Cross-order 문제를 해결한 패킷 분포 기반 트래픽 분류 시스템

Traffic Classification Based on Packet Distribution without Cross-order Problem

안현민 [0], 함재현, 김명섭

고려대학교 컴퓨터정보학과

{queen26, jhham, tmskim}@korea.ac.kr

요 약

오늘날의 네트워크는 다양한 응용으로 인해 트래픽이 복잡 다양해지고 있고, 트래픽의 응용 별 분류의 중요성은 날이 갈수록 증가하고 있다. 기존에도 트래픽의 응용 별 분류에 대한 많은 연구가 이루어졌었으나 트래픽을 완벽하게 분류해내는 방법은 개발되지 않은 실정이다. 최근에는 플로우의 통계 정보를 이용한 방법이 많이 연구되고 있다. 하지만 트래픽 수집지점의 차이로 인해 트래픽의 통계적 특징이 달라지는 Cross-order 문제에 대해서는 다루어지지 않고 있다. 본 논문에서는 해당 문제에 대해 정의하고 해결 방안을 제시하며, 이를 적용한 통계적 특징 기반의 트래픽 분류 방법론을 제안한다. 제안하는 방법은 플로우 내 첫 N 개 패킷의 페이로드 크기, 전송 방향과 순서를 이용하여 응용 고유의 시그니쳐를 생성하고, 이를 이용하여 트래픽을 분류하는 방법이다. 제안하는 방법은 학내 망에서의 실험을 통해 검증한다.

Keywords: Application-level Traffic Classification, Application Identification, Signature-based Classification, Statistics-based Classification

1. 서론

응용계층 트래픽 분류의 중요성에 의해 현재까지 많은 연구들이 진행되어 왔다. 기존의 연구들은 포트 기반이나 페이로드 기반의 분석 방법들이 많이 제안되었다[1]. 인터넷이 지금과 같이 발달하기 전에 해당 연구들은 매우 효과적이고 정확하였다. 하지만 응용의 변화에 따라 트래픽이 변하기 때문에 오늘날의 응용 별 트래픽 분류는 더욱 힘들어지고 있다. 따라서 보다 정확한 응용 별 트래픽 분류를 위하여 다양한 분류 방법에 대한 지속적인 연구가 필요하다. 최근 몇 년간 암호화 된 트래픽 분류에 강하며 빠르고 정확한 분류 성능을 가진 플로우 통계 정보를 이용한 트래픽 분류 방법이[2,3] 많이 연구되었다. 하지만 이들은 플로우가 끝나고 통계 정보를 생성하기 때문에 실시간 트래픽 분류가 불가능하다. 이를 극복하기 위해 플로우의 처음 N 개 패킷만을 이용하여 트래픽을 분류하는 연구들이 진행되어 왔으나, 속성(Feature) 추출 계산의 오버헤드와 Machine Learning(ML) 알고리즘의 계산 복잡도가 높아 high-speed backbone 네트워크상에서의 실시간 트래픽 분류는 어려운 실정이다. 또한, 해당 연구들은 트래픽을 응용 프로토콜 단위로 분류한다. 동일한 응용 프로토콜을 사용하는 여러 응용들을 하나로 분류하기 때문에 개별 응용 단위로 적용되는 여러 네트워크 관리 및 운영 정책에 적용하기 힘들다. 무엇보다 수집 지점의 차이로 인해 Full-duplex TCP 트래픽의 통계적 특징에 큰 영향을 끼치는 문제에 대한 처리가 전무하다.

[1] 본 연구는 BK21 플러스 사업 및 2012 년 정부(교육과학기술부)의 재원으로 한국연구재단(2012R1A1A2007483)의 지원을 받아 수행된 결과임.

Full-duplex TCP 세션은 동시에 양방향으로 데이터를 송수신 할 수 있다. 이때 전송 경로 중간에서 패킷 교차가 일어나는데, 종점 호스트에서는 응용 계층에서의 트래픽 재정렬로 인해 패킷의 동시 전송이 문제가 되지 않으나 트래픽 수집 지점에서는 패킷 교차로 인해 순서가 뒤바뀌고 트래픽의 통계적 특징에 영향을 끼치며, 일관성을 잃게 되는 문제가 생긴다. 본 논문에서는 이러한 문제를 Cross-order 라 정의하고 해결 방안을 제시한다. 또한, 이를 적용한 통계적 특징 기반 트래픽 분류 방법을 제안한다. 본 논문에서 제안하는 방법은 수집된 응용 플로우의 패킷 분포를 이용해 고유한 시그니쳐를 추출하고, 분석 대상 플로우와의 유사도 계산을 통해 응용 트래픽을 빠르고 정확하게 분류하는 방법이다. 시그니쳐의 추출은 응용 플로우의 Cross-order 해결 단계, 각 플로우 내 처음 N 개 패킷의 크기, 순서 및 방향, 즉 플로우 내 패킷의 분포 정보를 이용한 특징 벡터화 단계, 플로우 특징 벡터들의 유사도에 기반한 동일 프로세스 플로우들의 그룹핑 단계, 그리고 그룹들의 최적화 단계를 통하여 이루어진다. 트래픽 분류는 입력되는 패킷을 이용한 플로우 생성 단계, 생성된 플로우들의 Cross-order 해결 단계와 특징 벡터화 단계를 거친 뒤 시그니쳐와의 유사도 매칭을 통해 분류된다.

본 논문에서 제안하는 방법은 통계 정보를 수집하는 범위를 플로우의 처음 N 개 패킷으로 한정하여 실시간 분류가 가능하다. 또한, 속성으로는 플로우 내 패킷의 전송 순서, 방향과 페이로드 크기만을 사용하기 때문에 빠르게 속성을 추출할 수 있으며, 트래픽 분류를 위해 필요한 유사도 검사는 단순 산술계산으로만 이루어지기 때문에 계산 복잡도가 낮아 대용량의 트래픽을 다루는 high-speed backbone 네트워크 상의 실시간 트래픽 분류에도 문제없이 적용 가능하다. 또한, 개별 응용을 단위로 트래픽 분류를 행하기 때문에 개별 응용 단위로 적용하는 네트워크 관리 및 운영을 위한 정책들에 적용할 수 있다. 마지막으로, 트래픽 수집지점의 차이로 인해 발생하는 Cross-order 해결로 정확도를 더욱 높인다.

본 논문은 다음과 같은 순서로 구성된다. 2 장에서는 통계 정보를 이용해 트래픽을 분류하는 기존 연구들을 살펴보고, 3 장에서 Cross-order 의 정의와 해결방안에 대해 설명한다. 4 장에서 제안하는 방법에 대해 자세히 설명하고, 5 장에서 학내 망에서 수집한 트래픽을 이용한 성능 검증 실험을 기술한다. 마지막으로 6 장에서 결론 및 향후 연구에 대해 기술한다.

2. 관련 연구

응용 트래픽 플로우의 통계적인 특성을 이용한 트래픽 분류 방법은 최근 몇 년간 많은 관심을 받으며 연구되었다. 그 대부분은 머신러닝(ML, Machine Learning) 알고리즘을 이용한다. 이러한 방법은 응용 별 인터넷 트래픽의 특징이 될 수 있는 항목(port number, flow duration, inter-arrival time, packet size)들을 머신러닝 알고리즘에 적용하여 트래픽을 분류한다. 이 방법은 최근 증가하고 있는 암호화된 트래픽의 분석에 용이하며, 패킷의 페이로드 정보를 분석하지 않기 때문에 개인정보 침해의 문제가 없고 트래픽을 빠른 속도로 분류할 수 있다는 장점을 가진다. 또한, 머신러닝의 고급 알고리즘을 이용함으로써 트래픽을 응용 별로 분류함에 있어 다른 방법에 비해 보다 높은 정확도를 제공한다는 것이다.

표 1 은 기존의 트래픽 플로우 통계정보를 속성으로 사용하는 머신러닝 알고리즘 기반 트래픽 분류방법들을 비교한 것이다. Feature Extraction Range 는 속성을 추출하기 위해 조사하는 플로우 범위를 나타낸다. Full flow 는 전체 플로우 단위에서 속성을 추출하며, 플로우가 끝나야만 속성 추출이 가능하므로 실시간 트래픽 분류가 불가능하다. Feature Computation Cost 는 속성 추출 계산의 오버헤드를 나타내며 Low 는 계산이 필요 없는 경우, Medium 은 간단한 산술 계산이 필요한 10 개 이하의 속성을 사용하는 경우, High 는 복잡한 계산이 필요한 속성을 사용하는 경우이거나 간단한 산술계산이 필요한 속성을 10 개 이상 사용하는 경우를 나타낸다. ML Algorithm 은 관련 연구에서 사용하는 머신러닝 알고리즘의 종류로써 계산 복잡도를 유추할 수 있게 한다. 마지막의 Traffic Class 는 관련 연구의 트래픽 분류 단위로 Protocol 은 응용 계층 프로토콜을 의미하고 Application 은 개별 응용을 의미한다.

표 1 에서 보는 바와 같이 일부 연구들은[6,7,8,9] 트래픽을 분류하기 위한 속성들로 플로우 전체 단위의 통계적 특성을 사용하기 때문에 실시간 트래픽 분류에 사용할 수 없다. 이러한 문제를 해결하기 위해 플로우의 처음 N 개 패킷에서 속성을 추출하는 방법들이 연구

되었으나[4,5,10] 속성 계산 오버헤드와 사용하는 머신러닝 알고리즘의 높은 계산 복잡도로 인해 high-speed backbone 네트워크에서 실시간 분류에 적용하기엔 무리가 있다. 또한, 대부분의 연구들에서 트래픽을 분류하는 단위의 정의가 응용 프로토콜이기 때문에 분류 결과가 상세하지 않고, 이로 인해 개별 응용단위의 분류를 필요로 하는 네트워크 관리 및 운영 정책에 적용하기 어렵다. 그리고 모든 연구에서, 트래픽 캡쳐지점의 차이로 인해 일어나는 Full-duplex TCP 세션의 통계 정보의 변화에 대한 문제는 다루어지지 않았다.

표 1. 플로우의 통계적 특징을 이용한 머신러닝 기반 트래픽 분류 방법론

Related Work	Feature Extraction Range	Feature Computation Cost	ML Algorithm	Traffic Class
Bernaille et al. [4]	Partial flow	Low	K-Means, GMM, HMM	N Protocols, two applications
Tomaz Bujlow et al. [5]	Partial flow	High	C5.0	N Protocols, three applications
Y. Jin et al. [6]	Full flow	Medium	Modular architecture (Three linear ML Algorithms)	N Protocols
J. Tan et al. [7]	Full flow	Medium	SVM optimized by Particle Swarm Optimization	N Protocols, one application
R. Yuan et al. [8]	Full flow	Medium	SVM	N Protocols
Runyuan Sun et al. [9]	Full flow	High	Probabilistic Neural Networks	N Protocols
C. yin et al. [10]	Partial flow	Low	HMM	N Protocols, five applications

위에서 나열한 단점들을 극복하기 위해 본 연구에서는 플로우의 첫 N 개 패킷에서 산술계산이 필요 없는 속성들을 추출하며, 유사도 측정 시 1 차 산술계산만을 필요로 하는 알고리즘을 제안한다. 또한 제안하는 방법은 트래픽을 개별 응용 단위로 분류한다. 그리고 Cross-order 문제를 정의하고, 해결하는 알고리즘을 제안하고 적용하여 플로우 단위 99% 이상의 분류 정확도를 나타낸다.

3. Cross-order 의 정의와 그 해결 방안

Full-duplex TCP 세션은 두 종점 호스트에서 통신이 이루어질 때, 동시에 양방향으로 데이터를 송수신 할 수 있다. 즉, 한 호스트 A 에서 상대 호스트 B 로 전송 중인 패킷이 B 에 도달하기 전에 B 에서도 A 에 패킷을 전송할 수 있다. 이렇게 되면 두 패킷은 상대 호스트에 도달 하기 전에 교차를 이루게 된다. 이 경우, 각 호스트에서 패킷을 캡쳐할 시에 패킷의 순서가 뒤바뀌게 되고 이것을 Cross-order 라고 정의한다. 다음의 그림 1 은 Cross-order 의 예를 보여준다. C1 부터 C4 는 각각 다른 위치의 트래픽 캡쳐 지점을 나타내는데, 캡쳐하는 패킷의 순서가 모두 다른 순서를 나타낸다. Cross-order 를 해결하지 않을 경우, 같은 응용에서 발생하는 트래픽들도 매번 다른 순서로 수집될 뿐만 아니라, 한 지점에서 수집한 트래픽을 이용해 추출한 통계 정보 기반의 트래픽 분류를 위한 패턴들은 다른 지점에서는 사용할 수 없게 된다.

그러나 동시에 양방향으로 데이터를 송수신할 수 있는 Full-duplex TCP 세션의 특성 상 Cross-order 를 해결하는데 정답이 되는 수집 순서는 없다. 네트워크 트래픽 상황에 따라 얼마든지 변할 수 있기 때문이다. 따라서 Cross-order 를 해결하려면 기준이 되는 수집 순서를 정하고 그 기준에 따라 패킷 순서를 정렬해야 한다. 기준이 되는 수집 순서는 Server 에서 받아들이는 순서와 Client 에서 받아들이는 순서가 될 수 있다. 본 논문에서는 Client 에서 받아들이는 순서를 기준 순서로 하는 방법을 제안한다. 대부분의 트래픽이 Client 에서 Server 로 요청하고, Server 에서 이에 응답하는 형식으로 이어지기 때문이다. 즉 Client 의 요청에 따라 트래픽이 변화하기 때문이다.

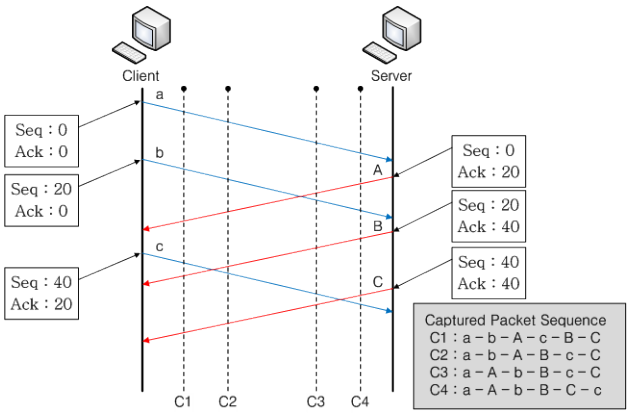


그림 1. Cross-order 의 예시

다음 그림 2 는 Cross-order 의 탐지 알고리즘이다. n 번째 패킷인 P(n)이 Forward 패킷 일 때 n+1 번째 패킷인 P(n+1)에선 Cross-order 문제가 발생하지 않는다. P(n) Backward 패킷이고 P(n+1) 이 Forward 패킷이며, P(n)의 Ack 넘버가 P(n+1)의 Seq 넘버 보다 같거나 작으며, P(n)의 Seq 넘버가 P(n+1)의 ACK 넘버보다 크거나 같을 때 Cross-order 가 발생한다.

```
Remove all non-payload packets from the packet sequence
1: procedure Detect abnormal Packet Sequence
2:     Input : packet sequence of a TCP flow
3:     if ( P(n) ==Bp && P(n+1) ==Fp ) {
4:         if ( P(n)(Ack) <= P(n+1)(Seq) && P(n)(Seq) >= P(n+1)(Ack) ) } then Cross-order
5:         else normal
6:     }
7: end procedure
```

Payload packet : a packet with payload data

Non-Payload packet : a packet without payload data

P(n): n-th payload packet in a TCP flow

Forward packet (Fp) : a packet from client to server

Backward packet (Bp) : a packet from server to client

그림 2. Cross-order 탐지 알고리즘의 슈도코드

```
Remove all non-payload packets from the packet sequence
1: procedure Resolve Cross-order
2:     Input : packet sequence of a TCP flow
3:     if ( P(n) and P(n+1) are Cross-order ) {
4:         change P(n) and P(n +1);
5:         n = n - 1;
6:         goto 3;
7:     }
8: end procedure
```

그림 3. Cross-order 해결 알고리즘의 의사 코드

그림 3 은 해결 알고리즘의 의사코드이다. 트래픽 수집 중 P(n)과 P(n+1)사이에 Cross-order 가 탐지 되면 두 패킷의 순서를 바꾸며 Cross-order 를 해결한다. 이 때, Cross-order 가 연속적으로

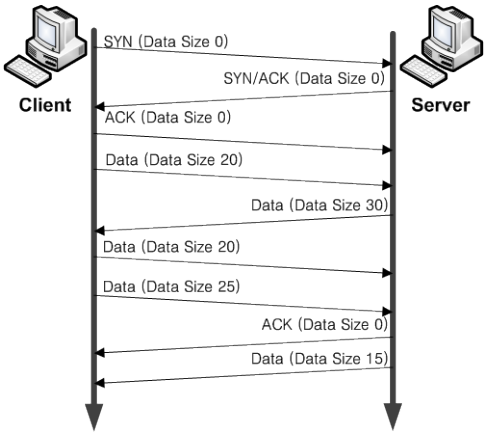
일어날 경우를 대비하여 뒤바뀐 P(n)과 P(n-1)과의 사이에 Cross-order 문제가 없는지 탐지한다. 연속적인 Cross-order 가 일어났을 경우, 모든 Cross-order 가 해결되어 패킷이 Client 기준의 순서로 정렬될 때까지 되짚어 간다. Cross-order 가 더 이상 탐지되지 않는다면 n+1 번째 패킷으로 돌아가 수행을 재개한다.

4. 패킷 분포 기반의 응용 트래픽 분류 방법

본 장에서는 제안하는 패킷 분포 기반의 트래픽 분류 방법에 대해 기술한다. 패킷 분포 기반 응용 트래픽 분류를 위해서는 우선 각 응용의 트래픽을 단위로 하여 고유한 시그니쳐를 수집하여야 한다. 응용 트래픽 분류와 관련된 기존의 많은 논문에서 응용 프로토콜을 트래픽 단위로 하여 트래픽을 분류하였지만, 본 논문에서는 개별 응용을 트래픽 단위로 하여 트래픽을 분류하기 때문이다. 개별 응용 단위의 트래픽 분류는 기존의 응용 프로토콜 단위의 분류에 비해 상세한 분석이 가능하여 더 다양한 분야에 활용될 수 있다는 장점이 있다. 제안하는 방법은 먼저 시그니쳐의 정의와 최적의 시그니쳐 생성 방법, 생성된 시그니쳐 기반의 응용 트래픽 분류 방법의 순서로 기술한다.

4.1 시그니쳐의 정의

일반적으로 플로우의 초기 몇 개의 패킷은 응용 프로그램에 의해 미리 정해진 규약대로 통신한다. 그러므로 각 응용 트래픽 플로우의 첫 N 개의 패킷에 해당하는 페이로드 크기와 방향, 즉 패킷 분포 정보는 응용 프로그램마다 달라 응용을 탐지할 수 있는 시그니쳐로 사용될 가능성이 크다. 본 논문에서는 플로우를 일반적으로 사용되는 5-tuple (source IP, destination IP, source port, destination port, protocol)을 기준으로 양방향으로 전송되는 패킷의 순서 집합이라 정의한다. 패킷 순서는 Client 에서 받아들이는 순서를 기준으로 삼는다.



Feature Vector = (+20, -30, +20, +25, -15)

그림 4. TCP 플로우의 특징 벡터 추출

트래픽 플로우의 패킷 분포 특징 벡터는 하나의 플로우를 내부 패킷들의 전송 순서와 방향, 그리고 payload 크기로 이루어진 순서집합으로 나타낸 것이다. Payload 크기는 정수로 표현하고 전송 방향은 +/-로 표현한다. TCP 의 경우에는 Client 에서 Server 로 전송하는 Forward 패킷의 방향을 +라 표현하고, Backward 패킷의 방향을 –라 표현한다. UDP 는 Client 와 Server 의 구분이 명확하지 않으므로 첫 번째 패킷과 동일한 전송방향을 +라 표현하고 반대방향을 –라 표현한다. 또한, 특징 벡터는 Payload 가 존재하는 패킷만으로 이루어진다. 즉 TCP 세션의 SYN, ACK 등의 control packet 들은 제외한다. 이는 규칙적이지 않은 control packet 의 발생이 패킷 분포 정보에 영향을 끼치는 것을 방지하기 위함이다. 그림 4 와 같은 통신을 하는 양방향 플로우를 특징

벡터화하면 해당 벡터는 control packet(SYN, SYN/ACK, ACK)들을 제외한 Data 패킷들의 순서 집합인 (+20, -30, +20, +25, -15)의 값을 가진다.

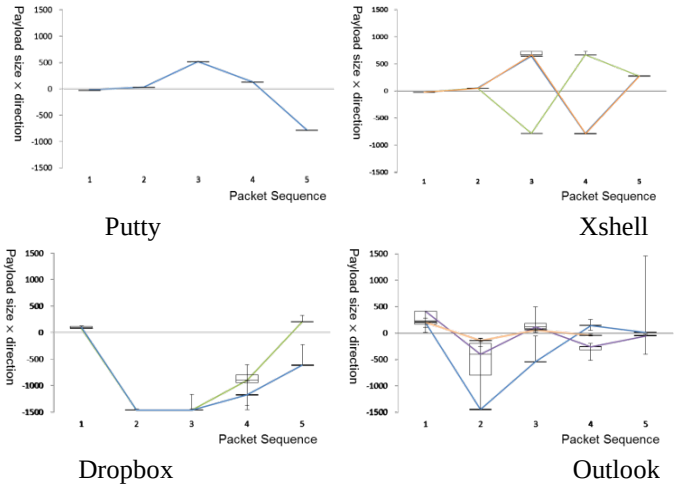


그림 5. 4 응용 트래픽 플로우 특징 벡터

그림 5 는 각 응용의 플로우 특징 벡터들을 boxplot 형태의 그래프로 나타낸 것이다. 동일한 전송 순서와 방향을 가진 벡터들을 하나의 그룹으로 표현하고 있으며, 이는 꺾은선 그래프로 구분된다. 세로축은 패킷의 페이로드 크기와 방향의 곱을 나타내며 가로축은 패킷의 순서이다. 하나의 패킷의 페이로드 크기는 -1460 에서 +1460 까지의 값을 가질 수 있다. 그림 2 에서 나타내는 응용은 4 개로 dropbox, Microsoft outlook, putty, xhell 이 그것이다. 모든 응용의 플로우들이 PuTTY 처럼 초반 N 개 패킷에서 전부 동일한 페이로드 사이즈와 전송 순서 및 방향으로 통신하지 않는다. Xshell, Dropbox 처럼 2, 3 개의 특정 패턴으로 나뉘어지기도 하고, Outlook 처럼 여러 개의 특정 패턴들로 나뉘어지기도 한다. 하지만 다른 응용 들과 비교하여 봤을 때 응용 플로우 특징 벡터들은 각 응용 별로 고유한 특징을 나타냄을 알 수 있다.

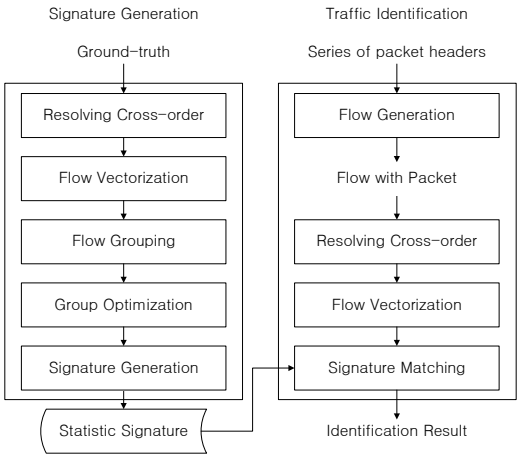


그림 6. 패킷 분포 기반 응용 트래픽 분류 방법의 순서도

4.2 패킷 분포 기반 트래픽 분류 방법

본 논문에서 제안하는 방법론의 시그니쳐는 응용 트래픽 플로우의 패킷 분포 정보로 구성된, 다른 응용 프로그램과 구별되는 응용 프로그램의 고유한 특징이다. 시그니쳐는 응용의 트래픽 플로우를 특징 벡터화 한 뒤 벡터간의 유사도를 기반으로 벡터들을 그룹핑하고, 각 그룹의 벡터들을 대표하는 대표 특징 벡터와 해당 그룹의 벡터들을 모두 포함할 수 있는 distance threshold 벡터의 조합으로 추출한다. 한 응용에서 발생하는 트래픽 플로우의 모든 특징 벡터를 해당 응용의 시그니쳐로 사용하게 되면 시그니쳐의 개수가 증가하여 관리 및 응용 트래픽 탐지에 과부하가 걸린다. 따라서 해당 응용의 특징 벡터들을 그룹핑하여 각 그룹을 대표할 수 있는 대표

특징 벡터와 거리임계 벡터의 조합으로 소수의 시그니쳐를 구성하는 것이 최적의 시그니쳐라 할 수 있다. 그림 6 은 본 논문에서 제안하는 패킷 분포 정보 기반 트래픽 분석 방법의 순서도로 크게 두 부분으로 나뉜다. 시그니쳐 추출부와 트래픽 분류부이다. 시그니쳐 추출부에서는 먼저 수집된 응용 플로우에서 Cross-order 를 해결한 뒤 플로우를 특징 벡터화 한다. 그 다음 벡터들의 유사도를 기반으로 그룹핑하고, 마지막으로 그룹들을 최적화 작업을 수행한 뒤 시그니쳐를 추출한다. 트래픽 분류부는 입력되는 패킷을 이용해 플로우를 생성하고 Cross-order 를 해결한 뒤 벡터화한다. 벡터화 된 플로우들을 시그니쳐와 유사도 매칭을 통하여 응용을 판단한다.

4.2.1 시그니쳐 추출

시그니쳐 추출은 먼저 수집된 플로우에서 Cross-order 를 제거한 뒤 시작한다. 플로우 f_k 의 특징 벡터 v_k 는 수식 (1)과 같이 표현한다. 그리고 v_k 의 i 번째 인자인 $v_{k,i}$ 는 수식 (2)와 같이 표현한다. $d_{k,i}$ 와 $s_{k,i}$ 는 각각 f_k 의 i 번째 패킷의 전송 방향과 payload 크기를 나타낸다. 따라서, $d_{k,i}$ 는 +혹은 –의 값을 가지며 $s_{k,i}$ 는 -1460 부터 +1460 까지의 정수 값을 가진다.

$$v_k = (v_{k,1}, v_{k,2}, ..., v_{k,N}) \tag{1}$$

$$v_{k,i} = d_{k,i} \times s_{k,i} \tag{2}$$

플로우 그룹핑은 각 벡터들의 유사도를 이용해 이루어진다. 두 벡터 사이의 유사도는 각 패킷별 거리를 사용한다. 즉, 두 벡터 사이의 유사도는 거리 벡터(벡터 d)로 표현된다. 두 벡터 v_k 와 v_j 사이의 거리 벡터 $d(v_k, v_j)$ 는 수식 (3)과 같이 표현한다. 플로우 그룹핑은 거리 벡터 d 의 값이 threshold 이하일 때 이루어지며, 본 논문에서는 초기 threshold 값을 10 으로 정하였다. 즉, 거리 벡터 d 의 각 요소의 값이 10 이하일 때 하나의 그룹으로 그룹핑된다. 각 벡터 그룹에서 하나의 시그니쳐를 추출한다.

$$d(v_k, v_j) = \{|v_{k,1} - v_{j,1}|, |v_{k,2} - v_{j,2}|, ..., |v_{k,N} - v_{j,N}|\} \tag{3}$$

시그니쳐 s 는 수식 (4)와 같이 대표 벡터와 임계 벡터의 조합으로 표현된다. c 는 벡터 그룹의 대표 특징 벡터이고 그룹 내 모든 벡터들의 각 요소의 평균값이 c 벡터의 요소가 된다. t 는 벡터 그룹의 모든 벡터들을 포함할 수 있는 distance threshold 벡터이다. 따라서 벡터 그룹의 모든 벡터들은 수식 (5)를 만족한다. $V(s)$ 는 시그니쳐 s 가 추출되는 벡터그룹이다.

$$s = (c, t) \tag{4}$$

$$d(v, c) \le t \ \ for \ \forall v \in V(s) \tag{5}$$

한 응용의 시그니쳐는 여러 개일 수 있다. 따라서 최적의 시그니쳐를 추출하는 방법이 필요하다. 최적의 시그니쳐 Set S 를 구하기 위한 조건은 수식(6)과 같다. 최소 개수의 대표 특징 벡터, 최소값의 distance threshold 벡터값의 총합, 그리고 마지막으로 시그니쳐가 포함할 수 있는 최대 개수의 응용 플로우 특징 벡터 개수이다. 여기서 $|S|$ 는 set S 의 시그니쳐 개수를 의미하며, $t(s)$ 는 시그니쳐 s 에 대응하는 distance threshold 벡터이며, $|V(s)|$ 는 벡터 그룹 $V(s)$ 의 플로우 특징 벡터 개수를 의미한다.

$$\text{S such that} \begin{Bmatrix} \text{minimize } |S| \\ \text{minimize } \sum t(s) \\ \text{maximize } \sum |V(s)| \end{Bmatrix} \tag{6}$$

마지막으로 최적의 시그니쳐 set S 를 구하기 위해 그룹 최적화 단계를 거친다. 벡터 그룹의 c 벡터는 그룹 내 모든 벡터들의 평균이므로 플로우 특징 벡터 하나가 그룹핑 될 때 마다 재계산된다. 따라서 그룹핑 단계가 완료된 후엔 한 그룹에 속해 있었으나 최종 그룹에는 속하지 않는 플로우들이 발생하는데, 오분류를 없애기 위해 해당 플로우들을 제외한다. 또한, 모든 그룹핑 작업이 끝났으므로 t 의 값을 최소값으로 변경한다. 최소값은, 그룹 내 모든 플로우를 포함할 수 있는 각 요소 별 최소 값이다. 그룹 최적화단계가 끝나면 시그니쳐 set S 를 추출한다. 각 시그니쳐들은 표 2 와 같은 속성들을 가진다.

ID 는 대상 시그니쳐의 응용 프로세스 이름이다. Proto 는 대상 flow group 의 L4 protocol 로 TCP 또는 UDP 의 값을 가진다. 즉 시그니쳐는 같은 응용이라 하더라도 L4 protocol 별로 각각 추출된다. Dim 은 group 에 속한 flow vector 들의 dimension 이다. 즉 flow 그룹에 속한 플로우들은 동일한 dimension 을 가진다. Dimension 은 곧 페이로드 패킷의 개수를 의미하므로 1 에서 N 까지의 값을 가질 수 있다. 하지만 본 연구에서는 최소값을 3 으로 정하였다. 이는 1, 2 차원의 벡터로는 분별력 있는 시그니쳐를 추출할 수 없기 때문이다. 또한, N 의 값은 최대 5 로 정하였다. Centroid Vector(C-vector)는 시그니쳐의 대표 벡터로 c 벡터이다. Distance threshold vector (T-vector) 는 t 벡터로 그룹의 포함 범위를 나타낸다.

표 2. 시그니쳐의 속성

Attribute	ID	Proto	Dim	C-vector	T-vector
Description	Application Name	L4 protocol: TCP/UDP	Dimension of vector	centroid vector	Distance threshold vector
Example	Skype	UDP	5	(+20,+30,-50,+20,-30)	(10, 10, 10, 10, 10)

4.2.2 트래픽 분류 방법

트래픽 분류 단계에서는 생성된 응용의 시그니쳐를 바탕으로 대상 네트워크의 트래픽을 수집하고 실시간으로 분류한다. 먼저 입력되는 패킷들로 플로우를 생성한다. Cross-order 를 해결한 플로우는 이후 특징 벡터화 되며 모든 시그니쳐와 포함 여부를 확인한다. 분석 대상 플로우가 어떤 응용에서 추출된 시그니쳐에 포함되었을 경우 해당 응용에서 발생한 플로우라 판단한다. 트래픽 분류는 즉, 타겟 플로우 벡터 v 에 대해 L4 Proto 와 Dim 이 같고, 수식 (7)의 포함 조건을 만족시키는 시그니쳐 s_i 를 찾는 작업이다.

$$s_i \text{ such that } d(c_i, v_i) \le t \text{ for } s_i = (c_i, t_i) \tag{7}$$

5. 실험 및 결과 분석

본 장에서는 제안하는 패킷 통계 정보 기반 응용 트래픽 분류 방법의 성능을 검증하기 위해 학내 망의 트래픽을 대상으로 분류한 결과를 설명한다. 본 연구에서 사용된 트래픽은 고려대학교 학내 망에서 인터넷으로 오가는 모든 트래픽을 대상으로 한다. 본 연구에서 분류에 사용되는 특징은 패킷의 방향을 포함하기 때문에 두 호스트 간 통신에서 서로 향하는 모든 트래픽을 수집해야 한다. 본 연구의 검증 네트워크인 학내 망은 최상위 라우터를 통해 인터넷과 연결되므로 모든 인터넷 트래픽을 해당 라우터에서 미러링하여 수집하였다. 시그니쳐를 생성하기 위한 training data set 은 프로세스를 기준으로 플로우와 플로우에 속한 페이로드 패킷(페이로드가 존재하는 패킷)으로 이루어졌다. 응용 계층의 트래픽을 분류하는 것이 목적이므로 전송 계층에만 나타나는 TCP 컨트롤 패킷(SYN, ACK, Keep-Alive 등)은 제외하였다.

Training data set 은 정확하게 프로세스 별로 수집하는 방법이 필요하다. 본 연구에서는 학내 망 내에 있는 일부 말단 호스트들에 TMA(Traffic Measurement Agent)를 설치하고 이를 통해 ground truth 를 생성한다[11]. Agent 를 통한 ground truth 생성 방법은 특정 분류 방법을 통해 분류한 결과를 사용한 것[12]보다 높은 신뢰성을 보장해 준다. 표 4 는 본 논문에서 사용한 ground truth 의 Traffic Class (응용 이름)와 그에 대한 간략한 설명, 그리고 트래픽 양을 나타낸다. 수집된 트래픽의

양은 동일한 기간 동안 수집한 것으로써 개별 응용의 사용 빈도 및 응용의 기능 별 트래픽 발생 양에 따라 차이가 있다.

본 논문의 실험 결과 평가 요소로는 completeness, accuracy 의 2 가지를 사용한다. Completeness 는 시그니쳐를 통해 전체 트래픽 중 얼마나 많은 양의 트래픽을 분석했는지에 대한 척도로써, 분석한 트래픽의 양을 나타낸다. accuracy 는 분류 결과와 ground truth 를 비교하여 분류 정확도를 나타낸다. Accuracy 는 overall accuracy 와 응용 별 precision, recall 값들로 표현된다.

표 4 는 전체 성능 검증 실험에 대한 completeness 와 overall accuracy 를 플로우, 패킷, 바이트 단위로 보여준다. completeness 를 먼저 살펴보면, flow 단위의 completeness 는 44.16%이나 Packet, Byte 단위의 completeness 는 90%에 다다른다. 이는 ground truth 중 가장 많은 양의 트래픽을 차지하는 Naverlive 의 분석 결과가 끼치는 영향이 크다. flow 단위로 전체 트래픽의 8.5%를 차지하는 Naverlive 트래픽에서, Packet, Byte 단위로는 전체 트래픽의 90%를 넘게 차지한다. 이러한 Naverlive 트래픽 분석 결과가 Flow 단위 Recall 값은 80.78%, Packet 단위 Recall 값은 91.10%를 나타냄으로써 전체 트래픽의 Flow, Packet, Byte 단위의 Completeness 값이 차이가 나는 결과를 불러온다.

표 3. Ground-truth

Traffic Class	Description of applications	flow (10^3)	pkt (10^3)	byte (10^6)
Skype	P2P communications	2.9	43.9	17.6
GomTV	internet TV service	15.2	2,637.0	2,515.6
Naverlive	video streaming	2.6	50,807.5	41,404.3
Nateon	instant messaging	0.9	337.1	62.5
Outlook	MS mail service	12.7	1,150.4	692.3
PuTTY	Telnet/SSH client	0.7	80.4	11.5
Xshell	Telnet/SSH client	1.2	177.1	22.8
Teamviewer	remote control	1.8	722.6	291.7
Dropbox	cloud file sharing	11.1	294.8	158.5

표 4. Completeness 와 Overall Accuracy

	completeness		Overall Accuracy	
	적용 전	적용 후	적용 전	적용 후
Flow	44.16%	44.38%	99.97%	99.99%
Packet	90.04%	90.30%	99.99%	99.99%
Byte	90.45%	90.68%	99.99%	99.99%

표 5 는 전체 성능 검증 실험의 결과를 각 응용 별 precision 값과 recall 값으로 나타낸다. 표 4 에서와 같은 completeness 문제는 표 5 에서도 볼 수 있다. 대부분의 응용에서 플로우 단위의 recall 보다 packet 단위의 recall 이 높은 점이다. 이는 시그니쳐 추출 시에 Dimension 의 최저값을 3 으로 고정하였기에 패킷 한, 두개를 포함한 플로우를 분석하지 못하는 영향이 크다. 또한 비정상적으로 종료되는 세션들의 영향도 있다. 특히 그 격차가 심한 GomTV 의 경우 Streaming 과 관련이 없는 적은 패킷을 포함한 플로우들의 분석률은 낮으나, 중요한 Streaming 관련 플로우들에 대한 분석률이 높다는 것을 알 수 있다. Flow, Packet 단위의 Recall 이 낮은 이유가 하나 더 있는데 이는 트래픽 분류 정책이다. 현재의 분류 정책으로는 분류 정확도를 위해 두 개 이상의 다른 응용으로 분류된 트래픽은 분류하지 않기 때문이다. 다른 적용 가능한 정책으로는 유사도가 더 높은 응용으로 분류하거나, 분류된 모든 응용을 리포트하여 네트워크 관리자에게 결정을 일임하는 방법이 있다.

표 5. 개별 응용의 Precision, Recall

Traffic Class	Flow				Packet			
	Precision		Recall		Precision		Recall	
	적용 전	적용 후	적용 전	적용 후	적용 전	적용 후	적용 전	적용 후
Skype	99.99%	99.99%	51.49%	51.49%	99.99%	99.99%	69.84%	69.84
GomTV	99.97%	99.99%	5.48%	5.48%	99.99%	99.99%	83.57%	83.57
Naverlive	99.99%	99.99%	80.78%	81.56%	99.99%	99.99%	90.84%	91.10
Nateon	99.97%	99.97%	85.31%	85.31%	99.99%	99.99%	98.39%	98.39
Outlook	99.98%	99.99%	70.85%	71.47%	99.96%	99.98%	64.55%	65.03
PuTTY	100%	100%	98.13%	98.13%	100%	100%	99.50%	99.50
Xshell	100%	100%	96.28%	99.26%	100%	100%	97.29%	99.15
Teamviewer	99.97%	99.99%	50.41%	50.88%	99.99%	99.99%	83.68%	83.74
Dropbox	99.98%	99.98%	97.01%	97.01%	99.98%	99.98%	97.89%	97.89

본 방법론의 트래픽 분류 정확도는 표 4, 5 에서 공통적으로 알 수 있듯이 매우 높은 성능을 나타낸다. 동일한 SSH 프로토콜을 사용하는 Xshell 과 PuTTY 두 응용의 트래픽도 정확도 100%로 분류한다. 또한, Cross-order 를 해결함으로써 방법론의 변화가 아닌, 트래픽 수집 시 필요한 패킷 순서의 일정한 정렬만으로도 정확도 및 분석률이 향상됨을 볼 수 있다. 하지만 오직 Xshell 과 PuTTY 만이 정확도 100%를 보이는 분류 결과를 나타내었다. 이는 다른 TCP 세션의 이상동작인 패킷 Retransmission, Out-of-order 등과 관계가 있다. 추후 이를 해결하기 위한 연구를 진행할 계획이다.

6. 결론 및 향후연구

본 논문에서는 플로우에 속한 초반 N 개 패킷들의 전송 순서와 방향, 그리고 페이로드 크기를 속성으로 하여 시그니쳐를 생성하고 이를 이용해 트래픽을 분류하는 방법을 제안하였다. 본 방법론은 속성 추출 시 계산이 필요 없으며, 트래픽 분류를 위한 유사도 측정 시 단순 산술계산만을 필요로 하므로 high-speed backbone 네트워크에서의 실시간 트래픽 분석에도 문제 없이 적용할 수 있다. Payload 데이터를 조사하지 않기 때문에 암호화 된 트래픽의 분류도 가능하다. (6)의 조건을 만족하는 시그니쳐를 추출함으로써 시그니쳐의 관리가 수월하며 높은 정확도를 가진다. 또한, 개별 응용을 트래픽 분류 단위로 이용하여 Protocol 을 단위로 사용하는 기존 연구들보다 더욱 세밀한 분석 결과를 도출 할 수 있고 이로 인해 다양한 네트워크 운영 및 관리 정책에 적용하기 알맞은 방법이다. 마지막으로, 트래픽 수집 지점에서 생길 수 있는 문제인 Cross-order 를 해결함으로써 동일한 방법로을 사용하여 트래픽을 분류하더라도 분석률과 정확도가 향상되었고, 추출된 시그니쳐는 다른 네트워크에서 트래픽 분류에 사용되더라도 일관성을 유지한다. 하지만 100%를 만족하지 않는 분류 정확도 문제, TCP 세션의 이상동작 문제, 미분류 트래픽의 처리 문제와 HTTP 트래픽의 분류 방법 등 앞으로 해결해야 할 과제들이 남아있다. 향후 통계 정보 기반 트래픽 분류 방법에 영향을 끼치는 여러 요소들과, 더욱 정확한 분류 결과를 나타내는 방법, 그리고 HTTP 트래픽의 정확한 분류 방법에 관하여 연구할 계획이다.

7. 참고 문헌

[1] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, et al., "A Survey on Internet Traffic Identification," IEEE Communications Surveys and Tutorials, vol. 11, pp. 37-52, 2009.

[2] T. T. T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning," IEEE Communications Surveys and Tutorials, vol. 10, pp. 56-76, 2008.

[3] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and Future Directions in Traffic Classification," IEEE Network, vol. 26, pp. 35-40, January-February 2012.
[4] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in Proc. of ACM CoNEXT conference, 2006.
[5] T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 Machine Learning Algorithm," in Proc. of International Conference on Computing, Networking and Communications (ICNC), pp. 237-241, 2012.
[6] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z. L. Zhang, "A Modular Machine Learning System for Flow-Level Traffic Classification in Large Networks," ACM Transactions on Knowledge Discovery from Data, vol. 6, pp. 1-34, 2012.
[7] J. Tan, X. Chen, and M. Du, "An Internet Traffic Identification Approach Based on GA and PSO-SVM," Journal of Computers, vol. 7, pp. 19-29, 2012.
[8] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate internet traffic classification," Information Systems Frontiers, vol. 12, pp. 149-156, April 2010.
[9] S. Runyuan, Y. Bo, P. Lizhi, C. Yuehui, Z. Lei, and J. Shan, "Traffic classification using probabilistic neural networks," in Proc. of International Conference on Natural Computation (ICNC), pp. 1914-1919, 2010.
[10] C. Yin, S. Li, and Q. Li, "Network traffic classification via HMM under the guidance of syntactic structure," Computer Networks, vol. 56, pp. 1814-1825, April 2012.
[11] B. C. Park, Y. J. Won, M. S. Kim, and J. W. Hong, "Towards automated application signature generation for traffic identification," in Proc. of IEEE Network Operations and Management Symposium (NOMS), pp. 160-167, 2008.
[12] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "GT: picking up the truth from the ground for internet traffic," ACM SIGCOMM Computer Communication Review, vol. 39, pp. 12-18, 2009.

안 현 민

2012년 고려대학교 컴퓨터정보학과 (학사)

2012년~현 재 고려대학교 컴퓨터정보학과 (석사)

관심분야:네트워크 관리 및 보안, 트래픽 모니터링 및 분석

함 재 현

1999년 동국대학교 컴퓨터공학과 (학사)

2001년 포항공과대학교 컴퓨터공학과 (석사)

2001년~현재 국방과학연구소 선임연구원

2012년~현재 고려대학교 컴퓨터정보학과 (박사)

관심분야:전술통신망 관리, 네트워크 관리, 트래픽 모니터링 및 분석

김 명 섭

1998년 포항공과대학교 전자계산학과 (학사)

1998년~2000 년 포항공과대학교 컴퓨터공학과 (석사)

2000년~2004 년 포항공과대학교 컴퓨터공학과 (박사)

2004년~2006 년 Post-Doc., Dept. of ECE, Univ. of Toronto, Canada.

2006년~ 현재 고려대학교 컴퓨터정보학과 조교수

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석, 멀티미디어 네트워크