

# 트래픽 수집지점에서 발생하는 TCP패킷중복 및 역전문제 해결 방법

이수강\*, 안현민\*, 김명섭<sup>o</sup>

## A Method to Resolve TCP Packet Out-of-order and Retransmission Problem at the Traffic Collection Point

Su-Kang Lee\*, Hyun-Min An\*, Myung-Sup Kim<sup>o</sup>

### 요약

최근 급격한 인터넷의 발전으로 효율적인 네트워크관리를 위해 응용 트래픽 데이터 분석의 중요성이 강조되고 있다. 네트워크 관리를 위해 관리자는 트래픽 데이터를 각각 어떠한 응용에서 발생 하였는지 탐지할 수 있어야 한다. 응용을 탐지하기 위한 방법들 중 하나인 통계정보 트래픽 분류방법을 사용하여 트래픽을 분류할 수 있지만, 이러한 통계정보를 그대로 사용하여 분류하기에는 트래픽 수집지점에서 발생하는 패킷 역전, 재전송에 의한 패킷 중복과 같은 문제점들이 있다. 본 논문에서는 응용에서 발생된 트래픽의 탐지 및 분석률 향상을 위해 패킷 역전 문제와 재전송에 의한 패킷 중복 문제를 탐지하고 개선하는 방법론을 제안하였다. 이렇게 제안한 개선 방법론을 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.

**Key Words** : retransmission, out-of-order, statistic signature, network management, traffic analysis

### ABSTRACT

With the rapid growth of Internet, the importance of application traffic analysis is increasing for efficient network management. The statistical information in traffic flows can be efficiently utilized for application traffic identification. However, the packet out-of-order and retransmission occurred at the traffic collection point reduces the performance of the statistics-based traffic analysis. In this paper, we propose a novel method to detect and resolve the packet out-of-order and retransmission problem in order to improve completeness and accuracy of the traffic identification. To prove the feasibility of the proposed method, we applied our method to a real traffic analysis system using statistical flow information, and compared the performance of the system with the selected 9 popular applications. The experiment showed maximum 4% of completeness growth in traffic bytes, which shows that the proposed method contributes to the analysis of heavy flow.

\* 본 논문은 BK21 플러스 사업 및 2012년 정부(교육과학기술부)의 재원으로 한국연구재단(2012R1A1A2007483) 및 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단-차세대정보 컴퓨팅기술개발사업(2010-0020728)의 지원을 받아 수행된 연구임

• First Author : Korea University Department of Computer and Information Science, sukanglee@korea.ac.kr, 학생회원

◦ Corresponding Author : Korea University Department of Computer and Information Science, tmskim@korea.ac.kr, 종신회원

\* Korea University Department of Computer and Information Science, queen26@korea.ac.kr, 학생회원

논문번호 : KICS2014-03-102, Received March 28, 2014; Revised May 27, 2014; Accepted May 27, 2014

## I. 서론

초고속 인터넷의 보급과 인터넷 기반의 서비스가 다양화됨에 따라 네트워크 관리의 중요성이 강조되고 있다. 이러한 상황 속에서 효과적인 네트워크의 관리를 위해 해당 트래픽을 발생시킨 응용을 탐지할 수 있어야 한다. 통계정보 트래픽 분류방법<sup>1,2)</sup>은 트래픽을 발생시킨 응용을 탐지하고 분류하는 방법들 중 하나이다.

통계정보 트래픽 분류방법은 수집된 패킷의 크기와 전송방향, 전송순서, 그리고 수집된 시간 등의 feature를 사용하여 분류한다. 하지만 수집된 통계정보를 그대로 사용하여 트래픽을 분류하기에는 한계가 있는데 그것은 TCP세션에서 발생하는 패킷 역전 문제(Out-of-order), 패킷 재전송에 의한 패킷 중복 문제(Retransmission)이다. 이러한 문제를 해결하지 않고 그대로 사용한다면 통계정보를 이용하여 분류할 때 사용되는 트래픽의 feature가 달라져서 100% 정확한 트래픽 분류가 어렵다.

패킷 역전 문제는 송신측이 패킷을 보낼 때 단일경로가 아닌 여러 경로를 통해 전송될 때 해당 경로의 상황에 따라 패킷들의 순서가 송신 측이 보낸 순서가 아닌 다른 순서로 수신측에 전달될 수 있다. 마찬가지로 서로 다른 경로를 통해 전송될 경우 해당 경로의 상황에 따라 수신측에서 보낸 순서가 아닌 다른 순서로 수집지점을 통과하게 된다. 트래픽 수집지점에서 발생하는 패킷 역전 문제는 이러한 패킷들을 원래의 순서가 아닌 수집지점을 통과한 순서대로 패킷을 저장하기 때문에 발생하는 문제이다.

패킷 재전송에 의한 패킷 중복 문제는 송신측에서 보낸 패킷이 유실되어 수신측에 전달되지 않거나 전송된 패킷에서 오류가 발견되었을 경우 수신측에 요청에 의해 송신측은 해당 패킷을 재전송하게 된다. 이때 중간에 위치한 수집지점에는 재전송된 패킷과 원래의 패킷이 중복되어 수집될 수 있다. 이렇게 수집지점에서 동일한 Sequence번호를 가진 패킷이 중복되어 저장되는 문제를 패킷 재전송에 의한 패킷 중복 문제라 한다.

본 논문에서는 트래픽 수집지점에서 발생하는 패킷 역전, 패킷 중복 문제를 해결하는 알고리즘을 제안하고자 한다. 이러한 문제를 수집지점에서 처리하면 특정 응용에서 발생시킨 원래 트래픽의 통계정보와 동일한 트래픽 통계정보를 수집할 수 있다. 이렇게 수집한 통계정보를 바탕으로 해당 응용의 통계정보 특성을 시그니처로 만들고 이를 이용하여 해당 응용을 정확히 탐지할 수 있다.

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

## II. 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 방법들은 응용별 인터넷 트래픽의 특징이 될 수 있는 항목(Packet 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 응용별로 분류함에 있어 다른 방법에 비해 보다 높은 정확도를 제공한다는 것이다.

기존의 일부 연구들<sup>3-6)</sup> 트래픽을 분류하기 위한 속성들로 플로우 전체 단위의 통계적 특성을 사용하기 때문에 실시간 트래픽 분류에 사용할 수 없다. 이러한 문제를 해결하기 위해 플로우의 처음 N개 패킷에서 속성을 추출하는 방법들이 연구되었으나<sup>7-9)</sup> 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.

기존의 모든 연구에서는 트래픽 수집지점에서 발생하는 패킷 역전 문제와 패킷 재전송에 의한 중복 패킷 문제로 인해 일어나는 통계 정보의 변화에 대한 문제는 다루어지지 않았다. 따라서 본 연구에서는 트래픽 수집지점에서 발생하는 패킷 역전 문제와 패킷 재전송에 의한 중복 패킷 문제를 정의하고 해결하는 알고리

증을 제안하고자 한다.

### III. 문제 정의

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

TCP는 두 중점 호스트 사이에 신뢰성 있는 스트림 형태의 채널을 제공함으로써 상위계층(응용 프로그램) 사이에 투명한 데이터 전달을 제공한다. 중점 간 신뢰성 있는 데이터 송수신 보장을 위해 TCP는 Sequence와 Acknowledge, checksum 등의 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. 수신 측에서 오류를 발견하면 오류가 발견된 데이터를 버리고 Ack를 보내지 않음으로써 송신 측에 오류가 났음을 알려거나 최근 정상적으로 수신한 데이터에 대한 Ack를 반복하여 보냄으로써 송신 측이 Ack 이후의 데이터에서 오류가 발생한 것(또는 패킷이 분실된 것)을 알 수 있도록 하여 해당 패킷을 재전송 하도록 한다.

두 중점 호스트 간에 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 중점 호스트에서 발생하는 패킷 역전 문제라 하며 수신 측의 TCP는 이러한 문제 해결을 위해 패킷의 Sequence를 확인하고 패킷의 순서를 재 정렬하게 된다.

그림 1에서 Host A가 Host B에게 여러개의 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 수행한다. 하지만 트래픽 수집지점(CP)에서 수집된 트래픽을 저장하는 시스템(TCS)의 TCP 전송계층에서는 재 정렬 작업을 실행하지 않으므로 TCS에서

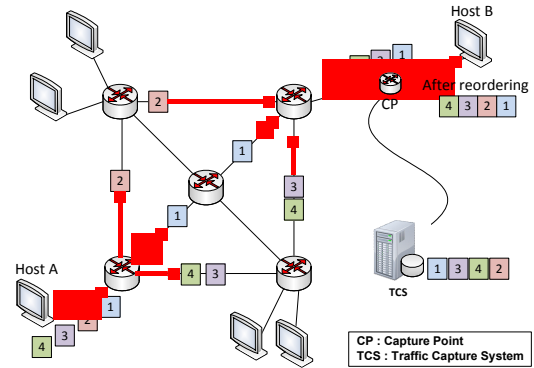


그림 1. 패킷 역전 문제  
Fig. 1. Packet Out-of-order problem

장되는 트래픽은 Host A에서 보낸 패킷 순서와 맞지 않게 된다. 이러한 문제를 수집지점에서 발생하는 패킷 역전(Out-of-order) 문제라 한다.

패킷 재전송은 송신측에서 보낸 패킷이 수신측으로 전송 중 어떠한 이유로 유실되어 전달이 되지 않았거나 받은 패킷에 오류가 발견되었을 경우 수신측 요청에 의해 발생하는 것으로, 송신측은 오류로 인해 수신측까지 전송되지 못한 데이터와 해당 데이터의 Sequence 값을 가진 패킷을 재전송 하게 된다.

그림 2는 Host A가 Host B에게 여러 개의 패킷을 보낼 때 재전송 패킷이 발생하는 상황을 나타낸 그림이다. Host B는 Host A가 보낸 패킷 1, 2, 3 총 3개의 패킷을 받는다. Host B의 TCP는 2번 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 수집지점(CP)의 트래픽 수집시스템(TCS)의 TCP 전송계층

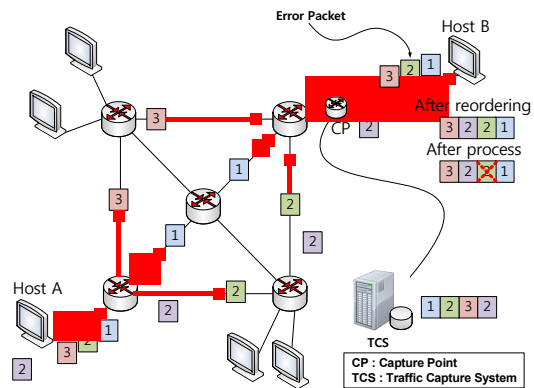


그림 2. 패킷 재전송에 의한 패킷 중복 문제  
Fig. 2. Packet Retransmission problem

에서는 재전송 패킷에 대한 특별한 처리과정이 없으므로 오류가 발견된 원래의 패킷과 재전송된 패킷이 모두 수집되며 이러한 문제를 수집지점에서 발생하는 패킷 중복 문제라 한다.

이와 같이 수집지점에서는 패킷 재전송에 의한 패킷 중복 문제로 인해 동일한 Sequence를 갖는 패킷이 여러 개 수집될 수 있으며 보통 재전송된 패킷은 원래의 패킷과 데이터, 크기가 동일하다. 하지만 TCP는 성능 향상을 목적으로 재전송할 때 더 전송할 데이터가 있다면 최대패킷크기 범위 내에서 패킷을 재조립하여 보내게 된다. 이것을 패킷의 재패킷화(Repaketization)라고 하며 재패킷화 된 패킷이 재전송될 경우 그림 3과 같은 문제가 발생한다.

그림 3은 재패킷화 된 패킷이 전송 될 경우 수집지점에서 일어날 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 패킷(P1과 P2)의 Sequence값은 160으로 같다. 따라서 수집지점에서는 재전송된 패킷만 삭제하면 문제가 해결된다.

하지만 Case2의 경우에는 원래의 패킷과 재전송된 패킷의 크기가 다르므로 이후에 오는 패킷(P3)의 Sequence값은 달라진다. 따라서 수집지점에서는 재패킷화 되어 재전송된 패킷을 삭제하고 또한 이후에 올 수 있는 패킷(P3)도 탐지하여 삭제해야 문제를 해결할 수 있다.

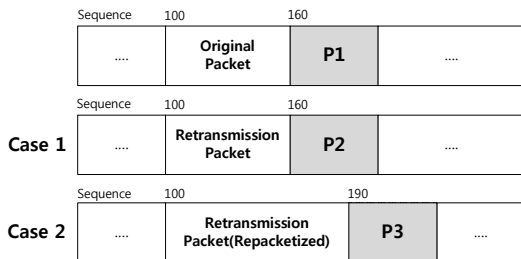


그림 3. 재패킷화 된 패킷의 패킷 중복 문제  
Fig. 3. Retransmission problem with repacketized

## IV. 해결 방법

### 4.1 패킷 역전 문제 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.

본 절에서는 기존 연구<sup>10)</sup>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. 아래 그림 4의 경우 패킷 a와 b는 같은 방향 패킷이며 패킷 a와 d는 반대 방향 패킷이다. 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

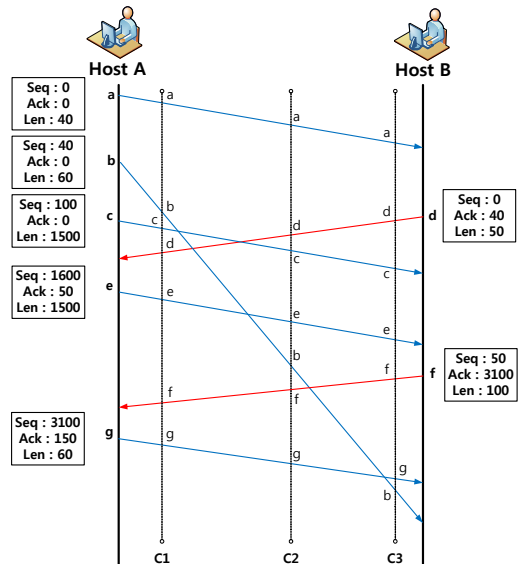


그림 4. 트래픽 수집지점에서 발생하는 패킷 역전 문제  
Fig. 4. Collect points by Out-of-order problem

패킷 순서를 복원할 수 있다.

그림 4는 여러 수집지점(C1, C2, C3)에서 각각의 수집지점에서 패킷 역전 문제가 발생하는 것을 나타낸 그림이다. Host A와 B는 a-b-d-c-e-f-g의 순서로 패킷을 주고받았다. 하지만 수집지점 C1에서는 a-b-c-d-e-f-g, 수집지점 C2에서는 a-d-c-e-b-f-g, 수집지점 C3에서는 a-d-c-e-f-g-b의 순서로 패킷이 수집되었다. 이는 수집지점별로 저장되는 패킷의 순서가 원래의 패킷순서가 아닌 수집지점을 통과하는 순서로 저장되기 때문이다.

따라서 수집지점에서 이러한 문제를 해결하기 위한 특별한 처리가 없다면 수집지점에 저장된 패킷은 원래의 패킷 순서와 다른 순서를 갖고 저장될 것이다.

표 1은 패킷 역전 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)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.

표 1. 패킷 역전 문제 해결 알고리즘  
Table. 1. Algorithm for Out-of-order problem

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 number
P(n).ack : n-th packet's acknowledge number
P(n).dir : n-th packet's direction
P(n).len : n-th packet's payload length
1 module Solution for the <i>Out-of-order</i> 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).dir && P(i).seq < P(n).seq
7: find P(j) which P(j).dir == P(i).dir && 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
13: put P(n) after P(m); end module;
14: }
15: put P(n) after P(i);
16: }
17: end module;

(줄 10) 만약 방향이 다른 패킷들(P(m))중 P(m).ack == P(n).seq + P(n).len을 만족하는 패킷 P(m)이 존재하면 P(m)은 P(n)의 응답패킷이므로 P(n)은 P(m)의 바로 앞에 위치시킨다. (줄 12) P(m).ack == P(n).seq 을 만족하는 패킷 P(m)이 존재하면 P(n)은 P(m)의 응답패킷이므로 (줄 13) P(n)은 P(m) 바로 뒤에 위치하게 된다.

#### 4.2 Retransmission 해결 방법

본 절에서는 기존 연구<sup>3)</sup>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 sequence 값을 갖게 되면 중복 패킷으로 탐지한다. 기존 연구에서 패킷 중복 문제를 해결할 때에는 오류가 일어난 원래의 패킷을 삭제하고 재전송된 패킷을 저장하였다. 이러한 경우에는 일반적인 중복 패킷이 있을 경우에는 문제가 되지 않는다. 4.2.2절에서 설명하는 재패킷화된 중복 패킷의 경우에 기존 방법대로 패킷 중복 문제를 해결한다면 원래의 패킷은 삭제하고 재패킷화된 패킷을 저장하게 된다. 이로 인해 발생하는 문제는 4.2.2절에서 자세히 설명한다. 재패킷화되어 재전송되는 패킷은 원래의 응용이 일반적으로 발생시키는 패킷은 아니다. 본 논문에서는 응용에서 의도했던 원래의 패킷의

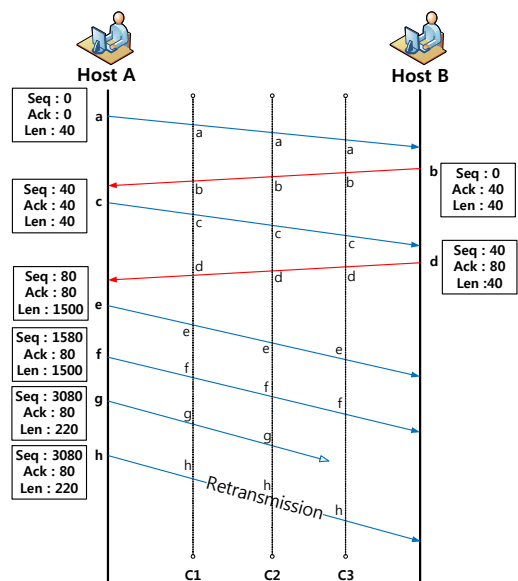


그림 5. 트래픽 수집지점에서 발생하는 패킷 중복 문제  
Fig. 5. Collect points by Retransmission problem

순서를 찾는 것이므로 재전송된 패킷이 아닌 원래의 패킷을 저장하고 재전송된 패킷을 삭제해야 한다. 또한 재전송된 패킷이 이전의 패킷과 sequence가 같으면서 그 크기가 다른 재패킷화 된 패킷일 경우에는 일반적인 패킷 중복 문제와 다르게 처리해야 한다.

#### 4.2.1 일반적인 패킷 중복 문제 해결

전송된 패킷에서 오류가 발견되었거나 일정 시간이 지나도 응답 패킷을 받지 못할 경우 재전송 요청에 의해 재전송 패킷이 발생한다. 재전송된 패킷은 원래의 패킷의 sequence와 방향이 같다.

그림 5는 일반적인 재전송 패킷이 발생한 패킷 중복 문제를 나타낸 그림이다. Host A와 Host B는 서로 패킷들을 주고받는다. 수집지점 C1, C2에서 패킷 h의 sequence와 패킷 g의 sequence가 같고, 방향이 서로 같으므로 패킷 h는 패킷 g의 재전송 패킷임을 알 수 있다. 재전송되어 패킷의 크기가 같고 동일한 sequence를 갖는 패킷을 삭제하여 일반적인 패킷 중복 문제를 해결할 수 있다.

#### 4.2.2 재패킷화 된 패킷 중복 문제 해결 방법

전송된 패킷에서 오류가 발견되었거나 일정 시간이 지나도 응답 패킷을 받지 못할 경우 재전송 요청에 의해 재전송 패킷이 발생한다. 이 때 성능 향상의 목적으로 더 전송할 데이터가 있다면 패킷의 최대 크기 범위 내에서 패킷의 크기를 늘려서 재전송을 한다. 이를 패킷의 재패킷화(repacketization) 라고 하며 이러한 패킷이 재전송 되는 현상을 재패킷화 된 패킷 중복 문제라 한다.

재패킷화 된 패킷이 재전송될 경우 이후에 오는 패킷의 sequence가 달라지게 되므로 재패킷화 되어 재전송된 패킷을 삭제하고 이와 관련된 sequence값을 갖는 패킷까지 삭제해야 재패킷화 된 패킷 중복 문제를 해결한다.

그림 6은 재패킷화 된 재전송 패킷이 발생한 중복 패킷 문제를 나타낸 그림이다. Host A와 Host B는 서로 패킷들을 주고받는다. 수집지점 C1, C2에서 패킷 g와 h의 sequence와 방향, 패킷의 크기가 서로 같으므로 패킷 h는 패킷 g의 일반적인 재전송 패킷이다. 하지만 패킷 g와 i는 sequence와 방향은 같지만 방향이 다르므로 패킷 i는 패킷 g의 재패킷화 된 재전송 패킷임을 알 수 있다. 패킷 j는 패킷 g나 h의 sequence값과 맞는 acknowledge값을 갖지만 패킷 k는 재패킷화 된 패킷인 i의 sequence와 관계하는 응답번호를 갖게 된다. 이 때 재패킷화 되어 재전송 된 패킷 i, 패킷 i의 sequence와

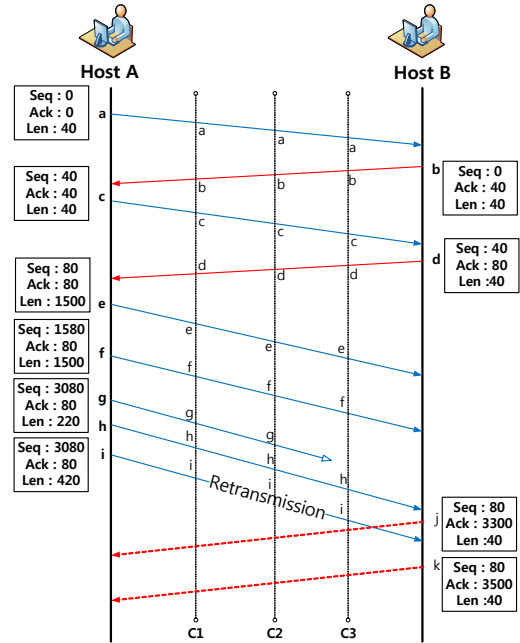


그림 6. 재패킷화 된 패킷의 패킷 중복 문제  
Fig. 6. Repacketized packet's retransmission problem

관계하는 패킷 k를 모두 삭제해야 재패킷화 된 패킷 중복 문제를 해결할 수 있다.

#### 4.2.3 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 아

표 2. 패킷 중복 문제 해결 알고리즘  
Table. 2. Algorithm for Retransmission problem

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 number
P(n).ack : n-th packet's acknowledge number
P(n).dir : n-th packet's direction
P(n).len : n-th packet's payload length
1: module Solution for the <i>Retransmission</i> 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;



나라도  $P(k)$ 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.

### V. 실험

본 장에서는 4장에서 제안된 패킷 역전 문제, 패킷 중복 문제 해결 알고리즘의 성능을 평가하기 위해 두 가지 실험을 하였다. 본 장에서는 통계적 특징 기반 시그니처 추출 시스템, 트래픽 분석 시스템을 소개한 후 패킷 역전 문제, 패킷 중복 문제 처리 전 후의 실험 결과를 분석하고자 한다.

#### 5.1 통계적 특징 기반 시그니처 추출 시스템

본 논문에서 성능평가에 사용하는 통계적 특징 기반 시그니처 추출 시스템은 그림 7과 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)<sup>[12]</sup>를 생성한다. 이렇게 생성된 정답지로 SGS(Signature Generation System)에서 시그니처를 추출한다.

이러한 Agent를 통한 정답지 생성 방법은 특정 분류 방법을 통해 분류한 결과를 사용한 것<sup>[13]</sup>보다 높은 신뢰성을 보장해 준다. 제안하는 패킷 역전 문제, 패킷 중복 문제 해결은 시그니처를 추출하기 전 단계인 점선으로 된 사각형으로 표시한 TCS에서 행해야 한다.

그림 8은 본 논문에서 성능 평가에 사용하는 트래픽 분류 시스템이 동작하는 환경이다. 먼저 TCS에서 Validation Network의 트래픽을 수집하고 TAS(Traffic Analysis System)에서 시그니처를 이용하여 수집된 트래픽을 응용별로 분류한다. 시그니처 추출 시스템과

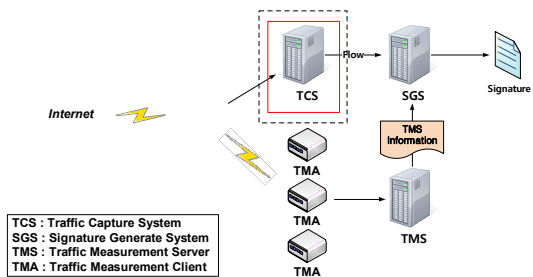


그림 7. 통계적 특징 기반 시그니처 추출 시스템 구성도  
Fig. 7. Signature extraction system configuration based on statistical features

마찬가지로 제안하는 Out-of-order, Retransmission 문제 해결은 점선으로 된 사각형으로 표시한 TCS에서 행해야 한다.

#### 5.2 실험 결과

본 장에서는 4장에서 제안된 패킷 역전 문제, 패킷 중복 문제를 해결하는 알고리즘의 성능을 평가하기 위해 두 가지 실험을 하였다. 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 시그니처 기반 트래픽 분류 시스템<sup>[11]</sup>은 앞서 플로우의 첫 5개 패킷의 페이로드 크기와 전송 방향, 순서를 이용하여 5차원의 플로우 벡터로 표현한 뒤 그룹화 하여 시그니처를 추출하고, 마찬가지로 플로우의 첫 5개 패킷을 플로우 벡터로 표현한 뒤 추출된 시그니처를 적용해 트래픽을 분류하는 시스템이다.

표 3은 본 논문에서 제안한 알고리즘을 실험에 사용한 트래픽 데이터에 적용한 결과이다. 패킷 역전 문제는 플로우 기준으로 전체의 1.33%, 패킷 기준으로 0.0125%, 바이트 기준으로 0.0185%를 차지하였다. 패킷 중복 문제는 플로우 기준으로 전체의 1.33%, 패킷 기준으로 1.93%, 바이트 기준으로 1.32%를 차지하였다. 재패킷화 문제는 플로우 기준으로 0.12%, 패킷 기준으로 0.0032%, 바이트 기준으로 0.57%를 차지하였다. 또한 패킷 역전 문제와 재패킷화 문제가 동시에 일어난 경우는 플로우 기준으로 0.08%를 차지하였다.

표 4는 본 논문에서 제안한 방법론의 적용 전과 후의 트래픽 분석량을 비교한 표이다. 대부분의 응용들은 플로우, 패킷, 바이트 각각 분석량이 상승한것을 확인할 수 있었다. Outlook과 XShell의 분석량은 플로우, 패킷, 바이트 각각 분석량이 하락하였다. 플로우 분석량 하락의 원인을 조사한 결과 재전송 문제 처리 전에는 많은 양의 재전송 패킷이 발생하여 이러한 재전송

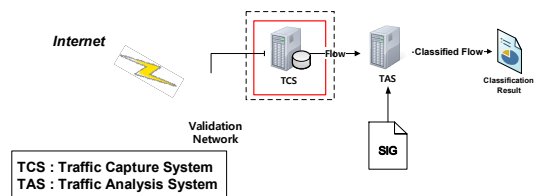


그림 8. 트래픽 분석 시스템 구성도  
Fig. 8. Traffic classification system configuration

표 3. 실험 데이터의 Out-of-order, Retransmission, Repacketization 문제 발생 비율  
Table. 3. Out-of-order, Retransmission, Repacketization rate in experimental traffic data

State	Flow		Packet		Byte	
	#	%	#	%	B	%
Normal	1,496,192	98.53	5,162,799	97.72	3,147,074,560	98.08
① Out-of-order	35	0.0023	665	0.0125	596,097	0.0185
② Retransmission	20,262	1.33	102,206	1.93	42,486,693	1.32
③ Repacketization	1,896	0.12	17,116	0.0032	18,425,930	0.57
① + ②	3	-	-	-	-	-
② + ③	1226	0.08	-	-	-	-
① + ② + ③	1	-	-	-	-	-
Total	1,518,385	-	5,282,786	-	3,208,583,280	-

표 4. Out-of-order, Retransmission, Repacketization 문제 해결 전 후 실험 결과  
Table. 4. Result of experiment before resolve problems / after resolve problems

Application	Total			Analyzed					
	Flow	Packet	Byte	Flow		Packet		Byte	
				before	after	before	after	before	after
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	105,957	25,833K	22,387,609K	73,338	70,388	21,039K	21,996K	18,167,867K	19,184,551K
Total	120,264	50,152K	41,792,549K	85,267	82,296	42,686K	43,989K	35,461,070K	36,769,071K

패킷을 바탕으로 시그니처가 생성 되었다. 이러한 재전송 패킷들은 본 논문에서 제안한 패킷 중복 문제 해결 알고리즘에 의해 제거되어 처리 후 생성된 시그니처와 분석 결과 하락에 영향을 주는 것을 확인 하였다.

패킷 중복 문제 해결 전에는 분석되었지만 해결 후에 분석되지 않은 플로우들을 분석한 결과 해당 플로우들은 재전송 패킷이 2~3개로만 이루어진 패킷 양이 적은 플로우였다. 패킷 중복 문제 해결 후에는 이러한 재전송 패킷들은 삭제되고 해당 플로우에는 원래의 패킷 1개만 남기 때문에 시그니처로 생성되지 않았다. 따라서 트래픽 분석기(TAS)는 해당 플로우들을 자동적으로 분석하지 않았던 것이다. 이러한 이유로 플로우 분석량이 하락했음에도 불구하고 패킷과 바이트의 분석량은 대폭 증가했다. 이러한 결과는 트래픽 망에 큰 부담을 주는 heavy 플로우를 더 정확히 분석할 수 있음을 의미한다. 결론적으로 본 논문에서 제안한 알고리즘을 적용함으로써 여러 응용이 발생시키는 트래픽

픽 중에서 실제 트래픽 망에 부담을 줄 수 있는 heavy 플로우의 시그니처를 생성하고, 생성된 시그니처를 바탕으로 트래픽 데이터의 분석을 또한 향상시킬 수 있다는 것이다.

## VI. 결론 및 향후 연구

본 논문에서는 TCP세션에서 발생하는 패킷의 역전(Out-of-order)문제, 재전송(Retransmission)문제, 그리고 재패킷화(Repackitization)문제에 대해 기술하였다. 그리고 위 문제를 해결하는 알고리즘을 제안하였으며 이를 실제 트래픽 분석 시스템에 적용시킴으로써 응용별 최대 4%의 탐지 및 분석률 향상을 보였다. 또한 전체 플로우 기준 총 분석량은 3% 감소하였지만 전체 패킷, 바이트 기준 4%의 분석률이 증가하였고 이는 실제 트래픽 망에 부담을 줄 수 있는 heavy플로우의 분석률이 증가하였음을 보였다.



향후 연구로는 본 논문의 타당성을 주장하기 위해 본 논문의 실험에서 사용된 트래픽의 용량(32GB)보다 많은 실험데이터를 사용함으로써 4% 이상의 분석률 향상을 보일 예정이다. 또한 플로우 분석률이 하락한 응용에 대한 연구도 계획이다.

### References

[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. Rizzo, 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 (Su-Kang Lee)



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 (Hyun-Min An)



2012년 : 고려대학교 컴퓨터 정보학과 졸업  
 2012년~현재 : 고려대학교 컴퓨터 정보학과 석사과정  
 <관심분야> 네트워크 관리 및 보안, 트래픽 모니터링 및 분석

김 명 섭 (Myung-Sup Kim)



1998년 : 포항공과대학교 전자  
계산학과 졸업

2000년 : 포항공과대학교 컴퓨  
터 공학과 석사

2004년 : 포항공과대학교 컴퓨  
터 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