논문 2007-44TC-9-5

내부 버퍼와 단일 엔트리 캐싱을 이용한 다단계 패킷 분류 가속화 구조

(Fast Multi-Phase Packet Classification Architecture using Internal Buffer and Single Entry Caching)

강 대 인*, 박 현 태*, 김 현 식*, 강 성 호**

(Daein Kang, Hyuntae Park, Hyunsik Kim, and Sungho Kang)

요 약

새로운 인터넷 서비스가 등장하면서 진보된 인터넷 응용 기능을 처리하기 위한 패킷 분류 기능은 라우터의 중요한 동작으로 요구되고 있다. 다수의 패킷 필드를 대상으로 하는 패킷 분류 동작은 복잡하며 상대적으로 많은 시간을 요구하기 때문에 빠른 패킷 분류를 위한 알고리즘과 하드웨어 구조에 대한 연구가 활발히 진행되고 있다. 본 논문에서는 가속화된 패킷 분류 기능을 제공하기 위해 내부 버퍼를 사용한 다단계 패킷 분류 구조를 제안한다. 주소 필드 검색기와 다음 필드 검색기 사이에 내부 버퍼를 사용함으로써 송신 주소와 수신 주소의 검색 시간 차이로 인해 발생하는 지연 시간을 줄일 수 있게 되었다. 또한 동일 IP 주소 헤더 정보를 갖는 연속된 패킷의 입력으로 인한 성능 개선의 저하를 방지하기 위해 단일 엔트리 캐싱을 사용하여 성능 개선을 보장하였다. 제안하는 구조는 간단하며 검색 알고리듬에 국한되지 않고 보편적으로 적용될 수 있는 일반성을 갖고 있다.

Abstract

With the emergence of new applications, packet classification is essential for supporting advanced internet applications, such as network security and QoS provisioning. As the packet classification on multiple-fields is a difficult and time consuming problem, internet routers need to classify incoming packet quickly into flows. In this paper, we present multi-phase packet classification architecture using an internal buffer for fast packet processing. Using internal buffer between address pair searching phase and remained fields searching phases, we can hide latency from the characteristic that search times of source and destination header fields are different. Moreover we guarantee the improvement by using single entry caching. The proposed architecture is easy to apply to different needs owing to its simplicity and generality.

Keywords : Packet Classification, Internal Buffer, Multi-Phase Search, Parallel Search, Caching

Ⅰ. 서 론

패킷 분류 기능은 라우터에 도달하는 패킷들을 정해진 규칙에 따라 분류하는 동작을 말한다. 이러한 동작은 라우터가 접근 제어, 서비스 품질 차별화 (QoS: Quality of service), 가상 사설망(VPN: Virtual private network) 등의 추가적인 네트워크 서비스를 가능하게 하는 기본적인 구성 요소가 된다. 패킷을 분류하기 위해서는 라우터로 도달하는 각 패킷이 패킷 분류를 위한 규칙의 집합과 비교되어야 한다. 각 규칙은 하나 혹은 그 이상의 패킷 헤더 필드, 우선순위나 적용 동작과 같은 관련 값들을 포함하고 있다. 패킷 헤더 필드들은 송신 IP 주소, 수신 IP 주소, 포트 번호, 프로토콜 TCP/IP의 특정 헤더 필드들을 포함한다. 특정 규칙이 포함하고 있는 모든 필드들에 대한 규정을 패킷이 모두 만족할 때 패킷이 규칙에 매칭 되었다고 규정하며, 패킷이 만족하는 규칙이 선택되면 해당 규칙마다 정의된 동작이 적용된다.

* 학생회원, ** 평생회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)
접수일자: 2007년4월11일, 수정완료일: 2007년9월6일

일반적으로 복수개의 필드를 대상으로 하는 패킷 분류 동작은 복잡하고 시간이 많이 드는 동작이며, 규칙 테이블의 수가 증가하고 회선의 속도가 증가함에 따라서 처리 속도는 점차 더욱 중요한 성능 지표가 되고 있다. 라우터가 처리해야 하는 패킷 수의 관점에서 작은 크기의 40바이트의 패킷을 가정함으로써 성능의 한계 조건을 고려할 수 있다. 이 경우, 라우터는 초당 125,000,000개의 패킷을 처리해야 하며, 패킷 처리를 위한 다른 동작들까지 고려하면 하나의 패킷을 분류하는데 걸리는 시간은 8ns에 불과하다. 최근 온 칩 메모리의 접근 속도는 SRAM의 경우, 1~5ns이며, DRAM의 경우, 10ns 정도가 소요되고, 오프 칩 메모리의 경우 더욱 긴 메모리 접근 시간이 필요하다. 이러한 수치는 매우 빠른 속도의 패킷 처리를 위한 알고리듬과 하드웨어 구조의 필요성이 증대되고 있음을 의미한다.[1] 본 논문에서는 기존 연구에서 제시된 다단계 병렬 처리 패킷 구조에 내부 버퍼를 사용하여 처리 속도를 가속화하는 구조를 제안한다.

본 논문의 이후 구성은 다음과 같다. 제안하는 구조와 관련된 기존 연구 내용을 Ⅱ장에서 다루고, Ⅲ장에서는 제안하는 구조를 설명한다. 시뮬레이션 결과와 성능 측정의 결과의 분석은 Ⅳ장에서 다루게 되며, 마지막으로 Ⅴ장에서 논문을 결론짓는다.

Ⅱ. 기존 패킷 분류 구조

일반적으로 패킷 분류 동작은 알고리듬적인 접근 방법과 구조적인 접근 방법으로 나누어 설명할 수 있으며, 각각의 접근방법은 서로의 약점을 보완하는 해결책을 제시하는 대안이 되고 있다.[2] 이 장에서는 하드웨어 구조적인 측면에서의 성능 개선을 가져온 기존 구조들을 살펴보도록 한다.

Michael E. Kounavis 등은 효과적인 패킷 분류를 위한 구현 방향을 제시하였다. 제시하는 방향에 따르면 다차원의 패킷 분류 동작은 두개의 하위 동작으로 나뉘어 처리되어야 한다.[3] 이는 송수신 IP 주소를 대상으로 하는 이차원 검색 동작과 나머지 필드를 대상으로 하는 n-2 차원의 검색 동작을 의미한다. 이는 IP 주소의 검색 동작은 프리픽스 검색을 기반으로 하며, 나머지 필드의 검색은 일반적인 범위 검색의 성격을 갖기 때문이다. 구별되는 성격의 검색 구조에 대해 최적화된 검색 동작을 적용함으로써 전체 성능을 개선할 수 있다. 이와 같은 검색 구조는 다단계(multi-phase) 검색 구조를 통해서 구현될 수 있다. 다단계로 구성되는 패킷 분류 구조는 각 단계를 따라 기존 검색 결과의 집합(chunk)에 대한 검색을 수행한다. 각 단계의 검색 대상에 대한 검색 결과는 해당 패킷에 부합될 수 있는 가능성이 있는 필터들의 부분 집합이므로 각 단계를 지나면서 부합될 수 있는 집합의 크기는 점점 작아지게 된다.[4] 다단계의 검색 구조를 이용한 접근은 사용 메모리의 크기가 큰 반면, 검색 속도의 증가를 얻을 수 있다.

제한된 하드웨어의 동작속도를 가정할 때, 가장 널리 사용되는 성능 개선의 접근 방법은 병렬 처리이며 패킷 분류에 있어서 분해 검색(decomposition technique)은 병렬 처리에 매우 적합한 접근 방식이다. 분해 검색은 복수개의 필드를 대상으로 하는 검색 동작을 하나의 필드를 대상으로 하는 여러 개의 검색 동작으로 분해하여 처리함으로써 추가적인 하드웨어의 병렬 수행을 통해 처리 속도의 향상을 기대할 수 있다.[5] 이러한 처리 구조는 어떤 각 분리된 차원의 헤더 필드가 어떤 규칙에 부합하는지를 지시하는 벡터 값들을 비트별 논리곱을 통해서 간단히 결과를 얻는 병렬 비트 벡터(Parallel bit vector)에서 적용되기 시작하였으며, 유사한 분해 검색의 접근은 재귀적 흐름 제어(RFC: Recursive flow control), 병렬 패킷 분류(P2C: Parallel packet classification)와 같은 방식을 통해 개선되고 발전되었다.[6]

Ⅲ. 제안하는 패킷 분류 구조

1. 하드웨어 구조

패킷 분류 동작의 가속화를 위한 제안하는 구조는 기존 패킷 분류 구조에서 언급된 바와 같이 병렬 검색 구조가 사용되었으며 주소 검색 단계와 다음 검색 단계가 순차적으로 연결되어 동작하는 다단계 패킷 분류 구조를 차용하였다. 주소 검색 단계와 뒤이은 검색 단계 사이에 내부 버퍼를 삽입하여 지연 시간을 감소시키도록 하였다. 내부 버퍼의 삽입에 대한 내용은 다음 절에서 자세히 다루도록 한다.

송수신 IP 주소 필드는 각각 32비트 프리픽스로 구성된 대칭적인 검색 구조를 갖고 있으며, 패킷 분류 동작 중에서 가장 높은 연산과 시간을 요하는 부분이다. 일반적으로 IP 주소 필드를 거치면 해당 패킷을 만족하는 규칙의 수는 다섯 개 내외로 감소하게 된다.[2] 송수신 주소 검색의 성능 개선이 전체적인 패킷 분류 동작의 성능을 향상시키는 가장 중요한 요소이므로 주소 필드

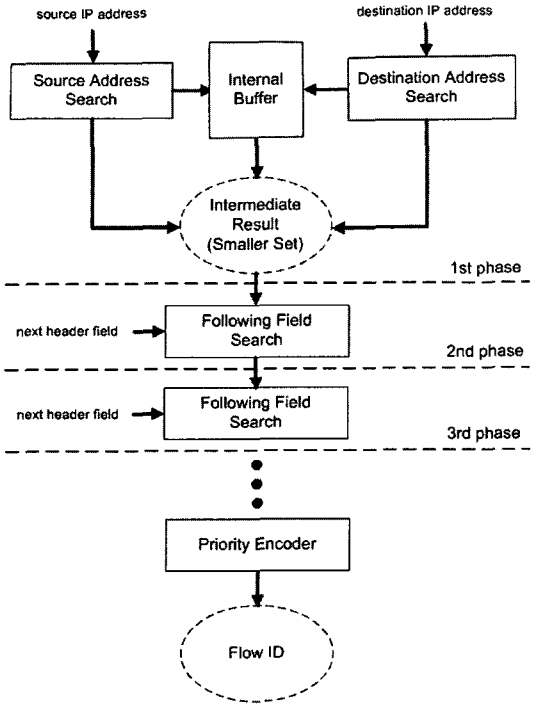


그림 1. 제안하는 구조의 데이터 흐름도
Fig. 1. The data flow diagram of the proposed scheme.

에 대한 병렬처리를 적용하였다.

패킷 분류 동작은 수신 IP주소를 대상으로 가장 길이가 긴 프리픽스를 찾는 일차원 IP 주소 검색(IP Address Lookup)과 달리 여러 개의 필드를 대상으로 하는 다차원 검색이다. 하나의 패킷 헤더 정보가 복수개의 규칙에 만족할 수 있기 때문에 만족하는 규칙들 가운데 상호간의 우선순위를 기준으로 하나의 규칙을 선택하게 된다. IP 주소 필드는 복수개의 전송(transport) 레벨의 필드와 관련되어 있기 때문에 가장 높은 우선순위(priority)의 규칙을 찾기 위해서는 IP 주소에 부합되는 결과들을 전송 레벨에서 다시 검색하여야 한다. 전송 레벨의 필드 검색은 프리픽스 검색을 요하는 IP주소 검색과 달리 완전 일치 검색이나 범위 검색의 성격을 갖기 때문에 별개의 단계로 구분하였다.

이러한 구분은 다른 검색 구조를 갖는 패킷 헤더 필드들에 적합한 다른 검색 알고리듬이나 하드웨어 구조를 적용할 수 있기 때문에 검색 대상에 따른 유연성을 가질 수 있는 구조적인 기반이 된다. 또한 파이프라이닝에 적합한 구조이기 때문에 파이프라이닝을 통해 전체 처리량을 향상 시킬 수 있다. 전송 레벨과 관련된 필터들의 조합은 상대적으로 작기 때문에 n-2 차원의 검색을 위한 작고 특정 동작을 하드웨어로 구현이 가능하다는 점도 주소 검색과 나머지 필드 검색을 분리하는 근거가 되었다. IP 주소검색을 거친 중간 결과 집합이 작기 때문에 송수신 주소 쌍에 대한 검색은 다른 헤더 필드의 검색 보다 앞서 이루어져야 한다.

2. 검색 지연시간 제거(waiting time hiding)

본 논문에서 제안하는 구조는 패킷 분류의 기준이 되는 필터들의 상당수가 IP 송신지 주소와 수신지 주소가 다른 길이의 프리픽스를 갖고 있다는 관찰에 근거하였다. 이는 하나의 패킷에 대해 IP 송신 주소와 수신 주소를 병렬적으로 검색할 때, 검색 시간의 차이가 발생할 수 있다는 것을 의미한다. 본 논문에서는 신뢰성 있는 필터의 집합으로부터 제안하는 구조를 검증하기 위해 ClassBench라는 성능 측정 도구를 사용하였으며, 이를 통해 실제 사용되고 있는 필터의 집합에서 추출한 특성 변수 파일을 사용하여 원하는 크기의 규칙 집합을 생성하고 이에 상응하는 패킷 데이터를 얻었다.[7]

그림 2는 ClassBench로부터 생성한 하나의 특정 패킷 분류 규칙 테이블을 대상으로 각 필터의 IP 송신 주소와 IP 수신 주소의 프리픽스 길이 차의 분포를 나타낸 것이다. 10,000개 정도의 규칙을 갖는 규칙 집합을 대상으로 하였다. 그림 2에 나타난 규칙 집합의 각 필터 당 프리픽스의 길이 차의 평균값은 5.5였으며, 다른 특성 변수 파일을 이용한 11개의 추가적인 실험에서 얻은 필터 당 평균 프리픽스 길이 차는 4.1에서 21.7을 나타냈다.

제안하는 구조는 앞 절에서 기술한 바와 같이 2차원 병렬 주소 검색과 다단계 패킷 분류 구조를 기반으로 하고 있다. 다단계 패킷 분류 구조의 하나의 단계로서의 2차원 병렬 주소 검색이 사용되는 경우, 주소 검색의 중간 검색 결과 값은 송신 주소와 수신 주소의 검색이 모두 완료된 시점에서 다음 검색 단계로 전달되게

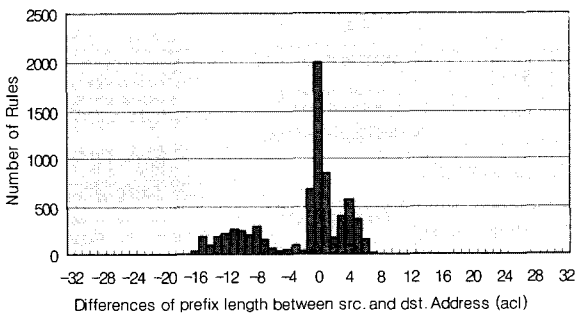


그림 2. 송수신 IP 주소 필드의 프리픽스 길이 차 분포
Fig. 2. The data flow diagram of difference between prefix length of IP address pair.

되는 것이 일반적인 동작이다. 검색이 먼저 완료된 쪽의 주소 검색 동작은 상대적으로 느린 주소 검색 동작이 완료되는 시점까지 동작을 멈추게 되는 지연 시간은 검색 동작이 계속되는 과정에서 축적되어 전체 라우터의 성능 저하를 가져온다. 이러한 문제점을 해결하기 위해서 주소 검색 단계와 이후 필드의 검색 단계 사이에 내부 버퍼를 사용하는 구조를 고안하였다. 내부 버퍼를 사용함으로써 송신 주소 필드의 검색과 수신 주소 필드의 검색은 상대 주소의 검색 완료 여부와 관계없이 연속적으로 수행될 수 있다. 하나의 패킷의 주소 필드 중에서 먼저 검색 연산이 완료된 필드의 중간 결과 값은 버퍼에 저장된다. 이후에 느린 쪽의 주소 검색이 끝나게 되면 양쪽의 중간 검색 결과는 병합되어 다음 필드의 검색 단계로 전달된다. 이러한 일련의 동작을 통해서 검색 시간이 오래 걸리는 쪽의 느린 검색 과정은 감추어질 수 있으며, 이는 버퍼를 통해 중간 결과 값을 저장함으로써 검색 유닛을 중지 없이 사용할 수 있기 때문이다.

그림 3은 2차원 병렬 주소 검색을 기반으로 한 다단계 패킷 분류 구조에서 일어날 수 있는 타이밍 다이어그램을 표현한 것이다. 첫 번째와 두 번째 행은 다단계 검색 구조의 첫 번째 단계에서 수행되는 2차원 병렬 주소 검색의 송신 주소(SA: source address) 검색 시간과 수신 주소(DA: destination address) 검색 시간을 각각 표현한 것이다. 세 번째 행은 주소 검색 이후 단계에서 수행되는 주소 검색 이후의 첫 번째 임의 필드에 대한 검색 시간을 나타낸다. 앞서 기존 연구에서 기술된 바와 같이 주소 검색 이외의 나머지 필드에 대한 검색은 경우의 수가 주소 검색에 비해 현저히 적기 때문에 주소 검색보다 적은 시간이 소요되며, 이를 반영하여 그림 3의 타이밍 다이어그램에서는 각 주소 검색보다 적

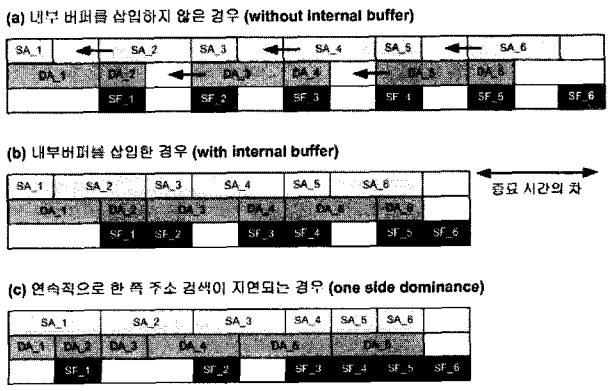


그림 3. 버퍼 삽입에 따른 기대 동작
Fig. 3. Expected operations by buffer insertion.

은 시간이 소요됨을 가정하여 표현하였다. 검색 필드의 약어 뒤에 표시된 인덱스는 들어오는 패킷의 순서를 지시한다.

그림 3의 (a)는 제안하는 내부 버퍼를 사용하지 않은 기존 구조의 타이밍 다이어그램을 나타낸다. 동일한 시간에 병렬적으로 검색을 시작하므로 각 패킷의 검색의 시작 시간이 일치한다. (b)는 (a)와 동일한 패킷이 입력되었을 때, 제안하는 내부 버퍼를 사용한 경우의 타이밍 다이어그램을 표현한 것이다. 내부 버퍼를 사용한 경우, 입력 패킷에 대한 한쪽 검색 동작이 완료된 후에도 멈춤 없이 다음 패킷에 대한 주소 검색이 가능하다. 이로 인해 주소 검색 유닛이 기다리는 시간이 없어지게 되고 연산을 수행하지 않던 시간을 다음 패킷의 연산에 사용할 수 있게 되었다. 결과적으로 다음 패킷의 시작 시점이 이전 패킷의 완료 시점과 갖게 되어 유휴 시간을 절약할 수 있게 되었다. 그림 3의 (c)는 송수신 주소의 한쪽 필드에서 연속적으로 느린 검색 결과를 보이는 경우에 대한 동작을 나타낸 것이다. 연속된 패킷에 대해 한 쪽 주소 필드의 검색 시간이 느린 경우라 할지라도 반대쪽의 주소 검색은 유휴 시간 없이 검색이 진행되고 있다. 이렇게 저장된 시간은 느린 쪽 주소가 바뀜과 동시에 가시화 되어 나타난다.

3. 내부버퍼의 적용

패킷 분류의 동작은 N개의 패킷 분류 규칙이 라우터에 존재한다고 할 때, S개 비트의 패킷 헤더를 T 비트의 인덱스로 치환시키는 문제로 생각할 수 있다. (T = logN, T << S)[4] 그러므로 각 검색 단계를 거치면서 원래의 헤더 필드를 나타내는 비트의 수는 점차 감소한다.

그림 4는 제안하는 구조와 신호 흐름을 나타내고 있다. 2차원 병렬 주소 검색과 다음 단계의 검색 유닛 사

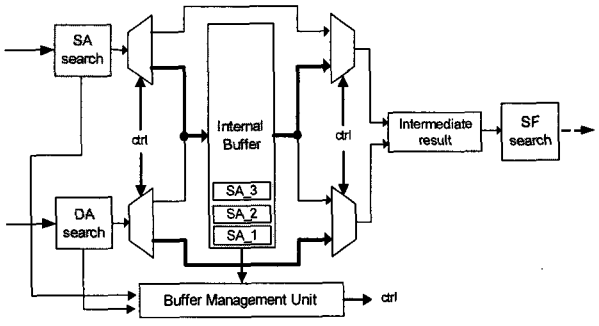


그림 4. 패킷 분류 가속화를 위한 하드웨어 구조
Fig. 4. Hardware architecture of the proposed scheme.

이에 제안하는 내부 버퍼와 버퍼 제어 유닛으로 이루어진다. 추가적으로 두개의 2:1 멀티플렉서와 1:2 디멀티플렉서가 사용된다. 버퍼 제어 유닛은 송수신 주소 검색 유닛과 버퍼로부터 정보를 제공받아서 현재 어떤 주소 검색 유닛이 지연되고 있는지의 상태 정보를 생성하며, 버퍼가 가득 찼는지 비었는지 등의 현재 버퍼 상태를 나타내는 값을 제공하게 된다. 송수신 주소 검색은 독립적으로 이루어지게 되며, 그 검색 결과 값은 검색 시간의 차이가 없는 경우나 상대 IP 주소 검색 결과가 버퍼링되어 있는 경우에는 멀티플렉서를 통해서 직접 다음 검색 단계로 전달되며, 상대 IP주소 검색 보다 미리 검색이 종료된 경우에는 버퍼에 저장된다.

그림 4에서 표현되어 있는 버퍼의 상태를 살펴보면, 세 개의 패킷에 대한 송신 주소 검색의 중간 결과 값이 버퍼에 쌓여있다. 이는 첫 번째 패킷의 수신 주소 검색이 지연되어 세 개의 송신 주소를 검색하는 동안 검색이 완료되지 않았기 때문이며, 네 번째 송신 주소 검색(SA_4)이 완료되기 이전에 첫 번째 수신주소(DA_1) 검색이 완료되는 경우를 가정하면, 첫 번째 수신 주소의 검색 결과는 버퍼 제어 유닛에서 발생한 멀티플렉서 제어 신호에 의해 버퍼를 거치지 않고 그림 3의 굵은 선으로 표시된 아래쪽 신호선을 따라서 전달된다. 이와 동시에 버퍼 제어 유닛은 수신 주소 검색 유닛으로부터 DA_1의 검색 완료를 전달받아서 버퍼에서 첫 번째 송신 주소의 검색 결과 값을 아래쪽 굵은 신호선을 따라 다음 단계로 전달하도록 제어한다. 이와 같은 과정으로 첫 번째 패킷에 대한 각 주소의 검색 결과가 결합되어 다음 검색 단계로 전달되는 동작이 수행된다. 이후 버퍼에 쌓여있는 검색 결과에 해당하는 수신지 주소 검색이 모두 완료되면 버퍼에 쌓인 결과는 하나씩 출력된다. 송신 주소의 검색 결과가 연속적으로 늦어지면 버퍼 제어 유닛에서 발생하는 상태 신호가 반전되게 되고 수신 주소 검색 유닛에서 발생한 중간 검색 결과가 버퍼에 쌓이기 시작한다. 이러한 경우 검색의 버퍼링과 병합은 그림 4의 가는 신호선을 따라 전달된다.

4. 단일 엔트리 캐싱의 적용(single entry caching)

인터넷 트래픽의 지역성에 대한 연구가 진행되고 있으며 트래픽의 IP 주소는 공간적 지역성(Spatial locality)은 존재하지 않으나 시간적 지역성(Temporal locality)은 갖고 있는 것으로 알려져 있다. 시간적 지역성은 라우터의 위치에 따라 정도를 달리하며 백본 라우터 보다는 말단 라우터에서 지역성이 나타난다.[8] 본 논문에서 제안하는 구조는 패킷의 송신 IP 주소와 수신 IP 주소의 검색이 독립적으로 수행될 때 발생하는 검색 시간의 차이 전제하고, 이를 내부 버퍼를 이용하여 극복하고자 하는 것이다. 이러한 접근은 송신 주소와 수신 주소의 검색 시간에 차이를 발생시키는 송신 혹은 수신 IP 주소 검색이 교차되어 발생함으로써 지연시간을 상쇄시킨다는 가정 하에 전개되었다. 다음 장에서 살펴볼 실험 결과에서 확인할 수 있듯이 내부 버퍼를 이용한 지연 시간의 감소는 트래픽의 지역성과 관련이 있으며 지역성이 클수록 성능의 개선 정도는 감소하게 된다. 이는 동일한 송수신 IP 주소를 갖은 패킷이 연속적으로 라우터에 들어오는 경우, 검색 시간이 오래 걸리는 한 쪽 IP주소의 검색 시간이 연속적으로 누적되어 휴지 시간의 교차 상쇄를 저해하는 요소로 작용하기 때문이다.

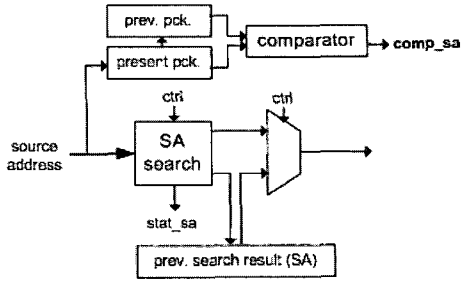


그림 5. 단일 엔트리 캐싱 (송신지 주소 검색부)
Fig. 5. The single entry caching (source address search).

이러한 문제를 해결하기 위해 동일한 송신 혹은 수신 IP주소를 갖는 패킷이 연속적으로 들어오는 경우, 직전의 검색 결과를 저장하여 사용하기 위한 단일 엔트리 캐싱(Single entry caching)을 도입하였다. 단일 엔트리 캐싱은 이전 패킷의 IP 주소 헤더 정보와 검색 결과를 저장하였다가 새로 들어온 패킷의 정보와 비교하여 주소 헤더가 일치하면 검색 과정을 중단하고 저장된 검색 결과를 다음 검색 단계로 전달한다. 이를 통해 동일한 검색 동작을 생략하여 검색 지연 시간의 누적을 방지하도록 하였다.

그림 5는 단일 엔트리 캐싱을 위한 하드웨어 구조를 나타낸다. 이전 패킷과 현재 패킷의 비교를 위한 두 개의 레지스터와 비교기가 사용되었고 이전 검색 결과를 저장하기 위한 레지스터와 입력 값을 선택하기 위한 멀티플렉서가 추가되었다. 주소 정보는 검색 유닛과 비교기 입력으로 동시에 들어가며, 직전 주소 정보와 일치하는 경우, 비교기의 결과 값(comp_sa)은 버퍼 제어 유닛의 입력 값으로 전달된다. 버퍼 제어 유닛의 제어 신

호(ctrl)는 검색 유닛의 동작을 중지 시키고 멀티플렉서가 이전 검색 결과를 선택하도록 제어한다. 수신지 주소 검색부도 그림 6과 동일한 하드웨어 구성을 갖는다.

IV. 실험 및 성능평가

패킷 분류를 위한 규칙 테이블은 ClassBench에서 제공하는 접근 제어 리스트(acl: Access control list), 방화벽(fw: Firewall), IP 체인(ipc: IP chain)의 seed-file을 통해 생성하였으며. 성능의 비교는 제안하는 내부 버퍼를 사용하는 구조와 내부 버퍼를 사용하지 않는 기존 구조를 대상으로 수행되었다. 실험 결과를 바탕으로 타당한 내부 버퍼의 크기를 제안하였으며, 단일 엔트리 캐싱를 추가하였을 때의 성능 개선 정도를 내부 버퍼만 사용하였을 때와 비교 기술하였다.

1. 성능 비교

실험에서는 기본적인 프리픽스 검색 구조인 1-Bit Trie와 보다 적은 최대 메모리 접근 횟수를 보장하는 이진 프리픽스 트리(BPT: Binary prefix tree)[9]를 사용하였다. 표 1에서는 검색에 필요한 메모리 접근 횟수를 비교함으로써 성능을 비교하고 있다. 약 10,000개의 규칙을 갖는 여섯 개의 규칙 테이블을 추출하였으며, 이에 상응하는 패킷 데이터를 100,000개 생성하여 실험하였다. 복수 개의 필드로 구성되는 각각의 규칙에서 송신 주소와 수신 주소만을 분리하여 2차원 주소 병렬 주소 검색에 사용하였다. 각 규칙들은 여러 개 필드의 조합으로 이루어져 있으므로 추출한 송신 주소와 수신 주소의 엔트리는 원래 전체 규칙 테이블의 엔트리 개수인 10,000 개 보다 적다. 송, 수신 주소의 분리된 규칙의 수는 표 1의 네 번째 행에 표시되어 있다.

성능 실험의 결과는 표 2에서 보는 바와 같이 검색 알고리듬으로 이진 프리픽스 트리를 사용한 경우에는 1.13%에서 45.36%까지의 시간 단축이 있었고 표 3에서

표 1. 패킷 분류 규칙 테이블 정보
Table 1. Rule table characteristics.

	규칙 개수	송신 주소 규칙 개수	수신 주소 규칙 개수	패킷 개수
DB1	9,836	5,489	902	98,360
DB2	9,351	1,840	1,073	93,564
DB3	9,404	3,408	6,734	95,146
DB4	8,833	3,839	6,759	88,331
DB5	8,815	3,325	5,399	88,716
DB6	10,000	4,678	8,874	100,003

표 2. BPT를 사용한 경우의 비교 실험 결과
Table 2. Simulation results using BPT search algorithm.

BPT	평균 메모리 접근횟수		전체 검색 시간 (메모리 접근 횟수)		감소율 (%)
	송신 주소	수신 주소	기존 구조	제안 구조	
DB1	5	5	642,767	545,021	15.21
DB2	9	12	1,237,088	1,131,759	8.51
DB3	12	16	1,614,485	1,596,187	1.13
DB4	11	12	1,482,056	1,142,610	22.90
DB5	9	14	1,338,725	1,317,199	1.61
DB6	31	32	5,906,610	3,227,454	45.36

표 3. 1Bit-Trie를 사용한 경우의 비교 실험 결과
Table 3. Simulation results using 1B-Trie.

1-Bit Trie	평균 메모리 접근횟수		전체 검색 시간 (메모리 접근 횟수)		감소율 (%)
	송신 주소	수신 주소	기존 구조	제안 구조	
DB1	30	28	3,052,965	2,965,200	2.87
DB2	22	26	2,725,485	2,461,895	9.67
DB3	13	22	2,793,055	2,152,994	22.92
DB4	13	21	2,222,648	1,886,721	15.11
DB5	13	19	2,526,483	1,723,310	31.79
DB6	14	28	3,151,002	2,810,030	10.82

와 같이 1Bit-Trie를 사용한 경우에는 2.87%에서 31.79%까지의 시간 단축이 있었다.

2. 버퍼 크기의 결정

내부 버퍼 크기의 추정은 라우터의 입력 버퍼가 최대치에 도달하기 전에 제안하는 내부 버퍼가 최대치에 도달해서는 안 된다는 가정 하에서 시작한다. 내부 버퍼의 크기가 감당할 수 있는 최대치 도달이 라우터의 입력 버퍼 최대치에 도달 이전에 발생한다면 성능 향상을 위한 추가적인 내부 버퍼가 기존의 구조의 처리량을 저해하는 요소로 작용하는 경우에 해당하기 때문이다. 때문에 버퍼 크기의 최대값을 계산하기 위해서는 제안하는 구조에서의 이득이 발생하지 않는 최악의 경우를 고려해야 한다. 이러한 경우는 송수신 주소 중에 한쪽의 주소 검색이 한 번의 메모리 접근으로 이루어지고 다른 한쪽 주소의 검색이 32번(1B-Trie 알고리듬을 가정)의 메모리 접근으로 이루어지는 경우가 연속되는 경우이다. 라우터의 입력 버퍼의 크기는 일반적으로 200ms를 보장할 수 있어야 한다. 200ms 동안에 들어오는 패킷의 양은 회선의 속도를 10Gbps로 가정하였을 때 250Mbyte에 해당하며 평균 패킷의 크기를 1500byte

표 4. 버퍼 크기 시뮬레이션 결과
Table 4. Simulation result of buffer sizing.

	최대 필요 버퍼 크기 (Kbytes)	
	BPT	1-Bit Trie
DB1	80.2	10.6
DB2	65.9	22.8
DB3	46.7	152.7
DB4	26.0	46.9
DB5	48.5	44.0
DB6	21.6	90.1

라 가정하면 166K개의 패킷이 라우터의 입력 버퍼에 저장되는 시간이다. 제안하는 내부 버퍼에 저장되는 정보는 패킷 헤더 중의 송수신 IP 주소 정보만 저장되게 된다. 게다가 32비트로 표현되는 주소 정보 또한 규칙 테이블에서 추출된 1차원 검색의 가짓수를 표현할 수 있는 만큼의 비트 수로 인코딩되어 버퍼에 저장되므로 제안하는 내부 버퍼에 저장되는 정보의 크기는 패킷 전체의 데이터와 비교할 때 매우 작다. 166K개의 패킷에 해당하는 정보가 내부 버퍼에 쌓이기 위해서는 260Kbyte의 저장 공간이 필요하다. 260Kbyte의 최대 추정치도 온 칩에서 구현하기에 큰 무리는 없으나 이론적인 상한 값을 전제로 하고 실제 데이터를 이용한 시뮬레이션을 통해서 버퍼 크기를 실험하였다.

실험에서는 버퍼의 허용치를 초과하여 발생할 수 있는 지연 시간을 가정하지 않고, 고정된 버퍼의 크기를 제한하지 않음으로써 라우터가 휴지 없이 동작하는 것을 가정하였다. 생성한 입력 값에 대해 필요한 최대 버퍼 크기는 실험 결과는 표 4에서 필요 버퍼 크기로 표기된 항목에서 확인할 수 있다. 실험을 통해 표 4에서 보는 바와 같이 이진 프리픽스 트리의 경우에는 21.6Kbyte에서 80.2Kbyte, 1-Bit Trie의 경우에는 10.63Kbyte에서 152.7Kbyte 사이의 최대 필요 버퍼 크기를 얻었다. 이는 이 절의 초기에 추정한 최대 필요 버퍼크기인 260Kbyte 보다 훨씬 작은 양의 크기이다.

2. 단일 엔트리 캐싱을 통한 성능 개선

제안하는 구조에서 성능 개선의 정도 차이를 야기하는 요인은 규칙 테이블 자체의 규칙보다 규칙의 적용을 받는 패킷의 입력 순서가 더욱 지배적인 영향 미친다. 연속되는 패킷이 동일한 IP 주소 헤더 정보를 갖는 경우에는 병렬 검색의 관점에서 명백히 지연 시간이 누적됨을 알 수 있다. 이러한 지연 누적 현상을 방지하기 위한 단일 엔트리 캐싱을 적용하였으며, 캐싱 동작이 적

표 5. 단일 엔트리 캐싱의 적용 결과
Table 5. Simulation result of inserting single entry caching.

	검색 시간 (메모리 접근 횟수)		개선율 (%)
	캐싱 사용 안함	캐싱 사용	
DB1	3,052,965	2,823,709	4.77
DB2	2,725,485	2,310,148	6.16
DB3	2,793,055	1,941,470	9.82
DB4	2,222,648	1,769,092	6.23
DB5	2,526,483	1,648,636	4.33
DB6	3,151,022	2,649,037	5.73

용되어 걸리는 지연 시간은 3 클럭 싸이클로 설계하여 실험하였다. 이는 버퍼 제어 유닛의 입출력 지연, 현재 패킷 헤더의 입력 지연을 고려한 값이다.

ClassBench의 패킷 생성 도구를 사용하여 지역성이 없는 데이터와 지역성을 갖는 데이터를 생성하여 비교하였다. 지역성의 정도는 Pareto 분포함수를 이용하여 제어되며 확률 밀도 함수는 아래 수식 (1)과 같다.

$$P(x) = \frac{ab^a}{x^{a+1}} \quad (1)$$

지역성이 없는 경우는 수식 (1)에서 a=1, b=0의 값을 입력하였고 지역성을 갖는 패킷 데이터는 a=1, b=0.01의 값을 입력하여 생성하였다. 검색 알고리듬은 1-Bit Trie를 사용하였다.

표 5에서 보는 바와 같이 캐싱을 사용한 경우 사용하지 않은 경우에 비해 4.77%에서 6.23%의 속도 향상을 얻을 수 있었다. 동일한 IP 주소 헤더 정보를 갖는 연속된 패킷이 라우터로 들어오는 빈도가 커질수록 비례하여 검색 속도는 향상된다. 라우터의 위치에 따라 패킷 데이터의 지역성 정도는 다르기 때문에 캐싱으로 인한 성능 향상을 일반화하기는 어렵다. 하지만 적은 하드웨어 비용으로 같은 IP 주소 헤더 정보를 갖는 연속된 패킷에서 발생하는 명백한 지연시간을 방지할 수 있음을 확인하였다.

V. 결 론

본 논문에서는 빠른 패킷 분류를 위한 하드웨어 구조를 제안하고 있다. 실험을 통해 2차원 병렬 주소 검색과 이후 필드의 검색 단계 사이에 내부 버퍼를 삽입하여 지연 시간을 줄임으로써 검색 속도를 가속화하였으며, 검색 시간을 대변하는 메모리 접근 횟수는 규칙 테이블에 따라 1.13%에서 45.36%의 감소가 있었다. 또한

같은 주소 헤더 필드를 갖는 연속된 패킷에서 발생할 수 있는 지연시간을 단일 엔트리 캐싱을 사용하여 방지함으로써 추가적인 성능 개선을 보장하였다. 본 구조는 2차원 병렬 주소 검색에서 각 주소 검색에 필요한 시간이 다를 수 있는 알고리듬이라면 제한 없이 적용될 수 있는 장점을 갖고 있으며, 추가되는 하드웨어가 합리적인 크기의 버퍼와 간단한 제어 회로만을 요구하므로 적은 하드웨어 비용으로 적용이 가능하다. 실험을 통해 추정한 버퍼크기는 10,000개 정도의 규칙 크기를 갖는 규칙 테이블에 대해 10.6Kbyte에서 152.7Kbyte의 저장 공간이 필요하다는 것을 실험을 통해 입증하였다.

참 고 문 헌

[1] X. Sun, S. Sahni, and Y. Zhao, "Packet Classification Consuming Small Amount of Memory," *IEEE/ACM Trans.* Networking, 2005.

[2] David E. Taylor, "Survey and Taxonomy of Packet Classification Techniques," Tech. Report WUCSE-2004-24, Department of CSE, Washington University in St. Louis, 2004.

[3] M.E.Kounavis, A.Kumar, H. Vin, R. Yavatkar and A. T. Campbell, "Directions in Packet Classification for Network Processors", In Proceedings of Second Workshop on Network Processors (NP2), Feb. 2003.

[4] P. Gupta and N. McKeown, "Packet classification on multiple fields," in Proc. ACM SIGCOMM, Comput. Commun. Rev., vol. 29, Sep. 1999, pp. 147 - 160.

[5] V. Srinivasan, S. Suri, G. Varghese, and M. Waldvogel, "Fast and Scalable Layer Four Switching," in ACM Sigcomm, June 1998.

[6] J. van Lunteren and T. Engbersen, "Fast and scalable packet classification," IEEE Journal on Selected Areas in Communications, vol. 21, pp. 560 - 571, May 2003.

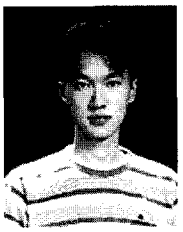
[7] D. E. Taylor and J. S. Turner, "ClassBench: A Packet Classification Benchmark," Tech. Rep. WUCSE-2004-28, Department of Computer Science & Engineering,Washington University in Saint Louis, May 2004.

[8] K. Rajan and R. Govindarajan. A Heterogeneously Segmented Cache architecture for a packet forwarding engine. In Int. Conf. on Supercomputing, 2005.

[9] N.Yazdani and P.S.Min, "Fast and Scalable Schemes for the IP Address Lookup Problem," Proc. IEEE HPSR2000, pp 83-92, 2000.

저 자 소 개

강 대 인(학생회원)
2005년 연세대학교 전기전자 공학과 학사 졸업
2007년 연세대학교 전기전자 공학과 석사 졸업
<주관심분야 : 고속네트워크 관련 SoC설계 및 네트워크 프로세서 설계>



박 현 태(학생회원)
2004년 연세대학교 전기전자 공학과 학사 졸업
2006년 연세대학교 전기전자 공학과 석사 졸업
2007년 현재 연세대학교 전기 전자공학과 박사 과정
<주관심분야 : 고속네트워크 관련 SoC설계 및 네트워크 프로세서 설계>

김 현 식(학생회원)
2006년 연세대학교 전기전자공학과 학사 졸업
2007년 현재 연세대학교 전기전자공학과 석사 졸업
<주관심분야 : 고속네트워크 관련 SoC설계 및 네트워크 프로세서 설계>

강 성 호(평생회원)
1986년 서울대학교 제어계측 공학과 학사 졸업
1988년 The University of Texas, Austin 전기 및 컴퓨터공학과 석사 졸업
1992년 The University of Texas, Austin 전기 및 컴퓨터공학과 박사 졸업
1992년 미국 Schlumberger Inc. 연구원
1994년 Motorola Inc. 선임 연구원
2007년 현재 연세대학교 전기전자공학과 교수
<주관심분야 : SoC 설계 및 응용, DFT, SoC 테스트>