Speedo: Realistic Achievable Bandwidth in 802.11 through Passive Monitoring

Malik Ahmad Yar Khan and Darryl Veitch
ARC Special Centre for Ultra-Broadband Information Networks (CUBIN), an affiliated program of
National ICT Australia (NICTA). Dept. of Electrical and Electronic Engineering University of Melbourne, Australia.
Email: {a.malik,d.veitch}@ee.unimelb.edu.au

***Abstract*—End to end available bandwidth is often constrained by IEEE 802.11 hops in the access network. Available bandwidth measurement is particularly challenging in the wireless environment because of adaptive data rates, a time varying channel, and CSMA/CA based contention instead of simple FIFO queueing. We present a novel estimation scheme for IEEE 802.11 networks in infrastructure mode, based on passive monitoring, without any need for access point cooperation or protocol modifications. We employ an empirical map of relationships local to the client station, and direct measurements of channel utilisation, combined with the known packet size dependent overheads. In testbed experiments we show how the method provides a reliable lower bound of the achievable IP bandwidth, which is also a good point estimate in the case where sources are not greedy. We compare against *probegap*, an alternative method which also measures channel utilisation, but does not take rate or PER into account.**

***Index Terms*—passive monitoring, IEEE802.11, available bandwidth, RSSI**

I. Introduction

The literature on end-to-end techniques for network measurement, including both active probing and passive monitoring, is now quite extensive. However, this literature is overwhelming wireline based, because the dominant model describing link packet delays in wired networks breaks in the more complex wireless case. This wired model is both simple, and a good abstraction for many store and forward network devices in use today [1]. It is characterised by a well defined constant link capacity, FIFO scheduling of packets, and very low and near-constant Packet Error Rates (PER). In contrast, in 802.11 networks the capacity varies widely depending on conditions, scheduling is contention based, and interference and contention effects result in highly variable PER.

Not surprisingly, the very different nature of the 802.11 MAC results in measurement techniques based on the wireline model simply failing [2], [3]. Furthermore, results show more complex behaviour as a function of parameters, such as packet size, than one expects from wireline experience. A new model is needed specific to 802.11 networks to form the foundation for the design of end-to-end measurement methods. Finding a simple, parsimonious one is very challenging given the number of factors influencing wireless performance. There has been little work in this space [3], however the problem is important since even a single wireless link in a path invalidates the use of wireline measurement methods applied to the entire path. Furthermore, since such access links are often path bottlenecks, even 1-hop results, measured from the wireless station, are of immediate interest to applications and the user.

This paper addresses the problem of how a wireless station can determine the wireless bandwidth available to it. We consider the common case of a 802.11 network in 'infrastructure mode', and assume that the RTS/CTS reservations scheme is not used. Our method does not require association to, or cooperation from, the AP, and is based on passive monitoring performed at the client station only. It works for all current 802.11 variants, uses only commonly available hardware, and is not based on any special feature which is likely to become unavailable in the future.

Our approach takes into account each of the key factors which determine available bandwidth in a systematic way:

(i) Effective use of physical layer information: by itself, signal strength is a very poor indicator of performance, but we show how it can be effectively used when combined with other parameters.

(ii) Incorporating the right link layer rate: this has a huge impact, consider that a given window in time can fit many more packets at a high rate (say 54 Mbps under 802.11a) than at a low rate (say 6 Mbps).

(iii) Incorporating inherent packet size dependency: one simple and large source of packet dependency of bandwidth is due to the known overhead structure of 802.11 MAC. We account for it explicitly.

(iv) Separating physical and contention components of PER: driver rate selection algorithms currently reduce rate under high PER. We explain how this is the wrong thing to do and emphasize the importance of isolating rate decisions from contention effects.

The paper is structured as follows. Section 2 introduces the 802.11 standard, our testbed, and the enormous importance of packet size and data rate, and section 3 briefly describes relevant prior work. Section 4 carefully outlines the principles and arguments underlying Speedo, and uses experimental data to argue for an empirical approach. Section 5 evaluates Speedo in a series of test scenarios in the laboratory, and also uses experiments to show how Speedo compares to probegap. We conclude in Section VI.

II. Background

A. The 802.11 Wireless LAN Standard

The original IEEE 802.11 standard detailed the Physical (PHY) and Medium Access Control (MAC) of Wireless Local Area Networks (WLANs). Subsequent revisions (namely 802.11b a, g, and n) improve the physical layer but retain the same MAC. Variants such as 802.11s for mesh networking, or 802.11e which supports different service classes, do alter the MAC, but only in ways which can be easily accounted for by our approach. We now describe details of the 802.11{a,b,g,n} MAC for the widely used *infrastructure mode*, where a central Access Point (AP) bridges the data between all participating (or *associated*) stations (STAs).

The *Distributed Co-ordinated Function* (DCF), the default MAC mechanism to transmit and receive frames, uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), which operates as follows. When a frame is ready for transmission, the station waits until the channel is sensed continuously idle for a Distributed Inter-Frame Space (DIFS) time interval. If this occurs straight away, *and* the last transmitted frame was not from this station, transmission proceeds. Otherwise, to minimize collisions and prevent channel capture, a backoff counter is used to defer transmission. The counter is initialised uniformly at random to a value c in the *contention window* $[0, C_w]$ measured in slots. It is decremented for each idle slot beyond the DIFS interval following the first (and any subsequent) periods sensed busy, and is frozen otherwise. The frame is transmitted when c reaches zero. The destination station, after receiving the frame correctly, transmits an acknowledgment (ACK) frame back to the source after a Short Inter-Frame Space (SIFS) interval. Since DIFS $>$ SIFS, ACKs will not collide with data packets provided the former can be heard by all stations.

Frames may be corrupted due to physical layer issues, collisions of contending data frames, or collisions of ACK and data frames involving hidden terminals. If the transmission is unsuccessful. the contention window size C_w, initially C_w^-, is doubled (up to a max. of C_w^+) and retransmission attempted.

B. Testbed

Our testbed consists of wireless clients, access points, PCs and traffic generators. We chose Linux as the operating system in order to benefit from open source driver code, and focussed on 802.11a,e to avoid interference from neighboring 802.11{b,g} networks. Experiments were conducted both in a busy office environment, and a large empty room, both during the day and at night.

The wireless hardware consists of Linksys [4] and Netgear cards, based on Atheros chipsets [5], which can operate in each of 802.11{a,e,b,g}. We used the Multiband Atheros Driver for WIreless FIdelity (MADWIFI) [6], a Linux kernel driver for Atheros-based Wireless LAN devices. We used the β release madwifi-old rather than the more current (madwifi-ng, then an α release), as this enabled data rate to be held constant for controlled experiments. The PCs are each Pentium III or higher, running Fedora Core 5, kernel version 2.6.18.

We make extensive use of accurate passive monitoring. Most wireless card manufacturers (including Linksys and Netgear) provide a *monitor mode*, similar to promiscuous mode in wired Ethernet, whereby all valid packets within range of the card on a given channel are captured, even those not destined for it. Exceptions occur when packets are generated at the physical layer, and so never pass by the driver based monitoring. This includes beacon frames if the monitor is in an AP, and ACKs on the data receiver side. The MADWIFI driver supports client or AP mode being used simultaneously with monitor mode. Because there are known problems with monitor mode [7], [8] (monitors can fail to see all packets due to resource constraints), we subsequently installed a sniffer node (which sends no traffic) to check monitor performance. The sniffer uses a Linksys card whereas the client station uses Netgear. We found the monitor to report losses reliably, even under high load. Monitored IP packets have MAC headers and physical layer PRISM headers appended, and are subsequently caught via the *libpcap* [9] capture library and stored. The packet traces are later filtered by *tcpdump* and processed by programs we wrote in *C++* and *Matlab* to access the information we require at different protocol levels:

IP: packet size, PC system timestamp;
MAC: retry bit;
PHY: RSSI, data rate, resent packet counter.

The per-packet Received Signal Strength Indicator (RSSI) is available in all IEEE 802.11 compliant cards. In our case it is measured over each preamble and physical layer header and is hence independent of packet size. The mapping between RSSI to SNR is not standardized by IEEE, but is supplied by vendors. For our cards, RSSI = SNR, and S = SNR + W where S is signal strength and W is room temperature thermal noise taken to be -95 dBm.

We use the 'resent packet counter' (the number of times the packet has been sent) to calculate Packet Error Rate (PER) precisely in client stations. To enable this using the madwifi-old driver some modifications were needed. The MAC retry bit is used for PER estimation of packets captured by the monitor but originating from other stations.

To simulate competing background IP traffic we use open source generators. We primarily use the Distributed Internet Traffic Generator (D-ITG) [10] to generate TCP and UDP traffic of various types, in particular (pseudo) Poisson UDP traffic. Some experiments were also performed using Iperf [11], which generates periodic UDP traffic only. We used RUDE [12] in our implementation of *probegap*, to provide a finer level of control of the probing stream.

C. IP Bandwidth over IEEE 802.11

IP transport over IEEE 802.11 incurs both deterministic and random overheads. When using DCF access, the total effective duration in seconds of an IP packet of size l bits transmitted at rate r, including the MAC level ACK transmitted at rate r_a, and an effective backoff time of c slots, is

$$T(l, r, c) = \left(A + \frac{h_a}{r_a} \right) + \frac{h + l}{r} + c\tau, \qquad (1)$$

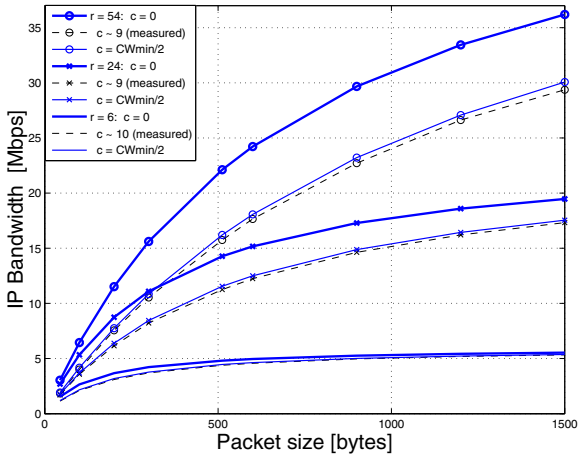

Fig. 1. IP bandwidth in IEEE 802.11a,e for $r = \{6, 24, 54\}$ Mbps (Netgear). For each r: IPB$(l, r, 0) >$ IPB$(l, r, C_w^-/2) >$ IPB$(l, r,$ measured). Measured curves are very close to the predicted average of $\bar{c} = C_w^-/2$.

where $A = \text{DIFS} + \text{SIFS} + (\text{Pre} + \text{PHY})2$ sums the bits in the guard bands, preamble, and physical header, h and h_a are the header sizes of data packets and ACKs respectively, and τ is the slot width. The preamble and physical layer header are always transmitted at the lowest rate. The rate r for data depends on network conditions, whereas the ACK rate $r_a = \min(r, \max\{r_s\})$ depends on the AP *specified rates* $\{r_s\}$ in the network settings. Our monitoring provides the actual value used.

Since l IP bits are transmitted in $T(l, r, c)$ seconds, the IP bandwidth is just

$$\text{IP Bandwidth} = \text{IPB}(l, r, c) = \frac{l}{T(l, r, c)} \quad \text{bps.} \qquad (2)$$

Figure 1 plots IPB(l, r, c) in the case of 802.11e as a function of l using rates $r = \{6, 24, 54\}$ Mbps (and $r_s = \{6, 12, 24\}$). Curves are shown for $c = 0$: an upper bound corresponding to zero contention, and $c = C_w^-/2$: an average value corresponding to a single greedy user selecting c uniformly in $[0, C_w^-]$, with no errors or retransmissions. The key observation here is the heavy dependence of IP bandwidth on both packet size and data rate.

For each r we also run experiments using a single greedy UDP Poisson packet stream (from a station to an AP) for which we expect the achievable bandwidth to be close to, but, due to physical layer errors, a little below IPB$(l, r, C_w^-/2)$. The measurements bear this out well. When performing the same experiments using 802.11a however, the measurement curves coincided almost exactly with the $c = 0$ bounds. A large number of other experiments confirmed the that the backoff behaviour of our driver under 802.11a was far from the standard. This motivated our use of 802.11e for the testing of Speedo, and provided an opportunity to illustrate how different 802.11 variants are easily accomodated.

We use the 802.11e extension to 802.11a in best effort mode only, where it behaves essentially like 802.11a, apart from some header size changes and 'DIFS' becoming the slightly larger 'AIFS' (we call this 'DIFS' here).

D. The Idea of Available Bandwidth

If the capacity of a link C is constant, and the traffic rate given by a stationary process $X(t)$, then the available bandwidth can be defined as the expected free capacity $B = C - \mathbf{E}[X(t)]$. As has often been pointed out however, this measure of unused capacity may not correspond to what applications actually see. For example sources running TCP will react to a new flow, establishing a new equilibrium where the bandwidth share for the new flow may be larger than B.

In 802.11 networks, even if traffic sources are not reactive (such as simple UDP streams), the contention based 802.11 MAC [13] acts as a fairness mechanism so that, again, what is measured as free may not be what is experienced, and furthermore will depend on parameters such as the number of competing streams, which may be unknown. Accordingly, Speedo aims to provide a lower bound, an *achievable bandwidth*, to the *available bandwidth*. We define the latter operationally as the bandwidth that was obtained during our validation experiments involving a greedy source as defined below. In a nutshell, we find that if sources do not back off, our achievable bandwidth comes close to the measured available bandwidth, and is typically (but not necessarily) lower than it. If they do, it is a lower bound whose tightness is a function of several factors, notably packet size and packet error rate.

III. Previous Work

Compared to the extensive wireline literature, there are few papers in the area of bandwidth estimation in 802.11 networks. The paper most closely related to ours is [3], which aims specifically to estimate available bandwidth, and highlights as we do the importance of measuring utilisation, or equivalently idle fraction, as a metric of relevance beyond FIFO systems. However there are many important differences, including measuring IPB through active probing rather than calculation, and an active probing based method to measure utilisation via the delay distribution of probes rather than through careful passive monitoring. It also makes no use of physical layer information such as signal strength. As an active method, it requires cooperation from both the client and server side, whereas our approach is passive. Nonetheless, we compare against the associated 'probegap' tool from [3] in section V-C. The workshop paper [14] also points out the importance of utilisation (the 'channel time proportion') and points out as we do that contention should allow a nominal available bandwidth to be achieved since other sources will suffer due to backoff.

In [15], the 'DietTOPP' method is presented, and used to measure the available bandwidth in a hybrid wired and wireless setting. Wireless links in isolation are not considered, and the capacity of the wireless link is assumed to be fixed. Their technique (based on the wireline TOPP method) reports a single bandwidth prediction assumed to be relevant for all packet sizes. The bandwidth results depend on probe size which contradicts the TOPP theory on which the method is based.

IV. SPEEDO

The aim of Speedo is to combine the factors with the greatest impact on IP level available bandwidth together in a way which is simple, robust, and practically achievable. These factors are packet size, data rate, PER, signal strength, and channel utilisation, *relative to the link between the AP and our client station*.

Other parameters which would give insight into available bandwidth, notably the number of active users of the AP which impacts strongly on contention, are also important, however for simplicity, robustness, and computational reasons we do not attempt to measure or infer these directly.

The underlying idea is very simple. First, passive measurement is used to measure the *channel utilisation* ρ, the proportion of time when the channel is free. Second, RSSI and the physical component of PER are used to determine the highest achievable data rate r_l^* relative to an IP packet of size l. The bandwidth estimate for a flow of packets of this size is then simply given by

$$B(l) = (1 - \rho) \cdot \text{IPB}(l, r_l^*, c) \tag{3}$$

for some suitably chosen c. The definition composes two distinct tasks, the measurement of ρ, and the determination of r_l. We describe these separately below, and describe the role of PER in section IV-C, before returning to discuss and refine equation (3) in section IV-D.

A. Channel Utilisation ρ

The 802.11 MAC protocol is far from a FIFO queue. As a result, FIFO specific measures of system load, such as one-way delay, are not direct measures of system state and so are difficult to exploit. The system utilisation however is a meaningful metric for a much wider class of systems , and so is a clear candidate as a means to tackle available bandwidth.

In Speedo the station measures ρ by individually accounting, over each time block of duration T_ρ (currently set to $T_\rho = 1$ [sec]), the channel busy time due to each frame seen by the station monitor (including SIFS, DIFS bands etc.).

It is important to note that this is a precision monitoring approach, involving a minimum of (potentially problematic) protocol based inference, such as assuming an ACK is paired to a single data packet, or that certain rates are used - to the extent possible everything is explicitly measured. In this way we are including diverse effects which may be otherwise difficult to measure or infer. For example repeated unnecessary retransmissions of data or ACKs are taken into account simply by noting which frames actually appear. In addition, although hidden terminals will result in some exchanges being only partially heard, leading to ρ being measured low, they will still be at least partially included indirectly via the accounting of packet missing due to contention-generated loss. We compensate for this by inflating ρ based on the assumption that for each frame seen with a retry bit set there was another unrecorded one of the same size, sent at the same rate, which was not recorded.

In in fact there are no hidden terminals, that is if the channel acts like a wired hub, where 'everyone hears everything', then, collisions and driver anomalies aside, our monitoring approach will miss very little that is happening. The reported utilisation ρ will be close to exact.

Note that the MAC mechanisms which infer when the channel is busy, currently used to support collision avoidance (also known as Virtual Collision Avoidance), are ideally suited to the measurement of ρ, but are not used for this purpose. Doing so would result in both greater accuracy and much reduced overhead for the station monitor.

B. Data Rate Determination

Utilisation gives a measure of how much 'free space' there is in the wireless channel, but does not tell us how it will be used. Clearly, the channel data rate will have a dramatic impact on bandwidth (regardless of how it is defined), as will the choice of packet size, as Figure 1 illustrates in a particular case. Whereas the user might control or know the relevant packet size, the question of which data rate to apply is less straightforward.

A tempting pragmatic answer is simply to duplicate the decisions of the drivers' rate selection algorithm (assuming this can be done). However, a given algorithm may make a poor decision which we may not want to replicate. For example, a key problem with current algorithms is that they will react to high PER by lowering the rate. However, if the measured high PER is dominated by contention, whereas the physical component is low, then this is precisely the wrong thing to do. Lowering the rate will only increase ρ and decrease IP bandwidth, and hence make contention worse. In contrast, Speedo aims to determine immediately what an achievable rate at acceptable PER would be based on current conditions.

The determination of an achievable rate is closely connected to PER, and in that context, signal strength is of fundamental importance. However, researchers agree (see for example [16]) with wireless users that RSSI can be a poor indicator of achievable throughput. One reason is that PER combines physical layer and contention related components, and RSSI is directly related only to the former. We began our work with experiments designed to characterise the relationships between rate and the above parameters in practice.

We collected RSSI and measurements of PER at an AP running 802.11a for the packets of a simple greedy UDP source (using iPerf) which saturated the channel between the AP and the station. Six typical locations about an office area were used in order to obtain a spread of RSSI and hence SNR. At each of these, for each of the 802.11a rates $r = \{6, 9, 12, 18, 24, 36, 48, 54\}$ Mbps, and packet sizes $l = \{48, 540, 1500\}$ bytes, data was collected over 5.5 minutes. The results showed no clear pattern of any kind. In particular grouping results together based on similar RSSI gave no consistent behaviour with respect to any other variable. Hypothesizing that complex interference effects makes results at different locations incompatible, we next stratified the data according to location and reexamined it. The parameter de-

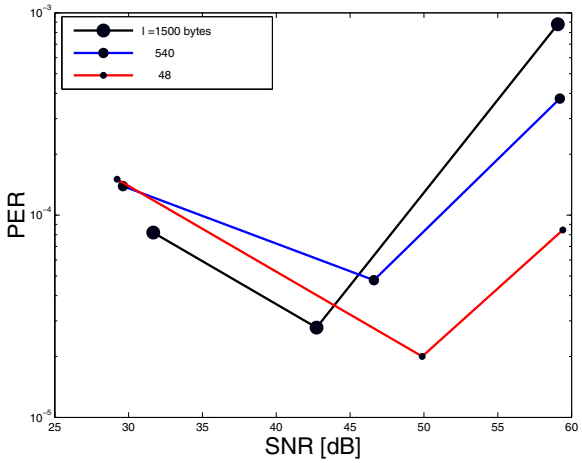

Fig. 2. Dependence of PER on SNR, with $r = 6$ Mbps.

pendencies became considerably simpler, but still no coherent conclusions could be drawn. Note that the same station was shifted between locations, so that software and hardware were constant.

As there is only a single source here, measured PER consists of the physical component only. This is in fact our aim. The greediness of the source simply enables us to gather per-packet physical PER measurements more quickly. Although it will generate backing off, this will nonetheless not lead to contention, and so backing off plays no role here. In what follows by PER we mean the *physical component of PER only* unless otherwise stated, and not the contention or total PER. This distinction lies at the core of our approach.

We next moved to a simpler test suite involving three locations, with the first two being line of sight to the AP, and the third not. The locations were chosen to offer a spread of SNR, and the average values were $(30, 45, 60)$ dB. An example of the results appears in Figure 2, where PER actually *increases* when SNR is increased (note also the considerable dependence on packet size). Note that SNR and PER values showed no sign of non-stationarity in any of these tests. Further examples of our findings can be found at [17].

With even the simplest scenario in a near-static environment producing non-monotonic behaviour in key parameters, we concluded that it was not practical to expect simplistic models, or results of theoretical studies based on restrictive assumptions, to be of use in rate prediction. Accordingly, we move to an empirical approach, letting the data tell us what the station specific mapping between PER, RSSI, packet size and rate is. To achieve this, we view the data as effectively a scatter plot in four dimensions. It can be used to provide r as an implicit function of l, physical PER and RSSI, which we call the *map*.

A representation of the map, using the same test suite data above, is given in Figure 3. Here SNR has been discretised into three sets, corresponding to each of the three locations, and visualised via the fill tone of the disks, whereas packet size is indicated by the disk radius.

The lack of monotonicity is clearly seen, for example in the many cases when disks at fixed r are not in order of size, nor of tone. Of more relevance here however is the fact that, for a given fixed PER cutoff threshold (a vertical line in the map), the (r, l, RSSI) combinations which lie to the left or right of it follow no simple relationship.

The map represents a memory of the physical parameter relationships found in the environments experienced by the AP-station pair. The role of RSSI is effectively to select which of these environments is active at a given time, but cannot on its own be used to predict r. For example the three locations here could correspond to moving a laptop between an office, a colleague's office and a meeting room, in which case a measurement of the current RSSI would effectively select the applicable sub-map (such as the black one in Figure 3). With greater mobility, RSSI would play a more dynamic role, operating on smaller timescales, selecting values of most relevance to conditions encountered previously.

Rate determination algorithm (for a given map)

- Measure average RSSI s over time interval T_s.
- Select a packet size l of interest.
- Determine an acceptable PER threshold value $e(l, s)$.
- Set r^* to be the highest rate of points in the map with l and s such that PER does not exceed $e(l, s)$.
  If no such rate exists, set r^* to the lowest data rate.

How to choose $e(l, s)$ is addressed in the next subsection. Note that, although a map is in some sense just recording the prior decisions of a given driver/card combination, this algorithm is not constrained to duplicate those decisions as it free to choose which points in the map to recommend.

In this paper, we generate maps 'off-line' using a single greedy source and no contention as described above. We are currently working on methods to measure maps on-line.

Although the example map of Figure 3 was generated using 'downloads', that is data packets flowed from the AP to the station, normally the initialisation and use of the map will include both downloads and uploads. This is as it should be, since Speedo aims to measure the wireless channel only,

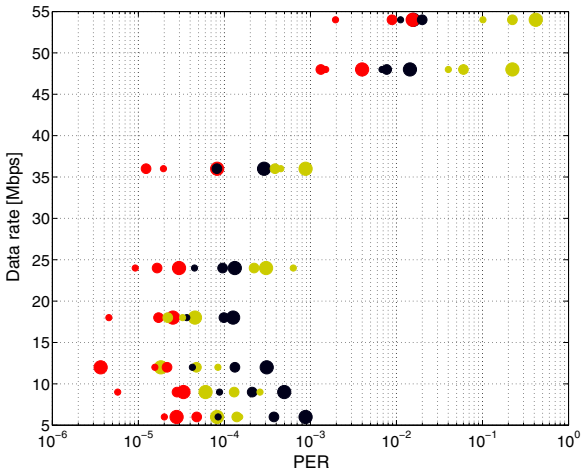

Fig. 3. The empirical map used to select rate according to **physical** PER. Symbol size is proportion to packet size $l = \{48, 540, 1500\}$ bytes and fill tone proportional to discretised RSSI≈SNR= $s = (30, 45, 60)$ dB, (green,red,black) (graduated greyscale in printout). Only (l, r, s) tuples on the left of a given chosen PER are achievable.

which as a shared medium is in most respects agnostic as to data direction. In particular, channel utilisation is inherently symmetric, as is the IPB formula. In the maps generated below to validate Speedo, we use uploads.

C. PER selection

A target BER which is commonly used in 802.11 is 10^{-5}, corresponding to PER$= 0.12$ for a 1500 byte packet. We adopt $e^* = 0.05$ and use it to select feasible values of r within the map. Note that this works because the map is based on physical PER only, and the key point regarding PER selection is that it *must not be influenced by MAC level contention*.

D. The Speedo Algorithm

We now complete the definition of Speedo.

We first determine of value of c to be used in equation 3. In the case of a single greedy user, a value of $C_w^-/2$ is appropriate as it corresponds to the average backoff of any station.

Now consider the issue, deferred from section IV-A, of selecting a value of c to be included in the ρ measurement. The problem here is we do not know the number of sources. We therefore conservatively choose $c' = C_w^-/2$ for ρ calculation. Note that this will tend to inflate ρ, resulting in an underestimate of available bandwidth.

Combining the approaches adopted in this section, a more complete equation describing Speedo can now be given as

$$B(l, s^*, e^*; c, c') = (1 - \rho(c')) \cdot \text{IPB}(l, r^*(l, s^*, e^*), c) \quad (4)$$

s^* selects the sub-map of relevance based on current RSSI,
e^* selects an acceptable physical level performance,
r^* is given by the map, using the current environment, acceptable physical performance, and packet size of interest,
c is set to $C_w^-/2$ to reflect the average amount of backoff the client station frames will suffer,
c' is set to $C_w^-/2$ to overestimate the average amount of backoff other frames will suffer,
$1-\rho(c')$ is the mainly measured, but partially inferred, fraction of time when the channel is available, and
IPB adjusts for the r and l dependent overhead.

When contention effects are small, Speedo should accurately implement the idea of filling available space with packets of an achievable size, though with some underestimation due to c'. When contention effects are not small, or competing traffic is elastic, competing traffic can back off and Speedo's prediction will be an underestimate. Hence we expect Speedo to give a value which is in general *achievable*, but the actual *available* bandwidth obtainable can be larger.

V. Performance

In this section we test the performance of Speedo in a number of different stationary scenarios using 802.11e. Two physically different 'RSSI environments' will be used, for which sub-maps are established off-line via a prior map measurement phase as described above. The combined map is shown in Figure 6.

We begin by describing our validation methodology in the context of particular examples from the systematic results of section V-B. The results shown below (except for probegap comparisons) are for a single repetition (one realisation) only, however we are confident in their stability and reproducibility based on numerous repetitions not reported here.

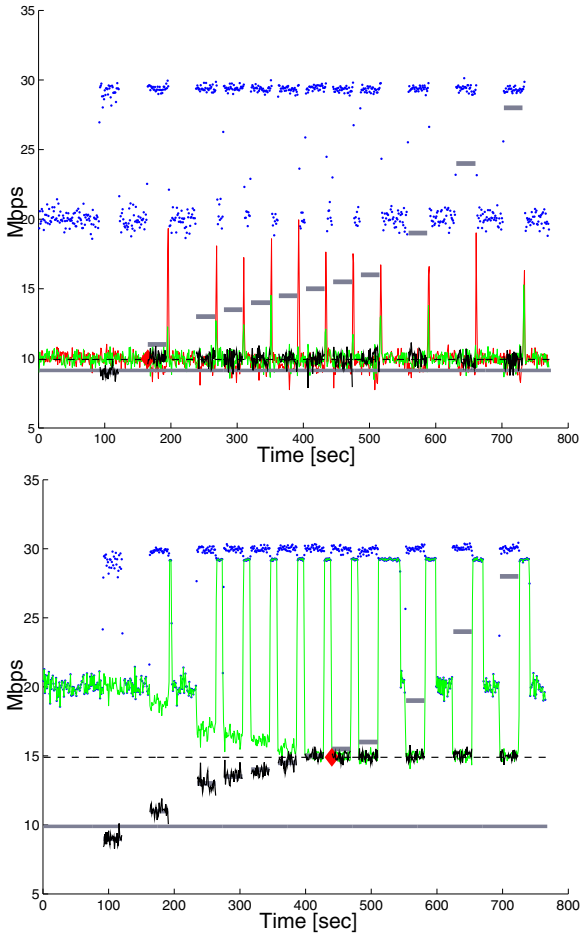

Fig. 4. Validation of Speedo in simple scenarios showing station and total throughputs. Top: The VT offered load (grey steps) rise steadily but VT throughput saturates (black dashed line) very close to the Speedo prediction (lowermost grey line). Bottom: The CT stations back off once VT triggers fair sharing, resulting in Speedo giving a lower bound.

A. Validation Methodology with Examples

The top plot in Figure 4 shows the details of a typical experiment involving $N = 2$ *cross traffic* (CT) stations (green and red), each of which uses ITG to send a Poisson cross traffic stream of rate 10Mbps toward the AP. For the first 60 seconds of the experiment a third station, the client station using Speedo, listens and measures ρ, and then makes its achievable bandwidth prediction shown by the thick grey horizontal line (at around 9Mbps).

After estimation the validation phase begins, whereby *validation traffic* (VT) from the client steadily ramps up its offered load (shown by the grey steps), which continues to rise even as the VT throughput (black curve) saturates, which happens almost immediately in the top plot. Between each step the VT is paused so that any CT traffic backlogs can clear, an operation clearly visible by the spikes in CT throughput. In

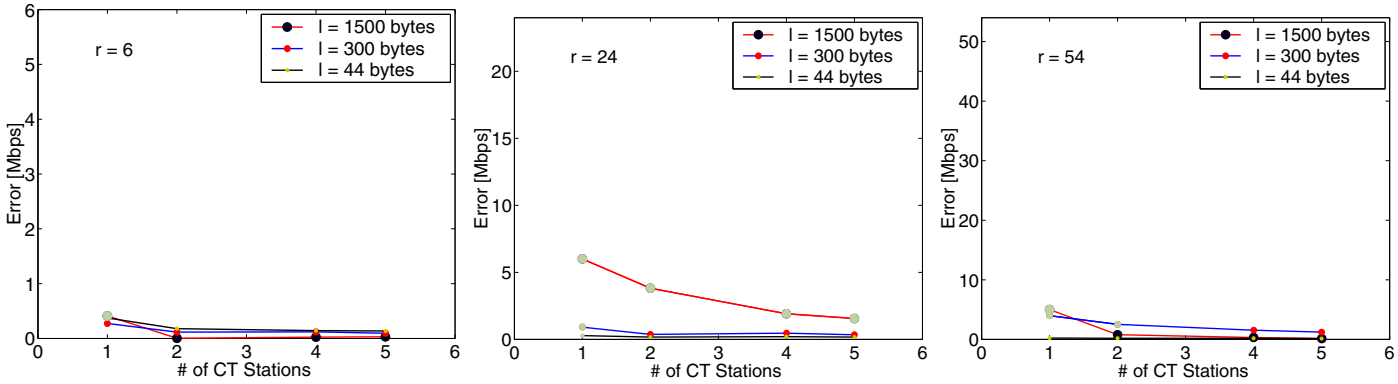

Fig. 5. Homogeneous validation scenarios covering $r = \{6, 24, 54\}$, $l = \{44, 300, 1500\}$, and $N = \{1, 2, 4, 5\}$ sources at fixed aggregate CT bandwidth (see table I) for $r = \{6, 24, 54\}$ respectively, using high RSSI (black sub-map). Larger errors are due to MAC fairness operating (shaded grey).

this homogeneous experiment, both CT and the VT streams share the same $(l, r) = (1500\,\text{bytes}, 54\,\text{Mbps})$. In the case of VT this combination is authorised by the high quality (black) environment in the map (Figure 6) using $e^* = 0.05$.

The VT is again a Poisson UDP stream generated by ITG, and the blue points across the top of the plot show the total throughput over all three stations. The VT saturation level (black horizontal dashed line) is based on averaging the VT from a point (marked by the large red symbol) clearly past the saturation onset. The error in Speedo's prediction is given by the difference between this saturation level and the grey prediction line below it.

This first example shows, as expected. Speedo's achievable bandwidth estimate being below, but close to, the available bandwidth determined by the VT saturation. The bottom plot in Figure 4 shows a scenario which is similar in all respects to the first, except that only one CT station is present, providing the same aggregate CT of 20Mbps. In this case, contention forces the CT to back off, effectively becoming starved of resources and hence greedy. The fair share property of the MAC window contention algorithm is then effectively 'activated', resulting in the VT stealing bandwidth from the CT station.

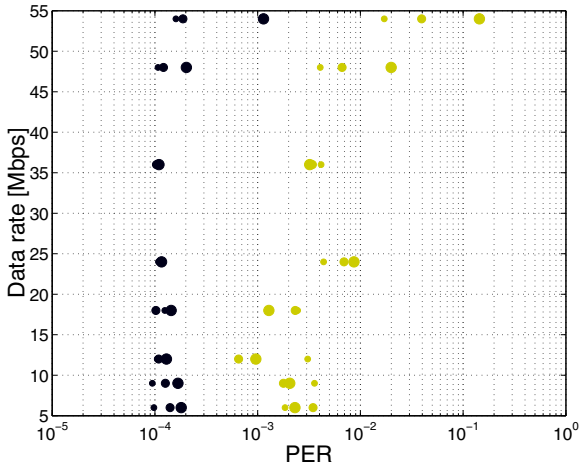

Fig. 6. The map used for Speedo validation experiments. Two physical environments are used with discretised RSSI≈SNR= $s = (41, 24)$ dB, (black,green). $l = \{44, 300, 1500\}$ bytes.

Speedo's achievable bandwidth prediction is indeed achieved, but as expected is well below the available bandwidth in this case. Again we see that queued CT drains as VT pauses, and when VT is on, total traffic is constant even as the share varies.

B. Test Cases

The methodology described above was repeated systematically over a range of parameters, using both the green and black sub-maps. The detailed findings (PER, SNR etc) were in general terms in agreement with the examples given above. The main exceptions were cases when CT stations backed off despite the absence of the window contention fairness effect. This seems to be due to data rates being higher than was warranted, resulting in high physical and very high total PER for those stations. As ever, backing off results in the client station gaining more bandwidth than Speedo would predict.

Figure 5 gives the error in Speedo's estimates (using the high RSSI black sub-map) over a range of homogeneous configurations, where the CT and VT streams all share the same (l, r) (complete details given in table I). In each figure the aggregate CT load is kept constant as the number N of CT stations varies. Each plot corresponds to a value of r, and the vertical scale is chosen to match, allowing absolute and relative errors to be immediately grasped. We see that the errors are typically small. In fact in a number of cases parameters are such that one or more CT streams is starved and bandwidth sharing kicks in, creating a positive error. In particular, this occurs in almost all cases when $N = 1$ and all cases with errors above 3Mbps. We have marked such cases by 'greying out' the corresponding points. We therefore see that in essentially all cases, significant errors are due to backing off effects, and not to other causes. In all cases Speedo's achievable bandwidth was shown to be just that. We tried other configurations with more heterogeneous parameters and found the same result. Finally, we considered cases where the VT is replaced by a greedy TCP stream, to see if it can utilise the bandwidth which Speedo considers achievable. We ran several such scenarios, and found in most cases results very similar to those witnessed above using UDP validation traffic. The achievable bandwidth is used, and if the CT streams do not backoff, TCP can go no further, whereas if fairness is activated,

r	l	Aggregate offered (pkts/sec)
54	1500	1666
54	300	3750
54	44	3125
24	1500	1250
24	300	2083
24	44	3125
6	1500	250
6	300	833
6	44	2500

TABLE I
VALIDATION EXPERIMENTS (CT AGGREGATE OFFERED LOAD).

it can. The exceptions have some strange artifacts which we could not explain, but it is likely they are due simply to bugs. We are continuing to investigate this point.

Station	r	l	offered (pkts/sec)
1	24	1500	316
2	24	1500	200
3	24	300	216
4	6	44	417
5	6	1500	57

TABLE II
CT SCENARIO USED WITH PROBEGAP (LOW SNR SUB-MAP)

C. Speedo versus Probegap

As described in Section III, probegap has a number of disadvantages compared to Speedo in terms of using active probes, and requiring AP cooperation. It also requires that r be supplied to it by some other capacity estimation tool. Here we focus on its estimation accuracy.

Probegap estimates ρ by measuring the knee in an empirical CDF of delays of active probes, related to the probability of finding the channel idle and therefore experiencing a minimum delay. In practice however, the knee may be very indistinct or even absent, and its value is a sensitive function of details of the knee detection algorithm. This results in highly variable ρ estimates. Since probegap, like Speedo, is built on using ρ to measure available space in the channel, this is serious drawback.

We performed comparisons using parameter values where the knee could at least be found. We use our own knee finding algorithm, since the original one was not clearly specified. The CT is described in table II using the low SNR (green) sub-map, and since probegap requires rate to be supplied a priori, here we used fixed r values for both methods.

As table III shows, Speedo outperforms probegap by a considerable margin for the parameters and scenarios we tried.

VI. FUTURE WORK

In this paper we have introduced *Speedo*, a method to provide a lower bound, which we call *achievable bandwidth*, to the available bandwidth that a greedy source could obtain from a WiFi network. The method has a number of inherent advantages, including being based on passive monitoring, not requiring assistance from the access point, is directly applicable or easily adaptable to all 802.11 variants, and based only on driver monitoring capabilities likely to be available into the future. It is based on the careful application of a zero contention model, based on a direct measurement of channel utilisation, to the realistic case where contention may or not be present. In the former case, we show in controlled experiments in an office environment that the measured achievable bandwidth is indeed in general achieved and is close to the measured available bandwidth. In the latter case, the station can steal more bandwidth from other sources but Speedo continues to deliver useful results. Speedo incorporates physical layer effects by the use of a novel way to capture the key dependencies of physical packet error rate (PER), the *map*, which records the station-specific relationship between PER, packet size, RSSI, and the data rate to select.

Speedo Mbps	Probegap Mbps	Saturation Mbps	l bytes	r Mbps
8.9	12.02	9.43	1500	54
5.23	6.49	4.96	1500	24
1.58	2.30	1.95	1500	6

TABLE III
RESULTS OF PROBEGAP AND SPEEDO COMPARISON

There are many directions for future work. In particular, our experiments, although fairly realistic in several ways, were conducted in stationary environments where stations had fixed rates, and cross traffic streams were mainly simple Poisson processes. These assumptions need to be relaxed. A system capable of initialising and refreshing a map in a scalable way is another area of importance and a key step to deployable software.

REFERENCES

[1] N. Hohn, D. Veitch, K. Papagiannaki, and C. Diot, "Bridging router performance and queuing theory," in *Proc. ACM Sigmetrics*, New York, June 12-16 2004, pp. 355–366.

[2] B. Melander, M. Björkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proc. Globecom'00*, Nov 2000, pp. 415–421.

[3] K. Lakshminarayanan, V. Padmanabhan, and J. Padhye, "Bandwidth estimation in broadband access networks," in *Proc. ACM Internet Measurement Conference (IMC)*, 2004.

[4] Linksys, "http://www.linksys.com."

[5] "Atheros," http://www.atheros.com.

[6] "Multiband atheros driver for wireless fidelity," http://www.madwifi.org.

[7] J. Yeo, M. Youssef, T. Henderson, and A. Agrawala, "An accurate technique for measuring the wireless side of wireless networks," in *WiTMeMo'05*. Berkeley, CA, USA: USENIX, 2005, pp. 13–18.

[8] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage, "Jigsaw: solving the puzzle of enterprise 802.11 analysis," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, 2006.

[9] "Packet capture library," http://www.tcpdump.org/.

[10] "Distributed internet traffic generator (D-ITG)," http://www.grid.unina.it/software/ITG/.

[11] Iperf, "http://dast.nlanr.net/projects/iperf/."

[12] "Real-time UDP data emitter (RUDE)," http://rude.sourceforge.net/.

[13] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," in *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, 2000, pp. 535–547.

[14] S. H. Shah, K. Chen, and K. Nahrstedt, "Available bandwidth estimation in ieee 802.11-based wireless networks," in *Proc. 1st ISMA/CAIDA Workshop on Bandwidth Estimation (BEst 2003)*, 2003.

[15] A. Johnsson, B. Melander, and M. Bjrkman, "Bandwidth Measurement in Wireless Networks," in *Proc. Mediterranean Ad Hoc Networking Workshop*, 2005.

[16] A. Balachandran, P. Bahl, and G. Voelker, "Hot-spot congestion relief and service guarantees in public-area wireless networks," *SIGCOMM Computer Communication Review*, vol. 32, no. 1, 2002.

[17] D. Veitch, "Lessons in reproducibility for 802.11 (or, is determinism dead?)," in *Intimate 2007, Workshop on Methods and Tools for Network Analysis*, Carré des Sciences, Paris, France, July 9-10 2007.