Experiences with a Mobile Testbed

Kevin Lai, Mema Roussopoulos, Diane Tang, Xinhua Zhao, Mary Baker

Stanford University

Abstract. This paper presents results from an eight-day network packet trace of MosquitoNet. MosquitoNet allows users of laptop computers to switch seamlessly between a metropolitan-area wireless network and a wired network (10 Mbit/s Ethernet) available in offices and on-campus residences. Results include the amount of user mobility between the wired and wireless networks, the amount of mobility within the wireless network, an examination of application end-to-end delays, and an examination of overall packet loss and reordering in the wireless network.
We find that the average mobile host switches between the wired and wireless networks 14 times during the trace and moves within the wireless network five times. Round trip latencies in the wireless network are very high, with a minimum of 0.2 seconds. Even higher end-to-end delays, of up to hundreds of seconds, are due to packet loss and reordering. These delays cause users to change their usage patterns when connected to the wireless network. We conclude that latency is a critical problem in the wireless network.

1 Introduction

The popularity of portable computers combined with the growth of wireless networks and services has led to many efforts to make mobile computing an everyday reality. To achieve this goal, designers need information about the behavior of mobile hosts (portable computers) and the characteristics of the networks they use. Although existing studies of the characteristics of local-area wireless networks are helpful for investigating in-building mobile environments [9] [19] [23], there is currently little packet-level information available about wide-area wireless networks. This information is important if we are interested in mobility outside of small areas such as homes and offices. Understanding wide-area wireless network performance in terms of latencies, packet rates and packet loss will help us design, model, simulate and optimize protocols and applications to perform well in this environment, while information about the day-to-day behavior of mobile hosts and their usage patterns in real systems will help us choose which protocols and applications to tackle.

This paper presents the results from an eight-day trace of a network of mobile hosts that incorporates a wireless system available throughout our metropolitan area. The network, called MosquitoNet, allows users of laptop computers to stay connected to the wireless network while moving around our area, and to switch seamlessly between the wireless and a wired network. Switching seamlessly means that all ongoing network applications continue working when a host's network connection changes.

Our results include the amount of user mobility between the wired and wireless networks, the amount of mobility within the wireless network, a comparison of usage patterns between the wired and wireless networks, an examination of application end-to-end delays, and an examination of overall packet loss and reordering in the wireless network. To summarize our findings:

- Mobility: On average, a mobile host switches between the wired and wireless networks 14 times during the eight days (with a minimum of three times and a maximum of 34). On average, a mobile host moves within the wireless network five times during the eight days (with a minimum of one move and a maximum of 14 moves). We found seven distinct locations at least one-half mile apart in the wireless network that were visited by mobile hosts. The widest spread locations were 70 miles apart.
- Latencies: Round trip latencies in the wireless network are high, with a minimum of 0.2 seconds. Much higher end-to-end delays (up to hundreds of seconds) result from packet retransmissions due to loss and reordering. High latencies and the rate of packet loss and reordering prevent hosts from fully utilizing the available bandwidth of the wireless network.
- Optimizations: Simply changing telnet from character mode to line mode improves its interactive response by requiring 50% fewer round trips. Batching NFS meta-data requests can reduce them by as much as 20%, but NFS still suffers from lack of data prefetching.

Many other wide-area wireless networks will also have high latencies relative to local-area technologies, so we conclude that contending with latency presents a critical challenge for making wide-area mobile data networks successful. Some simple techniques for improving the behavior of applications sensitive to high end-to-end delays, such as using local line editing in telnet, can vastly improve users' perceptions of network behavior. Additionally, it is clear that file systems such as NFS that use a request/response (RPC) model will not survive in the wireless environment unless they are optimized to prefetch data or combine multiple operations into single requests (such as looking up information for a whole directory at once).

The remainder of this paper is organized as follows. Section 2 describes the project goals and the environment studied. Section 3 describes the tracing and analysis tools. Section 4 describes the overall and per-application usage patterns of hosts on the wired and wireless networks, as well as end-to-end delays, packet loss, and packet reordering in our wireless network. Section 5 looks at some possible optimizations for telnet and NFS in the wireless network. Section 6 describes related work, and Sect. 7 presents conclusions and some future work.

2 The System Under Study

This section describes the goals of the MosquitoNet project and the hardware and software that make up our testbed.

2.1 Goals

The goal of the MosquitoNet project is to work towards providing "anywhere, anytime" Internet connectivity for mobile hosts. A host should be allowed to remain continuously connected to the Internet. If cost or battery power are concerns, then the mobile host may disconnect as appropriate, but it should be able to reconnect seamlessly whenever desired. If users have control over when to disconnect, then they can choose to do so gracefully, perhaps by synchronizing file caches or unlocking locked files.

For continuous connectivity, a mobile host must be able to switch to the best network available in its current location. The metric for choosing the network, if more than one network is available, could be performance, cost, available resources in the network, security, or some combination of the above and other features. We do not believe that a single, globally available wireless network will provide mobile hosts with the best performance or cost. Instead, we believe users will want the generally better performance of wired networks where they are available. They may also choose local-area wireless networks over wide-area wireless networks when available. While it is not yet possible for mobile hosts to find a practical wireless network in all locations, we believe the recent growth of wireless services is evidence that this is a direction worth pursuing.

2.2 Hardware

The MosquitoNet testbed currently consists of eight laptop computers (our mobile hosts), a router, and two networks. The laptops are IBM Thinkpad 560's, Samsung SensLite 200's and a Panasonic ProNote CF-11. The router is a 90MHz Pentium. The laptops and the router all run Linux. The two networks are 10 Mbit/s Ethernet and packet radio. We picked the packet radio network for two reasons. First, it is available throughout our metropolitan area, so it allows us to experiment with continuous wide-area mobility. Second, it is sufficiently different in characteristics from Ethernet to bring forth the problems and challenges that arise for applications, protocols, and users when connectivity switches between such extremes.

The packet radio network consists of Metricom spread-spectrum, frequency-hopping radios [17]. These operate in the 902-928 MHz band of the unlicensed spectrum allocated by the FCC for low-power devices. Each laptop is equipped with one such radio connected via the serial port. We consider the Metricom network to be one large wireless IP subnet. Radios may communicate with each other peer-to-peer when within range, but otherwise they communicate through Metricom's packet-switched, geographically routed network, which forwards packets between "poletop" transceivers spread around our metropolitan area. Packet forwarding within the wireless network is done at the link level and is invisible to higher-level protocol layers.

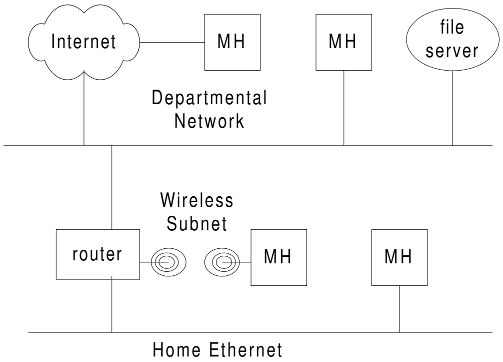

Fig. 1. This figure shows the layout of the MosquitoNet testbed. The router forwards packets between the wireless subnet, the "home" Ethernet of the mobile hosts, and the departmental Ethernet. Mobile Hosts (MH) can connect to their home network, or they can connect to ("visit") the wireless subnet or other Ethernet subnets available around the Internet. Note that the file server is on the departmental Ethernet, so all file server traffic to the mobile hosts runs through the router.

The layout of the MosquitoNet testbed is shown in Fig. 1. The router forwards packets between three networks: the wireless subnet, the mobile hosts' "home" Ethernet subnet, and the building-wide departmental Ethernet. Packets sent by mobile hosts visiting "foreign" networks such as the wireless subnet pass through the router (Sect. 2.3). Also, any packets from mobile hosts on their home Ethernet will go through the router when accessing hosts or services outside that network, such as our file server.

Although the radios are sold as Hayes AT modem emulators, we have written a device driver in the Linux operating system that uses their lower-level packet switching interface. This allows the one radio on our router to communicate with multiple client radios. Otherwise, we would need to run a SLIP or PPP service and attach a bank of radio modems to our router to support its point-to-point connections with multiple laptop clients. The router would require one radio per concurrently active client.

Compared to local-area networks, wide-area packet radio networks generally have lower bandwidth and higher latencies for the same level of power. Metricom radios are no exception. Their air transmission speed is 100 Kbits per second. We run our serial ports at 115,200 bits per second to allow for the full bandwidth of the radios. The radios are half-duplex, which means that data traveling through an intermediate radio has its bandwidth cut to half the air transmission speed. Per-packet latencies are very high, with a minimum round trip time of 200ms.

The Metricom radios have one additional characteristic that is not common to all packet radio networks: they do not support broadcast. A Metricom radio can only send a packet to one other radio at a time. To do so, the radios must tune their frequency hopping sequence to match each other. To handle broadcast protocols such as ARP, we currently establish one well-known radio hardware address as the address of an ARP server [1]. Hosts periodically let this server know their hardware-to-IP address mappings, and hosts can query this server for other hosts' mappings.

2.3 Software

To support mobility, it must be possible to switch seamlessly between networks, i.e., without disrupting ongoing network connections. The MosquitoNet prototype uses a mobile IP protocol [22] to achieve this. With mobile IP, connection-oriented protocols such as TCP can continue to use the "home" IP address of the mobile host, even if it switches to a network interface with a different IP address. Our mobile IP protocol supports the use of "foreign agents" if they are found in the networks to which a mobile host connects, but we prefer the extra flexibility that is provided when a mobile host acquires its own "care-of" IP address in the networks it visits [2].

Mobile IP uses encapsulation of network packets, and the size of the extra header reduces the maximum transmission unit (MTU) available when a mobile host is connected to a "foreign" network (any network, including the wireless, that is not its home Ethernet subnet). The radios have a small MTU to begin with (1024 bytes), which is further reduced to 1004 bytes with encapsulation.

3 Data Collection

This section describes how we gathered and analyzed our network traces. To gain a general picture of how mobile hosts are used, and to gather data to compare wired and wireless usage, we set aside eight days (from 4:00am one Saturday morning until 4:00am on Sunday morning) during which our research group used only our laptops as display and input devices.

We record two types of information for our traces: a tcpdump record of all packets sent over the network interfaces, and a record of the radio network status. We use tcpdump to trace IP and ARP activity on both the radio and Ethernet interfaces of each laptop, as well as on the radio and home Ethernet interfaces of our router. The radio status information includes the list of visible poletops and their signal strengths. We use this information to determine the physical location of the mobile host within the wireless network.

4 Users, Latencies and Packet Loss

We received feedback about the usability of the wireless network from all of our users. The overwhelming consensus is that high end-to-end delays are the biggest problem. Even typing a single character in telnet or doing a directory listing in NFS can incur a frustrating delay. Users report that X11 and NFS are unusable. They also report that telnet can be painful, but they use it anyway, since there are no better alternatives. However, all users reported web browsing (HTTP) to be acceptable on the wireless network. We believe this is the case for two reasons. User expectation allows for longer delay of operations while web browsing, since this can be slow even on wired networks when accessing sites over the Internet. Also, many web browsing operations do a more satisfying amount of work per user request than do more interactive applications such as telnet, which incurs a round trip delay for a single character echo.

In the rest of this section, we examinine host mobility and telnet and NFS behavior. We present evidence that high end-to-end delays cause users to change their behavior. These delays are due to the high per-packet round trip time of the wireless network and to packet loss and reordering in the network.

4.1 Host Mobility

We investigate the mobility of hosts in these networks. We are interested in answering questions such as whether users take advantage of the wireless network when the wired network is not available, how often they switch between the wired and wireless networks, and how much they move around geographically within the wireless network.

The first data column of Table 1 shows the number of times each host switches between the wired and wireless networks over the eight days. A host switches to the wired network when its user inserts an Ethernet PCMCIA card that is connected to the network. The movement between networks varies significantly across hosts, with the average number of switches being 14, the minimum three and the maximum 34. The host that switched networks 34 times also had the most traffic overall. Since users often leave the radios on even when they are using the Ethernet, we do not count the wireless network as being active if the Ethernet interface is active as well.

The second data column of Table 1 shows that users take advantage of the wireless network at many locations. The table presents the number of times a host uses the wireless network, moves by at least a half-mile, and then uses the wireless network in the new location. The average number of moves is five, with a minimum of one and a maximum of 14. There are seven distinct locations represented in this data. Known visited locations include the Computer Science department, the student campus residence area, the home of a faculty member, a cafe on the south side of Palo Alto, Fremont, San Francisco, and the beach in Santa Cruz. San Francisco and Santa Cruz are about seventy miles apart.

Table 1. This table shows the number of times a given laptop switches between the wired and wireless networks and the number of times it changes its physical location within the wireless network. Note that this is not a count of the number of user sessions, as these may extend across network switches.

Host Name	ant	bee	butterfly	dragonfly	junebug	midge	termite	weevil
Num. of Network Switches	11	8	12	34	16	3	13	18
Num. of Moves	2	5	5	5	8	3	1	14

Unfortunately, using poletop information alone, we are not able to detect movement within a one-half mile radius. For instance, we are unable to distinguish between a laptop user using his laptop in his office and at a nearby library. Thus, this data represents a lower bound on the amount of mobility while using the wireless network.

4.2 End-to-End Delays in the Wireless Network

In this section, we look at end-to-end delays experienced in the wireless network by telnet (over TCP) and NFS (over UDP), two of the applications users reported as being slow. We find that the high delays have two causes. The first cause is the high minimum round trip latency of the wireless network. The second cause is a high level of packet retransmissions, due to loss and reordering in the network.

We calculate the end-to-end delays for telnet and NFS in a manner that reflects what the users actually experience with these applications. For telnet, the delay is the difference between when data is sent and when the first acknowledgment for that data is received. For NFS, the delay is the difference between when the first packet for an NFS request is sent and when the first reply is received.

Figure 2 shows the distribution of round trip times seen for telnet and NFS traffic. These results show delays in the wireless network ranging from 0.2 seconds to several hundred seconds. Note that even the minimum latency is noticeable to users, since previous reports show that users begin to find interactive response time slow when it exceeds 100 to 200ms [12]. The median delays are painful, but not hopeless: the telnet median is 0.97 seconds and the NFS median is 0.6 seconds. Sixty percent of telnet delays are 1.3 seconds or less, and sixty percent of NFS delays are 0.7 seconds or less. However, the high end of the scale is clearly intolerable.

From the figure, we see that telnet and NFS delays differ in distribution. This is because TCP and NFS retransmission strategies and parameters differ. NFS uses a fixed exponential backoff starting at 0.7 seconds. However, TCP in general waits longer than NFS to retransmit, and so its overall delays will be larger. To set its retransmission timer, TCP uses an adjustable mechanism based on its estimate of the average round trip time and the average deviation of round trip

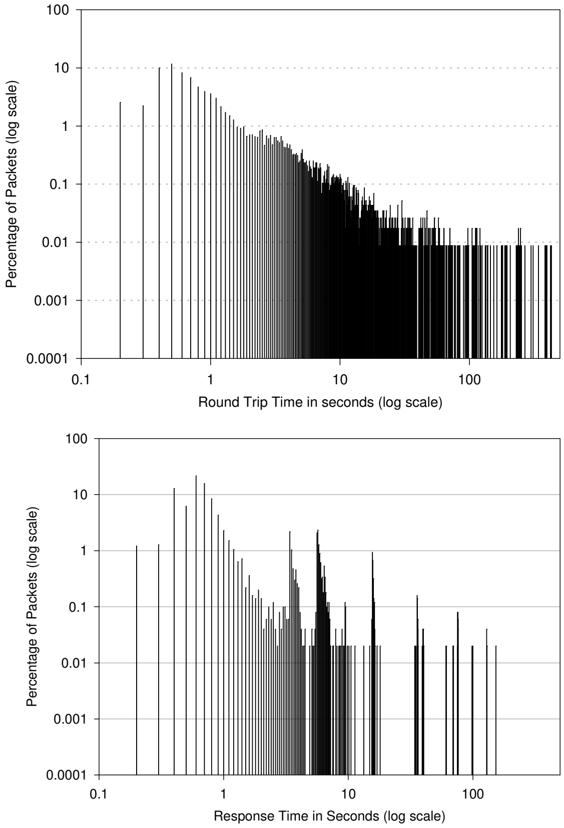

Fig. 2. This figure shows the percentage of telnet packets (top) and NFS request packets (bottom) versus response time for the wireless network.

times [13]. For TCP's retransmission timeout to be as small as NFS's, it would have to calculate the deviation in round trip times as being less than 125ms. However, even under ideal conditions, TCP's average estimate of the deviation is 225ms. (This result was measured by pinging packets peer-to-peer between nearby radios on two otherwise inactive hosts. There was an average round trip time of 335ms over the 1800 samples.) TCP's adjustable mechanism also causes the smoother distribution of telnet delays, while NFS shows two peaks (of about three percent) between one and ten seconds, which correspond to its retransmission timeouts.

Although NFS delays are generally shorter than telnet delays, we find that users are more frustrated with NFS than telnet. We believe this is because a user can see what progress telnet is making, even if it is slow. In contrast, when NFS is slow, users cannot see what is actually happening.

The very long delays of telnet and NFS are due to packet retransmissions. Figure 3 shows the number of telnet and NFS requests that undergo a given number of retransmissions. In both cases, 80% or more of the requests are transmitted only once (no retransmissions). One NFS request is retransmitted 26 times!

The reasons for these retransmissions are packet loss and a small amount of reordering. Table 2 summarizes these results. The data is divided into packets sent from the router to the mobile hosts and packets sent from the mobile hosts to the router. For transmissions from the mobile hosts to the router, we see a 25.6% packet loss rate. This is clearly high enough to cause severe performance degradation in the network.

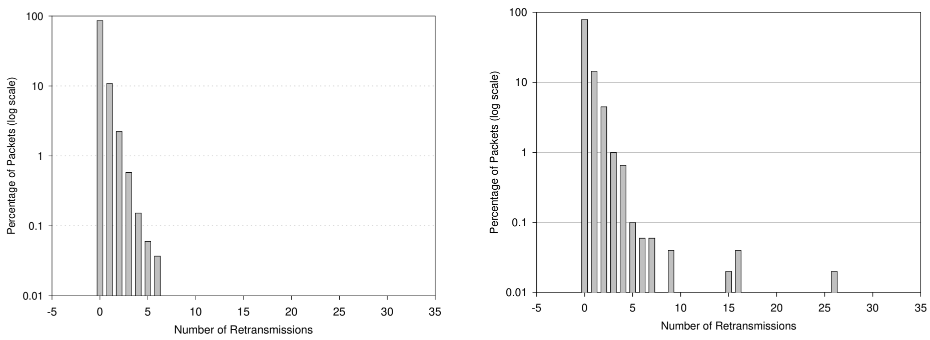

Fig. 3. This figure shows the percentage of telnet packets (left) and NFS request packets (right) that undergo a given number of retransmissions.

It is interesting to note that packet loss is much lower in the reverse direction, from the router back to the mobile hosts (3.6%). This is because a mobile host will retransmit requests until network conditions are good enough (the mobile host is within range of a poletop, or a temporary condition of network interference or congestion has eased). When conditions are good, the mobile host's packets will get through to the router. Because the responses to these packets follow quickly, they will usually experience the same good conditions and will successfully reach the mobile host on the first try.

Table 2 also shows packets arriving out of order. One of the reasons for this is that packets can take multiple routes from a source to a destination in our radio network. Since a Metricom radio is usually in range of several poletops through which it could send packets, packets sent later through a faster route will arrive before packets sent earlier.

Table 2. This table summarizes the characteristics of the wireless network. We classify packets into two categories: packets sent from the mobile hosts to the router and packets sent from the router to the mobile hosts. The numbers in parentheses give the actual number of packets involved.

Network Characteristic	Mobile Hosts to Router	Router to Mobile Hosts
Packets lost	25.6% (13,232)	3.6% (2,143)
Packets received in order	72.3% (37,380)	91.3% (53,832)
Packets received out of order	2.1% (1,103)	5.1% (3,059)
Average out-of-order distance	3.04	3.9

We are able to calculate packet loss and reordering by matching packet "signatures" on the hosts and the router. The packet signature is a 32 bit CRC. A packet sent by a host is considered lost if its signature does not appear in the router's trace, and vice versa. Packet A is considered to be received out of order if it arrives at the destination after some packet B that was sent later. The out-of-order "distance" of a packet A is the number of packets sent after it but received before it.

This packet reordering can contribute to retransmissions in protocols, such as TCP, that are sensitive to it. TCP normally accounts for minimal out-of-order delivery of packets, but treats packets as lost if they are delivered at an out-of-order distance of three or more. Each out-of-order packet causes a duplicate ACK to be sent, and three duplicate ACKs cause the sender to perform a fast retransmit [26]. Unfortunately, the average out-of-order distance in MosquitoNet is greater than three, so some of these packets are treated as lost and will thus contribute to the total number of retransmissions.

5 Application Optimizations for the Wireless Network

In this section, we consider some optimizations for telnet and NFS over the wireless network. Since we are unable to reduce the latency of the wireless network, we look for ways to avoid and hide end-to-end delays. We can avoid delays in telnet by batching together as many characters as possible into a packet. For NFS, we can hide delays by prefetching data and avoid delays by batching together lookups of files in one directory into a single request.

Telnet handles long end-to-end delays poorly because users must suffer the round trip time for every character they type. In addition, placing only one character in a packet is inefficient use of the network and router. With Nagle's algorithm, which sometimes batches together characters in interactive traffic, this problem would not be as severe [18]. However, telnet applications sometimes turn off this algorithm because it increases the end-to-end delay for terminal keystrokes that generate multiple characters [25]. Our telnet traffic does not use Nagle's algorithm.

One possible optimization is to use “line mode telnet” [5]. This allows an entire line to be typed and possibly edited before a packet is sent. For a line 10 characters long, line mode telnet would use 2 packets sent and received, while regular telnet would use 22 packets sent and received. More importantly, the user would only suffer the round trip delay once instead of 11 times (once for each character and once for the return key). This optimization is more effective for users than Nagle’s algorithm, because it gives users immediate feedback for what they have typed and allows them to edit without suffering any network latency.

We compared the trace of one of our heavy telnet users using line mode telnet for one day with the same user’s trace (using regular, or “character mode” telnet) during our eight-day period. Table 3 summarizes the TCP payload sizes of packets sent from the mobile host to the router. We omit TCP ACKs because they are not sent as a direct result of user actions.

Table 3. This table compares the payload size for packets sent from the mobile hosts to the router using character mode telnet and line mode telnet.

Telnet Mode	Number of Samples	Minimum Size	Median Size	Average Size	Maximum Size	Total Size
Character	650	1 byte	1 byte	8.52 bytes	698 bytes	5544 bytes
Line	302	1 byte	4 bytes	18.44 bytes	697 bytes	5571 bytes

Although the total payload bytes sent from the mobile host to the router for line mode telnet is approximately the same as for character mode, the line mode trace has 54% fewer packets. This means that the router had about 50% less incoming load on it and the user suffered the round trip time 50% less frequently for the same amount of data typed. All the users who have tried this mechanism report that it makes telnet acceptable on the wireless network. As a result of this experiment, we are developing a mobile shell that uses this technique but is more sophisticated about operations such as file editing that do not work well in line mode.

We also look at possible optimizations for NFS. Overall, the request/response (RPC) nature of NFS is awkward in a network with high per-packet latencies. A new request will not be started until the previous one has received a reply. The key to improving NFS performance is to reduce the number of end-to-end delays seen by users. We can do this by caching and prefetching more data, pipelining requests, or batching requests into single packets when possible.

NFS would benefit from techniques such as prefetching, as indicated by Fig. 4, which shows the distribution of NFS request types on the wired and wireless

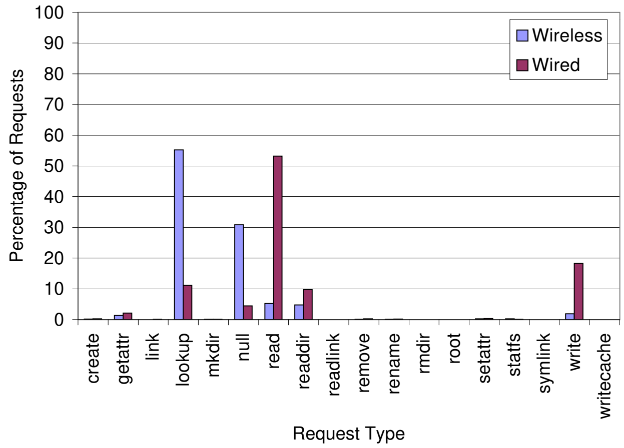

Fig. 4. This figure shows the distribution of NFS request types on the wired and wireless network.

networks. Users tend to decrease dramatically the number of data requests they make when switching from the wired network to the wireless network (from 185,014 to 357). As a result, techniques to improve data transfers, by prefetching, and to reduce needed transfers, by more aggressive caching, are essential in making file system performance tolerable on the wireless network.

Batching together multiple requests into single packets might also help reduce end-to-end delays. For example, we could speed up listing large directories if all the directory information could be transferred in one READDIRPLUS operation, rather than in many LOOKUP operations as is currently the case. This optimization is in the NFS version 3 specification [7]. To determine the possible benefit of the READDIRPLUS operation, we look at how many associated LOOKUP requests follow a READDIR request. The READDIRPLUS request would return attributes in addition to filenames, which is equivalent to batching the READDIR and LOOKUP requests. Given the 1004-byte MTU of the radios (including encapsulation overhead), we can fit the attributes and filenames for a maximum of six files in a packet. Thus, if six or fewer LOOKUPS follow a READDIR, they can be batched into one request packet.

Table 4 shows the results of batching. The number of batchable LOOKUP and READDIR operations is reduced by a factor of six (from 18,527 to 3,015 requests in the wired case), and the number of original meta-data operations is reduced by over 20% (from 73,704 to 58,192 requests). This would somewhat improve the speed of directory listings, which was one of the operations particularly reported as slow by our users. Note, however, that this corresponds to only a six percent improvement in overall requests on the wired network, so it might not improve overall NFS performance significantly.

Table 4. This table shows how batching LOOKUP requests with their associated READDIR requests into a single READDIRPLUS request would affect our NFS workload.

Type	Wired	Wireless
Original Requests: total, (data;meta-data)	258,718, (185,014; 73,704)	5,031, (357; 4,674)
Number of READDIR's with more than 1 LOOKUP following	1,070	164
Number of LOOKUPs following READDIR average, (minimum; maximum)	16.3, (1; 307)	15.25, (1; 20)
Number of original LOOKUP and READDIR requests that we can batch together	18,527	2,665
Number of batchable LOOKUP and READDIR requests after batching	3,015	438
Resulting number of requests after batching originals: total, (data; meta-data)	243,206, (185,014;58,192)	2,804, (357; 2,447)

6 Related Work

We have found little literature about packet-level studies of wide-area networks, although [11] includes a latency measurement of 15 seconds on average for a round trip TCP/IP packet over CDPD [8]. Also, a study of asymmetry on TCP performance [4], found the mean end-to-end round trip delay through a Metricom packet radio network to be 2.5 seconds, vastly larger than our measured mean of 0.97 seconds for telnet over TCP. While their measurements were made from a bulk TCP transfer, rather than collected over the course of days, the reason for this significant difference in mean delays remains unclear.

In contrast, there are several packet-level studies of local-area wireless networks. Work by [9] and [19] looks at the types of packet errors found in these environments, and [23] also reports throughput. More recent work by [21] gives an extensive study of the packet loss, latencies, and bandwidth of the WaveLAN network as seen by a host moving between buildings (but within the line of sight of their WavePoint basestations) and within buildings. Packet loss between buildings reaches as high as almost 40%, and corresponding latencies reach almost one second. In another scenario, while in an elevator, packet loss reaches 100%. Otherwise, packet loss seems to be quite low, averaging from less than one percent to about five percent, and latencies seem to hover around 10ms. We can conclude that the behavior of this wireless network depends greatly on the location of the mobile host, but that it generally shows an order of magnitude better latency, packet loss and bandwidth than the wide-area network in our study.

Besides measurements of the WaveLAN network, [19] and [21] also present a method to observe and predict the behavior of applications in a wireless network in a repeatable manner. The authors first measure end-to-end behavior of round trip packets in a local-area wireless network (WaveLAN). This information is distilled into a model of packet loss and delays. A modulation layer is then placed between applications and the wired network, and this layer drops and delays packets according to the model. The results of running several workloads on the wired network using this modulation technique closely predict the behavior of those workloads when run in the real wireless network. It would be very useful to repeat this effort in a wide-area wireless network.

There are many efforts to adapt applications and protocols to networks with weak connectivity. Coda [16] uses a technique called hoarding to cache copies of needed files on a laptop so they will be locally available during periods of poor connectivity. This form of aggressive caching and prefetching, along with cache consistency protocols, allows mobile hosts to avoid issuing many requests over a poor link. Rover [14] uses queued, non-blocking remote procedure calls to allow applications to continue processing even if the remote procedure call must wait until network connectivity is reestablished. Thus Rover's queuing techniques can help hide the poor connectivity and high latencies found in a wireless network environment.

The concept of proxies has been used to improve performance of many protocols such as X, HTTP and TCP in low-bandwidth and high-latency environments [15] [27] [3]. As seen in [3], this technique is particularly helpful for avoiding TCP's backoff and retransmission features, which were developed for wired networks in which packet loss indicates congestion [6]. Unfortunately, there are many networks a mobile host might visit (particularly networks not under the control of the mobile host's own administrative authorities) in which it may not be possible to place proxies in useful positions. Nonetheless, our experiences in MosquitoNet lead us to believe that proxies should be adopted where possible.

7 Conclusions and Future Work

The high latencies and rate of packet loss and reordering prevent hosts from fully utilizing the available bandwidth on the wireless network. Other wide-area wireless networks, such as CDPD, also have high or even higher latencies. Unfortunately, network manufacturers currently tend to focus on increasing bandwidth rather than reducing latency [24]. We need to convince network manufacturers that reducing latency is just as crucial to the success of wide-area wireless networks as increasing bandwidth.

We must also search for ways to optimize the behavior of applications sensitive to high end-to-end delays. We have at least two options. The first is to avoid latency from applications. The local line editing technique we tried for telnet was simple, yet effective enough to change users' perception of telnet performance on the wireless network. We will soon switch to using our mobile shell, which incorporates this optimization and other features. The second option is to hide end-to-end delays. NFS, for example, will not survive in the wide-area wireless environment without techniques such as prefetching data. We must either fix NFS, or we must switch to a more sophisticated file system (such as Coda [16]) that provides these techniques.

We plan to incorporate into MosquitoNet several other techniques that will improve the performance of our wireless network. For instance, we are currently experimenting with adding more than one radio onto our router. We do this by creating a virtual radio device that actually uses one or more radios for incoming packets and one or more radios for outgoing packets. To the outside world, all the radios appear to be functioning as one single device. This way, we can minimize some of the latencies caused by the router's single radio switching between incoming and outgoing traffic.

8 Acknowledgements

We would like to thank the USENIX Association, Daimler-Benz Research and Technology and the Xerox Palo Alto Research Center for supporting the student authors of this paper. This work was also supported in part by MITI (Ministry of International Trade and Industry) through the "Advanced Software Enrichment Project" of IPA (Information-technology Promotion Agency, Japan).

References

1. Armitage, G.: Support for Multicast over UNI 3.0/3.1 based ATM Networks. Internet Request for Comments 2022, November 1996.
2. Baker, M., Zhao, X., Cheshire, S., Stone, J.: Supporting Mobility in MosquitoNet. Proceedings of the 1996 USENIX Conference, San Diego, CA, January 1996.
3. Balakrishnan, H., Padmanabhan, V., Seshan, S., Katz, R.: A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. Proceedings of SIGCOMM'96, August 1996.
4. Balakrishnan, H., Padmanabhan, V., Katz, R.: The Effects of Asymmetry on TCP Performance. Proceedings of Mobicom'97, September 1997.

5. Borman, D.: Telnet Linemode Option. Internet Request for Comments 1184, October 1990.
6. Caceres, R., Iftode, L.: Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. IEEE Journal on Selected Areas in Communications, vol. 13, no. 5, June 1995.
7. Callaghan, B., Pawlowski, B., Staubach, P.: NFS Version 3 Protocol Specification. Internet Request for Comments 1813, June 1995.
8. Cellular Digital Packet Data Specification. Release 1.0. July 19, 1993.
9. Eckhardt, D., Steenkiste, P.: Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network. Proceedings of SIGCOMM'96, August 1996.
10. Fulton, J., Kantarjiev, C.: An Update on Low Bandwidth X (LBX); a Standard for X and Serial Lines. Proceedings of the 7th Annual X Technical Conference, January 1993,
11. Housel, B., Lindquist, D.: WebExpress: A System for Optimizing Web Browsing in a Wireless Environment. Proceedings of Mobicom'96, November 1996.
12. Jacobson, V.: Compressing TCP/IP Headers for Low-Speed Serial Links. Internet Request for Comments 1144, February 1990.
13. Jacobson, V.: Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno. Proceedings of the 18th Internet Engineering Task Force, p. 365, September 1990.
14. Joseph, A., deLespinasse, A., Tauber, J., Gifford, D., Kaashoek, M.: Rover: A Toolkit for Mobile Information Access. Proceedings of the 15th ACM Symposium on Operating Systems Principles, December 1995.
15. Kantarjiev, C., Demers, A., Frederick, R., Krivacic, R., Weiser, M.: Experiences with X in a Wireless Environment. Proceedings of the Usenix Mobile and Location-Independent Computing Symposium, August 1993.
16. Kistler, J., Satyanarayanan, M.: Disconnected Operation in the Coda File System. Proceedings of the 13th ACM Symposium on Operating Systems Principles, October 1991.
17. Metricom, Inc.: URL: http://www.metricom.com.
18. Nagle, J.: Congestion Control in IP/TCP Internetworks. Internet Request for Comments 896, January 1984.
19. Nguyen, G., Katz, R., Noble, B., Satyanarayanan, M.: A Trace-Based Approach for Modeling Wireless Channel Behavior. Proceedings of the 1996 Winter Simulation Conference, 1996.
20. Noble, B., Nguyen, G., Satyanarayanan, M., Katz, R.: Mobile Network Tracing. Internet Request for Comments 2041, October 1996.
21. Noble, B., Satyanarayanan, M., Nguyen, G., Katz, R.: Trace-Based Mobile Network Emulationr. Proceedings of SIGCOMM'97, September 1997.
22. Perkins, C.: IP Mobility Support. Internet Request for Comments 2002, October 1996.
23. Reynolds, N., Duchamp, D.: Measured Performance of a Wireless LAN. Proceedings of the 17th Conference on Local Computer Networks, September 1992.
24. Satyanarayanan, M.: Keynote Address to ACM Mobicom'96. November 1996.
25. Stevens, W.: TCP/IP Illustrated, Volume 1. Addison-Wesley, 1994, p. 269.
26. Stevens, W.: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Internet Request for Comments 2001, January 1997.
27. Watson, T., Bershad, B.: Local Area Mobile Computing on Stock Hardware and Mostly Stock Software. USENIX Symposium on Mobile and Location-Independent Computing, August 1993.