Ruckus WiFi Evaluation

Stuart Cheshire, B.A., M.A., M.Sc., Ph.D.
<publications @ stuartcheshire.org>

23rd April 2006

Revised 29th December 2006

Introduction

According to Ruckus Wireless, higher transmission power and intelligent use of a six-element antenna array give their MF2900 (base station) and MF2501 (client) 802.11g wireless devices superior performance compared to existing 802.11g products. By individually enabling or disabling each element independently, the device has 63 different software-controllable antenna configurations, any one of which may be the best under a given set of circumstances. For example, powering two adjacent elements simultaneously has the effect of forming a directional beam, giving increased range in that direction, and increased noise rejection of signals from other directions.

I discovered that the Ruckus devices do indeed achieve superior performance in three areas:

1. Increased range.
2. Increased throughput at any given range.
3. More consistent throughput.

In my mind it is the third aspect, more consistent throughput, that is the most significant benefit for audio, video, and other modern applications. Most of today's WiFi devices exhibit pretty variable throughput. Over the course of a minute their throughput can fluctuate fairly dramatically as environmental conditions change, such as people and pets moving around, or interference from cordless telephones and microwave ovens. This variability can be problematic for playback of time-based media such as audio and video. While lulls in network throughput can be survived by the use of sufficient buffering at the playback end, large buffers are undesirable both because they increase hardware cost and because they result in less responsive user interaction.

By switching antenna configuration several times per second, in conjunction with techniques to monitor how well the wireless link is performing from moment to moment in terms of application-layer traffic, Ruckus devices are able to avoid getting stuck in a low-throughput rut for extended periods of time. All WiFi networking is inherently variable, meaning that instantaneous throughput of any given link forms a probability distribution. Considering only the average of that distribution is a gross simplification. By focusing not only on improving the mean, but also on improving the low end of the distribution, Ruckus devices improve the worst-case scenario that audio and video devices have to be engineered to cope with. For example, for a video playback device playing a 10Mb/s data stream over WiFi, knowing that it can rely on getting 10Mb/s 99.99% of the time is a lot more useful than knowing it can get 20Mb/s 50% of the time (but maybe only 1Mb/s 5% of the time).

About the Author

I'm knowledgeable about networking from the Ethernet layer up, but I'm not an expert on analog radio-frequency electronics. Accordingly, I treated the Ruckus devices as "black boxes". I'm not concerned so much with what they do at the radio-frequency level or how they do it; I'm interested in measuring the end results of whatever they do, to see if it yields tangible benefits at the IP-layer and above. As such, I may have garbled some of the explanations of how they work. For Ruckus's own explanations of how their BeamFlex [BeamFlex] and SmartCast [SmartCast] technologies work, please consult the Ruckus web site.

I'm not affiliated in any way with Ruckus Wireless, nor have I received any payment for these experiments. My scientific curiosity was caught by their claims, and I was interested to investigate for myself. Here I present the results of my own experiments with the Ruckus MF2900 (base station) and MF2501 (client).

Evaluation

For the tests, I used AirTunes music streaming from iTunes to an AirPort Express base station. The test was not simply to see whether or not it worked — the 1Mb/s required by AirTunes is not hard to achieve — but to examine the packet traces to see how *easily* the required throughput was achieved. The AirPort Express base station includes several seconds of buffering to accommodate occasional periods of network slowness, so it is rare for the network to be so bad that playback suffers user-perceivable glitches. However, examination of the packet traces offers insight into how heavily the base station had to fall back on that buffer to survive occasional 'sags' in throughput. It is reasonable to assume that these results at 1Mb/s are roughly representative of the results we would see at any desired data rate (e.g., 5-10Mb/s for DVD-quality MPEG-2 video), though of course higher data rates would be achieved at correspondingly shorter wireless distances. I chose 1Mb/s AirTunes streaming because streaming music from iTunes to AirPort Express is a common real-world application of wireless networking used by countless iTunes users today on both Mac and Windows.

The music source was a Power Mac G5 Quad (two dual-core 2.5GHz G5 processors) with 2GB RAM, Mac OS X 10.4.6, and iTunes 6.0.4.

The playback device was an Apple AirPort Express wireless base station running firmware version 6.3.

Music was played from iTunes to a single AirPort Express. This engages the original TCP-based mode of "AirTunes" music streaming, which makes the packet trace amenable to analysis using tcptrace [tcptrace]. Playing music to more than one AirPort Express (or simultaneously via the computer's built-in speakers and one or more AirPort Expresses) engages the newer UDP-based mode of "AirTunes" music streaming. The new AirTunes UDP mode is intended to recover from packet losses better than TCP, but the issues are complex and the question has not yet been definitively answered, which is why the TCP mode is still supported. In any case, there's no equivalent to tcptrace for analyzing an AirTunes UDP flow, so these experiments were performed using the TCP mode.

Inspection of the tcptrace plots yields insights into the behaviour of the underlying WiFi network, and hence provides guidance that applies to any transport protocol attempting to deliver data over that network. If we find in any given test that no packets were lost, then for that test the question of retransmission is moot, and indeed any retransmission would merely have been a waste of bandwidth. In addition, if we see that a tcptrace plot shows that

packets were lost, but TCP retransmitted every one promptly (on the order of one network round-trip time), and retransmitted no unnecessary packets, then it's trivial to see that this is the theoretical optimum behaviour for maximizing goodput and timeliness, and consequently there's no way any other transport protocol, however creative or specialized, could have done better. In fact the TCP "fast retransmit" mechanism did generally do a good job of retransmitting data promptly, but in a couple of cases where fast retransmit was not invoked there was a one-second timeout, which is unfortunate. However, this fixed one-second minimum timeout is an artifact of the FreeBSD-derived TCP stack in Mac OS X, rather than being an intrinsic limitation of the TCP protocol design itself.

AirTunes sends music encoded using the Apple Lossless codec, which typically achieves compression amounts between 0% and 50% depending on the nature of the audio. For these tests, the 205-second track "Kids in America" from "Kim Wilde: The Singles Collection 1981-1993" was played. The raw size of this file as 16-bit stereo 44.1kHz data is 34.59MB. Compressed with Apple Lossless, the size is 25.88MB, a reduction of 25%. To play this song without audio dropouts or similar glitches, the network has to sustain a goodput data rate of at least 1.06Mb/sec. The song was imported into iTunes in Apple Lossless format, to avoid possible file-size artifacts that could be introduced by importing as MP3 or AAC and then converting to Apple Lossless at playback time.

Packet traces were captured at the source using tcpdump:

```
sudo tcpdump -i en0 -w filename.dmp tcp port 6000
```

Packet traces were analyzed by using tcptrace 6.6.7 [tcptrace] to create a round-trip time plot, a throughput plot, and a TCP time/sequence plot, which were viewed using jPlot [jPlot]:

```
tcptrace --nores_addr -lr -T -A100 -y -R -S -zxy filename.dmp
java -jar jPlot.jar a2b_rtt.xpl a2b_tput.xpl a2b_tsg.xpl
```

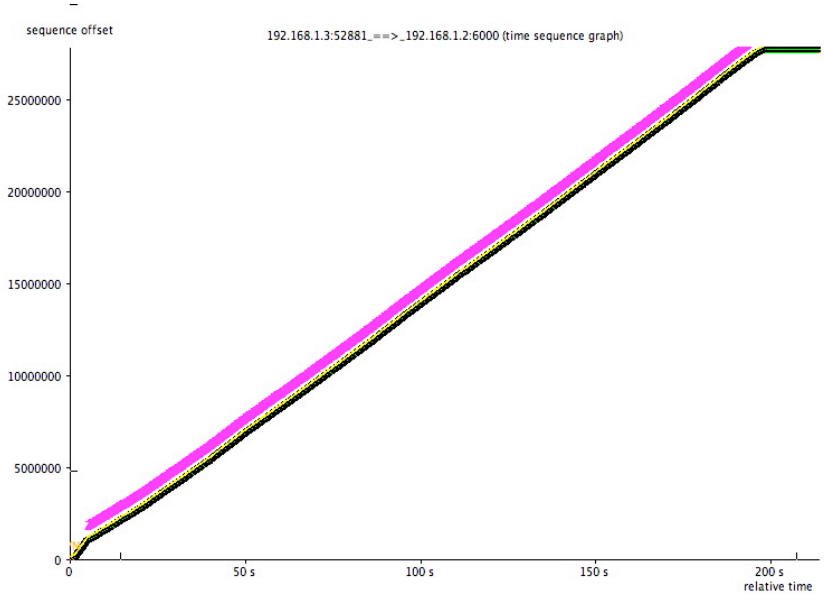
In the various tests, the music from the G5 to the AirPort Express takes different paths. In the first test, the control, the G5 and AirPort Express are directly connected via Ethernet. As expected on a lightly loaded Ethernet, no packets were lost, and the music played flawlessly. Subsequent experiments tested communication from G5 to AirPort Express via the Ruckus WiFi hardware, via an AirPort Extreme base station, and related variations.

For fully understanding the plots, familiarity with TCP and tcptrace [tcptrace] is helpful. In brief:

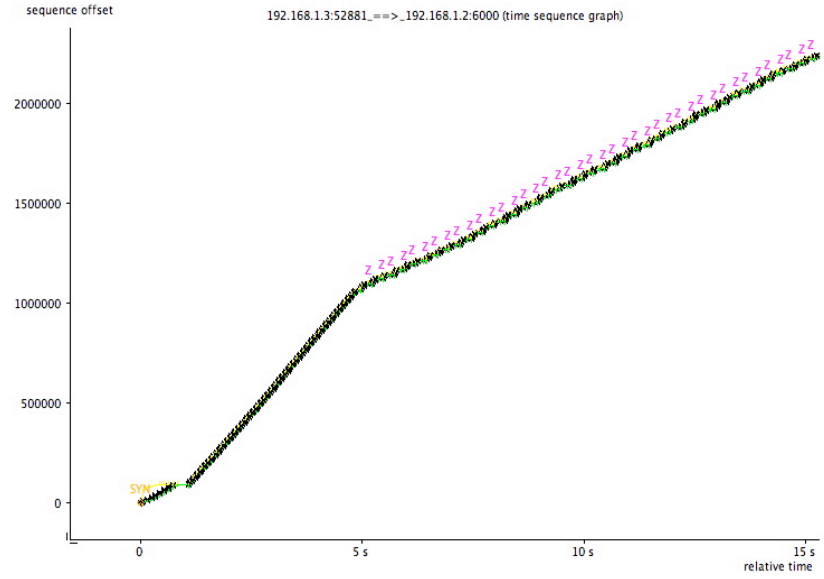
1. TCP time/sequence plot shows time on the horizontal axis, and TCP sequence numbers (i.e. aggregate bytes transferred) on the vertical axis.
2. Each vertical black line with arrowheads represents a single transmitted packet; the height indicates the number of data bytes carried by the packet, and the horizontal position indicates the time it was sent.
3. The green line shows what sequence numbers have been acknowledged by the receiver. No legal packet (black line) should appear below the green line. The horizontal distance from a packet (black line) to the point where the green line pops up to a level at or above the top of that black line gives an indication of network round-trip delay.
4. The yellow line indicates the upper limit of the receiver's offered window. No legal packet should appear above the yellow line.
5. The packets, acks, and window ceiling all steadily trending upwards to the right indicate a nice smooth reliable packet flow.

1. G5 Ethernet to AirPort Express via Ethernet

The three-minute TCP time/sequence plot for the Ethernet connection shows, not surprisingly, a textbook-perfect, uneventful, data transfer.



A close-up of the initial part of the plot reveals detail about what's happening. For about the first second, iTunes is performing some connection setup with the AirPort Express. For the next four seconds, iTunes is sending data to the AirPort Express faster than real time to fill up the playback buffer. Then the playback buffer is full, and for the remaining three minutes, the transfer proceeds at exactly real time. Every time the AirPort Express plays one block of data from its buffer (approximately 9000 bytes, 6 or 7 TCP packets) it reads one more block of data from the TCP connection, the AirPort Express's TCP stack raises its receive window ceiling by that much, and iTunes responds by providing the next block of data.

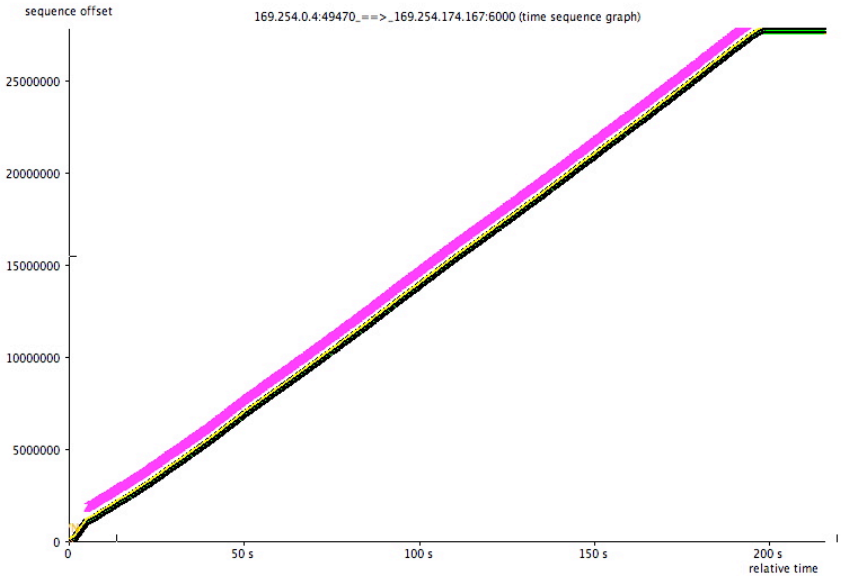


Tcptrace reported no retransmitted packets. Counting only full-sized non-retransmitted TCP segments, the minimum, average and maximum round-trip times reported were: 1.2, 49.7, 120.1 ms, with a standard deviation of 22.2 ms.

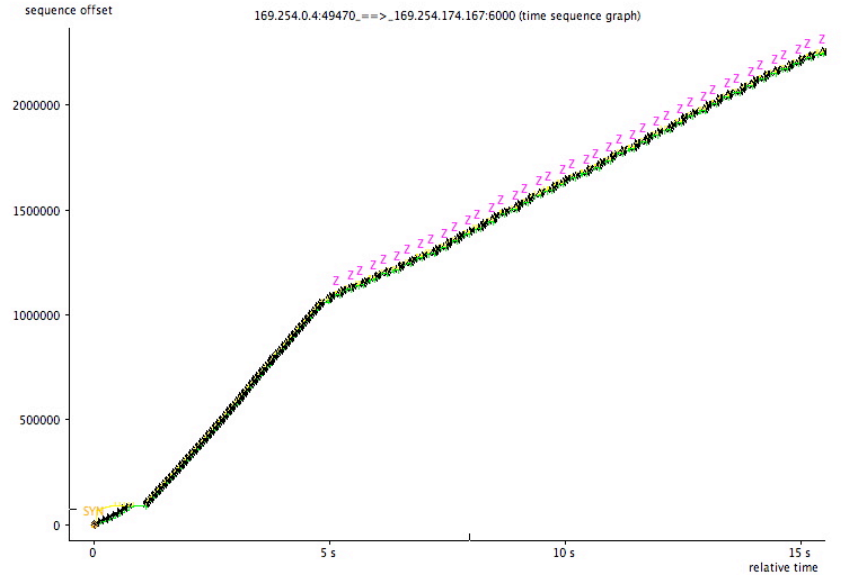
2. Ruckus 2900 to Ruckus 2501

In this test the G5 was connected via Ethernet to the Ruckus 2900 base station. The Ruckus 2501 client was associated via 802.11g to the Ruckus 2900 base station, and the AirPort Express was connected to the Ruckus client via Ethernet.

In this configuration it was hard to find a setup that offered any interesting challenge to the Ruckus hardware. In the end, to approach the range limit, I had to move the Ruckus 2501 client and AirPort Express to another house altogether. With the Ruckus client and base station approximately 160 feet apart, with the wireless signal traveling through several walls (including two exterior walls) the Ruckus software finally reported "poor" signal quality. However, even with "poor" signal quality, AirTunes still played perfectly.



A close-up of the initial part of the TCP time/sequence plot looks indistinguishable from the direct Ethernet test.



Tcptrace reported no retransmitted packets. The minimum, average and maximum round-trip times reported were: 19.7, 46.2, 105.7 ms, with a standard deviation of 15.9 ms.

3. Ruckus 2900 to AirPort Express

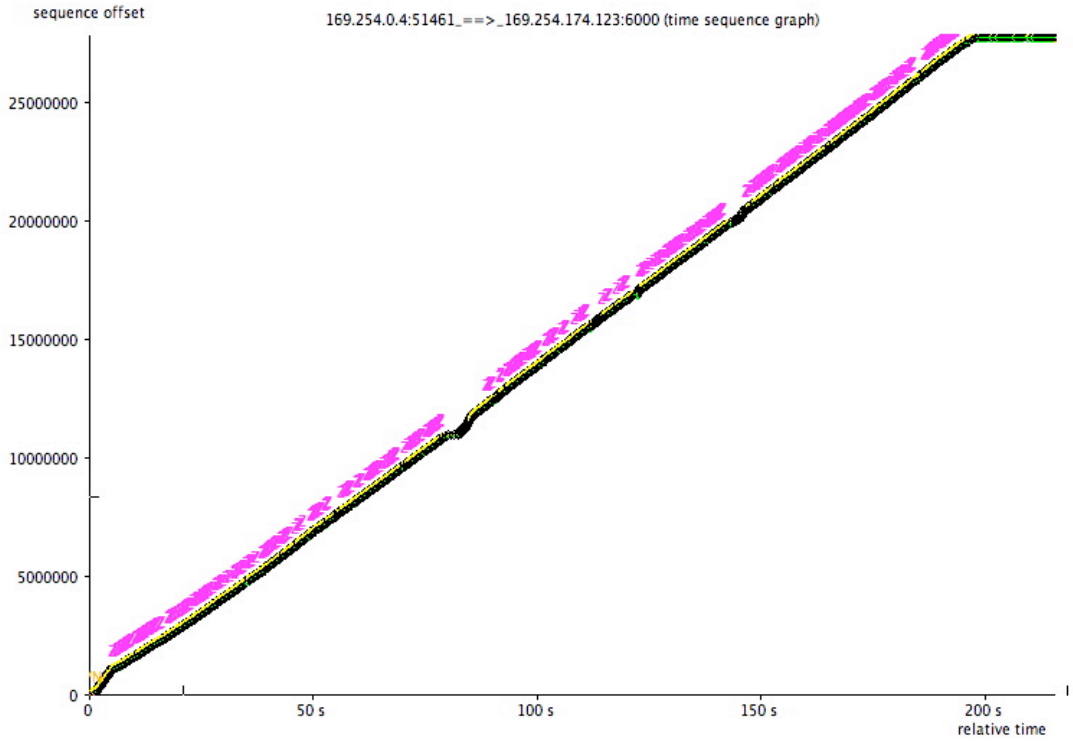
For the next test, I reconfigured the AirPort Express to communicate directly with the Ruckus 2900 base station. However, the AirPort Express couldn't see the Ruckus 2900 base station. The "AirPort Admin Utility" tool showed no available networks for the AirPort Express to join, even tough the Ruckus 2501 client right next to the AirPort Express could see the Ruckus 2900, my home AirPort Extreme, and *five* other WiFi networks, presumably in other nearby houses. This would seem to confirm that the Ruckus hardware really does have superior sensitivity and range — even when communicating with standard WiFi base stations, the Ruckus 2501 client was able to 'see' more networks than the AirPort Express. I moved the AirPort Express a few feet closer to the Ruckus 2900 base station to get it in range.

Even before considering the time/sequence plot, the summary statistics reported by tcptrace showed obvious differences: the minimum, average and maximum round-trip times reported were: 6.8, 121.3, 2379.7 ms, with a standard deviation of 201.9 ms. The 2379.7 ms figure is interesting. Tcptrace calculates round-trip time statistics only from non-retransmitted packets. This is because an ack for a retransmitted packet is ambiguous — is it acknowledging receipt of the retransmitted packet, or is it acknowledging receipt of the original packet, which arrived late? From this fact we can conclude that at least one non-retransmitted packet took over 2.3 seconds to be delivered. The packet was not lost and retransmitted after a couple of seconds delay; it was transmitted just once, and it simply took the wireless network over 2.3 seconds to deliver it to its destination. Had TCP (or any other transport protocol) decided to retransmit this packet, that would have been a mistake, and would have squandered already-scarce wireless bandwidth with pointless duplication of data.

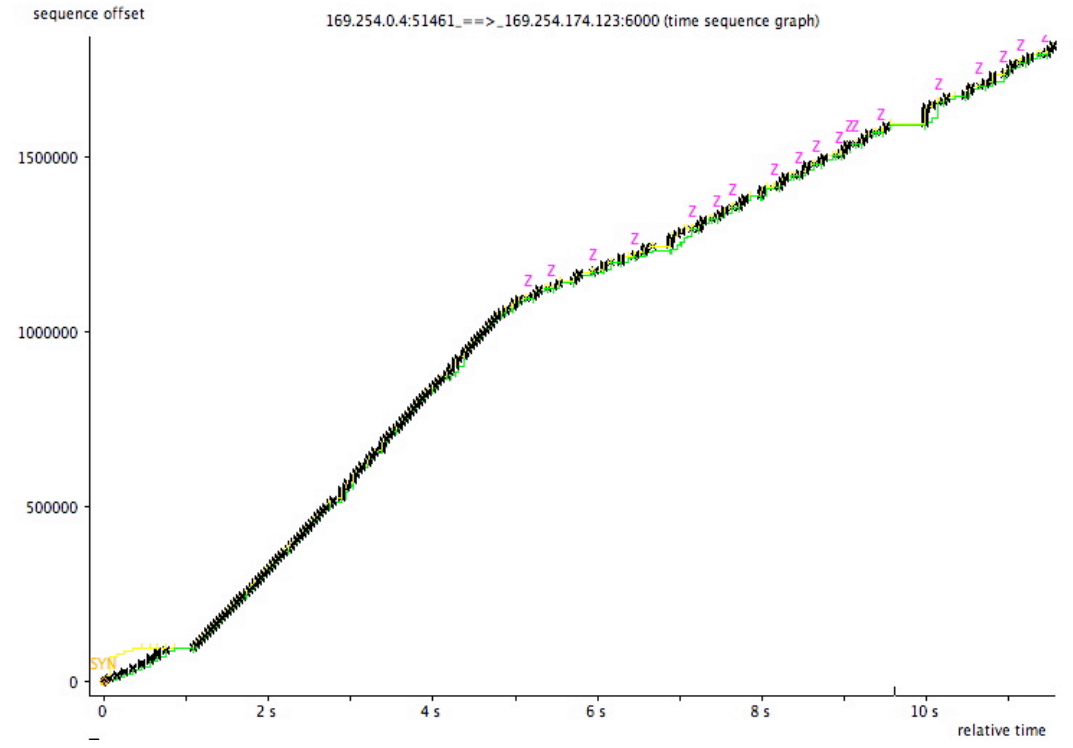
Tcptrace did report a single retransmission. As described at the end of this section, TCP's "fast retransmit" mechanism recovered from this loss perfectly, retransmitting only the single lost packet, retransmitting it promptly as soon as the receiver notified the sender of the loss, and doing so without halting or delaying the ongoing smooth data flow of new packets. The reason that IP-level packet losses are rare is that the WiFi layer has its own link-layer retransmission scheme, masking most losses. When a packet is lost due to interference or other problems, the WiFi link-layer will retransmit it automatically, stepping to a lower data rate if necessary. This means that wireless interference is usually manifested at the IP layer not as loss, but as reduced throughput and consequently increased delay.

It is rare for there to be so much interference that the WiFi link-layer actually gives up on a packet and allows the loss to be visible to the higher layers. This means that when WiFi performance is marginal, this is usually seen not as losses and retransmissions, but a reduced throughput and increased delay. On the TCP time/sequence plot, reduced throughput is evidenced by the slope of the green ack line becoming shallower. The slope of the green ack line is an indication of the rate at which packets are arriving at the receiver; when it gets shallower that means the arrival rate has dropped. When the throughput drops, packets spend more time sitting in the queue waiting to be transmitted; this it what causes the delay experienced by any given packet to increase. On the TCP time/sequence plot, increased delay is evidenced by a widening horizontal gap between a packet and its corresponding ack. The net result of these two effects is the appearance of visual 'sags' in the time/sequence plot when the network is struggling.

Even from just cursory visual inspection it is clear that this TCP time/sequence plot is not as "smooth" as the first two:

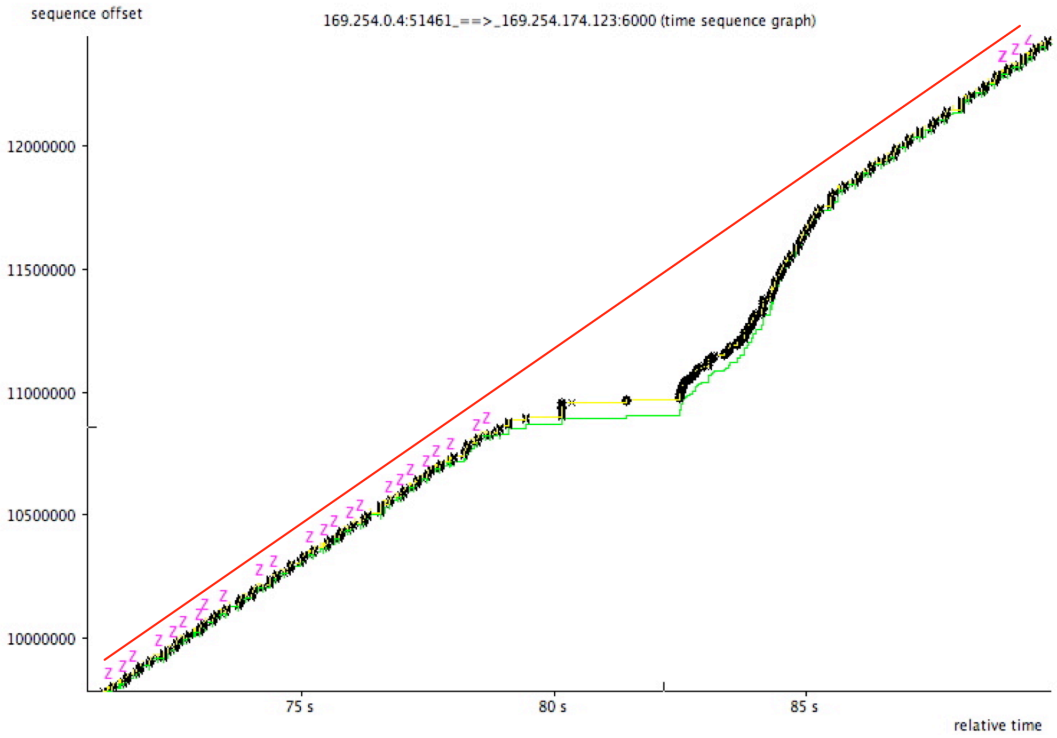


A close-up of the initial part reveals similar visual evidence that the data flow is a little intermittent and jerky instead of being steady and smooth.



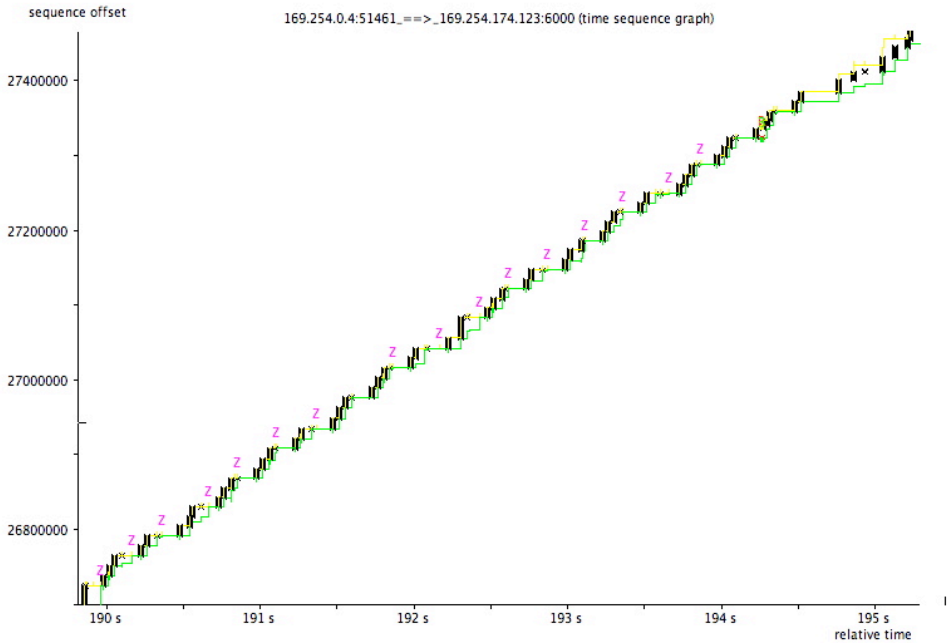
At various points on the overview plot, visual ‘sags’ in the straight line are evident. Given that this is a three-minute trace, any anomalies large enough to be visible at this scale indicate severe problems. Zooming in on any part of the trace reveals countless smaller ‘mini-sags’ indicating that the network was constantly having trouble keeping up with required data rate.

The plot below focuses on the large sag approximately 80 seconds into the trace:

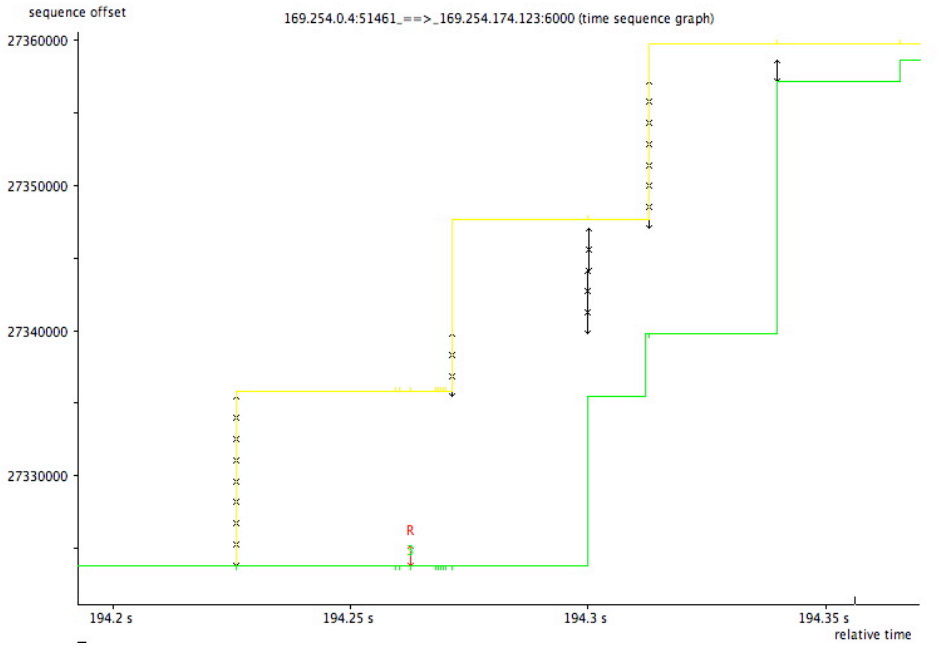


At around the 79-second mark, some variation in network conditions causes the packet rate to drop precipitously, as evidenced by the sudden slowdown in acknowledgements. At around the 82.5-second mark, the network conditions recover, and the waiting packets are delivered, as evidenced by the rapid arrival of acknowledgements for those packets. From 82.5-86.0 seconds, the plot shows a steeper slope, roughly twice real-time, as the connection makes up for lost time, sending as fast as it can until the playback buffer is full again. The superimposed red line shows the general trend of the slope, and, as expected, once the connection has caught up again after the interference, the packets end up back on the same line they would have been on had the interference not occurred. In this case, no user-perceivable audio disruption occurred, since the playback buffer in the AirPort Express base station is larger than 3.5 seconds, so it was able to weather this particular storm.

Before concluding this section, it is useful to examine the one TCP packet loss that occurred in this test. The plot below shows an apparently steady data transfer:



However, zooming in on the time period around 194.25s shows that a packet was actually lost:



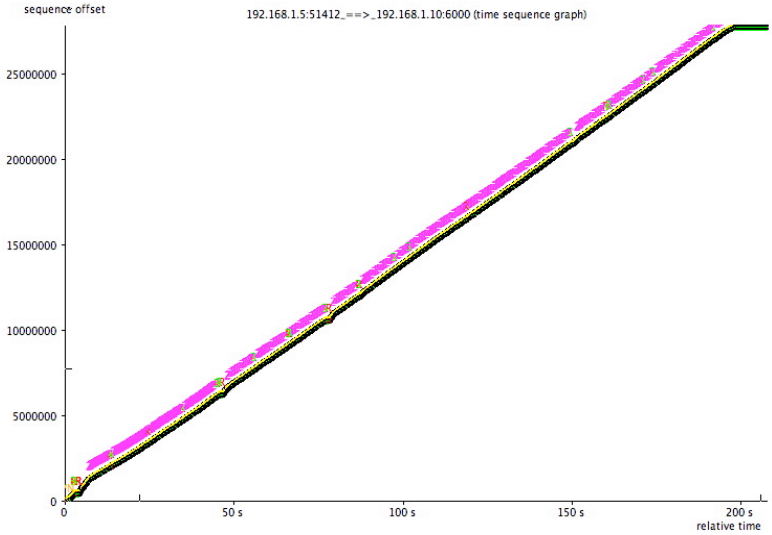
At 194.22s, the original packet was sent. At 194.26s, the arrival of a stream of identical acks from the receiver informs the sender that this packet was lost, and invokes TCP's fast retransmit mechanism. The single lost packet is retransmitted, while normal transmission of new data continues uninterrupted. At time 194.30s a cumulative ack is received, acknowledging both the lost packet and seven subsequent packets (which arrived at the receiver *before* the retransmission of the lost packet, and were held by TCP until the retransmission arrived).

4. AirPort Extreme 802.11g to AirPort Express

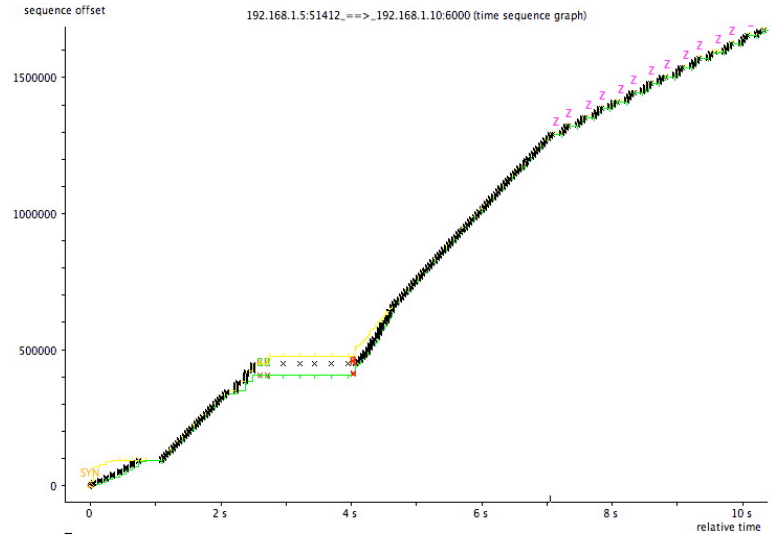
For this test I associated the AirPort Express with my AirPort Extreme 802.11g base station. For the AirPort Express to get a signal I had to move it still closer to the base station. Inside the second house, the AirPort Express was not able to detect the signal from the AirPort Extreme at all. I moved it outside, so it was about 100 feet from the base station, with only one exterior wall in the way.

In this case the signal was very weak, and the behaviour was quite different. For some reason, the base stations seemed a lot more willing to give up on packets and simply discard them. Tcptrace reported 35 retransmitted packets, but, perhaps as a result of not trying so hard, the round-trip times were better. The minimum, average and maximum round-trip times reported were: 2.6, 51.5, 144.4 ms, with a standard deviation of 27.4 ms. These figures (at a distance of 100 feet) are better than the Ruckus-to-AirPort Express test (140 feet) but still worse than the initial Ruckus-to-Ruckus test (160).

The TCP time/sequence plot shows several glitches:



In particular, there was a glitch during the initial buffer filling. This is the kind of event that would cause problems for any device that optimistically decided to start playback before it had built up its full playback buffer:



5. AirPort Extreme 802.11g to AirPort Express with "Interference Robustness"

The AirPort configuration software offers an option called "Interference Robustness". It's not documented clearly exactly what this does, and indeed a Google search reveals much confusion and speculation on the subject. The consensus seems to be that it reduces receiver sensitivity to improve noise rejection, and disables some "good neighbor" behavior when transmitting, so that the base station ignores competing signals and transmits anyway, instead of waiting its turn. The implication would seem to be that Interference Robustness improves your own performance at the expense of those around you sharing the spectrum. In the case where those around you sharing the spectrum are other people trying to use WiFi, if everyone decided to enable Interference Robustness this would seem to have the potential to make performance and reliability worse for all concerned, so everyone loses. In the case where the competing interference is coming from a microwave oven, ignoring it and transmitting at the same time is probably not going to affect the microwave oven's ability to cook your food. In any case, I decided to give Interference Robustness a chance and see if it would improve performance. With Interference Robustness enabled, tcptrace reported fewer retransmitted packets — 18 instead of 35 — but the round-trip times were a little worse. The minimum, average and maximum round-trip times reported were: 2.2, 53.3, 162.1 ms, with a standard deviation of 28.5 ms.

6. AirPort Extreme 802.11g at 500mW to AirPort Express

Although the FCC rules for the 2.4GHz ISM band allow Point To Multipoint transmitters up to 1W with an appropriate antenna [SeattleW], consumer 802.11 base stations like Apple's AirPort typically have a maximum transmit power of 32mW. This is partly because of cost considerations (higher-powered transmitter components are more expensive) but also because higher transmitter power can be counter-productive. 32mW is plenty for a school classroom, an apartment building, or a small house. In fact, many school installations have the transmit power intentionally set even lower than the factory-default 32mW, because what they want to do is to maximize total bandwidth, by giving every classroom its own dedicated short-range base station, rather than having everyone share the limited bandwidth of a single high-power base station serving the whole school.

More expensive enterprise-market base stations such as Cisco's are available with transmit power up to 100mW.

According to Ruckus Wireless's web site, the RF power output is configurable up to "23 dBm for 802.11b/g (200mW)" [specs]. Their product information page says, "The Ruckus MetroFlex is built with a super high output power amplifier that provides twice the output power of traditional Wi-Fi systems." [info]

Some of the Ruckus gateway's superior performance is probably due to their six-element antenna array, but some of it is almost certainly a result of simple brute-force power. To level the playing field a little, I boosted the AirPort base station's power using a 500mW external amplifier with a 4 dB omni-directional antenna [booster].

In this setup, tcptrace reported 14 retransmitted packets, down from 18 in the previous test. The minimum, average and maximum round-trip times reported were a little worse: 2.1, 56.4, 206.6 ms, with a standard deviation of 32.4 ms.

7. AirPort Extreme 802.11g at 500mW to AirPort Express with “Interference Robustness”

For the final combination, I tried combining all the options available to the AirPort Extreme: 500mW brute-force amplifier, 4dB gain external antenna, plus Interference Robustness enabled.

In this setup, tcptrace reported 9 retransmitted packets, down from 14 in the previous test. The minimum, average and maximum round-trip times reported were: 2.5, 53.9, 609.0 ms, with a standard deviation of 36.9 ms. The 500mW amplifier appears to have reduced packet loss at the expense of a poorer worst-case delay.

Conclusions

The Ruckus devices really do seem to work every bit as well as Ruckus Wireless claims. I'd expected the performance gains to be modest, difficult to ascertain with certainty in the face of all the other real-world variables that affect wireless performance. In reality, the differences were so dramatic that there was little doubt. The table below summarizes the results of each test, including number of retransmissions, average round trip time, and worst-case round trip time.

Astonishingly, for the Ruckus-to-Ruckus wireless test at 160 feet, both the worst-case and average round-trip time were better even than the Ethernet test. This seems inconceivable, given that the Ruckus devices were themselves connected via Ethernet at both ends.

My hypothesis to explain this anomaly is that I performed the Ethernet test on my existing home network, whereas the Ruckus tests were performed using a direct connection to the G5's second Ethernet port. I had assumed that when sending just 1Mb/s over 100Mb/s Ethernet, a little other traffic would make no significant difference. Indeed, the other traffic didn't cause any actual problem, but it did apparently have enough effect to be statistically observable in the results. To test this hypothesis, I repeated the first Ethernet test by connecting the AirPort Express directly to the G5's second Ethernet port. Those results are shown on line eight below, and do indeed show that this improves the average and worst-case round trip times slightly, enough to beat the 160-foot Ruckus-to-Ruckus test.

Finally, it was clear from the results that even at 160 feet, the Ruckus-to-Ruckus link was outperforming all the other wireless configurations at just 100 feet. I did one last Ruckus-to-Ruckus test, this time at a distance of 200 feet, with multiple interior and exterior walls, trees, and other obstructions in between. These results are shown on line nine below. As we would expect, they are a little worse than the 160-foot test, but there were still no packet losses, and amazingly, the average round-trip delay for Ruckus at 200 feet still managed to beat all the AirPort configurations at half the distance, even the test where AirPort was given the benefit of a 500mW booster, an external antenna, and the "Interference Robustness" feature enabled.

Experiment	Range	Re-xmits	Min	Avg	Max	Std Dev
1. Ethernet (home network)	n/a	0	1.2	49.7	120.1	22.2
2. Ruckus to Ruckus (160)	160	0	19.7	46.2	105.7	15.9
3. Ruckus to AExp	140	1	6.8	121.3	2379.7	201.9
4. AirPort Extreme	100	35	2.6	51.5	144.4	27.4
5. AirPort Extreme + Interference Robustness	100	18	2.2	53.3	162.1	28.5
6. AirPort Extreme + 500mW	100	14	2.1	56.4	206.6	32.4
7. AirPort Extreme + 500mW + Interference Robustness	100	9	2.5	53.9	609.0	36.9
8. Ethernet (dedicated)	n/a	0	1.2	40.9	100.0	12.0
9. Ruckus to Ruckus (200)	200	0	11.5	49.0	282.3	25.7

Other Observations about the Ruckus Devices

In terms of raw 802.11 link-layer performance, the Ruckus devices beat the Apple AirPort base stations. However, in terms of user-friendliness, the Ruckus devices fell short. A couple of times while trying to set up both the Ruckus 2900 base station and the Ruckus 2501 client, I mis-typed a configuration parameter, rendering the device completely unusable, leaving me hunting for a paper clip to poke in the reset hole to get it back to factory-default state. All it would take to significantly improve the reliability, robustness, and user-friendliness of the Ruckus devices would be some simple software refinements to incorporate the Zeroconf networking techniques [Zeroconf] (what Apple calls "Bonjour").

Automatic Link-Local Addressing

The Ruckus devices currently ship configured with a fixed hard-coded IP address.

Instead of using fixed hard-coded IP addresses, Ruckus devices should use IPv6 link-local addresses [RFC 2462] or IPv4 link-local addresses [RFC 3927], or both.

Fixed hard-coded addresses have a several significant drawbacks. One is that you if you connect two Ruckus devices using the same fixed hard-coded IP address to the same network, they conflict with each other, and neither device can establish reliable IP-based communications. A related issue is that by hijacking a 192.168.x.x IP address — an IP address explicitly reserved for allocation by the end-user [RFC 1918] — if the user connects a Ruckus device to a network where some other device is already legitimately using that address, the Ruckus device will conflict with the legitimate device, disrupting the user's network.

Finally, using a particular fixed unicast address means that to communicate with the Ruckus device from a computer (e.g. to configure the device) requires that the computer be configured to have an IP address in the same logical subnet as the Ruckus device's fixed hard-coded addresses. If the computer in question doesn't have an address in the correct logical subnet, reconfiguring it to a different address is disruptive to the computer (e.g. if the user then loses connectivity with the web site where he or she was reading the instructions how to configure the Ruckus device). If the computer in question *does* already have an address in the correct logical subnet, then this implies that the local network is already configured to use the 192.168.x.x address range, and it is very likely that the Ruckus device using address 192.168.0.1 has just disrupted operation of the user's existing network gateway with that address, thereby breaking Internet access for *all* machines on the local network. This puts the Ruckus device in a "Catch 22" situation — either it can't communicate with the user's computer without onerous reconfiguration, or it *can* communicate with the user's computer, but it takes down the entire network in the process.

Link-local addresses solve these problems. Link-local addresses are guaranteed to not conflict with any other device, and link-local addresses are guaranteed to always be reachable, independent of configuration or misconfiguration. Since a link-local address is, by definition, known to be on the local link, the user doesn't have to change their computer's IP address to make a link-local destination address reachable. In today's world, IPv6 link-local addresses are preferable, and are fully supported by Mac OS X, Windows XP, Linux, FreeBSD, Solaris, etc. If a device has limited firmware that simply doesn't support IPv6, then in today's world IPv4 link-local addresses are an acceptable alternative.

(If this point needs any further reiteration, consider that the instructions given in the Ruckus MF2501 Quick Setup Guide don't actually work. The Quick Setup Guide instructs you to set your computer to address 192.168.1.100/24, but the default IP address for the MF2501 is 192.168.0.254/24, which is a different logical subnet, so this is a non-working configuration.)

In contrast, the Apple AirPort base stations do support IPv6 link-local addresses, in conjunction with Bonjour (DNS Service Discovery [DNS-SD] and Multicast DNS [mDNS]) which means that there is simply no way for the user to accidentally misconfigure them at the IP level such that this IPv6 link-local address becomes non-functional.

Network Names

The Ruckus 2501 client wouldn't associate with a wireless network with an apostrophe in the name (e.g. "Stuart's network"). I had to change the name of my home network before the Ruckus 2501 client could connect to it. This seems like a simple bug that should be fixed.

It would be nice if the 2501 "Survey" screen offered the option to sort by various columns (e.g. sort by RSSI).

Safari Autofill

For some reason, the web administration login page on the Ruckus 2900 and 2501 fails to trigger Safari's autofill feature. It can be a big time-saver to have Safari autofill the necessary username and password every time the device asks for it, so it would be good if the Ruckus login page worked with this.

Asymmetric Hardware

The Ruckus MF2900 and MF2501 look externally identical, apart from the label on the bottom, yet one is apparently predestined to spend its life acting only as a base station, and the other only as a client. In contrast, an Apple AirPort Express can be configured by the user to act as a base station, or as a client of some other base station. There would be clear benefits to users, vendors, and installers if a Ruckus device could similarly be configured on-site to behave either way. That way you could never find yourself in the situation of having the wrong kind, or having two clients or two base stations by mistake. Especially given that the two devices look physically identical, distinguished only by a (not particularly memorable) model number, this seems like it could be an easy and frustrating mistake for an installer to make.

I have discussed these issues with Ruckus, and they plan to address them in future software releases. By the time you read this, these issues may already have been fixed.

References

[BeamFlex] http://www.ruckuswireless.com/technology/beamflex.php

[DNS-SD] http://www.dns-sd.org/

[info] http://www.ruckuswireless.com/products/metroflex/

[jPlot] http://tcptrace.org/jPlot/

[booster] http://www.macwireless.com/html/products/antennas_boosters/airport_boosters.php

[mDNS] http://www.multicastdns.org/

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996. http://www.ietf.org/rfc/rfc1918.txt

[RFC 2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998. http://www.ietf.org/rfc/rfc2462.txt

[RFC 3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005. http://www.ietf.org/rfc/rfc3927.txt

[SeattleW] http://www.seattlewireless.net/InterpretingFccRegulations

[SmartCast] http://www.ruckuswireless.com/technology/smartcast.php

[specs] http://www.ruckuswireless.com/products/metroflex/specs.php

[tcptrace] http://www.tcptrace.org/

[Zeroconf] http://www.zeroconf.org/

Revision History

23rd April 2006 First Version

29th December 2006 Added information about Ruckus's 200mW transmit power