제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

네트워크 보조 프로세서를 사용한 고속 패킷 필터링

이홍석*, 김종수*, 정기현*, 최경희**

*아주대학교 전자공학부, **아주대학교 정보통신 대학원
e-mail: hsyi98@csl.ajou.ac.kr

Fast Packet Filtering using Network Coprocessor

Hongseok Yi*, Jongsu Kim*,Kihyun Chung*, Kyunghee Choi**

*Dept. of EE, Ajou University
** Graduate School of I&C, Ajou University

요 약

사용자의 인터넷 서비스 고속화 요구가 증대되면서 스위치나 라우터, 보안 장비와 같은 인터네트워킹 장비들의 성능 향상 요구도 커지고 있는데 이는 패킷을 처리하는 양과 속도를 향상시켜야 한다는 것을 의미한다. 스위치와 같은 라인 인터페이스 장치들은 주로 전용 하드웨어로 설계되므로 네트워크의 트래픽 처리 성능을 보장 받을 수 있으나, 보안 장비나 네트워크 응용 장비들은 일반 서버 기반에서 트래픽을 처리하는 경우가 많아 시스템적으로 성능 향상에 제약을 받거나 성능 향상을 위해서 높은 비용을 지불해야 한다. 근래의 이러한 통신 관련 장비들에서의 패킷 처리 방식은 단순한 연결 차원을 넘어 패킷을 분석하고 연결을 제어하는 모습을 보이고 있는데 이러한 추가 작업 때문에라도 시스템에 많은 부하가 발생한다. 이러한 트래픽의 분석 처리를 빠르게 하기 위해서는 입력된 데이터와 설정된 규칙간의 비교와 판단이 빨라질 필요가 있는데, 본 논문에서는 이를 위해 기존에 연구된 몇 가지 S/W 적인 해결 방법과 H/W 적인 방법들을 분석하고, 더 나은 검색 성능을 위해 H/W 기반 네트워크 보조 프로세서를 이용한 방식을 제안하고 실험을 통하여 검증하였다.

1. 서론

인터넷 사용자들의 서비스 향상 요구는 끝없이 증가하고 있으며 이에 따른 서비스 수위를 맞추기 위해서 인터넷 회선 망 업체들은 Cable, ADSL (Asymmetric Digital Subscriber Line) 등의 고속 인터넷 서비스에 이어 최근에는 VDSL(Very high data rate DSL) 서비스를 시작하였고, 학내망, 기간망의 경우 백본(Backbone)이 10G bps 로 구축하기 시작했다. [7] 이러한 고속 인터넷 서비스는 인터네트워킹 장비인 스위치나 라우터가 고속 통신을 할 수 있도록 고성능화 되었기에 가능한 일이다. 인터넷 속도와 인터넷 서비스는 서로의 질을 향상시키는 요인이지만 보안 서비스만은 예외이다. 보안 서비스는 안전한 네트워크 접속을 위해 속도를 희생하기 때문이다. 근래 대형 보안 사고가 잇따르면서 성능 못지 않게 보안에 대한 관심이 증가하고 있는데 이미 많은 기업들이 보안 관련 장비인 침입 탐지 시스템, 침입 방지 시스템, 방화벽 등의 솔루션을 채택하고 있다. 특히 과거보다 빠른 망의 속도 때문에 자기 복제가 가능한 웜(Worm) 등으로 인한 피해가 더 커져 앞으로 보안 장비 사용이 더욱 크게 늘어 날 것으로 보인다. [8]

이러한 보안 시스템이 망에 설치되어 있다고 해서 무조건 통신이 느려지는 것은 아니지만, 트래픽이 집중되거나 분석해야 할 사항이 많게 되면 병목 현상의 주 요인이 될 수 있다. 그러므로 가능한 망이 고속화 된 정도에 따라 보안 장비 또한 고성능을 발휘할 수 있어야 망에서 사용할 수 있는 최대 대역폭을 활용할 수가 있다.

방화벽과 같은 보안 장비들은 외부 망에서 내부 망으로 들어오는 패킷을 검사하여 입력 허용(Permit) 혹은 거부(Discard)를 판단하는데, 보통 내부 망을 보호하는 정책을 테이블로 만들어서 보관한다. 방화벽은 입력으로 들어오는 모든 패킷에 대해 룰 테이블(Rule Table)을 검색하여 설정된 정책에 부합하는 패킷인지 아닌지를 확인하는데, 이때 허용 혹은 폐기를 결정하는 것을 패킷 필터링(Packet Filtering)이라고 한다. 입력 패킷과 정책을 비교하기 위해 룰 테이블을 찾을 때 걸리는 시간은 성능에 민감한 요인으로 작용하므로 이 방법을 찾는 것이 보안 장치의 성능을 높이고 네트워크의 대역폭을 보장하는 길이다.

근래 인터넷 서비스나 침입 공격이 다양해짐에 따라 내부 네트워크를 보호하기 위한 정책도 많이 추가 되고 있는데 이에 따라 룰 테이블의 크기도 커지고 있다. 그러므로 적은 메모리 사용용량, 빠른 검색, 빠른 업데이트 시간, 룰의 복잡성 제거, 구현의 용이

성 등의 요건을 충족시킬 수 있는 알고리듬이 절실해 지고 있다. 이에 룰 검색을 할 수 있는 방안으로 제시되고 있는 몇몇 소프트웨어적 방식과 하드웨어적 방식을 직접 실험을 통해 비교해 보고 본 연구에서 제안하는 네트워크 보조 프로세서를 이용한 방식과 메모리 사용량, 탐색 속도 면에서 살펴 보았다.

이 논문의 2 장에서는 기존에 발표된 소프트웨어, 하드웨어 기반의 탐색 알고리듬을 언급하고, 3 장에서는 일반 컴퓨터 시스템에서 작동하는 소프트웨어를 돕기 위해 하드웨어로 설계된 네트워크 보조 프로세서를 이용한 S/W, H/W 복합 설계 방식을 제안하였다. 4 장에서는 실험을 통해 알고리듬 별로 패킷 필터링에 대한 성능을 분석하였고, 5 장에서는 결론 과 향후 과제에 대해서 언급하였다.

2. 룰 탐색 알고리듬

입력된 패킷에 포함된 몇 가지 정보 필드(Field)를 이용하여 검색한다고 할 때 그 필드가 어떤 룰에 속하는지를 알아내기 위해서 패킷을 해석하고 이미 정의된 정보들과 비교하는 과정이 룰 탐색(Rule Search)이다. 예를 들어, <표 1> 과 같이 입력으로 들어온 패킷에서 찾고자 하는 특정 정보(Field1, Field2)를 조합한 것이 룰(Rule; R*n*)이 되고, 이들을 모아놓은 것이 룰 테이블(Rule Table)이다. 입력된 패킷에서 추출한 정보가 각각 A3, B3 라면 이는 R3 에 매치 되어 알려지게 된다.

<표 1. 룰 테이블>

Rule	Field 1	Field 2
R1	A1	B1
R2	A2	B2
R3	A3	B3
R4	A4	B4
R5	A5	B5

일반적으로 4 계층(Layer 4; Session Layer)에서 룰을 탐색하여 패킷의 통과 여부를 결정하는 4 계층 필터링의 경우 발신지/목적지 주소(Source/Destination IP), 발신지/목적지 포트(Source/Destination Port), 프로토콜(Protocol) 등의 5 개의 필드를 이용하여 탐색을 하며, 설정되는 룰의 개수는 적용되는 망의 크기, 속도, 방식에 따라 달라지지만 보통 수 백~수 만개 사이 이며 계속 늘어나는 추세이다. 그러므로 룰과의 비교시간이 성능에 많은 영향을 주게 되는데 과거에 사용되었던 몇 가지 알고리듬을 이용한 방식을 살펴 보면 다음과 같다.

2.1 순차 탐색(Linear search)

순차 탐색 알고리듬은 가장 간단히 구현 할 수 있는 알고리듬으로 룰 테이블의 첫번째 항목부터 차례대로 입력 패킷의 선택된 필드와 비교해 나가는 검색 방법이다. 이 방식의 장점은 구현이 간단하며, 메모리 크기가 룰의 크기와 비례하여 증가하므로 메모리 사용량이 적으며, 앞으로 소개할 해쉬(Hash)나 Cross Product 알고리듬에서 해야 하는 사전 계산이 필요치 않다는 것이다. 대신 룰이 복잡해도 사용은 가능하지만 검색 후 결과값을 내는 데 걸리는 시간이 일정치 않으며 룰의 개수가 많을 경우 성능의 저하가 심해 좋지 않은 알고리듬이다.

2.2 해쉬(Hash)

순차 탐색과 같이 1 차원적 탐색 기법으로 여러 분야에 많이 사용되는 기법이다. 해쉬 알고리듬에서는 입력된 필드 중에서 일부 혹은 전체를 추출하여 키(Key)값을 생성하기 위해서 해쉬 함수(Hash Function)를 이용한다. 이 키 값이 바로 찾고자 하는 룰이 있는 위치가 된다.

이 알고리듬은 해쉬 함수를 잘 만드는 것이 중요한데, 그것에 따라서 해쉬의 성능이 좌우되기 때문이다. 기본적인 해쉬 알고리듬에서는 원본 데이터 내에서 몇 가지 부분 값을 얻어와서 그 값으로 키를 만들고 이 키 값이 있는 곳에 룰을 저장한다. 만약 같은 키 값이 있을 경우에는 Linked List 로 구현한다. 좋지 못한 해쉬 함수는 어느 한 값으로 키가 집중되어 알고리듬의 성능이 순차 탐색 정도로 크게 나빠지게 된다.

한편, 대부분 룰 테이블은 룰의 상대적 위치에 따라 내부망의 보안 정책이 달라지기 때문에 해쉬 알고리듬을 적용하기 전에 미리 룰을 분석하여야 한다는 단점이 있다. 그러나 해쉬 함수를 잘 만들 경우에 이 알고리듬은 자료 수에 상관없이 상수의 검색길이를 가지는 매우 빠른 검색 방법이다.

2.3 벡터 곱(Cross Product)

벡터 곱 알고리듬은 1 차원 적인 앞의 두 방식과는 다르게 룰의 탐색이 지역적(Geometric) 방식으로 이루어 진다. 표 2 와 같은 룰 테이블이 있을 경우 표 3 와 같이 종류를 열(Column)별로 나누어서 그 내에 있는 모든 항목을 적어주고 Wildcard(*) 는 String Default 로 맨 나중에 적어준다.

<표 2. 일반 룰 테이블의 예>

Dest IP	Src IP	Dest Port	Src Port	Prot	Act
A	*	25	*	*	Allow
A	*	23	*	TCP	Allow
B	T1	55	*	UDP	Block
B	*	*	125	*	Block
B	T2	*	*	TCP	Allow
*	*	*	*	*	Block

이 방식에서는 5 개의 Tuple 에 대해서 각각의 필드값을 가지고 조합하여 검색을 하게 되는데, 위의 경우 표 4 와 같은 3*3*4*2*3 = 216 개의 Producting 테이블이 생성되게 된다. 이 알고리듬도 사전 룰 처리가 필요하다는 점과 메모리 소요량이 많다는 점이 단점이지만 매우 빠른 검색을 할 수 있는 장점 때문에 속도 위주의 프로그램에서는 선호 하는 방식이다.

<표 3. 룰 테이블을 각 열 별로 나눈 테이블>

3 * 3 * 4 * 2 * 3 = 216

Dest IP	Src IP	Dest Port	Src Port	Prot
A	T1	25	125	TCP
B	T2	23	def.	UDP
def	def	55		def
		def		

<표 4. Cross Producting 테이블>

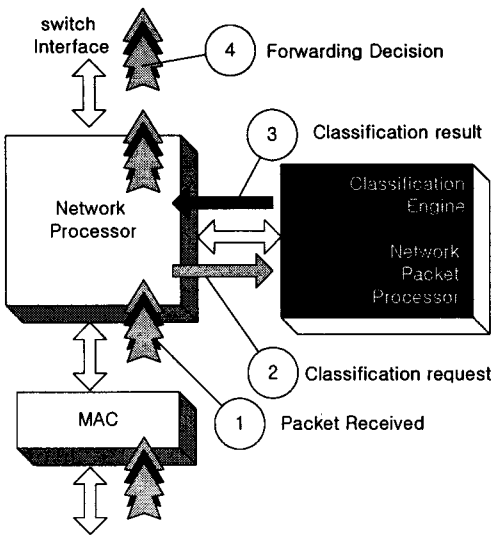
Num	Cross Product	Matching Filter
1	A,T1,25,125,TCP	Filter 1
2	A,T1,25,125,UDP	Filter 1
3	A,T1,25,125,def	Filter 1
4	A,T1,25,def,TCP	Filter 1
5	A,T1,25,def,UDP	Filter 1
6	A,T1,25,def,def	Filter 1
...	. . .	.
215	def,def,def,def,UDP	Filter 6
216	def,def,def,def,def	Filter 6

2.4 하드웨어 기반 알고리듬

하드웨어를 이용하여 룰을 탐색하는 방법에도 여러 가지 방식이 있는데, FPGA 를 이용하여 LookUp 테이블을 구성하는 방식이나 [5], 비트맵(Bitmap Intersection) 알고리듬을 하드웨어로 구현하여 룰 탐색을 도와주는 방법도 있으며[6], TCAM(Ternary Content Addressable Memory)을 이용한 것들도 있다. 하드웨어를 통한 룰 탐색 방식의 공통적인 장점은 탐색 속도가 매우 빠르고 또한 컴퓨터 프로세서의 자원을 소모하지 않기 때문에 부하 분산의 효과가 있다는 점이다. 즉, 이들의 지원을 받아서 패킷을 전송 혹은 폐기를 결정하게 되면 소프트웨어만으로 구성한 알고리듬 보다 우수한 성능을 낼 수 있게 된다. 그러나 대부분 일반 프로세서와 연동하기 위한 회로 설계와 추가 비용이 들며 인터페이스에 많은 제약과 시간이 걸린다는 점과 하드웨어 제어를 위한 프로그램도 작성이 까다롭다는 단점이 남는다.

3. 제안한 탐색 모델

하드웨어 방식의 단점에도 불구하고 전체 시스템의 성능을 향상시키기 위해서는 추가 장치를 설치해야 하는데 가능한 한 가지 용도로 고정되어 변형이 어려운 방식을 피하고 룰도 현재적 시점에 맞게 큰 크기를 지원할 수 있는 방법을 채택해야 한다. 또한 인터페이스가 시스템과 고속으로 연결되어 인터페이스에 의한 성능 저하도 피해야 한다. 이러한 사항들을 고려하면 Multi-TCAM 방식을 선택할 수도 있으나 H/W 인터페이스 및 S/W 제어에 대한 부담으로 선택의 폭이 좁다. 그래서 본 연구에서는 네트워크 보조 프로세서(Network Coprocessor) 중의 하나인 NCP(Network Classification Processor)를 사용하여 현 시스템의 룰 탐색 작업을 보조할 수 있도록 시스템을 구성하였다. 이 시스템 구성은 그림 1 과 같다.



<그림 1. NCP 와 NPU 와의 system 의 구조>

NCP 는 고속 패킷 처리를 위한 시스템에 많이 사용되는 네트워크 프로세서(Network Processor) 등과 고속 메모리 버스인 SSRAM(Synchronous SRAM) 버스와 직접 인터페이스 되어 인터페이스에 의한 손실이 적으며 내부에 룰 설정을 위한 메모리를 보유하고 있어 원 시스템의 메모리 자원을 차지하지 않고 데이터 처리를 수행할 수 있다.

시스템의 패킷 처리 과정을 보면, 본 시스템인 네트워크 프로세서에 패킷이 들어왔을 경우(1) 그 패킷을 SSRAM 인터페이스를 통해서 NCP 에게 보내게 된다. (2) NCP 는 그 내부에 설정된 룰 테이블을 참조하여 탐색 과정을 수행하게 되고 찾은 룰의 결과 값을 네트워크 프로세서에 알려주게 된다. (3) 그 후 NPU 는 그 결과에 따라서 패킷을 원하는 곳에 전달 혹은 폐기의 절차를 수행하게 된다. (4) NCP 는 일반 프로세서 시스템으로도 대체할 수 있으므로 사용에 제약은 심하지 않다.

룰의 설정은 본 프로세서 시스템에서 역시 SSRAM 인터페이스를 통해 설정할 수 있으며 관리를 위해 양 측에 룰 설정 정보를 보유하게 할 수 있다.

<표 5. 각 알고리듬에 따른 이론적 성능비교>

Algorithm	Space	Time	Power
Table	2^w	1	1
Linear	n	n	n
CAM	n	1	n
NCP using B-tree	2n+(2n-p)/(p-1)	$\log_p 2n$	$(p-1)\log_p 2n$

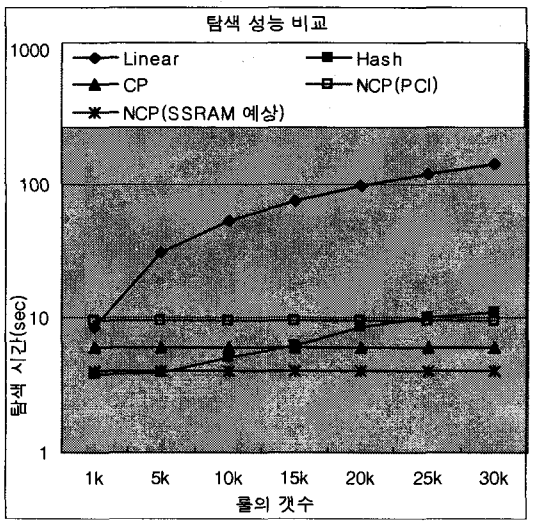
- w: 분류를 위해서 사용되는 비트의 너비(width)
- n: 룰의 개수
- p: B-tree 의 order

이 방식의 특징은 2 계층부터 7 계층까지의 폭 넓은 탐색이 가능하고, 또 룰의 복잡도 면에서 보면, 기존

의 소프트웨어나 하드웨어(Bit mask)에서는 1:1 매칭이나 그룹 부합 정도의 기능밖에 수행하지 못했지만, 여기서는 크기 (<, >) 비교 연산도 수행할 수 있다. 일반적으로 단순 CAM 방식의 룰 복잡도를 1 x F 라고 하고, TCAM 이 Bit mask 기능을 추가하여 2 x F 라고 할 때, NCP 의 룰 복잡도는 <, > 까지 추가하여 4 x F 가 된다. [3]에서 제시했던 각 알고리듬에 따른 성능을 표로 나타내어 요약하면 표 5 와 같다.

4. 각 알고리듬의 룰 탐색 성능 평가

실험환경을 다음과 같이 구성하였다. 입력되는 패킷을 파일로 읽어서 룰 테이블과 비교를 하는데, 시간 측정은 입력 패킷의 필드 정보와 룰 테이블간의 비교 시간 만을 측정하였다. 입력으로 들어오는 패킷의 수는 630,000 개 이다. 그림 2 는 알고리듬 별 룰의 개수에 따른 시간 측정 표이다. 위 그림에서 X 축은 설정한 룰의 개수를 의미하고, Y 축은 63 만개의 입력 데이터를 처리하는데 걸리는 시간을 나타낸다. 63 만개의 데이터는 [9]에서 패킷의 IP 헤더(Header)만을 모아놓은 파일을 이용하였고, 설정된 룰은 일반 방화벽의 룰로 설정하였다.



<그림 2. 각 알고리듬 별 성능비교>

그래프를 보면 순차 탐색 알고리듬은 룰의 개수에 따라 탐색 시간이 선형적으로 증가하여 가장 나쁜 성능을 보이고 있으며, 해쉬 알고리듬은 룰의 개수가 증가함에 따라 약간의 속도 저하가 있었으며 Cross Product (CP) 알고리듬의 경우에는 룰의 개수와 상관없이 일정하고 빠른 탐색 속도를 보여주고 있다. 또한 Network Classification Processor(NCP) 도 룰의 개수와 상관없이 일정한 성능을 보여주고 있으나 해쉬와 CP 방식보다 전반적으로 낮은 성능을 보이고 있다. 그러나 NCP 가 성능이 떨어지는 요인은 현재 실험을 위해서 PC 기반에서 사용할 수 있는 33Mhz 32bit PCI 기반으로 변경되어 설계 되었기 때문이다. 버스를 거쳐 오는 시간이 내부 데이터 처리보다 빠를 수는 없다. 실제 적용을 위해서는 133Mhz SSRAM 버스에 인터페이스 될 예정이므로 성능은 CP 방식 보다 낮은 탐색 시간이 표시될 것이다.

5. 결론 및 향후 향후 과제

본 논문에서는 패킷 분석 시스템에서 사용할 여러 룰 탐색 방법에 대한 비교를 통하여 하드웨어 기반 룰 탐색 방법이 패킷의 대용량 고성능 처리에 적절한 대안이 될 수 있음을 실험을 통하여 검증하였다. 앞의 NCP 실험은 성능을 완벽히 낼 수 없는 환경에서 실험하여 정확한 성능을 도시할 수 없었으나 낮은 버스 속도를 감안하면 현재의 성능도 상당히 높은 수준으로 분석될 수 있다.

앞으로 NCP 가 최적의 성능을 낼 수 있는 SSRAM 인터페이스 환경에 적용하여 성능 향상 정도를 측정할 예정이며, 향후 네트워크 프로세서와 연동하고 다중 채널을 사용하여 Layer 4 는 물론 Layer 7 까지 테스트 하여 세부 분석에 적합한 시스템의 중요 엔진으로 연구해야 할 것이다.

참고문헌

[1] Pankaj Gupta and Nick McKeown "Algorithms for Packet Classification," IEEE Network, vol: 15:2, pp. 24-32, March/April 2001

[2] V.Srinivasan and G.Varghese "Fast and Scalable Layer 4 switching" Proceedings of the ACM SIGCOMM'98, pp. 191-202, 1998

[3] Sundar Iyer, Ramana Rao Kompella, and Ajit Shelat "ClassiPI: an Architecture for Fast and Flexivle Packet classification" IEEE Network, pp33-41, March/April 2001

[4] PMC-SIERRA, "ClassiPI Network Classification Processor Data Sheet," www.pmc-sierra.com, 2001

[5] 박우종 "고속 패킷 처리를 위한 IP lookup scheme" 대한전자공학회 추계종합학술대회 논문집(1), pp. 213-216 2000 년, 11 월

[6] T. V. Lakshman and Dimitrios Stiliadis. "High speed policy-based packet forwarding using efficient multi-dimensional range matching." Proceedings of ACM SIGCOMM, pages 203--214, September 1998.

[7] 이찬구, 김대영, "10 기가비트 이더넷 기술동향," 한국통신학회논문지 16 권 12 호 pp. 59-69, 1999 년 12 월

[8] 최준호, 김판구 "네트워크상에서 바이러스 차단을 위한 방화벽 시스템의 설계 및 구현" 정보처리학회 논문지 C v. 8, pp. 397-404, 2001 년 08 월

[9] Passive Measurement and Analysis project, National Laboratory for Applied Network Research. http://moat.nlanr.net/PMA