http://dx.doi.org/10.7840/kics.2016.41.10.1309

스마트폰 통신성향을 고려한 TCP 개선방안

이 준 엽◆, 김 현 순*, 이 웅 희*, 김 황 남°

TCP Algorithm Improvement for Smartphone Data Transmissions

Joon Yeop Lee◆, Hyunsoon Kim*, Woonghee Lee*, Hwangnam Kim°

요 약

본 논문에서는 저용량 데이터의 전송빈도가 높은 스마트폰의 데이터 통신 성향을 참조하여, 스마트폰의 특성에 알맞은 알고리즘을 제시한다. 스마트폰의 사용량은 최근 비약적으로 늘어났으며, 실시간 지도검색, 대중교통 정보 확인, 게임, SNS 등의 사용 빈도가 높아지게 되었다. 이러한 스마트폰에서 주로 일어나는 저용량 데이터 통신은 TCP의 혼잡회피 단계가 나오기 이전에 데이터 전송이 끝나는 경우가 대부분이므로, 본 논문에서는 혼잡회피가 아닌 다른 TCP 관련 알고리즘을 조정하여 초기 통신 속도의 향상을 구현하였다. 본 논문에서 제시하는 알고리즘은 리눅스의 Quick ACK과 네이글 알고리즘(Nagle's algorithm)의 조절을 통하여 불필요한 지연을 줄이고, 짧은 통신에서도 안정적으로 높은 전송속도를 유지할 수 있도록 TCP를 개선하였다.

Key Words : Data transmission, TCP, Delayed ACK, Quick ACK, Smartphone

ABSTRACT

This paper suggests adjusting TCP for smartphones that often have small size data transmission tendency. Usage of smartphones has been risen dramatically in recent years, including frequent usage of real-time map search, public transportation search, online games, and SNS. Because the small size data transmission ends before the phase of the TCP congestion avoidance, this paper suggests an algorithm that increases the transmission speed ahead of the traffic congestion event. The algorithm reduces unnecessary delay by data size-driven adjustment of the Linux Quick ACK and Nagle's algorithm. Therefore, TCP is improved to maintain a high transmission rate steadily in small data transmission.

Ⅰ. 서 론

TCP 최근 스마트폰의 사용량이 증대됨에 따라, 휴대기기를 통한 데이터 통신의 빈도가 증가하고 있다[1]. 스마트폰의 사용량 증가는, Social Network Service(SNS)이용, 스마트폰 온라인 게임, 실시간 지도와 경로검색 등의 활용 빈도가 비약적으로 높아지게 되었으며, 이러한 어플리케이션들은 대부분 0.5 MB 이하의 저용량 데이터를 불연속 적이고 불규칙한 빈도로 전송하는 통신 성향을 가지고 있다. 실제로 스마트 폰으로 각종 서비스 사용 시에 발생하는 데이터량을 My Data Manager라는 어플리케이션을 활용하여, 직접 각종 서비스를 사용해 보면서 데이터 사용량을 측정 하였다. 그 결과, 사진전송은 0.4 MB, 웹서핑(네이버 메인 화면)의 경우 0.5 MB, 구글지도의 위치검색은 0.41 MB정도였으며, 게임이나 SNS 메시지

※ 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2015R1D1A1A01059151)
◆ First Author : School of Electrical Engineering, Korea University, charon7@korea.ac.kr, 학생회원
° Corresponding Author : School of Electrical Engineering, Korea University, hnkim@korea.ac.kr, 종신회원
* School of Electrical Engineering, Korea University, {gustns2010, tgorevenge}@korea.ac.kr, 학생회원
논문번호 : KICS2016-07-170, Received July 31, 2016; Revised September 9, 2016; Accepted September 12, 2016

등은 그보다 더더욱 적은양의 데이터만을 사용하였다. 또한 스마트폰의 특성상, 사용자가 필요로 하거나 요청할 때만 데이터 전송이 이루어지므로, 통신 빈도도 매우 불규칙하다고 볼 수 있다.

그런데, 현재 널리 쓰이고 있는 TCP통신은, 고용량 데이터의 지속적인 전송시에 가장 큰 영향을 주는 혼잡회피(congestion avoidance) 의 개선에 중점을 두고 개발되고 있다. 혼잡회피 알고리즘은 혼잡상황에서 얼마나 빠르게, 그리고 합리적으로 대역폭을 점유하는지에 그 목적이 있으며, 그 종류로는 가장 널리 쓰이는 new Reno[2], 다수의 스마트 기기에서 사용된 CUBIC[3], 통신 왕복시간(Round Trip Time)을 기반으로 최적화된 Vegas[4] 등의 수많은 알고리즘이 존재한다. 하지만, 이러한 혼잡회피의 개선만으로는 스마트폰에서 주로 보이는 불규칙한 저용량 통신성능을 개선하기는 힘들다. 저용량 데이터 통신은, 매우 짧은 시간 동안에만 통신을 하므로 TCP로 통신할 경우 혼잡회피단계로 넘어가기도 전에 slow-start단계에서 통신이 끝나는 경우가 대부분이기 때문이다[5].

저용량 통신으로 인하여 일찍 통신이 끝나게 될 경우에는, 다양한 혼잡회피 알고리즘이 개입될 여지가 적으므로, 스마트폰 사용자의 체감 통신품질을 향상시키기 위해서는 초기 통신속도를 확보하는 방식으로 알고리즘을 개선해야 한다. 이러한 통신속도 개선은 스마트폰이 내포할 수밖에 없는 이동성 문제에도 대응할 수 있게 된다. 스마트폰은 무선으로 차량, 전철 등의 교통시설 내에서 사용하는 빈도가 높으며, 그만큼 이동성으로 인한 통신 불안정성을 내포하고 있다. 이동성으로 인해 handoff나 통신지역 이탈을 경험할 확률이 높아지므로, 통신시간이 길어지고 지속될수록 사용자의 체감 통신품질은 handoff로 인해 하향될 가능성이 높아지게 된다. 따라서 초기 전송속도를 향상시켜 통신시간 자체를 줄인다면, 이동성으로 인한 통신 불안정이 일어날 확률을 낮출 수 있게 된다. 이러한 통신속도의 증가는 회선의 부담을 증가시킬 수 있다고 생각될 수 있으나, 최근의 LTE, Wi-Fi 망의 보급률을 생각한다면[6], 이러한 소량의 데이터를 기존보다 고속으로 보낸다 하여도 회선에 큰 부담이 되지 않는다. 비록 속도가 빠르다 하여도, 앞서 말하였듯이 저용량 데이터의 전송 빈도가 많기 때문에, 총 데이터 전송량은 적기 때문이다.

본 논문에서는 이러한 배경과 스마트폰의 통신 특성을 고려하여, 초기 전송속도를 개선한 ARI (ACK Reception Inducer) 알고리즘을 제시하려 한다. 본 논문에서 제시하는 알고리즘은 Quick ACK[7]과 네이글 알고리즘(Nagle's algorithm)[8]의 사용방식을 개선하도록 하였으며, 이를 통해 불필요한 지연을 제거하여 통신속도를 향상시키고, 짧은 통신시에도 패킷이 안정적으로 전달될 수 있도록 개선하였다. 이러한 개선점을 통하여 결과적으로 사용자의 체감 통신품질을 향상시키는 것이 가능하리라 생각한다.

Ⅱ. 관련 이론

본 절에서는 ARI 알고리즘이 적용되는 TCP의 작동방식과, 네트워크의 복잡도를 감소시키지만 그만큼 통신 속도를 저하시키는 공통된 특징을 가진 Delayed ACK과 네이글 알고리즘(Nagle's algorithm)의 특징에 대해 서술한다. 또한, 기존의 다른 통신속도 향상 관련 연구와 본 논문에서 제시하는 연구가 어떠한 차이가 있는지를 서술한다.

2.1 TCP 의 Slow-start와 혼잡제어

대부분의 데이터 통신에서는, 데이터가 수신자에게 확실하게 전달하기 위해 신뢰할 수 있는 전송 프로토콜인 TCP 방식으로 통신을 한다[9]. TCP에서는 데이터를 받으면 응답인 ACK을 보내게 되는데, 이 ACK이 오기 전에 얼마만큼의 데이터를 미리 연속적으로 보내두는지를 Congestion Window (CWND)의 크기를 통해 결정하게 된다. 결과적으로 CWND의 크기는 전반적인 통신 속도를 결정하게 되며, 지나치게 CWND를 크게 하면 회선복잡도가 늘어나는 부작용을 가져오게 된다. 따라서 혼잡제어 알고리즘을 통하여 CWND를 적정량만큼 증가시키는데 그 종류로는 통신이 시작한 후에 빠른 속도로 CWND를 증가시키는 Slow-start방식과 한번 이상의 패킷 손실이 일어난 이후에 작동하는 방식인 혼잡회피 방식이 있다. Slow-start 방식은 앞서 말하였듯이 통신이 시작한 직후부터 작동하며, CWND의 크기가 한번의 ACK이 올때마다 1씩 상승하도록 되어있어서 매우 빠르게 CWND가 커지도록 하는 방식이다. 하지만 이렇게 커지다보면 회선의 허용 대역폭을 초과하거나, 다른 통신과의 충돌로 인하여 패킷손실이 일어날 확률이 높아진다. 이를 방지하기 위해 threshold라는 한계 값을 초과하여 CWND가 커지거나, 한번 이상의 패킷손실이 발생하게 되면, TCP 에서는 혼잡회피 방식을 사용하게 된다. 혼잡회피는 ACK이 올 때마다 1/CWND만큼 CWND의 크기를 상승시키므로, 느리게 CWND의 크기를 상승시키지만 그만큼 네트워크의 혼잡도를 크게 올리지 않으면서, 통신 속도 향상시킬 수 있다.

2.2 Delayed ACK과 Quick ACK

Delayed ACK은 2개 이상의 ACK이 모이거나, 일정시간 이상 ACK의 생성을 기다린 다음에 한꺼번에 ACK을 전송하는 알고리즘이다[10]. Delayed ACK은 TCP 통신에서 반드시 발생하는 ACK의 전송 횟수를 줄임으로 써 회선의 혼잡도를 낮출 수 있기에, TCP 통신에서는 기본적으로 항상 Delayed ACK을 사용하도록 되어있다. 대용량 데이터 전송으로 인해 장기적인 통신이 이루어질 시에는 Delayed ACK을 통하여 발생하는 통신량을 줄이는 것이 결과적으로 더 효율적이지만, 스마트폰에서 이루어지는 단시간, 저용량 데이터 전송시에는 이러한 Delayed ACK이 오히려 ACK이 느리게 전달되는 현상을 가져오게 되어서, 전송속도의 저하를 가져오게 된다. 이는 CWND의 크기가 작아 데이터 전송량이 적을 수밖에 없는 통신 초기에도 이러한 문제가 발견되었으며[11], 이를 해결하기 위해 리눅스에서는 Quick ACK 알고리즘을 사용하고 있다. Quick ACK은 최초의 2개의 ACK에 대해서는 Delayed ACK 알고리즘을 적용하지 않고 ACK을 즉시 전송하도록 하는 알고리즘이다[7]. 이를 통하여 초반에 느리게 데이터를 전송할 시에 ACK이 많이 발생하지 않는 상황이어도, 즉시 ACK을 전송자에게 보냄으로 써 통신이 불필요하게 지연되지 않도록 할 수 있다.

2.3 네이글 알고리즘 (Nagle's algorithm)

네이글 알고리즘은 보낼 수 있는 데이터를 바로 패킷으로 만들지 않고, 가급적이면 데이터를 모아서 하나의 패킷에 모두 담아 전송하는 방식이며 이는 그림 1에서 그 방식을 보여주고 있다. 앞서 설명한 Delayed ACK과 마찬가지로 네트워크 회선의 혼잡도를 줄이고, 데이터를 처리하는 프로세서의 부담을 줄이는데 그 목적이 있으며, 이러한 특성 덕분에 모든 TCP에서 기본적으로 사용하도록 되어있다[8]. 하지만 그림 1의 네이글 알고리즘을 켰을 경우와 껐을 경우에서 보이듯이, 네이글 알고리즘을 사용하면 같은 양의 데이터를 전송하여도 전송 시간 자체가 더 지연되어 결과적으로 통신이 느려지는 부작용이 있다[12].

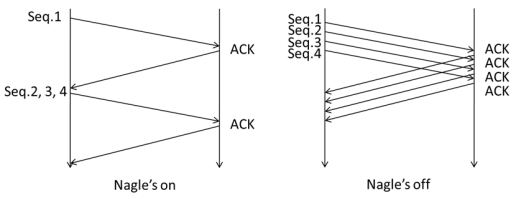

그림 1. 네이글 알고리즘의 통신 과정과 부작용
Fig. 1. Transmission sequence of Nagle's algorithm and side effect

2.4 관련 논문과 연구

기존에도 TCP를 개선하여, 통신속도를 향상시키고자 하는 연구는 다양하게 이루어지고 있었다. [13]에서는 무선 네트워크 통신환경은 그 불안정성으로 인하여, 통신 끊김이 자주 발생하고 있으므로, 이를 보완하기 위해 3 Duplicated ACK 과 Retransmission Timeout 의 빈도를 참조하여 CWND의 증가량과 감소량의 비율을 바꾸는 알고리즘을 제시하고 있다. CWND의 크기의 변동폭을 조정한 또 다른 논문으로는[14,15]이 있으며, 이러한 논문들은 기존의 혼잡회피 알고리즘을 변형하거나 대체하는 알고리즘을 제시하고 있으며, 짧은 통신 보다는 장기간 통신이 이루어지는 동안 통신 환경의 변화를 감지 후 CWND의 크기 조정에 반영하여 통신 속도를 조절하는데 초점이 맞춰져 있다. 그러나 ARI 알고리즘은 통신 초기의 불필요한 지연을 제거함으로 써 짧은 통신이 자주 이루어져도 통신성능이 향상시키도록 하고 있으며, 기존의 통신기기가 어떠한 TCP 혼잡회피 알고리즘을 사용하여도 상관없이 동작한다는 특징을 가지고 있다. ARI 알고리즘처럼 통신 초기의 속도를 상승시키는 논문으로는 [16]이 있으며, 이 논문에서는 Slow-start 단계에서 Round Trip Time을 기반으로 계산된 높은 값의 CWND를 사용하는 알고리즘을 제시하고 있다. 하지만 이러한 CWND 조정 방식은 통신속도의 향상을 가져올 수는 있으나, 혼잡제어와 관계없는 Delayed ACK과 네이글 알고리즘으로 인한 통신지연은 막을 수가 없다. 이러한 지연으로 인하여 통신시간의 편차가 커지게 되면, 이는 사용자의 통신 체감 품질을 저하시키는 원인이 될 수 있다. 반면 ARI 알고리즘은 앞서 언급한 통신지연을 제거함으로 써, 저용량 데이터 전송시 예상보다 과도하게 통신시간이 길어지는 현상이 없다는 특징을 가지고 있다.

Ⅲ. ARI 알고리즘 설명

본 절에서는 2절에서 설명한 배경을 바탕으로 TCP에서 속도조절에 영향을 주는 요소와 그 영향에 대해 분석하여 ARI 알고리즘의 구성요소가 선택된 배경을 설명하고, 리눅스의 Quick ACK과 TCP 관련 알고리즘을 활용하여 단기통신 속도를 향상시키는 ARI 알고리즘의 구조와 그 효과에 대해 서술한다.

3.1 TCP 파라미터를 활용한 통신속도 조절 방법

단기적인 통신에서 전송속도를 높이는 방법으로는 여러 가지 방법이 있을 수 있으나, Slow Start과정을 변경하면 속도 증가 폭이 너무 크고, 이로 인해 다른 사용자의 통신품질을 크게 해칠 우려가 있다. 혹은 최초 CWND의 크기를 크게 하여 통신을 시작할 수도 있으나, 이는 회선의 통신품질을 전혀 고려하지 않기 때문에 회선의 복잡도만 늘리는 결과를 가져올 수도 있다. 따라서 혼잡회피가 작동되지 않을 정도의 짧은 통신에서 전송속도를 향상시키기 위해서는, 데이터전송이 성공하였음을 나타내는 ACK을 기반으로 전송속도 컨트롤을 해야 회선의 통신품질을 고려할 수 있다. 송신자가 ACK을 자주 받도록 함으로써 그에 따른 CWND의 크기가 빠르게 늘어나도록 유도한다면 통신 회선의 복잡도를 고려 할 수 있으므로, 타 사용자의 통신을 침해하지 않으면서 전송속도를 올릴 수 있다[17]. 이러한 특징을 만족시키기 위해, 본 논문에서 제시하는 알고리즘은 리눅스의 Quick ACK과 TCP의 네이글 알고리즘(Nagle's algorithm)을 조정하여, 같은 양의 데이터를 전송하더라도 결과적으로 ACK을 더 빠르게 받을 수 있도록 유도하였다.

3.2 적용방법과 효과

ARI 알고리즘은 먼저 Delayed ACK의 발동을 막는 Quick ACK의 적용범위를 확장하였다. Quick ACK의 적용횟수인 Quick_ACK_count를 계산하기 위해서는, 일단 전송 횟수가 가장 많은 저용량 데이터의 크기(most_sent_data_size)값을 데이터 전송 단위인 segment의 크기(segment_size)로 나누어 총 몇 개의 segment를 전송하게 되었는지를 계산한다. 이렇게 얻어진 segment 개수의 2배값을 Quick_ACK_count로 적용함으로 써 대부분의 저용량 데이터 통신시에는 항상 Quick ACK이 적용될 수 있도록 유도할 수 있다. 그리고 송신자도 바로 데이터를 보내어 ACK이 생성될 수 있도록 유도하고, 이렇게 생성된 다량의 ACK을 지연 없이 보내기 위하여 TCP에서 기본적으로 사용되는 네이글 알고리즘(Nagle's algorithm)을 본 알고리즘에서는 작동하지 않도록 설정 하였다. 이렇게 적용된 알고리즘을 통해 통신 초기에 발생할 수 있는 Delayed ACK과 네이글 알고리즘의 지연이 사라짐으로 써, 언제나 안정적으로, 빠른 통신이 가능해진다. 또한, Quick ACK을 활용하여 알고리즘을 구성하면, 만일 장기간의 통신이 이루어져서 Quick ACK의 적용범위를 벗어나는 경우에는 자연스럽게 Delayed ACK이 적용되므로, 기존의 TCP 통신과 동일하게 네트워크 혼잡도를 줄이는 방식으로 통신을 진행하는 덕분에 통신회선의 부담을 줄일 수 있다. 본 알고리즘의 의사코드는 다음과 같다.

Algorithm 1 ARI

```
1: int Quick_ACK_count
     = most_sent_data_size / segment_size * 2
2: Nagle's algorithm (False)
3: while (data_sending = true)
4:       if (count =< Quick_ACK_count)
5:             Quick_ACK(segment)
6:             count = count +1
7:       else
8:             Delayed_ACK(segment)
9: end
```

TCP 통신에서 Delayed ACK은 두 번째 ACK의 생성이 지연될 경우, Delayed ACK timeout 만큼의 시간을 기다린 다음에 ACK을 전송하게 된다. 이 Delayed ACK timeout은 윈도우는 200ms로 고정되어 있고, 리눅스는 기본값 40ms에서 통신 상태에 따라 가변적으로 변하도록 되어있다. 0.5MB 이내의 저용량 통신이 채 500ms가 걸리지 않는다는 점을 감안하면, Delayed ACK timeout으로 인한 시간손실은 매우 크다고 볼 수 있다. 또한 Delayed ACK timeout 값까지 기다리지 않더라도, Delayed ACK은 그 특성상 두 번째 ACK을 기다리는 만큼의 지연이 반드시 발생하게 된다. 따라서 Quick ACK을 초기 통신시에 여러 번 적용하는 ARI 알고리즘은 Delayed ACK으로 인한 지연을 해결할 수 있으며, 네이글 알고리즘(Nagle's algorithm)에서 최대 전송단위 (Maximum transmission unit) 가 모일 때 까지 기다림으로 인해서 발생되는 지연도 제거되므로, 즉시 ACK 전송이 가능해진다. 이렇게 빠르게 받아진 ACK은 결과적으로 CWND의 빠른 성장을 유도하여 통신 속도 향상효과를 나타내게 된다. 통신이 필요할 때, ARI 알고리즘은 Quick ACK의 적용범위를 변경한 다음, 네이글 알고리즘을 사용하지 않도록 TCP 소켓을 설정하여 통신을 시작한다. 특히 Quick ACK은, 앞서 언급하였듯이 TCP 데이터 전송이 시작할 때 사용되는 알고리즘이다. 따라서 TCP 통신 소켓이 열린 순간부터 즉시 ARI 알고리즘의 효과가 나타나게 되므로, 채 0.1초가 걸리지 않는 지극히 짧은 저용량 통신에서도 ARI 알고리즘은 항상 적용된다.

저용량 통신시의 성능향상이 ARI 알고리즘의 목적

이므로, 각각의 어플리케이션에서 전송받는 데이터의 크기와 빈도를 history로 저장한 다음, 어플리케이션의 통신 성향에 따라 ARI를 유동적으로 적용한다면 회선의 복잡도를 높이지 않으면서 ARI를 사용할 수 있게 된다. 네이글 알고리즘을 사용하지 않으면 데이터 전송속도를 상승시킬 수 있지만, 장기간 데이터 전송을 하게 되면 프로세스와 통신회선의 부담을 증가시키게 된다. 따라서 history를 참조하여 스트리밍이나 클라우드 서비스처럼 대용량 통신이 자주 사용되는 어플리케이션일 경우에는 네이글 알고리즘만은 다시 사용하도록 변경함으로 써, 회선의 부담을 줄일 수 있다. 또한 most_sent_data_size값은 자주 전송받는 데이터 크기 값을 미리 입력해두고 사용할 수도 있으나, history를 참조하여 가장 자주 전송받는 데이터 크기를 적용시키도록 한다면 ARI 알고리즘이 적용되면서도 일시적인 대용량 통신이 발생할 때 Quick ACK이 너무 많이 적용되어 ACK이 지나치게 많이 발생하는 일을 막을 수 있게 된다.

Ⅳ. 성능 평가

본 논문에서 제시한 ARI 알고리즘은 스마트폰의 통신을 개선하기 위하여 개발되었으므로, 그 성능을 입증하기 위하여 실제 스마트폰에 ARI 알고리즘을 적용하여 실험하였다. 본 절에서는 스마트폰의 커널(Kernel)코딩을 통하여 고속통신에서 발생할 수 있는 프로세싱으로 인한 지연을 줄였고, 다양한 통신실험 진행을 위해 Odroid를 활용하여 구성한 무선통신 실험 환경에 대하여 설명하고, 측정된 전송속도와 전송시간을 기존의 TCP와 비교하여 해설하였다.

4.1 실험 환경

본 알고리즘의 성능을 입증하기 위해, 스마트폰 기기인 갤럭시 넥서스를 활용하여 실제 통신성능을 입증하였다. Quick ACK을 포함한 TCP관련 파라미터는 운영체제와 응용프로그램 수행의 핵심에 해당되는 커널(Kernel)에 구현되어 있다. TCP 파라미터 자체는 응용 프로그램(application)단계에서 조절할 수도 있으나, 커널을 직접 변경하여 알고리즘을 조정하면 모든 응용프로그램에 자연스럽게 적용되면서, 컴퓨터 내부에서의 처리시간도 줄어들게 되어 매우 빠른 고속통신이 이루어지는 상황에서도 지연 없이 알고리즘이 적용된다는 장점이 있다. TCP 통신에서 사용하는 segment 크기의 기본값은 536B이고 자주 사용되는 간단한 응답 메시지의 크기가 2 KB 이내인 점을 감안하여, 단순 메시지는 언제나 즉시 전송되고 ACK을 받아볼 수 있도록 하기 위해 Quick ACK의 적용 값을 기존 값인 2개의 4배까지 적용되도록 조정하였다. 이렇게 Quick ACK의 파라미터를 변경한 커널을 갤럭시 넥서스에 탑재하였고, 네이글 알고리즘(Nagle's algorithm)이 적용되지 않도록 설정한 상태에서 데이터를 중계하는 Access Point (AP) 를 거쳐서 Odroid와의 저용량 데이터 통신을 다수 실행하도록 실험환경을 구성하였으며 이는 그림2에서 보여주고 있다. Odroid는 소형 컴퓨터 기기로, linux OS구동이 가능하여 다양한 실험에 활용할 수 있는 기기이며, 현재의 통신상황을 파악하고, 통신환경을 쉽게 조절하기 위해서 이러한 소형 기기와 스마트폰 간의 통신을 구성하였다. 이렇게 실제 무선기기를 활용하여 현실적인 실험환경 구축하였으며, Odroid내부에 자동으로 실험을 진행하고 기록하는 스크립트를 작성하여 이러한 현실의 환경에서도 다수의 실험을 진행할 수 있도록 하였다. 실험은 저용량 데이터 통신을 각 데이터 용량별로 30번씩 실행하였으며, 이때의 전송시간과 전송속도를 측정하였다. 앞서 언급한 저용량 데이터는 실제 스마트폰을 통해 할 수 있는 사진전송, 웹서핑, 지도검색 등의 활동을 고려하여, 0.1 MB, 0.3 MB, 0.5 MB, 1 MB의 데이터를 각각 전송하도록 하였다. 또한 대조군으로 ARI가 적용되지 않은 기존의 TCP 방식의 커널이 탑재된 스마트폰으로도 동일한 방식으로 실험을 진행, 전송속도와 전송시간을 측정하여 결과 값을 서로 비교하였다. 이러한 실험은 갤럭시 넥서스에서 기본으로 사용하고 있으면서 모바일 환경에서 널리 쓰이는 TCP CUBIC 알고리즘과, 가장 기본적인 TCP 알고리즘인 TCP Reno를 기반으로 ARI알고리즘을 적용하여 실험하였다. 이를 통하여 ARI 알고리즘이 TCP 알고리즘의 종류에 관계없이 통신속도 향상을 보이는지를 확인하였다.

그림 2. 무선 통신 실험환경
Fig. 2. Wireless communication experiment environment

4.2 실험 결과

표 1과 표 2 에서는 각 데이터 크기별로 30회의 반복실험을 통해서 얻어진 전송시간의 평균값을 보여주

고 있다. 그리고 표 3과 표 4에서는 실험에서 측정한 전송속도 값의 평균값을 나타내고 있으며, 각각의 표에는 CUBIC과 Reno에 ARI 알고리즘이 적용되었을 때 기존대비 몇 %의 전송시간 감소와 통신속도 향상이 이루어졌는지를 Rate 항목에서 보여주고 있다. 그림 3, 4는 0.1 MB, 0.3 MB, 0.5 MB, 1 MB의 데이터를 각각 TCP CUBIC과 TCP Reno를 기반으로 통신하였을 때의 전송속도를 측정하여 나타낸 그래프이며 총 30번의 통신에서 측정된 전송속도를 나타내고 있다. 실험 대상이 저용량 데이터이니 만큼, 미세한 값의 차이와 변동을 보여주고 있지만, ARI 알고리즘을 통해 개선된 TCP에서는 전반적으로 높은 통신속도를 보여주고 있다. 특히 각 그림에서는, 기존의 TCP통신 방식으로 통신할 경우에는 일시적으로 통신 속도가 대폭 저하되는 현상이 발견되고 있는데, 일정한 크기의 데이터를 전송하는 본 실험의 특성상, 지연으로 인하여 전송시간이 길어지게 되면 그만큼 통신속도가 낮게 계산된다. 그림 3, 4의 결과에서 확인할 수 있듯이, ARI 알고리즘을 사용하여 Delayed ACK과 네이글 알고리즘으로 인한 지연을 제거한 통신에서는 급격한 통신속도 저하 현상이 발생하지 않았으며, 이는 ARI 알고리즘이 적용되면 불필요한 지연이 발생하지 않는다는 점을 증명하고 있다.

표 1. CUBIC 기반 실험의 평균 전송시간 비교 (sec)
Table 1. Comparison of average transmission time in CUBIC based experiment (sec)

Data size	TCP CUBIC	ARI-CUBIC	Rate
0.1 MB	0.107	0.100	-6.5%
0.3 MB	0.267	0.207	-22.5%
0.5 MB	0.367	0.300	-18.3%
1 MB	0.840	0.583	-30.5%

표 2. Reno 기반 실험의 평균 전송시간 비교 (sec)
Table 2. Comparison of average transmission time in Reno based experiment (sec)

Data size	TCP Reno	ARI-Reno	Rate
0.1 MB	0.103	0.100	-2.9%
0.3 MB	0.337	0.217	-35.6%
0.5 MB	0.500	0.330	-34%
1 MB	1.133	0.630	-44.4%

표 3. CUBIC 기반 실험의 평균 전송속도 비교 (Mbits/sec)
Table 3. Comparison of average throughput in CUBIC based experiment (Mbits/sec)

Data size	TCP CUBIC	ARI-CUBIC	Rate
0.1 MB	12.043	14.153	17.5%
0.3 MB	12.438	14.133	13.6%
0.5 MB	12.474	14.903	19.5%
1 MB	12.756	14.350	12.5%

표 4. Reno 기반 실험의 평균 전송속도 비교 (Mbits/sec)
Table 4. Comparison of average throughput in Reno based experiment (Mbits/sec)

Data size	TCP Reno	ARI-Reno	Rate
0.1 MB	10.330	14.193	37.4%
0.3 MB	10.022	13.496	34.7%
0.5 MB	9.692	12.536	29.3%
1 MB	10.541	13.477	27.9%

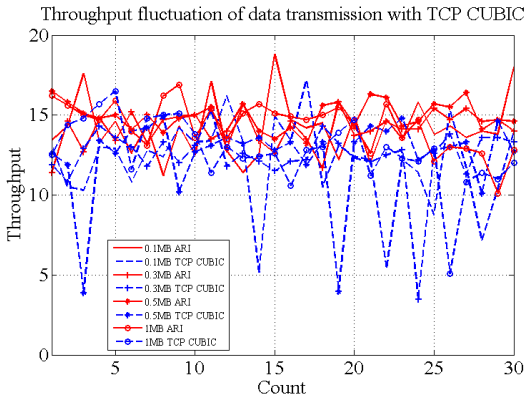

그림 3. TCP CUBIC을 통한 데이터 전송 시의 전송속도 변화
Fig. 3. Throughput fluctuation of data transmission with TCP CUBIC

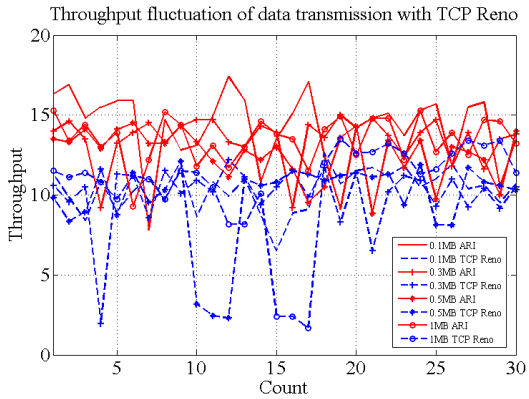

그림 4. TCP Reno를 통한 데이터 전송 시의 전송속도 변화
Fig. 4. Throughput fluctuation of data transmission with TCP Reno

V. 결 론

스마트폰에서 주로 이루어지는 통신 어플리케이션을 볼 때, 대부분의 통신이 혼잡회피 단계까지 진행하는 경우가 적으므로, 혼잡회피가 아닌 TCP의 다른 파라미터를 개선하여 저용량 데이터를 전송하는 짧은 통신에서도 전송속도를 향상시킬 수 있음을 보였다. 본 논문에서 제시하는 ARI 알고리즘은 통신 초기의 데이터 전송 시에 통신 속도 측면에서 약점을 가지고 있는 Delayed ACK과 네이글 알고리즘(Nagle's algorithm)의 적용방식을 기존 TCP와 달리 함으로 써, 불필요한 지연시간을 제거하고 항상 안정적인 통신속도와 통신시간을 확보할 수 있게 하였다. 본 TCP 개선안을 적용한다면, 데이터 전송시간 감축과 전송속도 향상을 통해 스마트폰 사용자의 체감 통신품질을 향상시킬 수 있을 것으로 예상한다.

References

[1] S. Choi and S. Han, "A study on determinants of consumers' choice of mobile data service," *J. KICS*, vol. 40, no. 1, pp. 115-123, 2015.

[2] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 26. no. 4, pp. 270-280, 1996.

[3] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64-74, 2008.

[4] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," *ACM SIGCOMM Commun. Architectures, Protocols Appl.*, vol. 24, no. 4, pp. 24-35, 1994.

[5] Y. Zaki, et al., "Adaptive congestion control for unpredictable cellular networks," in *Proc. ACM Conf. Special Interest Group on Data Commun.*, 2015.

[6] S.-W. Lee, et al., "Empirical analysis of induced demand resulted from LTE service launching," *J. KICS*, vol. 37, no. 8, pp. 741-749, 2012.

[7] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *Proc. PFLDnet*, pp. 1-6, 2008.

[8] S. D. Strowes, "Passively measuring TCP round-trip times," *Commun. ACM*, vol. 56, no. 10, pp. 57-64, 2013.

[9] K. Chae, T. H. Nguyen, M. Park, and S. Jung, "A study on advanced TCP snoop algorithm considering the feature of network layer," in *Proc. KICS Int. Conf. Commun.*, pp. 581-582, 2013.

[10] N. Kim, et al., "A scalable video coding (SVC)-Aware retransmission scheme for multimedia streaming in IEEE 802.11 WLANs," *J. KICS*, vol. 39, no. 2, pp. 95-101, 2014.

[11] J. Chen, et al., "TCP with delayed ack for wireless networks," *Ad Hoc Networks*, vol. 6, no. 7, pp. 1098-1116, 2008.

[12] S. Cheshire, *TCP performance problems caused by interaction between Nagle's algorithm and delayed ACK*(2005), Retrieved Sept., 30, 2016, from http://www.stuartcheshire.org/papers/NagleDelayedAck.

[13] A. Khurshid, M. H. Kabir, and R. Das, "Modified TCP newreno for wireless networks," *NSysS*, pp. 1-6, 2015.

[14] H. Gururaj and B. Ramesh, "Performance analysis of hstcp for optimizing data transfer rate in mobile ad-hoc networks," *Int. J. Comput. Appl.*, vol. 123, no. 15, 2015.

[15] W. Bao, V. W. Wong, and V. Leung, "A model for steady state throughput of tcp cubic," *GLOBECOM 2010*, pp. 1-6, 2010.

[16] Y. Zhang, N. Ansari, M. Wu, and H. Yu, "Afstart: An adaptive fast tcp slow start for wide area networks," *Commun.(ICC)*, pp. 1260-1264, 2012.

[17] M. Allman, S. Flayd, and C. Partidge, *Increasing TCP's initial window*, RFC3390, 1998.

이 준 엽 (Joon Yeop Lee)

2014년 8월 : 고려대학교 전기전자전파공학부(공학사)
2014년 9월~현재 : 고려대학교 전기전자공학과 석박사통합과정
<관심분야> 통신공학, 네트워크공학

김 현 순 (Hyunsoon Kim)

2010년 6월 : University of Illnois, Urbana-Champaign Computer Science 학부 졸업
2016년 8월 : 고려대학교 전기전자공학과 박사 졸업
<관심분야> 통신공학, 컴퓨터공학

이 웅 희 (Woonghee Lee)

2013년 2월 : 고려대학교 방사선학과(보건학사)
2013년 2월 : 고려대학교 전기전자전파공학부(공학사)
2013년 3월~현재 : 고려대학교 전기전자공학과 박사과정
<관심분야> 통신공학, 네트워크공학

김 황 남 (Hwangnam Kim)

1992년 3월 : 부산대학교 컴퓨터공학과(공학사)
1994년 2월 : 서울대학교 컴퓨터공학과(공학석사)
2004년 2월 : 미국 Urbana-Champaign 소재 Illinois 주립대학 컴퓨터과학과(공학박사)
1994년~1999년 : LG 전자 주임연구원
2000년~2001년 : Bytemobile 소프트웨어 엔지니어
2004년~2005년 : 미국 Urbana-Champaign 소재 Illinois 주립대학 Post Doctorate Fellow
2005년~2006년 : 삼성전자 책임연구원
2006년~현재 : 고려대학교 전기전자전파공학부 교수
<관심분야> 통신공학, 네트워크공학, 융합IT, CPS