http://dx.doi.org/10.7840/kics.2015.40.1.70

공유된 병목 링크를 경유하는 MPTCP의 성능 평가

뉴옌 반 딘◆, 노 승 환°

Performance Evaluation of MPTCP over Shared Bottleneck Link

Nguyen Van Dien◆, Soonghwan Ro°

요 약

본 논문에서는 TCP와 병목 링크(bottleneck link)를 공유하는 MPTCP(Multipath TCP)의 성능을 실험을 통해서 측정하고 결과를 제시하였다. 실험을 통해서 MPTCP의 공정성(fairness)과 성능은 지연, 패킷 손실 등과 같은 전체 네트워크의 조건에 크게 영향을 받는다는 것을 알 수 있었다. 이상적인 일반적인 네트워크 조건에서 MPTCP는 TCP보다 상대적으로 성능이 높아서 TCP와 불공정(unfair) 하지만, 네트워크의 지연이 증가하고 패킷 손실이 증가함에 따라 처리율(throughput)이 감소하며 TCP보다도 처리율이 감소하는 것을 알 수 있었다.

Key Words : Multipath TCP, fairness, latency, throughput, Jain's Fairness Index(JFI)

ABSTRACT

In this paper, we present experimental results evaluating the performance of the Multipath TCP over shared bottleneck path in series of benchmark tests. In summary, we find that the Multipath TCP's fairness as well as its competitive responds to the change of network conditions such as latency, loss rate… MPTCP is extremely unfair and powerful with regular TCP in ideal network conditions but its throughput decreases clearly even less than regular TCP in worse network conditions with very high latency, higher packet loss rate.

Ⅰ. 서 론

최근 IETF에 의해서 Multipath TCP(MPTCP) 프로토콜이 표준화 되었으며 구현된 프로토콜을 리눅스 커널에 설치할 수 있다[1]. MPTCP는 두 호스트 사이에 여러 경로 사이의 응용계층에서 실행이 가능하며 동시에 단일 TCP 접속도 허용한다. 아직 인터넷 트래픽의 95% 이상이 TCP에 의해서 수행이 되고 있다. 따라서 MPTCP는 기존의 TCP가 존재하는 경로 상에서 함께 동작을 해야 한다. 결과적으로 기존의 TCP와 함께 동작을 할 경우에 MPTCP의 성능을 포함하여 어떻게 동작을 할 것인가에 대한 고찰과 기존의 TCP에 미치는 영향 등을 고려해야 한다. 따라서 MPTCP가 실용화되기 위해서는 다양한 네트워크 환경에서 MPTCP의 성능과 전체 네트워크에 미치는 영향을 평가하는 것이 매우 중요하다.

이러한 문제를 해결하기 위해서 본 논문에서는 일반적인 TCP와 MPTCP가 병목 구간(bottleneck path)을 공유하는 경로에서 트래픽 경쟁을 하는 테스트 환경과 시나리오를 구성하였다. 또한 실제 네트워크 환경과 조건을 동일하게 하기 위해 지연 및 패킷 손실 등과 같은 파라미터를 변화시켜 가며 처리율과 공정성을 측정하였다.

※ 본 연구는 2014년 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임.(2011-0012003)
◆ First Author : Kongju University, Department of Information and Communication Engineering, diennv@kongju.ac.kr, 학생회원
° Corresponding Author : Kongju University, Department of Information and Communication Engineering, rosh@kongju.ac.kr, 종신회원
논문번호 : KICS2014-11-464, Received November 19, 2014; Revised December 12, 2014; Accepted December 12, 2014

Ⅱ. 연구 배경

최근 multi-homing과 Multipath TCP 주제가 많은 관심을 받고 있으며 IETF에 의해서 워킹 그룹(working group)에 의해 표준화 되었다. SCTP (Streaming Control Transmission Protocol)[2]는 multi-homing을 고려해서 경로가 실패하는 경우 문제를 해결할 수 있도록 설계되었다. SCTP는 호스트가 여러 경로를 동시에 사용할 수 있도록 하고 있다. 그러나 SCTP가 다양한 운영 체제에서 구현이 되었음에도 불구하고[3], 일부 응용에서만 사용되며 보편적으로 사용되고 있지 않다. SCTP가 글로벌 인터넷에서 사용될 때의 첫 번째 단점은 개발자가 SCTP를 사용할 수 있도록 응용 프로그램을 변경해야 한다는 것이며, 두 번째로 NAT 또는 방화벽(firewall)과 같은 다양한 네트워크 장비들이 SCTP를 인식하지 못하고 SCTP 패킷을 차단한다는 것이다.

IETF의 MPTCP 워킹그룹에서는 2012년부터 기존의 TCP를 다중 경로로 확장하는 개발을 진행하였으며[4], 이 확장으로 인해 호스트가 여러 개의 인터페이스를 통해서 하나의 접속에 속하는 패킷을 여러 개의 경로로 전송할 수 있게 한다. 또한 이러한 TCP의 확장은 SCTP의 단점인 응용 프로그램을 변경하거나 경로상의 네트워크 장비에서 차단되는 문제들을 해결할 수 있었다. 이러한 구현을 기반으로 다양한 파라미터가 MPTCP에 미치는 영향에 대한 많은 연구가 진행되었다[5]. 따라서 이러한 영향에 대해서 이해를 하는 것은 추후 MPTCP의 성능을 향상시키기 위해서 매우 유용하게 사용될 수 있다. 비록 실험실 내에서 진행되는 실험에는 많은 제약이 있을 수 있으나 본 논문의 결과를 통해서 새로운 전달 계층의 프로토콜인 MPTCP의 성능을 예측하는데 많은 도움을 줄 수 있을 것으로 기대한다.

Ⅲ. MPTCP의 개요

3.1 MPTCP의 구조

MPTCP[4]는 트래픽을 전달하기 위한 새로운 전달 계층 프로토콜이다. MPTCP의 특징은 multi home을 갖는 호스트가 여러 개의 인터페이스를 이용해서 단일 TCP 접속을 확장을 하는 것이다. 이러한 확장은 다음과 같은 장점을 제공한다. MPTCP 프로토콜에서는 하나의 링크로 전송하는 것이 실패를 해도 혼잡제어를 이용해서 트래픽을 혼잡된 경로에서 다른 경로로 전환함으로써 혼잡을 감소시킬 수 있다.

둘째, 병렬로 추가되는 경로와 인터페이스로 효율을 높일 수 있다. MPTCP의 설계는 많은 요구사항의 영향을 받았으며, 그 가운에 가장 중요한 것은 응용 계층과 네트워크의 호환성이다. 응용 계층의 호환성은 TCP에서 실행되는 응용 프로그램은 변경 없이도 MPTCP에서도 실행이 된다는 것을 의미한다. 또한 네트워크의 호환성의 목적은 MPTCP 패킷이 경로상의 네트워크 장치들을 안전하게 통과해야 한다는 것이다. 따라서 MPTCP는 응용계층에 기존의 TCP 접속과 동일하게 보이도록 하기 위해 추가적인 옵션으로 기존의 TCP를 변경하는 방식으로 설계되었다. 그러나 네트워크 계층에서는 MPTCP의 각 서브 플로우(sub-flow)는 각 세그먼트가 새로운 TCP 옵션을 전달하는 TCP 플로우로 인식된다. MPTCP는 이러한 서브플로우의 생성, 제거 및 데이터를 전달하기 위한 활용을 관리한다. 하나의 MPTCP내에서 운영되는 서브플로우의 수는 고정되어 있지 않으며, TCP 접속이 유지되는 동안에 변경될 수 있다. 그림 1은 MPTCP 프로토콜 스택을 보여준다.

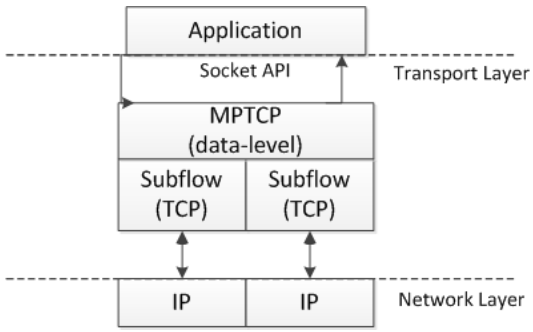

그림 1. MPTCP 프로토콜 스택
Fig. 1. MPTCP Protocol Stack

3.2 MPTCP의 동작

MPTCP는 TCP의 확장으로 구성되었으며 RFC6824[4]로 표준화 되었다. MPTCP에서는 단일 TCP 접속으로 다른 IP 주소로 데이터를 동시에 송수신이 가능하게 하였다. 본 절에서는 이 동작에 관한 개요를 설명한다. 일반적인 TCP와 마찬가지로 MPTCP 세션은 크게 초기 접속, 데이터 전송 및 접속 종료의 세 단계로 구분된다. 초기 접속 단계에서 TCP와 MPTCP의 차이는 다중 경로가 설정되기 전에 4-way 핸드셰이크 동작이 이루어진다. 우선 MPTCP의 4-way 핸드셰이크에서는 TCP의 3-way 핸드셰이크와 유사하게 동작하지만 SYN, SYN/ACK 그리고 ACK 패킷은 MPTCP_CAPABLE 옵션을 동반한다. 그 다음에

TCP 접속이 이루어지면 클라이언트는 자신이 가지고 있는 다른 주소를 TCP 세그먼트에 ADD_ADDR 옵션으로 전송한다.

새로운 주소의 교환으로 두 번째 TCP 접속이 생성되며, 이 접속이 새로운 서브플로우가 된다. 이 서브플로우는 초기 서브플로우에서 서버로부터 전송된 토큰을 포함하는 MP_JOIN 옵션을 갖는 SYN 세그먼트를 전송함으로써 첫 번째 서브플로우와 링크된다. 서버는 초기 서브플로우에서 클라이언트로부터 선택된 토큰을 포함하는 MP_JOIN 옵션을 갖는 SYN+ACK 패킷으로 응답하며, 클라이언트는 3-way 핸드셰이크를 종료한다. 그림 2는 이러한 초기 MPTCP 접속과정을 보여준다.

이러한 두 개의 서브플로우는 하나의 MPTCP 접속 내부에서 서로 링크되어 있으며, 두 번째 단계에서 데이터를 전송하기 위해서 사용될 수 있다. MPTCP가 데이터 전송을 시작하면 언제든지 생성 또는 제거될 수 있는 서브플로우를 통해서 신뢰성 있고 순서가 바뀌지 않는 데이터 전송을 위해 두 가지의 원칙이 적용된다. 첫 째 각 서브플로우는 일반적인 TCP 접속과 동일하며, 32비트의 시퀀스 번호 공간(sequence numbering space)을 갖는다. 이것은 MPTCP가 프록시 또는 트래픽 전처리 장치(traffic normalizer)와 같은 경로상의 복잡한 네트워크 장치를 통과할 수 있도록 하는데 매우 중요하다. 둘째로 MPTCP는 64비트의 시퀀스 번호 공간을 가지며, DSN_MAP와 DSN_ACK의 두 개의 TCP 옵션에서 이 데이터 시퀀스 번호를 사용한다. 호스트에서 각 서브플로우로 TCP 세그먼트를 전송할 때 64비트와 서브플로우에서 사용되는 32비트 시퀀스 번호를 매핑하기 위해서 DSN_MAP 옵션을 사용한다. 이러한 방법으로 다른 서브플로우로 전송된 데이터 순서가 바뀌어서 데이터 전송에 실패를 했을 때 다른 서브플로우에서 재전송이 가능하다. MPTCP에서 수신된 세그먼트는 두 개의 레벨에서 응답(acknowledgement)되어 진다. 첫 째로 각 서브플로우에서 수신된 세그먼트에 대한 응답으로 각 서브플로우에서 TCP 누적(cumulative) 또는 선택적(selective)인 응답(acknowledgement)이 사용된다. 그리고 두 번째로 수신 호스트로부터 누적 응답을 제공하기 위해서 데이터 시퀀스 레벨에서 DSN_ACK 옵션이 리턴 된다. 세그먼트가 손실되었을 때 수신측에서 수신된 32비트 시퀀스 번호를 감지하고 기존의 TCP 재전송 메카니즘을 이용해서 손실을 복구한다. 서브플로우가 실패하면 MPTCP는 이 실패를 감지하고 응답이 없는 데이터를 접속이 되어 있는 서브플로우를 통해서 재전송한다.

송신자 더 이상 전송할 데이터가 없다고 통보를 할 때에는 데이터 시퀀스 신호의 일부로 FIN 옵션이 송신된다. 이 신호는 일반적인 TCP의 FIN과 같은 의미를 가지면 동일하게 동작하지만 접속 레벨(connection level)에 속한다. 일단 MPTCP 접속의 모든 데이터가 수신되면 이 메시지에 대해서 접속 레벨(connection level)에서 DATA_ACK 메시지로 응답(acknowledge)된다. 일단 호스트의 DATA_FINS가 DATA_ACKs로 응답되어지면 접속은 종료된 것으로 간주된다. 그리고 두 호스트가 모든 서브플로우에게 FINs를 전송해야 한다. 또한 MPTCP와 기존의 TCP의 큰 차이는 혼잡제어 방식이다. MPTCP에서는 TCP 플로우와 불공정(unfair) 되는 것을 막기 위해서 TCP의 혼잡제어

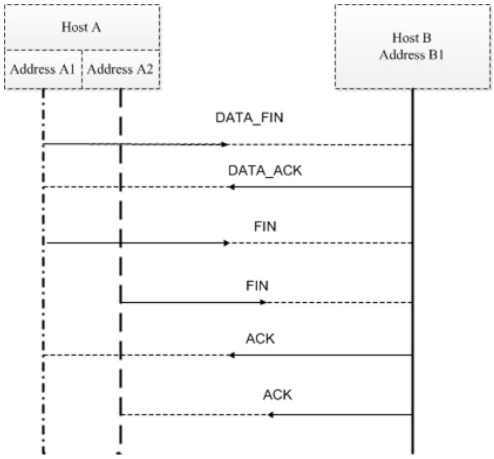

그림 2. 초기 MPTCP 접속 과정
Fig. 2. Initial Connection Process of MPTCP

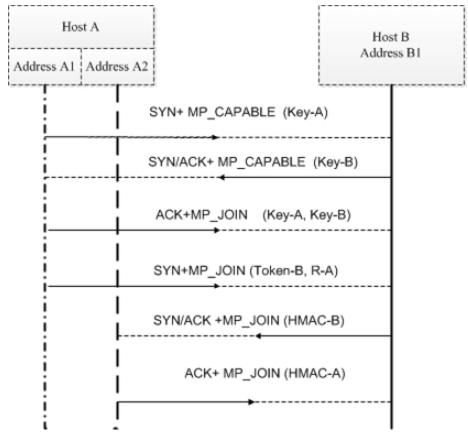

그림 3. MPTCP 접속 종료
Fig. 3. Releasing of MPTCP Connection

표 1. MPTCP 옵션
Table 1. MPTCP Options

옵 션	설 명
MPTCP_CAPABLE	Multipath capable
MP_JOIN	Join connection
DATA_SEQUENCE_SIGNAL	Data Ack and Data Sequence Mapping
ADD_ADDR	Add address
REMOVE_ADDR	Remove Address

방식을 사용하면 안 되며, 특히 두 프로토콜이 병목 링크를 공유하는 경우에 심각한 문제가 발생할 수 있다. 이 문제를 해결하기 위해서 LIA(Linked Increases Algorithms)[7]이 제안되었다. 이 제안은 혼잡제어 알고리즘을 이용하고 있으며, 많은 연구에서 LIA의 유효성을 입증했다. 따라서 본 논문에서는 LIA 알고리즘을 이용해서 실험을 실시하였다.

Ⅳ. 실험환경 구성

4.1 실험환경

본 논문에서는 MPTCP의 처리율(throughput)과 공정성 지수(fairness index)를 측정하기 위해 테스트 환경을 구성하여 테스트를 실시하였다. 테스트 베드에서는 MPTCP 사용자와 TCP 사용자가 병목 링크를 공유하도록 구성되었다. 또한 네트워크에 지연(latency)과 패킷 손실(packet loss)과 같은 부하를 주기 위해서 dummy-net 브리지[9]를 사용하였다. 인터넷 접속에서 100ms 이하의 지연은 보통 수준이지만 일반적으로 25ms 이하의 지연이 요구된다. 또한 위성을 통한 인터넷의 평균 지연은 500ms 또는 그 이상이다[6]. 따라서 본 논문에서는 이러한 지연 구간에 대해서 실험을 실시하였다. 패킷이 손실되었다는 것은 혼잡상황이 발생했다는 것으로 간주된다. 일반적으로 1% 미만의 패킷 손실 율은 양호한 것으로, 1~2.5%의 패킷 손실 율은 허용 수준인 것으로 각각 간주된다. 따라서 테스트 베드에 패킷 손실 율은 0.005~0.03까지 4단계에 걸쳐서 측정을 하였다. 본 논문에서 공정성을 측정하기 위해서 다음과 같이 주어지는 평균 사용자 처리율을 위한 Jain's Fairness Index(JFI)[8]을 사용하였다.

$$\text{Fairness(throughput)} = \frac{\left\{\sum_i^n T_i\right\}^2}{n * \left\{\sum_i^n T_i^2\right\}} \quad (1)$$

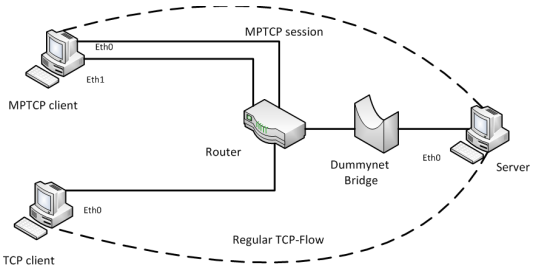

그림 4. 테스트 베드의 네트워크 토폴로지
Fig. 4. Network Topology for Test Bed

그림 4의 테스트 베드에서 모든 링크는 100Mbps의 대역폭을 갖는다. 그리고 dummy-net 브리지는 다양한 RTT(Return Round Trip) 지연시간과 패킷 손실 율을 갖도록 설정할 수 있어서 다양한 네트워크 조건을 구성할 수 있다. 또한 IP address filter(ipfw)는 각각의 서브플로우에 독립적으로 영향을 미칠 수 있다. MPTCP의 서버와 클라이언트는 Ubuntu 13.10 운영체제에서 동작되고, TCP는 클라이언트는 Reno 알고리즘을 사용하며, MPTCP는 표준화된 혼잡제어 방식인 LIA 알고리즘을 사용한다. 또한 네트워크의 성능을 측정하기 위해서 iperf[10]를 사용한다. 본 논문에서는 다른 환경을 고려하기 위해서 두 종류의 네트워크 동종(homogeneous)과 이종(heterogeneous) 네트워크를 각각 구성하였다. 동종 네트워크에서는 모든 서브플로우의 지연과 패킷 손실 율을 동일하게 하였지만 이종 네트워크에서는 두 서브플로우의 조건을 서로 다르게 하기 위해서 브리지를 사용해서 필터를 추가하였다. 즉, 첫 번째 서브플로우의 지연과 패킷 손실 율을 변경하였으며, 두 번째 서브플로우와 TCP 접속은 변경하지 않아서 두 서브플로우의 지연과 패킷 손실 율이 다르게 하였다. 따라서 두 번째 서브플로우와 TCP 접속은 0.5ms 미만의 지연과 패킷 손실이 없는 양호한 상태를 항상 유지하지만 첫 번째 서브플로우는 양호한 상태에서 그렇지 않은 상태로 변화한다. 동일한 파라미터를 사용했을 때 5번씩의 측정이 이루어졌으며, 각 측정은 10분간 지속되었고, 평균 측정결과를 나타내었다.

4.2 동종 네트워크에서의 MPTCP 성능

4.2.1 RTT에 따른 MPTCP의 성능

첫 번째 테스트는 네트워크의 지연시간을 변경시키면서 실시되었다. TCP와 MPTCP 클라이언트는 동시에 시작과 종료되었으며 각각의 접속에 대한 처리율을 송신자 측에서 측정하였다. 그림 5에는 측정된 처

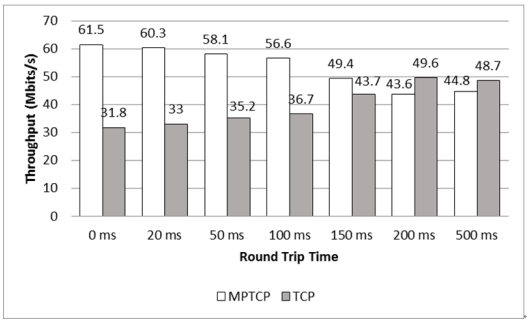

그림 5. RTT에 따른 MPTCP와 TCP 처리율
Fig. 5. Throughput of MPTCP and TCP in Terms of RTT

리율을 보여준다. 그림 5에서 지연이 100ms 보다 작은 범위의 양호한 상태에서는 MPTCP의 성능이 TCP 보다 거의 2배 정도 높은 것으로 나타난다. 그러나 RTT가 증가함에 따라 MPTCP의 성능이 TCP와 비슷해지며, 심지어 RTT가 200ms에 가까워지면 MPTCP의 성능이 TCP의 성능에 떨어지는 것을 알 수 있다. 각 프로토콜의 처리율로부터 JFI를 그림 6과 같이 계산할 수 있다. 그림 6에서 네트워크의 상태가 나쁠 때 MPTCP의 공정성은 1에 가까워져서 가장 좋은 것을 알 수 있다. 그림 5와 그림 6에서와 같이 전파 지연시간이 증가함에 따라 MPTCP의 성능은 감소하는 것을 알 수 있다.

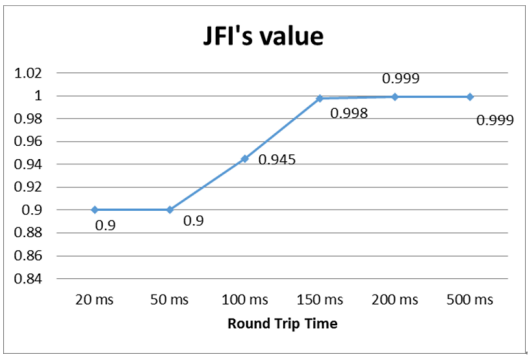

그림 6. RTT에 따른 JFI 변화
Fig. 6. JFI Value in Terms of RTT

4.2.2 패킷 손실 율에 따른 MPTCP의 성능

이 실험은 첫 번째 실험과 유사하지만 네트워크의 지연시간 대신에 패킷 손실 율을 변경시키면서 실험을 수행하였다. 처리율과 계산된 JFI를 그림 7과 그림 8에서 각각 보여주고 있다. 두 그림의 형태는 첫 번째 결과와 매우 유사하며, 이는 네트워크가 양호한 상태

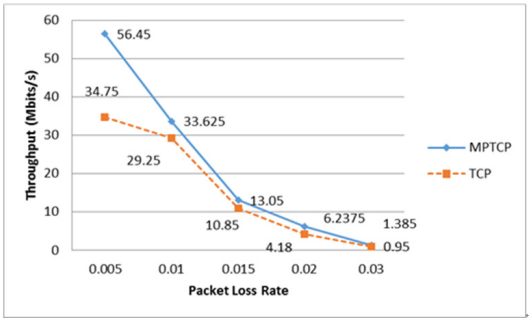

그림 7. 패킷 손실 율에 따른 MPTCP와 TCP의 처리율
Fig. 7. Throughput of MPTCP and TCP in Terms of Packet Loss Rate

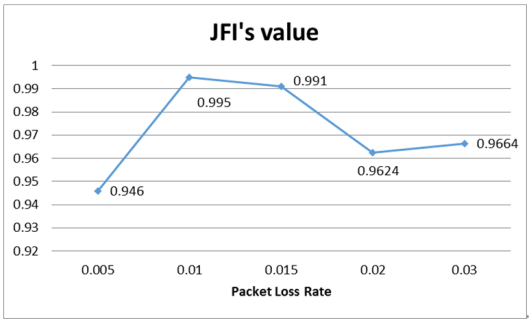

그림 8. 패킷 손실 율에 따른 JFI의 변화
Fig. 8. JFI Value in Terms of Packet Loss Rate

에서 MPTCP이 경쟁력이 있다는 것을 보여준다.

4.3 이종 네트워크에서의 MPTCP 성능

4.3.1 RTT에 따른 MPTCP의 성능

이 실험에서는 첫 번째 서브플로우의 지연만을 변경시키면서 실험을 수행하였다. 그림 9에서 두 서브플로우의 서로 다른 지연 시간이 MPTCP의 처리율에

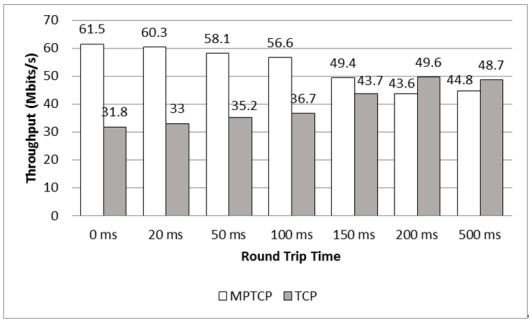

그림 9. 서브플로우의 지연 시간 불균형에 따른 MPTCP와 TCP의 처리율
Fig. 9. Throughput of MPTCP and TCP in Terms of Unbalanced Latency between Sub-flows

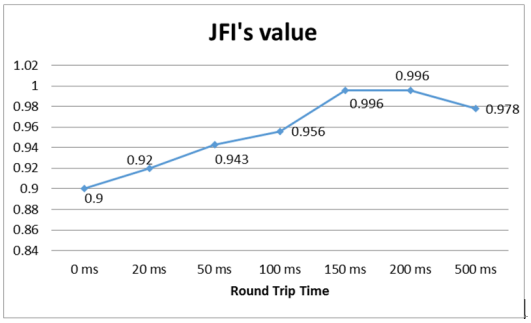

그림 10. 서브플로우의 지연시간 불균형에 따른 JFI 변화
Fig. 10. JFI Value in Terms of Unbalanced Latency between Sub-flows

영향을 미치는 것을 알 수 있다. 구체적으로, 두 서브플로우의 지연 시간의 차가 100ms보다 작으면 MPTCP 접속의 처리율이 TCP보다 높은 것을 알 수 있다.

그러나 한 서브플로우의 RTT가 크게 증가하면서 점차적으로 MPTCP가 TCP와 공정하게 되는 것을 알 수 있다. 특히 MPTCP의 처리율을 두 서브플로우 사이의 지연시간이 200ms 이상으로 커질수록 감소하는 것을 알 수 있다. 그림 10의 JFI에서도 두 서브플로우의 차이가 작을수록 공정성도 함께 감소하는 것을 알 수 있다.

4.3.2 패킷 손실 율에 따른 MPTCP의 성능

동종 네트워크의 측정과 유사한 방법으로 패킷 손실 율이 이종 네트워크에 미치는 영향을 측정하였다. 이 결과는 그림 11과 그림 12에 나타내었다. 또한 MPTCP는 일반 네트워크 상황에서는 매우 효율적이라는 것을 알 수 있었다. 그러나 하나의 서브플로우의 성능이 떨어지면서 MPTCP의 처리율이 감소하여 특

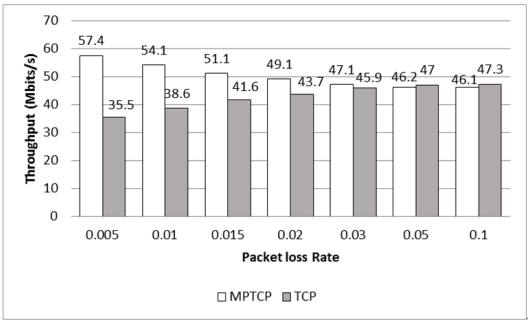

그림 11. 서브플로우의 패킷 손실 율 불균형에 따른 MPTCP와 TCP 처리율
Fig. 11. Throughput of MPTCP and TCP in Terms of Unbalanced Packet Loss Rate between Sub-flows

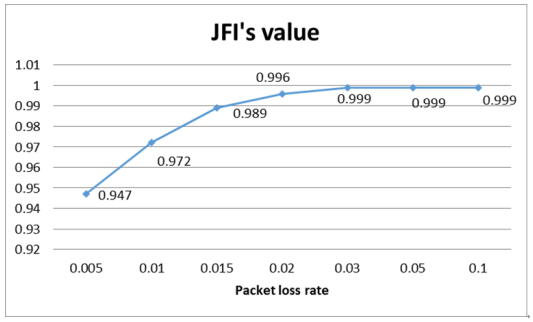

그림 12. 서브플로우의 지연시간 불균형에 따른 JFI 변화
Fig. 12. JFI Value in Terms of Unbalanced Packet Loss Rate between Sub-flows

정한 조건에서는 TCP의 처리율보다도 낮아지는 것을 알 수 있다.

모든 실험으로부터 MPTCP의 동작이 동종 네트워크와 이종 네트워크에서 동일하다는 것을 알 수 있다. 병목 링크를 공유하는 네트워크에서 네트워크의 전파 지연이 양호한 상태에서는 MPTCP의 처리율이 높으며, TCP에 비교해서 불공정하다는 것을 알 수 있다. 그러나 전파지연이 변화하면서 점차 MPTCP의 처리율과 공정성이 나빠지는 것을 알 수 있다. 그 원인에는 두 가지 이유가 있다. 첫 째로 이유는 혼잡제어가 직접적으로 관련되어 있다. 본 논문에서는 MPTCP 혼잡제어의 표준인 LIA[7]을 사용하였으며, 이 알고리즘에서 혼잡 윈도우의 크기는 다음과 같다.

$$w_r = \begin{cases} w_r + \min\left(\frac{N_{bytes} * mss_r}{w_r}, \frac{a * N_{bytes} * mss_r}{\sum_{r=0}^{n} w_r}\right), ACK event \\ w_r * 0.5 \qquad\qquad\qquad\qquad\qquad\qquad , loss\ event \end{cases} \quad (2)$$

식 (2)에서 α는 TCP 플로우에 나쁜 영향을 주지 않기 위해서 사용되는 MPTCP 접속의 침입도(aggressiveness)를 통제하는 파라미터이며, 식 (3)과 같이 계산된다.

$$a = \sum_r W_r \frac{\max_r \frac{w_r}{srtt_r^2}}{(\sum_r \frac{w_r}{srtt_r})^2} \quad (3)$$

W_r : r_{th} 서브플로우의 혼잡 윈도우 크기
mas_r : r_{th} 서브플로우의 최대 세그먼트 크기
$srtt_r$: r_{th} 서브플로우의 완화된(smoothed) RTT(Round Trip Time)

식 (3)으로부터 $srtt_r$이 크면 a 값은 $1/w_r$(Reno의 증

가 지수(increase index))보다 작아진다. 예로 이종 네트워크에서 첫 번째 서브플로우의 지연시간이 매우 높게 설정된다면 두 번째 서브플로우와 TCP 접속의 지연시간은 감소하면서 같은 값을 갖는다. MPTCP의 성능은 다음과 같이 계산된다.

$$B \approx \frac{W_1(a)}{SRTT_1} + \frac{W_2(a)}{SRTT_2} \approx \frac{W_2(a)}{SRTT_2} \quad (4)$$

식 (4)에서 $W_1(a)$, $W_2(a)$는 각각 첫 번째와 두 번째 서브플로우 혼잡 윈도우 크기의 함수이다. 이러한 경우에 변수 α는 다음과 같다.

$$\alpha = \frac{W_2}{\left(W_2 + \frac{SRTT_2}{SRTT_1} * W_1\right)^2} \leq \frac{W_2}{W_2^2} = \frac{1}{W_2} \quad (5)$$

식 (5)의 경우에 MPTCP의 성능은 TCP의 처리율보다 낮을 것이며 측정결과와 일치한다. 또 다른 이유는 리눅스 내부의 MPTCP의 스케쥴러 모듈에 의해 발생한다. 리눅스의 스케쥴러에서는 가장 낮은 지연을 갖는 경로로 우선적으로 전송을 한다. 따라서 낮든 RTT를 갖는 서브플로우로 먼저 전송을 할당한 후에 다음으로 낮은 RTT를 갖는 서브플로우를 통해서 전송을 하려고 한다. 따라서 서브플로우간에 불균형이 심한 경우에 상태가 나쁜 링크는 데이터를 전송하는데 낮은 우선순위를 갖는다.

V. 결 론

본 논문에서는 MPTCP와 TCP 호스트가 병목 링크를 공유하는 테스트 베드를 구성하여 성능을 측정함으로써 MPTCP의 성능을 보여주었다. 측정 결과 LAN과 같이 낮은 지연과 패킷 손실 율을 갖는 정상적인 네트워크 조건에서는 MPTCP의 성능이 뛰어나다는 것을 알 수 있었다. 그러나 네트워크의 지연이 높거나 혼잡 정도가 심한 네트워크에서는 TCP에 비해서 성능이 높지 않다는 것을 알 수 있었다.

추후 연구로는 다른 혼잡 제어 방식에 의한 MPTCP의 성능을 관찰하고, 관찰한 결과 발생하는 문제를 해결하기 위한 새로운 알고리즘을 제안할 것이다. 따라서 본 논문에서 실시한 성능 측정을 바탕으로 LIA의 장점을 유지하면서 지연이 높은 네트워크에서 처리율을 개선하며, 서브플로우에서 지연이 서로 다른 서브플로우 사이에서 데이터의 이동을 원활하게 하는 알고리즘을 개발할 수 있을 것으로 기대한다.

References

[1] http://www.multipath-tcp.org/

[2] R. Stewart, *Stream control transmission protocol,* IETF RFC 4960, Sept. 2007.

[3] J. Iyengar, P. D. Amer, and R. R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw. (TON),* vol. 14, no. 5, pp. 951-964, Oct. 2006.

[4] C. Raiciu, M. Handley, and O. Bonaventure, *TCP extensions for multipath operation with multiple addresses*, RFC 6824, Jan. 2013.

[5] S. Barre, C. Paasch, and O. Bonaventure, "Multipath TCP : From theory to practice," in *Proc. Int. IFIP TC 6 Conf. Netw.*, vol. I, pp. 444-457, May 2011.

[6] http://compnetworking.about.com/od/speedtest/a/network_latency.htm

[7] C. Raiciu, M. Handley, and D. Wischik, *Coupled congestion control for multipath transport protocols*, RFC 6356, Oct. 2011.

[8] R. Jain, D. Chiu, and W. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer systems*, Technical Report DEC-TR-301, Digital Equipment Corporation, Maynard, Mass, USA, 1984.

[9] http://www.technogumbo.com/tutorials/Network-Bandwidth-Latency-and-Delay-Simulation-Tutorial/Network-Bandwidth-Latency-and-Delay-Simulation-Tutorial.php

[10] https://iperf.fr/

[11] W. Seok, M. Lee, and M. Lee, "Receiver-initiated slow start for improving TCP performance in vertical handoff," *J. KICS*, vol. 38B, no. 08, pp. 597-606, Aug. 2013.

[12] B.-H. Oh, H. Kim, and J. Lee, "Packet scheduling scheme and receiver-based recovery scheme for MPTCP in heterogeneous networks," *J. KICS*, vol. 37B, no. 11, Nov. 2012.

[13] T. Yim, Y. Kyung, T. M. Nguyen, G. Hong, and J. Park, "A fast and scalable mobile flow management method for IP-based mobile

networks," *J. KICS*, vol. 39B, no. 1, pp. 8-16, Jan. 2014.

뉴옌 반 디엔 (Nguyen Van Dien)

2013년 8월 : Electronics and Telecommunication on Engineering of Hanoi University of Science and Technology, Vietnam (공학사)
2013년 9월~현재 : 공주대학교 정보통신공학과 (공학석사과정)
<관심분야> OFDM, MPTCP, Mobility management

노 승 환 (Soonghwan Ro)

1987년 8월 : 고려대학교 전자공학과 (공학사)
1989년 8월 : 고려대학교 전자공학과 (공학석사)
1993년 8월 : 고려대학교 전자공학과 (공학박사)
1994년 3월~현재 : 공주대학교 정보통신공학부
<관심분야> 이동성 관리 프로토콜, 임베디드 시스템