TMA(Traffic Measurement Agent)를 이용한 인터넷 응용 트래픽 분류[1)]

윤성호*, 노현구*, 김명섭*
*고려대학교 컴퓨터정보학과
e-mail:{hiption, navvy84, tmskim}@korea.ac.kr

Internet Application Traffic Classification using Traffic Measurement Agent

Sung-Ho Yoon*, Hyun-Gu Roh*, Myung-Sup Kim*
*Dept of Computer Information Science, Korea University

요 약

네트워크를 사용하는 응용프로그램의 종류가 다양해지면서 네트워크 트래픽의 응용별 분류는 효율적인 네트워크 관리에 있어 그 중요성이 커지고 있지만, 오늘날 응용프로그램의 특징인 유동적인 포트번호 사용 및 패킷의 암호화 등은 트래픽의 응용별 분류를 더욱 어렵게 하고 있다. Well-known 포트기반의 응용별 분류방법의 단점을 극복하기 위하여 머신러닝 알고리즘과 Signature 기반 분석 방법들이 연구되고는 있지만 주장하는 높은 분석률에 비하여 실제 네트워크 트래픽에 적용하기에는 신뢰성이 부족하다. 본 논문에서는 일부 종단 호스트에 설치된 TMA(Traffic Measurement Agent)로 부터 수집한 응용프로그램의 트래픽 사용 정보를 기초로 하여 전체 네트워크 트래픽의 응용프로그램을 판별하는 응용 트래픽 분류 방법론을 제안한다. 제안된 방법론은 트래픽 플로우들의 상관관계를 이용하여 TMA 호스트 트래픽으로부터 TMA가 설치되지 않은 호스트에서 발생한 트래픽들의 응용을 판단하며, 분류된 결과에 대하여 높은 신뢰성을 보장한다. 제안된 방법론은 학내 네트워크에 적용하여 그 타당성을 검증하였다.

1. 서론

네트워크를 사용하는 응용프로그램의 종류가 다양해지면서 네트워크 트래픽의 응용별 분류는 효율적인 네트워크 관리에 있어 그 중요성이 커지고 있다. 그러나 오늘날 많은 응용프로그램들이 유동적인 포트번호를 사용하고 패킷을 암호화하여 전송하고 있어 네트워크 트래픽의 응용별 분류는 쉬운 문제가 아니다. 기존의 well-known 포트 기반의 응용별 분류방법의 단점을 극복하기 위하여 머신러닝 알고리즘[1]과 Signature[2] 기반 방법들이 연구되고는 있지만 그들이 주장하는 높은 분석률에 비하여 실제 네트워크 트래픽에 적용하기에는 여러 가지로 부족하다. 이는 시간과 장소에 따라 다양한 트래픽 특성을 보이는 네트워크 환경에 제안된 방법론을 일괄적으로 적용할 수 없고, 일단 적용되었다 하더라도 응용별 분류 결과를 검증할 수 없어 신뢰도를 예측하기 어렵기 때문이다.

본 논문에서는 기존 연구의 방법론이 실제 네트워크에 적용되었을 경우, 분석의 정확도를 검증하지 못하는 문제를 해결하기 위하여 대상 네트워크 내부의 일부 종단 호스트에 응용 프로그램의 트래픽 사용 정보를 모니터링 하기 위한 TMA(Traffic Measurement Agent)를 설치하고 이로부터 수집한 정보를 바탕으로 전체 네트워크 트래픽을 응용프로그램으로 판별하는 트래픽 분류방법을 제안한다. 즉, TMA가 설치된 일부 호스트를 이용하여 전체 네트워크 트래픽을 분석하는 것이다. 본 논문에서는 응용별 트래픽 분류를 트래픽을 발생시킨 응용프로그램의 이름을 기준으로 분류하는 것으로 정의한다. 이를 위하여 응용프로그램을 서비스 단위로 묶어 트래픽 분류 기준을 정의하였고, 이 기준에 따라 실제 트래픽에서 분석한다. 제안된 방법론은 기존 연구에서 초점을 맞춘 분석률의 향상 보다는 분석된 트래픽의 정확도에 초점을 맞추었다. 본 논문에서는 학내 인터넷 트래픽을 대상으로 제안된 방법론의 타당성을 검증하였다.

본 논문의 구성은 다음과 같다. 2장에서는 TMA의 동작원리와 수집하는 정보의 내용을 기술하고, 3장에서는 네트워크 트래픽의 분류 기준에 대해 정의한다. 4장에서는 TMA 정보를 바탕으로 트래픽 상관관계를 이용한 네트워크 트래픽 분석 알고리즘을 프로토콜별로 제시한다. 5장에서는 제안된 방법론의 성능 평가 및 결과를 기술하며 마지막으로 6장에서는 결론 및 향후 연구를 제시한다.

2. TMA(Traffic Measurement Agent)동작

정확한 트래픽의 분석을 위해서는 그 시간대의 응용프

1) 이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2007-331-D00387)

로그램이 사용하는 IP port의 실제 프로세스 정보를 수집하는 것이 중요하다. 이를 위해 TMA를 네트워크 내의 종단 호스트들에 설치해서 분석에 필요한 데이터를 수집한다.

<표 1> TMA 정보

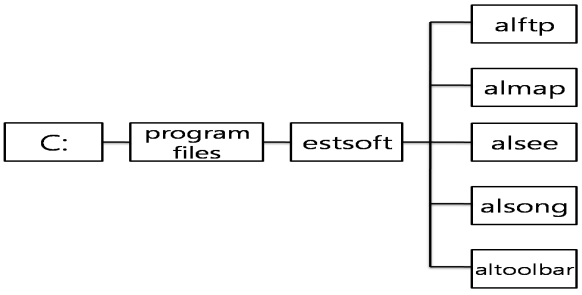
Process name
IP address(local, remote)
port number(local, remote)
state(start, continue, end, server)
protocol
path

TMA는 해당 호스트에서 실행중인 프로세스들이 사용하는 소켓정보를 주기적으로 수집하여 프로세스의 path 정보와 함께 지정된 서버로 제공한다. <표 1>은 TMA가 제공하는 정보들이다. 프로토콜이 TCP일 경우, <표 1>에서 제공하는 모든 정보를, UDP일 경우, remote IP와 remote port를 제외한 모든 정보를 보낸다.

3. 응용프로그램의 서비스 별 분류

네트워크 트래픽의 응용별 분류를 응용프로토콜별 분류로 보는 기존의 방식은 하나의 응용프로토콜이 다양한 응용 프로그램에서 사용되는 현재의 네트워크 환경에서 적합하지 않다. 따라서 본 논문에서는 응용별 분류를 해당 트래픽을 발생시킨 응용 프로그램의 이름을 기준으로 분류하는 것으로 정의한다. 개별 응용프로그램들은 다시 서비스 단위로 그룹으로 묶어 코드화함으로써 응용프로그램 단위 분류에서 생길 수 있는 문제들을 해결하였다. 이러한 서비스와 응용프로그램의 2단계 분류 기준은 동일한 이름을 가지는 응용들의 분류 오류를 방지할 수 있고, 개별 응용프로그램의 트래픽 사용 분포뿐만 아니라 서비스별 분포를 알 수 있어 트래픽 관리에 더 효율적이다.

서비스는 단일 목적을 가지는 개별 응용프로그램들의 집합이다. 즉, 응용 프로그램들을 개발하고 배포한 회사의 이름이 그 응용프로그램들의 서비스명이 된다. 본 연구에서는 TMA가 제공하는 응용프로그램의 path 속성을 이용하여 응용프로그램들의 서비스명을 자동으로 결정하고 코드를 설정하였다. 대부분의 응용들은 설치에 있어 계층적 디렉터리 구조를 가지므로, Common keyword 처리와 예외처리 후, 최상위 폴더 이름을 서비스 이름으로 지정한다.



(그림 1) estsoft 계층적 구조

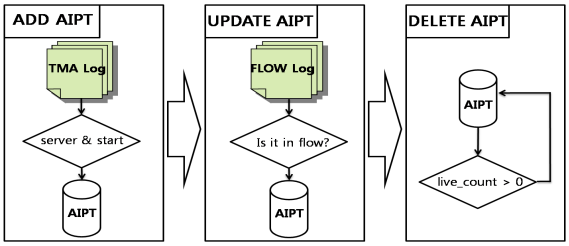
(그림 1)은 estsoft사에서 제공하는 응용프로그램들의 디렉터리 구조를 보여준다. 이 경우, “C:”, “Program files”는 common keyword이기 때문에 이를 제외한 최상위 폴더가 서비스명으로 결정된다. estsoft사에서 만든 5개의 응용프로그램은 "estsoft"라는 서비스명과 코드를 할당 받게 된다. 또한 각각의 응용프로그램들도 고유코드를 할당 받는다. 학내 인터넷 망을 이용해 응용프로그램들에 코드를 부여한 결과, 213개의 서비스 코드와 701개의 응용프로그램 코드를 할당하였다.

4. 분석 방법

TMA에서 수집되는 정보와 상관관계가 프로토콜별로 다르기 때문에 효과적인 분석을 위해 TCP와 UDP를 분리하여 분석한다. 본 논문에서는 단순히 TMA의 정보를 맵핑하는 Straight Forward Algorithm(SFA)을 비교분석용으로, 트래픽의 상관관계를 이용하여 non-TMA 호스트의 트래픽까지 분석하는 Traffic correlation Algorithm(TCA)를 정의한다. 본 논문에서 사용하는 플로우의 정의는 NG-MON[3]에서 정의한 5-tuple 정보(source IP, source port, destination IP, destination port, protocol)가 같은 패킷들의 모임이다.

4.1 TCP 분석 알고리즘

TCP는 하나의 서버에 여러 클라이언트가 통신을 하기 때문에 Application IP Port Table (AIPT)에 서버 호스트의 IP와 port 정보를 등록하고 이를 바탕으로 플로우의 응용프로그램 이름을 결정한다. 이러한 알고리즘을 통해 TMA가 설치되어 있지 않는 호스트의 트래픽도 분석할 수 있다. 따라서 항상 유효한 서버 포트정보를 AIPT내의 유지시키는 것이 매우 중요하다.

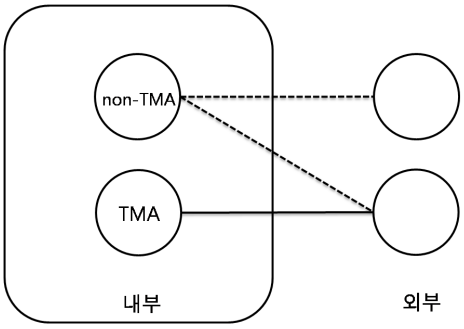


(그림 2) AIPT(TCP) 생성 메커니즘

(그림 2)은 AIPT(TCP)에 등록되는 서버 포트정보의 추가 삭제 메커니즘을 보여준다. 우선, (ADD) 현재 연결이 시작된 TMA 정보 중 새로 생성된 서버 포트정보를 추출하여 AIPT(TCP)에 추가시킨다. (UPDATE) AIPT(TCP)내의 서버 포트정보 중에서 실제 플로우에 존재하는 정보만 AIPT(TCP)에 유지 시킨다. (DELETE) TMA 정보에서 해당 서버 포트정보가 삭제되었고, 실제 플로우 분석에서 해당 서버 포트정보의 참조가 없으면 AIPT(TCP)에서 서버 포트정보를 삭제한다.

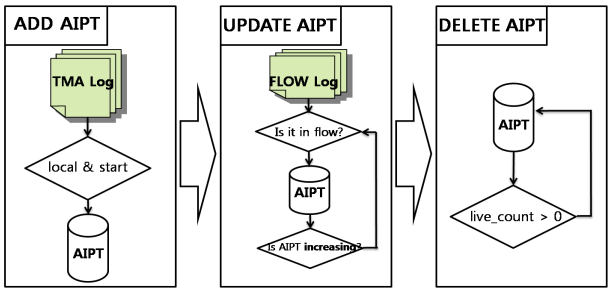
4.2 UDP 분석 알고리즘

UDP 통신은 서버 클라이언트 관계가 모호하기 때문에 TCP와 같이 서버정보를 이용할 수는 없지만 하나의 소켓으로 여러 호스트와 통신이 가능하므로 플로우 상관관계를 이용하여 응용프로그램을 결정할 수 있다.[3] 플로우 상관관계기반 분석은 TMA가 설치 된 호스트들의 UDP 호스트정보(IP, port, 프로그램이름)로부터 플로우의 연결을 추적해 TMA가 설치되지 않은 호스트들의 트래픽 플로우를 분석하는 것이다. (그림 3)은 UDP 상관관계기반 분석을 보여준다.



(그림 3) UDP 상관관계 기반 분석

내부의 TMA 호스트를 통해 외부 호스트 정보를 수집할 수 있기 때문에 그 외부 호스트와 연결된 내부의 non-TMA 호스트의 정보를 알 수 있다. 이 non-TMA호스트는 또 다른 외부 호스트를 찾는 정보가 된다.

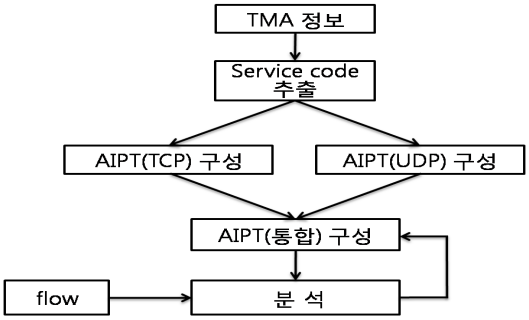


(그림 4) AIPT(UDP) 생성 메커니즘

(그림 4)은 AIPT(UDP)에 등록되는 호스트정보의 추가 삭제 메커니즘을 보여준다. (ADD) TMA 정보에서 UDP 소켓이 생성될 경우 UDP 호스트 정보를 AIPT에 등록한다. (UPDATE) TMA가 설치되어 있는 호스트와 실제 프로우를 비교하여 그 호스트와 연결되어 있는 non-TMA 호스트의 UDP 호스트정보를 추가한다. 새로 추가된 호스트를 포함한 정보를 이용하여 또 다른 호스트들을 추가시킨다. 이러한 과정은 새로 추가되는 호스트가 없을 때까지 반복한다. (DELETE) TCP와 같은 방법으로 삭제한다.

4.3 분석 방법

TMA를 이용한 플로우 분석은 (그림 5)와 같은 순서로 진행된다.



(그림 5) 플로우 분석 순서도

TMA정보 중 path 항목의 정보를 이용해 3장에서 설명한 서비스 단위와 응용프로그램 단위로 코드를 부여한다. 코드는 실제 분석 전에 생성하여 DB에 저장된다. TMA정보 중 프로세서 정보를 TCP, UDP 프로토콜별로 나누어 4.1장, 4.2장에서 설명한 방법으로 AIPT(통합)를 만들다. 생성된 AIPT(통합)를 실제 플로우와 맵핑하여 실제 플로우의 응용프로그램 이름을 결정한다. 또한 분석된 플로우 정보로 부터 AIPT(통합)의 정보를 갱신하거나 추가한다.

5. 실험 결과

이 장에서는 TMA의 정확도를 측정하고, 4장에서 설명한 알고리즘을 적용했을 경우와 그렇지 않을 경우를 비교, 분석 하며, TMA가 설치되어 있지 않은 호스트의 분석률과 정확도를 측정한다. 또한, 학내 네트워크에서 발생한 플로우를 분석하여 응용프로그램 별로 분석 결과를 보여준다. 이 모든 실험은 24시간 동안 1분마다 얻어진 값을 기준으로 하였다.<표 2>는 4장에서 설명한 알고리즘을 적용했을 때와, 그렇지 않을 때의 실험 결과를 개수로 나타낸 것이다.

<표 2> 실험결과

			TCA	SFA
f	f_d	f_d^T	2,120,403	1,302,574
		f_d^N	11,213,265	0
	f_u	f_u^T	1,070,712	1,888,541
		f_u^N	25,282,635	36,495,900

* f : flow
d : determined
u : undetermined
T : TMA host에서 발생시킨 flow
N : non-TMA host에서 발생시킨 flow

5.1 TMA 정확도 측정

분석되지 않은 플로우 중 TMA 호스트가 발생시킨 플로우의 개수를 확인하여 TMA 호스트가 발생시킨 전체 플로우에 대한 비율을 확인한다.

• Missing Rate(MR): TMA가 해당 호스트에서 실제 생

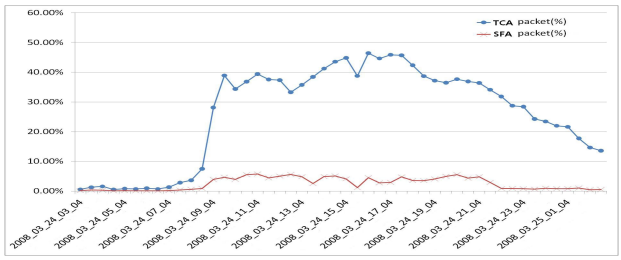
성하는 플로우를 놓치는 비율

$$MR = \frac{\sum f_u^T}{\sum f_d^T + \sum f_u^T} = 33.6\%$$

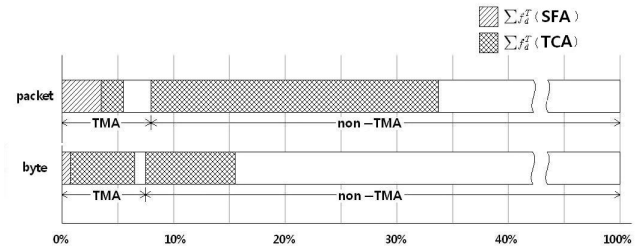
TMA는 순간적으로 생기는 트래픽과 비정상적으로 외부에서 발생하는 트래픽(스캐닝, 오류)은 인지하지 않으므로 MR을 가지고 있다. TCA의 성능을 극대화시키기 위해서 MR을 최소로 하는 TMA의 성능 개선이 필요하다.

5.2 알고리즘 성능 비교

(그림 6)은 알고리즘을 적용 했을 때(TCA)와, 그렇지 않을 때(SFA)의 실험 결과를 시간대 별로, (그림 7)은 하루 동안 분석한 결과를 플로우의 개수와 바이트 단위로 분석한 것이다.



(그림 6) 시간대 별 분석률



(그림 7) 전체 분석률

TCA는 AIPT를 이용하기 때문에 (그림 6)에서와 같이 실험 초반에는 SFA와 비슷하지만, AIPT에 등록되는 호스트가 많아지면서 SFA보다 향상된 분석률을 보인다. (그림 7)은 TCA가 non-TMA에서 발생한 플로우 중 30.7%를 분석한 것을 보여준다.

• Analysis Rate Using Algorithm(ARA): 알고리즘을 적용할 경우 분석률

$$ARA = \frac{\sum f_d(TCA)}{\sum f} = 33.6\%$$

• Analysis Rate NOT Using Algorithm(ARNA): 알고리즘을 적용하지 않고 TMA정보만을 가지고 분석했을 경우 분석률

$$ARNA = \frac{\sum f_d(SFA)}{\sum f} = 3.2\%$$

5.3 non-TMA 호스트 정확도 측정

분석되는 플로우 중 non-TMA 호스트가 발생한 플로우가 정확히 분류되어 있는지 확인하기 위해, 동일한 플로우를 대상으로 특정 호스트의 TMA정보를 제외시키고 분석하고, 이를 제외 시키지 않았을 때의 결과와 비교함으로써 non-TMA 호스트의 분석된 플로우의 정확도를 측정한다.

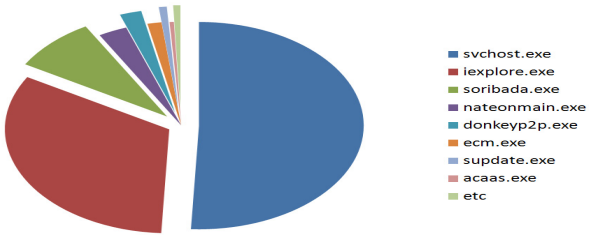
• non-TMA Host Accuracy(NTA): non-TMA 호스트의 알고리즘 적용시 정확도

$$NTA = \frac{Flow\ which\ is\ distributed\ by\ TMA}{\sum f_d^N} = 100\%$$

실제 분석된 플로우를 TMA 로그와 비교해 본 결과, 모두 정확하게 분석되었다.

5.4 KOREA Univ. 인터넷 망 분석

(그림 8)은 하루 동안 학내 네트워크를 TCA로 응용별 분석한 것을 보여준다.



(그림 8) 응용 별 분석

6. 결론 및 향후 과제

본 연구에서는 TMA를 이용한 TCA과 이를 이용한 실제 플로우 분석결과를 소개하였다. 3.2%의 플로우 정보를 이용해 33.6%의 플로우를 정확하게 분석하였다. 머신러닝 알고리즘과 Signature 기반 방법 보다는 분석률이 낮지만, 신뢰도 측면을 개선시켰다. 앞으로 TMA를 더욱 개선시키고, TMA와 플로우에서 제공하는 정보를 이용하여 좀 더 높은 분석률을 가지는 알고리즘 연구가 필요하다.

참고문헌

[1] 정광본, 최미정, 김명섭, 원영준, 홍원기, "ML 알고리즘을 적용한 인터넷 애플리케이션 트래픽 분류," The Committee on Korean Network Operations and Management (KNOM), Vol. 10, No. 2, Dec. 2007, .

[2] 박병철, 원영준, 김명섭, 홍원기, "자동화된 어플리케이션 레벨의 Signature 생성", Proc. of the Korean Network Operations and Management (KNOM) 2007, 제주대학교, Jeju, Korea, Apr. 26-27, 2007, pp.110-119.

[3] Se-Hee Han, Myung-Sup Kim, Hong-Taek Ju, and J. Won-Ki Hong, "The Arcitecture of NG-MON: A Passive Network Monitoring System," Proc. of the IFIP/IEEE Distributed Systems: Operations and Management (DSOM) 2002, Montreal, Canada, Oct. 21-23, 2002, pp. 16-27.