UDT 환경에서 RTT 예측에 의한 Sync-Interval 구간의 Rate Control 기법

안도식*, 왕기철**, 조기환*
*전북대학교 전자정보공학부
**한국과학기술정보연구원 융합자원 연구실
e-mail : rokmcads@jbnu.ac.kr

A Rate Control Method for Sync-Interval Period based on RTT Estimation in the UDT Environment

Do-sik An*, Gi-cheol Wang**, Gi-hwan Cho*
*Div. of Computer Science and Engineering, Chonbuk National University
**Dept. of Computing and Networking Resources, KISTI

요 약

오늘날 대부분의 네트워크는 수십 Gb/s 를 지원하는 광네트워크다. 이러한 고속 네트워크에서 대부분 TCP 전송 프로토콜을 사용하고 있다. TCP 전송 프로토콜은 AIMD 방식의 특성으로 인해 고속 네트워크에 적합하지 않다. 반면 UDT 는 DAIMD 방식을 사용하기 때문에 고속 네트워크에서 가용대역폭을 충분히 활용 가능하다. 그러나 UDT 는 sync-interval 간격으로 rate control 이 이루어 지기 때문에 sync-interval 동안 가용 대역폭을 충분히 활용하지 못하는 문제점이 발생한다. 본 논문에서는 RTT 예측을 통한 sync-interval 구간에서의 rate control 기법을 제안한다. sync-interval 구간 동안 RTT 예측을 통해 rate control 을 함으로써 기존 UDT 에 비해 고속 네트워크 환경에서 보다 빠르게 가용대역폭을 활용할 수 있다. 네트워크 시뮬레이션 결과 기존 UDT 에 비해 throughput 및 안정성이 향상되었다.

1. 서론

오늘날 대부분의 국가간 연구망은 수 Gb/s 에서 수십 Gb/s 를 지원하는 광네트워크이다. 그러나 기존 TCP 메커니즘으로는 네트워크의 고속화된 성능을 충분히 활용하지 못한다[1][2].

TCP 전송 프로토콜에서 제공되는 AIMD(Additive increase multiplicative decrease)방식은 혼잡으로부터 빠른 회피와 데이터 전송의 신뢰성을 제공한다. 그러나 AIMD 방식은 빠른 가용대역폭 확보가 어렵고, 혼잡 윈도우의 빈번한 변화로 고속 네트워크의 이용 효율이 감소한다. 이러한 TCP 의 문제를 극복하기 위해 HSTCP[3], STCP[4], BIC TCP[5] 등 많은 프로토콜들이 연구되었다. 그러나 여전히 빠른 가용대역폭 확보에 어려움이 있다.

반면 UDT(UDP -based Data Transfer) 프로토콜은 DAIMD(Decrease Additive Increase Multiplicative Decrease) 방식을 사용한다[6]. 전송 시작 단계에서 증가인자 값을 빠르게 증가시키고, 시간이 지나면서 증가인자 값을 점차적으로 감소시킴으로써 고속네트워크 환경에서 빠른 가용대역폭 확보가 가능하다.

UDT 는 혼잡 회피를 위해 sync-interval 시간(0.01 초) 간격으로 rate control 과 window control 을 수행한다. rate control 은 패킷 sending period 를 조절하는 주요한 메커니즘이다. 한편, window control 은 unacknowleged 패킷의 수를 제한하는 dynamic threshold 를 명시하기 위해 사용되는 보완적인 메커니즘이다. 이러한 window control 은 혼잡 윈도우 크기와 현재의 버퍼사이즈 (UDT 의 플로우 계산은 수신 측에서 수행 됨) 사이의 최소값을 피드백 함으로서 간단한 flow control 을 포함한다.

ACK 패킷에 대한 rate control 은 sync-interval 간격으로 이루어 지기 때문에 0.01 초 동안 일정한 패킷 sending period 로 전송된다. sync-interval 시간 동안 네트워크 상태가 안정적일 경우 신속한 rate control 이 이루어 지지 않는 문제점이 있다. 본 논문에서는 ACK 패킷으로부터 수신된 RTT 값을 통해 네트워크 상태를 예측하여 sync-interval 동안 rate control 을 수행하는 기법을 제안한다. RTT 값의 변화량에 따라 RTT 를 예측하여 패킷 sync-interval 동안 sending period 를 가변적으로 조절한다. 제안 기법은 기존 UDT 에 비해 빠르게 네트워크 가용 대역폭을 활용할 수 있고, 혼잡이 발생 하였을 경우에 sync-interval 구간 동안 패킷 sending period 증가시킴으로써 혼잡을 빠르게 회피 한다.

본 논문은 다음과 같이 구성된다. 2 장에서 관련 연구를 소개하고, 3 장에서 RTT 예측을 통해 sync-interval 구간에서의 rate control 기법을 설명한다. 4 장에서는 네트워크 시뮬레이션 결과를 보여주며 이를 분석하고, 5 장에서 결론을 맺는다.

2. 관련 연구

윈도우 기반 컨트롤은 대량으로 데이터를 전송하면서 송신측이 링크가 혼잡하다는 것을 인식 할 때까지 많은 패킷을 전송하게 된다. 게다가 버스트 트래픽(burst traffic)은 라우터의 BDP(Bandwidth-Delay Product) 만큼 큰 용량의 버퍼를 요구하지만, 고속 네트워크와 같은 high BDP 인 경우에는 현실적이지 못하다.

이러한 문제를 해결하기 위해 TCP 의 경우 평균 interval(RTT/cwnd) 시간으로 혼잡 윈도우 안에서 패킷을 보내는 방법이 제안 되었다. 하지만, 패킷 sending period 를 정하기 위해서 TCP 의 혼잡 알고리즘을 사용하는 것은 때로는 throughput 을 감소시키거나 standard TCP 와 공존하는 경우에 정상적으로 동작하지 않는 문제점이 있다[7]. TCP 는 패킷 전송을 규정하기 위해 self-clocking 메커니즘을 사용 하는데 이는 역방향 트래픽이나 인터럽트 통합효과에 의해 throughput 이 감소될 수 있다.

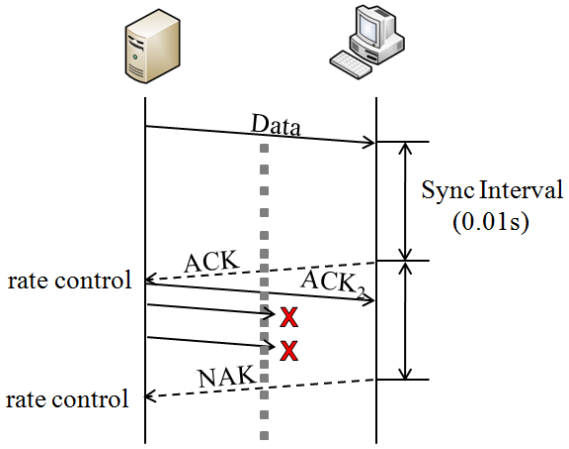
높은 BDP 를 가진 링크에서는, 효율적인 rate control 메커니즘으로 패킷 sending period 를 조절하는 것이 더 좋은 해결 방안이 될 수 있다. 하지만, rate control 은 연속적 손실 상황을 야기 할 수가 있다. 혼잡이 발생하고 NAK 패킷이 손실 되었을 때, 송신측은 loss report 나 timeout event 를 받기 전 계속적으로 데이터를 보내는 경우가 발생한다. 따라서 unacknowledged 패킷의 개수로 임계치를 명시하는 rate control 과 함께 보완적인 window control 이 사용 되어야 한다.

UDT 는 이 두 가지의 메커니즘을 조합한다. UDT 에서의 rate control 은 패킷 sending period 를 조절 한다. Window control 은 unacknowleged 패킷의 수를 제한하는 dynamic threshold 를 명시하기 위해 사용되는 보완적인 메커니즘이다. 이러한 window control 은 혼잡 윈도우 크기와 현재의 버퍼사이즈(UDT 의 플로우 계산은 수신 측에서 수행 됨) 사이의 최소값을 피드백 함으로서 flow control 을 한다. 혼잡 윈도우 크기는 패킷 AS(Arrival Speed) 와 sync-interval, RTT 의 가중치 곱과 합으로 계산된다. 모든 패킷에 대해 acknowledge 하는 프로토콜의 경우, 이동 할 수 있는 최대 패킷의 양은 전송 속도와 RTT 의 곱이다. 하지만, UDT 의 경우에는 매 sync-interval 시간마다 ACK 패킷이 전송되기 때문에 sending rate 와 (sync-interval + RTT)의 곱이 된다.

UDT 의 rate control 은 (그림 1) 에서와 같이 ACK 패킷을 수신한 경우와 NAK 패킷을 수신한 경우에 동작한다. 패킷 sending period 는 ACK 패킷을 수신하면 감소 시키고 NAK 패킷을 수신하면 증가 시킨다. ACK 패킷을 수신할 경우 식 (1)에 따라 *inc* 값을 계산한다.

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

식 (1)에서 B 는 가용 대역폭, MSS 는 최대 세그먼트 크기를 나타낸다. 식(1)을 통해 *inc* 값을 결정하고 다음 식 (2)를 통해 패킷 sending period 인 P 를 감소시킨다.



(그림 1) UDT Rate Control

$$P = \frac{P \times syncInterval}{P \times inc + syncInterval} \qquad (2)$$

NAK 패킷을 수신할 경우 P 는 식 (3)과 같이 1/8 만큼 증가 한다.

$$P = P \times 1.125 \qquad (3)$$

UDT 는 sync-interval 시간을 간격으로 rate control 을 하기 때문에 0.01 초 동안은 동일한 패킷 sending period 를 갖는다. 1Gb/s link capacity 의 연결에서 MTU 값을 1500 으로 가정 했을 때 sync-interval 동안 약 873 개의 패킷이 전송된다. 패킷 sending period 를 UDT 의 기본값인 1 μs로 가정 했을 때 sync-interval 동안 약 803 개의 패킷이 전송된다. 따라서 sync-interval 혼잡이 발생하지 않아도 일정한 주기로 패킷을 전송하기 때문에 가용 대역폭을 충분히 활용하지 못하는 문제점이 발생한다.

3. RTT 예측을 통한 sync-interval 구간에서의 rate control 기법

본 논문에서는 RTT 예측을 통해 네트워크 상태를 판단하여 sync-interval 구간동안 패킷 sending period 를 조절하는 rate control 기법을 제안한다. RTT 예측은 ACK 패킷으로부터 받은 RTT 값을 토대로 RTT_{t-2} 에서 RTT_t 사이의 차이에 가중치로 RTT_{t+1} 을 예측한다. 예측된 RTT_{t+1} 의 값과 RTT_t 값을 비교해 RTT 증감율 값인 R_{idr} 을 계산한다. R_{idr} 값이 1 이하가 되면 α 값의 구간에 따라 패킷 sending period 를 감소하고, R_{idr} 값이 1 이상이 될 경우 β 에 따라 패킷 sending period 를 증가 혹은 유지한다.

제안 기법의 동작과정은 다음과 같다.

ⅰ) RTT 예측

Sync-interval 구간 동안 rate control 을 하기 위해서는 먼저 RTT 를 예측 해야 한다. t+1 시간의 RTT 인 R_{t+1}

은 다음 식 (4)에 표현된 공식으로 예측한다.

$$R_{t+1} = \frac{\left((R_{t-1} - R_{t-2}) \times w_1 + (R_t - R_{t-1}) \times w_2\right)}{2} + R_t \quad (4)$$

식 (4)에서 w_i 는 RTT 차이 값의 가중치를 나타낸다. 가중치 값 w_1 과 w_2 의 합은 1이다. 가중치는 w_1 보다 w_2 를 더 크게 설정함으로써 최근 네트워크 상황을 반영한다. 네트워크 시뮬레이터로 가중치 값을 변경하여 실험한 결과 w_1 은 0.125, w_2 는 0.875로 설정하는 것이 가장 좋은 성능을 보였다.

ii) RTT 증가율 계산

예측된 R_{t+1} 값과 R_t 값을 비교하여 RTT 증감율 값인 R_{idr} 을 식 (5)와 같이 계산한다.

$$R_{idr} = \frac{R_{t+1}}{R_t} \quad (5)$$

네트워크 시뮬레이터로 각 RTT별 R_{idr} 값의 분포도를 실험한 결과 0.998 ~ 1.0 구간이 가장 많았다.

iii) R_{idr} 값에 따른 패킷 sending period 증감값 결정

계산된 R_{idr} 에 따라 α, 1, β 구간별로 패킷 sending period 증감값인 *PktSPW* 을 결정 한다.

$$PktSPW = \begin{cases} 0.99 & (R_{idr} \le \alpha) \\ 0.999 & (\alpha < R_{idr} \le 1) \\ 1.0 & (1 < R_{idr} < \beta) \\ 1.001 & (R_{idr} > \beta) \end{cases}$$

R_{idr} 값이1.0 이하일 경우에는 RTT가 감소하는 것으로 예측한 것이기 때문에 패킷 sending period를 α 값에 따라 감소시킨다. 만약 α 값이 1.0 에 근접할 경우 *PktSPW* 값이 0.99가 될 가능성이 높아진다. *PktSPW* 값이 0.99가 되는 경우가 빈번히 발생하게 되면 sync-interval 구간 동안 혼잡이 발생할 가능성이 높아 진다. R_{idr} 값이1.0 이상일 경우에는 RTT가 증가하는 것으로 예측한 것이기 때문에 패킷 sending period 를 β 이하에서는 유지하고 β 이상에서는 1.001만큼 증가시킨다. β 값이 1.0 에 근접하게 되면 sync-interval 동안 패킷 sending period 증가함에 따라 성능이 저하될 수 있다. 따라서 최적의 α, β 값을 설정해야 한다. α, β 값의 설정은 4장 성능 평가 단계에서 다루기로 한다.

iv) sync-interval 구간에서의 rate control

PktSPW 값은 ACK 패킷을 수신하면 RTT에 따라 다시 계산된다. ACK 패킷을 수신하면 기존 UDT의 rate control기법에 따라 패킷 sending period를 계산한다. sync-interval 구간에서는 ACK를 수신한 후에 계산된 패킷 sending period 값에 *PktSPW* 을 곱해서 rate control 을 한다. sync-interval 구간에서 rate control은 매 16개 패킷마다 이루어 진다. 각 패킷 마다 rate control 을 수행할 경우 패킷 sending period가 급격하게 증가하여 혼잡이 발생할 가능성이 높아진다. 또한 각 패킷 마다 sending period가 달라질 수 있기 때문에 수신측에서 bandwidth 측정이 어려워 진다.

Algorithm

```
#ACK packet received by receiver
R_{t+1} = ((R_{t-1} – R_{t-2}) * w_1 + (R_t-R_{t-1})*w_2)/2 + R_t;
R_idr = R_{t+1} / R_t;

if ( R_idr  ≤  α)
      PktSPW = 0.99;
else if(R_idr > α && R_idr  ≤ 1.0)
      PktSPW = 0.999;
else if(R_idr  ≥ 1.0 && R_idr <  β)
      PktSPW = 1.0;
else
      PktSPW = 1.001;

# Rate control in sync-interval
If (0 == next_sequence_number % 16)
      PktSP = PkrSP * PktSPW;
```

(그림 2) RTT 예측을 통한 sync-interval 구간에서의 rate control 알고리즘

(그림 2)는 RTT 예측을 통해 sync-interval 구간에서의 rate control 알고리즘이다. *PktSP* 는 패킷 sending period 를 나타낸다.

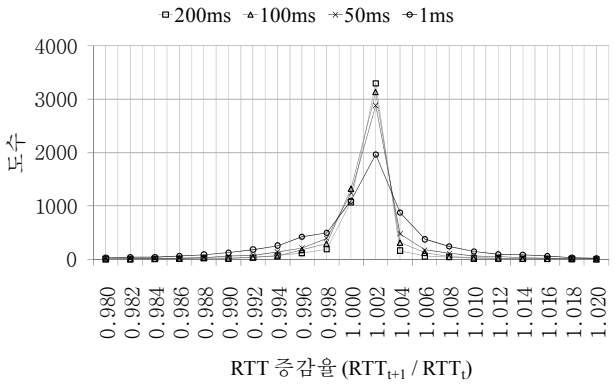
4. 성능 평가

본 논문에서 제안한 sync-interval 구간에서의 rate control 기법을 네트워크 시뮬레이터인 NS-2 를 이용하여 성능 실험을 하였다[8]. 실험에 사용한 파라미터는 다음 <표 1> 과 같다.

<표 1> 시뮬레이션 파라미터

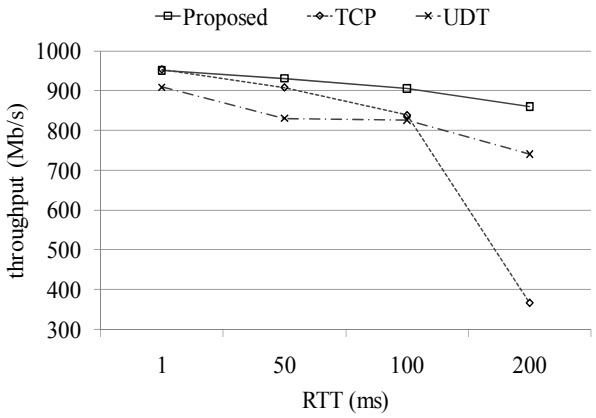
파라미터	값
Link Capacity	1 Gb/s
MTU	1500 byte
RTT	1ms, 50ms, 100ms 200ms
시뮬레이션 시간	60 초

(그림 3)은 각 RTT 별로 R_{idr} 값의 도수분포 그래프이다. 0.998 ~ 1.0 구간이 가장 많은 분포를 보였으며 그 다음이 1.0~1.02 구간 이다. 많은 분포를 차지하고 있는 구간의 RTT 의 변화량이 작고 네트워크가 안정화 되어 있는 구간이기 때문에 rate control 을 적용하게 되면 오히려 성능이 떨어진다. 따라서 RTT 의 변화량이 크고 분포가 많은 구간의 값을 정하는 것이 바람직하다. 이후 실험에서 α 값은 0.996, β 값은 1.004 로 설정 하였다.

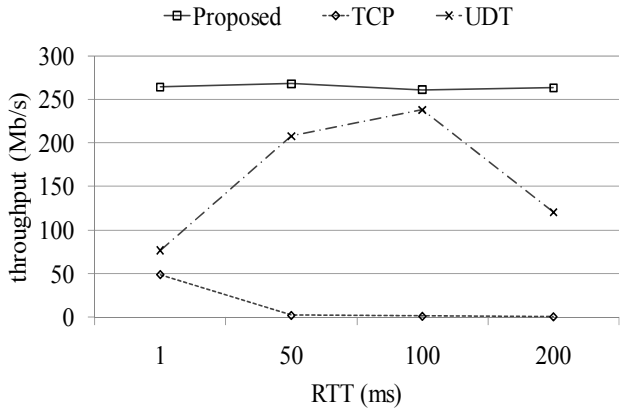


(그림 3) RTT 증감율 값의 구간별 분포

(그림 4)는 제안 기법을 TCP, UDT 와 성능 비교실험 그래프다. RTT 가 낮은 경우에 제안 기법과 다른 프로토콜 간의 큰 성능차이를 보이지 않지만 RTT 가 클 경우 큰 폭의 성능 차이를 보인다. 특히 RTT 200ms 환경에서 TCP 에 비해 약 130%, UDT 에 비해 약 14%의 성능 향상을 보인다.



(그림 4) 제안 기법의 성능비교



(그림 5) 1% 패킷 loss 상황에서의 성능비교

(그림 5)는 패킷 loss 를 1%로 설정하였을 때 TCP, UDT 와 성능 비교실험 그래프다. TCP 는 UDT 와 제안 기법에 비해 확연한 성능 차이를 보이고 있다. 또한 UDT 는 RTT 별로 성능 차이가 큰 반면 제안 기법은 RTT 가 증가하더라도 안정적인 성능을 보이고 있다. 제안 기법은 RTT 를 예측하여 sync-Interval 구간 동안 rate control 을 하기 때문에 네트워크 상태를 빠르게 반영한다.

5. 결론

본 논문에서는 RTT 예측을 통해 sync-interval 구간에서의 rate control 기법을 제안한다. 제안 기법은 크게 4 단계로 RTT 예측, 증감율 계산, 패킷 sending period 증감값 결정, sync-interval 구간에서의 rate control 로 구성된다. RTT 예측을 통해 현재 네트워크 상태에 따라 sync-interval 구간동안 rate control 을 함으로써 고속 네트워크 환경에서 가용 대역폭을 빠르게 확보 할 수 있다. 또한 네트워크 상태가 급격히 나빠지는 경우 패킷 sending period 를 증가시킴으로써 혼잡을 방지 한다.

시뮬레이션 결과 제안 기법이 200ms RTT 에서 UDT 에 비해 14% 성능 향상을 보였으며, 패킷 loss 가 발생했을 경우 다른 프로토콜에 비해 안정적인 성능을 보였다. 향후 연구 과제로 여러 flow 들에 대한 공정성에 대한 비교연구를 통해 실제 네트워크에 적용하는 연구를 하고자 한다.

참고문헌

[1] A. Chien, T. Faber, A. Falk, J. Bannister, R. Grossman, and J. Leigh, “Transport protocols for high performance: Whither TCP,” *Communications of the ACM*, 46(11), pp. 42-49, 2003

[2] W. Feng and P. Tinnakornsrisuphap, “The failure of TCP in high-performance computational grids,” *IEEE conference on Supercomputing '00*, pp. 4 - 10, 2000

[3] S. Floyd, “HighSpeed TCP for large congestion windows,” *IETF*, *RFC 3649*, Experimental Standard, 2003

[4] T. Kelly, “Scalable TCP: Improving performance in highspeed wide area networks,” *ACM Computer Communication Review*, *32(2)*, pp. 83-91, 2003

[5] L. Xu, K. Harfoush, and I. Rhee, “Binary increase congestion control for fast long-distance networks,” *IEEE Infocom '04*, pp. 2514-2524, 2004

[6] Y. Gu and R. L. Grossman, “UDT: UDP-based data transfer for high-speed wide area networks,” *Computer Networks*, *51(7)*, pp. 1777-1799, 2007

[7] A. Aggarwal, S. Savage, and T. Anderson, “ Understanding the performance of TCP pacing,” *IEEE Infocom '00*, pp 26-30, 2000

[8] Network Simulator, ns-2, http://www.isi.edu/nsnam/ns