https://doi.org/10.7840/kics.2023.48.5.612

다중 이동통신 사업자의 LTE 망을 활용한 Multipath TCP 성능 평가

김 응 협◆, 서 원 경*, 조 유 제°

Performance Evaluation of Multipath TCP Using Multiple LTE Internet Service Provider Networks

Eung-Hyup Kim◆, Won-Kyeong Seo*, You-Ze Cho°

요 약

인터넷 상에서 이용되는 콘텐츠들이 다양해지고 품질이 높아지면서 요구되는 네트워크 대역폭이 급증하였다. IETF에서는 전송계층 표준 프로토콜인 MPTCP (Multipath TCP)를 제안하며, 다중 인터페이스 기반의 여러 서브 플로우를 하나로 aggregation하여 높은 대역폭의 서비스를 지원하고 있다. 따라서 MPTCP 기능을 가지고 있는 장비를 활용하여 WiFi/LTE와 같은 이기종 네트워크 또는 여러 사업자의 LTE 망을 aggregation하면 사용자에게 더 높은 품질의 서비스를 제공할 수 있다. 하지만, 인터넷에 존재하는 모든 장비에 MPTCP 기능을 추가하는데 어려움이 있어 전송 경로 상의 MPTCP 프록시를 활용한 다중 경로 aggregation 기술들이 연구되고 있다. 본 논문에서는 네트워크 처리량을 높이기 위해 MPTCP라우터 및 MPTCP프록시를 활용하여 KT와 SKT의 LTE 사업자 망을 aggregation하였고 실제 환경에서 성능개선을 위한 테스트를 수행하였다. 또한 기존에 제안된 MPTCP 스케줄러 및 혼잡제어 알고리즘을 LTE 환경에 적용하고 단일 LTE 사업자 망과 다중 LTE 사업자 망에서의 성능을 비교 분석하였다.

Key Words : LTE link aggregation, MPTCP proxy, MPTCP scheduler, congestion control

ABSTRACT

As the contents used on the Internet are diversified and the quality has increased, the required network bandwidth has increased rapidly. The IETF proposed MPTCP (Multipath TCP), a transport layer standard protocol, to support high bandwidth services by aggregating multiple subflows on various interfaces. Therefore, higher quality service can be provided to users by aggregating heterogeneous networks such as WiFi/LTE or LTE networks of multiple operators using equipment with MPTCP function. However, since it is difficult to add the MPTCP function to all devices on the Internet, multipath aggregation techniques using an MPTCP proxy on the transmission path are being researched. In this paper, KT and SKT LTE networks were aggregated using an MPTCP router and MPTCP proxy to increase network throughput, and a test for

※ 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 Electronics Engineering, Kyungpook National University, ehkim@ee.knu.ac.kr, 학생회원
° Corresponding Author : School of Electronics Engineering, Kyungpook National University, yzcho@ee.knu.ac.kr, 종신회원
* Department of Military Electronic Communication, Yeungjin University, wkseo@yju.ac.kr
논문번호 : 202208-190-D-RE, Received August 23, 2022; Revised October 13, 2022; Accepted October 22, 2022

performance improvement was performed in a real environment. In addition, the previously proposed MPTCP scheduler and congestion control algorithm was applied to the LTE environment, and the performance of the single LTE operator network and the multi-LTE operator network were compared and analyzed.

Ⅰ. 서 론

인터넷 환경이 복잡해지고 다양한 서비스가 인터넷을 통해 제공되면서 요구되는 네트워크 대역폭이 빠르게 증가하고 있다. 특히 2009년 국내 출시된 스마트폰의 영향으로 무선 트래픽의 수요가 폭발적으로 증가했고 트래픽 폭증에 대응하기 위해 다양한 무선통신 기술이 개발되었다. 이동전화 인터넷 기술은 LTE에서 업로드 기준 최대 75 Mbps까지 속도가 빨라졌고, WiFi 기술은 IEEE802.11 ac 및 ad에서 각각 6.9 Gbps, 6.7 Gbps 수준으로 빨라졌다. 무선 기술의 자체 속도를 빠르게 개발하는 방법과 더불어 여러 무선 채널을 묶어서 속도를 개선시키는 방법도 연구되고 있다. LTE의 경우 Carrier Aggregation 기술을 사용하여 LTE-A, LTE-A Pro로 진화하였으며 업로드 및 다운로드 대역폭을 크게 증가시켰다. 그 외에 <그림 1>과 같이 LTE 및 WiFi 등의 이기종 망을 묶어서 대역폭을 늘리는 기술도 연구되고 있으며 최근 출시된 스마트폰의 경우 LTE와 WiFi의 인터페이스가 장착되어 있어 MPTCP 통신이 가능하다.

MPTCP (Multipath TCP)[1]는 트래픽의 고가용성 및 확장성을 위해 데이터 스트림을 분할하고 서브플로우를 생성하여 데이터를 송수신하는 기술로서 기존의 TCP[2] 통신을 수행하는 노드들 사이에 다중 인터페이스 및 다중 경로가 존재할 경우 둘 이상의 경로로 데이터를 전송하는 것이 가능하다. 하지만 MPTCP는 기존 단일 경로 TCP에서 이슈가 되었던 혼잡제어, 흐름제어, 오류제어 외에 서브플로우 간의 스케줄링 및 공평성 문제도 고려되어야 한다. 하나의 MPTCP 세션에 성능 차이가 큰 다수의 이종 경로가 존재할 경우 단일 경로를 사용하는 TCP 연결보다 처리량이 더 낮을 수 있다[3]. 처리량 저하를 발생시키는 요인으로 Out-of-order 도착 문제 및 Head-of-Line 블로킹 문제가 있으며 MPTCP에 적용된 스케줄러 및 혼잡제어 알고리즘에 따라 네트워크 성능이 변화한다[4-8].

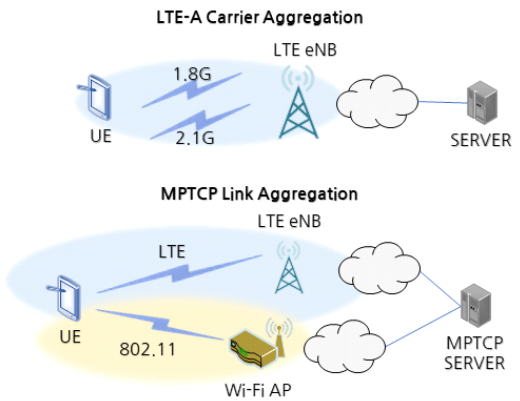

그림 1. Aggregation 기술 : LTE-A, MPTCP
Fig. 1. Aggregation technologies : LTE-A, MPTCP

MPTCP 통신은 MPTCP 기능이 탑재된 노드 사이에서 이용 가능하다. 스마트 폰의 경우 다양한 통신 인터페이스를 탑재하여 기본적으로 MPTCP 이용이 가능하지만 대부분의 PC 및 서버들은 일반 TCP를 사용하고 있어 추가적인 작업이 필요하다. 하지만 네트워크에 연결된 모든 장비에 MPTCP 기능을 추가하는 것은 현실적으로 불가능하기 때문에 이러한 문제를 해결하기 위해 MPTCP 프록시를 사용한다[9]. MPTCP 프록시는 MPTCP 연결과 일반 TCP 통신을 위해 데이터 매핑 기능을 수행한다. 따라서 <그림 2>와 같이 네트워크 경로 상에 MPTCP 프록시를 적절히 배치하면 사용자의 단말이나 서버의 MPTCP 지원 여부와 관계없이 MPTCP 통신을 이용할 수 있다.

본 논문에서는 데이터 전송 속도를 높이기 위해 홈 라우터 및 MPTCP 프록시를 구축하고 서로 다른 LTE 사업자 망을 aggregation하여 하나의 LTE 사업자 망을 사용하는 환경과 성능을 비교하였다. 또한 서로 다른 LTE 사업자 망을 사용하는 실제 환경에서 MPTCP 스케줄러 및 혼잡제어 알고리즘의 조합을 변경하며 성능 분석을 수행하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 기존에 제

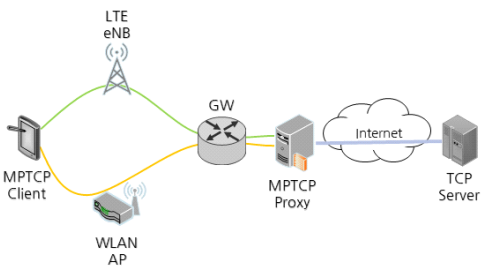

그림 2. MPTCP 프록시
Fig. 2. MPTCP proxy

안된 MPTCP 스케줄러 및 혼잡제어 알고리즘의 동작에 대해 설명한다. Ⅲ장에서는 다양한 환경에서 MPTCP 스케줄러 및 혼잡제어 알고리즘 성능 실험 결과를 소개한다.

Ⅱ. MPTCP 스케줄러 및 혼잡제어 알고리즘

본 장에서는 기존에 제안된 대표적인 MPTCP 스케줄러 및 혼잡제어 알고리즘의 동작 방법 및 특징에 대해서 살펴본다.

2.1 MPTCP 스케줄러

리눅스 커널 5.6 버전에서 MPTCP 기능이 추가되고 minRTT 스케줄러가 기본 스케줄러로 적용되었다[10]. 현재 MPTCP 스케줄러의 성능 개선을 위해 왕복 지연시간 (RTT), 단방향 지연 (OWD), 버퍼 크기 (buffer size), 혼잡 윈도우 (cwnd) 등의 매개변수를 고려하여 많은 연구가 진행되고 있다. 본 논문의 실험에 사용된 MPTCP 스케줄러의 종류와 특징은 다음과 같다.

1) minRTT: 리눅스 MPTCP의 기본 스케줄러로 사용되고 있는 minRTT는 생성된 서브플로우 중 가장 낮은 RTT를 갖는 경로로 cwnd가 가득 찰 때까지 패킷을 스케줄링 한다. 가장 낮은 RTT를 갖는 서브플로우의 cwnd가 가득차면 그 다음으로 높은 RTT를 갖는 서브플로우로 패킷을 스케줄링한다.

2) round robin: round robin 스케줄러[11]는 RTT나 기타 매개변수를 고려하지 않고 모든 서브플로우를 순차 순환 방식으로 스케줄링한다. 서브플로우 사이의 성능 차가 큰 환경에서 round robin 방식으로 패킷을 전송할 경우 HoL 블로킹 및 out-of-order 문제가 발생하여 심각한 성능저하를 유발한다. 따라서 리눅스 커널에 설치된 round robin 스케줄러에는 연속으로 전송할 세그먼트 수를 조절하는 num_segments와 완전한 round robin 방식으로 동작할지 여부를 결정하는 cwnd_limited의 설정 인자를 제공한다. cwnd_limited는 true가 기본 값으로 모든 서브플로우의 cwnd를 채우도록 설정되어 있으며 완전한 round robin 방식으로 동작하는 false값은 성능저하로 인해 교육 및 테스트 용도로 권장된다.

3) Redundant: Redundant[12,13]는 모든 서브플로우에 중복된 트래픽을 전송하는 스케줄러로서 대역폭을 희생하여 낮은 대기시간을 달성하고자하는 환경에 적합하다. 중복된 트래픽을 전송하기 때문에 생성된 서브플로우 중 가장 처리량이 높은 서브플로우보다 더 좋은 성능을 낼 수 없다. 하지만 네트워크 용량이 충분하고 대용량 파일 전송이 아닌 주기적 비콘이나 즉각적인 알림을 용도로 사용하는 망에 적용하기 적합한 스케줄러이다.

4) BLEST: BLEST (BLocking ESTimation-based)는 HoL 블로킹 문제를 예방하기 위한 proactive형태의 스케줄러이다[14]. 이용 가능한 빠른 서브플로우의 cwnd를 minRTT 스케줄러와 동일한 방법으로 채우고 느린 서브플로우의 사용 여부는 알고리즘 1과 같이 수식을 적용하여 결정한다. MPTCP 연결이 시작되면 최초 λ값이 1로 설정되며 패킷 재전송의 여부에 따라 가감된다. 만약 X × λ > Y일 경우 느린 서브플로우는 사용되지 않고 빠른 서브플로우의 동작이 끝날 때 까지 대기한다.

5) ECF: ECF (Earliest Completion First) 스케줄러[15]는 BLEST와 마찬가지로 빠른 서브플로우의 cwnd가 가득차서 사용할 수 없더라도 느린 서브플로우를 사용할지 여부를 결정한다. 그러나 ECF는 패킷의 도착시간에 중점을 두고 <그림 3>과 같이 빠른 서브플로우와 느린 서브플로우를 사용하였을 때, 전송이 완료되는 시간을 비교하여 스케줄링을 수행한다.

Algorithm 1. BLEST 스케줄러 매커니즘

```
if fastest subflow x_f is available then
    return x_f
else if slower subflow x_s is available then
    rtts = RTT_s/RTT_f
    X = MSS_f × (CWND_f + (rtts -1)/2) × rtts
    Y = |MPTCP send window| − MSS_s × (inflight_s + 1)
    if X × λ ≤ Y then
        return x_s
return no available flow
```

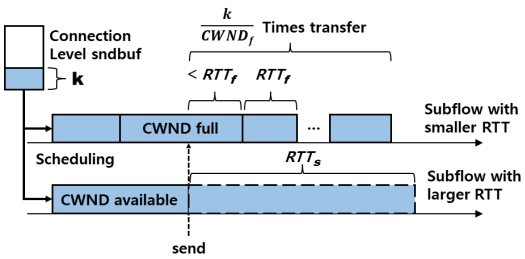

그림 3. ECF 스케줄링
Fig. 3. ECF scheduling

2.2 혼잡제어 알고리즘

2.2.1 Coupled 혼잡제어 알고리즘

Coupled 혼잡제어 알고리즘은 MPTCP의 처리율이 단일 경로의 TCP보다 높아야하고 다른 TCP경로에 대한 공평성(Fairness) 및 친화성(Friendly)을 보장해야 하며 마지막으로 혼잡한 링크의 트래픽을 분산시킬 수 있도록 경로 간에 혼잡정도가 균형을 이루어야 한다. Coupled 혼잡제어 알고리즘의 종류 및 특징은 다음과 같다.

1) LIA: LIA(Linked Increases Algorithm)[16]는 앞서 설명하였던 Coupled 혼잡제어 목표를 고려하여 제안되었다. LIA는 cwnd를 증가시키기 위한 증가인자인 α에 RTT(Round Trip Time)고려하여 식 (1)과 같이 표현하였고, r번째 서브플로우의 혼잡윈도우 크기인 wr은 식 (2)와 같이 계산하여 ACK를 받을 때마다 증가시킨다.

$$\alpha = \sum_r w_r \times \frac{\max_r \frac{w_r}{\mathrm{rtt}_r^2}}{(\sum_r \frac{w_r}{\mathrm{rtt}_r})^2} \tag{1}$$

$$w_r = w_r + \min(\frac{\alpha}{\sum_r w_r}, \frac{1}{w_r}) \tag{2}$$

패킷 손실 발생 시, 일반적인 TCP와 동일한 방법으로 cwnd 크기를 반으로 줄인다. 여러 경로의 cwnd를 균형 있게 조절하고 공평성을 유지하는 장점이 있지만 대역폭이 크고 높은 지연이 발생하는 환경에서 성능저하가 발생하여 대역폭 사용 효율이 떨어진다.

2) OLIA: OLIA(Opportunistic Linked Increases Algorithm)[17]는 LIA의 Flappy 문제를 개선하기 위해 제안되었다. 각 서브플로우가 가지는 cwnd의 크기에 따라 cwnd의 증가 속도가 상대적으로 달라진다.

3) Balia: Balia(Balanced linked adaptation)[18]는 OLIA와 마찬가지로 TCP Reno의 확장이며 친화성과 균형에 초점을 맞춘 혼잡제어 알고리즘이다. 혼잡회피 단계에서 AIMD (Additive Increase Multiplicative Decrease) 방식으로 동작하며 서브플로우의 수가 하나일 경우 TCP Reno와 동일하게 동작한다. ACK가 정상적으로 수신되면 식 (3)과 같이 cwnd가 증가되며, 손실이 발생하면 식 (4)와 같이 cwnd가 감소한다.

$$\frac{\frac{\mathrm{cwnd_i}}{\mathrm{rtt_i^2}}}{(\sum k \frac{\mathrm{cwnd_k}}{\mathrm{rtt_k}})^2} \times \frac{1+\alpha i}{2} \times \frac{4+\alpha i}{5} \tag{3}$$

$$\frac{\mathrm{cwnd_i}}{2} \times \min(\alpha_i, 1.5) \tag{4}$$

4) wVegas: wVegas(weighted Vegas)[19]는 지연 기반의 TCP Vegas를 확장한 알고리즘으로 큐에 발생하는 지연시간을 혼잡 신호로 인식한다. 각 서브플로우 r에 대해 정의된 임계값과 비교하여 cwnd크기를 증가 또는 감소시킨다. 식 (5)에서 α와 β는 TCP Vegas에서 정의된 임계값이며 kr은 기대 처리율과 전체 처리율의 비율이다.

$$\begin{cases} \text{if } \frac{W_r}{RTT_{\min}} - \frac{W_r}{RTT} < k_r{}^*\alpha,\ W_r + 1 \\ \text{if } \frac{W_r}{RTT_{\min}} - \frac{W_r}{RTT} > k_r{}^*\beta,\ W_r - 1 \end{cases} \tag{5}$$

2.2.2 Uncoupled 혼잡제어 알고리즘

1) Reno: AIMD 방식으로 동작하는 Reno[20]는 초기에 제안된 대표적인 손실기반 혼잡제어 알고리즘이다. 기존 알고리즘인 Tahoe와 동일하게 ACK를 정상적으로 수신하게 되면 혼잡 윈도우 크기를 2배씩 증가시키고 timeout 발생 시, 혼잡 윈도우 크기를 1로 설정하고 ssthresh (slow start threshold)의 값 까지 빠르게 복구한다. 하지만 fast retransmit (timeout 발생 전 세 개의 중복 ACK 수신)인 경우 알고리즘 2와 같이 혼잡윈도우 크기를 절반으로 줄이고 새로운 ACK가 도달할 때까지 대기하는 빠른 회복 단계를 수행한다. 새로

Algorithm 2. Reno 혼잡 윈도우 메커니즘

3−duplicated ACK

$ssthresh$ = cwnd

cwnd = $\frac{\mathrm{cwnd}}{2}$

Timeout

$ssthresh$ = $\frac{\mathrm{cwnd}}{2}$

cwnd = init cwnd

Fast Recovery

cwnd = cwnd × 2, if cwnd < $ssthresh$

cwnd = cwnd + 1MSS, if cwnd ≥ $ssthresh$

운 ACK를 수신하면 혼잡 윈도우 크기를 1씩 증가시키는 혼잡 회피 단계로 동작한다.

2) CUBIC: CUBIC[21]은 기존의 BIC (Binary Increase Congestion control)을 개선한 알고리즘으로 BIC의 다양한 구간으로 인해 복잡하게 구현된 함수를 단순하게 변경하고 다수의 플로우가 서로 다른 전파지연을 가진 환경에서 RTT 공평성 문제를 해결하기 위해 제안되었다. CUBIC은 아래 식 (6)과 같이 혼잡 윈도우 크기가 감소된 마지막 시점을 기준으로 경과된 시간인 t값을 수식에 적용하여 RTT 공평성을 개선하였다. Wmax는 혼잡 윈도우 크기가 감소하기 직전의 크기를 나타내며, 수식 (7)에서 β와 c값은 혼잡 윈도우의 증가 속도를 조절하는 역할을 하며 기본 값이 각각 0.7, 0.4로 설정되어있다.

$$cwnd = C(t-k)^3 + W_{max} \quad (6)$$

$$K = \sqrt[3]{\frac{W_{max}\beta}{c}} \quad (7)$$

3) BBR: BBR[22]은 네트워크 속도를 향상시키기 위해 구글에서 자체 개발한 모델기반 혼잡제어 알고리즘으로 기존의 패킷 손실을 이용하여 혼잡여부를 판단하는 Reno, CUBIC와 달리 라우터를 통해 데이터 처리량과 처리시간을 확인하여 손실될 패킷을 미리 예측한다. 이러한 예측을 통해 해당 네트워크에 적절한 데이터를 전송하기 때문에 기존의 손실기반 TCP 혼잡제어 알고리즘과 비교하여 더 효율적인 성능을 얻는다. 구글은 BBR 혼잡제어 알고리즘을 유튜브 및 클라우드 환경에 적용하고 기존의 패킷손실 기반 혼잡제어 알고리즘과 비교하여 처리량 및 큐잉 지연 (Queuing Delay)이 개선됨을 확인하였다.

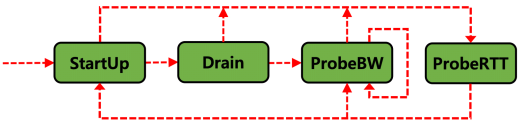

그림 4. BBR의 4가지 동작 모드
Fig. 4. 4-operation mode of BBR

BBR은 <그림 4>와 같이 StartUp, Drain, ProbeBW, ProbeRTT의 4가지 모드로 동작한다. StartUp 모드에서 ACK를 수신하면 대기열이 발생할 때까지 송신하는 데이터의 양을 약 2.89배씩 증가시킨다. 데이터 처리율이 최대치에 도달하면 StartUp 모드를 종료하고 버퍼에 데이터를 제거하기 위해 Drain 모드를 수행한다. ProbeBW 모드에서 pacing_cycle을 이용하여 가용 대역폭의 크기를 주기적으로 확인한다. ProbeRTT 모드에서 RTTmin가 10초 이상 갱신되지 않을 경우 혼잡 윈도우 크기를 4로 제한하여 새로운 RTTmin를 측정한다.

Ⅲ. 실험 및 평가

3.1 사업자가 다른 다중 LTE 환경에서 성능 평가

3.1.1 테스트베드 환경 구성

다중 이동통신 사업자의 LTE 망을 이용하는 환경에서 네트워크 성능 평가를 위해 <그림 5>와 같이 테스트베드 실험 환경을 구축하였다. Iperf3 클라이언트 및 서버, MPTCP 라우터, 스마트 폰, MPTCP 프록시 서버를 이용하여 테스트베드를 구축하였으며 각 장비의 정보는 표 1과 같다.

"Iperf"[23] 툴을 이용하여 TCP 트래픽을 발생시켰으며, MPTCP 스케줄러 및 혼잡제어 알고리즘의 조합

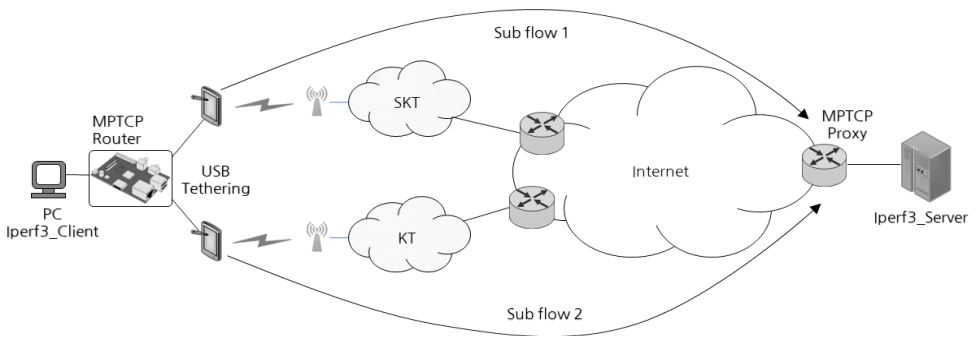

그림 5. 테스트베드 구성도
Fig. 5. Test bed configuration

표 1. 테스트베드 장비 사양
Table 1. Test bed equipment specifications

구분	내용
Iperf3_Client	· CPU: Intel Silver N5030 @ 1.10GHz · Memory: 4 GBytes · SSD: 57 GBytes · NIC: 1 Gbps · OS : Windows 10
MPTCP Router	· Raspberry Pi 4 Model B · Ethernet: 1Gbps NIC · WAN1: LTE USB Tethering(SKT) - iPhone X · WAN2: LTE USB Tethering(KT) - Galaxy S21 · OS: openMPTCProuter
MPTCP Proxy	· oVirt Virtual Machine · CPU: 8 Cores · Memory: 8 GBytes · HDD: 300 GBytes · OS: CentOS 8
Iperf3_Server	· CPU: Intel i7-2600 @ 3.40GHz · Memory: 12 GBytes · SSD: 500 GBytes · NIC: 1 Gbps · OS : Ubuntu16.04

에 따라 30초씩 10회 반복하여 성능 테스트를 진행하였다. 실험에 사용된 MPTCP 스케줄러는 minRTT, round robin, Redundant, BLEST, ECF이며, 혼잡제어 알고리즘은 LIA, Balia, OLIA, wVegas, Reno, CUBIC, BBR이다.

MPTCP 라우터는 Raspberry Pi 4 Model B 기기에 LTE 통신이 가능한 모바일 스마트폰 2대를 USB테더링으로 연결하였으며, 각 모바일 스마트폰의 이동 통신사는 SKT 및 KT를 사용하였다. MPTCP 라우터는 "openMPTCProuter"[24] v0.59가 설치되었으며 MPTCP 프록시 설치를 위해 오픈소스 기반의 가상화 솔루션인 "oVirt"[25]를 이용하여 실험환경을 구축하였다. Iperf3 클라이언트와 MPTCP 라우터 간 통신을 위한 물리적 요소인 NIC 및 케이블의 대역폭은 1 Gbps이며, 목적노드인 Iperf3 서버가 위치한 네트워크도 동일한 대역폭으로 구성하였다. 실험 시작 전 NIA (한국지능정보사회진흥원)의 LTE 인터넷 속도측정 앱을 이용하여 각 통신사의 인터넷 속도를 측정하였다. SKT를 사용하는 스마트폰에서 다운로드 18.89 Mbps, 업로드 15.94 Mbps, 지연시간 25.8 ms로 측정되었으며, KT를 사용하는 스마트폰에서 다운로드 37.45 Mbps, 업로드 38.17 Mbps, 지연시간 40.2 ms로 측정되었다.

3.1.2. MPTCP 스케줄러에 따른 혼잡제어 알고리즘 성능 비교

<그림 6>은 다중 이동통신 사업자의 LTE 망 환경에서 MPTCP 스케줄러에 따른 혼잡제어 알고리즘의 성능을 측정한 결과이다. 각각의 차트에서 나타나는

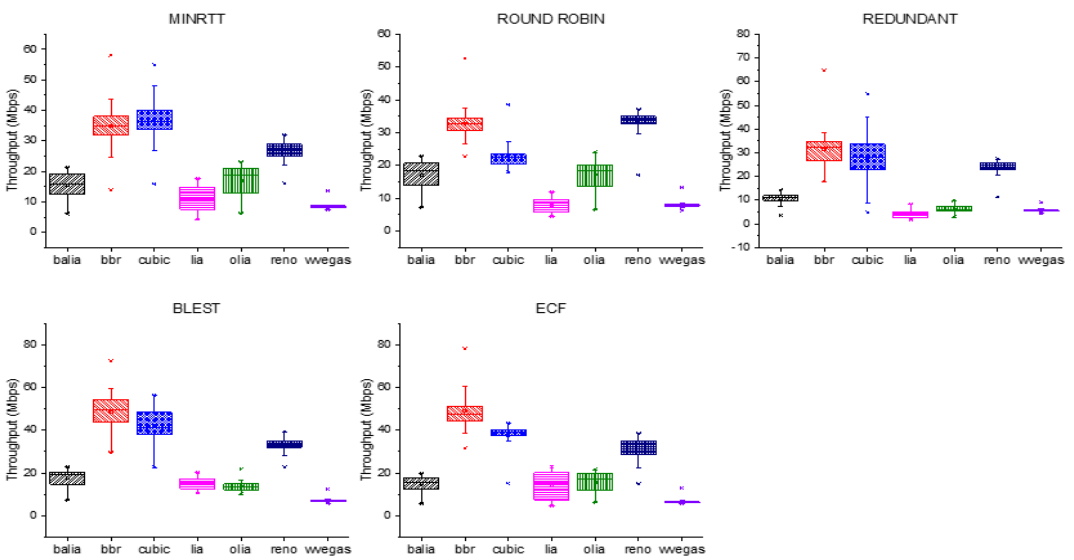

그림 6. 다중 LTE 사업자 환경에서 MPTCP 스케줄러에 따른 혼잡 제어 알고리즘 성능
Fig. 6. Congestion control algorithm performance according to MPTCP scheduler in multiple LTE ISPs

공통적인 특징으로 적용되는 MPTCP 스케줄러의 종류에 관계없이 Coupled 혼잡제어 알고리즘과 Uncoupled 혼잡제어 알고리즘 사이의 뚜렷한 성능차이가 발생하는 것을 확인할 수 있다. Uncoupled 혼잡제어 알고리즘의 평균 처리량은 BBR, CUBIC, Reno의 순으로 높게 측정되었고 Coupled 혼잡제어 알고리즘의 경우 LIA, OLIA, Balia의 성능은 비슷하나 wVegas의 성능이 상대적으로 낮은 것을 확인할 수 있었다. 특히 BLEST 스케줄러를 사용하는 BBR 알고리즘의 경우 평균 처리량이 48.6 Mbps로 가장 높게 측정되었고, 이 결과는 실험시작 전 측정하였던 각 통신사 LTE 업로드 평균속도의 합(54.11 Mbps) 대비 89.8%에 해당하는 성능이다. BBR은 round robin, Redundant, ECF 스케줄러를 사용할 때도 가장 높은 평균 처리량을 보였다. 반면 CUBIC 혼잡제어 알고리즘은 minRTT 스케줄러를 사용하는 경우를 제외하고 BBR보다 다소 낮은 성능을 가졌으며, Reno 혼잡제어 알고리즘은 round robin 스케줄러와의 조합을 제외하고 Uncoupled 혼잡제어 알고리즘 중에서 가장 낮은 처리량을 보였다. 이를 통해, 서로 다른 사업자 망으로 두 개의 서브플로우를 이용해 데이터를 전송하는 실험에서는 서브플로우가 독립적으로 동작하는 Uncoupled 혼잡제어 알고리즘이 BLEST 스케줄러를 이용할 때 평균적으로 가장 높은 처리량을 제공하는 것을 알 수 있다. 이는 서로 다른 특성을 갖는 두 개의 사업자 망을 통한 데이터 전송에서, 전송률 및 지연 시간의 차이로 인해 발생할 수 있는 MPTCP의 성능 저하를 BLEST 스케줄러를 이용해 예방할 수 있기 때문이다. 또한 MPTCP 라우터와 MPTCP 프록시 서버 사이에 서로 다른 LTE 사업자 망을 사용하여 공유되는 병목 링크 구간이 거의 없기 때문에 Uncoupled 혼잡제어 알고리즘의 성능이 높은 것으로 분석된다. 반면 Coupled 혼잡제어 알고리즘인 Balia, LIA, OLIA의 경우 성능이 좋은 하나의 LTE 사업자 망을 사용할 때보다 총 처리량이 낮게 측정되었다. 성능저하가 발생한 원인으로 Balia, LIA, OLIA의 경우 <그림 7>과 같이 최초 통신이 시작된 후 처리량을 증가시키는 구간이 단일 LTE 사업자 망(KT)을 사용하는 경우 평균 10초 정도로 측정되었으나 서로 다른 LTE 사업자망을 동시에 사용할 경우 처리량 증가 구간이 늘어나고 처리량 증가 속도가 감소한 요인으로 분석된다.

<그림 8>은 LTE 환경에서 MPTCP 사용여부에 따른 BBR, CUBIC, Reno 알고리즘의 처리량을 측정한 결과이다. BLEST, ECF, minRTT 스케줄러를 사용할 경우 BBR, CUBIC, Reno 알고리즘에서 처리량을 향상되는 것을 확인할 수 있었다. Balia, LIA, OLIA와 달리 MPTCP 사용여부에 따른 처리량 증가 구간에 큰 변화가 없었으며, 단일 사업자 망을 사용한 실험 및 서로 다른 사업자 망을 aggregation하여 사용한 실험 모두 처리량의 변화 구간이 유사하게 나타났다.

3.2. 단일 사업자 LTE 환경에서 성능 평가

3.2.1 테스트베드 환경 구성

하나의 LTE 사업자망을 사용하는 환경에서 각 사업자망(KT, SKT)의 혼잡제어 알고리즘 성능 평가를 위해 MPTCP 라우터에서 실험에 불필요한 WAN인터페이스는 제거하였으며 나머지 환경은 3.1.1.의 실험환경과 동일하게 설정하였다.

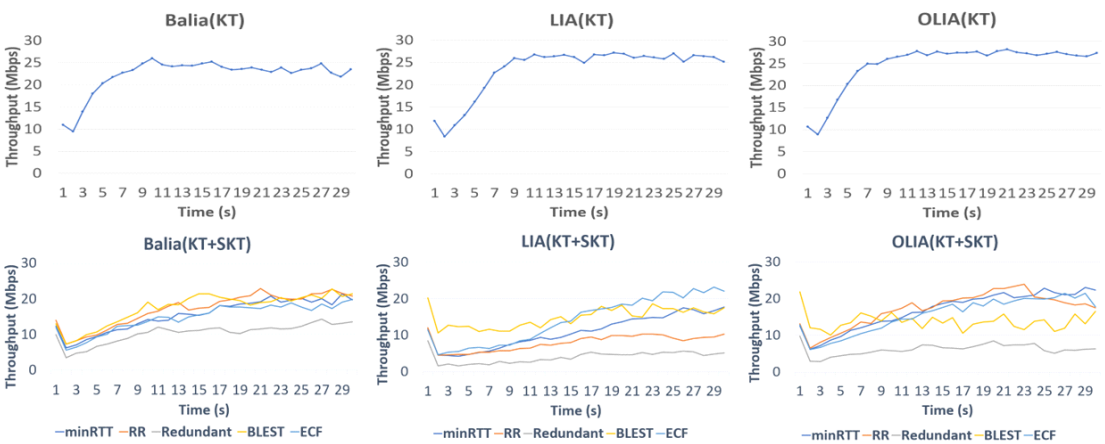

그림 7. LTE 환경에서 Balia, LIA, OLIA 성능 비교 (단일 사업자 & 다중 사업자)
Fig. 7. Comparison of Balia, LIA, OLIA performance in LTE environment (single ISP & multiple ISPs)

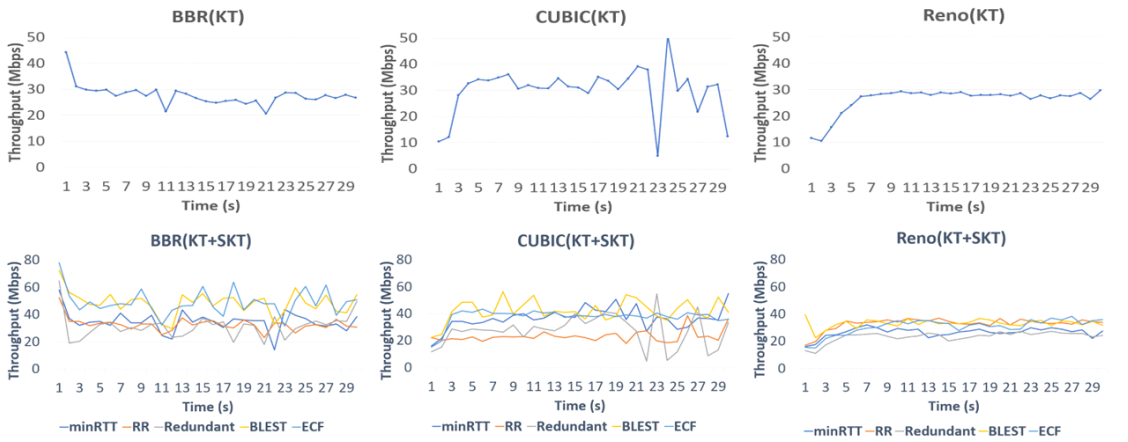

그림 8. LTE 환경에서 BBR, CUBIC, Reno 성능 비교 (단일 사업자 & 다중 사업자)
Fig. 8. Comparison of BBR, CUBIC, Reno performance in LTE environment (single ISP & multiple ISPs)

3.2.2 혼잡제어 알고리즘 성능 비교

<그림 9>는 각각의 LTE 망(KT, SKT)을 이용하여 혼잡제어 알고리즘의 성능을 측정한 결과이다. 각 통신사에서 CUBIC과 BBR 알고리즘의 처리량이 다른 알고리즘과 비교하여 상대적으로 높게 측정되었고 KT의 실험환경에서 평균 처리량이 각각 30.1 Mbps, 27.7 Mbps, SKT의 실험환경에서 평균 처리량이 각각 16.14 Mbps, 18.36 Mbps로 나타났다. 전체적인 알고리즘의 성능 또한 KT망을 이용할 때 다소 높게 측정되었다. LTE를 이용한 실험에서 실험 장소 및 기지국의 상태와 같은 다양한 변수들이 네트워크 성능에 영향을 줄 수 있다. 본 실험을 수행하였던 환경에서는 SKT 망에서 KT 망보다 상대적으로 많은 혼잡이 발생하였고, 그 결과 KT망에서 wVegas를 제외한 나머지 혼잡제어 알고리즘의 성능에 큰 차이가 발생하지 않았다. 반면 혼잡이 많이 발생한 SKT망에서는 혼잡제어 알고리즘 간의 다소 높은 성능 차이가 확인되었다. BBR의 경우 앞서 설명한 모델기반의 알고리즘 특성으로 인해 혼잡이 많이 발생하는 환경에서 다른 혼잡제어 알고리즘과 비교하여 상대적으로 높은 처리량을 나타내었으며, CUBIC의 경우 Reno와 비교하여 빠른 cwnd의 복구가 성능 차이의 원인으로 분석된다.

wVegas의 경우 MPTCP 사용여부와 관련 없이 LTE를 이용하는 환경에서 다른 혼잡제어 알고리즘과 비교하여 성능저하 현상이 크게 발생하는 것을 확인할 수 있었다. LTE를 이용하는 환경에서 wVegas의 성능저하 요인으로 지연기반 알고리즘의 특성을 생각해 볼 수 있다. 손실기반 알고리즘인 CUBIC은 cwnd의 크기를 지속적으로 증가시키지만 지연기반 알고리즘인 wVegas는 다른 경쟁 플로우에 의한 대기 지연을 네트워크 혼잡으로 인식하여 cwnd 크기가 감소하고 총 처리량의 성능저하로 이어진다.

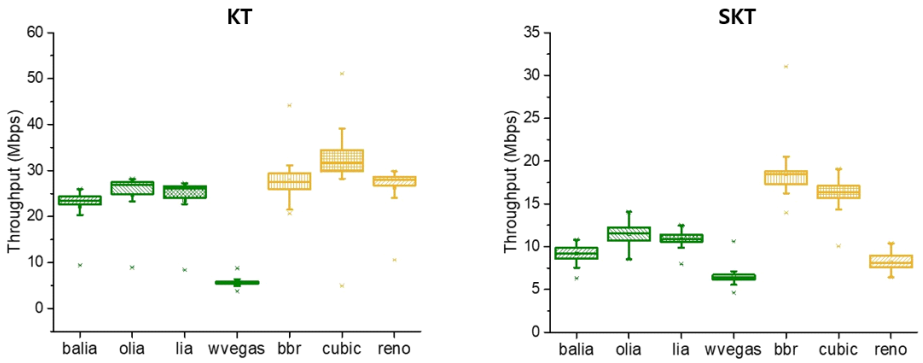

그림 9. LTE 환경에서 혼잡제어 알고리즘 성능
Fig. 9. Congestion control algorithm performance in LTE environment

Ⅳ. 결 론

본 논문에서는 데이터의 전송 속도를 높이기 위한 방안으로 서로 다른 LTE 사업자 망을 aggregation 하여 사용할 수 있도록 테스트 환경을 구축하였고 실제 환경에서 MPTCP 스케줄러 및 혼잡제어 알고리즘에 따른 성능을 분석하였다. 기존의 Coupled 혼잡제어 알고리즘의 경우 느린 혼잡윈도우 증가와 패킷 손실로 인한 성능 저하가 발생하였으며, 성능이 좋은 하나의 LTE 사업자 망을 이용하여 데이터를 전송할 때 보다 더 낮은 성능을 나타내기도 했다. 반면, Uncoupled 혼잡제어 알고리즘인 Reno, CUBIC, BBR의 경우 link aggregation이 이루어진 망에서 각 서브플로우의 대역폭을 적절히 활용하여 총 처리량이 증가하는 결과를 가져왔다.

MPTCP 스케줄러 및 혼잡제어 알고리즘이 네트워크 성능에 영향을 주는 추가적인 요인을 분석하기 위해 동일 LTE 사업자 망 내의 link aggregation 환경 또는 클라우드 데이터센터 환경과 같이 병목 구간이 많이 발생하는 다양한 환경에서 실험을 진행할 예정이다. 또한 차량이나 고속열차와 같은 이동성이 존재하는 환경에서도 link aggregation을 통해 성능 개선이 이루어질 수 있는지 확인해 볼 것이다.

References

[1] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and S. Barre, "*TCP extensions for multipath operation with multiple addresses*," RFC 6824, IETF, version 18, 2019. (https://doi.org/10.17487/rfc6824)

[2] J. Postel, "*Transmission control protocol specification*," RFC 793, IETF, 1981.

[3] S. C. Nguyen, X. Zhang, T. M. T. Nguyen, and G. Pujolle, "Evaluation of throughput optimization and load sharing of multipath TCP in heterogeneous networks," *Int. Conf. Wireless and Optical Commun. Netw.*, pp. 1-5, 2011. (https://doi.org/10.1109/wocn.2011.5872966)

[4] C. Paasch, R. Khalili, and O. Bonaventure, "On the benefits of applying experimental design to improve multipath TCP," *ACM CoNEXT*, Dec. 2013. (https://doi.org/10.1145/2535372.2535403)

[5] Y. J. Song, G. H. Kim, and Y. Z. Cho, "Performance evaluation between TCP congestion control algorithms," *J. KICS*, vol. 44, no. 11, pp. 2102-2112, 2019. (https://doi.org/10.7840/kics.2019.44.11.2102)

[6] G. H. Kim, C. H. Park, J. K. Kim, W. J. Eom, Y. J. Song, W. K. Seo, and Y. Z. Cho, "MPTCP scheduler research trends and problem analysis," *J. KICS*. vol. 45, no. 09, pp. 1570-1586, 2020. (https://doi.org/10.7840/kics.2020.45.9.1570)

[7] I. Khan and K. Chen, "EBA: Efficient bandwidth aggregation for connected vehicles with MPTCP," in *IEEE Internet of Things J.*, vol. 9, no. 8, pp. 5812-5823, Apr. 2022. (https://doi.org/10.1109/jiot.2021.3065911)

[8] S. D. Sathyanarayana, J. Lee, J. Lee, D. Grunwald, and S. Ha, "Exploiting client inference in multipath TCP over multiple cellular networks," in *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 58-64, Apr. 2021. (https://doi.org/10.1109/mcom.001.2000911)

[9] T. You, C. Park, H. Jung, T. Kwon, and Y. Choi, "Multipath transmission architecture for heterogeneous wireless networks," in *Proc. Int. Conf. ICTC 2011*, pp. 26-31, 2011. (https://doi.org/10.1109/ictc.2011.6082544)

[10] *MPTCP Linux Implementation*, [Online]. Available: http://www.multipath-tcp.org/.

[11] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proc. ACM CSWS*, pp. 27-32, 2014. (https://doi.org/10.1145/2630088.2631977)

[12] A. Frömmgen, A. Rizk, T. Erbshäußer, M. Weller, B. Koldehofe, A. Buchmann, and R. Steinmetz, "A programming model for application-defined multipath TCP scheduling," in *Middleware '17,* pp. 134-146, 2017. (https://doi.org/10.1145/3135974.3135979)

[13] B. Walker, V. A. Vu, and M. Fidler, "Multi-headed MPTCP schedulers to control latency in long-fat/short-skinny heterogeneous networks," in *Proc. CHANTS '18, ACM,* pp. 47-54, 2018.

(https://doi.org/10.1145/3264844.3264847)

[14] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," *IFIP Netw. and Wkshps.*, pp. 431-439, 2016. (https://doi.org/10.1109/ifipnetworking.2016.7497206)

[15] Y. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proc. CoNEXT ACM*, pp. 147-159, 2017. (https://doi.org/10.1145/3078505.3078552)

[16] C. Raiciu, M. Handley, and D. Wischik, "*Coupled Congestion Control for Multipath Transport Protocols*," RFC 6356, Oct. 2011. (https://doi.org/10.17487/rfc6356)

[17] R. Khalili, N. Gast, M. Popovic, and J. Le Boudec, "MPTCP is not pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1651-1665, 2013. (https://doi.org/10.1109/tnet.2013.2274462)

[18] Q. Peng, A. Valid, J. Hwang, and S. Low, "Multipath TCP: Analysis, design and implementation," *IEEE/ACM, Trans. Netw.*, vol. 24, no. 1, pp. 596-609, 2016. (https://doi.org/10.1109/tnet.2014.2379698)

[19] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," *2012 20th IEEE ICNP*, pp. 1-10, 2012. (https://doi.org/10.1109/icnp.2012.6459978)

[20] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno, and SACK TCP," *Comput. Commun. Rev.*, pp. 5-21, vol. 26, no. 3, Jul. 1996. (https://doi.org/10.1145/235160.235162)

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

[22] N. Cardwell, Y. Cheong, C. S. Gunn, S. H. Yeganch, and V. Jacobson, "BBR: Congestion-Based congestion control," *ACM Queue*, vol. 14, no. 5, 2016. (https://doi.org/10.1145/3012426.3022184)

[23] iperf, from https://iperf.fr.

[24] openMPTCProuter, from https://www.openmptcprouter.com/

[25] oVirt, from https://www.ovirt.org/

김 응 협 (Eung-Hyup Kim)

2012년 : 영진전문대학교 컴퓨터정보공학과 학사

2014년 : 경북대학교 전자공학부 석사

2016년~현재 : 경북대학교 전자공학부 박사과정

<관심분야> MPTCP 혼잡제어, SDN/NFV, 무선 애드혹 네트워크, 클라우드 컴퓨팅

[ORCID:0000-0003-0448-1094]

서 원 경 (Won-Kyeong Seo)

2005년 : 경북대학교 전자전기컴퓨터학부 학사

2007년 : 경북대학교 전자공학부 석사

2012년 : 경북대학교 전자공학부 박사

2012년~2014년 : 지웰주식회사 대표이사

2015년~현재 : 영진전문대학교 국방전자통신전공 교수

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

[ORCID:0000-0001-5920-4858]

조 유 제 (You-Ze Cho)

1982년 : 서울대학교 전자공학과 학사

1983년 : KAIST 전기전자공학 석사

1983년 : KAIST 전기전자공학 박사

1989년~현재 : 경북대학교 전자공학부 교수

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

2002년~2003년 : NIST(미국국립표준연구소) 객원연구원

2017년 : 한국통신학회 회장

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

[ORCID:0000-0001-9427-4229]