

Wireshark를 이용한 Application Layer Protocol 트래픽 분석

제출자: 201421093 김민형

목차

1. 관찰 목표 및 예상 결과
2. 관찰 방법
3. Traffic 분석
4. 새로 알게 된 지식
5. 관찰 후 소감

1. 관찰 목표 및 예상 결과

필자는 영화나 예능을 볼 때는 항상 토렌트를 사용한다. 그래서 P2P 파트를 수업할 때 막막한 단어들 사이 낯익은 단어를 들으니 매우 반가웠다. 필자가 평소에도 사용하는 네트워크를 주제로 과제를 진행해야만 흥미를 갖고 할 수 있을것 같아 이번 과제의 주제로 선택하였다.

현재의 토렌트는 bitTorrent 프로토콜을 사용한다. 지금의 P2P는 많은 모델을 거쳐 역사를 가지고 발전하여 현재의 P2P 형식인 토렌트가 있게 되었다. 토렌트를 사용할 때는 토렌트 파일을 다운로드하고 해당 토렌트 파일을 열면 파일 다운로드와 업로드가 동시에 이루어진다. 하지만 종종 P2P 참가자가 없는 파일은 다운로드가 진행되지 않아 해당 파일을 삭제하고 다른 토렌트 파일을 찾아서 다운로드 해야 한다. 이번 과제를 통해 토렌트를 사용하며 다른 peer들 그리고 피어 하나를 집중적으로 선택 분석하여 어떻게 상호작용을 하는지 알아볼 예정이다.

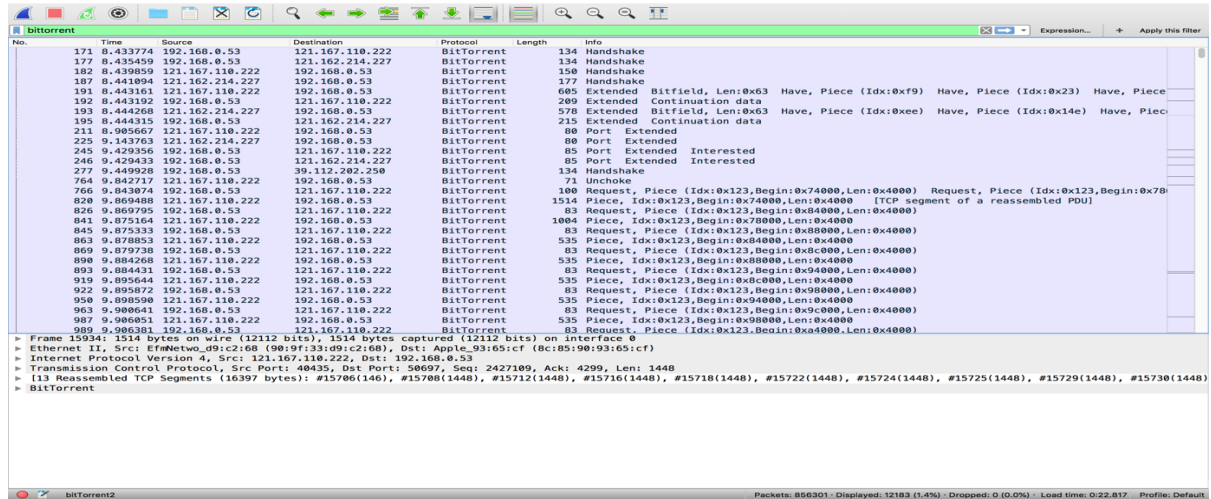
예상되는 결과는 분산 파일 시스템에 참가하는 순간부터 피어들과 접촉을 시도 할 것이고 접촉에서 생기는 request, response가 관찰될 것이다. 그 이후부터는 쪼개진 데이터를 꾸준히 수신할 것이다. 만약 transport layer protocol로 UDP를 사용한다면 packet loss와 동시에 재전송이 일어나 중간 중간 누락된 패킷이 존재할 것이다. 만일 TCP를 사용한다면 packet loss는 일어나지 않을 것이고 패킷이 온전히 필자의 컴퓨터로 들어올 것이다. 필자의 개인적인 생각으로는 파일의 손실이 일어나지 않는 TCP를 사용하는 것이 옳다고 생각한다.

2. 관찰 방법

네트워크는 아주대학교 기숙사 용지관의 Wi-Fi를 사용하여 진행하였다. 유튜브의 'Top 10 wireshark filters' 동영상을 참고하여 과제에 필요한 filter를 선별하여 WireShark에 적용할 생각이다. WireShark를 통해 현재 P2P 형식 네트워크를 대표하는 uTorrent를 이용하여 분산 해시 테이블 (Distributed Hash Table, DHT)을 이용하는 bitTorrent 프로토콜을 통해 운반되는 패킷을 심층적으로 분석한다. 또한, ip.addr 필터를 이용하여 그 많은 피어 중 하나의 IP를 선택하여 일대일 관계에서의 상호작용을 분석한다.

3. Traffic 분석

uTorrent를 이용하여 784.6MB에 해당하는 동영상 파일을 다운 받은 후 WireShark로 'bittorrent' 프로토콜만 볼 수 있게 필터링 하였다.

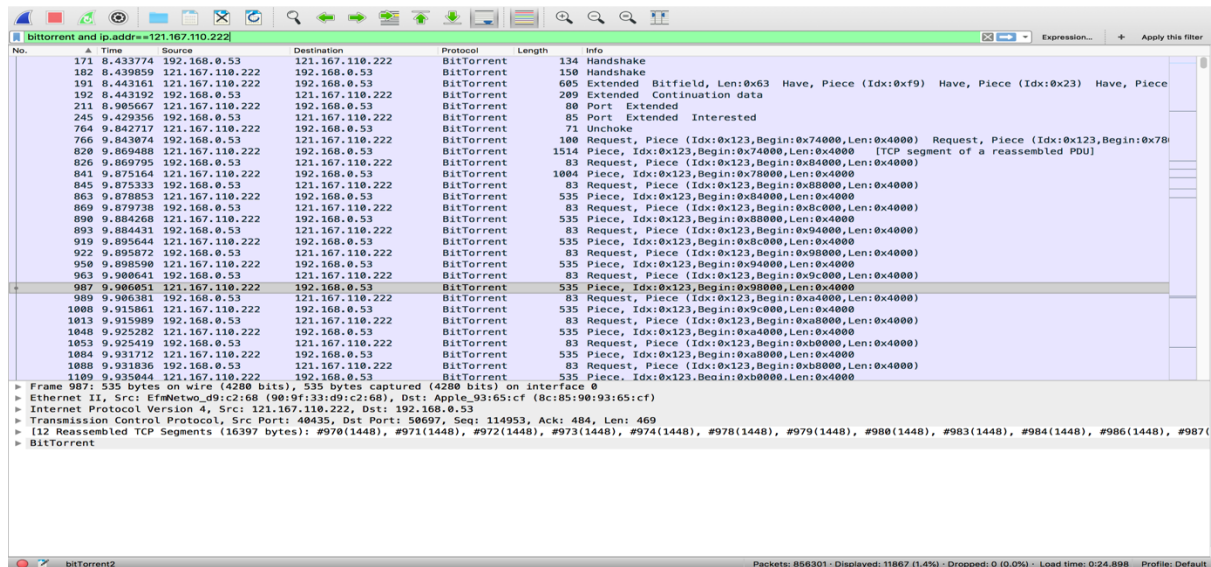


The screenshot shows the Wireshark interface with the 'bittorrent' filter applied to the packet list. The packet list displays various bittorrent protocols such as Handshake, Extended Bitfield, and Request. The packet details pane shows the selected packet's structure, including fields like 'Request', 'Piece', and 'Info'. The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
171	0.433774	192.168.0.53	121.167.110.222	BitTorrent	134	Handshake
177	0.435459	192.168.0.53	121.162.214.227	BitTorrent	134	Handshake
182	0.438059	121.167.110.222	192.168.0.53	BitTorrent	158	Handshake
187	0.441094	121.162.214.227	192.168.0.53	BitTorrent	177	Handshake
191	0.443161	121.167.110.222	192.168.0.53	BitTorrent	685	Extended Bitfield, Len:0x63 Have, Piece (Idx:0xf9) Have, Piece (Idx:0x23) Have, Piece
192	0.443192	192.168.0.53	121.167.110.222	BitTorrent	209	Extended Continuation data
193	0.444268	121.162.214.227	192.168.0.53	BitTorrent	578	Extended Bitfield, Len:0x63 Have, Piece (Idx:0xee) Have, Piece (Idx:0x14e) Have, Piece
195	0.444315	192.168.0.53	121.162.214.227	BitTorrent	215	Extended Continuation data
211	0.905667	121.167.110.222	192.168.0.53	BitTorrent	80	Port Extended
225	0.143763	121.162.214.227	192.168.0.53	BitTorrent	80	Port Extended
245	0.420356	192.168.0.53	121.167.110.222	BitTorrent	85	Port Extended Interested
246	0.429433	192.168.0.53	121.162.214.227	BitTorrent	85	Port Extended Interested
277	0.449928	192.168.0.53	39.112.202.258	BitTorrent	134	Handshake
290	0.442717	121.167.110.222	192.168.0.53	BitTorrent	71	Unchoke
766	0.843074	192.168.0.53	121.167.110.222	BitTorrent	100	Request, Piece (Idx:0x123, Begin:0x74000, Len:0x4000) Request, Piece (Idx:0x123, Begin:0x78000, Len:0x4000) [TCP segment of a reassembled PDU]
826	0.869795	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x8000, Len:0x4000)
841	0.875164	121.167.110.222	192.168.0.53	BitTorrent	1004	Piece, Idx:0x123, Begin:0x78000, Len:0x4000
845	0.875333	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x8000, Len:0x4000)
863	0.878853	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x84000, Len:0x4000
869	0.879738	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x8000, Len:0x4000)
890	0.884268	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x88000, Len:0x4000
893	0.884431	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x94000, Len:0x4000)
919	0.895644	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0xc000, Len:0x4000
922	0.895872	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x98000, Len:0x4000)
950	0.898590	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x94000, Len:0x4000
963	0.900641	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x9c000, Len:0x4000)
987	0.906051	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x98000, Len:0x4000
989	0.906381	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0xa4000, Len:0x4000)
Frame 15934: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0						
Ethernet II, Src: EfMetwo-d9:c2:68 (90:9f:33:d9:c2:68), Dst: Apple-93:65:cf (8c:85:90:93:65:cf)						
Internet Protocol Version 4, Src: 121.167.110.222, Dst: 192.168.0.53						
Transmission Control Protocol, Src Port: 48435, Dst Port: 58697, Seq: 2427109, Ack: 4299, Len: 1448						
[13 Reassembled TCP Segments (16397 bytes): #15706(146), #15708(1448), #15712(1448), #15716(1448), #15718(1448), #15722(1448), #15724(1448), #15725(1448), #15729(1448), #15730(1448)						
BitTorrent						

<사진 1> bittorrent 프로토콜 필터를 적용

첫 번째 줄(No.171)과 두 번째 줄(No.177)을 통하여 두 개의 다른 IP주소와 handshake를 한 것을 보아 해당하는 IP들과 접촉한 192.168.0.53 는 필자의 로컬 IP 주소임을 알 수 있다. 여기서 handshake는 말 그대로 약속이다. 약속은 사람과 사람 사이의 인사 혹은 '나는 당신을 맞이한다'로 표현될 수 있다. 마찬가지로, 컴퓨터 네트워크에서의 handshake는 상대방이 나와 파일을 공유할 수 있는지를 패킷을 보냄으로써 확인하는 단계이다. 여기서 내가 보낸 패킷을 상대방이 받고 파일을 공유하겠다고 결정하면 나에게 handshake 패킷을 다시 보내올 것이다. 그렇다면 handshake 패킷을 서로 주고받은 이후의 과정은 어떠한가? 피어 하나와의 상호작용을 쉽게 보고자 필자는 ip.addr 필터를 추가하였다.



The screenshot shows the Wireshark interface with the 'bittorrent and ip.addr==121.167.110.222' filter applied. The packet list displays various bittorrent protocols such as Handshake, Extended Bitfield, and Request. The packet details pane shows the selected packet's structure, including fields like 'Request', 'Piece', and 'Info'. The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
171	0.433774	192.168.0.53	121.167.110.222	BitTorrent	134	Handshake
182	0.438059	121.167.110.222	192.168.0.53	BitTorrent	158	Handshake
191	0.443161	121.167.110.222	192.168.0.53	BitTorrent	685	Extended Bitfield, Len:0x63 Have, Piece (Idx:0xf9) Have, Piece (Idx:0x23) Have, Piece
192	0.443192	192.168.0.53	121.167.110.222	BitTorrent	209	Extended Continuation data
211	0.905667	121.167.110.222	192.168.0.53	BitTorrent	80	Port Extended
245	0.429356	192.168.0.53	121.167.110.222	BitTorrent	85	Port Extended Interested
764	0.842717	121.167.110.222	192.168.0.53	BitTorrent	71	Unchoke
766	0.843074	192.168.0.53	121.167.110.222	BitTorrent	100	Request, Piece (Idx:0x123, Begin:0x74000, Len:0x4000) Request, Piece (Idx:0x123, Begin:0x78000, Len:0x4000) [TCP segment of a reassembled PDU]
826	0.869795	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x8000, Len:0x4000)
841	0.875164	121.167.110.222	192.168.0.53	BitTorrent	1004	Piece, Idx:0x123, Begin:0x78000, Len:0x4000
845	0.875333	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x8000, Len:0x4000)
863	0.878853	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x84000, Len:0x4000
869	0.879738	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x8000, Len:0x4000)
890	0.884268	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x88000, Len:0x4000
893	0.884431	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x94000, Len:0x4000)
919	0.895644	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0xc000, Len:0x4000
922	0.895872	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x98000, Len:0x4000)
950	0.898590	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x94000, Len:0x4000
963	0.900641	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0x9c000, Len:0x4000)
987	0.906051	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x98000, Len:0x4000
989	0.906381	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0xa4000, Len:0x4000)
1008	0.915061	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0x9c000, Len:0x4000
1013	0.915080	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0xa0000, Len:0x4000)
1048	0.925282	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0xa4000, Len:0x4000
1053	0.925419	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0xb0000, Len:0x4000)
1084	0.931712	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0xb8000, Len:0x4000
1088	0.931836	192.168.0.53	121.167.110.222	BitTorrent	83	Request, Piece (Idx:0x123, Begin:0xb0000, Len:0x4000)
1109	0.935044	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x123, Begin:0xb8000, Len:0x4000
Frame 987: 535 bytes on wire (4280 bits), 535 bytes captured (4280 bits) on interface 0						
Ethernet II, Src: EfMetwo-d9:c2:68 (90:9f:33:d9:c2:68), Dst: Apple-93:65:cf (8c:85:90:93:65:cf)						
Internet Protocol Version 4, Src: 121.167.110.222, Dst: 192.168.0.53						
Transmission Control Protocol, Src Port: 48435, Dst Port: 58697, Seq: 114953, Ack: 484, Len: 469						
[12 Reassembled TCP Segments (16397 bytes): #970(1448), #971(1448), #972(1448), #973(1448), #974(1448), #978(1448), #979(1448), #980(1448), #983(1448), #984(1448), #986(1448), #987(1448)						
BitTorrent						

<사진 2> ip.addr 필터를 추가

우선 두 번째 줄(No.182)을 보면 handshake 가 되돌아온 것을 볼 수 있다. 이것은 해당 peer 가 데이터 공유 의사를 밝힌 것이다. 그 이후 패킷에는 piece 라는 용어가 많이 보인다. 그리고 이것을 서로 주고받는 형태를 띠고 있다. Piece 는 분산된 각 파일의 작은 정보 덩어리(chunk)다. bitTorrent 프로토콜에서 이렇게 일정한 크기로 쪼개진 조각의 piece 는 고유의 idx 를 가진다. 각 피어는 보유하고 있는 piece 들의 idx 값을 메시지 속에 포함해 보냄으로써 현재 본인이 어떤 piece 들을 보유하고 있는지 알린다. 사진에서 보면 계속해서 필자의 IP 주소로부터 peer 의 IP 주소로 piece 를 request 하는 것을 볼 수 있다. 이는 현재 내가 필요로 하는 piece 를 상대방이 보유하고 있는지 확인하기 위함이다. 여기서 만일 상대 피어가 해당 piece 를 가지고 있다면 Have piece 라는 메시지를 보내온다.

1590	10.0521...	121.167.110.222	192.168.0.53	BitTorrent	535	Piece, Idx:0x43,Begin:0xc000,Len:0x4000
1592	10.0524...	192.168.0.53	121.167.110.222	BitTorrent	92	Have, Piece (Idx:0x123) Request, Piece (Idx:0x43,Begin:0x18000,Len:0x4000)

<사진 3> Have, Piece 메시지

만일 상대 피어가 나와 동등한 다운로드가 아닌 다운로드를 이미 마친 seeder 라면 have piece 메시지를 보내오지 않고 자신이 seeder 라고 밝히면서 모든 piece 를 보유하고 있다고 알린다.

그 다음 과정을 살펴보자.

No.	Time	Source	Destination	Protocol	Length	Info
122172	27.4282...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122173	27.4283...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122176	27.4285...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122178	27.4292...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122179	27.4292...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122180	27.4292...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122181	27.4292...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122642	27.4780...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122643	27.4780...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122644	27.4781...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122646	27.4783...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122648	27.4785...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122649	27.4785...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122651	27.4786...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122652	27.4788...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122653	27.4788...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
122656	27.4791...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123000	27.5295...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123001	27.5295...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123002	27.5295...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123003	27.5295...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123008	27.5307...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123009	27.5307...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123010	27.5307...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123011	27.5307...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123012	27.5308...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123021	27.5314...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
123398	27.5784...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data

Frame 7337: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
 Ethernet II, Src: EfmNetwo_d9:c2:68 (98:9f:33:d9:c2:68), Dst: Apple_93:65:cf (8c:85:90:93:65:cf)
 Internet Protocol Version 4, Src: 121.167.110.222, Dst: 192.168.0.53
 Transmission Control Protocol, Src Port: 40435, Dst Port: 50697, Seq: 1673029, Ack: 3192, Len: 1448
 [13 Reassembled TCP Segments (16397 bytes): #7268(328), #7271(1448), #7272(1448), #7274(1448), #7275(1448), #7276(1448), #7278(1448), #7331(1448), #7332(1448), #7333(1448), #7335(1448)]
 BitTorrent

<사진 4> Continuation data

사진을 보면 121.167.110.222 의 IP 주소를 가진 피어가 필자의 컴퓨터로 일정한 길이(1,514byte)의 패킷을 수차례 반복하여 보내는 것을 볼 수 있다. 1514 의 수치가 뜻하는 것을 설명하기 위해 사진을 하나 더 추가하겠다.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
▼ Frame	100.0	11867	100.0	17718174	1750 k	0	0	0
▼ Ethernet	100.0	11867	0.9	166138	16 k	0	0	0
▼ Internet Protocol Version 4	100.0	11867	1.3	237340	23 k	0	0	0
▼ Transmission Control Protocol	100.0	11867	97.7	17314696	1710 k	0	0	0
▼ BitTorrent	123.0	14592	145.7	25821016	2551 k	11735	24139333	2385 k
Malformed Packet	1.1	134	0.0	0	0	132	0	0

Display filter: bittorrent and ip.addr==121.167.110.222

Help Copy Close

<사진 5> Protocol Hierarchy

위 그림은 프로토콜 계층에서 각 계층의 종류와 처리한 패킷의 개수, 비율 등을 볼 수 있다. Transport layer 는 TCP 를 사용하고 그 하위 계층인 Network layer 는 Internet Protocol Version 4 를 이용한다. 그리고 그 하위 계층인 Data Link layer 는 Ethernet 을 사용하는 것을 알 수 있다. 통상 Ethernet 프레임에서는 패킷의 헤더가 14byte 를 차지하고, 그 상위 계층인 Network layer 에서는 IP 헤더가 20byte 를 차지한다. 그리고 그 상위 계층인 Transport layer 에서는 TCP 헤더가 20byte 를 차지한다. 그리고 나머지 1,460byte 에는 bitTorrent 의 정보가 들어있다.

Layer 1: Physical layer - the actual hardware

Layer 2: Data Link layer (frames) - data transfer method (802x ethernet - 14-byte header). MAC and LLC

Layer 3: Network layer (packets) - IP network protocol (routes messages using best available path). 20-byte IP header

Layer 4: Transport layer - TCP, UDP (deals with packet sequence, error handling). ~20-byte TCP header

Layer 5: Session layer - User interface with the network, controls data sessions, security, name lookup.

Layer 6: Presentation layer - prepares data, translates between network and user applications.

Layer 7: Application layer - provides the ability for user applications to interact with the network.

<참고 자료> 각 계층의 헤더가 차지하는 byte 수

사진 5 에서 보면 BitTorrent 프로토콜이 처리한 패킷 수와 하위 계층들이 처리한 패킷 수가 다르다는 걸 볼 수 있다. 그 차이는 2,725 개다. 이 정보를 바탕으로 필자는 2,725 개의 패킷이 하위 계층의 프로토콜로 전달하는 과정에서 일련의 오류가 발생했다고 짐작하였다.

그런데 패킷을 아래로 내리다 보면 특이한 점을 볼 수 있다. 간혹 검은색으로 칠해진 패킷을 볼 수 있다. 이 검은색은 WireShark 에서 packet loss, 재전송, 세그먼트의 문제 등을 뜻한다. 그렇다면 transport layer protocol 로 reliable 한 운반을 책임지는 TCP 를 기반 bitTorrent 에서 검은색 패킷은 무슨 뜻일까?

7693	12.1792...	121.167.110.222	192.168.0.53	BitTorrent	1186	[TCP Window Full] Piece, Idx:0x0, Begin:0x64000, Len:0x4000	[TCP segment of a reassembled PDU]
------	------------	-----------------	--------------	------------	------	---	------------------------------------

<사진 6> 검정색을 띄는 패킷

메시지 정보를 보면 [TCP Window Full]이라고 적혀있다. 이 메시지는 수신자 측에서 Read()를 수행하지 않아 버퍼에 데이터가 쌓이게 되면 window 가 가득 차게 된다. 그러므로 송신자에게 이 사실을 알려 더는 piece 를 전송하지 못하도록 한다. 즉 현재 수신자의 버퍼에는 읽지 않은 데이터가 가득함을 의미한다. 이런 현상은 송신자가 수신자의 window 사이즈 보다 더 많은 데이터를 보낼 때 발생한다. 이 외에도 다른 종류의 검정색 패킷들이 있다. 아래사진과 같이 재전송 이슈 또는 이전 세그먼트를 capture 하지 못했을 때도 검은색으로 표시한다.

127016	27.9803...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
127083	27.9871...	121.167.110.222	192.168.0.53	BitTorrent	1514	[TCP Fast Retransmission] Continuation data
127488	28.0323...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data

<사진 7> 검정색을 띄는 패킷2

126628	27.9350...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data
126670	27.9387...	121.167.110.222	192.168.0.53	BitTorrent	1514	[TCP Previous segment not captured] Continuation data
127010	27.9800...	121.167.110.222	192.168.0.53	BitTorrent	1514	Continuation data

<사진 8> 검정색을 띄는 패킷3

필자는 bitTorrent 프로토콜에서 Transport layer 프로토콜로 가는 과정에서 일련의 오류가 발생한 패킷의 수가 2,725개로 짐작했었다. 그렇다면 이 검은색을 띄는 패킷의 개수와 일치할 것인지 확인해보겠다. 우선 검은색 패킷을 모아 보기 위해 해당 패킷의 상세 정보를 살펴보았다.

[Coloring Rule Name: Bad TCP]

[Coloring Rule String: tcp.analysis.flags && !tcp.analysis.window_update]

<사진 9> 검정색 패킷의 Coloring Rule 정보

모든 검은색 패킷은 tcp.analysis.flags and !tcp.analysis.window_update 이라는 규칙하에 분류되고 있었다. 이제 "tcp.analysis.flags and !tcp.analysis.window_update and ip.addr==121.167.110.222" 를 필터에 입력하여 검정색 패킷만을 따로 모아서 볼 것이다.

The image shows a Wireshark packet capture interface. The top filter bar contains the expression: `tcp.analysis.flags and ip.addr==121.167.110.222 and !tcp.analysis.window_update`. Below the filter bar, a list of packets is displayed, with many packets colored black. The packet details pane shows the structure of a selected packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol fields. The packet list shows various TCP segments, many of which are colored black. The packet details pane shows the structure of a selected packet, including Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol fields.

<사진 10> 검정색 패킷 필터링

관찰 결과 아쉽게도 수치는 일치하지 않았다. 화면 아래에 1,695개의 패킷이 보여지고 있다고 하며 이는 필자가 추측한 숫자 2,725와 약 천 개의 오차가 발생했다. 아마 필자가 놓친 몇 가지의 패킷들이 더 있을 것으로 판단된다. 이에 대해서는 더 많은 공부가 필요할 것 같다.

4. 새로 알게 된 지식

과제를 수행하기 전에 Transport layer protocol에서 UDP와 TCP의 차이를 공부하고 분석을 시작했었다. UDP는 불확실한 전달을 수행하고 TCP는 확실한 전달을 책임진다고 알고 있었다. 그러나 트래픽 분석을 하는 도중 TCP 프로토콜을 사용하는 bitTorrent 프로토콜도 검은색 패킷이 발생하는 것에 대해 혼란을 겪었다. 패킷 상세 정보와 인터넷 검색을 통하여 이것은 packet loss에 해당하는 문제는 아니라는 것을 알았다. Buffer의 용량 초과 그리고 재전송과 같은 이슈들이 발생하였는데 필자는 분석 이전에는 이러한 이슈들도 확실한 전달을 책임지는 TCP 프로토콜에서는 발생하지 않을 것으로 생각했기 때문에 새로운 걸 알게 되었다. 그리고 P2P 환경에서 peer들과 어떠한 방식으로 상호작용하는지를 알 수 있었고 수업에서 배우지 않은 많은 프로토콜이 존재한다는 것을 알게 됐다.

5. 관찰 후 소감

과제를 처음 시작할 때는 고화질 영상을 다운로드 했다. 그렇게 하니 패킷의 수가 너무 많아 필터를 적용할 때마다 매우 많은 시간이 로딩하는 데에 소요됐다. 그래서 중간에 다시 저용량의 동영상을 다운로드 하여 분석을 재진행하였다. P2P 분석이다 보니 트래픽을 캡처하였을 때 너무 많은 IP 주소들이 분석에 방해되었다. 그래서 하나의 피어만 붙잡고 트래킹한 것이 분석에 매우 도움이 되었다. 그리고 직접 많은 검색을 하며 분석을 진행하니 알고자 하지 않았던 지식도 덩으로 함께 얻을 수 있어 좋은 시간이었다고 생각한다.