JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 8 : 1508~1516 Aug. 2017

Bufferbloat 환경에서 MPTCP 성능 개선을 위한 대역폭 측정 기반 스케줄러 설계

김민섭 · 한기문 · 이재용 · 김병철*

Design of Bandwidth Measurement based Scheduler for Improving MPTCP Performance in Bufferbloat Environment

Min Sub Kim · Ki Moon Han · Jae Yong Lee · Byung Chul Kim*

Department of Radio and Information Communications Engineering, Chungnam National University, Daejeon 34134, Korea

요 약

Multipath TCP (MPTCP)는 다중 경로를 지원하는 전송계층 프로토콜이다. MPTCP가 가진 다중 경로 중에서 한 경로에 지연시간이 급격히 증가하는 “bufferbloat”가 발생하게 된다면 경로 간의 패킷 도착시간 차이로 수신버퍼에서 HoL blocking이 발생하여 bufferbloat가 발생된 경로뿐만 아닌 다른 경로의 성능도 저하되는 문제가 있다. 본 논문에서는 이와 같은 문제를 해결하기 위해 대역폭 측정 기반의 스케줄러를 제안한다. 대역폭 측정 기반 스케줄러는 각 서브플로우의 대역폭을 측정하여 이를 기반으로 패킷 스케줄링을 하도록 설계하였다. 제안한 스케줄러 검증을 위해 리눅스커널에 제안한 스케줄러를 구현하고 bufferbloat가 발생하는 테스트베드를 구성하여 기존 MPTCP와 비교분석 하였다. 실험결과 제안한 스케줄러가 bufferbloat 환경에서 기존 MPTCP보다 성능이 크게 개선됨을 보였다.

ABSTRACT

Multipath TCP (MPTCP) is a transport layer protocol that supports multipath transmission. If a bufferbloat occurs in one of the subflows of MPTCP, HoL blocking occurs at the receiver due to the difference in packet arrival time among paths. In MPTCP, HoL blocking degrades not only the performance of the path where bufferbloat occurs, but also the performance of other paths. In this paper, we propose a bandwidth measurement based scheduler to solve this problem. Bandwidth measurement based scheduler is designed to measure the bandwidth of each subflow and to perform packet scheduling based on it. In order to verify the proposed scheduler, we implemented the proposed scheduler in the Linux kernel and constructed a testbed in which bufferbloat occurs. Experimental results show that the proposed scheduler has better performance than the legacy MPTCP in bufferbloat environment.

키워드 : 다중경로 TCP, MPTCP 스케줄러, 버퍼블로트, Head-of-Line blocking, 가용 대역폭 측정

Key word : Multipath TCP, MPTCP Scheduler, bufferbloat, Head-of-Line blocking, Available Bandwidth Estimation

Received 27 June 2017, **Revised** 10 July 2017, **Accepted** 17 July 2017

* **Corresponding Author** Byung Chul Kim(E-mail:byckim@cnu.ac.kr, Tel:+82-42-821-6867)
Department of Radio and Information Communications Engineering, Chungnam National University, Daejeon 34134, Korea

Open Access https://doi.org/10.6109/jkiice.2017.21.8.1508 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

오늘 날의 통신망에서 각 노드들의 네트워크 버퍼는 반도체 기술 발달과 가격의 하락으로 인하여 크기가 크게 확장되었다. 이로 인하여 패킷들이 큰 버퍼에 갇혀 전송 되지 못하는 bufferbloat [1]현상이 발생하였다. 그리고 bufferbloat 현상은 큰 버퍼로 인하여 Round Trip Time (RTT)이 급격하게 증가하여 TCP 성능저하에 원인이 된다. 이러한 성능저하는 기존 TCP를 확장한 Multipath TCP (MPTCP) [2]에서도 발생하는데 한쪽 경로가 bufferbloat로 인해 지연시간이 급격하게 늘어나게 되면 다른 경로로 먼저 도착한 패킷이 bufferbloat로 도착하지 못한 패킷이 도착할 때 까지 대기하고 있는 상태가 발생한다. 이러한 현상을 HoL(Head-of-Line) blocking [3]이라 부르며 HoL blocking은 MPTCP의 전체적인 성능저하를 가지고 온다.

본 논문에서는 MPTCP에서 bufferbloat로 인한 HoL blocking을 방지하기 위해 대역폭 측정 기반 스케줄링을 제안한다. 제안한 스케줄러는 각 경로의 가용 대역폭을 지속적으로 측정하고 측정된 대역폭으로 각 경로별 예상 윈도우를 계산한다. 그리고 계산 된 예상 윈도우 값을 이용하여 각 경로 별로 대역폭에 알맞은 패킷 양을 할당하는 방식으로 동작한다. 이 방안은 급격히 대역폭이 저하되는 subflow의 인입 데이터를 줄여주어 사전에 HoL blocking을 해결하고자 제안하였다.

제안한 스케줄러의 성능분석을 위해 bufferbloat가 발생하는 네트워크 환경을 구성한 테스트베드를 구성하고 제안한 스케줄러와 기존 MPTCP 스케줄러를 비교 분석하였다. 실험결과 제안한 스케줄러가 기존 MPTCP 스케줄러보다 bufferbloat 환경에서 다중 경로를 통한 트래픽 전송 성능을 크게 개선함을 확인하였다.

본 논문의 구성은 Ⅱ장에서는 MPTCP 프로토콜과 bufferbloat를 기술하고 TCP 대역폭 측정 기법을 사용한 프로토콜과 가용 대역폭 측정 기법에 대해 소개한다. Ⅲ장에서는 bufferbloat로 인한 MPTCP 성능저하 문제와 이를 해결하기 위한 기존연구를 기술하며 제안한 스케줄링 알고리즘을 소개한다. Ⅳ장에서는 제안한 스케줄러를 bufferbloat 환경을 구성한 테스트베드에서 기존 MPTCP 스케줄러와 비교 분석 하였다. 마지막으로 5장에서는 본 논문의 결론을 맺는다.

Ⅱ. 관련 연구

2.1. Multipath TCP

MPTCP는 2013년 IETF에서 표준화된 전송 계층 프로토콜로서 서로 다른 IP 주소를 가지고 있는 여러 개의 인터페이스를 이용하여 동시에 데이터 전송이 가능한 프로토콜이다. 이전에 다중 경로를 지원하는 전송 계층 프로토콜은 SCTP [4]가 제안되었으나 NAT와 같은 네트워크 환경에서 정상적으로 동작 하지 않으며 기존 TCP와 UDP로 구성된 응용계층에서 모두 SCTP로 변경해야 하는 문제가 발생하였다. 하지만 MPTCP는 기존 TCP 프로토콜에 옵션을 확장하였기 때문에 NAT 문제를 해결하고 기존 TCP 응용을 그대로 사용할 수 있다는 장점을 가진다.

그림 1은 MPTCP 프로토콜 구조를 나타낸 것이며, MPTCP계층을 살펴보면 MPTCP계층과 Subflow계층으로 나뉜다. MPTCP계층에서는 경로 관리 및 혼잡 제어, 스케줄링 등을 수행하며 그 중 MPTCP 스케줄러는 상위 계층에서 내려온 데이터를 서브플로우로 분배하는 역할을 담당한다. 그리고 Subflow계층은 독립적인 TCP 세션으로 기존 TCP처럼 동작한다.

MPTCP 동작과정을 살펴보면 송신측 MPTCP가 응용 계층에서 전달 받은 패킷에 Data Sequence Number (DSN)라는 새로운 시퀀스 번호를 부여한다. 그 후 스케줄러가 각 서브플로우로 패킷을 분배하게 되며 각 서브플로우는 별도의 TCP 세션으로 동작한다. 수신측 MPTCP에서는 각 Subflow계층에서 패킷을 받아 MPTCP계층으로 전달하고 MPTCP계층에서는 송신측에서 부여한 DSN 순서에 맞게 정렬하여 응용계층으로 전송하는 방식으로 동작한다[2].

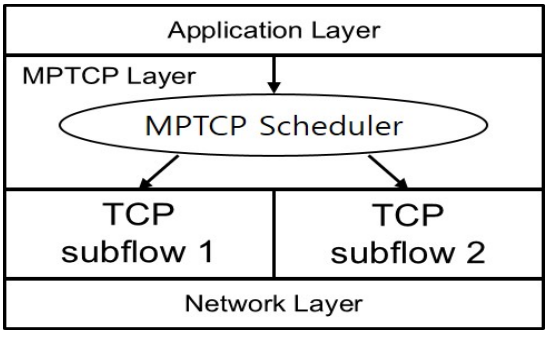


Fig. 1 MPTCP protocol stack

2.2. Bufferbloat

3G와 LTE 같은 이동통신망은 링크 환경에 따라 변조방식이나 코드를 변경하기 때문에 전송대역폭이 가변적으로 변화한다. 이 때 bufferbloat는 대역폭 가변 현상으로 대역폭이 줄어들고 이때 과도한 버퍼크기로 패킷이 손실 되지 않으며 버퍼에 계속 쌓여 지연시간이 증가하는 것을 말한다. MPTCP를 사용 할 때 bufferbloat가 한 쪽 경로에 발생하면 경로 별 패킷 도착 시간 차이로 HoL blocking이 발생한다. 예를 들어 그림 2처럼 Wi-Fi와 같은 지연시간이 작은 경로와 3G와 같은 지연시간이 큰 경로를 MPTCP 다중 경로로 사용할 때 작은 지연시간을 가진 빠른 경로로 도착한 패킷은 큰 지연시간의 느린 경로로 패킷이 도착 할 때 까지 MPTCP 수신버퍼에서 대기하게 된다. 그리고 이 상황이 지속 되어 빠른 경로의 수신버퍼까지 가득 차게 된다면 빠른 경로의 서브플로우도 수신버퍼의 제약으로 성능저하가 발생한다. 이러한 이유로 HoL blocking은 네트워크 상황이 좋지 않은 경로뿐만 아니라 네트워크 상황이 좋은 경로 또한 성능을 저하시키기 때문에 MPTCP 처리량 성능이 단일 TCP를 사용한 처리량 성능보다 저하되는 경우가 발생하게 된다[5].

2.3. 가용 대역폭 측정 기법

TCP 대역폭 측정 기법은 TCP가 지나가는 병목 링크의 가용 대역폭을 측정하는 기법이다. 대역폭 측정 기법을 사용한 TCP 프로토콜은 TCP Westwood [6]와 이를 개선한 TCP Westwood+ 그리고 TCP TIBET, TCP Jersey [7-9]등이 있다. 본 절에서는 TCP 대역폭 측정 기법의 시작인 TCP Westwood의 대역폭 측정 기법에 대해서 소개한다.

TCP Westwood는 종단 간 네트워크의 대역폭을 측정하고 측정한 대역폭을 바탕으로 이상적인 윈도우 크기를 계산하여 혼잡제어에 사용한다. TCP Westwood의 동작은 TCP Reno 프로토콜 스택에서 송신측 알고리즘을 일부 수정 한 것으로 일반적인 동작은 TCP Reno와 동일하게 동작한다. 하지만 TCP Reno와 다르게 종단 간 대역폭 측정을 지속적으로 수행하고 측정한 대역폭을 이용하여 이상적인 윈도우 크기를 계산한다. 그리고 손실이 발생한 경우 TCP Reno는 윈도우 크기를 절반으로 줄이지만 TCP Westwood의 경우 측정 대역폭으로 계산한 윈도우 크기로 줄이게 된다.

TCP Westwood의 가장 큰 특징은 네트워크 계층의 정보 없이 종단 간 대역폭 측정을 하는 것으로 이를 위해 TCP Westwood는 그림 3처럼 ACK 도착시간의 간격동안 보낸 패킷 량을 통하여 대역폭을 계산한다.

$$b_i = \frac{d_i}{(t_i - t_{i-1})} \tag{1}$$

$$cwnd^{TCPW} = \frac{\hat{b} \times RTT_{\min}}{MSS} \tag{2}$$

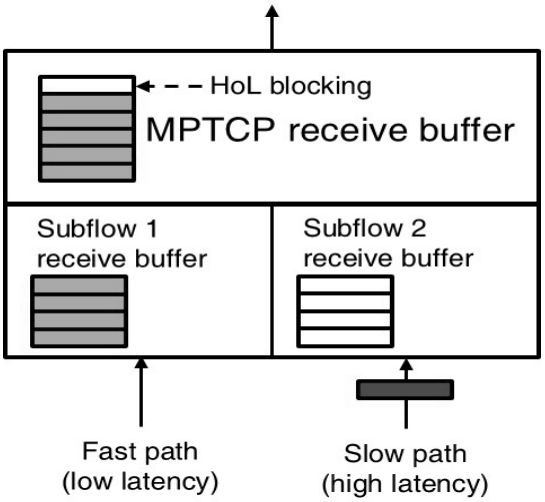


Fig. 2 HoL blocking in MPTCP

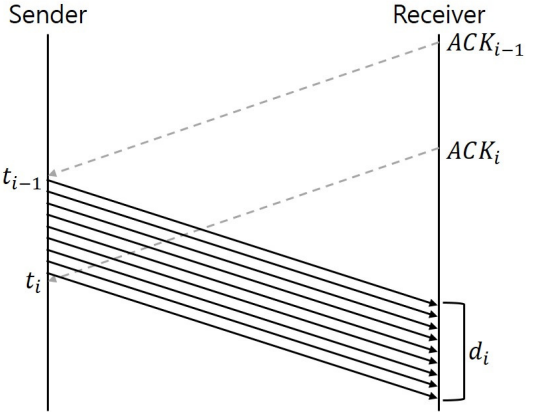


Fig. 3 Bandwidth measurement technique of TCP Westwood

대역폭 계산은 식 (1)을 이용하여 샘플 대역폭 b_i를 계산하는데 이때 d_i는 ACK 간격동안 보낸 패킷 량을 의미한다. 계산된 샘플 대역폭 b_i는 로우패스필터를 이용하여 $\hat{b_i}$로 사용한다. $\hat{b_i}$는 3 duplicated ACK나 timeout이 발생할 때 식 (2)를 이용하여 예측한 대역폭 크기에 맞는 윈도우 값을 계산하여 혼잡 윈도우로 사용한다. 식 (2)에서 $RTT_{\min}$은 경로 상에서 가장 작은 RTT를 의미하며 MSS는 최대 세그먼트 크기를 나타낸다. 식 (2)에서 RTT를 사용하면 이상적인 윈도우 크기를 계산할 수 있지만 $RTT_{\min}$을 사용하는 이유는 윈도우 크기를 작게 설정하여 병목 링크에서의 backlog를 줄일 수 있기 때문이다[6].

Ⅲ. 제안한 스케줄링 알고리즘

3.1. Bufferbloat로 인한 MPTCP 성능 저하 문제

MPTCP를 사용할 때 초기에는 경로 별 특성이 비슷하더라도 특정 경로에 대역폭이 갑자기 줄어서 bufferbloat현상이 발생하여 경로 별 특성이 상이해 질 경우 2.2절에서 살펴본 HoL blocking이 발생하게 된다. HoL blocking은 성능이 좋지 않은 경로뿐만 아니라 성능이 좋은 경로의 성능 또한 저하시키기 때문에 다중 경로를 통한 성능 향상을 볼 수 없다. 해당 문제를 개선하기 위해서 많은 연구들이 선행되어 왔으나 기존 연구들은 HoL blocking이 발생하면 이를 빠르게 개선하는 방향으로 연구가 진행되어왔다. 선행된 연구에는 느린 경로의 서브플로우를 액티브모드에서 백업모드로 전환하는 방법[10]과 느린 경로에서 도착하지 않은 패킷을 빠른 경로로 재전송하는 기법[11, 12], 그리고 느린 경로에 페널티를 부여하여 혼잡 윈도우 크기를 줄이는 기법[13, 14] 등이 있다. 이러한 기존 해결방법들은 HoL blocking이 발생하면 이를 대처하기 위한 방법으로 HoL blocking을 미연에 방지할 수 없다. 따라서 본 논문에서는 MPTCP가 가진 서브플로우의 대역폭을 지속적으로 측정하고 이를 기반으로 경로 별 대역폭에 비례한 패킷 스케줄링을 통하여 HoL blocking 문제를 사전에 방지하고자 한다.

Bandwidth Measurement technique

```
#Calculates the size of transmitted data using the
first sequence number of the previous data and the
first sequence number of the current data.

if(timer > RTT){
     snd_data = Seq_num of the first byte of date
              - pre_snd_una
     pre_snd_una = Seq_num of the first byte of data
}

#Calculate the expected bandwidth using updated param
eter values
     estimated_BW = 0.825 * estimated_BW
                  + 0.125 * (snd_data / RTT)

#Calculate window size using sRTT, MSS and estimated
bandwidth
est_bw_window {
     return max((estimated_BW * sRTT)/ mss, 1 )
}
```

Fig. 4 Bandwidth measurement method of the proposed scheduler

3.2. 대역폭 측정 기반 스케줄링 알고리즘

bufferbloat 발생의 근본원인은 전송대역폭이 급감하기 때문이므로 제안한 스케줄러는 bufferbloat를 예측하기 위해 대역폭 측정을 사용하여 설계하였다. 제안한 스케줄러는 대역폭 측정 기법을 적용하여 각 서브플로우의 가용 대역폭을 측정하고 측정한 대역폭의 비율에 맞추어 서브플로우 별 전송 스케줄링을 하는 방식으로 동작한다.

먼저 제안한 스케줄러의 대역폭 측정 기법은 2.3절에서 소개한 TCP Westwood의 대역폭 측정 기법을 기반으로 설계하였다. 그림 4는 제안한 스케줄러의 대역폭 측정 pseudo code이다. 대역폭 측정 과정은 식 (1)을 사용하여 샘플 대역폭을 측정한다. 하지만 기존 식 (1)과 다르게 ACK 간격이 아닌 RTT를 interval로 사용하였다. 그리고 식 (2)을 사용하여 각 서브플로우의 대역폭에 알맞은 윈도우 크기를 계산한다. 이 경우도 TCP Westwood의 경우 $RTT_{\min}$을 사용하지만 제안한 알고리즘은 $sRTT$를 사용하여 이상적인 윈도우 크기를 계산하도록 설계하였다.

다음으로 제안한 스케줄러의 동작과정을 그림 5를 이용하여 나타내었다. 스케줄링을 시작하면 대역폭 측정 유무를 구분하고 측정을 하지 않은 경우에는 대역폭 측정을 하게 된다. 대역폭을 측정 후 계산된 윈도우 값을 각 서브플로우의 최대 서비스 할당량으로 지정하고 최대 서비스 할당량이 가장 큰 서브플로우를 선택한다.

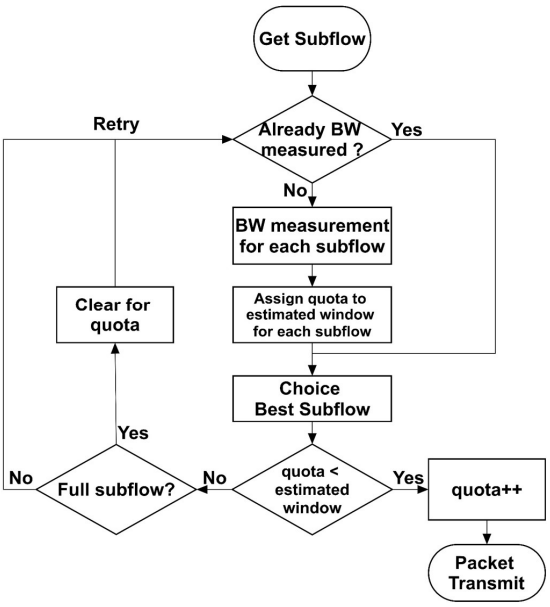


Fig. 5 The flow chart of the proposed scheduler

선택한 서브플로우의 서비스 할당량을 채우면 다음 서브플로우로 전환되며 모든 서브플로우가 할당량을 채울 경우 대역폭 측정값을 초기화하고 다시 대역폭 측정을 하는 방식으로 동작한다.

제안한 스케줄러의 구현을 위해 리눅스 MPTCP kernel v.90 [15]버전의 Round-robin 스케줄러에 제안한 스케줄러 알고리즘을 적용하였다. 다음 장에서는 bufferbloat 환경을 구성한 테스트베드에서 실험을 진행한 결과에 대해서 살펴보겠다.

Ⅳ. 성능분석

4.1. 실험환경

제안한 스케줄러를 실험하기 위해 그림 6과 같은 테스트베드를 구성하였다. 테스트베드에서 사용한 모든 PC는 Ubuntu 14.04 LTS를 운영체제로 사용하고 추가적으로 MPTCP sender와 receiver에는 MPTCP kernel v0.90을 설치하였다. 그리고 UDP sender와 receiver를 크로스 케이블로 각 서브플로우에 연결하여 공유링크를 구성하였다. 다음으로 테스트베드에 bufferbloat 환경을 구현하기 위해서 각 서브플로우의 공유링크 노드

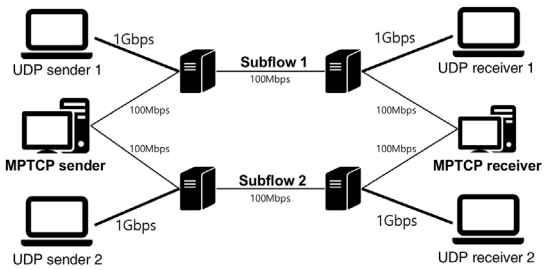


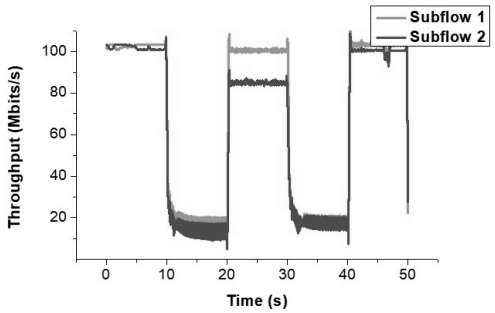
Fig. 6 MPTCP testbed network topology

들의 송신 큐 크기를 변경하였다. 리눅스에서 기본 송신 큐의 크기는 1000 패킷이지만 공유링크 노드 송신 큐 사이즈는 운영체제가 가질 수 있는 최대 크기 패킷으로 설정하였다. MPTCP와 공유링크의 링크속도는 100Mbps로 구성하고 UDP는 1Gbps의 링크속도로 구성하였다. MPTCP 설정은 혼잡제어로 LIA 혼잡제어 [16]를 사용하고 경로관리는 설정하지 않았다. MPTCP 스케줄러는 제안한 스케줄러와의 성능을 비교하기 위해서 LowRTT [17]와 Round-robin을 사용하여 실험을 진행하였다. 그리고 네트워크 성능 측정 툴로는 iperf [18]를 사용하여 TCP 트래픽과 UDP 트래픽을 발생시켰다.

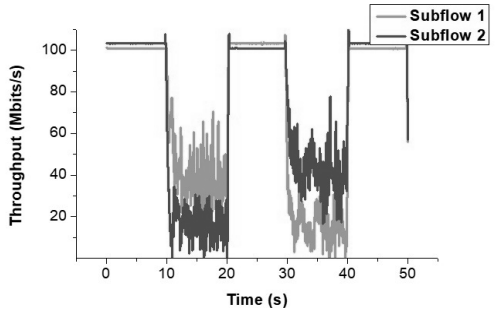
4.2. 실험결과

실험은 구성한 테스트베드에서 MPTCP 트래픽을 50초 동안 발생을 시키고 10초~20초 구간에서 Subflow 2에 80Mbps의 UDP 트래픽을 10초간 발생시켰으며 30초~40초 구간에서 Subflow 1에 80Mbps의 UDP 트래픽을 10초간 발생시켜 양 쪽 서브플로우에서 bufferbloat 현상이 발생하도록 하였다. 실험순서는 Round-robin 스케줄러와 LowRTT 스케줄러 [16], 제안한 스케줄러 순서로 진행하였다. 실험 결과로 그림 7과 그림 8은 각 MPTCP 스케줄러 별 처리량 성능과 RTT를 그래프로 나타내었다.

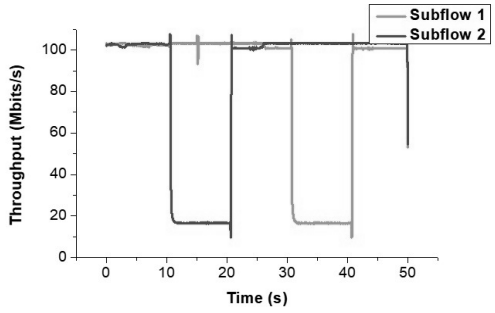
먼저 Round-robin 스케줄러의 실험결과를 보면 50초 동안 평균 129Mbits/s의 전송속도로 총 812Mbytes를 전송하였다. 그리고 그림7(a)의 10초 ~ 20초와 30초 ~ 40초에 bufferbloat 현상이 발생하면 HoL blocking으로 좋은 경로의 성능도 저하되는 것을 확인 하였다. Round-robin 스케줄러가 3개의 스케줄러 중에 가장 낮은 처리량 성능의 원인은 Round-robin 스케줄러가 경로



(a) Round-robin scheduler

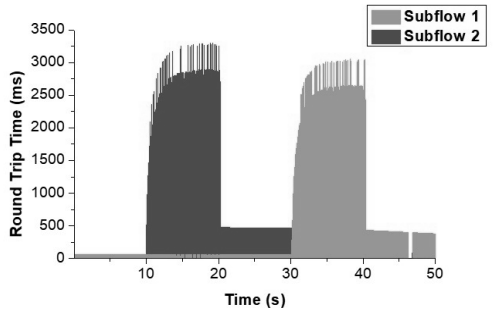


(b) LowRTT scheduler

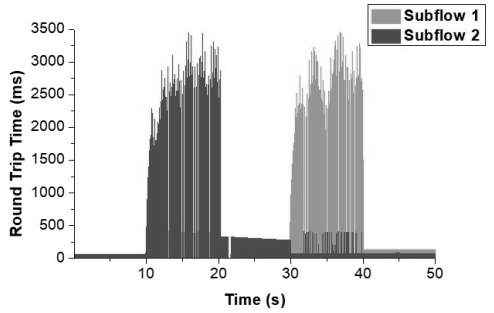


(c) Bandwidth measurement based scheduler

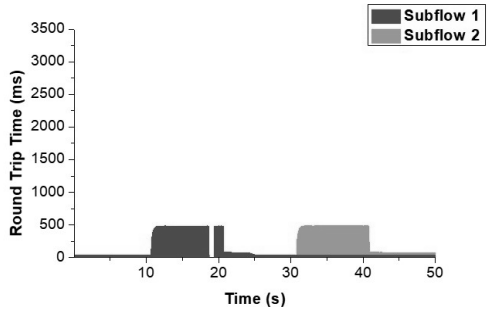
Fig. 7 Throughput graph of existing MPTCP scheduler and proposed scheduler



(a) Round-robin scheduler



(b) LowRTT scheduler



(c) Bandwidth measurement based scheduler

Fig. 8 Round Trip Time graph of existing MPTCP scheduler and proposed scheduler

상황에 대한 고려가 전혀 없기 때문이다. Round-robin 스케줄러는 bufferbloat가 발생하는 구간에도 양쪽 서브플로우에 동일한 양의 트래픽을 보내게 되고 그림 8(a)처럼 bufferbloat가 발생한 서브플로우의 RTT가 3200ms 이상으로 증가하는 모습을 볼 수 있다. 이러한 다중 경로상의 RTT 차이가 심해지면 HoL blocking 발생하게 되고 bufferbloat 구간에서의 Round-robin 스케줄러의 처리량 성능은 단일 TCP를 사용한 경우보다 낮은 처리량 성능을 보여주었다.

다음으로 LowRTT 스케줄러는 50초 동안 평균 142Mbits/s의 전송속도로 총 893Mbytes를 전송하였다. LowRTT 스케줄러는 RTT를 기반으로 RTT가 작은 경로를 우선적으로 스케줄링하고 느린 경로에 윈도우 크기를 절반씩 줄이는 페널티를 부여하는 스케줄러이다. 하지만 그림 7(b)를 보면 10초~20초, 30초~40초의 bufferbloat 구간에서 HoL blocking이 발생하여 좋은 경로의 성능도 저하된 것을 볼 수 있다. LowRTT 스케줄러가 HoL blocking이 일어난 원인은 윈도우크기를 절

반 씩 줄이는 페널티 부여방식이 RTT당 1번씩만 적용되어 급작스러운 bufferbloat 현상에 대응하지 못하기 때문이다. 그리고 혼잡한 경로에 부여된 페널티로 인해 윈도우크기가 작게 줄어 bufferbloat가 발생한 혼잡한 링크를 다시 비 혼잡한 경로로 착각하게 한다. 혼잡한 경로를 비 혼잡한 경로로 착각한 LowRTT 스케줄러는 양쪽 경로에 데이터를 급격하게 보내며 이로 인해 혼잡한 경로의 RTT가 다시 증가하여 또 다시 페널티를 부여하게 된다. 이 과정 속에서 HoL blocking 발생하고 해결되는 것을 반복하기 때문에 RTT 그래프가 그림 8(a)처럼 연속적인 RTT증가가 아닌 그림 8(b)와 같은 굴곡진 형태로 나타난다. 결국 LowRTT 스케줄러는 Round-robin 스케줄러보다 성능저하를 감소시켰지만 안 좋은 경로의 적절한 패킷 양을 찾지 못하여 HoL blocking이 발생하였다.

마지막으로 제안한 스케줄러는 50초 동안 평균 168Mbits/s의 전송속도로 총 1056Mbytes를 전송하여 3개의 스케줄러 중에서 가장 우수한 처리량 성능을 보여주었다. 제안한 스케줄러의 경우 대역폭을 측정하고 대역폭에 비례하여 패킷 스케줄링을 하기 때문에 그림 7(c)에서 10초~20초 bufferbloat 구간에서는 Subflow2는 UDP트래픽을 제외한 남은 대역폭만큼 전송하고 Subflow1는 100Mbps를 유지하는 것을 확인하였다. 그리고 다음 bufferbloat 구간인 30초~40초에서도 Subflow1은 UDP트래픽을 제외한 남은 대역폭만큼 전송하고 Subflow2는 100Mbps를 유지하는 것을 확인하였다. 이러한 결과는 제안한 스케줄러에서 bufferbloat가 발생한 10초~20초, 30초~40초 구간에서 가용 대역폭 20M를 측정하여 2:10의 비율로 패킷을 스케줄링 하였기 때문이다. 그림 8(c)의 RTT 그래프처럼 bufferbloat 구간의 RTT가 500ms 이상으로 증가하지 않고 유지되어 HoL blocking으로 인한 성능저하가 발생하지 않았다.

4.3. 비교분석

bufferbloat 환경을 구현한 테스트베드에서 기존 스케줄러와 제안한 스케줄러를 실험한 결과를 바탕으로 비교분석을 하였다. 표 1은 기존 스케줄러와 제안한 스케줄러의 총 전송량과 최대 RTT 증가량에 대해서 비교한 것으로 총 전송량에서 Round-robin 스케줄러가 가장 낮은 전송량을 보여주었으며 제안한 스케줄러는 가장 높은 전송량을 보였다. 그리고 최대 RTT에서는 Low-RTT 스케줄러가 가장 좋지 않은 결과를 보여주었으며 제안한 스케줄러는 기존의 Round-robin 스케줄러, Low-RTT 스케줄러와 다르게 최대 RTT가 가장 낮은 것을 볼 수 있다.

본 실험을 통하여 제안한 스케줄러는 대역폭 측정 기법을 이용하여 각 서브플로우의 가용 대역폭 크기를 측정하고 측정한 대역폭 크기에 비례한 패킷 스케줄링으로 bufferbloat 환경에서 HoL blocking 현상을 미연에 방지하여 MPTCP 성능저하를 줄였다.

Table. 1 The result of bufferbloat test

	Round-robin	Low-RTT	Proposed
Total traffic	812Mbytes	893Mbytes	1056Mbytes
Maximum RTT	3298ms	3449ms	490ms

Ⅴ. 결 론

본 논문에서는 bufferbloat 환경에서 HoL blocking으로 인한 MPTCP 성능저하를 해결하기 위해서 대역폭 측정 기법을 이용한 대역폭 기반 스케줄러를 제안하였다. 이는 bufferbloat의 근본원인이 전송대역폭이 급변하는 네트워크 환경에서 발생하기 때문에 대역폭 급감을 스케줄러에 반영하기 위해서였다. 제안한 스케줄러는 각각의 서브플로우의 가용 대역폭을 측정하고 이를 바탕으로 각 경로 별 대역폭 크기에 비례하여 패킷 스케줄링을 하도록 설계하였다. 제안한 스케줄러는 리눅스커널에 구현하였으며 bufferbloat가 발생하는 테스트베드를 구성하여 기존 MPTCP 스케줄러와 비교 분석하였다.

실험결과 bufferbloat 환경에서 기존 MPTCP 스케줄러는 대역폭급감으로 인한 지연시간이 급격하게 증가하기 때문에 HoL blocking이 발생하여 성능저하를 일으켰다. 그에 반해 대역폭 기반 스케줄러의 경우 대역폭에 비례하여 패킷을 스케줄링 하도록 설계되어있기 때문에 HoL blocking이 발생하는 것을 방지 하는 효과를 볼 수 있었다. 하지만 제안한 스케줄러는 기존의 연구되어왔던 스케줄러와 다르게 느린 경로에 대한 페널티를 부여하지 않고 측정 대역폭에 비례한 패킷 스케줄링

만 구현하였기 때문에 성능을 개선하기 위해서는 느린 경로에 페널티를 부여하는 알고리즘을 추가로 필요하다. 차후 연구에는 대역폭 측정 기반 스케줄러에 페널티를 부여하는 방법을 추가할 예정이다.

ACKNOWLEDGMENTS

This work was supported by research fund of Chungnam National University

REFERENCES

[1] J. Gettys, “Bufferbloat: Dark buffers in the Internet,” *IEEE Internet Computing*, vol. 15, no. 3, pp. 95-96, May/June 2011.

[2] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, “TCP Extensions for Multipath Operation With Multiple Addresses,” RFC 6824, IETF, Jan. 2013. [Internet]. Available: https://tools.ietf.org/html/rfc6824.

[3] M. Scharf, and S. Kiesel, “Head-of-line Blocking in TCP and SCTP: Analysis and Measurements,” in *Proceedings of the IEEE GLOBECOM*, San Francisco, pp. 1-5, Nov. 2006.

[4] R. Stewart, “Stream Control Transmission Protocol,” Internet Requests for Comments,” RFC 4960, IETF, Sep. 2007. [Internet]. Available: https://tools.ietf.org/html/rfc4960.

[5] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, “Experimental evaluation of multipath TCP schedulers,” in *Proceedings of the ACM SIGCOMM Capacity Sharing Workshop*, Chicago, pp. 27-32, Aug. 2014.

[6] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi and R. Wang, “TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links,” in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, Rome, pp. 287-297, July 2001.

[7] L. A. Grieco and S. Mascolo, “Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 25-38, Apr. 2004.

[8] A. Capone, L. Fratta, and F. Martignon, “Bandwidth estimation schemes for TCP over wireless networks,” *IEEE Transactions on Moblie Computing*, vol. 3, no. 2, pp. 129-143, July 2004.

[9] K. Xu, Y. Tian, and N. Ansari, “TCP-Jersey for Wireless IP Communications,” *In IEEE Journal on Selected Areas in Communication*, vol. 22, no. 4, pp. 747-756, May 2004.

[10] S. Ferlin, T. Dreibholz, and O. Alay, “Multi-path transport over heterogeneous wireless networks: Does it really pay off?,” in *Proceedings of the IEEE GLOBECOM*, Austin, pp. 5005-5011, Dec. 2014.

[11] F. Yang, Q. Wang, and P. Amer, “Out-of-order transmission for in-order arrival scheduling policy for multipath TCP,” in *Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops*, Victoria, pp. 749-752, May 2014.

[12] S. Ferlin, O. Alay, O. Mehani et al., “BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks,” in *Proceedings of the IFIP Networking Conference and Workshops*, Vienna, pp. 431-439, May 2016.

[13] S. H. Baidya, and R. Prakash, “Improving the performance of multipath TCP over heterogeneous paths using slow path adaptation,” in *Proceedings of the IEEE International Conference on Communications*, Sydney, pp. 3222-3227, Jun. 2014.

[14] M. S. Kim, J. Y. Lee, and B. C. Kim, “Improving the performance of Multipath TCP using Delay Alerted Path-blocking Scheduler in Heterogeneous Networks,” *The Journal of the Institute of Electronics and Information Engineers*, vol. 54, no. 2, pp. 28-37, Feb. 2017.

[15] MultiPath TCP Linux Kernel implementaion [Internet]. Available: http://mptcp.info.ucl.ac.be/.

[16] C. Raiciu, M. Handley, and D. Wischik, “Coupled Congestion Control for Multipath Transport Protocols,” RFC 6356, IETF, Oct. 2011. [Internet]. Available: https://tools.ietf.org/html/rfc6356

[17] J. H. Hwang, J. Yoo, “Packet scheduling for Multipath TCP,” in *Proceedings of the 7th International Conference on Ubiquitous and Future Networks*, Sapporo, pp. 177-179, July 2015.

[18] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool [Internet]. Available: https://iperf.fr/.

김민섭(Min Sub Kim)

2016년 2월 충남대학교 정보통신공학 공학사
2016년 3월 ~ 현재 충남대학교 전자전파정보통신공학과 컴퓨터네트워크 석사 재학
※관심분야 : 이동통신 네트워크, 네트워크 성능분석

한기문(Ki Moon Han)

2016년 8월 충북대학교 정보통신공학 공학사
2017년 3월 ~ 현재 충남대학교 전자전파정보통시공학과 컴퓨터네트워크 석사 재학
※관심분야 : 이동통신 네트워크, 네트워크 성능분석, 머신러닝

이재용(Jae Yong Lee)

1988년 2월 서울대학교 전자공학과 공학사
1990년 2월 한국과학기술원 전기전자공학부 공학석사
1995년 2월 한국과학기술원 전기전자공학부 공학박사
1990년 ~1995년 디지콤 정보통신 연구소 선임연구원
1995년 ~ 현재 충남대학교 전파정보통신공학과 교수
※관심분야 : 무선 메쉬망, MANET, 네트워크 성능분석

김병철(Byung Chul Kim)

1988년 2월 서울대학교 전자공학과 공학사
1990년 2월 한국과학기술원 전기전자공학부 공학석사
1996년 2월 한국과학기술원 전기전자공학부 공학박사
1993년 ~ 1999년 삼성전자 무선네트워크 사업부 선임연구원
1999년 9월 ~ 현재 충남대학교 전파정보통신공학과 교수
※관심분야 : 유/무선 인터넷, 이동통신 네트워크, 이동성 처리