韓國 컴퓨터情報學會 論文誌
第12卷 第3號, 2007. 7.

2007-12-3-2-3

TCP/IP 네트워크에서 ARED 알고리즘의 성능 개선

남 재 현*

Improve ARED Algorithm in TCP/IP Network

Jae-Hyun Nam*

요 약

인터넷에서 종단간 혼잡제어 방식을 지원하기 위해 제안된 라우터에서 큐에 적용되어 패킷을 폐기하는 방법으로 능동적 큐 관리(AQM:Active queue management) 방법이 적용되고 있다. IETF에서 제안된 AQM 알고리즘은 RED(Random Early Detection) 방식이다. RED 알고리즘은 망의 높은 처리율과 낮은 평균 지연을 얻기 위해 네트워크에서 운영된다. 하지만 평균 큐 길이는 네트워크의 혼잡 레벨에 민감한 결과를 갖게 된다. 본 논문에서는 RED 성능에 영향을 미치는 파라메터의 민감성을 감소시키고 성능을 개선시키기 위해 정련된 적응 RED(RARED; Refined Adaptive RED)를 제안한다. 시뮬레이션을 통해 네트워크 전체의 개선된 RARED 알고리즘을 관찰하고 RARED가 패킷 폐기율의 감소와 성공적인 전송률의 개선이 이루어짐을 보여준다.

Abstract

Active queue management (AQM) refers to a family of packet dropping mechanisms for router queues that has been proposed to support end-to-end congestion control mechanisms in the Internet. The proposed AQM algorithm by the IETF is Random Early Detection (RED). The RED algorithm allows network operators simultaneously to achieve high throughput and low average delay. However, the resulting average queue length is quite sensitive to the level of congestion. In this paper, we propose the Refined Adaptive RED(RARED), as a solution for reducing the sensitivity to parameters that affect RED performance. Based on simulations, we observe that the RARED scheme improves overall performance of the network. In particular, the RARED scheme reduces packet drop rate and improves goodput.

▸ Keyword : 혼잡제어(Congestion Control), RED(Random Early Detection), 능동적 큐 관리(Active Queue Management), TCP/IP, DT(Drop Tail), FRED(Flow RED), WRED(Weight RED), RIO(RED with In and Out)

• 제1저자 : 남재현
• 접수일 : 2007.4.16, 심사일 : 2007.4.17, 심사완료일 : 2007. 4.28.
* 안동과학대학 의료공학과 조교수

Ⅰ. 서 론

현재 대부분의 인터넷 응용이 사용하고 있는 TCP (Transmission Control Protocol) 혼잡(Congestion) 제어 방식은 인터넷에서 혼잡으로 인해 네트워크가 붕괴되는 것을 예방하기 위해 널리 사용되고 있다. 이것은 1988년에 Jacobson이 혼잡제어 방식을 보완한 것 외에는 초기 전송 구조를 그대로 유지하고 있다[1][2].

송신 윈도우를 네트워크의 혼잡으로 패킷의 손실이 발생할 때까지 선형적으로 증가시키면서 전송하다가 패킷의 손실이 설정된 임계치에 의해 탐지되면 송신 윈도우를 반으로 감소시킨 후 손실된 패킷을 재전송하고, 다시 선형적으로 송신 윈도우를 증가시키면서 전송하는 것을 반복한다. 따라서 TCP의 전송 능력은 전송 속도에 의해 결정되는 것이 아니고 수신 윈도우의 버퍼 크기와 RTT(Round Trip Time) 그리고 네트워크의 혼잡상태 등의 영향을 받는다.

이러한 윈도우 기반의 방법은 네트워크의 트래픽 용량을 초과한 후 패킷 손실을 검출하고 전송량을 감소시키는 문제점을 갖고 있다. 따라서 큐 관리 기법은 네트워크의 혼잡을 제어하는 중요한 역할을 위해 라우터에서 이루어진다.

현재 인터넷에서 가장 널리 사용되는 능동적인 큐 관리(AQM:Active Queue Management) 방법은 현재 IP 라우터에서 사용되는 DT(Drop Tail)이다. DT는 모든 기법 중에서 가장 간단한 알고리즘으로서 패킷을 선택적으로 폐기하는 것이 아니라 이용 가능한 버퍼의 공간이 없으면 즉시 패킷을 폐기한다[4][12][16].

하지만 DT는 글로벌 동기화 문제와 공정성에 대한 문제점을 가지고 있다. 능동적인 큐 관리(AQM) 기법은 인터넷에서 종단 간 혼잡제어 방법을 지원하기 위해 제공되었으며 라우터의 큐에서 패킷의 폐기 방법에 대해 적용된다.

능동적 큐 관리(AQM) 방식은 혼잡이 예상되는 라우터에서 먼저 패킷을 폐기시킨다.

RED(Random Early Detection)는 DT의 문제점을 해결하기 위해 제안되었다. TCP의 처리율과 공정성(fairness)을 향상시키기 위해 Floy와 Jacobson에 의해 제안된 RED 알고리즘이 있다[3][4][5].

RED 알고리즘은 혼잡을 회피하고 적은 지연과 높은 처리율을 유지하기 위한 목적으로 제안되었다.

RED는 라우터 버퍼에서 혼잡을 제어하기 위해 확률적으로 패킷을 폐기하거나 표시하고 큐 크기에 대해 EWMA (exponentially weighted moving average)를 이용하여 혼잡을 검출한다.

RED 알고리즘은 버퍼에서의 평균 큐 길이를 이용하여 혼잡제어를 수행한다. RED는 패킷이 도착할 때마다 큐의 평균치를 구하고 이것을 미리 설정한 파라메터인 최소 큐 임계값(minimum threshold:minth)과 최대 큐 임계값(maximum threshold:maxth)과 비교한다. 평균 큐 길이가 최소 큐 임계값 보다 작을 때에는 네트워크의 링크 사용이 낮은 수준에 머물러 있다고 판단할 수 있으므로 모든 패킷은 정상적으로 처리된다.

하지만 평균 큐 길이가 최소 큐 임계값과 최대 큐 임계값 사이에 있을 때 도착하는 패킷은 확률에 의해 임의적으로 폐기되거나 혼잡 통보(congestion notification)를 위한 비트(ENC)를 마크한다.

이 확률은 0에서부터 RED 내에 정의된 상수인 최대 확률(maximum probability:maxp)까지의 범위 내에 존재한다. 평균 큐 길이가 최대 큐 임계값 보다 클 때에는 평균 큐 길이가 최대 큐 임계값 밑으로 떨어질 때까지 도착하는 모든 패킷을 폐기하거나 ENC 비트를 마크한다[6][7][9].

RED의 성능을 해결하기 위해 많은 연구들이 RED 알고리즘을 수정하거나 개선을 하였다. 하지만 다양한 상황의 혼잡 상태를 제어하기 위한 좋은 성능의 RED 파라메터 값을 얻는 것이 아직 연구과제로 남아있다.

따라서 현재 인터넷에 적용된 적응적 접근방식을 주장하고 있으며 평균 큐 크기에 대한 트래픽의 부하에 대해 결론적으로 의미하는 것은 트래픽에 따라 가정을 갖고 접근해서는 안된다는 것이다.

최근 트래픽 부하에 따라 RED 파라메터를 조절하는 방식을 수행하는 적응적 RED(ARED; Adaptive RED)가 사용되고 있다.

본 논문에서는 ARED의 성능을 개선하고 낮은 패킷의 폐기율과 높은 처리율을 얻기 위해 평균 큐 크기를 조절하도록 하여 ARED의 최대 확률 maxp를 재정의 하고 RED의 기본 구조를 가진 ARED와 최소 큐 임계값과 최대 큐 임계값 사이에서 평균 큐 크기를 조절하는 파라메터인 maxp를 새롭게 정의한 RARED (Refined Adaptive RED) 알고리즘을 제안한다.

본 논문의 구성은 다음과 같다. 제2장에서는 Floyd가 제안한 ARED와 관련 연구에 대해 설명하고 제3장에서는 높은 처리율과 낮은 패킷 폐기율을 얻기 위해 확률 파라메터와 평가방법을 적용한 RARED 알고리즘을 제안한다. 제4장에서는 ns-2 시뮬레이터를 사용하여 제안된 RARED 알고리

즘을 시뮬레이션을 하며, 측정된 결과를 기존의 기법과 비교 분석한다[18]. 제5장에서는 결론을 다루며, 향후 연구해야 할 방향을 제시한다.

II. 관련연구

RED 알고리즘은 버퍼 오버플로우에 의한 패킷의 손실을 막기 위해 IETF에서 제안된 대표적인 기법이다.

이 기법은 초기에 네트워크의 혼잡 상태를 탐지하여 이를 종단 호스트에 통보함으로서 큐 오버플로우로 인한 패킷의 손실이 발생하기 전에 패킷 전송율을 줄이는 것이다[3][10].

RED 알고리즘의 기본적인 목적은 낮은 평균 큐잉 지연과 높은 처리율을 달성하는 것이다. 또 평균 큐 길이에 대해 최대 큐 임계값과 최소 큐 임계값 사이에서 패킷의 폐기율을 최소화 하고 공정성을 유지하기 위해 개선하도록 하는 것이다. RED 알고리즘의 문제점은 라우터의 링크를 공유하고 있는 다양한 트래픽의 형태에 대해 구분하지 않고 모두 동일한 기준에 의해 패킷의 폐기와 전송을 결정한다는 것이다.

FRED(Flow RED)에서는 트래픽의 형태를 TCP 혼잡제어 방식에 순응하지 않는 UDP 트래픽과 TCP의 혼잡제어에는 순응하지만 허용하는 범위 내에서 사용 가능한 대역을 모두 차지하는 트래픽, 그리고 허용하는 대역을 사용하는데 민감하게 반응하지 않는 텔넷(telnet) 서비스로 구분하고 있다[8]. FRED에서는 라우터의 큐에서 각 TCP 연결이 차지하고 있는 큐 길이를 추적함으로서 트래픽의 유형에 관계없이 모두 공정하게 링크를 사용할 수 있도록 하고 있다.

RED에서 maxp를 고정된 값으로 사용하지 않고 큐의 변화에 따라 maxp 값을 조정한다면 더 높은 링크의 사용 효율을 얻을 수 있다[9].

ARED(Adapted RED)는 라우터가 처리하는 트래픽의 부하에 따라 큐 길이의 변화를 감시하여 maxp를 변화시킴으로서 링크의 효율을 높이는 방안을 제안하고 있다[9][10].

WRED(Weight RED)는 IP precedence 비트로 트래픽을 구분하여 별도의 RED 큐를 운영하며, 각 큐에 대해 별도의 파라메터를 적용함으로서 트래픽 클래스에 따라 다른 패킷 폐기율을 적용하여 우선순위가 낮은 패킷을 먼저 폐기하도록 한다.

RIO(RED with In and Out)은 WRED와 같이 RED의 확장된 알고리즘으로 볼 수 있다. RIO 알고리즘은 서비스 사용 계약에 따라 두개의 큐를 만들어 RED 알고리즘을 적용한다[9]. 큐 길이가 최대 큐 임계값과 최소 큐 임계값 사이에 있을 경우 서비스 계약을 준수한 패킷은 폐기없이 전송하지만 서비스 계약을 위반한 패킷은 임의로 선택하여 폐기한다. 만약 최대 큐 임계값을 초과하게 되면 두개 큐의 패킷은 모두 임의로 선택되어 폐기되지만 서비스 계약을 위반한 큐의 패킷은 더 높은 확률로 폐기된다. RIO 알고리즘은 계약을 준수하는 사용자의 공정한 링크 대역을 이용하도록 보장하는 것을 목적으로 하고 있다[9][11].

이들 알고리즘들은 slow-start나 End-system congestion Indication과 같은 종단 시스템 혼잡제어 기법에 의해 효율적으로 반응한다. 이들 TCP 알고리즘들이 큐의 오버플로우에 의해 패킷이 손실될 때 전송률을 줄이는 것인데, 이것은 라우터에서 패킷이 폐기되고 소스에서 패킷의 손실을 탐지하기 까지 많은 시간이 소요되고 그 사이에 많은 패킷이 폐기된다는 것이다.

III. 제안된 알고리즘

RED와 ARED가 가진 성능을 개선하기 위해 RARED를 제안한다. RARED 알고리즘은 미리 설정된 큐 길이에 근접하도록 평균 큐 길이를 유지한다.

제안된 RARED 알고리즘은 미리 설정된 큐 길이에 근접하도록 평균 큐 길이를 유지하고 원하는 평균 큐잉 지연을 반영하도록 maxp를 적용한다. 이것은 보다 신속하게 목표값에 이르도록 평균 큐 길이를 유도한다.

제안된 알고리즘을 위해 미리 설정된 큐 길이와 함께 현재의 평균 큐 길이를 비교한다.

제안된 RARED 알고리즘은 그림 1에서 나타낸것과 같이 패킷을 폐기하거나 표시하는 방법을 사용한다.

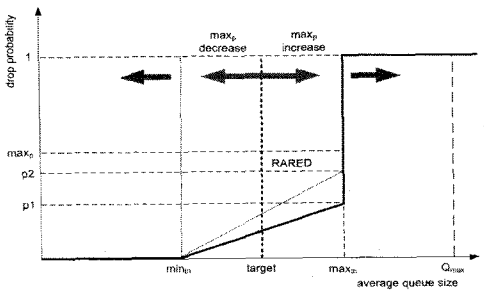

그림 1. RARED 패킷 폐기 기능의 설정
Fig. 1 The evolution of the RARED dropping function.

RARED 알고리즘에서 구현된 파라메터는 Floyd가 ARED 알고리즘에서 정의한 4개의 파라메터를 사용한다 [9][13].

ARED와 차이점은 0에서부터 미리 정의된 최대 확률 maxp는 미리 설정한 큐 길이에 근접한 평균 큐 길이를 유지하도록 적용하고, 최대 확률 maxp가 천천히 적응되고 전형적인 RTT(Round Trip Time) 보다 큰 시간 영역에 있고, 작은 간격으로 동작되도록 한다.

maxp는 [0.01, 0.5]의 범위 내에 존재하도록 구성한다. 또 maxp가 증가하고 감소하는 값을 곱하는 대신에 AIMD(Additive-increase multiplicative-decre ase) 방식을 사용한다.

For each interval seconds:

if (avg > target and maxp <= 0.5)

$$\alpha = 0.25 \times \frac{avg - target}{target} \times max_p$$

$max_p = max_p + \alpha$

elseif (avg < target and max_p >= 0.01)

$$\beta = 1 - \left(0.17 \times \frac{target - avg}{target - min_{th}} \right)$$

$max_p = max_p * \beta$

Variable:

avg : Current average queue size

Fixed Parameters:

interval : time; 0.5 seconds

target : Target for average queue size ;

$[min_{th} + 0.48 * (max_{th} - min_{th}),$

$min_{th} + 0.52 * (max_{th} - min_{th})]$

α : increment max_p

β : decrement max_p

그림 2. RARED 알고리즘
Fig. 2 The Refined Adaptive RED algorithm.

그림 2는 RARED 알고리즘을 나타낸 것이다. 최대 확률 maxp는 천천히 증가하며, 보다 작은 시간 범위를 지속하기 위해 평균 큐 길이에서 변화되는 응답은 패킷을 폐기하는 확률이 동적으로 빈번하게 동작하는 것을 허용할 수 있도록 했다.

최대 확률 maxp의 적용은 보다 긴 시간 범위를 요구하여 보다 긴 시간동안 관찰 할 수 있도록 한다.

IV. 시뮬레이션

4.1 시뮬레이션 환경

RARED 알고리즘을 평가하기 위해 RED 알고리즘과 ARED 알고리즘을 이용하였다. 시뮬레이션 네트워크는 그림 3과 같으며 ns-2 시뮬레이터를 사용하여 간단한 병목 현상을 갖도록 구현하였다[13][14][15][17].

네트워크는 두 대의 라우터로 구성하였으며, TCP와 UDP 노드로 연결하였다. TCP 소스는 S0, S1으로 설정하고 UDP 소스는 S2, ..., SN으로 설정하여 10Mbps의 속도와 1ms의 전파지연을 허용하도록 라우터 R1에 연결하였다. 라우터 R1은 10Mbps의 링크를 통하여 라우터 R2에 연결하였으며, 20ms의 전파지연을 갖는다. 목적지 노드 D0, D1, ..., DN은 10Mbps의 링크로 연결되며 1ms의 전파지연을 갖도록 구성하였다.

라우터의 버퍼는 1000바이트의 패킷을 위해 100 패킷을 허용하도록 한다. 트래픽은 UDP 트래픽(1Mbps, 병목 대역폭의 10%)의 양은 고정하고 장시간의 연결이 유지된 TCP 트래픽으로 이루어지도록 혼합하였다. 장시간의 연결을 유지하는 TCP 노드는 실험의 개시에 시작하여 실험의 전체 주기 동안 유지하도록 큰 FTP 전송으로 설정하였다. 모든 시뮬레이션에서 큐 가중치(Queue Weight) wq는 0.002로 설정하고 minth는 15 패킷으로 설정하였다. 최대 큐 임계치 maxth는 45 패킷으로 설정하였다.

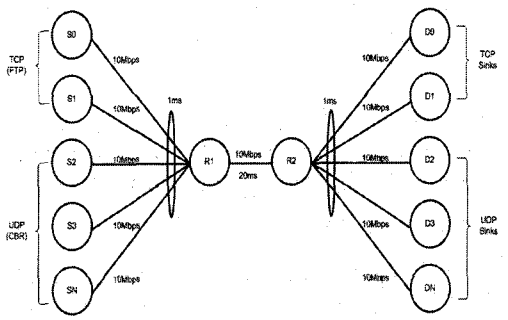

그림 3. ns-시뮬레이션 네트워크
Fig. 3 Network configuration for ns simulations

RARED 알고리즘의 평가에 사용된 측정 방법에 대해 나타낸다면 첫째, 라우터에서 TCP의 goodput은 라우터에서 최고의 사용 가능한 자원을 반영하여 측정한다[16]. TCP에서 goodput을 정의하면 중복된 패킷을 포함하여 모든 TCP를 위한 전달되는 대역폭에서 트래픽의 전체를 의미한다. 둘째, TCP와 UDP의 패킷 폐기율이다. 패킷 폐기율은 라우터의 입력 포트에 도착하는 패킷의 전체 수를 나누어 폐기된 패킷의 수를 정의한다. TCP goodput과 패킷의 폐기율은 약간의 중복된 측정을 가지게 되며, 패킷의 폐기율은 UDP를 사용한 응용을 위해 매우 중요한 역할로 충족시킬 수 있다.

이 측정 방법에 대해 추가적인 사항은 제안된 알고리즘의 제한사항이다. 첫째 이 측정 방법의 모든 네트워크 환경은 라우터를 기반으로 한다. 둘째 공정성(fairness)에 관련된 측정 방법은 고려하지 않았다.

4.2 시뮬레이션 결과분석

그림 4와 그림 5는 TCP goodput과 TCP/UDP의 패킷 폐기율을 나타낸 것이다. 그림 4에서 10%의 UDP 트래픽을 포함한 25개의 장시간의 TCP 트래픽을 지속하는 노드에 연결하여 TCP goodput과 TCP/UDP 패킷 폐기율을 관찰하였다.

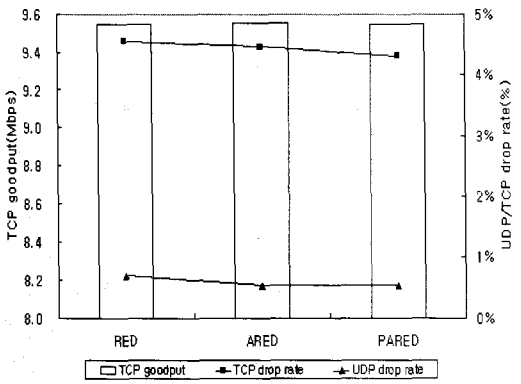

그림 4. TCP goodput과 TCP/UDP 패킷 폐기율
Fig. 4 TCP goodput(left) and TCP/UDP packet drop rate(right)(25 long-lived TCP connections with 10% UDP traffic)

그림 4에서 나타난 RARED 알고리즘의 시뮬레이션 결과를 보면 RED와 ARED보다 TCP goodput은 약간 높거나 비슷하게 나타나고 TCP/UDP 패킷 폐기율은 RED와 ARED보다 약간 낮게 나타남을 알 수 있다.

표 1은 UDP 트래픽을 포함한 25개의 장시간 TCP 연결을 지속한 TCP/UDP 패킷 폐기율을 보여준다.

TCP 패킷의 폐기율은 ARED 알고리즘 보다 우수한 성능을 나타내지만 RED 알고리즘과 비교하면 유사하거나 조금 나은 성능을 나태내는 것을 알 수 있다.

UDP 패킷 폐기율은 RED 알고리즘과 ARED 알고리즘 보다 우수한 성능을 나타내고 있다.

표 1. TCP/UDP 패킷 폐기율(%)
Table 1 TCP/UDP packet drop rate(%)(25 long-lived TCP connections)

		RED	ARED	RARED
TCP Packet drop rate	1 UDP	4.56	4.46	4.31
	2 UDP	4.44	4.95	4.83
	3 UDP	6.12	6.13	6.12
	4 UDP	6.69	6.84	6.43
UDP Packet drop rate	1 UDP	0.70	0.54	0.54
	2 UDP	0.66	0.66	0.53
	3 UDP	0.79	0.77	0.76
	4 UDP	0.83	0.87	0.71

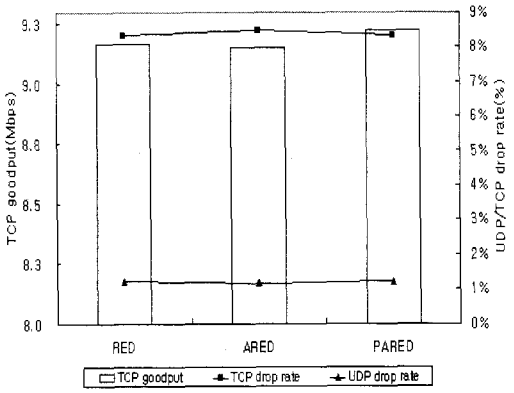

그림 5. TCP goodput과 TCP/UDP 패킷 폐기율
Fig. 5 TCP goodput and TCP/UDP packet drop rate(50 long-lived TCP connections with 10% UDP traffic)

그림 5는 10%의 UDP 트래픽을 가진 50개의 장시간 TCP 연결을 지속하여 동일한 방법으로 측정한 것을 나타내었다. RARED 알고리즘의 시뮬레이션 결과를 보면 RED와 ARED보다 약간 높은 TCP goodput을 보이고 약간 낮은 TCP/UDP 패킷 폐기율을 보여준다.

표 2는 UDP 트래픽을 포함한 50개의 장시간 TCP 연결을 지속한 TCP/UDP 패킷 폐기율을 나타내었다.

표 2. TCP/UDP 패킷 폐기율(%)
Table 2 TCP/UDP packet drop rate(%)(50 long-lived TCP connections)

		RED	ARED	RARED
TCP Packet drop rate	1 UDP	8.32	8.48	8.33
	2 UDP	11.04	11.00	10.33
	3 UDP	14.14	13.34	13.06
	4 UDP	15.35	15.38	15.05
UDP Packet drop rate	1 UDP	1.22	1.16	1.21
	2 UDP	1.31	1.57	1.41
	3 UDP	1.69	1.66	1.67
	4 UDP	1.67	1.49	1.52

그림 6에서 10%의 UDP 트래픽을 포함한 50개의 장시간 TCP 연결을 지속한 트래픽에 대한 RARED의 순간적인 큐 길이와 평균 큐 길이를 나타내었다.

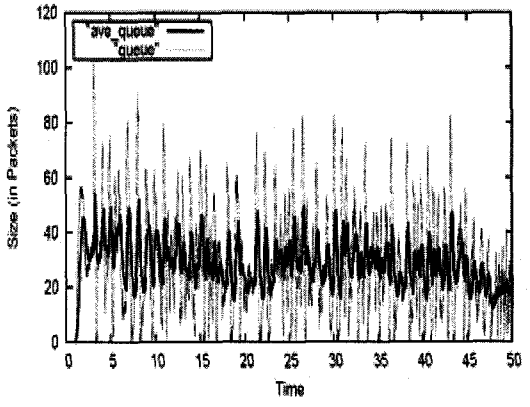

그림 6. RARED 큐 길이와 평균 큐 길이
Fig. 6 Average queue size and instantaneous queue size with RARED(50 long-lived connections with 10% UDP traffic)

그림에서 알 수 있듯이 평균 큐 길이는 미리 정의한 큐 길이에 비해 매우 조밀하게 나타나고 있다.

그림 7은 RED와 ARED 그리고 RARED의 평균 큐 길이에 대해 나타낸 것이다. RARED 알고리즘은 혼합된 트래픽에 대해 RED와 ARED에 비해 보다 안정적인 평균 큐 길이를 유지하고 있다.

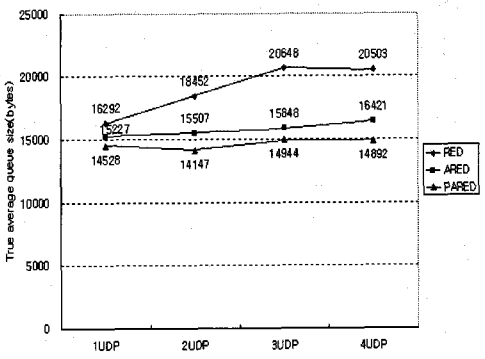

그림 7. RED, ARED, RARED의 평균 큐 길이
Fig. 7 Average queue size of RED, ARED and RARED(50 long-lived TCP connections with 1~4 UDP traffic)

V. 결 론

RED 알고리즘은 동적 큐 관리 방법에서 단순하고 효과적인 방법으로 알려져 있다.

하지만 평균 큐 길이가 혼잡 레벨에 매우 민감하고 RED 파라메터의 설정으로 인해 진보적이라고 단정하기는 어렵다. 트래픽 부하에서 ARED알고리즘은 매우 다른 결과를 나타냄을 알 수 있었다.

제안된 알고리즘은 트래픽 부하에 기반을 두고 폐기 확률을 적용하였으며 혼잡한 네트워크에서 패킷의 폐기율을 효과적으로 감소시킬 수 있고 혼합 트래픽에서도 평균 큐 길이를 안정적으로 유지할 수 있었다.

시뮬레이션 결과를 보면 RED와 ARED 알고리즘보다 RARED 알고리즘이 goodput과 패킷 폐기율에서 더 나은 성능을 가지며 평균 큐 길이도 안정적인 것을 알 수 있다.

향후 다양한 네트워크 트래픽 특성을 고려한 시뮬레이션을 지속적으로 수행하여 RED와 ARED가 가지고 있는 파라메터의 최적화에 관한 연구를 수행하여 이에 대한 해결방안을 연구하려고 한다.

참고문헌

[1] V. Jacobson, "Congestion Avoidance and Control," Proc. ACM SIGGOM '88, pp.314-329.

[2] Postel, J., Ed., "Transmission Control Protocol Specification," SRI Internati- onal, Menlo Park, CA, RFC-793, Sep, 1981.
[3] S. Floyd, et el, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC2309, April. 1998
[4] S. Floyd, and M. Jacobson, "Random Early Detection Gateways for Conge- stion Avoidance," IEEE/ACM Transac- ti ons on Networking, 1(4):397-413, Aug. 1993.
[5] S. Floyd, K. Fall, "Promoting the of End-to-End Congestion Control in the Internet," IEEE/ACM Transactions on Networking, Vol.7, No.4, Aug. 1999.
[6] S. Floyd, "TCP and Explicit Congestion Notification," Computer Communication Review,34(3):10-23, October 1994.
[7] K. Ramakrishnan, J. Raj, "A Binary Feedback Scheme for Congestion Avoidance in Computer Network," ACM Transactions on Computer Systems, 8(2):158-181, 1990.
[8] D. Lin, R. Morris, "Dynamics of Random Early Detection", Proc. of ACM SIGCOMM, Sept. 1997.
[9] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED : An Algorithm for Increasing the Robustness of RED's Active Queue Management,", http://www.icir.org/floyd/papers/adaptiveRed.pdf.
[10] W. Feng D. Kandlur, D. Saha, and K. Shin, "A Self-Configuring RED Gateway," INFOCOM'99, May 1999.
[11] T. Ziegler, S. Fdida, C. Brandauer, and B. Hechenleitner, "Stability of RED with Two-way TCP Traffic," IEEE ICCN, October 2000.
[12] V. Firoiu, and M. Borden, "A Study of Active Queue Management for Congestion Control," INFOCOM, 2000.
[13] S. Floyd and Kevin Fall, "Ns Simulator Tests for Random Early Detection (RED) Queue Management," April 1997.
[14] T. Bonald, M. May, and J. Bolot. "Analytic Evaluation of RED Performance," INFOCOM'00, 2000.
[15] S. Floyd, "RED: Discussions of Setting Parameters", http://www.icir.org/floyd/REDparameter.txt, November, 1997.
[16] G. Iannaccone, M. May, and C. Diot. "Aggregate Traffic Performance with Active Queue Management and Drop from Tail", ACM Computer Communication Review, July 2001.
[17] "ns-2 network simulator",http://www.isi.edu/nsnam/ns/.

저 자 소 개

남 재 현

1998년 2월 명지대학교 컴퓨터공학과, 공학박사
2001 ~ 현재 안동과학대학 의료공학과 조교수