Feasibility Study of Mesh Networks for All-Wireless Offices

Jakob Eriksson*
UC Riverside
jeriksson@cs.ucr.edu

Sharad Agarwal
Microsoft Research
sagarwal@microsoft.com

Paramvir Bahl
Microsoft Research
bahl@microsoft.com

Jitendra Padhye
Microsoft Research
padhye@microsoft.com

ABSTRACT

There is a fair amount of evidence that mesh (static multihop wireless) networks are gaining popularity, both in the academic literature and in the commercial space. Nonetheless, none of the prior work has evaluated the feasibility of applications on mesh through the use of deployed networks and real user traffic. The state of the art is the use of deployed testbeds with synthetic traces consisting of random traffic patterns.

In this paper, we evaluate the feasibility of a mesh network for an all-wireless office using traces of office users and an actual 21-node multi-radio mesh testbed in an office area. Unlike previous mesh studies that have examined routing design in detail, we examine how different office mesh design choices impact the performance of user traffic. From our traces of 11 users spanning over a month, we identify 3 one hour trace periods with different characteristics and evaluate network performance for them. In addition, we consider different user-server placement, different wireless hardware, different wireless settings and different routing metrics.

We find that our captured traffic is significantly different from the synthetic workloads typically used in the prior work. Our trace capture and replay methodology allows us to directly quantify the feasibility of office meshes by measuring the additional delay experienced by individual transactions made by user applications. Performance on our mesh network depends on the routing metric chosen, the user-server placement and the traffic load period. The choice of wireless hardware and wireless settings has a significant impact on performance under heavy load and challenging placement. Ultimately we conclude that for our traces and deployed system, under most conditions, *all-wireless office meshes are feasible*. In most cases, individual transactions incur under 20ms of additional delay over the mesh network. We believe this is an acceptable delay for most applications where a wired network to every machine is not readily available. We argue that our results are scalable to a network of over 100 users.

Categories and Subject Descriptors: C.2.2 [Network Protocols]: Applications

General Terms: Measurement, Performance, Experimentation

*Jakob Eriksson interned at Microsoft Research during this work.

MobiSys'06, June 19–22, 2006, Uppsala, Sweden.

Keywords: Wireless office, mesh network

1. INTRODUCTION

Recently, static multi-hop wireless networks, or "mesh" networks, have attracted research [9, 8], commercial [2, 3] and standardization [1] interest. Unlike traditional ad-hoc wireless networks that have been motivated by mobile scenarios like the future battlefield, mesh networks have commercial applications such as community wireless access [19, 15]. In such networks, most of the nodes are either stationary or minimally mobile. We are motivated by the all-wireless office scenario [5]. In this application, offices with PCs are cooperatively interconnected by ad-hoc wireless links instead of Ethernet links, and few servers or proxies have wired connectivity to a corporate network or the Internet. This scenario is useful for small or low cost businesses and rapid deployment of short-term office space. Mesh networks are a natural solution for this space as they do not require the installation of any additional network equipment or wires, and potentially offer significant reduction in network administration (no access points or switches to maintain).

Despite significant activity in mesh networking, we are often met with considerable skepticism regarding the performance of mesh networks. None of the prior work has realistically answered the following question : *Are mesh networks feasible for real-world network applications ?* The majority of prior work has relied on simulation based evaluation, where typically the traffic patterns and node placement are synthetic. Such evaluation is inadequate given the complex nature of wireless propagation which is difficult to model and can have a drastic impact on the performance of the network. Recently, physical testbeds have been deployed and detailed wireless measurement studies have shown the impact of wireless propagation on performance [8]. Testbeds have also been employed to evaluate the relative performance of routing metrics [9]. Unfortunately, to the best of our knowledge, all the prior work in evaluating mesh networks has relied on synthetic traffic. The typical traffic pattern consists of running non-overlapping TCP bulk transfers between randomly selected pairs of nodes that last a few minutes [9, 8, 10].

We evaluate the feasibility of all-wireless office mesh networks. Not only do we employ an actual mesh network deployed in an office building, we also capture and evaluate traces of office users from the same building. Our mesh network consists of 21 nodes with multiple IEEE 802.11 radios (multiple radios offer significant benefits and are commonly considered [16, 11]). Our traffic is obtained at the socket layer from 11 users with PCs connected via Ethernet to the corporate network. We present a technique that allows us to replay this traffic on the mesh network in a realistic fashion. This is used for the purpose of evaluating the performance of several network configurations. We quantify the additional de-

lay experienced by individual network transactions made by user applications. This is a far more direct performance metric for our scenario than the typical metric of cross-sectional network throughput. Unlike previous mesh studies that have examined routing design in detail, we examine how different office mesh design choices impact user traffic. Specifically, we ask the following questions:

- Can we use a wireless mesh network to support an entire office? At what scale and performance penalty?
- How do various network design choices, such as node placement, hardware, wireless band and routing metrics impact application performance?

Toward these questions, we make the following contributions:

- We find that the captured traffic is significantly different from the synthetic workloads used in prior work. While the majority of traffic is TCP, not all sessions are bulk transfers nor HTTP traffic. There are periods of varying load, varying traffic flow sizes and varying flow overlap and they have a significant impact on network performance.
- An administrator can choose between several wireless network devices for deploying an office mesh. We evaluated two pairs of devices and found the choice had a significant impact on performance. In some cases there was a difference in median delay of 70-100 ms and difference in transfer success of 10%.
- Many such devices allow the administrator to pick one of multiple IEEE 802.11 bands. We find that with the same hardware, switching from IEEE 802.11a to IEEE 802.11g almost doubled median delay in some cases, even though both bands specify the same bandwidth.
- An administrator has a choice of several routing metrics from prior work. Of the five routing metrics we considered, two performed very poorly and often failed to transfer the entire load, while the remaining three offered very similar performance.
- In an all-wireless office, application servers and proxies connected to the wired network could be co-located or distributed across the office. We found that the placement of users and servers affected the results by changing typical path lengths and thereby affecting the delay experienced by applications.
- In the majority of configurations, the additional median delay experienced by network transactions due to the mesh network is under 20ms.

To the best of our knowledge, this paper presents the first testbed and trace based performance evaluation of an all-wireless office mesh network. In most cases, individual transactions incurred under 20ms of additional delay over the mesh network. We believe this is an acceptable delay for most applications [6, 20, 13] where a wired network to every machine is not readily available. However, it is possible that applications designed for wired, local-area conditions with strong latency constraints might suffer under these conditions. However, we believe such applications are few and we did not observe any in our traces. Ultimately we conclude that for our traces and deployed system, under most conditions, *all-wireless office meshes are feasible for a "typical office"*. We argue that our results are scalable to a network of over 100 users.

2. RELATED WORK

Prior work has developed several components for mesh networks including routing protocols, routing metrics and channel assignment schemes. Each of these require evaluation techniques for determining the effectiveness of the proposed system. As far as we know, all of this prior work has relied on synthetic traffic models, typically of random traffic patterns.

In [9, 10], Draves et. al. compare several routing metrics for mesh networks. For the evaluation, they measured the throughput achieved by 2-3 minute bulk-transfer TCP sessions, and did this both for single connection and multiple concurrent connection scenarios. This traffic model is completely synthetic and was not based on any observed network usage.

In [8], the ETX metric is proposed and compared to shortest hop for two routing protocols. They evaluated this on a per node pair basis, measuring the throughput achieved by a non-TCP CBR flow between source and destination, over a period of 30 seconds. In [7], the RoofNet network is presented and the performance of the network is evaluated. This evaluation again used one flow on a pair of nodes, at a time. These publications do not directly address the effects of multiple simultaneous flows, nor the relevance of the workload used.

In [18], the authors present Hyacinth, an architecture for multichannel wireless mesh networks. Their evaluation involves selecting 30 nodes at random to generate flows, directed at one of several gateways, throughout the simulation. The rate of each flow is chosen randomly between 0-3 Mbps, and each flow is said to represent an aggregate of user flows. In [22], multiple metrics are compared using NS-2 simulations. Again, this is done with CBR flows from randomly selected nodes, terminated at one out of several gateway nodes. While the issue of multiple simultaneous flows is addressed in these two publications, it is not clear that the traffic model with randomly picked source nodes sending traffic to gateways only is a realistic model. First, in most networks, traffic flows more heavily from gateways and servers to clients, than in the reverse direction. Second, in our captured traffic traces that we describe later, gateways are only one of several large contributors to traffic load. Third, it is unclear if CBR flow control and random selection of rate is representative of most traffic - in our traces, the vast majority of traffic is over TCP.

As shown by [21], user traffic loads can be far more complex than what we observe in synthetic traffic workloads from prior work. While the evaluation work done so far is highly relevant, we believe trace-based evaluation is the next step in accurately modeling and evaluating mesh networks. Packet level captures are frequently used to evaluate various performance aspects of queuing and routing protocols for the wired Internet. In our CARE methodology, we capture traffic from user PCs at the socket level to faithfully observe transport layer effects on the mesh network.

Recently, Campos and Jeffay introduced TMIX [14] for trace-based network performance evaluation, that captures packet level traces and reverse-engineers them to acquire a socket-level trace. Using PC clusters, they replay large volumes of traffic with high accuracy, over a single high-bandwidth wired link. While CARE and TMIX share several common goals, they are also distinctly different. In particular, CARE is targeted at entire mesh networks, whereas TMIX targets single-link, high-bandwidth wired connections. Also, CARE does not require any operating system modifications for accurate replay performance and does not require sophisticated techniques for reverse engineering socket-level semantics.

In [12], Liu et al present a model for "direct execution" of routing protocol implementations in simulation environments. Part of the functionality they describe is a means to record traffic, position and connectivity traces, and to replay these in their simulator. We approach the issue differently by capturing traffic from real users, and replaying this on an actual testbed.

With all this prior work and many others not cited for conciseness, the skepticism of mesh performance we receive is justified. Thus we pick a valid and concrete target for mesh networks and find that all-wireless offices on mesh networks are feasible.

3. CARE - CAPTURE, ANALYSIS, REPLAY AND EVALUATION

In order to determine the feasibility of an all-wireless mesh office, we need to evaluate the performance of real office traffic. However, since wireless mesh networks are not widely deployed today, it is not feasible for us to acquire traces from them. Instead, we capture user traffic on office PCs with wired Ethernet connectivity and replay them on a mesh testbed deployed in nearby offices. To that end, we developed CARE, which is a tool-set for capturing user traffic, analyzing it, replaying it on another set of machines and evaluating the outcome.

3.1 Capture - Socket Level Traffic Capture

It is not sufficient to capture packets leaving and entering office PCs and replay them on a mesh testbed. Transport protocols such as TCP adapt to the available network conditions and this can dramatically influence the rate at which packets are sent and received. Since the corporate Ethernet network has vastly different properties (bandwidth, delay and loss) than a multi-hop wireless network, it is not representative to replay every packet at the same time it was observed on the wire.

Instead, we capture traffic on office PCs before they reach the transport layer, and then replay them on the multi-hop wireless testbed just before the transport layer. Specifically, we capture socket calls made by the application layer. Socket level traces differ from packet level traces in that they are independent of lower layer issues such as maximum transmission units, transmission errors, acknowledgment packets, packet drops, packet reordering etc. Any transport layer behavior on Ethernet will be masked by the capture and any transport behavior on the mesh network will faithfully be experienced by the replay.

It is possible that application or user behavior above the transport layer might change depending on the network conditions. In our methodology, we are unable to account for this factor. However, we believe that if the delay experienced by user traffic on the mesh network is not significantly larger than that of the wired network, the behavior above the transport layer will not change. Thus we expect that any results with low additional delay on the mesh network will be valid.

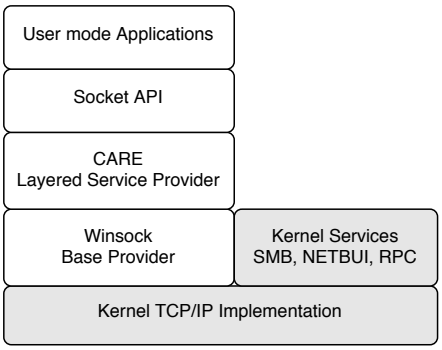

Figure 1: Layered Service Provider in Windows XP Stack

To capture socket level traces, we make use of the Layered Service Provider (LSP) interface in the Windows XP network stack. An LSP is a loadable library that can act as an indirection layer between all applications on a machine and the kernel TCP/IP implementation. This is shown in Figure 1. By loading a custom-designed LSP, we can augment the network stack with logging functionality. It allows us to intercept each socket call (such as a $connect$, $send$, $recv$, $close$) from the application, and record details to disk. These details include a 64 bit time-stamp, socket identifier, IP address, port, protocol (in the case of connect) and bytes received or sent. With our logging, the LSP code totals 18003 lines of C++ code. It compiles down to a 52 KB DLL and uses additional DLLs and executables to be inserted into the stack.

Alternative approaches to socket level capture include instrumenting application binaries to record all socket calls and instrumenting socket libraries to do the same. We used the LSP interface because it was designed to allow exactly this functionality and has minimal impact on the user being monitored. Installing our LSP code requires a reboot, and has no noticeable effect on network connectivity, especially since it only records a small amount of information on every socket call.

Instead of socket level capture, it may be possible to capture packet-level traces and reverse-engineer an approximate socket level trace from this. This would require modeling TCP behavior such as accounting for any packet loss and maintaining accurate estimates of the window sizes to infer what delays are due to applications not sending data versus TCP windows being full. We believe this requires significantly more effort and the reverse-engineering can potentially introduce errors.

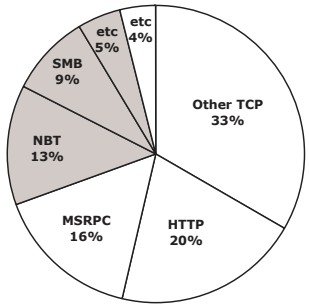

Figure 2: Traffic Volume Distribution on Sample User Machine (gray slices not captured)

There is a drawback to LSP monitoring on the Windows XP platform. As shown in Figure 1, certain protocols such as SMB, RPC, NetBUI/NBT, LDAP and ICMP are implemented in the kernel and are not above the LSP layer. However, for Remote Procedure Call (RPC), there exists a setting in the Windows Registry that forces RPCs to run in user mode, making it visible to the LSP. This affects all RPC calls, such as MS Exchange. We enabled this setting on all the office user machines from which we captured traffic.

Figure 2 shows in gray the fraction of traffic that our LSP did not capture, on a sample user machine. The NetBUI/NBT and SMB file transfer protocol account for about 22% of traffic. While capturing this LSP traffic, we also captured packet traces at the kernel level in parallel, and examined the missing 22% of traffic. We found that the vast majority of this traffic originated from intrusion detection systems (IDS) on the corporate network. They periodically scan all machines for malicious files and registry entries. None of this IDS traffic is captured by the LSP, as it uses SMB and NetBUI which are implemented in the kernel. However, as this is rather specific to the corporate network we examined, we believe that excluding this traffic from the data set may actually improve the relevancy of our results. While our LSP would miss any actual user file sharing on SMB, we did not find any instances of this in the sample user machines. Finally, UDP accounts for less than 0.04% of the traffic we captured. Given this insignificant amount, we do not consider UDP when replaying our traces.

3.2 Analysis - Preparing Traces for Replay

After installing the LSP capture code on a sample set of user machines, we obtain several files from each machine. Each file represents a unique instance of the Winsock DLL, typically an instance of an application such Internet Explorer. We post-process these traces and break them down into *sessions* and *transactions*, as shown in Figure 3. A session consists of a sequence of transactions. Each session can have at most one transaction in progress at any time. Transactions start at the time specified in the trace, subject to the completion of previous transactions. Each transaction consists of one $send$ operation, and zero or more $receive$ operations.

For example, a user goes to the URL `http://www.cnn.com/index.html`. The session would represent all the network traffic involved in browsing to this site. The connect would specify the IP address of the web site and port 80. Each transaction would represent a specific object. For instance, the first one could be index.html and the second could be an image that is referred to in the index.html. Each transaction is composed of a request and a response - the request could be the HTTP GET command for index.html and the response would be all the data bytes associated with that object. Pipelining support in HTTP/1.1 is not problematic here. HTTP pipelining bundles multiple requests into one, and the responses are received sequentially. This type of file transfer is correctly modeled as a single transaction. Recall that our traces are captured at the socket layer, and thus no TCP effects are captured here, as intended. Some protocols, such as Gnutella, use packetized traffic over TCP. In this case, each request and response would represent individual packets. However, such protocols did not constitute a noticeable fraction of our traffic.

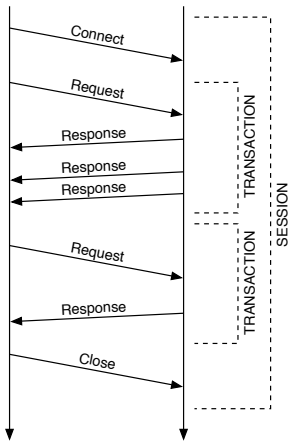

Figure 3: Transactions are Request/Response Exchanges in a Session

For ease of replay, we create a file for each session and create a schedule file that dictates when each socket session begins. We also map each end point to a machine on our mesh testbed for replay. CARE supports arbitrary mappings between captured destination host names and mesh nodes, as well as between captured traces and mesh nodes. As described in the next section, we use a variety of different placements in our evaluation.

3.3 Replay - Playback of a Trace on a Mesh

To understand the performance that applications would experience from a wireless mesh, we need to replay the captured traces on our operational testbed. We have developed client and server replay software that will run on the testbed. Each client software emulates one user, and it initiates requests and awaits responses from the servers on other machines. Depending on the user that the client is emulating, it is be given the appropriate session files. The server does not need any traces as it is instructed by the client for each transaction. Our client consists of 537 lines of Perl code and the server is 260 lines.

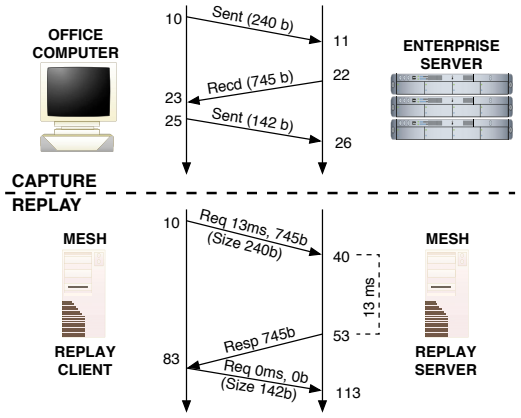

Figure 4: Replay of Captured Trace

Strict timing is followed for the start of sessions. That is, if the original user trace had only two sessions, session 1 beginning at time t_1 and session 2 beginning at time t_2, then in the replay, the client will start session 1 at time 0 and then session 2 at time (t_2 - t_1). Once a session has started, each client is responsible for sequentially executing the transactions within it. For each transaction, it sends a request to the server, containing a list of transmissions that the server is expected to execute, specifying time and size for each operation. The request is padded with empty space to ensure that the request has the size specified in the trace [1].

Timing of transactions within a session is illustrated in Figure 4, where there is one session and two transactions inside it. Here, the recorded trace will contain three entries:

- time 10, sent 240 B to server
- time 23, received 745 B from server
- time 25, sent 142 B to server

In the replay, the client will wait until time 10, then send a 240 B packet to the server with an embedded request. The server will wait the requested 13 ms before sending the requested 745 B response. Since the client received the response after the time at which the 142 B send was to occur, it immediately sends the 142 B packet, requesting nothing in return. Thus, session start times are always preserved, and transaction start times are honored if possible.

We designed our replay mechanism to preserve, to the largest extent possible, the intervals between the start of transactions. Alternatively, we could have preserved the time between the end of one transaction, and the start of another. In practical terms, our chosen approach tends to err on the side of caution, as the shorter inter-transaction intervals tend to generate a somewhat higher load than an actual application would. The alternative tends to err on the side of lower load, and does not appear suitable for performance evaluation purposes.

[1] In some cases, this request will be larger than what the trace specifies, due to the size of the instructions supplied in the request. However, we have verified that this rarely happens as our instructions within the request are rather small.

3.4 Evaluation - Replay Performance

Network performance evaluation has typically been done in terms of throughput or end-to-end delay. These are valuable performance metrics that probe the boundaries of network capacity. However, in this paper we want to determine the performance penalty incurred, on actual usage, by the use of a multi-hop wireless mesh network instead of a wired LAN. The performance metric we use for this is *transaction time*, specifically the increase in transaction time incurred by the use of a wireless network. We discuss this choice of metric below.

During replay, each client stores to disk a record of all operations with 64-bit timestamps to aid in evaluation. Each transaction is recorded as *transaction duration in trace, replay duration, transaction size*. Here, the replay duration time is strictly larger than the transaction duration in the trace, since any wait times in the trace between requests and responses are replayed as wait times on the server. Transactions that contain no response are not recorded in the results, as the response time of such a transaction is undefined.

Note that the transactions in the user traces as recorded incorporate four main delay components between a request and a response : round-trip time from the real user machine to the edge of the corporate network, round-trip time from the corporate network to the destination on the Internet (if the destination is remote), transmission time related to the size of data and processing time on the remote server. In our replay, we consider this entire delay as a single wait time - when a client sends a request to the server, this wait time is embedded in the request. Upon receiving the request, the server will wait for the requested time and then send the requested amount of data. Thus the final recorded delays will have both the original delay and the additional delay due to the mesh network. We believe this is an accurate portrayal of a wireless office, because in that scenario there will also be the mesh delay, delay on the corporate network, delay on the Internet and delay on the server. The delay incurred on the corporate network is typically under 1 ms, while as we will show, the delay on the mesh network is typically an order of magnitude larger.

For our trace-based performance evaluation, we use transaction time as the evaluation metric. In prior work [10, 8], overall network throughput has been the main evaluation metric. This does not directly apply to trace based evaluation, as it is relatively rare for actual LANs to be utilized to the maximum for extended periods of time. For example, assume that in a one-hour trace, we capture 100 MB of transfers. Even in networks with 1 Mbps channels, most routing protocols will manage to transfer 100 MB in one hour, and thus will all achieve a similar total throughput. Session completion time is another potential evaluation metric. However, as many of our recorded sessions include large wait periods (several minutes), this would not capture the performance of interactive sessions. We believe that transaction time is a suitable evaluation metric, striking a balance between throughput and end-to-end delay.

Since the wait times in transactions between requests and responses already includes server delay, we need to be certain that our replay server does not incur significant additional delay. Our replay server is multi-threaded and uses a thread-pool to avoid fork overhead. Nonetheless, we evaluate the performance of our replay mechanism using the wired network for all communication. To test the extreme limits, we employed a single client machine and a single server machine, replaying traffic from all the captured machines simultaneously. The real experiments in the next section spread this load across 22 machines.

Figure 5 shows the results of the wired replay experiment. The median additional delay incurred by all transactions was 1.1 ms, and the average 1.7 ms. Part of this delay is due to the Ethernet delay, and part of it is due to processing and queuing delay on the client and server machines. There are 12 transactions (out of approximately 13000) that took between 32 and 256 ms of additional delay to complete. These are delays specific to our corporate Ethernet setup and do not apply to the wireless mesh experiments. The same experiment between a client and server on the same machine over the loop-back interface did not experience the large delays. We conclude that replay typically adds about 2ms of additional delay to our results, and given the delay incurred by the mesh network is about an order of magnitude larger, we do not consider this to be problematic.

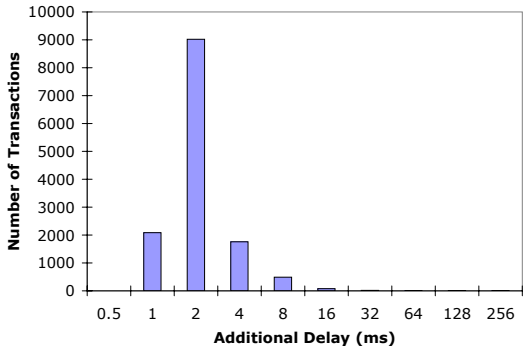

Figure 5: Replay Performance Between 2 Machines over Ethernet (rounding up to next delay bin)

4. EXPERIMENTAL SETUP

We believe our evaluation of an office mesh network has high fidelity and realism because of three aspects of our experimental setup. First, we describe the traffic we captured from typical users in our office environment. Second, we describe our operational mesh system that we replay these real user traces on. We consider a typical office mesh network to have nodes scattered across an entire floor of 100 offices with most routes of length of 2-5 hops. Third, we describe the mesh routing software on our testbed which allows us to evaluate the performance of the complete system.

4.1 Traces of Office Machines

As described in the previous section, we use our LSP to capture traffic traces for our experiments. We installed our LSP on 11 desktop computers of 11 users, each of which was a primary office PC for the user in question. Typically each machine had about 1GB of main memory, dual 3GHz to 4GHz Pentium IV processors, and an Ethernet connection to one of two LANs connected to our corporate enterprise network at Microsoft. The users were scattered across the same floor as our testbed in Figure 10, which is described in the next subsection. They were a mix of graduate student summer interns and full-time research employees. The traffic to and from each machine was captured for about a month.

Capture Period	Aug. 2005 to Sep. 2005
Capture Hosts	11
Unique IP Addresses	1490
Total Traffic	16.8 GB
Average Traffic per IP	11900 KB
Median Traffic per IP	34 KB

Table 1: Characteristics of Captured Traffic

In Table 1 we show some broad characteristics of the user traffic that we captured. While there is a large number of source and

destination IP addresses in the traffic leaving or entering the 11 capture hosts, the vast majority of these addresses contribute a small amount of load, with a few dozen hosts representing the majority of traffic. To understand this phenomenon further, we present Figure 6. Through knowledge of the internal network and servers deployed on our corporate network, we are able to identify the service provided by most of the IP addresses in our traces. Each bar represents an application class, and shows the number of bytes sent to and received from our capture hosts. The first bar represents traffic associated with Microsoft Exchange servers, which host email, calendars, address books and public folders (discussion bulletin boards). The Domain Controller traffic includes log-in, and various authentication protocols.

Machines on the Microsoft corporate network connect to the Internet via a number of application and socket level proxies, which are typically Microsoft ISA (Internet Security and Acceleration) servers. Web browsers, such as Internet Explorer, are automatically configured to use the application level HTTP proxy. Other applications, such as secure shell (ssh) are forced to use the socket-level proxy by a firewall client that operates below our LSP monitor. All application level proxy traffic (typically only HTTP traffic) is represented by the "Proxies" bar, while the socket level proxy traffic corresponds to the "Other External" bar.

Source Depot is a code repository and version control system used internally by Microsoft, and can be thought of in the same way as CVS. The "Other Internal" bar represents traffic to internal hosts other than Source Depot servers, Exchange servers and Domain Controllers.

We captured traffic to and from each machine for about a month. Given the long duration of our capture and the large number of evaluation parameters, it is not feasible for us to consider the entire capture for replay. Instead, we pick three traffic periods of one hour each with different load characteristics. However, due to interns leaving and new employees joining during the capture period, we do not have captures from all machines with overlapping dates when each person was actively using their machine. To create more realistic data sets containing all captured machines, we use traces from each user from the same day of the week and time of day, but not necessarily the same week. The three traffic load periods are described in Table 2. We specifically targeted time periods with three different load characteristics to show how the mesh network performs in each.

Name	Day / Time	Load (MB)	Session Count	Transaction Count
Heavy	Fri 18:00-19:00	587.51	306	9600
Medium	Tue 10:00-11:00	83.27	969	38757
Light	Tue 13:00-14:00	19.72	415	2970

Table 2: Traffic Periods Employed

While each period has different total load, within each period the sizes of transactions also vary. In Figure 7, we show the CDF of transaction sizes for each of the three load periods. Notice that while for the heavy period most transactions are between 1 KB and 1 MB in size, for the medium period it is between 100 B and 10 KB. Figures 8 and 9 show the distribution of session and transaction concurrency for every millisecond. There are many periods during which no transactions are ongoing - this is another reason why we use transaction level delay as the metric instead of overall throughput. Of the remaining time periods, it is common for more than one transaction to be active at the same time, more so for the medium and light periods than the heavy period. The fidelity that arises from using real traffic is not present in the synthetic traces

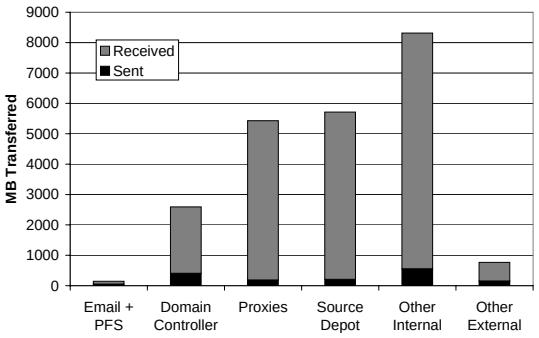

Figure 6: Distribution of Traffic by Type

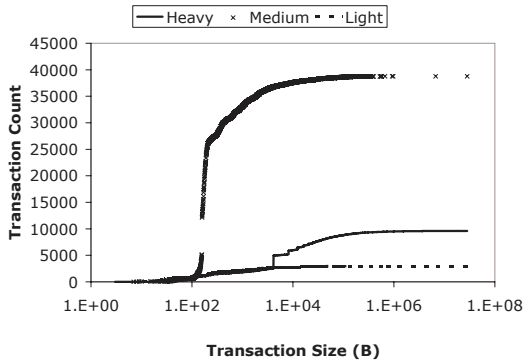

Figure 7: CDF of Transaction Sizes in 3 Load Periods

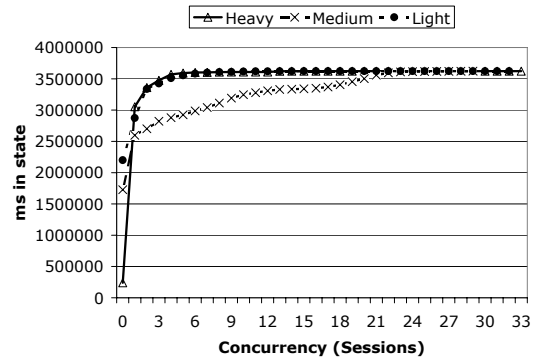

Figure 8: CDF of Session Concurrency in 3 Load Periods

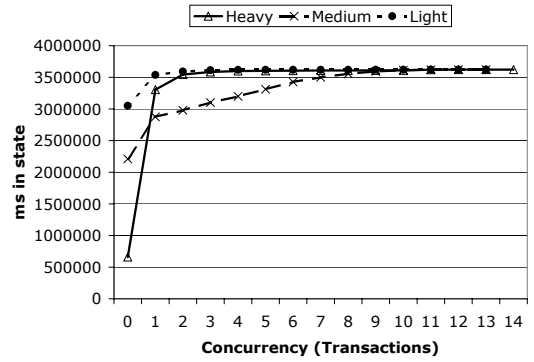

Figure 9: CDF of Transaction Concurrency in 3 Load Periods

typically used in prior work - often there is no concurrency and the goal is to maximize throughput and so the transaction sizes are the MTU.

4.2 Office Testbed

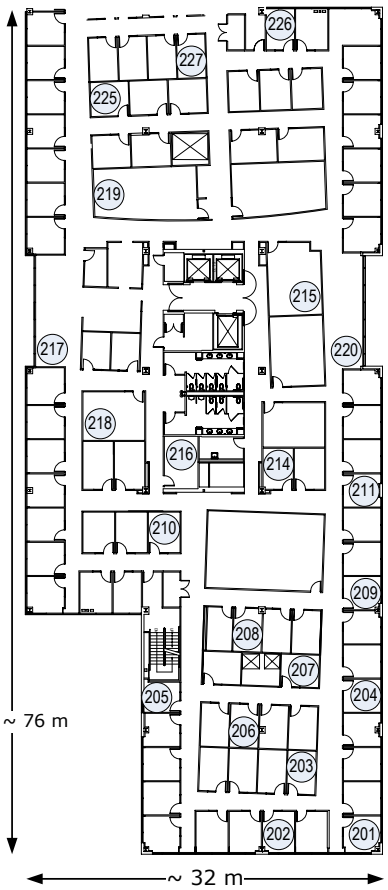

Figure 10: Office Testbed Deployment

The results presented in this paper are from replaying the captured traffic on a 21-node wireless testbed shown in Figure 10. While the size of our testbed is limited, we discuss how our results apply to much larger scenarios in Section 5.9. Our testbed is located on one floor of a fairly typical office building, with the nodes placed in offices, conference rooms and labs. Unlike wireless-friendly cubicle environments, our building has rooms with floor-to-ceiling walls and solid wood doors. The nodes are in fixed locations and did not move during the experiments reported here. While our node deployment is influenced by building design and office layout, we consider multiple client-server placement scenarios.

The nodes are all Hewlett-Packard model d530 SFF PCs. Each of these machines has a 2.66GHz Intel Pentium 4 processor with 512MB of memory. They all run Microsoft Windows XP. The TCP stack included with XP supports the SACK option by default, and we left it enabled. All of our experiments were conducted over IPv4 using statically assigned addresses.

Each node has three IEEE 802.11 radios. One of them is a NetGear WG 111U device that is connected via USB 2.0. The second is a Proxim ORiNOCO ComboCard Gold connected to the PC via a Psism PCD-TP-202CS PCI-to-Cardbus adapter card. The third is also a PCI card connected by another Psism adapter and is either a

Name	WG	WAG/ WAB	Proxim	Xmit Power	RTS
A	a 56	a 36	off	100%	Off
B	a 56	a 36	off	100%	On
C	a 56	g 10	off	100%	Off
D	off	g 10	a 56	100%	Off
E	off	g 10	a 56	50%	Off
F	off	g 10	a 56	12.5%	Off

Table 3: Testbed Configurations (a,g are IEEE 802.11 bands and 10,36,56 are channels)

NetGear WAG 511 or a NetGear WAB 501 card. All these models of 802.11 devices are multi-band radios. In each of the experiments we performed only two devices were enabled at the same time. The NetGear WAG or WAB card was always enabled, and either the NetGear WG or the ORiNOCO was enabled as noted.

We configured each device for ad-hoc mode, but we also considered several different parameter settings in our experiments, as shown in Table 3. We varied the IEEE 802.11 frequency band (a, b or g) and associated channel number, the transmit power level and the RTS/CTS threshold. When the RTS/CTS threshold is at the default of 2346 bytes, no RTS/CTS packets are generated as the MTU is 1500 bytes. To turn it on we set the threshold to 100 bytes. We left the remaining parameters at the default setting for the radios. In particular, the cards all perform auto-rate selection. In future work, we plan to explore the impact of rate control. While there are some 802.11a and 802.11b access points in our building, we verified that they had no significant impact on our results by sniffing for traffic and comparing night time results with day time results.

4.3 MCL : Mesh Connectivity Layer

We use the LQSR protocol as implemented in the ad-hoc routing framework called the Mesh Connectivity Layer (MCL) [10]. The MCL driver is available both in source code and binary form to the research community [2]. Architecturally, MCL is a loadable Windows driver. It implements a virtual network adapter - essentially an interposition layer between layer 2 (the link layer) and layer 3 (the network layer). To higher-layer software, the ad-hoc network appears to be just another Ethernet link, albeit a virtual link. To lower-layer software, MCL appears to be just another protocol running over the physical link.

The MCL adapter routes packets using LQSR. The LQSR implementation in MCL is derived from DSR. It includes all the basic DSR functionality, including Route Discovery (Route Request and Route Reply messages) and Route Maintenance (Route Error messages). LQSR uses a link cache instead of a route cache, so fundamentally it is a link-state routing protocol. MCL has a variety of link-quality metrics for LQSR. In this paper, we consider the following five metrics:

- **Hop Count (HOP).** This is the most basic metric. The cost of a path is defined as the total number of links in it. Hop count does not require any active measurements to compute the metric, other than periodic broadcasts to determine adjacency.
- **Per-hop Round Trip Time (RTT).** RTT is based on measuring the round trip delay experienced by unicast probes between neighboring nodes [4]. The cost of a path is defined as the sum of per-hop delays along the path. RTT uses unicast probes to each individual neighbor to determine the round-trip time.
- **Per-hop Packet Pair Delay (PktPair).** PktPair is based on

[2]http://research.microsoft.com/netres/software.aspx

measuring the delay between a pair of back-to-back probes to a neighboring node, in order to determine the idleness of the channel. This estimate is translated into the available bandwidth between the two nodes, and the cost of a path is defined as the minimum bandwidth across all the links in it. PktPair uses unicast probes to each individual neighbor.

- **Expected Transmission Count (ETX).** The ETX metric [8] measures the expected number of transmissions, including retransmissions, needed to send a unicast packet across a link. ETX starts with measurements of the underlying packet loss probability in both the forward and reverse directions (through the use of one-hop broadcast probe packets) and then calculates the expected number of transmissions.
- **Weighted Cumulative Expected Transmission Time (WCETT).** This is the only metric in our comparison that explicitly takes channel diversity into account [10]. WCETT combines ETT, which estimates the transmission time on a given link, with a measure of the channel diversity on a path, to improve performance in where nodes have multiple interfaces.

4.4 Mapping of Office Users to Office Testbed

In addition to varying the wireless hardware, transmit power level, band and channel, RTS/CTS and the time period from the trace, we also vary the mapping of captured users and servers to machines in our testbed. Recall that we have 11 captured users and 21 nodes in our testbed. We use some of the remaining nodes to represent servers for the application classes shown in Figure 6. It is quite likely that in an all-wireless office, some servers may be deployed specifically for the office users, while others may be accessible in the corporate network through the few nodes that have wired connectivity. We consider scenarios where the wireless network has its own Domain Controllers, Source Depot servers, email server, public folders. All other traffic, including traffic to the Internet proxies and to other corporate machines will go to the machines with wired connections.

When we replay the captured traffic, each capture machine IP address is replaced with the corresponding assigned testbed machine's MCL IP address. Each application server's IP address is replaced with the testbed machine that we assign to the server. All other IP addresses are assigned to a default machine on the testbed.

In Table 4, we show the three different placement scenarios we consider. Central placement represents the case where the servers are all in the middle of the testbed and the users are scattered across the network. Distant placement has the servers in the two ends of the network. In the case where there are two servers for each application class, we map individual application servers to one of the two testbed machines at random.

We also consider a third "extreme" scenario. It assumes there are only two machines in the wireless network with wired connectivity, and all traffic goes to either one of them. Further, we examine what would happen if all the remaining 19 nodes represented users. Since we have only 11 captured users, we replicate 8 of them with a one hour time-shift onto 8 more testbed machines. We use this scenario to test the limit of mesh network performance.

5. RESULTS

We now present the results of our feasibility study of an all-wireless office. First, we benchmark our testbed and the various testbed configurations from Table 3 with synthetic traffic patterns as in prior work. Second, we present results from repeated tests - we show that the performance of the testbed is fairly stable and the results are repeatable. In the subsequent sections we examine the performance of office mesh when configured with different routing metrics, load periods, user and server placement, and network configurations including RTS/CTS, different wireless hardware, transmit power levels and channels.

	Central	Distant	Extreme
User 01	203	203	203
User 02	205	205	205
User 03	206	206	206
User 04	208	208	208,207
User 05	209	209	209,210
User 06	211	211	211,214
User 07	226	215	215,216
User 08	225	217	217,202
User 09	218	218	218,204
User 10	227	219	219,225
User 11	204	220	220,226
Domain Controller 1	214	204	201
Domain Controller 2	215	226	227
Source Depot 1	217	227	201
Source Depot 2	217	227	227
Email	220	202	201
PFS	220	202	227
Proxy 1	219	201	201
Proxy 2	216	225	227
Default	216	225	227

Table 4: User-Server Placement Scenarios (node locations in Figure 10)

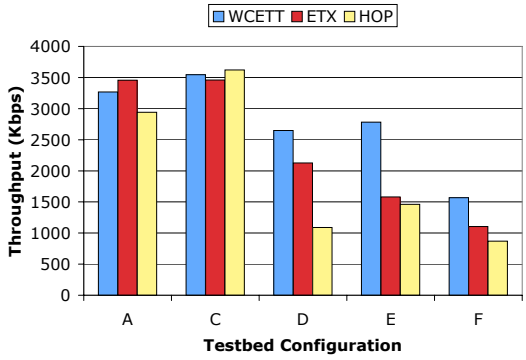

Figure 11: Median Throughput using Synthetic Traffic

5.1 Mesh Performance with Synthetic Traffic

Prior work [10] employed a similar testbed with random traffic patterns to evaluate the relative performance of three routing metrics : WCETT, ETX and HOP. We repeat those experiments for the different testbed configurations in Table 3. This allows us to compare our testbed's performance with prior work and provide a rough estimate of the throughput of the testbed. As in prior work, this synthetic trace is generated as follows : every 3 minutes, a sender node and receiver node are selected at random; a TCP flow transfers as many bytes as possible in 2 minutes followed by 1 minute of silence; this is repeated 100 times, making a total of 5 hours.

Figure 11 shows the median throughput achieved across the 100 flows for each of the 3 metrics and for each of the testbed configurations. This graph can be compared to the "two radios" bars in Figure 5 of [10]. We see that testbed configuration E gives the closest relative and absolute performance to that of the prior work. Interestingly, configurations A and C disagree with prior work - the performance of the metrics is very similar. To understand this issue, we present Figure 12 which shows the median route length of the 100 transfers in each case. The performance of the metrics diverge

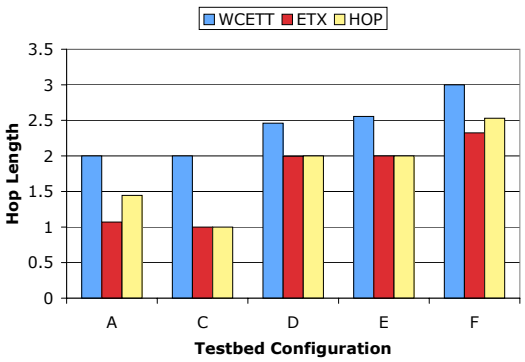

Figure 12: Median Route Length using Synthetic Traffic

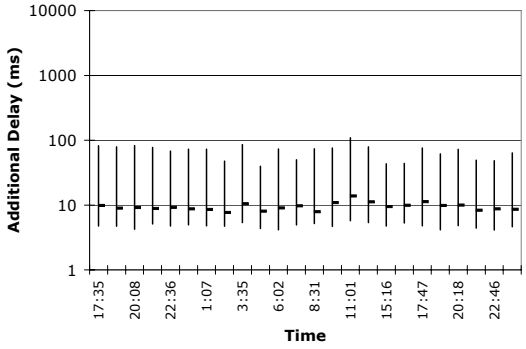

Figure 13: Performance Variation Across Repeated Runs of Medium Traffic Period, Distant Placement, WCETT Metric, Testbed Configuration A

when the route length exceeds 2.5. This is understandable because WCETT optimizes for self-interference of flows. If most routes are 1 or 2 hops long, then even random selection of which channel each hop will traverse will provide sufficient channel diversity. Beyond 2 hops, intelligent choice of channel diversity can improve performance. Note that as explained in [10], even though WCETT sometimes picks longer paths, because it takes link bandwidth into account, these longer paths can provide higher throughput.

5.2 Mesh Stability in Office Environment

Before we present the main results, it is important to establish the stability of the testbed. During the course of a day there will be variations in the office environment, such as due to occupants moving around and closing or opening doors. There will also be more direct changes in the RF environment, such as due to microwave oven use, vacuum cleaners, cordless phones and other IEEE 802.11 nodes. Individual transactions can experience additional loss or delay due to temporary changes in link performance. However, for the majority of transactions, we do not expect significant variation across an entire day. Thus we want to examine how the median delay experienced by transactions varies across repeated experiments.

Figure 13 shows the results of repeatedly running a specific traffic period, user and server placement, metric and testbed configuration. We have repeated these experiments with other settings and have found similar results, and thus we only present one graph for conciseness. As shown in Table 2, the medium load traffic period has 969 sessions, each of which has multiple transactions. We calculate the additional delay incurred by each transaction on the mesh and present the median value, with error bars showing the 5th and 95th percentiles. The figure shows that the median value has relatively little variation across the 24 runs - the minimum median value is 7.67 ms, the maximum is 13.81 ms and the standard deviation is 14%. Combining this variability with the overhead of replay in Figure 5, we conclude that the median additional delay performance reported by the testbed within 24 hours is stable within an error of about 10 ms. The 95th percentile is stable within 70 ms and the 5th percentile error is about 5 ms.

5.3 Performance across Different Traffic Load Periods

We now examine what performance a mesh network would offer to office users under the three different load periods we identified in Table 2. We consider the five different metrics described earlier to further examine if the choice of metric plays a significant role.

Figures 14, 15, and 16 show the additional transaction delay incurred in each of the three load scenarios. Each bar represents one of the five metrics, with the top and bottom numbers giving the 95th and the 5th percentile respectively, and the tab in the middle indicates the median additional delay. Recall from the previous experiments that differences under 10 ms in the median could be due to the natural variation in an office environment.

In the light and medium traffic periods, the additional delay incurred by most transactions is quite small - typically under 10ms. Recall that a session would for instance be a web browser going to a particular site, and it would be made of multiple transactions - each image could represent a transaction. We believe that an additional 20ms on top of the delay incurred in traversing the Internet, proxies and the server is tolerable for a user on a wireless network. In the heavy load period, while WCETT, ETX and HOP do provide delay under 20ms, PKTPAIR and RTT perform significantly worse. Further, PKTPAIR appears to be the worse metric since it has the highest 95th percentile in all cases.

We do not examine this difference in metric performance in more detail. It is not the goal of this paper to do a detailed study of metrics. Instead, since a mesh operator today is faced with a choice of metrics, we want to evaluate if for the office mesh scenario the choice matters. We find that 2 metrics suffer significantly, while the remaining 3 provide roughly equivalent performance under multiple load conditions. Note that however, the synthetic traces in Figure 11 and prior work show that WCETT provides significant improvements over HOP and ETX. One of the main differences between the synthetic trace and our captured traces is concurrency - Figure 9 shows that many transactions overlap, whereas the synthetic traces only have one active flow at any time. In such a scenario, it is possible that cross traffic interference dominates self interference (WCETT optimizes for the latter) and that cross traffic interference varies faster than what the routing metric can adapt to.

5.4 Performance across Different Traffic Load Periods and User-Server Placement

An office mesh operator also needs to decide where to place servers and wired gateways in relation to users. In the previous three graphs, we placed the servers in the middle of the network, thereby providing relatively short paths to most users. We now consider the distant placement, where the servers are at the two extreme ends of the network, and some users will have relatively short paths to some servers, while others will have long ones. These results are given in figures 17, 18 and 19.

While the median delay for the light and medium traffic periods is similar to central placement, the 95th percentile is higher. So while a small set of transactions suffer more, for the majority

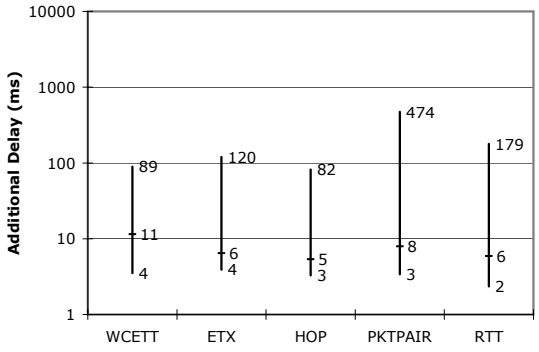

Figure 14: Performance over Light Traffic Period, Central Placement, Testbed Configuration A

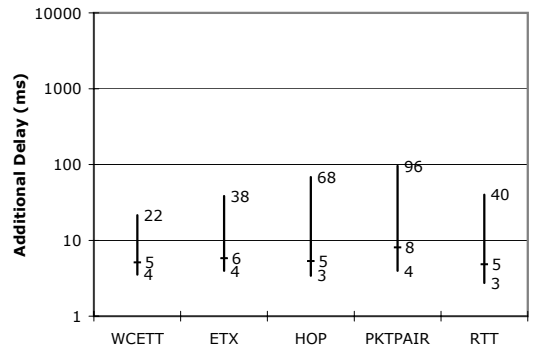

Figure 15: Performance over Medium Traffic Period, Central Placement, Testbed Configuration A

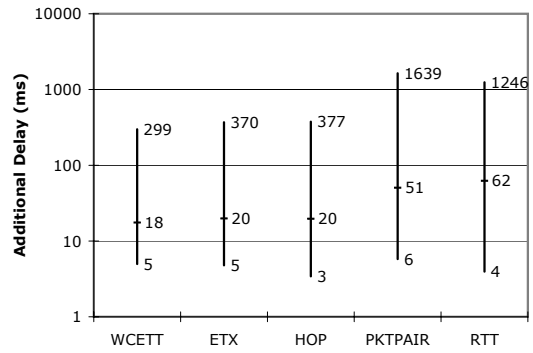

Figure 16: Performance over Heavy Traffic Period, Central Placement, Testbed Configuration A

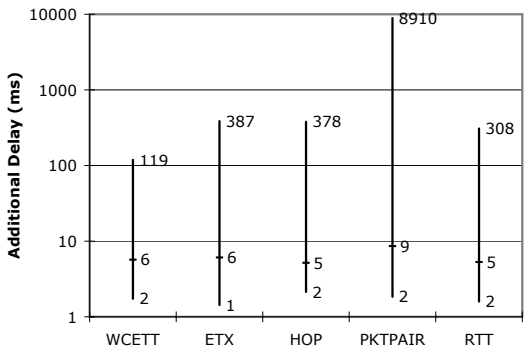

Figure 17: Performance over Light Traffic Period, Distant Placement, Testbed Configuration A

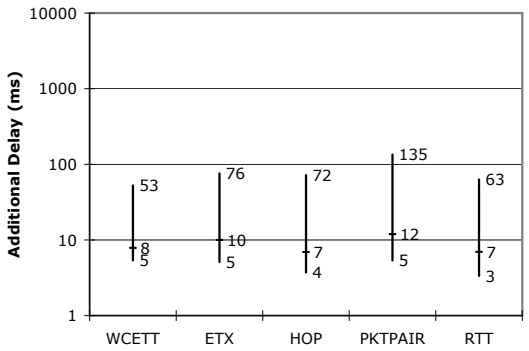

Figure 18: Performance over Medium Traffic Period, Distant Placement, Testbed Configuration A

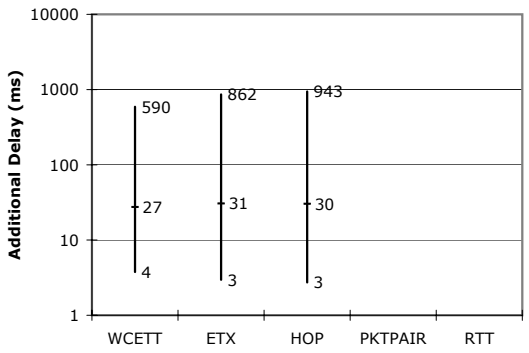

Figure 19: Performance over Heavy Traffic Period, Distant Placement, Testbed Configuration A

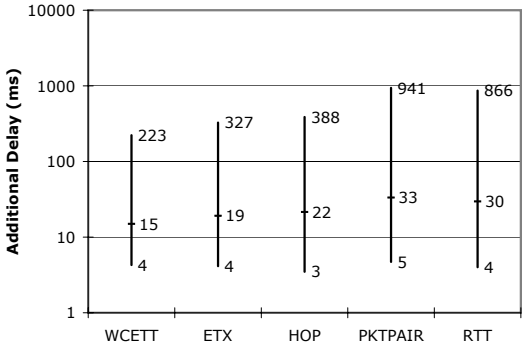

Figure 20: Performance over Heavy Traffic Period, Central Placement, Testbed Configuration B

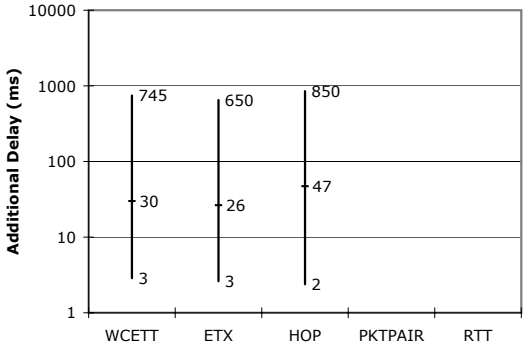

Figure 21: Performance over Heavy Traffic Period, Distant Placement, Testbed Configuration B

of transactions the delay is still acceptable. For the heavy traffic period, the median delay for WCETT, ETX and HOP has also increased but to around 30ms. PKTPAIR and RTT perform so poorly that most of the sessions are not able to transfer all their bytes and thus we do not show the delay values. For the experiments where more than 20% of the transactions did not finish in time, we do not consider them in the graphs. We have repeated these experiments and looked at detailed MCL statistics to confirm that poor choice of routes and probing overhead cause PKTPAIR and RTT to suffer.

We find that server placement has a direct effect on average path length and is crucial for achieving good performance. In our experiments, poor server placement could result in 3 times or more longer delays than a good server placement.

5.5 Performance across Different RTS/CTS Settings

Our office environment of dense node deployment and obstacles like doors and humans, coupled with different user-server placement can potentially affect the hidden-terminal problem. This is the case where wireless carrier sense is ineffective and packets interfere at the receivers. The RTS/CTS mechanism was designed to solve this problem, but the default setting in the drivers for all the wireless cards we use is to turn it off. In network configuration B, we enable RTS/CTS by setting the driver threshold to 100 Bytes. This means that for data packets of length at least 100 Bytes, the driver will enforce a full RTS-CTS-DATA-ACK exchange to mitigate hidden terminal problems.

Figures 20 and 21 show the results for the heavy traffic period in central and distant placement respectively. The results for light and medium periods in both placements are similar between configuration A and configuration B. In distant placement (Figure 21), the performance of WCETT, ETX and HOP is similar (within the experimental variance) to Figure 19. PKTPAIR and RTT still perform poorly and do not transfer most of the bytes. However, with central placement in Figure 20, while the first three metrics perform similarly to Figure 16, PKTPAIR and RTT improve both the median and 95th percentile.

While turning on RTS/CTS improves performance for PKTPAIR and RTT in one scenario, those metrics tend to perform poorly in other scenarios and we would not choose them for an all-wireless office. Between WCETT, ETX and HOP, turning on RTS/CTS does not have a significant impact on performance.

5.6 Detailed Performance Analysis : Hop Length, Transaction Sizes, Completions

To further understand the central versus distant server placement choice an operator has to make, we present Figure 22. For each experiment of metric, traffic load and user-server placement in mesh configuration A, we plot the byte averaged route length. That is, we multiply the number of hops for each transaction by the number of bytes transferred in it, and divide the sum by the total number of bytes transferred in each load scenario.

Clearly the distant placement in all cases requires longer route lengths than in the corresponding central placement. If all the servers are in the middle of the office, then on average the routes will be shorter. If routes are shorter, there are fewer transmissions on links (because fewer links are traversed by each packet), thereby reducing interference and increasing air time for more transactions. Also, each hop increases the delay experienced by the transaction. This explains why the median delay increases from the central placement to distant placement scenarios.

However, while this explains the difference in median delay, it does not explain why some transactions (albeit a small number) take significantly longer to complete. In some cases, the 95th percentile is significantly higher than the median additional delay. There can be two explanations for this. First, random temporary interference in the environment (from office occupants, microwave ovens, etc.) can cause individual transactions to suffer before routing has a chance to react. Second, given the limited bandwidth of IEEE 802.11 links, large byte transfers will correspondingly take longer to complete. In Figure 23, we show the additional delay experienced by each transaction and the number of bytes transferred by it. The results are similar for all heavy period experiments. It clearly shows a strong positive correlation between the size of the transaction and the additional delay incurred by it. However, it also shows that for some transactions the delay can get much higher than what is common for that transfer size. For the light and medium period experiments, the correlation is far weaker because they have far fewer large transactions.

In our experiments, the performance metric we consider is the additional transaction time incurred over the mesh testbed. However, for certain configurations, many transactions did not finish. Figure 24 shows the total number of bytes transferred in all the experiments with the heavy load period. As mentioned earlier, RTT and PKTPAIR suffer by not completing all transactions. We believe that this is due to the high overhead incurred by these metrics since they use unicast messages to each neighbor to measure link quality. The amount of overhead simply does not leave enough room for actual traffic to get through. Thus we would not choose them for an all-wireless office.

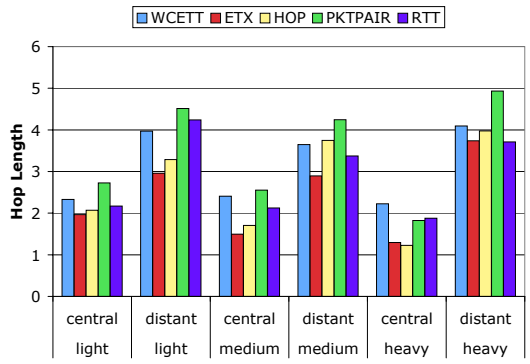

Figure 22: Byte Averaged Route Length for Configuration A

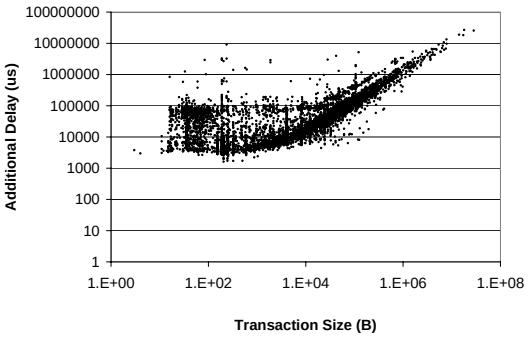

Figure 23: Delay Size Correlation for Heavy Traffic Period, Central Placement, WCETT Metric, Testbed Configuration A

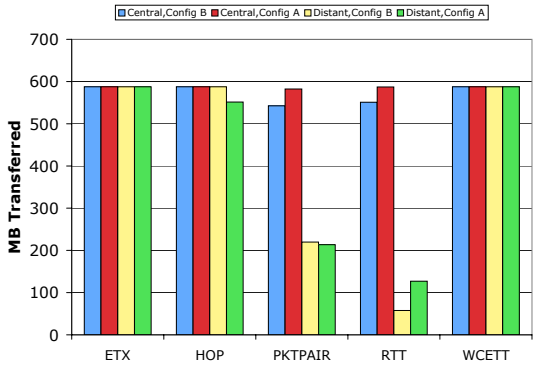

Figure 24: Total Bytes Transferred During Heavy Load Period

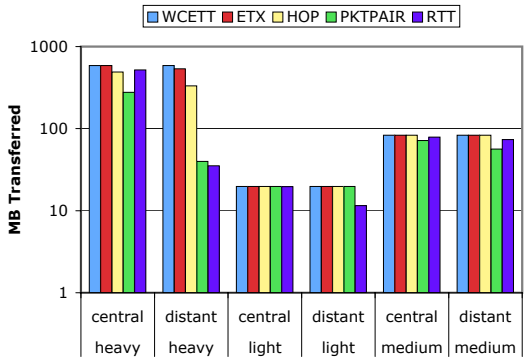

Figure 25: Bytes Transferred, Testbed Configuration C

5.7 Performance with Different Hardware, Bands and Power

An office mesh operator also has the choice of what wireless devices to employ, which of the IEEE 802.11 {a,b,g} standards to use, and whether reducing the transmit power can improve spatial re-use in such a dense office environment. While we do not attempt to evaluate all the available devices, we do want to determine if different devices impact performance, perhaps due to different error rates and transmit range. We now consider testbed configurations C, D, E and F.

Figure 25 shows the number of bytes transferred by each of the 5 metrics in all combinations of the traffic periods and user-server placement - again PKTPAIR and RTT do not transfer most of the bytes. Figure 26 shows the median additional delay incurred by transactions in this configuration. The light and medium load periods incur acceptable median delay in all cases and the heavy load period with distant placement is borderline. It is interesting to compare these results with configuration D in Figures 27 and 28. The two configurations use the same band, channel and power settings, but different hardware. Now the heavy load period with distant placement suffers more, both in terms of median delay and total number of bytes transferred. If we refer back to Figure 11, we see that configuration D provides significantly lower throughput than C. Also, compare configurations A and C, where the same hardware is used but different bands for one of the devices. Even through IEEE 802.11g offers the same link bandwidth as 802.11a, there is a significant reduction in performance in using it. As we reduce the transmit power, we find that configuration E at 50% power performs similarly to D at 100%, but F at 12.5% suffers. The median additional delay for configuration F is in Figure 29. Both placements of heavy load incur unacceptably high delay.

These experiments show that different devices significantly impact performance - an operator should evaluate the available hardware to determine the best one for the office mesh. We did not see any benefit from spatial reuse by reducing the transmit power.

5.8 Performance in Extreme Scenario

We now examine the limits of performance of the office mesh network - with the best configuration and metrics, how much load can be tolerated? We consider the “extreme” scenario from Table 4 in configuration A. Recall that we took the heavy load period, and replicated some of those users with a one hour time shift to more machines. With this we have 19 users. The remaining 2 machines are servers and each user sends traffic to both servers.

Figure 30 shows the median additional delay experienced by transactions in this scenario. Again, PKTPAIR and RTT are un-

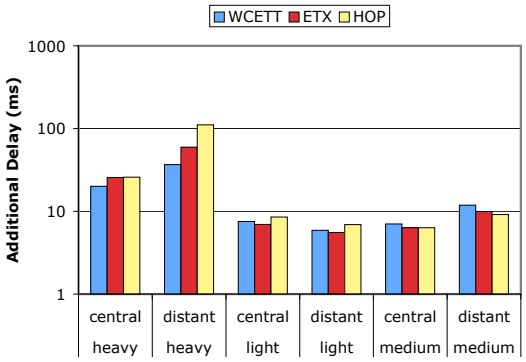

Figure 26: Median Additional Delay, Testbed Configuration C

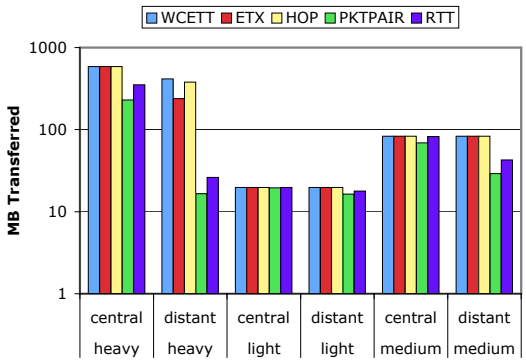

Figure 27: Bytes Transferred, Testbed Configuration D; Graphs for Configurations E and F are similar, except PKTPAIR and RTT Perform Worse

able to transfer even 25% of the total bytes and thus we do not present their data. WCETT, ETX and HOP all transfer about 90% or more. The median delay in this case is also acceptable, and is within the experimental variance of the heavy load scenario. We conclude that if these 8 replicated users were to perform similarly to the 11 captured users, a multihop-wireless mesh network using configuration A can support their traffic with acceptable delay for most transactions.

5.9 Scaling to a 100-User Office

An office of only 19 users is somewhat limited. The office floor in Figure 10 has about 80-100 offices. If all 80-100 offices had occupants and all used a wireless mesh network, then we would need to support about 100 users. We did not consider experiments at such a large scale because we did not have consent from enough users to monitor their traffic, and we do not have such a large testbed, even though our current testbed does geographically cover the entire area. However, note that in all experiments we considered only 2 channels across the testbed. IEEE 802.11a has 13 orthogonal channels. In theory, we can set up 6 parallel mesh networks, each of which covers the entire floor with about 22 nodes using 2 channels. In this way, we can provide good performance for 114 users. The few nodes with wired connectivity to the corporate enterprise network can potentially have up to 12 wireless interfaces to accept traffic from all 6 mesh networks. However, the performance of peer-to-peer traffic may suffer if one participant is in a different mesh network than the other. We posit the network can be allocated by organizational boundaries or by a dynamic channel assignment algorithm (such as [17]).

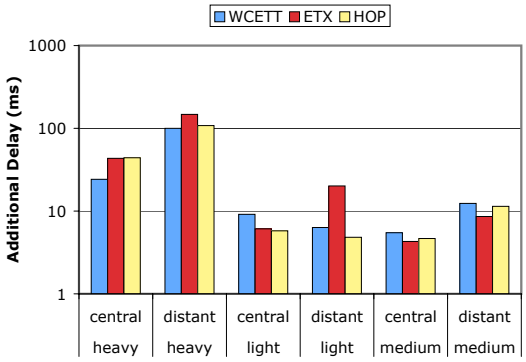

Figure 28: Median Additional Delay, Testbed Configuration D; Graph for Configuration E is similar

6. LIMITATIONS

Our study has some limitations:

- We only consider one testbed deployment. We did not attempt to measure the performance over a smaller or larger geographic distance, over a smaller or larger number of nodes nor with node mobility. We mitigated this by considering different wireless hardware, bands, transmit power settings, RTS/CTS settings.
- We did not consider subjecting our testbed to purposeful external interference. During the operation of our testbed, office doors were opened and closed by building occupants, occupants moved around, occupants re-arranged offices and used microwaves and cell-phones. It is possible that in the operation of a mesh network other additional forms of interference may occur that we did not experience. However, we show that during the operation of our testbed, the existing

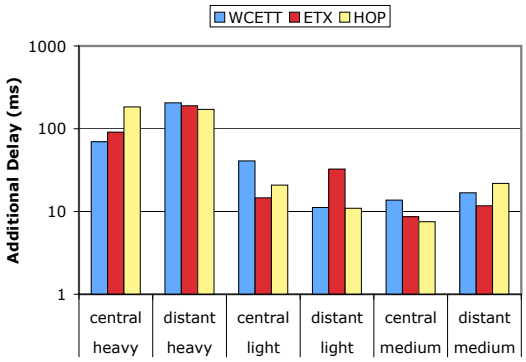

Figure 29: Median Additional Delay, Testbed Configuration F

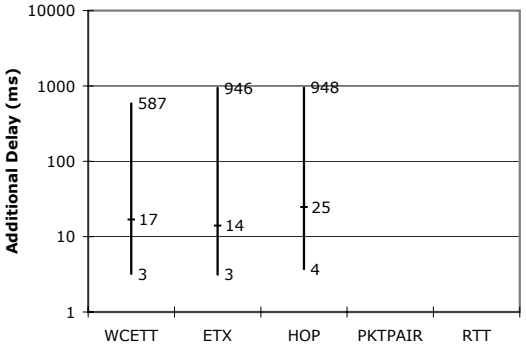

Figure 30: Performance over Extreme Traffic and Placement, Testbed Configuration A

forms of interference did not significantly impact the stability of our findings.

- We captured traffic from only a single set of corporate network users. There are some limitations of our capture methodology that omitted certain classes of traffic, which we found to be primarily intrusion detection traffic. We do not claim that this traffic trace is representative of all user traffic. Nonetheless, we believe the all-wireless office is a valid mesh usage scenario and thus we believe our trace-based evaluation is more representative of the performance that will be achieved in such networks than when using synthetic traces. We increase the confidence in our results by running experiments on a variety of sampled time periods.

7. CONCLUSIONS

In this paper, we examined the feasibility of offices with PCs that are cooperatively interconnected by an ad-hoc wireless mesh network, and with few servers or proxies that have wired connectivity to a corporate network or the Internet. We are not aware of any prior work that has evaluated an application scenario for mesh networks using both a deployed system and actual network traces. This paper fills that gap.

Our methodology includes capturing socket-level traces from office PCs. This traffic differs from synthetic traffic workloads in several ways, including host and size distribution, and concurrency of connections. We replay this traffic on our mesh network that is co-located with the trace participants. Our accurate replay mechanism introduces only an additional delay of 1.7ms, which we believe is a conservative estimate. Our mesh network experiences the natural environmental variability that a typical office mesh network will experience, and introduces under 10ms of error.

We examined a large set of design choices facing an administrator deploying an office mesh network - which routing metric to use, which IEEE 802.11 hardware to use, which bands to use, where to place users versus servers. In addition, we examined a few wireless settings that might shed light on performance, including transmit power levels and RTS/CTS thresholds. We examined multiple distinct load periods from our traces.

We find that the routing metric choice has a significant impact on network performance when the offered load grows close to network capacity. Metrics that make use of unicast probes to each neighbor incur high overhead, and suffer tremendously as contention for the medium increases. Server placement, having a direct effect on average path length, is crucial for achieving good performance. In our experiments, poor server placement could result in 3 times or more longer delays than a good server placement. The choice of hardware and IEEE 802.11 band can significantly impact delay.

Nonetheless, in the majority of our evaluation scenarios, the additional delay incurred by most individual transactions was under 20ms. Since we used traces from a single LAN and evaluated a mesh network within a single building, our results may not apply to all scenarios. However, we considered many different scenarios, including office time periods, different hardware, different wireless settings and different server placement. Thus we believe that wireless mesh technology, using modern metrics and intelligent server placement, has evolved to the point where deploying a wireless office can be realistically considered. Given the number of orthogonal channels available in IEEE 802.11a, we believe our results from a 21 node system are scalable to over 100 nodes. Given our results, we believe office mesh networks are feasible when wired connections to every PC are not readily available, and reduction in administration overhead is a significant factor.

8. REFERENCES

[1] IEEE 802.11s working group. http://grouper.ieee.org/groups/802/11/.
[2] Mesh Networks Inc. http://www.meshnetworks.com.
[3] Radiant Networks. http://www.radiantnetworks.com/.
[4] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi-radio unification protocol for IEEE 802.11 wireless networks. In *BroadNets*, 2003.
[5] I. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: A survey. In *Elsevier Computer Networks*, 2005.
[6] N. Bhatti, A. Bouch, and A. Kuchinsky. Integrating user-perceived quality into web server design. In *Computer Networks*, 2000.
[7] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MOBICOM*, 2005.
[8] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MOBICOM*, 2003.
[9] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *ACM SIGCOMM*, 2004.
[10] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh network. In *IEEE MOBICOM*, 2004.
[11] M. Dynamics. Performance analysis of three competing mesh architectures. http://www.meshdynamics.com/MDPerformanceAnalysis.html.
[12] J. L. et al. Simulation validation using direct execution of wireless ad-hoc routing protocols. In *PADS*, 2004.
[13] K. Farkas, O. Wellnitz, M. Dick, X. Gu, M. Busse, W. Effelsberg, Y. Rebahi, D. Sisalem, D. Grigoras, K. Stefanidis, and D. Serpanos. Real-time service provisioning for mobile and wireless networks. In *Elsevier Computer Communications*, 2005.
[14] F. Hernandez-Campos, F. Smith, and K. Jeffay. Generating realistic TCP workloads. In *CMG*, Dec. 2004.
[15] R. Karrer, A. Sabharwal, and E. Knightly. Enabling Large-scale Wireless Broadband: The Case for TAPs. In *ACM Hotnets*, 2003.
[16] M. K. Marina and S. R. Das. A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. In *BroadNets*, 2005.
[17] K. Ramachandran, E. Belding, K. Almeroth, and M. Buddhikot. Interference-aware channel assignment in multi-radio wireless mesh networks. In *IEEE INFOCOM*, 2006.
[18] A. Raniwala and T. Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *INFOCOM*, 2005.
[19] Seattle wireless. http://www.seattlewireless.net/.
[20] P. Selvidge. Examining tolerance for online delays. In *Usability News 5.1*, 2003.
[21] D. Tang and M. Baker. Analysis of a local-area wireless network. In *IEEE MOBICOM*, 2000.
[22] Y. Yang, J. Wang, and R. Kravets. Designing routing metrics for mesh networks. In *WiMesh*, 2005.