Mininet을 활용한 Initial Window 크기 변화에 따른 TCP 혼잡제어 성능 평가

서유나, 이재용*, 김병철
충남대학교 전파정보통신공학과
sun02847@o.cnu.ac.kr, *jyl@cnu.ac.kr, byckim@cnu.ac.kr

Performance Evaluation of TCP Congestion Control with Varying Initial Window Sizes Using Mininet

Seo Yu Na, Lee Jae Yong*, Kim Byung Chul
Department of Radio and Information Comm. Eng. Chungnam National Univ.

요 약

본 논문은 다양한 Initial Window 크기에서 TCP 혼잡제어 알고리즘에 따른 성능 분석을 위해, 먼저 짧은 파일 전송에서 핵심 지표인 흐름 완료 시간(FCT)을 평가했다. CUBIC은 IW가 클수록 FCT는 감소하고, RTT가 클수록 FCT는 증가했다. pacing을 적용한 CUBIC은 초기 버스트와 큐잉 지연을 완화해 특히 높은 RTT에서 FCT의 개선 폭이 컸다. BBR은 IW와 RTT 변화에 덜 민감하면서, 낮고 안정적인 FCT를 보였다. 대용량 전송에서는 처리량이 안정화되는 시점(수렴 시간)과 처리량 변화를 관찰했다. CUBIC은 초기 손실로 인한 처리량의 진동이 컸으며, 일부 높은 RTT와 큰 IW 조합에서 수렴 시간이 오히려 증가했다. pacing을 적용한 CUBIC은 비교적 빠르게 수렴했으나, 높은 RTT와 큰 대역폭에서는 초기 pacing으로 처리량의 최대치 도달이 지연되었다. 한편, BBR은 IW 변화에 대한 민감도가 낮고 목표 처리량에 안정적으로 수렴했다.

Ⅰ. 서론

전송 계층의 대표 프로토콜인 TCP는 신뢰적인 데이터 전송과 혼잡제어로 네트워크 안정성을 보장한다. 초기 혼잡 윈도우(Initial Window)는 TCP 3-way handshake 직후 ACK 없이 전송 가능한 데이터의 최대 크기(MSS×세그먼트 수)로, 짧은 흐름의 지연에 직접적인 영향을 준다[1].

흐름 완료 시간(Flow Completion Time[2])은 전송 시작부터 마지막 패킷 수신까지의 시간으로, 짧은 파일 전송 성능의 핵심 지표이다. IW가 클수록 FCT는 감소하지만, IW가 지나치게 크면 손실과 재전송으로 FCT가 증가할 수 있다[3]. 대용량 전송에서는 혼잡제어 알고리즘에 따라 처리량(throughput) 안정화 시점이 달라져, 수렴 시간(convergence time)이 핵심 지표가 된다.

본 논문은 Mininet[4] 환경에서 IW 크기에 따른 TCP 전송 성능을 평가한다. 짧은 파일 전송에서는 FCT를, 대용량 전송에서는 수렴 시간을 측정하고, RTT·대역폭·혼잡제어 알고리즘(CUBIC[5], paced CUBIC, BBR[6])별 성능을 비교하여 최적 파라미터 설정의 근거를 제시한다.

Ⅱ. Initial Window와 TCP 혼잡제어 알고리즘 연구 동향

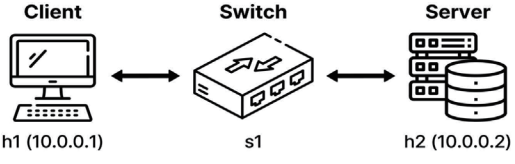
최근 연구에서는 IETF가 권고한 IW10과 달리, 일부 CDN 사업자는 콘텐츠 특성 및 네트워크 상황에 맞춰 IW를 조정하며, IW10보다 큰 IW50, 드물게는 IW100까지도 사용하는 사례가 보고되었다[1].

큰 IW의 효과는 네트워크 조건에 따라 달라진다. 대역폭과 큐가 충분한 환경에서는 FCT가 감소하지만, 저대역폭이나 작은 큐 환경에서는 초기 버스트(burst)로 FCT가 증가할 수 있다. 이를 완화하기 위해 pacing을 적용해 초기 데이터를 RTT 구간에 분산 전송한다. CUBIC은 손실을 혼잡 신호로 인식해 혼잡 윈도우(cwnd)를 증가시키는 손실 기반 혼잡제어 알고리즘이다[5]. 반면 BBR은 대역폭(BtlBw)과 왕복 지연(RTprop)을 추정해 BDP에 맞춰 cwnd와 pacing rate을 조정하는 모델 기반 혼잡제어 알고리즘이다[6]. 이로 인해 BBR은 IW 크기 변화에 덜 민감하고 작은 IW 환경에서도 성능 저하가 적어 유리하지만, 높은 RTT와 큰 IW 조합에서는 paced CUBIC이 더 낮은 FCT를 보인다[3]. 종합하면, 고정된 단일 권장값보다는 상황에 따라 조정 가능한 IW 정책이 필요하다[1].

Ⅲ. 실험 환경 구축 및 파라미터 설정

본 논문은 Ubuntu 20.04 LTS에서 Mininet[4]과 GUI 도구 Miniedit을 활용해, <그림 1>과 같이 클라이언트(h1) - 스위치(s1) - 서버(h2)로 이루어진 단일 토폴로지를 구성했다. 링크 지연과 대역폭은 tc(netem/tbf)로 설정하고, 트래픽 생성과 측정은 iperf3로 수행했다. 각 조건을 5회 반복 측정 후 iperf3 로그에서 추출한 FCT와 처리량의 평균값을 사용했다.

실험은 두 가지 시나리오로 구성하였다. 첫째, 100KB의 짧은 파일 전송 시 IW(1, 2, 4, 10, 20, 50), RTT(30/100ms), 대역폭(10/30/100Mbps)을 변화시키며 FCT를 측정했다. 둘째, 파일 크기 제한 없이 100초 동안 전송 수행 시 IW(1, 4, 10, 50), RTT(30/100ms), 대역폭(10/100Mbps)을 변화시키며 처리량이 안정화되는 수렴 시간을 측정했다. 두 시나리오 모두 혼잡제어 알고리즘은 CUBIC, paced CUBIC, BBR을 대상으로 했다.



<그림 1> Mininet 환경에서의 단일 링크 실험망 구성

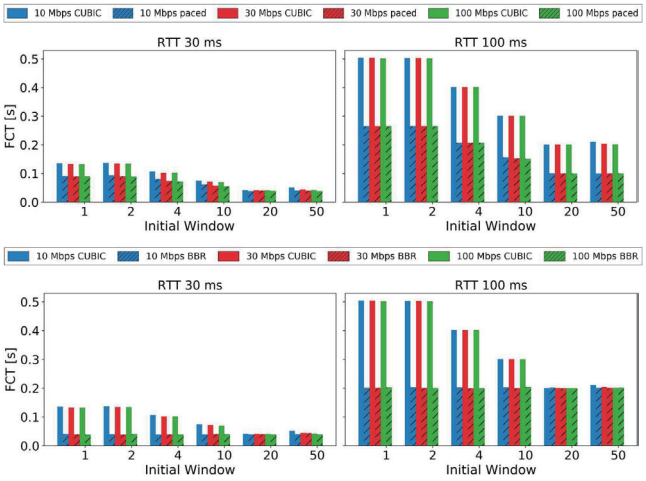
Ⅳ. IW 크기에 따른 TCP 혼잡제어 알고리즘별 성능 평가

본 실험에서는 짧은 파일 전송 성능을 분석하기 위해 FCT를 관찰했다. <그림 2> (a)는 CUBIC과 paced CUBIC의 FCT를 비교한 결과이다.

첫째, IW가 커질수록 FCT가 감소했으나, IW가 임계치를 넘으면 FCT가 오히려 증가했다. 예를 들어 CUBIC은 대역폭 10Mbps에서 IW=50인 경우, IW=20보다 FCT가 증가했다. 둘째, RTT가 클수록 지연 누적으로 FCT가 증가하였다. 셋째, 100KB 전송은 병목이 거의 없어, 대역폭을 늘려도 FCT 개선이 미미했다. 넷째, CUBIC에 pacing을 적용하면 초기 버스트가 완화되어, RTT가 큰 환경에서 FCT가 크게 감소했다.

다음으로 <그림 2> (b)는 CUBIC과 BBR의 FCT를 비교한 결과이다.

첫째, CUBIC은 IW 증가에 따라 FCT가 감소한 반면, BBR은 BDP 기반 제어로 IW 변화에 거의 영향을 받지 않았다. 둘째, 전반적으로 BBR은 CUBIC보다 낮은 FCT를 보였으며, IW가 작은 경우 두 알고리즘 간 성능 차이가 커졌다. 셋째, BBR은 RTT 증가에 따른 영향도 상대적으로 작았다. 또한 IW=20, RTT=100ms인 환경에서 paced CUBIC은 FCT가 약 0.1초, BBR은 약 0.2초로, 이와 같이 높은 RTT와 큰 IW를 가진 환경에서는 paced CUBIC이 BBR보다 더 낮은 FCT를 보임을 알 수 있다.



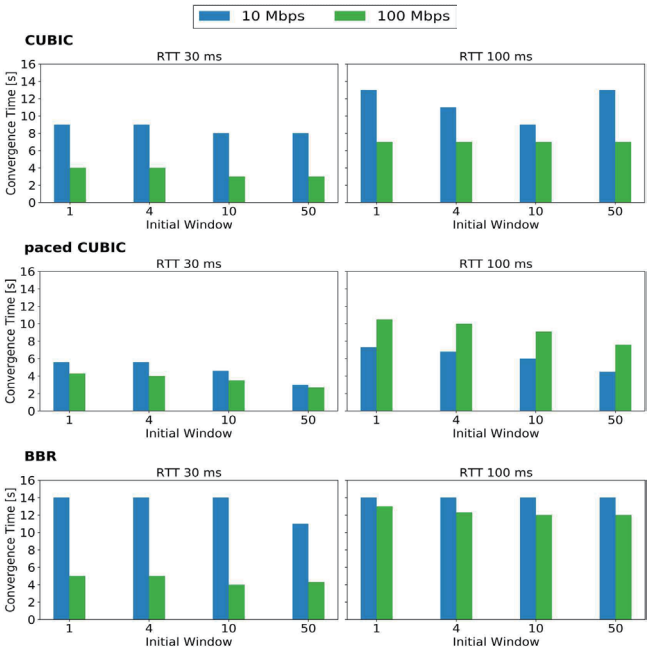
<그림 2> 평균 FCT: (a) CUBIC vs paced CUBIC (b) CUBIC vs BBR

다음 실험에서는 iperf3로 100초 동안 전송하며 혼잡제어 알고리즘의 long-term throughput을 평가했다. CUBIC, paced CUBIC, BBR을 대상으로 처리량이 안정 상태에 도달하기까지 걸린 수렴 시간을 비교했다.

<그림 3> (a)는 CUBIC의 결과이다. CUBIC은 초기에 처리량 급상승 후 손실과 재전송이 반복되어 전반적으로 처리량의 진동이 크게 나타났다. IW나 대역폭이 증가할수록 수렴 시간은 감소했지만, RTT가 증가할수록 ACK 지연으로 수렴 시간이 증가했다. 다만 IW=50, RTT=100ms, 대역폭 10Mbps 환경과 같이 고지연과 저대역폭 조건에서 큰 IW로 인한 초기 버스트로 손실이 커져, 오히려 IW=10보다 수렴 시간이 증가했다.

<그림 3> (b)는 paced CUBIC의 결과이다. CUBIC에 pacing을 적용하면 초기 버스트가 억제되어 손실이 줄고 전반적으로 더 빠르게 수렴했다. 다만 RTT=100ms에서는 대역폭이 커질수록 pacing으로 전송이 분산되어 큰 대역폭을 즉시 활용하지 못해 수렴 시간이 오히려 증가했다.

<그림 3> (c)는 BBR의 결과이다. 초기 버스트(StartUp) 이후 처리량은 병목 대역폭에 맞춰 수렴하며 비교적 진동이 작게 나타났다. BBR은 BDP를 기준으로 동작하기 때문에, IW 크기에 따른 수렴 시간의 변화가 크지 않았다. 또한 대역폭이 클수록 수렴 시간이 감소했고, RTT가 증가할수록 더 큰 BDP를 채워야 하기 때문에 수렴 시간이 증가했다.



<그림 3> 평균 Convergence Time: (a) CUBIC (b) paced CUBIC (c) BBR

Ⅴ. 결론

본 논문에서는 Mininet을 활용하여 RTT, 대역폭, 혼잡제어 알고리즘 변화에 따른 IW 크기의 성능 영향을 분석하였다.

짧은 파일 전송에서 CUBIC은 IW가 커질수록 FCT가 감소했으며, pacing 적용 시 높은 RTT 환경에서 FCT가 크게 개선되었다. BBR은 IW와 RTT 변화에 덜 민감하며 낮고 안정적인 FCT를 유지했다. 대용량 전송에서는 CUBIC의 처리량 진동이 컸고, paced CUBIC은 전반적으로 더 빨리 수렴했다. BBR은 IW 변화에 둔감하며 안정적으로 수렴했다.

결과적으로 IW, RTT, 대역폭, 혼잡제어 알고리즘의 조합이 TCP 전송 성능 최적화의 핵심 변수임을 확인하였다.

참 고 문 헌

[1] J. Rüth, C. Bormann, and O. Hohlfeld, “Large-Scale Scanning of TCP’s Initial Window,” Proc. ACM Internet Measurement Conf. (IMC), London, UK, pp. 1 - 7, Nov. 2017.

[2] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, “Why Flow Completion Time is the Right Metric for Congestion Control,” ACM SIGCOMM Computer Communication Review, vol. 36, no. 1, pp. 59 - 62, Jan. 2006.

[3] J. Rüth, I. Kunze, and O. Hohlfeld, “TCP’s Initial Window - Deployment in the Wild and its Impact on Performance,” IEEE Trans. Network and Service Management, vol. 16, no. 3, pp. 1 - 14, Sep. 2019.

[4] “Mininet: An Instant Virtual Network on your Laptop (or other PC),” Online. Available: https://mininet.org/.

[5] Ha, S., Rhee, I., and Xu, L., “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” ACM SIGOPS Operating Systems Review, 42(5), pp. 64 - 74, Jul. 2008.

[6] Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and Jacobson, V., “BBR: Congestion-Based Congestion Control,” ACM Queue, 14(5), pp. 20 - 53, Oct. 2016.