IXP1200 Network Processor에서의 Route Lookup Algorithm에 관한 연구

정영환[0] 박우진 황광섭 배국동 안순신
고려대학교 전자공학과
{youngh, progress, hanriver, gooibil, sunshin}@dsys.korea.ac.kr

Route Lookup Algorithm in IXP1200 Network Processor

Young-Hwan Jung[0] Woo-Jin Park Kwang-Seop Hwang Gook-Dong Bae Sun-Shin An
Computer Network Lab. Dept. of Electronics Eng., Korea University

요 약

최근 동영상, 음성 등 멀티미디어 트래픽의 급속한 증가로 네트웍에서 요구되는 전송률은 점점 증가될 전망이다. 이에 따라 라우터도 요구되는 전송률에 대처해 나가기 위해 고속화 되어가고 있다. 이러한 고속 라우터 개발을 가능하게 하는 여러가지 Route Lookup 알고리즘들이 연구되어 왔다. 본 논문에서는 효율적인 Lookup 알고리즘의 원리와 여러 가지 Lookup 알고리즘들에 대해서 알아보고 IXP1200 Network Processor를 이용한 포워딩 엔진 개발에 있어서 사용된 Route Lookup 알고리즘을 분석하여 그 효율성에 대해서 연구해 본다.

1. 서 론

현재까지 연구된 다양한 route lookup 알고리즘이 존재하지만 각 알고리즘마다 가지는 동일한 목표는 메모리 참조 회수를 최대한 줄이면서 메모리를 적게 소비하는 것이다 [3]. 일반적으로 메모리는 속도가 느리기 때문에 lookup의 성능에 크게 영향을 끼치게 된다. 이러한 주요 bottleneck 인 메모리를 적게 참조 할수록 빠른 lookup이 가능해 지게 된다. 또한 메모리의 사용을 최소화 해야 하는 이유는 lookup 속도를 높이기 위해서는 고속의 메모리를 사용해야 하는데 고속의 메모리는 가격이 비싸고 집적도가 떨어지기 때문에 많은 양의 고속의 메모리를 사용할 수 없기 때문이다. 또한 IPv6가 도입되게 되면 주소길이는 32비트에서 128비트로 급격하게 커지기 때문에 lookup에 사용되는 메모리 양을 고려하지 않으면 안 된다. 따라서 속도가 느리지만 가격이 싼 DRAM과 속도가 빠르지만 가격이 비싼 SRAM을 적절히 공유해서 메모리 참조 회수를 최대한 줄이는 알고리즘을 설계한다면 가장 바람직할 것이다

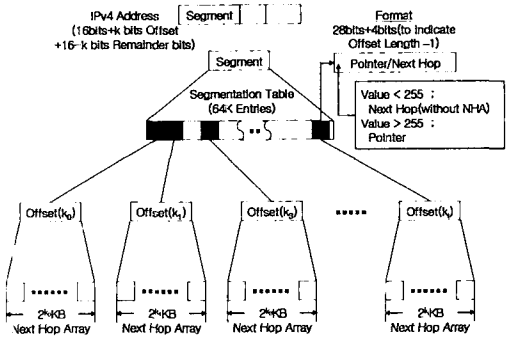
2.관련 연구

2.1 Patricia Trie

CIDR이 소개되면서 기존의 hash기법을 대체하는 Patricia Trie라 불리는 radix trie의 한가지 형태가 구현 되었다. 이 알고리즘은 어떤 길이의 Prefix도 수용할 수 있고 longest prefix matching을 검색하기 위해 사용할 수 있다. 검색은 trie traversing시에 한 번에 주소의 한 비트를 조사함으로써 수행된다. Trie의 leaf 노드에 도달하였을 때, 주소는 그 노드에 저장되어 있는 route 정보 와 비교되어진다. 만약에 맞는 것이 있다면 longest matching 되는 것을 찾은 것이다. 만약 맞지 않는다면 backtracking 과정을 통 해서 그 leaf 노드의 parent 노드로 찾아가야 한다.

보통의 경우 O(W), backtracking 시 최악의 경우 $O(W^2)$의 iterations이 필요하다. (W는 주소길이와 같은 trie의 최대 높이(height)이다.)

2.2. Indirect Lookup with Variable Offset Length [5]



[그림 1] Indirect Lookup with Variable Offset Length

Indirect-lookup with variable offset length에서는 [그림 1]에 나타낸 것과 같이 prefix의 16비트 이후의 부분을 offset이라하고 그 길이를 k로 한다. 2^{16}개의 entry를 가지는 segmentation table과 2^k개의 entry를 가지는 offset table로 구성된다. Segmentation table의 entry는 직접 next hop을 가지거나(prefix length<=16, entry value<=255), offset table의 pointer를 가진다(prefix length > 16, entry value>255). Prefix length가 16보다 큰 경우 그 길이가 segmentation table에 기록되는데 최대가 16이므로 4비트가 prefix length 용으로 사용된다. Offset table은 2^k개의 entry를 가지며 next hop을 표시

하는데 1바이트가 사용되므로 offset table의 크기는 각 segment당 2^k바이트가 된다. 이 방법을 사용하면 lookup 시 최소 1번(prefix length <= 16), 최대 3번(prefix length >= 20)의 메모리 참조를 하게 된다. 또한 필요한 memory가 1.5MB~2MB / 4000 segments로 비교적 적다.

3. Route Lookup 알고리즘의 설계

3.1 설계 시나리오

기본적으로 Longest Prefix Matching 규칙을 따르고 위에서 알아본 두 가지 알고리즘 중 적은 메모리 참조로 lookup이 가능하고 메모리도 효율적으로 사용할 수 있는 Indirect lookup with variable offset length방법을 변형하여 segmentation table을 두 개를 두어 더욱 적은 메모리 참조가 일어나도록 하는 구조를 가지고 있다.

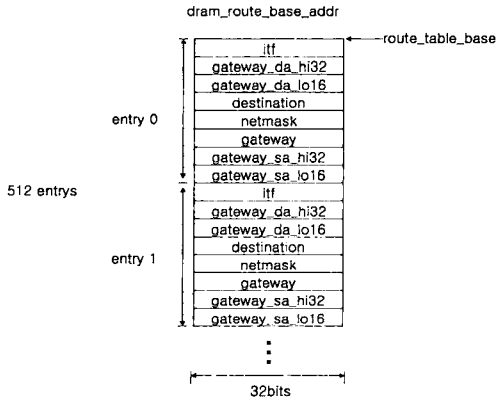
- SRAM과 SDRAM의 효율적인 공유 사용

많은 메모리 공간을 필요로 하는 실제적인 route정보는 SDRAM에 저장하고 route lookup에 필요한 trie 구조는 SRAM에 둔다. SRAM에서 lookup을 진행하여 결과 값으로 SDRAM에 있는 route entry의 위치를 받아서 SDRAM을 참조하여 lookup을 완료하는 과정이다. 이렇게 하면 속도가 느린 SDRAM 참조는 마지막에 한번만 일어나고 여러 번의 메모리 참조가 필요한 실제적인 lookup 과정은 SRAM에서 이루어지기 때문에 빠르고 효율적인 lookup이 가능해진다.

- 메모리 참조 회수를 줄이면서 적은 메모리 공간을 사용하는 lookup table의 구조

메모리를 단순히 32 비트로 index된 2^{32}개의 entry를 가지는 table로 만든다면 단 한번의 메모리 참조로 lookup이 이루어 질 수 있으나 이는 대용량의 메모리를 필요로 하기 때문에 바람직하지 않다. 또한 메모리 용량만을 고려하여 단순히 binary tree를 사용하게 된다면 메모리 참조가 여러 번 발생하여 빠른 lookup이 불가능해진다. 이러한 점들을 고려하여 16, 8, 4 비트로 index된 table을 trie 구조로 만들어서 적절한 structure의 크기를 유지하면서 빠른 lookup이 가능하도록 설계되었다.

3.2 Route Table 구조

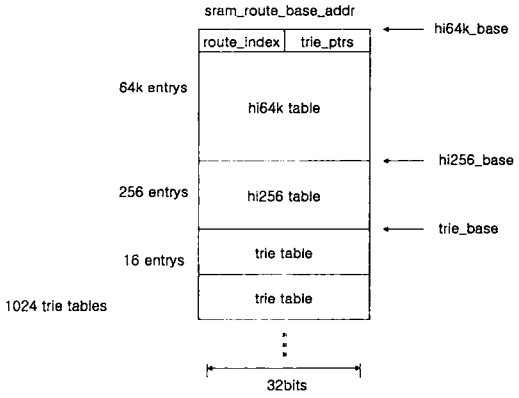


[그림 2] Route Table 구조

SDRAM에 위치한 route table은 최대 512개의 entry를 가지며 하나의 entry는 8개의 주소 공간을 차지한다.

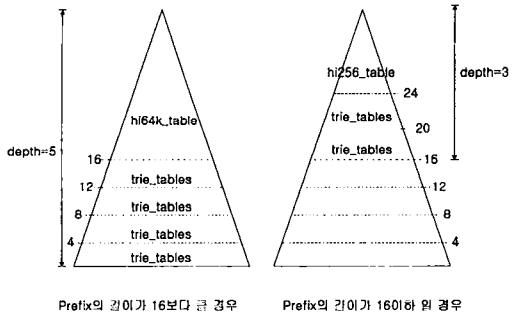
SRAM의 trie table에서 lookup을 진행하여 얻어진 route entry index를 이용하여 SDRAM의 route table을 참조하여 outgoing interface를 결정 한다.

3.3 Trie Table 구조



[그림 3] Trie Table 구조

주소의 상위 16비트로 index된 hi64k_table과 상위 8비트로 index된 hi256_table을 두어 lookup의 시작 위치로 사용한다. 그리고 주소의 4비트 조각 단위로 index되는 최대 1024개의 trie_table을 두어 lookup trie의 중간 노드로 사용한다. trie table의 entry의 0-15비트 위치에는 다음 trie table 을 가리키는 pointer가 들어있고 16-31비트 위치에는 lookup 결과인 route table index가 들어있다.



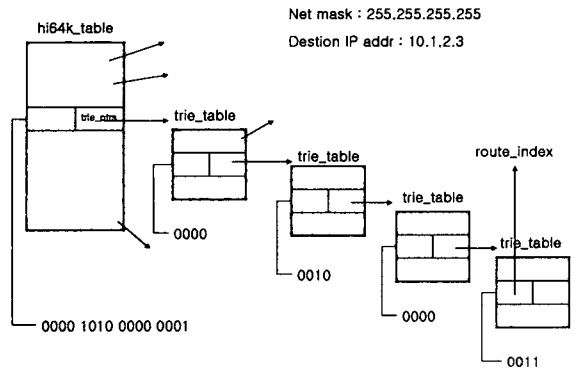
[그림 4] Lookup Trie 구조

hi64k_table과 hi256_table 둘 중 어느 것을 시작 위치로 선택할 것인가와 lookup depth는 destination IP address의 network prefix의 길이에 의해 결정된다. Prefix의 길이가 16 이하이면 hi256_table에서 시작하며 그렇지 않으면 hi64k_table에서 시작한다. 이렇게 lookup의 시작 위치를 두 개로 나누는 이유는 prefix의 길이가 긴 주소를 검색할 때 메모리 참조 회수를 줄이기 위함이다.

Trie lookup 과정을 살펴 보면 destination IP address

의 상위 31-16비트 혹은 31-24 비트를 hi64k_table이나 hi256_table에서 검색하여 다음 trie_table을 찾아낸다. destination IP address의 나머지 비트들은 prefix의 길이까지 4비트 조각 단위로 trie_table을 검색하여 다음 trie_table의 pointer나 결과 route index를 찾아낸다.

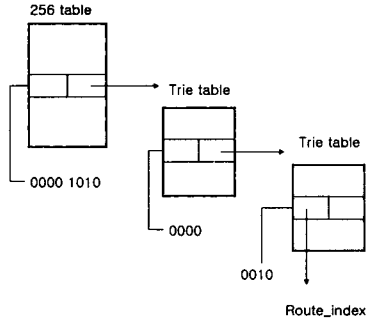
예를 들어, prefix의 길이가 32인 경우에는 destination address의 31-16 비트로 index된 hi64k_table에서 다음 trie_table을 찾아내어 destination address의 15-12 비트로 index하여 다음 trie_table을 찾아낸다. 찾아낸 trie_table에서 destination address의 11-8 비트로 index하여 다음 trie_table 을 찾는다. 이런 식으로 검색을 진행 하다가 마지막 3-0 비트로 trie_table을 index하여 결과 값인 route_index를 찾게 된다. 이런 과정을 예를 들어 그림으로 나타낸 것이 [그림-5] 이다



[그림 5]

이 경우가 최악의 경우로서 1번의 hi64k_table 참조와 4번의 trie_table의 참조를 하기 때문에 총 5번의 SRAM 참조가 일어난다. 최선의 경우를 생각해 보면 Prefix의 길이가 8일 때 hi256_table에서 바로 route_index를 찾을 수 있으므로 1번의 SRAM 참조가 일어난다.

또 다른 예로 prefix의 길이가 16이고 destination IP address가 10.2.2.1일 때 lookup 과정을 [그림-6]에 나타내었다.



[그림 6]

이 경우는 prefix의 길이가 16이기 때문에 address의 31-16 비트만을 검색하면 된다. 31-24 비트에서 한번, 23-20 비트에서 한번, 19-16비트에서 한번, 총 세 번의 SRAM 참조를 한다.

Network prefix 길이에 따른 lookup 시작 위치, 결과 route_index의 위치, SRAM 참조 횟수(trie의 depth)를 표로 정리하면 [표-1]와 같다.

Prefix 길이	시작 위치	route_index 위치	SRAM 참조 횟수
32	hi64k_table	0-3 bit indexed trie	5
28	hi64k_table	4-7 bit indexed trie	4
24	hi64k_table	8-11 bit indexed trie	3
20	hi64k_table	12-15 bit indexed trie	2
16	hi256_table	16-19 bit indexed trie	3
12	hi256_table	20-23 bit indexed trie	2
8	hi256_table	hi256_table	1
4	hi256_table	hi256_table	1
0	hi256_table	hi256_table	1

[표 1]

4. 결론 및 향후 연구

Route lookup algorithm 설계에 있어서 가장 먼저 고려해야 할 두 가지 사항은 메모리 참조 회수와 Structure의 크기이다.

본 논문에서는 route lookup trie 구조를 IP address에 의해 Index된 table을 trie로 만들어 메모리 참조 회수와 structure의 크기를 적절히 조화시켜 효율적이고 빠른 lookup이 가능하도록 한 algorithm을 분석하여 그 효율성에 대해서 연구해 보았다.

향후 IPv6가 활성화 되게 되면 주소의 길이가 극적으로 증가되기 때문에 현재의 route lookup algorithm으로는 필요한 메모리 양도 크게 늘 것이고 lookup 속도 또한 현저하게 떨어질 것으로 예상된다. 향후에는 IPv6 주소 체계에서 사용 가능한 새로운 개념의 lookup algorithm에 대해서 연구하고 IXP1200 Network Processor상에서 설계하는 것이 과제로 남아있다.

5. 참고 문헌

[1] "IXP 1200 Hardware Reference Manual", Intel, June, 2001

[2] "IXP1200 Gigabit Ethernet Example Design", Intel, august, 2001

[3] "Small Forwarding Tables for Fast Routing Lookups", Mikael Degermark

[4] "IP Routing Lookups Algorithms Evaluation", Lukas Kencle, July, 1998

[5] "A Fast IP Routing Lookup Scheme for Gigabit Switching Routers", Nen-Fu Huang"