Journal of The Institute of Electronics and Information Engineers Vol.52, NO.7, July 2015 http://dx.doi.org/10.5573/ieie.2015.52.7.063
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2015-52-7-8

UDT 플로우 간 공평성 향상을 위한 혼잡도 기반의 가용대역폭 추정 기법

(Congestion Degree Based Available Bandwidth Estimation Method for Enhancement of UDT Fairness)

박 종 선*, 장 현 희**, 조 기 환***

(Jongseon Park, Hyunhee Jang, and Gihwan Cho©)

요 약

End to end 데이터 전송프로토콜에서 가용대역폭을 정확하게 추정하는 것은 매우 중요하다. UDT (UDP based Data Transfer)는 송신자에서 주기적으로 보낸 패킷 쌍을 이용하여 수신자에서 현재 링크의 최대 전송 가능한 속도 (MTR: Maximum Transfer Rate)를 추정하고 송신자에서 EWMA 알고리즘을 통해 MTR을 결정한다. 가용대역폭의 크기는 MTR과 현재 전송속도 차이로 구하기 때문에 정확한 가용대역폭 추정을 위해서는 정확한 MTR 추정이 우선시된다. 하지만 트래픽 영향으로 인한 혼잡상황에서 다수의 UDT 플로우가 경쟁하는 경우 심각한 공평성 문제를 초래한다. 본 논문에서는 혼잡도를 기반으로 한 MTR 추정 알고리즘을 제안한다. 혼잡도는 병목구간에서 트래픽 영향으로 인한 혼잡상태를 나타내는 상대적인 지수이며 패킷 쌍들의 도착간격을 이용하여 구할 수 있다. 그리고 혼잡도에 따라 EWMA 알고리즘을 세분화함으로써 실제에 근접한 가용대역폭 추정이 가능하다. Ns-2 시뮬레이션 실험 결과 제안기법은 다수의 경쟁플로우가 동일 링크를 점유하는 상황에서 데이터 전송거리에 따라 UDT 보다 확연하게 높은 공평성을 보이는 것을 확인하였다.

Abstract

In the end to end data transfer protocols, it is very important to correctly estimate available bandwidth. In UDT (UDP based Data Transfer), receiver estimates the MTR (Maximum Transfer Rate) of the current link using pair packets transmitted periodically from sender and, then sender finally decides the MTR through EWMA (Exponential Weighted Moving Average) algorithm. Here, MTR has to be exactly estimated because available bandwidth is calculated with difference of MTR and current transfer rate. However, when network is congested due to traffic load and where competing flows are coexisted, it bring about a severe fairness problem. This paper proposes a congestion degree based MTR estimation algorithm. Here, the congestion degree stands a relative index for current congestion status on bottleneck link, which is calculated with arriving intervals of a pair packets. The algorithm try to more classify depending on the congestion degree to estimate more actual available bandwidth. With the network simulation results, our proposed method showed that the fairness problem among the competing flows is significantly resolved in comparison with that of UDT.

Keywords : End to end data transfer protocol, Available bandwidth, UDT, Pair packet, Fairness.

* 학생회원, 전북대학교 전자정보공학부(Division of Computer Science and Engineering, Chonbuk University)
** 정회원, 한국폴리텍대학(Korea Polytechnic College)
*** 정회원, 전북대학교 컴퓨터공학부(Division of Computer Science and Engineering, Chonbuk University)
© Corresponding Author(E-mail: ghcho@chonbuk.ac.kr)
※ “이 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2014065816)”
Received ; April 20, 2015 Revised ; June 13, 2015 Accepted ; June 25, 2015

Ⅰ. 서 론

광 네트워크 기술의 발달로 인해 고 대역폭을 지원하는 네트워크 장비와 네트워킹 기술이 크게 발전하고 있다. 최근에는 엔터테인먼트 콘텐츠 제작, 관리 및 제공에 이용되는 디지털 미디어부터 과학용 연구에 사용되는 데이터가 급격하게 증가하면서 글로벌 규모로 대량 데이터를 신속하고 안정적으로 전송하는 것이 큰 이슈가 되고 있다[1].

TCP (Transmission Control Protocol)는 사실 상 표준으로써 대부분 응용을 위한 전송프로토콜로써 사용되고 있다. 하지만 TCP Reno와 TCP NewReno와 같은 전통적인 TCP는 LFN (Long Fat pipe Network)에서 제공되는 가용대역폭을 충분히 활용하지 못하기 때문에 낮은 전송률을 보인다[2~4]. 이러한 이유로 LFN에서 가용대역폭을 보다 신속하게 점유하기 위한 고속전송프로토콜에 대한 많은 연구가 진행되어 오고 있다. 이는, TCP의 경우 혼잡 윈도우의 크기를 크게 설정하는 방법이거나 응용계층에서 별도의 혼잡제어 기능구현을 통해 UDP (User Datagram Protocol)를 이용하는 방법이 될 수 있다[5~7]. 하지만 이러한 고속전송프로토콜은 같은 네트워크 대역폭을 공유하는 경우 경쟁플로우에 영향을 미쳐 공평성 문제를 야기한다. 따라서 높은 전송률과 함께 다른 경쟁플로우에 영향을 최소화함으로써 공평성을 높일 수 있는 기법이 추가적으로 필요하다.

전송프로토콜과 경쟁플로우들의 관점에서 TCP Reno의 공평성과 전송률은 HSTCP (High Speed TCP)나 STCP (Scalable TCP)와 같은 고속전송프로토콜이 같은 링크를 공유할 경우 낮게 측정된다[8]. 이는 TCP Reno를 비롯한 HSTCP나 STCP가 패킷 손실을 네트워크 혼잡으로 간주하고 패킷 손실 발생 시까지 계속 전송률을 증가시키기 때문이다. 한편, TCP Westwood나 FTCP (Fast TCP)가 TCP Reno와 공존할 경우에는 TCP Reno가 상대적으로 높은 공평성과 전송률을 보인다. 이는 TCP Westwood나 FTCP가 ACK (Acknowledgement) 패킷의 도착간격을 이용하여 네트워크 상태를 예측하고 현재 가용대역폭을 계산하기 때문이다. 하지만 수신자에서 응답으로 보낸 ACK 패킷이 송신자로 되돌아오는 과정에서 트래픽들의 영향을 받기 때문에 정확한 가용대역폭 추정이 어렵다.

UDT (UDP based Data Transfer)는 UDP 기반 전송프로토콜로써 응용계층에서 별도의 혼잡제어 기능을 추가적으로 구현하여 TCP 수준의 전송 신뢰성은 물론 Rate 제어를 통해 높은 전송률을 보인다[9]. Rate 제어는 수신자에서 패킷 쌍을 통해 추정한 가용대역폭을 기반으로 수행되며 가용대역폭이 클수록 매우 공격적으로 패킷을 전송한다. 송신자는 주기적으로 패킷 쌍을 보내는데 이 패킷 쌍이 수신자에 도착했을 때 벌어진 도착간격을 이용하여 링크의 최대 전송 가능한 속도 (MTR)을 추정한다. 이 값이 ACK와 함께 송신자로 전송되면 송신자는 EWMA (Exponentially Weighted Moving Average) 알고리즘을 통해 MTR을 갱신한다. MTR과 현재 전송속도의 차이를 이용하면 가용대역폭의 크기를 구할 수 있다. 하지만 병목구간에서 트래픽 영향으로 인한 혼잡 발생 시 실제 보다 높은 가용대역폭이 추정되어 경쟁플로우 간의 공평성 문제가 발생한다. 이를 해결하기 위해서는 정확한 가용대역폭이 추정되어야 한다. UDT의 경우 정확한 가용대역폭 추정을

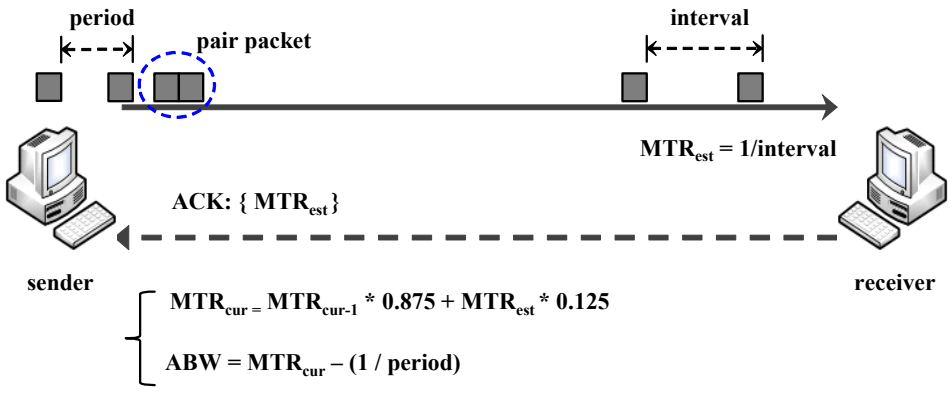


그림 1. UDT의 패킷 쌍을 이용한 MTR 추정 및 가용대역폭 추정
Fig. 1. MTR estimation and available bandwidth estimation with a pair packet of UDT.

위해서는 정확한 MTR의 추정이 우선시된다.

본 논문에서는 혼잡도 기반의 가용대역폭 추정 기법에 대해 제안한다. 혼잡도는 현재 링크의 혼잡상태가 어느 정도인지를 나타내는 지수로 표현되며 패킷 쌍들의 도착간격을 이용하여 구할 수 있다. 혼잡도에 따라 MTR 추정을 위한 EWMA 알고리즘이 세분화된다. 제안기법은 네트워크 상태를 충분히 반영함으로써 보다 실제에 근접한 가용대역폭 추정이 가능할 것으로 판단된다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 UDT 전송 프로토콜의 가용대역폭 추정 및 Rate 제어에 대해 살펴보고 Ⅲ장에서 패킷 쌍을 이용한 혼잡도 기반의 가용대역폭 추정 기법에 대해 기술한다. Ⅳ장에서 네트워크 시뮬레이터를 통한 성능을 측정, 분석하고 Ⅴ장에서 본 논문의 결론을 맺는다.

Ⅱ. UDP 기반의 고속전송프로토콜

UDT는 UDP 기반의 전송프로토콜로써 LFN 환경에서 TCP의 대안으로 Gu[10]에 의해 제안되었다. TCP 기반의 전송프로토콜들이 AIMD (Additive Increase Multiplicative Decrease) 전송메커니즘으로 인해 LFN에서 전송률이 제한되는 반면 UDT는 차별화된 D-AIMD(Decreased-AIMD) 전송메커니즘을 통해 높은 전송률을 보장한다. 이번 장에서는 UDT 가용대역폭 추정 및 Rate 제어에 대해 살펴보고 분석한다.

1. 가용대역폭 추정 및 Rate 제어

그림 1은 UDT의 패킷 쌍을 통한 가용대역폭 추정과 Rate 제어를 간략하게 도식하여 나타낸다. 송신자와 수신자는 독립적인 타이머 (Sync-time: 매 10ms마다 동작)를 기반으로 송신자는 Rate 제어를 수행하고 수신자는 현재 링크의 최대 전송 가능한 속도, MTR을 추정한다. 송신자는 16번째 패킷마다 연속적인 두 개의 패킷 쌍을 전송하며 수신자는 패킷 쌍의 도착간격을 이용하여 MTR을 추정할 수 있다. 추정된 MTR은 Sync-time마다 Control 채널을 통해 ACK 패킷에 포함되어 송신자로 전송되고 송신자는 EWMA 알고리즘을 통해 MTR을 갱신한다. EWMA를 통해 MTR을 구하면 가용대역폭 (ABW: Available Bandwidth)은 식 1과 같이 MTR과 현재 전송속도의 차이로 구할 수 있다. 현재 전송속도 ($Rate_{cur}$)는 그림 1의 패킷 간의 전송주기 (period)의 역수로 구할 수 있다.

$$\mathrm{ABW} = \mathrm{MTR_{cur}} - \mathrm{Rate_{cur}} \quad (1)$$

Rate 제어는 다음 Sync-time 동안 전송할 패킷들의 전송주기 (Period)를 결정하는 것을 의미한다. 추정한 가용대역폭이 큰 경우 주기는 짧아지고 반대의 경우 주기는 늘어난다. 주기가 짧다는 것은 Sync-time 동안 상대적으로 더 많은 패킷을 전송할 수 있음을 의미한다. 가용대역폭에 따른 전송주기를 구하기 위해 식 2와 같이 먼저 증가될 패킷의 양을 계산한다.

$$inc = \max(10^{\lceil \log_{10} B \rceil - 9}, \frac{1}{1500}) \times \frac{1500}{MSS} \quad (2)$$

식 2에서 B는 가용대역폭, MSS는 최대 세그먼트 크기를 나타낸다. 식과 같이 증가할 패킷의 양은 가용대역폭의 크기에 비례함을 알 수 있다. 식 2에 의해 패킷 간의 전송주기는 식 3과 같이 계산된다. 식 3에서 P′는 이전 전송주기이며 전송주기 P는 inc의 값이 클수록 감소하는 것을 알 수 있다.

$$P = \frac{P' \times Synctime}{Synctime + P' \times inc} \quad (3)$$

패킷 손실이 발생한 경우의 전송주기 P는 식 4와 같이 이전 전송주기 P'를 1/8만큼 증가하여 전송할 패킷의 양을 줄인다.

$$P = P' \times 1.125 \quad (4)$$

2. 가용대역폭 추정 및 Rate 제어에 대한 분석

1절의 식을 통해 알 수 있듯이 UDT Rate 제어는 전적으로 가용대역폭 크기에 따라 결정됨을 알 수 있다. TCP에서 AIMD 전송메커니즘으로 인해 제한되는 전송률 개선을 위해 다양한 기법들이 복잡한 구조의 혼잡제어를 수행하는 것을 고려하면 UDT는 비교적 단순한 구조를 갖는다. 그럼에도 불구하고 UDT는 Rate 제어를 통해 TCP와 비교하여 LFN 환경에서 매우 높은 전송률을 보인다. TCP 대안으로 제시된 UDT 전송률 극대화 전략은 크게 다음과 같이 요약할 수 있다.

- 패킷 쌍을 통한 수신자 기반의 가용대역폭 추정: 수신자 기반의 가용대역폭 추정은 RTT (Round Trip Time)를 이용하는 것에 비해 트래픽의 영향

을 최소화함으로써 보다 정교한 추정이 가능하다.

- 독립적인 Sync-time: 송신자와 수신자가 독립적인 타이머를 기반으로 송신자는 수신자로부터 보낸 패킷에 대한 응답을 기다릴 필요 없이 연속적인 패킷의 전송이 가능하다.
- Rate 제어: 패킷의 전송주기를 조절함으로써 신속한 가용대역폭 확보가 가능하다. Rate 제어는 추정한 가용대역폭의 크기가 클수록 한꺼번에 대량의 패킷을 신속하게 전송할 수 있다.

하지만 UDT는 트랙픽 영향으로 인한 혼잡 시 실제보다 가용대역폭이 크게 추정된다. 이는 EWMA 알고리즘을 통한 가용대역폭 추정에서 찾을 수 있다. 식 5와 같이 EWMA를 통한 MTR 추정은 이전 MTR과 패킷 쌍을 통해 추정한 MTR의 가중치 합으로 구한다.

$$MTR_{cur}=MTR_{cur-1}\times 0.875+MTR_{est}\times 0.125 \quad (5)$$

식과 같이 고정적으로 현재 MTR에 높은 가중치를 두는 것을 알 수 있다. 이는 네트워크 혼잡이 발생했을 경우 패킷 쌍을 통해 추정한 MTR 보다 큰 값이 추정되고 결과적으로 가용대역폭이 크게 추정되는 원인이 된다. UDT는 가용대역폭의 크기에 따라 대량의 패킷을 전송하기 때문에 이 경우 패킷 손실률과 함께 경쟁플로우에 영향을 미치게 된다. 또한 UDT는 일정 Sync-time 마다 Rate 제어를 수행하기 때문에 현재 네트워크 상태를 충분히 반영할 수 없기 때문에 부정확한 가용대역폭 추정은 더욱 큰 문제를 야기할 수 있다. 이를 해결하기 위해서는 현재 네트워크 상태를 충분히 반영한 가용대역폭 추정이 필요하다.

Ⅲ. 제안 기법

이번 장에서는 트래픽 영향으로 인한 혼잡도 기반의 가용대역폭 추정 기법에 대해 기술한다. 혼잡도는 패킷 쌍들의 도착지연을 이용하여 구하고 혼잡도에 따라 적응적으로 EWMA 알고리즘을 변경한다. 제안기법은 네 혼잡 시 실제에 근접한 가용대역폭을 추정함으로써 플로우 간의 공평성을 크게 개선할 수 있을 것으로 판단된다.

1. 패킷 쌍 도착지연을 이용한 혼잡도 계산

End-to-end 데이터 전송 프로토콜의 경우 경쟁플로우들 간의 순간적인 영향으로 인해 병목구간의 혼잡상태를 예측하는 것은 매우 어려운 일이다. TCP Vegas의 경우에는 ACK의 RTT를 이용하여 네트워크 상태를 선점적으로 예측함으로써 플로우 간의 공평성을 크게 향상시킨다[11]. 송신자에서 패킷을 전송한 후 그에 대한 ACK를 수신하였을 때 RTT의 도착간격을 계산하면 병목구간의 지연상태를 판단할 수 있다. 이는 패킷 손실을 네트워크 혼잡으로 간주하는 혼잡제어 기법에 비해 손실률을 줄이고 플로우 간의 공평성을 크게 높인다[12]. 하지만 RTT를 이용하는 경우 ACK 패킷이 되돌아오는 과정에서 트래픽의 영향을 받아 정확도가 떨어지게 된다. 이 경우에는 RTT를 이용하는 것보다 OWD (One Way Delay)를 이용하는 것이 정확도를 높일 수 있다. OWD는 수신자에서 패킷의 도착 간격을 이용하여 링크 상태를 추정한다. UDT는 송신자에서 주기적으로 보낸 패킷 쌍이 수신자에 도착했을 때 도착지연을 측정하고 이를 이용하여 MTR을 추정한다. 따라서 이 값을 이용하여 현재 링크의 혼잡도를 계산할 수 있다.

송신자에서 보낸 패킷이 수신자에 도착 할 때까지의 시간은 병목구간에서의 혼잡상태를 반영한다. 패킷의 도착시간 T_{delay}는 식 6과 같은 요소를 포함하여 계산된다.

$$T_{delay} = T_{trans} + T_{proc} + T_{prop} + T_{que} \quad (6)$$

T_{trans}는 전송 지연, T_{proc}은 프로세스 지연, T_{prop}는 전파 지연 그리고 T_{que}는 병목구간 라우터의 큐 지연을 의미한다. 일반적으로 큐에서의 지연 시간을 혼잡상태 판별에 이용한다. 본 논문에서는 패킷 쌍을 도착간격을 이용하여 큐 지연이 어느 정도인지를 추정한다. 그림 1과 같이 송신자는 16번째 패킷 마다 연속적인 패킷 쌍을 보내며 수신자는 일정 Sync-time 마다 패킷 쌍들의 도착시간을 이용하여 MTR을 추정한다. MTR은 도착시간의 역수로 계산하며 ACK에 포함되어 다시 송신자로 전송된다.

본 논문에서는 수신자에서 패킷 쌍을 통해 추정한 MTR이 ACK에 포함되어 송신자에 도착했을 때 이를 이용하여 네트워크 혼잡도를 계산한다. 혼잡도 (CD: Congestion Degree)는 0에서 1 사이의 값으로 계산되며 식 7과 같이 구할 수 있다.

$$CD = \frac{Delay_{cur} - Delay_{min}}{Delay_{max} - Delay_{min}} \quad (7)$$

식에서 $Delay_{cur}$는 ACK 수신 후 구한 현재 패킷 쌍의 지연을 $Delay_{max}$는 최대 지연, $Delay_{min}$은 최소 지연을 나타낸다. 최대, 최소 지연은 현재까지 ACK를 수신한 패킷 쌍들의 값을 임의의 크기의 큐에 저장하고 매 ACK가 도착할 때마다 비교하면서 구한다. 본 논문의 경우 큐의 크기를 500으로 설정하고 500개의 값들 중에서 최소와 최대 지연 값을 찾는 방법을 사용한다. 수신자로부터 ACK 수신 주기가 Sync-time (10ms)인 점을 고려하면 대략 5초 정도의 시간이며 이는 네트워크 혼잡도를 반영하기에 충분한 시간일 것으로 판단된다. 그림 2는 혼잡도 계산을 위한 알고리즘을 간략한 슈도코드로 나타낸다. $Delay_{min}$은 100000으로 크게 설정하고 $Delay_{max}$는 0으로 설정하여 연결 초기단계서부터 $Delay_{cur}$가 수신되었을 때 값들의 비교가 가능하도록 한다. 혼잡도가 1에 근접할수록 혼잡상태가 심각한 수준이며 반대로 0에 근접할수록 혼잡상태가 양호한 것으로 간주할 수 있다.

알고리즘 1: 혼잡도 계산 알고리즘

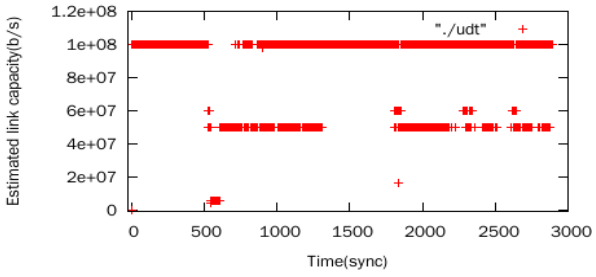
```
# 수신자로부터 ACK 수신 후 혼잡도 계산
  Delay_min = 100000;
  Delay_max = 0;

  processCtrl on ACK {

  # ACK 수신 후 Delay_curr push 및 pop
    delay_queue.push(Delay_curr);
    delay_queue.pop();

  # 기록된 지연 값들의 비교 후 최소, 최대 지연을 추출
    Delay_min = delay_queue.min();
    Delay_max = delay_queue.max();

  # 최소, 최대 지연 업데이트
    if(Delay_curr < Delay_min) Delay_min = Delay_curr;
    if(Delay_curr > Delay_max) Dlay_max = Delay_curr;

  # 혼잡도 계산
    CD = (Delay_curr - Delay_min) /
         (Delay_max - Delay_min);
  }
```

그림 2. ACK 수신 후 패킷 쌍 도착지연의 비교를 통한 혼잡도 계산 알고리즘

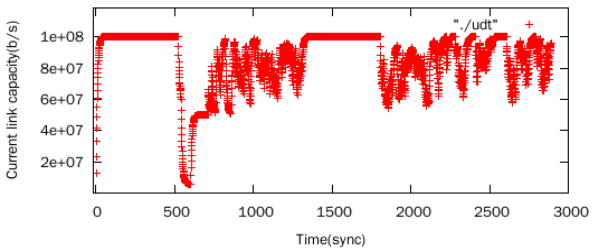
Fig. 2. Congestion degree calculation algorithm using pair packet delays when ACK arrives.

2. 혼잡도를 이용한 MTR 추정 기법

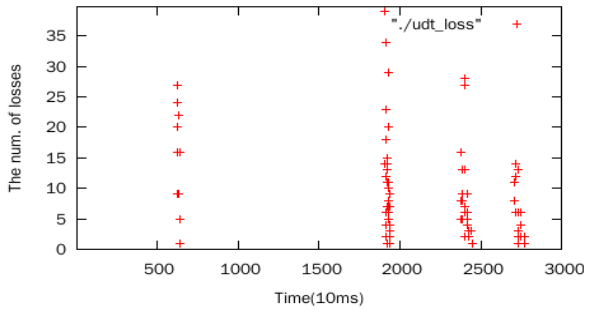
수신자로부터 패킷 쌍을 통해 추정한 MTR을 ACK를 통해 수신 후 송신자는 EWMA 알고리즘을 통해 현재 MTR을 갱신한다. 하지만 패킷 쌍을 통해 추정한 값을 EWMA 알고리즘을 이용하여 갱신할 경우 트래픽 영향으로 인한 혼잡 시 실제보다 높은 가용대역폭이 추정되는 문제가 발생한다. 높게 추정된 가용대역폭은 다음 Sync-time 동안의 전송 패킷의 양을 증가시키고 혼잡 시 패킷 손실과 공평성 문제의 원인이 된다. 그림 3



(a)



(b)



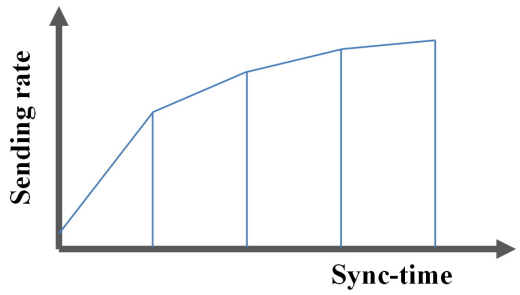
(c)

그림 3. (a) 수신자에서 패킷 쌍 도착지연을 이용하여 추정한 MTR (b) 송신자에서 EWMA 알고리즘을 이용하여 추정한 MTR (c) 10ms 단위로 측정한 패킷 손실의 수

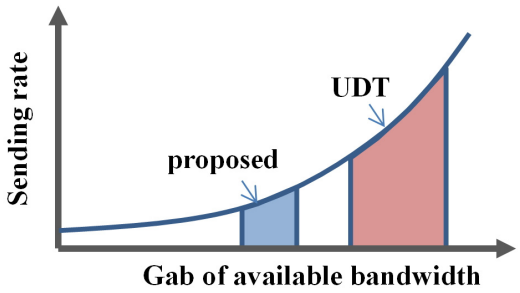
Fig. 3. (a) Estimated MTR using arriving interval of pair packet on receiver side (b) Estimated MTR using EWMA algorithm on sender side (c) The number of packet losses measured every 10ms.

은 UDT의 MTR 추정 결과를 확인하기 위해 네트워크 시뮬레이터를 이용하여 측정한 실험결과를 나타낸다. 실험은 네트워크 대역폭을 100Mb/s, 링크지연을 60ms 그리고 큐는 Drop-tail 방식으로 충분히 크게 설정하고 하나의 UDT 플로우를 이용하여 측정한 결과이다.

병목구간 혼잡 상태를 고려하여 UDT 전송 5초 후 주기적으로 50Mb/s UDP 트래픽을 전송하면서 실험하였다. 그림 3의 (a)는 수신자에서 패킷 쌍을 통해 추정한 MTR이며 (b)는 EWMA 알고리즘을 통해 추정한 MTR을 나타낸다. (a)의 경우 50Mb/s UDP 트래픽이 생성될 때마다 MTR이 50Mb/s로 추정되는 것을 확인할 수 있다. 하지만 EWMA를 통해 추정한 (b)의 경우 (a)의 경우 보다 상대적으로 큰 값이 추정되는 것을 알 수 있다. 이는 트래픽 영향으로 인한 혼잡 시 패킷 쌍을 통해 추정한 MTR의 크기가 감소하지만 EWMA를 통해 이전 값에 더 큰 가중치를 두기 때문이다. 그림 3의 (c)는 10ms 단위로 발생한 패킷 손실률을 측정하여 나타낸다.



(a)



(b)

그림 4. (a) Sync-time 마다 가용대역폭 점유를 위한 sending rate 변화량 (b) UDT와 제안기법의 가용대역폭 추정에 따른 sending rate 비교

Fig. 4. (a) Sending rate to occupy available bandwidth every sync-time (b) comparison of sending rate of UDT and proposed method based on available bandwidth estimation

그림 4의 (a)는 UDT의 D-AIMD 중 AI 즉, 가용대역폭점유를 위한 전송률 변화를 나타낸다. UDT는 가용대역폭의 크기에 따른 패킷 간의 전송 주기를 조절함으로써 신속한 가용대역폭 점유가 가능하며 그림 (a)와 같이 가용대역폭 확보를 위해 처음 전송률을 크게 하여 대량의 패킷을 전송하고 차츰 전송률을 줄여나가는 방법을 사용한다. 가용대역폭 확보를 위해 처음 패킷의 전송량을 크게 증가시키기 때문에 혼잡상태에서 다수의 경쟁플로우가 공존하는 경우 플로우 간의 공평성 문제를 야기하는 원인이 된다. 따라서 본 논문에서는 트래픽 영향으로 인한 혼잡 시 혼잡도를 반영하여 실제에 근접한 가용대역폭 추정을 위한 EWMA 알고리즘을 제안한다. 그림 4의 (b)는 UDT와 제안기법의 가용대역폭 추정과 전송률의 크기를 비교하여 나타낸다. 제안기법은 UDT가 가용대역폭이 클수록 전송률이 매우 공격적인 점을 고려하여 혼잡상태에 따라 가용대역폭의 변화를 최소화하는 것을 목표로 한다.

그림 5는 제안하는 EWMA 알고리즘을 나타낸다. MTR_{est}는 패킷 쌍을 통해 추정한 값이 ACK를 통해 송

알고리즘 2: MTR 추정을 위한 혼잡도 기반의 EWMA 알고리즘

패킷 쌍을 통해 추정한 MTR과 이전의 MTR을 비교 후 추정한 MTR이 이전 MTR 보다 작은 경우 혼잡도를 기반으로 서로 다른 가중치를 적용

$if(MTR_{cur-1} > MTR_{est})$

$if(CD < a)$

$MTR_{cur} = MTR_{cur-1} \times 0.875 + MTR_{est} \times 0.125$

$else\ if(a <= CD <= \beta)$

$MTR_{cur} = MTR_{cur-1} \times 0.125 + MTR_{est} \times 0.875$

$else$

$MTR_{cur} = MTR_{est}$

$else$

$MTR_{cur} = MTR_{cur-1} \times 0.875 + MTR_{est} \times 0.125$

그림 5. (a) MTR 추정을 위한 혼잡도 기반의 EWMA 알고리즘

Fig. 5. (a) Congestion degree based EWMA algorithm for MTR estimation.

신자로 도착 했을 때의 값이며 MTR_{cur-1}는 이전 값을 나타낸다. 제안기법은 먼저 이 두 값을 비교하고 MTR_{est}가 MTR_{cur-1} 보다 작은 경우에 혼잡도를 통한 MTR 추정을 수행한다. MTR_{est}이 MTR_{cur-1} 보다 작다는 것은 현재 링크 상태가 이전보다 혼잡해졌음을 의미한다. 제안기법의 경우처럼 이 두 값을 먼저 비교한 후 혼잡도를 기반으로 MTR을 추정하는 것은 보다 세분화되고 실제에 근접한 가용대역폭을 추정하기 위함이다.

혼잡도는 ACK가 도착할 때 마다 그 값들을 일정 기간 동안 저장하여 그 중에서 최소, 최대 값을 찾고 기록된 값들을 서로 비교하기 때문에 현재 링크의 혼잡정도가 어느 정도인지를 추정할 수 있다. 이는 기존 기법들이 링크의 혼잡을 혼잡과 혼잡이 아닌 상태로 양분하는 것에 비해 정확한 상태의 판별이 가능하다.

본 논문에서는 추정한 MTR이 이전 값보다 작은 경우에 혼잡도를 α, β 임계 값에 따라 다음과 같이 세 단계로 구분하여 EWMA 알고리즘의 가중치를 변경한다.

i) CD < α 인 경우,

혼잡도가 α보다 작은 경우에는 병목구간의 혼잡이 크지 않은 것으로 판별하여 기존 EWMA 알고리즘을 유지한다.

ii) α <= CD <= β 인 경우,

패킷 쌍을 통해 추정한 MTR에 가중치를 더 부여함으로써 곧 발생할 혼잡에 대비할 수 있다.

iii) CD > β 인 경우,

혼잡이 심각한 수준으로 패킷 쌍을 통해 추정한 MTR 즉, 더 작은 값의 MTR을 그대로 사용함으로써 가용대역폭 추정의 크기를 크게 줄일 수 있다.

추정한 MTR이 이전의 값 보다 큰 경우에는 기존과 같은 EWMA 알고리즘을 사용한다. 이와 같이 MTR을 서로 비교하고 MTR 추정을 위한 EWMA 알고리즘을 세분화함으로써 추정한 MTR이 이전보다 작은 경우에는 혼잡도에 따라 신속하게 가용대역폭 추정의 크기를 줄일 수 있다. 또한 추정한 MTR이 이전보다 큰 경우에는 급격한 전송률의 증가를 방지할 수 있다.

Ⅳ. 성능 평가

1. 실험 환경

제안기법의 성능 평가를 위해 네트워크 시뮬레이터

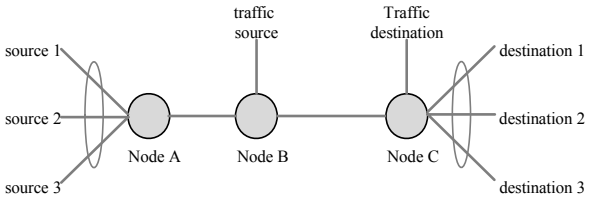


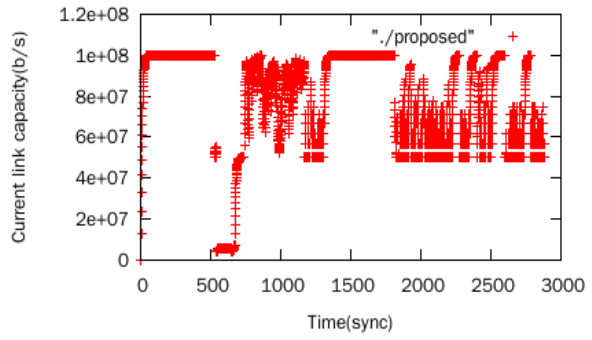
그림 6. 성능실험을 위한 시뮬레이션 환경
Fig. 6. Network topology for simulation.

인 Ns-2[13]를 이용하여 그림 6과 같은 시뮬레이션 환경을 구성하고 성능을 측정한다. 성능은 트래픽으로 인한 병목구간의 혼잡이 있는 상황에서 데이터 전송거리에 따른 제안기법의 Throughput을 측정하고 UDT와 비교한다. 또한 플로우 간의 공평성을 측정하기 위해 다수의 플로우가 공존하는 상황에서 각각의 Throughput을 측정하고 성능을 서로 비교한다. 실험을 위해 세 개의 경유 노드를 그림 6과 같이 연결하고 노드 A와 노드 B는 네트워크 대역폭을 100Mb/s, 링크지연을 10ms 그리고 큐는 Drop-tail 방식으로 충분히 크게 설정한다. 그리고 노드 B와 노드 C는 링크지연을 변경하면서 실험한다. 혼잡을 고려하여 노드 B와 노드 C 사이에 UDP 트래픽이 주기적으로 생성되도록 설정한다. 트래픽은 UDT 플로우 전송 후 5초 뒤 2초간 전송, 1초간 중지로 설정하여 실험한다.

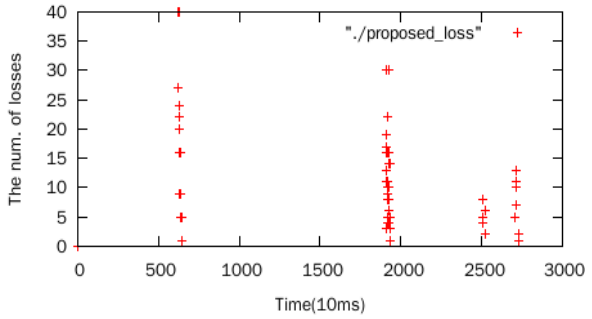
2. 제안기법을 통한 MTR 추정

성능을 측정하기에 앞서 제안기법의 MTR 추정 결과를 확인하기 위해 그림 3의 실험과 동일한 환경을 구성하고 그 결과를 UDT와 비교한다. 그림 7의 (a)는 EWMA 알고리즘을 통한 MTR 추정 결과를 나타낸다.

그림 3 (b)의 UDT 결과와 비교하여 제안기법인 그림 7 (a)의 추정 값이 혼잡 시 UDT 보다 낮게 추정된다. 실험에서 50Mb/s의 주기적인 트래픽을 생성한 것을 고려하면 제안기법이 UDT 보다 실제에 근접한 값이 추정되는 것을 확인할 수 있다. 이로 인한 손실률은 그림 7의 (b)와 같이 대략 20초 이후부터 UDT에 비해 낮은 손실률을 보인다. 특히 25초에서 30초 사이의 손실률은 UDT 보다 확연히 낮게 측정되는 것을 확인할 수 있다. 이는 트래픽 영향으로 인한 혼잡 시 실제에 근접한 가용대역폭을 추정함으로써 전송률을 감소시킨 결과로 판단된다.



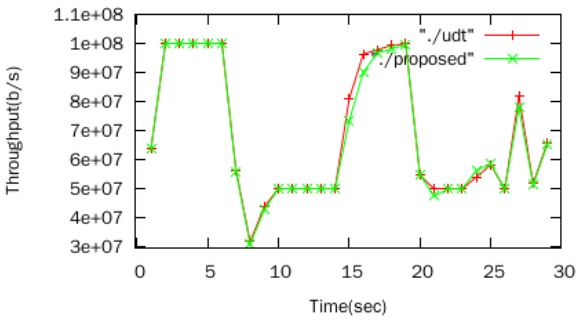
(a)



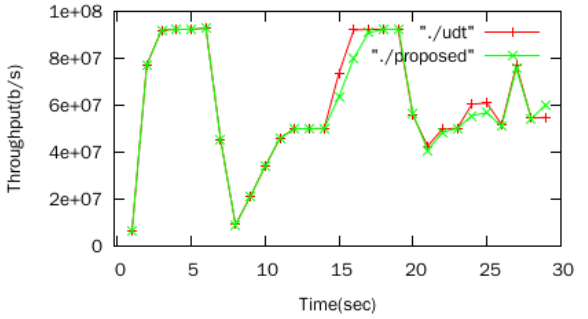
(b)

그림 7. (a) 제안기법의 EWMA 알고리즘을 통해 추정한 MTR (b) 10ms 단위로 측정한 패킷 손실률

Fig. 7. (a) Estimated MTR using proposed EWMA algorithm on sender side (b) The number of Packet losses measured every 10ms.



(a)



(b)

그림 8. (a) 링크지연이 10ms인 구간에서 측정한 Throughput의 성능 비교 (b) 링크지연이 50ms 구간에서 측정한 Throughput의 성능 비교

Fig. 8. (a) Comparison of throughput measured in 10ms of link delay (b) Comparison of throughput measured in 50ms of link delay.

3. 데이터 전송거리에 따른 성능 비교

그림 8은 링크지연 즉, 데이터 전송거리에 따른 제안기법의 Throughput을 UDT와 비교하여 나타낸다. 링크지연은 노드 B와 노드 C 사이를 10ms, 50ms로 설정하여 두 구간에서의 Throughput을 측정한다. 실험을 위해 노드 B와 노드 C 사이에 UDP 트래픽을 50Mb/s 속도로 주기적으로 전송하고 1초 단위로 Throughput을 계산한다.

네트워크 대역폭은 노드 A와 노드 B, 노드 B와 노드 C 모두 100Mb/s로 설정하고 큐는 Drop-tail 방식으로 충분히 크게 하여 설정한다. 그림 8의 (a)는 노드 B와 노드 C 사이의 링크지연을 10ms로 설정하여 측정한 실험결과이다. 제안기법과 UDT 모두 비슷한 결과를 보인다. 대략 15초에서 20초 사이 UDT에서 높은 성능이 측정되지만 매우 미미한 차이를 보인다. 이는 하나의 UDT 플로우가 링크를 독점하는 경우 가용대역폭을 충분히 활용할 수 있는 결과로 판단된다. 링크지연을 50ms로 늘려서 측정한 (b)의 경우에도 미미한 Throughput의 차이를 보이며 거의 동일한 결과의 성능을 보이는 것을 확인할 수 있다.

4. 플로우 수에 따른 공평성 성능 비교

UDT는 트래픽 영향으로 인한 혼잡 시 실제보다 높게 추정된 가용대역폭을 기반으로 매우 공격적인 Rate 제어를 수행하며 별도의 혼잡제어 기능이 없는 UDT는 플로우 간 공평성 문제가 발생한다. 이러한 공평성 문제는 다수의 UDT 플로우가 동일 링크를 점유하는 상황에서 더 심각하다.

그림 9는 제안기법과 UDT의 플로우 간 공평성을 비교하여 나타낸 그래프이다. 공평성 비교 실험은 UDT와 제안기법 모두 세 개의 플로우가 동시에 동일 링크를 점유하게 하고 각각의 Throughput을 측정하여 얼마나 대역폭을 공평하게 점유하는지를 비교한다. 실험은 네트워크 대역폭과 큐의 크기를 그림 8과 동일하게 설정

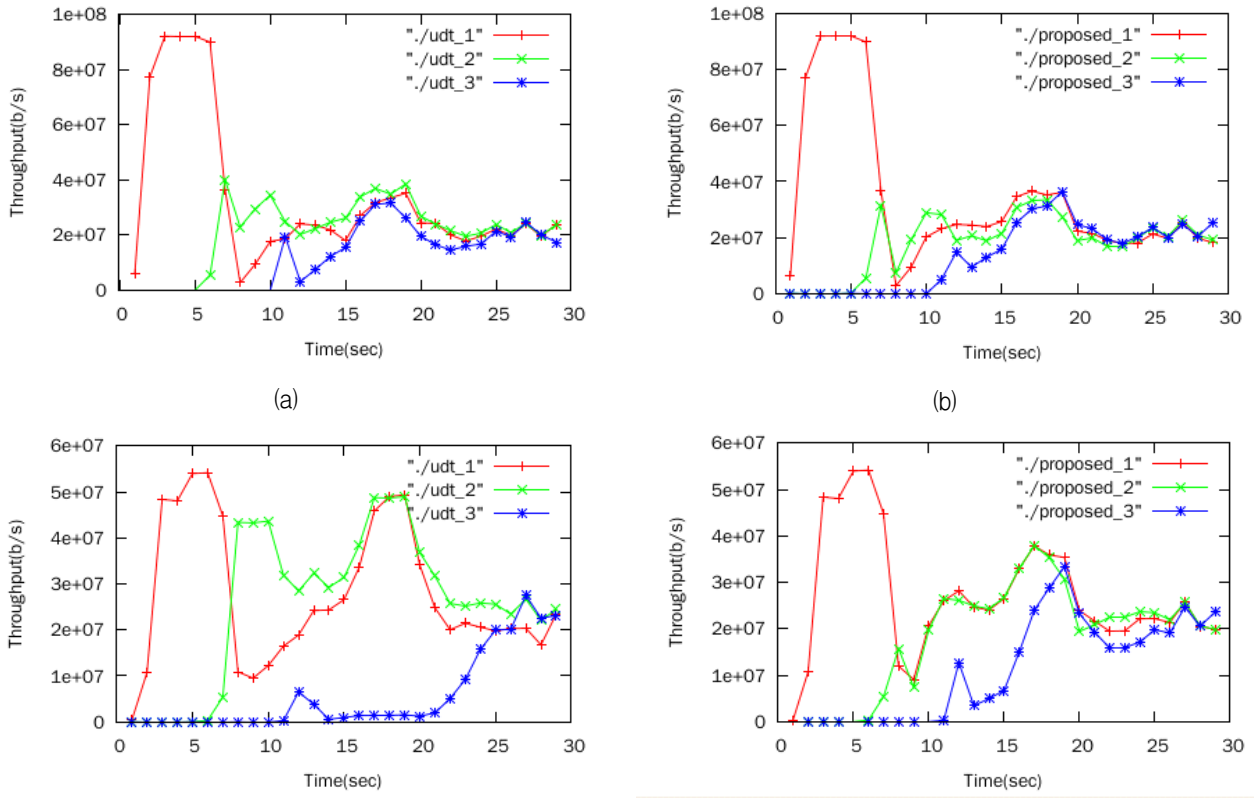


그림 9. (a) 링크지연이 50ms 구간에서 측정한 UDT의 throughput 및 플로우 간의 공평성 비교
(b) 링크지연이 50ms 구간에서 측정한 제안기법의 throughput 및 플로우 간의 공평성 비교
(c) 링크지연이 100ms 구간에서 측정한 UDT의 throughput 및 플로우 간의 공평성 비교
(d) 데이터 전송거리가 100ms 구간에서 측정한 제안기법의 Throughput 및 플로우 간의 공평성 비교

Fig. 9. (a) Comparison of UDT throughputs measured in 50ms of link delay and fairness
(b) Comparison of proposed method's throughputs measured in 50ms of link delay and fairness
(c) Comparison of UDT throughputs measured in 100ms of link delay and fairness
(d) Comparison of proposed method's throughputs measured in 100ms and fairness

하되 노드 B와 노드 C 사이의 링크지연을 각각 50ms와 100ms로 설정한 후 성능을 측정한다. 그리고 각각의 플로우는 첫 번째 플로우를 전송하고 5초의 간격으로 나머지 플로우들을 전송한다.

그림 9의 (a)와 (b)는 링크지연이 50ms인 구간에서 각각의 성능을 측정하고 그 결과를 비교하여 나타낸 그래프이다. UDT의 경우 경쟁플로우가 진입하는 순간 Throughput이 줄어들면서 시간에 따라 점차적으로 공평하게 네트워크 대역폭을 점유해 나간다. 하지만 (b)의 제안기법과 비교하여 낮은 공평성을 보이는 것을 확인할 수 있다. 특히, 두 번째 플로우가 진입했을 경우 UDT의 Throughput을 제안기법과 비교해보면 제안기법에서 보다 높은 공평성을 보이고 있다. 대략 18초 이후 구간에서 플로우 간의 공평성도 제안기법에서 높게 측정된다.

그림 9의 (c)와 (d)는 링크지연이 100ms인 구간에서의 공평성을 비교한 그래프이다. 이 경우 UDT와 제안기법의 공평성은 매우 확연한 차이를 보이고 있다. UDT의 경우 두 번째 플로우가 진입하는 순간 매우 심각한 공평성 문제가 발생하고 있다. 특히 세 번째 플로우가 진입한 경우 공평성은 매우 심각한 수준으로 떨어진다. 이에 비해 제안기법은 두 번째 플로우가 진입하는 순간 두 플로우 간의 공평성이 매우 높은 수준으로 한동안 유지되면 세 번째 플로우가 진입하는 순간에도 점증적으로 공평성이 개선되어 가는 것을 확인할 수 있다. 이는 제안기법의 경우 트래픽 영향이 있을 때마다

혼잡도를 측정하고 혼잡도에 따른 가용대역폭의 크기를 추정한 결과라 판단된다.

V. 결 론

UDT의 Rate 제어는 가용대역폭을 신속히 점유할 수 있지만 병목구간에서 트래픽 영향으로 인한 혼잡시 매우 공격적인 특성으로 인해 경쟁플로우 간 심각한 공평성 문제가 발생한다. 이는 Rate 제어를 수행하기 위한 가용대역폭이 상대적으로 높게 추정되는 문제에서 기인한다. UDT 특성 상 정확한 가용대역폭 추정을 위해서는 혼잡이 발생했을 때 패킷 쌍을 통한 보다 정확하고 실제에 근접한 MTR 추정이 우선시되어야 한다.

본 논문에서는 패킷 쌍의 도착지연을 이용하여 혼잡도를 계산하고 혼잡도를 기반으로 한 MTR 추정기법에 대해 제안하였다. 혼잡도는 송신자에서 보낸 패킷 쌍이 수신자에 도착했을 때의 도착간격을 이용하여 현재 링크의 혼잡상태를 추정하였다. 그리고 혼잡도를 기반으로 EWMA 알고리즘을 세분화하고 각 구간에 따른 가중치를 서로 다르게 적용함으로써 MTR을 추정하였다. 제안기법의 패킷 쌍을 이용한 혼잡도는 RTT를 이용하는 경우와 비교하여 트래픽의 영향을 최소화하고 정확하게 혼잡상태를 추정할 수 있는 이점이 있다. 또한 혼잡도에 따라 MTR 추정을 달리함으로써 혼잡 시 신속하게 가용대역폭의 크기를 줄이고 안정적으로 전송률을 조절할 수 있을 것으로 판단된다.

네트워크 시뮬레이터를 통해 실험한 결과 제안기법은 트래픽이 있는 구간에서 UDT와 비슷한 Throughput이 측정되는 것을 확인하였다. 하지만 다수의 플로우가 공존하는 상황에서는 UDT에 비해 높은 공평성을 보였다. 공평성은 데이터 전송거리가 증가함에 따라 확연하게 높아지는 것을 실험을 통해 확인할 수 있었다. 특히, 링크지연이 100ms인 구간에서는 UDT와 비교하여 117%의 향상된 공평성을 보였다.

REFERENCES

[1] http://asperasoft.com/ko/site/
[2] J. A. Arokkiam, W. Xiuchao, K. N. Brown and C. J. Ireland, “Experimental Evaluation of TCP Performance over 10Gb/s Passive Optical Networks (XG-PON).” in Proc. of GLOBECOM, pp. 2223-2228, Dec. 2014.
[3] M. A. Alrshah, and M. Othman, “Performance Evaluation of Parallel TCP, and its Impact on Bandwidth Utilization and Fairness in High-BDP Networks Based on Test-bed,” in Proc. of MICC, pp. 23-28, Nov. 2013.
[4] J. Wang, F. Gao, J. Wen, C. Li, Z. Xiong and Y. Han, “Achieving TCP Reno Friendliness in FAST TCP over Wide Area Networks,” in Proc. of ICNC, pp. 445-449, 2014.
[5] M. Meiss, “Tsunami: A High-Speed Rate-Controlled Protocol for File Transfer,” www.evl.uic.edu/eric/atp TSUNAMI.pdf, 2009.
[6] B. Eckart, X. He, Q. Wu and C. Xie, “A Dynamic Performance-Based Flow Control Method for High-Speed Data Transfer,“ IEEE Transaction on parallel and distributed systems, vol. 21, no. 1, Jan. 2010.
[7] J. Park, S. Kim, G. Hwang, and G. Cho, “A Design and Implementation of Bulk Data Transmission Tool based on UDT,” Journal of the Institute of Electronics and Information Engineers, TC 49(2), PP. 23-31, 2012.
[8] J. Masaki, G. G. D. Nishantha and Y. Hayashida, “Development of a High-Speed Transport Protocol with TCP-Reno Friendliness,” in Proc. of ICACT, pp. 174-179, Feb. 2010.
[9] Y. Gu and R. L. Grossman, “UDT: UDP-based Data Transfer for High Speed Wide Area Networks,” Computer Networks, vol. 51, no. 7, pp. 1777-1799, May 2007.
[10] Y. Gu, “UDT: A High Performance Data Transport Protocol,” Ph.D Thesis, Laboratory for Advanced Computing, Univ. of Illinois at Chicago, 2005.
[11] M. Podlesny, C. Williamson, “Providing Fairness between TCP NewReno and TCP Vegas with RD Network Services,” pp. 1-9, June 2010.
[12] H. Jung, S. Kim, H. Yeom, S. Kang, and L. Libman, “Adaptive Delay-Based Congestion Control for High Bandwidth-Delay Product Networks,” INFOCOM, pp. 2885-2893, Apr. 2011.
[13] http://www.isi.edu/nsnam/ns/

저 자 소 개

박 종 선(학생회원)
2009년 조선대학교 전자공학과 학사 졸업.
2012년 전북대학교 전자정보공학부 석사 졸업.
2012년~현재 전북대학교 전자정보공학부 박사 과정.
<주관심분야 : 대용량데이터전송, 센서네트워크, 무선네트워크보안>

조 기 환(정회원)
1985년 전남대학교 계산통계학과 학사 졸업.
1987년 서울대학교 계산통계학과 석사 졸업.
1996년 영국 Newcastle 대학교 전산학과 박사 졸업.
1987년~1997년 한국전자통신연구원 선임연구원.
1997년~1999 목포대학교 컴퓨터과학과 전임강사.
1999년~현재 전북대학교 컴퓨터공학부 교수.
<주관심분야 : 이동컴퓨팅, 컴퓨터통신, 분산처리시스템, 무선네트워크, 무선네트워크보안>

장 현 희(정회원)
1987년 원광대학교 전자계산공학과 학사 졸업
1998년 숭실대학교 정보과학대학원 컴퓨터네트워크 전공 석사 졸업
2006년 전북대학교 컴퓨터 정보학과 박사 수료
1991년~현재 한국폴리텍대학 김제캠퍼스 유비쿼터스시스템과 교수
<주관심분야 : 이동컴퓨팅, 정보통신, 무선네트워크>