論 文

트랜스포트 프로토콜 HSTP와 TP4의 성능에 대한 시뮬레이션 연구

正會員 朴 章 然* 正會員 崔 陽 熙*

A Simulation Study on the Performance of Transport Protocol HSTP and TP4

Jang Yeon Park*, Yang Hee Choi* *Regular Members*

要 約

큰 대역폭과 낮은 비트 에러율을 가진 고속망환경에서 기존의 낮은 대역폭과 높은 비트 에러율을 가진 망환경에서 개발된 트랜스포트 프로토콜을 사용할 경우에 비효율적인 구조와 오버헤드로 인하여 성능이 크게 저하한다. 따라서 고속망환경에 알맞는 새로운 프로토콜로 HSTP(High Speed Transport Protocol)가 제안되었다. 본 연구는 HSTP의 3가지 에러제어 기법과 TP4의 2가지 에러제어 기법을 이용했을 때의 성능과 HSTP의 rate control을 이용하여 흐름제어를 할 경우의 성능을 시뮬레이션 연구를 통하여 평가, 비교하였다.

ABSTRACT

The conventional transport protocols, developed on top of the low-bandwidth high-error network services, exhibit poor performances when operating in the new high-bandwidth low-error networks. A new transport protocol HSTP(High Speed Transport Protocol) was proposed to take advantage of the new network environment. This paper gives a performance evaluation by simulation technique of the proposed HSTP and conventional TP4(Transport Protocol class 4) in their data transfer characteristics. The performance variation by HSTP and TP4 are investigated when their various error control techniques are employed ; three different techniques for HSTP, and two for TP4. The performance enhancement in HSTP is also evaluated when the rate control technique is used as part of the flow control mechanisms.

*서울大學校 컴퓨터工學科
Dept. of Computer Eng., Seoul National Univ.
論文番號 : 93-130

I. 개 요

기존의 컴퓨터 네트워크는 수십 *Kbps*에서 수 *Mbps* 정도의 낮은 전송 대역폭과 높은 전송 비트 에러율

(*Bit Error Rate*; *BER*)때문에 통신 프로토콜은 이를 극복하는데 기능상의 촛점이 맞추어져 왔다. 그러나 현재의 새로운 네트워크 환경은 광케이블, 초고집적 반도체 기술 등의 발전으로 인하여 수백 *Mbps*에서 수 *Gbps*의 넓은 대역폭을 제공하고 있으며 낮은 *BER*로 인해 전송매체상에서의 오류로 인한 패킷 손실도 무시할 수 있을 만큼 감소하게 되었다. 사용자들의 서비스 요구도 문자데이타나 저해상의 그래픽으로부터 음성, 영상, 오디오, 고해상의 그래픽등 대용량의 멀티미디어 정보처리와 실시간 처리, 분산 처리등으로 변화하고 있는 추세이다.

기존의 OSI(Open Systems Interconnection) 프로토콜 스택이나 TCP/IP는 네트워크의 대역폭이 작고 에러율이 높으며 시스템 자원이 부족하던 시기에 개발되었기 때문에, 대역폭의 효율적 사용과 에러의 발견 및 회복을 효율적으로 수행하는 것이 설계의 주된 목표였다. 따라서 기존 프로토콜의 설계는 다음과 같은 원리에 기초하고 있다[1].

- 전송비용을 감소시키기 위해서 전송 사전처리와 사후처리를 시스템에서 최대한 수행한다.
- 전송매체에서의 에러발생율이 높기때문에 에러 발견과 회복을 위한 프로토콜 처리가 최적화되도록 한다.
- 비교적 단순한 흐름제어 기법을 사용한다.

위에서 볼 수 있듯이 기존 프로토콜은 네트워크의 부하를 최소한 줄이기 위해서 프로토콜 처리 시스템에서의 부하를 증가시키고 있다. 그러나 현재의 고속망 환경의 특징은 값싸고 풍부한 대역폭이어서 오히려 프로토콜 처리속도가 주된 성능제한요소로 등장하고 있다. 따라서 고속망 환경에서 기존 프로토콜의 구조적인 문제점으로 인하여 발생하는 성능 저하를 해결하기 위하여 기존 프로토콜의 기능을 변경, 확장하고 프로토콜 처리시간을 최소화하는 방법과 현재의 고속망환경에 알맞게 구조화된 새로운 프로토콜을 설계하는 방법등이 동원되고 있다. 두 방법중 기존 프로토콜의 확장 또는 효율적인 구현은 장기적인 해결책이 되지 못하므로 새로운 프로토콜의 설계가 활발히 연구되고 있다. 이중에서도 프로토콜중 가장 성능저하의 큰 요소인 트랜스포트 프로토콜을 집중적으로 연구하고 있는데, 새롭게 제안된 트랜스포트 프로토콜중에서 대표적인 것이 HSTP(High Speed Transport Protocol)이다[2,3]. 새로운 프로토콜의 설계는 다음과 같은 원리에 기초한다[1].

- 대역폭이 낭비되더라도 고속망에서 성능의 최대 장애물인 프로토콜 처리시간을 최소화하기 위해서 프로토콜 처리 요구량을 최소화한다.
- 광케이블을 사용한 새로운 고속망은 에러율이 작기 때문에 에러회복을 위하여 과도한 처리를 하지 않도록 프로토콜 처리를 최적화한다.
- 기존의 흐름제어 기법보다 더욱 효율적인 흐름제어를 채택한다.

본 연구에서는 고속망 환경에서 새롭게 설계된 HSTP와 기존의 국제표준 트랜스포트 프로토콜인 TP4(Transport Protocol class 4)[4,5]를 시뮬레이션 연구를 통하여 정량적인 관점에서 비교, 평가하여 고속망 환경에서 HSTP가 기존 프로토콜보다 효율적인지의 여부를 검토하고자 한다. 특히 HSTP와 TP4의 주요 차이점인 흐름제어와 에러제어 기법을 중심으로 성능비교를 하였다. HSTP는 에러제어에서 go-back-N과 selective repeat를 허용하며, 순서에 어긋나게 수신된 패킷의 저장 여부도 선택하게 하고 있다. 또한 TP4도 순서에 어긋나게 수신된 패킷을 보관할 수도 있고 버릴 수도 있다. 따라서 어떠한 조합의 에러제어 파라미터를 선택하는 것이 가장 효율적인 것인가를 주요 시뮬레이션 대상으로 삼았다. 흐름제어의 경우는 HSTP에서 제공하는 rate control을 이용하여 윈도우 흐름제어보다 나은 성능을 얻을 수 있는 망환경과 가장 효율적으로 흐름제어를 할 수 있는 파라미터 값을 알아보는데 촛점을 두었다.

Ⅱ와 Ⅲ에서는 HSTP와 TP4에 대해서 간단히 설명하였고, Ⅳ는 시뮬레이션 모델과 파라미터, 가정에 대하여 설명하였다. Ⅴ는 시뮬레이션 성능 평가를 한 결과이다. Ⅵ에서 본 연구의 결론을 제시하고자 한다.

Ⅱ. HSTP (High Speed Transport Protocol)

OSI 모델의 트랜스포트 계층에 위치한 프로토콜로서 현재 국제표준기구의 고속 트랜스포트 프로토콜 표준화 작업에 제출되어 있는 HSTP[2]는 기존의 XTP(Express Transfer Protocol)[6,7]에 기초하고 있다. XTP는 트랜스포트 계층과 네트워크 계층의 일부를 포함하는 프로토콜 인데, 여기에서 네트워크 계층의 기능을 없애고, 순차값 범위(sequence space)

를 32비트에서 64비트로 확장하는 등, 약간의 수정을 가한 것이 HSTP이다. HSTP는 ISO 8348[8,9]에서 정의된 비연결 네트워크 서비스상에서 동작하도록 설계되었고, ISO 트랜스포트 프로토콜 서비스 정의(ISO 8072)에 기초한 HSTS(High Speed Transport Services)[3]를 상위 계층에 제공한다.

본 연구에서는 일대일 연결방식(peer-to-peer connection mode) 서비스를 제공하는 HSTP를 대상으로 하였다.

HSTP의 중요 특징으로 병렬 처리가 가능한 프로토콜 구조, rate 제어에 의한 흐름제어, 선택적 재전송에 의한 에러제어, 수신확인응답신호의 전송시기 결정, 적은 수의 타이머 등을 들 수 있다.

가. 프로토콜 구조

앞 절에서 언급하였듯이 현재의 고속망에서 성능의 장애 요소는 프로토콜을 처리하는데 소요되는 시간이다. 기존의 프로토콜은 하나의 순차적인 프로토콜 기계(protocol machine)로 설계되었기 때문에 하드웨어에 의한 병렬 수행이 어렵다. 이에 반해 HSTP는 하드웨어에 의한 프로토콜 처리를 설계개념으로 하였기 때문에 VLSI 구현에 의한 병렬 처리가 효율적으로 수행될 수 있도록 병렬적 구조(Parallel Architecture)를 가진 프로토콜 기계로 설계되었다.

나. 흐름제어

HSTP는 출력제어를 위한 윈도우 흐름제어외에 Rate Control과 스케쥴링의 기능을 제공한다. 고속망에서는 대역폭과 전파지연의 곱이 매우 크므로 양단 시스템사이에서 전송중인 패킷의 수가 매우 많다. 이러한 환경에서 윈도우 흐름제어만을 사용하여 출력제어를 할 경우에는 다음과 같은 문제점이 발생한다. 첫째로는 윈도우 흐름제어에서는 할당된 버퍼의 양만큼은 아무런 제약조건없이 전송할 수 있으므로 네트워크의 트래픽을 급격히 증가시킬 수 있다. 이의 결과로 네트워크를 통한 전송중 트래픽 밀집(congestion)에 의한 패킷 손실이 발생한다. 현재 고속망에서는 *BER*에 의한 패킷손실이 매우 적으므로, 에러의 주된 요인이 전송중에 트래픽 밀집에 의한 패킷손실이며 윈도우 흐름제어로는 이를 방지할 수 없다. 두번째로는 수신 시스템이 처리할 수 있는 용량 한도이상의 패킷이 연속적으로 도착할 경우에는 버퍼공간이 비어있더라도, 수신 시스템에서의 스케쥴링 문제 등으로 인하여 패킷의 손실이 발생한다.

윈도우 흐름제어의 위와 같은 문제점으로 인하여 HSTP에서는 송신 시스템에서 단위 시간동안(*burst* 간격) 전송할 수 있는 데이타의 양을 제한하여 패킷 전송 속도를 제어하는 rate control을 도입하고 있다. HSTP에서는 *rate*와 *burst*의 두 파라미터를 사용하여 rate control을 수행한다. 수신자는 자신이 수신처리할 수 있는 초당 데이타 옥텟(octet)의 수를 *rate*로 정의한다. 또한 한꺼번에 수신하여 처리할 수 있는 데이타의 양을 *burst*로 정의한다. 예를 들면 $1Gbps$ 전송로로 연결된 수신자가 초당 10^5개의 옥텟을 처리할 수 있으며, 이를 다섯번에 나누어 20,000 옥텟씩 받도록 한 경우 *rate*는 10^5, *burst*는 20,000이 된다. 따라서 전송로상으로 보면 20,000 옥텟이 $1Gbps$의 속도로 $0.16msec$ 동안에 한꺼번에 전송되고 0.2초가 될 때까지 $199.84msec$는 idle 상태로 있게 된다. 이와 같이 고속망에서 rate control의 잇점은 버스트 주기(*burst/rate*)동안에 허용된 *burst*만큼의 패킷만이 송신부에서 전송될 수 있으므로, 송신부가 빠르더라도 수신부의 능력 이상의 속도를 데이타를 보내지 않아 오버플로우에 의한 패킷손실을 없앨 수 있다.

다. 에러제어

HSTP에서는 에러제어를 위해서 go-back-N과 selective repeat 기법을 제공한다. 두 에러제어 메카니즘에서 어느 것을 사용할 것인지는 연결 설정시에 결정한다.

본 연구에서는 에러 회복을 위한 재전송 기법과 순서에 어긋난 패킷이 도착하였을 때의 수신자에서의 처리 방법에 따라 표 1에 나타나 있는 3가지의 프로토콜 구현 방법에 대한 분석을 수행하였다.

표 1. HSTP의 에러제어 기법
Table 1. Error Control Techniques in HSTP

A. 전송자는 selective repeat 재전송을 한다. A.1. 수신자는 순서에 어긋나게 도착한 패킷을 저장한다.
B. 전송자는 go-back-N 재전송을 한다. B.1. 수신자는 순서에 어긋나게 도착한 패킷을 저장한다. B.2. 수신자는 순서에 어긋나게 도착한 패킷을 버린다.

라. 응답전송시기

HSTP에서는 수신 시스템의 상태정보를 알기 위하여 송신부는 데이타 패킷이나 제어 패킷의 *SREQ* 비트를 사용하여 응답 전송 시기를 명령한다. *SREQ*를 언제 세트해서 응답을 요청할 것인가 하는 것이

성능에 큰 영향을 미치는 중요한 요인이다. 본 연구에서는 TSDU(Transport Service Data Unit) 단위로 응답요청을 하도록 하였다. 상위계층에서 트랜스포트로 전달된 TSDU는 여러개의 TPDU(Transport Protocol Data Unit)로 나누어 전송되는 데, TSDU의 마지막 부분을 포함하는 TPDU의 *SREQ* 비트를 세트하였다.

마. 타이머

HSTP에서 전송된 데이타의 응답을 기다리기 위한 타이머로서 WTIMER가 있다. WTIMER는 트랜스포트 연결당 하나만 사용하여, *SREQ* 비트가 세트된 패킷이 전송될 때 마다 재시동되고, 응답을 받지 못하고 시간이 종료되면 동기화 핸드세이크(synchronizing handshake) 과정을 시작하는 것으로 본 연구에서 가정하였다.

Ⅲ. ISO 트랜스포트 프로토콜 공급4(TP4)

ISO TP4는 국제표준기구에서 제정된 표준 트랜스포트 프로토콜이며, 다섯개의 등급(class)중에서 가장 복잡하며 우수한 성능을 제공한다[3,4]. ISO TP4는 전송되는 패킷이 허용될 수 없는 정도로 손상되거나 손실되는 서비스를 제공하는 C형 네트워크 서비스 위에서 동작한다. 따라서 TP4는 에러검출과 재전송에 의한 에러회복의 기능을 제공하고 있다. TP4는 출력제어를 위해 크레디트에 기초한 윈도우 흐름제어를 사용한다. 송신측에서는 수신측으로부터의 응답을 기다리지 않고서도 윈도우에서 할당된 갯수 만큼의 데이타 트랜스포트 프로토콜 데이타 단위(DT-TPDU)를 전송할 수 있다. 수신측에서는 특정한 수의 DT-TPDU를 수신하였을 때, 에러없이 순서에 맞게 도착한 DT-TPDU의 순차값과 송신측의 윈도우를 갱신하는데 필요한 정보인 크레디트(credit)를 포함하는 응답 트랜스포트 프로토콜 데이타 단위(AK-TPDU)를 송신측으로 전송한다. TP4에서는 negative acknowledgement가 사용되지 않고, 재전송 타이머가 종료되었을 때 재전송을 수행하는 positive acknowledgement를 사용한다.

재전송을 제어하는 타이머는 국부 재전송 타이머라고 불리우는데, 이 타이머들은 모두 똑같은 종료값을 갖는다. 시간종료 기법은 두가지 서로 다른 방법에 의해서 구현될 수 있다. 한가지는 트랜스포트 연결당 하나의 타이머가 사용되는 것이고 다른 한가지는 전송되는 DT-TPDU당 하나의 타이머를 사용하는 것이다. 후자에 경우에서는 윈도우의 크기가 W일때 W개만큼의 타이머가 필요하므로 타이머 처리에 의한 오버헤드를 적게 하기 위해서 W를 작게 하면 고속망에서 거의 stop-and-wait 방식이 되어 성능이 나빠지고, W를 크게 하면 타이머 수에 증가로 인하여 오버헤드가 증가하므로 처리율이 크게 감소한다. 따라서 고속망에서는 이 방법이 매우 부적절함을 알 수 있다. 본 연구에서 TP4의 타이머는 트랜스포트 연결당 하나의 타이머를 사용하고 있다.

고속망에서는 전송중인 데이타 패킷(DT-TPDU)의 수가 증가한다. 이러한 상황에서 데이타 패킷당 하나의 ACK를 전송하면 데이타 전송방향과 반대방향으로 같은 수 만큼의 ACK가 전송될 것이므로 프로토콜처리시스템에서 데이타와 ACK를 처리하는 작업부하가 똑같게 되어 효율이 떨어진다. 또한 고속망에서는 패킷이 짧은 시간간격을 두고 연속적으로 도착하기 때문에 여러개의 패킷에 대한 ACK를 축적하였다가 전송하여도 패킷당 하나의 ACK를 보낼 때에 비해서 delay 증가 요인이 아주 작다. 따라서 n개의 패킷이 도착하였을 때 하나의 ACK를 축적하여 전송하는 방법이 효율적이라고 판단되는데, 본 연구에서 HSTP의 ACK는 하나의 TSDU에 대해서 하나씩 전송되므로 약 4개의 TPDU당 하나의 ACK가 전

표 2. TP4 프로토콜 파라미터 선택

Table 2. Selected Protocol Parameters for TP4

• 트랜스포트 연결당 하나의 타이머 사용
• 4개의 패킷에 대한 ACK를 축적하였다가 전송
• 타이머 종료시에 송신자(sender)는 응답을 기다리는 모든 DT-TPDU를 재전송하고 타이머를 재시동
• 순서에 맞게 도착한 DT-TPDU를 수신하면 수신자(receiver)는 DT-TPDU를 저장하고 특정한 수의 DT-TPDU를 수신한 후에만 AK-TPDU를 전송 • 순서에 어긋나는 DT-TPDU가 도착했을 경우의 처리전략 – DT-TPDU를 버린다. – DT-TPDU를 저장한다.

송된다. 이것과 동일한 환경을 위해서 TP4에 대해서도 4개의 TPDU에 대한 ACK를 축적하였다가 하나의 ACK를 전송한다.

본 연구에서 TP4에 대해서 연구된 프로토콜 전략은 다음 표 2와 같이 순서에 어긋나게 도착한 패킷을 버리느냐, 저장하여 재순서화 하느냐에 따라서 두가지 방법을 채택하였다.

Ⅳ. 시뮬레이션 모델

그림 1은 본 연구에서 HSTP와 TP4의 구조적인 개념모델의 성능을 평가하기 위해서 동일하게 사용한 모델이다. 이 모델은 한 방향의 데이타 전송과 반대 방향의 응답(ACK) 전송을 가정한 것이다. 시뮬레이션 모델은 크게 송신자(sender), 수신자(receiver), 네트워크 가블러(network garbler)의 세가지 요소로 구성되어 있다. 데이타는 송신자로부터 네트워크 가블러를 통해서 수신자로 전송되고, 수신자에서 전송되는 응답(ACK)신호도 네트워크 가블러를 통하여 송신자로 보내진다.

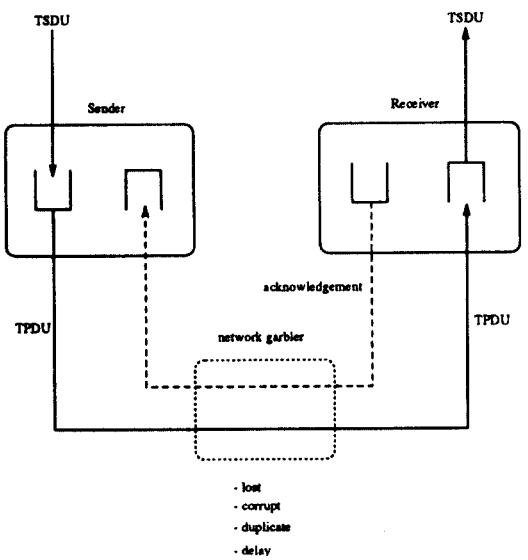


그림 1. 시뮬레이션 모델
Figure. 1. Simulation model

가. 시뮬레이션 구성요소

송신자는 상위계층에서 도착한 TSDU를 TPDU로 패킷화하여 저장하는 무한큐를 가지고 있고 데이타 패킷과 제어패킷의 전송과 ACK의 수신을 처리한다. 단일 프로세서로 구성된 송신시스템내에서는 제어패킷의 처리가 데이타패킷의 처리보다 높은 우선순위를 가지며 프로세서의 차지(preempt)는 허용되지 않는다.

수신자는 네트워크 가블러에서 수신한 TPDU를 저장하기 위한 유한큐를 가지고 있으며, 데이타패킷과 제어패킷의 수신, 자신의 상태정보를 송신자에게 알리기 위한 ACK 패킷의 전송 기능을 수행한다. 송신자와 마찬가지로 제어패킷의 처리가 데이타 패킷의 처리보다 높은 우선순위를 가지며 프로세서의 차지(preempt)는 허용되지 않는다. 시뮬레이션에서 측정하는 TSDU 전송지연시간은 TSDU가 상위계층으로부터 송신자의 무한큐에 도착한 시간부터 TSDU가 수신부로 에러없이 전송되어 상위계층으로 전달될 때까지의 시간이다.

재전송되는 패킷은 상위계층에서 새로이 도착한 패킷보다 먼저 전송된다. 같은 우선순위의 패킷은 FIFO 방식에 의하여 서비스 받는다.

네트워크 가블러는 네트워크 시뮬레이터로서 네트워크 환경에서의 여러가지 에러 현상들을 구현한다. 실제 망 환경에서는 링크의 *BER*에 의한 패킷손실(lost)과 패킷에러(corrupt) 뿐만아니라 중간 노드의 저속도와 버퍼 부족에 의한 패킷손실도 발생하며, 중간 노드에서의 패킷 재전송에 의한 패킷중복 등이 발생할 수 있다. 앞에서 기술한 바와 같이 광케이블로 구현된 신뢰성있는 망환경에서는 *BER*에 의한 에러발생보다는 고속으로 인한 중간노드에서의 패킷과잉등으로 인한 패킷손실과 중복발생이 더 큰 에러의 요인이 되었다. 따라서 이러한 망환경에서 에러를 모델링하기 위해서 네트워크 가불러가 사용되었다. 본 시뮬레이션 모델에서 네트워크 가블러는 패킷손실(lost), 패킷중복(duplicate)의 확률값을 파라미터로 전달받아 주어진 확률값에 따라서 가블러를 지나가는 메세지에 에러를 첨가시킨다. 가블러의 기능을 모델링한 것은 그림 2에 나타냈다.

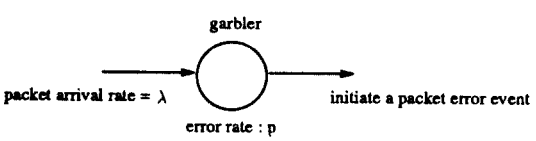


그림 2. 가블러 모델
Figure. 2. Garbler model

poisson 분포에 따라 가블러에 도착하는 패킷의 도착률 λ, 에러에 대한 파라미터로 주어진 확률을 p 일 때, 에러의 발생은 도착한 패킷에 대하여 에러확률 p를 갖는 베르누이 프로세스에 의하여 결정된다. 시간 t동안 가블러에 도착한 패킷의 평균수는 λt이면 이 기간동안 가블러에서 발생하는 에러의 횟수 n은 이항분포에 따른다.

$$n = \lambda t \times p \times (1-p)$$

나. 트래픽 모델 및 파라미터

상위 계층으로 부터 HSTP나 TP4의 전송 트랜스포트 개체(entity)로 전송을 요구하는 TSDU는 Poisson 분포에 따라서 도착한다. 도착하는 TSDU의 크기는 1개에서 8개의 TPDU를 동일한 확률로 갖도록 분포되었다. 이때 도착하는 TPDU의 초당 평균갯수를 λ라고 한다.

하위 계층은 HSTP나 TP4연결에 대해서 $100Mbps$의 대역폭을 독점적으로 제공하여 준다. 현재의 고속망은 대단히 신뢰성이 높아 링크의 BER이 10^{-9}이하이다. 본 모델에서 링크의 BER은 10^{-9}으로 한다. BER에 의한 패킷에러 이외에 전송중에 네트워크에서 발생하는 중복전송의 확률은 0.01%로 고정한다. 패킷 손실 발생율은 시뮬레이션에 따라 바뀌가면서 실험한다.

TPDU의 크기는 $1K$ 옥텟으로 한다. $1K$ 옥텟의 TPDU를 전송하는데 걸리는 시간 T_f는 약$0.078ms$이다. 트래픽 부하 ρ는 $\rho = \lambda \times T_f$로 정의한다. 따라서 송신자에 도착하는 패킷의 도착률 λ는 ρ/T_f에 의해서 결정된다.

양단 시스템사이의 전파지연(propagation delay) T_p는 충분히 큰 경우를 대상으로 삼아서 $20ms$로 한다.

윈도우 흐름제어에서 처리율이 최대가 되기 위한 버퍼 크기 K_r의 조건은 데이타 전송로상에 서비스율이 가장 낮은 것이 μ_m일때 다음 식을 만족해야 한다.

$$\frac{K_r}{\mu_m} \geq 2T_p$$

본 연구의 성능평가에서 버퍼의 크기는 윗식을 만족하도록 한다.

V. 성능 평가 결과

이 절에서는 HSTP와 TP4의 구조적인 차이점인 에러회복 기법과 HSTP의 rate control을 이용하여 출력을 제어했을 때의 성능을 시뮬레이션 결과를 이용하여 비교, 분석하였다. 본 연구의 결과는 batch mean방법을 이용하여 95%의 신뢰도를 가지고 10%의 오차를 허용한다.

가. 에러제어

본 연구에서 대상이된 HSTP와 TP4의 프로토콜 구현 전략은 표 3에 따라 구분하였다.

본 시뮬레이션 연구는 HSTP와 TP4의 구현모델에 대한 성능 비교가 아니라 구조적인 개념모델에 대한 성능을 비교하기 위한 것이므로 선택된 프로토콜 구현에 대한 복잡도는 모두 같다고 하여 양단 시스템에서의 패킷처리시간은 모두 $0.1ms$로 한다. 이때 송신자와 수신자의 서비스율 μ_s와 μ_r은 다음과 같다.

$$\mu_s = \mu_r = \frac{1\ packets}{(0.1+0.078)\ ms} \fallingdotseq 5.6\ packets/ms$$

버퍼크기 K_r은 윈도우 흐름제어가 최대의 성능을 가질 조건인 $K_r > \mu_m \times 2T_p \fallingdotseq 224$를 만족하도록 선정한다. 본절에서는 이식을 만족하도록 K_r의 크기를 $256K$ 옥텟으로 한다.

이러한 환경에서 표 3의 다섯가지 프로토콜 전략을 시뮬레이션한 것을 그림 3에 나타내었다.

그림 3에서는 네트워크 가블러에서의 패킷손실율이 0.05%일 때, ρ를 변경시키면서 5가지 프로토콜 전

표 3. 연구된 HSTP와 TP4의 프로토콜 파라미터
Table 3. HSTP and TP4 Protocol Parameters studied in the paper

구 분	프로토콜	에러 재전송	순서에 어긋난 패킷 처리
방법 1	HSTP	selective repeat	저장
방법 2	HSTP	go-back-N	저장
방법 3	HSTP	go-back-N	버림
방법 4	TP4	ACK를 기다리는 모든 DT-TPDU 전송	저장
방법 5	TP4	ACK를 기다리는 모든 DT-TPDU 전송	버림

략에 대한 성능을 측정한 결과를 보여준다. 이 때 수신자는 송신자에서 전송된 모든 패킷을 수신할 수 있다고 가정하고 HSTP에서 rate control을 수행하지 않을 때의 성능이다. 따라서 이 그림은 HSTP와 TP4의 에러제어 기법에 대한 성능을 비교하여 보여준다.

방법 1은 에러회복을 위해 selective repeat를 사용하므로 다른 4가지 방법에 비해 월등한 성능을 보인다. HSTP와 TP4가 에러 재전송을 위해 go-back-N과 순서에 어긋난 패킷을 저장하는 방법 2와 방법 3에 대해서 처리율이 24*Mbps*가 될 때까지는 HSTP가 TP4보다 나은 성능을 보이다가 처리율이 그 이상이 될 때는 TP4보다 감소하는 데, 이는 본 연구에서 HSTP의 응답 요구를 위하여 *SREQ*를 매 TSDU 끝에 해당하는 TPDU를 세트했는데, 이로 인해 그림 4과 같은 문제점이 발생하기 때문이다. 그림 4를 설명하면 다음과 같다. 송신자에서 1, 2, 3번 패킷을 전송하고 3번 패킷의 SREQ 비트를 세트하였다. 전송도중에 2번 패킷이 손실되어 1, 3번 패킷만 수신자에 의해 수신되었다. 수신자는 3번 패킷의 ACK 요청에 의해 2번 패킷에 에러가 발생하였다는 정보를 송신자에게 전송한다. 이시간 이후부터 2번 패킷이 에러없이 수신될 때까지 수신자는 ACK 요청에 대해서 2번 패킷에 에러가 발생하였다는 사실을 송신자에게 알린다. 따라서 이를 수신하는 go-back-N 송신자는 2번 패킷부터 재전송을 시작해야 하므로 트래픽 용량이 증가할 수록 오버헤드로 인하여 성능이 급격히 감소한다. 그러나, 방법 1의 selective repeat 송신자는 에러가 된 패킷만을 재전송하므로 이로 인한 오버헤드가 별로 크지 않으므로 좋은 성능을 유지할 수 있다. 마찬가지의 이유로 인해서 방법3이 방법 5보다 낮은 성능을 띠게 된다. 위와 같은 이유에도 불구하고 본 연구에서와 같은 SREQ 세트 방법을 사용한 이유는 이 방법이 방법 1에 대해서는 가장 좋은 성능을 유지하기 때문에 HSTP에서 이 ACK 전략을 이용했다. HSTP의 go-back-N 재전송을 사용할 경우에는 SREQ를 세트한 패킷의 수를 작게 유지하는 것이 에러 발생시 재전송에 의한 오버헤드를 작게 할 수 있으므로 성능을 높일 수 있다.

그림 3의 결과를 통해 HSTP에서는 방법 1이, TP4에서는 방법 4가 가장 좋은 에러제어 기법이며, 또한 HSTP의 방법 1은 TP4의 방법 4보다 우수한 성능을 제공해 줌을 알 수 있다.

그림 5는 $\rho = 0.15$일 때, 방법 1−5에 대하여 가블러에서의 패킷 손실율을 증가시킬때의 성능을 보여

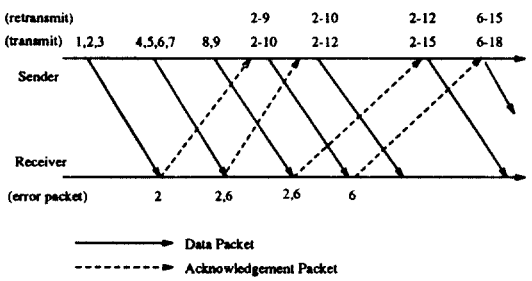


그림 4. 본 연구의 ACK 전략에 따른 에러 발생시에 패킷 중복 전송

Figure. 4. The influence of ACK scheme on the packet retransmission when error occured

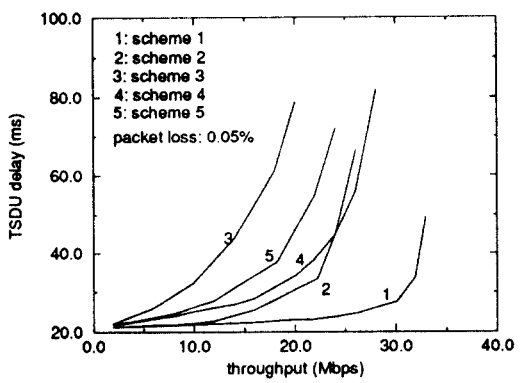


그림 3. 방법 1−5의 성능

Figure. 3. Performance of scheme 1−5

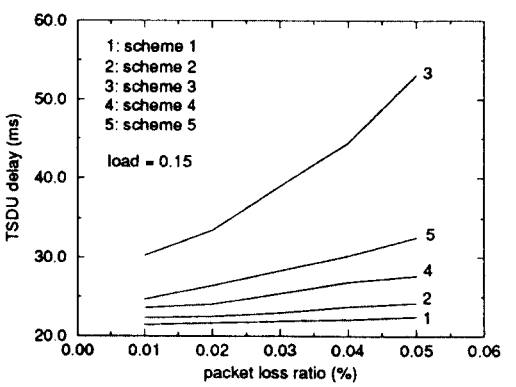


그림 5. 패킷 손실율에 따른 방법 1−5의 성능

Figure. 5. Performance of scheme 1−5 versus packet loss rate

준다. 방법 1이 패킷 손실에 따른 재전송을 위한 오버헤드가 가장 작으므로 패킷손실이 증가해도 성능에 큰 영향을 미치지 않고 가장 좋은 성능을 보인다.

그림 3, 5의 결과에서 HSTP의 방법 1이 주어진 환경에서 가장 우수한 에러제어 기법임을 알 수 있으며, 이는 고속망환경에서 일반적으로 성립할 것으로 기대된다.

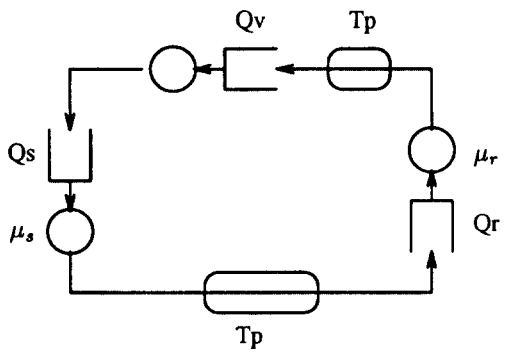


그림 6. 윈도우 흐름제어에 대한 연속적인 fluid model
Figure. 6. Fluid model of a window-based flow control

나. Rate Control

대역폭과 전파지연의 곱이 큰 고속망환경에서 송신자의 전송속도가 전송경로상에 노드나 수신자의 처리속도에 비해 매우 빠른 경우에 윈도우 흐름제어만으로는 효율적으로 흐름제어를 할 수 없다[10]. 효율적인 흐름제어를 하기 위해서는 직접적으로 데이타의 전송속도를 제어할 수 있는 메카니즘이 필요하다. HSTP에서는 *rate*와 *burst* 두 파라미터를 이용하여 rate control을 제공한다.

본절에서는 송신자의 처리속도가 수신자의 처리속도보다 빠르고 일정시간동안 수신자에서 수신할 수 있는 패킷 수에 한계가 있을 때, 윈도우 흐름제어만으로 흐름제어를 하는 경우와 윈도우 흐름제어와 더불어 rate control을 이용해서 전송속도를 제어하는 경우의 성능을 비교하고 HSTP에서 rate control을 수행할 때 파라미터의 값을 어떻게 선정할 때 가장 좋은 성능을 보이는지를 측정하고자 한다. TP4는 rate control의 기능이 없다. 따라서 본절에서는 HSTP 방법 1을 시뮬레이션 대상으로 한다.

본 연구의 시뮬레이션 모델에 대하여 윈도우 흐름제어를 fluid model을 써서 모델링하면 그림 6에 나타낸 것과 같다. 그림 6에서 수신자의 피드백 정보를 송신자에게 연결하여 주는 가상큐 Q_v는 패킷도착률 λ로 송신자에게 패킷을 전달한다.

전송된 패킷이 에러없이 수신자에게 전달되고 수신자가 전달된 패킷을 모두 에러없이 받아 처리할 수 있을 때, 이 모델의 파라미터에 대한 식은 다음과 같다.

수신자의 내용 : $Q_r(t) = \int IN_r(t)dt - \int OUT_r(t)dt$

수신자의 서비스율 :

$$OUT_r(t) = \begin{cases} \mu_r & \text{if } Q_r(t) > 0 \\ \min(IN_r(t),\ \mu_r) & \text{otherwise} \end{cases}$$

수신자의 도착율 : $\mathrm{IN}_r(t) = OUT_s(t - T_p)$

송신자의 내용 : $Q_s(t) = \int IN_s(t)dt - \int OUT_s(t)dt$

송신자의 서비스율 :

$$OUT_s(t) = \begin{cases} \mu_s & \text{if } Q_s(t) > 0 \\ \min(IN_s(t),\ \mu_s) & \text{otherwise} \end{cases}$$

송신자의 도착율 : $\mathrm{IN}_s(t) = OUT_v(t)$

송신자의 내용 : $Q_v(t) = \int IN_v(t)dt - \int OUT_v(t)dt$

가상큐의 서비스율 :

$$OUT_v(t) = \begin{cases} \mu_v & \text{if } Q_v(t) > 0 \\ \min(IN_v(t),\ \mu_v) & \text{otherwise} \end{cases}$$

가상큐의 도착율 : $\mathrm{IN}_v(t) = OUT_r(t - T_p)$

초기조건 : $Q_v(t) = K_r$

위 식에서 $\mu_s > \mu_r$, $\lambda > \mu_r$이고 수신자의 버퍼 K_r이 $K_r/\mu_r \geq 2 \times T_p$를 만족하도록 충분히 크게 주어지면 최대처리율 γ는 다음과 같다.

$$\gamma = OUT_r(t) = \mu_r$$

그러나, 고속망환경에서 송신자의 속도가 수신자의 속도에 비해서 매우 빠르면 수신자에게 연속적으로 전달되는 패킷의 수는 대역폭과 전파지연의 곱의 증가와 비례하여 증가함으로써 수신자의 버퍼공간이 비어 있음에도 불구하고 수신자 시스템내에서의 스케쥴링 및 자원공유등으로 인하여 연속적으로 도착

하는 패킷들을 모두 받아놓지 못하고 손실되는 패킷이 존재하게 된다.

이와같이 연속적인 패킷도착에 의하여 수신자에서 손실되는 데이타의 양을 $Loss(t)$라고 하면 위 식의 일부는 다음과 같이 변한다.

수신자의 도착율 : $\mathrm{IN}_r(t) = OUT_s(t-T_p) - Loss(t)$

송신자의 도착율 : $\mathrm{IN}_s(t) = OUT_v(t) + Loss(t-T_p)$

따라서 $Loss(t)$에 의한 오버헤드로 인하여 처리율이 μ_r 이하로 감소하게 된다.

이같은 현상은 윈도우 흐름제어로는 전송가능한 데이타의 양만을 제어하고 데이타의 속도는 제어할 수 없는데서 발생한다. rate control은 송신자에서 전송되는 데이타의 전송속도 제어를 통해 수신자에서 처리할 수 없을 만큼의 고속 전송에 의한 에러손실을 막을 수 있어서 위식의 $Loss(t)$를 0이 되게 함으로써 최대처리율을 μ_r로 유지할 수 있을 것이다.

본 연구에서 rate control을 시뮬레이션하기 위한 파라미터와 시나리오는 다음과 같다.

송신자에서 연속된 패킷의 고속전송으로 인하여 수신자에서 손실이 발생하기 위해서는 송신자의 처리속도가 수신자의 처리속도보다 클 때 발생한다. 이와같은 상황을 모델링하기 위해서 송신자의 패킷처리시간은 $0.1ms$이고 수신자의 패킷처리시간은 $0.15ms$으로 한다. $1K$ 옥텟의 TPDU를 전송하는 데 걸리는 시간은 약 $0.078ms$이다. 이때, 송신자와 수신자의 서비스율 μ_s와 μ_r은 다음과 같다.

$$\mu_s = \frac{1\ packets}{(0.1+0.078)\ ms} \fallingdotseq 5.6\ packets/ms$$

$$\mu_r = \frac{1\ packets}{(0.15+0.078)\ ms} \fallingdotseq 4.4\ packets/ms$$

패킷 전송로상에서 서비스율이 가장 낮은 것이 μ_m일때, 윈도우 흐름제어에서 처리율이 최대가되기 위해서는 버퍼의 크기 K_r이 적어도 $2\times\mu_m\times T_p$보다는 커야한다. 본절에서는 윈도우 흐름제어의 성능이 최상일 때, rate control의 성능을 측정, 비교하기 위하여 버퍼의 크기 K_r이 $\mu_m\times T_p$의 2.5배인 $220K$ 옥텟으로 한다.

트래픽 부하 ρ는 0.5로 한다. 이때 λ는 다음과 같다.

$$\lambda = \frac{0.5}{0.078ms} \fallingdotseq 6.4\ packets/ms$$

따라서 λ, μ_s, μ_r 사이의 관계는 다음과 같다.

$$\lambda > \mu_s > \mu_r$$

전파지연이 충분히 길고 수신자가 송신자에서 전송된 패킷을 모두 받지 못하고 손실이 발생할 때, 윈도우 흐름제어만으로 흐름제어를 했을 때의 최대처리율과 윈도우 흐름제어와 더불어 rate control을 *burst*의 값을 바꿔가면서 수행했을 때의 최대처리율을 시뮬레이션한 결과가 그림 7-8이다. 그림에서 *burst*가 0일 때는 윈도우 흐름제어만으로 흐름제어를 했을 경우의 값을 나타낸 것이다.

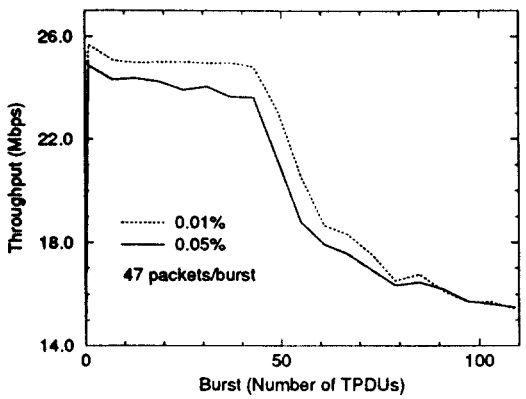


그림 7. 수신자가 $10ms$동안 47개 이상의 패킷을 받을 수 없을때, 버스트 변화에 대한 최대처리율

Figure. 7. burst vs. maximum throughput(receiver receives maximum 47 packets per $10ms$)

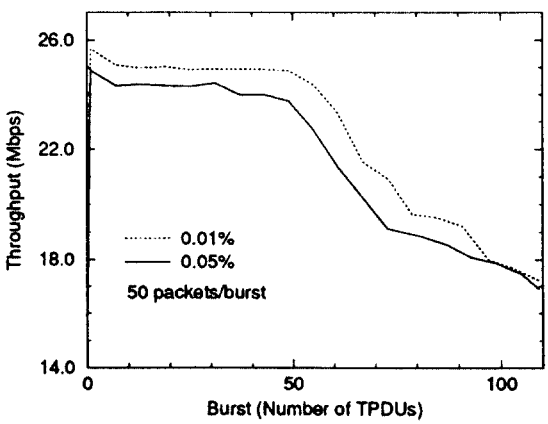


그림 8. 수신자가 $10ms$동안 50개 이상의 패킷을 받을 수 없을 때, 버스트 변화에 대한 최대처리율

Figure. 8. burst vs. maximum throughput(receiver receives maximum 50 packets per $10ms$)

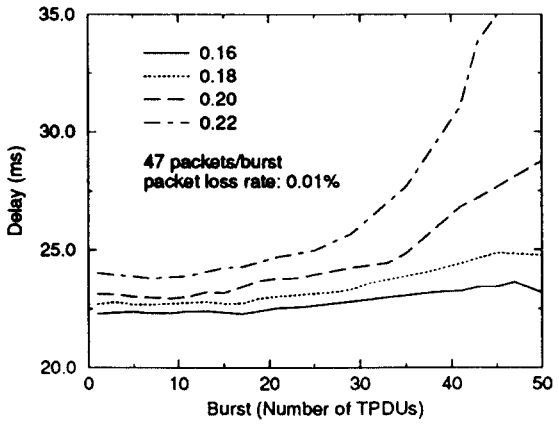


그림 9. 패킷손실율이 0.01%일때 트래픽 부하에 따른 버스트와 전송지연

Figure. 9. burst vs. delay(packet loss rate = 0.05%)

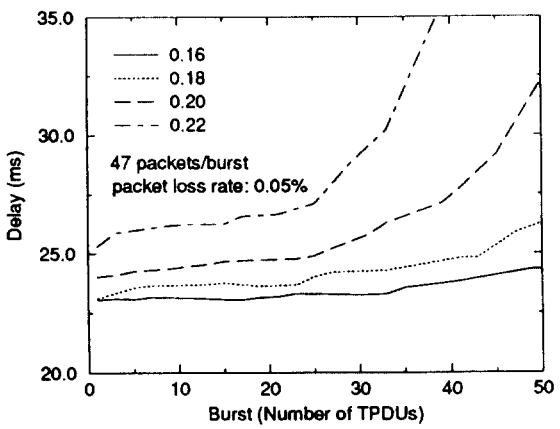


그림 10. 패킷손실율이 0.05%일때 트래픽 부하에 따른 버스트와 전송지연

Figure. 10. burst vs. delay(packet loss rate = 0.05%)

송신자와 수신자가 10*ms*동안 처리할 수 있는 패킷의 수는 56개와 44개이다. 이동안 수신자는 47개와 50개씩의 패킷만을 버퍼안에 받아들일 수 있다. 윈도우 흐름제어에서는 송신자가 크레디트(credit)만 있으면 언제든지 데이타를 전송할 수 있다. 따라서 송신자가 일정시간동안 수신자가 받아놓을 수 있는 패킷수보다 많은 양의 데이타를 전송하면 수신자는 이동안 44개의 패킷은 처리하여 내보내고 각각 3개와 6개는 버퍼링하고 그 이상의 것은 처리할 수 없어서 손실되게 된다. HSTP와 rate control을 이용하여 송신자가 일정시간동안 전송하는 패킷의 수를 수신자가 받아 놓을 수 있는 패킷의 수로 제한하여 전송하면 패킷손실을 방지할 수 있다.

HSTP에서 *rate*는 초당 전송가능한 최대의 데이타 양이다. 따라서 *rate*의 값은 상위계층에서 요구하는 전송률로 선정할 수도 있고 연결설정시에 전송로상에서의 최대전송률로 선정할 수도 있다. 그림 7-10의 실험에서 $\mu_m = \mu_r = 4400(packets/ms)$이므로 *rate*의 값은 4400 패킷에 해당하는 크기로 선정한다.

*rate*의 값을 전송로상에 최대처리율로 선정하고 패킷손실율이 0.01%일 때와 0.05%일 때, 버스트의 크기를 변경시키면서 성능이 가장 좋을 때의 버스트 값을 알아내고자 한다. 버스트의 크기는 그림 7, 8에서는 1개의 패킷크기에서부터 송신자와 수신자사이에서 전송될 수 있는 최대패킷수인 $\mu_s \times T_p = 112$개의 패킷크기까지 변경시키면서 결과를 측정하였다. 시나리오에서 10*ms*동안 수신자가 받을 수 있는 최대 패킷수를 47개와 50개로 하였는데 버스트의 크기가 이때보다 커질수록 성능이 급격히 감소하여 결국에는 윈도우 흐름제어(버스트=0)만으로 흐름제어를 했을 때의 성능과 비슷해지고, 버스트의 크기가 이보다 작을 때는 거의 비슷한 값을 갖는 최대의 성능을 나타낸다. 실제 환경에서 송신자가 수신자가 일정시간 동안 얼마만한 데이타를 수신할 수 있는지를 알기 어려우므로 성능이 급격히 떨어지는 기준점을 알기는 어려울 것이다. 그런데 버스트의 크기가 이러한 기준점보다 작을 때는 대체로 비슷한 성능을 유지하면서 $Loss(t)$를 최소화할 수 있다는 것을 알 수 있다. 이는 버스트의 크기는 작을수록 *rate*에서 주어진 데이타 양을 1초동안 치우침없이 전송할 수 있으므로 과도한 밀집전송을 피할 수 있기 때문이다.

그림 9, 10에서는 1개에서 50개까지의 패킷크기를 버스트 값으로 갖는 경우에 트래픽 부하 변화에 따른 전송지연을 측정한 결과이다. 버스트의 값이 50보다 클 때는 그림 7, 8의 결과에서 성능이 급격히 감소하므로 제외했다. 결과에서 트래픽 부하 ρ가 0.16, 0.18일 경우에는 버스트 크기와 관계없이 거의 비슷한 전송지연을 갖지만 ρ가 0.2, 0.22로 커질수록 버스트 크기의 증가는 데이타의 연속적인 전송으로 인한 패킷손실을 유발하므로 재전송에 필요한 오버헤드로 인하여 전송지연이 증가한다. 이를 통해 트래픽 부하가 작을 경우에는 윈도우 흐름제어나 *burst*의 값이 큰 rate control을 할 경우에도 패킷이 연속적으로 밀집되게 전송될 확률이 희박하므로 윈도우 흐름제어를 사용할 때와 rate control을 이용했을 때 성능의 큰

차이가 없다는 것을 알 수 있다. 그러나, 트래픽 부하가 증가하면 윈도우 흐름제어나 *burst*의 값이 큰 rate control을 할 경우에 패킷이 연속적으로 밀집되게 전송될 확률이 증가하므로 이로 인한 패킷손실 발생 확률이 커져 성능이 감소한다. 따라서 이 경우에는 *burst*의 값을 작게 하여 rate control을 하므로써 패킷의 밀집전송을 회피하여 성능을 향상시킬 수 있다.

그림 7-10의 결과를 통해 수신자의 처리속도에 비해서 과도하게 높은 처리속도를 갖는 송신자가 있는 고속망환경에서는 윈도우 흐름제어와 더불어 rate control을 이용하여 전송속도를 제어하므로써 성능을 향상시킬 수 있다는 것을 확인할 수 있다. 또한 rate control을 하여 최적의 성능을 얻기 위해서는 *rate*의 값은 상위계층에서 요구하는 최대처리율로 선정하고 *burst*는 작은 수의 패킷을 선정하는 것이 효율적인데 실험의 결과에 따르면 패킷 하나 크기로 선정하면 데이타의 고속전송으로 인한 손실을 최소화하여 성능의 향상을 얻을 수 있다.

그림 11에 나타난 결과는 패킷손실율이 0.05%이고 수신자가 10*ms*동안 50개이하의 패킷을 받을 수 있을 때, 윈도우 흐름제어만을 사용했을 때와 윈도우 흐름제어와 더불어 버스트의 크기가 패킷크기 하나인 rate control을 사용했을 때의 최대성능을 보인 것이다. 곡선중에 파선은 똑같은 환경에서 수신부에서 수신되는 모든 패킷은 일단 받을 수 있을 때의 성능이다.

송수신 속도차이에 의한 수신부에서의 패킷손실이 없을 때는 본절에 앞부분 수식중에 $Loss(t)$가 0이므로 윈도우 흐름제어로 흐름제어할 때, $K_r/(\mu_m \times T_p)$이 2가 될 때까지 처리율이 증가하다가 2보다 클때 최대처리율을 보인다. 그러나, 송수신 속도차이에 의하여 수신부에서의 패킷손실이 발생할 경우에는 $Loss(t)$에 의하여 윈도우 흐름제어만으로 흐름제어를 할 경우에는 $K_r/(\mu_m \times T_p)$가 2가 될 때까지 성능이 증가하여 가능한 최대처리율을 보이지 못하고 값이 2가 되기전에 처리율이 정체되게 된다. 반면에 윈도우 흐름제어와 더불어 rate control로 직접적으로 송신자에서 수신자로 데이타 전송속도를 제어할 경우에는 $Loss(t)$를 최소화 또는 0으로 유지할 수 있으므로 데이타 손실이 없을 때, 윈도우 흐름제어를 할 때와 같은 결과를 얻을 수 있다.

대역폭 × 전파지연의 값이 큰 고속망에서 송신자의 전송속도 μ_s가 수신자의 처리속도 μ_r에 비해 클 경우에는 데이타의 전송속도를 직접적으로 제어할 수 있는 메카니즘없이 윈도우 흐름제어로 흐름제어를 하면 수신자가 충분한 버퍼공간을 갖고 있더라도 μ_s로 도착하는 패킷을 수신하지 못하고 손실되는 상황이 발생한다. 손실된 패킷의 재전송으로 인한 오버헤드 때문에 송수신자간의 연결에서 얻을 수 있는 최대처리율 μ_r을 얻을 수 없게 된다. 그러나, rate control을 이용하여 데이타의 전송속도를 직접적으로 제어할 경우에는 송신자가 전송속도를 직접 제어할 수 있으므로 $Loss(t)$를 0으로 유지하여 최대처리율 μ_r을 얻을 수 있을 것이다.

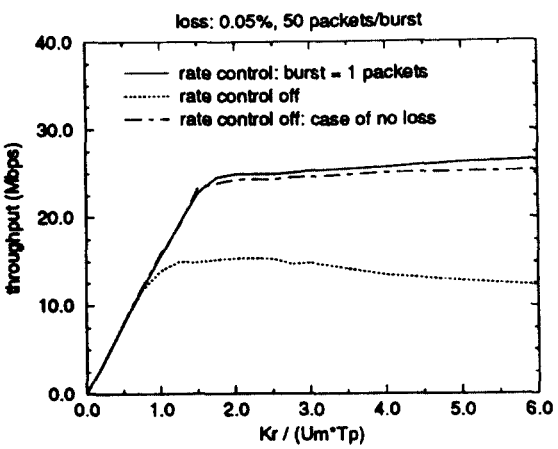


그림 11. $K_r/(\mu_m \times T_p)$ 변화에 따른 최대처리율
Figure. 11. $K_r/(\mu_m \times T_p)$ vs. maximum throughput

Ⅵ. 결 론

본 연구의 목적은 고속망환경에서 구조적으로 HSTP가 기존 프로토콜인 TP4보다 성능 향상을 보이는지의 여부와 rate control의 효용성을 시뮬레이션을 통한 성능평가로 검토하는 것이다.

대역폭 × 전파지연의 값이 크고 수신자가 처리할 수 있는 속도이상으로 송신자가 데이타를 전송할 수 있는 고속망환경에서는 selective repeat 에러 재전송과 순서에 어긋나게 도착한 패킷을 저장하였다가 재순서화하는 에러제어 기법과 윈도우 흐름제어와 더불어 데이타 전송속도를 직접적으로 제어하는 rate control 기법을 제공하는 HSTP가 다른 프로토콜 전략을 갖는 HSTP나 기존의 TP4보다 성능 향상을 보인다는 것을 시뮬레이션에 의한 결과로 확인했다.

송신자가 수신자에 비해서 월등히 빠른 처리속도를 가졌고 대역폭 × 전파지연의 증가로 인한 수신자

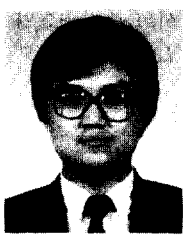
에서의 패킷손실이 일어나는 고속망환경에서 윈도우 흐름제어와 더불어 *rate*의 값을 트랜스포트 연결상에 최대처리율이나 상위계층에서 요구하는 최대전송률로 설정하고 패킷하나 크기에 해당하는 *burst* 값으로 rate control을 제공하는 HSTP가 고속의 데이타 전송으로 인한 수신부에서의 패킷손실을 최소화하여 처리율과 전송지연에 있어서 효율적인 성능을 보인다.

본 연구에서는 HSTP와 TP4에 대한 구현모델은 고려하지 않고 비교하였는데, HSTP는 병렬처리를 최대화할 수 있도록 설계된 프로토콜이므로 이를 반영한 HSTP 구현에서는 본 시뮬레이션 연구에서 보여진 것보다 훨씬 나은 성능을 보일 것으로 예측된다.

참 고 문 헌

1. W. Doeringer, D. Dykeman, M. Kaiserswerth, B. Meister, H. Rudin, R. Williamson, "A survey of light-weight transport protocols for high-speed networks," *IEEE Trans. Commun.*, vol.COM-38, pp.2025-2039, Nov. 1990.
2. USA, ISO/IEC JTC1/SC6 N : SD4-028, "High speed transport protocol, working draft," San Diego, July 1992.
3. USA, ISO/IEC JTC1/SC6 N : SD4-026, "High speed transport service (HSTS)," San Diego, July 1992.
4. Transport protocol specification for open systems interconnection for CCITT applications, CCITT Recommendation X.224, 1988.
5. Transport service definition for open systems interconnection for CCITT applications, CCITT Recommendation X.214, 1988.
6. Protocol Engines, Inc., XTP Protocol Definition, Revised 3.6, Jan. 1992.
7. R. Sanders, A. Weaver, "The Xpress Transfer Protocol(XTP)-a tutorial," *ACM SIGCOMM*, 1991.
8. ISO/IEC JTC1 : IPS-DC-Network Service Definition, ISO 8348.
9. ISO/IEC JTC1 : IPS-DC-Network Service Definition, Addendum 1 : Connectionless-Mode Transmission. ISO 8348 ADD1. 1987.
10. C. A. Eldridge, "Rate controls in standard transport layer protocols," *SIGCOMM CCR*, pp. 106-120, July 1992.

朴 章 然(Jangyeon Park) 정회원
1968년 4월 20일생
1992년 2월 : 서울대학교 공과대학 컴퓨터공학과(학사)
1992년 ~ 현재 : 서울대학교 공과대학 컴퓨터공학과(석사과정)



崔 陽 熙(Yanghee Choi) 종신회원
1955년 7월 27일생
1975년 2월 : 서울대학교 공과대학 전자공학과(학사)
1977년 2월 : 한국과학원 전기 및 전자공학과(석사)
1984년 6월 : 프랑스 국립 전기통신대학 전산과(공학박사)
1977년 ~ 1979년 : 한국전기통신연구소 연구원
1981년 ~ 1984년 : 프랑스 국립 전기통신연구소 연구원
1988년 ~ 1989년 : IBM Thomas J. Watson Research Center 방문연구원
1984년 ~ 1991년 : 한국전자통신연구소 책임연구원
1991년 ~ 현재 : 서울대학교 컴퓨터공학과 조교수
서울대학교 중앙교육연구전산원 부원장