https://doi.org/10.7840/kics.2024.49.1.95

TCP BBRv2와 CUBIC 알고리즘 간의 대역폭 공정성 개선

조 효 섭◆, 조 유 제°

Improvement of Bandwidth Fairness between TCP BBRv2 and CUBIC

Hyo-Seop Cho◆, You-Ze Cho°

요 약

현재 인터넷상에서 90% 이상의 트래픽은 전송 계층 프로토콜인 TCP(Transport Control Protocol)를 사용하고 있다. 네트워크 속도 개선을 위해 2016년에 구글은 새로운 개념의 혼잡제어 알고리즘인 BBRv1 (Bottleneck Bandwidth Round-trip Propagation Version1)를 제안했다. BBRv1 알고리즘은 작은 버퍼에서 다수의 문제점을 야기하였고, 구글은 2019년에 BBRv1 알고리즘의 문제점을 개선한 BBRv2 알고리즘을 제안했다. 하지만, BBRv2 플로우는 2 BDP 이상의 큰 병목 버퍼에서 CUBIC 플로우와 병목 대역폭을 공유할 때 대역폭 공정성 문제를 야기했다. 본 논문에서는 BBRv2와 CUBIC 알고리즘 간의 대역폭 공정성 문제를 개선하기 위한 F-BBRv2 알고리즘을 제안하고 시뮬레이션을 통해 성능향상을 보였다.

키워드 : TCP, TCP 혼잡제어, BBRv2 알고리즘, CUBIC 알고리즘, 구글
Key Words : TCP, TCP congestion control algorithm, BBRv2 algorithm, CUBIC algorithm, Google

ABSTRACT

Currently, over 90% of Internet traffic uses the TCP (Transmission Control Protocol). In 2016, Google proposed a new congestion control algorithm called BBRv1 (Bottleneck Bandwidth Round-trip Propagation Version 1) to improve network speed. However, BBRv1 had several issues with small buffers, leading Google to propose BBRv2 in 2019. Unfortunately, when flows of BBRv2 and CUBIC share bottleneck bandwidth in a large buffer (2 BDP or more), it causes fairness issues. This paper proposes the F-BBRv2 algorithm to address the fairness issue between BBRv2 and CUBIC.

Ⅰ. 서 론

TCP (Transport Control Protocol)는 인터넷 상에서 신뢰성 있는 데이터를 전달하기 위한 전송 계층 프로토콜이다[1]. 현재 인터넷 상에서 90% 이상의 트래픽은 TCP를 사용한다. TCP에서 혼잡제어 알고리즘은 TCP의 주요 기능 중 하나이며, 네트워크 혼잡을 미리 예방하고 대역폭을 공정하게 분배하여 최적의

※ This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by Ministry of Education (No. NRF-2018R1A6A1A03025109) and by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2023R1A2C1003928).
◆ First Author : School of Electronic and Electrical Engineering, Kyungpook National University, yg01175@knu.ac.kr, 학생회원
° Corresponding Author : School of Electronic and Electrical Engineering, Kyungpook National University, yzcho@knu.ac.kr, 종신회원
논문번호 : 202307-139-B-RN, Received July 3, 2023; Revised October 14, 2023; Accepted October 16, 2023

성능을 유지하는 데 중요한 역할을 한다. 이를 통해 데이터 전송 중에 혼잡상황이 발생하더라도 안정적인 통신이 가능해진다[2].

TCP가 1980년대부터 사용해 오고 있는 손실기반 혼잡제어 (loss-based congestion control) 알고리즘은 기본적으로 패킷 손실을 혼잡으로 판단한다. 특히, 패킷 손실이 혼잡보다 채널의 비트 오류에 의해 자주 발생하는 Wi-Fi나 LTE 등의 무선 인터넷 환경에서 기존의 손실기반 혼잡제어 알고리즘을 사용할 경우 goodput의 저하, 전송지연 증가 등 불필요한 성능저하가 발생할 수 있다. Reno나 CUBIC과 같은 손실기반 혼잡제어 알고리즘을 비롯하여 기계 학습이나 지연기반 혼잡제어 등 다양한 알고리즘이 제안되었다[3-6]. 현재 CUBIC은 대부분의 장비와 리눅스 커널에서 TCP의 기본 혼잡제어 알고리즘으로 사용되고 있다.

CUBIC은 손실기반 혼잡제어 알고리즘으로, 병목 버퍼를 최대한 활용하기 때문에 버퍼 크기에 따라 지연 시간이 길어지며, 종단 간 지연 시간을 증가시킨다. 또한, CUBIC은 패킷 손실을 혼잡의 신호로 인식하기 때문에 버퍼 크기에 따라 성능이 크게 달라진다. 만약 병목 버퍼 크기가 작으면 잦은 패킷 손실로 처리량이 줄어들게 되지만, 큰 버퍼 크기를 사용하면 패킷이 오랫동안 버퍼에 머물게 되어 버퍼블로트 현상이 발생할 수 있다[7].

구글은 2016년에 새로운 혼잡제어 알고리즘인 BBRv1(Bottleneck Bandwidth and Round-trip propagation time Version 1)을 제안하여, 이전의 손실기반 혼잡제어 알고리즘을 대체하고 대기열 지연을 제한하면서 높은 처리량을 달성했다[8]. BBRv1은 기존의 알고리즘과는 다르게, 최대 병목 대역폭(BtlBw)과 최소 왕복 지연 시간(RTprop)을 측정하고, 이러한 매개변수를 사용하여 네트워크 경로를 모델링한다. BtlBw와 RTprop의 곱으로 구성된 대역폭 지연 곱(BDP: Bandwidth Delay Product)에 기반하여, 페이싱 속도, 혼잡 창 등의 제어 매개변수가 계산되어 전송 속도를 조절한다. 따라서 BBRv1은 모델 기반 혼잡제어 알고리즘이라고 할 수 있다. 그러나, BBRv1 알고리즘은 병목 버퍼 크기가 작을 때 과도한 패킷 재전송과 CUBIC 알고리즘과의 낮은 대역폭 공정성 문제점이 존재한다.

구글은 2019년에 BBRv2 알고리즘을 제안하여, BBRv1이 가지고 있는 두 가지 문제점인 작은 병목 버퍼에서의 과도한 패킷 재전송과 CUBIC 알고리즘과의 대역폭 공정성 문제를 해결하기 위해 패킷 손실률과 ECN (Explicit Congestion Notification) 비율을 측정하여 혼잡상황을 감지하도록 설계했다[9-10].

BBRv2 알고리즘이 오픈소스로 공개되었고, 이후 많은 성능 평가 논문들이 제시되면서 세 가지 문제점이 논의되었다. 첫번째로, 모바일 네트워크와 같이 빠르게 변화하는 네트워크 환경에서의 낮은 링크 사용률 문제. 두번째로, 병목 버퍼 크기가 큰 경우 손실 기반 알고리즘을 적용한 플로우와 대역폭 공정성 문제. 마지막으로, BBRv2 플로우들이 다른 출발 시간을 가질 때 발생하는 RTT (Round-Trip Time)의 차이로 인한 대역폭 공정성 문제이다.

<그림 1>은 CUBIC 알고리즘이 적용된 플로우와 BBRv2 알고리즘이 적용된 플로우가 대역폭이 50Mbps로 한정되고 병목 버퍼 크기가 큰 상황에서 발생하는 대역폭 공정성 문제를 보여준다. 두 알고리즘을 적용한 각 플로우간 대역폭 공정성 문제가 발생하는 이유는 다음과 같다.

CUBIC 알고리즘은 손실기반 혼잡제어 알고리즘 중 하나로, 병목 버퍼를 최대로 채워 동작한다. 반면, BBRv2 알고리즘을 적용한 플로우는 CUBIC 플로우의 지속적인 패킷 손실로 인해 inflight_hi 설정에 영향을 준다. 이로 인해 병목 버퍼가 크고 대역폭이 한정된 네트워크 환경에서 대역폭 공정성 문제가 발생한다.

본 논문에서는 BBRv2 알고리즘에서 측정 가능한 최저 RTT인 RTprop의 변동성과 CUBIC 플로우간 관계에 대해 설명하고, 이를 활용하여 플로우간 공정성 문제를 해결하기 위한 F-BBRv2 (Fairness-BBRv2) 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 BBRv2 알고리즘과 관련된 연구들에 대해 설명하고, Ⅲ장에서는 BBRv2 알고리즘의 동작 개요를 다룬다. Ⅳ장에서는 F-BBRv2 알고리즘을 제안하며, Ⅴ장에서는

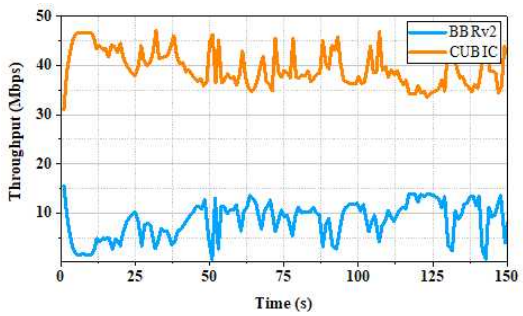


그림 1. BBRv2 및 CUBIC 알고리즘을 적용한 플로우간 시간 경과에 따른 처리량 변화 (버퍼 크기 = 4 BDP)
Fig. 1. Throughput change over time between flows with BBRv2 and CUBIC algorithm (Buffer size = 4 BDP)

NS-3(Network Simulation-3) 시뮬레이션을 이용하여 BBR 알고리즘들의 성능을 평가한다. 마지막으로 Ⅵ 장에서는 본 논문의 결론을 맺는다.

Ⅱ. 관련 연구

구글에서 발표한 BBRv2 알고리즘이 오픈소스로 공개된 이후 많은 논문들이 TCP BBRv2 알고리즘의 성능을 평가해왔다.

Kfoury 등은 다양한 네트워크 환경에서 BBRv2 알고리즘의 성능을 평가했다[11]. 그 결과, 무작위 패킷 손실이 있는 단일 경로 네트워크 환경에서 BBRv2 알고리즘을 적용한 플로우는 CUBIC 알고리즘을 적용한 플로우보다 처리량이 높았다. 또한, 병목 버퍼 크기가 작고 한정된 대역폭에서 BBRv2와 CUBIC 알고리즘을 적용한 플로우간 대역폭 공정성 문제가 개선된 것으로 평가되었다. 그러나, 병목 버퍼 크기가 클 때 두 알고리즘의 서로 다른 동작 특성으로 인해 BBRv2 알고리즘을 적용한 플로우는 CUBIC 알고리즘을 적용한 플로우보다 적은 대역폭을 차지하는 문제점이 있다.

또한, 병목 버퍼에 적용된 AQM (Active Queue Management) 정책이 BBRv2와 CUBIC 알고리즘을 적용한 플로우간 대역폭 공정성에 미치는 영향에 대해서도 설명한다. 많은 AQM 정책 중에서 FQ-CoDel (Flow Queue CoDel) 정책을 병목 버퍼에 적용하면 하나의 병목 버퍼에서 각 플로우 큐가 구분되어 누적되므로 대역폭 공정성 문제의 솔루션임을 제시했다.

Song 등은 BBRv1과 BBRv2 알고리즘의 동작을 비교 분석하고 Mininet 에뮬레이터를 사용하여 다양한 네트워크 환경에서 BBRv2 알고리즘의 성능을 평가했다[12]. BBRv2 알고리즘은 기존의 BBRv1 알고리즘보다 잦은 네트워크 변동에 대한 응답성이 낮으며, 시작 시간이 다른 BBRv2 플로우들이 동일한 병목 링크 대역폭을 공유할 때 대역폭 공정성 문제가 발생한다는 것을 지적했다. 또한, 병목 버퍼 크기가 0.2 BDP와 같이 매우 작을 때, BBRv2와 CUBIC 플로우는 한정된 병목 대역폭을 공정하게 공유하지만 병목 버퍼 크기가 2 BDP 이상일 경우 CUBIC 플로우가 더 많은 대역폭을 차지하면서 플로우간 대역폭 공정성 문제가 발생하는 것을 확인했다.

Ⅲ. BBRv2 동작 개요

3.1 BBRv2 혼잡제어 알고리즘

<그림 2>는 Kleinrock의 최적화 동작지점을 보여주며, (B) 지점은 버퍼 내의 패킷 수가 최대치에 도달하여 버퍼 오버플로우로 패킷 손실이 발생하는 지점을 나타낸다. 손실 기반 혼잡 제어 알고리즘은 버퍼를 최대한 채워 동작하므로 (B) 지점에서 동작한다. 그러나 BBR 알고리즘들은 최대 병목 대역폭인 BtlBw와 최소 RTT인 RTrpop의 곱인 BtlBw * RTprop = BDP를 계산하여 Kleinrock의 최적화 지점인 <그림 2>의 (A) 지점에서 동작한다. Kleinrock의 최적 동작 지점은 병목 버퍼에 잉여 패킷을 채우지 않고 대역폭을 100% 활용하며 최소 지연 시간을 달성할 수 있는 지점으로, 이는 Leonard Kleinrock이 1979년에 증명한 것이다[13].

Kleinrock의 최적화 지점에서 동작하는 BBR 알고리즘들은 병목 버퍼 크기에 관계없이 CUBIC 알고리즘 보다 2~20배 높은 처리량 달성했으며, 현재 BBRv1 알고리즘은 실제 구글과 유튜브 서버에서 사용하고 있다[14]. 하지만, BBRv1 알고리즘은 패킷 손실을 혼잡의 신호로 인지하지 않기 때문에 병목 버퍼 크기가 작을 때 높은 패킷 재전송률, 손실 기반 혼잡 제어 알고리즘을 적용한 플로우와 낮은 대역폭 공정성 문제점 등 다수의 문제점이 존재한다[15-17]. 본 문제점을 개선하기 위해 구글은 BBR의 두 번째 버전인 BBRv2를 제안했다.

<표 1>에서는 두 BBR 버전 간의 차이점이 요약되어 있다. BBRv2 알고리즘은 혼잡을 제어할 때 패킷 손실과 ECN률을 활용하며, 미리 정의된 임계 값 (loss_threshold = 2%, ECN_threshold = 50%)을 초과하는 패킷 손실률과 ECN 률을 감지할 때 inflight data를 제한하는 최대점인 inflight_hi를 사용한다. 또

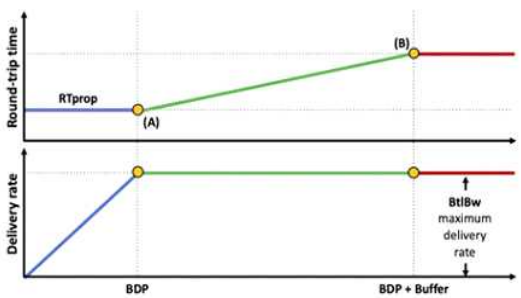


그림 2. Kleinrock의 최적 동작 지점
Fig. 2. Kleinrock's optimization point.

표 1. BBRv1과 BBRv2 알고리즘 비교
Table 1. Comparison of BBRv1 and BBRv2 Algorithms.

	BBRv1	BBRv2
Input value	Maximum bottleneck bandwidth RTprop	Maximum bottleneck bandwidth RTprop **Packet loss rate** **ECN rate**
Exit StartUp	Throughput stabilizer	Throughput Stabilizer or Loss and ECN rates exceed threshold
Inflight data in ProbeRTT	4 packets	**BDP / 2**

한, BBRv2는 ProbeRTT 단계에서 Congestion Window Size (CWND)의 크기를 BDP/2로 설계하여 BBRv1에서 발생한 처리량 변동 문제를 개선했다.

3.2 BBRv2의 네 가지 동작 과정

<그림 3>에서 볼 수 있듯이, BBRv2 알고리즘은 기존의 BBRv1 알고리즘에서 발생하는 pacing_gain 사이클 한계점을 극복하기 위해 ProbeBW 단계를 재설계했다.

3.2.1 StartUp

<그림 3 (a)>는 BBRv2 알고리즘에서 최대 병목 대역폭을 측정하는 StartUp 단계를 나타낸다. 본 단계에서는 각 RTT마다 전송 속도를 기하급수적으로 높여 최대 병목 대역폭을 빠르게 측정한다. 이를 위해 BBRv2 알고리즘에서는 pacing_gain 파라미터를 2.8로 설정하고, 전송 가능한 데이터 량 파라미터인 cwnd_gain을 3 BDP로 제한한다.

$$kHeadroom = inflight\_hi * headroom$$
$$Headroom = 0.15 \quad (1)$$

StartUp 단계에서는 패킷 손실이 임계 값을 초과하지 않고, 3개의 ACK 샘플에서 측정된 병목 대역폭 값이 25% 이상 증가하지 않는 지점을 BtlBw로 인식한다. 해당 BtlBw에 따라 최대 inflight data를 결정하는 inflight_hi가 결정된다. 그러나, 만약 StartUp 단계에서 패킷 손실 및 ECN 비율의 임계 값을 초과하면

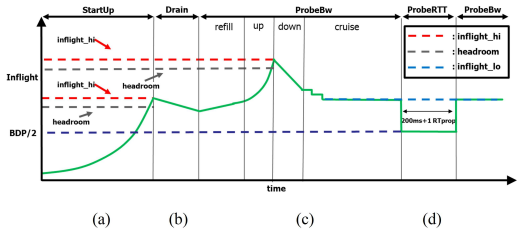


그림 3. BBRv2 알고리즘의 라이프 사이클
Fig. 3. BBRv2 algorithm life cycle.

StartUp 단계를 즉시 종료하고 다음 단계로 진입하기 위해 임계 값을 초과한 지점을 inflight_hi로 설정하고 다음 단계로 넘어간다. 또한, 다른 플로우와 대역폭 공정성 향상을 위한 여유 공간인 kHeadroom을 결정하게 된다. kHeadroom은 inflight_hi 값의 15%로 계산되며, 이 값은 식 (1)과 같이 계산된다.

3.2.2 Drain

<그림 3(b)>는 Drain 단계를 보여주며, 본 단계에서는 StartUp 단계에서 생성된 큐를 제거하기 위해 전송 속도를 높인다. 이를 위해, pacing_gain 값의 역수를 취하여 큐를 빠르게 제거한다. 즉, StartUp 단계에서는 전송 속도를 기하급수적으로 높여 BtlBw를 측정하는 반면, Drain 단계에서는 병목 버퍼에 쌓여 있는 큐를 빠르게 제거하기 위해 전송 속도를 낮춘다.

3.2.3 Probe Bandwidth

<그림 3(c)>는 ProbeBW (Probe Bandwidth)를 보여준다. 본 단계는 BBRv2 알고리즘에서 가장 많은 시간을 차지하는 단계로, 가용 가능한 대역폭을 측정하고 안정적으로 데이터를 전송한다.

BBRv2 알고리즘의 ProbeBW 단계는 4개의 단계로 세분화되어 있다. 첫 번째로, ProbeBW:Refill 단계는 추가적인 가용 대역폭을 측정하기 전에 패킷 손실을 방지하기 위한 단계로 inflight data 양을 선형적으로 증가시킨다. 두번째로, 추가적인 가용 대역폭을 빠르게 측정하기 위해 ProbeBW:UP 단계에 진입한다. 만약, 패킷 손실과 ECN이 임계 값을 초과하지 않고 1.25 * BDP에 도달하면 inflight_hi과 kHeadroom을 재설정하고 본 단계를 종료한다. 그러나, 패킷 손실과 ECN이 임계 값을 초과하는 경우 과도한 패킷 손실을 방지하기 위해 즉시 inflight_hi를 설정하고 본 단계를 종료한다. 세 번째로, ProbeBW:Down 단계는 이전 단계에서 inflight data 양을 급격하게 증가시켰기 때문에 병목 버퍼에 생긴 큐를 보상하기 위한 단계이다. 마지막으로 ProbeBW:Cruise 단계는 낮은 큐를 유지

하기 위해 세 라운드 동안 다운 과정을 거치고 패킷 손실과 ECN이 임계 값을 초과하지 않으면 안정적으로 데이터를 전송하며 순항한다.

3.2.4 ProbeRTT

ProbeRTT는 RTprop을 측정하는 단계이다. 본 단계는 <그림 3 (d)> 에 해당한다. RTprop가 측정될 때의 타임스탬프는 항상 유지되며, 새로 측정된 RTT가 기존 RTprop보다 낮으면 즉시 RTprop 값이 업데이트되고, 새로운 RTprop가 측정될 때의 타임스탬프가 기록된다. 그러나 5초 동안 새로운 RTprop이 측정되지 않으면 강제로 ProbeRTT 단계로 이동한다. 기존의 BBRv1 알고리즘에서는 ProbeRTT 단계 진입 시 CWND를 4개의 패킷으로 설정했다. 이로 인해 손실 기반 혼잡 제어 알고리즘을 적용한 플로우와 처리량 변동 문제가 발생했다. 해당 문제를 해결하기 위해 BBRv2 알고리즘에서는 CWND를 BDP/2로 감소시켰다.

Ⅳ. F-BBRv2 알고리즘

본 장에서는 BBRv2와 CUBIC **플로우간**의 대역폭 공정성 문제를 해결하기 위한 F-BBRv2 알고리즘을 제안한다. F-BBRv2는 RTprop의 변화를 활용하여 CUBIC 알고리즘을 사용하는 플로우를 감지하고 pacing_gain 파라미터를 조정하여 기존 BBRv2와 CUBIC **플로우간**의 대역폭 공정성 문제를 해결한다.

F-BBRv2 알고리즘은 BBRv2 알고리즘에 두 가지 연산을 추가하여 대역폭 공정성 문제점을 개선한 알고리즘이다.

첫번째 연산은 한정된 대역폭에서 CUBIC 알고리즘을 적용한 플로우를 감지하는 방법이다. <그림 4>에서 볼 수 있듯이 BBRv2 알고리즘을 적용한 플로우들이 한정된 대역폭을 공유할 때 RTprop의 변동률이 적고 1.2배 이상 상승하지 않는 것으로 나타났다. 하지만, BBRv2와 CUBIC 플로우가 대역폭을 공유할 때는 RTprop가 급격히 상승하고 변동성이 커지는 것을 확인할 수 있다. 이는 CUBIC 알고리즘이 손실 기반 혼잡 제어 알고리즘이기 때문에 버퍼 오버플로우로 인해 패킷 손실이 발생하기 전까지 전송 속도를 늦추지 않기 때문이다. 이에 따라 병목 버퍼에는 최대한 많은 큐가 생성되어 큐잉 지연이 증가하고, 이로 인해 RTprop 값이 상승하게 된다. 따라서, F-BBRv2 알고리즘은 이를 이용하여 RTprop의 상승 임계 값을 1.2로 정의하고 RTprop 값의 변동성을 이용하여 CUBIC 알고리즘을 적용한 플로우를 감지한다.

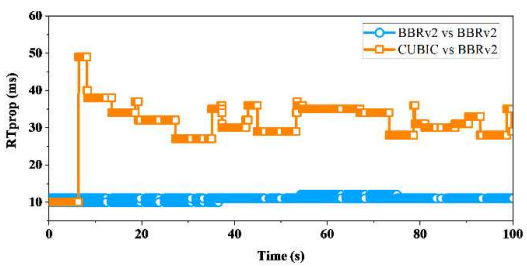


그림 4. RTprop 변화
Fig. 4. RTprop variation.

두번째 연산은 감지한 알고리즘에 따라 전송 속도를 조정하는 동작이다. 기존의 BBRv2 알고리즘은 다른 플로우와 대역폭 공정성을 향상하기 위해 식(1)에 따라 inflight_hi에 15%의 kHeadroom 공간을 남겨둔다. 그러나, 다른 플로우와 대역폭 공정성 문제를 개선하기 위해 적용한 kHeadroom은 병목 버퍼 크기가 클수록 BBRv2와 CUBIC **플로우간**의 대역폭 공정성 문제를 심화시킨다. 이를 개선하기 위해 F-BBRv2 알고리즘은 CUBIC 알고리즘을 적용한 플로우가 감지되면 전송 속도를 15% 증가시켜 보다 공격적으로 공유 대역폭을 점유하도록 설계했다. 또한, F-BBRv2 알고리즘은 병목 버퍼에 생긴 잉여 큐로 인해 상승하는 RTprop 값을 기준으로 추가 전송 속도 상승을 계산하기 때문에 기존의 모델링 기반 BBR 알고리즘과의 대역폭 공유에서 성능 저하를 야기하지 않고 손실 기반 혼잡 제어 알고리즘만을 감지하여 대역폭 공정성 문제점을 개선하도록 설계했다.

<알고리즘 1>은 CUBIC 플로우를 감지하고 전송 속도를 조절하는 F-BBRv2 알고리즘의 과정을 간단한 의사 코드로 보여준다.

$RTprop_{current}$는 현재 측정된 RTprop 값을 의미하며, RTprop는 연결 후 처음 측정된 RTprop 값을 의미한다. $RTprop_{current}$가 미리 정해 놓은 RTprop 상승 임계 값을 곱한 값보다 작으면, 전송에 관여된 모든 파라미터는 BBRv2 알고리즘과 동일하게 적용된다. 그러나 (RTprop × 상승 임계 값)을 초과하면, CUBIC 알고리즘을 적용한 플로우가 같은 대역폭을 공유함을 감지한다. CUBIC 알고리즘을 적용한 플로우를 감지한 후, 전송 단계 중 안정적으로 많은 양의 데이터를 전송하는 ProbeBW:Cruise 단계에서 전송 속도 파라미터인 pacing_gain을 15% 높인다.

알고리즘 1. F-BBRv2 알고리즘 동작
Algorithm 1. F-BBRv2 algorithm operation.

Algorithm 1: Adaptive pacing_gain method

Initialization
  $RTprop = First\ RTprop\ value\ after\ connection$

case $BBRv2 \rightarrow mode = BBRv2\_STARTUP$ **then**
  $pacing\_gain = 2/ln2$
case $BBRv2 \rightarrow mode = BBRv2\_DRAIN$ **then**
  $pacing\_gain = 0.75$

case $BBRv2 \rightarrow mode = BBRv2\_PROBE:BW$ **then**
  If $BBRv2 \rightarrow BBRv2\ PROBE:CRUISE$ **then**
    If $RTprop_{current} > \alpha * RTprop$ **then**
      $pacing\_gain = 1.15$
    else
      $pacing\_gain = 1$
  else
    $pacing\_gain = bbr \rightarrow cycle$

case $BBRv2 \rightarrow mode = BBRv2\_PROBE:RTT$ **then**
  $pacing\_gain = 1$

V. 실험 결과

본 장에서는 F-BBRv2 알고리즘의 성능을 평가하기 위해 단일 경로와 다중 경로의 시나리오를 구성하고, NS-3를 사용하여 실험을 수행했다. <표 2>는 실험에 사용된 네트워크 파라미터 매개변수의 설정 값을 나타낸다. 각 시나리오의 목적에 따라 특정 네트워크 매개변수를 변경하였고, 해당 매개변수의 변화와 시나리오 따른 F-BBRv2 알고리즘의 성능을 평가한다.

표 2. 각 시나리오에 대한 실험 매개변수
Table 2. Experimental parameters for each scenario.

	Scenario 1	Scenario 2
Access bandwidth	50 Mbps	
Bottleneck bandwidth	50 Mbps	
RTT	40 ms	
Bottleneck buffer size	4 BDP	0.5, 1, 2, 4, 7 BDP
Packet loss rate	0, 0.5%	0%
Simulation time	150 Seconds	

5.1 시나리오 1: 단일 경로

F-BBRv2 알고리즘은 다중 경로에서 CUBIC 알고리즘을 사용하는 플로우와 대역폭 공정성을 개선하기 위한 알고리즘이다. 따라서, 단일 경로에서도 BBRv2 알고리즘과 비교하여 성능이 저하되지 않아야 한다. 이를 위해 <그림 5>와 같은 단일 경로 시나리오에서 실험을 진행했다.

<표 3 (a)>는 무작위 패킷 손실이 없는 상황에서 BBRv2와 F-BBRv2이 적용된 플로우의 평균 처리량을 보여준다. BBRv2와 F-BBRv2 플로우는 모두 47.6 Mbps의 동일한 처리량을 보이며, <표 3 (b)>에서 0.5%의 무작위 패킷 손실이 발생할 때도 두 알고리즘의 처리량이 42.2 Mbps로 동일하다는 것을 확인할 수 있다. 기존의 BBRv2 알고리즘과 마찬가지로 두 알고리즘 모두 평균 처리량이 감소하였지만 BBRv1 알고리즘과 비교하면 동일하게 유지되었다.

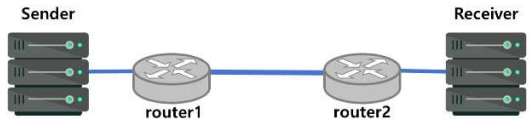


그림 5. 시나리오 1 구성도
Fig. 5. Scenario 1.

표 3. Packet loss rate에 따른 BBR 알고리즘들의 평균 처리량
Table 3. Average throughput of BBR algorithms with respect to packet loss rate.

(a) Random packet loss = 0%

Congestion control algorithm	Average throughput
BBRv1	42. 9 Mbps
BBRv2	47. 6 Mbps
F-BBRv2	47. 6Mbps

(b) Random packet loss = 0.5%

Congestion control algorithm	Average throughput
BBRv1	40. 2 Mbps
BBRv2	42. 2 Mbps
F-BBRv2	42. 2 Mbps

5.2 시나리오 2: CUBIC 플로우와 공존

BBR과 CUBIC 플로우가 함께 대역폭을 공유할 때, 대역폭 공정성을 평가하기 위해 다중 경로 시나리오를 구성했다. 이를 위해 <그림 6>과 같은 시나리오 구성도로 실험을 수행했다.

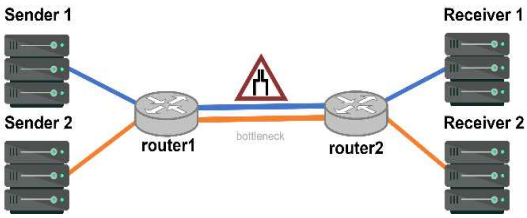
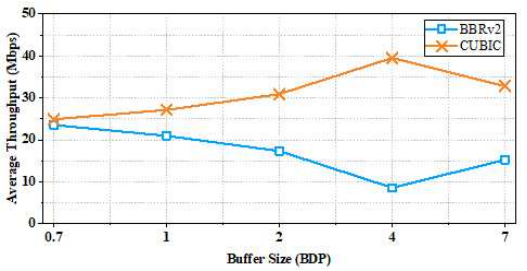


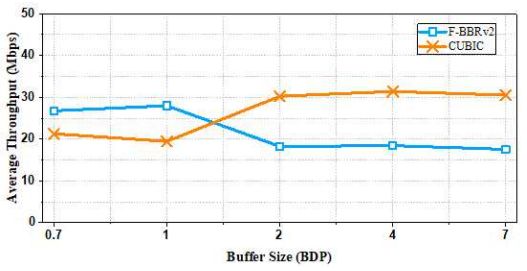
그림 6. 시나리오 2 구성도
Fig. 6. Scenario 2.

대역폭 공정성 평가에서는 병목 버퍼 크기가 중요한 역할을 한다. 따라서, 병목 버퍼 크기가 0.5 ~ 7 BDP까지 변화하는 여러 실험을 진행했다. 본 실험에서는 Sender 1은 각 실험마다 세 가지 버전의 BBR 알고리즘을 차례로 사용하였고, Sender 2는 CUBIC 알고리즘을 고정적으로 사용했다.

<그림 7>는 대역폭이 50 Mbps로 한정된 상황에서 버퍼 크기에 따른 각 플로우의 평균 처리량을 보여주고 있다. 그 중 <그림 7(a)>에서는 병목 버퍼의 크기가 1 BDP일 때, 병목 버퍼의 잦은 오버 플로우로 인해, BBRv2 플로우에서 패킷 손실이 발생하게 된다. 이는 BBRv2 알고리즘에서 정의한 패킷 손실률과 ECN 임계 값을 초과하게 되며, 이로 인해 BBRv2 플로우는 inflight data 양을 제한하여 CUBIC 플로우와 평균 처리량 격차는 6.2 Mbps 이다. 하지만, 병목 버퍼 크기가 2 BDP 이상일 때는 BBRv2 플로우는 CUBIC 플로우로 인한 패킷 손실로 inflight_hi가 설정되고 이로 인해 BBRv2 플로우의 전송 가능한 inflight data의 증가가 제한된다. 따라서 BBRv2는 시간이 흐름에 따라 대역폭을 예측할 수 없게 된다. 그 결과, 병목 버퍼 크기가 4 BDP일 때, 두 플로우의 평균 처리량 격차는 31.5 Mbps로 크게 벌어지게 된다.



(a) BBRv2 vs CUBIC



(b) F-BBRv2 vs CUBIC

그림 7. 병목 버퍼 크기에 따른 플로우의 평균 처리량
Fig. 7. Average throughput according to bottleneck buffer size.

<그림 7(b)>는 병목 버퍼 크기가 2 BDP 이하일 때 F-BBRv2와 CUBIC 플로우의 평균 처리량을 나타내며, 두 플로우의 평균 처리량 차이는 7 Mbps이다. 기존의 BBRv2 알고리즘과 비교하면, F-BBRv2 알고리즘도 패킷 손실과 ECN의 임계 값을 적용하여 CUBIC 플로우와 평균 처리량 차이가 0.8 Mbps로 동일하다.

병목 버퍼 크기가 4 BDP일때 F-BBRv2와 CUBIC 플로우의 평균 처리량 차이는 12 Mbps, 기존의 BBRv2와 CUBIC 플로우의 평균 처리량 격차가 19 Mbps로 약 35%의 감소를 보인다. 이는 F-BBRv2 알고리즘이 CUBIC 플로우를 감지하고 전송 속도를 높여 대역폭을 공격적으로 차지하기 때문이다.

5.3 플로우간의 대역폭 공정성

본 장에서는 대역폭 공정성을 수치적으로 평가하기 위해 Jain Fairness Index (JFI)를 도입했다. JFI는 식 (2)와 같이 계산되며 식 (2)에서 x_i는 각 플로우의 평균 처리량을 나타낸다. JFI 값은 0.5부터 1.0사이의 값을 가지며, 1.0에 가까울수록 대역폭 공정성이 좋은 것을 나타낸다[18].

$$F = \frac{(x1 + x2)^2}{2(x_1^2 + x_2^2)} \quad (2)$$

<그림 8>은 <그림 6>과 같은 다중 경로 시나리오에서, 150초 동안 버퍼 크기에 따른 각 플로우의 평균 처리량을 측정하여 JFI 그래프로 나타낸 것이다.

<그림 8>에서 병목 버퍼 크기가 0.7 BDP일 때, BBRv1 알고리즘을 사용한 플로우는 JFI 값이 0.74로 대역폭 공정성이 낮은 것으로 나타났다. 그러나, BBRv2와 F-BBRv2 플로우는 JFI 값이 0.98 이상으로 매우 높은 대역폭 공정성을 보여준다. 하지만, 병목 버퍼 크기가 4 BDP일 때 BBRv2 플로우의 JFI 값은 0.7로 매우 낮은 공정성을 나타내며, F-BBRv2 알고리즘을 사용한 플로우의 JFI 값은 0.92로 높은 대역폭 공정성을 보여주며, 기존의 BBRv2 알고리즘의 대역폭 공정성 문제가 개선된 것을 보여준다.

F-BBRv2 알고리즘을 사용한 플로우는 BBRv1 플

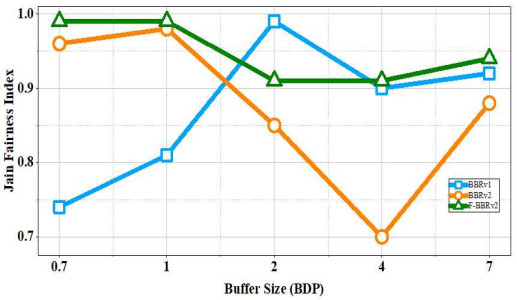


그림 8. 버퍼 크기에 따른 Jain fairness index
Fig. 8. Jain fairness index according to buffer size.

로우와 비교하여 대역폭 공정성이 18% ~ 25% 향상되었다. 또한, BBRv2 알고리즘을 사용한 플로우와 비교해 병목 버퍼 크기가 2 BDP 이하인 경우에도 성능 저하가 없었으며, 2 BDP 이상일 때 대역폭 공정성은 최소 6%에서 최대 21%까지 증가했다.

Ⅵ. 결 론

본 논문에서는 BBRv2 알고리즘의 대역폭 공정성 문제를 해결하기 위해 F-BBRv2 알고리즘을 제안했다. BBRv2와 CUBIC 플로우는 각각 서로 다른 동작 특성을 가지고 있기 때문에, 병목 버퍼 크기가 큰 경우 CUBIC 플로우의 잦은 패킷 손실로 인해 CUBIC 플로우가 잉여 버퍼 공간을 차지하면서 두 플로우간 대역폭 공정성 문제가 발생하는 것으로 나타났다. 이러한 문제를 해결하기 위해, RTprop의 변화에 임계값을 설정하고 이를 초과할 때 F-BBRv2 알고리즘의 전송 속도를 조절하여 대역폭 공정성 문제를 개선했다.

실험을 위해 NS-3를 이용하여 다중 경로 시나리오를 구성하였고, 제안된 F-BBRv2 알고리즘의 성능을 확인했다. 실험 결과, F-BBRv2 알고리즘은 병목 버퍼 크기와 관계없이 Jain Fairness Index가 0.9 이상으로 나타나며, 기존 BBRv2 알고리즘 대비 최대 21%의 성능 개선을 보였다.

References

[1] J. Postel, "*Transmission control protocol*," RFC 0793, IETF, 1981.

[2] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, "Host-to-Host congestion control for TCP," in *IEEE Commun. Surv. & Tuts.*, vol. 12, no. 3, pp. 304-342, Third Quarter 2010. (https://doi.org/10.1109/SURV.2010.042710.00114)

[3] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314-329, Aug. 1988. (https://doi.org/10.1145/205447.205462)

[4] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.,* vol. 42, no. 5, pp. 64-74, 2008. (https://doi.org/10.1145/1400097.1400105)

[5] H. Jiang, et al., "When machine learning meets congestion control: A survey and comparison," *Comput. Netw.*, vol. 192, 2021. (https://doi.org/10.1016/j.comnet.2021.108033)

[6] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465-1480, Oct. 1995. (https://doi.org/10.1109/49.464716)

[7] J. Gettys and K. Nichols, "Bufferbloat: Darkbuffers in the internet," *ACM Queue*, vol. 9, no. 11, 2011. (https://doi.org/10.1109/mic.2011.56)

[8] N. Cardwell, et al., "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 59-66, 2017. (https://doi.org/10.1145/3009824)

[9] 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.

[10] M. Alizadeh, et al., "Data center tcp (dctcp)," in *Proc. ACM SIGCOMM*, 2010. (https://doi.org/10.1145/1851182.1851192)

[11] E. F. Kfoury, et al., "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband," *Comput. Commun.*, vol. 161, pp. 212-224, 2020. (https://doi.org/10.1016/j.comcom.2020.07.018)

[12] Y. J. Song, et al., "Understanding of BBRv2: Evaluation and comparison with BBRv1

congestion control algorithm," in *IEEE Access*, vol. 9, pp. 37131-37145, 2021, doi: 10.1109/ACCESS.2021.3061696. (https://doi.org/10.1109/access.2021.3061696)

[13] 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.

[14] G.-H. Kim, et al., "Standardization and research trends of BBR congestion control algorithm," *J. KICS*, vol. 44, no. 9, pp. 1713-1722, 2019. (10.7840/kics.2019.44.9.1713)

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

[16] 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.

[17] K. Sasaki, et al., "TCP fairness among modern TCP congestion control algorithms including TCP BBR," in *Proc. IEEE 7th Int. Conf. CloudNet*, Oct. 2018. (https://doi.org/10.1109/cloudnet.2018.8549505)

[18] R. Jain, D. Chiu, and W. Hawe, "*A quantitative measure of fairness and discrimination for resource allocation in shared computer system*," DEC-TR-301 Tech. Rep., Sep. 1984. (https://doi.org/10.48550/arXiv.cs/9809099)

조 효 섭 (Hyo-Seop Cho)

2021년 : 대구대학교 전자전기공학부 졸업

2023년 : 경북대학교 전자 공학부 석사

<관심분야> 차세대 전송 계층 프로토콜, 차세대 이동 네트워크, TCP 혼잡제어, MPTCP 혼잡제어, QUIC

[ORCID:0000-0001-6201-7551]

조 유 제 (You-Ze Cho)

1982년 : 서울대학교 전자공학과 학사

1983년 : 한국과학기술원 전기전자공학 석사

1988년 : 한국과학기술원 전기전자공학 박사

1989년~현재 : 경북대학교 전자공학부 교수

1992년~1994년 : Univ. of Toronto in Canada, 방문교수

2002년~2003년 : NIST(미국국립표준연구소) 객원 연구원

2017년: 한국통신학회 회장

<관심분야> 차세대 이동 네트워크, 무선 애드혹 네트워크, 이동성 관리 기술, 차세대 전송 계층 프로토콜

[ORCID:0000-0001-9427-4229]