Measuring NFS Performance in Wireless Networks*

Cynthia D. Rais and Satish K. Tripathi
Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
College Park, MD 20742
{*cldavis, tripathi*}*@cs.umd.edu*

December 18, 1995

Abstract

Technological trends suggest that soon communication networks will consist of a high speed wired backbone with numerous wireless Local Area Networks. Mobile computing and wireless subnetworks are increasingly in demand. Mobile routing solutions provide wireless LANs with seamless connectivity to backbone wired systems. However, these solutions do not provide acceptable performance. Wireless networks have distinct transmission characteristics which present challenges to achieving efficient performance. Performance over wireless links is limited by high error rates, mobility, and low bandwidth. We have studied the performance of TCP and NFS over a wireless network. The prevalence of these protocols means that mobile hosts will frequently use them when communicating with stationary hosts. Measurements have been collected to determine the response of these protocols in the presence of various error patterns. These measurements show that NFS and TCP performance suffer extreme degradation due to these wireless link characteristics. Unexpectedly, NFS performance is not better than an TCP FTP file transfer. NFS performance over wireless links is limited by large packet sizes, long retransmission timeouts, and slow response to losses. Our goal is to understand the effects of wireless communication on these protocols and improve performance without requiring changes to the current network Infrastructure.

1 Introduction

The development of lightweight, battery powered, portable computers and wireless adapter cards has greatly increased the demand for mobile computers. Mobile users utilize wireless networks to achieve continuous and flexible access to the resources of stationary backbone networks. The advantages of mobile computers ensure that they will be a pervasive part of the computing infrastructure in the near future. However, there are many technical challenges to surmount in the design of mobile and wireless systems before the ideal of ubiquitous network access can be realized.

Much research has been conducted in recent years to obtain and optimize mobility routing solutions. The amount of research in this area is indicative of the current demand for mobile computing. The Internet Engineering Task Force (IETF) has recently defined the Mobile Internet

*This work is supported in part by NSF grant CCR 9318933 and IBM equipment grants.

Protocol (Mobile IP) which provides the necessary routing support for mobile hosts within the Internet Protocol (IP) [Sim94]. Mobile IP incorporates solutions from various mobility support proposals [IDJ93], [PB94], [Rek94], [TT93], [WYOT93].

Although the current routing solutions provide mobility support for wireless systems and give users the freedom to move through the network and maintain connectivity, these solutions fail to provide acceptable performance in the presence of the difficult characteristics introduced by wireless links and mobility. The high error rates, low bandwidth, and temporary network disconnections experienced by wireless networks all contribute to inferior performance.

The technical challenges of mobile computing stem from the differences between wired and wireless systems. Unlike wired networks, which are virtually error free, wireless networks are error prone due to signal propagation characteristics and limitations. Errors on the wireless medium are often bursty due to signal interference, signal fading, and multipath effects. As the relative position of corresponding hosts changes, one host is likely to move into a fading zone, to the limit of propagation range, or into an area of high interference. This bursty error pattern has a significant negative impact on the throughput achieved on wireless links. The bursty nature of errors causes high packet loss rates and correlated losses, rendering many traditional error recovery policies inefficient.

The performance of mobile systems is further limited by low bandwidth and host motion. The bandwidth on wireless networks (several megabits per second) is much lower than that on wired networks. This is due to the power limitations of mobile computers and the limited spectrum available for use in wireless computing. Host migration also limits performance by causing temporary disconnections and latency periods while rerouting occurs. The Mobile IP solution allows mobile hosts to move transparently within IP, but there is a high performance cost for this mobility and for the flexibility allowed by wireless networks. Improved methods are required to achieve consistent and reliable performance for mobile users.

The Transport Control Protocol (TCP) and the Network File System (NFS) are two prevalent protocols which were optimized for wired networks and therefore suffer performance degradations in wireless systems. NFS is used throughout industry and academia to provide transparent access to remote file systems. These two protocols are so widespread that mobile users will undoubtedly need them to communicate with both wired and wireless hosts. Since NFS is prevalent throughout the Internet, mobile users will interact with this protocol whenever they access files available on traditional wired networks. Mobility and high error rates cause poor performance in both TCP and NFS.

Many studies have investigated the performance of TCP over wireless networks [CI94], [YB94], [BB94], but NFS performance has not yet received this attention. We believe that it is important to study the dynamics of these protocols and determine which features cause the poor performance in the presence of errors. This understanding will allow us to propose solutions which will be applicable to a variety of environments and not just to our own testbed.

Our experiments monitored NFS and TCP connections at the packet level and allowed us to study the effects of losses on the connection dynamics. We studied the behavior of these protocols in various error conditions. These studies provided us with throughput information as well as graphs of the distribution of the packets and the acknowledgments. This information provided insight into the cause of performance limitations and allowed us to study the interaction of the errors, losses, and retransmission patterns.

Our experiments demonstrate poor performance of NFS over wireless links. Burst errors significantly degrade the NFS throughput. Unexpectedly, in the presence of burst errors NFS performance was often worse than that of a TCP file transfer using the file transfer protocol (FTP). The NFS performance degradation results from long idle periods between retransmissions, large packet sizes and fragmentation, and unbalanced speeds between the mobile host and the NFS server. The next section reviews related studies on TCP and provides background on the TCP and NFS protocols. Section 3 discusses error characteristics on wireless media and the testbed on which our experiments were carried out. Section 4 presents measurements, graphs, and data which shows the performance of TCP and NFS over bursty wireless links. The last section summarizes our results and discusses future research directions.

2 Background and Review of Related Work

The performance of TCP over wireless links has been an area of active research recently. TCP is a reliable transport protocol optimized for wired networks and stationary hosts. It uses congestion windows and round trip timers to control the flow of traffic into the network. TCP interprets increases in round trip times as packet losses due to congestion and then uses back-off and recovery algorithms to slow the flow of messages into the network.

The premise underlying these congestion control policies is that packet losses in the typical wired TCP/IP environment are due primarily to congestion. While this premise is true for many existing networks, it is not true for networks with mobile hosts and wireless links. The burst losses on wireless links cause long idle periods when transmissions consistently fail. Since these losses are not a result of congestion, decreasing the transmission speed causes unnecessary throughput reduction. The losses and delays which occur on wireless systems in combination with TCP's congestion control algorithms can be detrimental to performance in wireless and mobile networks.

Some studies investigate the effect of adding link layer retransmissions to improve the throughput of the wireless link [BKT95],[DCY93]. Others propose higher layer methods for improving the TCP performance over wireless links, such as partitioning the connection [YB94], [BB94] or notifying the transport layer of host motion [CI94].

2.1 Wireless Link Retransmissions

Selective retransmission can be implemented to react to the high loss rates over the wireless link. One possible approach is the use of a link layer protocol to control retransmissions on the wireless link and hide such losses from the higher layers. Since errors occur more frequently on this link, some researchers claim that retransmission over the wired part of the connection waste bandwidth and time. Other researchers argue that competing retransmission schemes in the link and transport layers will reduce end-to-end throughput even though the link utilization is increased [CI94], [YB94]. DeSimone et al. [DCY93] have shown that link layer retransmissions can adversely affect the end-to-end mechanisms of reliable transport protocols reducing end-to-end throughput and increasing the wireless link utilization.

One possible method is to introduce a low level protocol, a Channel State Dependent Packet scheduler, to control packet transmission [BKT95]. This method maintains a queue for each destination and switches between these hosts based on the quality of the 'channel' to each host. If one packet is lost the channel would be assumed to be lossy for some period of time, during which other "good" channels would be given priority to transmit. This would mean that the lost packet would be retransmitted only after it was highly likely that the burst error period was over. Determining the quality of each wireless channel is a difficult part of this method. The link transmitters must be able to determine when the channel is in a burst period and predict the length of the burst. Studies would have to be conducted in individual environments to determine reasonable burst error patterns. This method improves the fairness of the link layer utilization because it prevents head-of-the-line blocking when the packet at the head of the queue has to be repeatedly transmitted over a bad channel. This method could be used in conjunction with all transport layer protocols such as NFS and TCP.

2.2 Partitioned Connection

Two studies, [BB94],[YB94], propose a separation of the transport session into the wired portion and the wireless portion to prevent the high loss characteristics of the wireless segment from causing retransmission over the wired network. The base station or router which joins the wired and wireless networks must buffer the packets transmitted to or from the mobile host.

The disadvantages of this approach include lack of continuity on the end-to-end connection and a requirement for a large buffer space at the base station for each transport session to any of the mobile hosts in that cell. The faster speed of the wired network could cause a large accumulation of data at the base station. If the mobile host migrates, it is difficult and inefficient to forward the contents of the buffer to the new base station. If buffered packets have been acknowledged by the base, then consistency problems could result from disconnection or migration by the mobile host.

Bakre and Badrinath consider the performance of TCP in the presence of mobile host cell switches [BB94] and propose Indirect TCP (I-TCP). This study uses a wireless testbed and simulated cell switches. Yavatkar and Bhagawat have studied the performance of TCP in the presence of wireless links. This study considers the effects of both mobility and wireless link characteristics, and uses simulation for the wireless segment of the connections [YB94].

Both proposals, [BB94],[YB94], claim dramatic throughput improvement, but only end-to-end throughput comparisons are given. In the first study, throughput values are given for TCP and I-TCP. On a wide area connection, with a cell switch, the I-TCP enables 19.12 KB/s compared to a 8.89 KB/s throughput for regular TCP [BB94]. In the second study, the throughput for a wide area connection with ten percent loss is 1.8 KB/s originally and 3.6KB/s with the partitioned connection [YB94]. These are very different values and can not be fairly compared since the environment is different. Both methods show an improvement, but it is not clear how effective they would be when applied in another wireless network.

All of these TCP studies confirm the premise that the performance of TCP over wireless links is in much need of improvement. However, these studies fail to indicate which features of the protocols are responsible for the poor performance. Since these protocols are so complex, it is very difficult to determine that a given scenario will apply to a different environment.

2.3 Network File System

The Network File System (NFS) is a protocol that permits transparent, remote access to file systems in a heterogeneous network of machines [Gro89], [WLea85], [PJS+94]. NFS defines the traditional file system operations, such as reading directories and creating, deleting, writing, and reading files. Servers provide resources to the network, and clients access resources over the network. Any machine can be a client, a server, or both client and server. NFS servers are stateless, since they do not maintain contextual information about clients. The major advantage of this lack of state information is robustness against server, client, or network failures and simplification of crash recovery at the servers. For example, if a server fails, the client simply must repeat the attempt to complete the operations until the server is repaired [WLea85].

The NFS protocol is implemented using the Remote Procedure Call (RPC) protocol [Gro88]. NFS uses RPC to make procedure calls such as READ, WRITE, and GETATTR. Procedures in NFS are synchronous, which means that the client blocks until the server returns the results. Therefore, when a procedure returns, the client can assume that the operation was completed and that all data reside on stable storage.

NFS is usually implemented over the User Data Protocol (UDP), which is an unreliable transport layer protocol. Since UDP does not provide reliability, NFS must have its own policies to control acknowledgments and retransmissions. NFS can also be implemented over TCP; in which case, TCP provides the necessary features for retransmission, congestion control, and recovery from losses. Nowicki discusses the use of TCP to implement NFS and proposes that NFS improvements can be obtained through the use of better timer algorithms and transfer size adjustments [Now89]. The use of TCP may not always be beneficial since the round trip TCP timers would also be timing the service time at the file server. These service times are much less well behaved than the round-trip times of a pure transport protocol. Use of TCP to implement NFS also provides interoperability problems, since clients using TCP could not communicate with server that used only UDP and vice versa.

NFS is designed to be robust in the presence of failures but not to be efficient in the presence of errors. In wired networks, the loss of several contiguous packets usually indicates that the server is down or disconnected from the network. NFS is designed so that the client will retry until the server is again accessible. This type of server or network failure is infrequent and may take several hours to bring the server back on line. During this period NFS clients continue to retransmit file requests. Further action is effectively blocked until the file request is answered. The retransmission timers are much longer than those in TCP. After each unsuccessful retransmission, the timers increase up to a maximum of about 2 minutes. Then NFS will retry every 2 minutes until the file server is back on-line. These long retransmission periods are not ideal for losses on wireless links, where the typical disconnection period is on the order of milliseconds rather than minutes. The longer retransmission wastes seconds of clear channel transmission time. The retransmission policies of NFS are dependent on the implementation, but all versions have retransmission timers much longer than optimal for the burst errors of wireless networks.

When reading and writing large files, multiple requests must be sent to the server for a single file. To improve performance, NFS uses large packets, usually eight kilobytes, and leaves it to IP to perform fragmentation and reassembly. On wired systems, this proved to substantially increase performance [SGK+85]. Wireless systems typically have smaller maximum transmission unit (MTU) sizes than their wired counterparts [YB94], which causes fragmentation of all large read and write requests. If one fragment is lost, the entire eight kilobyte packet must be retransmitted. This increases retransmissions in wireless systems that already suffer from high loss rates.

2.4 NFS Performance on Wired Networks

We conducted experiments to determine which NFS procedures are most affected by the errors on the wireless media. The procedures executed most frequently or procedures that require the longest response time are most susceptible to losses due to burst errors. We focused our study on the most susceptible procedures. The nfswatch routine was used to record seventeen hours of NFS activity. Measurements were taken on various weekdays between the hours of 9:00 am and midnight. Less than one percent of NFS packets were lost due to buffer overflow at the monitor. Table 1 shows the top five procedures of the sixteen NFS procedures and the percentage of calls to each of these procedures.

Procedure	*Percentage called*
GETATTR	47.79
LOOKUP	30.34
READ	12.60
READDIR	6.7
WRITE	2.04
Other	0.56

Table 1: Most frequently called NFS Procedures

Eighty percent of the calls were to the LOOKUP and GETATTR procedures. These two procedures are called prior to most NFS actions. The LOOKUP procedure returns a file handle and attributes for a specified file. The GETATTR procedure returns file attributes given a file handle. The execution times for these procedures are approximately ten milliseconds. Their execution times are very brief compared to the READ and WRITE procedures. The READ and WRITE procedures are called only about fifteen percent of the time, but the duration of these calls are the longest of the NFS procedures. Most READ and WRITE packets are eight kilobytes in length. Therefore, to write a large file, a WRITE call is executed for each segment of eight kilobytes. The READ and WRITE NFS procedure calls are consequently most susceptible to losses on the wireless medium since they are of the longest duration.

3 Experimental Environment

Our experiments were conducted on a wireless LAN testbed to study the effect of wireless link error characteristics on NFS and TCP performance. The error characteristic of radio frequency links are distinct in that they are prone to burst errors. In the following, we describe the errors and error models of the wireless links and our experimental testbed. Our testbed had to be controlled to avoid interference from other Ethernet traffic.

3.1 Error Characteristics of Wireless Media

Unlike the Ethernet or ATM media which are virtually error free, wireless transmissions are prone to errors and losses due to signal interference, multipath effects, fading, and signal propagation limitations.

RF LANs have time varying error characteristics as the relative position of the communicating hosts changes. Especially at the edge of propagation range, the channel quality is highly unstable and error prone, as shown by [DR94]. The experiment conducted by Duchamp and Reynolds examined indoor radio propagation performance and determined that the region just beyond the reliable range is highly unstable and the quality of the channel varies greatly and non-linearly. When unstable conditions occurred, smaller packets were captured much more successfully than larger packets [DR94].

Mobility causes additional errors not only as hosts move out of range of the base station but also as they move into fading zones. Fading is a rapid fluctuation in the signal strength caused by multipath propagation. When the transmitted signal follows multiple paths to the receiver, the different lengths of these paths cause wave interference as the signal paths add and cancel. Interference patterns create zones in which transmission signals are weak and losses frequent. Changes in topography can cause slower changes in signal level as a result of shadowing from objects or signal interference from neighboring hosts [Fre78]. Frequency hopping wireless LANs are especially prone to burst errors as they may hop onto a frequency which is especially susceptible to interference. This interference is likely to decrease with the next frequency change [BKT95].

The performance of RF systems is limited by the frequent periods of burst losses caused by fading, shadowing, and interference. During a period of burst losses a host receives only a weak signal and all packet transmission attempts to a specific destination fail with a very high probability. These burst losses effectively create a period of disconnection from the base station. Typical observed burst periods on RF wireless LANs are in the range of 100 to 500 milliseconds [BKT95].

Several studies have developed finite state Markovian models to characterize the bit errors observed on RF channels [SF94],[WM95],[SKKF93]. In [SKKF93], the burst error channel in specified by the three state Markov model. A bit sequence is broken into guard sections and burst sections. The guard sections are error free sections of at least an arbitrarily determined length, and the burst sections are the sections between guard sections. The first state represents a guard section. The second state represents the error state. Transitions from the error state to the third state occur when error free bits are received. Transitions back to the guard state occur only after the necessary error free length has occurred. Together the second and third states represent a burst error.

The burst losses can be characterized by a two state Markov error model [BKT95]. The two states consist of a 'bad' state and a 'good' state. The bad state represents the time a channel is in burst error mode, and the good state represents the time when a channel is error free. A transfer from the good state to the bad state occurs with some probability, p1. A transfer from the bad state to the good state occurs with some probability q1. When in the good state, transmitted packets are received with a high probability. When in the bad state, the majority of transmitted packets are lost. Note that for most wireless systems a single bit error translates to the loss of an entire packet. The probabilities p1 and q1 can vary greatly between wireless systems. This approximate characterization of the wireless channel is sufficient to illustrate the effects of losses on the behavior of transport sessions.

3.2 Testbed Environment

Our experiments were run in the Mobile Computing Multimedia Laboratory (MCML). The lab network consists of two Ethernet LANs and an Infrared wireless subnet. The wireless subnet is connected to the Ethernet LAN via a base station which routes traffic to the wireless subnet. Experiments were conducted using the network monitors, tcpdump and nfswatch, running on a DEC Alpha machine. These monitors collected information about the packets on the Ethernet LAN. These monitors provided packet level information with timing accuracy on the order of 10 microseconds.

Connections were established between a mobile host on the wireless subnet and stationary hosts on the Ethernet. These connections were monitored to determine information such as packet retransmission timeouts, losses on the wireless link, and throughput. Random losses on the Infrared link at close range (5 meters) were rare. An error model was implemented at the kernel level of the wireless base station. The model gave us control over the types of losses on the wireless subnet. The error model dropped packets to create periods of burst losses on certain wireless channels. During a burst period all packets on that wireless channel were dropped. Between one and five connections were established simultaneously with the mobile host. Each connection was independently characterized with an error model.

Several types of error modes were used in our experiments. The burst periods were either of a fixed length and at fixed intervals or were at uniformly distributed intervals and of uniformly distributed lengths. The models used included the following (for the uniform distribution the mean value is given):

1. Random losses
2. 10ms Burst loss period at 200ms intervals, fixed length
3. 10ms Burst loss period at 400ms intervals, fixed length
4. 100ms Burst loss period at 1 sec intervals, fixed length
5. 100ms Burst loss period at 1 sec intervals, uniform distribution
6. 200ms Burst loss period at 4 sec intervals, uniform distribution

3.3 Measurement Validation

A number of factors had to be controlled in order to get reliable performance results. Performance between the wireless and wired subnets could be influenced by a high load on the wired subnet or by heavy loads on the source or destination machines. These factors were observed whenever measurements were taken to ensure that only minimal interference occurred.

Batched commands were used to collect data samples throughout the day to determine the amount and type of traffic on the local Ethernet network. The traffic on the Ethernet could affect the performance of the connections if the the network load was high. Even when the load was the highest, only a portion of the Ethernet capacity was being utilized. Figure 1 shows network activity at various times of day in terms of packets per second. Although the packet size can vary, this information provided a general comparison between the activity level at various times of day.

Network activity was high during the middle of the day and the night. The activity was the lowest during the early morning hours of four to eight. This was the best time to perform our measurements to avoid the high load periods which could affect our measurements by delays and interference of other Ethernet traffic.

4 Performance over Wireless Media

TCP and NFS experience similar performance degradation in the presence of errors. Since no previous studies have been conducted on NFS performance, TCP studies can be used for comparison

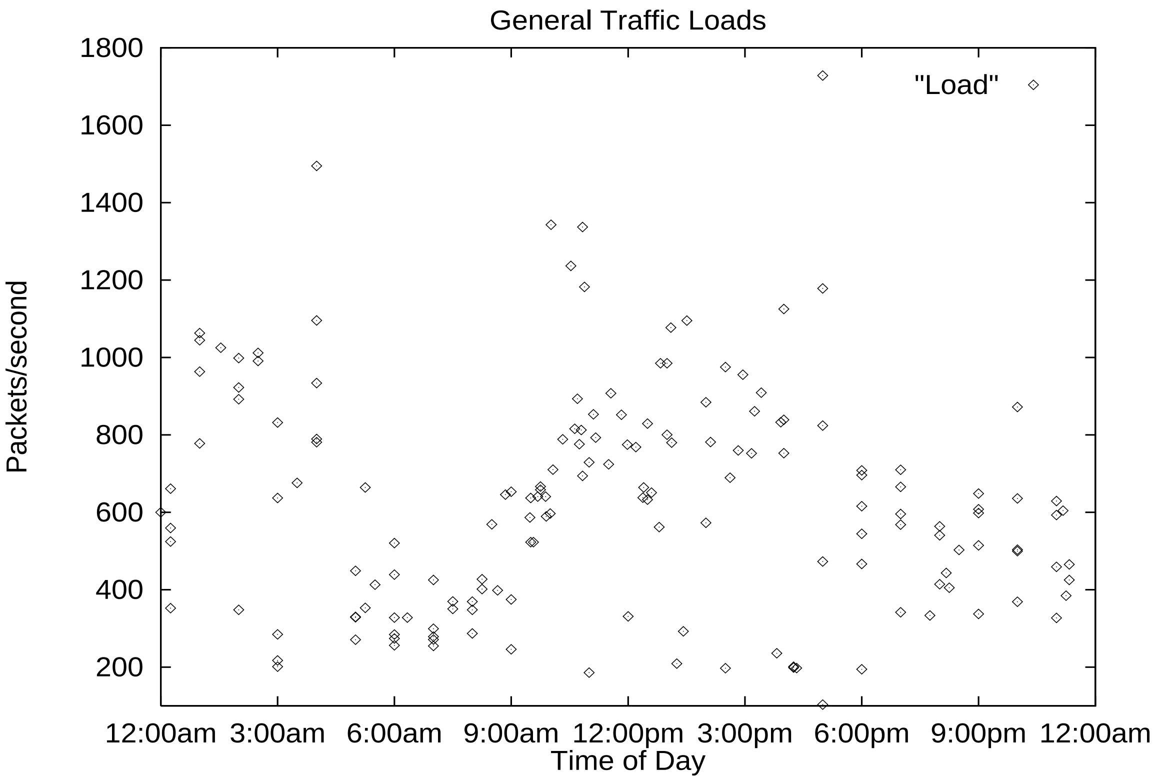

Figure 1: General traffic load on a local area network

and for insight. Most of the solutions proposed for TCP will not be applicable to NFS due to the differences in the function and policies of NFS, but comparison between retransmission policies, packets sizes, and throughput values provides a starting point for analyzing NFS performance.

4.1 TCP

4.1.1 Retransmission pauses

We have measured TCP retransmission characteristics over wireless links to determine the granularity of the idle times induced by random losses and by burst losses. Figure 2 shows a plot of sequence number versus time, in seconds, for a TCP connection from the wired to the wireless network. This graph shows the long idle periods that can result from burst losses. Here consecutive losses cause idle periods of approximately 2, 6, and 7 seconds.

Caceres and Iftode observed that mobile host cell switches caused unacceptably long pauses in TCP, between 1 and 7 seconds [CI94]. These values are almost identical to the pauses seen in figure 2. They attribute these long pauses in communication to the current retransmission and congestion control polices in TCP, and they propose a fast retransmission scheme, which notifies the transport layer of host migration and thereby reduces these pauses. Our measurements show that the same performance problems occur within a single wireless cell due to burst errors even when the mobile host is not migrating.

Figure 3 shows a detailed view of another TCP connection with idle periods due to losses and

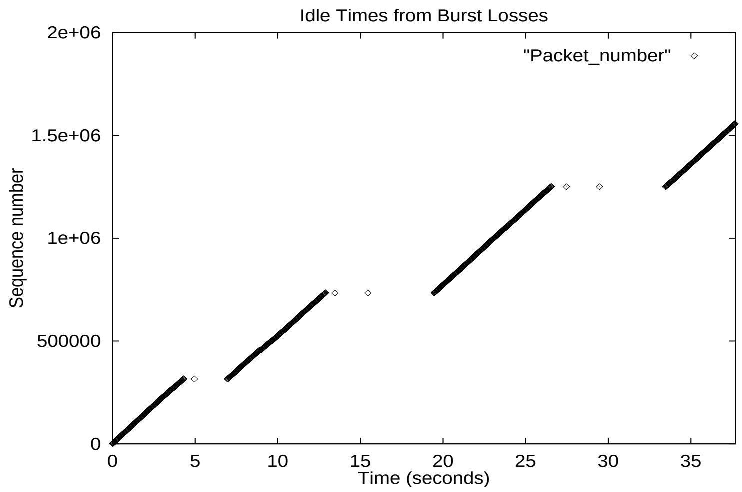

Figure 2: TCP Connection with burst losses, Throughput = 41.3 KB/s

retransmission timers. When losses occur, the backoff period is doubled after each consecutive retransmission. The idle periods between retransmissions is 0.9, 2, and 4 seconds. When losses occur in the wireless media, TCP incorrectly interprets the losses as congestion in the network and slows its transmissions. These performance measurements indicate that the majority of idle time is due to retransmission timeout periods (backoff intervals) rather than to the slow start periods. When losses occur, the backoff period is doubled after each consecutive retransmission.

In comparison to the idle time, the slow start takes very little time to return to the normal window size; it is only takes a few hundred milliseconds. Here the window size is small and there are usually only two packets outstanding. This small window size is typical of wireless networks due to the low bandwidth of the links. This observation indicates that the chief cause of performance degradation in the presence of random losses is due to the idle time while retransmissions occur.

TCP's fast retransmit mechanism was incorporated into TCP to decrease the latency of loss detection. When a threshold number (usually three) of back to back duplicate acknowledgments are received, TCP immediately retransmits the current packet instead of waiting for the timer to expire. The arrival of three duplicate acks is a strong indication that a packet loss has occured rather than just a packet delay due to congestion. This fast retransmit fails to work on wireless links, however. This is because window sizes are small (sometimes even less than three packets) and back to back packets and acks are likely to be lost during burst periods. If there are no packets ready for transmission then duplicate acks cannot be sent and fast retransmit will not be triggered.

In figure 3 the periods between the first loss and the first retransmission is after about 900 milliseconds. Typical burst errors last several hundred milliseconds. If the burst error length had been 300ms, then the first packet retransmission (after 900 ms) would be successful and we would have waited 600ms longer than necessary for retransmission. We can see from these graphs that 600 ms is enough time to send about thirty packets. This idle periods is an obvious performance degradation. The shows that TCP's retransmission timer policies are not suited to a bursty wireless link. Ideally, we would be better able to match the retransmission periods and the lengths of the typical burst periods.

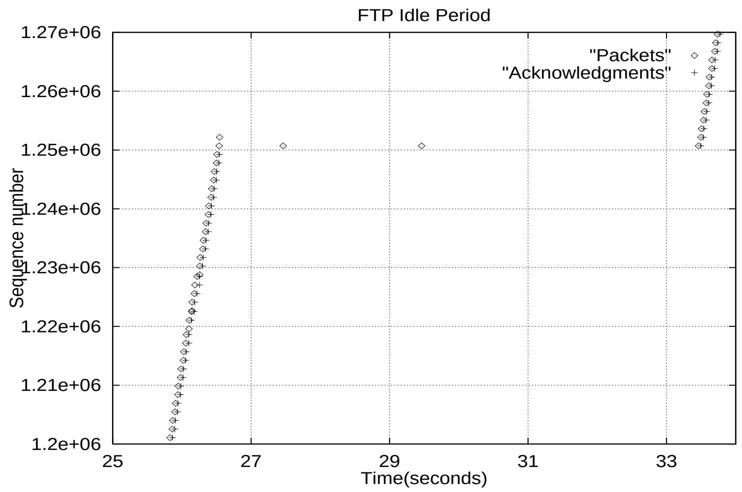

Figure 3: TCP Retransmission Pauses

4.1.2 Performance Degradation

One major difference between the transfers with burst errors and the transfers without burst errors is the consistency. The performance in the presence of even short burst errors (100ms) results is large fluctuations in throughput. Table 2 shows a clear performance degradation in the transfers that experience burst errors. The files transferred were 3.57 MB. In these experiments, the burst errors had a mean value of 100ms with a mean value of 1000ms between burst error periods. The burst periods only occur ten percent of the time, but the decrease in performance was much higher than ten percent. The average throughput with no burst errors was 54.2 KB/sec and the average throughput with burst errors was 27.7 KB/sec (as shown in table 2). This is a decrease of 50 percent in the throughput. The variance between transfer rates is caused by the different number of losses due to burst errors. Consecutive losses of a single packet cause the most degradation since the retransmission timeout doubles after each loss.

As an example of the long idle times and performance degradation suffered from burst errors, compare the graphs in figure 4. The two curves in the graph show the same 95 KB file transfer first with only random errors present and then with burst errors. The first curve shows a transmission that experiences no burst error period. Two packets are lost due to random errors. The throughput of this transfer is 15.4 KB/sec. The second curve shows a transfer that experiences 10ms burst errors every 200ms. This transfer of the same file takes almost three times as long, 15 packets are lost due to burst errors. The throughput is only 6.3 KB/sec, less than half of the throughput of the non-burst error case.

	No Errors	*100ms Burst Errors*
	58	46
	59.7	36
	60	32
	50	5.3
	50.5	26
	47	18
Average	54.2	27.2

Table 2: Throughput Values, Kilobytes/second

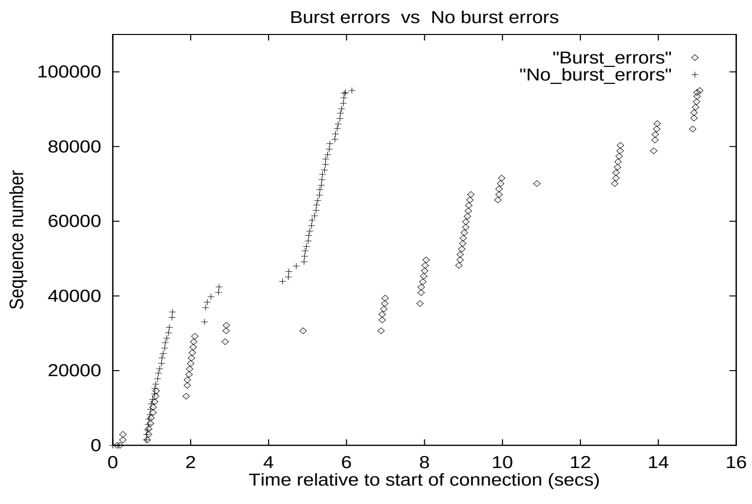

Figure 4: Affect of Burst Errors on Throughput

4.2 NFS Performance on Wireless Networks

4.2.1 Throughput Degradation

Our experiments have shown the penalty inflicted on TCP by burst errors: performance degradation and inconsistency. Similarly, burst errors cause significant degradation to the performance of NFS over wireless links. When we compare the performance of NFS in the presence of burst errors, random errors, and no errors, the results show the burst errors have the most detrimental effect on the performance of NFS. Table 3 and figures 5 and 6 compare NFS reads and writes in the presence of three different error models:

1. No burst errors

2. 100ms burst errors, with 1 sec good periods (fixed length)

3. 100ms mean burst errors, with 1 sec mean good periods (uniform distribution)

As table 3 shows, the performance is best for the no error situation and worst with the uniformly distributed burst errors. This table shows average values that were calculated based on repeated writes and reads. Figure 5 compares write performance for these three models and figure 6 compares read performance.

Error Model	*NFS write*	*NFS read*
No burst errors	25.2 KB/s	94.2 KB/s
Fixed Length	19.6 KB/s	38.7 KB/s
Uniform Distribution	16.3 KB/s	21.6 KB/s

Table 3: Performance of NFS with burst errors

Figures 5 and 6 show that the fixed length burst error model and the uniformly distributed error model cause similar effects to NFS performance. Transfers during either of these burst error models have consistently lower throughput values than transfer without burst errors. The throughput values for non-burst error cases are 24.5 KB/s for the write and 94.3 KB/s for the read. The large difference in these two values are due to the longer write response times and also to several random errors in the NFS write transfer. For the writes the throughput values are 20.7 KB/s and 18.7 KB/s for the fixed and uniformly distributed burst lengths. For reads the throughput values are 38.3 KB/s and 21.6 KB/s. The uniform distribution model has even worse performance than the fixed model, but both suffer from decreased performance. This indicates that a simplistic fixed length burst error model can provide realistic insight into the behavior of NFS with burst errors as well as a distributed model. The simpler fixed length model can be used to gain understanding of the causes of NFS performance degradation.

Our data allows us to investigate the reasons for the performance degradation in the presence of burst errors. In the remainder of this section we examine the features of NFS which contribute to this inferior performance.

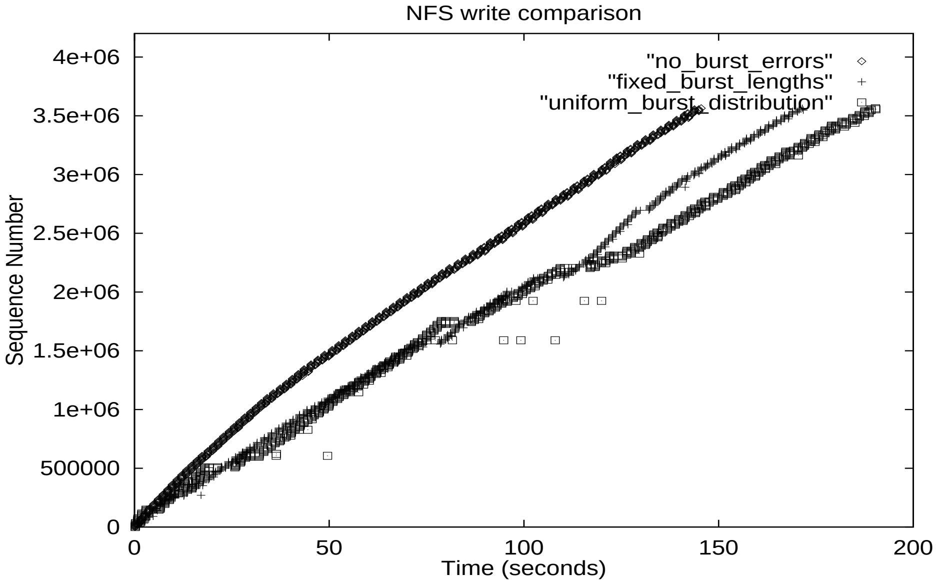

Figure 5: NFS write with and without burst errors

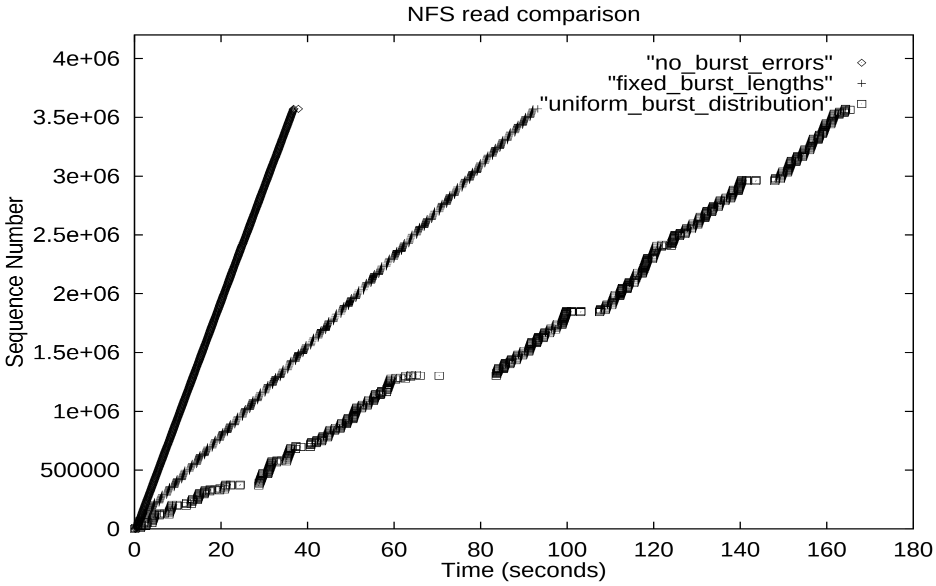

Figure 6: NFS read with and without burst errors

4.2.2 Retransmission Pauses

In an environment where losses are frequent, the retransmission policy has a significant impact on the system performance. Figure 7 illustrates NFS response to burst losses. Here long burst periods cause consecutive losses and decrease throughput. The throughput is 27.7 KB/s, and this is much less than the 94.4 KB/s throughput that is achieved when no errors occur. Table 4 shows the increasing retransmission timeout values. The retransmission timeouts increase to a maximum of 90 seconds. The graph in figure 8 also shows the increasing retransmission timeouts. The idle period prior to the first retransmission is 1.1 seconds, and by the fourth retransmission the timeout has increased to about 13 seconds. This demonstrates the long timeout periods in NFS.

Table 4: NFS Retransmission Intervals, seconds

1.1	2.2	4.4	13.1	4.4	8.8	17.6	52.8	17.6	35.2	60	90	30	60	90

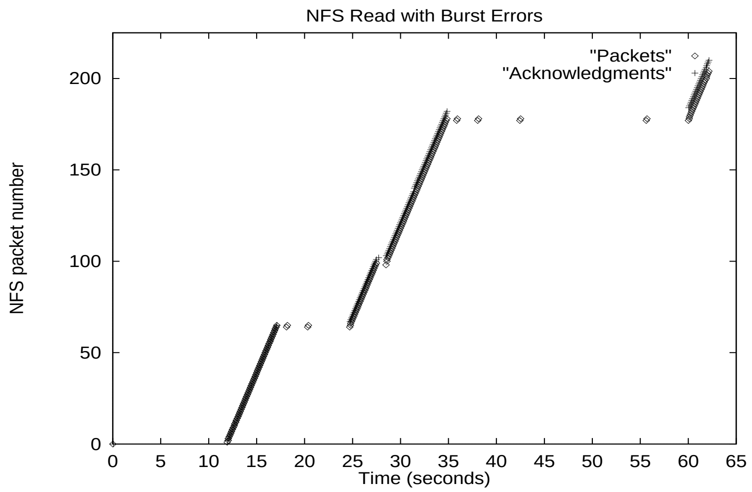

Figure 7: NFS response to losses, NFS read

NFS has been designed for use in the wired environment where disconnections are expected to be rare and lengthy. The burst periods in wireless systems will usually be on the order of 100 milliseconds. This means that any loss will create an idle period of at least one second, 10 times longer than necessary for a 100 millisecond burst period. Methods compatible with the existing structure are needed to ameliorate this effect on NFS performance.

4.2.3 NFS Procedures

If a NFS request is lost, the client retries after a certain timeout period. Most client requests are very quick, such as the GETATTR, LOOKUP, and READLINK procedures which usually take about ten milliseconds. Other procedure calls, such as READDIR, vary depending on the file involved and take about fifteen to twenty milliseconds. The READ and WRITE procedures take the longest and their return times vary the most. In one file copy experiment, READ requests took between 40 and

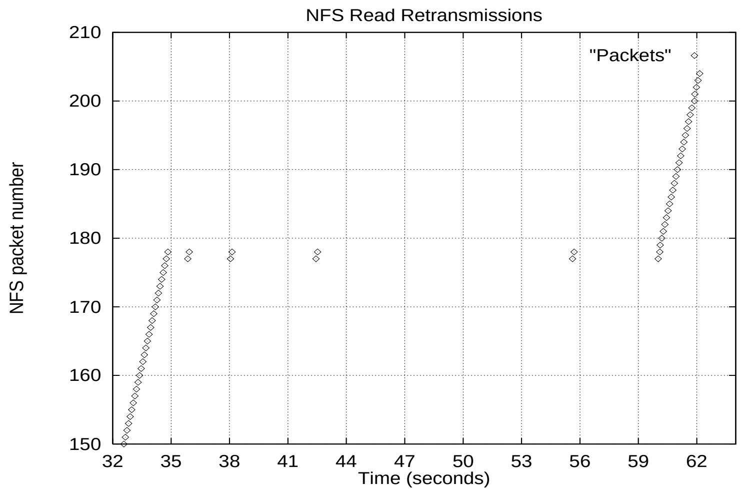

Figure 8: NFS Retransmission Pauses

150 milliseconds to complete, with an average of 80 milliseconds. READ requests were consistently shorter than WRITE requests, which took between 100 and 900 milliseconds, with an average of 550 milliseconds.

Even in the absence of errors, writes were much slower than reads for the same file. A WRITE request is not acknowledged until the request is written onto stable storage. This is one reason for the longer WRITE reply times. When no errors occur, a sequence of five WRITE requests take about one second, and the corresponding sequence of READ requests take about 450 milliseconds, less than half the time. Consecutive READ requests are transmitted every 15 milliseconds and usually the acknowledgment is received prior to the next request. WRITE requests are sent in groups of five requests at 20 millisecond intervals. This means that if a 200ms burst period occurs it is likely that all five WRITE requests will be lost. Thus write losses occur in groups, as seen in figure 9 where five packets are retransmitted and then four of those are transmitted yet again. The throughput is obviously greatly reduced; in this case throughput is 4.4 KB/s instead of the 27 KB/s that is attained when no burst errors occur.

When a packet is lost, the client retransmits the request after 1.1 seconds. This idle period, while the client waits for the server response can dramatically increase the amount of time required to complete an application program request. For example, if a LOOKUP request is lost, it will take 1100 milliseconds instead of 10 milliseconds for the request to complete. This difference would be hardly noticeable unless a high percentage of the packets were lost. During burst periods these timeout periods could have a dramatic effect on performance.

4.2.4 Large Packet Size

NFS reads and writes are most vulnerable to performance degradation from the burst errors of wireless links. These two procedures transfer the bulk of the data. The typical packet size is eight kilobytes. This large packet size requires fragmentation which makes the packet more prone to losses.

To investigate the effect of the large NFS packet size in the presence of errors, we transferred files using both NFS and TCP's FTP (file transfer protocol). By comparing the write and reads of both of these protocols, we determined that the large packet size can decrease performance over error-prone wireless links. Table 5 shows throughput values for NFS and TCP reads and writes. This table shows that, when burst losses occur, FTP performance is about twice as good as NFS performance.

In table 5 the NFS write throughput is about half of NFS read throughput. This is due to the longer write reply times and the vulnerability of write transfers to multiple consecutive packet losses during burst periods, as explained in the previous section. The FTP read values and FTP write values are different because the data is sent *from* the slower mobile host in the read case and *to* the mobile host in the write case.

Function	Packet Size	Throughput	Average Throughput
NFS write	8 KB	5.6 KB/sec	
	8 KB	4.7 KB/sec	
	8 KB	0.98 KB/sec	
	8 KB	2.74 KB/sec	
	8 KB	1.44 KB/sec	3.1 KB/sec
FTP write	0.5 KB	3.27 KB/sec	
	0.5 KB	1.7 KB/sec	
	0.5 KB	2.7 KB/sec	
	0.5 KB	5.6 KB/sec	
	0.5 KB	11.9 KB/sec	5.0 KB/sec
NFS read	8 KB	1.70 KB/sec	1.5 KB/sec
	8 KB	1.30 KB/sec	
FTP read	0.5 KB	1.63 KB/sec	
	0.5 KB	3.3 KB/sec	
	0.5 KB	3.9 KB/sec	
	0.5 KB	0.43 KB/sec	2.3 KB/sec

Table 5: Throughput and Packet Size

This data is not conclusive, but indicates that NFS does not perform better than an FTP file transfer as would be expected. This is attributed to both large NFS packet size and longer NFS retransmission timeout values. The traces of NFS performance (figure 9) show that NFS write is often very inefficient in the retransmission of packets. Many more packets are retransmitted in the NFS transfers than in the FTP transfers. The NFS Write in figure 9 shows almost 100 percent packet retransmission, and the FTP write in figure 10 has less than ten percent retransmission. For the same size files there are 5 times less NFS packets. Thus if one packet in NFS is lost eight Kilobytes must be retransmitted, compared to only 512 bytes for a lost FTP packet. This performance difference between NFS and FTP writes is due, in part, to the fact that a loss of a single fragment causes retransmission of the entire eight kilobyte packet. It is also due to speed inequities between the server and the client, as we discuss in the following section.

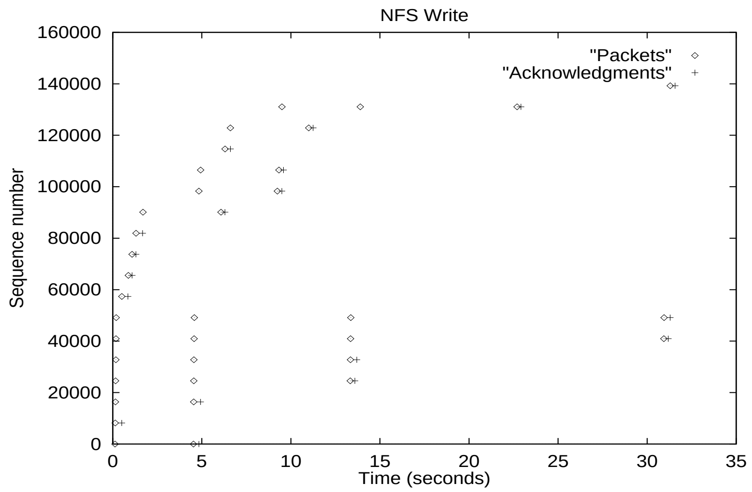

Figure 9: NFS WRITE, Throughput = 4.41 KB/s

4.2.5 Unbalanced Server and Client

If there exists a great disparity between the speed of the source and the destination machines in any data transfer, there is the danger of the faster machines overwhelming the receiver with data. When the source is faster than the receiver, it often means that entire windows of data are sent before the receiver can respond. A wired TCP session between a much faster sender and a slow host is displayed in figure 11. An entire window of packets is sent before the receiver sends the first acknowledgments. In this case, there is one acknowledgment sent for each group of packets. As soon as the sender receives an ack, it sends another entire window of packets in rapid succession.

As an example of this phenomena we compared the transfer of a file (using NFS write) between two stationary hosts of comparable speed and then a transfer between a fast and a slow host. The throughput varied greatly. In the first case, the throughput was 85.3 KB/sec. In the latter case, the performance was only 4.4 KB/sec. Figure 12 shows the second, unbalanced hosts case. This transfer was via the Ethernet so the losses in figure 12 were unrelated to the wireless media. They were caused by the lost packets a the slower receiver losing packets since the receiver is too slow to process all of the write requests. From this we determine that the rate at which NFS read and write requests are sent by the client is of central importance.

In our environment, our NFS server is more powerful than our mobile host. In the error-prone wireless environment, that often means that entire sequence of data packets are retransmitted due to a single lost packet at the beginning of the sequence of packets. This dramatically reduces the performance of a transfer, but could be improved by optimizing the patterns of read and write requests sent by the mobile host.

4.2.6 Performance Improvement

Our experiments with NFS over the wireless link clearly displayed a dramatic decrease in performance when burst errors were introduced. In the first case we introduced no errors into the wireless link and we read a file of size 3.57 MB. The throughput for this read was 94.4 KB/s. We tried the same experiment and introduced a randomly distributed burst error model. In this model, there

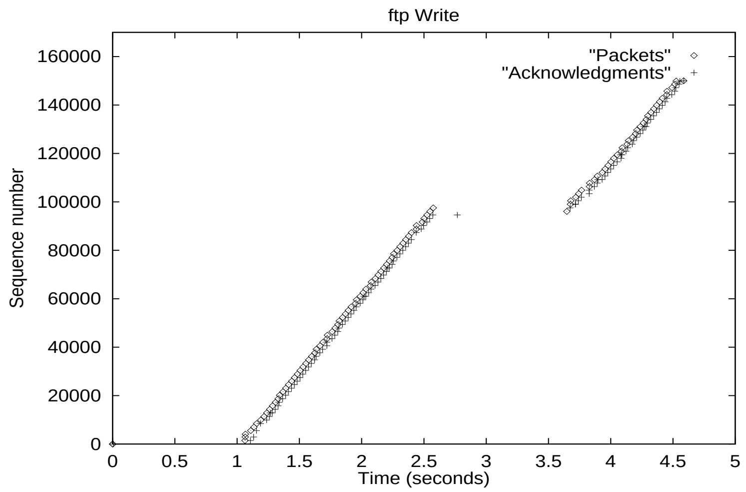

Figure 10: FTP write, Throughput = 32.7 KB/s

are 'good' periods with a mean of 1000ms (with no losses) and alternating 'bad' periods with a mean of 100ms. With even these short burst periods the performance was severely reduced. The same read request only achieved 22 KB/s. Thus burst periods occurring 10 percent of the time caused performance degradation of 75 percent. In addition, the burst errors caused high variability in the NFS performance.

The performance of NFS over wireless networks clearly requires much improvement. The methods for achieving this improvement are not as obvious. The methods suggested for TCP performance improvement can be considered, but, in fact, none of these methods will be effective for NFS.

The connection partitioning solution, [BB94],[YB94], cannot be easily used with NFS for several reasons. Since NFS modifies files systems, consistency is of central importance. In the partitioned connection solution, reliability cannot be guaranteed since the mobile host's base station sends acknowledgments to the source before the mobile host has received the packets. Migration or buffer overflow can introduce consistency problems. If the base station is required to forward these packets after a mobile host moves, no reliability exists in the case of a base station failure. TCP relies on the higher layer, the application, to catch any packets that are lost due to these problems. NFS, however, is an application and can not depend on any higher level to catch consistency problems that arise.

For consistency reasons, it is important that the acknowledgments be sent to the client only after successful completion of the client's request. This prevents the base station from acknowledging packets for the mobile host. Instead it is possible to use a reliable link layer to improve the performance of the wireless link without retransmitting wireless losses over the wired portion of the connection. A reliable link layer protocol is used on many wireless LANs. Link layer acknowledgments are sent by the receiver, and the transmitter will retry several times if no link level ack is received. In the presence of burst errors, however, successive retransmissions are likely to fail. A channel dependent retransmission policy, as suggested in [BKT95], would handle this problem by deferring link level retransmissions until it was probable that the burst period had ended.

NFS performance is harmed by both large packet size and by the retransmission policy. Although the NFS server should not be modified, it is desirable to change the policies of the mobile

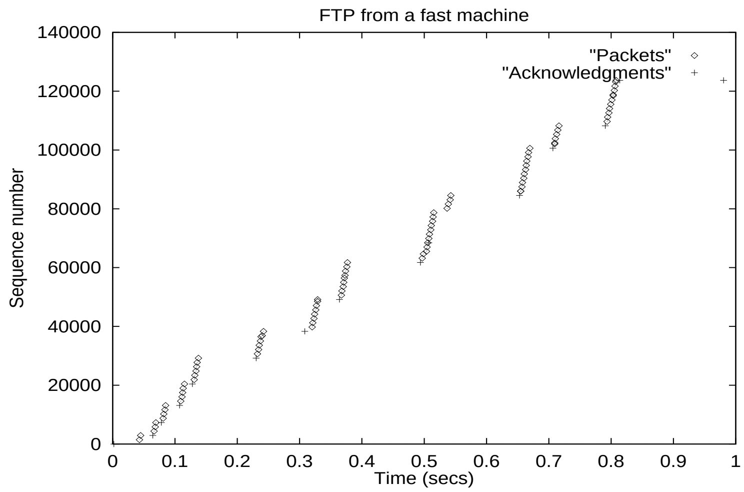

Figure 11: TCP session from a faster machine, Throughput = 114.5 KB/s

client. Fragmentation is unavoidable when using large packet sizes, such as those used in NFS. Large packets and the associated fragmentation are preferable to the overhead of smaller packets when transmitting over wired networks. Over wireless links the large packet size can be detrimental due to packet losses and fragmentation. The NFS packets will often have to be fragmented several times before they are send over the wired and then the wireless networks.

Performance may be improved by performing reassembly at the base station rather than end-to-end reassembly, but this would be complicated to implement and add overhead. Since performance is better on the wired link for larger packet sizes, the base station could reassemble the mobile host's small packets. Performance may also be improved by optimizing the NFS retransmission timers on the mobile host. This would decrease the idle time when a packet is lost. It would be helpful for the mobile host to be able to identify between losses, delays due to migration, and NFS server failures in order to determine the most effective retransmission policy. If a burst period has caused losses, then the most effective retransmission times would be on the order of several hundred milliseconds. If losses were due to a server failure, retransmission times should be much longer, increasing from an initial value of several seconds. If the delays and losses are due to host motion, the retransmission times should be in between these two ranges, on the order of seconds. Caceres and Iftode [CI94] propose notification of host motion to the transport layer, which might also help determine the appropriate retransmission policy in this case. Better link level technologies are needed to enable effective interpretation of wireless link losses.

Alternative solutions for improving NFS performance include caching files on the mobile host or intelligent file prefetching. These methods have been suggested by those developing file systems for mobile hosts [HKM+88],[Sat90],[SKK+90], and mobile clients should be developed with these intelligent file handling capabilities. The ability to prefetch and cache frequently needed files in essential to attain acceptable application performance for mobile hosts.

Future research will investigate these methods for improving NFS performance over wireless links. The most promising solutions address the problem at the source: either at the error prone link level with link level retransmission or at the NFS level with modified NFS retransmission policies and disk caching on the mobile client.

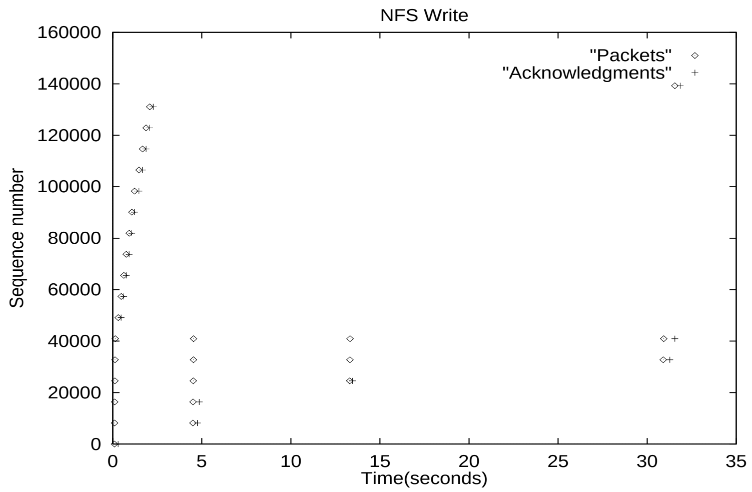

Figure 12: Transfer to a much slower machine, Throughput = 4.4 KB/s

5 Conclusion

The current Mobile IP protocol provides the functionality to allow a mobile user to migrate through a network. Performance on wireless systems suffers from lower bandwidth and bursty loss patterns. Propagation techniques and link level protocols are being developed to alleviate some of these limitations. Due to these current limitations, many of the protocols developed for use in the wired environment must be modified to work efficiently in the wireless domain.

TCP and NFS both suffer from long idle periods due to losses and interference. These protocols must be improved to allow satisfactory communication with the network backbone. Out experiments have provided insight into the complexities of these two protocols in the presence of various types of error patterns. Further research will be conducted to devise solutions to the performance problems investigated in this paper. Link level solutions, such as Channel State Dependent Packet Scheduling [BKT95], require added functions in the link layer transmitters. Successful link level solutions have the advantage of improving the performance of both TCP and NFS, as well as other applications. NFS performance can be improved by modifying the file system behavior of the mobile host. Adding simple caching and pre-fetching options or adding base station buffering are the most straightforward methods for improving the slow performance of NFS over wireless links. There are many issues to be resolved before wireless systems will achieve the goal of performance transparency.

References

[BB94] Ajay Bakre and B.R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. Technical report, Rutgers University, October 1994. DCS-TR-314.

[BKT95] P. Bhagwat, A. Krishna, and S. K. Tripathi. Using Channel State Dependent Packet (CSDP) Scheduling to Improve Throughput Over Wireless LANs. Technical Report RC 20093, IBM T. J. Watson Research Center, 1995.

[CI94] Ramon Caceres and Liviu Iftode. The Effects of Mobility on Reliable Transport Protocols. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, June 1994.

[DCY93] Antonio DeSimone, Mooi Choo Chuah, and On-Ching Yue. Throughput Performance of Transport-Layer Protocols over Wireless LANs. In *Proceedings of the Conference on Global Communications (IEEE GLOBECM)*, pages pages 542–549, 1993.

[DR94] Dan Duchamp and Neil F. Reynolds. Measured Performance of a Wireless LAN. In *Proceedings of the 17th Conference on Local Computer Networks*, pages p. 494–499, September 1994.

[Fre78] Ricard C. French. Error Data Predictions and Measurements in the Mobile Radio Data Channel. *IEEE Transactions on Vehicular Technology*, 27(3):p. 110–116, August 1978.

[Gro88] Network Working Group. RPC: Remote Procedure Call Protocol Specification. RFC 1057, June 1988.

[Gro89] Network Working Group. NFS: Network File System Protocol Specification. RFC 1094, 1989.

[HKM+88] J.H. Howard, M.L. Kazar, S.G. Menzees, D.A. Nichols, M. Satyanaraynan, R.N Sidebotham, and M.J. West. Scale and Performance in a Distributed File System. *ACM TOCS*, 6(1):51 – 81, 1988.

[IDJ93] John Ioannidis, Dan Duchamp, and Gerald Q. Maguire Jr. IP-based Protocols for Mobile Internetworking. In *Proceedings of Winter USENIX*, San Diego,CA, January 1993.

[Now89] Bill Nowicki. Transport Issues in the Network File System. *ACM Communication Review*, 19(2):pp 16–20, 1989.

[PB94] Charles Perkins and Pravin Bhagwat. A Mobile Networking System Based on the Internet Protocol. *IEEE Personal Communications Magazine*, 1(1), 1994.

[PJS+94] Pawlowski, C. Juszczak, P. Staubach, C. Smith, D. Lebel, and D. Hitz. NFS Version 3: Design and Implementation. In *Proceedings of Summer USENIX*, Boston, Massachusetts, 1994.

[Rek94] Yakov Rekhter. An Architecture for Transport Layer Transparent Support for Mobility. *Journal of High Speed Networking*, pages p 6–17, January 1994.

[Sat90] M. Satyanaraynan. Scalable, Secure and Highly Available Distributed File Access. *IEEE Computer*, 23(5):9 – 21, 1990.

[SF94] F. Swarts and H.C. Ferreira. Markov Characterization of Digital Fading Mobile VHF Channels. *IEEE Transactions on Vehicular Technology*, pages pages 977–985, November 1994.

[SGK+85] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and Implementation of the Sun Network Filesystem. In *Proceedings of Summer USENIX*, Portland, Oregon, 1985.

[Sim94] W.A. Simpson. IP Mobility Support. IETF Network Working Group, Internet Draft, available via ftp from the internet archives, September 1994.

[SKK+90] M. Satyanaraynan, J.J. Kistler, P. Kumar, M.E. Okasaki, E.H. Siegel, and D.C. Steere. Coda : A Highly Available File system for a Distributed Workstation Environment. *IEEE Transactions on Computers*, 39(4):447 – 459, 1990.

[SKKF93] Takuro Sato, Manabu Kawabe, Toshio Kato, and Atsushi Fukasawa. Throughput Analysis Method for Hybrid ARG Schemes Over Burst Error Channels. *IEEE Transactions on Vehicular Technology*, 42(1), February 1993.

[TT93] Fumio Teraoka and Mario Tokoro. Host Migration Transparency in IP Networks. *Computer Communication Review*, January 1993.

[WLea85] Dan Walsh, Bob Lyon, and Gary Sager et. al. Overview of the Sun Network File System. In *Proceedings of Winter USENIX*, pages p. 117–124, January 1985.

[WM95] H.S. Wang and N. Moayeri. Finite State Markov Channel - A Useful Model For Radio Communication Channels. *IEEE Transactions on Vehicular Technology*, pages pages 163–171, february 1995.

[WYOT93] Hiromi Wada, Takashi Yozawa, Tatsuya Ohnishi, and Yasunori Tanaka. Mobile Computing Environment Based on Internet Packet Forwarding. In *Proceedings of Winter USENIX*, San Diego, CA, January 1993.

[YB94] Raj Yavatkar and Namrata Bhagawat. Improving End-to-End Performance of TCP over Mobile Internetworks. In *Workshop on Mobile Computing Systems and Applications*, pages p 146–152, Santa Cruz, California, December 1994.