트래픽 측정에서 Long-lived flow 구분을 위한 플로우 엔트리 형성시 메모리 절약 방법

박종서, 김철우, 장주욱
서강대학교 전자공학과
e-mail : netofpos@eeca1.sogang.ac.kr, kps0517@eeca1.sogang.ac.kr, jjang@sogang.ac.kr

Reducing Entry Flow in Cache Memory Measuring Traffic and Long-Lived Flows

Jong Suh Park, Cheol-Woo Kim, Juwook Jang
Dept. of Electronics , Sogang University

요 약

본 논문에서는 트래픽 측정시 Long-lived flow 구분을 위한 플로우 엔트리 형성시 메모리를 절약할 수 있는 방법을 두가지 방식, Sample and hold 와 Multistage Filter 알고리즘을 가지고 실험을 통해 증명하였다. 라우터에서 Long-lived flow 구분시, 측정구간의 80%가 지난 뒤에 들어오는 새로운 플로우을 엔트리에 등록하지 않는 특성을 가지므로서 결과적으로 메모리를 20% 절감할 수 있는 방법을 COMNET III 네크워크 시뮬레이션을 통해서 분석하였다.

1. 서론

네트워크 운영 및 관리에서 링크 용량 예측, 공급 및 증대 그리고 DoS(Denial Of Service)예방에 중시를 두고 있다. 즉 이런 점들을 정확히 해결하기 위해선 네트워크 트래픽의 정확한 측정이 필요하다[8]. 네트워크에 유통되는 패킷 정보를 수집하고 분석함으로써 이용자별, 시간대별, 프로토콜별, 응용별로 특정 네트워크를 출입한 패킷을 파악하여 그 네트워크의 특성을 파악할 수 있게 한다. 특히 망을 운영하는 입장에서는 링크에서 영향을 끼치는 Large Flow 에 유념하게 된다[10]. Large Flow 는 일반적으로 링크에서 1%이상을 차지할 때 고려를 하게 된다. Large Flow 의 정확한 분석을 위해선 백본망 내부의 트래픽 상황[6,7]을 모니터링 해야하는 툴들이 필요하고 현재 많은 툴들이 소프트웨어적으로 구현되어 상품화되고 있다. 그 중에서도 저장하는 방법을 기존에는 소트웨어적으로 처리한 반면에 현재는 캐쉬 메모리를 사용하여 하드웨어적으로 구현하여 성능을 향상시켰다. 하지만 여전히 실시간성 및 확장성에는 한계가 있고 메모리사용에 따른 바용의 문제가 따르게 된다. IETF RTFM(Real Time Flow Measurement)[9] 워크그룹에서도 이런 점들을 해결하기 위해 연구가 진행중이다.

본 논문에서는 이러한 부분을 해결하기 위해 제시된 Sample and hold 및 Multistage Filter 방식에서 플로우 구분을 위한 엔트리 형성시 메모리를 절약할 수 있는 방법을 시뮬레이션을 통해 증명하였다. 라우터에서 Long-lived flow 구분시, 측정구간의 80%가 지난 뒤에 들어오는 새로운 플로우을 엔트리에 등록하지 않는 특성을 가지므로서 결과적으로 메모리를 20% 절감할 수 있다는 것을 시뮬레이션을 통해 증명하였다.

2. 관련 연구

현재 많은 네트워크 모니터링 소프트웨어들중에서 시스코에서 출시된 "Net Flow"를[2] 보면 주기적으로 패킷을 샘플링을 통해 저장하는 방식을 사용하고 있다[3]. 플로우의 구분을 위해 IP Protocol Type, Type of Service 그리고 Input interface identifier 도 같이 사용하면서 플로우의 정보를 제공한다. 패킷을 캐쉬를 이용하여 동작하는데 특정 패킷이 기존에 존재하는 플로우에 속하는지 여부에 따라 캐쉬에 새로운 플로우 엔트리를 생성할지를 결정한다. 이 방법은 캐쉬와 패킷 저장에 대한 판단을 하는 장점을 가지고 있지만, 샘플링 주기와 캐쉬의 크기에 의존해야하는 맹점을 가지고 있다. 이와 관련된 향상된 방법으로 제시된 Sample and hold 방식과 Multistage Filter 방식이 있다[1].

본 논문에서는 기존의 연구되었던 두방식을 개선시킬 수 있는 방법을 제시하고 이에 따른 시뮬레이션을

결과를 통해 개선된 부분을 설명하고자 한다.

3 장에서는 메모리 절약을 위한 방법에 대해서 설명하였고, 4 장은 시뮬레이션에 따른 결과 그리고 5 장 결론으로 이루진다.

3. 플로우 엔트리 형성시 메모리 절약 방법

기존 sample and hold 및 multistage filter 는 전 구간에 대하여 새로운 모든 플로우에 대하여 엔트리를 등록한다. 하지만 측정구간 T 시간동안 샘플링하게 되면 실제 large-flow 가 될 가능성이 있는 플로우는 측정구간의 처음에 위치한 엔트리에 등록된 프롤우일 가능성이 높다. 따라서 측정구간의 αT 의 시간이 지난 뒤에 들어오는 새로운 플로우에 대하여 새로운 엔트리를 등록하지 않음으로써 메모리를 절약할 수 있다.

새로운 플로우가 들어 왔을 경우 등록구간 R 은

$$R = T \times \alpha \qquad (1)$$

가 된다.(단, $0.5 \leq \alpha \leq 1$)

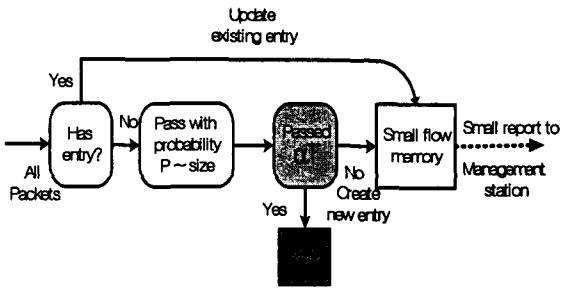
즉, 등록 구간 R 이 지난 다음에 들어오는 새로운 플로우는 새로운 엔트리를 생성, 등록하지 않는다. 이때 하나의 플로우는 하나의 TCP 컨넥션으로 연결 되어 있으며, source 와 destination 이 같아도 하나의 TCP 커넥션이 끊어진 후의 연결은 다른 플로우로 간주하였다.

3.1 Sample and hold 방식 및 제안된 엔트리 메모리 절약방법

Sample and hold 는 모든 패킷에 대하여 일정 확률 P 로 샘플링을 하고, 이때 엔트리에 등록되지 않은 새로운 플로우이면 새로운 엔트리를 등록하기 위하여 메모리을 할당하고, 등록한다[1]. 그리고 확률 P 로 샘플링한 플로우에 대해서는 샘플링 구간의 샘플링 확률 P 에 걸리지 않더라도 엔트리에 등록되어 있으므로 엔트리만을 갱신한다.

제안 방법은 샘플링 구간에 새로운 플로우가 들어오게 되면 새로운 엔트리를 등록하기 위하여 등록구간 R 이 지난 뒤에 들어오는 새로운 플로우인지 판단한 후 구간 R 이 지나지 않았으면 새로운 플로우에 대하여 엔트리를 등록하고 그렇지 않으면 등록하지 않고 버린다.

그림 1 은 이에 대한 알고리즘으로 측정구간 T 에서 αT 이후에 들어오는 새로운 플로우는 엔트리를 등록시키지 않고 버리는 것을 보여준다.

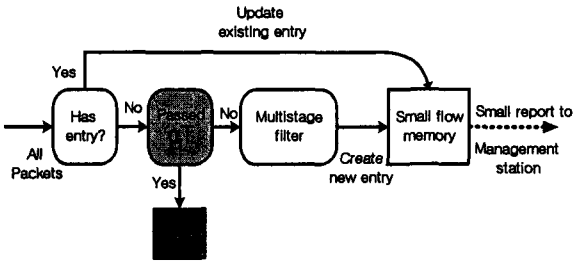


<그림 1> Sample and hold 방식에서의 엔트리 메모리 절약 Flow

라우터를 지나는 패킷이 poisson 하므로[4] 구간 R 이 지난 뒤에 새로운 플로우에 대하여 등록하게 되면 예상 메모리 이득은 $R = T \times \alpha$ 이므로 $(1-\alpha)$값만큼의 이득을 얻을 것으로 보여진다.

3.2 Multistage filter 의 엔트리 메모리 절약

라우터를 지나가는 모든 패킷에 대하여 엔트리가 있는지 여부를 확인하고, 엔트리가 등록되지 않은 새로운 플로우에 대해서는 새로운 엔트리를 등록하고 업데이트한다[1]. 이때 αT 이후에 들어오는 새로운 플로우가 모든 패킷에 대하여 검사하므로 정확하게 새로운 플로우를 찾아 등록시키므로 더 많은 메모리 이득을 볼 수 있을 것으로 보인다.



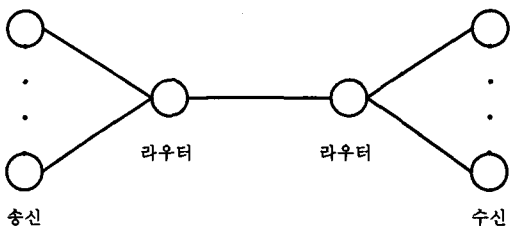
<그림 2> Multistage filter 방식에서의 엔트리 메모리 절약 Flow

샘플링하지 않고 모든 패킷에 대하여 등록구간 R 이 지난 뒤에 새로운 플로우가 들어 오는 지만 판단하면 된다. 메모리 이득은 sample and hold 와 같이 $R = T \times \alpha$ 이므로 $(1-\alpha)$값만큼의 이득을 얻을 것으로 예상된다.

4. 실험 및 결과

실험은 COMNET Ⅲ를 이용하여 실험하였으며, 그림 3 과 같이 토폴로지를 구성[5]하여, sample and hold 및 multistage filter 에 똑 같은 조건으로 flow 를 발생시켰다.

패킷 생성은 모든 송신단에서 Poisson 하게 했으며, 플로우의 길이 또한 특정한 송신단에서 많이 발생 되지 않도록 랜덤하게 발생시켰다. 패킷 도착 간격 시간(interarrival)도 랜덤하게 했다. 같은 송신단에서 같은 수신단으로 보내는 플로우라 하더라도 하나의 TCP 커넥션을 하나의 플로우로 보았다. 이때, 측정구간과 α 값을 달리하며 각각의 결과를 다음과 같이 얻었다.

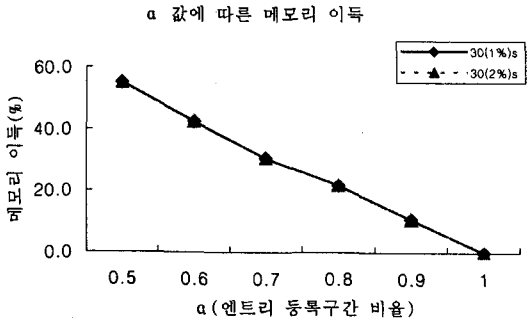


<그림 3> 시뮬레이션 토폴로지

4.1 Sample and hold의 결과

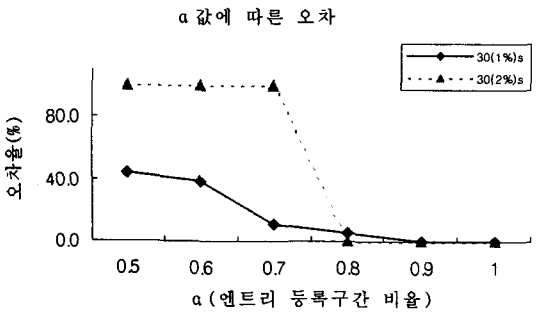
Sample 구간 0 에서 30 초 사이에 10%의 확률로 샘플링 했을 경우 총 4376 개의 패킷 중 2312 개의 패킷이 샘플링 되었다. 이때 large flow 를 총 트래픽의 1%로 하게 되면 패킷이 24 개이상 들어오는 플로루가 large 플로우가 되어 18 개의 large flow 를 찾을 수 있었으며 2%인 경우에는 1 개의 large flow 를 찾을 수 있다.

그림 4 는 α값에 따른 메모리 이득을 보여준다. 같은 트래픽에 대해서 large-flow 에 대한 기준만을 달리하였으므로 1%일 때와 2%일 때가 같은 결과가 나오는 것을 알 수 있다. 또한, α값의 변화에 따라 선형적으로 감소하는 것을 볼 수 있다. α가 1 인 경우는 기존 알고림즘에 제안 방법이 적용 되지 않은 것과 같다.



<그림 4> Sample and hold 에서 α 값에 따른 메모리 이득결과

그림 5 는 α값에 따른 오차로 large-flow 가 적은 경우 오차가 급격히 변화하는 반면 많은 경우에는 서서히 변화 하는 것을 볼 수 있다. 그래프에서 α값이 0.8 일 때, 1%를 large-flow 라 한 경우는 오차가 5.6%이고, 2%로 한 경우에는 0%임을 확인하였다.

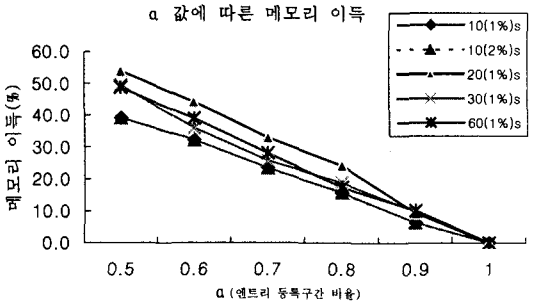


<그림 5> Sample and hold 에서 α 값의 따른 오차결과

4.2 Multistage filter의 결과

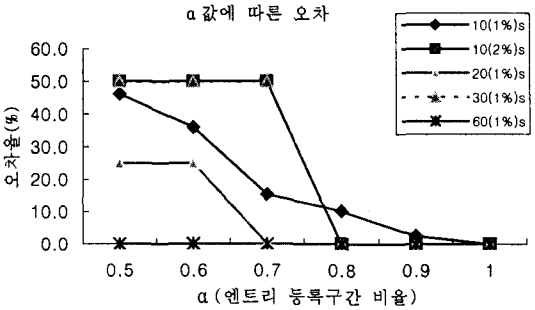
모든 패킷을 확인 한 후 엔트리를 갱신하기 때문에 sample and hold 보다는 좀더 정확하게 large-flow 를 찾을 수 있다. 반면에 좀더 많은 저장 공간을 필요로 한다. 1 분 동안 총 1024 개의 플로우가 흐르는 환경에서 각 측정 구간 10, 20, 30, 60 초에 대한 그래프로 large-flow 를 1%으로 하고 측정하였다. 이때, 구간 10 초만 1%와 2%으로 나누어 확인하였다.

그림 6 은 α값에 따른 엔트리를 형성시 메모리 이득을 보여준다.



<그림 6> Multistage filter 에서 α 값에 따른 메모리 이득결과

그림 7 은 α값의 따른 오차로 α가 0.8 일 때 오차가 거의 없음을 볼 수 있다. 측정구간을 10 초로하고 large-flow 를 전체 트래픽의 1%로 하면 large-flow 의 수가 많아져 large-flow 의 분포가 넓게 나타나므로 α값 0.8 에서 10%의 오차를 볼 수 있었다. 반면에 2%로 하였을 경우에는 0 가 됨을 확인 하였다. 전체적으로 α값이 0.8 일 때 대체로 오차가 적으며 이득 또한 20%정도를 얻을 수 있음을 확인 하였다.



<그림 7> Multistage filter 에서 에서 α 값의 따른 오차결과

5. 결론

제안된 알고리즘의 실험 결과에서 볼 수 있듯 sample and hold 와 multistage filter 의 두 경우에 대해서 측정 구간의 80%가 지난 뒤에 새로 들어오는 플로우에 대해 새로운 엔트리를 형성하지 않고 측정하여도 오차가 없을 확인하였다. 이때 약 20%의 메모리의 이득이 있음을 확인 시켜준다. 또한 large-flow 에 대한 기준에 따라 오차율이 달라 짐을 볼 수 있는데 대체로 기준이 적은 경우 오차가 발생함을 보여 주었다.

대체로 측정구간을 30 초로 한 경우에 대하여 타당한 결과를 보인다. 이는 추후과제로 각 트래픽량에 따라 large-flow 의 기준을 어느 정도로 하는가에 대한 것과 트래픽량에 따라 엔트리 형성시 메모리 이득이 어느 정도 되는지 알아 보아야 한다.

참고문헌

[1] Cristian Estan, George Varghese, New Directions in Traffic Measurement and Accounting. ACM SIGCOMM 2002.

[2] Cisco netflow http://www.cisco.com/warp/pubplc/732/tech /netflow.

[3] Sampled netflow. http://cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120limit/120s/120s11/12s_sanf.htm..

[4] David McDysan. QoS & Traffic Management in IP & ATM Networks. 2000.

[5] COMMNET manual 2.1.1,2001

[6] N. Duffield, C. Mills, and G. Ruth. Traffic flow measurement : Architecture. RFC 2722, Oct. 1999.

[7] A. Feldmann et al. Deriving traffic demands for operational IP networks: Methodlogy and experience. In ACM SIGCOMM, Aug. 2000.

[8] R, Mahajan et al. Controlling high bandwidth aggregates in the network.

[9] N. Brownlee, C. Mills, and G. Ruth. Traffic flow measurement: Architecture. RFC 2722, Oct. 1999.

[10] W. Fang and L. Peterson. Inter-as traffic patterns and their implications. In IEEE GLOBECOM, Dec. 1999.