Journal of The Institute of Electronics and Information Engineers Vol.54, NO.2, February 2017 https://doi.org/10.5573/ieie.2017.54.2.028
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2017-54-2-4

이종 망에서의 지연경보 경로차단 스케줄러를 이용한 MPTCP성능 개선방안

(Improving the performance of Multipath TCP using Delay Alerted Path-blocking Scheduler in Heterogeneous Networks)

김 민 섭*, 이 재 용**, 김 병 철**

(Min Sub Kim, Jae Yong Lee©, and Byung Chul Kim)

요 약

Multipath TCP (MPTCP)는 다중 경로를 동시에 사용하여 전송효율과 안정성을 얻을 수 있는 4계층 프로토콜로 현재 IETF를 통해 표준화되어 사용되고 있다. MPTCP는 비슷한 특성의 다중 경로를 망에서 사용하는 경우 단일 TCP보다 좋은 성능을 얻을 수 있지만 특성이 다른 망에서 사용하는 경우 다중 경로 간 지연시간의 차이로 인해 다중경로간의 패킷 도착시간의 차이가 발생하여 수신버퍼에서 패킷을 reordering하는 문제가 발생한다. 이러한 패킷 reordering 문제로 인해 이종 망에서의 MPTCP는 단일 TCP를 사용하는 것보다 성능이 저하되는 문제점을 가지고 있다. 따라서 본 논문에서는 MPTCP 각 경로상의 지연시간을 측정하여 지연시간이 크게 증가하는 경로를 차단하여 수신버퍼의 패킷 reordering을 줄이고, 차단한 경로에 작은 크기의 복제 패킷을 보내어 지속적인 지연시간측정을 하여 해당 경로의 네트워크의 혼잡이 줄어들면 차단을 해제하는 지연경보 경로차단 스케줄링을 제안하였다. 제안한 스케줄링의 성능분석을 위해 리눅스커널에 제안한 스케줄링을 구현 설치하고 테스트베드를 구성하여 실험을 통해 제안 스케줄링이 이종 망에서 MPTCP의 성능저하를 감소시키는 것을 확인하였다.

Abstract

Multipath TCP (MPTCP), which is a layer 4 protocol that can get the reliability and the efficiency of the transmission by using multipath transfer, was standardized by the IETF. MPTCP provides superior performance when compared to a single TCP when used in a homogeneous network with similar network characteristics. However, MPTCP degrades performance when used in heterogeneous networks with different network characteristics. In this paper, we propose 'Delay-alerted path-blocking scheduler'. It measures the delay of each path and blocks the path with a long delay to reduce the order of packets in the receive buffer. If the duplicated packet is sent to the blocked path to measure the delay and the congestion on the blocking path is reduced, the blocked path is unblocked. For performance analysis, the proposed scheduler was implemented in the Linux kernel and improved performance was obtained in the test bed. We also confirmed that the proposed scheduler reduces the degradation of MPTCP performance in real wireless networks with heterogeneous path characteristics.

Keywords : Multipath TCP, Bufferbloat, MPTCP scheduler, Heterogeneous network, TCP

* 학생회원, ** 평생회원, 충남대학교 정보통신공학과
(Dept. of InfoComm Engineering, Chungnam National University)
© Corresponding Author (E-mail : jyl@cnu.ac.kr)
※ 이 연구는 2016년도 교육부의 재원으로 한국연구재단의 지원(No.2016-907089)과 충남대학교 학술연구비에 의해 지원되었음.
Received ; October 5, 2016 Revised ; December 1, 2016
Accepted ; January 23, 2017

Ⅰ. 서 론

최근 무선 단말들은 웹서핑과 같은 기본적인 서비스 뿐만 아니라 HD 스트리밍 서비스 또는 고품질 음원서비스와 같이 데이터 사용량이 높은 서비스의 수요가 지속적으로 증가하고 있다. 이러한 서비스 변화에 따라 사용해야하는 트래픽 양은 계속 증가하고 있지만 무선

네트워크의 자원은 유한하기 때문에, 무선자원을 효율적으로 이용하는 것은 매우 중요한 문제라 할 수 있다.

최근 단말들은 2개 이상의 네트워크 인터페이스를 보편적으로 지원하기 때문에 효율적인 다중 인터페이스 관리는 사용자가 체감할 수 있는 성능을 향상 시킬 수 있다. Multipath TCP (MPTCP)는 여러 개의 인터페이스를 이용하여 동시에 데이터 전송이 가능한 transport 계층의 프로토콜로서, 이를 통하여 무선 자원의 효율성을 높일 수 있다. MPTCP는 2013년 IETF에서 표준화가 되었고[1] 국내 이동 통신사에서 KT는 '기가 LTE', SKT는 '밴드 LTE 와이파이', LGU는 '기가 멀티 패스'라는 명칭으로 일부 사용자에 한에서 서비스를 제공하였다. 또한 PC에서 사용하는 경우 리눅스 공식 배포 판에는 포함되어 있지 않으나 MPTCP를 위한 OS 및 소켓 API패치가 오픈소스 형태로 공개되어있다. 스마트폰 단말기에는 Apple의 'iPhone'에 내장되어 있는 'siri' 어플리케이션에 MPTCP를 적용하여 사용되고 있으며, 안드로이드 기반 스마트폰인 삼성 '갤럭시 S7'과 엘지 'G5'에도 MPTCP를 탑재하여 LTE와 WiFi를 동시에 사용할 수 있다. 구글 레퍼런스 스마트폰에는 리눅스용 MPTCP 패치를 적용하여 MPTCP를 사용할 수 있다.

MPTCP와 유사하게 다중 경로를 사용하는 프로토콜로 SCTP(Stream Control Transmission Protocol)[2]가 이전에 제안되었으나, NAT(Network Address Translation)나 NAPT(Network Address Port Translation)와 같은 네트워크 환경 하에서 정상적으로 동작을 하지 않는다는 문제점이 있었다. 하지만 MPTCP의 경우 기존 TCP 프로토콜에 옵션을 추가하였기 때문에 이러한 문제를 해결하였다. 또한 기존 TCP 옵션 필드를 사용하기 때문에 기존 TCP를 사용하는 어플리케이션을 그대로 사용할 수 있어 호환성 측면에서도 장점을 가진다.

현재 MPTCP의 경우 여러 네트워크를 동시에 사용하여 데이터를 전송 할 때, 전송경로의 특성이 비슷한 경우에는 기존 TCP보다 좋은 성능과 안정성을 얻을 수 있다. 하지만 전송경로의 특성이 서로 상이한 경우 네트워크 전송 속도의 차이가 발생할 수 있고, 빠른 경로의 패킷이 수신 단 버퍼에 먼저 쌓여 reordering이 필요하게 된다. 전송속도가 느린 네트워크에서 손실이 발생할 경우, 이를 복구하기 위해서 더 오랜 시간이 필요할 뿐만 아니라 수신측에서 빠른 네트워크를 통해서 온 패킷들이 어플리케이션 계층으로 올라가지 못하고 대기하는 현상이 발생하기 때문에 기존 TCP보다 성능이 저하된다[3].

현재 LTE, LTE-A, WiFi 등 다양한 네트워크가 존재하고 이러한 네트워크가 계속 늘어날 것이므로 특성이 다른 다양한 네트워크를 다중 경로로 구성하게 될 경우, 이러한 성능 저하 문제는 지속적으로 제기될 것이다. 따라서 서로 다른 특성을 고려하여 MPTCP의 다중 경로를 효율적으로 관리하는 연구가 추가적으로 필요하다.

본 논문에서는 지연시간이 다른 경로로 인한 MPTCP의 성능저하 문제를 해결하기 위한 스케줄링 방법을 제안한다. 제안된 방법은 경로 별 sRTT (smoothed Round Trip Time)을 측정하여 네트워크 혼잡도를 결정하고 지연시간이 늘어난 경로를 차단하여, 느린 경로로 인한 수신 단 처리 지연 현상을 최소화 할 수 있다. 차단된 경로로는 복제 패킷을 전송함으로 sRTT를 측정하여 차단된 경로의 혼잡을 지속적으로 측정한다. 차단된 경로의 혼잡이 완화된 경우 다시 다중 경로 전송에 참여하게 된다.

본 논문의 구성은 다음과 같다. 제 2장에서는 MPTCP에 대해 설명하고 기존 MPTCP에서 발생할 수 있는 문제점과 이를 해결하기 위한 기존연구를 기술한다. 제 3장에서는 제안한 스케줄링 알고리즘에 대해 설명과 구현방법에 대해 다루었으며, 제 4장에서는 제안한 스케줄러를 검증하기 위한 유선 테스트베드 실험 환경 및 성능 분석 결과를 기술하였다. 제 5장에서는 제안한 스케줄러를 실제 무선네트워크에 적용하여 실험 분석한 결과를 기술하였으며 마지막으로 제 6장에서 본 논문의 결론을 맺는다.

Ⅱ. MPTCP 및 문제점 분석

1. MPTCP

MPTCP는 IETF에서 표준화가 되었으며 기존 TCP 프로토콜을 확장한 4계층 프로토콜이다[1]. 기존 TCP를 확장한 개념이기 때문에 TCP를 사용하는 어플리케이션은 그대로 MPTCP를 사용 할 수 있다.

MPTCP 구조는 그림 1과 같이 MPTCP와 서브플로우의 2계층으로 구성되어있다. 서브플로우 계층은 하나의 독립적인 TCP 세션으로 동작하여 서브플로우 별로 흐름제어 및 연결제어를 하며, MPTCP 계층에서는 각각의 서브플로우에 대한 연결제어 및 서브플로우 간의 트래픽 분배와 전체 패킷 ordering 기능을 수행한다. MPTCP는 그림 2와 같이 연결을 수행 할 때는 TCP처럼 3-way-handshaking을 사용하여 연결을 한다. 이때

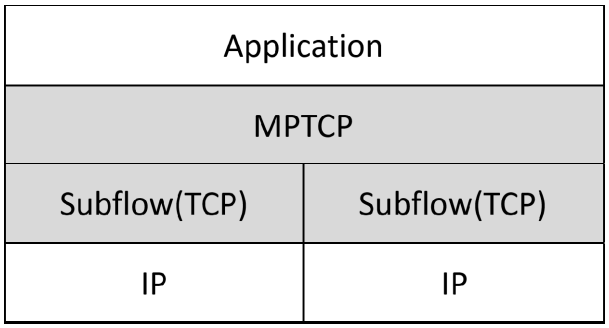


그림 1. MPTCP 프로토콜 스택
Fig. 1. MPTCP protocol stack.

MP_CAPABLE 옵션을 이용하여 서로 key를 주고받는다. 추가적으로 다른 서브플로우와 연결을 위해서는 4-way-hand shaking을 사용하여 연결을 한다[4]. client는 서브플로우를 추가하기 위해 SYN에 MP_JOIN 옵션을 보내게 된다. 해당 MP_JOIN을 받은 server는 SYN+ACK와 함께 HMAC과 MP_JOIN을 보낸다. HMAC과 MP_JOIN옵션을 받은 client는 확인을 위해 다시 ACK를 통해 MP_JOIN과 HMAC을 전송한다. 해당 ACK를 받은 server는 ACK를 전송하며 연결이 완료된다. 연결이 끝난 뒤에는 데이터 전송을 시작하며 MPTCP는 서브플로우 계층에서 사용하는 sequence number 뿐만 아니라 MPTCP 계층에서 사용하는 Data Sequence Number(DSN)도 사용한다. 각 서브플로우는 MPTCP 계층에서 받은 데이터를 가지고 독립적인 TCP세션으로 동작하며, 수신 측에서는 도착한 패킷을 서브플로우 별 처리 후, MPTCP 계층으로 넘겨주게 된다. 이 때 송신측에서 부여한 DSN를 이용하여 각 서브플로우 별 패킷을 ordering 하게 되며, 이 후 상위 계층보내진다.

2. Bufferbloat

Bufferbloat[5]는 과도한 버퍼 크기로 인하여 망에서 사용자 패킷의 지연이 커지는 현상으로서, 요즘 인터넷에서 종종 관찰된다. 당초 과도한 버퍼 크기는 3G/4G 셀룰러 망에서 폭발적인 트래픽의 손실을 방지하기 위해 도입되었으나, 오히려 큰 버퍼로 인하여 트래픽의 손실이 발생하지 않아 TCP 혼잡제어가 제대로 이뤄지지 않는 부작용을 초래하고 있다. 즉, TCP는 대부분 손실 기반의 혼잡제어를 사용하는데 중간에 과도한 버퍼 크기로 인하여 손실이 일어나지 않으면, 혼잡제어가 제대로 역할을 수행하지 못하는 현상이 발생한다[6]. 혼잡제어가 제대로 이루 어지지 않으면 결국 망 혼잡을 알 수 없게 되어 적절한 망 제어가 이루어지지 않고, BDP(Bandwidth Delay Product)보다 훨씬 많은 패킷이유입되게 된다. 이

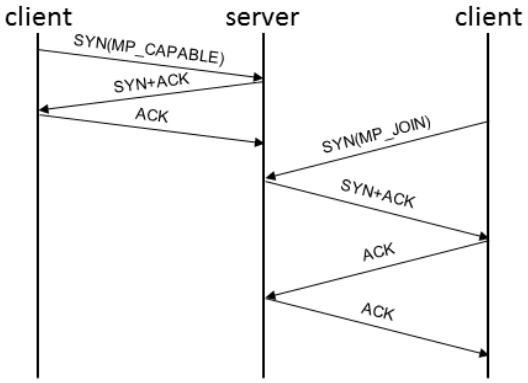


그림 2. MPTCP 연결과정
Fig. 2. MPTCP connection establishment.

때 패킷들은 중간의 큰 버퍼에 장시간 저장되고, 이로 인하여 사용자간의 RTT는 점점 증가하게 된다. MPTCP의 경우 서브플로우가 별도의 TCP 세션으로 동작하기 때문에 bufferbloat현상은 TCP뿐만 아니라 MPTCP에서도 일어난다.

3. 이종 망에서 MPTCP 성능 저하 문제

MPTCP에서 서브플로우 별로 전혀 다른 특성을 가지고 있는 네트워크를 통해 패킷 전송이 될 수 있는 경우를 간과해서는 안 된다.

예를 들어서 서브플로우 1은 RTT 값이 큰, 즉 느린 경로의 특성을 가질 수 있고, 서브플로우 2는 RTT 값이 작은, 즉 빠른 경로의 특성을 가질 수 있다. 이때 경로간의 패킷 도착시간에 많은 차이가 나게 되므로, 그림 3과 같이 빠른 경로의 서브플로우 2 패킷들이 수신 버퍼에 먼저 도착해 아직 도착하지 않은 서브플로우 1의 패킷을 기다리게 되는 수신버퍼 제한 현상이 발생한다. 만약 제한된 수신버퍼의 크기로 인해 버퍼가 채워지면, 빠른 경로를 가지는 서브플로우 2의 성능에도 영향을 주게 되어 MPTCP의 전체적인 성능저하가 일어나게 된다. 이러한 현상을 MPTCP 수신버퍼에서의 Head-of-Line Blocking (HoL Blocking)[7]이라고 한다. HoL Blocking이 발생하는 이유는 현재 MPTCP 스케줄링이 경로에 대한 고려 없이 데이터를 전송하기 때문이다.

이종 망에서 MPTCP 성능저하 문제를 개선하기 위해 많은 연구들이 선행되었는데, 먼저 각 서브플로우의 네트워크 상황을 보고 해당 서브플로우를 백업모드나 액티브모드로 전환하는 기법[8]이 있다. 해당 방법은 단순히 느린 경로를 차단하여 수신 단에서 발생하는 패킷 reordering을 줄이는 것이다. 하지만 느린 경로의 서브플로우가 백업모드로 전환된 후 네트워크 정보를 얻지

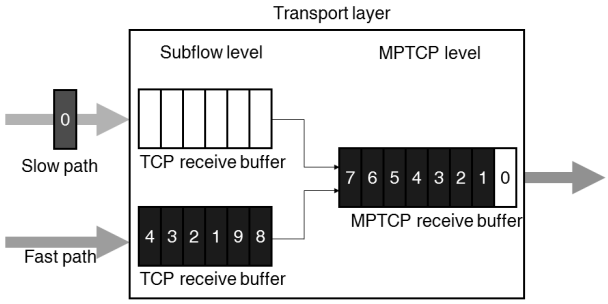


그림 3. MPTCP 수신버퍼 HoL 블로킹
Fig. 3. HoL blocking in MPTCP receive buffer.

못하여 상황이 개선된 후 다른 경로 전송이 복구되지 않는 문제점을 가지고 있어 해당 경로의 상황이 개선되어도 망의 자원을 효율적으로 이용하지 못할 수 있다는 단점을 가지고 있다. 다음으로는 느린 경로에서 도착하지 않은 패킷을 빠른 경로로 재전송하는 기법[9~10]이 있다. 이 방법은 HoL Blocking의 원인이 되는 패킷을 빠른 경로로 재전송하여 HoL Blocking을 최소화 하는 방법이다. 하지만 느린 경로로 전송되는 패킷에 대한 조절이 없어 느린 경로의 네트워크로 전송되는 패킷이 많을 경우 여전히 성능 상의 문제점을 일으킬 수 있다. 마지막으로 서브플로우별로 혼잡 윈도우를 조절하는 방법으로 느린 경로에 패널티를 부여하여 패킷 양을 줄이는 기법[11]이 있다. 이 방법은 느린 경로로 보내는 패킷의 양을 줄여 HoL Blocking이 일어나는 빈도를 줄일 수는 있지만 적은 양의 패킷이라도 새로운 패킷들이 느린 경로를 통하여 계속적으로 전송되기 때문에 HoL Blocking이 일어나 수신 단에서 패킷 reordering이 발생한다.

이러한 기존 해결방법들은 HoL Blocking으로 인한 패킷 reordering 문제를 해결할 수 없거나 네트워크 자원을 효율적으로 이용하지 못하는 단점이 여전히 존재한다. 따라서 느린 경로로 인한 HoL Blocking문제를 해결하면서 네트워크의 상황을 지속적으로 관찰하여, 필요시 재활용 할 수 있는 새로운 스케줄링이 필요하다.

Ⅲ. 제안 알고리즘

본 장에서는 2, 3절에서 언급한 기존 MPTCP의 문제점을 해결하기 위한 스케줄링 알고리즘을 제안한다. 제안한 스케줄링 알고리즘은 지연시간을 이용하여 느려진 경로를 차단하는 방식으로 Delay Alerted Path-blocking (DAPB) 라고 정의 하였다. DAPB 스케줄러는 2가지 과정으로 나누어서 볼 수 있다. 첫 번째는 해당 서브플로우가 속한 망의 혼잡을 일정 지연시간 값을 기준으로

Delay Alerted Path-blocking algorithm 1

```
srtt_i : smoothed round-trip time of subflow i
srtt_thresh : smoothed round-trip time threshold
value

for each subflow i do
      if srtt_i < min_srtt then
         min_srtt = srtt_i;
      end if
end for
srtt_thresh = max (3*min_srtt,
initial_srtt);

for each subflow i do
      if srtt_i > srtt_thresh then
         goto DAPB algorithm;
      end if
end for
goto round-robin algorithm;
```

그림 4. 지연경보 경로차단 스케줄러 알고리즘 1
Fig. 4. Delay Alerted Path-blocking Scheduler algorithm 1.

구분 하고, 두 번째로 혼잡한 망과 혼잡하지 않은 망에서의 서브플로우의 데이터 전송을 다르게 동작하는 방식으로 나누어 볼 수 있다.

1. sRTT를 이용한 스케줄링 알고리즘

제안하는 스케줄링 알고리즘의 경우 지연시간의 차이로 늦게 도착하는 패킷들 때문에 일어나는 reordering으로 인해 수신버퍼의 블로킹 문제를 줄이는 것이 목적이다. 먼저 스케줄링을 위해서 혼잡한 경로를 파악하고 해당 서브플로우를 차단하는 알고리즘을 수행하게 된다. 혼잡한 경로를 파악하기 위해 각 서브플로우의 smoothed round-trip time(sRTT)를 이용하여 혼잡을 판단하게 된다. 혼잡을 판단하기 위한 기준은 그림 4의 알고리즘으로 계산하게 된다.

먼저 모든 서브플로우 중에서 최소 sRTT 값을 저장한다. 이를 이용하여 혼잡 기준 값으로 최소 sRTT 값의 3배 한 값과 초기 sRTT 값 중에 큰 값을 선택한다. 위의 기준대로 계산된 혼잡 기준 값을 이용하여 특정 서브플로우가 혼잡 기준 값보다 더 크다면 해당 서브플로우를 차단하고 다른 서브플로우로 전송하는 패킷과 동일한 복제 패킷을 전송하는 DAPB 방식의 데이터 전송을 수행하고 모든 서브플로우가 혼잡 기준 값보다 작은 경우 다중 경로를 통해 데이터를 순차적으로 보내는 기존 round-robin 방식의 데이터 전송을 수행하게 된다. 본 논문에서는 그림 4의 알고리즘에서 혼잡 기준에 사용되는 초기 sRTT 값은 모든 서브플로우가 원활히 동작하는 경우에도 약간의 지연시간 차이로 인해 특정 서브플로우가 차단되는 것을 막기 위함이다. 예를 들어

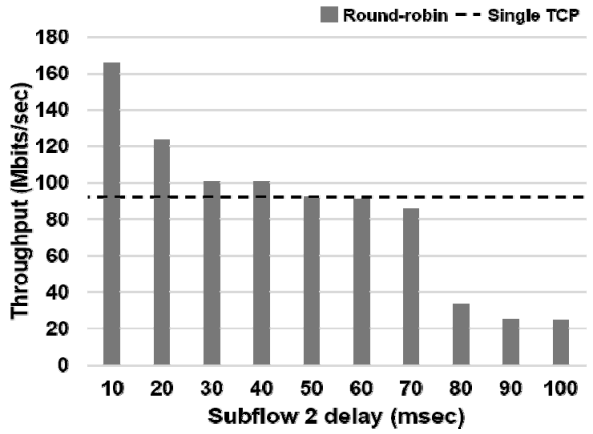


그림 5. 서브플로우간 지연시간차이로 인한 MPTCP round -robin 스케줄러 성능 분석

Fig. 5. MPTCP round-robin scheduler performance analysis due to delay difference between sub-flows.

그림 7의 테스트베드 환경에서 MPTCP Round-robin 스케줄링을 사용할 경우 한 쪽 경로가 100M/10ms이고 다른 경로의 지연시간을 dummynet을 이용하여 점점 증가하게 될 경우 그림 5에서와 같이 한쪽 경로의 지연시간이 50ms를 넘을 때 MPTCP의 성능이 단일 TCP보다 저하되게 된다. 따라서 그림 5의 결과를 바탕으로 테스트베드에서의 초기 sRTT 값을 50ms로 지정하였다.

2. 경로 별 전송 스케줄링

혼잡 기준 값을 이용하여 서브플로우의 혼잡을 구분한 뒤에 그림 6처럼 2가지의 알고리즘으로 데이터 전송을 하게 된다.

일반적으로 네트워크가 혼잡하지 않아 모든 서브플로우의 sRTT가 혼잡 기준 값보다 작은 경우 그림 6의 round-robin 알고리즘으로 동작하여 TCP 송신 버퍼의 데이터를 순차적으로 각 서브플로우의 윈도우 크기만큼 할당하여 전송하게 된다. 그러다가 MPTCP가 가진 서브플로우 중에서 혼잡한 서브플로우가 존재하여 특정 서브플로우의 sRTT가 혼잡 기준 값보다 큰 경우 그림 6의 DAPB 알고리즘으로 동작하게 된다. DAPB 방식은 혼잡한 서브플로우로 인해 다른 서브플로우의 성능이 저하되는 것을 막기 위해 혼잡한 서브플로우를 차단하는 방법으로 동작하게 된다. 하지만 기존의 서브플로우를 차단하는 스케줄링과 다르게 차단한 서브플로우에는 다른 서브플로우로 전송되는 패킷과 동일한 데이터를 전송하여 차단된 서브플로우의 혼잡 측정을 수행한다. 이러한 복제 패킷의 경우 네트워크가 혼잡 하지 않은 서브플로우를 통해서 이미 수신측에 도착하므로 복제

Delay Alerted Path-blocking algorithm 2

```
srtt_thresh : smoothed round-trip time threshold
value
first_subflow : In the MPTCP connection list, the
first subflow
skb : tcp socket buffer

round-robin algorithm:
round-robin next_segment()
{
tcp_sock = first_subflow;
      do
        skb = tcp_send_head;
        dup_data = skb;
        return skb;

        tcp_sock = tcp_sock -> next
      while !tcp_sock then
}

DAPB algorithm:
DAPB next_segment()
{
tcp_sock = first_subflow;
      do
        if tcp_sock->srtt >
         srtt_thresh then
                tcp_sock->cwnd = 1;
                skb = dup_data;
                return skb;
        end if

        skb = tcp_send_head;
        dup_data = skb;
        return skb;

        tcp_sock = tcp_sock -> next
      while !tcp_sock then
}
```

그림 6. 지연경보 경로차단 스케줄러 알고리즘 2

Fig. 6. Delay Alerted Path-blocking Scheduler algorithm 2.

패킷을 수신한 서브플로우 계층에서는 ACK를 보내지만 MPTCP계층에서는 중복된 데이터로 판단하여 버리게 된다. 이러한 이유로 복제 패킷은 다른 서브플로우의 성능에 영향을 주지 않고 네트워크의 혼잡을 측정할 수 있다. 이 때 복제 패킷이 다른 서브플로우에 영향을 주지 않더라도 서브플로우가 가지고 있던 혼잡 윈도우 크기만큼을 전송하는 것은 오히려 해당 네트워크의 혼잡을 야기할 수 있기 때문에 윈도우 크기를 1로 지정하여 복제 패킷으로 인한 혼잡을 최소화하고 서브플로우의 네트워크 혼잡을 지속적으로 측정하게 된다.

이러한 지속적인 측정을 통해서 해당 네트워크의 혼잡이 줄어들게 되어 모든 서브플로우의 sRTT가 혼잡 측정 기준보다 낮아지게 된다면 다시 round-robin 알고

리즘으로 전환되어 다중 경로 전송을 한다. 차단된 서브플로우를 해제되는 혼잡 기준 값은 기존 값에 a(0.5) 배를 곱하여 사용함으로서 혼잡과 비 혼잡으로의 oscillation 현상을 방지하여 안정적인 동작이 가능하도록 하였다.

3. DAPB 알고리즘 구현

제안한 알고리즘 구현을 위해서 리눅스 PC에 MPTCP kernel v0.90[13]을 사용하였다. 커널 파일 내부에는 redundant 스케줄링이라는 모든 서브플로우에 동일한 패킷을 전송하는 스케줄링이 있으며 제안한 DAPB 스케줄링은 기본적으로 redundant 스케줄링 코드를 이용하여 구현 하였다. redundant 스케줄링 코드 내부에는 redundant 패킷 전송 함수가 있으며 이 함수를 DAPB 알고리즘에 맞게 윈도우 크기 조절 함수와 최대 세그먼트 크기 조절 함수를 추가 구현하였다. 또한 round-robin 패킷 전송 함수를 구현하였으며 함수 도입부에는 서브플로우의 sRTT를 측정하여 혼잡 기준 값을 계산하고 이를 바탕으로 round-robin 방식의 전송 함수와 DAPB 방식의 전송 함수로 이동하는 조건 함수를 구현하였다.

위와 같은 구성을 통해서 DAPB 스케줄링은 sRTT를 이용하여 경로 별 전송 스케줄링을 구현하였으며 다음 장에서는 구현한 알고리즘을 유선 테스트베드와 실제 무선 네트워크에서 실험한 성능 분석을 다루었다.

Ⅳ. 유선 테스트베드에서 성능 분석

1. 실험환경

본 논문에서는 유선 테스트베드에서 이종 망 실험을 위해 테스트베드를 구성하여 실험을 진행하였다. 테스트베드는 그림 7과 같이 MPTCP 서버와 클라이언트를 2개의 서브플로우로 연결하였다. 또한 각 서브플로우의 네트워크에 지연시간을 다르게 적용하기 위해서 dummynet[12]을 사용하였다. dummynet은 네트워크 에뮬레이션 툴로서 네트워크 프로토콜을 테스트하기 위해 설계되었고 기본적으로 네트워크 계층과 응용 계층사이에 존재하며 대역폭, 지연, 패킷 손실 등을 설정 할 수 있다. MPTCP 서버와 클라이언트는 Ubuntu 14.04 LTS 운영체제를 사용하고 MPTCP kernel v0.90을 추가적으로 설치하였다. 그림 7의 테스트베드에서 모든 링크는 100Mbps의 대역폭을 가지며 네트워크의 성능 측정을 위해 iperf[14]를 사용하였다. MPTCP 설정에는 MPTCP 혼잡제어 방식으로 OLIA[15]를 사용하였으며 MPTCP 스케줄러의 경우 DAPB 스케줄러와 성능을 비교하기 위해 기본적으로

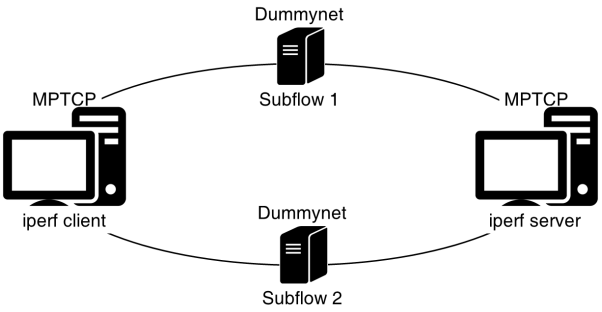


그림 7. MPTCP 유선 테스트베드 네트워크 구성
Fig. 7. MPTCP wire testbed network topology.

제공하는 LowRTT[16]와 round-robin를 사용하여 실험을 진행하였다.

2. 실험결과

가. 이종 망에서 대역폭 차이로 인한 성능비교

구성한 테스트베드를 이용하여 서브플로우 1의 네트워크 특성을 100M/10ms로 설정하고 서브플로우 2의 지연시간을 10ms, 대역폭을 100M부터 10M까지 변화를 주었다. 각 대역폭 별로 iperf를 사용하여 100초간 데이터를 전송 하였으며 스케줄러는 기본적으로 제공하는 round-robin 스케줄러와 LowRTT 스케줄러, 그리고 제안한 DAPB 스케줄러를 사용하였다. 그림 8은 서브플로우 2의 대역폭별 MPTCP 평균 전송속도를 각 스케줄러 별로 나타낸 것이다. 그림 8을 보면 서브플로우 2의 대역폭이 줄어들어 MPTCP 성능이 저하되지만 성능차이는 3개의 스케줄러가 비슷한 결과를 보여 주었다.

나. 이종 망에서 지연시간 차이로 인한 성능비교

동일한 테스트베드를 이용하여 서브플로우 1의 네트워크 특성을 100M/10ms로 설정하고 서브플로우 2의 대역폭을 100M, 지연시간은 10ms부터 100ms까지 변화를 주었다. 실험은 이전 실험과 동일하게 진행하였다. 그림 9은 서브플로우 2의 지연시간별 MPTCP 평균 전송속도를 각 스케줄러 별로 나타낸 것이다. 그림 9의 서브플로우 2의 지연시간이 80ms 이상인 부분을 보면 round-robin 스케줄러가 가장 안 좋은 결과를 보여주었으며 제안한 DAPB 스케줄러가 가장 좋은 결과를 보여주었다. 이러한 결과는 round-robin 스케줄링의 경우 경로에 대한 컨트롤을 전혀 하지 않고 해당 서브플로우의 혼잡 윈도우 크기만큼씩 번갈아가면서 데이터를 보내기 때문에 서브플로우 2의 지연시간이 80ms를 기점으로 수신버퍼에서 패킷 reordering이 발생하여 서브플로우 1

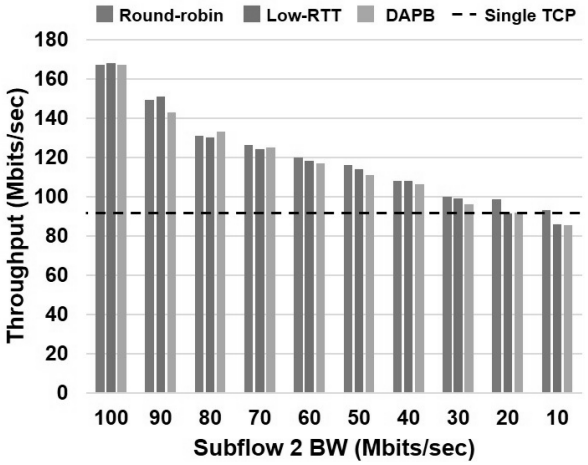


그림 8. 서브플로우 대역폭 차이에 따른 스케줄러 성능 그래프

Fig. 8. Scheduler performance graph with subflow band width difference.

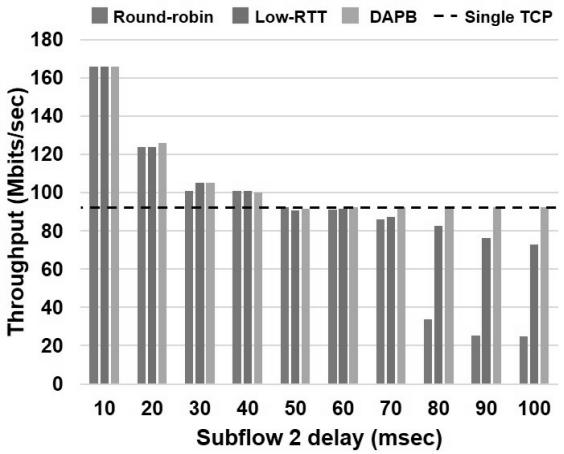


그림 9. 서브플로우 지연시간 차이에 따른 스케줄러 성능 그래프

Fig. 9. Scheduler performance graph with subflow delay difference.

의 영향을 주어 단일TCP보다 성능이 낮아지기 시작하였다. 서브플로우 2의 지연시간이 80ms 이상인 경우에 100M/10ms의 환경을 지닌 서브플로우 1이 있음에도 불구하고 27.5Mbit/s라는 매우 낮은 성능을 보였다.

다음으로 LowRTT는 round-robin보다는 개선된 성능을 가지고 있지만 서브플로우 2의 지연시간이 80ms 이상으로 늘어난 시점부터 단일TCP 성능보다 낮은 결과를 보였다. LowRTT 스케줄링은 리눅스 MPTCP kernel의 기본 스케줄링으로 낮은 RTT 순으로 데이터를 전송하는 스케줄링이다. 이러한 방식은 서브플로우 2의 지연시간이 증가 할수록 서브플로우 1로 대부분의 패킷이 전송되지만 서브플로우 2에도 적은양의 패킷이 전송되어 수신버퍼에서 패킷 reordering이 일어나 MPTCP 성능이 저하되는 원인이 된다.

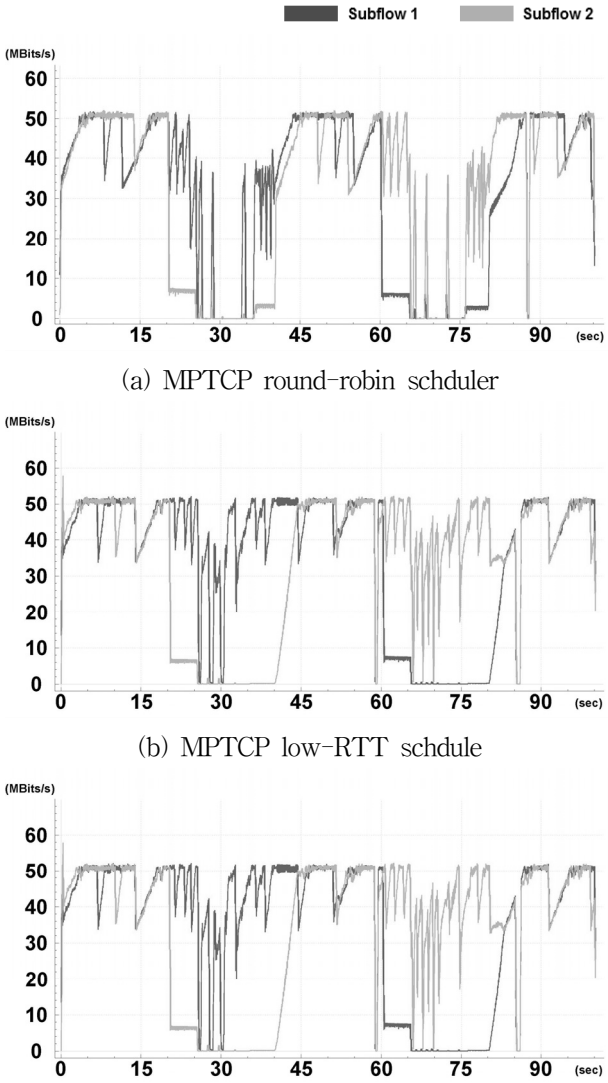
이에 반해 DAPB의 경우 서브플로우 2의 지연시간이 크지 않은 경우에는 round-robin으로 동작하여 기존의 round-robin의 비슷한 성능을 보여주었다. 하지만 round-robin과 다르게 서브플로우 2의 지연시간이 혼잡 기준값 이상으로 증가될 시에 서브플로우2의 경로를 차단하고 서브플로우1의 단일 경로만으로 동작하여 서브플로우2로 인한 HoL Blocking으로 서브플로우1의 성능이 저하되는 것을 줄일 수 있었다.

지연시간 차이에 따른 스케줄러 성능비교를 통해서 이종 망에서 기존의 MPTCP 스케줄러 보다 제안한 DAPB가 안정적인 성능을 제공한다고 볼 수 있다.

다. Bufferbloat 발생 환경에서의 스케줄러 성능비교

이종 망에서 기존 MPTCP 스케줄러와 제안한 스케줄러의 MPTCP 성능에 대해 분석을 하였다. 먼저 서브플로우 1과 서브플로우 2의 특성을 50M/10ms로 설정하여 서로 동일한 망을 구성한 뒤 iperf를 이용하여 100초간 데이터를 전송하였다. 그리고 실험 시작 20초부터 30초까지는 서브플로우 2의 지연시간을 10ms에서 2000ms까지 증가시키고 30초부터 40초까지는 다시 10ms로 감소시켰다. 그 후 60초부터 80초까지는 서브플로우 1의 지연시간을 서브플로우 2 와 동일하게 변화를 주었다. 80초 이후에는 초기 설정으로 100초까지 실험을 진행하였다. 해당 실험을 각 스케줄러 별로 실험을 진행하였고 그림 10을 통해서 각 스케줄러 별 성능을 알 수 있었다.

먼저 MPTCP round-robin 스케줄러는 0초부터 100초까지 평균 58.0Mbit/s의 전송속도를 보여주었다. 이는 한쪽 서브플로우에 bufferbloat와 같은 지연시간이 갑자기 증가하게 된 경우에도 해당 서브플로우에 데이터를 전송하여 수신측 수신버퍼의 제한이 발생하기 때문이다. 그림 10 (a)의 round-robin 스케줄러의 그래프를 보면 20초에서 40초, 60초에서 80초 동안 지연시간이 커진 서브플로우뿐만 아니라 다른 서브플로우의 성능도 저하되어 20초~40초 구간과 60초~80초 구간의 성능이 평균 19.5Mbit/s로 단일 TCP 성능인 40.8Mbit/s보다 좋지 않은 성능이 나왔다.



(a) MPTCP round-robin schduler

(b) MPTCP low-RTT schdule

(c) MPTCP DAPB schduler

그림 10. 이종 네트워크에서 스케줄러 성능 그래프
Fig. 10. Scheduler performance graph in heterogeneous networks.

다음으로 그림 10 (b)의 MPTCP LowRTT 스케줄러는 평균 64.6Mbit/s의 전송속도를 보여주었다. LowRTT 스케줄러는 RTT를 기반으로 빠른 서브플로우에 데이터를 주로 전송하고 느린 서브플로우에는 데이터를 소량 전송한다. 하지만 느린 서브플로우에 전송되는 소량의 데이터로 인하여 그림 10 (b)를 보면 20초~40초에 서브플로우 1의 성능과 60초~80초의 서브플로우 2의 성능이 불안정한 모습을 보이며 평균 38.91Mbit/s로 단일 TCP성능보다 낮게 나왔다. 하지만 그림 10 (c)의 제안한 DAPB 스케줄러의 경우 가장 안정된 성능 그래프를 보여주는데 20초~40초구간의 서브플로우 2의 지연시간이 증가하였을 때 서브플로우 2의 경로를 차단하고

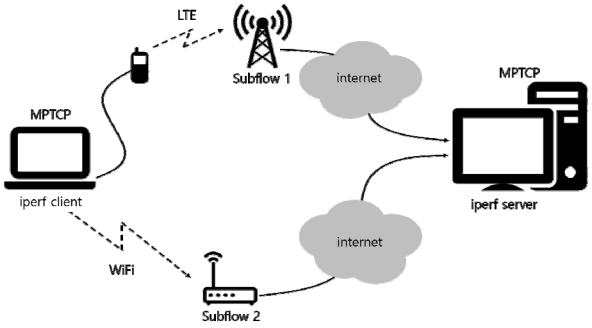


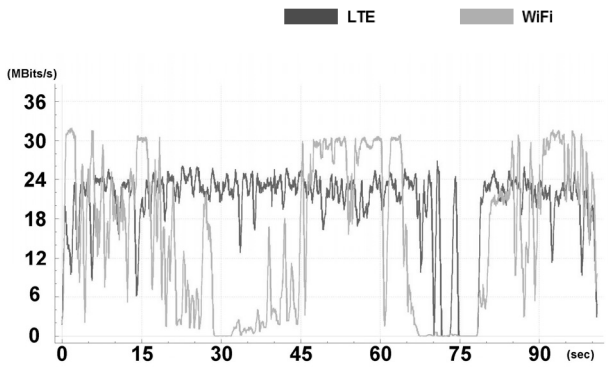
그림 11. MPTCP 무선네트워크 구성
Fig. 11. MPTCP wireless network topology.

서브플로우 1을 통하여 패킷을 전송한다. 40초 이후 서브플로우 2의 지연시간이 10ms로 줄어들어 다시 서브플로우 2 경로 차단을 해제하고 서브플로우 1과 서브플로우 2를 사용하여 동시에 데이터를 전송한다. 60초~80초 구간에서 서브플로우 1의 지연시간이 증가하여 서브플로우 1이 차단되고 서브플로우 2를 통해서 데이터 전송을 진행하다가 80초 이후 서브플로우 1의 지연시간이 10ms로 변경되었을 때 다시 양쪽 서브플로우로 데이터를 전송하게 된다. 제안한 스케줄러는 평균 68.7 Mbit/s라는 가장 높은 전송속도를 보여주었으며 buffer bloat가 발생한 20초~40초 구간과 60초~80초 구간에서 최소 단일TCP의 성능을 보장하였다.

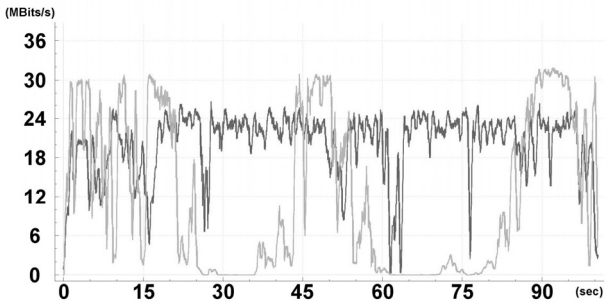
Ⅴ. 실제 무선네트워크에서 성능 분석

1. 실험환경

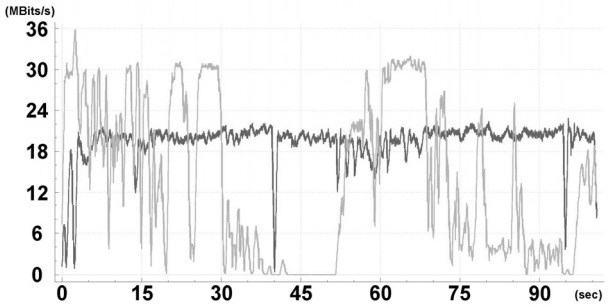
앞서 살펴본 테스트베드에서의 성능이 아닌 실제 무선네트워크에서의 성능 분석을 위해서 그림 11의 네트워크 구성을 하였다. MPTCP의 서버와 클라이언트는 Ubuntu 14.04 운영체제에서 동작되고, MPTCP 클라이언트는 무선랜카드를 이용하여 802.11n을 지원하는 AP에 연결을 하였다. 추가적으로 USB테더링을 사용하여 LTE에 연결하였다. MPTCP 서버와 클라이언트는 기본적으로 OLIA 혼잡제어를 사용하며, MPTCP 스케줄러의 경우 제안하는 DAPB 스케줄러와 성능을 비교하기 위해 기본적으로 제공하는 LowRTT와 round-robin을 사용하여 실험을 진행하였다. 본 논문에서는 iperf를 사용하여 네트워크 성능을 분석하였으며 실험진행은 WiFi 성능과 LTE 성능이 비슷한 성능을 지니는 위치에서 시작하여 실험중간에 WiFi 수신감도가 약한 위치로 이동하여 WiFi 서브플로우의 지연과 패킷 손실률을 높이고 다시 원위치로 돌아오는 과정을 반복하였다.



(a) MPTCP round-robin schduler



(b) MPTCP low-RTT schduler



(c) MPTCP DAPB schduler

그림 10. 무선 네트워크에서 스케줄러 성능 그래프
Fig. 10. Scheduler performance graph in wireless network.

2. 실험결과

LTE와 WiFi를 사용한 무선 네트워크에서 다중경로를 구성하여 기존 MPTCP 스케줄러와 제안한 DAPB 스케줄러의 성능을 비교 분석하였다. 실험은 서브플로우 1을 LTE로 사용하고 서브플로우 2를 WiFi를 사용하였다. 그리고 LTE의 성능이 변하지 않는 일정한 장소 내에서 WiFi의 수신감도가 강한 -60dBm인 장소에서 수신감도 약한 -90dBm인 장소로 이동을 하여 WiFi의 성능에 변화를 주어 실험을 진행하였다. 해당 실험을 각 스케줄러 별로 실험을 진행하였고 그림 10을 통해서 각 스케줄러 별 성능을 알 수 있었다.

먼저 MPTCP round-robin 스케줄러로 100초간 iperf를 사용하여 실험을 하였을 때 60초부터 WiFi의 성능이 낮아지기 시작하여 70초~75초 구간에서 수신버퍼의 제약으로 인해 LTE의 성능도 줄어드는 결과를 얻을 수 있었다. 다음으로 LowRTT 스케줄러를 이용하여 동일한 실험을 하였을 때 20초와 50초 구간에서 WiFi의 성능이 줄어들어 점차 보내는 패킷 양을 줄이지만 지연시간이 크고 패킷 손실률이 높은 WiFi 서브플로우를 통하여 데이터를 전송하기 때문에 25초와 60초 구간에서 LTE의 성능이 줄어드는 것을 볼 수 있었다. 마지막으로 DAPB 스케줄러의 경우 40초 구간에 WiFi의 성능이 좋지 않아 RTT가 증가할 경우 WiFi 서브플로우를 차단하여 LTE 서브플로우 성능저하를 최소화 시켜 LTE의 성능을 일정하게 유지시키는 결과를 얻을 수 있었다.

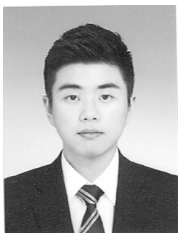
V. 결 론

본 논문에서는 이종 망에서 MPTCP가 동작할 때 경로간의 지연시간의 차이로 인해 발생하는 MPTCP 성능저하를 해결하기 위해서 지연시간을 이용하여 경로를 차단하는 DAPB 스케줄러를 제안하였다. 제안된 스케줄링 알고리즘은 각각의 네트워크의 지연시간을 측정하고 이를 통해서 네트워크의 혼잡을 구분하는 일정 기준값을 선정하였다. 그리고 기준 값을 바탕으로 서브플로우 네트워크가 기준 값 이상으로 혼잡하게 되면 해당 경로를 차단하여 수신버퍼의 제한으로 인한 다른 서브플로우의 성능저하를 최소화 시켜 MPTCP의 경로상황이 좋지 않더라도 가장 좋은 경로의 TCP 성능을 보장하게 되어 안정적인 성능을 유지할 수 있다. 또한 기존의 경로를 차단하거나 느린 경로에 패널티를 부여하는 스케줄링의 단점을 보완하기 위해서 차단한 경로에는 다른 서브플로우의 성능에 영향을 주지 않는 복제패킷을 전송하여 네트워크의 혼잡을 지속적으로 측정하고 혼잡한 네트워크의 상황이 좋아질 경우 다시 다중경로를 사용하여 MPTCP의 성능을 발휘 할 수 있다.
본 논문에서는 유선 테스트베드를 통하여 임의적으로 네트워크 파라미터를 조정하여 실험을 하고 실제 무선 네트워크에 적용하여 실험을 하였다. 차후 연구에는 이동단말에 MPTCP를 설치하고 제안한 스케줄러를 이용하여 실험을 할 예정이다.

REFERENCES

[1] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, “TCP Extensions for Multipath Operation With Multiple Addresses”, IETF document RFC 6824, Jan 2013.
[2] R. Stewart, “Stream Control Tramsmission Protocol”, IETF document RFC 4960, Sep 2007.
[3] S. C. Nguyen, and T. M. T. Nguyen, “Evaluation of multipath TCP load sharing with coupled congestion control option in heterogenous”, IEEE GIIS, Aug 2011.
[4] C. Paasch and O. Bonaventure, “Multipath TCP : Decoupled from IP, TCP is at last able to support multihomed hosts”, Communications of the ACM, Vol 12, Issue 4, pp. 51-57, Apr 2014.
[5] J. Gettys, and K. Nichols, “Bufferbloat: Dark buffers in the Internet”, Communications of the ACM, Vol 55, Issue 1, pp. 57-65, Jan 2012.
[6] H. Jiang, Y. Wang, K. Lee, and I. Rhee, “Tackling bufferbloat in 3G/4G networks”, in Proc. of ACM Conf. on IMC, pp. 329-342, Nov 2012.
[7] M. Scharf, and S. Kiesel, “Head-of-line Blcoking in TCP and SCTP: Analysis and Measurements”, IEEE GLOBECOM, pp. 1-5, Nov 2006.
[8] S. Ferlin, T. Dreibholz, and O. Alay, “Multi-path transport over heterogeneous wireless networks: Does it really pay off?”, IEEE GLOBECOM, pp. 1-6, Dec 2010.
[9] C. Paasch, R. Khalili, and O. Bonaventure, “On the benefits of applying experimental design to improve multipath TCP”, in Proc. of 9th ACM Conf. Emerg. Netw. Experim. Technol, pp. 393-398, Dec 2013.
[10] C. Raiciu et al, “How hard can it be? Designing and implementing a deployable multipath TCP”, in Proc. of 9th USENIX. on NSDI, pp. 29-29, Apr 2012.
[11] S. H. Baidya, and R. Prakash, “Improving the performance of multipath TCP over heterogeneous paths using slow path adaptiation”, IEEE ICC, pp. 3222-3227, Jun 2014.
[12] M. Carbone, and L. Rizzo, “Dummynet Revisited”, ACM SIGCOMM CCR, Vol 40, Issue 2, pp. 12-20, Apr 2010.
[13] MultiPath TCP Linux Kernel implementaion, http://mptcp.info.ucl.ac.be/.
[14] R. Khalili, N. Gast, M. Popovic, and J. Le Boudec, “MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution”, IEEE/ACM Transactions on Networking, Vol 21, Issue 5, pp. 1651-1665, Oct 2013.
[15] iPerf - The TCP, UDP and SCTP network band width measurement tool, https://iperf.fr/
[16] JH. Hwang, and J. Yoo, “Packet scheduling for Multipath TCP”, in Proc. 7th International Conf. on Ubiquitous and Future Networks, pp. 177-179, July 2015.

저 자 소 개



김 민 섭(학생회원)
2016년 충남대학교 정보통신공학과 학사
2016년～현재 충남대학교 전자전파정보통신공학과 석사과정
<주관심분야: TCP, 이동통신 네트워크, 데이터 통신, MPTCP>

이 재 용(평생회원)
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
<주관심분야: 데이터 통신, 인터넷, 네트워크 성능분석>

김 병 철(평생회원)
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
<주관심분야: 이동인터넷, 이동통신 네트워크, 데이터 통신>