Robust Rate Adaptation in 802.11 Networks

Starsky H.Y, Hao Yang, Songwu Lu and Vaduvur Bharghavan

Prepared by Meganne Atkins
Presented by Mostafa Uddin
Rate Adaptation

- Definition - The method used to dynamically select the transmission rate based on time-varying and locate independent channel quality

Goal: Optimize the transmission goodput at the receiver
The IEEE 802.11 Standard and Rate Adaptation

- Transmission Rates for 802.11 variations:
  - 802.11b 4 rate options
  - 802.11a 8 rate options
  - 802.11g 12 rate options

- Each Transmission Rate has different modulation and coding schemes

Rate Adaptation is critical to performance, but left undefined
Importance of Rate Adaptation

Assumption: Performance is defined as throughput.
Related Work

- Rate Adaptation Algorithms Metrics:
  - Probe Packets
    - ARF
    - AARF
    - SampleRate
  - Consecutive successes/losses
    - ARF
    - AARF
    - Hybrid Algorithm
  - Physical Layer metrics
    - Hybrid Algorithm
    - RBAR
    - OAR
  - Long-term statistics
    - ONOE

Commercially Deployed: ARF, SampleRate and ONOE
Solution Space for Rate Adaptation

- **Estimation**
  - Which layer to use?
  - Which message to use?
  - How to estimate?

- **Action**
  - Sequential rate adjustment
  - Best rate adjustment
Issues with Current Algorithms

- Current Metrics are limited in scope
- Simulations do not show flaws in the algorithms
- Performance loss
- 802.11 non-compliant algorithms
  - RBAR

Flawed design guidelines = Flawed algorithms
Current Design Guidelines

1. Decrease Transmission Rate upon severe packet loss
2. Use Probe Packets to assess new rate
3. Use consecutive transmission success/losses to decide rate increase/decrease
4. Use PHY metrics to infer new transmission rate
5. Long-term smothered operation produces best average performance
Experimental Methodology

- Programmable AP Platform
  - Supports 802.11 variations a/b/g
  - Per-frame Control functionality
  - Real-time tracing
  - Transmission rate control functionality
  - Low feedback delay
Experimental Methodology

- Experimental Setup
  - Static/Mobile Clients
  - 802.11 a/b
  - With/Without Hidden Stations
Guideline: Decrease Transmission Rate upon severe packet loss

Channel Conditions Worsen → Lower Transmission Rates

What if a hidden station exists?

<table>
<thead>
<tr>
<th></th>
<th>ARF</th>
<th>AARF</th>
<th>Sample Rate</th>
<th>Fixed Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodput (Mbps)</td>
<td>0.65</td>
<td>0.56</td>
<td>0.58</td>
<td>1.46</td>
</tr>
<tr>
<td>Loss Ratio</td>
<td>61%</td>
<td>60%</td>
<td>59%</td>
<td>60%</td>
</tr>
</tbody>
</table>

Goodput decreases!

Many packet loss scenarios exist, algorithms cannot be limited to fading/path loss
Guideline: Use Probe Packets to assess new rate

- Issues
  - Successful probes can be misleading
  - Unsuccessful probes can incur severe penalties

Small # probes → Inaccurate rate adaptation
Probing is too sensitive
Guideline: Use consecutive transmission success/losses to decide rate increase/decrease

- Statistically the success rate for this method is sub par
  - After 10 consecutive success (28.5%)
  - After back-to-back failures (36.8%)

Statistics are not substantial enough to base an algorithm on
Guideline: Use consecutive transmission success/losses to decide rate increase/decrease (cont’d)

Frame Retry – Turned Off
Rate Adaptation – Turned Off
Fixed Transmission Rate – For highest throughput

Realistic scenarios =
Randomly distributed loss behaviors
Guideline: Use PHY metrics to infer new transmission rate

Metrics can not be directly used to estimate transmission rates
Guideline: Long-term smothered operation produces best average performance

- Issues
  - Long-term rate estimation and rate change over time ≠ Best average performance

<table>
<thead>
<tr>
<th>Sampling intervals (ms)</th>
<th>5000</th>
<th>1000</th>
<th>500</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP Goodput (Mbps)</td>
<td>14.9</td>
<td>15.3</td>
<td>16.5</td>
<td>17.1</td>
</tr>
</tbody>
</table>

![Graph showing throughput vs adaptation interval]
Guideline: Long-term smothered operation produces best average performance (Cont’d)

Large Sampling periods do not lead to more accurate rate estimations
Robust Rate Adaptation Algorithm Goals

- Improve performance
- Manage varying dynamics
- Easy to implement
- Fit the IEEE 802.11 Standard
RRAA Design

- Short-term loss ratio
  - Assess
  - Adapt
- Adaptive RTS
  - Leverage
  - Filter
RRAA Design - Modules
RRAA-BASIC – Loss Estimation

```plaintext
1  R=highest_rate;
2  counter=ewnd(R);
3  while true do
4    rcv_tx_status(last_frame);
5    P = update_loss_ratio();
6      if( counter == 0 )
7    if    (P > PMTL) then R = next_lower_rate();
8      elseif (P < PORI) then R = next_high_rate();
9    counter = ewnd(R);
10   send(next_frame,R);
11  counter--;
```

\[
P = \frac{\#\text{lost_frames}}{\#\text{transmitted_frames}}
\]
RRAA-BASIC – Rate Change

1. \( R = \text{highest\_rate}; \)
2. \( \text{counter} = \text{ewnd}(R); \)
3. \( \text{while true do} \)
4. \( \quad \text{rcv\_tx\_status(last\_frame);} \)
5. \( \quad P = \text{update\_loss\_ratio}(); \)
6. \( \quad \text{if} (\_\text{counter} == 0) \)
7. \( \quad \quad \text{if} (P > \text{PMTL}) \text{ then } R = \text{next\_lower\_rate}(); \)
8. \( \quad \quad \text{elseif} (P < \text{PORI}) \text{ then } R = \text{next\_high\_rate}(); \)
9. \( \quad \quad \text{counter} = \text{ewnd}(R); \)
10. \( \quad \text{send(next\_frame,R);} \)
11. \( \quad \text{counter}--; \)
12. \( \)

\[
P^*(R) = 1 - \frac{\text{Throughput}(R_-)}{\text{Throughput}(R)} = 1 - \frac{\text{tx\_time}(R)}{\text{tx\_time}(R_-)}
\]
Estimation Window Size

<table>
<thead>
<tr>
<th>Rate (Mbps)</th>
<th>Critical Loss Ratio (%)</th>
<th>$P_{ORI}$</th>
<th>$P_{MTL}$</th>
<th>ewnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>N/A</td>
<td>50.00</td>
<td>N/A</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>31.45</td>
<td>14.34</td>
<td>39.32</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>22.94</td>
<td>18.61</td>
<td>28.68</td>
<td>20</td>
</tr>
<tr>
<td>18</td>
<td>29.78</td>
<td>13.25</td>
<td>37.22</td>
<td>20</td>
</tr>
<tr>
<td>24</td>
<td>21.20</td>
<td>16.81</td>
<td>26.50</td>
<td>40</td>
</tr>
<tr>
<td>36</td>
<td>26.90</td>
<td>11.50</td>
<td>33.63</td>
<td>40</td>
</tr>
<tr>
<td>48</td>
<td>18.40</td>
<td>4.70</td>
<td>23.00</td>
<td>40</td>
</tr>
<tr>
<td>54</td>
<td>7.52</td>
<td>N/A</td>
<td>9.40</td>
<td>40</td>
</tr>
</tbody>
</table>

1. \( R = \text{highest\_rate}; \)
2. counter = ewnd(R);
3. while true do
4. \( \text{rcv\_tx\_status(last\_frame)}; \)
5. \( P = \text{update\_loss\_ratio}(); \)
6. if (counter == 0)
7. \( \text{if} \quad (P > P_{MTL}) \text{ then } R = \text{next\_lower\_rate}(); \)
8. \( \text{elseif} \quad (P < P_{ORI}) \text{ then } R = \text{next\_high\_rate}(); \)
9. \( \text{counter} = \text{ewnd}(R); \)
10. send(next\_frame,R);
11. counter--;
Miscellaneous Issues

- **Idle Stations**
  - Refresh the window

- **Multiple Active Stations**
  - More stations, shorter estimation windows

- **Variable Packet Size**
  - Packet groupings
Adaptive RTS Filter - Design

- Suppressing hidden-station-induced loss options
  - Turn RTS on (every frame)
    • Large Overhead
  - RTS on frame loss / RTS off frame success
    • RTS oscillations
Adaptive RTS Filter - Scheme

1. `RTSWnd = 0;`
2. `RTScounter = 0;`
3. while `true` do
4.   `rcv_tx_status(last_frame);`
5.   if (!RTSOn and !Success) then
6.     `RTSWnd++;`
7.     `RTScounter = RTSWnd;`
8.   elseif (RTSOn xor Success) then
9.     `RTSWnd = RTSWnd/2;`
10.    `RTScounter = RTSWnd;`
11.   if (RTScounter > 0) then
12.     `TurnOnRTS(next_frame);`
13.    `RTScounter--;`

<table>
<thead>
<tr>
<th>RTSWnd</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTScounter</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Loss</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Frame transmitted

DATA
RTS
Integrating RRAA-BASIC & A-RTS

- **RRAA-BASIC**
  - Channel Fluctuations
- **A-RTS**
  - Hidden terminals

```python
1    while true do
2       rcv_tx_status(last_frame);
3       A-RTS();
4       if(!RTSFail) then
5           RRAA_BASIC();
6       if(RTSWnd > 3) then
7           fix_re_tx_rate();
```
Performance Evaluation – Static Clients

- Other Algorithms
  - ARF
  - AARF
  - SampleRate
- UDP and TCP
- 802.11 a/ b Channels
Performance Evaluation – Static Clients

(a) UDP goodput in 802.11a.
(b) TCP throughput in 802.11a.
Performance Evaluation – Static Client
Performance Evaluation– Mobile Clients

![Graph showing performance evaluation for different mobile clients.](image)
Performance Evaluation – Hidden Stations

![Graph showing rate distribution and UDP goodput for different rates and protocols.](image)
Performance Evaluation– Field Trials

![Graph showing average TCP throughput (Mbps) for Static clients and Mobile client. The x-axis represents different client types: ARF, AARF, SampleRate, RRAA. The y-axis represents the throughput in Mbps, ranging from 0 to 5. The graph compares the performance of static clients and mobile clients for each client type.](image-url)
Conclusions

- Rate Adaptation Algorithms
  - Differentiate between loss behaviors
  - Adapt to realistic scenarios
  - Handle hidden stations
PROBLEMS

- Number of Work Stations
- RTS failure/ Data transmission Failure
Work Cited

- S. H.Y, H. Yang, S. Lu and V.Bharghavan. Robust Rate Adaptation in 802.11 Networks
- Chart (Slide 15) - Figure 3.5 from J.Bicket. Bit-rate Selection in Wireless Networks. MIT Master’s Thesis, 2005