Performance of Transmission Rate Changing in Ad-Hoc Networks

Mark Zalar

May 2, 2007

MS Project Report

Advised by Dr. Daniel Mossé

1 Introduction and Project Overview

The lifetime of wireless ad-hoc networks is often limited by the battery life of devices in those networks. Work in [1] proposed a method of dynamic rate selection to conserve power and increase the lifetime of such a network. Dynamic rate selection works by dynamically selecting transmission rates for each node in a given path based the node's available battery power. Transmitting at a lower rate consumes less power, but causes data to travel more slowly. An implementation of the dynamic rate selection algorithm in the ns-2 network simulator yielded promising results. Networks using the algorithm were shown to last 10 times longer than networks with no power management. In addition, the dynamic rate selection algorithm enabled 7.5 times more packets to be delivered.

The purpose of this project was to take the work done in [1] and create a real implementation using mobile devices. Upon successful completion of this implementation, its performance would be compared with the results found in [1]. After running some preliminary experiments, it became apparent that the performance of the hardware was poorer than expected. Furthermore, inconsistency was seen in the hardware's behavior during different experiments. Thus, work on the project moved in the direction of quantifying the observed performance issues.

Although dynamic rate selection was not implemented in the end, time was spent looking into DSR [2] [3] as a routing protocol. DSR was chosen primarily because it was used in the simulation in [1].

The next section details the equipment used and the setup of the experiments for this project. Section 3 describes the experiments that were performed and summarizes the results of those experiments. Section 4 lists conclusions that have been drawn from the experiments' results.

2 Equipment and Setup

2.1 Equipment

Equipment used in this work included four laptops, three wireless cards, a PDA, a PCMCIA CompactFlash adapter, and a power measurement device. The laptops consisted of a Dell Latitude C510/C610 running Red Hat Enterprise Linux 4.0, a Dell Inspiron 4100 running Red Hat Enterprise Linux 4.0, a Toshiba 1200-S121 running Red Hat Enterprise Linux 4.0, and a

Dell Inspiron 8200 running a SLAX 5.0.6 Standard Edition LiveCD. The operating systems on all three of the laptops running Red Hat Enterprise Linux 4.0 were identical installs performed by the Computer Science Department's tech staff and were running Linux kernel 2.6.9-42.0.3.EL. The SLAX LiveCD had kernel 2.6.12.2. Each of the three wireless cards used were CompactFlash cards including two Linksys WCF12 cards and one D-Link DCF-660W. All three wireless cards were 802.11b compliant and capable of transmitting at four data rates: 1, 2, 5.5, and 11 Mbps. A SanDisk CompactFlash PCMCIA adapter was used to allow the laptops to use the wireless cards. A Sharp Zaurus SL-5500 PDA was used running OpenZaurus 3.5.4 with Linux kernel 2.4.18. Finally, a Seasonic Power Angel power monitor was used to measure power consumption.

2.2 Setup

The setup for the experimentation consisted of a small ad-hoc wireless network between one of the laptops and the PDA. In all cases, the laptop acted as the sender, and the PDA acted as the receiver. The CompactFlash wireless cards described above were used at the sender and receiver in all cases. The particular card used at the sender and receiver was varied for some experiments. This setup was originally intended to act as a preliminary setup, which would eventually migrate to an all-PDA ad-hoc wireless network. Observed inconsistency and worse-than-expected performance in the initial setup, however, prompted more experimentation using this setup. In the end, this setup was the only one used throughout this work.

3 Experiments and Results

Several different experiments were run to gather information for this project. The first experiment was a simple ping test to observe rate changing performance. Next, a series of UDP experiments were used to observe rate changing performance more accurately by measuring transmission time in one direction instead of roundtrip. Another UDP experiment was performed to determine how long rate changes take when done between every packet sent. Finally, a series of power measurement tests were performed to approximate how much power is consumed while transmitting data at different rates and changing rates.

3.1 Ping

The first experiment performed was designed to test the rate-changing ability of the wireless cards informally. The experiment consisted of repeatedly pinging the PDA with the laptop while periodically issuing rate change commands. The ping command included the option of a 512-byte packet size, and the pings occurred once per second. A Perl script was used to issue the ping command and to timestamp each response. The resulting response times and timestamps were recorded and plotted. Rate change requests were issued via a separate Perl script that was run alongside the ping command Perl script. The Perl script that issued rate change commands attempted to change the transmission rate of the laptop's wireless card by running iwconfig every 20 seconds for two minutes.

The experiment was performed three times, twice with the Dell Latitude laptop and once with the Toshiba 1200-S121. Each time the experiment was performed in the same location using roughly the same placement of the equipment. All of the ping experiments were done using the same two wireless cards. One of the Linksys WCF12 cards was used with the laptop and the D-Link DCF-660W was used with the PDA. These cards were used in this combination because Red Had Enterprise Linux 4.0 supported the Linksys cards by default. The D-Link card was used with the PDA because, during informal tests performed while setting up the experiments, the Linksys cards seemed to communicate more consistently with the D-Link card than each other.

3.2 Ping Results

Initial results from this experiment were less than promising. There was hardly any variation between the ping response times, and the variation that was present had absolutely no correlation to the reported transmission rates. In addition, the rate change requests reportedly failed roughly 17% of the time. This means that upon querying the device after issuing a rate change command, 17% of the time it reported that it was transmitting at a rate other than the one specified. Upon repeating this experiment on the same equipment, the only change seen in the results was that the wireless card never reported a transmission rate other than the one specified. There was, however, no indication from the response times that the transmission rate ever actually changed.

The same experiment was repeated on the Toshiba 1200-S121. Results from this experiment were slightly more promising, but still poorer than expected. Rate change requests reportedly succeeded 100% of the time, and some correlation was seen between the reported transmission rate and the ping response times. This means that, in some cases, the ping response times noticeably varied with the transmission rates. Ping response times were approximately 12 milliseconds while transmitting at 1Mbps and approximately 8 milliseconds while transmitting at 2Mbps. These response time variations were not seen for all transmission rates, but were noticeable nonetheless. The importance of the observed correlations was that they indicated rate changes were actually occurring, not just being reported.

Results of the ping experiments, overall, indicated inconsistent hardware behavior. Although the experiment with the Toshiba 1200-S121 laptop showed a noticeable correlation between the ping response times and the transmission rates, the fact that this correlation was not seen for all transmission rates means that the performance was not consistent. Similarly, the two experiments with the Dell Latitude showed inconsistent reporting of transmission rate by the wireless card.

3.3 UDP

After seeing results with the Toshiba laptop, a new experiment was devised to test the rate changing capabilities of the wireless cards more formally and precisely. Since pinging measures roundtrip time, every reported response factored in time that was undesirable. To determine whether a packet is transmitted at a certain transmission rate, it only matters how long it takes the packet to reach its destination. The time a response takes to return to the sender is irrelevant. Thus, the new experiment was designed to measure transmission time in only one direction.

Using UDP, the laptop would repeatedly send the PDA 1000-byte packets containing a sequence number and a timestamp. When the PDA received a packet, it would read the sequence number and timestamp. The sequence number was used to determine if any packets had been dropped since the last one received. By subtracting the packet's timestamp from the current time, the PDA could determine roughly how long it took the packet to reach its destination. The PDA would record and store the sequence numbers of the packets it received along with the timestamps and the computed travel time.

At the laptop, the Perl script used to change rates in the ping experiments was used again here to issue rate change commands and to record the reported transmission rates of the laptop's wireless card. In place of the Perl scrip used to issue the ping command, a C program was written to send 1000-byte packets over a UDP connection as described above. Like the ping experiments, packets were transmitted at one-second intervals. In this case, however, the rate was changed every 200 seconds. The reason the rate was changed every 200 seconds instead of every 20 seconds as in the ping experiments was to address speculation that the poor and inconsistent performance seen in the ping experiments was due, in part, to the fact that the rate was changed too often. Allowing more time between rate changes gave the sender and receiver more time to stabilize at each rate. Most of the UDP experiment consisted of sending either 3200 or 3600 packets. Longer experiments, consisting of sending 16000 packets, were also conducted.

These experiments resulted in the first clear distinction between the different transmission rates, but the results of the experiments were still far from perfect. Due to a variety of inconsistencies and poor performance, the UDP experiment was repeated 26 times (22 short experiments and 4 long experiments) using several different laptops and combinations of wireless cards. Because of consistently poor results with the Dell Latitude, laptop use shifted to primarily the two Dell Inspiron laptops.

3.3.1 Clock Skew

One interesting issue with the UDP experiments was computing the time it took each packet to reach its destination. As mentioned earlier, this computation was done by subtracting each packet's timestamp from the time that the packet arrived at the receiver. The sender and receiver's clocks were not synchronized, so the difference between packet timestamp and arrival time was usually much larger or smaller than expected. In addition, plotting these values revealed an interesting property in that all the plots were sloped, indicating that packets were being received consistently and increasingly close together or far apart. The slopes were not related to the variation seen that corresponded to the varying transmission rates. In addition, the slopes were different for each sender-receiver combination, which indicated that they were hardware related. Since the transmission of packets consistently occurred at one-second intervals, the packets should not have been received consistently and increasingly close together or far apart. Thus, the slopes were attributed to clock skew between the sender and receiver.

To remedy the clock skew, the slopes of the plots for each sender-receiver pair was approximated. Next, the product of the slope and packet number was subtracted from each plotted data value. Then, each data point was adjusted by the difference between the sender and receiver clocks to compensate for the lack of synchronization. While not a perfect solution, these adjustments produced much more readable graphs (see Figures 1 through 5 discussed later). Since the skew was directly related to the hardware used, the slope of each plot for experiments where the same sender and receiver were used was roughly the same. Thus, one good approximation of the skew between the two clocks allowed easy adjustment of all plots for experiments using that equipment pair.

3.4 UDP Results

Perhaps the most interesting results came from the UDP experiment. This experiment was repeated more than any other experiment. Over the course of running this experiment, several noticeable trends emerged.

Similar to the inconsistencies seen in the results of the ping experiments, there were instances where the exact same hardware configuration would fail to change rates during one run of the experiment, and change rates perfectly during another run. This was noticeable when the runs were plotted. When the sender failed to change rates, there was no variation in the time differences between the send and receive times. When the sender successfully changed rates, however, there was clear variation in the time differences between the send and receive times that perfectly corresponded to the rate changes. In some of the cases where the laptop failed to change rates, the wireless card reported most or all rate changes as successful. In other instances of failed rate changing, the wireless card accurately reported that it failed to change rates. The behavior seemed as if the senders lost their settings in some instances and maintained them in others. One possible, but unverified, explanation for this behavior is that large amounts of wireless interference were present at the times when the sender failed to change rates, preventing it from operating properly. Whether or not this was the cause, these results indicated that the ability to change rates was not reliable.

As mentioned in Section 2.4, several UDP experiments were done where 16000 packets were transmitted. In all cases, when this longer version of the experiment was run, the final 2000 packets of the experiment were dropped. In some cases, the number of dropped packets at the end of the experiment was even higher. This means that, at best, none of the long experiments successfully transmitted any packets during the final half hour of the experiment. In one instance, all packets after packet 6422 were dropped. These results indicate that this experiment is not sustainable for prolonged periods. This behavior also seemed as if the senders lost their settings and were unable to reestablish them. Perhaps the uncommon nature of the experiment, prolonged ad-hoc communication with sender-specified transmission rates, was difficult for the devices being used to maintain. Although not verified, this observed behavior may have also been caused by large amounts of wireless interference.

Another interesting trend observed from the UDP experiment results was that the number of dropped packets varied significantly between experiments. In all cases, at least one packet was dropped each time a rate change occurred. Sometimes many more packets were dropped over the course of the experiment. The number of dropped packets seemed to correspond, at least in part, to the combination of wireless cards used. For example, Figures 1 and 2 both show the results of experiments where different laptops were used with the exact same combination of wireless cards. For both experiments, the same Linksys card transmitted to the D-Link card. Throughout the experiments, a significant number of packets were dropped as indicated by the black dots. Figures 3 and 4, on the other hand, show the results of experiments where the same laptops were used and few packets were dropped. In fact, so few packets were dropped, they are not visible due to the density of the data points in Figures 3 and 4. In this case, the D-Link card was the sender and the Linksys card was the receiver. These results indicate that simply swapping the wireless cards determined whether few or many packets were dropped.

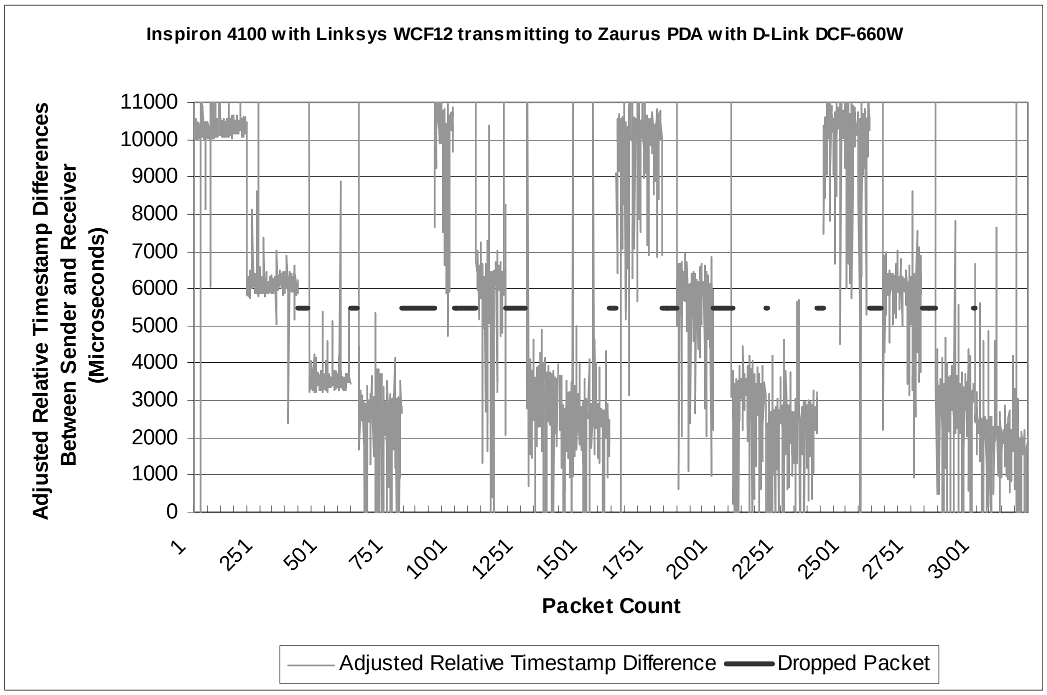

Figure 1: Adjusted relative timestamp differences between sender and receiver. Due to asynchronous and skewed clocks, the plotted data points were adjusted. They are only accurate relative to one another, not their absolute values.

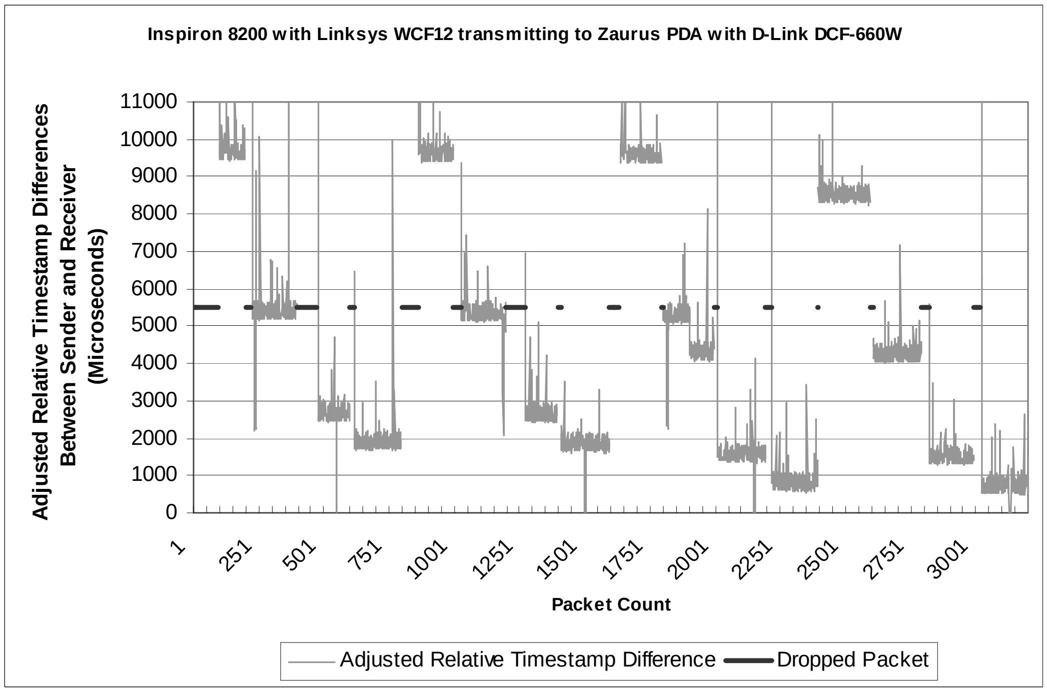

Figure 2: Adjusted relative timestamp differences between sender and receiver. Due to asynchronous and skewed clocks, the plotted data points were adjusted. They are only accurate relative to one another, not their absolute values.

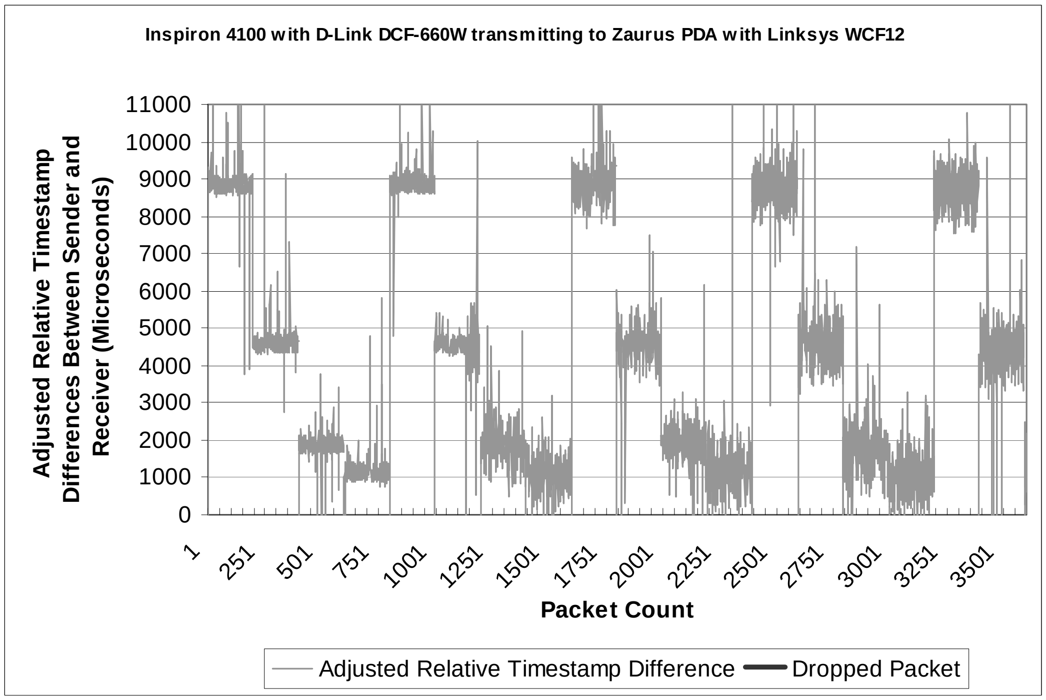

Figure 3: Adjusted relative timestamp differences between sender and receiver. Due to asynchronous and skewed clocks, the plotted data points were adjusted. They are only accurate relative to one another, not their absolute values.

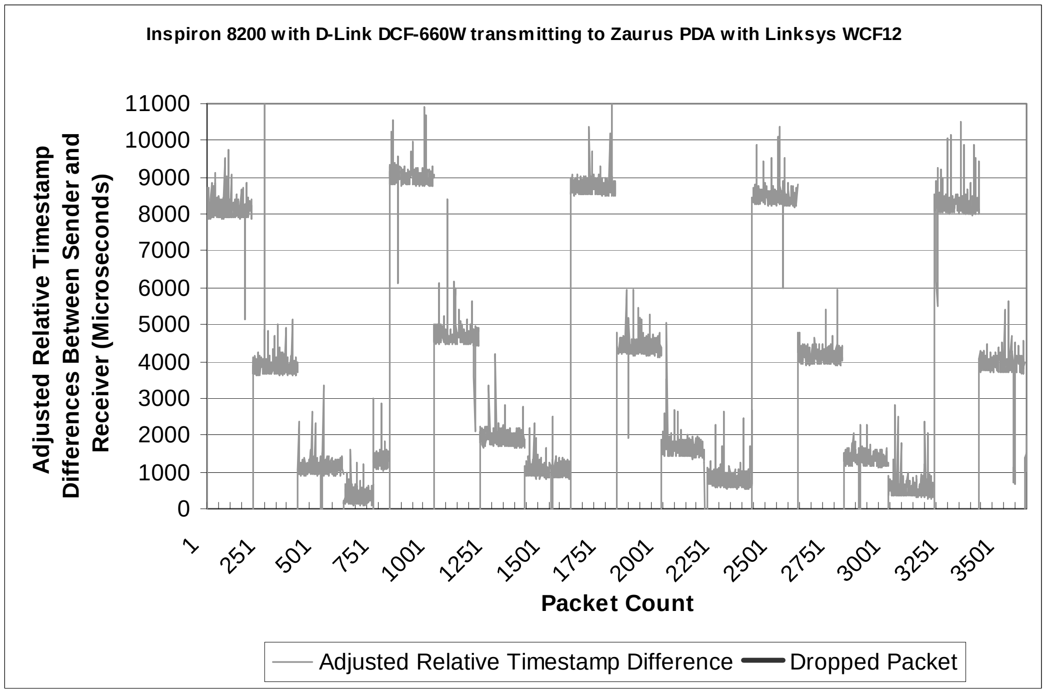

Figure 4: Adjusted relative timestamp differences between sender and receiver. Due to asynchronous and skewed clocks, the plotted data points were adjusted. They are only accurate relative to one another, not their absolute values.

A final interesting trend observable from Figures 1 through 4 is that, when the Inspiron 4100 laptop was used, there was typically much more variation between the adjusted relative timestamp differences of received packets at any given rate than when the Inspiron 8200 was used. To investigate this trend further, an experiment was performed using the Inspiron 4100 running the same SLAX LiveCD that the Inspiron 8200 ran. The purpose of this experiment was to determine whether hardware or software was the culprit of the observed variation. In all previous experiments, the Inspiron 4100 ran its installed Red Hat Enterprise Linux 4.0 operating system. Figure 5 shows the result of this experiment. What is interesting to note is that the only difference between the experiment shown in Figure 4 and the experiment shown in Figure 5 is the internal hardware of the laptops. All wireless cards and software were identical in those experiments.

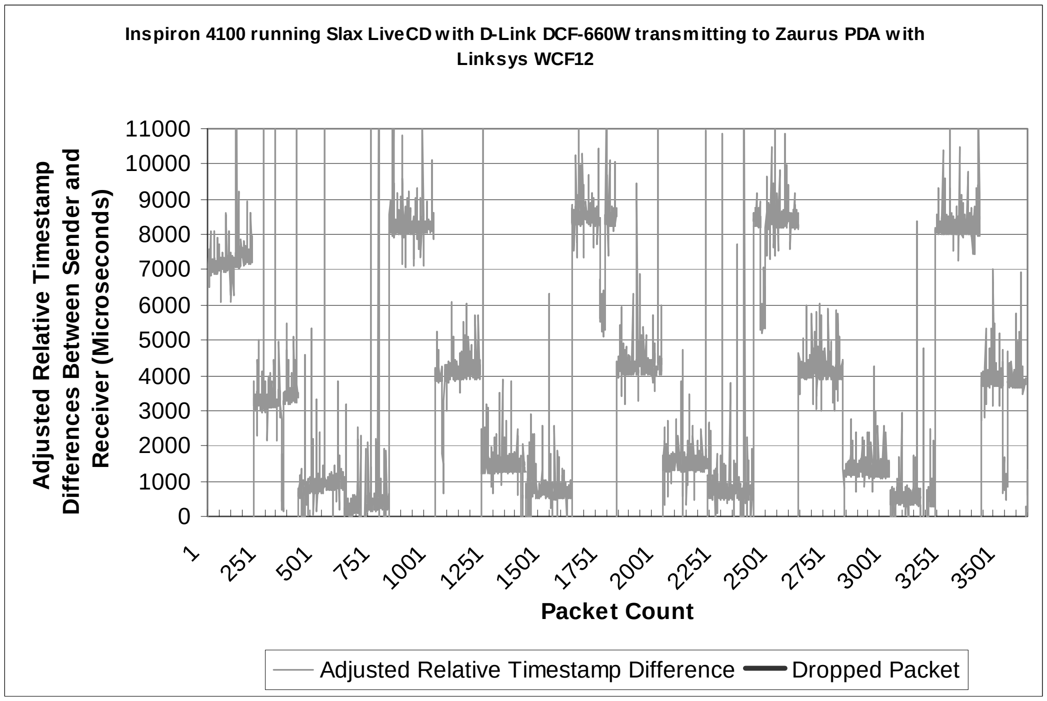

Figure 5: Adjusted relative timestamp differences between sender and receiver. Due to asynchronous and skewed clocks, the plotted data points were adjusted. They are only accurate relative to one another, not their absolute values.

The results of the UDP experiment showed that some hardware configurations performed better than others did. After comparing all the results, as mentioned earlier, it was determined that using the Inspiron 8200 with the D-Link DCF-660W wireless card transmitting to the PDA with a Linksys WCF12 was the hardware configuration that yielded the best results. Thus, this hardware configuration was used for all remaining experiments described here.

3.5 Rate Change Frequency

In all runs of the UDP experiment, at least one packet was dropped every time a rate change was made. This indicated that it consistently took a significant amount of time to change the transmission rate of the wireless cards. Thus, the next set of experiments was designed to determine the typical amount of time required to transition to a new transmission rate. All experiments up to this point had paused one second between each packet sent. Determining the amount of time required to transition to a new transmission rate called for the ability to pause for varying amounts of time between sending packets and changing rates. This experiment was designed to attempt to change the transmission rate after each packet was transmitted. Rate change requests were issued after every packet sent to mimic the worst-case behavior. In the worst case, the dynamic rate selection algorithm would choose to change the transmission rate after sending only a single packet at a given rate. The time waited between sending two consecutive packets is depicted as 2*t* in Figure 6.

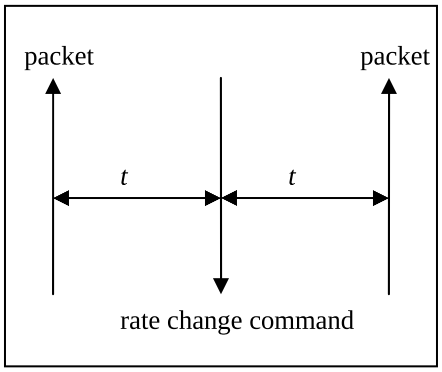

Figure 6

A series of fifteen experiments were performed. The first five consisted of sending 500 packets. In each case, *t* was initially set to 7 seconds and decreased by 50 milliseconds every five seconds. As *t* became too small for the rate changing to handle, more packets were dropped. For each experiment, the value of *t* was found for the point after which all packets were dropped.

Next, a series of ten experiments were performed where the value of *t* remained constant for 500 packets. The purpose of these experiments was to determine the percentage of packets lost for different values of *t*. Although the sender may be able to transmit some packets for a given value of *t*, performance is still unacceptable if a high percentage of the packets are dropped.

Due to the nature of this experiment, the UDP program and corresponding Perl script used in the previous experiments were significantly modified. The functionality of the Perl script was incorporated into the UDP program so that more control could be enforced over when rate change commands were issued as well as how often they were issued. The receiver still performed the same computation of the time it took each packet to reach its destination by subtracting each packet's timestamp from the time that each packet arrived at the receiver. Also like the previous experiments, the receiver kept track of packet sequence numbers in order to detect dropped packets.

3.6 Rate Change Frequency Results

The results of these experiments were a clear indication of why a dynamic rate selection implementation on the given hardware would be impractical. As already described, the initial five tests were designed to decrease the value of *t* and determine the *t* values after which all packets were dropped. These values ranged from 2.25 seconds to 2.4 seconds. The results of these experiments indicated that 2.4 seconds was, perhaps, the minimum reasonable amount of time to wait between changing rates and sending packets. The following ten tests were designed to determine the number of packets dropped for different values of *t* sustained over longer intervals. The results of these tests are summarized in Figure 7. Waiting 2.4 seconds did not eliminate packet loss, but did result in significantly better performance than waiting less than 2.4 seconds. Waiting 2.3 seconds resulted in nearly 12% packet loss, whereas waiting 2.4 seconds or more resulted in, at worst, 3% packet loss. No pattern was seen in when, during the experiments, packets were lost. Because this waiting occurred after transmitting each packet and after each change of rate (see Figure 6), the total amount of time waited between transmitting each packet for a *t* value of 2.4 seconds was 2*t*, or 4.8 seconds.

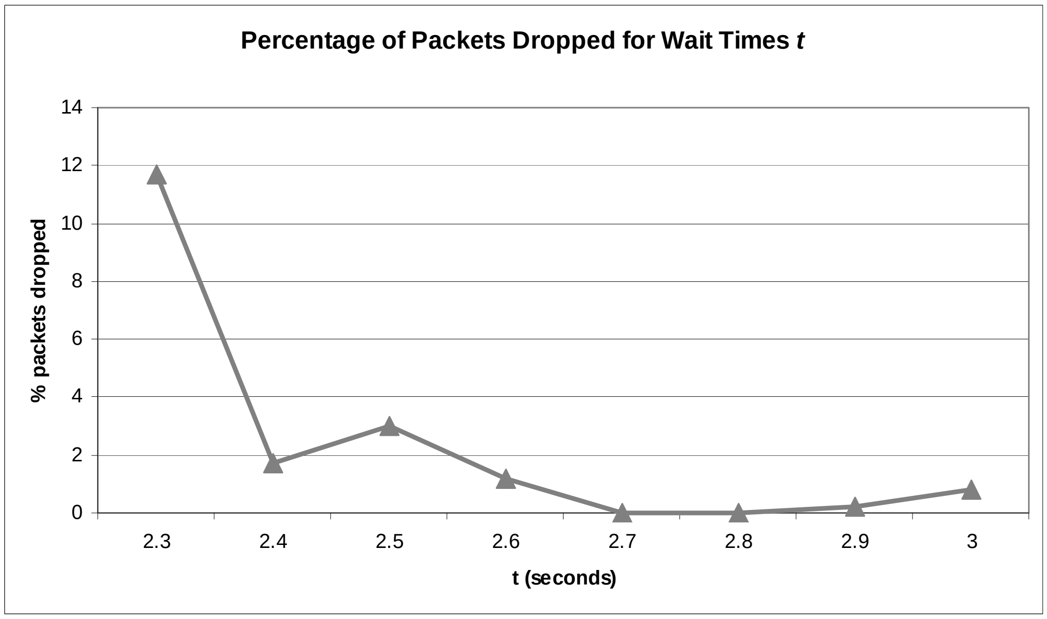

Figure 7

3.7 Power Consumption

The results of the ping and UDP experiments also raised the question of exactly how much power can be saved by using dynamic rate selection. If the amount of power consumed when transmitting at 1Mbps is significantly less than the power consumed when transmitting at 11Mbps, perhaps the inconsistencies and poor performance observed during the UDP and ping experiments is tolerable. Thus, a set of power consumption experiments were designed to measure the amount of power consumed during packet transmission and rate changing. Because the Power Angel device is designed to measure power at a wall outlet, power measurements were taken for the entire sender device. In other words, the design of the measurement device forced measurement of the power consumed by the entire sender laptop rather than only the power consumed by the wireless card.

A series of three experiments were performed. The first experiment measured the power consumed by the laptop in an idle state. This means that, for the duration of the experiment, zero packets were transmitted. The second experiment measured the power consumed by the laptop while constantly transmitting packets at each of the transmission rates. Separate measurements were taken for each transmission rate. The final experiment measured the power consumed by the laptop while constantly issuing rate change commands.

In all of the power consumption experiments, the sender laptop was configured identically. The machine was booted with only a flash drive and mouse attached. The wireless card was then configured, and the appropriate program was run to perform the desired task to be measured. At that point, the Power Angel was plugged in, and the laptop's battery was removed. In all cases, the receiving PDA's wireless card was positioned roughly 1.5 feet from the laptop's wireless card.

3.8 Power Consumption Results

Action	Watts	Power/Rate
Idle	27.056	-
Transmitting at 1Mbps	30.822	30.822
Transmitting at 2Mpbs	32.014	16.007
Transmitting at 5.5Mbps	33.058	6.011
Transmitting at 11 Mbps	33.040	3.004
Changing Transmission Rate	41.026	-

Table 1

Table 1 summarizes the results of the average power consumption for the experiments. These results show that changing the transmission rate consumes the most power. Transmitting packets at 1Mbps consumes the least amount of power aside from the idle state. One noticeable issue with this data is that the power consumed by transmitting at 5.5Mbps was reported as more than the power consumed by transmitting at 11Mbps. The reason for this discrepancy in the data is likely due to a combination of the inability of the device used to measure power precisely enough and the conditions under which power was measured. The most precise measurement possible with the Power Angel is 10Wh. Thus, there is potential for the measurements to be affected by factors other than the transmission of packets. Perhaps the laptop's fan ran more during the experiment measuring the 5.5Mbps transmission rate. Although not perfectly accurate, the power measurements do show potential for the power that can be saved by transmitting at a lower rate. According to the above results, approximately 6.7% $\left((33.040 - 30.822)/33.040\right)$ less total power was required for the laptop to transmit at 1Mbps than at 11Mbps. Considering only the power consumed by the wireless card, it took 37.1% $((33.040 - 30.822)/(33.040 - 27.056))$ less power to transmit at 1Mbps than at 11Mbps.

4. Conclusions

Several conclusions can be drawn from the results of this work. First, the poor performance and inconsistencies observed in the various experiment results are, at least in part, hardware dependent. The results of several UDP experiments using two nearly identical hardware configurations showed differences in the variation between the adjusted timestamp differences of received packets at any given rate. The number of dropped packets during the UDP experiments seemed to depend on the sender-receiver combination of the wireless cards. In addition, for all long versions of the UDP experiment, the final 2000+ packets were always dropped, indicating that prolonged communication between those specific devices in that manner was not reliable. All of these observations support the claim that the performance of dynamic rate selection depends on the hardware used. Perhaps there are hardware combinations that would allow dynamic rate selection to work perfectly, but none of the configurations tried in this work showed such potential.

Another conclusion is that it hardly seems worthwhile to do dynamic rate selection based on the results of the rate change frequency experiments. The rate change frequency experiments' results showed that when a rate change is performed between transmitting consecutive packets, at least 4.8 seconds should be waited between sending the two packets. In almost all applications where the dynamic rate selection algorithm would be used, 4.8 seconds is an unacceptably long time to wait.

Based on the long required wait times, the amount of power saved by transmitting at a lower rate also does not fully justify implementing dynamic rate selection. The power consumption experiments did show, however, that power was saved by transmitting at a lower rate. Although it may seem that there would be few instances where it would be worthwhile to delay data transmission by 4.8 seconds to save power, such scenarios certainly exist. For example, if the ad-hoc mobile network is relatively stationary or if the nodes in the network transmit infrequent and/or small amounts of data, rate changes will not frequently be necessary. Perhaps the most promising aspect of the results to consider is that dynamic rate selection was developed with small portable devices in mind. Only 6.7% less total power was consumed by the laptop when transmitting at a lower rate. If the device doing the transmitting consumed much less power overall, the total savings would be much greater. The fact that the wireless card consumed 37.1% less power when transmitting at 1Mbps than at 11Mbps shows just how much potential for power savings exists.

Therefore, the overall conclusion that was drawn from this work is that implementing dynamic rate selection on a per packet basis with the hardware used is not feasible. Any performance that could be gained by dynamic rate selection is surely nullified by the poor performance and inconsistencies seen in the hardware. The amount of power that can be saved by dynamic rate selection, however, is promising. Furthermore, even given the poor performance and inconsistencies, dynamic rate selection may be worthwhile for some special scenarios. Perhaps future hardware will make the implementation of dynamic rate selection more worthwhile for a wider range of applications.

References

[1] N. AbouGhazaleh, P. Lanigan, S. Gobriel, D. Mossé, R. Melhem. Dynamic Rate-Selection for Extending the Lifetime of Energy-Constrained Networks. *In Workshop on Energy Efficient Wireless Communication Networks, in conjunction with IPCCC 2004.*

[2] D. B. Johnson, D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.*

[3] D. B. Johnson, D. A. Maltz, Josh Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5, pages 139-172, Addison-Wesley, 2001.*

[4] J. Tourrilhes. Wireless LAN resources for Linux. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/.

[5] OpenZaurus. http://openzaurus.linuxtogo.org/wordpress/.

[6] Sharp Zaurus Developer Site. http://www.ossh.com/zaurus/mirrors/docs.zaurus.com/linux_compiler_setup_howto.shtml.

[7] J. John Selbie. A Crash Course in UNIX TCP/IP Socket Programming. http://www.fortunecity.com/skyscraper/arpanet/6/cc.htm.