논문 19-44-09-15 The Journal of Korean Institute of Communications and Information Sciences '19-09 Vol.44 No.09
https://doi.org/10.7840/kics.2019.44.9.1713

BBR 혼잡제어 알고리즘 표준화 및 연구 동향 분석

김 건 환•, 송 영 준*, 박 창 훈*, 조 유 제°

Standardization and Research Trends of BBR Congestion Control Algorithm

Geon-Hwan Kim•, Yeong-Jun Song*, Chang-Hoon Park*, You-Ze Cho°

요 약

현재 인터넷상에서 이용되는 웹 응용들의 90% 이상의 트래픽이 전송계층 프로토콜로 TCP를 사용하고 있다. 구글은 2016년 말 웹 응용들의 네트워크 속도 향상을 위한 새로운 혼잡제어 알고리즘으로 BBR (Bottleneck Bandwidth and Round-trip propagation time)을 제안하였다. BBR은 손실기반 혼잡제어 알고리즘과는 달리 병목링크의 버퍼를 가득 채우지 않으며 동작하기 때문에 패킷 손실을 줄이고 전송지연을 최소화할 수 있어 최근 많은 연구자들의 관심을 끌고 있다. 본 고에서는 BBR 알고리즘에 대한 동작 특성과 최신 표준화 동향을 살펴보고 연구개발 동향 및 이에 대한 현재의 연구 이슈를 고찰한다.

키워드 : TCP 혼잡제어, BBR 알고리즘, 연구 동향, 표준화 동향, 구글
Key Words : TCP congestion control, BBR algorithm, Research issue, Standardization trends, Google

ABSTRACT

In the current internet, more than 90% of the web applications use TCP as transport layer protocol. Google in late 2016 proposed the Bottleneck Bandwidth and Round-trip propagation time (BBR) as a new congestion control algorithms to improve the overall network performance. Unlike the existing loss-based congestion control algorithms, BBR works without filling the buffer of the bottlenecks, thus it reduces packet losses and minimizes the end-to-end delays; therefore, draws attention from many researchers. In this paper, firstly, we observe the BBR's behavior characteristics, then decribe the current standardization, R&D and research issues of BBR algorithm.

Ⅰ. 서 론

TCP가 1980년대부터 사용해 오고 있는 손실기반 혼잡제어 (loss-based congestion control) 알고리즘은 기본적으로 패킷 손실을 혼잡으로 판단한다. 과거의 낮은 전송 대역폭, 스위치와 라우터의 작은 버퍼 크기,

※ This research was supported by Next Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. NRF-2017M3C4A7083676).
※ This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2018R1A6A1A03025109).
※ This research was supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1006249).
• First Author : School of Electronics Engineering, Kyungpook National University, kgh76@ee.knu.ac.kr, 학생회원
° Corresponding Author : School of Electronics Engineering, Kyungpook National University, yzcho@knu.ac.kr, 정회원
* School of Electronics Engineering, Kyungpook National University, {syj5385, pch4495}@knu.ac.kr, 학생회원
논문번호 : 201905-080-B-RN, Received May 13, 2019; Revised July 15, 2019; Accepted July 15, 2019

패킷 손실이 대부분 버퍼범람 (buffer overflow)에 의해 발생하는 유선 중심의 인터넷 환경에서는 비교적 성능상의 문제 없이 잘 동작하였다. 하지만, 패킷 손실이 혼잡보다 채널의 비트 오류에 의해 빈번히 발생하는 WiFi나 LTE 등의 무선 인터넷 환경에서는 기존의 TCP Reno, NewReno, BIC, CUBIC 등 손실기반 혼잡제어 알고리즘은 goodput의 저하를 겪는다. 또한, 스위치와 라우터 등의 하드웨어의 발전, 광전송, WiFi, LTE 등 유무선 통신 기술의 지속적인 발전으로 1990년대 이후 네트워크의 전송용량이 1,000배 이상 향상되었지만 이러한 전송용량의 증가는 오히려 기존 손실기반 혼잡제어 알고리즘의 높은 지연시간을 초래했다. 따라서, 무선 인터넷 사용의 확대와 하드웨어 성능의 발전에 부합하여 새로운 TCP 혼잡제어 알고리즘의 필요성이 대두되었다.

구글은 2016년 말 기존 TCP의 손실기반 혼잡제어 알고리즘을 대체하기 위해, 새로운 개념의 혼잡기반 혼잡제어 (congestion-based congestion control) 알고리즘인 BBR (Bottleneck Bandwidth and Round-trip propagation time)을 발표했다[1].

일반적으로 TCP 플로우의 종단 간 처리율은 병목링크의 데이터 전송률인 bottleneck bandwidth (BtlBw)에 의해 결정된다. 그리고, 데이터 전달의 최소 왕복지연은 종단 간 링크의 round-trip propagation time (RTprop)이 된다. 네트워크 경로를 물리적인 파이프에 비유하면 RTprop은 파이프의 길이, BtlBw는 최소 직경에 해당된다. BBR은 Kleinrock의 최적 동작점에서 동작하도록 종단 간 RTprop과 병목링크의 BtlBw를 probing을 통해 주기적으로 측정하여, 동적으로 전송률을 제어하는 새로운 패러다임의 혼잡제어 방안이다[2].

구글은 이미 BBR을 혼잡제어 알고리즘으로 도입하여 Youtube 서버와 자체 클라우드 플랫폼에 적용하였으며, 실험을 통해 TCP BBR은 기존의 TCP CUBIC에 비해 처리율이 2~20배까지 향상됨을 보였다[1][3]. TCP BBR은 현재 리눅스의 네트워크 스택에 구현되어 공개되었으며, BBR 버전 2.0으로의 개선을 진행 중이다. 구글은 궁극적으로 BBR을 IETF (Internet Engineering Task Force)의 TCP 혼잡제어 표준 알고리즘으로 만들기 위해 노력하고 있다[4].

최근 다양한 시나리오에 TCP BBR을 적용한 성능 평가 실험들이 활발히 이루어지고 있으며, 구글이 보고한 BBR의 큰 성능 개선과는 별개로 재전송률의 증가, 손실기반 혼잡제어 알고리즘과의 공존 문제, 서로 다른 RTT (Round-Trip Time)을 갖는 BBR 플로우 간 공평성 문제 등이 지적되고 있다[14-18].

본 고에서는 TCP BBR 알고리즘의 동작 과정과 그 특징, 연구 동향 및 최신 표준화 동향을 살펴보고 문제점을 분석한다. 본 고의 구성은 다음과 같다. Ⅱ장에서는 BBR 알고리즘의 특징 및 동작 과정에 대해 설명한다. Ⅲ장에서는 현재 IETF에서 진행되고 있는 BBR 표준화 동향에 대해서 살펴본다. Ⅳ장에서는 BBR 관련 연구들을 소개하고 Ⅴ장에서 본 논문의 결론을 맺는다.

Ⅱ. BBR 동작 개요

2.1 TCP 혼잡제어 알고리즘의 동작점

기존의 TCP에서 사용하는 손실기반 혼잡제어 알고리즘은 혼잡으로 판단하기 위한 지표로 패킷 손실을 사용한다. 송신 측은 병목링크에서 패킷 손실이 발생할 때까지 지속적으로 전송률을 증가시키고, 패킷 손실이 발생하면 혼잡으로 판단하여 다시 전송률을 낮추는 과정을 반복한다. BDP (Bandwidth-Delay Product)를 넘어서는 데이터를 전송하게 되면 초과 데이터는 병목링크에서 대기열을 생성하고 종단 간 전송지연의 증가를 초래한다. 초과 데이터의 양이 병목링크의 최대 버퍼 크기에 도달하면 대기열은 가득 차게 되고, 새롭게 도착하는 패킷은 폐기된다. <그림 1>의 (B)점이 병목 버퍼의 범람으로 인해 패킷 손실이 발생하는 지점으로 손실기반 혼잡제어의 동작점이 된다.

TCP 플로우의 종단 간 처리율은 병목링크의 데이터 전송률에 의해 결정된다. 병목링크의 데이터 전송률 BtlBw와 종단 간 링크의 RTprop의 곱인 BtlBw * RTprop = BDP 만큼의 데이터양이 전체 링크 상에 전달되어야 링크의 가용 용량을 100% 활용할 수 있

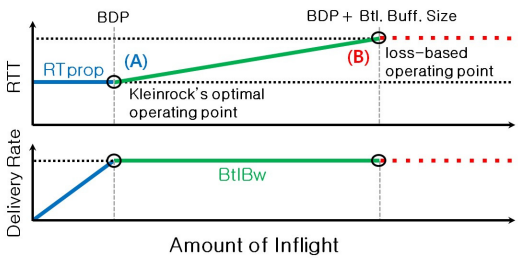


그림 1. 전송 중인 데이터양(amount of inflight)에 따른 전송률(delivery rate)과 RTT의 변화
Fig. 1. Delivery rate and RTT according to the amount of inflight

다. 1979년 Leonard Kleinrock은 <그림 1>의 (A)점이 최소의 전송지연이면서 최대의 throughput을 얻을 수 있는 최적의 동작점임을 증명하였다. BBR은 Kleinrock의 최적 동작점에서 동작하도록 종단 간 RTprop과 병목링크의 BtlBw를 주기적으로 측정하여, 병목 구간이 포화 상태를 유지하지만 혼잡은 발생하지 않도록 동적으로 전송률을 제어하는 혼잡제어 방안이다.

2.2 BBR 혼잡제어 알고리즘

<그림 2>는 BBR의 데이터 전달 과정을 나타낸다. BBR은 주기적으로 측정하는 대역폭, 종단 간 전송지연 샘플을 사용하여 파이프의 가용 BtlBw와 RTprop에 대한 명시적 모델을 구축한다. BtlBw와 RTprop 측정치는 파이프 내에 적절한 양의 패킷을 유지하기 위해 전달 중인 데이터 (inflight)의 양을 늘리거나 줄이는 일종의 probing 상태머신에 공급된다. 이어서, 상위 계층으로부터 전달된 데이터를 전송계층 프로토콜인 TCP가 퀀텀 (quantum) 크기로 자르고 최대 데이터양 (cwnd)을 초과하지 않도록 주어진 속도 (rate)로 데이터를 전달한다.

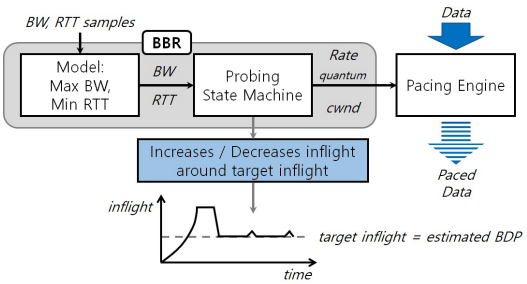


그림 2. BBR의 데이터 전달 과정
Fig. 2. Data delivery process of BBR algorithm

2.3 BBR의 네 가지 동작 과정

BBR 알고리즘은 <그림 3>과 같이 네 개의 동작 구간으로 구분된다.

2.3.1 Start up

BBR 플로우의 시작 단계에서는 각 RTT마다 지수함수적 증가를 통해 전송률을 크게 높인다. 세 번의 연속된 전송률 증가 시도에 대해 전송률의 증가가 25%를 넘지 못한다면, 병목 대역폭을 발견한 것으로 간주하고 Start up 단계를 종료한다. Start up 단계에서의 pacing_gain은 약 2.885이며 전달 가능한 최대 데이터양은 3 BDP로 제한된다. 이는 파이프를 채우

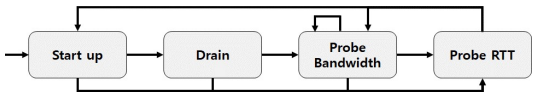


그림 3. BBR 동작 과정
Fig. 3. BBR operational process

는 1 BDP를 제외하고 2 BDP의 초과 대기열이 생성될 수 있음을 의미한다.

<그림 4>는 TCP BBR 플로우가 시작된 후 1초 동안의 RTT와 pacing_gain의 변화를 나타낸다. 회색의 수직선은 BBR 동작 과정의 전환을 의미한다. BBR은 Start up 구간에서 최대 전달 가능한 데이터양인 3 BDP에 도달한 후, 더 이상의 전송률 증가가 이루어지지 않는 평평한 구간을 보인다. 이 때의 RTT는 TCP BBR 플로우 RTprop의 3배이며, <그림 4>는 120ms의 RTT를 보인다.

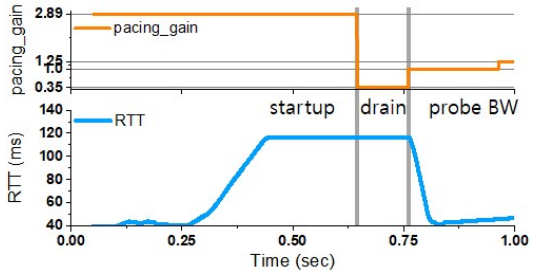


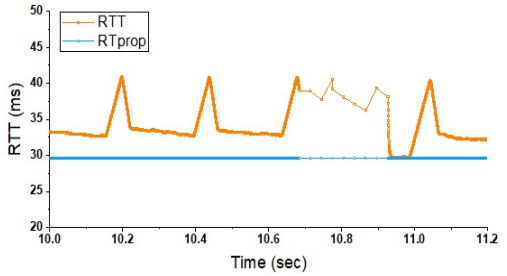
그림 4. 10Mbps BtlBw, 40ms RTprop에서 BBR 플로우의 시작 후 1초간 RTT 및 pacing_gain 변화
Fig. 4. RTT and pacing_gain for 1 second after start of BBR flow at 40ms RTprop and 10Mbps BtlBw

2.3.2 Drain

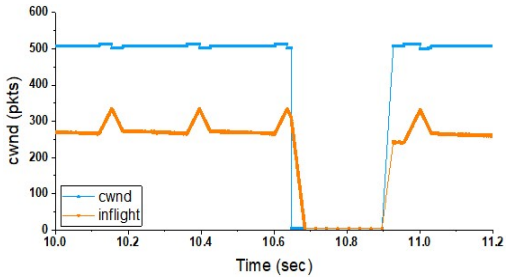
Start up 단계에서 병목 대역폭을 발견하면 Drain 단계가 수행된다. 이전 단계에서의 신속한 전송으로 인해 발생할 수 있는 대기열을 제거하기 위한 과정이다. Drain 단계에서는 초과 대기열을 제거하기 위해 Start up 단계에서의 pacing_gain 값의 역수를 사용한다. 이 때의 pacing_gain은 약 0.35이며, 전달 중인 데이터의 양이 1 BDP로 감소하면 Drain 과정을 종료하고 Probe Bandwidth 단계를 수행한다. <그림 4>의 0.75초 부근에서 약 120ms로 유지되던 RTT가 Drain 단계를 거쳐 급격하게 감소하는 것으로부터 Start up 단계에서 생성된 초과 대기열이 제거되었음을 알 수 있다.

2.3.3 Probe Bandwidth

BBR은 가용 대역폭을 탐색하기 위해 대부분의 시간을 Probe Bandwidth에 머무른다. 이 단계는 [1.25,



(a) RTT와 RTprop



(b) cwnd와 inflight

그림 5. 100Mbps BtlBw, 30ms RTprop에서 단일 BBR 플로우 동작
Fig. 5. Single BBR flow behavior at 30ms RTprop and 100Mbps BtlBw

0.75, 1, 1, 1, 1, 1, 1]의 pacing_gain 값들로 8 사이클을 구성하며 각 사이클은 RTprop동안 지속된다. 먼저, pacing_gain이 1.25일 때는 가용 대역폭이 있는지 알아보기 위해 더 많은 데이터를 전송한다. 이어지는 0.75에서는 앞선 탐색 과정에 의해 발생할 수 있는 초과 대기열을 제거한다. 나머지 여섯 사이클에서는 탐색 과정에서 얻은 최대 병목 대역폭을 토대로 pacing_gain = 1의 일정한 속도로 전송한다. 1.25 구간에서 증가된 전송률이 전달률의 증가를 초래하면 새롭게 측정된 최대 전달률이 즉시 새로운 전송률로 사용되고, 일정하게 유지된다면 기존 전송률을 그대로 사용한다.

2.3.4. Probe RTT

Probe Bandwidth 과정 중에 10초 동안 새로운 RTprop 값 (더 낮은 측정값)이 측정되지 않는다면 Probe RTT 단계를 수행한다. Probe RTT에서는 링크의 실제 RTprop을 측정하기 위해서 최대 패킷 수를 4개로 줄여 병목에서의 가능한 모든 대기열을 제거한다. 최소 RTprop의 측정을 보장하기 위해 이 단계는 200ms + RTprop 동안 지속되며, 새로운 최솟값이 측정되면 RTprop 값으로 갱신하고 이 값은 10초 동안 유지된다.

Probe RTT 과정은 <그림 5.(b)>의 10.7초 구간에서 잘 관찰된다. 그림의 10.7~10.9초 약 200ms 동안 cwnd 값이 4로 감소했으며, 이는 전달하는 패킷의 수가 4개로 제한되었음을 나타낸다. 이를 통해 많은 대기열이 제거되고, <그림 5.(a)>의 약 10.9초에서 RTprop 값을 측정할 수 있게 된다. Probe RTT의 종료 이후에는 Probe RTT를 시작하기 전의 전송률로 복귀하여 Probe Bandwidth 과정을 이어간다.

Ⅲ. BBR 표준화 동향

구글이 제안한 BBR 혼잡제어 알고리즘은 2016년 10월 netdev 1.2에서 리눅스 TCP 성능을 향상시키기 위한 방안으로 소개되어, Linux v4.9를 통해 공개되었다[5,6]. 2016년 11월 서울에서 열린 IETF-97에서는 BBR의 상세한 동작 방식과 다중 BBR 플로우의 수렴 과정에 대해서 설명하였다. TCP BBR은 많은 패킷 손실이 발생하는 환경에서도 높은 처리율을 보장하였고 낮은 지연을 보였다[3].

IETF-98에서는 BBR의 성능 개선을 위해 고려중인 세 가지 주요 변경 사항에 대해 소개하였고 BBR을 통한 다양한 연구주제 및 기회에 대해 언급하였다. 또한, 손실기반 혼잡제어 알고리즘인 TCP CUBIC, Reno와의 공평성 실험은 병목 버퍼의 크기에 따라 상반된 결과가 나타남을 보여주었다. 본 회의에서는 대기 지연과 패킷 손실을 줄이고 얕은 버퍼에서 손실기반 혼잡제어와 공존 가능하며, TCP BBR 플로우 간 RTT 공평성 문제를 완화하기 위해서 BBR의 동작을 수정하는 방안을 모색하였다[7]. 같은 해의 7월에 열린 IETF-99에서는 Youtube에서의 QUIC (Quick UDP Internet Connection)[8] 프로토콜에 BBR 알고리즘을 적용하여 사용할 수 있음을 보고했으며[9], 전달률 추정과 BBR 혼잡제어 알고리즘에 대한 RFC를 출판했다[10,11].

IETF-100에서는 BBR 혼잡제어 알고리즘 공개에 따른 후속 연구로부터 보고된 문제점들을 개선하기 위해서 알고리즘의 동작을 변경한 BBR 버전 2.0에 대한 테스트가 진행 중임을 보고하였다[12]. 이는 BBR 버전 1.0에서 지적되었던 얕은 버퍼에서의 높은 재전송률을 줄이고 공평성을 개선하여 결과적으로 BBR 알고리즘의 성능을 향상시키는 것에 초점을 두었다. 본 회의에서는 BBR 버전 2.0으로의 변경 점에 대한 상세한 설명은 없었으며 기존 BBR 버전 1.0과의 성능 비교 결과만을 공개하였다.

2018년 7월의 IETF-102에서 비로소 BBR 버전 2.0

으로의 변경 사항으로, 손실기반 혼잡제어 알고리즘과의 공평성을 개선하기 위한 대역폭 탐색 주기 조절 방안, 패킷 손실과 대기 지연을 줄이기 위해 여분의 공간을 두는 방안, Start up 구간을 더 빠르게 종료하기 위한 새로운 추정 방안, Probe RTT의 수행 간격과 전달률 감소 폭을 기존 10초에서 2.5초로, 기존 4개 패킷으로의 제한을 전달률의 절반으로만 제한하도록 변경하는 방안을 소개하였다[13]. 이러한 변화들을 통해 손실기반 혼잡제어 알고리즘과의 공평성이 개선되었고 BBR 버전 1.0보다 안정적으로 동작함을 보고했다[4].

Ⅳ. BBR 성능 평가 연구 동향

본 장에서는 BBR 관련 연구들을 살펴보고 관련 이슈들에 대해 알아본다.

4.1 BBR 알고리즘 평가

초기 구글에서는 TCP BBR이 CUBIC에 비해 처리율이 2~20배까지 향상될 수 있음을 보고했다[1]. 이에 따라, TCP BBR의 성능을 다방면으로 평가하는 논문이 수행되었다[14,15].

논문 [14]에서는 리눅스 커널에 구현된 TCP BBR 알고리즘을 사용하여 실제 테스트베드를 구성하였다. 테스트베드 실험을 통해 다양한 이슈들을 소개하였고 이러한 이슈들의 주요한 원인이 되는 BBR 알고리즘의 설계 문제점을 지적하였다. 테스트베드 실험에서는 RTT, 플로우의 수, 병목 버퍼 크기를 변수로 설정하였고 네트워크 처리율, 대기 지연, 패킷 손실, 공평성에 대한 평가를 수행하였다. 위 논문에 따르면, BBR의 설계상 의도된 동작은 병목링크에 오직 하나의 플로우만 존재할 때 잘 관찰되며 동일한 병목링크에 다수의 플로우가 존재하는 경우에는 의도된 동작을 보이지 않았다. 이는 <그림 1>의 BBR 알고리즘 동작점 (A)가 다수의 플로우가 병목 지점을 지나는 경우에 대한 개별 송신자의 관점을 제대로 반영하지 않기 때문이다. 따라서, 각 송신자는 사용 가능한 대역폭을 실제보다 과평가하여 과도한 데이터를 보내게 된다. 이로 인해 TCP BBR은 대기 지연의 증가, 높은 패킷 손실률, 편향된 대역폭 사용과 같은 내재된 문제들을 초래한다. 특히, 높은 패킷 손실률은 <그림 6>과 같이 6개의 CUBIC 플로우 간 경쟁보다 6개의 BBR 플로우 간 경쟁에서 더욱 심각하게 발생하는 것을 확인할 수 있다. 이는 병목 버퍼의 크기가 0.8 BDP로 작은 상황에서 더욱 큰 차이를 보인다.

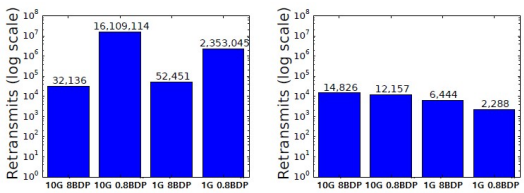


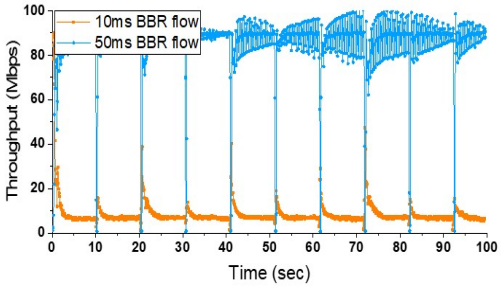
그림 6. 1Gbps, 10Gbps 병목 대역폭에서 버퍼 크기에 따른 패킷 재전송률[14]
Fig. 6. Retransmissions according to buffer size at 1Gbps, 10Gbps BtlBw

한편, 논문 [15]에서는 Mininet 에뮬레이션을 통해 TCP BBR 알고리즘에 대한 평가를 수행하였다. 논문 [14]와 마찬가지로 BBR 플로우 간 공평성 문제와 TCP CUBIC과의 공존 상황에 대해 평가하였다. 뿐만 아니라, TCP BBR 플로우 간의 동기화 과정에 대해 면밀히 분석하였으며 플로우 간의 안정화가 이루어지기까지의 시간은 새로운 BBR 플로우의 합류 시점에 크게 의존하는 것을 확인하였다. 새로운 BBR 플로우가 합류한 후 안정화 상태에 이르기까지 최대 30초가 소요될 수 있으며, 이는 비교적 짧은 시간 동안만 존재하는 현대의 인터넷 플로우 특성에 부정적인 영향을 미칠 것으로 판단하였다.

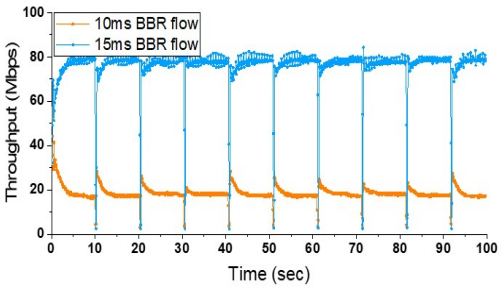
4.2 BBR 플로우 간 공평성

앞서 언급한 논문 [14]에 따르면, 동일한 RTT를 갖는 다수의 TCP BBR 플로우 간의 경쟁에서 일부 플로우가 억제되는 상황이 발생한다. 특정 실험 환경에서 일부 TCP BBR 플로우는 나머지 플로우보다 우세했으며 이러한 결과가 반복하여 나타났다. 다른 실험 환경에서는 각 플로우들이 유사한 전송률을 가졌지만, 플로우 간에 완벽하게 이상적인 공유는 나타나지 않았다. 하지만 동일한 RTT를 갖는 BBR 플로우 간의 공평성 문제는 특정한 실험 환경에서만 나타났으며, 대역폭이나 병목 버퍼 크기의 변화에 따라 항상 발생하지는 않았다. 이보다는, 동일한 논문에서 다루고 있는 BBR 플로우 간의 RTT 차이에 따른 공평성 문제가 병목 버퍼의 크기에 따라 분명하게 발생하며 그 격차가 굉장히 크기 때문에 주목할 만하다.

50ms TCP BBR 플로우와 10ms TCP BBR 플로우가 경쟁하는 경우, <그림 7.(a)>와 같이 50ms BBR 플로우가 대부분의 대역폭을 차지하는 것을 확인할 수 있다. 10ms BBR 플로우는 Probe RTT 직후 처리율이 증가하는 모습을 보이지만, 이러한 증가는 극히 일시적이며 이후로는 낮은 처리율을 갖는다. TCP BBR의 긴 RTT 플로우를 향한 편향은 두 개의 플로우 간 지연 차이가 단지 5ms인 경우에도 발생한다.



(a) 10ms BBR 플로우 vs 50ms BBR 플로우



(b) 10ms BBR 플로우 vs 15ms BBR 플로우

그림 7. BBR 플로우 간의 RTT 차이에 따른 처리율
Fig. 7. Throughput according to RTT difference between BBR flows

10ms BBR 플로우가 15ms BBR 플로우와 경쟁하는 상황에 대한 <그림 7.(b)>로부터 두 플로우 간의 큰 격차를 확인할 수 있다.

이처럼, 긴 RTT를 갖는 플로우가 짧은 RTT를 갖는 플로우를 과도하게 압도하는 것은 전통적인 TCP 혼잡제어 알고리즘에서의 짧은 RTT 플로우를 향한 편향과는 정반대의 결과이다. TCP BBR 알고리즘에서의 긴 RTT 플로우에 대한 편향은 특정 사용자가 악의적으로 지연을 늘려서 많은 대역폭을 독차지할 수 있기 때문에 심각한 문제를 초래할 수 있다. 또한, <그림 8>에서와 같이 두 TCP BBR 플로우 간의 지연 차이가 증가할수록 공평성 문제 또한 계속해서 악화

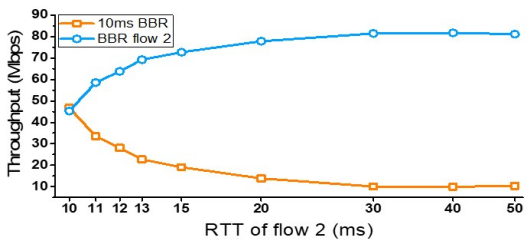


그림 8. BBR 플로우 간의 RTT 차이에 따른 처리율 변화
Fig. 8. Changes in throughput according to RTT differences between BBR flows

됨을 확인할 수 있다.

구글은 이미 동기화가 이루어진 BBR 플로우에 새로운 BBR 플로우가 합류한 상황에서의 동기화 과정을 보였다[1]. 새롭게 시작된 플로우는 정확한 RTprop의 측정이 이루어지지 않아서 병목 대역폭을 과대평가하며, 더 많은 대역폭을 차지하게 된다. 새로 합류한 플로우의 우세는 다음 Probe RTT가 시작될 때까지 유지된다. <그림 9>는 새로운 플로우가 60초에 합류하면서 약 20초 동안 플로우 간 공평성이 크게 변화하며, 합류한 플로우가 이 구간 동안 많은 대역폭을

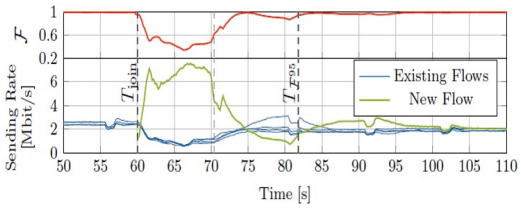
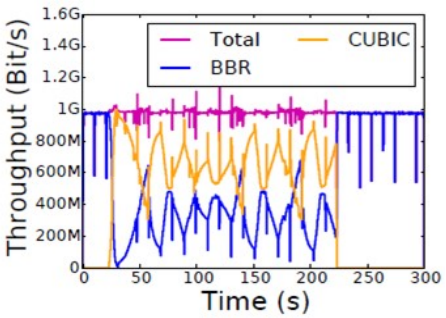
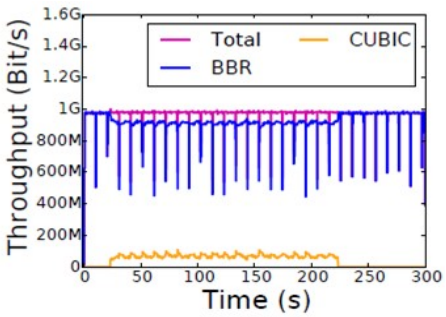


그림 9. 새로운 BBR 플로우의 합류에 따른 공평성 변화 [15]
Fig. 9. Fairness changes with the join of new BBR flow



(a) 1Gbps BtlBw, 8 BDP 버퍼



(b) 1Gbps BtlBw, 0.8 BDP 버퍼

그림 10. 버퍼 크기에 따른 BBR vs CUBIC 처리율 변화 [14]
Fig. 10. BBR vs CUBIC throughput variation with buffer size

차지함을 보인다.

논문 [15]에서는 이러한 TCP BBR의 동기화 과정에 대해 면밀히 살펴보았고, 신규 BBR 플로우의 합류 시점이 링크의 공평성에 주된 영향을 미치는 것을 밝혔다. 또한, BBR 플로우의 합류로 인한 공평성 감소가 누적 효과를 갖는지 확인하였으며, Probe RTT 간격인 10초 이내의 간격으로 플로우가 합류한 경우에는 동기화 상태가 되기까지의 시간이 더욱 소요되었고 10초 이후의 합류에 대해서는 누적 효과가 존재하지 않았다.

4.3 손실기반 혼잡제어 알고리즘과의 공평성

구글에 의해 새롭게 제안된 BBR 혼잡제어가 실제 인터넷에 적용되기 위해서는 기존에 사용 중인 혼잡제어 알고리즘과 공존할 수 있어야 한다. BBR이 리눅스 TCP의 기본 혼잡제어 알고리즘인 CUBIC과 공통 병목링크를 가지는 상황을 평가한 여러 연구들이 진행되었다[14-18].

먼저 [14]에서는 <그림 10>과 같이 병목링크의 버퍼가 큰 경우에는 TCP CUBIC, 버퍼가 작은 경우에는 TCP BBR이 더 많은 대역폭을 차지함을 보였다. 손실기반 혼잡제어인 TCP CUBIC은 버퍼가 가득 찰 때까지 전송률을 증가시키기 때문에 버퍼가 클수록 TCP CUBIC의 대역폭 점유량은 증가한다. 긴 대기열을 생성하는 TCP CUBIC의 동작 특성은 Probe RTT 단계를 수행하는 TCP BBR이 더 높은 RTprop를 측정할 수 있도록 하지만, 이미 우위에 있는 TCP CUBIC을 역전할 만큼 전송률이 증가하지는 않는다. 반면, <그림 10.(b)>와 같이 병목 버퍼가 0.8 BDP로 작은 경우에는 TCP BBR이 크게 우세한 모습을 보인다. 작은 버퍼 환경에서 TCP BBR에 의해 발생하는 과도한 패킷 손실은 같은 병목링크를 공유하는 TCP CUBIC이 크게 억제되게 한다. 이로부터, TCP BBR과 TCP CUBIC이 공동 병목링크를 공유할 때는 병목 버퍼의 크기에 크게 의존한다는 것을 알 수 있다. 경쟁 중인 두 혼잡제어의 공동 병목링크의 버퍼 크기에 따른 평균 처리율 차이를 확인하기 위해 테스트베드 실험을 수행하였으며, 그 결과는 <그림 11>과 같다.

논문 [16]에서는 대기열 스케줄링 알고리즘인 TailDrop과 CoDel을 사용할 때의 TCP BBR과 TCP CUBIC의 공평성을 비교하였으며, 각 10개 플로우에 대한 평균 처리율은 TCP BBR이 전반적으로 높게 나타났다. 특히, 각 혼잡제어를 사용하는 플로우의 최소 RTT가 크게 설정된 경우에는 CoDel을 사용하는 TCP BBR 플로우가 TCP CUBIC 플로우를 압도했다.

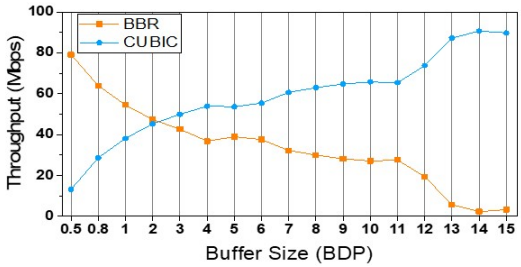


그림 11. 버퍼 크기에 따른 BBR과 CUBIC의 평균 처리율 차이
Fig. 11. Average throughput difference between BBR and CUBIC according to buffer size

논문 [17]에서는 TCP BBR과 TCP CUBIC 간의 성능을 공평성, 패킷 손실률, 네트워크 처리율, 처리율의 변화량, 지연시간 측면에서 살펴보았다. 평가를 위해 두 개의 가상머신으로 테스트베드를 구성하였으며, 실제 인터넷을 사용한 실험을 수행하였다. TCP CUBIC은 병목 버퍼의 크기 변화에 대해서 공평한 모습을 보인 반면, TCP BBR은 병목 버퍼 크기의 증가에 따라 변화가 나타났다. 두 혼잡제어 간의 경쟁상황에 대한 실험에서는 병목 버퍼 크기가 증가함에 따라 TCP BBR이 우세하다가 공평함을 보인 후 TCP CUBIC이 역전하는 모습을 보였다. 이로부터, TCP CUBIC과 TCP BBR 간에 공평성을 위한 최적 버퍼 크기가 있음을 예상하였고, 실험을 통해 약 1.5 BDP가 최적 버퍼 크기임을 주장했다. 또한, 하나의 TCP BBR 플로우와 두 개의 TCP CUBIC 플로우 간의 경쟁 실험에서는 TCP BBR 플로우가 약 절반의 처리율을 가졌고, 두 개의 TCP CUBIC 플로우는 서로 간의 경쟁에 의해 TCP BBR 플로우보다 낮은 처리율을 보였다.

4.4 무선 환경에서의 TCP BBR 알고리즘 평가

TCP BBR이 유선 네트워크 환경에서는 낮은 지연과 효율적인 대역폭 사용을 이끈다고 소개되었지만, 무선 네트워크에서의 성능은 명확하게 평가되지 않았다.

논문 [19]에서는 에뮬레이션과 실제 이동통신 네트워크를 통해 TCP BBR과 TCP NewReno, TCP CUBIC의 성능을 비교하였다. 다른 특징을 갖는 세 개의 오퍼레이터를 이용하여 파일을 다운로드하는 실험에서 TCP BBR이 다른 혼잡제어 알고리즘보다 비교적 낮은 지연을 가지며 빠르게 완료함을 보였다. 하지만 병목 버퍼의 크기가 작게 설정된 오퍼레이터를 이용한 실험에서는 긴 RTT를 갖는 BBR 플로우가 가장 빠른 완료 시간을 보였고, 이러한 긴 BBR 플로우

의 빠른 완료 시간은 이와 경쟁하는 짧은 RTT를 갖는 BBR 플로우를 억압하는 결과를 초래하였다. 따라서, 이동통신 환경에서 TCP BBR은 다른 비교 혼잡제어 알고리즘보다 나은 성능을 보였지만, 오퍼레이터의 네트워크 상태와 설정 또는 플로우의 유형에 따라서 항상 일정한 결과를 제공하지는 못했다.

논문 [20]에서는, LTE 네트워크에서의 TCP 혼잡제어 알고리즘에 따른 성능 비교 실험을 수행하였다. 실제로 차량을 주행하는 동안 LTE 네트워크를 통해 수집된 데이터를 기반으로 RTT, SINR, 네트워크 처리율, 핸드오버 상황에서의 처리율, 재전송률과 타임아웃 시간에 대해서 평가하였다. 실험을 통해 TCP BBR이 LTE 네트워크 환경에서 SINR이 낮고 핸드오버가 발생하는 상황에서도 높은 처리율을 가지는 것을 확인했으며, TCP CUBIC보다 이동통신 네트워크에 더욱 적합한 혼잡제어 알고리즘임을 주장했다.

4.5 TCP BBR 알고리즘 성능 개선 연구

앞서 언급한 TCP BBR의 공평성 문제를 해결하기 위한 연구가 소개되었다. 먼저 [21]에서는 TCP BBR 플로우 간의 RTT 공평성 문제가 발생하게 된 원인을 상세히 파악하였고 개선된 알고리즘을 제안하여 RTT 차이가 큰 TCP BBR 플로우 간의 공평성이 크게 개선됨을 보였다. 이들은 RTT 공평성 문제의 원인으로 병목 지점에서의 과도하고 빠른 대기열의 형성과 상대적으로 더 많은 양의 데이터를 보내는 긴 RTT 플로우가 버퍼의 대부분을 차지하기 때문이라고 주장하였다. 따라서, Probe Bandwidth 구간에서의 가용 대역폭 탐색주기를 제한함으로써, 긴 RTT 플로우와 짧은 RTT 플로우가 동일한 시간 동안 탐색을 수행하도록 하였다. 이를 통해, 기존 TCP BBR 알고리즘보다 병목 대기열 생성량이 대폭 감소하였고 RTT 공평성이 크게 개선되었다.

TCP BBR 플로우 간의 RTT 공평성 문제에 대한 개선 연구 뿐만 아니라, [22]에서는 병목 버퍼가 작을 때의 TCP BBR의 공격적인 성향을 완화하기 위해 변형된 BBR을 제안하였다. 병목 버퍼가 작은 상황에서 TCP BBR 플로우에 의해 발생한 많은 패킷 손실이 TCP CUBIC 플로우의 약화를 이끌었다. 이를 해결하기 위해, 기존 TCP BIC의 이진 탐색 기법을 사용하여 BBR의 전송률을 조절해서 TCP BBR이 과도하게 데이터를 전송하지 않도록 하였다. 즉, TCP BBR을 손실에 민감하도록 변형하여 손실의 발생에 따라 전송률을 감소시켜서 TCP CUBIC 플로우가 적절한 대역폭을 차지할 수 있도록 하였다. 제안된 방안은 기존 TCP BBR 알고리즘보다 낮은 재전송률을 보였고 TCP CUBIC과의 경쟁에서 상대적으로 개선된 공평성을 보였다.

V. 결 론

무선 인터넷 사용의 확대와 하드웨어 성능의 발전으로 기존의 손실기반 혼잡제어 알고리즘은 오히려 처리율의 저하와 긴 전송지연을 초래했다. 따라서, 현대의 인터넷 사용 특성에 부합할 수 있는 새로운 TCP 혼잡제어 알고리즘에 대한 연구가 지속적으로 진행되어왔다. 이에 구글은 웹 응용들의 네트워크 속도를 향상하기 위한 새로운 방안으로 최소 전송지연과 최대 전송률을 가지는 BBR 혼잡제어 알고리즘을 제안했다. 구글은 BBR을 IETF의 TCP 혼잡제어 표준 알고리즘으로 만들기 위해 노력하고 있으며, 향후 TCP 및 QUIC 표준 혼잡제어 알고리즘으로의 채택이 기대된다. 현재 공개된 BBR 버전 1.0에서 보고된 다양한 문제점에 대한 개선 및 성능 향상 연구와 향후 공개될 BBR 버전 2.0에 대해 보다 많은 연구가 진행될 것으로 예상된다.

References

[1] N. Cardwell, et al., "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 59-66, 2017.

[2] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *Proc. IEEE ICC*, pp. 43.1.1-43.1.10, Jun. 1979.

[3] N. Cardwell, et al., "BBR congestion control," in *Proc. ICCRG at IETF 97th Meeting*, Nov. 2016.

[4] N. Cardwell, et al., "BBR v2: A model-based congestion control," in *Proc. ICCRG at IETF 104th Meeting*, Mar. 2019, from https://datatracker.ietf.org/meeting/104/materials/slides-104-iccrg-an-update-on-bbr-00.

[5] N. Cardwell, et al., "Making linux TCP fast," in *Proc. Linux netdev 1.2 Conf.*, Oct. 2016, from https://netdevconf.org/1.2/slides/oct5/04_Making_Linux_TCP_Fast_netdev_1.2_final.pdf.

[6] N. Cardwell, et al., *BBR for Linux TCP in Linux* v4.9, from https://git.kernel.org/pub/scm/

linux/kernel/git/torvalds/linux.git/commit/?id=0f8782ea14974ce992618b55f0c041ef43ed0b78.

[7] N. Cardwell, et al., "BBR congestion control: An update," in *Proc. ICCRG at IETF 98th Meeting*, Mar. 2017, from https://www.ietf.org/proceedings/98/slides/slides-98-iccrg-an-update-on-bbr-congestion-control-00.pdf.

[8] Y. Cui, et al., "Innovating transport with QUIC: Design approaches and research challenges," *IEEE Internet Computing*, vol. 21, no. 2, pp. 72-76, 2017.

[9] N. Cardwell, et al., "BBR congestion control: IETF 99 update," in *Proc. ICCRG at IETF 98th Meeting*, Jul. 2017, from https://www.ietf.org/proceedings/99/slides/slides-99-iccrg-iccrg-presentation-2-00.pdf.

[10] Y. Cheng, et al., "*Delivery rate estimation*," Jul. 2017, Internet-Draft draft-cheng-iccrg-delivery-rate-estimation-00, IETF, work in progress.

[11] N. Cardweel, et al., "*BBR Congestion Control*," Jul. 2017, Internet-Draft draft-card well-iccrg-bbr-congestion-control-00, IETF, work in progress.

[12] N. Cardwell, et al., "BBR congestion control: IETF 100 update: BBR in shallow buffers," in *Proc. ICCRG at IETF 100th Meeting*, Nov. 2017, from https://datatracker.ietf.org/meeting/100/materials/slides-100-iccrg-a-quick-bbr-update-bbr-in-shallow-buffers/.

[13] N. Cardwell, et al., "BBR congestion control work at Google IETF 102 update," in *Proc. ICCRG at IETF 102th Meeting*, Jul. 2018, from https://datatracker.ietf.org/meeting/102/materials/slides-102-iccrg-an-update-on-bbr-work-at-google-00.

[14] M. Hock, et al., "Experimental evaluation of BBR congestion control," in *Proc. ICNP*, Toronto, ON, Canada, Oct. 2017.

[15] D. Scholz, et al., "Towards a deeper understanding of TCP BBR congestion control," in *Proc. IFIP Networking*, Zurich, Switzerland, May 2018.

[16] K. Sasaki, et al., "TCP fairness among modern TCP congestion control algorithms including TCP BBR," in *Proc. IEEE 7th Int. Conf. Cloud Netw. (CloudNet)*, Oct. 2018.

[17] R. J. Borgli and J. Misund, "Comparing BBR and CUBIC congestion controls," from https://www.uio.no/studier/emnet/matnat/ifi/INF5072/v18/inf5072_example1.pdf.

[18] Y. J. Song, et al., "Fairness improvement between BBR and CUBIC congestion control algorithm for TCP," in *Proc. JCCI*, May 2019.

[19] E. Atxutegi, et al., "On the use of TCP BBR in cellular networks," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 172-179, Mar. 2018.

[20] F. Li, et al., "TCP CUBIC versus BBR on the highway," in *Proc. Passive and Active Measurement*, pp. 269-280, Mar. 2018.

[21] S. Ma, et al., "Fairness of Congestion-Based Congestion Control: Experimental Evaluation and Analysis," *CoRR*, 2017, from https://arxiv.org/abs/1706.09115.

[22] Y. Zhang, et al., "Modest BBR: Enabling better fairness for BBR congestion control," in *Proc. IEEE Symp. Computers and Commun. (ISCC)*, Natal, Brazil, Jun. 2018.

김 건 환 (Geon-Hwan Kim)

2015년 2월 : 경북대학교 전자공학부 졸업

2017년 2월 : 경북대학교 전자공학부 석사

2017년 3월~현재 : 경북대학교 전자공학부 박사과정

<관심분야> 차세대 이동 네트워크, 드론 ICT 융합 기술, TCP 혼잡제어

[ORCID:0000-0003-2739-8939]

송 영 준 (Yeong-Jun Song)

2019년 2월 : 경북대학교 전자공학부 졸업
2019년 3월~현재 : 경북대학교 전자공학부 석사과정
<관심분야> TCP 혼잡제어, 미래 네트워크, 전송 계층 프로토콜

[ORCID:0000-0001-7586-5795]

박 창 훈 (Chang-Hoon Park)

2019년 2월 : 경북대학교 전자공학부 졸업
2019년 3월~현재 : 경북대학교 전자공학부 석사과정
<관심분야> MPTCP, QUIC, 다중경로 전송 계층 프로토콜

[ORCID:0000-0002-5066-2980]

조 유 제 (You-Ze Cho)

1982년 : 서울대학교 전자공학과 졸업
1983년 : 한국과학기술원 전기전자공학 석사
1988년 : 한국과학기술원 전기전자공학 박사
1989년~현재 : 경북대학교 전자공학부 교수
1992년~1994년 : Univ. of Toronto in Canada, 객원교수
2002년~2003년 : NIST(미국국립표준연구소) 객원연구원
<관심분야> 차세대 이동 네트워크, 무선 애드혹 네트워크, 이동성 관리 기술, 차세대 전송 계층 프로토콜

[ORCID:0000-0001-9427-4229]