The Transactions of the Korea Information Processing Society
TKIPS Vol.13, No.8, pp.349~355
eISSN: 3022-7011
https://doi.org/10.3745/TKIPS.2024.13.8.349

Optimizing Transmission Efficiency with Dynamic Bandwidth Aware Congestion Control (DBAC) in High-Speed Multipath Networks

Han Kimoon†

ABSTRACT

This paper proposes a new congestion control algorithm, Dynamic Bandwidth Aware Congestion Control (DBAC), to enhance data transmission efficiency in modern network environments that require the use of multiple paths. Traditional single-path TCP has limitations that are addressed by Multipath TCP (MPTCP), which can utilize multiple paths simultaneously to maximize bandwidth and improve transmission reliability. However, MPTCP suffers performance degradation in high-speed, long-distance networks due to path characteristic asymmetry. To overcome this, DBAC combines TCP CUBIC for paths with high BDP and LIA for regular paths, optimizing resource utilization and fairness. Experimental results show that DBAC significantly improves resource utilization and transmission performance, effectively using over 80% of the bandwidth on high BDP paths, compared to less than 20% with traditional LIA.

Keywords : MPTCP, Congestion Control, High-speed Network, Dynamaic Bandwidth Aware

고속 다중 경로 네트워크에서 동적 대역폭 인식 혼잡제어(DBAC)를 통한 전송 효율 최적화

한 기 문†

요 약

본 논문은 다중 경로를 사용해야 하는 현대 네트워크 환경에서 데이터 전송 효율을 향상시키기 위한 새로운 혼잡 제어 알고리즘인 동적 대역폭 인식 혼잡 제어(DBAC)를 제안한다. 전통적인 단일 경로 TCP는 한계가 있어 다중 경로 TCP(MPTCP)가 여러 경로를 동시에 활용하여 대역폭을 극대화하고 전송 신뢰성을 향상시키지만, MPTCP는 경로 특성의 비대칭성으로 인해 고속 장거리 네트워크에서 성능 저하를 겪는다. 이를 해결하기 위해 DBAC는 BDP가 큰 경로에는 TCP CUBIC을, 일반 경로에는 LIA를 결합하여 자원 활용과 공정성을 최적화한다. 실험 결과, DBAC는 자원 활용과 전송 성능을 크게 향상시켜, 기존 LIA가 20% 미만의 대역폭을 사용하는 것에 비해 높은 BDP 경로에서 80% 이상의 대역폭을 효과적으로 사용하는 것으로 나타났다.

키워드 : MPTCP, 혼잡제어, 고속네트워크, 동적 대역폭 인식

1. 서 론

현대 네트워크 환경에서의 데이터 전송은 다양한 경로와 복잡한 요구사항을 충족해야 한다. 전통적인 전송 제어 프로토콜(TCP)[1,2]은 단일 경로를 통해 데이터를 전송하는 방식으로 설계되었으며, 이는 여러 경로를 동시에 사용할 수 있는 현대의 네트워크 환경에서 한계를 보인다. 이러한 문제를 해결하기 위해 다중 경로 TCP(Multipath TCP)[3]가 도입되었다. MPTCP는 기존 TCP의 확장으로, 하나의 연결에서 여러 경로를 사용할 수 있게 하여 대역폭을 극대화하고 전송의 신뢰성을 높인다.

MPTCP는 Ethernet, Wi-Fi, 4G, 5G, 위성통신 등 다양한 네트워크 인터페이스를 동시에 활용하여 데이터 전송을 가능하게 한다. 이는 스마트폰, 태블릿, 노트북 등 여러 기기가 여러 네트워크에 동시에 연결될 수 있게 하며, 하나의 경로에서 발생하는 병목 현상을 피하고 보다 안정적인 데이터 전송을 지원한다. 특히, MPTCP는 지연, 손실, 비용, 대역폭 등 네트워크의 조건에 따라 최적의 경로를 선택하거나, 여러 경로를 동시에 사용하여 더 높은 전송 속도를 달성할 수 있다.

MPTCP의 주요 장점 중 하나는 기존의 TCP와의 호환성이다. MPTCP는 기존 TCP 연결을 유지하면서도 추가적인 경로를 통해 데이터를 전송할 수 있어, 네트워크 장애 시 다른 경

† 정 회 원 : 국방과학연구소 연구원
Manuscript Received : July 9, 2024
First Revision : July 18, 2024
Accepted : July 31, 2024
* Corresponding Author : Han Kimoon(kmhan14@add.re.kr)

로로 트래픽을 자동으로 전환하여 연속성을 보장한다. 이는 네트워크의 효율성을 높이고, 데이터 전송의 안정성을 강화하는 데 큰 역할을 한다. 그러나 고속 장거리 네트워크를 포함하는 다중 경로에서는 경로 특성의 비대칭성으로 인해 MPTCP의 트래픽 전송 성능이 저하되는 문제가 발생한다. 특히, 높은 대역폭-지연 곱(BDP)을 가지는 경로에서는 기존의 혼잡제어 알고리즘인 LIA(Linked Increase Algorithm)는 느린 Cwnd 증가 현상과 패킷 손실이 발생했을 절반으로 감소시키는 현상으로 인해 최대 자원을 활용하지 못하는 지속적인 성능 저하를 겪는다.

본 연구에서는 고속 장거리 네트워크 환경에서 MPTCP의 자원 활용을 최적화하기 위한 새로운 혼잡제어 DBA(Dynamic Bandwidth Aware) 알고리즘을 제안한다. 제안 알고리즘은 동적으로 BDP를 감지하고, BDP가 큰 경로에서는 RTT에 의존적이지 않은 CUBIC을 활용하여 빠르게 Cwnd를 증가하여 최대 자원을 활용한다. 일반 경로에서는 LIA를 결합하여 사용한다. 이를 통해 네트워크 자원의 효율적인 활용과 공정성을 동시에 달성하고자 한다.

2. 관련 연구

이 절에서는 MPTCP 혼잡제어에 대해 소개하고, 고속 장거리 네트워크에서 성능이 검증된 CUBIC에 대해 설명한다. 마지막으로 현재 MPTCP 혼잡 제어의 한계를 설명한다.

2.1 MPTCP 혼잡제어

MPTCP는 하나의 세션에서 여러 경로를 통해 데이터를 전송함으로써 네트워크 자원의 활용도를 극대화하는 것을 목표로 한다. LIA[4]는 MPTCP의 혼잡 제어 알고리즘 중 하나로, 경로 간의 공정성을 유지하면서 다중 경로에서의 트래픽 분배를 최적화하기 위해 설계되었다. Equation (1), (2)와 같이 증가계수 α를 계산하고, 각 서브플로우의 Cwnd(Congestion Window)를 조정하여 전체적인 전송 속도를 증가시키는 방식으로 작동한다. LIA는 세 가지 주요 목표를 달성하려고 한다. 첫째, 다중 경로 흐름은 단일 경로 흐름보다 더 많은 대역폭을 사용하지 않아 공격적이지 않도록 한다. 둘째, 사용 가능한 최상의 경로에서 단일 경로 TCP와 유사한 성능을 제공하고 전체 성능 향상을 시킨다. 셋째, 병목 링크에서 기존의 단일 경로 TCP와 공정하게 대역폭을 공유한다. 각 서브플로우의 Cwnd를 개별적으로 관리하는 대신, 전체적인 네트워크 상태를 고려하여 조정한다. 이를 통해 각 서브플로우가 독립적으로 동작하여 발생할 수 있는 불공정성과 비효율성을 줄인다. 경로 간 RTT(Round-Trip Time)의 차이를 고려하여 Cwnd를 증가시키는 방식을 사용하며, 이를 통해 네트워크 자원의 효율적인 활용을 충족시킨다.

$$\alpha = \sum_r w_r \times \frac{Max_r \frac{w_r}{rtt_r^2}}{(\sum_r \frac{w_r}{rtt_r})^2} \quad (1)$$

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

MPTCP OLIA(Opportunistic Linked Increase Algorithm) 혼잡 제어[5]는 LIA의 한계를 보완하기 위해 제안된 MPTCP 혼잡 제어 알고리즘이다. 특히 이기종 네트워크 환경에서의 성능 향상과 공정성을 목표로 한다. OLIA는 LIA와 유사하게 여러 경로 간의 트래픽 분배를 최적화하지만, 좀 더 공격적인 방식으로 Cwnd를 조정한다. 주요 특징은 다음과 같다. 기회주의적 접근을 통해 경로 간의 혼잡 상태에 따라 더 유리한 경로에 트래픽을 집중시킨다. 기존의 단일 경로 TCP와의 공정성을 유지하면서 다중 경로 자원을 최대한 활용한다. 이기종 네트워크 환경에서도 높은 성능을 제공하도록 설계되었다. 경로별 혼잡 상태를 실시간으로 모니터링하고, 혼잡이 적은 경로로 더 많은 트래픽을 분산시킨다. 이는 경로 간 RTT의 차이를 적극 활용하여 트래픽을 분배하는 방식으로, 네트워크 자원의 활용도를 높인다. OLIA는 LIA보다 더 동적인 방법을 사용하여 경로 선택과 트래픽 분배를 최적화하며, 이를 통해 네트워크 전송 성능을 극대화한다.

2.2 TCP CUBIC 혼잡 제어

CUBIC[6]은 고속 네트워크 환경에서 TCP 성능을 최적화하기 위해 개발된 혼잡 제어 알고리즘이다. 이는 특히 대역폭이 크고 지연 시간이 긴 네트워크에서 효율적으로 작동하도록 설계되었다. CUBIC은 기존의 TCP 혼잡 제어 알고리즘인 RENO와 TCP BIC의 한계를 극복하기 위해 제안되었다. CUBIC의 주요 특징은 다음과 같다.

삼차 함수 기반 증가: CUBIC은 Cwnd 크기를 시간의 삼차 함수로 증가시킨다. 이는 Cwnd의 크기가 시간이 지남에 따라 비선형적으로 증가하게 하여, 네트워크의 혼잡 상태에 더 잘 적응할 수 있도록 한다. 패킷손실 이후 빠르게 윈도우 크기를 복구하고, 이후 점진적으로 대역폭을 탐색하는 과정을 거친다.

RTT 독립적: Cwnd 크기의 증가가 RTT와 무관하게 이루어지므로, 다양한 지연 시간을 가진 네트워크 환경에서도 안정적인 성능을 제공한다. 이는 네트워크의 지연 시간에 따라 Cwnd 크기가 영향을 받지 않도록 설계되어, 짧은 RTT를 가진 흐름과 긴 RTT를 가진 흐름 간의 공정성을 높인다.

빠른 회복: 패킷 손실 이후 Cwnd 크기가 빠르게 회복되도록 설계되어, 대역폭을 빠르게 재활용할 수 있다. 이는 고속 네트워크에서 데이터 전송 속도를 유지하는 데 중요한 역할을 한다.

이와 같은 특성으로 CUBIC은 대역폭이 크고 지연 시간이 긴 네트워크에서도 높은 대역폭 활용률을 유지하면서 안정적인 혼잡 제어를 할 수 있다. 데이터 센터 간 연결이나 국제적인 네트워크 백본과 같은 환경에서 CUBIC은 TCP 성능을 크게 향상시킬 수 있다. 리눅스 커널, 윈도우, 애플 스택에서도 기본 TCP 혼잡 제어 알고리즘으로 채택되어 널리 사용되고 있다.

2.3 높은 BDP 경로에서 MPTCP 성능

MPTCP 혼잡제어의 개선에 대한 다양한 연구에도 불구하고, MPTCP는 여러 경로를 통해 데이터를 전송함으로써 전송 성능을 향상시키고 네트워크의 신뢰성을 높이는 강력한 프로토콜이다. 그러나 지연 시간이 길고 종단 간의 거리가 먼 네트워크에서 MPTCP는 성능저하 문제가 나타난다.

1) 높은 BDP의 영향

고지연 장거리 네트워크는 높은 BDP를 특징으로 한다. 이는 네트워크의 데이터 전송 시 Cwnd 크기를 크게 해야 함을 의미한다. 그러나 MPTCP의 기존 혼잡 제어 알고리즘인 LIA와 OLIA는 높은 BDP 환경에서 네트워크 자원을 효율적으로 활용하지 못하고 대역폭 사용률이 낮아지는 성능저하가 발생한다[7].

2) 경로 간의 비대칭성 문제

경로별로 지연 시간, 대역폭, 패킷 손실률 등이 크게 달라질 수 있는 비대칭적인 특성을 가진다. 이러한 비대칭성은 MPTCP가 각 서브플로우의 상태를 효율적으로 관리하고 트래픽을 적절히 분배하는 데 어려움을 겪게 한다. 특히, 긴 지연 시간을 가진 경로에서 패킷이 지연됨에 따라 전체 전송 속도가 감소할 수 있다[8].

3) 혼잡 제어의 한계

MPTCP는 다양한 경로에서 동시에 데이터를 전송함으로써 혼잡 제어를 수행해야 한다. 그러나 고지연 환경에서는 Cwnd 크기를 적절히 조정하는 것이 어려워진다. 이는 패킷 손실이나 지연이 발생할 때 Cwnd를 신속하게 조정하지 못해 전송 성능이 저하되는 결과로 나타난다[9].

4) 네트워크 관리의 복잡성

효율적인 운영을 위해서는 경로 선택, 서브플로우 관리, 패킷 스케줄링 등 다양한 네트워크 관리 작업을 포함하는 정교한 네트워크 관리가 필요하다. 이러한 복잡성은 MPTCP의 구현과 운영을 어렵게 만들며, 네트워크 관리자의 추가적인 노력이 요구된다[10].

고지연 장거리 네트워크에서 MPTCP를 효과적으로 활용하기 위해서는 이러한 문제점을 해결할 수 있는 새로운 혼잡제어 알고리즘과 네트워크 관리 기법이 필요하다. 이를 통해 네트워크 자원의 효율적인 활용과 전송 성능의 향상을 달성하고자 한다.

3. 제안 혼잡제어

이 절에서는 MPTCP가 긴 지연 시간과 큰 대역폭을 가진 이기종 네트워크에서 사용될 때 링크 활용도를 향상시키기 위한 DBA 혼잡제어 알고리즘 설계를 소개한다. 높은 대역폭을 감지한 후 Cwnd가, 연결된 증가 방식으로 증가하도록 LIA의 증가 계수를 적절히 계산하는 알고리즘을 포함하여 CUBIC으로 동작한다.

일반적인 MPTCP 혼잡 제어는 ACK가 수신될 때 RENO와 유사한 AIMD(Additive Increase Multiplicative Decrease) 기술을 사용하여 Cwnd를 조작한다. 일반 유선 네트워크에서는 최소 10^{-6}의 패킷 오류율이 발생하기 때문에 Cwnd가 최대 링크 용량에 도달할 수 없으므로 광대역 링크의 대역폭을 충분히 사용할 수 없다. 따라서 LIA와 유사한 혼잡 제어는 큰 BDP 네트워크에서 Cwnd 증가 속도를 늦춘다. MPTCP 혼잡 제어를 설계할 때는 큰 BDP 경로를 가진 다중 경로 네트워크를 고려해야 한다.

DBAC(Dynamic Bandwidth Aware Congestion Control)는 BDP가 큰 경로를 가진 서브플로우에서 CUBIC 혼잡 제어를 사용하여 작동한다. BDP가 작은 서브플로우에서는 LIA와 같이 Cwnd를 AIMD 기술을 사용하여 조정하면서 MPTCP 증가 계수 α를 참조하여 여러 서브플로우의 Cwnd를 연결된 증가 방식으로 제어한다. 큰 BDP를 가진 서브플로우의 Cwnd를 그대로 사용하면 원래 LIA의 값과 다르게 α값이 비정상적으로 왜곡된다. 따라서, BDP가 큰 서브플로우의 Cwnd를 가상 LIA 윈도우 값으로 변환한 다음 LIA의 α를 계산하는 알고리즘을 설계하였다. LIA가 각 경로의 공정성을 고려하여 동작하는 것과 같이 Cwnd를 계산한다. CUBIC이 동작하는 BDP가 큰 경로는 LIA Cwnd와 결합하여 혼잡 제어를 수행하여 네트워크 자원을 최대로 활용할 수 있는 장점이 있다. 그러나 Equation (1), (2)에서 볼 수 있듯이 CUBIC의 Cwnd 값을 사용하여 LIA의 증가 계수 α를 계산하여 활용하면 작은 BDP 경로에서 Cwnd의 증가 속도가 너무 느려진다. 이는 모든 사용 가능한 경로의 자원을 최대한 활용하려는 MPTCP의 설계 목표에 반한다. 따라서 CUBIC이 동작하는 경로에서 LIA가 작동하는 것처럼 정규화된 가상 Cwnd 값을 계산하고, 이를 사용하여 다른 경로의 증가 계수 α를 계산할 필요가 있다.

DBAC 알고리즘은 다음과 같이 작동한다. BDP가 작은 경로에서 Cwnd를 증가시킬 때, LIA처럼 동작하여 Cwnd 증가 계수 α를 채택한다. BDP가 작은 경로의 혼잡 제어는 CUBIC의 정규화된 가상 Cwnd를 적용하여 큰 BDP 경로에서 MPTCP LIA가 작동하는 것처럼 보인다면, BDP가 작은 경로의 Cwnd 변화율은 기존 LIA와 크게 다르지 않다. 이는 병목에서 다른

단일 경로 TCP와의 공정성을 유지할 수 있다. CUBIC Cwnd를 LIA의 가상 Cwnd로 정규화하는 방법은 RENO와 CUBIC의 평균 Cwnd Equation (3), (4)의 비율로 나타낼 수 있다. p는 패킷 손실률을 나타내고 고정된 상수 값을 활용하였다. 동일한 패킷 손실률이 있는 네트워크 환경에서 가상 RENO 윈도우 크기 $Virtual\,W_{RENO}$는 Equation (3), (4)에 표시된 두 Cwnd의 비율인 Equation (5)가 된다. $Current\,W_{CUBIC}$은 BDP가 큰 경로에서 CUBIC으로 동작했을 때 Cwnd 값이다.

알고리즘의 성능 평가를 위해, 리눅스 커널 MPTCP 버전의 기본 LIA 소스 코드를 기반으로 필요한 매개변수, CUBIC의 증가 및 감소 알고리즘을 추가하여 구현하였다.

$$Avg\,W_{RENO} = \sqrt{\frac{3}{2p}} \tag{3}$$

$$Avg\,W_{CUBIC} = 1.054 \times \sqrt[4]{RTT^3} \div \sqrt[4]{p^3} \tag{4}$$

$$Virtual\,W_{RENO} = Current\,W_{CUBIC} \times \frac{Avg\,W_{RENO}}{Avg\,W_{CUBIC}} \tag{5}$$

$$W_{DBAC} = Virtual\,W_{RENO} \times \sqrt{\alpha_1} \times \frac{Avg\,W_{CUBIC}}{Avg\,W_{RENO}} \tag{6}$$

Fig. 1은 CUBIC의 Cwnd를 정규화하여 얻은 가상 TCP RENO Cwnd를 사용하여 LIA의 증가 계수 α를 계산하는 DBAC 흐름도를 보여준다. MPTCP 세션이 다중 경로를 사용하여 설정되고 각 서브플로우의 특성이 식별된다. BDP가 큰 경우 DBAC로 선택한다. 반대로 작은 경우 LIA가 동작한다. 해당 경로의 전송된 Cwnd, RTT, MSS(Maximum Segment Size) 값을 사용해서 BDP를 측정하고, 설정한 임계값을 기준으로 시작 판단 조건인 BDP의 크고 작음을 결정한다. DBAC는 큰 BDP 경로에서 작동하며, 일부 사전 정의된 전역 변수에 가상 Cwnd와 지연 시간을 저장한다. BDP가 작다고 인식되면 증가 계수가 계산된다. 계산 매개변수는 나머지 서브플로우에서 수신된 Cwnd와 지연 시간, 그리고 큰 BDP 경로에 대해 전역 변수에 저장된 가상 Cwnd와 지연 시간이다. 큰 BDP 경로의 Cwnd는 Equation (5)에 의해 가상 RENO Cwnd로 변환된다. Fig. 1에서 $\alpha 1$은 Equation (1)과 같이 RENO Cwnd, 나머지 경로의 다른 Cwnd 및 RTT를 사용하여 계산된다. 그 후 가상 RENO Cwnd는 $\sqrt{\alpha_1}$로 곱해지고, 최종적으로 LIA 알고리즘이 적용된 Cwnd를 얻는다. RFC6356[4]에 LIA 평균 Cwnd는 Equation (3)에 증가계수 $\sqrt{\alpha}$가 곱해진 값으로 명시되어 있다. 최종적으로, 가상 LIA Cwnd는 $\sqrt{\alpha_1}$를 곱하여 만들어져야 한다.

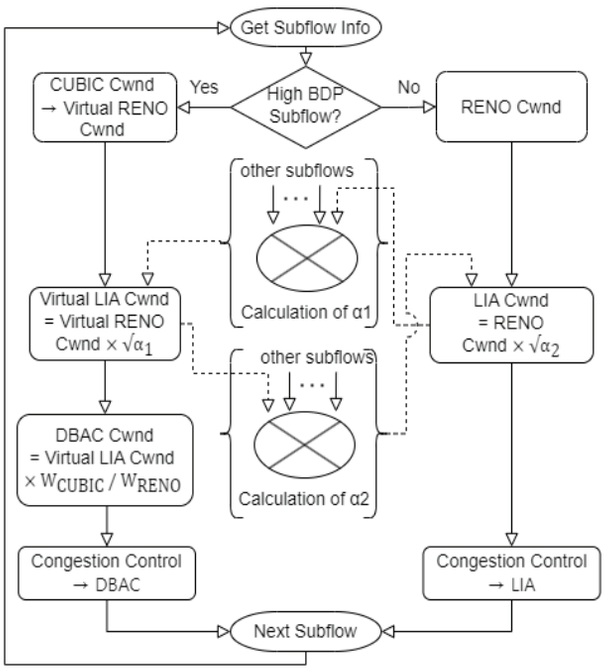

Fig. 1. DBAC Flow Chart

리눅스 커널의 LIA 소스코드에서는 계산의 복잡성을 줄이고 부동 소수점 숫자의 조작을 제거하기 위해 정규화된 α값을 사용하여 계산을 수행하였다. CUBIC의 Cwnd를 가상 RENO Cwnd로 변경하기 위해, Equation (3)~(5)을 사용하여 커널에 RTT 값에 해당하는 변환 테이블을 만들었다.

증가 계수 $\alpha 2$는 최종적으로 얻어진 가상 Cwnd, 나머지 경로의 다른 Cwnd 및 RTT를 사용하여 계산된다. 다른 경로의 Cwnd를 참조하는 부분은 점선 화살표로 표시된다. $\alpha 2$는 LIA가 나머지 작은 BDP 경로에서 작동할 때 사용된다. $\alpha 1$과 $\alpha 2$의 값은 알고리즘이 진행됨에 따라 올바른 값으로 수렴하도록 반복적으로 계산된다. 큰 BDP 경로에서는 $\alpha 1$이 Equation (6)에 표시된 대로 DBAC Cwnd를 계산하는 데 사용된다. 그런 다음 W_{DBAC}는 큰 BDP 경로로 전송되는 데이터 양을 제어하는 데 적용된다. 작은 BDP 경로에서는 광대역 경로가 공정성을 유지하기 위해 LIA 서브플로우 중 하나로 인식된다. 따라서 DBAC가 사용될 때 일반 경로에서 Cwnd 증가율이 감소하지 않으며, 자원은 공정성 측면에서 기본 LIA와 동일한 방식으로 공유된다.

4. 실험 결과

이 절에서는 작은 테스트베드에서 실험을 통해 DBAC 알고리즘의 MPTCP 처리량과 공정성을 평가하였다.

4.1 실험 환경

DBAC와 LIA의 성능을 비교하기 위해, Fig. 2와 같이 실험

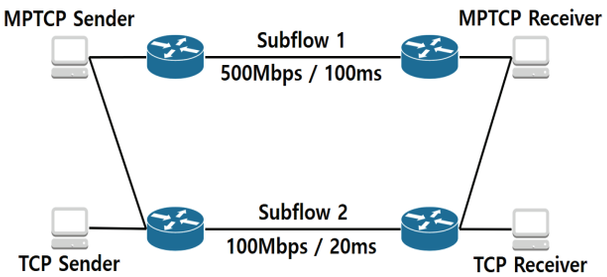

Fig. 2. Testbed Network Topology

Table 1. Experimental Setup

Path	BDP	Bandwidth (Mbps)	Delay(ms)	Error Rate
Path 1	High	500	100	10^{-6}
Path 2	Low	100	20	

환경을 구성하였다. 총 8대의 PC가 Fig. 2에 나타난 테스트베드를 위해 사용되었다. 모든 PC에 리눅스 OS 우분투 18.04 LTS를 설치하였다. MPTCP 송신자와 수신자에서는MPTCP 그룹이 제공한 커널의 기본 LIA 소스 코드를 수정하여 DBAC를 구현하였다. 광대역 이기종 네트워크를 구성하기 위해 버퍼 크기와 수신 윈도우 설정을 적절히 변경하였다.

실험을 위한 주요 환경 변수 값은 Table 1과 같다. 데이터가 전송되는 서브플로우 1과 2의 모든 중간 지점의 버퍼 크기는 해당 BDP의 약 100%로 설정하였고, 수신 윈도우 크기는 혼잡제어 성능을 제한하지 않을 정도로 충분히 크게 설정하였다. 각 NIC의 대역폭은 1 Gbps로 설정되었다. 서브플로우 1의 중간 노드에서 Dummynet의 QoS 설정을 사용하여 대역폭을 500 Mbps, 지연 시간을 100 ms로 구성하였다. 서브플로우 2의 물리적 대역폭은 100 Mbps로 설정되고 지연 시간은 20 ms로 설정하여 작은 BDP 경로를 구성하였다. 각 노드에서 정적 라우팅을 설정하여 MPTCP 송수신기가 서브플로우 1과 2의 경로를 각각 통과하도록 하였다. TCP 흐름은 서브플로우 2와 동일한 경로를 통과하도록 하였다. 패킷 손실률은 일반 유선 네트워크의 손실률인 10^{-6}으로 서브플로우 1과 2의 병목 링크에 설정되었다.

TCP 트래픽은 Iperf 도구를 사용하여 생성되었으며, 최대 세그먼트 크기는 1460 바이트로 설정되었다. Wireshark를 사용하여 총 전송 시간과 평균 전송속도를 측정하였다. 가상 Cwnd의 값을 측정하기 위해 소스 코드에 특정 변수를 정의하였으며, 그 변수의 값은 tcp_probe 함수에서 읽고 그래프로 표시하였다.

4.2 실험 결과

Fig. 3은 장시간 유지된 MPTCP 세션의 실험 결과를 보여준다. Fig. 2의 테스트베드를 구성하여 MPTCP 송신기가 서브플로우 1과 2를 통해 500초 동안 트래픽을 전송한다. 단일 경로 TCP 송신기는 서브플로우 2와 동일한 경로를 통해 트래픽을 전송하여 공정성 성능을 비교한다. 서브플로우 1은 BDP가 큰 경로이며, 버퍼 크기를 포함한 최대 Cwnd 크기는 8332이다. Fig. 3의 서브플로우 1 그래프에서 LIA Cwnd는 최대 약 1300까지 증가한다. Cwnd는 증가와 감소를 반복하지만 최대 용량에 도달할 수 없다. BDP가 큰 경로에서도 링크 자원의 20% 미만 사용되며 나머지 자원은 낭비된다.

Fig. 4는 BDP가 큰 경로에서 DBAC 혼잡제어로 작동하는 서브플로우 1의 Cwnd 그래프를 나타낸다. 가상 Cwnd는 서브플로우 2의 증가 계수 계산에 영향을 미친다. Fig. 5는 BDP가 작은 경로를 통해 전송된 서브플로우 2의 DBAC Cwnd를 보여준다.

Fig. 3과 4의 결과를 비교하면, 기본 LIA와 이 논문에서 제안한 DBAC의 성능 차이를 명확히 알 수 있다. 먼저, 서브플로우 1의 전송량을 비교하면, 예상대로 DBAC 전송량이 LIA 방식보다 훨씬 크다는 것을 알 수 있다. Fig. 3과 5에서 BDP가 작은 서브플로우 2의 Cwnd 변화를 비교해 보면 유사한 패턴을 보인다. 이는 정규화된 가상 Cwnd를 사용하여 증가 계수 α를 계산하는 것이 LIA에서의 연결된 Cwnd를 계산하는 것과 유사함을 보여준다.

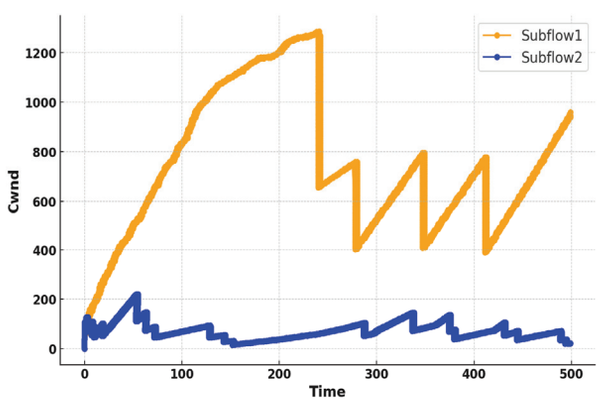

Fig. 3. LIA Congestion Window Variation Graph

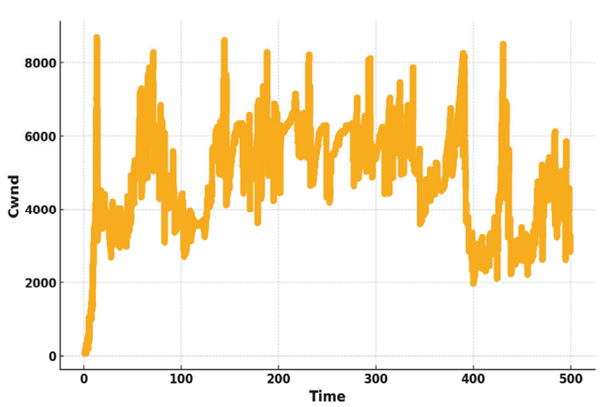

Fig. 4. DBAC Congestion Window Variation Graph(Sub 1)

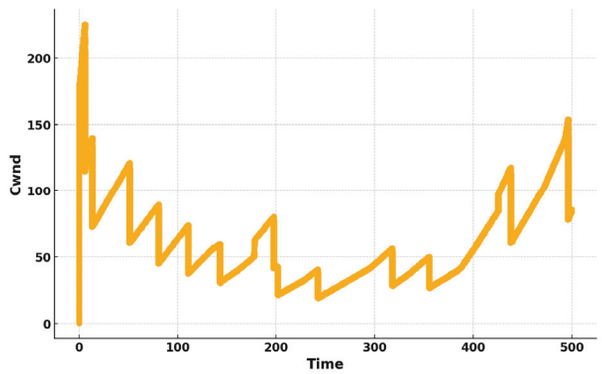

Fig. 5. DBAC Congestion Window Variation Graph(Sub 2)

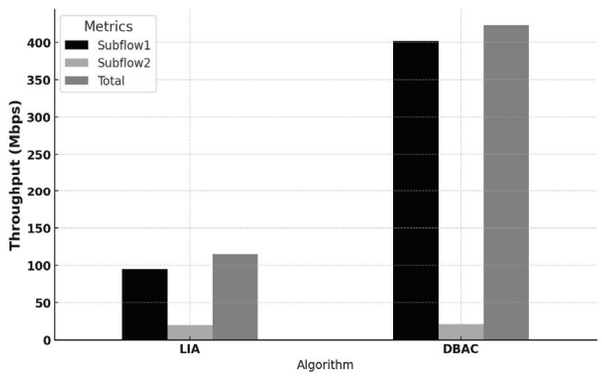

Fig. 6. Average Thrpughput Comparison

Fig. 6은 Fig. 3~5에서 보여준 실험 결과의 평균 전송 성능을 나타낸 그래프이다. Wireshark 도구를 사용하여 각 경로 및 전체 경로에 대한 패킷을 캡처하고 평균 전송 성능을 분석하였다. LIA, DBAC의 성능을 평가하기 위해 서브플로우 1, 2 및 전체 전송 속도를 보여준다.

LIA의 서브플로우 1은 다른 트래픽에 방해받지 않으므로 충분한 시간이 지나면 최대 전송 속도인 500 Mbps에 가까운 속도를 사용해야 한다. 그러나 잦은 패킷 손실과 느린 Cwnd 증가로 인해 경로의 전송 속도는 약 95 Mbps로 제한된다. 이는 경로 자원의 약 1/5만 사용된다는 것을 의미한다.

DBAC 서브플로우 1은 전체 대역폭의 약 80%인 400Mbps 이상을 사용한다. LIA를 사용할 때보다 4배 더 많은 자원을 사용하는 것이다. 제안한 DBAC를 사용하면 큰 BDP 경로의 자원 활용 효율성이 크게 향상된다는 것을 알 수 있다.

DBAC의 설계 목적은 큰 BDP 경로에서는 MPTCP 혼잡 제어가 CUBIC 혼잡 제어처럼 작동하도록 하고, 작은 BDP 경로에서는 LIA처럼 작동하도록 하는 것이다. 따라서 LIA가 동작하는 경로에서는 CUBIC이 동작되는 경로가 LIA가 작동되는 것처럼 가정해야 한다. 정규화된 가상 Cwnd 값을 수신하여 다른 서브플로우의 증가율 계산에 적용해야 한다. RENO로 변환된 Cwnd를 그대로 사용하면 LIA 동작과 다른 왜곡이 발생한다. Equation (5)에서 계산된 $Virtual\ W_{RENO}$는 RENO가 독립적으로 운영된다고 가정할 때의 Cwnd 값이며, 이는 LIA가 적용되었을 때와는 $\sqrt{\alpha_1}$ 만큼 차이가 나는 값이다. $Virtual\ W_{RENO}$에 $\sqrt{\alpha_1}$를 곱하여 얻은 값을 사용하여 $\alpha 2$를 다시 계산하고, $\alpha 2$를 다른 증가율 계산에 반복적으로 적용해야 한다. 서브플로우 2의 평균 전송 속도를 비교해보면, LIA와 DBAC는 각각 20 Mbps와 21 Mbps로 유사하다.

실험 결과를 통해, DBAC는 큰 BDP 경로에 사용하면 전송 성능이 크게 향상됨을 확인하였다. MPTCP 연결이 이루어지면, 큰 BDP 경로에서 자원을 충분히 활용한다. 또한, 가상 Cwnd 변환 공식을 적용함으로써 작은 BDP 경로가 원래 LIA와 유사하게 작동하고 MPTCP의 성능 기준을 준수하는 것을 확인하였다.

5. 결 론

이 논문에서는 새로운 MPTCP 혼잡 제어 DBAC 알고리즘을 설계하고, 큰 BDP 경로를 포함하는 다중 경로 네트워크에서 MPTCP의 전송 성능을 향상시키는 성능 평가를 수행했다. 고속 장거리 네트워크에서는 기존의 MPTCP 혼잡 제어가 패킷 손실과 느린 Cwnd 증가로 인해 트래픽 전송 성능이 저하된다. DBAC는 큰 BDP 경로에서 CUBIC처럼 작동하며, 경로 자원을 효율적으로 활용한다. 작은 BDP를 가진 일반 경로에서는 LIA와 같은 방식으로 혼잡을 제어하도록 설계되어 MPTCP 자체의 전송 성능 공정성을 유지한다. MPTCP 공정성 목표를 충족하는 Cwnd의 적절한 결합을 위해, 먼저 CUBIC Cwnd를 RENO Cwnd로 정규화한 다음 이를 사용하여 증가 계수 α를 계산한다. 큰 BDP 경로에서는 동일한 병목 대역폭을 공유하는 단일 경로 CUBIC 흐름과의 공정성을 유지하기 위해 증가 계수 α를 채택하여 CUBIC Cwnd를 수정한다. BDP 크기가 혼합된 비대칭 이기종 네트워크 테스트베드를 구성하고, 리눅스 커널 구현 코드를 사용하여 LIA와 DBAC의 성능을 실험적으로 비교하였다. 실험 결과, 제안한 DBAC는 큰 BDP 경로에서 대역폭 자원의 80% 이상을 효과적으로 활용하는 반면, LIA는 동일한 경로에서 대역폭의 20% 미만을 사용하는 것으로 나타났다. DBAC를 사용함으로써 고속 다중 경로 네트워크에서 자원 활용과 트래픽 전송 성능이 크게 향상되었으며, 단일 경로 CUBIC 흐름과 경쟁할 때 MPTCP 공정성 목표도 유지되는 것으로 확인되었다.

References

[1] V. Jacobsen and M. J. Karels, “Congestion avoidance and control,” ACM CCR, Vol.18, No.4, pp.314-329, 1995.

[2] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, “The NewReno modification to TCP’s fast recovery algorithm,” RFC 6582, Internet Engineering Task Force, 2012.

[3] A. Ford, C. Raiciu, M. Handley, and O. Bonaventur, "TCP extensions for multipath operation with multiple addresses," RFC 6824, Internet Engineering Task Force, 2013.
[4] C. Raiciu and M. H. D. Wischik, "Coupled congestion control for multipath transport protocols," RFC 6356, Internet Engineering Task Force, 2011.
[5] R. Khalili, N. Gast, and M. Popovic, "Opportunistic linked-increases congestion control algorithm for MPTCP," IETF, 2014.
[6] I. Rhee et al., "CUBIC for fast long-distance networks," RFC 8312, Internet Engineering Task Force, 2018.
[7] N. R. Thakur and A. S. Kunte, "Smart congestion control and path scheduling in MPTCP," In: Choudrie, J., Mahalle, P., Perumal, T., Joshi, A. (eds) IOT with Smart Systems. Smart Innovation, Systems and Technologies, Vol.312. Springer, Singapore, 2023.
[8] L. Chao, C. Wu, T. Yoshinaga, W. Bao, and Y. Ji. "A brief review of multipath TCP for vehicular networks," *Sensors*, Vol.21, No.8, pp.2793, 2021.
[9] M. Aljubayri, T. Peng, and M. Shikh-Bahaei, "Reduce delay of multipath TCP in IoT networks," *Wireless Netw*, Vol.27, pp.4189-4198, 2021.
[10] M. Chen, M. W. Raza, X. Zhou, T. Dreibholz, and Y. Tan "A multi-parameter comprehensive optimized algorithm for MPTCP networks," *Electronics*, Vol.10, No.16, pp.1942, 2021.

한 기 문

https://orcid.org/0009-0000-3810-0945
e-mail : kmhan14@add.re.kr
2016년 충북대학교 정보통신공학부(학사)
2019년 충남대학교 전자전파정보통신공학과 컴퓨터네트워크(석사)
2019년 ~ 현 재 국방과학연구소 연구원
관심분야 : 네트워크 성능분석, 위성통신