Journal of KIIT. Vol. 16, No. 8, pp. 45-55, Aug. 31, 2018. pISSN 1598-8619, eISSN 2093-7571 45
http://dx.doi.org/10.14801/jkiit.2018.16.8.45

MCubic(Multipath Cubic)의 설계, 구현 및 다중 경로 TCP의 성능 평가에 관한 연구

노승환*, 류장열**, 최진규***

Design and Implementation of MCubic(Multipath Cubic) and Performance Evaluation of Multipath TCP

Soonghwan Ro*, Jangryeol Ryu**, and Jinkyu Choe***

요 약

본 연구에서는 새로운 다중 경로 TCP 프로토콜인 MCubic(Multipath Cubic)에 대한 설계하고 구현하였다. MCubic은 다중 경로 TCP를 지원하기 위한 TCP Cubic의 확장된 개념이다. 본 연구에서 구현된 프로토콜의 실험결과 MCubic은 고속의 장거리 네트워크에서 유리한 TCP Cubic의 장점을 유지하면서, 트래픽 이동(traffic shift), 처리율(throughput) 개선 및 공정성(fairness) 등 다중 경로 전송의 요구사항을 만족하는 것으로 나타났다. 본 연구에서는 짧은 거리와 높은 지연을 갖는 경로의 다중 TCP접속에서 처리율을 높이기 위한 MCubic 알고리즘의 접근 방법과, 이 경로를 통해서 이루어지는 트래픽 이동에 대해서 제안하였다. 본 연구에서는 제안된 프로토콜을 실험실 환경에서 구현함으로써 기존의 시뮬레이션에 의한 실험결과보다 신뢰성을 높였다.

Abstract

We present Mcubic, a new Multipath TCP congestion control algorithm, from design to implementation. Mcubic is an extended version of TCP Cubic. Our experiment results indicate that Mcubic still maintains significant advances over TCP Cubic for high-speed long-latency networks and satisfies the new requirements for multipath transmission, which include traffic shifting, throughput improvement, and fairness. We highlight the approach taken by the Mcubic algorithm to improve the Multipath TCP connection throughputs in both short and long latency paths and the traffic shifting efficiency through the paths. In this study, the proposed protocol is implemented as a test bed, and the reliability of the experiment result is higher than that of simulation.

Keywords
MPTCP(Multi Path TCP), congestion control, MCubic, TCP cubic, fairness, window growth function

* 공주대학교 정보통신공학부(교신저자)
- ORCID: https://orcid.org/0000-0001-6091-796X
** 공주대학교 전기전자제어공학과 교수
- ORCID: https://orcid.org/0000-0003-4704-1320
*** 한남대학교 전자공학과
- ORCID: https:/orcid.org/0000-0002-2435-9806

· Received: Jun. 20, 2018, Revised: Aug. 06, 2018, Accepted: Aug. 09, 2018
· Corresponding Author: Soonghwan Ro
Dept. of Information and Communication Engineering, Kongju Univ. Korea
Tel.: +82-41-521-9197, Email: rosh@kongju.ac.kr

Ⅰ. 서 론

MPTCP(Multipath TCP)는 인터넷에서 다중호밍(Multi-homing)을 지원하는 새로운 전달계층 프로토콜이다. 이 프로토콜은 여러 개의 인터페이스를 갖는 단말 장치에 매우 유용하게 사용될 수 있으며, 현재 리눅스 커널에서 구현이 되어 있는 상태이다[1]. 그러나 혼잡제어와 관련하여 새로운 프로토콜이 설계되고 동작하기 위해서는 다양한 연구가 진행이 되어야 한다.

새로이 제안되는 혼잡제어가 동작되기 위해서는 다음과 같은 요구사항을 만족시켜야 한다[2].

- 처리율(Throughout)의 향상 : MPTCP는 가장 좋은 경로에서 최소한 단일 경로 TCP이상의 성능을 보여야 한다.
- 무해(No Harm) 해야 함 : MPTCP는 기존의 TCP에게 해를 주지 않는 방식으로 동작을 해야 한다. 즉, 병목 링크에서는 단일 경로 TCP와 같이 동작을 해야 한다.
- 혼잡의 균형 : MPTCP는 가장 혼잡이 적은 경로를 사용해야 한다.

위 목표 중에서 첫 번째 목표는 단일 경로 TCP와 비교했을 때 활용 가능한 경로를 효율적으로 사용해야 한다는 것을 나타내며, 두 번째와 세 번째는 공정성(Fairness), 민감성(Responsiveness) 그리고 MPTCP의 트래픽 이동(Traffic Shifting)을 보장하는 것이다. MCCA(Multipath Congestion Control Algorithm)은 위와 같은 목표를 잘 만족시키며, 설계 목적을 달성하기 위해서 서브 플로우(Subflow)의 전송 율을 가장 잘 유지하고 조율해야 한다. 본 연구에서 다루는 LIA(Linked Increases Algorithm)는 현재 MPTCP의 혼잡 제어의 목표를 가장 잘 달성하고 있다. 그러나 LIA는 지연이 높은 네트워크에서는 성능이 저하되는 것을 이전의 연구 [3]에서 보여주고 있다. 이것은 LIA가 지연이 높은 네트워크에서 첫 번째 목표를 완벽하게 달성하지 못했다는 것을 의미한다. 결론적으로 본 연구의 목적은 지연이 높을 수 있는 광역망에서 첫 번째 요구사항을 효율적으로 만족시킬 뿐만 아니라 나머지 조건을 만족시키는 알고리즘을 찾는 것이다.

2011년 [4] 연구에 의하면 약 30,000개의 웹 서버의 25%가 TCP Cubic을 사용하고, 20%가 TCPBic, 그리고 15%~20%가 TCP Compound를 사용한다고 알려졌다. TCP Cubic은 단일 경로에서 가장 많이 사용되는 알고리즘이며, 리눅스에서 기본적으로 사용되는 TCP 프로토콜이다. 많은 연구에 의해서 [5][6]에서 TCP Cubic은 장거리 고속 네트워크에서 처리율, 공정성 그리고 민감성 면에서 가장 효율적인 것으로 밝혀졌다. TCP Cubic의 장점은 이 프로토콜을 다중 경로 전송으로 확장하는데 매우 유리하게 적용될 수 있다. 본 연구에서는 MPTCP 프로토콜로써 TCP Cubic의 확장개념인 MCubic(Multipath Cubic)을 제안하였다. 제안된 알고리즘은 분석적인 방법과 실험적인 방법으로 분석되었다.

Ⅱ. 관련연구

MPTCP의 혼잡제어를 위해서 TCP New RENO를 확장한 LIA[7]와 같은 알고리즘 들이 제안되었다. LIA는 혼잡 윈도우 크기 증가 인자를 식 (1)과 같이 함으로써 RTT(Round Trip Time)의 불일치를 보상한다.

$$\min\left(\frac{\alpha}{\sum_{r=0}^{n} w_r}, \frac{1}{w_r}\right) \tag{1}$$

식 (1)에서 n은 서브플로우의 수를 나타내고 w_r은 r^{th} 서브플로우의 혼잡윈도우를 나타낸다. 그리고 α는 알고리즘의 파라미터를 각각 나타내며, 이며 MPTCP의 공격성를 나타낸다. 이 파라미터의 계산은 식 (2)와 같이 주어진다.

$$\alpha = \Sigma_r W_r \cdot \frac{\max_r \frac{W_r}{rtt_r^2}}{(\Sigma_r \frac{W_r}{rtt_r})^2} \tag{2}$$

위 식에서 W_r는 서브플로우 r^{th}의 혼잡 윈도우 크기이며, rtt_r는 서브플로우 r^{th}의 RTT이다. LIA의 설계는 통합 자원을 최적화하는 것과 민감도 사이에서 균형를 이루는 것이다.

최근 LIA의 변형으로 OLIA(Opportunistic Linked Increases Algorithm)[8] 알고리즘이 제안되었으며, LIA를 사용하는 MPTCP의 "non-flappy" 현상을 방지할 수 있을 것으로 기대된다. OLIA에서는 서브플로우 r의 각 ACK마다 식 (3)에 따라서 W_r를 증가시킨다.

$$\frac{\alpha}{\sum_{r=0}^{n} w_r} + \frac{\alpha_r}{w_r} \tag{3}$$

$$Where, \alpha_r = \begin{cases} \frac{1}{N_R \bullet N_{BnM}}, & \text{if } i \in B \cap i \notin M \\ -\frac{1}{N_R \bullet N_{MnB}}, & \text{if } i \in M \cap i \notin B \\ 0, & otherwise \end{cases} \tag{4}$$

위 식에서 W_r는 혼잡 윈도우를 나타내고, B는 시간 t에서의 최상의 경로를 갖는 서브플로우의 집합이고, M은 시간 t에서 가장 큰 혼잡 윈도우 크기를 갖는 서브플로우의 집합을 나타낸다. 또한 N_R은 MPTCP 접속의 전체 서브플로우 수를 나타내고, N_{BnM}는 집합 B에는 포함되나 집합 M에는 포함되지 않는 서브플로우의 수를 나타내며, N_{MnB}는 집합 M에는 포함되나 집합 B에는 포함되지 않는 서브플로우의 수를 나타낸다. 따라서 OLIA의 윈도우는 링크 상태가 가장 좋지만 윈도우 크기가 가장 작은 서브플로우의 윈도우 크기가 가장 빠르게 증가하며, 최대 윈도우를 갖는 서브플로우의 윈도우 크기는 늦게 증가한다.

wVegas(weighted Vegas)[9]는 TCP Vegas의 확장된 개념이다. wVegas는 지연에 기반(Delay-based)하는 MPTCP 알고리즘이며, 혼잡에 대한 신호로 패킷의 큐잉 지연을 사용한다. 이 알고리즘에서는 가중 인자(Weight Factor)를 각 서브플로우에 할당하고 혼잡에 따라서 적응적으로 조절한다. 각 RTT마다 서브플로우 r에서 k_r에 의해서 스케일 된 임계지점에서의 윈도우 크기의 증가와 감소는 식 (5)와 같다.

$$\begin{aligned} &\text{if } W_r/RTT_{\min} - W_r/RTT < k_r \bullet \alpha \ then \\ &\quad \text{increase } W_r \ by \ 1 \\ &\text{if } W_r/RTT_{\min} - W_r/RTT > k_r \bullet \beta \ then \\ &\quad \text{decrease } W_r \ by \ 1 \end{aligned} \tag{5}$$

α와 β는 두 개의 임계값이며 TCP Vegas에 의해서 정의된다. $RTT_{\min}$는 최소의 RTT이며, 서브플로우 r의 현재 RTT를 나타내고, k_r는 전체 처리율에 대한 기대 처리율의 비율이다.

이 밖에도 Cubic를 확장한 MPCubic[10]에 대한 연구가 진행이 되었으나 시뮬레이션에 의해 제안된 알고리즘에 대해서 성능평가를 실시하여 프로토콜의 검증이 충분하게 이루어졌다고 볼 수 없다.

III. MCubic

3.1 TCP Cubic 윈도우 증가 함수

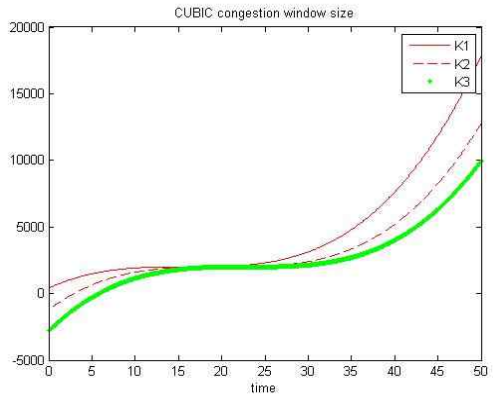
MCubic 알고리즘에 대한 설명을 하기 전에 먼저 TCP Cubic에 대해 알아본다. TCP Cubic의 주된 특성은 윈도우 증가 함수에 있다. 프로토콜의 이름에서 알 수 있듯이 윈도우 증가함수는 용적(Cubic)에 대한 함수이며 다음 함수에 의해 결정된다.

$$W_{cubic} = C(t-K)^3 + W_{\max} \tag{6}$$

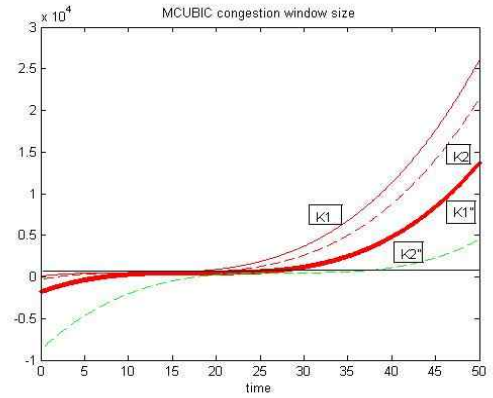
식 (6)에서 C는 스케일 인자(Factor)이며, t는 마지막으로 윈도우를 감소시킨 후로부터의 경과한 시간이고, $W_{\max}$는 윈도우가 감소하기 이전의 윈도우 크기를 나타낸다. 그리고 $K = \sqrt[3]{W_{\max}\beta/C}$ 이고, β는 패킷 손실이 발생했을 때 윈도우를 감소시키기 위해 적용되는 승수 감소 인자(Multiplication Decrease Factor)를 나타낸다. 즉, 윈도우 크기가 $\beta W_{\max}$로 감소한다.

그림 1은 3개의 서로 다른 K변수에 대해서 식 (6)에서의 윈도우 증가의 형태를 보여주며, $K_1 < K_2 < K_3$의 크기를 갖는다. 일반적으로 윈도우 크기는 감소된 윈도우로부터 매우 빠르게 증가하지만 $W_{\max}$에 가까워질수록 증가율이 감소하며, $W_{\max}$ 값에 이르면 증가는 거의 멈춘다. 그리고 그 다음에 더 이상의 대역폭이 있는지를 확인하면서 다시 윈

도우가 천천히 증가하기 시작하여 $W_{\max}$에서 멀어지면 증가 속도를 높인다. 이와 같이 $W_{\max}$ 근처에서 증가 속도를 줄이는 것은 프로토콜의 안정성을 향상시키고, 네트워크 자원의 활용성을 높이며, $W_{\max}$에서 멀어질수록 윈도우 증가 속도가 높아지는 것은 프로토콜의 확장성을 보장한다. [11]의 연구와 [12]의 TCP Cubic 소스 코드에서 K변수는 혼잡 윈도의 크기가 $W_{\max}$ 값에 도달할 때까지의 시간으로 나타나 있다. 실제로 그림 1(a)에 나타난 것과 같이 K 값이 증가함에 따라 $(K_3 > K_2 > K_1)$ $W_{\max}$에 도달하는 시간이 증가하는 것을 알 수 있으며, 선의 경사가 감소하는 것을 알 수 있다. 이것은 K 값이 TCP Cubic의 윈도우 증가 정도를 결정한다는 것을 나타낸다.



(a) K 변수에 따른 Cubic의 변화
(a) Cubic with varying K variable



(b) Cubic과 MCubic
(b) Cubic and MCubic
그림 1. 두 경우의 윈도우 증가함수
Fig. 1. Window increase function in both cases

3.2 MCubic의 설계 목표 및 전략

기존의 혼잡제어 알고리즘을 분석한 결과 기존의 알고리즘의 핵심은 다음과 같은 두 가지의 일반적인 설계 제약에 의해 제안되었다.

• 제약 1 : 단일 TCP와 링크를 공유할 때 각 서브플로우의 공격성(Aggressiveness)를 감소시킨다. 각 서브플로우는 단일 TCP와 링크를 공유할 때 공정성(Fairness)를 만족시키기 위해 윈도우 증가율이 느려서 각 서브플로우가 점유하는 링크가 단일 TCP 플로우가 동일한 링크에서 점유하는 것보다 작게 한다. 그러나 모드 서브플로우의 전송률은 최소한 단일 TCP가 최상의 경로에서 보여주는 최대 전송률만큼은 보장이 되어야 한다. 이 제약은 MPTCP의 첫 번째와 두 번째 목표를 얻기 위한 가장 중요한 요소이며, 예로 LIA와 OLIA에서는 모든 서브플로우에 동일한 α 파라미터를 사용해서 전체플로우의 공격성을 제어한다. α값은 항상 1보다 작아서 다중 경로플로우가 단일 경로 TCP가 점유할 수 있는 용량보다 크지 않도록 보장한다.

• 제약 2 : 모든 서브플로우의 전송률이 협력해서 민감성(Responsiveness) 뿐만 아니라 트래픽 이동이 발생할 수 있도록 한다. 네트워크의 상태는 항상 일정한 것이 아니기 때문에 네트워크의 상태가 바뀔 때에 신속하게 대역폭을 획득하거나 해제하는 것도 매우 중요하다. 민감성 특성은 얼마나 빨리 MPTCP 알고리즘이 네트워크 상태에 따라서 반응을 하는지를 나타내는 것이다. 따라서 알고리즘의 핵심 중에 하나는 각 서브플로우에서 전송률을 처리하는 절차를 결합시키고 서로 협력하도록 함으로써 트래픽 이동 뿐만 아니라 부하 균형이 되도록 하는 것이다. 따라서 혼잡에 의해 발생하는 대역폭 손실은 보상될 수 있으며 다른 서브플로우의 전송률을 높임으로써 회복될 수 있다. 결과적으로 한 서브플로우에서 발생하는 혼잡은 다른 경로의 전송률을 높일 수 있게 된다. 이러한 알고리즘은 모든 서브플로우의 품질 지수의 함수를 사용함으로써 가능하게 된다. 만일 이 품질 지수 중에 하나가 자신의 상태를 변화하면 이 변화에 반응해서 다른 서브플로우에 즉시 변화를 일으키게 한다. 예로 LIA와 OLIA에서 사용되는 α인자는 모든 서브플로우의

예상되는 전송률에 대한 함수이며, 공격성뿐만 아니라 가장 혼잡이 적은 경로로 트래픽을 이동 시키는 제어를 하는데 사용된다. 반면에 wVegas에서 사용되는 가중치는 예상되는 전송률을 모든 서브플로우의 전체전송률과 비교해서 대역폭 경쟁에 대한 공격성을 정량화한다. 결과적으로 혼잡이 적은 서브플로우 경로는 더 큰 가중치를 얻어서 공격적으로 대역폭을 얻기 위해 경쟁을 할 수 있다. 따라서 wVegas는 네트워크의 혼잡 상황이 변화하는 것에 좀 더 민감하고 트래픽 이동을 적절하게 이룰 수 있고 더 빠르게 수렴하게 된다.

앞에서 언급한 것과 같이 K 변수는 TCP Cubic을 사용하는 각 플로우의 전송률을 결정한다. 따라서 K 값이 크면 증가율이 낮게 되고, 따라서 MCubic에서는 다중 경로 전송을 위해서 K_{mptcp}라고 부르는 수정된 K 값을 사용한다. 본 연구는 앞의 두 가지 설계 제한으로부터 시작한다. C1에서 각 서브플로우는 단일 경로 TCP 플로우보다 혼잡 윈도우를 더 빠르게 증가시키지 않는다. 따라서 K_{mptcp}는 원래의 K보다 커야 하며, C2에서 변수 K_{mptcp}는 서브플로우의 품질 지표의 함수이어야 한다는 것이다. 따라서 MCubic의 K_{mptcp} 값을 결정하고 분석해야 한다. TCP Cubic 윈도우 증가함수에서 K 변수는 윈도우 최대 크기에 도달해서 링크의 제한된 용량까지 도달할 때까지의 시간을 나타낸다. 따라서 비율 $\frac{1}{K}$는 플로우의 증가율을 나타낸다. 만일 MPTCP에 속하는 n개의 플로우가 있다고 가정하면 각 플로우의 윈도우 증가함수는 변수 K_i를 갖는 혼잡 제어 알고리즘을 사용하며, 증가율은 $\frac{1}{K_i}$이 된다. 따라서 MPTCP 접속의 증가율은 $\sum_{i=1}^{n}\frac{1}{K_i}$이 된다. 단일 경로와 비교했을 때 MPTCP 접속은 $\gamma=\frac{\sum_{i=1}^{n}\frac{1}{K_i}}{\frac{1}{K_i}}$ 시간만큼 빠르다. 이 것은 MPTCP 접속이 TCP Cubic을 사용하는 단일 경로 TCP에 대해서 공정하지 못 하다는 것을 의미한다. 따라서 공정성을 확보하기 위해서 MCubic는 MPTCP 접속이 혼잡 제어 방식으로 TCP Cubic을 사용할 때, 각 서브플로우의 증가율을 증가분 γ의 역에 해당하는 만큼 감소시킨다. 그렇게 하면 이론적으로 전체 MPTCP의 증가율은 감소할 것이다. 감소시킨 후에 i^{th}번째 서브플로우의 감소율은 식 (7)과 같다.

$$\frac{1}{K_{mptcp}^{i}}=\frac{1}{K}*\frac{\frac{1}{K_i}}{\sum_{i=1}^{n}\frac{1}{K_i}}=\frac{\frac{1}{K_i^2}}{\sum_{i}^{n}\frac{1}{K_i}} \tag{7}$$

예상되는 MPTCP 접속의 전송률은 식 (8)과 같다.

$$\sum_{i=1}^{n}\frac{1}{K_{mptcp}^{i}}=\sum_{i=1}^{n}\frac{\frac{1}{K_i^2}}{\sum_{i=1}^{n}\frac{1}{K_i}}\geq\frac{1}{n}*\frac{(\sum_{i=1}^{n}\frac{1}{K_i})^2}{\sum_{i=1}^{n}\frac{1}{K_i}}=\frac{1}{n}*\sum_{i=1}^{n}\frac{1}{K_i} \tag{8}$$

서브플로우의 품질이 일정한 경우, 즉 $\frac{1}{K_1}\approx\frac{1}{K_2}\approx\ldots\approx\frac{1}{K_n}$일 때 $\frac{1}{n}*\sum_{i=1}^{n}\frac{1}{K_i}\approx\frac{1}{K_i}$이 되며, 식 (2)를 다시 정리하면 식 (9)와 같이 된다.

$$\sum_{i=1}^{n}\frac{1}{K_{mptcp}^{i}}\geq\frac{1}{K_i} \tag{9}$$

이것은 MCubic을 사용함으로써 예상되는 MPTCP 접속의 전송률은 항상 단일 경로 TCP의 전송률보다 높다는 것을 의미하며, 또한 TCP Cubic과도 함께 잘 동작할 수 있다는 것을 의미한다. 따라서 MCubic는 첫 번째와 두 번째 목표를 충분히 만족시킨다는 것을 알 수 있다. 이 결과는 그림 1(b)에 나타나 있으며, 그림에서 K_1,K_2 변수를 갖는 단일 경로 TCP Cubic과 K_1',K_2' 변수를 갖는 두 개의 서브플로우로 이루어진 MCubic의 윈도우 증가함수를 보여주고 있다. MCubic을 사용함으로써 K_1',K_2' 변수를 갖는 서브플로우의 각 전송률은 K_1',K_2' 변수를 갖는 단일 경로 TCP의 전송율보다 항상 낮다는 것을 알 수 있다. 반면에 식 γ에 의해서 하나의 서

브플로우는 $\frac{1}{K_i}$ 비율만큼 감소하게 되며, 전체적으로 $\sum_{i=1}^{n}\frac{1}{K_i}$ 만큼 감소하게 된다. 따라서 다른 서브플로우의 전송율을 즉시 증가할 수 있게 하며 혼잡이 발생했을 때 트래픽 이동이 가능하다는 점에서 매우 중요하다. 마지막으로 MCubic을 사용함으로써 n개의 서브플로우에 속하는 각 i^{th} 서브플로우는 식 (10)에 따라서 자신의 윈도우 크기를 주기적으로 조정한다.

$$W_i(t) = C(t-K_{mptcp}^{\ i})^3 + W_{\max}^{\ i} \quad (10)$$

$$K_{mptcp}^{\ i} = K_i^* \gamma \ with \gamma = \frac{\sum_{i=1}^{n}\frac{1}{K_i}}{\frac{1}{K_i}} \text{ and } K_i = \sqrt[3]{W_{\max}^{\ i}\beta/C} \quad (11)$$

식 (10)과 (11)에서 C는 scaling factor이며 구현할 때 0.4로 설정하였다. 정상적인 조건에서 $K_{mptcp}^{\ i}$는 항상 원래의 값 K_i보다 크다는 것을 보장한다. 하나의 서브플로우에서 혼잡이 발생했을 때 다른 서브플로우의 K_{mptcp}가 즉시 감소함으로써 다른 서브플로우의 전송률이 증가하고 혼잡되는 서브플로우의 전송률을 보상하게 된다.

IV. 구현 및 성능평가

4.1 MCubic 알고리즘의 리눅스에서의 MCubic 구현

본 연구에서는 리눅스 커널 3.14.17[1]에서 지원되는 MPTCP에 MCubic을 구현하였다. 기본적으로 MCubic는 Cubic과 유사하며, 본 연구에서 제안된 알고리즘은 Cubic 증가 함수에서의 K 변수에 적용되었다. 패킷 손실이 발생했을 때 느린 시작과 승수 감소(Multiplicative Decrease) 방식은 TCP Cubic과 동일하다. 따라서 다중 경로 TCP에서 Cubic TCP의 친화 함수(Friendliness Function)은 불필요하다. 결과적으로 Cubic TCP 친화 함수에서 사용된 몇 개의 변수를 없애고 K_{mptcp}의 재 연산에 필요한 함수를 사용하였다.

4.2 테스트베드와 성능평가

4.2.1 분리된 병목 시나리오

첫 번째 테스트베드(그림 2)에서는 공정성과 트래픽 이동 관점에서 다른 MCCA, 특히 TCP Cubic과 비교해서 MCubic의 성능을 평가하고자 한다.

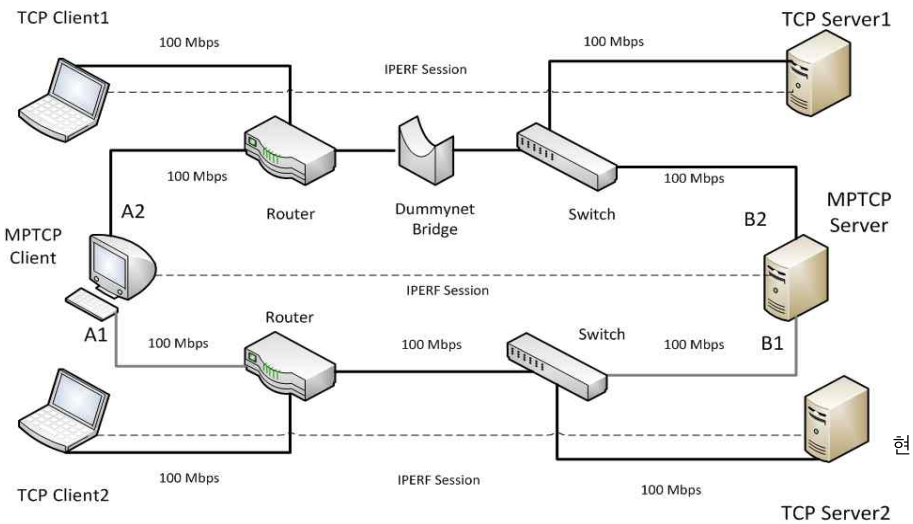


그림 2. 분리된 병목 시나리오
Fig. 2. Distinct bottleneck scenario

현

이를 위해서 그림 2의 테스트베드는 대칭적인 토폴로지를 가지며, 플로우 A1-B1과 A2-B2 플로우 서로 분리된 병목 링크를 갖고, 단일 경로 TCP Cubic과 링크를 공유한다. 알고리즘의 트래픽 이동 능력을 관찰하기 위해 두 개의 경로를 비대칭적으로 구성하였으며, A1-B1은 100Mbps, 그리고 두 번째 링크 A2-B2는 다양한 조건에서 측정하기 위해 10, 20, 30 그리고 50Mbps 용량으로 변화를 주었다. 링크의 용량을 수월하게 제어하기 위해서 각 경로에 dummynet bridge를 사용하여 링크의 용량을 원하는 대로 설정하였다. 단일 경로 TCP와 MPTCP의 플로우는 동일하게 시작하고 종료하였으며, 시간 동안 각 링크의 처리율(Mb/s)을 측정하였다.

그림 3에서는 다른 혼잡제어 방식에 따른 두 개의 단일 경로 TCP 플로우와 다중 경로 접속의 전체 처리율을 보여준다. 그림 3에서, MCubic 방식의 다중 경로 접속은 다른 방식보다 가장 적은 대역폭을 갖게 하며, 특히 TCP Cubic 인 경우에 가장 적은 대역폭을 갖게 한다. 이 것은 설계 제약 C1을 적용한 결과이며, 링크를 공유하는 MCubic의 각 서브플로우의 전송율을 낮춤으로써 이루어졌다. 또한 이 것은 링크를 공유하는 각 서브플로우가 덜 공격적이 되도록 한다. 이 결과는 MCubic가 다른 알고리즘보다 두 개의 단일 경로 TCP 플로우와 친화적(Friendly) 하다는 것을 보여준다. 또한 이 결과는 식 (12)과 같이 계산된 공정성 지수(Fairness Index)에서도 알 수 있으며, 표 1에서 계산결과를 보여준다.

이 시나리오에서는 TCP 플로우와 MPTCP 접속이 병목 링크를 공유하기 위해서는 공정성이 필요하다. 사용자 평균 처리율에 대한 공정성을 계산하기 위해서 식 (12)와 같이 Jain's Fairness Index[13]가 사용되었다.

$$Fairness(throughput) = \frac{\left\{\sum_{i=1}^{n} T_i\right\}^2}{n^*\left\{\sum_{i=1}^{n} T_i^2\right\}} \qquad (12)$$

식 (12)에서 T_i은 i_{th} 속의 처리율이고, n개의 개체가 경쟁을 한다. 실험 결과 MCubic의 트래픽 이동 효과로 인해서 모든 테스트의 경우에 MCubic의 공정성 지수가 가장 높았다. 특히 원래의 TCP Cubic과 비교했을 때 가장 좋은 결과를 보여줬다.

표 1. 분리된 병목 시나리오에서 공정성 지수
Table 1. The fairness indices of distinct bottleneck scenario

	10Mbps	20Mbps	40Mbps	50Mbps
LIA	0.66	0.7	0.74	0.82
OLIA	0.66	0.7	0.74	0.82
WVEAS	0.65	0.68	0.72	0.8
MCUBIC	0.66	0.7	0.75	0.82
CUBIC	0.63	0.65	0.68	0.75

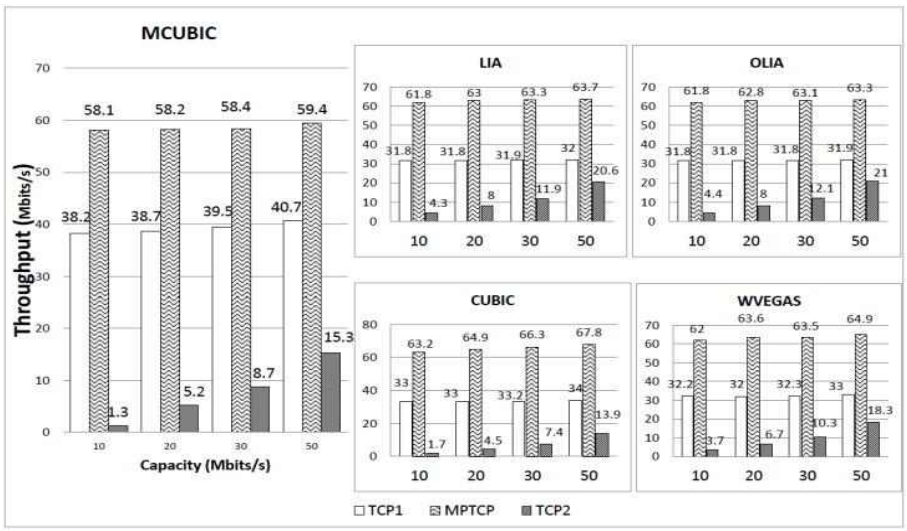


그림 3. 분리된 병목 시나리오에서 다양한 혼잡제어 방식에 대한 단일 경로 TCP 플로우와 MPTCP 접속의 처리율
Fig. 3. Comparison of throughputs between TCP and MPTCP of distinct bottleneck scenario

4.2.2 공유된 병목 시나리오

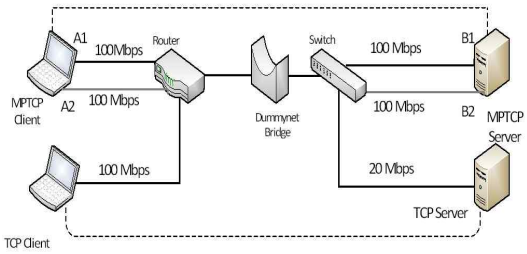


그림 4. 공유된 병목 시나리오
Fig. 4. Shared bottleneck scenario

이 시나리오(그림 4)에서는 MPTCP와 단일 경로 TCP Cubic 간에 병목 링크를 공유하는 구성을 하였다. 테스트베드의 모든 링크는 100Mbps의 용량을 갖는다. 분리된 병목 시나리오와 동일하게 단일 경로 TCP 플로우와 MPTCP 플로우는 동시에 시작되고 종료된다.

공유된 병목 시나리오에는 두 가지 경우가 있으며, 하나는 MPTCP 내의 접속이 동일한 지연을 갖는 경우(Case 1 : 링크의 지연이 동일한 경우)와 다른 지연을 갖는 경우(Case 2 : 링크의 지연이 다른 경우)로 구분된다.

가) 공유된 병목 시나리오에서 링크의 지연이 동일한 경우(Case1)

이 경우에 네트워크의 지연이 높은 상황에서 다양한 테스트를 했을 때 MPTCP의 효율을 관찰하고자 한다. [14][15]에서 인터넷의 지연이 100ms 이하이면 정상적인 경우로 간주되지만 위성을 통한 인터넷의 지연은 통상적으로 500ms 또는 그 이상으로 알려져 있다. 따라서 본 실험에서는 그 이상의 지연을 갖는 조건을 dummy-net bridge를 사용해서 구성하였다. 실험에서는 송신측에서 개별 접속에 대한 처리율(Mb/s)을 측정하였으며, 그림 5에서 결과를 보여준다.

지연이 100ms 이하의 좋은 조건에서는 다른 프로토콜과 유사하게 MCubic으로 구현된 MPTCP의 결과가 단일 경로 TCP의 처리율보다 높다는 것을 알 수 있다. 그러나 위성 인터넷 수준으로 지연(500ms)이 증가해도 TCP Cubic의 장점을 유지하며, 처리율은 단일 TCP 보다 높거나 같은 수준을 유지한다. 다른 방식과 비교할 때에는 단일 TCP 플로우보다 감소한다. 이 것은 첫 번째 설계 목표를 달성하지 못하는 결과이며 표 2에서 다중 경로 대 단일 경로의 비율을 계산함으로써 다시 설명되었다. MCubic을 사용함으로써 비율은 항상 1보다 크거나 같은 값을 유지하는 반면에 다른 프로토콜은 지연이 큰 네트워크에서는 1보다 작은 값을 보였다. 따라서 MCubic은 지연은 큰 네트워크에서 장점을 갖는다는 것을 알 수 있다.

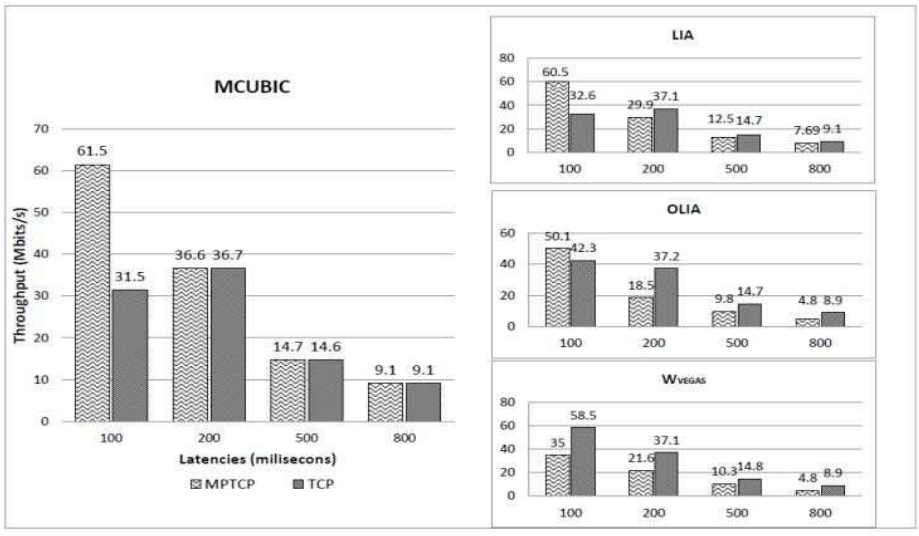


그림 5. 공유된 병목 시나리오에서 링크의 지연이 동일한 경우 단일 경로와 TCP와 MPTCP의 비교
Fig. 5. Comparison of throughputs between TCP and MPTCP of shared bottleneck scenario in homogeneous case

표 2. 공유된 병목 시나리오에서 링크의 지연이 동일할 때 공정성 지수
Table 2. The fairness indices of shared bottleneck scenario in homogeneous case

	100ms	200ms	500ms	800ms
LIA	1.86	0.8	0.86	0.75
OLIA	1.2	0,5	0.66	0.54
WVEGAS	0.6	0.6	0.7	0.54
MCUBIC	1.95	1	1	1

나) 공유된 병목 시나리오에서 링크의 지연이 다른 경우(Case 2)

이 경우는 MPTCP의 첫 번째 서브플로우와 단일 경로 TCP 플로우를 동일하게 좋은 조건인 100ms 지연을 갖도록 하였다. 그러나 두 번째 서브플로우의 링크는 dummynet bridge를 사용해서 300, 400 그리고 500ms의 높은 지연을 갖도록 변화시켜가면서 실험을 하였다.

그림 6은 두 개의 서브플로우가 비대칭일 때의 테스트 결과를 보여준다. 그림 6에서 MPTCP의 성능은 단일 경로 TCP 보다 떨어진다. 그 이유는 MPTCP의 수신 버퍼에서 패킷의 순서를 재 정렬하는데 시간이 많이 걸리는 때문이다. 따라서 RTT가 증가할수록 성능은 저하된다. 그리고 MCubic을 사용하는 MPTCP에서도 이러한 현상은 발생한다. 그럼에도 불구하고 MCubic은 MPTCP와 단일 경로 TCP와의 처리율 비를 유지하며, 이 결과는 표 3에서 알 수 있다.

표 3. 지연이 높은 네트워크에서 지연이 다른 경우 단일 경로 TCP와 MPTCP의 처리율 비
Table 3. Comparison of throughputs between TCP and MPTCP of delayed network in heterogeneous case

	100:300ms	100:400ms	100:500ms
LIA	0.56	0.43	0.35
OLIA	0.55	0.4	0.31
WVEGAS	0.32	0.25	0.21
MCUBIC	0.74	0.47	0.42
CUBIC	0.98	0.52	0.42

V. 결 론

본 연구에서 지연이 긴 링크 조건에서 다중 경로 전송(MPTCP)에 대한 혼잡제어에 대해서 새로운 알고리즘을 제안하고 성능을 측정하였다. 본 연구에서는 처리율과 같은 성능을 향상시키기 위한 방안인 MPTCP의 다중 경로 전송 방식으로 TCP Cubic의 확장 개념인 MCubic을 제안하였다. 이를 위해서 프로토콜의 설계 목표를 정하고, 제안된 프로토콜에 대해서 분석하였으며, 테스트베드를 구성해서 실험하였다.

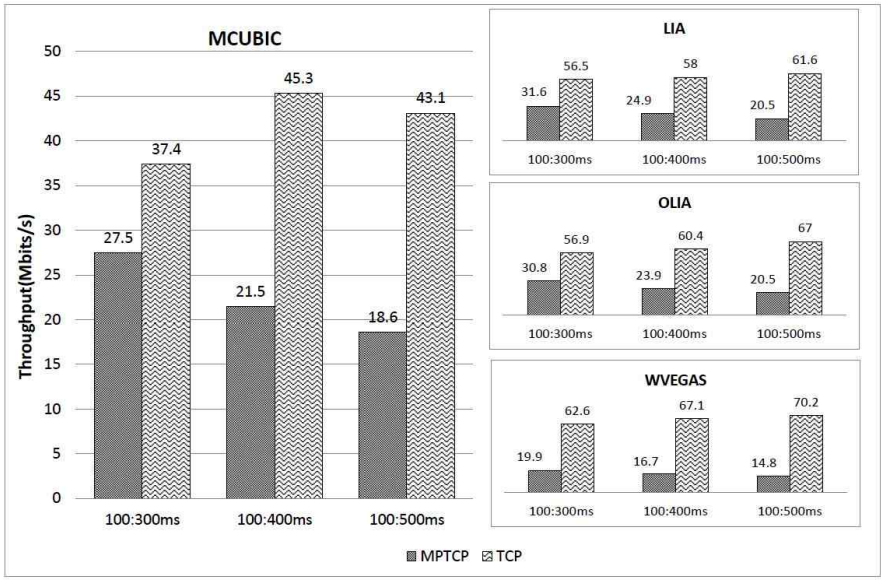


그림 6. 공유된 병목 시나리오에서 링크의 지연이 다른 경우 단일 경로와 TCP와 MPTCP의 처리율 비교
Fig. 6. Comparison of throughputs between TCP and MPTCP of shared bottleneck scenario in heterogeneous case

제안된 프로토콜은 두 가지의 제한사항을 고려해서 설계되었다. 두 가지의 제한사항은 MPTCP의 각 서브플로우는 링크를 공유하는 단일 경로 TCP에 대해서 공격성이 낮아야 한다는 것이며, 다른 하나는 서브플로우 사이에 공정성와 부하 균형을 유지하기 위해서 트래픽 이동이 이루어져야 한다는 것이다. 실험 결과 분리된 병목을 갖는 시나리오와 공유하는 병목 링크를 갖는 시나리오에서 제안된 MCubic은 MPTCP의 성능을 향상시킨 것을 확인하였다.

References

[1] Christoph Paasch, Sebastien Barre, et al., "Multipath TCP implementation in the Linux kernel", available from http://www.multipath-tcp.org

[2] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, Mar. 2011.

[3] Nguyen Van Dien and Soonghwan Ro, "Performance Evaluation of MPTCP over a Shared Bottleneck Link", International Journal of Computer and Communication Engineering, Vol. 3, No. 3, pp. 176-185, May 2016.

[4] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP Congestion Avoidance Algorithm Identification", IEEE/ACM Transactions on Networking, Vol. 22, No. 4, pp. 1311-1324, Aug. 2014.

[5] Ing. Luis Marrone, Lic. Andr´es Barbieri, and Mg. Mat ‘as Robles, "TCP Performance - CUBIC, Vegas & Reno", JCS&T, Vol. 13, No. 1, pp. 1-8, Apr. 2013.

[6] Sangtae Ha, Injong Rhee, and Lisong Xu, "CUBIC : A New TCP-Friendly High-Speed TCP Variant", ACM SIGOPS Operating System Review, Vol. 42, No. 5, pp. 64-74, Jul. 2008.

[7] C. Raiciu, M. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, Oct. 2011.

[8] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J. Y. Le Boudec, "Non-Pareto Optimality of MPTCP: Performance Issues and a Possible Solution", ACM CoNext, Dec. 2012.

[9] Cao, Y., Xu, M., and Fu, X., "Delay-based congestion control for multipath TCP", International Conference on Network Protocols, 30 Oct. 2012.

[10] Tuan Anh LE, et al., "A Multipath Cubic TCP Congestion Control with Multipath Fast Recovery over High Bandwidth-Delay Product Networks", IEEE Trans. on Communication, Vol. E95, B, No. 7, pp. 2232-2244, Jul. 2012.

[11] Brett Levasseur, Mark Claypool, and Robert Kinicki, "A TCP CUBIC implementation in ns-3", Proceedings of the 2014 Workshop on ns-3, Article No. 3, May 2014.

[12] http://lxr.free-electrons.com/source/net/ipv4/tcpcupic.c [accessed: Jun. 01, 2018]

[13] 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.

[14] http://compnetworking.about.com/od/speedtests/a/network_latency.htm. [accessed: Jun. 01, 2018]

[15] Y. J. Cho, H. S. Kim, and P. J. Park, "Smart Outlet System for Single-person Household based on IoT (Internet of Things)", Journal of DCS, Vol. 18, No. 5, pp. 895-904, Aug. 2017.

저자소개

노 승 환 (Soonghwan Ro)

1987년 : 고려대학교 전자공학과 (공학사)
1989년 : 고려대학교 전자공학과 (공학석사)
1993년 : 고려대학교 전자공학과 (공학박사)
2003년 : 영국버밍엄 대학교 초빙연구원
1994 ~ 현재 : 국립 공주대학교 정보통신공학부 교수
관심분야 : 적외선 시스템, 영상처리, 임베디드시스템, Mobile IP

류 장 렬 (Jangryeol Ryu)

1982년 : 인하대학교 전자공학과 (공학사)
1985년 : 충남대학교 전자공학과 (공학석사)
1995년 : 충남대학교 전자공학과 (공학박사)
2013년 : 미국 LWIT 빙문연구교수
1992년 ~ 현재 : 국립 공주대학교 전기전자제어공학부 교수
관심분야 : 반도체소자, LED, 디스플레이부품

최 진 규 (Jinkyu Choe)

1980년 : 고려대학교 전자공학과(공학사)
1982년 : 고려대학교 전자공학과 공학석사
1987년 : 고려대학교 전자공학과 공학박사
2005년 ~ 2006년 : 미국 University of Arizona 방문 교수
2018년 8월 현재 : 한남대학교 전자공학과 교수
관심분야 : 통신망 성능평가, 디지털시스템 설계, Embedded System