중복 응답 임계값의 완만한 조절 기법을 통한 향상된 TCP-DAD†

박민우*, 김종명*, 한영주**, 권윤주***, 이길재***, 정태명****
*성균관대학교 전기전자컴퓨터공학과
**성균관대학교 컴퓨터공학과
***한국과학기술정보연구원
****성균관대학교 정보통신공학부
e-mail : {mwpark,jmkim,yjhan}@imtl.skku.ac.kr, {yulli,giljael}@kisti.re.kr, tmchung@ece.skku.ac.kr

Smooth Dupthresh Mechanism for Advanced TCP-DAD

Min-Woo Park*, Jong-Myoung Kim*,Young-Ju Han**, Yoonjoo Kwon***, Giljae Lee***,
Tai-Myoung Chung****
*Dept. of Electrical and Computer Engineering, Sungkyunkwan University
**Dept. of Computer Engineering, Sungkyunkwan University
***Korea Institute of Science and Technology Information
****School of Information Communication Engineering, Sungkyunkwan University

요 약

오늘날 네트워크는 장비의 발달로 인해 패킷 재배치(packet reordering)가 빈번히 일어나고 있다. 패킷 재배치는 TCP 의 성능을 저하시키는 문제점을 가지는데, 이러한 문제점을 해결하기 위해 TCP-DAD 메커니즘이 제안되었다. TCP-DAD 는 중복 응답 임계값을 동적으로 조절함으로써, TCP 의 패킷 재배치 문제를 완화하는 메커니즘이다. 하지만 TCP-DAD 는 중복 응답 임계값을 조절하는 메커니즘에 문제가 있어 패킷 손실이 일어날 경우, 패킷 손실에 대한 복구가 늦어지고 심지어 재전송 타이머가 불필요하게 종료되어 TCP 의 성능저하를 야기하기도 한다. 본 논문에서는 이러한 TCP-DAD 의 문제점을 해결하기 위해 새로운 중복 응답 임계값 조절 기법으로 Smooth-dupthresh 를 제안한다. NS-2 를 이용한 시뮬레이션을 통해 Smooth-dupthresh 기법을 적용한 TCP-DAD 가 기존의 TCP-DAD 보다 평균적으로 약 8% 향상된 성능을 보임을 증명하였다.

1. 서론

패킷 재배치란 전송과정 중에 패킷의 순서가 뒤섞여서 먼저 보낸 패킷이 나중에 보낸 패킷 보다 늦게 도착하는 현상을 말한다. 이러한 현상은 망의 부하 분산을 위한 다중 경로 라우팅(multi path routing), 라우트 플루터링(route fluttering) 기법이나, 라우터의 발달로 인한 라우터의 병렬처리 능력 등 다양한 요인들에 의해 보다 자주 발생하게 되었다. 패킷 재배치는 수신 측에게 중복 응답(duplicate acknowledgement)을 전송하게 하는데, 이를 수신한 송신 측은 네트워크가 혼잡하지 않음에도 불구하고 중복 응답에 의한 신속 재전송 신속 복구(fast retransmission and fast recovery)를 수행하게 된다. 이러한 혼잡 제어가 일어나면, 슬로우 스타트 임계값(Ssthresh)과 혼잡 윈도우(congestion window, cwnd)의 크기가 줄어들어 TCP 의 성능을 떨어지는 문제가 발생한다.

그 동안 이러한 문제점을 해결하기 위해 다양한 메커니즘이 제안되었으며 특징에 따라 총 네 종류로 분류할 수 있다. 첫 번째 종류로 중복 응답에 대해 기존의 TCP와 동일하게 신속 재전송 신속 복구를 수행한 후에, 혼잡 제어가 패킷 재배치에 의해 수행된 것이 확인되면 저하시킨 성능을 복구해주는 메커니즘들이 있으며 Eifel algorithm 이 대표적이다[1]. 두 번째 종류로 TCP-DCR(Delayed Congestion Responds TCP)[3]과 같이 중복 응답을 패킷 재배치에 의한 것으로 판단해서 혼잡 제어를 최대한 지연시켜 수행하는 메커니즘이 있다. 세 번째 종류로 모호한 중복 응답을 통한 혼잡 제어를 수행하지 않고 오직 타이머를 통해서만 혼잡 제어를 수행하는 메커니즘이 있다. TCP-PR(TCP for persistent packet reordering)이 여기에 속한다[4]. 마지막으로 중복 응답을 패킷 손실에 의한 것으로 판단하고 혼잡 제어를 수행하되, 패킷 손실의 발생을 판단하는 시점을 조절하는 방법들이 있는데, Blaton-allman algorithms이나 TCP-DAD(TCP dynamically adjusted duptresh)가 여기에 속한다[1,5].

† "본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음" (IITA-2008-C1090-0801-0028)

본 논문에서는 이러한 메커니즘 중 TCP-DAD의 문제점을 보완하는 Smooth-dupthresh 메커니즘을 제안한다. 기존의 TCP-DAD는 중복 응답 임계값을 패킷 재배치가 일어난 순간의 중복 응답 길이로 결정하기 때문에 보편성이 떨어진다. 또한 한번 증가된 중복 응답 임계값은 줄어드는데 오랜 시간이 걸리기 때문에 혼잡에 대응이 느리고 불필요하게 재전송 타이머가 만료 되는 등 TCP의 성능을 저하를 일으킨다. Smooth-dupthresh 메커니즘은 중복 응답 임계값을 보다 고르게 변화하도록 하여 이러한 TCP-DAD의 문제점을 완화하는 기법이다.

본 논문의 구성은 다음과 같다. 2장은 TCP-DAD의 메커니즘에 대해 간략히 안내하고, 3장에서는 본 논문에서 제안하는 Smooth-dupthresh 메커니즘과 이를 적용한 TCP-DAD-SD를 소개한다. 4장에서는 NS-2를 이용한 시뮬레이션 결과를 통해 TCP-DAD-SD의 성능을 보이고, 끝으로 5장에서는 결론과 함께 향후 연구해 나갈 방향에 대해 기술할 것이다.

2. 관련연구

2.1. TCP-DAD

TCP-DAD 는 패킷 재배치에 의해 혼잡하지 않은 상황에서 혼잡 제어를 수행하는 기존의 TCP 의 문제점을 해결하기 위해 제안된 기법이다. TCP-DAD 는 네트워크 상황에 따라 중복 응답 임계값을 동적으로 조절해 혼잡 제어의 발생 시기를 조절하여 패킷 재배치에 의한 성능 저하를 막는다. TCP-DAD 는 패킷 재배치를 빈번하게 겪는 세션에 대해서는 중복 응답 임계값을 늘려 혼잡 제어를 지연시키며, 패킷 손실을 빈번하게 겪는 세션에 대해서는 중복 응답 임계값을 낮게 유지하여 빠르게 혼잡 제어를 수행하는 메커니즘이다. 이와 같이 다른 메커니즘들과 달리 TCP-DAD 는 패킷 재배치나 패킷 손실 두 상황을 모두 고려하여 어느 한쪽으로 편향되지 않은 실용적인 해결방안을 제안한다.

TCP-DAD 의 동적 중복 응답 임계값 조절 메커니즘에 대해서 살펴보면, 이는 세가지 부분으로 이루어져 있다. 먼저 중복 응답 임계값을 설정하는 부분으로, 패킷 재배치가 검증되면 발생되며, 검증될 때까지 수신한 중복 응답의 개수로 중복 응답 임계값을 설정한다. 지속적인 축소 부분은 신속 재전송 신속 복구가 일어날 때마다 수행되며, 이는 매번 중복 응답 임계값을 1 씩 감소 시켜준다. 마지막으로 중복 응답 임계값의 재설정 부분으로, 재전송 타이머가 만료되거나 검증 타이머가 만료될 때까지 신속 재전송 신속 복구가 일어나지 않을 경우에 수행된다. 재설정이 수행되면 중복 응답 임계값이 기본값인 3 으로 재설정된다.

이와 같이 TCP-DAD 는 중복 응답 임계값을 패킷 재배치가 발생한 순간의 수신한 중복 응답의 개수로 설정한다. 따라서 한 순간의 패킷 재배치로 인해 중복 응답 임계값이 결정된다. 이렇게 결정된 중복 응답 임계값의 경우 보편적으로 사용하기에 부적합하다. 그 뿐만 아니라 값이 크게 설정되면 불필요하게 재전송 타이머가 만료되거나 혼잡 제어가 지연되어 TCP 의 성능을 저하시키는 문제가 발생한다.

3. TCP-DAD-SD

이 장에서는 본 논문에서 제안하는 Smooth dupthresh 기법을 적용한 TCP-DAD-SD 메커니즘을 소개한다.

3.1. 동작과정

<표 1>을 통해 TCP-DAD-SD(TCP-DAD-Smooth dupthresh)의 동작 과정을 확인할 수 있다. 이러한 TCP-DAD-SD 의 동작과정은 크게 3 가지 기능으로 세분화 할 수 있는데, 이는 Smooth-dupthresh 기법이 사용된 중복 응답 임계값 조절 기능, 패킷 재배치 검증 기능, 성능 복구 기능으로 나뉜다.

<표 1> TCP-DAD-SD 의 동작과정

Event	Code
Initialization	1 cwnd = 1 2 ssthresh = +∞ 3 dupthresh = 3 4 retransmitTag = FALSE
Received dupacks	5 if (++dupacks == 1) 6 set detect timer 7 else if (dupacks == dupthresh) 8 fastRetransmit() 9 fastRecovery() 10 retransmitTag = TRUE 11 else 12 send new packet
Received acks	13 if (detect timer pending) 14 calculate(dupthresh, dupacks) 15 ssthresh = prevssthresh 16 cwnd *= 2 17 cancel detect timer 18 retransmitTag = FALSE
Detect timer expire	19 if (retransmitTag == FALSE) 20 fastRetransmit() 21 fastRecovery() 22 dupthresh = 3 23 else 24 retransmitTag = FALSE
Retransmission timer expire	25 dupthresh = 3 26 retransmitTag = FALSE
Calculate	27 dupthresh=α*dupthresh + (1-α)*dupacks

TCP-DAD 의 중복 응답 임계값 조절 기능은 패킷 재배치가 검증되면 그 때의 수신한 중복 응답 개수를 중복 응답 임계값으로 사용하였으나, Smooth-dupthresh 기법은 세션이 시작된 이후 누적된 중복 응답 임계값과 중복 응답 개수를 통해 새로운 중복 응답 임계값을 구해내는 방법이다.

Smooth-dupthresh 에서 중복 응답 임계값은 수식 (1)과 같이 구해지며 dt 는 중복 응답 임계값, dt_S 는 중복 응답 임계값으로 적용될 새로운 값, dt_{pre} 는 이전의 중복 응답 임계값 그리고 dt_M 은 패킷 재배치를 검증한 순간 수신한 중복응답의 개수를 의미한다. 여기서 a 는 dt_{pre} 와 dt_M 의 비율을 나타내는데, 이러한 a 값은 사용자의 튜닝 작업에 따라 조정될 수 있다.

$$dt_S = \alpha \times dt_{pre} + (1-\alpha) \times dt_M \qquad (1)$$

TCP-DAD-SD 는 기존의 TCP-DAD 와 달리 중복 응답 임계값을 지속적으로 감소하는 기능을 사용하지 않는다. TCP-DAD-SD 에서는 이전 중복 응답 임계값을 누적하여 통계적인 값으로 새로운 중복 응답 임계

값을 구하는데, 이전 값을 1 씩 지속적으로 감소시키는 기법을 사용하면 이전 값이 가지는 통계적 정보가 사라진다는 단점이 존재하기 때문에 이러한 기능을 삭제하였다.

잘못 설정된 중복 응답 임계값으로 인해 불필요하게 재전송 타이머가 만료되거나 혼잡 제어에 대응이 늦어지는 현상이 일어날 수 있다. 이러한 현상이 반반복해 일어나는 것을 막기 위해 위와 같은 상황이 발생하면 중복 응답 임계값을 3 으로 재설정한다..

패킷 재배치를 검증하기 위한 수단으로 검증 타이머를 사용한다. 검증 타이머는 첫 번째 중복 응답을 수신하면 설정되는 타이머이다. 이때 그 크기는 왕복시간(RTT) 으로 설정된다. 검증 타이머는 작동된 시간 내에 중복 응답을 일으킨 패킷에 대한 긍정 응답을 수신하면 타이머를 중단하고 패킷 재배치에 의한 중복 응답으로 판단한다. 타이머 이내에 재전송한 패킷에 대한 응답이 오지 않으면 패킷 손실에 의한 중복 응답으로 판단한다.

검증 타이머의 장점은 다른 검증방법들과 달리 검증에 있어 추가적인 메시지를 생성 하지 않기 때문에 추가 비용이 들지 않아 경제적이다.

신속 재전송 신속 복구가 수행된 후에 해당 패킷에 대해 패킷 재배치가 검증되면, TCP-DAD-SD 는 패킷 재배치에 의한 잘못된 신속 재전송 신속 복구가 발생한 것으로 판단한다. 이 경우 불필요한 혼잡 제어를 통해 저하된 TCP 의 성능을 복구하는 기능이 수행된다. 그 과정은 수식 (2)와 같다. 여기서 *prevSsthresh* 는 이전의 중복 응답 임계값을 의미하며, $Ssthresh_{cur}$ 는 복구된 새로운 중복 응답 임계값을 의미한다. *PrevSsthresh* 는 신속 재전송 신속 복구에 의해 혼잡 제어가 수행되기 직전에 현재의 중복 응답 임계값으로 설정된다. $Cwnd_{cur}$ 은 현재의 혼잡 윈도우의 크기를 나타낸다.

$$Ssthresh_{cur} = prevSsthresh \qquad (2)$$

$$Cwnd_{cur} = 2 \times Cwnd_{cur} \qquad (3)$$

3.2. 대표 시나리오

TCP-DAD-SD 의 동작 과정을 대표 시나리오들을 통해 보다 자세하게 살펴보자.

3.2.1. 패킷 재배치에 의한 중복 응답이 발생한 경우

특정 패킷에 대해 중복 응답 임계값만큼의 중복 응답을 수신하게 되면, 송신자는 혼잡에 의한 패킷 손실로 받아 들이고 신속 재전송 신속 복구 기법을 수행한다. 이를 통해 중복 응답 임계값과 혼잡 윈도우(cwnd)의 값을 제한하고, 해당 세션의 전체 트래픽을 반으로 줄인다. 하지만, 검증 타이머를 통해 신속 재전송 신속 복구가 불필요하게 수행 되었다면, 강제로 저하시킨 TCP 의 성능을 복원하고 Smooth Dupthresh 를 구해 새로운 중복 응답 임계값으로 사용한다.

3.2.2. 패킷 손실에 의한 중복 응답이 발생한 경우

특정 패킷에 대해서 중복 응답 임계값 이상의 중복된 응답을 수신하게 되면 혼잡으로 판단하고 신속 재전송 신속 복구 기법을 수행하여 해당 세션의 성능을 반으로 줄인다. 이후 첫번째 중복 응답에 의해 발생된 검증 타이머가 만료될 때까지 중복 응답을 발생시킨 특정 패킷에 대한 긍정 응답을 수신하지 못하면, 패킷 손실에 의한 중복 응답으로 판단하고, 별다른 조절 없이 통신을 지속한다.

3.2.3. 중복 응답 임계값이 잘못 설정된 경우

중복 응답을 수신하여 검증 타이머를 작동하였으나 중복 응답 임계값 동적 조절 기법에 의해 중복 응답 임계값이 기존 TCP 의 3 보다 커서 검증 타이머가 만료되는 시점까지 신속 재전송 신속 복구가 수행 되지 못하는 경우가 발생한다. 이 경우 검증 타이머가 종료하는 시점에서 강제적으로 신속 재전송 신속 복구 기법을 수행해 준다. 또한 중복 응답 임계값이 잘못 설정되었기 때문에 신속 재전송 신속 복구 기법이 수행되지 못한 것으로 판단하고 이 값을 3 으로 초기화 해준다.

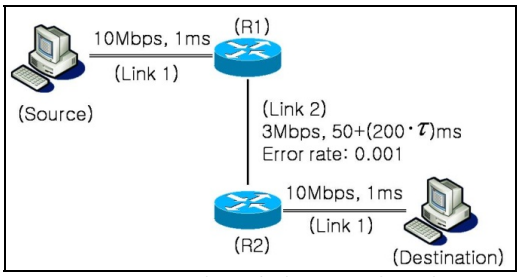
3.2.4. 재전송 타이머가 만료되는 경우

재전송 타이머가 만료될 경우 현재 설정된 중복 응답 임계값이 잘못 설정된 것으로 판단하고 기존의 TCP 값인 3 으로 초기화 한다.

4. 시뮬레이션

4.1. 시뮬레이션 환경

시뮬레이션은 NS-2 를 통해서 수행되었다. 이때 토폴로지는 공정한 성능평가를 위해 참고문헌에서 사용한 토폴로지를 바탕으로 구성했다[2]. 토폴로지는 (그림 1)과 같다.



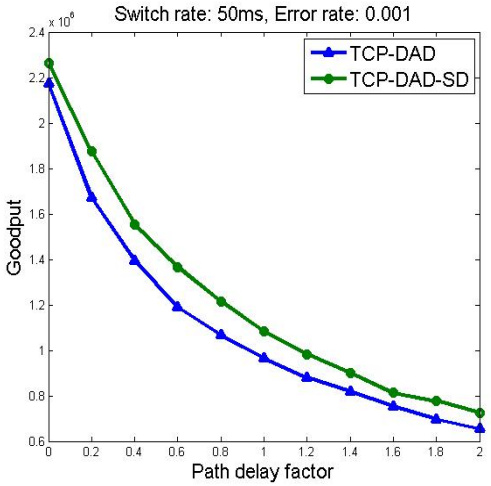
(그림 1) 시뮬레이션 토폴로지

네트워크는 (Source), (Destination), (R1), (R2) 4 개의 노드와 2 종류의 링크로 이루어져 있다. (Link 1)은 10Mbps 대역폭, 1ms 의 전파지연을 가지며, (Link 2)는 3Mbps 대역폭, 평균 50+200 τ ms 의 전파지연을 가진다. 이때 표준 편차는 (200 τ)/3 ms 이며, 2 는 0 부터 2 사이의 값으로 0.2 간격으로 설정하였다. 오류 모델을 적용하였으며, 이때 오류율은 0.001 을 적용하였다.

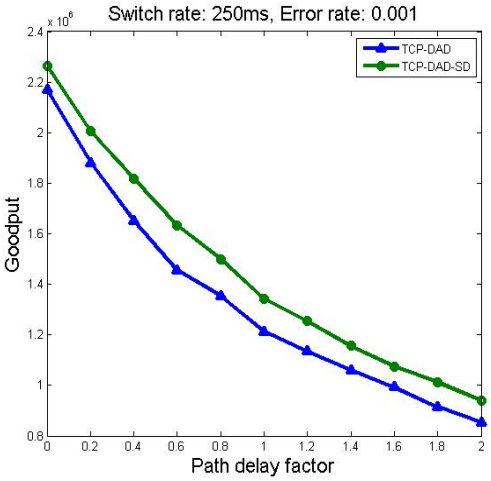
설정한 네트워크 매개변수로 400 세그먼트 크기의 버퍼를 (Source), (Destination)에 설치하였으며, 이때 세그먼트의 크기는 1500bytes 로 정의하였다. 혼잡 윈도

우의 최대 크기는 500 세그먼트로 설정하였다. 패킷 재배치를 발생시키기 위해 값을 특정 시간간격으로 바꿔 주었는데 이때 그 간격은 50ms, 250ms 의 크기로 시뮬레이션 하였으며, 10 초에 전송을 시작하여 1010 초에 전송을 종료하는 방법으로 총 1000 초간 진행되었다. 패킷 재배치는 난수에 따라 발생하는데 이때 씨드 값(seed value)를 총 20 가지 다른 값을 적용하여 시뮬레이션 하였다.

4.2. 시뮬레이션 결과



(그림 2) 패킷 재배치 발생 간격 50ms 오류율 0.001



(그림 3) 패킷 재배치 발생 간격 250ms 오류율 0.001

시뮬레이션 결과는 위의 (그림 2), (그림 3)과 같다. 시뮬레이션 결과 TCP-DAD-SD 가 TCP-DAD 에 비해 평균 약 8% 높은 성능을 냈다. 이러한 성능차이는 TCP-DAD 의 경우 패킷 재배치가 발생할 때마다 중복 응답 임계값이 급격히 변하기 때문에 생긴다. 급격하게 증가한 중복 응답 임계값의 경우 재전송 타이머가 만료되는 상황을 초래하기 쉬운데 비해 TCP-DAD-SD 의 경우 누적된 중복 응답 임계값을 취하기 때문에, 중복 응답 임계값이 점진적으로 증가하여 TCP-DAD 에 비해 보다 안정적인 값을 갖는다. 또한 TCP-DAD-SD 의 경우가 보다 빠르게 혼잡에 대응할 수 있어서 시뮬레이션 환경에서 보다 높은 성능을 나타내었다.

이러한 시뮬레이션 결과를 통해서 TCP-DAD-SD 가 기존의 TCP-DAD 보다 높은 성능을 띠는 것을 확인할 수 있다.

5. 결론

본 논문에서는 TCP-DAD 의 핵심이 되는 기법인 동적 중복 응답 임계값 조절 기법을 보완한 새로운 기법인 Smooth-dupthresh 를 제안하였다. 이는 기존의 TCP-DAD 의 중복 응답 임계값이 패킷 재배치가 발생할 때마다 급격하게 변하던 것에 비해 Smooth-dupthresh 의 경우 점진적으로 변화하게 설계되었다. 따라서, 기존의 TCP-DAD 가 가지고 있던 불필요하게 재전송 타이머가 만료되는 문제와 혼잡 제어의 대응 시기가 늦춰지는 문제점을 완화하였으며, NS-2 시뮬레이션을 통해 이를 증명하였다.

향후 연구로는 TCP-DAD-SD 의 최적화를 위한 α 값에 관한 연구와 검증 타이머의 크기에 의해 발생될 수 있는 불필요하게 재전송 타이머가 만료되는 이벤트를 예방하기 위한 최적화된 검증 타이머의 크기에 관한 연구를 수행할 것이다.

참고문헌

[1] R. Ludwig and R.H. Katz, “The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions,” ACM SIGCOMM Computer Comm. Rev., vol. 30, no. 1, pp. 30-36, Jan. 2000.

[2] K.-C. Leung, O. K. Li and D. Yang, “An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges,” IEEE TRANSACTIONS ON PARALLELAND DISTRIBUTED SYSTEMS, vol. 18, no. 4, Apr. 2007.

[3] S. Bhandarkar, et al., “TCP-DCR: A Novel Protocol for Tolerating Wireless Channel Errors,” IEEE TRANSACTIONS ON MOBILE COMPUTING, vol. 4, no. 5, Sep. 2005.

[4] S. Bohacek, et al., “A New TCP for Persistent Packet Reordering,” IEEE/ACM TRANSACTIONS ON NETWORKING, vol. 14, no. 2, Apr. 2006.

[5] 박민우, 이제민, 김종명, 한영주, 정태명, “TCP-DAD: 중복 응답 임계값의 동적 조절을 통한 TCP 의 패킷 재배치 내성 향상 기법”, 한국정보처리학회 추계학술발표대회, vol14, issue 2, Nov 2007

[6] A. Sathiaseelan and T. Radzik, “Improving the Performance of TCP in the Case of Packet Reordering”, Lecture Notes in Computer Science, vol. 3079, p. 63-73, June/July 2004.

[7] E. Blanton and M. Allman, dzik, “Improving the Performance of TCP in the Case of Packet Reordering”, Leication Review, vol.32, issue 1, p.20-30, Jan. 2002.

[8] S. Floyd, et al., An Extension to the Selective Acknowledgement(SACK) Option for TCP, IETF RFC 2883, July 2000.