論文2002-39CI-4-5

QoS를 이용한 인터넷 원격제어의 임의 시간지연 해결 방법

(A Solution for Random Time-delay in Teleoperation via Internet using QoS)

許 慶 茂 *, 沈 鉉 承 **

(Kyung-Moo Huh and Hyun-Seung Shim)

요 약

본 논문에서는 인터넷을 이용한 원격제어시 문제점으로 지적되는 동적으로 불규칙하게 변화하는 예측불가능한 시간지연을 해결할수 있는 방법을 제시하고, 이를 적용한 실험결과를 보인다. 즉 원격제어 대상 시스템에 요구되는 일정한 대역폭을 확보하여 불규칙적으로 변화하는 시간지연을 정적으로 예측가능하도록 고정시키고 실시간성을 확보함으로써, 인터넷을 이용한 원격제어시 시간지연 문제를 해결할 수 있음을 보인다. 또한 전송률을 일정하게 유지함으로써 원격제어시 지연시간을 설정한 시간내에 확실하게 처리할 수 있음을 보인다. 그리고 본 논문에서 제안한 방법과 기존의 TCP, UDP를 이용한 원격제어 방법을 각각 적용한 실험결과를 통해 비교함으로써, 제안한 방법의 우수성을 보인다.

Abstract

In this paper, we propose a solution for random time-delay in teleoperation via internet using QoS, and show the experimental results of our proposed method. By ensuring a constant bandwidth required for the specific telecontrolled system, we change irregular random time-delay to a predictable time-delay, and so we can control the system with the time-delay shorter than the specified time-delay. Through the experimental results, we show the effectiveness of our proposed method.

Ⅰ. 서 론

원격제어를 함에 있어 가장 먼저 해결해야 할 문제점은 시간지연의 문제점이다.[1,2] 시간지연 문제가 해결되지 않으면 원격제어 자체가 불가능하게 되어 원격제어의 만족할 만한 성능을 얻지 못하게 된다. 또한 시간지연이 동적으로 예측 불가능하게 변화하면 원격제어기를 불안정하게 만들어, 때에 따라서는 제어기 자체를 고장내는 원인이 되기도 한다.[3] 따라서 기존의 원격제어 방법에 있어서는 이 시간지연문제를 해결하기 위하여 다양한 방법을 연구하여 왔다. 대표적인 연구방법으로는 외란 관측기와 가상모델 등이 있다. 그러나 이러한 방법들의 기본가정은 시간지연이 예측 가능한 범위내에 있어야 한다는 것이다. 따라서 인터넷을 이용하여 원격제어를 구현할 때에는 이러한 방법들을 이용할수 없다. 왜냐하면 인터넷은 공유되어 있는 자원이기 때문이다. 즉, 사용자가 많아질수록 대역폭은 좁아지고 그로 인하여, 시간 지연은 동적으로 예측 불가능하게 불규칙적으로 변화하는 것이다.[4] 이는 원격제어에 치명적이

* 正會員, 檀國大學校 電子컴퓨터學部
(Dept. of Electronics & Computer Engineering dankook University)
** 學生會員, 檀國大學校 電子컴퓨터學部
(Dept. of Electronics & Computer Engineering dankook University)
※ 이 연구는 2001학년도 단국대학교 대학연구비의 지원으로 연구되었음.
接受日字:2001年10月8日, 수정완료일:2002年6月10日

다. 따라서 본 논문에서는 인터넷을 이용한 원격제어시스템을 구축하기 위하여 기존의 인터넷에서의 예측 불가능한 시간지연을 예측가능한 시간지연으로 바꾸는 방법에 대하여 논의하고, 실험하고자 한다. 이를 통해 기존의 원격제어에서 연구되어왔던 다양한 방법들의 기본 가정을 충족시킬 수 있는 기본적인 실험결과를 제공하고자 한다.

Ⅱ. 인터넷 이용 원격제어

인터넷의 범용적인 성공으로 인해 원격제어 또한 전격적으로 인터넷을 이용하게 되었다. 인터넷을 이용한 원격제어의 접근은 주로 프로토콜에 기반하여 이루어지게 된다.[4] 대표적인 프로토콜로는 TCP/IP 가 있다. 또한 실시간성을 갖기 위하여 최근에는 RTP/RTCP를 이용하여 인터넷 실시간 적용이 이루어지고 있다. 그동안 인터넷을 이용하여 원격제어를 하기 위하여 TCP, UDP, 그리고, RTP를 이용하여 원하는 목적을 달성하기 위하여 노력해왔으나 만족할만한 성능, 즉 실시간성을 얻지 못하였다. 그 이유로는 서론에서 언급한 바와 같이 시간지연이 동적으로 예측 불가능하게 변화하는 데에 있다. 그 근본적인 원인으로는 지금의 인터넷의 성공을 이끈 IP의 최대 노력(best effort) 서비스 모델에 기인한다. 최대 노력 서비스 모델이란 사용자, 적용처에 상관없이 어느 것이든 전송하기 위하여 최대한 노력한다는 것이다. 다시 말하면, 신뢰성과 실시간성의 보장없이 무조건 보내게 된다. 그리고 인터넷을 이루는 각 장비의 능력을 초과하는 과다한 전송이 요구될 시에는 무조건 패킷을 버리는 서비스를 말한다.[5] 이로 인하여 인터넷을 이용하는 사용자가 많아질수록 패킷은 버려지고, 시간지연은 끊임없이 예측불가능하고 불규칙적으로 변화하는 것이다. 이런 상황에서는 원격제어를 위한 실시간성과 안전성을 확보할 수 없으며, 제어 자체가 불가능하게 된다.

따라서 본 연구에서는 원격제어의 실시간을 확보하고 동적으로 변화하는 시간지연을 예측 가능하도록 하기 위하여, QoS(Quality of Service)가 적용된 인터넷을 이용한 원격제어 방법을 제시한다. QoS란 최대 노력 서비스 모델을 대체하는 서비스 모델로서 적용 서비스 질을 보장하는 서비스를 말한다.[6] 즉 특정 적용에 일정한 대역폭을 확보하여 동적으로 불규칙적으로 변화하는 시간지연을 정적으로 예측가능하도록 고정시키고 실시간성을 확보하는 서비스 모델을 말한다.

1. 서비스 모델의 확장

최대 노력 서비스 모델은 TCP/IP의 기본 모델로서 현재 인터넷의 성공을 이끈 요소중 하나이다. 이 모델의 기본적인 관점은 현재 존재하는 다양한 이기종의 네트워크가 서로 간에 최대한 결합할 수 있는 가능성을 제공하는 것과, 어떤 극한 상황에서도 호스트간의 연결가능한 경로를 제공하는 것에 그 목적이 있었다. 따라서 어느 곳에서 어떤 패킷이 들어오든지 간에 그 패킷을 목적지까지 보낼 수 있도록 최대한 시스템이 노력할 수 있도록 한다. 여기서 주의해야할 것은 최대 노력의 의미이다. 즉 최대 노력이란 시간지연에 대한 보장을 언급한 것은 아니다. 이 모델은 패킷이 목적지까지 가기 위한 가장 가까운 다음 홉(hop)으로 전송만을 보장할 뿐 일정한 시간내의 전송을 보장하지는 않는다. 즉, 시스템이 과부하시 이 모델을 제공하는 시스템은 계속해서 패킷을 버림으로써 최대한 노력을 할 뿐이다. 또한 모든 패킷을 동등하게 처리함으로써 다양한 패킷에 따른 선별적인 효율성을 제공하지도 않는다. 따라서 QoS는 표 1과 같이 최대 노력 서비스 외에 다른 서비스 모델을 제공하여 시스템의 효율성을 보장한다. 여기서 제어 부하 서비스(controlled load service) 모델이란 과부하가 아닌 상태에서의 최대 노력과 같은 역할을 하는 서비스 모델을 말하며,[7] 보장 서비스(guaranteed service) 모델은 제어 부하 서비스(controlled load service)외에 예측가능한 시간을 보장하여 실시간 기능을 제공하는 서비스 모델을 말한다.[8] 또한 각 서비스 모델의 하위 계층으로 다양한 템플릿을 제공하여 여러 가지 상황을 수용하도록 하고 있다.[9]

표 1. QoS의 서비스 모델 확장
Table 1. Extension of service model in QoS.

현재의 서비스 모델	QoS의 서비스 모델
최대 노력 서비스 (단일)	최대 노력 서비스
	제어 부하 서비스
	보장 서비스

2. QoS 구성

과거의 인터넷은 파일전송, 이메일과 같은 단순한 적용만이 존재하였다. 그러나 현재는 인터넷을 이용한 다양한 속성을 가지고 있는 적용이 나타나고 있다. 따라서 QoS는 이러한 적용에게 다양한 서비스 모델을 제공

하여 선택하도록 하고 있다. 또한 각각의 공통적인 성격에 대하여 표준화된 여러 가지 구성요소를 제공하여 이종간의 혼합이 가능하도록 하고 있다. 대표적인 표준화 집단으로는 DiffServ(differentiated service)와 IntServ(intergrated service)가 있다. DiffServ는 각 패킷에 우선 순위를 부여하여 과부하시 패킷의 효율성을 증대시키기 위한 표준화작업을 수행하고 있으며,[10] IntServ는 각 응용 및 시스템간의 QoS 체결을 위한 프로토콜과 응용의 카테고리를 구분하고, 그리고 QoS를 제공하기 위한 각 시스템 디바이스의 요구사항등에 대하여 표준화를 하고있다.[11]

IntServ에서 제정하고 있는 구성요소중 하나로 서로 다른 이기종 시스템간의 QoS 체결을 어떻게 할 것인가에 대한 RSVP 프로토콜이 있다. 즉 RSVP(Resource reSerVation Protocol)는 시스템간 자원확보를 위한 셋업 역할을 하는 프로토콜이다. RSVP를 이용한 시스템간의 셋업순서는 그림 1과 같다.[6,9,12]

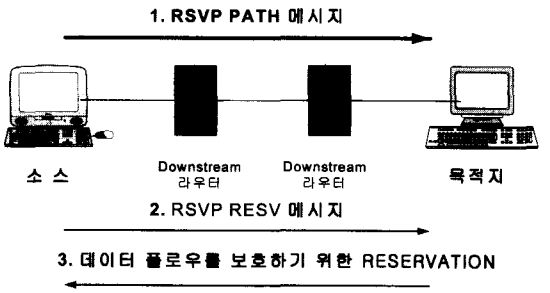


그림 1. RSVP의 시그널링 순서
Fig. 1. Signaling sequence of RSVP.

RSVP는 먼저 수신측까지 PATH 메시지를 보내서 경로를 결정한다. 또한 PATH 메시지에 데이타를 보내기 위해 필요한 자원의 정보를 함께 보낸다. 그러면 각각의 라우터는 자원의 정도가 현재의 상태에서 가능한지 확인한 후 PATH 메시지를 라우팅 테이블을 참고하여 수신측으로 보낸다. 수신측에서는 도착한 PATH 메시지에 포함되어 있는 자원정보를 이용하여 자원을 할당한 후 자신의 정보와 함께 송신측에게 RESV 메시지를 보낸다. RESV 메시지를 받은 라우터들은 RESV의 예약정보를 이용하여 자원을 할당하고, 메시지를 송신측에게 보낸다. 결국 송신측과 수신측간에 예약된 데이타 흐름이 형성되고 데이타를 전송하게 된다. 그리고 각각의 시스템들은 PATH와 RESV 메시지를 주기적으로 전송하여 PATH와 RESV 상태정보를 유지하여 예약된 자원을 유지하게 된다. 이를 소프트 상태 모델이라 한다.[6,9,12]

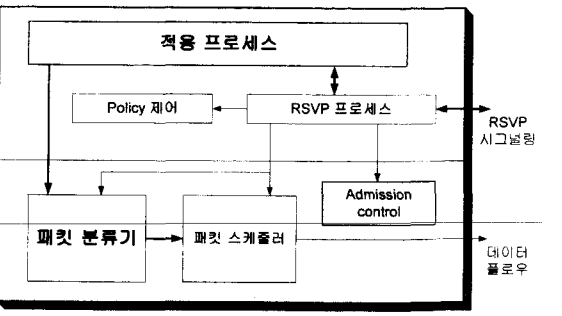


그림 2. 호스트에서의 RSVP 구성요소
Fig. 2. RSVP components on a host.

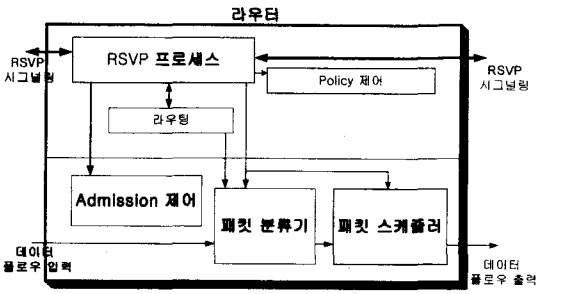


그림 3. 라우터에서의 RSVP 구성요소
Fig. 3. RSVP components on a router.

그림 2와 3은 각각 호스트와 라우터에서 필요한 RSVP 구성요소로서 RSVP 프로세스(그림 상단부분)와 트래픽 제어(그림 하단부분, TC)로 나뉜다. 이 중 트래픽 제어 부분의 성능에 따라 각 시스템의 성능이 결정

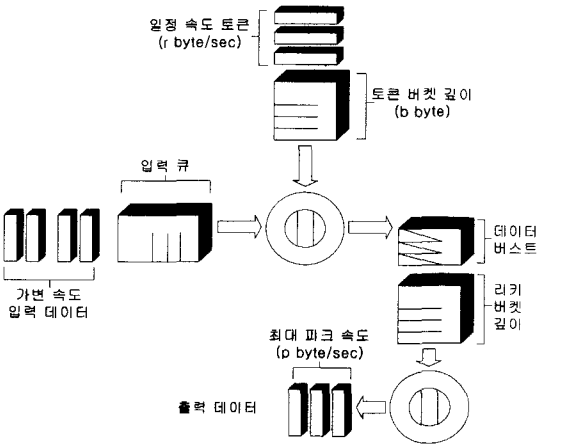


그림 4. 리키 버켓을 갖는 토큰 버켓
Fig. 4. Token bucket with leaky bucket.

된다. 트래픽 제어의 기본 성능을 결정하는 부분으로 패킷을 효율적으로 처리하는 큐잉 알고리즘이 있다. 큐잉 알고리즘에 따라 QoS는 자원예약의 정보를 구성한다. QoS에서 이용하는 기본 알고리즘은 그림 4와 같다. QoS는 그림 4의 큐잉 모델의 파라미터인 <b, r, p>와 M (maximum datagram size), 그리고 m(minimum size of packet)으로 예약정보를 구성한다. 즉, <b, r, p, M, m>이 QoS의 기본 파라미터가 된다. 이를 이용하여 각 시스템(호스트 포함)은 자원을 할당하여 지정된 대이타 플로우의 예약을 주기적으로 수행하게 된다.

3. 인터넷의 지연시간 계산

공유된 인터넷에서의 시간 지연은 임의 시간지연임을 알수 있다. 이러한 인터넷의 시간 지연은 다음과 같이 정의할 수 있다.

지연 시간 = 고정 지연시간 + 큐잉 지연시간
=(광속전파 지연시간 + 전송 지연시간)
+ 큐잉 지연시간 (1)

광속전파 지연시간
=거리/광속 (2)

전송 지연시간
= 패킷 크기/대역폭 (3)

큐잉 지연시간=nondeterministic (4)

시간지연으로는 첫번째 광속전파지연이 있다. 이것은 광속에 의한 시간지연으로 원격조정의 두 지점간의 거리를 알고 있다면 계산 할 수 있다. 광속은 진공에서는 3.0*(10**8)m/s, 케이블에서는 2.3*(10**8)m/s, 광섬유에서는 2.0*(10** 8)m/s로 전달된다. 두번째 전송지연시간는 단위 데이터를 전송하는 데 걸리는 시간이다. 패킷 교환기가 외부 링크로 패킷을 전달하기 전에 얼마동안 저장해야 되기 때문에 네트워크 내부에서 큐잉지연이 있을수 있다. 따라서 광속전파지연시간과 전송지연시간은 계산가능하고 데이터 전송 중 고정된 항목임을 알수 있다. 그러나 세번째 시간지연 항목인 큐잉지연은 현재 네트워크 자원에 접속중인 사용자에 의해 결정되므로 계산이 불가능하다. 따라서 인터넷을 이용한 원격제어에서 시간지연 예측시 시간지연은 큐잉지연에 의하여 변화함을 알 수 있다. 본 연구에서는 기존의 인터넷의 최대 노력 서비스 모델을 QoS 모델로 확장하였을시 큐잉지연으로 인한 임의 시간지연이 측정 및 계산가능한 고정된 시간지연으로 확보될 수 있음을 보이고자 한다.

네트워크에서 여러 사용자가 공유하는 자원으로는 링크의 대역폭과 버퍼가 있다. 버퍼는 링크를 통해 전달되기를 기다리는 패킷이 큐로 보관되는 장소이다. 패킷은 링크를 통해서 전달될 차례를 기다리며 버퍼에 보관되는데, 링크를 사용하기 위해서 라우터에서 경쟁하기도 한다. 특히 많은 패킷이 하나의 링크를 사용하기 위해서 경쟁하는 경우 큐는 그 즉시 포화상태가 될 수 있으며 일정수의 패킷은 전달이 취소되기도 한다. 이러한 상태를 혼잡 상태라하며, 큐잉지연은 바로 이 상황에서 발생한다. 따라서 QoS는 기존의 최대 노력 서비스 모델을 확장하여 자원의 할당과 다양한 스케줄링 알고리즘을 이용하여 혼잡상태를 해결하며 보장된 자원의 할당하에서 파라미터를 통하여 시간지연을 계산 가능하도록 한다.

큐잉지연을 계산화하기 위해서는 우선 패킷을 처리하기 위한 버퍼에서의 패킷의 처리속도를 살펴보아야 한다. 그림 5는 지연시간 계산을 단순화하기 위하여 그림 4의 큐잉의 버퍼모델중 토큰 버켓 모델만을 도시해 놓은 그림이다. 토큰 버켓 모델이란 VBR(variable bit rate)로 들어오는 패킷들을 자원의 낭비없이 수용 가능토록 설계된 버퍼 모델이다. 그림 5에서와 같은 모델에서 패킷의 최대 처리시간은 식 (6)과 같다.

최대 처리시간 = b/r (sec) (6)

최대 처리량 = (b + r*t) (7)

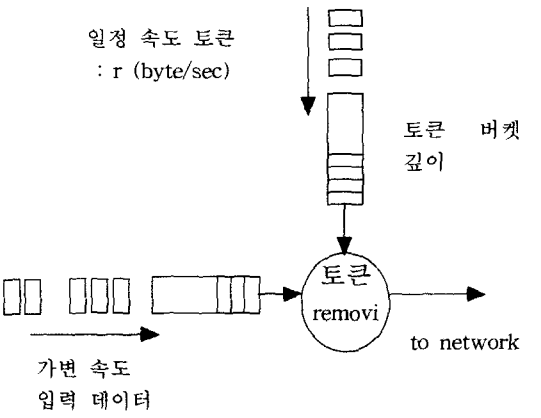


그림 5. 단순 토큰 버켓 모델
Fig 5. Simple token bucket model.

두번째로 살펴보아야 할 사항은 버퍼에 저장되어 있는 패킷을 처리하는 큐잉 알고리즘이다. 큐잉 알고리즘

으로는 다음의 네가지 방법이 있다.

① FIFO : FIFO 큐잉의 개념은 라우터에 처음 도착한 패킷이 맨 먼저 전송된다. 이것은 패킷이 얼마나 중요한지 또는 어느 흐름에 속하는지에 상관없이 이루어지므로 종단 노드에게 혼잡제어에 대한 모든 책임을 미룬다.

② 우선 순위 FIFO : 각 패킷마다 우선 순위를 부여하여 이 우선 순위 계층에 따라 다중 큐를 구현하여 우선 순위가 높은 FIFO 큐의 패킷이 먼저 전송되도록 한다. 이 방법의 문제점은 모든 발신지가 패킷에게 높은 우선 순위를 부여하는 것을 어떻게 막을 수 있는가이다.

③ FQ(Fair Queueing) : 각 발신지별로 독립적인 큐를 운용하는 알고리즘이다. 따라서 어떤 한 발신지가 다름 흐름에게 피해를 주면서 네트워크 용량에서 자신이 할당 받은 용량을 임의로 증가시키지 못하게 된다.[13]

④ WFQ(Weighted Fair Queueing) : WFQ는 FQ의 변형으로 각 흐름에 어떤 가중값이 할당되는 방식이다. 이 가중값은 논리적으로 라우터가 해당 큐를 서비스할 때 얼마나 많은 비트가 전송되어야 하는가를 지정해준다. 이는 곧 해당 흐름이 할당 받아야 하는 링크의 대역폭 비율을 효과적으로 조절하게 된다.

이 중 QoS는 WFQ를 기본으로 큐잉 알고리즘을 선택하여 계산하고 있다. 그림 6은 기본적인 WFQ 의 다

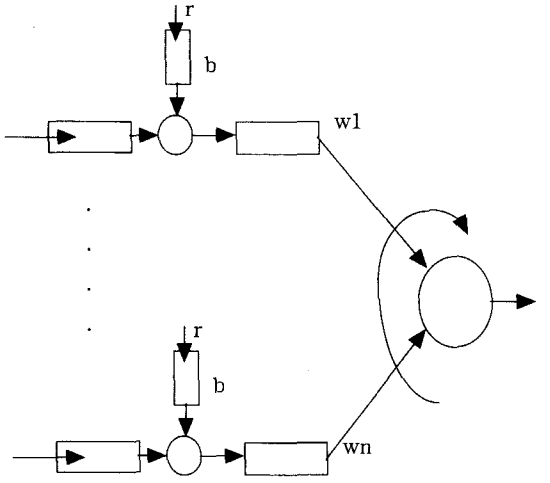


그림 6. WFQ 스케줄링에 의한 n-multiplexed 토큰 버켓 플로우

Fig. 6. N-multiplexed token bucket flows with WFQ scheduling.

이어그램을 보이고 있다. 이 그림에서의 최대 시간지연은 식 (8)과 같이 계산되어질 수 있다.

$$d_{max} = \frac{bi}{(C * \frac{Wi}{\Sigma Wj})} \tag{8}$$

(dmax는 최대시간 지연(sec), bi는 버켓 깊이 (bytes), C는 링크 용량 (bytes/sec), w는 가중치, i는 현재 보내는 클래스, Σw는 전체 클래스의 가중치를 말한다.)

여기서 WFQ를 이용하여 QoS를 지원시 QoS의 모든 서비스 모델을 지원하는 간략화한 WFQ 모델은 그림 7과 같다.[14] 그림 7을 보면 보장 서비스는 WFQ에 의해 독립적으로 관리되어지며, 제어 부하 서비스는 최대 노력 서비스보다 우선 순위가 높은 패킷으로 분리되어 다루어지고 있음을 알 수 있다. 따라서 QoS를 지원하는 각 라우터에서의 최대 지연시간은 식 (9)와 같다.

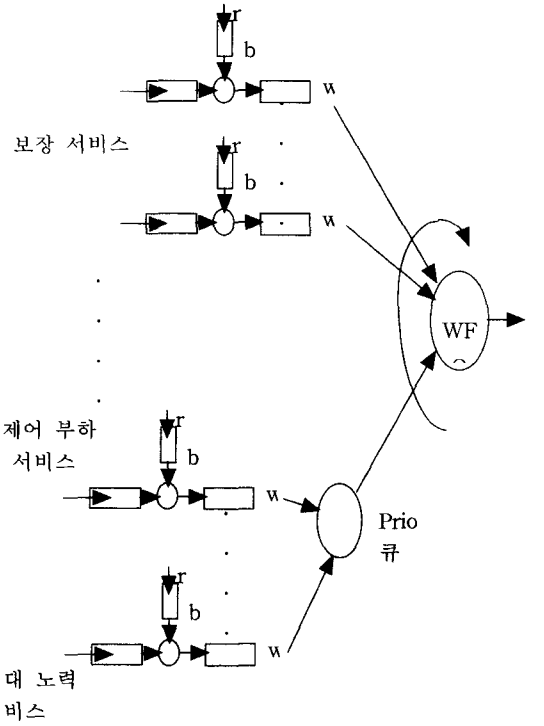


그림 7. QoS 모델을 적용한 WFQ 스케줄링

Fig. 7. WFQ scheduling adapted for QoS model.

$$d_{max} = \frac{bi}{ri} + \frac{bi}{(C * \frac{Wi}{\Sigma Wj})} \tag{9}$$

위의 식 (9)는 WFQ에서의 시간지연 항목을 계산한 것이다. QoS는 식 (9)를 더 일반화하여 시간 지연을 식

(10), (11)과 같이 계산한다.[8]

$$dmax = \frac{b}{R} + \frac{Ctot}{R} + Dtot \quad (\text{if } p>R>=r) \quad (10)$$

$$dmax = \frac{M}{R} + \frac{Ctot}{R} + Dtot \quad (\text{if } R>=p>=r) \quad (11)$$

여기서 b는 버켓 깊이(바이트), M는 최대 전달 단위(바이트), R은 수신자측에서 요구하는 데이터 rate r이다. 그리고 Ctot와 Dtot는 각 네트워크 요소에서 발생하는 오차 성분으로서, Ctot는 rate에 의존하는 오차 성분들의 총합이며 Dtot는 rate에 의존하지 않는 오차 성분들의 총합이다. 오차 성분인 Ctot과 Dtot은 계산되어 전체 dmax의 이탈 항목으로 처리된다. 그 다음으로 중요한 항목은 각각의 라우터에서의 시간지연을 어떻게 양쪽의 종단 호스트로 전달하는 가이다. 전달하는 방법은 앞에서 살펴본 바와 같이 RSVP 프로토콜을 이용하여 전달한다. 이를 위해 QoS는 각 파라미터들을 표2와 같이 정형화하였다.[8] 여기서 S는 슬랙 성분으로서 각 라우터에서 Ctot 과 Dtot를 계산하여 S에 저장하여 전송되어진다. 따라서 RSVP는 각 객체들을 각 서비스별로 정의하여 자원예약을 수행하게 된다

표 2. QoS 사양과 파라미터
Table. 2. specification and parameters of QoS.

QoS 사양	파라미터	전송방향
TSpec (트래픽 사양)	<b,r,p,m,M>	발신자 -> 수신자
RSpec (수신자 TSpec)	< R, S >	수신자 -> 발신자

4. 전체 시스템 구성

본 연구에서는 인터넷을 이용한 원격제어시 문제점인 시간지연을 해결하기 위하여 QoS를 이용한다. QoS를 테스트하기 위해서는 직접 테스트 망을 구축하고 필요한 컴포넌트를 설치하여야 한다. 실험을 위하여 구축된 테스트망은 그림 8과 같다.

o 테스트망의 시나리오는 다음과 같다.

- 마스터(192.168.1.3)는 슬레이브(10.1.5.4)에게 일정한 제어신호를 보낸다. 이때, 192.168.1.6과 10.1.5.6은 서로 가능한 많은 패킷을 보내 네트워크에 과부하를 일으킨다.
- TCP, UDP등을 테스트하고 QoS을 적용하기 전과 후의 성능을 테스트하여 차이점을 알아본다.

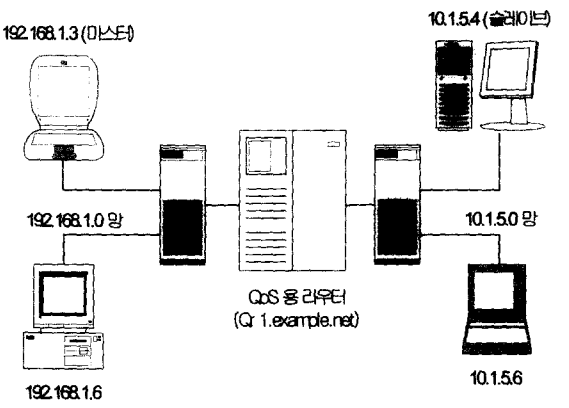


그림 8. QoS 테스트망
Fig. 8. Network for testing QoS.

테스트망의 각 시스템의 구성요소는 표 3과 같다.

표 3. 시스템의 구성요소
Table. 3. components on each system.

시스템	항목	내용
호스트	플랫폼	윈도우 2000
	교통 제어l	QoS 패킷 스케줄러
	RSVP프로그램	GQoS API를 이용한 프로그램
라우터	플랫폼	FreeBSD 3.4
	교통 제어	ALTQ 2.2
	RSVP프로그램	rsvp4.2a4의 rsvpd
허브	속도	10M 스위칭 허브

III. 결과 및 고찰

본 연구에서 제시한 방법의 우월성을 확인하기 위하여 기존의 프로토콜의 장단점을 검토해보고 QoS를 적용했을때 그 결과를 비교하였다.

1. 패킷의 증가에 따른 전송률의 변화

그림 11과 표 4는 TCP로 전송시 패킷의 크기를 변화시켜 보냈을때의 전송률의 변화를 나타낸다.

패킷의 크기가 커질수록 데이터의 전송률이 증가함을 알수 있다. 이는 전송할 데이터가 증가하면 프로토콜의 데이터에 대한 적용의 데이터 비율이 증가하고, 그로 인한 데이터의 전송율이 향상될수 있음을 나타낸다. 이것은 TCP의 한 요소인 흐름제어의 영향으로 인하여 윈도우 크기가 증가함에 따라 성능이 좋아지기 때문이다.

표 4. 패킷의 크기에 따른 최대 전송률
Table 4. maximum bandwidth of each packet size.

byte 항목	최대전송률(kbyte./sec)
1024	169.3
2048	216.5
4096	308.6
8192	410.3
16384	445.3
32768	529.9

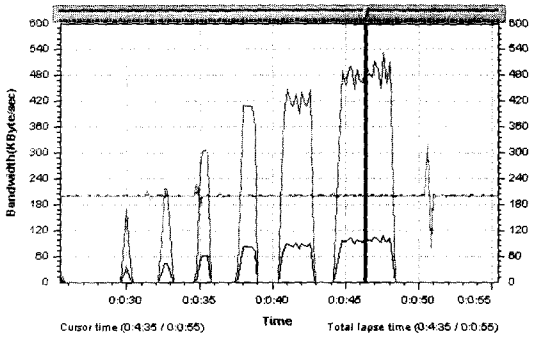


그림 11. 패킷크기에 따른 전송률
Fig. 11. Bandwidth of each packet size.

2. TCP, UDP 그리고 중첩된 입출력의 비교

현재 인터넷에서 가장 많이 이용되는 프로토콜로는 TCP와 UDP가 있다. UDP는 IP를 캡슐화한 프로토콜일 뿐 IP의 기능외에 다른 기능을 제공하지 않는다. TCP는 IP에서 제공하지 않는 신뢰성과 효율성을 위한 플로우 제어와 혼잡제어 기능을 제공한다. 플로우 제어란 패킷의 흐름을 환경에 따라 동적으로 조절할 수 있는 기능을 말하며 혼잡제어란 네트웍이 혼잡할 시에도 신뢰성을 보장할 수 있게 하는 기능을 말한다. 따라서 이론상 신뢰성에서는 TCP가 UDP보다 더 나은 성능을 보이며, 속도면에서는 UDP가 TCP보다 더 나은 성능을 보인다.[4] 또한 마이크로소프트사에서는 소켓을 이용한 가장 빠른 방법은 중첩된 입출력(overlapped I/O)을 사용하는 것이라고 주장하므로, 중첩된 입출력도 테스트한다. 여기서 중첩된 입출력이란 소켓의 동작과 데이타 스택의 입출력이 동시에 이루어지는 것을 말한다.[15]

o 테스트방법 :

- 프로토콜의 "echo" 기능을 이용한다.
- 8192(8K바이트)의 패킷을 1000번 보낸다.
  (총전송량 16384000바이트(=8192*1000*2))
- 부하가 없을 때와 있을 때를 테스트한다.
- 총 10번씩 테스트를 한다.
- 모든 실험은 동일한 조건하에서 실험한다.

그림 12는 부하가 없을 때의 결과로서, (a)는 TCP의 전송률, (b)는 UDP의 전송률 결과이고, (c)는 중첩된 입출력의 전송률 결과이다.

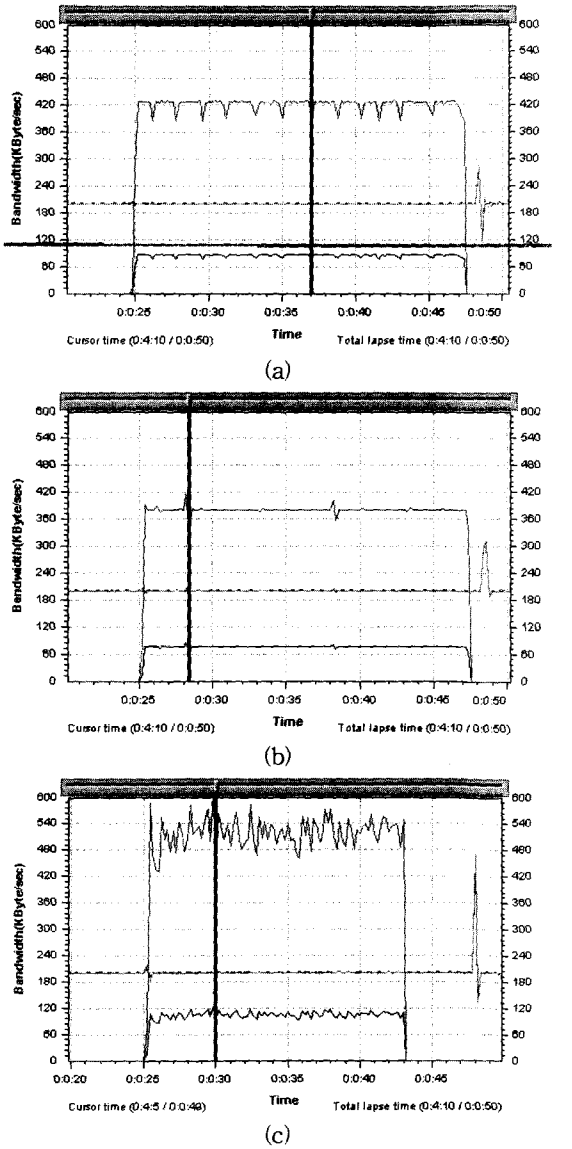


그림 12. 부하가 없을 때의 결과
Fig. 12. The results in the case of no load

그리고 그림 13은 부하가 있을 때의 결과로서, (a)는 TCP의 전송률, (b)는 UDP의 전송률의 결과이고, (c)는 중첩된 입출력의 전송률 결과이다.

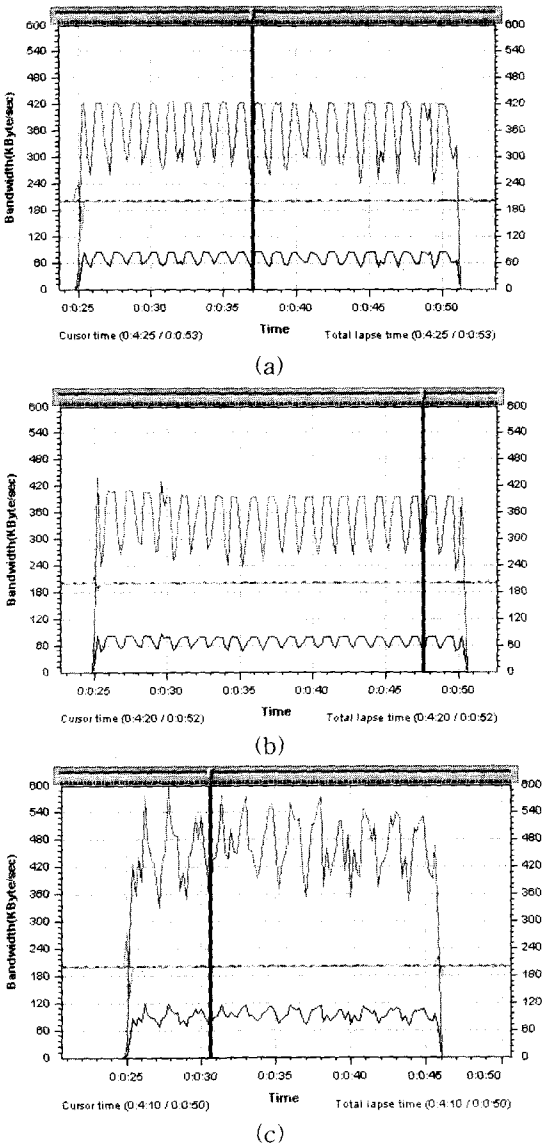


그림 13. 부하가 있을 때의 결과
Fig. 13. The results in the case of load existence

부하가 없을때와 있을때의 실험결과를 살펴보면 다음과 같은 사실을 알수 있다.

1) 중첩된 입출력의 결과가 가장 빠르지만 변화의 정도가 크다. TCP와는 약 170kbyte/sec 정도의 차이를 보였고 변화의 정도는 약 130kbyte/sec 정도 더 컸다.

2) TCP와 UDP는 속도와 전송 시간에 있어서 거의 차이가 없었다. 그 이유로는 실험 방법에 있다. 즉 보내고 받은 데이터를 비교한 실험이 아닌 단순히 전송속도를 측정하기 위한 "echo"기능을 이용한 것이므로 이 실험에서 TCP와 UDP는 거의 차이가 없었다. 그리고 본 실험에서 이용한 마이크로소프트의 UDP 소켓은 8K 이상의 데이터를 보낼 수 없었다. 이것은 표준 버클리 소켓에도 해당되는 항목으로서 UDP 구현의 한계라고 볼 수 있다. 그리고 UDP의 경우 본 실험에서는 부하가 있는 경우에도 데이터 손실이 일어나지 않았다. 이것은 실험환경에서 마스터와 슬레이브의 거리가 가깝기 때문이다. 표 6에 따르면 거리 0.05km이내의 로컬네트워크 환경에서는 데이터 손실이 제로임을 알 수 있다.[16]

표 5. 부하가 있을때와 없을때의 최대/최소 전송률(kbyte/sec)
Table 5. max/min bandwidth of no load and load.

항목 / 프로토콜	부하가 없을 때		부하가 있을 때	
	최대	최소	최대	최소
TCP	427	382	423	241
중첩 I/O	598	431	604	328
UDP	419	358	395	228

3) 부하시 실험한 모든 프로토콜은 네트워크에 과부하를 걸어주기 위한 신호의 영향을 그대로 받아서 부하가 없을 때보다 변화의 정도가 커졌다. TCP의 경우 부하가 없을때의 (최대-최소)값과 부하시의 (최대-최소)값은 약 137(kbyte/sec)의 차이를 보였다. 따라서 원격제어에서 인터넷을 이용할 시 최대 노력 서비스 모델을 기반으로한 프로토콜(TCP, UDP등)을 이용한다면 현재 네트워크상태에 따라 원격제어 자체가 불가능하게 될 수 있음을 알수 있다.

표 6. 인터넷상의 시간지연 파라미터
Table 6. Time-delay parameters for internet

호스트	거리 (km)	평균 지연시간(ms)	표준 이탈	손실 율
로칼 (local)	0.05	0.998	0.715	0.00
동일 도메인	30	8.10	5.35	0.08
다른 시(city)	150	17.20	9.74	0.80
다른 대륙	10000	326.3	27.20	41.4

3. QoS 테스트

본 실험은 QoS의 세가지 서비스 모델인 최대 노력 서비스, 제어 부하 서비스, 보장 서비스 모델을 테스트 하였다.

표 7. 테스트 서비스 모델
Table 7. Service model of each test.

QoS 서비스 모델	프로토콜	트래픽 사양
최대 노력 서비스	TCP	
제어 부하 서비스	RSVP TCP	H263QCIF
보장 서비스	RSVP TCP	G711

본 실험환경은 그림 8과 같은 10M 이더넷 네트워크 상에서 수행된다. 하지만 간단한 실험의 구성으로 인하여 트래픽 생성 방법이 필요하다. 따라서 10M의 용량을 2M로 줄여 테스트하였다. 그림 14는 10M 이더넷의 대역폭을 2M로 줄였을 때의 그림 9와 같은 과부하 파형의 그림이다. 그림 14에서 보는 것과 같이 2M인데도 3M 정도의 대역폭이 나오는 것은 2M를 실제 10M 이더넷을 소프트웨어적으로 프로그램을 통하여 줄였기 때문이다. 즉 물리적으로는 10M이지만 소프트웨어적으로 2M 환경으로 임의적으로 만든 것이므로 오차가 있음을 알수 있다. 또한 그림 14의 잡음 파형은 네트워크에 과부하를 일으키기에 충분함을 알수 있다.

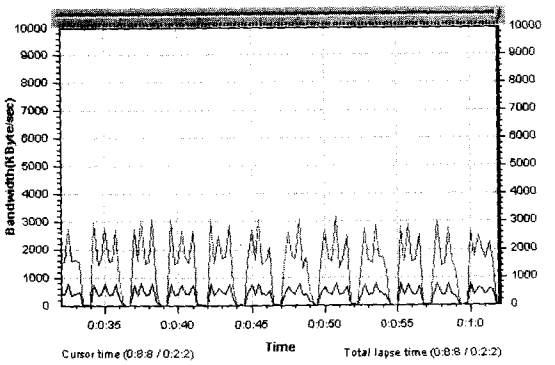


그림 14. 2M 이더넷에서의 과부하 파형
Fig. 14. Resulting wave in case of load and 2M ethernet.

대역폭 확보 실험 결과는 다음과 같다.

(1) TCP

그림 15는 TCP 의 전송률 결과로서, (a)는 무부하시

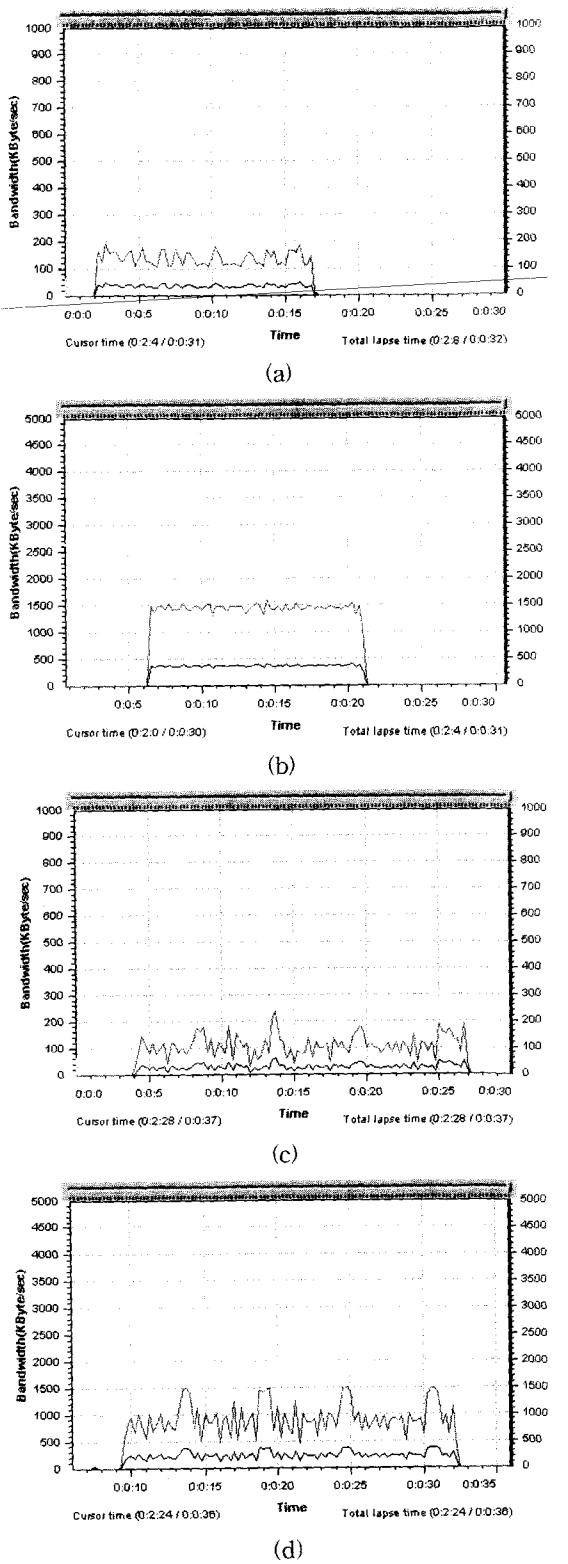
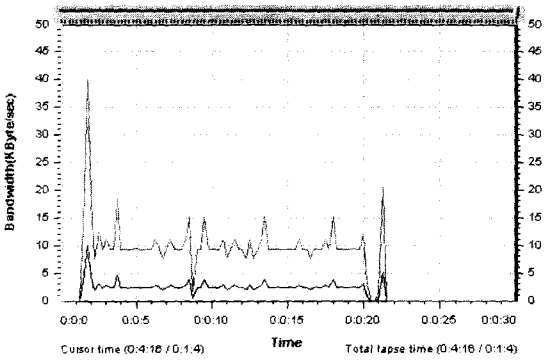


그림 15. TCP 의 전송률
Fig. 15. Bandwidth of TCP.

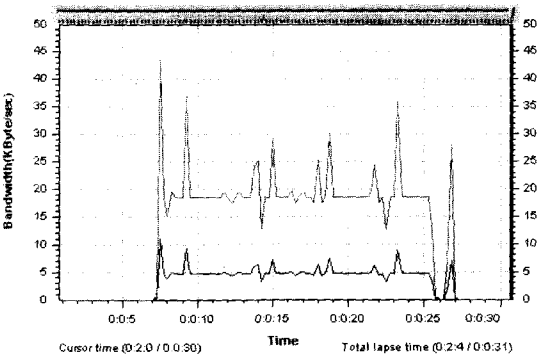
송신측 전송률, (b)는 무부하시 수신측 전송률, (c)는 부하시 송신측 전송률의 결과이며, (d)는 부하시 수신측 전송률의 결과이다. 이 그림에서와 같이 TCP의 경우 부하가 있을 때 부하가 없을 때와 비교하여 파형이 심하게 흔들리고 전송시간도 더 길어졌음을 알수 있다. 따라서 TCP를 이용하여 원격조종을 할 시에 네트워크의 환경과 시간에 따라 원격조종이 안될 수도 있다는 것을 의미한다.

(2) H263QCIF

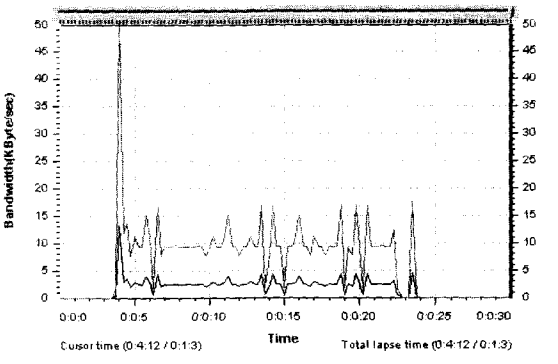
그림 16은 H263QCIF의 전송률 결과로서, (a)는 무부하시 송신측 전송률, (b)는 무부하시 수신측 전송률, (c)는 부하시 송신측 전송률의 결과이며, (d)는 부하시 수신측 전송률의 결과이다. 이 그림에서와 같이 H263QCIF를 이용한 제어 부하는 과부하시에도 일정한 대역폭과 전송시간을 유지함을 알 수 있다.



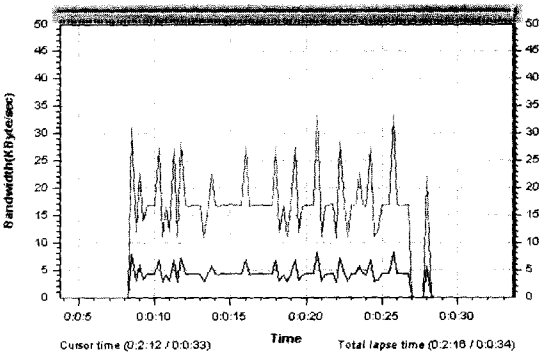
(a)



(b)



(c)

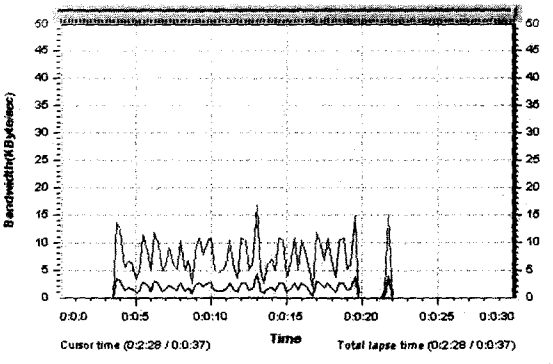


(d)

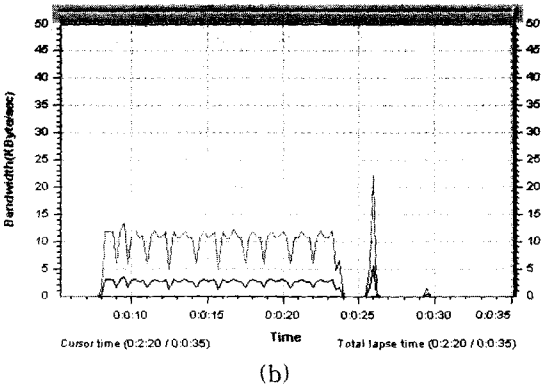
그림 16. H263QCIF 의 전송률
Fig. 16. Bandwidth of H263QCIF.

(3) G711

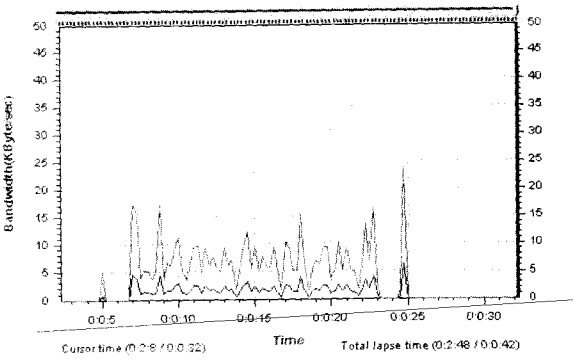
그림 17은 G711의 전송률 결과로서, (a)는 무부하시 송신측 전송률, (b)는 무부하시 수신측 전송률, (c)는 부하시 송신측 전송률의 결과이며, (d)는 부하시 수신측 전송률의 결과이다. 이 그림에서와 같이 G711을 이용한 제어 부하는 과부하시에도 일정한 대역폭과 전송시간을 유지함을 알 수 있다.



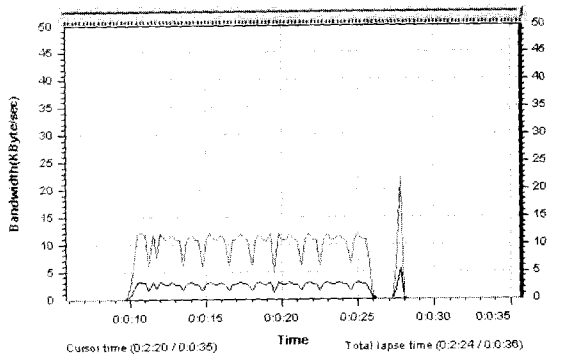
(a)



(b)



(c)



(d)

그림 17. G711의 전송률
Fig. 17. Bandwidth of G711.

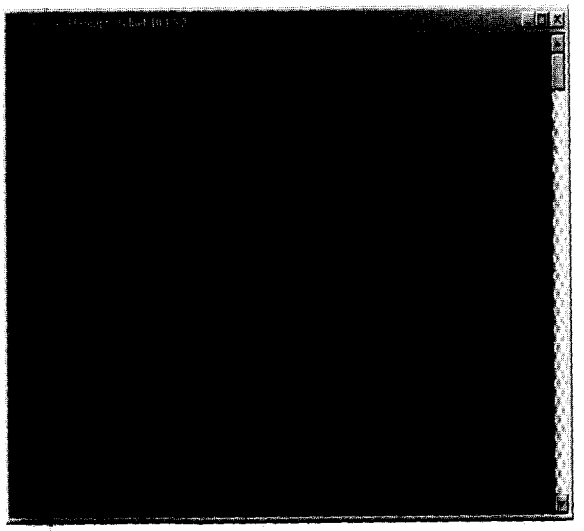


그림 18. rstat을 이용한 자원예약 확인
Fig. 18. Resource allocation using rstat.

부하시 송신측 전송률 결과이며, (d)는 부하시 수신측 전송률의 결과이다. 이 그림에서와 같이 G711을 이용한 보장 서비스 실험은 부하가 없을때와 비교하여 거의 변함이 없음을 알 수 있다. 따라서 기존의 인터넷에서 QoS를 이용할시 원활한 원격조종을 할수 있음을 알 수 있다.

그림 18은 라우터를 텔넷(telnet)으로 접속하여 rstat (라우터의 QoS의 자원예약을 확인하는 벤치마크 프로그램)를 이용하여 라우터에서의 H263QCIF 템플릿의 예약전과 후 그리고 예약이 끝난 후의 자원예약 상황을 확인한 그림이다. 예약이 체결된 것을 알수 있으며 18Kbps의 자원이 확보되어 있음을 알수 있다.

Ⅳ. 결 론

본 연구에서는 인터넷을 이용한 원격제어시 문제점으로 지적되는 동적으로 불규칙하게 변화하는 예측 불가능한 시간지연을 해결할수 있는 방법을 제시하고, 이를 적용한 실험결과를 보였다. 즉 특정 어플리케이션에 일정한 대역폭을 확보하여 동적으로 불규칙적하게 변화하는 시간지연을 정적으로 예측 가능하도록 고정시키고 실시간성을 확보함으로써, 인터넷을 이용한 원격제어시 시간지연문제를 해결할 수 있음을 보였다. 또한 전송률을 일정하게 유지함으로써 원격제어시 지연시간을 설정한 시간내에 확실하게 처리할 수 있음을 알수 있다.

참 고 문 헌

[1] Ken Goldberg, Michael Mascha, Steve Gentner, Nick Rothenberg, Carl Sutter, Jeff Wiegley, "Desktop teleoperation via the world wide web", IEEE International Conference on Robotics and Automation, Vol. 1, pp. 654~659, 1995.

[2] Kevin Brady, "Internet-based remote teleoperation", Proceeding of 1998 IEEE International Conference on Robotics and Automation, pp. 65~70, May 1998.

[3] Kazumasa hirai, Yoshiaki satoh, "Stability of a system with variable time delay", IEEE Tr. on Automatic Control, Vol AC-25, No. 3, pp. 552~554, June 1980.

[4] Fiorini P., Oboe R., "Internet-based telerobo-

tics : problems and approaches", Internatio- nal Conference of Advanced Robotics, pp. 1~6, 1997.

[5] Larry L. Peterson, Bruce S. Davie, Computer networks : a system approach 2nd, Morgan Kaufmann, 2000.

[6] David Durham, Raj Yavatkar, Inside the internet's resource reservation protocol, John Wiley and Sons, 1999.

[7] Shenker S., Patridge. C., Specification of controlled load quality of service, IETF Internet Draft, July 1995.

[8] Shenker S., Patridge. C., Specification of guaranteed quality of service, IETF Internet Draft, July 1995.

[9] David Iseminger, Windows 2000 quality of service, Macmillan Technical Publishing, 1999.

[10] S. Blake, D. Black, M. Carlson, E.Davies, Z. Wand, W, Weiss, An architecture for differentiated services, RFC2475, Dec. 1998.

[11] J. Wroclawski, The use of rsvp with IETF integrated services, RFC2210, Sep. 1997.

[12] Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S, Resource reservation protocol (RSVP) - version 1 functional specification, RFC 2205, Sep. 1997.

[13] A. Demers, S. Keshav, S. Shenker, "Analysis and simulation of a fair queuing algorithm", Journal of Internetworking Research and Experience, pp. 3~26, 1990.

[14] Sugih Jamin, "A measurement-based admission control algorithm for integrated service packet networks", Ph.D Dissertation, pp. 1~107, Aug. 1996.

[15] Ralph Davis, Win32 network programm- ing, Addison-Wesley, 1996.

[16] Roberto Oboe, Paolo Fiorini, "A design and control environment for internet-based telerobotics", International Journal of Robotics, Spring 1998.

저 자 소 개

許 慶 茂(正會員)

1979년 서울대학교 전자공학과 졸업, 한국과학기술원 전기및전자공학과 석사(1981), 동대학 박사(1989). 현재 단국대학교 전자컴퓨터학부 전자공학전공 부교수

沈 鉉 承(學生會員)

1988년 단국대학교 전자공학과 졸업, 동대학 석사(2001)