방화벽의 세션 테이블 관리기법 비교연구

고재현*, 정기현*, 최경희**
*아주대학교 전자공학과
**아주대학교 컴퓨터공학과
e-mail : pharos99@gmail.com

A Study on Firewall for Session Table Management Mechanism

Jae-Hyun Koh*, Gi-Hyun Jung*, Kyung-Hee Choi**
*Dept. of electronic Engineering, Ajou University
**Dept. of Computer Engineering, Ajou University

요 약

본 논문은 최근의 네트웍 장비의 기본적인 기능인 stateful inspection 을 지원하기 위해 생성되는 세션 테이블들의 구조와 그 효율성을 확인한다. 그를 위해 LINUX, FreeBSD, OpenBSD 등의 운영체제에서 사용되고 있는 방화벽 소프트웨어들의 세션 테이블 구조 및 특징을 확인하고 실제 실험을 통해 구조적 특징 및 트래픽의 지역성이 테이블의 탐색에 걸리는 오버헤드를 줄이는 데에 어떠한 영향을 미치는지 실제 실험하였으며, 트리 구조를 가지는 세션 테이블이 worst case 시의 테이블 탐색시간을 줄여줌으로써 전체적인 패킷 처리시간을 줄여줄 수 있는 구조임을 확인한다.

1. 서론

네트워크 기술의 빠른 발전에 따라 네트워크 서비스의 종류 또한 다양화 되어가고 있으며, 이에 따라 네트워크 서비스를 제공하는 장비들의 성능이 중요한 이슈가 되어 왔다. 방화벽이나 부하 분산기(load-balancer)등의 장비는 서비스의 필요에 의하여 세션이라고 하는 정보를 저장 및 관리해야 할 필요가 생겼으며, 이에 따라 많은 양의 세션을 저장 관리하기 위한 오버헤드 또한 늘어나고 있다. 이러한 오버헤드에 효과적으로 대처하기 위해 복수개의 패킷 처리장치를 가지는 네트웍 프로세서[11]를 사용하여 패킷 처리 속도를 높이거나 라우팅 테이블의 구조 개선[12]이나, 필터링을 룰들의 최적화된 적재[13]등의 패킷 처리 속도를 높이기 위한 노력이 있어왔다.본 논문에서는 이러한 노력과 더불어 세션 정보를 저장 관리하는 기법들에 대하여 초점을 맞추고 있다.

본 논문의 2 장에서는 세션을 유지 관리하는 대표적인 공개 방화벽에 대하여 알아보고, 이러한 방화벽들의 세션 관리 기법에 대하여 알아본다. 3 장에서는 성능 평가에 사용된 트래픽에 대해 알아보고 이와 함께, 실제 실험에 사용될 세션 테이블 구조에 대한 기술, 그리고 성능평가 분석을 하고, 4 장에서는 결론을 짓도록 한다.

2. 공개 방화벽들의 세션 관리 기법

<표 1> 공개 방화벽들의 사용 현황

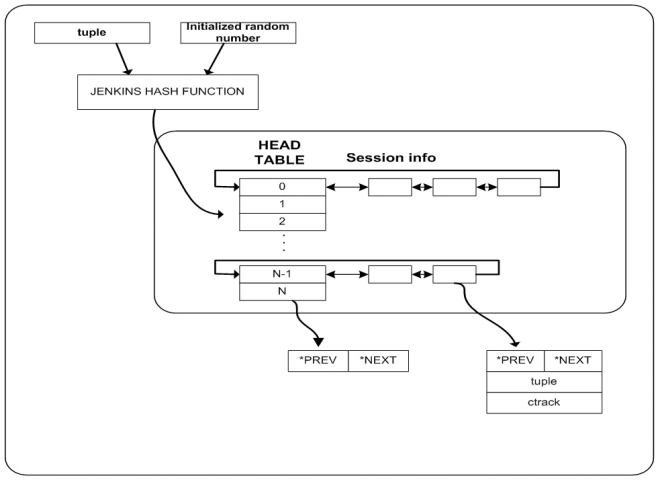
OS	*IPFILTER*	*IPFW*	*IPTABLES*	*PF*
Darwin	Yes	Yes		
DragonFly BSD	Yes	Yes		Yes
FreeBSD	Yes	Yes		Yes
Hewlett-Packard UP-UX	Yes			
SGI IRIX	Yes			
Linux			Yes	
Mac OS X	Yes	Yes		
NetBSD	Yes			Yes
OpenBSD				Yes
QNX	Yes			
Sun Solaris	Yes			

표 1 은 본 논문에서 참고된 방화벽들의 사용 현황에 대하여 요약한 것이다. IPFilter 를 제외한 방화벽들은 운영체제에 포함되거나, 밀접한 관련을 가지고 있으며, 현재에도 활발히 사용되고 있는 방화벽 소프트

웨어 들이며, 지속적인 갱신이 이루어 지고 있다.

2.1 IPtables[1]의 세션 테이블 구조

Linux 에서 사용되고 있는 IPtables[1]모듈은 모듈의 형태로 사용 가능하며, Linux 운영체제를 사용하는 시스템에서 대표적으로 사용되는 방화벽 소프트 웨어이다. IPtables 는 세션 테이블을 conntrack 이라고 불리는 별도의 모듈로 관리하는데 이곳에 사용되는 세션 테이블 관리 기법을 살펴보면 다음과 같다.

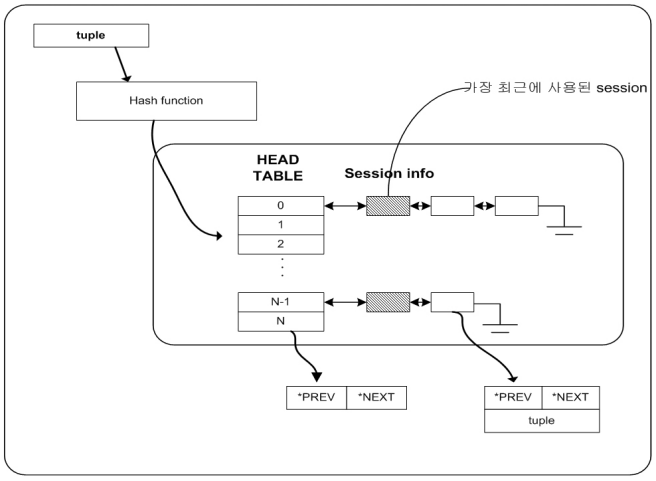


(그림 1) IPtables 의 세션 테이블 구조

Jenkins 해시 함수[8]를 사용하고 있으며 해시 함수에 의해 나온 색인 값에 따라 이중-연결 리스트(doubled-linked list)로 구현되어 있다. 세션의 추가가 리스트의 맨 앞쪽에서 일어나게 되는데 이를 통해 IPtables 의 세션 테이블이 트래픽의 지역성을 고려한 구조를 가지고 있음을 알 수 있다.

2.2 IP Firewall[2]의 세션 테이블 구조

IP Frewall 은 OpenBSD 진영에서 유지되고 있으며, 몇몇 운영체제로 이식되어 있다. IPtables 와 유사하게 구성되어 있으며, 그 구조는 다음과 같다.

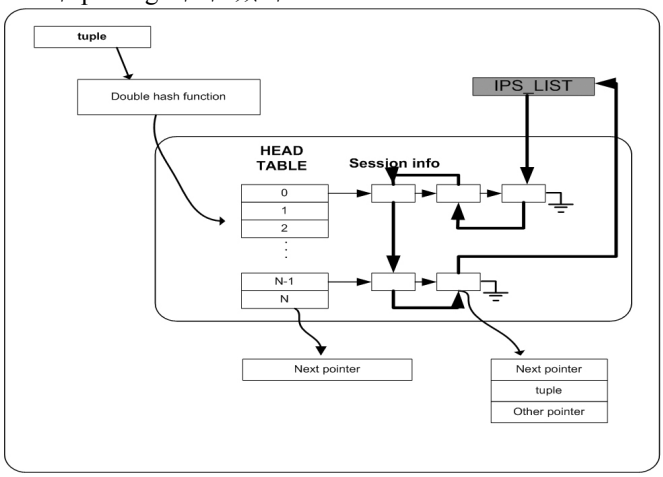


(그림 2) IPFirewall 세션 테이블 구조

거의 모든 구조가 IPTables 와 같지만, IP Firewall 의 경우에는 세션의 생성시 뿐만 아니라 세션이 사용되는 시점에도 리스트의 맨 앞으로 정보를 이동 시킨다. 이는 트래픽의 지역성을 가장 잘 고려한 경우라 할 수 있다.

2.3 IP Filter[5]의 세션 테이블 구조

OS independent 하게 개발되었으며 가장 많은 종류의 OS 에 porting 되어 있다.

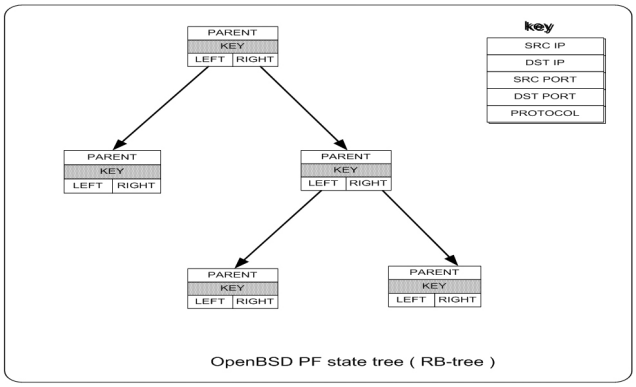


(그림 3) IPFilter 세션 테이블 구조

IPtables 와 다른점은 이중-연결 리스트가 아닌연결 리스트로 구현되어 있다는 점이며, 타임아웃과 같은 별도의 세션관리를 위한 연결 리스트를 유지하고 있다.

2.4 Packet Filter[4]의 세션 테이블 구조

OpenBSD 진영에서 개발하고 유지 관리되는 방화벽 소프트웨어로 다음과 같은 구조를 가진다.



(그림 4) Packet Filter 세션 트리구조

IPFirewall, IPtables, IPFilter 와는 다른 트리구조로 세션 테이블을 만들어 사용하고 있으며 RB-tree 구조를 가진다. 이는 balanced-tree 구조와 유사하며, 탐색시의 worst case 를 최소화 하는 구조를 가진다.

3. 성능 평가

3.1 실험에 사용된 traffic

실험에 사용된 트래픽은 DEC-PKT[7] 과 MRA[10] 트래픽으로 DEC-PKT 트래픽은 네시간에 걸친 Digital

Equipment Corporation 과 인터넷 사이의 트래픽이며, 이중 첫 트래픽을 제외한 트래픽이 실험에 사용되었으며, MRA 트래픽은 NLANR site 에서 제공하는 트래픽으로 OC12c (622Mbit/s)Pos link 에서 수집된 트래픽으로 IP, Protocol 헤더의 정보만을 제공한다[10]. 모두 트래픽의 IP, Protocol 헤더만을 제공하고 있으며 각각이 제공하는 형식을 분석하여 사용하였다.

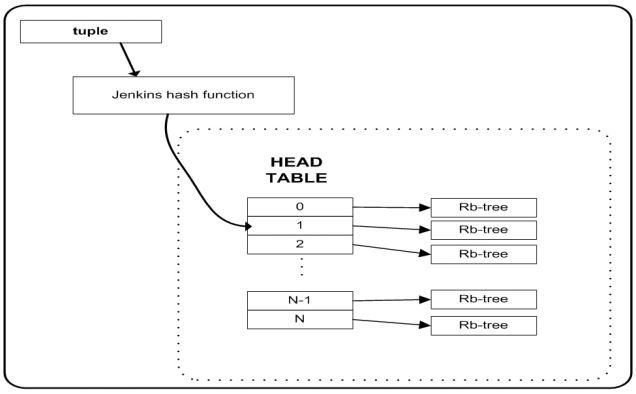
3.2 성능에 사용된 세션 관리 기법

2 장에서 설명한 방화벽의 테이블 구조를 기반으로 대표적인 두 가지 테이블 구조를 만들어 내어 실험에서 사용하였으며 각각의 장점을 혼합시킨 구조도 실험하였다. 사용된 구조들을 살펴보면 다음과 같다.

3.2.1 해시함수 & 이중연결 리스트 & 지역성

IPtables, IPFirewall, IPFilter 가 모두 해시와 연결 리스트 (혹은 이중-연결 리스트)를 사용한 구조를 사용하고 있으며, 이중 트래픽의 지역성을 가장 잘 적용시킨 IPFirewall 의 구조를 그대로 사용하였다.

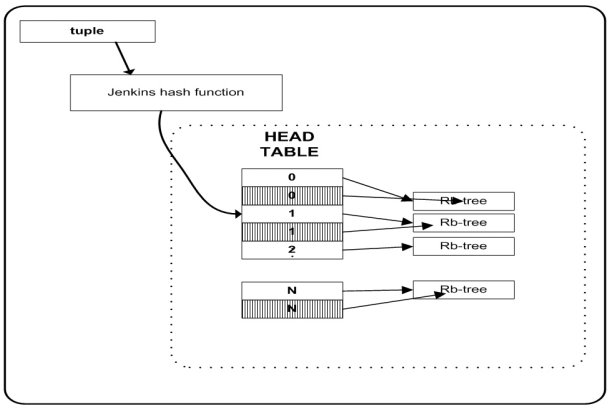
3.2.2 해시 & Rb-tree



(그림 5) 실험에 사용된 RB-tree 구조

Rb-tree 구조를 실험하기 위해 해시와 Rb-tree 를 연결 시킨 구조를 실험하였다.

3.2.3 해시 함수 & RB-Tree & 지역성



(그림 6) RB-tree 에 지역성을 추가한 구조

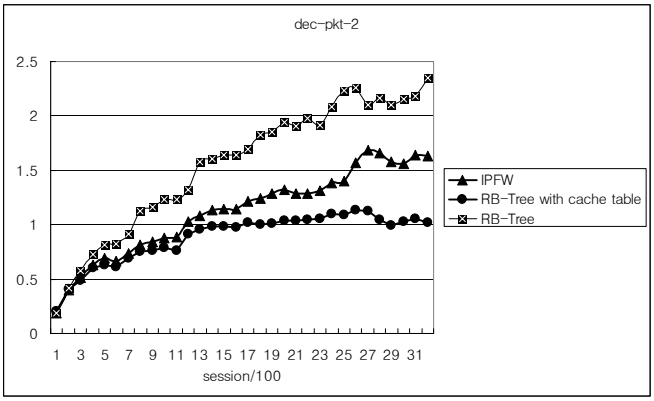
본 논문에서는 Rb-tree 구조에 트래픽의 지역성을 적용시킨 그림 6 과 같은 구조도 실험하였다. 빛금친 부분은 각 tree 마다 가장 최근에 사용된 세션 정보를 가리키게 되어 있으며 세션 탐색시에 이 정보를 먼저 찾아 봄으로써, 트래픽의 지역성을 고려한 구조를 가진다.

모든 구조들은 헤드 테이블의 수를 1024 개로 하고 실험하였다.

3.3 성능 평가 결과

그림 7 에서 10 까지는 각각의 트래픽에 대한 테이블 구조 실험 결과이다. x 축은 세션 비교 횟수이며, y 축은 관리되는 총 세션 수이다. 즉 비교 횟수가 낮을수록 좋은 성능을 발휘하는 구조라 할 수 있다.

3.3.1 테이블 구조 실험 결과



(그림 7) dec-pkt-2 트래픽 실험 결과

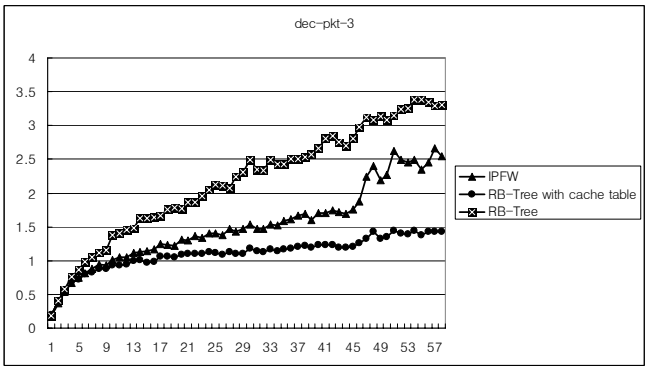
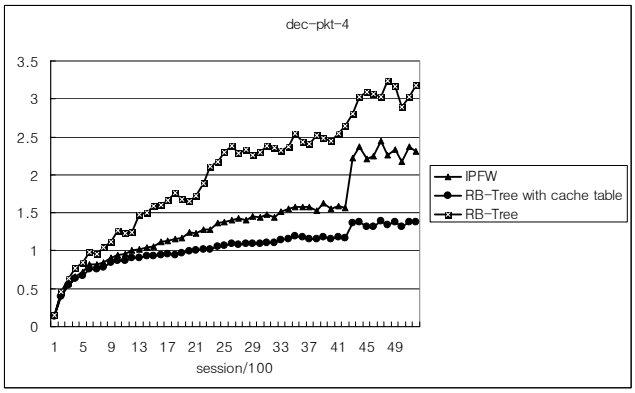
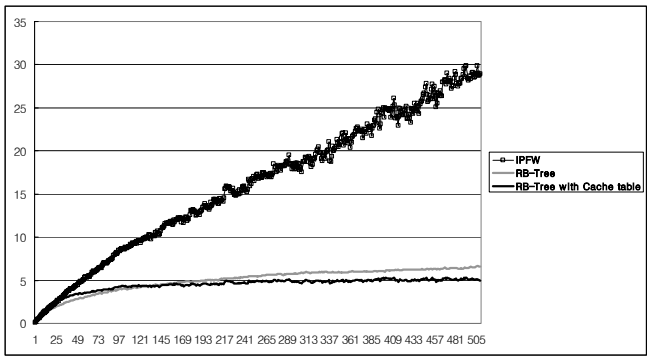


그림 8. dec-pkt-3 트래픽 실험 결과



(그림 9) dec-pkt-4 트래픽 실험 결과

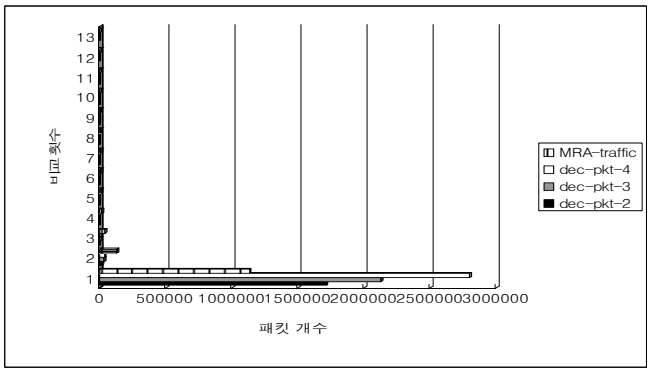


(그림 10) MRA 트래픽 실험 결과

DEC-PKT 트래픽은 상대적으로 관리되는 세션의 수가 적으며, 최종적으로 트래픽의 지역성을 고려한 Rb-tree 구조가 가장 높은 성능을 발휘함을 알 수 있다. 또한 트래픽의 지역성을 고려한 RB-tree 와 일반 RB-tree 를 비교함으로써 트래픽의 지역성을 고려한 경우가 약 60%정도의 성능향상 효과를 보임을 알 수 있다. 그림 10 은 MRA 트래픽 실험 결과인데, MRA 트래픽의 경우 유지되는 세션의 수가 높기 때문에 트리 구조를 사용하는 것이 해시와 연결 리스트로 구성된 테이블 구조를 사용하는 것보다 많은 장점을 가짐을 알 수 있으며, 이 경우에도 트래픽의 지역성을 고려한 트리 구조가 좀더 나은 결과를 보임을 알 수 있다.

3.3.2 트래픽의 지역성 실험

다음의 그림은 트래픽의 지역성을 측정한 것이다. 이는 세션을 찾게 되는 경우 몇 번의 비교만에 원하는 세션을 찾게 되는가를 나타낸 것으로 살펴보면 다음과 같다.



(그림 11) 지역성 성향 분석

지역성을 고려한 트리 구조가 가지는 단점은 하나의 정보만을 가지고 있게 된다는 점인데, 그림 11 을 보면 거의 모든 트래픽의 경우에서 첫 세션정보가 가리키는 정보가 찾으려고 하는 세션 정보일 확률이 가장 크다. 따라서 지역성을 고려한 트리 구조가 가지는 단점을 그리 크지 않다는 것을 알 수 있다.

4. 결론 및 향후 과제.

본 논문은 세션 테이블 구조에서 트래픽의 지역성과 세션 테이블 구조의 효과를 공개 방화벽들의 구조를 토대로 실험하여 지역성과 트리구조를 동시에 고려한 경우가 가장 좋은 결과를 가져온다는 사실을 확인하였다. 향후 과제로는 이러한 구조를 토대로 좀더 높은 성능을 낼 수 있는 세션 테이블 구조에 대하여 연구가 진행될 것이다.

5. 참고 문헌

[1] www.netfilter.org
[2] www.freebsd.org
[3] www.nlanr.com
[4] www.OpenBSD.org
[5] http://coombs.anu.edu.au/~avalon/
[6] Neeraj Gulati, “Local Area Network Traffic Locality: Characteristics and Application” July, 1992
[7] http://ita.ee.lbl.gov/html/contrib/DEC-PKT.html
[8] B. Jenkins. Algorithm alley: Hash Functions. Dr. Dobb’s Journal, September 1997
[9] Jahangir Hasan, Satish Chandra, Vijaykumar, T.N., “Efficient use of memory bandwidth to improve network processor throughput”, Computer Architecture, 2003. Proceedings. 30th Annual International Symposium on 9-11 June 2003 Page(s):300 – 311
[10] http://pma.nlanr.net/PMA/Sites/MRA.html
[11] Intel IXP2800 Network Processor Hardware Reference Manual, May 2003
[12] Marcel Waldvogely, George Varghesez, Jon Turnerz, Bernhard Plattnery, “Scalable High Speed IP Routing Lookups”, ACM SIGCOMM '97, pp. 25-36, Cannes, September 1997
[13] 강진원, 정기현, 임강빈, 최경희, 최준혁, , "네트워크 프로세서에서 효율적인 cross-product table 을 이용한 패킷분류기 구현" 한국정보처리학회 추계학술발표대회, 한국정보처리학회, 11(2), pp.1401-1404, 2004.11