# 가상 시작 알고리즘: 심 우주 통신에서의 타임아웃과 가상 ACK를 이용한 TCP 혼잡제어 알고리즘

오준석, 하태윤, Arooj Masood, Demeke Shumeye, 이철, 이동현, 조성래

중앙대학교

{jsoh, tyha, arooj, demeke, clee, dhlee}@uclab.re.kr, srcho@cau.ac.kr

# Virtual Start Algorithm: TCP Congestion Control Algorithm using Time-out and Virtual ACK in Deep Space Communication

Junsuk Oh, Taeyun Ha, Arooj Masood, Demeke Shumeye, Cheol Lee,

Donghyun Lee, Sungrae Cho

Chung-Ang Univ.

요 약

심 우주 통신(deep space communication)에서 높은 왕복시간(Round-Trip Time, RTT)과 전체 소요시간을 줄이는 것은 해당 분야의 중요한 문제로 여겨지고 있다. 본 논문에서는 가상 ACK를 이용한 가상 시작 알고리즘(Virtual Start Algorithm)을 바탕으로 통신의 평균적인 소요시간을 줄이는 새로운 TCP 혼잡제어 알고리즘을 제안한다.

## Ⅰ. 서 론

TCP에서의 혼잡제어는 네트워크의 혼잡으로 인한 붕괴 상황을 방지하기 위하여 다양한 혼잡제어 방법을 이용한다. 일반적으로 혼잡제어는 패킷(packet)을 송신하는 송신자가 ACK를 수신하는 과정을 바탕으로 네트워크의 수용량을 결정한다. 그러나 송신자는 처음부터 네트워크의 수용량을 알 수 없으며, 네트워크의 수용량은 지속적으로 변경되기 때문에 실제 환경에서는 일반적으로 느린 시작과 혼잡 회피, 빠른 회복을 이용한다.

앞서 언급한 혼잡제어 방법을 이용할 수 있는 일반적인 통신환경과 다르게 2,000,000km 이상의 거리를 의미하는 심 우주에서의 통신환경은 평균적으로 40분의 RTT를 가진다. 이와 같은 통신환경에서 통신지연, 패킷 손실 등에 의한 타임아웃이 발생할 경우 혼잡 창(Congestion Window, cwnd)의 크기는 기하급수적으로 감소하게 되며, 이로 인한 소요시간의 증가는 비효율적인 통신을 일으킨다.

본 논문에서는 TCP 혼잡제어 알고리즘의 느린 시작에서 타임아웃이 발생할 경우 가상 ACK를 바탕으로 소요시간을 효과적으로 줄이는 가상 시작 알고리즘을 제시하고, 이를 바탕으로 심 우주 환경에서 적용하기 어려운 일반적인 TCP 혼잡제어 알고리즘의 대안을 제시하고자 한다.

## Ⅱ. 본론

### Ⅱ-1. 가상 ACK와 가상 시작 알고리즘

일반적으로 느린 시작은 네트워크의 수용량을 결정하기 위해 cwnd의 크기를 1 MSS에서 시작하여 임계값에 도달할 때까지 2배씩 증가하는 방식으로 실행한다. 이와 같은 과정에서 네트워크의 통신지연으로 인한 타임아웃이 발생할 경우 cwnd의 크기는 1 MSS로 감소한 후 다시 느린 시작을 시작하며, 이는 추가적인 소요시간을 필요로 하게 된다.

앞서 언급한 느린 시작 알고리즘의 비효율성은 낮은 소요시간을 필요로 하는 심 우주 통신환경에서 부적합함을 시사한다. 이에 따라 본 논문에서는 통신지연으로 인한 타임아웃이 발생할 경우 실제 ACK가 수신되지 않

**Algorithm 1** Virtual Start Algorithm

```
1:  Initialize cwnd = 1 MSS, tempcwnd = 0 MSS, virtualcwnd = 0 MSS, dupACK-
    count = 0, vsthresh = 64, RTT = 2400s, MeasuredTime = 0s, timeoutsize = 3600s,
    timeoutcount = 0;
2:  while True do
3:      Transmit packets by cwnd size;
4:      while True do
5:          if dupACKcount == 3 then
6:              Call fast recovery algorithm;
7:          else if cwnd >= vsthresh then
8:              Call congestion avoidance algorithm;
9:          else
10:             tempcwnd = cwnd;
11:             while MeasuredTime <= timeoutsize do
12:                 if duplicateACK == True then
13:                     Case of duplicate ACK;
14:                     Case of 3 duplicate ACKs;
15:                 else if actualACK == True then
16:                     not ACKed packet = ACK;
17:                 else
18:                     Case of new ACK;
19:             if timeoutcount == 1 then
20:                 if virtualACK == True then
21:                     Case of timeout;
22:                 else
23:                     timeoutcount = 0;
24:             if timeoutcount == 0 then
25:                 virtualcwnd = tempcwnd;
26:             if timeout == True then
27:                 timeoutcount = timeoutcount + 1;
28:                 not ACKed packets = virtualACK;
29:                 Case of new ACKs;
30:             Transmit packets by cwnd size;
```

그림 1. 가상 시작 알고리즘(Virtual Start Algorithm)

았더라도 가상 ACK를 바탕으로 실제 ACK가 수신된 것으로 가정하여 cwnd의 크기 감소 없이 그대로 실행하며, 다음 타임아웃 이내에 실제 ACK가 수신된다면 아무 문제가 없는 것으로 간주한다. 그러나 기존과 같이 네트워크의 혼잡으로 인한 붕괴 상황을 방지할 필요성이 있음에 따라 다음 타임아웃 이내에 실제 ACK가 수신되지 않는다면 이를 통신지연이 아닌 패킷손실로 간주하여 타임아웃을 실행하게 된다. 결론적으로 해당 라운드에 송신한 패킷에 대한 ACK가 해당 라운드가 아닌 다음 라운드까

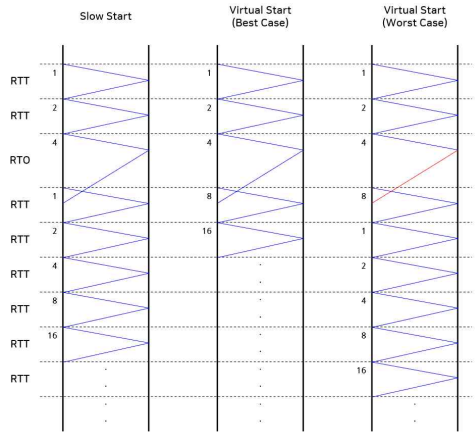


그림 2. 임계값이 16인 경우의 느린 시작과 가상 시작의 통신

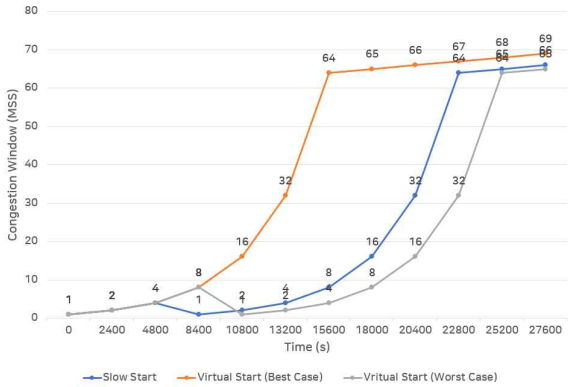


그림 3. 임계값이 64인 경우의 느린 시작과 가상 시작의 통신

지 수신될 수 있도록 하여 송신한 각각의 패킷에 대해 한 번의 기회를 더 부여하는 방식으로 실행하게 된다.

그림 1은 가상 ACK을 이용한 가상 시작 알고리즘을 제시한다. 해당 알고리즘은 5번째와 7번째 줄과 같이 3 duplicate ACKs, cwnd ≥ vsthresh가 발생할 경우 빠른 회복 알고리즘과 혼잡 회피 알고리즘을 실행하는 등 느린 시작에서의 타임아웃을 제외한 부분을 기존과 동일하게 설정한다. 24-25번째 줄은 타임아웃이 발생할 경우에 대비하여 해당 라운드의 cwnd의 크기를 보존하는 것을 의미하며, 26-29번째 줄은 타임아웃이 발생할 경우 실제 ACK를 수신한 것처럼 가상 ACK를 바탕으로 한 번의 기회를 더 부여하는 것을 의미한다. 이후 19-23번째 줄은 이전 라운드가 타임아웃이었다면 현재 라운드에서 이전 라운드의 실제 ACK가 수신되었는지를 확인하고 수신 여부에 따라 타임아웃을 실행하거나 이전 라운드의 타임아웃을 없던 것으로 간주하는 것을 의미한다. 타임아웃이 발생한 후 다음 라운드에서 빠른 회복 알고리즘을 실행하는 경우는 일반적으로 통신지연이 발생하더라도 이전 라운드의 패킷이 먼저 송신되었기 때문에 그에 대한 실제 ACK가 먼저 수신되어야 하므로, 3 duplicate ACKs가 이전 라운드의 실제 ACK보다 먼저 발생하는 경우는 없다고 가정한다.

### Ⅱ-2. 느린 시작과 가상 시작

cwnd = 4 즉, 세 번째 라운드에서 타임아웃이 발생한다고 가정할 때, 느린 시작과 가상 시작을 비교한 경우는 그림 2, 그림 3과 같다. 단, 그림 2와 그림 3의 각 라운드에서 패킷과 패킷 사이의 시간은 RTT에 비해 매우 작으므로 0에 수렴한다고 가정한다. 느린 시작이 cwnd의 크기를 1 MSS로 초기화하고 다시 시작하는 것에 비하여 최선의 경우 즉, 패킷손실이 아닌 통신지연의 경우 가상 시작은 cwnd의 크기를 8 MSS로 변경하고 재시작 없이 다음 라운드에서 실제 ACK를 수신하여 소요시간을 줄이는 것을 확인할 수 있다. 그러나 최악의 경우 즉, 패킷손실의 경우 가상 시작은 한 번의 기회를 더 부여하므로 느린 시작에 비하여 동일한 라운드에 도달하기까지 1 RTT가 더 소요되는 것을 확인할 수 있다.

$$Total\ Time\ of\ Slow\ Start = (N + K + 1) \times RTT + RTO \quad (1)$$

$$Total\ Time\ of\ Virtual\ Start = \begin{cases} N \times RTT + RTO, & in\ case\ of\ delay \\ (N + K + 2) \times RTT + RTO, & in\ case\ of\ loss \end{cases} \quad (2)$$

$$Average\ Time\ of\ Virtual\ Start = \left(N + \frac{K}{2} + 1\right) \times RTT + RTO \quad (3)$$

그림 2와 그림 3의 느린 시작과 가상 시작의 소요시간은 각각 (1)과 (2)로 나타낼 수 있다. $N$+1은 cwnd = 1에서 시작하여 오류 없이 임계값까지 도달하는 라운드의 개수를 의미하며, $K$는 처음 라운드부터 오류가 발생하는 라운드까지의 개수에 -1을 합한 값을 의미한다. $RTO$는 오류가 발생하여 타임아웃까지 대기하는 시간을 의미한다. (2)는 통신지연과 패킷손실이라는 2가지 경우가 존재하며, 이를 (3)과 같이 평균적으로 나타낼 수 있다. 결론적으로 (1)과 (3)을 바탕으로 느린 시작이 가상 시작보다 $K/2 \times RTT$ 만큼의 소요시간이 더 필요하다는 것을 확인할 수 있으며, 이는 본 논문에서 제시한 가상 시작이 느린 시작보다 효율적임을 시사한다.

## Ⅲ. 결론

본 논문에서는 RTT가 높은 심 우주 통신환경에 적용할 수 있는 새로운 TCP 혼잡제어 알고리즘을 제시한다. 기존의 TCP 혼잡제어 알고리즘의 느린 시작과 다르게 본 논문의 가상 시작은 통신지연, 패킷손실 등으로 타임아웃이 발생할 경우 다시 시작하지 않고, 가상 ACK를 바탕으로 다음 라운드에서 실제 ACK를 수신할 수 있도록 한다. 이와 같은 가상 시작 알고리즘은 평균적으로 느린 시작 알고리즘에 비하여 낮은 소요시간을 가지며, 심 우주 통신환경에서 효율적임을 의미한다.

본 논문에서는 느린 시작 중 타임아웃이 발생하는 경우에 대하여 논의한다. 빠른 회복, 혼잡 회피 중 타임아웃에 가상 ACK를 적용할 수 있음을 논의하는 것은 이후 심 우주 통신환경에서 TCP 혼잡제어 알고리즘의 전체 소요시간을 줄이기 위한 주요 논점이 될 수 있다고 판단한다.

## ACKNOWLEDGMENT

본 연구는 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아수행된 연구임 [NRF-2018M1A3A3A02066018]

## 참 고 문 헌

[1] I. Petrov and T. Janevski, "Improved TCP Slow start algorithm," 2013 21st Telecommunications Forum Telfor (TELFOR), Belgrade, 2013, pp. 121-124.

[2] N. Li, Y. Tu and Z. L. Deng, "Satellite Network Oriented TCP Slow Start Algorithm," 2019 IEEE 19th International Conference on Communication Technology (ICCT), Xi'an, China, 2019, pp. 1116-1119.

[3] O. B. Akan, H. Fang and I. F. Akyildiz, "Performance of TCP protocols in deep space communication networks," in IEEE Communications Letters, vol. 6, no. 11, pp. 478-480, Nov. 2002.