# 패킷 역전 및 중복 문제를 해결한 통계기반 트래픽 분석 시스템

# Statistics-based Traffic Analysis System without Out-of-order and Retransmission Problem

이수강, 안현민, 심규석, 김명섭

고려대학교 컴퓨터정보학과

{sukanglee, queen26, kusuk007, tmskim}@korea.ac.kr

## 요 약

인터넷이 발전함에 따라 네트워크의 규모는 나날이 증가하고 있으며, 네트워크 내에서 발생하는 응용 트래픽 분석의 중요성 또한 강조되고 있다. 이러한 상황 속에서 네트워크 관리자는 효율적인 네트워크의 관리를 위해 트래픽을 발생시킨 응용을 탐지할 수 있어야 한다. 응용을 탐지하는 방법들 중 하나인 통계기반 트래픽 분석 방법은 패킷의 통계정보를 이용하여 응용을 탐지한다. 하지만 수집지점에서 수집되는 패킷은 통계정보의 일관성을 저해시키는 패킷 역전 문제 및 재전송에 의한 중복 문제로 인해 정확한 응용 탐지에는 한계가 있었다. 본 논문에서는 이러한 문제점들을 해결하는 방법론을 제시하고 실제 트래픽 분석 시스템에 적용시킴으로써 응용별 바이트 기준 최대 4%의 탐지 및 분석률 향상을 보였다. 이는 제안한 방법론이 실제 트래픽 망에 부담을 줄 수 있는 heavy 플로우의 분석에 기여함을 확인하였다.

**Keywords**: Out-of-order, Retransmission, Application classification, Network Management

## 1. 서론

현재 인터넷의 급격한 발전으로 인해 네트워크의 규모는 나날이 증가하고 있으며, 네트워크 내에서 발생하는 응용 트래픽 분석의 중요성 또한 강조되고 있다. 이러한 상황 속에서 네트워크 관리자는 효율적인 네트워크의 관리를 위해 트래픽을 발생시킨 응용을 탐지할 수 있어야 한다. 트래픽의 통계정보를 사용하여 응용을 분류하는 방법[1,2]은 트래픽을 발생시킨 응용을 탐지하고 분류하는 방법이다. 통계정보 트래픽 분류방법은 패킷의 크기와 전송방향, 전송순서, 그리고 수집된 시간 등의 속성값들을 사용하여 응용을 분류한다. 수집지점에서 수집된 패킷의 통계정보를 아무런 처리과정 없이 사용할 경우 패킷의 역전(Out-of-order), 재전송에 의한 패킷 중복(Retransmission)와 같은 문제점들로 인해 트래픽의 feature가 달라져서 100% 정확한 응용 분류가 어렵다.

수집지점에서 발생하는 패킷 역전 문제는 패킷들이 단일 경로가 아닌 여러 경로를 통해 전송될 때 발생한다. 이 때 패킷이 전송되는 경로의 상태에 따라 전송되는 패킷들의 순서는 송신측이 보낸 순서가 아닌 다른 순서로 수신측에 전달된다. 원래 순서가 아닌 다른 순서의 패킷들은 데이터가 전송되는 경로의 중간에 위치한 수집지점에서 트래픽이 수집된다. 이렇게 수집되어 저장된 패킷의 순서는 송신측이 보낸 패킷의 순서와 맞지 않게 되는데 이러한 문제를 수집지점에서 발생하는 패킷 역전 문제라 한다.

이 논문은 2012 년 정부(교육과학기술부)의 재원으로 한국연구재단(2012R1A1A2007483) 및 2013 년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보 컴퓨팅기술개발사업(2010-0020728) 및 2013 년 BK21 플러스 사업의 지원을 받아 수행된 연구임.

수신측은 송신측으로부터 받은 패킷에서 오류가 발견되었을 경우 오류가 발견된 패킷과 동일한 패킷을 송신측에게 재전송 요청을 한다. 또는 송신측은 수신측에게 보낸 패킷의 응답패킷을 일정 시간내에 받지 못할 경우 해당 패킷을 재전송 하게 된다. 이렇게 발생한 재전송 패킷은 수집지점에서 중복되어 수집되는데 이러한 문제를 수집지점에서 발생하는 패킷 중복 문제라 한다.

본 논문에서는 트래픽 수집지점에서 발생하는 패킷 역전, 패킷 중복 문제를 해결하는 알고리즘을 제안하고자 한다. 이러한 문제를 수집지점에서 처리하게 되면 수집되는 트래픽의 통계정보는 각 응용의 트래픽 통계정보와 항상 동일한 트래픽을 수집할 수 있다. 이렇게 수집한 통계정보의 특성을 바탕으로 트래픽을 발생시킨 응용별 시그니쳐를 만들어 사용하게 되면 분석이 요구되는 트래픽 데이터에서 해당 응용을 정확히 탐지할 수 있다.

본 논문은 다음과 같은 순서로 구성된다. 2장에서는 통계정보를 이용해 트래픽을 분류하는 기존 연구들을 살펴보고 3장에서는 수집지점에서 발생하는 패킷 역전 문제와 패킷 중복 문제를 정의한 후 종점 호스트에서 발생하는 패킷 역전, 패킷 중복 문제와의 차이점을 기술한다. 4장에서는 수집지점에서 발생하는 패킷 역전, 패킷 중복 문제를 해결하는 방법을 제시하고 5장에서는 제안한 방법의 성능을 평가하기 위한 실험 결과를 기술한다. 마지막으로 6장에서는 결론과 향후 연구 계획을 언급한다.

## 2. 관련 연구

응용 트래픽 플로우 정보의 통계적인 특성을 이용한 트래픽 분류 방법은 최근 몇 년간 많은 관심을 받으며 연구가 진행되어 왔다. 그 대부분은 머신러닝(Machine Learning) 알고리즘을 이용한다. 이러한 방법들은 응용별 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 빠른 속도로 분류할 수 있다는 장점을 가진다. 또한 머신러닝의 알고리즘을 이용함으로써 트래픽을 응용별로 분류함에 있어 다른 방법에 비해 보다 높은 분석 정확도를 제공한다.

기존의 일부 연구들은[3,4,5,6] 트래픽을 분류하기 위한 속성들로 플로우 전체 단위 패킷의 통계적 특성을 사용하기 때문에 실시간 트래픽 분류에 사용할 수 없다. 이러한 문제를 해결하기 위해 플로우의 처음 N개 패킷에서 속성을 추출하는 방법들이 연구되었으나[7,8,9] 속성 계산 오버헤드와 사용하는 머신러닝 알고리즘의 높은 계산 복잡도로 인해 초고속 대용량 네트워크에서 실시간 분류에 적용하기엔 무리가 있다. 또한 대부분의 연구들에서 트래픽을 분류하는 단위의 정의가 응용의 프로토콜이기 때문에 분류 결과가 상세하지 않고, 이로 인해 개별 응용단위의 분류를 필요로 하는 네트워크 관리 및 운영 정책에 적용하기 어렵다. 기존의 모든 관련연구에서는 트래픽의 통계적 특징을 사용하지만 패킷 역전문제, 재전송에 의한 패킷 중복문제와 같은 통계적 특징 변화에 대한 문제는 다루어 지지 않았다. 따라서 본 연구에서는 트래픽 수집지점에서 발생하는 패킷 역전 문제와 재전송에 의한 중복 패킷 문제를 정의하고 해결하는 알고리즘을 제안하고자 한다.

## 3. 문제 정의

본 장에서는 수집지점에서 일어나는 패킷 역전 문제와 재전송에 의한 패킷 중복 문제를 정의한다. 또한 종점 호스트에서 발생하는 패킷 역전, 패킷 중복 문제와 트래픽 수집지점에서 발생하는 패킷 역전, 패킷 중복 문제와의 차이점을 기술한다.

TCP는 두 종점 호스트 사이에 신뢰성 있는 세션을 제공함으로써 상위계층(응용프로그램) 사이에 투명한 데이터 전달을 제공한다. 종점 간 신뢰성 있는 데이터 송수신 보장을 위해 TCP는 sequence와 Acknowledge, checksum등의 필드와 재전송, 흐름제어 등의 방법을 사용한다. 데이터가 목적지의 TCP에게 잘 전달되었는지 확인하기 위해 Ack(Acknowledge)를 사용한다. 데이터를 수신한 상대방은 데이터가 오류 없이 잘 도착했는지 확인 후 문제가 없으면 송신 측으로 Ack를 보낸다. 또한 오류를 확인하기 위해 checksum을 사용하고 데이터의 전달 순서를 확인하기 위해 sequence를 사용한다. 수신측에서 오류를 발견하면 오류가 발견된 패킷을 버리고 Ack를 보내지

않음으로써 송신측에 오류가 발생했음을 알리거나 최근 정상적으로 수신한 데이터에 대한 Ack를 반복하여 보냄으로써 송신측이 Ack이후의 데이터에서 오류가 발생한 것 또는 패킷이 유실된 것을 알 수 있도록 하고 해당 패킷을 재전송 하도록 한다.

### 3.1 패킷 역전 문제

두 종점 호스트 사이에 전송되는 패킷들은 단일 경로가 아닌 여러 경로를 통하여 전송될 수 있다. 패킷이 여러 경로로 전송 될 경우, 각각의 경로의 상황에 따라 송신 측에서 보낸 패킷을 수신 측에서 받을 때 송신 측에서 보낸 순서와는 다른 순서로 받게 될 수 있다. 이러한 문제를 종점 호스트에서 발생하는 패킷 역전 문제라 하며 수신측의 TCP는 이러한 문제 해결을 위해 패킷의 Seqenuce를 확인하고 패킷 의 순서를 재 정렬하게 된다.

그림 1에서 Host A가 Host B에게 여러 개의 패킷들을 전송한다고 가정할 때 패킷들은 각각 다른 경로로 전송될 수 있으며, Host B가 전송받는 패킷들의 순서는 Host A가 보낸 순서와 다를 수 있다. 이때 Host B의 TCP 전송계층에서는 원래의 순서대로 재 정렬 작업을 수행하여 상위 계층으로 전달한다. 그러나 트래픽 수집지점에서는 이러한 재정렬 작업을 수행하지 않으므로 TCS에 저장되는 패킷의 순서는 Host A 에서 보낸 패킷의 순서와 맞지 않게 된다. 이러한 문제를 수집지점에서 발생하는 패킷 역전(Out-of-order)문제라 한다.

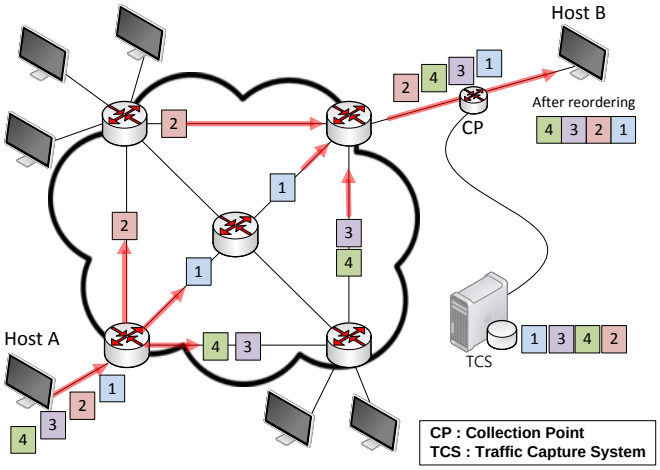


**그림 1. 수집지점에서 발생하는 패킷 역전 문제**

### 3.2 패킷 중복 문제

패킷 재전송은 송신측에서 보낸 패킷이 수신측으로 전송중에 어떠한 이유로 유실되어 수신측이 해당 패킷을 받지 못하였거나 받은 패킷에 오류가 발견되었을 경우 수신측 요청에 의해 발생하는 것으로, 송신측은 오류로 인해 수신측까지 전송되지 못한 데이터와 해당 데이터의 sequence값을 가진 패킷을 재전송 하게 된다.

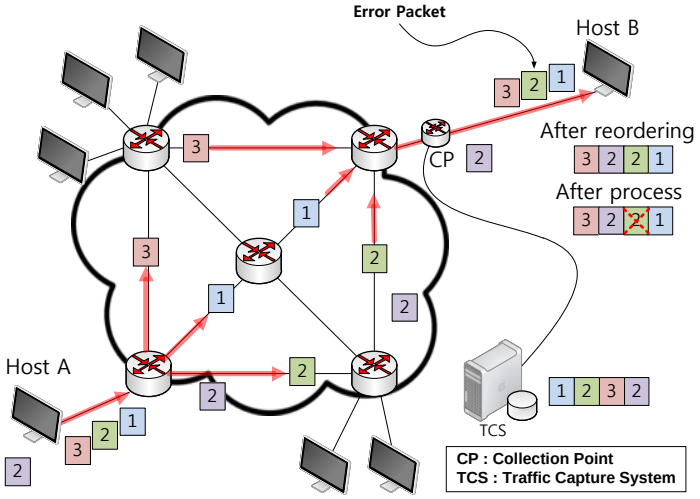


**그림 2. 수집지점에서 발생하는 재전송에 의한 패킷 중복 문제**

그림 2는 Host A가 Host B에게 패킷을 보낼 때 재전송 패킷이 발생하는 상황을 나타낸 그림이다. Host B는 Host A가 보낸 패킷 총 3개의 패킷을 받는다. Host B는 Host A가 보낸 패킷중 2번째 패킷에서 오류를 발견하고 Host A에게 2번 패킷의 재전송을 요청하게 된다. Host A는 재전송 요청에 의해 2번 패킷을 다시 보내게 되고 Host B는 재전송된 패킷을 받고 오류가 발견된 패킷은 삭제한다. 하지만 중간에 위치한 수집지점에는 오류가 발생한 원래의 2번패킷과 재전송된 2번패킷을 중복하여 저장한다. 중복 저장된 패킷은 서로 동일한 sequence를 갖는 패킷이다. 이러한 문제를 수집지점에서 발생하는 재전송에 의한 패킷 중복(Retransmission)문제라 한다.

### 3.3 재패킷화 문제

재전송에 의해 중복되어 저장되는 패킷은 서로 동일한 sequence값을 갖는 패킷이다. 또한 보통의 경우는 원래의 패킷과 재전송된 패킷은 데이터와 그 크기가 같다. 그러나 TCP는 성능 향상을 목적으로 재전송할 때 더 전송할 데이터가 있다면 최대패킷크기 범위 내에서 패킷을 재조립하여 보내게 된다. 이것을 패킷의 재패킷화(Repacketization)이라고 하며 재패킷화 된 패킷이 재전송 될 경우 그림 3과 같은 문제가 발생한다.

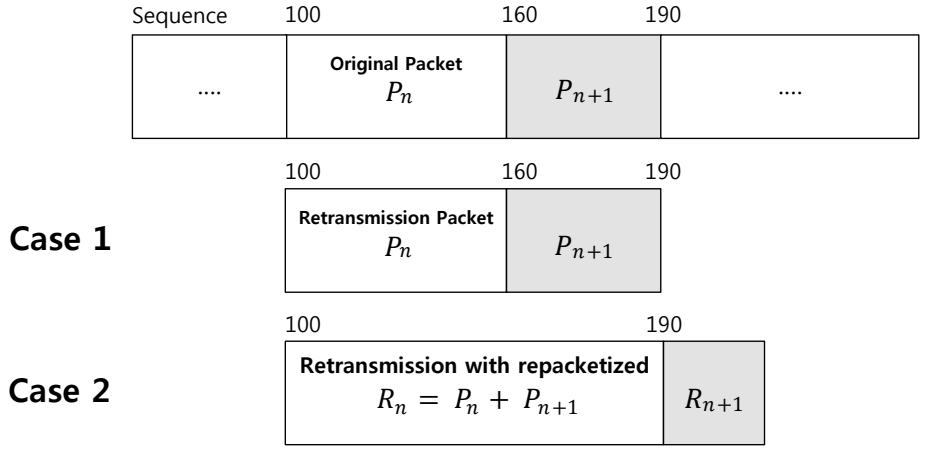


**그림 3. 재패킷화된 패킷의 재전송 문제**

그림 3은 재패킷화 된 패킷이 전송 될 경우 발생할 수 있는 문제를 설명하기 위한 그림이다. 원래의 패킷(Original Packet)은 수집지점을 통과 후 수신측으로 전달이 되었다. 수신측에서는 해당 패킷의 오류를 발견하고 송신측에게 재전송을 요청하였다. 송신측에서는 재전송 요청을 받은 후 패킷을 다시 보내게 되는데 Case1은 동일한 데이터를 가진 재전송 패킷을 보내는 경우이고 Case2는 재패킷화 되어 그 크기가 원래의 패킷보다 큰 패킷을 보내는 경우이다. Case1의 경우에는 원래의 패킷과 재전송된 패킷의 크기가 같으므로 이후에 오는 패킷의 sequence값은 160으로 같다. 따라서 수집지점에서는 재전송된 패킷만 삭제하면 문제가 해결된다. 하지만 Case2의 경우는 원래의 패킷과 재전송된 패킷의 크기가 다르므로 이후에 오는 패킷의 sequence값은 원래의 패킷과 맞지 않게 된다. 따라서 수집지점에서는 재패킷화 되어 재전송된 패킷을 삭제하고 또한 이후에 전송될 수 있는 재패킷화된 패킷과 관계하는 패킷또한 삭제해야 문제를 해결할 수 있다.

## 4. 해결 방법

본 장에서는 수집지점에서 발생하는 패킷 역전 탐지 및 해결 모듈, 패킷 재전송에 의한 패킷 중복 탐지 및 해결 모듈을 각각의 모듈로 구분하여 각각의 모듈에 대한 알고리즘을 기술한다.

### 4.1 패킷 역전 문제 해결 방법

본 절에서는 기존 연구[10]에서 패킷 역전 문제를 탐지하고 해결하는 알고리즘의 한계점을 분석하고 이를 개선한 알고리즘에 대해 기술한다. 본 논문에서 표현하는 패킷의 방향은 출발지 주소와 목적지 주소에 의해 결정된다. 플로우 기준으로 처음 발생한 패킷의 방향을 순방향으로 정의한다. 순방향 패킷과 출발지 주소, 목적지 주소가 서로 반대일 경우 반대방향 패킷으로 정의한다.

기존 연구에서는 순서에 맞지 않는 패킷의 올바른 위치를 찾을때 해당 플로우 내의 같은 방향의 패킷만 검사하여 위치를 정하였다. 순서에 맞지 않으면 같은 방향의 패킷들 사이에서 패킷의 위치를 정하였고, 반대 방향으로 전송되는 주변 패킷과의 관계는 고려하지 않았다. 따라서 역전이 발생한 패킷의 응답패킷이 있어도 응답패킷은 반대방향의 패킷이므로 검사를 하지 않고 위치를 찾게 된다. 따라서 원래의 패킷 순서를 찾기 위해서는 같은 방향뿐만 아니라 옮겨질 위치까지의 역방향 패킷과의 관계도 생각하여 처리해야 정확한 원래의 패킷 순서를 복원할 수 있다.

표 1은 패킷 역전 문제를 해결하기 위한 알고리즘의 의사코드이다. (줄 3) 알고리즘의 입력으로 들어온 패킷 P(n)과 방향이 같은 패킷들 중 가장 가까이 위치한 P(k)를 찾는다. (줄 4) 이렇게 찾은 P(k)와 P(n)의 sequence값을 비교한다. P(n)의 sequence값이 P(k)의 sequence값보다 크면 순서가 바뀐 패킷을 탐지하고 문제를 해결한다. (줄 6) P(n)과 방향이 같으면서 P(n)의 sequence값보다 작은 값을 갖는 P(i)를 찾는다. (줄 7) 이렇게 찾은 P(i)와 방향이 같으면서 P(i)의 sequence값보다 큰 패킷들 중에서 가장 가까이 위치한 P(j)를 찾는다. 이렇게 찾은 P(i)와 P(j)사이에 P(n)과 반대방향을 갖는 패킷이 없다면 P(n)은 P(i)의 뒤에 위치시킨다. (줄 10) 만약 방향이 다른 패킷들(P(m))중 P(m).ack == P(n).seq + P(n).len을 만족하는 패킷 P(m)이 존재하면P(n)은 P(m)의 앞에 위치시킨다. (줄 12) 또는 P(m).ack < P(n).seq + P(n).len을 만족하는 패킷 P(m)이 존재하면 (줄 13) P(n)은 P(m)의 뒤에 위치시킨다.

```
    Remove all non-payload packets from the packet sequence
    P(n) : n-th packet in a TCP flow
    P(n).seq : n-th packet's sequence nunmber
    P(n).ack : n-th packet's acknowledge nunmber
    P(n).dir : n-th packet's direction
    P(n).len : n-th packet's payload length
1   module Solution for the Out-of-order problem
2:  Input : P(n) in a TCP Flow
3:  find P(k) which P(k).dir == P(n).dir && biggest k in 0 <= k < n
4:  if(P(k).seq > P(n).seq)   // out-of-order detect
5:  {
6:      find P(i) which P(i).dir == P(n) && P(i).seq < P(n).seq
7:      find P(j) which P(j).dir == P(i) && smallest j in 0 <= j < k
8:      for each P(m) from P(j-1) to P(i+1)
9:      {
10:         if(P(m).dir != P(n).dir && P(m).ack == P(n).seq + P(n).len)
11:             put P(n) before P(m);       end module;
12:         if(P(m).dir != P(n).dir && P(m).ack < P(n).seq + P(n).len
13:             put P(n) after P(m);        end module;
14:     }
15:     put P(n) after P(i);
16: }
17: end module;
```

**표 1. 패킷 역전 문제 해결 알고리즘**

### 4.2 패킷 중복 문제 해결 알고리즘

본 절에서는 기존 연구[3]에서 패킷 재전송에 의한 중복 문제를 탐지하고 해결하는 알고리즘의 문제점을 분석하고 이를 개선한 알고리즘에 대해 기술한다.

패킷 중복 문제는 플로우 내 패킷들의 방향과 sequence값을 비교하여 탐지한다. 동일한 플로우 내에 두 개 이상의 패킷이 같은 방향이면서 동일한 sequence값을 갖게 되면 중복 패킷으로 탐지한다. 또한 같은 방향, 동일한 sequence를 가지면서 패킷의 크기가 다른경우에는 재패킷화된 패킷 중복 문제로 탐지한다. 기존 연구에서는 중복 패킷이 발생하면 오류가 났던 패킷은 삭제하고 중복 패킷을 저장하였다. 하지만 원래의 통계정보를 찾는 것이 목적이므로 본 논문에서는 원래의 패킷은 저장하고 중복된 패킷을 삭제하였다.

```
Remove all non-payload packets from the packet sequence
P(n) : n-th packet in a TCP flow
P(n).seq : n-th packet's sequence nunmber
P(n).ack : n-th packet's acknowledge nunmber
P(n).dir : n-th packet's direction
P(n).len : n-th packet's payload length
1:  module Solution for the Retransmission problem
2:  Input : P(n) in a TCP Flow
3:  find P(k) which P(k).dir == P(n).dir && biggest k in 0<= k < n;
4:  if( P(k).seq == P(n).seq ) // Retransmission
5:      Delete P(n);
6:  else if( P(k).seq + P(k).len !=P(n).seq )
7:      Delete P(n);
8:  end module;
```

**표 2. 패킷 중복 문제 해결 알고리즘**

표 2는 패킷 재전송에 의한 중복 패킷 문제를 해결하기 위한 알고리즘의 의사코드이다. (줄 3) 알고리즘의 입력으로 들어온 P(n)과 방향이 같은 패킷들 중 가장 가까이 위치한 P(k)를 찾는다. (줄 4)이렇게 찾은 P(k)의 sequence값과 P(n)의 sequence값을 비교하여 같으면 패킷 재전송에 의한 중복 패킷 문제를 탐지하고 삭제한다. (줄 6) 만약 P(n)이 P(k)의 재전송 패킷이 아니라도 P(k)의 sequence값과 패킷의 페이로드 길이를 더한 값이 P(n)의 sequence값과 같이 않을 경우에는 P(n)이 이전에 재패킷화 되어 재전송된 패킷과 관계하는 패킷이라 간주하여 삭제된다.

## 5. 실험

본 장에서는 4장에서 기술한 패킷 역전 문제, 패킷 중복 문제를 해결하는 알고리즘의 성능을 평가하기 위해 두 가지 실험을 하였다. 첫 번째 실험에서는 제안한 알고리즘을 실제 트래픽에 적용하여 패킷 역전, 중복 문제의 발생 빈도, 해결 정도를 살펴 성능을 평가한다. 두 번째 실험에서는 해당 문제를 해결하기 전과 해결한 이후의 응용 분석량을 비교하여 성능을 평가한다.

### 5.1 실험 데이터

본 논문에서 기술한 알고리즘의 성능 평가를 위해 9개의 응용을 선정하여 실험용 트래픽 데이터를 수집하였다. 통계정보를 이용한 시그니쳐 생성용으로 수집한 데이터는 85GB이며 이중 TCP는 28GB, 57GB이다. 이렇게 생성된 시그니쳐를 이용하여 분석하기 위해 수집한 데이터의 총량은 약 366GB이며, 이중 TCP는 82GB, UDP는 291GB이다. 분석용 트래픽중에서 뮤토렌트(µTorrent). 응용에서 발생한 트래픽은 348GB이다.

### 5.2 실험 환경

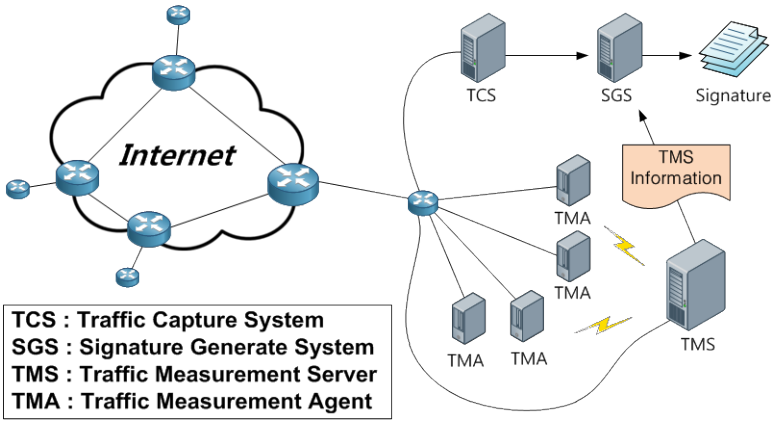


**그림 4. 통계적 특징 기반 시그니쳐 추출 시스템**

본 논문에서 성능평가에 사용하는 통계적 특징 기반 시그니쳐 추출 시스템은 그림 4와 같은 환경에서 동작한다. 먼저 TCS(Traffic Capture System)에서 Validation Network의 트래픽을 수집하고 TMS(Traffic Measurement Server)에서 TMA 로그(Traffic Measurement Agent)를 추출 및 이용하여 정답지(Ground Truth)[12]를 생성한다. 이렇게 생성된 정답지로 SGS(Signature Generation System)에서 시그니쳐를 추출한다. 이러한 Agent를 통한 정답지 생성 방법은 특정 분류 방법을 통해 분류한 결과를 사용한 것[13]보다 높은 신뢰성을 보장해 준다. 제안하는 패킷 역전 문제, 패킷 중복 문제 해결은 시그니쳐를 추출하기 전, 패킷을 수집하는 TCS에서 행해야 한다.

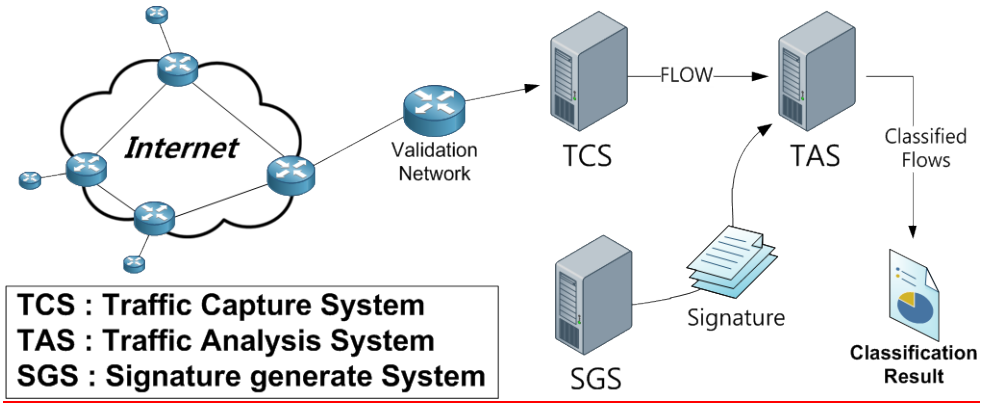


**그림 5. 트래픽 분류 시스템**

그림 5는 본 논문에서 성능 평가에 사용하는 트래픽 분류 시스템이 동작하는 환경이다. 먼저 TCS에서 Validation Network의 트래픽을 수집하고 TAS(Traffic Analysis System)에서 시그니쳐를 이용하여 수집된 트래픽을 응용별로 분류한다. 시그니쳐 추출 시스템과 마찬가지로 제안하는 Out-of-order, Retransmission 문제 해결은 점선으로 된 사각형으로 표시한 TCS에서 행해야 한다.

### 5.3 실험 결과

본 절에서는 4장에서 제안한 패킷 역전, 중복 문제를 해결하는 알고리즘을 5.2절에서 설명한 실험환경에 적용시킨 결과를 기술한다. 본 논문에서 제안하는 알고리즘의 성능평가를 위해 두 가지 실험을 진행하였다. 첫 번째 실험에서는 제안된 알고리즘을 실제 트래픽에 적용하여 문제의 해결 정도를 살펴 성능을 평가한다. 두 번째 실험에서는 이상 동작의 처리 후 통계적 특징 기반 트래픽 분류 시스템의 성능이 향상되는지 검증하기 위해 패킷 역전 문제, 패킷 중복 문제 해결 전 후 시스템에 적용 실험하여 성능을 비교 평가한다. 두 번째 실험에서 사용하는 통계 시그니쳐 기반 트래픽 분류 시스템[11]은 앞서 플로우의 첫 5개 패킷의 페이로드 크기와 전송 방향, 순서를 이용하여 5차원의 플로우 벡터로 표현한 뒤 그룹화 하여 시그니쳐를 추출하고, 마찬가지로 플로우의 첫 5개 패킷을 플로우 벡터로 표현한 뒤 추출된 시그니쳐를 적용해 트래픽을 분류하는 시스템이다.

| state | Flow | | Packet | | Byte | |
|---|---|---|---|---|---|---|
| | # | % | # | % | GB | % |
| **Normal** | 118,657 | 74.68 | 64,781K | 70.97 | 59.35 | 70.72 |
| **Out-of-order** | 50 | 0.03 | 28K | 0.02 | 0.02 | 0.02 |
| **Retransmission** | 36,974 | 23.27 | 23,295K | 25.52 | 22.1 | 26.33 |
| **Repacketization** | 3,186 | 2.1 | 3,169K | 3.47 | 2.45 | 2.91 |
| **TCP Total** | 158,867 | | 91,275K | | 83.92 | |

**표 3. TCP 세션에서 패킷 역전, 중복, 재패킷화 발생 비율**

표3은 본 논문에서 사용된 분석용 트래픽에서 발생하는 패킷 역전, 중복, 재패킷화 문제의 발생정도를 나타낸 표이다. 패킷 역전 문제는 전체 TCP 트래픽 기준 Flow 0.03%, Packet 0.02%, Byte 0.02퍼센트로 문제가 많이 발생하지 않았다. 재전송에 의한 패킷 중복 문제는 전체 TCP 트래픽 기준 Flow 23.27%, Packet 25.52%, Byte 26.33%를 차지하였다. 재패킷화 문제 또한 Flow 2.1%%, Packet 3.47%, Byte 2.91%를 차지하였다.

| Application | Total | | | Analyzed | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Flow | Packet | Byte | Flow | | Packet | | Byte | |
| | | | | not resolved | resolved | not resolved | resolved | not resolved | resolved |
| skype | 1,141 | 22K | 8,593K | 751 | 794 | 15K | 16K | 6,978K | 7,400K |
| naverlive | 1,218 | 22,421K | 18,159,291K | 1,015 | 1,026 | 20,049K | 20,395K | 16,246,821K | 16,537,135K |
| gomTV | 981 | 826K | 810,302K | 701 | 702 | 745K | 746K | 732,854K | 733,415K |
| xshell | 403 | 64K | 8,333K | 389 | 388 | 63K | 62K | 8,130K | 8,107K |
| teamviewer | 485 | 231K | 74,359K | 426 | 431 | 196K | 197K | 62,835K | 62,875K |
| nateon | 299 | 103K | 12,119K | 272 | 273 | 102K | 102K | 11,682K | 11,684K |
| dropbox | 4,642 | 124K | 66,631K | 4,599 | 4,612 | 122K | 123K | 65,505K | 65,774K |
| putty | 266 | 27K | 4,115K | 261 | 262 | 27K | 27K | 4,080K | 4,096K |
| outlook | 4,872 | 496K | 261,193K | 3,515 | 3,420 | 324K | 321K | 154,314K | 154,029K |
| uTorrent | 417,724 | 422,910K | 374,298,568K | 216,016 | 208,475 | 157,435K | 158,246K | 136,173,646K | 137,105,776K |
| Total | 432,031 | 447,224K | 393,703,504K | 227,945 | 220,383 | 179,078K | 180,235K | 153,466,845K | 154,690,291K |

**표 4. 패킷 역전, 중복, 재패킷화 문제 해결 전 / 후 응용 분류 실험 결과**

표 4는 본 논문에서 제안한 알고리즘으로 패킷 역전, 중복, 재패킷화 문제 해결 전, 후의 응용 분류 실험 결과를 나타낸 표이다. 대부분의 응용들은 문제 해결 후 플로우, 패킷, 바이트별 분석량이 상승한 것을 확인할 수 있었다. 그러나 xShell, outlook은 플로우, 패킷, 바이트별 분석량이 각각 소폭 하락하였다. 해당 분석되지 않은 플로우들을 살펴본 결과 재전송 패킷 2~3개로 이루어진 플로우들 이었다. 패킷 중복 문제 해결 후에는 이러한 재전송 패킷들은 1개를 제외하고 모두 삭제되어 해당 플로우가 분석 대상에서 제외되어 나타나는 현상이었다. 또한 uTorrent의 경우에는 플로우는 분석량이 줄었음에도 불구하고 패킷과 바이트 분석량이 늘어나는 현상을 보였다. uTorrent 또한 재전송 패킷들로 인해 해당 플로우들이 분석 대상에서 제외되어 나타나는 현상으로 확인되었고, 이러한 결과는 트래픽 망에 큰 부담을 주는 heavy플로우를 더 정확히 분석한 것을 의미한다.

결론적으로 본 논문에서 제안한 알고리즘을 적용함으로써 여러 응용이 발생시키는 트래픽 중에서 실제 트래픽 망에 부담을 줄 수 있는 heavy플로우의 시그니쳐를 생성하고, 생성된 시그니쳐를 바탕으로 트래픽 분석률 또한 향상시킬 수 있다는 것이다.

| | Accuracy | | | Completeness | | |
|---|---|---|---|---|---|---|
| | Flow | Packet | Byte | Flow | Packet | Byte |
| **Not resolved problem** | 99.80% | 99.11% | 99.40% | 52.76% | 40.04% | 38.98% |
| **Resolved problem** | 99.82% | 99.20% | 99.40% | 52.01% | 40.30% | 39.29% |

**표 5. 실험 결과 분석률 및 정확도**

표 5는 실험 결과의 분석률 및 정확도를 나타낸 표이다. 본 논문에서 제시한 알고리즘을 적용하였을 때가 적용하지 않았을 때보다 플로우를 제외한 패킷과 바이트 측면에서는 적게나마

좋은 결과를 나타냈다.

## 6. 결론

본 논문에서는 TCP 세션에서 발생하는 패킷의 역전 문제, 재전송 문제 그리고 재패킷화 문제에 대해 기술하고 해결하는 알고리즘을 제안하였다. 그리고 제안한 알고리즘을 실제 트래픽 분석 시스템에 적용하여 응용별 최대 4%의 탐지 및 분석률 향상을 보임으로써 알고리즘의 성능을 입증하였다. 특히 uTorrent 응용의 경우 플로우 분석량이 8,000개 감소했음에도 불구하고 바이트 분석량은 1GB 가량 상승하였다. 이러한 결과는 트래픽 망에 큰 부담을 주는 heavy플로우의 분석량이 증가하였음을 나타낸다.

향후 연구로는 본 논문의 실험에서 분석률이 하락한 응용에 대한 다양하고 구체적인 연구를 진행하여 분석률 하락의 원인을 밝히고, 분석률을 향상시키기 위해 연구할 계획이다.

## 참고 문헌


[1] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in Proc. CoNext 2006. Conf. Future Netw. Technol., 2006.

[2] Y.-T Han and H.-S. Park, "Game traffic classification using statistical characteristics at the transport layer," ETRI J., vol. 32, no. 1, pp. 22-32, Feb. 2010.

[3] 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 Trans. Knowledge Discovery from Data, vol. 6, pp. 1-34, 2012.

[4] J. Tan, X. Chen, and M. Du, "An internet traffic identification approach based on GA and PSO-SVM," J. Computers, vol. 7, pp. 19-29, 2012.

[5] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based machine learning method for accurate internet traffic classification," Inf. Syst. Frontiers, vol. 12, pp. 149-156, Apr. 2010.

[6] S. Runyuan, Y. Bo, P. Lizhi, C. Yuehui, Z. Lei, and J. Shan, "Traffic classification using probabilistic neural networks," in Proc. ICNC, vol. 4, pp. 1914-1919, Yantai, Shandong, Aug. 2010.

[7] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in Proc. ACM CoNEXT, Lisboa, Portugal, 2006.

[8] T. Bujlow, T. Riaz, and J. M. Pedersen, "A method for classification of network traffic based on C5.0 Machine Learning Algorithm," in Proc. ICNC, pp. 237-241, Maui, HI, Feb. 2012.

[9] C. Yin, S. Li, and Q. Li, "Network traffic classification via HMM under the guidance of syntactic structure," J. Comput. Netw., vol. 56, pp. 1814-1825, Apr. 2012.

[10] H. M. An, J. H. Choi, J. H. Ham, and M. S. Kim. "A method to resolve the limit of traffic classification caused by abnormal TCP session," KNOM Rev., vol. 15, no. 1, pp. 31-39, Dec. 2012.

[11] J. W. Park and M. S. Kim "Performance improvement of the statistic signature based traffic identification system," J. KIPS, vol. 18-C, no. 4, Aug. 2011.

[12] B. C. Park, Y. J. Won, M.-S. Kim, and J. W. Hong, "Towards automated application signature generation for traffic identification," in Proc. IEEE Network Operations and Management Symp. (NOMS), pp. 160-167, Salvador, Bahia, Apr. 2008.

[13] 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 Commun. Rev., vol. 39, pp. 12-18, 2009.



**이 수 강**

2014년:고려대학교 컴퓨터 정보학과 졸업

2014년~현재:고려대학교 컴퓨터 정보학과 석사과정

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



**안 현 민**

2012년:고려대학교 컴퓨터 정보학과 졸업

2012년~현재:고려대학교 컴퓨터 정보학과 석사과정

<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

**심 규 석**
2014년 : 고려대학교 컴퓨터 정보학과 졸업
2014년~현재:고려대학교 컴퓨터 정보학과 석사과정
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

**김 명 섭**
1998년:포항공과대학교 전자 계산학과 졸업
2000년:포항공과대학교 컴퓨터 공학과 석사
2004년:포항공과대학교 컴퓨터 공학과 박사
2006년:Post-Doc. Dept. of ECE, Univ. of Toronto, Canada
2006년~현재: 고려대학교 컴퓨터정보학과 부교수
<관심분야> 네트워크 관리 및 보안, 트래픽 모니터 링 및 분석, 멀티미디어 네트워크