2002년 한국정보처리학회 춘계학술발표논문집 제9권 제1호

고속의 패킷 포워딩 엔진을 위한 병렬 IP 주소 검색 기법

박재형*, 장익현**, 김진수***
*전남대학교 전자컴퓨터정보통신공학부
**동국대학교 정보통신공학과
***건국대학교 컴퓨터.응용과학부
e-mail:hyeoung@chonnam.ac.kr, ihjang@dongguk.ac.kr,
jinsoo@kku.ac.kr

A Parallel IP Address Lookup Scheme for High-Speed Packet Forwarding Engine

Jaehyung Park*, Ikhyeon Jang**, Jinsoo Kim***
*Department of Electronics, Computer, and Information Engineering, Chonnam National University
**Department of Information Communication Engineering, Dongguk University
***Division of Computer and Applied Science, Kunkuk University

요약

포워딩 엔진은 외부 인터페이스를 통해서 들어오는 패킷에 대해서 IP 주소를 기반으로 목적지로 향하는 다음 홉을 결정한다. 이러한 고속의 패킷 처리를 위한 포워딩 엔진을 설계함에 있어서 IP 주소 탐색은 중요한 요인이다. 본 논문에서는 검색경로 압축 트라이에 기반한 IP 주소 탐색 알고리즘을 병렬화하는 기법을 제시한다. 제시된 병렬화를 통해서 IP 주소 탐색의 메모리 접근 횟수를 줄일 수 있으며, 고성능의 패킷 포워딩 엔진에 적용할 수 있다.

1. 서론

인터넷을 통한 서비스 및 응용 프로그램이 다양해짐에 따라서 인터넷 사용자 수가 증가하여 트래픽의 양이 급격하게 증가하고 있다. 전달 매체의 발달로 트래픽을 전달하는 링크는 높은 대역폭과 고속의 전송이 가능하게 됨에 따라서, 패킷을 처리하는 라우터가 인터넷 망의 성능에 병목이 되고 있다[5].

라우터는 라우팅 제어 프로세서, I/O 인터페이스들, 포워딩 엔진과 그들간의 연결을 지원하는 스위칭 패브릭으로 구성되어 있다. 그 중 패킷 전달을 담당하는 포워딩 엔진은 라우터에 있어서 중요한 역할을 수행한다. 포워딩 엔진은 입력 인터페이스를 통해서 들어오는 패킷을 목적지와 출력 인터페이스가 결합된 포워딩 테이블을 참조하여 패킷을 전달한다. 이러한 과정이 IP 주소 검색이며, 목적지 주소와 가장 길게 일치하는 프리픽스를 찾는 작업이다[2].

고속의 IP 주소 검색을 위해서 많은 기법들이 제시되었다. 그러한 기법 중에 Patricia Trie[6]는 radix trie의 특별한 형태로서 현존하는 라우터에 구현되었다. Patricia Trie는 하나의 자식 노드를 갖는 노드가 없는 검색경로 압축 트라이로써, 검색시에 재귀적인 후진탐색이 필요하여 $O(W^2)$번의 메모리 접근이 요구된다, 이 때 W는 트라이의 최대 높이이다. 이와 같은 재귀적인 후진탐색을 피하기 위해서 동적인 프리픽스 트라이가 제시되었으며[1], 이 기법의 메모리 접근 횟수는 $O(W)$이다. 고속의 IP 주소 검색을 위해서 Patricia Trie의 변형들이 계속 연구 중이다.

본 논문에서는 고속의 패킷 포워딩 엔진을 위한 병렬 IP 주소 검색을 지원하는 구조를 제시하고, 검색

경로 압축 트라이 알고리즘을 병렬화하는 기법을 제시한다. 제시된 기법은 검색경로 압축의 특성을 이용함으로써 IP 주소 검색 시에 메모리 참조 횟수를 줄일 수 있다. 메모리 접근 횟수를 줄임으로써 고속의 패킷 포워딩 엔진에 적용될 수 있다.

2. 라우터의 포워딩 엔진

포워딩 엔진은 라우터에 입력되는 패킷의 목적지로 향하는 인터페이스로 패킷을 전달하는 역할을 수행한다. 포워딩 엔진에서의 IP 패킷 처리 흐름은 다음과 같다. 들어오는 패킷에서 IP 헤더 정보를 추출하고 헤더의 무결성을 검사하고 포워딩 테이블에서 출력 인터페이스에 대한 정보를 구한 다음 패킷을 변경하여 전달하는 순서로 처리한다.

포워딩 엔진에서 패킷을 처리함에 있어서 성능에 큰 영향을 미치는 요소는 포워딩 테이블을 검색하는 IP 주소 검색이다. 고속의 패킷 처리를 지원하는 패킷 포워딩 엔진은 IP 주소를 검색하는 검색 엔진이 따로 구성된다[4]. 대부분의 포워딩 엔진은 트라이 기반의 메모리로 구성된 검색 엔진으로 구현된다.

3. 검색경로 압축 트라이

트라이는 트리와 유사한 자료 구조이며[3], 하나 이상의 자식 노드를 갖는 노드로 구성된다. 특정 키에 대한 검색은 트라이의 루트에서 시작하여 가장 긴 것과 일치하는 것을 찾기 위해서 내려가면서 탐색한다. 각 노드에서는 특정 키의 일부분을 이용하여 다음에 탐색하여야 할 노드를 결정한다. 그림 1은 0000, 0001, 001*, 1000, 1001, 11*의 6개의 프리픽스로 구성된 이진 트라이의 예를 보여준다.

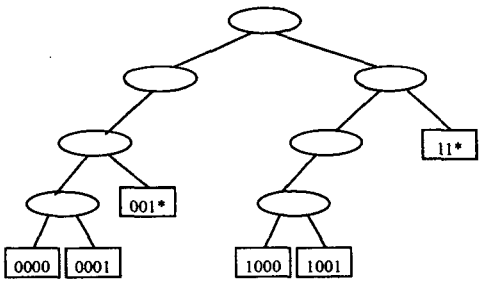

그림 1 이진 트라이의 예

검색경로 압축 트라이는 트라이를 기반으로 구성되며, 그림 1의 예에 대한 검색경로 압축 트라이는 그림 2와 같이 표현된다. 하나의 자식 노드를 갖는 노드를 제거하기 위해서 각각의 가지 노드에 비트 번호가 부가된다. 이렇게 함으로써 원래의 이진 트라이의 높이보다 더 낮은 높이를 유지할 수 있다. 즉, 특정 키에 대한 탐색시에 메모리 접근 횟수를 줄이는 요인이 된다. 검색경로 압축 트라이의 노드에 부가된 비트 번호는 가지 노드에서 탐색시에 검사해야할 비트 번호를 의미한다.

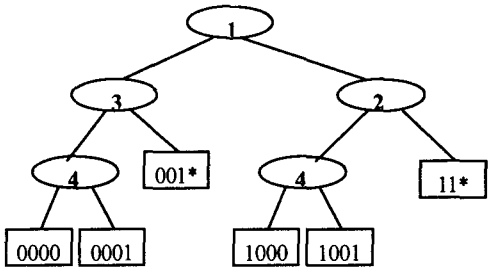

그림 2 검색경로 압축 트라이의 예

4. 병렬 IP 주소 검색 기법

본 절에서는 패킷 포워딩 엔진에서 패킷 처리시에 병목 요인이 되는 검색 엔진의 병렬 구조를 제안한다. 병렬 검색 엔진의 구조는 그림 3에서 보는 바와 같이 제어기와 트라이 기반의 메모리와 선택기로 구성된 N개의 병렬 엔진으로 구성되어 있다.

제어기는 특정 키에 대한 탐색을 수행하는 것은 물론 트라이의 노드에 대한 추가 및 삭제 기능도 수행한다. 기본적으로 제어기는 트라이의 순회 명령을 통해서 탐색, 추가, 삭제 기능을 수행한다. 트라이 기반 메모리는 검색경로 압축 트라이를 구성하는 키들을 포함하는 노드를 포함하고 있다. 그리고 선택기는 특정 키의 비트 패턴으로 자신의 메모리에 존재하는지 그렇지 않은지를 판단한다. 만약 특정 키가 자신의 메모리에 존재할 것이라고 판단되면 선택기는 제어기를 구동시켜서 해당 기능을 수행한다.

IP 주소 검색을 병렬화하기 위해서는 특정 키로 이루어진 노드들을 여러개의 병렬 엔진의 메모리에 분배를 시켜주어야 한다. 이러한 분배 기법은 특정 키가 해당 병렬 엔진의 메모리에 있는지를 검사하는 선택기의 로직과 관련된다.

본 논문에서는 임의의 k개의 비트를 이용하여 해당 병렬 메모리에 분배하는 방법을 제시한다, 이 때 N은 2^k이다. 포워딩 테이블에 노드로 구성될 IP 주소의 특정 비트별로 구분하는 방법으로 선택기는 그림 4에서 보는 바와 같이 구성된다.

5. 산술적인 결과

본 절에서는 우선 IP 주소의 특성에 대해서 살펴보자. IP 주소는 크게 4개의 클래스로 구분된다. 클래스 A, B, C는 일반적으로 사용하는 주소이고, 클래스 D는 멀티캐스트를 위해서 특별한 목적으로 사용하는 주소이다. 본 논문에서는 클래스 D에 대한 내용은 제외한다. 왜냐하면 대부분의 라우터에서 멀티캐스트 용 라우팅 테이블이 따로 관리되기 때문이다. 각각의 주소의 특성은 그림 5에 보여진 것과 같이 총 32비트

클래스	형식
Class A	0nnnnnnnhh...h
Class B	10nnnnnnnnnnnnnnhh...h
Class C	110nnnnnnnnnnnnnnnnnnnnnnhh...h
Class D	1110mm...m

표 1 IP 주소 클래스

중에서 앞쪽의 비트는 클래스를 나타내는데 쓰이고 나머지 비트들이 IP 주소의 프리픽스로 쓰인다.

IP 주소 형태는 표 1에서 보여주는 바와 같이 클래스 별로 클래스를 지칭하는 비트들과 네트워크 주소, 그리고 호스트의 주소로 표현된다. 각각 클래스 별로 네트워크 주소를 나타내는 비트가 7, 14, 21이다.

우선 CIDR 환경에서 현존하는 라우터의 IP 주소프리픽스의 특성을 알아보기로 하자. 그림 5는 네트워크 엑세스 포인트 중의 중요한 3지점인 MaeEast, AADS, PacBell에서 추출한 프리픽스의 분포이다. 대부분의 IP 주소 프리픽스가 24 비트에 포함된다.

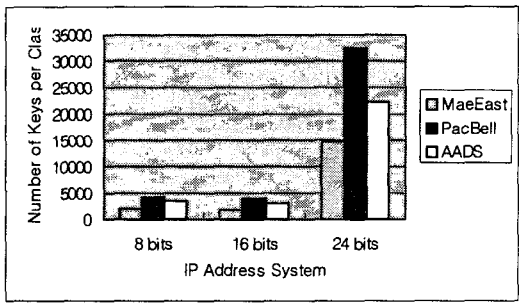

그림 5. IP 주소 프리픽스의 분포

선택기에서 해당 병렬 검색 엔진에 존재하는지를 검사하는 임의의 비트가 IP 주소가 어느 클래스인지를 구별하는 필드- 클래스 A일 경우 첫 번째 비트, 클래스 B일 경우 두 번째 비트까지, 클래스 C일 경우 세 번째 비트까지 -를 포함하는 경우를 고려하자. 이 경우에는 클래스 A와 B에 해당하는 IP 주소 프리픽스의 개수가 상당히 적으므로 특정 병렬 엔진의 검색경로 압축 트라이 구조의 메모리에는 상당히 많은 수의 노드가 포함되고 다른 엔진에는 상당히 적은 수의 노드가 포함되게 되어 메모리의 사용의 불균형을 초래한다.

선택기에서 검사하는 임의의 비트가 IP 주소의 프리픽스의 뒷 부분에 위치하는 경우- 8 비트 프리픽스일 경우 8에서부터 31까지, 16비트 프리픽스일 경우 16에서 31까지, 24 비트 프리픽스일 경우 24에서 31까지 -를 고려하자. 이러한 경우에는 선택하는 가장 마지막 비트까지 확장을 해야한다. 즉, 마지막 선택기 비트가 16이라면, 8비트 프리픽스를 갖고 있는 모든 노드를 16비트로 확장을 하여 노드의 수가 2^8개 증가하는 방식의 확장이 필요하다.

앞에서 설명한 두 가지 경우의 단점을 피하기 위해서는 클래스를 표시하는 비트 - 처음 4개의 비트 - 다음부터 임의의 k개를 프리픽스 확장을 줄이는 비트들로 선택하는 방식이다. 즉, 5번째 비트부터 순서적으로 $(5+k)$번째 비트까지 선택하는 방법으로 선택기 로직을 구현한다. 모든 IP 프리픽스에 대해서 검색할 확률이 동일하다고 가정한다면, 모든 병렬 검색 엔진이 선택될 확률 또한 동일하고 메모리 접근 횟수는

$$\frac{1}{N}\sum_{i=0}^{N-1} W_i < = \frac{1}{N}\sum_{i=0}^{N-1}(W-k) = W-k$$

이다, 이 때, W_i는 병렬 검색 엔진 i의 메모리에 구성된 검색경로 압축 트라이의 최대 높이이고, W는 하나의 검색경로 압축 트라이로 구성하였을 때의 최대 높이이다. 각 병렬 검색 엔진의 트라이는 하나의 검색경로 압축 트라이에서 k비트를 압축한 트라이이므로 병렬 검색 엔진 i의 최대 높이는 $(W-k)$보다 적거나 같다.

한번의 IP 주소 검색을 위해서 접근하는 메모리 접근 횟수가 k만큼 감소함을 알 수 있다.

6. 결론

본 논문에서는 고속의 패킷 포워딩 엔진을 설계하는데 있어서 병목 요인이 되는 IP 주소 검색을 병렬로 수행할 수 있는 구조와 기법에 대해서 제시하였다. 검색경로 압축 트라이에 기반하여 병렬 IP 주소 검색 기법을 제시하였으며 검색경로 압축 트라이의 본질적인 특성을 통해서 IP 주소 검색을 위한 메모리 접근 횟수를 감소시켰다.

참고문헌

[1] W. Doeringer, G. Karjoth, and M. Nassehi, "Routing on Longest Matching Prefixes",

IEEE/ACM Transaction on Networking, vol.4, pp.86-97, Feb. 1996.

[2] V. Fuller, T. Li, J. Yu, and K. Varadhan, " Classless Inter-Domain Routing (CIDR) and Address Assignment and Aggregation Strategy", RFC1519, Sep. 1993.

[3] E. Horowitz and S. Sahni, "Fundamentals of Data Structures in C", Computer Science Press, 1993.

[4] MMC Networks, "EPIF4-L3 Reference Manual", 1998.

[5] S. Keshave and R. Rharma, "Issues and Trends on Router Design", IEEE Communication Magazine, vol.36, no.5, pp.144-151, May 1998.

[6] D. Morrison, "PATRICIA-Practical Algorithm To Retrieve Information Coded In Alphanumeric", Journal of ACM, vol.5, no.4, pp.514-534, Oct. 1968.

[7] S. Nilsson and G. Karlsson, " IP-Address Lookup using LC-Tries", Journal of Selected Areas in Communications, vol.17, pp.1083-1092, Jun. 1999.

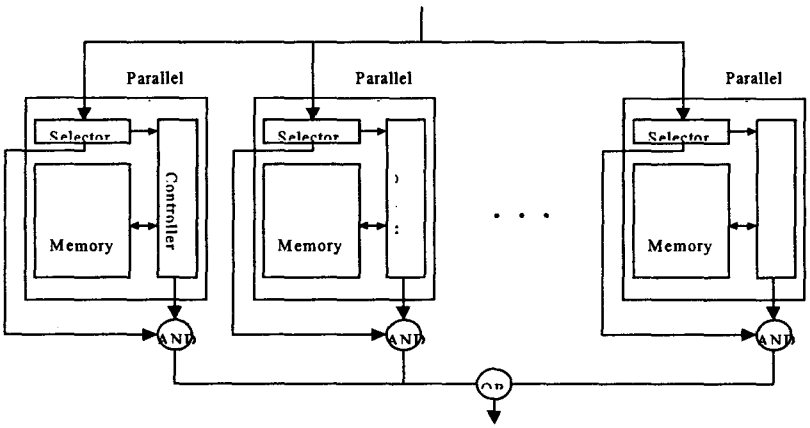

그림 3 병렬 검색 엔진의 구조

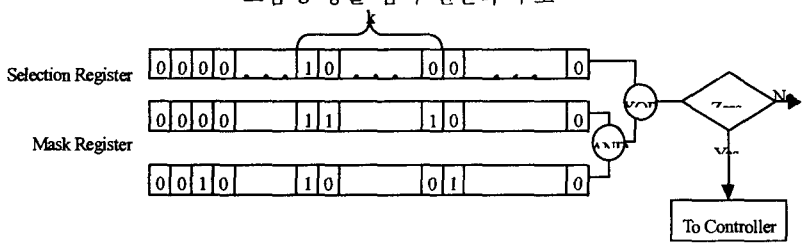

그림 4 선택기의 구조