논문 2013-50-2-5

모바일 플랫폼을 위한 네트워크 환경 측정 시스템 설계 및 구현

(The Design and Implementation of Network Measurement System for Mobile Platforms)

김 강 희*, 여 진 주*, 김 진 혁*, 최 상 방**

(Kanghee Kim, Jinjoo Yeo, JinHyuk Kim, and SangBang Choi)

요 약

모바일 네트워크 사용량이 급증함에 따라 트래픽 수요 문제를 해결하기 위한 많은 연구가 이뤄지고 있다. 특히 네트워크 환경 측정 분야는 정확한 분석을 통해 네트워크상에 발생되는 문제들의 원인을 찾아냄으로써 트래픽 수요 문제를 해결할 수 있는 기반을 제공한다. 특히 최근 스마트폰의 수요가 늘어남에 따라 모바일 플랫폼 특성이 네트워크에 미치는 영향을 고려한 측정시스템이 필요하다. 이에 본 논문에서는 모바일 플랫폼을 위한 네트워크 환경 측정 시스템을 설계하였다. 설계된 시스템은 클라이언트를 통하여 얻은 패킷의 정보를 통하여 패킷 전송간의 지연시간과 throughput을 실시간으로 계산한다. 그리고 측정 시 클라이언트인 모바일 단말기에 요구되는 계산량을 줄임으로써 모바일 단말기에 걸리는 부하를 최소화하였다. 설계한 시스템을 통하여 네트워크 자원을 최대로 사용하였을 시 Wi-Fi 망이 3G 망보다 짧은 전송지연시간, 높은 최대 throughput, 낮은 손실률을 가지고, Android가 iOS보다 짧은 전송지연시간과 높은 최대 throughput을 가지며, UDP가 TCP보다 긴 전송지연시간, 높은 최대 throughput을 가진다는 것을 확인하였다.

Abstract

As a rapid increase of mobile network usage, many studies on solution for network traffic's demand problem have been done. Especially network environment measurement area provides basis for solving network traffic's demand problem by finding causes of problems through accurate network analysis. However, as increase of demand for smartphone, we should consider effects of mobile platform's property measuring mobile network. In this paper, we design a network traffic measurement system considering mobile platform. Through the information from packets, this system calculates packet transmission delay and throughput. We minimize computation cost required for a mobile device that is a client in this system. When fully using network resources, we found that Wi-Fi has shorter transmission delay, higher maximum throughput and lower loss rate than 3G, Android has shorter transmission delay and higher maximum throughput than iOS, and UDP has longer transmission delay and higher maximum throughput through this system.

Keywords : Mobile Platform, Wi-Fi, 3G, network traffic measurement

Ⅰ. 서 론

최근 모바일 환경은 스마트폰의 대중화와 모바일 플랫폼의 발달로 인해 급격한 변화를 겪고 있다. 2012년 2월말 기준 국내 스마트폰 가입자 2,479만명에 이르며,

* 학생회원, ** 평생회원, 인하대학교 전자공학과
(Dept. of Electronic Engineering, Inha University)
※ 이 논문은 정부(교육과학기술부)의 재원으로 한국연구재단의 중점연구소 지원사업으로 수행된 연구임 (2012-0005858)
접수일자: 2012년11월4일, 수정완료일: 2013년1월20일

2011년 10월 말 대비 9.8% 증가하였다[1]. 그리고 미국 무선통신협회 CTIA가 공개한 하반기 조사 보고서에 의하면 2011년 미국의 모바일 데이터 사용량은 전년 대비 123% 증가하여 9천만GB에 달한다고 한다[2].

이처럼 급증하는 데이터 트래픽 수요를 충족시키기 위해 이동통신사들은 기지국 추가 구축을 통해 커버리지와 데이터 용량 확충에 총력을 기울이고 있다. 그러나 최신 단말의 출시 및 모바일 네트워크 시장의 확대에 따라 모바일 데이터 트래픽 수요는 더욱 급증할 것으로 예상된다.

이렇듯 기하급수적으로 증가하는 데이터 통신량으로 인해 네트워크 전송 지연 및 QoS (Quality of Service) 감소, 트래픽 오버플로우가 발생할 수 있다. 또한 모바일 단말기의 상대적으로 부족한 연산 처리 능력도 데이터 전송 지연의 원인이 된다. 뿐만 아니라, 모바일 네트워크환경은 높은 링크 에러와 채널 간섭이 존재하는 무선망을 사용하기 때문에 유선망에 비해 전송 지연, 비순차 전송, 패킷 손실을 겪을 확률이 높아진다. 따라서 모바일기기 사용자에게 적절한 QoS를 보장하기 위해 모바일 플랫폼 상에서 이루어지는 무선 네트워크 환경의 정확한 측정이 필요하다[3-6].

이에 본 논문에서는 모바일 플랫폼 상의 무선 네트워크의 특성을 반영한 측정 시스템을 설계하였다. 클라이언트/서버 기반의 이 측정 시스템은 서버와 클라이언트 간에 측정 패킷을 전송하여 각 시스템이 기록한 타임스탬프를 통해 전송지연시간, throughput, 패킷 손실률을 계산한다. 특히 클라이언트인 모바일 기기가 수신한 측정 패킷에 타임스탬프를 기록 후 서버인 PC로 재전송하는 기능만을 수행하도록 하여 모바일 기기의 처리 성능 부족이 측정시스템에 미치는 영향을 최소화하였다. 그리고 패킷 크기, 생성 트래픽의 대역폭, 전송 프로토콜을 사용자가 설정함으로써 측정하기를 원하는 어플리케이션에 해당하는 트래픽을 생성하여 네트워크의 성능을 평가할 수 있다. 본 논문은 다음과 같이 구성된다. Ⅱ장에서는 모바일 플랫폼 기반으로 동작하는 무선 네트워크의 특징을 설명하고, Ⅲ장에서는 무선 네트워크 환경 측정 시스템의 구조를 설명한다. Ⅳ장에서는 모바일 플랫폼 상의 무선 네트워크 환경을 측정하고 결과를 분석하였으며, Ⅴ장의 결론으로 본 논문을 마무리한다.

Ⅱ. 관련 연구

METAWIN monitoring system[7]은 지속적으로 3G 네트워크 트래픽을 모니터링 하면서, 네트워크의 상태와 성능을 분석하는 도구이다. 특히 METAWIN은 TCP 의 congestion control, fast recovery 등의 특성을 이용하여 링크에 연결되어 있는 모든 네트워크 개체를 확인하지 않고 종단 시스템에서의 정보를 이용하여 네트워크를 단순화시킴으로써 분석에 걸리는 소모비용을 줄여준다. 그리고 네트워크 사용자의 패턴 분석, 위험요소 및 문제에 대한 조기 탐색, Core Network과 Radio Access Network의 혼잡 감지 등의 추가 기능을 가지고 있다. 그러나 3G 네트워크 트래픽을 TCP/IP로 한정하였기 때문에, 동영상 스트리밍, 영상 통화, 게임 등의 멀티미디어 서비스 (UDP/IP)를 주로 사용하는 환경에는 적합하지 못하다.

Seawind[8]는 실제 네트워크에 전송되는 패킷을 이용하여 무선 네트워크 환경에서 발생할 수 있는 전송 지연, 대역폭 제한 등을 반영할 수 있는 에뮬레이터이다. Wi-Fi환경 및 3G와 같은 GPRS (General Packet Radio Service) 및 UMTS (Universal Mobile Telecommunication System) 환경에 대한 테스트를 수행할 수 있는 환경을 제공한다. 그리고 느린 전송속도 및 상/하향 비대칭 채널에 특성 및 전송 지연, 높은 손실률, 무선 단말의 이동으로 인한 핸드오버 (handover) 및 블랙아웃 (blackout) 현상을 제공하며 분산 환경의 지원으로 테스트 베드와 설정 노드를 분리한 운영이 가능하다. 또한 버퍼 오버플로우로 인한 혼잡환경을 제공하기 위하여 가상의 background load를 지원하며 채널 특성을 각각의 큐로 모델링하기 때문에 다양한 시나리오에 적용할 수 있다. 하지만 다량의 노드가 존재할 경우 생성 큐가 많아져 실행 속도가 느려지며 다양한 모바일 어플리케이션에 대하여 최적화된 구조가 아니라는 단점이 있다.

Ⅲ. 모바일 네트워크 환경 측정 시스템

3.1 모바일 네트워크 환경 측정 요소

본 논문에서는 최대 throughput, 평균 전송지연시간, 패킷 손실률을 측정 요소로 정의한다. 서버는 사용자가 임의로 설정한 패킷 크기 p_{size}[bytes]와 전송 대역폭

BW[bytes/s]을 이용하여 측정 트래픽을 생성한 뒤 모바일 클라이언트로 전송한다. 모바일 클라이언트를 거쳐 다시 서버로 들어오는 측정 패킷의 타임스탬프를 이용해 지연 시간을 계산한다. 이 때 계산된 지연 시간과 수신 패킷 크기를 이용하여 측정 종료 후 throughput을 계산한다. 측정 패킷 p의 시퀀스 번호가 i이고 마지막 시퀀스 번호를 N이며 시퀀스 번호 i를 가진 측정 패킷로부터 계산된 전송지연시간이 RTT_i[ms]라 할 때, throughput은 식 (1)에 의해 결정된다.

$$\text{Throughput} = \sum_{i=1}^{N} p_{size} \times 10^3 \times 8 / \sum_{i=1}^{N} RTT_i [bps] \quad (1)$$

같은 패킷 크기에 대하여 대역폭을 변화시키면서 측정을 수행하였을 때, 패킷 손실이 존재하지 않고 throughput이 가장 높을 때 이를 최대 throughput으로 결정한다. 측정 트래픽이 패킷을 손실하지 않으면서 throughput이 가장 높게 측정되었다는 것은 그만큼 사용 가능한 네트워크 자원을 최대로 활용했다는 것을 의미한다.

패킷이 도착될 때마다 계산되는 전송지연시간은 네트워크 상태에 따라 다양하게 변하므로 특정 패킷 도착 시 계산된 전송지연시간을 이용하여 네트워크의 성능을 판단할 수 없다. 따라서 측정이 종료된 후 각 측정 패킷으로부터 식 (2)를 통하여 전송지연시간 RTT_i 들의 합을 총 수신 패킷 수 N_{recv}로 나누어 평균 전송지연시간 $average\ latency$을 계산한다. N_{recv}은 마지막 시퀀스 번호인 N에서 손실 패킷 개수 N_{loss}를 제외한 나머지 패킷의 개수가 된다.

$$N_{recv} = N - N_{loss}$$
$$average\ latency = \sum_{i=1}^{N} RTT_i / N_{recv} [ms] \quad (2)$$

패킷 손실률은 측정이 끝난 후 수신한 패킷 수 대비 손실로 판단된 패킷 개수의 비율을 계산하여 도출한다. UDP는 TCP와 달리 손실 복구 과정이 없기 때문에 측정 도중 패킷 손실이 발견될 수 있다. 그러나 수신된 패킷들의 시퀀스 번호들 중 비어있는 시퀀스 번호를 손실로 판단하더라도 해당 시퀀스 번호를 가진 패킷이 일정 시간 이후 도착하는 비순차 전송 현상이 발생할 수 있다. 따라서 패킷 손실률 측정을 위해 패킷 손실 또는 비순차 전송 패킷을 구별하는 방법이 필요하며 그 방법을 3.4절에서 자세히 다룬다.

3.2 모바일 네트워크 환경 측정 시퀀스

3.2.1 기본 네트워크 환경 측정 시퀀스

PC 기반의 네트워크 환경을 측정하는 서버/클라이언트 모델은 그림 1과 같다. 우선 클라이언트가 서버에 접속요청을 보내며, 연결이 성립되면 클라이언트는 사용자가 설정한 크기에 맞추어 패킷을 생성하며, 식 (3)에 의해 타이머의 값을 설정하여 생성 패킷이 전송되는 간격 $interval$을 조절함으로써 사용자가 설정한 대역폭 BW에 맞는 측정 트래픽이 전송되도록 한다.

$$interval = \frac{1000}{BW / p_{size}} [ms] \quad (3)$$

서버는 클라이언트로부터 수신한 측정 패킷에 기록된 타임스탬프를 이용하여 단방향 전송지연시간, throughput을 계산하고 측정 패킷의 시퀀스 번호를 이용하여 패킷 손실률을 계산한다. 계산 진행 중에 서버는 수신 패킷에 타임스탬프를 기록한 뒤 측정 패킷을 클라이언트로 다시 돌려보낸다. 클라이언트는 서버로부터 측정 패킷을 수신하자마자 측정 패킷에 기록된 총 3개의 타임스탬프들과 도착 직후 기록한 타임스탬프를 이용하여 양방향 전송지연시간, throughput을 계산하고

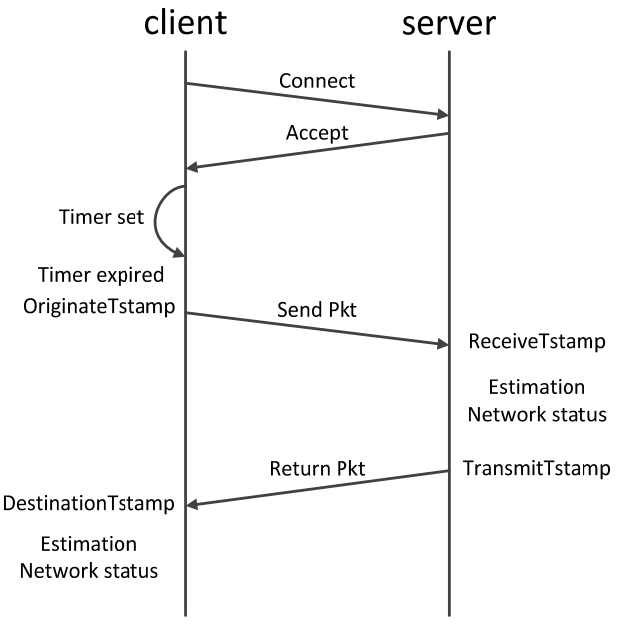

그림 1. 기본 네트워크 환경 측정 시퀀스
Fig. 1. Sequence of basic network measurement.

측정 패킷의 시퀀스 번호를 이용하여 패킷 손실률을 계산한다.

이러한 기능들은 PC에서 수행할 때 처리지연이 크게 발생하지 않지만 iOS나 Android 플랫폼을 사용하는 모바일 기기에서 수행할 경우 PC보다 뒤처지는 처리성능 때문에 처리지연이 길어져 네트워크 성능을 저하시키는 원인이 된다. 따라서 이 모델을 사용할 경우, 정확한 모바일 네트워크 측정을 보장하지 못하는 문제가 있다.

3.2.2 모바일 플랫폼을 위한 네트워크 환경 측정 시퀀스

본 논문에서 설계한 측정시스템에서 모바일 기기의 네트워크 환경을 측정하는 서버/클라이언트 모델은 그림 2와 같다. 모바일 클라이언트는 서버에 접속하여 트래픽 생성에 필요한 변수를 전송하는 것과 서버로부터 수신한 패킷에 타임스탬프를 기록하여 서버로 다시 돌려보내는 것 이외의 다른 기능을 수행하지 않기 때문에 계산비용이 줄어들어 모바일 기기에 걸리는 부하가 줄어든다. 서버는 클라이언트로부터 연결요청을 받고 연결이 성립되면 모바일 클라이언트로부터 수신한 대역폭, 패킷 크기를 식 (3)에 대입하여 전송시간간격을 설정하고 해당 시간에 측정 패킷을 모바일 클라이언트에 보낸다. 서버로부터 측정 패킷을 수신한 모바일 클라이언트는 다른 작업을 수행하지 않고 측정 패킷에 타임스탬프만을 기록한 후 다시 서버로 측정 패킷을 전송한다. 모바일 클라이언트가 타임스탬프를 기록한 측정 패킷을 다시 서버로 반환하면 서버는 패킷에 기록된 타임스탬프들을 이용하여 양방향 전송지연시간과 throughput을 계산하고 측정 패킷의 시퀀스 번호를 이용하여 패킷 손실률을 계산한다.

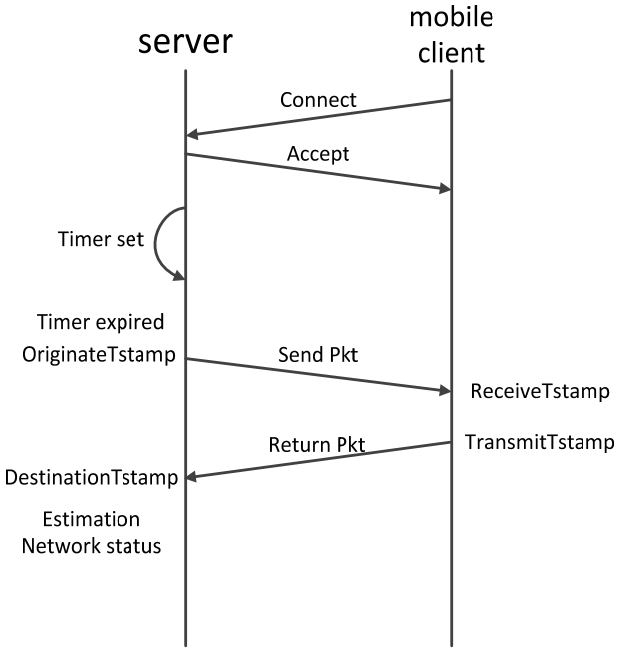

그림 2. 모바일 플랫폼을 위한 네트워크 환경 측정 시퀀스

Fig. 2. Sequence of network measurement for mobile platform.

측정 시스템 구현을 위해 만든 PC 버전의 서버 프로그램은 PC에서 접속 시 기본 네트워크 환경 측정 시퀀스를 따르며 iOS나 Android 플랫폼을 사용하는 기기에서 접속 시 모바일 플랫폼을 위한 네트워크 환경 측정 시퀀스를 따르도록 설계하였다.

3.3 모바일 네트워크 환경측정 시스템 구조

이 장에서는 본 논문에서 제안하는 모바일 플랫폼 상의 네트워크 특성을 반영한 측정 시스템의 구조에 대하여 설명한다.

3.3.1 모바일 네트워크 환경측정 시스템 구성

그림 3은 모바일 플랫폼을 위한 네트워크 환경측정 시스템 구조를 나타낸다. 측정 시스템은 서버/클라이언트 기반의 모델을 사용한다.

서버 (PC)는 트래픽 생성, 타이머 이벤트 핸들러, 타임스탬프, 전송용 소켓, 수신용 소켓, 통계 및 로그 블록으로 구성된다. 트래픽 생성 블록은 타이머 핸들러의 신호에 맞추어 패킷을 생성한다. 타이머 이벤트 핸들러 블록은 클라이언트로부터 받은 패킷 크기와 대역폭 값을 이용하여 패킷이 전송되는 시간 간격을 설정하고, 해당 시간에 트래픽 생성 블록에 신호를 보내 패킷을 생성한다. 서버의 타임스탬프 블록은 *OriginateTstamp*와 *DestinationTstamp* 두 블록이 있다. *OriginateTstamp* 블록은 패킷 생성 후 시간을 패킷 페이로드에 기록하며, *DestinationTstamp* 블록은 패킷 도착 직후 시간을 패킷 페이로드에 기록한다. 전송용 소켓 블록은 클라이언트의 수신용 소켓과 연결되어 트래픽 생성 블록에서 생성한 패킷을 클라이언트로 전송한다. 수신용 소켓 블록은 클라이언트의 전송용 소켓과 연결되어 클라이언트로부터 초기 설정값들이 기록된 패킷을 수신하여 데이터를 추출하고 그 값을 트래픽 생성 블록과 타이머 이벤트 핸들러 블록으로 전달하며, 측정 시작 후에는 클라

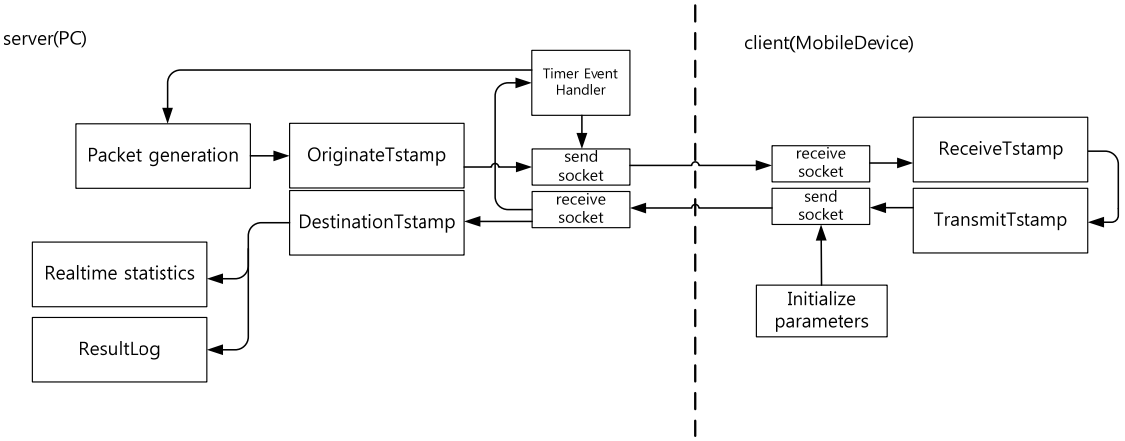

그림 3. 모바일 플랫폼을 위한 네트워크 환경측정 시스템 구조
Fig. 3. Structure of network measurement system for mobile platform.

이언트의 타임스탬프 과정을 거친 측정 패킷을 수신하여 통계 및 로그 블록으로 전달한다. 통계 및 로그 블록은 수신용 소켓 블록으로부터 패킷을 수신할 때마다 패킷 페이로드로부터 추출한 데이터를 이용하여 throughput, 전송지연시간, 손실 패킷 개수를 계산하고 값을 버퍼와 텍스트에 저장한다. 그리고 측정이 끝난 뒤 수집된 데이터를 이용하여 최대 throughput, 평균 전송지연시간, 손실률을 계산한다.

클라이언트 (모바일 기기)는 설정값 초기화, 타임스탬프, 전송용 소켓, 수신용 소켓 블록으로 구성된다. 파라미터 초기화 블록은 사용자가 입력한 패킷 크기, 대역폭, 전송프로토콜 값을 패킷 페이로드에 기록 후 전송용 소켓으로 전달한다. 클라이언트의 타임스탬프 블록은 *ReceiveTstamp*와 *TransmitTstamp*가 있다. *ReceiveTstamp*는 수신용 소켓 블록으로부터 받은 패킷의 페이로드에 시간을 기록하며, *TransmitTstamp*는 전송용 패킷 블록에 패킷을 전송하기 전 시간을 패킷 페이로드에 기록한다. 전송용 소켓 블록은 서버의 수신용 소켓 블록과 연결되어 파라미터 초기화 블록과 *TransmitTstamp* 블록으로부터 패킷을 수신하여 서버의 수신용 소켓 블록으로 전송하며, 수신용 소켓 블록은 서버의 전송용 소켓 블록과 연결되어 수신 패킷을 *ReceiveTstamp* 블록으로 전달한다.

3.3.2 모바일 네트워크 환경측정 시스템 동작 흐름

시스템 시작 시, 모바일 기기에서 패킷 크기와 전송률, 전송 프로토콜을 설정하고, 패킷의 페이로드에 이값들을 작성하여 서버로 전송한다. 클라이언트와 연결된 서버가 데이터를 수신하면 서버는 이 설정값들을 이용하여 시퀀스 번호, 패킷 크기, 타어머, 전송 프로토콜을 정한다. 타이머 이벤트 핸들러는 패킷 출발 간 시간 간격을 생성하며, 패킷 출발 시간마다 패킷을 생성시킨다. 생성된 패킷은 *OriginateTstamp*가 기록된 후 클라이언트로 전송된다. 패킷 도착 즉시 클라이언트는 *ReceiveTstamp*를 패킷에 기록한다. 기록이 끝난 후 바로 서버에 패킷을 전송하기 전 *TransmitTstamp*를 패킷에 기록한다. 이 패킷을 수신한 서버는 마지막으로 *DestinationTstamp*를 패킷에 기록한다. 서버는 패킷의 페이로드로부터 메시지 길이, 타임스탬프, 전송 프로토콜을 추출하여 실시간 및 최종 통계와 로깅을 동시 수행하게 된다.

3.3.3 모바일 네트워크 환경측정 시스템의 RTT 측정

네트워크 전송지연시간을 측정 방법은 크게 OTT (One-way Trip Time), RTT (Round Trip Time) 두 가지가 있다. 서버와 클라이언트까지 걸리는 단방향 전송지연시간인 OTT를 측정하기 위해서는 양단의 시간 동기화가 필요하다. 네트워크를 이용한 시간 동기화 방법은 Time Protocol[9], NTP (Network Time Protocol)[10], SNTP (Simple Network Time Protocol)[11] 등 여러 가지 방법이 있지만 NTP를 제외한 다른 방법들은 정확도가 수십 ms 정도로 낮기 때문에 적용에 제약이 있으며, NTP를 이용한 방법 역시 동기화에 적지 않은 시간이 소요되기 때문에 모바일 플랫

폼을 사용하는 기기에 적지 않은 계산 비용이 발생하여 실사용에 어려움이 있다. 이에 본 논문에서는 모바일 네트워크 환경 측정 시스템에서 전송지연시간을 구하기 위해 OTT 대신 RTT를 이용함으로써 OTT 이용 시에 시간 동기화에 걸리는 계산비용 없이 전송지연시간을 더 정확히 구할 수 있도록 하였다.

구현된 모바일 네트워크 환경측정 프로그램은 RTT를 측정할 때 SNTP와 동일한 방법으로 구현되었다. RTT 측정을 위해 총 4개의 타임스탬프가 사용된다. 서버 측에는 *OriginateTstamp*와 *DestinationTstamp*가 사용된다. *OriginateTstamp*는 서버가 패킷을 생성한 시각의 타임스탬프이며 *DestinationTstamp*는 서버가 클라이언트로부터 패킷을 받았을 때의 타임스탬프이다. 클라이언트 측에는 *ReceiveTstamp*와 *TransmitTstamp*가 사용된다. *ReceiveTstamp*는 클라이언트가 서버로부터 패킷을 수신하였을 때의 타임스탬프이며 *TransmitTstamp*는 클라이언트가 서버로 패킷을 다시 돌려보낼 때의 타임스탬프이다. 타임스탬프 값들을 식 (4)에 대입하여 시스템을 거쳐간 패킷의 RTT(Round Trip Time)를 계산하게 된다.

$$RTT = (Destination\,Tstamp - Originate\,Tstamp) - (Transmit\,Tstamp - Receive\,Tstamp) \quad (4)$$

3.4 패킷 손실 및 비순차 전송 감지

UDP는 TCP와는 달리 손실 복구를 하지 않기 때문에 네트워크에 장애로 인한 패킷 손실과 비순차 전송이 발생할 수 있다. 이에 본 논문에서는 UDP를 사용함으로써 발생하는 패킷 손실에 대하여 패킷의 시퀀스 번호를 이용한 감지 프로시저를 설계하였다. 설계한 프로시저는 패킷 손실 감지 프로시저, 비순차 전송 감지 프로시저, 정상 수신 감지 프로시저 세 부분으로 나눌 수 있다.

감지 프로시저는 우선 수신 패킷의 시퀀스 번호 *LastSeq*와 이전 수신 패킷의 시퀀스 번호 중 가장 높은 번호 *HighestSeq*를 비교한다. 만약 *LastSeq*가 *HighestSeq*+1 보다 크다면 패킷 손실 감지 프로시저를 수행한다. 이 조건은 수신 패킷과 이전에 수신한 패킷 사이에 최소 1 이상의 시퀀스 번호 차이가 존재한다는 것을 의미하며, 즉 다시 말해 패킷 수신 간에 손실 가능성이 존재함을 의미한다. *LastSeq*와 *HighestSeq*간에

```
    // Packet Loss Detection Procedure
 1   Set LastSeq = Seqnum of packet received

    // Loss Detection
 2  if LastSeq > HighestSeq + 1 then
 3   Set expectNum = LastSeq  -  (HighestSeq+1)
 4   for i=0 to expectNum do
 5    Set lossCandidate.LossCandSeq = HighestSeq + 1 + i
 6    Set next lossCandidate
 7   endfor

    // out of order Detection
 8  elseif LastSeq < HighestSeq then
 9   while lossCandidate.LossCandSeq is not null
 10   if lossCandidate.LossCandSeq == LastSeq then
 11    Remove lossCandidate.LossCandSeq
 12   endif
 13   Set next lossCandidate.LossCandSeq
 14  endwhile

    // otherwise
 15 else then
 16  HighestSeq = LastSeq
 17 endif
```

그림 4. 패킷 손실 감지 프로시져의 의사코드
Fig. 4. Pseudo code of Packet Loss Detection Procedure.

비어있는 시퀀스 넘버를 가진 패킷들을 손실로 판단할 수 있으나 아직 링크 상에 존재할 수 있을 가능성이 있다. 따라서 *LastSeq*와 *HighestSeq*간에 비어있는 시퀀스 번호를 가진 패킷들을 손실 패킷 후보군으로 정의하고 시퀀스 번호들을 구조체 배열에 저장한다.

패킷 손실 감지 프로시저의 조건과는 달리 *LastSeq*가 *HighestSeq*보다 작은 경우 비순차 전송 감지 프로시저를 수행한다. 이 조건은 순차적으로 서버에 수신되어야 할 시퀀스 번호보다 더 작은 시퀀스 번호를 가진 패킷이 수신되었다는 것을 의미하며, 즉 다시 말해 이전에 손실 패킷 후보군으로 정의한 패킷이 서버에 도착하였다는 것을 의미한다. 따라서 프로시저는 손실 후보군으로 정의했던 패킷의 시퀀스 번호와 수신 패킷의 시퀀스 번호가 일치하면 해당 시퀀스 번호를 구조체 배열에서 삭제하게 된다.

마지막으로 위의 두 조건에 해당하지 않으면 수신 패킷은 정상적인 시퀀스로 수신된 것이므로 정상 수신 감

지 프로시저를 수행하게 된다. 프로시저는 *LastSeq*를 *HighestSeq*로 설정한다.

그림 4는 프로시저의 의사코드를 나타낸다.

Ⅳ. 모바일 네트워크 환경 측정 및 분석

이 장에서는 제안한 측정 시스템을 이용하여 모바일 네트워크 성능을 평가하기 위한 실험 환경을 정의하고, 모바일 플랫폼 상의 Wi-Fi와 3G의 네트워크 성능, Android와 iOS 플랫폼을 사용하는 기기의 네트워크 성능, TCP와 UDP의 네트워크 성능을 측정하고 이에 대한 분석 결과를 기술한다.

4.1 실험 환경

모바일 플랫폼 상의 네트워크의 성능을 측정 및 평가하기 위해여 iOS와 Android를 사용하는 기기와 PC 서버 간의 네트워크 트래픽을 측정하고 결과를 비교하였다. 여기서 iOS는 애플, Android는 구글의 모바일 플랫폼이다.

제안한 측정 시스템에 의한 모바일 네트워크 성능을 측정하기 위해 실 장비들로 시스템을 구성하였다. 각 플랫폼 상의 무선 네트워크를 측정하기 위해 애플의 아이패드2 (iOS 4.3), 모토로라의 Zoom (Android 3.0)을 사용하였다. 제안한 모바일 네트워크 환경 측정시스템 방식을 이용, 각 모바일 기기의 어플리케이션에서는 전송 프로토콜, 패킷 크기, 대역폭을 서버에 전송하게 되며, 이를 수신한 서버가 이 설정값들을 이용해 해당 트래픽을 생성하고 모바일 기기에 전송한다.

Wi-Fi 환경측정을 위해 그림 5 와 같은 테스트 베드를 구성하였다. Wi-Fi 네트워크는 주로 공유기와 같은 하나의 무선 Base Station에 무선 장치들이 연결되고 무선 Base Staion은 유선망에 연결되어 Base Station이 유/무선망을 연결하는 라우터의 역할을 수행한다. PC가 Base Station에 유선 연결되고 모바일 기기가 Base Station에 무선 연결되면 모바일 기기에서 서버로 접속을 시도한다. 접속이 성공하면 패킷을 Base Station을 거쳐 주고받으며 측정을 수행하게 된다.

3G 환경측정을 위해 그림 6과 같은 테스트 베드를

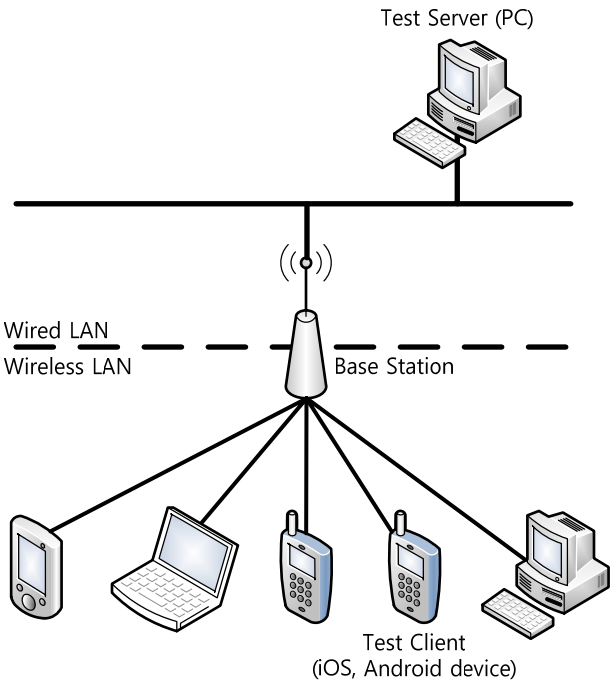

그림 5. Wi-Fi 환경측정 구조
Fig. 5. Structure for Wi-Fi network measurement.

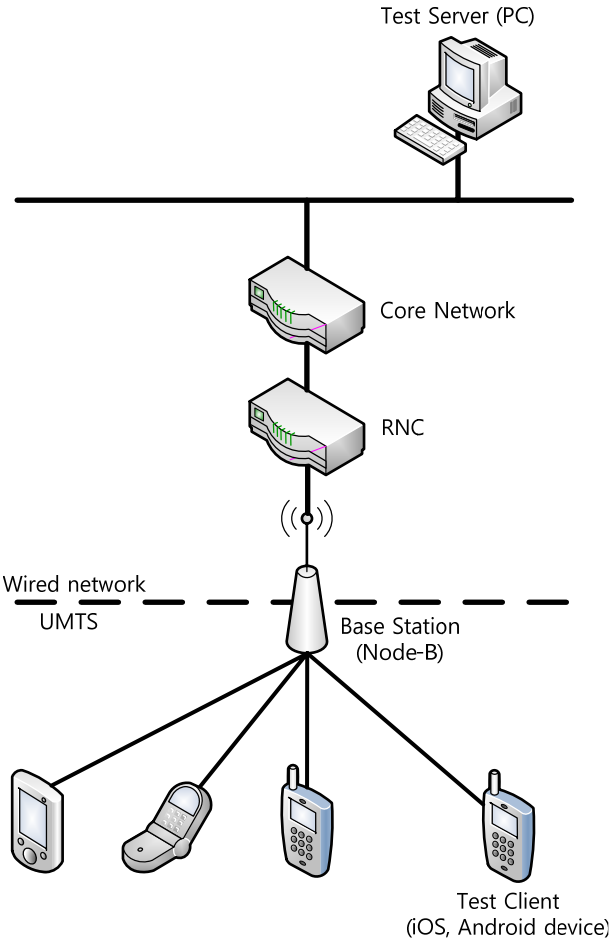

그림 6. 3G 환경측정 구조
Fig. 6. Structure for 3G network measurement.

구성하였다. WCDMA라 불리는 3세대 UMTS 네트워크는 각각의 단말을 UE(User Equipment)라 명명한다. 또한 무선 Access Network에서 UE들의 접속을 관리하는 Base Station의 역할을 하는 Node-B와 연결을 관리하는 RNC(Radio Network Controller), 망 내에서의 패킷의 전달 및 기존의 유선망과 이어주는 Core Network로 이루어져 있다. UMTS는 통신 사업자에 의해 네트워크가 구성되기 때문에 단일 로컬 네트워크에서의 측정이 불가능하다. 따라서 Node-B에 모바일 기기를 연결한 후 통신 사업자에 의해 감추어진 유선 네트워크를 통해 서버와 연결이 성립되었을 때 측정을 수행하게 된다.

4.2 모바일 네트워크 성능 분석

측정시간 종료 후, 서버가 계산한 평균 전송지연시간, 최대 throughput, 패킷 손실률을 종합하여 그래프로 나타낸 후 결과를 비교하였다.

4.2.1 Access network에 따른 특징

Access network에 따른 모바일 네트워크의 성능을 알아보기 위해 Android의 무선 통신 모드를 Wi-Fi와 3G로 바꾸어가며 측정을 수행하였다. iOS가 아닌 Android 플랫폼에서 Access network에 따른 트래픽 측정을 수행한 이유는 Android와는 달리 iOS가 그래픽 작업에 더 많은 CPU와 메모리를 할당하며 이에 따라 긴 처리지연이 발생하기 때문이다. 따라서 측정 시 iOS보다 Android를 사용함으로써 Wi-Fi와 3G가 가진 네트워크 자원을 더 활용할 수 있다. 그리고 전송 프로토콜을 UDP를 사용함으로써 측정 시스템이 Wi-Fi와 3G가 가진 자원을 최대로 이용할 수 있도록 하였다.

그림 7은 설정 패킷 크기에 대한 Wi-Fi와 3G의 최대 throughput을 비교한 그래프이다. Wi-Fi의 경우 패킷 크기가 증가할수록 최대 throughput이 증가하는 것을 알 수 있다. 패킷 크기와 대역폭을 각각 1000bytes, 7.2Mbps로 설정하였을 때, 약 7.19Mbps 최대 throughput이 측정되었다. 반면 3G의 경우 패킷 크기와 대역폭을 각각 1000bytes, 400Kbps로 주었을 때, 약 393Kbps의 낮은 최대 throughput이 측정되었다. 특히, 100bytes의 UDP 패킷을 자주 전송하고자 할 때 throughput이 111kbps으로 낮았다. 이는 네트워크의 장애에 의해 발생되었다기보다는 통신 사업자가 전송률을

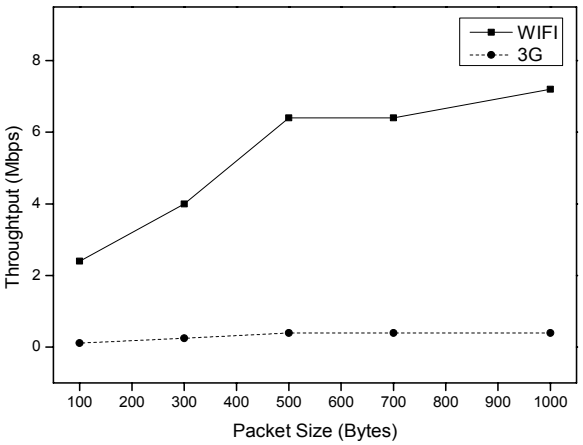

그림 7. Wi-Fi와 3G의 최대 throughput 비교
Fig. 7. The Comparison with maximum throughputs of Wi-Fi and 3G.

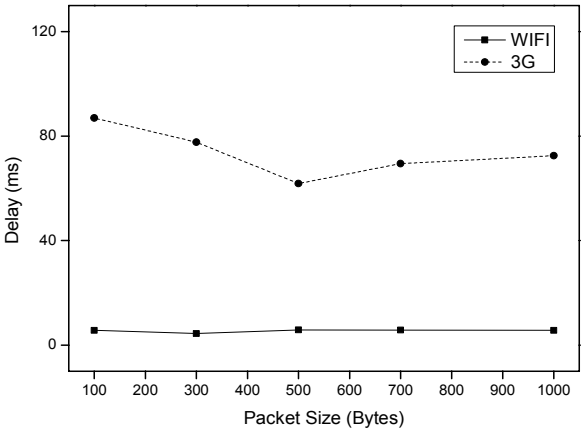

그림 8. Wi-Fi 와 3G 의 평균 전송지연시간 비교
Fig. 8. The Comparison with average delays of Wi-Fi and 3G.

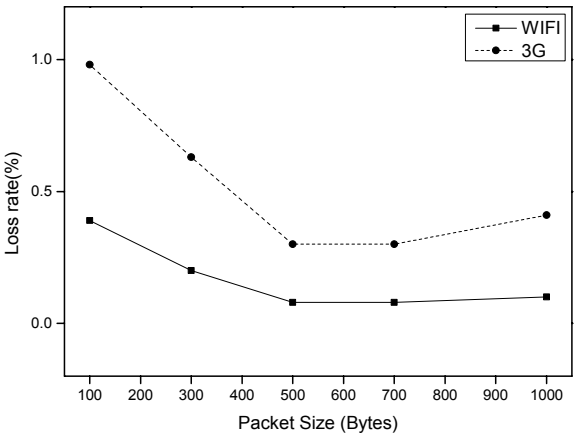

그림 9. Wi-Fi와 3G 의 손실률 비교
Fig. 9. The Comparison with 손실률s of Wi-Fi and 3G.

정책적으로 제한하는 것으로 보인다.

그림 8은 측정된 최대 가용 대역폭 사용 시, Wi-Fi와 3G의 평균전송지연을 비교한 그래프이다. Wi-Fi는 전송지연시간이 5.8ms 이하의 낮은 수치로 측정되었으며, 3G의 경우 61~87ms의 높은 평균전송지연을 가졌다. 이는 3G가 Wi-Fi보다 좁은 가용 대역폭과 높은 링크 에러를 가지고 있어 전송 장애의 가능성이 높고, 다수의 망 가입자가 Base Station 하나에 접속을 시도하여 측정 트래픽이 다른 트래픽에 의해 간섭받을 가능성이 높기 때문이다. 이러한 장애 요인이 클수록 패킷 손실률도 높아지게 된다.

그림 9는 Wi-Fi와 3G의 손실률을 비교한 그래프이다. 패킷의 크기를 100bytes로 설정하였을 때, 최대 가용 대역폭 근처에서의 손실률은 Wi-Fi에서 0.39%를 갖지만 3G 환경에서는 0.98%의 손실률을 갖는 것으로 확인하였다. 패킷 크기가 작으면 같은 대역폭을 사용할 때, 더 많은 패킷을 전송하므로 그만큼 패킷 손실의 가능성이 높다. 특히 3G는 Wi-Fi보다 가용 자원이 부족하므로 더 많은 손실이 발생하게 된다.

4.2.2 모바일 플랫폼에 따른 특징

iOS를 탑재한 아이패드2와 Android를 탑재한 Zoom을 무선 네트워크로 서버에 연결한 후 패킷 크기와 대역폭을 조정하면서 측정을 수행하였다. 통신 방식은 Wi-Fi를, 프로토콜은 UDP를 사용함으로써 가용 자원을 최대로 사용하도록 하였다. 그림 10은 설정한 패킷 크기에 대한 Android와 iOS의 최대 throughput을 비교한 그래프이다. Android의 최대 throughput은 iOS보다 2~4.5Mbps이상 높은 수치가 측정되었다. 이는 Android와는 달리 iOS가 그래픽 작업에 더 많은 자원을 할당하기 때문에 Android가 iOS보다는 네트워크 작업에 더 많은 자원을 할당하게 됨으로써 높은 throughput을 유지할 수 있기 때문이다. 특히 두 플랫폼은 패킷 크기가 작을수록 최대 throughput이 낮은데, 같은 대역폭을 사용 시 패킷 크기가 작을수록 더 많은 패킷이 생성되어 링크 혼잡이 커지기 때문이다.

그림 11은 측정된 최대 가용 대역폭 사용 시, Android와 iOS의 평균전송지연을 비교한 그래프이다. Android는 2.79~5.67ms의 짧은 전송 지연 시간이 측정되었으며, iOS는 최대 23.38ms의 긴 전송 지연 시간이 측정되었다. 이는 Android보다 iOS가 네트워크 작업 처리 후 자원을 반환하는 횟수가 더 많아 처리지연이 더 길다는 것을 보여준다. 패킷 크기가 작을수록 트래픽이 많기 때문에 처리지연시간은 길어지고 이에 따른 평균 전송지연시간도 길어지게 된다.

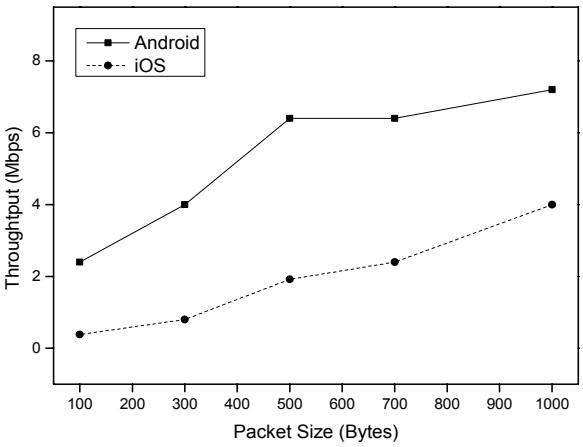

그림 10. Android와 iOS의 최대 throughput 비교
Fig. 10. The comparison with maximum throughputs of Android and iOS.

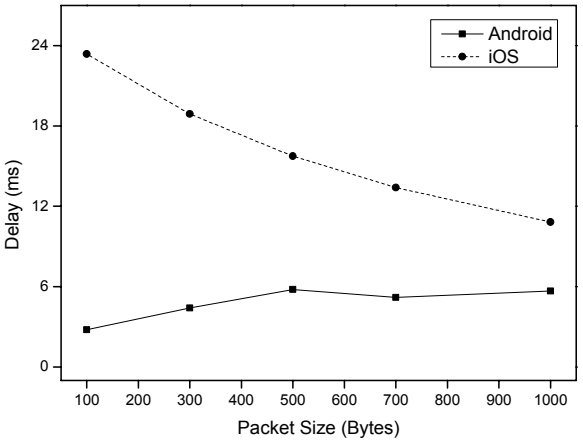

그림 11. Android와 iOS의 평균 전송지연시간 비교
Fig. 11. The comparison with average delays of Android and iOS.

4.2.3 전송 프로토콜에 따른 특징

Android를 탑재한 Zoom을 무선 네트워크를 통해 서버에 연결한 후 패킷 크기, 대역폭, 전송 프로토콜을 조정하여 측정을 수행하였다. 통신 방식은 Wi-Fi를 사용함으로써 각 전송 프로토콜이 링크를 최대로 사용하도록 하였다. 그림 12는 할당 패킷 크기에 대한 TCP와 UDP의 최대 throughput을 비교한 그래프이다. TCP의

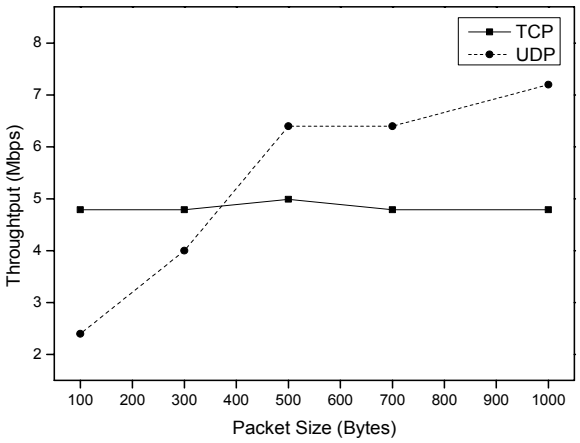

그림 12. TCP와 UDP의 최대 throughput 비교
Fig. 12. The Comparison with maximum throughputs of TCP and UDP.

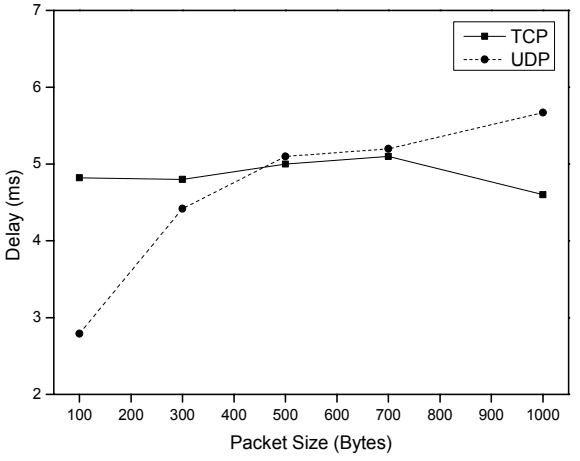

그림 13. TCP와 UDP의 평균 전송지연시간 비교
Fig. 13. The Comparison with average delays of TCP and UDP.

경우 UDP와 달리 흐름제어를 수행함으로써 채널의 부하 상태에 따라 전송량을 제어하기 때문에 안정적인 데이터 전송을 수행한다. 이에 반해 UDP는 흐름제어를 수행하지 않기 때문에 채널의 부하 상태에 따라 throughput이 다르다. 특히 패킷 크기가 100bytes일 때 최대 throughput이 TCP가 UDP보다 2.39Mbps 더 높게 측정되었다. 이것은 UDP와는 달리 TCP의 경우 ACK을 받기 전까지 생성 패킷을 버퍼를 채우고 있다가 ACK을 받은 후 버퍼의 패킷을 전송하는 nagle algorithm이 사용되어 UDP보다 적은 패킷을 전송하게 되고 링크의 혼잡이 그만큼 줄어들어 높은 throughput을 낼 수 있기 때문이다. 그러나 패킷의 크기가 증가함에 따라 UDP의 최대 throughput이 증가하게 되어 패킷 크기 500bytes부터는 TCP보다 높은 throughput을 유지하고 있다. TCP는 혼잡 제어와 손실 복구 알고리즘으로 인해 일정한 throughput을 유지하려고 하는 반면 UDP는 링크 혼잡이나 패킷 손실에 관계없이 가용 대역폭을 최대로 사용하려하기 때문이다[12].

그림 13은 측정된 최대 가용 대역폭 사용 시, TCP와 UDP의 평균 전송지연시간을 비교한 그래프이다. 패킷 크기가 100bytes일 때, TCP가 UDP보다 평균전송지연이 2.03ms 긴 것을 알 수 있다. 이것은 nagle algorithm의 영향으로 인해 전송이 지연되기 때문이다. 그리고 TCP는 패킷 크기에 따라 평균 전송지연시간이 일정하게 유지되는 반면 UDP는 패킷 크기가 커질수록 전송 지연이 길어지는 것을 알 수 있다. UDP가 혼잡 제어 기능이 없기 때문에 패킷 크기가 커짐에 따라 링크 혼잡으로 인해 처리 지연, 큐잉 지연의 발생확률이 높고, 이는 평균 전송지연시간의 상승으로 이어지기 때문이다.

V. 결 론

본 논문에서는 모바일 네트워크의 특성을 반영한 트래픽 측정 시스템을 제안하였다. 제안 측정시스템은 직접 링크에 부하를 줌으로써 네트워크를 최대한 사용할 수 있는 최대 대역폭을 찾아낸다. 또한 모바일 기기의 성능을 고려하여 최소한의 계산 비용을 가지도록 설계하였다. 제안한 시스템으로 모바일 네트워크의 성능을 평가하기 위해, Android와 iOS 플랫폼을 탑재한 실 장비를 이용하여 모바일 네트워크 측정 시스템을 구성하였다. 측정 결과, 모바일 네트워크는 Access Network, 플랫폼, 전송 프로토콜의 차이에 따라 throughput과 평균 전송지연시간의 차이가 발생하였다. Android 플랫폼에서 Wi-Fi는 3G보다 throughput이 약 20배 높았으며, 약 1/15배 짧은 평균 전송지연시간을 가졌다. 무선 통신 모드를 Wi-Fi, 전송 프로토콜을 UDP를 사용하였을 때, Android는 iOS보다 약 2배의 throughput을 가지며, 약 1/4배의 짧은 평균 전송지연시간을 가졌다. Android 플랫폼과 Wi-Fi 사용 시, UDP는 패킷 크기가 100bytes일 때, TCP보다 약 2배 낮은 throughput을 갖고 패킷 크기가 500bytes 이상일 때 더 높은 throughput을 가졌다. 또한 UDP는 패킷 크기가 100bytes일 때 TCP보다 평

균 전송지연시간이 약 1/2배 짧고 패킷 크기가 500bytes 이상일 때 더 긴 평균 전송지연시간을 가졌다.

참 고 문 헌

[1] 방송통신위원회, “2012년 상반기 스마트폰이용실태 조사 결과발표,” Available at : http://isis.kisa.or.kr/board/?pageId=060200&bbsId=3&itemId=798
[2] CTIA, “The Wireless Association® Semi-Annual Survey Shows Significant Demand by Americans for Wireless Broadband,” Available at: http://www.ctia.org/media/press/body.cfm/prid/2171
[3] J. Y. Chung, Y. Choi, B. Park and J. W. Hong, “Measurement analysis of mobile traffic in enterprise networks,” Network Operations and Management Symposium (APNOMS), pp. 1-4, Sep. 2011.
[4] S. Kim, X. Wang, H. Kim, T. Kwon and Y. Choi, “Measurement and Analysis of BitTorrent Traffic in Mobile WiMAX Networks,” Peer-to-Peer Computing (P2P), pp. 25-27, Aug. 2010.
[5] 김진혁, 신광식, 윤완오, 이창호, 최상방, “멀티미디어 데이터 특성 모델링에 기반한 네트워크 트래픽 생성기의 구현,” 대한전자공학회지, 제47권, CI편, 제6호, 103-112, 2010년 11월.
[6] 김진혁, 신광식, 윤완오, 이창호, 최상방, “멀티스레드 어플리케이션을 위한 실시간 성능모니터의 구현,” 대한전자공학회지, 제48권, CI편, 제3호, 82-90쪽, 2011년 5월.
[7] F. Ricciato, P. Svoboda, J. Motz, W. Fleischer, M. Sedlak, M. Karner, R. Pilz, P. Romirer-Maierhofer, E. Hasenleithner and W. Jäger, “Traffic monitoring and analysis in 3G networks: lessons learned from the METAWIN project,” E & I ELEKTROTECHNIK UND INFORMATIONSTECHNIK, Vol. 123., No. 7-8, pp. 288-296, Jun. 2006.
[8] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko and K. Raatikainen, “Seawind : a Wireless Network Emulator,” In Proceedings of 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems, Sep. 2001.
[9] RFC 868 - Time Protocol, Available at: http://www.faqs.org/rfcs/rfc868.html#b
[10] RFC 1305 - Network Time Protocol (Version 3) Specification, Implementation and Analysis, Available at: http://www.faqs.org/rfcs/rfc2030.html#b
[11] RFC 2030 - Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, Available at : http://www.faqs.org/rfcs/rfc2030.html#b
[12] Z. Fu, X. Meng and S. Lu, “How Bad TCP Can Perform In Mobile Ad Hoc Networks,” Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on, pp. 298-303, 2002.

저 자 소 개

김 강 희(학생회원)
2011년 인하대학교 전자공학과 학사 졸업.
2012년~현재 인하대학교 전자공학과 석사과정.
<주관심분야 : 컴퓨터 네트워크, 무선 센서 네트워크, SoC>

여 진 주(학생회원)
2009년 인하대학교 전자공학과 학사 졸업.
2009년~현재 인하대학교 전자공학과 석사과정.
<주관심분야 : 멀티미디어 통신, 무선 통신, 네트워크 >

김 진 혁(학생회원)
2009년 인하대학교 전자공학과 학사 졸업.
2011년 인하대학교 전자공학과 석사 졸업.
2011년~현재 인하대학교 전자공학과 박사과정
<주관심분야 : 멀티미디어 통신, 무선 통신, 컴퓨터 네트워크, 병렬 및 분산 컴퓨팅>

최 상 방(평생회원)
1981년 한양대학교 전자공학과 학사 졸업.
1981년~1986년 LG 정보통신(주).
1988년 University of washinton 석사 졸업.
1990년 University of washinton 박사 졸업.
1991년~현재 인하대학교 전자공학과 교수
<주관심분야 : 컴퓨터 구조, 컴퓨터 네트워크, 무선 통신, 병렬 및 분산 처리 시스템>