Evaluation of the Minstrel Rate Adaptation Algorithm in IEEE 802.11g WLANs

Dong Xia[1,2] Jonathan Hart[1] Qiang Fu[1]

[1]School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600
Wellington 6140, New Zealand

[2]Ministry of Social Development, PO Box 1556
Wellington 6140, New Zealand

Email: {dong.xia, jonathan.hart, qiang.fu}@ecs.vuw.ac.nz

***Abstract*—Rate adaptation varies the transmission rate of a wireless sender to match the wireless channel conditions, in order to achieve the best possible performance. It is a key component of IEEE 802.11 wireless networks. *Minstrel* is a popular rate adaptation algorithm due to its efficiency and availability in commonly used wireless drivers. However, despite its popularity, little work has been done on evaluating the performance of *Minstrel* or comparing it to the performance of fixed rates. In this paper, we conduct an experimental study that compares the performance of *Minstrel* against fixed rates in an IEEE 802.11g testbed. The experiment results show that whilst *Minstrel* performs reasonably well in static wireless channel conditions, in some cases the algorithm has difficulty selecting the optimal data rate in the presence of dynamic channel conditions. In addition, *Minstrel* performs well when the channel condition improves from bad quality to good quality. However, *Minstrel* has trouble selecting the optimal rate when the channel condition deteriorates from good quality to bad quality.**

I. INTRODUCTION

The IEEE 802.11 standards specify multiple data rates that can be used by a wireless sender, depending on the modulation and coding schemes used at the PHY layer. High data rates transmit data faster than low data rates, however high data rates are more susceptible to bit errors. This means more packets are lost on low quality wireless channels with high bit error rates (BERs). Low data rates take longer to transmit packets over the link, however they are more resistant to bit errors and are more likely to be successfully transmitted over a poor quality wireless link.

Wireless channels are extremely variable and can be affected by a number of different factors, such as interference from other wireless devices, multi-path fading and signal attenuation. As such, one of the key components of an 802.11 system is the rate adaptation mechanism, which adapts the data rate used by a wireless sender to the wireless channel conditions. Rate adaptation is optimization problem: if we use a rate that is too high, many of the packets will be dropped due to bit errors, however if we use a rate that is too low, the wireless channel is not fully utilized.

Rate adaptation is crucial to wireless network performance, and therefore much research has been conducted on how to design effective rate adaptation algorithms. *Minstrel* [1] is a popular, recently-developed rate adaption algorithm, although there is currently very little literature studying its performance.

Therefore, in this paper we present an experimental study which analyzes the performance of *Minstrel* by comparing it against fixed rates in a real-world IEEE 802.11g wireless testbed. Fixed rates provide a performance benchmark which a good rate adaptation algorithm should be at least able to meet, if not exceed. We therefore consider an evaluation against fixed rates to be an important tool in the analysis of rate adaptation algorithm behavior.

To fully understand the performance of *Minstrel* we conducted experiments of three different scenarios: static channel transmission, dynamic channels with fast variation, and dynamic channels with slow variation. This work builds on our previous study which evaluates *Minstrel* [2]. In this paper we consider different scenarios to provide a more detailed evaluation. We have also further analyzed our results in order to provide deeper insights into *Minstrel* and rate adaptation algorithm design. Our results show that whilst *Minstrel* performs well in static wireless channel conditions, the algorithm has difficulty selecting the optimal data rates in more dynamic channel conditions. Furthermore, *Minstrel* performs well compared to fixed rates when the channel conditions improve from a poor quality starting point, however it has trouble selecting optimal rates when the channel conditions deteriorate from a high quality starting point.

The rest of the paper is organized as follows. We discuss related work in Section II. Section III describes the *Minstrel* rate adaptation algorithm, and Section IV outlines our experiment methodology. Section V presents and discusses our results, and finally Section VI concludes the paper.

II. RELATED WORK

The topic of rate adaptation has been heavily researched. Previous research generally either targets the design of new rate adaptation algorithms [3]-[10], or compares the performance of different rate adaptation algorithms [11]-[13]. The *SampleRate* algorithm [14] was previously considered one of the best publicly available algorithms, and is often used for comparison in these studies. *Minstrel* works similarly to *SampleRate*, although it differs in the statistic used to measure the success of each rate. *SampleRate* uses the expected transmission time, whereas *Minstrel* uses the expected throughput.

Ancillotti et al. evaluated three different rate adaptation algorithms in wireless mesh networks [15]. This evaluation includes *SampleRate*, and there is some comparison between rate adaptation algorithms and fixed rates. However, the focus of the paper is on the correlation between the number of senders and the rate adaptation algorithms. The evaluation did however show in a particular scenario that the performance of some of the fixed rates was better than the evaluated rate adaptation algorithms.

Wong et al. proposed an algorithm called Robust Rate Adaptation Algorithm (RRAA) [16], which was evaluated against *SampleRate* and earlier algorithms such as Auto Rate Fallback (ARF) and Adaptive Auto Rate Fallback (AARF). Acharya et al. proposed Wireless cOngestion Optimized Fallback (WOOF) [17], a rate adaptation algorithm that attempts to identify congestion related packet losses. WOOF was evaluated against other solutions including *SampleRate.* Much of this work was conducted before *Minstrel* was developed, so the authors were unable to compare with *Minstrel*. As such, performance evaluations of *Minstrel* are scarce.

The most closely related works are the evaluations done by Yin et al. [12][13]. These are the only previous evaluations of *Minstrel* to our knowledge. In one paper the four rate adaptation algorithms found in the *MadWiFi* driver (namely ONOE, AMRR, *SampleRate* and *Minstrel*) are evaluated against one another [12]. The second paper compares the algorithms found in the *mac80211* Linux driver framework [13], namely *Minstrel* and PID. These evaluations use a wired testbed with co-axial cables to emulate wireless channels. This is a clever solution because the algorithms are evaluated in an environment that is free of interference, which means the experiments are able to produce repeatable results. These evaluations show that *Minstrel* performs far better than the other algorithms. Therefore we chose not to include other schemes such as *SampleRate* in our evaluation. Moreover, a wired testbed cannot fully reproduce the complex channel environments found in real-world 802.11 deployments, therefore we feel that real-world experimental studies are also important for fully understanding rate adaptation behavior.

Whilst comparisons among different rate adaptation algorithms are certainly very useful, most of these studies do not directly compare against fixed rates. We feel that comparing rate adaptation against fixed data rates is an important step towards the design of effective rate adaptation algorithms.

III. THE MINSTREL ALGORITHM

There are three parts to the *Minstrel* algorithms: the retry chain mechanism, the rate decision process and the statistic calculations.

A. Retry Chain

The *Minstrel* rate adaptation algorithm uses a mechanism called a *multi-rate retry chain,* which enables it react to short-term variations in channel quality. The retry chain consists of four rate-count pairs, named $r0/c0$, $r1/c1$, $r2/c2$, and $r3/c3$. A packet is first transmitted at rate $r0$ for $c0$ attempts. If these attempts are not successful, *Minstrel* transmits the frame at rate $r1$ for $c1$ attempts. The process continues until either the packet is successfully transmitted or ultimately discarded after ($c0 + c1 + c2 + c3$) unsuccessful transmission attempts. The following section discusses how *Minstrel* chooses the *r*-values in the retry chain.

B. Rate Selection

There are two categories of transmission: normal transmission occurs 90% of the time and sampling transmission occurs for the remaining 10% of packets. Table I gives a summary of the rate selection decisions.

1) Normal Transmission: During normal transmission the *r*-values in the retry chain are chosen as follows: *r0* is set to the rate that achieves the highest expected throughput, *r1* is the rate with the second highest expected throughput, *r2* is the rate with the highest probability of success, and finally *r3* is set to the lowest available data rate.

2) Sampling Transmission: Minstrel relies on having accurate statistics about the success rate of transmissions at each data rate. Of course, it has to attempt to send packets at each data rate in order to have statistics on them. 10% of the data frames are sent as sampling transmissions, where a random rate not currently in the retry chain is chosen to sample. The the *r*-values are chosen as follows: *r0* is set to whichever is higher out of the sample rate or the rate with the highest expected throughput, and *r1* is set to whichever is lower. *r2* and *r3* remain the rate with the highest probability of success and the lowest available rate respectively.

C. Statistics Calculation

The final piece of the puzzle is how *Minstrel* calculates the probability of success and expected throughput for each data rate. *Minstrel* maintains the probability of successful transmission at each data rate as an Exponentially Weighted Moving Average (EWMA). This probability is based on the historical success rate of packet transmissions at each data rate. This probability is used to estimate the throughput of each rate and the retry chain is re-evaluated based on this estimate every 100ms. In each 100ms sampling window, the success rate, R_S, is calculated for each data rate based on the

TABLE I. MINSTREL RETRY CHAIN [1]

Rate	Sampling Transmission		Normal Transmission
	Random < Best	***Random > Best***	
r0	Best rate[1]	Random rate	Best rate
r1	Random rate	Best rate	Second best rate
r2	Best probability[2]	Best probability	Best probability
r3	Base rate[3]	Base rate	Base rate

[1] The data rate that gives the highest throughput
[2] The data rate that has the highest rate of successful transmissions
[3] The lowest available data rate

historical observation of packet successes and failures as in (1), where N_S is the number of packets transmitted successfully at the data rate and N_T is the total number of packets attempted at the data rate.

$$R_S = N_S / N_T \tag{1}$$

$$P(t + 1) = R_S \times (1 - \alpha) + P(t) \times \alpha \tag{2}$$

The R_S value is then used to alter the measured value for the probability of success for each data rate using the EWMA expression (2). The EWMA parameter α is used to determine how much weight is given to the R_S value from the new sampling period. The default value of α is 0.25 which means the new probability of success is comprised of 75% of the new sample and 25% of the previous probability of success. Finally *Minstrel* calculates an expected throughput for each data rate, T, as in (3).

$$T = P_{success} \times (B / t) \tag{3}$$

The expected throughput T is the number of bytes B transferred in time t (i.e. the raw throughput of the rate) scaled by the probability of success $P_{success}$ which is maintained by the EWMA expression (2). This results in an expected throughput for each rate which is based on the station's previous observations of the proportion of packets that have been successfully transmitted at the data rate.

IV. Experiment Methodology

A. Testbed Platform

Figure 1 shows the network topology of the testbed. A wireless Access Point (AP) is connected via Ethernet to a sender PC. We use *iperf* to send UDP traffic from the sender PC to a laptop receiver, which is wirelessly associated with the AP. The AP and laptop use wireless cards containing the Atheros AR5414 chipset. The AP is based on the MikroTik Routerboard R52 platform and runs the Linux-based OpenWrt backfire 10.03 [18] operating system with a 2.6.32 kernel. The server PC and client laptop run Ubuntu 10.04 with a 2.6.23 kernel. The AP used 802.11g mode.

This real-world evaluation was conducted in an office building on a university campus. We are aware of other wireless network operating in the area. While interference from these networks can affect our experiment system, it also reflects the complex nature of a real-world wireless environment and the fact that wireless networks often do operate in the vicinity of one another. It is important to know how wireless systems behave in the presence of interference,

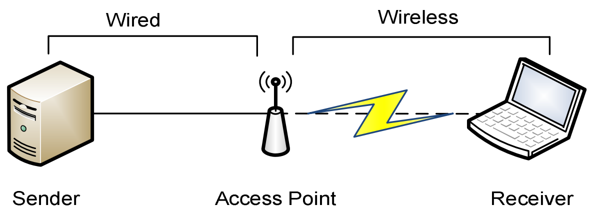

Figure 1. Testbed topology

rather than only unrealistic isolated environments. In light of this, each experiment is carried out 20 times in order to maintain a high level of statistical significance. We ran each experiment in two different locations to ensure our results weren't caused by abnormalities of a particular location. We observed similar results at either location.

B. Experiment Design

The experiments focus on three different scenarios: static channel transmission, fast variations in channel quality and gradual changes in channel quality. We use the transmission power of the AP to control the wireless channel conditions. When the transmission power is high, the wireless channel conditions are good. Then as we decrease the transmission power of the AP, the wireless channel conditions deteriorate, the channel becomes more susceptible to interference, and this emulates a lossy or poor quality wireless channel.

The goal of our evaluation is to compare *Minstrel* to fixed rates to analyze the performance of the algorithm. By fixed rates, we mean that we disable rate adaptation and manually set the wireless cards to use a specific rate. This gives us a baseline indication of the performance capabilities of the wireless channel in each situation. We conduct each experiment with each of the 802.11g fixed rates, and also with *Minstrel*.

1) Static Channel: In this scenario, the transmission power is fixed for the duration of experiments. This means the quality of the wireless channel is relatively stable during each experiment. We perform different experiments with different transmission powers ranging from 17 to 1 dBm. This experiment provides basic performance comparison between *Minstrel* and fixed rates.

2) Rapid Channel Variation: In this scenario we create large variations in the wireless channel quality, i.e. the channel will jump from good quality to poor quality quickly, and vice versa. This could happen, for example, in a situation when some external interference briefly causes a drop in channel quality. This is implemented by changing the AP's transmission power from 17 dBm to 1 dBm and back in a periodic fashion. The transmission power is held at 17 dBm for 2s before jumping to 1 dBm. We run different experiments in which the duration that the transmission power stays at 1 dBm before jumping back up is varied from 2s to 10s.

3) Gradual Channel Variation: Sometimes the channel quality will change gradually, for example, if a wireless client moves spatially with respect to the AP. In this scenario, we gradually change the wireless channel condition by increasing or decreasing the transmission power during the experiment. We consider both low to high and high to low changes. We run experiments with different *update periods*, i.e. the time value inbetween consecutive power level updates. We vary this period from 2 to 5 seconds. At the beginning of the experiment the transmission power is either set to 17 dBm or 1 dBm, then during the experiment the transmission power is increased or decreased by 2 dBm increments at each update period until it reaches 1 dBm or 17 dBm.

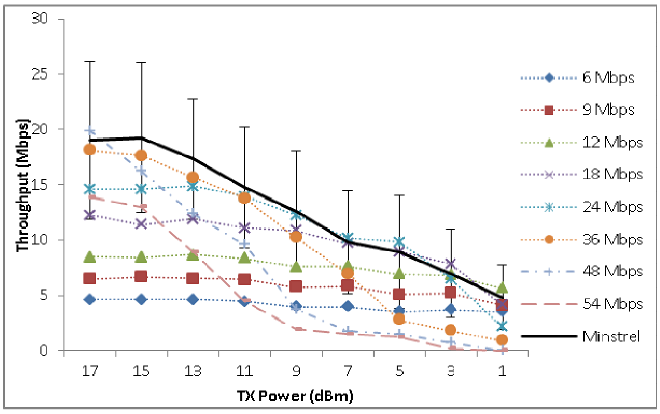

Figure 2. Throughput with a static channel

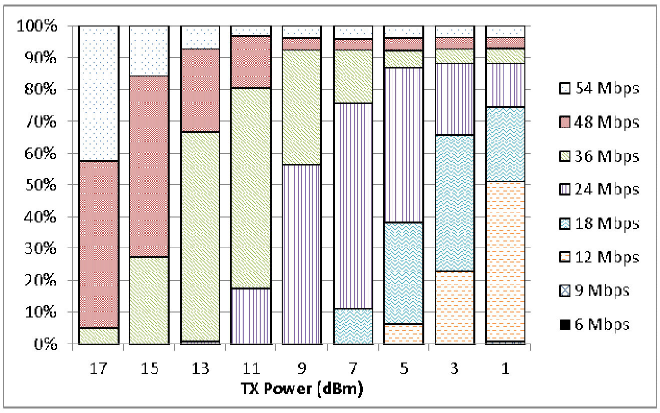

Figure 3. *Minstrel* rate selection breakdown in a static channel

V. RESULTS

A. *Static Channel*

Figure 2 shows the throughput of *Minstrel* and each of the fixed rates. The transmission power is static during each experiment, and we conduct experiments at a number of different power levels. The throughput of *Minstrel* decreases as the transmission power decreases. In addition, *Minstrel* achieves comparable performance to the best-performing fixed rate at all power levels. This experiment also shows that reducing the transmission power has a significant performance impact on the higher data rates, because more frames are lost due to the lower quality wireless channel.

Figure 3 shows a percentage breakdown of the data rates attempted by *Minstrel* in the static channel experiment at each different power level. When the transmission power is set to 17 dBm (the highest quality channel), *Minstrel* mostly attempts to use 54 Mbps and 48 Mbps. However, at the lower transmission power levels the channel drops more packets, and *Minstrel* attempts the lower data rates more often. When the transmission power is set to 1 dBm, more than 70% of attempts are made at the 12 Mbps and 18 Mbps rates. By the nature of *Minstrel* sampling algorithm, if the randomly selected rate is higher than the current best rate, the random rate is used first, otherwise the random rate is used second.

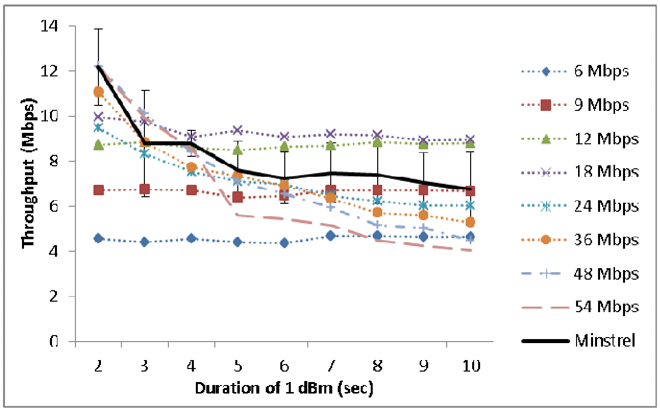

Figure 4. Throughput with rapid channel variation

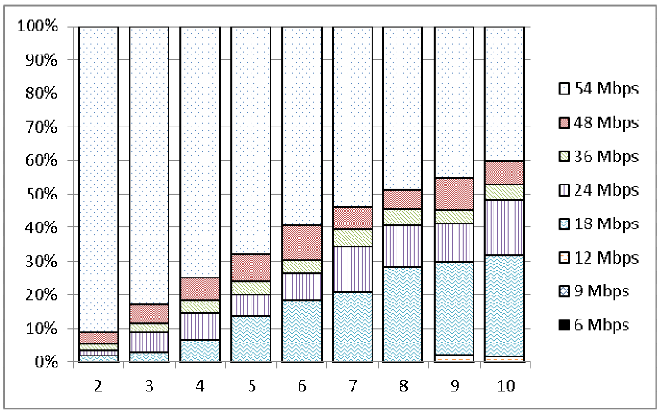

Figure 5. *Minstrel* rate selection breakdown with rapid channel variation

Higher data rates always have a higher chance of being sampled. As Figure 3 shows, even when the transmission power is low, a significant proportion of packets that are still attempted at high data rates. This means that *Minstrel* has better statistics on the high data rates because it uses them more often than the low data rates. However, *Minstrel's* tendency to sample and select higher data rates doesn't incur a significant performance cost in stable channel conditions.

B. *Rapid Channel Variation*

In this section, we study the performance of *Minstrel* in dynamic channel conditions by periodically switching between high and low transmission power levels.

Figure 4 shows the performance of fixed rates and *Minstrel* in relation to the duration of the low transmission power. As the duration of the low quality channel increases, the performance of *Minstrel* and the high fixed rates (54, 48, and 36 Mbps) suffers significantly. This poor performance is expected for the high rates, as they drop more packets due to bit errors in a low quality wireless channel. The performance of *Minstrel* and high rates stabilizes after the low power duration increases past 5 seconds. In contrast, the performance of the low rates is relatively consistent even as the low power duration increases. We expect that sudden changes would reduce *Minstrel's* performance suffer, however it is surprising to see that *Minstrel* is outperformed in most cases by the 12 Mbps and 18 Mbps fixed rates, which unable to adapt at all.

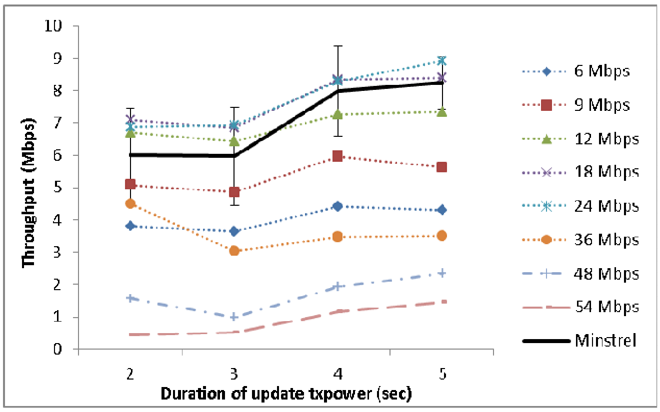

Figure 6. Throughput with gradually deteriorating channel conditions

The rate breakdown statistics for *Minstrel* are shown in Figure 5. In Figure 4, we have observed that the best throughput is achieved by the 54 Mbps and 48 Mbps rates when the duration of low power is low (2-3 seconds). When the duration is greater than 4 seconds, the 12 Mbps and 18 Mbps rates have the best throughput. In Figure 5, it can be seen that *Minstrel* most frequently selects 54 Mbps over the other rates for transmission. This makes *Minstrel* comparable to the best rates (54/48 Mbps) in terms of throughput when the low power duration is short. As the duration increases, *Minstrel* uses the 54 Mbps rate less frequently and the 18 Mbps rate more frequently. However *Minstrel* does not use the 18 Mbps and 12 Mbps rates enough, which leads to the poor performance we observed. The rightmost bar in Figure 5 shows the case when the cycle consists of high power for 2 seconds and low power for 10 seconds, in other words a 5:1 ratio of low power duration to high power duration. However the figure shows that *Minstrel* attempts low and high rates almost equally, when it clearly should use low rates a much greater percentage of the time. This experiment indicates that *Minstrel's* tendency to use higher rates becomes costly in highly dynamic channel conditions.

C. Gradual Channel Variation

In this experiment, we further analyze the behavior of *Minstrel* in dynamic wireless environments. However in this scenario the channel conditions gradually change from bad to good or vice versa so that *Minstrel* has time to adapt. The transmission power increases or decreases gradually, and the time period between power level updates is varied from 2 seconds to 5 seconds.

Figure 6 shows the throughput of *Minstrel* and fixed rates as the transmission power decreases from 17 dBm to 1 dBm, with different update periods. The 18 Mbps and 24 Mbps fixed rates outperform *Minstrel*, although the performance difference between these two fixed rates and *Minstrel* becomes less significant as the duration of update period increases. When the update period is short the channel is relatively dynamic, but as the update period lengthens the channel becomes relatively static. *Minstrel* updates its retry chain every 100ms, which means it updates 20 times in a 2 second period and 50 times in a 5 second period. *Minstrel* is able to use the optimal rate a greater proportion of the time when the update period is longer.

From looking at the rate distribution (not shown), we can see that *Minstrel* attempts to use low data rates more often as the transmission power decreases. However, fixed rates still outperform *Minstrel*, suggesting that *Minstrel* makes incorrect rate selection as the channel quality deteriorates.

Table II shows the rate most frequently used by *Minstrel* (outside the brackets), and the fixed rate that achieves the highest ideal throughput (inside the brackets) for each transmission power update period at different power levels. The ideal throughput, T_I, is calculated as in (4) as the data rate multiplied by the ratio of successful attempts (N_S) to total attempts (N_T).

$$T_I = \text{Rate} \times (N_S / N_T) \qquad (4)$$

Minstrel should strive to use the rate with the highest throughput most often, however during these experiments it often fails to do so, as indicated by the bold entries in Table II(a). *Minstrel* often makes incorrect rate choices when the power level is low and when the transmission power update period is short. Again, *Minstrel* tends to select a rate higher

TABLE II. RATE MOST FREQUENTY ATTEMPTED BY MINSTREL (OUTSIDE BRACKETS) VS. RATE WITH HIGHEST IDEAL THROUGHPUT (INSIDE BRACKETS)

TX Power (dBm)	2s update period	3s update period	4s update period	5s update period
17	36(36)	36(36)	36(36)	36(36)
15	36(36)	**36(24)**	36(36)	36(36)
13	**36(24)**	24(24)	36(36)	36(36)
11	24(24)	24(24)	24(24)	24(24)
9	**24(12)**	**24(12)**	24(24)	24(24)
7	**24(12)**	**24(12)**	**24(12)**	24(24)
5	**24(12)**	**24(12)**	**24(12)**	**24(12)**
3	**24(12)**	**24(12)**	**24(12)**	12(12)
1	**24(12)**	**24(12)**	**18(12)**	12(12)

(a) Deteriorating channel conditions

TX Power (dBm)	2s update period	3s update period	4s update period	5s update period
17	**24(18)**	**24(18)**	**24(18)**	12(12)
15	**24(18)**	18(18)	18(18)	18(18)
13	24(24)	18(18)	18(18)	24(24)
11	24(24)	**18(24)**	24(24)	24(24)
9	24(24)	24(24)	36(36)	36(36)
7	36(36)	36(36)	36(36)	36(36)
5	36(36)	36(36)	36(36)	36(36)
3	36(36)	36(36)	48(48)	48(48)
1	48(48)	48(48)	48(48)	48(48)

(b) Improving channel conditions

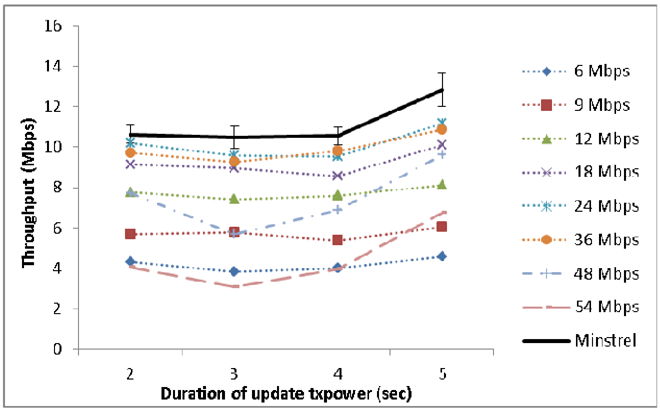

Figure 7. Throughput with gradually improving channel conditions

than the optimal rate. However, as the update period increases, *Minstrel*'s selection of rates becomes more accurate. This explains why *Minstrel*'s performance improves with the longer update period.

Figure 7 shows the throughput of fixed rates and *Minstrel* when the transmission power increases from 1 dBm to 17 dBm. *Minstrel* has superior performance over the fixed rates in all cases. This is the positive outcome of the algorithm which aggressively samples higher rates over lower rates. This means that as soon as the channel conditions improve, *Minstrel* is quickly able to take advantage and use higher rates. Table II(b) again shows the rate most often used by *Minstrel* and the rate with the highest ideal throughput in each situation. We can see that in this scenario when the channel conditions are improving, *Minstrel* becomes more accurate in choosing the rate with the highest ideal throughput. Moreover, as the update period increases, *Minstrel* has more time to adapt and is able to choose the correct rate more accurately.

VI. CONCLUSION

In this paper, we evaluate the performance of *Minstrel* against fixed rates in a real-world IEEE 802.11g wireless testbed. Our results show that *Minstrel* can achieve a performance comparable to the best performing fixed rates when the wireless channel conditions are relatively static. With static channels, *Minstrel* can usually accurately identify the best rates to use. However, in the presence of dynamic channel conditions, *Minstrel* is significantly outperformed by certain fixed rates. This is due to *Minstrel's* tendency to select rates that are too high for the current channel conditions. By analyzing *Minstrel's* rate choices we can see than *Minstrel* attempts to use high rates too often in a rapidly changing wireless channel. We also looked at gradual changes in channel quality, and discovered that *Minstrel* has poor performance when the wireless signal strength deteriorates from high to low. Conversely, *Minstrel* outperforms fixed rate when the signal strength increases from low to high. Again, this is due to *Minstrel's* tendency to select high rates. This behavior is undesirable when the channel conditions are deteriorating, and *Minstrel* pays a performance penalty. However it does give *Minstrel* the ability to quickly make use of higher rates when the channel conditions improve.

VII. ACKNOWLEDGEMENT

We are grateful to InternetNZ for its support in this project.

REFERENCES

[1] Minstrel Rate Adaptation Algorithm Documentation, http://madwifi-project.org/browser/madwifi/trunk/ath_rate/minstrel/minstrel.txt

[2] D. Xia, J. Hart and Q. Fu, "On the performance of rate control algorithm minstrel", *Proceedings of the 23rd IEEE International Symposium on Personal, Indoor and Mobile Radio Communications* (PIMRC), 9-12 September 2012, Sydney, Australia.

[3] A. Kamerman and L. Monteban, "WaveLAN II: A High-performance Wireless LAN for the Unlicensed Band", *Bell Labs Technical Journal*, vol. 2, no. 1, Fall 1997, pp. 118-133.

[4] M. Lacage, M.H. Manshaei and T. Turletti, "IEEE 802.11 rate adaptation: a practical approach", *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM), 4-6 October 2004, Venice, Italy, pp. 126-134.

[5] ONOE Readme, http://madwifi-project.org/browser/madwifi/trunk/ath_rate/onoe

[6] B. Sadeghi, V. Kanodia, A. Sabharwal and E. Knightly, "Opportunistic media access for multirate ad hoc networks", *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking* (MobiCom), 23-28 September 2002, Atlanta, Georgia, pp. 24-35.

[7] J. Kim, S. Kim, S. Choi and D. Qiao, "CARA: collision-aware rate adaptation for IEEE 802.11 WLANs". *Proceedings of the 25th IEEE International Conference on Computer Communications* (INFOCOM), 23-29 April 2006, Barcelona, Spain.

[8] D. Qiao and S. Choi, "Fast-responsive link adaptation for IEEE 802.11 WLANs", *Proceedings of IEEE International Conference on Communications* (ICC), 16-20 May 2005, Seoul, Korea, vol. 5, pp. 3583-3588.

[9] H. Rahul, F. Edalat, D. Katabi and C. Sodini, "Frequency-aware rate adaptation and MAC protocols", *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking* (MobiCom), 20-25 September 2009, Beijing, China, pp. 193-204.

[10] M. Vutukuru, H. Balakrishnan and K. Jamieson, "Cross-layer wireless bit rate adaptation", *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 17-21 August 2009, Barcelona, Spain, pp. 3-14.

[11] D. Qiao, S. Choi and K.G. Shin, "Goodput analysis and link adaptation for IEEE 802.11a wireless LANs", *IEEE Transactions on Mobile Computing*, vol. 1, no. 4, October 2002, pp. 278-292.

[12] W. Yin, K. Bialkowski. J. Indulska and P. Hu, "Evaluations of madwifi MAC layer rate control mechanisms", *18th International Workshop on Quality of Service* (IWQoS), 16-18 June 2010, Beijing, China.

[13] W. Yin, P. Hu, J. Indulska and K. Bialkowski, "Performance of mac80211 rate control mechanisms", *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile systems* (MSWiM), 31 October – 4 November 2011, Miami, Florida, pp. 427-436.

[14] J. Bicket, "Bit-rate Selection in Wireless Networks", MSc thesis, Dept. of Electrical Engineering and Computer Science, MIT, February 2005.

[15] E. Ancillotti, R. Bruno and M. Conti, "Experimentation and performance evaluation of rate adaptation algorithms in wireless mesh networks", *Proceedings of the 5th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks* (PE-WASUN), 27-28 October 2008, Vancouver, Canada, pp. 7-14.

[16] S.H. Wong, H. Yang, S. Lu and V. Bharghave, "Robust rate adaptation for 802.11 wireless networks", *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking* (MobiCom), 24-29 September 2006, Los Angeles, CA, pp. 146-157.

[17] P.A.K. Acharya, A. Sharma, E. Belding, K.C. Almeroth and K. Papagiannaki, "Rate adaptation in congested wireless networks through real-time measurements", *IEEE Transactions on Mobile Computing*, vol. 9, no. 11, November 2010 pp. 1535-1550.

[18] OpenWrt Backfire Readme, http://backfire.openwrt.org/10.03/README