Experimental Evaluation of TCP Performance in Multi-rate 802.11 WLANs
Naeem Khademi, 07.06.2012 11:30

Rate Adaptation (bit-rate selection)

- **Rate Adaptation (RA):** predicts changes in the channel condition (random?) and reacts to them by changing the bit-rate to maximize the system throughput
  - LL-based RAs (e.g. “frame loss” metric) vs. PHY-based RAs (e.g. “SINR” metric)

- Several *Modulation and Coding Schemes* (MCS) in 802.11 std.
  - 1, 2, 5.5, 11 Mbps *(802.11b)*
  - 6, 9, 12, 18, 24, 36, 48, 54 Mbps *(802.11a/g)*
  - 32 MCS with up to 1~4 spatial streams (8 MCS for each), 20/40 MHz, 400/800 ns GI gives 6.5~600 Mbps – e.g. more than 128 different bit-rates *(802.11n)*
    - *(still open for research!)*
  - 1.73 Gbps per STA? *(802.11ac draft)*
Rate Adaptation (bit-rate selection) (#2)

- **A good RA mechanism? rapidly responds** to short-term channel quality degradations and **quickly recovers** when the situation has improved => highest PHY rate when noise is negligible!

- **802.11 RA**: vendor-specific
  - **Atheros (qualcomm)**: ath5k, ath9k, madwifi => *AMRR, ONOE, SampleRate* and *Minstrel*
  - **Broadcom**: b43 => ported *Minstrel*
  - **Intel**: iwlan => (?)

Also shown in our previous work: [N. Khademi, M. Welzl, and R. Lo Cigno, "On the uplink performance of TCP in multi-rate 802.11 WLANs", NETWORKING 2011]

PHY-LL/MAC/Transport RA*/DCF/TCP (cross-layer interactions)

- **Low-noise environments:**
  - Should use the *highest PHY rate* when noise is negligible
  - Can’t simply disable the RA due to the *random nature* of the channel

- **Sub-optimal TCP performance** when a LL-based RA is unable to distinguish the frame collisions from the noise – depending on the RA’s design (most RAs do this to some extent...)
  - not the case for “downlink-only” traffic scenario
  - [S. Choi et al., “Cross-layer analysis of rate adaptation, DCF and TCP in multi-rate WLANs,” INFOCOM 2007]

Also shown in our previous work: [N. Khademi, M. Welzl, and R. Lo Cigno, "On the uplink performance of TCP in multi-rate 802.11 WLANs", NETWORKING 2011]
RA* / DCF / TCP downlink-only traffic

\[ P_{\text{sta}} = 1 - (1 - \tau_{\text{sta}})^{n-1} \]

Conditional to attempt rate when \( q(\text{sta}) \neq 0 \)

Collision probability

DCF: AP's channel access ratio = (2:n+1)

Collision type(s) = ACK-ACK, DATA-ACK

\( N_{\text{active}} < 2 - 3 ; n < 100 \)

[J. Choi et al., INFOCOM 2007]

RA* / DCF / TCP downlink-only traffic (#2)

\[ P_{\text{sta}} = 1 - (1 - \tau_{\text{sta}})^{n-1} \]

M/M/1 birth-date chain in

[J. Choi et al., INFOCOM 2007]

\[ \tau_{\text{sta}} = \frac{\tau_{\text{op}} (1 - p_{\text{op}})}{n(1 - P_{\text{sta}})} \]

This scenario is too simplistic!

uplink traffic?! mixed-mode traffic?!
RA*/DCF/uplink-TCP

(1) Downlink TCP: $\tau_{sta} = \frac{\tau_{ap}(1 - p_{ap})}{n(1 - p_{sta})}$

(2) Uplink TCP: $\tau_{sta} = \frac{n\tau_{ap}(1 - p_{ap})}{(n - 1)(1 - p_{sta})}$

$N_{active} = n$;

(1), (2): $O(1/n)$

Collision type(s) = DATA-DATA (n:n+1), DATA-ACK

ns-2: collisions of TCP/ACK packets in two scenarios: download-only (DL-) and upload-only (UP-) – 54 Mbps fixed rate, 8 nodes, 100 ms wired link delay

<table>
<thead>
<tr>
<th>MAC protocol</th>
<th>IEEE 802.11g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet size</td>
<td>1500 bytes</td>
</tr>
<tr>
<td>Application</td>
<td>FTP</td>
</tr>
<tr>
<td>TCP congestion control</td>
<td>SACK (low kernel scale 3,16,3)</td>
</tr>
<tr>
<td>Propagation model</td>
<td>Rician fading</td>
</tr>
<tr>
<td>Medium access</td>
<td>DCF/DHCP (30 nodes, same)</td>
</tr>
<tr>
<td>CWMin</td>
<td>150</td>
</tr>
<tr>
<td>CWMax</td>
<td>1024</td>
</tr>
<tr>
<td>SIFS</td>
<td>100µs</td>
</tr>
<tr>
<td>DIFS</td>
<td>512µs</td>
</tr>
<tr>
<td>DIFS time</td>
<td>9µs</td>
</tr>
<tr>
<td>RTS/CTS timer</td>
<td>Enabled</td>
</tr>
<tr>
<td>Node-to-AP distance</td>
<td>15m</td>
</tr>
<tr>
<td>Number of clients</td>
<td>8</td>
</tr>
<tr>
<td>Optional RA mechanism</td>
<td>AARF exists</td>
</tr>
</tbody>
</table>
RA(AARF*)/DCF/TCP (ns-2 simulations)

ns-2: aggregate interface queue occupancy:
(a) Download; (b) Upload; (c) AARF-upload

*AARF: primarily used in WaveLan II devices, is an extension of the Auto-Rate Fallback (ARF) with a BEB mechanism. It decreases the bit-rate over a fixed number of consecutive losses (e.g. \( \theta_d = 2 \))

A practical question about madwifi

- Many RA mechanisms have been proposed in the literature but mostly not being implemented in Wi-Fi devices
- It’s practical to study the currently deployed RA mechanisms
  - Vendor-specific
  - Proprietary
- madwifi (parent of ath5k/ath9k)
  - For Atheros-based transmitter chipsets
  - Open-source driver on the top of ath HAL
  - RA suite including: AMRR, ONOE, SampleRate, Minstrel
  - LL-based
Adaptive Multi-Rate Retry (AMRR)

- Initially designed to work on the top of AR5212 HAL but works on variety of Atheros chipsets
  - HAL allows to create unbounded FIFO queues containing tx descriptors to schedule the packet transmissions
  - Descriptor: tx status field, pointer to the outgoing packet, the size of data and four pairs of rate/retry count fields ([r0, c0], [r1, c1], [r2, c2], [r3, c3]) (a.k.a retry chain)

```c
struct amrr_node {
  /* AMRR statistics for this node */
  u_int amn_tx_try0_cnt;
  u_int amn_tx_try1_cnt;
  u_int amn_tx_try2_cnt;
  u_int amn_tx_try3_cnt;
  u_int amn_tx_failure_cnt;

  /* AMRR algorithm state for this node */
  u_int amn_success_threshold;
  u_int amn_success;
  u_int amn_recovery;

  /* rate index etc */
  u_int8_t amn_tx_rtx0;     /* series 0 rate index */
  u_int8_t amn_tx_rate0;    /* series 0 h/w rate */
  u_int8_t amn_tx_rate1;    /* series 1 h/w rate */
  u_int8_t amn_tx_rate2;    /* series 2 h/w rate */
  u_int8_t amn_tx_rate3;    /* series 3 h/w rate */
  u_int8_t amn_tx_rate0_sp; /* series 0 short preamble h/w rate */
  u_int8_t amn_tx_rate1_sp; /* series 1 short preamble h/w rate */
  u_int8_t amn_tx_rate2_sp; /* series 2 short preamble h/w rate */
  u_int8_t amn_tx_rate3_sp; /* series 3 short preamble h/w rate */
  u_int amn_tx_try0;        /* series 0 try count */
}
```

- If (num_tx_frames > 10) && (tx_frames_err < 10%) r0 = r0 + 1;
- else if (num_tx_frames > 10) && (tx_frames_err > 33%) r0 = r0 - 1;
- else continue;
SampleRate

- Tries to maximize the **throughput** by sending packets at the bit-rate that has the **smallest average “packet transmission time”** including the time required to recover from losses as measured by recent samples
  

- **Sampling:** Periodically (on every 10th transmission) sends packets at a **random bit-rate** that might perform better than the current one

- Stops probing at a bit-rate with 4 successive failures

- In **madwifi:** EWMA with a smoothing rate of 95% (instead of the original 10 sec window)

SampleRate (#2)

- **Calculating the transmission time:**

\[
\text{tx\_time}(b, r, n) = \text{DIFS} + \text{backoff}(r) + (r + 1) \times (\text{SIFS} + \text{ACK} + \text{header} + (n \times 8/b))
\]

packet length in bytes  number of retries

bit-rate

A data frame transmission

An example of SampleRate’s statistics table in /proc
Minstrel

- Initially implemented by D. Smithies, has now become the default RA mechanism in madwifi (and all other Atheros-based drivers!), also has been ported to other drivers that use mac80211 framework
  - Takes into account the successfullness of packet transmission
  - EWMA-based (smoothing factor of 75%)
  - Rate selection periods of 100 ms to populate the retry chain (tunable in the driver)
  - Sampling frames a.k.a “look-around frames” (10% by default)

\[ p(b) = p_{\text{success}}(b) \cdot \frac{tx \_ in \_ bits}{time \_ elapsed} \]
\[ p_{\text{success}}(b) = \frac{tx \_ success \_ num}{tx \_ num} \]

Minstrel (#2)

- Minstrel’s retry chain:

<table>
<thead>
<tr>
<th>try</th>
<th>“look-around” rate</th>
<th>normal rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>random &lt; best</td>
<td>random &gt; best</td>
</tr>
<tr>
<td>1</td>
<td>best throughput</td>
<td>random rate</td>
</tr>
<tr>
<td>2</td>
<td>random rate</td>
<td>best throughput</td>
</tr>
<tr>
<td>3</td>
<td>best probability</td>
<td>best probability</td>
</tr>
<tr>
<td>4</td>
<td>lowest base rate</td>
<td>lowest base rate</td>
</tr>
</tbody>
</table>

An example of Minstrel’s statistics table in /proc
Experimental Setup

- Two different test-beds:
  - **Emulab** (Flux/University of Utah)
  - **NDlab** (ND/IFI/UiO)

- 802.11b/g, 100 sec *iperf* TCP traffic,
  50 runs with 95% confidence interval,
  MSS=1448, tcp_[snd/rcv]_buff=256 KB

Test-bed setup

<table>
<thead>
<tr>
<th>Test-bed</th>
<th>Emulab</th>
<th>NDlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>Pentium III 600 MHz</td>
<td>Dell OptiPlex GX280</td>
</tr>
<tr>
<td>Memory</td>
<td>256 MB</td>
<td>1 GB</td>
</tr>
<tr>
<td>802.11 device</td>
<td>D-Link DWL-AG540</td>
<td>D-Link DWL-T120</td>
</tr>
<tr>
<td>Chipset</td>
<td>AR5212</td>
<td>AR5001X</td>
</tr>
<tr>
<td>RX queue length</td>
<td>200 pkts</td>
<td></td>
</tr>
<tr>
<td>Driver (madwifi)</td>
<td>0/0/3.3, 0/0/4</td>
<td>0/0/4</td>
</tr>
<tr>
<td>OS</td>
<td>PC4, PC6</td>
<td>PC14</td>
</tr>
<tr>
<td>Linux kernel</td>
<td>2.6.18.6, 2.6.30.6</td>
<td>2.6.35.11</td>
</tr>
<tr>
<td>Node numbers</td>
<td>6-26</td>
<td>10</td>
</tr>
</tbody>
</table>

Statistics are gathered from madwifi kernel modules and tools, /proc and horst 802.11 analyser

Experimental Evaluations (TCP SACK)

Rate distribution of TCP data packets (8 nodes):
(a) AMRR; (b) SampleRate; (c) Minstrel
Experimental Evaluations (TCP SACK – Mixed Mode)

Mixed upload/download traffic using TCP SACK – 8 nodes

Experimental Evaluations (TCP SACK) (#2)

TCP SACK throughput – 8 nodes: (a) emulab; (b) NDlab
Experimental Evaluations (TCP CC)

Aggregate TCP throughput – 8 nodes

Experimental Evaluation (Contention Effect )

Aggregate throughput vs. contention level:
(a) AMRR; (b) SampleRate; (c) Minstrel
Why is minstrel performing better?

- The main differences between 
  \textit{SampleRate} and \textit{Minstrel}:
  – Metric to get the \textit{“best throughput rate”} (\textit{tx\_time} vs. \textit{successful\_data\_delivery})
  – Sampling method (sample only when the \textit{“best throughput rate”} fails!)

\textbf{SampleRate example:}
1500 bytes packets
\textit{best throughput rate} = 48 Mbps
sampling at 54 Mbps
\textit{perfect\_tx\_time(48)} * \textit{=} 672 \mu s
\textit{perfect\_tx\_time(54)} * \textit{=} 644 \mu s
\textit{tx\_time(54)} = 738 \mu s when failure with 4 retries

* These values are calculated based on EWMA used in the madwifi rate statistics table

Why is minstrel performing better? (#2)

- \textit{Minstrel}:
  – Sampling is distributed \textit{homogeneously} among different rates
    • Less sensitivity to a single probing failure at a certain rate
  – More sampling frames in each rate-selection process (100 ms)
    • Despite \textit{SampleRate} that samples on every 10th packet transmission
  – Incorporates the data delivery probability (potential throughput gained)
    • Despite \textit{SampleRate} which uses \textit{tx\_time} which is misleading for the rates with very close \textit{tx\_times}
Conclusive Remarks

- Cross-layer interactions of madwifi RA* / 802.11 DCF / TCP through simulations and real-life measurements in two different test-beds

- Uplink TCP performance can be affected by the choice of RA mechanism in infrastructure-based WLANs, even in low-noise environments

- Recommends Minstrel for deployment in 802.11 devices

Q&A