AR UDT 환경에서 네트워크 상태에 적응적인 혼잡제어 기법

안도식*, 조기환**
전북대학교 *전자정보공학부, **컴퓨터공학부
e-mail : {*rokmcads, **ghcho}@jbnu.ac.kr

An Adaptive Congestion Control Method on Network Condition in the AR UDT Environment

Do-Sik An*, Gi-Hwan Cho**
*Div. of Electronics and Information Engineering, Chonbuk National University
**Div. of Computer Science and Engineering, Chonbuk National University

요 약

고속 네트워크 환경에서 AR UDT(Adaptive Rate control UDT)는 표준 전송 프로토콜인 TCP에 비해 뛰어난 성능을 보인다. UDT(UDP-based Data Transfer)를 기반으로 하는 AR UDT의 혼잡제어는 네트워크 상태를 예측하여 패킷 간 전송시간을 변화시킴으로써 기존 UDT보다 향상된 성능을 보인다. 그러나 AR UDT는 네트워크 상태 예측의 오차가 클 뿐만 아니라 rate control만을 공격적으로 조절하기 때문에 수신 버퍼의 초과로 인해 안정적인 성능을 기대하기 어렵다. 본 논문에서는 AR UDT환경에서 네트워크 상태에 따라 적응적으로 혼잡제어를 하는 기법을 제안한다. RTT(Round Trip Time)의 변화량에 따라 네트워크 상태를 예측하여 flow control과 rate control을 적응적으로 조절한다. 네트워크 시뮬레이션 결과를 통하여 AR UDT에 비해 전송속도와 안정성이 향상되었음을 보였다.

1. 서론

오늘날 네트워크 환경과 컴퓨팅 기술의 발달로 인해 대용량 데이터의 고속 전송 요구가 꾸준히 증가하고 있다. 이러한 고속 네트워크 환경에서 표준 전송 프로토콜인 TCP는 충분한 성능을 발휘하지 못한다. TCP의 AIMD (Additive Increase Multiplicative Decrease)방식은 혼잡으로부터 빠른 회피와 데이터 전송의 신뢰성을 제공한다. 그러나 네트워크 대역폭과 지연의 곱이 큰(HBDP: High Bandwidth Delay Product) 환경에서는 빠른 가용대역폭 확보가 어렵고, 혼잡윈도우 증가속도로 인한 대역효율 문제가 발생한다.

반면 UDP는 TCP보다 전송속도가 빠르고 부하가 적은 반면 전송 신뢰성을 보장받지 못한다. 이러한 문제점을 극복하기 위해 많은 프로토콜들이 연구되었으며 특히 UDT (UDP-based Data Transfer)[1-3]는 HBDP 네트워크 환경에서 뛰어난 성능을 보이는 것으로 알려져 있다. UDT는 기존 UDP의 전송 신뢰성 문제를 극복하였고, 응용계층에서 동작하기 때문에 커널 수정 없이 쉽게 활용할 수 있다. 특히 전송 시작 단계에서 증가인자 값을 빠르게 증가시키고, 시간이 지나면서 증가인자 값을 점차적으로 감소시킴으로써 TCP에 비해 가용대역폭을 빠르게 확보한다. 그러나 UDT는 고정된 시간(sync interval) 간격으로 혼잡제어를 하기 때문에 네트워크 상태가 불안정한 경우에 신속한 가용대역폭 확보에 어려움이 있다.

AR UDT(Adaptive Rate control UDT)[4]는 네트워크 상태를 예측하여 sync interval 시간동안 적응적으로 rate control을 수행함으로써 빠르게 가용대역폭을 확보할 수 있다. 네트워크 상태를 RTT(Round Trip Time) 변화량에 따라 4구간으로 설정하여 패킷간 전송 시간인 sending period를 조절함으로써 rate control을 수행한다. AR UDT의 네트워크 상태 예측은 t-2 구간 까지 만을 고려하기 때문에 예측 값의 오차가 크다. 또한 혼잡제어 기법 중 rate control만을 공격적으로 수행하기 때문에 수신 버퍼의 초과로 인한 연속적 패킷 손실이 우려된다.

본 논문에서는 네트워크 상태를 보다 정교하게 판단하여 flow control과 rate control을 조절하는 적응적 혼잡제어 기법을 제안한다. 네트워크 상태 판단에 예측 보정 값을 추가함으로써 AR UDT보다 정교한 판단이 가능하다. 또한 rate control과 flow control을 적응적으로 조절함으로써 전송속도와 안정성을 향상시킨다.

본 논문의 구성은 다음과 같다. 2장에서 관련연구로 UDT와 AR UDT를 살펴보고, 3장에서 네트워크 상태에 따른 AR UDT 혼잡제어 기법을 설명한다. 4장에서는 네트워크 시뮬레이션 결과를 보여주고 이를 분석하며, 5장에서 본 논문의 결론을 맺는다.

2. 관련연구

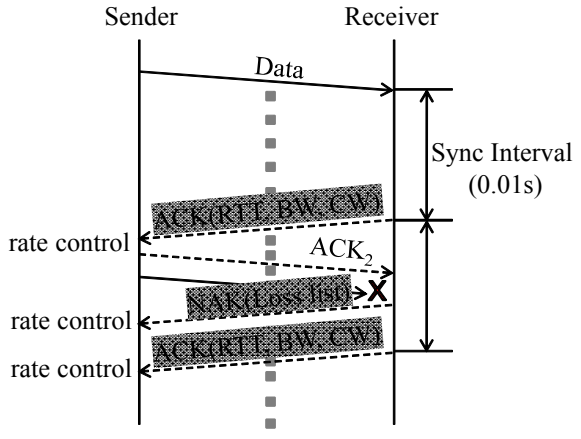
2.1 UDT(UDP-based Data Transfer)

UDT는 UDP기반의 데이터 전송프로토콜로 응용계층에서 동작하기 때문에 별도의 커널 수정 없이 사용이 가능하다. TCP나 UDP의 socket API를 통하여 채널을 설정하는 것과 유사하게 응용계층에서 채널을 설정하여 데이터를 전송할 수 있다[1].

UDT의 혼잡제어 기법은 TCP와 유사한 DAIMD (Decrease Additive Increase Multiplicative Decrease) 알고리즘을 사용한다. DAIMD 혼잡제어 기법은 초기 단계에 전송량을 급격하게 증가시키고, 차츰 증가량을 줄이는 방법으로 최대한 빨리 윈도우 크기를 증가시킨다. DAIMD의 혼잡제어는 flow control과 rate control을 통해 이루어진다.

Flow control은 송신측이 ACK 패킷을 수신하지 않고 전송할 수 있는 패킷 수를 제한하는데 사용된다. 이러한 flow control은 수신 측의 혼잡 윈도우 크기와 가용 버퍼 크기 사이의 최솟값을 송신 측에 전달함으로써 이루어진다. 혼잡 윈도우 크기는 패킷 도착 속도와 sync interval, RTT의 가중치 곱과 합으로 계산된다. 여기서 sync interval은 rate control이 수행되는 구간시간이며 0.01초의 상수 값이다.

Rate control은 패킷을 보내고 다음 패킷을 언제 보내야 할지에 대한 간격을 조절하는 것이다. 즉, 수신 측 버퍼의 오버플로우를 방지하기 위해서 수신측이 결정한 윈도우 값을 바탕으로 송신 측에서 패킷 간의 전송 간격인 sending period를 조절한다. (그림 1)은 UDT의 rate control을 나타낸다. Rate control은 ACK 패킷과 NAK 패킷을 수신한 경우에 실행된다. ACK 패킷에는 RTT, BW(bandwidth), CW(congestion window)가 포함되며 BW값에 따라 sending period를 조절함으로써 rate control이 실행된다. NAK 패킷을 수신하게 되면 sending period를 1/8만큼 증가시켜 패킷을 천천히 전송함으로써 연속적인 패킷 손실을 방지한다.



(그림 1) UDT rate control

UDT는 데이터 손실이 발생하지 않을 경우 0.01초의 sync interval 동안 약 833개의 패킷이 전송된다(1Gb/s link capacity, MTU 1500). sync interval 구간동안 일정한 주기로 패킷을 전송하기 때문에 네트워크 상태가 불안정할 경우 신속한 가용대역폭의 확보가 어렵고, burst 트래픽의 발생 가능성이 높다[4, 5].

2.2 AR UDT(Adaptive Rate control UDT)

AR UDT는 네트워크 상태에 따라 sync interval 구간에서 sending period를 가변적으로 변화시키는 rate control 기법이다. UDT에 비해 공격적인 rate control로 HBDP환경에서 신속한 가용대역폭 확보가 가능하다. AR UDT는 (그림 2)와 같이 네트워크 상태 판단 단계, sending period weight 값 설정 단계, sync interval 구간에서의 sending period 변화 단계로 구분된다.

네트워크 상태 판단은 다음 sync interval 에서의 RTT를 예측에 따라 판단한다. t+1시간의 RTT는 t-2와 t-1에서의 RTT차이와 t와 t-1구간에서의 RTT 차이에 대한 평균값을 t시간의 RTT에 더함으로써 계산된다. 예측된 RTT와 현재 RTT를 비교하여 증감률 R_{idr}을 계산한다.

Sending period weight 값 설정은 계산된 R_{idr}에 따라 네트워크 상태를 4단계로 구분하여 weight 값을 결정한다. 예측된 RTT의 변화량이 1 이하의 값일 경우 네트워크 상태가 안정적인 것으로 판단하여 0.99또는 0.999의 weight값을 설정하고, 그 이상인 경우 1.0 또는 1.001 값을 설정한다.

sync interval 구간에서의 sending period 변화는 이전 단계에서 설정된 weight값에 따라 sending period를 변화시키는 rate control을 수행 하는 단계이다. sync interval 구간동안 16개 패킷 마다 weight값에 따른 sending period를 변화시킴으로써 공격적인 rate control을 수행한다.

Algorithm

```
# 네트워크 상태 판단
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;

# sending period weight 값 결정
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;

# sync-interval 구간에서의 sending period 변화
If (0 ==  next_sequence_number % 16)
      PktSP = PkrSP * PktSPW;
```

(그림 2) AR UDT 알고리즘

AR UDT는 네트워크 상태에 따라 rate control을 수행한다. 네트워크 상태를 현재 t 시간과 t-2 사이의 RTT만을 고려하여 판단하기 때문에 정확한 RTT의 예측이 어렵고 오차가 매우 크다. 또한 sync interval구간에서 flow window 크기는 고려하지 않고 rate control만을 공격적으로 수행하기 때문에 수신 버퍼의 초과로 인한 패킷 손실이 우려된다.

3. 네트워크 상태에 따른 AR UDT 혼잡제어

본 논문에서는 네트워크 상태를 보다 정교하게 예측하여 flow control과 rate control을 수행하는 적응적 혼잡제어 기법을 제안한다. 네트워크 상태는 AR UDT와 유사하게 RTT를 기반으로 판단한다. 그러나 RTT를 예측함에 있어 보정 값을 추가하여 예측 오차율을 감소시킨다. 또한 AR UDT가 rate control을 수행한 반면 제안 기법에서는 rate control에 따라 flow control을 병행한 혼잡제어 기법을 사용한다. 제안 기법은 크게 네트워크 상태 판단 단계, 혼잡제어 weight값 결정 단계, sync interval 에서 혼잡제어 단계로 구분된다.

3.1 네트워크 상태 판단

네트워크 상태는 이전 ACK패킷에 포함된 RTT를 기반으로 다음 sync interval 에서의 RTT를 예측하여 변화량으로 판단한다. 네트워크 상태 판단을 위해 먼저 다음 sync interval 구간에서의 RTT인 R_{est}를 식 (1)과 같이 계산한다. RTT 예측 값은 현재시간 t에서 이전 i시간동안 각 구간의 RTT변화량 평균을 구한 후에 현재 RTT인 R_t와 보정값인 CV를 더함으로써 계산된다. 시간 구간별 RTT차이 값인 $\triangle RDif_i$ 는 $RDif_{i+1} - RDif_i$ 를 나타낸다.

$$R_{est} = \frac{\left(\sum_{i=0}^{n-1} \triangle RDif_i\right)}{n} + R_t + CV \qquad (1)$$

보정 값 CV는 식 (2)와 같이 계산된다. t-1시간에서 예측한 R_{est}값과 t시간의 실제 RTT 차이를 t-j 시간동안의 평균으로 구한다. 이때 t-j구간에서의 오차값 중 최댓값인 $\triangle EV_{max}$와 최솟값인 $\triangle EV_{min}$은 평균 계산에서 제외시킨다. 시간 구간별 오차값인 $\triangle EV_j$ 는 $R_{t+1} - R_{t\_est}$ 로 계산된다.

$$CV = \frac{\left(\left(\sum_{j=0}^{n-1} \triangle EV_j\right) - \triangle EV_{\max} - \triangle EV_{\min}\right)}{(n-2)} \qquad (2)$$

보정 값을 적용하여 계산된 R_{est}값과 현재 t시간의 RTT를 비교하여 RTT 증감비율 값을 식 (3)과 같이 계산한다.

$$R_{idr} = \frac{R_{est}}{R_t} \qquad (3)$$

R_{idr} 값이 1.0 이하일 경우는 RTT가 감소함으로써 네트워크 상태가 혼잡하지 않다는 것을 의미하고, 1.0 이상이일 경우는 혼잡상태를 나타낸다. R_{idr} 값에 따라 α, 1.0, β 구간별로 네트워크 상태를 4단계로 세분화 한다. α 와 β 값은 AR UDT와 같이 설정된다.

3.2 혼잡제어 weight값 결정

네트워크 혼잡상태에 따라 혼잡제어를 위한 FCW(Flow Control Weight)값과 RCW(Rate Control Weight)값을 다음과 같이 결정한다.

- $R_{idr} \leq \alpha$
  $RCW = 0.99$, $FCW^{nRC_{t-1}}$
- $\alpha < R_{idr} \leq 1.0$
  $RCW = 0.999$, $FCW^{nRC_{t-1}}$
- $1.0 < R_{idr} < \beta$
  $RCW = 1.0$, $FCW^{nRC_{t-1}}$
- $R_{idr} > \beta$
  $RCW = 1.001$, $FCW^{nRC_{t-1}}$

각 구간에 따라 RCW 값은 0.99, 0.999, 1.0, 1.001 값을 설정한다. $R_{idr} > 1.0$ 구간은 네트워크 상태가 혼잡하지 않은 구간으로 0.99와 0.999의 값을 설정 한다. 네트워크 상태가 혼잡에 접어드는 상태인 $1.0 < R_{idr} < \beta$ 구간에서는 weight값을 1.0으로 설정하여 상태유지를 하고, 그 이상일 경우는 1.001값으로 설정하여 rate control 과정에서 sending period를 증가시키도록 한다. FCW값은 RCW값에 nRC_{t-1} 값의 승으로 설정된다. nRC_{t-1} 값은 이전 sync interval구간에서 보내진 총 패킷을 16으로 나눈 몫이다. 이것은 sync interval구간에의 rate control 수와 같다.

3.3 혼잡제어

혼잡제어는 RCW와 FCW값에 따라 flow control과 rate control이 수행된다. Flow control은 기존 UDT의 flow window 크기 값에 FCW를 곱해서 계산된다. 이러한 Flow control은 ACK 패킷을 수신 후에 실행된다. 즉 RCW값에 따라 rate control이 진행될 때 sync interval에서 전송되어질 패킷의 변화율만큼 flow window 크기를 조절하는 것이다.

Rate control 은 sync interval 구간동안 매 16개 패킷마다 실행된다. ACK 패킷 수신 후에 UDT의 rate control 알고리즘에 따라 sending period가 계산된다. sync interval 구간에서는 sending period값에 매 16개 패킷마다 RCW값을 곱해서 sending period를 다시 계산한다. 각 패킷 마다 rate control을 수행할 경우 패킷 sending period가 급격하게 증가하여 혼잡이 발생할 가능성이 높아진다. 또한 각 패킷 마다 sending period가 달라질 수 있기 때문에 수신측에서 bandwidth 측정이 어려워진다.

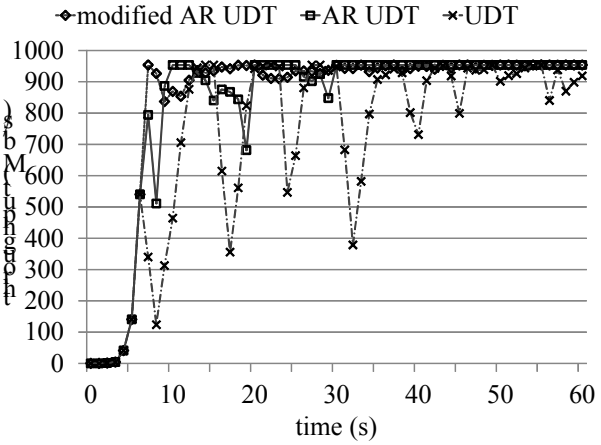
4. 성능평가

본 논문에서 제안한 AR UDT의 적응적 혼잡제어 기법을 네트워크 시뮬레이터인 NS-2[6]를 이용하여 성능 실험하였다. 제안 기법의 성능을 UDT, AR UDT와 비교하여 throughput 관점에서 성능평가 하였다. 실험에 사용한 파라미터는 다음 표 1과 같다.

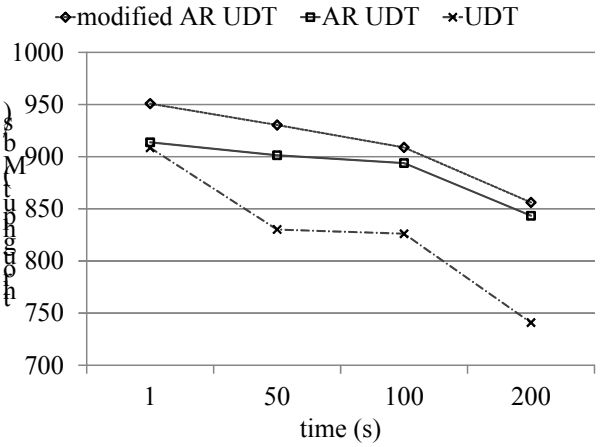
<표 1> 시뮬레이션 파라미터

파라미터	값
Link Capacity	1 Gb/s
MTU	1500 byte
RTT	1ms, 50ms, 100ms, 200ms
Queue option	DropTail
시뮬레이션 시간	60 sec.

(그림 3)은 RTT 200ms 환경에서의 성능비교 그래프다. 기존 UDT는 8, 17, 32초에서 400Mb/s 이하의 저조한 성능을 보이고 있으며 AR UDT는 19, 29초에 불안정한 모습을 보인다. 반면 제안 기법인 modified AR UDT는 전체적으로 안정된 성능을 보였다. 제안기법, AR UDT, UDT의 평균 throughput은 각 각 856 Mb/s, 843Mb/s, 740Mb/s 로 제안기법이 가장 안정적이며 좋은 성능을 보였다.



(그림 3) RTT 200ms 환경에서의 성능비교



(그림 4) RTT에 따른 성능비교

(그림 4)는 RTT별로 제안 기법, AR UDT, UDT의 throughput 성능비교 그래프다. 제안 기법은 UDT대비 RTT 200ms 환경에서 약 16% 성능향상을 보였다. AR UDT대비에서는 1ms 환경에서 약 4%의 성능향상을 보였다. 제안 기법은 RTT가 낮은 상황에서 AR UDT에 비해 큰 성능차이를 보이고 있다. 그 이유는 제안 기법에서는 네트워크 상황에 따라 flow window 크기를 조절하기 때문에 수신 버퍼의 초과로 인한 패킷 손실이 적기 때문이다.

5. 결론

본 논문에서는 AR UDT에서 rate control만을 수행함에 따른 성능저하와, 네트워크 상태 예측 시 오차값이 큰 문제를 해결하는 기법을 제안하였다. 제안 기법은 네트워크 상태 판단 시 예측된 값에 오차 보정값을 추가함으로써 AR UDT에 비해 향상된 네트워크 상태 판단이 가능하다. 또한 혼잡 제어 시에는 rate control뿐만 아니라 flow control도 같이 수행되기 때문에 수신 버퍼의 초과로 인한 성능 저하를 방지한다.

시뮬레이션 결과 제안 기법은 RTT가 낮은 환경에서 throughput의 성능향상이 컸고, 전체적으로 안정된 성능을 보였다. 향후 연구과제로 TCP 패킷과의 공정성에 대한 비교연구를 통해 실제 네트워크에 적용하는 연구를 하고자 한다.

참고문헌

[1] 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

[2] Y. Gu and R. L. Grossman, “UDTv4: Improvements on Performance and Usability,” Gridnets 2008, vol. 2, no. 1, pp. 9-23, Oct. 2008

[3] Y. Gu, X. Hong, and R. L. Grossman, “Experiences in Design and Implementation of a High Performance Transport Protocol,” Pro. on ACM SC'04, pp. 6-12, Nov. 2004

[4] 안도식, 왕기철, 김승해, 조기환, “네트워크 트래픽 상태에 적응적인 UDT Rate Control 기법,” 전자공학회 논문지, 43권 TC편, 3호, pp. 241-249, 2011

[5] 안도식, 왕기철, 조기환, “UDT환경에서 RTT예측에 의한 Sync-Interval 구간의 Rate Control 기법,” 한국정보처리학회 추계학술대회, 17권, 2호, pp. 1038-1041, 2010

[6] Network Simulator, ns-2, http://www.isi.edu/ns nam/ns