한국정보전자통신기술학회논문지(jkiiect)'18-06, Vol.11 No.3
논문 18-11-03-259 http://dx.doi.org/10.17661/jkiiect.2018.11.3.259

Bufferbloat 환경에서 MPTCP 성능 개선을 위한 전송 상태 모니터링 방법

정일형*, 이재용**, 김병철***

Transmission status monitoring method for improving the performance of MPTCP in Bufferbloat environment

Il Hyung Jung*, Jae Yong Lee**, Byung Chul Kim***

요 약 Multipath TCP (MPTCP)는 다중 경로를 통해 데이터를 전송하므로 향상된 네트워크 성능을 기대할 수 있으나, 예고 없이 전송 경로에서 발생되는 Bufferbloat은 해당 경로의 성능뿐만 아니라 다른 경로의 성능 역시 저하시켜 단일 TCP 보다 오히려 좋지 않은 성능이 발생한다. 이와 같은 Bufferbloat 문제를 해결하기 위해 본 논문에서는 송신단에서 MPTCP 레벨의 전송 상태 모니터링 방법과 HoL 패킷 재전송 알고리즘을 제안하였다. 제안한 알고리즘은 송신단 독자적으로 Bufferbloat 감지를 가능하게 하며, 전송 상태 모니터링 버퍼에서 HoL 패킷을 식별하여 정상 경로로 재전송함으로 HoL Blocking문제를 해결할 수 있도록 하였다. NS-3기반 시뮬레이션 결과 제안한 알고리즘은 기존 MPTCP 대비 전송 성능은 최대 22.8%까지 향상되었고, 송신 버퍼 및 수신 버퍼에 대기하는 평균 패킷 수는 기존 대비 각각 44.3%와 9.2% 수준으로 감소되었음을 확인하였다.

Abstract Multipath TCP (MPTCP) can be expected to provide improved network performance because it transmits data through multiple paths. However, Bufferbloat, which unexpectedly occurs in the transmission path, degrades not only the performance of the corresponding path but also the performance of other paths, so that the performance is worse than that of a single TCP. In this paper, we propose the transmission status monitoring method at the sender's MPTCP level and also HoL packets retransmission algorithm in order to solve the Bufferbloat problem. The proposed algorithm enables Bufferbloat detection by the sender side independently, and it can resolve the HoL blocking problem by identifying the HoL packet in the proposed transmission status monitoring buffer and retransmitting it to the normal path. Simulation results based on NS-3 show that the proposed algorithm achieves the improved throughput performance up to 22.8% compared to the existing MPTCP, and the average number of queued packets in the sender and receiver's buffers is decreased to 44.3% and 9.2%, respectively.

Key Words : Bufferbloat, Congestion control, Head-of-Line blocking, Multipath TCP, Transmission status monitoring

1. 서론

TCP (Transmission Control Protocol)는 오늘날 가장 널리 이용되는 전송 계층 프로토콜로, 신뢰성 있는 종단 간 통신 방식을 제공한다. 오늘날 인터넷 환경은 TCP가 규정된 초기와 많은 부분에서 급속하게 변하고 있으며, 특히 단일 인터페이스를 장착했던 유무선 단말기들의 WiFi

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education(NRF-2017R1D1A1B03028766), (NRF-2015R1D1A3A01018044).
*Korea Aerospace Research Institute
**Corresponding Author : Dept. of Radio and InfoComm Engineering, Chungnam National University (jyl@cnu.ac.kr)
***Dept. of Radio and InfoComm Engineering, Chungnam National University
Received May 14, 2018 Revised May 21, 2018 Accepted May 29, 2018

와 3G (또는 4G)와 같은 다중 인터페이스 장착은 일반화 된지 오래이다. 따라서 기존 TCP에서 한걸음 더 나아가 다중 인터페이스를 효과적으로 지원하기 위해 SCTP [1]가 주목을 받았으나, 기존 응용 프로그램, NAT (Network Address Translation) 및 방화벽과의 호환성 문제에 봉착하여 실제적인 적용에는 많은 한계에 부딪쳤다.

이러한 호환성 문제를 해결하기 위해 기존 TCP 기반 MPTCP [2]가 제안되었으며, IETF (Internet Engineering Task Force)에 의해 2013년 표준화되었다. MPTCP의 각 서브플로우는 기존 TCP를 이용하기 때문에 NAT와 방화벽과 같은 기존 인프라를 통해 데이터 전송이 가능할 뿐더러 TCP의 기존 Option과 응용 프로그램을 별도 변경할 필요 없이 그대로 사용할 수 있는 매우 큰 장점이 있다. 또 데이터를 MPTCP의 다중 경로를 통해 전송하므로 높은 전송 속도와 함께 안정화된 데이터 전송도 가능하다.

MPTCP는 경로 간 동일한 성능일 때 다중 경로를 통한 장점을 얻을 수 있지만, 경로 간에 성능 차이가 있거나 [3-7] 한쪽 경로가 전송 역할을 정상적으로 하지 못할 경우 [8] 심각한 성능 저하 현상이 발생한다. 특히 경로 간 지연 시간 차이가 발생할 때 성능 저하 현상이 심화되며, 먼저 수신단에 도착한 패킷이 늦게 도착한 패킷을 기다리는 HoL (Head of Line) Blocking [9] 현상에 주로 기인한다. 즉, MPTCP 레벨 시퀀스 번호인 DSN (Data Sequence Number)이 순서대로 조립될 때까지 빠른 경로의 패킷이 장시간 수신 버퍼에 머물러 재정렬 절차를 거치며, 심화될 경우 수신버퍼 넘침 현상까지 발생하여 데이터가 손실 된다. Bufferbloat [10-11]도 주목해야 할 현상으로서, 용량이 늘어난 중간 노드에 데이터가 갇혀 전송 시간이 급격히 증가되는 현상이다. MPTCP 경로는 기존 TCP를 이용하므로 MPTCP에서도 Bufferbloat이 발생되며, 전송 경로에서 증가된 지연 시간은 궁극적으로 수신단에서 HoL Blocking을 유발시키므로 MPTCP의 전체 성능 저하를 일으킨다.

본 논문에서는 Bufferbloat으로 인한 MPTCP의 성능 저하를 방지하기 위해 송신 버퍼를 모니터링하여 Bufferbloat을 판별하고 재전송을 수행하는 알고리즘을 제안하였다. Bufferbloat 발생에 따라 송신 버퍼도 영향을 받음을 본 논문에서 발견하였고, 이를 바탕으로 MPTCP 레벨에서 전송 상태를 모니터링 할 수 있는 기능을 제안함으로서 MPTCP 전체 전송 상태를 상위에서 감시할 수 있게 하였다. 따라서 서브플로우 또는 수신측으로부터 별도 정보를 받지 않고도 문제 경로를 판별하고 HoL 패킷을 찾아내 정상 경로로 재전송하는 알고리즘을 본 논문에서 제안하였다.

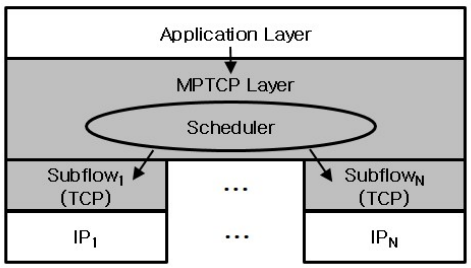


그림 1. MPTCP 프로토콜 스택
Fig. 1. MPTCP protocol stack

본 논문의 2장에서는 MPTCP 프로토콜 구조와 한계점에 대해서 기술하고, HoL Blocking과 Bufferbloat을 해결하기 위한 기존 연구 내용을 살펴본다. 3장에서는 제안한 전송 상태 모니터링 기반 Bufferbloat 감지 방법 및 Bufferbloat을 해결하기 위한 알고리즘을 소개한다. 4장에서는 제안한 알고리즘 검증 내용이 기술되었으며, NS-3 시뮬레이션에서 Bufferbloat 환경을 통해 기존 MPTCP와 비교 분석 하였다. 마지막으로 5장에서는 본 논문의 결론과 향후 연구 계획을 기술한다.

2. 관련 연구

2.1 Multipath TCP

MPTCP는 Fig. 1과 같이 MPTCP 계층과 서브플로우 계층으로 구성되어 있다 [2]. MPTCP 계층에서는 각 서브플로우의 연결 동작을 제어하며, 응용 계층으로부터 내려온 데이터를 MPTCP 스케쥴러를 이용하여 각 서브플로우로 할당하는 역할을 담당하고 있다. 각 서브플로우로 데이터가 할당 될 때 MPTCP 레벨의 시퀀스 번호인 DSN이 부여되고, 서브플로우 레벨에서 할당되는 기존 TCP 시퀀스 번호와 함께 전송된다. 그리고 수신단에서는 TCP 시퀀스 번호를 이용하여 각 서브플로우 별 시퀀스 정렬 절차를 거친 후, DSN을 이용하여 MPTCP 레벨 전체 시퀀스 정렬 절차를 수행한다. 각 서브플로우에서는 해당 IP 위에 TCP

세션으로 동작하며, 각 서브플로우 별 패킷 전송 및 혼잡 제어를 독립적으로 담당한다.

따라서 MPTCP 레벨에서는 각 서브플로우별 네트워크 상태에 따라 데이터 할당을 주로 수행한다고 할 수 있다. MPTCP 스케쥴러가 각 서브플로우의 RTT (Round Trip Time)등 각 네트워크 정보를 이용하여 데이터를 분배하지만, 전송된 이후 정상적인 ACK를 받았는지 또 받지 못했다면 얼마만큼 ACK가 지연되었는지, 어떤 경로에서 문제가 생겼는지 등 전송 이후 결과는 MPTCP 레벨에서 알 수 없다. 따라서 본 논문에서는 MPTCP 레벨의 전송 상태 모니터링 방법을 제안하여 전송 상태 감시 기능을 강화하고자 하며, 각 서브플로우의 전송 중 발생한 문제점을 수신단 도움 없이 송신단에서 손쉽게 해결 하고자 한다.

2.2 이종망에서 MPTCP 성능

앞서 서론에서 기술한 바와 같이 MPTCP는 서브플로우 간 비슷한 성능을 가질 때 MPTCP의 장점을 충분히 살릴 수 있지만, 경로 간 다른 특성을 가질 때, 특히 수신단에 도착하는 시간 차이가 벌어질 때 HoL Blocking이 발생하여 단일 TCP보다 낮은 성능을 가지게 된다. 이러한 문제를 해결하기 위해 Lowest RTT 스케쥴러[3]는 가장 낮은 RTT를 가진 서브플로우로 가용 윈도우가 없어질 때까지 데이터를 보내고, 전송할 윈도우가 없어지면 그 다음 낮은 RTT를 가진 서브플로우로 전송함으로서 빠른 경로로 데이터를 우선적으로 전송하였다. [4]는 전송할 데이터가 최대한 빨리 도착하는 경로를 예측하는 스케쥴러를 제안함으로서 지연 경로로 인한 영향을 최소화 하도록 하였고, [5]에서는 재전송 경로의 특성에 따른 성능을 분석함으로서 어떤 경로로 재전송을 수행해야 하는지 제시하였다. [6]은 수신 버퍼 사이즈를 최소화하면서 MPTCP 성능을 향상시키기 위한 방법을 제시하였고, [7]은 2개의 서브플로우가 Chunk의 시작 지점과 끝 지점부터 각각 독립적으로 전송하게 함으로서 전체 데이터 전송 시간을 줄이기 위한 방법을 제시하였다. 위와 같은 선행 연구 [3, 4, 6, 7]는 MPTCP에서 다른 지연 시간을 가지는 경로가 존재 할 때, 스케쥴러 및 수신 버퍼를 통해 성능을 높이기 위한 내용으로서, 전송 중 동적으로 발생하는 Bufferbloat으로 인하여 경로의 지연 시간이 극적으로 변하는 문제를 직접적으로 해결할 수 없다. 또한 2.1절에서 설명한 바와 같이 MPTCP 스케쥴러는 이미 전송한 데이터의 상태를 알 수 없어 문제가 되는 HoL 패킷을 식별할 수 없으므로, 재전송을 통한 HoL Blocking 해결에는 어려움이 있다고 할 수 있다.

2.3 MPTCP에서 Bufferbloat 현상

반도체 기술 발달에 따른 메모리 가격 하락과 함께 트래픽 증가로 인한 데이터 손실을 방지하기 위해 유무선 경로의 중간 노드에 증가된 메모리 용량은, 전체 전송 경로의 메모리 용량을 과도하게 증가시켜 Bufferbloat 문제[10-11]를 발생시켰다. 이와 같은 메모리 용량 증가는 손실 기반 TCP 혼잡 제어를 무력화시키고 혼잡 윈도우 사이즈를 지속적으로 증가하게 함으로서, 혼잡 상황에서도 전송 경로로 데이터 유입이 허용된다. 따라서 혼잡 상황에 전송 경로로 유입된 데이터는 중간 노드의 증가된 메모리에 갇혀 장시간 대기 하게 되며, 이로 인하여 10초 이상의 RTT도 쉽게 관찰 되고 있다 [12].

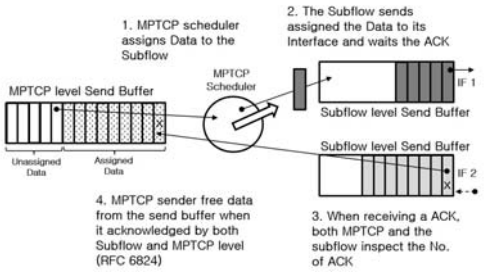


그림 2. MPTCP 송신측에서 데이터 전송 및 ACK 절차
Fig. 2. Data transmission and ACK procedures at MPTCP sender

Bufferbloat을 해결하기 위해서 [3]에서 제안한 BM (Bufferbloat Mitigation)은 sRTT (Smoothed RTT)가 증가되면 서브플로우의 혼잡 윈도우를 줄이도록 하여 Bufferbloat경로로 데이터 유입을 최소화하도록 하였다. 하지만 Bufferbloat 경로로 완전히 데이터를 억제하지 않으므로 Bufferbloat 경로로 전송된 일부 데이터에 의해 지속적으로 HoL Blocking이 발생하여 성능 저하 현상이 일어날 수 있다. [12]는 송신측의 혼잡 윈도우 예측을 통해 해결하고자 하였으나, 예측의 부정확성으로 성능이 오히려 안 좋아지는 단점을 가지고 있다. [13]은 sRTT를 모니터링 하여 지연 경로로 데이터 전송을 억제하는 알고리즘을 제

안하였으나, 데이터 전송이 억제되면 sRTT 적용도 늦어지므로 회복 시점을 파악하기까지는 너무 많은 시간이 소요되는 한계가 있다. [8]은 여러 단계의 경로 유효 판단 절차를 거쳐 의심 경로로 인한 성능 감쇠를 최소화하는 방법을 제안하였으나 송수신단에서 복잡한 판단 절차가 동반되며, 억제된 경로의 활성화 방법도 추가로 필요하다. [14]는 수신 버퍼 모니터링을 통해 Bufferbloat 문제를 해결하였으나, 수신단으로부터 Bufferbloat 여부 및 HoL 패킷 정보를 수신해야 하는 절차가 수반되어야 한다. 앞서 언급하였듯이 Bufferbloat 현상은 전송 경로에서 동적으로 생성 및 소멸되므로, 이종망에서 MPTCP의 성능 저하를 해결하기 위한 지금까지의 기존 방법으로는 한계가 있다고 할 수 있다. 또, Bufferbloat기간 동안 큰 용량의 메모리에 갇혀 전송 지연되고 있는 패킷이 다른 경로로 전달되지 못할 경우 HoL 패킷이 수신단에 도착할 때까지 HoL Blocking을 유발시켜 모든 경로에서 심각한 성능 저하를 일으킨다. 따라서 다른 경로로 HoL 패킷의 재전송은 Bufferbloat을 신속하게 해결하기 위해 필수적이다고 할 수 있다. Bufferbloat을 해결하기 위해 해당 경로 억제할 경우, Bufferbloat 경로의 회복시점을 최대한 빨리 인지해야 하며, 이는 MPTCP의 경로 자원 활용도와 밀접한 관계가 있다고 할 수 있다. 수신단에서 Buffbloat을 인지하여 송신단으로 전송해주는 방법[8, 14]은 중간에 해당 정보가 손실 될 수 있고 정보 전달 시간이 발생하므로 더 간단하고 신뢰성 있는 해결 방법이 필요하다. 따라서 본 논문에서는 위에서 제시된 Bufferbloat 선행 연구[8, 10-14]를 면밀히 검토하여 송신단에서 Bufferbloat 문제를 해결하기 위한 방법을 제시하고자 한다.

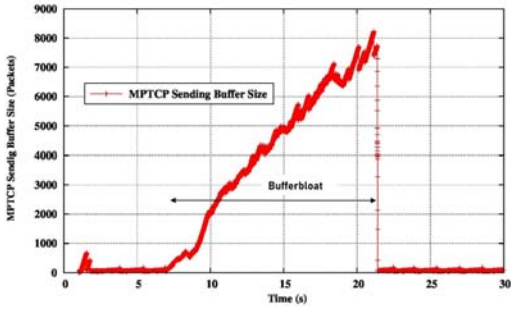


그림 3. Bufferbloat이 발생할 때 MPTCP 송신 버퍼에 있는 패킷 수 변화

Fig. 3. Variations in the number of packets in MPTCP sending buffer when bufferbloat occurs

3. 송신 버퍼 모니터링 기반 Bufferbloat 판별 및 재전송 알고리즘

3.1 Bufferbloat 발생에 따른 MPTCP 레벨 송신 버퍼 크기 변화

MPTCP 송신단은 Fig. 2와 같이 크게 MPTCP 레벨 송신 버퍼, 스케쥴러 그리고 각 서브플로우별 송신 버퍼로 구성되어 있다 [2]. 응용 계층에서 전송될 데이터는 MPTCP 레벨 송신 버퍼에 저장되고, MPTCP 스케줄러정책에 따라 각 서브플로우에 데이터가 할당 된다. 할당된 데이터는 DSN과 서브플로우 시퀀스 번호 및 전송 인터페이스 정보 등이 추가되어 전송되며, 두 레벨의 ACK가 올 때 까지 MPTCP 및 서브플로우 레벨의 버퍼에 저장된다. 즉 송신단에서 ACK를 수신했을 때, 서브플로우 레벨의 ACK 뿐만 아니라 MPTCP 레벨 ACK 번호까지 만족되어야 각각의 송신 버퍼에서 해당 데이터를 삭제할 수 있다고 표준 문서 [2]에서 명시하고 있다.

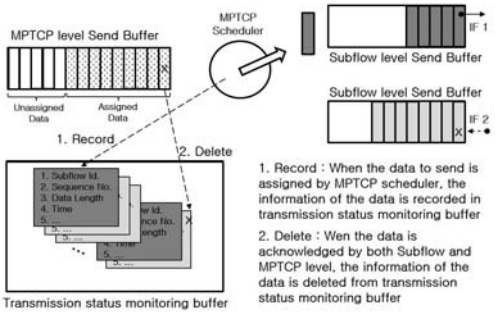


그림 4. 제안된 MPTCP 레벨 전송 상태 모니터링 방법

Fig. 4. Proposed transmission status monitoring method at MPTCP level

본 논문에서는 이와 같은 점에 착안하여 Bufferbloat 현상이 일어났을 때 MPTCP 레벨 송신 버퍼 크기를 모니터링 하였다. NS-3기반 MPTCP 시뮬레이터 [15]를 이용하여 MPTCP 서브플로우를 2개 생성하였고, 경로 중간에 있는 버퍼 크기를 매우 크게 하여 시뮬레이션 시작 6초부터 16까지 발생된 혼잡 트래픽을 통해 Bufferbloat 상황을 만들었다. Bufferbloat이 발생했을 때 MPTCP 송신 버퍼의 크기는 Fig. 3에서 확인 할 수 있다. Bufferbloat 경로로 전송된 패킷은 저장 공간이 매우 큰 중간 노드에서 정

체되고 있기 때문에, 송신단에서 해당 패킷의 ACK를 수신할 수 없다. 따라서 수신하지 못한 Bufferbloat 경로의 ACK로 인하여 송신 버퍼의 데이터는 삭제되지 못하고 쌓여가며, 이러한 증가 추이는 서브플로우 레벨 뿐만 아니라 Fig. 3과 같이 MPTCP 레벨에서도 관찰 된다. 따라서 본 논문에서는 이와 같이 MPTCP 송신 버퍼를 모니터링함으로서 Bufferbloat 상황을 송신단 단독으로 간단하게 인지하고자 한다.

3.2 MPTCP 레벨 전송 상태 모니터링 방법

앞 절에서 설명한 MPTCP 레벨 송신 버퍼 모니터링을 통해 Bufferbloat 상황을 감지할 수 있지만, MPTCP 레벨 송신 버퍼에 있는 데이터는 DSN 및 각 서브플로우의 정보가 포함되지 않았기 때문에, 어떤 서브플로우가 무엇이 문제인지 현재 MPTCP 레벨 송신 버퍼구조로는 식별하기 힘들다. 따라서 본 논문에서는 Fig.4와 같이 MPTCP 레벨에 송신 이력 모니터링 버퍼를 제안하여, MPTCP 레벨에서 Bufferbloat 문제를 해결하고자한다.

제안한 MPTCP 레벨 전송 상태 모니터링 버퍼는 Fig. 4에서 확인할 수 있듯이 스케쥴러가 서브플로우에 데이터가 할당 될 때 해당 데이터의 정보를 생성하며, MPTCP와 서브플로우 레벨의 ACK를 모두 받아 양쪽의 송신 버퍼에서 삭제될 때 해당 데이터 정보를 삭제한다. 데이터가 서브플로우에 할당될 때 전송 상태 모니터링 버퍼에는 서브플로우 식별자(Id), DSN, 데이터 길이 및 할당된 시간 정보 등을 포함하여 메모리에 저장한다. 제안한 전송 상태 모니터링 버퍼에는 데이터가 할당된 시간순으로 순차적으로 저장되어 있기 때문에 맨 앞단은 현재 ACK를 기다리고 있는 데이터의 정보가 위치하고 있다. 따라서 Bufferbloat이 발생했을 때, 전송 상태 모니터링 버퍼의 맨 앞단에 있는 기록되어 있는 데이터 정보를 통해 HoL 패킷의 시퀀스 번호 뿐만 아니라 Bufferbloat이 발생한 인터페이스 정보 등 재전송에 필요한 정보를 쉽게 알아 낼 수 있다. 또 송신 버퍼에 남아 있는 경로별 패킷 개수 등을 통해 재전송 경로를 결정 할 수 있고, ACK 대기 시간 등 문제 해결을 위한 다양한 정보를 얻어낼 수 있다.

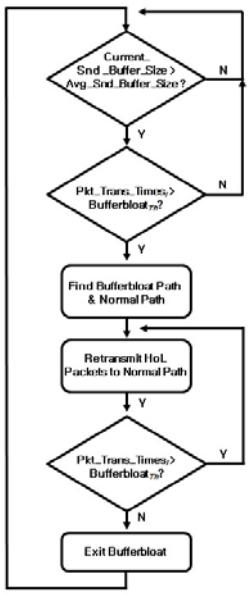


그림 5. 제안된 Bufferbloat 감지 및 재전송 알고리즘
Fig. 5. Proposed Bufferbloat detection and retransmission algorithm

3.3 전송 횟수 예측을 이용한 Bufferbloat 판별 및 재전송 알고리즘

Bufferbloat 발생시 3.1절의 송신 버퍼 크기 변화 및 3.2절의 MPTCP 레벨 전송 상태 모니터링 방법을 이용하여, 구체적인 Bufferbloat 해결 알고리즘을 본 절에서기술한다. 본 논문에서 제안한 알고리즘은 Fig. 5와 같이 현재 MPTCP 송신 버퍼 크기를 모니터링 하여 평균 송신 버퍼 크기보다 증가 했을 때 Bufferbloat 의심 상황으로 인식한다. 이 때 전송 상태 모니터링 버퍼를 통해 서브플로우 i 송신 버퍼에 남아 있는 패킷 수 $No.Pkt_i$를 구할 수 있고, 이를 경로의 최대 전송 윈도우크기 $MaxCwnd_i$로 나누면, 식 (1)과 같이 서브플로우 i의 예상 전송 RTT 횟수 $TransmitTimes_i$를 구할 수 있다. $TransmitTimes_i$가 높다는 것은 송신단에서 ACK를 받지 못하고 대기하는 패킷이 많다는 것이므로, $TransmitTimes_i$가 $Bufferbloat_{Th}$보다 높은 값일 때 Bufferbloat 상황으로 인식 할 수 있다. 그리고 Bufferbloat 판단 값인 $Bufferbloat_{Th}$ 는 Bufferbloat이 발생하지 않았을 때 각 경로의 sRTT비율을

바탕으로 설정된다. 즉 경로간 sRTT 차이가 거의 없을 때는 송신 버퍼에 남아있는 패킷수가 적으므로 $Bufferbloat_{Th}$는 3의 값을 가지며, sRTT가 2배 이상 발생할 때는 ACK 수신에 상대적으로 긴 시간이 소요되어 송신 버퍼에 쌓이는 패킷수가 많으므로 $Bufferbloat_{Th}$는 10까지 비례적으로 증가된다.

$$Transmit\,Times_i = \frac{No.\,Pkt_i}{Max\,Cwnd_i} \quad (1)$$

이와 같은 방법으로 Bufferbloat으로 판단하였을 때, 3.2에서 설명한 바와 같이 송신 버퍼의 앞단에 있는 데이터는 수신단으로부터 ACK를 기다리고 있는 HoL 패킷의 데이터이므로, 전송 상태 모니터링 버퍼를 통해 문제가 되는 서브플로우의 식별자 및 HoL 패킷의 DSN 번호 등 패킷 재전송에 관한 정보를 알아낼 수 있다. 또한 버퍼블롯 경로를 제외하고 2개 이상의 서브플로우가 연결되어 있을 때, 가장 많이 송신 버퍼에 데이터가 남아 있는 서브플로우가 데이터를 가장 많이 전송한 성능 좋은 서브플로우이므로 재전송 경로로 선출된다. 이와 같이 본 논문에서 제안한 HoL 패킷 재전송 방법은 재전송할 HoL 패킷을 별도로 수신단으로부터 통보받지 않고, 전송 상태 모니터링 버퍼를 통해서 인지하여 재전송을 수행하므로, 절차가 매우 간단해 질뿐만 아니라 신속하게 Bufferbloat 상황을 해결 할 수 있다.

Bufferbloat 경로의 가용 여부를 신속히 알아내서 데이터를 전송하는 것은 다중 경로 사용를 사용하는 MPTCP에 매우 중요한 사항이다. 본 논문에서는 $Transmit\,Times_i$가 $Bufferbloat_{Th}$보다 높은 Bufferbloat 상황 동안 문제 경로의 패킷을 정상 경로로 모두 재전송하고 따로 문제 경로에 대한 전송 양을 제약하지 않는 방법을 사용한다. 이와 같은 방법은 정상 경로로 재전송양은 많아 질 수 있지만, Bufferbloat 경로로 데이터를 계속 전송하도록 함으로써 경로의 상태가 회복될 때 sRTT 등을 이용한 별도 판단 절차 없이 바로 문제 경로를 사용할 수 있는 장점이 있다.

4. 성능 분석

4.1 시뮬레이션 환경

표 1. 시뮬레이션에서 Bufferbloat을 발생시키기 위한 주변 트래픽
Table 1. Cross traffic to generate Bufferbloat in the simulation

	Time of occurrence	Max traffic	Average traffic
1st Cross traffic	10 ~ 20 (s)	23 (Mbps)	16 (Mbps)
2nd Cross traffic	35 ~ 43 (s)	25 (Mbps)	17.5 (Mbps)

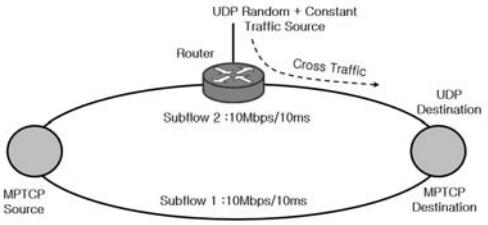
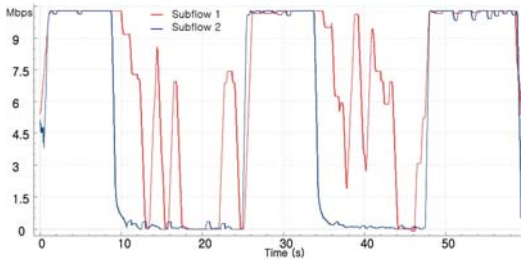
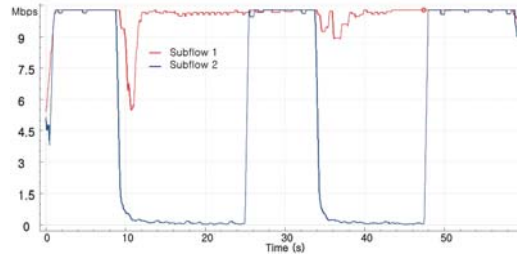


그림 6. NS-3에서 MPTCP 시뮬레이션 구성도
Fig. 6. MPTCP simulation configuration in NS-3

본 논문에서는 제안한 알고리즘을 검증 및 비교하기 위해 NS-3기반 MPTCP 시뮬레이터[15]를 사용하였고, 송신단에 전송 상태 모니터링 버퍼, Bufferbloat 판단 알고리즘 및 재전송 기능을 새로 구현하였다. 망 구성도는 Fig. 6과 같이 2개의 MPTCP 서브플로우를 생성시켰으며, 2개 서브플로우의 대역폭과 지연 시간은 모두 10Mbps에 10ms를 가진다. 서브플로우 2에서 데이터 전송 중 혼잡 상황을 만들기 위해, 주변 트래픽은 Table 1과 같이 랜덤 함수를 사용하여 불규칙하게 생성하였다. 1차 주변 트래픽은 시뮬레이션 시작 후 10초에서 20초까지 발생되며, 2차 주변 트래픽은 35초부터 43초까지 총 8초 동안 UDP 소스에서 라우터를 통해 UDP 목적지까지 흐른다. 따라서 라우터부터 UDP 목적지까지가 서브플로우 2와의 병목 지역으로 작용한다. 주변 트래픽은 평균 16Mbps 이상 발생되므로, 주변 트래픽 발생 시간 동안 서브플로우 2의 트래픽은 중간 라우터의 큰 버퍼에 갇혀 매우 높은 전송 지연을 유발하도록 하였다. 따라서 이러한 구성도는 Bufferbloat이 발생한 시간 동안 경로의 특성을 극적으로 변화시켜 Bufferbloat으로 인한 성능 비교를 가능하게 한다.



(a) Existing MPTCP (Round Robin)



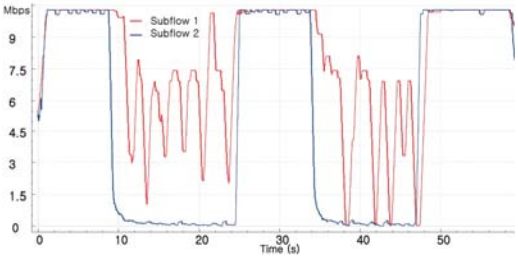
(b) Proposed MPTCP (Round Robin)

그림 7. 기존 MPTCP와 제안된 MPTCP의 성능 비교 (Round Robin)
Fig. 7. Throughput comparison of existing MPTCP and proposed MPTCP (Round Robin)

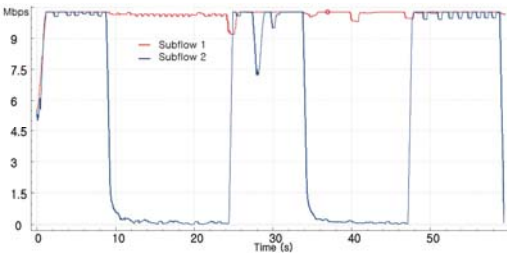
시뮬레이션에서 사용된 혼잡 제어는 MPTCP 경로 간 지연 시간 차이가 발생할 때도 좋은 성능을 발휘하는 LIA [16] 표준을 사용하였다. 스케쥴러는 MPTCP에서 가장 보편적으로 사용하는 Round Robin과 Lowest RTT [3]를 이용하여, Bufferbloat 상황에서 본 논문에서 제안한 알고리즘을 적용하였을 때와 그렇지 않았을 때 각각의 성능이 어떻게 변하는지 비교하였다. MPTCP 트래픽은 대용량으로 60초 동안 발생시켰으며, 수신 버퍼는 3000개까지 패킷을 저장할 수 있도록 설정하였고 모든 패킷의 크기는 1400 Bytes이다.

4.2 실험결과

앞서 설명한 시뮬레이션 환경에서 Bufferbloat이 발생하였을 때, 알고리즘을 적용하지 않은 기존 MPTCP의 Round Robin과 Lowest RTT 스케쥴러 성능은 각각 Fig. 7(a)와 Fig. 8(a)와 같다. 시뮬레이션 시작 후 10 ~ 20초와 35 ~ 43초 동안 발생한 주변 트래픽으로 인해 서브플로우 2 뿐만 아니라 서브플로우 1의 성능도 심각하게 영향 받았음을 본 성능 그래프로 확인 할 수 있다. 이러한 성능 저하는 앞서 설명한 HoL Blocking에 기인하며, 서브플로우 2의 지연 시간으로 인해 서브플로우 1까지 영향받음을 알수 있다. Lowest RTT는 지연 시간이 짧은 경로의 전송할 가용 윈도우가 없을 경우 전송시간이 긴 경로로 전송하므로, 경로별 지연 시간이 다른 환경에서 대용량 트래픽 전송할 경우 Round Robin과 성능이 비슷함[3, 4, 17]을 본 그래프로도 확인할 수 있다. 1차 주변 트래픽은 시뮬레이션 시작 10초부터 10초 동안, 2차는 35초부터 7초 동안 발생하였지만, 중간 라우터에 쌓여 있던 주변 트래픽이 최종 목적지 까지 전달되어 Bufferbloat이 해소된 시점은 주변 트래픽 종료 시간보다 더 늦은 26초와 48.8초 정도임을 그래프를 통해서 알 수 있다.



(a) Existing MPTCP (Lowest RTT)



(b) Proposed MPTCP (Lowest RTT)

그림 8. 기존 MPTCP와 제안된 MPTCP의 성능 비교 (Lowest RTT)
Fig. 8. Throughput comparison of existing MPTCP and proposed MPTCP (Lowest RTT)

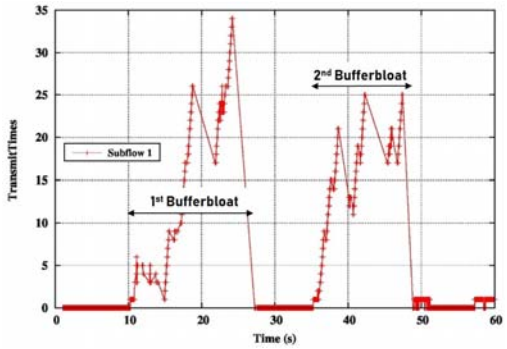


그림 9. Bufferbloat이 발생할 때 서브플로우 1의 $Transmit\,Times$ 변화

Fig. 9. Variations of $Transmit\,Times$ of Subflow 1 when bufferbloat occurs

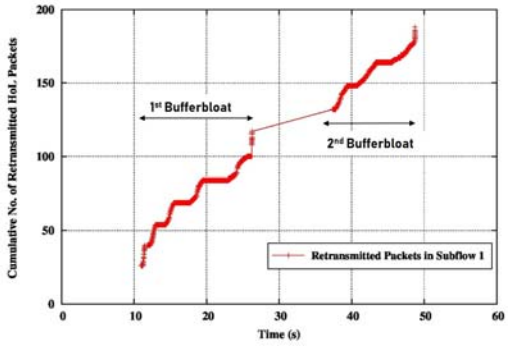


그림 10. 서브플로우 1로 재전송된 HoL 패킷 누적 개수

Fig. 10. Cumulative number of Retransmitted HoL Packet to subflow 1

본 논문에서는 송신단에 전송 상태를 모니터링할 수 있는 버퍼를 MPTCP 레벨에서 생성하였으며, 이를 통해 전송 RTT 횟수를 예측하여 Bufferbloat을 감지하는 방법을 제안하였다. 시뮬레이션 기간 중 전송 RTT 횟수 예측 값은 Fig. 9에서 확인할 수 있고, Bufferbloat 기간 동안 정상 경로의 예상 전송 RTT 횟수가 증가됨으로서, 본 논문에서 제안한 방법으로 Bufferbloat 현상을 정확히 인지할 수 있음을 보여주고 있다. 이와 같은 값 증가는 Bufferbloat 발생 기간 동안 지연 경로의 패킷이 제때 도착하지 못하여 수신단에서 HoL Blocking이 일어났고, 이 때문에 정상 경로의 송신단은 MPTCP 레벨의 ACK가 지연됨으로서 송신 버퍼에 쌓여 있는 패킷 개수가 지속적으로 늘어남에 기인한다고 할 수 있다. 또 본 논문에서는 전송 상태 모니터링 버퍼를 통해 HoL 패킷을 식별하고, 재전송 하는 방법을 제안하였다. 이와 같은 방법으로 Bufferbloat을 감지하고 수신단에서 기다리고 있는 HoL 패킷이 재전송이 되면 어떤 스케쥴러를 사용하든지 향상된 성능을 기대 할 수 있다. 본 논문에서 제안한 알고리즘이 적용된 Round Robin 스케쥴러의 경우 Fig. 7(b), Lowest RTT 스케쥴러의 경우 Fig. 8(b)를 통해 향상된 성능을 확인 할 수 있다. 본 알고리즘을 적용할 경우 Bufferbloat 감지 초반 또는 회복된 경로로 전송이 집중될 때 잠시 성능에 영향 받는 구간이 발생하지만, Round Robin의 경우 12.29Mbits/s에서 15.09Mbits/s로 전체 성능이 증가하였고, Lowest RTT는 13.23Mbits/s에서 15.17Mbits/s로 증가하여 매우 개선된 성능 결과가 있음을 확인하였다. 제안한 알고리즘은 앞서 기술했듯이 Bufferbloat 기간 동안 Bufferbloat 경로로 계속 데이터가 전송되기 때문에 프로브 패킷 역할을 수행한다. 따라서 전송 상태 모니터링 버퍼에서 예상 전송 RTT 횟수와 HoL 패킷 발생 유무를 모니터링을 통해 Bufferbloat 경로의 가용 유무를 쉽게 판단할 수 있으므로, Bufferbloat이 종료되는 시점인 26초와 48.8초에 문제 경로가 바로 재사용됨을 Fig. 7(b)와 Fig. 8(b)를 통해서 알 수 있다. Fig. 10은 정상 경로인 서브플로우 1로 전송된 HoL 패킷의 누적 개수를 나타내고 있으며, Bufferbloat 기간 동안 HoL 패킷이 모두 재전송되므로 HoL Blocking으로 성능 저하가 발생하지 않는다.

표 2. MPTCP 송신 버퍼에 있는 대기 패킷 개수

Table 2. Number of queued packets in MPTCP sending buffer

	MPTCP		Proposed MPTCP	
	Max (Packets)	Average (Packets)	Max (Packets)	Average (Packets)
Round Robin	2643	255	1316	135
Lowest RTT	2324	264	693	117

표 3. 수신 버퍼에 있는 대기 패킷 개수

Table 3. Number of queued packets in receiver's buffer

	MPTCP		Proposed MPTCP	
	Max (Packets)	Average (Packets)	Max (Packets)	Average (Packets)
Round Robin	2446	149	347	15
Lowest RTT	1774	173	289	16

표 4. 수신단에서 재정렬에 소요 되는 시간
Table 4. Reordering time at receiver

	MPTCP		Proposed MPTCP	
	Max (ms)	Average (ms)	Max (ms)	Average (ms)
Round Robin	2874	90	423	4
Lowest RTT	2601	109	163	3

제안한 알고리즘으로 인하여 MPTCP 레벨 송신 버퍼 사이즈와 수신 버퍼사이즈 변화는 Table 2와 3을 통해 확인 할 수 있다. MPTCP 송신 버퍼의 경우, 재전송된 HoL 패킷으로 인해 정상 경로의 패킷은 ACK를 모두 수신하여 송신버퍼에서 없어지므로, 제안한 알고리즘의 송신 버퍼의 크기가 대폭 감소하였음을 Table 2를 통해 알 수 있다. 그리고 Bufferbloat 경로의 패킷은 ACK를 받을 때까지 송신 버퍼에 남아 대기하고 있으므로, 송신 버퍼에 저장된 평균 패킷 숫자는 (Round Robin은 135개, Lowest RTT는 117개) Bufferbloat으로 인한 지연 패킷에 기인한다고 할 수 있다. Table 3은 재정렬을 위해 수신 버퍼에 머물러 있는 패킷 크기를 나타내고 있고, 제안한 알고리즘을 적용할 경우 송신단 보다 훨씬 큰 폭으로 대기 패킷 수가 줄어들었음을 확인 할 수 있다. 즉 송신단에서 HoL 패킷이 모두 재전송되었으므로 수신단에서는 DSN 순서대로 조립되어 응용계측으로 전달되었고, 더 이상 기다리고 있는 패킷이 없으므로 수신 버퍼에 대기하고 있는 평균 패킷 수는 기존 Lowest RTT 기준으로 9.2% 수준으로 감소되었다 . 그리고 재정렬을 위해 수신단에 머무는 패킷 숫자가 감소됨으로서, 재정렬 과정에서 소요되는 시간도 줄어들었음을 Table 4를 통해 알 수 있다. 따라서 본 알고리즘을 적용할 경우, Lowest RTT의 경우 소요 시간이 평균 109ms에서 3ms로 감소하였고, Round Robin은 90ms에서 4ms로 감소하여, 실시간 스트리밍 서비스 또는 긴급 데이터 전송 등 시간에 민감한 응용 프로그램에 적합함을 알 수 있다.

5. 결론

Bufferbloat은 주변 네트워크 환경에 따라 불규칙적으로 발생 및 소멸되며, 발생 기간 동안 데이터 전달 시간을 심각하게 지연시켜 손실 기반 혼잡 제어 방법을 무력화 시킨다. MPTCP에서도 Bufferbloat이 발생되며 Bufferbloat 경로의 데이터 전송 지연 뿐만 아니라, 수신단에서 지연된 패킷으로 인한 HoL Blocking 현상이 발생하여 다른 경로의 성능까지 심각하게 저하시킨다. 서로 다른 특성을 가지는 이종망에서 MPTCP 성능 향상 을 위한 기존 연구는 동적인 특성을 갖는 Bufferbloat 문제 해결에 한계가 있고, MPTCP 스케쥴러는 경로의 특성을 반영하여 전송양을 조정할 수 있지만, 이미 문제 경로로 전송된 HoL 패킷을 해결하기에는 한계가 있다고 할 수 있다.

따라서 본 논문에서는 Bufferbloat 문제를 해결하기 위해 송신단의 MPTCP 레벨에서 데이터 전송 상태를 모니터링하기 위한 방법과 전송 RTT 횟수를 예측함으로서 Bufferbloat을 감지하기 위한 방법을 제안하였다. 본 논문에서 제안한 방법은 수신단의 도움 없이 송신단에서 독립적으로 Bufferbloat 경로를 판단할 수 있을 뿐만 아니라 송신단에서 수신단이 기다리고 있는 HoL 패킷을 식별하여 재전송 동작을 수행하게 한다. 본 논문에서는 모든 HoL 패킷을 정상 경로로 전송함으로서, 수신단의 HoL Blocking 현상을 없애도록 하였고, Bufferbloat 경로로도 계속 데이터를 전송하게 하여 Bufferbloat 문제가 해결될 때 문제 경로를 바로 사용하도록 하였다.

제안한 알고리즘은 NS-3에 구현하였으며, Round Robin과 Lowest RTT 스케쥴러에 적용하여 성능을 비교 분석하였다. 시뮬레이션 결과 Bufferbloat 기간 동안 기존 MPTCP는 모든 경로에서 심각한 성능 저하가 일어났으나, 제안한 알고리즘을 적용할 경우 정상 경로에서 Bufferbloat으로 인한 성능 저하 문제가 거의 발생하지 않아 뚜렷한 성능 개선이 있음을 확인하였다. 그리고 본 결과에서는 Bufferbloat 종료 후 Bufferbloat 경로가 바로 사용됨으로서 MPTCP의 다중 경로의 자원을 충분히 활용하고 있음을 보였다. 또한 송신단 및 수신단 버퍼의 뚜렷한 크기 감소를 통해 HoL Blocking 현상을 해결되었음을 확인하였다. 차후 연구에서는 전송 상태 모니터링 방법을 더욱 확장 시켜 MPTCP 스케쥴러와 연동하여 이종망 및 Bufferbloat 환경에서도 성능 개선 결과를 얻을 수 있는 방법을 제안 및 구현할 예정이다.

REFERENCES

[1] R. Stewart, "Stream Control Tramsmission Protocol," Internet Requests for Comments, IETF, RFC 4960, Sep. 2007.

[2] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation With Multiple Addresses," Internet Requests for Comments, IETF, RFC 6824, Jan. 2013.

[3] P. Christoph, F. Simone, A. Ozgu and B. Olivier, "Experimental Evaluation of Multipath TCP Schedulers," in Proceeding of the ACM SIGCOMM Capacity Sharing Workshop, Chicago, pp. 27-32, Aug. 2014.

[4] Y. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths," in Proceeding of the ACM SIGMETRICS, 2017.

[5] S. Chattopadhay, S. Nandi, S. Shailendra & S. Chakraborty, "Primary Path Effect in Multi-Path TCP: How Serious Is It for Deployment Consideration?," in Proceeding of the ACM MobiHoc, 2017.

[6] M. Li, A. Lukyanenko, S. Tarkoma, Y. Cui and A. Yla-Jaaski, "Tolerating path heterogeneity in multipath TCP with bounded receive buffers," Computer Networks, vol.64, pp. 1-14, 2014.

[7] Y. E. Guo, A. Nikravesh, Z. M. Mao, F. Qian and S. Sen, "Accelerating Multipath Transport Through Balanced Subflow Completion," in Proceeding of the ACM MobiCom, 2017.

[8] B. H. Oh, J. Lee, "Feedback-Based Path Failure Detection and Buffer Blocking Protection for MPTCP," IEEE/ACM Transactions on Networking, vol.24, no.99, pp. 3450-3461, 2016.

[9] M. Scharf, and S. Kiesel, "Head-of-line Blcoking in TCP and SCTP: Analysis and Measurements", IEEE GLOBECOM, pp. 1-5, Nov. 2006.

[10] V. Cerf, V. Jacobson, N. Weaver, and J. Gettys, "BufferBloat: What's Wrong with the Internet?," ACM Queue, vol. 9, no. 12, 2011.

[11] J. Gettys, and K. Nichols, "Bufferbloat: Dark buffers in the Internet", Communications of the ACM, Vol 55, Issue 1, pp. 57-65, Jan. 2012.

[12] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3G/4G Networks," in Proceeding of the 2012 ACM Internet Measurement Conference (IMC), 2012.

[13] S. Ferlin-Oliveira, T. Dreibholz, and U. Alay, "Tackling the Challenge of Bufferbloat in Multi-Path Transport over Heterogeneous Wireless Networks," in Proceeding of the 22nd International Symposium of Quality of Service (IWQoS'14), Hong Kong, China, pp. 123-128. IEEE, May 2014.

[14] I. H. Jung, J. Y. Lee and B. C. Kim "Design of a Retransmission Algorithm for HoL Packets of Receiver Buffer to Improve Performance of MPTCP with a Bufferbloat Path," J. IEIE, vol. 55, no. 2, pp. 39-49, Feb. 2018.

[15] M. Kheirkhah "Multipath tcp in ns-3," Apr. 2015. [Internet]. Available: http://dx.doi.org/10.5281/zenodo.32691

[16] C. Raiciu, M. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," Internet Requests for Comments, IETF, RFC 6356, Oct. 2011.

[17] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in Proceeding of IFIP Networking, pp. 1222-1227, 2016.

저자약력

정 일 형(Il-Hyung Jung) [정회원]

- 2000년 조선대학교 전자공학과 학사
- 2003년 한국정보통신대학원대학교(현재 KAIST) 석사
- 2014년 ~ 현재 충남대학교 전파정보통신공학과 박사과정
- 2004년 ~ 2005년 (주) 유비쿼스 연구원
- 2005년 ~ 현재 한국항공우주연구원 선임연구원

<관심분야> 데이터 통신, 자동제어, 센서 계측

이 재 용(Jae-Yong Lee) [정회원]

- 1988년 서울대학교 전자공학과 학사
- 1990년 한국과학기술원 전기 및 전자공학과 석사
- 1995년 한국과학기술원 전기 및 전자공학과 박사
- 1990년 ~ 1995년 디지콤 정보통신연구소 선임연구원
- 1995년 ~ 현재 충남대학교 전파정보통신공학과 교수

<관심분야> 데이터 통신, 인터넷, 네트워크 성능 분석

김 병 철(Byung-Chul Kim) **[정회원]**

- 1988년 서울대학교 전자공학과 학사
- 1990년 한국과학기술원 전기 및 전자공학과 석사
- 1996년 한국과학기술원 전기 및 전자공학과 박사
- 1993년 ~ 1999년 삼성전자 CDMA 개발팀
- 1999년 ~ 현재 충남대학교 전파정보통신공학과 교수

<관심분야> 이동인터넷, 이동통신 네트워크, 데이터 통신