

碩士學位論文

# 예측기반 RED 알고리즘



濟州大學校 大學院

情報工學科

金 福 心

2001 年 12 月


# 예측기반 RED 알고리즘

指導教授 安 基 中

金 福 心

이 論文을 工學 碩士學位 論文으로 提出함

2000 年 12月

 제주대학교 중앙도서관  
金福心の 工學 碩士學位 論文을 認准함

審査委員長 \_\_\_\_\_ 印

委 員 \_\_\_\_\_ 印

委 員 \_\_\_\_\_ 印

濟州大學校 大學院

2000 年 12 月

# Prediction based RED Algorithm

Bog-Sim Kim

(supervised by professor Khi-Jung Ahn)

A thesis submitted in partial fulfillment of the requirement for the degree



of Master of Engineering

2000. 12.

This thesis has been examined and approved.

Department of Information Engineering

GRADUATE SCHOOL

CHEJU NATIONAL UNIVERSITY

## 목 차

Summary .....	i
I. 서 론 .....	1
II. TCP/IP 망에서의 트래픽 제어기술 .....	3
1. TCP/IP망의 구조 .....	3
2. TCP/IP망에서의 트래픽 제어기술과 요구사항 ...	8
III. 예측기반 RED 알고리즘 .....	17
1. 예측기반 RED .....	17
IV. 시뮬레이션 및 성능분석 .....	26
1. 시뮬레이션 모델 .....	26
2. 성능측정 방법 .....	27
3. 시뮬레이션 결과 .....	28
V. 결 론 .....	36
참고문헌 .....	38

## Summary

There are many complex and various networks in the world. We need the various substructured-techniques to link them efficiently each other. The techniques which apply to each network users should be different in it's place, time, person and environment to serve their requirements, and it is necessary to serve the environment of various interconnected networks with the method which is simple and more efficient. and easy to use.

In Internet, the management of traffic congestion involves the policy which drops a packet in advance. In this case, router drops one or more packets to improve the network performance before when its output buffer becomes full. RED(Random Early Detection) is a good example for this technique.

The Combination of TCP and RED algorithm is very effective in preventing from performance degradation in Internet. It's so useful to control the traffic congestion while keeping the network performance high. But it uses the same Packet Dropping Probability regardless of the length of RTT(Round Trip Time). Without having the chance to use the bandwidth which can be used for long RTT, the unbalance with short RTT happens. To reduce the unbalance with long RTT connection, we need another way to calculate the probability of packet dropping.

a new congestion avoidance algorithm, prediction based RED Algorithm is proposed to estimate the queue length variation and applied to the packet drop probability to control the traffic congestion proactively. And the simulation result showed me it can improve the translation and packet loss rate.

# I. 서론

오늘날 컴퓨터 통신은 우리의 일상 생활에 점점 더 중요한 역할을 하고 있으며, 생활방식과 업무처리 방식을 바꾸어 놓았다. 역시 업무에 대한 결정은 과거보다 더 신속해야 하며 의사 결정자에게는 정확한 정보의 신속한 입수가 요구되고 있다.

인터넷에서 주로 사용되고 있는 TCP(Transmission Control Protocol)는 송신단에 일정 시간동안 Ack(Acknowledgment) 패킷이 도착하지 않아 타임아웃이 발생하거나, 데이터 패킷의 손실을 의미하는 중복 Ack을 받으면, 네트워크에 혼잡이 일어났다고 가정한다. 그리고 혼잡 윈도우(congestion window)의 크기를 줄이고 전송 속도를 천천히 증가시키는 저속출발(Slow start)과 혼잡회피(Congestion Avoidance)를 실행하여 네트워크 혼잡을 완화시키고 패킷 손실을 줄인다. 즉, 전송 중에 발생하는 패킷 손실의 원인을 네트워크 혼잡에 두고 이를 줄이기 위한 방향으로 개선되어 온 TCP 프로토콜은 통신의 양 종단간 신뢰성 있는 데이터 전송을 보장하며 인터넷에서 가장 보편적으로 쓰인다.

TCP 통신망에서의 혼잡 제어는 크게 두 가지로 나눌 수 있다. 첫째로는 단말 혼잡 제어(end-to-end congestion control)를 들 수 있다. 이는 TCP connection의 양단, 즉 전송 측과 수신 측에서 패킷을 제어하는 것을 의미한다. 두번째로는 게이트웨이 혼잡 제어를 들 수 있다. 이는 단말 혼잡 제어를 보완하기 위해서 통신망의 게이트웨이 상에 혼잡 제어 기법을 구현하는 방법이다. 가장 대표적인 방법으로는 RED(random early detection)를 들 수 있다. RED는 기존의 Tail-Drop 방식의 게이트웨이의 단점을 보완하여 보다 나은 성능을 나타낸다.

체증으로 인한 네트워크의 처리율 저하와 지연 증가를 예방하거나 이러한 현상이 발생했을 때 이를 해소하기 위해서 여러 가지 체증회피 혹은 체증 제어 방안들이 이용되고 있다. 신뢰성 있는 종단간 데이터 전송을 위해 현재 인터넷망에서 널리 사용되고 있는 TCP의 체증 회피 및 제어 알고리즘은 윈도우 흐름 제어 및 망으로부터의 묵시적 신호를 이용한다. 이 알고리즘은 커백션이 네트워크에서 일정량의 체증을 겪거나 수신측 윈도우에 의해 제한을 받기 전까지는 새로운 TCP 연결이 지수적인 전송

를로 윈도우를 전송하는 것을 허용한다. 패킷 손실로 인해 체증이 검출되면 TCP는 윈도우 크기를 반으로 줄이고 체증 회피 단계를 시작한다.(S. Floyd 와 V. Jacobson 1993) (W. Stevens, 1997)

RED(Random Early Detection)알고리즘은 망으로부터 TCP 소스로 전해지는 망 상태 신호를 적절히 조절함으로써 TCP프로토콜이 갖는 혼잡 회피 기능의 효과를 극대화한다. 망으로부터 TCP 소스로 전해지는 목시적인 망 상태 신호라 함은 패킷의 손실을 의미하는데 RED 알고리즘이 망 내부의 라우터에 구현되면 라우터 버퍼가 넘칠 때(체증)까지 기다리지 않고 사전에 그 조짐을 감지하여 인위적으로 패킷을 폐기해 줌으로써 TCP 프로토콜이 미리 트래픽의 망 내 유입을 조절할 수 있게 한다. 이렇게 함으로서 체증을 사전에 예방하고 결과적으로 망의 수율을 높이고 지연을 줄이게 된다. 그러나 TCP의 윈도우 흐름 제어에 기반하는 혼잡 회피/제어 알고리즘은 TCP연결간 성능향상의 문제점을 가지고 있다.

TCP 체증 회피/제어 알고리즘이 망 내부의 RED 알고리즘과 결합될 경우, 이 불공정성이 어느 정도 해소되는 것으로 연구 결과가 보고되어 있다. 그러나 이는 단일 라우터 내부의 정보에만 전적으로 의존하는 것이므로 그 효과에 있어 일정한 한계가 있다.(S. Floyd 와 V. Jacobson 1993).

RED 알고리즘은 라우터에서 평균 큐 사이즈가 어느 정도 되면 확률을 구하여 패킷을 폐기하기 시작한다. 이때 TCP 연결의 상태에 상관없이 혼잡을 경험한 TCP 연결의 패킷이 그렇지 않은 것보다 폐기될 확률이 상대적으로 커지게 된다.

본 논문에서는 패킷 손실율을 적게 하고, 지연을 최소화하면서 처리량을 증대시키고 더 나아가서는 Link의 효율을 증가시키기 위한 해결책으로, 패킷 폐기 확률을 계산, 적용하는데 있어 다음에 오는 큐 길이를 예측하여 예측치를 적용시킨 알고리즘을 사용함으로써 처리율의 성능향상을 보였다.

2장에서는 TCP/IP 망에서의 트래픽 제어기술과 기존 RED알고리즘에 대해서 기술하며, 3장에서는 본 논문에서 사용된 개선된 RED 알고리즘 즉, 예측기반 RED알고리즘을 소개한다. 4장에서는 시뮬레이션 및 성능분석으로 모의 실험 개선된 알고리즘을 통해 나타난 모의 실험 결과를 제시하며, 5장에서는 요약 및 결론을 맺는다.

## II. TCP/IP 망에서의 트래픽 제어기술

### 1. TCP/IP망의 구조

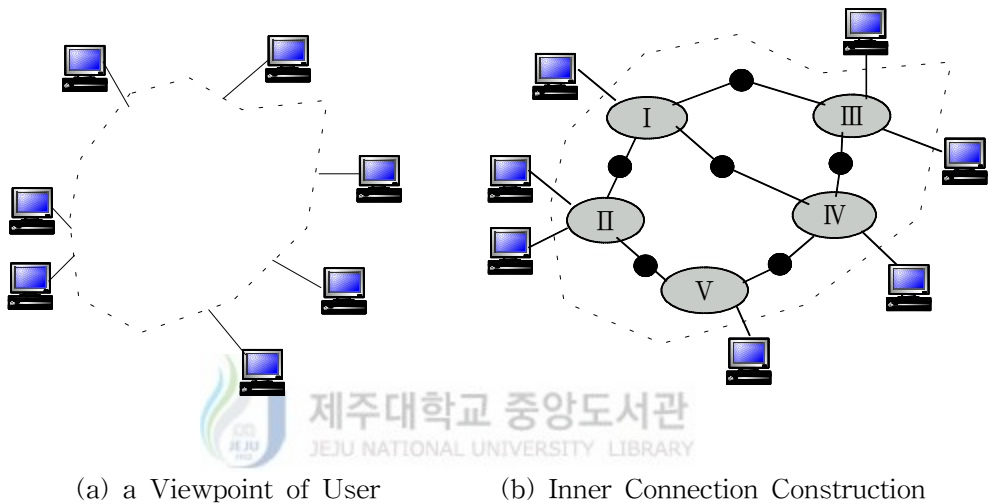


Fig. 1 TCP Network Construct

#### 1) TCP/IP

##### (1) TCP/IP의 배경

TCP/IP는 인터넷워킹과 상호 운용성을 위한 해결책 중 하나이다. 단순히 두개의 프로토콜임을 넘어서 TCP/IP는 DEC(Digital Equipment Corp., Maynard, MA)의 미니 컴퓨터와 Sun Microsystems Inc(Mountain View, CA)의 워크스테이션 같은, 서로 다른 통신매체를 가능하게 해주는 구조(architecture)이다. 이 통신은 LAN(Local Area Network), MAN(Metropolitan Area Network), WAN(Wide Area Network) 등을 TCP/IP는 모두 지원한다.



오늘날의 인터넷은 1969년에 ARPANET(Advanced Research Projects Agency Network)으로 태어났고, DARPA(U.S Defense Research Projects Agency)가 후원하였다. ARPANET의 목적은 패킷 스위칭이라는 통신 기술의 생존능력을 시험하고 결정하는 것이었다. 원래의 ARPANET을 구축하는 계약은 Bolt, Branket, & Newman(지금은 BBN Communication Inc. Cambridge, MA)에 주어졌다.

ARPANET은 1969년 9월 SRI(Standard Research Institute), UCSB(University of California at Santa Babara), UCLA, 및 University of Utah등 네 곳에서 개통되었다. 초기시험은 성공적이었고 ARPANET은 급속히 성장하였다. 그이후 대학과 산업연구기관들의 협조적 네트워크를 구성되었고, NFS는 1981년 CSNET(Computer science Network)에 대한 자금지원을 승인하였다. 1894년 ARPANET는 MILNET(비밀 군사용)와 ARPANET(비군사용)로 구분되었다. 1990년 6월 미 국방부는 ARPANET을 폐기하였다.

ARPANET으로부터는 LAN과 패킷스위칭 같은 수많은 데이터 통신 기술에 중요한 영향을 끼치게 되는 교훈들을 얻게 되었다. 최근에 NSFNET골격은 44.736Mbp의 T-3을에 의한 송신으로 변환되었다. 현재 IPv4 인터넷 프로토콜을 사용하였다. 점점 더 많은 사람들이 인터넷에 접속을 진행중이며, 이런 추세로 나간다면 IP주소가 향후 부족할것으로 예상된다. 또한 새로운 멀티미디어 트래픽을 효과적으로 수용하기위해서도 IPv4의 개정이 필요하게 되었다. 이를 충분히 수용하기 위해서 IPv6(IPng)가 나오게 되었고, 지금도 이것의 적용하는 문제를 두고 검토하는 중이다. 이런 추세로 나간다면 곧 IPng로 바뀌게 될 수도있다.

TCP/IP상에서의 원활한 흐름을 제어하기 위해서는 트래픽 제어 기술의 필요한데, 다음 절에서 TCP/IP에 대한 소개 및 구성형식을 살펴보겠다.

## (2) TCP

TCP는 가장 일반적으로 데이터 전송에 사용되는 전달 계층 프로토콜이다. 파일 전송, 전자 메일, 원격 단말 접속, Web 접속, 클라이언트-서버 등의 응용 프로그램을 지원하며, 또한 TCP는 종단간 프로토콜이다. TCP는 직접적으로 네트워크의 상태를 파악할 수 없지만 상대방 종단과의 대화를 통해 네트워크의 상태를 유추하고 이에 따라 흐름 제어, 혼잡 제어를 수행한다.

TCP는 연결 지향성(connection-oriented)이고 신뢰성 있는 바이트 스트림(byte stream) 서비스를 제공한다. 연결 지향성이라는 말은 TCP를 이용하는 2개의 응용(클라이언트와 서버)이 데이터를 교환하기 전에 서로 TCP 연결을 확립해야 함을 의미한다. 이처럼 TCP 연결에는 반드시 서로 통신하는 두 개의 종단점이 필요하다. 따라서 브로드캐스팅(Broadcasting)이나 멀티캐스팅(Multicasting)과 같은 개념은 TCP에 적용되지 않는다.

응용 데이터는 TCP가 전송하기에 적합한 크기로 나뉘어진다. TCP에 의해 IP로 전달되는 정보단위는 세그먼트(segment)라고 부른다. TCP는 세그먼트를 보낼 때 타이머를 설정하고 상대방으로부터 세그먼트를 수신했다는 것을 알리는 Ack(Acknowledgement)를 기다린다. 만약 시간 내에 Ack가 되돌아오지 않으면 세그먼트는 재전송된다. TCP는 헤더와 데이터에 체크섬(Checksum)을 이용한다. 이것은 종단 대 종단 체크섬으로 목적은 데이터가 전송중에 변화되었는지를 검사하기 위한 것이다. 세그먼트가 정확하지 않은 체크섬을 가지고 도착하면 TCP는 해당 세그먼트를 버리고 그것을 받았다는 Ack를 보내지 않는다.(강문철 2001)(W.R. Stevens 1998)



■ TCP Header (RFC793 1981)(RFC1122 1989)

TCP 데이터는 TCP 헤더와 함께 IP 데이터 그램(Datagram)에 Fig. 3과 같이 포함되어 있다. TCP는 프로토콜 데이터 단위를 세그먼트로 한다. 헤더 형식은 아래의 Fig. 2와 같다. 옵션이 지정되지 않으면 보통 크기는 20바이트이다.

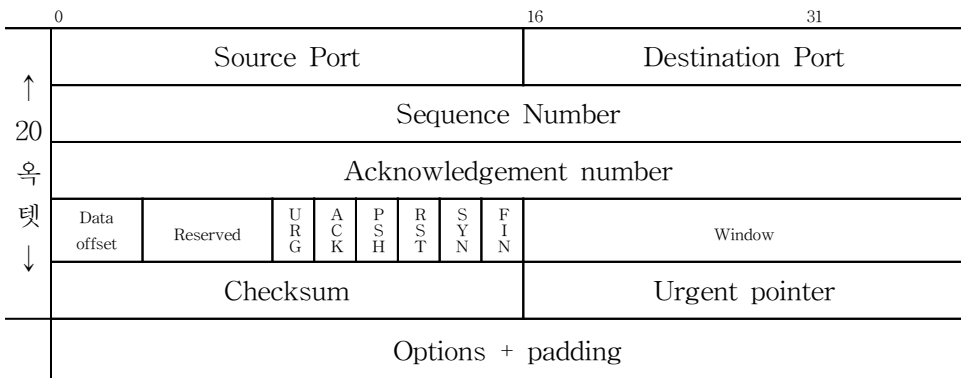


Fig. 2 TCP Header

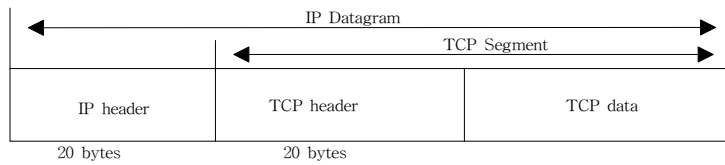


Fig. 3 Encapsulated TCP Segment of IP Datagram

각 필드의 내용과 사용은 아래와 같다.

- Source port(16 bits) : Source TCP user
- Destination port(16 bits) : Destination TCP user
- Sequence number(32 bits) : 세그먼트의 첫 번째 바이트를 나타내는 것으로 송신 호스트의 TCP로부터 수신 호스트의 TCP로 가는 데이터 스트림의 바이트를 구분
- Acknowledgement number(32 bits) : 송신 호스트가 수신하고자 하는 다음 순서 번호
- Data offset(4 bits) : 헤더에서 32비트 워드의 수
- Reserved(6 bits) : 미사용 영역
- Flags(6 bits)
  - URG : 긴급 포인터 유효
  - ACK : 확인 응답 번호 유효
  - PSH : 수신 호스트는 데이터를 가능한 빨리 응용계층으로 전송
  - RST : 연결을 재설정
  - SYN : 연결을 초기화, 순서 번호 동기화
  - FIN : 데이터 전송 종료
- Window(16 bits) : Ack없이 전송될 수 있는 세그먼트 수
- Checksum(16 bits) : 에러 검색 코드
- Urgent Pointer(16 bits) : 긴급 데이터의 마지막 바이트가 갖는 순서 번호를 나타내기 위해 세그먼트의 순서 번호 필드에 추가되는 양의 정수 옵셋
- Option(variable) : 기능에 따라 지정됨

■ IP Header RFC791 1981)(RFC1349 1992)

여기서는 공식적으로 정의되어 있는 IP버전 4에 대해서 살펴보겠다. IPv4는 앞으로 IPv6으로 대체되겠지만 현재는 TCP/IP 네트워크들에 쓰이고 있는 표준 IP이다.

IP 실체들 사이의 프로토콜은 IP 데이터그램 형식을 보면 가장 잘 알수 있으며 Fig. 4와 같다.

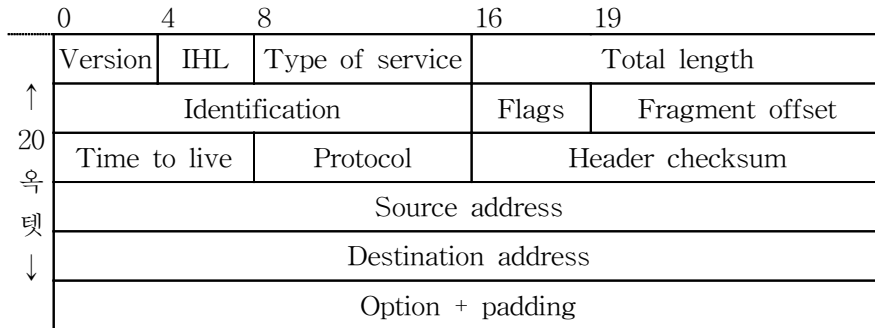
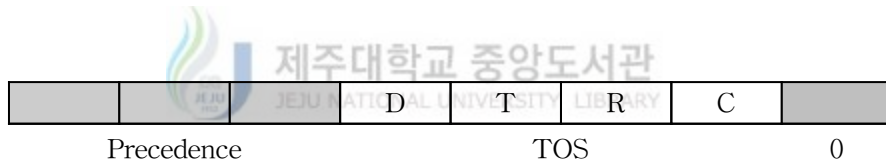


Fig. 4 IP datagram



D : Minimize delay

R : Maximize reliability

T : Maximize throughput

C : Minimize cost

<precedence>

<tos>

111 Network control

1000 Minimize delay

110 Internetwork control

1000 Maximize throughput

101 Critical

0010 Maximize reliability

100 Flash override

1000 Minimize monetary cost

011 Flash

1000 Normal service

010 Immediate

001 Priority

000 Routine

Fig. 5 IP service type

- 버전(4비트) : 버전번호로 4이다.
- IHL(internet header length)(4비트) : 헤더의 길이, 32비트 워드 단위로 표시.
- TOS(type of service)(8비트) : 데이터그램을 처리하는 방법을 표시
- 전체길이(16비트) : IP 데이터그램의 전체 길이. 2byte(16bit)로서 65,536까지 정의
- 식별자(16비트) : 단편화(fragmentation)에 이용. 각단편은 순서번호로서 식별.
- 플래그(3비트) : 현재 2 비트만 정의, 단편화를 처리.
- 단편 오프셋(fragment offset)(13비트) : 본래 데이터그램 내의 데이터 오프셋을 나타내는 포인터. 64비트 단위로 표시.
- 수명(time to live)(8비트) : 데이터그램을 폐기하기 전에 전달될수 있는 홉수.
- 프로토콜(8비트) : 목적지에서 데이터 필드를 수신할, 다음 상위 레벨의 프로토콜. 어느 상위계층 프로토콜 데이터가 데이터그램에 캡슐화되었는지를 표시(TCP, UDP, ICMP 등)
- 헤더 체크섬(16비트) : 헤더에만 적용되는 에러검출 코드이다.
- 발신지 주소(32비트) : 데이터그램의 원래 발신지를 표시
- 목적지 어드레스(32비트) : 데이터그램의 최종목적지를 표시
- 선택기능(가변) : IP 데이터그램의 많은 기능을 제공. 경로설정, 타이밍, 관리, 명령을 제어하는 필드를 갖는다.

## 2. TCP/IP 망에서의 트래픽 제어기술과 요구사항

본 절에서는 TCP 알고리즘의 요구 사항 및 흐름제어를 구현하기 위해 사용되는 파라미터들과 망의 상황을 송신원에 전달하기 위해 사용되는 TCP와 IP 프로토콜의 구조에 대해서 알아본다.

### 1) 알고리즘 구현시 요구사항

TCP/IP상에서 서비스를 제공하기 위한 알고리즘 구현시 요구사항을 만족시키기는

어렵지만, 고려해야 할 요구사항은 다음과 같다(Jain 등 1996), (Jain 등 1997b)

### (1) 효율성(efficiency)

트래픽 제어 알고리즘에서 가장 중요하게 요구되고 있는 사항은 효율성이다. 효율성은 망 내의 데이터에 대한 처리율(throughput)로 표현될 수 있다. 자원을 공유하는 망 내에서 여러 송신원들은 망 자원을 차지하기 위하여 경쟁하게 된다. 망 자원 중에 가장 대표적인 것은 대역폭이다. 즉, 망 내의 여러 송신원들은 대역폭을 많이 얻기 위하여 경쟁하게 되는데, 우수한 트래픽 제어 알고리즘은 전체 송신원들에게 할당하는 대역폭의 총합이 망내에서 제공할 수 있는 대역폭에 최대한 가깝게 설계된 알고리즘이다. 동시에 반드시 전송하고자 하는 모든 송신원들은 최소 QoS(Quality of Service)를 만족시켜야 한다.

### (2) 공정성(fairness)

모든 연결들에 대역의 공평한 할당을 하는 것은 트래픽 제어의 목적중 하나이다. 트래픽 제어 알고리즘은 자원의 할당을 요구하는 송신원에게 자원을 공평하게 분배하여야 한다. 즉 어느 한 송신원이 망의 자원을 독점하도록 해서는 안된다(Pisanpattana-kul 등 1998), (Yamamoto 등 1998). 공정성을 향상시키기 위한 대표적인 것으로 max-min 할당이라는 대역폭 할당 알고리즘이 있다. 이 알고리즘은 모든 경쟁하는 송신원 중에서 가장 작은 대역폭을 할당받는 송신원의 대역폭의 크기를 최대로 하는 알고리즘이다(Hou 등 1997), (Vandalore 1998). max-min 할당은 이론적인 값으로 여러 가지 원인에 의하여 실제 알고리즘에서는 구현되기 힘들다. 따라서 실제 알고리즘에서는 불공정성이 나타나며 이를 정량적으로 표현한 것이 공정성 인자(fairness index)이다. 1부터 n까지의 송신원이 이론적인 max-min 알고리즘에 의하여 할당받은 대역폭을 벡터로 표시한  $\{y_1, y_2, y_3, \dots, y_n\}$ 과 실제 구현된 알고리즘에 의하여 할당된 대역폭을 벡터로 표시한  $\{z_1, z_2, z_3, \dots, z_n\}$ 에 의하여 공정성 인자를 다음의 식에 의하여 구할 수 있다.

$$fairness\ index = \frac{[\sum_i x_i]^2}{n \sum_i x_i^2} \quad (1)$$

여기에서  $x_i = \frac{y_i}{z_i}$  이다. (Fahmy 등 1998a), (Jain과 Babic 등 1996), (Lai 등 1998), (Nabeshima 등 1999)

### (3) 낮은 부하(low load)

트래픽 제어 알고리즘을 위한 제어 신호는 가능한 낮은 부하를 가져야 한다. 즉 제어 신호 자체가 망내에서 부하가 될 수 있으므로 이를 최소화하여야 한다. 제어 신호가 너무 많으면 대역폭이 제어 신호에 의하여 낭비되며 제어 신호가 망의 체증을 유발시킬 수 있다.

### (4) 반응성(responsiveness)

트래픽 제어 알고리즘은 망 내의 목표 전송률과 연결된 송신원의 수가 지속적으로 변할 때, 이에 따라 사용 가능한 자원을 동적으로 할당하여야 한다.

### (5) 신속성(fastness)

트래픽 제어 알고리즘은 망 내의 부하에 따라 신속하게 변화하여야 한다. 즉 망 내의 가용대역폭이 발생한 경우 이를 즉각적으로 감지하여 송신측에 제어신호를 보냄으로써 이를 사용할 수 있도록 하여야 한다. 어떤 알고리즘이 같은 효율성과 공평성을 가지고 있다면 보다 신속하게 반응하는 알고리즘이 우수한 알고리즘이다. 트래픽 제어 신호가 너무 늦으면 망의 상태 변화에 제대로 대처하지 못하여 체증을 유발시킬 수 있다.

### (6) 강건성(robustness)

트래픽 제어 알고리즘은 제어 신호의 일부가 손실되더라도 급격하게 망의 성능을 저하시켜서는 안된다. 즉, 전송시의 오류, 데드락(deadlock), 제어 신호의 손실 등이 발생한 경우에도 망의 성능을 잘 유지시켜야 한다. 그리고 손실된 제어 신호 이후에 정상적으로 전송된 제어 신호는 신속하게 망의 상태를 원상 복구시킬 수 있어야 한다.

## 2) 종단간 트래픽 제어기술

엔드 시스템들과, 전체적으로 연결되어 있는 네트워크들이 좋은 성능을 얻으려면 트랜스포트 프로토콜의 설계와 구현이 아주 중요하게 된다. 트랜스포트 프로토콜은 애플리케이션들과 네트워크기능 사이에 인터페이스를 제공하며 애플리케이션들은 원하는 QoS를 요구할 수 있다. TCP와 같은 커넥션중심(connection-oriented)의 트랜스포트 프로토콜은 애플리케이션 데이터의 전체 흐름을 논리적인 스트림들로 나누고 이들 스트림에 자원들을 다르게 할당할 수 있다. 마지막으로 데이터단위들의 전송과 재전송을 위한 트랜스포트 프로토콜의 방식들은 네트워크의 혼잡(congestion)에 심각한 영향을 미친다.

#### (1) 흐름제어와 오류제어

통신 링크, 네트워크, 인터넷네트워크의 성능을 결정하는 기본적인 메커니즘이 흐름제어와 에러제어이다. 흐름 제어 기술은 전송에러로 인한 PDU(Protocol Data Unit)들의 분실이나 혼잡으로 패킷을 폐기하는 것이다.

그런데 흐름과 오류제어 기술의 성능을 모델링하기가 아주 어렵다. 가장 간단한 경우는 포인트-투-포인트 링크로 연결되어 있는 두 장치 사이에 사용되는 링크 제어 프로토콜인데 여기서는 두 장치 사이의 일정한 전파 지연, 일정한 데이터 속도, 확률적인 오류율 그리고 트래픽의 통계적인 특성만 신경을 쓰면 된다.

네트워크나 인터넷네트워크 전체로 흐름과 에러 제어를 다룰 때에는(예, TCP 수준에서) 분석이 아주 복잡하게 되며 가변적인 전파 지연, 가변적인 데이터 속도, 네트워크에 들어가는 다른 트래픽 원천들이 만든 혼잡 영향, 동적인 라우팅 결정의 영향, 여러 논리 커넥션 사이의 상대적인 우선 순위와 그 밖의 요인들도 고려해야 한다.

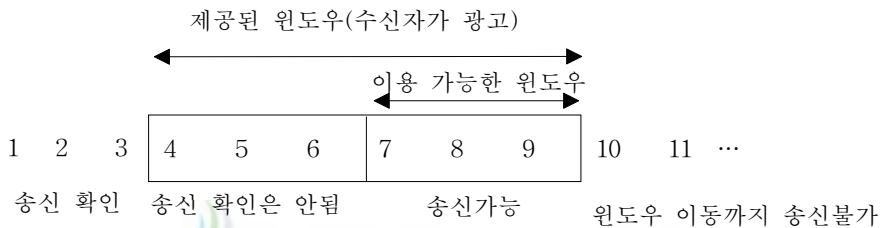
#### (2) 흐름제어

TCP 데이터 세그먼트를 송수신하는 컴퓨터는 CPU와 네트워크 대역폭의 차이 때문에 서로 다른 데이터 속도로 작동할 수 있다. 결국 수신자가 처리할 수 있는 것보다 훨씬 더 빠른 속도로 송신자가 데이터를 보낼 가능성이 많다. TCP는 송신자가 보낸 데이터의 양을 제어하는 흐름 제어 메커니즘을 구현한다. TCP는 흐름 제어를 구현하기 위해 슬라이딩 윈도우를 사용한다.(K.Sivan 1998)



■ 슬라이딩 윈도우 기법

슬라이딩 윈도우 기법은 Fig. 6과 같이 나타낼 수 있다. 이 그림에서 바이트 1에서 11까지 번호를 붙였다. 수신 호스트에 의해 알려지는 윈도우는 광고된 윈도우 (Advertised Window; awnd)라 불리고, 바이트 4에서 9까지를 차지하고 있다. 이것은 수신 호스트는 3까지의 모든 바이트에 Ack를 전송했다는 것과 6바이트의 윈도우 크기를 알려 준다는 것을 의미한다. 윈도우의 크기는 Ack 번호와 관계가 있다. 송신 호스트는 이용 가능한 윈도우를 계산하는데, 이것은 즉시 송신할 수 있는 데이터가 얼마인가를 나타낸다.



제주대학교 중앙도서관  
Fig. 6 Sliding Window Mechanism

시간이 경과하여 수신 호스트가 Ack를 전송하는 것은 슬라이딩 윈도우를 오른쪽으로 이동하게 한다. 윈도우의 양쪽 끝이 이동하므로써 윈도우 크기는 증가하거나 감소된다. 윈도우의 양쪽 끝이 이동하는 것은 닫힘(close), 열림(open), 수축(shrink)으로 표현된다. 먼저 데이터가 보내지고 Ack가 수신될 때 닫힘이 행해지게 된다. 그리고 상대방 수신 호스트가 Ack를 읽고, TCP의 수신 버퍼에 공간이 비어 있을 때에 열림이 행해진다. 마지막 줄어드는 상대방에 대응하기 위해서 TCP도 윈도우를 줄여야 하는데 이때 수축이 행해지나 호스트 요구사항 RFC등에서 사용하지 말 것을 강력하게 주장하고 있다.(W. R. Stevens 1998)

(3) 혼잡제어(Congestion Control)

TCP는 종단간 흐름제어만을 지원하고 간접적인 방법으로 인터넷의 혼잡을 추정한다. 더욱이 네트워크이나 인터넷에서의 지연은 변할 수도 있고 매우 클 수도 있기 때

문에 TCP 개체의 혼잡 정보를 믿을 수만은 없다. 다양한 TCP 개체들은 상호간의 협조관계가 없다. 그래서 네트워크의 상태를 유지하기 위해 상호 협력하지 않는다. 단지 이용할 수 있는 자원을 더 차지하려고 하는 best-effort 서비스를 수행할 뿐이다. 그래서 TCP에서의 혼잡 제어는 매우 어렵고 복잡하지만 지금까지의 많은 연구들에 의해 상당한 성과를 보이고 있다. 혼잡제어 알고리즘들은 크게 재전송 타이머 관리와 윈도우 관리로 나누어 볼 수 있다.(강문철 2001)

#### ■ 재전송 방식

링크제어 프로토콜과 마찬가지로 TCP도 흐름제어 방식을 갖고 있을 뿐만 아니라 에러제어도 하고 있다. 그러나 TCP의 경우 링크제어 프로토콜에 있는 명시적인 부정 ACK나 리지는 없고 긍정 ACK만 있다. 이 긍정 ACK만 이용하면 주어진 타임아웃기간 내에 ACK가 도착하지 않았을 시 재전송을 한다.

#### ■ 저속출발(Slow Start)

송신 호스트는 수신 호스트의 광고윈도우 크기에 따라서 다수의 세그먼트를 네트워크에 전송하면서 연결을 시작한다. 그런데 이것은 2대의 호스트가 같은 LAN상에 있으면 문제가 없지만, 송·수신 호스트 사이에 라우터와 느린 링크가 존재한다면 문제가 발생한다. 즉, 패킷들은 중간 라우터에서 빨리 처리되기 어렵기 때문에 중간 노드에서 기다려야 한다. 그런데 중간 라우터의 저장 공간이 부족하게 되면 버퍼 오버플로우(overflow)가 발생하게 되어 처리율을 상당히 떨어뜨리게 된다. 그래서 slow start라는 알고리즘이 나오게 되었다(V. Jacobson 1988)(W. Stevens 1997). 이것은 네트워크에 전송되는 패킷의 통신 속도를 다른 쪽 종단으로부터 되돌아온 ACK의 통신 속도와 맞추기 위한 것이다.

slow start는 송신 호스트의 TCP에 혼잡 윈도우(cwnd; congestion window)라고 하는 다른 윈도우를 사용하게 한다. cwnd는 새로운 TCP 연결이 설정될 때 하나의 세그먼트로 초기화된다. 그리고 ACK가 수신될 때마다 한 세그먼트씩 증가한다. cwnd는 바이트 단위로 관리되지만 slow start에서 항상 세그먼트 단위로 증가한다. 송신 호스트는 cwnd와 awnd(Advertised Window; awnd)의 최소값까지 전송 할 수 있다. cwnd는 송신 호스트에 의해 행해진 흐름 제어이고, awnd는 수신 호스트에 의해 행해

진 흐름제어이다. cwnd는 인지된 망 혼잡에 대한 송신 호스트의 평가를 기본으로 하며 awnd는 수신 호스트에서 이용할 수 있는 버퍼 공간의 양과 관련이 있다.

송신 호스트는 하나의 세그먼트로 전송을 시작하고 Ack을 기다린다. Ack가 수신되면 cwnd는 “2”로 증가하여 두 개의 세그먼트를 전송한다. 두 개의 세그먼트에 해당하는 Ack가 수신되면 혼잡 윈도우는 “4”로 증가한다. 이렇게 cwnd의 크기는 기하 급수적으로 증가 한다. 수신 호스트에 의해 Ack가 지연될 수 있기 때문에 정확히 기하급수적으로 증가하지는 않는다. 그리고 인터넷의 수용력에 도달하면 중간 라우터는 패킷을 버림으로써 송신기에게 cwnd가 너무 크다는 것을 알린다. 또한 slow start는 연결을 통해서 데이터 흐름을 초기화하는 방법이다.(V. Jacobson 1988)

#### (4) 혼잡 회피(Congestion Avoidance)

데이터는 빠른 LAN에서 느린 WAN으로 전송되거나 다수의 입력 스트림(stream)이 출력 수용량이 적은 라우터에 도달할 때 제대로 전송이 이루어지지 못하고 혼잡이 발생한다. Congestion Avoidance는 패킷 손실을 다루기 위한 방법이다.

이 알고리즘은 패킷 손실이 1%보다 적다고 가정하므로 발생한 패킷 손실이 송·수신 호스트의 사이 네트워크의 어떤 지점에서 발생한 혼잡에 의한 신호가 된다. 패킷 손실은 타임아웃 발생과 중복 Ack(duplicate Ack)의 수신에 의해 알 수 있다.

Congestion Avoidance와 slow start는 서로 다른 목적을 가진 독립적인 알고리즘이다. 그러나, 혼잡이 발생하면 TCP는 망의 패킷 전송률을 감소시키고 다시 slow start를 수행한다. 그리고 Congestion Avoidance와 slow start는 함께 구현되고 각 연결을 관리하기 위해서는 다음 두 개의 변수를 필요로 한다. 혼잡 윈도우(cwnd)와 slow start 한계값(ssthresh; slow start threshold)이 그것이다. (전인재 1999)

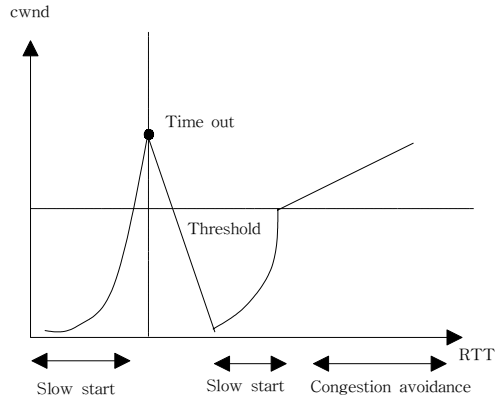


Fig. 7 Cwnd Variation of Slow Start and Congestion Avoidance

(5) 빠른 재전송 및 복구(FRR : Fast Retransmit and Recovery)

수신측에서 순서가 어긋난 세그먼트를 받으면 TCP는 이전에 이미 보냈던 확인 메시지와 같은 번호로 즉시 확인 메시지를 보낸다. Fig. 8처럼 같은 번호를 단 확인 메시지가 중복되어 나타난다는 것은 수신측에서 받은 데이터 순서에 이상이 생겼거나 데이터 분실이 일어난 경우라고 송신측에서 생각한다.

빠른 재전송 및 복구(FRR) 알고리즘은 같은 번호가 붙은 확인 메시지 3개를 연속해서 받으면 그 번호로 시작하는 패킷이 분실되었다고 생각하고 해당 패킷만 즉시 보낸다. 이렇게 되면 분실된 패킷이 빠르게 복구되고 망이나 다른 패킷에 커다란 영향을 주지 않는다. 이러한 FRR이 성공하게 된 배경에는 망 속도가 빨라지고 에러가 일어날 확률이 작아져 문제가 된 단 하나의 패킷만 재 전송함으로써 복구가 빨라질 수 있다는 생각이다. 하지만 여러 패킷이 분실된 경우라면 이 알고리즘은 별 효과를 보지 못할 것이다.(장혁수와 주우석 2000)

3) RED(Random Early Detection)

RED는 평균 큐 길이에 따라 망의 체증 정도를 결정하여 이에 따라 패킷을 마크 혹은 폐기시키는 확률을 결정한다. 이에 따라 송신 호스트는 해당 패킷의 처리 여부에 따라 전송속도를 조절한다.

RED는 패킷을 마크하는 척도로 매 패킷의 도착할 때마다  $Q_{avg}$ 를 구하고, 이것을

미리 정해 놓은 파라미터인 최소 큐 한계값( $TH_{\min}$ )과 최대 큐 한계값( $TH_{\max}$ )과 비교한다.

$Q_{\text{avg}}$ 가  $TH_{\min}$  보다 작을 때에는 모든 패킷은 정상적으로 처리되고  $TH_{\max}$  보다 클 때에는 마크되거나 폐기된다. 그리고  $Q_{\text{avg}}$ 가  $TH_{\min}$  와  $TH_{\max}$  사이에 있을 때는 도착하는 패킷은 폐기확률  $P_a$ 의 적용을 받는다. 이 확률은 0에서부터 RED내에 정의된 상수인 최대확률( $P_{\max}$ )까지의 범위내에 존재한다.



### III. 예측 기반 RED 알고리즘

본 논문에서는 버퍼에서의 평균 큐 길이를 이용하여 체증 제어를 수행한다. 일반적으로 패킷이 도착할 때마다 PRED 알고리즘은 다음과 같은 절차를 수행한다.

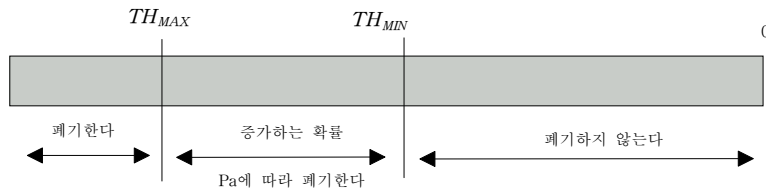


Fig. 9 PRED Buffer

#### 1. 예측기반 RED



이 알고리즘은 FIFO 출력큐에 새로운 패킷이 도착할 때마다 큐가 비어있는지의 여부를 파악하여 큐가 비어있지 않으면 평균 큐 길이  $Q_{avg}$ 는 EWMA(Exponential Weighted Moving Average)를 통한 low-pass filter를 사용하여 다음과 같이 계산한다.

$$Q_{avg} \leftarrow (1 - w_q)Q_{avg} + w_q Q_{size} \quad (2)$$

(  $Q_{avg}$  : average queue size     $w_q$  : queue weight

$Q_{size}$  : current queue size)

그 다음 큐 길이를 최소값과 같이 비교하여 최소값보다 크면 예측 큐 평균( $D_{avg}$ )을 구하는데, 여기에서  $D_{avg}$ 는  $Q_{size}$ 와  $Q_{avg}$ 사이의 편차를 이용한다.

$$D_{avg} \leftarrow (1-w_d)D_{avg} + w_d(Q_{size} - Q_{avg}) \quad (3)$$

(  $D_{avg}$  : predictive average queue size

$w_d$  : predictive queue weight )

이 예측 큐 길이( $Q_{PL}$ )를 두 한계치 Fig. 9과 같이 비교한다.  $Q_{PL}$ 가 하한치  $TH_{min}$ 보다 작으면 체증이 최소이거나 없는 것으로 가정하여 이 패킷을 큐에 넣는다.  $Q_{PL}$ 가 상한치  $TH_{max}$ 보다 크면 체증이 심하다고 가정하며 이 패킷을 버린다.  $Q_{PL}$ 의 정확한 값에 따라  $Q_{PL}$ 를 상한선에 이르게 증가시키는 확률  $P_a$ 를 계산한다. 큐가 이 영역에 있으면 이 패킷을 확률  $P_a$ 로 버리며 확률(1- $P_a$ )로 큐에 들어가게 한다.

기본적으로 알고리즘의 첫 부분(큐 크기 계산)은 허용할 수 있는 폭주정도를 결정하며 알고리즘의 패킷 폐기결정은 현 수준의 체증이 있을 때 폐기되는 패킷들의 빈도를 결정한다.



1) 평균 큐길이와 예측 큐 길이 계산

앞 큐 길이들의 지수비중(exponentially weighted) 평균을 이용하여 식 2에서와 같이 평균 큐 길이를 계산하며, Qsize와  $TH_{min}$ 를 비교하여 Qsize가  $TH_{min}$ 보다 크면 식3을 이용한다. Qsize가 Qavg보다 작으면 식4로 구하며, 아니면 식3을 이용한다.

$$D_{avg} \leftarrow Q_{size} - Q_{avg} \quad (4)$$

Idle 기간동안 라우터가 전송할 수 있었지만 전송하지 않은 작은 패킷들의 숫자 m을 계산하여 큐가 비어있는 기간을 고려한다.

$$m \leftarrow f(\text{time} - q_{time})$$

$$Q_{avg} \leftarrow (1-w_q)^m Q_{avg} \quad (5)$$

( time : current time       $q_{time}$  : start of queue idle time,

for(t) : a linear function of time t )

라우터에서의 임시적인 체증을 없애기 위하여 평균 큐길이를 사용하며, 가중치  $w_q$  는 실제 큐 크기의 변화에 따라  $Q_{avg}$ 가 얼마나 빨리 변하는가를 결정한다. 아주 작은 값인 0.002(Floyd, S와 Fall, K 1997)를 권하고 있다. 결과적으로  $Q_{avg}$ 는 실제 큐 크기의 변화보다 상당히 뒤에 일어나게 한다. 이렇게 작은 비중을 사용하면 알고리즘이 짧은 폭주의 체증에 반응하지 않게 된다.

예측 큐 길이는 평균 큐 길이와 예측 큐 평균 길이를 더한 값으로 한다. 즉,

$$Q_{PL} \leftarrow Q_{avg} + D_{avg} \quad (6)$$

## 2) 패킷 폐기의 결정

$Q_{PL}$ 이  $TH_{min}$ 보다 작으면 들어오는 패킷을 큐에 넣으며,  $Q_{PL}$ 가  $TH_{max}$ 보다 크면 들어오는 패킷을 자동적으로 폐기한다. 임계영역은  $Q_{PL}$ 의 값이 두 한계치 사이에 있을 때이다. 이 영역에서는 PRED가 들어오는 한 패킷에 두 요인에 의존하는 폐기 확률을 할당한다.

- $Q_{PL}$ 가  $TH_{max}$ 에 가까워질수록 폐기확률이 더 높아진다.
- $Q_{PL}$ 가 임계영역에 있으면 연속적으로 패킷들을 폐기 않도록 하기 위해 count 값에 따라 count값이 클수록 폐기할 확률이 더 높게 된다.

$$P_b = P_{max} \times \frac{Q_{PL} - TH_{max}}{TH_{max} - TH_{min}} \quad (7)$$

먼저 식7 처럼  $P_b$ 를 계산하는데 이는  $Q_{PL} = TH_{min}$ 에서의 0부터 시작하여  $Q_{PL} = TH_{max}$ 에서의 어떤 최대치  $P_{max}$ 까지 선형으로 증가하는 값이다.

PRED 알고리즘의 경우 폭주 원천이 불이익을 당하지 않게 폐기를 할 때 비교적



공평하게 간격을 두려고 한다. 따라서  $P_b$ 를 직접 쓰지 않고 폐기를 결정할 때 사용하는 확률인 두 번째 확률  $P_a$ 를 계산하는데 사용하고 있다.

$$P_a = \frac{F \times P_{\max}}{1 - \text{count} \times F \times P_{\max}} \quad (8)$$

$$\left( \text{단, } F = \frac{(Avg + D_{Avg}) - TH_{\min}}{TH_{\max} - TH_{\min}} \right)$$

$P_{\max} = 0.02$ 를 권하고 있다(Floyd, S와 Fall, K. 1997)(Floyd, S와 Fall, K. 1997). 체증의 수준이 높으면 PRED가 RED보다 훨씬 높은 처리율을 제공한다.

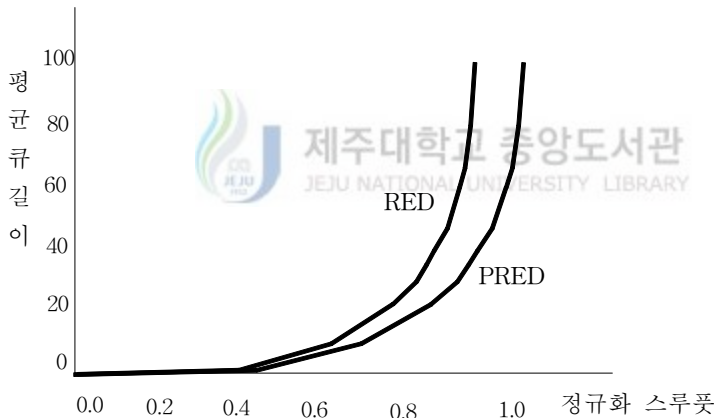


Fig. 11 Performance Comparing of RED and PRED

PRED는 종단간 체증 제어 알고리즘과 연계하여 큐 길이를 예측하여 사전에 패킷을 폐기하는 알고리즘으로서, 가끔 발생하는 버스트 트래픽은 허용하며 TCP의 혼잡 회피/제어 기능이 동작된다.

### 3) 큐 변화량과 평균 큐 길이

기존 RED 알고리즘이 패킷을 폐기하는 방법은  $Q_{Avg}$ 를 식2과 식9를 이용하여 확률

$P_a$ 를 적용하여 패킷을 처리한다.

$$P_a = \frac{P_b}{1 - count \times P_b} \quad (9)$$

$$\left( F = \frac{avg - TH_{min}}{TH_{max} - TH_{min}} \right),$$

$$P_b = F \times P_{max} \quad (0 \leq F \leq 1)$$

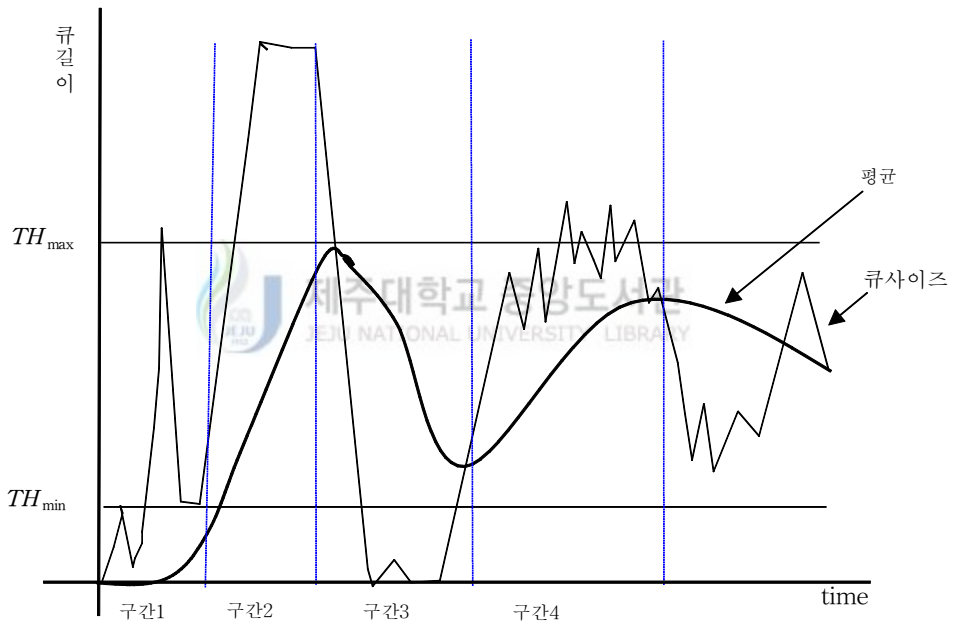


Fig. 12 RED

RED를 사용한 환경에서는 위 Fig. 12에서 같이 살펴 볼 수 있다.

우선 Fig. 12에서 구간1, 구간2, 구간3에서 문제점들을 발견할 수 있는데 아래와 같다.

- 구간 1에서 실제 큐 길이는 최대 Q길이에 도달하고 있어 패킷 폐기를 해서 제어가 필요하나  $Q_{Avg}$  값보다 작은 상태에서 폐기확률을 적용할 수가 없다.
- 구간 2는 체증이 발생하여 패킷이 폐기되고 있으나  $Q_{Avg}$  값이  $TH_{max}$  밑에

있어 체증을 인식하지 못한 상태이다.

- 구간 3에서는 전역동기화(Global Synchronization)가 발생하여 현재  $Q$ 가 비어 있음에도 불구하고  $Q_{Avg}$ 가  $TH_{max}$  가까이에 있어 도착하는 패킷에 대해 확률  $P_a$ 가 적용되어 폐기 처리한다.
- 순간적인 버스트 한 데이터에 대해 민감하게 반응하는 것을 억제해야 한다. 체증이 발생시킬 수 있는 적당한 버스트 한 데이터에 대해서는 폐기확률을 적용하여 패킷을 폐기하여야 체증을 막을 수 있다.

이를 보완하기 위하여 본 논문에서는 큐 변화량을 적용한 가중평균 길이와 큐차이를 적용하여 폐기확률을 적용하고자 한다.

- 구간1, 2는 급격하게 변하는 큐의 변화를  $Q_{avg}$ 가 인식할 수 있으려면  $W_q$ 를 크게 해야 하나 이 방법은 전체 효율에 지장을 초래할 수 있으므로 새로운 변수를 식 2과 유사하게 도입하여 식3과 식8에서와 같이 폐기확률에 적용하여 체증을 방지할 수 있도록 하였다.
- 구간3은 현재의 큐가  $TH_{min}$  보다 작을 때는 들어오는 패킷을 폐기하지 않아야 한다.  $Q_{Avg}$ 가  $TH_{max}$ 에 가깝게 있어 높은 폐기 확률을 적용하고 있다. 이를 해결하기 위해 식10와 같은 방법을 적용하여 패킷 폐기 여부를 결정한다.

$$D_{avg} = -Q_{Avg} \quad (10)$$

$$\text{즉, } Q_{Avg} + D_{Avg} = Q_{Avg} + (Q - Q_{Avg}) \quad (11)$$

- 체증을 야기할 만큼 버스트 한 데이터가 지속되면 (순간적인 버스트가 아님) 이를 인식하여 식3을 이용하여 폐기확률을 높여 적용한다.

RED는 새로운 패킷이 라우터에 들어왔을 때마다 평균 큐 길이를 계산하게 된다.

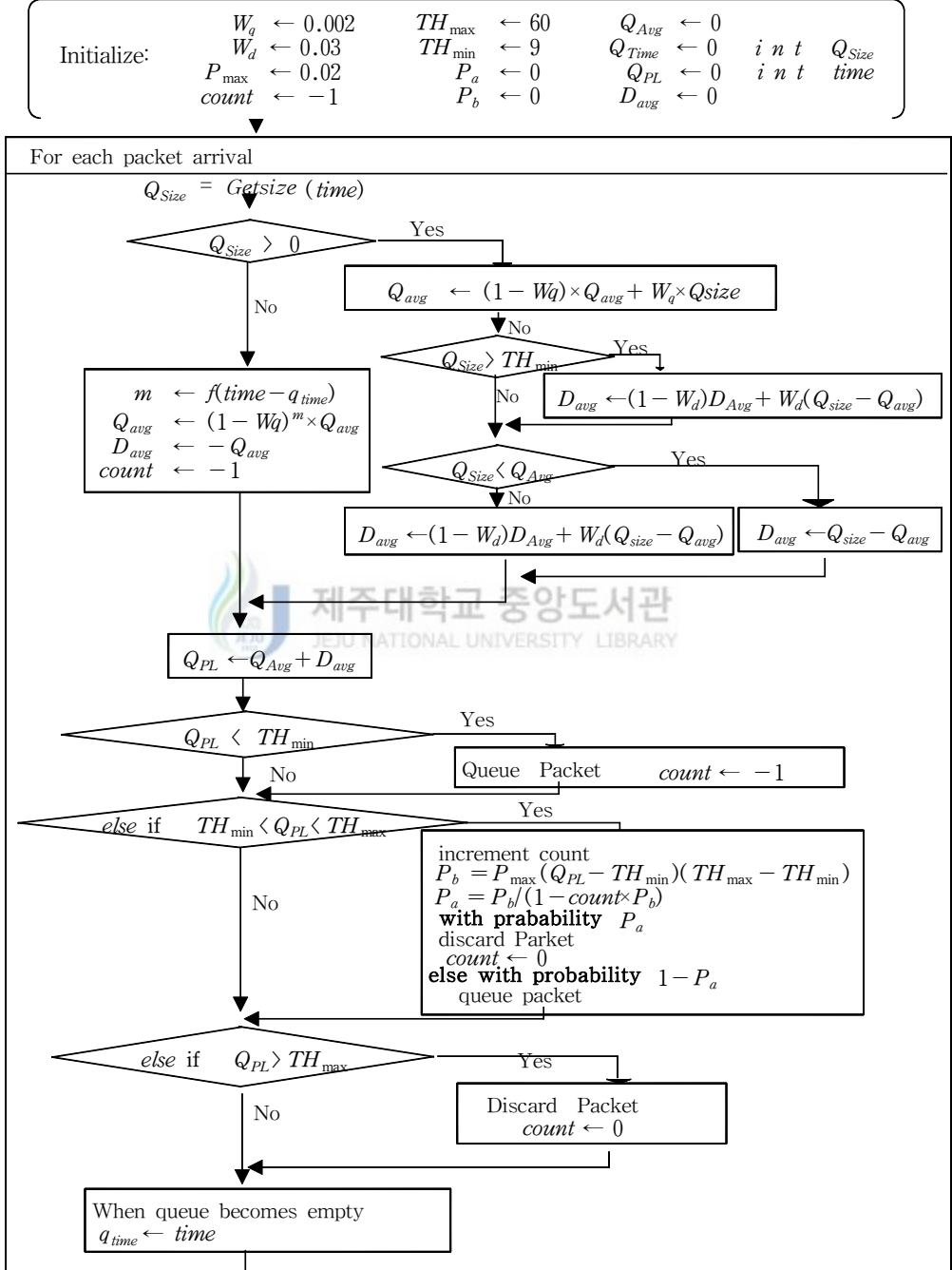
현재 큐 길이에 대하여 가중치를 두어 평균 큐 길이가 계산되었다(S. Floyd 와 V. Jacobson 1993). 여기에는 다음에 오는 패킷에 대해서 한 개의 버스트 한 패킷에 대해서는 폐기확률을 적용하지 않지만 어느 정도 연속적인 버스트 한 패킷에 대해서는 체증을 야기할 가능성이 있으므로 높은 폐기 확률을 적용시켜야 한다.

즉 또한 라우터에서 패킷을 폐기하게되면 큐 길이에 영향을 주게 되는데 지금 폐기했다 하더라도 당장 큐의 길이가 줄어들지 않고, 어느 정도 시간이 지나야 큐 길이가 줄어들게 된다. 따라서 예측 평균 큐 길이를 적용하면 이러한 단점을 보완하여 폐기 확률이 평균 큐 길이에 적응할 수 있도록 하고자 한다.

본 논문의 알고리즘은 예측기반 RED(predictive RED : PRED)라 하였으며, 제안되는 알고리즘은 다음과 같다.



TABLE 1. PRED 알고리즘



Saved Variables :

$Q_{PL}$  : predictive queue size

$D_{avg}$  : Average of predictive queue size

$q_{time}$  : start of queue idle time

$count$  : packets since last discarded packet

$Q_{Size}$  : current queue size

$Q_{avg}$  : Average of queue size

Fixed Parameters :

$W_q$  : queue weight

$W_d$  : Predictive queue weight

$P_{max}$  : Maximum value for  $P_b$

$TH_{min}$  : Minimum threshold for queue

$TH_{max}$  : Maximum threshold for queue

Other :

$P_a$  : current packet-marking probability

$P_b$  : temporary probability used in calculation

time : Current Time

for(t) : a linear function of time t

Fig. 13 Predict RED Algorithm

## IV. 시뮬레이션 및 성능분석 결과

### 1. 시뮬레이션 모델

본 논문에서 사용한 Network 환경은 Fig. 14과 같이 구성하였으며 다음과 같은 전제조건을 두었으며, 일반적인 사항은 고려하지 않았다.

- ① 모든 링크의 대역폭은 80Mbps
- ② 링크와 gateway 사이의 delay time은 1, 2, 3, 5, 4, 4 ... msec으로 반복적용
- ③ Sink 노드와 Router 사이의 delay time : 2msec

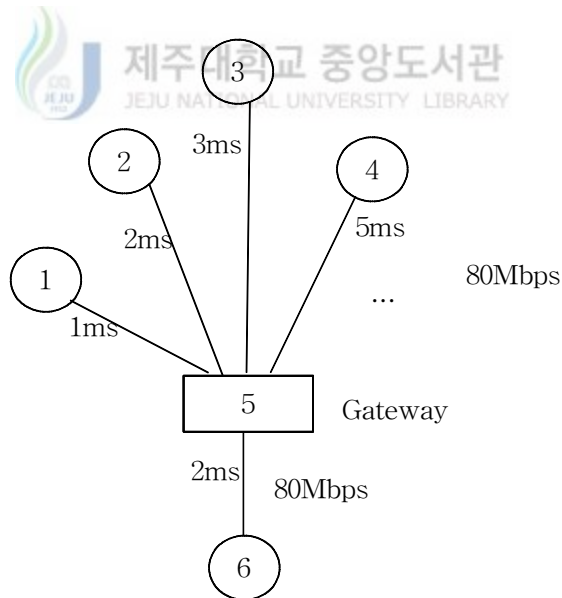


Fig. 14 Simulation network

시뮬레이션은 링크 수를 늘여가면서 Gateway 형태를 RED, PRED의 방식을 비교

하였으며, 시뮬레이션 시간은 10초, Gateway 큐 길이는 100패킷, 1패킷은 1000byte로 하였고 Ack packet은 40byte로 하였다.

체증인지 방법은 gateway가 패킷을 폐기하였다는 것을 링크 송신노드가 10msec 지나서 알 수 있도록 하였다.

RED의 매개변수는 기존 RED알고리즘에 사용했던 파라미터는 RED인 경우  $P_{max}$ 는 0.02,  $Wq$ 는 0.002,  $TH_{max}$ 는 60,  $TH_{min}$ 는 9, 패킷 크기는 1000, 윈도우 크기는 1024를 사용하였다(S. Floyd 와 V. Jacobson 1993)(유영석과 홍석원 1997).

최소 한계값( $TH_{min}$ )과 최대 한계값( $TH_{max}$ )은 RED 파라미터에 중요한 값이지만 수학적으로나 시뮬레이션만으로 정확하게 정할 수 있는 파라미터는 아니다.

$TH_{min}$ 는 가급적 크게 주는 것이 어느 패킷이나 마크없이 그대로 허용하기 때문에 망이 최대의 효율을 발휘할 수 있게 해준다. 또한  $TH_{max}$  값과  $TH_{min}$  값의 차이는 크게 해주는 것이 좋다. 이 폭이 적은 경우는 전역 동기화(global synchronization)가 발생하기 쉽기 때문에 시뮬레이션을 통해 볼 때  $TH_{max}$ 는 최대 큐 길이의 60~70% 정도 해주면 되고,  $TH_{min}$ 은  $TH_{max}$ 의 20%가 바람직하다(유영석과 홍석원 1997).

본 논문에서 사용된 파라미터 중 예측 큐 평균 길이를 구하기 위하여 사용된 예측가중치는 평균 큐 길이를 큐 변화량에 적용하기 위하여 0과 1사이의 값으로 정하였으며 시뮬레이션을 통하여 적절한 값으로 0.03를 정하였다. 이 값은 일반적인 것은 아니다.

수정된 RED 알고리즘과 기존 RED 알고리즘의 성능을 비교하기 위해 Microsoft Visual C++ 6.0을 사용하여 시뮬레이션을 실행하였다.

## 2. 성능측정 방법

시뮬레이션 성능측정은 링크의 효율성, 패킷 손실율, 평균 큐 길이와 예측 평균 큐 길이에서 폐기 확률을 이용하여 성능을 측정하였으며, 계산방법은 다음과 같다.



1) Link utilization

$$Link\ utilization = \frac{total\ packet\ 전송한\ 패킷}{total\ 시뮬레이션\ 시간} \quad (11)$$

2) Packet Loss Rate

$$Packet\ Loss\ Rate = \sum_{i=1}^n \frac{(drop\ packet\ count)}{n} \quad (12)$$

drop packet count = congestion packet + marked packet with packet Pa  
n : simulation 횟수

3) 폐기 확률의 비교

- 평균 큐 길이의 이용한 폐기 확률

식2, 식8

- 예측 평균 큐 길이의 이용한 폐기 확률

식3, 식7, 식13



제주대학교 중앙도서관  
JEJU NATIONAL UNIVERSITY LIBRARY

### 3. 시뮬레이션 결과

시뮬레이션 결과는 Fig. 15에서 Fig. 18에 걸쳐 나타내었다. 그림에서 시뮬레이션은 커넥션 5개부터 20를 갖고 나타내었고 Fig.15 와 Fig.16은 커넥션이 5개 일 때의 RED 와 PRED 결과를 표현하였다.

패킷 손실율도 random하게 처리하는 특성으로 인하여 측정된 값을 사용하였으며, 시뮬레이션의 정확도를 위하여 시뮬레이션 시간을 10sec로 설정하였다.

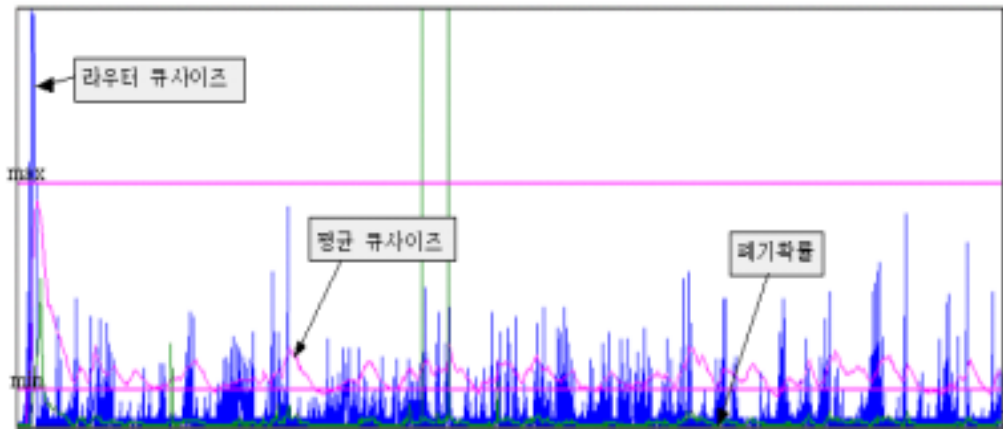


Fig. 15 Performance of RED

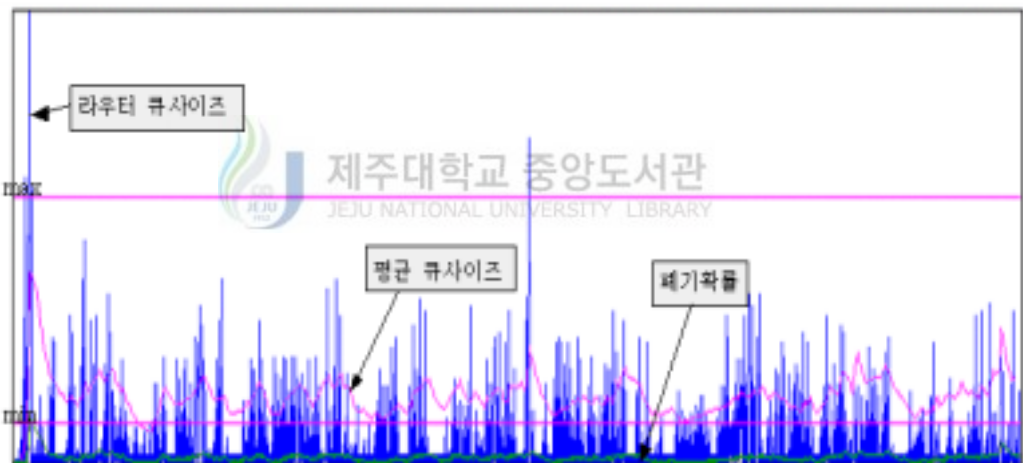


Fig. 16 Performance of PRED

전송률은 RED는 49,128 PRED 55,627 이며, 손실률(혼잡)은 RED는 232이고, PRED 200 이다. 결과적으로 개선된 알고리즘이 훨씬 성능이 좋은 것으로 나타나고 있다.

아래 표는 커넥션 5개부터 20개까지의 시뮬레이션 상황을 표로 나타내었다. 전송률에 있어서는 PRED가 RED보다 훨씬 전송률이 높은 차이를 볼 수 있으며 손실률에서는 PRED가 RED보다 손실이 적은 것을 볼 수 있다.

Table 2. Simulation result

커넥션수	전송률		손실률	
	RED	PRED	RED	PRED
5	49,128	55,627	232	200
6	54,569	61,082	363	333
7	57,069	66,803	420	367
8	64,123	70,428	446	391
9	68,044	73,987	494	485
10	69,876	78,100	605	553
11	72,185	82,658	697	609
12	76,904	85,462	687	682
13	80,961	85,205	711	681
14	84,010	87,664	895	759
15	86,387	89,947	943	868
16	87,799	91,119	995	940
17	90,097	92,002	1,151	1,066
18	91,742	93,340	1,180	1,162
19	93,264	94,867	1,226	1,165
20	94,240	95,759	1,370	1,305

아래 Fig. 17 Fig. 18은 위 Table. 2에서 전송률과 손실률 기준으로 RED와 PRED의 상관관계에 대해서 살펴보았다.

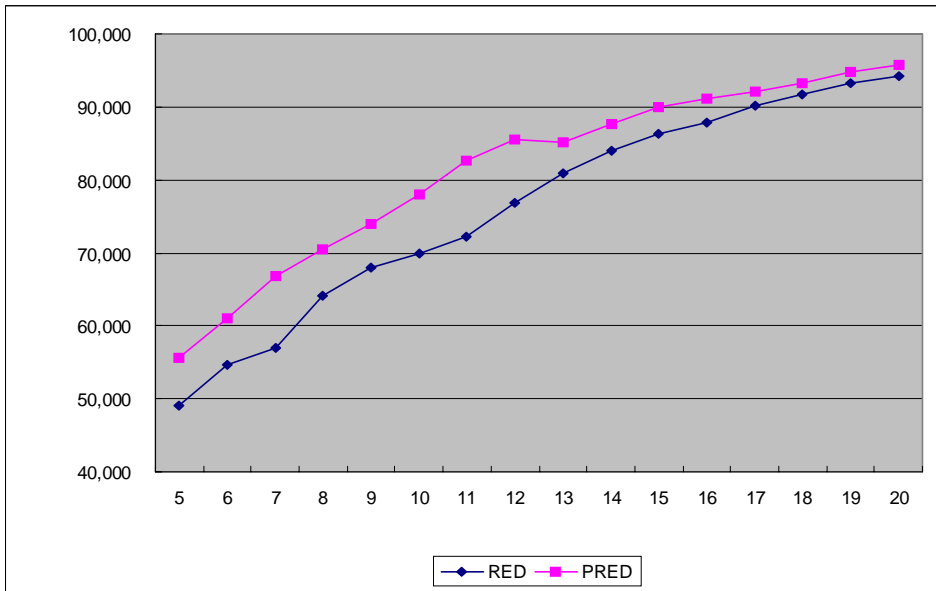


Fig. 17 Comparison of the Performance

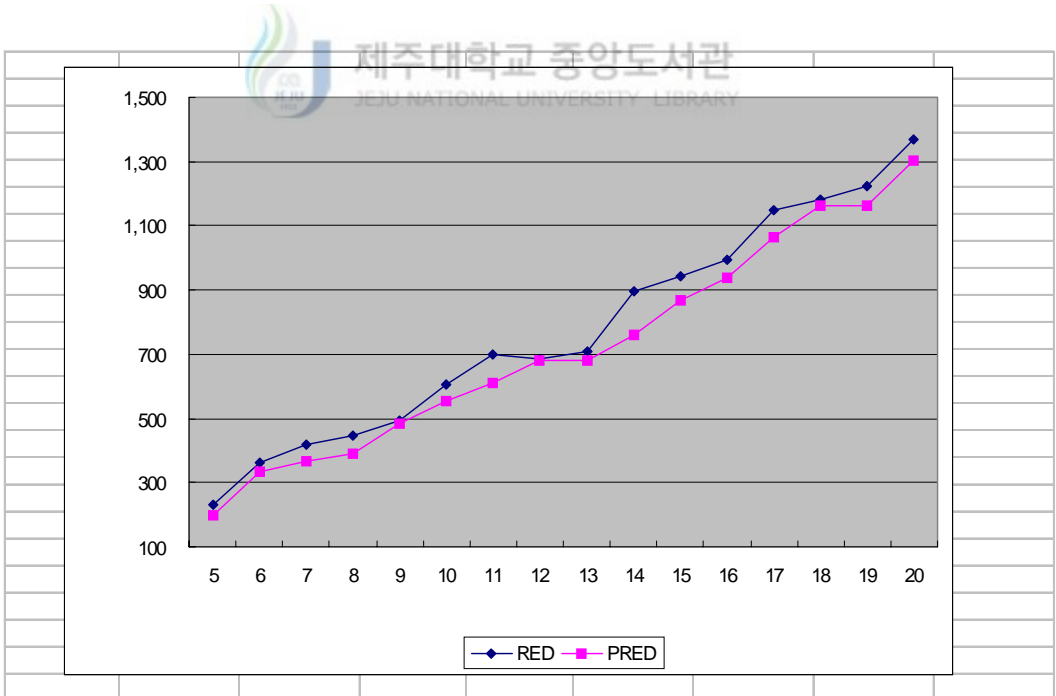


Fig. 18 The number of dropped Packets

아래 Table. 3에서는 wd값을 0.01에서 0.05까지 변경시키면서 시뮬레이션 한 결과를 나타내었고, Fig. 19와 Fig. 20은 상관관계를 그래프로 표현하였다.

Table 3. Simulation Result from 0.01 to 0.05

커넥션 수	wd=0.01		wd=0.02		wd=0.03		wd=0.04		wd=0.05	
	전송	혼잡	전송	혼잡	전송	혼잡	전송	혼잡	전송	혼잡
5	55448	318	56306	318	55627	200	56278	314	58866	316
10	78734	832	77300	806	78100	553	74576	822	77882	782
15	88508	1144	88296	1234	89947	868	87734	1178	88552	1040
20	94822	1506	95242	1556	95759	1305	95428	1432	95020	1496

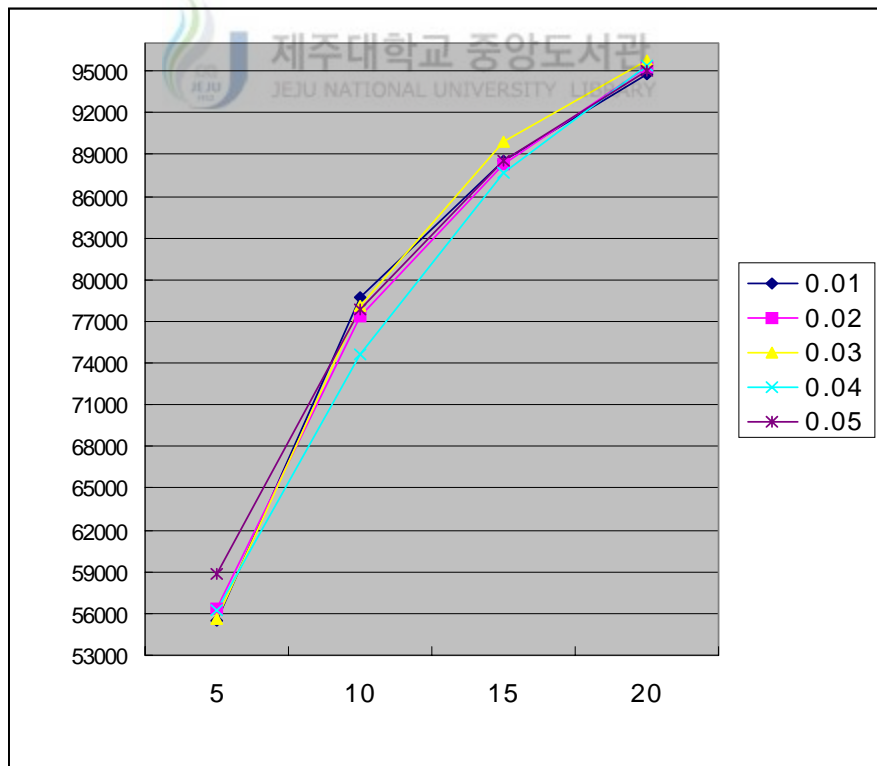


Fig. 19 Comparison of the Transmission

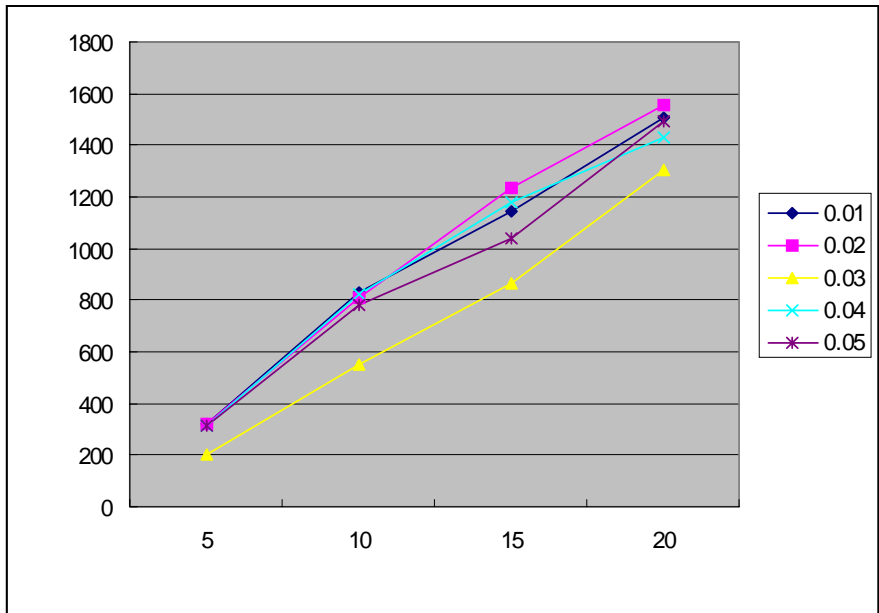


Fig. 20 The number of dropped Packets

아래 Table. 4는 wd를 0.001에서 0.003까지의 시뮬레이션한 결과를 표시하였고, Fig. 21과 Fig. 22 역시 상관관계를 표시하였다.

Table 4. Simulation Result from 0.001 to 0.003

커넥션 수	wd=0.001		wd=0.002		wd=0.003		wd=0.03	
	전송수	혼잡	전송수	혼잡	전송수	혼잡	전송	혼잡
5	55318	310	54518	320	53376	326	55627	200
10	76948	810	75030	828	76474	826	78100	553
15	87896	1190	89246	1194	88122	1198	89947	868
20	95014	1508	94872	1524	94302	1610	95759	1305

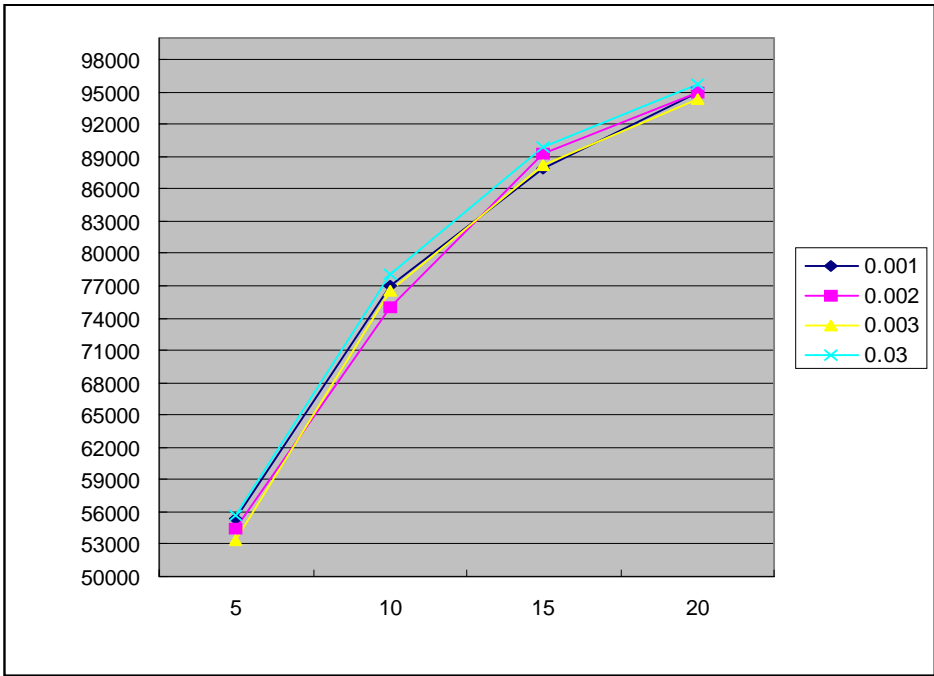


Fig. 21 Comparison of the Transmission

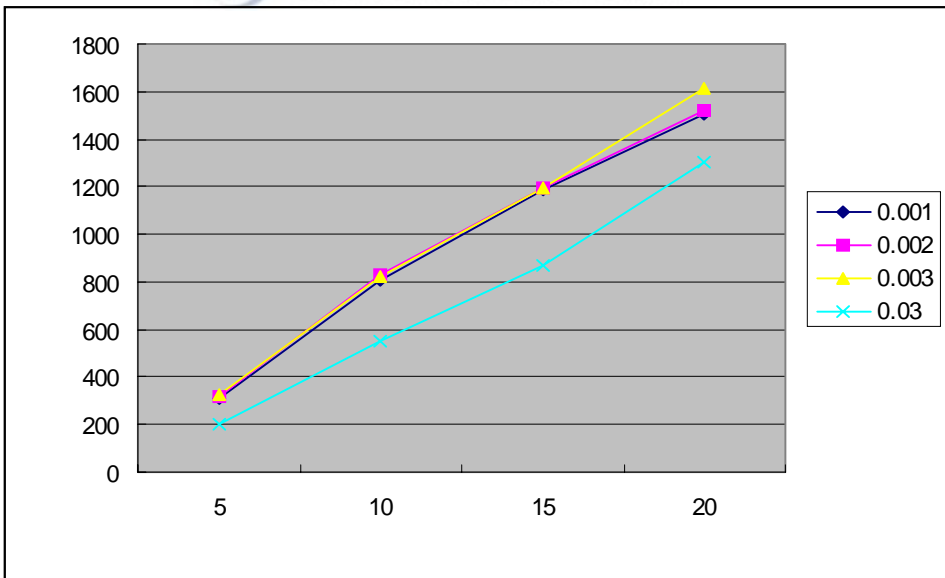


Fig. 22 The number of dropped Packets

위의 결과들로 살펴보았듯이 Wd가 0.01, 0.02, 0.04, 0.05일 때 0.6%, 0.72%, 1.72%, -0.28% 전송량을 볼 수 있으며 손실률은 23%, 25.2%, 21.9%, 19.5%의 감소를 가져왔으며, Wd가 0.001, 0.002, 0.003일 때 1.35%, 1.81%, 2.29%의 전송량의 증가와 23.4%, 손실률은 24.3%, 26.1%의 감소를 가져왔다.

이와 같이 전송률과 손실률이 다른 파라미터보다 성능이 좋음을 알아 보았다.





## V. 결론

네트워크에서 원활한 데이터 전송은 중요하다. 또한 처리률을 극대화하고, 지연을 최소화하며 각 연결들 간의 트래픽 제어와 같은 여러 가지 방법들이 끊임없이 연구되고 있다. 인터넷의 경우도 마찬가지다. 최근에는 응용계층의 다양화로 인해 데이터의 유형이 여러 가지로 나뉘어져 있으며 TCP 상에서의 서비스는 best-effort 주로 지원한다.

TCP는 종단간 혼잡제어를 수행하며, 종단 노드가 패킷 손실이나 응답 시간의 변동에 의해 간접적으로 네트워크의 부하를 추정하고 그것에 따라서 혼잡 제어를 한다. 하지만 종단 노드만으로 발생한 혼잡을 해소하는 것은 쉬운 일이 아니며 종단 노드에서의 혼잡 제어 방안들이 중요시되고 있다.

중간 노드에서 혼잡 제어를 하는 방법은 능동적인 큐 관리 기법의 대표적인 RED 알고리즘이 있다. 이 알고리즘은 라우터에서 평균 큐 사이즈를 낮게 유지함으로써 패킷 손실과 큐잉 지연을 최소화하고, 체증 발생시 확률적으로 패킷을 폐기시킴으로서 전역동기화(global synchronization)를 막을 수 있다. 그리고 평균 큐 사이즈를 사용하여 일시적으로 폭주하는 트래픽에 의한 영향을 줄일 수 있어서 높은 링크 이용률을 갖게 한다는 장점이 있다. 하지만 평균 큐 길이가 어느 정도 되면 확률을 구하여 패킷을 폐기하기 시작한다. 이때 TCP 연결의 상태에 상관없이 혼잡을 경험한 TCP 연결의 패킷이 그렇지 않은 것보다 폐기될 확률이 상대적으로 커지게 된다.

본 논문에서는 RED 알고리즘의 전송률과 손실을 문제를 해결하기 위해 기존 RED의 큐 평균과 예상 큐 평균을 구하여 서로 더함으로써 큐 길이를 예상할 수 있도록 하였다. 그 결과 많은 양의 전송량과 손실률에 있어서 개선됨을 Table. 2 에서 살펴보았다. 전송률인 경우 RED에 비해서 PRED가 커넥션이 5~20 일 때 13.2%, 11.9%, 17.1%, 9.8%, 8.7%, 11.8%, 14.5%, 11.1%, 5.2%, 4.3%, 4.1%, 3.8%, 2.1%, 1.7%, 1.7%, 1.6% 이고 평균적으로 7.7%의 전송률의 상승을 보였으며, 손실률인 경우 16.0%, 9.0%, 14.4%, 14.1%, 1.9%, 9.4%, 14.4%, 0.7%, 4.4%, 17.9%, 8.6%, 5.9%, 8.0%, 1.5%, 5.2%, 5.0% 이며 평균적으로 전송률은 7.7% 증가했고, 손실률은 8.5% 감소했다.

앞으로 연구과제는 RED가 효율적이면서도 간단하다는 장점이 있다고 해도, 개선의 여지가 있다는 것을 본 연구를 통해서 입증해 보였다. 그리고, 측정 파라미터를 가변적으로 변화를 시키는 방안과,  $Q_{PL}$  설정에 대해서도 보다 더 연구가 필요하고 다양한 방식으로 접근할 필요가 있다.



## 참고문헌

A.Mankin "Random Drop Congestion Control. Proceedings of ACM Sigcomm '90 Conference", pages 1-7, 1990

Croll, Alistair/ Packman, Eric "Managing Bandwidth" Prentice Hall 1999. 7월

Demers. A. Keshav. S, Shenker.S. "Analysis and Simulation of a Fair Queueing Algorithm, Internetworking" Research and Experience, vol.1. p.3-26,1990

E. Hashem "Analysis of Random Drop for Gateway Congestion Control, Report LCS Tr-465, Laboratory for Computer Science, MIT, Cambridge, MA, 1989



이지형, 이상연, 정충교, "RED알고리즘의 공정성 개선 방안", 통신학회 하계종합학술 발표회 논문집 p.970-973, 1999년 7월

Floyd, S., and Fall, K. "Router Mechanisms to Support End-to-End Congestion Control." Proceedings, Sigcomm'97, 1997.

Forouzan, Behrouz A./ Fegan, Sophia Chung "TCP/IP", McGraw Hill ,1999

Jain, R., S. Kalyanaraman, R. Goyal, S. Fahmy and R. Viswanathan, 1996, Aug., ERICA Switch Algorithm : A Complete Description, ATM Forum/96-1172.

Jain, R., S. Fahmy, S. Kalyanaraman and R. Goyal, 1997b, The Eritha Switch Algorithm for ABR Traffic Management in ATM Networks, Part II :

Requirements and Performance Evaluation, IEEE/ACM Transaction on Networking

장혁수, 주우석 “인터넷 통신 프로토콜 및 응용” 도서출판 2000. 12월

전인재, “네트워킹의 혼잡 정보를 이용한 TCP 연결간의 공정성 개선 방안”, 강원대학교 논문집, 1999.12.

강문칠 편역, “최신 네트워크 설계”, 세명서관, 2001년 1월

표병훈, 이기영 “큐 변화량을 적용한 RED알고리즘 개선”, 통신학회 하계종합학술 발표회 논문집 p.1035-1038, 2001년 7월

RFC 1349, Type of Service in the Internet Protocol Suite, 7월 1992.

RFC 791, Internet Protocol, 9월 1981.



S. Floyd and V. Jacobson “Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networking

유영석, 홍석원 “체증 제어를 위한 Random Early Detection(RED) 알고리즘의 파라미터 분석”, 통신학회 추계발표 논문집 p.41-44, 1997년 11월.

V. Jacobson “Congestion Avoidance and Control”, Proceedings of SIGCOMM '88, pp314-329, August 1988

Vegesna, Srinivas, “IP Quality OF Service”, Cisco Systems, 2000년

W.Feng, D.Kandlur, D.Saha, K.Shin “Techniques for Eliminating Packet Loss in Congested TCP/IP Networks, UM-CSE-TR-349-97, November. 1997

W.Richard Stevens "TCP/IP Illustrated, Volume 1" ,1998

W.Stevens, "TCP Slow start, Congestion Avoidance, Fast Retransmit, and Fast Recovery algorithm", Internet RFC 2001, January 1997

Zhang.L. "A New Architecture for Packet Switching Network Protocols, MIT LCS TR-455, Laboratory for Computer Science, Massachusetts Institute of Technology, August, 1989.



## 감사의 글

엣그제 입학한 것 같은데 벌써 2년이란 세월이 흘렀습니다. 돌이켜보면 참 파란만장했던 그 시간들이 이제는 짧음을 아쉬워하게 합니다. 누구나 힘들었던 시절이 가장 기억에 남고, 화제 거리가 되듯 늦게 학문의 길로 들어섰던 저가 무사히 이 길을 갈 수 있었던 것은 주님의 은총과 언제나 옆에서 지도와 충고를 해주셨던 안기중 교수님이 계셨기 때문입니다. 교수님께 감사를 드립니다.

본 논문이 완성되기까지 지도편달을 아끼지 않으신 김장형 교수님, 곽호영 교수님, 변상용 교수님, 이상준 교수님, 송왕철 교수님께도 감사드립니다.

또한 이 논문을 쓰는 동안 같이 밤샘을 해주시고 자료와 대학원 생활에 도움을 주신 박사과정 김동춘 선배님, 김대영 선배님, 그리고 같은 연구실의 정태백씨, 현병철씨, 강영심 그리고 세근, 창범, 혜정, 은범에게도 감사의 글을 전합니다.

그리고 언제나 같은 배를 타서 희노애락을 같이 하고 서로에게 의지하며 큰 힘이 되었던 대학원 동기인 강명화 선생님, 이종현, 양영수에게도 이 지면을 빌어서 고마움을 전합니다.

끝으로 딸의 뒷바라지를 아끼지 않으셨던 어머니님, 그리고 가족들, 이제 1월에 웨딩마치를 올릴, 늘 불평을 받아주고 위로를 해주었던 광준씨, 시부모님, 바쁘다는 이유로 모임에 결석대장이었지만 언제나 기특하게 여겨주는 친구들 모두에게 이 논문을 드립니다.