통계기반 트래픽 분석 시스템의 패킷 역전 및 패킷 중복에 관한 연구

이수강, 심규석, 안현민, 김명섭

고려대학교 컴퓨터정보학과

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

요 약

최근 급격한 인터넷의 발전으로 효율적인 네트워크 관리를 위해 응용 트래픽 데이터 분석의 중요성이 강조되고 있다. 네트워크 관리자는 효과적인 네트워크의 관리를 위해 트래픽을 발생시킨 응용을 탐지할 수 있어야 한다. 특정 응용을 탐지하기 위한 여러가지 방법들중 하나인 통계정보 트래픽 분류 방법을 사용하여 트래픽을 분류할 수 있지만 수집지점에서 수집되는 트래픽에는 패킷 역전, 재전송에 의한 패킷 중복과 같은 문제점들이 있다. 본 논문에서는 이러한 문제점들을 해결하는 방법론을 제안하고 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.

1. 서 론

최근 급격한 네트워크의 고속화와 다양한 서비스의 등장으로 오늘날의 트래픽은 복잡 다양해지고 있다. 이러한 상황 속에서 효과적인 네트워크 관리의 중요성 또한 강조되고 있다. 효과적인 네트워크의 관리를 위해서는 트래픽을 발생시킨 응용을 탐지할 수 있어야 한다. 트래픽의 통계정보를 사용하여 트래픽을 분류하는 방법[1,2]은 트래픽을 발생시킨 응용을 탐지하고 분류하는 방법들 중 하나이다.

통계정보 트래픽 분류방법은 수집된 패킷의 크기와 전송방향, 전송순서, 그리고 수집된 시간 등의 feature 를 사용하여 분류한다. 수집지점에서 수집된 통계정보를 아무런 처리과정이 없이 사용하기에는 문제점이 있는데 그것은 수집지점에서 발생하는 패킷 역전 문제(Out-of-order), 패킷 재전송에 의한 패킷 중복 문제(Retransmission) 이다. 이러한 문제를 해결하지 않고 사용한다면 분류할 때 사용되는 트래픽의 feature 가 달라져서 100% 정확한 트래픽 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. 동시에 중간에 위치한 수집지점에서 트래픽을 수집하게 되면 수집되는 패킷의 순서 또한 송신측이 보낸 패킷의 순서와 맞지 않게 되는데 이러한 문제를 수집지점에서 발생하는 패킷 역전 문제라 한다.

수신측은 송신측으로부터 받은 패킷에서 오류가 발견되었을 경우 오류가 발견된 패킷과 동일한 패킷을 송신측에게 재전송 요청을 한다. 또는 송신측은 수신측에게 보낸 패킷의 응답패킷을 일정 시간 내에 받지 못할 경우 해당 패킷을 재전송 하게 된다. 이렇게 발생한 재전송 패킷은 중간에 위치한 수집지점을 통과하며 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 트래픽 통계정보는 각 응용의 트래픽 통계정보와 항상 동일한 트래픽 통계정보를 수집할 수 있다. 이렇게 수집한 응용의 통계정보 특성을 바탕으로 시그니쳐를 만들고 이를 사용하여 분석이 요구되는 트래픽 데이터에서 해당 응용을 정확히 탐지할 수 있다.

본 논문은 다음과 같은 순서로 구성된다. 2 장에서는 통계 정보를 이용하여 트래픽을 분류하는 기존의 연구들을 살펴보고 3 장에서는 본 논문에서 제안하는 패킷 역전, 중복 문제의 해결 방법에 대해 기술한다. 4 장에서는 제안한 해결 방법을 적용한 실험을 통해 본 논문의 타당성을 입증하고 마지막으로 5 장에서는 결론과 향후 연구를 언급한다.

2. 관련연구

최근 응용 트래픽의 통계적인 특성을 이용한 분류 방법은 많은 관심을 받으며 연구되었다. 대부분

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

의 연구들은 응용별 트래픽의 특징이 될 수 있는 항목(Port number, Connect duration, Packet inter-arrival time, Packet size)들을 머신러닝(Machine Learning) 알고리즘에 적용하여 트래픽을 분류한다. 이러한 방법은 최근 증가하고 있는 암호화된 트래픽의 분석에 용이하며, 패킷의 페이로드 정보를 분석하지 않아 상대적으로 오버헤드가 적고 개인정보 침해의 논란을 회피할 수 있는 장점을 가진다. 또한 머신러닝의 고급 알고리즘을 이용하므로 트래픽을 응용별로 분류함에 있어 다른 방법에 비해 보다 높은 정확도를 제공한다.

기존의 몇몇 연구들은[3,4,5] 트래픽을 분류하기 위한 속성들로 트래픽 전체 단위의 통계적 특성을 사용하기 때문에 실시간 트래픽 분류에 사용되지 못한다. 따라서 실시간 트래픽 분류를 위해 플로우의 처음 N 개의 패킷에서 속성을 추출하여 통계적 특성을 찾는 방법들이[6,7] 연구되었다. 그러나 속성 계산의 높은 오버헤드로 인해 빠른 속도의 네트워크에서의 실시간 분류에는 적용하기 어려웠다.

또한 기존의 모든 연구에서는 수집지점에서 발생할 수 있는 패킷 역전, 재전송에 의한 중복 패킷들 때문에 야기되는 통계 정보의 변화에 대한 문제는 다루지 않았다. 따라서 본 연구에서는 트래픽 수집지점에서 발생하는 패킷 역전 문제와 패킷 재전송에 의한 중복 패킷 문제를 정의하고 해결하는 알고리즘을 제안하고자 한다.

3. 본 론

본 장에서는 수집지점에서 일어나는 패킷 역전 문제와 패킷 재전송에 의한 패킷 중복 문제를 해결하기 위한 알고리즘을 제안한다.

송신측이 보낸 데이터가 목적지에 잘 전달 되었는지 확인하기 위해 수신측은 데이터가 오류없이 잘 도착한 것을 확인하여 송신측에게 응답패킷(Acknowledge)를 보낸다. 만약 전송된 패킷에서 오류가 발견되면 송신측에게 동일한 패킷의 재전송을 요청한다.

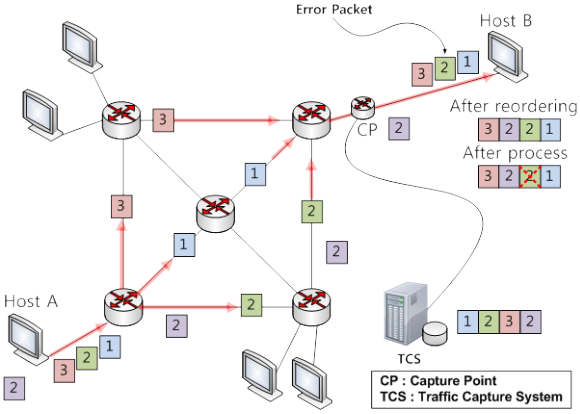

Fig. 1. Packet Retransmission problem

그림 1 은 Host A 에서 Host B 로 전송한 패킷들 중 오류가 있는 패킷으로 인해 동일한 패킷이 재전송 됨으로써 해당 패킷이 수집지점에 중복되어 저장되는 문제를 나타내는 그림이다. Host A 는 Host B 에게 3 개의 패킷을 보낸다. Host B 는 전송된 3 개의 패킷들의 오류를 검사한다. 이 때 전송된 패킷들 중 2 번 패킷에서 오류를 발견하고 Host B 는 Host A 에게 재전송 요청을 하게 된다. Host B 는 재전송된 패킷을 받음으로써 Host A 에서 원래 보내고자 한 패킷 3 개를 전부 받았다.

하지만 트래픽 수집지점(CP)를 통과한 패킷은 Host A 가 원래 보내려고 했던 3 개가 아닌 4 개의 패킷이다. 트래픽 수집 시스템(TCS)에서는 저장되는 패킷들에 대해 중복 처리를 하지 않으므로 원래 Host A 가 보내려고 했던 원래의 통계정보와 맞지 않게 된다. 이러한 문제는 특정 응용에서 보내고자 하는 통계정보와 수집지점에 저장되는 통계정보를 맞지 않게 하므로 통계 정보를 사용하여 해당 응용을 정확하게 탐지하는데 어려움이 있다. 이러한 문제를 수집지점에서 발생하는 재전송 패킷에 의한 패킷 중복(Retransmission) 문제라 한다.

또한 TCP 는 성능 향상의 목적으로 재전송할 데이터가 더 있다면 최대 패킷크기 범위 내에서 패킷을 재조립 하여 보내게 된다. 이것을 패킷의 재패킷화(Repacketization)라고 한다. 일반적인 재전송 패킷은 원래의 패킷과 Sequence 와 크기, 데이터가 같은 패킷이다. 반면에 재패킷화 되어 재전송되는 패킷은 원래의 패킷과 Sequence 만 같고 크기, 데이터는 다르다.

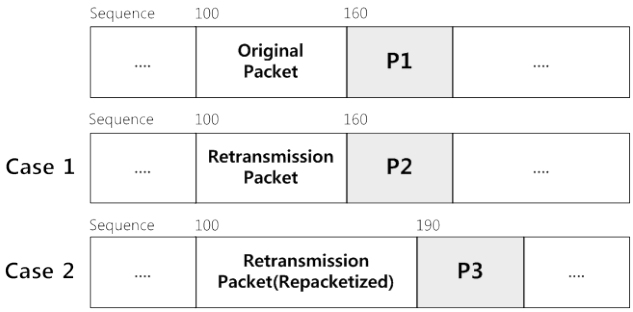

Fig. 3. Retransmission problem with repacketization

그림 2 는 재패킷화 되어 재전송 된 패킷으로 인해 다음에 올 수 있는 패킷의 Sequence 변화를 나타낸 그림이다. 재전송 된 패킷이 발생할 경우 원래의 통계정보를 유지하는 목적으로 원래의 패킷은 저장하고 재전송 된 패킷은 삭제한다. Case1 의 경우 원래의 패킷과 재전송된 패킷의 크기, 데이터가 같은 일반적인 패킷 중복 문제이다.

따라서 재전송 된 패킷을 삭제하면 이 후에 올 수 있는 패킷의 sequence 값이 서로 같으므로 문제가 없다. 하지만 Case 2 의 경우는 재패킷화 된 패킷이 재전송 되어 이 후에 올 수 있는 P1 과 P3 의 Sequence 값이 달라지게 된다. 따라서 재패킷화 되어 재전송 되는 패킷의 다음 패킷은 저장되는 원래의 패킷과 Sequence 가 맞지 않으므로 이와 같은

이후에 올 수 있는 패킷들까지 삭제해야 한다.

TCP 의 전송계층에서는 패킷들이 단일 경로가 아닌 여러 경로를 통하여 전송되어 패킷들의 순서가 바뀌었을 때에는 재정렬 작업을 수행하여 상위 계층(Application)에서 사용하는데 문제가 없도록 한다.

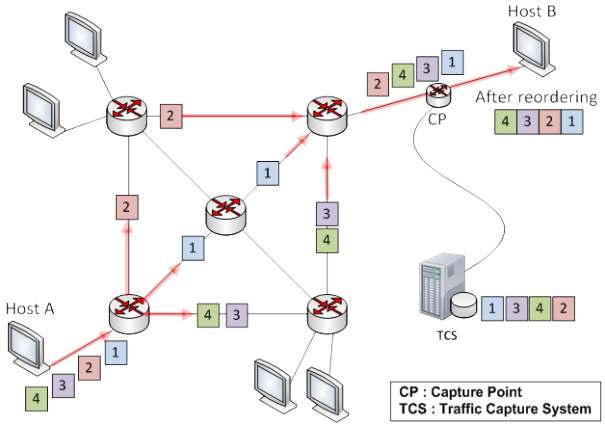

Fig. 3. Packet Ouf-of-order problem

그림 3 은 Host A 에서 보낸 패킷들이 여러 경로를 통해 전송될 경우 트래픽 수집지점(CP)에서 패킷의 순서가 맞지 않게 저장되는 것을 나타내는 그림이다. Host A 가 보낸 패킷은 여러 경로를 통해 전송되면서 원래의 순서가 아닌 다른 순서로 전달되었다. 이 때 Host B 는 자신의 전송계층에서 재정렬 작업을 하여 상위 계층에게 전달하기 때문에 Host B 는 데이터의 신뢰성을 보장받는다.

반면에 수집지점(CP)에서는 수집되는 패킷들에 대해 별도의 재정렬 작업을 하지 않으므로 트래픽 수집 시스템(TCS)에는 원래의 패킷 순서가 아닌 다른 순서로 저장되는 경우가 생긴다. 따라서 수집지점에서 저장되는 통계 정보는 원래의 통계정보가 일치하지 않으므로 통계 정보를 사용하여 해당 응용을 정확하게 탐지할 수 없다.

표 1 은 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 sequence 값을 비교한다. P(n)의 sequence 값이 P(k)의 sequence 값보다 크면 순서가 바뀐 패킷을 탐지하고 문제를 해결한다. (줄 6) P(n)과 방향이 같으면서 P(n)의 sequence 값보다 작은 값을 갖는 P(i)를 찾는다. (줄 7) 이렇게 찾은 P(i)와 방향이 같으면서 P(i)의 sequence 값보다 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).len 을 만족하는 패킷 P(m)이 존재하면 P(n)은 P(m)의 앞에 위치시킨다. (줄 12) 또는 P(m).ack < P(n).seq + P(n).len 을 만족하는 패킷 P(m)이 존재하면 (줄 13) P(n)은 P(m)의 뒤에 위치시킨다.

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 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).sez)    // 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 2 는 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 sequence 값과 P(n)의 sequence 값을 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 sequence 값과 패킷의 페이로드 길이를 더한 값이 P(n)의 sequence 값과 같이 않을 경우에는 P(n)이 이전에 재패킷화 되어 재전송된 패킷과 관계하는 패킷이라 간주하여 삭제된다.

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 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;
```

4. 실 험

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	-

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

본 장에서는 3 장에서 제안한 패킷 역전 문제, 패킷 중복 문제를 해결하는 알고리즘의 성능을 평가하기 위해 두 가지 실험을 하였다. 첫 번째 실험에서는 제안한 알고리즘을 실제 트래픽에 적용하여 문제의 해결 정도를 살펴 성능을 평가한다. 두 번째 실험에서는 이상 동작의 처리 후, 처리 전의 결과와 비교하여 성능을 평가한다.

표 3 은 본 논문에서 제안한 알고리즘을 실험에 사용한 트래픽 데이터에 적용한 결과이다. 패킷 역전 문제는 플로우 기준 1.33%, 패킷 기준 0.0125%, 바이트 기준 0.0185 를 차지하였다. 패킷 중복 문제는 플로우 기준 1.33%, 패킷 기준 1.93%, 바이트 기준 1.32%를 차지하였다. 재패킷과 문제는 플로우 기준 0.12%, 패킷 기준 0.0032%, 바이트 기준

0.57%를 차지하였다. 또한 패킷 역전 문제와 재패킷화 문제가 동시에 일어난 경우는 플로우 기준 0.08%를 차지하였다.

표 4 는 본 논문에서 제안한 방법론을 적용하기 전, 후의 응용 분석량을 비교한 표이다. 대부분의 응용들은 알고리즘 적용 후 플로우, 패킷, 바이트 별 각각 분석량이 상승한 것을 확인할 수 있었다. uTorrent 의 경우 플로우의 분석량은 하락하였으나 패킷, 바이트의 분석량은 상승하였다. 문제 해결 전에는 분석되었지만 해결 후에 분석되지 않은 플로우들을 분석한 결과 해당 플로우들은 재전송 패킷이 2~3 개로만 이루어진 플로우였다. 패킷 중복 문제 해결 후에는 이러한 재전송 패킷들은 삭제되고 해당 플로우 또한 분석되지 않았다. 이러한 이유로 플로우 분석량이 하락했음에도 불구하고 패킷과 바이트의 분석량은 대폭 증가하였다. 이러한 결과는 트래픽 망에 큰 부담을 주는 heavy 플로우를 더 정확히 분석한 것을 의미한다.

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

5. 결론 및 향후 연구

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.

전체 플로우 기준 총 분석량은 3% 감소하였지만 전체 패킷, 바이트 기준 4%의 분석률이 증가하였다. 이것은 본 논문에서 제안한 알고리즘을 적용하여 플로우 내의 패킷 역전, 중복 문제를 해결함으로써 실제 트래픽 망에 부담을 줄 수 있는 heavy 플로우의 분석량이 증가하였음을 보였다.

향후 연구로는 본 논문의 실험에서 분석률이 하락한 응용에 대한 원인에 대한 다양하고 구체적인 연구를 진행하고 면밀히 조사하여 분석률 하락의 원인을 밝히고 분석률을 향상시키기 위해 연구할 계획이다.

6. 참고 문헌

[1] L.Bernaille, R. Teixeira, and K. Salamatian, "Early Application Identification," In: CoNext 2006. Conference on Future Networking Technologies. 2006.

[2] Young-Tae Han and Hong-Shik Park, "Game Traffic Classification Using Statistical Characteristics at the Transport Layer," ETRI Journal, Vol.32, No.1, Feb., 2010, pp.22-32.

[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 Transactions on 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," Journal of 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," Information Systems Frontiers, vol. 12, pp. 149-156, April 2010.

[6] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in Proc. of ACM CoNEXT conference, 2006.

[7] 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.