TCP/IP 망에서의 혼잡 제어 방식에 관한 연구

임 재 홍*

A Study on a Congestion Control for TCP/IP Networks

Jae-Hong Yim

<目 次>

Abstract

One of the serious problems in interconnected networks is traffic congestion which is caused by the difference of transmission speed or workload between the high speed network and the lower speed network.

This paper presents a congestion control scheme for interconnected networks and applies this scheme to the TCP/IP(Transmission Control Protocol/Internet Protocol) protocol suite. In the TCP/IP protocol, the source quench mechanism is used to control congestion, but this mechanism has many defects. For this reason, the proposed congestion control scheme is applied to the TCP/IP protocol to improve the performance.

By the simulation, this paper evaluates the performance of the scheme and the results show that the proposed scheme is more stable in the overload condition.

* 한국해양대학교 이공대학 전자통신공학과 전임강사

Ⅰ. 서 론

최근 통신환경에는 놀랄만한 변화들이 급속하게 일어나고 있다. 먼저 통신속도 측면에서 광전송 기술의 발달과 FDDI(Fiber Distributed Data Interface), DQDB(Distributed Queue Dual Bus) 그리고 B - ISDN(Broadband - Integrated Services Digital Network)과 같은 고속망의 등장으로 데이타 전송속도가 수 Gbps까지 증가하게 되었고, 망에서 제공하는 서비스 품질도 월등히 향상되고 있다. 또한 하드웨어 기술의 발달로 CPU의 처리 속도가 급속하게 증가되고 있고, 가격 또한 낮아지고 있는 추세이다[12, 17].

망의 속도와 품질이 향상되면서 새로운 문제들이 나타나게 되었는데, 첫째로 기존에는 망의 전송 속도가 전체적인 성능 저하의 주 요인이었으나 이제는 각 컴퓨터 시스템에서 수행되는 통신 프로토콜이 전체 성능에 커다란 영향을 미치게 되었다. 따라서 저속의 낮은 품질을 제공하던 망을 지원하기 위해 복잡하게 설계되었던 TCP(Transmission Control Protocol)나 TP4(Transport Protocol class 4) 대신에 고속망에 적합한 XTP(Xpress Transfer Protocol)나 HSTP(High - Speed Transport Protocol)와 같은 고속의 트랜스포트 프로토콜이 등장하게 되었다[8, 9, 10, 11, 12]. 두번째로 망의 전송 속도 증가로 인한 혼잡(congestion)이 커다란 문제로 나타나게 되었다. 기존에는 혼잡 문제를 해결하기 위하여 sliding window 메카니즘을 이용한 흐름 제어(flow control) 방식을 사용하였지만, 흐름 제어만으로는 혼잡 문제를 효과적으로 해결하기가 어려워 속도 제어(rate control)라는 개념이 나오게 되었다[7, 8, 9].

현재 세계 최대의 전산망인 인터넷의 표준 프로토콜로 사용되고 있는 TCP/IP는 망 부하의 증가와 망의 속도 차이로 인해 발생하는 혼잡 문제가 커다란 문제점으로 나타나고 있다. 이를 해결하기 위해 현재는 source quench 메카니즘을 사용하고 있지만 여러 문제점이 존재하고 있다[1, 2, 3, 4, 5, 15, 16].

본 논문에서는 혼잡 문제를 해결하기 위한 새로운 혼잡 제어 메카니즘을 제안하고 이를 IP 레벨에 적용시켜 성능 향상을 도모한다. 또한 시뮬레이션을 통하여 제안된 메카니즘의 성능을 평가한다.

Ⅱ. 혼잡 제어(Congestion Control)

1. 속도 제어(Rate Control)

망 부하의 증가로 인해 서비스 사용자가 망 서비스 사용을 제한받아서 서비스의 효율이 떨어질 때 그 망은 혼잡 상태에 있다고 정의한다. 망이 혼잡 상태에 있게 되면 데이타 패킷의 큐잉 지연(queueing delay) 시간이 증가하게 되고, 패킷이 손실될 수 있으며, 패킷이 손실되었을 경우에는 재전송으로 인해 실제 유효한 데이타의 처리량이 감소하게 된다.

혼잡 상태를 제어하기 위해서 기존에는 sliding window 메카니즘을 이용한 흐름 제어(flow control) 방식을 많이 사용하였지만, 흐름 제어만으로는 혼잡 문제를 효과적으로 해결하기가 어려워 속도 제어(rate control)라는 개념이 나오게 되었다[7, 8, 9].

속도 제어는 두 가지 방법으로 구현될 수가 있는데, 그림 1의 (a)에서 보는 바와 같이 데이타의 burst와

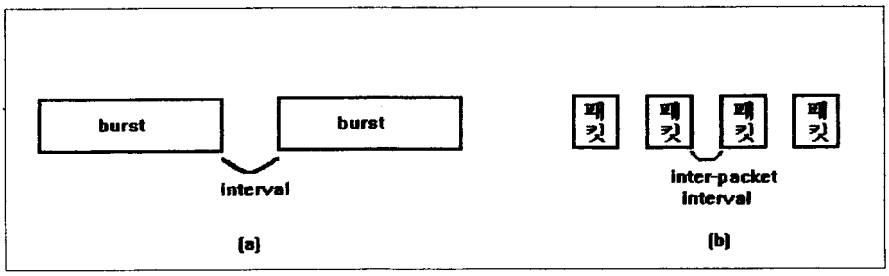

그림 1. 속도 제어 방식

burst사이 간격을 조절하는 방법과, (b)에서와 같이 각각의 패킷 사이의 간격을 조절해서 보내는 방법이 있다[14].

1) XTP(Xpress Transfer Protocol)의 속도 제어 방식

XTP는 VLSI 칩 형태인 하드웨어적으로 구현하기 위해 개발된 고성능 프로토콜이다. XTP는 실시간 데이타그램 전송, 멀티캐스팅 그리고 효율적인 벌크 데이타 전송 등의 서비스를 제공하는데, 기존의 트랜스포트 계층(transport layer)과 망 계층(network layer)을 하나의 전달 계층(transfer layer)으로 통합하였다.

XTP에서는 속도 제어를 위하여 위에서 기술한 두 가지 방식 중 첫 번째 방법을 이용한다. 즉, 수신측에서 RATE와 BURST 값을 송신측으로 넘겨주면 송신측은 그 값을 이용해서 한 BURST당 할당되는 RTIMER 값을 구하게 된다. 그리고 나서 RTIMER 동안 한 BURST만큼의 데이타를 보내고 RTIMER 시간중 여분의 시간은 기다리다가 다시 다음 RTIMER 시간에 다음 BURST를 보내게 된다. 여기에서 RATE는 수신측에서 매초당 받아들일 수 있는 최대 byte 수가되고, BURST는 수신측에서 하나의 단위로 여겨지는 최대 byte 수로써, 수신측이 한번에 연달아 계속 받아들일 수 있는 패킷의 최대량이라 보면 된다. 사용되는 RATE 값과 BURST 값은 수신측에서뿐만 아니라 전송 경로상에 존재하는 각각의 노드에서도 그 값을 변경시킬 수 있다. 즉, 제어 패킷의 RATE 영역과 BURST 영역에 자신이 원하는 값을 설정할 수 있다. 결국 송신측에서는 경로상의 노드중에 속도가 가장 느린 시스템의 속도에 맞추어 데이타를 보내게 된다[7].

2) VMTP(Versatile Message Transaction Protocol)의 속도 제어 방식

VMTP는 분산 운영 체제에서 높은 성능의 통신 서비스를 제공하기 위해서 개발된 트랜스포트 계층의 프로토콜로서 RPC(Remote Procedure Call)와 같은 일반적인 처리 - 지향(transaction - oriented) 통신을 지원하며, IP 데이타그램 상에서 동작한다. 또한 많은 양의 데이타 전송을 위해서도 효율적 전송을 위한 속도 조절과 선택적 재전송을 통한 스트림 모드의 전송을 지원하고 있다.

VMTP에서 사용되는 속도 제어 방식은 위에서 기술한 방식 중 두 번째 방법을 이용하고 있다. 즉, 패킷간의 시간 간격을 조절하므로써 속도를 조절한다. 송신측과 수신측은 패킷간의 적절한 시간 간격인 inter - packet time을 협상하여 정한 후 데이타를 전송한다. 그리고, 속도 조절은 패킷 손실로 인해 수신측으로부터 요구되어 오는 선택적 재전송 요구나 송신측의 타이머에 기반을 두고 행해진다. 예를 들면, 수

신측에서 계속 재전송을 요구해 오거나 보낸 데이타에 대해 일정한 시간이 지났는데도 ACK(acknowledgment)가 오지 않을 때, 즉 송신측의 타이머가 timeout 되었을 때는 중간에서 패킷이 손실된 것을 암시하므로, 송신측에서는 inter - packet time을 늘리도록 하는 방식이다. 이와 같은 방법은 만약 패킷의 길이가 일정하게 정해져 있다면, XTP에서 사용된 속도 제어 방식에서 BURST의 크기가 한 패킷의 크기로 정해진 경우와 같다[8].

2. 기존 속도 제어 방식의 문제점

1) XTP의 문제점

XTP에서는 하나의 BURST 양을 짧은 시간에 한꺼번에 보내게 되므로 망의 중간 노드에서 혼잡이 발생할 확률이 높다는 문제점이 있다. 또한 실제로 혼잡이 발생했을 때 이것을 송신측에 알려주는 메카니즘이 없고, 혼잡을 줄이기 위해 속도를 재조정하는 방법에 대한 명시가 없다. 그리고 송신측의 RATE는 경로상에 존재하는 모든 노드중에서 처리 능력이 가장 느린 시스템의 속도로 정해지는데, 이 값이 송신측에 알려지기 까지의 지연 시간이 문제가 된다.

2) VMTP의 문제점

VMTP에서는 실제로 혼잡이 발생했을 때 이것을 송신측에 알려주는데 걸리는 피드백 지연 시간과 송신측의 timeout이 문제가 된다. 즉, 속도 조절이 패킷 손실로 인해 수신측으로부터 요구되어지는 선택적 재전송 요구나 송신측의 타이머에 기반을 두고 있으므로, 수신측으로부터 혼잡이 발생했다는 것을 암시하는 재전송 요구 패킷이 송신측에 도착할 때까지, 혹은 송신측의 타이머가 timeout될 때까지는 송신측이 데이타를 계속 빠른 속도로 보내게 되므로 혼잡 현상을 더욱 악화시키는 결과를 초래할 뿐만 아니라, 이런 데이타들은 손실될 가능성도 높다. 따라서 이러한 피드백 지연 시간의 문제점을 줄이기 위해 각각의 스위칭 노드 사이에서 혼잡 제어를 할 필요가 있다. 또한, VMTP에서의 혼잡 제어는 중간 스위칭 노드에서 혼잡 문제가 직접 제어되지 않고, 미리 예방되지도 않는다. 그리고 혼잡으로 인해 전송 속도를 줄인 후, 혼잡 현상이 없어졌을 때 전송 속도를 다시 복원시키는 메카니즘이 없다.

Ⅲ. 제안하는 혼잡 제어 방식

제안하는 혼잡 제어 방식은 제어 메카니즘이 데이타의 경로를 따라 각각의 스위칭 노드에서 수행된다. 적용 대상이 되는 프로토콜의 종류에 따라 제어 메카니즘이 각각의 노드 사이에서 수행될 수도 있고 혼잡이 발생한 노드와 송신측 사이에서 수행될 수도 있다. 각 노드와 노드 사이에서 트래픽 제어를 하게 되면 양단 시스템간에서 제어를 해줄 때보다 제어 패킷의 피드백 지연 시간이 훨씬 줄어들게 되므로, 보다 신속하고 정확한 트래픽 제어를 할 수 있다[12, 13]. 하지만 IP 경우에서와 같이 각 노드와 노드 사이에서 제어 메카니즘을 수행하기에 부족한 점이 많은 경우에는 제어를 노드와 송신측 사이에서 수행한다. 이 제어 메카니즘이 수행되기 위해선 각 스위칭 노드에서 데이타 트래픽을 모니터링하고 이에 대한 속도를 제어

해야 한다.

본 논문에서 제안하는 제어 메카니즘에는 다음과 같은 패러미터들이 이용된다.

B=total buffer size

B_l=level of packets queued

R_i=input rate

R_o=output rate

R_n=new transmission rate

P_r=processing rate of the node

T=required time for control packet to arrive at the previous node or the sender

α=sensitivity of rate adjustment($0<\alpha\leq 1$)

β=buffer reservation factor($0<\beta\leq 1$)

δ=rate adaptation factor($0<\delta\leq 1$)

속도 제어의 기본 방식은 데이타를 다음 노드의 속도에 맞추어서 보내는 것이다. 데이타의 전송 속도는 수신 노드가 초당 받아들일 수 있는 패킷의 양인 P_r보다 절대로 커지면 안된다. 전송 속도가 P_r보다 커지게 되면 수신 버퍼가 남아있어도 데이타는 버퍼링되지 못하고 손실되어 버린다.

본 논문에서 제안하는 방식에서는 혼잡을 버퍼의 상태로 파악한다. 즉, 버퍼에 큐잉되어 있는 데이타의 양이 일정 수준을 넘으면 혼잡 상태로 간주한다. 그림 2에 각 노드의 버퍼 상태가 나타나 있다.

여기서, R_i는 항상 Pr보다 작아야 한다. 한편, R_i가 R_o보다 커지게 되면 B_l이 계속 증가하게 되어 결국에는 버퍼 오버플로우가 발생하게 되고, R_i가 R_o보다 작게 되면 B_l이 점차로 감소하여 결국 버퍼는 빈 상태가 된다. 이런 현상을 막기 위한 제어 메카니즘이 다음 절에 제시된다.

1. 혼잡 상태 제어

먼저 버퍼가 혼잡 상태에 이르는 경우로 $R_i>R_o$일 때, 즉 버퍼로 들어오는 데이타의 속도가 버퍼를 빠져나가는 속도보다 빠른 경우에 $T\cdot(R_i-R_o)\geq\beta\cdot(B-B_l)$이면 버퍼 오버플로우가 발생하게 된다. 여기서 T는 제어 패킷이 그 전 노드나 송신측까지 가는데 걸리는 시간이므로 $T\cdot(R_i-R_o)$는 혼잡 상태를 알리는 제어 패킷이 도착지까지 가는 동안 계속해서 버퍼에 쌓이는 데이타의 양이 된다. 따라서, 제어 패킷이 도착지까지 가는 동안 계속해서 버퍼에 쌓이는 데이타의 양이 남아있는 버퍼의 양보다 커지게 되면, 즉 $R_i>R_o$

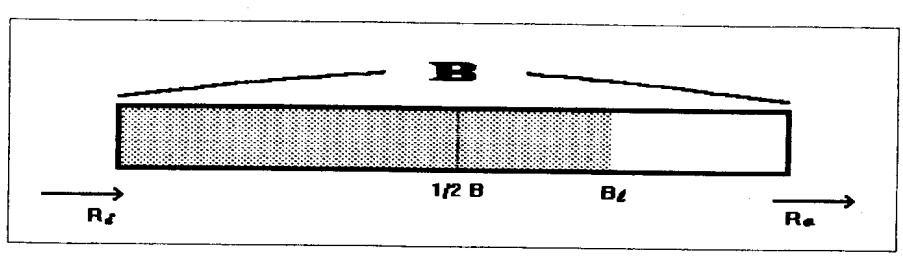

그림 2. 각 노드의 버퍼 상태

일 때 $T \cdot (R_i - R_o) \geq \beta \cdot (B - B_l)$이면 버퍼 오버플로우가 발생하므로 스위칭 노드는 이렇게 되기 전에 미리 제어 패킷을 보내 속도를 줄이도록 한다. 여기서 β값은 제어 패킷이 도달할 때까지 버퍼에 데이타가 계속 쌓이고 나서도 남아있는 여유 분의 버퍼 양을 조절하는 패러미터이다. β값이 1에 가까울수록 남는 버퍼의 양은 줄어들게 되므로 버퍼 이용 효율은 높아지게 되지만 그만큼 오버플로우의 위험성은 증가하게 된다.

또 한 가지 고려해야 할 사항은 송신측의 속도를 얼마 만큼 줄일 것인가 하는 문제이다. 새로이 요구되는 송신 속도 R_n이 단순히 R_o와 같아지게 되면 제어 패킷이 앞 노드에 도달할 때까지 쌓이는 데이타의 양으로 인해 증가한 B_l이 계속 증가한 상태로 유지되므로, R_n은 R_o보다 작아야지만 증가했던 버퍼 큐잉 레벨이 감소하여 다시 정상 상태를 유지할 수 있게 된다. 이를 위해 패러미터 α가 이용되는데, 이 α값에 따라 송신측의 속도가 결정된다.

2. 저속 상태 제어

위의 경우와는 반대로 송신측에서 보내는 데이타의 속도가 너무 느리거나 혼잡 상태가 해제되어 송신측의 전송 속도를 증가해도 무방한 경우이다. 즉, 전송 속도 R_i가 R_o보다 작을 때에는 송신측의 데이타 전송 속도를 증가할 필요가 있는데, 본 논문에서 제안하는 방식은 무조건 속도를 높여주는 것이 아니라 버퍼가 일정 수준 이상 차있으면 속도 증가를 일단 유보하고 버퍼가 일정 수준 이하로 비게 될 때 속도를 증가시킨다. 여기서 일정 수준의 버퍼 양은 패러미터 δ값을 조정하므로써 정할 수 있다. 속도 증가 메카니즘은 속도 감소의 경우와 비슷하다. R_i가 R_o보다 작을 때 제어 패킷이 도착지까지 가는 동안 계속해서 줄어들고 난 후의 버퍼 큐잉 레벨이 전체 버퍼 크기의 δ배 보다 작게 되면, 즉 $R_o > R_i$일 때 $(B_l - T \cdot (Ro - Ri)) \leq \delta \cdot B$일 때 송신 속도를 증가시킨다. 여기서 주의할 점은 증가될 전송 속도 R_n은 패킷 처리 속도인 P_r보다 커서는 안된다는 점이다.

이와 같은 방식으로 혼잡 상태로 인해 R_o보다 작아졌던 R_i가 일정 기간이 지나고 나면 다시 R_o와 같아지게 되고, 버퍼의 큐잉 레벨은 항상 전체 버퍼 크기의 일정 수준 이상이 되도록 하여 버퍼 이용 효율을 높이고자 하였다.

그림 3에 혼잡 상태와 저속 상태를 방지하기 위한 전체적인 속도 제어 알고리즘을 나타내었다.

Ⅳ. IP 레벨에의 적용

1. TCP/IP 프로토콜의 개요

현재 TCP/IP는 ARPANET(Advanced Research Project Agency Network)을 비롯한 패킷 교환망 뿐만 아니라 LAN(Local Area Network) 및 무선 데이타망 등에서 가장 널리 사용되고 있으며, 세계 최대의 전산망인 Internet의 표준 프로토콜로 사용되고 있다. TCP/IP는 OSI 7 계층 중 망 계층에 해당하는 IP와 전송 계층에 해당하는 TCP및 UDP(User Datagram Protocol)로 구성되어 있는데, 그 전체적인 구조가 그림 4에 나타나 있다.

```
Loop :   if ( Ri  >  Pr )
         { Rn = Pr ;
          Goto Loop ;
         }
         else if ( Ri ≥ Ro and T · (Ri - Ro) ≥ β · (B - Bi) )
         { Rn = α · Ro ;
          Goto Loop ;
         }
         else if ( Ri < Ro and  ( Bi - T · (Ro - Ri) ) ≤δ · B )
         {
           if ( Ro > Pr )
           {  Rn = Pr ;
             Goto Loop ;
           }
           else
{  Rn = Ro ;
             Goto Loop ;
           }
         }
```

그림 3. 속도 제어 알고리즘

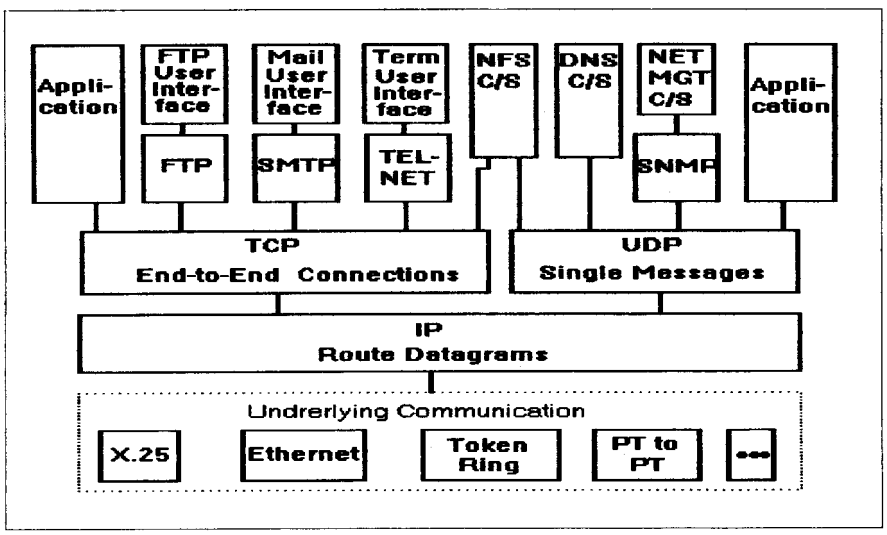

그림 4. TCP/IP 프로토콜의 구조

TCP/IP가 사용될 수 있는 망의 형태는 단일 LAN이나 WAN(Wide Area Network) 혹은 IP 라우터를 통한 서로 다른 망 간에서 사용될 수 있는데 그 한 예를 그림 5에 나타내었다. 그림에서와 같이 지역적으로 떨어져 있는 서로 다른 망들이 라우터나 게이트웨이를 통해 서로 연결될 수 있다.

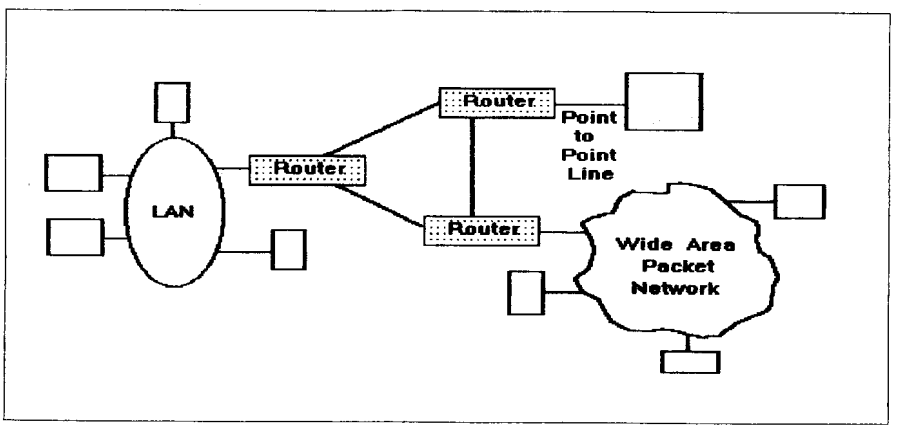

그림 5. TCP/IP를 사용하는 망 형태의 예

2. ICMP(Internet Control Message Protocol)

OSI 참조 모델에서 망 계층에 해당하는 기능을 하는 IP는 신뢰성이 없는 비접속형 데이타그램 전송 서비스를 제공하므로 문제가 생겼을 때 이를 알려주어야 할 필요성이 있다. 이러한 목적에서 사용되는 프로토콜이 ICMP(Internet Control Message Protocol)이며 IP의 한 부분에 해당된다[1].

호스트를 포함한 각각의 노드에서는 링크가 다운되어 데이타그램이 목적지에 도달할 수 없는 경우나 원하는 목적지를 찾지 못하는 경우, 혹은 노드가 혼잡 상태에 빠진 경우 등 문제가 발생했을 때 데이타그램을 버려야하는데, 이 때 이 노드에서는 문제 발생을 알려주는 ICMP 메시지를 원래의 송신측에 보내주게 되고, ICMP 메시지를 받은 측에서는 그에 따른 적절한 조치를 취하게 된다. ICMP 메시지는 그림 6에서 보는 바와 같이 IP 데이타그램을 통해 전달된다.

ICMP 메시지는 크게 두 가지 종류로 나뉘는데 첫 번째는 문제가 발생했을 때 이것을 알려주는데 사용되는 에러 메시지이고, 두 번째는 여러 유용한 정보를 얻고자 할 때 쓰이는 정보 메시지이다. 에러 메시지

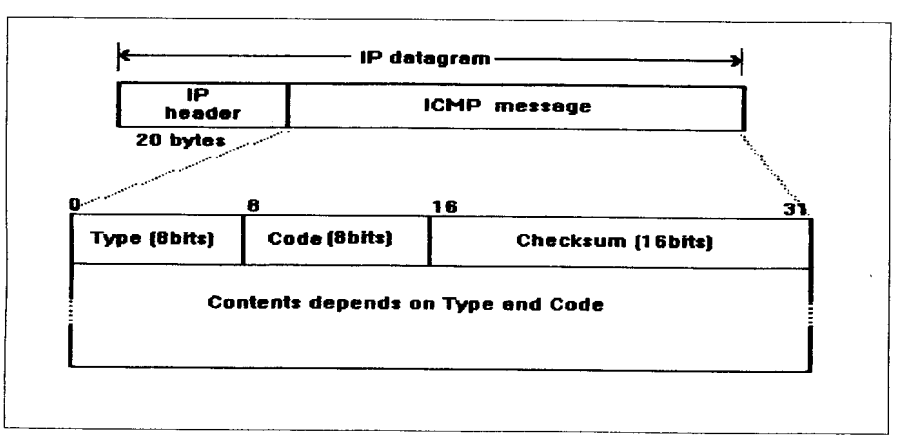

그림 6. ICMP 메시지 구성

에는 데이타그램이 목적지에 도달할 수 없을 때 사용되는 것과 데이타그램의 타이머 필드가 timeout 되었을 때 사용되는 것, 패러미터에 문제가 있을 때 사용되는 것, 혼잡이 발생했을 때 사용되는 것, 그리고 라우트를 변경할 필요가 있을 때 사용되는 것들이 있다. 한편, 정보 메시지는 호스트간이나 호스트와 라우터 사이에서 시스템이 지금 살아 있는지, 데이타그램을 처리하는데 시간이 얼마나 걸리는지, 또 어떠한 어드레스 마스크가 사용되는지 등에 대한 정보를 교환하는데 사용된다. 여기서 주목해야 할 항목은 source quench 메시지 항목이다.

1) Source Quench Message

TCP/IP에서는 수신측의 버퍼가 오버플로우 되지 않도록 하기 위해서 양단간에 흐름 제어를 하고 있는데, 각 노드 사이에서는 특별히 명시된 흐름 제어가 행해지지 않고 있기 때문에 IP 모듈에서는 중간 노드에서의 혼잡 문제를 완화하기 위해서 source quench 메카니즘을 사용한다.

보통 속도가 다른 망을 서로 연결할 때에는 중간 노드에서 혼잡 상태가 발생하기 쉬운데, 예를 들면 10 Mbps의 두 LAN이 56Kbps의 속도를 가지는 WAN을 통해 연결되는 경우에 LAN과 WAN을 연결해 주는 게이트웨이에서는 두 망의 속도 차이로 인해 혼잡 상태가 발생하기 쉽다. 즉, 이러한 혼잡 상태로 인해 다음 망으로 데이타그램을 내보내기 위한 송신 버퍼가 오버플로우 되면 이때 게이트웨이에서는 데이타를 송신 버퍼로 버퍼링하지 못하므로 데이타그램을 그냥 버리면서 source quench 메시지를 원래의 송신측에 보내게 된다[2, 3].

즉, source quench 메시지는 혼잡 상태를 완화하기 위해 송신측에 데이타의 송신 속도를 줄여달라고 요청하는 역할을 한다.

Source quench 메시지의 구조가 그림 7에 나타나 있다. 여기에서 Internet Header와 64bits의 데이타그램의 데이타는 이 source quench 메시지를 받은 호스트에서 문제를 일으킨 데이타그램을 어떠한 전송 계층의 프로세스가 사용했는가를 알아내는데 사용된다. 보통 포트 넘버가 데이타그램의 처음 부분에 들어있게 되므로 이것을 이용하여 상위 전송 계층의 프로세스를 찾게 된다.

그림 8에서 보듯이 게이트웨이의 각 링크에는 송신 버퍼와 수신 버퍼가 존재한다. 기존의 IP 모듈에서는 다른 쪽 망으로 나가는 송신 버퍼가 꽉 차있어서 더이상 데이타그램을 버퍼로 큐잉시키지 못하게 될 때, 이 데이타그램을 버리면서 이 데이타그램의 원래 송신측에 source quench 메시지를 보내서 혼잡 문제

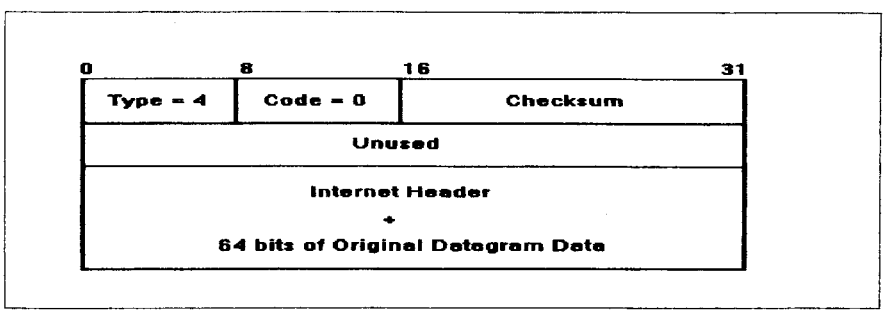

그림 7. Source Quench 메시지의 구조

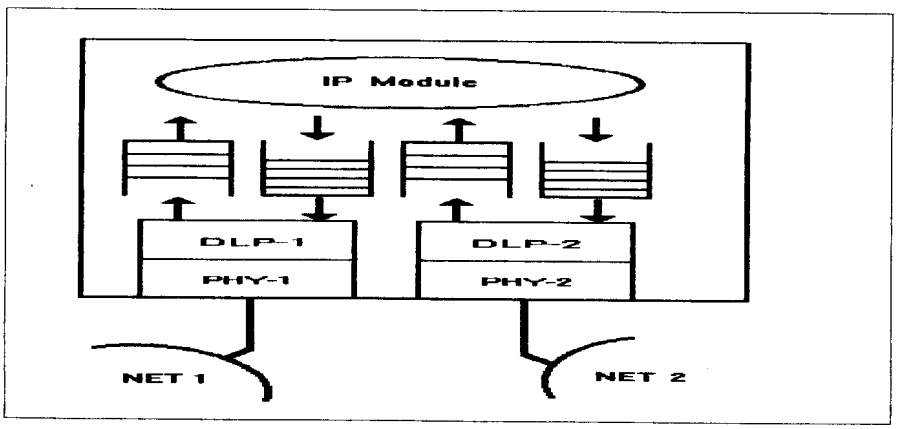

그림 8. 게이트웨이의 버퍼 상태

를 해결하려고 하였다.

2) Source Quench 메카니즘의 문제점

Source quench 메카니즘에는 발생한 문제에 대한 명확한 명시가 없기 때문에 source quench 메시지를 받은 호스트 입장에선 적절한 대응을 하기가 어렵게 된다. 즉, 이 source quench 메시지가 극심한 혼잡 상태를 나타내는 것인지 아니면 혼잡 상태로 이르고 있다는 것을 나타내는 것인지, 혹은 버스트 패킷들이 혼잡 상태를 야기하는 지와 같은 정확한 명시가 없기 때문에 호스트 입장에선 그에 따른 적절한 대응을 하기가 어렵다.

또한, source quench 메시지는 불공평하게 전달될 수 있다. 예를 들면, 어느 특정한 커넥션이나 호스트에 속하는 데이타그램들로 인해 버퍼가 거의 차게 되었을 때 부하량이 적은 다른 쪽 커넥션이나 호스트에 속하는 데이타그램이 오버플로우를 발생시키게 되면, 부하량이 많은 쪽이 아니라 적은 쪽으로 source quench 메시지가 전달될 우려가 있다.

한편, 이 source quench 메시지가 원래의 송신측에 도달할 때까지의 지연 시간이 문제가 된다. 이 메시지가 송신측에 도달할 때까지는 송신측이 계속해서 빠른 속도로 데이타를 보내게 되므로 중간 노드에서는 계속해서 데이타가 버려지고 source quench 메시지도 계속 보내지게 된다. 이렇게 되면 이 제어 메시지로 인해 역방향의 자원이 낭비되고 중간 노드의 CPU 사용 시간이 낭비되는 결과를 초래하기도 한다.

3. Source Quench 메카니즘의 수정

1) 혼잡 제어 방식

본 논문에서는 기존의 source quench 메카니즘의 문제점을 개선하기 위하여 3장에서 제시하였던 혼잡 제어 방식을 사용하여 버퍼 오버플로우를 방지한다. 3장에서 제시된 혼잡 제어 메카니즘이 각각의 노드 사이에서 제어가 실행되기 위해서는 각각의 스위칭 노드에 전송 계층까지 구현되어 있어야 한다. 그래야만 각 노드에서 현재 사용 중인 커넥션에 대한 정보를 알 수 있고 이에 따라 각 커넥션의 속도를 개별적으

로 조절할 수 있게 되는 것이다.

그러나 TCP/IP 망에서는 각 스위칭 노드에 망 계층인 IP 계층까지만 구현되어 있기 때문에 각 노드 사이에서 제어가 수행되기에는 미흡한 점이 많다. 왜냐하면, IP 계층에서는 TCP 계층의 여러 커넥션을 구별하지 않고 일률적인 비접속형의 데이타그램 서비스를 제공하기 때문에, 중간 노드에서 속도를 줄이게 된다면 원하지 않는 데이타의 속도까지 같이 줄어드는 결과를 초래하므로 각 노드 사이에서 제어를 수행하는 것이 불가능하다. 하지만 각 스위칭 노드와 소스 호스트간의 제어를 하는 경우에는 문제가 없으므로 3장의 메카니즘이 노드와 소스 호스트간에 수행되는 방식으로 TCP/IP 망에 적용될 수 있다. 즉, IP 모듈이 버퍼의 상태를 모니터링 하고있다가 3장에서 제안된 메카니즘에 의해 버퍼가 오버플로우 될 위기에 처하면 source quench 메시지를 소스 호스트로 보내게 된다. 또한 본 논문에서는 UDP가 문제를 일으킨 경우는 무시하기로 한다. UDP인 경우는 비교적 적은 양의 데이타를 주고 받기 위해 사용되는 프로토콜이므로 혼잡 제어 메카니즘에서는 무시해도 별 무리가 따르지 않는다. 실제로 BSD계열의 시스템에서 UDP를 이용하는 프로세스가 문제를 일으켜 source quench 메시지를 받은 경우에는 이를 무시하고 있다. 3장의 메카니즘이 IP 모듈, 즉 ICMP의 source quench 메카니즘에 적용되려면 게이트웨이에서는 송신 버퍼에 데이타가 들어오는 속도인 R_i와 데이타가 빠져나가는 속도인 R_o를 모니터링 해야 하고 각 호스트까지 데이타가 전달되는데 걸리는 시간인 T를 미리 알고 있어야 한다.

한편, TCP/IP 망에서는 각기 다른 대역폭을 제공하는 여러 망들이 서로 연결되어 있으므로 전체적인 망의 성능은 가장 느린 망에 따라 좌우된다. 따라서, 각 호스트는 여러 라우터로부터 source quench 메시지를 받을 수 있으므로 이때에는 가장 느린 망의 속도에 따라 데이타를 전송해 주어야 한다.

2) 속도 제어 방식

혼잡 문제를 해결하기 위하여 각 스위칭 노드는 호스트에 새롭게 요구되는 데이타 전달 속도를 알려 주어야 하는데, 여기서 약간의 고려해야 할 사항이 생기게 된다. 즉, 버퍼 내에 쌓이는 데이타그램들은 한 호스트로부터 온 것이 아니라 여러 호스트로부터 온 것이기 때문에 버퍼 내에 쌓여 있는 데이타의 양에 비례하게 속도를 줄일 필요가 있다. 예를 들면 지금 게이트웨이가 받고 있는 데이타그램이 게이트웨이가 직접 연결된 망에 속하는 호스트로부터 온 것일 수도 있고, 아니면 다른 라우터에 의해서 연결되어 있는 다른 망에 속해 있는 호스트로부터 온 것일 수도 있기 때문에, 데이타의 양에 따라 데이타를 많이 보내고 있는 쪽의 속도를 데이타를 적게 보내는 쪽보다 그만큼 더 줄일 필요가 있는 것이다. 이렇게 하기 위해서는 IP 모듈이 어디서부터 온 데이타가 버퍼 내에 얼마만큼 쌓여 있는가를 알고 있어야 한다.

이를 위해 본 논문에서는 데이타의 수신 상태를 체크하는 테이블을 이용한다. 즉, IP 모듈에서 데이타그램을 송신 버퍼에 보낼 때 이것이 TCP용인지 UDP용인지를 확인하여 UDP용인 경우에는 혼잡 제어에서 무시를 하므로 그냥 송신 버퍼로 내보내고 TCP인 경우에는 수신 테이블에 그 데이타그램의 송신측 어드레스가 존재하면 그 어드레스에 해당하는 데이타의 바이트 카운트를 데이타그램의 양만큼 증가시키고, 존재하지 않으면 테이블에 어드레스를 기록한 후에 바이트 카운트를 증가시킨 다음, 데이타그램이 버퍼를 빠져나갈 때마다 카운트를 감소시킨다. 수신 테이블에는 바이트 카운트 이외에도 각 송신측으로부터

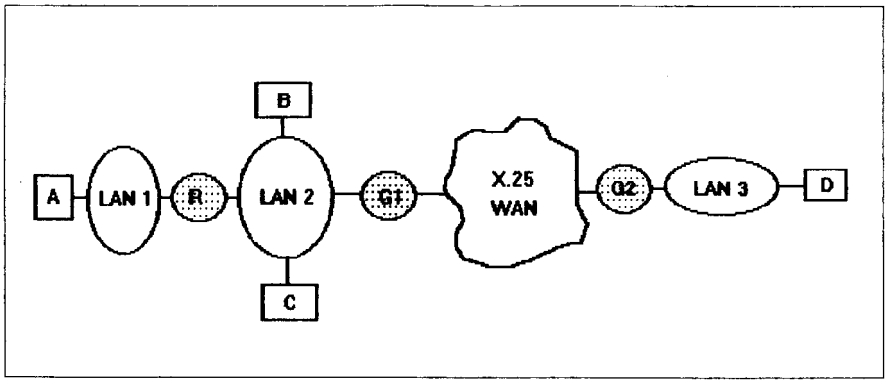

그림 9. 커넥션간의 속도 제어 문제

온 소스의 포트 넘버가 기록된다. 따라서, IP 모듈은 이 테이블을 참조하므로써 지금 각 소스로부터 온 데이타가 송신 버퍼 내에 얼마만큼 쌓여 있는지를 알 수 있게 되고, 이에 따라 데이타의 속도를 줄일 필요가 있을 때에는 현재 버퍼를 차지하고 있는 전체 데이타의 양에 대한 각 소스로부터 온 데이타 양의 비에 따라 각 소스의 속도를 줄이게 된다.

3) 호스트에서의 각 TCP 접속에 관한 문제

본 논문에서 제안하는 혼잡 제어 방식이 IP 레벨에 적용되기 위해서는 송신측과의 인터페이스에 관한 고려가 필요하다. 즉, 혼잡 제어 방식에 의해 속도를 줄이는 경우, 송신측에서는 혼잡 문제를 일으킨 프로세스가 사용하는 커넥션의 속도만을 줄여야 하고, 다른 커넥션의 속도에는 영향을 미치면 안되는 것이다.

그림 9와 같은 망의 형태를 예로 들어보자. 현재 호스트 A가 3개의 커넥션을 통해 각각 호스트 B, C, D와 통신을 하고 있다고 가정하자. 여기서 게이트웨이 G1에서 혼잡 상태가 발생하여 본 논문에서 제안된 혼잡 제어 방식에 의해 호스트 A가 게이트웨이 G1으로부터 속도를 줄여달라는 정보를 받았다고 가정하자. 이 경우에 호스트 A에서는 호스트 D와 연결된 커넥션의 속도만을 줄여야 하고 나머지 커넥션의 속도는 영향을 받으면 안된다.

이를 위해서는 G1으로부터 송신측으로 전달되는 제어 패킷 내에 어떠한 커넥션의 속도를 줄여야 하는가에 대한 정보가 있어야 한다. 그러나, 각 스위칭 노드에서는 망 계층인 IP 레벨까지 구현되어 있으므로 IP 위의 계층인 TCP에서 사용하는 커넥션에 대한 정보를 모른다는 데에 문제가 생긴다. 이 문제를 해결하기 위해 본 논문에서는 다음과 같은 방식을 제시한다.

기존의 source quench 메카니즘에서는 송신측에서 문제를 일으킨 프로세스, 즉, 문제를 일으킨 프로세스가 TCP를 사용하는지 혹은 UDP를 사용하는지, 또한 어떠한 포트를 사용하는지를 찾기 위해 ICMP 메시지 안에 존재하는 문제가 된 데이타그램의 헤더와 포트 넘버 부분을 이용하였다.

본 논문에서 제시하는 메카니즘은 UDP인 경우는 무시하므로 송신측에 포트 넘버만 알려주면 송신측에서 그에 해당하는 TCP커넥션의 속도만을 줄이면 된다. 그런데, 혼잡이 발생한 노드를 거치는 커넥션이 동시에 여러 개일 수 있으므로 혼잡이 발생한 노드에서는 이 복수의 커넥션을 가리키는 각 포트 넘버를 송

신측에 알려줄 필요가 있다. 이 문제를 해결하기 위해 각 스위칭 노드에서는 데이타그램을 송신 버퍼로 내보내면서 송신측 어드레스의 바이트 카운트를 증가시키는 동시에 데이타그램 내에 존재하는 송신측의 포트 어드레스 부분을 수신 테이블에 기록하여 둔다. 한편, 수신 테이블의 포트 넘버 영역이 다 차게 되면 다시 처음 부분부터 기록하게 하므로써 수신 테이블의 포트 넘버 영역이 계속 갱신될 수 있도록 해준다.

4) 속도 증가 문제

3장에서 제안한 혼잡 제어 방식을 TCP/IP 망에 적용시키는 데에 있어서 마지막으로 고려해야 할 문제는 혼잡 상태가 해제되거나 송신측의 전송 속도가 너무 느린 경우에 데이타의 전송 속도를 증가시키는 경우이다.

TCP/IP 망인 경우 제어가 각 라우터와 소스 호스트간에 이루어지므로, 한 라우터의 요구에 의해 속도를 증가시키는 경우, 이것에 의해 다른 라우터에서 혼잡 문제가 발생할 수 있다. 그림 10과 같은 망의 형태를 예로 들어보자.

호스트 A에서 호스트 B로 데이타를 전송하는 경우에 데이타의 전송 속도는 R3에서 요구하는 1Mbps로 맞추어 진다. 이런 경우에 R1의 버퍼는 비게 된다. 이때 R1에서 속도 증가를 요구해 호스트 A에서 전송 속도를 증가하게 된다면 이것이 R3의 혼잡 상태를 가중시키는 결과를 초래하므로 속도 증가의 경우에는 전체 망의 상태에 따라 신중하게 처리되어야 한다.

이 문제를 해결하기 위해서 본 논문에서는 각 호스트의 TCP 계층에서 혼잡 라우터 테이블(congesting router table)을 이용한다. 혼잡 라우터 테이블에는 각 커넥션에 따라 source quench 메시지를 보낸 라우터의 리스트와 각 라우터에서 요구한 속도가 기록되어 관리된다. 따라서, 속도를 증가시키는 경우에는 혼잡 라우터 테이블에 존재하는 전송 속도 중에서 최소의 속도를 가지는 라우터에서 속도 증가를 요구하였을 경우, 그리고 새롭게 요구된 속도가 테이블 내의 다른 속도 보다 작은 경우에만 속도를 증가시킨다. 그리고, 요구된 속도가 현재 제어 메시지를 보낸 라우터의 속도를 제외한 테이블 내의 최소의 속도보다 큰 경우에는 새롭게 요구된 속도가 다른 라우터의 혼잡 상태에 영향을 주지 않게 하기 위하여, 전송 속도를 제어 메시지를 보낸 라우터의 속도를 제외한 테이블 내의 최소의 속도로 데이타를 전송한다.

그림 11에 본 논문에서 제안하는 방식을 적용하기 위해 변경된 source quench 메시지의 구조를 나타내었다.

새롭게 수정된 source quench 메시지에는 기존의 source quench 메시지 안에 들어가던 IP 헤더와 64

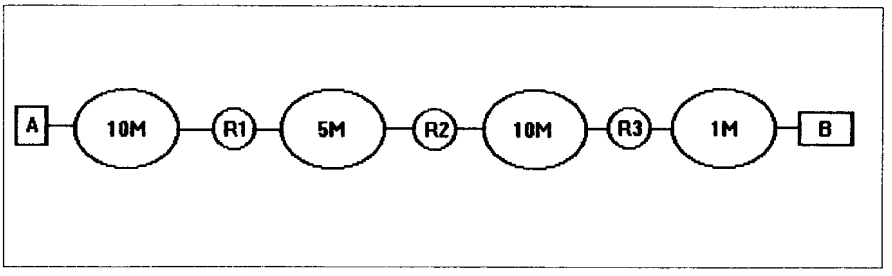

그림 10. 속도 증가의 문제점

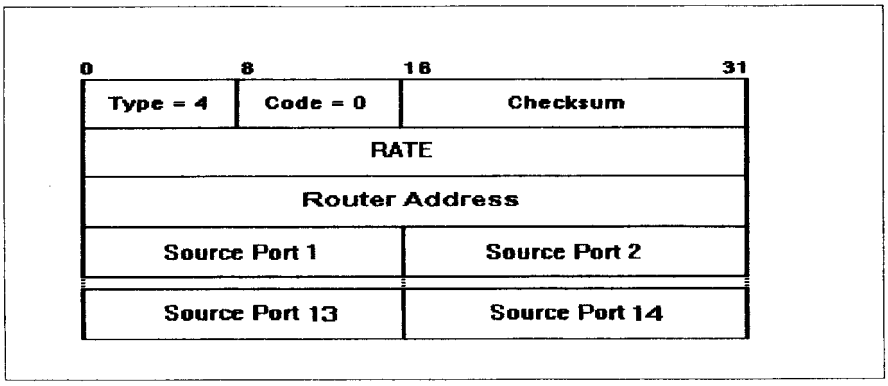

그림 11. 수정된 source quench 메시지 구조

bits의 데이타 대신에 새롭게 요구되는 전송 속도 RATE와 제어 메시지를 보내는 라우터의 주소, 그리고 포트 넘버가 들어간다. 위의 메시지를 받은 송신측의 IP 계층에서 IP 헤더를 제거하고 TCP 계층에 source quench 메시지 부분을 전달해 주면 TCP 계층에서는 라우터 주소와 RATE를 혼잡 라우터 테이블의 각 커넥션 부분에 기록하고 포트 넘버 영역을 참조하여 그에 해당하는 커넥션, 혹은 여러 개의 커넥션 속도를 RATE값에 맞도록 조정한다.

5) IP 모듈에서의 혼잡 제어 알고리즘

지금까지 기술한 IP 모듈에서 혼잡 상태를 제어하기 위한 알고리즘을 그림 12, 13, 14에 나타내었다.

Ⅴ. 시뮬레이션

1. 기본 사항

본 논문에서 제안하는 혼잡 제어 방식의 성능을 평가하기 위해 수행하는 시뮬레이션에서 기본이 되는

```
Loop :    c = buffer_check ( ) ; /* check the buffer condition */
          switch(c) {
            case 1 :  /* congest condition */
                      call send_decreasing( ) ;
                      break ;
            case 2 :  /* slow rate condition */
                      call send_increasing( ) ;
                      break ;
            default :  /* normal condition */
                      break ;
          }
        Goto Loop ;
```

그림 12. 게이트웨이에서의 혼잡 제어 알고리즘

```
buffer_check ( )
{
  n  = number of senders ;
  if ( Ri = Ro )
    return 0 ; /* normal condition */

  /* calculate the anticipated total buffer level */
  For ( k = 1 ; k ≤ n ; n = n+1 ) /* for each sender */
  {  /* ratio of the each sender's buffer count */
     γ  = byte count for kth sender ÷ Bt ;
     /* anticipated buffer count for each sender */
     Bc = Tk · γ · ( Ri - Ro) ;
     /* the anticipated total buffer amounts */
     Bt = Bc + Bt ;
   }
   if ( Bt ≥ β · (B - Bt) )  /* congest condition */
         return 1 ;
   else if ( ( ( Bt + Bt ) ) ≤ δ · B) /* slow rate condition */
         return 2 ;
   return 0 ; /* normal condition */
}
```

그림 13. 게이트웨이에서의 버퍼 모니터링 알고리즘

```
send_decreasing( )
{
  For ( k = 1 ; k ≤ n ; n = n+1 )
  {
     γ  = byte count for kth sender ÷ Bt ;
     Rn = α · γ · Ro ;
     send rate decreasing control packet ;
   }
}

send_increasing( )
{
  For ( k = 1 ; k ≤ n ; n = n+1 )
  {
     γ  = byte count for kth sender ÷ Bt ;
     Rn = γ · Ro ;
     send rate increasing control packet ;
   }
}
```

그림 14. 게이트웨이에서의 제어 패킷 전송 알고리즘

사항은 다음과 같다.

1) 소스 호스트에서 데이타를 보내는 형태로 FTP 소스와 On - Off 소스가 사용된다. FTP 소스의 특징은 최대 패킷 크기만큼의 데이타를 패킷에 실어 계속해서 보내고 수신측으로부터 최소의 패킷 크기를 가지는 ACK를 받게 된다. On - Off 소스는 데이타를 일정한 시간 간격을 두고 최대한의 속도로 보내고; 흐름 제어를 해주지 않는 특징이 있다.

2) 흐름 제어 방식은 고정된 윈도우 크기를 가지는 Generic 흐름 제어와 Jacobson에 의해 고안된 slow start 방식과 Karn에 의해 고안된 Karn's 알고리즘이 첨가된 JK 흐름 제어 방식이 사용된다.

3) 흐름 제어에 사용되는 윈도우의 크기는 600이고 각 라우터에서는 First - Come - First - Service 큐잉 방식이 사용되고, 각 라우터의 버퍼 용량은 440Kbytes이다.

4) 사용되는 오류 제어 방식은 Go - back - N 방식이 사용된다.

5) 사용되는 패킷의 크기는 FTP 데이타인 경우에는 1000bytes, ACK인 경우에는 40bytes가 사용된다.

6) 각 링크의 전송 지연 시간은 10M 링크인 경우에는 10msec, 5M 링크인 경우에는 20msec, 1M 링크인 경우에는 40msec, 560K 링크인 경우에는 70msec라고 가정하였다.

7) 시뮬레이션의 대상이 되는 망의 형태는 두 고속망이 하나의 저속의 망으로 연결되어 있는 경우를 가정하였다.

본 논문에서의 시뮬레이션은 REAL 4.0 네트워크 시뮬레이터를 사용하였다[18].

2. 시뮬레이션 방법 및 결과

두 고속망이 하나의 저속망으로 연결되어 있을 때, 중간 라우터에서 발생하는 혼잡 문제에 대해 각 혼잡 제어 방식에 대한 성능 평가를 수행한다.

1) Generic 흐름 제어

첫 번째 수행할 시뮬레이션은 호스트 A가 고정된 윈도우의 크기를 사용하는 흐름 제어 방식을 사용하고, 호스트 B는 on - off 형태로 데이타를 0.5초 동안 2Mbps의 속도로 전송하고 0.5초 동안은 데이타를 전송하지 않는 것을 반복하는 호스트인 경우이다. 이 경우, 처음에 호스트 A에서는 윈도우 크기만큼의 데이타와 그 동안 수신측으로부터 도착한 ACK 수만큼의 데이타를 빠른 속도로 보내고, 그 후에는 ACK가 도

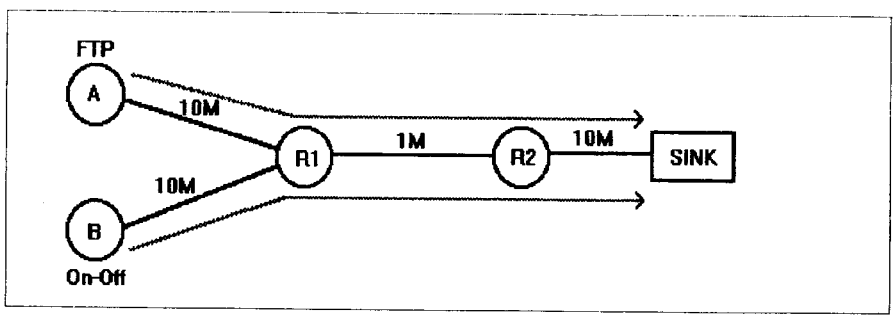

그림 15. 시뮬레이션 망 형태

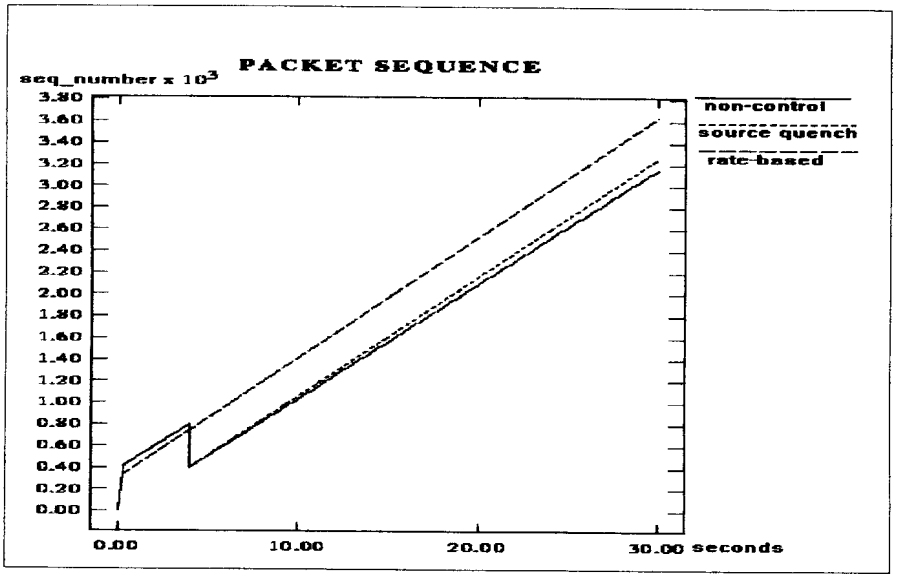

그림 16. Generic packet sequence

착할 때마다 그에 해당하는 양만큼의 데이타를 전송한다.

혼잡 제어를 해주지 않는 경우 R1에서는 호스트 A에서 보낸 데이타중 처음에 빠르게 보낸 데이타의 일부가 유실되고 호스트 B에서 보낸 데이타는 흐름 제어를 해주지 않기 때문에 계속해서 일부가 유실된다.

혼잡 제어 방식으로 source quench 방식을 쓰는 경우에는 R1에서 버퍼가 오버플로우 될 때 ICMP메시지를 호스트 A와 B로 전달해 주므로 처음 순간에만 데이타의 일부가 유실된다.

혼잡 제어 방식으로 본 논문에서 제시하는 방식을 쓰는 경우에는 R1에서는 버퍼의 일정량이 차게 되면 각 호스트로 적절한 속도로 데이타를 보내 줄 것을 알려주므로 데이타의 손실이 전혀 없게 된다.

그림 16에 호스트 A에서 보낸 패킷의 순서에 대한 그래프가 나와 있다.

그림 16에서 source quench나 non - control 방식일 때에는 패킷 손실로 인한 재전송을 하는 모습이 나타나 있고 본 논문에서 제안하는 rate - based 방식을 사용할 때에는 패킷 손실이 없기 때문에 패킷 순서가 계속 증가하는 모습이 나타나 있다.

그림 17은 라우터 R1에서의 버퍼 큐잉 상태를 보여 준다.

그림 17에서 non - control 방식일 때에는 버퍼가 계속 오버플로우 되는 모습이 나타나 있고 source quench 방식일 때에는 처음에 오버플로우 되었다가 속도를 줄여 queue length가 감소한 모습이 나타나 있으며, rate - based 방식을 사용할 때에는 queue length가 위험 수위에 이르렀을 때 속도를 줄여 queue length가 줄어들다가 일정 수준을 유지하는 모습이 나타나 있다.

표 1에 호스트 A에서 보낸 데이타에 대한 시뮬레이션 결과를 나타내었다.

표 1에서 처리량(throughput)은 라우터 R1에서 매초당 보낸 패킷의 평균 갯수이고, 재전송(retransmission rate)은 매초당 보낸 재전송 패킷의 평균 갯수이다. 표 1에 나타난 바와 같이 본 논문에서 제안하

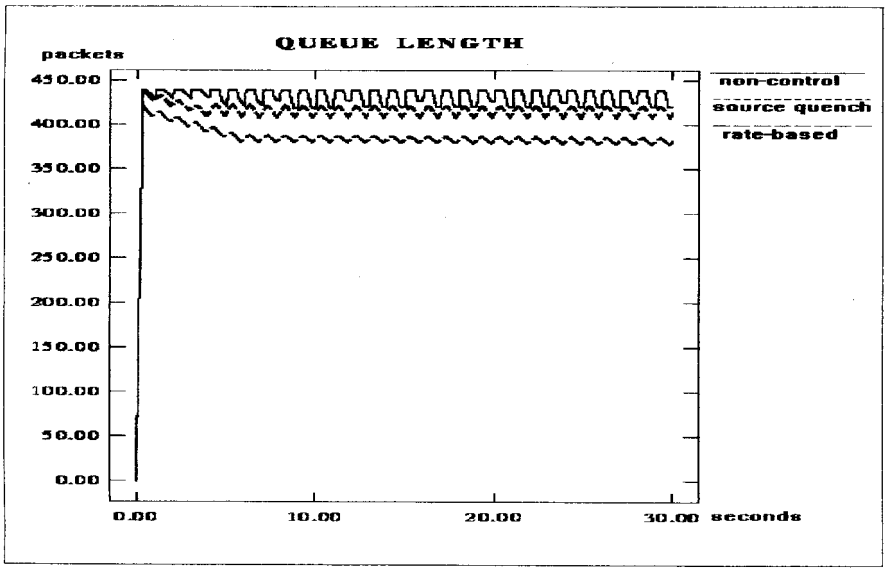

그림 17. Generic queue length

<표 1> Generic 시뮬레이션 결과

HOST A	Throughput (packets/s)	Queueing Time (sec)	Round Trip Delay (sec)	Drop (packets)	Retransmit Rate (packets/s)
Non Control	105.83	3.34	3.57	19	13.79
Source Quench	109.52	3.23	3.31	19	13.79
Rate - based Control	109.83	2.96	3.08	0	0

는 방식을 사용할 때 가장 좋은 성능을 나타냄을 알 수 있다.

2) JK 흐름 제어

두 번째 수행할 시뮬레이션은 호스트 A가 JK 흐름 제어 방식을 사용하고, 호스트 B는 on - off 형태로 데이타를 0.5초 동안 1.5Mbps의 속도로 전송하고 0.5초 동안은 데이타를 전송하지 않는 것을 반복하는 호스트인 경우이다.

JK 흐름 제어에서 사용하는 slow start 방식은 송신측에서 흐름 제어를 위한 윈도우와 혼잡 문제 해결을 위한 혼잡 윈도우를 가지고 데이타를 전송하는 방식이다. 혼잡 윈도우의 크기는 처음에 1 이었다가 ACK를 받을 때마다 지수 함수적으로 증가하기 때문에 기존의 방식에서 처음에 윈도우 크기만큼 데이타를 빠르게 보냄으로써 발생하였던 라우터에서의 오버플로우 문제를 해결하고, 혼잡이 발생하였을 때 그 크기가 다시 줄어들어 혼잡 문제 해결에 도움을 준다. 하지만, slow start 방식을 쓰는 경우에도 망의 부하가 과중한 경우에는 기존의 일반적인 흐름 제어 방식에서와 마찬가지로 버퍼 오버플로우가 발생할 수 있다.

그림 18에 JK 흐름 제어를 사용할 때 호스트 A에서 보낸 패킷의 순서에 대한 그래프가 나와 있다.

그림에서 보듯이 혼잡 제어를 해주지 않거나 기존의 source quench방식을 사용하는 경우에는 버퍼 오

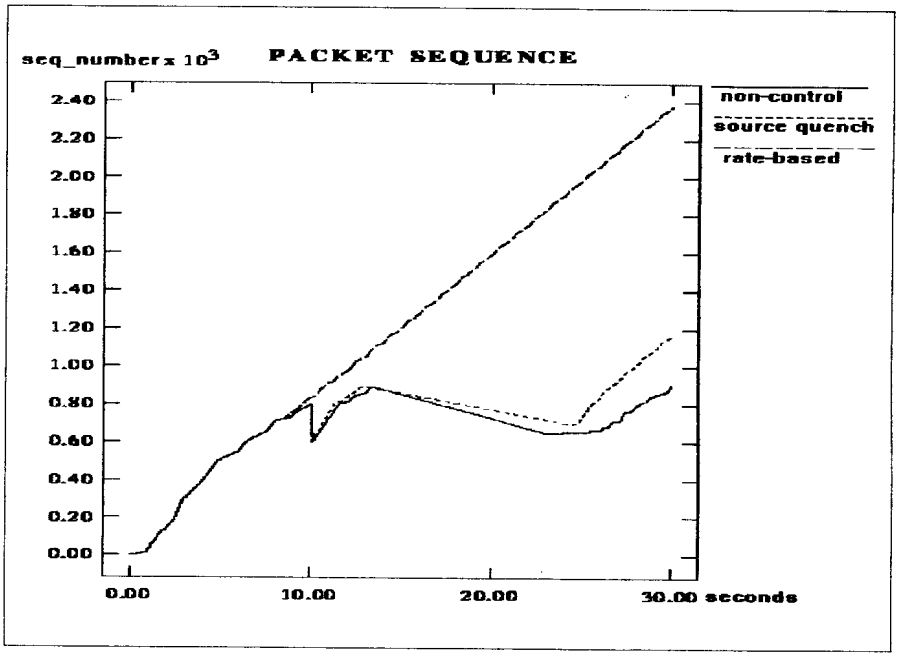

그림 18. JK packet sequence

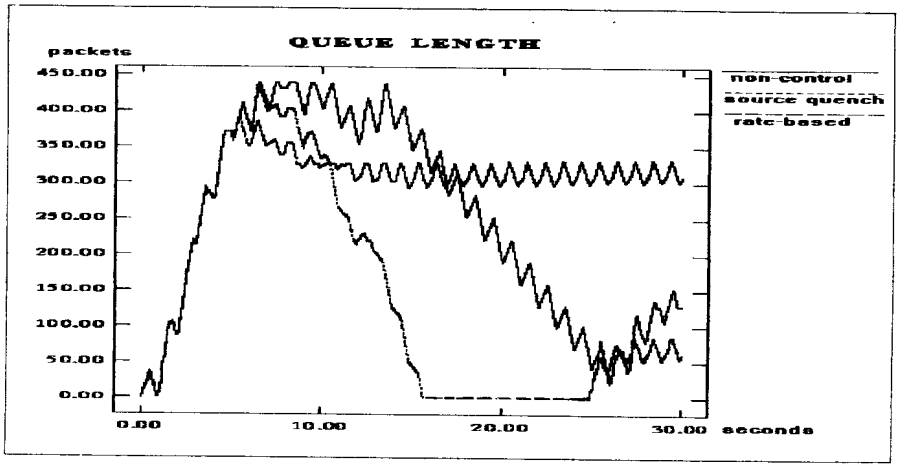

그림 19. JK queue length

버플로우로 인한 패킷의 손실로 재전송을 하는 모습이 나타나 있고, 본 논문에서 제안하는 방식을 사용하는 경우에는 버퍼 오버플로우가 발생하지 않기 때문에 계속해서 패킷을 순서대로 전송하는 모습이 나타나 있다.

그림 19는 JK 흐름 제어를 사용할 때 라우터 R1에서의 버퍼 큐잉 상태를 보여 준다. 혼잡 제어를 해주지 않는 경우나 기존의 source quench 방식을 쓰는 경우에는 버퍼 오버플로우가 발생했을 때, 송신측의 혼잡 윈도우의 크기가 줄어서 보내는 데이타의 양이 적어짐으로 인해 버퍼 큐잉 레벨이 급격히 감소하는

<표 2> JK의 시뮬레이션 결과

HOST A	Throughput (packets/s)	Queueing Time (sec)	Round Trip Delay (sec)	Drop (packets)	Retransmit Rate (packets/s)
Non Control	36.76	1.69	1.80	4	0.10
Source Quench	44.69	1.43	1.55	0	0.10
Rate - based Control	74.31	2.35	2.47	0	0

모습이 나타나 있다.

표 2에 JK 흐름 제어 방식을 사용하는 경우 호스트 A에서 보낸 데이타에 대한 시뮬레이션 결과를 나타내었다.

3. 결과 분석

본 논문에서 제안한 혼잡 제어 방식을 사용하여 시뮬레이션을 수행한 결과, 제안한 혼잡 제어 방식이 기존의 방식보다 망이 과부하 상태에 있을 때 보다 안정한 상태를 나타냄을 알 수 있었다.

본 논문에서 제안한 혼잡 제어 방식은 고속망이 하나의 저속망으로 연결되어 있는 경우에 최적의 성능을 나타내며, 속도가 더 느린 망이 계속해서 연결되는 경우에는 속도를 줄인 후에도 그 전에 앞의 라우터에 큐잉되어 있던 데이타에 의해 버퍼 오버플로우가 발생할 가능성이 있다. 그러나, 이 경우에도 본 논문에서 제안하는 방식이 기존의 제어 방식보다 혼잡 상태 해결에 보다 효과적임을 알 수 있다.

Ⅵ. 결 론

본 논문에서는 현재 통신 망에서 커다란 문제점으로 나타나고 있는 혼잡 문제 해결을 위해 혼잡 상태 방지를 위한 제어 메카니즘을 제안하였다.

Internet의 표준 프로토콜로 사용되고 있는 TCP/IP는 혼잡 문제를 해결하기 위해 source quench 메카니즘을 사용하고 있는데, 이 방식에는 여러 미흡한 점이 많다. 이에 본 논문에서 제안하는 혼잡 제어 방식을 IP 모듈에 적용시킴으로써 source quench 메카니즘의 단점을 보완하고 전체적인 망의 성능 향상을 도모하였다.

본 연구를 토대로 하여, 향후에는 TCP/IP 프로토콜뿐만 아니라, 새롭게 등장하고 있는 여러 고속의 전송 프로토콜에도 적용될 수 있는 제어 방식을 개발해야 할 것이다.

참고문헌

1) J. Postel, RFC 792, "Internet Control Message Protocol"
2) John Nagle, RFC 896, "Congestion Control In IP/TCP Internetworks"
3) R. Braden, J. Postel, RFC 1009, "Requirements for Internet Gateways"
4) A. Mankin, RFC 1254, "Gateway Congestion Control Survey"

5) W. Prue, J. Postel, RFC 1016, "Something a Host Could Do with Source Quench : The Source Quench Introduced Delay(SQuID)"
6) Van Jacobson, "Congestion Avoidance and Control," ACM SIGCOMM 88 Stanford California August 16 - 19, 1988
7) "XTP Protocol Definition," Protocol Engines, Rev 3.6, January 1992
8) David R. Cheriton, "VMTP as the Transport Layer for High - Performance Distributed Systems," *IEEE Communication Magazine*, June 1989
9) Bemd Heinrich, Kai Jakobs, Thomas Meuser, "XTP, VMTP or TCP/IP?," Silicon Valley Net. Conference, 1992
10) Willibald A. Doeringer, Doug Pykeman, "A Survey of Light - Weight Transport Protocols for High - Speed Networks," *IEEE Transaction on Communications*, Vol. 38, No. 11, November 1990
11) Sylvie Dupuy, Wassin Tawbi, Eric Horlait, "Protocols for high - speed multimedia communications networks," Computer Communication, Vol. 15, No. 6, July/August 1992
12) Partho P. Mishra, Hemant Kanakia, "A Hop by Hop Rate - based Congestion Control Scheme," *Computer Communication Review*, Vol. 22, No. 4, October 1992
13) Nen - Fu Huang, Chiung - Shian Wu, "A Distributed Congestion Control For High - speed Internetworks," Proceedings of the First International Conference on Computer Communication and Networks, 1992
14) Thomas F. La Porta, Mischa Schwartz, "Architectures, Features, and Implementation of High - Speed Transport Protocols," *IEEE Network Magazine*, May 1991
15) W. Richard Stevens, "TCP/IP Illustrated, Volume 1, The Protocols," Addison - Wesley, 1994
16) Douglas E. Comer, "Internetworking with TCP/IP, volume I," Prentice - Hall, 1991
17) Craig Partridge, "Gigabit Networking," Addison - Wesley, 1994
18) Columbia University, "REAL 4.0," Manuals