O que foi feito nas duas semanas foi:

1. O estudo do protocolo QUIC através da continuação da implementação(DQUIC):

* Estudei que um pacote QUIC é composto na verdade de vários frames QUIC que servem para a troca de parâmetros. Por exemplo: Um frame QUIC serve para avisar para o cliente ou servidor para aumentar o fluxo de dados máximo(caso o endpoint suporte o fluxo será alterado).
* Os dados deverão ser transformados em bytes para ser alocados no buffer e serem enviados por UDP(Descobri isso através da pesquisa do código MsQUIC, que cada frame possui uma lógica específica para codificação e decodificação). Obviamente a decodificação também será feita, mas ainda não sei como o programa irá identificar que tipo de quadro ele é para saber o código correto de decodificação(específico do quadro).
* A maior dificuldade fica por conta da criptografia, já que os dois se retroalimentam, com o protocolo QUIC enviando o pacote específico do protocolo TLS, a criptografia(AEAD) ocorrendo em seguida(praticamente igual ao tls 1.2), e quando a mesma é completa o protocolo QUIC segue com sua comunicação criptografada(1-RTT).
* Estudo da RFC 4492 e tentativa de implementação da SHA -256 para o TLS com a RFC 6234

1. Releitura de alguns artigos:

* O artigo de Kumar(uma de minhas principais referências no Testbed), mostra que o dispositivo usado para a análise de desempenho foi: Raspberry Pi 3 Model B. Provavelmente tenho que comprar tal dispositivo para agilizar a pesquisa.
* Não sei se o MsQUIC funcionaria no Sistema do Raspberry Pi, mas seria interessante tentar(pois ele é usado como principal biblioteca para a implementação do MQTT pelo EMQX).
* Não achei na internet alguma implementação no arduino ou ESP do protocolo QUIC(nem mesmo do QUIC com MQTT).
* Pelo que eu entendo do artigo de Kumar, ele implementou o MQTT por cima do protocolo QUIC por si próprio para retirar coisas desnecessárias do código(pois como o protocolo faz parte do projeto CROMIUM ele tem muitas funcionalidades extras para auxiliar no navegador).

1. Execução do MQTT com QUIC

* Existem 3 bibliotecas que o EMQX disponibiliza para a utilização inicial do MQTT com o QUIC. É importante lembrar que o mesmo é uma adaptação inicial do protocolo, não possuindo tantas features que um navegador com o QUIC integrado possui(melhor criptografia com e maior desempenho com algoritmos de compressão)
* O único broker conhecido por mim por enquanto é o EMQX. Kumar implementou um broker com o protocolo QUIC, mas ainda não usei pois eu não consegui entender completamente como o protocolo funciona a nível de código(alocação de memória, criptografia e máquina de estados)

Considerações importantes:

* Como o raspberry pi tem um desempenho considerável, o protocolo QUIC com o MQTT funcionando no meu computador e no computador do professor Ramon, o mesmo irá ocorrer tranquilamente no Raspberry pi, desde que o linux ou windows seja instalado.
* Em paralelo, eu estou usando como material de estudo uma implementação do protocolo QUIC.
* Algoritmos de criptografia estão implementados nas RFC’s. O SHA por exemplo: <https://datatracker.ietf.org/doc/html/rfc6234#section-4.1>

Trabalhos futuros:

* Fazer com que o protocolo QUIC com MQTT funcione no computador, para assim adaptar ao Raspberry pi.
* Depois que funcionar, adquirir o Raspberry Pi para fazer os testes.

Tutorial de instalação do .NET 8 no raspberry pi: <https://gist.github.com/ramonsmits/b15d97965bcfacc19920be2e84b49c4e>

valores do raspberry pi 4 e 5:

<https://www.amazon.com.br/Raspberry-Pi-Computer-Model-RAM/dp/B07TC2BK1X?source=ps-sl-shoppingads-lpcontext&ref_=fplfs&psc=1&smid=A3G4P6N8I4F88S>

<https://produto.mercadolivre.com.br/MLB-4425428278-raspberry-pi-5-8gb-novo-_JM?matt_tool=18956390&utm_source=google_shopping&utm_medium=organic>