Available online at www.sciencedirect.com
SCIENCE DIRECT®
Ad Hoc Networks xxx (2006) xxx–xxx
Ad Hoc Networks
www.elsevier.com/locate/adhoc

Performance benchmarking of wireless Web servers ☆

Guangwei Bai, Kehinde Oladosu, Carey Williamson *

Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4

Received 9 September 2004; received in revised form 17 July 2005; accepted 10 January 2006

Abstract

The advent of mobile computers and wireless networks enables the deployment of wireless Web servers and clients in short-lived ad hoc network environments, such as classroom area networks. The purpose of this paper is to benchmark the performance capabilities of wireless Web servers in such an environment. Network traffic measurements are conducted on an in-building IEEE 802.11b wireless ad hoc network, using a wireless-enabled Apache Web server, several wireless clients, and a wireless network traffic analyzer. The experiments focus on the HTTP transaction rate and end-to-end throughput achievable in such an ad hoc network environment, and the impacts of factors such as Web object size, number of clients, and persistent HTTP connections. The results show that the wireless network bottleneck manifests itself in several ways: inefficient HTTP performance, client-side packet losses, server-side packet losses, network thrashing, and unfairness among Web clients. Persistent HTTP connections offer up to 350% improvement in HTTP transaction rate and user-level throughput, while also improving fairness for mobile clients accessing content from a wireless Web server.

Keywords: Ad hoc networks; Network traffic measurement; IEEE 802.11b WLAN; Web performance

1. Introduction

Two of the most exciting and fastest-growing Internet technologies from the past 10 years are the World Wide Web and wireless networks. The Web has made the Internet available to the masses, through its TCP/IP protocol stack and the principle of layering. Wireless technologies have revolutionized the way people think about networks, by offering users freedom from the constraints of physical wires. Mobile users are interested in exploiting the full functionality of the technology at their fingertips, as wireless networks bring closer the "anything, anytime, anywhere" promise of mobile networking.

A natural step in the wireless Internet evolution is the convergence of these technologies to form the "wireless Web": the wireless classroom, the wireless campus, the wireless office, and the wireless home. In fact, the same technology that allows Web clients to be mobile (i.e., wireless network interfaces) also enables the deployment of wireless Web servers.

☆ This paper is a significantly extended version of prior work published at MWAN'04 [1]. This current version (July 2005) has been revised and updated according to the journal reviewer comments received on March 23, 2005.

* Corresponding author. Tel.: +1 403 220 6780; fax: +1 403 284 4707.

E-mail address: carey@cpsc.ucalgary.ca (C. Williamson).

1570-8705/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2006.01.001

Wireless Web servers play a useful role in *short-lived networks*. A short-lived or *portable* network is created spontaneously, in an *ad hoc* fashion, at a particular location in response to some event, either scheduled or unscheduled. The network operates for some short time period (minutes to hours), before being disassembled, moved, and reconstituted elsewhere.

There are several distinguishing characteristics of a portable short-lived network. Often, the location of the needed network is not known a priori. There may not be *any* existing network infrastructure, either wired or wireless, at the needed location. In addition, the time at which the network is needed may not be known. Deployment may need to be spontaneous, with unknown (but often bounded) operating duration. The number of users for the network is typically small (e.g., 10–100), bandwidth requirements are moderate, and the geographic coverage area for the network is limited. More importantly, there is often a need for either data dissemination or data collection at the site of the network. In most cases, the data access requirement is for a "closed" set of specialized content, rather than general Internet content.

Examples of deployment scenarios for short-lived networks are sporting events, press conferences, conventions and trade shows, disaster recovery sites, and classroom area networks. The potential for entertainment applications (e.g., media streaming, home networking, multi-player gaming) is also high. In many of these contexts, an ad hoc wireless network with a wireless Web server as an information repository provides a suitable solution.

In this paper, we explore the feasibility of wireless Web server deployment in classroom area networks. The paper starts with empirical measurements from wireless Web server usage in a classroom environment to show the practicality of its operation. These measurements are then augmented with laboratory tests to determine experimentally the upper bounds on achievable performance. In particular, we focus on the performance capabilities of an Apache Web server running on a laptop computer with an IEEE 802.11b wireless LAN interface. We study in-building Web performance for wireless Web clients. All mobile computers are configured in ad hoc mode, since no existing network infrastructure is assumed. The clients download content from the wireless Web server. A wireless network analyzer is used to collect and analyze traces from the experiments, with traffic analysis spanning from the Medium Access Control (MAC) layer to HTTP at the application layer.

Our experiments focus on the HTTP transaction rate and end-to-end throughput achievable in an ad hoc wireless network environment, and the impacts of factors such as number of clients, Web object size, and persistent HTTP connections. The results show the impacts of the wireless network bottleneck, either at the client or the server, depending on the Web workload. Persistent HTTP connections offer significant improvements both in throughput and in fairness for mobile clients accessing content from a wireless Web server.

The remainder of this paper is organized as follows. Section 2 provides background information on IEEE 802.11b, TCP, and HTTP. Section 3 presents an overview of the classroom measurements from our study. Section 4 describes the experimental methodology for lab-based measurements. Section 5 presents the measurement results and analyses. Finally, Section 6 summarizes the paper and describes ongoing work.

2. Background and related work

2.1. The Web and Web performance

The Web relies primarily on three communication protocols: IP, TCP, and HTTP. The Internet Protocol (IP) is a connection-less network-layer protocol that provides global addressing and routing on the Internet. The Transmission Control Protocol (TCP) is a connection-oriented transport-layer protocol that provides end-to-end data delivery across the Internet [2]. Among its many functions, TCP has flow control, congestion control, and error recovery mechanisms to provide reliable data transmission between sources and destinations. The robustness of TCP allows it to operate in many network environments. Finally, the Hyper-Text Transfer Protocol (HTTP) is a request–response application-layer protocol layered on top of TCP. HTTP is used to transfer Web documents between Web servers and Web clients. Currently, HTTP/1.0 [3] and HTTP/1.1 [4] are widely used on the Internet.

The overall performance of the Web depends on the performance of Web clients, the Web server, and the network in between. The primary challenge in the context of wireless ad hoc networking is the wireless channel, which is often characterized by limited bandwidth, high error rates, and interference from other users on the shared channel. The obvious concern is that TCP and HTTP may suffer degraded performance over wireless ad hoc networks.

2.2. Wireless Internet and IEEE 802.11b WLANs

Wireless technologies are playing an increasingly prominent role in the global Internet infrastructure. One of the popular technologies in the wireless LAN market is the IEEE 802.11b standard. This "WiFi" (Wireless Fidelity) technology provides low-cost wireless Internet capability for end users, with up to 11 Mbps data transmission rate at the physical layer.

The IEEE 802.11b standard defines the channel access protocol used at the MAC layer, namely Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). It also defines the frame formats used at the data link layer: 128-bit preamble, 16-bit Start-of-Frame delimiter, 48-bit PLCP (Physical Layer Convergence Protocol) header, followed by a 24-byte MAC-layer header and variable size payload, which can be used for carrying IP packets.

In ad hoc mode, frames are addressed directly from the sender to the intended receiver using the corresponding MAC address in the frame header. Frames that are correctly received over the shared wireless channel are acknowledged almost immediately by the receiver. Unacknowledged frames are retransmitted by the sender after a short timeout (e.g., 1–10 ms), using the same MAC protocol.

2.3. Related work

There is growing literature on wireless traffic measurement and Internet protocol performance over wireless networks [5–12]. For example, Tang and Baker [11,12] discuss wireless network measurements from two different environments: a local area network, and a metropolitan area network. More recently, Balachandran et al. [5] report on network performance and user behaviour for general Internet access by several hundred wireless LAN users during the ACM SIGCOMM conference in San Diego in 2001. They find that for this set of technology-literate users a wide range of Internet applications are used, user behaviours are diverse, and overall bandwidth demands are moderate. Kotz and Essien [13] characterize campus-wide wireless network usage at Dartmouth College, but focus only on infrastructure mode using access points.

Our work differs from these in that we consider both a Web server and Web clients in the same wireless ad hoc network environment. The ad hoc scenario is of greater interest to us than the infrastructure-based scenario because of the "any time, any where" property for deployment, and the opportunity for peer-to-peer interaction in classroom, entertainment, and gaming applications. To the best of our knowledge, our work is the first to evaluate a wireless Web server in a short-lived wireless ad hoc network.

3. Empirical measurements

In January 2003, one of the authors (Williamson) was assigned to teach a graduate-level networking course in a "legacy classroom" environment that had neither wired nor wireless Internet access. Since much of the course content was provided on the Web (see http://www.cpsc.ucalgary.ca/~carey/CPSC601.38/archive/2003/), the solution was to create a mirrored copy of the course content and make it available in the classroom using a wireless Web server. The prototype was tested in the classroom in February 2003, during the course modules on wireless networking and network traffic measurement. Students were provided wireless laptops and PDAs for use in the classroom at this time.

Fig. 1 shows an example of the network traffic measurements from the classroom environment. Following the introductory part of the lecture that explained the experimental setup, the 14 students (sharing eight laptops and two PDAs) were allowed to download course content, review prior lecture notes, and begin preliminary work on a course assignment involving a 6 MB trace file. The graphs in Fig. 1 show the wireless network activity for a 25-min portion of the classroom measurements.

Fig. 1(a) shows the total number of TCP/IP packets transmitted on the wireless LAN per one-second interval during the trace. The traffic is bursty, with a high peak-to-mean ratio. The peak traffic rate approaches 700 packets per second. All packet exchanges take place directly between the Web clients and the Web server, over the shared WLAN. There is no multi-hop forwarding required in the classroom environment, and very limited host mobility.

Fig. 1(b) shows the total number of TCP/IP bytes exchanged across the WLAN, which correlates strongly with the number of packets exchanged. The peak data rate achieved is approximately 5.0 Mbps. This user-level throughput is typical for an IEEE 802.11b WLAN.

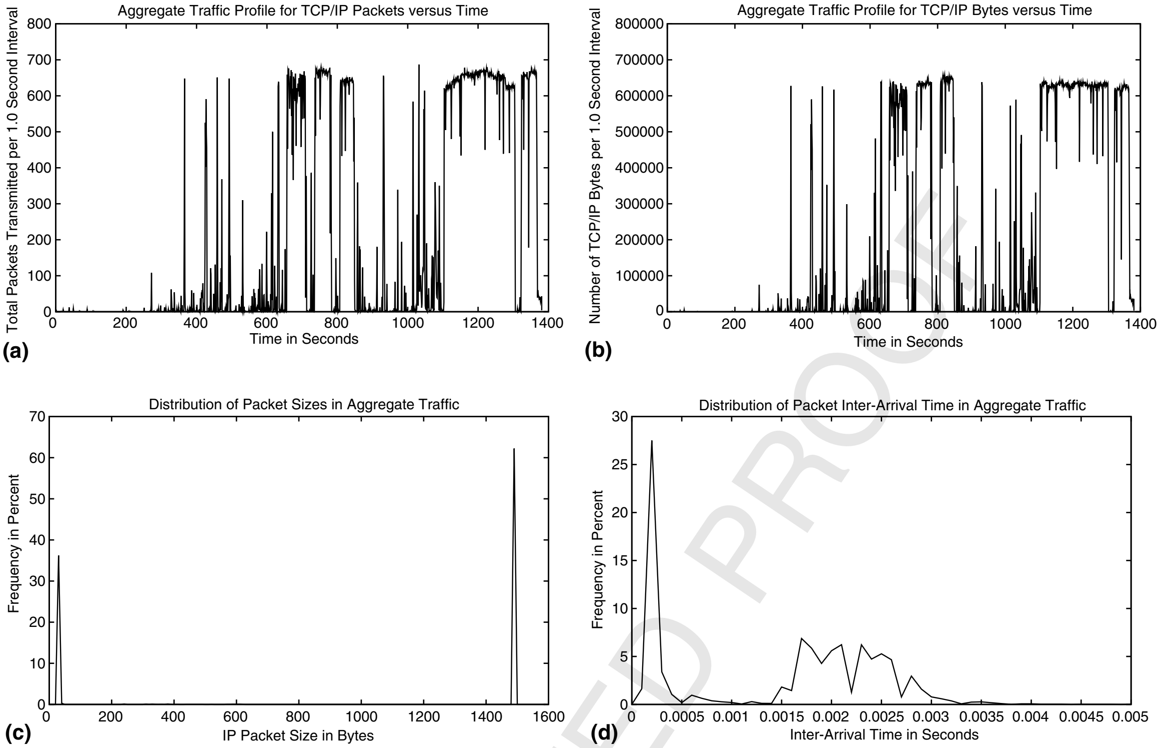

Fig. 1. Aggregate traffic measurements from portable wireless classroom area network: (a) packets versus time, (b) bytes versus time, (c) packet size distribution and (d) packet inter-arrival time distribution.

Fig. 1(c) shows the frequency distribution of the IP packet sizes observed. The distribution has two main peaks: one at 1500 bytes for full-size TCP/IP packets, and one at 40 bytes for TCP acknowledgements (ACKs). The peak for ACKs is lower because of TCP Delayed-ACKs, which typically result in one TCP ACK sent for every two TCP data packets received. A small proportion of other IP packet sizes are observed, but the distribution is clearly dominated by the two peaks.

Fig. 1(d) shows the distribution of the packet inter-arrival times on the WLAN. The tall peak on the left reflects the inter-arrival times between a TCP ACK and the next TCP data packet. The broader hump represents the typical time spacing between TCP data packets. There is significant dispersion to this distribution because of the nature of the CSMA/CA MAC protocol in IEEE 802.11b. A small percentage of inter-arrival times exceed 5 ms; these are not shown on the plot.

Fig. 2 illustrates the per-client activity for the six busiest Web clients. Clearly, the bursty aggregate traffic arises from the highly bursty behaviours of the individual clients. A single client is able to obtain most of the WLAN capacity when needed (e.g., Client 3 at time 760 s), while sharing the WLAN capacity if other clients are active (e.g., Clients 2, 4, and 6 around time 1200 s).

Our measurement experiences in the classroom environment lead to the following research questions:

- What is the maximum workload that a wireless Web server can handle in an IEEE 802.11b classroom area network?
- How does the wireless network performance bottleneck manifest itself?

The rest of the paper provides answers to these questions.

4. Experimental methodology

4.1. Experimental setup

Our laboratory experiments are conducted on an IEEE 802.11b wireless LAN in the Department of Computer Science at the University of Calgary.

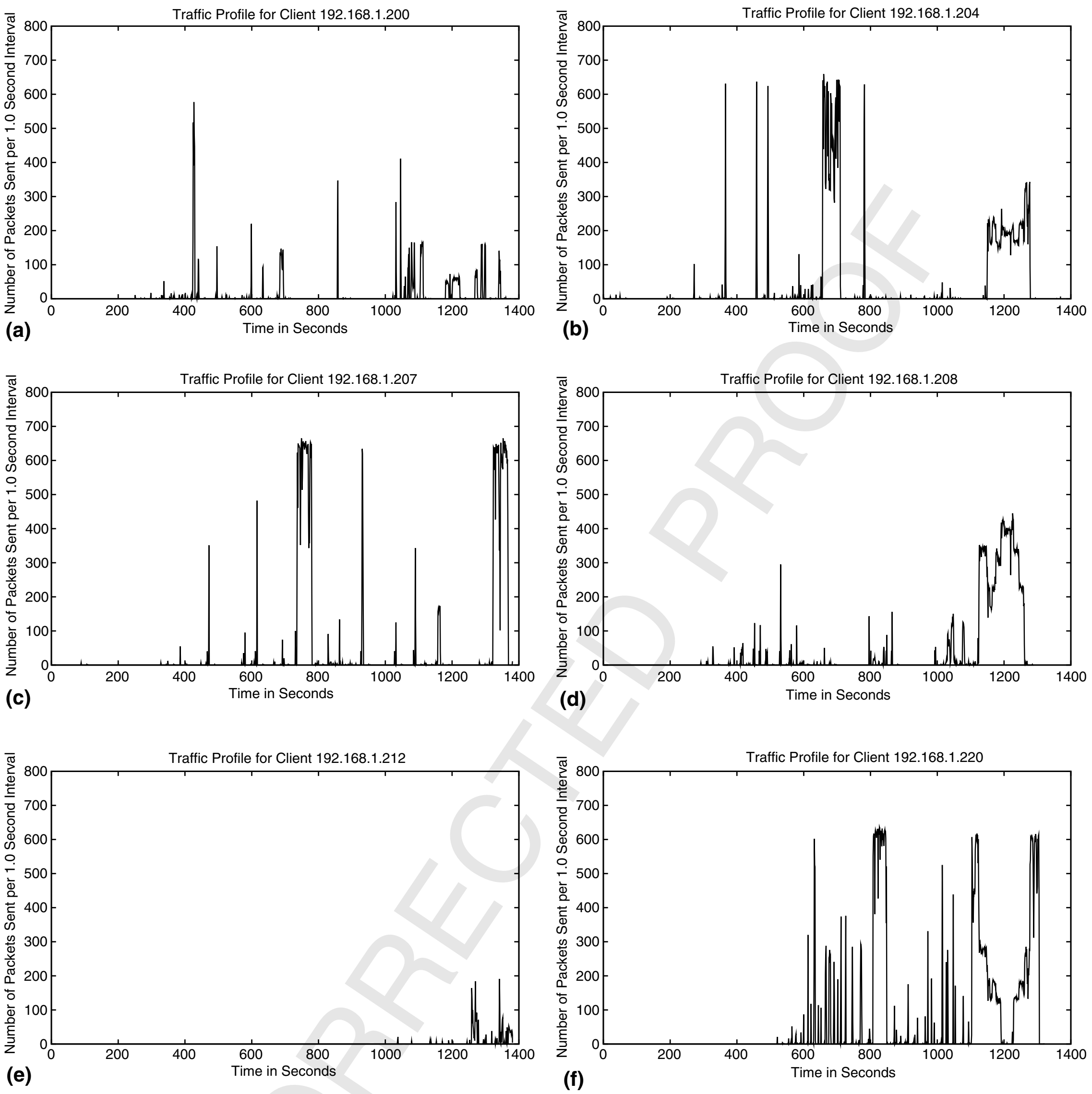

Fig. 2. Per-client traffic measurements from portable wireless classroom area network: (a) Client 1, (b) Client 2, (c) Client 3, (d) Client 4, (e) Client 5 and (f) Client 6.

The simple testbed shown in Fig. 3 consists of several mobile clients and one Web server. In addition, we use a wireless network analyzer to monitor the wireless channel.

Each of the client and server machines is a Compaq Evo Notebook N600c running RedHat Linux 7.3 and X windows. Each machine is equipped with a 1.2 GHz Mobile Intel Pentium III with 512-KB L2 cache and 128 MB of 133 MHz RAM. These represent well-resourced machines that are near state-of-the-art. All unnecessary OS processes were disabled prior to conducting measurements, to reduce contention for system resources.

Each laptop has a Cisco Aironet 350 Series Adapter for access to the IEEE 802.11b wireless LAN. The wireless cards are configured to operate in ad hoc mode. The cards are configured to use the Distributed Coordination Function (DCF) mechanism as the MAC protocol, with a (fixed) physical-layer transmission rate of 11 Mbps, and a maximum retry limit of 16 for MAC-layer retransmissions. The IP addresses for the laptops are

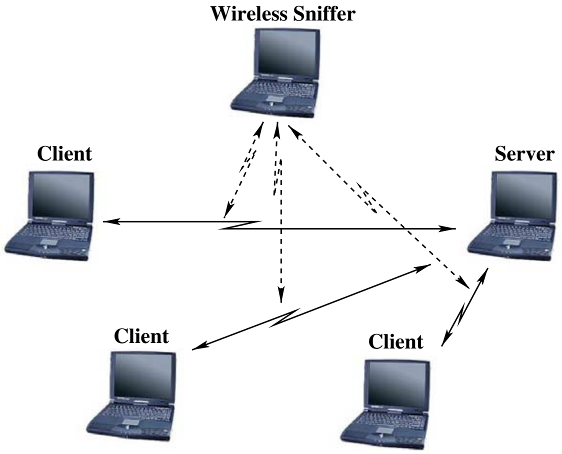

Fig. 3. Experimental setup for measurements.

assigned manually. We set the network-layer Maximum Transmission Unit (MTU) as 1500 bytes, and disable MAC-layer fragmentation. All client laptops are within line-of-sight of the server, and all laptops use a transmit power of 100 mW.

During our experiments, these laptops are the only machines operating on the wireless LAN. We do not consider node mobility, multihop, or ad hoc routing issues in our experiments; these important issues are studied in separate papers [14–16].

In our experiments, `httperf` [17] is used to generate client requests to the Web server. `httperf` is a Web workload generation tool developed at Hewlett–Packard Laboratories for Web performance measurement. It provides a flexible means for generating HTTP workloads and for measuring server performance.

The Web server in our experiments is an Apache HTTP server (Version 1.3.23). This version is a process-based implementation of Apache, which is a flexible and powerful HTTP/1.1-compliant Web server [18,19]. Apache is currently widely deployed on the Internet, used by approximately 70% of all Web sites [20].

Network traffic measurements are collected using a wireless network analyzer. The analyzer used is SnifferPro 4.6. This analyzer provides real-time capture of all observed traffic on the wireless LAN. Its wireless network card operates in promiscuous mode, recording all activity on the wireless LAN (i.e., frame transmissions, acknowledgements, CRC errors, collisions, and MAC-layer retransmissions). Decoding of the captured traces enables protocol analysis at the MAC, IP, TCP, and HTTP layers. After recording statistics about wireless network behaviour, we convert the traces to an ASCII format for detailed TCP traffic analysis with our own software tools.

In our experimental setup, the IEEE 802.11b wireless LAN is the performance bottleneck. The rationale for this observation is quite obvious, since the Apache Web server can easily sustain workloads in excess of 100 Mbps [19,21,22], yet the maximum user-level throughput theoretically achievable on an IEEE 802.11b WLAN is about 6 Mbps [23]. However, it is not clear how the WLAN bottleneck will affect Web protocol performance.

4.2. Experimental design

A one-factor-at-a-time experimental design is used to study the impacts of many factors on wireless Web server performance, including HTTP transaction rate, number of clients, transfer size, and HTTP protocol version. The experimental factors are summarized in Table 1. The values in bold font show the default levels used.

4.3. Web workload model

The experiments use synthetic Web workloads, which are easy to generate, analyze, and reproduce. While results would differ for other workloads (e.g., HTTP session models, used in workload generators such as SURGE [24]), our goals are to determine an upper bound on achievable performance, and to understand behaviour under overload conditions, using the simplest scenarios possible.

The experiments are conducted using `httperf` as an open-loop workload generator. We invoke `httperf` on the client machine, and send requests to the server at a specified rate to retrieve a target Web object repeatedly. Each test lasts 2 min, with each TCP connection issuing one or more HTTP requests, depending on the workload being gener-

Table 1
Experimental factors and levels for wireless Web server benchmarking

Factor	Levels
Number of clients	**1**, 2, 3, 4
Per-client TCP connection	
Request rate (per second)	**10**, 20, 30, ..., 160
HTTP transfer size (KB)	**1**, 2, 4, 8, ..., 64
Persistent connections	**No**, yes
HTTP requests per connection	**1**, 5, 10, 15, ..., 60

ated. The “user abort” timeout in `httperf` is set to 5 s. This timeout value is used when establishing a TCP connection, when sending an HTTP request, when waiting for a reply, and when receiving a reply. If no forward progress is made on any of these activities during the allotted time, the client aborts the corresponding call and reports it as an error.

4.4. Performance metrics and instrumentation

Performance data in our experiments come primarily from `httperf` and the wireless network analyzer, though we also collect some performance data, such as `netstat` information, on client and server machines as well. The `httperf` tool reports application-layer statistics on HTTP behaviours (e.g., reply rate, throughput, response time, error rate). These statistics are used for a high-level overview of the performance results. Detailed performance data are available from the wireless network analyzer, enabling traffic analysis from the MAC layer to the HTTP layer. These traces are used to assess wireless channel contention, TCP protocol behaviours, and HTTP transaction performance.

4.5. Validation

Since our experiments record both application-layer and network-layer measurements, it is possible to do a sanity check on the data to ensure proper interpretation of the experimental results.

The first validation test checked the timestamps on the TCP SYN requests to ensure that `httperf` was generating workloads at the specified request rate. For example, at a rate of 10 connections per second, a new TCP SYN request should appear on the network every 0.1 s. This property was verified for the Cisco Aironet 350 wireless network cards used in our experiments.

The second validation test compared network packet traces collected using the wireless network analyzer with those collected using `tcpdump`. This comparison identified a subtle but important point: traces collected using the wireless network analyzer represent the *analyzer’s view* of the activity on the wireless channel, which is *not necessarily the same* as those of the client or the server. Because the receive antenna for each machine operates independently, the received signals could differ for each device. One machine could interpret a received frame as successful, while another could reject it as a “Bad CRC”. In other words, “what you see at the Sniffer is not necessarily what you got at the client or server”.

This measurement artifact manifests itself in several ways: successful TCP connections for which either the client’s opening SYN or the server’s SYN ACK was not seen; MAC-layer retransmissions of frames that were already received perfectly; and TCP acknowledgements for segments that were never sent. We have quantified this anomaly as affecting fewer than 1% of the TCP connections studied, and thus have *not* made efforts to filter this artifact from the measurements with pre-processing. Pre-processing would involve inserting some packets with unknown timestamps into the trace, and removing other packets from the trace. Running `tcpdump` on the client and the server is one way to avoid this problem, since it only records packets that actually traverse the TCP/IP protocol stack. However, the `tcpdump` overhead would affect the measurement results.

While `tcpdump` was not run for the experiments shown in the paper, it was used extensively to help understand system behaviour during preliminary tests. We also used `netperf` [25] to determine the maximum user-level throughput achievable between client and server for large transfers on our wireless LAN. Throughput is typically 5–6 Mbps, depending on the TCP transfer size, socket buffer size, operating system, MTU, wireless card, and driver configuration used [26].

5. Experimental results

This section presents selected measurement results from our experiments with a wireless Web server in a wireless ad hoc network.

5.1. Experiment 1: request rate

The purpose of the first experiment is to determine the maximum sustainable load for the wireless Web server. In this experiment, only a single Web client machine is used. The client, server, and Sniffer laptops are all less than 1 m apart. The wireless channel is assumed to be excellent. The size of the Web object retrieved from the server is 1 kilobyte (KB). The experiments start with a request rate of 10 requests per second, using non-persistent connections. That is, there is exactly one HTTP “GET” request per TCP connection; “TCP connection rate" and "HTTP transaction rate" are thus synonymous for this experiment. When one test is complete, the test is repeated with the next higher HTTP transaction rate, from 10 to 160 requests per second. Each HTTP/1.0 transaction generates 10 TCP packets (six sent by the client, four by the server), as shown in Fig. 4(a). Each TCP packet requires access to the IEEE 802.11b WLAN for the transmission of the frame and its corresponding MAC-layer acknowledgement.

Fig. 5 shows the application-layer performance results reported by `httperf` for this experiment. The plots show the successful HTTP transaction rate in Fig. 5(a), the achieved user-level throughput in Fig. 5(b), the user-perceived response time in Fig. 5(c), and the "user abort" error rate in Fig. 5(d). In all four graphs, there are two regimes: the "normal" operating regime for feasible loads, and the "overload" regime generated by the open-loop workload.

Fig. 5(a) shows the successful HTTP transaction rate as the offered load increases. The HTTP transaction rate increases linearly at first with offered load (as expected), up to about 85 requests per second. Beyond this point, there is some instability, and a drop to a lower plateau. Qualitatively similar results are observed in experiments with the same client and server laptops in a 10 Mbps wired-Ethernet LAN, though the peak HTTP transaction rate in an Ethernet LAN is 380 requests per second, higher by more than a factor of 4. Clearly, the channel access overhead in the wireless ad hoc network limits the performance.

The low HTTP transaction rate in the wireless ad hoc network is explained by the bottleneck at the *client* network interface, where packets wait at the link-layer queue for medium access on the WLAN. Fig. 6 shows this behaviour for high load on a specially instrumented Linux kernel; the client queue in Fig. 6(a) fills in about 10 s.

With the default queue size setting of 100 in the Linux kernel, many packet drops occur from this link-layer queue, even *before* the packets make it to the network. The server does not receive enough requests to keep it busy, so its queue in Fig. 6(b) does not fill.

Increasing the client queue size limit is pointless, since the packet arrival rate to the queue exceeds the packet service rate from the queue. We have verified this experimentally with other (larger) settings for the queue size. The steady-state packet loss rate is the same, regardless of the queue size limit. The only things that change are the time required to fill the

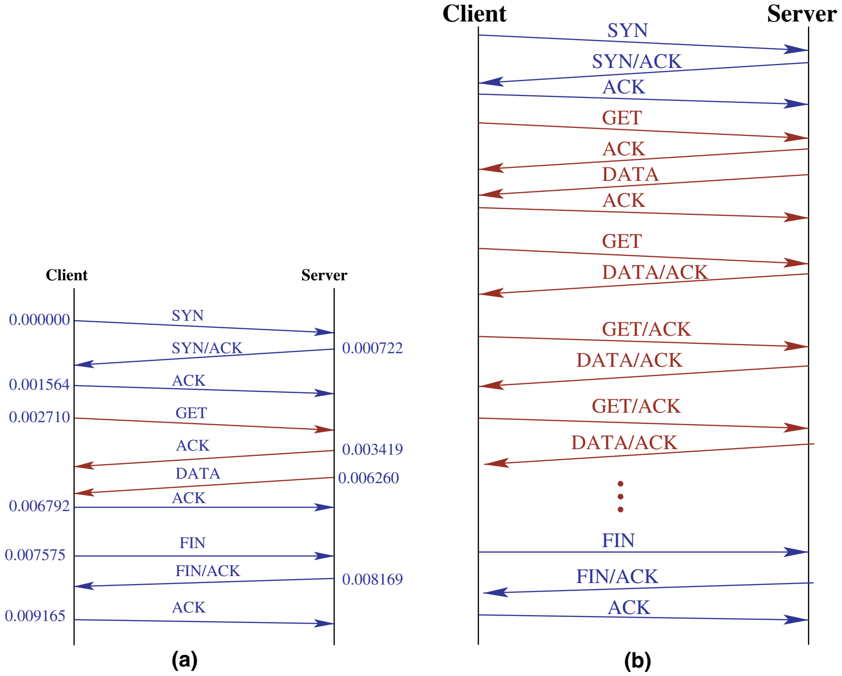

Fig. 4. Example of HTTP transactions using TCP: (a) non-persistent (e.g., HTTP/1.0) and (b) persistent (e.g., HTTP/1.1).

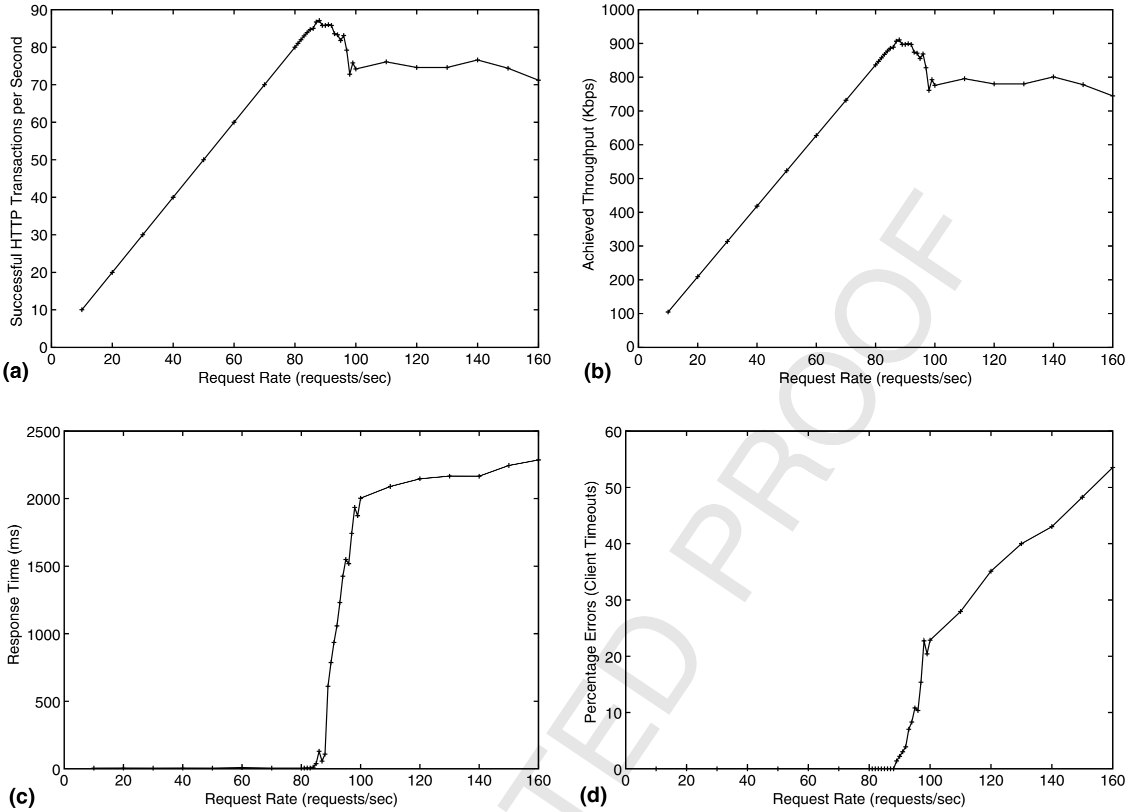

Fig. 5. `httperf` Performance results for Experiment 1 varying request rate (one client, 1 KB, non-persistent): (a) successful transactions, (b) achieved throughput, (c) response time and (d) error rate.

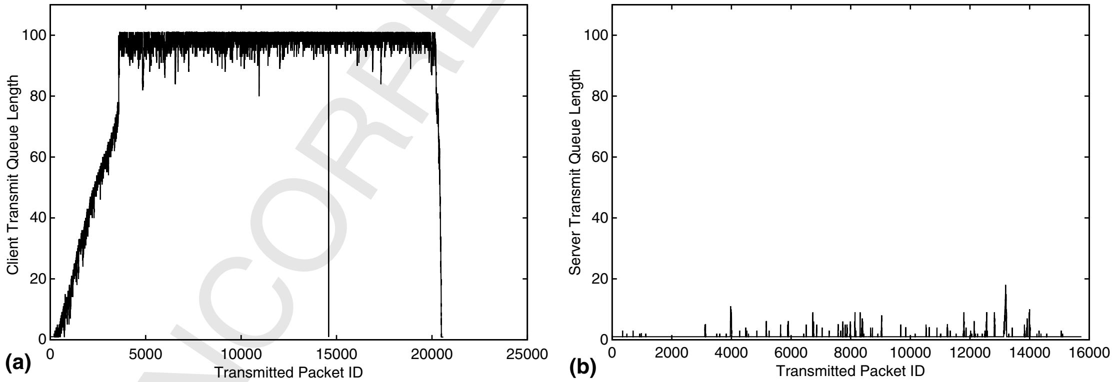

Fig. 6. Link-layer transmit queue behaviour for Experiment 1 (one client, 1 KB, non-persistent): (a) Client and (b) server.

queue, and the average delay for packets that are waiting for transmission on the WLAN.

The Linux kernel has no flow control or back-pressure mechanism to prevent `httperf` from overflowing the queue. While each TCP connection sends only one data packet, the control packet overhead and the sheer number of active TCP connections eventually overwhelms the queue. Aggregate coordination of multiple TCP flows is required to solve this problem [27], as is a more robust Linux kernel that checks for and signals queue overflow to the application layer.

The performance limit is also reflected in Fig. 5(b), which shows the application-layer throughput as a function of offered load. The peak throughput achieved is just under 1 Mbps, far from the nominal 11 Mbps capacity of the IEEE 802.11b wireless LAN. By contrast, experiments on the 10 Mbps wired-Ethernet LAN achieve a throughput of 3.8 Mbps.

With non-persistent connections, most of the packets are small control packets, and the TCP connection establishment overhead is high relative to the connection lifetime. Each transaction requires a three-way handshake for TCP connection setup, followed by a 74-byte HTTP GET request, a 1 KB HTTP response, and then a three-way handshake to close the TCP connection. A typical HTTP transaction (10 packets) takes about 9 ms on the wireless LAN. This HTTP transaction time is about four times longer than that observed in similar tests on a 10 Mbps Ethernet LAN. Again, the wireless MAC protocol overhead limits HTTP transaction performance.

Fig. 5(c) shows the average response time for the successful HTTP transactions. At low load, the response time is about 9 ms, with slight fluctuation as the offered load increases from 10 to 85 requests per second. When the transaction rate exceeds 85 requests per second, the response times increase significantly, eventually exceeding 2 s.

Fig. 5(d) shows `httperf` "user abort" errors from client-side timeouts. Under overload, aborts occur frequently.

Fig. 7 presents detailed measurement results for this experiment, based on traces collected by the wireless network analyzer. In Fig. 7, we show selected measurement results for low load (first row of graphs), medium load (second row), and high load (third row), as well as an overload scenario (bottom row). On each row, there are two graphs: a 60-s time-series plot of the TCP connection duration, defined as the elapsed time from first packet to last packet for successful HTTP transactions; and a marginal distribution (pdf) plot of the TCP connection duration.

The top row in Fig. 7 represents low load: 10 requests per second. The TCP connection duration in Fig. 7(a) fluctuates between 8 and 12 ms. The marginal distribution in Fig. 7(b) has a mean of 9.7 ms. Qualitatively similar results would be observed in an infrastructure-based WLAN scenario, except the transaction latency would be higher because of the additional round-trip time to the server on the wired network.

The second row in Fig. 7 represents medium load: 50 requests per second. Here, the time series plot in Fig. 7(c) shows greater variation. In particular, two large spikes are evident. The cause for these anomalies is the X windows system running on the client and server; disabling the X server and its daemon processes on both machines eliminates the spikes. The presence of the spikes is tolerable, since the spikes are brief (10–30 ms) and have minimal impact (e.g., the server or the client is 10–30 ms late in generating a SYN ACK, ACK, or FIN ACK) on the few (4 out of 3000) unlucky connections affected. Furthermore, these results arguably reflect realistic operating conditions, since Linux clients and servers are likely to run X windows in a classroom environment. Other than the two spikes, performance is relatively stable at this load. The mean TCP connection duration in Fig. 7(d) is 10 ms.

The third row of Fig. 7 represents high load (80 requests per second), approaching the previously-determined limit of 85 requests per second. In these graphs, there is more variability in the connection duration in Fig. 7(e), including some spikes, and a slight skew to the marginal distribution in Fig. 7(f). A separate analysis (not shown here) shows short-range correlation in the connection durations, implying queueing delays somewhere in the system; this queueing occurs at the client network card.

The bottom row of Fig. 7 represents an overload situation with 100 requests per second. In this scenario, the sustained overload eventually saturates the client's link-layer queue, leading to packet drops, retransmissions, and even TCP resets to abort failed transactions, as indicated by the `httperf` results in Fig. 5(d).

The effect of the queue buildup is apparent in Fig. 7(g): the connection durations initially grow with time, until the erratic overflow behaviour occurs. Note that the graphs in Fig. 7(g) and Fig. 7(h) have different vertical scales than the graphs above them: some successful TCP connections take over 20 s to complete. The unusually long durations arise because there is no `httperf` timeout for the closing FIN handshake in TCP. Many of the successful TCP connections last 3 s or more. These results represent connections that had a "SYN drop" at the client link-layer queue on the initial connection request: if the SYN retransmission 3 s later (a TCP default) is successful, the transaction proceeds as usual. If unsuccessful, `httperf` aborts the connection before the next TCP retrans-

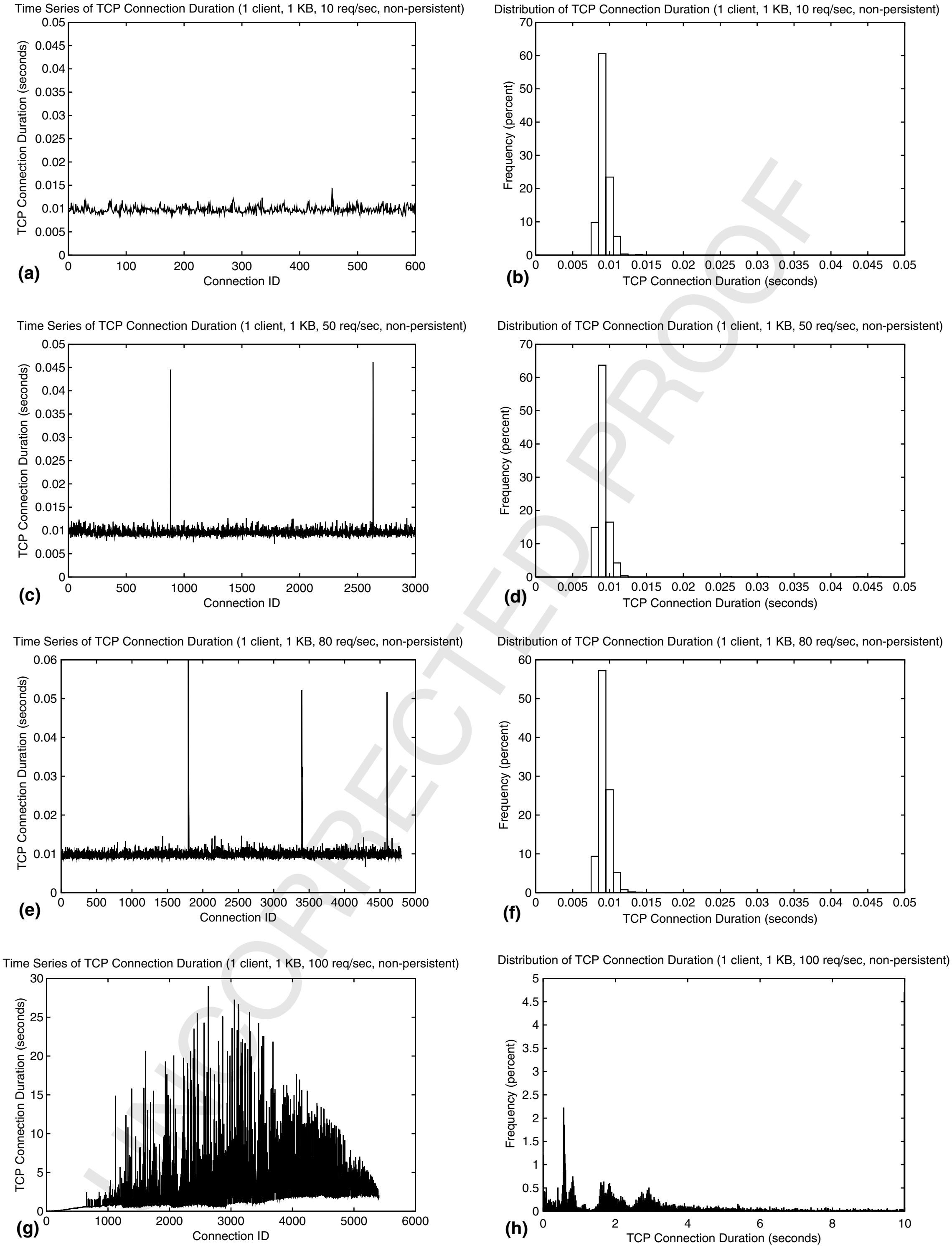

Fig. 7. Network traffic measurement results for Experiment 1 varying request rate: behaviour of TCP connection duration as a function of load (one client, 1 KB, non-persistent): (a) time series (low load), (b) marg. dist. (low load), (c) time series (med. load), (d) marg. dist. (med. load), (e) time series (high load), (f) marg. dist. (high load), (g) time series (overload) and (h) marg. dist. (overload).

mission (6 s later), because of the 5-s timeout for client aborts.

5.2. Experiment 2: multiple clients

The next experiment uses multiple client machines to generate HTTP requests to the wireless Web server, using the same methodology as in Experiment 1. With two or more clients, a higher aggregate throughput is achieved (110 HTTP transactions per second), about 30% higher than the throughput achieved with a single client.

The higher throughput observed implies that the bottleneck is now at the server's wireless network interface. Fig. 8 confirms that this is the case. This graph shows the link-layer transmit queue behaviour from a high load experiment with two clients. Fig. 8(a) shows the client-side queue, while Fig. 8(b) shows the server-side queue. Since both clients behave similarly, results from only one client are shown. While each client generates requests at a rate below the peak determined in Experiment 1, the server experiences significant channel access delays to send its packets, some of which are large TCP data packets. Qualitatively similar results would be observed in an infrastructure-based WLAN scenario, except the queue would occur at the Access Point, rather than at the server.

Fig. 9 indicates a new performance problem: unfairness for multiple clients under overload. That is, one client obtained a higher proportion of the throughput at the expense of another.

Fairness problems can occur in wireless networks for many reasons. Unfairness can be caused by load imbalance [28], heterogenous transmission rates [29], differences in wireless channel quality [30], contention patterns in the wireless channel access [31], or packet losses at a point of congestion shared by competing upstream and downstream flows [32]. However, the unfairness problem that we observe is different from any of these identified in the literature.

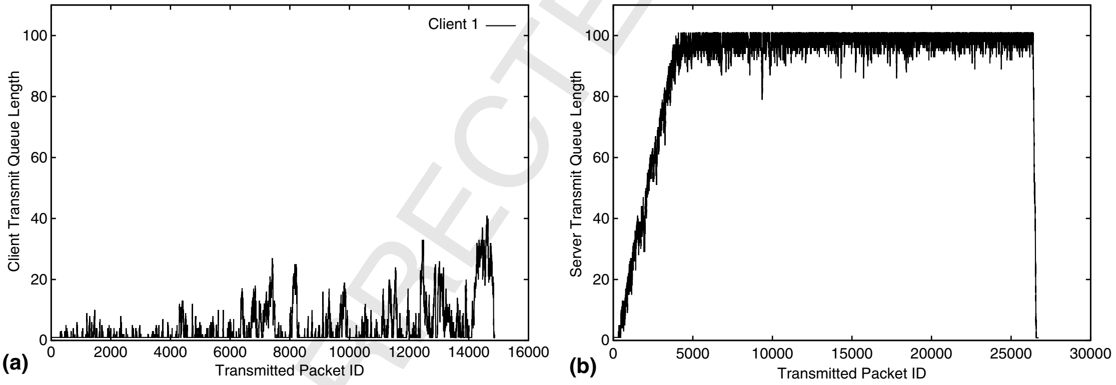

Fig. 8. Link-layer transmit queue behaviour for Experiment 2 (2 clients, 1 KB, non-persistent): (a) Client 1 and (b) Server.

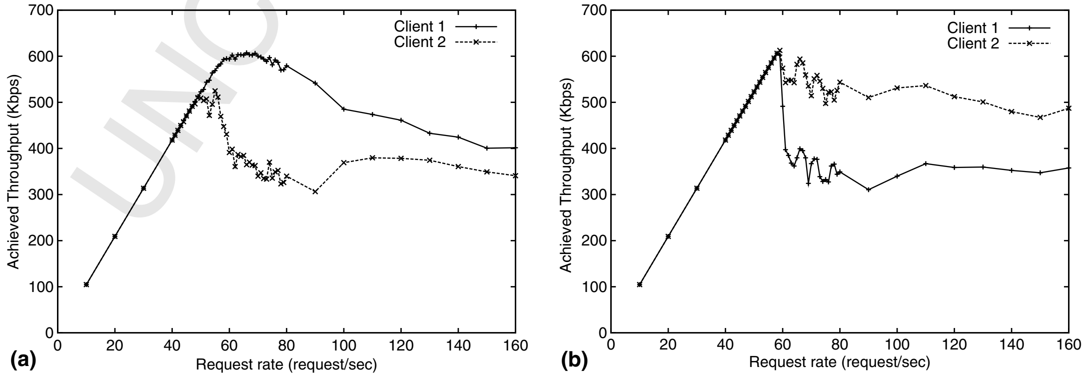

Fig. 9. Unfairness problem with two clients: (a) Test 1 and (b) Test 2.

A careful investigation of the traces shows that the relative phasing (i.e., synchronization) between the client machines is an important issue. Because each client generates requests deterministically at the same rate using identical hardware and software, the relative phasing of clients at startup determines the relative ordering of requests in the server queue. While the relative phasing may change each time the experiment is run (see Fig. 9), we have observed the unfairness problem repeatedly in our overload experiments with two clients and with three clients.

Fig. 10 presents detailed measurement results for the unfairness problem in an overload scenario. In this experiment, Client 1 sent its first TCP SYN request to the Web server slightly later than Client 2. The TCP connections created by Client 1 experience much longer time on average, and a disproportionately large share of the TCP resets and client aborts.

Further investigation of the link-layer queue behaviour shows transient bottleneck effects at both the client and the server, though packet drops at the server dominate. Client 1 experiences more packet losses than Client 2.

Table 2 summarizes the packet-level statistics for Client 1 and Client 2. In these experiments, Client 2 starts first, and Client 1 starts a random short time later. All transactions have a structure similar to that shown in Fig. 4(a) for HTTP/1.0.

The values highlighted in bold font in Table 2 show the large discrepancies in TCP-layer retrans-

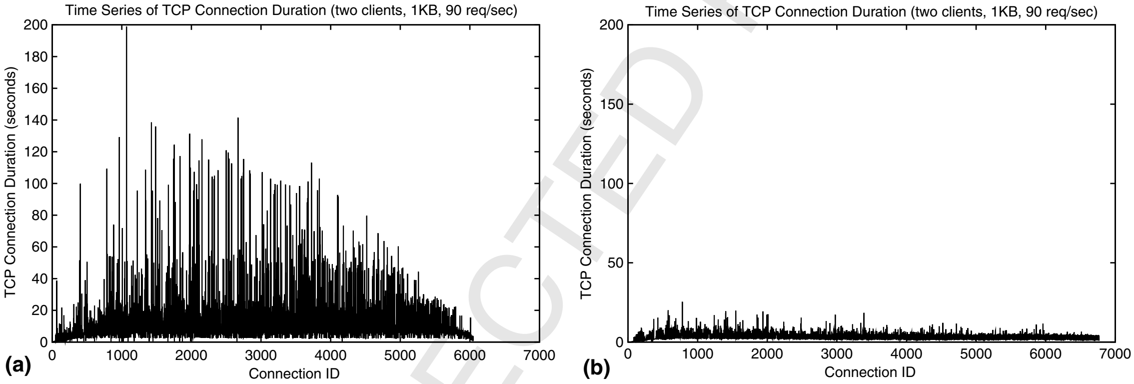

Fig. 10. Time series of TCP connection duration (two clients, 1 KB, 90 req/s): (a) Client 1 and (b) Client 2.

Table 2
Detailed packet statistics for unfairness problem with two clients

Item	Client 1				Client 2			
HTTP rate (req/s)	10	50	80	90	10	50	80	90
HTTP transactions	1200	6000	9600	10,800	1200	6000	9600	10,800
Start time (s)	0.250	0.226	0.323	0.440	0.000	0.000	0.000	0.000
SYNs	1199	5875	12,913	14,947	1200	5997	14,290	17,063
SYN Retxmit (TCP)	0	0	3535	4404	0	0	4726	6284
SYN Retxmit (MAC)	1	141	407	515	0	23	552	648
SYN ACK Retxmit (TCP)	0	0	**1391**	**1679**	0	0	**695**	**244**
SYN ACK Retxmit (MAC)	4	258	945	935	6	241	1084	1073
GET Retxmit (TCP)	0	0	1072	913	0	0	1325	1251
GET Retxmit (MAC)	0	122	254	226	3	117	340	349
DATA Retxmit (TCP)	0	0	**142**	**226**	0	0	**0**	**1**
DATA Retxmit (MAC)	1	86	188	199	0	70	217	248
FINs	1199	5953	6072	5953	1200	5986	7216	6863
FIN Retxmit (TCP)	0	0	206	184	0	0	167	106
FIN Retxmit (MAC)	0	57	200	183	1	130	297	280
FIN ACK Retxmit (TCP)	0	0	**1100**	**1161**	0	0	**0**	**14**
FIN ACK Retxmit (MAC)	0	22	258	248	1	60	274	245

missions experienced by the two clients (e.g., 1161 FIN ACK retransmissions for Client 1, versus 14 for Client 2). These large differences all occur in table rows for *server-generated* TCP packets in the HTTP transactions, and the differences manifest themselves at the TCP layer, rather than at the MAC layer. Client 2 experienced much better performance than Client 1.

The easiest way to explain this phenomenon is to think of the server as sending a pair of back-to-back packets to the link-layer queue, where the first packet is from Client 2, and the second packet is from Client 1. If the queue has ample room, then both packets will be accepted. If the queue is full, then both packets will be accepted. However, if the queue has room for just 1 packet, then the packet for Client 2 will be queued for transmission, while the packet for Client 1 will be dropped. The statistics in Table 2 indicate that the latter case happens quite frequently, especially for SYN ACK and FIN ACK packets.

For the synthetic `httperf` workloads, the relative phasing of sources has an important impact on TCP fairness and overall Web performance. While these phasing effects are unlikely to occur in human-generated Web client workloads, we speculate that heterogenous client hardware (e.g., fast versus slow) could lead to similar unfairness problems. Randomization may be required to break up these phasing effects.

5.3. Experiment 3: persistent HTTP connections

The next experiment considers persistent HTTP connections. With a persistent connection, multiple HTTP transactions can be sent on the same TCP connection, prior to it being closed [4]. This approach amortizes the TCP overhead across multiple HTTP transactions, and improves HTTP server performance [33,34].

In this experiment, the TCP connection rate is 10 requests per second, and the transfer size is 1 KB.

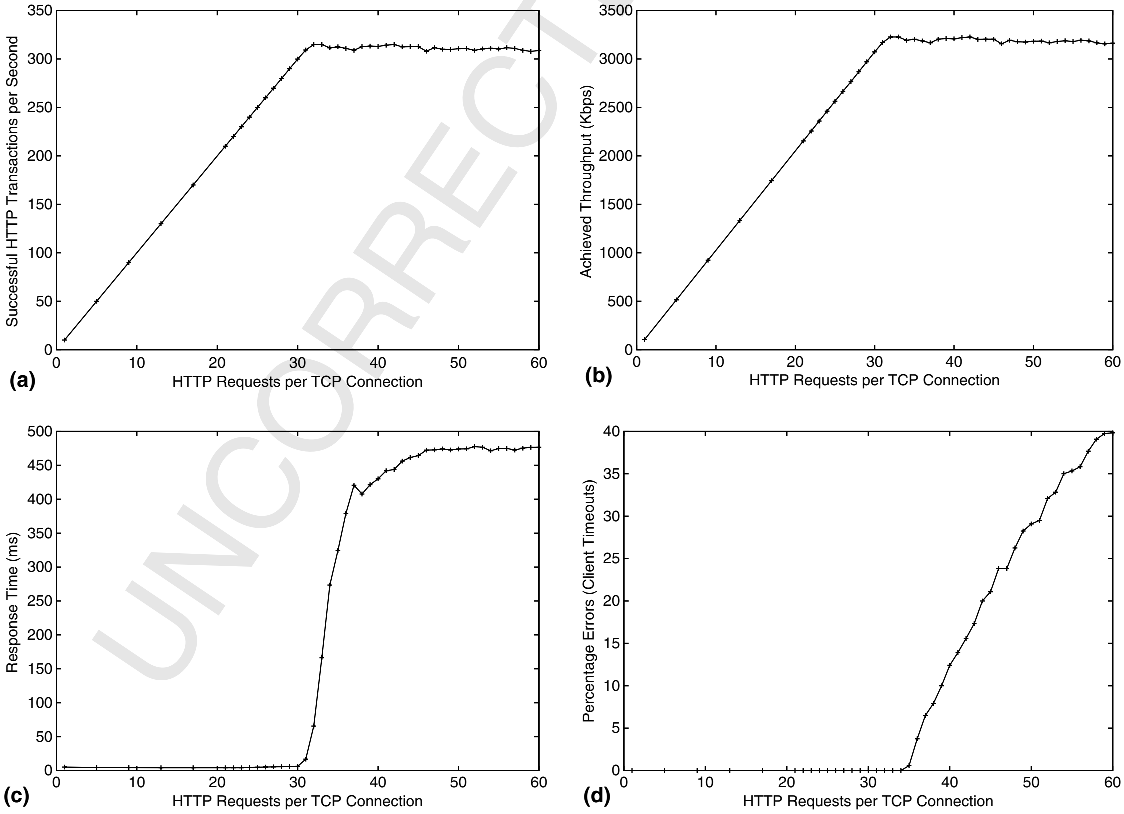

Fig. 11. `httperf` Performance results for Experiment 3 with persistent connections (one client, 1 KB, 10 conn/s, persistent): (a) successful transactions, (b) achieved throughput, (c) response time and (d) error rate.

The number of HTTP transactions per TCP connection is varied.

Fig. 11 shows the application-layer performance results reported by `httperf` for this experiment. Fig. 11(a) shows that the successful transaction rate increases as the number of HTTP requests per connection is increased. The highest rate achieved is 320 HTTP transactions per second. User-level throughput in Fig. 11(b) reaches a peak of 3.2 Mbps with 32 HTTP transactions per TCP connection. Beyond that point, server throughput is relatively stable, though the average HTTP response time in Fig. 11(c) increases sharply.

These results show that persistent connections offer a 350% improvement in performance over non-persistent connections. Compared to the results in Fig. 5(b), the maximum throughput has increased from 900 Kbps to 3.2 Mbps. In the 10 Mbps wired-Ethernet experiments, persistent connections double the performance from 380 to 760 HTTP transactions per second. The user-level throughput reaches 7.8 Mbps.

Clearly, persistent connections offer many advantages: fewer control packets (TCP SYN and FIN) on the network, and amortization of the TCP handshakes over many HTTP transactions. These advantages apply to any network environment, wired or wireless, but they are particularly important when the wireless LAN is the bottleneck.

While the performance advantages of persistent connections are generally well-known, their primary benefit on the Internet is in reducing the number of round-trip times (RTTs) between client and server. In the wireless ad hoc network scenario, the RTT is negligible, yet persistent connections are still highly beneficial.

The primary benefit is the reduction in the number of WLAN packet transmissions. With persistent connections (see Fig. 4(b)), the first HTTP transaction inside the TCP connection requires only four TCP packets (GET, ACK, DATA, ACK) instead of 10, while subsequent HTTP transactions in the same TCP connection typically require only two packets, since TCP can piggyback ACKs on out-

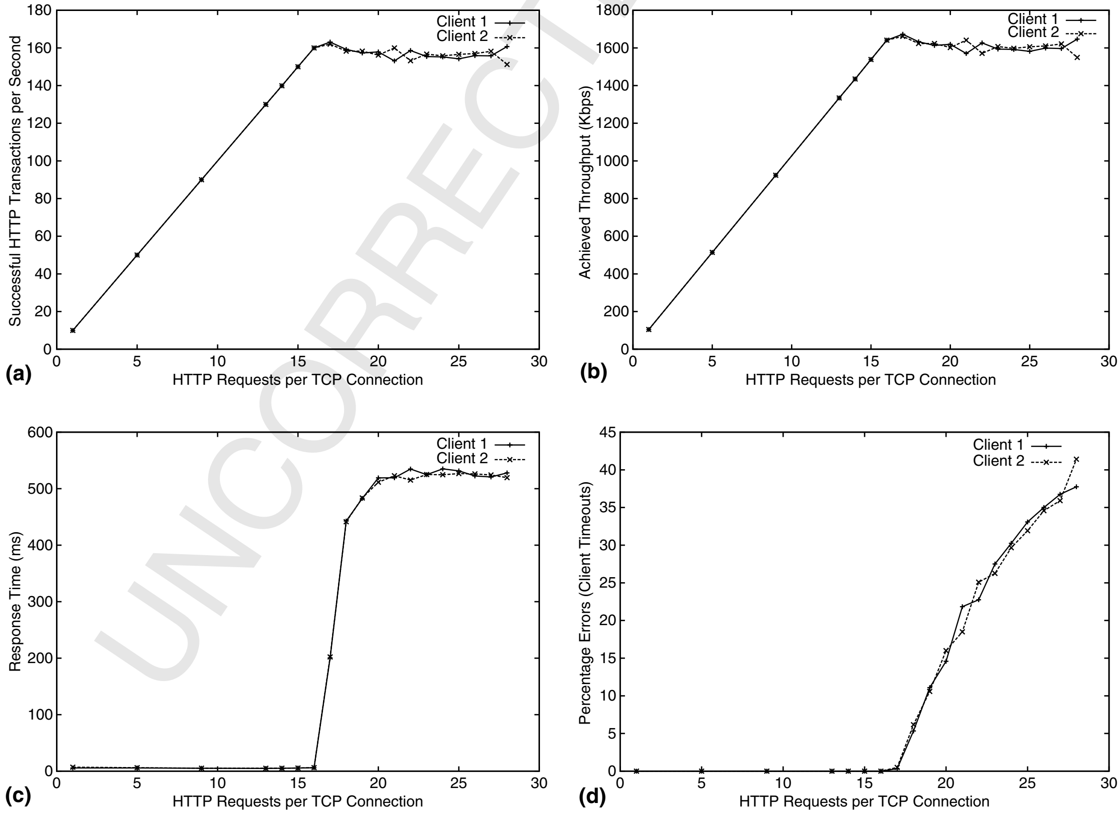

Fig. 12. `httperf` Performance results for Experiment 3 with persistent connections (two clients, 1 KB, 10 conn/s, persistent): (a) successful transactions, (b) achieved throughput, (c) response time and (d) error rate.

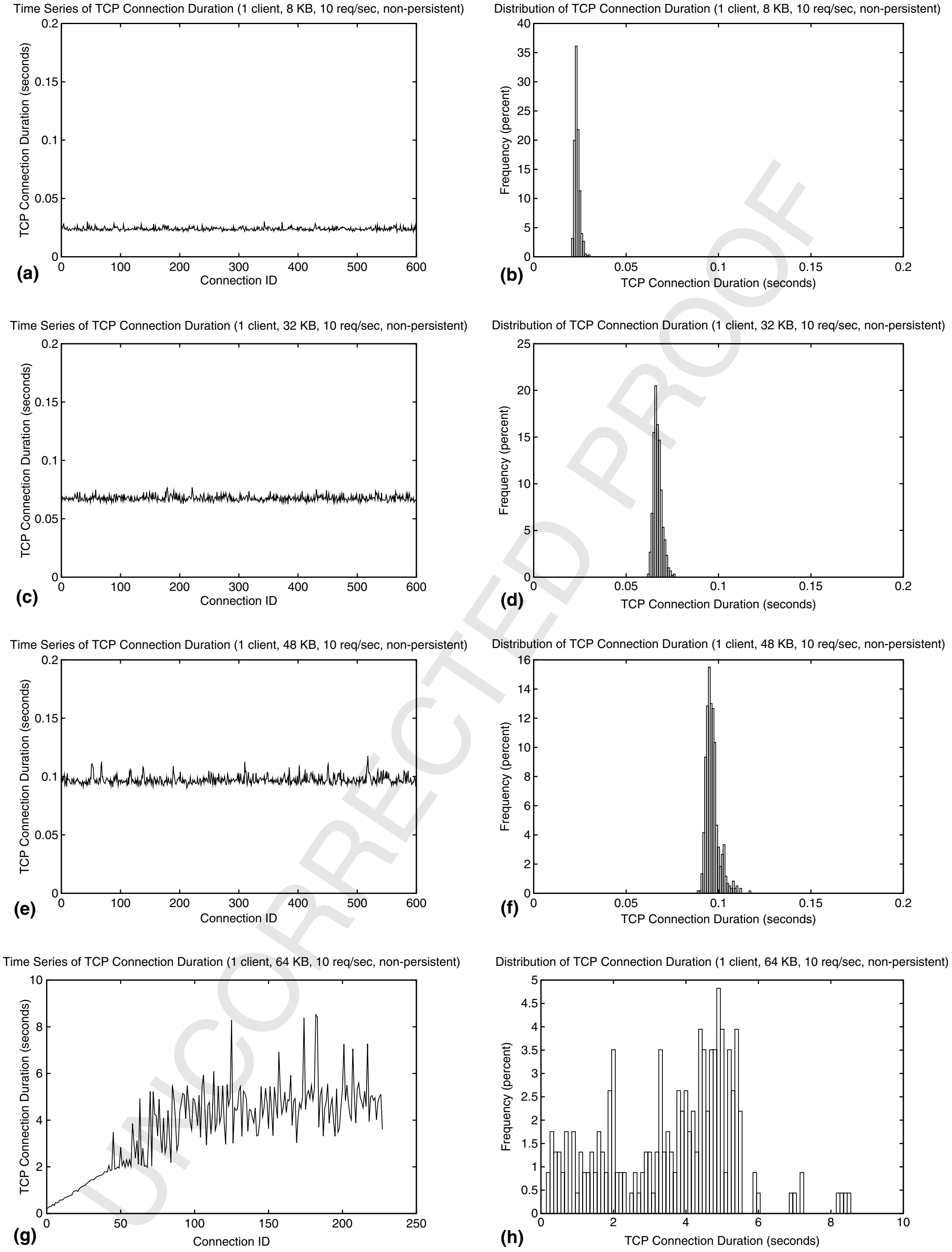

Fig. 13. Network traffic measurement results for Experiment 4: behaviour of TCP connection duration as a function of HTTP transfer size (1 client, 10 req/s, non-persistent): (a) time series (8 KB), (b) marg. dist. (8 KB), (c) time series (32 KB), (d) marg. dist. (32 KB), (e) time series (48 KB), (f) marg. dist. (48 KB), (g) time series (64 KB) and (h) marg. dist. (64 KB).

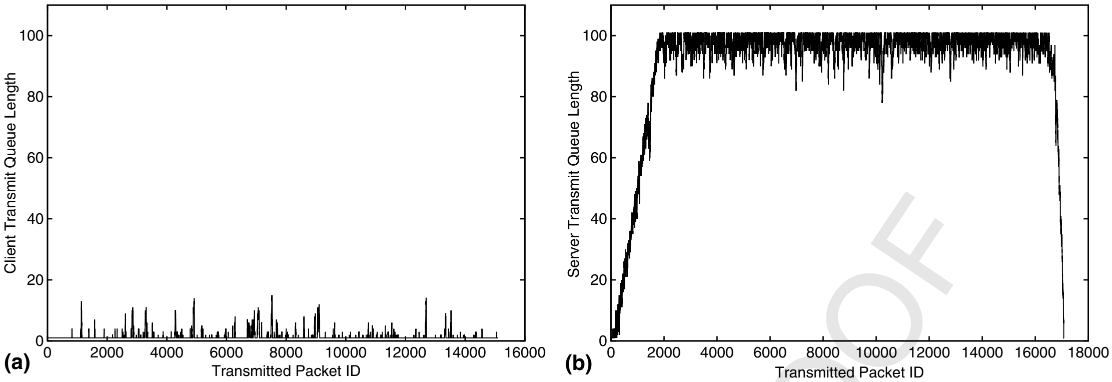

Fig. 14. Link-layer transmit queue behaviour for Experiment 4 (one client, 64 KB, non-persistent): (a) Client and (b) server.

bound GET and DATA packets. This five-fold reduction in the number of TCP packets per HTTP transaction dramatically reduces the demand on the wireless LAN medium access bottleneck, improving HTTP performance dramatically.

Fig. 12 shows the results from the persistent connection experiment with two clients. As expected, the total HTTP transaction rate for the server remains the same (320 HTTP transactions per second). The two clients share the server and network resources equally. This observation indicates that the unfairness problem noted earlier for two clients is primarily related to the packet loss dynamics during TCP handshaking. Losses of data packets within a TCP connection are less serious, because they can often be recovered efficiently using TCP's fast retransmit mechanism, rather than a timeout.

5.4. Experiment 4: transfer size

The next experiment studies the impact of HTTP response size on network throughput, for a single client issuing 10 requests per second to the server. The transfer size is 1 KB for the first run of the experiment, and is then increased to 2 KB, 4 KB, and so on in the subsequent runs.

Fig. 13 presents the results from this experiment, for four selected transfer sizes: 8 KB, 32 KB, 48 KB, and 64 KB. These values represent light load, medium load, heavy load, and overload conditions for the wireless Web server.

Fig. 13 shows the obvious result that as the HTTP transfer size increases, the mean TCP connection duration increases, as does the variance and skew of the distribution. The 8 KB transfers complete in about 24 ms each, representing an average throughput of 2.8 Mbps, including HTTP header overhead. The 32 KB transfers complete in about 67 ms, for an average throughput of 3.9 Mbps. The results for 48 KB transfers and for 64 KB transfers represent samples from just below and just beyond the "saturation point". That is, a 48 KB transfer completes on average in just under 100 ms (4.1 Mbps), which means that the server can keep up with a sustained arrival rate of 10 requests per second. A 64 KB transfer, on the other hand, takes well over 100 ms on average, so the open-loop workload generator creates overload. Experiments on the 10 Mbps wired-Ethernet LAN show that the server can handle 10 requests per second for 96 KB transfers before the dropoff in performance occurs. The peak throughput achieved is 8 Mbps.

In this experiment, the wireless network bottleneck is at the *server* network interface, since the server transmits more packets than the client, and larger packets as well. The `httperf` request rate is modest (10 requests per second), placing little stress on the client-side queue. Fig. 14 illustrates the queue buildup at the server, while Fig. 13(g) shows the impact of this queue on HTTP response time, which increases by more than an order of magnitude. The large delay is due to the sizes of the queued data packets.

Detailed analysis of the 64 KB scenario reveals a new performance problem: about 50% of the TCP connections are aborted with a TCP reset[1] prior to completion. However, relatively few (less than 2%) of these connections failed during the opening TCP handshake; most were aborted partially

---

[1] These TCP resets are caused by the 5-s client abort timeout in `httperf`, for a transfer that theoretically should take 130 ms. Human users may behave differently.

through the transfer. On average, each of the reset connections sent 68 packets and 47 KB of data.

Network bandwidth is the scarce resource in this experiment. The main concern is "network thrashing": a large portion of the wireless channel bandwidth is consumed by TCP connections that eventually abort (i.e., partial transfers). While the average throughput at the *network* layer exceeds 5 Mbps, the effective *user-level* goodput is about 2.2 Mbps.

Fig. 15 summarizes the `httperf` results for this experiment, including the throughput drop. Admission control would be required for HTTP requests to prevent a wireless Web server from experiencing this form of congestion collapse.

5.5. Experiment 5: miscellaneous

Additional experiments have studied more general Web workloads, including different HTTP request arrival processes [35], stochastically chosen HTTP response sizes [21], media streaming content [36], node mobility [14], and multi-hop wireless ad hoc networks [16]. These results are briefly summarized here.

In general, the measurements from these scenarios are qualitatively similar to the foregoing results, though much more complicated to analyze. Typical results show good user-level performance at low to moderate load, even for large transfer sizes and for media streaming applications. At high load or overload, performance degrades substantially. One experiment in [35] illustrates the impact of the HTTP request arrival process. When the arrival process is changed from Deterministic to Poisson to Self-Similar, the increasing variability in the arrival process triggers greater queueing fluctuations and a less distinct saturation point, but the behaviour under overload is fundamentally the same. Experiments varying transmit power and wireless channel conditions illustrate similar results [35].

Separate experiments with multi-hop wireless ad hoc networks [16] show that user-level TCP throughput drops dramatically with each additional

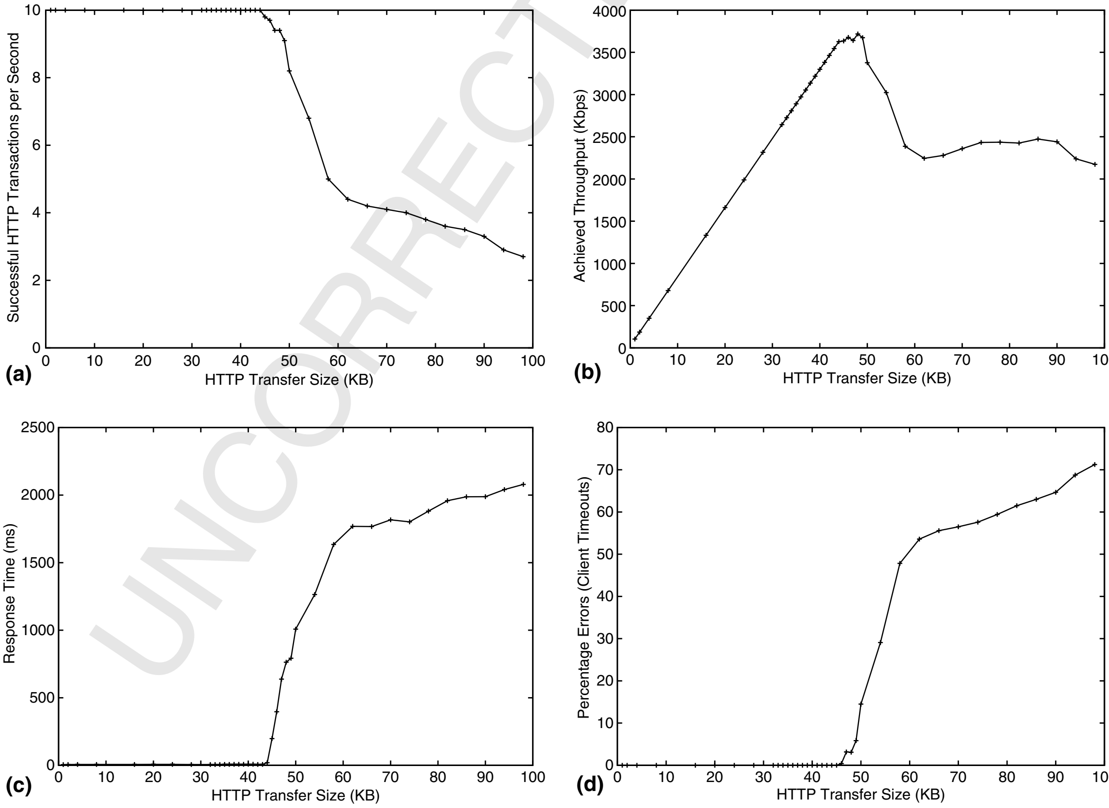

Fig. 15. `httperf` Performance results for Experiment 4 varying HTTP transfer size (one client, 10 req/s, non-persistent): (a) successful transactions, (b) achieved throughput, (c) response time and (d) error rate.

hop in the routing path. The drop in throughput occurs primarily because of the contention for the shared wireless channel at each routing hop, and the bidirectional nature of the network traffic flows. Additional factors are the overhead of the ad hoc routing protocol, and the non-deterministic behaviours of the MAC-layer protocols.

Additional experiments have considered wireless media streaming performance in a single-hop wireless ad hoc network [36]. Empirical measurement results show that the IEEE 802.11b wireless ad hoc network can support up to eight concurrent unicast MPEG-4 streams, each with 400 Kbps video and 128 Kbps audio. Adding one more stream destroys the quality of service for all clients, because of packet losses at the server's wireless network interface.

Node mobility in the ad hoc network can cause a "bad apple" phenomenon [14], wherein the aggregate network performance effectively degrades to that of the client with the worst wireless channel quality. In particular, one client with poor or transient wireless connectivity can degrade throughput and cause packet losses for all clients in the network [14,36]. The problem arises because of a transient Head of Line (HOL) blocking problem: when the packet at the front of the server's link-layer queue undergoes excessive retransmissions to the "bad apple" client, the queue fills and overflows, dropping packets for all clients.

Other researchers have confirmed the presence of these types of performance anomalies in (54 Mbps) IEEE 802.11g wireless networks as well [30]. These authors have considered TCP, UDP, and media streaming workloads in an infrastructure-based IEEE 802.11g WLAN, finding dramatic performance differences depending on the wireless channel quality for each of the clients.

Separate papers in our own research group have used simulation to evaluate the efficacy of novel MAC-layer protocols to solve these types of problems [15,37].

6. Summary and conclusions

This paper studies the performance of a wireless Web server in a short-lived wireless ad hoc network, such as a classroom area network. Application-layer and network-layer measurements are used to assess performance capabilities and limitations. In particular, the experiments focus on HTTP transaction rate and user-level throughput, as a function of request rate, number of clients, transfer size, and HTTP protocol features. Measurements were conducted on an IEEE 802.11b wireless LAN, using a wireless-enabled Apache Web server, several wireless client laptops, and a wireless network analyzer.

Our experiments show that wireless Web servers can provide 1 KB HTTP transaction rates of 110 connections per second for non-persistent HTTP and 320 HTTP transactions per second for persistent connections, with throughputs ranging from 1 to 3 Mbps. Several interesting performance problems are observed: a bottleneck at the wireless network interface for either the client or the server, depending on the workload; unfairness amongst clients due to packet losses during TCP connection handshaking; and a network thrashing problem for large HTTP transfers under overload. The use of persistent HTTP connections can overcome the inefficiencies of the IEEE 802.11b MAC protocol, tripling the effective HTTP transaction rate, while also improving fairness for clients accessing the wireless Web server.

Simulation models have been used to reproduce many of the behaviours observed in our experiments, and to predict performance for up to 100 clients [38]. Few of the performance problems identified in this paper (e.g., packet loss, phasing effects, unfairness, network thrashing) are seen in the classroom environment with human clients, because of the lower average workloads generated (i.e., due to think times, randomization, low request rates, and browser caching effects). Nevertheless, our study is valuable in identifying the performance problems that must be overcome to make the wireless Web server solution scale well (e.g., in undergraduate classrooms with 150 students, or sports venues with thousands of spectators).

Experiments with a 54 Mbps IEEE 802.11a wireless LAN remain for future work. We suspect that many of the performance problems observed in this paper apply equally well to 802.11a ad hoc networks.

Acknowledgements

Financial support for this research was provided by iCORE (Informatics Circle of Research Excellence) in the Province of Alberta, the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the Canada Foundation for Innovation (CFI). The authors are grateful to Martin Arlitt, Tianbo Kuang, and Nayden Markatchev for their technical support and contributions to this work.

References

[1] G. Bai, K. Oladosu, C. Williamson, Performance issues for wireless Web servers, in: Proceedings of International Workshop on Mobile and Wireless Ad Hoc Networking (MWAN), Las Vegas, NV, June 2004, pp. 59–65.

[2] W. Stevens, TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994.

[3] RFC 1945: Hypertext Transfer Protocol—HTTP/1.0. Available from: <http://www.ietf.org/rfc/rfc1945.txt>.

[4] RFC 2616: Hypertext Transfer Protocol—HTTP/1.1. Available from: <http://www.ietf.org/rfc/rfc2616.txt>.

[5] A. Balachandran, G. Voelker, P. Bahl, P. Rangan, Characterizing user behavior and network performance in a public wireless LAN, in: Proceedings of ACM SIGMETRICS, Marina Del Rey, CA, June 2002, pp. 195–205.

[6] B. Bennington, C. Bartel, Wireless Andrew: experience building a high speed, campus-wide wireless data network, in: Proceedings of ACM MOBICOM, Budapest, Hungary, September 1997, pp. 55–65.

[7] T. Hansen, P. Yalamanchili, H.-W. Braun, Wireless measurement and analysis on HPWREN, in: Proceedings of Passive and Active Measurement Workshop, Fort Collins, Co, 2002, pp. 222–229.

[8] R. Krashinsky, H. Balakrishnan, Minimizing energy for wireless Web access using bounded slowdown, in: Proceedings of ACM MOBICOM, Atlanta, GA, September 2002.

[9] B. Noble, M. Satyanarayanan, G. Nguyen, R. Katz, Trace based mobile network emulation, in: Proceedings of ACM SIGCOMM, Cannes, France, September 1997, pp. 51–61.

[10] H. Singh, P. Singh, Energy consumption of TCP Reno, TCP NewReno, and SACK in multihop wireless networks, in: Proceedings of ACM SIGMETRICS, Marina Del Rey, CA, June 2002, pp. 206–216.

[11] D. Tang, M. Baker, Analysis of a metropolitan-area wireless network, in: Proceedings of ACM MOBICOM, Seattle, WA, August 1999, pp. 13–23.

[12] D. Tang, M. Baker, Analysis of a local-area wireless network, in: Proceedings of ACM MOBICOM, Boston, MA, August 2000, pp. 1–10.

[13] D. Kotz, K. Essien, Analysis of a campus-wide wireless network, in: Proceedings of ACM MOBICOM, Atlanta, GA, September 2002.

[14] G. Bai, C. Williamson, The effects of mobility on wireless media streaming performance, in: Proceedings of Wireless Networks and Emerging Technologies (WNET), Banff, AB, Canada, July 2004, pp. 596–601.

[15] T. Kuang, C. Williamson, A bidirectional multi-channel MAC protocol for improving TCP performance on multihop wireless ad hoc networks, in: Proceedings of 7th ACM International Symposium on the Modeling and Simulation of Wireless and Mobile Systems (MSWiM), Venice, Italy, October 2004, pp. 301–310.

[16] A. Gupta, I. Wormsbecker, C. Williamson, Experimental evaluation of TCP performance on multihop wireless ad hoc networks, in: Proceedings of 12th IEEE Symposium on Modeling, Analysis, and Simulation of Computers and Telecommunication Systems (MASCOTS), Volendam, Netherlands, October 2004, pp. 3–11.

[17] D. Mosberger, T. Jin, Httperf—a tool for measuring Web server performance, ACM Performance Evaluation Review 26 (3) (1998) 31–37.

[18] Y. Hu, A. Nanda, Q. Yang, Measurement, analysis, and performance improvement of the apache Web server, International Journal of Computers and Their Applications 8 (4) (2001).

[19] E. Nahum, M. Rosu, S. Seshan, J. Almeida, The effects of wide-area conditions on WWW server performance, in: Proceedings of ACM SIGMETRICS Conference, Cambridge, MA, June 2001, pp. 257–267.

[20] Available from: <http://www.netcraft.com/survey>.

[21] C. Williamson, R. Simmonds, M. Arlitt, A case study of Web server benchmarking using parallel WAN emulation, Performance Evaluation 49 (1–4) (2002) 111–127.

[22] M. Arlitt, C. Williamson, Understanding Web server configuration issues, Software: Practice and Experience 34 (2) (2004) 163–186.

[23] J. Jun, P. Peddabachagari, M. Sichitiu, Theoretical maximum throughput of IEEE 802.11 and its applications, in: Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA'03), Cambridge, MA, April 2003, pp. 249–256.

[24] P. Barford, M. Crovella, Generating representative Web workloads for network and server performance evaluation, in: Proceedings of ACM SIGMETRICS Conference, Madison, WI, June 1998, pp. 151–160.

[25] Netperf: a network performance benchmark. Available from: <http://www.netperf.org>.

[26] T. Kuang, F. Xiao, C. Williamson, Wireless TCP performance problems: a case study, in: Proceedings of SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Montreal, PQ, Canada, July 2003, pp. 176–185.

[27] L. Eggert, J. Heidemann, J. Touch, Effects of ensemble-TCP, ACM Computer Communication Review 30 (1) (2000) 15–29.

[28] Y. Bejerano, S. Han, L. Li, Fairness and load balancing in wireless LANs using association control, in: Proceedings of ACM/IEEE MOBICOM'04', Philadelphia, PA, September 2004, pp. 315–329.

[29] M. Heusse, F. Rousseau, G. Berge-Dabbatel, A. Duda, Performance anomaly of 802.11b, in: Proceedings of IEEE INFOCOM'03, San Francisco, CA, March 2003, pp. 836–843.

[30] J. Gretarsson, F. Li, M. Li, A. Samant, H. Wu, M. Claypool, R. Kinicki, Performance analysis of the intertwined effects between network layers for 802.11g transmissions, Technical Report WPI-CS-TR-05-09, Computer Science Department, Worcester Polytechnic Institute, May 2005.

[31] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, M. Gerla, The impact of multihop wireless channel on TCP throughput and loss, in: Proceedings of IEEE INFOCOM, San Francisco, CA, March 2003.

[32] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, P. Sinha, Understanding TCP fairness over wireless LAN, in: Proceedings of IEEE INFOCOM, San Francisco, CA, March 2003.

[33] V. Padmanabhan, J. Mogul, Improving HTTP latency, Computer Networks and ISDN Systems 28 (1995) 25–35.

[34] S. Spero, Analysis of HTTP performance problems. Available from: <http://sunsite.unc.edu/mdma-release/http-prob.html>.

[35] K. Oladosu, Performance and robustness testing of wireless Web servers, M.Sc. Thesis, Department of Computer Science, University of Calgary, August 2003.

[36] X. Cao, G. Bai, C. Williamson, Media streaming performance in a portable wireless classroom network, in: Proceedings of IASTED European Conference on Internet Multimedia Systems and Applications, Grindelwald, Switzerland, February 2005, pp. 246–252.

[37] T. Kuang, Q. Wu, C. Williamson, MRMC: a multi-rate multi-channel MAC protocol for multi-radio wireless LANs, in: Proceedings of 1st International Workshop on Wireless Networks and Communication Systems (WiNCS), Philadelphia, PA, July 2005.

[38] G. Bai, C. Williamson, Simulation evaluation of wireless Web performance in an IEEE 802.11b classroom area network, in: Proceedings of 3rd IEEE International Workshop on Wireless Local Networks (WLN), Bonn, Germany, October 2003, pp. 663–672.

Guangwei Bai is currently an Associate Professor in the Department of Computer Science at the Nanjing University of Technology in China. He previously worked as a Research Associate in the Department of Computer Science at the University of Calgary in Canada. He received his B.Eng. and M.Eng. from the Xi'an Jiaotong University in China, both in Computer Engineering. He received his Ph.D. in Computer Science from the University of Hamburg in Germany. He worked at the GMD—German National Research Center for Information Technology, Germany, as a Research Scientist, from 1999 to 2001. Since 2001, he has been working at the University of Calgary, Canada, as a Research Associate. His research interests are in traffic measurement and modeling, workload characterization and performance analysis of the Internet, wireless networks, and multimedia communication systems.

Kehinde Oladosu is a Ph.D. student in the Department of Computer Science at the University of Western Ontario. He holds a B.Sc. (Honours) in Computer Engineering from Ladoke Akintola University of Technology, Nigeria, and an M.Sc. in Computer Science from the University of Calgary, Canada. His research interests include network traffic measurement, network simulation, Web server performance, grid computing, and high performance computing.

Carey Williamson is an iCORE Professor in the Department of Computer Science at the University of Calgary, specializing in Broadband Wireless Networks, Protocols, Applications, and Performance. He holds a B.Sc.(Honours) in Computer Science from the University of Saskatchewan, and a Ph.D. in Computer Science from Stanford University. His research interests include Internet protocols, wireless networks, network traffic measurement, network simulation, and Web server performance.