높은 RTT 변동이 있는 네트워크에서의 QUIC을 위한 향상된 손실 감지 기법

김민기*, 조유제
경북대학교
gms03158@knu.ac.kr, yzcho@ee.knu.ac.kr

Improved Loss Detection Scheme for QUIC in Networks with High RTT Variations

Min-Ki Kim*, You-Ze Cho
School of Electronic and Electrical Engineering, Kyungpook National University

요 약

QUIC은 빠른 인터넷 서비스를 제공하기 위해 Google이 개발한 전송 프로토콜이다. QUIC은 TCP보다 검색 지연을 데스크톱에서 8%, 모바일에서 3.6% 줄일 수 있다. 하지만 모바일 네트워크와 같이 RTT가 급격하게 변하는 환경에서는 패킷 암호화 및 탄력적이지 못한 손실 감지 기법 때문에 TCP보다 성능이 떨어진다. 본 논문에서는 RTT가 변하는 환경에서 QUIC의 성능을 향상시키기 위해 시간 기반 손실 감지 기법을 개선한 Dynamic-QUIC (D-QUIC)을 제안하고 성능을 평가하였다. 성능평가 결과 D-QUIC은 파일을 다운로드 하는데 걸리는 시간을 기존 QUIC에 비해 최대 52% 향상을 보인다.

Ⅰ. 서 론

QUIC [1]은 빠른 인터넷 서비스를 제공하기 위해 Google이 2012년에 제안하고 2021년에 표준으로 채택된 새로운 전송계층 프로토콜이다. Google이 발표한 논문 [2]에 따르면 QUIC은 이미 Chrome, YouTube 등 다양한 서비스에 적용되었고, 2017년 기준 글로벌 트래픽의 7% 이상을 차지하고 있다. 같은 논문에서 실제 네트워크에서 사용자의 경험을 조사한 결과 QUIC을 사용하였을 때가 TCP를 사용했을 때보다 검색 지연을 데스크톱에서 8%, 모바일에서 3.6% 줄일 수 있다고 밝혔다.

하지만 QUIC은 개발된 지 얼마 되지 않은 만큼 완전히 TCP를 대체하기엔 부족하다. 특히 QUIC은 모바일 네트워크와 같이 RTT가 빠르게 변하는 환경에서 성능이 급격하게 저하된다. RTT가 급격하게 변하는 환경에서는 패킷이 전달되는 속도가 일정하지 않아 송신측에서 전달한 순서와 다르게 수신측에 패킷이 도달하는 패킷 재정렬이 빈번히 발생하기 때문이다. 패킷 재정렬이 전송 프로토콜의 성능에 악영향을 끼치는 이유는 패킷 재정렬이 발생하면 실제로 패킷이 손실되지 않았더라도 해당 패킷을 손실로 간주하여 재전송하는 가짜 재전송이 발생할 수 있기 때문이다. QUIC은 패킷이 암호화되어있어서 네트워크상에서 재정렬을 감지하여 보완할 수 없기 때문에 TCP보다 패킷 재정렬에 취약하다 [3]. 관련 연구 [4]에 따르면 RTT가 변하는 환경에서 QUIC이 TCP보다 40% 이상까지도 성능이 낮게 측정된다.

본 논문에서는 RTT가 빠르게 변하는 환경에서 QUIC의 성능을 개선시키기 위해 QUIC에서 사용하는 시간 기반 손실 감지 기법을 개선한 Dynamic-QUIC (D-QUIC)을 제안한다. 또한 기존의 QUIC과 D-QUIC의 성능을 비교 및 평가한다.

Ⅱ. 본론

A. D-QUIC의 향상된 손실 감지 기법.

QUIC은 손실 감지 기법으로 패킷 임계값과 시간 임계값을 사용할 수 있다 [3]. 패킷 임계값을 사용하는 기법은 앞서 전송한 패킷에 대한 확인 패킷이 이후 전송한 패킷에 대한 확인 패킷이 도착하기 전까지 도착하지 않으면 해당 패킷을 손실로 간주하고 재전송하는 방법이다. 시간 임계값을 사용하는 기법은 전송한 패킷에 대해 일정시간이 지나도 확인 패킷이 도착하지 않으면 해당 패킷을 손실로 간주하고 재전송하는 방법이다. QUIC이 사용하는 패킷 임계값은 3이고, 시간 임계값은 식 (1)과 같다.

$$\max(\frac{9}{8}max(SRTT, RTT_{latest}), 1ms) \quad (1)$$

하지만 기존의 임계값들은 RTT의 변동을 빠르게 반영하지 않기 때문에 급변하는 RTT로 인한 패킷 재정렬을 손실로 잘못 판단할 수 있다.

본 논문에서는 위와 같은 문제를 개선하기 위해 손실 감지 기법을 시간 임계값을 사용하는 방법으로 고정하고 시간 임계값을 식 (2)와 같이 수정한 D-QUIC을 제안한다.

$$\max(\frac{9}{8}max(SRTT, RTT_{latest}) + 4RTT_{VAR}, 1ms) \quad (2)$$

RTT_{VAR}를 시간 임계값에 추가함으로써 D-QUIC은 RTT의 변화를 빠르게 감지할 수 있게 된다. 결과적으로, 기존의 시간 임계값을 초과하는 패킷 재정렬이 발생하더라도 이를 손실로 잘못 판단하는 빈도를 줄일 수 있다. 따라서 가짜 재전송이 줄어들게 된다.

B. 실험 환경

D-QUIC의 성능을 평가하기 위해 golang 기반의 QUIC 구현 [5]에 수정된 시간 임계값을 적용하였다. 실험은 Mininet에서 진행되었으며, 링크의 대역폭, RTT, 손실률과 RTT jitter를 설정하기 위해 리눅스에서 제공하는 Token Bucket Filter와 NetEm을 사용하였다. 실험에 사용된 네트워크 매개변수는 표 1과 같이 설정하였고, 실험에 사용된 토폴로지는 그림 1의 네트워크 구성을 사용하였다.

실험은 10 MB 크기의 파일을 다운로드 하는데 걸리는 시간을 측정하였고, 각 시나리오마다 30번씩 실험을 반복했다. 혼잡제어 알고리즘은 CUBIC을 사용하였다.

표 1. 실험에서 고려된 네트워크의 특성.

Bandwidth [Mbps]	25, 50, 75, 100
RTT [ms]	10, 30, 50, 100
Loss Rate [%]	0.1
Jitter	10% of RTT

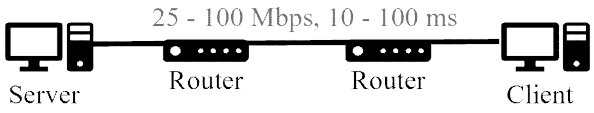

그림 1. 실험에 사용된 네트워크 구조.

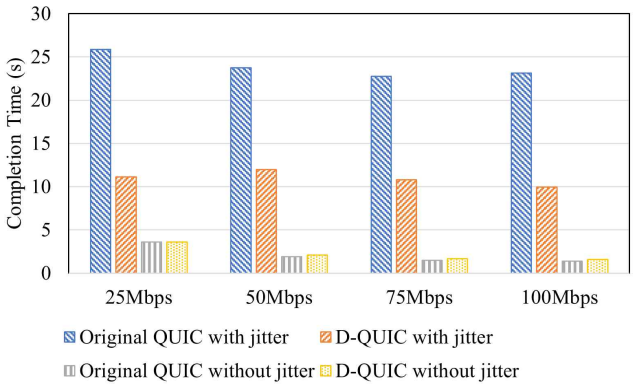

그림 2. RTT가 30 ms인 링크에서 대역폭이 25, 50, 75, 100 Mbps 일 때 QUIC과 D-QUIC의 다운로드 완료 시간.

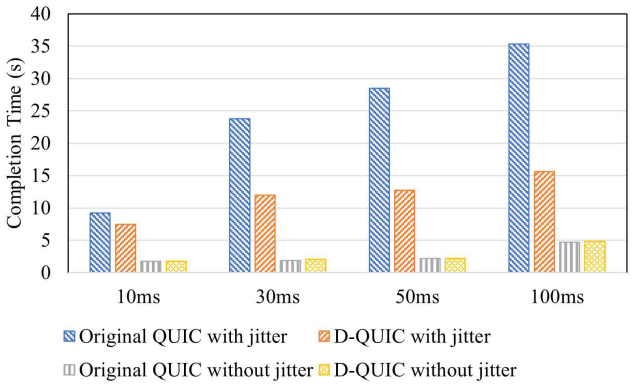

그림 3. 대역폭이 50 Mbps인 링크에서 RTT가 10, 30, 50, 100 ms 일 때 QUIC과 D-QUIC의 다운로드 완료 시간.

C. 실험 결과

그림 2는 RTT가 30 ms, 링크의 대역폭이 각각 25, 50, 75, 100 Mbps인 환경에서 10 MB 크기의 파일을 다운로드 하는데 걸리는 시간을 나타낸 그래프이다. RTT에 jitter가 있는 상황에서 QUIC의 성능이 매우 낮은 것을 볼 수 있다. 각각 다운로드 완료시간은 25.8, 23.7, 22.7, 23.1 초가 걸린다. 이는 시간 임계값이 RTT의 변동을 빠르게 반영하지 않기 때문에 확인 패킷을 받는데 걸리는 시간이 1.125 SRTT나 1.125 RTT_{latest}보다 길 경우 실제 패킷 손실이 일어나지 않더라도 곧바로 손실로 간주하여 패킷을 재전송하기 때문이다. 이러한 결과는 jitter가 대역폭보다 QUIC의 성능에 더 큰 영향을 미친다는 것 또한 보여준다. 반면 D-QUIC은 시간 임계값이 RTT의 변동 정도를 빠르게 반영할 수 있기 때문에 가짜 재전송이 줄어들어 QUIC보다 더 나은 성능을 보인다. 다운로드 시간은 각각 11.1, 12, 10.8, 10초가 걸렸다. RTT에 jitter가 없는 환경에서는 RTT_{VAR}가 0이 되어 QUIC과 D-QUIC이 비슷한 성능을 보인다.

그림 3은 대역폭이 50 Mbps인 링크의 RTT가 각각 10, 30, 50, 100 ms인 환경에서 10 MB 크기의 파일을 다운로드 하는데 걸리는 시간을 나타난 그래프이다. 이를 통해 RTT가 급격하게 변하는 환경에서 D-QUIC이 더 효과적으로 작동한다는 것을 알 수 있다.

하지만 jitter가 있는 환경과 없는 환경을 비교하면 시간 임계값에 RTT의 변화를 반영하였더라도 지터가 있는 환경에서의 D-QUIC 성능이 지터가 없는 환경에서보다 낮은 것을 볼 수 있다. 이는 RTT_{VAR}가 적응하는 속도보다 더 급격하게 RTT의 변동이 발생할 경우 여전히 가짜 손실을 감지할 수 있기 때문이다.

Ⅲ. 결론

본 논문에서는 모바일 네트워크와 같이 RTT가 변하는 네트워크에서 QUIC의 성능을 향상시키기 위해 시간 기반 손실 감지 기법을 개선한 D-QUIC을 제안하고 성능을 평가하였다. 성능 평가 결과 D-QUIC은 QUIC에 비해 최대 52%까지 더 빠른 다운로드 완료 시간을 보였다. 하지만 시간 임계값에 RTT의 변화를 반영하였어도 지터가 없는 환경에서만큼 좋은 성능을 가지진 못하였다.

추후 연구를 통해 적응 속도를 넘어서는 급격한 RTT 변동이 있을 때 발생하는 문제를 해결하고 이를 다중 경로 QUIC까지 확장할 예정이다.

ACKNOWLEDGMENT

This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by Ministry of Education (No. NRF-2018R1A6A1A03025109) and by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1006249).

참 고 문 헌

[1] J. Iyengar, and M.Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” RFC 9000, May 2021. Available: https://www.rfc-editor.org/rfc/rfc9000.html

[2] A. Langley *et al*, “The QUIC transport protocol: Design and Internet-scale deployment,” SIGCOMM: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pp. 183 - 196, 2017.

[3] J. Iyengar, and I. Swett, “QUIC Loss Detection and Congestion Control,” RFC 9002, May 2021. Available: https://www.rfc-editor.org/rfc/rfc9002.html

[4] J. Manzoor, L. Cerdà-Alabern, R. Sadre, and I. Drago, “On the Performance of QUIC over Wireless Mesh Networks,” Journal of Network and Systems Management 28, pp. 1872 - 1901, 2020.

[5] lucas-clemente/quic-go, https://github.com/lucas-clemente/quic-go, October 2022.