 캐시에 있는 정보를 이용한다.

패킷을 처리하는 순서는 다음과 같다.

1. 입력 큐에서 패킷이 전달되면 전처리 프로세서에서 플로우를 분류하고 패킷을 처리한다. 이를 헤더와 해쉬 키는 L2-4프로세서로, 페이로드는 DB로 보내진다. 동시에 L7프로세서에 헤더와 해쉬 키, 페이로드가 모두 보내진다.
2. 패킷의 헤더와 해쉬 키가 들어오면 L2-4프로세서는 해쉬 키를 주소로 사용하는 글로벌 캐시를 참조하여 들어온 패킷이 처리된 적이 있는 플로

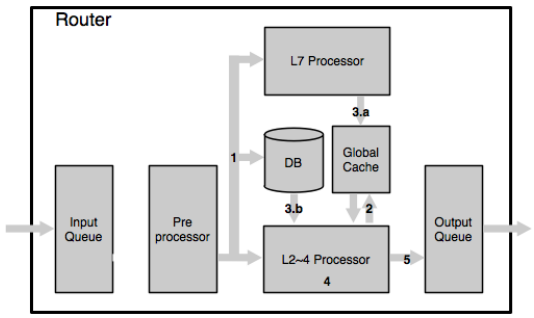

그림 3. 제안된 네트워크 프로세서의 구조

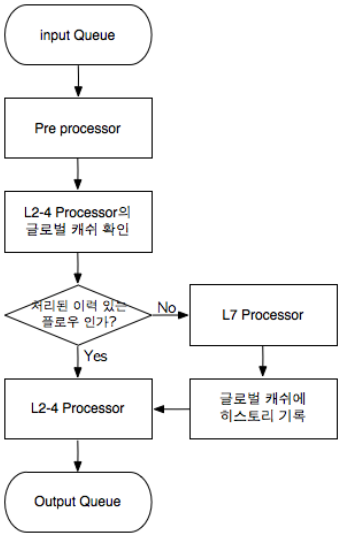

그림 4. 제안된 구조의 패킷 흐름도

우인지 판단한다.

3.a 글로벌 캐시를 참고하여 플로우가 처리된 이력이 없다고 하면, L7프로세서가 패킷을 처리하는 동안 기다리고, 처리가 끝나면 글로벌 캐시의 히스토리에 기록하고 L2~4프로세서는 이력을 참조하여 패킷을 처리한다.

3.b L2-4프로세서가 글로벌 캐시의 이력을 참고하여 과거에 처리된 이력이 있는 플로우로 판단하면, L7프로세서의 처리 과정을 거치지 않고 이력을 이용하여 바로 L2~4프로세서에서 패킷을 처리하게 된다.

4. 과정 3에서 처리한 패킷은 L2-4프로세서에서 트래픽 관리에 사용된다.

5. L2~4프로세서에서 나온 패킷은 출력 큐에 저장이 된다.

Ⅳ. 시뮬레이션

일반적인 구조와 제안된 구조를 SystemC로 설계하고 EEMBC와 SNORT를 이용하여 테스트를 수행하였다. 시뮬레이션에서 사용한 테스트 벡터는 Shmoo Group에서 제공하는 Defcon 9 capture the Flag Data를 이용하여 데이터를 검증하였다.[6,7]

이 실험에는 Intel의 Core2Duo E8400 3GHz와 2*2GB의 메모리, Asus P5K SE 메인보드, 1Gbit의 network interface 등으로 구성되어있다. 운영체제는 Ubuntu Linux 9.04를 사용하였다.

L2~4의 벤치마크 프로그램인 EEMBC는 기본적으로 프로세서의 성능을 측정하는 프로그램이지만, 네트워크 프로세서의 경우 패킷의 처리 시간을 얻을 수 있다. L7의 처리 시간을 측정하는데 사용된 SNORT는 인터넷 트래픽을 감시하고, 패킷의 페이로드 부분을 정해진 룰셋과 비교하여 패킷의 길이당 걸린 시간을 얻을 수 있는 프로그램이다.

4.1 EEMBC 와 SNORT

EEMBC[8]는 버퍼의 크기를 512KB(374 패킷), 1MB(720 패킷), 2MB(1412패킷), 4MB(2824 패킷) 4가지로 나눠서 시뮬레이션이 진행된다. 각 크기에 따라서 한 iteration에서 처리하는 패킷의 양이 달라진다. 10,000번의 iteration을 수행하였을 때 버퍼 사이즈에 따라 그림5와 같이 수행되었다.

SNORT[9]는 기본 룰셋을 적용하여 실험을 진행하였다. L7의 프로세싱은 패킷의 길이, 룰셋의 적용 방법, 패킷에 포함된 데이터의 종류 등에 영향을 받기 때문에 같은 길이의 패킷이 같은 처리 시간을 갖는 것은 아니다. 실험을 통해 구해진 데이터의 신뢰도를 높

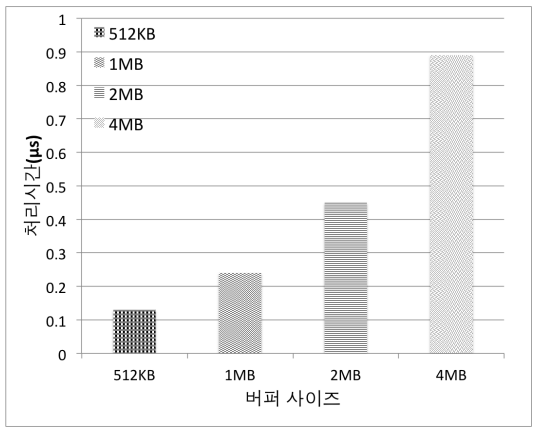

그림 5. L2~4프로세서의 처리 속도를 EEMBC를 통해서 얻은 결과

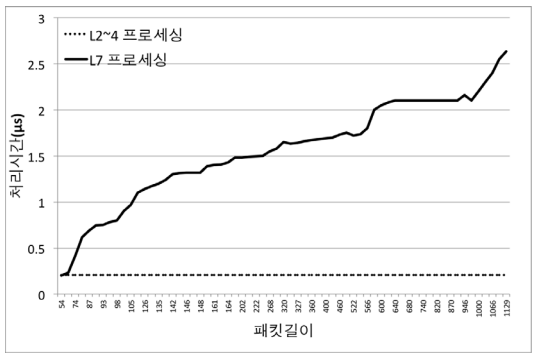

그림 6. L2~4와 L7 패킷 처리시간 비교

이기 위해서 회귀분석(regression analysis)를 이용하여 데이터를 구하였다.

L2~4의 경우 헤더만을 처리하기 때문에 패킷의 길이에 따라서 처리 시간이 변화하지 않고, L7의 경우는 페이로드의 길이와 내용에 따라서 처리시간이 달라짐을 알 수 있다. 하지만 L7의 경우 페이로드가 길어지면서 처리시간이 포화된다. 그림 6은 L2~4와 L7프로세서의 패킷 처리 시간을 비교한 것이다. L2~4와 L7의 처리 시간이 패킷 길이가 커질수록 차이가 많이 발생하는 것을 알 수 있다.

4.2 일반적인 네트워크 프로세서 구조와 제안하는 네트워크 프로세서 구조

일반적인 네트워크 프로세서의 구조는 L2~4, L7의 처리를 하나의 프로세서에서 직렬로 처리하고, 제안하는 구조에서는 병렬로 처리한다는 것이 가장 큰 차이점이다. 하지만 L2~4의 처리시간이 너무 짧아 전체적으로 L7의 처리 시간에 영향을 받는다. 그림 7은 일반적인 네트워크 프로세서에서 사용하고 있는 L2~4와 L7을 직렬로 처리할 때 걸리는 시간과 병렬로 처리할 때의 시간을 비교하였다. 글로벌 캐시를 사용하지 않는다면 시간 차이가 거의 발생하지 않는 것을 알 수 있다.

제안된 구조에서는 동일한 플로우의 패킷이 들어오는 확률을 이용하여 실험하였다. 동일한 플로우가 각각 0%, 25%, 50%, 75%, 100%의 비율로 들어온다고 가정했을 때 결과는 그림 8과 같이 보여진다. 동일한 플로우가 들어오는 비율이 증가함에 따라서 처리시간이 감소하는 것을 알 수 있다. 이는 앞으로의 인터넷 패킷이 IPTV, VOD 등의 스트리밍을 기반으로 한 서비스에서 연속된 패킷이 발생하므로 L2~4와 L7의 처리를 제안된 구조로 처리하는 것이 타당하다는 것을 보여준다.

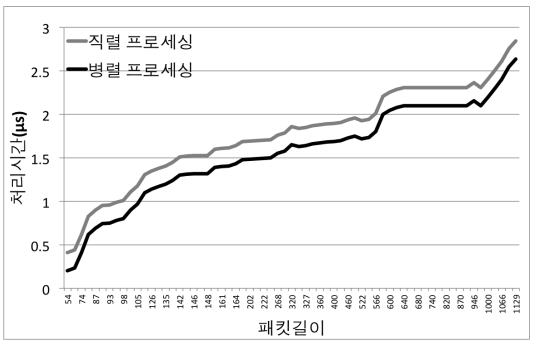

그림 7. 직렬 처리시간과 글로벌 캐시를 사용하지 않은 병렬 처리시간 비교

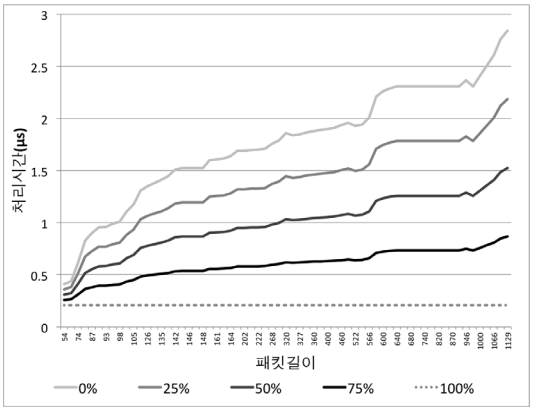

그림 8. 같은 플로우의 패킷이 연속해서 들어올 확률에 따른 L2~4와 L7의 처리 시간

위의 결과에서 1000byte 이상의 길이가 긴 패킷이 연속된 플로우로 들어온다고 가정했을 때 확률에 따라서 각각 29.88%, 85.24%, 222.84%, 1158.92%의 성능 향상효과가 있으므로, IDC자료에서 예상한 50% 정도가 IPTV, VOD등의 스트리밍에서 발생되는 패킷일 때 약 85%의 성능 향상이 있을 것이다. 그림 9는 위의 확률을 로그 스케일을 사용하여 나타낸 것이다.

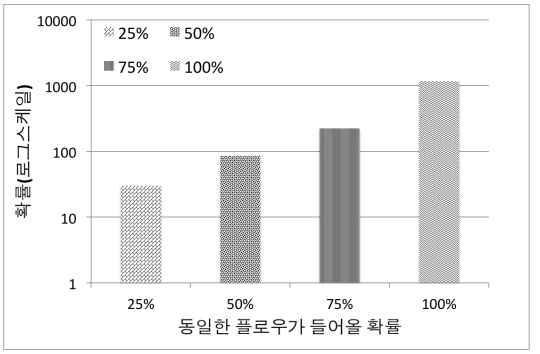

그림 9. 동일한 플로우가 들어올 확률에 따른 시뮬레이션 결과

V. 결 론

증가는 네트워크 트랙픽의 종류와 양을 효율적으로 처리하기 위해 플로우 개념이 도입되었다. 플로우를 이용하여 네트워크 트래픽을 처리하는 현재의 시스템에서 기존의 네트워크 프로세서의 성능으로는 앞으로 증가가 예상되는 네트워크 트래픽을 처리하기에 어려움이 따른다. 따라서 본 논문에서는 앞으로 증가가 예상되는 네트워크 트래픽의 속성과 이를 이용한 보다 진보된 네트워크 프로세서의 구조를 제안한다.

IDC에서 2008년에 발표된 보고서에 따르면 앞으로의 네트워크 트래픽은 IPTV, VOD등의 스트리밍 서비스에 의해서 발생되는 양이 전체의 50% 이상으로 증가할 것으로 예상되고 있다. 이러한 결과는 앞으로의 네트워크 트래픽은 동일한 플로우의 패킷이 연속해서 발생할 수 있음을 보여준다.

앞으로 다가올 인터넷에서 트래픽을 어플리케이션 레이어 까지 보다 효율적으로 처리하기 위해 본 연구에서는 Global Cache를 이용하는 구조를 제안한다. 시뮬레이션에서 연속된 플로우의 패킷이 증가할수록 제안된 구조에서는 네트워크 처리 속도가 빨라짐을 볼 수 있다. 앞으로의 네트워크가 IPTV, VOD등 연속된 플로우의 패킷이 증가하기 때문에 앞으로의 네트워크에서 본 논문에서 제안된 네트워크 프로세서의 구조를 통해서 큰 성능 향상이 기대된다. IDC에서 예상한 결과와 시뮬레이션을 통해서 얻은 결과를 종합해 볼 때 기존에 비해 약 85%의 성능 향상이 이루어졌다.

참 고 문 헌

〔1〕 P. C. Mark A. Franklin, Network processor design: issues and practices, *2nd ed., 2003.*

〔2〕 R. Ennals, et al., “Task Partitioning for Multi-core Network Processors,” *ed, 2005, pp. 76-90.*

〔3〕 “Cisco Visual Networking Index - Forecast and Methodology, *2008-2013,” A Cisco White Paper2009.*

〔4〕 N. Dukkipati, et al., “Processor Sharing Flows in the Internet,” *ed, 2005, pp.271-285.*

〔5〕 T. Wolf, et al., “Predictive scheduling of network processors,” *Computer Networks, Vol. 41, pp.601-621, 2003.*

〔6〕 D. L. Schuff, et al., “Conservative vs. Optimistic Parallelization of Stateful Network Intrusion Detection,” in *Performance Analysis of Systems and software, 2008. ISPASS 2008. IEEE International Symposium on, 2008, pp. 32-43.*

〔7〕 Shmoo Group DEFCON 9 Capture the Flag Data. Available: *http://ictf.cs.ucsb.edu/data/defcon_ctf_09/*

〔8〕 SNORT. Available: *www.snort.org*

〔9〕 EEMBC. Available: *www.eembc.org*

박 재 원 (Jae-won Park) 준회원

2009년 2월 수원대학교 전기공학과 학사

2009년 9월~현재 연세대학교 전기전자공학과 석사 과정

<관심분야> 네트워크 프로세서, MPI, 컴퓨터 아키텍쳐

정 원 영 (Won-young Chung) 정회원

2005년 8월 연세대학교 전기전자공학과 학사

2005년 9월~현재 연세대학교 전기전자공학과 석박통합과

<관심분야> 네트워크 프로세서, MPI, 컴퓨터 아키텍쳐

김 현 필 (Hyun-pil Kim) 정회원

2005년 2월 연세대학교 전기전자공학과 학사

2005년 9월~현재 연세대학교 전기전자공학과 석박통합과

<관심분야> 네트워크 프로세서, SVC, 컴퓨터 아키텍쳐, 멀티미디어 프로세서

이 정 희 (Jung-hee Lee) 정회원

1984년 2월 경북대학교 전기공학과 학사

1990년 2월 경북대학교 전기공학과 석사

1984년~현재 전자통신연구원 옴니 플로우 프로세서 개발팀

<관심분야> 네트워크 자원 관리, 차세대 인터넷, 인터넷 QoS

이 용 석 (Yong-surk Lee) 정회원

1973년 2월 연세대학교 전기공학과 학사

1977년 2월 University of Michigan, Ann Arbor 석사

1981년 2월 University of Michigan, Ann Arbor 박사

1993년~현재 연세대학교 전기전자공학과 교수

<관심분야> 마이크로프로세서, 네트워크 프로세서, 암호화 프로세서, SoC