Coding for QUIC Reference Implementation

François Michel
July 26, 2019

UCLouvain, Louvain-la-Neuve, Belgium
• The draft changed a lot compared to its previous version
• Our homemade version at UCLouvain was frozen on an old version of mpquic-go (complicated to maintain).
• We decided to propose a new simple, small version of a QUIC implementation supporting the Coding For QUIC extension.
  • restarted from scratch, supports most of the features of coding-for-quic-03
  • the goal is to stay up-to-date with the quic-go upstream.
The source code is available here:
https://github.com/francoismichel/coding-for-quic-go
Currently only supports block codes: XOR and Reed-Solomon, in a modular way
Uses RECOVERED frames to avoid hiding the packet losses to the QUIC loss-detection mechanism
REPAIR frame, source symbol medatada and RECOVERED frame can be defined differently by each FEC Scheme
Adding a new block code

Currently, all block codes (XOR and RS) implement this interface

```
type BlockFECScheme interface {
  GetRepairSymbols(block *FECBlock, n uint) ([][]*BlockRepairSymbol, error)
  RecoverSymbols(block *FECBlock) ([][]*BlockSourceSymbol, error)
  CanRecoverSymbols(block *FECBlock) bool
}
```

If you want to implement a new block code, you just need to implement these three methods.
What to do next

- Adding a sliding-window FEC Scheme!
  - Wire the implem with the SWIF codec once it is done
- Add a proper test suite
- More clever symbols scheduling and use new applications above this implementation (An example is already present for HTTP/3+FEC)
Other interesting implems to look at

- rQUIC (see the presentation in < 5min !)
- We will present a new QUIC implem at SIGCOMM 2019, revisiting the way an extension such as FEC can be deployed and used, stay tuned!
Thank you very much! Questions?