TCP 트래픽을 이용한 능동적인 큐 관리

양 진 영[†] · 이 팔 진[†] · 김 종 화[†††]

요 약

IP 네트워크 상에 TCP 데이터 트래픽의 제공은 처리율과 공정성을 향상시키기 위해 특별한 기법이 필요하다. 여기에는 DT와 RED와 같은 많은 기법들이 제안되었다. RED 알고리즘은 폭주를 회피하고 적은 지연과 높은 처리율을 유지하기 위한 목적으로 제안되었다. 현재의 TCP/IP 환경에서 TCP 근원지는 Slow-Start 단계에 들어감으로써 드롭된 패킷에 반응하지만, 네트워크 이용률은 급속히 떨어진다. 폭주를 탐지하고, 이를 무작위로 선택된 연결에 통보하므로써 RED는 글로벌 동기화 및 공정성 문제를 유발한다. 본 논문에서는 성능을 향상시키기 위해 공정성을 유지하고, 글로벌 동기화 문제를 해결할 수 있는 능동적인 큐 관리 알고리즘을 제안한다. 제안된 알고리즘은 버퍼 크기와 임계치 변화에 따른 goodput, 처리율, 공정성의 평가인자를 이용하여 기존의 기법인 DT, RED와 비교 분석을 수행하고, 제안된 기법의 타당성을 보인다.

A Study on the Active Queue Management Scheme with TCP Traffic

Jin-Young Yang[†] · Pal-Jin Lee[†] · Jong-Hwa Kim[†††]

ABSTRACT

Supporting TCP data traffic over IP network requires special mechanisms to improve throughput and fairness. Several schemes have been proposed such as DT (Drop Tail), Random Drop and RED (Random Early Detection). RED algorithm was presented with the objective of avoiding congestion and maintaining the average queue length in a region of low delay and high throughput. In the current TCP/IP environment, TCP sources reacts to dropped packets by entering in the slow start phase, but network utilization will drop drastically. By detecting congestion and notifying only a randomly selected fraction of connection, RED causes to the global synchronization and fairness problem. In this paper, we study the active queue management algorithm to improve performance. and introduce an algorithm to maintain its fairness. It is also alleviates the global synchronization problem. Finally, we compare performance of proposed method with RED and Drop Tail in terms of goodput, network utilization and fairness.

키워드 : TCP 트래픽(TCP Traffic), 폭주 회피(Congestion Avoidance), DT, RED

1. 서 론

폭주는 링크에 유입된 트래픽 양이 용량을 초과할 때 통신 링크상에서 발생한다. 초과 트래픽은 버퍼가 채워짐에 따라 급속히 큐잉 지연을 증가시키며, 극단적인 경우 버퍼 오버플로우의 발생으로 인한 패킷이 손실된다. 이를 해결하기 위해서는 폭주 제어와 회복 및 회피 알고리즘이 필요하다. 인터넷상에서 사용중인 TCP는 적응적인 윈도우 기반의 흐름제어 기법을 제공하며, TCP에 의해 개발된 폭주 회피와 제어 알고리즘은 이용 가능한 자원을 최대로 사용하는데 그 목적이 있다[9, 12].

TCP의 처리율(throughput)과 공정성(fairness)을 향상시키기 위해 Floyd와 Jacobson에 의해 제안된 RED(Random Early Detection) 알고리즘이 있다[1, 4]. 이 밖에도 처리율과 공정성을 향상시키기 위한 다양한 방법이 연구되어 오고 있다. Heinanen과 Kilkki는 경쟁하는 연결들 사이에 이용 가능한 자원의 공정성을 위해 Fair Buffer Allocation (FBA)을 제안하였다[11]. RED는 종종 발생되는 버스트에 관계없이 큐의 상태에 따라 패킷 전송률을 일시적으로 감소시키므로써 평균 큐 크기를 가능한 낮게 유지하여 목적지에서 폐기될 불완전한 패킷으로 인한 congestion collapse를 피할 수 있는 기법이다. 그러나 이 방법은 패킷을 무작위로 드롭시키기 때문에 글로벌 동기화 문제를 가져오며, 패킷을 비례적으로 드롭시키기 때문에 공정성에 부정적인 효과가 있다[13]. RED에서 최적의 큐 크기를 결정하기 위해서는 네트워크 특성을 파악하여야 하며, 최대 임계치인 $\max_{th}$의 최적 값은 라우터에서 허용되는 최대 평균 지연에 묶여 있기 때문에 네트워크 상에서 운영되는 다양한 응용의 유형과 연관지어져야 한다[7].

† 정 회 원 : 초당대학교 컴퓨터과학과 교수
†† 정 회 원 : 목포대학교 정보공학부 교수
논문접수 : 2001년 4월 24일, 심사완료 : 2001년 5월 3일

따라서 본 논문에서는 처리율을 향상시키기 위해 능동적인 큐 관리 알고리즘을 제안한다. 이는 연결의 상태에 따라 평균 큐 크기가 최소 임계치와 최대 임계치 사이에 존재할 때 들어오는 패킷을 선택적으로 드롭시키기 때문에 RED가 갖는 문제인 공정성과 글로벌 동기화 문제를 해결할 수 있다. 그리고 제안된 방법은 버퍼 크기에 따른 goodput, 처리율, 그리고 공정성의 평가인자를 이용한 기존의 폭주제어 기법인 Drop Tail, RED와 비교 및 분석을 수행하며, 시뮬레이션을 통해 제안된 방법의 타당성을 검증한다.

본 논문의 구성은 다음과 같다. 제2장에서는 폭주 회피와 관련된 다양한 알고리즘을 비교 분석하며, 제3장에서는 폭주 회피를 위한 능동적인 큐 관리 기법을 제안한다. 제4장에서는 시뮬레이터를 이용하여 제안된 기법의 시뮬레이션을 수행하며, 얻어진 결과와 기존의 기법과의 비교 분석한다. 마지막으로 제5장에서는 결론을 다루며, 향후 수행되어야 할 방향을 제시한다.

2. 관련 연구

게이트웨이로부터의 폭주 탐지는 추정된 병목(bottleneck) 서비스 시간, 처리율의 변화, 종단간 지연 변화, 그리고 패킷 탈락으로 추정할 수 있다. 폭주 제어를 위한 많은 게이트웨이 알고리즘이 연구되어 왔다[12]. 이들은 Random Drop, Drop Tail, Source Quench, Congestion Indication, Selective Feedback Congestion Indication, Bit-around Fair Queuing 등이다. Random Drop은 패킷을 통계적으로 드롭시켜 폭주를 제공한 트래픽의 사용자에게 피드백을 전달하는 기법으로, 들어오는 모든 트래픽으로부터 무작위로 선택된 패킷은 사용자의 평균 전송률에 비례하여 드롭되며, 이때 선택된 패킷은 균일한 분포를 갖는다. 이 기법은 폭주 회복 기법(congestion recovery mechanism)으로 큐에 도착하는 마지막 패킷을 드롭시키는 것이 아니라 큐로부터 패킷이 무작위로 선택된다.

Source Quench는 RFC-792 ICMP(Internet Control Message Protocol)의 메시지로서 게이트웨이가 데이터그램(datagram)을 드롭시키므로써 폭주에 반응하면 게이트웨이는 드롭시킨 데이터그램의 소스에 ICMP Source Quench 메시지를 전송하는 폭주 회복 기법이다. Drop Tail은 모든 기법 중에서 가장 간단한 알고리즘으로서, 패킷을 선택적으로 드롭시키는 것이 아니라 이용 가능한 버퍼의 공간이 없을 때 곧 바로 드롭시킨다.

Drop Tail과 Random Drop 게이트웨이의 단점은 많은 연결로부터 패킷이 드롭 된다는 것과 동시에 처리율의 감소로 윈도우가 감소한다는 것이다. ERD(Early Random Drop) 게이트웨이는 Drop Tail과 Random Drop을 조금 더 향상시킨 것인데 이는 큐 사이즈가 어떤 임계치를 초과할 때마다 고정된 확률을 가지고 들어오는 패킷을 드롭 시킨다는 것이다. Floyd는 이들 중 어떤 것도 misbehaving 연결을 잘 다룰 수 없다고 지적하고 있다. [4, 10]에서의 Drop Tail은 다수 연결들 사이에 폭주가 발생한 링크의 이용률이 낮은 글로벌 동기화 문제를 제시하고 있다.

버퍼 오버플로우에 의한 패킷 손실을 막기 위해 또 다른 큐 관리 기법이 제안되었다. 이 기법은 초기 폭주를 탐지하여 이를 종단 호스트에 통보하므로써 큐 오버플로우로 인한 패킷 손실 전에 패킷 전송률을 줄이는 것이다. 이 중 대표적인 기법은 IETF에서 제안된 RED(Random Early Detection)이다[1].

이들 알고리즘들은 Slow-Start나 End-system Congestion Indication과 같은 종단 시스템 폭주 제어 기법에 의해 효율적으로 반응한다[12]. TCP는 네트워크에 폭주가 발생하면 많은 연결들이 가용자원에 대해 경쟁한다. TCP는 네트워크 링크를 공유하는 개별 연결들의 전송률을 조절하는데, 이를 위해 윈도우 기반(window-based)의 프로토콜을 이용한다. 여기에는 Slow-Start, 폭주회피(congestion avoidance), Fast Retransmit과 Fast Recovery가 있다[14]. 이들 TCP 알고리즘들이 현재 네트워크 상에서 갖는 문제는 큐의 오버플로우에 의해 패킷이 손실될 때 전송률을 줄이는 것인데, 이는 라우터에서의 패킷이 드롭되고 소스에서 패킷의 손실을 탐지하기까지는 많은 시간이 소요됨과 동시에 많은 패킷이 드롭된다는 것이다.

3. 능동적인 큐 관리 알고리즘

RED에서 가지고 있는 불공정성 효과를 줄이기 위해 패킷을 비례적으로 드롭시키므로써 선택된 연결(C_i)에 무작위로 폭주를 지시하는 대신 큐에 입력된 가장 많은 패킷을 갖는 연결들에 대해 선택적으로 드롭시키는 것이다[13]. 또한 Floyd는 서로 다른 네트워크와 트래픽 조건에 따라 드롭 수준과 드롭 확률을 동적으로 조정하여 최적의 평균 큐 크기를 결정하는 것을 강조하고 있다. 이 알고리즘은 평균 큐 크기를 계산하기 위해 식 (1)과 같은 EWMA(exponentially weighted moving average)를 이용한다[4].

$$avg = (1 - w_q)avg + w_q * Queue\_Size \quad (1)$$

그리고 패킷을 드롭시키는 확률 식 (2)의 p_a는 0~ $\max_p$ 사이의 값을 가지며, avg가 최소 임계치 $\min_{th}$에서 최대 임계치 $\max_{th}$에 이르기까지 선형적으로 변한다.

$$p_a = p_b/(1 - count * p_b) \quad (2)$$

$$p_b = \max_p(avg - \min_{th})/(\max_{th} - \min_{th}) \quad (3)$$

그러나 p_a는 p_b로 인하여 큐로 들어오는 패킷을 무작위

로 드롭시키기 때문에 각 연결(connection) 에 대한 공정성(fairness)을 유지할 수 없다. 폭주가 증가함에 따라 자원을 적게 사용하는 패킷보다는 많이 사용하는 패킷이 더 많이 드롭되기 때문이다. 따라서 이를 해결하기 위해서는 패킷을 무작위로 탈락시키는 것이 아니라 각 연결에 대한 부하의 상태를 고려하여 큐로 들어오는 패킷을 선택적으로 탈락시킬 필요가 있다.

식 (2)에서 avg가 $\min_{th}$와 $\max_{th}$ 사이에 존재할 때 연결의 상태에 따라 들어오는 패킷을 선택적으로 드롭시키기 위한 확률 p_b는 다음과 같이 계산할 수 있다.

```
[1] : Initialize parameter
    avg = 0 ;
    count = -1 ;
[2] : for n do [3-6] until max n
    begin
      [3] : calculate avg
            if the queue is nonempty
                  avg = (1 - w_q) · avg + w_q · q
            else
                  m = f(time - q_time)
                  avg = (1 - w_q)^m avg
      [4] : if min_th ≤ avg ≤ max_th
            increment count
            calculate probability p_a
                  compute L (Load Ratio)
                  compute A_fair
                  compute p_b
                  p_a = p_b / (1 - count · p_b)
      with probability p_a
                  drop the arriving packet
                  count = 0
      [5] : else if max_th ≤ avg
      drop the arriving packet
      count = 0
      [6] : else count = -1
    end
[7] : when queue becomes empty
      q_time = time
```

(그림 1) 수정된 RED 알고리즘

연결 (C_i)에 대한 Load Ratio (L)가 식 (4)와 같다고 할 때

$$L = (\ \#\ \text{of packet from}\ C_i\ ) / (\ \text{Fair Allocation}\ ) \quad (4)$$

이다. 여기서 Fair Allocation (A_{fair})은 식 (5)와 같이 구할 수 있다.

$$A_{fair} = queue\_size\ /\ \#\ of\ active\ C_i \quad (5)$$

N_a를 연결 (C_i) 수, Y_i를 각 C_i에서 발생된 패킷 수라 할 때, C_i의 L은 식 (6)과 같이 구할 수 있다.

$$L = Y_i \cdot N_a\ /\ queue\_size \quad (6)$$

따라서 식 (2)의 P_b는 식 (7)과 같이 표현할 수 있다.

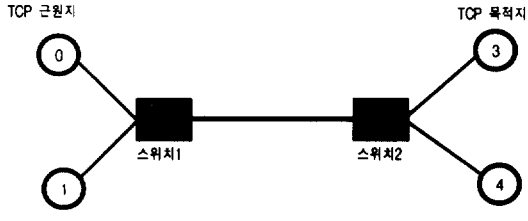
$$P_b = P_b \cdot Y_i \cdot N_a\ /\ queue\_size \quad (7)$$

최종 드롭핑 확률 p_a는 avg와 $count$가 증가함에 따라 증가하며, RED에서 패킷을 무작위로 드롭시켜 발생되는 글로벌 동기화 및 공정성의 문제는 패킷을 선택적으로 드롭시키므로써 해결할 수 있다.

4. 시뮬레이션

4.1 시뮬레이션 환경

제안된 기법의 시뮬레이션을 수행하기 위한 시뮬레이션 네트워크는 (그림 2)와 같으며, UNIX 환경을 기반으로 ns[5]와 Tcl and Tk[15]를 이용하여 수행하였다. 각 TCP 연결에는 서로 다른 전파 지연을 허용하며, 링크 대역은 TCP 근원지[목적지]에서 스위치까지는 10Mbit/s, 그리고 각 스위치 사이는 1Mbit/s로 설정하였다.



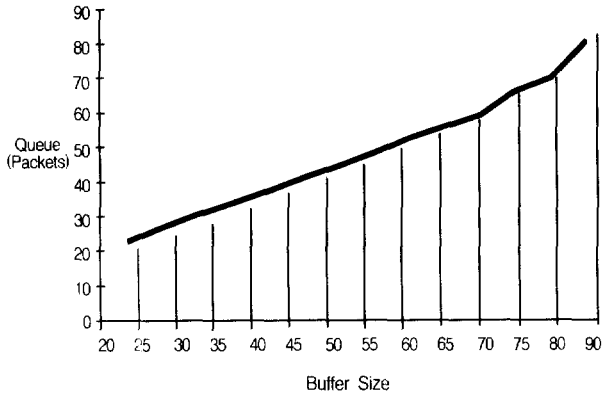
(그림 2) 시뮬레이션 네트워크

TCP 근원지와 스위치1 사이에 주어진 전파 지연은 각각 TCP와 UDP 트래픽에 대해 각각 10ms, 스위치1과 스위치2 사이는 40ms로 설정하였다. RED 및 제안된 능동적인 큐 관리 알고리즘의 시뮬레이션을 수행을 위해 사용된 파라메터는 W_q, $\min_{th}$, $\max_{th}$, $\max_p$이다[4]. W_q은 0.002, $\max_p$는 1/50로 설정하여 수행하였다. 버퍼 크기에 따른 오버플로우로 인한 패킷 손실을 확인하기 위해 DT에 대해서는 20~90 패킷으로 설정하고, RED와 개선된 기법에 대해서는 20~90 패킷으로 설정하여 버퍼 크기별로 최소 임계치를 변화시켜 시뮬레이션을 수행하였다. 그리고 패킷의 크기는 1000바이트로 하여 20초동안 수행하였다.

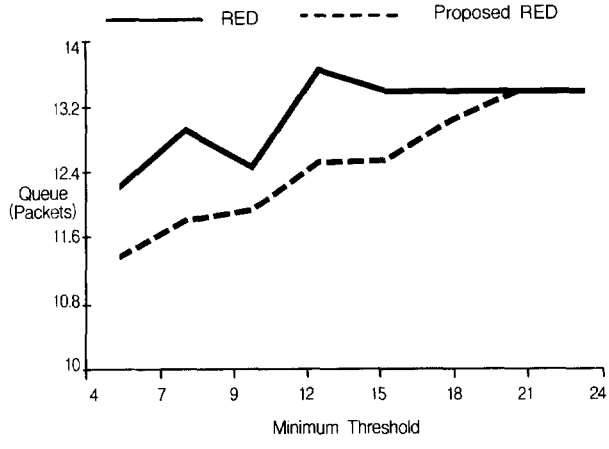
4.2 시뮬레이션 결과 분석

본 연구에서 제안된 기법의 타당성을 보이기 위해 벤치마크로서 DT(Drop Tail)와 RED(Random Early Drop)을 이용하였다. (그림 3)은 버퍼 크기를 증가시켜 가면서 나타낸 DT 알고리즘에서의 큐의 특성이며, (그림 4)는 버퍼 크기별로 최소 임계치의 변화에 따른 특성을 보여준다. 제안된 능동적인 큐 관리 기법의 특성이 임계치의 증가와 더불

어 안정됨을 볼 수 있다.



(그림 3) DT의 버퍼 크기에 따른 큐 특성

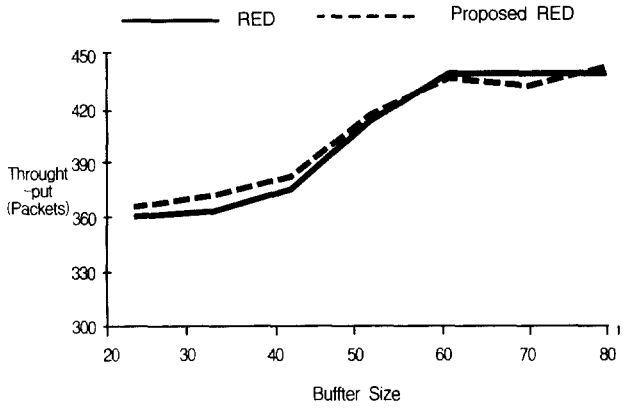


(그림 4) 임계치의 변화에 따른 큐의 특성

(그림 5)는 Drop Tail를 이용하여 버퍼 크기의 변화에 따른 트래픽 전송량을 보여준다.

(그림 5) DT에서의 버퍼 크기에 따른 전송된 패킷 수

(그림 6)은 RED와 제안된 기법과의 버퍼 크기에 따른 전송된 패킷 수를 보여준다. 이것은 버퍼 크기 증가와 함께 임계치도 증가시켜 나타낸 결과이다. 이때 최소 임계치와 최대 임계치는 [4]에서 제안된 바와 같이 $\max_{th}$를 3·$\min_{th}$으로 설정한 것이다.



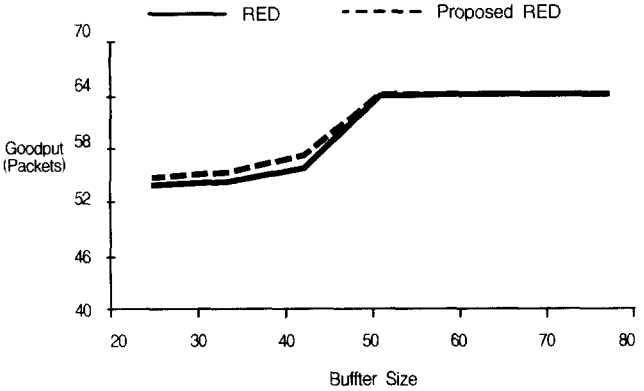
(그림 6) 버퍼 크기별 전송된 패킷 수

Goodput은 주어진 시뮬레이션 시간동안에 성공적으로 확인 신호를 받은 패킷들의 수로 계산되며, 공정성은 식 (8)를 이용하여 구할 수 있다[7].

$$Fairness\ Index = 1 - \frac{1}{N}\sum_{i=1}^{N}\left|\frac{t_i - \bar{t}}{\bar{t}}\right| \quad (8)$$

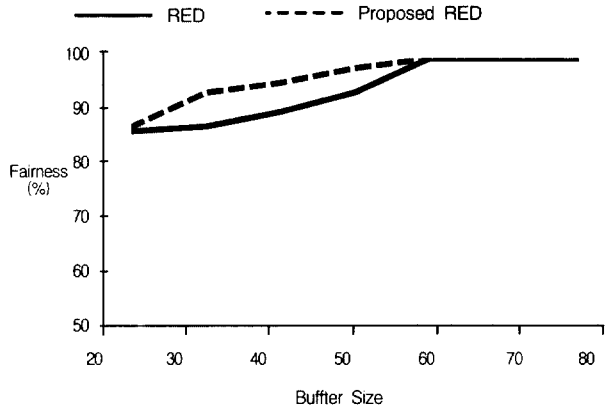
여기서 N은 소스의 수를 나타내며, t_i은 connection i의 처리율, $\bar{t}$는 평균 처리율을 나타낸다. (그림 7)과 (그림 8)은 각각 DT, RED와 제안된 기법과의 Goodput을 보여주고 있다.

(그림 7) DT의 Goodput



(그림 8) RED와 제안된 기법과의 Goodput

(그림 9)는 RED와 제안된 큐 관리 기법과의 공정성의 결과를 보여준다.



(그림 9) 공정성

본 시뮬레이션은 각 연결의 상태에 따라 패킷을 선택적으로 드롭시키기 때문에 DT나 RED에 비해 처리율과 goodput 측면에서 다소 높게 나타났다. 공정성 또한 증가된 처리율로 인하여 기존의 기법보다 향상된 결과를 얻을 수 있었다. 따라서 단지 큐 상태만을 이용하여 패킷을 무작위로 드롭시키는 기존의 기법보다는 큐 상태를 기반으로 연결 상태를 이용하여 패킷을 선택적으로 드롭시키는 것이 더 나은 결과를 얻을 수 있다. 그러나 연결 수가 극히 제한되어 있기 때문에 본 논문에서 얻어진 결과는 더 다양한 시뮬레이션이 필요하며, 이를 통해 제안된 방법의 타당성이 얻어질 것으로 사료된다.

5. 결론 및 향후 방향

폭주 회피 기법은 네트워크 트래픽 부하를 감시하여 병목지점에서의 폭주를 회피하는데 있으며, 이것은 패킷 드롭핑(packet dropping)에 의해 이루어진다. 현재 가장 널리 알려진 방법은 RED이다. 이 알고리즘은 평균 큐 크기를 낮게 유지하므로써 폭주를 회피한다. 그러나 평균 큐 크기를 최적으로 유지하기 위해서는 네트워크 특성을 파악하여야 하는데, 그렇지 못한 경우에는 글로벌 동기화 문제를 유발하고, 공정성을 떨어뜨린다.

본 논문에서는 IP 네트워크에서 TCP 트래픽을 중심으로 폭주 회피를 효율적으로 수행하기 위해 각 연결의 상태에 따른 능동적인 큐 관리 기법을 제안하였다. 제안된 방법은 각 연결에 대한 패킷 드롭핑의 공정성을 유지하고, 글로벌 동기화 문제를 해결하기 위하여 평균 큐 크기가 최소 임계치와 최대 임계치 사이에 존재할 때 들어오는 패킷은 연결 (C_i) 수, 연결에 따른 패킷 수, 큐 상태를 이용하여 부하를 계산하고, 이를 이용하여 들어오는 패킷의 드롭여부를 결정한다. 또한 패킷이 드롭될 연결의 결정은 RED에처럼 무작위로 이루어지는 것이 아니라 선택적으로 이루어지기 때문에 동기화와 공정성을 더욱 더 향상시킬 수 있다.

본 연구에서의 시뮬레이션은 제한된 TCP 트래픽만을 사용하였다. 따라서 더 많은 시뮬레이션 수행을 통한 결과의 타당성 검증이 필요할 것이며, 또한 다양한 네트워크 트래픽 특성을 고려한 시뮬레이션을 지속적으로 수행하여 RED가 가지고 있는 다양한 문제에 접근하고, 또한 이에 대한 해결방안을 연구할 것이다.

참 고 문 헌

[1] S. Floyd, et el, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC2309, April. 1998.

[2] Darius Buntinas, "Congestion Control Schemes for TCP/IP Networks," http : //www.cis.ohio-state.edu/~jain/cis788- 95/tcpip_cong.

[3] Mark Gaynor, "Proactive Packet Dropping Methods for TCP 'Gateways," http : //www.eecs.harvard.edu/~gaynor/final.ps.

[4] Sally Flyod and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, Aug. 1993.

[5] S. McCanne and S. Floyd. ns Network Simulator. http : //www.isi.edu/nsnam/ns/.

[6] Kevin Fall, Kannan Varadhan, The VINT project : ns Notes and Documentation, Feb. 25, 2000.

[7] Miguel A.Labrador and Sujata Banerjee, "Packet Dropping Policies for ATM and IP Networks," IEEE Communications Surveys, Vol.2, No.3, 1999.

[8] Vincent Rosolin, Olivier Bonaventure and Guy Leduc, "A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic," http : //montefiore. ulg.ac.be.

[9] Omar Elloumi, Hossam Afifi, "Improving RED Algorithm Performance in ATM Networks," GLOBECOM'97 IEEE Global Telecommunication Conference, 1997.

[10] Sally Floyd, Kevin Fall, "Promoting the of End-to-End Congestion Control in the Internet," IEEE/ACM Transactions on Networking, Vol.7, No.4, August 1999.

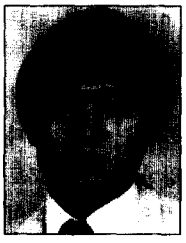
[11] J. Heinanen, K. Kikki, "A Fair Buffer Allocation Scheme," Unpublished Manuscript.

[12] A. Mankin, K. Ramakrishnan, "Gateway Congestion Control Survey," RFC : 1254, Aug.1991, http : //www-uxsup.csx.cam.ac.uk/netdoc/rfc/rfc1254.txt.

[13] Kevin Fall and Sally Floyd, "Simulation-based Comparison of Tahoe, Reno, and SACK TCP," http : //ee.lbl.gov.

[14] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control," Network Working Group RFC : 2581, April 1999, http : //www.isoc.org/inet2000/cdproceedings/rfc/rfc/2581.txt.

[15] Brent B. Welch, Practical Programming in Tcl and Tk, Prentice Hall PTR, 1997.

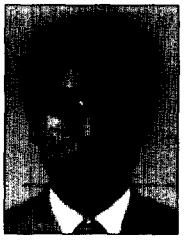


양 진 영

e-mail : jyyang@chodang.ac.kr
1983년 조선대학교 경영학과(경영학사)
1988년 조선대학교 전자계산학과 (공학석사)
2000년 목포대학교 컴퓨터공학과 박사수료
1997년~현재 초당대학교 컴퓨터과학과 조교수
관심분야 : TCP/IP, Traffic Control, MMI

이 팔 진

e-mail : pjlee@chodang.ac.kr
1986년 조선대학교 전산기공학과(공학사)
1988년 중앙대학교 전자계산학과 (이학석사)
1990년~1992년 목포대학교 조교
1995년 전북대학교 전자계산기공학과 (공학박사)
1995년~현재 초당대학교 컴퓨터과학과 조교수
관심분야 : TCP/IP, Networking QoS, Wireless Network



김 종 화

e-mail : kimjh@chungye.mokpo.ac.kr
1983년 조선대학교 전자공학과(공학사)
1985년 조선대학교 전자공학과(공학석사)
1989년 일본 동북대학 전자공학과(공학박사)
1991년~현재 목포대학교 정보공학부 부교수
관심분야 : Embedded System, IBS, Web-based control, HSI