Project Requirements

# Aims

QUIC is a well thought-out and robust protocol with a clear RFC to describe it. It’s likely that protocols developed in the future will adopt qualities from QUIC, so it’s important that we are able to parse and generate code from the QUIC RFC to support automatic code generation for future protocols. The codebase already supports code generation in Rust, but it is limited to one language. This project aims to extend the original codebase to parse QUIC-specific packet description structures and create a framework to allow for additional languages to be added more easily.

# Initial Goals

* Parse QUIC structures and generate internal representation code
* Alter the RFC parser to ensure that it can parse QUIC structure artwork
* Create a more modular way to parse RFC sections so that the code is more readable and easier to modify in the future
* Build a simpler framework for writing and organising formatted output code
* Write an output formatter to produce code in Python

# Stretch Goals

* Create custom error types for parsing to increase robustness
* Write an output formatter for Java
* Refactor the ASCII diagram parser code to be more modular
* Write code documentation

# Outcomes

* Speed up the codebase by removing unnecessary code
* Make the code more modular and therefore readable
* Potentially put forward the idea of using QUIC structures for all packet descriptions in the future. (Like Stephen’s augmented diagram models).

# Timeline

## December

* Finished with the parsing code
* Start work on the output formatter

## January

* Framework complete for output formatter
* MVP for Python code formatting

## February

* Work on stretch goals
* Mainly dissertation work