論文2003-40TC-8-4

네트워크 효율 향상을 위한 개선된 TCP 혼잡제어 알고리즘

(Modified TCP Congestion Control Algorithm to Improve Network Efficiency)

崔 址 賢 *, 金 大 永 *, 金 觀 雄 **, 鄭 炅 澤 ***, 田 炳 實 *

(Ji-Hyun Choi, Dae-Young Kim, Kwan-Woong Kim, Kyung-Taek Chung and Byoung-Sil Chon)

요 약

본 논문에서는 혼잡윈도우(CWnd) 와 RTT 파라메터를 사용해 흐름을 제어하는 개선된 TCP 혼잡제어 알고리즘을 제안한다. 제안된 알고리즘에서 혼잡윈도우를 TCP의 상태에 따라 저장된 RTT값으로 제어하여 혼잡상태를 회피한다. 또한 CWnd 값의 변동율을 줄이고 패킷 손실율을 낮춰서 네트워크 효율을 증대시킬 수 있다. 시뮬레이션을 수행한 결과 제안된 알고리즘이 버퍼 이용율과 네트워크 효율면에서 기존 TCP 알고리즘보다 우수한 성능을 보였다.

Abstract

In this paper, we propose an modified TCP congestion control algorithm using estimated RTT with congestion window parameter CWnd. Congestion window size is controlled with memorized RTT value on the congestion status. It can avoid occurrence of frequent congestion and reduce CWnd fluctuation. From simulation results, proposed algorithm shows great improvement on network efficiency and buffer utilization compared with original TCP algorithm.

Keywords : TCP, Congestion control, Slow-start, Internet

Ⅰ. 서 론

TCP/IP 프로토콜은 다양한 크기의 서로 상이한 제조업체가 만든 컴퓨터와 서로 다른 운영체제간의 통신을 가능하게 하였다. 이는 초기 목적과 당시의 예상을 훨씬 뛰어넘는 것으로 1960년대 후반 패킷 교환 네트워크의 연구 프로젝트로 시작한 것이 1990년대 들어서는 컴퓨터를 서로 연결하는 일반적인 통신 형식으로 폭 넓게 사용되게 된 것이다[1]. 이더넷(Ethernet)과 같은 대부분의 TCP/IP 기반 네트워크는 "Best-effort" 서비스를 제공하기 위한 목적으로 설계되었다. "Best-effort" 서비스는 자원이 허용하는 한 최대한 빨리 데이터를 전송하는 서비스이다.

현재 TCP 분야에서의 주된 연구로 TCP가 가지는 특성을 가능한 유지하고 기존의 TCP와의 호환성을 유지하며 이를 확장시키는 방안을 들 수 있는데 그 예로 TCP의 저 대역폭의 서비스 문제를 해결하기 위해 옵션 필드를 확장하여 TCP 윈도우의 크기를 늘리거나,

* 正會員, 全北大學校 工科大學 電子情報工學部
(Division of Electronic and Information Eng., Chonbuk Nat'l Univ.)
** 正會員, 韓國標準科學硏究院 人間情報그룹 Post. Doc.
(Korea Research Institute of Standards and Science)
*** 正會員, 群山大學校 工科大學 電子情報工學部
(Division of Electronic and Information Eng., Kunsan Nat'l Univ.)
接受日字:2003年6月7日, 수정완료일:2003年8月16日

RTT(Round Trip Time) 측정의 정확도를 향상시키는 방법들을 들 수 있다. 그 외 TCP의 프로토콜 기능 중 혼잡 제어 기능을 성실히 수행하도록 함으로서 패킷 손실로 인한 성능 감소의 효과를 줄이려는 방법이 연구되고 있는데 이는 크게 두 가지로 나누어 생각할 수 있다. 먼저 TCP의 혼잡제어 기능을 강화시켜 통신망내의 혼잡을 미리 예측하는 방법을 들 수 있고 다른 하나는 중간 노드들의 기능을 강화하여 TCP의 빈약한 혼잡 제어 때문에 발생할 수 있는 TCP의 글로벌 동기화 문제를 해결하고 조기의 혼잡 통지를 통해 불필요한 패킷의 손실이나 패킷 지연을 방지하는 것이다[2, 3].

본 논문에서는 IPoverATM 망의 운용에서 위에서 설명한 바와 같이 서비스 품질을 보장하지 않는 "Best effort" 방식인 ATM 클래스들 중 UBR 클래스를 사용하는 상황에서 TCP 계층에서의 흐름제어 시스템의 개선을 통해 네트워크의 수율을 향상시키기 위한 방안을 제안하고자 한다.

TCP에서 널리 사용되는 혼잡제어 방식은 slow-start와 congestion avoidance이다[4]. 제안된 알고리즘에서는 기존의 TCP의 혼잡제어 방식에 더하여 혼잡발생시의 RTT를 기억하고 이를 다음 혼잡상황 방지를 위한 파라미터로 사용한다. 초기 패킷 전송 후 목적지로부터 ACK가 오면 RTT를 알 수 있는데 네트워크에서 혼잡이 발생하여 큐에 셀이 쌓이면 셀이 전송되는 시간이 늘어나며 자연적으로 RTT도 증가하게 된다.

제안된 방식에서는 패킷이 손실되어 재전송되면 이때 가장 긴 RTT 값을 기억하여 다음 전송부터는 RTT가 혼잡발생시의 가장 긴 RTT 값을 넘지 않도록 제어하는 것에 기초 한다. 이에 따라 네트워크 상에서 패킷의 버스트한 흐름에 따라 TCP의 혼잡제어 메커니즘이 실행되어 계속적으로 발생하는 네트워크의 수율 변화를 혼잡상태에서의 상황을 기억하여 이후의 전송에서는 혼잡상황 발생 이전에 이를 차단해 이에 따른 패킷 손실을 방지하여 안정적인 네트워크 수율을 확보할 수 있게 된다. 본 논문은 2장에서 표준화된 TCP흐름제어와 문제점을 설명하고 3장에서 개선된 TCP 혼잡제어 알고리즘을 제안한다. 4장에서는 제안된 알고리즘의 성능을 컴퓨터 시뮬레이션을 통해서 평가하고 5장에서 결론을 맺는다.

Ⅱ. TCP의 흐름제어 기법

TCP는 흐름제어와 재전송을 수행하여 비연결성 지향인 IP가 가지는 신뢰성 없는 전송을 보완할 수 있어 현재 인터넷을 통한 데이터 전송에 광범위하게 사용된다.

원래의 TCP의 흐름제어는 slow-start와 congestion avoidance의 두 방식을 사용하여 제어하는데 TCP 연결이 설정되면, 흐름제어에 사용되는 두 변수 혼잡 윈도우(CWnd)와 slow-start 임계치(Wth)를 사용한다. 이 알고리즘의 전제는 패킷이 손상에 의한 손실이 매우 적다(1%보다 작음)는 것이다. 그러므로 패킷의 손실은 발신지와 목적지 사이의 네트워크 어딘가에 혼잡이 발생하고 있다는 것을 나타내는 신호가 된다. 패킷 손실을 나타내는 것은 2가지이다. 재전송 타이머의 만료와 중복된 ACK의 수신이다. congestion avoidance와 slow-start는 목적이 서로 다른 독립적인 알고리즘이다. 그러나 혼잡이 발생할 때 네트워크에서 전달되는 패킷의 전송율을 감소하기 위해 slow-start를 이용하며 현실적으로 이들 두 가지 기법들은 함께 구현되어 있다.

TCP 송신 측에서 최초 세그먼트 전송시 CWnd의 크기는 1-세그먼트부터 설정한다. 수신 측으로부터 ACK 세그먼트를 받을 때마다 윈도우 크기를 지수적으로 증가시킨다. 중복된 ACK 신호를 받거나 재전송 타이머가 만료되면, 혼잡상황이 발생한 것으로 간주하여 CWnd를 줄여서 전송률을 감소시킨다. 중복된 ACK 세그먼트를 받는 경우는 전송된 세그먼트의 일부분이 손실된 것으로 심각한 혼잡상태는 아니므로, congestion avoidance를 수행한다. 재전송 타이머가 만료된 경우는 전송한 모든 세그먼트가 유실되거나 수신측에서 전송한 ACK 세그먼트가 유실된 경우로, 이는 혼잡상황이 심각하다고 판단하고, slow-start를 수행한다. CWnd가 Wth에 도달

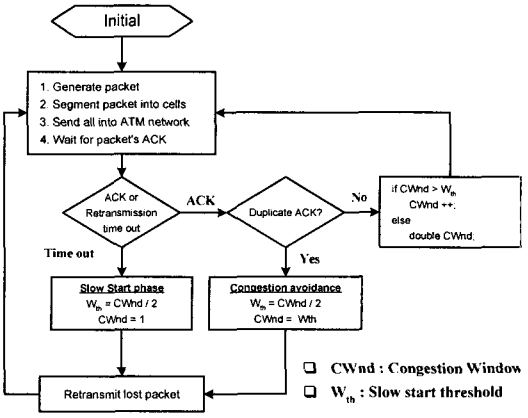

그림 1. TCP 혼잡제어 알고리즘 흐름도
Fig. 1. Flow chart of TCP congestion control algorithm.

하면, congestion avoidance를 수행한다[1 9]. <그림 1>은 TCP의 혼잡제어 알고리즘의 흐름도이다.

기존 TCP의 흐름제어 과정을 보면 중복 ACK의 발생 없이 정상적인 전송이 이루어진 경우 CWnd가 지수적으로 상승하게 되고 데이터의 전송률이 급격히 증가한다. 이에 의한 버퍼의 임계값 초과나 과다한 TCP source의 전송, 회선상태 등에 따라 중복 ACK나 Timer out에 의한 패킷 손실이 감지되면 일련의 TCP의 Congestion control이 동작하게 되고 전반적인 네트워크의 전송률이 급감하게 된다. <그림 2>는 이러한 기존 알고리즘의 반복적인 흐름제어를 보여준다.

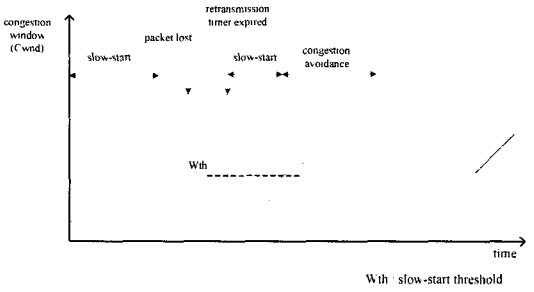

그림 2. slow-start와 congestion avoidance 메커니즘
Fig. 2. The slow-start and congestion avoidance mechanism.

Ⅲ. 개선된 TCP 흐름제어 알고리즘

TCP의 타임아웃과 재전송은 연결에 따른 RTT 측정을 기본으로 한다[7, 9, 10]. 따라서 TCP는 트래픽과 경로의 변화에 따른 시간변화를 감지하고 적절한 타임아웃 값을 유지해야 한다. 이를 위한 파라미터로 RTT 값을 사용하며 적당한 순서번호를 가진 바이트의 송신으로부터 ACK가 돌아올 때까지의 시간을 측정하게 된다.

RTT는 링크지연시간, 패킷 처리시간에 따른 패킷의 회신 시간과 큐잉 지연시간에 의해 결정된다. TCP 패킷의 버스트한 흐름은 스위치의 버퍼에 누적되는 데이터 양을 급작스럽게 증가시키고 이를 처리하기 위한 큐잉 시간의 연장으로 인해 발생하는 전송 패킷의 손실을 유발한다.

기존의 알고리즘은 위와 같은 패킷의 손실과 전송상의 문제점으로 인해 야기되는 유실에 따른 다양한 효율 감소 요인들을 비롯해 네트워크 내부의 혼잡이나 혼잡상태를 상황의 발생 후 이를 처리하는 소극적인 제어방식임에 비해 제안된 알고리즘에서는 혼잡발생시의 RTT를 기억하여 혼잡윈도우(CWnd)의 크기를 조절해 이를 다음 혼잡상황 방지를 위한 파라미터로 사용하는 보다 적극적인 제어방식이라 할 수 있다. <그림 3>는 제안된 알고리즘의 흐름도이다.

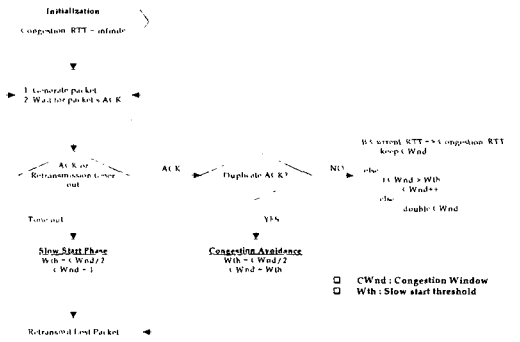

그림 3. 제안된 TCP 흐름제어 알고리즘의 흐름도
Fig. 3. Flow chart of proposed TCP flow control algorithm.

제안된 알고리즘에서는 효율적인 네트워크 성능향상을 위하여 아래의 RTT 파라미터들을 사용한다.

(1) Congestion_RTT : 혼잡 발생시의 RTT를 의미하며 패킷의 손실이 네트워크가 혼잡상황에서 발생한다는 가정하에서 혼잡상황의 재발생을 피하기위해 사용하는 파라미터이다. 초기상태는 ∞이다.
(2) Max_RTT : 수신된 RTT중 최대값을 의미하며 재전송 타이머의 설정을 위해 일반적으로 사용되는 파라미터이다.
(3) Current_RTT : 현재 수신되는 ACK 패킷의 RTT 값을 의미한다.
(4) Wth : slow-start 임계치
(5) CWnd : congestion window

다음은 네트워크의 효율향상을 위해 제안된 알고리즘의 TCP의 흐름상황에 따른 CWnd 설정 프로시져이다.

```
1. 패킷을 정상 수신했을 경우
TCPSender_ReceivePacket(Packet)
 Current_RTT = Packet.RTT;
 Max_RTT = max(Current_RTT, Max_RTT)
```

송신측으로부터 전송된 패킷을 해당 링크에 접속된 수신측에서 정상적으로 수신했을 경우 Current_RTT 값을 수신된 패킷의 RTT 값으로 수정, 유지한다.

Congestion_RTT와 재전송 타임아웃을 설정하기 위한 Max_RTT 역시 수신된 패킷의 RTT와 비교하여 최대 값으로 갱신한다.

2. Slow-start 상태인 경우
if (Current_RTT => Congestion_RTT)
keep CWnd
else increase CWnd, Wth normally

Slow-start 상황에서는 수신된 현재 RTT 값을 Congestion_RTT 값과 비교하여 현재의 RTT 값이 혼잡 발생시의 RTT 값 이상일 경우 CWnd의 증가를 방지한다. 그렇지 않은 경우는 기존 방식과 같이 처리한다.

3. Congestion avoidance 상태인 경우
if (current_RTT => cong_RTT)
keep CWnd
else increase CWnd, Wth normally

Slow-start의 경우와 마찬가지로 Congestion_RTT가 현재의 수신된 RTT값보다 큰 경우에만 CWnd를 증가시킨다.

4. Retransmit Timer 가 만료된 경우
set Wth = 1, CWnd = 1
set state to slow-start

Timer out 발생시 Slow-start를 실행하고 Wth와 CWnd를초기화 하게 되며 네트워크 흐름 상태는 slow-start로 설정되며 손실된 패킷을 재전송한다.

5. Three duplicate ACK 가 수신된 경우
Congestion_RTT = max_RTT;
max_RTT = 0.0;
reset max_RTT to base_RTT
set state to congestion avoidance

세 개의 중복 ACK의 수신은 혼잡상태를 의미한다. 따라서 혼잡상태의 RTT를 기억하기 위한 Congestion_RTT 값을 이때의 최대 RTT 값으로 갱신하고 Max_RTT 값을 초기화 한다.

초기 패킷 전송 후 목적지로부터 ACK가 도착하면 패킷으로 부터 RTT를 계산한다. 네트워크에서 혼잡이 발생하여 큐에 셀이 쌓이면 셀이 전송되는 시간이 늘어나며 자연적으로 RTT도 증가하게 된다. 제안된 방식에서는 위에서 설명한 흐름상황에 따른 처리 규정에 따라 패킷이 손실되어 재전송되면 이때의 가장 긴 RTT 값을 기억하여 다음 전송부터는 RTT가 혼잡발생시의 가장 긴 RTT 값을 넘지 않도록 한다.

이에 따라 네트워크 상에서 패킷의 버스트한 흐름에 따라 TCP의 혼잡제어 메커니즘이 실행되어 계속적으로 발생하는 네트워크의 전송률 변화를 혼잡상태에서의 상황을 기억하여 이후의 전송에서는 혼잡상황 발생 이전에 이를 방지해 이에 따른 패킷 손실을 방지하여 안정적인 네트워크 수율을 확보할 수 있게 된다.

Ⅳ. 성능 평가

■ 시뮬레이션 시나리오

제안된 알고리즘의 성능평가를 위해 C++ 로 작성된 Component 기반의 IP/ATM 시뮬레이터를 사용하였다[11]. IP 라우터와 ATM 스위치와 직접 연결된 peer-to-peer 모델로 두 대의 ATM 스위치에 6대의 IP 라우터가 결합된 네트워크로 <그림 4>와 같다. 각 IP 라우터는 3개의 TCP 연결이 연결되어 있고 각 TCP는 greedy 소스로 무한 크기의 파일을 전송한다.

TCP의 MTU(Maximum Transfer Unit)는 기본 960 바이트로 20 개의 ATM셀을 이루며, TCP의 시뮬레이션 파라미터는 <표 1>과 같다.

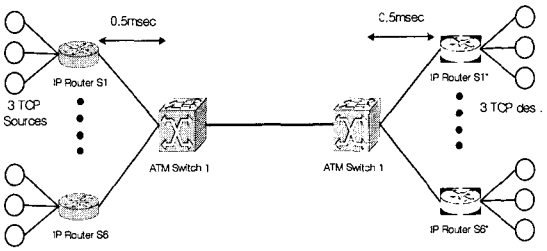

그림 4. 시뮬레이션 네트워크 모델
Fig. 4. Simulation network model.

네트워크의 모든 링크는 SONET OC-1인 51.84 Mbps이며 ATM 스위치간의 지연시간은 5/10/15/20 msec이고, IP 라우터와 ATM 스위치간의 지연시간은 0.5msec이다. <그림 4>의 왼쪽에 위치한 IP 라우터의 TCP 연결은 송신원이고, 오른쪽에 대응하는 워크스테이션의 TCP 연결은 수신원이고 각 워크스테이션 쌍은 하나의 UBR VC로 연결이 된다 <표 2>는 UBR VC의 트

표 1. 시뮬레이션 파라미터
Table 1. Initial simulation parameter.

Parameter	Value
ATM switch type	EPD
EPD threshold	400
Buffer size	500
Window Size	65,535 bytes
Retransmission Timer	50 msec
Maximum Segment Size	960 bytes
File size	infinite(∞)

표 2. UBR VC 파라미터
Table 2. characteristics of the UBR VCs.

UBR VCs	PCR	Router-router delay
VC 1 (S1 ↔ S1*)	51.84 Mbps	5/10/15/20 msec
VC 2 (S2 ↔ S2*)	51.84 Mbps	5/10/15/20 msec
VC 3 (S3 ↔ S3*)	51.84 Mbps	5/10/15/20 msec
VC 4 (S4 ↔ S4*)	51.84 Mbps	5/10/15/20 msec
VC 5 (S5 ↔ S5*)	51.84 Mbps	5/10/15/20 msec
VC 6 (S6 ↔ S6*)	51.84 Mbps	5/10/15/20 msec

래픽 파라미터를 보여준다.

■ TCP의 CWnd 추이

<그림 5>은 TCP 송신측의 시간에 따른 기존 알고리즘의 혼잡 윈도우의 추이를 보여준다. ATM 스위치에서 패킷 손실이 일어날 때 마다 TCP는 혼잡제어 메커니즘을 수행하고, 혼잡 윈도우의 크기를 줄이고 다시 혼잡 윈도우의 크기를 점진적으로 늘리는 과정을 반복하므로 혼잡 윈도우는 톱니 모양의 진행 패턴을 보인다.

반면 <그림 6>의 제안된 알고리즘이 보여주는 혼잡 윈도우의 추이를 보면 네트워크에서 급격한 전송량 증가 등에 따른 혼잡상황이 발생한 경우, 이때의 RTT를 기억하여 다음 혼잡 상태 직전에 혼잡 윈도우의 증가를

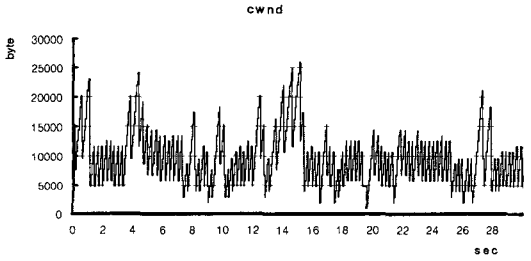

그림 5. 기존 알고리즘의 CWnd 추이
Fig. 5. CWnd of present algorithm.

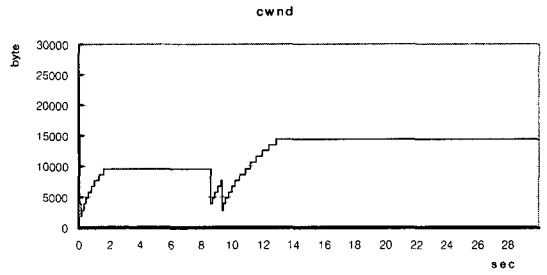

그림 6. 제안된 알고리즘의 CWnd 추이
Fig. 6. CWnd of proposed algorithm.

방지하므로 혼잡윈도우가 증가함으로서 급격히 커질수 있는 전송량을 사전에 방지하여 재 혼잡 상황을 감소시키므로 시간의 흐름에 따라 안정적인 혼잡 윈도우의 상태를 찾아감을 확인 할 수 있다. 예로 보여진 혼잡 윈도우의 추이는 양측에서 평균치에 근사한 결과를 보여준 source중 선정하였다.

■ 스위치 버퍼추이

<그림 7>부터 <그림 12>는 ATM 스위치에서의 버퍼 점유 추이를 영역형 그래프를 통해 보여준다. <그림 7>와 <그림 8>처럼 ATM 스위치의 지연시간이 5 ms인 경우 기존 알고리즘과 제안된 알고리즘 양측 모두에서 전송속도에 따른 지연이 적으므로 원활하게 셀의 이동이 진행되어 전체적으로 조밀한 상태의 효율적인 버퍼 이용 상황을 보이며, 제안된 알고리즘에서는 연결 후 8 초정도의 시간이 흐른 뒤에 혼잡윈도우가 안정상태를 유지하여 거의 완전한 버퍼 점유 상태를 보여주고 있다.

<그림 9>부터 <그림 12>에서는 10 ms, 20 ms까지 스위치 간 지연시간이 증가함에 따라 혼잡상황과 이에 따른 복구 지연에 걸리는 시간의 증가로 인해 조밀한 버퍼 이용을 보여주지 못하고 점차 버퍼의 공백상태가 길어짐을 파악할 수 있는데 10 ms의 스위치 간 지연시간에서 제안된 알고리즘은 약 17 초가 지난 후에는 안정된 혼잡윈도우 상태를 유지하여 다시 조밀한 버퍼 점유 상태를 보여준다. 스위치 간 지연시간이 20 ms인 경우, 버퍼의 점유 상태는 데이터의 전송과 처리시간의 전체적인 지연시간의 상승에 따른 결과로 매우 조밀하지 못한 모습을 보여주는데 제안된 알고리즘은 이 경우에 혼잡상태 재 발생 방지기능이 동작함으로서 기존의 알고리즘에 비해서는 복구시간이 길어짐에 따른 비효율적인 버퍼 낭비 현상을 어느 정도 완화시켜 주고 있음을 알 수 있다.

스위치의 버퍼추이는 1/1000초 단위로 측정하였으므로 조밀한 버퍼점유상태의 경우 실제로는 인접구간의 간격이 좁아진 것으로 보다 원활한 버퍼 점유의 변화를 의미한다.

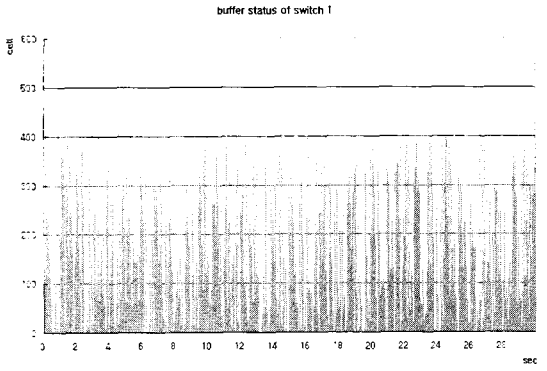

그림 7. 기존 알고리즘의 스위치 버퍼추이(delay = 5ms)
Fig. 7. Buffer status on 5ms delay of previous algorithm.

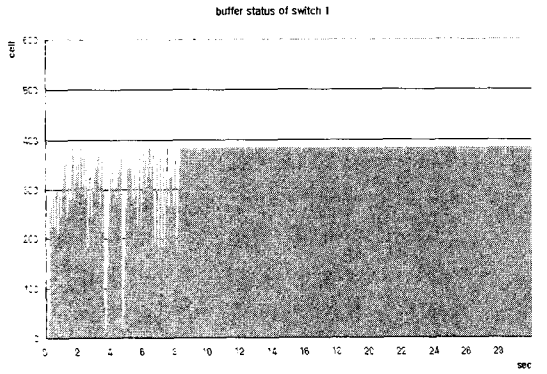

그림 8. 제안된 알고리즘의 스위치 버퍼추이(delay = 5ms)
Fig. 8. Buffer status on 5ms delay of proposed algorithm.

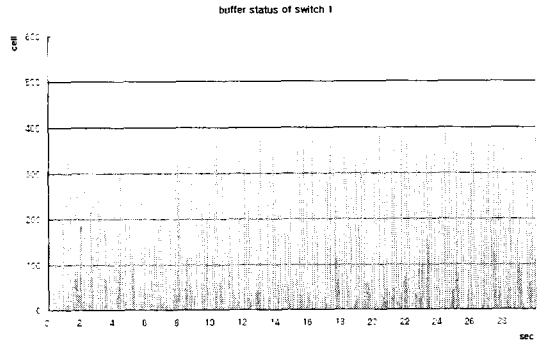

그림 9. 기존 알고리즘의 스위치 버퍼추이(delay = 10ms)
Fig. 9. Buffer status on 10ms delay of previous algorithm.

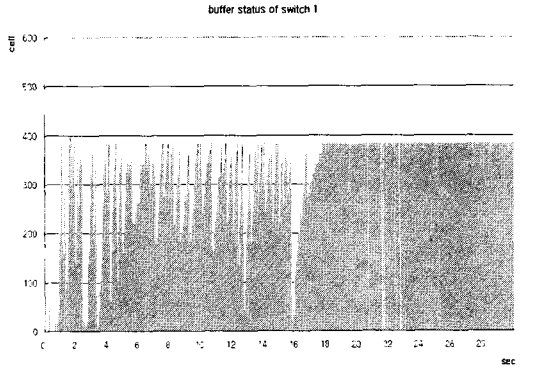

그림 10. 제안된 알고리즘의 스위치 버퍼추이(delay = 10ms)
Fig. 10. Buffer status on 10ms delay of proposed algorithm.

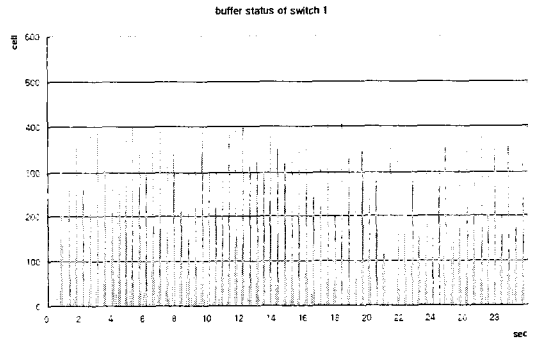

그림 11. 기존 알고리즘의 스위치 버퍼추이(delay = 20ms)
Fig. 11. Buffer status on 20ms delay of previous algorithm.

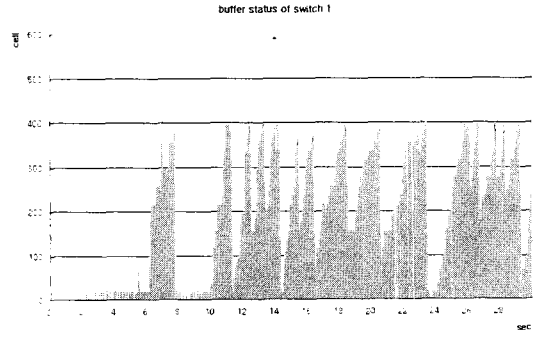

그림 12. 제안된 알고리즘의 스위치 버퍼추이(delay = 20ms)
Fig. 12. Buffer status on 20ms delay of proposed algorithm.

■ **버퍼효율**

시뮬레이션에서 사용된 ATM 스위치의 버퍼는 500 cell의 용량을 갖고 있다. <표 3>은 1 ms간격마다 측정한 버퍼의 점유 추이를 기록하여 이를 평균값을 표시한 것이다.

모든 스위치 간 지연시간에서 제안된 알고리즘은 기존의 알고리즘보다 우수한 평균 버퍼 점유량을 보여주는데 평균값의 경우 기존 알고리즘이 매 초마다 500 cell의 버퍼 용량중 169.4481의 점유량을 보이는데 반해 제안된 알고리즘은 50%를 상회하는 점유량을 보여주고 있다.

<그림 13>과 <표 4>는 단위시간당 버퍼의 점유량을 백분율의 이용효율로 나타내어 제안된 알고리즘과 기존 알고리즘의 버퍼 효율의 차이를 보여준다. 제안된 알고리즘은 평가된 모든 스위치 간 지연시간의 실험에서 기존의 알고리즘보다 평균값에서 22.56% 더 높은 버퍼 이용율을 보여준다.

표 3. 평균 버퍼 점유량
Table 3. Average buffer occupancy.

delay	기 존	제 안
5 ms	254.6249	363.2307
10 ms	182.7859	320.4337
15 ms	137.4517	254.1773
20 ms	102.9264	191.2083
평 균	169.4481	282.2625

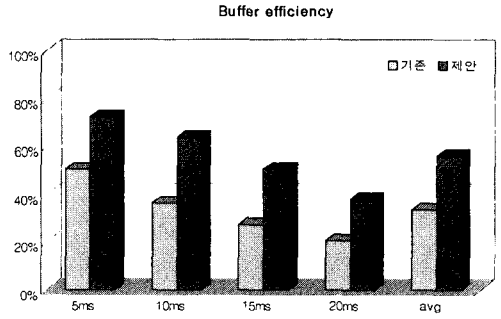

그림 13. 버퍼의 이용 효율
Fig. 13. Efficiency of buffer utilization.

표 4. 버퍼이용 효율
Table 4. Buffer efficiency.

delay	기 존	제 안	제안 - 기존
5 ms	50.92%	72.65%	21.73%
10 ms	36.56%	64.09%	27.53%
15 ms	27.49%	50.84%	22.99%
20 ms	20.59%	38.24%	17.65%
평 균	33.89%	56.45%	22.56%

■ **패킷 폐기율**

<그림 14>와 <표 5>는 기존 알고리즘과 제안된 알고리즘의 패킷 폐기율을 보여준다. 스위치 간 지연시간별로 종단의 각 VC의 링크마다 측정한 값을 종합하여 백분율로 표현하였는데 기존의 알고리즘은 모든 지연시간별 성능평가에서 대략 0.1%의 폐기율을 보인다. 패킷 폐기율이 높은 경우는 네트워크에서의 자원을 효율적으로 이용하지 못하고 이를 재전송함에 따른 비효율적인 전송양상을 보이게 되며 신뢰성 자체에도 문제를 갖게 된다. 지연시간의 상승에 따라 기존의 알고리즘도 점차 폐기율이 줄어들었는데 제안된 알고리즘은 15 ms와 20 ms의 스위치 간 지연시간의 성능평가에서는 0.1%도 안되는 매우 낮은 폐기율을 보여줌을 확인 할 수 있다.

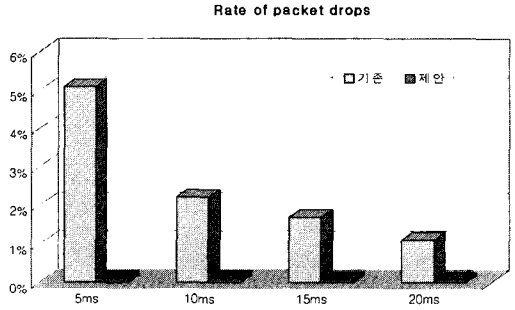

그림 14. 패킷 폐기율
Fig. 14. Packet drop rate.

표 5. 패킷 폐기율
Table 5. packet drop rate.

delay	기 존	제 안
5 ms	5.08739%	0.11568%
10 ms	2.22520%	0.10509%
15 ms	1.69941%	0.09212%
20 ms	1.12751%	0.07981%

■ **전체 TCP 수율 (TCP goodput)**

<그림 15>와 <표 6>은 제안된 알고리즘과 기존 알

고리즘의 스위치 간 지연시간별 TCP goodput을 나타낸다. 5 ms의 지연시간에서 1.06 Mbps, 10 ms의 지연시간에서는 3 Mbps, 15 ms와 20 ms의 지연시간에서는 4 Mbps에 달하는 큰 차이를 보이고 있다. 지연시간이 길어짐에 따라 네트워크 스위치의 처리 시간도 상승하게 되는데 이는 전체 네트워크의 응답 지연을 야기하게 된다.

따라서 성능 평가에서 보이는 바와 같이 지연시간이 연장됨에 따라 네트워크의 전체 수율 역시 점차적으로 감소하게 되는데 제안된 알고리즘은 지연시간이 길어져 전체 네트워크 성능이 저하되는 경우에도 효과적으로 이를 완화하여 주므로 기존 알고리즘에 비해 효율의 감소가 적은 모습을 보여준다.

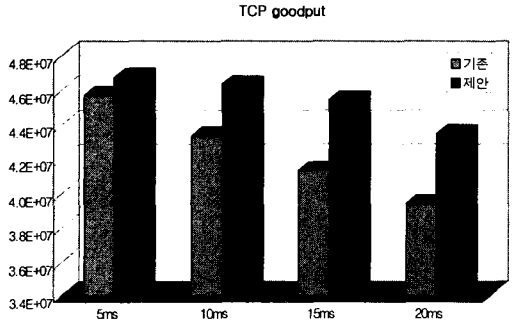

그림 15. 전체 TCP 수율
Fig. 15. Total TCP goodput.

표 6. 전체 수율
Table 6. Total TCP goodput.

delay	기 존	제 안
5 ms	4.57505E+07	4.68096E+07
10 ms	4.33674E+07	4.63727E+07
15 ms	4.13545E+07	4.54574E+07
20 ms	3.94122E+07	4.35044E+07

■ 효율 (Efficiency of TCP goodput)

<그림 16>과 <표 7>에서는 시뮬레이션을 통해 평가된 측정치를 통해 기존 알고리즘과 제안된 알고리즘의 실제 전송 효율을 비교해 보여준다.

제안된 알고리즘은 모든 스위치 간 지연시간의 성능 평가에서 기존의 알고리즘보다 실제 효율 면에서 월등한 성능을 보여주고 있다. 기존의 알고리즘이 스위치 간 지연시간이 연장됨에 따라 15 ms의 성능 평가에서 88.08306%, 지연시간 20 ms의 평가에서는 83.94610%의 열등한 전송효율을 보이는 것에 반해 제안된 알고리즘은 20 ms의 스위치 간 지연시간에서도 효율적인 네트워크 흐름제어로 92%를 상회하는 우수한 효율을 보여주며 15 ms와 20 ms의 지연시간에서 제안된 알고리즘과 기존 알고리즘의 효율 차이는 약 8.7%에 달하는 결과를 보여주고 있다.

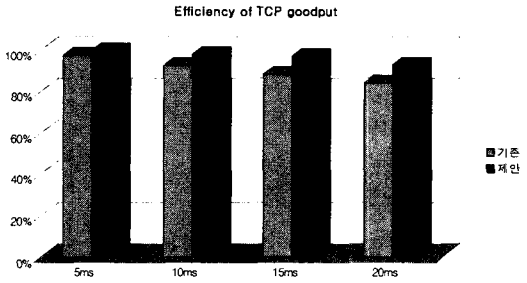

그림 16. 실제 효율
Fig. 16. Actual efficiency of TCP goodput.

표 7. 실제 효율
Table 7. Actual Effciency of each algorithm.

delay	기 존	제 안	제안-기존
5 ms	97.44639%	99.70210%	2.25571%
10 ms	92.37051%	97.77148%	5.40097%
15 ms	88.08306%	96.82208%	8.73902%
20 ms	83.94610%	92.66229%	8.71619%

V. 결 론

본 논문에서는 TCP의 폭주처리 상황에서 RTT 파라미터를 이용해 혼잡윈도우의 크기를 조절함으로서 기존의 알고리즘에서 단순히 ACK가 도착할 때마다 지수적으로 증가하였던 혼잡윈도우의 크기를 혼잡발생시 기억된 RTT에 따른 제어기법으로 혼잡의 재 발생을 방지하여 혼잡발생과 복구를 위해 소비되는 전송지연과 스위치 버퍼 이용률의 감소를 완화시키고자 하였다.

시뮬레이션 결과, 기존의 알고리즘이 계속적인 혼잡발생에 따른 제어에 의해 톱니모양의 혼잡윈도우 변화를 보여주었던 것에 비해 제안된 알고리즘은 시간이 흐름에 따라 점차 안정적인 혼잡 윈도우 크기를 유지함을 볼 수 있었으며, 스위치에서의 각 지연시간별 버퍼 추이와 이용효율에서도 평균 22.56% 향상되는 등 기존의 알

고리즘보다 우수한 성능을 보여주었다.

참 고 문 헌

[1] W. Stevens. TCP/IP Illustrated, volume 1 : the protocols. Addison Wesley, 1994.

[2] Sally Floyd, "A Report on Recent Developements in TCP Congestion control," IEEE Communication Magazine, pp 84~90, April, 2001.

[3] ITU-TS Recommendation I.121, "Broadband aspect of ISDN," 1988.

[4] W. Stevens, "TCP slow start, congestion avoidance, fast retrasmit and fast recovery algorithms," Internet RFC 2001, Jan. 1997.

[5] Uyless Black, "TCP/IP & Related Protocols", McGraw-Hill, 1994.

[6] S. Pappu and D. Basak, "TCP over GFR Implementation with Different Service Category: A Simulation Study," ATM Forum Contribution ATM97-0310, April 1997.

[7] J. Postel, "Transmission control protocol, protocol specification", Internet RFC 793, Sep. 1981.

[8] J. Nagle, "Congestion control in TCP/IP internetworks," Internet RFC 896, Jan. 1984.

[9] D. Clark, "Window and acknowledgement strategy in TCP," Internet RFC 813, July 1982.

[10] B. Braden, "Requirements for Internet hosts - communication layers," Internet RFC 1122, Oct. 1989.

[11] 김관웅, 배성환, 전병실, "ATM 망에서 인터넷 트래픽을 서비스하기 위한 효율적인 스케줄링 알고리즘에 관한 연구", 대한전자공학회 논문지 39권 TC편, 9호 pp. 382~289, 2002.

저 자 소 개

崔 址 賢(學生會員)

2001년 2월 : 전북대학교 전자공학과 졸업. 2003년 2월 : 전북대학교 전자공학과 대학원 석사 졸업. 2003년~현재 : 군 복무중. <주관심분야 : ATM, TCP/IP, Traffic management>

金 大 永(學生會員)

2002년 2월 : 전북대학교 전자공학과 졸업. 2003년 현재 : 전북대학교 전자공학과 대학원 석사과정 재학중. <주관심분야 : TCP/IP, Traffic Management, IPoverATM>

金 觀 雄(正會員) 第39卷 TC編 第6號 參照

鄭 炅 澤(正會員) 第39卷 TC編 第6號 參照

田 炳 實(正會員) 第39卷 TC編 第6號 參照