2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

라우터 프로토콜을 위한 자료구조의 구현 및 성능 평가

강유화*, 오재천**, 이승민**,전병준**, 정 태 의**
* 한국전자통신연구원
** 서경대학교 컴퓨터과학과
e-mail: tejeong@skuniv.ac.kr

Implementation and Performance Evaluation of Data Structure for Route Protocol

Yoo Hwa Kang*, Jae Chan Oh**, Seung Min Lee**, Byung-Jun Chon**,Tae Eui Jeong**
* Electronics and Telecommunications Research Institute
** Department of Computer Science, Seokyeong University

요 약

통신 및 네트웍 사용자의 증가와 실시간 처리를 요하는 멀티미디어 데이터 서비스의 증가로 인해 네트웍 트래픽의 증가, 서비스 품질 저하 및 접속 속도의 감소를 초래하게 되었다. 이러한 문제를 해결하기 위하여 기존 라우터 보다 용량이 크면서 한계를 극복할 수 있는 장비가 검토 중에 있으며 또한 라우터의 용량확장에 따른 테이블의 크기 증가와 효율적인 경로 선택을 위한 검색 시간 증가를 초래하게 되므로 라우팅 테이블의 구성방식에 대한 연구는 매우 중요하다. 본 논문은 Zebra 라는 TCP/IP 기반의 라우팅 프로토콜을 관리하는 프리 소프트웨어를 이용하여 라우팅 테이블의 구조와 구성방식에 대하여 분석하며, 비교 알고리즘의 구현 및 성능 비교와 평가를 통하여 그 정당성을 검증한다.

1. 서론

최근 인터넷 사용이 기하급수적으로 증가하고, 요구되는 어플리케이션 종류가 다양해짐 따라, 멀티미디어 데이터의 이동량이 확산되었으며, 네트웍 트래픽 증가로 서비스 품질의 저하 및 접속 속도의 감소를 초래하게 되었다. 이러한 제반적인 문제를 해결하기 위하여 서비스 제공자(SP: Service Provider, ISP: Internet Service Provider)는 초고속 기간망(Backbone)을 구축하지 않으면 안될 상황에 이르렀으며, 대용량 트래픽을 수용할 수 있는 보다 막강한 고성능 네트웍 장비가 필요하게 되었다. 이에 따라 또한 현재 네크웍을 확장하는 방안이 모색되고 있다. 네트웍에 사용되고 있는 기존의 스위치나 라우터를 이용하여 이 문제를 해결하기에는 여러 가지 문제점 및 한계성이 존재하므로 기존의 라우터 보다 용량이 크면서 두 장비(스위치, 라우터)의 기능을 동시에 제공할 수 있는 장비를 검토하게 되었다.

라우터는 동일한 전송 프로토콜을 사용하는 분리된 네트웍을 연결해주는 장치로 네트웍 계층간을 서로 연결하며, 다양한 네트웍 관리 기능을 수행한다. 또한 라우터나 기타 다른 인터-네트워킹 장치에 저장되어 있는 라우팅 테이블에 네트웍상 특정 목적지까지의 경로를 유지하여 여러 경로 중 가장 효율적인 경로를 선택한다. 따라서 라우터 용량확장에 따른 라우팅 테이블의 크기 증가와 효율적인 경로 선택을 위한 검색시간 증가를 초래하게 되므로, 라우팅 테이블의 구성 방식 또한 검토 되어야 할 문제 이다.

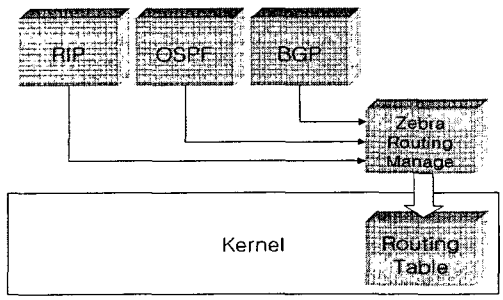
본 논문은 프리 소프트웨어인 Zebra 를 분석하여 기존 라우터에서 사용되는 라우팅 테이블 구성 알고리즘 분석과 성능평가를 통하여 그 정당성을 검증하는 논문으로, 제 1 장 서론, 제 2 장에서 관련연구에 대해서 살펴보며, 제 3 장은 라우터의 자료구조인 라우팅 테이블을 살펴보고, 라우팅

테이블 구성 알고리즘의 성능평가에 관해서는 4 장에서 기술하고, 마지막으로 5 장에서 결론을 맺는다.

2. 관련연구

Zebra 는 TCP/IP 를 기반으로 하는 라우팅 프로토콜을 관리하는 프리 소프트웨어 이다. 또한 하나의 프로토콜당 하나의 프로세스를 사용하며, BGP(Border Gateway Protocol)와 OSPF(Open Shortest Path First) 그리고 RIP(Routing Information Protocol)의 라우팅 프로토콜을 지원한다. 라우팅 프로토콜은 크게 기업의 근거리 통신망과 같은 자율 네트웍 내의 게이트웨이들 간의 라우팅 정보를 교환하는데 사용되는 프로토콜인 IGP(Interior Gateway Protocol)와 내부적으로 IGP 를 사용하는 도메인 간을 연결하는 인터넷 라우팅 프로토콜인 EGP(Exterior Gateway Protocol)로 나눌 수 있으며, 작은 규모의 네트웍에 주로 사용되는 RIP 와 대규모 계층적 구조에서 사용되는 OSPF 가 IGP 에 속하고, EGP 에 대한 예로는 BGP 를 들 수 있다.

그림 1 은 Zebra 의 구조로, 각 라우팅 프로토콜들은 Zebra 라는 라우팅 메니저를 통하여 라우팅 테이블을 접근하게 된다.



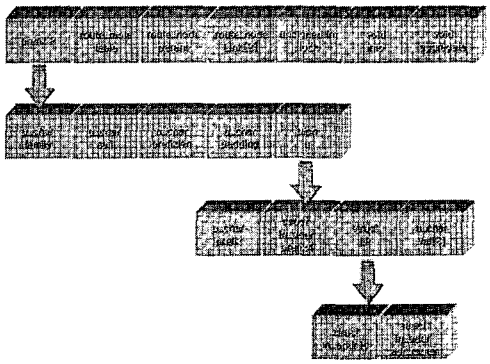
[그림 1] Zebra 의 구조

3. 라우팅 테이블의 구조와 구성

라우팅 테이블의 구조는 Radix 알고리즘을 이용한 이진 트리 형태를 이루고 있으며, 여러 필드를 가지는 노드들로 구성되어 있다.

3.1 라우팅 테이블의 구조

라우팅 테이블을 이루고 있는 각 노드들은 라우팅에 필요한 정보들을 유지하고 있으며, 이는 그림 2 에서 보는 것과 같이 크게 p, table, parent, link, lock, aggregate 로 이루어져 있다.



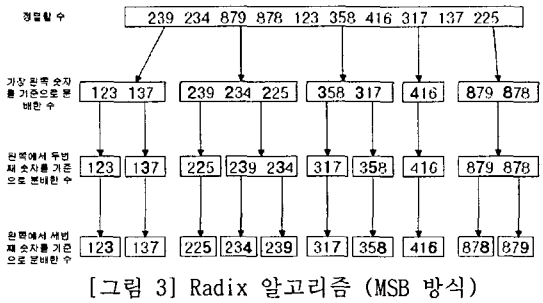
[그림 2] 라우팅 테이블의 노드구조

p 필드는 family, safi, prefixlen, padding, u 로 구성되어 있으며, table 과 parent 는 테이블과 부모를, link 는 '0' 과 '1' 의 값에 따라 좌측과 우측 자식노드를 가리킨다. lock 필드는 각 노드의 히트 수를 나타내며, lock 이 '0' 의 값을 가지게 되면 노드를 삭제하게 된다. info 와 aggregate 필드는 패킷 전송에 관한 정보와 IP Address/IP Network Mask 로 정해진 범위 값을 가리킨다.

① family: IP 버전(AF_INET, AF_INET6, AF_UNSPEC)safi: unicast, multicast 지원에 관한 정보
③ prefixlen: prefix 길이
④ padding: 패킷의 사이즈를 마쳐주기 위해 덧붙이는 padding 정보
⑤ u: prefix, prefix4(AF_INET), prefix6(AF_INET6), lp(AF_UNSPEC), val

3.2 라우팅 테이블의 구성

라우팅 테이블은 Radix 알고리즘을 이용하여 구성되었다. Radix 알고리즘은 기수의 값에 따라 분배하는 알고리즘으로 MSB(Most Significant Digit)부터 비교하는 방식과 LSB(Least Significant Digit)에서 MSB 방향으로 비교하는 두 가지 방식이 있다.

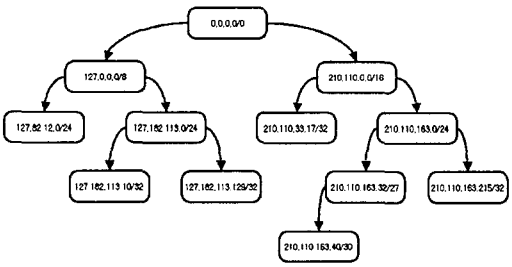


[그림 3] Radix 알고리즘 (MSB 방식)

그림 3 은 MSB 값부터 분배하는 방식으로 MSB 의 값을 읽어 그 값이 할당된 버켓에 분배하며 모든

대상에 대하여 1 차 분배가 끝나게 되면, MSB+1 의 값을 읽어 2 차 분배를 한다. 이를 LSB 까지 수행하게 되면 비로서 정렬이 되는 것이다.

라우팅 테이블은 2 진 기수법을 사용하여 Radix 알고리즘을 적용하고 있으며, 비트값에 따라 분배한다.



[그림 4] 라우팅 테이블의 구성 예

그림 4 는 라우팅 테이블의 간단한 구성 예로 이진 트리 형태를 이루고 있으며, 라우팅 테이블은 prefix 와 prefix length 에 의해 분배된다. 즉 추가하려는 노드의 prefix 중 이미 존재하는 노드의 prefix length+1 의 위치의 비트가 기수 값이 되어 ' 0' 일 때는 좌측 ' 1' 일 때는 우측으로 분배된다. 예를 들어 그림 4 의 prefix 값이 127.182.113.10 이며 prefix length 가 32 인 노드 A 가 존재하지 않는 다고 가정 할 때, 이 노드를 추가 시키는 과정은 다음과 같다.

① 루트 노드의 prefix length 값은 ' 0' 이므로 A 의 prefix 첫번째 비트에 의해 좌측노드로 이동하게 된다.
② 좌측 노드의 prefix length 는 ' 8' 이므로 8 비트 즉 1 바이트를 A 와 비교한다.
③ 1 바이트가 ' 127' 로 같으므로 A 의 9 비트 값이 1 이므로 우측 노드로 이동하게 된다.
④ 우측 노드의 prefix length 가 ' 24' 이므로 3 바이트를 A 와 비교한다.
⑤ 3 바이트가 127,182,113 으로 같으므로 A 의 25 비트 값이 ' 0' 이므로 좌측노드로 이동하게 되나 좌측노드가 존재하지 않으므로 이 위치에 추가하게 된다.

위의 결과 그림 4 의 라우팅 테이블을 얻을 수 있다. 그러나 이미 위와 같이 구성된 라우팅 테이블에 다시 prefix 와 prefix length 가 127.182.133.2/32 인 B 를 또 다시 추가 하게 되면 A 의 위치와 충돌하게 된다. 따라서 A 와 B 의 마지막 같은 비트값을 갖는 위치까지를 prefix length 로 하는 부모노드를 생성하여 추가 시킨 다음 A 와 B 를 재배치 시켜야 한다. 즉 prefix 와 prefix length 가 127.182.133.0/27 인 C 노드를 생성하여 A 와 B 의 부모노드로서 원래 A 의 위치에 위치 시켜야 한다.

3.3 라우팅 테이블의 구조와 구성방식에 따른 문제점

라우팅 테이블이 Radix 알고리즘을 적용한 이진트리 형태를 구성됨에 따라 여러 문제점을 가지고 있다. 그 첫번째는 2 진 Radix 알고리즘을 적용하여 이진트리 형태를 가지기 때문에 트리의 깊이가 길어진다는 것이다. 즉 추가, 삭제 등의 업데이트 과정에 있어서 보다 많은 트리의 노드를 순회하여야 한다는 것이며, 보다 많은 비교 연산을 수행하여야 한다. 두 번째는 트리가 가지는 일반적인 문제점으로 균형성을 들을 수 있다. 트리의 균형성이 깨져 한쪽 방향으로 경사를 이루게 되면 그 만큼의 평균 접근 시간이 증가 한다는 것이다. 세 번째 문제는 위에선 언급했던 추가적인 부모노드의 생성이다. 이는 불필요한 노드의 생성에서 오는 기억공간의 낭비와 그로 인해 접근해야 하는 노드 수의 증가로 탐색속도의 증가를 초래하게 된다. 따라서 보다 적합한 구조와 구성 알고리즘의 필요하다.

4. 라우팅 테이블 구성 알고리즘의 성능평가

본 논문에서는 Radix 알고리즘을 이용하여 구성된 이진트리 형태의 라우팅 테이블의 성능평가를 위하여 MLH(Modified Linear Hashing)을 비교 대상으로 선택하였다. MLH 은 Linear Hashing 이 정적으로 메모리를 할당함에 따라 생기는 메모리 낭비를 줄이기 위해 동적으로 변형한 것으로, 검색 시 이진트리 탐색이 루트를 경유해서 순차적으로 검색하는 반면에 MLH 은 해싱함수에 의해서 직접 또는 찾는 값에 인접한 노드부터 검색을 수행하기 때문에 검색시간을 줄일 수 있으며, 충돌이 없을 경우 이진트리에 비해 삽입 삭제할 노드의 위치를 빨리 찾을 수 있어 효율적이나 충돌이 발생했을 경우 충돌회비를 위해서 추가연산이 필요하게 된다.[5,6,7]

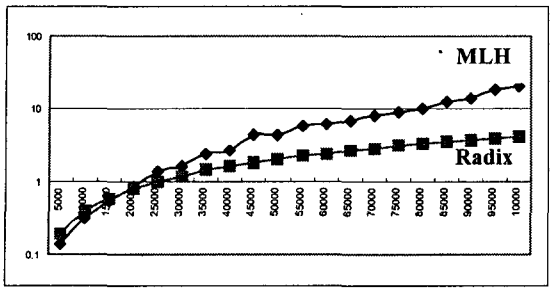
4.1 성능평가 방법

알고리즘의 성능 평가를 위하여 PTM(Performance Test Manager)이란 모델을 설계하였다. PTM 은 라우팅 테이블을 접근 및 제어 할 수 있는 TC(Table Controller)와 임의의 IP 를 생성할 수 있는 PG(Prefix Generator)를 가지고 있으며, 이를 이용하여 두 알고리즘으로 구성된 테이블을 제어하게 된다.

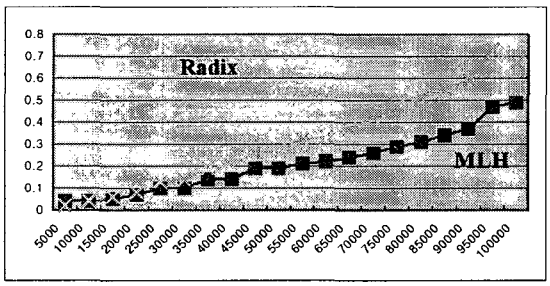
그림 5 와 6 은 Radix 알고리즘과 MLH 의 테이블의 노드 수와 시간과의 관계를 나타낸 것으로, 임의의 IP 를 가지고 라우팅 테이블을 생성하는데 드는 시간과 생성된 테이블의 메모리를 해제하는데 걸리는 시간을 측정한 것이다.

그림 5 에서 보는 것과 같이 MLH 은 테이블에 15000 개의 IP 를 추가하여 라우팅 테이블을 생성 할 때 까지는 Radix 알고리즘에 비해 테이블의 생성 시간이 적게 걸리나 추가 하려는 IP 의 수가 20000 개 이상에는 Radix 알고리즘이 보다 적은 시간을 보여

주고 있다. 이는 MLH 이 IP 를 추가 할 때 충돌이 발생하여 분할을 하는데 걸리는 시간이 증가 하기 때문이다.

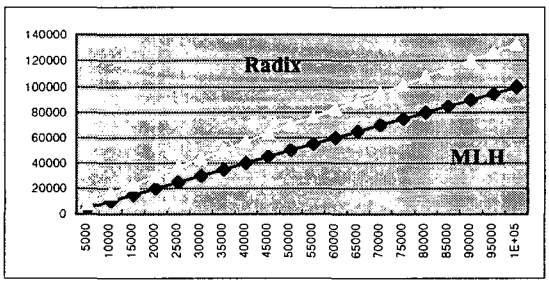


[그림 5] 생성시간 과 추가 IP 수

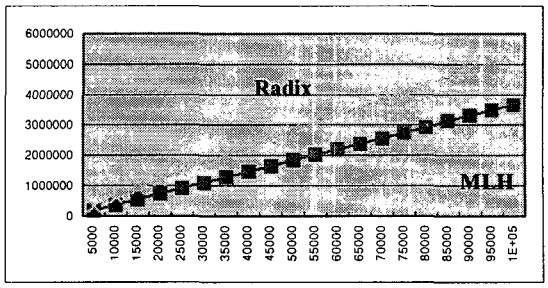


[그림 6] 해제 시간과 추가 IP 수

그림 7 과 8 은 임의의 IP 를 추가 했을 때 라우팅 테이블의 노드들의 수와 사용된 메모리를 나타낸 것이다. 그림 5 에서 보듯이 MLH 은 추가적인 노드의 생성이 존재하지 않지만, Radix 알고리즘은 노드를 추가할 때 발 허수노드가 발생하여 추가하려는 IP 의 수가 100000 일 때 약 30000 개의 허수 노드가 발생하는 것을 볼 수 있다.



[그림 7] 생성된 노드 수와 추가 IP 수



[그림 8] 할당된 메모리와 추가 IP 수

5. 결론

본 논문에서 Radix 라는 프리 소프트웨어를 이용하여 라우팅 테이블의 구조와 구성방식을 분석하였으며, 성능분석 위하여 MLH 을 비교 알고리즘으로 선택하였고 PTM 모델을 설계 하였다. 위에서 언급한 것과 같이 Radix 알고리즘은 MLH 보다 메모리의 낭비가 발생하나 시간은 MLH 보다 적게 걸리는 것을 보았다. Radix 알고리즘을 적용한 라우팅 테이블의 문제점과 성능분석을 토대로 추후에 대체 알고리즘에 대한 연구가 필요하다.

참고문헌

[1] Kunihiro Ishiguro, "The GNU Zebra Manual," Internet-draft, July 2000.

[2] Y. Rekhter, T. J. Watson, T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, March 1995.

[3] J. Moy, "OSPF Version 2," RFC 1583, March 1994.

[4] C. Hedrick, "Routing Information Protocol," RFC 1058, June 1988.

[5] Michael A. Harrison, James T. DeWolf, "Computer Algorithms: Introduction to Design and Analysis, Second Edition," pp.83-89, August 1993.

[6] Per-Ake Larson, "Performance Analysis of Linear Hashing with Partial Expansions," ACM Transactions on Database System, Vol. 7, No. 4, December 1982.

[7] Per-Ake Larson, "Dynamic Hash Tables," Communications of the ACM, Vol. 31, No. 4, April 1988.