제한된 메모리의 모바일 수신자를 고려한 수신자 기반 TCP 흐름 제어

(A Receiver-driven TCP Flow Control for Memory Constrained Mobile Receiver)

이 종 민 † 차 호 정 ††
(Jongmin Lee) (Hojung Cha)

요 약 본 논문은 제한된 메모리를 가진 모바일 수신자를 고려한 무선 상태 적응적인 TCP 흐름 제어 방법을 제안한다. 수신자에 의한 TCP 흐름 제어는 수신자에서 Advertised 윈도우를 조정함으로써 수행된다. 제안된 방법은 수신자가 사용 가능한 무선 대역폭과 패킷 전송 지연 시간을 동적으로 측정하며, 측정된 정보를 기반으로 Advertised 윈도우를 적절히 조정한다. 무선 상태를 반영한 Advertised 윈도우의 조정으로 인해 송신자의 전송 성능 향상과 종단간 패킷 전송 지연 시간을 줄일 수 있다. 제안된 방법은 수신자에서의 TCP 변경만으로 구현될 수 있고 송신자나 중간 라우터의 변경을 필요로 하지 않는다. 제안된 방법의 구현과 CDMA2000 1x 환경에서의 실험을 통해 수신자 버퍼 크기가 2896 Bytes일 경우, 흐름 제어를 사용할 때가 기존 방식보다 전송률을 약 5배 향상시킬 수 있음을 보인다. 또한, 수신자 버퍼 크기가 64 KBytes일 경우 흐름 제어를 사용할 때가 기존 방식보다 때보다 종단간 패킷 왕복 시간은 반 이하로 줄일 수 있음을 보인다.

키워드 : 모바일, TCP, 흐름제어

Abstract This paper presents a receiver-driven TCP flow control mechanism, which is adaptive to the wireless condition, for memory constrained mobile receiver. A receiver-driven TCP flow control is, in general, achieved by adjusting the size of advertised window at the receiver. The proposed method constantly measures at the receiver both the available wireless bandwidth and the packet round-trip time. Depending on the measured values, the receiver adjusts appropriately the size of advertised window. Constrained by the adjusted window which reflects the current state of the wireless network, the sender achieves an improved TCP throughput as well as the reduced round-trip packet delay. Its implementation only affects the protocol stack at the receiver and hence neither the sender nor the router are required to be modified. The mechanism has been implemented in real environments. The experimental results show that in CDMA2000 1x networks the TCP throughput of the proposed method has improved about 5 times over the conventional method when the receiver's buffer size is limited to 2896 bytes. Also, with 64Kbytes of buffer size, the packet round-trip time of the proposed method has been reduced in half, compared the case with the conventional method.

Key words : Mobile, TCP, Flow Control

1. 서 론

최근 무선 넷트웍 기술과 하드웨어 기술의 발달로 휴대 단말기에서도 음성 통신 위주의 기능뿐만 아니라 다양한 응용 프로그램을 실행시킬 수 있는 환경을 제공하게 되었다. 이러한 향상된 무선 넷트웍 환경과 진보된 휴대 단말기에 기반하여 기존의 음성 통신뿐만 아니라 무선 인터넷, 멀티미디어 서비스와 같은 다양한 기능이 제공되고 있다[1]. 무선 넷트웍을 통한 다양한 응용 서비스에서 사용되는 대표적인 프로토콜로는 TCP가 있다. TCP에서 통신시 사용되는 수신자의 버퍼는 패킷의 조립 및 버퍼링에 쓰인다. 휴대폰과 같은 이동 단말기는 데스크톱 컴퓨터와는 달리 연산 능력이나 메모리 등의

· 본 연구는 과학재단 특정기초연구사업(R01-2002-000-00141-0)에 의해 지원되었음
† 학생회원 : 연세대학교 컴퓨터과학과
jmlee@cs.yonsei.ac.kr
†† 종신회원 : 연세대학교 컴퓨터과학과 교수
hjcha@cs.yonsei.ac.kr
논문접수 : 2003년 9월 16일
심사완료 : 2003년 11월 6일

시스템 자원에 있어 많은 제약을 가지고 있다. 수신자가 제한된 메모리를 가지고 있는 이동 단말기의 경우, 수신자는 시스템 자원의 제한 때문에 작은 크기의 버퍼를 갖게 되며, 이는 송신자의 전송률을 제한하여 전체적인 전송 성능을 감소시킨다[2].

TCP는 넷트웍의 혼잡 상황을 제어하기 위해 윈도우 기반의 혼잡 제어 방식을 사용한다[3]. TCP 송신자는 혼잡 윈도우와 수신자의 버퍼 크기를 의미하는 Advertised 윈도우 중 작은 값에 따라 수신자로부터 응답을 받지 않고 넷트웍으로 전송 가능한 자료의 양을 조절함으로써 전송률을 조정한다. 오류 발생 확률이 적은 유선 환경에서는 최대 전송 성능을 얻기 위해 수신자의 Advertised 윈도우가 최소 넷트웍 허용 대역폭 이상으로 커야 한다. 유선 환경에 비해 상대적으로 대역폭이 낮고 오류 발생이 많은 무선 환경의 경우, 큰 Advertised 윈도우는 종단 라우터에서의 버퍼링 부하를 증가시키며 혼잡 발생 확률을 증가시킨다[4]. 또한, 무선 환경에서의 패킷 손실로 인해 재전송 되는 패킷이 종단 라우터에서의 버퍼링 지연에 의해 늦게 전송됨에 따라 패킷 전송 시간 초과가 발생하여 전송 성능의 감소를 초래한다[5]. 라우터에서의 패킷 버퍼링 부하에 따른 자료 전송의 지연을 줄이고 혼잡 상황을 피하며, 종단간 대역폭의 최대 활용을 위해서는 적절한 수신자 Advertised 윈도우 크기 조정에 의한 효율적인 흐름 제어 방식이 필요하다[6].

본 논문에서는 무선 환경에서 전송 성능의 향상과 넷트웍에서의 지연 및 혼잡 상황 발생을 최소화하기 위한 TCP 흐름 제어 방식을 제안한다. 제안하는 TCP 흐름 제어 방식은 제한된 메모리를 가지고 있는 수신자를 고려하였으며, 수신자에서 무선 상태에 따라 Advertised 윈도우를 조정함으로써 효율적인 TCP 흐름 제어를 수행한다. 제안된 흐름 제어 방식을 구현하고 실제 환경에서 실험하여 구현된 흐름 제어 방식의 효율성을 검증하고 성능을 분석한다.

논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 살펴보고, 3 장에서는 논문에서 제안하는 수신자에 의한 TCP 흐름 제어 방식에 대해 기술한다. 4 장에서는 실험 방법 및 결과에 대해 기술하고, 5 장에서 결론을 맺는다.

2. 관련 연구

일반적으로 무선 환경에 직접 연결되어 있는 수신자 단말이 무선 상태를 가장 잘 파악할 수 있다. 효율적인 흐름 제어를 위해서 수신자는 무선 상태 감시와 이에 따른 무선 넷트웍 상에서의 혼잡 제어를 수행하고, 송신자는 나머지 넷트웍에 대한 혼잡 제어를 수행하는 협력적인 혼잡 제어(Cooperative Congestion Control) 정책이 효과적이다[7]. 수신자에서의 흐름 제어 수행 방법은 수신자에서 송신자로 전달되는 Advertised 윈도우를 조정함으로써 수행되며, 크게 중간 라우터에서 Advertised 윈도우를 조절하는 방법과 수신자가 Advertised 윈도우를 조절하는 방법으로 구분할 수 있다.

Kalampoukas[8]와 Aweya[9]는 흐름 제어를 위해 넷트웍의 중간 라우터에서 수신자가 송신자에게 보내는 응답 패킷의 Advertised 윈도우를 넷트웍 상황에 맞도록 변경하는 방법을 제안하였다. TCP 자체의 변경이 필요 없다는 장점이 있으나 넷트웍 라우터의 변경을 필요로 하며, 무선 환경에 대한 고려가 없다는 문제점을 가지고 있다.

수신자가 Advertised 윈도우를 조절하여 흐름 제어를 수행하는 방법에 관한 연구로는 Spring[7], Fisk[10] 그리고 Andrew[11]의 연구가 있다. Spring은 종단 넷트웍이 저 대역폭인 넷트웍 환경에서 종단 넷트웍의 대역폭에 근거하여 수신자의 버퍼 크기를 동적 변경하는 방법을 제안하였다. Fisk는 수신자의 버퍼 크기를 송신자의 혼잡 윈도우의 크기보다 항상 크도록 동적 변경하여 TCP의 혼잡 제어 방법에 의해 전송률 제어가 이루어지도록 하는 방법을 제안하였다. Andrew는 종단 라우터에서의 혼잡 상황 정보에 바탕을 두어 수신자가 Advertised 윈도우를 조정함으로써 여러 세션의 TCP 흐름에 대한 공정한 대역폭 분배 방법을 제안하였다. 이들 연구들은 넷트웍의 중간 라우터 변경 없이 수신자의 TCP 수정만으로 구현될 수 있다는 장점이 있다. 그러나 무선 환경에 대한 고려가 없고 수신자가 제한된 시스템 자원을 가지고 있을 경우에 대한 해결책을 제시하지 못하고 있다.

무선 환경에서 수신자 버퍼 크기가 TCP 전송 성능에 미치는 영향을 조사한 연구로는 Mukhtar[6]의 연구가 있다. Mukhtar는 유무선 연동 환경에서 최상의 TCP 전송 성능을 얻기 위한 효율적인 Advertised 윈도우의 크기를 결정하는 방법을 제안하고, 수신자 버퍼 크기가 TCP 전송 성능에 미치는 영향을 연구하였다. 적절한 Advertised 윈도우 크기의 선택으로 무선 환경에서 TCP 전송 성능을 향상시킬 수 있으나 계산에 의해 결정된 값을 초기 연결 설정시 고정하여 사용함으로써 동적으로 변하는 넷트웍 상황에 올바로 대처하지 못하고, 수신자가 제한된 시스템 자원을 가지고 있을 경우에 대한 고려를 하고 있지 않다.

3. Advertised 윈도우 조정에 의한 TCP 흐름 제어

다음은 무선 환경에서 수신자에 의한 TCP 흐름 제어

를 수행하기 위한 시스템에 대해 기술한다. 이를 위하여 수신자에서 무선 상태를 감지하기 위한 무선 대역폭 측정 방법, 종단간 패킷 왕복 시간 측정 방법에 대해 기술한다. 그리고 무선 상태에 따른 효율적인 TCP 흐름 제어를 수행하기 위한 적절한 Advertised 윈도우 크기 계산 방법 및 제한된 수신자 자원을 고려한 효율적인 Advertised 윈도우 조절 방법에 대해 기술한다.

3.1 시스템 개요

무선 넷트웍은 유선 넷트웍에 비해 낮은 대역폭을 가지며 유무선 연동 환경의 종단에 위치한다. 유무선 연동 환경에서의 종단간 최대 대역폭은 전체 넷트웍의 종단에 위치한 무선 넷트웍의 대역폭에 의해 결정된다[7]. 논문에서 제안하는 수신자에 의한 TCP 흐름 제어는 수신자가 무선 넷트웍의 대역폭과 종단간 패킷 왕복 시간을 측정하여 무선 상태를 감지하고 적절히 Advertised 윈도우를 조정하여 송신자의 전송률을 조정한다. 그림 1은 전체 시스템의 구조를 보여준다. 그림에서 베이스 스테이션은 유무선 연동 환경에서 최종 라우터의 역할을 수행하며 수신자와 무선으로 연결되어 있다. 무선 환경이 좋아지면 프로토콜 상위 계층에서 측정되는 무선 대역폭은 실제 최대 무선 대역폭과 비슷해진다. 무선 환경이 악화되면 하위 프로토콜 계층에서의 오류 보정 및 복구 방법에 의해 상위 프로토콜 계층에서 측정되는 무선 대역폭은 작아지게 된다. 수신자는 무선 대역폭을 측정함으로써 무선 대역폭 변화에 따라 무선 상태 변화를 감지한다. 이 정보를 바탕으로 송신자의 전송률을 제한하기 위한 전송률을 결정한다. 송신자의 전송률이 무선 대역폭보다 작을 경우 TCP는 최적의 전송 성능을 얻을 수 없다. 송신자의 전송률이 무선 대역폭보다 클 경우, 이는 종단 라우터에서의 버퍼링 부하를 증가시켜 혼잡 발생 확률을 증가시킨다. TCP 전송 성능의 향상과 종단 라우터에서의 부하를 줄이기 위해 송신자의 전송률을 무선 대역폭으로 제한하도록 한다.

TCP 송신자의 전송률은 혼잡 윈도우와 수신자의 Advertised 윈도우에 의해 제한 받는다. 수신자는 송신자로 전달하는 Advertised 윈도우를 조절함으로써 송신자의 전송률을 제어할 수 있다. TCP에서 효율적인 전송을 위한 최소 Advertised 윈도우의 크기는 종단간 파이프 크기를 의미하는 대역폭 지연 산출(Bandwidth-delay product)에 의해 결정된다[12]. Advertised 윈도우의 크기는 수신자의 버퍼 크기를 의미하며 효율적인 TCP 전송 성능을 위한 최소 수신자 버퍼 크기는 대역폭 지연 산출에 의해 식 (1)로 구할 수 있다[13].

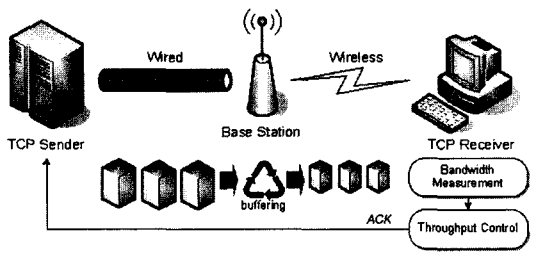

그림 1 시스템 구조

표 1 논문에 사용된 인자

P_i	수신자가 송신자로부터 i 번째로 수신한 TCP 패킷
ACK	수신자가 송신자로 보내는 응답 메시지
I_i	P_i와 P_{i-1}의 수신 시간 견격 (milliseconds)
$B_{net(i)}$	P_i 수신시 실제 무선 넷트웍 대역폭 (Bytes/sec)
$B_{calc(i)}$	P_i 수신시 측정된 무선 넷트웍 대역폭 (Bytes/sec)
$B_{adjust(i)}$	P_i 수신시 보정된 무선 넷트웍 대역폭 (Bytes/sec)
α	$B_{calc(i)}$를 위한 보정 상수
$RTT_{real(i)}$	P_i 수신시 실제 종단간 패킷 왕복 시간 (milliseconds)
$RTT_{calc(i)}$	P_i 수신시 측정된 종단간 패킷 왕복 시간 (milliseconds)
$Delay_{net(i)}$	P_i 수신시 넷트웍에서 발생한 지연 시간 (milliseconds)
$BaseRTT_{real(i)}$	P_i 수신시 혼잡 상황이 없는 경우의 최소 RTT (milliseconds)
$BaseRTT_{calc(i)}$	P_i 수신시 계산된 $BaseRTT_{real}$ (milliseconds)
W_{cn}	혼잡 윈도우 (Bytes)
W_{ad}	Advertised 윈도우 (Bytes)
$W_{ad(i)}$	수신자가 송신자에게 i 번째 보낸 Advertised 윈도우 (Bytes)
S_{packet}	TCP 패킷 크기 (Bytes)
S_{buffer}	수신자가 사용 가능한 메모리 (Bytes)
S_{calc}	계산된 최적의 Advertised 윈도우 (Bytes)
$S_{calc(i)}$	i 번째 계산된 최적의 Advertised 윈도우 (Bytes)
S_{adjust}	흐름 조절이 적용된 최종 Advertised 윈도우 (Bytes)
$S_{adjust(i)}$	i 번째 흐름 조절이 적용된 최종 Advertised 윈도우 (Bytes)

$$buffersize = bandwidth \times RTT \quad (1)$$

TCP에서 사용되는 버퍼는 연결 설정시 할당되며 동적으로 변경되지 않는다. 고정된 수신자 버퍼 크기는 고정된 Advertised 윈도우가 송신자에게 전달되도록 하며, 적절하지 못한 Advertised 윈도우 전달로 인해 전송 성능 감소와 혼잡 상황을 유도할 수 있다. 논문은 효율적인 송신자 전송률 제어를 위해 수신자 버퍼 크기와 상관없이 무선 상태 변화에 따라 Advertised 윈도우를 조정하도록 한다. 이와 같은 과정은 수신자에서 동적으로 수행되며 무선 상태에 따라 적응적으로 송신자의 전송률을 제어하게 된다. TCP에서의 효율적인 Advertised 윈도우 크기는 식 (1)로 계산한 버퍼 크기가 되며, 이를 계산하기 위해 무선 대역폭(B_{net}) 측정 방법과 종단간 패킷 왕복 시간(RTT_{real})의 측정 방법이 필요하다. 표 1은 본 논문에서 사용되는 인자들의 정의를 보여준다.

3.2 수신자에서의 B_{net}과 RTT_{real} 측정

송신자의 전송률이 무선 대역폭보다 클 경우 종단 라우터의 버퍼링 부하가 증가한다. 이로 인해 혼잡 상황 발생 확률이 커지고 종단간 패킷 왕복 시간이 길어져 TCP 전송 시간 초과에 의한 전송률 감소를 가져온다.

논문에서 제안하는 방법은 송신자 전송률을 무선 대역폭으로 제한하여 종단 라우터의 버퍼링 부하를 감소시키도록 하는 것이다.

수신자가 i번째 TCP 패킷 P_i를 수신했을 때의 실제 무선 대역폭 $B_{net(i)}$는 무선 상태에 따라 동적으로 변한다. 수신자는 $B_{net(i)}$의 측정을 통해 무선 상태 변화를 알 수 있다. 수신자는 $B_{net(i)}$을 유추하기 위해 순간 무선 측정 대역폭 $B_{calc(i)}$를 식 (2)에 의해 계산한다.

$$B_{calc(i)} = \frac{S_{packet}}{I_i} \quad (2)$$

$B_{calc(i)}$는 이전 패킷과의 시간 간격 I_i와 TCP 패킷의 크기 S_{packet}을 이용하여 구할 수 있다. 순간적으로 무선 상황이 악화되면 넷트웍 하위 계층의 오류 복구 방식에 의해 버퍼링된 패킷이 한번에 상위 계층으로 전달될 수 있다. 이러한 경우 $B_{calc(i)}$는 $B_{net(i)}$보다 커질 수 있다. 또한 지속적으로 무선 상황이 악화되면 TCP 전송 시간 초과가 발생하여 $B_{calc(i)}$가 $B_{net(i)}$보다 큰 차이로 작아질 수 있다. 급변하고 부정확한 $B_{calc(i)}$으로부터 $B_{net(i)}$을 유도하기 위해 $B_{calc(i)}$를 보정할 필요가 있다. i번째 계산된 무선 보정 대역폭 $B_{adjust(i)}$는 이전에 보정된 무선 대역폭 $B_{adjust(i-1)}$을 일정 비율 반영하고 현재 측정된 순간 무선 대역폭 $B_{calc(i)}$를 일정 비율 반영하여 $B_{net(i)}$과 유사한 값을 가지도록 한다. $B_{adjust(i)}$는 이전의 보정된 정보를 반영하므로 무선 상황 악화에 따른 순간적인 $B_{calc(i)}$의 변화에 대해 급변하지 않고 안정된 값을 유지하게 된다. 또한, $B_{calc(i)}$의 지속적인 악화에 대해서는 현재의 측정된 정보를 반영하므로 점차 $B_{calc(i)}$에 수렴하게 된다. 과거와 현재의 정보를 바탕으로 $B_{adjust(i)}$는 식 (3)으로 구한다.

$$B_{adjust(i)} = \alpha \times B_{adjust(i-1)} + (1-\alpha) \times B_{calc(i)}, \; where \; 0 \le \alpha \le 1 \quad (3)$$

$B_{adjust(i)}$를 계산하기 위한 α 값은 실험을 통해 결정할 수 있다. α 값이 클수록 과거의 정보를 많이 반영하고 α 값이 작을수록 현재의 정보를 많이 반영한다. $B_{calc(i)}$의 변화는 무선 상태의 변화를 의미한다. 급변하는 무선 상태의 변화에 대해 안정된 값을 가지도록 하기 위해 α는 큰 값을 가지도록 한다. α가 큰 값을 가짐으로써, 순간적인 무선 상태 악화의 경우 $B_{adjust(i)}$이 서서히 변하므로 서버의 전송률 제어가 바로 이루어지지 않아 무선 상태 회복시 원래의 전송률을 그대로 유지할 수 있다. 지속적인 무선 상태 악화의 경우, $B_{adjust(i)}$로 인한 서버의 전송률이 서서히 제한되므로 종단 라우터의 버퍼링 부하가 순간적으로 증가할 수 있다. 그러나, 무선 상태 회복에 대한 전송률 복구를 빨리 수행할 수 있고 시간이 흐름에 따라 $B_{adjust(i)}$가 $B_{calc(i)}$에 수렴함으로써 종단 라우터의 버퍼링 부하를 감소시킬 수 있다.

한편, i번째 TCP 패킷 P_i를 수신했을 때의 실제 종단간 패킷 왕복 시간 $RTT_{real(i)}$는 식 (4)에서와 같이 넷트웍 혼잡 상황이 없을 경우의 최소 패킷 왕복 시간 $BaseRTT_{real(i)}$와 넷트웍에서 발생한 지연 시간 $Delay_{net(i)}$의 합으로 표현할 수 있다.

$$RTT_{real(i)} = BaseRTT_{real(i)} + Delay_{net(i)} \quad (4)$$

$Delay_{net(i)}$는 라우터에서 버퍼링으로 발생한 지연 시간 및 무선 환경에서 오류 복구 과정으로 발생한 지연 시간을 모두 포함한다. 수신자는 식 (1)에 의한 효율적인 Advertised 윈도우 크기를 계산하기 위하여 $RTT_{real(i)}$을 알아야 한다. 수신자에서 $RTT_{real(i)}$를 유추하기 위한 계산된 종단간 순간 패킷 왕복 시간 $RTT_{calc(i)}$는 다음에 근거하여 구한다.

TCP 송신자에서의 전송률 조정은 혼잡 윈도우 W_{cn}과 Advertised 윈도우 W_{ad} 중 작은 값에 의해 수신자의 응답 메시지 ACK를 받지 않고 전송 가능한 자료의 양을 결정함으로써 수행된다. W_{cn}이 W_{ad}보다 같거나 클 경우, 수신자가 W_{ad}를 ACK로 전달한 시간과 해당 W_{ad} 만큼의 자료를 모두 받은 시간과의 간격이 계산된 종단간 순간 패킷 왕복 시간인 $RTT_{calc(i)}$가 된다. 그림 2는 수신자에서 $RTT_{calc(i)}$를 측정하는 방법을 보여준다. 수신자는 i번째 TCP 패킷 P_i를 수신했을 경우 다음 TCP 패킷 P_{i+1}을 요구하는 ACK와 함께 i번째 Advertised 윈도우 $W_{ad(i)}$를 송신자로 전달한다. W_{cn}이 W_{ad}보다 크게 되면 송신자의 전송률은 W_{ad}에 의해 제한된다. 즉, 송신자에서 ACK 없이 전송 가능한 자료의 양은 W_{ad}를 넘지 않는다. 수신자가 $W_{ad(i)}$를 전송한 후부터 n 번째 패킷인 P_{i+n}을 수신했을 때까지 받은 패킷의 합이 $W_{ad(i)}$와 같다면, $W_{ad(i)}$를 보낸 시점과 P_{i+n}을 받을 때까지의 시간 간격이 P_{i+n}에 대한 측정된 종단간 순간 패킷 왕복 시간인 $RTT_{calc(i+n)}$이 되며 식 (5)에 의해 계산한다.

$$RTT_{calc(i+n)} = \sum_{k=1}^{n} I_{i+k}, \; where \; n = \frac{W_{ad(i)}}{S_{packet}} \quad (5)$$

식 (4)에서 보듯이 $RTT_{real(i)}$는 $Delay_{net(i)}$를 포함하고 수신자에서 측정된 $RTT_{calc(i)}$ 역시 $Delay_{net(i)}$를 포함한다. $RTT_{calc(i)}$를 이용하여 서버의 전송률을 제어할 경우, $Delay_{net(i)}$가 지속적으로 서버의 전송률에 반영되므로 종단간 패킷 왕복 시간을 $RTT_{calc(i)}$ 이하로 줄일 수 없다. 따라서, $Delay_{net(i)}$를 최소로 하기 위하여 수신자가 $BaseRTT_{real(i)}$를 이용하여 송신자의 전송률을 제어하도록 하여 종단간 패킷 왕복 시간을 줄이도록 한다.

$$BaseRTT_{calc(i)} = RTT^{\min}_{calc(i)} \quad (6)$$

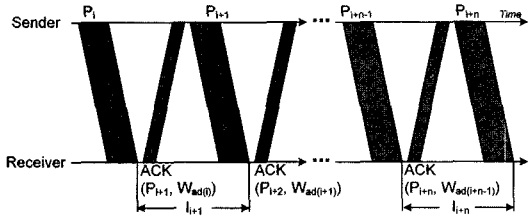

그림 2 RTT 측정

$BaseRTT_{real(i)}$를 유추하기 위한 혼잡 상황이 없을 경우의 최소 패킷 왕복 시간 $BaseRTT_{calc(i)}$는 식 (6)에서와 같이 $RTT_{calc(i)}$의 최소값으로 한다. $RTT_{calc(i)}$의 최소값을 $BaseRTT_{calc(i)}$로 사용하여 송신자의 전송률 조정에 반영함으로써 $Delay_{net(i)}$에 의한 영향을 최소화하고 종단간 패킷 왕복 시간을 최대한 줄이도록 한다.

i번째 보정된 무선 넷트웍 대역폭 $B_{adjust(i)}$는 식 (3)에 의해 계산할 수 있고, 혼잡 상황이 없을 때의 종단간 패킷 왕복 시간 $BaseRTT_{calc(i)}$는 식 (6)에 의해 계산할 수 있다. 따라서, 효율적인 TCP 전송을 위한 i번째 계산된 최적의 수신자 버퍼 크기 $S_{calc(i)}$는 식 (1), (3), (6)에 의해 식 (7)과 같이 $B_{adjust(i)}$와 $BaseRTT_{calc(i)}$의 곱으로 계산할 수 있다.

$$S_{calc(i)} = B_{adjust(i)} \times BaseRTT_{calc(i)} \quad (7)$$

3.3 Advertised 윈도우 조정

휴대용 단말기와 같이 제한된 시스템 자원을 가지고 있는 수신자는 실제 사용 가능한 메모리 S_{buffer}가 계산된 버퍼 크기 S_{calc}보다 작을 수 있다. 이러한 경우 송신자의 자료 전송률이 S_{buffer}에 의해 제한되므로 충분한 TCP 전송 성능의 향상을 이룰 수 없다. 논문에서는 S_{buffer}가 S_{calc}보다 작을 경우, 실제 메모리는 S_{buffer}만을 사용하고 서버에 전달하는 W_{ad}는 S_{calc}를 보정한 S_{adjust}를 전달하여 전송 성능을 높일 수 있도록 한다.

식 (7)에서 $B_{adjust(i)}$는 무선 상태에 따라 변하며 이에 따라 $S_{calc(i)}$도 변하게 된다. 특히, 연결 초기 단계에서는 측정된 $B_{adjust(i)}$와 $BaseRTT_{calc(i)}$ 값이 정확하지 않을 수 있으며, 이로 인해 $S_{calc(i)}$ 값이 정확하지 않을 수 있다. 급변하고 정확하지 않은 S_{calc}를 보정하기 위해 수신자는 TCP의 혼잡 윈도우 제어 방식에서 쓰이는 AIMD (Additive Increase Multiplicative Decrease)[14] 방법과 유사한 방법으로 W_{ad}를 조절하며, 이 방법에 의해 흐름 제어가 적용된 최종 Advertised 윈도우 S_{adjust}를 송신자로 전달한다. W_{ad} 조절 방법은 수신한 패킷의 순서가 올바를 경우, S_{calc}에 근접하도록 W_{ad}를 패킷 크기 S_{packet} 만큼 단계적으로 증감시킨다. 수신자가 S_{buffer}보다 큰 W_{ad}를 송신자에게 전달하게 되면 패킷 손실 발생시 송신자의 재전송 부하를 가중시킬 수 있다. 패킷 손실이 야기하는 송신자에서의 재전송 부하를 줄이고 최대의 TCP 전송 성능을 얻기 위해 수신자는 순서가 바뀐 패킷이 도착시 W_{ad}를 S_{buffer}로 줄이는 방법을 사용한다. 그림 3은 W_{ad} 조절 과정을 보여준다.

S_{calc}가 S_{buffer}보다 큰 경우, 수신자는 S_{buffer}를 변동하지 않고 S_{adjust}만을 바꾸게 된다. 무선 환경이 좋을 경우, S_{adjust}가 S_{calc}가 되므로 TCP 전송 성능을 높일 수 있다. 무선 환경이 나쁜 상황에서는 패킷 손실이 발생하며, S_{adjust}를 조절하지 않았을 경우에 비해 추가의 재전송이 필요할 수 있다. 그러나 지속되는 무선 환경의 악화에 대해서는 S_{adjust}가 S_{buffer}를 유지하게 되므로 추가

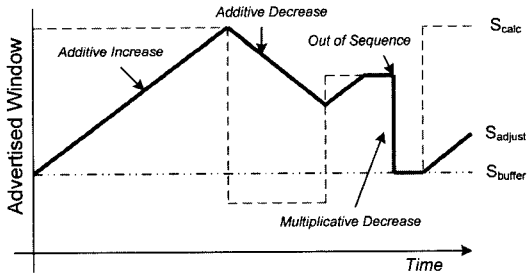

그림 3 Advertised 윈도우 조절

```
WHEN_RECEIVE_PACKET() {
    MEASURE_WIRELESS_BANDWIDTH();
    MEASURE_PACKET_ROUND_TRIP_TIME();
    CALCULATE_OPTIMAL_RECEIVER_BUFFER_SIZE();
    if IS_OUT_OF_SEQUENCE() then
        if Sbuffer < Scalc(i) then
            Scalc(i) = Sadjust(i) = Sbuffer;
        endif;
    endif;
        if Sadjust(i-1) < Scalc(i) then
            Sadjust(i) = Sadjust(i-1) + Spacket;
        else
            Sadjust(i) = Sadjust(i-1) - Spacket;
        endif;
    endif;
    SEND_ACK();
}
```

그림 4 Advertised 윈도우 조절 알고리즘

의 재전송을 필요로 하지 않는다. 그림 4는 수신자에서 수행되는 W_{ad} 조절 알고리즘을 보여준다. 알고리즘의 동작 과정은 다음과 같다. i번째 TCP 패킷 P_i가 도착하면 수신자는 수신된 패킷간의 간격을 이용하여 $B_{adjust(i)}$와 $BaseRTT_{calc(i)}$를 측정하고, 측정된 값을 기준으로 $S_{calc(i)}$를 계산한다. 수신된 패킷이 올바른 순서의 패킷이 아닌 경우 $S_{adjust(i)}$와 $S_{calc(i)}$를 S_{buffer}로 설정하고 송신자에게 알린다. 수신된 패킷이 올바른 순서의 패킷일 경우 이전에 흐름 조절이 적용된 Advertised 윈도우 $S_{adjust(i-1)}$를 $S_{calc(i)}$와 비교하여 $S_{adjust(i-1)}$가 $S_{calc(i)}$보다 작을 경우 $S_{adjust(i)}$를 패킷 크기 S_{packet}만큼 증가시키고, $S_{adjust(i)}$가 $S_{calc(i)}$보다 클 경우 $S_{adjust(i)}$를 S_{packet} 만큼 감소시킨다.

4. 실 험

다음은 논문에서 제안한 수신자에 의한 무선 상태 적응적인 TCP 흐름 제어 방법의 효율성을 검증하기 위한 실험 과정 및 결과를 기술한다.

4.1 실험 환경

본 논문에서 제안된 수신자에 의한 TCP 흐름 제어 기법은 리눅스 2.4.20 커널 코드를 수정하여 구현되었다. 수신자에 의한 TCP 흐름 제어 방식을 검증하기 위하여 구성한 시스템은 서버와 클라이언트로 구성된다. 서버는 인터넷에 연결된 데스크톱을 사용하고, 클라이언트는 제

안된 TCP 흐름 제어 방식이 구현된 리눅스가 탑재된 노트북을 사용하였다. 무선 넷트웍은 CDMA2000 1x를 사용하였으며 이를 위해 노트북에 CDMA2000 1x 지원 휴대폰을 연결하였다. 실험 자료의 분석을 위해서는 tcpdump [15], tcptrace[16] 프로그램을 사용하였다.

실험은 제안된 TCP 흐름 제어 기법이 올바로 구현되어 있는지 검증하는 실험과, 구현된 TCP 흐름 제어 기법의 성능을 기존 TCP와 비교하는 실험으로 구분하여 실험하였다. 실험에서 사용한 TCP 패킷의 크기는 이더넷에서 패킷의 단편화가 이루어지지 않는 최대 크기인 1448 Bytes를 사용하였으며, 1 MByte의 자료를 전송하는 실험을 수행하였다. 수신자의 버퍼 크기는 수신자가 제한된 메모리를 가진 환경을 위해 실험에 사용된 TCP 패킷 크기인 1448 Bytes의 두 배인 2896 Bytes와 시스템에서 사용 가능한 최대 버퍼 크기인 64 KBytes의 두 가지로 구분하여 실험하였다. 기존 TCP에서는 패킷 왕복 시간 보정시 이전에 측정된 정보를 90%를 반영하고 새로운 정보를 10% 반영하는 것이 효율적인 것으로 알려져 있다[12]. 이에 기반하여 본 실험에서는 측정된 순간 무선 대역폭 $B_{calc(i)}$를 보정하기 위한 보정 상수 α 값으로 0.9를 사용하였다.

4.2 구현된 흐름 제어 기법의 검증

논문에서 제안하는 흐름 제어 기법은 무선 대역폭과 종단간 패킷 왕복 시간을 동적으로 측정하며, 측정된 정보를 가지고 무선 상태 변화에 따라 Advertised 윈도우를 적절히 조절함으로써 TCP 흐름 제어를 수행한다. 구현된 흐름 제어 기법의 검증을 위해 무선 대역폭의 측정, 종단간 패킷 왕복 시간의 측정, Advertised 윈도우 조절 그리고 흐름 제어로 인한 TCP 전송률 조정이 올바로 동작하는 있는지를 확인하였다.

그림 5는 수신자에 의한 측정된 순간 무선 대역폭 $B_{calc(i)}$과 보정된 무선 대역폭 $B_{adjust(i)}$의 비교를 보여준다. $B_{calc(i)}$는 상당히 급변하는 반면, $B_{adjust(i)}$는 다소 안정된 값을 갖는다. 대부분의 경우에서 $B_{calc(i)}$가 이상적인 실제 무선 대역폭에 비해 다소 적은 값을 유지하는 것을 볼 수 있다. 이는 $B_{calc(i)}$가 프로토콜 처리 부하 등의 이유로 실제 무선 대역폭보다 작아지기 때문이다. $B_{calc(i)}$가 이상적인 실제 무선 대역폭보다 큰 경우는 오류 특성이 있는 무선 환경에서 넷트웍 하위 계층의 오류 복구 방식에 의해 버퍼링된 패킷들이 한번에 상위 계층에 전달되기 때문이다. $B_{calc(i)}$가 실제 무선 대역폭보다 큰 폭으로 작아진 경우는 무선 환경에서의 패킷 오류 및 손실로 발생한 재전송 때문이다. 그림 5의 실험 결과에 의해 구현된 수신자에 의한 무선 대역폭 측정 방법이 올바로 동작함을 알 수 있다.

그림 6은 수신자에서 측정된 종단간 패킷 왕복 시간 $RTT_{calc(i)}$를 보여준다. 송신자에서 측정한 종단간 패킷 왕복 시간과 수신자에서 측정한 $RTT_{calc(i)}$와의 차이가 거의 없음을 알 수 있다. 이는 수신자에서 수행되는 종단간 패킷 왕복 시간 측정 방법이 올바로 동작함을 입증한다. $RTT_{calc(i)}$가 큰 폭으로 증가한 것은 무선 환경에서의 패킷 오류나 손실로 인해 발생한 패킷의 재전송 때문이고 $RTT_{calc(i)}$가 송신자가 측정한 종단간 패킷 왕복 시간보다 큰 폭으로 크게 측정된 것은 서버의 전송률이 Advertised 윈도우 W_{ad}가 아니라 혼잡 윈도우 W_{cn}에 의해 제한 받았기 때문이다.

그림 7은 기존 TCP 방식과 논문에서 제안한 흐름 제어 방식에 대해 수신자 버퍼 크기에 따른 송신자의 전송률 변화를 보여준다. 송신자가 종단 넷트웍 대역폭보다 작은 전송률로 자료를 전송할 경우는 충분한 TCP 전송 성능을 얻을 수 없고, 송신자가 종단 넷트웍 대역폭 이상으로 자료를 전송할 경우는 종단 라우터의 버퍼링 부하를 가중시키어 혼잡 상황을 유도하게 된다. 그러므로 넷트웍이 허용하는 대역폭으로 TCP 전송률을 조정하는 것이 필요하다. 그림 7에서 기존 TCP의 경우, 수신자의 버퍼 크기가 2896 Bytes일 때 송신자의 전송률이 수신자 버퍼 크기 제한으로 낮아지는 것을 볼 수 있다. 수신자의 버퍼 크기가 64 KBytes 일때는 송신자의 전송률이 수신자 버퍼 크기에 제한을 받지 않아 무선 대역폭 이상으로 전송률이 높아지는 것을 볼 수 있

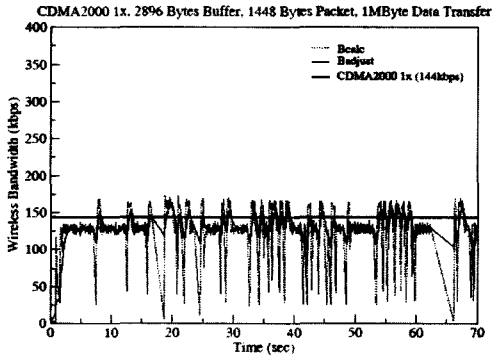

그림 5 수신자에 의한 무선 대역폭 측정 결과

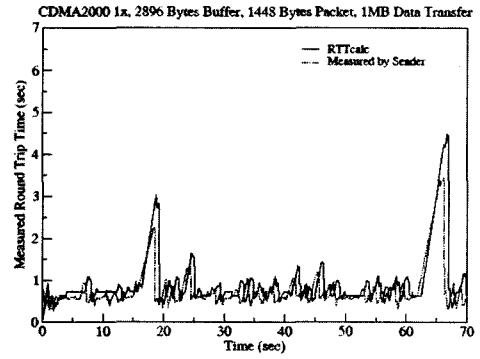

그림 6 수신자에 의한 종단간 패킷 왕복 시간 측정 결과

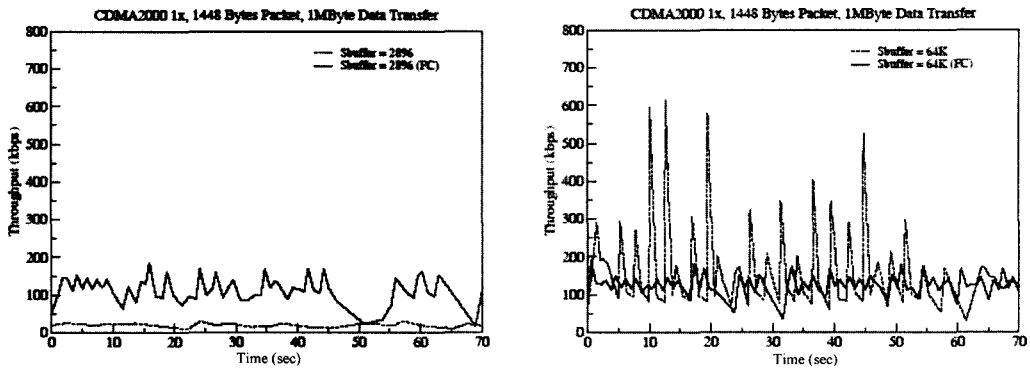

그림 7 송신자의 전송률 변화 (FC:제안된 방법)

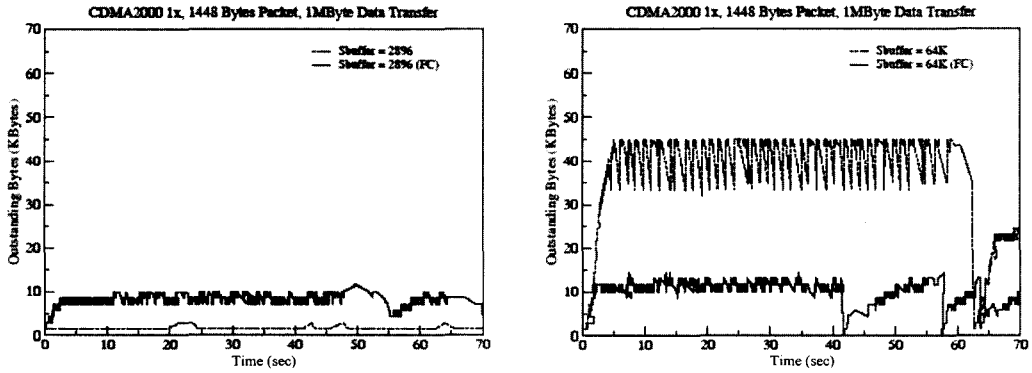

그림 8 Outstanding Bytes 변화

다. 흐름 제어를 수행할 때에는 수신자 버퍼 크기에 상관없이 CDMA2000 1x망의 이상적인 대역폭인 144kbps에 근접하도록 송신자의 전송률이 조정됨을 볼 수 있다. 이 실험을 통해, 제안된 수신자에 의한 TCP 흐름 제어 방식이 종단 넷트웍의 대역폭에 맞게 전송률을 조정하도록 흐름 제어를 수행하며, 구현된 흐름 제어 방식이 올바로 동작함을 알 수 있다.

그림 8은 기존 TCP 방식과 논문에서 제안한 흐름 제어 방식에 대해 수신자 버퍼 크기에 따른 Outstanding Bytes 변화를 보여준다. Outstanding Bytes는 송신자에서 수신자의 응답 메시지를 받지 않고 넷트웍으로 전송 가능한 자료의 양을 의미한다. 기존 TCP의 경우, 수신자 버퍼 크기가 2896 Bytes일 때 수신자 버퍼 크기 제한으로 Outstanding Bytes가 제한된 것을 볼 수 있다. 수신자 버퍼 크기가 64 Kbytes인 경우는 송신자의 전송률이 수신자 버퍼 크기에 제한 받지 않으므로 Outstanding Bytes는 커지게 된다. TCP 흐름 제어를 수행할 경우 수신자 버퍼 크기와 상관없이 Outstanding Byte를 약 10 Kbytes 내외의 일정 수준으로 유지하는 것을 볼 수 있다. 이는 제한된 메모리를 가지고 있는 수신자를 고려한 Advertised 윈도우 조절이 적절히 수행되었음을 의미한다.

그림 9는 기존 TCP 방식과 논문에서 제안한 흐름 제어 방식에 대해 수신자 버퍼 크기에 따른 송신자에서 측정된 종단간 패킷 왕복 시간을 보여준다. 기존 TCP의 경우, 수신자 버퍼 크기가 2896 Bytes일 때에는 송신자의 전송률이 낮아서 중간 라우터에서의 버퍼링 지연 부하가 적어지므로 종단간 패킷 왕복 시간이 작다. 수신자 버퍼 크기가 64 KBytes일 때에는 송신자가 종단 넷트웍 대역폭 이상으로 자료를 전송하게 되며, 종단 라우터의 버퍼링 부하를 증가시키어 종단간 패킷 왕복 시간이 증가함을 볼 수 있다. 흐름 제어를 수행할 경우, 종단 넷트웍 대역폭으로 송신자의 전송률을 조정함으로써 종단 라우터의 버퍼링 부하를 감소시켜 전체적인 종단간 패킷 왕복 시간이 기존 TCP의 수신자 버퍼 크기 2896 Bytes일 때와 비슷한 것을 볼 수 있다. 종단간 패킷 왕복 시간이 급격히 증가하는 것인 패킷 손실로 인한 재전송으로 발생한 것이다.

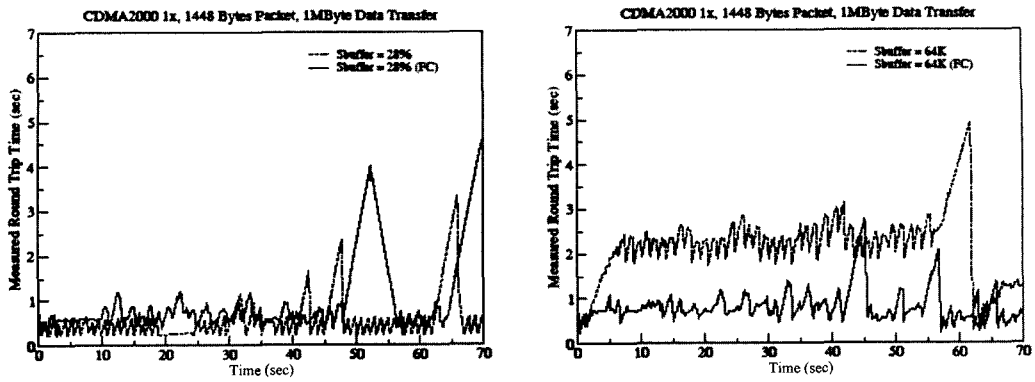

그림 9 종단간 패킷 왕복 시간 변화

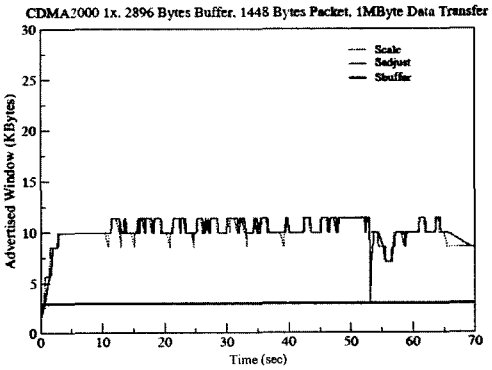

그림 10 TCP 흐름 제어

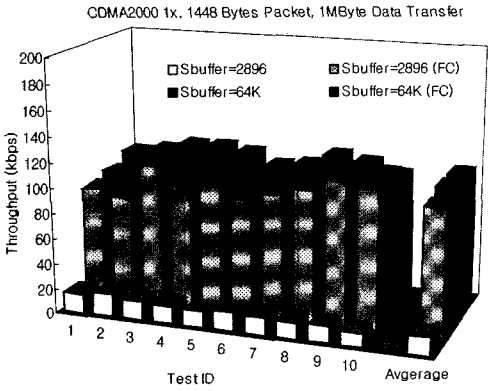

그림 11 전송률 비교

그림 10은 논문에서 제안된 TCP 흐름 제어 과정을 보여준다. 초기 연결 설정 단계에서 수신자에서 송신자에게 전달하는 Advertised 윈도우의 크기 S_{adjust}는 수신자에서 할당 가능한 메모리 크기 S_{buffer}로 설정된다. 수신자는 자료를 수신할 때마다 무선 넷트웍 대역폭과 패킷 왕복 시간을 동적으로 측정하여 최적의 버퍼 크기 $S_{calc(i)}$를 계산한다. $S_{calc(i)}$는 무선 넷트웍의 상태에 따라 가변적이며 보정된 무선 넷트웍 대역폭 $S_{adjust(i)}$는 급격한 $S_{calc(i)}$에 비해 완만한 변화를 보이고 있고 시간이 지날수록 $S_{calc(i)}$에 수렴함을 볼 수 있다. $S_{adjust(i)}$가 S_{buffer}로 급격히 떨어진 것은 수신자가 순서가 올바르지 않은 패킷을 수신하였을 때이며, 추가의 패킷 재전송 부하를 감소시키기 위해서이다. 이 실험을 통해 제안한 TCP 흐름 제어 방식이 올바로 구현되었음을 알 수 있다.

4.3 제안한 흐름 제어 기법의 성능 실험

구현된 수신자에 의한 TCP 흐름 제어 기법의 성능 실험을 위해 총 10회의 실험을 하였다. 10회의 실험을 통해 전송률, 종단간 패킷 왕복 시간, Outstanding Bytes를 비교함으로써 제안한 방식의 성능을 비교 분석하였다.

그림 11은 10회 실험에 대한 각각의 전송률과 평균을 보여준다. 실험 환경의 무선 넷트웍는 CDMA2000 1x으로 144kbps의 대역폭을 가진다. 기존 TCP의 경우, 수신자 버퍼가 2896 Bytes일 때는 전체적인 전송률이 낮은 것을 볼 수 있다. 이는 수신자 버퍼 크기가 작음으로 인해 송신자의 전송률이 수신자 버퍼 크기에 의해 제한을 받기 때문이다. 수신자의 버퍼가 64 Kbytes일 때는 송신자의 전송률이 수신자 버퍼 크기의 제한을 받지 않으므로 높은 전송률을 보인다. 수신자의 버퍼 크기가 클 경우 송신자의 자료 전송률은 수신자 버퍼 크기에 영향을 받지 않으므로 한정된 대역폭을 최대한 사용할 수 있으나 과도한 자료 전송으로 인한 종단 라우터의 버퍼링 부하를 증가시킬 수 있다. 흐름 제어를 수행할 경우, 수신자 버퍼 크기에 상관없이 모두 높은 전송률을 보인다. 흐름 제어를 수행함으로써 수신자 버퍼 크기가 작음에도 불구하고 한정된 무선 대역폭의 근접하게 전송률이 조정됨을 알 수 있다. 이는 본 논문에서 제안된 수신자에 의한 흐름 제어 방법이 실제 단말기의 버퍼 크기 이상의 Advertised 윈도우를 송신자에게 알림으로써 수신자 버퍼 크기에 의한 송신자의 전송률이 종단 넷트웍

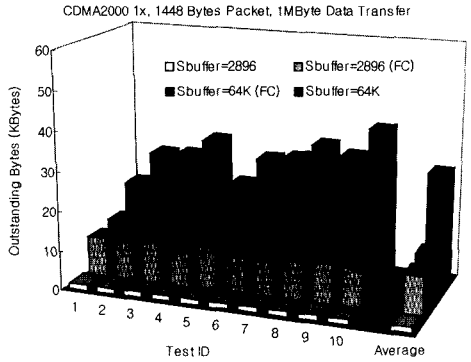

그림 12 Outstanding Bytes 비교

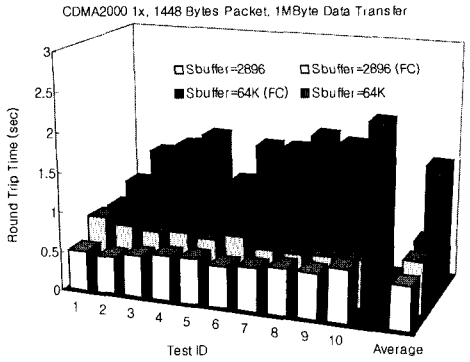

그림 13 종단간 패킷 왕복 시간 비교

대역폭 이하로 떨어지지 않도록 하기 때문이다.

그림 12는 10회 실험에 대한 각각의 Outstanding Bytes와 평균을 보여준다. Outstanding Bytes는 송신자가 수신자의 응답 없이 넷트웍으로 전송 가능한 자료의 양을 의미하며, 평균 Advertised 윈도우의 크기라 볼 수 있다. 그림 12에서 논문에서 제안한 TCP 흐름 제어를 수행할 경우, 수신자 버퍼 크기에 상관없이 Outstanding Bytes는 10 Kbytes 내외로 조정되는 것을 알 수 있다. 이는 제안된 흐름 제어 방법이 수신자 버퍼 크기에 상관없이 효율적인 Advertised 윈도우 크기를 계산하여 송신자에게 알리기 때문이다. 기존 TCP의 경우, 수신자 버퍼 크기가 2896 Bytes일 때는 송신자의 자료 전송량이 수신자의 버퍼 크기에 의해 제한받음을 알 수 있다. 수신자 버퍼 크기가 64 KBytes일 때는 송신자의 자료 전송량이 수신자의 버퍼 크기에 제한을 받지 않으나 과도한 자료 전송으로 인해 종단 라우터의 버퍼링 부하를 증가시켜 혼잡 상황을 유발할 수 있다.

그림 13은 10회 실험에 대한 각각의 평균 종단간 패킷 왕복 전송 시간과 그 평균을 보여준다. 기존 TCP의 경우, 수신자 버퍼 크기가 64KB일 때는 종단간 패킷 왕복 시간이 매우 커지는 것을 볼 수 있다. 패킷 왕복 시간의 증가는 수신자가 종단 넷트웍 대역폭 이상으로 자료를 전송하여 종단 라우터에서의 버퍼링 시간이 증가에 의한 지연 시간이 발생했기 때문이다. 라우터에서의 버퍼링 부하 증가는 혼잡 상황 발생 확률을 증가시키고 패킷 전송 시간 초과를 유도해 TCP 성능상의 저하를 가져올 수 있다. 수신자 버퍼 크기가 작은 경우는 송신자의 전송률이 수신자 버퍼 크기에 제한을 받아 충분한 전송 성능을 얻을 수 없는 반면, 종단 라우터에서의 버퍼링 부하를 발생시키지 않아 종단간 패킷 왕복 시간은 짧게 된다. 흐름 제어를 수행하는 경우, 버퍼 크기에 상관없이 상대적으로 낮은 패킷 왕복 시간을 갖는다. 이는 흐름 제어 방법이 종단 넷트웍 대역폭에 맞도록 송신자의 전송률을 조정함으로써 전송 성능의 향상과 함께 종단 라우터의 버퍼링 부하를 감소시키기 때문이다.

4. 결 론

본 논문에서는 수신자에 의한 무선 상태 적응적 TCP 흐름 제어 방법을 제안하였다. 수신자에 의한 TCP 흐름 제어 기법은 수신자에서 수행되며 송신자에게 전달하는 Advertised 윈도우의 크기를 종단 무선 넷트웍의 대역폭으로 제한하여 송신자의 전송률을 제한함으로써 수행된다. 수신자는 무선 상태 감지를 위해 무선 대역폭과 종단간 패킷 왕복 시간을 측정하며 최적의 Advertised 윈도우 크기를 동적 계산한다. 또한, 수신자의 한정된 버퍼 크기로 인해 송신자의 전송률이 제한 받지 않도록 실제 수신자의 버퍼 크기보다 큰 Advertised 윈도우를 송신자에게 전달할 수 있도록 하고 무선 환경의 상태 변화에 따른 효율적인 Advertised 윈도우 조정으로 TCP 전송 성능을 높이고 종단간 패킷 왕복 시간을 줄이도록 한다.

제안된 수신자에 의한 TCP 흐름 제안 방법은 실제 시스템으로 구현되었으며, 실험을 통해 그 성능을 검증하였다. 흐름 제어 기법을 사용했을 경우 수신자 버퍼 크기와 상관없이 무선 대역폭에 근접하는 전송 성능을 얻을 수 있었다. 흐름 제어 기법을 사용하고 수신자 버퍼 크기가 2896 Bytes인 경우, 흐름 제어를 하지 않았을 경우보다 약 5 배의 전송률 향상을 보였으며 수신자의 버퍼가 64 KBytes 일 경우와 비슷한 전송률을 보였다. 흐름 제어 기법을 사용하고 수신자 버퍼 크기가 64 KBytes인 경우, 흐름 제어를 수행하지 않았을 경우보다 종단간 패킷 왕복 시간은 반 이하로 줄일 수 있었으며, 수신자 버퍼가 2896 Bytes일 경우와 비슷하였다. 이로써, 제안된 수신자에 의한 TCP 흐름 제어 방법이 수신자가 제한된 자원을 가지고 있음에도 불구하고 무선 환경에 효율적인 TCP 흐름 정책으로 TCP 전송 성능 향상과 종단간 패킷 왕복 시간 감소를 가져온다는 것을 실험적으로 증명하였다. 종단간 패킷 왕복 시간의 감소

는 종단 라우터의 버퍼링 부하를 감소시켜 혼잡 상황 발생 확률을 줄이고, TCP 패킷 전달 초과 시간(RTO)을 감소시켜 TCP 송신자가 넷트웍 변화에 빠르게 반응하도록 한다.

본 논문에서 제안한 수신자에 의한 TCP 흐름 정책은 제한된 시스템 자원을 가지고 있는 휴대 단말기, PDA 등에 사용될 수 있으며 무선 환경뿐만 아니라 서로 다른 넷트웍이 연동된 환경, 특히 종단간 패킷 시간이 긴 위성 통신이나 국가간 통신 등에 적용 가능하다. 제안된 수신자에 의한 TCP 흐름 정책은 기존 TCP의 종단간 작동 방식을 유지하며 송신자와 중간 라우터의 변경을 필요로 하지 않고 수신자의 변경만으로 적용이 가능하다. 따라서, 수신자의 변경만으로 기존의 넷트웍 환경에서 사용되고 있던 여러 응용 프로그램들을 그대로 사용할 수 있으며, 수신자에 의한 효율적인 TCP 흐름 제어로 부가적인 성능의 향상을 이룰 수 있다.

향후 연구 과제로는 수신자에서 측정되는 무선 넷트웍 대역폭과 패킷 왕복 시간이 초기 연결 설정 단계에서 정확하지 못한 것을 개선하는 방법에 대한 연구가 있다. 또한, 정확한 무선 상태 감지를 위한 방법에 대한 연구와 무선 상태 정보를 바탕으로한 송신자에서의 효율적인 대역폭 분배 및 흐름 조절에 관한 연구가 있다.

참 고 문 헌

[1] L. Garber, "Will 3G really be the next big wireless technology?," *IEEE Computer*, Vol. 35, No. 1, pp.26~32, January 2002.

[2] C. Barakat, E. Altman, W. Dabbous, "On TCP Performance in a Heterogeneous Network: A Survey," *IEEE Communications Magazine*, Vol. 38, No. 1, pp.40~46, January 2000.

[3] V. Jacobson, "Congestion Avoidance and Control," *In Proceedings of ACM SIGCOMM*, pp.314~329, 1998.

[4] Xiaoming Zhou, Xichen Liu, John S. Baras, "Flow Control at Satellite Gateways," *Technical Reports, University of Maryland*, CSHCN TR 2002-19, 2002.

[5] Y. Bai, A. T. Ogielski, G. Wu, "Interactions of TCP and radio link ARQ protocol," *In Proceedings of 50th IEEE Vehicular Technology Conference (VTC)*, pp.1710~1714, September 1999.

[6] M. Ivanovich, R. Mukhtar, S. Hanley, H. Vu, P. Fitzpatrick, "Analysis of TCP Performance over Hybrid 'Fast Fixed-to-Slow Wireless' Buffered Links," *In Proceedings of IEEE Globecom*, Vol. 3, pp.1816~1820, San Antonio, November 2001.

[7] Neil T. Spring, Maureen Chesire, Mark Berryman, Vivek Sahasranaman, Thomas Anderson, Brian Bershad, "Receiver based management of low bandwidth access links," *In Proceedings of IEEE INFOCOM*, pp.245~254, 2000.

[8] L. Kalampoukas, A. Varma, K. Ramakrishnan, "Explicit window adaptation: a method to enhance TCP performance," *In Proceedings of IEEE INFOCOM*, pp.242~251, April 1998.

[9] J. Aweya, M. Ouellette, D.Y. Montuno, Z. Yao, "Enhancing network performance with TCP rate control," *IEEE Global Telecommunications Conference (GLOBECOM)*, Vol.3, pp.1712~1718, 2000.

[10] Mike Fisk, Wu-chun Feng, "Dynamic Right-Sizing in TCP," *In Proceedings of the Los Alamos Computer Science Institute Symposium (LACSI)*, Santa Fe, New Mexico, October 2001.

[11] L. Andrew, S. Hanly, R. Mukhtar, "Differentiated Capacity Allocation for End-to-End Connections with a Single Bottleneck Link," *In Proceedings of IEEE INFOCOM*, April 2003.

[12] W. R. Stevens, *TCP/IP Illustrated*, Volume 1, Addison Wesley, Massachusetts, 1994.

[13] Brian L. Tierney, "TCP Tuning Guide for Distributed Applications on Wide Area Networks," *Usenix ;login: Journal*, Vol. 26, No. 1, pp.33~39, February 2001.

[14] D. Chiu, R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN*, Vol. 17, No. 1, pp.1~14, June 1989.

[15] V. Jacobson, C. Leres, S. McCanne, *tcpdump*, URL : http://www.tcpdump.org, 1989.

[16] S. Ostermann, *tcptrace*, URL : http://www.tcptrace.org, 1994.

이 종 민

1999년 광운대학교 컴퓨터과학 학사
2001년 광운대학교 컴퓨터과학 석사
2003년~현재 연세대학교 컴퓨터과학과 박사과정. 관심분야는 멀티미디어 시스템, 운영체제, 컴퓨터 네트웍

차 호 정

1985년 서울대학교 컴퓨터공학 학사
1987년 서울대학교 컴퓨터공학 석사
1991년 University of Manchester 전산학 박사. 1993년~2001년 광운대학교 컴퓨터과학과 부교수. 2001년~현재 연세대학교 컴퓨터과학과 교수. 관심분야는 멀티미디어 시스템, 운영체제, 내장형시스템