UCLA Computer Science Department Technical Report CSD-TR No. 040023

CapProbe based Passive Capacity Estimation

Ling-Jyh Chen, Alok Nandan, Guang Yang, M. Y. Sanadidi, Mario Gerla
3803H, Boelter Hall, Department of Computer Science, UCLA
Los Angeles, CA 90095, USA

***Abstract*-- Knowledge of the link capacity is essential to better manage and utilize networks. Most capacity estimation techniques nowadays employ "active" approaches to probe the networks and estimate the link capacity. In this paper, we study the feasibility of "passive" capacity estimation techniques based on CapProbe, a recently proposed fast, low overhead, and accurate capacity estimation tool. We embed the CapProbe algorithm in two popular protocols: TCP and TFRC, and evaluate a passive version of CapProbe using both Internet measurement and simulation. The results show that the passive approaches can estimate capacity accurately and rapidly over a wide range of system parameters. Some exceptions are encountered in the TCP case where estimation accuracy may be lower due to hardware and operating systems issues, particularly when the bottleneck capacity is large. In addition, we present an example of using passive capacity estimation in a mobile computing scenario, where we enhance TFRC performance by adapting its sending rate according to estimated capacity before/after a vertical handoff. The proposed passive capacity estimation is shown to be suitable for "on-line" usage with high convergence speed and minimal overhead.**

***Index terms*-- passive measurement, capacity estimation, packet pair, CapProbe**

I. INTRODUCTION

Increasing network heterogeneity, both in terms of access technologies as well mobile environments, presents new challenges. The ability to estimate network properties, such as the capacity of a path[1] is of particular importance since the knowledge of the capacity of a path can be useful for many applications. For instance, multimedia servers would be able to determine appropriate streaming rates; and routing protocols and multicast application level protocols can use capacity information to build optimal routes/trees.

Mobile environments present newer challenges as they demand *faster* and *less intrusive* techniques for path estimation. A mobile end-host might traverse multiple networks en-route and vertical handoffs would enable seamless connectivity. It would be particularly useful if mobile hosts can perform vertical handoff to the "best" access technology based on the capacity information. Once the capacity estimation can be done fast and accurately, it is also applicable for the Internet servers to detect the occurrences of vertical handoff events on Mobile Hosts, since vertical handoffs usually result in drastic changes in path capacity. Dynamic link capacity in network access technologies such as 1xRTT can also impact application performance. Capacity estimation will facilitate providing a better picture about the underlying network conditions and hence enable application adaptation. It is particularly critical when at the end of the path there is a mobile node that dynamically hands-off to different access networks (e.g., a vertical handoff from 802.11b connection to 1xRTT connection). The capacity of the path is generally the capacity of the wireless link. It is an upper bound on the reception rate at the mobile client, and its knowledge at a server (or proxy) enables it to adjust its sending rate, and, possibly, content, to match the reception rate. Also desirable is that the time to obtain accurate estimates be small. For instance, if capacity estimates need to be used by multimedia servers to determine streaming rates, a significant delay in estimating capacity would be unacceptable. The new challenge in future mobile applications is the *speed* required for the estimate in case the receiver (e.g., a car or a plane) moves rapidly across different wireless domains.

Apart from the *speed* of estimation, another desirable property of an estimation methodology is that it should be *non-intrusive*, i.e., the amount of probing data injected into the network should be minimal so that other traffic is not affected. Existing estimation tools are either active, passive, or both. Most often the decision of whether the tool will be active or passive is determined by how invasive we can allow the tool to be without affecting application performance. Scalability of the measurement tool (i.e. if a large number of instances of the tool are run *simultaneously*) is also affected by the mode of the measurement.

Active estimation is characterized by introduction of packets for measurement purposes. From an end-user's perspective active measurements to find the network path characteristics is a reasonable approach. However, the scalability of such measurements and its impact on application traffic can be severe, particularly if a large number of users start simultaneously start estimating the network path characteristics. Pathchar [10], Pathrate [5], and CapProbe [12] all employ active estimation techniques.

Passive estimation on the other hand is less intrusive. The motivation for passively estimating the path is that the transport protocol should detect any changes in network path without an application or end-user explicitly initiating an active measurement. Most of the capacity estimation tools can be both active as well as passive. There have been some proposals for tools that are a combination of the active and passive measurements for scalability purposes [16]. Passive estimation has its disadvantages as well since it is usually limited to periods of time when there is application traffic. Newer tools that combine active and passive measurements do passive measurements when there is application traffic and revert to active measurement after a prolonged period of application traffic inactivity, to ensure a level of accuracy in the network characteristics measured.

[1] Capacity of a path here refers to the minimum physical link capacity among all links on the path.

In this paper, we propose and evaluate a *passive* version of CapProbe. We evaluate the passive CapProbe techniques using Internet experiments. The passive CapProbe techniques are embedded in TCP and TFRC [7]. Aiming at the emerging wireless and mobile computing scenarios, we also evaluated the passive CapProbe techniques in the vertical handoff scenarios.

The rest of the paper is organized as follows. In section 2, we present related work and give an overview of CapProbe. In section 3, two passive capacity estimation techniques, TFRC Probe and TCP Probe, are proposed. In section 4, we evaluate the accuracy and convergence speed of the proposed passive techniques using Dummynet and Internet experiments. In section 5, we validate the proposed techniques through simulation. In section 6, we evaluate the proposed passive techniques in wireless and mobile scenarios, and we present a "fast rate adaptation" algorithm to better utilize the network resources in vertical handoff scenarios. Section 7 concludes the paper.

II. RELATED WORK

Earlier approaches for estimating capacity relied on either delay variations among probe packets as in pathchar [10], or dispersion among probe packets as in Nettimer [15] and Pathrate [5]. Other tools such as pchar and clink [6] use variations of the same idea as pathchar. Pchar also uses regression to determine the slope of the minimum RTT versus the probing packet size. In [15], the authors propose kernel density techniques to statistically filter packet pair dispersions. Paxson showed that the dispersion distribution can be multi-modal (e.g. in multi-channel links), and proposed the use of Packet Bunch Modes, a technique involving packet trains of different lengths [20]. Dovrolis' [5] analysis clearly revealed that the dispersions distribution can indeed be multi-modal without multi-channels, and that the strongest mode in the multimodal distribution of the dispersion may correspond to either (1) the capacity of the path, or (2) a "compressed" dispersion, resulting in capacity over-estimation, or (3) the Average Dispersion Rate (ADR), which is lower than the capacity.

Pathrate [5], a tool resorting to increasing packet train lengths to induce the so-called Sub-Capacity Dispersion Range (SCDR) was developed by Dovrolis to select the appropriate capacity mode. Pathrate uses statistical techniques to identify the first mode following this range, and this mode provides a rather accurate estimate of the capacity. Pathrate and its predecessors rely on histogram construction with their inherent sensitivity to appropriate bin sizing, and require post-processing time delays for mode identification. Pathrate is an active estimation technique, which sends packet trains. Packet train methods may be intrusive particularly if many users simultaneously apply them. Thus, incorporating Pathrate or the active version of CapProbe into a transport protocol to enable widespread estimation might create scalability issues. Finally, Pathchar like tools have limitations with respect to the speed of estimation process as shown in [12].

A. CapProbe

CapProbe is a newly proposed capacity estimation technique, which has been shown to be fast and accurate over a large range of scenarios [12]. CapProbe requires only packet pair probing, and does not need packet trains. More importantly, CapProbe combines the use of dispersion measurements and end-to-end delay measurements to filter out packet pair samples distorted by cross traffic. In summary, a packet pair, which is launched into the network back to back, is dispersed at the bottleneck link according to the bottleneck capacity. The smaller the link capacity is, the larger the dispersion. If such dispersion would arrive at a destination unperturbed, it will be ideal for identifying the bottleneck capacity. The dispersion of those "distorted" samples might be either expanded or compressed, where "expansion" of dispersion leads to under-estimation and "compression" of dispersion leads to over-estimation of the capacity as showed in Figure 1.

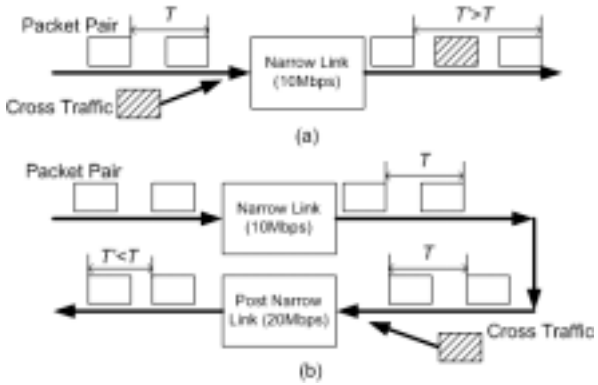

Figure 1: (a) Under-Estimation caused by "expansion" (b) Over-Estimation caused by "compression"

CapProbe is based on a simple and fundamental observation that both expanded and compressed probing samples must have experienced queuing delays on either the first or the second packet of the probing packet pair. This means that whenever an incorrect value of capacity is estimated, the sum of the delays of the packet pair packets, called the delay sum, includes cross-traffic induced queuing delay. A delay sum, which does not include any cross-traffic queuing delay, is referred to as the minimum delay sum. The dispersion of such a packet pair sample is not distorted by cross-traffic and will reflect the correct capacity. This sample can easily be identified since its delay sum will be the minimum among delay sums of all packet pair samples.

The capacity is thus estimated by the equation:

$$C = P / T \qquad (1)$$

where P is the sampling packet size, and T is the dispersion of the sample packet pair of the minimum delay sum.

Comparing with other capacity estimation techniques, such as pathchar [10] and pathrate [5], CapProbe is a computationally inexpensive scheme that only needs to keep track of the minimum delay sum, and the dispersion of the pair that produced it. CapProbe has no need to keep dispersion histograms, and it estimates capacity without any post-

processing; whereas other schemes perform post-processing to analyze the history of all samples information using statistical methods. The simplicity natural to CapProbe makes it amenable to be integrated in other data transmission protocols with minimal modification. All that is required is that the protocol data traffic includes back to back transmitted packets. Round trip estimation also has minor requirements on acknowledgments. The first implementation of CapProbe took advantage of ICMP where every ICMP request will be replied to immediately. However, such "out of band" capacity estimation sometimes is not appreciated in the mobile computing environments, especially when the link capacity is fairly small. In this paper, we will show the integration of CapProbe with other existing data transmission protocols, such as TCP and TFRC, so that the link capacity can be passively estimated. The details of the integration design will be presented in the following subsections, and the evaluation of the proposed integration will be shown in the simulation and measurements experiments sections.

III. PASSIVE CAPACITY ESTIMATION

In this section, we introduce two passive capacity estimation techniques based on CapProbe, namely TFRC Probe and TCP Probe. TCP-Friendly Rate Control (TFRC) and TCP are two popular transport protocols. We embed the CapProbe technique into these protocols so that they can benefit from capacity estimation as we will see shortly.

A. TFRC Probe

TCP-Friendly Rate Control (TFRC) is an equation based unicast multimedia streaming protocol proposed in [7]. TFRC mimics the TCP long-term throughput by utilizing the response function [19]:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)} \quad (2)$$

where T is the upper bound of the sending rate. T is a function of the packet size s, round trip time R, loss event rate p, and the TCP retransmission timeout value t_{RTO}.

The design goal of TFRC is flow controlled and TCP friendly transport of data and streams requiring no strict error control. Therefore, TFRC increases the sending rate slowly, rather than aggressively seeking out available bandwidth. For example, the maximum increase of the sending rate is 0.14 packets/RTT, or 0.22 packets/RTT with history discounting [7]. On the other hand, TFRC is also designed to slowly respond to data loss events, not to cut down the sending rate drastically upon every single loss event.

In order to achieve smoother data transmission in TFRC, the sender and the receiver are required to cooperate with each other. The sender is responsible for computing the smoothed round-trip time R using an exponentially weighted moving average, and determining the retransmit timeout value t_{RTO}. The sender is also responsible for adjusting its sending rate T_{actual} to be close to T, which is derived from the equation.

On the other hand, the receiver is responsible for calculating the loss event rate p and sending the information back to the sender once per round-trip time. The loss event rate is obtained by maintaining an array of the last eight loss intervals. This loss interval array is continuously updated and a weighted average of the loss intervals is computed. The reported loss event rate p is defined as the inverse of the weighted average.

Passive capacity estimation can be added to TFRC simply by sending a portion of data packets back-to-back and estimating the link capacity based on the measured dispersion and end-to-end delay, i.e. following the fundamental concepts of CapProbe. We call this extended TFRC with passive capacity estimation capability *TFRC Probe* in this paper.

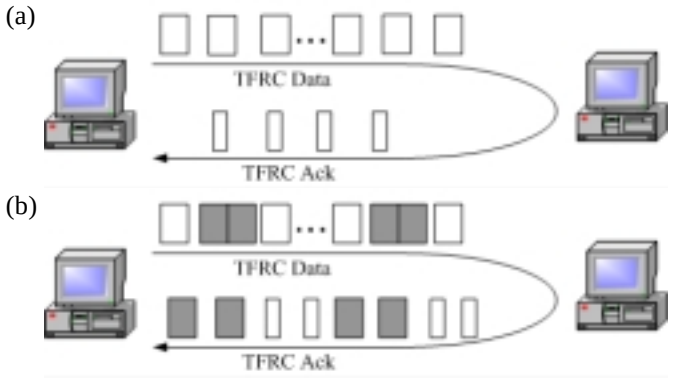

Figure 2: (a) original TFRC (b) TFRC Probe (the gray ones are back-to-back sampling packets and ACKs)

Figure 2 compares the difference between TFRC and TFRC Probe. In the original TFRC, as shown in Figure 2-a, transmission of data packets is paced and evenly distributed, based on the computed sending rate. This is beneficial to multimedia streaming applications which require a smooth and stable sending rate. For the purpose of CapProbe-based capacity estimation, however, paced transmission lacks the packet pairs that are crucial in the scheme. Additionally, the TFRC receiver sends only one ACK packet every RTT, while CapProbe requires the ACKs to be sent out immediately after the data packets are received. TFRC Probe must make a few necessary modifications in order to carry out effective capacity measurements.

The first modification in TFRC Probe is that after every n^{th} data packet is sent out, the TFRC Probe sender immediately transmits the next data packet without waiting for the pacing interval. In other words, TFRC Probe creates a back-to-back sampling packet pair every n packets. The default value of n is set to *20* in our experiments. The TFRC Probe receiver must acknowledge these occasional sampling packets, individually and immediately, upon their receipt, with ACKs that are equal in size to the data packets. When the ACKs arrive at the sender, it is able to estimate the capacity using Equation 1. Other than these changes, TFRC Probe operates in the same way as the original TFRC. Figure 2-b highlights the differences between the two schemes.

Since TFRC Probe changes the behavior of TFRC receiver (i.e., it immediately acknowledges the individual sampling packets using large ACKs at the size of the data packets), it

increases the amount of traffic on the reverse path, namely from the receiver to the sender. The increased traffic overhead is $\frac{A}{n/2}$ bps, where A is TFRC Probe data achieved rate (i.e. throughput), and n is the number of data packets between each sampling. In case that the reverse link can not afford the increased traffic overhead, the TFRC Probe sender can either decrease A or increase n to reduce the overhead.

The idea behind TFRC Probe is not restricted to the TFRC protocol only. The passive capacity estimation concept can be seamlessly applied to other UDP based application protocols (e.g. RAP, RTP, UDP based FTP and P2P file downloading) and the emerging data transmission protocols (e.g. DCCP [14]).

B. TCP Probe

TCP is the most popular transport protocol on the Internet today. Various congestion control algorithms have been extensively studied to better utilize the network resources. Knowledge of bottleneck link capacity can be helpful for TCP congestion control algorithms to determine an upper bound of the congestion window size. Additionally, several recent TCP studies, such as TCP Westwood [24] and Agile TCP [23], adjust their congestion window size "agilely" according to some well-designed functions of link capacity, achieved rate, queuing delay, and error rate estimate. It is obvious that such TCP variants will be benefited if the bottleneck link capacity can be estimated accurately and fast. Inspired by the simple and accurate CapProbe algorithm, it soon becomes interesting (and useful) if a CapProbe based passive capacity estimation extension can be embedded within TCP. The top design principle of such an extension is *simplicity*, i.e., changes to the existing TCP protocol should be minimal and preferably sender-side only. The extension should also be applicable to any existing TCP variant. To identify the TCP with passive capacity estimation, we call it *TCP Probe* hereafter.

Similar to TFRC Probe, the basic idea of TCP Probe is to send a portion of the data packets back-to-back as sampling packets. Fortunately, TCP does send some back-to-back data packets, mostly in *Slow Start* and also sometimes in *Congestion Avoidance*, upon receipt of an ACK packet. Once two ACKs of a packet pair are received by the TCP sender, the dispersion information can thus be used to estimate the path capacity using the CapProbe approach.

However, in order to confine the modifications required in TCP Probe to the sender-side only, and require no change at the TCP receiver, two problems arise. One of them is caused by the widely deployed "delayed ACK" [2], and the other is due to the different packet sizes of TCP data packets and ACKs.

Delayed ACK has been popularly deployed on most of Internet hosts. A TCP receiver with delayed ACK installed will acknowledge the received data on every other data packet. Therefore if two TCP data packets *i* and *i+1* are sent back-to-back from the sender, either packet *i* or *i+1* will be acknowledged, as shown in Figure 3-a.

The problem caused by delayed ACK can be solved by the "*inverted packet-pair*" technique. More specifically, when TCP needs to send back-to-back data packets with sequence numbers *'i'* and *'i+1'*, it swaps the sending order, i.e., packet *'i+1'* is sent before packet *'i'*. This swapped sending order will generate back-to-back ACKs on sequence numbers *'i-1'* and *'i+1'*. The delayed ACK receiver is, thus, forced to send an individual ACK for each data packet, as shown in Figure 3-b. The dispersion, *T*, of these two ACK packets are measured and the capacity estimation is therefore obtained by applying Eq. 1 with *P* equals to the size of the TCP data packet (instead of the ACK packet size) Note that this enhancement is applicable to all TCP variants.

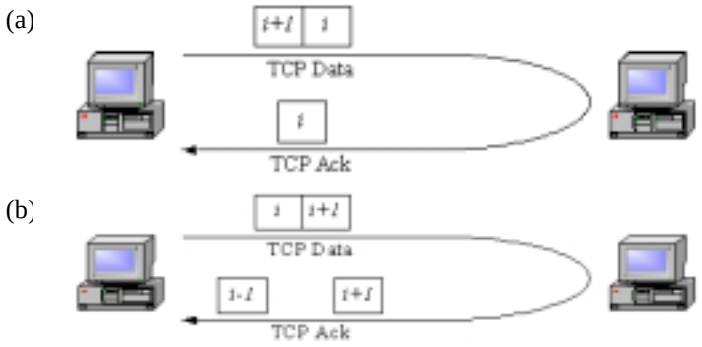

Figure 3: (a) back-to-back TCP packets will only be acknowledged once due to Delayed ACK; (b) *inverted* back-to-back TCP Probe packets will both be acknowledged

Moreover, since TCP Probe keeps the TCP receiver unchanged and does not enlarge the ACK packet size, the relatively small ACK packets (about 40 bytes) may result in inaccurate capacity estimation. From the previous study [12], the dispersion tends to become *compressed* if the size of the packet pair is small. Furthermore, due to the resolution limit and processing overhead of the operating system and other system issues, the dispersion measurement becomes more likely to be distorted. In a word, the capacity estimation tends to be less accurate. We also note that in addition to the above considerations, we are assuming symmetric forward and backward paths. Therefore the capacity estimated by data packet dispersion on the forward path, or measured by ACK dispersion on the backward path, would be the same. The case of asymmetric forward and backward paths will be addressed in a subsequent work.

In the rest of the paper, we will present a set of Internet measurements, supported by some simulations, to evaluate the performance of the proposed passive estimation techniques, namely TFRC Probe and TCP Probe, in terms of accuracy and speed. A real wireless application of such passive capacity estimation will also be introduced.

IV. EXPERIMENTS

In this section, we present experimental results to evaluate the accuracy and speed of the proposed passive capacity estimation techniques. We have implemented TFRC Probe by

modifying the TFRC implementation in [9] based on the algorithm presented in section 3.1. TCP Probe, in turn, has been implemented by modifying the TCP module in the Linux kernel (version 2.4.22) and following the algorithm proposed in section 3.2. We first perform a set of experiments in the "better-controlled" testbed environment to calibrate and verify the correctness of the proposed techniques and the implementation. We then move the experiments to the Internet, to for an evaluation in a more diverse and realistic scenario. To simplify the evaluation and better present the nature of the passive estimation results, we show the estimated capacity of using 20, 50, 100, 200, and 500 samples, instead of applying the convergence test as proposed in [12] in each test.

A. Dummy Net Experiments

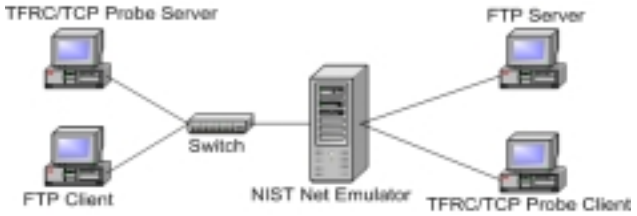

Figure 4: Testbed for Dummy Net Experiment

The first set of experiments is performed in the testbed configuration shown in Figure 4. The NISTNet emulator [17] is used to emulate bottleneck links of arbitrary capacities on the connection path. The cross traffic is generated by downloading a huge file from the FTP server to the client; the FTP connection shares the bottleneck link with the foreground TFRC/TCP Probe connection.

Different capacities (1, 5, 10, 20Mbps) are configured on the emulated NISTNet bottleneck link. For each capacity value the experiment is repeated three times. In each test run, we collect the estimated capacity after using 20, 50, and 100 samples, respectively, and show the results in Table 1 and Table 2.

From the experiment results, both TFRC Probe and TCP Probe estimate the bottleneck capacity accurately and fast. In all test cases, the passive estimation techniques are able to measure the bottleneck capacity within 20 samples. The estimated capacities are always within 10% of the actual values.

Table 1: NISTnet lab Measurements with TFRC Probe

Bottleneck Capacity		Run 1	Run 2	Run 3
1Mbps	20 samples	0.932	0.952	0.952
	50 samples	0.935	0.941	0.936
	100 samples	0.935	0.941	0.936
5Mbps	20 samples	4.822	4.785	4.816
	50 samples	4.822	4.820	4.816
	100 samples	4.822	4.832	4.820
10Mbps	20 samples	9.214	9.046	9.224
	50 samples	9.350	9.185	9.467
	100 samples	9.350	9.152	9.467
20Mbps	20 samples	19.025	18.865	18.572
	50 samples	19.025	18.865	19.156
	100 samples	19.112	18.865	19.110

Table 2: NISTnet lab Measurements with TCP Probe

Bottleneck Capacity		Run 1	Run 2	Run 3
1Mbps	20 samples	0.996	0.943	0.996
	50 samples	0.996	0.996	0.996
	100 samples	0.996	0.996	0.996
5Mbps	20 samples	5.038	4.854	4.919
	50 samples	4.884	4.623	4.919
	100 samples	4.850	4.623	4.192
10Mbps	20 samples	10.633	9.765	9.879
	50 samples	10.633	9.765	9.785
	100 samples	9.925	9.765	9.785
20Mbps	20 samples	19.667	20.000	19.368
	50 samples	19.347	19.399	19.522
	100 samples	19.347	19.385	19.004

Unit: Mbps

B. Internet Experiments

A set of Internet experiments were also performed to evaluate the proposed techniques in a more diverse and realistic scenario. Three Internet paths are selected for the experiments and are reported in the tables below. The path labeled UCLA is a local path on UCLA campus with 100Mbps bottleneck capacity, NTNU is a global path from US to Taiwan with 100Mbps bottleneck capacity, and DSL is a domestic DSL path with its bottleneck capacity at around 130Kbps.

For each path, the experiment is repeated three times. In each run, the estimated capacity is collected after 20, 50, 100, 200, and 500 samples, respectively. Since the estimated capacity converges to a stable value much faster in TFRC Probe than in TCP Probe, we only present the results of using 20, 50 and 100 samples in TFRC Probe experiments in Table 3. For TCP Probe, however, we list results at all sampling frequencies in Table 4. Note that, we only evaluate TCP Probe on UCLA and NTNU paths, because DSL path is asymmetric with 512Kbps bottleneck capacity in the forward direction and 130Kbps capacity in the backward direction

From the results, TFRC Probe always estimates the bottleneck capacity fast and accurately; it is able to get an accurate estimate within 20 samples in almost all test cases. The estimation results of TCP Probe, on the other hand, oscillate and do not converge to the actual capacity until 500 samples are obtained.

The reason for the performance gap between TFRC Probe and TCP Probe may be due to the different ACK packet sizes. In TFRC Probe, the size of the ACK packets which correspond to the back-to-back probing packets was chosen by us to be the same as that of the data (probing) packets, 1500 bytes in our experiments. This follows the original CapProbe design. In contrast, in TCP Probe the ACK size is dictated by the standard, namely 40 bytes.

Table 3: Internet experiment results of TFRC Probe

		Run 1	Run 2	Run 3
UCLA 100Mbps	20 samples	93.0 M	86.7 M	95.7 M
	50 samples	89.7 M	91.7 M	95.7 M
	100 samples	93.6 M	91.7 M	95.7 M
NTNU 100Mbps	20 samples	81.1 M	74.6 M	89.5 M
	50 samples	70.1 M	99.7 M	89.5 M
	100 samples	88.2 M	99.7 M	89.5 M
DSL 130Kbps	20 samples	135 K	136 K	130 K
	50 samples	133 K	130 K	132 K
	100 samples	133 K	130 K	132 K

Unit: bps

Table 4: Internet experiment results of TCP Probe

		Run 1	Run 2	Run 3
UCLA 100Mbps	20 samples	51.0 M	81.3 M	49.9 M
	50 samples	56.0 M	50.3 M	51.8 M
	100 samples	48.8 M	48.8 M	49.8 M
	200 samples	90.3 M	55.6 M	82.6 M
	500 samples	94.9 M	98.6 M	96.0 M
NTNU 100Mbps	20 samples	104.2 M	101.3 M	378.7 M
	50 samples	94.7 M	65.9 M	68.5 M
	100 samples	94.7 M	101.3 M	68.5 M
	200 samples	97.9 M	119.8 M	93.1 M
	500 samples	97.9 M	86.2 M	95.6 M

Unit: bps

The influence of packet size on capacity estimation has been studied in [12] [13]. The considerations of packet sizes in these studies apply to TFRC Probe. Probing packets of different sizes will experience different network dynamics and result in different estimation. Moreover, packets of small sizes tend to suffer the time granularity problem. For example, hardware and operating systems usually have limitations on the supported time resolution, which in turn limits the precision of timestamps. To better understand and verify the correctness of TCP Probe, we carry out supplementary simulations and present the results in the next section.

V. SIMULATION

In this section, we use simulation to verify the proposed passive techniques in a variety of configurations. A set of simulations are performed to evaluate the accuracy and speed of the capacity estimation, and different types of cross traffic are used to simulate different network dynamics. In the end, we compare the simulation to the measurement results and discuss potential challenges of passive capacity estimation.

Two passive capacity estimation methods (TFRC Probe and TCP Probe) were implemented in NS-2 simulator [18], where TCP Probe is an extended version of TCP NewReno. The topology we used in the simulation is shown in Figure 5, in which the bottleneck link (between node 3 and 4) is shared by all flows. The passive capacity estimation is performed on the path from node *1* to node *6*, and the cross traffic (if it exists) is generated from node 7 to 10, 8 to 9, 11 to 14, and 12 to 13 respectively.

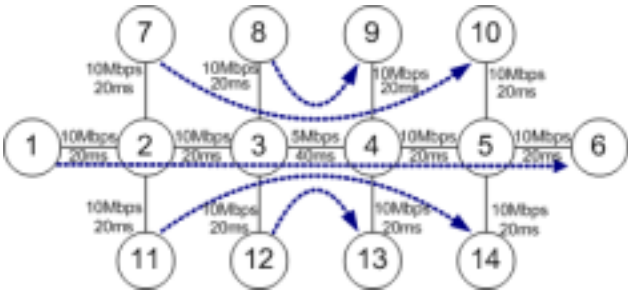

Figure 5: Simulation Scenarios

A. *Accuracy and Speed of Capacity Estimation*

In order to evaluate speed and accuracy of our probes with more diverse configurations, 3 types of cross traffic are employed in the simulation as shown in Table 5, with type 3 being Long Range Dependent [22]. Different capacity values are assigned on the link from node 3 to node 4 to create the bottleneck link. In each experiment, we collect the estimated capacity after 20 samples and 50 samples in order to evaluate the speed of the estimation, and we show the results in Table 6 and Table 7 below.

Cross Traffic	Description
Type 1	4 FTP flows (from node 7 to 10, 8 to 9, 11 to 14, and 12 to 13); 1500 bytes/packet
Type 2	4 CBR flows (from node 7 to 10, 8 to 9, 11 to 14, and 12 to 13); 500 bytes/packet; 80% load on the bottleneck
Type 3	16 Pareto flows with alpha = 1.9 (4 flows from node 7 to 10, 4 flows from 8 to 9, 4 flows from 11 to 14, and 4 flows from 12 to 13); 1000 bytes/packet; 80% load on the bottleneck

Table 5: Types of cross traffic

From the results, both TFRC Probe and TCP Probe show high accuracy in all the test cases, regardless of different types of cross traffic. In addition, in each test, it is always able to get the accurate estimation using just 20 samples. The results indicate that the proposed passive capacity estimation techniques are able to detect link capacity very accurate and fast.

Table 6: Accuracy of TFRC Probe capacity estimation

		Bottleneck Capacity			
		100 Kbps	500 Kbps	1 Mbps	5 Mbps
w/o CT	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M
w. CT Type 1	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M
w. CT Type 2	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M
w. CT Type 3	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M

Table 7: Accuracy of TCP Probe capacity estimation

		Bottleneck Capacity			
		100 Kbps	500 Kbps	1 Mbps	5 Mbps
w/o CT	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M
w. CT Type 1	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M
w. CT Type 2	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M
w. CT Type 3	20 samples	100 K	500 K	1 M	5 M
	50 samples	100 K	500 K	1 M	5 M

Unit: bps

B. Discussion

Comparing the simulation with measurement results shown in section 4, we note that the capacity estimation process of TCP Probe is slow and relatively unstable only in the Internet experiments. The reason that affects the performance of TCP Probe in Internet experiments is most likely due to the smaller ACK packet size. Recall that in TFRC Probe, the ACK packets that correspond to the back-to-back probing packets are of the same size as data packets. In TCP Probe, however, we intend to keep the changes as sender-side only and do not use special ACKs for probing data packets. These ACKs are typically 40 bytes each, much smaller than the regular data packets.

As reported in [12], packets of different sizes will experience different network dynamics. While operating in the Internet, the network dynamics are much more diverse than in the controlled NISTNet environment or in the simulator. The chance that both packets in a packet pair receive no buffering is smaller in the Internet. Moreover, smaller packets are more susceptible to the network dynamics, resulting in less accurate sampling results and thus a slower convergence speed to the correct capacity value. Since the algorithm of CapProbe relies on obtaining such an un-buffered packet pair sample, the estimation process of TCP Probe, using small ACK packets, is slower.

VI. APPLICATIONS

In this section, we present two applications of using passive capacity estimation in the wireless and mobile computing scenarios. In 6.1, we use TFRC Probe to passively monitor the capacity of wireless links. In 6.2, we use TFRC Probe to detect the occurrences of vertical handoffs, and we evaluate the performance of a "fast rate adaptation" algorithm in TFRC Probe where the vertical handoff notification is available. The experimental results are presented below.

A. "On-line" Monitor For Wireless Link Capacities

Knowledge of wireless link capacity is important for network management, pricing, and QoS support. However, differing from the wired links, properties of wireless links always change very dramatically and frequently. It soon becomes desired to have a scheme that enables "on-line" monitoring of wireless links with minimal overhead.

The passive capacity estimation techniques proposed in this work are appropriate for this purpose. In the following experiments, we evaluate TFRC Probe on 802.11b links (indoor environments with 11Mbps, 5.5Mbps, 2Mbps, and 1Mbps modes), and 1xRTT wireless links. The experiment results are shown in Table 8, where we report the estimated capacity using 20, 50, and 100 samples respectively in each test run.

Table 8: Wireless experiment results of TFRC Probe

		Run 1	Run 2	Run 3	Run 4	Run 5
802.11b 11Mbps mode	20 samples	7.9 M	7.9 M	8.0 M	7.9 M	8.0 M
	50 samples	8.0 M	8.0 M	7.9 M	8.2 M	8.0 M
	100 samples	8.0 M	8.0 M	7.9 M	7.8 M	7.9 M
802.11b 5.5Mbps mode	20 samples	3.7 M	3.9 M	3.8 M	3.8 M	3.8 M
	50 samples	3.7 M	3.7 M	3.8 M	3.8 M	3.8 M
	100 samples	3.9 M	3.9 M	3.8 M	3.8 M	3.7 M
802.11b 2Mbps mode	20 samples	1.7 M	1.4 M	1.8 M	1.7 M	1.8 M
	50 samples	1.7 M	1.4 M	1.8 M	1.7 M	1.8 M
	100 samples	1.7 M	1.4 M	1.7 M	1.7 M	1.4 M
802.11b 1Mbps mode	20 samples	858 K	933 K	859 K	911 K	897 K
	50 samples	858 K	933 K	903 K	911 K	898 K
	100 samples	858 K	933 K	873 K	903 K	898 K
1xRTT 150Kbps	20 samples	42.3 K	53.8 K	59.2 K	49.3 K	98.7 K
	50 samples	45.2 K	53.8 K	24.7 K	65.8 K	65.8 K
	100 samples	36.8 K	49.4 K	24.7 K	72.6 K	84.6 K

Unit: bps

We should also mention here that the effective capacity is usually smaller than the physical channel capacity in 802.11b and 1xRTT connections. For 802.11b connections, the effective capacity is reduced due to the MAC layer overhead (e.g. RTS/CTS packets and CSMA/CA mechanisms); whereas for 1xRTT connections, the effective capacity is determined by the number of assigned channels which are determined by the base station (ISP).

From the experiment results, TFRC Probe always estimates 70%~90% of the channel capacity on 802.11b links, which is very close to the analytical results of effective capacity presented in [25]. Moreover, the capacity estimation is very fast such that it is able to estimate the accurate capacity within 20 samples. On the other hand, for 1xRTT links, the estimated capacities vary a lot in the five runs. This is because the effective capacity of 1xRTT is determined by the channel allocation algorithm on the base station (ISP), which will change very frequently according to the traffic loads and the number of customers in the same area. Though there is no way to know the exact allocated channel capacity from the ISP, the estimated capacities are all within a reasonable range and are able to reflect the current allocated channel capacity.

Due to the high estimation accuracy, fast speed, and low overhead (i.e. no "out-of-band" traffic is generated in the passive techniques), TFRC Probe thus becomes a good "on-line" tool for monitoring a wireless link capacity. In the following subsections, we extend the same idea to detect the occurrences of vertical handoffs by monitoring the changes of

estimated capacity. Furthermore, we propose a "fast rate adaptation" for TFRC Probe to improve its throughput performance in vertical handoff scenarios.

B. *Vertical Handoff*

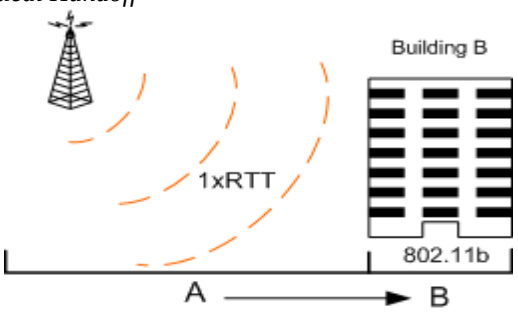

Figure 6: A vertical handoff scenario

In this section, we study the application of TFRC Probe in vertical handoff scenarios. A *vertical handoff* is a process of switching the ongoing network connection from one network interface/technology to the other [21]. For example, when a mobile device moves out of the 1xRTT network and into an 802.11b network (as shown in Figure 6), the handoff event would be considered as vertical.

A vertical handoff usually results in a drastic change in the link capacity. For instance, a vertical handoff from 1xRTT to 802.11b can easily witness around 100-fold increase in the channel capacity (from 150 Kbps to 11Mbps). However, though most data transmission protocols employ AIMD-based congestion control algorithms to adjust its sending rate according to the network dynamics, the adaptation process can not take aggressive advantage of the rapid change of resources in such vertical handoff scenarios [8]. For example, when a handoff occurs from LOW capacity to HIGH capacity (e.g. 1xRTT to 802.11b), no congestion loss is detected. An AIMD-based scheme will remain in *congestion avoidance* phase and linearly increase its congestion window (or sending rate) to probe the available bandwidth. In the opposite direction, when a handoff occurs from HIGH to LOW (e.g. 802.11b to 1xRTT), there is immediate packet loss at the moment of the handoff, and AIMD protocols will react promptly to such loss.

It is obvious that the application performance would benefit if they can be notified of the occurrence of vertical handoff [8]. For example, the AIMD-based protocols can be forced to enter the *slow start* phase when it is notified of the occurrence of a vertical handoff from LOW to HIGH capacity. Additionally, they can be forced to slow down their send rates in advance, as long as the notification of a vertical handoff from HIGH to LOW capacity can be predicted.

By monitoring the capacity of the ongoing wireless link, it soon becomes possible to detect the occurrence of a vertical handoff. Using the passive capacity estimation techniques, a vertical handoff notification can be generated when a drastic change in the estimated capacity is detected. In the following experiments, we tackle that the case of a vertical handoff from LOW to HIGH capacity link (e.g. 1xRTT to 802.11b). Besides detecting the occurrence of vertical handoff, a "fast rate adaptation" scheme is also proposed to force TFRC Probe to enter slow start phase while the detected vertical handoff is from low capacity link to high capacity link. Figure 7 shows the algorithm for detecting vertical handoff occurrences and "fast rate adaptation".

```
In recv_reports function on the TFRC sender:
// this function will be called when ACK
// packets are received

// initialization
n = 0; C_old = 0; Delay_SUM = INFINITE;

while (True) {
  if this is an ACK of TFRC Probe sampling packet {
    if this is the ACK of the first packet {
      measure RTT1;
    } else {
      measure RTT2;
      n++;

      // Perform CapProbe Algorithm
      if (Delay_SUM > RTT1 + RTT2){
         Delay_SUM = RTT1 + RTT2;
         C = PacketSize / (RTT2 – RTT1);
      }
    }
  }
  if (n==20) {
    // report the capacity every 20 samples
    if (C>5*C_old) {
      // generate a vertical handoff
      // notification and force TFRC to enter
      // slow start phase
      MODE = SLOW_START;
    }
    C_old = C;
    // reset CapProbe variables
    n = 0;   C = 0;  Delay_SUM = INFINITE;
  }
}
```

Figure 7: The algorithm of detecting vertical handoff and "fast rate adaptation" using TFRC Probe

In the deployed vertical handoff detection algorithm, the capacity estimation results are reported every 20 samples, which is based on the observation of Table 8 such that TFRC Probe can accurately estimate capacity using 20 samples in most test cases. Moreover, a vertical handoff notification is generated upon drastic change in estimated capacity, i.e. the new estimation is five times larger than the previous capacity estimation.

To provide seamless vertical handoff, a testbed was created using USHA [4]. Figure 8 shows the experiment results, where a seamless vertical handoff was performed from 1xRTT to 802.11b at around 30th sample.

In Figure 8, the capacity estimation, showed in dashed line, is reported after the 20th, 40th, 60th, 80th, and 100th sample (i.e. every 20 samples). A drastic capacity change is shown on the 40th sample, after the vertical handoff (from 1xRTT to 802.11b) occurs at around 30th sample, and the new capacity is reported on the 40th sample. Note that, the capacity estimation of 802.11b link reported in Figure 8 is lower than the results reported in Table 8. This is because the results of Figure 8 are based on outdoor experiments, whereas the results of Table 8 are collected from indoor experiments. While operating 802.11b in the outdoor environments, the effective capacity is

degraded by an increased number of retransmissions due to wireless channel impairments (e.g. fading and interference).

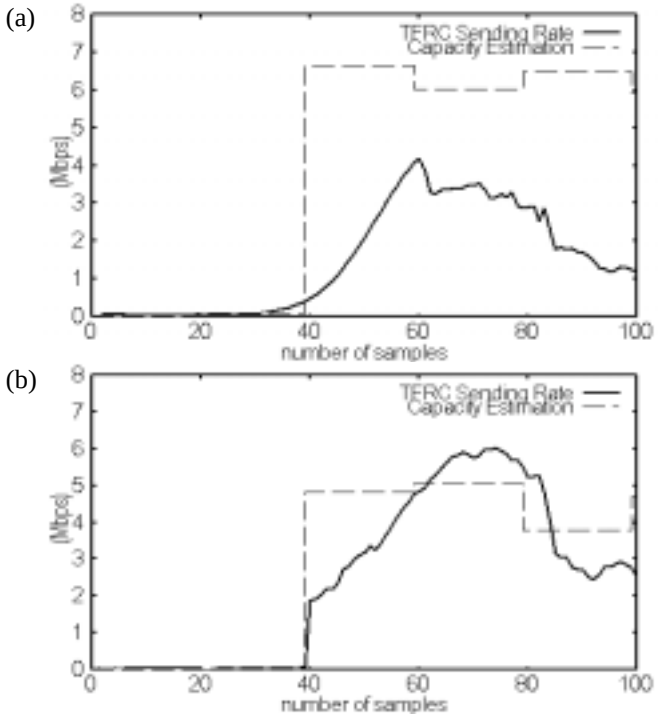

Figure 8: Sending rate of TFRC Probe in vertical handoff (from 1xRTT to 802.11b) (a) TFRC Probe without "fast rate adaptation" (b) TFRC Probe with "fast rate adaptation"

The performance improvement of "fast rate adaptation" algorithm is also shown in Figure 8. In Figure 8-a, the TFRC Probe sending rate increased very slowly after the vertical handoff, which is consistent with the *SlowCC* (slowly-responsive congestion control) feature of TFRC [1], where the maximum increase of the sending rate is showed to be 0.14 packets/RTT (or 0.22 packets/RTT with history discounting) [7].

On the other hand, while the "fast rate adaptation" algorithm is enabled, TFRC Probe will enter *slow start* phase immediately when a vertical handoff from LOW to HIGH capacity link is detected. Following the TFRC algorithm, TFRC Probe will keep itself in the slow start phase and increase its sending rate aggressively to probe the new available bandwidth. It will then switch to *normal* phase after a loss event. Figure 8-b shows that the sending rate increases from 50Kbps to 1.9Mbps very fast after the 40th sample, thus better utilizing the network resources than the original TFRC (as shown in Figure 8-a).

It should be mentioned that the proposed "fast rate adaptation" may potentially raise unfriendliness and unfairness problems with other coexisting flows, since TFRC Probe will aggressively probe the available bandwidth (entering slow start phase) when a drastic increase of link capacity is detected. To alleviate the potential negative effects, it is necessary to extend TFRC Probe by including for example random loss vs congestion discrimination algorithm as reported in [3]. Incidentally, the "fast rate adaptation" concept can also be applied to TCP Probe to better utilize the network resources in the case of frequent changes in the wireless medium. We defer a detailed study of this technique to future work.

VII. CONCLUSION

In this paper, we studied passive capacity estimation and proposed two CapProbe based passive techniques, TFRC Probe and TCP Probe, to estimate link capacity. Differing from the traditional active approaches, passive techniques can minimize the traffic overhead while estimating the link capacity. Using Internet experiments, we showed the high accuracy and convergence speed of the proposed passive techniques.

Two applications of TFRC Probe were evaluated in monitoring the wireless link capacity and detecting the occurrences of vertical handoffs. We also proposed the "fast rate adaptation" algorithm to enhance the application performance upon the notification of the vertical handoff from low capacity to high capacity link. Using testbed experiments, we showed TFRC Probe is able to adjust its sending rate much faster when this algorithm is enabled.

The future work of this study is to evaluate the passive CapProbe techniques within other data transmission protocols. Besides two way measurement, we plan to extend the proposed techniques to be capable of estimating the link capacity on both forward and reverse paths, i.e. one way measurement in both directions. Also, another future direction is to study and integrate the capacity estimation with other existing techniques (e.g. load estimation, error rate estimation, and loss discrimination) to benefit TCP and other data transmission protocols in the performance and efficiency improvement.

VIII. REFERENCES

[1] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. "*Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms*," In Proc. of ACM SIGCOMM 2001.

[2] R. Braden, "*Requirements for Internet Hosts – Communication Layers,"* IETF RFC 1122, Oct. 1989.

[3] S. Cen, P. C. Cosman, and G. M. Voelker, "*End-to-end differentiation of congestion and wireless losses,"* Networking, IEEE/ACM Transactions on, vol. 11, issue 5, 2003, pp 703-717.

[4] L.J. Chen, T. Sun, B. Cheung, D. Nguyen, and M. Gerla. "*Universal Seamless Handoff Architecture in Wireless Overlay Networks*," Technical Report TR040012, UCLA CSD, 2004.

[5] C. Dovrolis, P. Ramanathan, and D. Moore, "*What do packet dispersion techniques measure?"* in Proceedings of IEEE Infocom'01, 2001.

[6] A. B. Downey, "*Using Pathchar to Estimate Internet Link Characteristics,"* In Proc. of ACM SIGCOMM 1999.

[7] S. Floyd, M. Handley, J. Padhye, and J. Widmer. "*Equation-Based Congestion Control for Unicast Applications*," In Proc. of ACM SIGCOMM 2000.

[8] A. Gurtov and J. Korhonen, *"Measurement and Analysis of TCP-Friendly Rate Control for Vertical Handovers,"* submitted for publication, http://www.cs.helsinki.fi/u/gurtov/papers/vho.html
[9] Implementation of the TCP-Friendly Congestion Control Protocol (TFRC), www.icir.org/tfrc/code/
[10] V. Jacobson,*"Pathchar:A tool to infer characteristics of Internet paths"*, ftp://ftp.ee.lbl.gov/pathchar
[11] R. Jain, *"The art of computer systems performance analysis,"* John Wiley and sons, QA76.9.E94J32, 1991.
[12] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, M. Y. Sanadidi, "*CapProbe: A Simple and Accurate Capacity Estimation Technique*," to appear in ACM SIGCOMM 2004.
[13] R. Kapoor, L.-J. Chen, M. Y. Sanadidi, M. Gerla, *"Accuracy of Link Capacity Estimates using Passive and Active Approaches with CapProbe,"* in Proceedings of ISCC 2004.
[14] E. Kohler, M. Handley, and S. Floyd, *"Data Congestion Control Protocol,"* IETF Internet-Draft, draft-ietf-dccp-spec-06.txt.
[15] K. Lai and M. Baker, *"Measuring Bandwidth"*, In Proceedings of IEEE INFOCOM '99, p. 235-245.
[16] Bruce B. Lowekamp, *"Combining Active and Passive Network Measurements to Build Scalable Monitoring Systems on the Grid,"* Performance Evaluation Review 30(4):19-26, March 2003.
[17] NIST Net, http://snad.ncsl.nist.gov/itg/nistnet/
[18] Network Simulator (NS-2), *http://www.mash.cs.berkeley.edu/ns/*
[19] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. *"Modeling TCP Throughput: A Simple Model and its Empirical Validation*," In Proc. of SIGCOMM, 1998.
[20] V. Paxson, *"Measurements and Dynamics of End-to-End Internet Dynamics"*, Ph.D. thesis, Computer Science Division, Univ. Calif. Berkeley, April 1997.
[21] Mark Stemm, and Randy H. Katz. *"Vertical Handoffs in Wireless Overlay Networks*," ACM Mobile Networking (MONET), 1998.
[22] M.S. Taqqu, W. Willinger, R. Sherman, *"Proof of a fundamental result in self-similar traffic modeling,"* SIGCOMM Computer Communications Review, 27: 5-23, 1997.
[23] R. Wang, G. Pau, K. Yamada, M. Y. Sanadidi, and M. Gerla, *"TCP Startup Performance in Large Bandwidth Delay Networks,"* in Proc. of IEEE Infocom 2004.
[24] R. Wang, M. Valla, M. Y. Sanadidi, and M. Gerla, "*Adaptive Bandwidth Share Estimation in TCP Westwood*," In Proc. of IEEE Globecom 2002.
[25] WLAN: Expected Throughput, http://www.uninett.no/wlan/throughput.html