Performance validation of the Madwifi driver

Francisco Esteves

15 de Fevereiro de 2013
What is the Madwifi?

- Open source project initially developed for the FreeBSD systems
- Designed initially to allow the creation of community wireless networks
- Exclusive for the 802.11 a/b/g network cards with Atheros® chipsets.
Madwifi’s Objectives

1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations

- Create an advanced driver for the Unix/Linux environments
- Create a new hardware independent 802.11 network stack for the Linux systems
Madwifi Evolution

Timeline:

- Official start in 2005
- Ended in the first quarter of 2012
- Starting point for the ath5k driver
Relevant features

- Allows monitoring mode to capture packets
- Create and use several VAPs working in parallel
- Obtain and configure several parameters through the Linux Terminal using the Linux wireless tools (Wireless Extensions)
The driver is dependent of the HAL (Hardware Abstraction Layer)

- Allows the implementation of new rate control algorithms
- Is constituted by several *kernel modules*
# Summary

1. About Madwifi

2. Source Code Organization

3. Transmission / Reception

4. Objectives for Validation

5. Parameters and Configurations

6. Experimental Demonstration

7. Performance Validation

8. Final Considerations

---

First contact...
Source Code Organization

Main Objectives:

- Identify the main structures
- Understand the allocation and access mechanism for each structure
- Identify the transmission/reception scheme

Obstacles:

- Source code written by several users
- Lack of documentation about the driver’s internal functioning and organization
Summary

1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations
Summary

1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations

Tx/Rx Scheme

Transmission:

- Configurable transmission waiting queues
- Possibility of setting different contention window values for each queue (QOS)
Driver’s tools included

How to know what is transmitted or received by the driver?

- 80211stats, athctrl, athstats, wlanconfig...etc

athstats:

- Possibility of obtaining statistical data at a specific time interval
Performance Validation

Objectives:

- Obtain detailed statistical data about each transmission / reception
- Control and obtain confirmation of the contention window’s values
- Confirm the driver’s transmission timings
- Real time readings of the transmission / reception statistics
- Change the contention window’s values in real time
- Validate the theoretical results through the experimental results
Hardware and Configuration

- 802.11 a/b/g DWL–G520 network card with AR5212 Atheros® chipset
- MAC → CSMA/CA with ACK
- Supported modulations → OFDM e CCK

Static configuration for each transmission:
- Pre-established 11 Mbps bit rate
- Only 802.11g connections
- No RTS/CTS or retransmissions
- Only UDP packets
Obtaining Statistical Data

Summary
1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations

Starting with the driver’s tools

Modification of the athstats tool to maximize the gathered statistical data

Transmission analysis:

Reception analysis:
Controlling the Contention Window

- Setting the transmission queue with different contention window values
- Verifying the contention window’s values through the HAL

Contestion window behavior:

Contestion window’s values limited by the HAL \( 2^n - 1 : n \in [1:10] \)
Transmission Timings

How to achieve confirmation for the transmission timings?

- Driver’s monitor mode
- Capturing packets with Wireshark

Confirmation about the headers, preambles, frame’s sizes and bit rates
Summary

1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations

Throughput Calculation

- For each transmission and with the given packet size, a specific throughput will be expected according to the contention window’s value

- Using the obtained timings (Wireshark)
  - PLCP Preamble
  - PLCP Header
  - MAC Header
  - IP
  - UDP
  - ACK
Summary

1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations

Modifying driver’s parameters in real time

- Need of communicating with the driver in real time
- New module implementation for reading/writing specific driver’s parameters
- Real time statistical reading through the Linux proc filesystem
- Real time control for the contention window
Experimental Demonstration

Summary

1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations
Performance Validation

Performing several transmission with different contention window values

Minimizing the interference levels (Faraday Cage)

Confirm the expected throughput for each transmission

Performance comparison between each experimental scenarios
Performance Validation

1. About Madwifi
2. Source Code Organization
3. Transmission / Reception
4. Objectives for Validation
5. Parameters and Configurations
6. Experimental Demonstration
7. Performance Validation
8. Final Considerations

Transmission:

<table>
<thead>
<tr>
<th>Common Room</th>
<th>Faraday Cage</th>
</tr>
</thead>
</table>

Reception:

<table>
<thead>
<tr>
<th>Common Room</th>
<th>Faraday Cage</th>
</tr>
</thead>
</table>
## Performance Validation

Comparison between the theoretical and the experimental values for each test situation:

### Throughput (Packet Size = 990 bytes)

<table>
<thead>
<tr>
<th>CW</th>
<th>Theoretical (mbps)</th>
<th>Common Room (mbps)</th>
<th>Error (%)</th>
<th>Faraday Cage (mbps)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.52</td>
<td>7.08</td>
<td>5.85</td>
<td>7.51</td>
<td>0.13</td>
</tr>
<tr>
<td>3</td>
<td>7.46</td>
<td>6.75</td>
<td>9.52</td>
<td>7.42</td>
<td>0.54</td>
</tr>
<tr>
<td>7</td>
<td>7.34</td>
<td>6.52</td>
<td>11.17</td>
<td>7.32</td>
<td>0.27</td>
</tr>
<tr>
<td>15</td>
<td>7.1</td>
<td>6.38</td>
<td>10.14</td>
<td>7.08</td>
<td>0.28</td>
</tr>
<tr>
<td>31</td>
<td>6.67</td>
<td>5.94</td>
<td>10.94</td>
<td>6.64</td>
<td>0.45</td>
</tr>
<tr>
<td>63</td>
<td>5.95</td>
<td>5.03</td>
<td>15.46</td>
<td>5.91</td>
<td>0.67</td>
</tr>
<tr>
<td>127</td>
<td>4.89</td>
<td>4.09</td>
<td>16.36</td>
<td>4.84</td>
<td>1.02</td>
</tr>
<tr>
<td>255</td>
<td>3.61</td>
<td>2.18</td>
<td>16.9</td>
<td>3.56</td>
<td>1.39</td>
</tr>
<tr>
<td>511</td>
<td>2.37</td>
<td>1.93</td>
<td>18.57</td>
<td>2.33</td>
<td>1.69</td>
</tr>
<tr>
<td>1023</td>
<td>1.4</td>
<td>1.14</td>
<td>18.57</td>
<td>1.38</td>
<td>1.43</td>
</tr>
</tbody>
</table>

Average %: 13.35  Average %: 0.79
Final Considerations

- Very good potential for implementing new solutions
- Unavoidable barriers due to the HAL’s dependency
- Transmission throughput controllable through the contention window
- Experimental results almost identical to the theoretical results in a controlled environment
Thank You for Your Attention!