멀티패스 통신을 위한 효과적인 패킷 스케줄링 기법

강형규, 홍충선*
경희대학교 컴퓨터공학과
e-mail : hkkang@networking.khu.ac.kr, cshong@khu.ac.kr

An Efficient Packet Scheduling Scheme for Multi-path Communication

Hyeong Kyu Kang, Choong Seon Hong*
Dept. of Computer Engineering, Kyung Hee University

요 약

멀티패스 전송은 단대 노드간 다중 경로를 동시에 사용함으로써 효과적인 대용량 전송을 실현하는 미래 인터넷 설계의 한 부분이다. 이에 따라 draft-'TCP Extensions for Multipath Operation with Multiple Addresses'에서 MPTCP 가 언급되었으며 기존 TCP 를 활용한 다중 전송에서 발생할 수 있는 다양한 오픈 이슈가 나오게 되었다. 본 논문에서는 위 논의로부터 나온 다양한 오픈 이슈 중 수신 측에서 발생 할 수 있는 패킷의 재조합(packet reordering)을 줄이는데 초점을 둔다. 패킷의 재조합은 불필요한 에너지 소비와 빈번한 패킷 재전송(packet retransmission) 문제를 초래하며, 특히 에너지 효율이 중요한 모바일기기에 있어 반드시 해결되어야 문제라 할 수 있다. 이를 해결하기 위해 본 논문에서는 전송 경로들의 RTT 값을 비교하여 패킷의 스케줄링 하는 기법을 제안하였으며 시뮬레이션을 통해 성능을 검증하였다.

1. 서론

최근 스마트폰과 같은 통신기능을 갖춘 다양한 전자기기들이 한 개 이상의 네트워크 인터페이스를 가지게 됨으로써 다중 경로를 통한 효과적인 데이터 전송을 생각해볼 수 있게 되었다. 하지만 이런 멀티 패스 전송은 기존의 TCP 기반으로 전송하기에는 공평성 문제, 패킷 재조합, 흐름 제어와 수신 버퍼 고갈 문제 등 해결되어야 할 많은 문제점을 가지고 있다.

사실 멀티패스에 대한 연구는 최근의 이슈라 할 수 없다. 2000 년 초반부터 멀티패스가 이슈화 되었으며, 2004 년 IETF 에의해 SCTP(Stream Control Transmission Protocol)[1]라는 멀티패스 전송 기술이 표준화 되었다. 하지만 위에서 언급하였듯이 멀티패스에 적합하지 않은 TCP 의 특징 때문에 SCTP 는 동시 다중 전송이 아닌 다중 경로 중 하나의 경로를 선택하여 데이터를 전송하고 나머지 경로는 예비 경로로 사용하는 기법을 사용하게 된다. 그 이후 CMT(Concurrent Multipath Transfer)[2]와 같은 동시 다중 전송을 지원하는 기술이 다양하게 제안되긴 하였으나 이런 기술들은 기존 TCP 기반이 아닌 동시 다중 전송을 위한 새로운 구조를 제안함으로써 현재 인터넷 망에 적용하기에는 한계점을 나타냈다.[1]

그러나 최근에 멀티패스의 중요성과 TCP 기반의 멀티패스 전송 방식이 필요성이 다시 화두가 되면서 MPTCP[3]가 제안되었다. 이는 호스트가 다중 주소(IP Address)를 보유함으로써 여러 개의 TCP 연결 세션을 구성하는 방식을 사용한다. 또한 Resource pooling[4]이라는 기능을 구현함으로써 기존에 문제가 있었던 공평성과 흐름제어 문제 등을 해결하고 있다. 하지만 여전히 멀티패스 전송 기술은 해결되어야 하는 많은 문제를 가지고 있으며 아직도 다양한 오픈 이슈가 연구되고 있다.

본 논문의 목적은 멀티패스 전송에서 발생할 수 있는 다양한 이슈 중에서 수신 측 도착 패킷에 대한 순서 재조합 문제를 해결하는데 중점을 둔다. 이를 위해 2 장에서는 멀티패스 전송으로부터 발생할 수 있는 순서 재조합 문제들을 열거하고 3 장을 통해 문제에 대한 해결 방안을 제시한다. 4 장에서는 앞에서 제안한 기법을 평가하고 마지막으로 5 장에서 결론을 내린다.

2. 연구문제

이번 장에서는 멀티패스 전송에서 발생할 수 있는 문제에 대해 살펴본다.

2.1. TCP 윈도우 프로토콜

여기서는 멀티패스 전송에 따른 문제를 살펴보기에 앞서 기존 TCP 의 윈도우 프로토콜의 원리에 대해서 살펴본다. 이를 위해 프로토콜의 경우를 unrestricted 와 stop-and-wait 로 나누어 살펴보며, TCP 윈도우 프로토콜 기능의 명확성을 보이기 위해 프레임의 손실과

* "이 논문은 2010 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임"
(No. 2010-0027645). Dr. CS Hong is corresponding author.

손상은 없다고 가정하고 수신 측에서의 ACK timer 는 무시하기로 한다.

다음 값은 프로토콜의 기능을 정의하기 위한 요소이다.

R = bit rate
S = signal speed
D = distance between the sender and receiver
T = time to create one frame
F = number of bits in a frame
N = number f data bits in a frame
A = number of bits in an acknowledgment
W = window size

TCP 윈도우 프로토콜은 위에서 언급하였듯이 두 가지 경우가 발생할 수 있다. 첫 번째 경우는 발신자 윈도우가 최대 크기에 도달하지 못하는 경우로서 이 경우는 발신 프레임에 대한 첫 번째 ACK (acknowledgment)가 도착했음에도 모든 프레임 전송이 아직 끝나지 않은 상태를 말한다.

두 번째 경우는 위와 반대로 모든 프레임을 전송하였을 때 아직 첫 번째 ACK 가 도착하지 않은 경우이다. 위 두 경우는 수학적으로 W*프레임을 보내기 위한 시간과 하나의 프레임을 보내고 ACK 를 받는데 걸리는 시간의 비교를 통해 구별할 수 있다. 하나의 프레임을 만들고 보내는 시간은 T+F/R 이 되며, W 만큼의 프레임을 보내는데 걸리는 시간은 W(T+F/R)이 된다. 그리고 프레임을 보내고 ACK 를 받는데 걸리는 시간은 단 방향 전송 시간인 T+D/S+F/R 의 2 배인 2(T+D/S+F/R)이 된다. 결국 이는 다음과 같이 비교 수식에 의해 두 가지 경우로 나누어 질 수 있다.

경우 1 (unrestricted protocol):

$$W\left(T+\frac{F}{R}\right) > 2\left(T+\frac{D}{S}+\frac{F}{R}\right)$$

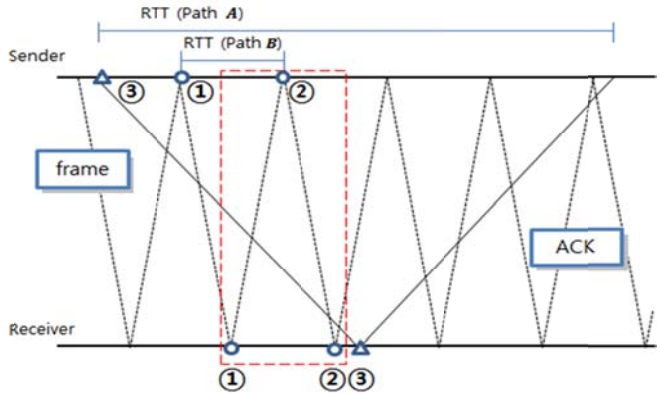
경우 2 (window-oriented stop –and – wait):

$$W\left(T+\frac{F}{R}\right) < 2\left(T+\frac{D}{S}+\frac{F}{R}\right)$$

TCP 기반의 멀티패스 구성에 있어 패킷의 재조합을 해결하기 위해서는 위 두 가지 경우가 고려되어야 한다. 즉, 전송되는 윈도우 사이즈 W 와 세그먼트 왕복 시간 RTT(Round Trip Time) = 2(T+D/S)+(F+A)/R 에 따른 수신 패킷의 동시 수신, 멀티 패스간 RTT 차이로부터의 패킷 순서 불균형 문제 등이 해결되어야 한다

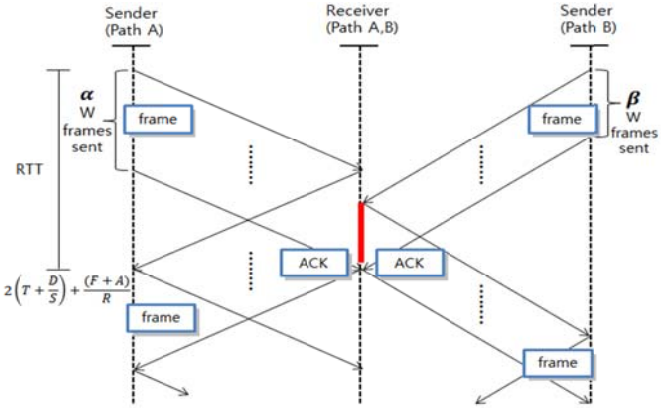
2.2. 멀티패스 상에서의 패킷 순서 불균형 문제

앞 절에서 언급하였듯이 기존 TCP 기반의 멀티패스를 구성하기 위해서는 몇 가지 고려되어야 하는 문제점이 있다고 하였다. 다음 (그림 1)과 (그림 2)는 멀티패스 상에서 패킷 순서 불균형 문제가 발생할 수 있는 그 예를 보여주고 있다. (그림 1)은 긴 RTT 를 갖는 Path A 와 짧은 RTT 를 갖는 Path B 가 존재할 경우 Path A 에서 보낸 프레임이 도착하기 전에 Path B 에서 두 개의 라운드가 진행됨으로써 패킷 순서가 어긋날 수 있다는 것을 보인다. 그림에서 볼 수 있듯이 패킷의 순서 균형을 유지하기 위해서는 Path A 에서 Path B 에 대한 다음 라운드에서 보낼 패킷①과 그 다음의 라운드 동안 보낼 패킷② 두 부분을 고려한 패킷 스케줄링이 이루어져야 한다



(그림 1) 두 개의 경로 사이에서의 RTT 라운드 차이에 따른 패킷 스케줄링

(그림 2)는 2 개의 패스 A, B 를 통해 데이터를 전송할 때, 패스 A 의 경우 짧은 RTT 시간과 긴 프레임 전송 시간을 가지며 패스 B 의 경우 긴 RTT 시간과 짧은 프레임 전송 시간을 가짐으로써 수신 측에서 중첩된 수신으로 인한 패킷 순서 불균형 문제가 발생할 수 있다는 것을 보인다. 이를 해결하기 위해서는 중첩되는 구간에서 최대한 순서에 맞게 수신할 수 있도록 발신 측에서 해당 구간의 범위를 예측하여 스케줄링 하여야 한다.



(그림 2) 멀티패스 전송에서의 중첩 수신

우리는 2.1 과 2.2 를 통해 멀티패스상에서 발생할 수 있는 패킷 순서 불균형 문제를 살펴보았다. 다음 장에서는 이상에서 보인 문제점을 고려하여 TCP 기반의 멀티 패스상에서 패킷 순서 균형을 이룰 수 있는 패킷 스케줄링 기법에 대해 제안하고 평가한다.

3. 제한사항

이번 장에서는 본 논문에서 제시하는 제한 사항으로 TCP 가 갖는 특성을 고려하여 멀티패스에서 패킷 재순서화를 최소화 하기 위한 기법을 소개한다.

3.1. 멀티패스상 RTT 기반 패킷 할당

TCP 전송에서 긴 RTT 를 갖는다는 것은 수신 측 ACK Timer 를 무시하였을 경우 전송지연이 길다는 것을 의미한다. 지연이 발생하는 이유에는 혼잡, 큐잉 딜레이 등 여러 가지 문제가 있을 수 있다. 이런 문제로 인해 패스간 RTT 의 차이가 생기게 되며, 심지어 이런 차이로부터 패스간 RTT 라운드의 횟수 차이가 발생하게 된다. 결국 이런 차이로부터 패킷 재조합 문제가 발생한다는 것을 2 장에서 확인하였다.

전송지연에 관련한 패킷 스케줄링 기법은 [5]에서 소개된바 있다. 하지만, CMT 기반으로 제안되었으며, RTT 라운드의 차이를 고려하지 않아 다중 경로 TCP 에 적용하는데 제약이 있었다. 따라서 본 논문에서는 각 패스의 RTT 값과 RTT 라운드의 차이를 비교하여 패킷 스케줄링 하는 기법을 제안한다.

우리는 송신과 수신 사이에는 N 개의 패스가 있다고 가정한다. 그리고 이 N 개의 패스는 RTT 가 짧은 순서대로 정렬되고 RTT 의 표기는 $[RTT_I^X]$ 와 같이 표기하게 된다. 여기서 I 는 정렬된 패스의 순서번호를 나타내며, X 는 RTT 라운드 번호를 나타낸다.

스케줄링은 크게 두 단계로 나뉘어진다. 첫 번째 단계는 RTT 라운드 카운팅 단계로서, 선택된 패스를 기준으로 자신보다 RTT 가 빠른 패스에 대한 포함 개수를 계산하는 단계이다. 과정은 다음과 같다:

1) 정렬된 패스 {1,…, i,…,N}중 바로 다음 RTT 가 시작되는 패스를 선택한다. 다음 RTT 가 시작하는 패스를 i 라고 하면, 패스 i 는 자신보다 RTT 가 빠른 {1,…,i-1}까지의 패스를 고려하여 스케줄링 하게 된다.
2) 빠른 RTT 패스는 패킷이 도착하는 시간 RTT/2 를 이용하여 패스 i 의 RTT/2 에 포함되는지 여부를 구하게 된다. 이때 주의점은 패스 i 보다 늦게 RTT 라운드가 시작하는 패스에 대해서만 생각하며 그 시간 차이는 d_I로 나타낸다.
3) 위 과정을 종합하여 식(1)을 얻을 수 있으며, 다음 식을 통해 패스 i 보다 빠른 RTT 패스들의 최대 포함 라운드인 X_n SET 을 구하게 된다. 식(1)에서 RTT 합의 계산은 마지막 라운드 RTT 만 절반의 값으로 계산되며, 그 이전 값들은 그대로 합산되게 된다. 그 이유는 (그림 1)의 ① ②의 RTT 를 보면 알 수 있듯이 마지막 RTT 라운드의 ACK 시간은 계산에 불필요하기 때문이다.

$$X_{n_{(n=1\ldots i-1)}} : \max\{d_{i-n} + \sum_{x=0}^{X_n-1} RTT_{i-n}^{x} + \frac{RTT_{i-n}^{X_n}}{2}\} < \frac{RTT_i}{2} \quad (1)$$

두 번째 단계는 앞 단계에서 계산된 RTT 개수 값과 각각의 해당 패스의 윈도우 사이즈, 패킷 사이즈를 함께 계산하여 최종적으로 순서에 맞는 데이터를 계산하는 단계이다. 이를 위해 우리는 식(2)로부터 식(4)를 유도한다. 식(2)는 윈도우 크기만큼의 해당하는 프레임을 생성해서 회선상에 데이터를 싣는데 까지 걸리는 시간을 나타낸 것이다.

$$\tau = W\left(T + \frac{F}{R}\right) \quad (2)$$

만약 프레임 생성 시간이 0 에 가깝다고 한다면 이 식은 식(3)과 같이 나타낼 수 있다. 결국 식(3)으로부터 전송하는데 걸리는 시간과 전송률을 곱하여 얻은 결과인 총 보낸 데이터의 양을 계산 할 수 있다.

$$\tau \cdot R = W \cdot F \quad (3)$$

최종적으로 첫 번째 단계에서 구한 패스 i 보다 RTT 가 빠른 패스의 X_n SET 과 식(3)으로부터 다음 식을 유도할 수 있게 된다.

$$\sum_{n=1}^{i-1}\sum_{x=0}^{X_n}(W_x^n \cdot F_n) \quad (4)$$

식(4)에서 n 은 RTT 가 짧은 순서로 정렬된 패스의 번호, x 는 라운드 번호를 나타낸다. W_x^n는 해당 패스의 RTT 라운드 동안의 윈도우 사이즈이며 이때 윈도우 크기는 이전 라운드의 윈도우 사이즈의 변화를 예상하여 계산 된다. 이때 계산은 TCP 의 AIMD (Additive Increase and Multiplicative Decrease)에 기반한다.

3.2. 중첩 수신 패킷의 순서를 고려한 패킷 할당

멀티패스 전송에서 발생하는 패킷 수신의 재조합 문제는 다중 경로를 통해서 동시에 패킷이 수신됨으로써 또한 발생할 수 있다. 그렇기 때문에 다중 경로를 통해 동시에 수신되는 패킷에 대한 스케줄링이 필요하다. 이 문제를 해결하기 위해서 우리는 [5]에서 제시된 확률 스케줄링 방법을 이용한다. 확률 스케줄링 은 각 경로의 대역폭의 상대 비율에 따라 패킷을 할당하는 기법으로 다음과 같이 나타낼 수 있다:

$$P_n = \frac{b_n}{\sum_{n=1}^{i} b_n} \ (1 \le i \le N) \quad (5)$$

식(5)는 모든 패스의 전송률의 합에 대한 각 패스의 전송률 비율에 확률로 나타낸 것이다. 하지만 이는 중첩 수신되는 경로의 집합을 고려하지 않았기 때문에 수식이 수정될 필요가 있다. 즉 다음과 같은 과정을 통해 수식이 구해져야 한다.

1) 각 패스의 식(2)전송시간과 RTT/2 전송 지연시간 그리고 RTT 시작시간을 계산한다.
2) 1)번 과정을 통해 중첩되는 시간을 계산하게 되는데 이 때 중첩 개수에 따라 여러 집합으로 분할한다. 예를 들어 처음에는 두 개가 중첩되고 잠시 후 세 개가 중첩될 수 있다. 그리고 다시 두 개가 중첩된다면, 이는 시간 흐름에 따라 두 개, 세 개, 두 개로 묶어 계산되어야 한다.

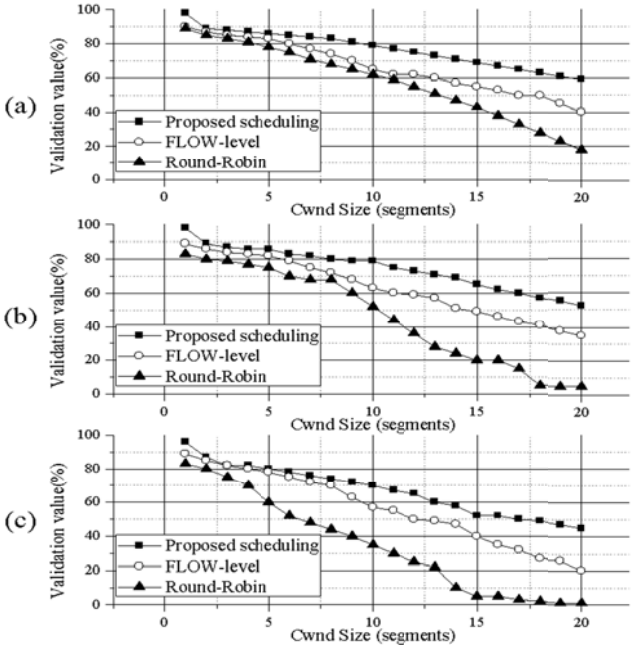
이런 과정을 거쳐 식(5)는 식(6)과 같이 수정되어야 한다. ‘set ‘은 2)에서 설명하였듯이 중첩 되는 묶음의 집합을 나타내며 b 는 전송률을 나타낸다.

$$P_n = \frac{b_n}{SUM(set\ b)} \ (1 \le i \le N) \quad (6)$$

결국 식(6)을 통해 패킷의 할당 비율을 패킷당 랜덤 확률을 적용하여 할당하게 된다.

4. 시뮬레이션

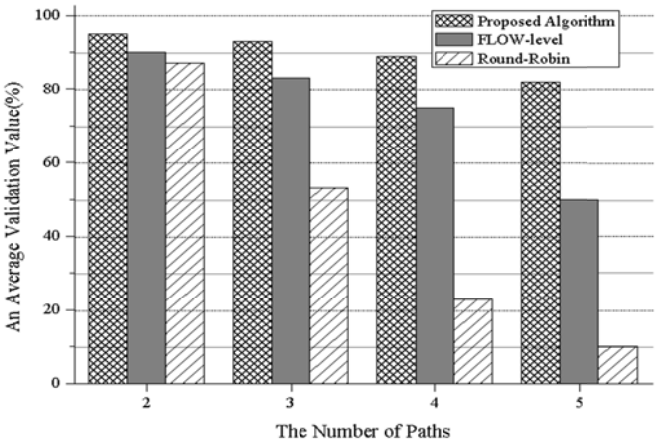
본 시뮬레이션에서는 MPTCP 의 기본적인 기능을 Omnet++[6]로 구현한 후 논문에서 제안하는 기법과 Round-Robin, FLOW-level scheduling 을 적용하여 성능을 비교하였다. 첫 번째 실험에서는 송신과 수신 노드 사이에 두 개의 패스를 구성하고 수신 측 최대 버퍼 사이즈는 1024 바이트로 설정하였다. 그리고 RTT 의 라운드 차이를 0, 1, 2 로 설정하여 패스의 라운드 차이에 따른 유효한 데이터 전송치를 측정한다. 여기에서 유효한 데이터 전송치란 순서에 맞게 잘 들어와서 최상위 계층까지 전달된 데이터의 수치를 의미한다. 만약 패킷 재조합과 버퍼 블로킹 문제가 발생하게 되면 데이터 전송이 많다고 하더라도 실제 유효한 데이터 전송치는 낮아질 수 있다.



(그림 3) RTT 라운드 차이와 혼잡 윈도우 사이즈 변화에 따른 유효 전송치

(그림 3)은 첫 번째 실험의 결과로 (a)부터 차례대로 RTT 차이가 0, 1, 2 일 때의 혼잡 윈도우 사이즈 변화에 따른 유효한 데이터 전송치를 나타낸다. 그림에서 알 수 있듯이 본 논문에서 제시한 스케줄링은 RTT 차이와 Cwnd 의 변화에도 완만한 유효 전송치를 보였다. 하지만 나머지 스케줄링은 Cwnd 가 증가함에 따라 유효 전송치는 매우 많이 떨어졌으며, 특히 Round-Robin 같은 경우 (b)와 (c)에서 전송 딜레이와 중첩 수신에 대한 경우를 전혀 고려하지 않아 급격히 떨어지는 현상을 보였다.

다음으로 두 번째 실험에서는 패스의 개수에 따른 전송치를 비교 하였다. 이 실험에서는 패스 개수에 따른 평균 전송치를 측정하였으며 이에 대한 결과는 (그림 4)와 같다. FLOW-level 과 Round-Robin 의 경우 각각 패스 개수가 3 개와 4 개에서 급격히 떨어지는 현상을 보였다. 반면 제안된 기법의 경우 패스가 증가 하더라도 80% 이상의 유효 전송치를 보였으며 다른 알고리즘과 비교하였을 경우 상당히 효과가 있었음을 보여줬다.



(그림 4)멀티패스 개수에 따른 평균 유효 전송치

5. 결론

우리는 미래인터넷 연구 주제인 멀티패스 전송 중 발생 할 수 있는 패킷 재조합 문제에 대해서 연구하였다. 재조합 문제는 단일 TCP 통신에서는 크게 문제가 되지 않을 수 있으나 멀티패스 전송에서는 성능을 급격히 떨어뜨리는 요소로 작용 할 수 있다. 즉, 본 논문에서는 이상의 문제로부터 멀티 패스의 성능을 극대화 하기 위한 기법을 소개 하였으며, 또한 이 기법을 Omnet++ 이용하여 Round-Robin, CWND-aware scheduling 과 비교하여 성능이 뛰어났음을 보였다.

향후에는 서두에서 제시했던 가정사항인 '변화가 적은 RTT 시간', '고정된 혼잡' 요소가 동적이게 고려되어 연구가 진행되어야 하며, 또한 본 연구가 MPTCP 에 적용될 수 있도록 MPTCP 에서 사용되는 혼잡 회피 기법이 함께 연구 되어야 할 것이다.

참고문헌

[1] Internet Engineering Task Force, "Stream Control Transmission Protocol", RFC4960, 2007

[2] Iyengar, J.R., Amer, P.D. and Stewart, R. "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths", IEEE/ACM Trans., Vol.14, No.5, pp.951-964, 2006

[3] A. Ford, "TCP Extensions for Multipath Operation with Multiple Addresses" draft-ford-mptcp-multiaddressed-03, 2010

[4] D Wischik, M Handley, MB Braun, "The Resource Pooling Principle", ACM SIGCOMM, Vol.38, No.5, Oct. 2008

[5] J Liao, J Wang, T Li, X Zhu, P, Zhang, "Sender-based multipath out-of-order scheduling for high-definition videophone in multi-homed devices, Consumer Electronics, IEEE Transactions on, Vol.56, pp.1466-1472, Oct. 2010

[6] Omnet++, http://omnetpp.org/