고속 무선 네트워크에서의 저지연 통신을 위한 TCP 이슈 분석

임희수, 박세웅
서울대학교, INMC

요 약

TCP (Transmission Control Protocol)는 IP (Internet Protocol)와 더불어 인터넷 핵심 프로토콜 중 하나로써, 현재 Web, e-mail, FTP 등 대다수의 애플리케이션에서 사용되고 있는 연결 지향적이고 신뢰성 전송을 보장하는 전송 계층 프로토콜이다. 유/무선 네트워크가 진화함에 따라 TCP 역시 보다 나은 서비스를 제공하기 위해 많은 연구가 이루어졌고, 이를 통해 최근에는 스트리밍과 같이 지연이 중요한 애플리케이션에서도 많이 사용되고 있다. 본 논문은 모바일 데이터가 폭증함에 따라 무선 네트워크에서의 TCP 성능이 큰 관심을 받고 있는 현 시점에서 최근에 이슈가 되고 있는 무선 네트워크에서 bufferbloat문제의 심각성과 이를 해결하기 위한 방안에 대해 다룬다.

Ⅰ. 서 론

TCP (Transmission Control Protocol)[1]는 1960년대 말 인터넷의 시초로 알려져 있는 미 방위고등연구계획국 (DARPA: Defense Advanced Research Projects Agency)의 ARPANET 프로젝트에서 시작하여 1982년에 프로토콜 스펙 (RFC 793)이 최초로 공개된 전송 계층 프로토콜로써, 네트워크 계층 프로토콜인 IP (Internet Protocol)와 더불어 TCP/IP라는 명칭으로도 널리 알려져 있다. 1983년, BSD Unix 4.2에 처음으로 TCP/IP가 지원된 이후, 리눅스 및 윈도우 등 다수의 운영체제에서도 TCP/IP를 전송/네트워크 계층 프로토콜로써 지원해 왔으며, 현재는 사용자들이 이용하는 대다수의 애플리케이션 (HTTP, FTP, E-mail, 스트리밍 등)에서 널리 사용되며 인터넷을 이루는 핵심 프로토콜로써 자리를 잡았다.

TCP는 〈그림 1〉에서 보여지는 것과 같이, 1982년도 프로토콜 스펙 발표 이후, 약 30여년간 유/무선 네트워크 성능 향상을 위해 수 많은 프로토콜이 개발 및 배포되어 왔다. 더욱이, 2013

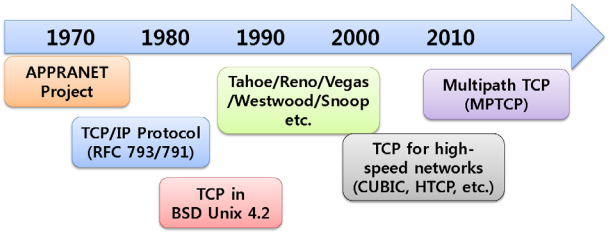


그림 1. TCP 진화 방향

년도에는 기존의 하나의 연결 당 하나의 경로만 지원했던 기존 TCP에서 보다 나은 성능 향상을 위해 다중 경로를 동시에 이용할 수 있게 TCP를 확장한 MPTCP (MultiPath TCP) [2]가 발표되었으며, 현재 리눅스 및 안드로이드 시스템을 통해 배포가 진행 중이며, 활발하게 연구가 진행되고 있다.

최근에 스마트폰, 태블릿 PC등의 보급으로 인해 모바일 트래픽이 폭증하고 있다. 이에 따라 대다수의 애플리케이션의 전송을 담당하는 TCP의 성능 역시 많은 관심을 받고 있다. 본 논문에서는 기존 무선 네트워크에서의 성능 향상 관점과 최근에 이슈가 되고 있는 bufferbloat 에 개념, 그리고 이를 해결하기 위한 기법들에 대해 다룬다.

Ⅱ. 배경지식

1. TCP 개요

TCP는 두 종단간 연결지향적이고, 신뢰할 수 있는 전송을 보장하며 혼잡 제어 및 흐름 제어 기능을 제공한다. 〈그림 2〉는 TCP 헤더 구조를 나타낸다. 종단 사이의 프로세스 간 전송을 담당하는 TCP는 IP주소와 port 번호를 이용하여 프로세스간 전송을 보장한다. 예를 들어, 웹 브라우징시에는 80번 port를 사용함으로써 서버와 웹 페이지 전송을 보장한다. 이 외에도 잘 알려진 port 번호로는 FTP (21), SSH (22), DNS (53), POP3 (110), HTTPS (443) 등이 있다.

Source port address (16 bits)			Destination port address (16 bits)
Sequence number (32 bits)			
Acknowledgment number (32 bits)			
Header Length (4 bits)	Reserved (4 bits)	Flags (8 bits)	Window (16 bits)
Checksum (16 bits)			Urgent pointer (16 bits)
Options and Padding ((0-320 bits)			

그림 2. TCP 헤더구조

TCP는 프로세스 간 전송 보장 이외에도 수많은 TCP flow가 공존했을 경우 네트워크가 혼잡해지지 않게 전송량을 결정하는 기법인 혼잡 제어 기법을 제공한다. 약 30여년 동안 수많은 TCP 혼잡 제어 기법이 개발되었으며[3] 그 진화 방향은 〈그림 3〉에 나타나 있다. 크게 패킷 손실을 네트워크 혼잡으로 인지하여 전송량을 줄이는 패킷 손실 기반 혼잡 제어 기법과 지연 증가를 네트워크 혼잡으로 인지하여 전송량을 줄이는 지연 기반 혼잡 제어 기법으로 나누어 진다. 현재 Linux에서는 CUBIC이, Windows에서는 Compound 기법이 디폴트 혼잡 제어 기법으로 사용되고 있다.

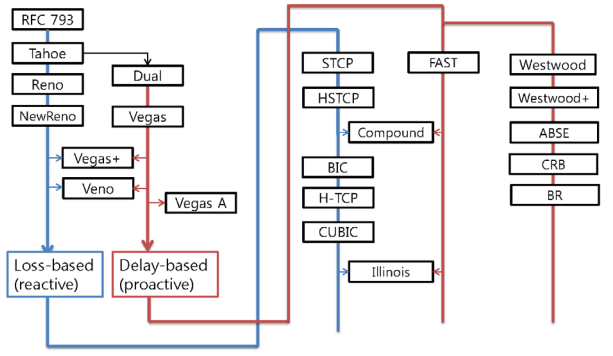


그림 3. TCP 혼잡제어 진화 방향

TCP 흐름 제어는 송신 측과 수신 측의 데이터 처리 속도 차이를 해결하기 위한 기법으로, TCP 수신측에서 처리할 수 있는 데이터를 TCP 헤더의 window 필드에 지정하여 (rwnd: 수신윈도우) TCP 송신측에게 전송한다. 이를 받은 TCP 송신측은 자신이 TCP 혼잡 제어를 통해 결정된 전송량 (cwnd: 혼잡윈도우)과 수신윈도우의 최소값을 자신의 전송량을 결정함으로써, 수신측의 데이터 처리 속도보다 더 많은 데이터를 전송하지 않는다. 예를 들어, 〈그림 4〉에서 TCP 송신측의 혼잡윈도우가 100개의 패킷으로 결정이 되었지만, TCP 수신측이 보낸 수신윈도우의 값이 70개의 패킷일 경우, 실제 TCP 송신측은 70개의 패킷만 전송할 수 있는 것이다.

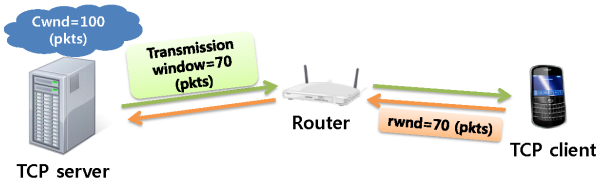


그림 4. TCP 흐름 제어

2. 무선 네트워크에서의 TCP 연구 개요

무선 네트워크가 진화함에 따라 TCP 역시 그에 맞게 진화되어 왔다. 〈그림 3〉에서 보이는 TCP Westwood [5]가 그 대표적이 예이다. TCP Westwood는 종단 대 종단 (end-to-end) 대역폭을 TCP 송신단에서 추정하고 이에 따라 혼잡윈도우 값을 결정하는 기법을 제안한다. 이는 네트워크 혼잡에 의한 패킷 손실과 채널 에러로 패킷 손실을 구분하기 위한 방안으로, 높은 링크 활용 (link utilization)을 위해서는 채널 에러로 인한 랜덤 손실 발생시 혼잡윈도우를 줄일 필요가 없기에 종단 대 종단 대역폭을 추정함으로써 패킷 손실과 혼잡에 의한 패킷 손실을 구분하려는 것이다.

무선 네트워크에서의 TCP 성능 향상을 위해서 고안된 TCP Snoop [6]은 기지국 단에서 링크단 재전송을 통해 채널 에러로 인한 패킷 손실을 TCP 송신단에 숨기는 기법이다. 이를 위해, 기지국은 TCP 패킷을 캐싱하고 수신단에서 DUPACK을 전송할 시, DUPACK은 기지국단에서 유실함으로써 송신단에게 손실 사실을 숨기고, 링크 재전송을 통해 이를 복구하는 기법이다.

Split TCP [7]은 무선 네트워크에서의 TCP 성능 극대화를 위해 기지국 단에서 TCP 연결을 분할하는 기법을 제안한다. 즉, 기지국과 TCP 송신단이 하나의 TCP 연결을 유지하고, 기지국과 TCP 수신단이 하나의 TCP 연결을 유지하는 것이다. 무선 네트워크 환경을 고려하여 기지국단에서 무선 TCP등을 최적화하여 사용할 수 있는 프레임워크를 제안하였다.

이와 같이 무선 네트워크에서 TCP 성능 향상을 위해 많은 연구가 진행되어 왔다. 하지만 주로 높은 링크 활용을 위해 혼잡에 의한 패킷 손실과 채널 에러로 인한 패킷 손실을 구분함으로써 불필요한 전송량 감소를 막는 것이 지금까지의 접근 방법이었다면, 최근에는 그 시각이 변하였다. LTE와 WiFi 네트워크가 제공하는 링크 단 재전송 기법의 발달로 인해 무선 단에서 채널 에러로 인한 패킷 손실이 크게 줄어들었고, 기존의 혼잡에 의한 패킷 손실과 채널 에러로 인한 패킷 손실을 구분하려는 움직임 보다는 사용자가 겪게 되는 딜레이를 줄이기 위한 연구가 활발하게 진행되고 있다.

Ⅲ. 무선 네트워크에서의 TCP 성능 저하 및 해결법

1. Bufferbloat

최근 무선 네트워크에서는 네트워크를 이루는 요소 (라우터, 기지국, 단말 등)의 버퍼 사이즈가 커짐에 따라 발생하는 불필요한 지연 증가 문제인 'bufferbloat' 문제가 이슈가 되고 있다. 이는 TCP 서버가 버퍼 오버플로우로 인한 패킷 손실이 발생하지 않는다면 전송률을 계속 증가시키는 패킷 손실 기반의 TCP 혼잡 제어 기법으로 인해 더 심화될 수 있고, delay-sensitive한 애플리케이션 사용자의 QoE (Quality of Experience)를 저하시킨다는 점에서 큰 이슈가 되고 있다[8]. Bufferbloat은 크게 downstream bufferbloat과 upstream bufferbloat으로 구분될 수 있다. 전자의 경우, 셀룰러 네트워크의 기지국 (LTE eNodeB) 혹은 Wi-Fi AP에 과도하게 많은 패킷이 쌓임으로써, delay-sensitive한 애플리케이션의 패킷이 불필요한 지연을 겪는 현상을 의미한다. 후자의 경우, 단말 단에서 외부 서버로 많은 양의 데이터를 업로드 하는 경우, 단말 내의 버퍼에 불필요하게 많은 데이터가 쌓이게 됨으로써, delay-sensitive한 애플리케이션의 패킷이 지연을 겪게 되는 현상을 말한다. 예를 들어, downstream bufferbloat의 경우 〈그림 5〉에서 외부 서버로부터 파일 전송 애플리케이션을 실행시키고 있는 TCP 수신측이 온라인 게임이나 모바일 메신저와 같은 delay-sensitive한 애플리케이션을 동시에 사용하였을 경우, LTE eNodeB나 WiFi AP에 미리 쌓여있던 파일 전송 애플리케이션의 패킷들로 인해 delay-sensitive한 애플리케이션의 패킷이 불필요한 지연을 겪게 된다.

2. 무선 네트워크에서의 Bufferbloat의 심각성

무선 네트워크에서의 downstream/upstream bufferbloat 문제의 심각성을 확인하기 위해 본 논문에서는 실제 실험을 통해 불필요한 지연 증가가 얼마만큼 나타나는지 확인해보았다. 실험을 위해 ubuntu 12.04가 설치되어 있는 외부 서버 삼성 갤럭시 S2 (모델명 E120K)를 각 종단으로 설정하였으며, iperf를 통해 데이터를 upstream/downstream 방향으로 발생시켜 수율과 round trip time (RTT)을 측정하였다. 자세한 실험환경은 〈표 1〉에 나타나 있다.

표 1. 실험환경

TCP server	Ubuntu 12.04 LTS (Kernel 3.2.0-54) Intel(R) Xeon(R) CPU E3-1270 V2 (3.50GHz) Congestion control: CUBIC with the timestamp option on
Smartphone	Samsung Galaxy S2 (Model E120K) OS: Kernel version 3.0.8, Android Icecream Sandwich (4.0.3) tcp_rmem_max = 2,560,000 Bytes tcp_wmem_max = 2,560,000 Bytes
LTE networks	Operator: Korea Telecom (KT) Bandwidth: 100 Mbps
Wi-Fi networks	Access Point: Qualcomm Atheros AR93xx Bandwidth: 54 Mbps (802.11g)

우선, downstream bufferbloat의 심각성을 확인하기 위해 TCP server로부터 iperf를 통해 트래픽을 다운로드 받는 환경을 고려하였다. 실험 독립변수로는 tcp_rmem_max (수신윈도우의 최대값을 제한해주는 시스템 파라미터) 값을 196,608 ~ 2,560,000 Bytes로 변화시켜 가며 실험을 진행하였다. 〈그림 6〉은 tcp_rmem_max 값이 커질수록, 수율은 더 이상 증가하지 않지만 RTT는 증가하는 것을 보여준다. 즉, 불필요한 지연을 보여주는 downstream bufferbloat이 실제 LTE 및 Wi-Fi 네트워크에서 발생한다는 것을 보여준다. 이는 앞서 설명한

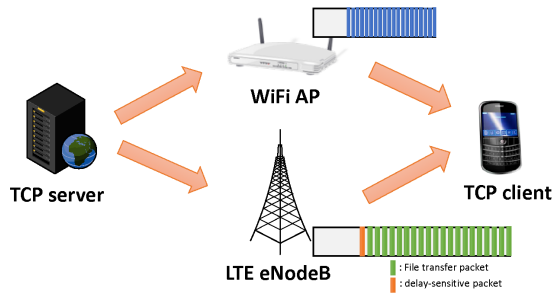


그림 5. Bufferbloat 개념도

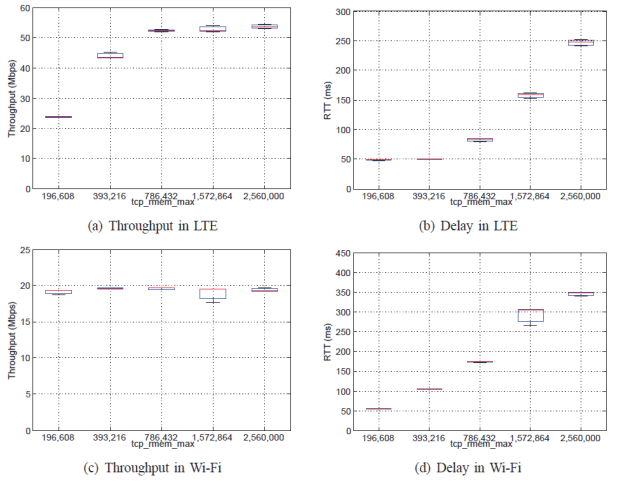


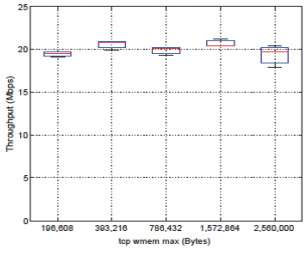
그림 6. Downstream Bufferbloat의 심각성

것과 마찬가지로 불필요하게 LTE eNodeB와 Wi-Fi AP에 많은 양의 패킷이 쌓여있기 때문에 발생하는 현상이다. 본 논문의 실험 환경에서는 tcp_rmem_max값이 LTE네트워크에서는 400 ~ 800KByte에서 Wi-Fi 네트워크에서는 200Kbytes 이하에서 최적의 수율 및 RTT 성능을 보여준다는 것을 확인할 수 있다.

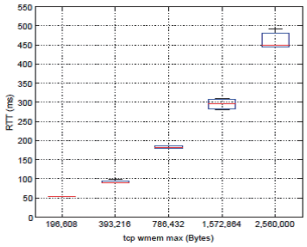
Downstream bufferbloat과 달리 upstream bufferbloat을 확인하기 위해 이번에는 LTE 네트워크에서 단말이 외부 서버로 iperf를 통해 데이터를 업로드 하는 환경을 고려하였다. Downstream bufferbloat 실험과 유사하게 이번 실험에서는 tcp_wmem_max (혼잡윈도우의 최대값을 제한해주는 시스템 파라미터) 값을 196,608 ~ 2,560,000 Bytes로 변화시켜 가며 수율 및 RTT를 측정하였다. 〈그림 7〉은 tcp_wmem_max 값이 커질수록, 수율은 더 이상 증가하지 않지만 RTT는 증가하는 것을 보여준다. 실제 LTE네트워크에서 단말에 불필요하게 많은 양의 패킷이 쌓여임으로 인해 upstream bufferbloat이 발생한다는 것을 보여준다. 본 논문의 실험 환경에서는 tcp_wmem_max값이 200Kbytes 이하에서 최적의 수율 및 RTT 성능을 보여준다는 것을 확인할 수 있다.

Downstrea/upstream bufferbloat은 위와 같이 실제 네트워크에서 흔하게 관측되는 현상으로 네트워크 별 최적의 tcp_rmem_max/tcp_wmem_max값이 설정된다면 RTT 증가를 그리 크게 경험하지 않을 수 있지만 네트워크 상황이 실시간으로 변하는 실제 환경을 고려한다면 최적의 tcp_rmem_max/tcp_wmem_max 값을 설정하는 것은 그리 쉬운 일이 아니다.

다음 절에서는 무선 네트워크에서 bufferbloat을 해결하기 위한 다양한 기법에 대해 다룬다.



(a) Upload throughput



(b) Upload RTT

그림 7. Upstream Bufferbloat의 심각성

IV. 무선 네트워크에서 Bufferbloat 해결을 위한 기법

1. Downstream Bufferbloat을 위한 기법

무선 네트워크에서의 downstream bufferbloat 문제를 해결하기 위해 많은 기법들이 개발되었는데, 기법이 동작하는 위치에 따라 크게 3가지로 분류된다. 첫째, 서버단 기술은 패킷 손실 기반 혼잡 제어 기법인 아닌 지연 기반 혼잡 제어 기법을 사용하는 것이다. TCP-Vegas[9], Fast TCP[10]등이 대표적인 지연 기반 혼잡 제어 기법이다. 하지만 지연 기반 혼잡 제어 기법은 손실 기반 혼잡 제어 기법과 공존하였을 경우 수율 저하라는 문제점을 초래한다는 단점을 지니고 있다[11]. 둘째, 라우터에서 특정 조건에 따라 패킷을 드랍시키는 active queue management (AQM) 기법을 사용하는 것이다. 패킷을 드랍시킴에 따라 TCP 서버가 전송량을 줄이게 유도할 수 있고, 결과적으로 라우터의 큐 길이를 적정 수준으로 유지할 수 있다는 장점이 있다. 대표적인 AQM 기법으로는 random early detection (RED) [12], exponential RED (E-RED) [13], random exponential marking (REM) [14]등이 있다. 하지만 복잡한 파라미터 설정과 모든 라우터를 업그레이드 해야한다는 deployment 이슈가 남아 있다. 최근에는 파라미터 설정을 간소화한 CoDel [15]이 제안되었지만, 이 역시도 마지막으로 실제 네트워크에 설치되어야 한다는 단점을 지니고 있다. 마지막으로, TCP 흐름 제어를 통한 수신단 기술이 있다. TCP 흐름 제어를 통해 서버의 전송량을 제어할 수 있으며, 수신단 기법은 모바일 단말의 펌웨어 업데이트를 통해 손쉽게 배포될 수 있다는 장점을 가지고 있다. 하지만 TCP-Vegas와 유사하게 경쟁 상황에서의 수율 저하라는 문제점을 지니고 있다. Downstream bufferbloat에 대한 대응 방법은 〈표 1〉에 정리되어 있다.

2. Upstream Bufferbloat을 위한 기법

Downstream bufferbloat해결을 위해서는 앞서 설명한 것처럼 송신단, 라우터, 수신단 기법으로 나누어 진다. 이와 달리 upstream bufferbloat 문제를 해결하기 위한 기법은 크게 queue discipline (qdisc)과 네트워크 단 기술로 분류된다. 첫째, 데이터를 발생시키는 단말단에서의 qdisc를 현재의 단일 전송 큐 구조가 아닌 다수의 전송 큐 구조로 변경하는 것이다. 대표적인 예로는, SFQ [16], WFQ [17], CBQ [18], AF [19], CBA [20] 등이 있다. 이들은 구체적으로 fair queueing [16][17][18]과 priority queueing [19][20]으로 나누어질 수 있다. Fair queueing의 경우 다수의 전송 큐를 각 TCP flow에 할당함으

표 2. Downstream bufferbloat 해결책

분류	대표 기법	특징
서버단 기술	TCP Vegas [9], FAST TCP [10]	패킷 손실 기반이 아닌 지연 기반의 TCP 혼잡 제어 기법 패킷 손실 기반의 TCP와 경쟁시 수율 저하
라우터 기술	RED [12], E-RED[13], REM [14],CoDel[15]	라우터에서 특정 조건에 따라 패킷을 드랍시킴에 따라 큐 길이를 적절하게 조절할 수 있음 파라미터 설정의 복잡성 및 라우터 업그레이드 필요
수신단 기술	DRWA [8]	TCP 흐름 제어를 통해 서버의 전송률 제어 펌웨어 업데이트를 통한 배포의 용이성 다른 TCP와의 경쟁시 수율 저하

로써 flow별 공평한 자원 할당을 목적으로 한다. 하지만 flow별 불필요하게 많은 메모리를 할당 해야하는 문제점과 복잡한 세부 설정문제를 지니고 있다. 이와 달리 priority queueing download TCP ACKs를 upstream 데이터 패킷보다 더 높은 priority를 주어 download 트래픽의 성능향상을 목적으로 제안된 기법이다. 하지만 download 트래픽의 성능 향상에만 초점을 맞추고 있기 때문에 interactive한 애플리케이션이 많이 사용되는 현 시점에서는 패킷 분류기법과 같이 사용되어야 한다는 한계점을 지니고 있다. 이와 달리 proxy 기반의 기법은 네트워트 중간에 위치한 proxy 서버가 인위적으로 모바일 단말의 upload 전송률을 제어함으로써 불필요한 지연을 발생시키지 않게 하는 기법이다. 하지만 이 역시도 scalability 측면에서의 문제점을 지닐 수 있으며 추가적인 연구가 필요하다. Upstream bufferbloat에 대한 대응 방법은 〈표 2〉에 정리되어 있다.

표 3. Upstream bufferbloat 해결책

분류	대표 기법	특징
Fair Queueing	SFQ[16], WFQ [17], CBQ [18]	다수의 queue를 각 TCP flow에 할당함으로써 공평한 자원 할당 복잡한 설정 기법
Priority Queueing	AF [19], CBA [20]	Download 성능 향상을 위해 download TCP ACK만을 먼저 처리 패킷 분류기법이 같이 사용되어야 함
Proxy-based rate control	RFRS [21], RRE [22]	Proxy 서버를 통한 upload 전송률 제어 Deployment와 scalability 문제점

V. 결 론

TCP는 유/무선 통신 기술의 발달 및 인터넷의 보급과 발맞추어 약 30여년동안 활발하게 연구, 개발되어 왔으며 이 과정에서 IP와 더불어 인터넷 핵심 프로토콜로써 자리매김하였다. 모바일 트래픽이 폭증하고, 고속망으로 발달한 무선 네트워크에서의 TCP 성능은 굉장히 중요한 이슈이며, 수율 뿐만이 아니라 지연 역시도 중요한 성능 지표로 주목을 받고 있다.

본 논문에서는 최근에 이슈가 되고 있는 무선 네트워크에서의 bufferbloat 문제를 실험을 통해 직접 확인하고, 이를 해결하기 위한 기법들과 각 기법들의 한계점에 대해 다루었다. Bufferbloat 문제는 기존에 제안된 기법이 가진 한계점을 해결함으로써 보다 나은 모바일 사용자 QoE 향상을 불러일으킬 수 있을 것으로 기대한다.

Acknowledgement

이 논문은 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2012R1A2A2A01046220)

참 고 문 헌

[1] “Transmission Control Protocol,” RFC 793, Sep. 1981.

[2] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, “TCP Extensions for Multipath Operation with Multiple Addresses,” RFC 6824, Jan. 2013.

[3] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP,” IEEE Communications Surveys & Tutorials, vol. 12, no. 3, 2010.

[4] http://www.multipath-tcp.org/

[5] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, “TCP Westwood: Bandwidth estimation for enhanced transport over wireless links,” in Proc. ACM MOBICOM, 2001.

[6] Hari Balakrishnan, Srinivasan Seshan, Elan Amir and Randy H. Katz, Improving TCP/IP Performance over Wireless Networks,” in Proc. ACM MOBICOM, 1995.

[7] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, “A Comparison of Mechanisms for

Improving TCP Performance over Wireless Links," IEEE/ACM Transactions on Networking, vol. 5, no. 6, pp. 756-769, Dec. 1997.

[8] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3G/4G Networks," in Proceedings of ACM IMC, 2012.

[9] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in Proceedings of ACM SIGCOMM, 1994.

[10] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in Proceedings of INFOCOM, 2004.

[11] J. Mo, R. La, V. Anantharam, and J. Walrand, "Analysis and Comparison of TCP Reno and Vegas," in Proceedings of INFOCOM, 1999.

[12] S. Floyd, and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking (ToN), vol. 1, issue. 4, 397-413, Aug. 1993.

[13] S. Liu, T. Basar, and R. Srikant, "Exponential-RED: A Stabilizing AQM Scheme for Low- and High-Speed TCP Protocols," IEEE/ACM Transactions on Networking, vol. 13, no. 5, pp. 1068-1081, Oct. 2005.

[14] S. Athuraliya , S. H. Low, V. H. Li, and Q. Yin, "REM: Active Queue Management," IEEE Network, vol. 15, issue 3, pp. 48-53, May 2001.

[15] K. Nichols and V. Jacobson, "Controlling Queue Delay," ACM Queue, vol. 10, issue 5, pp. 20-34, May 2012.

[16] P. E. McKenney, "Stochastic Fairness Queueing," in Proceedings of INFOCOM, 1990.

[17] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," Internetworking: Research and Experience, vol. 1, pp. 3-26, 1990.

[18] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks," IEEE Transactions on Networking, vol. 3, no. 4, pp. 365-386, Aug. 1995.

[19] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The Effects of Asymmetry on TCP Performance," in Proceedings of Mobicom, 1997.

[20] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions," in Proceedings of SIGMETRICS, 1998.

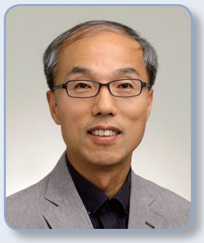
[21] Y. Xu, W. K. Leong, B. Leong, and A. Razeen, "Dynamic Regulation of Mobile 3G/HSPA Uplink Buffer with Receiver-Side Flow Control," in Proceedings of ICNP, 2012.

[22] W. K. Leong, Y. Xu, B. Leong, and Z. Wang, "Mitigating Egregious ACK Delays In Cellular Data Networks by Eliminating TCP ACK Clocking," in Proceedings of ICNP, 2013.

약 력

임 희 수

2008년 서울대학교 공학사
2010년 서울대학교 공학석사
2010년~ 2015년 서울대학교 공학박사
2015년~현재 삼성전자 책임연구원
연구분야: TCP/MPTCP 성능 최적화, 네트워크 프로토콜 설계, P2P 네트워크



박 세 웅

1984년 서울대학교 공학사
1996년 서울대학교 공학석사
1991년 University of Pennsylvania 공학박사
1991년~1994년 AT&T Bell Lab. 연구원
1994년~현재 서울대학교 전기컴퓨터공학부 교수
2009년~2011년 서울대 뉴미디어통신연구소 소장
2014년 IEEE Vehicular Technology Conference TPC Chair
2011년~현재 Journal of Communications and Networks, Co-EIC
IEEE Trans. on Wireless Communications (TWC), Editor
연구분야: 모바일 네트워크, 네트워크 성능분석, 스마트그리드