rQUIC
Another QUIC + FEC approach

Mihail Zverev, Pablo Garrido,
Ramón Agüero, Özgü Alay, Josu Bilbao
What is rQUIC?

- FEC prevents retransmissions, enabling robust and low latency communications.
- At the beginning of QUIC there was an unsuccessful intent to implement FEC.
- Recently F. Michel et al. have developed and presented a QUIC + FEC implementation.
- In parallel, there was another QUIC + FEC development, led by Pablo Garrido, with a different approach, and different results.

rQUIC

- Which QUIC + FEC is better and in which cases? Is it worth merging?

https://github.com/pgOrtiz90/quic-go-fec
As stated in ‘Coding for QUIC’ document:

Work on rQUIC started as addition of FEC to QUIC in the most practical and efficient possible manner. The focus was the resulting implementation. Therefore, no ID was consulted prior to this work.

Encoding after encryption was chosen for two main reasons:
1) Easier implementation
2) Easier scaling to QUIC-NC

NC = Network Coding
## Network Coding and encryption

<table>
<thead>
<tr>
<th>QUIC connection</th>
<th>QUIC connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encode</td>
<td>Encrypt</td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Receive</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Recode</td>
</tr>
<tr>
<td>Encrypt</td>
<td></td>
</tr>
</tbody>
</table>
• rQUIC is based on quic-go (https://github.com/lucas-clemente/quic-go). The base code was taken after v0.7.0 release.

• Rather than testing all existing coding schemes, the work focused on coding strategy implementation, only using XOR to code.

• In NC terms, generation sizes of $n$ are protected by 1 coded packet. Coding rate is adaptive.

• 4 bytes long FEC header is added with the following fields:
  • Type: protected, unprotected and coded.
  • BlockID: in NC terms, generation ID.
  • Ratio: generation size.
  • Count: packet order in FEC block (generation).
Red borders show new or modified QUIC blocks.
Adaptive coding rate

Adaptive coding rate reduces overhead in the absence of losses.

The algorithm is based on steering *residual losses*, which are packets that need to be retransmitted due to FEC failing to recover.

\[
\epsilon_i = \frac{\text{retransmissions}}{\text{transmissions} - \text{retransmissions}}
\]

Given the period \(i\) of length \(T\), the residual loss is computed as:

The residual losses are then averaged over \(N\) periods:

\[
\bar{\epsilon} = \frac{1}{N} \sum_{i=1}^{N} \epsilon_i
\]

The algorithm:

\[
\text{if } \bar{\epsilon} < \gamma \text{ then } \quad r = r \cdot (1 - \delta)
\]

\[
\text{else } \quad r = r \cdot (1 + \delta)
\]

end if

\(\delta\) and \(\gamma\) can be seen as aggressiveness parameters of the algorithm.

\(\delta\) and \(\gamma\) are configurable and determine the tolerance to FEC recovery failure.

After analysis of the behavior under different network topologies, we choose

\[
T = 3 \cdot RTT \quad \delta = 0.33 \quad \gamma = 1\%
\]

Evolution of rQUIC’s adaptive FEC ratio over time, for different link loss rates with 0 (no loss), 1, 2, 3 and 5%. 
**Evaluation**

**SIMULATIONS**

- **rQUIC fairness check**
  - QUIC session coexisting with (1) rQUIC session and (2) another QUIC session (25ms, 20 Mbps).
  - rQUIC does not impair QUIC

- **Emulation scenario**

  - 25 ms, 20 Mbps (WiFi/LTE)
  - 100 ms, 10 Mbps (2G/3G)
  - 400 ms, 1.5 Mbps (Satellite)
  - Overhead

- **Measured output:**
  - **Completion ratio**
    \[ \xi = \frac{\text{Completion Time rQUIC}}{\text{Completion Time QUIC}} \]
  - **Overhead**
    \[ \text{Overhead} = \frac{\text{FEC packets}}{\text{Total packets}} \]

**PHYSICAL SETUP**

- **Bulk Transfer**
  - (20 MB for WiFi/LTE and 2G/3G, 5 MB for satellite)

- **HTTP/2 transfer**
  - (flickr.com)
    - (30 objects, 1.776 KiB)

- **Physical setup** (provided by Simula)
Summary

- rQUIC is another modification of QUIC with FEC, different from the known one (by F. Michel).
- Although tested with only 1 coding scheme, it significantly improves bulk transfer traffic.
- Transparent design (to QUIC) which eases new coding schemes integration.
- With this implementation it is easier to give the next step: QUIC with Network Coding.

Upcoming features:
- More coding schemes (‘light-weightest’ first)
- Base code update (inclusion of new quic-go features)
- Current code improvements (such as adaptation in slow start phase and out of order packets management)
- Multipath
- Network Coding
Thank you

Mihail Zverev  mzverev@ikerlan.es
Pablo Garrido  pgarrido@ikerlan.es
Ramón Agüero  ramon@tlmat.unican.es
Özgü Alay  ozgu@simula.no
Josu Bilbao  jbilbao@ikerlan.es