Improving Robustness, Throughput, Latency and Channel Awareness in Software Defined Radios

Nov 2011

Octavian Sarca, PhD
Leigh Chang
Redline Communication Inc.
Overview

- Common wireless communication systems performance targets:
  + Throughput
  + Latency
  + Link Reliability: availability, robustness against interference

- Extra SDR challenges
  + Cannot take advantage ASIC speeds
  + Be cognitive and adapt with the channel

- Is there a secret sauce that we can use?

- Yes! The **Automatic Repeat reQuest (ARQ)** mechanisms

- Counterintuitive because we all “know” ARQ increases latency significantly and decreases throughput. But is it true?
Common ARQ schemes

- **WiFi ARQ**
  - Compensates for varying signal strength but also collisions
  - Essential for good operation
  - Stop-and wait – no new packet sent until ACK received for current
  - Selective Repeat for 802.11n aggregated packets
  - Uses multiple retries on different modulations → high latency → typically delay sensitive services (e.g. voice) use a scaled down retry scheme
  - The retries may worsen congestion

- **TCP ARQ**
  - Get a single ACK for a number of sent of packets (TCP window)
  - Selective Repeat ARQ
  - Multiple retries
  - High latency because of ACK latency
  - Designed to guarantee delivery even under network congestion
  - Not designed to deal with random PHY packet loss
Common ARQ schemes

- 802.16d (WiMAX-d)
  - ARQ was considered non-important → not mandatory
  - ARQ used a TCP-like scheme → very high latency
  - TCP ARQ could kick in before 802.16 ARQ → a lot of wasted bandwidth
  - Nobody used ARQ → larger link margins

- LTE and 802.16e (WiMAX-e)
  - Lessons learned from 802.16d:
    - ARQ is important for link budget
    - ARQ feedback must be fast
  - Two-tier ARQ:
    - Hybrid ARA (HARQ)
      - Fast feedback
      - Same modulation
    - Standard, TCP-like ARQ
New ARQ scheme

- Immediate ARQ feedback
  - For DL, schedule ARQ feedback in the UL in the same frame
  - For UL, BS knows immediately anyway

- Selective Repeat
  - No wasted bandwidth

- Single retry
  - At most one extra frame

- Use lower modulation and/or coding to ensure delivery
  - Improves PER by >10^4
Retry Modulation: same or lower?

Lower modulation or coding $\rightarrow >10^4$ BER improvement
Is latency really an issue with ARQ?

- **YES,** if using multiple retries (WiFi scheme)
  - 10 retries means 10x the latency

- **YES,** if there is no explicit NACK (WiFi scheme)
  - Transmitter waits for time-out to infer packet did not arrive at destination

- **YES,** if not using fast feedback (TCP-like scheme)
  - Selective-repeat is nice but let’s get that feedback immediately

- **NO, if done properly,** e.g. in a TDD system
  - Send packet in frame K, get feedback in the same frame
  - If needed retry in frame K+1 than stop
  - Guarantee retry success by lower modulation and/or coding
Latency depends on implementation

<table>
<thead>
<tr>
<th>Measurement</th>
<th>WiMaxD</th>
<th>WiMaxE/LTE</th>
<th>Redline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average DL delay no ARQ</td>
<td>2 frames</td>
<td>2 frames</td>
<td>0.5 frames</td>
</tr>
<tr>
<td>Average DL delay with ARQ</td>
<td>15-20 frames</td>
<td>6 frames</td>
<td>1.5 frames</td>
</tr>
<tr>
<td>Average UL-RTP delay no ARQ</td>
<td>3 frames</td>
<td>3 frames</td>
<td>1.5 frames</td>
</tr>
<tr>
<td>Average UL-RTP delay with ARQ</td>
<td>15-20 frames</td>
<td>6 frames</td>
<td>2.5 frames</td>
</tr>
</tbody>
</table>

WiMax frame size: 5/10/20ms
LTE Frame size: 2ms
Redline frame size: 2...20ms
Can ARQ improve throughput?

- **YES**, if combined with adaptive modulation:
  - Single retry on a lower modulation that boosts packet error rate (PER)
  - Push principal modulation and coding as high as possible relying on retries to cover for increased PER
  - Switch to a lower modulation and coding rate only when the bandwidth loss to retries exceeds the bandwidth that would be lost if switching to a lower PHY modulation and coding, i.e. if:

  \[ T_1 \cdot (1 + \text{PER}_1 \cdot T_2) > T_3 \]

  Where:
  - \( T_1 \) = time it takes to send the packet at main modulation and coding
  - \( T_2 \) = time it takes to send the packet at retry modulation and coding
  - \( T_3 \) = time it takes to send the packet at a lower modulation and coding
Can ARQ improve throughput?

Yes, by allowing a higher PHY rate!
Can ARQ improve link reliability

- YES, because it can be more aware of the channel
  - PER on main modulation and coding can run > 1% and still maintain overall PER < 10^-6 due to the retry
  - can easily sense when link worsens way before it becomes critical

- YES, because it can eliminate random over-the-air errors
Do we really need such complex ARQ?

- Why not use a better channel coding?
  + For same reasons file systems have error recovery mechanisms despite having Error Correcting Codes (ECCs) at physical level

- TCP or higher layer protocol (for UDP) will ensure anyway reliable delivery
  + TCP will interpret packet loss as a sign of congestion and it will lower the throughput to reduce congestion → very low throughput
  + Many UDP-based protocols actually brake when placed in high packet-loss networks

- Why not use adaptive modulation?
  + Proposed ARQ lowers modulation and/or coding only for retries
  + It is not worth lowering modulation for thousands of packets following a random error
  + However it is worth lowering modulation just to ensure that random error does not cause end-to-end packet loss
Questions?