컴퓨터 네트워크 과제: Chapter 3 Homework

미디어학과 박성범

1. 다음 문제에 답하시오. (Wireshark 연계 문제임)

(1) TCP SYN segment 전송시 sequence number는 어떻게 정해지는지를 설명하시오.

SYN segment를 전송할 때 sequence number는 0에서 4,294,967,295 사이의 숫자 중 랜덤으로 정해진다.

(2) TCP FIN segment 전송시 sequence number는 어떻게 정해지는지를 설명하시오.

FIN segment를 전송할 때는 직전에 상대방(client 또는 server)로부터 받은 ACK number를 sequence number로 한다.

(3) Wireshark을 사용하여 (1)과 (2)의 내용을 확인한 결과를 나타내시오. (Wireshark 사용 환경과 시나리오를 적고, 화면 캡쳐한 내용을 설명하시오.)

client(192.168.0.6)에서 ajou.ac.kr(202.30.0.19)에 접속해 TCP connection을 구축한다.

366	7.503091	192.168.0.6	202.30.0.19	TCP	66	57938 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
368	7.516772	202.30.0.19	192.168.0.6	TCP	66	80 → 57938 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460 SACK_PERM=1 WS=1
369	7.516920	192.168.0.6	202.30.0.19	TCP	54	57938 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0

먼저 client에서 SYN segment를 보냈다. 이어서 server에서 SYN, ACK 응답이 돌아왔고, client도 ACK segment를 전송했다. 이때 Ack number은 앞선 SYN, ACK의 Seq number 0에 1을 더한 값이다. ISN이 무작위로 정해질 것이라는 생각과는 다르게 매번 접속을 시도할 때마다 ISN이 0으로 설정됐는데, Wireshark가 relative sequence number를 보여주기 때문이었다.[1]

962	114.077800	192.168.0.6	202.30.0.19	TCP	66	52736 → 80 [SYN] Seq=1835063176 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
963	114.098480	202.30.0.19	192.168.0.6	TCP	66	80 → 52736 [SYN, ACK] Seq=2284714158 Ack=1835063177 Win=32768 Len=0 MSS=1460 SACK_PERM=1 WS=1
964	114.098610	192.168.0.6	202.30.0.19	TCP	54	52736 → 80 [ACK] Seq=1835063177 Ack=2284714159 Win=65536 Len=0

Wireshark의 설정을 바꾸니 Seq number가 랜덤으로 설정되는 것을 볼 수 있었다.

---

[1] PacketLife, "Understanding TCP Sequence and Acknowledgment Numbers", 2010.

281 1.805600	192.168.0.6	202.30.0.19	TCP	54	58058 → 80 [FIN, ACK] Seq=1535 Ack=76465 Win=138496 Len=0
282 1.812341	192.168.0.6	202.30.0.19	TCP	66	58066 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACI
283 1.817786	192.168.0.6	202.30.0.19	TCP	66	58067 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACI
284 1.820403	202.30.0.19	192.168.0.6	TCP	54	80 → 58058 [ACK] Seq=76465 Ack=1535 Win=65535 Len=0
285 1.820527	202.30.0.19	192.168.0.6	TCP	54	80 → 58058 [ACK] Seq=76465 Ack=1536 Win=65535 Len=0
286 1.821434	202.30.0.19	192.168.0.6	TCP	54	80 → 58058 [FIN, ACK] Seq=76465 Ack=1536 Win=65535 Len=0
287 1.821619	192.168.0.6	202.30.0.19	TCP	54	58058 → 80 [ACK] Seq=1536 Ack=76466 Win=138496 Len=0

연결을 끊기 위해 client에서 server로 FIN segment를 전송했다. 이때 Seq는 직전에 server로부터 받은 Ack number다. 이어서 서버에서 ACK segment가 왔고, Ack number는 client로부터 받은 FIN segment의 Seq number 1535에 1을 더한 1536이었다. 그리고 server가 FIN segment를 보내고, client는 ACK로 응답하며 TCP connection을 닫았다.

중간에 TCP connection을 구축하는 패킷이 함께 잡혔는데, 새 리소스를 요청할 때마다 TCP connection을 구축하고 닫는 과정이 반복되는 것을 볼 수 있었다.

2. 다음 문제에 답하시오.

(1) AIMD 방법을 psuedo code로 작성하고, 이를 설명하시오.

```
if timeout:
  cwnd /= 2             # timeout이 일어나면 cwnd를 절반으로 줄인다.
elif ACK_arrived:
  cwnd += 1 / cwnd      # ACK가 도착하면 기존 cwnd에 (1 / cwnd)를 누적한다.
```

(2) Slow Start 방법을 psuedo code로 작성하고, 이를 설명하시오.

```
cwnd = MSS      # 초기 cwnd를 1MSS로 설정한다.

if timeout:
  cwnd = 1      # timeout이 일어나면 cwnd를 1로 만든다.
elif ACK_arrived:
  cwnd += MSS   # ACK가 도착하면 cwnd를 1MSS 증가시킨다.
```

(3) AIMD와 비교한 Slow Start 방법의 장점을 적으시오.

AIMD는 연결 초기에 낮은 대역폭만을 사용해 통신 시간이 오래 걸리게 된다. 또한 네트워크가 혼잡해지는 상황에 미리 대응할 수 없다. 반면 Slow Start는 네트워크 상황을 빠르게 수집할 수 있고, 혼잡을 예측할 수 있다.

(4) AIMD와 비교한 Slow Start 방법의 단점을 적으시오.

Slow Start는 AIMD에 비해 네트워크 상황을 빠르게 수집할 수 있지만, 그만큼 혼잡을 유발할 위험성도 커진다. 더불어, 혼잡한 상황에서 timeout까지 대기하는 공백이 크다.

3. Congestion Avoidance (Tahoe) 방법과 관련한 다음 물음에 답하시오.

(1) Congestion Avoidance 방법을 psuedo code로 작성하고, 이를 설명하시오.

```
if timeout ¦¦ duplicated_ACK >= 3:
  GBN_ARQ()              # 패킷이 손실되면 GBN ARQ를 실행하고
  ssthresh = cwnd / 2    # SS threshold를 절반으로 줄이고
  cwnd = 1               # cwnd를 1로 설정한다. (Slow Start를 다시 시작)
elif ACK_arrived:
  if cwnd < ssthresh:
    cwnd += 1            # cwnd가 SS 한계점보다 작으면 cwnd를 1 증가시킨다. (SS)
  else:
    cwnd += 1 / cwnd     # 한계를 초과하면 기존 cwnd에 (1 / cwnd)를 누적한다. (CA)
```

(2) CA 방법이 필요하게 된 이유를 설명하시오.

네트워크가 혼잡해 지연이 커지면, 같은 데이터를 반복해서 보내게 되어 혼잡이 더욱 악화된다. 이를 해결하기 위해 혼잡 방지 알고리즘을 사용하며, Tahoe 방식에서 CA 상태에 진입하면 cwnd의 크기가 커지고, 이후 SS 진행을 보정할 수 있게 된다.

(3) CA 방법의 단점에 대하여 설명하시오.

초기 threshold의 값이 커 처음 SS에 진입할 때 패킷 손실이 발생한다. 또한 3개 이상의 중복된 ACK를 수신했을 때 SS 첫 단계부터 시작해 시간이 오래 걸린다.

4. TCP Reno version과 관련한 다음 물음에 답하시오.

(1) TCP Reno version의 방법을 psuedo code로 작성하고, 이를 설명하시오.

```
if timeout:
  GBN_ARQ()              # Timout이 발생하면 GBN ARQ를 실행하고
  ssthresh = cwnd / 2    # SS threshold를 절반으로 줄이고
  cwnd = 1               # cwnd를 1로 설정한다. (Slow Start로 진입)
elif duplicated_ACK >= 3:
  GBN_ARQ()              # ACK가 3개 중복되면 GBN ARQ를 실행하고
  ssthresh = cwnd / 2    # SS threshold를 절반으로 줄이고
  cwnd = cwnd / 2 + 3    # 바로 Fast Recovery & CA에 진입한다.
elif ACK_arrived:
  if cwnd < ssthresh:
    cwnd += 1            # cwnd가 SS 한계점보다 작으면 cwnd를 1 증가시킨다. (SS)
  else:
    cwnd += 1 / cwnd     # 한계를 초과하면 기존 cwnd에 (1 / cwnd)를 누적한다. (CA)
```

(2) TCP Reno version 방법이 필요하게 된 이유를 설명하시오.

TCP Tahoe version은 중복된 ACK를 수신한 뒤 SS부터 시작해 소요 시간이 오래 걸리는데, 이러한 단점을 개선한 것이 Reno version이다. TCP Reno version은 중복된 ACK를 3번 수신한 뒤 바로 Fast Recovery & CA에 진입한다.